



THE UNIVERSITIES OF LEEDS, SHEFFIELD AND YORK

The White Rose Grid
e-Science Centre



Exludus Evaluation for ETF White Rose Grid e-Science Centre

Christopher Mountford, University of York

29-May-2007

Abstract:

The Exludus replicator file sharing software is tested and analysed on the White Rose Grid. Its installation and use are described with reference to the possibility of deployment onto the National Grid Service. A small set of performance tests against a simple file serving system (NFS 3) and a more expensive cluster file system (IBM GPFS) are performed and it is found that Replicators aggregate file transfer speed scales linearly with the number of nodes involved in the tests, giving far superior performance to NFS in cases where large numbers of nodes require read access to a single file from the file server.

UK e-Science Technical Report Series

Report UKeS-2007-04

Available from http://www.nesc.ac.uk/technical_papers/UKeS-2007-04.pdf

Copyright © 2007 The University of York. All rights reserved.

Introduction

This document is an evaluation of the Replicator file replication software provided by Exludus. This software manages and provisions data files required for large compute tasks across nodes in a compute cluster. By using a variety of network technologies it is able to scale data throughput with the number of nodes in the cluster, giving very high aggregate bandwidth (of the order of $n*b$ where n is the number of machines, b is the bandwidth of the network cards).

We provide the results of the initial scaling tests performed on the York White Rose Grid cluster with comparison to NFS performance.

1 General Information

1.1 Provider

Exludus provides technologies for optimizing data intensive computational workloads and was founded in 2002 by Benoit Marchand. The company headquarters are located in Montreal, Canada, with offices also located in Kent and California.

The Replicator software is the core product offered by Exludus and is available as a stable product. The software is available in two versions, entry and enterprise with a version for edge grids under development.

1.2 Licensing

The replicator software is released in binary format and is closed source. Licenses are per core and include support as well as software binaries and updates for the license duration. Node locked and floating licenses are available.

1.3 Platform support

The software suite runs on Linux, Solaris and Mac OS X. A windows version of Replicator 2.1 will be released in April 2007.

The software was evaluated on the York White Rose Grid cluster, which consists of 24 dual processor, dual core Opteron machines. The computers are connected to a gigabit switch and a low latency infinipath network is provided for MPI and GPFSt. Tests of Replicator and NFS were performed over the gigabit network due to ongoing issues with the infinipath drivers.

1.4 Ongoing Support

Support is included with the software licenses. A complete installation service is offered with Exludus staff present to assist with the installation and initial testing of the software (a process which should take only one day in most cases). Response to support requests was good, with initial response within 24 hours and bug fixes being available within a few days at most.

2 System Management

2.1 Documentation

A basic guide for system managers covering installation and initial configuration is provided. Software installation is straightforward and ongoing maintenance requirements have so far proved minimal. In any case, installation and initial testing can be carried out by Exludus staff, who will also provide assistance in initial training for system administrators.

2.2 Server deployment

The server software should be deployed on a cluster head node (ie an node to which users have access as jobs must be submitted from this node). It is available for Solaris 10, linux 2.4

and greater and Mac OS X. The software does not require any unusual software libraries and the only hardware requirement is a sufficiently large disk partition for the files to be staged to. Installation for linux is from an rpm package. Init scripts are required for red hat and SUSE based systems. Firewalls are not an issue as the software is confined to operating on the cluster sub net, however a (configurable) port may be needed for the Replicator ftp client if this service is to be provided. The server on which the software is installed must be able to submit jobs to the cluster/grid Resource Manager. Integration is provided with the software for PBSPro and Sun Grid Engine and other job managers can be supported upon request. Configuration of the software is achieved through variables in the init script and these are clearly documented.

The network on which the software is deployed must support multicast and/or broadcast and a number of multicast routes will need to be added to client and server machines. A tool is available from exludus to test network connectivity prior to purchase to ensure the network is compatible. Some configuration changes may be required to switches to enable multicast. Support is provided for WAN use in other Exludus products.

