JIRIAF: JLAB Integrated Research Infrastructure Across Facilities

Gyurjyan Vardan[1], *Larrieu* Christopher[1], *Heyes* Graham[1], *Lawrence* David[1]

[1]JLAB, CST Division, 12000 Jefferson Avenue, Newport News, Virginia, 23606, USA

Abstract. The JIRIAF project aims to combine geographically diverse computing facilities into an integrated science infrastructure. This project starts by dynamically evaluating temporarily unallocated or idled compute resources from multiple providers. These resources are integrated to handle additional workloads without affecting local running jobs. This paper describes our approach to launch best-effort batch tasks that exploit these underutilized resources. Our system measures the real-time behavior of jobs running on a machine and learns to distinguish typical performance from outliers. Unsupervised ML techniques are used to analyze hardware-level performance measures, followed by a real-time cross-correlation analysis to determine which applications cause performance degradation. We then facilitate bad behavior by throttling these processes. We demonstrate that problematic performance interference can be detected and acted on, which makes it possible to continue to share resources between applications and simultaneously maintain high utilization levels in a computing cluster. For a case study, we relocated the CLAS12 data processing workflow to a remote data processing facility, preventing file migration and temporal data persistency.

1. Introduction

The JIRIAF project aims to combine geographically diverse computing facilities into an integrated science infrastructure. The term "integrated computing infrastructure" refers to combining different types of computer hardware and software to provide a unified environment for users. This type of system allows for easier management and maintenance of distributed resources of participating computing facilities.
There are several reasons why computing infrastructure integration is so important. First, it makes it possible to share data across multiple locations. Second, it helps reduce costs by reducing the need for duplicate equipment. Third, it improves security by allowing only authorized personnel to access sensitive data. Fourth, it provides better service by ensuring all users can access the exact services anytime. Finally, it reduces downtime by ensuring no interruption in service when one component fails. One of DOE computing facilities' most significant challenges today is integrating different computing resources into a unified infrastructure. This challenge has become even more difficult as technology evolves and we face hardware heterogeneity. The goal is to maximize data processing and science output, realizing that only an accurate prediction of resource consumption enables a proper selection, allocation, and integration of resources to maximize data-processing efficiency. To make things even more complicated, there are different types of integration. For example, data integration solutions allow users to access data across multiple systems. There are also application integration solutions that will enable applications to work together. One of the essential aspects of this integration is to utilize unused temporary resources for remote workloads without affecting local running jobs. Our approach is to define and monitor application-independent hardware-level performance measures and act on them in case of signs of performance degradation.

2. Architecture

JIRIAF project evaluates and implements the mentioned types of integration, utilizing widely adopted technologies.
The architecture of the JIRIAF is shown in Fig 1. The database holds information about available integrated resources and user job requests. The resource pool is furnished by JFM (JIRIAF facility manager), which periodically scrapes resource information from each computing facility. This information is then passed to JSC (JIRIAF central service), which starts a pilot job: JRM (JIRIAF resource manager) leasing JFM-reported resources in a facility. After JRMs run, JSC will immediately update the available resource DB table to match the user request table for workflow

migration (JMS: JIRIAF workflow resource matching service algorithm). User requests furnish the user workflow request table through the JFE (JIRIAF front end) component.

