# Site in a box: Improving the Tier 3 experience

# Site in a box: Improving the Tier 3 experience

**J M Dost**[1], **E M Fajardo**[1], **T R Jones**[2], **T Martin**[1], **A Tadel**[1], **M Tadel**[1], **F Würthwein**[1]

[1] University of California, San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA
[2] Monash University, Wellington Rd, Clayton VIC 3800, Australia

E-mail: `jdost@ucsd.edu`

**Abstract.** The Pacific Research Platform is an initiative to interconnect Science DMZs between campuses across the West Coast of the United States over a 100 gbps network. The LHC @ UC is a proof of concept pilot project that focuses on interconnecting 6 University of California campuses. It is spearheaded by computing specialists from the UCSD Tier 2 Center in collaboration with the San Diego Supercomputer Center. A machine has been shipped to each campus extending the concept of the Data Transfer Node to a cluster in a box that is fully integrated into the local compute, storage, and networking infrastructure. The node contains a full HTCondor batch system, and also an XRootD proxy cache. User jobs routed to the DTN can run on 40 additional slots provided by the machine, and can also flock to a common GlideinWMS pilot pool, which sends jobs out to any of the participating UCs, as well as to Comet, the new supercomputer at SDSC. In addition, a common XRootD federation has been created to interconnect the UCs and give the ability to arbitrarily export data from the home university, to make it available wherever the jobs run. The UC level federation also statically redirects to either the ATLAS FAX or CMS AAA federation respectively to make globally published datasets available, depending on end user VO membership credentials. XRootD read operations from the federation transfer through the nearest DTN proxy cache located at the site where the jobs run. This reduces wide area network overhead for subsequent accesses, and improves overall read performance. Details on the technical implementation, challenges faced and overcome in setting up the infrastructure, and an analysis of usage patterns and system scalability will be presented.

## 1. Introduction

Grid computing is an integral tool the LHC experiments depend on in order to carry out their research [1]. It is standard practice to spread scientific workflows over many disparate computing resources across Academic institutions and Federal labs spanning multiple countries and continents. Overlay systems have been created to facilitate running scientific workflows over the Grid. LHC Virtual Organizations such as CMS and ATLAS have developed CRAB [2] and PANDA [3] respectively, both built on top of the pilot submission paradigm. Pilot based submission systems allow physicists to temporarily claim remote computing resources they don't own for a limited duration of time in order to run their jobs. While grid computing is a necessity for LHC High Energy Physics, it also presents unique challenges for both end users and resource providers.

The main challenge grid users face is that they can't just submit any application they normally would run in their local batch computing cluster from within their local institution and expect it to work. The application must be first made "grid-ready." This typically includes ensuring input data is either pre-staged at the remote site, or making it available over the WAN by other means such as GridFTP or XRootD [4]. Another issue is ensuring the application can support a variety of operating systems (typically Enterprise Linux variants such as EL6 or EL7), and ensuring the relevant libraries the application was built against are available as well, either by shipping them along with the application, or providing them through CVMFS. Some workflows simply don't fit well within the constraints of the major workflow tools (CRAB, PANDA), so it is still useful to submit these separately in the local batch system. This is typically the "last-mile" type workflow that aggregates and filters the results of previous runs from the respective pilot systems, into smaller datasets for specialized analysis.

Analogous to the task of the end user, the major challenge of being a grid resource provider is ensuring their batch cluster is grid-ready, so that end users of the grid can successfully claim and run on their machines. In order to make a site grid-ready, the system administrators must correctly install, configure, and maintain the grid middleware software stack, which is provided by either the OSG [5] or EGI [6] respectively depending on which grid the site belongs to. One example service the grid software stack provides is the CE gatekeeper software, which is the public facing grid service that allows users into the site. Other important packages are the required security packages for GSI authentication with the users, which includes CA certificate bundles, tools for maintaining CRLs, creating GSI host certificates, and mapping authorized users to respective local user accounts (GUMS, ARGUS). Important configuration and services also need to be available from the worker nodes, including CVMFS mounts, and standard client tools for common tasks like data transfer. For all of the grid middleware to function correctly, site admins also need to ensure their networking is properly configured. This includes ensuring firewall rules are correct, e.g. that public facing services like the CE have open inbound ports, and all workers have open ports for outbound connections. Also, smaller sites should ensure that a NAT is installed and properly configured if needed. In addition to proper grid software management, resource providers need to correctly set up batch system priorities, for example, rules that enable their local users to run with higher priority than external users that come in from the grid.

