PROCEEDINGS OF SCIENCE

# An Analysis of Data Access Methods Within WLCG

**Shaun de Witt**[1]
*Science and Technology Facilities Council*
*Rutherford Appleton Laboratory, Harwell, Didcot, OX11 0QX, United Kingdom*
*E-mail:* shaun.de-witt@stfc.ac.uk

**Andrew Lahiff**
*Science and Technology Facilities Council*
*Rutherford Appleton Laboratory, Harwell, Didcot, OX11 0QX, United Kingdom*
*E-mail:* andrew.lahiff@stfc.ac.uk

With WLCG experiments making increasing use of the xrootd framework and changes to the tiered model away from the classic MONARC architecture to a mesh, new options for accessing data for analysis have become possible. Users are now able to both copy input data from the storage system to the compute nodes and read the input file locally (local read), or open the input data objects directly on the storage system (remote read) and only read the parts of the file which are necessary. However, currently no detailed analysis exists on the performance of each method under different operational scenarios and thus it is not clear under all conditions which method will give the best performance and how each impacts the load on the underlying storage system. In this paper we use specially crafted clients to look at the performance across a number of different storage systems used within WLCG for a number of different scenarios where the amount of same data is read from different sites. Specifically we present the results of reading different number of events from the file and the impact on the storage system and network with differing number of clients using the two access methods mentioned.

*International Symposium on Grids and Clouds (ISGC) 2015*
*15 -20 March 2015,*
*Academia Sinica, Taipei, Taiwan*

[1]Speaker



PoS(ISGC2015)020

## 1.Introduction

As well reported elsewhere, over the last few years many experiments WLCG have moved from the the Tiered MONARC model [1] to a mesh model [2]. In addition, WLCG has standardised on fewer protocols; *XrootD* [3], *gridFTP* and *WebDAV*. XrootD and WebDAV allow for 'redirection' which allows jobs to efficiently discover the location of a file even if it is not available locally. This is discussed further in section 1.2 below. Redirection should improve job efficiency significantly in cases where some of the input files within a dataset are not available locally, since there is no need to scheduled a transfer between the site hosting the data and the site running the job.

In this work, we present the results of investigating the relative efficiency of transferring input files to local storage and directly accessing it from there compared to reading directly from remote sites. We look at the effect of reading different percentages of the file, either directly either sequentially or using random access, and also look at the effect of different 'event sizes' on reading. Since the RAL Tier 1 currently does not support WebDAV, this work concentrates on XrootD and gridFTP. In addition, in this paper we remove the need to use redirection by placing files at the test sites and reading directly from the site.

While this work is based on the WLCG use cases, it is of relevance to a number of other science communities where data is stored in complex, multi-dimensional files or where data is stored in, for example, daily files but analysis is performed on only a subset of the data. Two examples of these are meteorology and seismology. In the former case, model data is typically stored in three dimensional data sets with axes of latitude, longitude and either isobaric levels or height above the geoid. Often, further analysis is done on the mesoscale (over a specific geographic region) or at specific atmospheric levels. In the latter case, at least in Europe, data is stored from several thousand sensors in daily files (single file for large groups of sensors), but typical analysis is only on a small fraction of this around a seismic event. In both cases, transferring a whole set of data files to an HPC node uses a large amount of bandwidth and the transfer time is a significant fraction of the analysis time. Being able to reduce this by only reading the part of the file of interest is clearly advantageous.

This paper consists of four sections. Section 1, the current section, introduces the work and explains the changes within WLCG which make this work relevant. The second section describes the test set up and test cases which were used in the analysis. The third section presents the results of testing in detail and tries to draw some inferences from this. The final section attempts to generate a conclusion from the results presented in the previous section.