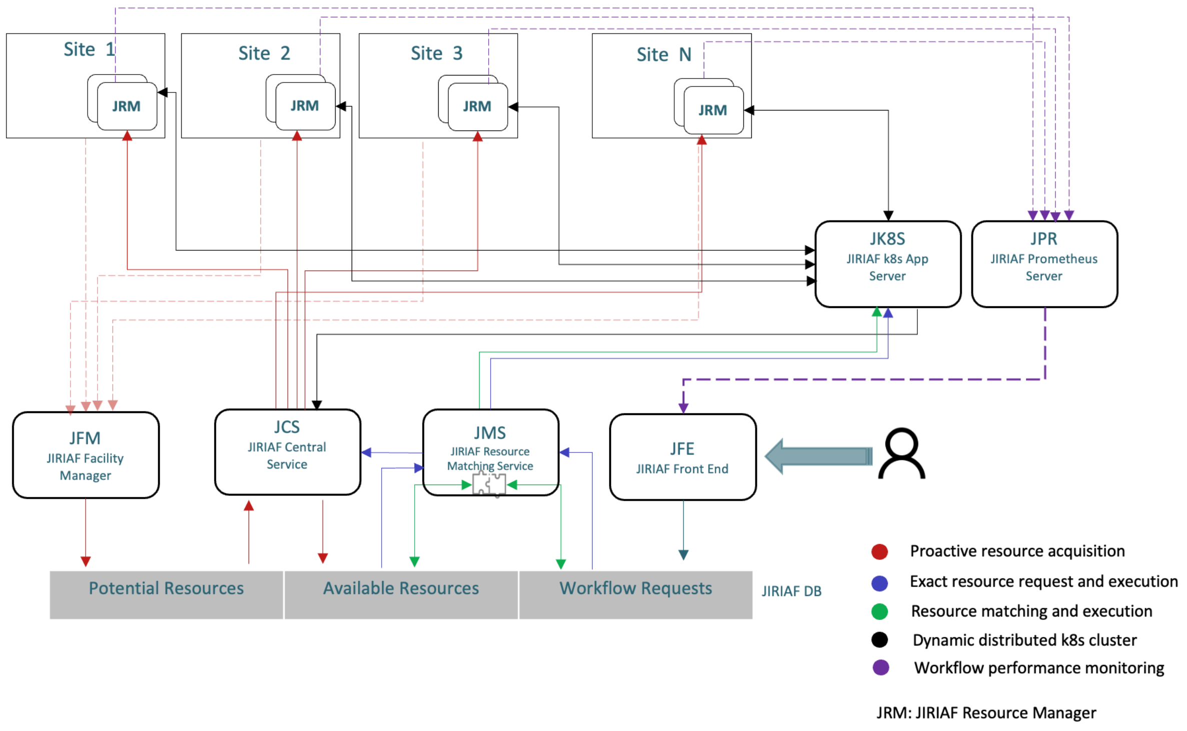
Site 1
Site 2
Site 3
Site N
JRM
JK8S
JIRIAF k8s App Server
JPR
JIRIAF Prometheus Server
JFM
JIRIAF Facility Manager
JCS
JIRIAF Central Service
JMS
JIRIAF Resource Matching Service
JFE
JIRIAF Front End
Potential Resources
Available Resources
Workflow Requests
JIRIAF DB
● Proactive resource acquisition
● Exact resource request and execution
● Resource matching and execution
● Dynamic distributed k8s cluster
● Workflow performance monitoring
JRM: JIRIAF Resource Manager

Fig 1. JIRIAF architecture

Implementing the JIRIAF resource manager (JRM) leveraging Kubernetes (k8s) presents a streamlined approach for orchestrating user workflows in distributed, opportunistic environments. By running the K8s node in user space, we bypass deploying a full-fledged Kubernetes cluster. This lightweight approach significantly simplifies the deployment process and reduces overheads. With the K8s node operating in user space, it registers the machine as a K8s worker node, making it a part of the dynamic Kubernetes cluster orchestrated by the JIRIAF core component (JK8S). Consequently, JRM is granted the capability to execute user workflow containers across these dynamically incorporated nodes. This approach not only maximizes the utilization of distributed resources but also benefits from the inherent scalability, fault tolerance, and extensibility that Kubernetes provides. It harmoniously fuses the JIRIAF framework's robustness with Kubernetes's flexibility, delivering an agile platform for efficient resource management and task orchestration in volatile computing environments.

