基于分层超图的可重构体系结构模型

沈来信[1,2] 曾国荪[1] 王 伟[1]
(同济大学嵌入式系统与服务计算教育部重点实验室 上海 201804)[1]
(黄山学院信息工程学院 黄山 245021)[2]

摘 要 计算机应用领域的广泛性导致了应用任务的多样性,同一种体系结构难以适应差异巨大的应用任务。通过分析应用任务的计算、存储和通信资源需求,对实现应用问题的程序结构进行理论分析和关键算粒的提取,形成具有独立功能的子算法簇。利用分层超图对应用程序结构和体系结构进行可视化描述,使用感知算法得到应用程序结构和部件忙闲信息等系统状态,利用体系结构的可重构性,使用决策算法完成子结构对子算法簇的合理配对。通过一个计算实例分析验证了可重构结构的有效性。图同构理论表明,这种可变体系结构是合理的,在解决不同复杂应用问题时,具有高效率、低功耗的特点。

关键词 子算法簇,分层超图,感知算法,可重构,决策算法
中图法分类号 TP302 **文献标识码** A

Reconfigurable Architecture Model Based on Layered Hypergraph

SHEN Lai-xin[1,2] ZENG Guo-sun[1] WANG Wei[1]
(National Engineering & Technology Center of High Performance Computers, Tongji University, Shanghai 201804, China)[1]
(Information and Engineering College, Huangshan University, Huangshan 245021, China)[2]

Abstract The breadth of computer applications has led to the diversity of application tasks, and the same kind of architecture is difficult to adapt vastly different application tasks. The resource requirement of computing, storage and communication of the task was analyzed. By analyzing the program structure theoretical and extracting the key granularity extraction of the real application, we got the sub-algorithm clusters which have independent function. Hierarchical hyper graph was used to visually describe the application structure and architecture, and the perception algorithm was utilized to get the system states such as application features and components busy information, then the architecture reconfigurabies and decision-making algorithm were utilized to achieve a reasonable matching of the sub-algorithm clusters and substructures. A computing instance analysis verifies the validity of the reconfigurable architecture. The graph isomorphism theory shows that this variable architecture is reasonable, has high-efficiency and low power consumption in solving different complex applications.

Keywords Sub-algorithm cluster, Layered hypergraph, Perception algorithm, Reconfigurable, Decision-making algorithm

1 引言

当计算模型、处理过程及对处理部件、存储部件和通信部件的需求都有着很大的差别时,同一种体系结构难以适应差别较大的不同应用,提供的应用服务具有效率低、质量低、能耗高、功耗高等缺点。

李国杰院士指出[1],无论是集成电路、高性能计算机,还是互联网和存储器,2020年前后都会遇到只靠现有技术难以逾越的障碍(信息技术墙),这种信息技术墙指:挖掘并行性和可扩展性困难,信息处理功耗高,复杂信息系统安全可靠性低。

应用任务的多样性决定了解决相应问题的计算机体系结构也应该是多样的,不同的应用对计算、存储、通信3种能力的要求均不一样。为了能够快速、有效地处理大量不同应用类型、不同计算特征和不同到达规律的应用任务,需要对应用任务进行刻画,以说明应用任务自身的各种内在属性,以表征应用任务对计算资源的需求。

邬江兴院士在第三届中国云计算大会上做了"云计算高效能之路"报告[2],报告中指出,基于通用的CPU部件的服务器在互联网应用中难以展现其"高性能",提出了"应用决定结构,结构决定效能"的理念,主旨思想是指部件可变的体系结构,让体系结构去适应不同的复杂应用,而不再是让应用去适

到稿日期:2012-07-21 返修日期:2012-10-28 本文受863专项(2007AA01Z425),973课题(2007CB316502),国家自然基金项目(90718015,60673157),安徽省优秀青年人才基金项目(2012SQRL183)资助。
沈来信(1979—),男,硕士,讲师,主要研究方向为图论、并行算法;曾国荪(1964—),男,博士,教授,博士生导师,主要研究方向为可信软件、并行计算;王 伟(1979—),男,博士,主要研究方向为信任管理、机器学习、信息检索。

