**The Scalable Commutativity Rule: Designing Scalable Software for Multicore Processors**

**可扩展交换性规则：多核处理器可扩展软件的设计**

**陈双-P1720632-华北计算所**

**介绍：**

我们知道，在接口中有什么潜在的可扩展性的机会呢，比如像系统调用API？还有在任何实现之前，都可以通过考虑接口规范来确定可扩展性机会吗？本文介绍了以下规则：每当接口操作通勤，他们可以以规模的方式来实现。这一规则有助于开发人员从接口设计开始，通过实现、测试和评估来进行更具伸缩性的软件的构建。

为了帮助开发人员应用该规则，一个新的工具名为Cuffter接受高级接口模型并生成通勤的测试，因此可以缩放。使用这些测试，通勤者可以评估实现的可伸缩性。我们将通勤应用到18个POSIX调用，并使用这些结果来指导一个新的研究操作系统内核的执行，称为Sv6。对于这些呼叫的通勤者生成的13664个测试中，有68%个是Linux的，而通勤者发现了许多限制应用程序可伸缩性的问题。

评估多核软件的可扩展性的最新技术是选择工作负载，在不同数量的内核中绘制性能，并使用诸如差异分析这样的工具来识别可伸缩性瓶颈。这集中开发人员在实际问题上的努力，但有几个缺点。不同的工作负荷或更高的核心计数往往表现出新的瓶颈。目前还不清楚哪些瓶颈是根本性的，因此开发人员可能会放弃，而不知道可扩展的解决方案是可能的。最后，这个过程发生在发展的后期。

**规则方案：**

设计级解决方案如改进的接口可能是不切实际的。本文提出了一种新的可扩展性方法，它从更高的层次开始：软件接口。这使得在实现之前和在必要的硬件可用以测量实现的可扩展性之前可能对可伸缩性进行推理。它可以突出固有的可伸缩性问题，导致交替的接口设计。它为可扩展接口的实现设置了一个明确的缩放目标。

可伸缩性通常被认为是一种实现属性，而不是一种接口属性，尤其是因为什么比例取决于硬件。然而，如果我们假设共享存储器多核处理器具有类似于MESI协议的缓存，则一般的可伸缩性参数是可能的。在这样的处理器上，核心可以对其高速缓存的数据进行可缩放的读取和写入，并可缩放地读取它在共享模式下缓存的数据。然而，写入由另一个核心最后读取或写入的高速缓存行是不可扩展的，因为一致性协议对每个高速缓存行的所有权更改进行序列化，并且因为共享互连可以序列化无关的传输。因此，我们说，如果它们的实现具有无冲突的内存访问，则没有一组操作缩放，其中没有核心写入由另一个内核读取或写入的高速缓存行。当内存访问是无冲突的，添加更多的核将产生容量的线性增加。这不是现代硬件复杂现实的完美模型，但它是一个很好的近似。

我们的方法的核心是这个可扩展的交换性规则：在任何情况下，多个操作通勤意味着无法使用接口区分它们的执行顺序，它们有一个实现，在这些操作期间内存访问是无冲突的。或者，更简洁地说，每当接口操作通勤时，它们可以以规模的方式实现。

文献中很好地建立了交换性和并发性之间的联系。然而，以前的工作集中于使用交换性来同时考虑执行操作的安全性。我们的工作是互补的：我们使用可交换性来推理可伸缩性。

交换性规则具有直观意义：当操作通勤时，其结果（返回值和对系统状态的影响）与顺序无关。因此，交换操作之间的通信是不必要的，并且消除它产生无冲突的实现。这种直观的规则在实践中是有用的，但不够精确，不能正式推理。

这个陈述的一个重要结果是我们称之为SIM交换性的一种新形式的交换性。交换性的通常定义非常严格，很少适用于系统软件中常见的有状态的复杂接口。SIM交换性，相反，是状态依赖的和基于接口的，以及单调的。当操作在特定的系统状态、特定的操作参数和特定的并发操作的上下文中通勤时，我们表明存在对于该状态和这些参数和并发操作是冲突的实现。这暴露了更多的机会将规则应用到实际接口，从而发现可伸缩的实现方式，而不是更传统的交换性概念。尽管其逻辑状态依赖性，SIM交换性是基于接口的：而不是要求所有的操作命令产生相同的内部状态，它需要通过接口来区分所产生的状态。因此，SIM交换性与任何特定的实现无关，使开发人员能够直接将规则应用于接口设计。

交换性规则为设计可扩展软件提供了新的途径：首先，分析接口的可交换性，然后设计一个在交换情况下缩放的实现。例如，考虑在POSIX类文件系统中创建文件。假设多个进程同时在同一目录中创建文件。

系统调用修改同一个目录，所以这个实现必须序列化对目录的访问。但是，如果两个文件有不同的名称，并且没有涉及硬链接或符号链接，那么这些操作就会通勤，因此，有一个针对这些名称进行缩放的实现。一个这样的实现将每个目录表示为由文件名索引的哈希表，每个桶具有独立的锁，从而创建不同命名的文件是无冲突的，不受哈希冲突的影响。在规则之前，我们试图通过分析我们可以想到的所有实现来确定这些操作是否可以伸缩。这个过程是困难的，没有指导的，并且本身没有扩展到复杂的接口，这激发了我们在接口方面推理可扩展性的目标。

复杂的接口会使得很难发现和推理所有交换的情况，即使给出了规则。通勤者使这个推理自动化。通勤者采取一种简化的象征性实现形式的接口模型，计算操作集合通勤的精确条件，并在这些条件下测试冲突自由的实现。该工具可以集成到开发过程中，以驱动初始设计和实现，逐步改进现有的实现，或者帮助开发人员理解接口的交换性。

**方法：**

本文用两种方法证明了交换性规则和通勤者的价值。在第4章中，我们探讨POSIX的交换性，并使用这两种理解来建议设计操作通勤的接口的准则，并提出对POSIX的特定修改，以允许更大的可扩展性。

从业者经常遵循迭代过程来提高可扩展性：设计、实现、度量、重复。通过大量的努力，这种方法使得Linux等内核能够很好地适应许多重要的工作负载。然而，Linux仍然存在许多可扩展性瓶颈，并且缺乏一种推理接口级别可扩展性的方法，目前还不清楚哪些瓶颈是其系统调用接口固有的。本文识别POSIX允许或限制可伸缩性的情况，并指出允许更大的实现可扩展性的特定接口修改。

多核处理器的多内核旨在通过避免内核中的共享数据结构来实现可扩展性。这些系统在消息传递的顶部使用分布式系统技术（例如名称缓存和状态复制）实现共享抽象。可以将交换性规则推广到分布式系统，并且将共享抽象所暴露的接口与它的可扩展性联系起来，即使使用消息传递实现。

**小结：**

可扩展的交换性规则为软件开发人员从软件接口开始理解和开发多核可扩展性提供了一种新的方法。我们定义了SIM交换性，它允许开发人员将规则应用到复杂的、有状态的接口。我们进一步介绍了通勤者，帮助程序员分析接口的可交换性，并测试实现在交换的情况下规模。最后，使用Sv6，我们表明，通过应用规则实现POSIX的广泛可扩展的实现是可行的，并且交换性对于实现真实硬件上的可扩展性和性能是必不可少的。最后希望的是程序员会发现交换性规则有助于产生通过设计可伸缩的软件。