STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXI**, Number 2, 2016

BABEŞ-BOLYAI UNIVERSITY'S HIGH PERFORMANCE COMPUTING CENTER

DARIUS BUFNEA, VIRGINIA NICULESCU, GHEORGHE SILAGHI, AND ADRIAN STERCA

Abstract. This paper presents the High Performance Computing Center of the Babeş-Bolyai University from its architectural point of view but also it takes a deep look at the HPC Center's usability from a researcher's perspective. Its hybrid architecture - Cloud and HPC - represents a novelty of this kind of data centers, and it was the result of the emphasized necessities based on the university research development. Besides containing an extensive presentation of the High Performance Computing Center, the aim of this paper is to be a useful support for researchers that need to interact and use the HPC facilities and its resources.

1. Introduction

1.1. **The necessity of the HPC's existence within the Babeş-Bolyai University.** Babeş-Bolyai University is a high level academic educational public institution aiming to promote and sustain the development of important scientific and cultural components within the local, regional, national and international community.

From several years the university's researchers from different research fields such as: Mathematics, Computer Science, Physics, Chemistry, Biology, Geography, Economical Studies, or Environmental Studies, had to face the lack of existence of a powerful computational infrastructure. The need of such an infrastructure was obvious from more than one decade, but until now there were just small initiatives that came from lower organization units of the university (faculties or specific research groups) that lacked financial and decisional

Received by the editors: October 12, 2016.

2010 *Mathematics Subject Classification.* 68N01, 68N99, 68N25, 68M01.

1998 *CR Categories and Descriptors.* C.1.4 [**Computer systems organization**]: Parallel architectures – *Distributed architectures*; D.1.3 [**Software**]: Concurrent Programming – *Distributed programming – Parallel programming*; F.1.2 [**Theory of computation**]: Modes of Computation – *Parallelism and concurrency*; I.3.1 [**Computing Methodologies**]: Hardware Architecture – *Parallel processing.*

Key words and phrases. High Performance Computing, Cloud Computing, Computing Infrastructure, Computationally Intensive Applications.

power, this leading to the acquisition and deployment of small, low performance and not very well integrated solutions.

Considering this, in March 2013 a meeting was organized at the university level that brought together researchers and university decision-making authorities with different areas of expertise. The performed analysis emphasized the strict necessity of a powerful High Performance Computing Center within the Babeş-Bolyai University, which could facilitate the development of research leaded by our university.

1.2. **The MADECIP infrastructure project.** The Center for High Performance Computing was born within the MADECIP project - "Disaster Management Research Infrastructure Based on HPC" ("Dezvoltarea Infrastructurii de Cercetare pentru Managementul Dezastrelor Bazat pe Calcul de Înaltă Performanţă") [6]. This project was granted to Babeş-Bolyai University in 2014, its funding being provided by the Sectoral Operational Programme "Increase of Economic Competitiveness", Priority Axis 2, co-financed by the European Union through the European Regional Development Fund "Investments in Your Future" (POSCEE COD SMIS CSNR 48806/1862).

Eight faculties of the university took part in this project lead by the Faculty of Environmental Science and Engineering, aimed to develop a research infrastructure for disaster management based on high performance computing. During the last decade, the number of disasters, both natural and manmade, has increased significantly. One of the most important steps to reduce human and economic losses is the prevention and mitigation of disasters. That is why disaster management was always an important research area of the Faculty of Environmental Science and Engineering, and implicitly of the Babeş-Bolyai University. Preventing and preparing for potential natural or manmade disasters can reduce social vulnerability and lay out the road to a more resilient future and sustainable communities. The MADECIP project started on March 2014 and ended on December 2015. Currently the HPC Center is being jointly operated by the Faculty of Mathematics and Computer Science and the Faculty of Economics and Business Administration. The main computational infrastructure acquired through this project is a high performance computing system designed to be used for different jobs types, either computation intensive or data intensive from a variety of scientific domains.

1.3. **Domains identified to benefit from HPC's existence.** Since its funding has been accessed through an infrastructure project aimed to support the research in disaster management, first there were identified certain research areas that support the specified domain:

- Modeling and Simulation for: torrents, floods, dangerous substances overflowing or dispersion in fluid or poriferous environments, dam breakdown, etc.;
- Big Data Analytics for specific data needed in disaster managements;
- Web interrogation, big databases management;
- GIS maps, Satellite image processing;
- Disaster Decision Support Tool - DDST;
- Tools for communications, informing and alarming in disaster management domain;
- Simulation / Visualization of scenarios for different types of disaster.

