New Generation Computing, 25(2007)409-424
Ohmsha, Ltd. and Springer

Grid-based Data Access to Nucleotide Sequence Database

Frank Zhigang WANG, Sining WU, Na HELIAN
Centre for Grid Computing
Cambridge-Cranfield High Performance Computing Facility
MK43 0AL, UK
frankwang@ieee.org
Zhiwei XU
Institute of Computer Technology, Chinese Academy of Sciences
Yuhui DENG, Vineet KHARE, Chenhan LIAO, Chris THOMPSON
Cambridge-Cranfield High Performance Computing Facility
MK43 0AL, UK
Michael PARKER
Cambridge e-Science Centre, Cavendish Laboratory
Cambridge University CB3 OHE, UK

Received 30 November 2006
Revised manuscript received 23 May 2007

Abstract The International Nucleotide Sequence Database Collaboration (INSDC) exchanges sequence data on a daily basis across its three member organizations in the USA, UK and Japan. This paper studies how this sequence database in MySQL can best take advantage of the increased transfer bandwidth of a Grid-optimized data communication protocol. Within the context of the UK Government Project Grid-oriented Storage (GOS) and the EC Project EuroAsiaGrid, GOS File System (GOS-FS) has been developed in our lab, which melds distributed file system technology with high performance data transfer techniques to meet the needs of WAN/Grid-based virtual organizations. A real-world test shows that the INSDC sequence database backing up operation, mysqldump, over the GOS-FS protocol beats those over the classic NFS protocol by 6 times over the link between Cambridge and Tokyo. Best of all, the multi-streamed GOS-FS protocol remains fully compatible with existing IP infrastructures.

Keywords: Nucleotide Sequence Database, Grid Computing, Life Science, Grid-based Data Access, MySQL

§1 Introduction

The International Nucleotide Sequence Database Collaboration (INSDC), comprising the DNA DataBank of Japan (DDBJ), the European Molecular Biology Laboratory (EMBL), and GenBank at NCBI, provides sequence databases of gene, transcript and protein predictions. These three organizations exchange data on a daily basis, as shown in Fig. 1 MySQL database table dumps are available for all databases underlying the INSDC system. These international sequence databases exceed 100 gigabases: approximately 65,369,091,950 bases in 61,132,599 sequence records in the traditional GenBank divisions and 80,369,977, 826 bases in 17,960,667 sequence records in the WGS division as of August 2006.[1)]

All data sets generated by the INSDC project are freely available to download. INSDC encourages users to directly extract information from their databases via SQL rather than downloading huge flat files. They offer a public MySQL interface that accepts SQL queries as user 'anonymous'. Client programs for accessing this interface are available via MySQL. INSDC also supports downloading of many more correlation tables via the highly customisable BioMart data mining tool.[1)]

MySQL databases can be deployed on the Network File System (NFS) in a Wide Area Network (WAN) environment.[2)] Although NFSv2 and NFSv3 are optimized for a LAN, NFSv4 adds a number of powerful new features, including compatibility with IPv6,[3)] efficient client-side caching, transparent replication and migration, and strong security, making it WAN/Grid-oriented.[3)]

Unfortunately, attempting to share or access a database file across the

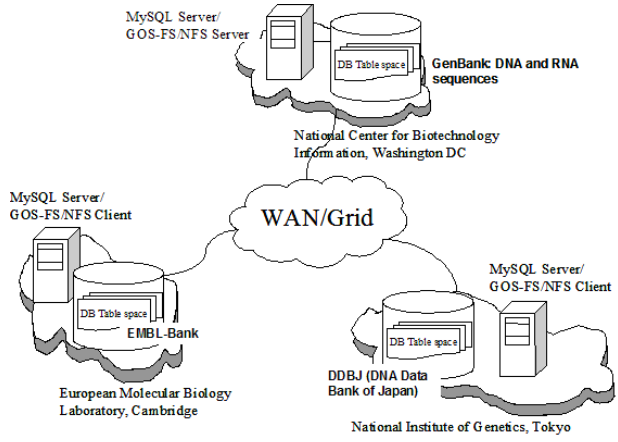

Fig. 1 The International Nucleotide Sequence Database Collaboration (INSDC) exchanges sequence data on a daily basis across the USA, UK and Japan. GOS-FS is used as an underlying engine to accelerate the database I/O. NFSv4 has also been mounted as a comparative protocol.

