Massive data transfer between supercomputers and grid clusters using SCP controlled by FTS

P Tallada\textsuperscript{1,2}, G Bernabeu\textsuperscript{1,3}, M Caubet\textsuperscript{1,3}, M Delfino\textsuperscript{1,4}, F Martínez\textsuperscript{1,3}, C Neissner\textsuperscript{1,2}, M Rodríguez\textsuperscript{1,3} and N Tonello\textsuperscript{1,2}

\textsuperscript{1} Port d’Informació Científica (PIC), Universitat Autònoma de Barcelona, Edifici D, ES-08193 Bellaterra (Barcelona), Spain
\textsuperscript{2} Also at Institut de Física d’Altes Energies, IFAE, Edifici Cn, Universitat Autònoma de Barcelona, ES-08193 Bellaterra (Barcelona), Spain
\textsuperscript{3} Also at Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
\textsuperscript{4} Also at Universitat Autònoma de Barcelona, Departament de Física, ES-08193 Bellaterra (Barcelona), Spain

Abstract. Data transfers between Supercomputing centers and Grid sites may be complicated since, in general, Supercomputing centers are not completely incorporated in the Grid environment. In particular, the Port d’Informació Científica (PIC) had to transfer data of cosmological simulations from the Barcelona Supercomputing Center (BSC) to Grid storage resources in order to post-process those data at PIC. The data connection is limited to the Secure Shell (SSH) protocol using the scp command. Besides a limit of bandwidth, we had to deal with a lack of automation and reliability of the transfers. PIC managed to control those scp transfers through gLite’s File Transfer Service (FTS). Therefore, the discs containing the data at the BSC are mounted via SSH to a local Storage Resource Management (SRM) server at PIC. Once established the access to the data through the SRM server, the data can be transferred to any other SRM server, including the storage system at PIC, using grid protocols. Although we cannot improve the bandwidth, we reach a level of automation and reliability comparable to transfers from the Tier-0 to Tier-1 sites of the Large Hadron Collider (LHC) experiments.

1. Motivation

The Port d’Informació Científica (PIC)\cite{1} is a center of excellence for scientific data processing and storage supporting scientific groups working in projects which require strong computing resources for the analysis of massive sets of distributed data.

Supercomputers offer a great amount of processing power but their strict security policies make data transfers very difficult. Most projects are then left with limited mechanisms to transfer their data in and out. For instance, the MARENostrum Institut de Ciències de l’Espai (MICE)\cite{2} project uses MARENostrum supercomputer at Barcelona Supercomputing Center (BSC)\cite{3} to carry out simulations of the large-scale structure of the universe. The related datasets are several terabytes in size and they have to be transferred to PIC storage facilities for post-processing and long-term storage.

Taking the MICE project as a reference point, the purpose of this work is to analyze all the choices that the supercomputing centers make available for their data transfers and implement a streamlined transfer procedure.
2. Original procedure

In order to transfer datasets to PIC, the strict security policies put in place by BSC have to be respected. The datasets are only accessible from a set of frontend nodes, where no additional software can be installed. Besides, network connectivity is strongly limited, as no external connections are allowed, neither incoming nor outgoing. The only exception to this is the availability of interactive Secure Shell (SSH) access[4]. Based on those restrictions, previous to this work the transfer of each dataset was made in three phases and required the intervention of two persons at least, as different privileges were needed in each phase.

In the first phase, the dataset was transferred using \textit{scp} to a Network File System (NFS) [5] share at PIC. This first step is unreliable as \textit{scp} transfers do not support resuming and in the event of a network glitch the transfer will abort and will need to be restarted from the beginning. Note also that \textit{scp} works by transferring data over the SSH channel, which is encrypted in nearly all implementations[6]. The process of encryption is cpu intensive and may limit the effective bandwidth of the transfer. See the Testing and Performance section for a comparison of the encryption algorithms available and how they affect the effective bandwidth.