As we have mentioned above, eight faculties from Babeş-Bolyai University were involved in the MADECIP project. Researchers from various disciplinary fields participated to numerous analysis and scenarios that have been done in order to specify and to emphasize the advantages that the HPC center could bring to many different researches conducted inside the university. Among these, we enumerate just a few of them:

- Molecular Physics;
- Applied Mathematics;
- High Performance Scientific Computation;
- Different areas of Optimization;
- Parallel programming: Models and Paradigms;
- Accelerators based parallel programming: GPU, Intel Phi;
- Frameworks and libraries to support high performance computation;
- Domain Specific Languages;
- Distributed programming;
- Image Processing: multimedia streaming, multimedia processing;
- Network traffic control;
- Big Data Analytics: data mining, statistics methods, fast search methods, knowledge discovery in large area networks, MapReduce computing, NoSQL databases.

2. Kotys HPC architecture

The Babeş-Bolyai University's HPC infrastructure (see fig. 1), which was named Kotys, after the Thracian deity of the wild nature, has a hybrid architecture. It consists in a mix of two components: the High Performance Computing itself and the Cloud Computing component. Although they are different from an architectural point of view, they are sharing a series of resources in cluster's physical architectures such as storage (over 72TB), the gigabit networking or the cooling system. In this paper we will focus more on

FIGURE 1. Kotys: Babeş-Bolyai University's HPC infrastructure

the scientific computing component of the HPC infrastructure since it is likely to be of the most interest to researchers from different academic fields.

2.1. **Cloud Computing component.** The Cloud Computing component is built around the IBM Flex System architecture [1]. It consists in ten IBM Flex System x240 virtualization servers, each of them being equipped with 128 GB of RAM, two Intel Xeon E5-2640 v2 processors and two 240 GB SATA SSDs. An additional management server in the same hardware configuration is also present. From the software point of view, the Cloud Computing component of Kotys is running IBM cloud with OpenStack manager 4.2 as the private cloud software, IBM Flex System Manager software stack for monitoring and management and VMware vSphere Enterprise 5.1 as virtualization software.

2.2. **High Performance Computing component.** The HPC component is built based on the IBM NeXtScale platform [2], following a classical cluster architecture (see fig. 2) where a head node serves as frontend for multiple compute nodes.

As a first particularity of Kotys, it has two head nodes: one designated for general use and basic task operations and one that serve as a backup and sometimes used for administrative purposes. The cluster has a total of 68

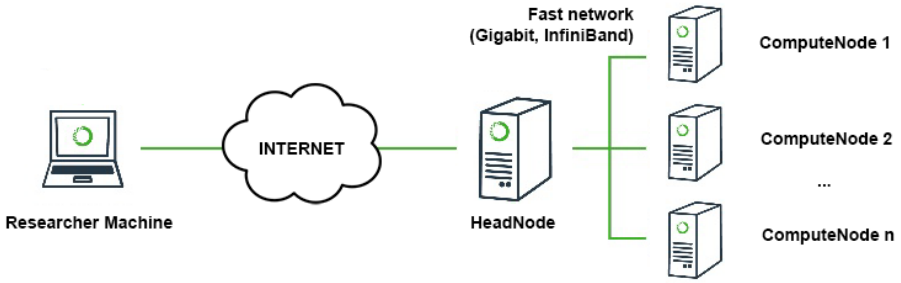

FIGURE 2. HPC cluster architecture

nodes, each of them consisting in a Nx360 M5 system. Both types of nodes are running Red Hat Enterprise Linux Server release 7 as operating system. The communications between the head nodes and compute nodes or between the compute nodes themselves are realized through a high speed InfiniBand network reaching 56Gbps. Kotys' performance based on the LINPACK benchmark [3] reaches 62 Tflops in Rpeak (theoretical) and 40 Tflops in Rmax (sustained). Each node is hosting 128 GB of RAM, two SATA 500 GB HDD and two 10-core Intel Xeon E5-2660 v3 processors running at 2.60GHz, giving a total number of 20 physical cores per node. In addition, 12 nodes are hosting two Nvidia Tesla K40 GPU [4] while 6 nodes are also equipped with an additional Intel Phi coprocessor SE10/7120 series [5].

