

The On-going Evolutions of Power Management in Xen

Gang Wei, Jinsong Liu, Jiajun Xu, Guanqun Lu, Ke Yu, Kevin Tian
Open Source Technology Center (OTC), Intel Cooperation

Abstract

While virtualization and power management technologies stay as hottest topics in computer research, increasing attractions have emerged when two are combined. Researches in this area [1] [2] [3] have focused mainly on global coordination framework and power budget policy in large scale data center. On the other hand, insufficient exploitations are seen on role of two important factors: green computing capabilities of VMM, and distinct designs of guest OS-es.

This paper looks into new evolutions of on-line power management technologies in Xen, relating to both hardware power-saving features and software optimizations for power. Several comparisons are made to help understand associated effects regarding to power efficiency: first, compare to old Xen revision for achieved improvements (Figure 1); second, compare to native OS for remaining gap; last, comparison between Xen and KVM is also presented which is the 1st data as known in this area. Latter two are included in Figure 2.

Then, another interesting work we present is to measure impact factors from VMs. We compare possible factors at two levels: OS difference (HVM vs. PV, Linux vs. Windows, etc.), and also power friendly implementations within OS-es (HZ, tick-less, etc.). Other factors such as timer mode could also draw some difference. The final result highlights the key role of VM itself in overall power efficiency, which could assist user choosing appropriate guest OS-es in real usage.

Idle power (cpuidle)

CPU Power states (C-states) could reduce power when CPU is idle, by closing some internal gates. Normally a deeper C-state closes more internal components (e.g. L2 cache) with less power consumed, but also incurs higher entry/exit latency in the meantime. **Figure 1** gives a clear view about

C3 effect on a two-core mobile platform.

Some side effects exist when deeper C-states are requested. For example, **system/TSC skew** could be observed on CPUs where TSC is stopped in deep C-states. Initial Xen implementation is simply straightforward to save/restore TSC at entrance/exit of deep C-state transition on each CPU, based on elapsed platform timer ticks. This however leads to increasing system/TSC skews if unequal deep C-state requests are issued on different CPUs. New algorithm is discussed in [4], which could keep TSC skew within a static range, however still with some limitations. Some CPU supports **always-running-TSC** even when it's in deep C-state. Xen skips software recovery process when this feature is reported by CPUID instruction.

Similarly local APIC timer could be stopped in deep C-state too, and thus impacts software timers. Xen adopts a **broadcast mechanism** by programming platform timers with a closest expected deadline out of all CPUs in deep C-state, and then issuing IPIs to wakeup idle CPUs with deadline expired in platform timer ISR. HPET broadcast is preferred, with PIT as backup. Here only one platform timer is used to serve all CPUs in deep C-states, which implicates less idle possibility on CPU which happens to be the interrupt target of platform timer. Then **MSI-based HPET** supported in some new chipsets could alleviate that limitation, which allows each HPET channel serving fewer or even only one CPU, and then in the latter case deliver wakeup message to the idle CPU without disturbing others. The associated patch is ready, but still requires more measurements before pushing to Xen upstream.

To reduce idle power consumption, two key aspects are normally considered: *break events* and *package C-states*. **Range timer**, [5] previously named deferrable timer, adds a configurable slop to deadline of each active timer, and then allows rendezvousing multiple timers at same expiration time point as long as that point sits between [expire, expire + slop] for those timers. This effectively reduces APIC

timer interrupts, especially when multiple VMs with similar virtual timer configurations run together.

On the other hand, **Menu governor**, not like its precedent (ladder governor) which simply moves up/down one step from previous state, now picks appropriate C-states directly just like clicking a menu. Three important factors are taken into consideration: latency of each C-state, nearest timer deadline, and estimation on last unexpected break event. This new algorithm could adapt to flexible performance/power requirement more intelligently. Nevertheless, this area is expected with more experiments and improvements in the future being its core role.

Shared resources by multi-core or SMT can only be powered down when all cores/threads sharing them enter same deep C-state, known as package C-states, which saves more power if compared to a package in partially idle. **Power aware scheduler**, is such an attempt to create more package C-states chances. Not like default algorithm in credit scheduler, where VCPUs are spanned across multiple packages to avoid shared resource confliction, power aware scheduler tempts to first saturate one package while leaving rest idle. Besides possibly more power saved, however, in the meantime it may affect performance to some degree in some cases, such as a memory intensive workload. Actual effect is still in analysis, and code may head toward wider discussion and acceptance in a near future.

Run-time power (cpufreq)

Dynamic CPU frequency/voltage scaling is another power-saving method especially when CPUs are in load line, allowing quick adjustment to frequency/voltage upon demand in small interval (ms) while incurring negligible transition overhead. Four governors are supported in Xen. **User-space governor** gives full control permission to user without Xen's intervention, which fits usages in diagnostic and research areas. **Performance governor** attempts to drive CPU in highest operation point. A typical usage is to activate **Intel Dynamic Acceleration** (also known as **Turbo mode**) feature, which may not be the default setting from BIOS. **Power-save governor**, just the reverse, always places CPUs in lowest frequency which consumes

smallest power with obvious performance drop-down expected. Last, **On-demand governor** monitors utilizations on each CPU and then adjusts operation points upon predefined thresholds and demands in small granularity. SPECpower poles in attached figures show the co-effect from both cpuidle and cpufreq (on-demand governor is used). Those provide a solid base for subsequent research on various workloads and virtualization scenarios.