WAN/Grid often proves to be a frustrating, time-consuming experience. Because (single-streamed) NFS were designed to run across a LAN, not a WAN/Grid, their 田 hatty nature is negatively affected by WAN latency: after the packets are out on the fibre, the transmitter must stop until it gets an acknowledgement for those packets; after the round-trip delay time after starting, the acknowledgement gets back to the sender and the second burst with the default buffer size can be transmitted. As a result, applications using a single socket stream will not fill the TCP pipe, i.e., the traffic load is less than its capacity. When the buffer size is taken as its default value of 64 kB, the capacity efficiency is about 1%.[4] On the other hand, the network bandwidth is doubling every 9 months, which even "outperforms" Moore's law.[5] In general, the design of TCP has been influenced by the need to enforce fair sharing of precious network resources. For this reason, TCP's behavior is unnecessarily conservative for data-intensive applications such as Grid computing on high bandwidth networks.

This paper studies how the INSDC sequence database in MySQL can best take advantage of the increased transfer bandwidth of a Grid-optimized data communication protocol, GOS-FS. We continue in Section 2 with a discussion of file-sharing architectures in a WAN/Grid environment, their related research and our uniqueness and innovation. Section 3 discusses our implementation of GOS-FS. Section 4 reports the response time improvement compared to NFS-based applications. Section 5 describes a real-world test that exploits the latency reduction of our GOS-FS prototype. Section 6 concludes with a discussion of ongoing research.

§2 Background and Related Work

There are a number of good examples that the emergence of commercially viable open source database systems empowers the data management research community to impact the design and implementation of actual products, through the collaboration between the research community and the database industry. In the Badger project,[6] University of Copenhagen and MySQL AB studied how MySQL/InnoDB fully utilizes the available I/O bandwidth using prioritized asynchronous I/O in Linux.

The parallel NFS (pNFS) [7] is an extension of the NFS families. In pNFS, scalable (aggregate) bandwidth can be claimed by simply adding multiple independent servers to the network. Currently our GOS-FS and pNFS have no direct comparability since the word "parallel" has different meanings in these two contexts. "Parallel" in our GOS-FS implementation means multi-stream network file protocol. The technique that we have designed and used is a point-to-point data access based on this multi-stream mechanism, although GOS-FS does not require all the data in a single file system to be accessible through a single exported network endpoint. In the pNFS context, "parallel" means pNFS allowing transfer of data to proceed in parallel from many client points to many data storage endpoints.

Besides pNFS, there are other parallel filesystems. Panasas ActiveScale File System (PanFS) is a parallel file system that turns files into smart

data objects and then dynamically distributes data activity across Panasas StorageBlades.[8] This enables parallel data paths to be created between Linux clusters and StorageBlades for high performance access to large files. IBM General Parallel File System (GPFS) provides high-performance I/O by "striping" blocks of data from individual files across multiple disks (on multiple storage devices) and reading/writing these blocks in parallel.[9] Parallel Virtual File System (PVFS) "stripes" file data across multiple disks in different nodes in Linux Clusters, or Beowulfs.[10] PVFS-WAN and Lustre-WAN are WAN-optimized.[11] After having analyzed their source codes, we found that, although these two support multiple connections (streams), each file is still transported on one of these connections. In other words, they lack the mechanism that divides data into partitions that are sent over several parallel streams simultaneously and allows applications to achieve optimal TCP performance in a grid environment.

So far the multi-streamed GOS-FS and its variations are the first of their kind and hence we shall compare it with the single-streamed NFSv4, as the developers of GridFTP had compared their multi-streamed GridFTP with single-streamed FTP in the past.[12] To our best knowledge, there is not a related work to date in the literature on multi-streamed filesystems as the underlying engine to accelerate database I/O. Therefore the work is believed to be timely and novel due to its recent introduction by us.[13]

§3 GOS-FS: The Underlying Network Filesystem Protocol for MySQL

3.1 Overview

An increasing number of MySQL servers have been deployed for very large database applications. The MySQL pluggable storage engine architecture [14] allows a database professional to select a specialized storage engine for a particular application need while being completely shielded from the need to manage any specific application coding requirements. The MySQL server architecture encapsulates the application programmer and DBA from all of the low-level implementation details at the storage level providing a consistent and easy application model and API [14]. So while there are different capabilities across different storage engines, the application is shielded from these. The MySQL pluggable storage engine architecture interfaced with a WAN/Grid-optimized Protocol, GOS-FS, is graphically depicted in Fig. 2. The pluggable storage engine architecture provides a standard set of management and support services that are common among all underlying storage engines. The storage engines themselves are the components of the database server that actually perform actions on the underlying data that is maintained in GOS-FS.

