Optimising LAN access to grid enabled storage elements

G A Stewart[1], G A Cowan[2], B Dunne[1], A Elwell[1] and A P Millar[1]

[1] Department of Physics, University of Glasgow, Glasgow, UK
[2] Department of Physics, University of Edinburgh, Edinburgh, UK

E-mail: g.stewart@physics.gla.ac.uk

Abstract. When operational, the Large Hadron Collider experiments at CERN will collect tens of petabytes of physics data per year. The worldwide LHC computing grid (WLCG) will distribute this data to over two hundred Tier-1 and Tier-2 computing centres, enabling particle physicists around the globe to access the data for analysis.

Although different middleware solutions exist for effective management of storage systems at collaborating institutes, the patterns of access envisaged for Tier-2s fall into two distinct categories. The first involves bulk transfer of data between different Grid storage elements using protocols such as GridFTP. This data movement will principally involve *writing* ESD and AOD files into Tier-2 storage. Secondly, once datasets are stored at a Tier-2, physics analysis jobs will read the data from the local SE. Such jobs require a POSIX-like interface to the storage so that individual physics events can be extracted.

In this paper we consider the performance of POSIX-like access to files held in Disk Pool Manager (DPM) storage elements, a popular lightweight SRM storage manager from EGEE.

1. Introduction

When the Large Hadron Collider at CERN begins to run at luminosities sufficient for physics studies, it will produce around 15 petabytes of data a year. In order to analyse such a large quantity of information, the Worldwide LHC Computing Grid (WLCG) has been created. This is an international collaboration of physics laboratories and institutes around the world, encompasing 3 different grid infrastructures.

The UK's Grid for particle physics (GridPP) [1] started in 2001 with the aim of creating a computing grid that would meet the needs of particle physicists working on the next generation of particle physics experiments (i.e., the LHC). To meet this aim, participating institutes were organised into a set of Tier-2 centres according to their geographical location. ScotGrid [2] is one such distributed Tier-2 computing centre formed as a collaboration between the Universities of Durham, Edinburgh and Glasgow. The work we describe in this paper was undertaken at the University of Glasgow's grid computing cluster, which we describe below.

Current estimates suggest that the *maximum* rate with which a physics analysis job can read data via the root framework [3] is 10MB/s (and some experiment frameworks around root read considerably more slowly than this). We want to study the response of DPM to a large number of client requests to determine how well it scales and if it can maintain the 10MB/s access rate when running on production quality hardware that is currently operational on the WLCG grid.

Physics analysis code will use the POSIX-like LAN access protocols to read data, which for DPM is the Remote File I/O protocol (RFIO).

The structure of this paper is as follows. Section 2 describes the storage middleware technology that we use in this study. Section 3 discusses the storage, compute and networking hardware that is employed to perform the tests and compares it to that which would be used in production. Section 4 explains the reasoning behind our testing methodology. We present and interpret the results of this testing in Section 5 and conclude in Section 6.

2. Storage middleware

The Disk Pool Manager (DPM) [4], [5] is a storage middleware product created at CERN as part of the EGEE [6] project. It has been developed as a lightweight solution for disk storage management at Tier-2 institutes. *A priori*, there is no limitation on the amount of disk space that the DPM can handle, but it has no interface to a mass storage system.

2.1. Architecture

The DPM consists of the following components,

- DPM server (`dpm`): keeps track of all requests for file access.
- DPM name server (`dpnsdaemon`): handles the namespace for all files under the control of DPM.
- DPM RFIO server (`rfiod`): handles the transfers for the RFIO protocol (See section 2.2).
- DPM GridFTP server (`dpm.ftpd`): handles data transfers requesting use of the GridFTP protocol (See Section 2.2).
- SRM server (`srmv1`, `srmv2`, `srmv2.2`): receives the SRM requests, passing them on to the DPM server.

The protocols listed above will be described in the Section 2.2. Figure 1 shows how the components can be configured in an instance of DPM. Typically at a Tier-2 the server daemons (`dpm`, `dpns`, `srm`) are shared on one DPM *headnode*, with separate large disk servers actually storing and serving files, running `gsiftp` and `rfio` servers.

