Addressing Resource Constraints on Aurora with Admin Access Nodes

Peter Upton
Ben Lenard
Ben Allen
Cyrus Blackworth
pupton@anl.gov
blenard@anl.gov
bsallen@anl.gov
cblackworth@anl.gov
Argonne National Laboratory
Lemont, Illinois, USA

ABSTRACT

Administrator Access Nodes (AANs) are an alternative to the traditional reliance on a single Administrative Node for all aspects of system administration in an HPE Performance Cluster Manager (HPCM) managed supercomputer cluster. At the Argonne Leadership Computing Facility (ALCF), managing the Aurora supercomputer, one of the largest HPCM managed systems with over 10,000 nodes, requires a team of skilled developers and administrators. These professionals perform tasks such as parsing log files, issuing power commands, and connecting to nodes via SSH. These tasks have typically been performed solely on the Admin Node. However, this centralization can result in resource constraints due to simultaneous resource requirements by multiple administrators. To address these issues, AANs, which include custom tools for interacting with HPCM, scripts to replicate some Admin Node functionality on AANs, and synchronization tools for configuration and authentication files, were implemented. The introduction of AANs has alleviated resource constraints, simplified workflows, and improved system manageability. Further improvements should focus on increased integration with HPCM and improving usability with the goal of enhancing AAN capabilities and administrative efficiency for Aurora's complex environment.

ACM Reference Format:
Peter Upton, Ben Lenard, Ben Allen, and Cyrus Blackworth. 2025. Addressing Resource Constraints on Aurora with Admin Access Nodes. In *Proceedings of HPE Cray User Group (CUG '25).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Administrator Access Nodes (AANs) are a novel solution to the challenges posed by the centralized use of the Virtual Admin Node used by HPE High Performance Cluster Manager (HPCM) to manage supercomputers like Aurora. This concept should be considered in similar situations where decentralization is not possible or practical.

Aurora, housed at the Argonne Leadership Computing Facility (ALCF), is one of the world's first exascale supercomputers. As such, it presents significant management challenges due to its complexity and scale, and requires a team of more than 100 skilled developers and administrators. These professionals rely on many tools and resources to perform tasks such as parsing very large log files, issuing power commands, and connecting to nodes via SSH. These tasks are typically executed on the administrative node, which is a virtual machine that maintains the log files, administrative tools, and node lists.

However, the central dependency on the administrative node leads to resource constraints and system instability due to high demands from multiple admin users at the same time. To address this problem, we introduced AANs, which complement existing node types such as User Access Nodes (UANs), compute nodes, and leader nodes. AANs mitigate resource constraints on the Admin Node by enabling admin users to perform many tasks external to the Admin Node, thus increasing the stability of Aurora and improving the administrator experience.

In Aurora nodes are managed by HPCM, a complex collection of software components packaged as an operating system image based on a Linux distribution. The Admin Node is hosted on a virtual machine, which increases overhead and constrains the computing resources available to it. AANs were introduced to mitigate these issues by providing an alternative platform for administrative tasks. The implementation of AANs includes the development of custom tools and scripts to provide administrators with an operational experience similar to what they have on the Admin Node. These tools allow for interaction with HPCM and ensure that required configuration files are synchronized with the Admin Node.

1.1 Implementation and Features

The implementation and features of AANs include custom tools written to facilitate their integration with HPCM. AANs were required to mitigate the administrative node's resource constraints. Implementation details are also discussed, including HPCM specific APIs and how they are used by the AAN tools. The tools and node images are created and maintained to keep the AANs' configuration

CUG '25, May 04–08, 2025, New York, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

in sync with other node types. There are limitations and possible enhancements to the AANs. These will all be discussed later.

AANs are modeled after UANs and include additional administrative utilities and access to log and configuration files. Custom tools such as `cm-remote`, `cmn`, and `admin-sync` were developed to ease the transition to AANs and enhance their functionality. The features of AANs include available mount points, utilities (such as console, dsh, tmux), files, and integration with HPCM APIs.

The following is an overview of the custom tools included in the AAN.

- `cm-remote` emulates the behavior of the cm command to facilitate administrator transition, with caching and command redirection as needed.
- `cmn` is a thin wrapper for interfacing with HPCM APIs, designed for use in scripts and pipelines.
- `admin-sync` is a simple tool for syncing configuration files.

