**摘要**

本发明公开一种基于多线程共享内存通信模式的动态线程到核映射优化方法，该方法检测程序运行的开始和结束，其间周期性地执行一个线程到核映射机制，解决NUMA架构处理器上隐式通信不平衡导致的程序性能下降问题。该机制包括以下步骤：利用基于缺页错误检测的方法检测程序线程间的通信量，其间引入额外的页错误提升检测精度；定义通信量矩阵统计通信量信息；利用基于图划分的分组算法计算线程的合理映射；根据映射计算结果迁移线程，完成一轮映射。

作者：张兴军，鲁晨欣，李靖波，董小社，周剑锋，韩立，纪泽宇

1.一种基于多线程并行应用程序的共享内存通信模式下的线程到核的动态映射方法，其特征在于，在NUMA架构的多核处理器节点上，在线程创建后的程序运行过程中，周期性地执行线程到核动态映射机制，直到线程退出，程序执行完毕。

2.根据权利要求1的方法，

其中通过线程到核动态映射机制实现每一轮的映射过程，由于机制中调用了关键的内核函数，故整体实现为Linux操作系统的内核模块。其特征在于，包括以下步骤：

步骤1，多线程程序在运行环境下，检测并统计线程间通信量。在多线程并行程序运行过程中，利用本发明中引入的通信检测机制，周期性统计不同线程ID之间进行基于共享内存模式的通信。持续性地对每次通信进行记录，统计为线程间通信量矩阵A，n表示程序运行的线程总数，矩阵元素a(i,j) (i)表示线程i和线程j的通信量数值。本步骤的主要目标是收集并统计线程间的通信量信息。

步骤2，计算线程分组。从步骤1中统计得到的通信量矩阵A，转化为表示线程间通信量的无向图，顶点集V是所有线程ID的集合，边集E的元素是二元，表示线程i与线程j之间的通信过程，边上的权值W(i, j)是线程i与线程j的通信量。使用Scotch mapping library的双递归算法，算法输入表示线程间通信量的无向图G和工具Hwloc测试出的节点上系统架构信息，层次化地计算出对线程合理划分后的分组，输出分组结果。分组原则是尽可能将通信量较高的线程，分组数量及组内成员数量由系统架构决定。

步骤3，迁移线程。调用内核函数为每个线程依次设置其cpu亲和度，将程序中的各线程依据分组计算的结果，迁移到对应的cpu核心上。至此，执行完一轮线程到核的映射过程，随着程序持续运行，间隔一段时间间隔，再次执行以上步骤，直至程序运行结束。

3.根据权利要求2所述的映射执行步骤，其中检测并统计线程间通信量包括：

检测线程间的通信量，通过基于内存缺页错误的方法进行。使用Linux操作系统的Kprobe内核函数组，检测程序运行时系统中产生的每次缺页错误，统计并记录产生缺页错误的线程ID和内存物理地址。此外，周期性地引入额外页错误，若如此做，尽管增加了一部分软缺页错误，但整体机制增强了线程间通信统计的精确度。以及

统计线程间的通信量，定义一个哈希表，对内存物理地址空间进行按照一定粒度进行划分，索引为划分后的块ID，哈希表项为一个队列，记录了在该内存块上产生缺页错误的不同的线程ID. 规定队列大小为4，以保证统计的精确度。每次当队列中记录入一个新的线程，将队首元素出队，然后计该新线程与仍在队列中的其他线程之间产生一次通信，以此记录，随时更新通信矩阵。

4.根据权利要求3所述的基于内存缺页错误的通信检测方法，

其中，通过周期性地更改已经产生过内存缺页错误的页表项的内容，引入额外的缺页错误。具体地，删除掉页表中某个内存页的present保留位，如若此，下次有线程想访问该内存页中的数据时，系统就会产生软缺页中断。以及

