

Xen 系统中 CPU 间歇性故障的自适应策略

沈 霆, 李明禄, 翁楚良

(上海交通大学电子信息与电气工程学院, 上海 200240)

摘要: Xen 虚拟化环境没有考虑 CPU 的间歇性故障带来的影响。基于此, 建立模拟 CPU 间歇性故障的时间模型, 在该模型下未修改的 Xen 系统中的虚拟机会立刻崩溃。提出一种自适应的策略来改进 Xen 的 CPU 调度, 该策略主动跟踪 CPU 的状态变化, 将发生故障的 CPU 上的虚拟处理器迁移到可用的其他 CPU 上。实验结果表明, 当 CPU 间歇性故障频繁发生时, 应用该策略可以使虚拟机继续稳定地工作, 性能平滑地降低。

关键词: Xen 系统; 虚拟化; 多核技术; 间歇性故障; 自适应策略

Self-adaptive Strategy for CPU Intermittent Faults in Xen System

SHEN Ting, LI Ming-lu, WENG Chu-liang

(School of Electronic Information and Electric Engineering, Shanghai Jiaotong University, Shanghai 200240, China)

【Abstract】 Xen does not consider the impact of CPU intermittent faults. On the basis of this, this paper builds a time model for simulating intermittent fault in Xen. In the model, domains in unmodified Xen system will crash immediately. Therefore the paper proposes a self-adaptive strategy which does CPU scheduling in Xen. It spontaneously traces the change of CPU status and migrates virtual processors mapped in faulty CPUs to available CPUs. Experimental result shows that the virtual machine can keep running steadily and the performance of workloads in Xen reduces smoothly while intermittent occurs frequently.

【Key words】 Xen system; virtualization; multicore technology; intermittent fault; self-adaptive strategy

1 概述

Xen^[1]是一个在 IA-32(x86, x86-64), IA-64 及 PowerPC 970 等架构基础上的虚拟机监视器。Xen 的 Hypervisor(管理程序)介于物理层和 Xen 的虚拟机(domain)之间, 它保证每个正在运行的虚拟机获得一定的 CPU 时间。

间歇性故障是硬件领域的普遍现象。间歇性故障指的是硬件在一段时间内突发的故障, 通常由电压、温度等因素的波动而引起。CPU 间歇性故障会对多核处理器的性能和安全造成影响^[2], 因此, 也会对 Xen 的调度性能产生影响。本文在 Hypervisor 中建立一个模拟 CPU 间歇性故障的时间模型, 在该模型下, 未经修改的 Xen 系统中的虚拟机会立刻崩溃。

本文分析了 Xen 中默认使用的也是最常用的 CPU 调度算法——Credit 调度算法, 发现该算法并未采集 CPU 上下线状态的变化情况。如果一个 CPU 突然下线, 在该 CPU 上执行和等待的线程就会终止。在 Xen 中, 该线程是指 VCPU(虚拟处理器), Xen 的每一个 domain 关联一个或多个 VCPU。在该 CPU 再次上线之前, 关联离线 CPU 上 VCPU 的 domain 就会无法正常工作, 因此, 导致该 domain 对应虚拟机的系统崩溃。

本文针对 Xen 调度算法存在的问题, 提出了应对 CPU 间歇性故障的自适应策略。该策略周期性地跟踪 CPU 的状态变化, 一旦某个 CPU 从在线状态变为离线状态, 就把发生故障的 CPU 上的负载转移到其他的 CPU 上。实验表明, 当间歇性故障频繁发生时, 应用了自适应策略的 Xen 系统中的虚拟机将能继续稳定地工作, 并且性能平滑地下降。

2 间歇性故障时间模型的模拟

文献[3]对 CPU 间歇性故障进行了深入的分析, 认为它会频繁地发生。而真实环境中的间歇性故障显然无法用于实

验目的, 因此, 本文需要自行模拟 CPU 间歇性故障。研究人员们通常使用低耗费的基于模拟的故障植入^[4]来评估系统的健壮性。一种常见的方案是模拟 Xen 系统的物理层, 但是模拟产生的 CPU 无法提供连续的时间, 因此, 也无法提供间歇性故障的连续时间模型; 另外, 这种情况下 Xen 中的客户操作系统所使用的 CPU 相当于被虚拟化了 2 次, 导致这个问题变得非常复杂, 难以精确地分析和模拟。

