**Chapter 1**

# Introduction

*gridGrid* *computing* is usually defined as sharing computational and data storage resources across organizational boundaries. In recent years, the development of virtualization technologies has enhanced the availability of services that are provided by grid computing. It has additionally enabled an evolution of the so-called *cloud computing*, which also utilizes a virtual environment on powerful computing infrastructures Based on the development of technologies and the philosophy of providing them to end users, this thesis focuses on the multidisciplinary that is research related to grid computing, as well as cloud computing. It discusses its utilization in biomedical research and its application in relation to the processing of medical information.

The term "medical information" is too broad and further work in this thesis focuses on the following selected areas: (1) the exchange and processing of medical images, (2) the analysis of human voice and (3) the modeling and simulation of human physiology. The author’s work was published in a series of peer-reviewed papers of international journals and peer-reviewed conference proceedings [1, 2, 3, 4, 5, 6, 7], which are included in this work as appendices. The author’s work and contribution was also presented in international conferences and published in the respective proceedings and transactions [8, 9, 10, 11]. The work was also popularized in local and regional conferences and their respective proceedings [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]. The author contributed to the utility model, which was registered by the Czech Industrial Property Office [24].

## 1.1 Thesis Goal

The hypothesis of this thesis is that the technologies that relate to grid computing and cloud computing may improve the processing of medical information in order to perform demanding tasks that are almost impossible or require onerous effort to achieve, using classical local or institutional resources. The particular goals of this thesis were:

• To study the latest achievements in the field of exchanging medical images and possible improvements using the grid computing and cloud computing technology.

• To identify use cases in other fields of biomedicine, which are suitable to utilizing the power of grid computing and cloud computing infrastructure.

• To develop and test the prototype application that utilizes grid or cloud technologies.

This thesis tries to discuss the hypothesis in different areas of biomedical research and its application. It tries to find answers to the following additional questions:

• *Is it beneficial to utilize grid computing and cloud computing technology for the processing of medical information and how do we do this?*  When work on this thesis begun, grid computing was believed to be an answer to scalability issues, e.g., for exchanging large amounts of data or carrying out demanding long-term computation.

• *What are the limitations of processing medical information in grid or cloud?*

• *How can the grid computing and cloud computing influence the direction of biomedical research?*  There was an idea that grid computing technology inspires the current architecture of distributed systems, e.g., exchanging medical images (explained in section 3.1) and influences the direction of information systems in hospitals.

Answers to these questions, which are based on the following chapters, are summarized in section 5.1.

## 1.2 Thesis Contribution

The author claims that the following contribution was made to state-of-the-art biomedical informatics and computational biology:

• Proposal of a grid infrastructure and pilot implementation of a grid-based system that exchanges medical images that are integrated with an existing distributed systems. The results were published in [1] and popularized in [12, 13, 14]. The author of this thesis customized the existing project, Globus MEDICUS, and deployed it in servers that are networked via the academic network, CESNET, and integrated with the existing regional PACS system. Other co-authors coordinated the work with operators of the regional PACS system and selected hospitals.

• Pilot implementation of a more generic infrastructure as a service for the community within the biomedical research [3, 19]. The author of this thesis proposed the idea of consolidating and sharing physical resources in order to provide a virtual environment for the specific needs of particular use cases. The pilot infrastructure was tested on examples of selected research projects.

• Proposal of a software architecture and implementation of a web-based service for the real-time remote analysis of a human voice. The results were published in [2] and popularized in [16, 20]. The author of this thesis designed and customized the existing network protocol in order to transfer voice signal losslessly. The author also deployed the application on a remote virtual server. Other co-authors implemented the algorithms and application in order to analyze voice signal.

• Improved methodology for the modeling of complex physiological systems [5, 6, 11, 10]. The author of this thesis contributed to the idea of building complex mathematical models from the basic components and keeping them in an understandable and maintainable form. Additionally, the author advised and implemented several basic blocks and models of a pulsatile cardiovascular system in Modelica language. The other co-authors implemented the library in order to model physiology, using an integrative approach. They also implemented the complex models, which integrated different domains together.

• Design and implementation of a system to estimate the parameters of complex mathematical models in order to validate or calibrate models of the human physiology were published in [4]. The gradual development of related technologies were published and popularized in [8, 9, 18, 22]. The author of this thesis designed architecture for a distributed parameter estimation algorithm, integrated models and implemented a pilot deployment, which utilized a scientific cloud computing infrastructure. Other co-authors implemented complex models of the human physiology in Modelica language and tested several algorithms for parameter estimation.

• Improved mathematical model of oxygen, carbon dioxide and hydrogen ion binding to hemoglobin [7]. The author of this thesis implemented this model in Modelica and identified its parameters. Other co-authors analyzed and proposed a new mathematical model, based on the basic physical and chemical laws and in relation to previously published studies.

• Simulation of complex models of the human physiology as part of a virtual simulator on portable and mobile devices, utilizing cloud computing [9, 21]. The author of this thesis contributed to the idea of a hybrid architecture of web simulators. This utilizes the infrastructure for parameter estimation in order to simulate complex models remotely and process/visualize the results locally. Other co-authors implemented complex models of the human physiology and implemented simulation scenarios for educational purposes.

• Virtual patient simulator prototype was registered as a utility model by the Industrial Property Office in the Czech Republic [24]. The author designed and developed a specific module to control the multiple instances of a virtual simulator within a virtual classroom via a web server application. Other co-authors designed and implemented models of the human physiology and clinically relevant educational scenarios. They also implemented a 3D visualization of selected scenarios using game engine Unity 3D[[1]](#footnote-2).

## 1.3 Thesis Structure

This thesis is interdisciplinary and therefore, the following chapters cover the topics from a technical and computer science point of view. They also touch on some topics that are related to medical science. Chapter 2 provides an overview of the state-of-the-art theory of computation, parallel computation and distributed computing, especially grid computing and cloud-computing.

Introduction to selected areas of biomedical research domains and related particular methods are introduced in chapter 3.1 for sharing medical images, in chapter 3.2 for voice science and chapter 3.3 for computational physiology.

Chapter 4 summarizes the general results that were obtained by the research methods in specific areas of biomedical research and applications. Chapter 5 discusses the achievements and answers of the hypothesis, as well as the questions that were stated at the beginning of the work. It also recommends further areas for research.

The appendices contain selected papers [1, 2, 3, 4, 5, 6, 7] that are most relevant to the topic of this thesis. These were published in international peer-reviewed journals or in peer-reviewed conference proceedings:

**Appendix**6 is the paper [1] *Processing of Medical Images in Virtual Distributed Environment,* published by ACM as part of the proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship.

**Appendix** is the paper [2] *Remote Analysis of Human Voice – Lossless Sound Recording Redirection* published in Analysis of Biomedical Signals and Images. Proceedings of 20th International EURASIP Conference (BIOSIGNAL).

**Appendix** is the paper [3] *Infrastructure for Data Storage and Computation in Biomedical Research,* published by Euromise s.r.o. in the European Journal of Biomedical Informatics.

**Appendix** is the paper [4] *Parameter Estimation of Complex Mathematical Models of Human Physiology Using Remote Simulation Distributed in Scientific Cloud,* published in the IEEE Xplore Digital Library as part of the proceedings of the 2014 IEEE-EMBS International Conference on Biomedical and Health Informatics.

**Appendix** is the paper [5] *Modeling of Short-term Mechanism of Arterial Pressure Control in the Cardiovascular System: Object-oriented and Acausal Approach,* published by ELSEVIER in Computers in Biology and Medicine 2014, **IF(2013): 1.475**.

**Appendix** is the paper [6] *Simple Models of the Cardiovascular System for Educational and Research Purposes,* published in Mefanet Journal 2014.

**Appendix** is the paper [7] *Adair-based Hemoglobin Equilibrium with Oxygen, Carbon Dioxide and Hydrogen Ion Activity,* published in Scandinavian Journal of Clinical and Laboratory Investigation 2014, **IF(2013): 2.009**.

**Chapter** **2**

# State-of-the-art

The processing of medical information deals with methods that connect different scientific domains, computer science, biomedical engineering and medicine, together with a common goal.

From a computer science (informatics) point of view, it is assumed that the processing of medical information is, in general, a computational problem, which is understood as a task that can be solved by a computer.

As some computationally hard problems are discussed later in this thesis, the next sections briefly introduce the theoretical and practical aspects, as well as the consequences, of the theory of computation, parallelism and distributed computing. Section 2.1 introduces some of the important problems, classed from the view of the computational complexity theory.

Parallel computation is introduced in some conditions where speedup is needed for computing concurrently. This theory is briefly covered in section 2.2.

The distribution of a parallel task via a computer network to other computers, servers and clusters is covered in section 2.3, with a focus on grid computing and cloud computing.

## 2.1 Computational Complexity

An algorithm is a set of operations that is used to accomplish tasks and solve problems. There are several ways of expressing algorithms, e.g., in text, in programming language, pseudo-code or flowcharts. Later in this thesis, kopenograms will be used as a graphical language for structured algorithms in order to supplement the Unified Modeling Language (UML) diagrams, as proposed by Kofranek et al. [25][[2]](#footnote-3).

The computational complexity theory classifies problems into several classes, according to the time or space needed by the algorithm in order to solve the problem. The time complexity of an algorithm is usually denoted by a big  notation and size of input problem  meaning that a time complexity denoted by  is not growing faster than the function . Formally , if, and only if, there is a constant  and positive integer  that for each : .

 denotes algorithms that take constant time, regardless of the input size. denotes linear time algorithms. For example, Figure 2.1 shows the sequential search algorithm in a pseudo-code and kopenogram, which 2.1needs to compare each record with a given key. This is used to find some items in an unsorted list or array. For example, if a single comparison takes  seconds and the list has  records, then the algorithm will take, at worst,  steps and the time complexity will be . [[3]](#footnote-4)

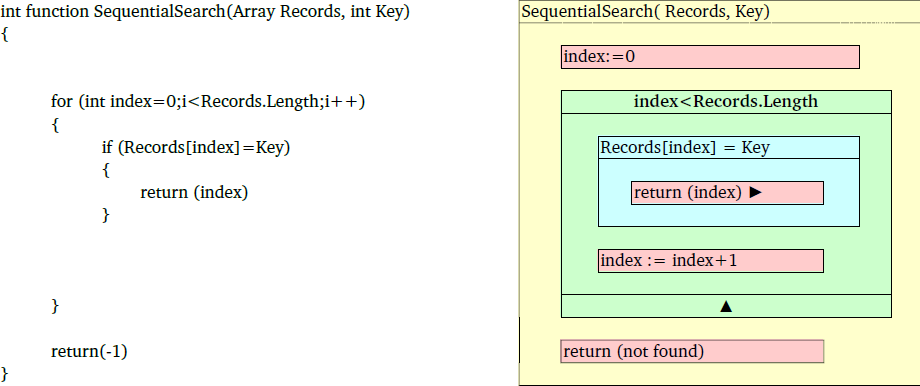


Figure 2.1: Pseudo-code (left) and kopenogram (right) of a sequential search algorithm with complexity . The red blocks are commands, i.e., setting the index or returning results. The green block represents the loop with entry condition. The index is incremented and in programming languages this can be achieved by, e.g., *for* cycle statement. The blue block is the condition (*if* statement). When this is fulfilled, then the inner blocks are executed and, in our case, the found index is returned. If no record is found, the loop will end with the last index within *Records* and *not found* sign is returned. 

Polynomial time algorithms are defined as the ones that take time complexity can be bounded by  for some constant . The class of problems, solvable by polynomial algorithm, is denoted as *class P* and is recognized as tractable - first noted by Cobham and Cook [28, 29]. Other algorithms, whose time cannot be bound by any polynomial functions, are called exponential. These are recognized as intractable, as published by Garey and Cook [30, 29]. For a relatively small input data, the exact solution can be found using the known exponential algorithm with a current computation power. However, for bigger input data, the time needed to solve the problem is far a beyond reasonable amount, as seen from Table 2.1.

A class of problem was identified where it is not known whether a polynomial time algorithm exists. However, if such a solution exists, it can be verified in polynomial time – class *NP-complete*. If some polynomial algorithms are found, then a derived algorithm for the future solving of some NP-complete problems will solve other problems, as well as in polynomial time. This was denoted by Cook and Karp [31, 32]. To solve such problems, the current best-known class of algorithm is based on a brute-force search of all of the possible values. It is an open question as to whether a better one exists.

A brute-force search is a general solving technique that generates all of the possible candidates of solution and checks if the problem satisfies the problem statement. All of the algorithms for solving a brute-force search suffer with exponential time complexity[[4]](#footnote-5).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [height=43pt,width=125pt]timecomplexityfunctioninput size | 10 | 20 | 50 | 100 |
|  | s | s | s | s |
|  | s | s | s | s |
|  | m  s | m  s | h m s | days |
|  | s | m  s | years | years |
|  | s | days | years | years |

Table 2.1: The computation time of algorithms with different time complexity functions, where one step of algorithm takes 1 millisecond. Examples of algorithm with polynomial time complexity  are compared to algorithms with exponential time complexity . It is important to note that, for the problems with an input size of 50 and greater, the exponential algorithm runs far beyond the reasonable time, compared to the age of the universe, which is currently estimated to be  years [35].35

If we presume that the technological update and computation speed will increase, the effect of the technological speedup is visible in Table 2.2. The effect on the polynomial algorithm is multiplicative. However, for the exponential algorithm, the technological speedup will only slightly increase the size of the computable problem. This is the reason why the problems with only exponential algorithm are denoted as intractable.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Present computer | 10 times faster | 100 times faster | 1000 times faster |
| 2\* | 3600000 | 36000000 | 360000000 | 3600000000 |
|  |  |  |  |  |
| 2\* | 1897 | 6000 | 18973 | 60000 |
|  |  |  |  |  |
| 2\* | 20 | 32 | 51 | 81 |
|  |  |  |  |  |
| 2\* | 21 | 25 | 28 | 31 |
|  |  |  |  |  |
| 2\* | 13 | 15 | 17 | 20 |
|  |  |  |  |  |

Table 2.2: Effect of computation speedup. The first value is the input size of data that is computable in one hour and the second value is the speedup that is achieved, compared to the value in first column.

NP-complete problems, which are currently solved by exponential algorithms (a better algorithm has not been found), are covered in the published works of M. R. Garey and D. S. Johnson [30]. The whole complexity theory is also covered in published works, e.g., Ch. Papadimitriou [36] or M. Sipser [37].

The technological speedup will mainly impact the class of problems, which are solvable by polynomial algorithm. Other non-exact methods are used to find at least some solution for the problems that are only solvable by exponential algorithms. Examples of these are:

•The *heuristic method* tries to eliminate the number of steps of computation by some implicit or explicit knowledge of the specific problem itself, e.g., eliminating solution classes that seem to not go to optimal solution. With a combination of a brute-search, the heuristic method reduces the size of all of the possible solution candidates to check.

• The *randomization method* uses a non-deterministic methods in some level of computation. For example, the Monte-Carlo method is used to compute problems using pseudo-random generated values and, after several iterations, statistical methods are used to compute the expected value and standard deviation.

• *Restriction on input data* – this is another form of using the explicit knowledge of the problem instance and it may reduce all of the possible values to be checked.

• *Approximation algorithm* – this not only finds some good solutions but also, it quantifies how far from the optimal solution the result is, with some degree of probability.

## 2.2 Parallelization

If a sequence of instructions can be divided into parts, which can be computed independently in parallel by multiple processors, then it is possible to achieve some computation speedup using current computational technology. A speedup of a computation on P processors can be defined as:

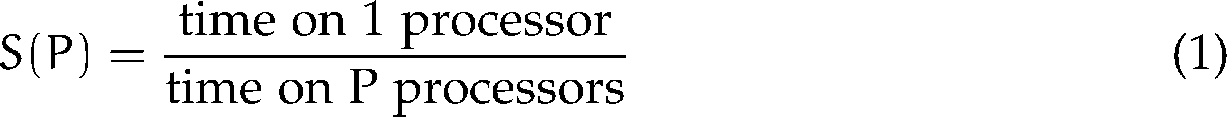
 (2.1)

Figure 2.2 shows serial and parallel execution computations of the same algorithm.

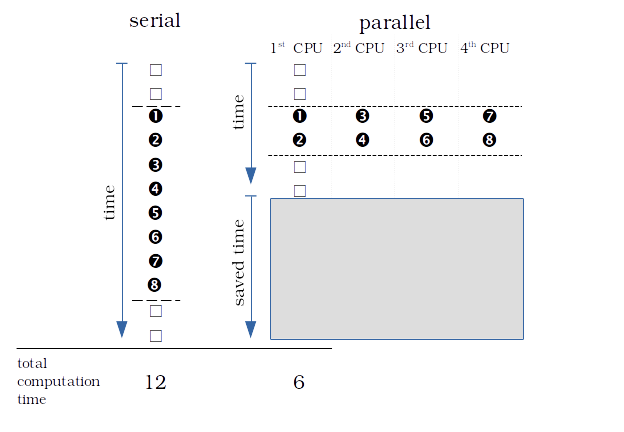
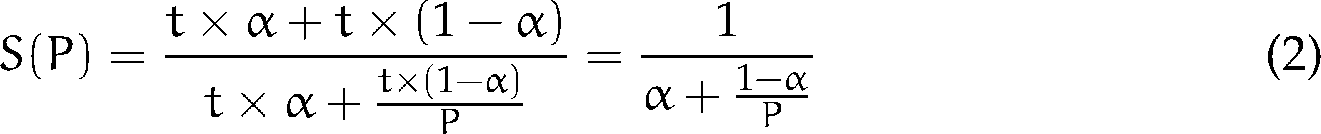
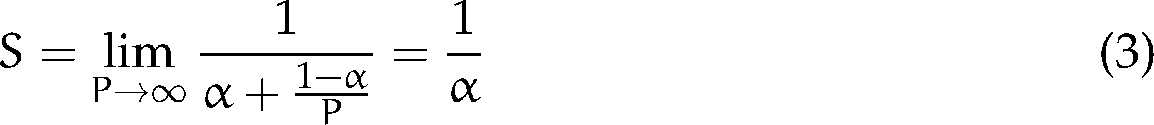


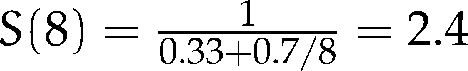
Figure 2.2: Comparison of serial and parallel executions of instructions. The instructions with numbers can be executed in parallel. In this case, the serial computation takes 12 cycles, the parallel computation on four CPUs takes six cycles and the speedup is two. If we have eight CPUs, then the computation will be finished in five cycles and the speedup will be  times . .

Assume  as a fraction of the computation in one processor, which cannot be parallelized,  is a fraction of the computation in one processor, which can be parallelized by  processors, and  is the time needed to compute the process on one processor. Assume that the overhead of parallelization is small and can be disregarded. Then, the speedup can be computed as:

 (2.2)

On an unlimited number of processors, a theoretical upper bound of speedup can be formulated, which depends on  only, denoted as Amdahl’s law [38]:

 (2.3)

For example, when a 33% of a computation cannot be parallelized (), then the speedup on eight processors can be theoretically  and theoretical speedup on an unlimited number of processors can be . See more at Figure 2.3.

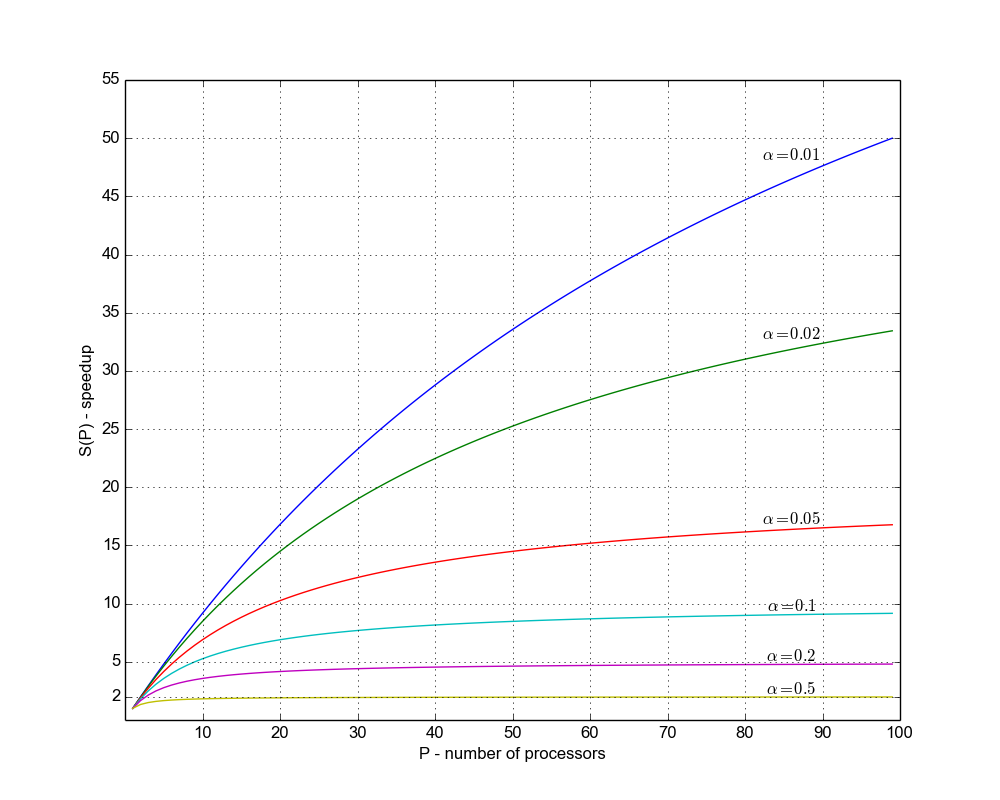
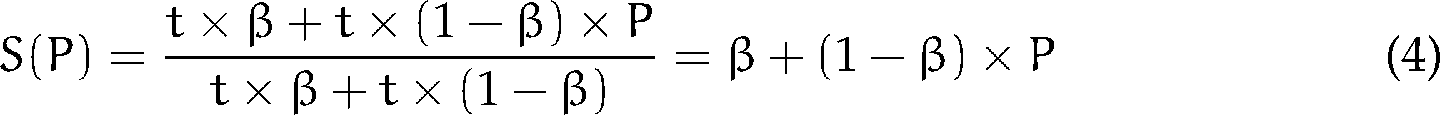


Figure 2.3: Speedup gained on 1 to 100 processors per Amdahl’s law for different  values.

