Thai Journal of **Math**ematics
Volume 19 Number 3 (2021)
Pages 739–751

http://thaijmath.in.cmu.ac.th

ISSN 1686-0209

TARA: A Year in Review

Putt Sakdhnagool[1], Manaschai Kunaseth[1], Apivadee Piyatumrong[1], Chompoonut Rungnim[1], Viwan Jarerattanachat[1], Wirote Udomsiripinij[1], Kittithorn Tharatipayakul[2], Takdanai Suwan[1] and Piyawut Srichaikul[1]

[1] *NSTDA Supercomputer Center (ThaiSC), National Science and Technology Development Agency (NSTDA), Pathum Thani, 12120, Thailand*
e-mail : thaisc@nstda.or.th

[2] *Data Center and High-Performance Computing Management Section (DCM), National Science and Technology Development Agency (NSTDA), Pathum Thani, 12120, Thailand*

Abstract In 2018, National Science and Technology Development Agency (NSTDA) has initiated a national project to establish National Science and Technology Infrastructure (NSTI) to enabling advanced scientific research and development in Thailand. As a result, the TARA cluster has been deployed and operated by NSTDA Supercomputer Center (ThaiSC) since February 2019. The cluster is a heterogeneous cluster with a theoretical peak performance of 500 teraflops. The key mission of TARA is to support large-scale computing demand from wide range of computational science applications in Thailand. In this paper, we describe the detailed cluster information and service operation of TARA cluster in the past year including the design criteria, hardware specification, management, and user support and experience. We also discuss about the challenges we encountered, and lesson learned from these experiences.

MSC: 68-00
Keywords: Supercomputer center, Deployment and Design, High performance computing

Submission date: 26.04.2021 / Acceptance date: 03.08.2021

1. Introduction

During the past few years, advances in computational science disciplines, such as computational material science, bioinformatics, and artificial intelligence, have gained major attentions from Thai government to help solving problems in the country. However, lack of existing advanced computing infrastructure prevents Thailand to progress in such direction. Therefore, In 2018, National Science and Technology Development Agency (NSTDA) initiated a project to build a national infrastructure for high performance computing (HPC) in Thailand. As a result, NSTDA Supercomputer Center (ThaiSC) is officially established from this initiative to provide HPC infrastructure service to accelerate scientific research and development in Thailand.

TARA (Figure 1) is the first computing infrastructure project from ThaiSC with the main mission to 1) provide HPC computing demand within NSTDA and 2) serve as a

Published by The Mathematical Association of Thailand.

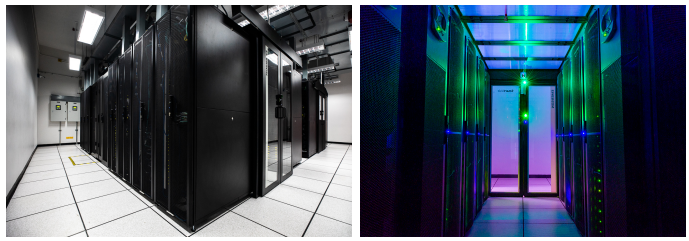

FIGURE 1. TARA Cluster. Left: A containment for the TARA cluster. Right: Racks with computing nodes.

proof-of-concept platform before the development of national-scale HPC cluster afterward. Hardware acquisition of TARA was completed in January 2019, while the actual deployment was begun in February 2019.

In this paper, we describe our experiences with TARA – from the design, deployment, management, to user supports and experiences. The key objective is to provide technical configuration, deployment information, and the lesson learned from various challenging issues for any institutions who want to start building HPC infrastructure from scratch. The paper is organized as follows. Section 2 describes the requirements and technical specification of TARA. Section 4 reviews the management of the TARA cluster. In Section 5, we discuss user support operation. Section 6 reviews our lesson learned from operating TARA for a year. Lastly, Section 7 concludes the paper with a future plan for the TARA cluster.

2. TARA Cluster Design

One of TARA cluster's goals is to support research activities within NSTDA. We designed the cluster based on the applications and research fields gathered from the questionnaires and interviews with NSTDA research staffs. We identified the following requirements based on application types.

