Intelligent Operating System Deployment Strategy Based on KiERA for Optimizing Supercomputers

Seungwoo Rho
Div. of National Supercomputing
Korea Institute of Science and Technology Information
Daejeon, South Korea
seungwoo0926@kisti.re.kr

Jinseung Ryu
Div. of National Supercomputing
Korea Institute of Science and Technology Information
Daejeon, South Korea
jsu@kisti.re.kr

Kimoon Jeong
Div. of National Supercomputing
Korea Institute of Science and Technology Information
Daejeon, South Korea
kmjeong@kisti.re.kr

***Abstract*—Supercomputers are essential tools in modern scientific and industrial research, performing complex computations and large-scale data processing swiftly and accurately through high-performance computing (HPC). To maximize the performance of these supercomputers and ensure stable operations, efficient operating system deployment and real-time resource management are crucial. However, current commercial cluster management tools rely on foreign software, leading to increased costs in procurement and maintenance. To address this issue, the Korea Institute of Science and Technology Information (KISTI) implemented its proprietary intelligent cluster management system, KiERA, to enhance the operating system deployment performance for its fifth supercomputer, Nurion. This study focused on adapting the existing proprietary operating system, initially used exclusively for Nurion, to work effectively with KiERA. The performance of KiERA's operating system deployment was evaluated on 169 compute nodes and compared to the commercial software, Bright Cluster Manager (BCM). Experimental results demonstrated that KiERA completed node deployments in approximately 430 s on average, showcasing improved deployment efficiency over the existing method. Furthermore, the study suggests the potential for deploying up to 13,000 nodes simultaneously in large-scale supercomputer environments. This research contributes to improving supercomputer operational efficiency while reducing dependency on foreign software.**

Keywords—provisioning, supercomputer, cluster management, operating system, deployment

I. INTRODUCTION

Supercomputers have become indispensable tools in modern scientific research and various industries, particularly for rapidly and accurately processing large-scale data and performing complex calculations. They are widely used in climate change prediction, genetic analysis, drug development, and astrophysical simulations. High-performance computing (HPC) is crucial in significantly accelerating research and development. With advancements in cutting-edge technologies, such as artificial intelligence, big data analysis, and machine learning, the importance of supercomputers continues to grow [1].

To maximize the performance of supercomputers and ensure stable operation, an efficient operating system deployment, real-time monitoring, and resource management for large-scale nodes are essential [2]. Cluster management systems are crucial for handling these tasks intelligently. These systems detect the status of nodes in real time, rapidly deploy operating systems, and proactively identify potential issues to ensure stable system operations. However, most commercial cluster management tools are foreign software, and certain technical limitations to their application exist in domestic system environments optimized for local needs. Additionally, these foreign software solutions incur increased initial deployment, maintenance, and license costs over time.

For instance, the Korea Institute of Science and Technology Information's (KISTI) fifth supercomputer, known as "Nurion," manages its 8,437 compute nodes using the commercial software Bright Cluster Manager (BCM) [3]. This software is used for tasks such as operating system deployment and maintenance. However, BCM operates under a licensing model that incurs costs per node. As the number of nodes increases, the associated financial burden rises. Moreover, the license should be renewed annually, further augmenting ongoing maintenance costs.

To address this issue, KISTI developed its intelligent cluster management system (KiERA) in 2019 [4]. KiERA offers features such as remote hardware-monitoring tools and the ability to deploy operating systems across large-scale nodes quickly. It aims to reduce reliance on commercial software such as BCM. This study focuses on applying KiERA to 169 computing nodes of the Nurion supercomputer and compares its performance in deploying operating systems with that of the existing BCM solution. The aim is to evaluate the efficiency of operating system deployment and suggest ways to improve the performance compared with traditional deployment methods.

These research outcomes are expected to help the KISTI implement an optimized next-generation cluster management system tailored to the needs of domestic research institutions. Additionally, it aims to reduce the reliance on foreign software, lower long-term maintenance costs, and enhance the stability of system operations.