2.3 Client deployment

Clients are deployed on each cluster node. Similar to the server node sufficient disk space must be made available to cache local copies of the files. If it is not possible to create a dedicated disk partition, files can be cached to a dedicated directory on an already existing partition (we used /var/tmp) with a trivial change to the configuration files (we were also able to achieve this using symlinks with even less effort), however the documentation did not cover this in detail.

The Replicator user on each machine must be able to set/clear complex values on the resource manager.

Administration of the current version of the software (v. 2.0.1) is straight forward as the software is stable and requires little administration after deployment. The ports used by the software are documented clearly and can be adjusted in the init scripts if required by firewalls installed on the client nodes.

2.4 Our Deployment Experience

Initial installation of the software was performed over a period of 3 days, however a large part of this time was due to a bug that the installation to our hardware exposed in the Exludus software. Ignoring the time taken for a patch to be produced and released the installation and testing took approximately 6 hours from start to a usable system using the documentation provided. Re installation (upgrade to a new version) took approximately 30 minutes.

2.5 User account management

The software makes use of the unix user accounts. Any local user on the server can make use of the service, no additional authentication is required or supported. The services run under a dedicated non root user account. No user authentication is performed, the file they provide is replicated to all nodes with world readable permissions whilst it is in use. Exludus have stated that better authentication and security will be available in future replicator versions however no roadmap is yet available.

2.6 High availability features

The software has so far proven to be highly stable, however the basic version (which we are using) does not allow multiple gateways, so failure of the gateway machine will cause any ongoing data transfer to fail and will make the submission of jobs which require Replicator impossible until the machine is brought back into service. The only other single point of failure is the network hardware. The enterprise edition supports multiple gateways, allowing for fail over should a gateway machine fail, however we have not tested this feature.

2.7 *External Services*

The software does not make use of any external software.

2.8 *Scalability*

Within our cluster the file transfer rate scaled, as expected, linearly with the number of nodes on which the clients ran (maximum 20 nodes). Studies by Exludus on large (100s of nodes) clusters have shown linear scaling of aggregate transfer rate. The software as tested is designed around single cluster operations. However features which support multiple clusters (such as dynamic grouping and multi gateway/gateway failover support) are provided by the enterprise edition.

3 **User Experience**

User interaction with Replicator has a very shallow learning curve, such that users should be able to use the software with very little training required. The simplest method provides a command line tool which acts much like the SGE/PBS qsub command. Using this method, users can submit a job using the command line tools they are used to.

This method requires simple modifications to the job script (to use the replicated file location) and provision of the path to the files which must be replicated on the command line. More sophisticated data staging can be provided by using the Exludus MLP (meta language processor) to describe a job and its data requirements using a simple scripting language. Both methods (including the syntax of the meta language) are described in the replicator documentation.

The basic changes required to a user job are:

- modify paths to replicated files in job script, for example, changing \$HOME/data.dat to \$SPOOL_DIRECTORY/data.dat.
- Submit the jobs using the command line tool, using the same syntax as for SGE/PBS (depending on which is in use). -OR- write a job data requirement description using the meta language provided and submit it to the MLP.

Jobs will be held on the server and can be scheduled to the resource manager as soon as the transfer of the data files is complete. The file cache is automatically purged of data files when all tasks in a job are complete.

Job submission via grid middleware is not supported, however an experienced administrator may be able to provide this via modifications to the globus gram job manager script for both PBS and SGE (this is not something we attempted as all job submission is currently via command line on our systems).

A sample meta language file and modified SGE job script is provided in appendix A for a sample BLAST run..

3.1 *Documentation and usability*

As with the system documentation, user documentation is simple and easy to understand, with examples being provided for the basic use cases. The combined user and technical documentation is 50 pages. This documentation is sufficient as the user interface is extremely simple. Any user who is used to command line submission should be able to pick up replicator use using the documentation provided within half an hour by going through the provided examples. The task definition file format is easy to follow and user documentation largely consists of a description of the syntax of this file and a guide to job submission.