In the second phase, the integrity of the transferred data had to be checked. Although both the SSH protocol and its underlying transport, TCP/IP, have mechanisms to detect data corruption and intentional data tampering [7], the MICE project deemed necessary to carry out an additional test to ensure the integrity of the datasets and detect aborted transfers. Accordingly, a semi-automated process computed the Message-Digest algorithm 5 (MD5) [8] checksum of the files, both on the BSC and on the NFS at PIC, and then compared the values. Should any mismatch be detected, the file would have to be transferred again.

The third and final phase consisted in the transfer of the dataset from the NFS share to Grid[9] storage. The transfer was done using the \textit{srmcp} command, which uses a standard protocol called Storage Resource Management (SRM)[10] intended for interfacing with Grid storage services. This protocol ensures the integrity of the data by computing the ADLER-32[11] checksum of the file as it is being read and comparing it with the value returned by the storage media once it has been stored. That checksum is stored in a catalog to enable integrity assurance at the time the files are read again.

3. Objectives

The procedure described above is too cumbersome and prone to errors, as there are multiple possible points for failure. Based on the original procedure, we have defined a set of main objectives for improvement, each of them mapped to a set of actions or behaviors that should be supported in order to achieve it.

The first objective is to optimize the use of resources. On one hand, in the old procedure each dataset must be on local storage at PIC, such as an NFS share, before it can be transferred to Grid Storage. This intermediate copy on NFS doubles the amount of time and disk space requirements for the transfer. On the other hand, the use of two different checksum algorithms also approximately doubles the time required for integrity testing, as the value computed in the first check cannot be reused in the second.

The second objective is to improve transfer management and monitoring. On one hand, the bandwidth in Grid sites is a scarce resource and it is usually used concurrently by multiple users. The new procedure should take that into account and ensure that every user gets a share of it, while at the same time trying to boost channel utilization. On the other hand, users should be able to monitor their transfers including the status, speed and time remaining to completion, and be notified of relevant state changes.

The third objective is to increase transfer reliability. Currently, as the procedure is mostly manual, any incident during the transfer requires a human action to resume it, and that hinders efficiency. To prevent that, there has to be some mechanism to detect transfer failures, and
initiate corrective actions to try to recover from that situation, such as a limited number of automated retries.

The next section covers all the changes implemented in the transfer procedure in order to achieved the objectives proposed here.

4. Implementation
The most important change in the procedure, and the one that enables us to achieve the first objective, is the suppression of the intermediate copy and its associated checksum operation. The intermediate copy was needed because the SRM protocol used for transfers in the Grid environment only allows transferring between either local or grid endpoints[12]. As our files are remote, they have to be copied first to a local storage at PIC before they can be transferred to Grid storage. If we could access those files as if they were local, we would be able to spare that copy.

There are several software solutions that implement this specific behavior. The one we have used is called SSH File System (SSHFS)[13] and it is based on the Filesystem in Userspace (FUSE)[14] library of the Linux kernel. With this software, a remote SSH directory can appear and function as a local filesystem, with Portable Operating System Interface (POSIX)[15] behaviour. Therefore, all I/O operations over that filesystem are intercepted and routed as SSH File Transfer Protocol (SFTP)[16] actions over the corresponding SSH connection.

The second objective is to improve the management and monitoring of the transfers. Most large data movements at PIC come from the Large Hadron Collider (LHC) [17] experiments and are handled by a software called File Transfer Service (FTS) [18]. This software queues up the transfers, distributes the available bandwidth in logical channels, assigned to either users or endpoints, and monitors all its states in real time. However, for BSC transfers to use the FTS service, both transfer endpoints must use the SRM protocol, usually provided by an SRM server.

Depending on the method used to access the actual data, several SRM server implementations are available. As the SSHFS system provides us with a local POSIX view of the remote SSH data, we are restricted to a SRM server that exports data from a local filesystem, and we chose BeStMan[19]. Using this software allows us to achieve the second objective, using FTS to manage and monitor the transfers.

