Scaling Up a CMS Tier-3 Site with Campus Resources and a 100 Gb/s Network Connection: What Could Go Wrong?

Matthias Wolf, Anna Woodard, Wenzhao Li, Kenyi Hurtado Anampa, Benjamin Tovar, Paul Brenner, Kevin Lannon, Mike Hildreth, Douglas Thain

University of Notre Dame, Notre Dame, IN 46556, USA

E-mail: `{mwolf3|awoodard}@nd.edu`

Abstract. The University of Notre Dame (ND) CMS group operates a modest-sized Tier-3 site suitable for local, final-stage analysis of CMS data. However, through the ND Center for Research Computing (CRC), Notre Dame researchers have opportunistic access to roughly 25k CPU cores of computing and a 100 Gb/s WAN network link. To understand the limits of what might be possible in this scenario, we undertook to use these resources for a wide range of CMS computing tasks from user analysis through large-scale Monte Carlo production (including both detector simulation and data reconstruction.) We will discuss the challenges inherent in effectively utilizing CRC resources for these tasks and the solutions deployed to overcome them.

1. Introduction

Within the Worldwide LHC Computing Grid (WLCG) [1] tier structure, Tier-3 sites represent the local computing resources that individual university groups use to perform the final stages of data analysis. The typical Tier-3 site is composed of an interactive login server that runs a batch system giving users access to a dedicated cluster of computing resources as well as some amount of storage. Optionally, a site might include components like an Open Science Grid (OSG) [2] Compute Element (CE) or Storage Element (SE). The University of Notre Dame (ND) operates a moderately sized site of this type, with dedicated resources of approximately 1000 CPU cores and approximately 700 TB of disk operated in an HDFS cluster with a replication factor of two, yielding approximately 350 TB usable capacity. The ND Tier-3 site includes an OSG CE and SE. The ND Tier-3 provides a Squid server for HTTP caching for FronTier [3] database queries and CERN Virtual Machine File System (CVMFS) [4] access. ND also operates a local XRootD [5] redirector to provide access to ND's Tier-3 storage across the network.

The ND Tier-3 infrastructure is deployed as part of ND's Center for Research Computing (CRC). The CRC serves ND's the computational infrastructure needs of nearly all ND's research endeavors from psychology and economics to biology and physics. The CRC administers over 25,000 compute cores on approximately 2,000 servers. Roughly one-third of these resources are available to any member of the ND research community on a fair-share basis. The remaining two-thirds of the resources are owned by specific faculty research groups. On nearly all resources, two resource schedulers run simultaneously: UGE and HTCondor. (The ND Tier-3 is an exception, using only HTCondor and not UGE.) UGE provides access to the shared resources for anyone in the ND community, and faculty owners can also use it to access their dedicated systems. However, all resources (both shared and faculty-owned) run the HTCondor scheduler as well. Whenever a resource—whether shared or faculty-owned—is idle in UGE, HTCondor is started, making the resource available to any CRC user on an opportunistic basis. As soon as a user, such as a member of the research group that owns the resource, schedules a job on it via UGE, any work running through HTCondor is immediately evicted and the resource leaves the HTCondor pool until it becomes idle again. The scale of resources available via HTCondor varies depending on the UGE load, but is typically in the range of 1,000 to 10,000 CPU cores.

Using the Lobster workflow management tool [6, 7], the ND CMS group has managed to augment their dedicated Tier-3 resources with additional CRC resources accessed opportunistically through HTCondor. Lobster is designed to allow CMS workloads to run on non-dedicated resources. Lobster uses Parrot [8] to distribute the CMS and OSG software to the opportunistic resources and leverages Work Queue [9, 10] to schedule tasks to the opportunistic resources. When needed, input data is made available on the opportunistic resources via XRootD, using either ND's local XRootD redirector or the CMS AAA Data Federation [11]. The opportunistic resources rely on the dedicated ND Tier-3 infrastructure for any other services, such as Squid caching. The configuration of resources used is depicted in Figure 1.

