计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20131960
52(5):1198-1209,2015

访存敏感的增量式 MPSoC 应用映射

王一拙[1] 左 琦[2] 计卫星[1] 王小军[1] 石 峰[1]
[1](北京理工大学计算机学院 北京 100081)
[2](北京市计算中心 北京 100094)
(frankwyz@bit.edu.cn)

Memory-Aware Incremental Mapping of Applications to MPSoC

Wang Yizhuo[1], Zuo Qi[2], Ji Weixing[1], Wang Xiaojun[1], and Shi Feng[1]
[1](*School of Computer Science and Technology*, *Beijing Institute of Technology*, *Beijing* 100081)
[2](*Beijing Computing Center*, *Beijing* 100094)

Abstract The modern multiprocessor system-on-chip (MPSoC) systems normally use network-on-chip (NoC) as their interconnection architecture. Application mapping is one of the key issues in the NoC-based MPSoC design. It maps the tasks of the applications to the nodes of the NoC topology. Many NoC-based MPSoC systems have a shared memory node to store data of the applications. For such MPSoC systems running multiple applications, a memory-aware incremental mapping strategy is proposed. In the strategy, memory access characteristics of the applications are obtained by offline analysis, which classify the applications into hot and non-hot applications. Then, different mapping algorithms are selected according to the memory access characteristic of an oncoming application at runtime. Hot applications are distributed as close as possible to the shared memory and the non-hot applications are distributed as far as possible from the shared memory, according to the proposed strategy. In addition, the application internal and external communication contents are minimized. Experimental results show that the proposed technique saves the communication energy cost by 34.6% on average, and increases the performance by as much as 36.3%, compared with the strategy using a greedy region selection and random mapping algorithms. Moreover, the proposed technique works well on different scale NoCs.

Key words multiprocessor system-on-chip (MPSoC); network-on-chip(NoC); application mapping; task mapping; memory-aware

摘 要 现代多处理器片上系统(multiprocessor system-on-chip, MPSoC)通常采用片上网络(network-on-chip, NoC)作为其基本互连结构,应用映射是基于片上网络互连的 MPSoC 设计中的关键问题,应用映射决定应用划分成的各个任务到片上网络节点的分配.许多基于片上网络互连的 MPSoC 系统将共享存储作为网络中的独立节点,针对这类 MPSoC 系统,提出一种访存敏感的增量式动态映射策略.该策略离线分析获取应用的访存特征,运行中当应用到达系统时,根据其访存特征选择不同的映射算法,将热点应用围绕共享存储器布局,非热点应用远离共享存储器布局,并最小化应用间以及应用所含任务间的通信链路竞争.模拟实验表明:与贪恋区域选择加随机节点映射的策略相比较,提出的策略对

收稿日期:2013-12-27;**修回日期**:2014-05-05
基金项目:国家自然科学基金项目(61300011,61300010)

系统整体通信功耗平均节约 34.6%,性能提升可达 36.3%,并能适应不同片上网络规模.

关键词 多处理器片上系统;片上网络;应用映射;任务映射;访存敏感

中图法分类号 TP302

无论是通用计算还是嵌入式计算领域,包含多个处理单元(processing element, PE)的并行体系结构已成为系统设计的主流,通用计算机系统领域主要体现为多核、众核处理器的广泛应用,嵌入式系统领域主要体现为多处理器片上系统(multiprocessor system-on-chip, MPSoC)技术的蓬勃发展[1].MPSoC 在片上集成多个同构或异构的处理单元(CPU,DSP,FPGA,ASIC 等)、存储单元、互连单元等,通过多个处理单元的并行计算提高系统性能.

应用映射是 MPSoC 设计流程中十分重要的一环,是针对给定的 MPSoC 体系结构,将目标应用划分成多个任务并将这些任务分配到各个处理单元上执行的过程.在 MPSoC 设计领域,应用的划分和映射常被作为 2 个独立的问题来研究,也有研究将二者结合起来,用映射结果的反馈信息进一步优化划分[2].本文只针对划分好的应用,研究任务到处理单元的映射策略,不考虑划分问题.

MPSoC 应用映射技术可分为静态映射和动态映射 2 类[3],静态映射策略在设计时完成任务到 MPSoC 处理单元的分配,因为是离线进行,静态映射可采用复杂的算法得到优化的映射结果.大多数的静态映射方法针对单应用 MPSoC 系统,如文献[4-11],这些方法将性能和能耗作为优化参数,主要针对特定应用的 MPSoC 平台.从实现上来看,静态映射方法又可分为确定性方法和启发式方法,确定性方法一般通过整数线性规划得到确定的任务分配方案[4,12-14],分支限界法常用来具体求解此类问题[15-16];启发式方法通过对任务映射空间的启发式搜索逐渐获得接近最优解的任务分配方案,采用的典型技术包括:遗传算法[5,8-9]、蚁群算法[6-7,10-11]、粒子群优化算法[17-18]等.另外,也有一些研究针对多应用负载的静态映射,文献[19-20]提出基于场景的应用映射方法,一个场景包含一系列同时活动的应用,这类方法的缺点是场景数量随应用数量呈指数型增加,应用较多时基于场景的方法就很难驾驭了.

动态映射策略在运行时动态确定任务的分配,因此算法要相对简单,否则会对应用程序的总体执行时间造成较大影响.常见动态映射算法包括 FF(first free),最近邻(nearest neighbor, NN),PL(path load),BN(best neighbor),MMCL(minimum maximum channel load),MACL(minimum average channel load)等,文献[21]对这些算法进行了介绍和比较,这些算法将任务的计算量和通信特征作为任务分配的依据,以控制系统总的通信量为主要目标,没有考虑应用的数据访问特征.而对许多应用,特别是多媒体应用而言,划分后的任务具有明显的数据访问特征,某些任务需要进行大量访存操作,某些任务则很少访存.这些应用映射到包含有独立存储模块的 MPSoC 系统上时,显然应对任务的访存特征加以考虑.文献[22]提出了访存敏感的映射策略,文献[23]在 MPSoC 中通过改进的交叉开关连接多端口共享存储器,并根据任务的数据访问特征,将数据访问量大的任务尽量安排在多端口共享存储器周围,从而减小通信开销.文献[22-23]的映射策略虽然考虑了任务的访存特征,但却只是针对单应用的静态映射.文献[24]根据应用中多个任务的计算量需求调整功耗约束,设计了运行时增量映射策略.文献[25]设计了用户行为敏感的动态映射策略,将具有高通信率或者在系统中存在时间长的应用判定为关键应用,关键应用选择最小化应用内部链路竞争的分配算法,而其他应用选择最小化全局竞争的分配算法.文献[26]拓展了这一策略,在功耗优化过程中考虑用户行为,并提出一种轻量级的机器学习算法在运行时改进用户模型.

