2010 12th IEEE International Conference on High Performance Computing and Communications

# Two Optimization Mechanisms to Improve the Isolation Property of Server Consolidation in Virtualized Multi-Core Server

Kejiang Ye, Xiaohong Jiang, Deshi Ye, Dawei Huang

*College of Computer Science, Zhejiang University*
*Hangzhou 310027, China*
{yekejiang,jiangxh,yedeshi,davidhuang}@zju.edu.cn

***Abstract*—Virtualization brings many benefits such as improving system utilization and reducing cost through server consolidation. However, it also introduces isolation problem when running multiple virtual machine workloads in one physical platform. Additionally, with the advent of multi-core technology, more and more cores are built into one die in today's data center that will share and compete for the resource like cache. It's worthy to study the isolation of server consolidation in modern multi-core platform. However, to our knowledge there are few work done on the isolation property especially the fault isolation property when one of the virtual machine workloads is attacked in server consolidation. In this paper, we study the isolation property from performance perspective and provide two optimization methods to improve the isolation property. We first define the isolation property and quantify the performance isolation in consolidation and propose a VM-level optimization method. Then we study the fault isolation by introducing a misbehavior virtual machine in server consolidation scenario and propose a core-level cache-aware optimization method to improve the fault isolation. Experimental results show that our two optimization methods can effectively improve the performance isolation and fault isolation with 29.39% and 19.52% respectively. What's more, Oprofile/Xenoprof toolkits are used to find out the factors affecting isolation property from the hardware events level.**

***Keywords*-Cache-aware; Fault Isolation; Multi-core; Server Consolidation; Virtualization**

## I. Introduction

Virtualization technology [1, 5, 19, 20] has brought a lot of benefits such as improving resource utilization, reducing costs, easing management of computer servers and consolidating multiple server workloads into a single physical platform which named server consolidation. Server consolidation [3, 4, 13] is one of the most common scenarios of virtualization, which enables to consolidate multiple server workloads into a single physical platform to maximize the system utilization. Although server consolidation offers great potential to improve resource utilization and reduce cost, it may also introduce new challenges in managing the consolidated servers such as isolation.

With development of multi-core technology, consolidation of multiple workloads will incur resource competition issues such as core competition and L2 cache competition. What's more, the scheduling of virtual machines across the physical cores is also a notable issue. In this paper, we will perform a detailed performance study about the isolation property in server consolidation scenario on the multi-core platform, based on which we propose two optimization mechanisms to improve the isolation property. Fig. 1 shows a typical server consolidation scenario in a modern multi-core platform.

We divide the isolation property into **performance isolation** and **fault isolation**. **Performance isolation** indicates the performance effects when consolidating several workloads into one physical servers, and **fault isolation** indicates the performance effects when there exist a misbehavior workload which will affect other workloads. Our goal is to quantify the performance isolation and fault isolation of server consolidation and identify the factors that affect the isolation, and then improve the isolation property using efficient optimization methods based on the our experimental discovery. Fig. 2 shows a consolidation optimization mechanism to improve the performance isolation. The resource characterization analysis can be done through various methods, such as workloads profiling analysis or tool analysis like Xentop.

The main contributions of the paper are summarized as follows: $(i)$ Performance isolation. To explore the isolation under different consolidation scenarios, we formally define the concept of performance isolation and design a variety of experiments using typical workloads in data center to verify its correctness. After the evaluation, one VM-level optimization mechanism is proposed to improve the performance isolation. $(ii)$ Fault isolation. We also define the concept of fault isolation and try to evaluate it quantitatively by introducing a misbehavior virtual machine with a fork bomb. Based on the experimental discovery, we propose a cache-aware core scheduling mechanism to improve the fault isolation. $(iii)$ Profiling analysis. We use the Oprofile/Xenoprof toolkit to profile the hardware events (CPU cycle, L2 cache misses, etc.), helping explain how the multi-core architecture features influence the isolation.

The rest of the paper is organized as follows. In Section II, we introduce the background of server consolidation and the motivation to study the isolation property in multi-core platform. Section III describes our evaluation methodology to address the issues of isolation evaluation. In Section IV we

978-0-7695-4214-0/10 $26.00 © 2010 IEEE
DOI 10.1109/HPCC.2010.95

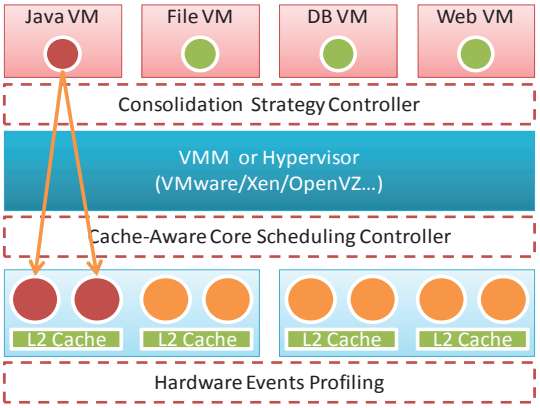


Figure 1. A Typical Server Consolidation Scenario on Multi-core Platform.

