Resource Isolation Method for Program’s Performance on CMP

Ti Guan\textsuperscript{1a}, Chunxiu Liu\textsuperscript{2b}, Zheng Xu\textsuperscript{1c}, Huicong Li\textsuperscript{1d} and Qiang Ma\textsuperscript{1e}

\textsuperscript{1}State Grid Shandong Electric Power Company, Jinan 250001, Shandong Province, China
\textsuperscript{2}State Grid Shandong DeZhou Electric Power Company, Dezhou 253008, Shandong Province
Email: \textsuperscript{a}sddl_zdh@163.com, \textsuperscript{b}1957626310@qq.com, \textsuperscript{c}xuzheng7788@126.com, \textsuperscript{d}lhc_111@163.com, \textsuperscript{e}xtusys@126.com

Abstract. Data center and cloud computing are more popular, which make more benefits for customers and the providers. However, in data center or clusters, commonly there is more than one program running on one server, but programs may interfere with each other. The interference may take a little effect, however, the interference may cause serious drop down of performance. In order to avoid the performance interference problem, the mechanism of isolate resource for different programs is a better choice. In this paper we propose a light cost resource isolation method to improve program’s performance. This method uses Cgroups to set the dedicated CPU and memory resource for a program, aiming at to guarantee the program’s performance. There are three engines to realize this method: Program Monitor Engine top program’s resource usage of CPU and memory and transfer the information to Resource Assignment Engine; Resource Assignment Engine calculates the size of CPU and memory resource should be applied for the program; Cgroups Control Engine divide resource by Linux tool Cgroups, and drag program in control group for execution. The experiment result show that making use of the resource isolation method proposed by our paper, program’s performance can be improved.

1. Introduction

In recent years, data center and cloud computing are more popular, which make more benefits for customers and the providers. However, in data centre or clusters, commonly there is more than one program running on one server, but programs may interference with each other. The interference may take a little effect, however, the interference may cause serious drop down of performance. For programs running in cluster or data centers, serious performance dropdown is not allowed. In order to avoid the performance interference problem, the mechanism of isolate resource for different programs is a better choice.

The common resource isolation method is virtual machine and container. But both the methods are all have shortcomings. For example, the virtual machine often takes a high cost, which consumes a large amount of resource of maintenance the virtual machine. At the same time, the container consumes little resource for resource isolation, but if it running the program processing the database, the database is too large for the container, and can not be used for other programs.

In this paper we propose a light cost resource isolation method to improve program’s performance. This method uses Cgroups to set the dedicated CPU and memory resource for a program, aiming at to guarantee the program’s performance. There are three engines to realize this method: 1) Program Monitor Engine top program’s resource usage of CPU and memory and transfer the information to
Resource Assignment Engine; 2) Resource Assignment Engine calculates the size of CPU and memory resource should be applied for the program; 3) Cgroups Control Engine divide resource by Linux tool Cgroups, and drag program in control group for execution.

The experiment result shows that making use of the resource isolation method proposed by our paper, program’s performance can improve 8.65%. The average CPU utilization improves 24.5%, and the average memory utilization improves 21%.

2. Related work
There are many methods for resource isolation of programs.

The first one is the virtual machine. Virtual machine is a fully integrated computer system with a complete hardware system functioning in a fully isolated environment [1, 2, 3]. Virtual machine can be used for workload consolidation and share hosting. The program can be running in virtual machine with an un-interference execution environment. But the shortcoming is the low performance because of the high cost.

The second method is the container. Container is used for consistent development and deployment. For example, Docker and Linux container. Docker [4, 5] is the lightweight container to isolate the programs. It has the advantage of the lower cost, but the shortcoming is the data base commonly not in the container. Similar with Docker, the Linux container also has these characteristic.

Cgroups [6] is a Linux resource isolation tools for resource management. Commonly realize resource isolation have to careful allocation and scheduling of physical resources, and Cgroups can realize the function. It can be used easily, and set a control groups for program. For each group, Cgroups can set the size of resource according the operator. Then we can drag the program to the control groups for execution [7].

3. Resource isolation method
Linux resource isolate tool Cgroups is used in our paper for resource isolation. Cgroups is based on the Linux kernel and can be set flexibly. In order to improve the program’s performance, the following module should be commit, and the flow chart is in Figure 1.