AANs have effectively mitigated the resource constraints associated with Aurora's Admin Node, leading to increased manageability and efficiency of the system at scale. Looking forward, potential areas for future work include further integration of HPCM's APIs and additional enhancements aimed at improving ease of use. These developments are expected to continue refining and expanding the capabilities and efficiency of AANs, contributing to more effective administrative tooling for managing and troubleshooting Aurora's complex environment. The AAN strategy may be useful for mitigating the issues that result from a centralized management system. This is especially relevant as new systems grow larger and more complex.

2 TERMINOLOGY

The following are key terms and concepts relevant to the management and operation of the Aurora supercomputer at ALCF. Understanding these terms is helpful for understanding the implementation and functionality of AANs within the Aurora system.

- **HPE Performance Cluster Manager:** HPE Performance Cluster Manager (HPCM), sometimes referred to as the Cluster Manager (CM), is a complex collection of multiple software components. Some of those are HPE specific. However, others are open-source. HPCM is packaged as an operating system image distributed by HPE. These components contribute to complexity due to its long history. The full description and operation of all HPCM components is beyond the scope of this paper. The primary interface to HPCM is the `cm` command.
- **Node:** In supercomputing systems like Aurora, a node is a defined unit running an instance of an operating system. All nodes within Aurora are managed by HPCM and are assigned specific roles, such as leader nodes, DAOS nodes, user access nodes, and compute nodes. Traditionally, node software is deployed via operating system images sent over the network.
- **Administrative Node**: The Administrative Node, also called the adminvm, is where the majority of components are installed and managed. It is the most critical node of the cluster, as the cluster cannot be managed without it. The Admin Node has the special hostname 'admin' [1].
- **User Access Node:** A User Access Node (UAN), also called a login node, is intended for initial system access and light tasks such as text editing, managing files, submitting jobs to the batch system, and monitoring jobs.
- **Physical Admin Node:** Physical Admin Nodes host the virtual administrative node, supporting the high-availability feature.
- **Quorum High-Availability:** Quorum High Availability (QHA) in the context of HPCM is a configuration method used to protect against hardware failure of the physical administrative node.
- **Domain Name System:** The Domain Name System (DNS) is the hierarchical system for naming nodes on internet protocol (IP) networks. In HPCM systems like Aurora, it is managed by the administrative node.
- **Application Programming Interface:** An Application Programming Interface (API) is used by software to interact with other software. The AANs utilize three of HPCM's APIs.
- **Representational State Transfer:** Representational State Transfer (REST) is an API architecture style used by some HPCM APIs.
- **JavaScript Object Notation:** JavaScript Object Notation (JSON) is a standard serialized data format used in some HPCM APIs.
- **Ansible:** Ansible is an automation and configuration management tool used by ALCF to manage many aspects of HPC system configuration.
- **Configuration Manger Database REST API**: The Configuration Manager Database REST API (CMDB REST API) is an API provided by HPCM for interfacing with HPCM's central configuration database.
- **Power Service REST API**: The Power Service REST API is an API provided by HPCM to control and view node power status.
- **Node Configuration Interface**: The node configuration interface is an undocumented interface used to trigger node reconfiguration.
- **Node Name Expression:** A Node Name Expression (NNE) is a string that can be interpreted by HPCM's node specification tools. This is further described in section 5.2

3 MOTIVATION

Aurora is configured to use a QHA configuration, which requires that the Admin Node is hosted on a virtual machine. This configuration, while helpful in maintaining system resilience in the event of hardware failures, introduces significant overhead and reduces the CPU and RAM resources available to admin users. The administrative node is the most critical component within Aurora, as it is the central location for managing, troubleshooting, and running the system. It handles a large volume of data, including log files with important information such as node console outputs, power statuses, and other operational metrics. The administrative node's critical role is underscored by its status as the authoritative storage within the HPCM architecture for all essential certificates, keys, and passwords required for node management. It is the central source of truth for all configuration data in the cluster.

The demands placed on the administrative node were too great when over 100 developers and administrators rely on it for parsing log files and other troubleshooting purposes. This necessity to parse extensive log files and manage various administrative tasks strained on the administrative node's resources.