Running a grid site is not easy, and it is particularly difficult for Tier 3 sites. Tier 3 sites are usually smaller in size, and their main disadvantage is not having dedicated staff to handle the grid infrastructure. A typical Tier 3 might have a general IT department that provides local batch systems to various research groups, but that won't likely have detailed knowledge on LHC specific grid middleware or know about specific VO requirements. This leaves the burden of maintaining the grid and VO specific software to a small team of physics professors and graduate students who would rather be running physics analysis than be systems administrators. In a joint effort between the Pacific Research Platform (PRP) [7], SDSC, OSG, ATLAS, and CMS, we have designed a prototype known as the "Tier 3 in a Box," which is a model that offloads the effort to maintain a Tier 3 site away from the local IT staff and physics researchers from a small site, and puts it in the hands of dedicated partner staff from a geographically nearby Tier 2 computing center. The working prototype, known as the UCLHC project, has been implemented across six University of California campuses – five Tier 3 sites and the UCSD Tier 2. The following Tier 3 sites are: UC Davis (CMS), UC Santa Barbara (CMS), UC Riverside (CMS), UC Santa Cruz (ATLAS), and UC Irvine (ATLAS). Each UC has a Tier 3 in a Box setup that is centrally maintained by admins at the CMS Tier 2 Center at UC San Diego. Each UC is interconnected via the PRP, which provides 100gbps connectivity between the science DMZs of each campus.

## 2. Tier 3 in a Box architecture

We provide a physical machine to each participating UC. The machine contains a dual socket motherboard providing 40 hyperthreads, 128 Gigabytes of ram, 48 Terabytes of disk (12 disks, each 4TB) configured in RAID6 with an XFS file system, and 2 x 10 Gbps network cards. For the UCLHC pilot program, the The PRP allows us to take full advantage of the 10gbps network cards given the 100gbps network infrastructure. All software is installed, configured, and maintained from a central Puppet instance running at UCSD. The goal of the Tier 3 in a Box machine is to provide everything needed to handle both the transferring of data over the grid, and also the ability to send user jobs to the grid to run on external resources. The primary software services we install on the machine consist of a complete HTCondor batch system [8], an XRootD server, redirector, and proxy cache [9], CVMFS, and an optional Squid proxy cache depending on site needs. It is effectively an enhanced Data Transfer Node (DTN), with additional job routing capabilities. Throughout the rest of the discussion the Tier 3 in a Box will be referred to as the DTN. Care is taken to ensure the system is flexible enough to integrate seamlessly with a variety of setups that may exist at each Tier 3. All of the maintenance required in keeping the services up to date and running fall under the responsibility of the UCSD T2 team. This frees up the burden on Tier 3 IT admins and scientists from needing a deep knowledge of LHC grid computing in order to facilitate their science. All they are required to maintain is the hardware. In addition, we expect a local physicist as contact who educates the group on the capabilities the DTN offers.

The two major components of the system – job submission and data transfer mechanisms are discussed in more detail in the following sections.

### *2.1. Job submission routing*

A full HTCondor batch system is installed on the DTN. This includes an HTCondor collector, negotiator, schedd, and startd. User jobs can be sent to the DTN to run directly on the provided 40 cores. In the standard Tier 3 in a Box setup, we do not allow direct login from the users. Instead, users continue to submit jobs from their existing submit host. By making minor changes to the submit file, they can submit to the DTN using HTCondor-C rather than to the site batch system. HTCondor-C is essentially schedd to schedd submission. Using HTCondor-C allows end users to continue to use the submit host they already have as an interactive node, and the DTN node is able to remain loginless for improved security. Submitting via HTCondor-C to the internal 40 cores of the DTN is only one of several new resource targets the Tier 3 in a Box provides. A set of domains have been defined so that users can specify the scope of where their jobs can run. Users specify the domain in their submit file. The domains are as follows: `local` sends jobs to the DTN, `site_local` sends jobs to the site batch system, `sdsc` sends jobs to Comet at the San Diego Supecomputer Center, and `uc` sends jobs to any of the other participating UC campuses. Domains can be combined so that user jobs can run over multiple resource types simultaneously.