II. RELATED WORK

A. *Intelligent Cluster Operation Management System (KiERA)*

KiERA is a web-based intelligent cluster operation management system developed to efficiently configure and manage internal clusters along with large-scale supercomputers introduced and operated by KISTI. KiERA is based on the Python Django Web framework and consists of four main services: Celery, Daphne, SoL-Broker, and uWSGI. Internally, it provides web services based on Nginx and uses RabbitMQ message queues and MariaDB for data processing

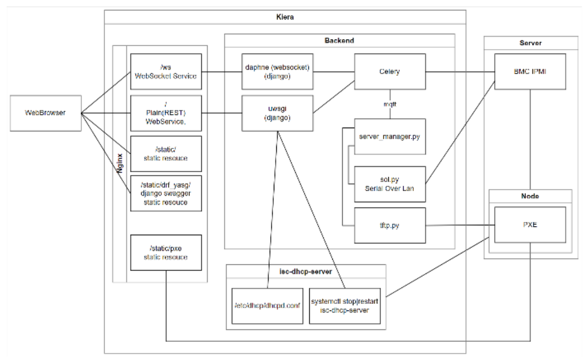

Fig. 1. Overall structure of KiERA

and management. Fig. 1 shows the overall structure of KiERA.

When users connect to the KiERA system through a Web browser, they can monitor the cluster status or execute commands. The frontend operates based on Django and provides static resources and user interface elements. WebSocket handles real-time communication services, enabling users to monitor cluster status in real time or send commands. It also offers HTTP-based REST APIs for executing cluster management commands or accessing resources. In the Fig. 1, Daphne handles real-time communication through WebSocket, whereas all other static services are processed by uWSGI. Celery functions as a scheduler, executing distributed and asynchronous tasks and scheduling all tasks received from Daphne and uWSGI.

The key features of KiERA include node management, group management, IPMI command execution for nodes/groups, cluster management, script management, image management, subnet management, preinstallation environment management, IPMI SoL scan functionality, and KiERA service management. The preinstallation environment used in KiERA (based on Ubuntu 22.04) supports the running of a lightweight operating system in memory to collect hardware information from nodes or install the actual operating system on the disk before deploying the actual nodes. This fixed preinstallation environment [5] allows for the easy support of various operating systems, hardware, and software, enabling quick handling of requirements and automation processes from a system maintenance perspective.

B. *Supercomputer No. 5: Nurion*

Supercomputer No. 5, Nurion (NURION) [6], is a Linux-based massively parallel cluster system with a theoretical peak performance of 25.7 Pflops, making it a high-performance computer. It consists of 8,437 compute nodes, including 8,305 compute nodes with many-core Knights Landing (KNL) CPUs and 132 CPU-only nodes. Nurion can also handle large-scale I/O requests through its 100G-based Intel Omni Path Architecture (OPA) high-performance interconnects and burst buffers. Fig. 2 illustrates the Ethernet network configuration of Nurion's computing nodes [7].

As illustrated in the Fig. 2, four 10G Ethernet switches were stacked to operate as a single switch and configured in a high-availability setup. The ports of the 10G Ethernet were connected to 1G Ethernet switches (edge switches) located in each rack and BCM management nodes.

