第 50 卷 第 10 期 西 安 交 通 大 学 学 报 Vol. 50 No. 10
2016 年 10 月 JOURNAL OF XI'AN JIAOTONG UNIVERSITY Oct. 2016

DOI：10.7652/xjtuxb201610009

面向众核系统的线程分组映射方法

巨涛，张兴军，陈衡，董小社
（西安交通大学电子与信息工程学院，710049，西安）

摘要：为了使应用线程更合理地映射到众核处理器具体处理核上，提出一种利用不同线程内部数据局部性及不同线程间数据相关性的特点、结合具体硬件架构特征的线程分组映射方法。通过计算数据重用距离，分析应用程序线程内部数据局部性，用线程相关性矩阵度量不同线程间的数据相关性；根据应用程序数据相关性及众核处理器硬件架构特点，通过设计数据相关性子树生成算法，将应用线程分为能反映不同线程数据访问特点的逻辑组；在线程逻辑分组的基础上，通过线程到处理核的绑定实现线程到具体处理器不同处理核硬件线程的合理映射。实验结果表明：与传统映射方法相比，该线程分组映射方法在不产生额外运行时开销的基础上，计算性能平均提高了 14%，能耗降低了 12%。该方法可以根据应用程序不同线程之间的数据相关性，将不同线程合理映射到具体众核处理器不同处理核上，在不引入额外运行时开销的基础上，提升众核系统的计算效能。

关键词：众核系统；线程映射；数据相关性；数据重用距离；线程逻辑分组
中图分类号：TP399 **文献标志码**：A **文章编号**：0253-987X(2016)10-0057-07

A Grouping Mapping Mechanism of Threads on Many-Core Systems

JU Tao, ZHANG Xingjun, CHEN Heng, DONG Xiaoshe
(School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: A grouping mapping mechanism of threads is proposed to reasonably map application threads to specific processing cores of a many-core processor according to the characteristics of applications. The mechanism bases on the data locality of intra-thread and the data correlation of inter-threads, and combines with the features of hardware architecture of many-core processor. The locality of intra-thread data is analyzed by computing the data reuse distance, and the correlation of inter-threads data is quantified by using a affinity matrix. Threads are divided into different logical groups by designing an algorithm to generate affinity spanning subtree. The reasonable mapping from application to core is realized by binding the thread to the processing core. Experimental results and a comparison with a traditional mapping mechanism show that the proposed mapping mechanism obtains nearly 14% improvement in computing performance and 12% reduction in energy consumption without introducing additional runtime overhead. The mechanism reasonably maps application threads to specific processing cores of many-core processors, and improves computing efficiency of many-core systems.

Keywords: many-core system; thread mapping; data affinity; data reuse distance; logical thread grouping

收稿日期：2016-02-26。 **作者简介**：巨涛(1980—)，男，博士生；董小社(通信作者)，男，教授，博士生导师。 **基金项目**：国家自然科学基金资助项目(61572394，U1304603)；国家高技术研究发展计划资助项目(2014AA01A302)；深圳市科技计划资助项目(JCYJ20120615101127404)。
网络出版时间：2016-07-21 **网络出版地址**：http://www.cnki.net/kcms/detail/61.1069.T.20160721.1102.002.html

如何在充分利用众核处理器高计算能力的同时降低系统能耗是众核系统面临的关键问题[1]。随着多核/众核技术的发展，众核处理器片内集成的处理器核数越来越多，进一步加剧了多个处理核对片上共享计算资源（例如共享缓存和共享带宽）的争用。在程序运行过程中，如果将具有频繁信息交互的线程分配到不同处理核的硬件线程之上，会引入较高的存储访问延迟，造成高的数据传输开销；如果将无数据相关性的多个线程分配到同一个处理核上，会因不同线程访问不同数据，导致共享缓存数据的频繁换入换出，造成高的共享存储访问冲突，增加额外传输开销。只有将应用程序数据局部性和处理器存储架构有效结合起来，实现应用程序到处理核的合理映射，才能降低不同线程之间的共享存储访问冲突、减少额外传输开销，提高计算资源利用率，提升应用程序的计算性能，降低异构系统整体能耗[2]。