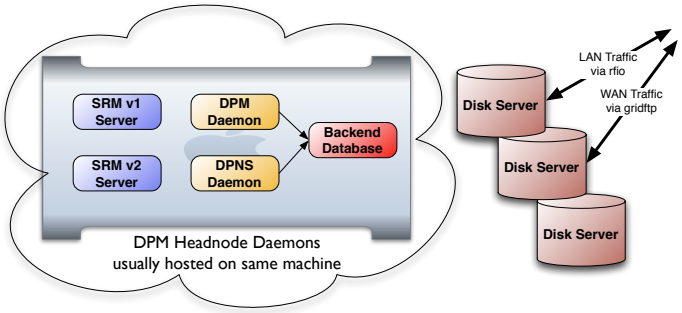

Figure 1. Shows one possible, and typical, configuration of the DPM server components.

2.2. Protocols

DPM currently uses two different protocols for data transfer:

- GridFTP: high performance streaming mode, typically used for wide area transfer of data files, e.g., movement of data from Tier-1 to Tier-2 storage.

- Remote File I/O (RFIO): GSI-enabled [8] protocol which provides POSIX [9] file operations, permitting byte-level access to files.

RFIO is the protocol that physics analysis code should use in order to read data stored within a DPM instance and is the protocol used to perform the tests in this paper. The RFIO library allows for the client to choose from four different modes of operation:

- `0`: Normal reading with one request to the server per read.
- `RFIO_READBUF`: An internal buffer is allocated within the client. Each call to the server fills this buffer and the user buffer is filled from the internal buffer. There is one server call per buffer fill.
- `RFIO_READAHEAD`: `RFIO_READBUF` is forced on and an internal buffer is allocated within the client. Then an initial call is sent to the server which pushes data to the client until end of file is reached, an error occurs or a new request comes from the client.
- `RFIO_STREAM` (V3): This read mode opens 2 connections between the client and server, one data socket and one control socket. This allows the overlap of disk and network operations. Data is pushed on the data socket until EOF is reached. Transfer is interrupted by sending a packet on the control socket.

The default RFIO buffer size is 128kB, but this can be configured by setting `RFIO IOBUFSIZE` in `/etc/shift.conf` on the client.

We will not use the SRM or GridFTP servers during these tests. (Optimising GridFTP transfers for Tier-2 centres has already been considered [7].)

2.2.1. Security As RFIO is GSI-enabled, clients using the protocol require to use an Certificate Authority signed X.509 Grid certificate. This encrypts and secures the rfio requests, although the data transport itself is not encrypted for performance reasons.

3. Hardware setup

3.1. DPM servers

The UKI-SCOTGRID-GLASGOW DPM server is composed of a single *headnode*, running the DPM, DPNS and SRM services of the DPM instance. This is a dual CPU Opteron 252 server with 8GB of RAM, running SL3.0.8 with a 2.4.21 kernel. The headnode's system disks are twin 147Gb Seagate SCSI drives, running linux software RAID. The 8 disk servers have identical motherboards and processors but run SLC4.5 with a 2.6.9 kernel. The disk servers have 24 disk Areca hardware PCI-X RAID controller. The data areas used for these tests consist of single RAID 6 arrays with 21 500GB Hitachi disks.

In the tests described below, unless otherwise stated, only 4 disk servers were utilised. See Figure 2.

3.2. Computing cluster

To facilitate the testing, we had use of the UKI-SCOTGRID-GLASGOW WLCG grid site [2]. The computing cluster is composed of 140 dual core, dual CPU Opteron 282 processing nodes with 8GB of RAM each. Being a production site, the compute cluster was typically processing user analysis and experimental Monte Carlo production jobs while our performance studies were ongoing. However, observation showed that jobs on the cluster were typically CPU bound, performing little I/O. As our test jobs are just the opposite (little CPU, I/O and network bound) tests were able to be performed while the cluster was still in production.[1]

[1] In fact this is quite reasonable, as in modern multicore SMP systems analysis jobs will share nodes with additional batch jobs, which will typically be CPU intensive.

