基于异构多核平台的同步数据流图帕累托优化与调度

顾玉磊[1,2] 朱雪阳[2] 晏荣杰[2] 张广泉[1]
(苏州大学计算机科学与技术学院 苏州 215006)[1]
(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)[2]

摘 要 同步数据流图被广泛用于多媒体和数字信号处理程序等流应用程序的建模。流应用程序须达到一定吞吐量才能流畅运行,利用异构多核处理器来进一步提高流应用程序的吞吐量已经成为当今嵌入式系统的发展趋势,但是提高吞吐量往往伴随着能耗的增加。为了解决这个问题,基于异构多核平台的同步数据流图系统模型,给出了求解所有能耗和吞吐量的帕累托优化点及其相应静态调度的方法。首先将系统模型转换为时间自动机网络,并将分析目标转换为时序逻辑公式;再使用实时模型检测工具UPPAAL寻找解决方案;最后对UPPAAL返回的结果进行分析,找出满足要求的调度。由于模型检测方法可对问题空间进行穷尽搜索,该方法得到的结果是精确的。该方法可帮助设计者在系统开发早期了解系统能耗和吞吐量的量化关系,有利于缩短系统的开发周期,降低开发成本。

关键词 同步数据流图,异构多核平台,帕累托优化,调度,模型检测
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.007

Pareto Optimization and Scheduling of Synchronous Dataflow Graphs on Heterogeneous Multicore Platform

GU Yu-lei[1,2] ZHU Xue-yang[2] YAN Rong-jie[2] ZHANG Guang-quan[1]
(School of Computer Science and Technology, Soochow University, Suzhou 215006, China)[1]
(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract Synchronous dataflow graphs (SDFGs) are widely used to model streaming applications such as multimedia and digital signal processing applications. Streaming applications are usually required to reach a high throughput to guarantee smooth running. Using heterogeneous multicore processors to improve the throughput of streaming applications has become a feasible solution. However, a higher throughput is usually achieved with the increase of energy consumption. We presented a method to explore the Pareto space of energy consumption and throughput and find the schedule of each Pareto point. A system model includes an SDFG description for the application and a heterogeneous multicore platform. The system model is transformed to a network of timed automata (NTA) and the optimization goals are formalized as temporal logic formulae. The NTA and formulae are then used as the input of the real time model checking tool UPPAAL. The Pareto points and corresponding schedules by the trace provided by UPPAAL can be obtained. Our method is exact, which is benefited from the exhaustive search nature of model checking technique. The method can be used to help designers to understand the quantitative relationship between energy consumption and throughput in the design period, shortening development cycles and reducing costs of system developments.

Keywords Synchronous dataflow graphs, Heterogeneous multicore platform, Pareto optimization, Scheduling, Model checking

1 引言

电子系统开发过程中,工程师需为设备设计出能耗较低、吞吐量较高的调度,以满足用户需要,但日益复杂的系统使工程师面临着巨大的挑战。当今电子设备中的一类重要程序是流应用程序,如音、视频处理程序。同步数据流图(Synchronous Dataflow Graphs, SDFGs)[1]被广泛用于流应用程序的建模。基于模型的性能分析优化方法能够帮助工程师在设计阶段发现并解决问题,从而有效缩短电子系统的开发周期,降低开发风险和成本。

本文基于同步数据流图和异构多核执行平台的系统模型对流应用程序在异构多核处理器上的执行问题进行建模,然后把系统模型转换为时间自动机网络,将分析目标形式化为时序逻辑公式,利用UPPAAL进行模型检测,获得满足性质

