

支持超线程的动态逻辑处理器在 HP-UX 11i v3 中的应用



介绍	3
英特尔安腾 2 处理器超线程 (HT) 技术介绍.....	3
硬件线程处理模式	3
硬件切换事件	3
共享内存结构	4
逻辑处理器.....	4
LCPU 和超线程 (HT)	5
LCPU 和处理器集	5
超线程 (HT) 技术的应用考虑因素	6
一般商业应用程序的考虑因素	7
您可获得的优势可能存在差异：证明超线程 (HT) 技术可让应用程序受益	7
超线程 (HT) 技术支持实现更大吞吐率，而不是更高的响应时间	7
拥有较高处理器利用率的应用程序从超线程 (HT) 中获益.....	7
应用程序的扩展能力会影响超线程 (HT) 的性能优势	7
拥有精密高速缓存管理能力的应用程序可能不会从超线程 (HT) 中受益	7
CPI 低于 1.5 的应用程序可能不会从超线程 (HT) 中受益.....	7
只存在很少停顿性三级高速缓存未命中问题的应用程序可能不会从超线程 (HT) 中获益.....	8
进程只适合三级高速缓存的应用程序可能不会从超线程 (HT) 中受益.....	8
内存延迟较低的平台不会通过使用超线程 (HT) 明显获益.....	8
能够与平台配置相结合实现高总线利用率的应用程序可能不会从超线程 (HT) 中明显获益	8
TPC-C 类应用可以受益于超线程 (HT)	8
由同构进程组成的应用程序可以受益于超线程 (HT)	8
应用程序进程的一般组合或许可能从超线程 (HT) 中受益.....	8
使用一般 HP-UX 线程和锁定机制.....	8
HP-UX 能够同时满足启用或关闭超线程 (HT) 能力的要求	9
不要只凭简单性能指标评测行为得出性能结论	9
高性能技术计算应用程序的考虑因素	9
大多数机械应用程序不会受益或会受到负面影响	9
您可获得的优势可能存在差异：技术计算	9

启用超线程（HT）技术	9
在固件级启用	9
系统配置信息	10
默认 PSET 的 LCPU 属性设置	10
更改下一次启动时的超线程（HT）设置	11
PSET 操作	11
pset_create(2).....	11
pset_assign(2) 和 pset_destroy(2).....	11
pset_setattr(2).....	12
psrset(1M).....	13
对联机添加与删除操作的影响.....	13
计时	13
内核和 LCPU 利用率	15
可变内核利用率.....	15
每线程和流程利用率	15
在计时中对比吞吐率和准确率	15
计时延长.....	16
拓扑信息	16
pstat_getprocessor(2).....	16
pstat_getdynamic(2)	17
mpctl(2).....	17
构建全面拓扑信息的示例.....	18
pset_ctl(2)	18
更多信息	20

介绍

代号名为 **Montecito** 的新一代英特尔® 安腾® 2 处理器具备多项高级功能与增强特性, 包括每处理器多个内核和每个内核能够处理多个硬件线程等。本文档说明了为支持名为超线程 (HT) 技术的全新多硬件线程功能, HP-UX 11i v3 操作系统做出的重大改进。

英特尔安腾 2 处理器超线程 (HT) 技术介绍

Montecito 通过在每个插座 (芯片) 上安装两个处理器内核, 实现了芯片级多处理能力。每个处理器内核均可提供双路硬件多线程处理能力, 其中功能单元在两个硬件线程 (即两个独立的指令流) 间共享, 但处理器状态会被复制。因而, 每个插座均可提供独立的处理器状态例程, 支持执行四个不同的线程 (即每个插座支持两个内核, 每个内核又支持两个硬件线程)。由于复制的架构状态, 每个硬件线程对于操作系统而言都是一个完整的处理器。在单个处理器内核上提供多个硬件线程处理能力, 主要是为了在多个硬件线程间共享利用率较低的资源, 全面提高吞吐率和性能。在 **Montecito** 中, 有效利用因内存停顿导致的空闲周期和空闲功能单元可提高吞吐率。

硬件线程处理模式

业界当前实现硬件多线程处理的方法主要有两种: 同步多线程 (SMT) 和临时多线程 (TMT)。SMT 支持硬件线程环境同时共享处理器资源。TMT 支持硬件线程环境以固定的时间间隔共享处理器资源。**Montecito** 超线程 (HT) 技术集两种方法于一体, 使得硬件线程能够基于 SMT 方法共享内存分级结构, 同时基于 TMT 方法共享内核。**Montecito** TMT 方法还通过硬件控制得到了进一步增强。后者能够在高延迟事件 (即内存停顿) 可能导致硬件环境切换时, 硬件线程时间总量失效前, 将内核指派给相应的硬件线程。这种修改被称作基于事件切换的多线程处理 (SoEMT)。

硬件切换事件

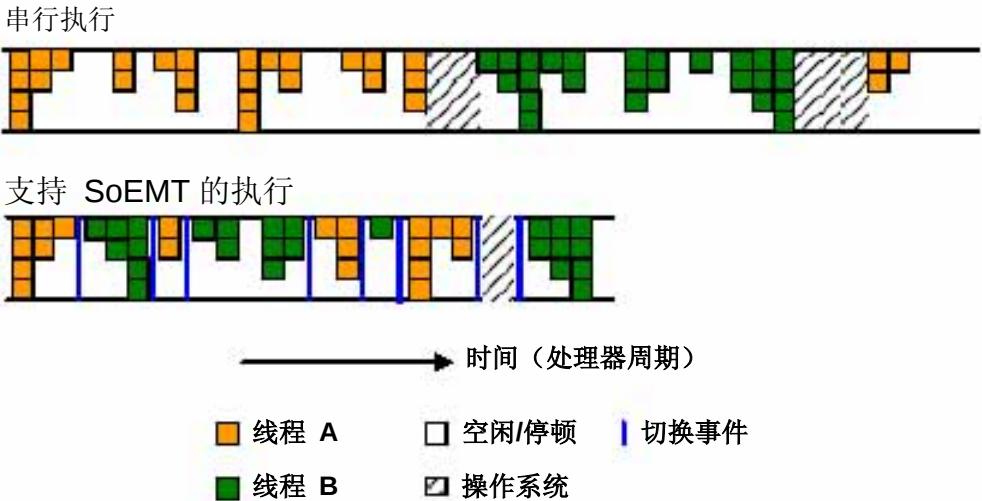
几种事件可能导致线程环境切换:

- 三级高速缓存未命中——前台线程引起的三级高速缓存未命中通常会导致内存停顿, 直至结构消除未命中问题。由于访问一级和二级高速缓存的延迟较低, 因而三级高速缓存未命中事件是主要的切换事件。
- 三级高速缓存未命中返回——三级高速缓存未命中返回事件说明后台硬件线程引发的三级高速缓存未命中故障已得到解决, 可以重新运行。根据硬件线程的紧急程度, 这一事件可能触发线程环境切换。
- 硬件超时——当线程时间总量失效时, 会发生线程环境切换, 这可以确保在后台和前台硬件线程之间保持公平。
- 软件切换提示——**Hint@pause** 指令用于将环境切换控制权交给软件。如可能, 前台线程将会把执行资源转给后台线程。通常在前台硬件不需要内核执行资源时, 会调用 **hint@pause** 指令。

- 节电模式——当前台线程进入节电模式时，内核资源将被转给后台线程。

此处的切换对于应用程序和操作系统而言完全透明。

图 1. 实例：内核上的非超线程（HT）执行与超线程（HT）执行比较。本例展示了这样一个场景，其中两个独立执行流的串行执行可同时在一個内核上进行，两个硬件线程之间的环境切换通过切换事件触发器来完成。



共享内存结构

Montecito 的 SMT 功能意味着内存资源在两个硬件线程间同时或并发进行共享。共享的内存资源包括：

- 一级和二级变换旁视缓冲器 (TLB)
- 一级、二级和三级指令与数据高速缓存
- 系统接口

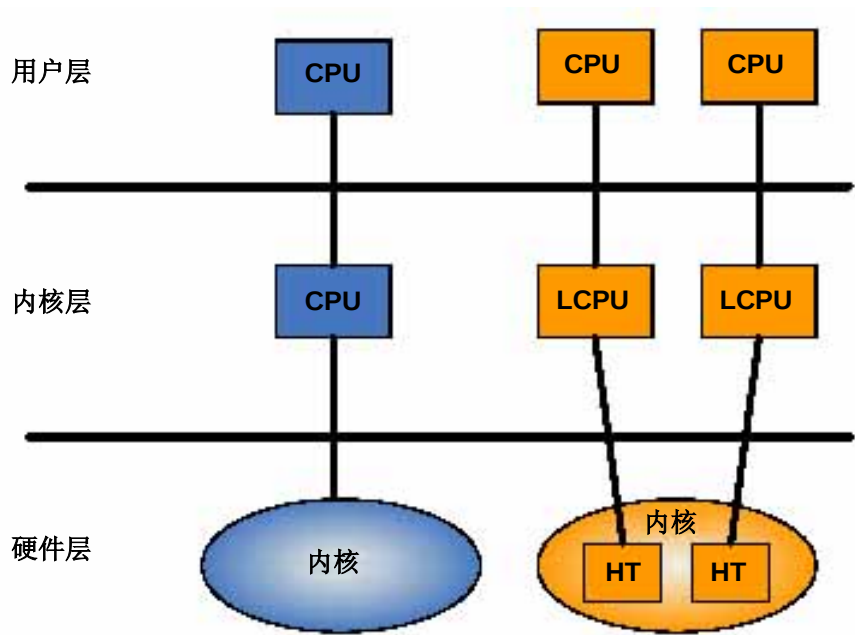
逻辑处理器

尽管功能单元在两个硬件线程间共享，但内核中的每个硬件线程都有着独立的处理器状态。也就是说，每个 Montecito 硬件线程对于应用程序和操作系统而言，都是一个独立的处理器。

在超线程（HT）技术推出之前，操作系统和应用程序始终可以使用全部处理器资源，而无需担心内核资源争用问题。然而，尽管功能性纠正和二进制兼容性可以得到保证，在内核上的两个硬件线程之间共享资源还是会产生其它影响。为解决问题，我们定义了名为逻辑处理器（LCPU）的软件抽象概念来代表 Montecito 硬件线程。

一个单线程内核被称作一个 CPU。除非被明确指定为硬件线程，否则线程通常是指软件执行流。同时，超线程（HT）和硬件线程可以交换使用。

图 2. 单线程处理器内核与多线程处理器内核比较



制定 LCPU 这一抽象概念非常必要，因为每一个硬件线程并非是一个真正的 CPU。尽管每一个硬件线程对于用户层应用程序而言类似于一个 CPU，但它们的作用并不完全与 CPU 相似。

LCPU 和超线程（HT）

在 HP-UX 11i v3 上，超线程（HT）技术可以应用于不同级别。Montecito 超线程（HT）技术可以通过固件设置在系统启动时启用或禁用。如果超线程（HT）技术被启用，操作系统将可以动态启用或禁用 LCPU 功能，以便内核线程能够在每一硬件线程上得到有序处理。

在超线程（HT）技术和 LCPU 功能都被打开时，内核上的两个硬件线程都可以得到利用。然而，如果超线程（HT）技术开启，但 LCPU 被禁用，内核上只有一个硬件线程可被利用，进而转变为一个单线程内核。LCPU 可通过使用 PAL_HALT_LIGHT 功能动态禁用。具体而言，这一调用可以通过停止指令执行，降低功耗和停止消耗内核资源，但它仍会保留高速缓存和 TLB 相关性，以应对外部中断请求。任意未被屏蔽的外部中断都会促使已停用的硬件线程恢复到正常状态，此时 LCPU 将被启用。

在操作系统内核中，每一个 LCPU 都代表着一个独立的运行队列，在该队列中的线程被安排在相应的硬件线程上运行。

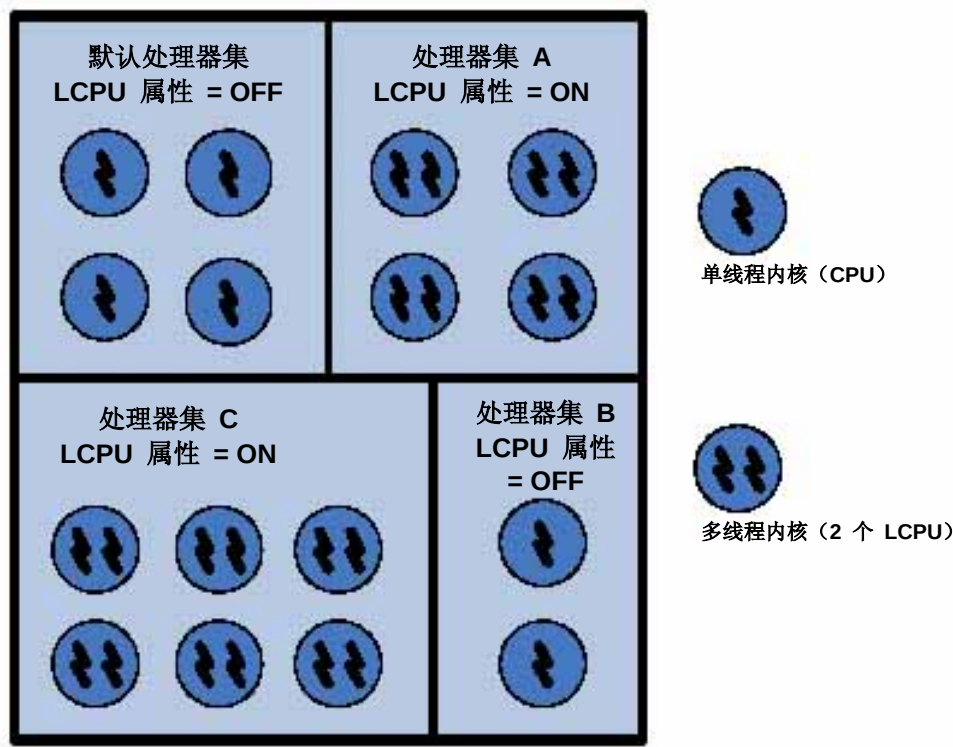
这一方法的主要优势在于，无论采用何种系统配置，所有供应用程序使用、并涉及到 CPU ID（例如 mpctl(2)、pstat(2)、pset_ctl(2) 等）的接口，都将逻辑 CPU 视作 CPU 对象。因而，现有应用程序将不会受到影响。