应计算机的体系结构。

计算机结构一般分为通用处理机、领域计算处理机、专用处理机和可定制/可重构处理机等。这些处理机或服务器统称为“计算部件”。将一组异构的最适合于特定应用业务以及业务负荷存在剧烈波动需求的计算部件，通过可重构的互联网络连接，构成最合适的计算体系结构，以达到结构匹配应用的效果，实现应用处理计算的高效能。

本文第 2 节给出了当前可重构计算、可重构体系结构、超图的研究现状；第 3 节介绍应用程序结构和体系结构的超图定义与描述；第 4 节讨论了由已知的应用程序超图得到体系结构超图的算法；第 5 节和第 6 节通过实例分析和模型检测进行算法描述和理论验证；最后总结全文，并展望后续的工作。

2 研究现状

可重构技术一直是国内外大学和研究机构的研究热点，主要集中在可重构器件、可重构体系结构和互联方式等。沈绪榜院士等提出了以指令流、数据流、构令流体系结构为坐标建立的新的 IDC 体系结构分类[3]。Shoaib 等提出了在动态的科学应用上可重构混合互联结构[4]。陈左宁院士等提出一种基于多虚空间多重映射技术的并行操作系统，解决了高性能计算系统的可扩展性问题[5]。龚育昌等开发了一种支持可重构混合系统的操作系统，其具有统一的系统对象抽象和通信接口[6]。Kiran 开发了一种可重构计算模型，它分别涉及到可重构结构、模型和算法[7]。Artundo 等使用可选择的光部件的可重构多处理器互联，将可重构光链路加入连接网络中，解决了通信连接的节点瓶颈难点[8]。

超图在描述多元关系时，具有线图不可比拟的优势，同时相对其他建模工具，它兼有图形直观表示和形式化描述与验证的优点。Selvakkumaran 等使用多目标的超图划分算法来最大化子域度的最小值[9]。徐洪珍等使用超图描述软件体系结构，用超图文法演绎软件结构的动态演化过程[10]。宋锐等利用 RSOM 树和类属超图实现了分布式图像检索方法[11]。王忠杰等利用分层超图实现了服务价值依赖模型[12]。Michal Karonski 等实现了随机超图中的阶段转换[13]。

由于应用程序结构的复杂性、多样性和关键粒度不同，本文使用分层超图对其进行建模，并由此得到了体系结构的超图，因此可以根据不同的应用特征得到可重构的体系结构。首先给出应用程序超图和体系结构超图定义。

3 应用程序结构超图和体系结构超图

应用程序是面向用户的问题解决的程序，应用程序结构是应用程序中子算法的组织方式，常用的是 DAG 图（无回路有向树），它能够表达二元关系，但是由于应用程序的复杂性，很多子算法间存在多元关系，再加上 DAG 图本身的无回路特征，因此用 DAG 图表达多元关系非常困难。下面给出普通图（线图）与超图（多元关系图）的比较。

常用的线图每条边只能连接两个节点，但实际应用中多元关系是普遍存在的，线图表达困难。超图中的超边允许连接多个节点，超图是线图的扩展，表达能力更强。下面给出超图的定义和应用程序结构的定义。

定义 1（超图 H，Hypergraph） 超图 H 是一个有序的二元组 $H=\langle N,E\rangle$，其中 V 是图中节点的集合，E 是超边的集合。E 中每一条超边都是 V 的一个非空子集。

定义 2（应用程序结构 AS，Application Structure） 应用程序是完成用户预定目标而设计的程序，一个应用程序可以用多个子算法协作（调用关系或参数传递关系）完成，AS 是描述这些子算法及协作关系的程序组织结构。

