Computer Performance Optimization

Systems—Applications—Processes
Computer Performance Optimization
Computer Performance Optimization

Systems—Applications—Processes
# Contents

1 **Introduction** ........................................ 1  
  1.1 Terminology ........................................ 1  
  1.2 Three Levels ....................................... 1  
  1.3 Structure of the Book .............................. 4  

2 **Performance Theory** ............................... 7  
  2.1 System Performance ............................... 7  
    2.1.1 Hardware Parameters ................. 8  
    2.1.2 CPU .................................... 12  
    2.1.3 Main Memory ............................ 17  
    2.1.4 Disks .................................. 21  
    2.1.5 I/O ................................... 23  
    2.1.6 Operating System Parameters ........ 25  
  2.2 Application Performance ....................... 26  
    2.2.1 Application Parameters ............... 26  
    2.2.2 Data Management Concepts ............ 28  
    2.2.3 Application Environment .......... 33  

3 **Performance Measurement** ....................... 35  
  3.1 General ........................................ 35  
  3.2 Measurement ................................... 35  
    3.2.1 Dynamic Data ......................... 35  
    3.2.2 Static Data ............................ 37  
  3.3 Preconditions ................................ 37  
    3.3.1 Examples from Performance Measurements ............ 39  
  3.4 Data Collection Proper ....................... 39  
    3.4.1 Test Data .............................. 39  
    3.4.2 Other Important Statistics and Indicators .... 40  
    3.4.3 Monitoring ............................ 42  
  3.5 Special Case Cloud Applications ............ 45  
    3.5.1 Introduction ............................ 45  
    3.5.2 Cloud Applications .................... 45  
    3.5.3 Procedure ............................. 46
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.3</td>
<td>Function Calls</td>
<td>74</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Program Structure</td>
<td>75</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Frequent Calls and Opens</td>
<td>75</td>
</tr>
<tr>
<td>5.2.6</td>
<td>File Sizes</td>
<td>75</td>
</tr>
<tr>
<td>5.2.7</td>
<td>File Handling</td>
<td>76</td>
</tr>
<tr>
<td>5.2.8</td>
<td>Sorts</td>
<td>76</td>
</tr>
<tr>
<td>5.2.9</td>
<td>Copying</td>
<td>76</td>
</tr>
<tr>
<td>5.2.10</td>
<td>GUI</td>
<td>76</td>
</tr>
<tr>
<td>5.2.11</td>
<td>Production Schedule</td>
<td>77</td>
</tr>
<tr>
<td>5.2.12</td>
<td>Training</td>
<td>77</td>
</tr>
<tr>
<td>5.3</td>
<td>Investments</td>
<td>77</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Utilisation of Disks</td>
<td>78</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Data Management</td>
<td>78</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Conclusions</td>
<td>79</td>
</tr>
<tr>
<td>6</td>
<td>Network Performance</td>
<td>83</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>83</td>
</tr>
<tr>
<td>6.2</td>
<td>Peculiarities</td>
<td>83</td>
</tr>
<tr>
<td>6.3</td>
<td>Evaluation</td>
<td>84</td>
</tr>
<tr>
<td>6.4</td>
<td>Tools</td>
<td>84</td>
</tr>
<tr>
<td>6.5</td>
<td>Networks</td>
<td>84</td>
</tr>
<tr>
<td>6.5.1</td>
<td>LAN</td>
<td>86</td>
</tr>
<tr>
<td>6.5.2</td>
<td>WLAN</td>
<td>87</td>
</tr>
<tr>
<td>7</td>
<td>Process Performance</td>
<td>101</td>
</tr>
<tr>
<td>7.1</td>
<td>Starting Position</td>
<td>101</td>
</tr>
<tr>
<td>7.2</td>
<td>Identifying Critical Business Processes</td>
<td>101</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Processes Relevant to Performance</td>
<td>101</td>
</tr>
<tr>
<td>7.3</td>
<td>Process Documentation</td>
<td>109</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Types of Documentation</td>
<td>109</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Integrity and Relevance</td>
<td>110</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Responsibilities</td>
<td>111</td>
</tr>
<tr>
<td>7.3.4</td>
<td>Versioning and Approval</td>
<td>111</td>
</tr>
<tr>
<td>7.4</td>
<td>Process Tests for New Applications</td>
<td>112</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Availability of Partner Systems</td>
<td>114</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Test Scripts and Test Data</td>
<td>114</td>
</tr>
<tr>
<td>7.4.3</td>
<td>Organisation of Tests</td>
<td>116</td>
</tr>
<tr>
<td>8</td>
<td>Best Practise Examples</td>
<td>119</td>
</tr>
<tr>
<td>8.1</td>
<td>Hewlett&amp;Packard: LoadRunner</td>
<td>119</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Operating Load</td>
<td>119</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Comprehensive Coverage of Technical Environments</td>
<td>120</td>
</tr>
<tr>
<td>8.1.3</td>
<td>Implementation</td>
<td>120</td>
</tr>
</tbody>
</table>
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>American National Standard Institute</td>
</tr>
<tr>
<td>ARP</td>
<td>Address Resolution Protocol</td>
</tr>
<tr>
<td>BPSK</td>
<td>Binary Phase Shift Keying</td>
</tr>
<tr>
<td>BSI</td>
<td>Bundesamt für Sicherheit in der Informationstechnik</td>
</tr>
<tr>
<td>CCK</td>
<td>Complementary Code Keying</td>
</tr>
<tr>
<td>CD</td>
<td>Compact Disk</td>
</tr>
<tr>
<td>CODASYL</td>
<td>Conference on Data Systems Languages</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CRM</td>
<td>Customer Relationship Management</td>
</tr>
<tr>
<td>CSMA</td>
<td>Carrier Sense Multiple Access</td>
</tr>
<tr>
<td>CSMA/CA</td>
<td>Carrier Sense Multiple Access with Collision Avoidance</td>
</tr>
<tr>
<td>CSMA/CD</td>
<td>Carrier Sense Multiple Access with Collision Detection</td>
</tr>
<tr>
<td>dBm</td>
<td>Decibel Milliwatt</td>
</tr>
<tr>
<td>DBMS</td>
<td>Data Base Management System</td>
</tr>
<tr>
<td>DSSS</td>
<td>Direct Sequence Spread Spectrum</td>
</tr>
<tr>
<td>EDP</td>
<td>Event Driven Process</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>FHSS</td>
<td>Frequency Hopping Spread Spectrum</td>
</tr>
<tr>
<td>GHz</td>
<td>Gigahertz</td>
</tr>
<tr>
<td>GUI</td>
<td>General User Interface</td>
</tr>
<tr>
<td>HR/DSSS</td>
<td>High Rate/Direct Sequence Spread Spectrum</td>
</tr>
<tr>
<td>Hz</td>
<td>Hertz</td>
</tr>
<tr>
<td>ID</td>
<td>Identifier</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronic Engineers</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISAM</td>
<td>Index Sequential Access Method</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial, Scientific, Medical</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>Kbit/s</td>
<td>Kilobits per second</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>Ldev</td>
<td>Logical Device</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>MBit/s</td>
<td>Megabits per second</td>
</tr>
<tr>
<td>MED</td>
<td>Main Expense Driver</td>
</tr>
<tr>
<td>MHz</td>
<td>Megahertz</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multiple Input Multiple Output</td>
</tr>
<tr>
<td>MPDU</td>
<td>MAC Protocol Data Units</td>
</tr>
<tr>
<td>MPL</td>
<td>Multi Programming Level</td>
</tr>
<tr>
<td>OFDM</td>
<td>Orthogonal Frequency Division Multiplexing</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection Model</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical Layer</td>
</tr>
<tr>
<td>QAM</td>
<td>Quadrature Amplitude Modulation</td>
</tr>
<tr>
<td>QPSK</td>
<td>Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Arrays of Independent Disks</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Data Base Management System</td>
</tr>
<tr>
<td>RPF</td>
<td>Relative Performance Factor</td>
</tr>
<tr>
<td>RTS/CTS</td>
<td>Request to Send/Clear to Send</td>
</tr>
<tr>
<td>SLO</td>
<td>Service Level Objective</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol</td>
</tr>
<tr>
<td>TP</td>
<td>Transaction Processing</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>XOR</td>
<td>eXclusive OR</td>
</tr>
</tbody>
</table>
1.1 Terminology

The subject of performance optimisation can be divided into three main parts of consideration:

- system performance,
- application performance and
- process performance.

All three areas are subjected to:

- theory,
- measurement,
- analysis and
- optimisation.

1.2 Three Levels

When talking about performance, people usually refer to system performance only—or to simplify it yet again: to the power of hardware, that is, processor and main memory. This is the reason why performance has been neglected during the last decades. At some stage, hardware became so cheap that optimising by programming techniques, for example, did not seem worthwhile, since manpower became more and more expensive. Hardware and hardware extensions were bought and as a result systems were running faster again, or otherwise they were configured so comfortably that performance problems just did not show up.

However, end-user experience spoke a different language. Negative perceptions of response times did not only play a psychological role but also affected throughput in daily business. However, the ratio between hardware investments to optimisation remained more or less constant over time. The reason for this is that generous hardware resources are exploited equally generously.
Only forty years ago, no one could afford memory allocations for blanks or binary zeros. Already at the variable declaration level and successively at the address level, every single byte had to be considered deliberately. Otherwise, major applications would never have taken off the ground. And finally, after the introduction of graphical user interfaces, C++ and Java and their derivatives structured programming in the classically sense was abandoned. Demands on ease of use, end-user queries, etc., played their contribution to resurrect old bottlenecks in new garments. In this way, the performance debate has become newsworthy again—and this time not only restricted to systems and hardware alone. Although according to Fig. 1.1, the three levels:

- system performance,
- application performance and
- process performance

are addressed one by one, the terminology in use initially refers to system and application performance only.

System performance includes (Fig. 1.2):

- hardware utilisation (memory, processor),
- configuration of system tables and
- I/Os

with all procedures and parameters relevant to system management.

Concerning application performance and its analysis, one has to take into account the interplay with system resources in detail, considering system resource calls, data management and I/Os, for example. Quite generally speaking, if you want to execute applications, you need systems to run them. Figure 1.3 shows the
overall context. Thus, in performance considerations, these two levels cannot be regarded as separate or just existing side by side.

Those elements playing the most important roles when analysing applications are as follows:
• program structure,
• data management philosophy and
• GUls.

At the top, process performance dictates the rest. Process performance does not refer to system processors, but to those business processes, which are supposed to be supported by appropriate applications in turn running on their respective systems. This means that in our context, those processes are not subject to the usual management consulting tools such as balanced scorecard, cycle time. The important aspects with respect to overall IT performance are grounded in the consideration that in conjunction with a general optimization effort, the necessity of certain processes and their required system support demanding more or less resources can be put into question:

• Which system support is really necessary?
• How is my throughput influenced by application performance?

On this basis, one has to decide eventually where to start with tuning first. This is the time when the cost-benefit question becomes really important.

In summary:

Performance problems are first realized at the process level when daily business suffers. At neuralgic points critical applications are identified. These in turn depend on the system on which they are running.

On the other hand, however, performance measurements begin at the system level and identify bottlenecks, which in turn relate to resource utilization by applications. Application analysis in the end challenges certain sub-processes.

The overall tuning package eventually includes a mix of measures at all three levels. Their prioritization for implementation depends on the expected economical benefit in an organization with respect to the costs necessary to do this.

1.3 Structure of the Book

Following this introduction, the next chapter deals with the overall aggregate of performance theory—separately for systems and applications. Concerning systems, the main components—operating system, CPU, main memory and I/O processes—will be discussed. Special emphasis in the context of applications theory is put on data management concepts.

The next following chapter outlines methods and possibilities for performance measurements. The difference between dynamic and static system data and other indicators will be explained. Besides the monitoring subject itself, performance problems could get special attention.

The chapter about analysis contains a number of graphical results extracted and consolidated from raw performance data. Conclusions from these insights are discussed in detail and in relation to each other.
Finally, potentials and possibilities for tuning to enhance performance will be presented. Again, initially these recommendations are made separately for systems and applications, but then mutual dependencies will be shown. Notably, the fact that different problems may represent an intangible mix will be taken into account, even when solutions without compromises seem remote.

A special chapter deals with the subject of network performance. Firstly, established network performance monitors are presented. Then, the performance of WLAN applications is treated.

A challenge of its own is the optimisation of business processes. In this book, we shall not continue to follow up on the usual consulting approaches from rationalisation projects, but will identify the possibilities presented by the optimisation of systems and IT applications. At the centre are the consequences for process diversity itself, on the one hand, and on the other hand, it is about identifying performance-sensitive processes.

At the end, we shall present four best practise examples of performance tools: HP LoadRunner, SQL Server, Unix commands and TuneUp.

The book is completed by a list of abbreviations, a subject index and a comprehensive checklist. The appendix contains a brief description of project management and controlling tools.
Here and in the subsequent chapters as well, we shall deal with systems and applications separately. In practise, however, both aspects are intertwined with each other, and changes in parameters on one level will, with near certainty, have impact on the other. But, there are at least two reasons for reducing the over all problems by separate considerations:

- Separate problem zones often are related to separate key challenges.
- Measures to solve problems can be specific to one level or another (without loosing sight of possible interactions between the two).

A general picture of performance complexities usually presents a problem mix and thus aggravates singling out of critical aspects individually. In this chapter, therefore, we shall look at the two levels—systems and applications—one after the other. Because of their distinct nature, the respective sections are structured differently.

2.1 System Performance

This section concerning system performance is again subdivided into

- hardware parameters,
- operating system parameters.

Again those are in turn intertwined, since both depend on each other. Both aspects constitute what is called system architecture and will in turn again be treated separately, where possible. The selection of operating system parameters available obviously depends on the hardware manufacturer. On the one hand, there exist a nearly infinite combination of operating adjustments with respect to a given hardware configuration of the same manufacturer—only constrained by the real existence of the applications to be supported. This brings us back to the already mentioned boundary conditions. On the other hand, the same version of an operating system can be deployed on different hardware configurations.

Let's look at the hardware first (Fig. 2.1).
2.1.1 Hardware Parameters

2.1.1.1 General Remarks

In actual fact, this caption should really read: system or hardware components. The overall system configuration itself can be subdivided into (Fig. 2.1. as 1.2):

- CPU
- main memory
- disk storage
- I/O channels.

Of course there are many more other elements belonging to the hardware domain, such as terminals, modems and communication components. They will not be treated separately here within the context of our performance considerations (such as configuring the cursor speed by mouse programs). The theory of system performance deals with those four elements mentioned above in this sequence. The importance of each component with respect to a specific performance situation depends on the type of application, the number of users and other factors. Therefore, no priorities should be discussed at this stage.

These four components have mutual influence on each other. This will be explained in the course of our considerations. Although the resources mentioned will initially be treated in isolation, their impact with regard to a specific
performance scenario depends—as already mentioned—on the applications themselves, which rely on these resources. To isolate a specific problem, one needs certain evidence to bring basic mechanisms into the open, which have influence on such a specific performance scenario with respect to the application in question.

Before attending to the resources and their specific features in detail, some general principles should be addressed. The following questions are part of the discussion:

- In which way can system performance be tested generally?
- When does measuring performance make sense?

Performance testing is a discipline, by which in the end influence can be exerted over people, processes and technology to avoid risks, which can arise when introducing new systems, upgrades or patch installations. In short, performance tests consist in creating a typical system load before installing new applications—to measure performance, analyse it and collect end user experience. The aim is thus to identify performance problems under production-like conditions and eliminate them beforehand. A well prepared performance test should be able to answer the following questions:

- Are response times satisfactory for end users?
- Can the application cope with the anticipated system load?
- Is the application capable to accommodate the number of transactions expected from the business cases?
- Will an application remain stable under the expected or even unexpected load scenarios?
- Will end users have a positive surprise, when going life, or rather not?

By answering these questions, performance testing helps to control the impact of changes and risks when introducing systems. In this sense, performance tests should comprise the following activities:

- emulation of dozens, hundreds or even thousands end users, which interact with the system,
- consistent repetitions of load sequences,
- measuring response times,
- measuring system components under load,
- analysis and final report.

Tests and analysis should take place before bottleneck situations occur. Otherwise, performance measurements would have to take place later on a running system, including later all relevant measures resulting from the analysis of performance data. This means that for example load distribution over a working day has to be estimated prior to applications going life. The same is true for the decision, which jobs to be launched at what times. Once throughput problems really occur after system introduction, the following sequence of actions makes sense (Fig. 2.2):
Initially measure CPU under load,
thereafter memory usage.

If CPU and memory show no problems, I/Os to disks should be investigated.

Besides preemptive performance measurements discussed so far measurements
on production systems should be executed at regular intervals and continuously—
especially then, when subjective suspicion about bad performance is reported: long
response times for dialogue applications and standard reports needing long execu-
tion times.

One problem, however, is the selection of a suitable yardstick. The subjective
sentiment of a user sitting in front of a mute screen waiting for the return of his
cursor—thus experiencing a pure response time problem—is normally not suffi-
cient to pass for the status of a measured quantity. Generally one would want to
examine—besides online functions or GUIs—additional batch runs, the execution
times of which can be quantified, to get the overall picture.

Performance tests thus present a tool, with which to find out, whether a system
update will lead to improvement or degradation. The conceptual design of such
tests in the preparation phase should not be underestimated. It depends on the
actual situation on site:

- number of users,
- expected throughput,
- existing hardware, etc.

In any case, it is important not to test isolated applications, but to either look at
simulated real conditions or systems as they are found, comprising a significant
number of system processes running in parallel.

Concerning individual users performance tests can reduce to tangible key
indicators:

- response times and
- dwell time.
These quantities have direct impact on work progress and thus on personal productivity supported by a suitable IT infrastructure. Business process steps accumulate purely operational time and waiting time for system responses or time for system non-availability. Waiting times are a direct consequence of the actual system utilisation in each process step. In this sense, performance tuning has a direct effect on productivity.

Against these demands stand the expectations of operations, emphasising a different direction. Its interest is orientated on maximising throughput on the job level. This means in reality high processor capacity utilisation. Both expectations—those of the end user and those of operations—have to be taken into consideration to arrive at tuning compromises. For this, certain bench marks are available which can be used as a case by case reference. These include not only technical data, but also estimates, which should help to avoid inefficient usage of system resources. Before taking first measures following a performance analysis, the following clarifications should be made:

- Is the general emphasis on throughput only?
- Is the performance of the overall system unsatisfactory?

Quantitatively these questions could be narrowed down to:
- transaction rates depending on response times,
- throughput rates depending on job execution times.

With respect to the user, there are the following key indicators:
- response times belonging to different transaction types,
- execution times for specific jobs that have been launched.

The performance of a complete system can be again be traced to its various components; but also bad production scheduling can be the real reason, when job control is unbalanced. When deploying measuring tools one has to take into account that they themselves will use up some resources.

An important key indicator when measuring is the so-called system time. It is the sum of the following items:
- controlling time sharing between several parallel processes,
- controlling I/Os,
- controlling swapping for main memory usage.

Experience shows the following benchmarks for a typical CPU:
- overall system time: 10–20 %,
- timesharing: 5–10 %,
- I/Os: 2–6 %,
- swapping: 1–2 %.

Figures 2.3 and 2.4 give an indication of the essential approach and sequences of a tuning process.
At the end of these introductory remarks, two more general observations: upgrading existing hardware is often not the panacea for performance bottlenecks, as will be shown further on. And performance problems grow exponentially with the data volume to be handled. After these basic considerations, the individual resources in question will now be examined separately.

2.1.2 CPU

CPU stands for central processing unit. In the context of performance, CPU investigations focus on power, that is, throughput through the processor. This power is measured in million instructions per second (mips). This, however, stands on paper only, which tells us something about the useable power and about overheads and task management without I/Os or waiting queue management for other resources not being available or about virtual memory management for that matter.
CPU power is a critical factor for those applications, which are strongly CPU bound, such as scientific and technical applications with long sequences of mathematical calculations.

One possible yardstick can be the “relative performance factor” (RPF) (see Table 2.1). This quantity gives some information about the power potential of a CPU with respect to production conditions as realistically as possible. For its determination, both online applications and batch processing are taken into account. For online, the transaction rate is taken as a measure, and for the latter, the number of jobs executed per unit time. These measures are weighed in accordance to their relative importance in a given application setting: 20 % batch, 80 % online for example. But a single number alone is not capable to provide a definitive assessment about overall CPU behaviour. Normally, the application landscape is too complex for that. Other factors of influence are given by the server architecture. All these considerations, however, do only make sense, when CPU performance and main memory capacity stand within some reasonable relationship to each other.

A CPU can be in the following states:

- user busy
- overhead management
- waiting
- idle.

“User busy” refers to the execution of application tasks; “overhead management” indicates that the CPU is busy with managing itself, while going through waiting queues or redistributing priorities for example. “Waiting” points to the fact that a required resource is not available, and “idle” means that there are no demands on the CPU at present. Overheads can be broken down as follows:

- memory management/paging,
- process interrupt control,
- cache management.

<table>
<thead>
<tr>
<th>No. CPUs</th>
<th>Increase CPU time (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5–10</td>
</tr>
<tr>
<td>4</td>
<td>10–15</td>
</tr>
<tr>
<td>6</td>
<td>15–20</td>
</tr>
<tr>
<td>8</td>
<td>20–30</td>
</tr>
<tr>
<td>10</td>
<td>25–35</td>
</tr>
<tr>
<td>12</td>
<td>30–40</td>
</tr>
<tr>
<td>15</td>
<td>35–45</td>
</tr>
</tbody>
</table>

Table 2.1 RPF determination

<table>
<thead>
<tr>
<th>CPU</th>
<th>RPF batch</th>
<th>RPF online</th>
<th>RPF weighted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2 Increase in management overheads in a multiple CPU environment (Siemens: BS2000/OSD performance-handbuch Nov. 2009)
These problems look entirely different in a multiprocessor environment (Table 2.2). This environment is constituted from central processing units on the one hand and I/O processors on the other. To optimally exploit such a configuration, applications have to be written accordingly. This includes allowing for parallel processing without mutual interferences.

Such configurations have to be distinguished from others, which utilise parallel processors for resilience purposes to guarantee continuous availability of systems (Fig. 2.5).

On the task level, a specific task can only be assigned to one particular processor and vice versa. This means of course that upgrading hardware by adding more processors has a limited impact to solve performance problems concerning response times. In the limit, only a quicker CPU assignment takes place without any other throughput improvement. At the same time, a significant acceleration of batch jobs can only be achieved by massive parallelism. Besides, one has to take care that by increasing the number of processors demand on synchronisation with main memory increases as well producing additional overheads. Table 2.3 shows as an example improvement potentials for the execution of independent tasks by adding more CPUs.

**Table 2.3** Improvement effect after adding CPUs (Siemens: BS2000/OSD performance-handbuch Nov. 2009)

<table>
<thead>
<tr>
<th>No. CPUs</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.7–1.9</td>
</tr>
<tr>
<td>4</td>
<td>3.3–3.6</td>
</tr>
<tr>
<td>6</td>
<td>4.6–5.2</td>
</tr>
<tr>
<td>8</td>
<td>5.8–6.6</td>
</tr>
<tr>
<td>10</td>
<td>6.8–7.8</td>
</tr>
<tr>
<td>12</td>
<td>7.8–8.8</td>
</tr>
<tr>
<td>15</td>
<td>8.8–9.9</td>
</tr>
</tbody>
</table>
Such upgrades generally mean that memory and disk drives will have to be adjusted upwardly as well.

Many manufacturers recommend reference levels for optimal utilisation of their CPUs. In the following, we shall concern ourselves with mono processors only. For online-oriented systems, CPU utilisation should not exceed 70% for main application modules. In excess of that value, management problems concerning waiting queues can occur with the consequence of longer wait times. Naturally, possible utilisation figures exceed these reference levels for a multiprocessor environment (Table 2.4).

