The Journal of Supercomputing
https://doi.org/10.1007/s11227-024-06041-9

The Egyptian national HPC grid (EN-HPCG): open-source Slurm implementation from cluster to grid approach

Mohammed Elshambakey[1] · Aya I. Maiyza[1] · Mona S. Kashkoush[1] · Ghada M. Fathy[1] · Hanan A. Hassan[1]

Accepted: 4 March 2024

Abstract

Recently, Egypt has recognized the pivotal role of High Performance Computing in advancing science and innovation. Additionally, Egypt realizes the importance of collaboration between different institutions and universities to consolidate their own computational and data resources into a unified platform to serve different disciplines (e.g., scientific, industrial, governmental). Otherwise, additional resources would be needed to be purchased with the associated cost, effort, and time difficulties (e.g., setup, administration, maintenance, etc.). Thus, this paper delves into the architecture and capabilities of the EN-HPCG grid using two different workload management systems: (i) Slurm (Open-Source) and (ii) PBS Pro (Licensed). This paper compares the performance of the grid between Slurm and PBS Pro in specific high-throughput computing (HTC) applications using the NAS Grid parallel benchmark (NGB) to determine which workload manager is more suitable for EN-HPCG. The evaluation includes grid-level performance metrics such as throughput, and the number of tasks completed as a function of time. Also, the presented methodology aims to assist potential partners in their decision-making process to join the EN-HPCG grid, with a focus on the site speed-up metric. Our results showed that, unless an open-source solution without cost and license problems is an obligation (in which case, Slurm is the viable solution), then it is not advisable to integrate a cluster with high-speed hardware with a cluster possessing outdated hardware when using the Slurm scheduler. In contrast, the PBS Pro scheduler takes into account online decision-making in a dynamic environment using a unified grid.

Keywords High-performance computing (HPC) · Grid · NGB benchmark · HPC cluster speedup · Grid performance metrics · EN-HPCG · Slurm · PBS scheduler · HPC grid profiling

Extended author information available on the last page of the article

1 Introduction

High-performance computing (HPC) is essential for a variety of fields, including data analytics, scientific research, and intricate computer simulations. Thorough monitoring and analysis are essential for optimizing cluster performance, wise resource allocation, and productive workload management [1].

The technology of a high-performance computing grid refers to a distributed computing infrastructure that uses linked computing resources to tackle complicated problems by utilizing their combined capacity. Clusters, supercomputers, servers, and even individual workstations might be considered among these resources. Because HPC grids are built to perform large-scale parallel processing jobs, a variety of scientific, engineering, and research applications can benefit from their use [2].

Resource management is of paramount importance in high-performance computing grid environments due to several critical factors that influence the efficiency, reliability, and overall success of computational tasks [3]. Efficient resource management ensures that computing resources, including processors, memory, storage, and network bandwidth, are used optimally. This optimization leads to better performance, faster job completion, and reduced idle time for hardware components. Resource management helps balance workloads across the HPC grid system, preventing individual nodes or clusters from becoming overloaded while others remain underutilized [4, 5]. This balance is crucial to maintaining the stability of the system and maximizing overall throughput.

PBS Pro and Slurm are both job scheduling and resource management systems commonly used in high-performance computing (HPC) environments. Additionally, PBS Pro and Slurm support multi-clusters or HPC grids. Although they serve similar purposes, they have different architectures and approaches.

PBS supports peer scheduling between multiple PBS clusters. This means that PBS clusters can communicate with each other directly to schedule jobs [6]. Administrators can also set up PBS to schedule jobs to particular clusters according to their resource needs by using PBS Multi-sched partitions. Moreover, it is possible to configure PBS to route jobs to particular clusters according to their job type or other criteria using qlists and PBS routing queues [7].

Conversely, Slurm facilitates *multi-cluster* scheduling [8, 9] via connecting multiple clusters to a common database, or having the databases of the different clusters communicate with each other. Additionally, Slurm clusters can merge into a single logical cluster using *federation* [10] which acts like *multi-cluster* but with coherent jobs IDs among all clusters. The *federation* can then receive job submissions from users, just like a single cluster would. However, *federation* allows job arrays to run only on the origin cluster (i.e., the cluster to which jobs were submitted). Thus, we consider only Slurm *multi-cluster*, not *federation*, in this paper. Another feature that Slurm supports made by authors in [11] is "hierarchy," which enables administrators to organize Slurm clusters in a hierarchy. This can be helpful when overseeing sizable clusters with several administrative tiers. "Partitions", another feature that Slurm provides [12], enable administrators to combine resources and designate them for particular users or groups. This can be helpful when managing resources on multi-cluster systems.

There are several benchmark tools available for assessing resource management tools with grid performance, such as the NAS Grid parallel benchmark. [NGB] [13, 14] and GridBench [15]. NASA created a set of benchmark programs known as the NAS (NASA Advanced Supercomputing) Grid Parallel Benchmarks, or NGB, to assess the functionality of HPC grid systems. In the HPC field, these benchmarks are frequently used to evaluate and contrast the computational capacities of various systems. They are made to mimic the amount of computing required for different engineering and scientific applications [13, 14].

Another benchmark tool is known as GridBench, and it consists of a set of benchmark tests that simulate typical grid workloads. These benchmarks are intended to approximate performance metrics for different Grid setups, identify key elements influencing the overall performance of applications, and provide application developers with preliminary estimates of the expected application performance [15].

The recent white paper in [16] highlights PBS Pro, part of the Altair HPCWorks platform, as a superior solution to open-source job schedulers. Australia's National Computational Infrastructure (NCI) chose PBS Pro, which demonstrated superior flexibility and reliability as a workload manager, providing confidence and comfort for a long-term partnership [17]. After over 20 years with the custom workload manager Cobalt, Argonne Lab switched to PBS Pro due to its robustness, scalability, and commercial support [18]. Kyoto University acknowledged the significance of the highly customizable nature of PBS Pro as a crucial feature that enhances productivity in cluster management [19]. PUNCH Torino recognized that by transitioning to PBS Pro, the resulting infrastructure not only facilitates the efficient completion of critical engineering tasks but also liberates their team from the challenges associated with managing an on-premises data center, providing a streamlined and headache-free operational environment [20]. Additionally, the National Supercomputing Center (NSCC) in Singapore acknowledges that PBS Pro caters to both current requirements and future demands, simplifying tasks such as job submission and management and facilitating secure data management along with remote 3D visualization. [21]. Australian Bureau of Meteorology chose PBS Pro because it takes into consideration the criticality of a whole-system outage [22].

Therefore, recently, we built the Egyptian National HPC Grid (EN-HPCG), which is considered to be the first national implementation of the grid concept rather than the cluster model [23]. The primary goal of EN-HPCG is to unify High-Performance Computing (HPC) resources in Egypt, establishing a national, intelligent, and diverse HPC grid. This grid aims to connect various national research institutes and university-based HPC facilities. The current consortium involves three key participants: the Informatics Research Institute (IRI) at the City for Scientific Research and Technological Applications (SRTA-City), the Faculty of Post-graduate Studies for Nanotechnology at Cairo University, Sheikh Zayed Branch, and the Faculty of Science at Ain Shams University (ASU), under the supervision of Academy of Scientific Research and Technology (ASRT). In this phase of the EN-HPCG grid, we utilize PBS Pro to leverage its benefits, including stable 24/7 support, the user portal, the admin portal, and a strong history of peer scheduling features.

Therefore, in this paper, we primarily evaluate popular resource management solutions, such as SLURM and PBS Pro, using the NGB benchmark on the Egyptian National HPC Grid (EN-HPCG). This study guides our decision to transition to the open-source Slurm workload management system in our EN-HPCG grid, aiming to minimize the cost of EN-HPCG sustainability. The paper is organized as follows Sect. 2 illustrates the related work of using multi-clusters, Sect. 3 presents the resources management system tools built by SLURM and PBS-Pro, Sect. 4 describes the experimental setup, Sect. 5 illustrates the evaluation experiments and results and Sect. 6 concludes the paper and presents future work.

Table 1 presents a comprehensive list of the abbreviations and terminologies used throughout this paper.

Table 1 List of abbreviations and terminologies

Abbreviation	Meaning
ASU	Faculty of science at ain shams university
CLI	Command-line interface
EN-HPCG	Egyptian national high-performance computing grid
GUI	Graphical user interface
HPC	High performance computing
HTC	High-throughput computing
IRI	The informatics research institute
PBS	Portable batch system
SLURM	Simple linux utility for resource management
NGB	NAS Grid parallel benchmark
NPB	Nas parallel benchmark
SRTA-city	City of scientific research and technological applications
QoS	Quality of service
SP	Scalar Penta-diagonal solver
ED	Embarrassingly distributed
NQS	Unix-based network queuing system
MOPS	Million operations per second
S_site	System/workflow speed-up
T_Grid	Workflow wall time using all grid resources
T_site	The optimum execution time using only the resources available in a given site
U_site	User speed-up
J	The number of completed jobs
Δt	Total time in seconds
$\hat{T}_{site}$	The mean duration between the start of the job and its completion using only the resources available
$\hat{T}_{Grid}$	The mean duration using all grid resources
n(t)	The number of tasks completed as a function of time
N_i	The number of processors used for job i