(1) **Computational Science**: Majority of computation researches fall into this category. The major applications are from computational chemistry, material science, and engineering. These applications are compute-intensive and requires low-latency communication overhead. While GPU implementations are available for some applications, they are not widely adopted by our users or well-prepared for production uses. As a result, users from this group still prefer a homogeneous, multi-core cluster with low-latency network over accelerator-based clusters.

(2) **Artificial Intelligence (AI) and Data Analytics**: Users from this group mainly use computing resources for model training, which is highly efficient on accelerator-based system such as GPUs. In addition, a large high-performance storage is needed to store large training datasets.

(3) **Bioinformatics**: Most of NSTDA's bioinformatics researches focus on de novo genome assembly and sequence analysis, which process massive amount of data. These applications are memory-intensive, requiring large memory capacity, I/O and network bandwidth, and storage space. Computing power is less important, so does low-latency network as most of them do not scale across nodes.

To satisfy the requirements, we designed TARA to be a heterogeneous cluster with low-latency, high-bandwidth network and a high-performance parallel storage. Table 1 shows an overview of TARA cluster.

Table 1. TARA Cluster Overview

Type	Target Workload	Nodes Count	Total Cores	Total Memory	Total GPUs	R_{peak}
Compute	Generic HPC	60	2400	11.25TB	-	184.32 TFlops
Memory	Memory intensive	8	1536	24.00TB	-	103.22 TFlops
GPU	GPU development	2	80	0.75TB	4	28.00 TFlops
DGX	GPU-intensive	3	120	1.50TB	24	187.20 TFlops
					Total	502.74 TFlops

2.1. Computing Nodes

TARA has four types of computing node: *Compute*, *Memory*, *GPU*, and *DGX*. The specifications of each node are as follows.

(1) **Compute Node:** The compute node is designed for generic HPC workload, which majority of scientific applications falls into this category. Each compute node has two 20-core Intel Xeon Gold 6148, 192 GB DDR4-2666 ECC RAM, and an EDR Infiniband card. There are 60 compute nodes in TARA cluster.

(2) **Memory Node:** The memory node is designed for memory-intensive workload. This type of workload has large memory footprint, which could get performance benefit from large memory pool. Each memory node has eight 24-core Intel Xeon Platinum 8160, 3 TB DDR4-2666 ECC RAM, and two EDR Infiniband cards. There are 8 memory nodes in the cluster.

(3) **GPU Node:** The GPU node is designed for GPU application development and testing. It consists of two 20-core Intel Xeon Gold 6148 processors, 384 GB DDR4-2666 ECC RAM, an EDR Infiniband card, and two Nvidia Tesla V100 GPUs with 16GB memory. TARA cluster has two nodes of this type.

(4) **DGX Node:** The dgx node is an Nvidia DGX-1 server, a specialized machine designed for GPU-intensive workload. Each machine has two 20-core Intel Xeon E5-2698 v4 processors, 512 GB DDR4-2666 ECC RAM, and four EDR Infiniband card, and eight Nvidia Tesla V100 GPUs with 32GB memory. The cluster has three DGX node; two are dedicated for bioinformatics and the remaining one is for general usage.

Node specifications are summarized in Table 2. All computing nodes run CentOS Linux version 7.6.

In addition to the computing nodes, TARA has two *frontend* nodes for user access and *administration node* for cluster services, such as node deployment, job scheduler, authentication system, license managers. The services are deployed on virtual machines running in the administration node, which uses VMware ESXi 6.7 as the hypervisor.

