# Feedback Computing in Leadership Compute Systems

Raghul Gunasekaran
*Oak Ridge National Laboratory*
*gunasekaranr@ornl.gov*

Youngjae Kim
*Oak Ridge National Laboratory*
*kimy1@ornl.gov*

## Abstract

Leadership class systems are heavily shared resource environments with users contending for shared system resources. This results in users experiencing huge performance variations, and also affects the overall throughput of the system. To alleviate the problem, system software tools must be built taking into consideration user requirements and resource availability, a feedback driven approach. Realizing a feedback-based compute environment for peta-scale systems have two challenging tasks. First, collecting discreet, coarse-grained system statistics from multiple systems using minimum system resources and without affecting the user jobs is a hard problem. Second, with discreet data collected from disparate sources the challenge is in associating the data for meaningful interpretations to drive feedback-based decision systems in real-time. In this paper, we elaborate on a feedback-based computing framework with respect to the peta-scale compute and storage system at the Oak Ridge Leadership Computing Facility. We describe our feedback-based approach for dynamic resource allocation, context-aware scheduling and application checkpointing.

## 1 Introduction

The Oak Ridge Leadership Computing Facility (OLCF) has been housing some of the world's fastest supercomputers in the past decade. Most recently the Jaguar system (2009-2013) [3], with a peak performance of 2.3 petaflops and was the world's fastest supercomputer in 2009 [6]. Currently, OLCF hosts the Titan [4] system, the second fastest supercomputer with a peak performance of 27 petaflops. The massive compute power is used to solve problems in multiple science domains such as climate, chemistry, astrophysics, materials, nuclear physics, quantum mechanics, and alternative energy sources. With over 400 scientific users across science domains, OLCF has one of the largest and fastest parallel file system, Spider [13], operating at a throughput of 1TB/s and storage capacity of 32 petabytes, supporting the massive compute infrastructure.

OLCF's infrastructure built for *capability computing*, maximum computing power to solve a single large problem in the shortest time. The workload is also dominated by *capacity computing*, where a number of users share the compute infrastructure to solve reasonably large problems. User's running concurrently on the compute infrastructure contend for shared resources, which impacts both individual users job performance and the overall system throughout. Also, the workflow and resource utilization characteristics of individual scientific applications and users are very different, creating bursts of active and idle periods of resource usage. Such mixed workloads pose unique operational challenges for OLCF and the scientific users to operate at scale. Individual scientific applications are well studied and documented during the development phase in a controlled environment. However, the behavior of the same application on a shared resource environment running at-scale concurrently with other applications is very different and can be only captured in production runtimes. Observing applications runtime behavior in conjunction with resource utilization and availability can provide useful feedback to provision system resources and better scheduling of jobs.

Computing facilities have extensively developed efficient tools and techniques to monitor extreme scale systems, and capturing system logs and metrics with minimum overhead. Developers and system administrators extensively rely on these tools for debugging software, fixing hardware failures and identifying anomalous user activities. However, currently most of these tools are being used only for monitoring the health of the system or capturing abnormal activities. In the paper, we recommend on how the data being collected can be used to design smarter tools. Analyzing and correlating data from all system components, we recommend a feedback-

based approach to make resource-aware system decisions, specifically for scheduling and storage resource consumption.

## 2 OLCF Overview

### 2.1 Infrastructure

Titan, OLCF's primary compute platform, has 18,688 compute nodes, each compute node has a 16-core AMD Opteron processor and a K20X Kepler GPU. The total system memory is 710TB, with 32GB for each Opteron and 6GB for the GPU per compute node. Apart from Titan, OLCF's hosts other high performance clusters(HPC) for development, data analytics and visualization purposes. Supporting this massive compute infrastructure, over 26,000 clients, is the Spider II filesystem. Spider II is a Lustre [14] based center-wide parallel file system, accessed by the compute infrastructure via the InfiniBand Scalable I/O Network (SION), as shown in Figure 1. Spider has 20,160 SATA drives managed by 36 DDN SFA10K couplets. Access to the Spider file system is via the 288 Lustre object storage servers(OSS). The filesystem is available as two namespaces, *atlas1* and *atlas2*, for load-balancing and capacity management purposes.