基于以上的原因, 模拟 Xen 系统的物理层是不合适的。但从 Xen 3.4 之后, 物理 CPU 的热插拔(hotplug)的基本模块已经实现。在 Dom0 中, 用户可以编写 C 程序, 使用 xenctrl.h 下的库函数 xc_cpu_offline/online 来下线和上线指定的物理 CPU。这种上下线是内核粒度上的, 因此, 可以使用这个模块来模拟 CPU 的间歇性故障。然而, 直接使用该方法上下线 CPU 有非常大的延迟。在工作负载较重时, 发出指令让 CPU 下线到 CPU 实际下线可能要数秒的时间, 因此, 无法用其建立有效的时间模型。延迟大的主要原因是: 在某个 domain 中发起的上下线 CPU 指令并非立刻执行, 而是要接受 Xen 调度程序的分配, 负载越高, 被调用到的延迟也越大。

本文在 Xen 的 Hypervisor 中加入了用于周期性上下线某个 CPU 的 Hypercall 实现, 所谓的 Hypercall 指的是在 domain 通过用户指令来执行系统级的特权操作^[5]。CPU 周期性上下线的 Hypercall 是该 Hypercall 的主要算法。其中, cpu_down

基金项目: 国家“973”计划基金资助项目(2007CB310900); 国家自然科学基金资助项目(90612018, 90715030)

作者简介: 沈 霆(1985 -), 男, 硕士研究生, 主研方向: 虚拟化技术; 李明禄, 教授、博士生导师; 翁楚良, 副教授

收稿日期: 2010-02-20 **E-mail:** clist@sjtu.edu.cn

和 `cpu_up` 是 Xen 内核提供的 API。实验表明，通过该算法，每一次上下线的时间开销大大降低，被控制在 3 ms 以下。

算法

```

1  function do_fault_op(cpu, duration, interval) //主函数
2  if cpu = -1 then //结束周期性上下线
3  if not cpu_online(cpu) then
4  cpu_up(faulty_cpu) //如果
5  end if
6  stop_timer(offline_ticker) //终止周期下线计时器
7  stop_timer(online_ticker) //终止周期上线计时器
8  else
9  faulty_cpu <- cpu //获取全局变量Faulty CPU
10 init_timer(offline_ticker, self_cpu_offline)
11 set_timer(offline_ticker, NOW()) //开启周期下线计
//时器
12 init_timer(online_ticker, self_cpu_online)
13 set_timer(online_ticker, NOW() +
MILLISECS(duration))
//滞后duration毫秒后开启周期上线计时器
14 end if
15 end function
16 function self_cpu_offline() //周期性下线Faulty CPU
17 cpu_down(faulty_cpu) //Xen内部函数调用(asm-x86\
smp.h)
18 set_timer(offline_ticker, NOW() + MILLISECS
(interval))
19 end function
20 function self_cpu_online() //周期性上线Faulty CPU
21 cpu_up(faulty_cpu) //Xen内部函数调用
22 set_timer(online_ticker, NOW() + MILLISECS
(interval))
23 end function

```

为了使该 Hypercall 用于模拟 CPU 间歇性故障的时间模型，需要将 Xen 在下线 CPU 时一些计划性的操作去除，比如强制暂停 CPU 调度器对系统中所有 VCPU 进行的调整。这些计划性的操作不符合间歇性故障对于突发性的要求。在去除这些操作之后，模拟 CPU 间歇性故障的时间模型就建立了。在该模型下，本文对 Xen 系统进行了测试。不管是否有工作负载，系统中的虚拟机都会崩溃。