一般的二元关系的应用程序图可用 DAG 图表示。

例 1 一个分布式 Web 应用中，关于旅游信息服务的 Web 应用有 3 个服务（Web Service），分别是地图服务（Map service）、旅游计划服务（TripPlanner service）和天气预报服务（Weather Service），每个 Service 包含若干操作（Operation），分别是 op1－op10，如图 1 所示。

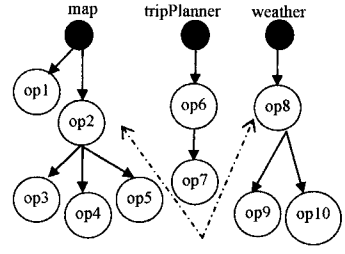

图 1 应用任务 DAG 图

图 1 中虚线表示 op7 服务依赖于 op2 和 op8，这个用 DAG 图无法直接表示出来（如果把服务依赖用实线表示，可能会形成回路，与 DAG 图的无回路有向树性质相悖），同时，多元关系表示的也是 DAG 图的缺陷。

不同的子算法具备不同的属性，如计算量（计算时间、数据结构等）、存储量（空间大小、存储类型等），可能具备不同的计算特征，如符合 torus 结构并行运算、mesh 结构并行运算、串行运算等，子算法之间协作关系具有不同属性，如通信量（通信时间、通信带宽等）、调用与被调用关系、1 对 1、1 对多关系等。

根据以上定义和分析，下面给出应用程序超图定义。

定义 3（应用程序超图 HAS，AS 的超图） 其是一个 6 元组 $HAS=(N,E,p,q,ON,OE)$。其中：N 是节点的集合，即子算法的集合；E 是超边的集合，即子算法之间协作关系集合；$p:E->N^*$ 表示子算法间调用关系到子算法的映射，表示一个子算法调用了其他子算法；其中 * 表示每个子算法可以调用多个子算法；$q:N->N^*$ 表示子算法间服务依赖关系，表示一个子算法服务依赖了其他子算法，其中 * 表示每个子算法可以依赖多个子算法。$ON:N\to\Sigma N$，$OE:E\to\Sigma E$ 分别是子算法上、子算法间关系的属性（时间或权值）函数，它们分别表示相应的计算资源、存储资源和通信资源的需求等。

这里的函数 p 描述了子算法之间的调用关系，这在 DAG 图中是可以描述的。但是函数 q 非常重要，描述了子算法之间的服务依赖关系，这在 DAG 图中却不能表示出来，因为如果加上这种服务依赖关系，就会破坏 DAG 图无回路的特性。由于超图的边又可以是一个子超图，如此递归定义下去，就形成了分层超图，可以根据应用程序的关键粒度要求，得到不同精度的分层超图，因此超图在表达能力和可视化上都要强于 DAG 图。

下面给出例 1 中的 Web 应用的应用程序超图描述，如图 2 所示。

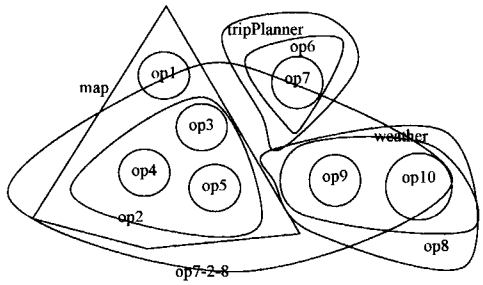

图 2　应用任务超图

在图 2 中，圆圈表示结点，超边把多个结点连接起来，同时，超边又可以是另一个超边的结点，op2 既是连接 op3、op4 和 op5 的边，同时又是 map 边的结点，边 op7-2-8 表示 op7 服务依赖 op2 和 op8。对比图 1 和图 2 可知，超图的描述能力远远强于 DAG 图。

