International Journal of Computer Theory and Engineering, Vol.2, No.6, December, 2010
1793-8201

OpenPC : A Toolkit for Public Cluster with Full Ownership

Z. Akbar, I. Firmansyah, B. Hermanto and L.T. Handoko

***Abstract*—The openPC is a set of open source tools that realizes a parallel machine and distributed computing environment divisible into several independent blocks of nodes, and each of them is remotely but fully in any means accessible for users with a full ownership policy. The openPC components address fundamental issues relating to security, resource access, resource allocation, compatibilities with heterogeneous middlewares, user-friendly and integrated web-based interfaces, hardware control and monitoring systems. These components have been deployed successfully to the LIPI Public Cluster which is open for public use. In this paper, the unique characteristics of openPC due to its rare requirements are introduced, its components and a brief performance analysis are discussed.**

***Index Terms*—distributed system, distributed applications, Internet applications, open source**

I. Introduction

The popularization of high performance computing (HPC) leads to great demands on highly skilled human resources in the field. Although, in recent years there are many emulators providing a practical platform for parallel programming on a single PC, comprehensive skills on parallel programming can in most cases be obtained by the users having comprehensive distributed environments with full ownership and access. Moreover, the parallel programming highly depend on the problems under consideration and the architectures of distributed computing environment where the codes are going to be executed on. Though building a small scale HPC is getting easier and cheaper in recent days, yet it requires unaffordable cost and expertise for most people or small research groups. In particular, a 'comprehensive and advanced' HPC environment actually needs more non-trivial efforts to realize. The comprehensive and advanced HPC environment here means the advanced environment deployed in the system (middleware, resource allocation management, etc), and is not just the great number of nodes.

Therefore enabling as wide as possible access to an advanced HPC machine is getting an important issue in, especially educating young generation with comprehensive knowledge in the field. As a key solution to overcome this issue one of us [1], has developed an alternative architecture for parallel environment available for public use called as LIPI (Lembaga Ilmu Pengetahuan Indonesia - Indonesian Institute of Sciences) Public Cluster (LPC). The initial prototype has succeeded in enabling full access for remote users to the whole aspects of HPC over an integrated web interface [2]. In LPC the web interface plays an important role to :

- Provide a user friendly interface concerning that most users are at the beginner level. Because LPC is originally intended for educational purposes, i.e. providing a training field to learn all aspects of distributed programming and computing environment.
- Keep the whole system secured from any potential attacks since the LPC is open for anonymous users. Everyone can start using the system once their online registration was approved. Though this simple procedure is definitely reducing the obstacles for users, instead the administrator has to pay special attention to secure the system. Hence, the main issue in this matter is how to replace the usual `ssh` based commands with the web based ones, while keeping the same level of degree of freedom.

During its first phase of implementation, the LPC enjoyed rousing welcome from the communities. We should remark that although the LPC was intended for educational purposes, the facility has also been used by some small research groups to perform more serious scientific tasks.

In the second phase of development, we have improved all existing components of LPC and also added some new components to meet recent requirements to be a real HPC machine. Those involve the basic architectures [3], the algorithm of multi block approach [4], the resource allocation [5] and the real-time control and monitoring systems [6], [7]. Finally we have recently bundled those components into an integrated toolkit, namely the Open Public Cluster (openPC) and release it for public use under GNU Public License [8].

In the present paper, we are introducing the current status of openPC, its concepts and detailed architectures. We also present the current implementation at the LPC which continues providing its services in Indonesia and surrounding regions. The paper is organized as follows. First the basic concept of LPC is introduced. It is followed by the global architecture and the detail of queue and resources management schemes. Before giving the conclusion the implementation of current system at LPC is explained.

II. CONCEPTS

As already mentioned briefly in the preceding section, the concept of public cluster is motivated mainly by a need to provide a public infrastructure for HPC. One of us has built a

All authors belong to the Group for Theoretical and Computational Physics, Research Center for Physics, Indonesian Institute of Sciences, Kompleks Puspiptek Serpong, Tangerang 15310, Indonesia, contact email : gftk@mail.lipi.go.id.

Z. Akbar is also with the Group for Bioinformatics and Information Mining, Department of Computer and Information Science, University of Konstanz, Box D188, D-78457 Konstanz, Germany.

I. Firmansyah is also with the Center for Environmental Remote Sensing, Chiba University, 1-33 Yayoi-cho, Inage-ku, Chiba, Japan.