3 Xen 的 CPU 调度分析

3.1 Xen 的 Credit 调度器

在 Credit 调度器中，每一个物理 CPU 维护一个本地的可执行 VCPU 的运行队列。VCPU 有 2 种优先级 `over` 和 `under`，`over` 表示一个 VCPU 用掉了当前的 credit，`under` 则反之。因此，运行队列可以根据优先级对队列上的所有 VCPU 进行排序。当一个 VCPU 加入到一个运行队列中时，它被安置到与它优先级相同的 VCPU 后面。

在每一个时钟周期(10 ms)中，正在运行的 VCPU 的 credit 会被减去一个固定值。为了平衡总的 credit，调度器使用了一个全局的 Accounting 线程重新计算每个 domain 可以获得多少 credit 并分配给 VCPU。如果一个 VCPU 的 credit 为负值，那么它的优先级变为 `over`，反之则为 `under`。

每个 CPU 会在每个 30 ms 时间片的最后阶段做出一个调度的决定，即中断当前正在运行的 VCPU 并取出自己的运行队列中的第 1 个 VCPU 作为下一个运行的线程。Credit 调度器会作如下的负载平衡：如果一个 CPU 在自己的运行队列中

找不到优先级为 `under` 的 VCPU，那么它将尝试从其他 CPU 的队列上偷取合适的 VCPU。

3.2 CPU 间歇性故障的影响

在 Credit 调度算法中，每个 CPU 关联一个正在运行的 VCPU 和一个运行队列。当所有 CPU 都保持正常状态时，理论上每个可运行的 VCPU 都会被调度到。然而，CPU 的间歇性故障的存在会摧毁这个观念。当某个物理 CPU 突然发生了一次故障，它会迅速地下线并且无法执行正常的线程，而调度器并不会意识到这个故障的发生。正在执行的 VCPU 和其运行队列上可执行的 VCPU 继续停留在暂时故障的 CPU 上，并且无法被其他在线 CPU 所发现，它们既无法执行也无法被迁出。直到离线的 CPU 恢复前，物理 CPU 无法获取这些 VCPU 并执行它们，导致这些 VCPU 所关联的虚拟机的操作系统崩溃。

4 应对间歇性故障的自适应策略

4.1 基本思想

针对 CPU 间歇性故障所造成的 Xen 虚拟机系统崩溃，本文设计了一个自适应的策略来解决该问题，并使系统变得更健壮。该策略由调度器和跟踪器构成。跟踪器周期性地跟踪所有启动的 CPU 的在线状态，一旦发现某个 CPU 下线，就启动恢复器。恢复器将该离线 CPU 运行队列上的所有 VCPU 迁移到运转正常的 CPU 上。图 1 显示了加入自适应策略后的 Xen 系统框架。

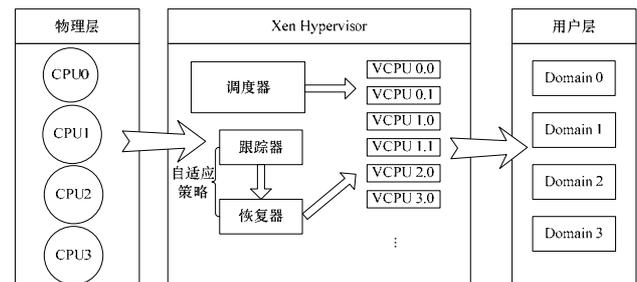


图 1 加入自适应策略后的 Xen 系统框架

4.2 CPU 在线状态的跟踪

Credit 调度器并不关心 CPU 何时下线，并主动做出调整策略，所以，需要在 Hypervisor 中增加一个跟踪器(tracer)，用以跟踪 CPU 的在线状态。该跟踪器每隔一段时间记录当前所有已启动的 CPU 的在线状态。一旦 CPU 被发现当前离线而上一次在线，跟踪器将该 CPU 的 ID 传给恢复器(restorer)。实际上，只需要跟踪 CPU 从在线状态变为离线状态这一事件，而不需要跟踪再次在线的情况，因为 Credit 算法的负载平衡机制会很快地让上线后的空闲 CPU 得到一定的负载。在具体实现上，需要维护一个二进制数组，Xen 系统初始化时记录所有已启动 CPU 的在线状态。之后周期性地与当前的 CPU 在线状态进行比较，如果发现 CPU 下线则启动恢复器，周期结束前用当前的 CPU 在线状态覆盖上一次的记录。