3. Development and prototyping

At the current stage of the project development, we are concentrating on two subprojects. The first project, dynamic evaluation and effective utilization of temporarily unallocated or idled compute resources from multiple providers, seeks to optimize resource allocation by harnessing the untapped potential of idle compute resources. This project aims to dynamically allocate and distribute computational tasks to underutilized resources across various providers by leveraging advanced algorithms and real-time monitoring. This not only enhances overall efficiency but also reduces energy consumption and costs.
The second project, time-critical workflow migration and data stream processing from JLAB to NERSC without file migration and temporal data persistency focuses on developing innovative methods for seamless workflow migration and data stream processing between the Jefferson Lab (JLAB) and the National Energy Research Scientific Computing Center (NERSC). This project strives to eliminate the need for traditional file migration and temporal data persistency by employing cutting-edge techniques such as in-transit data processing and just-in-time data provisioning. As a result,

this approach ensures uninterrupted data processing and analysis while minimizing latency and improving scientific productivity.

3.1. Workflow colocation and utilization of idled resources in a server

We can use unused resource capacity to run a batch job on the same server to increase resource utilization. These unused, idled resources are typical in almost all computing facilities primarily due to user overestimation of resources for a job. Even if resource estimates are relatively accurate, they are estimates for resource provisioning to handle peak load demands. Since applications rarely experience peak loads for long periods, we can run multiple tasks or processes on a computing farm server. Unfortunately, interference in shared resources, such as CPU caches and memory access paths for threads of different tasks, is almost inevitable, negatively affecting the performance of applications. As a result, users experience increased wall times, resulting in inefficient resource utilization. In JIRIAF, we categorize two tasks: a) local, mission-critical, and b) guest, opportunistic tasks. Guest tasks are JIRIAF-owned jobs. Since computing facility clusters have many similar local tasks on a large scale, it is possible to use a statistical approach to detect performance degradation of mission-critical tasks and address them by reducing interference from guest tasks. In computing, performance is critical in determining how well an application or system performs a given task.
One of the metrics used to measure application performance is the CPU cycles per instruction (CPI) [1]. CPI measures the average number of clock cycles required to execute an instruction on a processor. It is an important metric that reflects the processor's efficiency in managing instructions. A lower CPI indicates that the processor is more efficient in executing instructions, while a higher CPI suggests that the processor is less efficient. The CPI metric is calculated by dividing the number of clock cycles required to complete a set of instructions by the total number of instructions executed. So, CPI indicates the average number of clock cycles needed to execute each instruction. While CPI helps understand the overall efficiency of a CPU, it cannot be used to directly detect interference between two collocated processes on the same server. There are several reasons for this limitation [2]. The CPI metric measures the efficiency of the CPU, but it does not directly account for other shared resources like memory, I/O, and cache. Interference between collocated processes often occurs due to contention for these shared resources, resulting in performance degradation. CPI, on its own, cannot identify the root cause of the contention or which processes are involved. Modern processors have a hierarchy of caches (L1, L2, and L3) to minimize latency for frequently accessed data. Interference between processes can happen at different levels of this cache hierarchy.

