CPU Utilization Control System In Distributed Computing

C. K. Low, T.F. Ang, T.C. Ling, K.K. Phang, L.Y. Por

Abstract— Distributed computing system has been used as a tool to solve many large-scale parallel computational problems. However, due to the heterogeneity of the resources as well as applications, most of the dependent parallel tasks are not able to complete at the same time. If the waiting time for task processing increases, overall performance can be degraded significantly cause by the synchronization between parallel tasks. This research proposes a central processing unit (CPU) Utilization Control system, which is able to throttle the CPU usage by periodically, forces the associated processes to be idle for a short time. The released CPU power then can be allocated for other local jobs without scarifying the performance of dependent tasks.

Index Terms—Distributed Computing, CPU Utilization Control, Control System, Dependent Task, Resource Heterogeneity

1 Introduction

ONE of the major challenges in the distributed environment is the effectiveness of resource allocation for task execution [1]. Assigning right resources to the right task is crucial for the performance of task completion. It requires the understanding on task resource requirements. For example, given two machine X and Y, machine X might be more efficient on executing task A compared to machine Y. Instead, machine Y is the best to run certain types of applications. Various researches conducted in the discipline of task scheduling has addressed the above issues [2][3][4][5].

The inefficiency in task scheduling causes the failure in completing the remote tasks within the predicted time. The situation becomes worse with the presence of workflow job. Workflow job comprises of multiple dependent tasks, which have to be executed in a predefined order. Therefore, a task cannot start until its entire parent sub-tasks are completed. Since the execution time of each task is difficult to predict, those completed tasks have to wait for the unfinished tasks to be completed before the workflow job can move to the next processing stage.

Dynamic CPU utilization control (DCUC) mechanism is proposed in this article to maximize the resources utilization of associated machines and shorten the waiting time between the dependent tasks. This mechanism enables the estimation of resource capability in real-time and controls the total execution time needed by both local and remote tasks. This is achieved by reducing the CPU usage assigned to the remote tasks. The freed CPU resources can then be reallocated to other local tasks. As a result, this mechanism provides better resource utilization through reduction of local tasks makespan.

- *C.K. Low is currently pursuing masters degree program in computer science in University of Malaya, Malaysia. E-mail: keatlck@gmail.com*
- *T.F. Ang is a senior lecturer in computer science in University of Malaya, Malaysia. E-mail: angtf@um.edu.my*
- *T.C. Ling is an associate professor in computer science in University of Malaya, Malaysia. E-mail: tchaw@ um.edu.my*
- *K.K. Phang is an associate professor in computer science in University of Malaya, Malaysia. E-mail: kkphang@ um.edu.my*
- *L.Y. Por is a senior lecturer in computer science in University of Malaya, Malaysia. E-mail: porlip@ um.edu.my*

From the literature [5][7], we found out that there are no suitable off-the-shelf tools to be used in this research to dynamically control the CPU utilization. As a results, a new dynamic CPU utilization controller (DCUC Controller) is proposed and developed in this research. The DCUC Controller is developed in Microsoft .NET environment.

The paper is organized as follows. Section 2 discusses the literature review on CPU control system while the proposed system architecture is described in Section 3. Section 4 provides the snapshot to demonstrate how the allocation of CPU resources is control. Lastly, a conclusion for the conducted research is presented in Section 5.

2 Related works

There are various CPU control tools [8][9][10] in market, which are used to control the CPU core, its frequency and utilization. For Window operating systems such as Windows XP, Windows Vista, and Windows 7, its Windows Task Manager [6] is capable of controlling the CPU of a particular system.

By default, applications can run on all available cores of a processor. If a system has more than one processor core, affinity of an application can be assigned to control which core of the processor can be used by the application. Depend on the situation, users can choose to prioritize critical applications by assigning more processor cores to the application. This will eventually improves the overall performance of the system. Besides, certain legacy applications are optimized to run on systems with single core CPU. Problems would arise if these legacy applications were to run on machines with multiple cores [7].

Most of the computers today are equipped with CPU control mechanism embedded in the BIOS. The CPU control serves as a low-level power saving mechanism especially in laptops. For example, when a computer is running on battery power instead of AC (alternating current) power supply, the CPU control will throttle down the CPU frequency (dynamic frequency scaling) as a mean to conserve battery power.

Besides that, those recent Microsoft Windows based operating systems also allow the CPU usage to be adjusted through power settings. By throttling down the CPU's frequency, the mechanism does not only save battery power but also helps reduce the CPU temperature. However, the throttling does negatively impact the applications and operating systems performance. As a result, users need to decide the trade-off between system performance and battery life based on the nature of usage.