### 1.1 Changes to the WLCG Model

In the early days of WLCG, data management was built on a tiered model (the MONARC model) with three distinct tiers; the Tier 0 at CERN, a small number of large Tier 1 sites providing both computing facilities and persistent storage, and a larger number of Tier 2 sites which were associated with a Tier 1 each of which provided substantial computing power and limited, non-persistent storage (Figure 1). Data gathered from the detectors would be stored at the Tier 0 and one or two replicas (depending on experiment needs and resources) would be sent to the Tier 1. The principle of this model is that jobs would be sent to sites geographically close

to the data. So a job sent to a Tier 2 site would either already have the data set locally or, more normally, the job would transfer the data set from the associated Tier 1 to the Tier 2 where the job was to run, and would remove the data set from the Tier 2 at the completion of the job. In addition, Monte-Carlo simulations run at a Tier 2 would send there data to the corresponding Tier 1 for long term persistence. This model worked for many cases and would still work today if Tier 1 and Tier 0 sites were 100% reliable, never suffering data loss or temporary inaccessibility to data due to diskserver or network problems. Of course, this 100% reliability never happens and some fraction of jobs would fail due to missing input data. To overcome this, experiments would need to remove the inaccessible copy from the Tier 1, schedule a transfer from either another Tier 1 or the Tier 0 (in either case possibly requiring a staging from tape), and then resubmitting the job once the transfer was complete. While for small numbers of files, this may be deemed an acceptable overhead, if a Tier 1 site is offline for any significant period of time, for instance during scheduled upgrades or due to network problems, the associated Tier 2's are limited to only performing Monte-Carlo simulations and then only until there local storage in exhausted. This clearly causes a significant loss in computing resources for experiments.

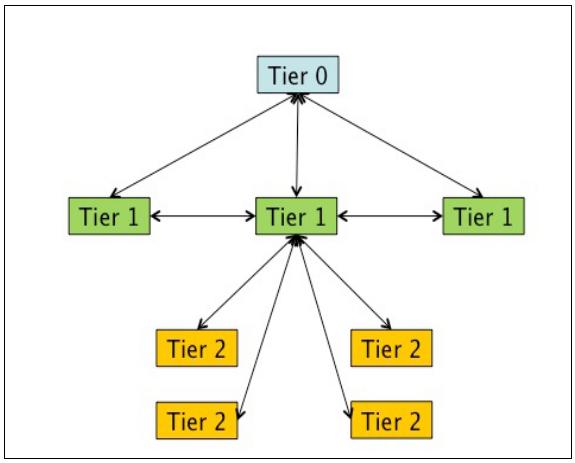


*Figure 1: WLCG MONARC Model showing Data Flows*

The new mesh model (Figure 2), when combined with a redirecting protocol such as *HTTP* or *XrootD* (q.v. below) eliminates the dependence of a Tier 2 on a specific, or indeed any, Tier 1. It allows sites to access data from any other site within an xroot based federation and allows the redirection mechanism to pick the most efficient site to use. The file, rather than being copied between sites and analysis performed on the local copy, can now be opened and read from the remote site directly. There is still an association between a Tier 2 and Tier 1 in that results from simulations are normally written to the associated Tier 1, but the strict hierarchy is broken for analysis and allows jobs to be scheduled anywhere within the federation. Currently there are two implementations of this within WLCG; AAA for CMS [4] and FAX for ATLAS [5]. The only significant difference is a minor change is overall deployment of redirector nodes which is not discussed in this document. This change means Tier 2's can operate fully to the needs of the experiment regardless of the state of the associated Tier 1.

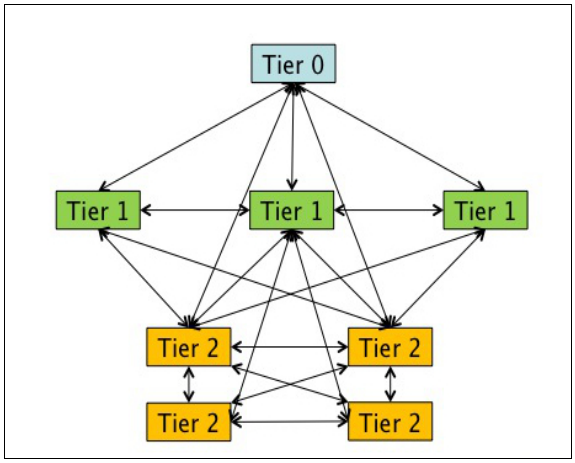


*Figure 2: WLCG Mesh Model showing Data Flows*

### 1.2 The Adoption of XrootD

In the early days of WLCG, most of the experiments used a mixture of protocols. *GridFTP* was standard for wide area transfers between sites, but within sites specific protocols were often used depending on the underlying storage technology; sites running dCache used *dcap*, CASTOR and DPM sites used *rfio* (and incompatible versions of rfio to complicate matters further), sites running Lustre and GPFS used the *file* protocol. Clearly this meant experiments had to deal with the complexity of dealing with different protocols when running jobs at specific sites. The exception to this was ALICE which focussed on XrootD from the start of operations[6]. The problem with the legacy protocols were that they had limited functionality and almost no interoperability. After review, it was decided to consolidate on three protocols which would be available at all sites; gridFTP and WebDAV for scheduled wide area transfers and XrootD for analysis work.

