International Conference
FROM DELC TO VELSPACE
Plovdiv, 26–28 March 2014

ARCHITECTURE FOR A GRID-RESOURCE MANAGEMENT AND SCHEDULING SYSTEM

Dilyana Totseva

Abstract. *In this study a generalized framework for real-time calculations roots of equations, on grid architecture is demonstrated. These calculations are done using different algorithms for factorization. The framework takes parts in the BOINC project – comprehensive, multi-resource grid computing system. Using this grid network, the calculations are distributed as separate processes and are submitted to computational resources without prior knowledge of the underlying job submission and queuing systems. Some of the algorithms presented in the previous articles take part in this framework which helps to reduce the calculation time and improve the accuracy of the results.*

Keywords: information systems, software architecture, grid, grid resource management

1. GRID RESOURCE MANAGEMENT AND SCHEDULING SYSTEM

In the research related to heavy calculation, there are a lot of barriers, which can be faced and can be resolved by participating into projects based on grid calculations. Finding resources can be a really expensive process and would limit the results which should be accomplished. Also, some of the expensive accessible resources need a lot of time to be approved.

This can be solved by building a grid computing based framework for real-time calculations of roots of equations which use the resources of already established open infrastructure of network resources. The grid resource management and scheduling system is based on the grid developing concept [1] – in distributed computing, different computers within the same network share resources. Using such network gives the abilities for establishing parallel calculations, [2] optimization of the time consumed for calculations, using free resources all over the world, automating the job calculations. Using the grid computing resources, the calculations are characterized with low latency and high-bandwidth interconnects.

One of the strongest advantages to participate in a grid-based network is the user’s ability to have access to constant supply of shared resources. [3]

2. SYSTEM ARCHITECTURE

2.1. OVERVIEW

After analyzing various platforms for grid computing, it was decided that the BOINC project be used as a distributed computing platform. By taking part in that project, since it is a widely used platform, we increase the chance of volunteers contributing to take part in the researches' calculations.

The presented architecture follows the Client-Server model supported by BOINC. The client server model is one of the most common application models of network computing and is a common type of distributed systems design architecture. On the server side we have the BOINC project, hosted on the BOINC server. The BOINC project's server supports large number of volunteers. Once the BOINC client is installed on the user's machine, it becomes responsible for the communication with various project servers. The BOINC client manages the applications, computes the results and then reports them back. Through the client side, the user is able to review the result of the current calculation and assign new tasks.

2.2. MODULES

Iterative algorithms for polynomial factorization based on known methods for simultaneous finding of all simple zeroes, part of previous research, have been included in the established project. On the client side, these algorithms for factorization have been represented as a separate application's business modules. These components include:

- Algorithm based on Dvorchuk's method for polynomial factorization into quadratic factors;
- Parallel algorithm for polynomial factorization proposed in recent years for finding roots of quadratic polynomial;
- Parallel algorithm for polynomial factorization established after researches for improvement the efficiency and the calculation of the first proposed,
- Weierstrass – Dochev factorization method;
- Tanabe method.

2.3. DISTRIBUTED ARCHITECTURE SYSTEM LAYERS

2.3.1. WEB INTERFACE

The first building block provides a client user interface for job submission and monitoring through which the users can automatically identify current calculation results and the computed factors of the polynomial. The application has a set of web pages, installed on a web server where all the session information is saved.

The user can access each page remotely with any browser. Mobile application interface will be supported also.

With the user interface we can follow the status of the projects in which we participate. We are able to push new jobs which are then processed through the business layer. Once the calculations are completed we are able to follow the received results.

2.3.2. BUSINESS LAYER AND DATA PROCESSING BLOCK

In this part of the system the calculation requests are processed. The framework provides an instrument for automated processing of the requests. In the current system we are concentrated on the push model: work is sent from some submitting node to some computational resource, which then accepts and processes the job, returning the results to the submitter. After the calculations are complete, the related information is stored on the data source. The grid based framework provides an abstraction layer that allows jobs to be submitted to computational resources without prior knowledge of the underlying job submission and queuing systems.

This framework has been designed to consider the possibility for new functionalities to be added in the future, including new implemented modules.
Additional authorization layer provides authorization capabilities through which we can restrict the users accessing the system and submitting the calculation requests.

