Journal of Computing and Information Technology - CIT 15, 2007, 4, 347–353
doi:10.2498/cit.1001143

Optimization of Data Transfer for Grid Using GridFTP

Branimir Radic[1], Vedran Kajic[2] and Emir Imamagic[1]

[1]University Computing Centre, University of Zagreb, Croatia
[2]Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

Grid is a highly distributed heterogeneous environment which interoperates towards completing a common goal. As such, grids performance is highly dependant on the speed of data transfer between systems of which grid consists. Most commonly used and a "de facto" standard for data transfer on grid systems is GridFTP [7].

While working on CRO-GRID Infrastructure [3] project, we faced problems with small data transfer speeds when transferring small files. In this paper we present a program we have developed which uses GridFTP and speeds up data transfer of large number of small or medium sized files. We also present empirical results of the performance measurements achieved while using the implemented system.

Keywords: grid, data transfer, GridFTP

1. Introduction

Grid is used to solve problems that demand more resources than a common system can provide. Grid can consist of different systems and can be used for numerous common purposes. This article will focus on grids which are made of a number of computer nodes and are used for conducting calculations that require high amount of computer resources.

In grid, user applications submitted to a grid system for execution are called jobs. There are two common types of jobs that are submitted to a grid computer system for execution: parallel and serial jobs.

For their execution serial jobs require only one node and therefore require little, if any, data transfer. During the execution there is no need for data transfer. Exceptions are interactive serial jobs – serial jobs that require user interaction during execution. These jobs are, by their nature, incompatible with the nature of grid and are, as a rule, not run on a grid. Serial jobs use grid merely as a source of nodes for executing and not as a highly distributed heterogeneous environment that it is. Therefore, jobs are not dependant in their execution to data transfer speed of the grid data management system.

Parallel jobs use multiple nodes for execution. Upon submission of a job, a job management system (JMS), which is a part of the grid that makes decision where the job will be processed, schedules the details of job execution: where can the job be processed, how many nodes will be used etc. Finally, when the resources needed for job execution are available, a job execution begins. At that moment, data transfer mechanisms of grid start conducting the necessary data transfers.

Data necessary for job execution must first be transferred to all the nodes where job execution takes place. During the job execution data must be transferred between nodes and, finally, when the job is completed, the results must be gathered and sent back to the user. Each of these phases of job execution requires that the data transfer be fast, reliable, fault tolerant and properly authorized.

One common need of the grid system is data replication which is needed both by the executing jobs and the grid system itself. For its work grid system requires a constant data flow.

In Section 2, grid data transfer requirements are described and in Section 3 we describe GridFTP, a "*de facto*" standard in data transfer. Section 4 discuses theoretical and Section 5 practical solutions to some shortcomings of currently used data transfer protocols. Finally, we give the test results and the conclusion in Sections 6 and 7. At the end, we propose future work.

2. Grid Data Transfer Requirements

Data transfer in grid is, as described earlier, a key part of any grid system. Functionalities of a transfer system, which are essential for the functioning of a grid, are support and interoperation with grid authorization and authentication mechanisms, third party-initialized and controlled data transfer, parallel data transfer and finally support for reliable and restartable data transfer. We shall describe all those functionalities with more detail in the next part of this article.

During job execution, one of the issues is where to execute the job, based on the rights and privileges of the job submitter. During the entire job execution all the data transfer that takes place is authorized by the user that has submitted the job and, therefore, data transfer mechanism must interoperate with the authorization and authentication mechanisms.

Job management system needs to be capable of migrating certain segments of the job in order to improve system performance and/or optimize resource utilization. Job migration is controlled by the JMS and needs to transfer data from one site to another in a way that JMS defines. The type of data transfer where command for transferring data does not come from the location that is the source or the destination of data transfer, is called third party initialized data transfer. Third party initialized data transfer is often used for other purposes such as data replication etc.

During execution of parallel jobs there is commonly a need to transfer data form one site to different sites. This is most efficiently done by transferring data through multiple parallel data streams. Also, if a job consists of multiple subtasks that work in parallel, those tasks must be capable of communicating their results to other nodes in parallel.

Grid is a heterogeneous environment and as such is far more subjective to various system failures than more homogenous systems. Data transfer is no exception. Grid systems span over large geofigureical areas and are connected with different speed networks between nodes. Data transfer can fail at any part of that network system or certain nodes where data is stored might be disconnected from grid. Therefore, grid must have a mechanism of seamless recovery. The performance of the grid greatly depends on the speed of data transfer. For that reason, grid must utilize network in the best possible way.