The `sdsc` and `uc` domains in particular are external grid resources made available by the Tier 3 in a Box. This is made possible by a glideinWMS [10] pilot pool, which is also hosted and maintained by admins at UC San Diego. User jobs with the relevant domain flags are sent to the DTN in the usual way through HTCondor-C. Once they arrive on the DTN schedd, the pilot pool Glidein Frontend initiates pilot submission requests to the relevant grid sites, and then the user jobs use the HTCondor Flocking mechanism to flock from the DTN into the pilot pool once pilots land on and reserve the requested resources. Beyond the currently supported submit domains, the Tier 3 in a Box is capable of allowing job submission to any external OSG site, but this feature has not been enabled yet.

### *2.2. Data transfer mechanisms*

The DTN is capable of sending and receiving data between jobs and sites over the XRootD protocol. An XRootD redirector is installed on the DTN, and the redirectors on each DTN are also connected to a regional redirector which is hosted on the same machine at UCSD as the glideinWMS Frontend. This creates an XRootD federation at the UC level. There is also an XRootD data server on each DTN, which allows sites to export files to the UC federation directly from disk space on the DTN itself, and optionally any portion of the site's local data storage, depending on the user's needs. This makes locally stored data sets available for user jobs running at every external grid site. In addition, the regional redirector statically redirects to the XRootD federation belonging to the respective VO: FAX [4] for ATLAS, and AAA [11] for CMS. This makes all datasets that are published in the VO level federation also available to the user jobs, no matter where they run.

The third XRootD component installed in the DTNs is the Caching Proxy. Care is taken to ensure any XRootD accesses from jobs at a given site are passed through the local DTN proxy cache, whether the data source is the XRootD server in the DTN of the user's home institution, or a server from the VO level XRootD federation. Any files that have been accessed for the first time will be stored on disk in the DTN proxy cache. Subsequent accesses thus only need to fetch the file from the cache, which in turn keeps data transfer traffic local to the site in which the job is running. This is done for two reasons. First, it protects the external data servers in the respective federations from being overloaded with too many client requests over the wide area network. Second, it can potentially increase user job throughput, since subsequent accesses of cached files stay within the local area network.

## 3. System scalability

To exercise the system, we performed a series of scale tests, focusing in particular on data transfer over XRootD. The goal was to first to understand what kind of client I/O load it takes to saturate the 10 gbps network cards, and in the process, to discover and fix any limitations that may be imposed by enabling the caching proxy. In order to set up the test, first a sample of 100 files was created and hosted at UCSC. The idea is that n jobs would be set up to run simultaneously at UCI, and each job would pick one of the 100 files at random and do an xrdcp to `/dev/null`. The idea is that over time, more and more files should be chosen that were accessed previously, and the job should thus access the file from the local UCI cache rather than from UCSC. As a control we first ran the test without the cache, to determine an optimal number for n. We ran three initial runs with 1, 2, and then 4 clients in parallel. As seen in Figure 1, we were able to get close to full bandwidth running 4 clients (1 GB/s = 8 gbps). Next we ran a test with 10 clients in parallel, and as expected we were able to saturate the network interface at 10 gbps as shown in Figure 2. Since bandwidth scales linearly without degradation, we can conclude that the 12 disks in RAID 6 are not a bottleneck for hosting the files. Note there was negligible writing happening simultaneously on the server side when these tests were performed.

Next, we enabled the xrootd cache at UCI. Surprisingly, we hit a major bottleneck

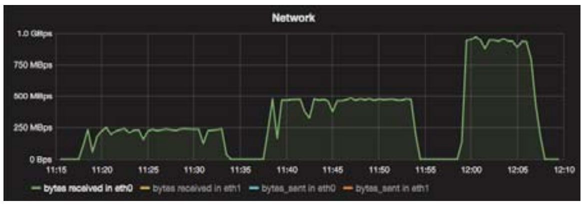


**Figure 1.** UCI DTN (caching disabled) inbound network (GB/s) while reading files from UCSC; trials shown running 1, 2, and 4 parallel clients

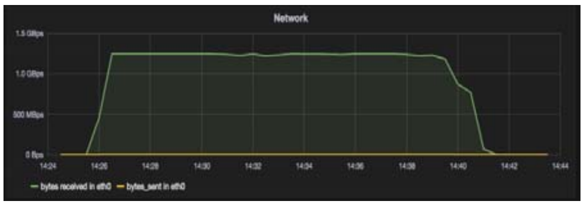


**Figure 2.** UCI DTN (caching disabled) inbound network (GB/s) while reading files from UCSC; trial shown running 10 parallel clients