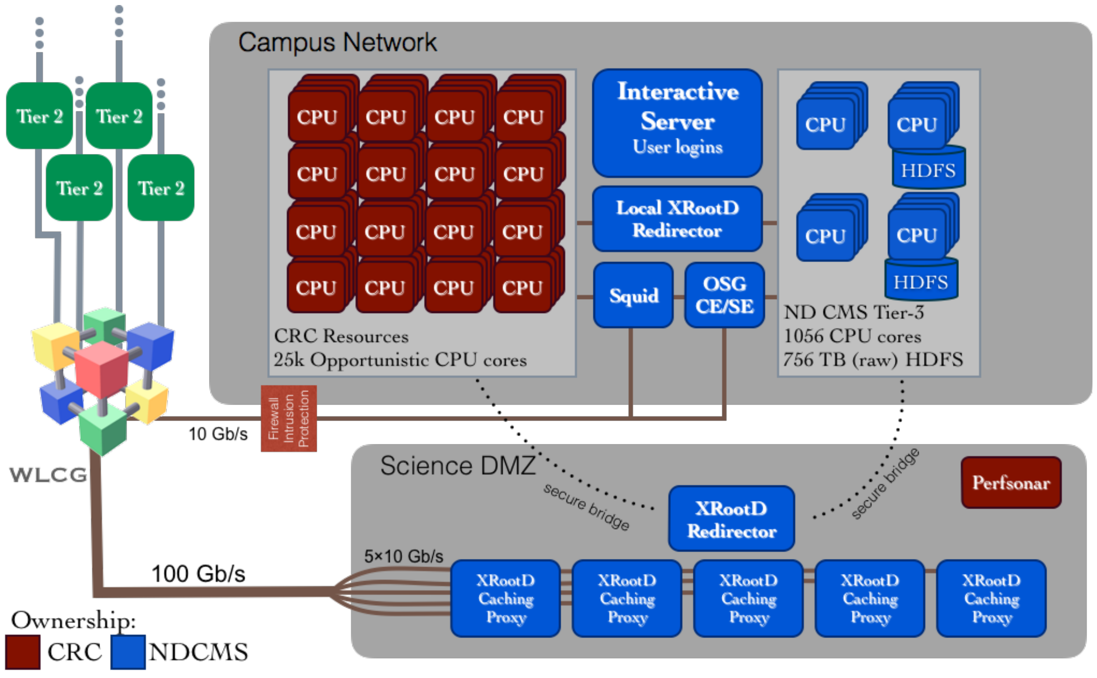


Figure 1. This diagram shows the configuration of dedicated ND Tier-3 (labeled NDCMS and shown in blue) and CRC (shown in red) resources used to extend the ND Tier-3 with opportunistic CRC resources. In general, resources from the campus network are unable to access the Science DMZ except by using the the "secure bridge" links as illustrated.

Recently, the CRC has deployed a Science DMZ with access to a 100 Gb/s network connection, improving the ND campus bandwidth by a factor of 10 from the previous 10 Gb/s capacity. This improved network bandwidth is extremely attractive, allowing a larger number of I/O-intensive CMS applications to run concurrently on opportunistic resources. However, CRC security policy prevent most computing resources from being deployed in the DMZ, which means that CMS jobs running on the opportunistic resources cannot directly access the 100 Gb/s network link. To realize the full potential of the CRC resources, it was necessary to deploy an XRootD proxy [12] infrastructure (described below) within the ND science DMZ.

What follows is a discussion of the various challenges encountered in attempting to scale the ND Tier-3 up to roughly 25,000 CPU cores and 100 Gb/s network bandwidth. Although not all obstacles have been overcome, the performance achieved already confirms the encouraging potential of this approach.