如果复杂应用有大量的子算法，可以把相关耦合性强的子算法形成子算法簇，属性分类可用超图实现，基于同类属性形成超边（分簇），基于群体属性形成聚类，即子算法簇，这样得到的子算法簇超图就是分层超图，顶层为子算法簇之间的关系的超图，第二层是子算法簇超图，是每个子算法簇内部关系超图，第三层是子算法超图，是子算法簇包含的子算法节点的属性超图，用来描述节点和边的组合属性，每个属性看作节点，属性间关系看作是边。

在对应用程序进行超图定义后，下面给出体系结构定义、体系结构超图的定义。

定义 4（体系结构 *AC*，Architecture）　即把一些部件（硬件、子系统或具有独立软件功能的结构等）按照某种方式组织在一起的结构，通过这些部件相互协作，完成指定的任务。

定义 5（*HAC*，AC 的超图）　*AC* 的超图定义为一个 6 元组 $HAC=(N1,E1,p1,q1,ON1,OE1)$。其中：$N1$ 是节点的集合，即部件的集合；$E1$ 是超边，即部件间互联关系的集合；$p1:E1->N1^*$ 表示部件间互联关系到部件的映射，其中 * 表示每个互联关系可以连接多个部件，$p1$ 主要描述的是部件间调用关系互联；$q1:N1->N1^*$ 表示部件到部件的映射，其中 * 表示每个部件可以依赖互联多个部件，$q1$ 主要描述的是部件和部件之间传递消息的互联；$ON1:N1\rightarrow\Sigma N1$，$OE1:E1\rightarrow\Sigma E1$ 分别是部件和部件间互联的属性（标记或权值）函数，其中 $OV1$ 表示一个部件的属性，如计算能力、存储能力等，$OE1$ 表示一个互联关系的属性，如通信能力、价格等，也可以是一个多元组，如通信能力 $e1$、忙闲信息 $e2$ 等。

若干个部件通过互联就形成了一个子结构，这时它就和子算法簇形成了对应，这时它既是结构中的一个点，也是子结构中的一条边，如 $E1'$（部件 $N1'$，部件 $N2'$，…），子结构中的节点或边，如果有组合属性（多种部件或多种互联类型）时，又可以设计成下一层的子结构，这样就形成了体系结构的分层超图。

这样我们就得到了应用程序和体系结构的超图描述。下面根据已知的应用程序超图生成体系结构超图，这里需要考虑应用程序的计算特征、计算部件种类、负载高低和个数。

4　可重构体系结构模型设计

为了适应不同的应用任务，需要设计不同的体系结构去匹配。为了更好地表达应用程序结构与体系结构，上一节中已经定义了它们的超图描述方式，同时考虑到应用任务的复杂性，在不同关键粒度情况下，需要设计不同精度的超图，这样就可得到分层超图。下面我们设计了一个根据应用程序结构得到可重构体系结构的计算模型，它可以用抽象的 6 元组 $\langle HAS,HAC,COMP,LE,SE,DS\rangle$ 表示，成员变量解释如下：

其中：*HAS* 表示应用程序超图，由应用程序算法特征得到；*HAC* 表示体系结构超图；*COMP* 表示计算部件集合，包括通用处理机、领域计算处理机、专用处理机、可定制/可重构处理机、其他独立的处理部件、通信部件等；*LE* 表示负载函数，其类型定义为 $LE:TIME->[0,1]$；*TIME* 表示时间类型；[0,1]表示 0 到 1 的闭区间，用于表示当前每个部件的负载高低；*SE* 表示感知函数，类型为 $SE:TIME->State$，可得到当前每个应用程序子算法簇的计算特征（不同特征匹配不同的计算部件）；*DS* 表示决策函数，根据子算法特征和部件或元结构的状态，把子算法分配到适合其计算特征的部件或元结构上运算。计算模型执行流程如下：

