3. Foundations of Grid Processing Architecture for the Comcute System

Piotr Brudło

Gdansk University of Technology,
Faculty of Electronics, Telecommunication and Informatics,
Computer System Architecture Department
e-mail: pebrd@eti.pg.gda.pl

Abstract

In the chapter, fundamental system algorithms and structures implemented in the Comcute system are described and analysed in detail. Layered architecture of the system model is highlighted. System tasks of the layers are elaborated, presented and described. Operational details of communication interfaces among layers are worked out and examined. The focus is put onto implemented system components with regard to their operability and efficiency. Scalability and system openness, as the key design factors of the implementation, are deliberatively taken into account. Important aspects of operability are addressed and the issues of validation, verification and deployment of the adopted system solutions are discussed. Practical application aspects of the Comcute system are described with respect to its final implementation and target installation in the form of grid processing in the open worldwide Internet.

Keywords*: Comcute, grid processing architecture*

3.1. Design issues

Several design versions of the Comcute system have been elaborated and successfully implemented. At the present stage, the system has been tested, verified and validated in its all fundamental functionalities. Scalability tests have been carried out within required application scope. In practical experiments, the Comcute has processed both system layered tasks and application computations according to design requirements. As applications, several instances have been run: breaking of DES codes with specified lengths, generation of great prime numbers of defined properties and research on text file similarities in compression processes. The obtained results of the experiments have allowed elaboration of general conclusions and experiences for deployment of the Comcute system in the form of grid computing into the Internet. The two processing paradigms have been considered: volunteer computing as well as obligatory computing [1], [2].

3.2. System architecture

The layered architecture has been elaborated for the Comcute system [3]. Layer *W* is responsible for task and data distribution, as well as for delivery of execution modules and data packages for processing. Layer *S* organises the processing, in layer *I* the processing is being generically carried out. Passing of results starts at layer *I* and goes up to layer *W* through layer *S*. This has been presented in conceptual scheme in Fig. 3.1. Moreover, above the layer *W*, there has been introduced high-level layer *Z*. Layer *Z* is to provide and serve as an entry interface to the Comcute system for clients who define and launch their applications.

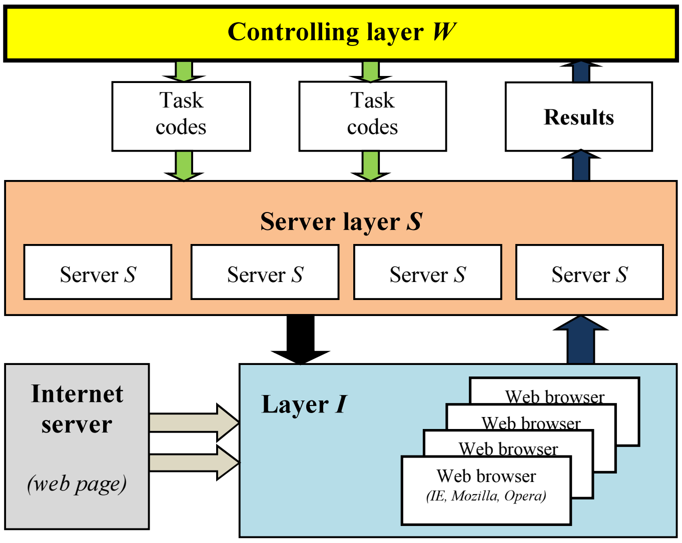

Fig. 3.1. General conceptual scheme for the Comcute system

The conceptual scheme (Fig. 3.1) illustrates operational implementation of layer *S* in cooperation with layers *W* and *I* in the process of execution of required computational sub-tasks. The sub-tasks are components of high-level applications delivered form layer *Z*. In practice, layer *S* receives complete program modules with attached data packages. Next, both program modules and data packages are being sent down to layer *I* [4].

3.2.1. Inter-layer cooperation

The conceptual scheme (Fig. 3.1) has been developed and elaborated with respect to realisation details. In Fig. 3.2, system architecture and complete operational flowchart are presented. Practical processing aspects have been deliberatively spotlighted. The architecture concentrates around four main conceptual components. They are: server *W*, servers *S*, Load Balancer and global user area (Internet), respectively.