However,  can sometimes be hard to estimate. Additionally, the computing of a fixed size problem on a high number of processors can misrepresent the speedup expectation. Therefore, Gustafson reformulated the law and described another approach to measure the fraction of the computation, which cannot be parallelized from computing on  processors. This approach also estimates the speedup of how long such a computation will take on a single processor. In this case, assume that the overhead of the parallelization is small and can be disregarded. The  is the "scaled fraction" of the computation on  processors, which cannot be parallelized [39]:

 (2.4)

This law presumes that fraction  will not change on a different number of processors, as seen in Figure 2.4.

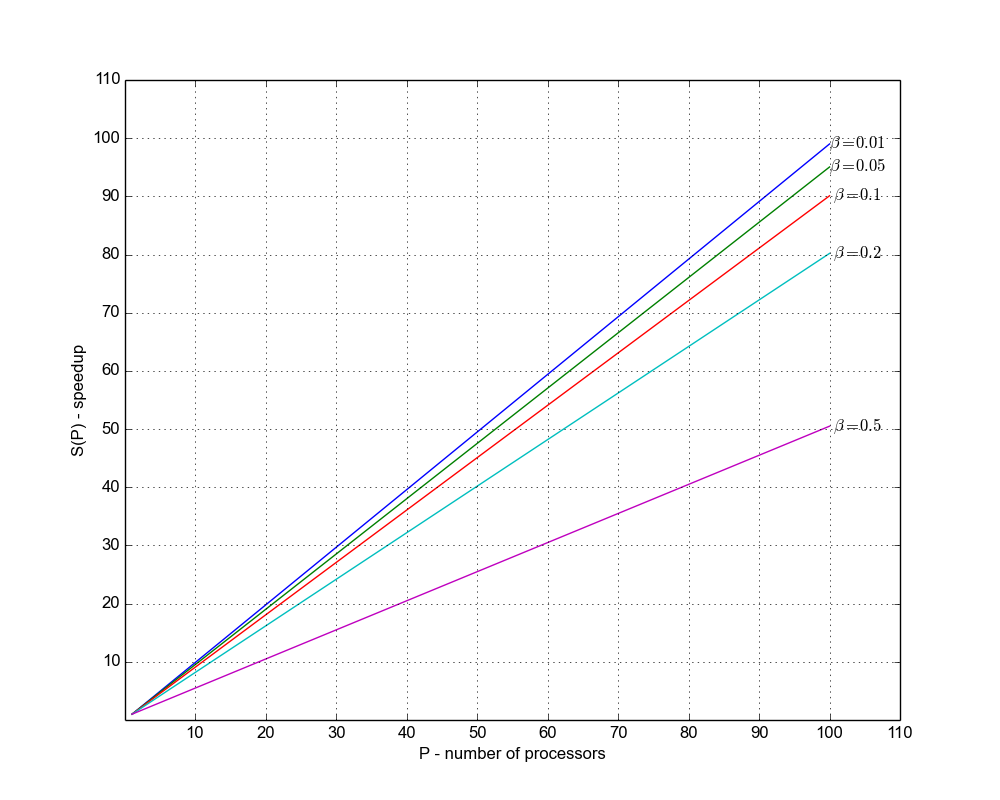


Figure 2.4: Speedup gained on 1 to 100 processors per Gustafson’s law for different  values.

Both laws disregard parallelization overhead, however, if there is a significant one, the speedup of parallelization will be degraded by the overhead. Amdahl’s law (2.3) is argument that maintains that the speedup is limited for current sized problems. However, bigger problems can be addressed with a higher number of processors and this should be considered, rather than Gustafson’s law (2.4).

### 2.2.1 Programming Model

There are several levels of how parallelism is realized:

• *Instruction level parallelism* - if the instructions are independent, they can be executed at the same time by a multiple central processing unit (CPU), e.g., by several cores of the multi-core processor. Programs are usually written as a sequence of instructions. Instruction parallelism depends mainly on the compiler’s capabilities to recognize or reorder the instruction to execute the instruction in parallel. In the multi-core processor era, instruction parallelism started to be systematically utilized.

• *Data parallelism* - the same operation is performed on multiple data, usually arrays. The instruction is distributed into multiple processors or processor cores and is executed on elements of data structure in parallel. This is currently the characteristic feature of a *General Purpose Graphical Processing Unit* (GPGPU), computing and programming API such as CUDA[[5]](#footnote-6) and OpenCL[[6]](#footnote-7)

•*Loop parallelism* - the computation may contain iteration on some large data structures. Such iterative processing is usually programmed as a loop and if  iteration is independent on , then the iteration can be executed in parallel by different processors.

• *Task parallelism* - the computation contains parts that are independent of each other. The computation of such parts can be scheduled and distributed into multiple processors and can be computed concurrently. For example, a master/worker pattern is realized (master sets up a pool of worker processed and a set of tasks that are distributed to them). Fork/Join pattern – the main process forks into several threads that execute concurrently and wait for the results to join back into a single process, which may, after some computation, fork again.

Looking at the way the processes interact, these are the most common forms:

• The *threads* are several concurrent execution paths that are independent but, in general, share the same memory. They are standardized, e.g., as POSIX threads (Pthreads) and implemented in many platforms. Further reading about Pthreads can be found in the published works of Butenhof [40]. There are other implementations that go beyond the POSIX standard, which have been introduced in other languages and programming environments.

• *Shared memory*. The *OpenMP[[7]](#footnote-8)* is a shared memory application interface, which is standardized and implemented by several compilers for C, C++ and Fortran. It also uses a multithreaded model, however, programming is task-oriented and more abstract than using threads, as described by Chapman et al. [41].

•*Message passing*. The *Message Passing Interface* (MPI) is a specification for performing task communication by passing messages between tasks. Further reading about MPI can be found in the published works of Pacheco [42].

More information about parallel programming models can be found in a survey by Diaz et al. [43].

Some algorithms can be easily divided into independent tasks, which can be computed in parallel. If there is no need to communicate among the parallel tasks, such algorithms are called embarrassingly parallel. For example,

• Operation on matrices [44] are currently used to render 2D and 3D graphics.

• Parameter studies, where the same computation is performed using different sets of input parameters [45].

• Brute-force search algorithm, where a subset of possible candidates for solution are generated and checked in parallel.

• Genetic algorithm and other evolutionary algorithms [46].

In contrast to embarrassingly parallel problems, there are inherent sequential algorithms that cannot be significantly speeded up by parallel computing. Both aspects of scalability (speed up gained by parallel computing) and effectivity (time demand based on the size of input data = time complexity) should be considered as highly scalable algorithms. These can be outperformed by sequential algorithms, which solve the same problem with a better time complexity, as noted by Madden [47].

Other of parallel computing and the design and build of parallel programs were published in the earlier works of I. Foster [45], D. Culler et al. [48] or T. Rauber and G. Räuder [49].

### 2.2.2 Summary

To summarize this section: parallel computing can introduce speedup on current computational technology and some computation problems may become feasible. Additionally, overhead, which is caused by parallelization and fraction of non-parallelizable parts, should be considered as it may degrade the expected speedup. In the case of exponential algorithm (e.g., for NP-complete problems), the speedup will only increase the size of the solvable problem only slightly (see Table 2.2) and some problems cannot be (or it is believed that they cannot) significantly speedup by parallel computing. In later text of this thesis, we mainly focus on tasking parallelism and distributed computing.

## 2.3 Distributed Computing Technologies

Distributed computing is based on the idea of spreading a computation task into a set of computers, which are connected via a computer network.

The main motivations of using distributed computing technologies are to: (1) share, store and exchange resources, (2) provide and consume computational services, (3) access a much higher capacity of storage and computation than is available locally and (4) connect users, developers and people.

To manage distributed computing, several challenges are maintained such as synchronization (the exchange of messages in a computation workflow) to achieve, e.g., mutual exclusion (when a task needs exclusive access to some resource), prevent deadlock (no progress is possible) or resource starvation (when resources – such as processor time - are not scheduled for a particular task for some reason and the task cannot finish computation). Distributed systems offer some sort of fault tolerance (managing fault of a node during computation) or security (encryption of communicating channels and stored data, authentication and authorization to access some resources or data), etc. The topic of distributed computing is covered in the published works of Tannenbaum [50].

An extreme example of distributed computing is the Internet. Here, computers are interconnected via TCP/IP protocols and servers (e.g., web servers) provide services, with some degree of security and fault tolerance. For example, the World Wide Web is based on HTTP protocol and HTML language and related technologies. Peer-to-peer services are based on TCP/IP or UDP, as well as the streaming of data.

For scientific purposes, distributed computing infrastructures evolved into sets of clusters, computing centers or individual computing resources that are owned by different subjects. A continuous effort is being made to join such resources into a federation of computational capacity via high-speed networks. This will allow a better virtual capacity to be obtained. Some minimal requirements were formulated, as well as defined standards for network protocols and services, which a distributed infrastructure should fulfill and provide. Such infrastructures are currently distinguished as grid computing or cloud computing and their users can gain access to a much higher virtual capacity than accessible locally. Users can also access remotely specialized devices, which are not available within their institution.

### 2.3.1 Programming Model

A parallel programming model (section 2.2.1) is used to realize distributed computing in a local computer or server. An additionally higher level of task interaction is realized via a shared distributed file system or by messages that are passed over a computer network. Looking at software layers, distributed computing usually incorporates one or several new layers. Some of these are called middleware delivering services or APIs for application and hide specific implementation across heterogeneous computing platforms (operating system and hardware).

#### System Architecture

As algorithms and programs are needed to solve an increasing number of problems and changing requirements, a view on software architecture is needed in order to construct and order several programs and algorithms into a more robust system, aiming to solve a broader set of problems.

Within distributed computing, the major software architecture is based on client-server architecture (e.g., Figure 2.5), peer-to-peer architecture or more layered architecture patterns.

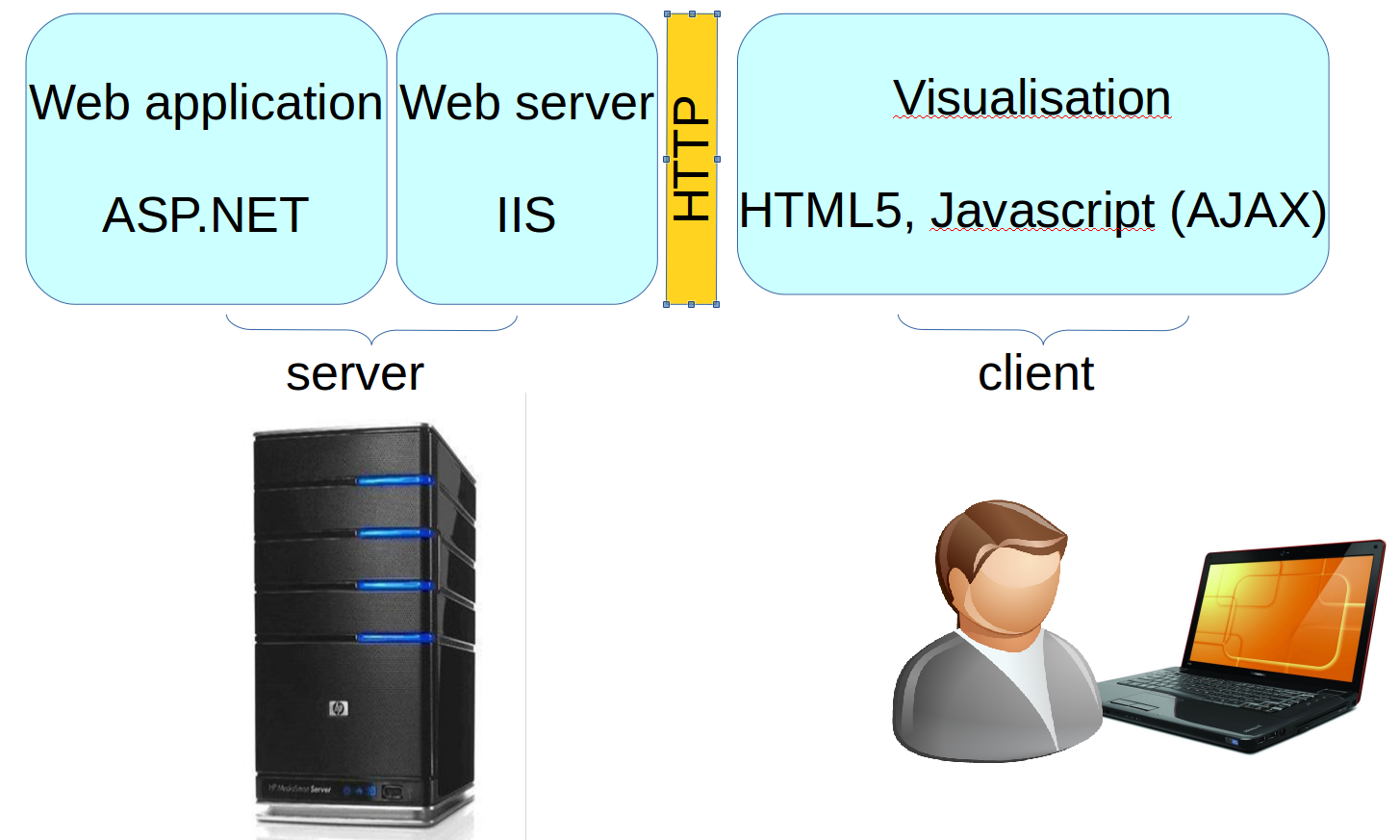


Figure 2.5: Example of client-server architecture involving a web server, which is middleware between web application and server platform. Client visualization in HTML5 web page communicates with the server via a HTTP protocol.

Service-oriented Architecture (SOA) is a high-level programming model, which is based on self-contained units of functionality – service – and covered with documented interfaces. SOA introduces a new service layer in client-server architecture, which separates service interface from its implementation. T. Erl. described SOA principles and paradigms [51].

Another approach represents the objects and data of a system as resources with a standard set of operations - create, read, update, delete (CRUD). Representation State Transfer (REST) specifies several architectural constraints that help scalability and performance. It presents functionality via a fixed number of operations and uniform resource locations, as proposed by Fielding [52]. The constraints of REST are that its application is statelessness, resource-orientated with uniform interface (CRUD) and hypermedia-driven, which should facilitate and optimize the processing of resources via current web based technologies, mainly HTTP protocol.

While SOA focuses on application design and easily turning application objects into distributed services, REST is rather a set of constraints on the architecture, which are used to handle the issues of distribution within the web, as noted by Vinoski [53].

The software architecture of enterprise applications, distributed systems and some repeating patterns are cataloged in published works, e.g., by Fowler et. al[54] or Nilsson [55]. Furthermore, Hohpe et al. discusses integration patterns, with a focus on the ways of connecting heterogeneous parts of the system [56].

#### Types of Computing Infrastructure

When we focus on the architecture of middleware and the philosophy of building a computing infrastructure, these main types of distributed infrastructures are distinguished for scientific computing and are relevant to the rest of this work:

1. *Service grid computing* is based on the idea that computing resources (servers, clusters and special hardware) are owned by some organizations but may be maintained by some collective organizations, with an effort to provide a collection of services in a best effort approach.

2. *Desktop grid computing* is based on the idea of connecting generic desktop computers and providing the idle computation time, e.g., as a screen saver or background process to the projects.

3. *Cloud computing* provides architecture to services, a platform or a whole infrastructure in a way that provides access as a service, with the impression of sole use of it by the user or process.

### 2.3.2 Research and Education Network

The fundamental part of any grid is the computer network, which connects resources that are distributed in different geographical locations, generally on the Internet.

The national grid initiative in the Czech Republic, METACENTRUM[[8]](#footnote-9), is interconnected via the high-speed network, CESNET 2, which utilizes the technology of transferring data over optical cables using Dense Wavelength Division Multiplexing (DWDM) [57], as seen in Figure 2.6.

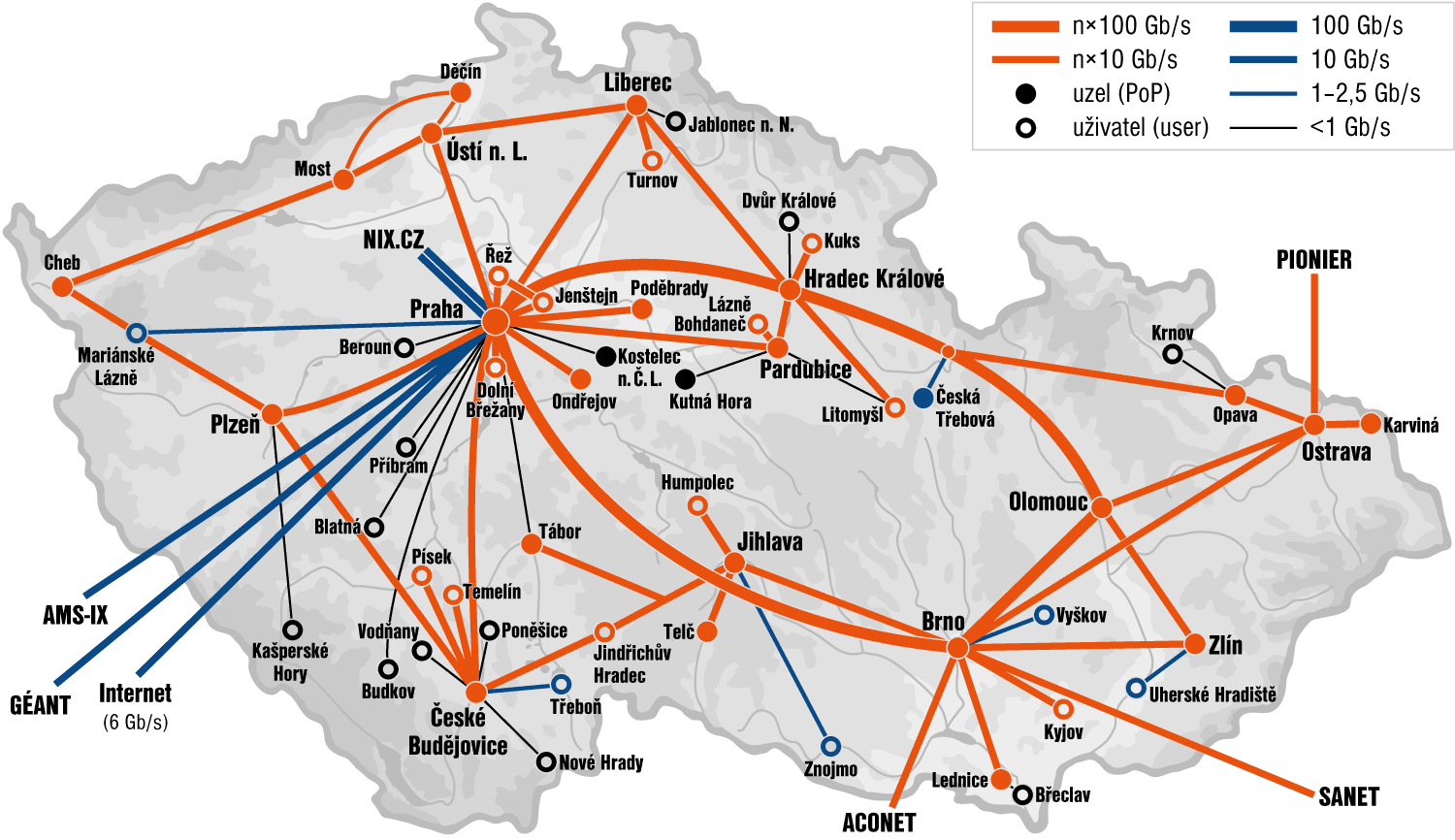


Figure 2.6: From December 2014, the CESNET2 network topology has been maintained by an association of universities and academy of science, CESNET. It mainly interconnects mainly departments of universities and academy of sciences via a rented physical network. It provides connection to the general Internet via the Czech NIX.cz (Neutral Internet eXchange), AMS-IX (Amsterdam Internet eXchange) and other channels to European research and education network GÉANT. Sources: [http://www.cesnet.czhttp://www.cesnet.cz](http://www.cesnet.cz/)

### 2.3.3 Service Grid Computing

Service grid computing is based on a basic set of services, which are implemented by a middleware. They provide uniform interface for job scheduling and execution within the computing infrastructure. The term *grid* is used to emphasize the analogy with an electric power grid, providing access to electricity [58]. Foster et al. [59, 58] and Chervenak et al. [60] describe "data" and "computational" grids as shared hardware and software resources, which provide reliable, consistent, pervasive and cheap access to high performance computational capacities. They also provide the effective and reliable execution of requests over data, which needs sensitive controlling of terabyte storage, data transfers to gigabits per second over global computer networks and the scheduling of such data transfer, with respect to computational needs. The services provided by grid are either tools or web services, following *Service-oriented Architecture* (SOA) for grid computing – Open Grid Service Architecture (OGSA) [61]. The security model and access to grid infrastructures are mainly proposed and implemented by a mutual authentication between users and resources via a public key infrastructure, using the X.509 certificate [62]. The administration and maintenance of such networked infrastructures are not trivial tasks and are performed by experts of institutional computing centers or national laboratories. Furthermore, interconnected sites are managed and coordinated at a national level or an international level. Such national organizations cooperate with similar national grid infrastructures in other countries. An umbrella organization in Europe – the European Grid Infrastructure (EGI)[[9]](#footnote-10) – was established during 2010. This supports the integration and coordination of activities of National Grid Initiatives (NGI) across national boundaries, with respect to the scientific collaboration across national boundaries.