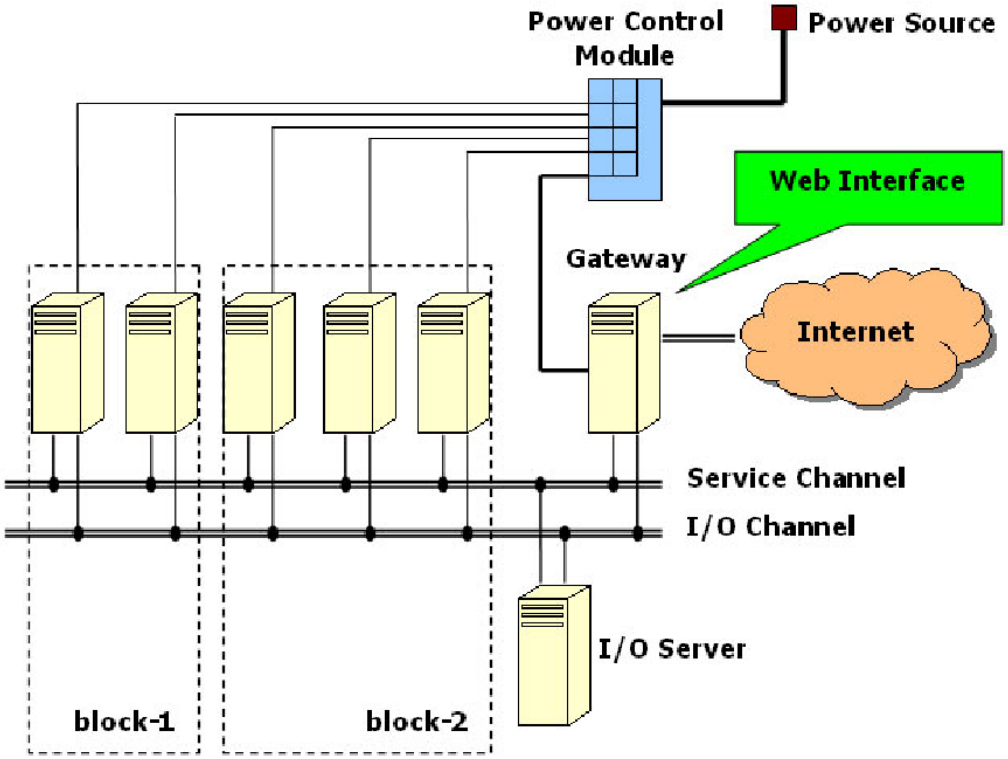


Fig. 1. The concept of public cluster with full ownership on each block of nodes using multi-block approach.

successful prototype of LPC with the following purposes [1] :

- Providing a practical but real training field for teachers and students to explore any aspects of HPC, that is not only developing the parallel programming itself.
- Improving the utilization of the existing HPC facilities. Since in some HPC centers the infrastructures tend to idle away due to limited users.

Clearly, these purposes are more relevant for some developing countries with limited HPC resources. However, the issues are, especially the second one is, actually relevant to any other developed regions as well. The limitations are in general caused by tight regulations due to mostly security concerns.

The first purpose above requires “full ownership“ by users on the cluster being used. It cannot be replaced by conventional clusters where the users just put their parallel jobs on the queues and let the machines do the rest. The full ownership here is defined as the user is provided with enough privileges to :

- Virtually, but fully own a block of nodes within a certain period without any job from the others in the same block.
- Choose any parallel environment, i.e. the middleware for job distributions, etc.
- Allocate the resources according to the user preferences within an assigned block.

Just to mention, in the near future the block owned by a user is able to be connected to any global grid relevant to the user’s project.

These concepts enable users to explore and perform any types of computational works on an HPC machines. This is the main reason we have argued that the system is suitable for educational purposes. Of course, the advanced users could also benefit from the system, since they can use it to perform any distributed computations with much more degrees of freedom. In contrast to any existing HPC machines, the public cluster is becoming a kind of really general purpose HPC machine. This would also improve the utilization and be good justification to obtain public funding to initiate and maintain an HPC center as well.

In order to realize the concepts, we have proposed a simple solution using the so-called multi-block approach [4]. As shown in Fig. 1, the public cluster, or just a set of parallel machine, is divided into several inhomogeneous blocks. Each of them consists of various number of nodes and forms a subset of resources. Utilizing the microcontroller-based control and monitoring systems, each node is controlled and monitored independently over web through the gateway. Throughout the paper we call the subset of nodes as a block.

We should remark here that according to the above mentioned concepts, openPC has completely different natures with another existing web-based clustering toolkits to enable ”remote access over web“ like OSCAR [9], or some grid portals for interfacing large scale distributed systems over web [10], [11], [12], [13], [14]. On the other hand, it does also not belong to the same category as Globus Toolkit (GT) which is intended to ”interface“ large scale grid computing [15]. We also recognize some partial web-interfaces for clusters

developed by several groups as done in [16], [17], [18]. Though the architecture of openPC is like a miniature of grid, since the blocks are connected each other as a single conventional parallel machine the physical topology is quite different. Moreover, openPC has a capability to control and monitor each single node that is irrelevant in tools like GT.

III. ARCHITECTURE

The openPC is a toolkit set dedicated to realize the concepts of public cluster. It consists of several components, ranging from hardware control to web interface. The toolkit has been developed with main purposes :

- As a single interface between users and the blocks assigned for each user with full ownership policy. This includes integrating all components and making them accessible to remote users over a user-friendly web interface.
- Providing as high as possible degree of freedom to users, while on the other hand keeping the whole system totally secured. The main concern is especially to keep the user and their jobs stay on the assigned blocks.