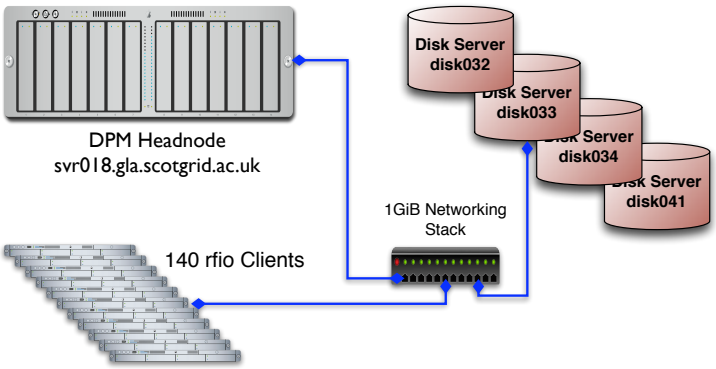

Figure 2. Shows the configuration of the UKI-SCOTGRID-GLASGOW DPM server in relation to the local computing cluster. Note that each host has a dedicated 1GiB connection to the switch stack, although, for clarity, not all of these are shown.

3.3. Networking

The LAN network of the cluster consists of a Nortel 5510 switch stack with each cluster node connected through a 1GiB ethernet connection. This limits the rate for each client to 1GiB, but more importantly limits the rate per-disk server to 1GiB. The internal bandwidth of the switch is 80GiB/s, which far exceeds any rate in this study.

4. Methodology

4.1. Phase space of interest

Analysis jobs will read data in storage elements, so we restrict our exploration to jobs which read data from DPM. We explore the effects of using different RFIO reading modes, changing the client's read block size and setting different RFIO buffer sizes.

We also examine the performance of the DPM system when the data the clients wish to read is found only on one disk server, across four servers or across nine servers. This will show how the read rates supportable by DPM systems scale to installations of different sizes.

4.2. RFIO client application

In order to explore the parameter space outlined above, we developed our own RFIO client application. Written in C, this application links against the RFIO client library for DPM (`libdpm`). The application was designed such that it can be used to simulate different types of file access pattern. We were interested in two basic patterns of access: in the first the client sequentially reads blocks from a file until the whole file is read; in the second the file is partially read by reading a block, then skipping some blocks, then reading again, etc. This last access pattern was used to simulate the file access of physics analysis code as the job jumps to different parts of the file when scanning for interesting physics events. Importantly, the client could be configured to set one of the RFIO options as described in Section 2.2.

4.3. Source data and client initialisation

As in general one expects that each analysis job is uncorrelated with the others, and so will be reading different data, 100 source data files of 1GiB size were seeded onto the DPM. Each client then read a single unique file. If this assumption were not valid, and several clients were reading the same data, then each disk server would have the opportunity to cache data from the file in question, possibly resulting in faster reads.

We also choose to stress the DPM server itself during our testing scenario by starting each of the clients within 1s. This should be considered a worst case scenario for the storage system, as in practice it is very unlikely that jobs will request opens in such close succession.

Thus, in general, our testing scenario is deliberately setup in order to stress the storage system; however, it is appropriate for establishing the performance limits of the DPM for the highly demanding LAN reading scenario demanded by LHC physics analysis.

5. Results

5.1. Client results

5.1.1. Complete File Reads Figure 3 shows the results for reading 1GiB files spread over 4 disk servers. As can be seen the DPM can handle approximately 10 file opens per second with little performance loss. After 10 clients file open times begin to degrade and for large numbers of clients opens can take a up to 16s.

Reading rate is 40-70MiB/s for a single client, and rises rapidly for small numbers of clients. As the number of clients increases the total rate plateaus off to a little below the maximum expected rate of 500MiB[2]. (N.B. we define data rate as `BYTES_READ`/(`Open Time + Read Time`).) However, clearly the client buffered modes, READBUF, READAHEAD and STREAMING perform considerably better than NORMAL mode. STREAMING and READAHEAD perform slightly better than READBUF as both aggressively read data from the server.