2.2. Parallel Storage

The cluster has 780TB of shared parallel file storage, running IBM Spectrum Scale parallel file system (also known as IBM General Parallel File System or GPFS). The

Table 2. TARA Node Specification

Type	Processor	CPUs	Cores/ CPU	Memory	GPU	Network
Compute	Intel Xeon Gold 6148	2	20	192GB	-	1x EDR IB
Memory	Intel Xeon Plat. 8160	8	24	3TB	-	2x EDR IB
GPU	Intel Xeon Gold 6148	2	20	384GB	2x Nvidia Tesla V100 16GB	1x EDR IB
DGX	Intel Xeon E5-2698v4	2	20	512GB	8x Nvidia Tesla V100 32GB	4x EDR IB

storage system consists of five I/O management nodes for Spectrum Scale service, four Lenovo DS6200 storage systems, and two SAN switches. The I/O management nodes and the storage systems are connected via two SAN switches for redundancy.

The file system is also configured with Declustered RAID [3] providing similar fault-tolerance level as RAID6 with faster rebuilding time.

2.3. Networks

TARA consists of three networks: *Infiniband*, *Service*, and *Management* networks.

(1) **Infiniband network**: This network is used for inter-process communication, data transfer, and parallel file system access. The network is a fat tree network [6] utilizing a low-latency, high-bandwidth, EDR Infiniband. All computing nodes, frontend nodes, administration node, and I/O management nodes are connected to this network.

(2) **Service network**: The service network is used for cluster services, e.g. job scheduling, authentication, license management, and etc. The network is a gigabit Ethernet connecting all servers in the cluster.

(3) **Management network**: The management network is a gigabit Ethernet network connecting baseboard management controllers(BMC) of all servers in the cluster for out-of-band management.

All nodes in the cluster do not have Internet access excepts the frontend and administration nodes, which are connected to external network via 10 gigabit Ethernet.

3. Data Center Preparation

Hosting an HPC cluster is challenging for existing data centers due to its power consumption and heat generation. While NSTDA has several data centers available across its research centers, all of them are designed for enterprise workloads, such as web services and databases. Hosting an HPC cluster on such facility was challenging since enterprise workloads typically require less power.

To accommodate HPC clusters, we upgraded the latest data center; doubling its power and air cooling capacity to 120kW and 146kW, respectively. We also expanded containment's wireways to support cables from cluster networks. The data center upgrade began a month before TARA's commission and continued until TARA's equipment arrived.

Even with the upgrade, power density of HPC cluster is still a challenge. The server racks in the data center each has 10kW power capacity; however, a dense HPC cluster

could easily draws 40kW of power per rack. As a result, TARA cluster could only utilizes half of each server rack due to power capacity, keeping remaining half emptied. We also moved an existing cluster to another data center, clearing up ten server racks, to make space available for TARA. Overall, TARA takes nearly 35% of the new data center's server racks.

4. Cluster Management

4.1. Node Provisioning

We use xCAT [13], an open-source software for automating deployment and management of HPC clusters, for node provisioning. All computing nodes, except DGX nodes, and frontend nodes are provisioned by xCAT.

To perform node provisioning, xCAT requires machine definition – including machine information and network configuration. Machine information, e.g. model and serial number, are collected by xCAT's automatic hardware discovery. Network configurations, such as IP addresses and hostname, are manually assigned after the definition is created by hardware discovery. This assignment was done by the vendor with our configuration for IP addresses. The final xCAT definitions were exported to an xCAT's stanza file for future use.

OS provisioning and machine configuration were done through xCAT automatic deployment. We defined a list of RPM packages to be installed during the OS installations and a set of post-installation scripts for setting services including job scheduler, parallel file systems, and so on. xCAT follows these configurations to installing OS on the machines.

4.2. Job Scheduling

TARA uses Slurm Workload Manager [14] for job scheduling and resource management. Slurm tracks cluster resource availability such as CPU cores, memory, and GPUs, and allocates them to jobs waiting in queues based on the job's priority. We built Slurm to use PMIx [2] as default process manager with UCX [11] for communication framework. TARA uses PMIx 3.1.4 [9] and OpenUCX 1.6 [12].

We created Slurm partitions (Slurm's term for job queue) for each computing node types in Section 2.1, namely *compute*, *memory*, *gpu*, and *dgx*. Table 3 summarizes partition configuration. The compute partition consists of 58 of 60 compute nodes. The remaining 2 nodes are reserved for development purpose. All memory and GPU nodes are available for the memory and gpu partition, respectively. For the dgx partition, only one DGX node is available. The remaining twos are reserved for bioinformatics workload. If users do not specify a partition for running a job, the job will run in the compute partition.