2.3.3. BOINC SERVER

BOINC (Berkeley Open Infrastructure for Network Computing) is a software system which provides the ability easily to establish scientific research while creating and operating with public-resource computing projects. The projects depend on communication requirements and are established with large storage - the project supports that requirements. University institutions, private companies and private PC owners can participate in multiple BOINC projects, sharing calculation resources and can control how this resources to be allocated. [4]

BOINC is oriented to single user and project. Once the project is set up, the platform automatically matches work, which to be processed with hosts suitable to execute it, taking into account estimated memory and disk requirements as well as architecture and operating system constraints. [5] [6]

Once the job is submitted, BOINC clients (i.e., server, desktop PCs) contact a server that acts as a central repository of work to retrieve jobs to be executed (pull).

2.3.4. DATABASE LAYER

Database layer contains three separate nodes: User Database which describes users, including their email address, name, web password, and authenticator; BOINC

Database stores information about the platforms, users, hosts, the work units and the received results; Application Database stores information relevant to the application as the algorithms included, the submitted jobs, the receive results.

2.3.5. DATA SERVER

Grid based software considers a specific job, finds suitable resources on the network and distributes the tasks. At a later stage, the task is monitored through the system - the current progress, the completed calculations, returned errors or failing tasks, if any. If needed, specific tasks can be rescheduled. Once the job is completed, all the results are collected by the data server.

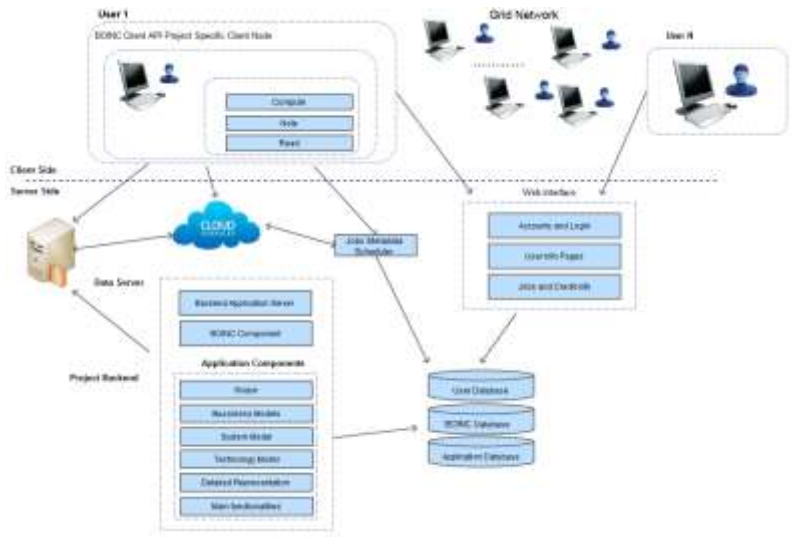

Figure 1. Architecture Blocks

3. APPLICATION COMPONENTS

3.1. CONTEXT Manager

With the context manager is possible to maintain the metadata information for the Application Components. Using the Context Manager could be resolved metadata about the application – application class name, application methods and their parameter information. At a later stage, the information is processed by the Jobs Invoker and Analyzer to construct the tasks and their dependencies.

4.1. APPLICATION PROGRAMMING MODEL

Iterative algorithms for polynomial factorization have been implemented as separate modules. Each of these algorithms is constructed as separate task which is called by the Job Scheduler. The framework is designed to consider the possibility for new implemented modules to be added as part of the grid calculations. Each method for polynomial factorization is presented as separate task and is called from the application. Using this approach is a least impact on the original grids' programming architecture.