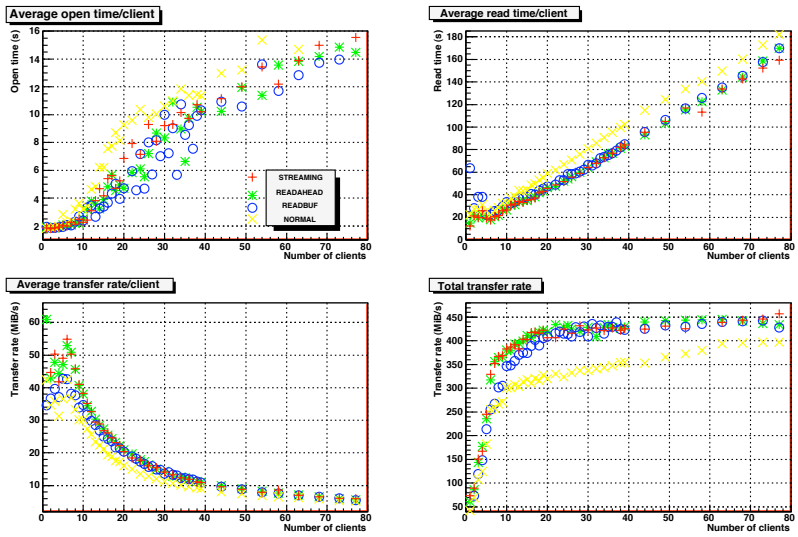

Figure 3. Complete sequential reading of 1GiB files stored in the DPM from the compute cluster, for varying read modes and client number. The block size was 1MiB and the data was spread over 4 disk servers. Error bars are smaller than the plotted symbols.

5.1.2. Partial file reads Results for partial file reads are shown in figure 4. Note that STREAMING mode, optimised for reading whole files, is not compatible with skipping sections of the file. In this case each client reads 1MiB from the source file, then skips 9MiB (simulating reading 10% of the events in, say, an AOD file). As expected the results for opening files are very similar to figure 3 – the `rfio_open()` call is exactly same as the previous case.

[2] The results we present are the client data rate, whereas the theoretical network bandwidth must also include IP and TCP overheads.

Read rates, as expected, are considerably lower than for complete reads, as the disk server has to skip large portions of the file, repositioning the reading heads. Maximum rates are, in this case, only 20MiB/s. In stark contrast the case when complete files are read, NORMAL mode performs better than the buffer reading modes. In particular the advantage of NORMAL mode increases as large numbers of clients are reached. This can be understood as the buffered modes read data which is not in fact needed by the client. This data is thus discarded, but has loaded both the disk server and the network, reducing performance.

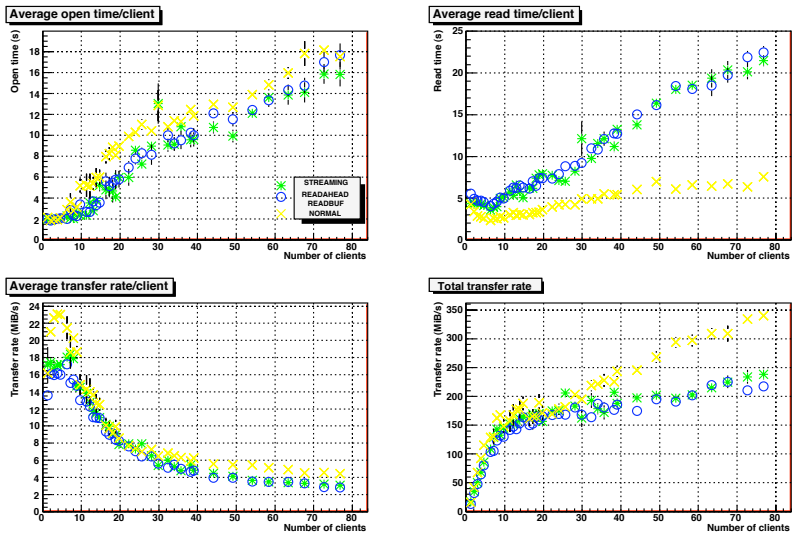

Figure 4. Results for partial (10%) reading of 1GiB files stored in the DPM from the compute cluster, for varying read modes and client number. The block size was 1MiB and the data was spread over 4 disk servers. Error bars are shown as black lines.

5.1.3. Read blocksize Figure 5 shows the results of varying the read block size used by clients when completely reading 1GiB files. Block sizes of 10kB and above clearly improve the read for the buffering clients; although for non-buffered NORMAL mode clients a read block size of 100kB is necessary to obtain good performance. Once a rate of >60MiB/s is achieved further increases in block size have almost no impact.

As before, STREAMING mode gives the best rate for a complete file read, with READAHEAD also performing well.