LCPU 和处理器集

尽管许多应用程序在超线程（HT）系统上能够很好地利用 LCPU 发挥出更卓越的性能，但也有一些应用程序可能无法利用这一优势，甚至有些应用程序的性能的确出现了下降。鉴于这一原因，我们将 LCPU 与 HP-UX 的现有分区技术进行了集成，亦即处理器集（PSET），以便能够支持在单个操作系统例程中同时使用 CPU 和 LCPU。

更具体地说，HP-UX PSET 的定义已进行了扩展，纳入了一个新的属性。这一属性允许在支持超线程（HT）技术的系统上，动态启用或禁用 PSET 中所有内核的 LCPU。HP-UX 客户可以根据运行在系统上的应用程序的需求来定制每一系统。在这一模式中，PSET 通过支持应用程序在不启用 LCPU 的处理器上运行，为无法与超线程（HT）技术有效配合工作的应用程序提供了一个“隔离区”。其它能够依托超线程（HT）技术发挥出更出色性能的应用程序，可以在启用了 LCPU 的 PSET 中运行。尽管这是 PSET 的一种新用途，但它通过使用现有和业经测试的机制，可以支持对超线程（HT）技术进行更精密的控制。此外，这一使用模式还支持对超线程（HT）处理进行控制，而无需对应用程序进行程序上的变化。这一方法具备全动态特性，可支持所有客户应用程序共存于一个操作系统例程中，而不考虑它们是否支持超线程（HT）。

图 3. 对带有多处理器集的单个操作系统例程进行分区，以支持 LCPU 功能被启用和关闭的组合



由于同一内核上的 LCPU 共享相同的资源，负责工作负载移植和 CPU 指派的接口需要能够在内核一级上进行精细操作。例如，如果一个 PSET 操作重新指派一个 LCPU，根据 LCPU 属性设置，它必须同时也重新指派该 LCPU 的另一半。否则，所产生的配置将会违背 PSET 协定，后者规定与 PSET 相关的应用程序拥有对该 PSET 内所有处理器的全部控制权。

这一配置协定同样也适用于其它操作，包括 iCAP、vPAR 和 OL*，要求在内核一级控制系统重配置。

超线程（HT）技术的应用考虑因素

尽管大多数应用程序能够受益于超线程（HT）技术，但也有一些应用程序可能出现性能下降的情况。

一般商业应用程序的考虑因素

通常，应用程序能够有效优化系统资源（例如，内存资源），但大多数商业应用程序在处理高速缓存访问模式时并不能做到全盘考虑。因而，应用程序（即使是已经优化的应用程序）通常会有 60% 或更多的时间处于停顿状态，等待通过将内存架构中某一位置的数据复制到高速缓存中，来解决高速缓存未命中问题。由于大多数应用程序实际上仅有不到 40% 的时间用来撤回指令，在一个内核中提供两个具备此类特征的线程将切实可行，并能够实现更高的利用率和吞吐率。

考虑到大多数应用程序进程较长的停顿时间，一些应用程序能够通过超线程（HT）技术实现切实的吞吐率优势。这一优势最高可达 15 到 20%，但更多情况下低于 10%。尽管超线程（HT）技术不太可能带来突飞猛进的优势，但其能够带来的潜在优势仍促使应用程序考虑使用。

您可获得的优势可能存在差异：证明超线程（HT）技术可让应用程序受益

对运行于具备超线程（HT）技术的系统上的应用程序进行分析，是判定应用程序性能表现的最有效方法。诸如 Caliper 等性能衡量工具可以帮助完成这一工作。此处我们需要确定超线程（HT）技术能否带来优势，而不是未经全面分析，便自动假定超线程（HT）技术能够让应用程序受益。

超线程（HT）技术可支持实现更大吞吐率，而不是更高的响应时间

从吞吐率的角度讲，启用超线程（HT）技术的内核比禁用超线程（HT）技术的内核要高出 30%。从严格的指令执行率角度讲，每个超线程（HT）可实现的执行率相当于不支持超线程（HT）的内核的 65%。如果应用程序具备充分利用更多线程执行以获得更高吞吐率的能力，则能够从超线程（HT）中受益。如果应用程序仅依赖于由处理速度决定的响应时间，则在含超线程（HT）的系统中会表现出较差的性能。

拥有较高处理器利用率的应用程序从超线程（HT）中获益

设计用于实现较高利用率，以便其它硬件线程能够带来更高吞吐率的应用程序，是应用超线程（HT）技术的主要选择。启用超线程（HT）技术类似于添加更多处理器。如果处理器利用率太低（低于 75-80%），应用程序可能无法从采用超线程（HT）中获得明显的吞吐率增加。

应用程序的扩展能力会影响超线程（HT）的性能优势

不能随增加的非线程 IPF 处理器内核（即不支持超线程的 Montecito）进行良好扩展的应用程序，通常会存在这一问题，主要原因是会受到资源共享瓶颈的困扰。此类应用将无法通过启用超线程（HT）能力获得任意性能优势。它们会将使用相同共享资源的同一内核上的超线程（HT），视作两个运行缓慢的 CPU，因而会进一步加剧某些可扩展性问题。

拥有精密高速缓存管理能力的应用程序可能不会从超线程（HT）中受益

能够精密管理高速缓存资源的商业应用程序，不会像普通应用程序一样在停顿上耗费大量的时间。此类应用程序在等待解决高速缓存未命中问题所花的时间较少，因而可能会增加线程争用内核资源的情况。