通信检测机制检测出的缺页错误，包含了两种情况：其一是线程想要访问某数据，但该数据并未在内存中，系统需要从磁盘中调取该块数据到内存，产生缺页中断；其二是检测机制额外引入的软缺页错误。对于前者产生的中断，由操作系统本身处理；由检测机制自行处理。自行处理过程的中断处理程序为：把产生缺页错误的内存地址块的现有内容给需要该块数据的线程，恢复内存地址的页表项。中断处理完毕后，系统恢复，程序得以继续执行。

5.根据权利要求2所述的映射执行步骤，

其中，在每次线程的分组计算过后，需要对通信量矩阵A做老化处理，即把上一次参与分组计算的通信量矩阵的所有元素值a(i,j) (i，乘以老化系数，若如此做，即降低了之前程序线程间通信量情况对后续映射分组计算的影响。一般地，为计算简单，设置，即改变之后的矩阵元素值变为. 本发明将老化系数作为影响动态映射方法性能的一个变量。

6.根据权利要求1的方法和2所述的映射执行步骤，

规定动态映射过程中，每一轮映射机制的执行之间有一定的时间间隔，记该时间间隔为interval. 为优化程序整体运行性能的同时减少映射带来的额外开销，interval的设置必须有一个时间范围。本发明中暂设初始的，若后一次的映射分组计算结果与前一次结果一致或仅存在很小偏差，表明程序的运行特性、线程的通信模式没有随时间发生明显的变化，故interval增加50ms；反之若前后两次计算结果相差很大，表明程序线程间通信模式变化很大，需要及时调整，故interval减少50ms.

整个程序运行过程，映射机制保证，本发明将interval的设置作为影响动态映射方法性能的一个变量。

**基于多线程共享内存通信的动态线程映射优化方法**

**技术领域**

本发明总体上属于计算机体系结构和高性能计算领域，更具体地，涉及多线程并行环境下基于共享内存通信的动态线程映射方法。

**背景技术**

在以OpenMP为代表的多线程并行的编程模型中，应用程序的线程之间通过共享内存空间交换和共享数据，我们称这一过程为隐式通信模式。在并行计算机的计算节点内部，多线程并行程序正是通过这一方式实现数据的交换和共享。

在众多并行计算机体系结构中，NUMA（Non-Uniform Memory Access）架构是指非均匀访问存储模型，这种模型将CPU的资源分开，以node为单位进行切割，每个node里有着独有的计算核心、内存等资源。NUMA架构提升了CPU性能，却造成了两个node之间资源交互非常慢，且在CPU增多的情况下，性能提升的幅度并不高。

当多线程并行应用程序运行在NUMA节点时，若分布在相邻核上的线程之间交换数据，一般将通过邻近几个核心共享的L2 Cache进行；若分布在同一块处理器内物理位置较远的核心上的线程之间交换数据，一般将通过一个处理器共享的L3 Cache进行；若分布在不同处理器上的线程之间交换数据，则只能借助片上互连，从其他处理器的内存中读取。很显然，以上列举的三种方式的代价是不同的，第二种略大于第一种的开销，而由于NUMA架构的特性，第三种开销远大于前两者。因此可以说，在NUMA架构上运行的多线程并行程序通信很不均匀，这就影响了程序运行时的访存效率，最终降低程序整体执行效率。因此，寻找一种合理的映射策略，将多线程并行程序运行时的线程合理分布到系统中的计算核心上，尽量避免跨处理器的远端通信，对提升程序运行性能具有重要的意义。

已有的静态线程映射方法是指在程序首次运行时利用工具对程序进行插桩，分析程序运行时性能。后续通过一系列决策，根据需求计算出最合适的映射方法，在以后相同的程序执行之前，调用已计算好的映射方法，设置线程的亲和度，将线程绑定到具体的核心上。这种静态的映射方式会有效提高具体应用程序的运行效率，但其方法针对具体的程序和具体的机器架构，故通用性差、可移植性差。