（1）模型初始化。首先对应用程序结构进行分析，得到若干子算法，根据关键粒度要求（子算法数与对应的处理部件数的比值），使子算法之间耦合性强的子算法聚类，形成子算法簇，得到应用程序超图 *HAS*；对通用处理机、专用处理机等计算部件进行初始化配置，得到部件集合 *COMP*，置 *HAC* 为空。

（2）感知算法：利用感知算法，分析应用程序超图中第二层节点（即子算法）计算特征及资源需求，得到各子算法的计算特征 *state*；分析部件集合 *COMP* 中部件的负载情况，得到每个部件的负载高低，将负载低于阈值的置为可用。

（3）决策算法：利用当前得到的 *HAS* 中的子算法节点的计算特征 *state* 和部件集合 *COMP* 中部件负载情况 *LE*，使用决策函数 *DS*，在 *COMP* 中取出适合子算法计算特征的部件或元结构去匹配 *HAS* 中的节点，并把部件或元结构加入到 *HAC* 和派生结构中。此处的元结构指的是并行体系结构，如 *mesh*、*torus* 等。

（4）不断执行（3），直到 *HAS* 中所有子算法节点都有对应的部件或元结构匹配。

（5）根据 *HAS* 中边的关系，使得 *HAC* 中对应的部件完成链路互联关系，得到对应的派生结构，即体系结构超图的第二层和第三层。

（6）不断执行（5），直到 *HAS* 中所有边都有对应的通信连路对应。此时得到子算法的派生结构超图 *HAC*。

（7）继续对每个子算法簇超图执行（2）－（6）操作，即完成第一层（子算法簇层）的设计，使得每个子算法簇有一个适合其计算的子结构去匹配，这样得到体系结构的第一层超图。算法结束。

在第（3）步中，主要根据应用程序结构的关键粒度，决定是否需要进行第三层的分层，如果子算法计算特征仅仅需要串行计算，那么对于这样的子算法只需分配一个合适的计算部件即可，没有必要分配一个计算体系结构去匹配。如果所有的子算法都不需要进一步划分，就不需要第三层分层，同样，如果子算法的属性包含多种计算特征或存储特征，这时部件又足够时，也可进行第三层分层。这些都是由感知算法根

据应用程序结构和部件的状态决定的。

模型中感知算法主要完成对计算任务的实现算法的计算特征和资源需求的获取，包括计算特征、存储特征和通信特征等，不同计算特征的子算法在不同部件或元结构上的运行效率差异很大。假设子算法 2 是一个大矩阵乘法运算，在专用处理器 GPU 或 Cell 上的运算效率是通用处理器的 50～100 倍。当用串行矩阵相乘算法时，时间复杂度为 $O(n^3)$；用 mesh 结构上的并行算法时，时间复杂度为 $O(3n)$；用 torus 结构上的并行算法时，时间复杂度为 $O(n)$。

决策算法完成子算法与部件或元结构的合理匹配：$S=\{s_1,s_2,\cdots,s_n\}$ 为子算法集合，$G=\{g_1,g_2,\cdots,g_n\}\cup\{o_1,o_2,\cdots,o_n\}$ 为部件和元结构集合，$P=\{p_1,p_2,\cdots,p_n\}$ 为子算法的计算特征集合，$L=\{l_1,l_2,\cdots,l_n\}$ 为 G 中部件或元结构的负载信息，DS 为决策算法，$S\times P\times L\rightarrow G$，即对于子算法 s，根据其计算特征 p 与部件或元结构负载 l，将其分配到适当的部件或元结构 g 上处理。

在大矩阵乘法中，我们就设计专用处理部件 GPU 或元结构 torus 结构去匹配该运算，其余子算法与部件或元结构的匹配过程类似。

5 实例分析

通过以上定义和分析，我们能够根据上一节定义的计算模型，由一个不同的应用得到一个和它匹配的可重构的体系结构。下面对该计算模型进行实例分析，通过一个实际的应用去分析该计算模型的执行过程。