3.2 *Joining the grid*

A user accesses the software using their existing unix login account.

3.3 *Security from the user perspective*

Access to the services make use of the users unix account, as such no additional work is required from the user to access the service once they are on the system. In many ways this could be compared to using a normal unix shared (ie NFS3) file system, once on a machine, the users (or there batch jobs) have access to there transferred files as soon as a transfer is completed. However users should be made aware that there data is transmitted unencrypted and can be viewed by anyone whilst it is in the cache.

The replicated files are completely unprotected and are stored readable by world, so users should be aware that these files should not contain sensitive information.

3.4 *Verification*

The command line tools provide feedback on job status (ie submission of job to resource manager, file transfer status).

3.5 *Performance*

A series of test have been run using the York White Rose Grid HPC cluster, which consists of 24 dual processor, dual core nodes. This was a case study using blast to perform a large number of searches across a protein dataset. All of the searches make use of the same (np) data set (in total 5GB) and run in parallel across all 4 cores on each machine.

Results from the tests are shown in figure 1.

As we see from these results, replicator maintains a constant data transfer time, irrespective of the number of machines to which the data must be transferred. However we see that the NFS transfer time increases linearly with more than 2 nodes due to sharing the link from the server to the switch. It appears at first glance that the data transfer takes considerably longer for replicator than NFS for single processors. Further investigation has shown that only about 1.4 GB of the data is actually used for the blast search. So for this case NFS is able to just cache the required blocks in memory rather than transferring the entire set of files as Replicator does. Even given this inefficiency we see that at any more than 12 machines, Replicator becomes quicker than NFS.

To try to overcome the slow NFS performance one might try caching the files to a local tmp directory (as Replicator does) using scp/ftp/gftp or simply copying over NFS. In this case we must transfer the files to each machine, which we find takes 125 seconds *per machine*, over 40 minutes to the whole cluster (for a job which can complete in 30 seconds!) compared to 110 seconds for Replicator to provision the files to the entire cluster.

Finally, as Replicator is asynchronous, so file transfers can take place whilst other jobs are running with minimal overhead. From the users perspective, they would submit a BLAST array job using replic submit: There would be a short delay (110 seconds in the example case) before tasks could begin to execute whilst files are staged, which can take place whilst other jobs are running. Once staging is complete, the users tasks are released and begin execution as cluster compute resources become available. As they are accessing the files on local scratch space, each task requires only 30 seconds to complete. This gives a task throughput of about 40 searches per minute.

We can compare this to the case using a central NFS server, where we get task throughput of approximately 5 searches per minute.

So, for this test case (which was chosen as it represents a task which has in the past been run on our hardware) we see an increase in throughput of approximately 8 times for a cost of