2 Related work

Many years ago, researchers delving into grid middleware laid the foundation for the sophisticated layer of software that today plays a pivotal role in managing the intricacies of computational grids. Managing a computational grid, which inherently comprises machines with various characteristics, poses a multifaceted challenge. Executing a job on a computational grid requires tasks such as establishing machine profiles, identifying allocated resources, evaluating specific work requirements, segmenting the job, and distributing the workload based on the available nodes and resources. To alleviate application developers from the intricacies associated with these actions, computational grids incorporate a software layer designed to conceal the complexities of this heterogeneous environment. Within this layer, various protocols and functions are imperative, providing support for various elements of the grid and facilitating adaptation to different operating systems, file systems, and communication protocols [24]. The middleware assumes a central role in computational grids, functioning across service, resource, and connectivity layers [25]. Comprising a synthesis of protocols, services, APIs, and SDKs [26], the middleware serves as a critical component that masks the hardware intricacies and communication complexities inherent in a grid environment. Leveraging a computational grid for diverse purposes becomes feasible only through the middleware layer's ability to obscure these underlying complexities. Noteworthy among the prominent projects dedicated to middleware development for grids are:

- **Globus** [27]: The Globus project, managed by the "Globus Alliance," features collaborative efforts involving institutions like the Argonne National Laboratory and the Institute of Information Sciences. This open-source toolkit serves as a facilitator for constructing computational grids and grid-based applications. Its capabilities extend beyond corporate, institutional, and geographic boundaries, ensuring seamless collaboration while preserving local autonomy.
- **Unicore** [28]: The Unicorn Project, supported by financial backing from the German Ministry of Education and Research and in partnership with entities such as ZAM and Deutcher, establishes a Java-based grid computing environment. This system ensures uninterrupted and secure entry into distributed resources, promoting smooth integration and efficient utilization.
- **Boinc** [29]: BOINC, developed by Berkeley University, is designed to be compatible with research projects such as SETI, focusing on the search for extraterrestrial intelligence. This open-source platform is utilized in scientific endeavors spanning diverse fields, including astrophysics, chemistry, molecular biology, medicine, and climatology, harnessing the computational power of personal computers.
- **HTCondor** [30]: HTCondor, also known as Condor, originated at the University of Wisconsin-Madison with the initial purpose of evaluating the advantages of intensive computing in campus research. This system excels in handling tasks related to computationally intensive or high-throughput computing

(HTC). It is specifically designed to accommodate large-scale, power-tolerant processing tasks for extended durations, spanning weeks to months.

Various articles and studies addressed optimization techniques and advancements in grid computing. The authors of [31] focused on optimizing Aurora middleware to reduce the computing environment overhead. Subsequent studies explored various aspects of grid computing, including QoS treatment in [32] and a multi-agent-based peer-to-peer network proposal in [33].

Further contributions to the enhancement of grid architecture were found in studies such as the development of an oriented grid for high-performance computing applications in [34] and improvements to the Globus middleware to increase throughput in bioinformatics applications in [35].

Various proposals and frameworks were presented, ranging from resource management in bio-grids [36] to integration of mobile computing with the grid in [37], interaction with running jobs in [38], and the application of peer-to-peer approaches in volunteer computing platforms in [39, 40]. Various articles covered different topics, including a new architecture for computational grids in [41], a comparison of grid and cloud computing in [42], web performance enhancement using the grid in [43], and a comparison of atmospheric data analysis models in cluster and computational grid environments in [44]. Middleware like Agent Team Works was detailed in [45], while the use of Hadoop in computational grids for a smart marketing model was proposed in [46].

Addressing the challenges posed by the complexity of developing grid applications and the limitations on the types of discovered resources when directly utilizing the jobs, the grid initiative introduces several technologies. Therefore, many alternatives were explored by the scientific and industrial communities to simulate multicluster concepts. Interactive methods were employed with the aim of creating a programming interface capable of defining both the computation process and the interaction with distributed resources. Most of the programming models utilized in this industry and applied within user programs were separated from the work distribution process by the manual submission of job description files to a batch system scheduler such as HTCondor [47], Slurm [48], and PBS [7].

The authors of [49] described the Coffea-casa (University of Lincoln, Nebraska) prototype analytical facility in the United States. This service used Dask for the computation distribution. It incorporated dedicated resources allotted through Fair-Share using an HTCondor scheduler and was built on top of a local Kubernetes cluster. Another analysis facility prototype [50] was developed at Fermilab with the label "Elastic Analysis Facility." The analytic facility implementation presented in this study aimed to integrate the various geographic clusters at INFN with a novel scheduler-client connection system, following the same general direction as the prototypes just discussed.

Numerous examples of HPC Grids at universities and scientific institutions across various locations can be found in non-federated approaches, such as [51–55], and in the federated approaches, such as [56]. The existing literature highlights numerous challenges in constructing a robust architecture for the HPC grid. These challenges encompass aspects like heterogeneity, programmability, scalability, and the interoperability and coupling of high-performance architectures or networks of computing nodes. Hence, in this paper, as a beginning, we intend our evaluation of our EN-HPCG grid to focus on widely used homogeneous resource management solutions, including SLURM and PBS Pro, employing the NGB Benchmark. Then, in our future work, we will extend our focus to address additional challenges, including the integration of heterogeneous resource management, throughput improvement, and the implementation of smart job allocation strategies.

3 Resource management systems (Slurm and PBS-Pro)

3.1 Slurm

Slurm, which stands for “Simple Linux Utility for Resource Management,” is an open-source cluster management and job scheduling system. It is widely used in high-performance computing (HPC) environments to allocate and manage computing resources such as CPU cores, memory, and GPUs efficiently across a cluster of interconnected computers.

Slurm provides a flexible and scalable framework for managing jobs and workflows on HPC systems. It allows users to submit and schedule jobs, monitor their progress, and control resource allocation. Slurm supports a variety of job types, including batch jobs, interactive jobs, and parallel jobs, making it suitable for a wide range of scientific and computational workloads.

One of the key features of Slurm is its ability to handle complex job dependencies and priorities. It supports job dependencies to ensure that certain jobs are executed only after their prerequisite jobs have been completed successfully. Additionally, slurm allows users to specify job priorities, enabling important or time-critical jobs to be allocated resources ahead of lower-priority jobs.

Slurm provides a command-line interface (CLI) for users to interact with the system and perform various tasks, such as submitting jobs, querying job status, and managing resources. It also offers a web-based graphical user interface (GUI) called "sview" to visualize and monitor the activity of the cluster.

In addition to job management, Slurm provides extensive accounting and reporting capabilities, allowing administrators to track resource usage, generate usage reports and enforce resource limits. It supports authentication and authorization mechanisms to ensure secure access to the system and resource allocation.

Slurm is highly configurable and can be customized to meet the specific requirements of different HPC environments. It is widely adopted in academic and research institutions, government laboratories, and industry settings to manage large-scale computational clusters and supercomputers.

Overall, Slurm plays a crucial role in optimizing resource utilization, improving job throughput, and facilitating efficient management of HPC systems, making it a popular choice for organizations working with computationally intensive workloads.

3.2 PBS

The lineage of the Portable Batch System (PBS) scheduler family can be traced directly back to Unix-based Network Queuing System (NQS), the inaugural batch scheduler developed with NASA funding. PBS is a workload manager and high performance computing (HPC) task scheduler that is designed to effectively manage and optimize the distribution of computing resources in demanding computing settings. Altair Engineering, a multinational technology business with a focus on data analytics, high-performance computing, and simulation, produced the commercial solution [6].

PBS Pro provides a robust framework for job submission, scheduling, and resource management in HPC clusters, supercomputers, and cloud environments. It allows organizations to maximize the utilization of their computing resources while ensuring fair and efficient job execution [57]. PBS have the following Key Features:

- **Policy-based scheduling:** PBS Pro employs a policy-based scheduler that allows administrators to define various policies for job prioritization, fair-share scheduling, and resource allocation.
- **Scalability:** The system is designed to scale efficiently, support large-scale parallel processing, and accommodate the computational demands of modern scientific and industrial applications.
- **Flexibility in job types:** It supports a wide range of job types, including parallel, serial, array, and checkpointing jobs, making it suitable for diverse scientific and engineering workloads.
- **Advanced job management:** PBS Pro includes advanced features such as job arrays, custom resource configurations, and checkpointing mechanisms to enhance job management capabilities.
- **Resource monitoring:** The system provides real-time monitoring of resources, allowing administrators to track system usage, diagnose issues, and optimize resource allocation.
- **Extensible architecture:** PBS Pro has a modular and extensible architecture, allowing organizations to tailor the system to their specific needs and integrate it with other tools and applications.
- **Multi-cluster support:** PBS Pro enables organizations to create a unified computing environment from distributed resources by managing multiple clusters. It uses peer scheduling, where the available resources in a local cluster are used first, and any remaining pending jobs are sent to the available connected clusters for execution. Once a job finishes, it is rolled back to the local cluster.

3.3 Feature comparison

Table 2 illustrates a comparison of Slurm and Altair PBS pro features. Although Slurm is an open-source scheduler, PBS Pro is superior in that it supports Windows operating systems and has a graphical user interface portal. Windows support could

Table 2 Features comparison among job schedulers (Inspired by [57])

Features	Slurm	PBS-Pro
Type	HPC	HPC
Actively developed	✓	✓
Cost/Licensing	Open source	$$
OS support	Linux [12]	Linux, Windows [59]
Language support	All (Java, Python,..)	All (Java, Python,..)
Access control/security	✓	✓
Grid support	Multi-cluster Federation [12]	Peer scheduling [7]
GUI (user portal)	–	Altair access [58]
GUI (admin portal)	sview [60]	Altair control [61]
Cloud bursting	✓ [62, 63]	✓ [63]
Parallel and array jobs	✓	✓
Queue support	✓	✓
User-friendly	✓	✓
Containers support	✓	✓
GPU support	✓	✓
Job type submission	–	✓
User-friendly	–	✓

be beneficial for software that is only available for Windows. However, containers could be a solution to run Windows-based software. Altair Access facilitates the seamless submission, monitoring, and visualization of HPC jobs on remote clusters, cloud infrastructure, and other computational resources through a web-based viewer. And, it offers ongoing support and comprehensive documentation, ensuring users have access to assistance and resources for optimal utilization of the platform [58]. In addition, PBS Pro allows users to submit their jobs based on job type; the routing queue will route the job automatically to the particular clusters that avail the required software.

