学术论文

多处理器系统中的数据局部性及其优化技术研究

杨学军，戴华东，夏　军
（国防科技大学计算机学院，长沙　410073）

[摘要]　数据局部性是多处理器系统中的重要研究方向之一。结合该领域目前国内外研究现状和我们近一阶段的研究进展，讨论了多处理器系统中的数据局部性及其优化问题。针对现有局部性度量模型存在的不足，提出了一种增强的可用于层次式并行计算机体系结构的局部性度量模型。在静态和动态局部性优化技术方面，分别探讨了基于投影分层的数据变换框架和基于瞬时访问信息的动态页迁移策略，并展开了系列相关的讨论。另外，针对利用数据局部性时必须解决的一个关键问题——存储一致性问题，进行了深入的研究，提出了以操作系统为中心的线程存储一致性模型。

[关键词]　计算机；多处理器系统；数据局部性；局部性度量模型；数据变换框架；页迁移；线程一致性模型

[中图分类号]　TP311.1；TP314　[文献标识码]　A　[文章编号]　1009－1742（2002）05－0044－09

1　引言

随着超大规模集成电路技术的飞速发展和微处理器性能的迅速提高，CPU 的速度和存储访问带宽之间的差距越来越大，存储系统已经成为现代计算机系统整体性能发展的瓶颈之一。为了解决这一问题，最通用的技术策略是将存储系统组织为层次式的多级结构，即采用包括寄存器、cache、二级 cache、主存以及辅存在内的由低到高的多级存储（在分布存储的多处理器系统中，位于其他结点的远程存储也可以看成是多级存储结构中的一个级别）。使用这一存储组织方式的一个基本的假定前提就是处理器发出的大多数存储访问都能够在具有更高速度的低级存储层次上命中，也就是说，处理器即将访问的指令和数据应当尽可能的被放置在离该处理器较近的位置，体现出局部的性质。但是由于价格和工艺等方面的原因，低级存储层次的容量通常都受到一定的限制，为了满足上述前提，通过各种手段来改善数据的局部性，提高存储系统的性能就显得十分必要和有意义了。

数据的局部性问题主要包括两个方面：时间局部性和空间局部性。其中，时间局部性是考虑到处理器最近访问过的存储单元通常在短期内会再次被访问，因此它们应该尽量保留在离处理器较近的低级存储层次。而空间局部性则是考虑到处理器即将访问的存储单元通常是它当前访问存储单元邻近的单元，因此应当从空间的角度将这些单元一起调度。

在计算机体系结构从单机发展到多处理器的历史中，数据局部性的作用呈现日益重要的趋势。早在单机时代，人们就通过引入指令和数据 cache 来改善数据局部性，以提高系统的访存效率。在共享存储多处理器系统中，由于多个处理器竞争共享存储器及存储总线，良好的数据局部性能够减少竞争，对系统的性能显得更为重要。在具有好的可伸缩性的分布存储多处理器系统中，远程存储访问延迟和本地存储访问延迟之间的差距通常能达到 1 到 2 个数量级，开发和改善并行应用程序的数据局部

[收稿日期]　2001－12－06；修回日期　2001－12－31
[基金项目]　国家杰出青年科学基金资助项目（69825104）
[作者简介]　杨学军（1963－），男，山东武城县人，国防科技大学计算机学院教授，博士生导师

性能够在很大程度上减少存储访问停顿时间，提高程序的执行效率，因而被提到更加重要的层次上，已经成为并行计算机系统研究的重点方向之一。

笔者针对多处理器系统中的数据局部性问题进行了系统的研究，从多个角度对这一问题进行了探讨。首先针对现有局部性度量模型存在的不足，提出了一种新的局部性度量模型。在此基础上，进一步从静态和动态局部性优化的角度入手，分别研究了基于投影分层的数据变换框架和基于瞬时访问信息的动态页迁移策略。另外，在多处理器系统中，为了改善数据的局部性，同一数据可能有多个副本，必须通过某种策略来维护这些副本的一致性。针对这一问题，提出了一种新的存储一致性模型——线程存储一致性模型。

2　局部性度量模型

提高数据访问的局部性问题是当前并行处理技术研究的重点，特别是随着今后新的超大规模并行计算机的出现，局部性问题将会变得越来越重要。不能充分开发程序的局部性，就不可能充分利用并行机所提供的各种计算资源，程序执行的加速比也就不能得到应有的提高。因此量化和评价局部性，并用它们来指导程序优化、预测程序执行性能以及改进系统设计就变得十分有意义。

国外目前在局部性度量模型方面已经做了不少的研究工作，Wolf[1]和 Mckinley[2]对循环的局部性给出了量化的评价方法，并用它们来指导各种循环变换。Kandemir[3]也使用了一种简单的循环局部性估算方法，并用它来决定对多个循环进行变换时的先后次序。另外，C. Salisburg[4]针对循环序列所发出的通信请求，给出了多处理机之间的通信局部性量化模型，但该模型假设处理机之间的通信延时都相同，因而不适用于具有层次式结构（处理机间的通信延时可能不同，这与处理器之间的跳步数有关）的并行机。上述的局部性估算方法虽然简单但估算精度不够高。为此，基于 C. Salisburg 的局部性量化模型，我们提出了一种增强的局部性量化模型，它体现了存储访问的空间局部性和时间局部性，且估算精度较高。所提出的局部性评价模型不仅限于单机，还能对在具有层次式结构的并行机上执行的程序进行局部性评测，因此，该局部性评价模型具有通用性。最后，我们想利用该局部性评价模型导出新的局部性优化方法。