本文针对多应用负载的 MPSoC,提出一种访存敏感的增量式动态映射策略,对每个新载入的应用结合其数据访问特征与系统当前配置进行映射.该策略将应用区分为热点应用(数据访问量大)和非热点应用,对热点应用与非热点应用分别采用不同的在线映射算法,基本思想是让热点应用围绕存储器,非热点应用尽量避免占用存储器附近的资源.本文映射策略结合离线分析与在线映射,其中,离线分析获取所有应用的通信特征与数据访问特征;在线映射首先为应用选择一个资源区域,随后将应用划分成的任务映射到选定区域内.模拟实验表明,本文映射策略能够减少热点应用的通信开销,从而降低系统的整体功耗,提高系统的性能.

1 系统模型

1.1 目标平台模型

片上网络(network-on-chip, NoC)已成为 MPSoC 互连架构的发展趋势[27],与文献[23,28]类似,本文面向含有共享存储器节点的基于 2D-Mesh 片上网络互连的 MPSoC,如图 1 所示,共享存储器通常位于网络的中心位置,以保证访问它的平均路径最短.另外,动态应用映射需要实时更新 MPSoC 的资源使用状况,因此图 1 中 MPSoC 采用 2 个分离的互连网络、数据网络(图 1 中实线)和控制网络(图 1 中虚线)来传递不同的信息.数据网络支持任务之间的数据传递,控制网络只负责通知应用启动与完成等相关的信号,映射策略的设计面向数据网络.图 1 左上角 PE 运行实时操作系统用于完成应用映射,负责初始分配、事件追踪与算法调整,称之为主 PE;其他 PE 不负责调度,只负责执行所分配的任务,称之为从 PE.本文研究的动态映射策略假设从 PE 能够满足应用的单任务节点所需求的处理能力,即应用不需要进行粒度上的重新组合划分.

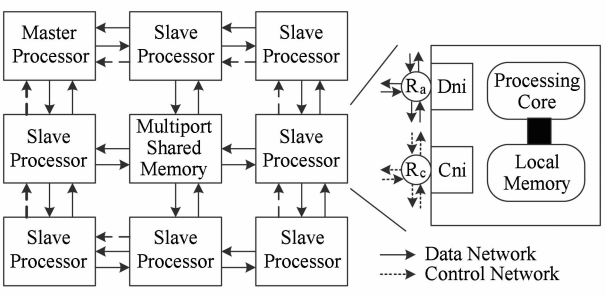


Fig. 1 Architecture of the target platform.

图 1 目标平台结构

1.2 多应用负载模型

本文研究多应用负载的 MPSoC 映射,首先将应用分为热点应用和非热点应用,随后引出应用建模.

定义 1. 热点应用.应用中含有大数据量访存操作的任务,能够采用数据分割的方式将这种任务划分为多个子任务,划分开的节点使用数据缓冲区来存取数据.

定义 2. 非热点应用.应用中不含大量数据操作的任务,每个任务所需要的数据能被本地的局域存储所容纳.

以往 MPSoC 映射策略研究中通常采用应用特征图(application characterization graph, ACG)[15]对应用进行建模,ACG 是一种有向图,每个节点表示一组任务,属于同一节点的任务在同一个 PE 上运行,节点上的数字表示其最小的计算量需求,连接节点的每一条有向边则表示节点之间的通信,边上的权重表示尾节点到头节点的通信量.每个节点都可以被指派到一个 PE 上,边上的权重则成为系统内通信量分析的依据.而对信号处理类应用,常使用同步数据流图(synchronous data flow graph, SDF)[29]来表示应用的特征.SDF 与 ACG 类似,都是有向图,每个节点表示操作或任务.不同的是,在 SDF 中节点之间的有向边表示数据的流动,可以由流来建模,并在其中可以加入 Buf 节点(数据缓冲区节点)来表示应用的数据访问特征[23].总的来看,非热点应用适合用 ACG 表示,热点应用更适合用 SDF 表示.

ACG 的任务间通信与 SDF 的数据传输在本质上一致,都是经由互连结构进行数据包传递.在片上网络互连的 MPSoC 中,这 2 种表示方式的边权重都是判定网络通信状况的依据.本文结合这 2 种应用表示方式的特点,对热点应用与非热点应用定义一种统一表示.

定义 3. 统一应用特征图(unified application characterization graph, UACG).一种加权有向图,节点对应任务(功能划分)以及数据缓冲区,节点间边代表任务间通信或任务与数据缓冲区间的数据传输,边权重代表通信(数据传输)量.

UACG 对全局通信量最大的节点做特殊标记使之成为该应用任务分配的起始节点.ACG 到 UACG 的映射相对直接,2 种图的计算需求与通信量的映射一一对应,只需标记起始节点.而 SDF 到

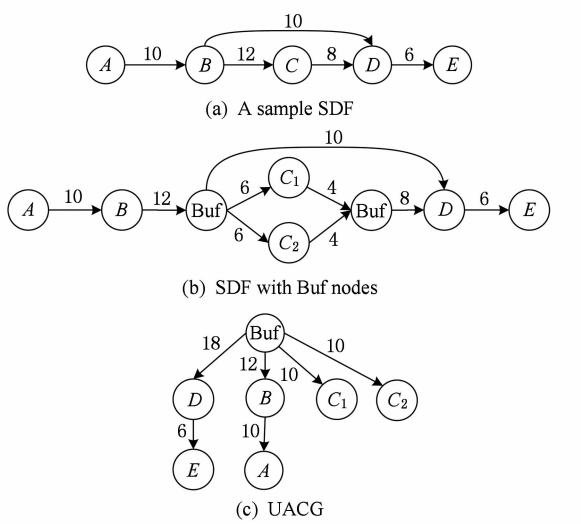


Fig. 2 An example of representing a hot application as UACG from SDF.

图 2 热点应用从 SDF 到 UACG 的表示过程示例

UACG 的转化则相对复杂,其中最关键的是引入 Buf 节点,并将其标记为根节点的过程. 图 2 展示了将一个热点应用的 SDF 转化为 UACG 的过程.

从图 2 可见,构建热点应用的 UACG 主要包含 2 个步骤:1)得到包含数据缓冲区节点的 SDF,这一步根据程序的访存行为完成数据并行化,如图 2 中任务 C 划分为 2 个并行任务 C_1,C_2,任务 B,C_1,C_2 和 D 之间的数据传递经由 Buf 进行;2)将包含 Buf 的 SDF 转化为以 Buf 为根节点的 UACG. 在由热点应用抽取所得的 UACG 中,Buf 必然是总通信量最大的节点,因此成为根节点,该节点在映射过程中可直接指派到共享存储器上. 需要特别指出,因为可能存在回路,所以这里的根节点只是一种表述,并非树的根节点.

2 映射策略框架

访存敏感的增量式映射策略框架如图 3 所示,包括离线和在线 2 部分. 其中,离线分析抽取应用的通信与数据访问特征,得到各应用的 UACG 表示;在线映射完成对应用的布局,由于应用在不同时刻到达系统,因此是一种增量式动态映射,在线映射首先为新到来的应用选择一个区域,该区域所包含的节点数目等于应用所需的资源数目且满足全系统通信约束,随后将应用所含的任务映射到该区域内. 每一个新到来的应用映射到处理器后,处理器的使用状况随之发生变化,应用运行结束时也要更新处理器使用状况.