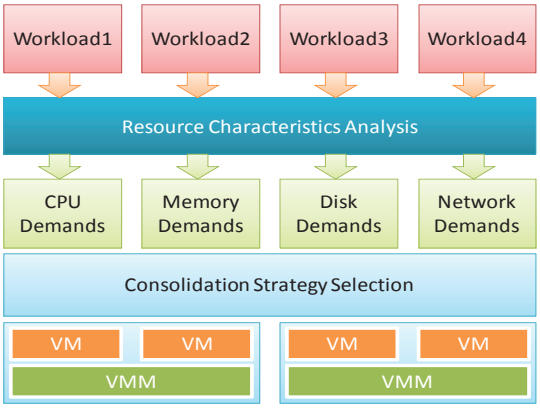


Figure 2. The VM-Level Consolidation Optimization Mechanism.

perform a variety of experiments to investigate how different consolidation strategies influence the performance isolation and propose a VM-level optimization mechanism. While in Section V, we investigate the fault isolation by introducing a misbehavior virtual machine running a fork bomb and propose a cache-aware core scheduling optimization mechanism. We discuss related work in Section VI. Finally, we summarizes our conclusion and future work in Section VII.

## II. Background & Motivation

The virtualization technology promises to give a better isolation property than the scenario that running multiple applications in one physical platform by multi-thread method. However, consolidating several workloads together leads to a certain degree of performance degradation. It is necessary to carefully study the isolation property in server consolidation scenario. Especially, when one of the consolidated workloads encounters a failure such as continuously consuming large amounts of resources, how will the other workloads be affected? We divided the isolation property into two types: performance isolation and fault isolation. We give a formal definition of performance isolation and fault isolation in the next section.

To the best of our knowledge, most of the previous studies focus on the performance degradation of server consolidation in real physical machine [4] or perform their experiments in the simulator platform to investigate multi-core effects on isolation property [10]. However, there are few studies focus on the isolation property of server consolidation in real multi-core platform, especially the fault isolation property. It is essential to quantify the isolation effects and identify the potential factors that may affect the isolation and then propose optimization methods.

The requirements of isolation of server consolidation are summarized as follows: $(i)$ What's is the isolation? How can we quantify it? How much performance isolation and fault isolation can be achieved in server consolidation? $(ii)$ What factors can affect the isolation property in modern multi-core platform? Is the shared resource contention issues (such as core, L2 cache contention) be a key factor? How will the scheduling strategy affect the isolation? $(iii)$ How can we optimize it?

## III. Experimentation Methodology for Isolation Property

### A. Workloads and Benchmarks Selection

We choose four typical server workloads in modern data center running in virtual machine in our experiments. They are java server, file server, database server, and web server. We choose SPECjbb2005 [2] as the java server benchmark, IOzone [17] as the file server benchmark, Sysbench [12] as the database benchmark and Webbench as the web server benchmark. The database used in the experiment is Mysql and web server used is Apache. Fig. 1 gives an example of scheduling scenario of server consolidation in 8-core platform, in which the java server is fixed to dedicated cores by varied scheduling strategies. The java virtual machine in red means it is attacked by the bomb.

In order to quantify the isolation property, we present the formal definition of isolation: Let $W = \{W_1, W_2, ..., W_n\}$ be the set of workloads, where $W_i$ represents the workload of $VM_i$. Denote by $P = \{P_1, P_2, ..., P_n\}$ the performance metrics of workload $W$. We use the following equation to quantify the performance isolation:

$$I_{perf} = \sum_{i=1}^{n} w_i (P_i^{Con} / P_i^{Ind}) \quad (1)$$

and fault isolation:

$$I_{fault} = \sum_{i=1}^{n} w_i (P_i^{Fault} / P_i^{Normal}) \quad (2)$$

In the performance isolation definition, $P_i^{Con}$ represents the performance of $W_i$ when running in consolidated mode, while $P_i^{Ind}$ represents the performance when running in individual mode. In the fault isolation definition, $P_i^{Fault}$

indicates the performance running with a misbehavior virtual machine, while $P_i^{Normal}$ indicates the performance without misbehavior virtual machine. Let $w_i$ be the weight of workload performance which reflects the importance of workloads in data center, and $\sum_{i=1}^{n} w_i = 1$. In this paper we set all the workloads have the same weight $w_i = 1/n$ which indicates all the workloads have equal importance. Obviously, we can conclude $I \in (0, 1]$.