已有的根据程序局部性特点进行任务分配的研究工作[3-5]，通过对程序数据及存储访问相关性进行剖分、离线分析，然后划分任务，不考虑运行平台物理架构特点，直接将线程映射到处理核上，不能客观反映不同线程在具体运行平台上运行时的数据相关性特点。文献[6-9]根据程序运行时局部性特点进行动态迁移后实现线程到处理核的映射，但是动态剖分程序局部性及动态迁移线程都会引入较高的额外运行时开销，有的还需特定的硬件支持[8]，限制了其通用性。在处理核数众多且存储结构复杂的众核系统上，由于同时要考虑计算性能和系统的整体能耗，以上线程映射方法不能满足众核系统高效能的计算需求。

本文针对以上问题，通过设计不同线程数据重用距离（data reuse distance，DRD）[4]计算方法，分析不同线程间的数据相关性。在程序运行前，结合程序本身固有的数据相关性和众核系统硬件架构特征，对程序线程进行逻辑分组。之后将不同线程组映射到不同处理核上，使程序数据局部性和运行平台架构特点较好匹配，尽量最大化核内数据共享，最小化核间数据交互，减少因存储访问延迟及共享存储冲突造成的过高额外存储访问及通信开销，在充分利用处理核计算资源的同时尽量降低系统能耗。

1 线程分组映射总体框架

首先根据运行平台所支持的最大硬件线程数，将应用程序划分为相应数量的应用线程。通过设计线程内数据局部性检测机制，剖分不同线程的存储访问数据，计算线程数据重用距离。根据数据重用距离信息，通过设计数据相关性判定方法，分析线程内部数据局部性及不同线程间的数据相关性。具体采用模式分类的方法将不同的线程归并为不同的局部性模式，根据线程不同的局部性模式，用相关性矩阵度量不同线程间的数据相关性。最后通过设计相关性子树生成算法，在相关性矩阵及相关性图的基础上，对应用线程进行逻辑分组，并结合众核系统硬件架构特征，将不同线程分配到能使程序局部性和硬件架构特点较好匹配的处理核上，实现计算任务到处理核的合理映射。总体实现框架如图 1 所示。

图 1 线程分组映射框架

2 程序数据局部性检测

通过剖分统计各个线程的访问数据，设计数据重用距离计算方法，分析不同线程内和线程间数据局部性。采用并行方式剖分线程的数据存储访问信息，并行计算不同线程的数据重用距离。具体参考文献[10-12]中的方法，使用 Intel Pin API 编写 Pin 工具，并行统计每个线程的存储访问数据、计算线程内每个数据的重用距离，根据线程内不同数据重用距离信息，计算反映整个线程数据局部性的平均数据重用距离。

2.1 访问数据统计

采用在平衡二叉树中插入结点的方式统计访问数据。二叉树每个结点用一个结构体表示 $N(T;E;F;W;R)$。每个数据项存储以下信息：T 为时间戳，记录数据被访问的次序；E 记录所访问的数据元素；F 记录数据被访问的次数；W 为权重，记录当前结点所包含的子结点数；R 为数据重用距离。

数据重用信息统计算法包含扫描访问数据、数据结点的插入、原数据结点删除、数据重用度的统计、数据重用距离计算 5 个过程，具体算法如下。

步骤 1 定义结点数据结构 $N(T;E;F;W;R)$ 及空二叉树。

步骤 2 调用 Pin 工具并行扫描不同线程所访问的数据变量，记录扫描到的数据时间戳 t_i 以及数据变量 d_i。

步骤 3 给待插入结点数据项赋初值：$T=t_i$；$E=d_i$；$F=1$；$W=1$；$R=\infty$。

步骤 4 中序遍历当前二叉树，判断待插入数据是

否包含在树中已有结点中，若待插入结点数据包含在二叉树中，则：①调用数据重用距离计算函数（见 2.2 节），计算当前数据重用距离 R；②统计待插入结点数据重用频度 F 及权重 W；③删除当前二叉树中查找的已有数据结点；④调整平衡二叉树，将待插入结点插入到当前平衡二叉树中。