A significant limitation of the HPCM's QHA is that they do not extend above the hardware level. This means that if the administrative node experiences a crash or becomes resource-starved, potentially due to out-of-memory (OOM) conditions or CPU over-utilization, then the entire cluster becomes unstable or fails in unexpected ways, introducing instability for the more than 10,000 nodes that make up the system. This issue poses a substantial risk to the system's overall functionality and reliability. Any disruption to the administrative node, whether due to accidental reboots or administrative errors, can have severe repercussions, compromising functionality of the entire Aurora system.

Because of these challenges, there was a need for creative stopgap solutions to address the resource-starvation issue. The introduction of AANs came from this need to alleviate the resource constraints on the administrative node and to enhance the overall stability and manageability of the Aurora supercomputer.

4 FEATURES OF AANS

In a conventional HPC environment, access to UANs is provided to users for code preparation and shell access. The number of UANs is directly correlated to the amount of resources available for these tasks. AANs serve this same purpose for administrator users; they enable administrator users to perform various tasks without relying on the administrative node, thus reducing the demand on it's limited resources and increasing the stability of the cluster as a whole. AANs are configured similarly to UANs while including additional tools and features for administrators.

4.1 General Tools and Resources Available on the AANs

- Packages used by administrators such as `parallel`, `pdsh`, `tmux`, `console` and similar are included.
- Read-only mounts of file systems that are used by the administrative node and leader nodes, including `/var/log/consoles` and `/var/log/HOSTS`.
- Administrator specific home directories and other shared network filesystems are mounted where appropriate.
- The AAN images are configured on physical nodes with local disk storage to provide scratch space for data manipulation.
- Specific paths on the Administrator node are exported read-only to the AANs. These paths include, `/var/log/` to provide log files and `/opt/clmgr/image/images` to provide node images.
- AAN images are directly hosted on physical nodes, resulting in greater performance.
- The Distributed Shell (`dsh`) package is provided and associated node group lists are synchronized from the administrative node.

To better enable use of the AANs as a management platform, SSH host-based authentication is enabled from the AANs to most nodes. This configuration allows administrative users to access to other nodes directly without needing SSH keys on the node. This is useful when the node being accessed is miss-configured or when certain resources such as shared home directories are unavailable during periods of maintenance or failure.

4.2 `cm-remote`

On the AANs, the `cm` command is provided by a tool called `cm-remote`. When managing the HPCM system, the majority of tasks are performed using the `cm` command. While it is not practical to replicate every option available in the `cm` command, several options can be feasibly re-implemented on AANs. To facilitate the transition to using AANs, the `cm-remote` tool was developed to emulate the behavior of the `cm` command as closely as practical. `cm-remote` attempts to parse and process `cm` commands locally but when the task is not possible to perform locally, it will run the command via SSH on the administrative node. Limitations of this approach are discussed in 4.2.2

4.2.1 Implementation of `cm-remote`. In order to provide the most consistent experience passible for administrator users, certain HPCM packages that would ordinarily be only installed on the administrative node are also partially installed on the AANs. One of these packages provides the Python environment that HPCM provided `cm` command uses. `cm-remote` uses this environment and included Python packages.

The below code is a portion of the `cm-remote` command. Note the many direct writes to `stderr` and `stdout`, which are used to emulate the `cm` command as closely as possible. This code does not demonstrate good software development practices, but is included here in order to show the challenges inherent in emulating another command in this way. With more time dedicated to the project, these patterns may be fixed.

```
# exit early if the command list is empty:
if len(command_list) == 0:
    sys.stderr.write("usage: cm [-h]\n")
    sys.stderr.write("          {aiops,chassis,monitoring,
         controller,cooldev,cvt,group,health,image,inventory,
         mgmtswitch,network,node,power,repo,sim,support,
         system,wlm} ...\n")
    logging.debug("got no arguments, exit early")
    exit(1)

if is_cacheable(command_list):
    format_exit_cache(run_remote_cache(command_string))
if command_list[0] == 'node':
    if command_list[1] == 'show':
        # cm node show shows info about nodes
        # user did not put -n flag
        for arg in command_list:
            # we can't process other cli opts
            if arg not in ['-n', '--node', '-I', '--image']
                 and arg[0] == '-':
                run_remote(command_list)
        node_expression = ''
        for i in range(len(command_list)):
            if command_list[i] == '-n' or command_list[i] ==
                 '--node':
```

```
            if len(command_list) < i + 1:
                sys.stderr.write("usage: cm node show [-h]
                     -n NODE ...\n")
                sys.stderr.write("cm node show: error:
                     argument -n/--node: expected one
                     argument")
                exit(2)
            node_expression = command_list[i+1]
    image = False
    for i in range(len(command_list)):
        if command_list[i] == '-I' or command_list[i] ==
             '--image':
            image = True
            logging.debug("got --image cli arg")
    if image == True:
```

As not all commands were able to be implemented in `cm-remote` directly; a method was required to determine which `cm` commands on the administrative node were the most used. In order to do this an analysis of the `cm` sub-commands executed on the administrative node was conducted to identify those most frequently used. This analysis enabled development effort to be focused on the `cm` commands which would be most likely to be frequently required by users in the future.