The granularity of resource allocation is at core-level for the compute and memory partition, and at node-level for the gpu and dgx partition. In these partitions, jobs can reserve the resources up to 5 days. We set the maximum wall time to 5 days based on the input from users. During the first year of operation, we observed a reasonable turn-over time for jobs. The cluster achieved 69.63% annual utilization with 93.90% daily utilization at peak usage.

In addition to the type-based partitions, we deployed three additional partitions: two preemptive partitions and one development partition. The preemptive partitions targets

computing nodes in the memory and dgx partition for encouraging users to use under-utilized resources. The partitions are named *memory-preempt* and *dgx-preempt*, respectively. These partitions have the same setting as their non-preemptive counterparts except jobs submitted to these partitions will have lower priority and can be killed by higher priority jobs (i.e. jobs submitted to memory and dgx partitions). The memory-preempt partition is available on three memory nodes. The dgx-preempt partition is available on the two reserved DGX nodes.

The development partition, named *devel*, is designed for debugging and interactive jobs. The partition uses two reserved compute nodes. The job's wall time is limited to 2 hours allowing fast turn-over rate. We created a wrapper script for submitting an interactive job, called `sinteract`*. This script automatically set Slurm's flags to puts task zero in pseudo terminal and enable X11 support for an interactive job. The script also checks for Slurm's account the job will charge from. If none is specified, it will ask users for the account.

Table 3. TARA Job Partitions

Partition	Node Type	#N	Max. Walltime	Preemptive	SU Charge
devel	compute	2	2 hours	No	1 SU/core-min
compute		58	5 days	No	1 SU/core-min
memory	memory	8	5 days	No	1.25 SU/core-min
memory-preempt		8	5 days	Yes	0.50 SU/core-min
gpu	GPU	2	5 days	No	130 SU/node-min
dgx	DGX	1	5 days	No	815 SU/node-min
dgx-preempt		2	5 days	Yes	400 SU/node-min

Job's priority is computed by Slurm's multifactor priority plugin based on several factors, e.g. job size, age, and account's fair-share. Fair-share is prioritized for job's priority while other factors are weighted equally. The scheduler give higher priory to large jobs. Job age in the partition also contributes to job's priority. The job age factor will reach its maximum after 7 days of waiting in the queue. We set partition priority factors to be equal on all partition except the preemptive ones, which has lower priority. We do not use quality-of-service for our priority factor. In addition to multifactor priority, we put a hard limit on resource usage. Each project's account can use up to 1,000 cores and run 100 jobs simultaneously. This configuration prevents cluster monopolization caused by users.

4.3. Accounting

Jobs running on TARA are charged for its resource usage. Instead of using a real currency, TARA uses *Service Unit (SU)*, which is equivalent to a core-minute of a compute node. For example, a job requesting 20 cores of a compute node for 15 minutes will be charged 300 SUs when the job is finished. We set the SU charge for each partitions, based on the cost of acquiring and operating nodes in each partition. Special discount is applied to preemptive partitions for encouraging users to use under-utilized resource. Table 3 summarizes SU charge.

To implement this accounting mechanism, we use Slurm accounting service called *Slurm Database Daemon (SlurmDBD)*. The service keeps record of resource usage by each project

*https://github.com/thaisc-hpc/slurm-sinteract

and prevents jobs from running when the usage exceeds the available budget. We implement this mechanism using Slurm components (displaying with `monospaced`, hereafter) as described below.

After user submit a project proposal and is approved, we create an `account` for the project along with `users` for the project's members. These `users` are linked to the `account`, creating an `association`. A `user` without `association` cannot run any jobs on the cluster. A `Quality of Service (QoS)` is created along with the `account` and set as the account's default `QOS`. We use this default `QoS` to apply resource limits to the `account`, acting as the project's wallet.