例 2　我们把例 1 中 3 个主要功能算法分别概称为子簇 1、子簇 2 和子簇 3。其中子簇 2 与子簇 1 有信息传递，子簇 2 与子簇 3 有信息传递。子簇 1 包含 3 个子算法，分别是子算法 1、子算法 2 和子算法 3；子算法 1 与子算法 2、子算法 1 与子算法 3 分别有信息传递。子簇 2 和子簇 3 也分别包好若干个子算法，此处略去。该 Web 应用的超图描述如图 3 所示。

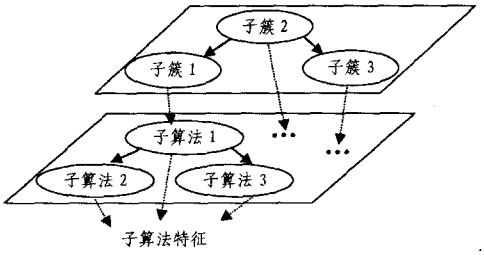

图 3　Web 应用程序分层超图

从图 3 中可以看出，子簇 1、子簇 2 和子簇 3 以及它们之间的关系构成了分层超图的第一层，此时子簇 1、子簇 2 和子簇 3 是结点，整个应用是一个超边。子簇 1 由子算法 1、子算法 2 和子算法 3 构成，此时，子算法之间的关系构成了分层超图的第二层，此时子算法 1、子算法 2 和子算法 3 等是点，而子簇 1、子簇 2 和子簇 3 是超边。这就构成了两层分层超图。

每个子算法又都有各自的计算特征，我们根据感知函数，得到子算法的计算特征，在设计子算法对应的派生子结构时按照这个计算特征，设计相应合理的体系结构去匹配它。

根据以上分析和可重构体系结构模型的计算模型执行流程，可得到该 Web 应用的体系结构超图，如图 4 所示。

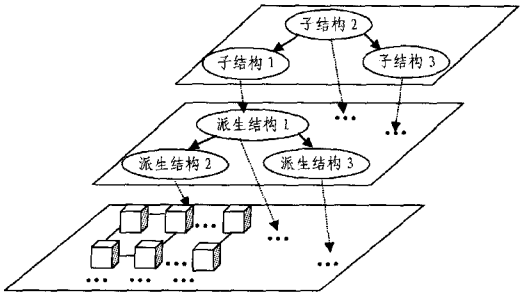

图 4　Web 应用的体系结构超图

图 4 中，子结构 1、子结构 2 与子结构 3 分别对应子簇 1、子簇 2 和子簇 3，子结构之间的关系也和子簇间的关系一致，这是体系结构的第一层超图。子结构 1 又派生出 3 个派生结构，即派生结构 1、派生结构 2 和派生结构 3，分别对应子算法 1、子算法 2 和子算法 3。3 个派生结构之间的关系和 3 个子算法间的关系一致。对子结构 2、子结构 3 同样处理，得到它们的派生结构，这样就得到了体系结构的第二层超图。

同时，根据感知算法，我们知道了各个子算法的计算特征，可根据这个计算特征分配相应的部件或子结构去匹配该子算法。例如对于上面的子算法 2，它的计算特征是需要一个 torus 结构，因此在体系结构超图的第三层就设计相应的子派生结构去响应相应的子算法计算，这时我们就在第三层设计一个 torus 结构，满足子算法 2 的计算需求，同样，对其他子算法按照这个规则匹配下去，就得到了体系结构的第三层。在这一层里，子派生结构中的元组就是计算部件，包括通用处理机、领域计算处理机、专用处理机、可定制/可重构处理机等，根据其不同的计算需求和特征，分配不同的计算部件或派生结构。