步骤 5　若待插入结点数据没包含在二叉树中，则直接将待插入结点插入到当前平衡二叉树中。

步骤 6　重复步骤 2～5 直到扫描完线程的所有访问数据，最后生成在结点中包含线程所有不同访问数据信息的平衡二叉树。

步骤 7　将树中重用距离为无穷大的结点的重用距离统一调整为树中总的结点个数 M（即树中根结点数据项 W 的值），表示当前最大的重用距离。

步骤 8　完成数据重用信息的统计。

2.2　数据重用距离计算

数据重用距离统计了相同访问数据最近两次访问间隔内不同访问数据的个数，是一个能较好反映应用程序数据局部性的指标[3-4]，本文通过设计数据重用距离计算方法，同时将其封装为独立的函数，在统计访问数据信息时直接调用。具体计算数据重用距离算法实现过程如下。

步骤 1　在当前树中查找包含待插入数据的结点，若无，则该数据为首次访问，重用距离设置为无穷大。

步骤 2　如果找到包含要插入数据的结点，则：①若查到的目标结点时间戳小于根结点时间戳，则该结点为根结点左子树结点，根结点右子树结点数加上其子树中时间戳大于目标结点时间戳的所有结点数即为待插入结点的数据重用距离；②若查到的目标结点时间戳大于根结点时间戳，则目标结点根结点右子树中，时间戳大于目标结点时间戳的结点数 r 即为待插入结点数据重用距离。

步骤 3　比较计算出的待插入结点数据重用距离和当前查找到的目标结点数据重用距离，将两者中较小值作为待插入结点最终的数据重用距离 R。

步骤 4　根结点左子树（或右子树）中，时间戳大于目标结点时间戳的数据结点数计算方法如下：①赋初值为 0；②将当前包含目标结点的右子结点的 W 值赋给 r，并将目标结点作为当前结点；③回溯到当前结点父结点；④如果当前父结点不为根结点且其时间戳大于目标结点时间戳，则将当前父结点的 W 值和其左子结点的 W 值相减后加到 r 中，将当前结点父结点作为当前结点，转到步骤 3 继续执行；⑤如果当前父结点时间戳小于目标结点时间戳，则将当前父结点作为当前结点，直接转到步骤 3 继续执行；⑥如果当前父结点为根结点，则计算完成，当前 r 值即为根结点左子树（或右子树）中时间戳大于当前结点时间戳的结点数。

2.3　线程平均数据重用距离计算

通过遍历生成的每个线程对应二叉树，计算反映每个线程内数据局部性的平均数据重用距离。设线程总数为 K，每个线程的平均数据重用距离为 R_j（$j=1,2,\cdots,K$），线程内每个数据的重用距离为 r_i，线程访问的不同数据总数为 M（平衡二叉树结点数），则线程平均数据重用距离为

$$R_j = \sum_{i=1}^{M} r_i \Big/ M \tag{1}$$

3　数据相关性判定

3.1　线程内数据局部性模式

根据数据存储访问特点，设置数据重用距离阈值 $D_{\min}$ 及 $D_{\max}$，以重用距离阈值为基准，将数据重用距离划分为 3 个不同的区间，每个区间对应一种局部性模式。局部性模式定义如下：数据共享模式（data sharing pattern，DSP），$R_j < D_{\min}$，该模式下线程所访问的数据有较强的时间局部性，数据间有较强的数据相关性；数据无关模式（data isolation pattern，DIP），$R_j > D_{\max}$，该模式下线程所访问数据的时间局部性差，数据之间相互独立，无数据相关性；数据依赖模式（data dependency pattern，DDP），$D_{\min} \leqslant R_j \leqslant D_{\max}$，该模式下线程所访问数据有一定的时间局部性，数据之间存在数据依赖关系，有一定的数据相关性。

数据重用距离阈值的选取对算法性能有重要的影响。如果阈值 $D_{\min}$ 选取得太小，会将一些有一定数据相关性的线程排除在 DSP 模式类之外，反之，则会将一些没有较好数据相关性的线程包含在 DSP 模式类中。同理，对阈值 $D_{\max}$ 的选取同样存在使线程局部性模式分类不准确的问题。本文数据重用距离阈值 $D_{\min}$ 和 $D_{\max}$ 是通过多次实验测试比较后获得的。通过对不同基准测试程序进行测试，计算出各个程序的数据局部性信息，分析不同程序数据局部性特性和数据重用距离的关系，比较所测得的具有较强数据共享性程序的数据重用距离，将其中最大的数据重用距离作为阈值 $D_{\min}$；比较具有相互独立访存关系程序的数据重用距离，将其中最小的数据重用距离作为阈值 $D_{\max}$。本文实验中 $D_{\min}$ 和