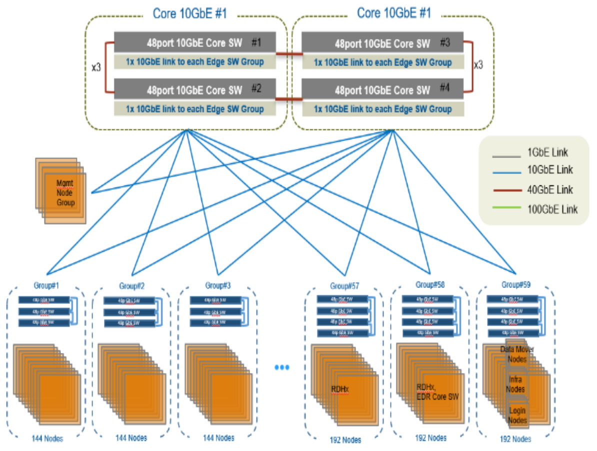

Fig. 2. Nurion Ethernet network configuration

Currently, the cluster operation management system used in Nurion is the BCM v8, with two main servers responsible for its operation. The BCM manages the deployment operating system on a file basis, and the rsync command provided by Linux is used to transfer files during OS deployment. The OS deployment process in Nurion is as follows: first, the two BCM main servers are deployed to approximately 130 CPU nodes through the BCM relay servers. Each relay server then deploys diskless RHEL 7.9 BCM-specific operating system files (approximately 5.1 GB) to the remaining 8,305 KNL computing nodes. The entire system deployment, including the CPU relay servers and KNL computing nodes, requires approximately 2 h in the BCM. The actual deployment of the KNL compute nodes requires approximately 50 min, with the remaining time spent deploying the relay servers accompanying the OS images used by the relay servers. When the relay servers complete the deployment to the compute nodes, they begin to perform the same role as the compute nodes.

III. Experiment

A. *Experimental Environment*

The experiment conducted in this study was designed to evaluate the deployment performance of a single KiERA server (ProLiant DL380 Gen10 Skylake, Intel Xeon Gold 6152 2.1G, 44 CPUs, 192 GB RAM) by configuring a KNL VLAN 169 node. The deployment network consisted of 1G Ethernet and 100G OPA, where Ethernet was used to download the TFTP-based grub bootloader (716 KB), HTTP-based initrd (approximately 110 MB), and kernel (approximately 6.8 MB). The OPA network was used to download large rootfs via HTTP. The deployed operating system used the RHEL 7.9 BCM-specific image that is currently in use. To use the BCM-specific images, the initial temporary root file system (initrd) and kernel were replaced with official files. The initrd was modified to include the OPA driver, enabling the downloading of the rootfs. Furthermore, the rootfs were converted from an individual file-based system to a single SquashFS image, thereby reducing the size to approximately 2 GB. Table I compares the rootfs file transfer methods of BCM and KiERA and summarizes the actual performance differences in file transfers.

TABLE I. COMPARISON OF ROOTFS TRANSFER METHODS BETWEEN BCM AND KIERA

	BCM	KiERA
Rootfs Transfer Method	rsync	http (wget)
Operating System Format	Individual files	squshfs
Rootfs Size (GB)	5.1	2
OPA-based rootfs Transfer Time (s)	214	26

As presented in Table 2, KiERA reduced the rootfs file size by more than 60% by converting the rootfs from an individual file format to a SquashFS-based compressed image format, thus reducing the number of files to one. This resulted in a significant reduction in the transfer time of OPA-based rootfs compared to that of BCM. However, because SquashFS is a read-only file system, KiERA combines it with an additional OverlayFS to provide a read–write environment.

B. Initialization Script for initrd

The operating system of Supercomputer No. 5, Nurion, operates on a diskless basis and runs optimized initialization scripts using the BCM. However, these initialization scripts are BCM-specific files and can only be used in clusters managed by the BCM. Therefore, separate modifications are required for their use in other cluster management tools. Therefore, this study created custom initialization scripts for a diskless operating system running on Nurion and applied them to KiERA. The officially supported initrd and kernel version (3.10.0-1160.119) were used for modifying the initrd. The primary purpose of this initialization script was to execute the scripts necessary at various stages of initialization, including kernel module loading, network configuration, and mounting of the root file system. The initialization script includes several scripts that execute related functions and processes at each stage. The entire process consisted of six stages with the following detailed steps:

- **Loading Script Sources** - Before subscript is executed, a script that defines the necessary functions and mounts the kernel space is loaded. Each function supports system configuration or recovery tasks during the boot process, such as displaying messages, reading kernel parameters, and providing a BASH shell environment.
- **System Information Output** - The script displays diskless image-related information about the KiERA system at the terminal, enabling users to clearly understand the environment in which they are working.
- **Emergency Shell Invocation** - An emergency shell is opened to allow the user direct access to the system if needed. This provides a shell that can be accessed in urgent situations when issues arise in the system and enables system monitoring through a serial console, if necessary.
- **Kernel Module Loading and Network Configuration** - Necessary kernel modules are loaded to prepare the system for additional use during the boot process, and the network interfaces are reconfigured. Particularly, the OPA driver module (hfl module) is loaded to enable the use of the OPA network. During network configuration, the dhcp client is re-executed to obtain a new IP address.
- **Root File System Download and Mounting** - The root file system is downloaded and mounted to prepare the system for use. In KiERA, the necessary data source files are first downloaded to obtain a root file system. The data source provides the network configuration information, instance metadata, and various environmental variables required for booting. The mounting process involves configuring the root file system using SquashFS and an overlay file system. SquashFS is used as a read-only file system, and OverlayFS is used to provide read–write capabilities. This enables the configuration of a dynamic file system during booting.
- **Service Registration and Linux Initialization** - Required services are registered in the system, and Linux starts to complete the booting process. This includes reading the kernel parameters, configuring systemd services, reporting the complete provisioning status, and setting the hostname. The final steps of the booting process involve dynamically configuring the root file system and correctly mounting the base system directories to establish a chroot environment. The system is then executed to initialize it and start the Linux.