下面从图同构角度论述计算模型的合理性和效率。

6 利用图同构理论论证计算模型的合理性

首先给出一个定理，当应用程序超图与体系结构超图同构时，体系结构最匹配相应的应用程序结构，此时计算效率高、功耗低。

定理 1　当应用程序超图与体系结构超图同构时，计算效率最优。

证明：如果两个超图同构，根据感知算法和决策算法，即两个超图中结点一一对应，就是说每个子算法簇都对应着一个最合理的子结构，同时子结构间的互联关系也和应用程序超图中的边对应，没有增加也没有减少，那么子结构之间的通信关系是满足完备性和最小性的。同时，根据子算法的计算特征状态 state，把每个子算法分配给不同的合理的派生结构，这样即达到并行计算，并且是最适合其计算特征的某类型部件或体系结构，没有子算法需要等待处理部件，也没有子算法被分配过多的处理部件，所有的子算法都是按需分配，在这种情况下，应用程序的任务完成效率是最高的，资源消耗小，从而达到了最优效率。

下面给出应用程序超图与体系结构超图可同构的论证。我们分别从两个角度来论述，即体系结构新建时和体系结构已存在时，如何应对应用程序结构的变化。

第一种情况：若体系结构没有建立，对应的体系结构超图也没有，则可以根据应用程序超图生成体系结构超图。对于

一个给定的应用任务，根据对其应用程序结构的分析，得到它的应用程序超图，然后利用上节中的计算模型，得到对应的体系结构超图，对于体系结构超图的第三层，把每个子派生结构看成是第二层超图的一个结点，这样就和应用程序超图中的子算法对应起来，这可以由子算法的关键粒度决定。

第二种情况：若某种应用的体系结构已建立，就得到了对应的体系结构超图，如果这时应用程序结构发生改变，则需要进行的是超图演化模型，那么需要对体系结构超图进行图变换，利用感知算法和智能决策算法，依据应用程序结构超图中的点或边的变化情况，对应修改体系结构超图中的点（子结构）或边（子结构间的互联关系），使体系结构超图保持与应用程序超图的同构关系。

通过以上方式，我们始终使体系结构超图与应用程序结构超图同构，这样，就能使得不同的应用与其最适合的体系结构匹配，实现可重构的体系结构，以满足不同应用需求。把该计算模型在我们开发的原型样机上进行了实际测试，结果表明，这种可重构计算模型是合理有效的。

结束语 在互联网复杂多样的应用中，单一的体系结构已经不能胜任，对于不同的计算任务，需要构建可重构的体系结构去匹配。我们首先用分层超图刻画应用程序结构和体系结构，然后对不同的应用程序结构超图利用超图的同构原理得到相应匹配的体系结构超图，使得应用程序超图中的子算法簇和关系与体系结构中的子结构和关系达到一一对应，达到整体上的并行计算和最优结构。同时对于子算法簇中的子算法，利用决策算法，在体系结构中使用最适合的子派生结构去匹配，从而达到局部的最优计算效果，这样就能使得不同的应用得到可重构的体系结构匹配，达到了效率最优的结果。

参 考 文 献

[1] 李国杰. 信息科学技术的长期发展趋势和我国的战略取向[J]. 中国科学：信息科学，2010，40(1)：128-138

[2] 邬江兴. 云计算高效能之路[R]. 第三届中国云计算大会报告. 2011

[3] 沈绪榜，等. 计算机体系结构的分类模型[J]. 计算机学报，2005，28(11)：1759-1766

[4] Shoaib K. Reconfigurable hybrid interconnection for static and dynamic scientific applications[C]// Proceedings of the ACM International Conference on Computing Frontiers. 2007：183-194

[5] 陈左宁，金怡濂. 基于多虚空间多重映射技术的并行操作系统[J]. 软件学报，2001，12(10)：1562-1568

[6] 乔磊，齐冀，龚育昌. 一种支持可重构混成系统的操作系统设计与实现[J]. 计算机学报，2009，32(5)：1046-1054

[7] Kiran B, Viktor K. Reconfigurable computing: architectures, models and algorighms[J]. Current Science, Special Section on Computational Science, 2000, 78(7): 828-837

