PROCEEDINGS OF SCIENCE

A Resource-saving Job Monitoring System of High Performance Computing using Parent and Child Process

Kajornsak Piyoungkorn [1]
National Electronics and Computer Technology Center
112 thailand Science Park, Phahonyothin Rd,
Khlong Nueng, Khlong Luang, Phathum Thani 12120
Thailand
E-mail: `kajornsak.piyoungkorn@nectec.or.th`

Phithak Thaenkaew
National Electronics and Computer Technology Center
112 thailand Science Park, Phahonyothin Rd,
Khlong Nueng, Khlong Luang, Phathum Thani 12120
Thailand
E-mail: `phithak.thaenkaew@nectec.or.th`

Chalee Vorakulpipat
National Electronics and Computer Technology Center
112 thailand Science Park, Phahonyothin Rd,
Khlong Nueng, Khlong Luang, Phathum Thani 12120
Thailand
`E-mail: chalee.vorakulpipat@nectec.or.th`

High performance computing has been more important in the past decade. In the present day, data used for processing becomes enormous. Where a high performance computing resource is needed to help process the data. Some scientific experiments involving big data. Which requires high speed data processing cannot be done by an ordinary computer system. Also, there is a need for support of parallel processing. The solution starts by dividing the job into a number of sections to be processed into parts and the processing unit each processing unit of data at the same time. Then, the system sends the calculated result back to the compiled. This mechanism will speed up the processing time to complete the task and generate more output at the same time. Therefore, a solution in this study is to maximize efficiency when using the resources of the computer which involves the processing power of the processor (CPU Cores).When the HPC system has a large number of concurrent users and requests processing resources that do not match the actual usage. Therefore requires a system to detect job requests that use inefficient computing resources to help users and system administrators to work effectively.

International Symposium on Grids & Clouds 2019, ISGC2019
31st March - 5th April, 2019
Academia Sinica, Taipei, Taiwan

[1]Speaker

© *Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).*

1. Introduction

The mechanism can be explained in a scenario as follows. In Thailand, there are government agencies National e-Science Infrastructure Consortium [1] that provide free of charge high performance computing services [2] to facilitate researchers to conduct their research. It is usual that many users request a resource overrun or uses computing resources in an inefficient way. This is because they are not aware of the over-consumption of the resources, leading to unnecessary high costs. For example, a user requests computing resources that does not match the actual usage. Resource requests are calculated in high numbers for maximum processing speed that does not correspond to actual usage, resulting in resource wasting. Thus, a negative effect will go to the hardware system and it will be a hindrance to other users who have to lose an opportunity to use it.

2. Related research

In our previous study [3], we start checking all running jobs in the system from number Job-ID. The CPU load that represents the CPU utilization is over or not 100%. Process-ID is then analyzed by the Job-ID. Displays user details of how many CPU-core requests are made.Then, it compares with the current CPU-load that works exactly as requested.We use the tolerance of 20%. If the current resource has a CPU-load greater than 120% or less than 80%, it will be assumed that the work is running, and the CPU-core request does not match the actual use. It means that the use of resources is not effective. The system alerts users via email or eliminates the process [4]. However, using this mechanism, the CPU-load is not correctly analyzed because some software usesunstable CPU-core. As a result, the decision of the system is wrong. Work diagram shown in Figure 1.