2. Challenges

Making use of the improved network and large-scale of opportunistic resources presented a number of challenges. The network-related challenges were in general independent of those created by the opportunistic resources. Therefore, we will describe these two classes of resources separately below.

2.1. DMZ and Network

There are two main categories of network-related resources: those connected to commissioning and debugging ND's new 100 Gb/s network connection and those connected with accessing that network connection via the Science DMZ in accordance with CRC security policy.

CMS data files are globally distributed via XRootD. Though the campus worker nodes are not directly part of the 100 Gb/s network, they can access it via proxy servers. Caching of CMS data files on the local proxy servers allows faster access by worker nodes and reduces incoming network traffic. The solution to the second set of problems starts with deploying a proxy service for XRootD access. Jobs running on CRC opportunistic resources use a secure bridge connection to access the XRootD proxy which then allows access to the global CMS AAA data federation. The XRootD proxy software [12] is provided as part of the overall XRootD software package. A single server in the Science DMZ serves as an XRootD redirector, sending connections from jobs running on the opportunistic resources to one of five XRootD proxy servers via the Cluster Management System daemon (cmsd). Each XRootD proxy server has a 10 Gb/s duplex network link, providing for a theoretical maximum aggregate XRootD throughput of 50 Gb/s. The drawback of this solution is that the added complexity complicates debugging network issues. However, this complexity also brings with it new flexibility, as the proxy service is also capable of providing local caching to improve performance of repeated access to remote files. Each proxy server is equipped with 24 TB storage disk for caching. The network and caching capacity of the system can be further scaled by adding additional caching proxy servers.

To tackle challenges like establishing the baseline performance of network connections between one local and one remote server, as well as to monitor the network for variances in performance over different sites at different times, we employ a 10 Gb/s tuned PerfSonar [13] node. This node runs automated tests to identify single data stream bottlenecks and baseline values between ND and selected WLCG Tier-1 and Tier-2 sites. PerfSonar measures the bandwidth, latency, and number of hops to each monitoring site and maintains a record of those statistics as a function of time. Figure 2 shows typical test results obtained. These test results allow us to identify and debug network problems associated with single links, up to the bandwidth of the PerfSonar server network card.

Nevertheless, a particular challenge arises because the goal is to aggregate multiple 10 Gb/s connections to fill the 100 Gb/s network link with transfers from multiple sites to multiple XRootD proxy servers. It can be hard to spot bottlenecks that are $\geq$ 10 Gb/s. For example, when debugging a network issue with one Tier-2 site, a 10 Gb/s bottleneck was discovered in what was supposed to be a fully 100 Gb/s route because one link carrying half the traffic was accidentally not upgraded from 10 Gb/s to 100 Gb/s. Tools like PerfSonar do not currently provide an automated way to test for such problems.