需要注意的是，虽然在本文设计的间歇性故障模拟环境下，CPU 是否在线可以在内存中快速读取。然而，真实环境下 CPU 是否在线可能需要中间件的支持，并带来一定的额外时间开销。尽管实现的细节会变得更复杂，但是跟踪器的基本原理和设计都是相同的。

4.3 从 CPU 故障中的恢复

Credit 调度器不会从离线 CPU 的运行队列上提 VCPU，这从根本上导致了系统的崩溃。一旦 CPU 的在线状态变化被

确认, 恢复器就针对该 CPU 进行恢复工作。这里所谓的从 CPU 故障中恢复指的并不是消除已经发生的 CPU 间歇性故障, 而是在故障已经发生的情况下如何调整系统以使之能继续运转。在本文建立的模拟环境下, 只需将 CPU 的运行队列上的 VCPU 迁移到其他可用的 CPU 上。如果是真实环境中, 则可能需要更多的工作。图 2 显示了恢复过程的实例。

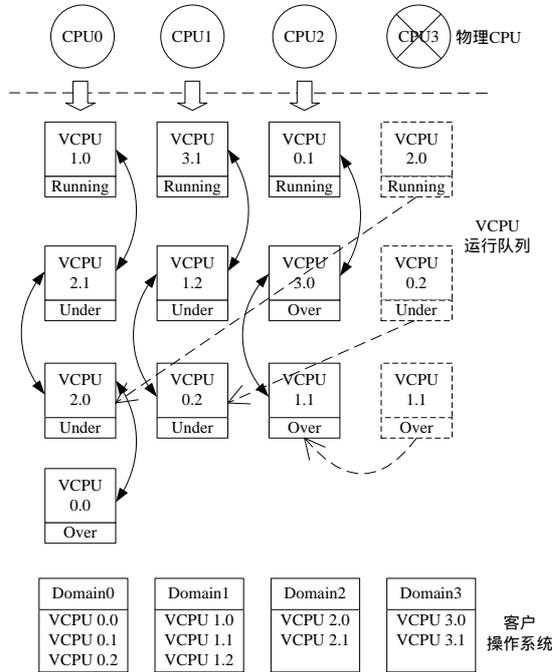


图 2 从 CPU 故障中恢复的过程

对于每一个离线 CPU 的运行队列上的 VCPU, 只需要重用 Credit 调度器中为 VCPU 选取合适的在线 CPU 的算法。因此, 每个 VCPU 可以被插入到预先指定的 CPU 的优先级队列中。当这些都完成时, 所有的 VCPU 都可以被正常地调度, 于是, 系统就可以保持正常的运转, 性能也随之平滑地下降。

5 实验评估

5.1 实验环境

所有的实验都在一台 Dell PowerEdge 1900 服务器上进行, 该服务器有双四核心 Xeon X5310 处理器和 2 GB 内存。Xen 的版本是 3.4.0, 使用默认的 Credit 调度器。虚拟机 Dom0 运行的是 Fedora Core 8.0 发行版, 而 Dom1 上运行的是 Debian 4.0。为了在 Xen 中测试自适应策略的性能, 本文在 Dom0 上配置了 8 个 VCPU, 在 Dom1 上配置了 4 个 VCPU, 它们的 weight 均为 256。

本文选择了 2 个来自 SPLASH-2 性能测试的基准包 (benchmarks suites) 作为工作负载来测试系统的性能: 第 1 个是 BARNES, 它用来解决天体物理中的 N-body 问题; 第 2 个是 LU, 它用分块算法作 LU 分解。

在实验中, 工作负载同时运行于 Dom0 和 Dom1 中。在负载 BARNES 中, 设置 body 的数量为 262 144 个, 并设置 4 个处理器; 在负载 LU 中, 设置 $N=4\ 096$ (分解 N 阶矩阵), $B=16$ (Block 的大小), 并同样设置 4 个处理器。