C. Experimental Procedure

The experiment was divided into four stages: discovery, commissioning (preinstallation environment), cluster configuration, and deployment. Each stage is outlined as follows. The experiment conducted in this study aimed to verify the efficiency of KiERA-based operating system deployment and was conducted in the following four stages:

- **Discovery Stage**

1) Board Management Controller (BMC) Network Information Registration: BMC ID, password, IP address, and port information are entered to configure the BMC network. 2) Target Node Scanning: The IPMI SoL scan feature is used to scan the entire BMC network, and the target nodes are temporarily registered. During this process, all temporarily registered nodes can be controlled by IPMI.

- **Commissioning (Preinstallation Environment) Stage**

1) Target node commissioning: Pre-boot eXecution Environment (PXE) boot settings were modified, and reboot commands were executed to initiate the commissioning process. 2) Hardware Information Collection and Registration: Through the commissioning process, hardware information such as the CPU, memory, storage devices, GPUs, and other PCIe devices for each node were collected and registered in the database.

- **Cluster Configuration Stage**

1) Cluster Creation: The cluster name, group name, number of nodes per rack, and node name prefix were entered. The target and head nodes were selected for inclusion in the cluster. 2) SSH Key Management: New SSH keys were generated or

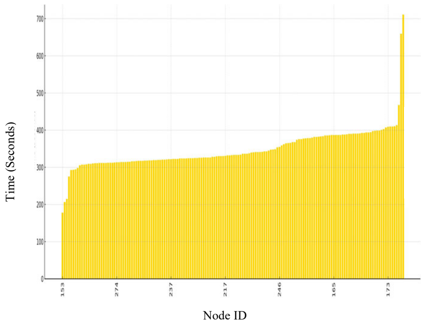

Fig. 3. Initrd download time per node

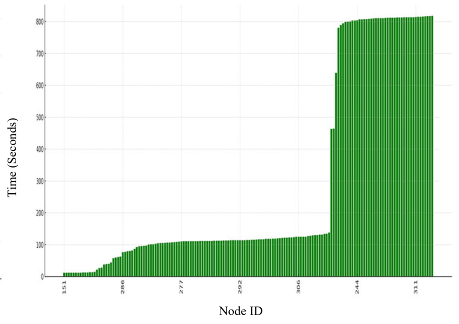

Fig. 4. Rootfs download time per node

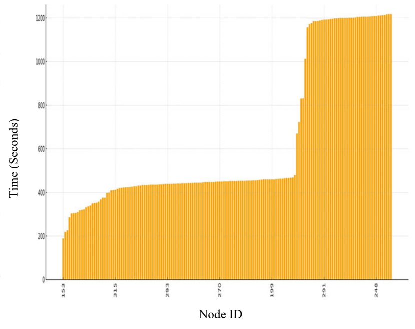

Fig. 5. Deployment completion time per node

existing keys were registered. 3) Network Configuration: Both Ethernet and OPA networks were collectively configured. For the cluster network configuration, when the initial IP address is entered, the remaining nodes are automatically assigned IP addresses sequentially.

- **Deployment Stage**

Image Preparation: We prepared the rootfs based on the RHEL 7.9 BCM-specific image, the modified official initrd, and the kernel. 2) Image Registration and Deployment: The prepared image files were registered in KiERA, and deployment was performed using the designated image. During this stage, the rootfs image compressed in SquashFS format was used, and high-speed transmission was facilitated through the OPA network.