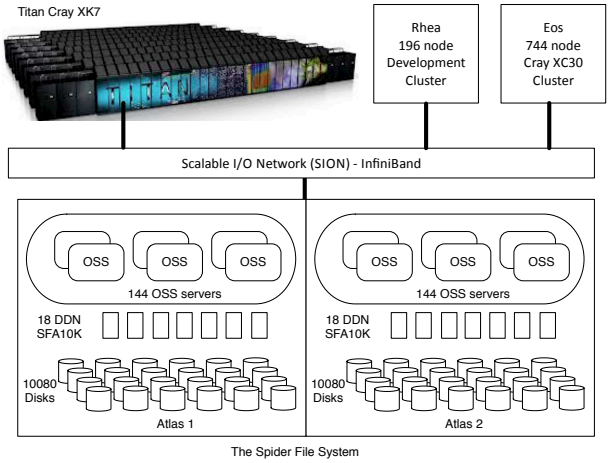


Figure 1: OLCF compute infrastructure.

### 2.2 Monitoring

The health of the compute platform is monitored via the RAS (Reliability, Availability and Serviceability) logs. The RAS logs provide a wealth of information on the status of the system, but in systems like Titan to manage the sheer volume of log generated only high level debug messages are generated. Similarly, the OSS's provide Lustre server-side error logs. To monitor the file system usage, a custom tool [11] developed in house, queries the

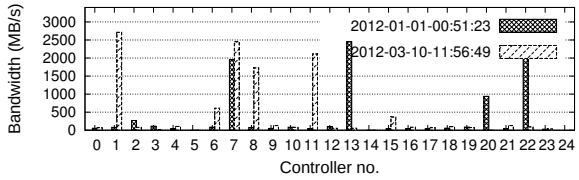


Figure 2: Snapshot of individual controller's bandwidth for two different times [7]. We show the statistics of performance data for each day above with (Min, Max, Avg, STD, Date). (0.0, 2449.0, 373.2, 724.6, 2012-01-01), (0.0, 2713.0, 465.5, 822.6, 2012-03-10).

RAID controller for the read/write bandwidth and IOPS data, along with the request size distribution at frequent intervals. This data is collected and stored in a Mysql database for offline analytics.

### 2.3 Challenges for Shared Resources

OLCF's compute platforms are shared resources, where a number of users run their simulations concurrently, causing a resource contention problem on network and storage subsystems. Lustre uses a native API called LNET (Lustre Networking) on top of the Lustre Networking Device (LND) layer, which enables routing between compute nodes and storage servers. A routing strategy with no link congestion awareness could inject increased traffic into the InfiniBand network, causing unbalanced network traffic loads [5]. Also, Lustre uses a locking mechanism for synchronized accesses to a file from multiple clients, in which all write operations can slow down for shared write access [15]. Similarly, storage servers can be overloaded by bursty I/O requests from multiple clients, causing uneven load distribution and contention on isolated servers [8]. Figure 2 shows the I/O loads imbalance observed across controllers over a period of time. We observe that a few controllers are overloaded whereas most are not, and they show very high bandwidth utilizations compared to others with much lower bandwidth utilization. Aware of the challenges on the shared compute platform because of resource contention, in this paper, we specifically discuss the opportunities for a feedback-based computing framework and how they can help in the development of integrated software stacks including parallel file system, I/O middleware, application-level schedulers, and system resource provisioning.

## 3 Feedback computing for HPC

In high-end computing systems, individual application's throughput is highly dependent on the resource availability. Also, the overall system performance is dependent

on the efficient usage of system resources. System software tools need to build on feedback-based scheme for provisioning resources based on availability and application requirements. However, the challenge and limitation towards building a feedback-based compute system for large-scale systems is in collecting system and log data. First, exhaustive logging will affect the system performance and consume resource, thus limiting the amount of data we log. Second, with minimum logging it is challenging to interpret useful information. Third, with over 26,000 active clients, array of switches and router, and tiers of storage devices we have a deluge of data to be monitored. Finally, user's resource utilization changes over time and it is a continuous learning process towards understanding user demands.