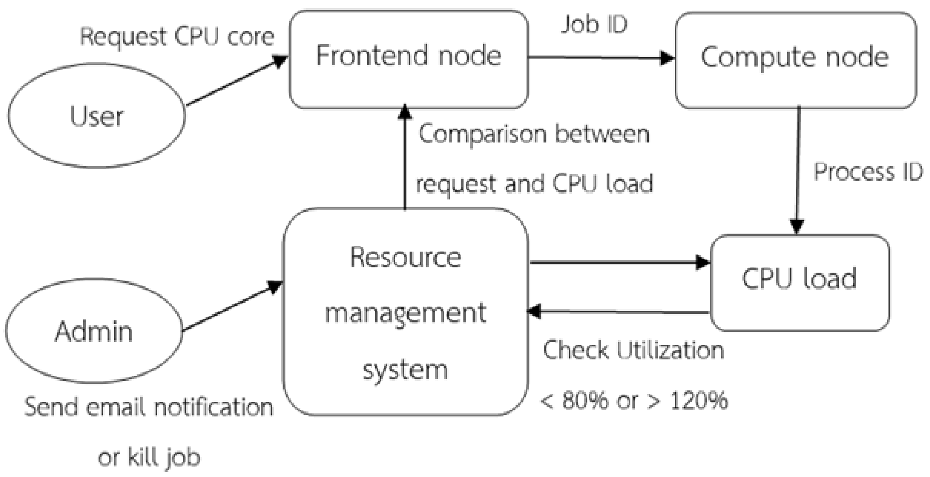


Figure 1: Previous study (Utilization method).

PoS(ISGC2019)034

3. Research methodology

In this study,the analysis of the run in the system uses a technique of parent [5] and child process [6] to determine how many CPU threads to be compared to the Job-ID number. Based on the preliminary results of the same experiment, it was found that the accuracy was increased from the original method to 20%. This method is more accurate than the original method, since the CPU-thread counts the number of CPU cores from the compute node currently in use, regardless of the CPU-load that is the variance of the number. And consistent with the CPU-core value that the user requested is numbered in integer format, such as CPU-threads 4 and CPU-cores 4. The new approach can determine whether the number of CPU-threads and CPU-cores are equal or not. In other word, it can answer if the request or usage of the resources is efficient, and finally it can automatically send alert to the user through email and terminate process if the user agrees.

3.1 Job Management System

A solution in this study is to maximize efficiency when using the resources of the computer which involves the processing power of the processor (CPU-core). For example, a user requests computing resources that does not match the actual usage. Resource requests are calculated in high numbers for maximum processing speed that does not correspond to actual usage, resulting in resource wasting. Thus, a negative effect will go to the hardware system and it will be a hindrance to other users who have to lose an opportunity to use it.

Each day, the HPC resources have to execute many jobs to produce productivity in scientific research. Normal job requests have to specify requirements such as number of CPU-core processors per compute node, number of compute nodes, Type of Queue (running length for Ex. short 1 day, medium 3 days and long 7 days) If there are users who request inefficient resources will effects.

- The other users have to wait longer for their job to execute.
- The administrator needs to check the HPC resources repeatedly.
- The hardware needs to run for long hours continuously to complete executing the job.
- The machine gets worn out faster than its life expectancy.
- The hardware overheating problem.
- The cooling system works hard in the data center room. The cost of maintaining the system and increasing electricity costs.

The Job Scheduler (PBS) [7] is responsible for managing jobs that are submitted to the HPC by users and usually provides four main functions, Job Submission, Schedules, Control, and Monitor. Which HPC resources are managed by a Job Management System works to manage the CPU resources of the HPC according to the jobs requested in the PBS script [8].

PoS(ISGC2019)034

3.2 e-Science HPC Cluster specification and PBS Script

The National e-Science Infrastructure Consortium is a non-profit organization operated by the government, that provides free HPC service to Academia and Research Institutions in Thailand. The users are experts from a variety of fields such as High Particle Physics, Chemistry, Biology, Nanotechnology, Pharmacy, and many more. For the HPC cluster used in system testing, the following items are listed below.

- Frontend node (Job Management System): CPU 16 Cores.
- Compute node: CPU 192 Cores x 5 nodes = 960 Cores.
- OS: Linux CentOS 7.
- Cluster Management: Beowulf Cluster.
- Job Scheduler: PBS/Torque Scheduler. [9]
- Scientific applications: Gaussion09, Quantum Espresso, Gromac, Abinit, Amber, and Autodoc.

Example PBS script from users to request resources.

```
#PBS -N test                 //name of job
#PBS -S /bin/bash            //Use Linux bash to execute script [10]
#PBS -l nodes=192:ppn=1      //User request resources (CPU Cores node)
#PBS -q long                 //User select queue (7 days)
```

The actual engagement of the HPC resources involves one compute nodes of 192 processors per node for 168 hours. If actually using only 32 core for 3 days. The HPC resources are lost for 48 hours without productivity. As a consequence, the queues waiting for their job execution increase continually, while the booked HPC resources are idle.