到稿日期:2014-10-12 返修日期:2014-12-07 本文受973课题(2014CB340701),江苏省自然科学基金(BK2011281),苏州市应用基础研究计划(SYG201241),江苏省研究生科研创新计划(KYLX_1247)资助。
顾玉磊(1991-),男,硕士生,主要研究方向为高性能计算和形式化方法,E-mail:guyl@ios.ac.cn;朱雪阳(1971-),女,博士,助理研究员,主要研究方向为嵌入式系统设计和形式化方法,E-mail:zxy@ios.ac.cn;晏荣杰(1977-),女,博士,助理研究员,主要研究方向为实时系统模型及验证;张广泉(1965-),男,教授,CCF高级会员,主要研究方向为网络软件工程、形式化方法、云计算和物联网(CPS)。

约束的仿真路径，由路径得到相应的静态调度、能耗和吞吐量，最终迭代求解处理器总能耗和流应用程序吞吐量的帕累托优化与调度问题。本文算法均集成于工具 iDFOS[2] 中，用户可直接在此工具中对同步数据流图在多核平台上的执行问题进行分析和优化。

2 相关工作

Karp 利用最大关键环(Max Cycle Mean, MCM)[3]、Groote 等人利用极大加代数[4]方法求解吞吐量，但都需将同步数据流图转化为同构同步数据流图(Homogeneous SDFGs)，其规模可能是原图的指数倍。Stuijk 等人在无限处理核个数的前提下计算吞吐量和缓存的帕累托优化点[5]。通过时间自动机和模型检测方法，Fakih 等人计算了异构多核系统中总线的最大延迟[6]。Madsen 等人分析了任务图在异构多核平台执行时的缓存占用和能耗大小[7]。而本文解决的是能耗和吞吐量的帕累托优化与调度问题。

3 系统模型定义与转换

3.1 系统模型与调度

同步数据流图由节点和边的集合构成，每个节点表示一个进程或处理模块，节点执行开始时会消耗一定数量的数据，执行结束时会生成一定数量的数据，每个节点在不同处理核上的执行时间可不同。边用于存储节点生成的数据：每条边有初始数据个数；边的源节点执行一次生成的数据个数，称为输出数率；边的目标节点执行一次消耗的数据个数，称为输入数率。其中各边的数率可不同。

当节点所有输入边上的数据个数均大于或等于输入数率时，节点允许执行。节点执行开始时，需在所有输入边上消耗与输入数率相等的数据。节点执行结束时，在所有输出边上生成与输出数率相等个数的数据。当节点没有输入边时，表示节点随时可以执行。

图 1 是表达式 $D\times(C\times(A+B))+(A+B)+(C\times D)$ 对应的同步数据流图 G_{AM}(虚线不属于同步数据流图)[8]。它共有 6 个节点和 6 条边。每条边的输入和输出数率均为 1，在图中省略。所有边的初始数据个数均为 0。节点 T_0 和 T_1 随时可以执行。当边 e_0 至少有 1 个数据时，T_2 才可以执行，T_2 执行开始时会消耗 e_0 上的一个数据，执行结束时在边 e_3 上生成一个数据。各节点在不同类型处理核上的执行时间如表 1 所列，如 T_0 在 ptA 和 ptB 类型处理核上的执行时间为 2 和 5。

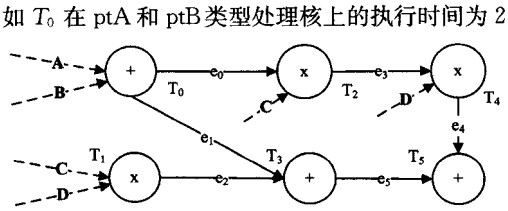


图 1 加乘运算同步数据流图 G_{AM}

表 1 节点在不同类型处理核上的执行时间

	T_0	T_1	T_2	T_3	T_4	T_5
ptA	2	3	3	2	3	2
ptB	5	7	7	5	7	5

执行平台由一批处理核组成，同一类处理核处于空闲和处理状态的功耗相同。表 2 所列的执行平台 Arch1 由一个 ptA 处理核和一个 ptB 处理核组成。

表 2 执行平台 Arch1

处理核类型	功耗		个数
	空闲	处理	
ptA	10	90	1
ptB	20	30	1

系统模型 $M(SDFG, Arch)$ 由同步数据流图和执行平台组成。如加乘运算同步数据流图 G_{AM} 在异构双核处理器 Arch1 上执行的系统模型可由 $M_{AM}=M(G_{AM}, Arch1)$ 表示。