3.2 Beyond FTP/GridFTP/NFS

FTP/GridFTP were the principal methods by which people moved files over the Internet. Bearing in mind the physical location of a FTP server, a user needs to specify the absolute path to that FTP server. In contrast to

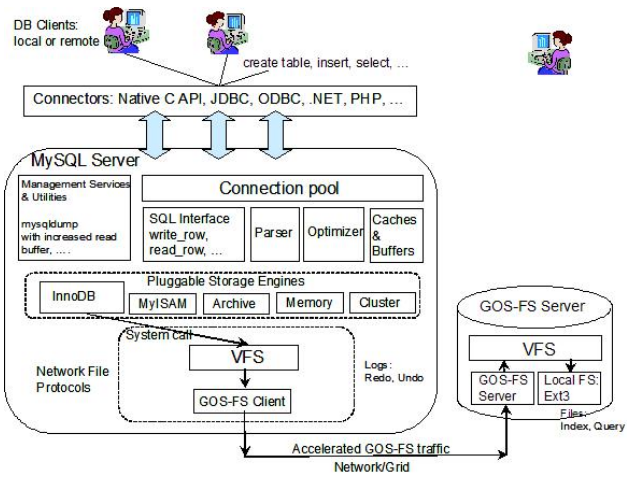

Fig. 2 The storage engines themselves are the components of the database server that actually perform actions on the underlying data that is maintained in GOS-FS

FTP/GridFTP protocols, a filesystem allows a user to use files on other network machines as if they were local. In addition, a filesystem also creates a single system image (SSI) unique to each user on each of a plurality of nodes in a Virtual Organization (VO). The Linux protocol that moves files across networks is NFS. NFS was developed by Sun Microsystems which allows a computer system to access files over a network as if they were on its local disks. This protocol has been incorporated in products by more than two hundred companies, and is now a de facto Internet standard. Once a client mounts a file directory on a remote file server (to its local file tree) via the NFS protocol, the client can access it as if it was local. For example, users can employ standard operating system commands to create, remove, read, write, and set file attributes for NFS-based files and directories. Because NFS was designed to run across a LAN (1-2ms of latency), users attempting to work with MySQL/NFS over a WAN must often deal with unacceptable wait times (60-1000 ms).[13)]

3.3 GOS-FS Security Mechanism

Grid applications not only require efficient transfer of a large amount of data in wide area networks but also demand security and authentication services that enhance data integrity and enable data access control. The security model of GOS is defined within the framework of Virtual Organization Membership Service (VOMS).[13)] The GOS-FS and its variations use public key infrastructure (PKI) associated with Grid Security Infrastructure (GSI) [15)] to authenticate identities of WAN/Grid members and to secure resource allocation to these members.[13)] The GOS-FS server requests and receives a host certificate from the CA. The GOS-FS (service) also needs to be registered in the Virtual Orga-

nization (VO) - Lightweight Directory Access Protocol (LDAP). Furthermore, the VO-LDAP generates a gridmap file to the GOS-FS server from the LDAP database for authorization and mapping. Beyond the role of certificate registration, the VO LDAP can perform other important roles for the VO, such as a service search engine and user authorization.

One of the interesting parts of a GOS-FS network is how often users communicate with the VO LDAP in relation to how often they communicate with GOS-FS servers. Essentially, the LDAP only needs to be accessed to initialize a session between a user and a GOS-FS server. After that, all communications can be handled between the user and the GOS-FS server. Data transfers that are broken off prematurely will need to be re-established, but it may not be necessary to authenticate again with the LDAP if the public key included in the certificate is still available to use. This design removes the VO LDAP as a potential storage I/O bottleneck.

3.4 GOS-FS Implementation

The emergence of high speed wide area networks makes grid computing a reality. However grid applications still have difficulties to achieve optimal TCP performance due to network tuning of TCP window size to improve the bandwidth and to reduce latency on a high speed wide area network.

Within the context of the UK Government Project Grid-oriented Storage (GOS) and the EC Project EuroAsiaGrid, a grid-based middleware solution known as "GOS-FS" has been developed in our lab, which melds distributed file system technology with high performance data transfer techniques to meet the needs of WAN/Grid-based virtual organizations. From its inception, the developed multi-streamed GOS-FS, via the fully-integrated parallel stream mechanism, is designed to deal with long-distance, bulk data, cross-domain and single-image file operations a capability lacking in the current versions of single-streamed NFS. GOS-FS alleviates WAN latency issues for Office/Database documents as well as other applications based on NFS standards. First-of-its-kind GOS-FS solution enables users to access and save Office/Database documents across the WAN/Grid at near-LAN speeds. The source code of the developed GOS-FS is of 40,000 lines in length and we have spent four plus man years in developing and revamping it. The corresponding work has recently been accepted by IEEE Transactions on Computers.[13)]