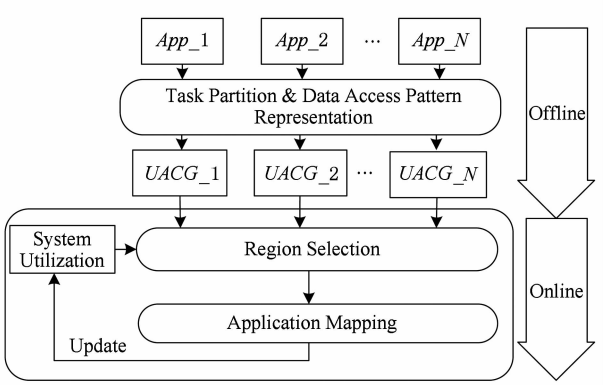


Fig. 3 Memory-aware incremental application mapping framework.

图 3 访存敏感的增量式映射策略框架

图 4 是本文映射策略在线部分的算法流程. 当一个应用到达系统时,先判断系统资源是否满足当前应用的需求,如果资源不足则拒绝该应用执行事件,具体实现为将当前事件放入等待队列中,等系统可用资源充足时再执行;如果当前资源满足应用需求,则根据应用的离线分析信息判断该应用是否为热点应用,对热点与非热点应用采用不同的映射方法. 动态变化的系统中热点应用到来的时间不确定,如果非热点应用首先进入系统,在其映射过程中则应尽量避免占用共享存储器附近的资源,以确保热点应用到达时能顺利获取所需资源. 另外,当系统内不存在可用的连续资源时,需将应用映射到非连续资源上. 图 4 所示访存敏感的增量式映射策略需解决的 4 个子问题(对应算法 P1~P4):P1 完成热点应用的区域选择;P2 将热点应用映射到 P1 选择的区域中;P3 为完成非热点应用的区域选择;P4 将非热点应用映射到 P3 选择的区域中.

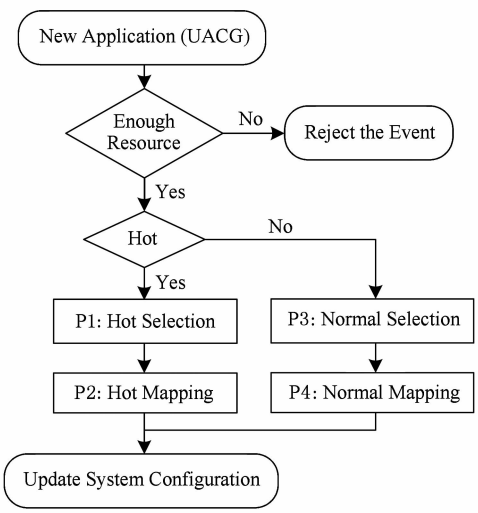


Fig. 4 Flow diagram of the online mapping algorithm.

图 4 在线分配策略算法流程图

本文与文献[16,26]使用相同的通信功耗模型,以曼哈顿距离衡量功耗. 为描述方便,先给出如下定义:

1) $MD(s_i=(x_i,y_i),s_j=(x_j,y_j))$ 表示节点 s_i 与 s_j 之间的曼哈顿距离,其中 x_i,y_i,x_j,y_j 表示在 mesh 系统中的 x 坐标与 y 坐标,$MD(s_i,s_j)=|x_i-x_j|+|y_i-y_j|$;

2) R 为一个包含若干节点的区域,该区域可以是连续的也可以是非连续的;

3) $L(R)$为区域 R 内部所有节点两两之间曼哈顿距离的总和.

用 R'表示当前系统中可获取的空余资源所构成的区域,R 表示为新到来的应用分配资源所占的区域,图 4 中的 4 个子问题描述如下:

1)热点应用区域选择子问题 P1:给定新到来

应用的 UACG(V,E)与系统的当前配置,找到 R'中的一个区域 R,使得:

① R 中的资源数目等于 UACG 的资源需求,即 $\|R\|=\|V\|$;

② 共享存储器在 R 内,即 $Loc(mem)\in R$;

③ R 中的节点以最小化 MD 为标准围绕共享存储器,即 $\forall pos\in R$,$\min\{MD(pos,mem)\}$;

④ $\min\{L(R)+L(R'-R)\}$[①].

2) 热点应用节点映射子问题 P2:给定新到来应用的 UACG 与已经选择好的区域 R(由 P1 生成),找到该应用在区域 R 内的一个映射,使得:

① UACG 中的根节点 Buf 映射到共享存储器上;

② 任务各自的资源两两不相同;

③ 以存储约束为第一要素选择任务围绕共享存储器;

④ 最小化应用内冲突(任务间链路竞争).

3) 非热点应用区域选择子问题 P3:给定新到来应用的 UACG 与系统的当前配置,找到 R'中的一个区域 R,使得:

① R 中的资源数目等于 UACG 中的资源需求;

② 共享存储器不在 R 内;

③ $\min\{L(R)+L(R'-R)\}$;

④ 优先选择远离共享存储器的资源.

4) 非热点应用映射子问题 P4:给定新到来应用的 UACG 与已经选择好的区域 R(由 P3 生成),找到应用在 R 内的一个映射,使得:

① UACG 每个节点都分配一个资源,节点不同、资源不同;

② 最小化应用内冲突.

3 算 法

第 2 节介绍了访存敏感的增量式映射策略的框架与所涉及的子问题,本节介绍用来求解上述每个子问题的算法.定义如下 4 个分量作为算法的基础,其中离散因子与向心因子的定义来自文献[24]:

定义 4. 离散因子 D(PE).资源节点 PE 的空闲相邻节点数等于节点 PE 的总度数减去已分配的相邻节点数.

对基于 2D-Mesh 的 MPSoC,四角顶点的总度数为 3,其他资源(包括边缘资源)总度数是 4,而与多端口共享存储器互连的资源总度数为 5,这里将多端口共享存储器作为额外度数.离散因子 D 是资源是否孤立的重要判断指标,D 越小越容易成为碎片节点,因此为保证区域连续性,同等条件下应优选 D 最小的资源节点.

定义 5. 向心因子 C(PE),资源节点 PE 到已选择子区域边界的曼哈顿距离.

向心因子衡量了尚未分配的资源与已选择子区域的紧密程度.扩张区域时,应选择具有较小 C 值的节点.

定义 6. 存储因子 M(PE),资源节点 PE 到共享存储器的曼哈顿距离.

与 D 和 C 不同,M 对不同应用具有不同的作用.热点应用选择的区域需紧密围绕共享存储器,因而 M 越小越适合;而非热点应用应尽量远离共享存储器,由此 M 越大越适合.MPSoC 结构确定后,每个节点的存储因子保持不变.