De facto standard in data transfer today is GridFTP. GridFTP has become a *de facto* standard because it fulfils all the demands named earlier and a number of functionalities not mentioned here, but useful in some grid systems. However, as there are numerous different types of grids and even more different application types, GridFTP can still be improved in order to benefit from some applications even better.

3. GridFTP

The GridFTP protocol is developed as a part of Globus Toolkit. Latest major distribution of Globus Toolkit is GT 4.0 [6] and with the release of that distribution came also a new version of GridFTP.

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The protocol was built based on File Transport Protocol (FTP) [5] and added a few additional features to meet requirements of data grid projects.

Features of GridFTP include support for striping, public key infrastructure (PKI) [9] based Grid Security Infrastructure (GSI) [8] support, third party transfers, partial file access, reliability/restart, large file support, data channel reuse, parallel transfers and other.

Obviously, GridFTP inherits the properties of FTP and extends the functionalities for high speed transfer of large files. In numerous tests GridFTP has proven as the best protocol for high speed transfer of large data files. However, whilst GridFTP inherits advantages of FTP protocol it also inherits the poor performance that FTP has when transferring multiple small files. To be honest, all data transfer protocols and programs perform poorly in case of multiple small files. Even the numerous protocols for peer to peer (P2P) data transfer, which are developing to satisfy demand for file sharing, work well only with large files.

There is a number of ways to improve the file transfer performance when transporting small files. The choice how to improve data transfer can only be made when we wager the pros and cons of each approach.

4. Improving Small File Data Transfer Speed – A Theory

In this parafigure we will describe possible solutions for improving data transfer speed and explain the reason why we think that the method we used is the best method for grid systems. First we need to explain the reasons why the data transfer of small files is such a problem for transfer protocols.

In general, every data transfer has a few basic steps: authentication, authorization and data transfer. During the first step both sides in the communication need to authenticate themselves. This means that both the source and the destination need to verify who they are in some reliable way. After that, based on the known information, a decision can be made weather the user who initiated the transfer has all the authorization (rights, privileges etc.) to perform the desired transfer. Finally, the data transfer can be started, conducted and finally completed.

In the worst case the entire process is repeated for every file transferred. Transport protocols (including GridFTP) have a mechanism for channel reuse. The channel that was opened when the first file was transferred is reused for every next file and both sides in the communication reuse the authorization and authentication data sent during the first transfer.

Reason for slow transport of small files is located in the transport layer and not in the session layer of the OSI model. Every time a file is transferred, details of the data transfer must be negotiated before the transfer initialization is complete. The message exchange that takes place before every transfer slows down the transfer of multiple files as the duration of that procedure is mostly dependant on the message turnaround time. Without modification of the protocol itself, speeding up the transfer of small files can be done only by reducing the impact this procedure has on the overall transfer execution.

Transfer of large files is often accelerated by increasing block size of the data blocks used for transfer. All the data transfer is done by sending packages. If size of the packages increases up to the maximal size where errors, and the need to resend data are negligible, then speed of data transfer while transferring large data files can be greatly increased. The same method cannot be used for transfer of small files, as the blocks will simply be sent only partially filled with data, and the data transfer speed will slow down rather than speed up. On the other hand, by reducing the size of blocks, speed of transferring small files will most likely stay the same.

As mentioned earlier, using data block size to force small files will not work. However, files could be compressed on the source side and transferred as one large file to be uncompressed at the destination. The speed of data transfer will greatly increase because, instead of waiting for the turnaround time of all small files, the procedure will have to wait only once. In practice, this theoretically simple and efficient solution has several drawbacks. First drawback is that data compression puts high strain on the processor. This puts high load on the processor of the source during the compression and on the processor of the destination during the decompression. For larger sets of data the system would come to a halt for long periods of time. Large data sets also cause another problem, the problem of disk space. For this method to be used, extra disk space is required, as besides large quantity of data that make the files that are transferred, compressed data must also be stored. This disk space is required both on the source and the destination for the duration of the transfer procedure.

During the part of the transfer procedure where the channel is waiting for the response, no actual data transfer is taking place. This “idle” time is, as explained earlier, equal or greater than the turnaround time. What happens is that with small files idle time share increases as the size of the file decreases and the distance between the sites increases. The time while the data transport is idle must be decreased. To achieve that, we decided to use multiple data streams.

5. Grid Transfer System for Small Files

The system we developed is an extension made in order to use GridFTP in a way that increase data transfer speed for small files. It was developed for fulfilling the needs which appeared on CRO-GRID [2].