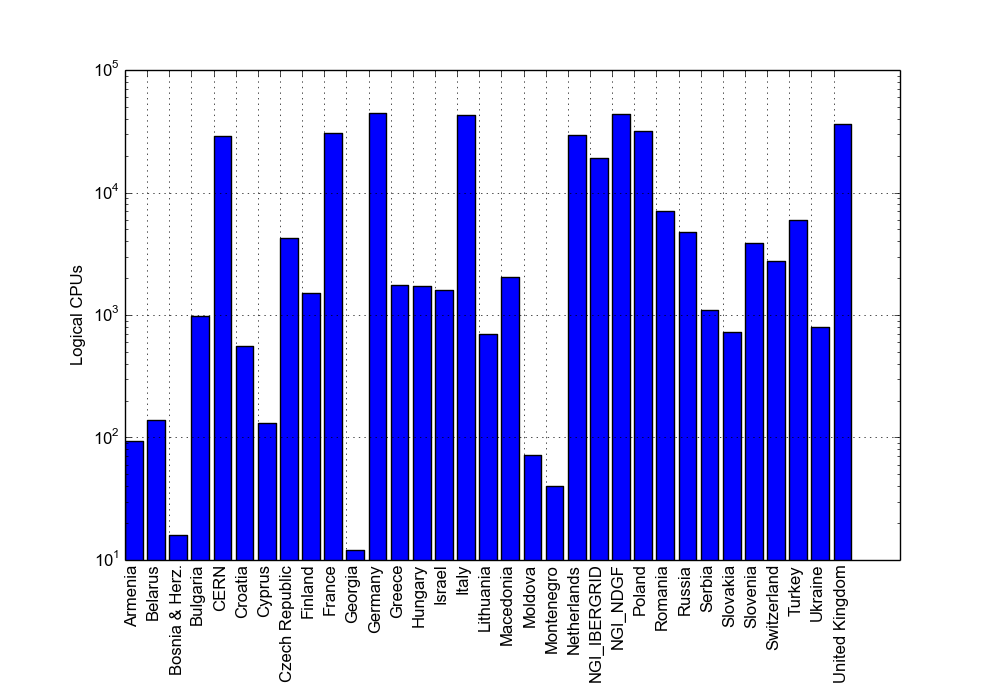
 

Figure 2.7: The countries that are involved in EGI and the number of CPUs within the EGI infrastructure in 2012. At the end of 2012, the total logical CPU capacity was 349,720 cores (in the graph per country). In March 2014, the CPU capacity was 433,957 cores. Sources: EGI Compendium 2013, EGI statistics at [http://www.egi.eu](http://www.egi.eu/). [http://www.egi.eu](http://www.egi.eu/)

In the last decade, there has been an acceleration in the growth of several production grid infrastructures for science. This is mainly due to the need for experiments of high-energy physics in order to process a large number of observed data in a reasonable time [63]. The Worldwide Large Hadron Collider Computing Grid (WLCG) was designed to store and process almost 30 PetaBytes of data per year in the period of 2009-2013 [64]. It is one of the largest grids to be deployed in a grid infrastructure. As the hardware infrastructure is built with a philosophy of federated access to resources, which are owned by research institutions, universities, etc., other scientists from different scientific disciplines can also become users of this powerful infrastructure. Due to the development of virtualization, the infrastructure may employ a larger set of applications and services and may be attractive for smaller scientific collaboration.

Several grid infrastructures were established, based on different grid middlewares. Condor is one of the earliest efforts to provide access to underutilized computers, while preserving the rights of the owners [65]. Major grid middlewares that are operational in EGI are glite [66][[10]](#footnote-11), ARC[[11]](#footnote-12) and Globus [67][[12]](#footnote-13).

Efforts are currently being made to maintain interoperability in order to connect different applications with different resources from different grid infrastructures. Riedel et al. reports on the efforts being made to involve technology providers, as well as deployment teams, in order to participate in open standards of security, data management, etc. [68].

### 2.3.4 Desktop Grid Computing

Joining desktop computers from an individual user to form a voluntary or desktop grid was popularized by a project that tries to identify uncommon signals from space to search for extraterrestrial intelligence (SETI@Home)[[13]](#footnote-14). It is based on the idea that a volunteer downloads a small client program, which executes in the background or instead of a screen saver. It downloads an amount of raw data from a server on the Internet in order to be analyzed. It then sends the result back to the server. In contrast to service grids, the authorization of users cannot be so strong for volunteer individuals and some other policies, e.g., redundancy, are implemented to eliminate bad or cheating results [69]. After the success of the SETI@Home, general-purpose frameworks were built in order to facilitate the development of projects that use a similar philosophy of computing on desktop computers, which are connected via the Internet such as BOINC [70], SZTAKI extension to BOINC [71, 72], XtremWeb [73] and others. Currently, there are a lot of similar projects that gain the same computer power as the SETI@Home project, e.g., the LHC@Home and its successor, LHC@Home 2. These projects were established and used to execute some selected tasks of the Large Hadron Collider (LHC) project on a desktop grid infrastructure [74, 75].

The average performance of BOINC projects is 8.073 PetaFLOPS, with 294,764 active volunteers computing on 502,238 computers (March 18, 2015). For example, SETI@Home’s 24 hour performance is 1.95 PetaFLOPS. Although desktop grids and service grids are two different approaches that are used to gather computing power from a large number of computing resources, efforts are being made to interoperate and share the capacity for such a project, e.g., the EdgE project, published by Kacsuk et al. [76] and Urbah et al. [77].

### 2.3.5 Virtualization

### Virtualization technology separates the physical hardware layer from the software environment, emulating a new virtual hardware layer. The hypervisor or virtual machine manager manages guest virtual machines and translates I/O operations between virtual device and physical devices. It also translates instructions from virtual CPUs to physical CPUs. This introduces some overhead and performance degradation of the virtual system, compared to the physical. However, recent virtualization technology has introduced several techniques that reduce overhead and eliminate specific hardware features and instructions, which are hard to virtualize, as reported by Barham et al. and Youseff et al. [78, 79]. Thanks to them, a virtual environment has been fine-tuned for an application that can be executed on almost any hardware and platform. Here, virtualization becomes a part of the solution to execute jobs of desktop grid or service grid computations on different physical platforms [80]. Currently, there are several commercial, free or even open-source virtualization implementations, which are provided by different vendors and hypervisors - VMWare[[14]](#footnote-15), XEN[[15]](#footnote-16), KVM[[16]](#footnote-17), VirtualBox[[17]](#footnote-18), etc. 2.3.6 Cloud Computing

In contrast to grid computing, where a user schedules jobs in order to access a shared environment and may be influenced by other users or by the environment, cloud computing provides access to a virtual software, platform or whole infrastructure. Consequently, the user or process is given the impression of sole use. Virtualization techniques have enabled the expansion of cloud computing, mainly on infrastructures that were built for another purpose. These can be rented when the primary infrastructure is not fully utilized [81].

Cloud computing can be characterized as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services). These can be rapidly provisioned and released with minimal management effort or service provider interaction [82].

Cloud computing makes computational power and storage as utilities or commodities that can be rented. With current commercial clouds, the commercial area has evolved in order to facilitate the scaling up that businesses needs with the computational demand, e.g., Amazon EC2[[18]](#footnote-19), Microsoft Windows Azure [[19]](#footnote-20), Google cloud[[20]](#footnote-21) and others.Cloud computing in research infrastructures was recently deployed, next to the already existing grid infrastructures, utilizing the same hardware resources. Some methods to integrate grid computing and cloud computing were proposed by Anjum et al. [83]. Access is provided to the same users of a grid infrastructure and currently, the most used platforms are Open­Nebula [84] and OpenStack [85].

Interoperability among cloud providers and a standardization of cloud computing, virtualization and related technologies are important as these would keep users from being locked into a specific cloud provider [86].

### 2.3.7 Application Model

Applications that are computed within a grid or cloud can be characterized by the quantity of tasks being performed, the size of the input data and the communication that needs to be carried out between concurrent tasks. Grid computing infrastructures are primarily utilized for computation, in which tasks take a long time. These are relatively loosely coupled and resources are used over a long period of time. Performance/capacity is usually mentioned in operations/CPUs per month or year and for such computations, the term High Throughput Computing (HTC) is used.

While HTC takes a long time, the High Performance Computing (HPC) is usually characterized by computing problems that have a small number of tasks. These are relatively tightly coupled and can be executed in a highly parallel environment. Performance is measured usually in operations per second [87, 88]. The grid infrastructure can involve HPC servers or clusters and thus, a job or task that requires such HPC hardware is scheduled and executed there.

Many Task Computing (MTC) aims to bridge HTC and HPC. While the computation usually takes a shorter amount of time, the data exchange is in MB rather than GB. Performance is measured in tasks per seconds rather than jobs per months or years and it involves computing much more heterogeneous problems, which are not "happily" parallel. However, middleware for HPC or HTC, which are present in grid computing infrastructures, may introduce some shortcomings. Therefore, Raicu et al. proposed and implemented a prototype task execution framework, which is suitable for MTC [89, 90, 91]. MTC seems suitable for performance on cloud computing technologies and has some benefits over classical grids. However, such clouds should be oriented for HPC systems and a generic public cloud may introduce a lower performance than expected [92].

### 2.3.8 Workflows and Gateways

A workflow is an abstract description of the process of computation and data manipulation, which is specified by an expert to express in order to express what should be done within a distributed system. It automates the process of computation by composing data manipulation steps and tasks, as well as solving failed events.

The workflow can be encoded in any programming or scripting language, however, some higher level languages have evolved. In the business domain, a Business Process Execution Language (BPEL) is an OASIS standard and has become one of the most used languages for describing the workflow of orchestration of web services and transaction steps [93]. In the scientific domain, different workflow systems are operational, including BPEL, with different capabilities. Yu, Zhao et al. published the taxonomies of some of the existing workflow systems [94, 95, 96]. Workflows in cloud computing are covered by web technologies programming languages (Javascript) [81].

The workflow system, which implements concrete workflow language, is usually tightly coupled with a specific grid computing or cloud computing infrastructure.

To connect the different grid infrastructures, a mutual workflow management system is used to integrate them, as proposed by Kacsuk et al. [97, 98], or an interoperability is solved. This is achieved by separating the abstract workflow representation and concrete implementation, which are shown on the selected existing workflow systems, as introduced by Planensteiner et al. [99].

Scientific gateways incorporate a higher level of services for specific scientific communities, e.g., as a web portal or desktop application in order to control the process of computation via a workflow [100]. Several frameworks were developed for building scientific gateways, e.g., Apache Airavata [101, 102] or WS-PGRADE/gUSE[103]. Furthermore, the concrete instances are available for a broader area of the scientific community.

**Chapter** **3**

# Methods

The general issue of utilizing grid or cloud computing infrastructures is the selection of the appropriate method that is used to integrate domain-specific computations into the grid or cloud infrastructure of a concrete provider.

Various tools are already available within current grid infrastructures, including open-source or licensed software for computation. The local scientific provider[[21]](#footnote-22) usually gives a list of the available applications. Alternatively, application databases are available in a broader environment, e.g., in the EGI.eu application database[[22]](#footnote-23). Additionally, the workflow systems and scientific gateways that are mentioned in section 2.3.8 try to hide the complexity of grid computing or cloud computing infrastructures and may also be used to integrate specific domains. In designing a new application, the programming model of parallel computing and/or distributed computing (in section 2.2.1 and 2.3.1) needs to be followed, utilizing the benefits of grid computing and/or cloud computing.The general approach to port applications to a grid infrastructure is to automatize what can be automatized, i.e., make scripts, configure system, prepare some user interface, integrate with existing applications, utilize protocol compatibility, etc. Efforts to obtain the first results are high, however, if the prepared template, script or application is reused for further computational requests, then the effort is much lower.

## 3.1 Sharing Medical Information

Use cases that relate to digital medical images involve image acquisition, preprocessing, storing and searching. Clinicians mainly use patient images for visualization and diagnostic purposes. Computer assisted methods facilitate the diagnostic process and involve image enhancement (to reduce the image noise and increase the contrast), image segmentation (to separate different types of structures from the background and from each other), quantification methods (to determine the structure shape, size and volume) and registration methods (to process and join multiple different images into one). Comprehensive concepts and digital techniques in medical imaging are presented in published works, edited by I. N. Bankman [104].

The acquisition of a medical image is covered with different modalities (different types of equipment and sensors) by radiologists or other specialists. The DICOM[[23]](#footnote-24) format and protocol has become an industrial standard in exchanging medical images electronically and in Picture Archiving Communication Systems (PACS). Holding the acquired DICOM images with the metadata and description, which are noted by experts, is currently part of the information systems in hospitals. Figure 3.1. shows the typical workflow of a medical image in a hospital3.1.

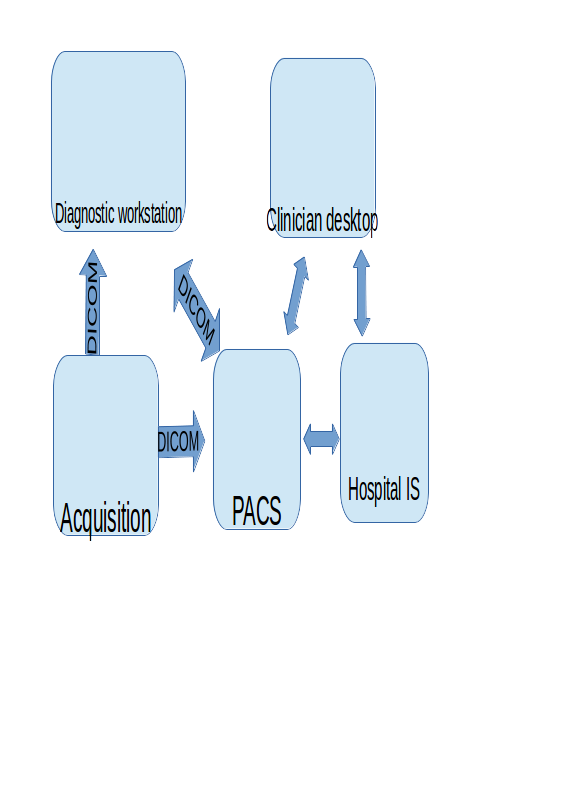


Figure 3.1: The typical workflow of a medical image in a hospital. Data acquisition is made by modalities (magnetic resonance, ultrasonography, X-ray radiography, etc.). By using the DICOM format and protocol, it can be directly transferred and visualized by a diagnostic workstation. With the metadata filled by an expert physician, the image is stored in PACS. Other desktops within the hospital can retrieve the image and review the report. The hospital information system may be involved in other workflows and communicate with other formats and standards (e.g., HL7).

As the data that are processed in hospital information systems contains sensitive information of real patients, they are protected. The national or international laws or agreements regulate the processing and storing of such data. The development of telecommunication and network technologies has enabled telemedicine – providing healthcare over a remote distance. This requires the sharing and exchanging of sensitive data of real patients among different healthcare providers and such data may be very valuable for further research. Security and encryption should be addressed and the DICOM standard itself does not appropriately solve security issues. Thus, encryption during the transferring of data over to a computer network must be ensured by other techniques.

In the Czech Republic, there are several projects in production that interconnect different hospitals, clinics and other healthcare organizations in order to exchange medical images. Project ePACS allows the interconnection of each participant’s PACS system via a dedicated Virtual Private Network (VPN) channel to the central node. The exchange of medical images is realized by routing the data flow from one VPN channel to the other[[24]](#footnote-25). Another approach is used in the project MEDIMED, which is held by Masaryk University in Brno. Instead of a dedicated VPN channel, they use Secure Sockets Layer (SSL) encryption over the standard TCP/IP communication. Regional hospitals and healthcare providers are interconnected via the MEDIMED servers, as presented by Slavicek et al. [105]. In other countries, cross-border teleradiology was tested in projects of Baltic e-health, R-Bay and others, which were published by Ross et al. [106] and Saliba et al. [107]. These projects focused on the sharing of medical images, as well as other knowledge and information.Access to a wide range of medical images is needed for the research of new processing and diagnostic methods, rare diseases, developing new detection algorithm, etc. DICOM records are "de-identified" (identification of patient records are deleted and only the date of birth and other data are kept) or anonymized (additional information is manipulated to prevent disclosure) in order to protect sensitive personal data but keep important information for research purposes. The Globus MEDICUS project, published by Erberich et al. [108, 109], is based on Globus Toolkit middleware in order to federate clinical and research application via a grid computing infrastructure. The project has been in hibernation since 2008 and no further development has been published[[25]](#footnote-26). Similar efforts were made by the Medical Data Manager project, which used gLite grid middleware and was published by Duque, Montagnat et al. [110, 111] [[26]](#footnote-27) or the MediGRID project, which was published by Krefting et al. [112, 113]. Additionally, the processing of images within selected use cases that are supported by a grid computing infrastructure was introduced [113]. The Health-e-Child project aimed to interconnect research institutions and hospitals in United Kingdom, France and Italy for the purpose of a grid-based healthcare platform for pediatric health-care [114]. The Neurist project developed architecture, connecting clinicians and researchers in order to improve research and the treatment of cerebral aneurysm. This provided the tools to analyze and interpret patient data and gave researchers access to a set of aneurysm data, published by Benkner et al. [115]. The SEAGRIN research project aimed to share knowledge, mainly for educational purposes, in semi-formally described semantics. Kuba et al. published this proposal and its implementation [116]. The storing of sensitive medical information, which is even de-identified or anonymized, is usually restricted. This led to the idea of storing such information within trusted institutions, e.g., hospitals, and to move and facilitate the deployment of grid services that store medical data to that institution. For example, pre-installed virtual machines can contain grid services and are deployed as a sealed grid, as proposed by Kuba et al. [117].

To summarize this section, in past years, digital medical image acquisition, storing, exchanging and processing has become common and it currently uses distributed computing techniques. Several efforts have been made to implement medical data management within grid or cloud infrastructures for research purposes and to integrate them with production infrastructures. Security is solved by authentication and authorization mechanisms, as well as by encrypting data and/or de-identification or anonymization but keeping minimal information that is required for research purposes. A related question is how easily the previously mentioned grid-based technologies can be integrated with current systems in hospitals or institutions. The following section describes selected methods that are used to integrate a pilot deployment of Globus MEDICUS with the current regional system for exchanging medical images - MEDIMED.

### 3.1.1 Methods to Share Medical Images in Grid

The Globus toolkit belongs to a group of the most used grid middleware (see section 2.3.3). The core service of Globus Toolkit is GridFTP – grid extension to File Transfer Protocol (FTP). This implements strategies such as *stripping data* into multiple pieces; the *parallel transfer of data,* utilizing stripped data parts to be transferred via different channels; *partial file transfer,* some applications may not need to access the whole file but rather, a smaller portion of it, etc., as described by Foster et al. and Allcock et al. [118, 119]. Other core services are Replica Location Service, which aims to localize data, and Globus Resource Allocation Management (GRAM), which provides web service and proxies to the lower level job scheduler’s implementation [118].

Next to core services, the domain-specific services might be implemented for the purpose of an application that uses the Open Grid Service Architecture (OGSA). Globus MEDICUS [108, 109] contains a DICOM Grid Interface Service (DGIS) and integrates the open-source PixelMed™ Java DICOM Toolkit[[27]](#footnote-28) into a web service, communicating via the DICOM protocol. Furthermore, it forwards queries to the underlying services within the Globus toolkit. DGIS acts as a gateway to a grid infrastructure. As communication via the DICOM protocol is not secured, it is recommended that the DGIS be installed on the location of the PACS system or the DICOM ready modality or software. When a DICOM study is uploaded into DGIS, it is anonymized and stored. A record is made into another service’s Meta Catalog service, which resides in the same domain or anywhere in the grid that is accessible via the Globus Toolkit. Such an anonymized database of DICOM records can be used to query via the DGIS interface and to, for example, integrate with web-based applications, showing records for research purposes. Furthermore, authentication and authorization can be achieved in this level. To integrate this system with an existing system for sharing the medical images (e.g., the MEDIMED project [105]), the special client software, "RediMed console", needs to be installed next to the DGIS. It needs to be configured as a local PACS system, whose records might be exchanged to other MEDIMED participants. The results of this particular deployment and integration are presented in section 4.1.