GOS-FS uses parallel streams to achieve very high transfer rates at a fraction of the memory cost. As shown in Fig. 3(c), GOS-FS opens multiple socket streams between the sender and the receiver for applications. The application data are then partitioned into segments. The number of segments is equal to the number of streams. The segments are sent through streams in parallel by different threads and are then reassembled by the receiver. The final data are presented to applications as if they were transmitted through a single socket. Figure 3 also illustrates why sending partitioned data through multiple socket streams can achieve near optimal bandwidth. In the figure the shaded areas represent socket buffer size and the large empty rectangles represent TCP pipe

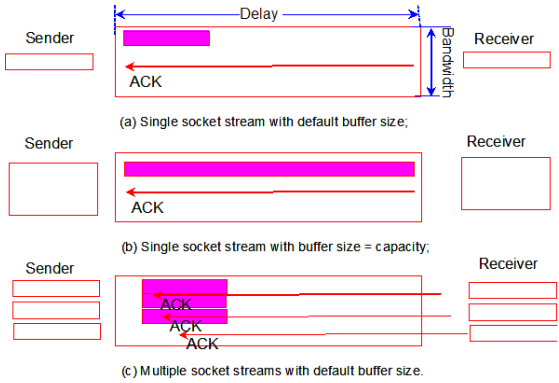

Fig. 3 Single or multiple TCP socket streams with different buffer sizes.

capacity (bandwidth*delay). Applications using a single socket stream will not fill the TCP pipe capacity. The capacity efficiency is about 1% when the buffer size is taken as its default value of 64 kB.[4)] Applications using either a single socket stream with an optimal buffer size or multiple socket streams can fill the TCP pipe, as shown in Fig. 3 (b) & (c). Chen [16)] showed that using parallel stream is more effective than tuning TCP window size. In addition, acknowledging the entire file requires the entire file to be retransmitted even if a single packet is lost, as shown in Fig. 3(b). Therefore applications using GOS-FS can obtain near optimal TCP performance without adjusting TCP window size.

Since Globus is becoming the overwhelming choice for grid development, non-Globus solutions will be difficult to melt seamlessly into popular grid environments. The GOS-FS uses Globus XIO to communicate with the parallel stream engine during a file read/write operation. As shown in Fig. 4, Globus XIO is broken into two main components: framework and drivers. The Globus XIO framework manages I/O operation requests that an application makes via the user API. The parallel stream driver is responsible for manipulating and transporting the user's data. The GSI driver performs necessary messaging to authenticate a user and the integrity of the data. The TCP driver executes the socket level transport code contained within to establish a connection to the given contact string.

The VFS (Virtual Filesystem Switch) is implemented in the kernel. This implementation conforms naturally to the standard POSIX semantics and provides Office/Database applications with seamless access to GOS-FS. A kernel-memory module, the GOS-FS Kernel Client, acts as a VFS interface. The GOS-FS server nominates a user-space daemon, the GOS-FS User Client, to penetrate the kernel/user boundary and to communicate with the Kernel Client.[13)]

Note that the Grid-based GOS-FS is a considerably complex piece of software. A detailed discussion of all the relevant modules is outside the scope of this paper.

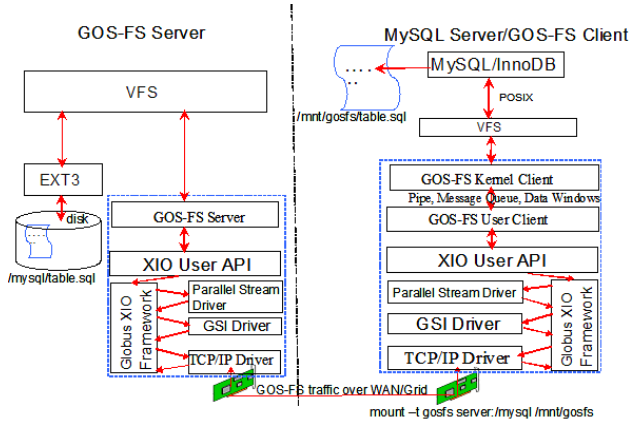

Fig. 4 The GOS-FS uses Globus XIO to communicate with the multi-stream engine during a database file read/write operation. A GOS-FS client mounts a file directory /mysql/table.sql on a remote file server to its local file tree /mnt/gosfs/ via the GOS-FS protocol, and thus the MySQL/InnoDB can access it as if it was local.

§4 Deploying MySQL/InnoDB on GOS-FS

4.1 Overview

The application programmer and DBA interact with the MySQL database through Connector APIs and service layers that are above the storage engines.[14] If application changes bring about requirements that demand the underlying storage engine change, or that one or more additional storage engines be added to support new needs, no significant coding or process changes are required to make things work. The MySQL server architecture shields the application from the underlying complexity of the storage engine by presenting a consistent and easy to use API that applies across storage engines.