The adoption of XrootD in particular allows the use of redirection. Data no longer needs to be copied to sites for analysis, but the data can be accessed directly in-situ wherever it resides within the WLCG data grid. In practice, most data is expected to be at the site where the job is run, purely to improve efficiency, but it is no longer the case that the data **must** be at the site. The redirection mechanism employed by XrootD is very similar to that employed by the global DNS system in that is is based on a hierarchical model (Figure 3). A client contacts a local redirector node at the site and if the data is at the site, then it is made immediately accessible. However, if the data is not locally available, the redirector redirects the request to a higher level redirector which will check all its child nodes. This is repeated until a copy of the file is located at which point a direct connection between the client and the storage system is established, or no node has responded regarding the file availability. In this way, a client can read the file exactly as if it was present locally.

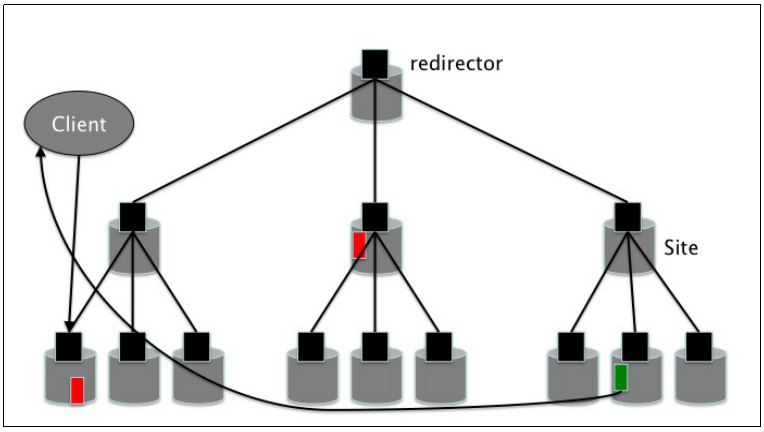


*Figure 3: Schematic of Redirection Hierarchy of XRootD*

## 2. Test Methodology and Setup

As stated in the introduction, this work looks at the relative benefits of accessing data directly at a remote site compared with copying it to local disk and then analysing it directly. In this case, all tests were performed with a single (virtual) client machine with 2 Xeon E5-2650L processors, 4GB of RAM, a shared 10GB ethernet connection and NFS attached storage system.

A specific test harness was developed for this work. The script would open a file at a specified site using a specified protocol. The test harness does not perform any specific analysis of the file, merely reading the file in one of two modes. The first mode read some percentage of the file sequentially starting at an offset of zero. The second mode reads a percentage of the file in a specified chunk size (representing the size of an 'event' within the data file), with seeks to random offsets between each read. So in this latter case, for a 1GB file and requesting to read 10% of the file (100MB) in 1MB chunks would result in 100 separate read operations. Since the offset used is random, it is possible that the same section of data is read multiple times, but this is not relevant for the purposes of these tests.

The rational behind these differing tests is that for normal WLCG operations, events are read and analysed sequentially within a file, with event sizes varying from a few 10's of kiloBytes to a few megaBytes. While reading the file sequentially, most disk storage systems attempt to improve performance by tuning a number of system parameters such as the TCP window size and the read-ahead of the disk devices. In particular, the read ahead can improve read performance significantly by caching the file in memory before the next read call. The ability to seek through the file will, to some extent, eliminate the effect of read-ahead on disks and thus allow a more systemmatic comparison of performance.