Resource limitation is configured through a `QOS` parameter called `GrpTRESMins`. The parameter indicates the numbers of trackable resource (`TRES`) minutes that can be used by jobs throughout the lifetime of the `QOS`. As SU does not exist in Slurm, we use `billing` – a type of `TRES` – to represent SUs. By setting `GrpTRESMins` with an amount of `billing` equal to the project's SU budget, we could use the `QOS` as the project's wallet. The `QOS` flag is set with a `NoDecay` flag, preventing `QOS` usage to be decayed. As a result, resource usage will keep accumulating in the `QOS` when jobs are running from the `QOS`. A job's `billing` usage is calculated based on the partition it runs on and the execution time of the job. When the resource usage reaches `GrpTRESMins`, users cannot run any job from the `account` unless additional budget is provided.

For checking project's SU balance, we have developed a utility, called `sbalance`[†]. The utility displays the remaining balance in a user-friendly format (Figure 2), compared to Slurm's `scontrol` utility. In addition, the utility can show a breakdown of SU usage by users and export the results to a file either in CSV or JSON format.

```
      Account  Description  Allocation(SU)  Remaining(SU)  Remaining(%)      Used(SU)
--------------------------------------------------------------------------------
    proj2032         refai        33333333       33009296         99.03        324037
    proj2120         graph       100001002       30092076         30.09      69908926
```

FIGURE 2. Example `sbalance` Output

4.4. File Systems

TARA uses IBM Spectrum Scale [10](also known as IBM General Parallel File System) for its parallel file system. IBM Spectrum Scale is high-performance clustered file system designing for providing rapid accesses to large data in parallel. The parallel file system has `/tarafs` path prefix and is organized as shown in Table 4.

(1) **Home**: Each user has a personal directory in this path for storing user's files and codes. The quota for each user is 50GB.

(2) **Project**: Each project has a project directory for sharing data among its members, such as software, data sets, and etc. The default quota is 200GB but can be increased upon user request.

(3) **Scratch:** Scratch space is a fast, high-performance storage, made of solid-state drives (SDD). It is designed for temporary files that require I/O performance, e.g. application cache, short-term data sets, and etc. The files are shared with

[†]https://github.com/thaisc-hpc/slurm-sbalance

project members and will be purged if there is no access in the past 60 days. We do not put any limit for this file system.

(4) **Large:** This storage space is a specialized storage designed for storing large files. The file system use large block size for improving performance. The file system is created as a part of special agreement and can be accessed by only a member of designated group.

(5) **Utility:** The utility space is an administrator-only storage for storing system-wide modules, applications, and system configurations.

Table 4. TARA File Systems

File System	`/tarafs` Path	Total Size	Quota	Block Size	Disk Type
home	`/data/home`	300TB	50GB/user	4MB	NL-SAS
project	`/data/project`		200GB/project. Expandable upon request	4MB	NL-SAS
scratch	`/scratch`	80TB	Unlimited for projects. Purges every 60 days	4MB	SSD
large	`/large`	400TB	Restricted access	8MB	NL-SAS
utility	`/utils`	10TB	Administrator only	4MB	NL-SAS

4.5. Software Environment

TARA provides software to users through a module system, called *Lmod* [7]. Users access applications by loading modules, a configuration file containing instructions for setting user's environment. Lmod automatically configures user environment and removes conflicting libraries based on the instructions contain in the module. This allows the cluster to provide multiple versions of the same application.

To build modules, we use *EasyBuild* [4], a framework for building and installing software on HPC cluster. EasyBuild automatically builds software and generating module file from *easyconfig* file, a configuration file containing instructions for building software and creating module file. We have built and installed 231 modules, including applications and libraries.

In the first year of operations, we focused on providing the basic development environment, which includes compilers and libraries. Users were responsible for building their own applications. We focused on two major aspects for the development stack: *compatibility* and *performance*. For compatibility, we provide a `foss` (stands for Free and Open Source Software) toolchain. The toolchain consists of open source software, such as GNU compilers, OpenMPI, and OpenBLAS, which are a common development tools for open source applications. For performance, we have a `intel` toolchain consisting of component from Intel Parallel Studio XE, including compilers, MPI, and MKL. These tools are highly optimized for Intel processors, which used in TARA computing nodes. In addition, we also provide CUDA and Python for AI development.