4.2 MySQL InnoDB

Fully integrated with MySQL Server, the InnoDB storage engine maintains its own buffer pool for caching data and indexes in main memory, as shown in Fig. 2. InnoDB also provides MySQL with a transaction-safe (ACID compliant) storage engine that has commit, rollback, and crash recovery capabilities. InnoDB stores all records inside a fixed-size unit which is commonly called a "page." A page contains records, but it also contains headers and trailers.

InnoDB separates log and data files (temporary files are organized as data files). Log files are managed as a circular structure to which redo log records are appended. Data files are organized in tablespaces. A tablespace consists of one or several operating system files. Each tablespace is structured in segments. Segments are associated to tables. For example, a table with a primary index is stored using a data segment and an index segment. Each segment is organized in

extents of 64 pages. Page size is fixed at 16KB. InnoDB issues the following I/O requests: Sequential writes of log records. A write request containing log records is submitted by a query thread at commit time, or if the cache that stores log records in memory is 50% full. In addition, the background server thread forces log records to disk every second.

- Random writes of dirty pages. If there are dirty pages in the buffer pool and if the I/O activity is low then pre-flushing takes places and InnoDB issues asynchronous write requests to write committed dirty pages to disk. Now, if there is memory pressure InnoDB issues synchronous write requests to free buffer space.
- Random reads for physical I/O. Random reads are submitted by query threads if the page they access is not in the database cache.
- Sequential reads during prefetching. The query thread performs prefetching as follows. If it accesses pages with a sequential pattern then it prefetches extents, one at a time. Otherwise, if a query thread accesses more than a tunable number of pages from the same extent, then the whole extent is prefetched. Pages are allocated in the database cache as soon as I/O requests are submitted. A query thread might access a page for which the I/O request has not yet completed. In that case the query thread waits on a latch and is notified when the I/O completes.

§5 Sequence/Genome Database Tests over MySQL/GOS-FS

5.1 Configuration and Methodology

For the performance tests, both response times and bandwidth utilization across the LAN/WAN infrastructure were logged. Nucleotide Sequence/Drosophila Melanogaster Genome databases [18] were used. The netperf tool [19] is used to measure maximum TCP bandwidth, allowing the tuning of various parameters and characteristics of the above experimental platform, like optimizing the default read buffer size of mysqldump to fill the TCP pipe. A standard stopwatch was used to take the response times. Each of these experiments was repeated 3 times.

We compared MySQL database populating/scanning performances over GOS-FS to those applications deployed over NFSv4. NFS and GOS-FS can run on precisely the same hardware and operating system. They can, in fact, coexist on the same machine and be used simultaneously. The performance differences we observed were due to the design and implementation of the filesystem protocols and were not artifacts of hardware, network, or operating system variation.

5.2 Populating/Scanning Sequence Databases Remotely via GOS-FS

The International Nucleotide Sequence Databases are designed to provide and encourage access within the scientific community to the most up to date and comprehensive DNA sequence information. The most important source of new data for the Sequence Databases is direct submissions from scientists. The

Sequence Databases depend on their contributors to help keep the database as comprehensive, current, and accurate as possible. On the other hand, repeated database scans and pattern matching are used to collect counts for the candidate itemsets in a typical data mining process of identifying patterns and relationships in the nucleotide bases or molecular structures, and of 3D protein structures.

Extensive performance tests were conducted to measure the effectiveness of MySQL over GOS-FS protocol, in terms of the response-time improvements delivered to users. We measured the response times, in seconds, of a remote client requesting MySQL. Results show that 94.9 seconds were consumed when populating a MySQL table with 100,000 records with 40 ms of RTT (Round Trip Time). When we added the GOS-FS server/client gateways to the equation, the same 100,000 records required 79.5 seconds to open across the link with #tcp=16, representing a response time improvement of 16.2%. We ran the same test, this time transferring 1,000,000 records. Without the aid of the GOS-FS acceleration, the table was populated in 980.3 seconds. When the GOS-FS server/client gateways were added to the test bed, the populating took 763 seconds across the link with #tcp=16. The response time improvement jumps to 22.2%. Next, we measured the response times of a remote client scanning of the populated database table with RTT=40-80 ms. When we scanned 100,000 records over the link, the scanning operation took 33.4 seconds with no acceleration applied. Over the same connection, using the same table, the table-scanning operation took just 22.5 seconds when the GOS-FS gateways were added to the configuration (#tcp=16). That represents a 32.6% response time improvement. The improvement jumps to 67.3% when scanning 1,000,000 records over the same link.