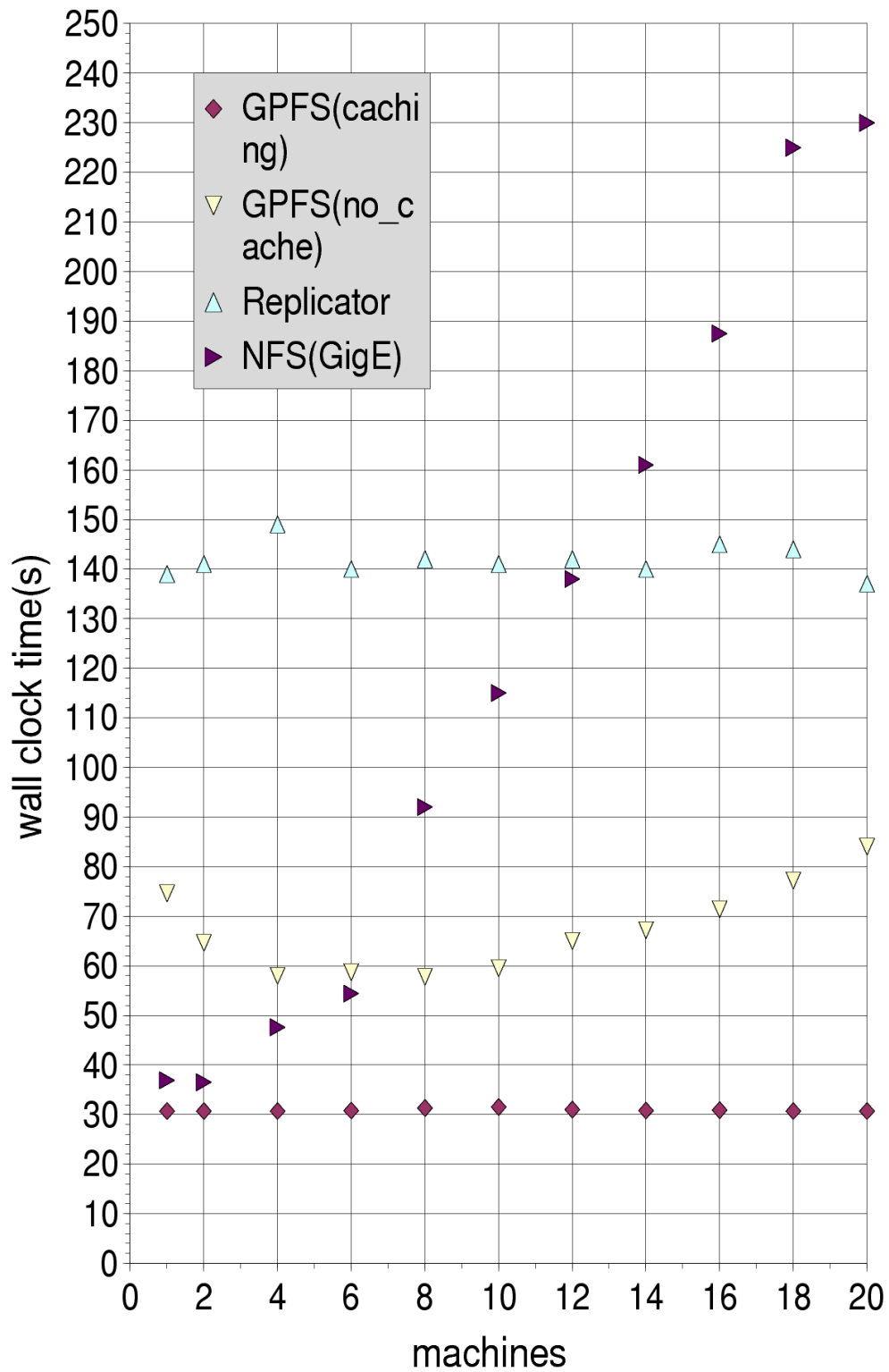


Illustration 1: Performance of NFS, GPFS and Replicator in simulated BLAST searching. Time taken is for file caching and processing combined. Processing time is approximately 30 seconds.

around £50 per node per year, approximately 5% of the node cost over its expected lifetime (4 years).

4 Development tools

The software is provided under a closed license and is designed to stand alone, sitting between the file system and resource manager. As such, no development APIs are provided for the Replicator software, however, a programming API is available for the Asynchronous Results Transfer.

5 Technical information

5.1 Architecture

The software is not a Service Oriented Architecture.