All experimental evaluations are performed on the Dell 2900 PowerEdge server with two Quad-core 64-bit Xeon processors at 2 GHz, with 6GB physical memory and 6MB second level cache. We use Ubuntu 8.10 with kernel version 2.6.27 in domain0, and the version of Xen hypervisor is 3.3.1, which has built-in support for Oprofile. All the virtual machines are configured with 4 vCPU and 1 GB memory size.

For each experiment, we also provide a detailed analysis of the corresponding Oprofile/Xenoprof statistical data and try to present the causes for the observed isolation effects. Three hardware counters are recorded for our isolation analysis, they are CPU_CLK_UNHALT (The number of cycles outside halt state is used to estimate the CPU time), INST_RETIRED (The number of retired instructions is a estimate of the number of instructions executed), and LLC_MISSES (The number of last level cache misses measures the number of times the memory references in an instruction miss the last level cache and access main memory). Then we compute the CPI and MPI using the formulas: $CPI = C_{num}/I_{Retired}$, $MPI = M_{num}/I_{Retired}$, where $C_{num}$ represents the number of CPU cycles, $M_{num}$ represents the number of cache misses, $I_{Retired}$ represents the number of retired instructions.

## IV. Performance Isolation Characterization and Optimization

### A. Workloads Characterization Analysis in Server Consolidation

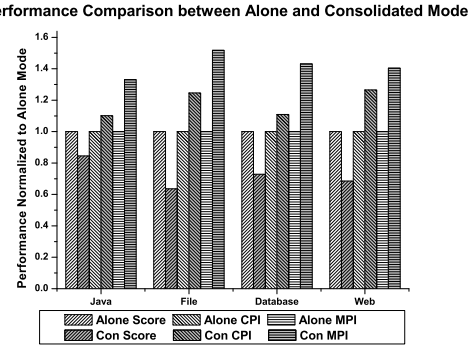


Figure 3. Performance Comparison between Alone and Consolidated Mode.

To understand the performance interference of each workload in server consolidation which will affect the performance isolation property, we have compared the performance running alone and in consolidated mode. As expected, the performance of each workload decreases due to the interference from other workloads within consolidation scenario. Fig. 3 shows the performance of the workloads running in alone and consolidated mode. We normalized the result to a fraction of the alone mode. We observe that compared with individual mode, the performance of each workload has a degradation in some extend in consolidated mode, while has a raise in CPI (Cycle per Instruction) and MPI (L2 Cache Miss per Instruction). Java server loses 15.5% performance, while has 10.20% increasing in CPI and 33.10% increasing in MPI. File server loses 36.36% performance, and has 24.57% increasing in CPI and 51.80% in MPI. Database server loses 27.17% performance, and has 10.93% increasing in CPI and 43.13% increasing in MPI. Web server loses 31.48% performance, has 26.59% increasing in CPI and 40.38% increasing in MPI. The reason for this loss in performance is due to the shared resource contention such as core, cache, memory, etc. The increase of CPI and MPI reflects performance degradation, the higher MPI means higer L2 cache misses rate and affects the performance.

With this experiment, we can compute the performance isolation of consolidating four servers according to the equation (1): $I_{perf} = \sum_{i=1}^{n} w_i(P_i^{Con}/P_i^{Ind}) = (P_{Java}^{Con}/P_{Java}^{Ind} + P_{File}^{Con}/P_{File}^{Ind} + P_{DB}^{Con}/P_{DB}^{Ind} + P_{Web}^{Con}/P_{Web}^{Ind})/4 = (0.84502 + 0.63641 + 0.72835 + 0.68518)/4 = 0.72374$, where $P_{Java}$, $P_{File}$, $P_{DB}$ and $P_{Web}$ represent the performance of java server, file server, database server and web server respectively. It is obvious that the performance isolation in server consolidation is less than 1 which means that there is isolation degradation in when consolidating multiple workloads into one physical machine.

### B. Performance Isolation with Various Consolidation Strategies

Next, we will investigate the performance effects and performance isolation when consolidating every two workloads together, which is helpful to understand which two workloads are suitable to consolidate together, and will give guidance to data center administrator to consolidate appropriate workloads together to achieve the best performance.