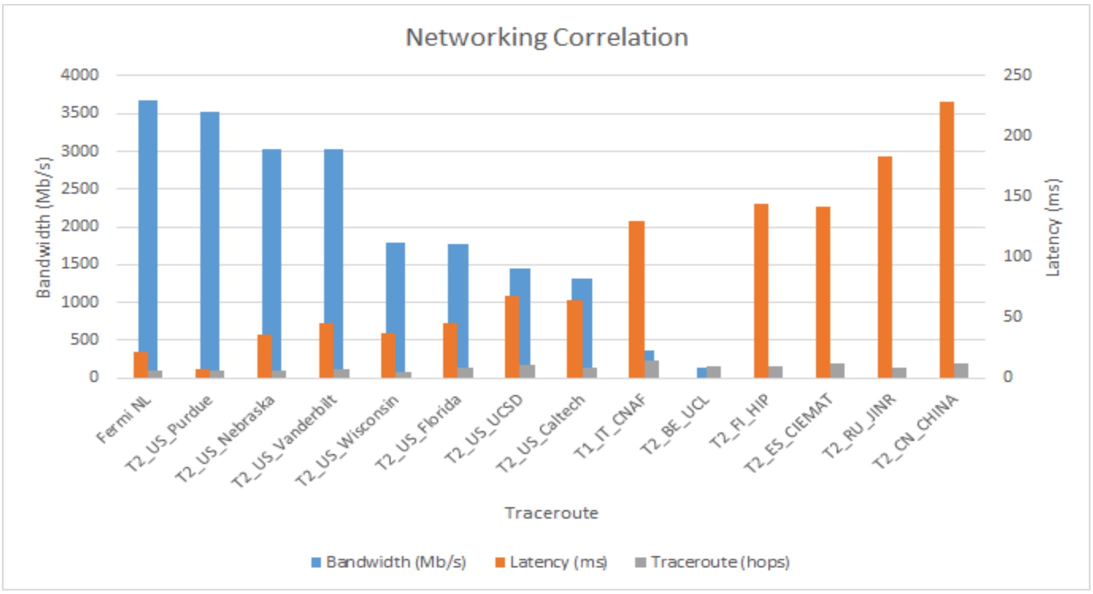
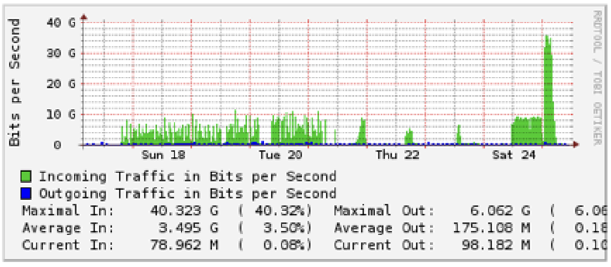


Figure 2. Measurements using of bandwidth, latency, and number of hops made using PerfSonar. There is a loose correlation between bandwidth and latency, but there is enough variation from one site to the next to suggest that further debugging of network connections could be beneficial.



```
🟩 Incoming Traffic in Bits per Second
🟦 Outgoing Traffic in Bits per Second
Maximal In:    40.323 G  ( 40.32%)  Maximal Out:    6.062 G  (  6.06
Average In:     3.495 G  (  3.50%)  Average Out:  175.108 M  (  0.18
Current In:    78.962 M  (  0.08%)  Current Out:   98.182 M  (  0.10
```

Figure 3. Average bandwidth traffic coming from the XRootD Servers.

Beyond debugging network issues, it was also necessary to overcome challenges within the XRootD proxy caching infrastructure itself. Although the primary motivation for deploying multiple proxy servers is network capacity, it was determined through system testing that multiple servers are also required to handle the large number of concurrent connections that can arise when several thousand tasks are running on the opportunistic resources in parallel. The high CPU, memory, and high disk I/O loads demanded by serving all of those connections makes having multiple servers desirable regardless of the actual bandwidth requirements. Therefore, the load balancing provided by the cluster management daemon is essential. Furthermore, because ND is the first site to use an XRootD caching proxy configuration at this scale, we were able to uncover a number of previously undiscovered bugs in the XRootD caching proxy software. In collaboration with XRootD developers (in particular Matevz Tadel from USCD and Andrew Hanushevsky from SLAC) over the course of 4 experimental releases, we were able to resolve all the bugs. The servers were tuned to deliver 6-7 Gb/s of data transfer for production workflows, and we have achieved a sustained aggregate average bandwidth over 35 Gb/s, as shown in Figure 3. We aim to achieve sustained rate of 50 Gb/s in the coming year with further tuning and additional servers.