本文考虑有效的同步数据流图，即不会出现死锁或缓存溢出。同步数据流图经过一次迭代后，所有边的初始数据个数与迭代前相同，不会影响下一次迭代。一次迭代中每个节点 a 的执行次数构成的向量称为迭代向量(Repetition Vector)，记为 $q(A)$。

系统模型的调度是指同步数据流图在执行平台上的多次迭代的时间安排和处理核映射，每个节点 a 的每次执行可由调度记录 $SchR(a,p,st,et)$ 表示，即将节点 a 分配到处理核 p 上执行，执行开始时间为 st，执行结束时间为 et。

本文的目的是调度系统模型，以获得能耗和吞吐量帕累托最优的调度。

3.2 系统模型转换到 UPPAAL 时间自动机网络

时间自动机[9](Timed Automata, TA)由六元组 $\langle L, X, V, E, Inv, l_0\rangle$ 定义。L 为有穷节点集合，X 为时钟集合，V 为整型变量集合，$E\subseteq L\times\Omega(X,V)\times U(X,V)\times L$ 为边集合，$Inv: L\to\Omega(X,V)$ 为每个节点指派约束作为不变式，$l_0\in L$ 表示初始节点。记 $k\in\mathbb{N}$，$x\in X$，$\sim\in\{<,\leqslant,=,>,\geqslant\}$，$\varepsilon$ 为常数或变量表达式，$\mathbb{N}$ 为非负整数集合。$\Omega(X,V)$ 表示 X 和 V 上的约束，可由 $g::=true\mid x\sim k\mid\varepsilon_1\sim\varepsilon_2\mid g_1\wedge g_2$ 描述。$U(X,V)$ 用于更新时钟和变量值。

$TA=\langle L, X, Z, V, E, Inv, l_0\rangle$ 对应一个执行系统 $\nabla(S, s_0, \to)$，S 为状态集合，$s_0\in S$ 为初始状态。状态由三元组 (l, x, v) 表示：l 是时间自动机的一个节点；$x: X\to R_0^+$ 记录每个时钟的一个非负实数值；$v: V\to\mathbb{Z}$ 记录每个变量的一个整数值。用 $x+\delta$ 表示将 x 中所有时钟值加 δ，$\alpha\vdash\beta$ 表示 α 中的时钟和变量满足约束 β，$g\subseteq\Omega(X,V)$，$u(x,v)\subseteq U(X,V)$，变迁关系 $\to$ 包含的两种变迁如下：

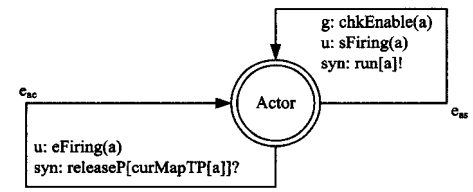
- 延时变迁(delay transition)：$(l,x,v)\to(l,x+\delta,v)$，当且仅当 $\forall\delta': 0\leqslant\delta'\leqslant\delta\Rightarrow(x+\delta', v)\vdash Inv(l)$，其中 δ 为正整数。
- 离散变迁(discrete transition)：$(l,x,v)\to(l',x',v')$，当且仅当 $\exists e=(l,g,u,l')\in E$，并且 $(x,v)\vdash g$，$(x',v')=u(x,v)$，$(x',v')\vdash Inv(l')$。

