**6.3 基于内存拥塞感知的线程级静态映射方案**

对于线程级并行的CFD应用程序在运行时存在以下两个问题:首先，在numa架构下,线程间跨节点通信，会导致线程跨节点访问数据，增加了通信开销，降低了访存效率。其次，节点内部短时间产生大量的内存访问会出现内存拥塞问题，其同样会降低访存效率及内存带宽。为了降低线程间的通信开销以及充分合理的利用内存带宽，以达到优化程序运行时间并提升访存性能的目的，本研究提出了一种基于内存拥塞感知的静态线程映射方案。

6.3.1问题描述

在多线程并行程序运行时，两个不同的线程依次对相同数据的访问和读写操作称为线程间的通信。线程间通信采用隐式模式进行的。图1描述了隐式通信模式在某个NUMA架构的多核处理器上的表现情况。当发生通信过程1时，两个核上的线程访问他们所在核心共享的L2 Cache上的数据，这种情况数据访问快，通信开销较小；发生通信过程2时，两个线程访问他们所在处理器的L3 Cache上的数据，这种情况通信开销较1更大；发生通信过程3时，两个线程的通信需要通过片间互连来完成，势必会访问远端内存上的数据，这种情况通信开销较1和2要大得多。前两种情况均属于local access,通信开销较小，第三种情况属于remote access,如果线程通信时发生过多的通信过程3，会导致性能大幅下降。



图1 隐式通信模式在NUMA架构处理器上的表现情况

其次，由于NUMA架构每个节点集成的核越来越多，如果每个核在较短时间内产生大量的内存访问，这样会对IMC (integrated memory controllers)造成很大的负荷,会产生内存访问拥挤，同样降低了访存效率。如图2，线程1与线程3有较多通信，线2与线程4同样有较多通信，如果将这两对线程对放在同一个节点内，这4个线程会对这个节点产生大量的内存访问。

图示

描述已自动生成

图2 内存访问拥挤示意

6.3.2方案描述

本研究提出了线程到核的映射机制，有效解决了上述两个问题，相较于一些现有研究提出的映射方法，该机制综合考虑了远端访问与内存拥塞两大问题。线程映射机制的主要过程如图3所示。

图示

描述已自动生成

图3 静态映射机制的流程图

映射机制的执行过程如下：

1. 流程开始，预运行一遍应用程序进行通信检测，分别得到时域通信列表与空域通信矩阵。
2. 得到的空域通信矩阵输入给分组算法，得到全部的线程对。
3. 时域通信列表结合全部的线程对得到线程对的时域通信情况。
4. 将所有线程对的时域通信情况进行时间戳聚类，得到线程与核的映射关系。
5. 将映射关系输入至线程绑定函数执行映射，结束流程。

具体实现细节如下：

1. 使用基于Pin tools实现通信检测模块。
2. 使用Numalize通信检测工具，Numalize基于Pin tools开发，以缓存行粒度追踪应用程序的所有内存访问,当不同线程访问缓存行时检测到线程间的通信，输出为通信矩阵。
3. 为了同时统计通信的时间戳信息，对Numalize进行修改，这样可统计每个通信事件的时刻信息，输出表头为（timestamp,t1,t2）的表格。
4. 基于贪心策略的分组算法及时间戳聚类算法实现映射计算模块。
5. 分析全局通信矩阵，借鉴Eagermap的贪心策略进行线程两两配对，使通信最频繁的两个线程组对得到thread\_pairs。
6. thread\_pairs结合时域通信情况得到，这些pairs的时间通信情况。
7. 针对pairs的时间通信情况利用kmeans对时间戳聚类，最后选择通信量最大那一类提取不同线程对，将这些对放在相邻的核心上(减少remote access)并分散到不同节点（缓解memory congestion）。得到线程与核的映射关系map={thread\_id:core\_id}
8. 使用调用hwloc\_set\_cpubind函数或设置numactl环境变量，完成线程映射。

遍历计算映射得到的map，使用hwloc库或者numactl将线程一一绑定至核。

6.3.3 测试结果

本研究基于32核服务器运行NAS Parallel Benchmark程序集。

服务器配置如下：

2numa节点，每个节点8核（每个核心为Intel Xeon E7-4809 v4 @ 2.10GHz）

每个核两个线程，共计16核32线程，32GB DRAM。

测试用NPB测试集的三个应用程序：

CG-含有不规则的长距离通信

SP-5对角线方程组

BT-3对角线方程组，计算与通信较平衡

实验采用3种映射算法：

Compact: Openmp内置映射策略，第i个线程放置第i个核心上。

Eagermap:使用贪心策略让线程两两组对,然后重复上述步骤使两个线程对组成更大对直到将所有线程合并为一个整体。

LMC(Locality and Memory Congestion-aware mapping): 本研究采用的算法。

![图表, 条形图

描述已自动生成]()

图4使用32线程三种算法测试结果对比

结论:线程数越高memory congestion的影响越大，LMC的优化效果越明显。对于NPB测试集的应用程序来说，CG这种通信密集型程序最容易发生内存拥塞，SP次之，BT的通信与计算相当,因此LMC对CG的效果最明显，对于BT几乎没有缓解内存拥塞方面的效果。

上述研究为后续利用E级计算机不同层次的计算资源，实现节点之间低交互、节点内部高效传输的通信机制以及内存带宽的合理利用提供了设计思路。