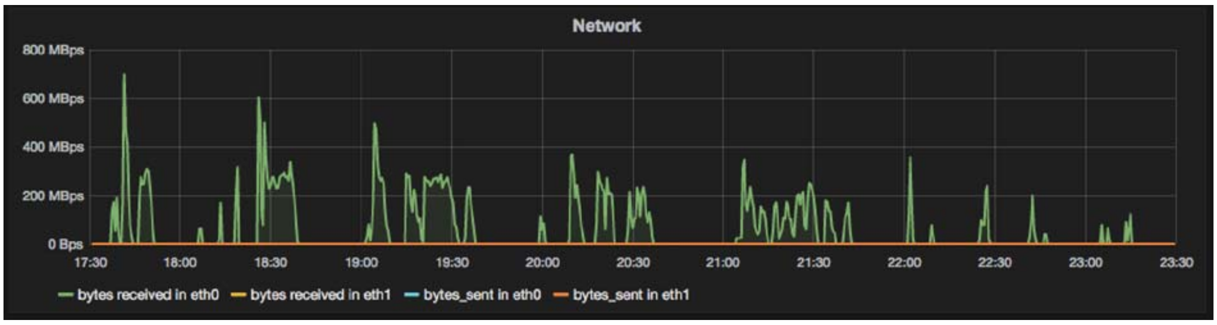


**Figure 3.** UCI DTN (caching enabled) inbound network (GB/s) while reading files from UCSC; default RAID stripe cache size of 256

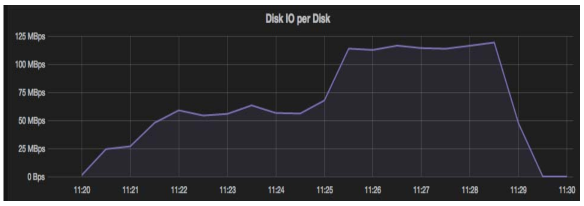


**Figure 4.** UCI DTN disk write (GB/s) during dd test; default RAID stripe cache size of 256

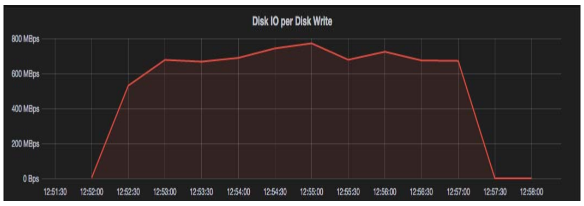
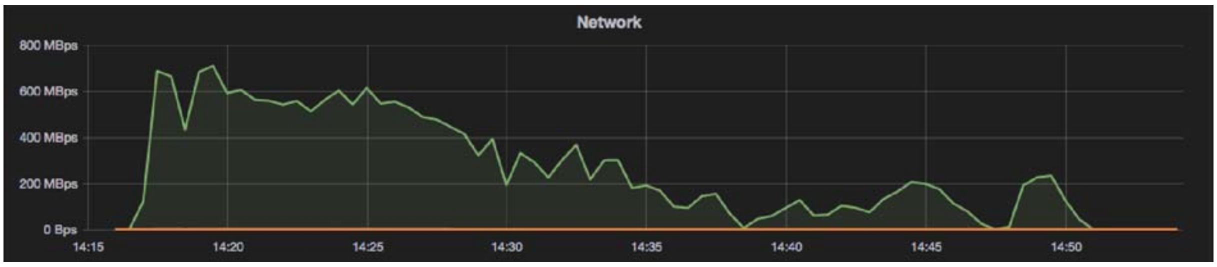


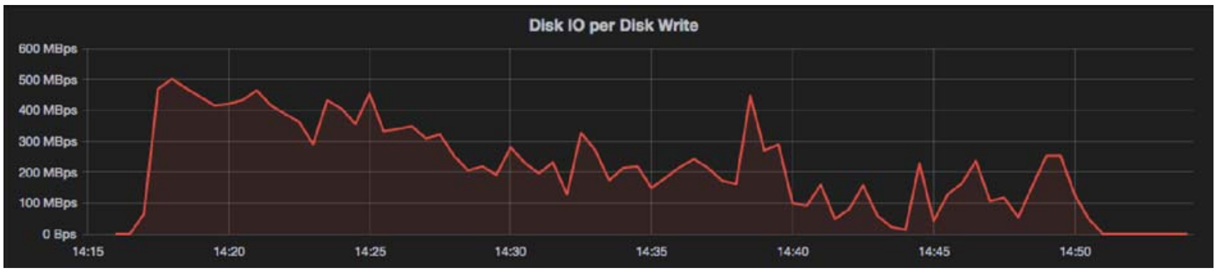
**Figure 5.** UCI DTN disk write (GB/s) during dd test; RAID stripe cache size increased to 32768