CPI 低于 1.5 的应用程序可能不会从超线程（HT）中受益

如果应用程序分析显示每指令周期数（CPI）低于 1.5，则超线程（HT）可能不会带来吞吐率增益。此类应用程序已实现了对内核资源的充分利用。为它们使用超线程（HT）只会增加线程争用内核资源的情况，进而可能导致性能下降。

只存在很少停顿性三级高速缓存未命中问题的应用程序可能不会从超线程（HT）中获益

如果应用程序分析显示只有很少的停顿性三级高速缓存未命中问题，则超线程（HT）可能不会带来任何吞吐率增加。此类应用程序可能不会在使用较大工作区时在一级和二级高速缓存之外引发未命中问题，因而不会触发充分利用超线程（HT）能力所需的三级高速缓存未命中事件。为它们使用超线程（HT）只可能增加线程争用内核资源的情况，进而可能导致性能下降。

进程只适合三级高速缓存的应用程序可能不会从超线程（HT）中受益

只适合三级高速缓存的进程在禁用超线程（HT）的情况下能够快速执行。在启用超线程（HT）时，使用内核上相邻硬件线程的两个此类进程可能会突然需要争用高速缓存资源，而这在以前根本不会出现。应用程序具备的有效利用高速缓存停顿时间的优势，将会很快被突然出现的高速缓存争用问题而抵消。

内存延迟较低的平台不会通过使用超线程（HT）明显获益

超线程（HT）通过指令执行利用三级高速缓存未命中停顿时间来提高吞吐率。低延迟平台只能提供较少的时间供利用，因为停顿只需较少的时间即可解决高速缓存未命中问题。因而，低延迟平台能够从超线程（HT）中获得的优势要低于高延迟平台。

能够与平台配置相结合实现高总线利用率的应用程序可能不会从超线程（HT）中明显获益

如果应用程序能够随高 CPI 和高停顿性三级高速缓存未命中率进行出色扩展，同时又在未启用超线程（HT）的情况实现了非常高的总线利用率，则不会从超线程（HT）中获得很大优势。尽管此类应用程序能够在某种程度上受益于超线程（HT）所能提供的额外计算能力，但对于业已饱和的总线所产生的压力可能会降低整体吞吐率。

TPC-C 类应用可以受益于超线程（HT）

工作行为类似于 TPC-C 性能指标评测的交易处理应用可以从超线程（HT）中受益。TPC-C 性能指标评测可通过使用超线程（HT）实现超过 20% 的性能改进。具备类似特征的应用程序，将可以通过正确组合 CPI 和停顿性三级高速缓存未命中，充分利用超线程（HT）能够带来的吞吐率增加。

由同构进程组成的应用程序可以受益于超线程（HT）

由同构进程组成的应用程序可以通过使用超线程（HT）获得较为显著的性能提升。它能够出色地利用处理器高速缓存资源，同时充分利用高速缓存未命中停顿时间来增加内核利用率，获得更高的吞吐率性能。然而，如果在一个内核上将高速缓存利用率非常高效的进程与非常低效的进程结合，将可以取得更高效效应。在一个内核上组合两个高速缓存利用率均非常高效（或低效）的进程，将会抵消超线程（HT）所能给应用程序带来的吞吐率增加。

应用程序进程的一般组合或许可能从超线程（HT）中受益

在高度利用的设备上，能够有效切合于处理器高速缓存，同时不需太多进程锁定或其它进程协调工作的应用程序进程组合，可以通过超线程（HT）使用获得一定的性能优势。

使用一般 HP-UX 线程和锁定机制

在同一个内核上执行的硬件线程之间存在着可能的交互。鉴于此，系统必须要对争用关键代码锁定的两个硬件线程进行协调，以避免任意可能损害应用程序性能的问题行为。标准 HP-UX 库线程与锁定机制已进行增强，用于有效协调硬件线程的行为。鉴于此，应用程序开发商应使用标准线程和锁定接口，并慎重考虑使用其自己的线程和锁定机制。

HP-UX 能够同时满足启用或关闭超线程（HT）能力的要求

一些应用程序可能存在一些能够充分利用启用超线程（HT）情况的进程，同时存在一些在禁用超线程（HT）的情况下表现出色的线程。HP-UX 中的 LCPU 模式支持此类应用程序在启用超线程（HT）的 PSET 中运行能够充分利用超线程（HT）的进程，同时在禁用超线程（HT）的 PSET 中运行不能充分利用 HT 的进程。这一超线程（HT）启用与禁用的组合可支持应用程序最大限度地提高超线程（HT）可带来的性能优势。

不要只凭简单性能指标评测行为得出性能结论

构建一个能够显示硬件线程表现不佳的人造代码流非常容易。然而，此类人造性能指标评测并不能完全代表商业应用程序，也不能作为判断超线程（HT）能否为给定应用程序带来优势的根据。

高性能技术计算应用程序的考虑因素

上述针对商业应用程序的建议同样也适用于高性能技术计算模式。然而，高性能技术应用程序通常会为包括高速缓存在内的整个内存结构优化访问模式，这意味着在线程时间量程结束时，伴随有线程切换只会出现较少的停顿时间。对于此类应用，一个小的（但不能是没有消耗）线程切换资源消耗就会导致在启用超线程（HT）的情况下出现性能损失。

大多数机械应用程序不会受益或会受到负面影响

大多数高性能技术应用程序（例如 LINPACK）能够很灵敏地识别高速缓存动态，并进行了调试以实现低 CPI。在许多此类应用程序上，停顿性三级高速缓存未命中情况非常稀少，以至于超线程（HT）切换资源消耗成为了一个重要因素，将会导致在启用超线程（HT）的情况下表现出较差的性能。此外，许多应用程序还针对具体的高速缓存大小进行了调试，因此受超线程（HT）共享资源的影响，高速缓存争用将会对它们造成明显延迟。

您可获得的优势可能存在差异：技术计算

如同商业应用程序一样，对运行于具备超线程（HT）技术的系统上的应用程序进行分析，是判定应用程序性能表现的最有效方法。诸如 Caliper 等性能衡量工具可以帮助完成这一工作。

启用超线程（HT）技术

启用超线程（HT）技术可以通过多种设置来实现。固件设置控制着 HT 功能的物理设置，同时操作系统可调命令设置控制着操作系统例程的 LCPU 功能。

在固件级启用

在可扩展固件接口（EFI）外壳，您可以配置系统以启用或禁用超线程（HT）技术。超线程（HT）设置会影响到物理分区中的所有处理器。因此，如果系统中配置有多个分区，您将可以为每一物理分区应用不同的设置。