For a better comprehension of Kotys' architecture, we summarize its component compute nodes in table 1:

* All cluster nodes in general can be used as part of the computational effort in a distributed and parallelized environment (e.g. MPI). However, some computational intensive applications or research activities do not require a distributed parallelized environment, rather than they require raw CPU power in a single host, single OS, multithread/parallel environment. For this kind of situations, Kotys has a number of reserved compute nodes, specified in the last column of table 1, that can be used by researchers in order to manually deploy and start their computational intensive applications in a single host environment. These compute nodes are not part of the node set used by the cluster's job scheduler for queuing and running distributed parallelized tasks in a cooperative manner. For this reason, when using these nodes, users may overlap, being their responsibility to check the nodes' usage and to use the

Compute node number	Hardware	Reserved for manual operations (non-LSF)*
compute001 ... compute050	2 x Intel Xeon E5-2660 v3 CPU, 10 cores per CPU	compute049, compute050
compute051 ... compute056	2 x Intel Xeon E5-2660 v3 CPU, 10 cores per CPU Intel Xeon Phi coprocessor SE10/7120 series (rev 20)	compute056
compute057 ... compute068	2 x Intel Xeon E5-2660 v3 CPU, 10 cores per CPU 2 x Nvidia Tesla K40 GPU	compute068

TABLE 1. Kotys' architecture

dedicated nodes in an elegant and fair fashion. The cluster administrators could take action in case the fairness rules are not respected. Later in this paper, section 4 exemplifies some use cases and best practices in using compute nodes' computational power in different scenarios.

3. Installed software

The following software is available on Kotys. It was either provided with the hardware itself (for example the management software), or was acquired separately through the same grant, based on the needs that were identified within the Babe-Bolyai research community.

- Integrated solution for the management of the HPC system: IBM Cluster Platform Manager. IBM Cluster Platform Manager is an integrated tool for defining, managing and monitoring clusters of up to 2500 nodes. It offers facilities that automate cluster provisioning and deployment, monitoring and reporting. In order to do this, IBM Platform Manager leverages xCAT technology and all operations are done through a web user interface. Provision is done automatically and in parallel for all the nodes, by specifying an image template in xCAT and rebooting the machines. Machines are added to the cluster or removed using the web interface and the job/LSF slots on each machine can be managed using this web interface. Also machines can execute remote commands using the Platform Manager Web GUI.
- Integrated management solution for the cloud system: OpenStack.

- Intel Parallel Studio, cluster edition: a development tools suite that helps squeezing every last flops of performance from the cluster's Intel hardware. It consists among others in a series of compilers for the C, C++ and Fortran languages, a high performance Python interpreter and an MPI implementation, all this optimized by Intel to take advantage of the latest state of the art Intel processors or Intel Phi coprocessors. To ensure best performance, we recommend building software using tools from this suite whenever possible.
- Rogue Wave TotalView: TotalView for High Performance Computing consists in a series of tools that allow easy debugging of parallel applications that run in a distributed environment. TotalView presents all the functions of a classical debugger such as running, stepping, halting line-by-line through code, offering in the same time support for multithreading when threads are running on hundreds or thousands of cores. As its name implies, it allows data visualization, program or memory states visualization and it performs all classical functions of a debugger in an intuitive graphical window based interface. TotalView Works with applications written in different programming languages such as C, C++ or Fortran offering support for different MPI implementations and for different specialized hardware: it supports CUDA debugging and Intel Xeon Phi coprocessors.
- Matlab: an engineering and scientific platform intended to be used by researchers for numerical computing. It has a wide applicability, being used in education but also by engineers in fields such as image processing and computer vision, signal processing and communications, economy, control design, biomathematics and much more. Matlab incorporates its own matrix-based programming language called MathWorks. Data visualization is also a very appreciated feature offered by Matlab, its built-in graphics being frequently used to gain data insights. It supports extensions, i.e. toolboxes, either official or community-contributed, beeing also able to interoperate with programs written in other programing languages such as C/C++, Java, C#, Fortran or Python.
- ANSYS CFD: a software package for integrated modelling and numerical simulation of fluid dynamics, heat transfer and turbulence modelling. It includes tools for 3D visualisation, analysis and powerful mathematical tools for problem solving.
- COMSOL Multiphysics: a software tool used for modelling and simulation of fluid dynamics and chemical reaction engineering. It is used for studying complex systems in which chemical reactions take place. The software models physical processes using a system of differential

equations and partial differential equations, it solves this system and also provides 3D graphical tools for visualizing all the mechanisms involved in the analyzed process.
- Gaussian: a software tool that computes the electronic structure of molecules and complex molecular systems and also for processing the obtained numerical data.