D_{max} 分别为每个测试时间间隔内数据访问量的 50%和 85%。

3.2 线程间数据相关性度量

分析出各个线程数据局部性模式后，将不同线程归并为不同的模式类，再分析不同模式类内不同线程间的数据相关性。通过比较同一个模式类内不同线程间所访问的相同数据个数，并记入线程相关性矩阵中，最后通过相关性矩阵来度量线程间的数据相关性。线程相关性矩阵反映了不同线程间的数据共享特性，矩阵的行标和列标分别代表不同的线程 ID，矩阵中的每个元素值代表对应行列所指线程间的数据共享量，矩阵元素值越大，表明对应线程之间数据共享性越好，线程间的相关性越强。计算出线程相关性矩阵后再将该矩阵转换成能直观反映线程间数据相关性的相关性图。相关性图是一个顶点代表不同线程 ID、边代表对应两线程间的数据共享量的无向图。

4 基于数据相关性的线程分组映射

本文基于数据相关性的线程分组映射方法 DagTM(data affinity grouping based thread mapping)，以反映不同线程存储访问特性的数据相关性为基础，结合众核处理器存储层次特点，分两步实现应用线程到处理核的静态映射。第一步，应用线程的逻辑分组。根据计算出的线程相关性图，结合众核处理器处理核所能同时支持的最大硬件线程数，将线程逻辑分组问题抽象成一个图的分解问题，即将线程相关性图分解为 K 棵子树。具体通过设计相关性子树生成算法实现线程的逻辑分组。第二步，在线程逻辑分组的基础上，结合众核处理器架构特点实现线程组内应用线程到众核处理器不同处理核不同硬件线程的映射。

4.1 相关性子树生成算法

(1)设 $G=(V,E)$ 是一个无向连通的加权图，其中顶点 V 表示不同线程的集合，边 E 代表线程间的数据相关性的集合。图中每条边 $(T_i,T_j)\in E$，且都有一个权值 $\omega(T_i,T_j)$，表示两线程间的共享数据量，如图 2a 所示。图 G 的总顶点数为 N_t，表示总的线程数；N_p 表示生成的每棵子树所要包含的节点数；K 表示最终生成的子树个数。

(2)从图 G 上生成 K 棵子树，每棵子树用 $S_k(k=1,2,\cdots,K)$ 表示。生成的每棵子树最多包含 N_p 个节点，同时保证当前生成的每棵子树边的权值之和最大，即满足如下约束条件

$$K=\lceil N_t/N_p \rceil \tag{2}$$

$$\omega(S_k)=\max\sum_{(T_i,T_j)\in S_k}\omega(T_i,T_j) \tag{3}$$

(3)通过借鉴 Prim 最小生成树算法，从无向连通图中生成满足上述约束条件的 K 棵子树。具体算法如下。

输入 一个加权无向连通图 $G=(V,E)$，其中顶点集合为 V，边集合为 E。子树中所包含的最大顶点数为 N_p。

步骤 1 查找图 G 中最大权值的边，将该边及相应的顶点加入到当前子树集合中。

步骤 2 以最大权值边为基础，搜索连接该边顶点其他边中权值最大的作为当前要生成子树的边，并将该边及对应顶点加入到当前子树集合中。

步骤 3 判断当前子树包含的顶点个数 N_k：①如果 $N_k<N_p$，且图 G 中剩余顶点集合不为空，继续查找和当前子树所包含顶点关联的其余顶点中最大权值的边，查到后，返回步骤 2 继续执行；②如果 $N_k=N_p$，则表明已生成一棵完整的子树。删除没有包含在当前子树边集中且和当前子树所含顶点有关联的边，将当前子树从图 G 中分离出来，生成一棵完整的子树 S_k，如图 2d 所示。

