多线程多核微处理器体系结构实例研究

——IBM Power5 微处理器

小组成员名单:

段玮、王茹、蔡嵩松、

李雷、刘奇、汪文祥

2006-5-26

目　录

1 引言

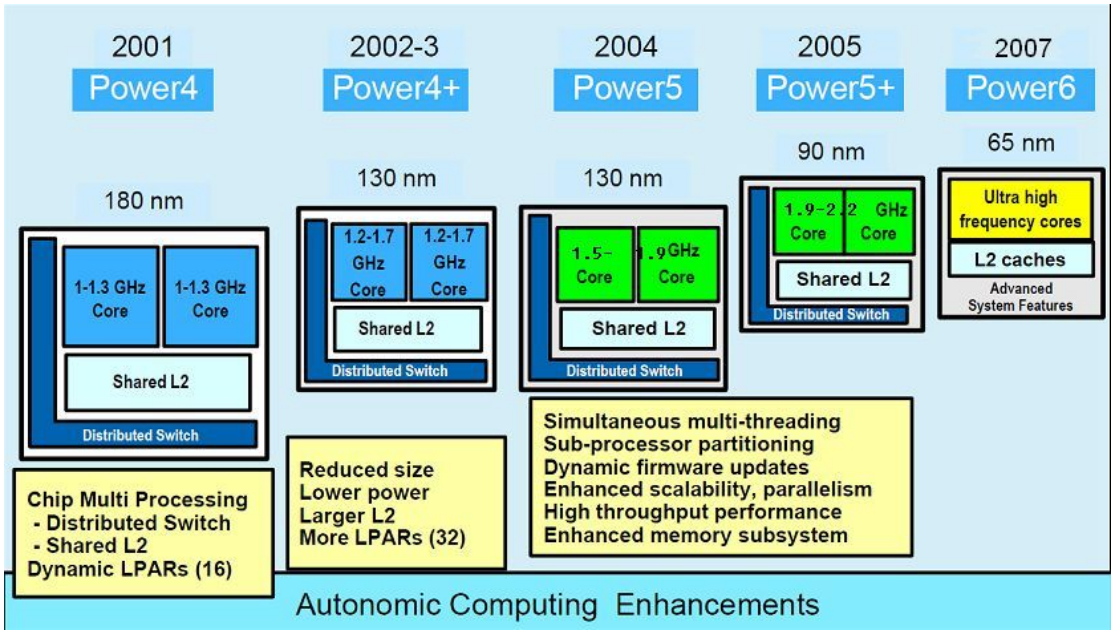
体系结构的研究，是科学也是艺术，其艺术的一面体现在我们总是要在多维的设计空间中找到一个尽可能优的解，而这一过程，虽然已经有了一些定量化的方法做支持，但具体地分析和最终定夺总归要由人主观地做出。受到软件程序特性影响和硬件工艺的物理限制，未来体系结构的发展已经不是仅仅通过提高主频就可以获得系统性能的大幅提升，同样使用大规模高复杂度的单核结构来开发 ILP，若想继续高速发展也面临着困境。为了避开眼前的这些难题，人们将目光转向了分布式的多核结构和片上 TLP 的开发。

体系结构技术发展的趋势是明显的，多线程和多核，作为两个关键技术方向必须得到我们的高度重视。即使是在未来的多型和可重构架构中，这两个领域的某些关键技术仍将为我们的设计提供有力的支持。对于这两个方向，学术界从上世纪末就开始了广泛的研究，进入二十一世纪后，在各个领域，含有多线程或多核特征的处理器层出不穷。作为技术分析，除了理论的研究外，对处理器实例的分析，更能给我们提供分析问题的新视角新思路。

本文将介绍小组对于 IBM 公司 2004 年推出的多线程多核处理器——POWER5——的学习和讨论。第 2 部分将对 POWER5 芯片及系统的结构进行分析。第 3 部分针对 POWER5 系统中操作系统所作的调整展开讨论。第 4 部分将提到 POWER5 实现中对于功耗的动态管理。第 5 部分给出 POWER5 的系统性能评测及分析。最后在第 6 部分，将给出课堂及小组讨论的内容。

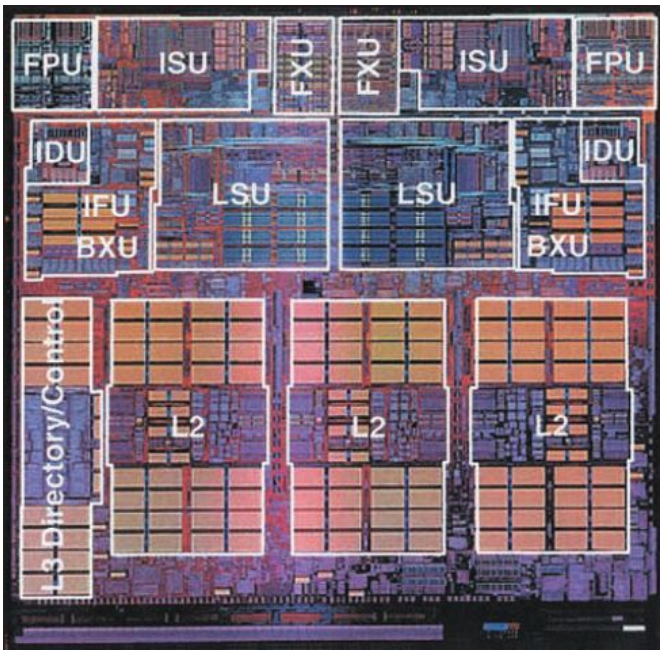
2 POWER5 处理器体系结构

2.1 IBM POWER 处理器 ROADMAP



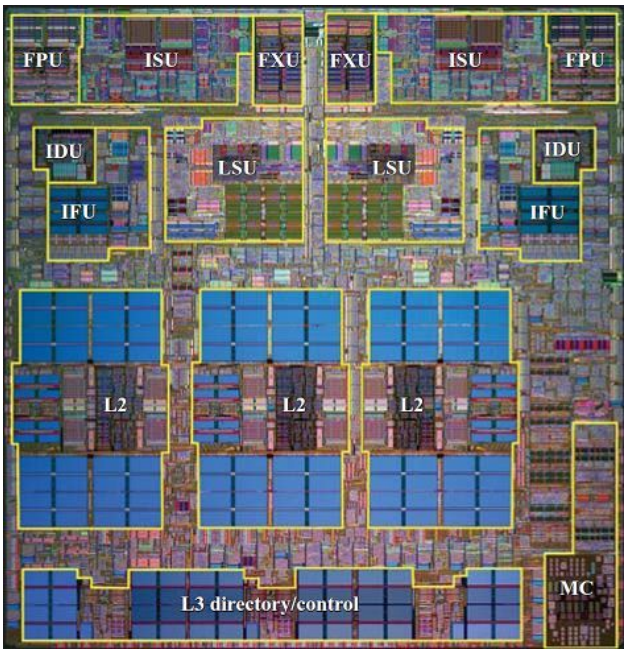
IBM 公司 1990 年推出第一款 POWER 处理器 POWER1。至今已经有 16 年的发展历史。

在 2001 年的时候 IBM 推出了第一款双核处理器 POWER4。它采用.18 工艺，主频最高达到 1.3GHz，支持动态逻辑分区功能。2002 年推出 POWER4 的增强型，区别在于改用了更加先进的.13 工艺，最高主频达到 1.7GHz，除此之外在结构上增大了二级 cache，动态分区更加灵活。2004 年的时候 IBM 推出了第一款双核多线程处理器，首次在处理器核中实现了多线程技术。它也是采用.13 工艺，最高主频达到 1.9GHz。此外一个很值得关注的地方就是处理器间互连和内存带宽有惊人的提高。POWER5 的增强型可以仅仅看作是 POWER5 的 90 纳米版本。IBM 计划于 2007 推出 POWER6 处理器，它采用 IBM 的超高主频技术，主频有望达到 5GHz 左右。此外在结构上是双核或者也有可能采用 4 核。核内仍然采用 SMT 技术，有可能采用 L4 cache。下面我们着重介绍 POWER4 和 POWER5 处理器。



上面是 POWER4 的处理器的 Die。我们可以清楚的看到它们的取指(IFU)、译码(IDU)、发射(ISU)、执行部件(FPU、FXU、BXU 等)以及 L2 cache 和 L3 的目录、控制部分。

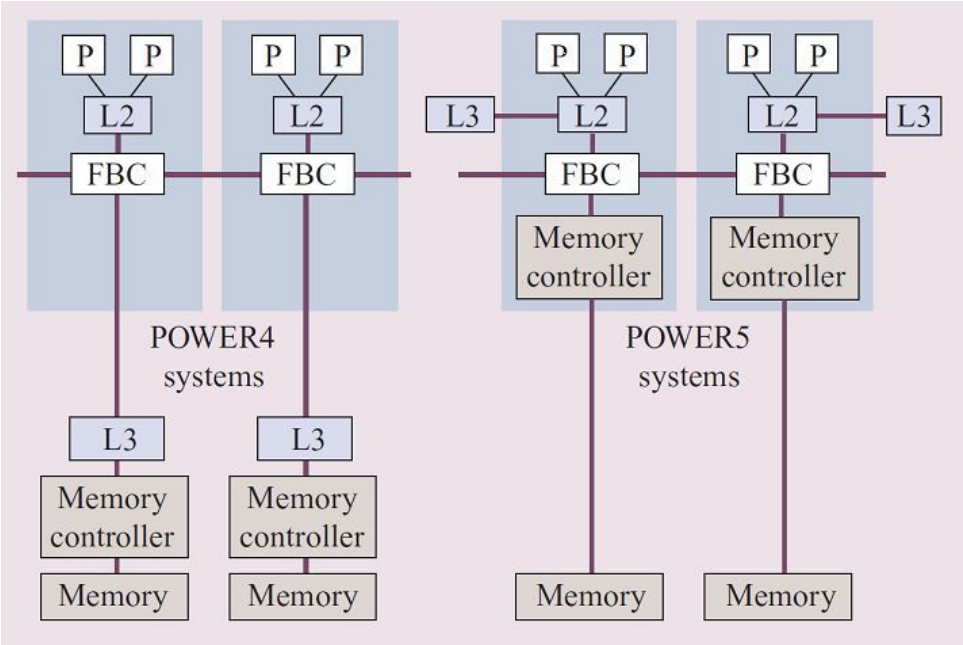
POWER4 的每个核是四发射的超标量处理器结构。



上页是 POWER5 的处理器的 Die。同样我们可以清楚的看到它们的取指(IFU)、译码(IDU)、发射(ISU)、执行部件(FPU、FXU 等)以及 L2 cache 和 L3 的目录、控制部分。才外一个主要的跟 POWER4 处理器不同的是它把内存控制器集成进了片内。