Three tests sites were selected for this testing. More sites would be necessary to complete the comparison, but these three were selected since they cover a reasonable range of distances. These sites are shown in table 1 below. It is important to note that the tests at STFC and Lancaster were performed on test systems and thus were not under any load during testing, while tests performed at ASGC were against their production instance and thus subject to more temporal variability.

| Site | Storage System | Distance(km) | RTT (ms) |
|---|---|---|---|
| RAL | CASTOR | 0 | 0.5 |
| Lancaster University | DPM | 350 | 26 |
| ASGC | DPM | 10300 | 300 |

*Table 1: Test Sites*

Fifty test files, each of 1GB generated from /dev/urandom, were created and each site was pre-seeded with this test data. The tests were then repeated for all 50 files and averaged to obtained a better comparison. In the case of STFC, the legacy *rfio* protocol was also tested for comparison. When data was copied to local disk it was read using the *file* protocol, and the transfer was performed using either the standard *xrdcp* command provided by xroot, or *rfcp* in the case where the legacy protocol was used.

## 3. Results

In this section, each subsection will show the results of testing at each site, and include some discussions around interpreting the results from that site. For interest, we also include results of testing using the CEPH cluster currently under development at STFC. It should be noted that as the sites at STFC and Lancaster University were test systems, and were not under load, the results represent a somewhat idealised case. In reality, many other parameters will affect performance such as the load in the disk servers, the number of concurrent transfers, the specific network cards being used and the setting of any disk tuning parameters and the RAID configuration employed (if any). In a 'real-world' situation, it is might be anticipated that direct reads from the storage system at a specific read size may give slightly better results since they will put less stress on the disk servers in question.

### 3.1 Rutherford Appleton Laboratory (RAL)

RAL is a large Tier 1 site based near Oxford in the UK. It supports all four major LHC experiments providing multi petabytes of storage and more than 10k compute cores. Tests performed at this site were against an unloaded test and development instance of the storage system. In this case, all of the test data resided on a single disk server and thus any variation in the response from different hardware is eliminated in these tests.

Figure 4 shows the results of testing the legacy protocol *rfio*. In this case, for brevity, we only tested two specific cases; reading the file sequentially from the storage and copying to local disk using *rfcp* and then reading the *file* using the file protocol. In this case, the behaviour is very obvious and in line with what could be anticipated; the time to read a percentage of the file increases linearly with the percentage read, while copying to local (NFS mounted) disk provides a nearly constant performance regardless of the percentage of the file read. It is clear that copying file to local disk and then accessing it using the file protocol is dominated by the transfer time between the stroage system and the local node, and writing the file to the local node. Reading the file directly from the storage system is more efficient if reading less than about 50% of the file. Below this, the effects of the time to write to local disk dominates.

Above 50% then the network overhead starts to dominate and in this case it is clearly more efficient to copy the file and analyse locally. In the case where data is copied to the local NFS mounted storage and read directly from there, the time to read the file is negligible and the time is dominated by the combination of network transfer and time to write to the local disk.

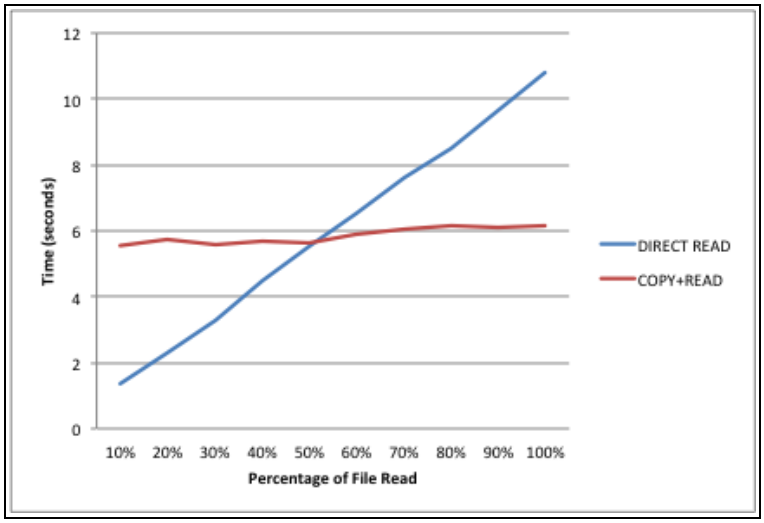


*Figure 4: Comparison of time to Read a File Using rfio*

In comparison, figures 5 and 6 show the equivalent using *xrootd* (copy to local storage using *xrdcp* as previously mentioned). Figure 5 indicates that in the scenarios tested, when copying to local storage, again the time is dominated by the transfer time and time to write to the locally mounted NFS disk. The actual transfer is approximately 20% faster using *xrdcp* compared to *rfcp*, which may simply be the effect that *xrootd* itself is a caching protocol with some read-ahead while *rfio* is not. However, testing indicates that directly reading from the storage system, at least for CASTOR at RAL, is always better than trying to copy and read.