Replicator consists of the following components:

- **Gateway(replicd):** Replicd should be installed on the gateway node and is responsible for replicating files across nodes running the client replicd package. The metalanguage processor and ftp front end must be installed on the same machine.
- **Client(replicd):** Replicd should be installed, running on client mode on all of the machines to which files can be replicated and jobs run.
- **Metalanguage Processor:** The metalanguage processor interprets the metalanguage file submitted by the user (or produced by the replic-qsub command) and uses replicd to distribute the file. It also interacts with the resource manager to handle scheduling of the job.
- **FTP front end:** This front end allows files to be submitted for replication by a user using a standard ftp client. It is also used by replic-qsub to replicate files.
- **job submission client (replic-qsub):** Allows submission of jobs from off the gateway node, requires the FTP front end and metalanguage processor be available on the gateway.

Jobs are submitted using a metalanguage file or the replic-qsub command line utility. On job submission, the job script is submitted to the resource manager and the replicd starts transferring data to the nodes. As each node completes the data transfer it notifies the resource manager (for example using SGE compex variables).

The data transfer is asynchronous and can take place whilst jobs are running on the node with low processor overhead (typically < 5% of a 4 core machine).

5.2 Standards and Specifications

The product works over standard TCP/IP networking.

5.3 Security

Replicator has minimal security provided for the caches. However, cached files are read only, so there is no risk of data corruption (the original files are not accessible using replicator, only cached copies). Users need to be aware however that there is no access control for files in the cache (they are stored readable by everyone, owned by replicator user), so this application is not yet suitable for use with sensitive/private information as it would be exposed to all users on the system.

Access to the replicator daemons themselves is controlled by requiring user logins (using pam) to use the command line clients and ftp service.

It should be noted that at present replicator is not suitable for sensitive data on a shared access system, however data access control should be provided in future versions.

5.4 Industrial support

Replicator runs on standard hardware/operating system. (Linux, solaris, MacOSX on x86, x86-64 and powerPC hardware).

5.5 Capabilities and functionality

Replicator provide rapid, asynchronous provisioning of files to a cluster of machines. In the right usage environment (large arrays of jobs with large data sets, for example parameter studies/BLAST jobs) this can considerably speed up (by 2 or more orders of magnitude) job throughput on a large cluster over that which could be achieved serving files over NFS. It is able to do this without requiring investment in new infrastructure (infiniband for example) and with minimal disruption to the running of the system. For the sort of jobs at which Replicator is targeted it has the potential to dramatically increase the productivity of an existing cluster by eliminating data transfer times.

Since the tests were carried out, Exludus have released art (asynchronous results transfer) which provides functionality similar to replicator for collecting large sets of results files in an asynchronous manner. This replicates the functionality provided by grid middleware (ie globus reliable file transfer) but may be suited to applications where the complexity of installing and using a full set of grid middle ware is not required.

6 Conclusions

6.1 Suitability to deployment within NGS

Replicator supports the architecture provided within NGS (x86 based, rhel 3,PBSpro) and it would be expected that no technical problems should be encountered with its deployment. As such, deployment within a single NGS node could be achieved with a minimum of time, however integration into GT preWS GRAM is not currently provided and would require the production of custom gram job submission scripts.

In cases where there are large numbers of jobs, each requiring access to the same data file, replicator provides for greatly increased efficiency of use of resources.

6.2 Summary

Replicator provides the following capabilities:

- Rapid provisioning of files to an entire cluster. File transfer times to the entire cluster being similar to the time taken to transfer to a single machine using more traditional methods (scp, ftp, grid ftp).
- Elimination of network bottle necks caused by a large number of machines requiring data transfer from a single centralised file server.
- Increased utilisation of existing resources as pre-staging of files (as well as return of results via ART) can be undertaken with low overhead whilst computational jobs are running. Eliminating “dead” time on cpu.
- Automatic management of file caches, requiring minimal intervention from users or system administrators.
- Low administration overhead.
- No requirement for expensive dedicated networks (eg infiniband), replicator works over existing ethernet interconnects, provided they support at least broadcast and ideally multicast.

It is most suited to use in tasks which are:

- Embarrassingly parallel with large data files which need to be provisioned at a large number of nodes simultaneously (example: parameter sweeps).
- Require a lot of time to transfer data to the node or return large result sets (using ART).