POWER5 的每个核同样是四发射的处理器结构，但是增加了 SMT 支持。

2.2 POWER5 处理器结构



上图是 POWER4 和 POWER5 处理器的结构上主要的变化，我们可以看到主要的变化有：

1、把 L3 cache 从到内存的通路上拿掉，作为 L2 cache 的牺牲 cache。这样做的好处是如果有 L2 的 miss，而且数据正好在 L3 中的话，可以大大降低 miss 的延迟。通常牺牲 cache 用于降低一级 cache miss 的惩罚，最早在 PA7200 处理器上得到应用，后来 K6、K7 处理器用大容量的二级 cache 作为牺牲 cache。

2、把内存控制器做到了处理器片内，进一步提高了内存通路的带宽、降低了内存存取的延迟。

通过上面的调整，主要带来的好处有：

1、降低了 L3 和内存的延迟。由于 L3 更加靠近 CPU 并且内存控制器做到了片内。L3 的延迟由 123 拍降低到 87 拍；内存的延迟由 351 拍降低到 220 拍。

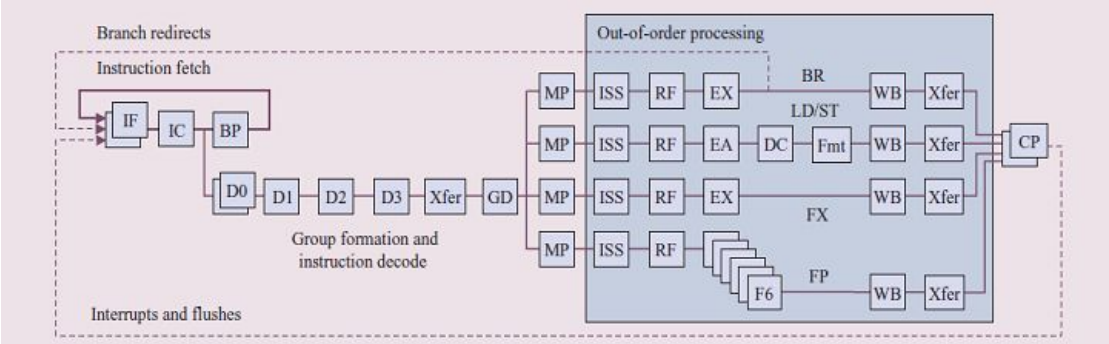
2、同时增大了 L3 和内存的带宽。内存的带宽由原来 POWER4 的 4GB/s 增加到 16GB/s。L3 的频率由原先的三分之一倍主频提高到二分之一倍主频。

3、此外，由于这样做提高了系统的集成度，进一步提高了系统的可靠性和稳定性。

2.3 POWER5 微体系结构

下面分两个部分介绍一下 POWER5 的微体系结构。第一部分是 POWER5 的流水线；第二部分是 POWER5 的 SMT 的设计考虑。

2.3.1 POWER5 流水线



首先给出各个部分的相应的意思，然后对个别地方做说明。

IF 取指部件，每个线程是分离的线程独立的 IF 和 PC。

IC 指令 Cache，每个周期一次通过 IF 部件得到 8 条指令。

BP 对 IC 中的指令进行扫描，如果遇到分支指令对分支进行预测。BP 采用的是类似于 Alpha 21264 中使用的锦标赛预测器。除了 branch-to-link-register 指令和 branch-to-count-register 指令，在扫描的同时，对分支地址也进行了预测。上述的两条指令的分支地址分别由硬件实现的 link stack 和 count cache 预测。

D0 即指令 buffer，这个阶段根据线程的优先级从指令 buffer 中一次取出最多 5 条来自同一个线程的指令给下一个阶段形成 group。

D1-D3 这些阶段形成 group。指令 group 形成的原因是为了减少要跟踪的指令的数目，降低因为 reorder 带来的开销。其实这个阶段的工作比较复杂，所以需要三个流水级来完成。Group 的形成的主要原则有 branch 指令要占据 group 的最后一个 slot，中间用 no-op 填充，如果没有分支指令最后一个 slot 的指令必须是 no-op 等等。同时在这个阶段会对每个 group 对应的 GCT 项进行填充。GCT 主要用来进行指令的 reorder。

GD 阶段根据 group 中不同指令的类型把指令 dispath 到不同的功能部件流水线上。根据功能部件功能的不同，主要分为分支指令、存取指令、定点指令和浮点指令流水线。

功能部件内部的 ISS 和 RF 分别为指令选取发射和取寄存器阶段。

CP 为 group 提交阶段，根据 GCT 重新组织顺序并提交。

2.3.2 POWER5 的 SMT 设计考虑

（1）资源的分配

	POWER4	POWER5
IFU	直接映射 64-KB L1 I-cache	两路相联 64-KB L1 I-cache
	4 项预取 buffer	大小不变,分开,每线程 2 项
	16 项 BIQ	大小不变,分开,每线程 8 项
	预测逻辑, Link stack, count cache, BHT	不变
IDU	8 项 IFB	每个线程 6 项,共 12 项
ISU	20 项 FIFO 结构 GCT	20 项队列 GCT
	发射队列：20 项 FPQ	24 项 FPQ
	80GPR, 72FPR, 32CR	120GPR, 120FPR, 40CR

LSU	32-KB 两路相联 L1 D-cache	32KB 四路相联 L1 D-cache
	1.45MB 片上 L2 cache	1.9MB 片上 L2 cache
	16MB 片外 L3 cache	36MB 片外 L3 cache，目录项片上
	8 项 LMQ	8 项 LMQ,加入线程控制
	128 项两路相联 D-ERAT	128 项全相联 D-ERAT
	32 项真实 LRQ	每个线程 16 项真实 LRQ 和 16 项虚拟 LRQ
	32 项真实 SRQ	每个线程 16 项真实 SRQ 和 16 项虚拟 SRQ

上面我们列出了 POWER5 处理器针对 SMT 的设计相对于 POWER4 处理器的主要的资源的变化情况。IFU、IDU、ISU 和 LSU 分别对应了取指部件、译码部件、发射部件和执行部件（以资源最复杂的 load store 流水作为例子）。

IFU 部分一级指令 cache 增加了相连度，然后预取 buffer 大小没有变化，分开设置。同样的 BIQ(Branch Information Queue，主要用来分支误预测后的回滚)大小没有发生变化，但是分开设置了。预测逻辑和预测使用到的主要的资源基本没有发生比较大的变化。

IDU 部分 IFU，即指令 buffer 大小发生了变化，由 8 项设置成了总共 12 项，分开设置。

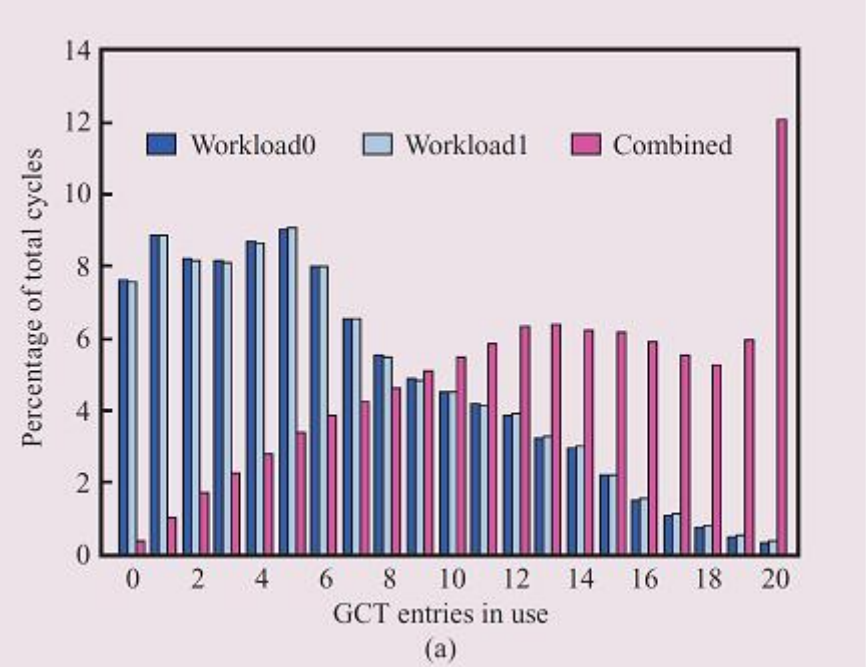
ISU 部分 GCT 的大小没有发生变化，但是结构组织形式发生了比较大的变化。原先的组织是采用 FIFO 的方式，现在采用链表的方式组织，从而能够让两个线程能够更好的共享。各个执行流水的对应的发射队列，没有发生很大的变化，只有 FPQ，即浮点运算的发射队列，由原来的 20 项增加到 24 项。然后重命名寄存器资源也发生了比较大的变化。大小增加了很多。

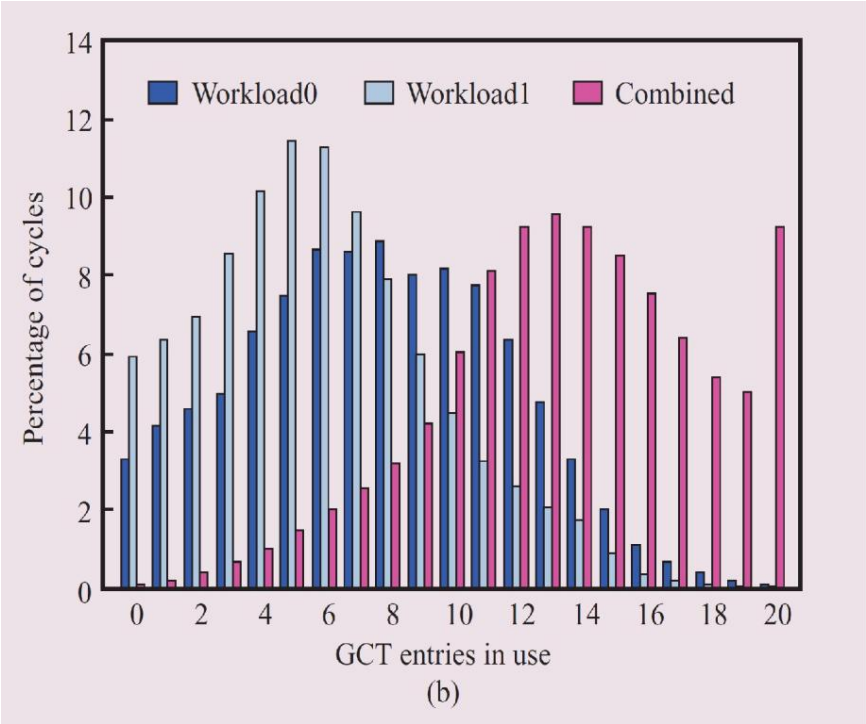
LSU 部件，L1 D-cache、L2 cache 和 L3 cache 采用普遍的方法是提高相连度和增大容量。LMQ（Load miss queue）主要作用是用来存放 L1 cache miss 的情况下需要到 L2 cache 里面去取的数据。地址翻译 cache 的 ERAT 的大小也没有发生变化，但是相连度增加了。LRQ 和 SRQ 的真实资源大小没有发生变化，但是是分开设置的，同时增加了相同大小的虚拟项，虚拟项的主要的作用是在译码阶段不用进行真正的 Load 或者 Store 地址的指定，而在指令发射阶段的时候必须把虚拟项的数据分配到真正的数据项中。