To control CPU tasks, certain routines are employed, which manage and supervise. The following parameters are important for task management:

- activation,
- initialisation,
- multiprogramming level,
- priority,
- multiprogramming level (MPL) per category,
- resource utilisation (CPU), main memory, paging activity,
- system utilities (CPU time, number of I/Os),
- assignment of access rights for main memory usage,
- assignment of CPU resources,
- state “active, ready”,
- de-initialisation,
- resource usage at waiting times,
- state “active, not ready”,
- expulsion.

Since a CPU is used by many tasks at the same time, utilisation is an important criterion for waiting times. The average service time by the CPU is, however, small in comparison with input and output times. This means that response times are influenced mainly by I/O processes, especially when reading or writing on storage media. The 70% utilisation is determined solely by those. On top of this, some manufacturers permit manual priority allocations, such that tasks that had been allotted low priority initially can now jump waiting queues. 100% utilisation is an ideal value, which will never be obtained in practice.

Priorities refer to access to CPU resources, I/Os are not taken into account by this. Normally, priorities are assigned in such a way that online tasks get higher priority with respect to batches. System programs on the other hand have even

<table>
<thead>
<tr>
<th>Load (%)</th>
<th>No. CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>2</td>
</tr>
<tr>
<td>80</td>
<td>4</td>
</tr>
<tr>
<td>85</td>
<td>6</td>
</tr>
<tr>
<td>90</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 2.4 Load as a function of the number of CPUs (Siemens: BS2000/OSD performance-handbuch Nov. 2009)
higher priorities than all other online functions. Within program processing, the system itself calculates priorities according to a particular algorithm, taking into account waiting queue positions, swap rate and other parameters. Allocation of an external fixed priority is possible, should, however, be handled with care, since this priority will not be adjusted dynamically thereafter and could lead to the expulsion of other processes. In this way, a few users, which deploy large programs, can be assigned low priorities against many users having to attend to only small tasks. At short notice, urgent business function could be assigned higher priorities as well as memory intensive programs blocking everything else, to liberate the system.

When talking about high CPU utilisation, the figure is around 90%. This does not necessarily mean a bottleneck situation. At high utilisation, however, it is important to watch the efficiency of CPU operations. The following measures should be observed:

- checking the level of the I/O rate,
- checking the level of the paging rate.

One measure for the efficiency of a given CPU at high utilisation is the ratio of waiting time against CPU/hardware service time.

When longer waiting times occur, this may be due to the following causes:

- non-optimal adjustment of system parameters,
- the overall CPU time requirements for all tasks are too high. An analysis of waiting queues could lead to the origins of this situation.

An important role concerning CPU management is played by the handling of interrupts. Obviously, swapping means the interruption of executing code. By this, a program or a part of a program is stopped, the code segment swapped out of main memory, and the task is directed into a waiting queue, from where it will gain access to resources later according to its priority (normally position in queue). Interrupts can have the following causes:

- priority-triggered resource access by the operating system,
- termination of a self-contained part of a program,
- technical error (“exception”),
- a longer pause when addressing a program page,
- I/Os,
- lacking data,
- status “idle” with time-out.

But even at low CPU utilisation levels, performance problems can occur. These can have the following reasons:

- I/O conflicts,
- high access rate to storage media by many users at the same time (central data files),
- paging rate too high,
- bottlenecks due to server tasks,
• inefficient parallel processing for certain programs,
• CPU time for the execution of certain programs,
• too many system calls,
• system self management,
• too many library routines,
• too many I/O routines,
• too many users in parallel,
• multiprocessor management,
• unfavourable CPU clock cycle per process,
• processor deadlock,
• sequence of machine instructions.

If performance problems continue to persist, even when CPU is “idle”, these problems are to be looked for elsewhere. By upgrading the CPU nothing will be gained.

2.1.3 Main Memory

One significant characteristic of main memory can be found in its nominal size, that is, the number of bytes available for central processing and its temporary storage for this purpose. The absolute size of main memory with respect to a performance scenario can be

• just right,
• too big or
• too small.

This sounds trivial. But the following has to be taken into consideration: an oversized memory, often recommended to compensate for bottlenecks, can lead to secondary CPU problems, when processing power does not follow in step, or will expose the limits of I/O channels. The reason for this is the number of possible tasks, which can be loaded and will exceed the capabilities of these two other resources in such a way that waiting queues will be generated.

Of course, an undersized main memory is the most obvious problem. Main memory has to deal with (Fig. 2.6).

• Program segments,
• data,
• cache partitions and
• parts of the operating system.

Normally, some part of main memory will constantly be occupied by some basic functions of the operating system: base memory as a fixed portion of the overall storage space, 10 % for example. Other portions of the operating system will be loaded later, when the need arises. An important portion will be needed for
user processes. This is the reason, why special care has to be taken, when updating the operating system with a new version, which may include additional functions. Reconfiguring has to take this into account.

Cache partitions are needed for storage segments containing data, which are addressed repeatedly. These partitions can be configured but also be assigned dynamically. The latter does not present a particular problem for the cache, even when bottlenecks may lead to restrictions concerning the cache portion and part of it is outsourced to a disk. Performance then degrades by the fact that some cache data now have to be read directly from the disk. On the other hand, intensive cache usage leads to interrupts and address mapping. This in turn generates additional CPU overheads.

Data occupancy of main memory has the following sources:

- user data,
- temporary addressing tables,
- special segments,
- system buffers.

User data occupy the address space, which has been assigned to a specific user for sorts for example. Special segments are used for messaging or communication buffers and system buffers for certain system tasks. The rest is available for program segments or for paging.

An important factor when judging main memory performance is the paging rate. Paging rate is defined by the number of instances per unit time, when actually memory-resident code pages being used for the execution of a program by the
CPU are temporarily removed, replaced by others and the original pages later again be reloaded. This sequence is called swapping.

There are some algorithms identifying those main memory entries, which are earmarked for swapping. Normally, the resource waiting queues are polled by each system cycle within the frequency of the system clock to find out which segments have not been addressed during the previous cycle. Those will be flagged as overlay candidates. During the next cycle, the overlay candidates are again checked and possibly swapped, if memory space is needed. If there are many overlay candidates and long dwell times for certain segments, which could be swapped without process interruption, only marginal memory management overheads will occur.

High paging rates can be critical for an efficient use of the main memory resource (Fig. 2.7). Reasons for this are as follows:

- too many processes,
- programs with a too large size.

Therefore, manufacturers release recommended maximum values. Memory management for a smoothly performing CPU should not account for more than a few percentage points. Otherwise paging rates, which are too high, will lead to bad response times, because CPU resources will be used for that. Background: in both cases mentioned above, the CPU first tries to reorganise main memory occupation by creating a concatenated address space (generating considerable overheads). If this does not work, the CPU gives up, that is, processes of low priority will be interrupted, swapping takes place into virtual memory or into the paging areas on the disks. Consequences are as follows:

![Diagram of memory management](image)

**Fig. 2.7** Paging
• lost CPU resources for memory management (increase in subjective response times),
• interrupted user processes (increase in subjective response times).

In case these problems become chronically, there is no way around an upgrade of main memory. One reason for this is that external swapping areas have to be specially configured. If they are too small, paging becomes a secondary bottleneck with additional I/O waiting times and corresponding memory management.

To avoid main memory bottlenecks, it makes sense to regularly and prophylactically take measurements with a suitable monitor. The following values should be captured:

• the number of pageable pages (size of main memory minus the number of pages in permanent residence necessary for system purposes),
• the number of globally used pages (number of pageable pages minus the number of freely usable pages).

Similarly as for the CPU, there are reference values for main memory. Usage of up to 75 % is supposed to be non-critical. By usage of continuously more than 90 % an upgrade is unavoidable. As is the case for CPUs, there is a task management with more or less similar triggers via control functions for main memory management including a page administration algorithm:

• activation
• deactivation
• forced deactivation
• expulsion.

Summarising, the following scenarios can happen:

Memory demand is higher than available. This creates a paging problem. This means that certain process sequences will be relocated temporarily onto decentralised storage media leading to noticeable performance degradation. Solutions can be gained either by hardware upgrades or application optimisation. On the organisational level, the number of users could theoretically be reduced, which is generally not regarded as being acceptable.

The opposite scenario is given, when sufficient memory is available. There is no third option between these two states. In this context, a specific program does not have to be loaded completely for the execution of certain functions. Program pages are loaded in accordance with their address sequences one by one leading in turn to swap-ins and swap-outs—thus paging processes. In any case, statistics show that most programs remain during 80 % of their execution time within only 20 % of their own code. Inactive processes are mainly caused by online applications—when users remain in front of their screens and do not enter anything, while the program itself remains open via the GUI. To optimise with respect to paging, the following measures can be applied:
• usage of common subroutines by several programs at the same time,
• code size optimisation,
• usage of common libraries,
• memory configuration with respect to the estimated number of users.

2.1.4 Disks

There are some simple facts that explain, why absolute disk space influences system performance: should totally available disk space be occupied by a theoretical 100 %, the system will grind to a halt. Files can no longer be created (even temporary or spool files) or even updated. But, of course, there are other problems, which will be noticed much earlier.

Every system coming close to 90 % disk space occupation will indicate performance problems regarding response times as well as anything as a smoothly running production environment. For problem-free operations, one needs free storage space for

• backups
• end of the month results or
• database reorganisation.

These are the reasons, why disk occupation should not exceed 75 %.

Normally users and administrators have only limited influence on the physical localisation of files on external storage media (volumes). One has to keep in mind that one physical volume may contain several logical volumes. Volumes are addressed via an identification number. Access times depend on the type of access technology, management of storage space, but also on the fact, whether a configuration uses several computers at the same time. Other impairments can be created due to parallel disk mirroring, especially in connection with the RAID concept. Redundant arrays of independent disks (RAID) function the other way round by grouping several physical volumes under one logical one. In this way, a corresponding data redundancy is created.

Manufacturers offer modules for device management with the following functionalities:

• monitoring of disk occupancy,
• reservation of disks,
• configuration,
• surveillance of access readiness,
• partitioning to optimise access to logically connected tables.

For all those reasons, it is important to identify and plan for storage locations prior to new installations, determine where which data should be placed and design an appropriate schema before effective initial loading.
Compromises need to be found between the objectives of throughput optimization of an individual process against the overall throughput for the whole disk. To measure a single process, a large test file is required. For the overall throughput of a single disk, this is more difficult. Actually, this can only be achieved by a simulation covering all processes. Thereafter, measures can be taken respecting the different interests of applications. Another compromise concerns the optimal usage of the storage medium with respect to throughput. In this case, one has to distinguish: large files having little storage efficiency should be placed on single defined disk, many small files on others.

Another aspect concerns the fragmentation of information (Fig. 2.8). It is indeed not the case that all data records belonging to a specific table should reside concatenated on a particular storage medium—not even initially. Files and data records are fractionated or split and distributed according to an internal algorithm over a disk together with corresponding pointers referring to respective subsequent addresses. During the life cycle of an application, this state of affairs degrades even more since files are updated by deletions or additions of data records. By this, gaps on the storage medium are created; then, these gaps are refilled. These new creations do not necessary belong to the same file nor are they in a specific logical sequence. In the end, search processes for queries will be prolonged and thus response times. From time to time, a defragmentation should be run. This results in gaps being filled up and associated data being brought closer to each other.
2.1.5 I/O

As already mentioned, there is a danger that a significant portion of CPU resources is required for the management of I/O waits. This is especially the case, when they are caused by process interruptions. Besides disk access themselves, important roles are played by the number and speed of communication channels and their controllers. Those are also required for terminals, modems and other communication hardware. A too small number of I/O controllers leads to communication bottlenecks. But on the other hand, a given CPU can only manage a limited number of controllers itself.

There are strong dependencies between the performance of external disk drives and the capacity of communication channels. Some manufacturers provide control mechanisms as part of their data management systems for input/output processes. The reaction time for input/output commands is generally one order of magnitude longer than that for CPU commands.

To begin with our interest shall centre on input/output to and from external storage media (Fig. 2.9). In programs, those calls are realised via intrinsics or I/O program calls (get, put, etc). SQL queries (read, write) are translated into equivalent execution instructions. Of importance for the execution of such or similar queries is the structure of the databases in question. The duration of search operations differs for sequential files, index sequential files or relational database management systems. Furthermore, execution times depend on the type of access. Reading access is obviously faster than updates, where a record has to be found first, then read and finally written to.

Once input/output is needed for the continuous execution of a program, this part of the code is put into waiting until the input/output sequence is completed. Thereafter the program continues to run. From the point of view of the CPU, the program rests in a waiting queue, which is controlled by a priority algorithm. This queue contains also parts of other competing programs, which means that the immediate execution of the program in question after the I/O is completed is not guaranteed, but will proceed only according to the actual internal priorities. In case the input/output happens via specific channels or special devices, there are again waiting queues, since other users demand access to those resources as well.

Input/output is thus a complex process, which succeeds in combination with various waiting queues and data buffers, into which the requested data records are read.

The so-called data cache helps to accelerate accesses (Fig. 2.10). Data cache is a buffer, which stores those data requested last, while the user or the program is occupied with different things. Now, is the last request for some reason repeated, this information is made available immediately from the cache. On top of these data, cache provides for a minimisation of disk accesses and the bundling of write operations (write behind method). Data caches can be configured. Data throughput is influenced by their size. Compromises have to be found between cache reservation in main memory, I/Os and CPU load for swapping purposes.
I/O time depends in turn on the technology of the system bus and the channel structure for the I/Os. In case of several externals disks, there have to be compromises between the total transfer rate across all disks and the individual transfer rate of single disks. Maximising the overall transfer rate impedes the individual transfer rates. To optimise, one can split applications in such a way that certain disks contain large files and others in turn many small files for less important I/O demands.

The optimum solution obviously is an even distribution of disk I/Os across all disks. Each disk should have its own controller. Shared controllers with several channels on the same bus should be avoided. To sum it up, transfer rates are thus dependend on a combination of disks, bus performance and controllers. Optimisation, however, does not necessarily succeed by making all components faster, since new bottlenecks can arise by a good performance of one component to the detriment of one of the others.
Performance data of disks do in turn depend on the positioning time (time to move the disk head from one data track to the next) in dependence of data contiguity. In case of high fragmentation and data distribution across many applications, frequent jumps of the head will be the consequence. This in turn leads to accelerations and decelerations. For this scenario, search operations themselves are much longer than reading or writing on their own. To improve this situation, the various possibilities for partitioning provided by the manufacturer should be put into use.

### 2.1.6 Operating System Parameters

Every operating system carries besides its capabilities to support applications also its proper functions, which influence and demand system resources. These comprise the following:

- program calls,
- certain commands,
- system control functions,
- configuration parameters,
- system tables,
- functional sub-systems: utilities.

Certain operating system commands can influence performance. Some of them are used for:
• communications control,
• job limits,
• logging,
• memory allocation,
• priority assignment.

Directly influencing performance are the following system control functions and configuration parameters:

• communication settings,
• job limits,
• logging,
• memory allocation,
• priority scheduling,
• settings relevant for data systems,
• spool parameters,
• time outs,
• cache sizes,
• system tables.

System tables need special attention. Generally, it is understood that table overflows will bring the system down. On the other hand, oversized tables will occupy too much main memory. Concerning utilities, one has to decide on the organisational level which circumstances will allow to provide certain users or applications with which utilities. These utilities may include:

• editors,
• system monitoring routines,
• debugging software,
• format handlers,
• compilers,
• database handlers.

Besides all these, it is important to check, whether new versions of operating systems contain features, which can positively be employed to improve performance. This should be an ongoing process.

2.2 Application Performance

2.2.1 Application Parameters

Although performance tuning on the basis of application parameters is usually time consuming and will normally not show visible results in the short run, it nevertheless is often without alternative to gain success in the medium and long term. These are some influencing factors:
• programming language,
• program architecture,
• number of subroutines and external calls,
• data management,
• I/O procedures,
• GUI or batch processing,
• size of code segments,
• communication processes with other applications.

If it can be chosen, the programming language plays an important role for the future performance of an application. This has to be put in relation to realisation times for a modular program architecture against the classical structured programs, which has lost in importance with the advent of modern programming languages. In a modular architecture, application modules are decoupled from each other although united under a common front end. This relieves main memory. Besides, certain modules from different parts of an application can be addressed by different programs or other program parts and in this way utilised in common—for example, error routines.

Although one should expect that every new release will deliver acceptable application performance, the results depend strongly on preparatory measures. They should take into account demands of the users and their performance expectations. The basic process sequences, which IT is going to support has to be taken into account. In the end, the following questions should be answered:

• Has the application architecture to be adapted to satisfy certain performance criteria?
• Will an upgrade of the IT infrastructure be required?
• Is performance degraded by GUI design?
• How will system load be affected by an expected increase of throughput?

Occasionally, some factors can be tuned with small effort—like I/O procedures or a switch from GUI to batch. Others demand higher investments. The transposition from one programming language to another or changes to the data model belong to the latter.

In addition to measured quantities, which are of importance to system performance itself, there are a number of characteristics with direct relevance to applications. Task specific demands on resources over time and at peak hours belong to these. They relate to the following:

• CPU time,
• I/O results including accesses to certain data tables and disk access statistics,
• waiting times,
• paging rate,
• channel transports including TCP/IP connections,
• possible communication events within a cluster configuration,
• accesses to main memory,
• response time statistics,
• cache hit rate,
• number of job executions.

This information can be gathered either by measuring monitors or partly from log files once the relevant transaction types have been flagged for recording. To record transactions, log files need to be adapted specifically with respect to two criteria:

• formatting
• selection.

Formatting means, that care has to be taken, that entries are disburdened from technical information. This requires that everything with relevance to transaction codes or data encryption, format declarations, etc. should be spared. Only readable text information should be deposited, which give an immediate indication to the analyst about what is happening with a particular transaction.

Only such transactions should be earmarked for recording that are of relevance to a later analysis. Negligible functions like “show date”, “return to main menu”, etc. should be suppressed.

Furthermore, averages and peak recordings over time for a typical working day are of interest. These will show time dependent bottlenecks, which could initially be decoupled by organisational measures without major technical efforts. Concerning the number of users behind resource utilisation, one has to differentiate between active ones and those, which are only logged in and remain idle otherwise.

2.2.2 Data Management Concepts

Data management is a vast area and presents a high potential for application tuning. All important computers are equipped with data management features in one way or another either as part of the operating system or as separate tools with possibilities to assign logical names to files and to specify a physical storage medium. The system or the tool organises the physical space and writes data to it or reads data from it according to the user instruction in question.

The commonly used expression “data base” in the widest sense means a collection of data records connected between each other with maximum coherence, stored in calculated redundancy and structured in such a way that different interests of many users can be satisfied (Fig. 2.11). In the following, different data management systems will be touched upon. It is unavoidable that historical structures, which do not represent the state of the art of file management today, will get their mention too (Figs. 2.12 and 2.13). Reason is that even of today in many organisations, traditional file handling is still in place, which has not been replaced by modern methods and often is a common cause for real performance problems.


2.2.2.1 Technical Conditions

In general, a data management system appears to a user as a software interface, which separates him from the operating system and at the same time from external...
storage hardware with respect to each access to a centrally controlled and integrated collection of data, which is shared between several applications and is called “data base”. The system provides tools to define the physical structure of the data base and its logical connections, to load data and modify them, to protect these data against incidental damage or unauthorised access and to cater for efficient data queries. A data management system is called “generalised”, if it is driven by a single user-oriented command language for all these different functionalities applicable to every new data base and independent from its internal organisation so to avoid the necessity to each time have to write new data management programs for every new data base.

But even in such a comfortable environment, there remain two unresolved problems:

- the optimisation of data mapping,
- the best possible usage of advanced database management functions.

Already in 1971, standards for database management system (DBMS) were developed by the CODASYL DataBase Task Group—at that time specifically for COBOL as main host language. They proposed a network structure to model relations between data records within a common data base (Fig. 2.14). Hierarchical relations were dealt with as being simply special cases in a chain of network files.

This widely used logical structure can be found in the cross linkage of data records, which organise items within that record hierarchically or in a tree structure. In most cases, most of these systems are capable of mapping a multilevel hierarchy, in which sub-items may consist in turn of other sub-items and so forth.

Purely logical networks are based upon the concept of interconnected files, where each one may be allocated an owner or master record and one or more member or slave records. The higher efficiency of networks against hierarchical structures is given by the fact that one single-record type may be associated with many others.
A typical DBMS comprises the following components (Fig. 2.15):

- data description compiler,
- data dictionary reports,
- data manipulation language,
- system modules,
- utility programs.

Soon after its introduction, the so-called relational model exerted great influence on data management as such (Fig. 2.16). Contrary to the models discussed so far, the relational model can be defined by simple rigorous mathematical concepts. Furthermore, its underlying structure permits simple queries, modifications, updates and restructuring of the database.
Altogether, the relational model is more user-oriented. It views a data base as a collection of n possible relations and homogeneous tables, in which each line corresponds to a single-data record, comprising n items appearing only once. When defining relations consistency and lack of redundancy can be guaranteed, once a set of formal rules is respected.

These five basic operations in relational algebra are as follows:

- selection
- projection
- product
- aggregation and
- difference.

The next logical step in the direction of new database technologies was the development of distributed databases. From a user point of view, its principles seem relatively straight forward and transparent: applications have local characteristics on a local platform. Within a communications network, database functions coordinate access to distributed databases and pretend to a user that the data are locally available. The knowledge about the real data distribution belongs to the DBMS alone. Changes in data distribution do not entail changes in programs.

To achieve this, the DBMS relies on a so-called reference architecture, comprising global and local schemata and the physical data assignment (Fig. 2.17).

### 2.2.2.2 Accesses

Independent from data management systems in operation, various operating systems offer data access modalities having impact on overall performance. Those comprise the following:
• buffered or non-buffered access,
• access with or without I/O wait.

In certain cases, both options permit the continuing execution of programs without I/O processes having necessarily been completed. Besides these, table sizes as such and the overall blocking philosophy play important roles, of course. Locking should be implemented on the lowest possible level (item), if feasible, never on table level and—if possible—also not on record level.

2.2.3 Application Environment

Application environment is more than just the GUI visible to the user and the corresponding software and data base, which supports his transactions—including batch. To understand the behaviour of the application properly, one has to look at the interplay between system components and the application itself. This means that it is always a matter of symbiosis between these two aspects. To describe the overall system, one has to take into account:

• the system nucleus: the operating system with its administration of the system resources on hand,
• job management,
• data management at operating system level,
• I/O control,
• other external resources (storage, modems, etc.).

Under certain circumstances, an application comprises also sub-systems and utilities like EXCEL© Sheets, directories and other elements. When setting up an application, one has to take care about additional variables like the following:
• application specific buffers and address space,
• authorisations,
• priorities,
• other operating parameters.

All these adjustments influence all over performance, especially the configuration of buffers and the address space allocation in main memory. This means that compromises concerning competing system requirements are inevitable. Figure 2.18 illustrates schematically the environment, in which applications operate.

For an application environment to go live, here are the essentials:

• the actual software release,
• the database,
• additional utilities,
• documentation,
• training,
• authorisations.
Chapter 3
Performance Measurement

3.1 General

To obtain transparency about a specific computer system, the first step would be to capture all possible analysable data. These data should relate to those variables, which have been discussed in the preceding theoretical chapter.

3.2 Measurement

As a consequence for measurements, one has to differentiate between dynamic data resulting from actual load scenarios on a system and static data relating to the configuration itself. Data are collected from different sources and then interpreted, thus leading to first conclusions, prior to an in-depth analysis.

3.2.1 Dynamic Data

Dynamic data are obtained under load as a function of time—generally under real or simulated production conditions. Other possible configurations include:

- benchmark configurations,
- an almost empty machine to test specific processes with the aim for later calibrations and scaling.