In addition, comparing the two figures, the impact of different 'event sizes' becomes very evident. For smaller read sizes, despite reading the same volume of data, the increased number of seeks (increasing by three orders of magnitude) has a massive effect, with the effect becoming worse with the number of events read. Reading 50% of the file with a read size of 1MB took about 2.25 seconds, while for 1kB read sizes, the time to read the same volume was more about 148 seconds. This additional overhead will be an accumulation of various factors including disk seeking, uncached reads, and network handshaking. We do not attempt to assess the dominant effect in this paper.

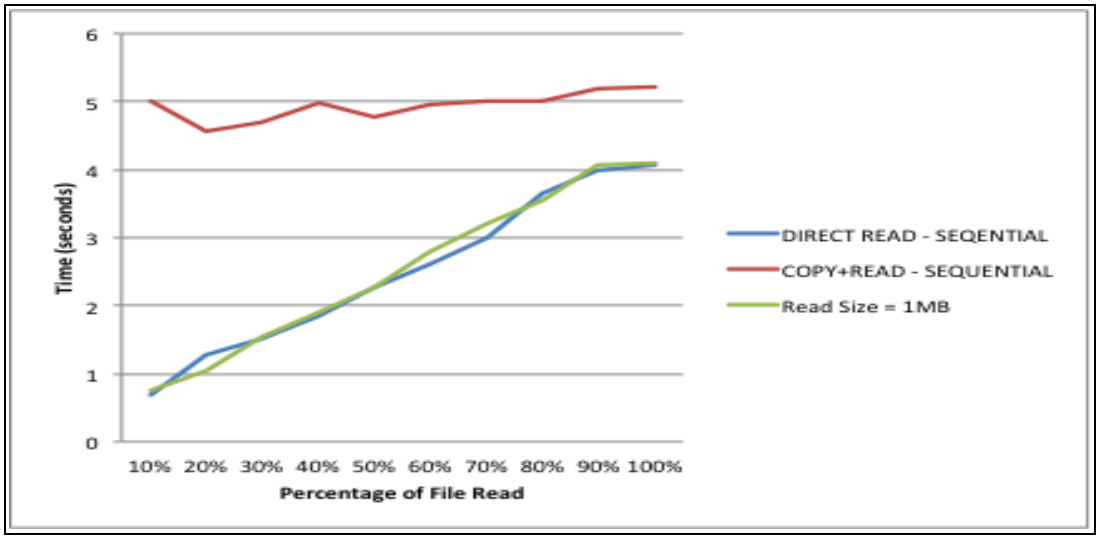


*Figure 5: Reading Data From RAL Tier 1 Using XrootD (read and blue lines are for a single read as a fraction of the file size, the green line indicates random reads of 1MB chunks until specified ercentage of file is read)*

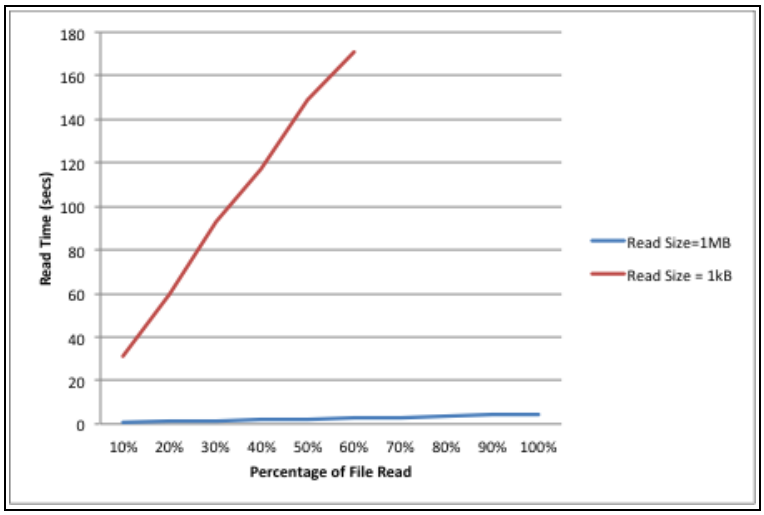


*Figure 6: Reading Data from RAL Tier 1 Showing the Effect of Different Read Sizes (reading chunks of the specified read size until the required percentage of the file size is achieved)*