表 1. HT 功能设置

固件设置说明	命令
禁用超线程（HT）功能	cpuconfig threads off
启用超线程（HT）功能	cpuconfig threads on

系统配置信息

新增的两个 `sysconf(2)` 选项可以支持应用程序开发人员查询系统的超线程（HT）功能，以及超线程（HT）功能设置的当前状态。

表 2. 使用 `sysconf(2)` 选项查询超线程（HT）功能

sysconf(2) 选项	说明
<code>_SC_HT_CAPABLE</code>	判断系统是否能够支持启用超线程（HT）功能
<code>_SC_HT_ENABLED</code>	判断系统当前是否启用了超线程（HT）功能

这一查询操作也可通过 `getconf(1)` 命令完成。

默认 PSET 的 LCPU 属性设置

从 11i V2 版本起，PSET 就成为了核心的 HP-UX 功能。系统启动时，它会被配置为一个 PSET，称作默认 PSET，其中包含了系统中的所有处理器。尽管用户可能从不会创建额外的 PSET，但用户会在无意中使用了 PSET 功能。

由于一些用户可能不会在其环境中使用 PSET 功能，同时希望在重启时保持 LCPU 属性，启用和禁用 LCPU 属性的工作通过可调操作来完成。

在新的版本中，我们引入了一个全新的可调命令 `lcpu_attr(5)` 来动态启用或禁用默认 PSET 的 LCPU 属性，原因如下：

- 用户无需了解 PSET 功能。可调命令是一个用于修改系统设置的完善基础设施，如果用户不希望使用 PSET 功能，设置默认 PSET 的 LCPU 属性将会隐含地对整个系统产生影响。
- 它提供了一种机制，支持动态启用或禁用默认 PSET 的 LCPU 属性，而无需使用 PSET 系统调用或相关命令。传统行为规定 PSET 系统调用不能修改默认 LCPU 属性。
- 此外，它还支持通过重启保持一致的 LCPU 属性，以使用户无需在重启后重新设置默认的 PSET LCPU 属性。

表 3. `lcpu_attr(5)` 的可调命令

说明	kctune(1M) 命令
获取默认 PSET 的最新 LCPU 属性设置	<code>kctune lcpu_attr</code>
在默认 PSET 中启用 LCPU 功能	<code>kctune lcpu_attr=1</code>
在默认 PSET 中禁用 LCPU 功能	<code>kctune lcpu_attr=0</code>
将默认 PSET 中的 LCPU 功能设定为默认设置，即禁用。	<code>kctune lcpu_attr=Default</code>

下表显示了固件超线程（HT）功能设置与默认 PSET LCPU 属性设置的不同组合。

表 4. 初始超线程（HT）状态和默认 PSET LCPU 属性

		默认 PSET LCPU 属性设置	
		LCPU 禁用	LCPU 启用
超线程（HT） 固件设置	超线程（HT）功能禁用	✓	✗
	超线程（HT）功能启用	✓	✓

Montecito 系统在发运时均禁用超线程（HT）功能，同时禁用默认 PSET 的 LCPU 属性。HP-UX 11iv3 安装流程会启用超线程（HT）功能。由于在启用超线程（HT）功能且同时禁用默认 PSET LCPU 属性时的系统性能，与禁用超线程（HT）功能时的性能一致，因此这一设置将使用户可以配置极为灵活的环境。

如果系统启动时禁用超线程（HT）功能，默认 PSET 的 LCPU 属性将被默认设定为禁用设置，并在重启时保持不变。

更改下一次启动时的超线程（HT）设置

为了避免在 EFI 提示符下进行不必要的超线程（HT）功能设置操作，我们对 `setboot(1m)` 命令进行了增强，以用来在基于 Montecito 的平台上启用或禁用超线程（HT）功能。这一配置更改适用于以后的启动操作。该功能等同于在 EFI 外壳中配置超线程（HT）功能。

表 5. 初始超线程（HT）状态和默认 PSET LCPU 属性

setboot(1M) 命令说明	命令
在下次重启时启用超线程（HT）功能	<code>setboot -m 启用</code>
在下次重启时禁用超线程（HT）功能	<code>setboot -m 禁用</code>

PSET 操作

现有 PSET 相关系统调用的接口定义并未变动，但由于系统配置在内核一级进行，我们在使用 PSET 相关系统调用过程中发现了几种细微的行为。请参阅《处理器集—技术白皮书》了解有关 PSET 的更多信息。

pset_create(2)

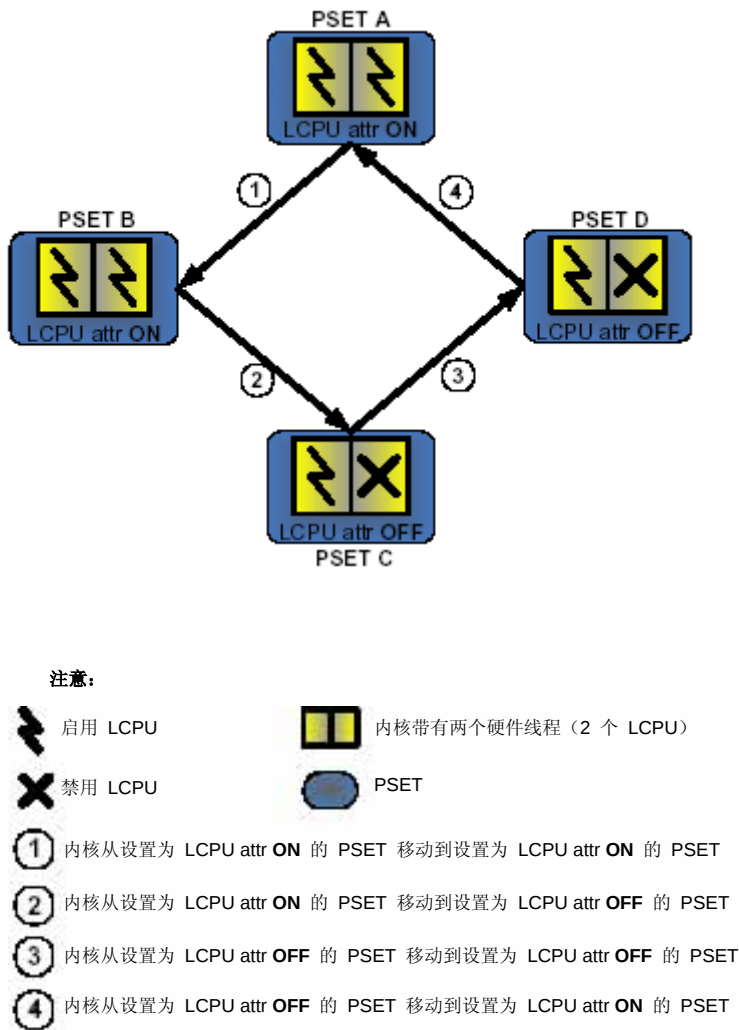
当使用 `pset_create(2)` 系统调用创建一个新的处理器集时，会继承默认 PSET 的 LCPU 属性。因此，如果创建时默认 PSET 的 LCPU 属性为：