5.3 Genome Database Moving via GOS-FS

The three organizations of the International Nucleotide Sequence Database Collaboration (INSDC) exchange data on a daily basis. Individual users can download from INSDC's sequence databases large or small segments of genome sequence from a variety of organisms (e.g., yeast, human, fly, worm, mouse, plants), preserving the gene annotation that is associated with that sequence. It takes about 6-10 hours to build a personal GenBank [1)] depending upon traffic and if including the updates. Database documents are typically accessed in chunks of data at a time, slowing down the transfer of files across the WAN. Increasing the read buffer size of mysqldump (the default is 256 kB) enables entire database file to be moved across the WAN before the data is actually required, so information is already localized when users need it. The result is more data sent in fewer trips and less WAN waiting time. In addition, applications using multiple socket streams in the underlying GOS-FS fill the TCP pipe to achieve optimal bandwidth. This combination reduces database file access time while reducing the number of performance-impacting round trips over the WAN.

The test in Fig. 5 measures the response time improvement (RTT=80ms, #tcp=16) as a function of the mysqldump read buffer size ranging from 1MB to 32 MB, namely database moving over NFS and over GOS-FS, respectively.

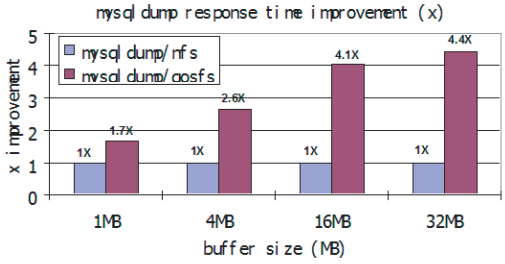

Fig. 5 Database moving via mysqldump/gosfs as a function of the buffer size over a link with rtt=80ms, #tcp=16. The database contains 1,000,000 records, occupying around 60MB.

In a database replication operation, mysqldump retrieves table contents from a table and buffer it in the database buffer accumulatively. When the contents fill the buffer, they will be dumped to disk by calling the underlying GOS-FS engine. In principle, the larger the buffer size, the more benefit the database moving operations gain from the increased bandwidth of GOS-FS. In MySQL populating/scanning (Section 5.2), a 64-block request just corresponds to 1 MB dumped contents and 64-block requests occupy only 40% of the overall requests. In a MySQL moving operation, a sequence of dumps need to be performed until the database is backed up and each dump flushes the entire content in the buffer, with a fixed size between 1MB to 32 MB, to disk. This is why the speedup of database moving operations is much higher than that of insertions and scans.

The test described in Table 1 measures the response time improvement (16 tcp streams) as a function of RTT in ms. In a Wide-Area Network environment with RTT = 40-320 ms, MySQL/GOS-FS outperforms the classic NFS by factors ranging from 3 to nearly 7 times, and this gap widens when RTT is increased still further. Note that the speed-of-light-in-fiber is 150,000km/s so the equivalent physical one-way distances for RTT values of 40, 80 and 160 ms are 3,000, 6,000, and 12,000km, respectively. These are relatively long distances since GOS-FS is designed to deal with long-distance, cross-domain and single-image data access operations, enabling users to connect remotely to the GOS-FS server.

We have written a shell script that will package and transfer a database for backup. The packaging results in two files with the extension of .sql. First the script collects all data from both MySQL database server and a local file system to temporary directory /backup. Next, script logins to ftp/gridftp server and creates a directory on the remote site. Eventually the script dumps the two files from /backup to ftp/gridftp server. This is a simple backup solution for a database administrator who runs his/her own ftp/gridftp server and MySQL server. In contrast, mysqldump is a client program for dumping or backing up mysql databases, tables and data. mysqldump command deployed on GOS-FS automates the above procedure.

For comparison, we have also listed the response time of using FTP/GridFTP in the tables. Observation shows that packaging a MySQL

Table 1 Response time(s) of moving a Genome database via mysqldump/gosfs. The link is with #tcp=16. The database contains 1,000,000 records, occupying around 60MB

rtt(ms)	mysqldump/nfs	mysqldump/gosfs	ftp	gridftp
40ms	43.8	14.4	51.8	22.9
80ms	80	19.7	92.2	32.3
160ms	154.2	26.4	172.2	52.9
320ms	346.5	55.8	361.2	87.5

database with a resulting size of 60 MB takes 8-32 ms in the packaging/ftp/gridftp solution. Contrarily, in the mysqldump/GOS-FS solution, there is no need to perform packaging since data collected from the MySQL database server are directly written to the remote site via the underlying GOS-FS engine. In generally, the mysqldump/GOS-FS solution outplays the conventional packaging/ftp/gridftp solution in terms of dumping time. Furthermore, the mysqldump/GOS-FS solution also creates a user-friendly SSI and relies on standard commands to backup databases whereas the packaging/ftp/gridftp solution is too tedious to package the databases and specify the physical address of the FTP server.