### 3.1.1 Testing ROOT Files

While not directly presented here, these tests were also repeated using experiment files in ROOT format. Two test cases were investigated, each representing a legitimate WLCG use case. These were:

- Copy the file to local disk and read events sequentially from there using the file protocol
- Open the file directly on the storage system using XrootD and read the events sequentially

In both cases, the file is opened using the ROOT framework [7]. Under testing at STFC, reading directly from the storage system was considerably quicker than in the tests

mentioned above. Reading the whole file (~1GB, ~1000 events) took less than 2 seconds when directly accessed from the storage system. Copying to local disk and reading from there took about the same time as the previous test (~5 seconds), demonstrating the dominance of the time to write the file to local disk.

### 3.2 University of Lancaster

The Tier 2 Site at the University of Lancaster is one of the Tier 2s associates with the RAL Tier 1, and thus the results presented here are of high significance. Within WLCG it supports the ATLAS experiment. Like the tests run at RAL, the storage system used at Lancaster was a test instance. Unlike the previous case, the test data was spread over a number of servers which, although of similar specification, is likely to introduce some variability in results.

In this case, while the legacy protocols can not be tested since it does not support direct read over a wide area network. However, the use of *gridFTP* is reported. For this protocol we have only looked at its use in the 'copy and read' scenario, where the file is copied to local storage on the client and then read with the *file* protocol.

Figure 7 shows the results of testing. As expected, whichever test scenario is considered, reading data from Lancaster is slower than reading from the RAL Tier 1. Since the network RTT is 50 times that of RAL, this is expected. Indeed for the case of copying the file using xrdcp, the time to transfer and read the file from local disk is about 30 times slower than was the case when performing the same test at RAL and thus it can be said that the XrootD implementation at Lancaster is quite efficient. The efficiency of transferring using gridFTP is also evident. Note that gridFTP transfers were performed using the client tool globus-url-copy with default parameters so was working in single stream mode with no parallelism. Further gains may have been possible by tuning the parallelism and the tcp block sizes, but this is outside the scope of the current work. One result which is currently not understood is that reading the file in 1MB read sizes is more efficient than reading the file sequentially, despite the increased number of seek operations performed. This may be due to the disk server optimisations previously discussed.

One result not shown on this figure is the effect of the different read sizes. For a 1kB read size, reading 10% of the file took more than 1900 seconds – 114 times longer than using using a 1MB read size and 15 times more than copying the file to the client and reading locally in 1kB read sizes.