我们的目标机器模型是非一致性存储的分布式共享主存的多处理机系统（NUMA），该模型支持全局共享地址空间。程序在执行过程中通过发出访存请求来直接访问物理上分布在各个处理单元上的内存中的任何数据。程序的访存请求可能是本地请求，也可能是远程请求。

程序在执行过程中会发出许多存储访问请求，这些请求可能由并行机中的任意结点提交。每个请求可以用一个四元组来描述，即（源结点，目标访存地址，数据长度，提交时间）。假设访存数据长度为定长，并且 cache line 为该长度的整数倍。根据请求提交的时间（arrive-time）对某个结点所提交的所有访存请求进行排序，可得到该结点所提交的访存请求序列 $\mathcal{J}$，即 $\mathcal{J}=p_1, p_2, \cdots$，其中 p_i 表示第 i 次访存请求，也就相当于由源结点和目标访存地址所确定的一条访存路径。所考虑的程序模型是具有循环结构（即由一个或多个循环构成）的程序，因为循环的局部性优化是编译优化的一个重要方面。当程序在并行机上运行时，每个结点都可能产生访存请求序列，有些结点所产生的访存请求序列可能为空，这表示该结点未被分配子任务，或虽被分配子任务但没有访存需求。

将访存请求序列分割成 n 个子序列，即 $\mathcal{J}=P_1, P_2, \cdots, P_n$。每个 P_i 被称为一个划分（partition），也就是对工作集[5]的一种估计，所有这些划分就构成了序列 $\mathcal{J}$ 的一个分割。可以用定义函数 $\mathcal{F}(\mathcal{J})$ 来创建 $\mathcal{J}$ 的分割。由于存储访问的局部性包括空间局部性和时间局部性，所以一个局部性度量标准应该刻画每个工作集中访存的时间局部性和空间局部性，而一个局部性度量标准应是访存请求序列和它的分割的函数。访存的空间局部性是指相邻的存储单元在不久的将来会被访问；访存的时间局部性是指最近被访问的存储单元在不久的将来会被再次访问。用同一个地址来表示访问时处于同一个 cache line 中的数据单元的目标访存地址，这样在含有相同访存请求数的工作集中，所含不同的 cache line 越少的工作集所体现的空间局部性越好，因为其所含的相邻存储单元较多，反之亦然。如果工作集中目标访存地址所位于的 cache line 被访问的平均频率越高，则其所体现的时间局部性越好，因为该工作集所包含的 cache line 反复出现，所以很有可能在不久的将来被频繁访问，反之亦然。根据前面对机器模型的假设，可以将并行机内部通信

网看作是一个多跳步网络，各 PE 通过它互联。因此，在这种环境下，由同一个源结点所发出的访存请求其访存延迟可能不相同，因为从源结点到达目标访存地址所经历的跳步数可能不同（即访存路径所含的跳步数可能不同）。在多跳步网络中，根据访存路径所含的跳步数，给网络中的每条访存路径都赋予一个权值。如果访存路径包含 i 个跳步，就赋予其权值 α_i。假设网络中最大的跳步数为 $m-1$，则访存路径可能具有的权值应为 α_0，α_1，α_2，…，α_{m-1}。考虑序列 $\mathcal{J}$ 的一个分割 P_1，P_2，…，P_n，令 M_i 为 P_i 中所有不同访存请求组成的集合，$P_i^{(j)}$ 为 P_i 中访存路径所含跳步数为 j 的访存请求组成的集合，$M_i^{(j)}$ 为 $P_i^{(j)}$ 中所有不同访存请求组成的集合，$|M_i|$和$|P_i|$分别为 M_i 和 P_i 中元素的个数，$|M_i^{(j)}|$和$|P_i^{(j)}|$分别为 $M_i^{(j)}$ 和 $P_i^{(j)}$ 中元素的个数。令 $\| M_i \| = \sum_{j=0}^{m-1} \alpha_j | M_i^{(j)} |$，并用 $\| M_i \|$ 来代表划分 P_i 所占的带权的不同访存路径总数。下面的公式定义了序列 $\mathcal{J}$ 的一个分割的空间局部性和时间局部性：

$$\text{空间局部性 } (\mathcal{J},\mathcal{F}) = \frac{1}{(1/n)\sum_{i=1}^{n}\sum_{j=0}^{m-1}\alpha_j | M_i^{(j)} |},$$

$$\text{或} \quad (\mathcal{J},\mathcal{F}) = \frac{1}{(1/n)\sum_{i=1}^{n}\| M_i \|}。$$