- OFF——则新创建的 PSET 的 LCPU 属性被设定为 OFF（禁用）。
- ON——则新创建的 PSET 的 LCPU 属性被设定为 ON（启用）。

pset_assign(2) 和 pset_destroy(2)

内核的指派可通过两个 PSET 关联调用来完成：`pset_assign(2)` 和 `pset_destroy(2)`。`pset_assign(2)` 系统调用会将源 PSET 的一个内核移植到目标 PSET。然而，`pset_destroy(2)` 会将源 PSET 的一个内核移植到默认 PSET。根据源 PSET 和目标 PSET 的 LCPU 属性，CPU 和 LCPU 的拓扑视图可能会发生意料之外的变化。

图 4. PSET LCPU 属性及 CPU 和 LCPU 配置转换表



例如，重新将内核从 LCPU 属性为 ON 的 PSET A 指派给 LCPU 属性为 OFF 的 PSET D，将会导致在内核重新指派完成后，只有两个 LCPU 显示在 PSET D 中。

在其它实例中，默认 PSET 的 LCPU 属性被设置为 ON，LCPU 属性为 OFF 的 PSET 被取消。被取消 PSET 中的内核将会被重新分配给默认 PSET，同时启用所有 LCPU。

如果目标 PSET 和源 PSET 的 LCPU 属性相同，重新指派的内核的 LCPU 将不会发生变化。

pset_setattr(2)

我们在 PSET 功能中引入了一个新的属性 PSET_ATTR_LCPU，以在 PSET 边界处动态启用或禁用 LCPU 功能。非默认 PSET（或用户创建的 PSET）的 LCPU 属性可以使用 pset_setattr(2) 系统呼叫进行开关。

具备相应权限的调用方可以配置 PSET 来启用或禁用 LCPU 功能。注意，平台只能在启用超线程（HT）功能的情况下才能支持 LCPU 属性。

表 6. LCPUs_ATTR_LCPU 属性值

PSET_ATTR_LCPU values	说明
PSET_ATTRVAL_ON	这一属性可启用 PSET 中的 LCPU。该值是禁用超线程（HT）功能的系统的默认值。
PSET_ATTRVAL_OFF	这一属性可禁用 PSET 中的 LCPU。该值是不支持超线程（HT）功能或禁用超线程（HT）功能的系统的默认值。
PSET_ATTRVAL_DEFAULT	根据超线程（HT）功能的设置，默认值可能发生变化。

面向默认 PSET 中的 PSET_ATTR_LCPU 属性可被设定为使用 `settune(2)` 程序命令。

psrset(1M)

PSET 命令 `psrset(1M)` 进行了增强以支持新的 LCPU 属性。它可用来切换非默认 PSET 的 LCPU 属性。

表 7. 初始 HT 状态和默认 PSET LCPU 属性

setboot(1M) 命令说明	命令
启用目标 PSET 的 LCPU 属性	<code>psrset -t <pset id> LCPU=ON</code>
禁用目标 PSET 的 LCPU 属性	<code>psrset -t <pset id> LCPU=OFF</code>

要切换默认 PSET 的 LCPU 属性，需使用 `kctune(1M)` 命令。

对联机添加与删除操作的影响

具备超线程（HT）技术的系统将会对集中联机添加与删除（OL*）操作产生影响，其中包括：惠普即时增容（iCAP）、惠普虚拟分区（vPAR）和 CELL/CPU OL*。由于启用或禁用 LCPU 属性会为特定 PSET 的内核添加或删除一个硬件线程，因此这些操作会得到 PSET 操作的有序协调。

此外，OL* 操作在内核一级进行。例如，同一内核上的两个硬件线程不能分属于不同的 vPAR。

CPU 联机添加（OLA）会保留初始指派给默认 PSET 的传统行为。根据默认 PSET 的 LCPU 属性，新添加的内核可以是单线程或多线程。CELL OLA 被视作一个 CPU OLA 集。

CPU 联机删除（OLD）也会保留传统行为，并支持删除任意内核（采用 Monarch 的内核除外）。此外，如果一些 PSET 属性禁止出于内核重新分配或 CPU OLD 的原因移除处理器，CPU OLD 会无法进行。

如欲了解详细信息，请参阅有关 iCAP、vPAR 和 CELL/CPU OL 相关的手册与白皮书*。

计时

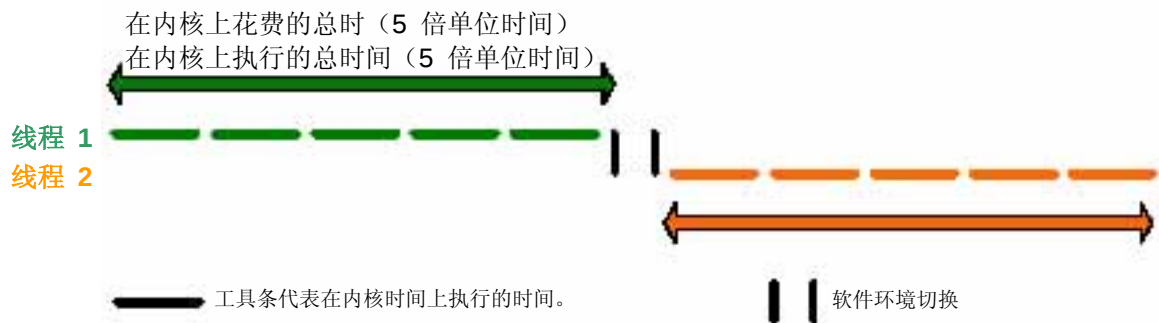
在超线程（HT）技术推出之前，单线程执行流会独占对核心资源的访问（包括机器周期），同时撤回指令或停止使用系统资源。因此，内核利用率可通过记录不同操作系统状态下的间隔计时器（ITC）来测量，例如用户、系统、空闲等。

对于操作系统在不同状态下进行测量的含义有两种基本观点：

- 内核上的执行时间（TeoC）——该方法包括指令流某时在任意两点间测量实际周期（通常为操作系统状态处理点），同时可以独占对内核的访问。
- 内核上花费时间（TsoC）——该方法包括某时测量指令流在任意两点间的单位周期内花费的时间（通常为操作系统状态切换点）。