上面是资源的分配情况下面我们用一个具体的例子——GCT 的设置来说明在系统的设计阶段是怎么样决定一个资源的组织和大小的。

对于系统来讲设计的流程大致如下：

1、首先会有一个精确的基于踪迹的 cycle 级别的模拟器来对最初的设计思路进行设计空间的探索和验证。比如我们初步的决定可以保持 GCT 的大小不发生变化，但是把组织形式由原先的 FIFO 方式组织成链表的形式。那么这个设计在模拟器上得到的初步结果如下：





上面的两个不同的图使用了不同的 workload。对于图 a 来讲，workload0 和 workload1 是相同的，都是 java 的服务器的应用，它们应用的特征是具有适度的 cache miss。而对于图 b 来讲 workload0 是一个组合优化的应用，具有较高的 cache miss；workload1 是一个棋盘布局的应用，具有较低的 cache miss。通过对上面结果的观察，我们可以看出对于这样的设计，在没有增加很大的开销的情况下，得到了比较好的模拟结果，这样使得我们可以利用这个设计进行下一步的验证。

2、两组用汇编语言编写的性能验证程序在系统可以点亮的时候就进行测试，通过这个手段可以对性能进行初步的验证。

3、一旦 Linux 移植好了，就在机器上进行 benchmark 测试机器的性能，以验证模拟器

的结果，但是 Linux 缺乏有效利用 POWER5 性能检测功能的机制。

4、一旦其它部分移植好了 AIX 可以运行了，会有更加详细和准确的测试结果。

5、最后就可以对数据进行有效的收集和利用统计软件 SAS 进行分析。

需要注意的是整个设计是一个反复迭代的过程。POWER5 具有多个工程版本。

（2）动态资源平衡

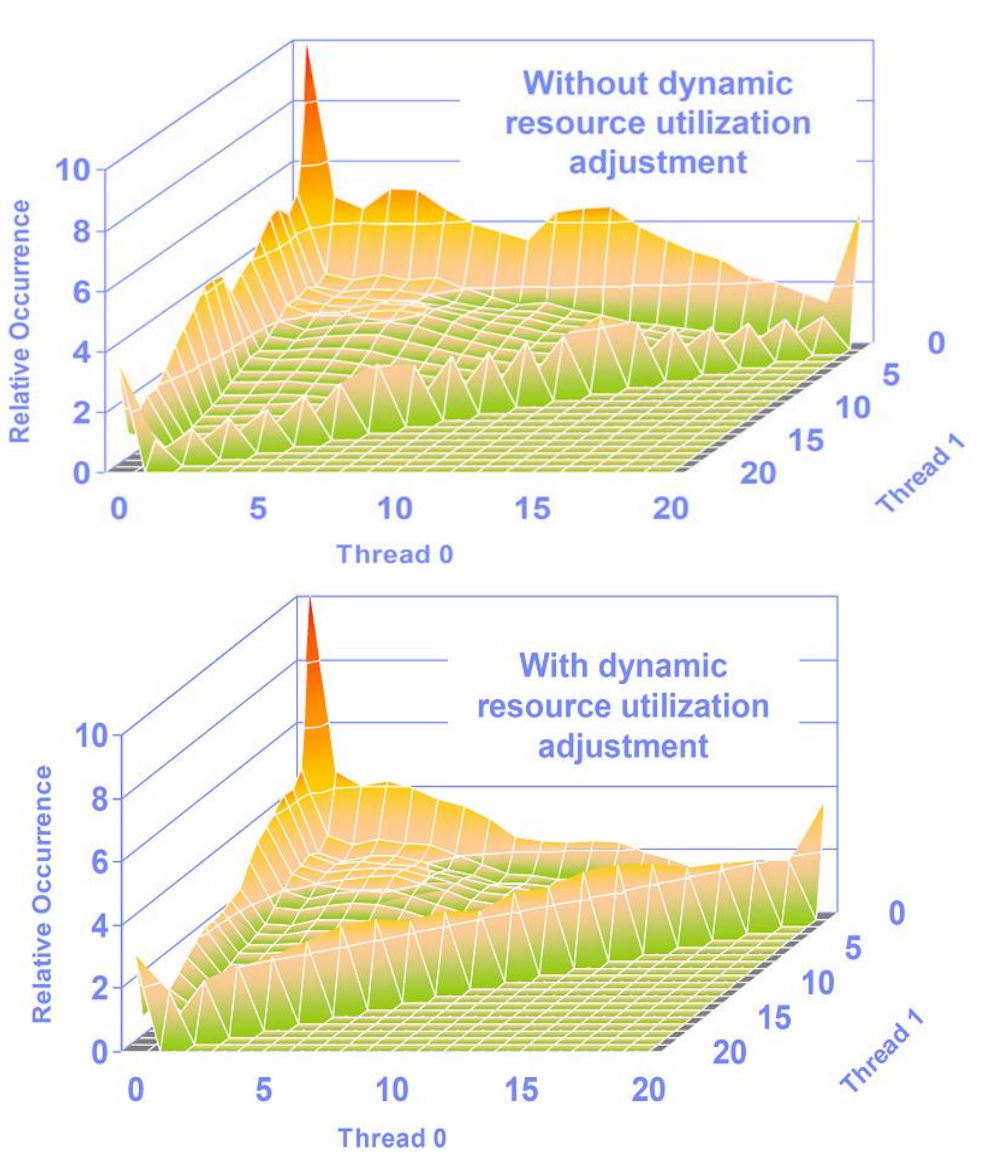
在一个实现了 SMT 技术的处理器核中，如果共享的资源长期被一个线程所占用，那么整体上会对系统的性能造成比较大的负面影响，因此就需要在处理器核中集成动态资源平衡逻辑。POWER5 处理器主要有一下的资源平衡策略：

A、当一个线程使用了过多的 GCT 表项的时候，处理器会降低这个线程的优先级别；

B、如果一个线程造成了比较严重的 L2 cache miss，那么处理器会抑制这个线程的指令译码；

C、如果一个线程正在执行一些长期难以完成的指令，比如说 sync，那么处理器会清空这个线程的指令队列，并且抑制这个线程的译码。

下面的图是没有使用和使用了动态平衡逻辑对 GCT 使用情况的统计：



可以明显的看出使用了动态资源平衡逻辑的情况，资源的分配更加的合理。

（3）线程优先级别的支持

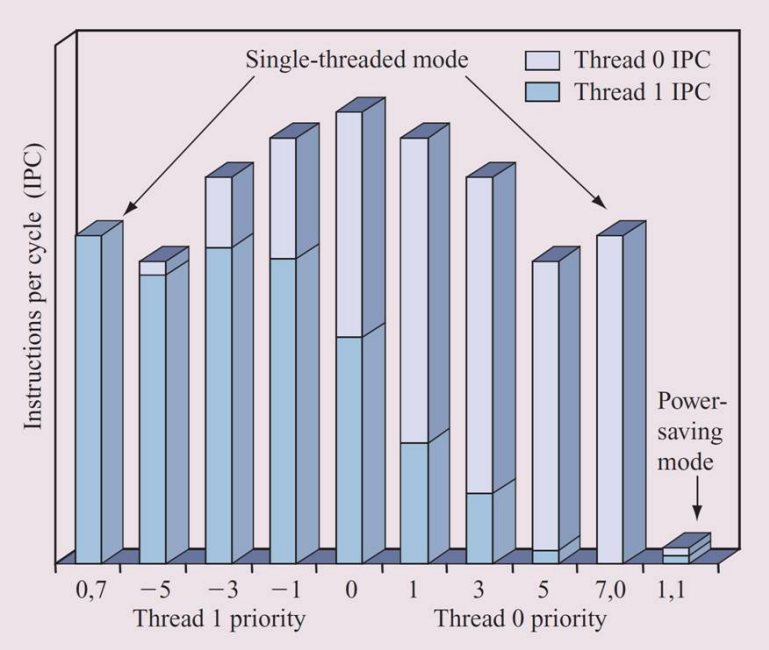
由于一下的原因，我们需要在处理器中增加线程优先级别的支持：

A、 需要降低正在进行自旋锁操作的线程的优先级别；

B、 需要降低正在进行 idle loop 的线程的优先级别；

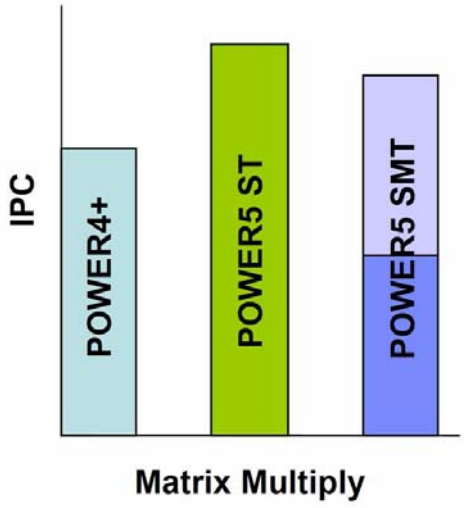
C、 有的时候我们有这样的需求，要求一个线程的优先级别比另外一个要高。

所以在 SMT 的处理器核中，我们需要支持线程优先级别。前面在描述流水线的时候，我们指出主要是在流水线的 D0 阶段加入的对线程的优先级别的支持。即如果线程的优先级别高 D0 会优先送给下面的 D1 阶段优先进行 group 的生成。POWER5 支持 0－7 一共 8 个级别的线程优先级别，其中如果两个线程的优先级别都设置为 1 的话，那么处理器会进入省电模式。下面是一个理想情况下的优先级别和 IPC 示意图：