线程映射问题是NP-hard问题，其基本解决思路是根据需求对线程进行基于图划分的分组计算。目前已有研究人员通过图论方法、启发式算法和数学规划方法来寻求接近最优的映射方法。但是这些方法存在容易陷入局部最优解、搜索效率低的问题，当并行程序的进程规模增大时，求解最优映射的时间开销过长。

**发明内容**

本发明的目的在于提供一种基于多线程共享内存通信模式的动态的线程到核映射优化方法，以解决上述问题。

本发明的技术方案是这样实现的：

本发明设计并实现了多线程环境下动态的线程到核的映射方法，方法中包含一种线程到核的映射机制，在程序运行中周期性地执行这一机制以实现动态性。本发明将该机制分为三个模块，借助Linux操作系统内核的相关函数功能加以实现。本发明中的方法最终实现为Linux系统的内核模块，具体包含以下步骤：

步骤1，检测程序运行开始，线程创建，程序的运行进入动态映射执行机制中；

步骤2，检测并统计线程间通信量。利用本发明中引入的通信检测机制，持续检测不同线程ID之间进行基于共享内存模式的通信信息。持续性地对每次通信进行记录，统计为线程间通信量矩阵；

步骤3，计算线程分组。从步骤2中统计得到的通信量矩阵，转化为表示线程间通信量的无向图. 使用Scotch mapping library提供的双递归算法，算法输入表示线程间通信量的无向图和工具Hwloc测试出的节点上系统架构信息，层次化地计算出对线程合理划分后的分组，输出分组结果。分组原则是尽可能将通信量较高的线程，分组数量及组内成员数量由系统架构决定；

步骤4，迁移线程。为每个线程依次设置其cpu亲和度，这一过程将程序中的各线程依据步骤3中分组计算的结果，迁移到对应的cpu核心上。至此，执行完一轮线程到核的映射过程；

步骤5，随着程序持续运行，间隔一段时间间隔interval，再次执行步骤2-4，直至检测到线程全部退出，程序运行结束。

进一步地，步骤2具体包括一下内容：

1）检测线程间的通信量。通过基于内存缺页错误的方法进行。使用Linux操作系统的Kprobe内核函数组，检测程序运行时系统中产生的每次缺页错误，统计并记录产生缺页错误的线程ID和内存物理地址。此外，周期性地引入额外页错误，具体地，删除掉某个已发生过缺页错误的内存页在页表项中的present保留位，若如此，下次有线程想访问该相同内存页中的数据时，系统就会产生软缺页中断。增加额外的缺页错误会增加程序运行时缺页错误的数量和系统缺页中断的次数，但整体增强了机制中线程间通信统计的精确度。

通信检测机制检测出的缺页错误，包含了两种情况：其一是线程想要访问某数据，但该数据并未在内存中，系统需要从磁盘中调取该块数据到内存，产生缺页中断；其二是检测机制额外引入的软缺页错误。对于前者产生的中断，由操作系统本身处理；由检测机制自行处理。自行处理过程的中断处理程序为：把产生缺页错误的内存地址块的现有内容给需要该块数据的线程，恢复内存地址的页表项。中断处理完毕后，系统恢复，程序得以继续执行。

2）统计线程间的通信量。定义一个哈希表，对内存物理地址空间进行按照一定粒度进行划分，哈希表大小为内存地址空间除以划分块的粒度，哈希的索引为划分后的块ID，每个哈希表项为一个队列，记录了在该内存块上产生缺页错误的不同的线程ID. 规定队列大小为4，以保证统计的精确度。每次当队列中记录入一个新的线程，将队首元素出队，然后计该新线程与仍在队列中的其他线程之间产生一次通信，以此记录，随时更新通信量矩阵。