定义 7. 空余因子 F,当前资源节点所属 1/4 分区剩余可用的资源数目.

这里的 1/4 分区是以共享存储器为中心,将整个 MPSoC 划分成左上、左下、右上、右下 4 个分区.该指标主要作用于热点应用,优先选择剩余可用节点数少的区域有利于贴近热点应用与已分配应用,减少可能造成的资源碎片.为减少计算量,系统整体保持 4 个空余因子并随着资源使用状况进行更新,不对每个资源节点进行空余因子计算.

图 5 是计算离散因子 D、向心因子 C 与存储因子 M 的示例.图 5 中资源 PE12,PE21,PE22,PE13

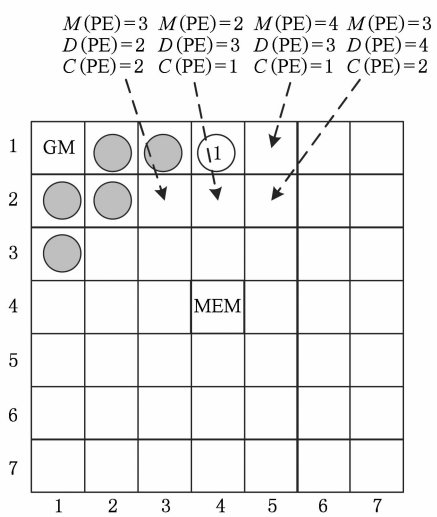


Fig. 5 An example of the calculation of the factors.

图 5 计算各类因子的示例

① 根据已有研究,详见文献[24],最小化外部冲突(应用间链路竞争)的标准是选择 $L(R)+L(R'-R)$最小的区域.

与 PE31 运行系统中已存在的应用,新进入系统的应用不能获取.新进入系统应用已经得到的资源为 PE14,对 PE14 周围资源进行 D(PE)和 C(PE)的计算,结果如图 5 所示.其中 GM 表示主 PE,MEM 表示共享存储器.

3.1 热点应用的区域选择 P1

新应用到达系统时,首先为该应用找到一个资源区域.区域选择算法需保证热点应用与已分配资源尽量贴近,提升区域连续性,减少系统内的资源碎片,也需要确保共享存储器在区域内部.对于热点应用,该区域以共享存储器为中心,对具有大量访存的节点,最小化 M(PE)是资源选择的首要条件,另外,由于 MPSoC 使用最小路径路由算法 XY,同时访存的节点应尽量分散在共享存储器不同的方向.在 M(PE)等同的情况下,选择 D(PE)最小的资源能减少区域的碎片,从而使区域能够更加接近凸形.对非大量访存节点,首要保证区域连续性并减少全局的碎片,因此选择最小化 D(PE),C(PE)与 F 的资源.如算法 1 所示,根据 UACG 得到密集访存节点的集合(Sm)与非密集访存节点的集合(Sn),优先为密集访存节点选择 PE 资源,随后对其他节点构成的集合选择 PE 资源.首个待分配的任务节点映射到系统中 F 值最小的 1/4 分区.

算法 1. 热点应用的区域选择算法.

输入:UACG(V,E);

输出:R.

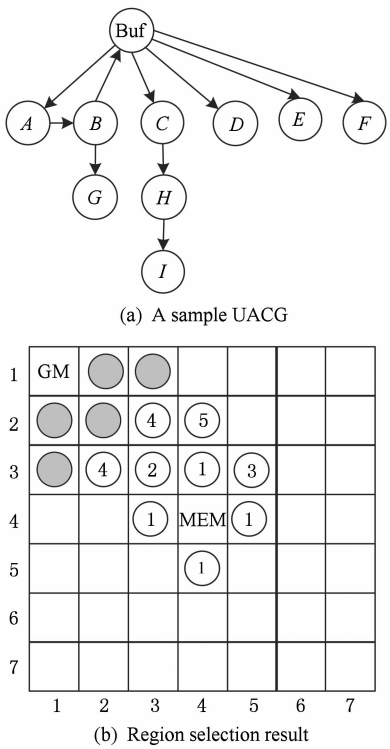
1) 将节点分成节点集合 Sm 和 Sn;

2) 将 Buf 映射到 MEM,然后以最小化 M(PE)+D(PE)为标准选择 PE,将选定的位置加入 R;

3) 更新 F 因子以及所有空闲 PE 的 D(PE),C(PE)因子,以最小化 M(PE)+D(PE)为标准继续选择 PE,且在同等条件下以最小化 F 为标准进行选择,将选定的位置加入 R,重复该步骤直到 R 的大小和 Sm 节点数相同;

4) 更新 F 因子以及所有空闲 PE 的 D(PE),C(PE)因子,以最小化 D(PE)+C(PE)+F 为标准选择 PE,并将选定位置加入 R,重复该步骤直到 R 的大小等于 UACG 的节点数.

图 6 是热点应用区域选择算法的示例.新到达系统的热点应用表示为图 6(a)的 UACG[①],共 9 个节点,其中 6 个为密集访存节点,3 个为非密集访存节点.图 6(b)显示 MPSoC 状态,主 PE(global manager,GM)位于位置 11,运行管理程序;PE12,PE13,PE21,PE22 与 PE31 运行已有应用,标记为黑色,新到达的应用不可获取这些资源.图 6(b)中围绕存储器用序号标记的区域为分配给当前应用的区域,其中标号相同的 PE 代表在同一步中被选中,标号的大小代表选择的顺序.当前新到达系统的热点应用首先获得共享存储器 MEM,将其包含进区域 R.随后,从 Sm 开始分配,PE34,PE45,PE54,PE43 被同时选中,因为这 4 个资源的 M(PE)=1 且均空闲,D(PE)均为 4,根据最小化 M(PE)+D(PE)为指导的原则,直接包含进区域.剩余 2 个访存节点在 M(PE)+D(PE)等同条件下选择最小化 F 的 PE,则 PE33 先被选中,PE35,PE53,PE55 条件等同,随机选择 PE35.最后是非访存节点的资源选择,根据最小化 D(PE)+C(PE)+F 的准则,首先 PE32 与 PE23 同时被选中,随后 PE24 被选中.



(a) A sample UACG

(b) Region selection result

Fig. 6 An example of the region selection for hot application's UACG.

图 6 热点应用 UACG 的区域选择示例

3.2 热点应用的节点映射 P2

热点应用的节点映射算法将应用中的任务映射到算法 1 选定的区域 R 上,映射遵循数据访问节点优先且最小化内部通信冲突的规则,如算法 2 所示.

① 进行区域选择时不考虑应用中各任务之间的具体通信量大小,算法仅与曼哈顿距离等相关,因此这里的 UACG 边上没有标出权重.

算法 2 首先广度优先遍历 UACG,依次指派具有存储访问的节点,确保同时访问存储节点分配到不同的子区域上(以存储器为原点的左上、左下、右上、右下 4 个子区域);随后根据剩余节点与已分配节点的通信量选择位置.

算法 2. 热点应用的节点映射算法.