Fig. 4 shows the result of performance effects when consolidating every two workloads together. We can draw four conclusions from the figures presented above: (1) Consolidating with other workloads will lead to different degree of performance loss. (2) It is not suitable to consolidate two same workloads in a single platform because of the contention of the same kind resource. Take java server as an example which is CPU intensive, when consolidate two java server together, there will be a heavy pressure of CPU resource, while bringing little demands on other resource

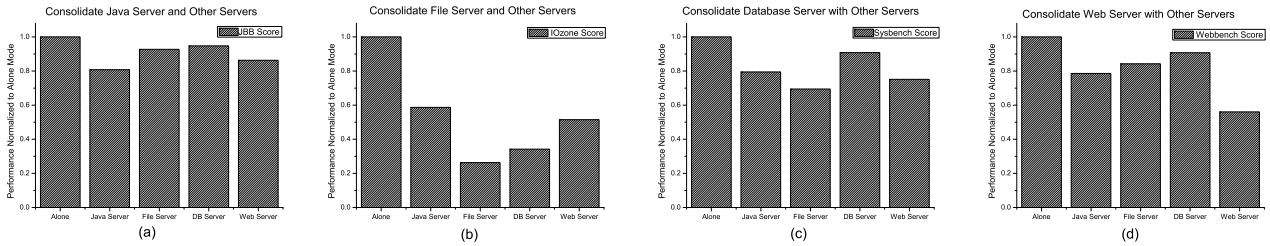


Figure 4. Performance Isolation with Various Consolidation Strategies: (a) Consolidate Java with Other Servers; (b) Consolidate File with Other Servers; (c) Consolidate DB with Other Servers; (d) Consolidate Web with Other Servers

like network bandwidth. (3) Java server and database are most friendly with each other, leading least performance loss, while file server and database are least suitable to consolidate together. It is because that java server consume a lot of CPU resource while database consumes a lot of I/O resource and consumes little CPU, the consolidation of the two workloads will make the best use of the different aspects of system resources. On other hand, file server and database server are both I/O intensive, when consolidating the two workloads, I/O becomes a performance bottleneck. (4) The file server is the most vulnerable to other workloads which is consistent with the changes of CPU utilization of file server in the previous experiment. So it's better to running file server in a stand-alone server.

### *C. Optimization of Performance Isolation with Suitable Consolidation Strategies*

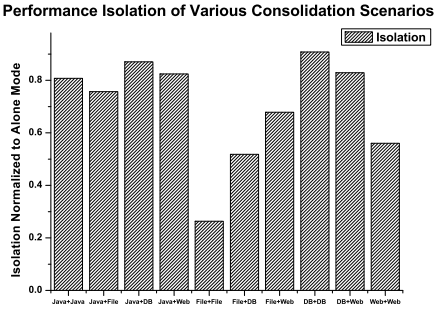


Figure 5. Quantify the Performance Isolation under Various Consolidation Scenarios.

Based on the above workloads characterization analysis and consolidation strategies analysis, we can make suitable consolidation strategy to improve the performance isolation.

Fig. 5 shows the performance isolations with all consolidation scenarios. The consolidation of database server and database server achieves best performance isolation since database server consumes CPU, memory and I/O resource instead of single resource and the performance isolation is 0.91. While consolidation of file server and file server results in heavy contention for disk I/O and achieves the worst performance isolation with 0.26. The average performance isolation of all the consolidation scenarios is 0.70. Obviously, our proposed VM-level consolidation optimization mechanism can achieve 29.39% performance isolation improvement compared with average performance isolation.

## V. Fault Isolation and Cache-Aware Core Scheduling Optimization

In this section, we will study the fault isolation and quantify it. It is very necessary to isolate misbehavior virtual machine and offer a secure virtualization environment. We write a fork bomb program in C to simulate the misbehavior virtual machine and perform several core scheduling experiments to study the cache effects on the fault isolation. Further more, one cache-aware core scheduling optimization mechanism is presented to improve the fault isolation. And also, we will investigate CPI and MPI which are gained by the hardware events profiling to analyze the potential architecture features affecting the fault isolation property.

We have designed three sets of experiments. Firstly, we study the case of consolidating two same workloads into a physical machine with both controlling the scheduling of the two workloads (one is normal, the other is subjected by the bomb). Due to the limited space, we take java server as an example in this paper. Secondly, we study the case of four virtual machines with the same workload (also take java server as an example) consolidated into a physical machine. At this time, we only control the misbehavior virtual machine scheduling to the dedicated cores while leave the other three virtual machine floating. Finally, we present a typical consolidation scenario in real data center that consolidating four typical servers and study the fault isolation. Through these experiments, we will investigate how the normal virtual machines can be affected by the misbehavior virtual machine under different core scheduling strategies. Based on experimental discovery, we propose a cache-aware core scheduling algorithm to improve the fault isolation.