To determine the most commonly used commands, the number of occurrences of each command in the administrative node's `cm.log` file was counted. The findings revealed that most of the commands were related to `cm power`, these commands are used to power the nodes on and off and determine power status. The remaining commands primarily served information-gathering purposes. These functions were added to `cm-remote`.

4.2.2 Utilization of SSH by cm-remote. As stated previously, commands whose functions could not be easily replicated in `cm-remote` are executed on the administrative node via SSH. `cm-remote` handles the SSH connection setup and the capture of the exit code, `stdout`, and `stderr`. Capturing the exit code this allows the use of conditional expressions on the AAN's `cm` command. Some options, such as those involving file paths in command line options, do not work transparently. However, most users do not use such options, and those who do are generally aware of the limitations of `cm` on the AANs. In the case that the administrative node refuses the SSH connection, an error message is displayed, instructing the user to retrieve the key from the administrative node if needed by using the `admin-sync` tool described in 4.4.

4.2.3 Verification and Retrieval of the Hosts File by `cm-remote`. A key function of the `cm-remote` tool is the validation and retrieval of the `/etc/hosts` file from the administrative node. The administrative node serves as the authoritative source for intra-cluster host names. To ensure that AAN users can utilize the same names as those on the administrative node, the `/etc/hosts` file from the administrative node must be retrieved. Upon invocation, `cm-remote` checks the `/etc/hosts` file to verify that it contains a hostname entry for the AAN it is running on. If such an entry exists, no update is necessary, as the other hosts can be assumed to be present. If not, it indicates a mismatch with the administrative node's file and `cm-remote` retrieves the file from the administrative node. Although hosts can be renamed, resulting in an undetected out-of-sync hosts file, in practice such changes are rare and are broadly communicated to the administrative team, allowing for manual updates.

4.2.4 Caching of Help Function Outputs. Certain commands consistently produce the same output regardless of when they are executed, `–help` commands are the clearest example of this. To minimize overhead and reduce the load on the administrative node, the output of these commands is cached by `cm-remote`. Any command with the `–help` flag has its results stored locally in a JSON dictionary residing in a file on the AAN's local filesystem. Subsequent requests for that command will return the previously cached output and will exit with the cached command's return code. Although updates to HPCM could potentially alter the help text and render the cached version obsolete, in Aurora's configuration the AAN's root filesystems reside in memory. Consequently, a reboot will clear the cache.

4.3 Direct API Calls from the Command Line Using `cmn`

The `cmn` command is a combined front end to the Power Service REST API, cmdb REST API, and the config-manager API. Since `cmn` uses APIs directly, it is faster and appropriate for use in automated scripts. Both commands are a work in progress. Use the `–help` option is provided by each for admin users to determine the current available features.

A significant limitation of `cm-remote` is its adherence to the command line interface provided by `cm`. This interface essentially reflects the interfaces of the underlying tools, which can lead to confusion due to the varying behaviors of different subcommands. Replicating these nuances is not feasible. The `cmn` tool, a straightforward API wrapper, addresses this issue by offering a more consistent and efficient experience. It still leverages the Python environment installed with HPCM, enabling it to utilize the same command line interface library employed by HPCM, making the interface familiar to administrative users even if it is not the same. `cmn` also serves as a fallback information provider in scenarios where SSH access is unavailable for technical or administrative reasons. Currently, `cmn` supports GET requests to the CMDB REST API, the "node-update-config" API, which instructs HPCM to update specific configurations, and the power control API.