输入:UACG(V,E),R,$Loc(mem)$;

输出:MPG(UACG—>R).

1) 将 Buf 映射到 $Loc(mem)$;

2) 遍历与 Buf 相连的节点,以最小化到 $Loc(mem)$ 的 MD 值和最小化 $\| subregion \| - \| subUACG \|$ 为标准进行节点映射;

3) 遍历剩余节点,以最小化到已分配的所有相邻节点的 MD 值总和为标准进行节点映射.

3.3 非热点应用的区域选择 P3

非热点应用的区域选择算法从当前系统空闲资源中选择一块能够满足当前应用映射所需的区域,目标是最小化外部冲突并保证所选区域的紧凑性.在区域选择的过程中以尽量远离共享存储器为指导,采用最大化 M(PE)−D(PE)−C(PE)作为判断条件.如算法 3 所示,首先按照某种规则选定一个节点(首节点)加入区域;之后按照判断条件逐步加入新位置,直到选取的位置总数与应用的资源需求量相等为止.非热点应用区域选择的指示条件 M(PE)−D(PE)−C(PE)决定了区域以波浪状向顶端延展.另外,首节点选择方式的不同将形成完全不同的区域.

算法 3. 非热点应用的区域选择算法.

输入:UACG(V,E);

输出:R.

1) 选定首个 PE;

2) 更新所有空闲 PE 的 D(PE),C(PE)因子,以最小化 M(PE)−D(PE)−C(PE)为标准选择 PE 加入 R. 重复该步骤直到 R 的大小等于 UACG 的节点数.

图 7 是非热点应用区域选择算法的示例.示例应用表示为图 7(a)中 8 个节点的 UACG,图 7(b)中的黑色节点已经分配给其他应用.GM 位于位置 11,运行管理程序.同图 6 一样,标号表示 PE 被选中的顺序,如果 2 个 PE 的标号一样,代表同时被选中.当前新到达系统的非热点应用首先获得远离共享存储器的边界资源 PE14;随后,PE15,PE16,PE17,PE27,PE26 被依次选中;最后 PE25 与 PE37 被同时选中.

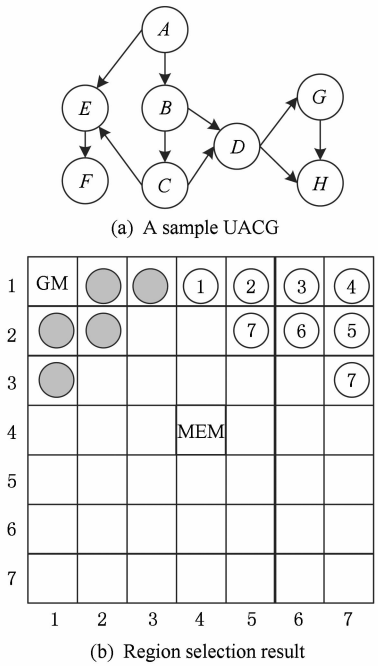


Fig. 7 An example of the region selection for non-hot application's UACG.

图 7 非热点应用 UACG 的区域选择示例

3.4 非热点应用的节点映射 P4

非热点应用的节点映射 P4 将应用中的任务映射到算法 3 选定的区域上,映射以最小化应用内任务间的总通信量为指导.算法与文献[24]的算法一致,步骤 1 找到所形成区域的中心位置,步骤 2 优先选择与其他节点通信总量最大的节点映射到该中心位置,步骤 3 遍历与中心节点关联的节点,并将这些节点按照总通信量递减的顺序逐次分配.

4 实验与分析

4.1 实验方法

为评估本文技术,我们对 Noxim[30] 模拟器进行了扩展,将动态映射算法集成到 Noxim 模拟器中,采用 TGFF[31] 工具生成多个应用负载的任务通信轨迹图,并将其作为模拟实验的输入.Noxim 是基于 SystemC 语言实现的片上网络模拟器,可用来测试片上网络的通信功耗和延迟等.实验中模拟器所需的功耗参数采用 Orion 2.0[32] 获得,所有参数均在 45 nm 工艺、0.8 V 电源电压、3 GHz 主时钟频率下得到.本文功耗和性能分析实验模拟的 MPSoC 片上网络规模为 7×7 节点,中心节点设为多端口共享存储器.

功耗和性能分析实验采用 5 组不同的工作负载,如表 1 所示,每组负载包含 10 个不同应用,为考察不同负载情况下映射策略的效果,5 组负载中热点应用和非热点应用的比例逐次变化. 表 1 中热点应用的访存节点数是指前一列任务节点数中所包含的密集访存节点个数. 系统采用单应用独占多端口共享存储器的资源管理方式,即 MPSoC 中正在运行一个热点应用时另一热点应用只能等待当前热点应用释放资源. 本文设定测试负载中热点应用进入系统的时间间隔超过单个热点应用的最坏运行时间,确保热点应用进入系统时多端口共享存储器处于可用状态.

Table 1 The Workload for Energy Consumption and Performance Analysis

表 1 功耗和性能分析所用负载信息

Workload ID	Hot Application			Non-Hot Application	
	Number	Number of Tasks	Number of Tasks with Heavy Memory Access	Number	Number of Tasks
1	1	8	4	9	3,5,5,8,9,9,11,12,12
2	3	5,8,9	3,4,5	7	3,4,5,5,6,7,8
3	5	3,5,8,9,12	2,3,4,5,6	5	3,5,8,9,12
4	7	3,4,5,5,6,7,8	2,2,3,3,3,4,4	3	5,8,9
5	9	3,5,5,8,9,9,11,12,12	2,2,3,3,4,4,5,5,6	1	8

实验比较本文算法和贪恋区域选择与随机映射算法,贪恋区域选择算法在新应用进入系统时,选择曼哈顿距离总和最小的区域为目标应用的映射区域,即以最小化为每步节点选择的依据,App 指当前系统中的应用.

$$E_{\text{comm\_total}} = \sum_{\forall App} \sum_{\forall i,\forall j} MD(v_i, v_j).$$

贪恋区域选择不考虑未来进入系统的应用情况,只考虑当前应用,可能造成之后其他应用的通信开销大幅增加[24],但算法实现简单、时间复杂度低,因此常被用于动态应用映射中[33-34]. 随机映射是指在采用贪恋区域选择算法获得和当前应用的任务数相匹配的 MPSoC 节点区域后,将任务节点一对一随机映射到选定区域的资源节点上.

4.2 功耗分析

首先,比较不同映射策略下系统的整体通信功耗,如图 8 所示,比较了区域选择算法与节点映射算法的 4 种不同组合,区域选择使用贪婪选择算法或本文的选择算法 P1,P3,节点映射采用随机映射或本文的映射算法 P2,P4. 实验结果以采用贪婪选择与随机节点映射的策略为基准归一化,由图 8 可见,本文策略结合了区域的优化选择与区域内部冲突的控制,因此优于其他 3 种组合,系统总体通信功耗较贪恋选择加随机映射策略平均节约 34.6%. 从不同工作负载组的结果来看,第 4 组本文策略的功耗节约最多(41%),第 1,5 组相对较少,说明负载中热点应用或非热点应用占绝大多数时不能充分发挥本文算法的效率. 另外,从 Greedy+P2P4,P1P3+Random 的结果看,单独在区域选择或节点映射方面的改进对功耗节约的贡献基本相似,具体情况与应用相关,如工作负载组 3,5 所对应的 P1P3+Random 效果较好;工作负载组 1,2,4 则 Greedy+P2P4 较好.