一条路径(trace)是状态和变迁的交替序列 $s_0\to s_1\to\cdots$，路径长度可以有限或者无限，$\to$ 可以是延时或离散变迁。

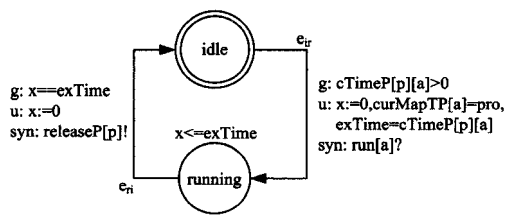
时间自动机网络[10](Network of Timed Automata, NTA)由多个时间自动机组成，可由 $nta_M(TA_0\parallel TA_1\parallel\cdots\parallel TA_{n-1}, K)$ 表示，其中 $\parallel$ 表示并发，K 为共享时钟和变量。时间自动机网络在 UPPAAL 中由声明(Declarations)、模板

(Template)和系统声明(System Declarations)组成:声明部分声明系统中的共享时钟和变量,被所有时间自动机共享;模板部分定义一组时间自动机模板;系统声明部分声明时间自动机网络的具体组成,即由每个模板实例化出的多个时间自动机共同组成描述系统的时间自动机网络。

UPPAAL 时间自动机模板中的条件表达式(guard)用于判断时钟和变量是否满足约束;更新表达式(update)用于更新时钟和变量;同步表达式(synchronization)形如 *sig*!和 *sig*?,其中 *sig*!表示发送同步信号,*sig*?表示接收同步信号。下文中用(g,u,*syn*)表示时间自动机边上的表达式标签,其中 g 为条件表达式,u 为更新表达式,*syn* 为同步表达式,如图 2 所示。



(a)节点时间自动机 TA_a



(b)处理核时间自动机 TA_p

图 2　时间自动机模板

对于一个由含 n 个节点的同步数据流图和含 m 个处理核的执行平台组成的系统模型 $M(SDFG,Arch)$,它与时间自动机网络 $nta_M(TA_a^0 \parallel TA_a^1 \parallel \cdots \parallel TA_a^{n-1} \parallel TA_p^0 \parallel TA_p^1 \parallel \cdots \parallel TA_p^{m-1},K)$等价。节点和处理核时间自动机模板如图 2 所示。在节点时间自动机 TA_a 中,*sFiring*(a)描述节点执行开始时的行为,消耗输入边数据并将节点执行次数增 1;*eFiring*(a)描述节点执行结束时的行为,在输出边生成数据;函数 *chkEnable*(a)用于检查节点是否有足够数据可供节点 a 执行并且执行次数未达到迭代次数的 $q(a)$倍。在处理核时间自动机 TA_p 中,*cTimeP*[p][a]表示节点 a 在处理核 p 上执行所需的时间,*curMapTP*[a]用于存储节点 a 当前分配执行的处理核。

在与系统模型等价的时间自动机网络中,节点 a 允许执行后,还需有空闲的处理核可分配才可以执行。当 TA_a 检查并发现节点允许执行后,TA_a 会通过 *run*[a]!发送同步信号,如果有处于空闲(idle)状态的处理核时间自动机 TA_p 通过 *run*[a]?接收同步信号后,触发 TA_a 的边 e_{as} 和 TA_p 的边 e_{ir},TA_a 消耗所有输入边数据,TA_p 更新时钟和变量后转换到运行(running)状态。当 TA_p 处于运行状态的时间等于节点在该处理核上所需的执行时间时,即节点执行结束后,TA_p 通过 *releaseP*[p]!发送同步信号,TA_a 通过 *releaseP*[p]?接收释放处理核的同步信号,触发 TA_a 的边 e_{ac} 和 TA_p 的边 e_{ri},TA_a 在所有输出边生成数据,节点执行结束,TA_p 转换到空闲状态。

由于系统模型和 UPPAAL 时间自动机网络语义的一致性,系统模型的帕累托优化与调度问题可通过与其等价的时间自动机网络来解决。

以系统模型 $M_{AM}=M(G_{AM},Arch1)$为例,其对应的时间自动机网络为 $nta_{AM}=nta_M(TA_a^0 \parallel TA_a^1 \parallel \cdots \parallel TA_a^5 \parallel TA_p^0 \parallel TA_p^1,K)$。

