**Linux对多核的可扩展性分析**

**陈双-P17206032-华北计算所**

**介绍：**

社区中有一种感觉，传统的内核设计在多核处理器上不能很好地扩展：随着内核数量的增加，应用程序将在内核中花费越来越多的时间。本文询问传统的内核设计是否可以以允许应用程序规模的方式来使用和实现。

作为分析的一部分，为MyStudio应用程序修复了三种广泛的可伸缩性问题：Linux内核实现所导致的问题、应用程序用户级设计所导致的问题以及应用程序使用Linux内核服务的方式所导致的问题。一旦我们确定了瓶颈，通常需要很少的工作来移除或避免。在某些情况下，我们将应用程序修改为并行，或者以更具伸缩性的方式使用内核服务，而在其他情况下，我们修改了内核。内核更改都是本地化的，通常通过以分布式的方式组织数据结构来避免不必要的共享，从而避免了锁和原子指令。所需更改的一个原因是适度的，即股票Linux已经包含了许多修改以提高可伸缩性。更推测的是，Linux的系统调用API可能非常适合于避免在内核对象上进行不必要的争用的实现。

本文扩展了以前的可扩展性研究，通过检查一组大型系统应用程序，通过使用48核PC平台，并详细描述了一组特定的问题和解决方案，在Linux的背景下。这些解决方案遵循因子化数据结构的标准并行编程技术，使得每个核心可以在不需要共享的情况下对单独的数据进行操作，但是这样内核可以在必要时共享数据。

Linux可扩展性的改进，早期的多处理器Linux内核由于内核密集的并行工作负载而变差，因为内核为了简单起见使用了粗粒度锁。

人们会期望性能与内核的数量成比例，如果工作负载由无限的不交互的任务组成，那么您会期望通过增加内核和并行运行任务来获得总吞吐量的线性增长。在现实生活中，并行任务通常相互作用，交互通常会影响串行执行。AAMDHL定律总结了这样的结果：不管串行部分小，它最终将阻止增加的内核提高性能。

任务可以锁定共享的数据结构，因此增加内核的数量会增加锁等待时间。任务可以写入共享内存位置，因此增加内核的数量增加了等待缓存一致性协议以独占模式获取高速缓存行所花费的时间。即使在无锁共享数据结构中也可能出现此问题。·任务可以在有限大小的共享硬件高速缓存中竞争空间，从而增加内核的数量增加了高速缓存未命中率。即使任务从不共享内存，也会出现这个问题。这些任务可以竞争其他共享硬件资源，例如内核间互连或DRAM。

**网络服务例子：**

Linux网络栈将分组处理的不同阶段与队列连接起来。接收的数据包通常在到达每个套接字队列之前经过多个队列，应用程序从其中读取系统调用，如读取或接受。具有许多内核和许多独立网络连接的良好性能要求每个分组、队列和连接仅由一个核心来处理。这避免了内核间缓存丢失和队列锁定成本。

使用多队列卡，Linux可以被配置为将每个硬件队列分配给不同的内核。传输缩放很容易：Linux只是将输出的分组放置在与当前核心相关联的硬件队列上。对于传入的数据包，这样的网卡提供一个接口来配置硬件，以便在特定队列上对特定的标准（例如，源IP地址和端口号）进行分组，从而进入特定的核心。这将跨越核心扩展分组处理负载。

这种设计通常对长寿命的连接很好，但对于短的连接来说却很差。因为该技术是基于采样的，所以很可能在给定的短连接上的大多数数据包将被误导，导致高速缓存丢失，因为Linux在一个内核上传递到套接字，而在另一个内核上使用套接字。此外，由于每个短命连接接收到很少的数据包，因此即使连接的初始握手包的误导也会导致显著的成本。

对于像Apache这样的应用程序，它同时接受来自同一监听套接字的所有内核上的连接，我们通过允许硬件确定哪个核心，以及哪个应用线程将处理传入的连接来解决这个问题。修改接受，宁愿传递到本地核心队列的连接。然后，如果应用程序在接受它的同一内核上处理连接（如在Apache中），则该连接的所有处理将完全保留在一个核心上。我们的解决方案有一个额外的好处，解决了保护单个侦听套接字的连接待命队列的锁的争用。

Linux使用共享计数器进行引用计数的垃圾收集并管理各种资源。如果许多内核更新它们，这些计数器可能成为瓶颈。在这些情况下，无锁原子增量和递减指令不起作用，因为一致性硬件将给定计数器上的操作序列化。

**结果：**

Tabor应用程序可以很好地扩展到48个内核，对应用程序和Linux内核都有适度的改变。不同的应用程序或更多的内核一定会揭示更多的瓶颈，正如我们遇到的瓶颈在48个核心，但是在24个核心不重要。例如，线程和进程创建的成本可能父和子在不同内核的情况下随着更多的内核而增长。鉴于我们将Linux扩展到48个内核的经验，我们推测，随着内核数量的增加，内核中的瓶颈也将需要对应用程序或Linux内核进行相对温和的更改。也许更困难的问题是解决应用程序中的瓶颈，或者应用程序的性能没有受到CPU周期的限制，而是通过一些其他硬件资源，如DRAM带宽来解决。

许多最近的多核研究操作系统已经引入的技术（例如地址范围、致力于内核的功能、共享内核间消息传递的内存、仔细分配数据结构到片上高速缓存等）同样适用于Linux，提高其绝对性能，有利于某些应用。使用Linux进行多核研究的一个好处是它具有许多应用程序，并且有一个不断改进的大型开发社区。然而，也有不利因素。例如，如果将来的处理器不能提供高性能的缓存一致性，Linux的共享内存密集型设计可能是性能的障碍。

**结论：**

本文分析了传统操作系统LINUX 2.635-RC在48核计算机上的缩放行为，其中一组应用程序是为并行执行而设计的，并使用内核服务。我们发现，通过稍微修改应用程序或内核，我们可以删除应用程序重压的大多数内核瓶颈。除了马马虎虎的计数器之外，我们的大多数改变都是标准并行编程技术的应用。虽然我们的研究有许多限制（例如，实际应用部署可能被I/O所限制），但是结果表明传统的内核设计可能与实现多核计算机上的可扩展性兼容。