To capture dynamic performance data, the market or suppliers of operating systems offer special monitors, which collect these data and visualise them on screen or make them available in file reports. Depending on the problem under consideration and on the application environment, such monitors will run hours, days or even weeks either during production or in a simulation environment by using specifically developed test cases. The most important information to be collected is the following:
• application programs currently in use,
• distribution of user frequency over a lengthy period of time,
• account statistics,
• processor load,
• main memory usage,
• memory management,
• number of running processes,
• frequency of file accesses (open table),
• overheads and interrupts,
• waiting queues,
• system table usage,
• I/Os,
• swap rate.

These data can be obtained either as snapshots of specific performance situations or as a dynamic collection over longer time intervals. Monitors can be supplemented by log files for system events against the backdrop of the production schedule. Furthermore, there are monitors, which register system interrupts and their internal transaction times along with addresses in main memory. This detailed information has to be brought into relation with the corresponding log files manually for deeper analysis later. Additional tools provide for

• selection of main memory segments,
• selection of specific users,
• task-related CPU time,
• momentary execution status of programs or
• access to selected files.

Some tools deliver process state overviews. These can comprise

• user-related information,
• % actual CPU,
• % actual memory,
• actual external storage media occupancy,
• process status (active, idle...).

Time-dependent data should be stored. Additionally, to monitor data, the usual system queries should be recorded:

• current sessions,
• jobs,
• I/Os,
• corresponding system logs.

There is a grey area between dynamic and static data with respect to

• the frequency of standard function calls by users and
• the production schedule.
3.2.2 Static Data

This information can be obtained from system tables and the application environment. It comprises

- database schemata; data management system,
- cache size and address,
- temporary file space allocation;

and general configuration parameters:

- memory size
- nominal CPU power
- operating system version
- number of peripherals
- job, session parameters
- spool parameters
- size of system tables
- virtual memory
- system buffers;

as well as communication parameters:

- protocols
- ports
- nominal data transfer rates
- communication buffers
- disk space occupancy
- production schedules and
- inactive files since a given date.

Other useful documentation includes:

- account structure,
- the most important data files,
- files not having been opened for a lengthy period of time.

3.3 Preconditions

Measuring performance is a team work. For this reason, the following organisational preconditions have to be created prior to the attempt (Fig. 3.1):

- definition of duration and identifiable terminal point of the measurements,
- inclusion in the planning phase of all persons and organisational units concerned,
- tight cooperation with end users and all interested parties in an organisation,
- selection of performance testers early on in the project,
• securing cooperation between performance testers and developers to prepare specific test cases and provide test data,
• communicating collected user experience regarding performance in the past to testers and test management,
• establishing feedback processes to developers and analysts,
• define time window during and outside working hours for measurements.

To obtain meaningful results, one single measurement will certainly not suffice. Often, several measurement cycles with different focuses have to be implemented to obtain results, the evaluation of which make sense. In this way, distinct views on the overall system and application environment are generated. After implementing optimisation measures (Chap. 5), new measurements should of course be undertaken to find out whether those measures had any effect. Thus, performance measurements are part of a continuing improvement process. To this end, key performance indicators (KPIs) can be defined. They depend among others on the weight of individual problems on the production cycle.

There are basically two possibilities to drill down to the core of these problems:
• measuring in the production environment under normal system load or
• measuring in a test environment, which simulates the first.

If possible, measurements should be taken in the normal production environment. To create a realistic test environment, redundant systems have to be put into place and the total production has to be mapped including all the users. Normally, this means high costs and an associated work load on the users personally. Batch operations, however, can be simulated more easily, especially with regard to access speed to storage media and files. The meaningfulness of the results depends on the scalability of the test environment.

Who should take part in the performance measurements? The following surely:
• the project leader, who coordinates all activities, controls the completion of measurement schedules, cares for the allocation of hardware and software as well as for resources and budgets,
• analysts, who in the end decide, if sufficient data have been collected, and thus have to be in close contact with the testers themselves,
• testers, of course, which execute special test cases above the usual end-user functions,
• application specialists from the development area as well as from the application area itself,
• system administrators having in-depth knowledge of the environment under investigation,
• end users.

3.3.1 Examples from Performance Measurements

3.3.1.1 Example: Tuning of System Parameters

System parameters, for example, are responsible for how batch applications should run (normally in the background). By awarding the wrong priorities, it can happen that batch jobs will be executed in priority, such that the interactive user will experience long response times, since the CPU is occupied with batch processes. This is just one example for the fact that dwell time themselves in waiting queues does not provide enough information for conclusions regarding CPU load.

3.3.1.2 Example: Bottleneck Analysis

Despite many tuning attempts, the overall requirement for CPU time taken across all applications is too high in total. Even though all possibilities for optimisation have been exhausted by parameter adjustments, there still remain some more options besides a direct hardware upgrade somewhere:

• organisational measures to avoid parallel execution of certain processes,
• application tuning,
• I/O adjustments.

3.4 Data Collection Proper

3.4.1 Test Data

Normally, there is little time available for performance investigation in comparison with—for example—acceptance tests. Therefore, it is all the more important to prepare test data already during planning stage to have ample time disposable for the test phase itself.
Without support from the users, there is little sense inventing test data. Furthermore, copies from production data are subject to directives resulting from data privacy protection laws, so that these data have to be made anonymous beforehand.

It is important to document user activities prior to and after measurements. In case of measuring under full production work, instructions have to indicate the sequence of transactions (Table 3.1). In case of a simulation, these steps have to be preset during the planning phase. In this way, a realistic system load can be obtained. Therefore, it is important to tie in key users. Every executed transaction has to be tracked in log files for later evaluation. If possible, input values and parameters should be documented as well (in conjunction with test data).

During measurements, users should log screen response times for all relevant transactions.

### 3.4.1.1 Test Documentation

See tables 3.1 and 3.2.

### 3.4.2 Other Important Statistics and Indicators

Besides those reports and statistics delivered from professional performance tools (Chap. 8), there is certain information, which can be obtained manually. These comprise

- average number of users connected to the system/application,
- maximum number of users connected to the system/application,
- subjective total time of transactions,
- data volume handled,
- number of I/Os per device,
- time slots with respect to processing peaks,

**Table 3.1** Transaction sequence

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Step sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter contract</td>
<td>Start Citrix client</td>
</tr>
<tr>
<td></td>
<td>Login</td>
</tr>
<tr>
<td></td>
<td>User ID and password</td>
</tr>
<tr>
<td></td>
<td>Transaction start</td>
</tr>
<tr>
<td></td>
<td>Transaction end</td>
</tr>
<tr>
<td>Update contract</td>
<td>(Start Citrix client)</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Transaction start</td>
</tr>
<tr>
<td></td>
<td>Transaction end</td>
</tr>
</tbody>
</table>

...
• waiting queue occupancy over time and task related,
• subjective response time statistics of key users,
• batch statistics (average CPU time, number of average reads or writes),
• queries launched.

All those indicators can be related either to:
• online applications or,
• batch processes.

Indicators referring to online include:
• transaction rate,
• response time,
• number of users (in parallel).

In very rare cases, transaction times do equal factual response times. Indicators belonging to batch include (Fig. 3.2):
• throughput rate
• dwell time

each per job.

General control parameters for job execution are:
• system job limit (number of jobs executing in parallel) and
• job priorities.

Additionally, a timer facilitates to start jobs at certain times during the day or at night. This makes performance smoothing possible. Similar effects can be obtained by streams, where several jobs are chained after another, calling each other via a JCL procedure (job control language). Besides, different operating systems offer a variety of job schedulers, permitting fine-tuning of the overall batch scheduling. Operating manuals document these tools. In total, online and batch managements are subject to a superior task management, taking care of their competition with system programs. Task management allows assigning a higher priority than online applications to jobs—temporary and in justified circumstances.

<table>
<thead>
<tr>
<th>User</th>
<th>Transaction</th>
<th>Input data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID1</td>
<td>Contract management</td>
<td>Contract no.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Customer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Desired date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID2</td>
<td>Contract cancellation</td>
<td>Contract no.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Customer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reason</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: Input parameters
3.4.3 Monitoring

In addition to the already mentioned reports and dynamic system information, performance monitors deliver excerpts of the impact of applications currently running:

- average processor usage,
- time-depending processor usage,
- average memory usage,
- time-depending memory usage,
- current processes with status,
- files opened,
- disk occupancy,
- communication traffic.

Figures 3.3, 3.4, 3.5 and 3.6 show typical screen reports of performance measurements (from the TuneUp © tool of TuneUp company).

Besides this general monitoring, some aspects need special attention. One of them concerns paging activity in main memory. As already outlined in the theoretical chapter, this concerns the efficient utilisation of main memory within its limits with regard to required address space. Paging or swapping always leads to delays in response times and thus strains the CPU as well. And then paging also
3.4 Data Collection Proper

Fig. 3.3 Overall processor usage

Fig. 3.4 Detailed processor usage
Fig. 3.5 Memory usage

Fig. 3.6 Drive usage
causes I/Os to the detriment of other user demands. Altogether, the fact that online applications cause more load than pure batch applications has to be born in mind, when later interpreting measured data.

3.5 Special Case Cloud Applications

3.5.1 Introduction

Considerations regarding performance measurements, analysis and optimisation have targeted mainly either stand-alone computers, dedicated server environments or at the utmost networks until recently. Some completely new developments have lead to an enlargement of this subject, but also making it more complicated. The reason for this is the growing number of cloud applications.

At this stage, one has to distinguish different constellations:

- cloud only,
- cloud in connection with central application components,
- cloud in connection with other networks.

In theory, one could regard all three types of cloud solutions as network solutions and thus cover this subject within the framework of network performance theory. Unfortunately, the cloud parts in these constellations elude the usual performance checks for two reasons:

- The system platforms of cloud providers are either not known or not accessible for direct performance measurements.
- Platforms of cloud providers can change dynamically (hardware, networking etc.).

The following will briefly outline these problems. They concern

- application specifications themselves
- general approach regarding
  - response time behaviour
  - end-user response times
- test constructs
- measurements and
- analysis.

3.5.2 Cloud Applications

Generally speaking, cloud applications can appear in one of two guises:

- All applications run entirely in a cloud environment.
- Some parts of them run in the cloud, others again on central systems (Fig. 3.7).
For applications existing only in cloud environments, performance measurements too have to take place in the cloud. This means that all test pieces too have to be developed, stored and executed in the cloud.

But even for the alternate scenario—cloud as well as central systems—it also does make sense to measure and test from the cloud and to control even local applications accessed from the cloud or accessing to cloud functionalities. This certainly concerns the measurement act as such. Of course, local transactions have to be executed as usual locally as well.

### 3.5.3 Procedure

Figures 3.8, 3.9 and 3.10 illustrate the various procedures for possible testing:

- applications entirely local (central); performance monitor in the cloud,
- monitoring purely cloud applications,
- monitoring applications residing in the cloud as well as locally (centrally).

### 3.5.4 Response Time Behaviour

Cloud applications, either being hybrid, that is, working in conjunction with central components, or being completely in the cloud, complicate the estimation of response times. The reason for this is that external systems are accessed. These resources are of course outside of any local control. Their power, operating system parameters and application environment not only are unknown but can also not be
Fig. 3.8 Test of local applications from within the cloud

Fig. 3.9 Applications and services in the cloud

influenced. This is especially the case when wide area networks (WANs) are in place. Overheads to manage such networks lead to additional waiting times. To obtain an approximate picture, there are methods to determine realistic WAN waiting times by calculation. This is cumbersome and can only proceed via WAN emulation (Fig. 3.11). The quantities used mean

\[ t_c \quad \text{cloud time} \]
\[ t_w \quad \text{WAN time} \]
\[ t_z \quad \text{central time} \]
3.5.5 End-User Response Times

For the same reasons, just like overall response times, end-user response times as well are difficult to interpret. Purely technical performance measurements with the objective to extrapolate are often not sufficient. For end users, their relevant response times refer to the end-to-end time slices of business processes.

This means that when we are talking about response times, we are concerned only about client applications although all accesses are controlled by web services. Since such scenarios quite often lie on a critical path, it is recommended to run appropriate simulations and related investigations prior to releasing those applications for production—this means prior to hosting an end-user client on a web server.
3.5.6 Test Scripts

Test scripts, which are useful locally and which have been developed for central applications, can turn out to be useless after applications have been moved to the cloud. The complexity even increases, when taking into account additional administrative and other demands:

- security adjustments
- firewalls
- network routing
- access control.

Authentication mechanisms for the cloud can be entirely different from internally applied certificates. The same is true for the combination of username and password for logins. As a consequence, old and proven scripts have to be adjusted or completely rewritten for the new environment.

One very important aspect is due to the fact that test and development environments in the cloud are much more dynamic than elsewhere. Without asking approval from his customers, the provider may change his configuration according to his own technical or economic criteria without disturbing hosted functionalities at all.

Because applications can be moved on a virtual platform, tests may suddenly point into a completely different direction, when executed. Utilised process resources change, are tuned or experience upgrades without prior warning. These things may happen automatically behind the scenes, controlled by the service provider. This is the backdrop against interpreting performance fluctuations and questions the usefulness of measurements.

3.5.7 Conducting Measurements

In practise, more obstacles have to be overcome. As soon as measurements are ready to start off, other difficulties to test applications out of a local environment will appear. Many monitor solutions on the market need special ports. Furthermore, firewalls may prevent certain types of measurements.

Some difficulties are due to remote monitoring itself. For this reason, some providers make their own performance data available. These may, for example, contain information about how many bytes per second are moved around. So, there exist some sorts of external monitoring, which can be of use under certain circumstances (Figs. 3.12 and 3.13).

To be able to normalise observed transactions in the end, one could use some sort of generic standard transaction, for example, which can be moved through the complete architecture. This enables to get an idea of waiting times. Exactly for the reason that a cloud environment is so dynamic, one needs at least one standard transaction for comparison. Otherwise, absolute results will be meaningless. The reasons are obvious:
• limitations because of bandwidth,
• distances and waiting times between different areas of the cloud provider,
• unknown locations of the virtual machines,
• waiting times in dependence of cloud movements.

To summarise it, monitoring has to fall in line with cloud dynamics. At the same time, it makes sense to undertake investigations and optimisation in real time. As soon as a bottleneck becomes apparent, diagnostic tools should be at hand, which are capable to compile profiles, drill down deeper into a process and find out which queries, for example, are currently running causing the bottleneck. Real time is important, because cloud scenarios can be repeated only within certain
limits, since boundary conditions may have changed completely the next time round for reasons of the dynamics explained above.

3.5.8 Analysis of Cloud Measurements

Dynamic changes in the cloud thus can influence measurement results. This is the case when—as already outlined—cloud topology changes during observation. And performance in the cloud is not necessary the same everywhere. Identical applications may experience important differences between different providers. The optimal constellation depends on

- the type of application,
- the integration of the user,
- the operating system,
- other platforms and their configurations.

As is the case for central applications, one has to bear in mind that large volumes of test data will result in corresponding costs, if they are held online permanently or over long stretches of time. It therefore makes sense to download measured data and to evaluate and archive them locally.

3.5.9 Conclusion

Performance monitoring and successive optimisation regarding cloud applications are complex processes. There are limits regarding comparison with classical scenarios. Of course, the basic philosophy holds

- identification of bottlenecks,
- criticality of I/Os,
- response time behaviour.

On the other hand, there are difficulties regarding the analysis of CPU power and main memory management. The reasons are obvious:

- lack of transparency regarding platforms provided,
- lack of knowledge regarding operating adjustments and configuration,
- variability concerning cloud platforms,

to mention but a few aspects. Therefore, it is important to drill down the processes and tasks under observation and identify their controllable portions—for example:

- local parts,
- decentralised/centralised data management,
- I/O share.
Tailor-made test scripts and close cooperation with cloud providers as well as the choice of a suitable monitor can be helpful.

### 3.5.10 Checklist

For the checklist in Table 3.3 to make sense, a decision must have been taken beforehand to conduct performance analysis of cloud applications or portions of them.

### 3.6 Automation

There are normally two approaches for performance tests [1] (Fig. 3.14)

- manual procedures and
- automated procedures.

Unfortunately, manual testing brings along some intrinsic disadvantages:

- difficulty to emulate the behaviour of a large number of users,
- efficient coordination of user operations,
- no exact response time measurements because of user subjectivity,
- objective comparisons between different test scenarios are difficult.

<table>
<thead>
<tr>
<th>Table 3.3 Checklist cloud performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which are subjectively the critical business processes in all system supported applications?</td>
</tr>
<tr>
<td>Which components are supported by the cloud?</td>
</tr>
<tr>
<td>Which components can be measured independently locally?</td>
</tr>
<tr>
<td>Where can assumed cloud bottlenecks be located (I/O, data transfer rate, cloud processors, cloud main memory, data model etc.)?</td>
</tr>
<tr>
<td>Does a suitable cloud monitor exist?</td>
</tr>
<tr>
<td>Are dedicated test scripts available?</td>
</tr>
<tr>
<td>Will the provider deliver his own performance data?</td>
</tr>
<tr>
<td>Should a monitor be deployed in the cloud or locally?</td>
</tr>
<tr>
<td>Should performance data be stored in the cloud or locally as a download?</td>
</tr>
<tr>
<td>Is a stable cloud configuration guaranteed?</td>
</tr>
<tr>
<td>After the measurement: can the response time fractions be distinguished?</td>
</tr>
<tr>
<td>How can optimisation measures be implemented?</td>
</tr>
<tr>
<td>Do contractual agreements with the provider regarding performance standards exist (service levels)?</td>
</tr>
<tr>
<td>Is the provider willing to implement optimisation proposals?</td>
</tr>
<tr>
<td>Which measures for improvement can be implemented locally?</td>
</tr>
<tr>
<td>Is a change of provider an option?</td>
</tr>
</tbody>
</table>
Often, manual testing is not very practical. If one wants to analyse a specific problem with exactitude, the same tests have to be repeated several times over. The system has to be tuned; thereafter, retests have to take place to find out whether tuning has achieved something positive. Automated testing makes these things easier. Certain performance tools facilitate to store and administer the relevant test scripts and their results. Furthermore, human errors during the testing phase are avoided by automation. Such tools consist of the following important components (Fig. 3.15):

- a control unit for simulating and controlling the load,
- a virtual user environment (user generator),
- automated test scripts for online simulation (have to be tailored with respect to the applications under investigation),
- a load generator,
a monitor,
an analysis engine.

These features facilitate

• to replace manual users by virtual ones,
• simultaneous deployment of many virtual users on a single machine under load,
• repetition of scenarios to verify the impact of adjustments,
• measuring real response times.

In this way, costs and resources can be saved.

Even when in principle manual testing is preferred, automated testing and measurement can also serve to support manual processes. Occasionally, it becomes necessary to simulate heavy loads when insufficient end-user resources are available. This is especially the case when performance has to be simulated for new applications. Under these circumstances, the following information can be obtained:

• repeated load cycles under identical conditions,
• end-to-end processes and their behaviour,
• automatic recording of response times,
• monitor reports for individual system components.

Although the effort for automated measurements during a test phase itself can be considerably less than load generation by many users, automation demands a lot of preparation work. This comprises

• definition of test objectives (neutralisation of bottlenecks, optimisation of execution times or others),
• documentation of business processes, sub-processes,
• selection of transactions to be measured,
• selection of the relevant database when simulating: synthetic data or copies of production data.

Another possibility offered by automation is the complete elimination of real users. Instead, scripts have to be drafted which serve to fill the GUIs automatically with data and continue execution autonomously. To make sure that a process completes itself, one has to take care of “input errors” and predict system behaviour under these circumstances.

During automated performance measurements, the monitoring will immediately show in which place of the infrastructure process problems are generated. In this way, all critical units can be polled and investigated:

• CPU
• main memory
• communication channels etc.

Another advantage of automation is the possibility to introduce such a monitoring either for identical transaction sequences or at distinct times without much additional effort.
Chapter 4
Analysis

4.1 General

Analysis of measured data is the next important step to solve performance problems—actually the most important one. To achieve this and if circumstances call for it—depending on the bottlenecks identified—sometimes specialists have to be recruited:

• database experts,
• application architects,
• administrators and others.

Besides the statistical information which does not have to be consolidated further, the following reports from the dynamic data area are of interest:

• average processor usage,
• time-dependent processor usage,
• average memory usage,
• time-dependent memory usage,
• files opened,
• disk occupancy,
• communication traffic.

All the data collected as described in Chap. 3 permit to come to conclusions about the actual state of computer resources for specific applications. On this basis, recommendations for improvements can be drawn. To these belong, for example, inference about CPU/processor usage (busy, idle, overheads, etc.). The statistical distribution of user accesses leads to insights about bottlenecks during the day or for batch executions during night time. In connection with storage media occupancy, I/O problems can be identified. The usage of main memory gives indications about swapping, about optimisation requirements for code segments or even about unused resources still available.

All these data together represent a tangled mass of information. To come to practical insights, it is important to search objective oriented. It helps to bear in mind the following questions:
• Are there still storage capacities left?
• How important are the consolidated overheads demanded by the system alone?
• How is main memory usage shared?
• Which are the programs generating critical loads?
• Do system tables’ occupancies have reached critical levels?
• How often are utilities called from application programs?
• How is the account utilisation?
• Are there still disk space reserves available?
• Which programming languages are in use?
• Are directories and registries clean?
• What is the ratio between batch and online?
• Are there bottlenecks in connection with communication channels?
• What are the hardware and operating system versions?

Besides the absolute evaluation of measured data, cross-checks between measured data types play an important role: how do my I/O bottlenecks relate to the swapping rate and to the size of main memory at the same time? Is the impact of the cache size important? Concerning all these considerations, the option of repeating measurements after one or more reconfigurations should be taken into account. Do the data allow deducing acceptable limits relative to the number of transactions or the number of parallel users?

The result of a comprehensive analysis should be goal oriented with the aim of evaluating the overall context, allowing summarising all individual observations to a comprehensive hypothesis—without necessarily looking for a single cause alone. Quite often, there will be a number of weak spots, however, which combine to some plausible scenario. In this way, solution approaches can be identified exceeding a pure collection of individual measures and being of a more strategic quality.

During analysis, the following questions have to be addressed:

• (in case of overload) How can relative priorities between different applications be adjusted?
• What are the true demands on performance with respect to different applications?
• How can a possible conflict between optimal response times (online) and optimal throughput (batch) be solved with respect to operating resources taking, for example, the paging rate into account?

4.2 Data Reduction

The complete interpretation of all collected information will finally lead to insights and an evaluation of the results including recommendations. The data reduction presented in the following is initially related to the dynamic part. Raw data from
measurements do not necessarily expose salient features immediately as a basis for decision makers or untrained observers to draw consequences with regard to a badly tuned system. Quite a lot of effort has to be invested to consolidate the captured data. Of course, there are differences between data delivered from some high-level monitor and such data coming closer to the machine level. For the first type, results or average values can be presented in most cases directly from standard reports, whereas for the second type, experience is necessary to filter out the kind of information, which is relevant for the problems at hand.

In general, measurement intervals are short, and interrupts between transactions are of the order of milliseconds. Interrupts for all executing parallel processes are, for example, related to associated time intervals in quasi-stochastical fashion. These in turn have to be mapped to the external task numbers from system administration.

Furthermore, these processes may hide one or more programs, commands or utilities. The task at hand is to find out the relationship between these programs and aggregated transaction times between the interrupts belonging to a specific program.

High-level monitors occasionally do this but in most cases without reference to specific programs or processes. They display percentile CPU usage over longer periods of time (minutes or hours); the same is true for main memory and I/Os. The relationship with user processes has to be found out via parallel monitoring (online) of process start and process end events and the associated load peaks of the CPU, of main memory and from the history of external media storage for those critical points in time (triggering). This may mean delving into the world of code segments themselves.

In the following, some examples of standard results from performance data reductions are presented and explained (Figs. 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9):

• average CPU usage,
• time-dependent CPU usage,
• relative CPU load,
• average disk accesses,
• time-dependent I/O waits,
• main memory management I/O processes,
• average main memory management activities,
• time-dependent main memory management activities,
• paging I/Os.