（4）单线程模式的支持

在某些情况下面，比如 workload 占用大量的浮点运算部件资源或者内存带宽，那么打开 SMT 的支持，有可能是不合算的，这种情况可以由下面的图示看出：

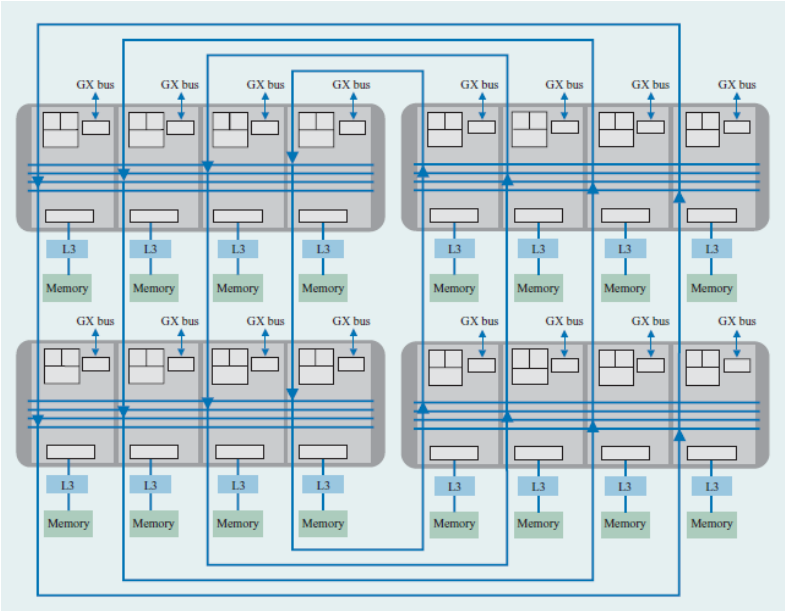


所以 POWER5 支持动态的 ST 和 SMT 模式的切换。系统在启动起来的时候是处于单处理模式下的，有三种方法可以进入 SMT 模式：

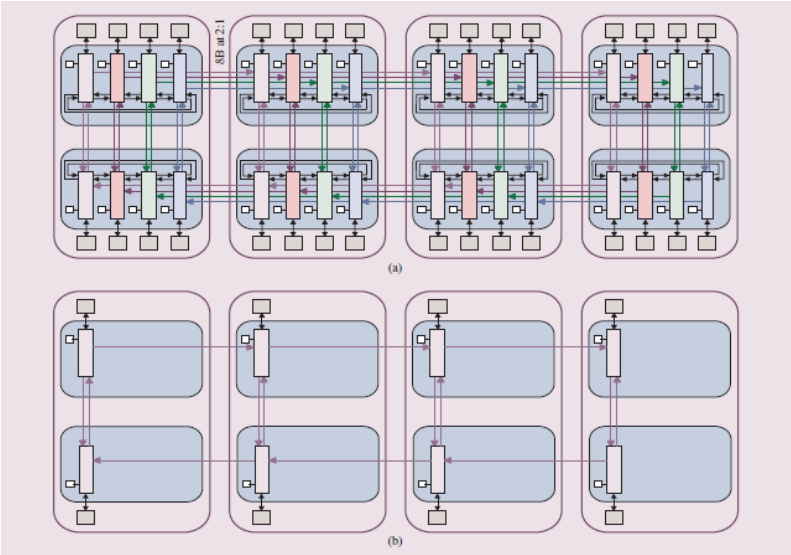
A、 可以通过对模式寄存器的读写打开 SMT；
B、 可以在系统有外部消息通知的情况下打开 SMT；
C、 协处理器可以协助激活 SMT 模式。

2.4 处理器互连

处理器间互连部分我们对照 POWER4 的互连方案进行说明。下面是示意图：



POWER4 处理器互连方案



POWER5 处理器互连方案

首先对 POWER5 的处理器互连方案进行简要的说明。高端服务器的处理器部分由至多 4 个 book 组成；每个 book 由 2 个 MCM 组成；而每个 MCM 包含 4 个 POWER5 处理器；每个处理器还包含了 2 个处理器核。

从上面的图示我们比较容易看出 POWER4 的互连方案采用的是广播的方式，而 POWER5 的方案是以环型连接(hybrid switch/bus[6])为主的方式。对它们进行比较我们可以

得到一下几点：

1、POWER5 比 POWER4 的处理器间通讯的数据流量要小；

2、但是在相同频率的情况下，POWER5 的通讯延迟要大于 POWER4。因为有的处理器间的数据要经过多拍才能到达目的地。

3、由于不是简单的将数据广播出去，POWER5 的控制器的复杂度要大于 POWER4。

4、POWER5 的处理器互连方案实现了很好的可扩展性，不会因为处理器的增加带来通讯量的激增。通过这样的结构可以构造 64 路的服务器系统，POWER4 只能构建 32 路的服务器系统。

3 POWER5 上操作系统的调整

3.1 多核技术引入的问题

所谓多核技术，就是一个圆片上面封装两个 cpu，8 个这样的圆片为一个 block，而多个 block 又可以组成一个大型处理器。这些排列整齐的 cpu 被编号后进行分区映射，不同的分区有不同的属性，有些是专用的，被映射到专用分区的 cpu 的编号是固定的，操作系统就被安装在这样的分区上，不同分区允许跑不同的操作系统；还有一些分区是共享的，被映射到共享分区的 cpu 是不固定的。

这就给操作系统的调度带来以下新问题：如何检测计算当前处理器有多少个核？怎样进行核间线程调度？

3.2 对应解决方案

3.2.1 核数目的计算

计算当前处理器有多少个核目前有两种方案：

（1）先检测有多少个 package，然后检测每个 package 中有多少个 cpu

（2）将 package 中的 cpu 编号，先检测 1 号 cpu 个数，再累加 2 号 cpu 个数，依此类推。

3.2.2 核间线程调度

进行核间线程调度，也就是进行 cpu 间任务分配，解决 NUMA 问题。这里使用的一个调度算法叫：affinity scheduling　　这种调度算法可用于解决访问时间不一致性问题。它的主要做法是，将关联紧密的两个线程，分配到同一个 CPU 上去运行。这种算法的两个不同形式：affinity domain scheduling & process-to-thread affinity；这两种不同形式主要的区别在于“ 关联范围” 不同。前一种形式由 linux 系统提供，“范围”包括 processor，caches，TLB，momery，以及其他的硬件资源，“范围”越底层，关联越紧密。后一种形式由 AIX 提供。“范围”区分很简单，按照进程划分，同一进程中的不同线程关联紧密。

3.3 多线程技术引入的问题

POWER5 SMT 允许每个微处理器核同时取得两条指令流，从 OS 角度看，有两套完整的寄存器组，因此，就好像每个内核有两个完整的逻辑处理器，也就是，每个内核拥有两个线程。这两个线程是同时运行还是一个运行一个休眠，硬件资源应该如何分配，都要操作系统协调调度。

由于在 SMT 机器上同时执行的一组线程在每个时钟周期都要竞争硬件资源，这使得执行不同的作业集合可能会获得不同的性能。

3.4 对应解决方案

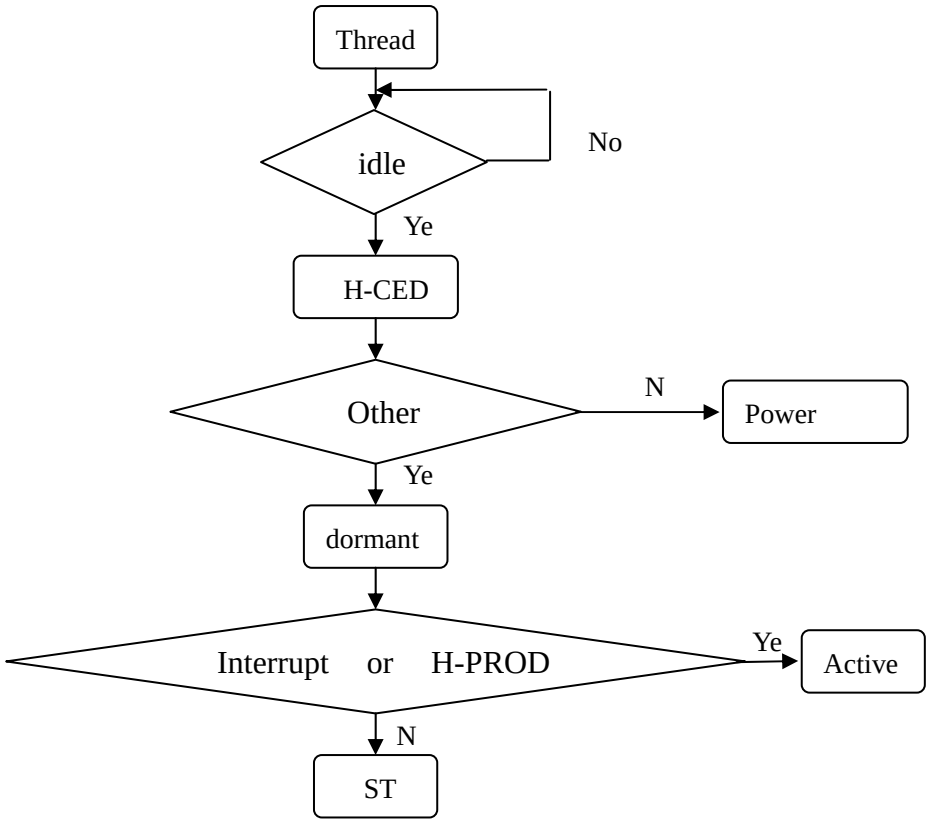
1、ST 模式和 SMT 模式间的转换

计算机启动的时候，微体系结构默认为 ST 模式，但当操作系统启动时，就会自动转入 SMT 模式，不论 AIX 还是 LINUX。

对于 AIX 和 LINUX，启动之初，都可以通过 hypervisor 的设置来选择模式。

在运行过程中，AIX 可以进行全局控制，通过 smtctl 命令来设置模式，并在 reboot 的时候，使用 DRAF 来通知计算机进行重新设置。而 LINUX 在运行中没有全局命令来改变系统状态，所以需要进行动态模式切换