## 3.2 Voice Science

With the introduction of objective data analysis and laryngoscopy methods, voice science emphasized the cooperation among laryngologists, speech pathologists and voice teachers. The human voice ranges from 50 Hz to about 1,000 Hz, but there are wide individual variations. For the analysis of a digitally recorded voice, either habitual or singing, the Discrete Fourier Transformation (DFT) is used to produce a frequency and amplitude analysis of the recorded input voice samples. One of the most used class algorithm to compute DFT is the class of Fast Fourier Transformation (FFT) with computational complexity  [120, 121]. The result of the analysis can be visualized in a voice range profile and the significant difference between an untrained and trained voice can be seen. Furthermore, some disorders can be quantitatively seen [122, 123].

Another method that is used to analyze vocal chords is laryngoscopy. The videostroboscopy and high-speed video in laryngoscope methods produce videos, which can be analyzed for the real movement of vocal chords. The videokymography method, which was introduced by Švec et al., complements the videostroboscopy. It allows the visualizing and analyzes of the movement of vocal cords. These movements are recorded by a high-speed camera on a standard TV or monitor, with an artificial image that is built from a recorded sequence of selected section [124, 125].

In the case of a recorded sound and further analysis, there is a question about how such a service can be integrated in a grid computing or cloud computing environment in order to provide access to a complex application for non-technical voice specialists. Additionally, analytical software was already developed and calibrated for selected types of microphones in the MS Windows platform [126, 127]. Therefore, I proposed and implemented a method that provides remote access to the analytical software. Section 3.2.1 describes how the analytical software was customized with a Remote Desktop Protocol (RDP). Results are described in section 4.2. A similar approach can be used for processing video recordings from a laryngoscope, however, the practical limits are discussed in section 5.

### 3.2.1 Methods for Remote Analysis of the Human Voice

Terminal access to some remote computational capabilities, e.g., remote command-line or remote execution, is another integration strategy that is used for some remote infrastructures. Secure Shell (SSH) is used to establish a secure channel via an unsecured network (e.g., the Internet) from an SSH client to a SSH server. This is a basic method that is used to access a grid computing infrastructure.

Remote Desktop Protocol (RDP) is a proprietary protocol that is used for desktop sharing. It was primarily developed in a Microsoft Windows platform, however, today, clients and servers exist for several other platforms. Next to remote command-line, remote execution allows the accessing of remote graphical desktop environments. The software for parameterized Voice Range Profile (ParVRP) and Voice Range Profile in Real time (RealVoiceLab) was already developed and calibrated for selected types of microphones in an MS Windows platform by Fric et al. [126, 127]. Its implementation is carried out in an MATLAB environment, utilizing Signal Processing Toolbox[[28]](#footnote-29). It is compiled with a MATLAB Compiler and distributed as an executable.Instead of migrating the application into some compatible platform for grid middleware, a virtual machine was introduced and access to the software was provided via a RDP protocol. RDP itself contains the redirection of several services, e.g., sound recording or drive access. As the default sound recording redirection introduces some sound degradation without control, I proposed, implemented and integrated a custom RDP plugin with the ParVRP and RealVoiceLab software in order to redirect the sound recording without the loss of information. The technical details are in Appendix .

The computation of frequencies and amplitude from the recorded samples utilizes the effective Fast Fourier Transformation, which has a time complexity of . The benefit of deploying such an application in distributed infrastructures is the immediate access to updated software. It also allows the collection of anonymized records of voice samples with analyzed results for further research and education purposes.

This type of application can be packaged as a virtual machine template and configured within different types of cloud infrastructures. Together with a script or web portal, the on-demand deployment can be automated. The client part (RDP client) needs to connect to the appropriate instance. The results of such a deployment are discussed in section 4.2.

## 3.3 Computational Physiology

A mathematical formalization of the fundamental knowledge and relation among a biological system – a mathematical model - is used as a base abstraction in order to utilize the current discoveries of the genomics and proteomics. It is also used to formalize the knowledge and construct a "Physiome Model". By definition, a model is the simplification of a complex reality.

Constructing the models and integrating them into a complex entity, which can be used for further purposes, is schematically illustrated in Figure 3.2. The measurements are carried out in laboratories or in hospitals. Lumped parameter models are usually represented as ordinary differential equations and differential algebraic equations. They characterize the reality as a topology of discrete elements. The imaging methods for processing and analysis (section 3.1) are used to construct 3D models from segmentation and generate mesh representations that are connected to physical principles.

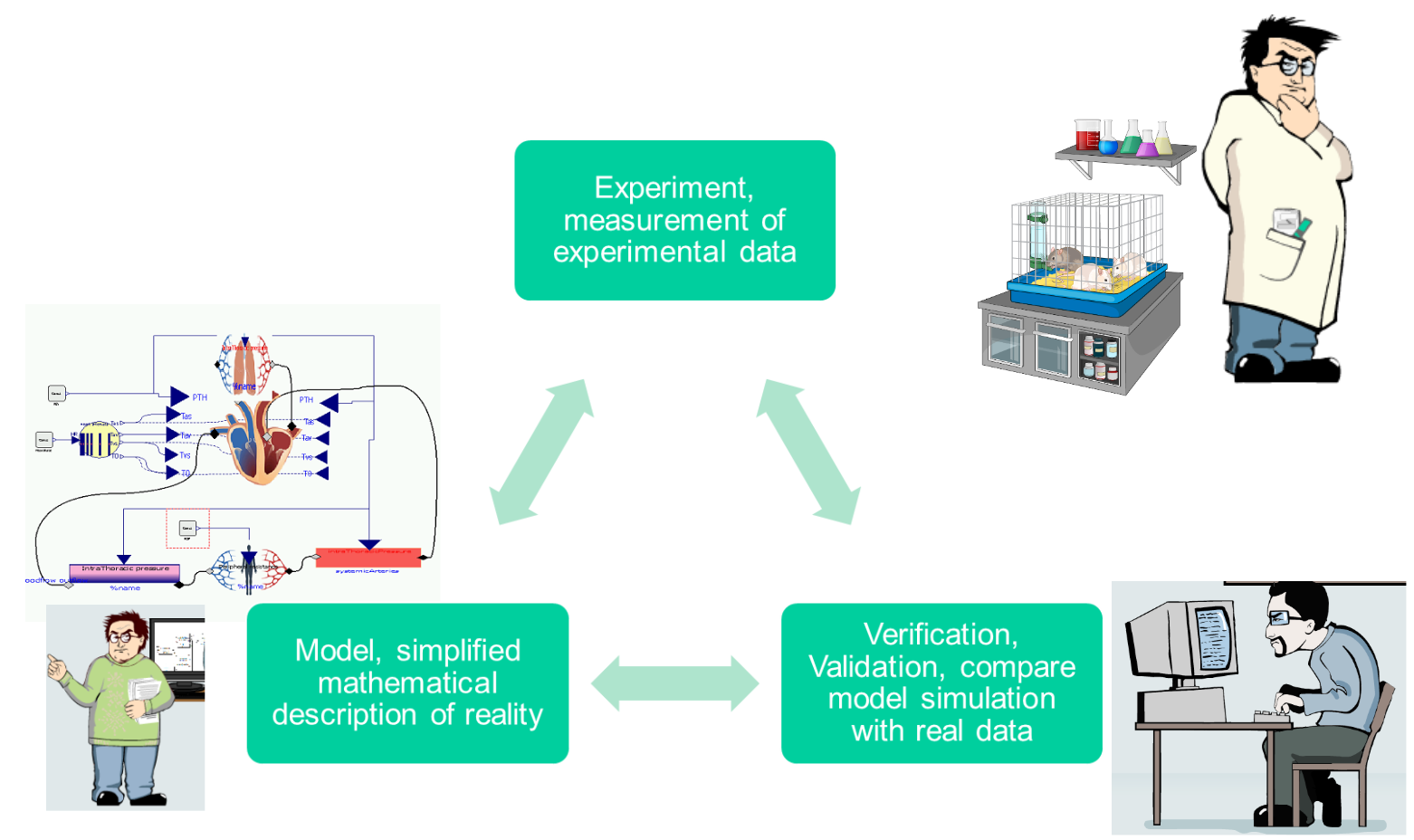


Figure 3.2: Schematic illustration of the scientific process. The experiments produce data that are interpreted and a hypothesis is formalized as a model. Validation compares the model simulation with the experiment, if the model satisfies the criteria - if it is in agreement with real experiments, then the validated model can be used for other purposes.

The application of mathematical modeling techniques towards biomedical research is sometimes called systems biology. This approach combines the reductionism and integration, as denoted by Kohl et al. [128]. Application towards clinical practice includes the quantification of the diagnostic index or treatment strategy. It is a goal to develop tools, database models and methods of several Physiome projects, e.g., VPH-Physiome project presented by Hunter et al. [129].

### One of the earliest complex and integrative modeling efforts was a model of circulation and its regulation, which was published by Guyton et al. in 1972 [130]. It continues as a "Human Model" or "HumMod" via derivative and technological upgrades, as introduced by Hester et al. [131, 132], with a focus on integration efforts. A different approach of modeling the human physiology is a database of smaller models, which focus on some particular physiological phenomenon. For example, the NSR Physiome project introduces a JSIM[[29]](#footnote-30) Java-based simulation system in order to support modeling in physiology. A repository of several hundred models was published using this system [133]. A similar effort is made by the IUPS Physiome project and repositories of the models are based on XML standard languages CellML and FieldML [134, 135]. The Systems Biology Markup Language (SBML) is used for modeling a biological system at the level of biochemical reaction and regulatory network. Another database collects several hundreds of curated and non-curated models [136, 137].3.3.1 Modeling Methodology

The methodology of formalizing mathematical models is influenced by the abilities of the underlying modeling language that is used. The Modelica language is an object-oriented, equation-based and acausal modeling language, which is standardized by the Modelica association[[30]](#footnote-31). JSIM, CellML, SBML or HumMod are domain specific languages and the tools that are able to work with them are primarily developed within physiological or systems biological communities. Other authors use commercial or industry standard tools for mathematical modeling and computing. For example, Kofranek et al. described Guyton’s 1972 model in MATLAB® Simulink [138] and the derivative HumMod in acausal object-oriented Modelica language [139, 10]. Fernandez et al. described models of a cardiovascular pulsatile system using MATLAB Simscape [140] and recently, in Modelica [141].Thus, there is an open debate as to whether in-house domain-specific language and tools, like JSIM, CellML and FieldML, SBML or HumMod, have reached their capabilities for representing complex models. Only the HumMod reached the integrative approach, building a complex integrative model of human physiology using a lumped parameter approach. I contributed to the idea of key features, which involve the acausal modeling technique and object-orientation. This keeps the complex model structure decomposed into understandable and maintainable parts and allows the complexity of models like HumMod to be covered.

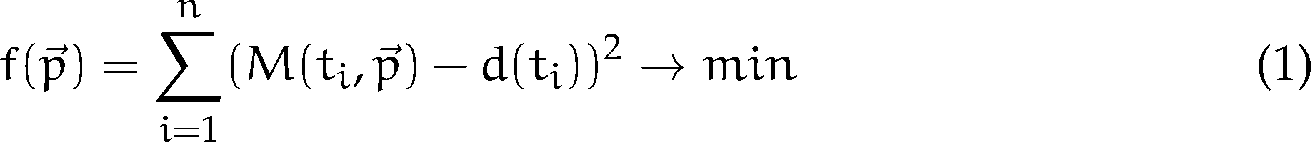
The paper [5] *Modeling of Short-term Mechanism of Arterial Pressure in the Cardiovascular System: Object-Oriented and Acausal Approach* in Appendix  published disputation about causal and acausal approach in using Modelica for modeling pulsatile cardiovascular system (CVS) and possible enhancement for more complex models.

The paper [6] *Simple Models of the Cardiovascular System for Educational and Research Purposes* in Appendix  , published detailed methodology of modeling lumped parameter pulsatile CVS in Modelica.

A common guide to the Modelica language and its capabilities can be found in the published works of Fritzson [142] or the on-line works of M. Tiller [143].

### 3.3.2 Identification of Physiological Systems

Usually, some knowledge of the system - the structure - is available and unknown coefficients (parameters) remain unknown. Once the model is formalized and constructed, a further problem is to estimate the model parameters so that the model reproduces a real world system. This procedure is sometimes called system identification and the objective of the parameter estimation is usually to minimize the following function (to find the least amount of differences between the predicted and measured values):

(3.1)

where  is the vector of values of parameters,  is model simulated at time  with the given parameter values  and  is the measured experimental value at time . In general, mathematical models of biological systems are, in most cases, non-linear and some of them are non-differentiable. Therefore, global optimization methods must be used. Algorithmically, the problem of parameter estimation was shown to belong to NP-complete problems [144], which implies that the current exact algorithm is brute-force search – trying all possible values of the parameters and simulate the model with them, finding the minimum of the objective function 3.1. Further reading about parameter estimation and system identification can be found in published works, edited by Eykhoff [145], or in the published works of Khoo [146, p. 159].

The heuristic methods (evolution strategies), randomization methods (Monte-Carlo) and others can be used to find at least some solution in a reasonable time. Evolution strategies have been used identified as robust, having the potential to utilize parallel computing, as shown by Moles at al. [147]

Parameter estimation and further analysis methods are part of specialized mathematical software. For example, Pruet et al. used Metropolis algorithm to produce a distribution of parameters in order to calibrate a model of the human cardiovascular physiology. This was further tested against the predictive ability of circulatory failure and statistical methods of the software, Wolfram *Mathematica,* were used [148]. The iterative improvement method in the software, MATLAB Simulink®, was used by Takahashi et al. in their estimation of two parameters of a simple cardiovascular model [149]. Furthermore, Abbass et al. compared several methods in their estimation of multiple parameters of a cardiovascular system in MATLAB Simulink® [150].

Maffioletti et al. published a GC3Pie framework, which utilized evolutionary algorithms. They also introduced workflow to identify parameters of models for economical predictions, using grid computing [151]. Humphrey et al. calibrated hydrology models, utilizing a commercial Windows Azure cloud computing infrastructure. This showed a significant speedup on the modified dynamically dimensioned search algorithm [152, 153].

### 3.3.3 Methods for Parameter Estimation

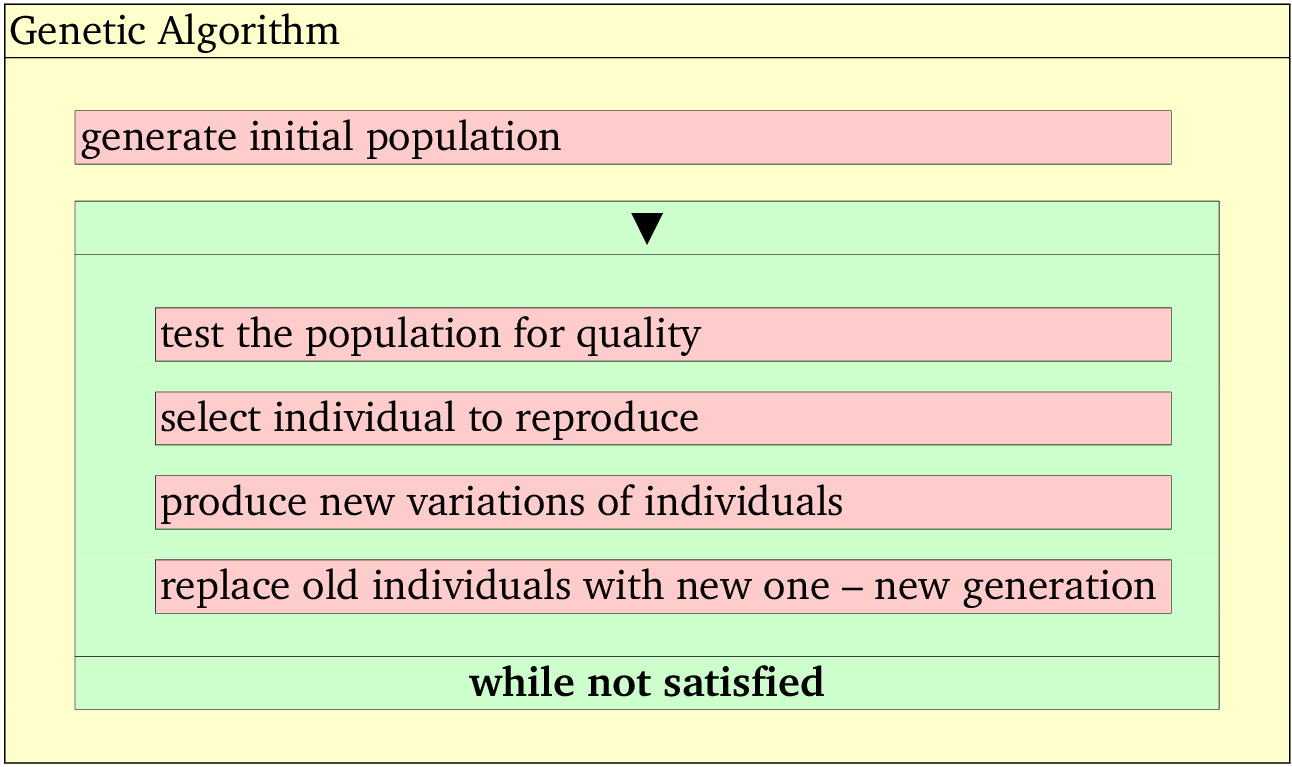


Figure 3.3: Kopenogram of a genetic algorithm.

An evolutionary algorithm can be used as a heuristic strategy for finding a global minimum or maximum. It can also be used to estimate the parameters of a model. A genetic algorithm is a type of evolutionary algorithm, which encodes individuals as binary string. It was introduced by the likes of Holland [154]. These algorithm steps are schematically presented in Figure 3.3.

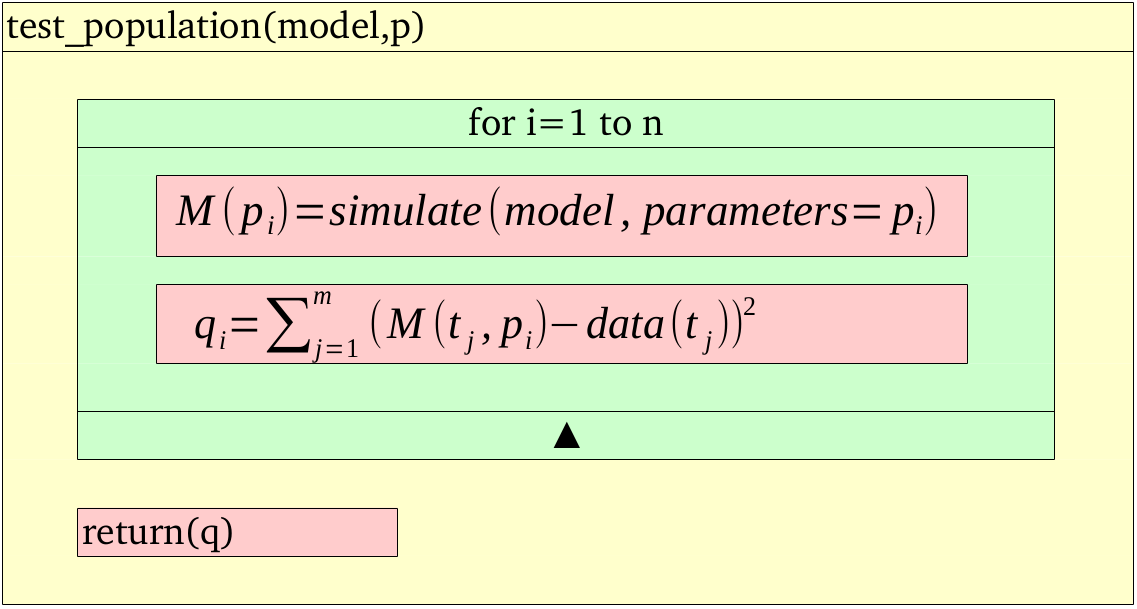


Figure 3.4: Kopenogram of a genetic algorithm and the specific test of a population for quality, in the case of parameter estimation. The model is simulated according to individual  with parameters  and the quality  is counted per the objective function 3.1. 3.1

The iteration within the loop " *while not satisfied*" depends on the previous iteration and thus, it cannot be parallelized. The step *test the population for quality* has an algorithmical structure in Figure 3.4 for the parameter estimation of tasks. Each iteration in the loop "*for i=1 to n*" is independent and therefore, loop parallelism (section 2.2.1) can be utilized and implemented here.