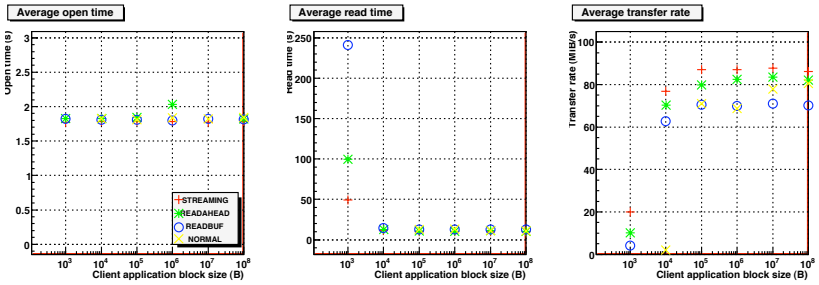

Figure 5. Variation in the file open time, read time and transfer rate for changing values of the read block size. In this case only a single client was reading. File size was 1GiB.

5.1.4. RFIO IOBUFSIZE The value of the internal API buffer used by RFIO clients in the default READBUF mode is set by the system administrator in `/etc/shift.conf`, rather than by clients.

Figures 6 shows the results of varying the value of RFIO IOBUFSIZE for READBUF mode, both for the cases when the whole file is read, or only 10% of the file is read.

For complete file reads it can be seen that the value of IOBUFSIZE has a negligible effect on data reading rates. In our case, as the client does no processing of the data, is can ingest the data as quickly as RFIO can deliver it.

For partial file reads increasing the value of IOBUFSIZE clearly hurts the overall rate considerably. This is caused by the internal buffer being filled before the client actually requests data. In the case of skipping through a file the client in fact does not require this data and so network and I/O bandwidth has been consumed needlessly.

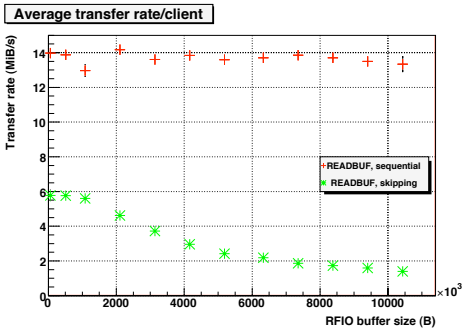

Figure 6. Reading 1GiB files stored in the DPM from 30 client nodes on the compute cluster. Whole file reads (red) and partial file reads (green) are shown, plotted against the value of RFIO IOBUFSIZE.

5.2. Server

5.2.1. RFIO open times Figure 7 clearly shows how the average open time increases from 2 seconds for a small number of clients (< 23) up to around 11 seconds for a larger number of clients (≥ 23). This results is dependent on the hardware used during the testing.

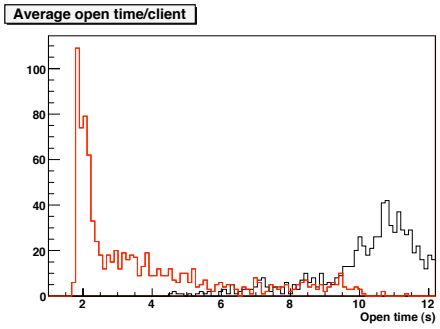

Figure 7. Histograms of the average open times measured during all testing runs. The left hand (red) histogram shows the case where the number of clients is < 23 while the right hand (black) histogram shows that for ≥ 23.

5.2.2. File open error rate While server performance clearly degrades when many clients simultaneously attempt to open files, most files are opened successfully, as can be seen in Figure 8. This is a substantial improvement over earlier versions of DPM, which could not support more than about 40 opens per second[12]. Even at very low numbers of clients, open errors can be observed. This indicates that client code should sensibly implement a backoff and retry strategy, to avoid a low level of file open errors from affecting job success rates.

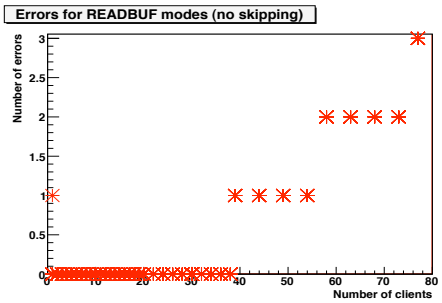

Figure 8. File open error rates, when multiple clients attempt to open files.

5.2.3. Server load and bottlenecks Figure 9 (left) shows typical CPU load on a single disk server during a test in which increasing numbers of clients make complete reads of different files held on the disk server. For small numbers of clients there is almost no I/O wait, and the network quickly saturates (see Figure 3). When the number of clients grows large then, even though files are being read in their entirety, head repositioning to service these multiple requests means that I/O wait grows - although at no time does the total transfer rate over the network drop.