The third and last objective is to enhance the transfer reliability. The are two main components involved in those transfers that we need to take care of in order to ensure a certain level of reliability. On one hand, the SSHFS system transmits data over an SSH connection that, depending on the I/O activity and the SSH server configuration, may be closed during an inactivity period. As SSHFS system relies on that connection remaining available, the SSH client must be configured to periodically send empty packets to keep the connection opened. Also, SSHFS has been configured to try to reestablish the connection, as long as there is I/O activity. On the other hand the FTS service has lots of configuration parameters that affect the transfers' reliability[20]. The most important are the number of retries and the waiting interval between them. The choice of a particular set of parameters helps us to achieve this last objective. An adequate setting should retry enough to spare the user from the burden of temporal failures, and only notify it when the failure requires manual intervention.

5. Testing and Performance
Once all the modifications have been implemented, it is necessary to test the resulting procedure to ascertain that the proposed objectives have been achieved, as well as compare its performance and reliability with the older procedure.

The first component to be tested is the SSH connection, as both procedures use it as a means to transfer the data. This connection is encrypted in a cpu intensive process that ultimately becomes the bottleneck. Before an SSH server and one of its clients establish the connection,
they negotiate for the encryption algorithm to use for that session. Therefore, selecting a fast encryption algorithm is the best way to boost transfer efficiency. Figure 1 shows the average throughput of the different encryption algorithm families.

![Figure 1. Cipher throughput](image)

For this test, several encryption algorithm families were tested using local in-memory copy operations to exclude any network or disk I/O interference. The results, presented in Figure 1, show that the ARCFOUR cipher family [21] can deliver up to 55% more performance than the default algorithm (AES) [22] with the same cpu power. On the other hand, using Triple-DES [23] can reduce up to 60% the efficiency of a transfer.

The second test tries to measure the overhead of the different software layers that abstract the data access, in order to ensure that they do not hinder the efficiency of the transfer. The methodology of this test is similar to the previous one, measuring the throughput of an in-memory copy operation using all the software layers, adding one at a time. In this test, the measures of the throughput using different software layers are all compatible within the measurement uncertainties, and with a mean value of 47 MB/s. Hence, the overhead associated with the additional software layers is negligible.

In this test, and bearing in mind the inherent precision shortcomings of the measures, we were unable to ascertain any significant difference in the throughput of the transfers when different numbers of software layers were used. In all cases, the throughput kept constant about 47 MB/s.

The third and final test involved testing the reliability and fault-tolerance of the transfers, as well as the resuming capabilities and other supported features. First, the scp command does neither support resuming nor reconnection retries so, in case of network failure, the transfer has to be restarted from the beginning. Second, SSHFS supports automatic reconnection [24] in case of network failure, but on extended network dropouts, the transfer resuming must be done manually. Finally, the FTS system takes advantage of the underlying layers’ features and supports automatic resuming and transfer parallelization. Table 1 summarizes the results of this test.

In the end, the tests come to demonstrate that the proposed solution is better in every aspect than the original procedure or, at least, that it performs equally well. The next section presents the concluding remarks of this work, as well as some pending tasks that seek to extend and enhance it.

6. Conclusions and future work
The enhanced transfer procedure that results from this work has been in production since May 2010, and dozens of terabytes have successfully been transferred between the BSC frontend nodes and the PIC storage facilities.
Table 1. Reliability and fault-tolerance test

|                  | On network failure | Reconnection       | Resuming             | Parallelization        |
|------------------|--------------------|--------------------|----------------------|------------------------|
| SCP              | Transfer is        | Not supported      | Not supported        | Supported but manual   |
|                  | cancelled          |                    |                      |                        |
| SSHFS            | Transfer is put in | Supported and      | Supported but        | Supported but          |
|                  | I/O wait           | configurable       | manual               | manual                 |
| FTS              | Not involved       | Not involved       | Supported and         | Supported and          |
|                  |                    |                    | configurable         | configurable           |

Nevertheless, there are still some tasks that may be undertaken in order to extend the functionality or the potential users of this work. First, all the infrastructure used could be shared with other projects with the same or similar requirements. If those projects have remote data which can be mounted and operated locally then they are perfect candidates to reuse the infrastructure. And secondly, the SRM server (BeStMan) is capable of responding to interactive requests from users who want to copy a reduced number of files. This way they do not have to resort to a full FTS transfer job. But as BeStMan uses a local filesystem as its storage backend it has no way of reading the stored checksum for the files requested, so those interactive operations have no integrity assurance. However, BeStMan provides some hooks [25] into its interface that can be used to read, calculate or even store the checksums in a custom location, enabling the integrity checking.