可见空间局部性与划分所占的带权的不同访存路径总数的平均数成反比。

$$\text{时间局部性} \quad (\mathcal{J},\mathcal{F}) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(\frac{| P_i |}{| M_i |}\right)^2},$$

用 $\frac{|P_i|}{|M_i|}$ 来表示划分 P_i 中的 cache line 被访问的平均频率。为了突出时间局部性的重要程度，对时间局部性的度量采用了平方根的形式。

这样，综合考虑空间局部性和时间局部性，得到了序列 $\mathcal{J}$ 的一个分割的局部性为：

$$L(\mathcal{F},\mathcal{J}) = \frac{\sqrt{\sum_{i=1}^{n}\left(\frac{| P_i |}{| M_i |}\right)^2}}{\sum_{i=1}^{n}\sum_{j=0}^{m-1}\alpha_j | M_i^{(j)} |},$$

$$\text{或} \quad L(\mathcal{F},\mathcal{J}) = \frac{\sqrt{\sum_{i=1}^{n}\left(\frac{| P_i |}{| M_i |}\right)^2}}{\sum_{i=1}^{n}\| M_i \|}。$$

基于上述局部性度量公式，给出了访存序列分割的基本规则，讨论了分离循环结构中序列的局部性、非分离循环结构中访存序列的局部性以及具有均衡重叠特性的循环序列的局部性问题。另外，我们还给出了循环序列自然局部性的一些性质，以及并行循环的局部性评测方法（略）。

3 静态局部性优化技术

对数据的局部性进行优化一般有两种途径：一种是基于编译技术的静态局部性优化，在编译时对并行应用程序进行数据变换和循环变换，从而提高程序的数据局部性；另外一种，在程序运行的过程中，根据进程和数据的分布情况，动态的复制或者迁移某些数据，从而改善整个系统中的数据局部性（见 4 节）。

3.1 基于投影分层的数据变换技术

近年来，许多研究人员就如何利用数据变换来提高数据访问的局部性问题作了大量的研究。与循环变换相比，数据变换具有许多优点，如数据变换不受数据相关性的限制，能优化松耦合嵌套循环以及显式并行的循环，且数据变换只影响指定进行数据变换的数组而不会影响其他数组。下面简单介绍基于投影分层技术的数据变换框架，并利用该框架来解决提高数据访问的空间局部性问题[6]。

所提出的数据变换框架的理论基础是空间解析几何，它首先将数据访问轨迹参数化，然后求出数据访问轨迹的空间解析方程，最后通过投影分层技术来提高数据访问轨迹的空间局部性。给定一个 m 维的数组 B，可确定它被访问的数据轨迹的参数方程（即由它每一维的数组下标表达式构成），通过将其中某个参数方程设为基准参数方程，可以很容易地求出数组 B 被访问的数据轨迹的空间解析方程。如果数组 B 的下标是仿射下标，那么其空间解析方程将为：

$$\begin{cases} a_{p_2}x_{p_1} - a_{p_1}x_{p_2} = a_{p_2}c_{p_1}(i_1,\cdots,i_{n-1}) - a_{p_1}c_{p_2}(i_1,\cdots,i_{n-1}) \\ \quad\vdots \\ a_{p_h}x_{p_1} - a_{p_1}x_{p_h} = a_{p_h}c_{p_1}(i_1,\cdots,i_{n-1}) - a_{p_1}c_{p_h}(i_1,\cdots,i_{n-1}) \\ x_{p_{h+1}} = c_{p_{h+1}}(i_1,\cdots,i_{n-1}) \\ \quad\vdots \\ x_{p_m} = c_{p_m}(i_1,\cdots,i_{n-1})。 \end{cases}$$

上述空间解析方程代表了 m 维空间上的一组平行直线，且最内层循环的一次执行所访问的数组元素都位于其中的一条直线上。用 X_1，…，X_m 表示

数据空间的坐标轴，且从左至右，X_1，…，X_m 分别对应数组的第一维至数组的最后一维。如果这组平行直线与 X_m 轴平行（本文假设编译器的缺省数据布局为按行存储，即按 X_m 轴方向存储），那么数组 B 将呈现出较好的空间局部性，否则，利用投影分层技术使得这组平行直线与 X_m 轴平行。即选择一组平行于 X_m 轴的参考轴，然后将不同的直线投影到不同的参考轴上。这样，位于同一直线上的数据元素将会按 X_m 轴存储（即按行存储），并且不同直线上的数据元素被投影到不同的列，所以不会发生因投影而引起的数据重叠。经过上述变换后，数组 B 将呈现出较好的空间局部性。以二维数组为例，利用投影分层技术使得数组具有较好的空间局部性的示意图见图 1。