步骤 4 在图 G 剩余顶点构成的边集中查找当前权值最大的边，之后转到步骤 2 继续执行，如图 2e 所示。

步骤 5 如果图 G 中剩余顶点构成的边集为空，则整个子树生成过程结束，如图 2g 所示。

输出 生成不同子树的 S_k，如图 2h 所示。

图 2a～图 2h 示意了图中包含的 8 个线程通过上面的相关性子树生成算法，在处理器所支持的最大 4 个硬件线程的情况下，被划分成两个线程组的整个过程。图中相同背景条纹的结点表示同一个线程组。

4.2 线程组到处理核的映射规则

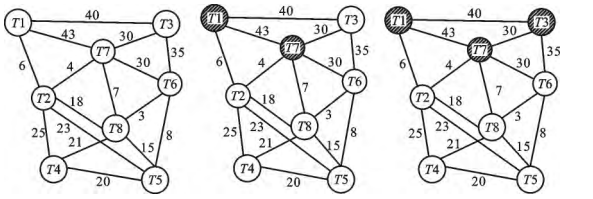
借鉴文献[5]中的映射算法，通过线程到处理核的静态绑定实现映射，具体映射规则如下。

规则 1 将同一线程组中的应用线程尽量分配到同一个处理核的不同硬件线程上。如果该处理核线程已经全部分配，则将应用线程分配到相邻处理核的硬件线程上。

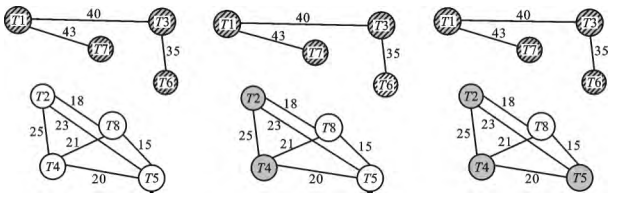
规则 2 将不同线程组中的应用线程分配到不同处理核的硬件线程上，使无共享数据的线程分散到具有独立缓存空间的不同处理核上。

4.3 线程分组映射实现

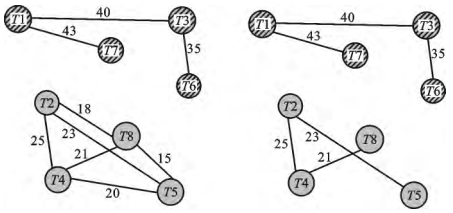
线程分组映射实现分为线程内数据局部性检



(a)初始 Affinity 图　(b)查找最大边(T_1，T_7)　(c)查找边(T_7，T_3)



(d)查找边(T_3，T_6)　(e)分解出子树 S_1　(f)查找边(T_2，T_4)和(T_4，T_5)



(g)查找边(T_2，T_5)　(h)完成两棵子树分解

图 2　相关性图的子树生成过程

测、线程间数据相关性度量、线程逻辑分组、线程组到处理核映射及执行 4 个部分。为说明具体的线程映射过程，用一个 8 线程并行应用程序到 Intel MIC 众核处理器(具体存储架构如图 3 所示，图中 P_i 和 T_i 分别表示不同的处理单元和线程)上的映射为例，比较传统 OpenMP 运行时的线程映射方法和本文 DagTM 方法的映射结果。图 4 比较了传统的 OpenMP 支持的 Compact、Scatter 映射方法和本文 DagTM 方法不同的映射结果。图 4a 中因 Compact 只考虑充分利用处理核资源，不考虑线程间的数据相关性，具有高共享量的线程 T_1、T_3、T_6、T_7 被分派到两个不同的处理核上，因要分别在两个处理核的片内缓存上存储相同共享数据，增加了额外的数据存储开销。图 4b 中 Scatter 方式主要考虑负载平衡，将所有线程均匀地分布在不同的处理核上，会引入过高的额外数据存储访问，同时还存在处理核不能充分利用、系统能耗过高的问题。图 4c 中 DagTM 同时考虑不同线程之间的数据局部性和运行平台架构特点，减少了额外数据访问及数据传输，在映射时充分利用处理核资源，以提高每个处理核的资源利用率，降低系统的整体能耗。