5.4 Real-world Test over EuroAsiaGrid

The following real-world tests demonstrate the value of the GOS-FS protocol accelerating MySQL over the EuroAsiaGrid network resources. Funded by the European Commission (EC), the EuroAsiaGrid consists of over 10 European and Asian partner institutions.[20)]

In a real-world test, the Cambridge-DDBJ (DNA DataBank in Japan) datalink results in 278ms of roundtrip latency and 23 router hops where the geographic distance is over 12,000 km. At both sites, the servers and the clients are connected to their LAN by a fast 100 Mbit/s Ethernet link. The data travels from Cambridge to London, then from London through a trans-Atlantic link to Huntington, the American West Virginia gateway, through a trans-Pacific link and finally arrives in Japan.

Over the lifetime of a connection, bandwidth and delay change (due to transitory queuing, congestion and route changes, etc) implies that the bandwidth-delay product (BDP) of the connection also changes. A 24 hour test supports this claim. Here we show the BDP between Cambridge and Tokyo at 10 minute intervals.

A Drosophila Melanogaster Genome database of a size of 652MB (122 tables) was backed up, since it was found that this was large enough to average out most of the network fluctuations. Fig.6 measures the real-world response time improvement as a function of the number of tcp streams (#tcp). Increases in the improvement with increased #tcp can be seen toward the right side of the graph. The maximum improvement goes up to 5.9x with 64 parallel streams. However, GOS-FS would eventually consume too much time in managing too

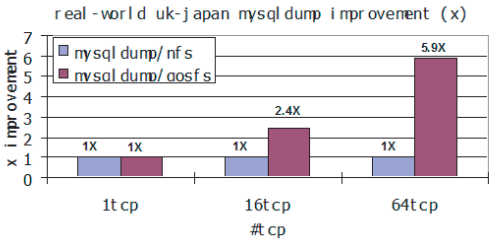

Fig. 6 Real-world response time improvement as a function of the number of tcp streams (#tcp).

many TCP streams, slowing down such an improvement with further increased #tcp.

§6 Conclusion

The Grid Systems for Life Sciences result in connectivity needs for clusters of servers numbering in hundreds of nodes. Such grids are often characterized by networks with large bandwidth-delay products. It is a good way to deploy databases on a network filesystem such as NFS. Unfortunately, all technology has strengths and weaknesses. NFS deployment shows weaknesses in terms of using only a small fraction of available bandwidth.

In this paper, our goal was to find out whether the International Nucleotide Sequence Databases on top of the multi-streamed engine, GOS-FS, take best advantage of the increased available I/O bandwidth. In conclusion, as an alternative of NFS, the GOS-FS data communication protocol [13] addresses the challenge of sharing files over a WAN/Grid with "LAN-like" performance, which is a highly beneficial aspect to the Grid-based Life Science e-Research community.

Acknowledgements

The project team is indebted to generous contributions from Dr. Len Oswaild, Dr. Jonathan Haynes and Dr. Howard Jeffrey of Cambridge-Cranfield HPCF, Prof. Zhenjiang Hu of The University of Tokyo, Ms. Xiao Lijuan and Prof. Zhiwei Xu of Institute of Computing Technology, Dr. Dr Manuel Sanchez and Prof Jose Garcia Carrasco of Murcia University, Dr. Mark Barker of Portsmouth University, and Dr. Mark Hayes of Cambridge e-Science Centre.

References

1) "International sequence databases exceed 100 gigabases," www.ncbi.nlm.nih.gov/Genbank/
2) Colaco, G. and Suggs, D., "Database Performance with NAS: Optimizing Oracle on NFS," *Technical Whitepaper*, May 2004.