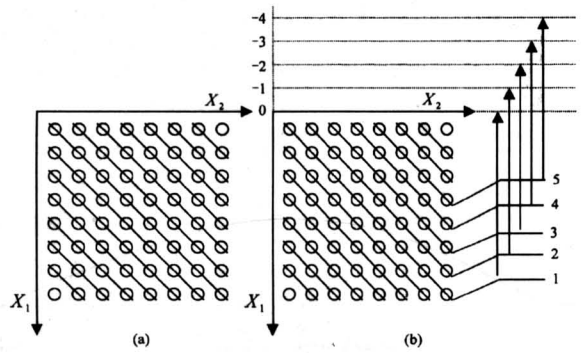

图 1　数据访问轨迹图（a）和数据访问轨迹的投影和分层图（b）

Fig.1　Data access traces (a) and Projection and delaminating of data access traces (b)

完成上述变换所需的数据变换矩阵可以分解为两部分：最后一行和前 $m-1$ 行。最后一行被称作投影行，它起着投影变换的作用；前 $m-1$ 行被称做分层行，它们起着数据分层的作用，即在不影响投影变换的条件下将不同直线上的点映射到不同的参考轴上，分层行是由空间解析方程左端变量前的系数所确定的。该框架还通过修改投影行来缩减访问间距，并且调整分层行之间的顺序，以便进一步提高空间局部性。求解投影行、分层行以及缩减访问间距和调整分层行之间顺序的具体步骤请参考文献［6］。

所提出的数据变换框架不仅能处理仿射数组下标，而且还能处理许多非仿射的更复杂的数组下标。通过使用数据访问轨迹的参数方程，该框架能很简便地确定了数据元素的最优存储布局，且确定数据变换矩阵的方法简单、直接，并尽量减小了变换后的数组访问间距。最后，通过对一组基准程序测试，验证了所提出的基于投影分层技术的数据变换框架的有效性。

3.2　伪共享的减少与消除

Eggers 和 Jeremiassen[7] 的研究表明，有许多程序虽然在单机上呈现出较好的空间局部性，但由于受伪共享的影响，它们在多处理机上的执行性能却较差。因此，如何减少甚至消除伪共享是一个影响程序执行性能的重要问题。研究了利用数据变换与调度技术来提高局部性并同时减少伪共享的方法，并在两者发生冲突时，提出了解决两者间矛盾的途径。由 Torrelas[8] 研究表明，在共享存储的机器上引起 cache 失效的最主要因素是较差的空间局部性而不是伪共享。因此，我们利用基于投影分层技术的空间局部性优化方法来提高数据访问的空间局部性，然后以此为基础再从简单的循环结构入手，考虑伪共享的减少和消除问题，并将之推广到一般的较复杂的循环结构。在解决伪共享问题时，给出了伪共享表现轻微的判别条件，并在伪共享表现轻微的情况下利用数组对准和填充技术消除伪共享，而在伪共享表现严重的情况下利用调度技术、数组逻辑结构还原技术和置换技术减少或消除伪共享。通过减少伪共享的操作后，数据局部性基本保持不变并有可能被进一步改善。我们的研究是利用和推广了 Chow 和 Sarkar[9] 的消除伪共享的调度技术以及 Cierniak 和 Wei Li[10] 的数组逻辑结构还原技术。

4　动态局部性优化技术

可以说，静态局部性优化技术是以程序为中心的，它更多的是从程序本身的角度考虑，对程序中的数组和循环等结构进行重构或变换，以期提高数据初始化分布的合理性。但是，它并没有把程序本身之外的其他因素考虑在内。在多处理器系统中，一个并行应用程序并不是孤立的存在着，它在运行过程中可能会遇到许多不可预测的情况，如由于系统为平衡负载引起的进程迁移或者抢占调度等，导致数据的局部性发生变化。另外，计算服务器工作集的特点千差万别，静态局部性优化技术的作用是有限的。事实上，Open MP 标准在制定的过程中，在是否针对局部性问题进行数据初始化静态分布方面也一直处于进退两难的地步。相比之下，动态局

部性优化技术在程序执行过程中根据实际的情况对数据分布进行调整，因而具有更强的自适应性和实时性。同时，在采用动态优化技术的情况下，数据初始化分布的优劣对程序的执行效率影响会相对减小很多，这在很大程度上减轻了程序员的负担。

我们的工作目前主要集中在分布共享存储系统中的动态页迁移和复制方面。尽管分布共享存储系统提供了对本地主存和远程主存数据访问的透明性，但分布存储结构引起的存储访问延迟不一致性对系统的性能也带来了较大的影响。因此，改善数据的局部性，减少不必要的通信开销，对应用程序的性能起着十分关键的作用。采用适当的页迁移技术能够减小 cache 容量和冲突失效，动态开发数据的局部性，平衡远程和本地访问延迟的不一致，达到优化系统性能的目的。

