Optimizing Resource Provisioning Across Diverse Computing Facilities with Virtual Kubelet Integration

Vardan Gyurjyan, Graham Heyes, Christopher Larrieu, David Lawrence, Jeng-Yuan Tsai

Thomas Jefferson National Accelerator Facility, Newport News, VA, USA

E-mail: {gurjyan, heyes, larrieu, davidl, tsai}@jlab.org

Abstract. The Jefferson Lab Integrated Research Infrastructure Across Facilities (JIRIAF) project addresses the critical challenges of managing large-scale distributed computing environments. This initiative presents the architecture and core components of JIRIAF, emphasizing its capability to efficiently migrate and scale workloads across multiple sites, utilize opportunistic resources, and maintain system integrity in user space. Central to JIRIAF's architecture is the JIRIAF Resource Manager (JRM), which employs a Virtual Kubelet to leverage Kubernetes in environments lacking root access. The proof of concept demonstrates JIRIAF's effectiveness through the deployment of data-stream processing pipelines on the Perlmutter system at NERSC, utilizing the CLAS12 event reconstruction application. Additionally, we simulated a queue system using a digital twin model to demonstrate the potential for enhancing real-time monitoring and control capabilities with a Dynamic Bayesian Network (DBN). The results highlight JIRIAF's robust framework for optimizing resource allocation and improving computational efficiency across heterogeneous high-performance computing environments.

1 Introduction

The Jefferson Lab Integrated Research Infrastructure Across Facilities (JIRIAF) project aims to streamline the management of large-scale distributed infrastructures. This initiative addresses several critical challenges faced in modern high-performance computing environments, including the efficient migration and scaling of workloads across multiple computing sites, the intelligent utilization of opportunistic resources to enhance overall efficiency, and the maintenance of system integrity while operating in user space. By leveraging advanced architectural designs and state-of-the-art technologies, JIRIAF provides a robust framework for resource management and computational efficiency. This document outlines the motivation behind JIRIAF, delves into its sophisticated architecture, highlights the core components such as the JIRIAF Resource Manager (JRM), and presents proof-of-concept implementations demonstrating its efficacy. Additionally, the integration of a digital twin model for simulated stream processing showcases the innovative approaches employed to optimize computational resource allocation in high-throughput systems.

2 Motivation

The primary motivation behind JIRIAF is to streamline the management of large-scale distributed infrastructures, addressing key challenges such as efficiently migrating and scaling workloads across multiple computing sites, intelligently utilizing opportunistic resources to enhance overall efficiency, and maintaining system integrity while operating in user space. A distinguishing highlight of JIRIAF is its ability

to utilize the Kubernetes framework [1] to orchestrate diverse HPC resources without requiring special administrative approvals or root-level access. This contrasts with traditional systems such as HTCondor [2], OpenStack [3], Apache Mesos [4], and Slurm [5], which typically require administrative privileges for effective deployment and operation. These constraints often limit their applicability in environments where users lack control over infrastructure configurations. JIRIAF overcomes these limitations by employing the Virtual Kubelet [6], which enables rootless Kubernetes-like functionality, allowing users to seamlessly execute containerized workflows across heterogeneous HPC facilities. By leveraging these unique innovations, JIRIAF ensures accessibility and user independence from administrative restrictions, making it a robust solution for modern distributed computing challenges.

3 Architecture

The JIRIAF architecture [7] is meticulously designed to enable seamless integration and efficient resource management across diverse computing facilities. At the heart of this system is the JFM (JIRIAF Facility Manager), responsible for maintaining a dynamic resource pool by periodically scraping data from each computing facility to ensure an up-to-date inventory of available resources. The JCS (JIRIAF Central Service) functions as the central command, initiating pilot jobs through the JRM (JIRIAF Resource Manager). The JRM, which can operate in userspace to accommodate heterogeneous HPC setups, leases resources reported by the JFM, awaiting utilization. The JMS (JIRIAF Matching Service Algorithm) then steps in to update the available resource database, aligning resources with user requests. The JFE (JIRIAF Front End) finalizes the process by managing user requests and populating the user workflow request table. This comprehensive architecture is depicted in Figure 1, illustrating JIRIAF's commitment to providing a seamless and efficient computing environment.