In the end, the overall response from the users has been very positive, because it has come with two main advantages. On one hand, the simplified procedure has made their job easier, as it integrates perfectly with the existing services and infrastructures. On the other hand, the failure rate has dropped significantly, not only due to automated retries, but also to a reduction in human errors.

Acknowledgments
The Port d’Informació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autònoma de Barcelona. This work was supported in part by the EU 7th Framework Programme INFRA-2007-1.2.3: e-Science Grid infrastructures Grant Agreement Number 222667, Enabling Grids for e-Science (EGEE) project.

References
[1] Port d’Informació Científica (PIC)
http://www.pic.es
[2] Marenostrum Institut de Ciències de l’Espai (MICE)
http://segre.ice.cat/fosalba/MICE
[3] Barcelona SuperComputing Center (BSC)
http://bsc.es
[4] MareNostrum User’s Guide, 2009
http://www.bsc.es/media/859.pdf
[5] S. Shepler (Storspeed, Inc.), M. Eisler, D. Novcek (NetApp), January 2010. Network File System (NFS) Version 4 Minor Version 1 Protocol
http://www.rfc-editor.org/rfc/rfc5661.txt
[6] OpenSSH client cannot disable encryption with "-c none"
http://bugzilla.mindrot.org/show_bug.cgi?id=877
[7] R. Braden, October 1989. Requirements for Internet Hosts – Communication Layers  
http://www.rfc-editor.org/rfc/rfc1122.txt

[8] R. Rivest (MIT Laboratory for Computer Science and RSA Data Security, Inc.), April 1992. The MD5 Message-Digest Algorithm  
http://www.rfc-editor.org/rfc/rfc1321.txt

[9] The Worldwide LHC Computing Grid  
http://lcg.web.cern.ch/LOG

[10] Storage Resource Management (SRM) Working Group  
https://sdm.lbl.gov/srm-wg

[11] P. Deutsch (Aladdin Enterprises), J-L. Gailly (Info-ZIP), May 1996. ZLIB Compressed Data Format Specification version 3.3  
http://www.rfc-editor.org/rfc/rfc1950.txt

[12] Lawrence Berkeley National Laboratory, September 2009. The Storage Resource Manager Interface Specification 2.2  
https://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html

[13] SSH Filesystem  
http://fuse.sourceforge.net/sshfs.html

[14] Filesystem in Userspace,  
http://fuse.sourceforge.net

[15] Portable Operating System Interface [for Unix]  
http://standards.ieee.org/develop/wg/POSIX.html

[16] J. Galbraith (VanDyke Software), O. Skaremmaa (F-Secure), July 2006. SSH File Transfer Protocol  
http://tools.ietf.org/id/draft-ietf-secsh-filexfer-13.txt

[17] The Large Hadron Collider  
http://public.web.cern.ch/public/en/lhc/lhc-en.html

[18] The gLite File Transfer Service  
http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/FTS

[19] Berkeley Storage Manager (BeStMan)  
https://sdm.lbl.gov/bestman

[20] YAIM Values for FTS Version 2.0  
https://twiki.cern.ch/twiki/bin/view/LOG/FtsYaimValues20

[21] K. Kaukonen, R. Thayer (SSH), July 1999. A Stream Cipher Encryption Algorithm "Arcfour"  
http://tools.ietf.org/id/draft-kaakonen-cipher-arcfour-03.txt

[22] National Institute of Standards and Technology (NIST), November 2001. Advanced Encryption Standard (AES)  
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[23] National Institute of Standards and Technology (NIST), May 2008. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher  
http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf

[24] sshfs(1) - Linux man page  
http://linux.die.net/man/1/sshfs

[25] Bestman Adler 32 Checksum  
https://twiki.grid.iu.edu/bin/view/Storage/BestmanAdler32Checksum