Besides the mentioned built-in CPU control mechanisms, efforts had been made to utilize third-party software in order to achieve better balance between system performance and battery life. For example, SpeedswitchXP [8] is a third party software that allows dynamic switching of the frequencies of CPU. SpeedStepXP enable users to control the CPU frequencies of Windows based operating systems. In Windows XP environment, users are restricted to manipulate the power scheme function. The power scheme function in this context refers to how the operating system should behave with or without AC power supply. SpeedswitchXP attempts to solve this limitation by offering users access to different type of power scheme setting.

ThreadMaster [9] monitors threads and handles high CPU utilization on a per application basis. This function is primarily designed for terminal servers to host multiple users. ThreadMaster can also be used to control the CPU utilization on each virtual machine when multiple virtual machines are running on the same server. ThreadMaster is a background service which implements a CPU quota mechanism on top of the standard thread scheduling algorithms of Microsoft Windows 2000. All running applications are observed by ThreadMaster. The CPU usage of a particular thread will be throttled by ThreadMaster in real time if it exceeds the utilization threshold. However, this program is not suitable for this research due to the following shortcoming:

i) Unable to control which application to clamp for CPU
ii) Percentage of CPU utilization is not controllable
iii) Unable to handle thread in multicore or hyper threading environment

Another CPU control tool, namely Battle Encoder Shirase (BES) [10] is found to be more suitable to address the problem above. BES is a tool that throttles the CPU usage of a process. For example, the CPU usage of a process can be adjusted from 100% down to 40%. By using this method, users can run both remote and local jobs simultaneously while maximizing the CPU utilization. BES can also act as an active CPU cooler software. CPU can be cool down instantly by limiting the CPU load. Traditional soft-coolers switch idle CPU into sleep-mode in order to save CPU energy. On the other hand, BES actively cool down the CPU by throttles down the "over warming" process i.e. intermittently constraining CPU to be idle for a short time. Some of the advantages of BES include:

i) Able to control which application to be clamped for CPU
ii) Able to control the percentage level of CPU utilization
iii) Able to handle thread in multicore or hyper threading environment

However, BES does not have the functionality to adjust the CPU dynamically. Any adjustments of the CPU utilization have to be done manually. Our proposed mechanism is intended to estimate resource capability in real-time and adjust the CPU utilization on-the-fly. None of the existing tool is suitable to fulfil this requirement. As a result, a dynamic CPU utilization controller is proposed and developed in this research.

3 System Architecture

The proposed CPU Utilization Controller is a platform for developing distributed application on distributed system. It harnesses the spare CPU cycles of a heterogeneous network which linked the PCs and servers on demand. System administrators can leverage a collection of tools to monitor and control the deployed infrastructure. The proposed CPU Utilization Control is based on the .NET framework. Since it is operating in Microsoft platform, it is unique from a technology point of view as opposed to the widely available Java based solutions.

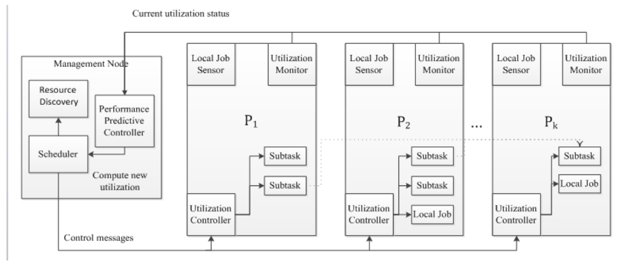


Fig. 1. Utilization control architecture

Fig. 1 shows the proposed new utilization control architecture in this research. In order to optimize the resource usage, the control architecture is embedded with a control loop, which execute in each checkpoint. Checkpoint in this context refers to the checkpointing technique that snapshot the current application state for recovery purpose. First, the control loop dynamically controls the utilizations of all the processors. Utilization monitor on each processor p_k sends its utilization u_l in the current checkpoint to the Performance Predictive Controller. A new utilization is generated by the controller for each subtask $T_{i,j}$ and the new utilization is sent to the Utilization Controller. The controller changes the utilization accordingly. Local job sensors are used to observe local jobs activities and the results are sent to Utilization Monitor. The detail functionality of each component is described as follow:

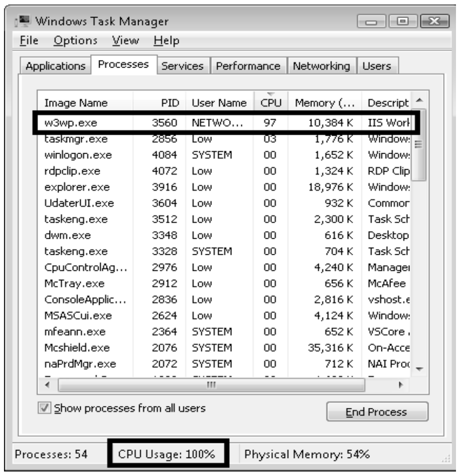
- **Performance Predictive Controller (PPC):** The controller that used to predict the behaviour of the running tasks. The prediction information is send to the scheduler to make scheduling decision.
- **Local Job Sensor:** The sensor extends the job monitor-

ing sensor in order to observe local jobs.