At OLCF, we have been collecting logs and aggregate statistics from Titan and Spider; and the data is being by pushed out by the management node and archived in a database for analysis purposes only, a permanent archive is stored elsewhere. Similarly, schedulers log are also dumped into the database. In Figure 3, we show a feedback-based compute system. A decision support system queries the database for relevant data, and supports both pull and push services feeding information back to the users and the scheduler. From the data collected, we have made useful interpretation, as detailed below, which in turn is driving the development of feedback-based computing tools, elaborated in the next section.

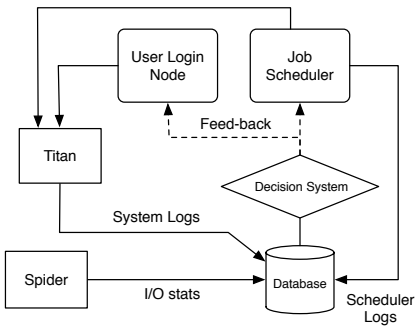


Figure 3: Feedback computing at OLCF

**Characterizing I/O workloads** provides useful insight on storage system utilization. An understanding of the workload helps define best practices for users, identifying bottlenecks, develop tools that can enable better utilization of resources, and also help plan for future storage systems. Aggregating the I/O trace data collected across RAID controllers, we were able to understand I/O usage patterns and synthesize workloads. A few interesting observations from our study [8]; first, the I/O workload tends to be write dominant with 60% writes and 40% reads. Second, a majority of our I/O requests are small (16KB) or large (multiples of 1MB) requests. Third, inter-arrival and ideal-time of I/O request follow a long tailed distribution. Finally, I/O activity tends to be burst, with period of high and low activity, and not all controllers operate at their peak at the same time.

**I/O Signature Identification (IOSI)** [9], captures the I/O access pattern of users by observing the throughput stats at RAID controller and correlating them with the scheduler's log. Using standard data mining techniques, IOSI identifies the access pattern common across a user's runtime obtained from the schedulers log. The process unlike client-side tracing tools, has zero-overhead and no user intervention. IOSI provides an estimate of user's I/O, which indeed can help in storage provisioning and I/O aware scheduling.

**Mining RAS logs** provides an insight on the current state of the system and to detect hardware, software and silent errors. System administrators monitor the logs for known software and hardware errors using Simple Event Correlator [12], an event processing tool that observes the log for predefined events, triggers notification and also launches external scripts. While silent errors are a sequence of log events that do not result in a system or application failure, but causes a significant performance degradation. Example errors such as network congestion and timeout messages appear periodically but only impact the system beyond a threshold. In a shared resource environment, though only few applications cause the problem other concurrently running applications are also impacted. Detecting silent errors is critical to distinguish between applications that are the source, and other applications that are the victim of such errors. Applications causing such errors are not at fault but are simply resource intensive, and identifying such applications can help make smarter systems decisions.

# 4 Feedback-based HPC paradigms

In this section we describe feedback-based system paradigms driven by data from monitoring tools, that alleviates resource contention and provisions for uniform resource utilization.

## 4.1 Dynamic Resource Allocation

### 4.1.1 Job placement

The choice and availability of compute and I/O resources play a significant role in system and application performance. Titan with over 18,000 nodes, jobs are scheduled ensuring that all compute nodes are used, maximizing the overall system utilization. Also TITAN's Gemini 3D interconnect network is anisotropic, meaning the *x, y*

and $z$ axis of the torus interconnect have different transfer rates. The performance of applications is very much dependent on the allocation of compute nodes. Primarily jobs are assigned nodes by the scheduler based on the job request FIFO queue, and a more sophisticated approach has been topology aware job scheduling [2], [1]. While FIFO queue simply ensures fairness of allocation, topology aware scheduling considers the physical infrastructure in node placement. In [2], topology awareness for MPI jobs is enforced by assigning node ranks based on the nodes locations in the 3D torus interconnect, maximizing bisection bandwidth and minimizing intra-job communication.