Under the full-ownership policy, a user is provided exclusively an HPC system without worrying about potential interferences with the others. All complexities are concealed behind the screen, although the users are still able to configure their own block as if an independent HPC machine like choosing the middleware for distributed environment and so on.

When a user submits a particular command through the web interface, the openPC performs several internal processes :

1) Recognizing whether the command is related to : the distributed computing, i.e. Portable Batch System (PBS), the operating system (OS), or the hardwares, i.e. `port com`. This selection procedure is crucial to determine how to treat and the forwarded target of each command.
2) If the command is a `port com` type, the openPC calls the library to execute further an appropriate command to the hardware modules (control and monitoring systems). In contrary, the PBS or OS type commands are passed to the I/O / master node through a newly developed layer called public cluster module as secure shell (`ssh`) wrapped commands.
3) On the other hand, each node within the block has two daemons running simultaneously, that is MoM (Machine Oriented Mini-server) and MPD. The PBS MoM is responsible for job management, while the `mpd` is the computation daemon to enable message passing among the nodes.

With these procedures and the hardware configuration depicted in Fig. 1, we can move forward to discuss the details.

A. Mapping a queue to a subset of resources

As briefly mentioned before, one of the main concerns is how to keep a user's jobs are sent to the designated nodes within the user's block. This can be achieved by employing a mapping procedure. In openPC, we have borrowed the Terascale Open-source Resource and QUEue manager (TORQUE) and Maui Scheduler to realize that requirement [19], [20]. TORQUE is actually an extension of PBS or definitely OpenPBS [21]. On the other hand, Maui is a "policy engine" to control "when, where, how" the available resources like processor, memory and disk space are allocated to particular jobs in the queue. Maui does not only provide an automated mechanism to optimize the available resources, but also to monitor the performances and analyze the problems occurred during the running periods.

Utilizing the TORQUE and Maui, the mapping procedure in openPC is done as follows :

1) Creating the host access list :
   The host access list is configured and activated in each queue such that only those nodes can recognize the queue. This will ensure that the queue is sending the jobs to the assigned nodes of a particular block owned by the user.
2) Creating the user access list :
   This access list limits the users who are allowed to submit jobs to particular queues. This method is used to avoid, either intentionally or unintentionally, submission to another queues.

Once a block has been created in a public cluster, it will remain with its initial nodes and user within the approved period of usage. This is suitable for the main purpose under consideration, that is providing a training field for learning HPC environment. The exclusive blocks allow the users to perform experimental jobs with high degree of freedom, or to explore any aspects of HPC, without any interferences among them. Moreover, in the current openPC each block has its own master node to improve the user degree of freedom, while reducing the load on I/O server. This allows the users to choose the desired parallel environment etc for their blocks. Anyway, it has been shown that up to a certain extent the multi blocks approach with single master node is still reliable [22].

Through these algorithms, we can define a queue which represents a closed subset of nodes owned by a single user without being interfered by another ones. This simple mechanism is the underlying idea to provide those blocks to public with full ownership. Therefore a block is a subset of nodes defined in a single queue.

B. Resources management scheme

A resource manager always attempts to choose the best combination of nodes to perform the submitted jobs automatically. In the conventional cluster, the users should determine how to distribute the jobs over the nodes manually by fixing certain parameters, or choosing among available queues. On the other hand, the administrator set up some queues for certain purposes which can be chosen by the users when submitting the jobs.

In contrary with this in the LPC, however we can not deploy the same approaches since the users (and also their jobs) are anonymous, while they basically own the blocks fully. Since in the LPC a queue is created once a block has been activated, the queue is exclusively limited to a particular user and consists of certain nodes assigned to the user. Hence the users have full

privileges to allocate the jobs and manage the nodes within their blocks. Using TORQUE and Maui. resource management within a block is performed in the same way as conventional clusters.

IV. IMPLEMENTATION

Now we are ready to discuss the implementation on the LPC. As shown in Fig. 2, LPC has 3 components :

1) Gateway node :
   This is the front-end interfacing the users with the public cluster over an integrated web-interface.
2) I/O / Master node :
   This node is responsible for all computational jobs in LPC. It executes the `pbs_server` and maui daemon to manage the whole cluster resources.
3) Nodes :
   The `pbs_mom` is executed in these nodes. The PBS MoM is responsible for communication with `pbs_server`, that is providing places for jobs submitted by `pbs_server`, responding the resource monitoring requests, reporting the finished jobs and so on.

On the other hand, as mentioned before the web interface at LPC handles three types of commands, the so called PBS, OS and com-port related commands. The first one is responsible for the computational works being performed. The second one is related to the file managements, while the last one is responsible for hardware control and monitoring systems.