Likewise, Figure 9 (right) shows the typical CPU load during a test when clients make partial reads of files. In this case I/O wait dominates very quickly. This is backed up by Figure 4, where it is seen that the network limit is not reached.

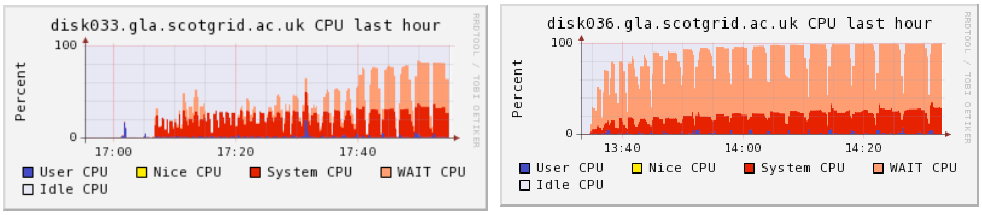

Figure 9. Left: Server CPU load measured by ganglia while undergoing multiple client streaming read tests - the number of simultaneous clients increases left to right. **Right**: Server CPU load measured by ganglia while undergoing multiple client skipping read tests - the number of simultaneous clients increases left to right.

5.2.4. Overall site performance: multiple disk servers In Figure 10 the effect of increasing the number of disk servers available to clients is shown, plotted against client number. For 1 and 4 disk servers, scaling up to a clear network bandwidth limit is rapid, occurring when each disk server is servicing about 5 clients. For the case of 9 disk servers maximum rates are not reached until more than 60 clients are reading. However, the rates for O(20) clients are clearly very good, with rates in excess of 800MiB/s being seen. In all cases data was read in 1MiB blocks.

It is possible that limitations on other parts of the site infrastructure, e.g., the switch stack become an issue. However the performance of DPM up to 9 servers and about 100TB of disk remains excellent.

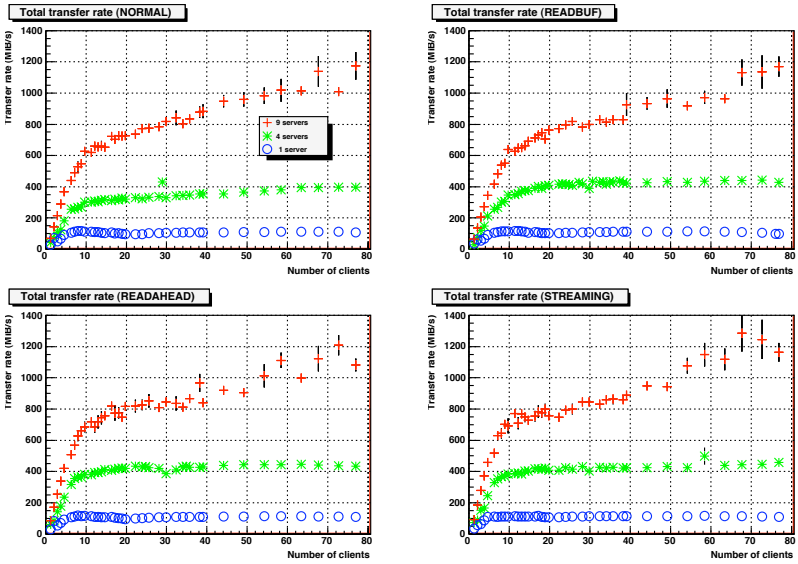

Figure 10. Total transfer rate for DPM clients with 1, 4 and 9 disk servers vs increasing client number. Clients were reading the entire file.

In Figure 11 a similar plot is shown, but for partial file reads (as usual, reading one 1MiB block and then skipping 9). In this case, rates are much lower, however for large numbers of clients scaling is linear with the disk server number, indicating that DPM will be able to scale to meet a site's needs as the hardware levels are appropriately increased.