With topology awareness, schedulers must also consider application requirements in the allocation of nodes. Applications with heavy inter-process communication are directly impacted by the choice of nodes. In a 3D torus setup an ideal node placement would be to select nodes uniformly across all dimensions. However, such a placement is practically not feasible for all jobs and not all applications are sensitive to placement. Further more challenging with an anisotropic interconnect, where job placement should factor in the dimensions with transfer bandwidth. For a feedback-based setup, the objective is to learn how an application's performance varies based on the job placement. We build a feedback-driven scheduler by identifying those applications that are sensitive to node location, feeding this information to the scheduler, and developing placement strategies that satisfy application requirements and also ensure 100% utilization of resources.

#### 4.1.2 File system selection

As described earlier, the Spider filesystem is available as two namespaces, and user and application domains by default are statically allocated one namespace. The static assignment is based on known application characteristics, I/O and storage usage. However, users within a domain can run concurrent jobs causing heavy load on a particular namespace. Nevertheless, users can request for allocation on either namespace and can pick either of the namespace for I/O. Given an option to choose either namespace; users have no workload information to pick the right namespace. Also, the choice of namespace is a run time decision, at the time the job is launched. This motivates the need for feedback-based service, which monitors the filesystem workload and help the user choose the namespace dynamically at runtime. The filesystem namespace selection utility is a service the user can query at runtime, and the utility provides the user with the lighty loaded namespace. In case both the namespaces are busy serving I/O requests, the decision is in identifying the namespace that best suites the user's I/O workload. The user's I/O access pattern can be mined using IOSI from past runs.

### 4.2 I/O-aware Scheduling

In general, jobs are scheduled based on their priority, create time, node count and other factors that ensure that the compute platform is fully utilized and the job is queued only for a reasonable amount of time. Common practice has been to schedule based on known application behavior and with system administrator intervention schedule for a dedicated run. However, to incorporate I/O smarts into the scheduler we have to factor in several factors: the current filesystem workload, estimated I/O workload based on jobs currently scheduled and in the case of a center wide filesystem I/O by external users. An I/O aware scheduler needs to factor in all these parameters for scheduling the next job from the queue, and also ensure utilization of all the compute resources.

### 4.3 I/O-Server Selection

Most parallel filesystems such as Lustre, GPFS, and PVFS balance I/O loads in terms of disk space, and ensure disk volumes grow at the same rate. Specifically, OLCF's Lustre filesystem, which stripes files into four units of 1MB on the OSS servers in a round robin fashion. However, I/O utilization varies across the servers over time, which we have empirically observed that some of the disk volumes are overloaded by a large number of read and write requests, referred to as the I/O load imbalance problem in the PFS [10]. The data usage stats collected at the controllers will help identify the overloaded server in real time, and this data can be relayed to the file system user by suggesting those servers which are lightly loaded; avoiding congestion and provisioning for improved performance.

### 4.4 Smart Checkpointing

Failures are a more serious concern in exa-scale systems, as the failure rate increase non-linearly as the systems scale. Checkpoint/restart is a dominant fault tolerant mechanism, and the overhead associated with checkpoint is significant. The checkpointed data needs to be stored in the persistent storage, and they will be read for restart, resulting in heavy bursts of read and write I/O's on the PFS. As described earlier, the shared PFS will need to be carefully managed for accessing the servers, otherwise, the expected I/O bandwidth will not be guaranteed. Thus, a feed-back mechanism between resource utilization data and checkpoint decision service, can help determine when to checkpoint and volume of data to checkpoint. Such smarts will results in a co-ordinated

checkpoint strategy between users, minimizing the performance degradation due to unexpected congested I/O servers.