以上两种测量都包括周期，同时撤回指令并停止使用系统资源。在单线程内核上，两种测量观点产生了相似的结果。无论处理器内核是主动撤回指令还是停止使用系统资源，执行流都不会放弃内核资源并且独占对内核的访问，直到操作系统强制将环境资源切换到另一个执行流。因此，通过读取 ITC 来测量内核利用率非常准确一致。

图 5. 单线程内核上的计时（CPU）



如图 7 所示，直到操作系统切换到其它指令环境，它才能保证当前的指令流独占内核资源。因此，TEoC 和 TSoC 的比例为 1:1。因此，采用哪种方法来测量内核利用率无关紧要。

然而，由于在 Montecito 系统上采用了超线程（HT）技术，内核成为了两个硬件线程间的共享资源，因此读取 ITC 的现有方法不能提供每个超线程（HT）技术准确的内核资源利用情况。间隔 ITC 是共享寄存器。ITC 仍在不断发展，而与当前内核上执行的硬件线程无关。

图 6. 超线程（HT）计时（两个 LCPU）。



由于硬件环境切换对于操作系统而言完全透明，通过读取 ITC 来测量硬件线程的内核资源利用率可以准确地测量 TsoC，但不能准确地测量 TeoC。

如图 8 所示，某时在任意两点间对 TSoC 和 TEoC 的测量是不同的。它们不再是 1:1 的比例，因为指令流不再独占访问内核资源。

在 HP-UX 11.31 v3 上，HP-UX 报告 TEoC 作为测量内核利用率的缺省视图。当启用超线程（HT）时，TSoC 方法还要考虑同一内核上相同硬件线程的内核利用率，这会干扰此设计，使超线程（HT）技术在应用程序和操作系统面前显示为一个独立的处理器。

然而，由于来自 ITC 的信息是唯一测量来源，操作系统要进行调整以反映 TEoC。此项调整被称为 50/50 近似算法。如果在同一内核上的相同超线程（HT）技术上调度相似或不同工作负载时，硬件线程调度程序会公平分配内核资源。因此，50/50 近似算法切实可行。

当相同超线程（HT）空闲时，它会紧密旋转，发出 hint@pause 指令。

50/50 近似算法适应这种情况并可相应进行调整。当一个超线程（HT）处理实用工作且其相同超线程空闲时，未空闲的超线程（HT）会利用 90 到 99% 的内核周期，因而相当于单线程内核的性能。

进行的多次试验都使用微性能指标评测和实际应用来确定 50/50 近似算法的可行性，结果都是 85 至 90% 甚至更高的准确性。

未来的安腾 2 芯片可能支持每超线程（HT）技术资源利用计算（RUC）。50/50 近似算法是 RUC 推出前的一种临时设计。当 RUC 推出后，将会取代 50/50 近似算法。

内核和 LCPU 利用率

区别内核利用率与 LCPU 利用率非常重要。一般来说，当 LCPU 得到充分利用后，内核也会得到充分利用。然而，当一个 LCPU 处理一项重要工作，而其同胞 LCPU 处于空闲状态时，内核可以得到充分利用，但 LCPU 却并非如此。空闲 LCPU 会主动发出 hint@pause 指令来将其内核资源分配给正在从事重要工作的同胞 LCPU。从而会出现这种情况：其中一个同胞 LCPU 处于空闲状态，但内核资源全部被处理重要工作的 LCPU 使用。较低的 LCPU 利用率一般会体现为较低的内核利用率。

可变内核利用率

LCPU 的计时和内核资源利用率不再是线性的。内核利用率会根据工作负载行为而有很大不同；计时会反映出这种行为，以便支持做出未来架构增强，例如使用 RUC。

每线程和流程利用率

线程和流程利用率测量采用 TEoC 来反映实际内核资源利用率。虚拟和设定计时器也依赖于 TEoC。

在计时中对比吞吐率和准确率

采用超线程（HT）技术的 Montecito 旨在最大限度地减少未充分利用的内核资源，从而提高吞吐率。然而，在特定环境下，保持准确的计时比提高吞吐率更为重要。超线程（HT）环境下 50/50 的近似利用率模式可提供合理的准确度，但是如果要求绝对的资源利用率测量，则惠普建议禁用超线程（HT）技术或禁用 LCPU 属性。

计时延长

以下字段被添加到 `pstat_getprocessor(2)` 界面。使用现有的字段（报告 TEoC）和新字段（报告 TSoC），可获得资源利用率。

表 8. `pstat_getprocessor` 计时延长。

字段	字段说明
<code>psp_raw_usercycles</code>	用于用户模式执行的 64 位周期计时器。此不可调节的计时器代表在内核上花费的时间。
<code>psp_raw_systemcycles</code>	用于系统模式执行的 64 位周期计时器。此不可调节的计时器代表在内核上花费的时间。
<code>psp_raw_interruptcycles</code>	用于中断模式执行的 64 位周期计时器。此不可调节的计时器代表在内核上花费的时间。
<code>psp_raw_idlecycles</code>	用于空闲模式执行的 64 位周期计时器。此不可调节的计时器代表在内核上花费的时间。
<code>psp_logical_cpu_idlecycles</code>	64 位周期计时器用于测量在空闲时每 LCPU 花费的挂钟时间。这是 LCPU 将在空闲周期内执行的运行中的总计时器，但由于来自空闲 LCPU 的 <code>hint@pause</code> 指令，它可能会将大部分周期分配给处理重要工作负载的同胞 LCPU。使用该计时器，可以计算内核和同一内核上的 LCPU 的利用率。

拓扑信息

旨在获得系统和 PSET 拓扑信息的现有接口正在扩展以提供有关分级系统配置的完整信息。

`pstat_getprocessor(2)`

`pst_processor` 数据结构中的其它字段正是 LCPU、CPU、内核、插座和未知域（LDOM）的分级体现。由于 `pstat_getprocessor(2)` 命令会返回 CPU 或 LCPU 粒度级的信息，根据 PSET 中的指定处理器是启用还是关闭 LCPU 属性，您可能需要一种可能的方法来提供有关指定处理器是 CPU 还是 LCPU 的信息：

```
struct pst_processor psp;
pset_attrval_t attr;

ret = pstat_getprocessor(&psp, sizeof(struct pst_processor), 1, 4);
if (ret == -1) {
    perror("pstat_getprocessor() failed\n");
} else {
    ret = pset_getattr(psp.pst_pset_id, PSET_ATTR_LCPU, &attr);
    if (ret != -1) {
        if (attr == PSET_ATTRVAL_ON) {
            printf("spu 4 is LCPU\n");
        } else {
            printf("spu 4 is CPU \n");
        }
    }
} else {
    perror("pset_getattr(PSET_ATTR_LCPU failed\n");
}
}
```