initially. Except for occasional peaks, on average we could not download files faster than 300 MB/s or 2.4 gbps from UCSC. Worse, there were large periods of time where our I/O went to zero. Figure 3 illustrates this. In our investigation we decided to run various tests with the dd tool. Writing 10 files simultaneously revealed that on average, our RAID 6 speeds couldn't exceed 60 MB/s. This is shown in Figure 4. The part of the plot where it went up to 115 MB/s was due to an fsync command run after all the clients finish. The write speed of the RAID 6 turned out to be the culprit responsible for choking our system. We discovered that by increasing `/sys/block/md0/md/stripe_cache_size` from the default size of 256 pages to the the maximum 32768 pages, we were able to remove the bottleneck. As shown in Figure 5, after the change, our write performance averages at 700 MB/s, which is more than a factor of 10 increase in performance.

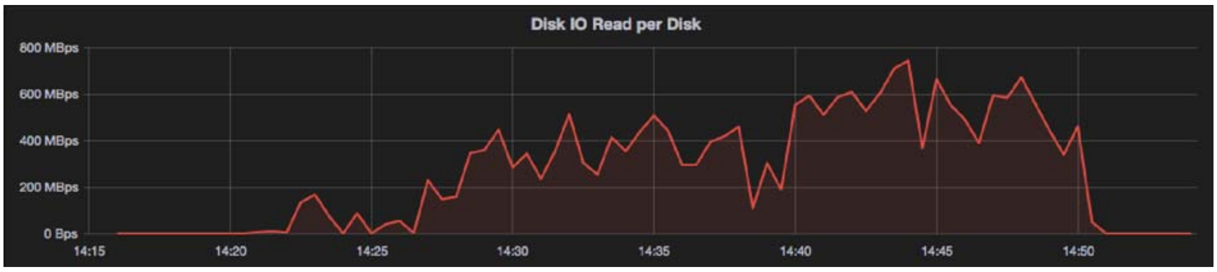
Figure 6 shows the test with caching enabled, repeated after the RAID tweak. As expected, Figure 6(a) shows high network utilization initially when the files were downloaded over the cache. The downloads started around 640MB/s. Over time, the network input went down, and the files were read off disk out of the cache (Figure 6(c)). Notice also that whether served from the network, or off disk, the read speed did not appear to exceed 700 MB/s, even though our tests without cache got up to 1.3 GB/s. This is because the incoming network is bound by the cache write speed, and also because read performance in RAID 6 degrades when there are simultaneous reads and writes (Figures 6(c) and 6(b)) happening. Had we let the tests continue until we were sure every file was cached, we would have expected to see the read speed from the cache eventually increase back up to 1.3GB/s since the cache would have no more files to write. Obviously, caching is of little use in the case of an xrdcp application of files within PRP given the excellent network bandwidth within the PRP. However, for access of data from the East Coast, Europe, or elsewhere in the world, the latency hiding that the cache provides is still important for typical applications in ATLAS and CMS. In addition, Tier-3 sites with less than 10Gbps WAN connectivity will benefit from the cache even within the west coast. The



(a) inbound network (GB/s)



(b) disk write (GB/s)



(c) disk read (GB/s)

**Figure 6.** UCI DTN (caching enabled) statistics while reading files from UCSC; RAID stripe cache size increased to 32768

namespace to be cached can be configured in the caching software. Different configurations can be deployed at each site in principle. In practice, this has not been done yet.

## 4. Challenges faced

In addition to working out how to properly scale XRootD, there were numerous obstacles we had to overcome while setting up the Tier 3 in a Box service to the various UCs. Perhaps the most common one was that a typical Tier 3 does not have the public IP space to assign to their worker nodes. So usually the head node doubles up as a NAT to provide outbound network for the workers. This complicates the DTN setup because it requires we set up dual network interfaces, one connected to the local area network for direct communication with the cluster, and the other to the wide area network, necessary to talk to the glideinWMS pilot pool and UC XRootD Federation. Care must be taken to ensure the HTCondor daemons are correctly configured on the node to listen and talk over the correct interface.