2、线程敏感调度（thread－sensitive scheduling）可以解决上述问题，其流程如下图：



当一个核内的两个线程中的一个在一段时间空先后，就会用 H_CEDE 告诉系统管理程序，它的状态是空闲的，如果另一条线程活跃，则系统就会让空闲的线程休眠，处理器就会自然切换成 ST 模式，而当有中断到达或者有一个线程的 H_PROD 的目标为一个休眠线程时，

此线程会被激活。

4 POWER5 功耗的动态管理

在当今的 CMOS 工艺中，无论从耗电方面还是散热方面，芯片功耗都是一个重要的设计参数。功耗分为动态功耗和静态功耗。动态功耗主要是电路的翻转功耗，静态功耗主要来自漏电功耗。POWER5 采用了.13 微米工艺，随着工艺尺寸的缩小，漏电功耗的作用越来越不可忽略。其次由于 POWER5 的芯片 core area 的增加和采用 SMT 带来的更高的指令执行速率，翻转功耗也增大了。另外，POWER5 的芯片间的总线的翻转功耗也不可忽视。所有的这些导致在相同运行频率和工艺基础上，POWER5 总功耗和功耗密度的相对于 POWER4 的增加。下面就来介绍一下 POWER5 为了降低功耗和功耗密度采取的动态功耗管理策略。

4.1 静态功耗的管理

静态功耗包括 CMOS 晶体管的漏电功耗和一些模拟电路中的始终存在的功耗，一般只考虑晶体管的漏电功耗。

$$P_{static} = I_{leakage} V_{DD}$$

$I_{leakage}$ 是没有开关活动存在时在电源两条轨线之间流动的电流，漏电流一个重要的来源是晶体管的亚阈值电流，即晶体管的“关不断”电流。在.18 或.25 的工艺中，阈值电压比较大，晶体管的栅压为 0 时，电流趋近于 0。但是在更小尺寸的晶体管中，由于工艺的原因，阈值电压变小，产生“关不断”的现象，即栅压为零时，n 管和 p 管间仍有电流流动，这就是漏电流。

漏电流由三部分组成：A、CMOS 管亚阈值电压漏电流；B、CMOS 管栅级漏电流；C、CMOS 管衬底漏电流。亚阈漏电是指当晶体管关闭但输入电压仍存在时，晶体管中从源到漏依然存在的微小电流，它会随着晶体管阈值工作电压的降低而成指数倍增长。栅漏电是指在存在栅极到源、漏级电压差的情况下，由于栅氧厚度的变薄而导致的从栅到源、漏区的隧穿电流。反偏结漏电是指源漏区与衬底之间形成的反相 PN 结造成的漏电。这里主要考虑晶体管的亚阈值电流。

其中*Vt* 代表零偏置阈值电压，*VT* 代表热电压（Thermal Voltage），$\mu 0$ 代表电子迁移率，*Weff/Leff* 代表器件的有效宽长比，*Vgs, Vds, Vsb* 分别代表栅与源、源与漏以及源与衬底间的电压差

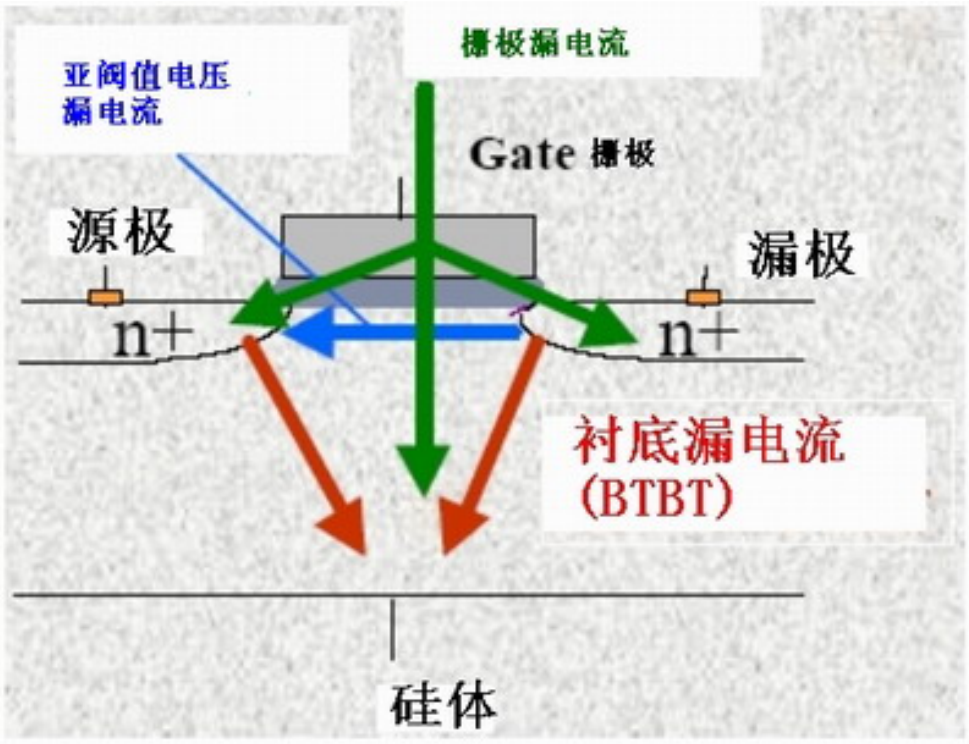


图 1　　晶体管中的漏电流

从上面的公式可以看出，为了减少静态功耗，有两个方法：减少V_{DD}和$I_{leakage}$。减少V_{DD}会降低晶体管的翻转速度，也就是会降低芯片的频率，所以要在性能和功耗之间找到一个平衡的点，不能单纯地降低V_{DD}。

最常用的方法是减少漏电流，常用的方法有两种：

$$I_{sub}=\mu_0\frac{w_{eff}}{L_{eff}}v_t^2\sqrt{\frac{q\varepsilon_s N_{cheff}}{2\phi_s}}\left(1-e^{\frac{v_{ds}}{v_T}}\right)\exp\left[\frac{\left(V_{gs}-V_t-\gamma V_{sb}+\eta V_{ds}\right)}{nV_T}\right]$$

（1）增大V_t——阈值电压

由公式分析知，V_t越大，晶体管关的越彻底，漏电流越小。

（2）减少C_{ox}——单位面积的栅氧电容

也就是增加栅氧层的厚度，降低栅极的隧穿电流，即减少图 1 中的栅极漏电流

域值电压越大，晶体管的反转速度越慢，芯片的频率也就越低。出于折中的考虑，POWER5 只在关键路径上采用低阈值器件，而 SRAM 阵列主要都是高阈值器件。具体来说，POWER5 中 30%的晶体管采用高阈值器件（主要用于存储阵列和非关键路径上）而只有 0.4%采用了低阈值器件（相对于 POWER4 的高于 7%的低阈值器件）。另外，POWER5 还采用了 IBM 的三倍阈值器件并增加栅的厚度使静态功耗降低了 40%。

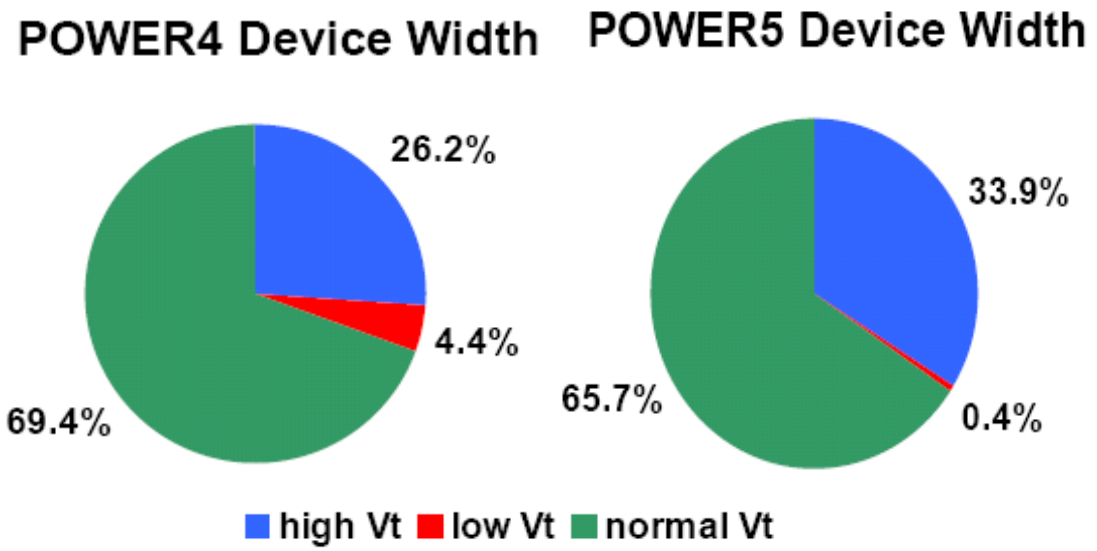


图 2　　POWER4 和 POWER5 中各种阈值器件的使用百分比

与 POWER4 相比，POWER5 的低阈值器件使用率减少了 90%，而高阈值器件的使用率增加了 30%。

4.2 动态功耗的管理

动态功耗主要由晶体管的充放电引起。负载电容 C_L 通过 P 管充电时，电压从 0 升到 V_{DD}，从电源吸取了一部分能量，该能量一部分消耗在 PMOS 器件中，其他则存在负载电容上。在由高至低的翻转期间，这一电容被放电，于是存放的能量被消耗在 N 管中。

$$P_{dyn} = C_L V_{DD}^2 f_{0\to1}$$　　$f_{0\to1}$ 代表消耗能量的翻转频率

常用的减少动态功耗的方法有：减少动态电路的使用，减少时钟网络的负载（减少负载电容），引进动态时钟门控。下面主要介绍动态时钟门控方法。　时钟门控就是不被使用的资源的时钟信号被关断来减少动态功耗，而不引起任何的 IPC 损失。在现代的高速微处理器中，大约 70%的动态功耗都消耗在时钟电路和对应的锁存器上。而主要的时钟功耗是在驱动锁存器的时钟树的叶结点上，所以，在底层的 clock buffer 上门控时钟信号会有效地减少动态功耗。