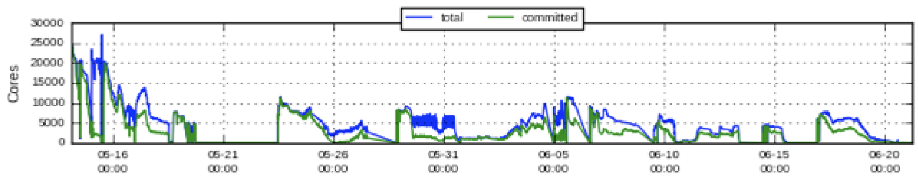


Figure 4. Plot showing number of cores used by Lobster, peaking around 25,000 cores. The 25,000 core peak was achieved when the ND CMS team was granted exclusive use of CRC resources during a maintenance downtime. The rest of the plot shows running done during regular CRC operations in competition with other users. Even during these periods, there are times when as many as 10,000 CPU cores can be utilized.

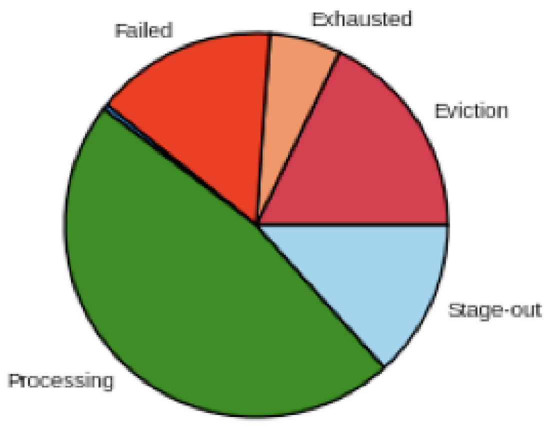


Figure 5. Accounting for time spent on processing versus time lost to job failures, jobs being killed for exhausting their resources, jobs being evicted, or delays in staging out that arising from the master becoming overloaded.

2.2. Workflow Management

Trying to make effective use of several thousand tasks using 10,000-20,000 CPU cores presents a number of workflow management challenges. The Lobster software was designed with these challenges in mind, and as the available resources through the CRC continue to increase, we have continued to evolve Lobster to meet the challenges discovered. In particular, the CRC has provided our team the opportunity to do scale testing using the full 25,000 CPU cores of CRC resources up to twice a year during bi-annual maintenance downtimes. The results of one such scale test are depicted in Figures 4 and 5. Below, we discuss the problems uncovered during this scale test and others like it as well as steps that are being taken to overcome these challenges.

A significant challenge involves the distribution of the software needed to run CMS jobs to the opportunistic resources. The opportunistic resources accessed via CRC are not configured for CMS. CMS software environment is delivered over the network via CVMFS + Parrot. To minimize the load on the network, workers maintain cache of software that persists between tasks for the lifetime of worker. This functionality is provided by Work Queue. Achieving the current scale of running required a concerted effort in tuning caching strategies to ensure efficient sharing of transferred files among tasks running on the same worker.

Another significant challenge involves the scheduling policies of the CRC opportunistic resources. Although HTCondor allows us to access the widest range of resources, this method comes with the drawback that when the resource is scheduled via UGE, the HTCondor job is evicted immediately and without warning, leading to lost work. Currently CMS applications have no ability to checkpoint, so progress on an evicted task is lost. Lobster seeks to mitigate these losses by running short tasks. This is made possible through the use of Work Queue workers that cache the software and environment needed for tasks to minimize the overhead as described above. Lobster uses a database to track task completion. Task output merged in separate step to produce conveniently sized output files. The merging part of the workflow has been carefully tuned to proceed in parallel with running other tasks to minimize the processing tail incurred by merging the output from completed tasks.

