附录四 Linux 下多核处理器相关知识

多核处理器是指在一枚处理器中集成两个或多个完整的计算引擎（内核）。多核技术的开发源于工程师们认识到，仅仅提高单核芯片的速度会产生过多热量且无法带来相应的性能改善，先前的处理器产品就是如此。他们认识到，在先前产品中以那种速率，处理器产生的热量很快会超过太阳表面。即便是没有热量问题，其性价比也令人难以接受，速度稍快的处理器价格要高很多。

基于以上事实，工程师们开发了多核芯片，使之满足‘横向扩展’（而非‘纵向扩充’）的方法，从而提高性能。

1. 在 Linux 下，如何确认是多核或多 CPU:

#cat /proc/cpuinfo

如果有多个类似以下的项目，则为多核或多 CPU:

processor : 0

.....

processor : 1

2. Linux 下，如何看每个 CPU 的使用率：

#top -d 1

之后按下 1. 则显示多个 CPU

Cpu0 : 1.0%us, 3.0%sy, 0.0%ni, 96.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st

Cpu1 : 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st

3. 如何察看某个进程在哪个 CPU 上运行：

#top -d 1

之后按下 f.进入 top Current Fields 设置页面：

选中：j: P = Last used cpu (SMP)

则多了一项：P 显示此进程使用哪个 CPU。

Sam 经过试验发现：同一个进程，在不同时刻，会使用不同 CPU Core.这应该是 Linux Kernel SMP 处理的。

4. 配置 Linux Kernel 使之支持多 Core：

内核配置期间必须启用 CONFIG_SMP 选项，以使内核感知 SMP。

Processor type and features ---> Symmetric multi-processing support

察看当前 Linux Kernel 是否支持（或者使用）SMP

#uname -a

5. Kernel 2.6 的 SMP 负载平衡：

在 SMP 系统中创建任务时，这些任务都被放到一个给定的 CPU 运行队列中。通常来说，我们无法知道一个任务何时是短期存在的，何时需要长期运行。因此，最初任务到 CPU 的分配可能并不理想。

为了在 CPU 之间维护任务负载的均衡，任务可以重新进行分发：将任务从负载重的 CPU 上移动到负载轻的 CPU 上。Linux 2.6 版本的调度器使用负载均衡（load balancing） 提供了这种功能。每隔 200ms，处理器都会检查 CPU 的负载是否不均衡；如果不均衡，处理器就会在 CPU 之间进行一次任务均衡操作。

这个过程的一点负面影响是新 CPU 的缓存对于迁移过来的任务来说是冷的（需要将数据读入缓存中）。

记住 CPU 缓存是一个本地（片上）内存，提供了比系统内存更快的访问能力。如果一个任务是在某个 CPU 上执行的，与这个任务有关的数据都会被放到这个 CPU 的本地缓存中，这就称为热的。如果对于某个任务来说，CPU 的本地缓存中没有任何数据，那么这个缓存就称为冷的。

不幸的是，保持 CPU 繁忙会出现 CPU 缓存对于迁移过来的任务为冷的情况。

6. 应用程序如何利用多 Core：

开发人员可将可并行的代码写入线程，而这些线程会被 SMP 操作系统安排并发运行。

另外，Sam 设想，对于必须顺序执行的代码。可以将其分为多个节点，每个节点为一个 thread.并在节点间放置 channel.节点间形如流水线。这样也可以大大增强 CPU 利用率。