CRO-GRID is a national initiative which consists of three technological (science and industry) projects: CRO-GRID Mediator [4], CRO-GRID Applications [1] and CRO-GRID Infrastructure. CRO-GRID Mediator is developing service-oriented, and WSRF [12] compliant grid middleware. They are developing new concepts and implementing them in an environment. CRO-GRID Applications is developing scientific applications which will be used on the grid for producing valuable results. Our work is part of the CRO-GRID Infrastructure project which is implementing a Croatian grid for science. The main goal of our project is to build a grid which will satisfy all the demands and challenges that other two CRO-GRID projects and the Croatian scientific community have placed.

As already explained, projects that are part of CRO-GRID Application use grid maintained by the CRO-GRID Infrastructure project. Those projects transfer different types of data during the execution, and some of them need to transfer a large number of small files.

GridFTP protocol has a built-in support for parallel data transfer. This built-in capability is used to successfully increase transfer speed of large files and is used during striped transfer. Our solution creates a separate stream for each file that is transferred. By creating multiple data streams the idle time mentioned earlier is reduced. Multiple files are transferred at the same time and while one transfer is waiting for the response, other transfers take place. The communication required for starting transfer places almost no strain on the system. Therefore, doing several data transfers in parallel does not interfere with the performance of the system. Our system manages GridFTP when multiple files are transferred. Number of data streams that are used can be changed. Decision on how much data streams to use depends on a number of factors. The higher number of streams, the higher strain is placed on the network. Load during the data transfer is placed primarily on the network. During data transfer the data streams interfere and slow down each other transfer. However, this decrease in data transfer speed is smaller than the speed decrease caused by the protocol message exchange when the files transferred are small enough.

Data transfer speed decrease caused by protocol message exchange increases as the file size decreases. On the other hand, speed decrease caused by multiple streams increases as the number of streams increases. If no other parameters influenced data transfer speed then it would be possible to find ideal number of streams for data transfer. However, the files that are transferred are never all of the same size. There are also numerous other parameters that must be taken into consideration, such as network speed, distance and dynamical parameters.

With this said, it is obvious that you first need to know what you are transferring and then where you are transferring it for creating an ideal solution. However, as our measurements proved, the benefit of using multiple streams is far greater than the detriment when transferring small files.

6. Performance Measurements

As some of the projects using CRO-GRID resources require transferring large number of small sized files, we tested our system on CRO-GRID.

The nodes we selected for testing are three kilometers distant. This means that the benefit would increase when more distant nodes are used. Network speed between those nodes is 1Gb/s, as is the case between all the nodes of CRO-GRID. The goal of our work is to make

TEST No.	File Size (kB)	Number of files
1	1	10 000
2	10	1 000
3	100	100
4	1	3340
	10	336
	100	33

Table 1. File number and the file size of the tests.

the transfer speed at least close to the speed manageable by the network. We tested the network with four types of transfers, spanning from transferring only small files to the transfer where size of the largest file was greater than combined sizes of all small files transferred.

File size is given in Table 1. This way, all tests transferred the same amount of data. The measurements were repeated ten times for each test and for each number of data streams. The results did not greatly change between different trails. Results of the measurements for each test are displayed in Figure 1.

The figure clearly shows that transfer time greatly decreases with the increase of the number of streams. It is also shown that somewhere around 20 parallel streams there is no further increase in speed in any of the cases. The measurement

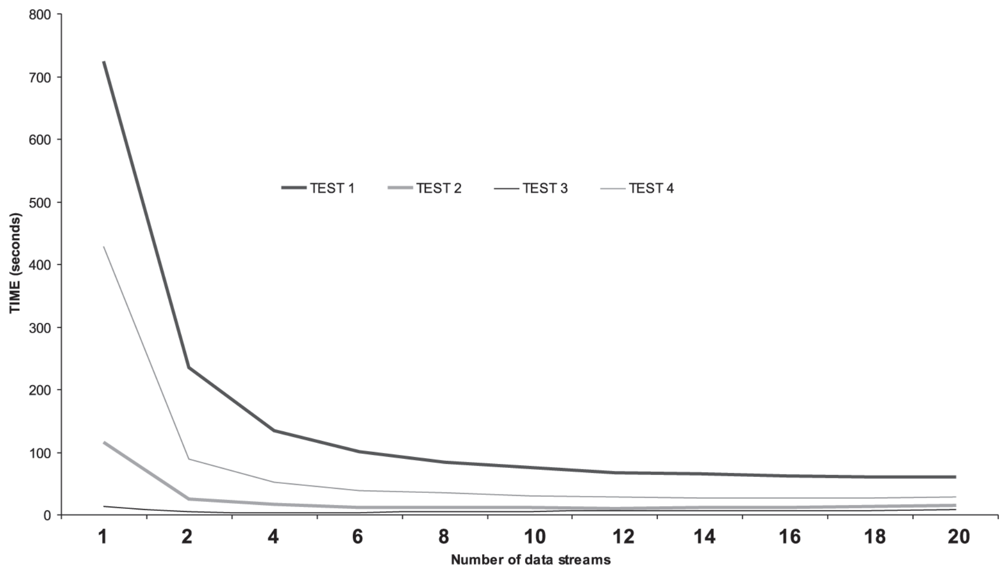

Figure 1. Transfer time in dependence on the number of streams.