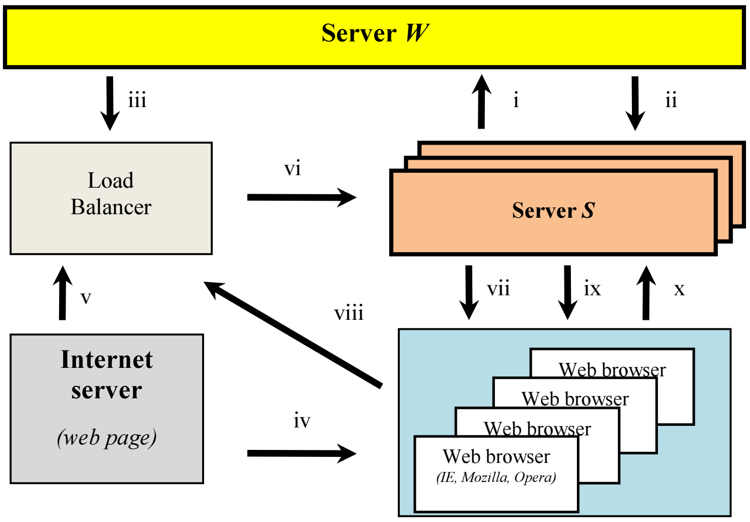

Fig. 3.2. Design of operational diagram of the Comcute system

Server *W* monitors operational activities and controls task execution at the higher layer. It distributes task modules to relevant servers *S*, and next, collects and combines the obtained results. Server *W* is the entry point for an application client. Servers *S* disseminate task modules to Internet users, control their executions and distribute data packages. Servers *S* verify correctness of the results obtained form Internet users. Consecutive component: Load Balancer connects Internet users to servers *S* having small temporary processing workload. In case an Internet user comes to the Comcute system, server *S* gets its own copy of a task module with relevant data packages.

According to the chart presented in Fig. 3.2, one may enumerate consecutive operational steps:

i. servers *S* are registered at server *W*
ii. server *W* distributes task modules and data packages
iii. server *W* determines task allocations according to Load Balancer data
iv. Internet user opens a web page
v. Internet user selects the link of Load Balancer
vi. Load Balancer attaches the Internet user to a selected server *S*
vii. web browser of the Internet user gets its task
viii. the task connects to the Load Balancer in order to get connection to server *S*
ix. the task gets its data package for processing
x. results are sent back from the Internet user to server *S* and to server *W*.

The proposed solution is sensible and clear, and has allowed achievement of specified operability and effectiveness. One may notice that the operational diagram incorporates key features of scalability and system transparency.

3.2.2. Applications

The Comcute system has been successfully implemented, tested and validated. In its current version, one may compute one's own processing tasks. As applications, several instances have been run: breaking of DES codes with specified lengths, generation of great prime numbers of defined properties and determination of text file similarities in compression processes.

3.2.3. Data packages distribution for tasks

Task distribution in the Comcute system is not limited by the system itself, and is merely defined by types (classes) of computation applications and their own inherent characteristics. A Comcute user should only deliver task code modules with relevant data packages and may control the distribution processes. The Comcute system does not confine the range of processing models, and allows running of almost any of currently used distributed network processing paradigms, e.g. cloud or grid computing.

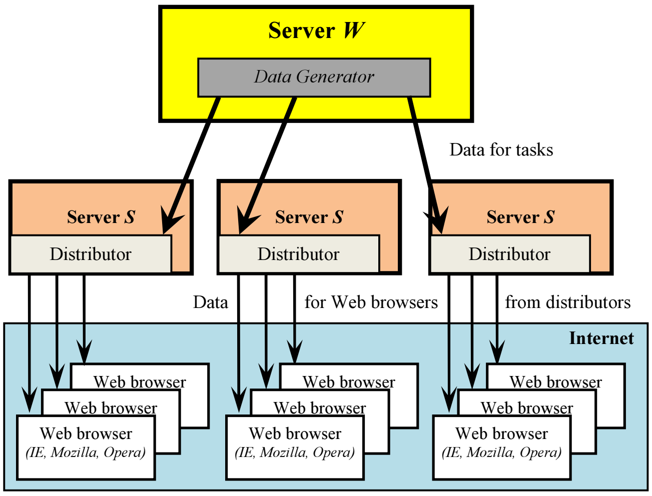

Fig. 3.3. Flowchart of task distribution model for the Comcute system

In the current version of the Comcute system, the two models of data distributions have been carefully tested and verified. In applications of great prime numbers generation and breaking DES codes, data packages are not correlated with one another, and so consecutive sub-executions can be processed separately and independently. Every data package can be practically processed by any random Internet user. Data distribution, is this case, is significantly simpler. Data is packaged into portions, and each package is assigned its unique identifier. The identifier is used in result collection processes, after completion of the tasks' executions. In case of text file content similarity determination, partial results have to be used in consecutive processing, and they should be stored temporarily in servers *S*. In general, in many cases, the data is connected inherently, which results in necessity of inner communication. The issue can be solved by use of functionalities of Distributor component in layer *S* (Fig. 3.3). In layer *S*, communication among servers is not considered, and there is no possibility of data exchange within the layer. Internet users should get relevant data packages, so correlated data should be placed onto the same server *S*. In server *W*, Data Generator distributes relevant data packages to selected servers *S*, solving the issues according to adopted optimisation procedures [5], [6].