4.2.1 时钟门控电路

下面以一个简单的 Clock-gating 电路来说明：

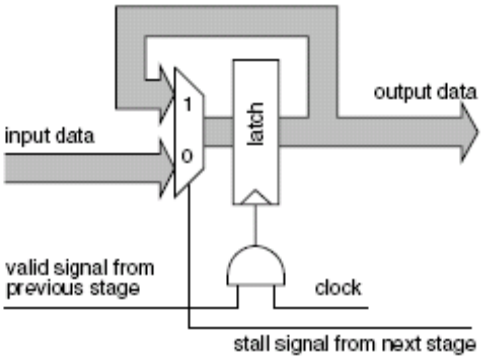


图 3　简单的 clock-gating 说明

latch 的时钟由 clock 和上一级发出的有效信号相与得到，只有前一级流水的 valid signal 有效即数据必须被保存的时候，锁存器才是时钟有效的，数据才能被锁存，否则就是透明的。其中在一些设计中，下一级流水的 stall signal 也可以作为时钟门控信号。如果下一级的数据被阻塞，多路开关选择输出数据再次锁存。在一些设计中，下一级流水的 stall signal 也可以作为时钟门控信号。

下图是 POWER5 中具体的时钟门控电路：

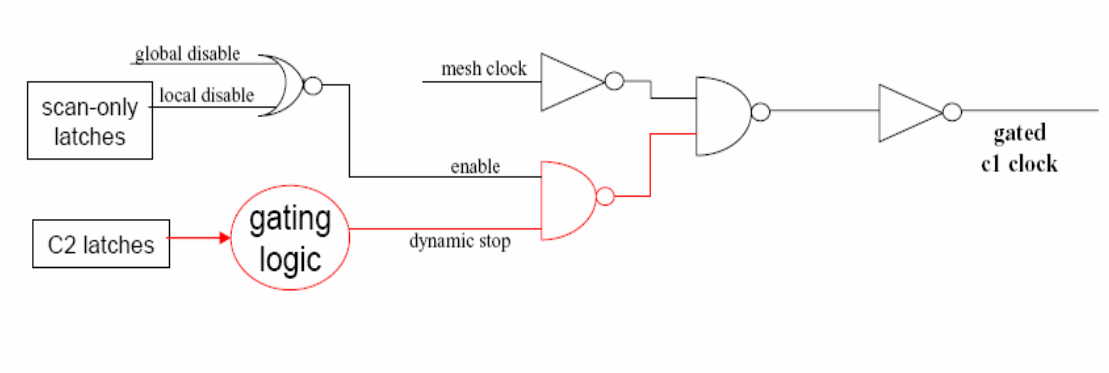
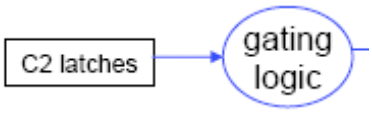
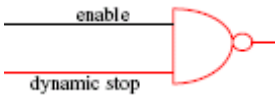


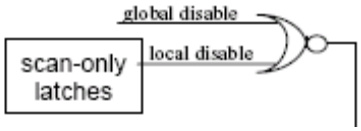
图 4　POWER5 中的 clock-gating



Cycle2 的 latches 产生 cycle1 的 latches 的门控时钟。



实际上的门控信号由两部分组成，即上一个 cycle 的 latches 经过门控逻辑产生的信号和非扫描状态的信号相与得到。



当在扫描测试时，会把时钟网络的 clock 禁掉，因为扫描测试的目的是检验 latch，所以要保证 latch 的时钟始终有效，不能禁掉。

4.2.2 门控结果

左面是没有采用始终门控的芯片温度，右面是采用了时钟门控的芯片温度，由两者的比较可以看出，时钟门控带来的收益是相当大的。

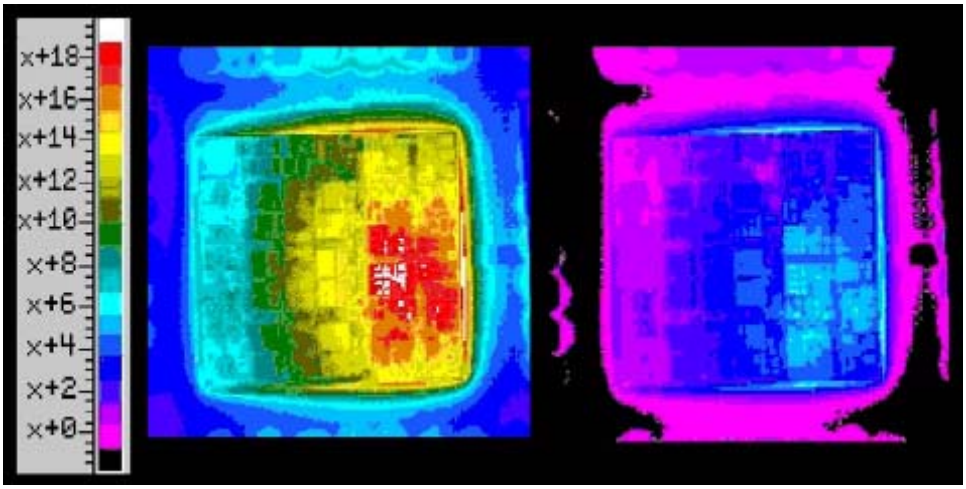


图 5 POWER5 处理器温度：未采用时钟门控 （左）和采用 时钟门控（右）

采用时钟门控减少了动态功耗和芯片温度。晶体管的静态漏电流和温度成指数关系，即温度越低，漏电流越小，所以采用时钟门控同时也减少了芯片的静态功耗。

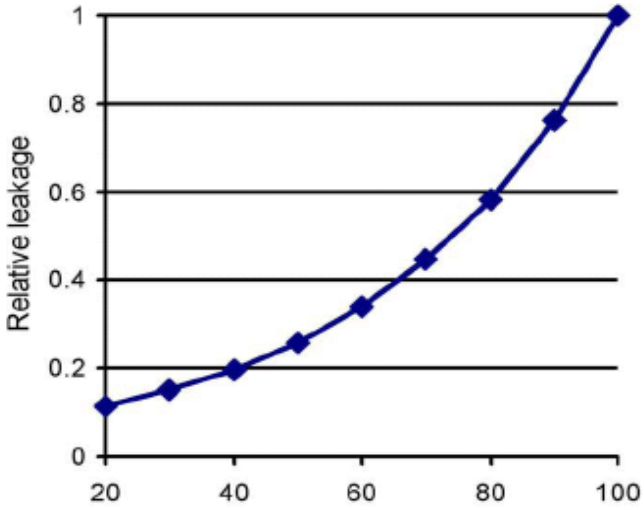


图 6 漏电流和温度的关系

4.2.3 局限性

时钟门控的主要局限性在时钟门控信号的时间问题和对 latches 的门控条件的分类。一些 latch 的组太小了，如果也考虑 clock-gating，增加设计复杂性和门控逻辑而外的功耗将得不偿失。由于连线延迟的问题， 根据 latch 输入的就近放置位置和配套门控逻辑的分组 latch 放置位置会产生矛盾。另外计算门控信号的逻辑会很复杂，Clock-gating 信号也可能会有很大的扇出。这些延迟会很难满足时序的要求。Clock-gating 会引起电压噪声，所以设计者加入去藕电容，这会增加漏电功耗，抵消一些 clock-gating 带来的好处。

4.3 热防护机制

Power5 中有 24 个数字温度传感器，每个传感器有对应一个环路震荡器，这个震荡器的频率是由温度控制的，震荡器控制计数器的计数，当温度过高时，震荡器的频率也会提高，然后计数器的值就会超过事先设定的域值，开始启动第一步热防护机制，在流水线中插入空拍来降低功耗，同时开始计数，如果在规定时间内温度仍然没有降下来，就启动第二步降温机制，通过减少一些功能部件的是用来降低处理器的吞吐量，待到降低功耗的目的。当温度降下来后，相应的震荡器的频率和计数器的计数也就降低了，到达域值后恢复正常。

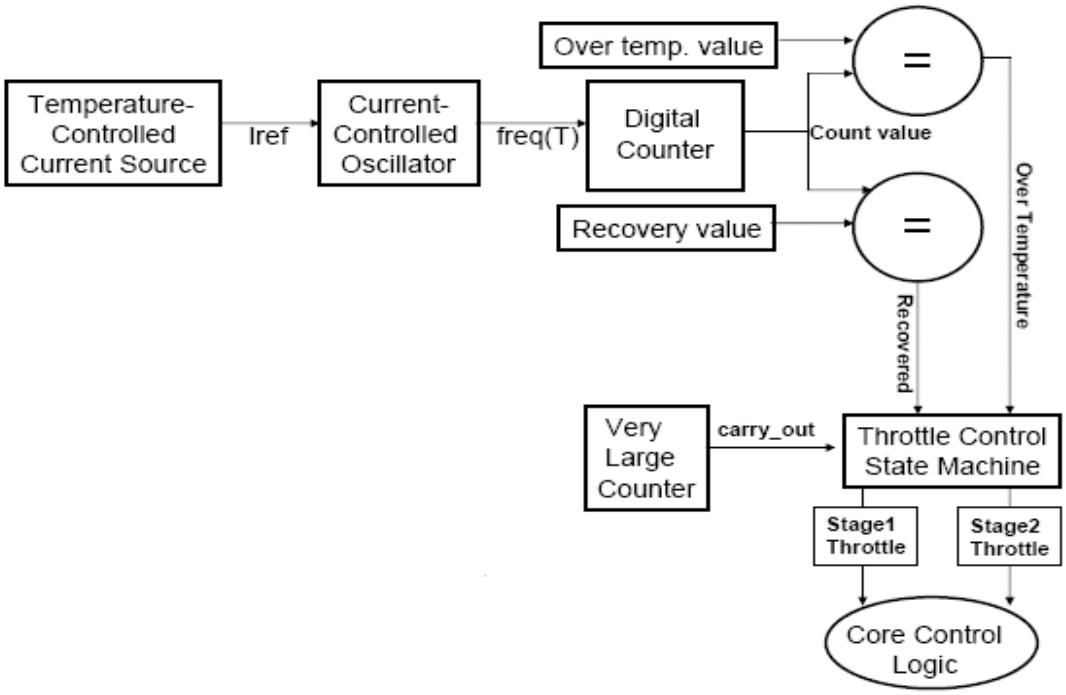


图 7　　POWER5 中的过热检测

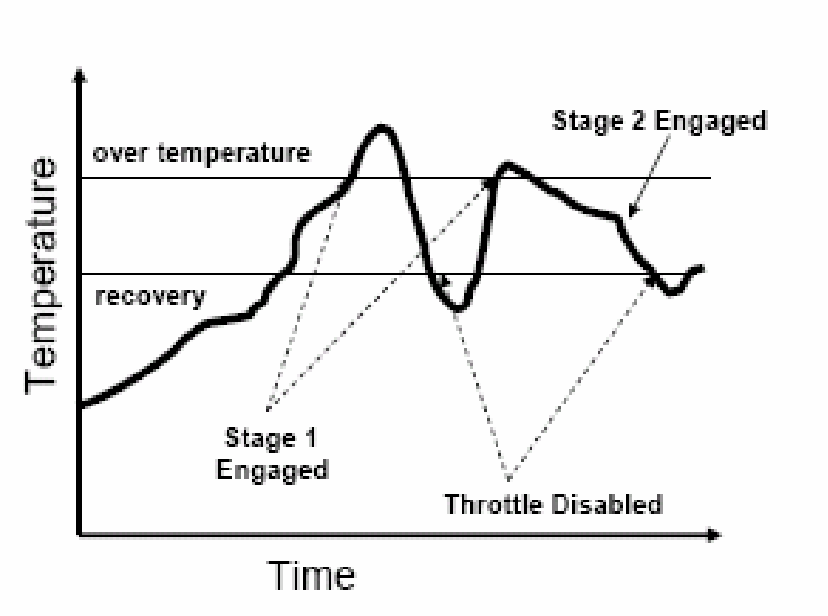


图 8　两步降温曲线图

下图是 POWER5 中指令队列单元温度过高时，中央控制逻辑采用第一步降温机制的温度变化情况。当器件的温度高于 84 摄氏度时，启动第一步降温，当温度降下来低于 81 摄氏度时，关闭热防护机制，周而复始。这里在一定时间内第一步都达到了降温的要求，就没有启用第二步。