4　能耗和吞吐量的帕累托优化与调度

实时模型检工具 UPPAAL[11] 可验证系统模型 $M(SDFG,Arch)$对应的时间自动机网络 nta_M 是否满足性质约束,当 nta_M 满足性质约束时,UPPAAL 会返回一条满足性质约束的可行路径,记为 σ。通过路径 σ 可得到系统模型的一个满足约束的调度,由调度可进一步计算该调度的所需能耗及吞吐量。通过在性质中加入能耗约束条件并进行最快路径(Fastest Trace)搜索,对系统模型调度进行优化,最终可迭代求解出所有能耗和吞吐量的帕累托优化点及相应的可行静态调度。

4.1　调度获取及能耗和吞吐量计算

我们关心的性质可由计算树逻辑(Computation Tree Logic,CTL)表达式 $EF\varphi$ 描述,其表示至少存在一条路径,在它的某个状态下条件 φ 为真。将时间自动机网络 nta_M 满足性质 $EF\varphi$ 记为 $nta_M \vdash EF\varphi$。由于节点时间自动机受迭代次数的限制,最终所有时间自动机将进入死锁状态,该性质可用 $EF\varphi$ *deadLock* 描述。因此用工具 UPPAAL 检测 $nta_M \vdash EF$ *deadLock* 时,会返回一条可行路径,在该路径中每个节点的执行次数等于 $q(a)\times iterN$,即同步数据流图进行了 *iterN* 次迭代,也就是说,该路径隐含了系统模型的一种可行的调度。

通过 UPPAAL 检测性质约束后返回的路径 σ,可得到其对应的系统模型调度。对于路径 σ 中由处理核自动机边 e_{ir} 触发的离散变迁集合,其中的每一条变迁对应了一条调度记录。通过边 e_{ir} 上的同步信号 *run*[a]?可得到执行节点;由边 e_{ir} 所在的处理核自动机可得到节点执行时分配的处理核;由变迁源状态的全局时钟可得到节点执行开始时间;执行结束时间可由节点执行开始时间加上节点所需执行时间得到。

吞吐量指单位时间内同步数据流图迭代的次数,可由 $iterN/iterT$ 得到,其中 $iterT$ 表示同步数据流图 *iterN* 次迭代执行的总耗时。因此通过所得调度的最后一条调度记录的结束时间和同步数据流图的迭代次数,就可以计算该调度的吞吐量。

调度的能耗指所有处理核所花费能耗的总和,可由 $\sum_{p\in Pro}[occT(p)\times usingPC(p)+(iterT-occT(p))\times idlePC(p)]$得到。其中 $occT(p)$表示处理核 p 处于运行状态的总时间,$idlePC(p)$和 $usingPC(p)$分别表示处理核 p 处于空闲和运行状态的功耗,Pro 表示处理核集合,所有变量均在时间自动机网络中定义。

4.2　帕累托优化

利用模型检测工具 UPPAAL 的最快路径搜索来检测 $nat_M \vdash EF$ *deadLock*,可优化系统模型调度,获得同步数据流图吞吐量最高的调度。为了得到给定能耗约束值 *ecC* 下吞吐量最高的调度,需在性质中加入能耗约束条件,可表示为 *EF*

$deadLock \wedge con(ecC)$。同步数据流图迭代结束后，执行平台所花费的总能耗应小于或等于能耗约束值 ecC，即 $\sum_{p\in Pro}[occT(p)\times usingPC(p)+(iterT-occT(p))\times idlePC(p)]\leqslant ecC$。由于迭代结束后的执行总耗时 $iterT$ 可由全局时钟 $glbClk$ 得到，因此整理后的能耗约束条件 $con(ecC)$ 如式(1)所示。

$$con(ecC)\equiv_{def} glbClk\leqslant \frac{ecC-\sum_{p\in Pro}[occT(p)\times(usingPC(p)-idlePC(p))]}{\sum_{p\in Pro} idlePC(p)} \tag{1}$$