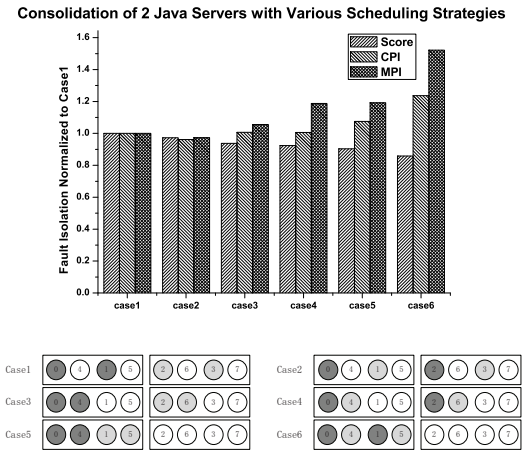


Figure 6. The Fault Isolation of Consolidating 2 Java Servers with Various Scheduling Strategies.

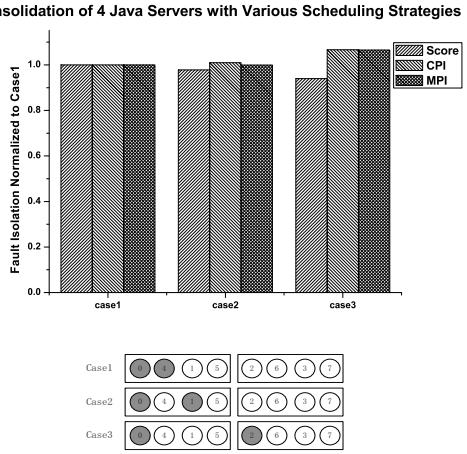


Figure 7. The Fault Isolation of Consolidating 4 Java Servers with Various Scheduling Strategies.

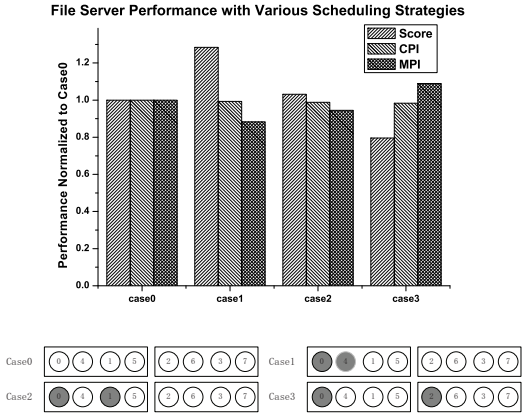


Figure 8. The Performance Effects of File Server when Consolidated with other Servers with Various Scheduling Strategies.

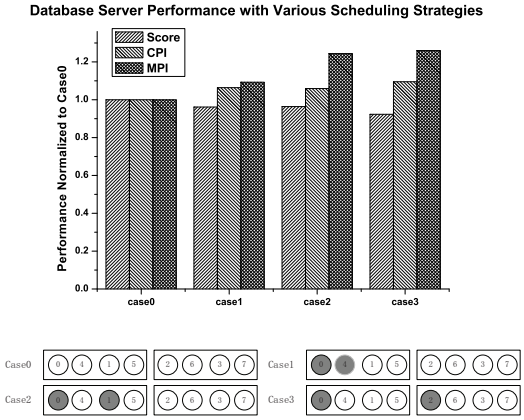


Figure 9. The Performance Effects of Database Server when Consolidated with other Servers with Various Scheduling Strategies.

### A. *Fault Isolation with Same Workloads*

Fig. 6 shows the fault isolation property of two java servers under six different scheduling cases. The cores belonging to the virtual machine which suffers bomb attack are marked in dark gray, while the cores belonging to the other normal virtual machine are marked in light gray.

In case 1, the two vCPUs of the misbehavior virtual machine are pined to two physical cores (in dark gray) in socket0 with no shared L2 cache, and vCPUs belonging to the normal virtual machine are pined in socket1. This case obtains a overall best fault isolation since maximize the utilization of cache resource for java server and the cache pollution in socket0 has little impact on the cache resource in socket1. While in case 2, the misbehavior cores and normal cores are mixed in the physical CPU (See Fig. 6), and achieve less isolation than case 1 because the misbehavior cores will affect the normal cores through L3 cache in the same socket.

In case 3 and case 4, the cores only use half the L2 cache resource in both sockets and can not achieve the performance as case 1 and case 2 apparently. The performance of case 4 is worse than case 3 is due to the shared L2 cache resource is polluted by the misbehavior virtual machine and causes a direct impact on the normal virtual machine.

While case 5 and case 6 only use one socket and get worse performance. Similarly, case 6 will obtain worse performance than case 5 due to the more serious cache pollution.