4.3 Findings

Initially, some general observations from a first glance at dynamic and static data will be presented. Such general findings vary in depth and content with respect to different systems and thus cannot be standardised. Following this, some exemplary
conclusions will be provided and a rough guide to where to look further, without already attempting a detailed analysis at this stage:

- **Main memory:**
  - How much is still freely available?
  - usage including availability for user processes.
- **swapping:** total overheads,
- **spikes concerning the size of data and code segments,**

Fig. 4.1  Average CPU usage over 56 h

![Average CPU usage over 56 h](image)

Fig. 4.2  Time-dependent CPU usage during 4.5 h

![Time-dependent CPU usage during 4.5 h](image)
identification and process management of critical programs requiring many data segments or causing frequent I/O blockages,
- I/O blockages observed during measuring intervals,
- system tables with critical occupancy levels,
- distribution of test data and test programs,
- intermediate storage of temporary files,
- proliferation of utilities or system calls,
- decentralisation of files for general usage over the account structure,

Fig. 4.3 Relative CPU load

Fig. 4.4 Average number of disk accesses during 56 h
• Which accounts are oversized (number and size of files)?
• layout of paging areas,
• utilisation of diverse tools, languages and database management systems,
• location of directories,
• allocations of virtual memory, temporary file domains and spool space,
• ratio of batch-to-dialog applications and their evolution over time,
• communication traffic between applications and hardware resources,
• I/O controllers: number and usage,
• state of the art concerning hardware technology.
The listed aspects represent just some selected leads. The list can be prolonged and depends strongly on the system and the application environments under investigation.
4.4 Evaluation

From Figs. 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, the following conclusions can be drawn with all necessary caution.

4.4.1 Average CPU Usage

These data contain average values over longer time intervals including information about no activity at all as well. A percentile value of "idle" >0 seems to indicate a surplus of machine resources. However, users connected to the system during intervals of heavy system load or executing resource-intensive tasks would not agree. Bad response times are caused, when many tasks compete for the same resource; in between, however, longer periods of low activity may still exist. In this way, the statistics leading to the average values in Fig. 4.1 are generated.

20 % "idle" indicates a healthy system in any case, especially since in the same context, "busy" stays at 30 %. The allotments of the CPU and their relative states to each other have to be evaluated in front of this situation. In the example, 20 % "I/O waits" is too high and indicates access problems to external storage media, perhaps also to a controller throughput problem (15 % would be acceptable). "Memory management" with 19 % is also too high. 5–8 % would be reasonable. Other overheads ("interrupt control stack activity") would be too high by as much as 50 %, if no disk caching was in place. With disk caching, this value could rise above 15 %, but together with a significant lowering of the value for I/O waits.

![Fig. 4.9 Paging I/Os](image-url)
(some 10 % or less). As a rule of thumb, disk caching requires about the same CPU resources for I/O waits plus overheads as I/O waits alone without caching.

4.4.2 Time-Dependent CPU Usage

Although Fig. 4.2 shows CPU “busy” as absolute percentiles, they are still useful indicators concerning fluctuations of user requests within a production cycle. The percentile values themselves are less interesting with respect to the overall tendency, which is expressed by this graph: coffee break between 09:00 and 10:00 h.

4.4.3 Relative CPU Usage

Figure 4.3 is more telling. There are three artificially separated measuring intervals for this example. The relationship histograms demonstrate the effective usage of CPU resources. “Idle” has been put aside. The 50 % relationship indicates that a major part of the investment for this particular machine has been wasted to make sure that it runs at all. The comparison with the average value of CPU “busy” indicates whether there is still an “idle” potential available. During the third period, there is not much left in this example, whereas during the first period, some capacity may still be left despite high overheads. For this system, all non-productive CPU activities apparently increase with CPU “busy”. Further, in-depth analysis is necessary to find out, which portions (overheads, main memory management or I/O waits) are responsible for this bottleneck.

4.4.4 Average Disk Accesses

Figure 4.4 shows anisotropies, if at all, with respect to external drives. For reasons of load distribution across existing I/O channels and to minimise crowding out some access attempts, accesses should ideally be distributed evenly. In this example, disk 1 is over utilised and disk 4 is under utilised. By analysing log files for file accesses, correlations between access rates and file locations on the disks can be established.

4.4.5 Time-Dependent I/O Waits

Figure 4.5 has to be seen in conjunction with Fig. 4.2 to obtain a correlation between CPU “busy” states and I/O events. In this example, however, CPU
“busy” went down between 09:00 and 09:30 h but not the I/O rate. This can have several causes. On the one hand, it is possible that CPU-intensive tasks slowed down, whereas I/O events were continuing. On the other hand, efficient data queries may have come to an end, whereas more intensive sequential search operations in different files were continuing (perhaps in batch mode). A comparison with production schedules and user processes could shed light on this problem.

### 4.4.6 Memory Management

Figure 4.6 shows average and maximum values. If both values are high, something is wrong. As soon as maximum and average values approach each other on a high level, this indicates that main memory usage approaches its limits. If strong spikes are observed with otherwise low average values, this is an indication of inefficient design of certain programs. Then, the search must go on there (code and data segment management).

### 4.4.7 Average Memory Management Activities

“Free space” available and a small percentage of swapping (for overlay candidates) are normal. But if in addition, “give up” and “hard request” events are registered or more than 8–10% swapping is going on, this means a lack of main memory capacity.

### 4.4.8 Time-Dependent Memory Management Activities

Here again, correlations with other time-dependent graphs should be investigated together with user process statistics to identify program-specific problems in parallel with those concerning main memory management in general.

### 4.4.9 Paging I/Os

Figure 4.9 is compared with Fig. 4.6 but this time with respect to paging areas (virtual memory against main memory). Of additional interest are the locations of these areas and a possible correlation with disk access distributions.
4.4.10 Conclusions

There are two more significant observations from the reduction in dynamic data: the first relates to a list of programs, which put high demands on CPU resources (deduced from total CPU times during a specific measuring interval and from correlations between the distribution of transaction times and the corresponding processes). This list provides the names of those application programs, the code of which should be checked to find hidden inefficiencies.

The second list is generated from log file analysis—with regard to “file call” events (including program “open” and “close”). These events have to be extracted from log files, generally by using a program written for that specific purpose. This extraction program should then consolidate and sort all calls with regard to the corresponding programs. Its result can be used to interpret other measurement data.

All graphs and their individual interpretation taken together point to the nature of performance degradation and resource bottlenecks:

- I/O channels
- main memory
- CPU or
- combinations thereof.

In our considerations, nothing was said about the consequences from all these observations until now. These have to be drawn in conjunction with other available information:

- static data and
- application environment.

There were also no graphs presented, concerning disk storage problems. Once disk space occupancy goes up or above 75 %, a critical level will be reached. It seems obvious to buy additional disks. But there are other possibilities as well.

A final word concerning prioritising (Fig. 4.10).

If no unambiguous conclusions can be reached from the measured data, measurements could be repeated with regard to some of critical aspects: firstly, online

![Diagram](image-url)

**Fig. 4.10** Relationship between background and main applications (Siemens BS 2000/OSD Performance-Handbuch Nov 2009)
applications are launched until they tune to a stable state, and then, measurements take place until sufficient statistics have been accumulated; thereafter, batch processes are launched on top of this with monitoring continuing. In this way, the results obtained are comparable with each other. The opposite way, of course, can be taken as well by starting with full load and then reducing it step by step. Only after unequivocal conclusions about the usage of machine resources by applications is available, should manual prioritising be attempted.
Chapter 5
Optimisation

Recommendations for improvement are the final steps of performance analyses. They can be grouped into:

• short term
• medium term
• long term,

where transitions from medium to long term are—as is often the case elsewhere—fluent. Together, all possible measures relate to the following categories (Fig. 5.1):

• system tuning
• application optimisation and
• investments.

Furthermore, all these recommendations are subject to:

• priority
• feasibility and
• costs.

Sometimes system tuning without hardware changes or application redesign can be realised very quickly. The same is true for an instant main memory upgrade or the update to a new version of the operating system. Sometimes this may take time due to delivery delays by the distributor. Application optimisation usually takes longer because of possible changes to the data model or because of reprogramming. But occasionally, even here, quick solutions are possible, when obvious design faults have been discovered. Unfortunately, the most lasting optimisation measures can only be realised in the long term and demand some patience. And they will incur corresponding costs as well. If the focus would be on short-term measures only, those would have been identified long ago within the exiting framework. The following sections deal with optimisation measures in the sequence mentioned above (Fig. 5.2):
5.1 System Tuning

Recommendations for system tuning depend on the intrinsic capabilities of a given system. This means that the examples discussed in Chaps. 3 and 4 are more or less probable, but not necessarily complete.

The data obtained as presented in Chap. 4 allow for conclusions about CPU or processor load (busy, idle, overheads, etc.). The statistical distribution of user accesses leads to insights about bottleneck periods during the day or for batch execution during night time. Taking the occupancy rate of storage media, I/O problems can be identified. The usage of main memory points to swapping with optimisation requirements regarding code segments or indicate spare resources. Quite often, it is a combination of CPU problems, I/O and memory size, determining performance. Therefore, recommendations have to be seen in this overall context.

5.1.1 File Distribution

From Fig. 4.4 and as a consequence of log file analysis regarding file opening events, conclusions can be drawn to the effect which files have possibly to be redistributed across external disk drives to arrive at a realistic equilibrium. This means copy
5.1 System Tuning

operations, which can only be executed outside normal production times. The time required for this depends on file sizes and the number of files concerned.

5.1.2 File Clean-Up

The objective of this measure is to avoid unnecessary hardware investment after external storage space has touched its limit (>75 % occupancy). The tedious works consist in drilling through every file directory and deleting all files no longer in use. A related measure would be to install account and user space limits to force the user to clean up his directory himself, after which these limits can be waived again. Another control mechanism would be to regularly audit the date of last access and to delete those files which have not been touched upon within a pre-defined period of time.

5.1.3 File Reorganisation

Because frequently data records for a single file are distributed across several storage media, performance degrades, since accesses have to follow a succession of pointers. There is the option of a complete reload, which can be time-consuming. But at the same time, fragmentation would be smoothed and temporary files deleted.

5.1.4 System Disk

Because there is a permanent competition between user data access and access to system information (tables, utilities, instructions, etc.) to the same disk (data access in turn entails always system function calls as well), these activities can cause serious bottlenecks for I/O waits. A simple way out of this dilemma would be to separate system files from the rest by locating user data principally onto external media. This may mean investments, but a better performance would justify this. In this way, one would obtain a clean system disk with data only relevant to the operating system or its utilities.

5.1.5 Paging Areas

Paging areas can be configured and localised. Once paging has been identified as a source for problems, one should attend to the paging area. There are no fixed rules
about how to treat this. Here, we recommend careful trial and error procedures, since everything depends on the system environment.

5.1.6 Virtual Memory

Important for virtual memory are its size and its location—similar to the paging areas. The overall size depends on the maximum number of user processes permitted and on the maximum size of code or data segments, if those have not already been restricted by other measures. Virtual memory should—if possible—be located on one single drive only. In this case, competing I/O accesses play an important role again.

5.1.7 Spool Area

The question about the absolute size of spool areas can be answered quite easily by taking the largest possible spool files likely to be generated simultaneously as a measure (no other restrictions present). Size and location can be configured.

5.1.8 System Tables

The default configuration settings from the manufacturer will fit a specific production environment only in rare cases. As soon as the system tries to surpass the configured size, it will abort and generate a corresponding error message. Thus, system tables do not necessarily present an aspect of performance as such, but they are essential for the functioning of a system itself. Nevertheless, a performance audit presents an opportunity to check table occupancies and to adjust their sizes upwards, if they come close to 100 %. To this end, interdependencies between tables have to be taken into account. Tables which are permanently under occupied (<50 %) reserve unnecessary memory space and should therefore be reduced in size.

5.1.9 Buffers

There are some buffers critical to system performance: to enable the operation of peripheral devices, communication buffers and a certain number of system buffers. If response time problems persist, while otherwise a system seems to be tuned optimally (no I/O problems and no memory problems), one should have a closer look at these buffers and their configuration. Sometimes recurring data loss is an
indicator pointing to a buffer problem. Computer handbooks normally give
instructions as to buffer sizes in dependence of a number of parameters relevant for
production and applications. Communication buffers can become critical in con-
nection with certain protocols and within networks (Chap. 6).

5.1.10 Priorities

Some systems preview the allocation of priorities for certain processes (batch,
dialog), others permit the assignment of a specific priority to certain programs, and
again others allow prioritising of user processes. There are complex tools available
to dynamically adjust priorities in connection with the resource history of a par-
ticular process. It is important to arrive at the right mix. The basic rule is that
batches run at a lower priority than dialog applications, when both execute in
parallel. Although batches are not triggered by the same urgency than other
applications, they hold on to resources allocated to them in the first place.

On the other hand, critical programs for important users may gain a higher
priority (sometimes only temporarily). Priority readjustments may help certain
processes—to the detriment of others. Sometimes this is desired. It is important
that both overall performance and the general availability of resources are not
really impeded by these measures. Quite often, the first will degrade, when stan-
dard priority management is put aside. In this case, tuning is a delicate affair and
should proceed iteratively.

5.1.11 File Sharing and Code Sharing

Data file sharing in this context means non-exclusive access. For multiuser
applications, locking on file level should be prevented—even for writes, because
otherwise transaction waiting times and blockages will occur for other users.
Locking at data record level should be preferred. Further information on this
subject can be found in Sect. 5.2 (application tuning).

Code sharing can be achieved by setting certain parameters at program com-
ilation time to allow several users to execute a specific program at the same time.
Once the code has been transferred to main memory, follow-up users will no
longer experience long load times. Furthermore, one saves on main memory space
since only one code copy is executed.
5.1.12 Versions

Sometimes new operating system versions come along with features contributing to performance improvements: more efficient microcode, device tuning, more efficient system functions and utilities, etc.

Independent of a particular system condition, an update is always recommended, when such features may possibly solve one or more identified problems. This has to be balanced against the costs for interrupting production, weekend work and risks coming along with such a change for applications and terminal devices. Compatibility assurances from suppliers have to be taken with prudence, because they are difficult to verify beforehand.

Sometimes an update should be avoided, when additional features require more main memory space. In this case, performance would degrade. Or else, an additional hardware upgrade would be required. It is more prudent to resist the change until beta-test installations, or more risk-willing user communities have gathered more experience. After this, a more solid evaluation can take place to come to a better decision.

5.1.13 Drivers and Protocols

Although both cannot be tuned their performance characteristics often depend on the operating system version. In this sense, the same reasoning as in the previous section should hold. They are often part of a complex picture in the sense that sometimes a special advantage in one direction may entail disadvantages in others. A careful evaluation should precede any quick decision.

5.1.14 Registries

During the lifetime of a system, registries may get corrupted. This can happen because of the following reasons:

- After software has been removed manually. As a consequence, programs do not get de-installed, but just the installation directories have simply been deleted: they leave their traces in the registry. Entries in the registry remain and from now on point to files no more in existence.
- After software has not been installed correctly or completely. This problem may arise when there was not enough disk space available at installation time or the installation routine had been erroneous. In this case, it is likely that new entries have been created in the registry before the copying procedure had been completed. The registry may thus contain entries pointing to non-existing files.
Furthermore, entries in the registry database may be missing. When trying to access non-existing entries, the system may become unstable.

- After files of a particular application have been destroyed. This scenario is possible, when, for example, installation directories for two different applications are used at the same time or individual files had just been deleted [5].

As a consequence, errors, error messages and blockings arise. In the context of a general clean-up, registries should be brought up to date too. Depending on the system, corresponding tools are available.

### 5.2 Application Tuning

The subject of application tuning as a whole is vast, and its implementation is time-consuming and may lead to high costs. But since it may contain a high potential for performance improvement costs should be measured against those improvements—including the comparison with hardware investments. Everything being equal, hardware investments may quite possibly result in the desired results more quickly, but in the course of software clean-up, applications could be modernised and updated at the same time. The following elaborations deal with some specific and interesting aspects, when attempting to improve software with regard to performance optimisation.

#### 5.2.1 Application Architecture

This is the highest entry level. If applications are found to function on the basis of a principally wrong design, one could then of course throw them out and start from scratch. Sometimes certain parts can be salvaged and enhanced with reasonable costs. The following architectural faults are possible:

- unstructured software,
- inefficient database design (pointers, entities, keys, indices),
- cumbersome menu philosophy,
- frequently recurrent code segments with high load generation,
- fatal locking strategy (relatively easy to mend),
- cumbersome handshakes and error handling,
- frequent calls to resource-intensive routines,
- creation of large temporary files,
- inefficient buffer utilisation,
- inefficient communication strategy,
- inefficient design of distributed databases.
Some of the mentioned points will reappear again further down in this section. If something at the general application level should be seriously wrong, the architecture as a whole should be overhauled completely—at program and data management level.

5.2.2 Sub-Systems

What has been discussed concerning applications as a whole up to now is equally valid for individual sub-systems. If a sub-system can be decoupled from the rest of an application, its performance can obviously be investigated separately. If a complete application could logically be disassembled, this would ease sub-system-related analyses and subsequent performance improvement measures. In addition, a particular module causing the main performance problems could be pointed out. Furthermore, this approach would give the opportunity to decide, for example, which sub-system—if at all—should perhaps be moved to the cloud.

5.2.3 Function Calls

This section deals with the dialog part of applications and refers to:

- creation
- modification
- deletion and
- querying

data records. Each of these activities has to be seen in the context of a single business transaction (for example: creation of a contract in a sales module). The impact on performance is determined by:

- record length
- table size and
- transaction frequency

within the existing context of data management. Outside of the software itself, bottlenecks can be reduced by organisational measures (regulating transaction frequencies) or eliminated by redesign of the database. Furthermore, there is an optimisation potential on the GUI level (reducing the number of item fields on the screen) or reducing plausibility checks among other things. Another possibility is offered by the technique of indirect updates: the user is left under the impression that he is working online; in reality, the entered information is stored in a temporary file, which is processed later in batch mode, when system load is low—under the condition that application logic supports this.
5.2 Application Tuning

5.2.4 Program Structure

There exist sufficient methodologies describing how programs should be structured. A side effect is that programs become readable and are easier to be updated on the module level. Furthermore, code segment proliferation is avoided, which would occur in a monolithic structure together with all associated overheads of a sequential logic and its consequences for performance. Once problematic programs have been identified, their structure can be analysed and possibly redesigned. In this process, critical loops can be eliminated. Another possibility would be to swap out redundant code segments into a callable routine, which can be used by several different modules.

5.2.5 Frequent Calls and Opens

These concern programs as well as files. For programs, there are the following options:

- Keep them resident in main memory.
- Define them as sharable for several users.
- Integrate them in the calling program, when they are small enough.

For files, the problem becomes more complex: share ability is also an option but may not suffice, or it may become a source of problems for data record integrity. Another possibility would be to open files which are accessed frequently in the background to avoid loading times. More expensive options are database and application redesign.

5.2.6 File Sizes

If file sizes create a problem, there are several possible approaches:

- Dissect larger files into several smaller ones pointing to each other.
- Clean up data regularly and compress files.
- Changes in program size are part of application restructuring or efficient segmentation.

If nothing at all will work or is impossible to carry out and load times suffocate record search times, more main memory will be necessary. In case of long search times, different file access methods should be envisaged.
5.2.7 File Handling

If file handling (database management) should be a problem itself (because of size, access method or data redundancy), normally only a new design of the database as part of the overall effort of application redesign will help. This may lead to the consequence that parts of the application code itself have to be rewritten. Occasionally, however, even small changes in the database schema may in turn eliminate bottlenecks at low cost.

5.2.8 Sorts

A well-known effect on performance comes along with the execution of indispensable sort processes. The basic rule is that sorts should be carried out in batch mode—at a time, when no dialog applications are running (at night). If even during night time performance bottlenecks should persist, other options have to be considered:

- breaking up long sorts into several short ones,
- running sorts in main memory,
- sorting by program and not by utility,
- adjusting sort parameters,
- critical review of sort input.

5.2.9 Copying

What has been outlined for sorts is equally relevant for copy processes in applications.

5.2.10 GUI

Throughput problems in connection with GUIs can have the following causes:

- resulting from the tool managing the GUI,
- menu philosophy (number of hierarchical levels),
- screen design,
- functional structure of the application.

In case the interface is somewhat decoupled from the rest of the application, adjustments can be realised relatively easily. Concerning the menu, compromises have to be found between application modularity and the number of steps the user
has to follow to successfully finish a function during a complete transaction. Concerning screen design, this concerns the number of item fields on the screen and their ergonomic effect.

### 5.2.11 Production Schedule

The efficient separation, distribution and planning of user tasks across a daily available production time period and their impact on system resources can serve at the same time as an organisational tool to avoid bottlenecks. It is assumed that all productive installations maintain production schedules. This is the instrument, which allows coordinating between batch and dialog, between daily and nightly work. If systems are not available at night or during weekends, these problems have to be solved on an organisational or political level. An additional factor of disruption can also originate from the planning of backup processes.

CPU load distributions plotted over the day give indications about peak times. These may correlate with user tasks, and one should try to straighten competing transactions within the user community or department.

### 5.2.12 Training

Sometimes recommendations concerning staff training help to avoid long-term performance problems. In this way, inefficiently structured applications can be avoided in the future. At the same time, acceptance for redesigns can be created. Training subjects include:

- database techniques,
- structured programming,
- efficient languages,
- communication technologies and
- system tuning itself.

### 5.3 Investments

From the insights gained in Chap. 4, frequently recommendations for hardware upgrades result. Sometimes this seems to be the simple and quickest way to eliminate a bottleneck. Such quick shots suggest in particular that expensive and time-consuming tuning measures concerning applications can be avoided. And hardware is cheap, too.
Often, one achieves one’s objectives with processor upgrades, additional servers and memory extensions in the short term. However, two consequences have to be born in mind:

- interdependencies between different performance problems and
- long-term developments.

Interdependencies, for example, can result from a memory problem. After extending the memory, however, the problem persists since additional I/Os, which are now possible, increase the load on the processor disproportionately. The other way round, there may be similar dependencies when increasing processor power, resulting in a successive memory bottleneck, etc.

Long-term symptoms after hardware upgrades are known. Soon after a relatively short period of relaxation, demand on memory rises again and processor power does no longer suffice. These phenomena are equally known from town and roadway planning. Where there is more space available, more will be used up. Causes are non-optimised applications. And thus, hardware investments and costs will always remain as an on-going option.

### 5.3.1 Utilisation of Disks

External disk drive resources are normally requested by several different users at the same time. The waiting time for a single access to such a disk can serve as a yardstick for utilisation. For orientation, its reference point is that waiting time should not exceed more than 30 % of hardware handling time. If a disk resource is allocated to a single task, utilisation may easily attain 100 %. When evaluating these figures, the current CPU configuration has to be taken into account. Another criterion may be the average system load relative to the average number of processes which reside in the I/O waiting queue without manual priority allocation during a fixed predefined period of time.

### 5.3.2 Data Management

A popular measure is the compression of data. This can happen on two levels:

- elimination of blanks or binary zeros between items or records and tables themselves,
- defragmentation of disks (Fig. 5.3).

These are obviously two completely different approaches. But both have the same objective besides the pure gain of additional storage space, which today does not represent a major cost factor: to reduce the number of accesses and optimise cache utilisation and thus increase query throughput. Furthermore, query
throughput can serve as a measure for performance assessment, when applications are I/O bound. When compressing, one has to bear in mind that a later decompression is heavily CPU bound and therefore not recommendable during times of high CPU utilisation. In the course of all these investigations, problematic queries creating repeated resource bottlenecks should be identified. For this purpose, log files are a useful source again. One of the consequences could be to redesign the data architecture.