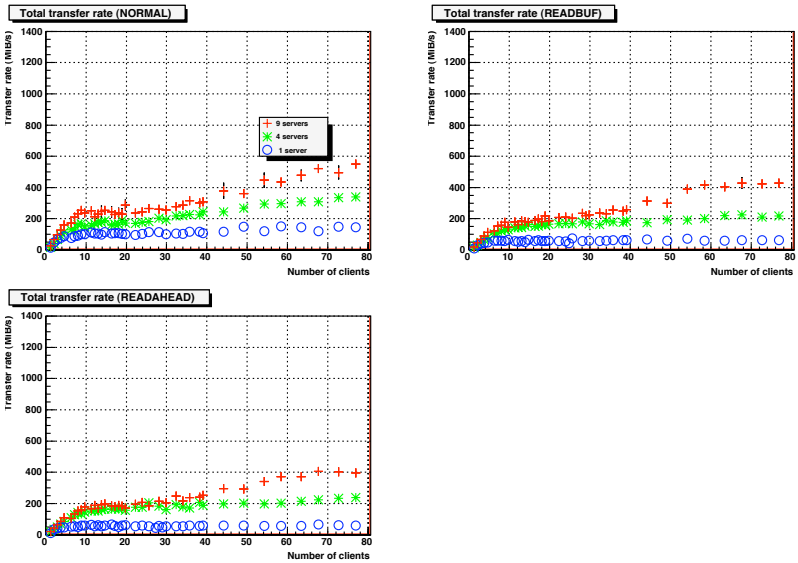

Figure 11. Total transfer rate for DPM clients with 1, 4 and 9 disk servers vs increasing client number. Clients were reading 10% of the file.

6. Conclusions

We have thoroughly tested the performance of the Disk Pool Manager storage element software under a wide range of client configurations, using both realistic client numbers and credible Tier-2 storage hardware.

Our conclusion, for RFIO clients, is that for complete data reads a buffered read mode should be used, with STREAM and READAHEAD performing best. However, for partial file reads NORMAL mode is optimal, as it prevents unwanted data being uselessly transferred to the client.

Clients will get better performance reading data in large blocks. Small reads carry a significant performance penalty. In this respect having clients which understand the data structures they are trying to read (e.g., root reading TTreeCache structures[13]) will help to optimise access.

The value of RFIO IOBUFSIZE should be left at its default setting of 128kB. In particular setting too high a value will penalise clients using the default READBUF mode to make partial reads of a file.

On the hardware used, DPM performed very well, although ocassional failures to open files can be seen. Client code would be best to adopt a retry strategy in the case of such errors.

Testing indicated that DPM will perform well on modern hardware at least up to the 100TB scale, with no particular performance problems coming to light.

Further testing scenarios will involve running more realistic experiment codes and testing scanrios where a Tier-2 would both be ingesting WAN data and exporting analysis data to LAN clients.

Acknowledgments

The authors would like to thank all those who helped in the preparation of this work. Funding for this work was provided by STFC/PPARC via the GridPP project. G Stewart is funded by the EGEE project.

References

[1] The GridPP Project. *http://www.gridpp.ac.uk/*

[2] The ScotGrid Project. *http://www.scotgrid.ac.uk/*

[3] Rene Brun and Fons Rademakers. *ROOT – An object oriented data analysis framework* AIHENP'96, Lasuanne, 1996.

[4] Disk Pool Manager. *http://twiki.cern.ch/twiki/bin/view/LCG/DpmAdminGuide*

[5] Lana Abadie *et al. DPM Status and Next Steps* CHEP07, Victoria, 2007. http://indico.cern.ch/contributionDisplay.py?contribId=389&sessionId=21&confId=3580

[6] Enabling Grids for E-scienceE. *http://www.eu-egee.org/*

[7] Greig Cowan, Graeme Stewart, Jamie Ferguson. *Optimisation of Grid Enabled Storage at Small Sites.* Proceedings of 6th UK eScience All Hands Meeting, Paper Number 664, 2006.

[8] Globus GSI Security. *http://www.globus.org/toolkit/docs/4.0/security/*

[9] POSIX Standard. *http://standards.ieee.org/regauth/posix/index.html*

[10] dCache Mass Storage System. *http://www.dcache.org/*

[11] Joint Academic Network. *http://www.ja.net/*

[12] Graeme Stewart and Greig Cowan. *rfio tests of DPM at Glasgow* WLCG Workshop, CERN January 2007. http://indico.cern.ch/contributionDisplay.py?contribId=101&sessionId=13&confId=3738.

[13] Rene Brun, Leandro Franco, Fons Rademakers. *Efficient Access to Remote Data in High Energy Physics* CHEP07, Victoria, 2007. http://indico.cern.ch/contributionDisplay.py?contribId=284&sessionId=31&confId=3580.