Another challenge is that the PRP network interconnects science DMZs between sites. Generally campus science DMZs have strict security rules about what services can be run on them, so we had to frequently coordinate with campus networking experts to ensure all the correct ports were open on the network level. We also had to ensure that the correct routes were set up so that each DTN could access all of the other DTN boxes at the participating UCs over

PRP, as well as the UCSD glideinWMS pool and UC XRootD federations, and also the FAX and AAA federations.

Finally, different unique setups were required depending on the batch system used at each Tier 3. HTCondor batch systems made the setup the most trivial. However, not all UC Tier 3 sites have CE services, so as a work around we had to set up Bosco [12] installations, which are effectively remote submission directly through an ssh login that the admins provide us on the submit hosts. For the one UC that does not use HTCondor, we had to invert the setup of the DTN. Rather than have it be loginless, we made it a new HTCondor submit host, and set it up to route into the local PBS batch cluster using Bosco. We trained the users to submit through the DTN using HTCondor, rather than direct PBS submission. This allows us to provide the same flexibility as the standard Tier 3 in a Box, but at the expense of having to create logins. We coordinated with the site admins so they are responsible for login management on the DTN.

## 5. Conclusions

The Tier 3 in a Box solution strengthens the infrastructure of a small site in two ways. First, it enables users to run beyond their local institution into external sites using HTCondor and glideinWMS. Second, the dynamic nature of the XRootD Federation allows users to access data seamlessly as if the jobs were running in their home institution. Beyond extending the capabilities of the small site, since the Tier 3 in a Box infrastructure is run by Tier 2 grid specialists, the sites benefit from these new functionalities without requiring new knowledge or in-house personnel to manage the services. In addition, the Tier 2 grid admins who run the service in turn are able to offer their knowledge and expertise to ensure the Tier 3 is functioning optimally, both for the local users, and also for external grid users.

**Acknowledgments**

We gratefully acknowledge the support of the US National Science Foundation under Grants No. PHY-1120138 (CMS), and ACI-1541349 (PRP) which supported the work in this paper.

## References

[1] Cairols J F S 2008 The grid computing face with the start of the lhc experiments *Proceedings of the 2008 The Second International Conference on Advanced Engineering Computing and Applications in Sciences* ADVCOMP '08 (Washington, DC, USA: IEEE Computer Society) pp 98–103 ISBN 978-0-7695-3369-8 URL `http://dx.doi.org/10.1109/ADVCOMP.2008.46`

[2] Spiga D *et al.* 2007 *Lect. Notes Comput. Sci.* **4873** 580–586

[3] Megino F B *et al.* 2015 *Procedia Computer Science* **66** 439 – 447 ISSN 1877-0509 4th International Young Scientist Conference on Computational Science URL `http://www.sciencedirect.com/science/article/pii/S1877050915033992`

[4] Bauerdick L *et al.* 2012 *Journal of Physics: Conference Series* **396** 042009 URL `http://stacks.iop.org/1742-6596/396/i=4/a=042009`

[5] Pordes R *et al.* 2007 *Journal of Physics: Conference Series* **78** 012057 URL `http://stacks.iop.org/1742-6596/78/i=1/a=012057`

[6] Bird I 2011 Computing for the large hadron collider vol 61 (Annual Reviews) pp 99–118 URL `http://www.annualreviews.org/doi/abs/10.1146/annurev-nucl-102010-130059`

[7] Pacific Research Platform `http://pacificresearchplatform.org/` Accessed: 2017-02-20

[8] Thain D, Tannenbaum T and Livny M 2005 *Concurrency - Practice and Experience* **17** 323–356

[9] Bauerdick L A T *et al.* 2014 *Journal of Physics: Conference Series* **513** 042044 URL `http://stacks.iop.org/1742-6596/513/i=4/a=042044`

[10] Sfiligoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Wurthwein F 2009 The pilot way to grid resources using glideinwms *2009 WRI World Congress on Computer Science and Information Engineering* vol 2 pp 428–432

[11] Bloom K *et al.* 2015 Any data, any time, anywhere: Global data access for science *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)* pp 85–91

[12] Weitzel D, Sfiligoi I, Bockelman B, Frey J, Wuerthwein F, Fraser D and Swanson D 2014 *Journal of Physics: Conference Series* **513** 032105 URL `http://stacks.iop.org/1742-6596/513/i=3/a=032105`