4.1 基于瞬时访问信息的页迁移策略

页迁移技术允许将单个结点读写访问的页面迁移到该结点，将多个结点只读共享的页面复制到各共享结点，从而变远程访问为本地访问，提高了系统的性能，但同时它也存在过分依赖体系结构、通用性差以及软硬件开销过大的缺点。对目前几种具有代表性的页迁移技术进行了分析，包括页故障触发[11]、cache 失效触发[12]以及 Lazy Home Migration 技术[13]等，发现通用性和硬件开销这两个问题基本上都是在信息收集阶段引起的。页迁移的实质是一种预测技术，它根据收集到的页面访问信息预测将来的访问情况，然后以此为依据作出迁移或者复制的决策。信息收集可以说是页迁移机制的基础，在这一阶段要统计和收集已有的页面访问信息，如周期性的记录各个结点对某一页面的页失效次数或 cache 失效次数。为了达到这一目的，现有的策略一般都需要特殊的页访问监控和记录硬件支持，如 SGI Origin 2000 系统为每个页都设置了各结点针对该页的硬件计数器来统计 cache 失效次数[14]，这不仅引起了额外的硬件开销，还限制了页迁移技术的通用性，如类似的策略就无法在缺乏特殊硬件支持的商用工作站或 PC 构成的 cluster 系统中应用。

页迁移的实质是一种预测技术，用确定的历史访问信息来预测未来不确定的访问存储情况，其效果未必好。基于以上这些考虑，设计了一套没有特殊硬件支持的通用的页迁移策略——基于页的瞬时访问信息的页迁移技术。

所谓瞬时访问信息，是指在某一时刻收集到的各结点对某个页的访问情况，即在某一时刻收集某个页面进入各结点 cache 中的 cache line 的状态信息。瞬时信息比陈旧的历史信息更有前瞻性，且不需要特殊的硬件支持，时间和空间开销小。而且，由于 cache 的状态信息很丰富，瞬时信息可以充分利用已记录的访问情况，在信息收集阶段只需统计信息，不用收集信息，从而减小了系统的开销。由瞬时访问信息的概念，引申出一些根据 cache 状态信息定义的比率，包括页的 cache 率，cache line 分布率，脏 cache line 分布率等，并根据这些比率设计了基于瞬时访问信息的动态页迁移策略，其决策过程如图 2 所示。

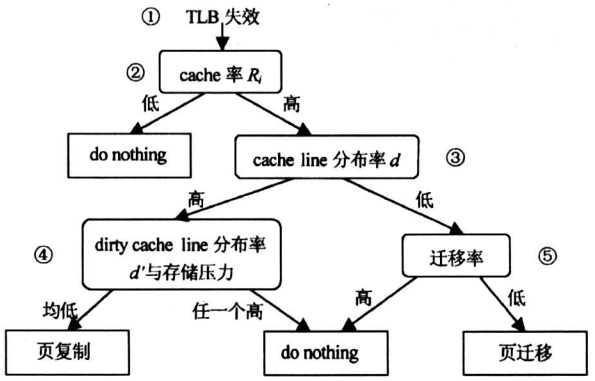

图 2 基于瞬时访问信息的页迁移决策树

Fig.2 Decision tree of instantaneous access information-based page migration policy

我们在 Linux 操作系统内核中针对 Alpha CPU 结构进行实现，通过 Trace Driven 方法模拟得到的性能结果是比较令人满意的。由于用精简的信息作为页迁移的决策基础，基于瞬时访问信息的页迁移机制的实现是在信息收集阶段，开销要小的多，并且无需额外的特殊硬件支持，既可以在硬件支持 cache 一致性的紧耦合 CC－NUMA 系统上应用，也可以在由虚共享软件支持单地址空间的松耦合 cluster 系统中应用。

4.2 Eager 的动态页迁移策略

由于许多并行计算机系统使用 remote cache 来缓存远程数据，相应留给页迁移技术的局部性优化空间更小，因此在选择迁移或复制的候选页时必须精挑细选，在进行迁移和复制时必须尽量及时，同时要避免操作开销过大，得不偿失。受准确性、及时性和低开销这三个目标驱使，必须结合整个系统

的动态实际情况设计适当的迁移策略。

Stanford 大学的 Chandra 等人曾经探讨过 cache 一致性共享存储多处理器系统中的进程调度和页迁移技术[15]，但他们的工作只是同时将这两种技术独立实现在操作系统中以改善系统性能，并没有考虑它们之间的协调和有机融合。近来，Nikolopoulos 等人设计和实现了面向多道程序运行环境的用户级动态页迁移策略[16]，将记录页的历史访问信息的计数器与操作系统调度信息相结合，用抢占和进程迁移作为页迁移的触发，取得了良好的性能改善效果。受到这些工作的启发，我们设计了一种 eager 的操作系统级动态页迁移策略，将现有页迁移技术中影响迁移决策的各种失效次数信息扩展为一个迁移 eager 程度的量化级别，每个页面究竟处于哪个 eager 级别不仅取决于各个结点对它的访问频率，还取决于该页所属的应用程序工作集属性以及操作系统的调度信息和全局线程队列的状态变化情况。通过这一策略，避免了以往的策略中必须等待某个页面成为“hot spot”才能成为迁移和复制候选对象的情况，使得策略的准确性和及时性都有提高。与 Nikolopoulos 等人的工作不同之处在于他们将策略实现在用户级的运行时间库，而我们的设计方案始终集中在操作系统内核级，这样不仅可以将页迁移和执行体调度技术更加紧密结合，允许直接获取系统的调度信息，还可以允许内核页的迁移和复制，从而使得用户模式和内核模式的应用程序都可以获利。另外，由于我们直接在操作系统内核中进行功能扩充和修改，因而可以具体针对页迁移引起的系统开销改进相应的实现手段。例如，页迁移策略最大的开销部分通常是在所有结点刷新 TLB 表操作上，为了避免全局操作，在内核里维护了一个 hash 表，每个表项指向一个链，用来记录究竟哪些结点的 TLB 表项中有当前页面的 cache line 信息。这样，当需要进行 TLB 表 flush 操作时，就可以有选择地针对某些结点进行，从而减小大量不必要的开销。这部分相关的实现工作正在进行中，我们将根据模拟实验的结果对这一策略的性能进行分析，并相应进行改进。