![Figure 1](image_url)

**Figure 1** The flow chart of resource isolation method.
Module 1: Program Monitor Engine. This module collects the program’s resource usage information about CPU and memory. The general performance profiling tools can be used to understand the program. In our paper we use Linux tool Top. We top program’s resource usage of CPU and memory and transfer the information to Module 2.

Module 2: Resource Assignment Engine. The input of the Resource Assignment Engine is the information passed from Module 1, and the output of the Resource Assignment Engine is the size of CPU and memory resource should be applied for the program. The Resource Assignment Engine is used to calculate the size of CPU and memory resource assigning to programs. The calculate method can be found in next section. Then transfer the information to Module 3. This part is very important for our paper. Because the resource assigning to programs have directly effect to the program’s performance. If the resource assigning to programs is too large, it is a waste of resource. And if the resource assigning to programs is too small, the program’s performance will not good. So the calculate method for assignment resource is the key point.

Module 3: Cgroups Control Engine. The input of the Cgroups Control Engine is the information passed from Module 2, and the output of the Cgroups Control Engine is to set the CPU and memory resource. This part is also important for our paper. The Cgroups Control Engine set the size of resource correctly according to the information passed from module 2. This model divide resource by Linux tool Cgroups, and drag program in control group for execution.

4. Implementation
This section explains the method of calculate the size of CPU and memory resource in detail. In Resource Assignment Engine, the input is the information passed from Program Monitor Engine, and the output of the Resource Assignment Engine is the size of CPU and memory resource should be applied for the program.

Assume that there are n programs have to scheduled to one server. There are p1, p2, ..., pn respectively. The CPU resource usage of these program when running alone are Cp1, Cp2, ..., Cpn, respectively, and the memory resource usage of these program when running alone are Mp1, Mp2, ..., Mpn, respectively.

Then the CPU resource calculated method should obey formula (1).

\[
C_{pi-schedule} = \frac{C_{pi}}{C_{p1} + C_{p2} + \ldots + C_{pn}} \cdot C_{server}
\]

In the formula, pi is the number of the program, Cpi is the resource usage of program pi when it running alone on a server. Cserver is the total CPU resource of the server. Cpi-schedule is the size of the CPU resource when program pi is scheduled along with other programs.

The memory resource calculated method is different with CPU resource. It is because of our observation in experiment. Commonly when program’s memory resource occupies below program’s memory resource when running alone, program’s performance would drop down dramatically. Based on the belief, the number of program can be scheduled onto one server should obey follow algorithm.

| Pseudo-code of calculating the number of program running on one server |
|---|
| 1: mem.sum=mem[p1] |
| 2: for i = p1 to pn do |
| 3: mem.sum=mem.sum+mem[i+1]; |
| 4: if mem.sum >mem.usage |
| 5: |
| 6: num.pro=i |
| 7: break; |
| 8: } |
| 9: end for |

Figure 2 Pseudo-code of calculating the number of program running on one server
In Figure 2, we sum the program’s usage of memory from p1 until the mem_sum more than the memory size of server. Then we can make sure the number of the program running on one server, which is pi. Then the programs which can not schedule on this server will be scheduled onto the next server.

After fix the number of program, the memory resource calculated method should obey formula (2).

\[
M_{\text{pi-schedule}} = \frac{M_{\text{pi}}}{M_{\text{pi}} + M_{\text{p2}} + \ldots + M_{\text{pn}}} M_{\text{server}}
\]   

(2)

In the formula, pi is the number of the program, Mpi is the memory resource usage of program pi when it running alone on a server. Mserver is the total memory resource of the server. Mpi-schedule is the size of the memory resource when program pi is scheduled along with other programs.

By the algorithm above, the size of CPU and memory resource should be applied for the program can be calculated reasonably.

5. Experiment
We conduct our experiments on a NUMA server, the server with two sockets and 4 cores on each socket. The detail server information is in TABLE 1. We select programs from NAS Parallel Benchmarks. NAS Parallel Benchmarks is a benchmark suite composed of multithreaded programs. The programs as workloads are in TABLE 2. The CPU and memory event collected by Top are listed in TABLE 3.

We compare three strategies. The first strategy is program running alone on a server. The second strategy is program co-running with other programs. The third strategy is program co-running with other programs, meanwhile with the resource restriction under Cgroups.