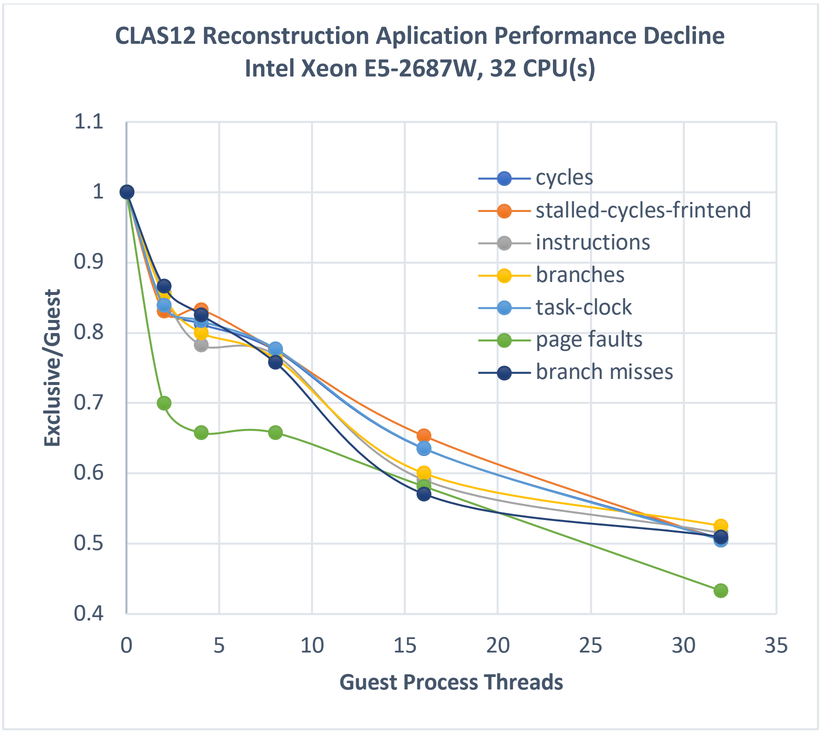

Fig 2. Interference between a multithreaded local application and a guest application on the same server.

As a metric, CPI does not provide granular information about cache behavior, making it inadequate to pinpoint the specific cache level where the interference occurs. Modern CPUs often support multithreading and parallelism to improve performance. This means multiple threads or processes can be executed concurrently on the same processor. The CPI metric measures the average performance across all the threads, masking the potential interference between specific threads or processes running concurrently. In most operating systems, processes are isolated, unaware of other processes running on the same server. As a result, it can be challenging to identify interference between collocated processes using only CPI data. Other tools and metrics, such as cache miss rates, memory bandwidth usage, and I/O throughput, are often required to diagnose interference issues. We need to rely on a combination of performance metrics, monitoring tools, and profiling techniques to detect interference between two collocated processes on the same server. JRM inspects these data to identify and mitigate the interference between collocated processes. The interference between the CLAS12 multithreaded reconstruction program and a guest application on the same server while both applications are demanding and occupying resources is depicted in Figure 2. The graphic demonstrates a decrease, expressed as a percentage, in several metrics that characterize the reconstruction program. These metrics include CPU cycles, branches, the number of instructions retired, branch misses, etc. They are susceptible and can be utilized to identify interference caused by guest processes on the performance of the mission-critical application. The pilot JIRIAF job (JRM) will continuously analyze hardware metrics over time and react to any guest-task correlated changes in the metrics streams. This will allow the pilot JRM to take action on any harmful interference caused by a JIRIAF job running on a remote cluster server.

3.2 Detection and mitigation of job interference in multi-tenant systems through hardware metric monitoring using ADWIN

In high-performance computing environments, particularly those supporting multi-tenant workloads, ensuring the unhindered execution of local, mission-critical jobs becomes paramount. A pertinent issue arises when there is an overlap or interference between the local jobs and guest jobs, which can potentially degrade the performance and output quality of the mission-critical processes. This interference often manifests as noticeable fluctuations in hardware metrics associated with the local job processes. To continuously ensure optimal performance, we have instigated vigilant monitoring of several hardware metrics, which provide real-time insights into the performance of the local job. Should any significant drift or variation in these metric values be detected, indicating potential interference from guest jobs, immediate corrective measures are initiated.

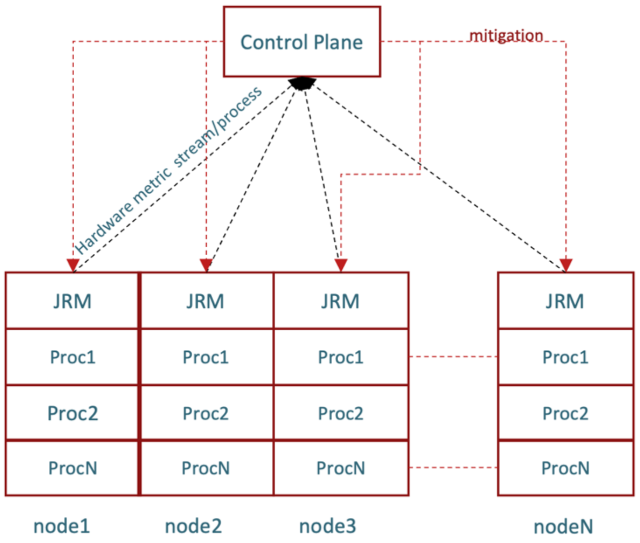

Fig 3. Hardware metric anomaly detection and job interference mitigation.