To provide portability and reproducibility of software, computer scientists and software developers have shifted their development environment toward container-based systems,

especially in the field of AI. To accommodate this change, TARA provides a container-based system, called *Singularity* [5]. Singularity is a file-based container system designed for HPC clusters. We decide to use Singularity instead of industry standard, i.e. Docker [8], due to its integration with job scheduler and its security features.

5. User Support

During the first year of service, we had 54 registered projects across various research disciplines, such as computational chemistry, material science, bioinformatics, and artificial intelligence. Majority of these users have limited experience in a time-sharing HPC cluster, leading to a high expectation for user support for the new cluster.

In this section, we will explain the design and the implementation of both user training (Section 5.1) and issue management (Section 5.2). Section 5.3 ends this section with user satisfaction from TARA users.

5.1. User Training

Primary focus of user training is to introduce how to use TARA as a scientific tool. In this regard, we did a survey on users, who express their interest in using TARA, about their past experience with HPC clusters. The results show a wide variety of experience, ranging from a complete novice to an expert. Some users never have access to any HPC clusters, while some are seeking to expand their computing capacity. This gives us a basis for designing user training.

We derived the training plan based on the finding. The courses are split between entry and advance courses in order to engage different level of users and increase the potential to utilize TARA efficiently. Table 5 shows the sessions we provided throughout the year of 2019, in total of 14 sessions.

Table 5. TARA User Training Sessions in 2019

Level	Sessions	#Session
Information	TARA Information Session	6
Entry	HPC School - Getting Started on TARA	4
Advance	Introduction to HPC Programming	1
	TARA Best Practice	1
	Intel AI workshop	1
	NVIDIA workshop	1
	Total	14

Information sessions aim to introduce TARA and bring general awareness to the research community. Entry-level training focuses on how to use an HPC cluster, including basic Linux command and basic job submission. Advance-level sessions bring a more in-depth training to the users, such as programming and how to use the cluster more efficiently. Some of the sessions are provided by hardware vendors, which focus on a more specific topic like AI and GPU computing.

Overall, we have seen a steady growth in the number of users during the year coming from different research areas.

5.2. Issue Management

Issue management is another important function to support users. Our team consists of both HPC and domain experts, covering most research areas done by our users. However, we only have an expert for each field. Accordingly, the expert may not response to user's need in a timely manner due to their availability. To handle the time of our team more effectively, we categorize issues submitted to us into three levels.

The first level (L1) is defined for trivial issues, ranging from resetting password to fixing obvious mistake in user's job submission script. If an issue requires more than a working week to resolve, software development, or cluster reconfiguration, it will escalate to a second level support (L2). Feature and application requests are an example of an L2 issue. If the issue involves service and operation policy, it will be classified as a third level issue (L3). L3 issues are handled through meeting among decision makers. For this L3, time cannot be guaranteed.

The designed workflow of issue management is materialized with the concept of Kanban board of 7 columns: *Issue, Pending Acknowledgement, In Progress, Resolved, Waiting for Users, Escalated to L3,* and *Closed*. The board in implemented using Trello [1]. Our issue management workflow is as follows.

When an issue is submitted through ThaiSC's email account, the issue card will be generated under column *Issue*. Our helpdesk person, who is also an expert for L1 issues, will handle the issue by considering the severity of the problem. If the issue appears is classified as L1 or L2, an expert will be notified and tagged to the card. After tagging is done, the helpdesk moves the card to *Pending Acknowledgement*. This is where our experts start working on the issue. The assigned expert acknowledge the issue by moving the issue's card to *In Progress*. When the issue is solved, the expert updates the card with detail of solutions. An email will be sent to the issue's owner filled with solutions or information. After that, the card will be moved to the *Resolve* column.