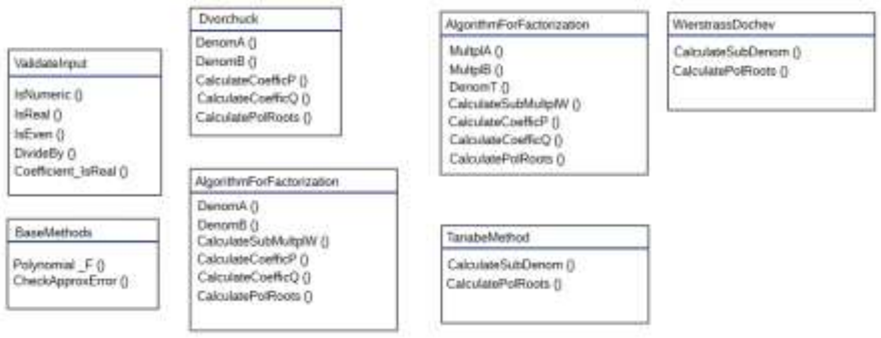

Figure 2. Programming Model

4.2. JOBS INVOKER AND JOB ANALYZER

When specific method invocation occurs within the application, the Job Analyzer method is triggered. It validates the method input data and verifies the method dependencies including the order of their execution. Part of the algorithms have been split in independent methods, it is Job Analyzer's task to validate that the current method state and that it is ready to be invoked. In the cases when we have a job with no dependencies, this task is in state 'ready for invocation.'

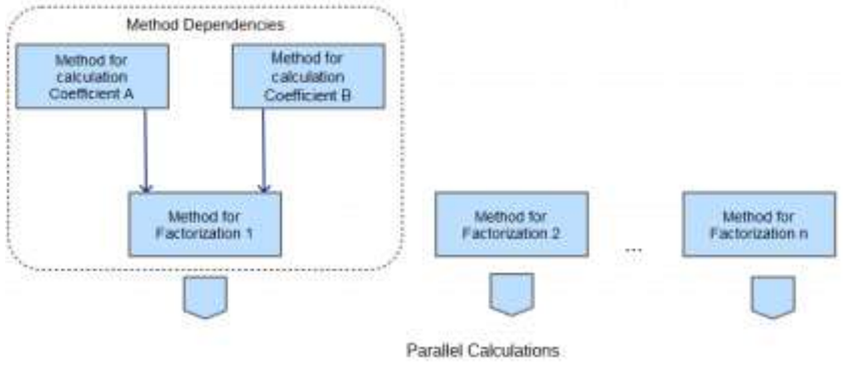
Method Dependencies
Method for calculation Coefficient A
Method for calculation Coefficient B
Method for Factorization 1
Method for Factorization 2
...
Method for Factorization n
Parallel Calculations

Figure 3. Job Analyzer

Following the diagram, we are able to run calculations of independent algorithms for polynomial factorization in parallel on the grid. For successfully completed calculations of the proposed algorithm, first its dependent calculations for Coefficients A and Coefficients B should be completed. Upon the successful completion of coefficients calculations, the tasks are sent for parallel execution on the grid. [7]

Once the application triggers a certain method call, this dynamically generates the job. Using two simple API methods and an additional flag in the database, we track which job is invoked of the framework for calculations. We leave the actual invocation of the method to Job Metadata Scheduler and Management.

4.3. JOBS METADATA SCHEDULER AND MANAGEMENT

Using the Job metadata scheduler, we are able to set up the calculation process, executed on a predefined period. First is selected the job, which we want to execute and the time period when this to be performed. Once the process is set up, on each job submitted through that layer we provide the input files to BOINC server set up. Using the scheduler, the server on predefined periods we check the current job status and the current results accomplished. Once the job is accomplished, we send the results back to the application and the user is able to observe and analyze them. [8]

The infrastructure for scheduling and sending jobs for calculation on the BOINC Client does not make any changes to the base software structure - these sides of the system are independent.

4. RESULTS VALIDATION

The calculations processed on our distributed system are algorithms for factorization and while the calculations are performed by grid resources, this could raise challenging concerns during execution. Scheduling algorithms for factorization is an important task for which several factors should be taken into consideration. One of those factors is the validation of the results.

In rare cases the results of the jobs could be with invalid data. This could be caused in the cases when project volunteers have consistent or sporadic hardware problems, typically causing errors in floating-point computation. Also some of them may maliciously return wrong results. [9]

For solving these specific issues for the established application are supplied two server-side methods:

```
ValidateParis_handler(RESULT& new_result, RESULT& canonical_result, bool& retry);
```

The handler is called by BOINC code and compares new received results to the valid one. In case of errors, it sets the new result's validate_state to either VALIDATE_STATE_INVALID or VALIDATE_STATE_VALID. If it has a recoverable error while reading an output file of either result, it returns retry=true, which causes the ValidatePairs_handler to arrange the workunit to be examined again in a few hours.

```
Assimilate_handler(WORKUNIT& wu, vector<RESULT> &results, RESULT& canonical_result)
```

The handler is called by BOINC code and it checks that all of the results are received. Then it creates a list of the results, reads the referenced result file and inserts the result and its signals into the Application Database.