将二元组(ec,thr)称为记录点，则帕累托优化点是指降低能耗 ec 或提高吞吐量 thr 都会使另一指标更坏的记录点。

通过最快路径搜索检测 $nta_M \vdash EF\ deadLock \wedge con(ecC)$，可得到能耗约束下吞吐量最大的调度。由调度进一步计算该调度下的实际能耗和吞吐量，可得到一个候选帕累托优化点。为了易于描述，下文用 $getMaxThrUnderEC(nta_M, ecC)$描述该过程。

通过算法 1 可得到所有的帕累托优化点，其中变量 *paretoPList* 用于存储所有帕累托优化点，*INT_MAX* 和 *R_MAX* 分别表示最大的整型数和浮点数，*removeLastP* 用于删除最后一个记录点。其首先调用 *getMaxThrUnderEC* 获得一个记录点，将其作为候选帕累托优化点加入 *paretoPList*。通过降低能耗约束，获得下一记录点，称为当前优化点。由于能耗约束值降低，因此当前优化点的吞吐量小于或等于候选帕累托优化点的吞吐量。如果两优化点的吞吐量相等，则将候选帕累托优化点删除，将当前优化点作为候选帕累托优化点加入到 *paretoPList*；否则候选帕累托优化点即为找到的帕累托优化点，并将当前优化点作为下一候选帕累托优化点加入 *paretoPList*。降低能耗约束重复上述过程直至无可行记录点存在。由于能耗约束逐渐降低，因此算法不会遗漏任何帕累托优化点。

算法 1 getAllParetoPoints(nta_M)

```
输入:时间自动机网络 nta_M
输出:所有的帕累托优化点
1. PatetoPList paretoPList =null
2. ecC =INT_MAX, thrC =R_MAX
3. while true
4.   optP=getMaxThrUnderEC (nta_M, ecC)
5.   if optP ==null then
6.     return paretoPList
7.   else
8.     if optP. thr ==thrC then
9.       paretoPList. removeLastP()
10.    end if
11.    paretoPList. add(optP)
12.    ecC =optP. ec-1, thrC =optP. thr
13.  end if
14.  end while
```

例如对系统模型 M_{AM}，可得到如图 3 所示的 4 个帕累托优化点及其对应的可行静态调度，其中 ec 表示能耗，thr 表示吞吐量。

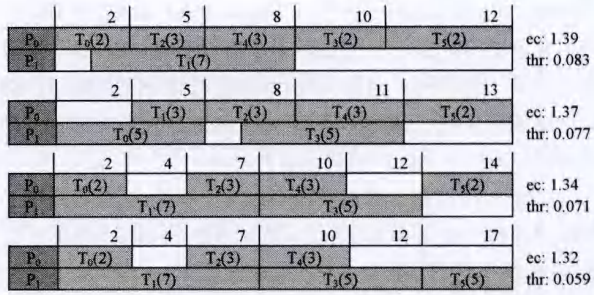


图 3　系统模型 M_{AM} 一次迭代执行的帕累托优化与调度

5　实验

5.1　实验设置

iDFOS[2]是同步数据流图的分析与优化工具，它以 XML 描述的执行平台和同步数据流图作为输入，用户可直接获得各种分析与优化结果而不需要关心优化过程。本文算法已经集成在工具 iDFOS 中，它将系统模型转换为 UPPAAL 时间自动机网络，然后利用工具 UPPAAL 提供的 verifyta 脚本[11]，通过算法 1 获得所有能耗和吞吐量的帕累托优化点及相应的可行静态调度。如果在工具 iDFOS 中输入能耗和吞吐量约束，iDFOS 可以给出小于或等于能耗约束并且大于或等于吞吐量约束的所有帕累托优化点。除了能耗和吞吐量约束外，iDFOS 中还集成了迭代次数等约束。