#### Architecture of System for Parameter Estimation

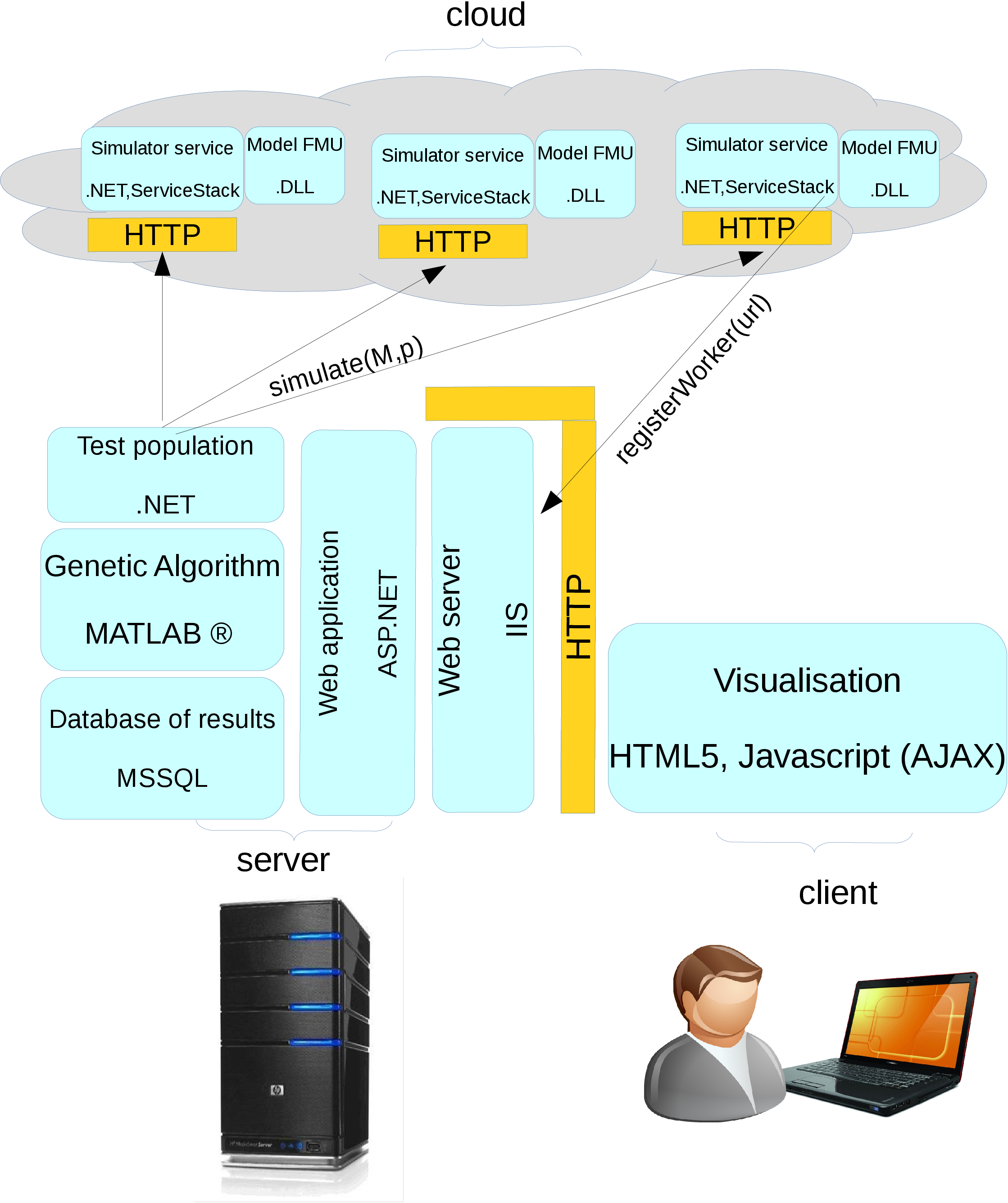


Figure 3.5: Architecture of a system that employs genetic algorithm and distributes the task *simulate* into a cloud computing environment.

The proposed architecture of the system for parameter estimation (Figure 3.5) was influenced by the need of some interactivity and for the overall accessibility for users, which is fulfilled by the web UI. The key part of the system is the model that is exported into a binary platform-dependent library. The specific model of a studied system that is implemented in Modelica is exported into a standard Functional Mockup Unit (FMU). This is standardized XML metadata, packaged together with a binary library .DLL (or .SO), following a standardized API [155][[31]](#footnote-32). In the time of writing this thesis, the most stable Modelica tool export was Dymola[[32]](#footnote-33), which exports to FMU for a MS Windows platform. The parallelization is implemented using threads. For this, a *test\_population* method is used, which, within a loop, follows a fork/join pattern – the created threads simultaneously ask for simulation results with a parameter set and the main process waits until all of the results are returned before computing the full vector of quality evaluation .

### Packaged with a .NET ServiceStack framework[[33]](#footnote-34), it exposes a simulation functionality as a RESTful web service, which can be accessed and orchestrated by the *test\_population* algorithm. The implementation of genetic algorithm is reused from MATLAB ™and, with a database of results in a SQL database, is integrated with an ASP.NET web application. This presents a web user interface and functionality to a user. Section 4.3 describes the result of applying the methods and deploying the designed system in a local cluster and cloud computing infrastructure4.3.3.3.4 Parameter Sweep

After the parameter estimation, further problem arise regarding the structural identification and analysis of sensitivity to the estimated parameter values [146, p. 176].

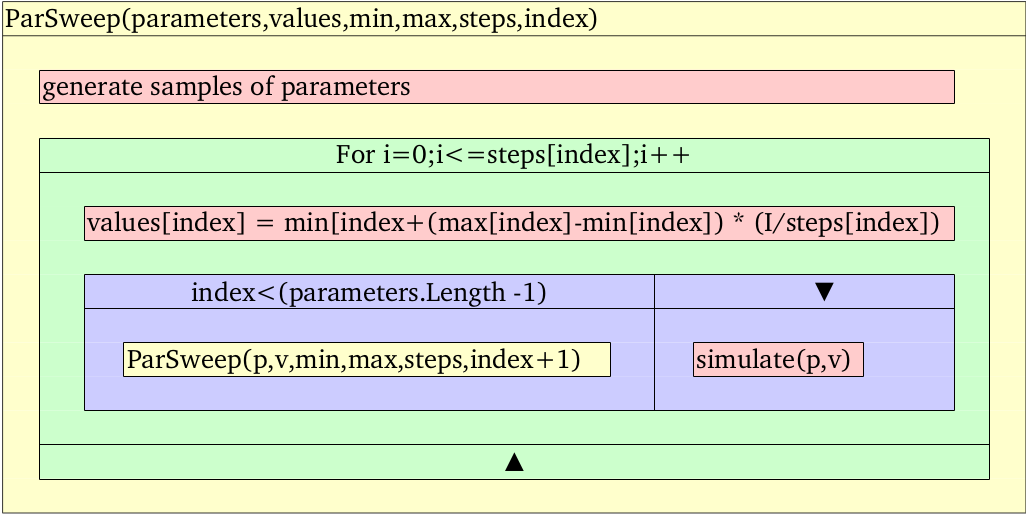


Figure 3.6: Kopenogram of a recursive parameter sweep algorithm. ,,, and  are arrays with the same dimension that hold the parameter name, value, starting and stopping value and the number of steps that have to be performed between the starting and stopping value per each . 

Parameter sweep (PS) is one of the techniques that is used for a sensitivity and uncertainty analysis, which is based on the changing selected parameters, simulating the whole model and quantifying the change on model behavior with different parameters. An uncertainty and sensitivity analysis tries to determine how a change in the value of a parameter contributes to the model output and how the estimation of parameter values is robust against errors of real data measurements. The various methods for carrying out an uncertainty and sensitivity analysis have been published, e.g., in reviews by Helton et al. [156] or in publications by Saltelli et al. [157, 158].

The recursive algorithm of a parameter sweep for exploring parameter space (in Figure 3.6) generates a tremendous number of simulations. Presuming that *simulate* operation takes a constant time for any parameters (which, in general, is not true), the time complexity of PS is  where  and  is number of parameters to be swept. For example, for 1,000 values for each parameter: . The large number of distinct simulation can take a tremendous amount of time on a single computer. However, in contrast to parameter estimation, each simulation is independent and a PS algorithm is determined as embarrassingly parallel. It is implemented in many grid computing projects and workflows, e.g., P-Grade portal, as published by Kacsuk et al. [98].

To perform parameter sweep algorithm on the models of human physiology in Modelica language, an export from Modelica is needed. The FMU standard, which is supported by many tools, exports FMU as

a BOINC platform [70][[34]](#footnote-35). Following the task parallelism and master/worker programming model (mentioned in section 2.2.1), it is customized. The Modelica model, which is exported as FMU for a Windows platform, is integrated with a BOINC wrapper. As a whole, it is integrated into a BOINC platform, which is deployed on a server, as seen in Figure 3.7.

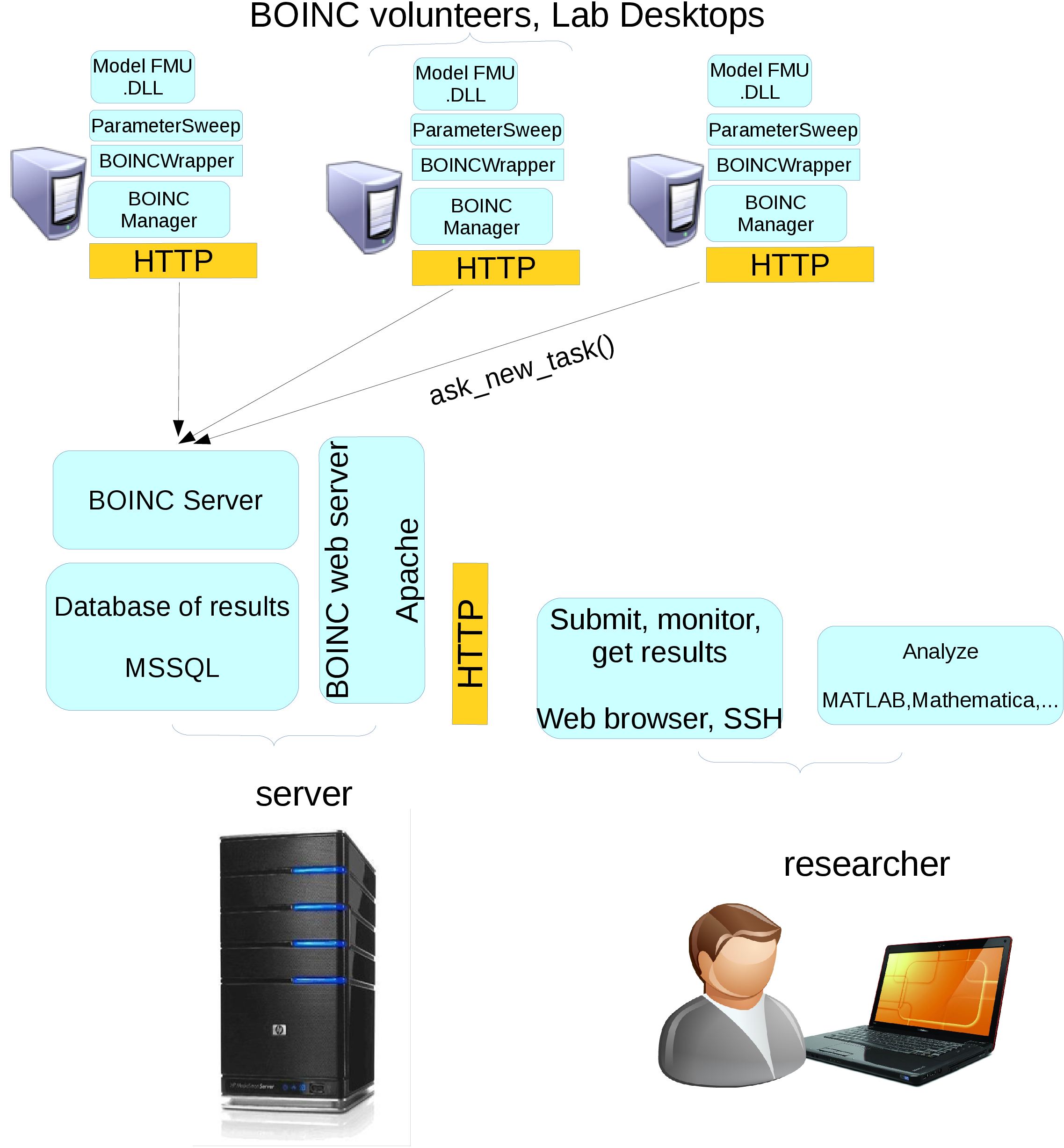


Figure 3.7: Architecture of a parameter sweep application. The whole parameter space is divided into smaller spaces, which are resolved by the BOINC workers

The results are described in section 4.3.

**Chapter** **4**

# Results

In previous chapters, we introduced different methods that are available for selected use cases in biology and medicine research. As each of the use cases and available systems were proposed on different operating system platforms, architecture and/or middleware, the virtualization was utilized to build the virtual infrastructures for the purposes of each project. The paper [3] *Infrastructure for Data Storage and Computation in Biomedical Research,* in Appendix  , describes the result of establishing virtualization on a physical infrastructure in order to share computational power among different platforms.

## 4.1 Medical Images

The pilot infrastructure of several servers was installed in several institutions in Prague, Czech Republic. Globus Toolkit and Globus MEDICUS were installed on them. The paper [1] *Processing of Medical Images in Virtual Distributed Environment,* in Appendix 6, published details about the integration of Globus MEDICUS with a MeDiMed project. It concludes that such integration via the DICOM protocol is almost seamless. Furthermore, if such a grid-based system is joined with a production system for exchanging clinical DICOM data, it could be beneficial for researchers.

## 4.2 Remote Access to Voice Analysis

The paper [2] *Remote Analysis of Human Voice–lossless Sound Recording Redirection,* in the Appendix  , published technical details and results of customizing a RDP protocol for lossless sound recording redirection. It also discusses remote access via a remote desktop feature of Windows platform to an application in order to analyze the human voice and produce a voice range profile for further use.

## Additionally, the custom RDP plugin with ParVRP and RealVoiceLab software in order to redirect a sound recording without the loss of information was packaged as a virtual machine template. This was deployed in the pilot virtual infrastructure next to the test instance of Globus MEDICUS. The virtual machine template was also deployed to different cloud computing infrastructures. The first was deployed to the Amazon EC2[[35]](#footnote-36) and the second to the pilot scientific cloud, MetaCloud[[36]](#footnote-37), which was launched at the beginning of 2012. In the EGI Technical Forum 2012, such a comparison was presented to the user and technical community within CESNET and EGI organization [23].4.3 Parameter Estimation

The paper [4] *Parameter Estimation of Complex Mathematical Models of Human Physiology Using Remote Simulation Distributed in Scientific Cloud,* in Appendix  , published the architecture and measurement of a speedup that was achieved on estimating parameters of three different types of models - from the non-complex, medium-complex and complex model. It concluded that only medium-complex and complex models may benefit from the architecture, as the communication overhead may become major for simple models and decrease the overall performance.

Additionally, a scientific result was published in the paper [7] *Adair-based Hemoglobin Equilibrium with Oxygen, Carbon Dioxide and Hydrogen Ion Activity* in Appendix  ,where a mathematical model of hemoglobin integrating O\s\do6(2), CO\s\do6(2) and H\s\up6(+) binding based on theoretical principles, which were verified on the parameter estimation algorithm system[4], together with methods available in Wolfram *MATHEMATICA* 9.0[[37]](#footnote-38).Thus, the overall performance and speedup estimation were tested against the Modelica implementation of the complex physiological model, HumMod [139]; the Modelica implementation of a model of hemodynamics of the cardiovascular system, presented by Meurs [159]; the model of binding gases to hemoglobin, named as Matejak2014 [7] and the trivial model of a curve  with four parameters , defined as  and named as "SinusCurve". round-mode=figures,round-precision = 4 .

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Complexity | Name | T1 | T2 | T3 | T4 |  |  |
| High | HumMod [139] |  |  |  |  |  |  |
| Medium | Meurs2011[159] |  |  |  |  |  |  |
| Low | Matejak2014[7] |  |  |  |  |  |  |
| Trivial | SinusCurve |  |  | x | x |  |  |

Table 4.1: Time spent in different parts of the parameter estimation algorithm for one processor deployment. Genetic algorithm works with a population of  individuals for  generations. T1 – is the whole time of the computation, T2 – is the time of the computation, which can be parallelized, T3 – time spent within the worker node, T4 – time spent in simulation,  – computed as  and  is the theoretical speedup limit per Amdahl’s law () eq.(2.3).2.3.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Complexity | Name | T1 | T2 | T3 | T4 |  |  |
| High | HumMod [139] |  |  |  |  |  |  |
| Medium | Meurs2011[159] |  |  |  |  |  |  |
| Low | Matejak2014[7] |  |  |  |  |  |  |

Table 4.2: Same as Table 4.1 but measured on a local cluster deployment, with reduced communication overhead.4.1.

The computation time of a single simulation depends on the model complexity number of the compared values. Based on the findings, the simulations of the models were divided into four groups, depending on its demand to compute 1,200 simulations. Fraction  and the speedup limit per Amdahl’s law are stated in Tables 4.1 and 4.2.

The difference between  and  is an overhead, which was introduced by the network communication between the genetic algorithm and the worker nodes that were deployed in the cloud deployment, provided by CESNET NGI department METACENTRUM[[38]](#footnote-39). The network overhead can be eliminated in serial implementations by directly integrating the simulation into a genetic algorithm. Therefore, Table 4.3 considers and compares a hypothetical serial execution time, which is estimated without the network overhead.4.3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Distributed In Cloud | | | | | Distributed In Local Cluster | | | |
|  | Overhead | | Est. Serial | | Overhead | | | Est. Serial | |
| Model Name | T2-T3 | Fraction |  |  | T2-T3 | | Fraction |  |  |
| HumMod [139] |  |  |  |  |  | |  |  |  |
| Meurs2011 [159] |  |  |  |  |  | |  |  |  |
| Matejak2014[7] |  |  |  |  |  | |  |  |  |

Table 4.3: Comparison in cloud deployment vs. local cluster deployment of communication overhead. Its fraction in the whole computation was introduced by the network transfer and latency. The estimated time and speedup, if the worker is replaced by a serial version of computation without communication overhead, is:  – estimated time of serial version of computation.  – estimated speedup of serial version of computation against the parallel on one processor.

The speedup was measured on 10 - 60 CPUs and compared in order to predict the speedup, as seen in Figure 4.1. Different measurements were carried out using 80 and 160 CPUs, as seen in Table 4.4.

|  |  |  |  |
| --- | --- | --- | --- |
| Complexity | Name | S(80) | S(160) |
| High | HumMod [139] |  |  |
| Medium | Meurs2011[159] |  |  |
| Low | Matejak2014[7] |  |  |

Table 4.4: Scalability on 80 CPUs and 160 CPUs of genetic algorithm with a population of 640 individuals for 20 generations on different clusters, performed about 10 times more simulation. The speedup estimated is based on the T1 column of Table 4.1.4.1

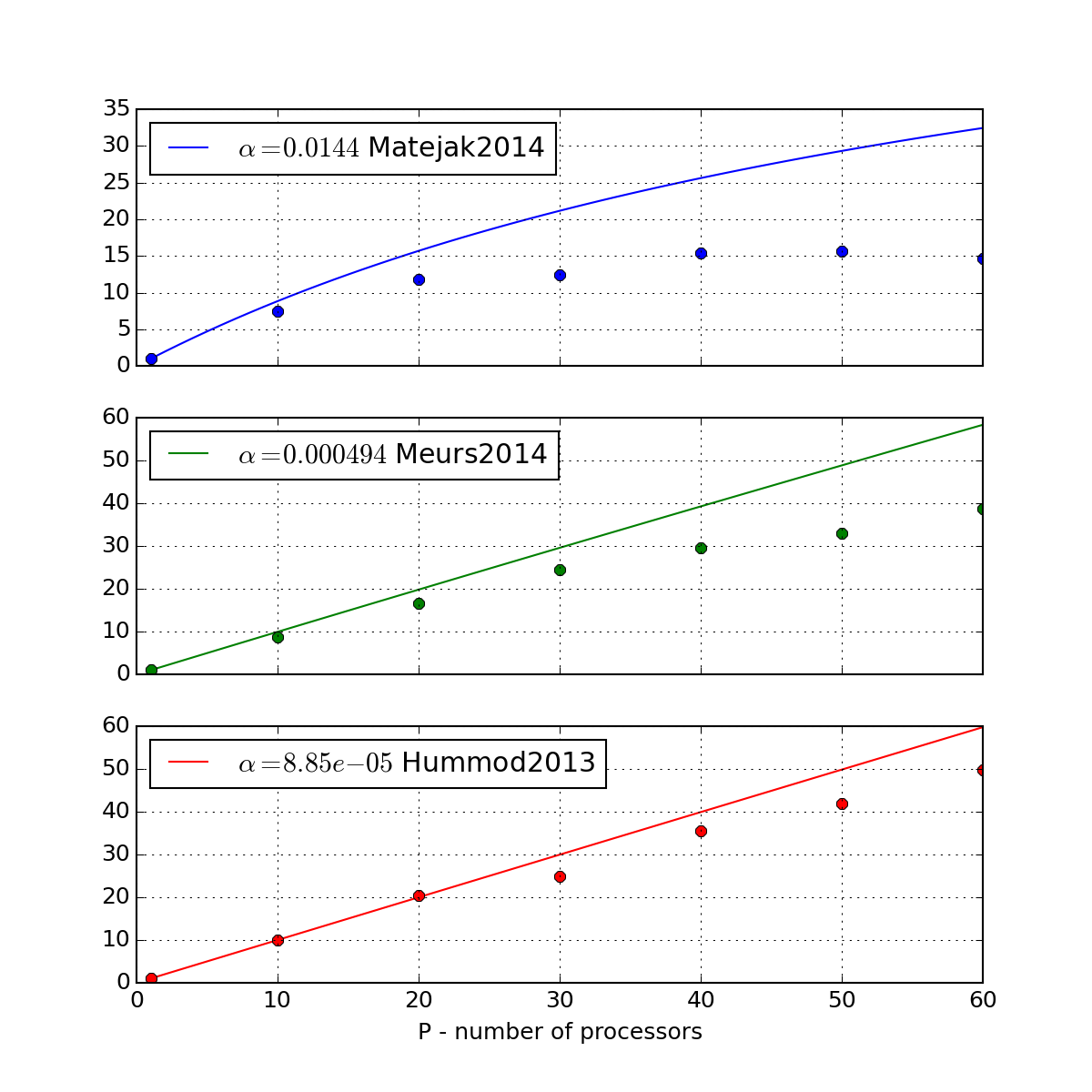


Figure 4.1: Estimated speedup (lines) per Amdahl’s law (eq.2.3 [38]) for different  of several Modelica models and real measured speedup (points) on cloud deployed on 1-6 virtual machines on physical hardware (2x 6-core Intel E5-2620 2GHz, 1Gbit/s Ethernet.) 2.338

To summarize the results, the simple models scale up to the 20 processors with a speedup of 15. The medium scales up to 80 processors with a speedup of about 72 and the complex models scale up to 160 processors with a speedup near to 160. Practically, after 20 generations, a good approximation was obtained and after another 180 generations, a good estimation was obtained. This implicates that the computation time could be reduced from five days to 48 minutes, in the case of HumMod and from 18 hours to 15 minutes, in the case of the medium complex model.