We have developed the so called LPC module as the public cluster module. Both PBS and OS commands are sent to the computing nodes via this LPC module using `ssh` protocol to ensure secure data transfer. We have deployed the RSA or DSA key based passwordless `ssh`, rather than using the standard Apache modules for the sake of security. In contrary, the com-port commands directly trigger the external shell scripts to send certain signals to the microcontrollers to control power modules etc. We have found that this mechanism is much more reliable for our purposes than, for instance, real-time platform like Java which could exhaust more resources.

Now let us discuss in more detail the LPC module, the implementation of the block approach to build independent subsets of nodes, and the whole procedures deployed in LPC.

A. LPC module : public cluster module

One of the main differences with another cluster / grid architectures is, as can be seen in Fig. 2, there is an additional layer called as LPC module in the I/O / master node. The main purposes of deploying this module are :

- Load balancing :
  Taking over all distributed computing from the gateway node. For instance, although the command to access PBS is submitted by gateway node, the query to the PBS is done by the LPC module.
- Security :
  Isolating the computing nodes completely from the gateway node. Since the end-users access the system over a web interface installed in the gateway node, this would keep them away from the computing nodes itself.
- Easy maintenance :
  Since the web interface and the distributed computing tools are separated at different nodes, maintenance works at one of them would not influence the others.

These purposes can actually be accomplished using the Remote Procedure Call (RPC) model. In LPC it is realized using `ssh` that has been found more secure than the others. Without this module, the architecture is similar to the existing web interface as HPC2N which is also interfacing the PBS over web [12]. In contrary, using the LPC module, all (PBS and OS) commands submitted from the gateway node are passed through this module. Thereafter, all activities related to the file system and job management are also handled by the module. Hence, the LPC module contains all pre-defined commands to recognize the incoming commands and passed the authorized ones to the `pbs_server`, maui daemon and file system. This additional layer improves the security since the unauthorized commands are discarded immediately to prevent inappropriate accesses by users.

B. Defining a block

Once a block of nodes has been activated, the host and user access lists associated to the block are automatically created. The queue is exclusively provided to a single user assigned to the block. This would ensure that the queue will send the jobs only to the assigned block and nodes as discussed before. This can be done using TORQUE and Maui by disabling the `acl_host_enable` in the host access list. Further, the users who could submit jobs to a particular queue are limited in the user access list.

Below is a real example of the node and user mapping configuration in a queue using Queue Manager (`Qmgr`) :

```
create queue block01
set queue block01 queue_type = Execution
set queue block01 acl_host_enable = False
set queue block01 acl_hosts = node01
set queue block01 acl_hosts += node04
set queue block01 acl_hosts += node03
set queue block01 acl_hosts += node02
set queue block01 acl_user_enable = True
set queue block01 acl_users = user01
set queue block01 resources_max.cput = 24:00:00
set queue block01 enabled = True
set queue block01 started = True
```

In this case, the queue name associated to a particular block is `block01`. It is owned by a user `user01`, and consisting of 4 nodes : `node01`, `node02`, `node03` and `node04`. This configuration realizes the requirements for LPC :

- Constraining `user01` to be able to access only 4 mentioned nodes within a cluster `block01`.
- In contrary, another users are forbidden to access those nodes. That means those nodes are accessible exclusively for the user `user01`.
- The number of nodes and the nodes itself are fixed within the assigned usage period.