4.3　页复制 + 多写协议

传统的页迁移技术在局部性优化能力方面存在一个明显的限制，就是为了预防乒乓现象，一般不允许对多进程写访问的页面进行迁移或复制，这在很大程度上制约了页迁移技术的广泛应用。目前一种被认可的改进方式是将页迁移和 remote access cache 相结合（这里所提到的 remote access cache 和 CC－NUMA 系统中的 remote cache 或 block cache 概念不同，它指的是在内存中专门开辟的一片用于缓存远程细粒度数据的存储区域，通常由专用的硬件控制器维护，如 Simple COMA 系统，或者是由软件通过可编程逻辑进行管理，如 Stanford FLASH 系统中的 MAGIC。），即对于细粒度和具有短期局部性特征的数据，采用 remote access cache 来管理；而对于页粒度和具有长期局部性特征的数据，考虑对其进行迁移或复制。但是这样也存在一个问题，就是候选迁移或复制页面的产生更加需要精心选择，因为 remote access cache 的使用影响了页迁移的决策，已有的访问信息不能如实反映页的访问情况。经过研究，将页复制技术和多写协议(multi-writer)[17]结合起来的实现方式，允许多个结点写访问的页面也可以被复制，在各个共享结点存在副本，而这些副本之间则采用多写协议来进行一致性维护，减少伪共享现象。目前，多写协议已经成功的在 Tread Marks 等虚共享系统中应用。我们相信，通过将页迁移/复制技术和多写技术的结合来允许多进程写共享页的复制，能够增强页迁移的适用范围，允许更多的局部性开发机会。

5　线程存储一致性模型

在多处理器系统中，在利用数据局部性时必须解决一个关键问题——存储和 cache 一致性的问题。无论是 cache 的引入还是像 4.3 节中所述的允许同一可写数据在多个结点存在副本，都导致了存储访问的非一致性和非原子性问题。另外，在分布存储系统中，由于局部和远程存储访问延迟的差异也会导致存储访问的非一致性问题。这些问题如果得不到解决，有可能使得系统中不同处理器在同一时刻观察到的同一共享存储单元的值不一致，从而产生程序员不希望见到的执行结果。为了保证程序正确执行，必须采用某种存储一致性模型对共享存储访问的顺序加以限制，也就是决定究竟什么时候某个处理器的访存结果为整个系统中的其他处理器所见以及该访存结果如何为其他处理器所见。因此，在探讨并行计算机系统的数据局部性及其优化技术时，对存储一致性模型的问题也进行了深入的研究。

目前比较有代表性的存储一致性模型包括顺序

一致性模型（SC）、弱一致性模型（WO）、释放一致性模型（RC）以及域一致性模型（ScC）等。但是，这些模型普遍存在可扩展性差和依赖体系结构级同步原语、不适合在操作系统级实现的缺陷，因为存储管理是操作系统的核心功能之一。针对这种现象，我们提出了一种新的存储一致性模型——线程存储一致性模型[18]，其核心思想是通过建立线程状态与同步事件之间的对应关系，在操作系统这一级对共享存储访问的事件顺序做出约束，从而实现一个更加适应操作系统线程管理和存储管理、更加便于开发数据局部性、更加放松的存储一致性模型。

首先定义了线程的扩展状态集合，其中比较关键的几个线程状态包括：

• thread-wait-interruptible 状态：处于该状态的线程正在等待某个事件的发生，并可以被相应的信号唤醒；

• thread-wait-uninterruptible 状态：处于该状态的线程因为硬件环境不能满足而等待，在任何情况下都不能被中断，只能用特定的方式来唤醒；

• thread-wakeup 状态：如果线程 t_a 能够唤醒处于以上两种等待状态之一的线程 t_b，则称线程 t_a 处于 thread-wakeup 状态。

随之，我们提出了成簇唤醒线程集和成簇唤醒操作的概念。设 S 为系统中一个线程的集合，令 $S=\{t \mid \forall t \in S$，state (t) = thread-wakeup $\wedge$ wakeset $(t) \subseteq S \wedge$ iff S 中所有线程均进入 thread-wakeup 状态时$\Rightarrow t$ 才能被唤醒$\}$，其中 state (t) 表示线程 t 的当前状态，wakeset (t) 表示线程 t 所唤醒的对象集合，称 S 是系统中的一个成簇唤醒线程集。若用 T_i 来表示处理器 P_i 上所有线程的集合，令 $S_i=\{t \mid t \in S \wedge t \in T_i\}$，其中 $i=1, \cdots, n$，则 S_i 是 S 在处理器 P_i 上所有线程的集合，称 S_i 为处理器 P_i 上的成簇唤醒线程集。相应的唤醒操作（wakeup）称为成簇唤醒操作。