The software list that Kotys is capable of running is not limited to the above. Other scientific software could also be installed to satisfy researchers' needs, however please consider that any installed software requires a valid licence.

4. Accessing cluster's computational resources

As other High Performance Computing facilities, Kotys can be accessed in two ways: via a terminal (i.e. a command line interface) and through a browser using a web based interface. In both ways the user must connect to `kotys.cs.ubbcluj.ro`, the command line interface being accessible using putty (or any other ssh client) and ssh as the protocol on port 2222, while the web based interface is accessible by connecting to the same server with any browser, port 8080, protocol https (i.e. the URL for accessing the cluster's web interface is `https://kotys.cs.ubbcluj.ro:8443`). Please pay attention that for both interfaces the default port is different from the default protocol port (22 for ssh, 443 for https) - this is due to security reasons and administrative constraints.

We recommend using the terminal interface in scenarios like the following ones:

- for compiling, building and testing user's custom parallel software. Please DO NOT use the command line interface for running software directly on the head node (neither on the users' dedicated head node, nor the administrators' dedicated one).
- accessing via the dedicated head node one of the reserved compute nodes for manual operations (please see section 2.2 for the dedicated compute nodes reserved for manual operations).
- launching jobs through the job scheduler, verifying a job status and checking out the final results once a certain job finishes its execution.

The sftp extension of the ssh protocol and port 2222 can also be used if one wants to copy his files from a local computer to the cluster (or download locally files from the cluster). These operations can be easily performed from a Windows operating system using the WinSCP client.

The web interface can also be used for job related operations, such as submitting and checking a job status or verifying its output when finished.

In the following sections of this paper, we will present some usage scenarios of cluster's computational resources suited for different type of applications. Researchers may choose the computational approach that best fits their needs.

From the beginning, we assume that the reader is familiar with the most commonly used UNIX/Linux commands and he or she has basic knowledge and understanding of the command line interaction via shell with a UNIX like operating system. If not, we kindly ask the reader to take a look at the following UNIX commands and their most important parameters: `cat`, `ls`, `cd`, `pwd`, `rm`, `mkdir`, `rmdir`, `mc`, `vi`, `finger`, `ps`, `kill`, `top`, `ssh`, `screen`. Further considerations about these commands are beyond the scope of this paper.

As we have previously mentioned, Kotys has a classical cluster architecture with a head node serving as frontend for multiple compute nodes. When accessing the head node console via a ssh client, one might notice the following:

- from the head node (designated by the `[user@headnode ]$` prompt), the user can connect using ssh without entering an username or a password to any other node within the cluster (this is done using private-public key authentication). The nodes are named from compute001 to compute068, the user may use the `cat /etc/hosts` command at any time to retrieve their names. In fact, many cluster oriented frameworks (including MPI) rely on this ability to do ssh without prompting for the username and password from one node to another in order to execute commands without user intervention on the other nodes.
- any file or directory presented on the head node file system in the user's home directory also exists in the same form on any other node. This is because the cluster is using a distributed file system where the compute nodes are mounting a head node exported file system via NFS.

4.1. **Single host, very intensive, non-distributed applications.** Some researchers may wish to access cluster's computational power but their algorithms are implemented following a classical programming paradigm, in a so called monolithic application, using at the very most multithreading as a way of parallelization, but no distributed computations. This type of researches just wants a computer capable of overnight running their application faster than their laptop or personal computer is, a multiprocessor/multicore system with oversized memory, as cluster's nodes are, being ideal to perform their computations.

Normally, this kind of applications could easily be run through a job scheduler such as LSF [14] [15] that provides load sharing across the cluster (more details about this are exposed in the following section). However, we imagined this type of usage scenario and reserved some nodes for manual, non-LSF

operations based on users' requests. Some researchers might prefer a more classical approach to run his or her experiments, not bothering with maybe new or complicated terms such as load sharing facility or job scheduler. Besides simplicity, direct run, no queuing or real time output are other advantages in this scenario. Also, considering that other users are not competing for resources, this guarantees 100% resource allocation to the user in the lack of other processes that might load the host.