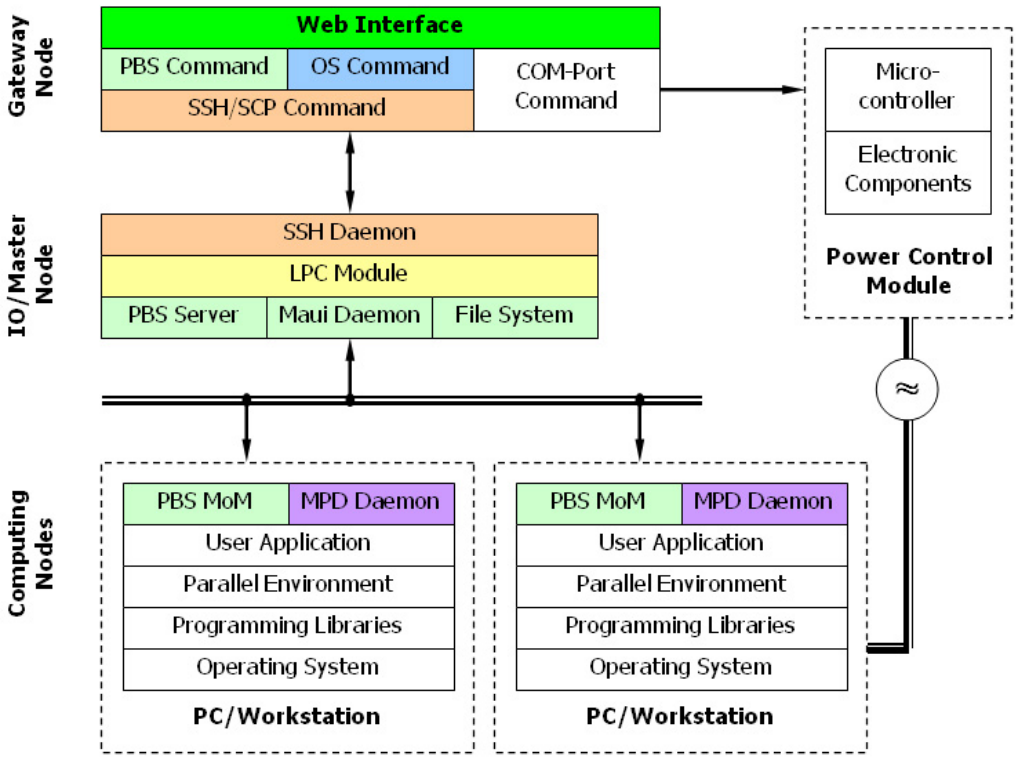


Fig. 2. The openPC components and interactions implemented on LPC.

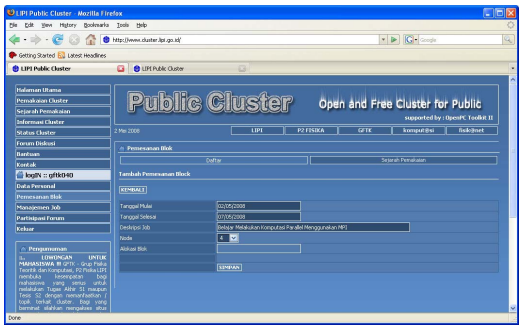
Although the users can reboot their nodes, this kind of configurations is automatically generated by the system. However, the users still have freedoms to choose any parallel environments available in the LPC. Anyway, LPC at time being provides standard MPI-based middlewares as OpenMPI [23], LAM/MPI [24] and MPICH2 [25].

C. Procedures and job managements

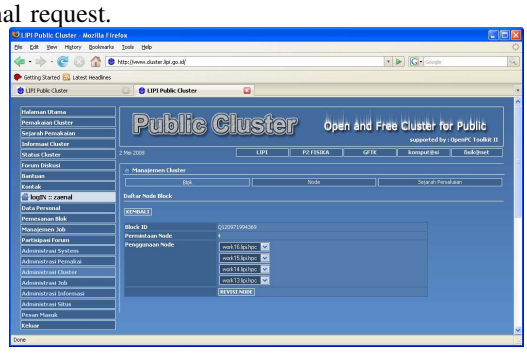
Here we describe the step by step procedures to perform a distributed computing job on LPC. There are two main procedures : the first one is the block activation by the administrator to turn up a block of nodes and make it available for the approved user. Secondly, the job management by users to perform any parallel jobs on the block allocated for them like compiling and executing the parallel programming.

First of all, we list the procedures for a block activation :

- A registered user apply for a block with certain number of nodes, the planned usage period and the detailed project description. We should note that the registration procedure for new users is trivial such that no need to explain it here.

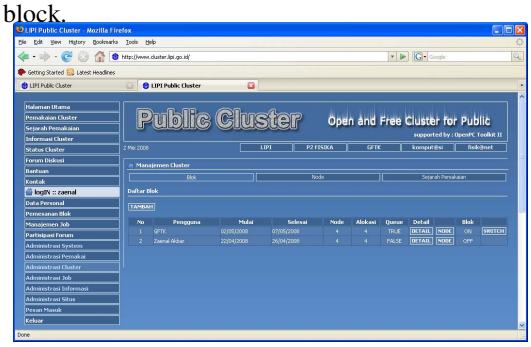


- The administrator verifies each request. We always put an attention to the congruity of the requested number of nodes with the planned job description. The allocated nodes are thereafter the subject of availability at the time, and the number is not necessary following the original request.

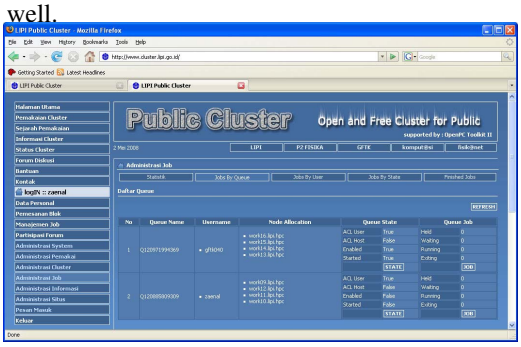


- Once approved, the allocated block and nodes are activated by the administrator. The nodes allocated for

the block are automatically turned on, and thereafter the system creates a particular queue by activating an access list for certain nodes and user as the single owner of his / her block.

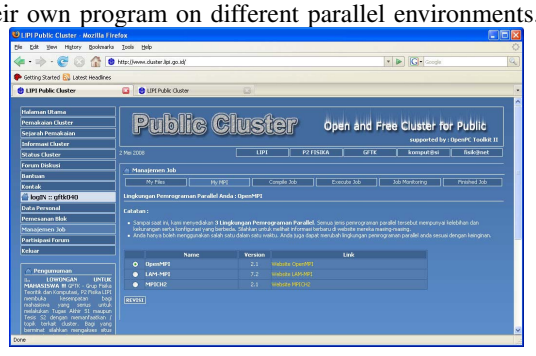


- During the running period, the administrator have a full access to monitor the block to ensure that everything works well.