The deployment on local cluster reduces the communication overhead. However, in order to compute concurrently, it is limited by the available processors. Thus, it should be considered for boundary cases like the simple models. The following statement can be made:

•If the alpha fraction is major, then a serial computation of parameter estimation algorithm, without communication overhead, will perform best. This is the case for the trivial function.

•If the alpha fraction is minor, but the network overhead is still major, a computation on a local cluster or virtual HPC cluster should be considered. This is the case for the low complex model simulation, "Matejak2014"[7].

• If the alpha fraction is minor and the network overhead is also minor, then the distributed computation, e.g., in a cloud-computing environment, is worth using. This is case for the medium and high complex model simulations of "Meurs2011"[159] and "HumMod2013"[139].

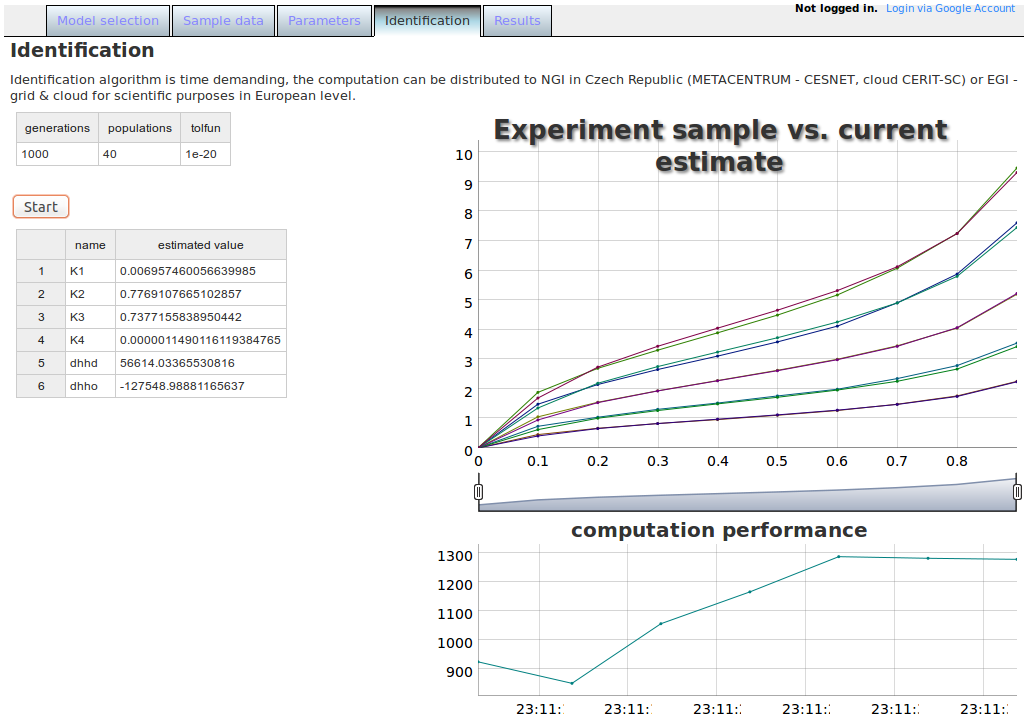


Figure 4.2: User interface of a web application for parameter estimation. In this case, for the model, Matejak2014 [7]. The top left table lists the parameters for the genetic algorithm (1,000 generations with a population size of 40 and a cumulative change in the generation limit, which ends the algorithm earlier). The middle left table shows the model parameters and current best values, which fit the sample data. The chart shows how the sample data fits with the model simulation. The right bottom chart shows the performance of computation in a number of model simulations, per second.7

### 4.3.1 Parameter Sweep

The desktop grid BOINC system was established for parameter sweep application. The established project, *Physiome@home,* and its project web page, <http://physiome.lf1.cuni.cz/ident3/physiome>, manages workunit tasks, which are sent to and executed by BOINC workers. The worker application is a packaged model that is exported as FMU for a Windows platform and wrapper application, which communicates with the BOINC manager on the desired volunteer computer.

### 4.3.2 Remote Simulation and Local Visualization

An extra outcome of an architecture for parameter estimation is a hybrid web simulator system, where a single instance of a worker node is utilized as a back-end for the simulation engine. The front-end, presented as a web application, is implemented using HTML and Javascript language and visualizes the simulation values that are obtained from the worker node performing the simulation, as seen in Figure 4.3. The results were published in [9] and popularized in [21].

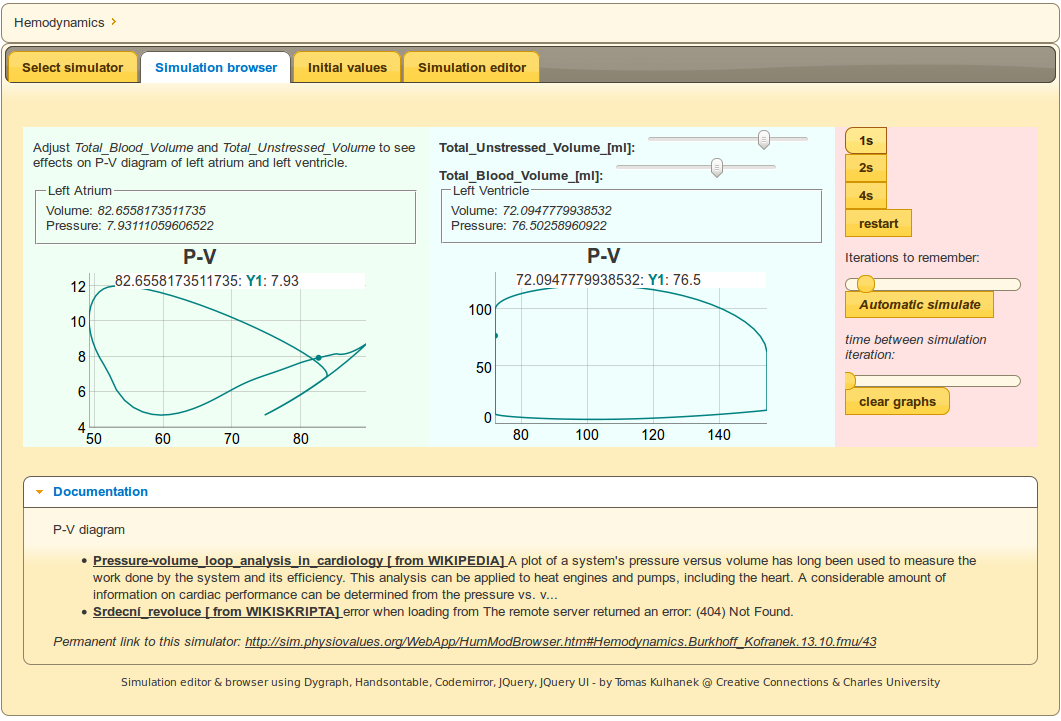


Figure 4.3: Web application to visualize simulation - in this case, pressure volume diagrams of the left atrium and left ventricle of the model of hemodynamics of a cardiovascular system.

**Chapter** **5**

# Conclusion

## 5.1 Discussion

The result presented in section 4.1 is an example how a standard format and protocol DICOM is utilized to integrate current production system in order to exchange medical images (MEDIMED [105]) and a grid-based solution (Globus MEDICUS [109]). Remote Desktop Protocol (RDP) is also a key point in integrating the application of a voice analysis [126] into a remote environment, which is accessible via the Internet. This is presented in section 4.2. In the case of parameter estimation, a key factor is the standard Functional Mockup Interface (FMU) [155], which allows the control and simulation of a physiological model in a customized tool that is not related to modeling technology. This is presented in section 4.3.

The selection of a joint element also increases the chances of reusability of such a system in future development, when requirements usually change and the reconstruction of a system or architecture is needed. For example, the presented solution, which is based on Globus MEDICUS, is, in general, a data warehouse that stores one or more copies of DICOM images. However, federated files and metadata that are stored within home institutions, which only share only network infrastructure to interchange the DICOM studies, seems to be a preferred and more acceptable solution by hospitals. Thus, in their further development, the authors of Globus MEDICUS followed a way of federation of medical images that are stored within home institutions, as published by Chervenak et al. [160]. The grid computing infrastructure seems to be suitable for research and educational purposes, rather than for clinical use.

In the case of remote voice analysis, remote access to an application via network protocol keeps the majority of user experience, as presented in section 4.2. Such a service can be deployed on any web server and the occasional need to educate or perform a higher number of analyses concurrently can be satisfied with cloud computing deployment. The application process for sound signal is currently being analyzed by Fast Fourier Transformation algorithm quite effectively. Another challenge is to analyze a sound signal that is connected with high-speed video or videokymography methods, which need to transfer, process and store large amounts of data. Such a future application might utilize the results of grid-based systems for sharing medical images.

In the case of the application for parameter estimation, presented in section 4.3, the computation is sensitive on communication overhead. For simple models, local High Performance Computing (HPC) resources are most beneficial. For medium and highly complex models, the deployment of worker nodes into a cloud computing environment is worth considering. The application for parameter sweep is embarrassingly parallel and suitable for High Throughput Computing (HTC), which is the focus of current grid computing infrastructures.

When porting an application to a grid environment, one important decision to consider is the platform of the used system. The architecture, which involves computational nodes that are deployed in a cloud computing infrastructure, is influenced by the fact that the model implementation is exported from a third party tool to the standard FMU library, as mentioned in section 3.3.3 for the MS Windows platform. This determines the platform of the worker node and the virtualization – or, in the case of parameter estimation, cloud computing is utilized on a prepared platform with a MS Windows Datacenter license. In the case of parameter sweep, a desktop grid computing BOINC worker and application for a MS Windows platform is only prepared for volunteers with the compatible system. To utilize the service grid infrastructure, an export of the model into a FMU library and implementation of the wrapper service must be done in the grid computing platform, which is usually a Linux based system. Another option is to use WINE[[39]](#footnote-40) – a compatibility layer that is capable of running Windows applications on several POSIX-compliant operating systems, such as Linux, Mac OSX and BSD. For smaller types of application and scientific community with their own tools, the question is whether or not to invest on porting their tools to a grid-specific platform and parallel programming model. In the case of integrating with a service grid middleware or with desktop grid framework, expert knowledge is needed to configure and customize the system. This is the case for the sharing of medical images (presented in section 4.1) and for parameter estimation and parameter sweep, which was tried with the desktop grid approach - BOINC framework 4.3.1. Virtualization facilitates the integration effort, as presented in the case of remote analysis of the human voice (section 4.2) and in the case of deployment of worker nodes in a cloud computing environment for parameter estimation (section 4.3). Based on previous results and ideas, the answer to the questions from the section 1.1 can be formulated:

•*Is it beneficial to utilize grid computing and cloud computing technology for the processing of medical information and how?*

Grid computing and cloud computing may significantly speedup parameter studies of medium and complex models in computational physiology. Such a speedup might influence its applicability in clinical use. For the case of sharing and processing medical images or analysis of voice signals, grid computing or cloud computing introduces technology that facilitates a cooperation among a community of users from different geographically dispersed areas and facilitates the sharing of large data sets.

•*What are the limitations of processing medical information in grid or cloud?*

Limitations are given by the effort needed to integrate or port an application to carry out computation or share data. The cost of porting an application to cloud computing is reduced by virtualization technology rather than to a grid computing environment, which needs additional work in order to adapt the application for a grid computing platform and API.

From a programming model point of view, limitations are given by the theoretical features of algorithms and the problems to be solved. Grid computing and cloud computing are not general solutions for hard problems (NP-complete problems), as discussed in section 2.1. However, connected with non-exact methods, a concurrent processing of many tasks may bring an acceptable non-exact solution.

• *How can the grid computing and cloud computing influence the direction of biomedical research?*

There are research infrastructures that were established in order to coordinate biology and medicine research, e.g., the Integrated Structural Biology Infrastructure for Europe (INSTRUCT)[[40]](#footnote-41), European Life Science Infrastructure for Biological Information (ELIXIR)[[41]](#footnote-42), European Biomedical Imaging Infrastructure (Euro-BioImaging)[[42]](#footnote-43) and others, which technologically rely on grid computing and cloud computing infrastructures for science. The purpose of these initiatives is to understand high-level phenotypes from genomic, metabolomic, proteomic, imaging and other types of data. They also require multi-scale mathematical models and simulations, as noted e.g. by Hunter et al. [163] in his strategy for Virtual Physiological Human (VPH)[[43]](#footnote-44).The integration with multidimensional models of geometrical, mechanical properties and the time-dependence of the compartments’ data, which is taken from medical and biological repositories, is a challenge for current "physiome" projects. It might be very beneficial for the current most complex models of human physiology, which are based mainly on a lumped-parameter approach. The area of parameter study that is presented in section 4.3 may deliver more exact lumped-parameter models and practical application for clinical and further research towards patient-specific health care, in silico trails and drug discovery.

Based on the previous answers, another question can be formulated for further research in the technology domain:

*How can biomedical research influence the direction of grid computing and cloud computing development?*

One area of discussion about this theme is how to preserve scientific data in order to prevent the loss of them [164, 165] and how to facilitate access to computational resources for large amounts of small scientific group, which have limited resources to port, integrate or customize their current tools and processes – to support the "long-tail" of science. The "long-tail" movement was first noted and described by Anderson [167] in the business domain. The long-tail term comes from a feature of statistical distribution, e.g., pareto distribution, where only a few (e.g., 20% – head) elements have a high probability of some events (e.g., product being sold), while the rest (e.g., 80% – tail) have a small probability. Thus, most businesses focus on hits (20% of products, the 80-20 rule). The expansion of the Internet and its related technologies have caused reduced sales, marketing and delivery costs for the products from the niche (80% of products) – long-tail. A strategy that focused on these kinds of products became profitable and successful, e.g., for companies such as Amazon or Apple [167]. Cloud computing technologies seem to be customizable and may be an enabling technology to focus on long-tail science consumers, as noted by Weinhardt et al. [166]. As a service, a platform may be a type of access to a cloud computing infrastructure, bridging the gap between programming application and utilizing the powerful scientific infrastructure. This keeps the focus on domain-specific research as it is achieved by research portals and workflows for grid computing infrastructures, as presented in section 2.3.8.

## 5.2 Summary

This thesis presents the infrastructure, which, thanks to virtualization technology, joined several domain-specific tools in the field of sharing and processing medical images, performing real-time voice analysis and simulating human physiology.

A seamless integration of grid-based PACS system was established with the current distributed system in order to share DICOM medical images. Access to real-time voice analysis application via remote desktop technology brings this type of service to any computer that can connect to the Internet. A system to support the analysis and building of complex models of human physiology in the phase of parameter estimation and parameter sweep was introduced. Furthermore, additional computational nodes can be flexibly joined by starting the prepared virtual machines in a cloud computing deployment. The methodology of building complex models of human physiology was contributed with the use of acausal and object-oriented modeling techniques. Methods for conducting a parameter study were shown, as well as the parameter study of complex models that gain substantial speedup by utilizing cloud computing deployment, which makes such kinds of complex studies applicable in physiological and biological research.

**Chapter** **6**

# Appendices A-G

Included separately

Bibliography

[1]Tomáš Kulhánek and Milan Šárek. Processing of Medical Images in Virtual Distributed Environment. In *Proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship*, page 10. ACM, 2009. [doi:10.1145/1551722.1551732](http://dx.doi.org/10.1145/1551722.1551732).

[2]Tomáš Kulhánek, Marek Frič, and Milan Šárek. Remote Analysis of Human Voice–lossless Sound Recording Redirection. *Analysis of Biomedical Signals and Images. Proceedings of 20th International EURASIP Conference (BIOSIGNAL)*, pages 394–397, 2010. URL: <http://bs2010.biosignal.cz/papers/1092.pdf>.

[3] Tomáš Kulhánek. Infrastructure for Data Storage and Computation in Biomedical Research. *European Journal for Biomedical Informatics (EJBI)*, 6(1):55–58, 2010. URL: <http://www.ejbi.org/img/ejbi/ejbi2010-1.pdf>.

[4] Tomáš Kulhánek, Marek Mateják, Jan Šilar, and Jiří Kofránek. Parameter Estimation of Complex Mathematical Models of Human Physiology Using Remote Simulation Distributed in Scientific Cloud. In *Biomedical and Health Informatics (BHI), 2014 IEEE-EMBS International Conference on*, pages 712–715, 2014. [doi:10.1109/BHI.2014.6864463](http://dx.doi.org/10.1109/BHI.2014.6864463).

[5] Tomáš Kulhánek, Jiří Kofránek, and Marek Mateják. Modeling of Short-term Mechanism of Arterial Pressure Control in the Cardiovascular System: Object Oriented and Acausal Approach. *Computers in Biology and Medicine*, 54(1):137–144, 2014. [doi:10.1016/j.compbiomed.2014.08.025](http://dx.doi.org/10.1016/j.compbiomed.2014.08.025).

[6] Tomáš Kulhánek, Martin Tribula, Jiří Kofránek, and Marek Mateják. Simple Models of the Cardiovascular System for Educational and Research Purposes. *MEFANET Journal*, 2(2):56–63, 2014.

[7] Marek Mateják, Tomáš Kulhánek, and Stanislav Matoušek. Adair-based Hemoglobin Equilibrium with Oxygen, Carbon Dioxide and Hydrogen Ion Activity. *Scandinavian Journal of Clinical and Laboratory Investigation*, 2014. [doi:10.3109/00365513.2014.984320](http://dx.doi.org/10.3109/00365513.2014.984320).

[8] Tomáš Kulhánek, J Šilar, Marek Mateják, Pavol Privitzer, Jiří Kofranek, and Martin Tribula. Distributed Computation and Parameter Estimation in Identification of Physiological Systems. *VPH conference*, 2010.

[9] Tomáš Kulhánek, Filip Ježek, Marek Mateják, Jan Šilar, Pavol Privitzer, Martin Tribula, and Jiří Kofránek. RESTful Web Service to Build Loosely Coupled Web Based Simulation of Human Physiology. *Transactions of Japanese Society for Medical and Biological Engineering*, 51(Supplement):R–32, 2013. [doi:10.11239/jsmbe.51.R-32](http://dx.doi.org/10.11239/jsmbe.51.R-32).

[10] J Kofránek, M Mateják, P Privitzer, M Tribula, T Kulhánek, J Šilar, and R Pecinovský. HumMod-Golem Edition: Large Scale Model of Integrative Physiology for Virtual Patient Simulators. In *Proceedings of World Congress in Computer Science 2013 (WORLDCOMP’13), International Conference on Modeling, Simulation and Visualisation Methods (MSV’13)*, pages 182–188, 2013.

[11] Marek Mateják, Tomáš Kulhánek, Jan Šilar, Pavol Privitzer, Filip Ježek, and Jiří Kofránek. Physiolibrary - Modelica Library for Physiology. In *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*, pages 499–505. Linköping University Electronic Press, 2014. [doi:10.3384/ecp14096499](http://dx.doi.org/10.3384/ecp14096499).

[12] Tomáš Kulhánek and Milan Šárek. Virtualizace a Integrace v Gridovém Pacs Systému (Cze) Virtualization and Integration in Grid Pacs System (eng). In *Mefanet 2008*. MSD Brno, 2008. URL: [http://www.mefanet.cz/res/file/mefanet2008/prispevky/25\\_kulhanek.pdf](http://www.mefanet.cz/res/file/mefanet2008/prispevky/25_kulhanek.pdf).

[13] Milan Šárek and Tomáš Kulhánek. Nové Směry Medicínských Aplikací Sdružení CESNET. In *Sborník Příspěvků MEDSOFT*, pages 145–148, 2009. URL: [http://www.creativeconnections.cz/medsoft/2009/Medsoft\\_2009\\_\%C5\%A0\%C3\%A1rek\\_Milan.pdf](http://www.creativeconnections.cz/medsoft/2009/Medsoft_2009_Šárek_Milan.pdf).

[14] Tomáš Kulhánek. Virtual Distributed Environment for Exchange of Medical Images. In D Kuželová, editor, *Proceeding of PhD. Conference 2009 was published as: Doktorandské Dny ’09*, pages 62–64. Institute of Computer Science/MatfyzPress, 2009. URL: <http://www.cs.cas.cz/hakl/doktorandsky-den/files/2009/sbornik-dd-2009.pdf>.

[15] Tomáš Kulhánek, Marek Frič, and Milan Šárek. Vzdálený Přístup k Virtuálním Výukovým a Výzkumným Aplikacím - Podpora Foniatrických Vyšetření. In Daniel Schwarz, Jaroslav Majerník, Ladislav Dušek, Stanislav Štípek, and Vladimír Mihál, editors, *Mefanet 2009*. MSD Brno, 2009. URL: [http://www.mefanet.cz/res/file/mefanet2009/prispevky/kulhanek\\_full.pdf](http://www.mefanet.cz/res/file/mefanet2009/prispevky/kulhanek_full.pdf).