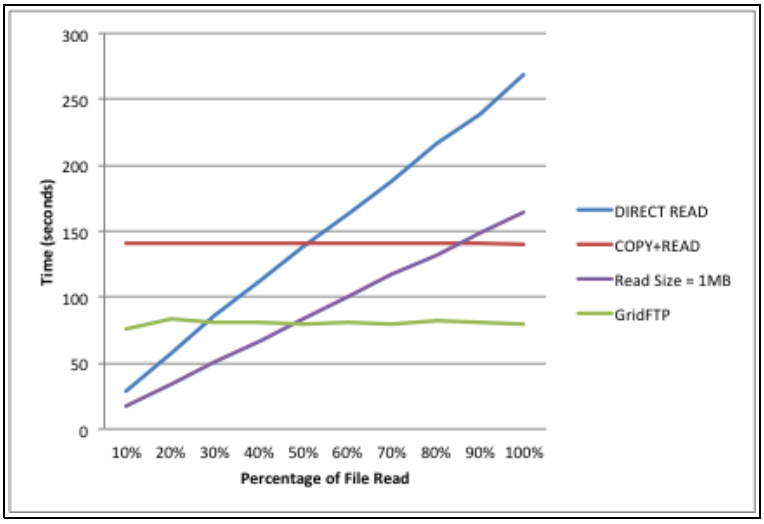


*Figure 7:Reading Data from Lancaster Tier 2*

### 3.3 Academia Sinica Grid Computing Centre (ASGC)

ASGC is a Tier-1 site primarily serving the ATLAS experiment. In this paper, it represents one of the more extreme sites likely to be used in any xroot federation. While some sites in Australasia are more distant geographically, they are not Tier 1 sites and the likelihood of being redirected from the UK to one of these sites is very small. Unlike the previously discussed sites, testing here was performed against a production instance. The impact of this is that results are more variable depending on current production needs at the site. Since the tests were performed shortly after an upgrade to the storage system, there was a reasonably significant level of activity to allow ATLAS to catch up the backlog of jobs.

The results are shown in figure 8. As can be seen, reading directly using xrootd from ASGC is problematic, with the problem worsening as the amount of data read increases. What this does not show is that there were in fact many timeouts and attempts to reconnect occurring during these sequential reads. However, reading the data in 1MB blocks significantly improves performance. However, it should be noted these tests were performed 12 hours after the sequential read tests and thus the storage system may have been under less load. Over this distance, use of *xrdcp* and *gridFTP* makes little difference. Recall that in both these cases the entire file is transferred to local storage at the client and only some percentage of the data is read by the test suite. Thus the apparent decrease in performance for low read percentages for gridFTP is likely an artefact of the storage system load. An attempt to read the data using a 1kB read size proved difficult; a single attempt to read 10% of the test file directly from ASGC was not complete after 15 hours, and thus is clearly impractical for any running job.

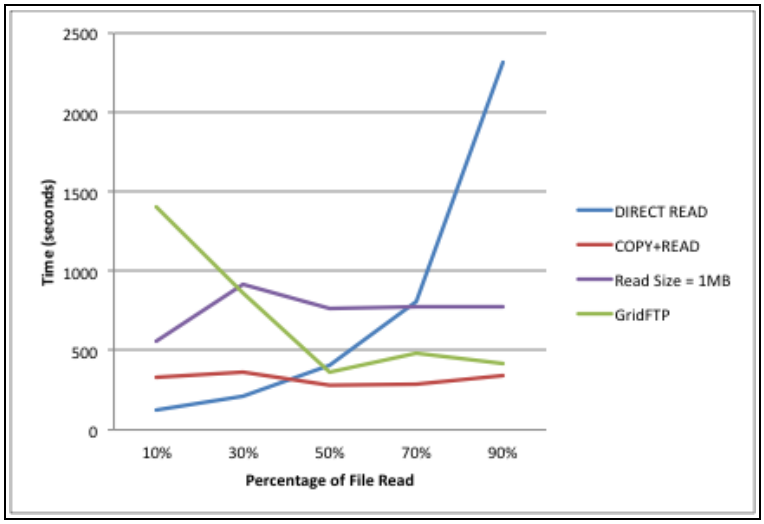


*Figure 8: Reading Data from ASGC Tier 1*

### 3.4 CEPH

As a final test, we repeated the testing using the xroot protocol on the CEPH development cluster at RAL. It should be noted that this cluster, as the name suggests, is currently under development. It was running the Giant release and made use of the new rados-striper library and erasure encoding (configured with 16 data chunks and 2 parity chunks – mimicking the RAID6 currently deployed at RAL in production). The hardware was not optimised for use with CEPH and the xroot interface is a pre-production version supplied by CERN.

The results are shown in figure 9. As can be seen, the performance is very linear and predictable, but not currently highly performant. However, unlike other systems tested where a direct connection between disk server and client is established, the xrootd redirector acts as a gateway and all data flows through this. Currently the redirector runs on a virtual machine using a shared 1Gb network card. Hence, while the performance is relatively poor, it is still reasonable given the current state of the system. However, it is clear that significantly more testing and optimisation is required before this can be put in to production.