3.3 Inefficient job inspection procedures

The procedure for checking inefficient work is shown using pseudo code and shown in Figure 2

```
1: Get value Job ID run time over 5 minutes. as string
2: Calculate Resource Usage from Job ID detail (Process ID)
3: Determine PBS script from Job ID (no. of CPU Core)
4: While not end of last Job ID do
5:      Compare Resource Usage and PBS Script Request;
6:      if CPU cores & Threads = PBS Script Request then
7:           go back to the beginning of current section;
8:      else
9:           Send notifications through email or terminate process
10: end while
11: finish
```

PoS(ISGC2019)034

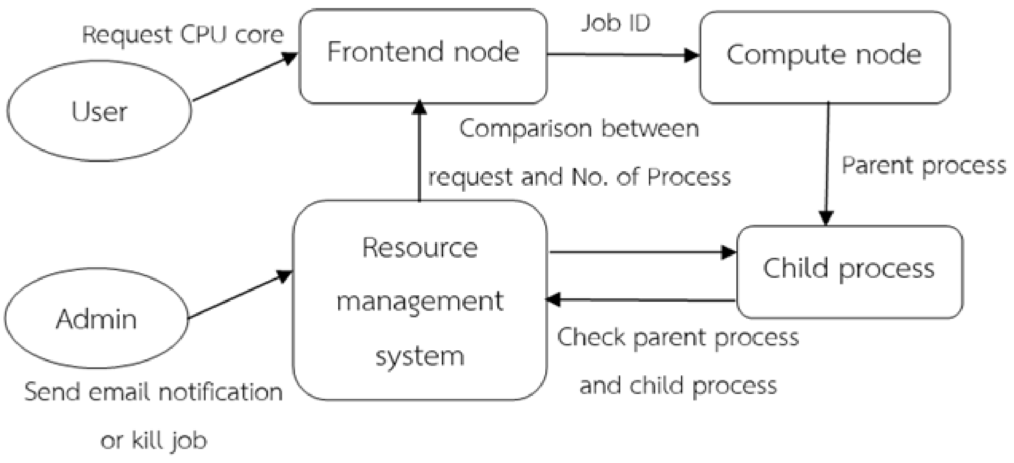


Figure 2: Current study (Parent and child process method).

3.3.1 Inefficient inspection of the job using the parent and child process method

We use the parent and child process. It is a method that can accurately determine the amount of CPU-core that is actually used more accurately than the CPU-load measurement method because the process that is executed by the user is constant from start to finish data processing. The elements of the parent and child process methods are as follows: parent process, fork(), child process, exec(), exit(), wait() which has the following details

The fork() is the system call function of the operating system used to create the process. The process is called a child process, and both processes are executed only when successfully created. The fork function does not require an argument and returns the process ID value.

To fork to think that the process is real, and the fork is to separate the body into another process, which is actually called the parent process and the another process is called a child process. The child process will have all the same features as the parent process.

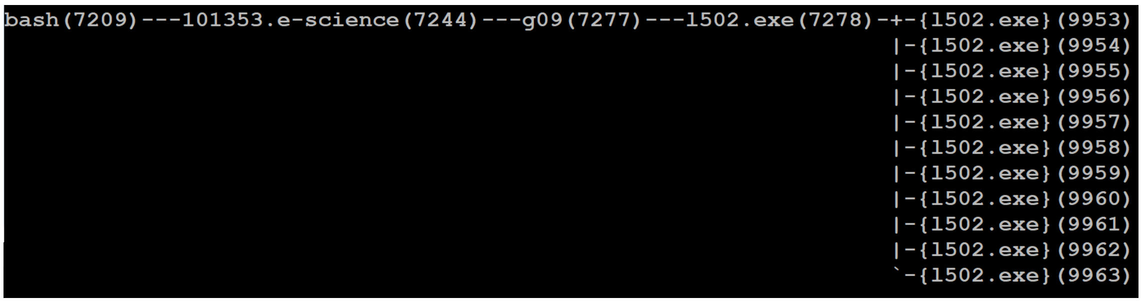
When getting a child process, it will work according to the order. When work is done, it will be eliminated exit (). For the parent process, when creating a successful child process, it will follow the following commands separately with the child process. The parent process waits for wait () until the child process is finished and eliminated therefore the parent process can be complete.