- **Utilization Monitor:** The utilization monitor uses the PerformanceCounter class in .NET framework to read the CPU utilization in each checkpoint signal time. The Processor performance category has the % Processor Time performance counter object, which has the _Total performance counter instance.
- **Utilization Controller:** The controller is implemented as a process running on each executor node. In each checkpoint period, the controller periodically reads the CPU utilization of the Performance Predictive Controller to generate new CPU utilization.

4 Prototype Implementation

A testbed which implements the above components is implemented to carry out testing and evaluations of the CPU utilization control prototype. The testbed consist of one executor which serves as a web service task. The prototype is then used to control the CPU utilization of the mentioned web service task. This allows us to test the efficiency and usability of the prototype.


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Show processes from all users
End Process

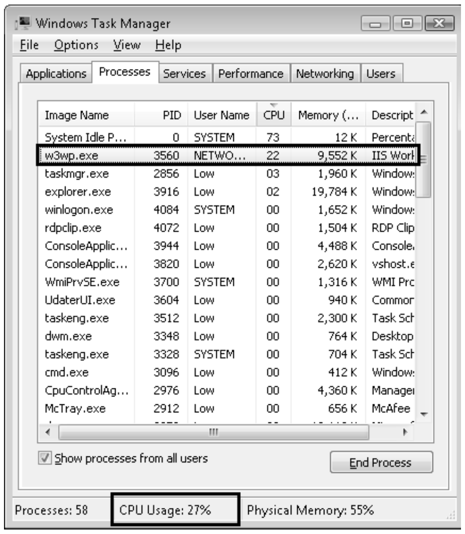
```
Image Name        PID   User Name  CPU  Memory (...  Descript
w3wp.exe          3560  NETWO...   97     10,384 K   IIS Work
taskmgr.exe       2856  Low        03      1,776 K   Window:
winlogon.exe      4084  SYSTEM     00      1,652 K   Window:
rdpclip.exe       4072  Low        00      1,324 K   RDP Clip
explorer.exe      3916  Low        00     18,976 K   Window:
UdaterUI.exe      3604  Low        00        932 K   Commor
taskeng.exe       3512  Low        00      2,300 K   Task Sch
dwm.exe           3348  Low        00        616 K   Desktop
taskeng.exe       3328  SYSTEM     00        704 K   Task Sch
CpuControlAg...   2976  Low        00      4,240 K   Manage
McTray.exe        2912  Low        00        656 K   McAfee
ConsoleApplic...  2836  Low        00      2,816 K   vshost.e
MSASCui.exe       2624  Low        00      4,124 K   Window:
mfeann.exe        2364  SYSTEM     00        652 K   VSCore .
Mcshield.exe      2076  SYSTEM     00     35,316 K   On-Acce
naPrdMgr.exe      2072  SYSTEM     00        712 K   NAI Proc

Processes: 54    CPU Usage: 100%    Physical Memory: 54%
```

Fig. 2. Original CPU utilization of web service task

Figure 2 showed the standard CPU utilization of a web service task, w3wp.exe that runs in Microsoft Windows 7 environment. The task w3wp.exe serves as the candidate task for our experiment. As we can see from the above figure, it is running on 97% CPU utilization with around 10,384 kilobyte of memory consumption. The overall CPU usage reaches 100%. This indicates that the web service task is consuming almost all of the available CPU power and forcing all local tasks to run with bare minimum resources.

The proposed CPU utilization control prototype was then being alerted of the situation and attempted to mitigate the problem. The prototype tried to decrease the CPU utilization of that web service task from 97% to 20%. The control program limited the CPU utilization by forcing the targeted process to periodically sleep for a very short time. Figure 3 shows the CPU utilization of Web Service task after adjusted to CPU 20%.