Overall, the helpdesk person will monitor the progress of card daily. In particular, those cards residing in the *Resolved* column will be followed up with the user if the issue is truly solved. If the user replies with satisfaction or does not reply after three days, then the card will be moved to *Closed* column. Otherwise, if the user gives feedback that the issue still persists, the case will be returned to *Pending Acknowledgement*, restarting the whole process again. If the issue requires managerial decision, it will be moved to *Escalated to L3*.

Incorporating this workflow with Trello, we have less trouble when handling user issues. Trello plays its role as a tool to hold information of issues, allowing our team to collaborate on those information systematically. As a result, we can monitoring issue progress and keep the issue to be responded and solved within a certain amount of time, setting Service Level Agreements (SLAs) that can be expected by users.

5.3. User Satisfaction

To evaluate our service, we sent a questionnaire consists of seven questions to our users: four for the cluster and the rest for user support. Cluster questions asked users about system performance, stability, computing capacity, and satisfaction with system performance. For user support, we asked about response time, troubleshooting capability, and satisfaction with the support process. Users give a score ranging from 1 to 5 for each question, where a score of 5 means very satisfied.

The results of user satisfaction survey are shown in Table 6, with a response rate greater than 50%. Overall, more than 80% of users surveyed were satisfied with the TARA service. We got an average score of 4.54 across all areas. System performance received the highest score with a score of 4.84. The lowest score belongs to computing capacity, with a score of 4.04. This low score comes from the demand for more computing resources, especially for compute and GPU nodes. From the survey, we gather user suggestions and deliberate it as one of the key aspects for resource planning and service improvement.

Table 6. Summary of TARA's user satisfaction

Category	Question	Score (out of 5.00)
Cluster	Performance	4.84
	Stability	4.28
	Computing Capacity	4.04
	Usage Satisfaction	4.48
Support Service	Response Time	4.72
	Troubleshooting Capability	4.72
	Service Satisfaction	4.72

6. Lesson Learned

In this section, we will discuss lessons learned from our experience with TARA during its first year of operation. We believe that these lessons are useful for anyone who try to build a large-scale cluster without any prior system.

6.1. System Design

Designing a system with an uncommon hardware specification poses two major challenges: performance expectation and vendor competition. During TARA design process, octa-socket machines were a desirable option for memory nodes because of large memory with a benefit of a large core count. In reality, while the memory node has served its purpose for memory-intensive application, we also observe that its parallel efficiency is low for other interested applications. This makes the node less attractive for other uses, resulting in under-utilization of computing resources. Such pitfall could be avoided if we had its performance information at hand. However, the rarity of such machine makes finding the performance information challenging.

Another challenge with octa-socket machine is that it significantly reduced a number of participated vendors. Only a small number of vendors had such machine in their catalogue. As a result, the procurement became less competitive. While we could not gauge the impact, we saw several vendors dropped out after TARA specification was published.

6.2. Cluster Management

Working with vendors is crucial for efficient deployment. We asked our vendor to handle the parallel file system and administrative node while our team focused on HPC-related systems, i.e. frontend and computing nodes, job scheduling, and module system. This work distribution allowed us to focus on the core services and speed up the system deployment.

Partnership with other HPC centers helps speedup the quality and features of HPC services and operations. Our partners have taught us several best practices, which significantly reduce learning time. Slurm, EasyBuild, and Singularity were introduced to us as a part of such collaboration. As a result, we had more time to spend on TARA-specific issues, such as accounting and utilities. Moreover, the partners also help pointing out several mistakes in their early stage.

Workforce is also an important factor for managing HPC clusters. Finding HPC experts is hard while career support is not well established. As of now, we are in the process of building up the career path for our HPC experts. On the other side, we are also developing an on-boarding training to quickly introduced new non-expert members to catch up with the team. While we are still in a learning and developing process for this aspect, we believe that it is worth mentioning to raise an awareness for a new HPC center.

6.3. User Support

Good user experience with the cluster is as important as providing the cluster. User experience can greatly varies. Some might have experience with large cluster while some are only familiar with workstations and desktops. Moreover, users from different fields also have different approaches and expectation for using the cluster. Thus, bringing a good user experience cannot be done using a single holistic solution.