Example of the workflow shown below. When users request resources from PBS Script and send the work to process.

```
#PBS -N gaussion09
#PBS -S /bin/bash
#PBS -l nodes=32:ppn=1
#PBS -q medium
```

From PBS script, users run the program gaussion09 by requesting resources by requiring 32 CPU cores. Actual work shown in Figure 3.

PoS(ISGC2019)034



```
bash(7209)---101353.e-science(7244)---g09(7277)---1502.exe(7278)-+-{1502.exe}(9953)
                                                                 |-{1502.exe}(9954)
                                                                 |-{1502.exe}(9955)
                                                                 |-{1502.exe}(9956)
                                                                 |-{1502.exe}(9957)
                                                                 |-{1502.exe}(9958)
                                                                 |-{1502.exe}(9959)
                                                                 |-{1502.exe}(9960)
                                                                 |-{1502.exe}(9961)
                                                                 |-{1502.exe}(9962)
                                                                 `-{1502.exe}(9963)
```

Figure 3: Check the parent and child process of the gaussion09 program.

Figure 3 shows the use of the parent and child process method will get the result. 1502.exe is the parent process (Process ID 7278) and the child process is 1502.exe (Process ID 9953-9963).

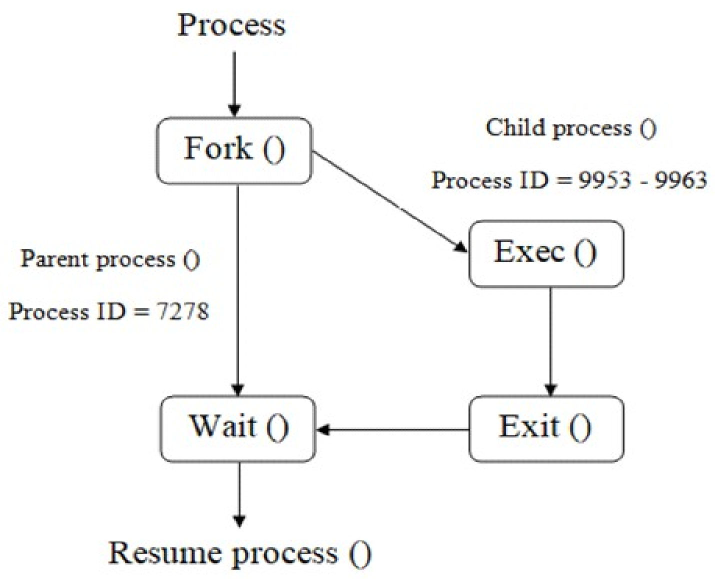


Figure 4: Parent and child process method.

Figure 4 shows the structure of the process that is running at that moment, how the parent process and the child process are stacked. Then add the two processes together and compare it with the PBS script that the user requested. The sum of the parent and child processes will be equal to 12 when compared to the requested PBS script 32 cores. In conclusion, this job is an inefficient processing process due to having 20 unused CPU cores. Which can be seen that users request resources not exactly the actual use. The system will notify the user via email and the administrator can decide whether to kill this job. When using this system on an e-sci HPC Thailand will detect inefficient work every 5 minutes and recorded in log files for analysis.