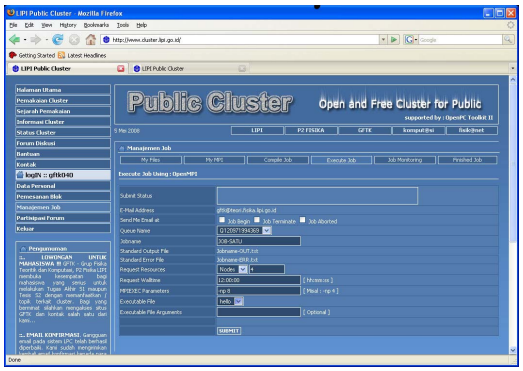


Once a user has been approved and assigned with a block of nodes, the user is ready to perform any jobs on it. Below is the procedures of job management at the LPC :

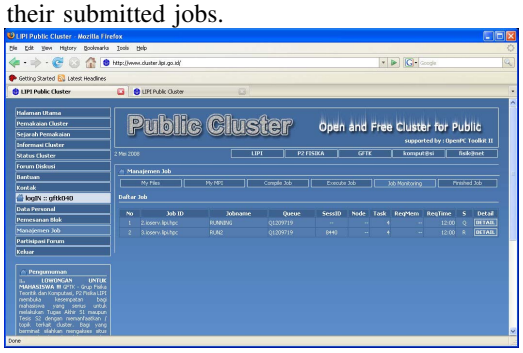
- The user should first choose a parallel environment from the available ones at the LPC. This freedom is quite unique and enables users to perform self-benchmarking of their own program on different parallel environments.



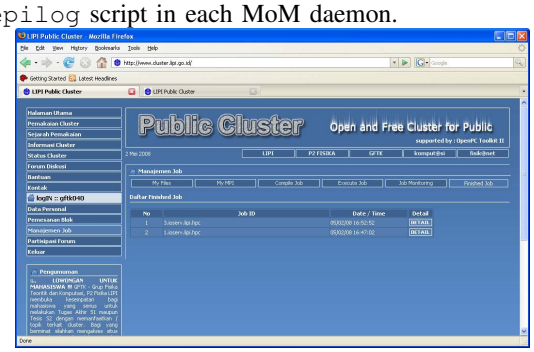
- The program compilations and executions are done through resource manager, in the case of LPC is TORQUE. Importantly, at this step the users are able to put any desired options relevant for the parallel environment being used. Again, the user is however forced to submit their jobs only within the allocated nodes through the queue assigned for them.



- The user can monitor, re-execute, suspend, stop and delete their submitted jobs.



- All executed jobs and its results including the detailed logs are stored by the system for user access later on. The users can download and evaluate the data subsequently. The detailed logs are generated as a job has been finished by adding an additional command to the `epilog` script in each MoM daemon.



V. PERFORMANCE

So far, the LPC has enjoyed numerous visits from both local and international visitors, especially from surrounding regions. Although at time being the interface are available only in Bahasa Indonesia, we receive tenths new registrations, and on an average ten percents are approved and becoming active users over a month. Most of them are college students conform the initial motivation of developing the LPC. Even not more than 10% of total approved users, some research groups also make use of LPC to perform much more advanced computational works.

For the sake of completeness, let us present a simple performance analysis on the reliability of multi-block approach before summarizing the paper. The performance analysis is done for the case of multi-block approach using a single master node as proposed originally in [22]. This approach is relevant

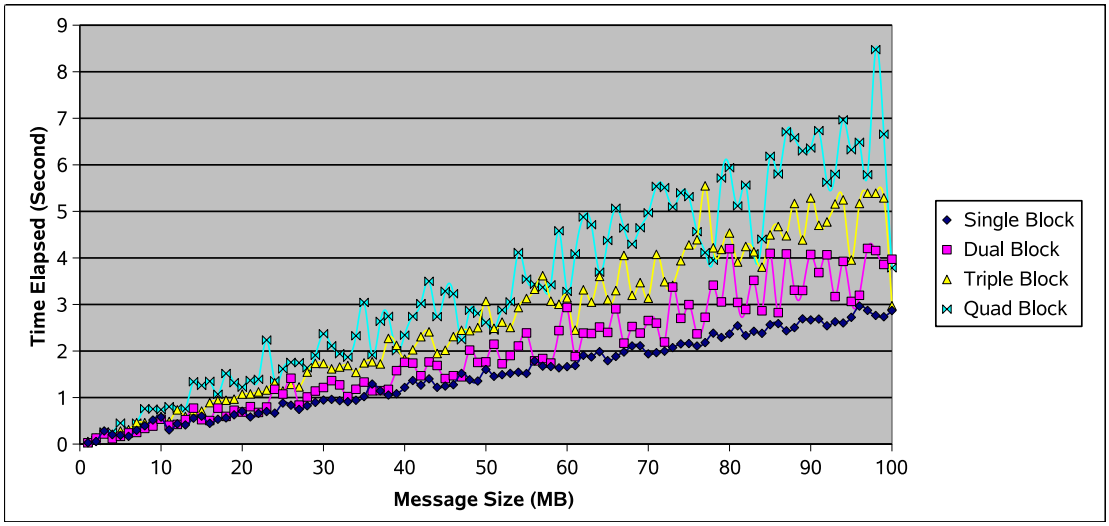


Fig. 3. Performance analysis for multi-block approach with single master for various block numbers.