Figure 3 shows the program’s performance with different strategy, and the baseline is the program’s performance when program running alone on a server. There are two workloads.

| Table 1 Server configuration |
|-----------------------------|
| **CPU** | **Intel Xeon E5620** |
| core | 4 cores@2.13G |
| L1 caches | 32K |
| L2 caches | 256K |
| L3 caches | 4M |
| Threads per core | 1 thread |
| Sockets | 2 |
| Memory | 8GB, DDR3 |

| Table 2 Workloads |
|-------------------|
| **Workloads** | **program** |
| Workload 1 | ft.C.4, mg.C.8 |
| Workload 2 | cg.C.4, mg.C.8 |

| Table 3 Collected events. |
|---------------------------|
| **Item** | **events** |
| CPU | %CPU |
| memory | %MEM |

The first workload is ft.C.4 and mg.C.8, and the second workload is cg.C.4 and mg.C.8. For workload 1 it is shows by the second strategy (program co-running with other programs) almost as 0, it
is because program ft.C.4 need almost all of the memory, when ft.C.4 co-running with other programs who use memory, ft.C.4 performance would drop down dramatically. Meanwhile, the co-running program’s performance would drop down dramatically too. However, with the third strategy (program co-running with other programs, meanwhile with the resource restriction under Cgroups), because the action of the program monitor engine and resource management engine, the two program’s performance are predicted dramatically drop down. Then the resource management engine schedules the two programs to different two servers to avoid performance’s serious download. Which is smart than strategy 2.

For workload 2, compare the second strategy, when using the third strategy, the performance of program cg.C.4 improves 20%, and the performance of program mg.C.8 decline 2.7%. And the average performance improves 8.65%. This experiment proves our resource isolation method is good for program’s performance.

Figure 4 shows the CPU utilization of the two workload for the three strategy. The CPU utilization when program is running alone is the baseline. It is shows that for workload 1, by the third strategy, the CPU utilization of are always better than the second strategy, and the average CPU utilization improve 24.5%. For workload 2, by the third strategy, the average CPU utilization improves 24.5%.
Figure 5 shows the memory utilization of the two workload for the three strategy. The memory utilization when program is running alone is the baseline. It is shows that for workload 1, when use the second strategy, the memory utilization drops down dramatically, resulting the performance’s dramatically dropdown. And we get the conclusion that the lack of memory resource would kill the program. For workload 2, compared with the second strategy, by the third strategy the average memory utilization improves 21%.

6. Conclusion
In this paper we propose a light cost resource isolation method to improve program’s performance. This method use cgroups to set the dedicated CPU and memory resource for program, aiming at to guarantee the program’s performance.

There are three engines to realize this method: 1) Program Monitor Engine top program’s resource usage of CPU and memory and transfer the information to Resource Assignment Engine; 2) Resource Assignment Engine calculates the size of CPU and memory resource should be applied for the program; 3) Cgroups Control Engine divide resource by Linux tool Cgroups, and drag program in control group for execution.

The experiment result shows that making use of the resource isolation method proposed by our paper, program’s performance can improve 8.65%. The average CPU utilization improves 24.5%, and the average memory utilization improves 21%.

Use footnotes sparingly (or not at all) and place them at the bottom of the column on the page on which they are referenced. Use Times 8-point type, single-spaced. To help your readers, avoid using footnotes altogether and include necessary peripheral observations in the text (within parentheses, if you prefer, as in this sentence).

7. References
[1] H. Jin, A. L. Zhong, S. Wu, and X. H. Shi. Virtual Machine VCPU Scheduling in the Multi-core Environment: Issues and Challenges. Journal of Computer Research and Development, 2011, 48(7): 1216-1224.
[2] M. Liu and T. Li. Optimizing virtual machine consolidation performance on numa server architecture for cloud workloads. In ISCA, 2014.
[3] J. Sugerman, G. Venkitachalam, and B. H. Lim. Virtualizing I/O Devices on VMware Workstation’s Hosted Virtual Machine Monitor. Proc. Usenix Annual Technical Conference, June 2001.
[4] C. Boettiger. An introduction to Docker for reproducible research. ACM SIGOPS Operating Systems Review, Special Issue on Repeatability and Sharing of Experimental Artifacts 49(1):71-79.

[5] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. Linux J., 2014(239), March 2014.

[6] Cgroups [EB/OL]. https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

[7] L. Wang, D. Qian, Z. Luan, G. Wei, R. Wang, H. Yang. Speeding up profiling program’s runtime characteristics for workload consolidation. PLoS ONE 12(4): e0175861.