A major disadvantage in this usage approach is the fact that the user must negotiate and manually plan resource sharing directly with other users, and in the same time there might be periods of time when the reserved nodes stay underloaded, while the rest of the cluster is intensively busy.

In this type of scenario, a researcher's typical to follow steps are:

(1) Copy his/her files in the personal home directory using WinSCP or any other secure file transfer protocol client.
(2) Access using Putty or any other secure shell client the cluster (i.e. the head node) using its name `kotys.cs.ubbcluj.ro`, protocol ssh, port 2222.
(3) From the head node, the user can connect via ssh to the dedicated compute nodes (see the last column of table 1). We renew here again the request NOT to run resource-intensive computations directly on the head node.
(4) Check the compute node status, usage and load (using commands such as `finger`, `ps auxf`, `top`).
(5) Use the programming language of choice in order to locally (on the specific compute node) build and run researchers' own code. For the moment C/C++, Java and python are available, but if needed, other programming languages can be installed if available in Red Hat 7 standard repository.

In order to obtain best performance, we advise the readers to check if their research software supports different accelerated hardware that Kotys incorporates such as NVidia CUDA or Intel Xeon Phi and consequently build (and run) their software with support for the respective hardware.

Please note that installing/upgrading different operating system components such as drivers (for example Nvidia CUDA drivers) or system wide libraries is not possible either in order not to break consistency with the other nodes or due to official technical support reasons. If necessary, users can locally build their own libraries (or libraries versions); please read any Linux/UNIX documentation related to configure and make for further considerations into this topic.

Future plans imply moving this type of applications to the cloud component of Kotys (cloud nodes having the same hardware characteristics as those of the HPC component), with the exception of the applications that need specialized hardware (NVidia CUDA or Intel Xeon Phi).

4.2. **MPI-based applications.** Researchers that want to efficiently reduce the computational time of their applications must start thinking parallel, i.e. they must adapt their algorithms to follow a parallel paradigm in order to be easily deployed and run on a parallel cluster such as Kotys. Once an algorithm is parallelized, a very much used implementation approach is that based on MPI (Message Passing Interface) API [7]. Message Passing Interface is a standardized programming interface for developing parallel programs, designed by an open group of researches and parallel computers providers alike. Being just an interface, it has a series of different implementations, either free or commercial, such as Open MPI [8], MVAPICH [9] or Intel MPI [10] - we just enumerate the MPI implementation available on Kotys. An exhaustive presentation of MPI is out of scope of this paper, we highly recommend to the reader references [12] and [13] as two excellent introductions to MPI. Also, [11] contains some basic MPI examples that the reader might find useful as an introduction to this topic.

Scientific parallel algorithms meant to be executed in a cluster environment tend to run for a long time. For this reason, we recommend in their implementation phase, to test and debug them using small data sets, and if possible, less iteration count. In this way, their execution time may be considerably reduced, which allows development, implementation and debugging cycles of a parallel algorithm to be performed swiftly even on a home or office personal computer. Although MPI was developed to be used mainly in a cluster oriented environment, it can be successfully used on a lower scale system such as on a personal computer with a single dual or quad core physical CPU. Even though MPI wants to be a portable programming interface, we recommend researchers to approach it in a Linux like environment. This is because most major Linux distributions offer an MPI implementation via their official software repository (which leads to an easy installation process) and, in the same time, the researcher is getting used to an environment similar to the one presented on a real cluster.

Considering that on Kotys there are installed three different MPI implementations, a basic requisite is that the user should be able to select and use the MPI implementation of choice in an easy and elegant manner. Hence, the `mpi-selector-menu` command is provided in order to achieve this. After changing the default MPI implementation, in order for the new one to take effect, the user should relogin or start a new shell (by running the `bash`

Open MPI	MVAPICH	Intel MPI	Effect
--version	--version	--version	Displays mpirun version
-hostfile | --hostfile | -machinefile | --machinefile	-f	-f | -hostfile	Specifies a file containing the nodes to distributively run the MPI task
-H | -host | --host	-hosts	-hosts	Specifies nodes in a comma separated list
-c | -np | --np | -n | --n	-n | -np	-n | -np	Specifies the number of processes
-N | -npernode | --npernode	-ppn	-ppn | -perhost	Specifies the number of processes to be launched per node
-output-filename | --output-filename	-outfile-pattern -errfile-pattern	-outfile-pattern -errfile-pattern	Redirects the standard output and standard error

TABLE 2. `mpirun` command variants

command). Although, the user may use any of the provided implementations, we recommend the use of Intel MPI, for its superior performance and better support offered for Intel CPUs.