for small scale clusters with limited hardwares, and are in particular suitable for processor / memory intensive computing works. So, it is quite interesting mechanism for public clusters where the blocks are consisting of small number of nodes.

The benchmarking result is given in Fig. 3. It has been done for $1 \sim 4$ block(s) of nodes with a single master node for all of them. The benchmarking is done by flooding the data to all nodes simultaneously where 30% of data go through the master node. The data size is set to $1 \sim 100$ MB with 1 MB incremental increase. The samples have been taken from the averages of six measurements for each size of data. From the figure, obviously the elapsed times are increasing as the flooded messages are greater. However for moderate data size, namely less than ~ 20 MB that is the considerable level of processor intensive works, the performance is still reliable. We can finally argue that the multi-block approach with a single master node is a good alternative for a small scale cluster divided into several independent blocks with few nodes in each of them.

We should note that this approach has been deployed at LPC during the initial period of implementation. Along with its development and various computational works performed on it, we have later on deployed multi-block cluster with independent master nodes in each block at LPC. Actually, we prefer to implement both approaches at LPC to optimize the existing infrastructures, since in some cases the users needs only few nodes in a block for very small scale computational works. The multi-block approach with a single master node suits this level of needs. We have unfortunately not succeeded in implementing both approaches simultaneously. Therefore, at time being all blocks at LPC have their own master nodes, regardless with its number of nodes [1]. There is another advantage on this architecture, that is it guarantees the total performance when the system is scaled up. This approach also improves the dependences among the blocks.

VI. CONCLUSION

We have introduced the whole aspects of openPC, an open toolkit to enable public access with full ownership to a cluster machine. We argue that such open facility is important to broaden opportunities for, in particular, young people and small research groups to perform and conduct advanced computing works using HPC environments. According to the feedback from our users in Indonesia during the last 3 years of implementation, the facility could provide a real experience on HPC, either for educational or real scientific purposes, as if they had their own HPC machines.

Now, we are entering the last phase of development. In this phase we are developing the MPI-based middleware dedicated for this kind of cluster to optimize the resources much better using particular resource allocation and management algorithms which fit its unique characteristics.

On the other hand, we also develop a web-service based module to connect a particular block of nodes in a public cluster to a global grid with any types of grid middlewares [26]. Through this web service, the users of public cluster are able to realize a collaboration with their global partners without having their own HPC machines. This work is in progress.

Lastly, concerning global audiences, the English version of web-interface and the complete manual is also under consideration. Since the whole system is now open for public under GNU Public License (GPL), volunteers to accomplish these works are welcome through SourceForge.net [8].

ACKNOWLEDGMENT

We greatly appreciate fruitful discussion with Slamet, B.I. Ajinagoro and G.J. Ohara throughout the initial period of the implementation of LPC. This work is funded by the Riset Kompetitif LIPI in fiscal year 2007 $\sim$ 2009 under Contract no. 11.04/SK/KPPI/II/2007 $\sim$ 2009, and the Indonesian Toray Science Foundation (ITSF) Research Grant 2006.