From our experience, getting start sessions are critical for raising awareness and bringing users to the cluster. As majority of our users do not have experience with time-sharing clusters, even after a getting start session, there are some barriers preventing them from using the cluster. One example of such barriers is a convenience of self-managed machines, e.g. global software installation and root privilege, which is not possible for HPC clusters. The users might also expect to access computing nodes in the same fashion as their own desktop. Engineering support and software availability are another critical barriers. Setting up development environments or installing software is too troublesome for some users, especially with restrictions of time-sharing clusters.

What we have learned so far is that communication is crucial and an in-person consulting is required to ultimately resolve the problem. As such, we found that having domain experts in the team is critical to the success of this type of user support. While the method might be inefficient and takes time, we have succeeded to bring in a few research groups to use TARA.

We found that Trello [1] is a good beginner tools for issue management because of its simplicity. While its functions might be limited and require manual management, it is easy enough to introduce the tool with our support process to a new member. We have also tried other helpdesk solutions but found that they have steep learning curve and are too complex for our need. However, when our service is expanded nation-wide, we might need to reconsider these options.

7. Conclusion

In this paper, we have described TARA, the first HPC cluster designed and operated by NSTDA Supercomputer Center (ThaiSC), from its design and management to user support and service operations. We also discussed lessons learned during our first year of operating TARA. This paper started with the requirements gathered to design TARA cluster accordingly. The result is a heterogeneous cluster combining with compute node, memory node, gpu node, and dgx node designed to support a broad range of HPC and

computational applications. In addition, we have presented the technical and policy management of TARA cluster, along with our lesson learned during the first year of operation. These information and experiences should be beneficial for HPC experts, HPC centers, and IT facility staffs in general.

References

[1] Atlassian-Trello. About trello - what's behind the boards, 2020.
[2] R.H. Castain, J. Hursey, A. Bouteiller, D. Solt, Pmix: Process management for exascale environments, Parallel Computing 79 (2018) 9–29.
[3] M. Hennecke. DSS-G declustered raid technology and rebuild performance. Technical report, Lenovo (2019).
[4] K. Hoste, J. Timmerman, A. Georges, S.D. Weirdt, Easybuild: Building software with ease, In 2012 SC Companion: High Performance Computing, Networking Storage and Analysis (2012) 572–582.
[5] G.M. Kurtzer, V. Sochat, M.W. Bauer, Singularity: Scientific containers for mobility of compute. PLOS ONE 12 (5) (2017) 1–20.
[6] C.E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. IEEE Transactions on Computers, C-34 (10) (1985) 892–901.
[7] R. McLay, K.W. Schulz, W.L. Barth, T. Minyard, Best practices for the deployment and management of production hpc clusters, In SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (2011) 1–11.
[8] D. Merkel, Docker: Lightweight linux containers for consistent development and deployment, Linux J. 239 (2014).
[9] PMIx-Process Manager Interface-Exascale. https://pmix.org/.
[10] F. Schmuck, R. Haskin. GPFS: A shared-disk file system for large computing clusters, In Proceedings of the 1st USENIX Conference on File and Storage Technologies, FAST 02, page 19–es, USA, 2002. USENIX Association.
[11] P. Shamis, M.G. Venkata, M.G. Lopez, M.B. Baker, O. Hernandez, Y. Itigin, M. Dubman, G. Shainer, R.L. Graham, L. Liss, Y. Shahar, S. Potluri, D. Rossetti, D. Becker, D. Poole, C. Lamb, S. Kumar, C. Stunkel, G. Bosilca, A. Bouteiller, Ucx: An open source framework for hpc network apis and beyond, In 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects (2015) 40–43.
[12] The Unified Communication X Library. http://www.openucx.org.
[13] xCAT-Extreme Cloud Administration Toolkit. https://xcat.org/.
[14] A.B. Yoo, M.A. Jette, M. Grondona, Slurm: Simple linux utility for resource management, In D. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, Job Scheduling Strategies for Parallel Processing, pages 44–60, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.