4 Experiments

4.1 Benchmark implementation

In this paper, to assess our EN-HPCG testbed, we employed the Nas Grid Benchmark (NGB) serial version 3.1 [14], a derivative of the Nas Parallel Benchmark (NPB). NPB is a widely recognized benchmark widely used to evaluate HPC cluster [64, 65]. we had to exhaust the grid resources for evaluating the grid performance, so we used the Embarrassingly Distributed (ED) kernel in the NGB benchmark, which illustrates the important category of grid applications that entails numerous independent executions of the same program with different input parameters. Such experiments are frequently carried out at NASA using Scalar Penta-diagonal solver (SP) as flow solvers [66]. There is no communication between any of the NQS kernels. There are several classes for the ED kernel based on its sizes S, W, A, B, C, D, and E with sizes of (9 x 1), (9 x 1), (9 x 1), (18 x 1), (36 x 1), (72 x 1), and (144 x 1), respectively. Figure 1 shows the data flow graph of the ED benchmark [67].

4.2 Slurm testbed settings

In this paper, we dedicate a part of the entire EN-HPCG grid to evaluate the Slurm scheduler compared to the PBS Pro that is already used in the Egyptian grid [23]. The grid testbed consists of two clusters, one allocated at the City of Scientific Research and Technological Applications (SRTA), and the other cluster allocated in the Faculty of Science at Ain Shams University (ASU). The cluster at SRTA called **slurmcluster2** and The cluster at ASU called **asuslurm** that are connected with the Slurm muti-cluster mode [68] as shown in Fig. 2. In multi-cluster mode, different clusters can communicate with one another, where a job can be submitted from one cluster to another to be executed there if convenient.

The head node of slurmcluster2 (headnode) hosts the slurmctld service to control the cluster. While the head node of asuslurm (asuslrhd) hosts slurmctld service to control the asuslurm cluster, as well as slurmd service to act as a compute node itself.

The headnodevm hosts the MariaDB database server, and the slurmdbd service for accounting services for both slurmcluster2 and asuslurm clusters. headnodevm also hosts Chronyd service for time synchronization between all nodes in both clusters. Time synchronization is important in Slurm multi-cluster mode because the choice of which cluster to be allocated a submitted job depends on the cluster that provides the earliest start time for the job. The common services on the headnodevm enable both slurmcluster2 and asuslurm to communicate in the multi-cluster mode and federation mode (i.e., all clusters can be viewed as a single cluster, but still one job can be executed on one cluster only). Future work can involve using separate database servers for each cluster in the multi-cluster mode. FreeIPA server is used to create the same user(s) across all clusters. NFS directories are shared for each user across each cluster nodes to simplify sharing resources (e.g., executable scripts, submission files,.. etc.). Users' authentication is done using the same Munge keys that are installed in a common directory

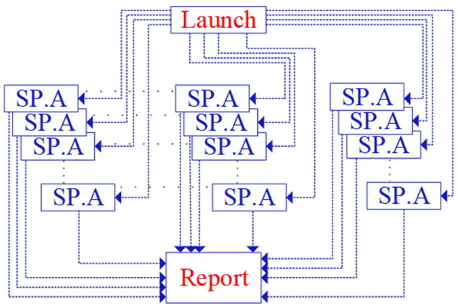

Fig. 1 Data flow graph of ED benchmark [67]

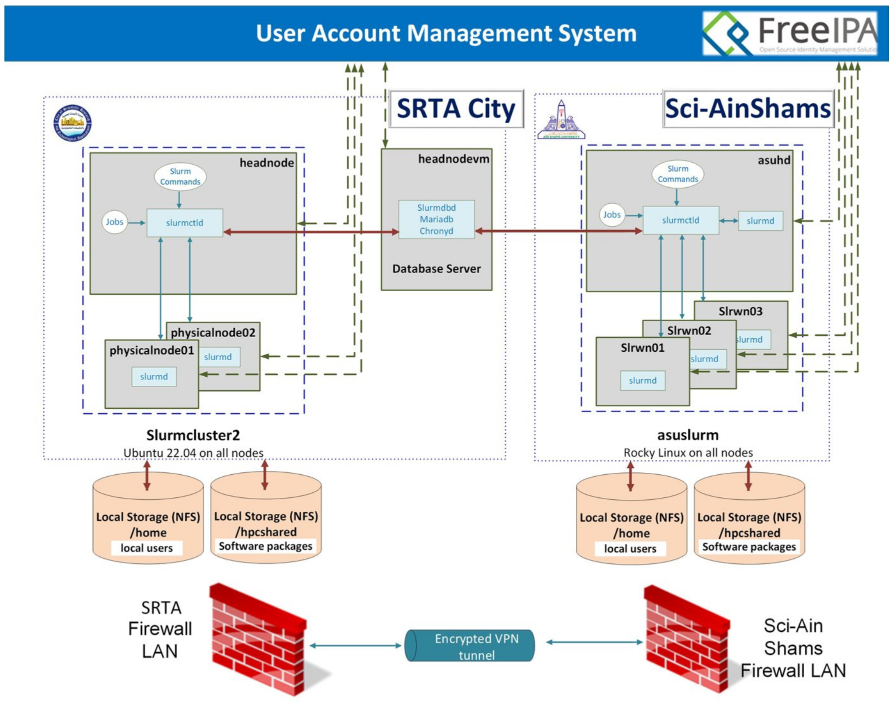

Fig. 2 Architecture of slurm multi-cluster testbed

path, in each cluster, to enable communication between different clusters. Future work may involve using separate munge keys for inter and intra-cluster(s) communication.

All clusters have Slurm 23.11.0-0rc1 installed from the source repository, which is the latest Slurm version (at the time of writing this document).

Nodes specifications of SRTA and ASU clusters are shown in Table 3.

Table 3 SRTA and ASU nodes specifications

	SRTA	ASU
No. nodes	2	4
Node names	physicalnode0[1-2]	asuslrhd, slrwn0[1-3]
No. CPUs per node	16 (single-threaded)/ 32 (multi-threading)	8
CPU speed (GHz)	2.7	2
CPU type	Intel(R) Xeon(R) CPU E5-2680	Intel(R) Xeon(R) E5335
Memory (GB)	physicalnode01: 193.363 physicalnode02: 128.851	Head node: 15.692 The others: 7.64

Table 4 Execution time and memory requirements of **sp.A** on different computational hosts

	Exec_time (sec)	Mops mops	Mem (MB)
asuslrhd	78.78	1079.07	102.31808
slrwn01	79.13	1074.24	102.322176
slrwn02	79.20	1073.31	102.322176
slrwn03	79.21	1073.19	102.322176
physicalnode01	32.25	2635.60	64.581632
physicalnode02	32.18	2642.06	64.581632

Table 5 Average, minimum and maximum execution time of sp.A on different computational Slurm nodes

	Avg_exec _time (sec)	Max_exec _time (sec)	Min_exec _time (sec)
asuslrhd	248	258	201
slrwn01	248.53	258	204
slrwn02	248.9	257	205
slrwn03	248.8	259	204
physicalnode01	46	48	45
physicalnode02	71	74	70

4.2.1 Benchmarking sp.A on different computational Slurm nodes

We ran sp.A on different computational nodes on SRTA and ASU to get (almost) correct estimations of execution times and memory requirements of each run (without Slurm). Estimated values are shown in Table 4.

However, it turned out that the concurrent multiple executions of sp.A jobs under Slurm increased each job execution time, as shown in Table 5 for the average, maximum, and minimum execution times. We justify the increase in job execution time due to job arrival and completion, which trigger the Slurm scheduler. When the scheduler is triggered, it may interrupt the execution of the currently running jobs to find a suitable allocation for the newly received jobs. Furthermore, performance degradation can occur when processes run on the same multi-core CPU and share resources such as last-level caches, memory controllers, system request queues, and prefetch bandwidth. This degradation could be as high as 200%, relative to the execution of processes in isolation [69]. Also, This could be the reason for the difference in execution time between the physicalnode01 ($\simeq$ 46 Sec) and physicalnode02 ($\simeq$ 71 Sec), as shown in Table 5.

4.3 PBS Pro testbed settings

To evaluate the performance of PBS Pro compared to Slurm, we used the same physical machines. The PBS Pro testbed consists of one cluster at SRTA (called

head-node) and another at ASU (called **asuhd**). The grid system consists of six main components, as shown in Fig. 3:

- **Altair PBS professional** [70]: It is employed for the supervision and coordination of grid resources, functioning as a remote resource management system within the grid context. Altair PBS Professional adeptly oversees high-performance computing (HPC) tasks and orchestrates HPC workloads across the entire grid computing infrastructure. The scalability of PBS Professional enables seamless support for systems of varying sizes, ranging from clusters to extensive supercomputers. This ensures optimal utilization of both hardware and software investments, maximizing the benefits for users.
- **Altair access** [58]: Users can submit, monitor, and visualize high-performance computing (HPC) jobs on remote clusters, clouds, and other resources through a Web-based viewer. Additionally, Altair Control [61] provides administrators with