For the pivotal task of drift detection, we employ the ADWIN (Adaptive Sliding Window Algorithm) unsupervised machine learning technique [3]. ADWIN's dynamic and adaptive sliding window approach makes it skillful at identifying subtle shifts in data streams, in this case, the monitored hardware metrics, by comparing averages over different window sizes. Mitigation strategies are swiftly deployed to alleviate the interference upon successfully detecting a drift. These mitigation approaches range from CPU hard-capping [4], which directly restricts the maximum CPU utilization for a process, to sophisticated container management techniques. For instance, we might enforce a reduction in resources allocated to the interfering guest job, a process known as vertical scaling. Alternatively, workflow migrations can be triggered, moving the guest job to a different computational resource or environment that won't conflict with mission-critical jobs. This combined approach of vigilant monitoring, robust drift detection via ADWIN, and versatile interference mitigation ensures that mission-critical jobs retain their integrity and performance even in dense, multi-tenant computational settings.

3.3 Remote data stream processing

This high-precision, high-throughput experiment requires advanced data processing techniques to handle the vast amount of data generated during its operations. To address this challenge, a collaboration between the Jefferson Lab (JLAB), Energy Sciences Network (ESnet), and the National Energy Research Scientific Computing Center (NERSC) has been established to develop and implement an efficient data-stream processing system.
The data-stream processing system involves real-time data streaming from JLAB to the NERSC Perlmutter High-Performance Computing (HPC) cluster via the EJFAT (ESnet-JLAB Fast Analysis Transport) load balancer at ESnet [5]. The EJFAT load balancer ensures a seamless and efficient data transfer between JLAB and NERSC, minimizing latency and maximizing throughput.

The CLAS12 experiment event reconstruction services [6] were ported into the ERSAP (Event Reconstruction Services and Analysis Platform) framework to build a data-stream processing workflow. ERSAP is a scalable, flexible, and modular platform designed to process large volumes of data in high-energy physics experiments [7].
The ERSAP-based event reconstruction workflow will be migrated and deployed using JIRIAF to the NERSC Perlmutter HPC cluster as part of the collaboration. Real-time data processing can commence once the ERSAP-based event reconstruction workflow has been successfully deployed on the NERSC Perlmutter HPC cluster.