3.3. Implementation specifications

In the implementation of the Comcute system, several classes of functionalities have been realised. Among others, one may enumerate the operational functionalities and software components like: user interface, inter-node communicator, layer *S* interface, workload monitor, task status monitor, task execution controller, repository for tasks, task distributor, results collector, repository of results, and many others. They are characterised below:

User interface
- reception of high-level applications for execution,
- controlling of tasks execution status,
- sending results to a user.

Inter-node communicator
- general communication with servers of layer *W,*
- communication with subsets of servers in layer *W,*
- sending messages to selected server sub-groups,
- reception of messages form selected servers of layer *W*.

Layer *S* interface
- reception of messages form selected servers of layer *S,*
- data sending to selected servers *S*,
- reception of results from servers of layer *S*.

Workload monitor
- determination of inner workload of a computing nodes,
- global determination of workload of servers in layers *W* and *S*,
- generation of ordered list of workloads of servers in a layer.

Task status monitor
- status control of tasks being in execution,
- reporting of task execution statuses.

Task execution controller
- task partitioning into execution module,
- distribution of modules with task codes,
- reception and analysis of partial results,
- verification of results of task execution,
- verification of task termination conditions,
- consolidation of partial results into complete ones.

Repository for tasks
- storing and delivering of execution codes for tasks,
- delivering and storing of data packages for tasks,
- collecting and storing partial results of tasks.

Task distributor

- deploying of tasks on servers in layer *S*,
- distribution of data packages for processing,
- reception of results form execution servers in layer *S*.

Results collector

- starting of execution code for results consolidation,
- storing of results in results repository.

Repository of results

- storing of complete final results,
- results presentation for Comcute users,
- verification of execution of the processed tasks.

In general, the specified functionalities and software components have been successfully tested and verified. The construction of the functionalities satisfies the system transparency and interoperability requirements and the software can be adapted to paradigms of grid or cloud computing in the Internet.

3.4. Conclusions

Architectural and conceptual foundations of the Comcute system have been elaborated and described. The Comcute system has been implemented, verified and validated. Several practical applications have been processed by the system. The results obtained for applications: DES codes breaking, great prime numbers generation and text file similarity determination by compression, have confirmed and proven the applicability of the global concept for practical realisation. At present, the system can be deployed in the Internet and can benefit from practically unlimited processing powers of personal computers in the global network for many various types of applications [7]. In general, one can notice that the structure and conceptual solutions of the Comcute system prefer massive processing of great number of small independent tasks, executed according to single instruction multiple data (SIMD) processing paradigm. However, also other types (classes) of distributed network processing, like grid or cloud computing, can be supported and realised in practice within acceptable desired level of efficiency and computing optimality.

References

1. Brudło P.: *Berkeley Open Infrastructure for Network Computing*, Chapter in monograph: „Distributed Calculations in Computer Systems of Grid Class Architecture", Publisher: Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdańsk, Poland, 2012, pp. 39-45.
2. Balicki J., Brudło P., Czarnul P., Kuchta J., Matuszek M., Szpryngier P., Szymański J.: *Functional Design of the Comcute System*, R&D technical report 33/2011, Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdansk, Poland, 2011.

3. Brudło P.: *Implementation Issues in the Comcute System*, Chapter in monograph: „Distributed Calculations in Computer Systems of Grid Class Architecture”, Publisher: Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdańsk, Poland, 2012, pp. 127-136.
4. Brudło P., Kuchta J., Szpryngier P., Szymański J.: *Requirements for Implementation of Selected Computational Methods for the Comcute System*, R&D technical report 36/2011, Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdansk, Poland, 2011.
5. Brudło P., Czarnul P., Kuchta J., Szpryngier P., Szymański J.: *Characteristics of Distributed Computations for the Comcute System,* R&D technical report 41/2011, Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdansk, Poland, 2011.
6. Balicki J., Bieliński T., Brudło P., Paluszak J., Szymański J: *Dissemination of Computations with Distribution Servers*, Chapter in monograph: „Distributed Calculations in Computer Systems of Grid Class Architecture”, Publisher: Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdańsk, Poland, 2012, pp. 113-126.
7. Brudło P.: *Security in Monitoring*, Chapter in monograph: „Distributed Calculations in Computer Systems of Grid Class Architecture”, Publisher: Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdańsk, Poland, 2012, pp. 175-184.