Users often have only vague idea of resource needs of tasks from local testing. When scaling workflows to large scales, this can present a number of problems. For example, in order to optimally pack tasks on resources, it is important to have good estimates of the number of CPU cores, the amount of memory, the amount of local storage, and the among of network bandwidth the task requires. The limited local testing a user is able to do usually yields imprecise information and users are faced with difficult choices. If they make too conservative an estimate of task resource needs, they risk leaving resources idle. The effects of this can be seen in Figure 4 where the "committed" line in the graph which represents the resources that are able to be used effectively is below the "total" line at times. If they are two aggressive, tasks will fail due to exhausted resources. This impact can be seen in the "Exhausted" fraction of the pie chart in Figure 5. Lobster addresses this challenge by using the `resource_monitor` [14] that measures resource usage on running tasks. The `resource_monitor` feeds task resources usage information back to Work Queue so that it can adjust task resource needs accordingly. Resource Monitor also prevents tasks from crashing opportunistic resources by exceeding requested resource allocation (for example, sending a node into thrashing for swap space due to excessive memory usage). Evolving task resource needs are communicated back to user via monitoring plots, allowing user to tune worker resource allocations to best match task needs. Further improvements are being pursued to make it easier to Lobster to adapt resource allocation and worker sizing with less manual intervention by users.

Another challenge facing the use of CRC resources is the dynamic nature of the resources, making transient problems likely and resulting in potentially thousands of failed tasks. Lobster has built in redundancy for failure prone aspects of tasks: cascading fall back of input and output transfer mechanisms, automatic task retrying, and blacklisting of problematic worker nodes according to a user-configurable exit code list. Lobster tracks task failures and successes and handles all retries without user intervention. These capabilities are essential for running at scale. Otherwise, users would spend all of their time trying to recover from transient failures and would not be able to keep their work progressing.

At the scales of resources we are trying to use, we have started to see scaling issues within the Lobster application itself. As resource utilization scales to approximately 25,000 cores, it becomes challenging for the Lobster master to keep up with returning tasks and starting new tasks. The effect of this can be seen in the relatively large fraction of the time spend in "stage-out" which indicates otherwise finished tasks waiting for their opportunity to return outputs to the Lobster master process. We are currently exploring techniques to alleviate the load on the master, including use of Work Queue foremen, delegating more accounting activity to the individual tasks, and optimizing database and file system access.

3. Conclusion

Although certainly not without difficulties, the initial experiment of scaling up the ND Tier-3 cluster using opportunistic resources from the CRC and taking advantage of the new 100 Gb/s network link has demonstrated great potential. After a few months of effort, we were able to achieve over 35 Gb/s network throughput from the CMS AAA data federation to opportunistic

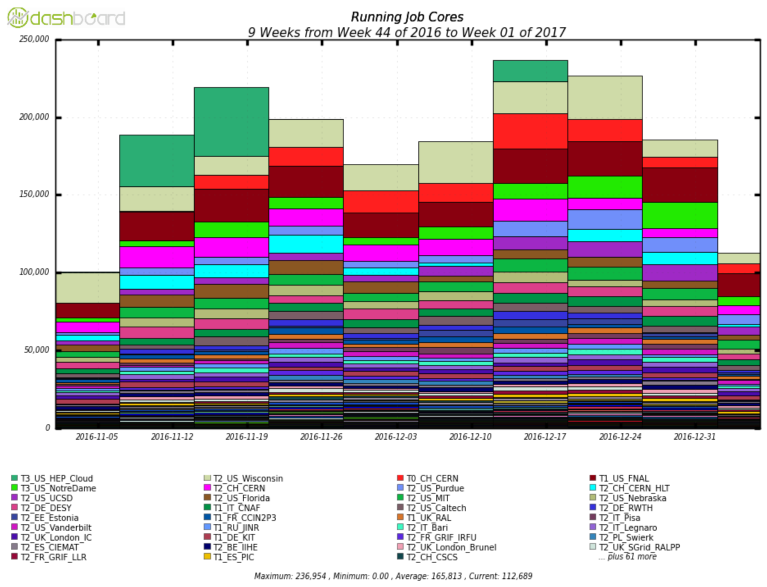


Figure 6. This plot shows the number of CPU cores utilized by running jobs at various CMS WLCG sites. The contributions from ND's Tier-3 site over the period shown are larger than most other Tier-1 and Tier-2 sites. The only Tier-3 site providing comparable CPU resources is the "HEPCloud" site which is an experimental facility operated by Fermi National Laboratory making use of AWS and Google Cloud resources. Note: 61 sites are compressed into the bottom-most contribution in the plot.