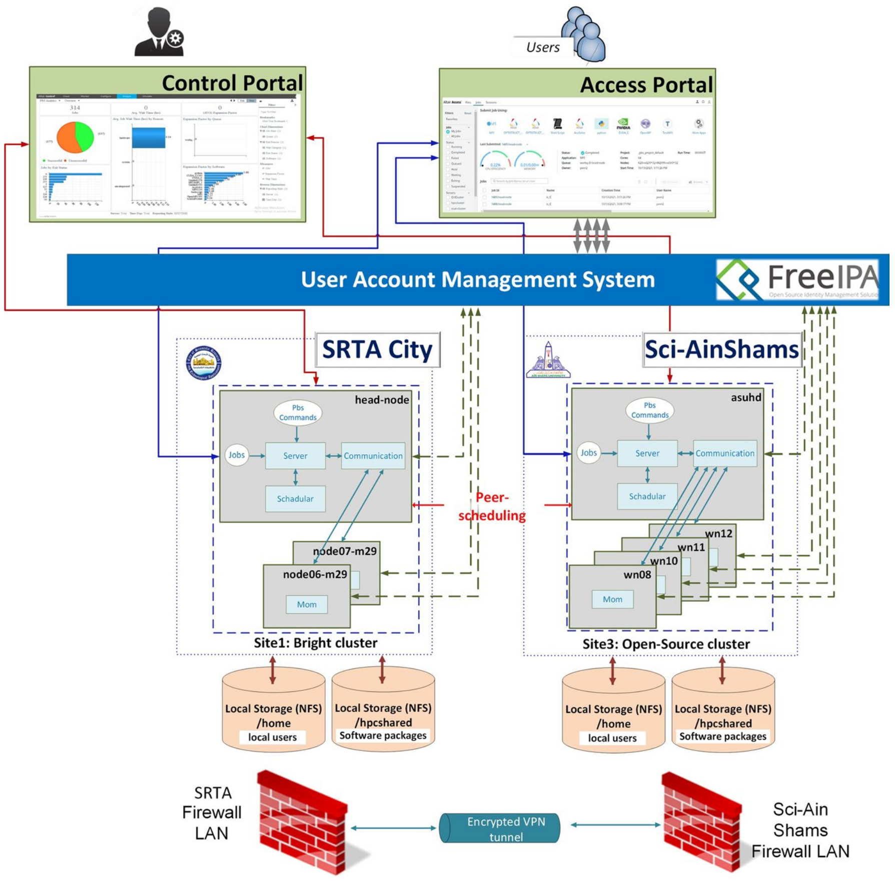

Fig. 3 Architecture of PBS testbed

the ability to control, monitor and manage the configuration environment of an HPC cluster using a Web application.

- **Shared NFS storage:** In the grid, each cluster has access to two NFS storage options: local and shared. The “home” directory is utilized by local users of the cluster to store their personal files. On the other hand, the "hpcshared" directory serves the purpose of storing and sharing all grid-installed software packages, making them accessible to all clusters within the grid. For optimal job execution speed, it is advisable to run applications from the home directory instead of the “hpcshared” directory.
- **FreeIPA DB system:** FreeIPA serves as a comprehensive solution for globally managing users’ accounts and groups [71]. It provides straightforward installation and user-friendly command lines and web-based management tools, empowering Linux administrators to oversee the identification, authentication, and access control aspects of Linux users’ accounts centrally.
- **Peer scheduler:** The peer scheduling feature is activated when users submit jobs to a busy or fully utilized cluster, automatically redirecting the jobs to another available cluster within the grid.

We take some nodes from PBS PRO grid with the same specification of the Slurm testbed to compare the results. PBS testbed specifications of SRTA and ASU clusters are shown in Table 6.

4.3.1 Benchmarking sp.A on different computational nodes

We ran again sp.A on different PBS Pro computational nodes on SRTA and ASU using the PBS pro testbed to obtain (almost) correct estimates of the execution times and memory requirements of each run (without PBS pro). The estimated values are shown in Table 7. Then, we ran concurrent multiple executions of sp.A jobs under PBS pro. Table 8 illustrates the average, maximum, and minimum execution times.

In general, the difference between the execution times between Slurm and PBS Pro, as give in Tables 5, and 8, could be due to the difference between the two

Table 6 SRTA and ASU nodes specifications of PBS Pro testbed

	SRTA	ASU
No. nodes	2	4
Nodes names	node0[6-7]-m29	wn[8,10-12]
No. CPUs per node	16 (single-threaded)/ 32 (multi-threading)	8
CPU speed (GHz)	2.7	2
CPU type	Intel(R) Xeon(R) CPU E5-2680	Intel(R) Xeon(R) E5335
Memory (GB)	128.851	wn08: 15.692 The others: 7.64

Table 7 Execution time and memory requirements of **sp.A** on different computational PBS Pro nodes

	Exec_time (sec)	Mops
wn08	81.14	1047.74
wn10	81.24	1046.43
wn11	81.81	1039.14
wn12	81.77	1039.67
node06-m29	31.84	2669.82
node07-m29	31.82	2671.29

Table 8 Average, minimum and maximum execution time of sp.A on different computational nodes

	Avg_exec _time (sec)	Max_exec _time (sec)	Min_exec _time (sec)
wn08	220.86	232	210
wn10	221.52	227	214
wn11	222.06	234	212
wn12	219.06	230	199
node06-m29	72.43	77	58
node07-m29	74.44	77	68

schedulers in the arrival and completion of jobs. When the scheduler is triggered, it may interrupt the execution of the currently running jobs to find a suitable allocation for the newly received jobs. We notice that the execution times in node06-m29 and node07-m29 are very close, although these servers are the same servers used in the Slurm testbed. We found that the Mom service in PBS Pro detects the available memory of the two nodes to be equal to 128 GB. In contrast, the **slurmd** service detects the available memory of the two nodes approximately equal to 192 and 128 GB, respectively.

4.4 Performance metrics

The performance of the Slurm and PBS Pro testbeds has been assessed using an intrusive benchmark that applies stress to all the testbed's resources ($n \geq N$), where n is the number of jobs, and N is the number of available processors. This experimental approach aims to accurately determine values for *Throughput*, *system Speedup*, *user Speedup*, and *the number of tasks completed as functions of time*.

4.4.1 Throughput

Throughput is a key performance metric that measures the rate at which a system can process a workload or a set of tasks over a given period. Throughput is often expressed in terms of tasks per second. It provides an indication of the system's efficiency in handling a large number of tasks simultaneously.

$$Throughput = \frac{J}{\Delta t} \tag{1}$$

Where J is the number of completed jobs, and Δt is the total time, in seconds, including communication and schedule time overhead, of the completed jobs.

4.4.2 System/workflow speed up

System/workflow speed-up S_{site} is used to evaluate the grid in terms of the execution times of the system or workflow job [72]. A workflow consists of multiple jobs that are usually submitted in bulk. Therefore, the speedup that a user might anticipate when executing a specific class of applications on the Grid can be characterized as

$$S_{site} = \frac{T_{site}}{T_{Grid}} \tag{2}$$

Where T_{Grid} is the workflow wall time using all Grid resources, and T_{site} is the optimum execution time using only the resources available in a given site.

4.4.3 User speed up

User speed-up, U_{site}, is used to evaluate the grid in terms of user perspective (i.e., the speed-up for running a single job submitted by a user using grid resources compared to site resources, instead of the speed-up of running the whole jobs submitted to the grid as defined in 4.4.2). Therefore, We determine the mean duration between the start of the job and its completion using only the resources available ($\hat{T}_{site}$) in a given site divided by the mean duration using all Grid resources ($\hat{T}_{Grid}$) as follows:

$$U_{site} = \frac{\hat{T}_{site}}{\hat{T}_{Grid}} \tag{3}$$

Where:

$$\hat{T}_{site} = \frac{1}{n}\sum_{i\in S} T_i \qquad , i = 0, 1, 2, ..., n-1 \tag{4}$$

$$\hat{T}_{Grid} = \frac{1}{n}\sum_{i\in G} T_i \qquad , i = 0, 1, 2, ..., n-1 \tag{5}$$

Where n is the number of jobs, and T_i seconds is the total time, including communication and schedule time overhead, of the job i.

4.4.4 Number of tasks completed as a function of time

The number of tasks completed as a function of time is given by:

$$n(t) = \sum_{i \in G} N_i E_i \tag{6}$$

Where N_i denotes the number of processors used for job i in the grid (G). E_i takes the value 1 if job i has completed and 0 if it is still running or in a queued state. E_i is calculated by:

$$E_i = \begin{cases} 1, \text{ for } \lfloor \frac{t}{\Delta T_i} \rfloor \geqslant 1 \\ 0, \textit{ otherwise.} \end{cases} \tag{7}$$

Where, ΔT_i seconds represent the total time from the beginning of the experiment to the completion of job i, encompassing communication and schedule time overhead. We drew inspiration from the $n(t)$ equation presented in [72] and subsequently modified it to align with our experiment design. In our adaptation, E_i should be equal to 1 when job i has been completed. This adjustment ensures consistency, addressing the occasional generation of 2 or 3 values in the equation from [72], which can occur when jobs experience prolonged waiting times before execution. In addition, in our case, N_i consistently equals 1, signifying the use of one processor per job.

5 Results

We evaluated the testbed performance in terms of the mentioned metrics in Sect. 4.4) by using the serial Embarrassingly Distributed (ED) kernel in NAS Grid benchmark with Class E (144 tasks) [67].

Section 5.1 shows the experimental results of the grid using Slurm in multi-cluster mode. Then, Sect. 5.2 illustrates comparative study results of using Slurm in multi-cluster mode versus PBS Pro in Peer-scheduling mode. As PBS Pro is already installed on the EN-HPCG, this study is performed to migrate to Slurm as an open-source scheduler if it has proven its effectiveness regarding users and the system.

5.1 Slurm multi-cluster results

To assess the testbed's performance in the presence of an intrusive benchmark in which the number of executed jobs exceeds the number of available processors. We executed 144 jobs from the ED-class E NAS Grid benchmark on our testbed, where the SRTA cluster consists of 2 nodes with 16 cores on each of them and the ASU cluster consists of 4 nodes with 8 nodes on each of them as shown in Table 3.

There are two cases for task submission. The first case is to submit jobs to the ASU cluster (ASU-SRTA), and the second case is to submit jobs to the SRTA cluster (SRTA-ASU). In both cases, tasks can be re-submitted from one cluster to the other if Slurm expects an earlier start time for the job on the other cluster.