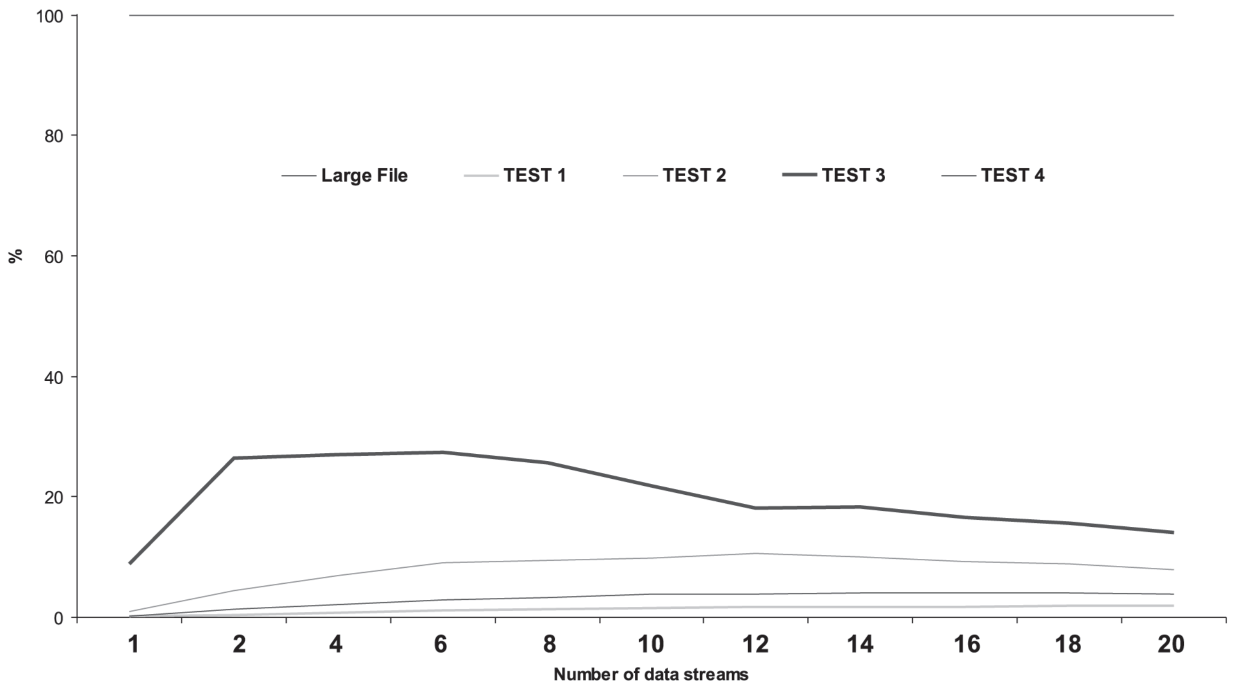

Figure 2. Data transfer compared to the speed of a large file transfer.

could be different if the sites were more distant and/or connected by a different speed network. We have already explained that the least benefit of this approach will have connected sites that are close by a fast connection. We can assume that this represents the worst case scenario in the sense how much acceleration can be achieved by creating multiple parallel streams. At the end of the data measurements we transferred 2GB file to see the actual potential that the data transfer can achieve. Data transfer of that file lasted 225.75 seconds, making the average data transfer speed 8.859 MB/s. The relative speed of data transfer achieved in these experiments can be seen in Figure 2. In Figure 2 we assumed that 100% is the speed that was achieved when transferring a 2 GB file.

As seen in Figure 1, the greatest acceleration is achieved when transferring large number of small files. It should be pointed out how ineffective the normal data transfer is in the case of such files. In Figure 2 we can see the average data transfer speed during all these attempts as compared to transferring a large file. Data speed is shown in the percentage how much of the potential data channel speed it uses presuming that large file transfer uses the entire potential. Clearly, in the case of small files efficiency of default GridFTP configuration is far too small to ignore.