5.3.3 Conclusions

The simple recommendation, the quick fix and in many cases even a guaranteed solution can be found in hardware investments. Sometimes this is more cost-effective, and often, it is the only reasonable way out. A hardware upgrade either on the basis of an existing installation or by migration to more powerful configurations can eliminate bottlenecks and includes the following options:

- additional external storage media in case of:
  - storage capacity limitations
  - access problems to data

- I/O controllers and faster disks in case of:
  - access problems
  - CPU too often in wait state

- main memory in case of:
  - process management problems
  - segment size problems for data and code
  - swapping/paging
• CPU power to:
  – accelerate processes
  – permit more parallel processing.

However, the more intelligent solution is to lower response times noticeable to the user by a combination of hardware investments and system and application tuning. Since, however, one type of hardware measure can neutralise another different one, these traps have to be avoided.

To begin with, machine problems have a tendency to develop from a singular cause: either I/O or main memory related (storage space and communications can be considered separately) with some CPU relevance in both cases. Thus, one should initially concentrate on one scenario only. But if there really is a problem mix (Fig. 5.4), then the situation becomes more complex. Examples are as follows:

By enlarging main memory, certain problems may be solved, for example, a bottleneck when loading processes. But by permitting a larger number of processes to run in parallel, more I/Os will be generated. This causes I/O waits for the CPU. On the other hand, by strengthening CPU power, similar secondary effects may arise, since faster process execution leads to a higher data frequency.

By eliminating I/O bottlenecks, CPU waits are brought down, meaning more processes can run. This in turn only makes sense, when sufficient main memory is available.

Fig. 5.4 Problem mix
If system performance problems cannot be solved by a single type of hardware measure but only through a combination of different approaches, one should reconsider the configuration as a whole (this may be more expensive, but will lead to fewer mistakes when implemented).

Finally, the right choice of measures is determined on the basis of a mix of hardware, system and application tuning within a predefined framework of costs and time.
Chapter 6
Network Performance

6.1 Introduction

After performance and tuning have been discussed in general in the previous chapters, this one will deal specifically with special aspects of these problem areas with respect to networks and distributed applications. Classical performance tools, of course, have initially been deployed for mainframe architectures. Although the basic philosophy remains the same, a number of specific features have to be observed with network applications. Because of these general similarities, the whole performance philosophy will not again be unrolled but instead reference is made directly to tools available on the market and their evaluation will be presented. Special emphasis is placed on WLAN applications later on.

6.2 Peculiarities

The most important peculiarities concerning performance measurements and tuning regarding computer networks in comparison with classical environments are the following:

- several independent CPUs,
- distributed main memory,
- distributed applications,
- distributed data management systems,
- high communications overheads,
- specific network architectures and
- specific network hardware.

All these peculiarities lead to additional degrees of freedom, so that performance of a network is driven by a different dynamic than for centralised structures. This not only leads to a higher complexity regarding detection and elimination of bottlenecks. Quite often the impact of tuning measures is more difficult to predict, so that performance optimisation proceeds along a trial-and-error approach.
In spite of this, there exist a multitude of monitors, which can be useful for optimisation.

6.3 Evaluation

In literature and in the Internet, one can find about 400 different tools regarding network performance. Many indeed support the core business of performance monitoring—as described in the preceding chapters. Others—most of them—cover aspects having only limited relevance to performance and exceed it functionally by far, including:

- security aspects,
- WLAN,
- network traffic,
- Internet traffic,
- infrastructure monitoring,
- vulnerability and
- tracers and other.

Some of these tools are available as freeware, others are marketed commercially. In the following section, a selection of interesting tools will be presented covering the most important aspects of network performance (Table 6.1).

6.4 Tools

Table 6.1 lists a selection of network performance tools.

6.5 Networks

In principle, performance considerations for network environments follow the same criteria relevant to a single CPU. For this reason, one typically investigates, for example, a TCP/IP connection with respect to:

- the overall transaction behaviour,
- the transaction rate,
- throughput concerning individual data transmission and
- throughput rate,

when transmitting from main memory to main memory under exclusion of accesses to external media to approach a possible problem step by step, although in reality there will be an actual mix of access types.
<table>
<thead>
<tr>
<th>Tool</th>
<th>Manufacturer/distributor</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Various products</td>
<td>AlertSite</td>
<td>Measurement, diagnosis and reports about availability and performance of servers and web applications</td>
</tr>
<tr>
<td>Advanced HostMonitor</td>
<td>Share-It</td>
<td>System management tool, server availability and performance</td>
</tr>
<tr>
<td>Chariot</td>
<td>NetIQ</td>
<td>Evaluation of performance of network applications, performance prognosis</td>
</tr>
<tr>
<td>Ganymede</td>
<td>Ganymede</td>
<td>Performance management for clients, servers, applications and integrated networks</td>
</tr>
<tr>
<td>SaaS®</td>
<td>Gomez</td>
<td>Website and web application monitoring</td>
</tr>
<tr>
<td>Hyperic HQ</td>
<td>Hyperic</td>
<td>Monitoring, analysis and performance control, availability of web infrastructure, services, hosts, applications</td>
</tr>
<tr>
<td>Internet control panel</td>
<td>PC tools</td>
<td>Availability</td>
</tr>
<tr>
<td>PRTC Network Monitor</td>
<td>Paessler</td>
<td>Network up-/downtime monitoring, detection of system failures, performance analysis</td>
</tr>
<tr>
<td>Little:eye</td>
<td>CBR Software</td>
<td>Management of IT infrastructure, error and performance management, inventory and configuration</td>
</tr>
<tr>
<td>NEPM (Network Equipment Performance Monitor)</td>
<td>Nova Software</td>
<td>Measurement and analysis of performance data, reporting via email, under UNIX or Windows NT</td>
</tr>
<tr>
<td>Various products</td>
<td>NetPredict</td>
<td>Monitoring, prognosis and optimisation of distributed applications</td>
</tr>
<tr>
<td>Network Performance Advisor</td>
<td>NLANR</td>
<td>Measurement, analysis and reports of network performance statistics</td>
</tr>
<tr>
<td>NetQos Performance Center</td>
<td>NetQos</td>
<td>End-to-end performance monitoring, traffic analysis, performance of individual elements</td>
</tr>
<tr>
<td>NetworksA-OK</td>
<td>NextSysSecure</td>
<td>End-to-end monitoring of networks, performance and security</td>
</tr>
<tr>
<td>NetCrunch</td>
<td>AdRem</td>
<td>Visualisation of the physical network topology, performance monitoring, reporting</td>
</tr>
<tr>
<td>Nime</td>
<td>NimTech</td>
<td>End-to-end performance in client server architectures</td>
</tr>
<tr>
<td>NPS</td>
<td>Network Performance Services</td>
<td>Various products for performance, system utilisation, application monitoring and security aspects</td>
</tr>
<tr>
<td>Various products</td>
<td>ResponseNetworks</td>
<td>Service level monitoring and measurements</td>
</tr>
<tr>
<td>SAA (Service Assurance Agent)</td>
<td>Cisco</td>
<td>Response times, network resources, availability, connect times, application performance</td>
</tr>
<tr>
<td>Server Nanny Network Monitor</td>
<td>Xenos Software</td>
<td>Monitoring of services and network components, problem reports, logging of performance data</td>
</tr>
<tr>
<td>Orion</td>
<td>Solar Winds</td>
<td>Monitoring and collection of router, switch, server data; CPU utilisation, memory usage, disk space occupation</td>
</tr>
</tbody>
</table>

(continued)
Of course, for all transaction processes, network elapsed times are always part of the game as well. Sometimes these can be higher as simple CPU times. For this reason, such behaviour has to be analysed sectionwise. If the analysis tool indicates acceptable CPU times, the reason for bad performance is within the network itself (excluding all other resources). Because every network brings along its own complexities, there is no room here to elaborate in detail all possible topologies and technologies. Only general advice will be given.

### 6.5.1 LAN

Possible configurations (Fig. 6.1):

![Network topology](image)

**Fig. 6.1** Network topology

<table>
<thead>
<tr>
<th>Tool</th>
<th>Manufacturer/distributor</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>StableNet PME</td>
<td>Infosim</td>
<td>Performance management, topology monitoring</td>
</tr>
<tr>
<td>SwitchMonitor</td>
<td>NetLatency</td>
<td>Network performance monitoring under Windows®, component utilisation and error tracking</td>
</tr>
<tr>
<td>Vantage</td>
<td>Compuware</td>
<td>Application performance management, analysis, response times, end-to-end</td>
</tr>
</tbody>
</table>

* SaaS Software as a Service
• Fixed line:
  – up to 64 Kbit/s
  – up to 2 Mbit/s
• Ethernet
• fast Ethernet
• gigabit Ethernet:
  – 10 Mbit/s
  – 100 Mbit/s
  – 1,000 Mbit/s.

Bottlenecks may be caused by:
• network gateways,
• the client system,
• network configuration,
• network topology,
• application topology in the network and
• network load from cloud applications.

Overall criteria are as follows:
• clean data transmissions (addressing, failure free),
• subjective response time experience for all users and
• good performance of individual components.

One has to bear in mind that nominal transmission rates include a significant part for protocol overheads. If individual CPUs are unable to follow up with their respective processing data packets may get lost causing additional resource demand for repeated transmission attempts. To generally reduce network strain, sometimes the following measures can help:
• avoiding I/O intensive transactions via the network,
• designing application topologies to allow local operations for a maximum number of users and
• file server should be the most powerful server around.

### 6.5.2 WLAN

Only at the end of the 1980s of the last century did the IEEE begin to develop suitable standards. In 1997, the first WLAN standard was published under the number 802.11. In the years following, several enhancements and extensions were made public, identified by a special letter at the end of the number.

In 1999, the IEEE published two more standards, the 802.11a and 802.11b. The latter is the standard most widely in use today. This goes for private applications,
companies and publicly available hot spots as well; 802.11b allows for gross transmission rates of up to 11 MBit/s, most of which is used up for protocol overheads; 802.11b works within the 2.4 GHz frequency range and uses the HR/DSSS procedure. In 2003, the standard 802.11g was adopted, permitting transmission rates of up to 54 MBit/s within the same frequency range; 802.11a uses frequencies of the 5 GHz range. In 2004, the standard 802.11i finally offered improved safety architecture.

The standards of the 802.11 group follow the International Organization for Standardization (ISO) definitions (Open Systems Interconnection Reference Model). This rather abstract model describes the communication between open and distributed systems on a functional basis along seven layers of protocol, built upon each other. Open means that the model is not bound to certain company standards, and distributed means a decentralised system environment.

If an application attempts to start a communication between two addresses within a network, all the ISO layers—each with its special assignment—are passed through in sequence. The lowest layer is the physical layer (PHY). The protocols belonging to this layer describe connection set-up and connection tear-down with the components in question and the transposition of data into physical signals, them being electrical, electromagnetic or optical.

Initially, the standard dealt with components of wireless networks allowing for transmission rates between 1 and 2 MBit/s. The envisaged radio technologies included frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum (DSSS) as frequency spreading alternatives.

The later versions of 802.11 specified wireless connections with transmission rates of up to 11, respectively 54 MBit/s using different techniques of frequency modulation: high rate direct sequence spread spectrum (HR/DSSS) and orthogonal frequency division multiplexing (OFDM).

When buying a WLAN component, the device designation is preceded by a reference to the standard version, for which the device has been released, for example: 802.11b Wireless USB Adapter. The figures relate to the IEEE standard for WLANs, and the letter indicates one of the various versions of the standard. These letters refer to the different task forces within the IEEE Work Group for WLAN standards.

Since in most countries the 2.5 GHz band can be used licence free and without restrictions, chiefly products using the standard 802.11b are in circulation. More recent products support 802.11g or even 802.11i. This raises the question of compatibility between the various versions. More and more products enter the market supporting several different versions. Netgear, D-Link, Lancom or 3Com offer access points for professional use supporting operational modes for 802.11a, b and g or 802.11b and g or alternatively b or g.

Some notebooks are equipped with integrated WLAN functionality by Intel’s Centrino Technology, so that a special adapter is not needed. Besides the support for 802.11b, a two-band solution 80211.a/b is possible supporting both modes 802.11b/g.

Table 6.2 summarises the standard version of the 802.11 family and their main features.
6.5.2.1 Power

The 2.4 GHz and the 5 GHz frequency ranges for a 802.11 WLAN configuration belong to the metric waves category having a range of some hundred metres unto some kilometres. When line of sight between sender and receiver is possible, the conditions for communication are optimal.

Despite the fact that the ISM band is license free, there are some regulations specific to certain countries. This concerns especially the transmission power. Within the EU, the upper limit is 20 dBm for the 2.4 GHz frequency band and 30 dBm for the 5 GHz band with respect to the effective transmission power. These regulations may neutralise a possible gain by employing a directional radio antenna, since in this case technical measures have to be employed to restrict the power again to remain within the confined region.

6.5.2.2 Controlled Data Transmission

In its simplest form, data transmission in a network functions via a point-to-point connection. Two computers are connected via a carrier medium: cable or radio frequency. Each transmission proceeds in three phases: connection set-up, connection control and connection tear-down. Modems have to synchronise during connection set-up before data can be transmitted. During transmission security, mechanisms are active to prevent erroneous transmissions.

Data transmission between sender and receiver is achieved via so-called protocols. They control the data exchange on several layers of communication.

6.5.2.3 Deployment of Routers

In case, more than two participants want to join in the simple point-to-point connection has to be replaced by a network. In order to locate the different participants, they have to have addresses.

In large networks, several routes between any two stations are possible. Routers can select the optimal path.

<table>
<thead>
<tr>
<th>Standard</th>
<th>Frequency band (GHz)</th>
<th>Data transmission rate (MBit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>802.11</td>
<td>2.4</td>
<td>2</td>
</tr>
<tr>
<td>802.11b</td>
<td>2.4</td>
<td>11</td>
</tr>
<tr>
<td>802.11g</td>
<td>2.4</td>
<td>54</td>
</tr>
<tr>
<td>802.11a</td>
<td>5</td>
<td>54</td>
</tr>
<tr>
<td>802.11h</td>
<td>5</td>
<td>54</td>
</tr>
</tbody>
</table>
If participants want to use the same carrier medium both for sending and for receiving, special rules to detect or prevent collisions have to be adhered to. The carrier sense multiple access (CSMA) pools the most popular ones of these.

### 6.5.2.4 Packet Switching

When telephoning, the two participants are provided with a fixed connection during the duration of the call. This is called line switching. The Internet by contrast works by packet switching. The transmitted information is cut into blocks called packets (Fig. 6.2).

Every message contains a header with information required for the data exchange—like the sender and destination address. The packets move autonomously through the net. And it is quite possible that packets—although belonging to the same message and originating from the same sender—will be conducted to their destination via different routes. Only at arrival will they be patched together again.

The advantage of packet exchange is the efficient usage of existing connections, since the data packets are small and they do not have to queue for long. The communication network is available to all participants. All stations can send packets in turn. Errors will be detected immediately. An erroneous packet will be resent. If a station fails, the total message will not be lost completely. The packets choose a different route to reach the destination address.
6.5.2.5 Bandwidth

An important role is played by the carrier medium transmission capacity. Any possible data rate depends on the bandwidth, that is, the frequency range of the signal transmission. By increasing the bandwidth, the amount of information to be transmitted per interval increases as well. Bandwidth is measured in Hertz (Hz) or kHz, MHz, GHz. The rate of data transmission is given in KBit/s or MBit/s. If a transmission in both directions is possible, this is called a duplex connection otherwise it is a simplex connection. A connection is called half-duplex, if the connection direction can be swapped.

6.5.2.6 Range of Radio Signals

Radio waves are electromagnetic, and they propagate in vacuum with the velocity of light. The received power decreases with the square of the distance, which means that the range of transmitters has a principle limit. The realistic distance a radio signal can travel depends also on signal attenuation, also called damping and of course on possible interference. And the range depends of course on the frequency in use. Signals having a low frequency can have a far range even at low intensity. They can also pass physical barriers such as walls. This is not the case for signals operating at high frequencies between transmitter and receiver.

Contrary to cable connections, the transmission medium in a radio network does not have visible bounds, which could be located easily. At the same time, the transmission medium is basically unprotected against unwanted signals coming its way. There is also no guarantee that all stations belonging to a wireless network can hear each other at any moment. There is always the possibility that stations may be hidden momentarily. The propagation of radio waves varies over time and is not necessarily isotropic in space. Figure 6.3 from the 802.11 specification of 1999 visualises this effect.

6.5.2.7 Channel Distribution

As was already pointed out above, WLANs use the frequency range offered by the ISM band. The 2.4 GHz band is divided between 2.4 and 2.4835 GHz into single channels with a width of about 22 MHz each and a gap of 5 MHz between them. Because of spread effects, there may be frequency deviations of 12.5 MHz in both directions against the nominal frequency assigned to the proper channel. Therefore, interference between adjoining channels is possible.
6.5.2.8 Channel Separation

Interference can be avoided by using only parallel channels with sufficient separation. The best method is to use only every fifth channel, which means that only three channels at maximum can be used in a single WLAN. By reducing interference signals, transmission power can be augmented. Figure 6.4 illustrates

Fig. 6.3 Representative distribution of intensities of radio signals (ANSI/IEEE 8.2.11, 1999 Edition)

Fig. 6.4 Channels 2, 7 and 12 are without overlap [6]
channel separation, and Table 6.3 gives the frequencies of different channels in the 2.4–2.5 GHz frequency band.

6.5.2.9 ISO: The Physical Layer

The problem to be solved on this layer concerns the fact that within the frequency range provided, many stations may want to transfer data at the same time. This happens, when the range of potential senders and receivers overlaps.

To be able to distinguish between the different participants, FHSS and DSSS are employed to generate the necessary signals. FHSS is the older method, whereas DSSS is used more widely today. Both procedures are not compatible. This means that within a WLAN, all components have to follow one or the other.

The spread spectrum procedures are less sensitive to interference or electronic disturbances than single channel methods.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Central frequency (MHz)</th>
<th>Frequency spread (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2412</td>
<td>2399.5–2424.5</td>
</tr>
<tr>
<td>2</td>
<td>2417</td>
<td>2404.5–2429.5</td>
</tr>
<tr>
<td>3</td>
<td>2422</td>
<td>2409.5–2434.5</td>
</tr>
<tr>
<td>4</td>
<td>2427</td>
<td>2414.5–2439.5</td>
</tr>
<tr>
<td>5</td>
<td>2432</td>
<td>2419.5–2444.5</td>
</tr>
<tr>
<td>6</td>
<td>2437</td>
<td>2424.5–2449.5</td>
</tr>
<tr>
<td>7</td>
<td>2442</td>
<td>2429.5–2454.5</td>
</tr>
<tr>
<td>8</td>
<td>2447</td>
<td>2434.5–2459.5</td>
</tr>
<tr>
<td>9</td>
<td>2452</td>
<td>2439.5–2464.5</td>
</tr>
<tr>
<td>10</td>
<td>2457</td>
<td>2444.5–2469.5</td>
</tr>
<tr>
<td>11</td>
<td>2462</td>
<td>2449.5–2474.5</td>
</tr>
<tr>
<td>12</td>
<td>2467</td>
<td>2454.5–2479.5</td>
</tr>
<tr>
<td>13</td>
<td>2472</td>
<td>2459.5–2484.5</td>
</tr>
</tbody>
</table>

Fig. 6.5 Frequency jumps (schematically) in the FHSS procedure
The FHSS Procedure (Fig. 6.5)

The frequency hopping spread spectrum was originally developed during the Second World War to control torpedoes. As indicated by its name, the radio signal is subdivided into small segments and hops within fractions of a second several times from one frequency to another. The quasi-random selection of the frequencies is achieved by the Gaussian frequency shift keying (GFSK) method. To be able to handle the signals, the receiver has to know the pattern of these hops.

Other sender/receiver pairs may at the same time use a different hopping, so that several data transmissions can run without interfering each other.

If in the unlikely case a collision takes place, the system sends the data packet again, until the receiver sends a handshake. The 2.4 GHz band for WLANs is subdivided into 75 subchannels with a width of 1 MHz each. The disadvantage of FHSS is the relatively high overhead with respect to the useful data generated by the frequency hops. Therefore, data transmission with FHSS is relatively slow. The maximum is 1 MBit/s.

The DSSS Procedure (Fig. 6.6)

The direct sequence spread spectrum procedure is widely used and works differently from FHSS. DSSS uses just one single 22 MHz wide channel without frequency hopping. The data stream is combined via an XOR operation with a so-called chip or chipping code. The data are represented by a random sequence of bits known only to sender and receiver. The zero is represented by inverted chipping code.

The code spreads the transmitted data over the available bandwidth. Longer chips need a wider bandwidth, but increase the probability that the data are transmitted correctly.

The advantage of DSSS is that the receiver can easily find out, whether the data have originated from the same sender that has generated the code. The procedure

![Fig. 6.6 Coding in the DSSS procedure](image)

1 bit

corresponding 8 chip code
also facilitates error checking, since bit patterns not corresponding to the code can be filtered out. If there are one or two bits in the pattern, which have not been transmitted correctly, an automated correction takes place without having to send the data again. Since the protocol overhead is lesser for FHSS, higher transfer rates are possible. Newer variants of 802.11 take that into account.

With DSSS, the data packets are extended by a 144 bit prefix; 128 bits are used for synchronisation and 16 bits for a start of frame field. This is followed by a 48-bit header with information about transmission rate, the length of information within a packet and a control code. Since only the header determines the transmission rate of the succeeding user data, the header itself is always transmitted with 1 MBit/s in the first place.

Even though the prefix will be removed in the ongoing process, its length is still taken into account, when calculating the transmission rate. The effective transmission rate thus is always less than the nominal rate.

The HR/DSSS Variant

An improved version of DSSS is called HR/DSSS. This is the frequency spreading procedure most commonly used in WLANs. Just as DSSS itself, HR/DSSS operates within the 2.4 GHz band. It uses a modulation technique called complementary code keying (CCK). Data can be transmitted at a rate of up to of 5.5 or 11 MBit/s.

The OFDM Method

Another option for signal generation in radio networks is called orthogonal frequency division multiplexing (OFDM). This technique operates within the 5 GHz band. Contrary to FHSS and DSSS, OFDM transforms the digital data into multiple analogue signals in parallel. The frequency bands are separated into four channels each, which are split again into another 52 subchannels of 300 kHz width each. The subchannels may overlap. Interference is avoided by scheduling. The advantage of this procedure is a much higher transmission rate.

The transmission rate depends on the modulation technique employed. With binary phase shift keying modulation (BPSK), 6–9 MBit/s can be obtained, with quadrature phase shift keying (QPSK) 12–18 MBit/s and with quadrature amplitude modulation (QAM) 24–36 MBit/s.

Furthermore, OFDM was applied to the 2.4 GHz frequency band with the aim to obtain similar transmission rates here as well. Table 6.4 summarises the modulation methods.

<table>
<thead>
<tr>
<th>Table 6.4 Modulation methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>Binary phase shift keying modulation</td>
</tr>
<tr>
<td>Quadrature phase shift keying</td>
</tr>
<tr>
<td>Quadrature amplitude modulation</td>
</tr>
<tr>
<td>64-QAM modulation</td>
</tr>
</tbody>
</table>
6.5.2.10 ISO: The Medium Access Layer

The procedures specified by WLAN standards for accessing the transmission medium differ from those in the 802 family defining accesses in local area networks generally. These differences are due to the nature of wireless transmissions. The 802.11 standard defines different services for the carrier access layer controlling data exchange within the network. These services concern the way, in which data are prepared for transmission and the necessary security provisions.

Frames and Fragmentation

If data are to be exchanged via a wireless connection, they have to be divided up and packaged in a suitable way. The 802.11 standard calls these MAC protocol data units (MPDUs). It defines exactly how such a data packet or frame should look like. There are different types of frames: data frames for user data, control frames for control data and management frames to manage the network operations.

A data frame consists of a MAC header, the frame body containing the user data and a frame check sequence, containing a check sum as a 32-bit cyclic redundancy code (CRC). Figure 6.7 shows the components of a data frame.