[16] Tomáš Kulhánek, Marek Frič, and Milan Šárek. Vzdálená Analýza Lidského Hlasu - Bezeztrátové Nahrávání Zvuku Přes ip Sítě. In *Sborník Příspěvků MEDSOFT*, pages 96–101, 2010.

[17] Tomáš Kulhánek, Jan Šilar, Jiří Kofránek, Marek Mateják, Pavol Privitzer, and Martin Tribula. Od Výukového Modelu k Identifikaci Fyziologického Systému. In Daniel Schwarz, Martin Komenda, Jaroslav Majerník, Stanislav Štípek, Vladimír Mihál, and Ladislav Dušek, editors, *Mefanet 2010*. MSD Brno, 2010. URL: <http://www.mefanet.cz/res/file/mefanet2010/prispevky/kulhanek-full.pdf>.

[18] Tomáš Kulhánek. From Educational Models Towards Identification of Physiological Systems. *Mefanet Report*, 04:69–72, 2011. URL: <http://www.mefanet.cz/res/file/reporty/mefanet-report-2011.pdf>.

[19] Tomáš Kulhánek. Infrastructure for Data Storage and Computation in Biomedical Research. In D Kuželová, editor, *Proceeding of PhD. Conference 2011 was published as: Doktorandské dny ’11*, pages 90–94. Institute of Computer Science/MatfyzPress, 2011. URL: <http://www.cs.cas.cz/hakl/doktorandsky-den/files/2011/sbornik-dd-2011.pdf>.

[20] Tomáš Kulhánek, Marek Frič, and Jaroslav Hrb. Vzdálená Analýza Lidského Hlasu v Reálném Čase. In *Sborník Příspěvků MEDSOFT*, pages 180–184, 2012. URL: [http://www.creativeconnections.cz/medsoft/2012/Medsoft\\_Kulh\'{a}nek\\_Tom\'{a}\v{s}.pdf](http://www.creativeconnections.cz/medsoft/2012/Medsoft_Kulh'{a}nek_Tom'{a}v{s}.pdf).

[21] Tomáš Kulhánek, Marek Mateják, Jan Šilar, Pavol Privitzer, Martin Tribula, Filip Ježek, and Jiří Kofránek. HYBRIDNÍ ARCHITEKTURA PRO WEBOVÉ SIMULÁTORY. In *Sborník Příspěvků MEDSOFT*, pages 115–121, 2013. URL: [http://www.creativeconnections.cz/medsoft/2013/Medsoft\\_2013\\_Kulhanek.pdf](http://www.creativeconnections.cz/medsoft/2013/Medsoft_2013_Kulhanek.pdf).

[22] Tomáš Kulhánek, Marek Mateják, Jan Šilar, and Jiří Kofránek. IDENTIFIKACE FYZIOLOGICKÝCH SYSTÉMŮ. In *Sborník Příspěvků MEDSOFT*, pages 148–153, 2014. URL: [http://www.creativeconnections.cz/medsoft/2014/Medsoft\\_2014\\_Kulhanek.pdf](http://www.creativeconnections.cz/medsoft/2014/Medsoft_2014_Kulhanek.pdf).

[23] Tomáš Kulhánek and Marek Frič. Remote Analysis of Human Voice - Environment for Voice Training and ORL Medicine. In *EGI Technical Forum, Prague*, Prague, 2012.

[24] Jiří Kofránek, Pavol Privitzer, Martin Tribula, Jan Šilar, Tomáš Kulhánek, Marek Mateják, and Filip Ježek. Virtual Patient Simulator, CZ Utility Model. Application number: 2014-30329. Issue number: 27613, 2014. URL: <http://isdv.upv.cz/portal/pls/portal/portlets.pts.det?xprim=10082946>.

[25] Jiří Kofránek, Rudolf Pecinovský, and Petr Novák. Kopenograms – Graphical Language for Structured Algorithms. In *International Conference Foundations of Computer Science*, pages 90–96, 2012.

[26] Rudolf Bayer and E. M. McCreight. Organization and Maintenance of Large Ordered Indexes. *Acta Informatica*, 1(3):173–189, 1972. [doi:10.1007/BF00288683](http://dx.doi.org/10.1007/BF00288683).

[27] Rudolf Bayer. Symmetric binary B-Trees: Data Structure and Maintenance Algorithms. *Acta Informatica*, 1(4):290–306, 1972. [doi:10.1007/BF00289509](http://dx.doi.org/10.1007/BF00289509).

[28] Alan Cobham. The Intrinsic Computational Difficulty of Functions. In *Proc. 1964 Congress for Logic,Methodology and Philosophy of Science.*, pages 24–30, 1964.

[29] Stephen A. Cook. An Overview of Computational Complexity. *Communications of the ACM*, 26(6):400–408, June 1983. [doi:10.1145/358141.358144](http://dx.doi.org/10.1145/358141.358144).

[30] M R Garey and D S Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences). *Computers and Intractability*, page 340, 1979.

[31] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *STOC ’71 Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971. [doi:10.1145/800157.805047](http://dx.doi.org/10.1145/800157.805047).

[32] Richard M. Karp. *Reducibility Among Combinatorial Problems*. Springer US, 1972. [doi:10.1007/978-1-4684-2001-2\_9](http://dx.doi.org/10.1007/978-1-4684-2001-2_9).

[33] Richard E Korf. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27:97–109, 1985.

[34] J Pearl and R E Korf. Search Techniques. *Annual Review of Computer Science*, 2:451–467, 1987. [doi:10.1146/annurev.cs.02.060187.002315](http://dx.doi.org/10.1146/annurev.cs.02.060187.002315).

[35] Planck Collaboration, P. a. R. Ade, N. Aghanim, M. I. R. Alves, C. Armitage-Caplan, et al. Planck 2013 results. I. Overview of Products and Scientific Results. March 2013. [doi:10.1051/0004-6361/201321529](http://dx.doi.org/10.1051/0004-6361/201321529).

[36] CH Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1995.

[37] Michael Sipser. *Introduction to the Theory of Computation*. 2012.

[38] Amdahl. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *Proceedings of AFIPS Conference*, pages 483–485, 1967.

[39] John L. Gustafson. Reevaluating Amdahl’s law. *Communications of the ACM*, 31(5):532–533, May 1988. [doi:10.1145/42411.42415](http://dx.doi.org/10.1145/42411.42415).

[40] David R. Butenhof. *Programming with POSIX Threads*. 1997.

[41] B Chapman, G Jost, and Ruud Van Der Pas. *Using OpenMP*, volume 10. MIT press, 2008.

[42] Peter S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, 1997.

[43] Javier Diaz, Camelia Muñoz Caro, and Alfonso Niño. A Survey of Parallel Programming Models and Tools in the Multi and Many-core Era. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1369–1386, 2012. [doi:10.1109/TPDS.2011.308](http://dx.doi.org/10.1109/TPDS.2011.308).

[44] Cleve Moler. Matrix Computation on Distributed Memory Multiprocessors. *Hypercube Multiprocessors*, pages 181–195, 1986.

[45] Ian Foster. *Designing and Building Parallel Programs*. Addison-Wesley Publishing Company, 1995.

[46] Erick Cantu-Paz. *Designing Efficient and Accurate Parallel Genetic Algorithms*. PhD Thesis, University of Illinois at Urbana-Champaign, 1999.

[47] PH Madden. Dispelling the Myths of Parallel Computing. *IEEE Design & Test of Computers*, (April):58–64, 2012. [doi:10.1109/MDT.2012.2230391](http://dx.doi.org/10.1109/MDT.2012.2230391).

[48] David Culer, Jaswinder Pal Singh, and Anoop Gupta. *Parallel Computer Architecture*. Morgan Kaufmann, 1998.

[49] Thomas Rauber and Gudula Rünger. *Parallel Programming: For Multicore and Cluster Systems*. Springer, 2013.

[50] Andrew S Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms, 2/E*. 2007.

[51] Thomas Erl. *SOA Principles of Service Design*. Prentice Hall, 2008.

[52] Roy Thomas Fielding. Chapter 5: Representational State Transfer (REST). *Architectural Styles and the Design of Network-based Software Architectures, Dissertation*, 2000.

[53] Steve Vinoski. REST eye for the SOA guy. *IEEE Internet Computing*, 11:82–84, 2007. [doi:10.1109/MIC.2007.22](http://dx.doi.org/10.1109/MIC.2007.22).

[54] Martin Fowler. *Patterns of Enterprise Application Architecture*. Pearson Education, Inc., 2003.

[55] Jimmy Nilsson. *Applying Domain-driven Design and Patterns*. Pearson Education, Inc., 2006.

[56] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns*. Pearson Education, Inc., 2002.

[57] Václav Novák and Josef Vojtěch. Deployment of a DWDM System with CLA Optical Amplifiers in the CESNET2 Network. *Networking Studies*, page 27, 2007.

[58] I Foster and C Kesselman. *The Grid 2: Blueprint For a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[59] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[60] A Chervenak, I Foster, C Kesselman, C Salisbury, and S Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*, 23(3):187–200, 2000.

[61] Ian Foster, Carl Kesselman, Jeffrey M Nick, and Steven Tuecke. The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration. *Grid Computing: Making the Global Infrastructure a Reality*, pages 217–249, 2003.

[62] Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A Security Architecture for Computational Grids. In *Proceedings of the 5th ACM Conference on Computer and Communications Security - CCS ’98*, pages 83–92, New York, New York, USA, November 1998. ACM Press. [doi:10.1145/288090.288111](http://dx.doi.org/10.1145/288090.288111).

[63] I. Bird, B. Jones, and K.F. Kee. The Organization and Management of Grid Infrastructures. *Computer*, 42(1):36–46, January 2009. [doi:10.1109/MC.2009.28](http://dx.doi.org/10.1109/MC.2009.28).

[64] Dagmar Adamova. Computing Framework for the LHC: Current Status and Challenges of the High Luminosity LHC Future. In *52 International Winter Meeting on Nuclear Physics, Bormio,Italy*, number January, 2014. URL: [http://pos.sissa.it/archive/conferences/212/021/Bormio2014\\_021.pdf](http://pos.sissa.it/archive/conferences/212/021/Bormio2014_021.pdf).

[65] M. Litzkow and M. Livny. Experience with the Condor Distributed Batch System. *IEEE Workshop on Experimental Distributed Systems*, 1990. [doi:10.1109/EDS.1990.138057](http://dx.doi.org/10.1109/EDS.1990.138057).

[66] E Laure, S Fisher, A Frohner, C Grandi, P Kunszt, et al. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12:33–45, 2006.

[67] I. Foster and C. Kesselman. Globus: a Metacomputing Infrastructure Toolkit, 1997. [doi:10.1177/109434209701100205](http://dx.doi.org/10.1177/109434209701100205).

[68] M Riedel, E Laure, L Field, J P Navarro, J Casey, et al. Interoperation of World-wide Production e-Science Infrastructures. In *Concurrency and Computation: Practice and Experience 21.8*, number March, pages 961–990. 2009. [doi:10.1002/cpe](http://dx.doi.org/10.1002/cpe).

[69] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: An Experiment in Public-Resource Computing. *Communications of the ACM*, 45(11):56–61, November 2002. [doi:10.1145/581571.581573](http://dx.doi.org/10.1145/581571.581573).

[70] DP Anderson. Boinc: A System for Public-resource Computing and Storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*. IEEE, 2004. [doi:10.1109/GRID.2004.14](http://dx.doi.org/10.1109/GRID.2004.14).

[71] Zoltan Balaton, Gabor Gombas, Peter Kacsuk, Adam Kornafeld, Jozsef Kovacs, et al. SZTAKI Desktop Grid: A Modular and Scalable Way of Building Large Computing Grids. In *Parallel and Distributed Processing Symposium*. IEEE, 2007. [doi:10.1109/IPDPS.2007.370665](http://dx.doi.org/10.1109/IPDPS.2007.370665).

[72] Peter Kacsuk, Jozsef Kovacs, Zoltan Farkas, Attila Csaba Marosi, Gabor Gombas, and Zoltan Balaton. SZTAKI Desktop Grid (SZDG): A Flexible and Scalable Desktop Grid System. *Journal of Grid Computing*, 7(4):439–461, September 2009. [doi:10.1007/s10723-009-9139-y](http://dx.doi.org/10.1007/s10723-009-9139-y).

[73] Gilles Fedak, Franck Cappello, and Universitc Paris Sud. XtremWeb : A Generic Global Computing System. In *Proceedings of First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 582–587, 2001. [doi:10.1109/CCGRID.2001.923246](http://dx.doi.org/10.1109/CCGRID.2001.923246).

[74] Werner Herr, D I Kaltchev, F Schmidt, and E Mcintosh. Large Scale Beam-beam Simulations for the CERN LHC using Distributed Computing. Technical Report June 2006, 2006. URL: <http://cds.cern.ch/record/972345>.

[75] N Hø imyr, J Blomer, P Buncic, M Giovannozzi, A Gonzalez, et al. BOINC Service for Volunteer Cloud Computing. *Journal of Physics: Conference Series*, 396(3):032057, December 2012. [doi:10.1088/1742-6596/396/3/032057](http://dx.doi.org/10.1088/1742-6596/396/3/032057).

[76] P. Kacsuk, Z. Farkas, and G. Fedak. Towards Making BOINC and EGEE Interoperable. *2008 IEEE Fourth International Conference on eScience*, pages 478–484, December 2008. [doi:10.1109/eScience.2008.111](http://dx.doi.org/10.1109/eScience.2008.111).

[77] Etienne Urbah, Peter Kacsuk, Zoltan Farkas, Gilles Fedak, Gabor Kecskemeti, et al. EDGeS: Bridging EGEE to BOINC and XtremWeb. *Journal of Grid Computing*, 7(3):335–354, September 2009. [doi:10.1007/s10723-009-9137-0](http://dx.doi.org/10.1007/s10723-009-9137-0).

[78] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, et al. Xen and the Art of Virtualization. *ACM SIGOPS Operating Systems Review*, 37:164, 2003. [doi:10.1145/1165389.945462](http://dx.doi.org/10.1145/1165389.945462).

[79] Lamia Youseff, Rich Wolski, Brent Gorda, and Chandra Krintz. Paravirtualization for HPC Systems. *Frontiers of High Performance Computing and Networking â€“ ISPA 2006 Workshops*, pages 474–486, 2006. [doi:10.1007/11942634\_49](http://dx.doi.org/10.1007/11942634_49).

[80] Miroslav Ruda, Zdeněk Šustr, Jivrí Sitera, David Antoš, Lukáš Hejtmánek, Petr Holub, and Miloš Mulač. Virtual Clusters as a New Service of MetaCentrum, the Czech NGI. In *Cracow Grid Workshop’09*, pages 66–71, 2009.

[81] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud Computing and Grid Computing 360-degree Compared. *Grid Computing Environments*, 2008. [doi:10.1109/GCE.2008.4738445](http://dx.doi.org/10.1109/GCE.2008.4738445).

[82] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing. 2011. URL: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>.

[83] Ashiq Anjum, Richard Hill, Richard McClatchey, Nik Bessis, and Andrew Branson. Glueing Grids and Clouds Together: A Service-oriented Approach. *International Journal of Web and Grid Services*, 8(3):248–265, 2012. [doi:10.1504/IJWGS.2012.049169](http://dx.doi.org/10.1504/IJWGS.2012.049169).

[84] Dejan Milojicic, Ignacio M Llorente, and Ruben S Montero. OpenNebula: A Cloud Management Tool. *IEEE Internet Computing*, 15(2):11–14, 2011. [doi:10.1109/MIC.2011.44](http://dx.doi.org/10.1109/MIC.2011.44).

[85] R Kumar, N Gupta, S Charu, K Jain, and SK Jangir. Open Source Solution for Cloud Computing Platform Using OpenStack. 3(5):89–98, 2014. URL: <http://www.ijcsmc.com/docs/papers/May2014/V3I5201427.pdf>.

[86] Sixto Ortiz. The Problem with Cloud-Computing Standardization. *Computer*, 44(July):13–16, 2011. [doi:10.1109/MC.2011.220](http://dx.doi.org/10.1109/MC.2011.220).

[87] Georg Hager and Gerhard Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. 2010. [doi:10.1201/EBK1439811924](http://dx.doi.org/10.1201/EBK1439811924).

[88] John Levesque and Gene Wagenbreth. *High Performance Computing: Programming and Applications*. CRC Press, 2010.

[89] Ioan Raicu, Ian T. Foster, and Yong Zhaoyo. Many-task Computing for Grids and Supercomputers. In *2008 Workshop on Many-task Computing on Grids and Supercomputers, MTAGS 2008*, 2008. [doi:10.1109/MTAGS.2008.4777912](http://dx.doi.org/10.1109/MTAGS.2008.4777912).

[90] Ioan Raicu. *Many-task Computing: Bridging the Gap Between High-throughput Computing and High-performance Computing*. PhD thesis, 2009.

[91] Ioan Raicu, Ian Foster, Mike Wilde, Zhao Zhang, Kamil Iskra, et al. Middleware Support for Many-task Computing. *Cluster Computing*, 13:291–314, 2010. [doi:10.1007/s10586-010-0132-9](http://dx.doi.org/10.1007/s10586-010-0132-9).

[92] Alexandru Iosup, Simon Ostermann, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and D Epema. Performance Analysis of Cloud Computing Services for Many-tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945, 2010.

[93] James Pasley. How BPEL and SOA are Changing Web Services Development. *IEEE Internet Computing*, 9(June):60–67, 2005. [doi:10.1109/MIC.2005.56](http://dx.doi.org/10.1109/MIC.2005.56).

[94] Jia Yu and Rajkumar Buyya. A Taxonomy of Scientific Workflow Systems for Grid Computing. *ACM SIGMOD Record*, 34(3):44, 2005. [doi:10.1145/1084805.1084814](http://dx.doi.org/10.1145/1084805.1084814).

[95] Yong Zhao, Ioan Raicu, and Ian Foster. Scientific Workflow Systems for 21st Century e-Science, New Bottle or New Wine? *IEEE Congress on Services - Part I, 2008.*, pages 467 – 471, 2008. [arXiv:0808.3545](http://arxiv.org/abs/0808.3545).

[96] V. Curcin and M. Ghanem. Scientific Workflow Systems - Can One Size Fit All? *2008 Cairo International Biomedical Engineering Conference*, 2008. [doi:10.1109/CIBEC.2008.4786077](http://dx.doi.org/10.1109/CIBEC.2008.4786077).

[97] Peter Kacsuk, Tamas Kiss, and Gergely Sipos. Solving the Grid Interoperability Problem by P-GRADE Portal at Workflow Level. *Future Generation Computer Systems*, 24:744–751, 2008. [doi:10.1016/j.future.2008.02.008](http://dx.doi.org/10.1016/j.future.2008.02.008).

[98] Peter Kacsuk. P-GRADE Portal Family for Grid Infrastructures. *Concurrency Computation Practice and Experience*, 23:235–245, 2011. [doi:10.1002/cpe.1654](http://dx.doi.org/10.1002/cpe.1654).

[99] Kassian Plankensteiner, Radu Prodan, Matthias Janetschek, Thomas Fahringer, Johan Montagnat, et al. Fine-Grain Interoperability of Scientific Workflows in Distributed Computing Infrastructures. *Journal of Grid Computing*, 11:429–455, 2013. [doi:10.1007/s10723-013-9261-8](http://dx.doi.org/10.1007/s10723-013-9261-8).

[100] Nancy Wilkins-Diehr. Special Issue: Science Gateways—Common Community Interfaces to Grid Resources. *Concurrency and Computation: Practice and Experience*, 19(6):743–749, April 2007. URL: <http://doi.wiley.com/10.1002/cpe.1098>, [doi:10.1002/cpe.1098](http://dx.doi.org/10.1002/cpe.1098).

[101] Marlon Pierce, Suresh Marru, Borries Demeler, Raminderjeet Singh, and Gary Gorbet. The Apache Airavata Application Programming Interface: Overview and Evaluation with the UltraScan Science Gateway. In *2014 9th Gateway Computing Environments Workshop*, pages 25–29. IEEE, November 2014. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7021845>, [doi:10.1109/GCE.2014.15](http://dx.doi.org/10.1109/GCE.2014.15).

[102] Shahbaz Memon, Morris Riedel, Florian Janetzko, Borries Demeler, Gary Gorbet, et al. Advancements of the UltraScan scientific Gateway for Open Standards-based Cyberinfrastructures. *Concurrency and Computation: Practice and Experience*, 26(13):2280–2291, September 2014. [doi:10.1002/cpe.3251](http://dx.doi.org/10.1002/cpe.3251).

[103] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai, and Istvan Marton. WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities. *Journal of Grid Computing*, 10:601–630, 2012. [doi:10.1007/s10723-012-9240-5](http://dx.doi.org/10.1007/s10723-012-9240-5).

[104] Issac N. Bankman, editor. *Handbook of Medical Imaging; Processing and Analysis*. 2000.

[105] K. Slavicek, O. Dostal, M. Javornik, and M. Drdla. MEDIMED - Regional Centre for Medicine Image Data Processing. *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 310–313, January 2010. [doi:10.1109/WKDD.2010.133](http://dx.doi.org/10.1109/WKDD.2010.133).

[106] Peeter Ross, Ruth Sepper, and Hanna Pohjonen. Cross-border Teleradiology-Experience From Two International Teleradiology Projects. *European Journal of Radiology*, 73(1):20–5, January 2010. [Doi:10.1016/J.Ejrad.2009.10.016](http://dx.doi.org/10.1016/j.ejrad.2009.10.016).