4.4 Secrets check

A simple tool, called `admin-sync`, was created to sync the secrets and some configuration files from the administrative node to the AAN. If something fails during usage due to missing secrets or configuration files, then admin users are advised to run `admin-sync` as a first troubleshooting step. This script uses the calling user's SSH access to the administrative node to copy SSH keys, secrets, tokens, and other files to the AAN. It was decided to not to schedule this sync so as not to clutter log files or create unexpected changes. In the event that major system configuration changes happen, a reboot of the AAN will be undertaken and `admin-sync` will be ran at that time.

5 AAN IMPLEMENTATION DETAILS

5.1 DNS Configuration

Round-robin DNS is a common method of load distribution, load balancing, or fault-tolerance by provisioning multiple redundant servers. The AANs are configured with a round-robin DNS setup, with two nodes dedicated to AAN tasks. During maintenance on an AAN node, administrator user logins are disabled on one node, allowing administrative users to finish their tasks before the node must be rebooted to perform updates. This process is then repeated for the other node. This helps to minimize user disruption.

5.2 Node Name Expressions as Used By `cmn` and `cm-remote`

A Node Name Expression (NNE) is a wildcard pattern used to specify a set of nodes, this pattern may be in expanded form or condensed form. NNEs are utilized throughout HPCM, `cm-remote`, and `cmn`. However, the expansion and condensing of these expressions are directly available only through `cmn`. In this context, "expanded" means that each node is specified by name, while "condensed" refers to the smallest string that selects the same set of nodes.

As documented in the HPCM Admin Guide[1], node names can include the following wildcards:

- `*`: Matches one or more characters.
- `?`: Matches exactly one character.
- `[ ]`: Matches any of the range of characters specified within the brackets.

Additionally, the `@` symbol can be used to specify custom groups in HPCM. The `*` and `?` wildcards are handled using standard Python regular expressions, while other patterns are managed with custom logic. Note that `cm-remote` and `cm` do not support the `@` symbol for group selection due to its complexity and low usage.

An example of a simple NNE is:

```
x4705c0s0b0n0,x4411c7s0b0n0,x4402c6s2b0n0
```

An example of a more complex NNE is:

```
[4102,4215]c7s6b0n0,x[4201,4220]c1s0b0n0,x4002c6s3b0n0,
    x4105c7s3b0n0,x4111c4s2b0n0,x4114c5s0b0n0,x4204c4s7b0n0
    ,x4213c2s6b0n0,x4214c2s7b0n0,x4219c0s0b0n0,
    x4302c2s3b0n0,x4305c5s2b0n0,x4305c6s1b0n0,x4420c0s6b0n0
```

Wherever possible, NNEs are processed locally to reduce the load on the administrative node. This involves expanding NNEs to determine the set of nodes they target. To achieve this, we need an up-to-date list of node names to check against the NNE. This list is obtained from either the CMDB API or the `/etc/hosts` file depending on which is accessible by the AAN tooling. The code for expansion is similar to that on the administrative node with minor modifications, and runs on the AAN. Condensing is done by a binary program provided by an HPCM package that is typically installed only on the administrative node but is partially installed on the AANs.

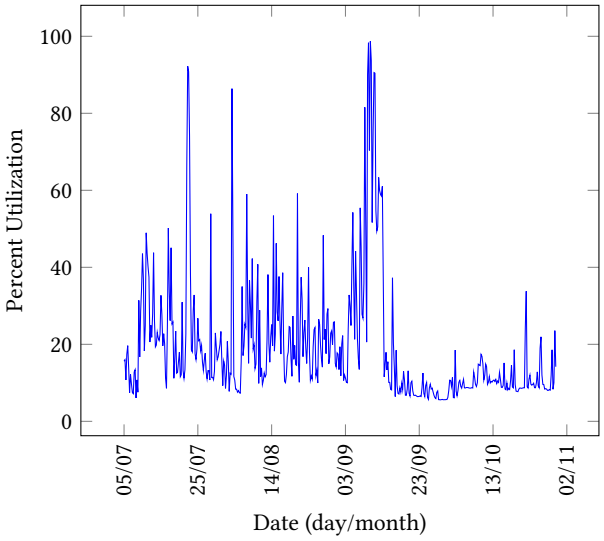
5.3 The History of HPCM and how it Relates to the AANs

The specifics of HPCM's development are not clear. Over time, various commands have been introduced, some of which use earlier tools, such as those prefixed with 'cmu*'. More recent commands are typically prefixed with 'cm'. This evolution has resulted in a complex and patchwork system of APIs and files. Because of this, not all commands are possible to implement on the AANs. It was decided that these earlier commands are not usually relevant to day-to-day tasks and do not need to be considered.