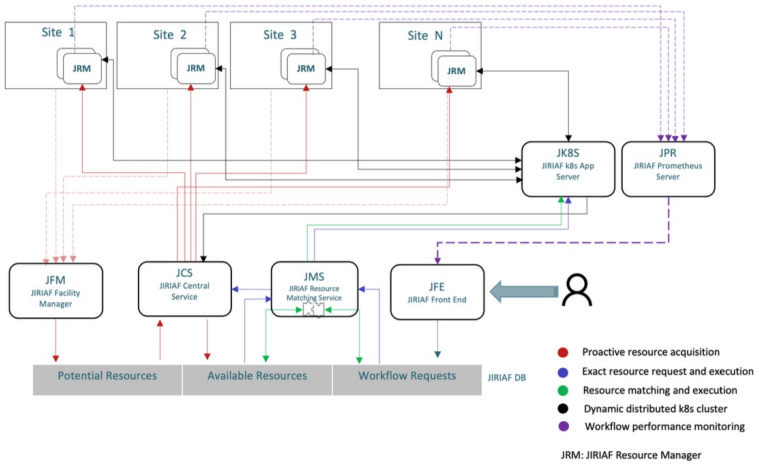

Figure 1: JIRIAF System Architecture and Workflow: This figure visually represents the sophisticated architecture of JIRIAF, emphasizing the roles and interconnectivity of its key components — the JFM, JCS, JRM, JMS, and JFE. By illustrating the flow of data and control across these components, the diagram elucidates the dynamic and efficient resource management system designed to optimize high-performance computing across heterogeneous environments.

4 Core Component - JIRIAF Resource Manager

JRM serves as an integral component of the framework, effectively leveraging the Kubernetes framework with Virtual Kubelet [6, 8]. A fundamental Kubernetes cluster comprises a master node/control plane that oversees cluster management and worker nodes with kubelets connected to the containerd socket for container execution. Given that installing a regular kubelet necessitates root credentials, which are typically unavailable to ordinary users at compute sites, we employ the Virtual Kubelet to circumvent this limitation while still utilizing the Kubernetes framework. The Virtual-Kubelet-Cmd (VK) is a virtual kubelet implemented using BASH commands and operates in userspace. It translates a container into a BASH script composed of several processes. This approach serving as JRM allows JIRIAF to execute

user applications as containers across various computing sites by simply running BASH commands in userspace, all the while ensuring unified control and monitoring through Kubernetes.

5 Proof of Concept

A 40-node reservation on the Perlmutter system at NERSC was activated to deploy data-stream processing pipelines. This deployment utilized the JIRIAF framework across the JIRIAF Kubernetes cluster nodes, each executing the CLAS12 event reconstruction application. This application was optimized to fully leverage all available processing cores within the ERSAP framework [9] (see Figure 2).

To demonstrate the effectiveness of JIRIAF, a proof of concept was conducted using the CLAS12 experiment [10]. Event streams were transmitted to the NERSC computing facility via the EJFAT transport system. JRMs/VKs were deployed on 40 nodes within the NERSC cluster for stream processing workflows. The ERSAP workflow was deployed for CLAS12 reconstruction.

JRMs/VKs of JIRIAF as agents were deployed by the SLURM batch job system at NERSC. These JRMs formed K8s nodes waiting for deployments. The ERSAP processing application was containerized and uploaded to the Shifter container hub at NERSC. A Kubernetes deployment applied to the Kubernetes API server on the control-plane at JLAB. The monitoring system scraped and stored metrics data on the control-plane at JLAB as shown in Figure 3.