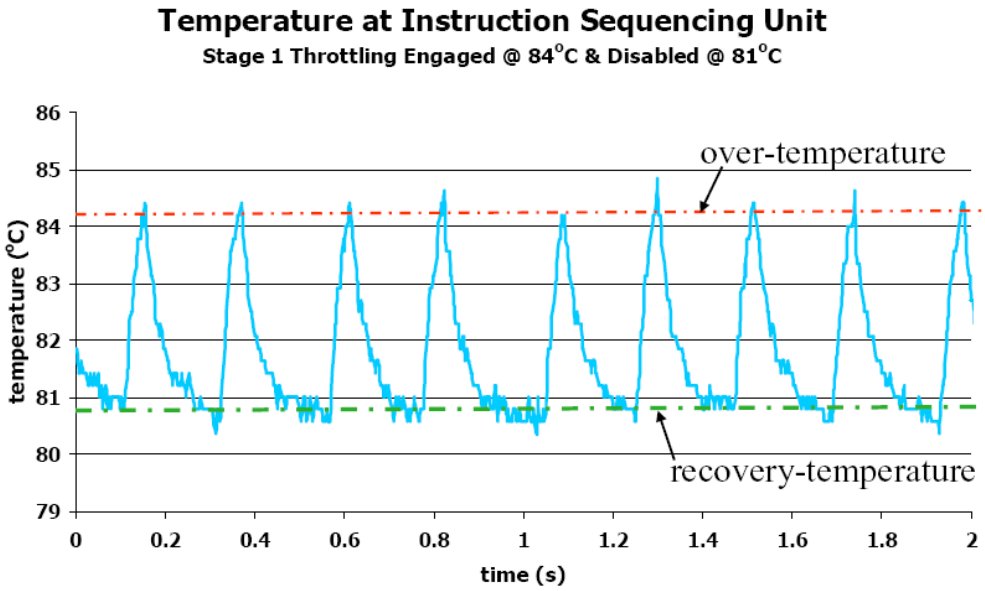


图 9　指令队列单元的温度

4.4 采用动态功耗管理策略后的改善

采用动态功耗管理策略后，动态功耗比没有相应措施减少了至少 25%，静态功耗比没有相应措减少了至少 50%，总功耗至少减少了 33%。

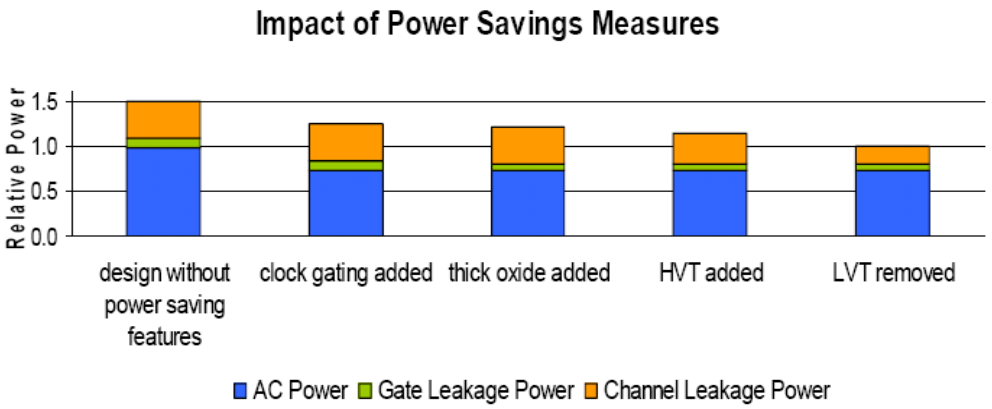


图 10　　POWER5 中 各种功耗管理策略带来的收益

由柱状图可以看出，采用 clock-gating，厚氧化层，增加高域值器件，减少低域值器件对动态和静态功耗都带来了好处。就单种方法而言，clock-gating 带来的收益最大。增加氧化层的厚度主要减少的是门的漏电功耗，加高域值器件，减少低域值器件主要减少的是沟道漏电功耗。

下图是 POWER5 处理器的温度用热敏感相机拍摄的结果，采用了动态功耗管理后，无论在单线程模式还是多线程模式，芯片的功耗都有了很大的改善。

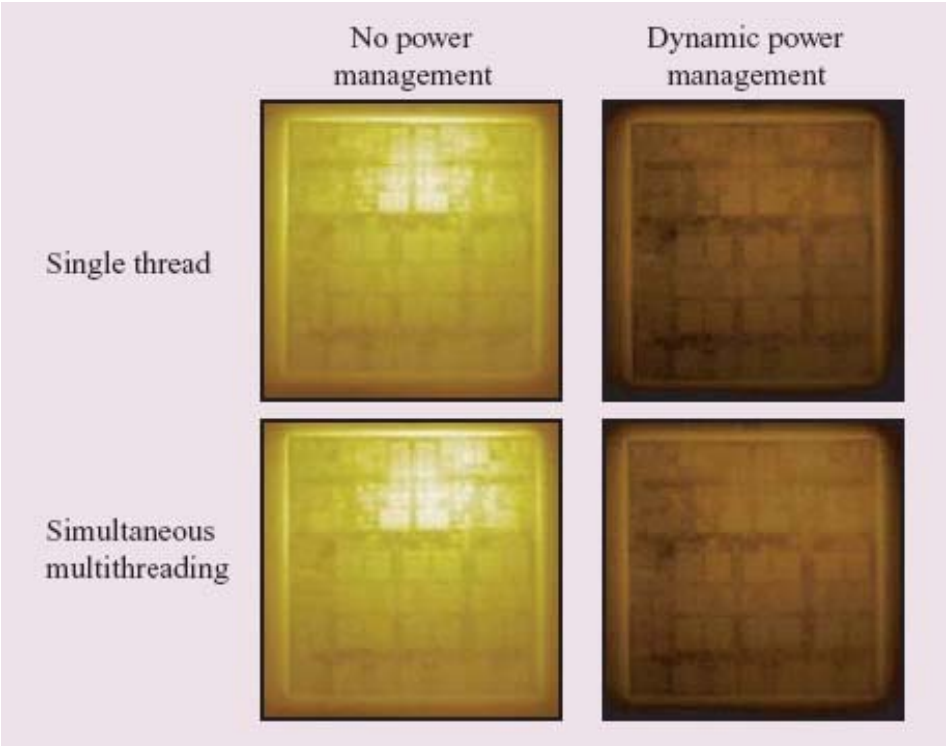


图 11　　热敏感相机拍摄结果

5 POWER5 性能评测

POWER5 最初的设计针对的是中高端商用服务器市场，采用了多线程技术后，具体得到了多大的性能提升，通过性能评测我们将得到答案。同时，详细地性能分析还能帮助我们分析出设计的每个部件局部的得与失。

5.1 测试环境

使用的是 IBM pSeries eServerP570 服务器，4 个 1.65GHz 的 POWER5 processors，使用 AIX special tools，内存为可变大小。

5.2 测试基准

使用八个基准测试程序：

SPEC SDM SDET 多用户环境下随机执行脚本
SPEC SFS97_R1 V3.0 网络文件系统服务器性能评测
SPECjbb 2000 评测 JAVA 服务器端性能
SPECjapp Sever 2004 评测 JAVA 企业级应用服务器性能
Netperf 评测多方面的网络性能
TCP_RR 应答机制
TCP STREAMING 系统带宽
SAP 模拟多用户和数据库交互
TPC-C 在线事务处理，模拟完整的计算环境

5.3 测试数据分析

5.3.1 吞吐率和 CPI 的变化

吞吐率在 SMT 模式下有普遍提升；TCP_RR 是用户应答性质的程序，故采用多线程会更有利于应答，减小 latency，所以它的吞吐率提升最突出

CPI 在采用了 SMT 技术后有所降低，但吞吐率的提升更快，因为有些程序有中断，这样的中断会影响 CPI，但是另一个线程的事务却是继续执行的；

5.3.2 SMT 技术对一级 cache 缺失率的影响

采用了 SMT 技术后所有程序的数据 cache 缺失率都有轻微的提高，但并不会很大；但有几个程序的指令 cache 缺失率有所下降，这是因为这几个程序都是指令密集型的，指令地址空间比较集中，能频繁地执行某一部分指令，当两个线程同时执行时，访存更频繁，将这些指令驻留在 cache 中，就能更有效地避免指令 cache 缺失。

5.3.3 SMT 技术对二级 cache 以及以下存储级的影响

数据在二级 cache 缺失后，在自身的三级 cache 中命中占相对较高的比例；而指令二级 cache 的缺失率就很少，而即使是有缺失的程序，也可以很方便地在自身地三级 cache 中命中；

因此我们可以得到如下结论：二级 cache 中取指令比取数据更有利于降低指令的 CPI. 三级 cache 的缺失率在 ST 模式下更低，这是由于 SMT 模式下双线程共享 cache，那么这必然会引起一定程度的容量缺失。但缺失率的增大程度是很小的，这于 SMT 所带来的性能提升相比是可以容忍的。