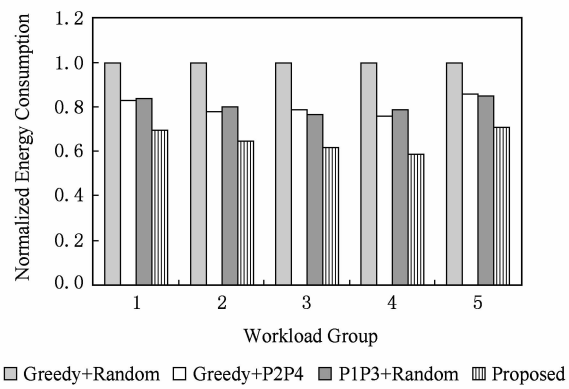


Fig. 8 Communication energy consumption comparison of the schemes.

图 8 采用不同映射策略时的系统通信功耗比较

其次,分析热点应用的通信开销,以验证映射策略对此类应用通信功耗的改善. 图 9 比较了 2 种映射策略下热点应用的通信功耗,与系统整体通信功耗的比较类似,以贪婪选择加随机节点映射的策略为基准归一化. 由图 9 可见,相比 Greedy+Random 策略,采用本文映射策略使热点应用的通信功耗平

均降低了约 42%,这主要归功于本文映射策略以应用的数据访问特征为重要约束条件,控制热点应用位置,使之环绕多端口共享存储器,并且令非热点应用远离多端口存储器.

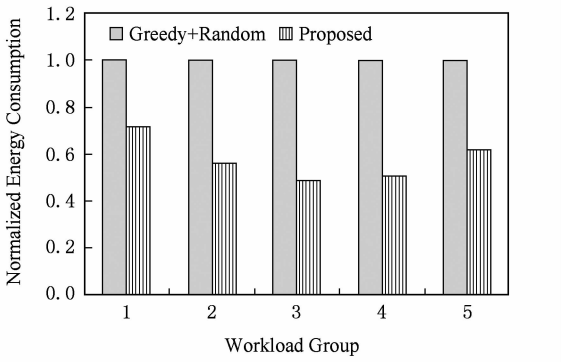


Fig 9. Communication energy consumption comparison of the schemes for hot applications.

图 9 热点应用的通信功耗比较

4.3 性能分析

为评估本文映射策略对系统性能的影响,比较贪婪选择算法、本文区域选择算法 P1,P3 与随机映射、本文节点映射算法 P2,P4 的 4 种组合下各工作负载组的执行时间,通过修改 Noxim 得到的执行时间为片上网络中发出第 1 个 flit(数据片)到接收最后一个 flit 之间的时钟周期数. 结果如图 10 所示,因热点应用执行时间主导着多应用负载的整体执行时间,工作负载组 1~5 随热点应用数量的增多执行时间也逐次增长. 与 Greedy+Random 策略相比较,本文映射策略在 5 组工作负载下对系统性能都有所提升,非热点应用占大多数时(组 1、组 2)性能提升较小,热点应用较多时性能提升基本和通信能耗的节约呈现相似状况,工作负载组 3~5 本文策略比较 Greedy+Random 平均获得了 31.5%的性能提升,最高为 36.3%(组 4). 另外,图 10 中各优化策略相对 Greedy+Random 性能提升的比例小于图 8 中通信能耗的节约比例,这是因为系统性能还涉及动态映射过程的开销等各个方面.

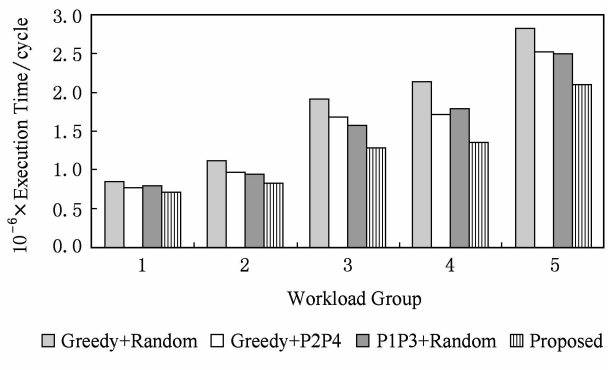


Fig. 10 Performance comparison of the schemes.

图 10 不同映射策略的性能比较

4.4 扩展性分析

4.2 节、4.3 节实验都是基于 7×7 节点 MPSoC 进行的,为评估本文映射策略对不同规模 MPSoC 的可扩展性,本节对 5×5 到 9×9 这 5 种不同规模的 MPSoC,通过 Noxim 模拟实验比较不同应用映射策略下的片上网络功耗,实验所采用负载情况如表 2 所示,不同规模的 MPSoC 使用不同的负载,规模越大负载越复杂.

Table 2 The Workload for Scalability Analysis

表 2 扩展性分析所用负载信息

MPSoC Size	Hot Application		Non-Hot Application	
	Number	Number of Tasks	Number	Number of Tasks
5×5	2	6,8	3	5,8,12
6×6	3	6,8,10	4	3,5,8,12
7×7	3	6,8,10	7	3,4,5,5,6,7,8
8×8	4	6,8,10,12	8	3,5,5,8,9,9,11,12
9×9	4	6,8,10,12	10	3,5,5,8,9,9,11,12,12,13

图 11 给出了不同规模 MPSoC 上本文映射策略与其他映射策略相比较功耗节约的百分比,用来比较的映射策略有 2 种:1)贪婪区域选择与随机节点映射的组合(Greedy+Random);2)本文区域选择与随机节点映射的组合(P1P3+Random). 由图 11 可见,采用本文映射策略并不会随着处理器规模增大而降低功耗节约的效果,反而会略有提升. 产生这种现象的根本原因是,随着片上互连网络规模的增大,贪婪选择更容易导致资源碎片的产生且随机节点映射会引发更大的通信距离;而本文映射策略在

不同规模的 MPSoC 上,都能保证热点应用围绕多端口共享存储器,并在保持应用间链路竞争尽量小的同时有效控制应用内部通信冲突.

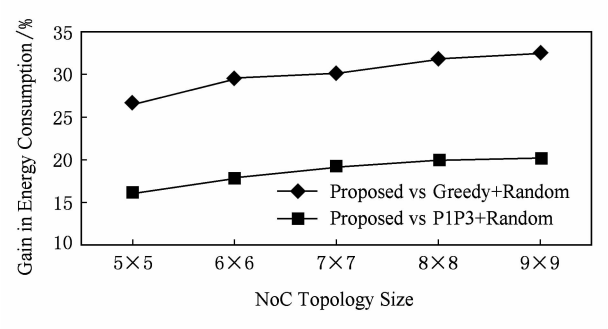


Fig. 11 Energy consumption comparison of the schemes on different size of NoC.