Two commands are important when working with MPI. The first one is the `mpicc` command which is a wrapper around a C compiler that allows the compilation of MPI programs. The second one is the `mpirun` command which allows the start of an MPI program, offering in the same time the possibility to configure certain attributes of the program's running instance, the most important being the number of processes to fork and the nodes where the forked processes should run.

Although all `mpirun` implementations offer the same functionality, their accepted parameters may be slightly different from one implementation to another. Table 2 contains some of the most frequently used parameters given to the `mpirun` command variants.

Running directly an MPI application through a command such `mpirun` is a feasible scenario when using a small private dedicated cluster. However, when running an MPI program on an enterprise level cluster, where multiple users

might run multiple parallel MPI tasks that compete each other for resources, a more resource sharing capable method should be adopted. Such a method should be aware of nodes' work load and resource usage (for example which nodes and how many cores are involved in an MPI intensive computation). The recommended method to run distributed parallel tasks on an enterprise level cluster is through the help of a job scheduler. A job scheduler is responsible for resource management and performs job scheduling, choosing the best available nodes to split and distribute tasks in order to provide load balancing and maximize hardware resource utilization. Job scheduling is performed by putting jobs in one or more queues that might have different priorities, from where jobs are executed in general in a FIFO order when resources are made available.

Kotys' job scheduler is called Platform Load Sharing Facility [14], [15], or simply LSF. LSF is IBM's proprietary job scheduler for their provided HPC systems. Even though it has a complex architecture (consisting in a series and distributed daemons and services needed to measure node load, provide redundancy or job scheduling and launching), jobs controlling and monitoring from the user side is performed in a very simple way. This is done either through an easy and intuitive web interface or through a series of OS integrated commands: `bsub`, `bjobs`, `bhist`, `bhosts`, `bpeek`, `bkill`, `bstop`, `bresume` - more details about each of them being available in [15]. Also, in [11] we give some detailed examples of running MPI tasks with and without the LSF scheduler.

When running an MPI task through the job scheduler (using the `bsub` command), the node list where the task should run is optional, although a host file might be specified as a parameter to the `bsub` command too. However, this is not recommended since the enforced nodes might not be available and in general, LSF can better select the most suitable nodes for the task to run. A host file is recommended only in the situation that a task must run on a specific hardware configuration (for example on nodes that contain a NVidia Cuda GPU), but even in this situation the task might be submitted without a host file to a queue configured to use only a subset of the cluster nodes. The same stands also when it comes to specifying a number of processes to be launched per node: in the absence of other constrains, the job scheduler might be able to more properly split the task over nodes, depending of the number of available cores on each node.

We enforce that on Kotys all MPI task to be run through the LSF scheduler, as this approach has undeniable advantages over a direct run without the LSF of the `mpirun` command.

4.3. **Non-MPI parallelized applications in a distributed environment.** A series of distributed computationally intensive software might fall in neither one of the above categories - for example, parallel and distributed research software built without MPI support or without resource sharing in mind, but designated to run on multiple nodes. Such applications are run especially on small private and dedicated clusters, in which resource sharing is not a constraint - i.e. the cluster is running almost exclusively a specific application that fully loads and consumes all cluster CPU power.

If such an application has to be run on Kotys, or if the researcher is unsure about the application's behavior in a parallel and distributed environment, we recommend the following to be taken into consideration:

- Do not directly run the application neither on the head node or the compute nodes controlled by the LSF, hence this can lead to an unfairness situation, disadvantaging LSF scheduled jobs;
- Run the application using the LSF. However, even if the application is scheduled by the LSF to run on a specific underloaded node (or nodes), if the application accept a hostsfile of some sort, the application might fork on the other nodes (specified in the hostsfile) starting non-LSF aware tasks. Once more, this can lead to an unfairness situation.
- If the application accepts a hostsfile of some sort, it might be run without the LSF on the reserved nodes for manual operations (see table 1).