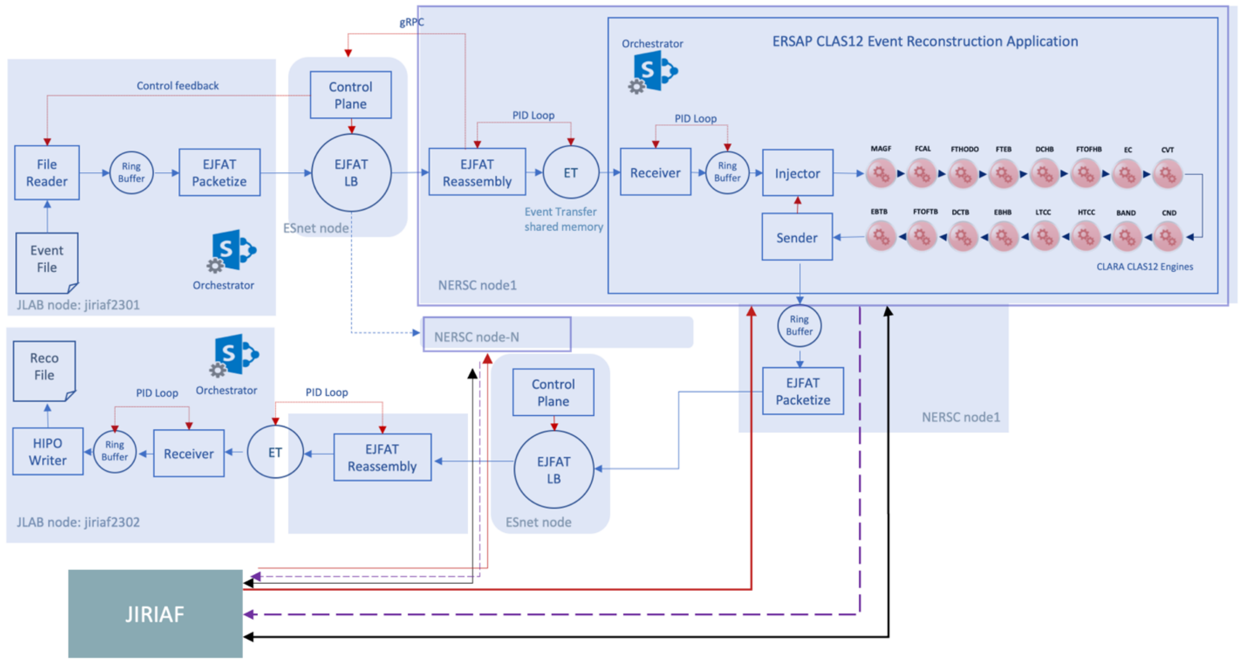

Fig 4. Real-time CLAS12 data-stream processing from JLAB to the NERSC

After each event reconstruction, the results will be immediately streamed back to JLAB through a second ESnet FPGA (Field-Programmable Gate Array) based load balancer. This efficient, real-time streaming approach eliminates the need for temporal data persistency and traditional file transfers, significantly reducing data storage and transfer overheads.
The collaboration between JLAB, ESnet, and NERSC will be an innovative solution for handling the massive data-stream processing requirements of the experiments. The partnership will enable real-time, efficient data processing and result streaming by leveraging the capabilities of the DOE HPC clusters, ERSAP-based event reconstruction workflows, and the EJFAT and FPGA-based load balancers. The findings of this implementation will educate us on how to increase the scientific output and productivity of ongoing scientific experiments.

4. Summary

The JIRIAF project's primary objective is to create an integrated computing infrastructure that seamlessly combines geographically dispersed computing facilities. This is achieved by consolidating diverse computer hardware and software to offer users a unified environment. Such integration fosters enhanced data sharing across different locations, cost savings, heightened security, improved service quality, and minimized downtime. This project is an innovative venture aiming to optimize computing resources, bolster real-time data processing, and facilitate workflow migrations. Collaborative efforts between entities such as JLAB, ESnet, and NERSC promise significant strides in handling expansive data-stream processing requirements for high-throughput scientific experiments.

5. Literature

1. Stijn Eyerman ,K. Hoste, and Lieven Eeckhout. Mechanistic-empirical processor performance modeling for constructing CPI stacks on real hardware. In International Symposium on Performance Analysis of Systems and Software, pages 216–226. 2011.
2. Alaa R. Alameldeen and David A. Wood. IPC is considered harmful for multiprocessor workloads. IEEE Micro, pages 8–17, 2006.
3. Albert Bifet and Ricard Gavaldà. 2007. Learning from Time-Changing Data with Adaptive Windowing. In Proceedings of the Seventh SIAM International Conference on Data Mining (SDM '07). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 443–448.
4. P. Lama, S. Wang, X. Zhou and D. Cheng, "Performance Isolation of Data-Intensive Scale-out Applications in a Multi-tenant Cloud," 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Vancouver, BC, Canada, 2018, pp. 85-94, doi: 10.1109/IPDPS.2018.00019.
5. M. Goodrich *et al*., "ESnet / JLab FPGA Accelerated Transport," in IEEE Transactions on Nuclear Science, doi: 10.1109/TNS.2023.3243871.
6. V. Ziegler et al., "The CLAS12 software framework and event reconstruction," Nuclear Instruments and Methods in Physics Research Section A, Volume 959, 11 April 2020
7. V. Gyurjyan et al., "ERSAP: Towards Better NP Data-Stream Analytics with Flow-Based Programming," in IEEE Transactions on Nuclear Science, doi: 10.1109/TNS.2023.3242548.