超线程加快Linux操作系统的速度（下）（4） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E8_B6_85_E7_BA_BF_E7_A8_8B_E5_c103_144854.htm 除了 2.5 调度程序中运行队列的更改之外，还要进行其它必要的更改，以使 Linux 内核能够利用 HT 达到最佳性能。Molnar 讨论过的那些更改（请再次参阅参考资料以获取有关此内容的更多信息）如下所示。 支持 HT 的被动的负载均衡： 用 IRQ 驱动的均衡操作必须针对各个物理 CPU，而不是各个逻辑 CPU。否则，可能会发生：一个物理 CPU 运行两个任务，而另一个物理 CPU 不运行任务；现有的调度程序不会将这种情形认为是“ 失衡的 ”。在调度程序看来，似乎是第一个物理处理器上的两个 CPU 运行 1-1 任务，而第二个物理处理器上的两个 CPU 运行 0-0 任务。现有的调度程序没有意识到这两个逻辑 CPU 属于同一个物理 CPU。“ 主动的 ” 负载均衡： 当一个逻辑 CPU 变成空闲，从而造成一个物理 CPU 失衡时，会出现这种情况。只有在现有的 1:1 调度程序中才不会出现这个机制。由空闲 CPU 引起的失衡可以通过常见的负载均衡器来解决。在使用 HT 的情况下，这种情形很特殊，因为源物理 CPU 可能只有两个任务在运行，而两个都可以运行。现有的负载均衡器不能处理这种情形，因为正在运行的任务难以迁移。而这个迁移是必需的 — 否则一个物理 CPU 费力地运行两个任务，而另一个物理 CPU 却保持空闲。 支持 HT 的任务挑选： 当调度程序挑选了一个新任务时，它在尝试从其它 CPU 接收任务之前，应该优先挑选所有共享同一物理 CPU 的任务。现有的调度程序只挑选那些被调度到特定逻辑 CPU 的任务。 支持 HT

的亲缘性： 这些任务应当设法 “ 盯牢 ” 物理 CPU，而不是逻辑 CPU。 支持 HT 的唤醒： 现有的调度程序只知道 “ 当前 ” CPU，不知道其任何 “ 伙伴 ” CPU。在 HT 上，如果一个逻辑 CPU 正在执行任务，其上的一个线程被唤醒了，而且其 “ 伙伴 ” CPU 是空闲的，那么该 “ 伙伴 ” CPU 必须被唤醒以立即执行刚唤醒的任务。 在撰写本文时，Molnar 已经提供了现有的内核 2.5.32 补丁程序，它通过引入共享运行队列（多个 CPU 可以共享同一个运行队列）的概念来实现上述所有更改。共享的、针对每个物理 CPU 的运行队列实现了上面所列的所有 HT 调度操作的需要。显然，这使调度和负载均衡变得复杂了，而且这对 SMP 和单处理器调度程序的影响仍然是未知的。 Linux 内核 2.5.32 中的更改旨在对带有两个以上 CPU 的 Xeon 系统产生影响，尤其是在负载均衡和线程亲缘性这些区域方面。由于硬件资源限制，我们只能测量其在我们单 CPU 测试环境中的影响。使用 2.4.19 中所使用的相同测试进程，我们在 2.5.32 上运行了三个工作负载：chat、dbench 和 tbench。对于 chat 而言，在有 40 个聊天室的情况下，HT 可以将速度提高 60%。整体提高幅度大约为 45%。对于 dbench 而言，27% 是最高的提高幅度，整体提高幅度大约为 12%。对于 tbench 而言，整体提高幅度大约为 35%。 表 7. 超线程对 Linux 内核 2.5.32 的影响 