IV. Result and Analysis

The deployment of the operating system to 169 computing nodes using KiERA revealed differences in the download times of the initrd and rootfs files as well as the overall deployment completion time for each node. In Fig. 3, the X-axis represents the node ID, and the Y-axis illustrates the download time. The results indicated that most nodes completed the initrd download within approximately 185 to 400 s. However, a few nodes experienced network bottlenecks, resulting in download times exceeding 450 s. This phenomenon highlights the bandwidth limitations of the Ethernet network when using the grub bootloader to download the initrd, and suggests that simultaneous deployment is feasible for up to 165 nodes. As illustrated in Fig. 4, the rootfs download commenced after the initrd download was completed. Approximately 14 nodes completed the rootfs download within 10 s, whereas the download time gradually increased to approximately 130 s for the 100th node. However, for nodes beyond the 100th node, the download time increased sharply, with some nodes requiring up to 800 s. This finding indicates a bandwidth limitation in the OPA network, where simultaneous deployment is optimal for up to 100 nodes. Fig. 5 illustrates the total deployment completion time, which combines the initrd and rootfs download times. The total deployment time ranged from 189 s to 1,217 s, with the rootfs download time having the most significant impact on the overall completion time. When 100 nodes were deployed simultaneously, the average deployment time was approximately 430 s (7 min and 10 s).

Based on these experimental results, it is estimated that KiERA can simultaneously deploy up to 13,000 computing nodes in a single deployment cycle (approximately 430 s) if 130 relay servers are used. However, this is a theoretical estimate, and further verification in large-scale supercomputer environments is required.

V. Conclusion and Future Work

In this study, we proposed a method for deploying operating systems in supercomputers using KiERA and conducted a performance comparison with the existing BCM system. The results showed that KiERA completed operating system deployment within an average of 7 min and 10 s for 169 nodes, demonstrating a significantly improved deployment efficiency compared with the BCM-based approach. When deploying more than 100 nodes simultaneously, network bottlenecks occur, causing a sharp increase in the deployment time. This issue can be attributed to the bandwidth limitations of OPA networks, necessitating further research to address this limitation. Additionally, the study suggested the potential for KiERA to deploy up to 13,000 nodes simultaneously, which can significantly enhance the operating system deployment efficiency in large-scale supercomputer environments.

Future research will focus on optimizing KiERA performance in large-scale supercomputer environments and exploring solutions to mitigate OPA network bottlenecks. Through these efforts, KiERA is expected to contribute to improving the operational efficiency of supercomputers, both domestically and internationally.

Acknowledgment

This research was supported by Korea Institute of Science and Technology Information (KISTI). (No. (KISTI)K24-L2-M1-C6, (NTIS)2710018524)

References

[1] D. A. Reed and J. Dongarra, 'Reinventing high performance computing: Challenges and opportunities,' *J. High Perform. Comput.*, vol. 12, no. 1, pp. 1–22, 2023.

[2] R. Riesen, B. Gerofi, Y. Ishikawa, and R. W. Wisniewski, "Operating systems for supercomputers and high performance computing," SpringerLink, vol. 1, pp. 1–12, 2018.

[3] Y. Kodama, K. Watanabe, K. Suzuki, and K. Masui, "Bright Cluster Manager: Powerful cluster provisioning," in Proc. ACM/IEEE Int. Conf. High Perform. Comput., 2020, pp. 123–132.

[4] S. Rho, J. Ryu, S. Kim, K. Oh, K. Moon, and M. Yoo, "A study on the technology for configuring online heterogeneous clusters using a remote management platform," in Proc. 2023 Korea Inf. Sci. Soc. Conf., 2023, pp. 31–33.

[5] Canonical Ltd., Canonical MaaS: Bare metal as a service, Canonical Whitepaper, 2020.

[6] Korea Institute of Science and Technology Information, Nurion User Manual, 2018.

[7] J. Browning, B. Winkler, R. Brightwell, M. Levenhagen, L. Archer, and A. Rodrigues, "Intel Omni-Path Architecture: Unveiling the secrets of the next generation InfiniBand," Hot Chips, vol. 30, 2018.