- Efficient provisioning of large files to large numbers of machines simultaneously for future use (eg. Updating large numbers of virtual machines on a campus grid of tens of thousands of machines).

In these circumstances, on large clusters (100+ nodes) significant improvements (theoretically 1-2 orders of magnitude) have been observed. Even on our small cluster and running tasks which are not ideal for replicator we see improvements over NFS for cases where a large number of nodes are being utilised. In a production environment (when the data transfer would be taking place whilst other jobs were running) we see throughput over 20 machines increase by approximately 700% given data which is too large to be cached.

It is possible at least in part to “pre-cache” data using the job script and scp/gftp, however, this eliminates one of the areas in which a shared file system is more efficient, ie transferring only that data which is accessed, which leads to a significant “dead period” (over 2 minutes per node on our cluster) where cpus are not being utilised and requires additional clean up tasks to be performed when all of the jobs are complete.

In summary, Replicator provides transparent, asynchronous data provisioning and return facilities to large clusters with a low learning curve for users and low overhead for system administrators.

7 Appendix A

Metadata and script file used in the example case:

7.1 Task description file

```
#
#Task Description file (*.tdf)
#
#
# Work Load Manager
#
#
WLM { #Job type for the WorkLoadManager
      WLM_SYSTEM = SGE;
}
#
# Data directives
#
DATA { # Files to be transferred
      TRANSFER /var/tmp/blastSTAGE/nr.00.phr;
      TRANSFER /var/tmp/blastSTAGE/nr.00.pin;
      TRANSFER /var/tmp/blastSTAGE/nr.00.pnd;
      TRANSFER /var/tmp/blastSTAGE/nr.00.pni;
      TRANSFER /var/tmp/blastSTAGE/nr.00.ppd;
      TRANSFER /var/tmp/blastSTAGE/nr.00.ppi;
      TRANSFER /var/tmp/blastSTAGE/nr.00.psd;
      TRANSFER /var/tmp/blastSTAGE/nr.00.psi;
      TRANSFER /var/tmp/blastSTAGE/nr.00.psq;
      TRANSFER /var/tmp/blastSTAGE/nr.01.phr;
      TRANSFER /var/tmp/blastSTAGE/nr.01.pin;
      TRANSFER /var/tmp/blastSTAGE/nr.01.pnd;
      TRANSFER /var/tmp/blastSTAGE/nr.01.pni;
      TRANSFER /var/tmp/blastSTAGE/nr.01.ppd;
```

```
TRANSFER /var/tmp/blastSTAGE/nr.01.ppi;
TRANSFER /var/tmp/blastSTAGE/nr.01.psd;
TRANSFER /var/tmp/blastSTAGE/nr.01.psi;
TRANSFER /var/tmp/blastSTAGE/nr.01.psq;
TRANSFER /var/tmp/blastSTAGE/nr.pal;
}
EXECUTE {
    DO /wrg/home/csci/cjm/blast/blast_rep.sh parms=[1:20;1];
}
CLEANUP {
    PURGEALL;
}
```

7.2 *Modified SGE Script file*

```
#!/bin/bash
#$ -S /bin/sh -N blast -l h_rt=04:00:00,arch=lx24-amd64 -j y
#$ -o /gpfs/home/csci/cjm/output
#$ -pe shmem 4

echo Processors: $NSLOTS

source /wrg/profile.d//blast-2.2.14

WORKDIR=/gpfs/home/csci/cjm/
BLASTDB=/var/tmp/replic/SYNC/

time blastall -p blastp -a $NSLOTS -d nr -i
$WORKDIR/takifugu_cftr.fasta >
/var/tmp/blastout.$JOB_ID.$SGE_TASK_ID
```

As can be seen, modifications to the script file consist solely of changing the script to use the cached files (in /var/tmp/replic/SYNC/).