5. CONCLUSION

There are many factors behind the continued interest in grid computing – the increasing availability of networked resources – PCs, workstations, servers combined with the increasing bandwidth on networks already reached into the gigabit range. On the other side, we also have evolution of key standards such as TCP/IP and Ethernet in networking. The established architecture and the presented design of the software system, using BOINC network, obtain resources, giving the power of expensive supercomputers, which otherwise would have been extremely expensive. Participating in the Berkeley Open Infrastructure for Network Computing, the architecture is focused on large-scale resource sharing in distributed systems in a flexible, secure and dynamic coordinated sharing way.
We have focused on a grid resource management and scheduling system based on resource collaboration models, combined with the ability to simply integrate business modules and dynamic applications. The used approach allows easy setup of scientific projects in which any volunteer users can take part and contribute.

ACKNOWLEDGMENTS

Research was partially supported by Fund Scientific Research MU13FMI002, Paisii Hilendarski University of Plovdiv.

REFERENCES

[1] Joseph, J., M. Ernest and C. Fellenstein, Evolution of grid computing architecture and grid adoption models, *IBM Systems journal*, Vol. 43, no 4, 2004.

[2] Pan, Y., Ch. Wu and W. Huang, A Grid Resource Broker with Dynamic Loading Prediction Scheduling Algorithm in Grid Computing Environment, *Proc. of the International Conference on Grid Computing Applications (GCA)*, 2008.

[3] Pan, Y., Ch. Wu, Ch. Liu, Hs. Yu and W. Huang, The Lightweight Approach to Use Grid Services with Grid Widgets on Grid WebOS, *CCGRID 2010*, 575–576.

[4] Anderson, D., BOINC: *A System for Public-Resource Computing and Storage*, University of California at Berkeley, ISBN: 0-7695-2256-4, 4–10.

[5] Mnaouer, C., Ragoonath, An Adaptive Priority Tuning System for Optimized Local CPU Scheduling using BOINC Clients, *Journal of Physics: Conference Series*, Vol. 256, ISBN: 978-1-61782-246-9, 200–216.
[6] Visegradi, J., Kovács and P. Kacsuk, Efficient extension of gLite VOs with BOINC based desktop grids, *Future Generation Comp. Syst*. 32, 2014, 13–23.
[7] Balaton, Z., G. Gombás, P. Kacsuk, A. Kornafeld, J. Kovács, Cs. Marosi, G. Vida, N. Podhorszki and T. Kiss, SZTAKI Desktop Grid: a Modular and Scalable Way of Building Large Computing Grids, *IPDPS*, 2007, 1–8.
[8] Narravula, S., A. Marnidala, A. Vishnu, K. Vaidyanathan and D. Panda, High performance distributed lock management services using network-based remote atomic operations In Cluster Computing and the Grid, 2007. CC-GRID 2007, Seventh IEEE International Symposium on, 2007, 583–590.
[9] Costa, F., L. Veiga and P. Ferreira, Internet-scale support for map-reduce processing, *J. Internet Services and Applications*, Vol. 4, (1), 2013, 1–17.

Faculty of Mathematics and Informatics
Paisii Hilendarski University of Plovdiv
236, Bulgaria Blvd.,
4003 Plovdiv, Bulgaria
e-mail: dilyana.totseva@gmail.com

АРХИТЕКТУРА НА РАЗПРЕДЕЛЕНА СИСТЕМА ЗА УПРАВЛЕНИЕ НА ПАРАЛЕЛНИ ИЗЧИСЛЕНИЯ

Диляна Тоцева

Резюме. *В тази статия е представена архитектура на система за управление на паралелни изчисления, използваща разпределени ресурси. Системата е включена към проекта BOINC – споделена, мулти-ресурна система за мрежови изчисления. Използвайки мрежата на BOINC, изчисленията са разпределени като паралелни независими процеси. Част от алгоритми за факторизация, представени в предишни проучвания и статии, са включени в представената система. Паралелните калкулации, извършени чрез разпределени ресурси, се използват за пресмятане корените на уравнения в реално време. С този подход е постигнато оптимизиране на времето необходимо за изчисления и подобряване точността на резултатите.*