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Show processes from all users
End Process

```
Image Name        PID   User Name  CPU  Memory (...  Descript
System Idle P...     0  SYSTEM     73         12 K   Percent
w3wp.exe          3560  NETWO...   22      9,552 K   IIS Work
taskmgr.exe       2856  Low        03      1,960 K   Window:
explorer.exe      3916  Low        02     19,784 K   Window:
winlogon.exe      4084  SYSTEM     00      1,652 K   Window:
rdpclip.exe       4072  Low        00      1,504 K   RDP Clip
ConsoleApplic...  3944  Low        00      4,488 K   Console.
ConsoleApplic...  3820  Low        00      2,620 K   vshost.e
WmiPrvSE.exe      3700  SYSTEM     00      1,316 K   WMI Prc
UdaterUI.exe      3604  Low        00        940 K   Commor
taskeng.exe       3512  Low        00      2,300 K   Task Sch
dwm.exe           3348  Low        00        764 K   Desktop
taskeng.exe       3328  SYSTEM     00        704 K   Task Sch
cmd.exe           3096  Low        00        412 K   Window:
CpuControlAg...   2976  Low        00      4,360 K   Manage
McTray.exe        2912  Low        00        656 K   McAfee

Processes: 58    CPU Usage: 27%    Physical Memory: 55%
```

Fig. 3. CPU utilization of web service task after controlled

Note that the CPU utilization control works in a dynamic way such that it controls the utilization based on the current behaviours of the task. The controlling of task utilization is done gradually from time to time in order to avoid abrupt change in CPU usage. This is because a sudden change in CPU usage will occasionally freeze the computer. Table 1 showed the summary of the CPU utilization of the task within the first 120 second.

TABLE 1
CONTROLLING UTILIZATION OF WEB SERVICE TASK TO CPU 20% BETWEEN 60S-120S

Time (s)	CPU Utilization (%) (Web Service)	Total CPU Utilization (%)
10	97	100
20	93	100
30	94	100
40	98	100
50	98	100
60	97	100
70	25	30
80	20	25
90	22	27
100	22	28
110	18	25
120	22	25

In the first 60 seconds, the web service was running on a compute intensive task and starts to use excessive processor resources (i.e. CPU 93-97%). The control program sensed the abnormal behaviour and started the CPU utilization control mechanism at 60 second. The prototype dynamically hunted down the offending thread, and clamped the CPU usage of that task. The CPU utilization of the web service task was being controlled between 18-25% at the period of 60s to 120s. The positive result showed that the control program was able to limit the CPU utilization of the tested task.

5 CONCLUTION

This research proposes a DCUC Controller which is able to use in dynamic CPU utilization control (DCUC) mechanism. Dynamic CPU utilization control (DCUC) mechanism is proposed to maximize the resources utilization but shorten the waiting time between the dependent tasks. This mechanism enables the estimation on resource capability in real-time and controls the total execution time needed by both local and remote tasks. It is achieved by reducing the CPU usage that assigned to the remote tasks. The freed CPU resources can be reallocated to other local tasks. As a result, it provides better resource utilization when the makespan of local tasks is reduced.

ACKNOWLEDGMENT

I would like to express my appreciation to my supervisor, Dr. Ang Tan Fong for his time, guidance, patience, understanding and valuable suggestions throughout the course of my research. I am grateful to all my friends from Network Research Lab, most importantly to Jen Funn, Ahmad Riza, Tiong Seng, Kar Guen, and Chun Yong during the 3 years of my study in University of Malaya. Last but not least, I would like to thank my family for their support. Without their support, I do not think I can make it.

REFERENCES

[1] Al Rawi, A.F. (2011). User priority aware scheduling and dynamic resource allocation in orthogonal frequency division multiple access. *Communications, IET, IEEE, 5(7),* 1006-1019.

[2] Zhou, Tianran. (2011). Hierarchical resource allocation for integrated modular avionics systems. *Systems Engineering and Electronics, IEEE,* 22(5), 780-787.

[3] Endo, P.T. (2011). Resource allocation for distributed cloud: concepts and research challenges. *Network, IEEE,* 25(4), 42-46.

[4] Ykman-Couvreur, C. (2011). Policy-Based Scheduling and Resource Allocation for Multimedia Communication on Grid Computing Environment. *Systems Journal, IEEE, 5(4),* 451-459.

[5] Wei Guo. (2008). Distributed Computing over Optical Networks. Paper presented at the Optical Fiber communication/National Fiber Optic Engineers Conference, 2008. OFC/NFOEC 2008.

[6] Windows Task Manager, Available from http://support.microsoft.com/kb/323527 [Accesed from November 2011]

[7] Jang Uk In. (2011). Linking run-time resource management of embedded multi-core platforms with automated design-time exploration. *Computers & Digital Techniques, IET, IEEE,* 5(2), 123-135.

[8] SpeedswitchXP - CPU frequency control for notebooks running Windows XP. (2012), Available from: http://www.diefer.de/speedswitchxp/ [Accesed from November 2011]

[9] ThreadMaster, Available from: http://threadmaster.tripod.com/ [Accesed from September 2011].

[10] Battle Encoder Shirase, Available from: http://mion.faireal.net/BES/ [Accesed from September 2011].