Whereas the length of the frame body can vary, all other fields and sequence are fixed. The first address contains the target address of the final destination, the second address is the original address of the sending station, and the remaining addresses control the forwarding.

Configuring an access point, a fragmentation threshold can be defined. This value determines the maximum packet length. If this length is exceeded, the packet is divided up into so many more fragments.
6.5.2.11 Collisions

The MAC layer controls primarily the data traffic within the radio network; 802.11 defines several services for that purpose. One aspect is the avoidance of collisions. Collisions would otherwise occur, when several stations are sending data packets at the same time.

CSMA/CA

The access to a radio channel in systems according to the 802.11 standard is controlled by a random procedure called carrier sense multiple access with collision avoidance (CSMA/CA). This technique permits the simultaneous access of several devices to the carrier medium. The working of CSMA/CA can be described in the following way.

When a station wants to send a data packet, it checks the carrier medium, whether other signals are present. If it cannot detect any signal, it waits for a short random time interval (interframe spacing) and then checks again, if the medium is clean. If this is the case, the data packet is transmitted. The station having received the packet checks its integrity. If everything is OK, a receipt is sent after another short interframe spacing. If the sending station does not receive an acknowledgement, it is assumed that a collision with a different data packet has taken place. The station waits again for a random time interval and then tries again.

RTS/CTS

An optional extension of CSMA/CA is RTS/CTS. This procedure is applied to master the problem of “hidden” terminal devices. These are devices, which sometimes cannot be reached because of signal attenuation. Initially, the station which wants to send emits a request to send packet to reserve a transmission channel. The receiver acknowledges this reservation with a clear to send package. All other stations remember the holding time established by the RTS/CTS packets and refrain from sending data themselves during this time interval. If more than one station tries to send data at the same time, CSMA/CA instructs all other stations safe one to refrain and try again later.

While configuring an access point, one can normally define a threshold. This value determines whether the packet transmission can be handled by the CSMA/CA or by the CSMA/CD method otherwise used in LANs as a function of packet size. In the latter case, the packet will be sent after a certain holding period.

6.5.2.12 Address Space

The protocols of the 802.11 MAC layer operate within the same address space generally provided for local networks adhering to the 802 standards family. To identify components, the medium access control address (MAC) is used. This is a unique 48-bit serial number assigned by the manufacturer of the relevant network.
component. The first 24 bits contain the manufacturers ID, assigned by the IEEE, and the rest is filled up by the manufacturer himself. This number is generally represented in hexadecimal code like 00-09-5b-e7-b3-5c with a hyphen as separator.

Addressing (Fig. 6.8)

With the help of MAC addresses, the MAC layer can get in touch with higher levels of the ISO model. In this way, it is possible to assign, for example, an IP address to a component with a specific MAC address. This mapping is done by the address resolution protocol (ARP).

Since both LANs and WLANs use MAC addresses, in the same way it is no longer possible to distinguish, whether a user utilises a LAN or a WLAN component at the Internet protocol level.

6.5.2.13 Options

The basic elements of the standard cannot be modified. When trying to tune WLAN applications, there are the following parameters to be considered:

- number of terminal devices attached to a single access point/router,
- power of communications controllers and
- selection of standard version permitting the highest data rate.

Of course, the development of the 802.11 has not come to an end yet. In the future, protocols with considerably higher data rates can be expected.

802.11n

Although 802.11n has not been finalised, yet components adhering to the actual state of the art are already on the market. Work on it had actually started as far back as 2003 with the 2.0 draft in September 2007. There are still outstanding issues to be attended to like support for some individual features. The most recent draft version is that of 10.0 from 15 May 2009.

802.11n works like everything else in the 2.4 and 5.0 GHz frequency bands. The target is a transmission rate of 600 MBit/s and a range of up to 300 m. However, these are theoretical values. A practical rate of 100 MBit/s is more

<table>
<thead>
<tr>
<th>#</th>
<th>IP Address</th>
<th>Device Designation</th>
<th>MAC Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>192.168.0.3</td>
<td>MYTRAVELMATEXP</td>
<td>00-00-e2-30-6e-82</td>
</tr>
<tr>
<td>2</td>
<td>192.168.0.4</td>
<td>DELLPROF</td>
<td>00-10-5a-bb-0b-cb</td>
</tr>
<tr>
<td>3</td>
<td>192.168.0.5</td>
<td>FUJI</td>
<td>00-30-f1-15-4b-6f</td>
</tr>
</tbody>
</table>

Fig. 6.8 Mapping of IP to MAC addresses in the access point list
likely. This is due to interactions with multiple types of components within a typical network. Since the new standard is backwards compatible with 802.11a, b and g, the rate may be even slower yet.

The standard proposes three major technological features: multiple input multiple output (MIMO), channel bonding and frame aggregation. In MIMO, multiple senders and receivers are used. Via spatial multiplexing data streams are divided up and then sent in streams over the same channel simultaneously. The receiver reassembles these streams using a complex algorithm. Besides this, MIMO focuses the energy of the radio beam into the desired direction of the receiver. By channel bonding, 802.11n systems can patch together two adjacent 20 MHz channels into a single 40 MHz channel, thus doubling the transmission rate. By combining individual data frames to larger packets, thus reducing the total number of frames and the associated overheads, payload again is increased.

Persistent problems with the standard at the moment are high power usage and security degradation. The latter is due to the fact that intrusion detection scans take twice as long as of now, leaving twice as much time for hackers before interception.

802.11ac

The 802.11ac is oriented at the 802.11a and thus will be operating in the 5.0 GHz band. To be able to use it, sending and receiving equipment have to be upgraded. The transmission channels available will provide a theoretical throughput of maximally 1 GBit/s. In addition, the 802.11ac will transmit at a lower frequency, so that walls will be less impedimental. Just as for the 802.11n, it will have MIMO and channel bonding.

802.11ad

In parallel to the 802.11ac, the 802.ad is under development. It will posses similar characteristics as the 802.11ac but operates in the 60 GHz band. Theoretical throughput is supposed to be 7 GBit/s. One disadvantage will be its low range, which means that it can be used only in one and the same room. Altogether four channels with 2.16 GHz width each will be required. This is significantly wider than for the 802.11n, so that the 802.11ad can do without channel bonding. By employing beam shaping techniques, the frequency rises so that walls will again present an obstacle.
Chapter 7
Process Performance

7.1 Starting Position

Process performance to be discussed in the present performance context does not refer to the classical approaches regarding, for example, work flow optimisation in companies, primarily also not the advantages of synergy effects or a measurement using the “Balanced Score Card”, but is treated as a by-product of system and application optimisation as elaborated so far.

Performance measurements and analysis provide the opportunity not only to realise improvements on systems but also to cast a critical view on the applications behind them regarding their business nature. In this sense, one has to differentiate between aspects regarding existing applications and those becoming relevant when new applications are to be introduced. For the latter, the important thing is to identify possible additional overheads and take appropriate measures beforehand.

7.2 Identifying Critical Business Processes

When looking at critical business processes, two things have to be born in mind (Fig. 7.1):

- Processes relevant to performance to optimise the associated system support,
- Identification of applications causing bottlenecks; then straighten them functionally with the aim to optimise the whole process.

The first objective is strongly related to application optimisation itself as already discussed.

7.2.1 Processes Relevant to Performance

Quite often only few processes are responsible for the major load on a system, for example, 15 % of all processes for 75 % of the total load. This means it is necessary to carry out a preselection of the transactions in question and classify them according to load impact, for example:
queries: generally done by many users and I/O intensive,
printouts: few users, low load,
data entry: these usually trigger transaction chains (read, create, verification and change).

During performance measurements, care has to be taken to include the most important transactions in the test process to obtain a realistic picture about critical business processes and their throughput performance. For this selection, key users, administrators and developers should be consulted. To begin with these, transactions should be documented. A classification could proceed along the lines shown in Table 7.1.

Complexity and impact may be judged subjectively, for example, by values ranging between 1 and 5. Complexity refers to chaining and computing intensity, impact tells something about influence on load with respect to all other applications.

Other important information is the distribution of transactions over time. The total business is not conducted all at once or in a constant manner over a working day—especially not, when an organisation operates with flexible working hours. Figure 7.2 shows such a distribution for some important transactions relative to the number of concurrent users. One can distinguish clearly the spikes shortly after business commencement and during early afternoon but also valleys of activity during lunch break.

7.2.1.1 Approach

The identification of processes relevant to performance is a systematic task, the effort of which depends strongly on the state of process documentation in an organisation. Once this base has been established, the identified processes can be included into performance measurement scenarios. Volume tests are generally very useful for this purpose. Figures 7.3 and 7.4 show a possible approach.

It is obvious that all this is an iterative procedure, which will command much time and resources. It is therefore indispensable to make a cost-benefit analysis.
during preparation phase, which gives indications whether the whole exercise is worth its while. Once the decision has been made, the whole subject of process optimisation should become a project in its own right, since probably several

<table>
<thead>
<tr>
<th>Transaction</th>
<th>No. of Parallel Users/Unit Time</th>
<th>Complexity</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1 Evaluation of transactions

Fig. 7.2 User frequency distribution

Fig. 7.3 Approach

<table>
<thead>
<tr>
<th>measures</th>
<th>objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial outlay estimate</td>
<td>project planning</td>
</tr>
<tr>
<td>preparation volume test</td>
<td>project organisation</td>
</tr>
<tr>
<td>volume test 1</td>
<td>data collection</td>
</tr>
<tr>
<td>reduction</td>
<td>selection</td>
</tr>
<tr>
<td>volume test 2</td>
<td>realistic basis</td>
</tr>
</tbody>
</table>
departments as well as responsible persons from IT have to be involved. For the project to be driven with the necessary priority, the backing of the organisational hierarchy is necessary. This will result in a project organisation as shown in Fig. 7.5.

The sponsor is normally the management board. In the end, it is this level at which decisions have to be taken, which processes have to be included in the investigations. The working groups are structured according to business content or along departmental lines. Initially, they have to document the processes, and then participate in the performance tests and finally are consulted when decisions about optimisation measures are to be taken. In the end, it is also them that have to implement these decisions and work with the results in the future. Above all, a steering committee watches over the whole project, takes care about the operative implementation and reports to the sponsor.
7.2.1.2 EDPs

It has proven to be useful to depict processes within the framework of EDP conventions. EDP means “event driven process”. Starting either from a predecessor event or a stimulus from outside a function is triggered, which either produces a follow-up function, a new event or a result. Figure 7.6 shows an example.

In Fig. 7.7, the left column labels the relevant procedure in plain language; the columns to the right list the participating systems. Every little rhomb depicts an event. The arrows reflect the information flow. Additional comments may be included.

Figure 7.8 shows a specific example for a business transaction.

7.2.1.3 MEDs

MED (Main Expense Driver) means a process, which causes a high work load for a user. This work load may have its causes based on

- the time necessary for a single transaction and
- the volume structure to be worked through in a given period of time.

Fig. 7.6 EDP depiction
In both cases, the degree of automation regarding system support plays a decisive role. The first thing to do thus is to identify the MEDs in an organisation, then document them as EDPs and finally draw conclusions in the direction of optimisation measures (Fig. 7.9). When determining critical MEDs, one has to distinguish (according to scenarios “new” or “old”):

- newly added MEDs,
- MEDs to be modified and
- MEDs to remain as they are.

All three types can be subject to optimisation measures. When classifying, there are two criteria to be observed:
Sorted according to work load, the following definition within an ABC analysis can be established:

- high (A)
- medium (B)
- low (C).

The weighting according to priorities is similar:

- existential (priority 1)
- important (priority 2)
- possibly dispensable (priority 3).

Table 7.2 shows some examples for MEDs for specific business activities.

Altogether, the MEDs can be documented in a master list (Fig. 7.10). In this list, they will be assigned an attribute combined from the results of the ABC analysis and the priorities (for example, B2).

7.2.1.4 Target Process Model

At the end of all the efforts, an overall process model is generated reflecting the new organisational world. This is preceded—as already mentioned—by iterative measures, such that the overall approach can be depicted as in Fig. 7.11.

This means that after the identification of MEDs and their EDP type compilation, an initial overall model has to be created. After executing different successive steps, one eventually arrives at a final target process model (Fig. 7.12),

<table>
<thead>
<tr>
<th>MED</th>
<th>Expense</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract management</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>Contract statistics</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>Billing</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
which will be binding for the future. But not only the MEDs do belong to this model, but also support processes, which have up to then not been subject to the optimisation considerations.
7.3 Process Documentation

7.3.1 Types of Documentation

One has to differentiate between documents, which are produced and required before, during and after a project. To these belong so-called reference documents, required before and during a measurement phase to create a common understanding for all parties involved. They consist of a mix of technical and process documentation. Table 7.3 shows a possible structure for a document directory.

Meaning of the items:

- cont. no.: sequential
- file name: incl. path
- author: responsible person (not necessarily the user, who has saved the latest version)
- document: full title, for example
  - acceptance directive
  - specification for …
  - test script for…
  - functional requirements for …
  - test schedule …
- version: Id. of approved version (according to ISO)
- date: date of approval.

Besides these, there are documents, which are produced during measurements reporting test results. Those may be, for example:

- readiness protocol
- problem store
- protocol of results.

And finally, there is the complete set of system documentation. The documentation to be delivered with the software is normally exhaustively listed in the implementation contract. It can be delivered on different media:

- paper (rarely)
- on data storage medium
- electronically with the software (online)
- electronically per link

<table>
<thead>
<tr>
<th>No.</th>
<th>File Name</th>
<th>Author</th>
<th>Title</th>
<th>Version</th>
<th>Date</th>
</tr>
</thead>
</table>
• or on more than one of these media.

This documentation normally comprises:

• service description:
  – depiction of business processes to be supported by the production software,

• production schedule:
  – sequential scheduling of jobs with their parameters and associated configuration requirements necessary for execution,

• inventory document, regarding:
  – software
  – database system
  – jobs
  – utility programs, etc.,

• operating concept:
  – as a supplement to the production schedule: data backup strategy, logging, support, etc.,

• user documentation:
  – work instructions structured according to the business responsibility of the staffer, in most cases as online help,

• customising documentation:
  – system and functional settings and default values for data items.

All these documents should hold the latest approval status prior to a measurement.

It is important that during preparation, all parties concerned agree, which of the documents are relevant to acceptance.

### 7.3.2 Integrity and Relevance

#### 7.3.2.1 Integrity

There are documents, which have to be ready at all costs before measuring and test begin, and others, that can be handed in later. The first comprise imperatively:

• approved functional specifications,
• approved technical specifications,
• test scripts,
7.3 Process Documentation

- readiness protocol.

To the latter belong:

- training documents
- process descriptions and others.

Of course, it is desirable to have all these documents in advance. Sometimes, this is not possible because of time constraints. Evidently, such documents that are generated during or after measurements (problem repository, acceptance protocol, etc.) cannot be provided before hand. All the software documentation will be delivered at the earliest after provision in any case, if new applications are concerned.

7.3.2.2 Relevance

One has to differentiate between technical and legally relevant documents. Surely any declaration of acceptance with all appendices (acceptance protocol, patch schedule…) belongs to the legal ones and is therefore relevant for the subsequent billing process. All other documents contain technical details enabling acceptance and production in the first place. However, their importance can be established from the fact that, for example, the whole system documentation is an acceptance object in its own right.

7.3.3 Responsibilities

Depending on the type of document, different responsibilities have to be assigned (Table 7.4).

Acceptance coordination has to take care that these documents are readily available in time and have been approved during the versioning process.

7.3.4 Versioning and Approval

The compilation of documents that do not have operative relevance (like a problem store or an acceptance declaration), but describe content should be subject to a versioning process. The relevant number coding could look like this:

<table>
<thead>
<tr>
<th>Table 7.4</th>
<th>Documents and responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional specification</td>
<td>Business units</td>
</tr>
<tr>
<td>Technical specification</td>
<td>Programmers</td>
</tr>
<tr>
<td>Work sheets for measurements</td>
<td>Acceptance coordination</td>
</tr>
<tr>
<td>Test scripts</td>
<td>Business units</td>
</tr>
<tr>
<td>System documentation</td>
<td>Contractor</td>
</tr>
</tbody>
</table>
x.y, where
• x: version number and
• y: reference to approval status.

The approval status could be coded as follows:
• 0: draft
• 1: in clearance
• 2: released

Refinements are possible.

During the clearance process, comments and addenda from the different parties concerned will be collected supplementing the next version. This whole document history is normally recorded at the end of the document in a table (Table 7.5), so that the evolution of the document can be traced back at a later stage. Sometimes, controversies about the latest valid status can arise, when during approval or distribution mistakes have been made. The history is a sure instrument to alleviate such doubts.

The already mentioned existing process documentation has to be adapted to the new environment. If there does not exist any documentation, it has to be created from scratch. For this, the EDP and MED have to be taken into account systematically. From those, training documents will be generated later. Work instructions describe the handling of the new or modified application software by the end-user.

Furthermore, documents describing production, the handling of interfaces and the launching of reports have to be drafted. Of importance is also the whole process of error handling: support levels and escalation paths including contact persons and telephone numbers. All these documents are part of the overall system package.

### 7.4 Process Tests for New Applications

Everything discussed until now about interface tests and simulations was done with a purely technical view on the problems. The question is, do interfaces function flawlessly by transmitting a given number of data records in a given format with acceptable performance? This means that all interfaces in a system
environment must be tested successively. Once no errors occur, there is no technolog-
nical reason why acceptance should not be granted. This view is defended in
most cases by developers and software suppliers. However, in addition, perform-
ance aspects can become relevant.

For the daily life of applications, correct functioning is assumed to happen. But
the deployment of interfaces is transparent to the user, however. His interests rest
with a functioning process—and specifically that part of a process, for which he is
responsible in his daily business. In this sense, he is interested in the result
regarding the business content at the end of a process chain. An in-depth test
methodology has to take that into account.

For measurements, this means integrative testing (Fig. 7.13). Concerning test
cases, all systems relevant to a particular process part (down to work instructions)
with all interfaces between them have to be included. Only after the complete
chain has been executed with positive results, the process can be accepted.
“Positive” in this case also means “with satisfactory performance”. Individual
investigations of interfaces are only then of relevance, when a specific problem
during the tests has popped up.

The reason for possible discrepancies concerning interface performance during
a process test in comparison with individual interface tests can be found in the
selection for acceptance criteria by the user community. A flawless transmission of
clean data without any deviation is not necessarily sufficient:

- The relations between data items have to be right.
- Plausibility checks have to function properly.
- Under certain circumstances specified items must be masked.
- Performance criteria have to be met.

Fig. 7.13 Integrated tests
7.4.1 Availability of Partner Systems

Integration tests determine the interplay of all systems taking part in a single process. This means:

- provision of test systems of all partners concerned,
- availability of all test systems,
- willingness to participate,
- tying in of all concerned into the measurement process,
- common error tracking,
- all over system support,
- support from all user departments involved.

At the same time, it has to be ensured that the release versions on all partner test systems correspond to those of their production environments. To arrive even at these basic conditions, there is not only planning effort required but also quite often the settling of political conflicts. In the end, this entails the secondment of test resources serving, for example, the possible advantage of a separate department in an organisation. In this case, one cannot always count on comprehension because quite often restricted budgets reserve members of staff for other tasks. Another difficulty is the technical availability of systems, because in other parts of the organisation, their system updates or tests may be running at the same time as well. For this reason, it is advantageous to control the organisation of integration tests from a high-level close to top management.

7.4.2 Test Scripts and Test Data

Test scripts are fed from three sources:

- functional specifications
- technical specifications
- business processes (EDP).

All three depend on each other. One would assume that functional specifications would be built upon documented business processes. These in turn are the basis for the technical specifications written by developers. This is the theory. Only when these preconditions have been fulfilled test scripts can finally be developed on the sole basis of business transactions. Unfortunately only in very few companies, one can find a complete and up-to-date documentation of business processes. This is even less the case for single functions, which have possibly been developed from ad hoc insights over and above the usage of existing software. Therefore, one has to assume a rather heterogenous document history, when suddenly tests are requested under time pressure.
The supplier always will refer to the latest approved status of the technical specification in question for his invoicing. From a legal point of view, this is completely okay. But, the end-user will give his approval only, when in deed, the new functionality supports his business process in such a way that he considers proper and efficient. This demand has to have priority. To avoid conflict at this stage, all parties concerned should be consulted. Compromises have to be agreed upon, if in spite of all communication efforts, disagreement persists after technical specifications have been drafted.

In no case should the customer be satisfied to accept test scripts from the development tests of the supplier only, because they are often purely technically oriented with little consideration for business processes. It is important to insist that tests should proceed on the business working level. This corresponds to the hierarchy in Fig. 7.14.

Figure 7.14 shows a basic schema. Depending on the complexity of the main process, the hierarchy can be deeper or flatter. Decisive for the drafting of test scripts is the lowest level (working level; work instructions).

When defining test data, the requirements of the test scripts have to be taken into account of course—especially concerning the combination of data item variants. From these, the demands on content are deduced. It has to be checked, whether these can be satisfied by a copy of the production database. In this case, it has to be decided, whether partial copies or the complete productive database have to be reproduced. In this case, the capacity limits of the test system and the expected performance with large data volumes have to be respected.

When the transactions under investigation create new data or present new algorithms, real data often do not suffice. In these cases, synthetic data have to be created—often into an empty database. If such requirements exceed more than some dozen data records, one has to allow for expenses to build up complex and numerous synthetic data. If necessary, large numbers of synthetic data have to be generated by programs especially written for this purpose.

---

**The decomposition of a business process xyz follows this hierarchy:**

```
main process
  ▶️ sub process level 1
    ▶️ sub process level 2
      ▶️ work instruction
```

The level "work instruction" allows for variants with respect to the variables "initial situation" and "input". The combination of these three items can be identified uniquely by a combination of numbers following the process hierarchy.

**Acceptance Criteria**

For each identified test case there are acceptance criteria, which can in turn be differentiated according to:

- expected output
- objective (with respect to the work instruction)

---

Fig. 7.14 Process hierarchy
Of course, performance considerations bring along their own aspects concerning the selection of test data. These aspects gain in importance as soon as changes to the data model had been implemented.

### 7.4.3 Organisation of Tests

The whole test scenario is based on two pillars:

- decomposition of a common process xyz into uniquely defined and identifiable test cases,
- implementation of these test cases in functional system compliant test process steps within a specific configuration including all relevant test data.

#### 7.4.3.1 Decomposition of Business Processes

The decomposition of a business process xyz follows this hierarchy:

- main process
  - sub process level 1
    - sub process level 2
      - work instruction

The level “work instruction” allows for variants with respect to the variables “initial situation” and “input”. The combination of these three items can be identified uniquely by a combination of numbers following the process hierarchy.

#### 7.4.3.2 Acceptance Criteria

For each identified test case, there are acceptance criteria, which can in turn be differentiated according to:

- expected output,
- objective (with respect to the work instruction),
- performance criteria (benchmarking).
7.4.3.3 References

Every test case with its acceptance criteria is to be documented in a test case catalogue. This catalogue contains also the assessment matrices for each test case, holding the following information:

- relevance
- test area
- test result
- comment
- tester and
- tracker ID.

7.4.3.4 Test Process Steps

During the design phase of distinct test process steps by the specialist teams comprising developers and test case designers, the supplier makes his test results from his development phase available. The test process steps are documented in a database as follows:

> Identification

ID:
Short text:

> Splitting of a particular test case into test process steps

Step 1:
> Description of test step

Step 2:
> Description of test step

Description of test data
...

Mapping of the software test sequence
...

Expected result
...

Description of test infrastructure
(Reference to infrastructure description to be issued at the first instance only and then stored in a central directory, thereafter only referred to)

Test execution
Step 1:
Date:
Result: for example, “the contract version had been created correctly”

> Protocol
7.4.3.5 Integrative Test Scripts

One of the essential preconditions, as already mentioned, is the preparation of integrative test scripts. Normally, those are developed by the testers of the proper systems concerned and presented to the partner systems for comparison and agreement. Figure 7.15 shows the schema of such a test script.