5.2 间歇性故障下虚拟机的性能

第 1 个系列的实验是设置固定的故障发生间隔和不同的故障持续时间。故障发生的间隔设为 2 000 ms, 故障持续的时间从 200 ms~1 000 ms, 步长为 200 ms。设置 CPU 7 为故障 CPU。以故障持续时间 400 ms 为例: 从每一个 2 000 ms

的开始, CPU 7 发生故障下线, 然后过了 400 ms, CPU 7 被修复并重新上线。本文首先测试了在未修改的 Xen 系统中的情况, 结果为虚拟机迅速崩溃, 因此, 无法量化, 然后测试应用了自适应策略的 Xen 系统的性能。从图 3 中可以看到, 随着故障持续时间的增加, 负载的运算时间明显的增长。这个增长的主要原因是 CPU 7 的离线时间使整体运算性能的下降, 这也说明, 应用了自适应策略, 系统可以继续运转, 且性能会平滑地下降。

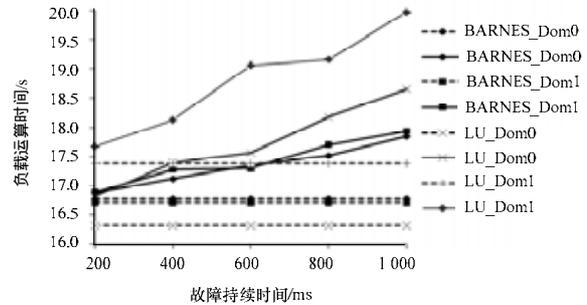


图 3 固定的故障发生间隔

第 2 个系列的实验是设置固定的持续时间和间隔时间比例, 该比例被设置为 0.25。如故障持续时间为 500 ms, 那么间隔时间为 2 000 ms。设置故障持续时间 250 ms~1 000 ms, 步长为 250 ms。该系列实验在未修改的系统中虚拟机依然迅速崩溃。从图 4 中可以看出, 随着故障持续时间的增长, 负载的运算时间降低了。这个现象的原因是, 虽然总的 CPU 损失时间相同, 但是检测和迁移过程在故障持续更频繁的情况下会耗费的更多。

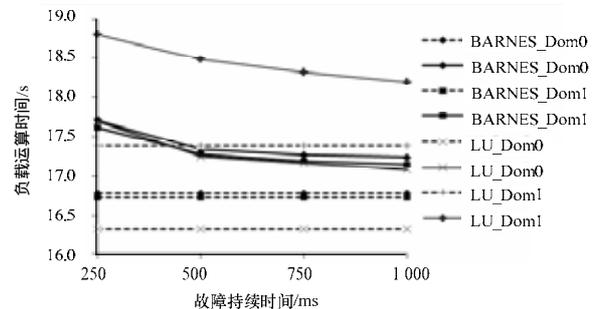


图 4 固定的故障持续时间和发生间隔比例

最后, 本文测试了整个系统的额外时间开销。应用了自适应策略的 Xen 系统在没有故障发生的情况下, 性能和未修改的系统几乎一样, 无法在实验图中区分。也就是说, 自适应策略的额外时间开销非常低, 几乎可以忽略不计, 这是因为 CPU 在线状态的跟踪过程读取的是内存数据, 因此, 消耗非常低。然而, 必须要说明的是, 如果要检测真实物理环境下的 CPU 故障, 则额外开销很可能会较大, 这也是本文系一步要解决的问题之一。

6 结束语

本文建立了模拟间歇性故障的时间模型。在该模型下, 未经修改的 Xen 系统中的虚拟机会迅速崩溃。对 Xen 的 CPU 调度进行了深入的分析, 并提出了应对 CPU 间歇性故障的自适应策略。实验表明, 应用自适应策略的 Xen 系统在间歇性故障频繁发生的情况下, 虚拟机可以稳定地工作, 性能平滑地降低。

(下转第 249 页)