[8] Artundo I, et al. Selective optical broad-cast component for reconfigurable multiprocessor interconnects[J]. IEEE Journal on Selected Topics in Quantum Electronics: Special Issue on Optical Communication, 2006, 12(4): 828-837

[9] Selvakkumaran N. Multiobjective hypergraph partitioning algorithms for cut and maximum subdomain-degree minimization [J]. IEEE Transactions on Computer Aided Design of Intergrated Circuits and System, 2006, 25(3): 504-517

[10] 徐洪珍，曾国荪. 基于超图文法的软件体系结构动态演化[J]. 同济大学学报，2011，39(5)：745-750

[11] 宋锐，等. 基于 RSOM 树和类属超图的分布式图像检索方法[J]. 信号处理，2009，25(8A)：264-267

[12] 王忠杰，等. 基于分层超图的服务价值依赖模型[J]. 计算机集成制造系统，2011，17(8)：1834-1843

[13] Karonski M. The phase transition in a random hypergraph[J]. Journal of Computational and Applied Mathematics, 2002(142): 125-135

(上接第 25 页)

对服务器状态及其切换进行建模，利用马尔可夫链构造服务器状态转换过程，使用排队论等相关数学分析技术建立服务器的能耗模型和性能模型，进而对状态的管理进行合理设置，达到降低能耗的目的。该策略在保证用户的请求能够及时得到响应的同时，使服务器能够获得休眠状态所带来的节能效果，又不会导致频繁地进行状态切换，从而产生额外的能耗。分析和实验验证了本文策略的有效性和优越性。

服务器的能耗和性能随着阈值的变化而变化，如何优化设置甚至动态设置状态转换中的阈值参数，找到使得能耗节省和性能保证的最优阈值平衡点是下一步的研究方向。另外如何优化调度各个服务器，从而进一步优化能耗，也是未来研究的方向。

参 考 文 献

[1] 黄海林，范东睿，许彤，等. 嵌入式处理器中访存部件的低功耗设计研究[J]. 计算机学报，2006，29(5)：815-821

[2] Gunaratne C, Christensen K, Nordman B, et al. Reducing the Energy Consumption of Ethernet with Adaptive Link Rate[J]. IEEE Transctions on Computers, 2008, 57(4): 448-461

[3] 吴琦，熊光泽. 非平稳自相似业务下自适应动态功耗管理[J]. 软件学报，2005，16(8)：1499-1505

[4] Chen Hua-cai, Jin Hai, Shao Zhi-yuan, et al. ClientVisor: Leverage COTS OS Functionalities for Power Management in Virtual Execution Environments[M]. Washington D C. USA: ACM Press, 2009: 62-71

[5] Nathuji R, Schwan K. Virtual Power: Coordinated Power Management in Virtualized Enterprise Systems[C]// Proc. of the 22nd ACM Symposium on Operating Systems Principles. Stevenson, WA, USA: ACM Press, 2007: 265-278

[6] 郭兵，沈艳，邵子立. 绿色计算的重定义与若干探讨[J]. 计算机学报，2009，32(12)：2311-2319

[7] 林闯，田源，姚敏. 绿色网络和绿色评价：节能机制、模型和评价[J]. 计算机学报，2011，34(4)：593-612

[8] Gupta M, Grover S, Singh S. A Feasibility Study for Power Management in LAN Switches[C]// Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04). Berlin, Germany, 2004: 361-371

[9] Anastasi G, Conti M, Gregori E, et al. A Performance Study of Power-saving Polices for Wi-Fi Hotspots[J]. Computer Networks, 2004, 45: 295-318

[10] Liao W H, Yen Wen-ming. Power-saving Scheduling with a QoS Guarantee in a Mobile WiMAX System[J]. Journal of Network and Computer Applications, 2009, 32: 1144-1152