## 5 Concluding Remarks

In this paper, we have discussed on a feedback-based computing framework that can alleviate system bottlenecks for better utilization of system resources. Though a few of the ideas elaborated in this paper have been discussed in the HPC community, practical realization has been minimal and challenging. One of the biggest obstacle is in monitoring and collecting the system data for enabling feedback based computing. We at OLCF have built custom tools to collect such useful statistics and working towards building such feedback-based smart tools.

## Acknowledgment

This work was supported by the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is managed by UT Battelle, LLC for the U.S. DOE (under the contract No. DE-AC05-00OR22725).

## References

[1] T. Agarwal, A. Sharma, and L. V. Kalé. Topology-aware task mapping for reducing communication contention on large parallel machines. In *Proceedings of the 20th International Conference on Parallel and Distributed Processing*, 2006.

[2] C. Albing, N. Troullier, S. Whalen, R. Olson, J. Glenski, H. Pritchard, and H. Mills. Scalable node allocation for improved performance in regular and anisotropic 3d torus supercomputers. In *Proceedings of the 18th European MPI Users' Group Conference on Recent Advances in the Message Passing Interface*, 2011.

[3] A. Bland, R. Kendall, D. Kothe, J. Rogers, and G. Shipman. Jaguar: The worlds most powerful computer. In *Proceedings of the Cray User Group Conference*, 2009.

[4] A. S. Bland, J. C. Wells, O. E. Messer, O. R. Hernandez, and J. H. Rogers. Titan: Early experience with the Cray XK6 at Oak Ridge National Laboratory. In *Proceedings of Cray User Group Conference (CUG 2012)*, May 2012.

[5] D. A. Dillow, G. M. Shipman, S. Oral, Z. Zhang, and Y. Kim. Enhancing i/o throughput via efficient routing and placement for large-scale parallel file systems. In *Proceedings of the 30th IEEE International Performance Computing and Communications Conference*, PCCC '11, pages 1–9, Washington, DC, USA, 2011. IEEE Computer Society.

[6] J. Dongarra, H. Meuer, and E. Strohmaier. Top500 supercomputing sites. `http://www.top500.org`, 2009.

[7] Y. Kim, S. Atchley, G. Vallée, and G. M. Shipman. Layout-aware i/o scheduling for terabits data movement. In *BigData Conference*, pages 44–51, 2013.

[8] Y. Kim, R. Gunasekaran, G. M. Shipman, D. Dillow, Z. Zhang, and B. W. Settlemyer. Workload characterization of a leadership class storage. In *Proceedings of the 5th Petascale Data Storage Workshop Supercomputing '10 (PDSW'10) held in conjunction with SC'10*, November 2010.

[9] Y. Liu, R. Gunasekaran, X. Ma, and S. Vazhkudai. Automatic identification of application i/o signatures from noisy server-side traces. In *Proceedings of the 12th USENIX conference on File and Storage Technologies (FAST)*, 2014.

[10] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, and M. Wolf. Managing Variability in the IO Performance of Petascale Storage Systems. In *SC*, 2010.

[11] R. Miller, J. Hill, D. A. Dillow, R. Gunasekaran, G. M. Shipman, and D. Maxwell. Monitoring tools for large scale systems. In *Proceedings of Cray User Group Conference (CUG 2010)*, May 2010.

[12] Risto Vaarandi. A data clustering algorithm for mining patterns from event logs. In *IEEE IPOM03 Proceedings*, 2003.

[13] G. M. Shipman, D. A. Dillow, D. Fuller, R. Gunasekaran, J. Hill, Y. Kim, S. Oral, D. Reitz, J. Simmons, and F. Wang. A Next-Generation Parallel File System Environment for the OLCF. In *Proceedings of Cray User Group Conference (CUG 2012)*, May 2012.

[14] Sun Microsystems Inc. Luste Wiki. `http://wiki.lustre.org`, 2009.

[15] B. Xie, J. Chase, D. Dillow, O. Drokin, S. Klasky, S. Oral, and N. Podhorszki. Characterizing output bottlenecks in a supercomputer. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 8:1–8:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.