We can conclude that the SPEC JBB is cache sensitive, more cache resource and better performance, and obviously it's better to affintize the vCPUs to the cores that having the same shared L2 cache. The shared cache resource will improve the performance of SPEC JBB. What's more, we note that it is better to affintize the misbehavior virtual machine into the cores away from the normal virtual machine, ensuring the isolation in the core level. Further, the CPI and

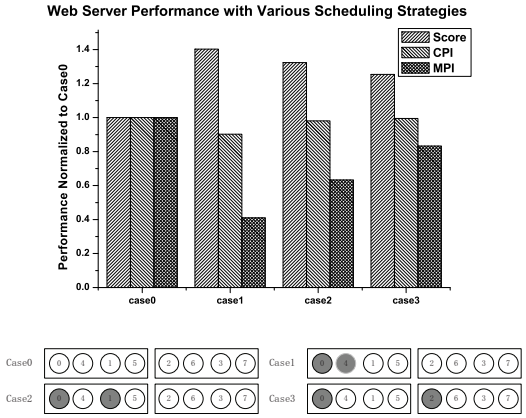


Figure 10. The Performance Effects of Web Server when Consolidated with other Servers with Various Scheduling Strategies.

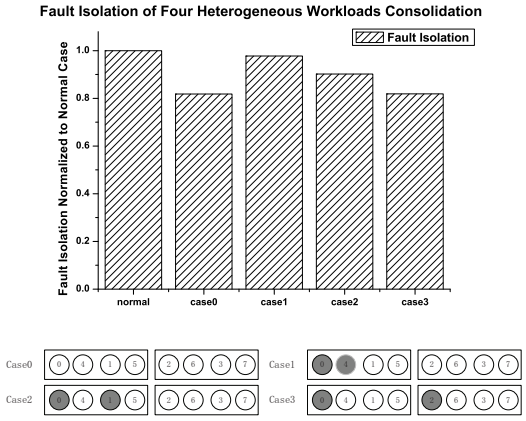


Figure 11. The Fault Isolation in the Consolidation of Four Heterogeneous Workloads under Various Scheduling Strategies.

MPI trend indicates the contention of L2 cache resource by misbehavior cores and normal cores will lead to visible L2 cache miss and also lead an increase on CPI. It once again proved that if we don't isolate the cache resource in the physical core level, the cache resource can be polluted by the misbehavior virtual machine quickly and leads to degradation of fault isolation.

Fig. 7 presents the result of consolidation with four same workloads with a misbehavior virtual machine. In this experiment we investigate the fault isolation of consolidating four same workloads, and only affintize the vCPUs belonging to misbehavior virtual machine to dedicated physical cores while leave the vCPUs of other virtual machines floating.

We take java server as an example. In case 1, the vCPUs belonging to the bomb virtual machine are affintized to the two physical cores that share the same L2 cache. In case 2, we affintize the vCPUs to the cores that have no cache sharing but in the same socket. And in case 3, we affintize the vCPUs to the different sockets. From the figure, we see that case 1 obtains the worst performance, case 2 the next, and case 3 the best performance.

From the CPI and MPI analysis, we find that MPI increase directly leads the performance degradation in case 3. The cache resource in case 3 is worst polluted, therefore when other three java servers are scheduled across the physical cores the L2 cache misses will be the main bottleneck and then lead to degradation of fault isolation.

### *B. Fault Isolation with Four Heterogeneous Workloads*

In this section, we will study a more realistic scenario in data center that consolidating four different typical workloads. The scheduling strategies are the same with the above experiments. Fig. 8 to 10 show the performance changes of file server, database server and web server, while the java server is attacked by the bomb and presents no result. From all the three figures, we find that case 1 achieves the best performance, while case 3 achieves the worst performance. In case 1, the two cores that affected by the bomb is limited in one L2 cache and have least impact on other workloads, while case 3 achieve the worst performance due to the heavier cache pollution. Besides, we also present the default scheduling scenario in case 0 as a baseline, and compare the results of other three cases with it to investigate the impact of scheduling strategy on the performance.

From Fig. 8 we find file server achieves better performance than normal scenario (case 0) in case 1 and case 2, while obtains worse performance in case 3 with a degradation of 23.39%. It is because the misbehavior virtual machine affects the two sockets and causes high cache miss rate (The increase of MPI in Fig. 8 can illustrate it). While in Fig. 9, compared with case 0, all the three cases lead to performance degradation with 3.80% in case 1, 3.60% in case 2, 7.67% in case 3. The reason for the slight decline can be explained by the obvious increase of MPI with about 9.28%, 24.18% and 26.02% respectively. Fortunately, we find the web server acquires a obvious increase of throughput compared with case 0. That is to say, the fault isolation in case 0 is not so good, and a huge performance increase will be achieved by using new cache-aware core scheduling strategy instead of using the default credit algorithm in Xen.

### *C. Cache-Aware Core Scheduling Optimization to Improve the Fault Isolation*

Because the current credit scheduler in virtual machine monitor is designed for SMP load balance, and can not schedule the two vCPUs with data sharing to the physical cores that sharing the L2 cache, It is necessary to study the new scheduling strategies according to the workloads characterization.