PoS(ISGC2019)034

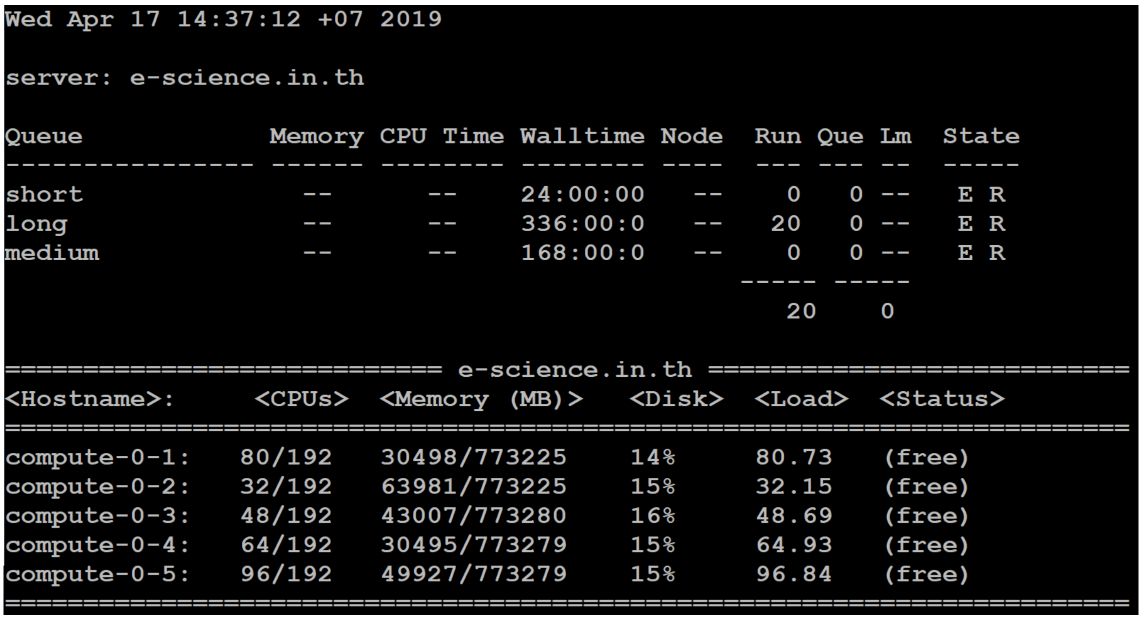
3.4 Implementing the parent and child process method on the e-Science HPC service

We test for real use. By simulating the event that the user requests inefficient work for 3 events as follows:

- Users request less resources than they actually use. (20 jobs)
- Users request more resources than they actually use . (20 jobs)
- Users request resources exactly as they are actually used. (20 jobs)

Each case has a total of 20 jobs. Total processing in the system will have a total of 60 jobs and takes 7 days to process. Job monitoring system with the parent and child process method will list all resources of the compute node and tell the CPU load to show the CPU status. Experimental results are shown in Figure 5.

PoS(ISGC2019)034



```
Wed Apr 17 14:37:12 +07 2019

server: e-science.in.th

Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
short              --      --    24:00:00   --    0   0 --   E R
long               --      --    336:00:0   --   20   0 --   E R
medium             --      --    168:00:0   --    0   0 --   E R
                                               ----- -----
                                                  20    0

============================ e-science.in.th ============================
<Hostname>:      <CPUs>  <Memory (MB)>   <Disk>  <Load>  <Status>
=========================================================================
compute-0-1:    80/192   30498/773225    14%     80.73   (free)
compute-0-2:    32/192   63981/773225    15%     32.15   (free)
compute-0-3:    48/192   43007/773280    16%     48.69   (free)
compute-0-4:    64/192   30495/773279    15%     64.93   (free)
compute-0-5:    96/192   49927/773279    15%     96.84   (free)
=========================================================================
```

Figure 5: Showing resource usage and CPU status.

Figure 5. The results of the experiment concluded that the system could detect inefficient work. Which users request less or more resources that actually use for a total of 40 jobs and eliminate inefficient job. Enabling the system to run tasks that users request exactly as they are actually used.

4. Results

We tested with 6 scientific programs as follows: Gaussion09, Quantum Espresso, Gromac, Abinit, Amber and Autodoc. That are the most used software. Then sending the job to process 10 times per program and request resources to process 16 CPU cores per job, then measuring t he efficiency of accuracy. The results can be summarized as shown in Table 1.