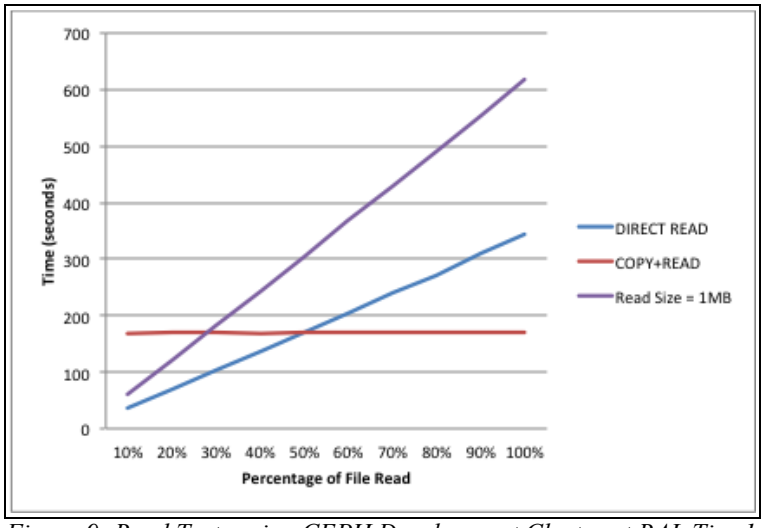


*Figure 9: Read Tests using CEPH Development Cluster at RAL Tier 1*

## 4. Conclusion

The results presented here do not give a clear answer to the original question on whether it is always better to read directly from a remote site or copy data to local disk and access the file from there. Based on the results presented, the answer to this question will depend on a number of factors including the details of the storage system, the distance between the client and the data source, the network connectivity, the optimisations applied to the disk servers, etc. However, it is possible to draw a few inferences can be made based on the evidence presented.

If data is organised in small events (< 500kB approximately) then it is always worth copying the data to local disk. The overheads associated with reading small events, particularly if they are read in a random order and require *seek* operations between reads, makes the time to read even 10% of the file highly inefficient compared to copying the file to local disk on the client. In the WLCG use case, this is quite relevant since event sizes are quite small. However, this is largely mitigated by the use of the ROOT framework which has been highly tuned for data access in the WLCG use case, demonstrating the significant advantages in the use of this analysis framework. For other disciplines such as meteorology and seismology, however, this may be more relevant, particularly for seismology where events last of the order of minutes but data is held in daily files.

While it is not clear based on these results whether it is better perform a copy-and-read operation or a direct read, if the former is chosen then it appears that in general *gridFTP* is a better protocol to use than *xrdcp*. The results here were performed without trying to perform any optimisation on the Globus client. In this case, gsiFTP performed better in one case and no worse in the second case tested. It can reasonably be expected that with tuning, this performance can be further enhanced. However, it is also likely that there will be no single tuning which will optimise performance across all sites used with WLCG.

To draw any further conclusions, it would be necessary to repeat these tests against many more sites at different distances and using different storage systems. While tools like perfSonar can give indications of network latency, the specific implementation of xrootd and the performance of the underlying storage system at sites would dominate the choice of access for any experiment. In addition to performing these tests, details of any tuning would also be needed on order to draw firm conclusions.

## 5. Acknowledgements

The authors would like to acknowledge the co-operation of Matt Droidge (University of Lancaster) and Felix Lee (ASGC) for there assistance and permitting the use of the storage systems at their respective sites. In addition, we would like to extend our thanks to Sebastien Ponce (CERN) for supplying the CEPH plugin for XrootD prior to its official release and support in addressing issues discovered quickly.

## References

[1] MONARC – Models of Network Analysis at Regional Centres for LHC Experiments, http://monarc.web.cern.ch/MONARC/.

[2] S. Campana, et. al., 2012 *J.Phys.: Conf. Ser.* **396** 032319 (doi:10.1088/1742-6596/396/3/032019)

[3] L.Bauerdick, et.al., 2012 *J.Phys.: Conf. Ser.* **396** 042009 (doi:10.1088/1742-6596/396/4/042009)

[4] K. Bloom, 2014 *J.Phys.: Conf. Ser.* **513** 042005 (doi:10.1088/1742-6596/513/4/042005)

[5] R. Gardner, et. al., 2014 *J.Phys.: Conf. Ser.* **513** 042009 (doi:10.1088/1742-6596/513/4/042049)

[6] L. Betez, http://aliceinfo.cern.ch/Documents/TDR/Computing.html

[7] M.Ballintijn, R. Brun, F. Rademakers, G. Roland, 2003 arXiv:physics/0306110