Figure 4 shows the grid performance, which is the actual grid throughput measured in jobs per second when running six distinct executions of the benchmark on various days of the week to overcome network bandwidth variations conditions. Because of Slurm scheduling and offloading tasks between two clusters (i.e., Slurm

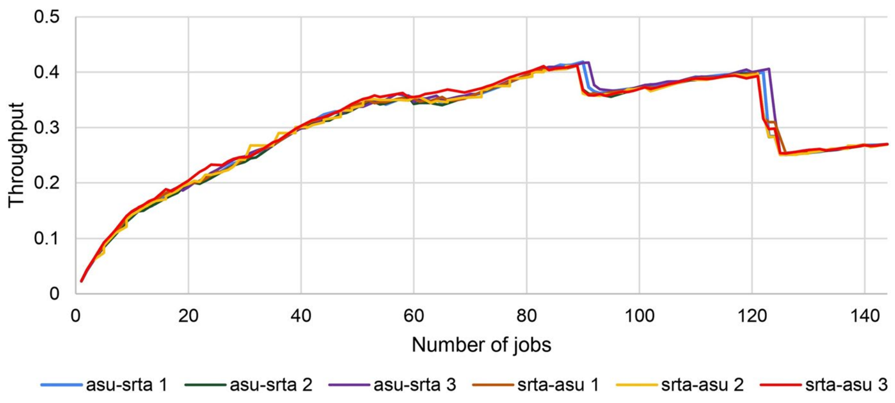

Fig. 4 Grid throughput using Slurm multi-cluster

assigns the job to the cluster with the earliest expected start time), the number of jobs executed on SRTA was 89 jobs and the number of jobs executed on ASU was 55 jobs, in the case of SRTA-ASU. In the case of ASU-SRTA, the number of jobs executed on ASU was 54 jobs and the number of jobs executed on SRTA was 90. Figure 4 shows that the grid obtains the same performance in the two different cases for the job submission. This means that the Slurm behavior does not depend on where the jobs are submitted, takes the offloading decision when the jobs are submitted, and doesn't take into consideration the local cluster for the jobs. It obtained high performance in 90 jobs and 120 jobs, then decreased after 120 jobs.

Figure 5 illustrates the performance of the grid and the performance of the ASU cluster and SRTA cluster without the Slurm multi-cluster scheduling. The performance was calculated by Eq. 1. The performance of the grid, represented by the solid line, is better than the performance of the ASU cluster, shown in the dotted red line, because the ASU cluster has limited assets and outdated hardware and takes more time to execute jobs than SRTA. The execution time for ASU jobs is 248 s,

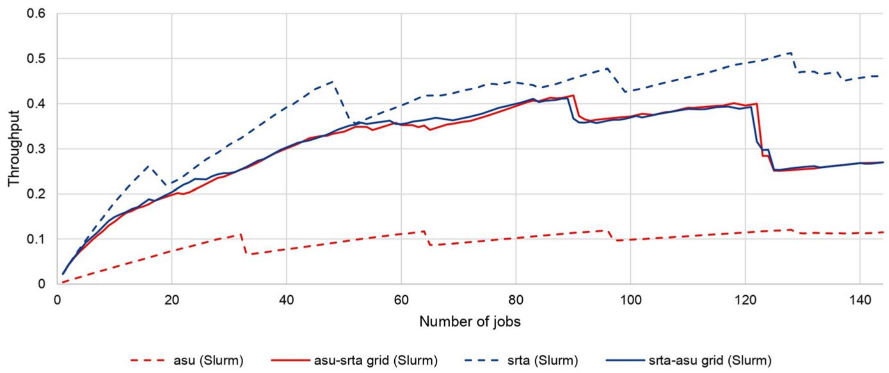

Fig. 5 Grid throughput vs. throughput of each cluster without Slurm multi-cluster scheduling

on average, as shown in Table 5. This means that it is beneficial for ASU to join the grid.

On the other hand, the performance of the SRTA cluster (the dotted blue line) is better than the performance of the grid and the ASU cluster (the solid and dotted red lines, respectively). The average job execution times in SRTA are 46 s and 71 s, respectively, on the two SRTA cluster nodes. Average job execution times on SRTA are less than the average job execution times on ASU (248 s), as shown in Table 5. Additionally, the average job execution times on SRTA are less than those on the grid, as the latter offloads some jobs to the ASU cluster which is slower than the SRTA cluster.

Figure 6 shows the number of executed jobs within a time interval using the grid (solid lines) compared to the number of executed jobs within a time interval using ASU and SRTA clusters (dotted red and dotted blue lines), respectively. The number of completed jobs within a time interval is calculated by Eq. 6. Figure 6 illustrates that executing 144 jobs takes 543 s using grid resources, 312 s using SRTA, and 1254 s using ASU. Figure 6 shows it is very beneficial for ASU to join the grid.

Figure 7 illustrates the speed up for SRTA and ASU clusters and the benefits for some clusters to join the grid. The speed up for each site was calculated by Eq. 2. As seen in Fig. 7, in this scenario, it is more beneficial for the ASU site to join the grid than the SRTA site. The grid is helpful in sites with limited assets or outdated hardware.

Figure 8 shows the speed up for SRTA and ASU clusters from the user's perspective. The user speed-up was calculated by Eq. 3.

5.2 Comparative study results of Slurm and Pbs pro in grid mode

This section compares the performance of Slurm in multi-cluster mode (Sects. 3.2 and 4.3) against the performance of PBS Pro is already installed on the Egyptian National High-Performance Computing Grid (EN-HPCG) [23].

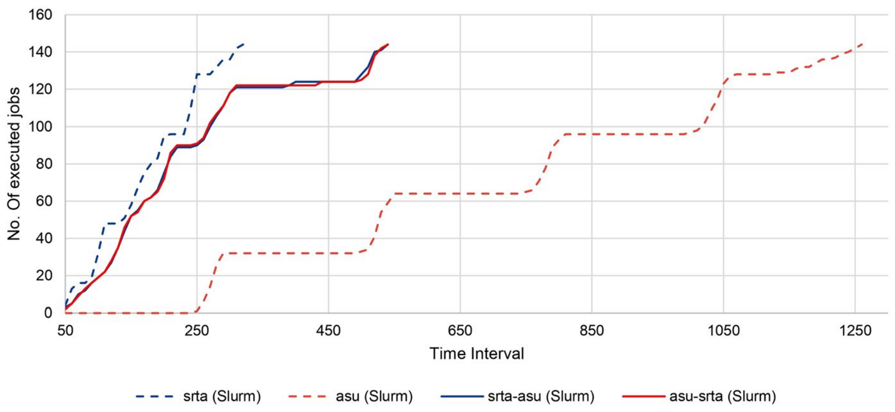

Fig. 6 Number of completed tasks as a function of time using Slurm

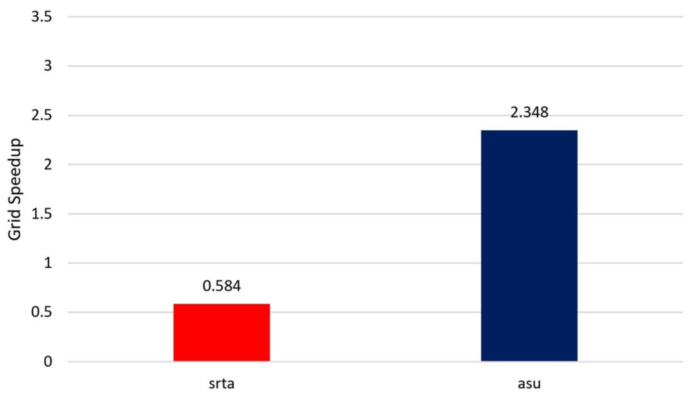

Fig. 7 Grid speedup for system/workflow using Slurm multi-cluster

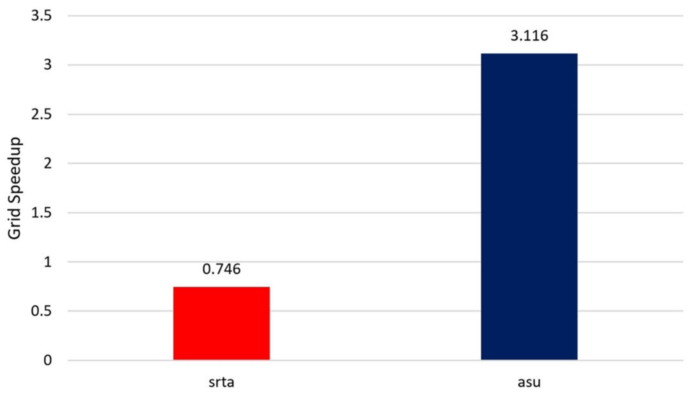

Fig. 8 Grid speedup for user using Slurm multi-cluster

Figures 9 and 10 illustrate the experimental performance of the grid throughput (with PBS Pro peer scheduling, and Slurm multi-cluster mode) compared to the throughput of the SRTA and ASU sites (without PBS Pro peer scheduling, or Slurm multi-cluster mode), respectively. The throughput of the grid is represented by the solid lines, whereas the throughput of each site is plotted as dotted lines. Red lines show the throughput of PBS Pro, whereas blue lines show the throughput of Slurm.

Under PBS Pro peer scheduling (i.e., grid using PBS Pro) and initial job submission to SRTA, 112 jobs were executed on SRTA, whereas ASU received 32 jobs. Throughput generally improves, especially after 96 jobs as illustrated in Fig. 9.