Scientific program	**User requests CPU without job monitoring**	**Utilization method (Previous study)**	**Parent and child process method**
	Accuracy / No. of jobs	**Accuracy / No. of jobs**	**Accuracy / No. of jobs**
Gaussion09	7 / 10	9 / 10	10 / 10
Quantum Espresso	6 / 10	7 / 10	9 / 10
Gromac	7 / 10	8 / 10	9 / 10
Abinit	7 / 10	9 / 10	10 / 10
Amber	7 / 10	8 / 10	10 / 10
Autodoc	6 / 10	7 / 10	9 / 10
Total	**40 / 60 (67%)**	**48 / 60 (80%)**	**57 / 60 (95%)**

Table 1: Measuring accuracy in detecting inefficient job.

We perform 3 performance measures:1. Users request resources without job monitoring system. 2. Use the method to check utilization (Previous study) and 3. Use the parent and child process. From the test, it was found that the examination by the parent and child process can detect the job that the user requests inefficiently with the most accurate resources (95%). Because the system checks the real time to a process that is actually used on the operation system directly, which the return value is a constant value and is in a format integer which can be compared directly with the value requested by the user in PBS Script.

Then measure the CPU-load performance of the compute node in total of 5 nodes. Each node has 192 CPU-cores. After detecting inefficient work from Table 1, the CPU-load and CPU temperature are measured. The results of the experiment can be shown as Table 2.

Compute nodes (CPU 192 Cores per nodes)	**User requests CPU without job monitoring**	**Utilization method (Previous study)**	**Parent and child process method**
	CPU load / Temp (^{o}c)	**CPU load / Temp (^{o}c)**	**CPU load / Temp (^{o}c)**
Compute-01	202.32 / 66^{o}c	192.05 / 52^{o}c	192.04 / 51^{o}c
Compute-02	112.46 / 57^{o}c	182.76 / 59^{o}c	193.01 / 53^{o}c
Compute-03	70.33 / 66^{o}c	193.55 / 54^{o}c	191.94 / 51^{o}c
Compute-04	96.08 / 59^{o}c	196.49 / 61^{o}c	192.03 / 53^{o}c
Compute-05	223.48 / 68^{o}c	189.72 / 56^{o}c	191.88 / 52^{o}c
Usage results	**Inefficiency**	**Good efficiency**	**Max efficiency**

Table 2: CPU performance measurement and CPU temperature averages.

Table 2, shows that the parent process has a CPU-load close to the CPU-cores, which is more effective than the original method. Due to the use of CPU, cost effective and highest efficiency Resulting in a lower average CPU temperature.

Then measure the temperature while the compute node is processed during the week to check for side effects in using the job monitoring system with the parent and child process.