According to previous experimental results discovery, we find that L2 cache miss rate will affect the performance isolation. What's more, different workloads have different characterization and consume different aspects of system

**Algorithm 1** Cache aware VM-Core scheduling strategy in multi-core platform.

**Input:**

The set of workloads in consolidation, $W$;

The set of performance of individual workload, $P^{Ind.}$;

The sign indicates whether the VM subjects a bomb, $f$;

**Output:**

The isolation of the whole system, $I$;

1: If $f = 1$,
2: Pin the misbehavior VM to the cores that have no cache sharing with other VMs;
3: Else if $W_i$ is cache sensitive,
4: Pin the $W_i$ to the cores that share the same L2 cache;
5: Else,
6: Let the remaining workloads floating across the remaining cores by the default scheduling algorithm;
7: Running the workloads, and get a new performance set $P^{Con.} = \{P_1^{Con.}, P_2^{Con.}, ..., P_n^{Con.}\}$ or $P^{Fault} = \{P_1^{Fault}, P_2^{Fault}, ..., P_n^{Fault}\}$, and calculate performance isolation through equation (III-A) or fault isolation through equation (2)
8: **return** $I$;

resource. Based on these findings, we present a cache aware core scheduling strategy to maximize both fault isolation, and minimize the failure affecting other workloads.

From the Algorithm 1, we first check if there is a misbehavior virtual machine which can be monitored by existing tools like Xentop, if there is a misbehavior virtual machine, we fix the virtual machine in the dedicated physical cores so that the cache belonging to other virtual machines will not be polluted. Then we will schedule the VM to the remaining cores according to the workload characterization.

Fig. 11 shows the fault isolation in all cases. The normal case indicates the normal server consolidation with no misbehavior virtual machine and case 0 indicates the default scheduling algorithm in Xen. We quantify the fault isolation by equation (2). We take case 0 as an example: $I_{fault}^{case0} = \sum_{i=1}^{n} w_i(P_i^{Fault}/P_i^{Normal}) = (P_{File}^{Fault}/P_{File}^{Normal} + P_{DB}^{Fault}/P_{DB}^{Normal} + P_{Web}^{Fault}/P_{Web}^{Normal})/3 = (0.65 + 0.98 + 0.823)/3 = 0.82$, where $I_{fault}^{case0}$ represents the fault isolation in case 0, while $P_{File}$, $P_{DB}$ and $P_{Web}$ represent the performance of file server, database server and web server respectively. It is noteworthy that java server presents no resource due to the bomb effect, so $P_{Java}^{Fault} = 0$ here. It is obvious, our proposed cache-aware core scheduling strategy achieves better fault isolation than Xen's default scheduling algorithm with 19.52% fault isolation improvement for case 1.

### D. Discussion

We also did the bomb experiment under native Linux. Unsurprisingly, the disruptive processes made the machine completely unusable for the other workloads, causing almost all the CPU time to be spent by the abnormal process. While in virtual machine, the bomb only leads the unusable for the virtual machine itself and affects the other virtual machine with a certain degree. It also demonstrates virtualization can ensure a certain degree of isolation.

Through the above experiments, we demonstrate that several factors can affect the performance isolation and fault isolation: 1) Workload types. Through the analysis of workloads characterization, we can ensure the isolation by consolidating suitable workloads. 2) Architecture features of multi-core platform. We find the contention for the shared resource can affect the isolation, especially the core and L2 cache resource. 3) Scheduling strategy. We present a cache-aware core scheduling optimization strategy to improve the fault isolation. Experimental results show that the new scheduling strategy can improve the fault isolation obviously with 19.52% compared with default Xen scheduling experiment.

Experimental results collected from hardware events profiling analysis using Oprofile/Xenoprof indicate that cache contention is a main factor affecting fault isolation which will cause apparent L2 cache misses.

## VI. Related Work

A lot of work has been done on the evaluation of virtualization performance [3, 5–7, 16] or approaches to improve the virtualization performance [11, 15, 18]. However, few work has referred to the isolation property especially not too much work has been done on the fault isolation in virtualization environment.

Govil et al. [8] presented a system called Cellular Disco which exploited the structure of the Origin2000 to increase performance and performance isolation by using NUMA-aware memory allocation to virtual machine. Denali [21] provided scalability as well as isolation for untrusted code, but it didn't provide any specialized mechanisms for performance isolation. Waldspurger [20] considered the problem of allocating memory across virtual machines in VMware ESX server. An idle memory tax was introduced to achieve efficient memory utilization while maintaining performance isolation guarantees. However, they didn't conduct a detailed evaluation of performance isolation property and fault isolation property, and have not considered the factors coming from CMP platform, which would affect isolation property due to core and cache interference when running several workloads simultaneously.