REFERENCES

[1] Z. A. et.al., "LIPI Public Cluster." [Online]. Available: http://www.cluster.lipi.go.id
[2] L. T. Handoko, "Public Cluster : mesin paralel terbuka berbasis web," *Indonesia Copyright no. B 268487*, 2006.
[3] Z. Akbar, Slamet, B. I. Ajinagoro, G. J. Ohara, I. Firmansyah, B. Hermanto, and L. T. Handoko, "Open and Free Cluster for Public," in *Proceeding of the International Conference on Rural Information and Communication Technology*, 2007, pp. 229–232.
[4] ——, "Public Cluster : parallel machine with multi-block approach," in *Proceeding of the International Conference on Electrical Engineering and Informatics*, 2007, pp. 207–209.
[5] Z. Akbar and L. T. Handoko, "Resource Allocation in Public Cluster with Extended Optimization Algorithm," in *Proceeding of the International Conference on Instrumentation, Communication and Information Technology*, 2007, pp. 286–289.
[6] I. Firmansyah, Z. Akbar, B. Hermanto, and L. T. Handoko, "Real-time control and monitoring system for LIPI Public Cluster," in *Proceeding of the International Conference on Instrumentation, Communication and Information Technology*, 2007, pp. 208–211.
[7] ——, "Microcontroller based distributed and networked control system for public cluster," in *Proceeding of the International Conference on on Quality in Research*, 2009, pp. 115–120.
[8] Z. Akbar, I. Firmansyah, B. Hermanto, and L. T. Handoko, "openPC : Toolkit for Public Cluster." [Online]. Available: http://www.sourceforge.net/projects/openpc/
[9] J. Mugler, "Open Source Cluster Application Resources (OSCAR)." [Online]. Available: http://oscar.openclustergroup.org
[10] Millennium Project Collaboration, "UC Berkeley Clustered Computing." [Online]. Available: http://www.ganglia.info
[11] Object Management Group (OMG), "Common Object Request Broker Architecture." [Online]. Available: http://www.corba.org
[12] E. Elmroth and M. Nylen and R. Oscarsson, "A User-Centric Cluster and Grid Computing Portal," in *Proceeding of the International Conference on Parallel Processing*, 2005, pp. 103–110.
[13] M. Thomas, S. Mock, M. Dahan, K. Mueller, D. Sutton, and J. R. Boisseau, "The GridPort toolkit: a system for building Grid portals," in *Proceeding of the 10th IEEE International Symposium on High Performance Distributed Computing*, 2001, p. 216.
[14] H. Gibbins and R. Buyya, "Gridscape: A Tool for the Creation of Interactive and Dynamic Grid Testbed Web Portals," in *Proceedings of the 5th International Workshop on Distributed Computing*, 2003, p. 837.
[15] I. Foster, "Globus Toolkit Version 4 : Software for Service Oriented Systems," *Journal of Computational Science and Technology*, vol. 21, pp. 513–520, 2006.
[16] P. Uthayopas, S. Phatanapherom, T. Angskun, and S. Sriprayoonsakul, "SCE : A Fully Integrated Software Tool for Beowulf Cluster System," in *Proceedings of Linux Clusters: the HPC Revolution*, 2001.
[17] A. Apon, K. Landrus, J. Mache, and E. Huxtable, "Using Shibboleth to Manage Access Control to Remote Cluster Computing Resources," in *Proceedings of SC2004, 17th ACM/IEEE High-Performance Computing, Networking and Storage Conference*, 2004.
[18] E. Landrus, "WebMPI - a Secure Cluster Web Interface using ShibbolethWebMPI," Master's thesis, University of Arkansas, 2005.
[19] Cluster Resources Inc., "Terascale Open-source Resource and QUEue manager (TORQUE)." [Online]. Available: http://www.clusterresources.com
[20] ——, "Maui Cluster Scheduler." [Online]. Available: http://www.clusterresources.com
[21] Altair Engineering Inc., "Open portable batch system (openpbs)." [Online]. Available: http://www.openpbs.org
[22] Z. Akbar and L. T. Handoko, "Multi and Independent Block Approach in Public Cluster," in *Proceeding of the 3rd Information and Communication Technology Seminar*, 2007, pp. 216–218.
[23] Open MPI Team, "Open MPI: Open Source High Performance Computing." [Online]. Available: http://www.open-mpi.org
[24] LAM Team, "LAM/MPI Parallel Computing." [Online]. Available: http://www.lam-mpi.org
[25] W. Gropp et.al., "MPICH2 : High-performance and Widely Portable MPI." [Online]. Available: http://www.mcs.anl.gov/mpi/mpich/
[26] Z. Akbar and L. T. Handoko, "GRID Architecture through a Public Cluster," in *Proceeding of the International Conference on Computer and Communication Engineering*, 2009, pp. 1016–1018.

Z. Akbar is a young researcher at the Group for Theoretical and Computational Physics, Research Center for Physics, Indonesian Institute of Sciences (LIPI). He is majoring on Computing Science, but his research is focused on the distributing system. Since 2009 he also joined the Group for Bioinformatics and Information Mining, Department of Computer and Information Science, University of Konstanz as an associate researcher and working on distribution system for bioinformatics problems. His URL is http://teori.fisika.lipi.go.id/ zaenal/.

I. Firmansyah is a junior researcher at the Group for Theoretical and Computational Physics, Research Center for Physics, Indonesian Institute of Sciences (LIPI). Since 2009 he also joined the Center for Environmental Remote Sensing, Chiba University as a research scientist. His research interest is mainly on distributed instrumentation system. URL is http://teori.fisika.lipi.go.id/ firmansyah/.

B. Hermanto is a junior researcher at the Group for Theoretical and Computational Physics, Research Center for Physics, Indonesian Institute of Sciences (LIPI). His research interest is mainly on distributed networking system. His URL is http://teori.fisika.lipi.go.id/ hermanto/.

L.T. Handoko is a senior researcher and Group Head at the Group for Theoretical and Computational Physics, Research Center for Physics, Indonesian Institute of Sciences (LIPI). He is also a visiting professor at the Department of Physics, University of Indonesia. His research interest covers broad area from theoretical particle physics to computational science. His URL is http://teori.fisika.lipi.go.id/ handoko/.