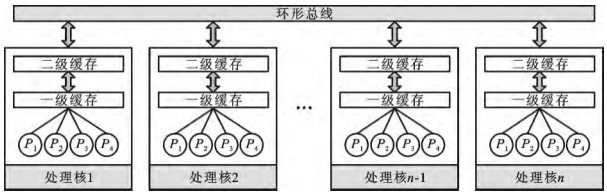
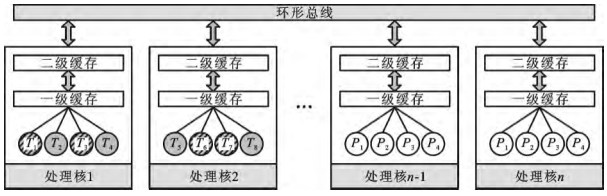
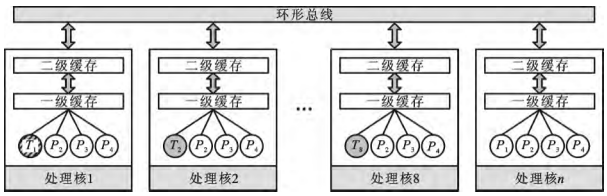


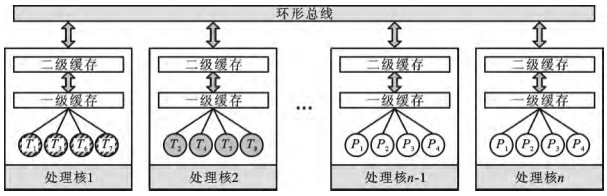
图 3　MIC 处理器存储层次图



(a)OpenMP 内置的 Compact 映射模式



(b)OpenMP 内置的 Scatter 映射模式



(c)DagTM 映射方法

图 4　不同线程到处理核硬件线程映射方式比较

5　实验评测及结果分析

5.1　测试平台及方法

采用 PARSEC[14] 基准测试集中的测试程序 Blackscholes、Ferret、Streamcluster、Raytrace、Bodytrack、Cannel、X264 和 Dedup，用 native 输入集。实验平台由 2 路 8 核 E5-2670 CPU 和 1 块 Xeon Phi 7110P MIC 卡构成众核系统，使用 Red Hat Enterprise Linux Server release 6.3 操作系统，Intel parallel_studio_xe_2013_update3_intel64 MIC 集成开发编译环境。通过 PAPI_5.4.1[15-16] 性能测试工具获取性能指标。分别对本文线程映射方法 DagTm 方法，OpenMP 支持的 Compact、Scatter 方法，以及理想情况下最优的映射方法 Oracle 从计算性能、能耗及额外开销 3 方面进行测试对比。

5.2　计算性能测试

图 5 显示了 DagTM 及其他 3 种映射方法相对

于 OS 默认映射方法 first-touch policy 的基准测试程序计算时间相对减少率。DagTM 平均计算性能提升了近 14%，Oracle 计算性能提升了 16%，Compact、Scatter 映射方法计算性能分别提升了 2%、3%。DagTM 计算性能达到了 Oracle 映射的 81.3%，优于其他两种映射方法。Compact 和 Scatter 在所有测试程序下性能提升都低于 DagTM，Blackscholes、X264、Dedup 等实例作为测试程序时甚至低于 OS 默认的映射方法，主要原因是 Compact 和 Scatter 没有考虑不同线程之间的数据相关性，会引起不同线程之间共享存储访问冲突及额外的数据传输开销。