本文实验在 24MB 缓存、284GB 内存、2.90GHz 主频的服务器上进行。第一个实验的目的是评估不同同步数据流图对本文算法执行效率的影响。第二个实验的目的是评估不同执行平台对本文算法执行效率的影响。本文实验均在 1 次迭代的条件下进行。所有实验的执行平台如表 3 所列，处理核的功耗见表 2，例如执行平台 Arlm3 包含了 1 个 ptA 类型的处理核和 3 个 ptB 类型的处理核。由于实验目的是评估本文算法的可用性，因此实验结果中的吞吐量和能耗的单位可视为一个标准单位。

表 3　实验中系统模型的执行平台

执行平台名		Arlm1	Ar2m0	Ar2m2	Arlm3	Ar3m1	Ar3m3	Ar4m4
核个数	ptA	1	2	2	1	3	3	4
	ptB	1	0	2	3	1	3	4

5.2　实验结果

5.2.1　综合实验

综合实验采用多个应用领域的同步数据流图，包括数字信号处理领域的调制解调器(Modem)应用程序，多媒体领域的 MP3 解码(MP3Decoder)、MP4 解码(MP4Decoder)和音频回波消除(AudioEchoCanceller)应用程序，以及加乘运算(AddMultiply)和二部图(Bipartite)。

本实验中，所有同步数据流图的执行平台均是 Arlm1，实验结果如表 4 所列。它给出了每个系统模型通过算法 1 得到的帕累托优化点数、迭代求解过程中调用 UPPAAL 脚本的次数、执行时间和每个系统模型的最大吞吐量及最小能耗对应的两个帕累托优化点。通过这两个帕累托优化点，可初步了解系统模型的能耗和吞吐量关系。同步数据流图的节点数、边数及迭代向量总和会影响状态空间的大小，进而影响算法的执行效率。通常节点数、边数及迭代向量总和越大，算法

执行时间越长；帕累托优化点数越多，执行时间也越长。从表4可以看出，就大多数同步数据流图在执行平台 Ar1m1 上的一次迭代执行而言，在本实验的环境下，本文算法有较好的执行效率，最长的执行时间仅约为 18s。

表4 执行平台 Ar1m1 上的综合实验结果

同步数据流图名	Modem	MP3Decoder	MP4Decoder	AudioEchoCanceller	AddMultiply	Bipartite
节点/边数	16/19	14/16	5/10	4/6	6/6	4/8
迭代向量总和	48	27	13	70	6	73
吞吐量	0.031	0.05	0.077	0.013	0.059	0.010
最小能耗	3840	2160	520	5600	1320	5840
最大吞吐量	0.031	0.053	0.125	0.018	0.083	0.016
能耗	3840	2190	720	6350	1390	6800
帕累托优化点数	1	2	6	26	4	33
UPPAAL 执行次数	2	4	10	27	5	34
执行时间/ms	18010	4650	120	16580	40	3360

5.2.2 加乘运算实验

加乘运算同步数据流图 G_{AM} 通过本文算法在不同执行平台下的优化结果如表5所列。当处理核个数增加时，算法执行时间相应增加。对于同步数据流图 G_{AM} 及8核处理器平台 Ar4m4，本文算法在本实验环境中的执行时间仅约为 3.6s，因此本文算法在多核执行平台上也有较高的执行效率。

表5 加乘运算在不同执行平台上的实验结果

执行平台名	Ar1m1	Ar2m0	Ar2m2	Ar1m3	Ar3m1	Ar3m3	Ar4m4
吞吐量	0.059	0.1	0.071	0.071	0.091	0.1	0.1
最小能耗	1320	1400	1590	1730	1580	1990	2290
最大吞吐量	0.083	0.1	0.1	0.083	0.1	0.1	0.1
能耗	1390	1400	1690	1870	1590	1990	2290
帕累托优化点数	4	1	2	2	2	1	1
UPPAAL 执行次数	5	2	4	4	4	3	3
执行时间/ms	40	20	240	230	240	910	3640