PBS peer scheduling is automatically activated to transfer tasks from a fully utilized site to another available site. The concept of SRTA-ASU involves submitting the entire benchmark workload, consisting of 144 tasks, to the SRTA site. Any excess tasks beyond the capacity of SRTA are then offloaded to the ASU site. Given that each task requires one core, congestion at the SRTA site occurs after 32 tasks,

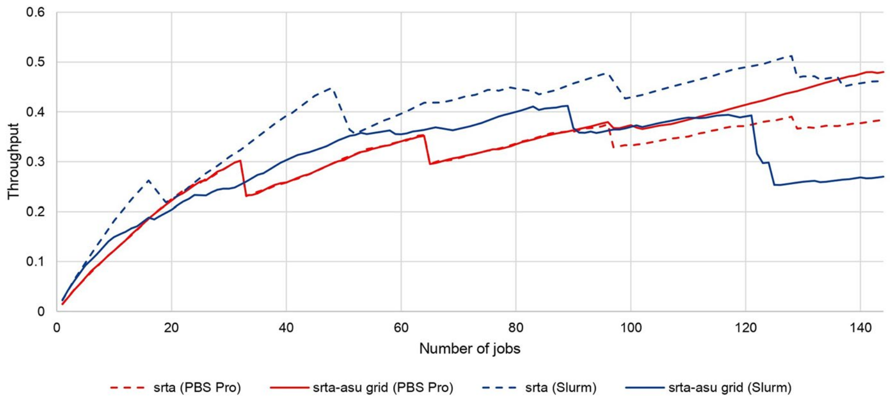

Fig. 9 Grid throughput vs. SRTA cluster throughput without PBS Pro peer scheduling, or Slurm multi-cluster

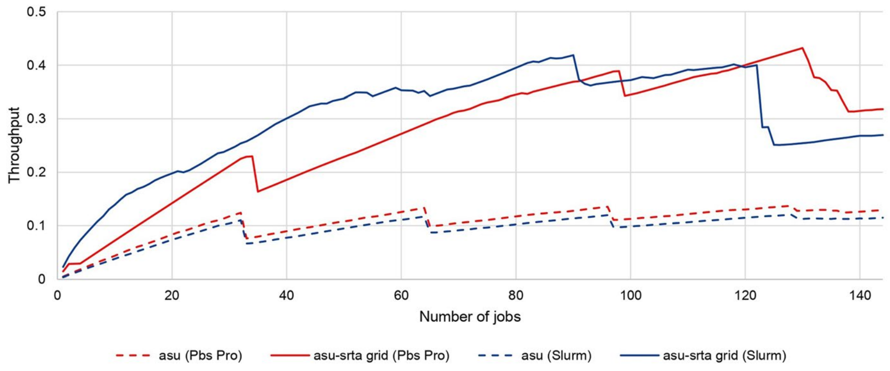

Fig. 10 Grid throughput vs. ASU cluster throughput without PBS Pro peer scheduling, or Slurm multi-cluster

leading to offloading the next 32 tasks to the ASU site. After processing 32 tasks, Asu also becomes busy, and the remaining tasks are queued within the grid until resources become available at either site. Although the ASU cluster is considered a slow cluster with low resources, the improvement occurred because 32 tasks were offloaded to the ASU cluster.

The two solid lines in Fig. 9 illustrate the performance of the grid using PBS Pro peer scheduling (red line) in comparison to the Slurm multi-cluster (blue line) scheduler. The grid performance of the Slurm scheduler is superior until 84 completed tasks. Afterward, the grid performance of both schedulers becomes similar. Subsequently, the grid performance of PBS Pro improves, surpassing that of Slurm, after 104 completed tasks. This is due to the difference in decision-making between peer scheduling in PBS Pro and multi-cluster scheduling in Slurm. PBS peer scheduling decides to offload a task to another cluster when there are sufficient resources available there. In other words, the job migrates to the cluster and starts running immediately in an online decision fashion. In contrast, Slurm multi-cluster decides to offload a job to the cluster with the earliest expected start time considering the cluster queue of running and pending jobs. Once Slurm decides to offload a job to a specific cluster, the job cannot be migrated to any other cluster, even if some resources become available at the other clusters (e.g., some running jobs at the other clusters have finished before their scheduled end times) [68]. Thus, the Slurm multi-cluster makes an offline decision about where each job should be allocated. Accordingly, Slurm decided to offload 55 jobs to the ASU cluster since their submission time. This decision resulted in a poor throughput after the completion of about 112 jobs due to the long running time of jobs at ASU.

When jobs are initially submitted to the ASU site using the PBS Pro peer scheduler, there is a significant improvement in throughput when utilizing the grid for all workflow tasks. The throughput was somehow consistent with the performance observed with the Slurm multi-cluster scheduler, as illustrated in Fig. 10.

As shown in the two solid lines in Fig. 10, the grid throughput of the Slurm multi-cluster scheduler outperforms the throughput of the PBS Pro peer scheduler until 92 completed tasks. After that, the grid throughputs of both schedulers become close. Subsequently, the grid throughput of the PBS Pro peer scheduler improves after 122 completed tasks. This difference is attributed to the online decision-making of PBS peer scheduling and the offline decision-making of the Slurm multi-cluster. In contrast to the PBS Pro peer scheduling results shown in Fig. 9, the throughput of PBS Pro peer scheduler in Fig. 10 declines after 130 completed jobs. The decline in throughput happens because ASU receives more jobs (46 jobs), when jobs are initially submitted to ASU as shown in Fig. 10, than the number of jobs when initially submitted to SRTA as in Fig. 9 (32 jobs). The increase in the number of allocated jobs to ASU in Fig. 9 than Fig. 10 is attributed to the online decision making of PBS Pro peer scheduling. However, the throughput decline in the case of Slurm multi-cluster happens earlier than PBS Pro peer scheduler.

Another observation is that the Slurm multi-cluster mode does not account for the submission host. This might be perceived as unfair for local users if there is a need to have a higher priority for running jobs in the local cluster. In contrast, PBS Pro initially allocates jobs in the local cluster. If there are no available resources, the job enters the queue state, and finally, it is offloaded only if there are available resources in another cluster. Considering the HPC grid as a dynamic environment, it is better to make the offloading decision just before accessing the resources, as is done in PBS Pro.

Figures 11 and 12 illustrate the number of performed tasks within a specified time interval, as calculated by Eq. (6), using the grid resources compared to the number of completed tasks within the same time interval on SRTA and ASU sites, respectively. The number of executed jobs as a function of time using the grid is represented by a solid line, whereas the number of executed jobs as a function of time using a single site is plotted as dotted lines. The number of executed jobs as a function of time using the PBS pro and Slurm schedulers are represented by the red and blue lines, respectively.

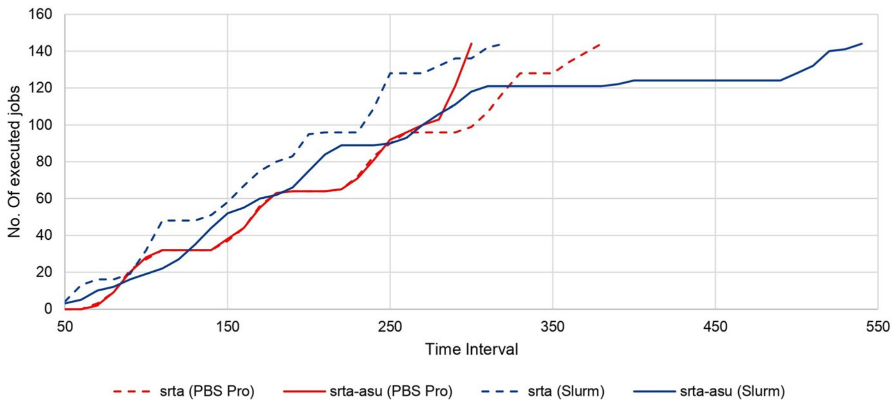

Fig. 11 Number of completed tasks as a function of time in srta

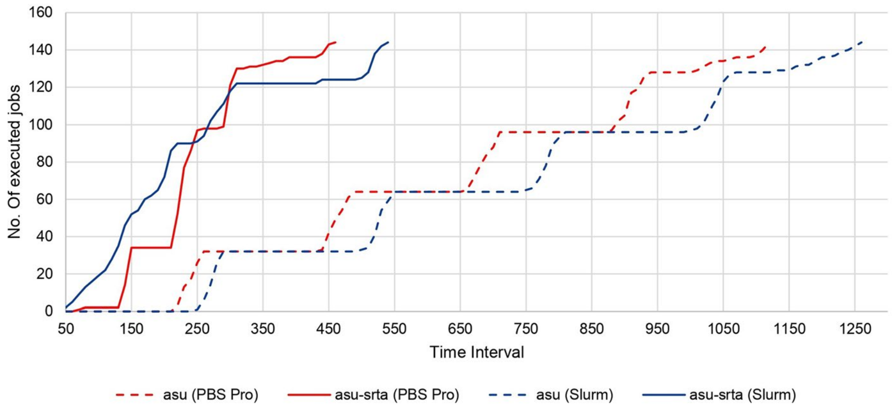

Fig. 12 Number of completed tasks as a function of time in asu

The effectiveness of the grid is apparent in all cases where jobs are initially submitted to ASU, whether using PBS Pro or Slurm. In addition, when employing the grid with the SRTA site, there is an increase in the number of executed tasks over time using Pbs Pro compared to Slurm. In contrast, the SRTA site does not gain any benefits when integrated into the grid using Slurm multi-cluster. The workflow jobs (144 tasks) took 534 s to complete in the SRTA-ASU grid scenario under Slurm multi-cluster scheduling, although they only required 312 s to finish under the PBS Pro peer scheduling.

Figure 13 illustrates the speedup of each site, evaluating the benefits gained by the site when integrated into the grid in terms of workflow tasks. Each site's speedup is calculated by Eq. (2).

Figure 14 illustrates the speedup of each site, evaluating the benefits gained by the site when integrated into the grid in terms of Quality of Service (QoS) for users. Each site's speedup is calculated by Eq. (3).