resources, thereby eliminating the previous 10 Gb/s campus network bottleneck. This level of network bandwidth is currently sufficient to provide for the scale of opportunistic computing we are routinely able to access, and can be scaled up further through investment in additional XRootD proxy servers. Through dedicated scale testing with the full scale of CRC resources, we have been able to identify a number of bottlenecks and performance issues in the Lobster workflow management software and have already begun to make improvements to address those challenges. Although Lobster is not currently ready to run efficiently at a scale of 25,000 CPU cores smoothly for many weeks, we have already demonstrated the ability to at the smaller scale of 5,000 to 10,000 CPU cores that are routinely available opportunistically from the CRC. Stable running at this scale is a noteworthy achievement in itself, enabling a single ND graduate student at times to harness resources equivalent to some of the largest Tier-1 or Tier-2 sites, as demonstrated in Figure 6.

References

[1] Worldwide LHC Computing Grid, `http://wlcg.web.cern.ch`.

[2] Open Science Grid, `https://www.opensciencegrid.org`.

[3] B Blumenfeld, D Dykstra, L Lueking, and E Wicklund. Cms conditions data access using frontier. *Journal of Physics: Conference Series*, 119(7):072007, 2008.

[4] CernVM File System, `http://cernvm.cern.ch/portal/filesystem`.

[5] A. Hanushevsky and D.L. Wang. Scalla: Structured cluster architecture for low latency access. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1168–1175, May 2012.

[6] Anna Woodard, Matthias Wolf, Charles Mueller, Nil Valls, Ben Tovar, Patrick Donnelly, Peter Ivie, Kenyi Hurtado Anampa, Paul Brenner, Douglas Thain, Kevin Lannon, and Michael Hildreth. Scaling Data Intensive Physics Applications to 10k Cores on Non-Dedicated Clusters with Lobster. In *IEEE Conference on Cluster Computing*, 2015.

[7] Anna Woodard, Matthias Wolf, Charles Nicholas Mueller, Ben Tovar, Patrick Donnelly, Kenyi Hurtado Anampa, Paul Brenner, Kevin Lannon, and Michael Hildreth. Exploiting Volatile Opportunistic Computing Resources with Lobster. In *Computing in High Energy Physics*, 2015.

[8] Douglas Thain and Miron Livny. Parrot: An Application Environment for Data-Intensive Computing. *Scalable Computing: Practice and Experience*, 6(3):9–18, 2005.

[9] Michael Albrecht, Dinesh Rajan, and Douglas Thain. Making Work Queue Cluster-Friendly for Data Intensive Scientific Applications . In *IEEE International Conference on Cluster Computing*, 2013.

[10] Peter Bui, Dinesh Rajan, Badi Abdul-Wahid, Jesus Izaguirre, and Douglas Thain. Work Queue + Python: A Framework For Scalable Scientific Ensemble Applications. In *Workshop on Python for High Performance and Scientific Computing (PyHPC) at the ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (Supercomputing)* , 2011.

[11] Kenneth Bloom and the CMS Collaboration. CMS Use of a Data Federation. *Journal of Physics: Conference Series*, 513(4):042005, 2014.

[12] LAT Bauerdick, K Bloom, B Bockelman, DC Bradley, S Dasu, JM Dost, I Sfiligoi, A Tadel, M Tadel, F Wuerthwein, et al. Xrootd, disk-based, caching proxy for optimization of data access, data placement and data replication. In *Journal of Physics: Conference Series*, volume 513, page 042044. IOP Publishing, 2014.

[13] PerfSonar, `http://www.perfsonar.net/`.

[14] Resource Monitor, `http://ccl.cse.nd.edu/software/resource_monitor/`.