When working through such a script, it is important to keep dates and hours to be able to hold resources (tester and systems) in readiness at the exact point in time agreed. However, there are limits with respect to planning. If during the test sequence in one or another system or at an interface a problem occurs an unplanned delay will result. Causes can be many:

- configuration error
- customising problem
- interface error
- software error in system A
- software error in system B
- long execution times.

Sometimes communication problems between operators suffice to bring the sequence to a halt. Delays and their dissolution depend on the type of problem. In any case, a delay causes replanning in the short term with all the associated difficulties with regard to resource readiness.

Objective of integrated acceptance is not the acceptance of a system but the acceptance of a complete process including performance aspects. For this reason as well, the signatures of all parties concerned from the departments in question are required on the acceptance protocol.

### Fig. 7.15  Script for integration tests

<table>
<thead>
<tr>
<th>No.</th>
<th>System</th>
<th>Description of Test Case</th>
<th>Variant</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System1</td>
<td>functional test 1</td>
<td>x1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>System1</td>
<td>functional test 2, export, import, functional test 3</td>
<td>x1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>System2</td>
<td>functional test 4</td>
<td>x1</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 8
Best Practise Examples

As already indicated in Chap. 6, the market offers an abundance of performance tools of different quality. Of course, all system suppliers have their dedicated optimisation tools and monitors available tailored to their own products. Furthermore, there are performance-relevant system commands and utilities as part of the operating systems themselves. In this chapter, we will present two such tools as examples and discuss performance facilities offered by two different operating systems:

- Hewlett&Packard: LoadRunner
- Microsoft: SQL Server 2008
- UNIX systems and
- Windows: TuneUp.

8.1 Hewlett&Packard: LoadRunner

HP LoadRunner [1] is especially useful to identify possible bottlenecks in an application or a system prior to putting it into service. In this way, potentially unavoidable system upgrades can be implemented already in advance. From a single point of control, end-to-end performance is measured and diagnosed. These are the salient features:

8.1.1 Operating Load

LoadRunner is able to simulate hundreds or thousands of virtual users—on site or in the cloud—executing different business transactions under future production conditions. In this way, bottlenecks and scalability problems can be discovered before they arise or during real production. Hence, future downtimes and performance problems are avoided.
8.1.2 Comprehensive Coverage of Technical Environments

The test environment supports many protocols and platforms. They comprise

- Web 2.0
- J2EE
- .NET
- XML
- SAP
- Oracle
- Microsoft Remote Desktop
- wireless Citrix and
- client/server applications.

Even when client–server applications have been migrated to Web or to JAVA, the same tool can be continued to be used. That makes renewed training of users unnecessary.

8.1.3 Implementation

The tool is operated by specialists mastering the following instruments:

- Visual scripting language,
- Data and autocorrelation wizards and
- ActiveScreen technology.

Among other things, scripts are generated at interface level by screen clicks.

8.1.4 Service-Level Objectives

There is a possibility to define service-level objectives (SLOs) for transactions as part of a stress test. This enables to identify very quickly those transactions and tests that will not meet essential business requirements.

8.1.5 Monitoring

The package includes HP SiteScope. This module offers real-time monitors capable of delivering detailed measured data of all system components under load. These include:
8.1 Hewlett-Packard: LoadRunner

- web server
- application server
- databases
- enterprise resource planning (ERP)
- Siebel
- CRM systems and
- firewalls.

8.1.6 Analysis

HP LoadRunner possesses analytical capabilities permitting automatic processing of all monitoring and diagnostic data. As a result, the main causes for performance bottlenecks are reported as well as those transactions which do not meet the requirements for business processes. This emerges from—among other things—individual analyses of end-users, regarding performance on system or code levels. Following the corresponding optimisation measures, further iterative measuring cycles should be applied.

HP LoadRunner offers some performance charts, including:

- “Percentile values” analyse the number of transactions in percentage, which are carried out within a defined time period.
- “Performance under stress” shows transaction times relative to the number of users at a given point in time as part of a chosen scenario.
- “Transaction performance” shows the average time required to execute transactions during each second of the current scenario.
- “Transaction performance summary” shows minimum, maximum and average of performance times for all transactions within a chosen scenario.
- “Transaction performance per user” shows performance times of a single user for a given scenario.
- “Transaction distribution” shows transaction performance over time.

Furthermore, HP LoadRunner offers two types of web analysis reports:

- “Connections per second” show the number of user connections to the Web during each second of a specific scenario.
- “Throughput” shows throughput per second on the server.

The analysis modules of HP LoadRunner establish correlations between the individual results as outlined above aiding to investigate user processes. The necessary charts to help solve performance problems can be customised individually.
8.1.7 Interface to HP Diagnostics

With the help of an interface to HP diagnostics, those transactions being chiefly responsible for application bottlenecks can be identified together with the corresponding main memory problems.

8.1.8 Scalability

Scalability is an important instrument to simulate real-operating conditions. This HP tool requires only few CPU and main memory resources per virtual user, so that realistic scenarios become possible.

8.2 Microsoft: SQL Server 2008

As mentioned earlier, operating systems themselves offer many features to aid in performance optimisations. One example is the SQL Server 2008 from Microsoft discussed in this section. Most of the important aspects are taken from the book “SQL Server 2008 Performance-Optimierung” by Holger Schmeling [3].

8.2.1 General

The operating system Windows© itself provides a system monitor offering a number of charts, which will not be discussed here. But the SQL Server has specific commands, which can trigger performance measurements. Here are some examples:

- **SET STATISTICS ON/OFF**
  - This command triggers the logging of I/Os. In addition, one can get the corresponding elapsed times by
  - **SET STATISTICS TIME ON.**
  - Its own proper SQL Server activity monitor provides the following views:

  - processes,
  - resource waiting instances,
  - data file I/Os and
  - a history of resource-intensive queries.
8.2.2 *The Profiler*

The SQL Server profiler is based on a similar philosophy as the logging of events, since it permits a selection of certain events and data to be measured. But it can do even more when supervising server activities. At its core is the so-called flow tracking. By this, an administrator creates a specific flow, which determines which events and data should be tracked and documented. These flows can be constructed individually, but there are also some standard workflows like protocols, etc. available:

- start of stored procedures
- log on/log off
- start of queries
- end of queries
- user information
- blocking and locking and others.

Besides many others, here are some more events, which can be included in a workflow:

- user error message
- deadlock graph
- scan started
- scan stopped

as well as events controlling the execution of stored procedures and batches.

The results can be edited selectively into table format. The following columns present one possible example:

- event description (text)
- StartTime
- EndTime
- Duration
- SPID (process ID)
- Reads
- Writes
- CPU (time)
- database name
- username.

8.2.3 *Performance Indicators*

The SQL Server adds another 800 performance indicators to the already existing Windows© system monitor. The description of these indicators can be found mainly in the online documentation. One has to distinguish performance indicators for:
the operating system and
the SQL Server.

8.2.3.1 Performance Indicators for the Operating System

The performance indicators for the operating system comprise:

- utilisation of the overall disk array
- processor time
- paging/second
- memory available
- length of the processor waiting queue.

8.2.3.2 SQL Performance Indicators

They comprise:

- user connections
- pending protocols about closed transactions
- transactions per second
- cache hit rate
- expected dwell time of pages in cache
- batch calls per second
- information about SQL compiles.

8.2.4 Other Monitoring and Reports

SQL Server 2008 offers additional tools, which, however, concern mainly database management and SQL queries. Some of them are of general interest:

- dynamic administration views with monitoring results, regarding
  - actual activities
  - I/Os
  - waiting states occurred and of course
  - SQL Server performance indicators
- statistical system functions
- server dashboard (total overview over performance data)
- batch execution statistics
- database usage
• data carrier usage
• query statistics.

8.3 UNIX Systems

UNIX operating systems provide performance features as well. The most important aspects of these have been taken from the book “System Performance Tuning” by Mike Loukides [4]. Here, only the monitor functions provided by the operating system shall be discussed. The measures that have to be taken after an analysis follow the basic philosophy of his book, taking into account UNIX-specific optimisation possibilities. Furthermore, in the following discussion, it is not differentiated between BSD, System V, XENIX and other variants.

8.3.1 Overview

There are a number of routines available, which can be deployed to control system activities. They comprise:
• cron: a utility that allows to initiate timer-driven command scripts—among others also those that control system supervision,
• uptime: report about average system load,
• ps: summary about running processes,
• iostat: report about disk utilisation,
• sar: CPU utilisation,
• sa: report about command and program calls,
• perfmeter: various statistics.

8.3.2 Average System Load

Uptime can be executed manually and shows each time the following information:
• total execution time of a system,
• mean system load during the past minute,
• mean system load during the past 5 min. and,
• mean system load during the past quarter of an hour as table entries.

Mean system load is defined as the average number of processes in the CPU waiting queue per unit time. The results reported give indications about possible problems regarding, for example, the CPU, but do not really express precisely their nature, so that from such indications, investigations have to proceed further and deeper.
### 8.3.3 Process Overview

*ps* enables the selection of additional parameters, with the help of which information can be detailed further. One example is Table 8.1 with the following data:

- **USER**
- **PID**: process ID
- **%CPU**: portion of overall CPU usage of the process
- **%MEM**: portion of main memory usage of the process
- **SZ**: size of virtual memory relevant to the process
- **RSS**: size of process part resident in main memory
- **TT**: terminal device, from which the process was launched
- **STAT**: process state with
  - R executable
  - T on hold
  - P paging
  - D I/O wait
  - S idle since less than 20 s
  - I idle since more than 20 s
  - Z closed
  - W removed because of surpassing the memory limit
  - N priority adjusted to a lower value.
- **TIME**: total CPU time up to now
- **COMMAND**: command or program.

### 8.3.4 Other Reports

Here are two further examples, which can be helpful in the performance analysis of UNIX systems. They are:

- performance overview and
- user charging.

#### 8.3.4.1 Performance Overview

Just as all the other utilities, *perfmeter* can be parameterised in different ways. For graphical displays, there are the following options:
-s time: measuring interval
-M start (max, min): scale of the display
-s param: system parameters to be displayed:
  - CPU utilisation
  - page paging per second
  - swap swaps per second
  - intr interrupts per second caused by I/Os
  - disk disk accesses per second
  - cntx process alternations per second
  - load executable processes during the past minute

host: computer ID.

8.3.4.2 User Charging

There are two types of statistics of interest:

- user profile and
- command or program summary.

As already expressed by its notation, the evaluation “user profile” deals with user-related information. It contains the following variables:

- UID
- LOGIN NAME
- CPU (MINS) (total time of reported period)
- KCORE-MINS [memory time in (kilobyte minutes)]
- CONNECT (MINS) session duration
- DISK BLOCKS (disk occupancy)
- # OF PROCS (number of processes launched)
- # OF SESS (sessions per day).

Commands under UNIX may also mean program calls. Finally, therefore, the following evaluation is possible with respect to a specific program (Table 8.2). The items mean:

COMMAND NAME program name
NUMBER CMDS number of calls per unit time

<table>
<thead>
<tr>
<th>COMMAND NAME</th>
<th>NUMBER CMDS</th>
<th>TOTAL</th>
<th>TOTAL MEAN</th>
<th>MEAN</th>
<th>HOG</th>
<th>CHARS</th>
<th>TRANSFD</th>
<th>BLOCKS</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND NAME</td>
<td>NUMBER CMDS</td>
<td>TOTAL</td>
<td>TOTAL MEAN</td>
<td>MEAN</td>
<td>HOG</td>
<td>CHARS</td>
<td>TRANSFD</td>
<td>BLOCKS</td>
<td>READ</td>
</tr>
<tr>
<td>NAME</td>
<td>CMDS</td>
<td>KCORE</td>
<td>MIN CPU</td>
<td>REAL</td>
<td>SIZE</td>
<td>CPU</td>
<td>FACTOR</td>
<td>TRANSFD</td>
<td>READ</td>
</tr>
</tbody>
</table>

Table 8.2 Command report
TOTAL KCOREMIN overall memory time for all calls
TOTAL CPU accumulated CPU time
TOTAL REAL total elapsed time
MEAN SIZE average memory required
MEAN CPU average CPU required
HOG FACTOR ratio of CPU to elapsed time
CHARS TRNSFD total number of characters transferred
BLOCKS READ total number of read or written blocks.

8.4 Windows: TuneUp

TuneUp is a product of TuneUp Company in Germany. The package can be deployed on Windows© computers to measure and optimise performance. In addition to the mere performance modules, it offers a collection of other useful tools for the administration and maintenance of the operating system. The following description is based on the extensive online documentation of the product.

8.4.1 Overview of All Modules

The following list gives an overview of TuneUp functions. The main modules offer more detailed features in turn (Fig. 8.1):

- Hard disk
  - Defragment
  - Check for errors
  - Securely delete data
  - Restore deleted files.

- Disk space
  - Gain disk space
  - Find and delete large amounts of data.

- Registry
  - Defragment
  - Clean
  - Edit.

- Computer
  - Turn on economy mode
  - Standard mode
– Turn on turbo mode
– Start 1-click maintenance
– Configure automatic maintenance
– Show system information.

• Recommendations
  – Increase performance
  – Fix problems
  – Optimise system start-up and shutdown.

• Windows
  – Fix common problems
  – Modify Windows settings
  – Change Windows appearance
  – Remove broken short cuts
  – Display and close running processes.

• TuneUp utilities
  – Undo changes
  – Check for updates
8.4.2 Selected Improvement Tools

The functions belonging to performance improvements are real optimisers. They offer the opportunity to normal users to improve the performance capabilities of their computers in a simple and comfortable way.

8.4.2.1 Hard Disk Defragment

Defragment is a typical defragmentation tool. Fragmentation is a common problem and leads to the splitting up of files, which are subject to regular updates. When defragmenting, storage space on a disk is organised in such a way that file fragments belonging together are again physically compounded. This shortens access times. Defragment takes care of those files which are frequently called and are located in first position on the hard disk.

8.4.2.2 Registry Defragment

In the registry of the computer, entries for all installed applications and for all hardware are located. During the lifetime of a computer, applications may be deleted, new ones installed, updates uploaded, etc. Because of this, the registry degrades over time—particularly because references no longer used after de-installation will still remain in the registry, even if they are no longer referred to. A situation comparable with a fragmented hard disk arises, leading to performance degradation. Registry defragment condenses the registry back to usable size.

8.4.2.3 Optimise System Start-up and ShutDown

A common performance problem with PCs becomes apparent immediately when starting the machine: sometimes it takes unreasonably long until a user gains control over the system. The reason is a long chain of initial programs being executed automatically during boot-up (virus scanner, communication modules, screen layouts, etc.). These are programs, which are triggered either from the auto start file or from the registry itself. Auto start programs can be administrated directly. This is not so simple for the registry. But TuneUp provides the facilities to do this. A list of all programs being launched automatically at system start-up is displayed. From this, those programs can be selected to be excluded in the future.
8.4 Windows: TuneUp

8.4.2.4 Show System Information

With the help of this facility, the following features can be displayed:

- overview
- Windows information
- display device information
- memory usage
- memory performance
- drives
- I/O devices
- communication
- system devices.

8.4.2.5 Gain Disk Space

Besides essential files and programs on the computer, there are two categories, occupying unnecessary storage space (Fig. 8.2):

![TuneUp Utilities 2012](image)

**Fig. 8.2** Gain disk space
• files and backups no longer in use,
• dispensable Windows® functions or programs.

Here, TuneUp can intervene.

8.4.2.6 Find and Delete Large Amounts of Data

This function brings transparency to the utilisation of a hard disk. Just because storage capacities have become so cheap, nobody cares anymore about the “dead wood”. Temporary files, graphics and pictures with high resolution populate the disks together with large amounts of historical or discarded drafts, obsolete gadgets or entertainment media.

TuneUp helps to discover these performance-inhibiting pieces of information and files and take appropriate measures: transferring or deleting.

8.4.3 Problem Solving

The functions used for problem solving have only limited impact on performance (Fig. 8.3). Problems discussed here, however, could have been created during performance optimisation actions.

8.4.3.1 Fix Problems

While transferring files from external media such as USB sticks or CDs as well as from the Internet to the file management systems FAT or NTFS of Microsoft®, errors may occur, which can inhibit later accesses to these files. Then, corresponding error messages will appear on the screen. This happens quite regularly, when, as a result of software updates, previous compatibility no longer exists.

The fix problem function provides for a thorough investigation of hard disks and files and repairs such errors automatically.

8.4.3.2 Short cut Cleaner

This function searches the system for invalid short cuts and references and deletes them. After this, error messages are avoided, when calling such references. Areas investigated can be:

• start menu
• desktop
• quick launch bar
• last used
• history lists.

8.4.3.3 Restore Deleted Files

After cleaning up the system to optimise performance or on other occasions, it may happen that inadvertently files have been deleted, which are needed again at a later stage. If no backup of the files in question exists, there will be a problem. The Undelete TuneUp module allows in most cases to reactivate deleted files.

8.4.4 System Maintenance

To maintain a system on a regular basis, two instruments are provided (Fig. 8.4).
8.4.4.1 1-Click Maintenance

1-click maintenance can be started manually by a timer in the background. This maintenance routinely checks the following information relevant to performance:

- registry entries
- superfluous files
- degree of fragmentation.

If not already set up automatically, the results are reported after a maintenance run for system optimisation, so that all proposed maintenance tasks can be carried out. Thereafter, a short report on the results about, for example, gained storage space will be displayed.

8.4.4.2 Registry Cleaner

This is a module in its own right. It cleans the registry of superfluous entries, which degrade the performance of the system.
8.4.5 Other Tools

Other tools relevant to optimisation and tuning include (Fig. 8.5):

- Uninstall Manager: displays all applications for selection of unwanted de-installation candidates.
- Process Manager: displays all running applications at a time, to be able to interrupt some of them, also the associated files as well as main memory and processor utilisation.
- Registry Editor: for a dedicated change in registry entries.
- Securely Delete Data: to delete data once and for all, which cannot be reconstituted afterwards.
- System Information: extensive overview of hardware and software.

8.4.6 Customising Windows ©

For the sake of completeness, two additional modules should be mentioned, which facilitate customising the Windows© operating system itself (Fig. 8.6).
8.4.7 TuneUp Utilities

These utilities work as a kind of rescue centre. Precisely, when implementing measures to improve performance, they can play an important role (Fig. 8.7). They allow to

- undo changes
- restore system settings and
- control all changes that have been implemented by other modules of TuneUp to the system and, if necessary, revoke them.
Finally, the “update check” takes care that either manually or automatically via a timer function the TuneUp package is supplied with the most recent updates by downloading them from the Internet.

Fig. 8.7 Rescue centre
Chapter 9
Limits and Overall Application Context

It is one thing to measure the performance of computers and networks, to draw up elaborate lists of recommendations and quite another to implement these recommendations realistically. Implementation has to be seen within the constraints of the user environment.

The machines in question do mostly not operate in academic or laboratory type environments, are no test sites or supplier prototypes, but have been bought and made to run for particular application purposes: stock control, money transfer, accounting, invoicing, factory floor control, etc. Thus, they form part of a vital infrastructure for the functioning of enterprises and administrations. They sometimes are so vital as to be irreplaceable for certain functions—either because the expertise and the tools to execute the same tasks manually have been lost or have never been practiced before or the manpower necessary to do this is no longer available. This all leads to the constraint of minimum interruption of machine and application availability—even for tuning measures.

As a consequence application requirements lead to constraints and may limit the menu of recommendations from the point of view of practicality. But, what can be a negative constraint can be used in different circumstances to take pressure of a machine. By streamlining user functions and office information flow, function frequencies can be reduced and thus load on the system. With this, we enter the domain of organisation of user departments. It is also the mapping of information strategy onto the real application world, which is put into question. In the end, a double compromise may be asked for:

- restricting tuning actions to practical measures and
- reviewing user functions within the organisation to supplement tuning measures.

This leads to the overall interplay between the structural organisation of a company, the general information flow and the software and hardware configuration supporting it. Within a project of more ambitious size and designation, these components cannot really be decoupled. From whatever end one starts—be it organisation or be it tools—both approaches are bound to meet sooner or later because they are intertwined. This is why really efficient solutions concerning installations above a certain size have to take both aspects into account.
Performance aspects and subsequent tuning, however, have to be considered within a wider context of applications and related investigations and studies concerning the subject of computing centres themselves. Related and complementary projects and activities concern:

- **general audit of computing centre:**
  This refers to the investigation of the whole relevant infrastructure, like the site itself, access security, operating ergonomics, air conditioning, electromagnetic shielding, dust levels, power supply, IT personnel (number, structure, qualification), etc.

- **IT long-term planning:**
  Reference is made to a strategic document for general management, dealing with an investment plan over a 3–5-year period and comprising application backlog, implementation plans, priorities, associated hardware, responsibilities, standard software, tools, methods and costs.

- **information strategy:**
  Information strategy can only go hand-in-hand with company medium and long-term strategy itself and comprises:
  - reporting requirements for management
  - analytical tools
  - decision-making support
  - link-ups and cloud computing.
  Related are information strategy audit and replacement strategy. Quite often investment in soft and hardware do not deliver a reasonable or any ROI or break-even. The obvious thing to do is to compare a company’s actual information needs with past projections and investment done and—if required—develop a suitable alternative to existing systems.

- **communications strategy:**
  This concerns the deployment of WLAN, mobile phones, PDAs, VoIP and other technologies and their security.

- **data security:**
  In the face of the proliferation of hackers and viruses, this subject is becoming more and more critical for the strategic, financial and confidential information systems of an organisation. Data access security needs auditing and concepts for improvement.

- **backup strategy:**
  This is important in case of a major system breakdown or destruction (water, fire, etc.). If no fault-tolerant or backup systems are available or if both destroyed as well, suitable strategies for external backup arrangements have to be developed (mobile systems, computer centre sharing, etc.). Parallel to such concepts, a general evaluation of hardware and software value and assets as well as a tag on replacement costs has to be done. This is quite often relevant for proper insurance cover as well.
The above list is probably far from complete but gives a good impression of the ad hoc and ongoing tasks relevant to major information system installations. In most cases, pressure on time and resources exceeds the capabilities of IT departments. So generally, the advice of external IT specialists has to be sought and bought. Justification is sometimes difficult but almost always calculable. And—quite generally speaking—a smoothly running IT production shop just has to be better than the improvised type of wizardry still to be found in quite a few places.
To prepare and follow through performance projects, the following nine checklists can be useful. They cover the following project phases:

- Basic Considerations (Table 10.1).
- Taking Stock (Table 10.2).
- Master Plan (Table 10.3).
- Test Scripts (Table 10.4).
- Static Data (Table 10.5).
- Dynamic Data (Table 10.6).
- Evaluation (Table 10.7).
- Implementation Plan (Table 10.8).
- Boundary Conditions (Table 10.9).