图 11 不同 NoC 规模下映射策略的能耗比较

5 总结与展望

本文设计了一种访存敏感的增量式 MPSoC 应用映射策略,该策略面向片上网络互连的 MPSoC,对包含热点应用(含有大量访存操作任务)的多应用负载进行动态映射,根据应用的数据访问特征选择不同映射算法.本文采用多端口共享存储器支持热点应用的并行访存,对热点与非热点应用均使用反映访存距离与区域连续性的各种因子作为映射指导.模拟实验表明:1)访存敏感的增量式映射策略能够根据应用的数据访问特征动态适应系统配置,改善热点应用的通信开销以及有效控制系统的总通信开销,从而达到节约片上网络功耗的目的;2)该映射策略具有良好的扩展性,功耗节约随着 MPSoC 规模的增大而略有增长.

进一步研究将使用真实负载对本文映射策略进行评估;并对其他拓扑结构的片上网络,如二维环网、基三片上互连网络[35]等设计访存敏感的增量式映射算法.

参 考 文 献

[1] Li Renfa, Liu Yan, Xu Cheng. A survey of task scheduling research progress on multiprocessor system on chip [J]. Journal of Computer Research and Development, 2008, 45(9): 1620-1629 (in Chinese)
(李仁发, 刘彦, 徐成. 多处理器片上系统任务调度研究进展评述[J]. 计算机研究与发展, 2008, 45(9): 1620-1629)

[2] Pop R, Kumar S. A survey of techniques for mapping and scheduling applications to network on chip systems, 04-4 [R]. Dublin, Ireland: Jönköping University, 2004

[3] Sahu P K, Chattopadhyay S. A survey on application mapping strategies for Network-on-Chip design [J]. Journal of System Architecture, 2013, 59(1): 60-76

[4] Bender A. MILP based task mapping for heterogeneous multiprocessor systems [C] //Proc of EURO-DAC'96. Los Alamitos, CA: IEEE Computer Society, 1996: 190-197

[5] Lei T, Kumar S. A two-step genetic algorithm for mapping task graphs to a network on chip architecture [C] //Proc of the 6th Euromicro Symp on Digital System Design. Los Alamitos, CA: IEEE Computer Society, 2003: 180-187

[6] Wang J, Li Y, Chai S, et al. Bandwidth-aware application mapping for NoC-based MPSoCs [J]. Journal of Computational Information Systems, 2011, 7(1): 152-159

[7] Zhao Peng. Research on the application mapping for multi-processor system-on-chips [D]. Changsha: National University of Defense Technology, 2010 (in Chinese)
(赵鹏. 多处理器 SoC 应用映射关键技术研究[D]. 长沙: 国防科学技术大学, 2010)

[8] Wang Xiaohang. Cross-layered application layer optimization framework for networks-on-chips [D]. Hangzhou: Zhejiang University, 2011 (in Chinese)
(王小航. 片上互连网络跨层交互的应用层优化框架[D]. 杭州: 浙江大学, 2011)

[9] Dong Wenxiao, Shen Haibin, Quan Li, et al. Power-aware mapping based-on genetic algorithm for Network-on-Chip [J]. Journal of Zhejiang University: Science Edition, 2010, 37(6): 654-656, 669 (in Chinese)
(董文箫, 沈海斌, 全励, 等. 基于遗传算法的片上网络低功耗映射[J]. 浙江大学学报: 理学版, 2010, 37(6): 654-656, 669)

[10] Yi Wei, Wang Jiawen, Pan Hongbing, et al. Ant colony chaos genetic algorithm for mapping task graphs to a network on chip [J]. Acta Electronica Sinica, 2011, 39(8): 1833-1836 (in Chinese)
(易伟, 王佳文, 潘红兵, 等. 基于蚁群混沌遗传算法的片上网络映射[J]. 电子学报, 2011, 39(8): 1833-1836)

[11] Deng Zhi, Gu Huaxi, Yang Yintang, et al. Artificial bee colony based low-energy consumption high-performance mapping in NoC [J]. Journal of Xidian University, 2012, 39(2): 114-119 (in Chinese)
(邓植, 顾华玺, 杨银堂, 等. 基于人工蜂群算法的低能耗高性能 NoC 映射[J]. 西安电子科技大学学报, 2012, 39(2): 114-119)

[12] Tosun S, Ozturk O, Ozen M. An ILP formulation for application mapping onto network-on-chips [C] //Proc of the 3rd Int Conf on Application of Information and Communication Technologies. Los Alamitos, CA: IEEE Computer Society, 2009: 1-5

[13] Srinivasan K, Chatha K S, Konjevod G. Linear programming based techniques for synthesis of network-on-chip architectures [J]. IEEE Trans on Very Large Scale Integration Systems, 2006: 14(4): 407-420

[14] Chou C, Marculescu R. Contention-aware application mapping for Network-on-Chip communication architectures [C] //Proc of the 26th IEEE Int Conf on Computer Design. Los Alamitos, CA: IEEE Computer Society, 2008: 164-169

[15] Hu J, Marculescu R. Energy-aware mapping for tile-based NoC architectures under performance constraints [C] //Proc of the 2003 Asia and South Pacific Design Automation Conf. New York: ACM, 2003: 233-239

[16] Hu J, Marculescu R. Energy-and performance-aware mapping for regular NoC architectures [J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 2005, 24(4): 551-562

[17] Zhou W, Zhang Y, Mao Z. Link-load balance aware mapping and routing for NoC [J]. WSEAS Transactions on Circuits and Systems, 2007, 6(11): 583-591

[18] Lei W, Xiang L. Energy-and latency-aware NoC mapping based on discrete particle swarm optimization [C] //Proc of the 2nd Int Conf on Communications and Mobile Computing. Los Alamitos, CA: IEEE Computer Society, 2010: 263-268

[19] Palermo G, Silvano C, Zaccaria V. Robust optimization of SoC architectures: A multi-scenario approach, embedded systems for real-time multimedia [C] //Proc of the 6th IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia. Los Alamitos, CA: IEEE Computer Society, 2008: 7-12

[20] Stralen P V, Pimentel A. Scenario-based design space exploration of MPSoCs [C] //Proc of the 28th IEEE Int Conf on Computer Design. Los Alamitos, CA: IEEE Computer Society, 2010: 305-312

[21] Carvalho E, Calazans N, Moraes F. Dynamic task mapping for MPSoCs [J]. IEEE Design and Test of Computers, 2010, 27(5): 26-35

[22] Orsila H, Kangas T, Salminen E, et al. Automated memory-aware application distribution for multi-processor system-on-chips [J]. Journal of Systems Architecture, 2007, 53(11): 795-815