在此基础上，给出了线程存储一致性模型的条件，如图3所示。在三个条件中，条件3是线程存储一致性模型的基本条件，在不考虑其他优化的情况下，它使用操作系统级的线程上下文切换事件来对共享存储访问的顺序作出限制，而传统的模型如SC、WO和RC等都是直接利用硬件或在程序中使用硬件能够识别的同步原语来进行顺序限制，与它们相比，线程存储一致性模型将这种体系结构依赖关系通过操作系统隐藏起来了，因而具有较为广泛的适应性。事实上，只要操作系统能够统一下层体系结构的多样性，线程存储一致性模型就能够不加修改的适应下层的多种体系结构。

若一个系统满足下列三个条件，则称该系统遵循线程一致性模型：

条件1 当处理器 P_i 执行成族唤醒操作时，在 P_i 上的成簇唤醒线程集 S_i 中，在最后一个线程进入 thread－wakeup 状态之前，所有以前相对于 S_i 中任一个线程执行完毕的共享存储访问操作必须相对于处理器 P_i 执行完毕；

条件2 当系统执行成簇唤醒操作时，在成簇唤醒线程集中，在最后一个线程进入 thread－wakeup 状态之前，所有以前相对于 S 中任一个线程执行完毕的共享存储访问操作必须全局执行完毕；

条件3 除满足以上两个条件之外的线程进入 thread－wakeu 状态之前，所有以前相对于该线程执行完毕的共享存储访问操作必须全局执行完毕。

图3 线程存储一致性模型的条件

Fig.3 Conditions of thread consistency model

条件1和条件2是在条件3的基础上对共享存储访问事件的顺序要求做了进一步的放松。当并行程序在运行过程中需要多个线程同步，例如遇到了栅栏同步操作时，如果同一个处理器上有 n 个线程（$n \geqslant 2$）参与此同步操作，每一个线程进入 thread-wakeup 状态时，并不要求它对共享存储的修改结果可见于其他线程，只有当该处理器上最后一个线程到达同步点，也就是进入 thread-wakeup 状态时，才完成相应的一致性维护，使相对于这簇线程中的任一元素执行完毕的存储访问操作结果在该处理器内部为所有参与同步的线程可见；只有当整个系统中参与该同步操作的最后一个线程到达同步点，才进行全局一致性维护，使相对于这簇线程中的任一元素执行完毕的存储访问操作结果为系统中所有参与同步的线程可见。根据我们的经验，类似的情况在并行程序设计过程中是十分常见的，通过这些放松，减少了系统中的冗余消息，从而大量降低了一致性维护的开销。

线程存储一致性模型和以往的各种模型有很大的不同，它是完全从操作系统的角度提出的一种新的存储一致性模型，其放松条件特别适合于多线程操作系统或者支持多线程的体系结构。线程存储一

致性模型的主要特点包括：

• 有机的将线程管理和存储管理结合在一起，适合操作系统级实现；

• 利用多线程的优势，充分将计算和通信重叠，隐藏通信延迟；

• 利用多线程体系结构的优势，通过模型的懒惰实现，减少维护一致性的消息数量，降低同步开销；

• 利用操作系统统一了底层体系结构的多样性，具有较好的通用性和扩展性；

• 在设计和实现时能够直接考虑存储空间与执行体的绑定以及页迁移或进程迁移等问题，利于开发数据局部性。

针对线程存储一致性模型的正确实现，我们从与其他模型的类比和形式化两方面进行了证明，并在 Linux 操作系统和 Pthreads 多线程库环境下进行了模拟实现，性能评价的工作正在进行中。

6　结论与进一步的工作

随着各种不同体系结构的多处理器系统在面向 Great Challenge 的科学计算和大型非数值事务处理等方面日益广泛的应用，数据局部性问题已经成为越来越关键的核心技术。笔者针对这一问题进行了系统的研究，从多个角度展开了探讨。局部性度量模型是量化和评价数据访问局部性的重要手段。为此，首先针对现有度量模型存在的不足，提出了一种增强的可用于层次式并行计算机体系结构的局部性度量模型。在此基础上，进一步从静态和动态局部性优化的角度入手，分别设计了基于投影分层的数据变换框架和基于瞬时访问信息的动态页迁移策略。另外，针对多处理器系统中的存储一致性问题，也进行了深入的研究，提出了以操作系统为中心的线程存储一致性模型。

需要指出的是，尽管我们是从多处理器系统的角度开展研究，但是，很多的研究结果同样也适合于单处理机模型，具有比较广泛的通用性。另外，目前国内外在局部性问题上展开的相关研究工作大多专注于某些方面，与同类工作相比，我们的研究和实现涉及了从理论模型到多种优化技术、到拓展领域，具有很强的系统性。目前，部分研究成果已经在国产并行计算机研制中取得成功应用。