表 9. pstat_processor(2) 字段和说明

字段	字段说明
psp_socket_id	CPU 或 LCPU 的插座 ID
psp_core_id	CPU 或 LCPU 的 内核 ID
psp_logical_thread_id	CPU 或 LCPU 的逻辑线程 ID
psp_sibling_cnt	在支持超线程（HT）的机器上，返回同一内核上的同胞硬件线程的数量。该数量不包括自身，因此在禁用超线程（HT）的系统上，返回的值为 0。
psp_sibling[PSP_MAX_SIBLINGS]	在支持超线程（HT）的机器上，返回一个 LCPU ID 阵列用于同一内核上的同胞硬件。阵列中的无效输入会返回“-1”。

pstat_getdynamic(2)

添加新的系统范围计数器，代表活动内核的总数以及系统支持的最多内核数。

表 10. pstat_getdynamic(2) 字段和说明

字段	字段说明
psd_active_cores	当前系统中活动内核的数量。
psd_max_cores	系统上可支持的内核数。

mpctl(2)

mpctl(2) 命令的拓扑信息部分进行了扩展，可提供有关 CPU、LCPU、内核和 LDOM 的分级信息。

表 11. mpctl(2) 系统呼叫选项，用于查找处理器内核拓扑信息。

字段	字段说明
MPC_GETNUMCORES_SYS	返回系统中启用的/活动处理器内核的数量。
MPC_GETFIRSTCORE_SYS	返回系统中初次启用的处理器内核的 ID。
MPC_GETNEXTCORE_SYS	在指定处理器内核 ID 之后，返回系统中下一次启用的处理器的 ID。
MPC_GETCURRENTCORE	返回调用线程的处理器内核 ID。
MPC_SPUTOCORE	返回包含指定 CPU 或 LCPU ID 的处理器内核的 ID。
MPC_GETNUMCORES	返回调用线程的 PSET 中处理器内核的数量。
MPC_GETFIRSTCORE	返回调用线程的 PSET 中第一个处理器内核的 ID。
MPC_GETNEXTCORE	在指定 CPU 或 LCPU ID 之后，返回调用线程的 PSET 中处理器内核的 ID。

字段	字段说明
MPC_LDOMCORES_SYS	返回指定位置域中启用的/活动处理器内核的数量。
MPC_LDOMCORES	返回位置域中指定给 PSET 的启用的/活动处理器内核的数量。

与 `pstat_getprocessor(2)` 类似，处理器级拓扑信息的缺省粒度是 CPU 或 LCPU。目前有多种不同的方法可生成拓扑信息，但这两例最常用于确定系统的配置。

构建全面拓扑信息的示例

范例程序在单一单元系统上执行，该系统具有八个内核和 16 个超线程（HT）。

两个 PSET（缺省 PSET 和用户创建的 PSET）配置有四个内核，分别指定给每个 PSET。

```

spu_t spu;
spu_t core;
ldom_t ldom;
pset_attrval_t attr;
psetid_t pset;
int ret;

spu = mpctl(MPC_GETFIRSTSPU_SYS, 0, 0);
while (spu != (spu_t)-1) {
    pset = pset_ctl(PSET_SPUTOPSET, 0, spu);
    ret = pset_getattr(pset, PSET_ATTR_LCPU, &attr);
    core = mpctl(MPC_SPUTOCORE, spu, 0);
    ldom = mpctl(MPC_SPUTOLDOM, spu, 0);

    printf("%s %d in core %d assigned to pset %d ",
           "(lcpu_attr %s) in ldom %d\n",
           (attr == PSET_ATTRVAL_ON) ? "LCPU" : "CPU",
           spu, core, pset,
           (attr == PSET_ATTRVAL_ON) ? "ON" : "OFF", ldom);

    spu = mpctl(MPC_GETNEXTSPU_SYS, spu, 0);
}

```

代码片断的范例输出：

```

LCPU 0 in core 0 assigned to pset 0 (lcpu_attr ON) in ldom 0
LCPU 1 in core 0 assigned to pset 0 (lcpu_attr ON) in ldom 0
LCPU 2 in core 2 assigned to pset 0 (lcpu_attr ON) in ldom 0
LCPU 3 in core 2 assigned to pset 0 (lcpu_attr ON) in ldom 0
LCPU 4 in core 4 assigned to pset 0 (lcpu_attr ON) in ldom 0
LCPU 5 in core 4 assigned to pset 0 (lcpu_attr ON) in ldom 0
LCPU 6 in core 6 assigned to pset 0 (lcpu_attr ON) in ldom 0
LCPU 7 in core 6 assigned to pset 0 (lcpu_attr ON) in ldom 0
CPU 8 in core 8 assigned to pset 2 (lcpu_attr OFF) in ldom 0
CPU 10 in core 10 assigned to pset 2 (lcpu_attr OFF) in ldom 0
CPU 12 in core 12 assigned to pset 2 (lcpu_attr OFF) in ldom 0
CPU 14 in core 14 assigned to pset 2 (lcpu_attr OFF) in ldom 0

```

pset_ctl(2)

`pset_ctl(2)` 命令的拓扑信息部分提供了有关指定 PSET 的 CPU、LCPU、内核和 LDOM 的分级信息。`pset_ctl(2)` 和 `mpctl(2)` 命令提供的这一信息对于指定 PSET 来说很相似。

表 12. pset_ctl(2) 系统呼叫选项，用于查找处理器内核拓扑信息。

字段	字段说明
PSET_GETNUMCORES	返回调用线程的 PSET 中内核的数量。
PSET_GETFIRSTCORE	返回指定 PSET 中第一个处理器内核的 ID。
PSET_GETNEXTCORE	在指定 CPU 或 LCPU ID 之后，返回指定 PSET 中下一个处理器内核的 ID。
PSET_LDOMCORES	返回位置域中分配给指定 PSET 的启用的/活动处理器内核的数量。

更多信息

- <http://www.hp.com/go/hpux11i>
- 请参阅 <http://docs.hp.com> 上的“HP-UX 处理器集 – 技术白皮书”和 <http://www.intel.com/design/itanium2/manuals/308065.htm> 上的《英特尔® 安腾® 2 处理器软件开发与优化参考手册双核更新版》。

© 2006 Hewlett-Packard Development Company, L.P. 本文所含信息如有更改，恕不另行通知。惠普产品与服务的全部保修内容在此类产品和服务随附的保修单中明确说明。本文中的内容信息不得视为任何附加的保证。惠普对本文中所包含的技术、编辑的错误或遗漏恕不负责。

Intel 和 Itanium 是 Intel Corporation 或其子公司在美国和其他国家/地区的商标或注册商标。

4AA0-7695ENW, 2006 年 10 月。