5.3.4 SMT 技术对 TLB 的影响

由于 TLB 的容量是不变的，当两个线程共享 TLB 表项时，引起 TLB 的缺失率增大也就是必然的了，但增大的幅度都很小，对于 SMT 来说这样的幅度是可以容忍的；TLB 指令缺失率总体上都很小，只有 SPECjapp 测试程序的指令缺失率较大，这是由于这个程序的性质就是访问的指令和数据的地址范围相隔很大，那么就会在 TLB 中不停地表项替换，TLB 缺失率也就大了；

但是 SPEC SFS 测试程序地 TLB 指令缺失率在 SMT 模式下反而有了大幅度地降低，这主要是由于 SFS 程序访问指令密集，分布范围小，频繁使用某地址范围内的指令，从而这些指令可以驻留在 TLB 中，从而采用 SMT 后的指令的增加更有利于这些指令的访问，所以 TLB 的指令缺失率在 SMT 模式下反而降低；

5.3.5 CPU 在 TLB 缺失后的处理时间

在 TLB 缺失后，CPU 去访问页表，各个程序的趋势和 TLB 缺失率的趋势是一致的；SMT 模式下访问页表的时间比 ST 下有所减小，这是因为 SMT 下双线程同时执行，当一条线程的指令 TLB 缺失后，CPU 去访问页表的同时，另外一个线程的指令可以继续访问 TLB，这样平均的 CPU 处理时间也就比 ST 下小了，所占总时间的比例也就小了。

5.3.6 SMT 技术对分支预测的影响

采用了 SMT 技术后，各测试程序的分支误预测率有所增大，因为 BHT 表并没有变，那么双线程共享时就必然会增加分支误预测率；虽然分支误预测率有所增大，但一个线程误预测回退时，另一个线程可以继续执行，这样分支误预测率的增大对性能的影响也就很小了。

5.3.7 已提交分组的平均指令数

采用了 SMT 技术后平均的指令数与 ST 模式下几乎没有变化；平均值为 2.3——2.75；这个指标对于衡量 CPI 并不是很准确，因为 POWER5 是按组提交的，某一个时钟周期内，还有某些 GROUP 没有提交，但是这些分组中已经完成的指令并没有计算在内；

5.3.8 GCT 的利用率

GCT 总的利用率都比较高，这也就反映了 CPU 资源的利用率都比较高；SMT 模式下 GCT 的利用率变化不大；GCT 中没有空闲 slot 以至于指令发射阻塞是由以下原因引起的：指令 cache 缺失、分支误预测、Store reorder queue 满。

6 讨论

6.1 课堂提问及解答

1、L3 移动的原因？
答：在 power5 中 L3 做 L2 的牺牲 cache，容量增大了，降低了 cache miss。最重要的是，在 power4 中 L3 和 memory controller，memory 在一条通路上。Power5 把 L3 分离走，消除了 L3 命中时对 core 和 memory 间的带宽的增加。
2、为什么不能通过单纯增加 core 的数量，而非通过 SMT 和 core 的结合来达到提高系统性能的目的？
答：core 内的复杂和 core 的数量的结合。单纯的单发射多核系统（T1）并不能充分利用 core 内的资源，并且对于紧密调度的线程来说，显然让它们在一个 core 内跑比在两个 core 内跑要好的多。
3、系统互联是怎么通信的？
答：通过 fabric controller 广播和环结构。
4、一个 core 逻辑分区怎么分的？
答：从大型机引入的逻辑动态分区，分区之间相互隔离开来。
5、讨论：SMT 不是长期的，CMP 才是长期的。举例，Alpha21464，八发射四线程。
6、时钟门控增加了硬件资源？
答：采用时钟门控，功耗减少了 30%-40%，相对于增加的硬件资源是划算的。一个时钟门控制一组 latch，而不是每个 latch 用一个时钟门控制。
7、采用时钟门控，除了面积上的开销，还给测试带来了负面影响（降低了覆盖率），给性能带来了什么影响?
答：如果延迟可控的话，不影响系统的性能。
8、降温方法的第一步，停顿流水线，给性能带来了什么负面影响？
答：降低了流水线的频率，使后续指令推迟执行，而不是停顿整个流水线。
9、影响静态功耗的因素之一降低阈值电压的局限性？
答：高阈值器件晶体管开关慢，影响主频。关键路径上多用低阈值器件，非关键路径上多用高阈值器件。
10、power5 的时钟哪来？
答：从外部的 pad 到锁相环，然后锁到高频。
11、(讨论)每个 partition 是一个逻辑 cpu，有些是专用的（物理 cpu 到逻辑 cpu 的映射是固定的），有些是 share 的（映射是动态的）。OS 和裸机之间通过 hypervisor 实现了虚拟化。
12、一个 core 上多个操作系统对资源的协调管理？
答：通过 hypervisor，类似系统调用。

13、affinity scheduling 的算法？
答：两种算法：一种针对 AIX；一种针对 linux
14、SMT 和 ST 是怎么切换的？
答：启动时；运行时通过命令
15、power5 相对于 power4 性能提高的原因？
答：不仅仅采用了 SMT，还有带宽增加这方面的原因
16、评测结果从何得来？
答：从内置的 monitor 监测到。
17、宏观多线程，微观单线程和 SMT 的比较？
答：前者通过软件实现 TLP，后者通过硬件实现 TLP。而软件开销显然比硬件开销大。
18、在关于运行模拟器得到 GCT 的合理设置的图中，横坐标 0-20 代表什么？这种比较说明了 SMT 的什么呢?
答：0-20 指 GCT 使用的项数。通过这个图，想说明 SMT 模式下 GCT 的结构与项数是合适的。
19、首先，GCT 指什么？SMT 模式下选择的 GCT 是多少？为什么是合适的?
答：GCT-Global Completion Table，主要用来标记 group 的发射顺序，从而能在提交阶段按顺序提交。Power5 和 power4 中，GCT 的大小并没有变化，为 20 项；组织方式由 FIFO 变为了链表。合适是因为，这是一个 tradeoff，GCT 实现开销是很大的，这个值是根据模拟的结果设置的一个合适的值。
20、针对热敏感相机拍摄的 4 个图中，对 power5 来说，对热最敏感的部位是什么？原因？
答：load/store 部件。因为，在乱序发射中，这个部件要保证内存的一致性，同时为解决三种相关：load hit load；load hit store；store hit load，还实现了相联查找；还从硬件上实现了预取机制。

6.2 其它问题的讨论

1、SMT 真的实现开销很小吗？

很多理论文章，包括 SMT 概念的提出者，都宣称 SMT 的实现只需要在动态超标量体系结构上做很少的改动。从 POWER5 的微体系结构来看，确实，几乎和 POWER4 没有大的改动。所以仅从这一点看，我们似乎比较乐观。

但是，SMT 的实现真的那么简单吗？通过分析 Alpha 21464 和 POWER5，以及 Xeon，我们会发现他们原有的超标量就已经做得很复杂。譬如它们的取指，它们的访存控制和优化，它们的大规模多端口寄存器文件，它们的大容量、高速的片上 cache 等。所以在这个基础上加入 SMT，可以很好的利用空闲的功能部件资源，因为前面因素的那些不会成为系统性能的瓶颈。

虽然有理论文章的结论指出，SMT 可以很好的容忍分支误预测的开销。但是我们对此总是有所疑虑。因为这些理论文章的结论建立在模拟器基础之上，它的准确与否取决于模拟器如何刻画实际情况，通常这种刻画建立在局部视角上，所以才会得到一些诸如 8 线程同步可以得到最高性能的结论。POWER5 中分支预测错的开销在 20 拍左右，那么如果像理论中所说的用已经进入流水的那些线程做弥补，那么从不同线程取指的策略就要有所变化，需要后续流水线的反馈信息，这样做仍旧使得取指部件的设计复杂度提升。除去取指阶段需要进行的分支预测，SMT 还需要很高的取值带宽，这也是伤脑筋的一个事情。

SMT 技术，说到底，仍旧是要扩大指令窗口并往里面填入更多有用的指令。这样势必导致集中控制部件设计复杂度提高和后端实现难度的加大。很难想象，如果 POWER5 仅仅

工作在 600-800MHz，它还有什么商业竞争力。现在 2E 是能跑到 1GHz 了，但是规模在那儿，如果做成 POWER5 那样，有难度。

2、POWER5 的性能提升，主要来自于 SMT 技术？

POWER5 相对于 POWER4 来说，性能又有了比较大的提升。但是这些好处仅仅是来自于 SMT 技术吗？有业内专业人士分析，POWER5 性能的提升主要来自于它的数据通路的改造，提高了近一倍的带宽同时大幅降低了延迟。对于商业应用吞吐量的需求，这样的调整无疑是相当明智的，好处更是显而易见的。

如果搞龙芯 3，要同曙光配套，结合问题 1 的分析，再根据我们现有的技术条件。也许我们需要全局地看问题。峰值性能，我们可以用更多的芯片提上去。我们有功耗和成本的优势，意味数量上我们可以考虑，而且功耗、散热等问题，我们应该也能应付。我们不需要拼单处理器绝对主频，相反，传输通路应该是我们考虑的重点。

3、放弃 SMT？

但是是不是就放弃 SMT 技术？如果放弃，也确实显得草率。作为开发 TLP 的一种方式，从目前看，这种方式仍是最好的。具体来看，2-线程比较现实，而且得到的好处也不少。问题是我们要不要扩单核的规模。我认为也可以不要。POWER5 的单线程模式和双线程模式的切换给了我们启发。从软件的特性出发，对于那些 SMT 拿不到好处的，就坚决地用 ST。软硬件协同，加强了硬件的适应能力。如果说可重构是一种最积极的提高硬件对程序特性适应能力的方式，那么这里这种简单化的做法也体现了这个主导思想。这个思想，在龙芯 3 的设计中就可以体现出来。如果在 2E 和将来 2F 的规模上 SMT 对绝大多数程序都没什么提高，那么完全可以不用 SMT 技术。如果还是有一部分的，那么还是可以考虑实现的。因为，从 POWER5 同 POWER4 的比较来看，不看 cache，增加的面积开销并不会很大。也就说，加入 SMT 技术，只要考虑妥当，龙芯 2 的短小精悍的特点还是可以保持的。

4、软件支持

这似乎一直就是一个问题，SMT 也罢，CMP 也好，软件的支持显得比以往任何时候都重要。可以说，遗留软件问题制约体系结构发展已经很多年了。直到现在，我们仍然要面对这个挑战。没有软件的支持，我们硬件做再大的努力，最终用户可以得到的好处微乎其微。相反，在恰当的地方引入软件有针对性地支持，不仅可以降低硬件设计复杂度，而且能获得很好的系统整体性能。