One might think that comparing the efficiency of our system only to GridFTP is not enough to make conclusion about systems efficiency. This would be true if the efficiency of the transfer was not caused by the underlying layers of the data transfer mechanisms. We took a quick test of other data transfer mechanisms, RootD [11] file transfer daemon of the Root analysis environment and RFIO file transfer program [10].

7. Conclusion

The tests have proven that our system can greatly increase data transfer speed. The case where the system works best is the case of very small files. Our test results have also proven the theory that with parallel data streams full channel potential cannot be achieved. In our system, by using the appropriate number of parallel streams, the data transfer speed can be greatly increased.

With this solution, projects of CRO-GRID that create large quantities of small files will be capable of functioning faster. However, the full potential of the data channel is still not realized. This solution might not be the best possible one, but, by using it, all the applications suffering from slow transfer due to transferring multiple small files can be improved. The system can either create multiple streams on demand, or be made to always create multiple streams. This feature can be adjusted to application needs.

8. Future Work

Our system can improve the data transport of special cases, but what it does not have is a built in logic. In a practical case where data file size greatly varies, the system can increase performance greatly, or not at all, depending on the number of data streams used. We intend to implement logic that will make the best possible decision based on transfer parameters and then create the number of data streams that is best for that particular transfer.

Also we plan to carry out more measurements to determine how parameters not included in these tests influence data transfer speed. Those measurements should greatly increase our understanding of the system behaviour and help us build and implement a more efficient logic.

9. Acknowledgments

All given results were achieved through the work on the TEST program – Technological research-development projects with the support of the Ministry of science, education and sports and are a product of the cooperation between University Computing Centre and the Faculty of Electrical Engineering and Computing.

References

[1] CRO-GRID APPLICATIONS, `http://www.cro-grid.hr/hr/apps/` [02/28/2007].

[2] CRO-GRID HOMEPAGE, `http://www.cro-grid.hr` [02/28/2007].

[3] CRO-GRID INFRASTRUCTURE PROJECT, http://www.srce.hr/crogrid/infrastructure/ [02/28/2007].

[4] CRO-GRID MEDIATOR, http://www.cro-grid.hr/hr/mw/ [02/28/2007].

[5] FTP, http://www.scit.wlv.ac.uk/rfc/rfc9xx/RFC959.html [02/28/2007].

[6] GLOBUS ALLIANCE, http://www.globus.org/ [02/28/2007].

[7] GRIDFTP, http://www.globus.org/datagrid/gridftp.html [02/28/2007].

[8] GSI, http://www-unix.globus.org/toolkit/docs/3.2/gsi/index.html [02/28/2007].

[9] PKI – PAGE, http://www.pki-page.org/ [02/28/2007].

[10] RFIO, http://hikwww2.fzk.de/hik/orga/ges/infiniband/rfioib.html [02/28/2007].

[11] ROOTD, http://root.cern.ch/ [02/28/2007].

[12] WSRF, http://www-106.ibm.com/developerworks/library/ws-resource/ [02/28/2007].

Received: June, 2007
Accepted: September, 2007

Contact addresses:
Branimir Radic
Emir Imamagic
University Computing Centre
University of Zagreb
J. Marohnića 5
10000 Zagreb
Croatia
e-mail: branimir.radic@srce.hr
vedran.kajic@fer.hr
emir.imamagic@srce.hr

BRANIMIR RADIĆ graduated from the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, in July 2004. His research interests are high performance computing, distributed computing, computer clusters and grid systems.

Before graduation, he worked on the CRistal project at CERN, Switzerland in the summer of 2003 and on the MidArc middleware project at Ericsson Nicola Tesla in the summer of 2002. He is currently working as a system administrator, researcher on the CRO-GRID Infrastructure project and researcher on the EGEE-II project at the University Computing Centre.

EMIR IMAMAGIĆ graduated from the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, in May 2004. His research interests are high performance computing, distributed computing, computer clusters and grid systems.

Before graduation, he worked on the AliEn Grid project at CERN, Switzerland in the summer of 2003 and on the MidArc middleware project at Ericsson Nicola Tesla in the summer of 2002. He is currently working as a researcher on the CRO-GRID Infrastructure project and the EGEE-II project at the University Computing Centre.

VEDRAN KAJIĆ graduated from the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, in July 2007. His research interests include grid computing, computer vision and pattern recognition.

He is currently enrolled as a PhD student at Cardiff University, School of Optometry and Vision Sciences in Professor Wolfgang Drexler's group. He is involved in post processing of the data acquired by Optical Coherence Tomofigurey (OCT) imaging technique, developing automatic retina layer segmentation system and, eventually, automatic disease detection.