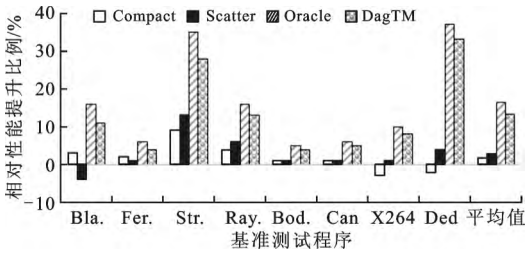


图 5 4 种映射方法相对于 OS 默认映射方法的性能提升

5.3 能耗测试

能耗也是众核系统考虑的主要问题，一方面通过充分利用计算资源、减少额外的数据传输来降低能耗；另一方面可通过 DVFS 技术动态调整处理核的工作状态，采用物理的方法来降低能耗。本文主要通过充分利用计算资源、减少计算时间来相对降低系统能耗。如图 6 所示，DagTM 方法平均能耗相对于 OS 默认方法降低了 12.6%，Compact、Scatter 方法相对能耗分别降低了 3.8%、2.4%，最优映射方法 Oracle 的相对能耗降低了 15%。由于 DagTM 方法在线程映射时考虑了数据相关性，可以减少数据访问冲突，提高共享资源利用率，降低数据传输开销，减少程序的运行时间，相应地减少了系统的整体能耗。

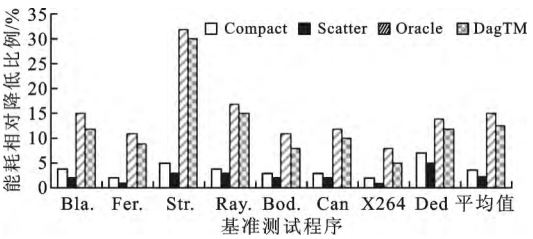


图 6 不同映射方法相对于 OS 默认映射方法的能耗降低比例

5.4 额外开销测试

DagTM 映射方法由于要在程序执行之前进行重用距离的度量及线程分组等预处理，会引入一定的额外预处理开销，但在程序执行过程中不会引入额外的运行时开销。图 7 显示了 DagTM 方法预处理所用时间占整个程序运行时间比例，作为衡量额外开销的指标。平均引入的额外预处理时间占总执行时间的 11%。在线程所访问数据时间局部性差的情况下，因 DagTM 映射方法不会引入额外的运行时开销，所以计算性能和系统默认的映射方法相当。

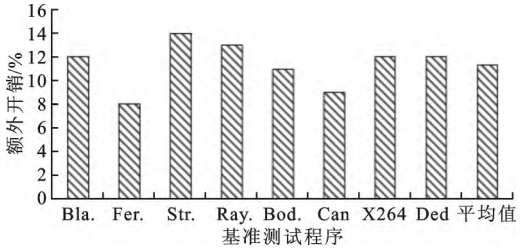


图 7 DagTM 额外开销

6 结 论

本文针对众核系统下线程到处理核的映射问题，通过计算数据重用距离来判定线程内部的数据局部性；用数据相关性矩阵度量线程间的数据相关性；根据线程相关性图，利用最小生成树实现不同线程的逻辑分组；最后通过线程组到处理核的绑定实现线程到处理核的映射。实验评测表明，本文线程分组映射方法在提升计算性能和降低能耗方面是有效的，可以根据应用程序不同线程之间的数据相关性，将不同线程合理映射到具体众核处理器不同处理核上，在不引入额外运行时开销的基础上，提高系统的计算性能，降低系统的整体能耗。

参考文献：

[1] BRODTKORB A R, DYKEN C, HAGEN T R, et al. State-of-the-art in heterogeneous computing [J]. Scientific Programming, 2010, 18(1): 1-33.

[2] 巨涛，朱正东，董小社. 异构众核系统及其编程模型与性能优化技术研究综述 [J]. 电子学报，2015，43(1)：111-119.
JU Tao, ZHU Zhendong, DONG Xiaoshe. The feature, programming model and performance optimization strategy of heterogeneous many-core system: a review [J]. Chinese Journal of Electronics, 2015, 43(1): 111-119.

[3] ZHANG Yuanrui, KANDEMIR M, YEMLIHA T. Studying inter-core data reuse in multicores [J]. ACM Sigmetrics Performance Evaluation Review, 2011, 39

(1)：25-36.

[4] WU Mengju，YEUNG D. Efficient reuse distance analysis of multicore scaling for loop-based parallel programs [J]. ACM Transactions on Computer Systems，2013，31(1)：1-37.

[5] MURALIDHARA S P，KANDEMIR M，KISLAL O. Reuse distance based performance modeling and workload mapping [C]// Proceedings of the 9th Conference on Computing Frontiers. New York，USA：ACM，2012：193-202.