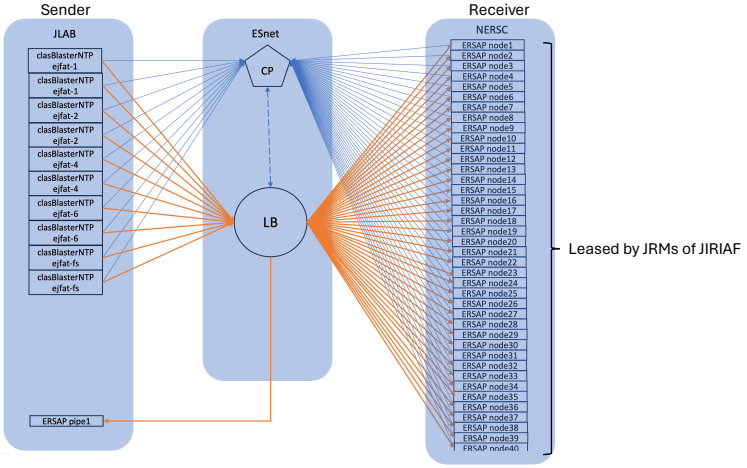

Figure 2: The ERSAP framework utilized in the JIRIAF deployment on the Perlmutter compute nodes at NERSC. The CLAS12 event reconstruction application ran on each node in the JIRIAF Kubernetes cluster, demonstrating the JIRIAF deployment's effectiveness in handling high-volume data-stream processing.

6 Digital Twin for Simulated Stream Processing System

In the broader context of our study on optimizing computational resource allocation in high-throughput systems, we simulated a queue system using a digital twin model to demonstrate the potential for enhancing real-time monitoring and control capabilities with a Dynamic Bayesian Network (DBN) [11]. The digital twin component leverages a DBN to simulate the behavior of a queue system, providing valuable insights into system dynamics and aiding in decision-making processes. We utilized the code from [12] to build our DBN model, demonstrating the practical application of their proposed framework.

6.1 Digital Twin Model and Methodology

The digital twin model was developed to mirror the state and behavior of a physical queue system, comprising a stream sender and receiver with a FIFO queue. The DBN framework was employed to capture dependencies among system variables, offering a probabilistic approach to real-time data assimilation and state estimation. Our experimental setup involved adjusting the event sending rates (λ) and measuring the resulting processing rates (μ) and observed queue lengths (Obs. L_q) under different computational

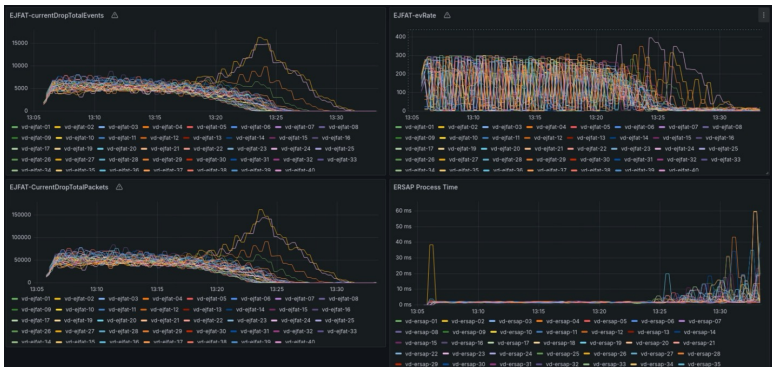

Figure 3: Monitoring system metrics scraped from applications during the JIRIAF deployment. The figure shows the metrics collected by the monitoring system, providing insights into the performance and resource utilization of the deployed CLAS12 event reconstruction application across the NERSC cluster nodes. These metrics are crucial for evaluating the effectiveness and efficiency of the JIRIAF framework in a high-performance computing environment.

capacities (16 and 32 threads). The theoretical queue length (Calc. L_q) was calculated using the M/M/1 queue theory equation:

$$L_q = \frac{\lambda^2}{\mu(\mu - \lambda)} \tag{1}$$

The data collected from these experiments were used to construct and validate the DBN model, enabling it to make accurate state predictions and recommend optimal control actions. The DBN structure is depicted in Figure 4, illustrating the relationships between the digital twin state ($D(t)$), control ($U(t)$), and observation ($O(t)$) nodes.