5.4 Maintenance and Updates to AAN Configuration

Node image configurations in Aurora, including configuration of the AANs, is managed using Ansible and tracked in a Git repository. Existing Ansible variables used to define the UANs are duplicated and modified as needed to create the AAN images. Using Git and Ansible enables precise replication of the configuration of AANs from the existing UAN setup whenever applicable changes are made that need to be propagated to the AANs from the UANs.

6 RESULTS



The above graph shows the percent utilization of the administrative node's CPU resources over approximately four months. This period includes the introduction of the AANs.

The left side of the graph shows a high utilization baseline average around 40% with many spikes reaching full utilization (100%). These spikes indicate periods of heavy usage. During these periods system instability occurred and the critical work of administrator users was disrupted.

The middle of the graph shows a sudden decrease in average utilization by about 25%. This indicates the transition to AANs. The decrease was rapid due to close collaboration with the users who required the majority of the resources. This collaboration allowed the transition to be smooth with limited disruption.

The second half of the graph shows a significantly more stable utilization average at around 15%. There is a lack of significant utilization spikes. This shows that the resource starvation issue was successfully mitigated. The stabilization in the graph was also reflected in the administrator user experience.

7 LIMITATIONS

7.1 Limitations of SSH and cm-remote

One of the significant limitations inherent with the use of SSH by `cm-remote` is the loss of fidelity. For example, file path arguments given on the AAN may not exist or have different contents on the administrative node. Additionally, if a path is provided in command output, the administrator user may be confused as the path refers to the location on the administrative node and not on the AAN. Furthermore, each command incurs significant overhead because a complete SSH session must be established for every command. Consequently, the `cm` command on the AANs must not be used in scripts to avoid issues with logging and excessive resource consumption due to repeated SSH connections.

Logging also presents challenges, as it is difficult to determine who executed specific actions and from what host, since all users run commands via SSH on the administrative node as root when using the `cm` command from an AAN. Logs for `cm-remote` are written to `/var/log/cm-remote.log` but they still must be collected outside the typical logging processes used by HPCM.

7.2 Limitations of the HPCM APIs

The HPCM APIs have several significant limitations that limit their effectiveness in the context of AANs. The most significant of these is the absence of per-user authentication mechanisms and cohesive secrets management. Currently, each REST API is secured using a single key, while the "cm-node-update" configuration API relies on certificate-based security. This approach lacks the granularity required for individual user authentication.

Furthermore, the Power API does not include all the functions available on the administrative node for power management. These functions include setting the power state for the full system and certain other power management related commands. This limits its utility in advanced cases.

The node update configuration API is the most limited, as it is more a simple messaging interface rather than a typical API and does not follow the REST pattern. This restricts its integration potential and increases the logic required by the client, in this case the AAN tooling.

8 POSSIBLE FUTURE WORK

8.1 Enhancing Existing Tools

Possible enhancements include expanding the functionality of the `cmn` command to support API write commands. The tool only supports HTTP GET operations when interacting with the CMDB REST API. Expanding its abilities would further decrease the load on the administrative node as administrative users could interact with the CMDB directly from scripts and the command line.

Another enhancement could be increasing the number of commands that the `cm-remote` tool is able to process on the AANs directly. This would reduce reliance on the SSH interface in order to further decrease the load on the administrative node.

Some other minor code quality and process improvements are also possible. For example the `/etc/hosts` file could be retrieved from the administrative node after a specified period or when certain cluster events are detected. This and other minor improvements would add additional stability and reduce the chance that administrative users encounter confusing error messages or other issues.

9 CONCLUSION

The deployment of AANs has significantly mitigated the resource constraints of Aurora's primary administrative node, thereby streamlining overall workflow and increasing system manageability at scale. With future refinements, AANs are expected to further enhance efficiency, with sustained benefits in managing Aurora's exascale computing resources. The AAN concept should be considered in similar situations where full decentralization of limited computing resources are not possible or practical.

ACKNOWLEDGMENTS

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH1135.

REFERENCES

[1] Hewlett Packard Enterprise 2019. *HPE Performance Cluster Manager Administration Guide* (1st ed.). Hewlett Packard Enterprise. Available at https://support.hpe.com/hpesc/public/docDisplay?docId=dp00004196en_us&docLocale=en_US.