### Table 10.1 Basic considerations

<table>
<thead>
<tr>
<th>Problem</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are there performance problems with your current IT environment?</td>
<td>Feedback from application users and from operating</td>
</tr>
<tr>
<td>Are the causes of these problems known?</td>
<td>Recurring problems with regard to specific functions/transactions at certain times</td>
</tr>
<tr>
<td>Are performance tests required because new software has to be introduced?</td>
<td>Prophylaxis</td>
</tr>
<tr>
<td>Are performance test required because existing applications will be upgraded?</td>
<td>Prophylaxis</td>
</tr>
<tr>
<td>Which applications are concerned?</td>
<td>If known</td>
</tr>
<tr>
<td>Is there experience with past performance optimisation?</td>
<td>Routine measurements, optimisations carried out</td>
</tr>
<tr>
<td>Have tuning measures been implemented, and should performance be evaluated again?</td>
<td>Iterative procedure</td>
</tr>
<tr>
<td>Which performance tools are available?</td>
<td>Already bought, as test versions, to be bought</td>
</tr>
<tr>
<td>Is existing personnel sufficiently qualified to carry out such measurements and analyse the results?</td>
<td>If not, external specialists have to be contracted</td>
</tr>
<tr>
<td>Should external specialists be engaged for these tasks?</td>
<td>Is budget available?</td>
</tr>
<tr>
<td>In the cause of performance activities, should business processes be investigated as well?</td>
<td>In agreement with user departments</td>
</tr>
</tbody>
</table>
### Table 10.2 Taking stock

<table>
<thead>
<tr>
<th>Object</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of computer</td>
<td>Manufacturer, type</td>
</tr>
<tr>
<td>Number of servers</td>
<td>Distinct according to deployment (data, applications, operating control)</td>
</tr>
<tr>
<td>Nominal CPU power</td>
<td>Per processor, in total</td>
</tr>
<tr>
<td>Main memory size</td>
<td>Individual, in total</td>
</tr>
<tr>
<td>Peripheral hard disks</td>
<td>Type, performance (speed)</td>
</tr>
<tr>
<td>Communication ports and controllers</td>
<td>Number, type, function</td>
</tr>
<tr>
<td>Modems</td>
<td>Number, type, function</td>
</tr>
<tr>
<td>Routers</td>
<td>Number, type, function</td>
</tr>
<tr>
<td>Operating system</td>
<td>Current version, patchlevel</td>
</tr>
<tr>
<td>Gateways</td>
<td>To where? From where?</td>
</tr>
<tr>
<td>LAN</td>
<td>Type, topology</td>
</tr>
<tr>
<td>WLAN</td>
<td>Number of access points, routers, communication terminals, applications</td>
</tr>
<tr>
<td>Other protocols</td>
<td>Bluetooth, Infrared</td>
</tr>
<tr>
<td>Standard software</td>
<td>Office, Internet, tools</td>
</tr>
<tr>
<td>Data management systems</td>
<td>SAM, ISAM, DBMS, RDBMS</td>
</tr>
<tr>
<td>Applications</td>
<td>All, optionally classified with regard to strategic importance or size (no. of users, data volume)</td>
</tr>
<tr>
<td>Interfaces to other systems</td>
<td>External systems; type and use</td>
</tr>
<tr>
<td>Cloud applications</td>
<td>Which ones?</td>
</tr>
<tr>
<td>Cloud provider</td>
<td></td>
</tr>
<tr>
<td>Backup configuration</td>
<td>If existing</td>
</tr>
<tr>
<td>Peripheral components</td>
<td>Type and number</td>
</tr>
</tbody>
</table>

### Table 10.3 Master plan

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Responsible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of duration and identifiable end point of measurements</td>
<td>Project leader</td>
</tr>
<tr>
<td>Involvement of all persons and organisational units concerned during planning phase</td>
<td>Testers, application specialists, system administrator</td>
</tr>
<tr>
<td>Close cooperation with end users and all interested parties within an organisation</td>
<td>End users</td>
</tr>
<tr>
<td>Selection of performance testers early in the project</td>
<td>Testers</td>
</tr>
<tr>
<td>Assuring cooperation between performance testers and developers to create and provide for certain test cases</td>
<td>Testers, application specialists</td>
</tr>
<tr>
<td>Communicating the collected user experience concerning existing performance to testers and test management</td>
<td>Project leader, end users, testers</td>
</tr>
<tr>
<td>Establishment of feed back processes to developers and analysts</td>
<td>Application specialists, analysts</td>
</tr>
<tr>
<td>Fixing time frames during and outside normal working hours for measurements</td>
<td>Project leader</td>
</tr>
<tr>
<td>Creating test scripts</td>
<td>Testers</td>
</tr>
<tr>
<td>Sub-process level 1</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td>Sub-process level 2</td>
<td></td>
</tr>
<tr>
<td>Work instruction</td>
<td></td>
</tr>
<tr>
<td>ID No.</td>
<td></td>
</tr>
<tr>
<td>Variant</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>Acceptance criteria</td>
<td></td>
</tr>
</tbody>
</table>

**Table 10.5  Static data**

<table>
<thead>
<tr>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database schema; data management system</td>
</tr>
<tr>
<td>Cache size and address</td>
</tr>
<tr>
<td>Temporary file space allocation</td>
</tr>
<tr>
<td>Main memory size</td>
</tr>
<tr>
<td>Basic memory occupancy (paging areas)</td>
</tr>
<tr>
<td>Nominal CPU power</td>
</tr>
<tr>
<td>Operating system version</td>
</tr>
<tr>
<td>Number of peripherals</td>
</tr>
<tr>
<td>Job, session parameters</td>
</tr>
<tr>
<td>Spool parameters</td>
</tr>
<tr>
<td>Size of system tables</td>
</tr>
<tr>
<td>Virtual memory allocation and size</td>
</tr>
<tr>
<td>System buffers</td>
</tr>
<tr>
<td>Account structures</td>
</tr>
<tr>
<td>Most important files</td>
</tr>
<tr>
<td>Files, not having been accessed for some time</td>
</tr>
<tr>
<td>Table 10.6 Dynamic data</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>Object</td>
</tr>
<tr>
<td>Programs currently in use</td>
</tr>
<tr>
<td>User statistics over time</td>
</tr>
<tr>
<td>Account statistics</td>
</tr>
<tr>
<td>Processor usage</td>
</tr>
<tr>
<td>Main memory usage</td>
</tr>
<tr>
<td>Memory management</td>
</tr>
<tr>
<td>Number of parallel processes</td>
</tr>
<tr>
<td>Frequency of file accesses (open table)</td>
</tr>
<tr>
<td>Overheads and interrupts</td>
</tr>
<tr>
<td>Wait queues</td>
</tr>
<tr>
<td>System table usage</td>
</tr>
<tr>
<td>I/Os</td>
</tr>
<tr>
<td>Swap rate</td>
</tr>
<tr>
<td>Addressing of main memory segments</td>
</tr>
<tr>
<td>Critical users</td>
</tr>
<tr>
<td>CPU time</td>
</tr>
<tr>
<td>Actual execution status of programs</td>
</tr>
<tr>
<td>Actual hard disk occupancy</td>
</tr>
<tr>
<td>Processor state (active, idle)</td>
</tr>
<tr>
<td>Parallel sessions</td>
</tr>
<tr>
<td>Jobs</td>
</tr>
<tr>
<td>System logs</td>
</tr>
<tr>
<td>Frequency of standard function calls by users</td>
</tr>
<tr>
<td>Production schedule</td>
</tr>
</tbody>
</table>
Table 10.7 Evaluation

<table>
<thead>
<tr>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average CPU usage</td>
</tr>
<tr>
<td>Time-dependent CPU usage</td>
</tr>
<tr>
<td>Relative CPU load</td>
</tr>
<tr>
<td>Average number of disk accesses</td>
</tr>
<tr>
<td>Time-dependent I/O waits</td>
</tr>
<tr>
<td>Main memory management I/O activity</td>
</tr>
<tr>
<td>Average main memory management activities</td>
</tr>
<tr>
<td>Time-dependent main memory management activities</td>
</tr>
<tr>
<td>Paging I/Os</td>
</tr>
<tr>
<td>File opens (selective, over time, which ones?)</td>
</tr>
<tr>
<td>Disk occupancy</td>
</tr>
</tbody>
</table>

Problems

<table>
<thead>
<tr>
<th>Is there surplus storage capacity?</th>
</tr>
</thead>
<tbody>
<tr>
<td>How large are the overall overheads claimed solely by the system?</td>
</tr>
<tr>
<td>Distribution of main memory usage for specific applications</td>
</tr>
<tr>
<td>Which are the critical programs with respect to system load?</td>
</tr>
<tr>
<td>Has system table usage reached critical levels?</td>
</tr>
<tr>
<td>How often are utility programs called from applications?</td>
</tr>
<tr>
<td>Account occupancy</td>
</tr>
<tr>
<td>Which programming languages are in use?</td>
</tr>
<tr>
<td>Are directories and registry clean?</td>
</tr>
<tr>
<td>What is the ratio batch/online?</td>
</tr>
<tr>
<td>Are there bottlenecks with communication channels?</td>
</tr>
<tr>
<td>How recent are the versions of hardware and operating system?</td>
</tr>
</tbody>
</table>

Table 10.8 Implementation plan

<table>
<thead>
<tr>
<th>Object</th>
<th>Measure</th>
<th>Dead Line</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>De-fragmentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cache size</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applications</td>
<td>Data model</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subroutines</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GUIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Investments</td>
<td>Main memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hard disks</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Tables 10.9  Boundary conditions

<table>
<thead>
<tr>
<th>Matter</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing centre audit</td>
<td>Applications, systems, infrastructure, personnel</td>
</tr>
<tr>
<td>Long-term IT planning</td>
<td>Level: top management</td>
</tr>
<tr>
<td>Information strategy</td>
<td>Implementation of IT planning by IT responsibilities</td>
</tr>
<tr>
<td>Communication strategy</td>
<td>As IT strategy</td>
</tr>
<tr>
<td>Data security</td>
<td>Concept: malware protection, protection against external non-authorised accesses</td>
</tr>
<tr>
<td>Backup</td>
<td>Concept/investments: system mirroring, fall-back options in case of disasters</td>
</tr>
</tbody>
</table>
Performance activities, be it measurements, tests, analyses or tuning measures, can of course be executed in parallel or sporadically during daily business. For reasons of efficiency and transparency, it is recommended to set up a project of its own or a sub-project, for example, for the implementation of new software. Such a project structure could be designed as in Fig. 11.1.

This appendix will discuss some features of project management as well as project controlling [7].

11.1 Project Management

11.1.1 Technical and Organisational Tools

Systems supporting project management are known under the abbreviation PMS: Project Management System. We will discuss some of its features, including the following aspects will be

- objectives and tasks of a PMS,
- technical possibilities and functional characteristics,
- the most important basic functions,
- specific functions in detail such as
  - task planning,
  - milestone plan,
  - capacity management,
  - Gantt diagram,
  - network diagram,
  - critical path and
  - cost control.
11.1.2 Objectives and Tasks

PMS is supposed to support projects above a certain complexity to ease implementation. The aim is to achieve transparency concerning:

- project progress,
- schedule bottlenecks,
- personnel bottlenecks
- and associated costs.

All this is necessary to arrive in the end at the projects objectives. The effort to maintain PMS data should be in an economically reasonable relation with respect to the overall project effort. There are small projects, where one can do without expensive data capture and preparation, or where Excel© sheets cover all required demands. On the other hand, however, there are highly complex projects requiring appropriate system support. To handle these PMS applications, qualified staff is needed. Depending on the degree of project complexity, PMS has to be selected providing the necessary capabilities.

Besides supporting scheduling, a PMS should be capable to report about project development and disruptions, deadline transgressions, capacity bottlenecks or cost overruns. In this respects, it acts as a controlling instrument.

To cover all these requirements, ongoing status information about project progression itself has to be captured over and above the basic information identified to establish a project plan. By this, original planning can be compared to current developments. The effort for this should not be underestimated. As for all business systems, the following rule applies for PMS as well: the quality of information a system delivers depends on data quality and therefore on the input from its users. Once the decision has been taken to deploy a PMS, data maintenance should be carried out seriously. Half-heartedness leads to unnecessary effort and erroneous project documentation.
PMS generally offer a large number of possibilities to create reports for project management without having to reformat them.

11.1.3 PMS Functions

It would be out of proportion in this section to portray the whole spectrum of possibilities a PMS offers. Therefore, the following will cover only the most important PMS functions:

- administration of tasks,
- Gantt diagram,
- milestone plan,
- capacity management and
- network diagram.

11.1.3.1 Administration of Tasks

Figure 11.2 shows a typical task list such as managed by most PMS. This list must cover all relevant individual tasks in a project. It becomes obvious already at this stage that a meaningful structuring of the overall project has to take place right from the beginning. The sequence of task processing is determined by start and end dates or by task duration. It is useful to take into account the time sequence as soon as a task is entered. This will facilitate graphical presentations later.

On the left-hand side, we have a running number allocated by the system. The next row contains the task name in long text. This is followed by the start date. When dates are allocated, the system will issue a warning when a proposed date

![Fig. 11.2 Tasks](image-url)
does not exist (for example, November 31st) or when it coincides with a Sunday or a public holiday. Thereafter, the task duration can be entered, so that the end date will be calculated. But it is also possible to define the end date, and the system then calculates the task duration. However, it is not possible to fix the duration and then trying to allocate an implausible end date.

The task can then be displayed in detail (Fig. 11.3) and supplemented with additional information.

For reasons of simplification, the other rider options will not be followed through here. Basically, the detailed presentation repeats the data from the task list. In addition, a field showing the percentile of completion of that task appears. This field can be updated manually.

### 11.1.3.2 Gantt Diagram

The Gantt diagram shows in a well-arranged fashion the most important project scheduling and progression data in one single chart (Fig. 11.4):
the task itself along with its name,
• task duration,
• start date of the task and
• end date of the task.

At the same time, a configurable calendar is displayed along the x-axis at the top. If necessary, a second calendar can be shown with a different scale. In this way, months can be depicted at the top and weeks on the second line or weeks above and days below. Below this calendar, individual tasks are shown as time bars.

Tasks can be sorted hierarchically by grouping certain tasks together as so-called collective tasks as outlined in Fig. 11.5.

The collective task “Performance” shown as a black bar comprises the individual tasks “Kick-off” and “Data Capture” and so forth. The collective task takes into account the earliest start date and the latest end date of all associated individual tasks.

At the same time, the degree of completion can be displayed as in Fig. 11.6 (% completed) from the tasks processing screen.

The degree of completion is visible as a small black bar within the task bar to document project progression. Once it has been decided to maintain this data item, it obviously means extra work to be done on a regular basis.

In the presented rather simplified Gantt depictions, here quite a number of refined possibilities have been left out:
• linking of tasks,
• documentation of task owners and
• capacity reports.

Altogether the Gantt diagram is quite possibly the most often applied chart in project management.
11.1.3.3 Milestone Plan

The milestone plan shown in Fig. 11.7 is a kind of Gantt diagram subset.

Again all tasks can be seen and the milestones are depicted as black rhombuses. Such a milestone may stand at the end of a task. The task is closed, once the milestone has been reached. But a milestone can also stand alone representing a once and for all task of very short duration—for example, for a single decision to be made. Milestones are always linked to a specific date and represent objectives or sub-goals. In this case, task scheduling and its duration should be done backwards.

The system displays the date or the final date for a milestone aside it. This chart also shows links between predecessor and successor tasks.

11.1.3.4 Network Plan

Some PMS provide for network charting. Being a project control instrument, a network plan displays not only chronological but also logical dependencies between tasks within a project. Depending on project complexity, network plans may contain hundreds of different tasks. A grand overview of such a network plan on a wall shows impressively the total complexity of major projects in one stroke. Other conclusions that can be drawn from network plans are critical paths. Once the dependencies between tasks become transparent, consequences can be derived as to what may happen, if a critical task within a chain of tasks risks not to be achieved.
11.2 Project Controlling

11.2.1 Peculiarities

IT line controlling functions basically with a view on running expenditures by organisational units. They are managed in a valid cost centre chart of accounts. Generally, major investments and projects with a time limit are excluded from these procedures. For these purposes, special financial resources are provided and planned separately. Thus, these should be subject to their own controlling mechanisms—the IT project controlling.

The following aspects play a particular role in IT project controlling, transcending classical controlling:

- splitting costs between different organisational units such as
  - IT,
  - user departments,
  - project organisation itself,
  - external support,
  - overheads (quality management, acceptance tests etc.)

- dedicated instruments for the controlling process such as
  - clearance procedures,
  - task confirmations,
  - project reports and other

- differently designed controlling mechanisms: IT project controlling can advance to a project controlling instrument in its own right having influence on the overall project success.

This section will discuss these particularities in more detail.

11.2.2 Objects

11.2.2.1 Direct IT Costs

These comprise:

- rent for server environments,
- disk drives,
- peripheral terminals,
- operations and support of test systems.
11.2.2.2 Costs Relevant to User Departments

These comprise:

- drafting functional specifications,
- consulting overheads,
- training,
- test participation and
- customising.

11.2.2.3 Project Management

These comprise:

- leadership and administration,
- support,
- communication and project organisation,
- reporting.

All these costs above are generated by both internal and external services. They have to be—as far as possible—budgeted right at project start. Furthermore, a separate budget item relating to cost for outlays has to be foreseen for.

11.2.3 Process Approach

The process approach in this context means measures to control expenditures. In this sense, they transcend a purely target performance comparison in reporting. They include:

- account assignment elements,
- cost rates,
- order value procedures and
- clearance procedures.

11.2.3.1 Account Assignment Elements

All objects of IT controlling belonging to a specific project have to be integrated into an overall cost framework. To achieve this, a hierarchical classification as shown in the example in Fig. 11.8 is useful.
Such a structure permits a unique allocation of cost elements, and at the same time, a consolidation of all accumulated costs up to the highest aggregation level, which is indispensable for management.

11.2.3.2 Charge Rates

In addition to hourly or daily rates agreed upon by contract to external service providers, there are internal charge rates in many companies between different organisational units, which have to be taken into account. If user department resources are requested, quite often such services have to be negotiated or budgeted separately. To the latter belong:

- supporting training activities,
- participation in drafting functional specifications,
- participation in measurements, etc.

11.2.3.3 Order Value Procedures

To limit budget overdraft at an early stage when buying external services, which are possibly not covered by frame contracts, most purchasing systems allow to set upper limits for the value of a single order. These restrictions can be limited to predefined hierarchical levels within an organisation. In this way, already at the stage of invitation to bid by the initiator, a warning will pop up, once a predefined value risks to be surpassed.

11.2.3.4 Clearance Procedures

The clearance of budgets or budget parts is part of a combined strategy executed in unison by requesters, project management and controlling. Most systems on the
market provide professional support for such procedures. Generally, the requester enters his planned demand in the system by using the available accounting structure, the project leader checks it, and controller clears it. The necessary workflows can be configured in most systems. It is important that this is preceded by prior automated checks against budget limits, charge rates and the time schedule in question before clearance proceeds manually.

11.2.4 Reporting

Reporting for IT projects can be quite extensive. The necessary system support is provided by many PMS (Sect. 11.1) and will thus not be discussed in depth here. These are the more important reports:

- task confirmations,
- project reports and
- budget reports.

11.2.4.1 Task Confirmations

A typical template for task confirmations is shown in Fig. 11.9. Besides the time sheet itself with associated tasks, the relevant accounting assignment element and possibly the order number for external partners are of prime importance to permit a unique reference to the budget.

11.2.4.2 Project Reports

Project reports can appear in different formats. There is no universal standard. Quite often a simple extract from the PMS does not suffice. The report should contain:

- progress in substance,
- risks,
- adherence to schedule and
- cost development.

Only the latter is of importance here. One has to take care that the cost relevant part of the project report should be formatted in such a way that a simple transfer to the budget report is made possible. By appropriate consolidations, this may lead directly to a proper management report.
Concerning project controlling, the following aspects should be taken into account. They comprise:

- project progression and clearances,
- risks,
- advantage/disadvantage considerations.

Project progression depends of course—besides on technical requirements—on the financial resources still available. The multistep clearance procedure serves to avoid overruns. This may lead to conflict of interest and re-prioritisation, when
several requesters have access to the same budgetary items. At this stage, a clear budget structure becomes important.

Even long before a pressing clearance decision, a target performance comparison will show the financial risks concerning project progress. It is the controllers duty to intervene anticipatory and in time and point out emerging bottlenecks. Then, it is still time to organise new resources or reallocate some. Against this backdrop, there exists the possibility for a reasonable evaluation of the total financial risk to a project. If the warning comes too late, decisions have to be taken under duress, sometimes meaning a shortcoming of rational considerations.
References

1. HP Loadrunner Documentation: 4AA1-3944ENW, 4AA1-4227ENW, 4AA1-5197ENW, 4AA1-5263ENW, 4AA2-3934ENW, 4AA3-0545ENW, (especially in sections: 3.6 and 8.1)
2. Siemens BS2000/OSD (2009, November) Performance-Handbuch, (Illustrations are separately referenced)
Index

8
802.11, 87

A
Account, 36, 37
Analysis, 51, 55, 121
Application, 26
Application architecture, 27, 73
Application tuning, 39, 73
Audit, 140
Automation, 52

B
Backup strategy, 140
Bandwidth, 91
Batch, 14, 27
Busy, 13, 62

C
Cache, 13
Cache partition, 17
Cache size, 26, 37
Channel distribution, 91
Channel separation, 92
Charge rates, 157
Clock cycle, 17
Cloud, 45
Cloud application, 45
Cloud provider, 45
Code segment, 27, 57
Communications buffer, 37
Communications strategy, 140
Compiler, 26
Controlling, 155
Costs, 155
CPU, 10–12, 19, 23, 54
CPU overhead, 18
CPU power, 37, 80
CPU usage, 57, 62, 63
CPU utilisation, 16
CSMA/CA, 97

d
Database schema, 37
Data cache, 23
Data management, 28, 78
Data management system, 30
Data reduction, 56
Data security, 140
Data transmission, 89
Database management system (DBMS), 30, 31
Debugging, 26
Defragmentation, 22, 78, 130
Disk, 21
Disk access, 57, 63
Disk caching, 62
Disk I/O, 24
Disk occupancy, 21, 42
Disk space, 21, 131
Documentation, 109
Driver, 72
DSSS, 94
Dwell time, 10, 41
Dynamic data, 35

E
Editor, 26
Evaluation, 62
Event driven process (EDP), 105, 114
Exception, 16
Expulsion, 15, 20
**Index**

**F**
- FHSS, 93
- File distribution, 68
- File handling, 76
- Fragmentation, 22
- Function call, 74

**G**
- Gantt diagram, 152
- GUI, 10, 27, 33, 76

**H**
- Hardware, 7, 8
- HP diagnostics, 122
- HP SiteScope, 120
- HR/DSSS, 95

**I**
- I/O, 11, 16, 23, 27, 36, 57, 79
- I/O channel, 17
- I/O controller, 60
- I/O wait, 63
- Idle, 13, 62
- Information strategy, 140
- Integrative test script, 118
- Integrative testing, 113
- Interrupt, 13, 127
- Intrinsic, 23
- Investment, 77

**J**
- Job limit, 26, 41

**K**
- KPI, 38

**L**
- LAN, 86
- LoadRunner, 119

**M**
- Main Expense Driver (MED), 105
- Measurement, 35, 37, 49
- Medium access layer, 96

**N**
- Network performance, 83
- Network plan, 154

**O**
- OFDM, 95
- Operating system, 25, 37
- Optimisation, 67
- Order value, 157
- Overhead, 13, 62

**P**
- Packet switching, 90
- Paging area, 60
- Paging I/O, 64
- Paging rate, 16, 18, 27
- Performance indicator, 123
- Performance monitor, 42
- Physical Layer (PHY), 93
- PMS, x, 149
- Priority, 15, 16, 26, 71
- Process, 101
- Process overview, 126
- Processor usage, 42
- Production schedule, 37, 77
- Profiler, 123
- Program architecture, 27
- Program segment, 17
- Programming language, 27
- Project management, 149

**Q**
- Queue, 13, 41

**R**
- RAID, 21
- Reorganisation, 69
Response time, 10, 41
RFP, 13
RTS/CTS, 97

S
Spool parameter, 26, 37
SQL Server 2008, 122
Static data, 37
Subroutine, 21, 27
Swapping, 11, 58
System buffer, 18, 37
System disk, 69
System maintenance, 134
System table, 25, 59, 70
System tuning, 67

T
Target process model, 107
Task, 14
Test script, 49, 110, 114

Throughput, 11, 27, 41
Throughput rate, 11
Transaction rate, 11, 41
TuneUp, 128
Tuning, 139

U
UNIX, 125
Upgrade, 27

V
Virtual memory, 37, 70

W
Wait, 13
WAN, 47
WLAN, 87