[6] MATTHIAS D，EDUARDO H M C，PHILIPPE O A，et al. kMAF：automatic kernel-level management of thread and data affinity [C]// Proceedings of the 23rd International Conference on Parallel Architectures and Compilation Techniques. New York，USA：ACM，2014：277-288.

[7] DING Wei，KANDEMIR M，YEDLAPALLI P，et al. Locality-aware mapping and scheduling for multicores [C]// Proceedings of the International Symposium on Code Generation and Optimization. Piscataway，NJ，USA：IEEE，2013：1-12.

[8] EDUARDO H M，MATTHIAS D，MARCO A Z，et al. Dynamic thread mapping of shared memory applications by exploiting cache coherence protocols [J]. Journal of Parallel and Distributed Computing，2014，74(3)：2215-2228.

[9] XIANG Xiaoya，DING Chen，LUO Hao，et al. HOTL：a higher order theory of locality [C]// Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems. New York，USA：ACM，2013：343-356.

[10] BACH M，CHARNEY M，COHN R，et al. Analyzing parallel programs with pin [J]. IEEE Computer，2010，43(3)：34-41.

[11] DEREK L，MILIND K，VIJAY S P. Accelerating multicore reuse distance analysis with sampling and parallelization [C]// Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques. New York，USA：ACM，2010：53-64.

[12] NIU Qingpeng，DINAN J，LU Qinda，et al. PARDA：a fast parallel reuse distance analysis algorithm [C]// Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium. Piscataway，NJ，USA：IEEE，2012：1284-1294.

[13] 张保，曹海军，董小社，等. 面向图形处理器重叠通信与计算的数据划分方法 [J]. 西安交通大学学报，2011，45(4)：1-5.
ZHANG Bao，CAO Haijun，DONG Xiaoshe，et al. Novel GPU data partitioning method to overlap communication and computation [J]. Journal of Xi'an Jiaotong University，2011，45(4)：1-5.

[14] BIENIA C，KUMAR S，SINGH J P，et al. The PARSEC benchmark suite：characterization and architectural implications [C]// Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. New York，USA：ACM，2008：72-81.

[15] DAN T，JAGODE H，YOU H，et al. Collecting performance data with PAPI-C [J]. Tools for High Performance Computing. Berlin，Germany：Springer Verlag，2009：157-173.

[16] WEAVER V M，JOHNSON M，KASICHAYANULA K，et al. Measuring energy and power with PAPI [C]// Proceedings of the IEEE International Conference on Parallel Processing Workshops. Piscataway，NJ，USA：IEEE，2012：262-268.

[本刊相关文献链接]

刘强，董小社，朱正东，等. 一种短作业环境下的延迟调度算法. 2015，49(2)：1-5. [doi：10.7652/xjtuxb201502001]

李亮，王恩东，朱正东，等. 应用动态生成树的 GPU 显存数据复用优化. 2013，47(10)：44-50. [doi：10.7652/xjtuxb201310008]

王龙翔，张兴军，朱国峰，等. 重复数据删除中的无向图遍历分组预测方法. 2013，47(10)：51-56. [doi：10.7652/xjtuxb201310009]

张保，董小社，白秀秀，等. CPU-GPU 系统中基于剖分的全局性能优化方法. 2012，46(2)：17-23. [doi：10.7652/xjtuxb201202004]

张保，曹海军，董小社，等. 面向图形处理器重叠通信与计算的数据划分方法. 2011，45(4)：1-5. [doi：10.7652/xjtuxb201104001]

曹仰杰，钱德沛，伍卫国，等. 一种面向多处理器系统的在线低功耗调度算法. 2010，44(8)：15-19. [doi：10.7652/xjtuxb201008004]

冯国富，董小社，丁彦飞，等. 面向 Cell 宽带引擎架构的异构多核访存技术. 2009，43(2)：1-5. [doi：10.7652/xjtuxb200902001]

官尚元，伍卫国，董小社，等. 分布式环境中高效信任管理的研究. 2009，43(6)：15-19. [doi：10.7652/xjtuxb200906004]

薛正华，刘伟哲，董小社，等. 基于思维进化的集群作业调度方法研究. 2008，42(6)：651-654. [doi：10.7652/xjtuxb200806001]

（编辑　武红江）