进一步的工作主要包括：

• 在基础理论方面，进一步探讨并行性、数据局部性与数据相关性之间的关系，完善懒惰的线程存储一致性模型；

• 进一步研究如何利用循环变换技术以及将数据变换与循环变换技术相结合来提高数据访问局部性；

• 进一步研究动态页迁移技术的准确性、及时性以及减少系统开销问题；

• 考虑多种自适应的策略相结合以优化局部性问题等。

这些都是很值得进一步开展研究的论题。

参考文献

[1] Wolf M, Lam M. A data locality optimizing algorithm [A]. Proc SIGPLAN Conference on Programming Language Design and Implementation [C], 1991. 30～44

[2] Mckinley K S, Carr S, Tseng C W. Improving data locality with loop transformation [J]. ACM Transactions on Programming Languages and Systems, July 1996, 18 (4): 424～453

[3] Kandemir M, Choudhary A, Ramanujam J, et al. A matrix-based approach to global locality optimization [J]. Journal of Parallel and Distributed Computing, 1999, 58: 190～235

[4] Salisburg C, Chen Z, Melhem R. Modeling Communication Locality in Multiprocessors [J]. Journal of Parallel and Distributed Computing 5, 1999, 6: 71～98

[5] Tanaka A. Extension of the working set for modeling spatial locality in program behavior [A]. Proc 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems [C], 1998. 27～35

[6] 夏　军，杨学军，曾丽芳，等. 基于投影分层技术的嵌套循环空间局部性优化方法 [R]. 长沙：国防科技大学计算机学院，2001

[7] Eggers S, Jeremiassen T. Eliminating false sharing [A]. Proc International Conference on Parallel Processing [C], 1991, I: 377～381

[8] Torrelas J, Lam M, Hennessey J. False sharing and spatial locality in multi-processor cache [J]. IEEE Transactions on Computers, 1994, 43(6): 651～663

[9] Chow J, Sarkar V. False sharing elimination by selection of runtime scheduling parameters [A]. Proc 26th International Conference on Parallel Processing [C], 1997

[10] Cierniak M, Li W. Recovering logical data and code structures [R]. Technique Report 591, Department of Computer Science, University of Rochester, 1995

[11] Black D, Gupta A, Weber W D. Competitive management of distributed shared memory [A]. Proceedings of COMPCON [C], March 1989. 184～190

[12] Verghese B, Devien S, Gupta A, et al. Operating system support for improving data locality on cc-numa compute servers [A]. Proceedings of the Seventh International Conference on Architectual Support for Programming Languages and Operating Systems (ASPLOS VII) [C], October 1996. 279～289,

[13] Baylor S, Ekanadham K, Jann J, et al. Lazy home migration for distributed shared memory systems [A]. Proceedings of International Conference on High Performance Computing [C], December 1997

[14] Laudon J, Lenoski D. The SGI origin: A ccNUMA highly scalable server [A]. Proceedings of the 24th Annual International Symposium on Computer Architecture [C], May 1997

[15] Chandra R, Devine S, Verghese B, et al. Scheduling and page migration for multiprocessor compute servers [A]. Proceedings Architecture Support for Programming Languages and Operating Systems [C], October 1994. 12～24

[16] Nikolopoulos D, Papatheodorou T, Polychronopoulos C, et al. User level dynamic page migration for multiprogrammed shared-memory multiprocessors [A]. Proceedings of the 29th International Conference on Parallel Processing [C], August 2000

[17] Carter J B, Bennett J K, Zwaenepoel W. Techniques for reducing consistency-related information in distributed shared memory systems [J]. ACM Transactinos on Computer Systems, August 1995, 13(3): 205～243

[18] Yang Xuejun, Dai Huadong. Operation system-centric memory consistency model——thread consistency model [A], The Fourth International Workshop on Advanced Parallel Processing Technologies (APPT'01) [C], September 2001. 26～36

Study on Data Locality and Its Optimization Technologies in Multiprocessor Systems

Yang Xuejun, Dai Huadong, Xia Jun

(*School of Computer Science, National University of Defense Technology, Changsha* 410073, *China*)

[**Abstract**] As one of the most important issues in multiprocessor systems, data locality is a main direction of studies on such systems. Combining the recent research achievements by the authors with the current state of studies in this field, a systematic discussion on data locality and its optimization technologies in multiprocessor systems was brought forth. Driven by the deficiency of current locality measurement models, an enhanced model suitable for hierarchy parallel computer architectures was proposed. Besides, from the viewpoint of static locality optimization technologies, a data transformation framework based on projection and delaminating policy was designed. While on the side of dynamic locality optimization technologies, a dynamic page migration mechanism based on instantaneous access information was implemented. More discussion about both static and dynamic optimization policies was presented in this paper. In addition, in-depth investigation on memory consistency model, which raised as a key problem in exploiting locality optimization, was also stressed and a new model centered on operating system——thread consistency model was put forward.

[**Key words**] computer; multiprocessorsystem; data locality; locality measurement model; data transformation framework; page migration; thread consistency model