Recently, Marty et al. [13] created a two-level virtual coherence hierarchy at a high level to help manage the cache resources of a many-core CMP when running consolidated workloads. Gupta et al. [9] proposed two mechanisms to improve the performance isolation across virtual machines in Xen, but only focused on the CPU and network resource.

However, they didn't consider the modern CMP architecture characteristics affecting the isolation property. One important work on the evaluating the performance isolation was done by Matthews et al. [14], where they compared the isolation property among Xen, Solaris Containers and OpenVZ. We extended their work to CMP case within server consolidation scenarios, and explored the impact on the isolation.

## VII. Conclusion and Future Work

In this paper we focused on the isolation property of server consolidation on multi-core platform. We proposed the possible factors that would affect the isolation property in multi-core platform and then performed detailed experiments to quantify the isolation and verify the factors leading to affecting isolation property. We not only quantify the performance isolation, but also the fault isolation. What's more, we present a VM-level optimization mechanism and a cache-aware core scheduling strategy to improve the isolation.

Our experiments show that: $(i)$ the isolation property closely relates to the workloads types since different workloads have different resource demands characteristic; $(ii)$ The multi-core architecture characteristics can significantly affect the isolation property due to the shared resource contention like core and cache; $(iii)$ Our proposed two optimization mechanism can improve the performance isolation and fault isolation efficiently.

Future work will include developing other effective mechanisms to improve the isolation property including efficient shared resource management mechanisms.

## VIII. Acknowledgments

We are grateful to the anonymous reviewers for their comments and suggestions on the paper. This work is funded by the National 973 Basic Research Program of China under grant NO.2007CB310900 and National Natural Science Foundation of China under grant NO. 60970125.

## References

[1] OpenVZ: Server virtualization open source project. *http://openvz.org/*, 2010.

[2] Standard Performance Evaluation Corporation, SPECjbb. *http://www.spec.org/jbb2005/*, 2010.

[3] P. Apparao, R. Iyer, and D. Newell. Implications of cache asymmetry on server consolidation performance. In *IISWC '08: Proceedings of IEEE International Symposium on Workload Characterization*, pages 24–32, Sept. 2008.

[4] P. Apparao, R. Iyer, X. Zhang, D. Newell, and T. Adelmeyer. Characterization & analysis of a server consolidation benchmark. In *VEE '08: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 21–30, 2008.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003.

[6] L. Cherkasova and R. Gardner. Measuring cpu overhead for i/o processing in the xen virtual machine monitor. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 24–24, 2005.

[7] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. Matthews. Xen and the art of repeated research. *USENIX annual Technical Conference*, pages 135–144, 2004.

[8] K. Govil, D. Teodosiu, Y. Huang, and M. Rosenblum. Cellular disco: resource management using virtual clusters on shared-memory multiprocessors. *ACM Trans. Comput. Syst.*, 18(3):229–262, 2000.

[9] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat. Enforcing performance isolation across virtual machines in xen. In *Middleware '06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, pages 342–362, 2006.

[10] N. E. Jerger, D. Vantrease, and M. Lipasti. An evaluation of server consolidation workloads for multi-core designs. In *IISWC '07: Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization*, pages 47–56, 2007.

[11] H. Kim, H. Lim, J. Jeong, H. Jo, and J. Lee. Task-aware virtual machine scheduling for i/o performance. In *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 101–110, 2009.

[12] Sysbench benchmark: *http://sysbench.sourceforge.net/*, 2010.

[13] M. R. Marty and M. D. Hill. Virtual hierarchies to support server consolidation. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 46–56, 2007.

[14] J. N. Matthews, W. Hu, M. Hapuarachchi, T. Deshane, D. Dimatos, G. Hamilton, M. McCabe, and J. Owens. Quantifying the performance isolation properties of virtualization systems. In *ExpCS '07: Proceedings of the 2007 workshop on Experimental computer science*, page 6, 2007.

[15] A. Menon, A. Cox, and W. Zwaenepoel. Optimizing network virtualization in Xen. In *Proc. USENIX Annual Technical Conference*, pages 15–28, 2006.

[16] A. Menon, J. Santos, Y. Turner, G. Janakiraman, and W. Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *VEE: Proceedings of the 1st ACM Conference on Virtual Execution Environments*, pages 13–23, 2005.

[17] IOzone benchmark: *http://www.iozone.org*, 2010.

[18] D. Ongaro, A. L. Cox, and S. Rixner. Scheduling i/o in virtual machine monitors. In *VEE '08: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 1–10, 2008.

[19] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005.

[20] C. A. Waldspurger. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, 2002.

[21] A. Whitaker, M. Shaw, and S. Gribble. Denali: Lightweight virtual machines for distributed and networked applications. Technical report, University of Washington, February 2002.