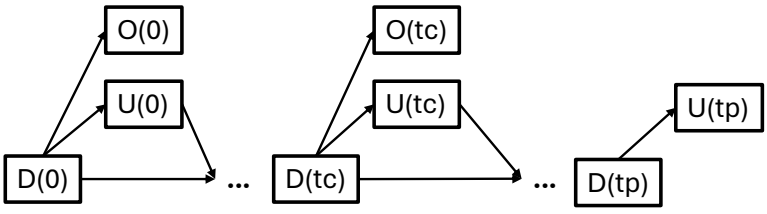

Figure 4: Dynamic Bayesian Network representation of the digital twin. It consists of the nodes D(t), O(t), and U(t).

6.2 States Evolving Over Time

The digital twin's state evolves as new observations o_t are assimilated over time as shown in Figure 5. The digital twin accurately tracks the ground truth state during periods of increasing queue lengths but exhibits a delay during periods of decreasing queue lengths. As time progresses, particularly beyond the 80-time unit mark, a noticeable variation in the predicted state occurs, indicated by the red shaded area. Overall, the digital twin demonstrates strong performance in aligning with the observed data for most of the time period.

7 Acknowledgements

This project is funded through the Thomas Jefferson National Accelerator Facility LDRD program. This material is based upon work supported by the U.S. Department of Energy Office of Science Office of Nuclear Physics under contract DE-AC05-06OR23177.

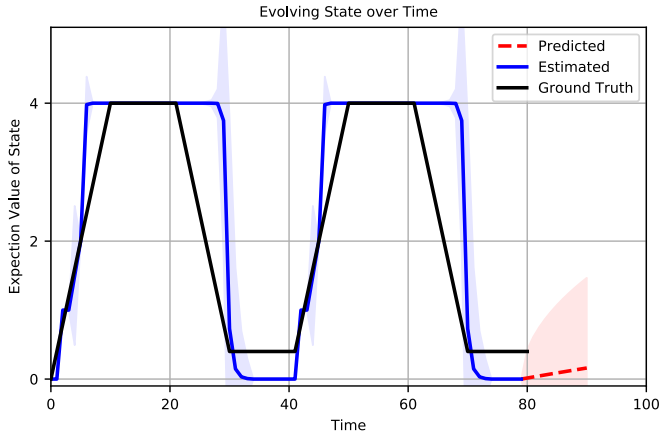

Figure 5: Evolving State Over Time. The plot shows how the digital twin assimilates observed data and estimates the state over time. The black line represents the ground truth, the blue line indicates the estimated state, and the red dashed line shows the predicted state. The red and blue shaded areas represent the variation of estimated and predicted states, respectively.

References

[1] Production-grade container orchestration. `https://kubernetes.io`. Accessed: 2024-07-14.

[2] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: The condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356, 2005.

[3] Openstack: The open source cloud computing software. `https://www.openstack.org`. Accessed: 2024-07-14.

[4] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *NSDI*, volume 11, pages 295–308, 2011.

[5] Moe Jette, Andy Yoo, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

[6] Virtual Kubelet. Virtual kubelet project. `https://github.com/virtual-kubelet/virtual-kubelet`. Accessed: 2024-07-14.

[7] Gyurjyan Vardan, Larrieu Christopher, Heyes Graham, and Lawrence David. Jiriaf: Jlab integrated research infrastructure across facilities. In *EPJ Web of Conferences*, volume 295, page 04027. EDP Sciences, 2024.

[8] Jefferson Lab. Jiriaf 0.1. `https://github.com/JeffersonLab/jiriaf-0.1`. Accessed: 2024-07-14.

[9] Gyurjyan Vardan, Abbott David, Goodrich Michael, Heyes Graham, Jastrzembski Ed, Lawrence David, Raydo Benjamin, and Timmer Carl. Streaming readout and data-stream processing with ersap. In *EPJ Web of Conferences*, volume 295, page 02025. EDP Sciences, 2024.

[10] S Boyarinov, B Raydo, C Cuevas, C Dickover, H Dong, G Heyes, D Abbott, W Gu, V Gyurjyan, E Jastrzembski, et al. The clas12 data acquisition system. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 966:163698, 2020.

[11] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. University of California, Berkeley, 2002.

[12] Michael G Kapteyn, Jacob VR Pretorius, and Karen E Willcox. A probabilistic graphical model foundation for enabling predictive digital twins at scale. *Nature Computational Science*, 1(5):337–347, 2021.