[23] Lee S, Yoon Y, Hwang S. Communication-aware task assignment algorithm for MPSoC using shared memory [J]. Journal of Systems Architecture, 2010, 56(7): 233-241

[24] Chou C, Ogras U, Marculescu R. Energy-and performance-aware incremental mapping for network on chip with multiple voltage levels [J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(10): 1866-1879

[25] Chou C, Marculescu R. User-aware dynamic task allocation in Networks-on-Chip [C] //Proc of DATE'08. New York: ACM, 2008: 1232-1237

[26] Chou C, Marculescu R. Run-time task allocation considering user behavior in embedded multiprocessor networks-on-chip [J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 2010, 29(1): 78-91

[27] Bjerregaard T, Mahadevan S. A survey of research and practices of Network-on-chip [J]. ACM Computing Surveys, 2006, 38(1): 1-51

[28] Lee J, Choi K. Memory-aware mapping and scheduling of tasks and communications on many-core SoC [C] //Proc of the 17th Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2012: 419-424

[29] Lee E A, Messerschmitt D G. Synchronous data flow [J]. Proceedings of the IEEE, 1987, 75(1): 1235-1245

[30] Fazzino F, Palesi M, Patti D. Noxim: Network-on-chip simulator [EB/OL]. [2010-10-16]. http://noxim.sourceforge.net

[31] Dick R P, Rhodes D L, Wolf W. TGFF: Task graphs for free [C] //Proc of the 6th Int Workshop on Hardware/Software Codesign. Los Alamitos, CA: IEEE Computer Society, 1998: 97-101

[32] Kahng A B, Li B, Peh L S, et al. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration [C] //Proc of DATE'09. Belgium: European Design and Automation Association, 2009: 423-428

[33] Murali S, Benini L, Micheli G D. Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees [C] //Proc of Asia and South Pacific Design Automation Conf. New York: ACM, 2005: 27-32

[34] Beretta I, Rana V, Atienza D, et al. Run-time mapping for dynamically-added applications in reconfigurable embedded systems [C] //Proc of the 21st Int Conf on Microelectronics. Piscataway, NJ: IEEE, 2009: 157-160

[35] Ji Weixing, Shi Feng, Qiao Baojun, et al. The study of an interconnection network for complex embedded systems [J]. Chinese High Technology Letters, 2007, 17(9): 886-890 (in Chinese)
(计卫星, 石峰, 乔保军, 等. 一种面向复杂嵌入式系统的互连网络研究[J]. 高技术通讯, 2007, 17(9): 886-890)

Wang Yizhuo, born in 1979. PhD. Lecturer. Member of China Computer Federation. His main research interests include computer architecure, parallel programming and embedded system.

Zuo Qi, born in 1983. Master. Software engineer. Her main research interests include computer architecture, parallel computing and cloud computing (zuoqi@bcc.ac.cn).

Ji Weixing, born in 1980. PhD. Associate professor. His main research interests include computer architecture, parallel computing and embedded computing (jwx@bit.edu.cn).

Wang Xiaojun, born in 1979. PhD. Lecturer. His main research interests include computer architecture and embedded system (bitwxjred9915@gmail.com).

Shi Feng, born in 1961. PhD. professor and PhD supervisor. His main research interests include computer architecure and embedded system(bitsf@bit.edu.cn).

2013 年《计算机研究与发展》高被引论文 TOP10

排名	论文信息
1	孟小峰，慈祥. 大数据管理:**概念、技术与挑战**[J]. 计算机研究与发展，2013，50(1)：146-169 Meng Xiaofeng and Ci Xiang . Big Data Management: Concepts, Techniques and Challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-169
2	傅颖勋，罗圣美，舒继武. **安全云存储系统与关键技术综述**[J]. 计算机研究与发展，2013，50(1)：136-145 Fu Yingxun, Luo Shengmei, and Shu Jiwu. Survey of Secure Cloud Storage System and Key Technologies. Journal of Computer Research and Development, 2013, 50(1): 136-145
3	李建中，刘显敏. **大数据的一个重要方面:数据可用性**[J]. 计算机研究与发展，2013，50(6)：1147-1162 Li Jianzhong and Liu Xianmin. An Important Aspect of Big Data: Data Usability [J]. Journal of Computer Research and Development, 2013, 50(6): 1147-1162
4	刘大有，陈慧灵，齐红，杨博. **时空数据挖掘研究进展**[J]. ，2013，50(2)：225-239 Liu Dayou, Chen Huiling, Qi Hong, and Yang Bo. Advances in Spatiotemporal Data Mining. Journal of Computer Research and Development, 2013, 50(2): 225-239
5	廖彬，于炯，孙华，年梅. **基于存储结构重配置的分布式存储系统节能算法**[J]. 计算机研究与发展，2013，50(1)：3-18 Liao Bin, Yu Jiong, Sun Hua, and Nian Mei. Energy-Efficient Algorithms for Distributed Storage System Based on Data Storage Structure Reconfiguration [J]. Journal of Computer Research and Development, 2013, 50(1): 3-18
6	贾冬艳，张付志. **基于双重邻居选取策略的协同过滤推荐算法**[J]. 计算机研究与发展，2013，50(5)：1076-1084 Jia Dongyan and Zhang Fuzhi. A Collaborative Filtering Recommendation Algorithm Based on Double Neighbor Choosing Strategy [J]. Journal of Computer Research and Development, 2013, 50(5): 1076-1084
7	武建佳，赵伟. **WInternet:从物网到物联网**[J]. 计算机研究与发展，2013，50(6)：1127-1134 Wu Jianjia and Zhao Wei. WInternet: From Net of Things to Internet of Things [J]. Journal of Computer Research and Development, 2013, 50(6): 1127-1134
8	余凯，贾磊，陈雨强，徐伟. **深度学习的昨天、今天和明天**[J]. 计算机研究与发展，2013，50(9)：1799-1804 Yu Kai, Jia Lei, Chen Yuqiang, and Xu Wei. Deep Learning: Yesterday, Today, and Tomorrow [J]. Journal of Computer Research and Development, 2013, 50(9): 1799-1804
9	熊金波，姚志强，马建峰，李凤华，李琦. **基于行为的结构化文档多级访问控制**[J]. 计算机研究与发展，2013，50(7)：1399-1408 Xiong Jinbo, Yao Zhiqiang, Ma Jianfeng, Li Fenghua, and Li Qi. Action-Based Multilevel Access Control for Structured Document [J]. Journal of Computer Research and Development, 2013, 50(7): 1399-1408
10	张振海，李士宁，李志刚，陈昊. **一类基于信息熵的多标签特征选择算法**[J]. 计算机研究与发展，2013，50(6)：1177-1184 Zhang Zhenhai, Li Shining, Li Zhigang, and Chen Hao. Multi-Label Feature Selection Algorithm Based on Information Entropy [J]. Journal of Computer Research and Development, 2013, 50(6): 1177-1184

数据来源:中国知网;统计日期:2014-12-18