本发明的技术效果如下：

1）优化了多线程并行程序在NUMA架构下运行的通信效率和访存效率。本发明提出的映射方法通过将通信频繁的线程映射到物理位置接近的计算核心上，尽可能地避免了两个线程之间通过处理器片间互连的高开销方式进行数据交换和共享，因此该映射方法优化了各线程访问内存数据的效率，整体上提升了程序的执行性能。

2）解决了传统映射无法适应通用的NUMA处理器平台和通用的多线程并行应用程序的问题。首先，本发明中使用Scotch mapping library提供的映射分组算法，获取计算平台的层次化结构并进行层次化分组计算，能够适应不同类型和不同规模NUMA多核处理器平台。其次，本发明采用动态的线程映射机制，在不同程序运行时均可采用本发明中的方法，因此独立于应用程序，具有通用性。

**附图说明**

图1是本发明的实施方案框架图；

图2是本发明实施所述的方法流程图。

**具体实施方式**

下面结合附图和实施例，对本发明的具体实施方法做进一步的详细解释。以下实施例用于说明本发明的具体实施方法，但并不会限制本发明的范围。

实施例所述的框架模块包括：

模块1线程映射机制模块，用于计算并实施每一轮线程映射；

模块2通信检测与统计，用于检测并统计程序运行时线程间通信量；

模块3映射分组计算，用于根据通信量对线程进行分组，得出映射结果；

模块4线程迁移及控制，用于执行计算好的线程映射，并对整体映射流程进行控制。

参照图1、图2所示，本发明所述动态的线程到核映射方法基本流程，包括：

步骤1：检测应用程序运行的开始和线程的创建，若检测出，进入映射机制；

优选地，利用系统内核probe函数检测do\_fork()，但不限于具体的实施方法。

步骤2：周期性地通过改变页表项的保留位，插入额外缺页错误。不断检测程序运行时系统中产生的缺页错误，记录产生缺页错误的内存物理地址所在块号和线程ID. 定义一个哈希表和一个通信矩阵：对内存物理地址空间进行按照一定粒度进行划分，哈希表大小为内存地址空间除以划分块的粒度，哈希的索引为划分后的块ID，每个哈希表项为一个队列，记录了在该内存块上产生缺页错误的不同的线程ID，规定队列大小为4，以保证统计的精确度。通信矩阵读取哈希表中的信息，若同一个哈希表项的队列中由不同线程ID，分别记这几个线程之间产生一次通信，将通信量矩阵的对应位置的元素加1.

步骤3：若检测出缺页错误由系统本身产生，则中断由系统自身处理；若页错误由步骤2额外引入，进入映射机制定义中断处理程序处理该中断。

步骤4：周期性地调取通信量矩阵信息，生成通信量无向图，然后对现有通信量矩阵做老化处理，令.

接着调用基于图划分的分组算法，将程序已有线程按通信量情况进行分组，尽可能使通信频繁的线程在同一个组，得出分组结果；

优选地，使用Scotch mapping library提供的算法进行分组，但不限于具体的实施方法。该算法获取hwloc工具检测出的体系结构信息，采取双递归的分层方法，每一层采用启发式的方法。

步骤5：根据分组计算结果，分别为每个线程设置相应的cpu关联掩码，该掩码规定了线程只能在哪个或哪些cpu上执行，如此在程序运行期间完成线程的迁移。

优选地，直接调用内核函数sched\_setaffinity(pid, sizeof(cpu\_set\_t), &mask)设置具体线程的具体关联掩码mask.

步骤6：检测线程是否全部退出，程序是否运行结束。若没有，根据上一步计算结果调整时间间隔interval，并经过interval时间后，重复步骤4-5；否则，程序运行结束，映射流程结束。

本发明适用于NUMA多核处理器平台下的任意多线程并行应用程序。

以上所述仅为本发明的较佳实施例而已，并不用以限制本发明，凡在本发明的精神和原则之内所作的任何修改、等同替换和改进等，均应包含在本发明的保护范围之内。



图1



图2