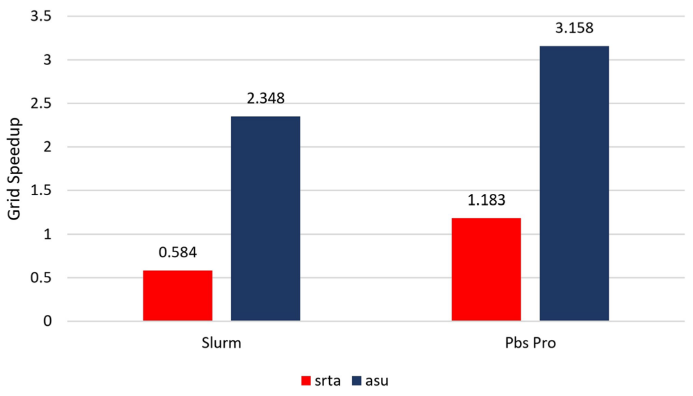

Fig. 13 Grid speedup for system/workflow

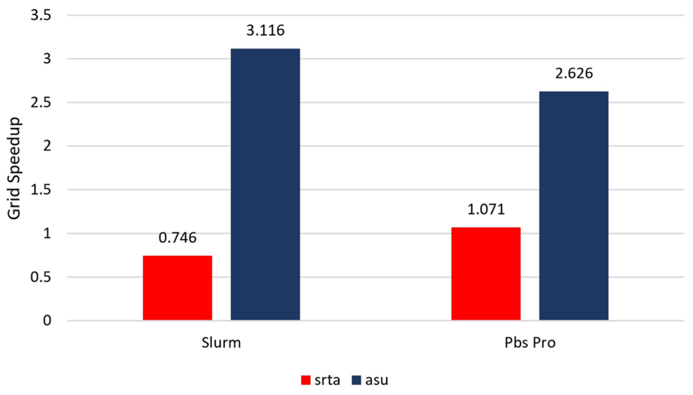

Fig. 14 Grid speedup for user

As depicted in the Figs. 13 and 14, the advantages of joining the grid are more pronounced for the ASU site compared to the SRTA site, whether utilizing Slurm or PBS Pro. The grid proves particularly beneficial for sites with constrained resources or aging hardware. In general, the PBS Pro scheduler achieves a faster speedup for both SRTA and ASU in terms of workflow tasks. However, the Slurm scheduler achieves a higher speedup for ASU in terms of individual tasks.

Based on the previous experimental results, it is not advisable to integrate a cluster with high-performance hardware with a cluster possessing aging or outdated hardware when using the Slurm scheduler. This is because the only site that would benefit from such integration is the one with outdated hardware. Slurm could be more efficient when integrating identical or converged clusters. However, if the scheduling algorithm in the Slurm multi-cluster mode can be changed to be comparable to the performance of PBS Pro peer scheduler (i.e., jobs are allocated to a cluster when required resources are free at this cluster, instead of early submission to a cluster based on the expected earliest start time), then Slurm multi-cluster performance can be similar to PBS Pro peer scheduler. We postpone the design of new scheduling algorithms to future work as it is beyond the scope of this paper.

6 Conclusion and future work

This study highlights the advantages of integrating clusters into a unified grid under specific workload conditions. Furthermore, it evaluates the effectiveness of well-known workload management systems in combining multiple clusters to form the required grid: (i) Slurm (Open Source) and (ii) PBS Pro (Licensed). The *mutli-cluster* and *peer scheduling* features of Slurm and PBS Pro, respectively, enable the integration of multiple clusters together into the grid environment without the usage of a specific middleware (e.g., Globus [27]). However, the underlying clusters must use the same workload manager to benefit from the Slurm multi-cluster or PBS Pro peer scheduling features. Based on this study, we are inclined to transition to the Slurm workload management system within our EN-HPCG grid, to reduce the costs associated with EN-HPCG sustainability. The experimental results have shown that Slurm is an efficient scheduler within an individual cluster or a grid based on identical hardware. In contrast, PBS Pro exhibits a more rapid speedup for both high-performance and outdated hardware when considering workflow tasks within the grid. Additionally, the PBS Pro scheduler is considerate of the online decision-making for the dynamic environment using a unified grid.

Therefore, in future work, there is an intention to investigate the Slurm Scheduler to enhance its performance. One avenue for improvement involves allowing each cluster to have its own database instead of using one common database for all clusters. Thus, if the network between the clusters fails, each cluster could still serve its users. Additionally, exploring the use of separate Munge keys for user authentication, as well as inter and intra-cluster(s) communication, is under consideration. Moreover, the design of AI-based and nature-inspired optimization scheduling algorithms for the multi-cluster system will be explored. As a different investigation, an extended comparison between Slurm and other open-source tools such as HTCondor and Globus will be conducted to reduce the costs associated with EN-HPCG sustainability.

In the exploration of prospective directions for future research, researchers may encounter various significant challenges. These challenges include effectively managing multi-schedulers, adapting to new architectures, integrating heterogeneous processors equipped with artificial intelligence chips and quantum processors, and seamlessly merging HPC grids with cloud servers. Furthermore, researchers could explore challenges associated with non-functional requirements, addressing issues such as energy consumption, scalability, and resilience [73].

Acknowledgements This study is part of the project named "Egyptian National High-Performance Computing Grid (ENHPCG)" which receives support and funding from the Egyptian Academy for Scientific Research and Technology (ASRT). Our sincere thanks go to the Ain Shams University Data Center team for their invaluable support in establishing the VPN connection between the two sites. We would also like to thank Yasser Abdullah and Ahmed Ibrahim for their cooperation in setting up and configuring the HPC cluster at the Faculty of Science, Ain Shams University.

Author Contributions ME contributed to conceptualization, methodology, implementation, validation, and manuscript writing. AIM done implementation, visualization, and manuscript writing. MSK and GMF helped in visualization and manuscript writing. HAH contributed to conceptualization, investigation, supervision, project administration.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB). Science and Technology Development Fund.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Sadeghibogar Z, Berti A, Pegoraro M, van der Aalst WM (2023) In: Proceedings of the 21st International Conference on Business Process Management (BPM 23), CEUR
2. Sadashiv N, Kumar SD (2011) In: 2011 6th International Conference on Computer Science & Education (ICCSE) (IEEE), pp 477–482. https://doi.org/10.1109/ICCSE.2011.6028683
3. Xu L, Wang Y, Lux T, Chang T, Bernard J, Li B, Hong Y, Cameron K, Watson L (2020) Modeling i/o performance variability in high-performance computing systems using mixture distributions. J Parallel Distrib Comput 139:87. https://doi.org/10.1016/j.jpdc.2020.01.005
4. Gupta A, Sarood O, Kale LV, Milojicic D (2013) In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (IEEE), pp 402–409. https://doi.org/10.1109/CCGrid.2013.65
5. Cocaña-Fernández A, Ranilla J, Sánchez L (2015) Energy-efficient allocation of computing node slots in HPC clusters through parameter learning and hybrid genetic fuzzy system modeling. J Supercomput 71:1163. https://doi.org/10.1007/s11227-014-1320-9
6. Altair Access Web User Guide. https://2021.help.altair.com/2021.1.1/accessweb/topics/jobs/pbs_peering_support_t.htm. Accessed 4 2023
7. PBS Professional 2021.1.2 Administrator's Guide. https://2021.help.altair.com/2021.1.2/PBS%20Professional/PBSAdminGuide2021.1.2.pdf. Accessed 4 2023
8. Schmitz M (2021) Moving to a multi-cluster HPC Slurm environment. Sandia National Lab. (SNL-NM), Albuquerque, NM
9. Slurm Multi-cluster. https://slurm.schedmd.com/multi_cluster.html. Accessed 3 2023
10. Slurm Federation. https://slurm.schedmd.com/federation.html. Accessed 3 2023
11. Zhang J, Lu X, Chakraborty S, Panda DK (2016) In: European Conference on Parallel Processing (Springer), pp 349–362. https://doi.org/10.1007/978-3-319-43659-3_26