4.4. **General recommendation for different application types.** In any of the situations described in section 4.2 and 4.3, if scientific well-spread and mature research software is supposed to be used, we recommend an in deep study of that software's documentation or support forums regarding its best practice usage scenarios in a parallel environment through a job scheduler.

If the researcher wishes to develop his or her own scientific software from scratch, we recommend an MPI based approach, as described in section 4.2.

As a final consideration, each user is asked to periodically check the amount of free space available on cluster's file system using the `df` command and also, the personal amount of data occupied within his or her personal directory using the `du -h` command. Any temporary files or folder input/output data that are not needed anymore should be removed in order to preserve cluster's resources.

5. Conclusions

We have presented in this paper the premises on which the HPC Center of the Babeş-Bolyai University was born, taking in the same time a deep look at the HPC system's architecture. We have also described a series of usage

scenarios, with the hope that the information presented in this paper will be useful for researchers from different academic fields that wish to access Kotys' resources.

For the future, we wish to strengthen the multidisciplinary research collaboration inside the Babeş-Bolyai University. Moreover, this paper is intended to be an invitation for collaboration with researchers outside the university as well.

Also, we invite all the potential users of the HPC system to study this paper and kindly ask them to referrer it in their research.

The official web page of the High Performance Computing Center [11] contains an extensive collection of resources, links and materials HPC related, that we strongly recommend for further reading and that might be helpful for researchers in their work.

References

[1] David Watts, Ilya Krutov, Flex System Enterprise Chassis Product Guide, Lenovo Press, first published April 2012, last updated August 2016, https://lenovopress.com/tips0863-flex-system-enterprise-chassis

[2] IBM NeXtScale System, Next-generation dense platform provides superior building-block approach for hyperscale computing, IBM Corporation 2013, https://www.ibm.com/midmarket/att/pdf/Nextscale_Datasheet.pdf

[3] Jack J. Dongarra, Piotr Luszczek, Antoine Petitet, The LINPACK Benchmark: past, present and future in Concurrency and Computation: Practice and Experience, 2003, 15:803820 (DOI: 10.1002/cpe.728), John Wiley & Sons, Ltd.

[4] Tesla K40 GPU Accelerator Overview, http://www.nvidia.com/content/PDF/kepler/nvidia-tesla-k40.pdf

[5] Intel Xeon Phi Coprocessor Architecture Overview, 2013, https://software.intel.com/sites/default/files/Intel%C2%AE_Xeon_Phi%E2%84%A2_Coprocessor_Architecture_Overview.pdf

[6] MADECIP project - "Disaster Management Research Infrastructure Based on HPC", http://madecip.granturi.ubbcluj.ro/

[7] MPI official website, http://mpi-forum.org/

[8] Open MPI: Open Source High Performance Computing - A High Performance Message Passing Library, https://www.open-mpi.org/

[9] MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE, http://mvapich.cse.ohio-state.edu/

[10] Intel MPI Library, Deliver Flexible, Efficient, and Scalable Cluster Messaging, https://software.intel.com/en-us/intel-mpi-library

[11] The official web page of the High Performance Computing Center, http://hpc.cs.ubbcluj.ro/

[12] William Gropp, Ewing Lusk, Anthony Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface (Scientific and Engineering Computation), 3rd edition, MIT Press, 2014, ISBN-10: 0262527391

[13] Wesley Kendall, Beginning MPI (An Introduction in C), Amazon Digital Services LLC, 2013, ASIN: B00HM7O0M8

[14] IBM Platform LSF Foundations, Version 8.3, May 2012, http://publibfp.dhe.ibm.com/epubs/pdf/c2253480.pdf
[15] IBM Platform LSF V9.1.3 documentation, September 2014, http://www.ibm.com/support/knowledgecenter/SSETD4_9.1.3/lsf_welcome.html

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: bufny@cs.ubbcluj.ro
E-mail address: vniculescu@cs.ubbcluj.ro
E-mail address: forest@cs.ubbcluj.ro

Babeş-Bolyai University, Faculty of Economics and Business Administration, 58-60 Teodor MIhali St., 400591 Cluj-Napoca, Romania
E-mail address: gheorghe.silaghi@econ.ubbcluj.ro