通过对表5的进一步观察还可以发现一些有价值的信息：当 ptA 类型的处理核个数增加到2时，G_{AM} 达到最大吞吐量 0.1，继续增加 ptA 或 ptB 类型的处理核个数都不会提升其吞吐量，反而会由于处理核空闲时产生的能耗，导致最大吞吐量下的能耗值增加。例如 G_{AM} 在执行平台 Ar2m0 和 Ar3m1 下所能达到的最大吞吐量均为 0.1，执行平台 Ar2m0 产生的能耗为 1400，而 Ar3m1 却达到 1590，因此就 G_{AM} 的一次迭代而言，处理核个数较少的执行平台 Ar2m0 优于 Ar3m1。

结束语 本文对流应用程序在异构多核处理器平台上执行时的能耗和吞吐量的帕累托优化与调度问题进行探讨。将从问题域抽象出的系统模型转换成时间自动机网络，再利用实时模型检测工具 UPPAAL 找到能耗和吞吐量的帕累托优化点及相应的可行静态调度。通过实验评估可以看出，本文算法对大多数同步数据流图在一定核数目平台下的优化分析有较好的可用性。由于篇幅限制，技术细节请参阅文献[12，13]。

参 考 文 献

[1] Lee E A, Messerschmitt D G. Static scheduling of synchronous data flow programs for digital signal processing[J]. IEEE Trans on Computer, 1987, 36(1): 24-35

[2] Institute of Software, Chinese Academy of Sciences. The iDFOS Tool[OL]. (2014)[2014]. http://lcs.ios.ac.cn/~zxy/tools/idfos.html

[3] Karp R M. A characterization of the minimum cycle mean in a digraph[J]. Discrete Mathematics, 1978, 23(3): 309-311

[4] Groote R, Kuper J, Broersma H, et al. Max-plus algebraic throughput analysis of synchronous dataflow graphs[C]// 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA). Cesme, 2012: 29-38

[5] Stuijk S, Geilen M, Basten T. Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs[C]// Proceedings of the 43rd annual Design Automation Conference. San Francisco, 2006: 899-904

[6] Fakih M, Gruttner K, Franzle M, et al. Towards performance analysis of SDFGs mapped to shared-bus architectures using model-checking[C]// Design, Automation & Test in Europe Conference & Exhibition (DATE). Grenoble, 2013: 1167-1172

[7] Madsen J, Hansen M R, Knudsen K S, et al. System-level verification of multi-core timed-automata[C]// Proceedings of International Federation of Automatic Control. Seoul, 2008: 9302-9307

[8] Bouyer P, Fahrenberg U, Larsen K G, et al. Quantitative analysis of real-time systems using priced timed automata[J]. Communications of the ACM, 2011, 54(9): 78-87

[9] Alur R, Dill DL. A theory of timed automata[J]. Theoretical Computer Science, 1994, 126(2): 183-235

[10] Behrmann G, David A, Larsen K G. A tutorial on Uppaal[J]. Formal Methods for Design of Real-time Systems, 2004, 3185: 200-236

[11] Uppsala University, Aalborg University. The UPPAAL Model Checker[OL]. (2005)[2014]. http://www.uppaal.com

[12] Gu Yu-lei, Zhu Xue-yang, Yan Rong-jie, et al. Pareto Optimization and Scheduling of Synchronous Dataflow Graphs on a Heterogeneous Multicore Platform[OL]. (2014)[2014]. http://lcs.ios.ac.cn/~zxy/papers/r14-14.pdf

[13] Zhu Xue-yang, Yan Rong-jie, Gu Yu-lei, et al. Static Optimal Scheduling and Mapping of Synchronous Dataflow Graphs on a Heterogeneous Multiprocessor Platform with Model Checking [OL]. (2014)[2014]. http://lcs.ios.ac.cn/~zxy/papers/r14-12.pdf