[107] Vanessa Saliba, Helena Legido-Quigley, Riina Hallik, Ain Aaviksoo, Josip Car, and Martin McKee. Telemedicine Across Borders: A Systematic Review of Factors That Hinder or Support Implementation. *International Journal of Medical Informatics*, 81(12):793–809, December 2012. [doi:10.1016/j.ijmedinf.2012.08.003](http://dx.doi.org/10.1016/j.ijmedinf.2012.08.003).

[108] S. G. Erberich, M. Bhandekar, M. D. Nelson, a. Chervenak, and C. Kesselman. DICOM Grid Interface Service for Clinical and Research PACS: A Globus Toolkit Web Service for Medical Data Grids. *International Journal of Computer Assisted Radiology and Surgery*, 1(ii):100–102, 2006. [doi:10.1007/s11548-006-0013-0](http://dx.doi.org/10.1007/s11548-006-0013-0).

[109] Stephan G Erberich, Jonathan C Silverstein, Ann Chervenak, Robert Schuler, Marvin D Nelson, and Carl Kesselman. Globus MEDICUS - Federation of DICOM Medical Imaging Devices into Healthcare Grids. *Studies in Health Technology and Informatics*, 126:269–278, 2007.

[110] H Duque, J Montagnat, J Pierson, L Brunie, and I E Magnin. DM 2 : A Distributed Medical Data Manager for Grids. In *Biogrid’03 workshop*, pages 138–147, Tokyo, Japan.

[111] Johan Montagnat, Ákos Frohner, Daniel Jouvenot, Christophe Pera, Peter Kunszt, et al. A Secure Grid Medical Data Manager Interfaced to the gLite Middleware. *Journal of Grid Computing*, 6(1):45–59, July 2007. [doi:10.1007/s10723-007-9088-2](http://dx.doi.org/10.1007/s10723-007-9088-2).

[112] Dagmar Krefting, Julian Bart, Kamen Beronov, Olga Dzhimova, Jürgen Falkner, et al. MediGRID: Towards a User Friendly Secured Grid Infrastructure. *Future Generation Computer Systems*, 25(3):326–336, 2009. [doi:10.1016/j.future.2008.05.005](http://dx.doi.org/10.1016/j.future.2008.05.005).

[113] Dagmar Krefting, Michal Vossberg, Andreas Hoheisel, and Thomas Tolxdorff. Simplified Implementation of Medical Image Processing Algorithms into a Grid Using a Workflow Management System. *Future Generation Computer Systems*, 26(4):681–684, 2010. [doi:10.1016/j.future.2009.07.004](http://dx.doi.org/10.1016/j.future.2009.07.004).

[114] K Skaburskas, F Estrella, J Shade, D Manset, J Revillard, et al. Health-e-Child: A Grid Platform for European Paediatrics. *Journal of Physics: Conference Series*, 119:082011, 2008. [doi:10.1088/1742-6596/119/8/082011](http://dx.doi.org/10.1088/1742-6596/119/8/082011).

[115] Siegfried Benkner, Antonio Arbona, Guntram Berti, Alessandro Chiarini, Robert Dunlop, et al. @neurIST: Infrastructure for Advanced Disease Management Through Integration of Heterogeneous Data, Computing, and Complex Processing Services. *IEEE Transactions on Information Technology in Biomedicine*, 14(6):1365–1377, 2010. [doi:10.1109/TITB.2010.2049268](http://dx.doi.org/10.1109/TITB.2010.2049268).

[116] Martin Kuba, Ondrej Krajícek, Petr Lesný, Jan Vejvalka, and Tomás Holecek. Grid Empowered Sharing of Medical Expertise. *Studies in Health Technology and Informatics*, 120:273–282, 2006.

[117] Martin Kuba, Daniel Kouřil, and Michal Procházka. Sealed Grid with Downloadable Services. In Thierry Priol and Marco Vanneschi, editors, *Towards Next Generation Grids*, pages 83–91. Springer US, 2007. [doi:10.1007/978-0-387-72498-0\_8](http://dx.doi.org/10.1007/978-0-387-72498-0_8).

[118] I Foster. Globus Toolkit Version 4:Software for Service-oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006.

[119] William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link, Catalin Dumitrescu, Ioan Raicu, and Ian Foster. The Globus Striped GridFTP Framework and Server. *Proceedings of the ACM/IEEE 2005 Supercomputing Conference, SC’05*, 2005(c):1–11, 2005. [doi:10.1109/SC.2005.72](http://dx.doi.org/10.1109/SC.2005.72).

[120] JW Cooley and JW Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, pages 297–301, 1965. URL: <http://www.jstor.org/stable/2003354>.

[121] M. Frigo and S.G. Johnson. The Design and Implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1386650>, [doi:10.1109/JPROC.2004.840301](http://dx.doi.org/10.1109/JPROC.2004.840301).

[122] Wendy DeLeo LeBorgne and Barbara D. Weinrich. Phonetogram Changes For Trained Singers Over a Nine-month Period of Vocal Training. *Journal of Voice*, 16(1):37–43, 2002. [doi:10.1016/S0892-1997(02)00070-X](http://dx.doi.org/10.1016/S0892-1997(02)00070-X).

[123] Floris L Wuyts, Louis Heylen, Fons Mertens, M D Caju, Raoul Rooman, P H de Heyning, and Marc De Bodt. Effects of Age, Sex, and Disorder on Voice Range Profile Characteristics of 230 Children. *ANNALS OF OTOLOGY RHINOLOGY AND LARYNGOLOGY*, 112(6):540–548, 2003.

[124] Jan G. Švec and Harm K. Schutte. Videokymography: High-speed Line Scanning of Vocal Fold Vibration. *Journal of Voice*, 10(2):201–205, 1996. [doi:10.1016/S0892-1997(96)80047-6](http://dx.doi.org/10.1016/S0892-1997(96)80047-6).

[125] Jan G Svec, Frantisek Sram, and Harm K Schutte. Videokymography in Voice Disorders: What to Look For? *The Annals of Otology, Rhinology, and Laryngology*, 116(3):172–80, March 2007. URL: <http://europepmc.org/abstract/MED/17419520>.

[126] Marek Frič. Parametrizovaný Fonetogram Obecných Řečových a Hlasových Projevů - ParVRP, 2007. URL: <http://zvuk.hamu.cz/vyzkum/dokumenty/TL12x.pdf>.

[127] Marek Frič, Tomáš Kulhánek, and Jaroslav Hrb. Systém Pro Vzdáleně Přístupnou Analýzu Hlasu RealVoiceLab, 2012. URL: <http://zvuk.hamu.cz/vyzkum/dokumenty/TL46x.pdf>.

[128] P Kohl, E J Crampin, T A Quinn, and D Noble. Systems biology: An Approach. *Clinical Pharmacology and Therapeutics*, 88:25–33, 2010. [doi:10.1038/clpt.2010.92](http://dx.doi.org/10.1038/clpt.2010.92).

[129] P. J. Hunter and M. Viceconti. The VPH-Physiome Project: Standards and Tools for Multiscale Modeling in Clinical Applications. *IEEE Reviews in Biomedical Engineering*, 2:40–53, 2009. [doi:10.1109/RBME.2009.2036204](http://dx.doi.org/10.1109/RBME.2009.2036204).

[130] A C Guyton, T G Coleman, and H J Granger. Circulation: Overall Regulation. *Annual Review of Physiology*, 34:13–46, 1972. [doi:10.1146/annurev.ph.34.030172.000305](http://dx.doi.org/10.1146/annurev.ph.34.030172.000305).

[131] Robert L Hester, Radu Iliescu, Richard Summers, and Thomas G Coleman. Systems Biology and Integrative Physiological Modelling. *The Journal of Physiology*, 589(Pt 5):1053–60, March 2011. [doi:10.1113/jphysiol.2010.201558](http://dx.doi.org/10.1113/jphysiol.2010.201558).

[132] Robert L Hester, Alison J Brown, Leland Husband, Radu Iliescu, Drew Pruett, Richard Summers, and Thomas G Coleman. HumMod: A Modeling Environment For the Simulation of Integrative Human Physiology. *Frontiers in Physiology*, 2(April):12, January 2011. [doi:10.3389/fphys.2011.00012](http://dx.doi.org/10.3389/fphys.2011.00012).

[133] Erik Butterworth, Bartholomew E. Jardine, Gary M. Raymond, Maxwell L. Neal, and James B. Bassingthwaighte. JSim, An Open-source Modeling System For Data Analysis. *F1000Research*, 2, May 2014. [doi:10.12688/f1000research.2-288.v2](http://dx.doi.org/10.12688/f1000research.2-288.v2).

[134] P J Hunter. The IUPS Physiome Project: A Framework For Computational Physiology. *Progress in Biophysics and Molecular Biology*, 85(2-3):551–69, January 2004. [doi:10.1016/j.pbiomolbio.2004.02.006](http://dx.doi.org/10.1016/j.pbiomolbio.2004.02.006).

[135] Tommy Yu, Catherine M Lloyd, David P Nickerson, Michael T Cooling, Andrew K Miller, et al. The Physiome Model Repository 2. *Bioinformatics (Oxford, England)*, 27(5):743–744, 2011. [doi:10.1093/bioinformatics/btq723](http://dx.doi.org/10.1093/bioinformatics/btq723).

[136] Michael Hucka, ABBJ Finney, Benjamin J Bornstein, Sarah M Keating, Bruce E Shapiro, et al. Evolving a Lingua Franca and Associated Software Infrastructure For Computational Systems Biology: The Systems Biology Markup Language (SBML) Project. *Systems Biology*, 1(1):41–53, 2004.

[137] Nicolas Le Novère, Benjamin Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, et al. BioModels Database: A Free, Centralized Database of Curated, Published, Quantitative Kinetic Models of Biochemical and Cellular Systems. *Nucleic Acids Research*, 34:D689–D691, 2006. [doi:10.1093/nar/gkj092](http://dx.doi.org/10.1093/nar/gkj092).

[138] J Kofránek and J Rusz. Restoration of Guyton´s Diagram for Regulation of the Circulation As a Basis for Quantitative Physiological Model Development. *Physiological research / Academia Scientiarum Bohemoslovaca*, 59(6):897–908, January 2010.

[139] J Kofranek, M Matejak, and P Privitzer. Hummod-large Scale Physiological Models in Modelica. In *Proceedings 8th Modelica Conference, Dresden, Germany*, pages 713–724, 2011.

[140] Javier Fernandez de Canete, P del Saz-Orozco, D Moreno-Boza, and E Duran-Venegas. Object-oriented Modeling and Simulation of the Closed Loop Cardiovascular System by Using SIMSCAPE. *Computers in Biology and Medicine*, 43(4):323–33, May 2013. [doi:10.1016/j.compbiomed.2013.01.007](http://dx.doi.org/10.1016/j.compbiomed.2013.01.007).

[141] J Fernandez de Canete, J Luque, J Barbancho, and V Munoz. Modelling of Long-term and Short-term Mechanisms of Arterial Pressure Control in the Cardiovascular System: An Object-oriented Approach. *Computers in Biology and Medicine*, 47:104–112, April 2014. [doi:10.1016/j.compbiomed.2014.01.006](http://dx.doi.org/10.1016/j.compbiomed.2014.01.006).

[142] Peter Fritzson and Peter Bunus. Modelica-a General Object-oriented Language for Continuous and Discrete-event System Modeling and Simulation. In *Simulation Symposium, 2002. Proceedings. 35th Annual*, pages 365–380. IEEE, 2002.

[143] Michael Tiller. Modelica by Example, 2014. URL: [http://book.xogeny.com](http://book.xogeny.com/).

[144] M. Hofmann. On the Complexity of Parameter Calibration in Simulation Models. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2(4):217–226, October 2005. [doi:10.1177/154851290500200405](http://dx.doi.org/10.1177/154851290500200405).

[145] P Eykhoff. *Trends and Progress in System Identification*. IFAC Series for Graduates, Research Workers and Practising Engineers; vol. 1. Pergamon, Oxford, 1981.

[146] Michael C K Khoo. *Physiological Control Systems*. IEEE press, 2000.

[147] Carmen G Moles, Pedro Mendes, and Julio R Banga. Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods. *Genome Research*, pages 2467–2474, 2003. [doi:10.1101/gr.1262503.](http://dx.doi.org/10.1101/gr.1262503.)

[148] William A Pruett, Leland D Husband, Graham Husband, Muhammad Dakhlalla, Kyle Bellamy, Thomas G Coleman, and Robert L Hester. A Population Model of Integrative Cardiovascular Physiology. *PloS one*, 8(9):e74329, 2013.

[149] Shuji Takahashi, Kenta Sakawa, Yoichi Shiraishi, Hiroshi Miyashita, and Science Technology. Modeling, Simulation and Parameter Estimation of the Cardiovascular System by Using Model Based Approach. pages 493–500, 2013.

[150] Mohamed a. Abbass, Emad ElSamahy, and Ahmed Genedy. An Optimized Cardiovascular Model: A Pattern Search Approach. *Proceedings - 2012 International Conference on Biomedical Engineering and Biotechnology, iCBEB 2012*, 1:45–48, 2012. [doi:10.1109/iCBEB.2012.62](http://dx.doi.org/10.1109/iCBEB.2012.62).

[151] Sergio Maffioletti, Riccardo Murri, B Jonen, and S Scheuring. Computational Workflows with GC3Pie. In *Proceedings of the EGI Community Forum*, 2012.

[152] Marty Humphrey, Norm Beekwilder, Jonathan L. Goodall, and Mehmet B. Ercan. Calibration of Watershed Models Using Cloud Computing. *2012 IEEE 8th International Conference on E-Science*, (1048222):1–8, October 2012. [doi:10.1109/eScience.2012.6404420](http://dx.doi.org/10.1109/eScience.2012.6404420).

[153] Bryan a. Tolson and Christine a. Shoemaker. Dynamically Dimensioned Search Algorithm for Computationally Efficient Watershed Model Calibration. *Water Resources Research*, 43(August 2006):1–16, 2007. [doi:10.1029/2005WR004723](http://dx.doi.org/10.1029/2005WR004723).

[154] John H Holland. *Adaptation in Natural and Artificial Systems*, volume Ann Arbor. 1975. URL: <http://www.citeulike.org/group/664/article/400721>.

[155] T Blochwitz, M Otter, M Arnold, and C Bausch. The Functional Mockup Interface for Tool Independent Exchange of Simulation Models. In *Proceeding of 8th Modelica Conference, Dresden*, 2011.

[156] J. C. Helton, J. D. Johnson, C. J. Sallaberry, and C. B. Storlie. Survey of Sampling-based Methods for Uncertainty and Sensitivity Analysis. *Reliability Engineering and System Safety*, 91:1175–1209, 2006. [doi:10.1016/j.ress.2005.11.017](http://dx.doi.org/10.1016/j.ress.2005.11.017).

[157] Andrea Saltelli, Stefano Tarantola, Francesca Campolongo, and Marco Ratto. *Sensitivity Analysis in Practice*. 2004. [doi:10.1002/0470870958](http://dx.doi.org/10.1002/0470870958).

[158] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis. The Primer*. 2008. URL: <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470059974.html>, [doi:10.1002/9780470725184](http://dx.doi.org/10.1002/9780470725184).

[159] W van Meurs. *Modeling and Simulation in Biomedical Engineering: Applications in Cardiorespiratory Physiology*. McGraw-Hill Professional, 2011.

[160] Ann L. Chervenak, Theo G M Van Erp, Carl Kesselman, Mike D’Arcy, Janet Sobell, et al. A System Architecture for Sharing De-identified, Research-ready Brain Scans and Health Information Across Clinical Imaging Centers. *Studies in Health Technology and Informatics*, 175:19–28, 2012. [doi:10.3233/978-1-61499-054-3-19](http://dx.doi.org/10.3233/978-1-61499-054-3-19).

[161] Bernhard Gleich and Jürgen Weizenecker. Tomographic Imaging Using the Nonlinear Response of Magnetic Particles. *Nature*, 435(7046):1214–7, June 2005. URL: <http://www.nature.com.ezproxy.techlib.cz/nature/journal/v435/n7046/full/nature03808.html>, [doi:10.1038/nature03808](http://dx.doi.org/10.1038/nature03808).

[162] Emine U. Saritas, Patrick W. Goodwill, Laura R. Croft, Justin J. Konkle, Kuan Lu, Bo Zheng, and Steven M. Conolly. Magnetic Particle Imaging (MPI) for NMR and MRI Researchers. *Journal of Magnetic Resonance*, 229:116–126, 2013. URL: <http://dx.doi.org/10.1016/j.jmr.2012.11.029>, [doi:10.1016/j.jmr.2012.11.029](http://dx.doi.org/10.1016/j.jmr.2012.11.029).

[163] Peter Hunter, Tara Chapman, Peter V Coveney, Bernard de Bono, Vanessa Diaz, et al. A Vision and Strategy for the Virtual Physiological Human: 2012 Update. *Interface Focus*, 3(2), February 2013. URL: <http://rsfs.royalsocietypublishing.org/content/3/2/20130004.abstract>.

[164] Timothy H. Vines, Arianne Y K Albert, Rose L. Andrew, Florence Débarre, Dan G. Bock, et al. The Availability of Research Data Declines Rapidly with Article Age. *Current Biology*, 24:94–97, 2014. [arXiv:1312.5670](http://arxiv.org/abs/1312.5670), [doi:10.1016/j.cub.2013.11.014](http://dx.doi.org/10.1016/j.cub.2013.11.014).

[165] P. Bryan Heidorn. Shedding Light on the Dark Data in the Long Tail of Science. *Library Trends*, 57(2):280–299, 2008. [doi:10.1353/lib.0.0036](http://dx.doi.org/10.1353/lib.0.0036).

[166] Christof Weinhardt, Arun Anandasivam, Benjamin Blau, Nikolay Borissov, Thomas Meinl, Wibke Michalk, and Jochen Stößer. Cloud Computing – A Classification, Business Models, and Research Directions. *Business & Information Systems Engineering*, 1:391–399, 2009. [doi:10.1007/s12599-009-0071-2](http://dx.doi.org/10.1007/s12599-009-0071-2).

[167] Chris Anderson. The Long Tail: How Endless Choice Is Creating Unlimited Demand. *Word Journal of the International Linguistic Association*, page 256, 2006. URL: <http://www.amazon.co.uk/dp/184413850X>, [doi:30](http://dx.doi.org/30).

1. <http://unity3d.com/> accessed March 2015 [↑](#footnote-ref-2)
2. [http://www.kopenogram.org](http://www.kopenogram.org/) accessed March 2015 [↑](#footnote-ref-3)
3. In searching a problem, the sequential search algorithm is a brute-force approach that tries all of the values. There are better approaches, e.g., a binary search algorithm on a sorted list, taking logarithmic time complexity , which outperforms the sequential search. B-trees are the most used structure for holding the sorted list of elements in production application or databases [26, 27]. [↑](#footnote-ref-4)
4. For example, the depth-first iterative-deepening algorithm for a brute-force search was shown to be optimal, compared to other standard brute-force search algorithms (depth-first search or breadth-first search) [33, 34]. [↑](#footnote-ref-5)
5. <https://developer.nvidia.com/cuda-zone> accessed February 2015 [↑](#footnote-ref-6)
6. <https://www.khronos.org/opencl/> accessed February 2015 [↑](#footnote-ref-7)
7. <http://openmp.org/> accessed February 2015 [↑](#footnote-ref-8)
8. [http://www.metacentrum.cz](http://www.metacentrum.cz/) department of CESNET as a national grid infrastructure [↑](#footnote-ref-9)
9. [http://www.egi.eu](http://www.egi.eu/) [↑](#footnote-ref-10)
10. [http://glite.cern.ch](http://glite.cern.ch/) accessed February 2015 [↑](#footnote-ref-11)
11. <http://www.nordugrid.org/arc/> accessed February 2015 [↑](#footnote-ref-12)
12. <http://toolkit.globus.org/toolkit/> accessed February 2015 [↑](#footnote-ref-13)
13. <http://setiathome.ssl.berkeley.edu/> [↑](#footnote-ref-14)
14. <http://www.vmware.com/> accessed March 2015 [↑](#footnote-ref-15)
15. [↑](#footnote-ref-16)
16. [↑](#footnote-ref-17)
17. [↑](#footnote-ref-18)
18. [↑](#footnote-ref-19)
19. [↑](#footnote-ref-20)
20. [↑](#footnote-ref-21)
21. [↑](#footnote-ref-22)
22. [↑](#footnote-ref-23)
23. [↑](#footnote-ref-24)
24. [↑](#footnote-ref-25)
25. [↑](#footnote-ref-26)
26. [↑](#footnote-ref-27)
27. [↑](#footnote-ref-28)
28. [↑](#footnote-ref-29)
29. [↑](#footnote-ref-30)
30. [↑](#footnote-ref-31)
31. [↑](#footnote-ref-32)
32. [↑](#footnote-ref-33)
33. [↑](#footnote-ref-34)
34. [↑](#footnote-ref-35)
35. [↑](#footnote-ref-36)
36. [↑](#footnote-ref-37)
37. [↑](#footnote-ref-38)
38. [↑](#footnote-ref-39)
39. [↑](#footnote-ref-40)
40. [↑](#footnote-ref-41)
41. [↑](#footnote-ref-42)
42. [↑](#footnote-ref-43)
43. [↑](#footnote-ref-44)