PoS(ISGC2019)034

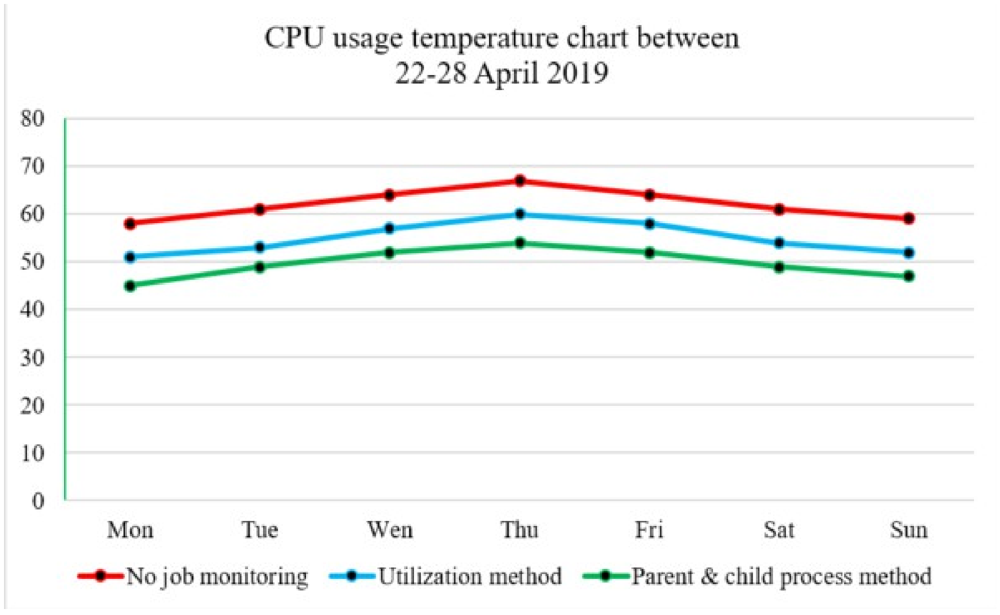


Figure 5: Measure CPU usage temperature (ºc) within 1 week.

From Figure 5 shows the CPU operating temperature in different ways. The test results can be concluded that the parent and child process methods can reduce the working temperature to the best. Can reduce the temperature from the utilization method to around 5ºc because the CPU-load operation has a full load working condition. The CPU-cores are used at full efficiency, every cores with an average temperature of 50ºc and the highest temperature is 54ºc. Since the HPC system will have a large number of users on the middle of the week Unlike using non-job monitoring, the temperature is up to 67ºc due to overload. This causes heat to the hardware system. Leading to premature hardware deterioration. The benefits of using the job monitoring system with the parent and child process will help the HPC system to work more efficiently. Reduce resource usage, resulting in lower operating temperature, extend system life. Hardware reduces maintenance costs and saves energy in the data center room as well.

5. Conclusion and future work

Job monitoring system using the Parent and child method developed to help manage resources. For providing HPC services with a large number of concurrent users. It is useful for helping users to be able to send jobs to run correctly and create awareness for users to use resources efficiently. Helps the work that is waiting to be processed in the queue runs faster. To help system administrators to manage resources more easily. As a result, use of resources effectively. Extend the life of the hardware and the cost of maintaining the system better.

In the future, The system also has errors in detecting certain types of tasks, such as jobs that have a delay in processing time. Will make the system see that the software is an inefficient resource request Which increases the efficiency of accuracy to be studied only with certain software types. The current working system is also a command line. Users and administrators must have knowledge of Linux operating systems, making it difficult for those who have previously used graphics. Which the system can be further developed into a beautiful GUI in the future.

PoS(ISGC2019)034

6. Acknowledgement

Special thanks to the National e-Science Infrastructure Consortium, National Electronics and Computer Technology Center (NECTEC), and the National Science and Technology Development Agency (NSTDA) Thailand.

References

[1] "National e-Science Infrastructure Consortium.", `http://www.e-science.in.th.`

[2] S.H Al-Khazraji, N.M. Abdullah, M.A. Younus Al-Sa'ati., *Building High Performance Computing Using Beowulf Linux Cluster: International Journal of computer Science and Information Security, Vol 12, No.4,* April 2014.

[3] K.Piyounkorn, P.Thaenkaew, N.Kasisopha, C. Vorakulpipat, *Automating Job Monitoring for an Ecosystem of High Performance Computing: Medes'17,* November 2017.

[4] M.Li, Y.S.Zhang, *Job Monitoring Analysis Tool for HPC Clusters: The 1st International Conference on Information Science and Engineering (ICISE2009),* 2009.

[5] "Parent Process.", `https://en.wikipedia.org/wiki/Parent_process.`

[6] "Child Process", `https://en.wikipedia.org/wiki/Child_process.`

[7] U. Schwiegelshohn, *How to Design a Job Scheduling Algorithm,* Dortmund, Germany 44221.

[8] "HPC Center for High-performance Computing", `https://hpcc.usc.edu/support/documentation/running-a-job-on-the-hpcc-cluster-using-pbs.`

[9] "The PBS Job scheduler", `http://www.arc.ox.ac.uk/content/pbs-job-scheduler`

[10] "Shell programming with bash", `http://matt.might.net/articles/bash-by-example.`

PoS(ISGC2019)034