3) Sun Microsystems, "Network File System," www.sun.com.
4) Andrew S. Tanenbaum, *Computer Networks (4th Edition)* Prentice Hall, 2002.
5) GridCafe, "Breaking Moore's law, A brief history of the Grid," gridcafe.web.cern.ch, 2006
6) Hall, C. and Bonnet, P., "Getting Priorities Straight: Improving Linux Support for Database I/O," *Proc. of the 31st VLDB Conference, Trondheim*, Norway, 2005.
7) Gibson, G., Welch, B., Goodson, G. and Corbett, P., "Internet-Draft, Parallel NFS Requirements and Design Considerations," October 18, 2004.
8) "The Panasas ActiveScale File System," www.panasas.com/panfs.html, 2006
9) "IBM General Parallel File System," www-03.ibm.com/, April 2006
10) Carns, P. H., Ligon III, W. B., Ross, R. B. and Thakur, R. , "PVFS: A Parallel File System For Linux Clusters," *Proc. of the 4th Annual Linux Showcase and Conference*, Atlanta, GA, pp. 317-327, October 2000.
11) "Lustre: A Scalable, High-Performance File System," www.lustre.org/docs/whitepaper.pdf, 2006
12) "Grid Ecosystem GridFTP," www.globus.org/
13) Wang, F., Wu, S., Helian, N., Deng, Y., Parker, A., Guo, Y. and Khare V. "Grid-oriented Storage: A Single-Image, Cross-Domain, High-Bandwidth Architecture," *IEEE Transaction on Computers*, April, 2007
14) Schumacher, R., "MySQL Developer Zone," MySQL AB, http://www.mysql.com/, October 19, 2004
15) GlobusWORLD, www.globusworld.com/program/program.php, 2006
16) Chen, J., Akers, W., Chen, Y. and Watson III, W., "Java Parallel Secure Stream for Grid Computing," http://www.ihep.ac.cn/ chep01/abstract/10-008.htm, 2005
17) MySQL AB, "MySQL Internals Manual," December, 2005.
18) European Molecular Biology Laboratory, http://www.ensembl.org/Drosophila_melanogaster.
19) netperf, www.netperf.org/netperf/NetperfPage.html.
20) Cambridge-Cranfield High Performance Computing Facility (CCHPCF), http://www.hpcf.cam.ac.uk/.

Frank Z. Wang: He is a Fellow of British Computer Society, Professor and Chair in e-Science and Grid Computing, Director of Centre for Grid Computing within the context of Cambridge-Cranfield High Performance Computing Facility (CCHPCF) [http://www.hpcf.cam.ac.uk/research.html]. He has been elected as the Chairman (UK & Republic of Ireland Chapter) of the IEEE Computer Society from January 2005.

Sining Wu: He received his PhD at Institute of Computing Technology (ICT), Chinese Academy of Sciences in 2004. He was involved in the development of 10-TFLops Dawning, listed in the Top 500 Supercomputer List. He is now a Research Officer at Centre for Grid Computing. His current research interests include distributed operating systems, storage systems and file systems.

Na Helian: She received the PhD degree in computer science in 1992. She has various working experiences in Japan, Singapore and UK. She is now a Senior Lecturer and the Director of MSc Data Mining Program at Department of Computing, Communication & Mathematics, London Metropolitan University, UK. She is the co-investigator of the UK Government EPSRC/DTI grant Grid-oriented Storage (GOS)

Zhiwei Xu: He received his PhD from University of Southern California. He is a professor of the Institute of Computing Technology, Chinese Academy of Sciences. His research interests are grid and cluster computing, computer architecture and secure operating system.

Yuhui Deng: He obtained his PhD in storage system in 2004. He had spent nearly 4 years in the National Key Laboratory of Data Storage System in China, being involved in 3 Chinese Natural Science Foundation projects and covering computer architecture, parallel I/O, network storage, virtual storage, etc. He is now a Research Officer at Centre for Grid Computing.

Vineet R. Khare: He received his BTech degree from India Institute of Technology (IIT) Kanpur, India, in 2001. His PhD degree in Computer Science is to be awarded by The University of Birmingham in December 2006. Currently he is working as a research officer at the Centre for Grid Computing. His research interests include swarm intelligence, co-evolution, multiple neural network systems and multi-objective evolutionary algorithms.

ChenHan Liao: He is a PhD candidate in Cranfield University, School of engineering, applied mathematic and computing group. His research topic is utilizing data mining technique to improve data storage system performance and exploiting data mining for intelligent data management.

Chris Thompson: He is the Professor of Applied Computation. After leaving Oxford University with a BA in Mathematics and an MSc in Numerical Analysis, he joined the Applied Maths Group at AEA Technology, Harwell. He has also worked at the Advanced Computer Research Facility, Argonne National Laboratory, Chicago and at Bergen Scientific Centre, IBM. He holds a D Phil from the University of Bergen.

Michael Andrew Parker: He holds a PhD from the University of London, and an MA from Oxford. He is now the Director of the Cambridge eScience Centre. He is responsible for grid computing initiatives across the University, covering a wide variety of projects in physics, life sciences, earth sciences, medicine, chemistry and engineering, all requiring the sharing of large computational and data resources. He sits on the Management Committees of the Cambridge-Cranfield High Performance Computing Facility.