Tools

Xenpm, as 1st power management tool in Xen, now provides rich control options associated to cpuidle and cpufreq, such as preferred governors, available P-states/C-states, sampling interval, thresholds, etc. Another important role of Xenpm is to check dynamic statistics in the fly, such as total/average C-state residency, transition count, etc. which provides 1st hand datum to understand underlying power/performance behavior. We expect to add more capabilities to Xenpm, such as power profile, etc. **Xentrace**, another powerful trace tool in Xen, is now extended to log every C-state/P-state transition message, which could be analyzed offline to acquire more meaningful information. One interesting usage on Xentrace, which we have in plan, is to construct and compare core/package C-state activities, as the guidance for next level tuning.

Power impact factors from VM

Guest OS implementation could impact power-saving degree although we are, and will continue, tuning Xen to a more power efficient VMM. One immediate example, which spurred our further investigation in this area, is about RHEL5 kernel (2.6.18) when we analyze power gap between RHEL5 and Windows XP HVM guests as shown in **Figure 2**. 1000HZ is the default tick interval in 2.6.18, which implicates 1000 virtual interrupts/sec. On the other hand, 2.6.18 doesn't incorporate clockevent framework, and thus local APIC is still programmed as periodical mode with period aligned to HZ too. When 2 VCPUs is configured in our test case, this adds another 2000 virtual interrupts/sec which sums up to 3000 virtual interrupts/sec. That means 3000 break events/sec if Xen accurately emulates

those timers which can easily defunct most power optimizations in Xen. For example, a rough calculation gives ~300us maximum average C-state residency while even entry/exit latency to deep C-state could be larger than 100us.

Two levels of factors are measured in this paper to help understand power friendship from different guest OS-es. First, we'll show an overall picture about power characteristics of guest OS-es, where both PV and windows HVM guests are observed adding less burden to Xen. Then, more comparisons are made on Linux HVM guest, by touching relevant kernel configuration options, such as HZ, idle tick-less, fully dynamic tick, high resolution timer, etc. Some of them could reduce virtual timer ticks a lot, e.g. given a fully dynamic tick model, while others may generate more break events such as high resolution timer. Other factors are also checked such as timer mode which could result various virtual timer interrupt injections. Our work tempts to encourage user to take power factor into consideration at future deployment, as a 'bad' OS could consume more power regardless of which VMM is used.

Conclusion

We introduce recent evolutions in Xen power management area. From attached figures, Xen had improved a lot since last presentation, and most importantly now reached an equal (idle) or better (SPECpower) position compared to KVM which stands on mature technologies of native Linux. But compared to native Windows gap is still obvious with possible cause from device PM. Last but not the least, this paper makes a fresh step into power impacts from OS-es, which provide meaningful guidance on OS choice at deployment. Our work in above portrayed areas will continue in depth, and also extend to wider scope such as device PM, with goal that Xen could be a good candidate for either research or production purpose as far as power efficiency is concerned.

References

[1] R. Nathuji et al. VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems. SOSP'07, 2007

[2] J. Stoess et al. Energy Management For Hypervisor-Based Virtual Machines. USENIX annual technical conference, 2007

[3] R. Nathuji et al. VPM Tokens: Virtual Machine-Aware Power Budgeting in Datacenters, HPDC'08, 2008

[4] G. Wei et al. [PATCH] CPUIDLE: revise tsc-save/restore to avoid big tsc skew between cpus,

<http://lists.xensource.com/archives/html/xen-devel/2008-12/msg00175.html>

[5] K. Fraser, Re: [Xen-devel] [PATCH 0/2] range timer support,

<http://lists.xensource.com/archives/html/xen-devel/2008-10/msg00776.html>

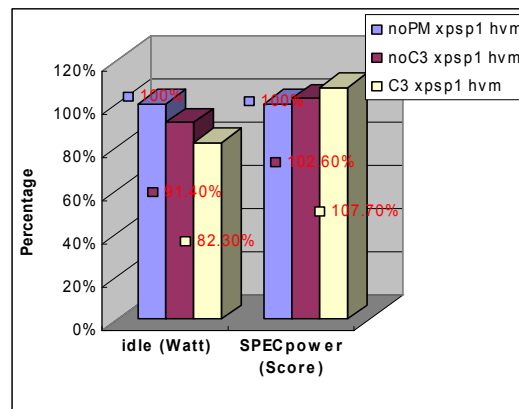


Figure 1

a) 'noPM' has both cpuidle and cpufreq disabled, and vice versa for other two cases. Compared to 'C3', noC3 has maximum C-state limited to C2.

b) For Idle, lower watt is better. For SPECpower, higher score is better. 'C3' case is best in all.

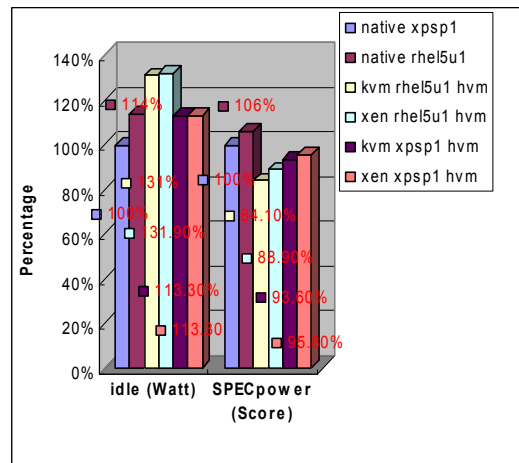


Figure 2

a) Xen has similar idle watt as Kvm, but SPECpower score of Xen is better

b) RHEL5 hvm guest has highest idle watt in both Xen/Kvm. Here RHEL5 kernel is the major reason with 3000 intr/s requirement