12. Quick Start User Guide of Slurm Workload Manger. https://slurm.schedmd.com/quickstart.html. Accessed 7 2023
13. Wijngaart RFVd (2001) In: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, pp 315. https://doi.org/10.1023/A:1015669003494
14. https://www.nas.nasa.gov/software/npb.html. Accessed 5 2023
15. http://www.cs.ucy.ac.cy/crossgrid/wp2.3/benchmarks.html. Accessed 6 2023
16. Doddam M (2024) Altair HPCWorks: a comprehensive solution for high-performance computing management over open-source job schedulers. Technical report. https://altair.com/resource/altair-hpcworks--a-comprehensive-solution-for-high-performance-computing-management-over-open-source-job-schedulers
17. PBS Professional Manages Workload for NCI Raijin, Largest Supercomputer in Southern Hemisphere(2014). https://www.hpcwire.com/whitepaper/pbs-professional-manages-workload-nci-raijin-largest-supercomputer-southern-hemisphere/
18. Trader T (2021) On the eve of exascale, Argonne lab turns to Altair PBS pro after 20+Years of rolling their own workload manager. Technical reports. https://www.hpcwire.com/2021/11/10/on-the-eve-of-exascale-argonne-lab-turns-to-altair-pbs-pro/
19. Kyoto University's HPC Resources Power Japan's Research Community. https://altair.com/customer-story/kyoto-university-provides-high-performance-computing
20. PUNCH Torino Partners with Altair to Build a New HPC Infrastructure. https://altair.com/resource/punch-torino-partners-with-altair-to-build-a-new-hpc-infrastructure?lang=en
21. NSCC Delivers HPC in Singapore–Petascale Computing Resources Support Education and Research Nationwide. https://altair.com/resource/nscc-delivers-hpc-in-singapore-petascale-computing-resources-support-education-and-research-nationwide?lang=en
22. Cylc & Altair's PBS Professional Power Weather Modeling at Australia's Bureau of Meteorology (2019). https://altair.com/newsroom/articles/Cylc-Altair-s-PBS-Professional-x2122-Power-Weather-Modeling-at-Australia-s-Bureau-of-Meteorology
23. Hassan HA, Kashkoush MS, Maiyza AI, Fathy GM (2023) In: 2023 33rd International Conference on Computer Theory and Applications (ICCTA) (IEEE, Accepted)
24. Colvero TA, Dantas MAR (2013) Portais em grids computacionais–requisição de serviços e compartilhamento de recursos, SIRC-XII Simpósio de Informática
25. Filho NPR, e Silva FJEL, Garcia LG, Santos C (2016) Computational grids: challenges and new trends. Int J Eng Invent 5(9):25–41
26. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid: enabling scalable virtual organizations. Int J High Perform Comput Appl 15(3):200. https://doi.org/10.1177/109434200101500302
27. https://www.globus.org/. Accessed 8 2023
28. Unicore. https://www.unicore.eu/. Accessed 8 2023
29. Boinc. https://boinc.berkeley.edu/. Accessed 4 2023
30. HT-Condor. https://research.cs.wisc.edu/htcondor/. Accessed 4 2023
31. Park A, Fujimoto R (2008) In: 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (IEEE), pp 149–156. https://doi.org/10.1109/DS-RT.2008.34
32. Burchard LO, Linnert B, Heine F, Hovestadt M, Kao O, Keller A (2005) In: 19th IEEE International Parallel and Distributed Processing Symposium (IEEE), pp 8. https://doi.org/10.1109/IPDPS.2005.62
33. Tang J, Zhang M (2005) In: 2005 First International Conference on Semantics, Knowledge and Grid (IEEE), pp 57–57. https://doi.org/10.1109/SKG.2005.34
34. Derbal Y (2007) In: 2007 Canadian Conference on Electrical and Computer Engineering (IEEE), pp 514–517. https://doi.org/10.1109/CCECE.2007.134
35. Krishnan A (2005) Gridblast: a globus-based high-throughput implementation of blast in a grid computing framework. Concurr Comput Pract Exp 17(13):1607. https://doi.org/10.1002/cpe.906
36. Wang L, Jie W (2008) Integrated resource management framework for bio-grid computing. Neural Parallel Sci Comput 16(4):517
37. Kumar A, Qureshi SR (2008) In: 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008) (Citeseer)
38. Rosmanith H, Volkert J (2008) Interactive techniques in grid computing: a survey. Comput Inform 27(2):199
39. Danelutto M, Fragopoulou P, Getov V, Costa F, Silva L, Kelley I, Taylor I (2008) In: Making Grids Work: Proceedings of the CoreGRID Workshop on Programming Models Grid and P2P System

Architecture Grid Systems, Tools and Environments 12-13 June 2007, Heraklion, Crete, Greece (Springer), pp 377–391. https://doi.org/10.1007/978-0-387-78448-9_30
40. Iamnitchi A, Foster I (2004) Grid resource management: state of the art and future trends. Springer, Cham
41. Touzene A, Sultan AY, AlMuqbali H, Bouabdallah A, Challal Y (2011) Performance evaluation of load balancing in hierarchical architecture for grid computing service middleware. Int J Comput Sci Issues 8(2):213
42. Israr H (2013) Architecture level mapping of cloud computing with grid computing. Int J Eng Sci Emerg Technol 5(1):7
43. ElAraby M, Sakre M, Rashad M, Nomir O (2012) Crawler architecture using grid computing. Int J Comput Sci Inf Technol 4(3):113
44. Hoffmann L, Kaufmann M, Lehmann C, Spang R, Riese M, Moore D, Remedios J (2007) In: Proceeding Envisat Symposium 2007 (Citeseer)
45. Fukuda M, Ngo C, Mak E, Morisaki J (2007) In: 2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (IEEE), pp 145–148. https://doi.org/10.1109/PACRIM.2007.4313198
46. Song Y, Wu M, Yang L (2012) Study of smart grid marketing system architecture based on hadoop platform of cloud computing. J Commun Comput 9(7):741
47. Tannenbaum T, Wright D, Miller K, Livny M (2001) Beowulf cluster computing with windows. MIT Press, Cambridge, pp 307–350
48. Jette M, Yoo A, Grondona M (2003) In: Workshop on Job Scheduling Strategies for Parallel Processing, Seattle, WA, June, vol 24, pp 37–51
49. Shadura O (2020) In: PyHEP 2020 Workshop, Zenodo, vol 4136273. https://doi.org/10.5281/zenodo
50. Flechas MA, Attebury G, Bloom K, Bockelman B, Gray L, Holzman B, Lundstedt C, Shadura O, Smith N, Thiltges J (2022) Collaborative computing support for analysis facilities exploiting software as infrastructure techniques. arXiv preprint arXiv:2203.10161. https://doi.org/10.48550/arXiv.2203.10161
51. https://sc.edu/about/offices_and_divisions/division_of_information_technology/rc/hpc_clusters/. Accessed 2 2024
52. High Performance Computing at Sheffield. https://docs.hpc.shef.ac.uk/en/latest/index.html. Accessed 2 2024
53. Aldinucci M, Bagnasco S, Concas M, Lusso S, Rabellino S, Demarchi D, Vallero S(2019) In: EPJ Web of Conferences, vol 214 (EDP Sciences), pp 07030. https://doi.org/10.1051/epjconf/201921407030
54. High Performance Computing at Innsbruck. https://www.uibk.ac.at/th-physik/howto/hpc/. Accessed 2 2024
55. High Performance Computing at Plymouth. https://www.plymouth.ac.uk/about-us/university-structure/faculties/science-engineering/hpc/what-is-a-cluster. Accessed 2 2024
56. High Performance Computing at Brazil. https://www.lncc.br/sinapad/sinapad.php?pg=sinapad#. Accessed 2 2024
57. Reuther A, Byun C, Arcand W, Bestor D, Bergeron B, Hubbell M, Jones M, Michaleas P, Prout A, Rosa A et al (2018) Scalable system scheduling for HPC and big data. J Parallel Distrib Comput 111:76. https://doi.org/10.1016/j.jpdc.2017.06.009
58. Altair Access. https://web.altair.com/hpc-anywhere-with-altair-access. Accessed 4 2023
59. Altair Release Notes Guide. https://2022.help.altair.com/2022.1.1/PBS%20Professional/PBS_RN_2022.1.1.pdf. Accessed 5 2023
60. Renker G, Stringfellow N, Howard K, Sadaf R, Trofinoff S (2011) Deploying Slurm on xt, xe, and future cray systems, Cray User Group
61. Altair Control. https://altair.com/control. Accessed 4 2023
62. Jette MA, Wickberg T (2023) In: Workshop on Job Scheduling Strategies for Parallel Processing (Springer), pp 3–23. https://doi.org/10.1007/978-3-031-43943-8_1
63. Milroy DJ, Misale C, Herbein S, Ahn DH (2021) A dynamic, hierarchical resource model for converged computing. arXiv preprint arXiv:2109.03739. https://doi.org/10.48550/arXiv.2109.03739
64. Hassan HA, Kashkoush MS, Azab M, Sheta WM (2019) Impact of using multi-levels of parallelism on HPC applications performance hosted on azure cloud computing. Int J High Perform Comput Netw 13(3):251. https://doi.org/10.1504/IJHPCN.2019.098579

65. Hassan HA, Maiyza AI, Sheta WM (2017) Impact of process allocation strategies in high performance cloud computing on azure platform. Scalable Comput Pract Exp 18(2):161. https://doi.org/10.12694/scpe.v18i2.1288
66. Van der Wijngaart RF, Frumkin M (2002) Nas grid benchmarks version 1.0, NASA Ames Research Center TR NAS-02-005
67. Frumkin M, Van der Wijngaart RF (2002) Nas grid benchmarks: a tool for grid space exploration. Cluster Comput 5:247. https://doi.org/10.1023/A:1015669003494
68. https://slurm.schedmd.com/multi_cluster.html. Accessed 9 2023
69. Dwyer T, Fedorova A, Blagodurov S, Roth M, Gaud F, Pei J (2012) In: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (IEEE), pp 1–11. https://doi.org/10.1109/SC.2012.11
70. PBS-Portable Batch System. http://www.altair.com. Accessed 3 2023
71. Vazquez A, Vazquez A (2019) Freeipa entity management, Practical LPIC-3 300: Prepare for the Highest Level Professional Linux Certification, pp 607–661. https://doi.org/10.1007/978-1-4842-4473-9_25
72. Montero RS, Huedo E, Llorente IM (2006) Benchmarking of high throughput computing applications on grids. Parallel Comput 32(4):267. https://doi.org/10.1109/INCET49848.2020.9154055
73. Navaux POA, Lorenzon AF, da Silva Serpa M (2023) Challenges in high-performance computing. J Braz Comput Soc 29(1):51. https://doi.org/10.5753/jbcs.2023.2219

Authors and Affiliations

Mohammed Elshambakey[1] · Aya I. Maiyza[1] · Mona S. Kashkoush[1] · Ghada M. Fathy[1] · Hanan A. Hassan[1]

✉ Aya I. Maiyza
amaiyza@srtacity.sci.eg

Mohammed Elshambakey
ayaibrahim88@yahoo.com

Mona S. Kashkoush
mkashkoush@srtacity.sci.eg

Ghada M. Fathy
gfathy@srtacity.sci.eg

Hanan A. Hassan
hali@srtacity.sci.eg

[1] City of Scientific Research and Technological Applications (SRTA-City), Alexandria, Egypt