doi:10.11887/j.cn.201401030 http://journal.nudt.edu.cn

基于关键路径和任务复制的多核调度算法*

谢志强[1],韩英杰[1],齐永红[1],杨 静[2]
(1.哈尔滨理工大学 计算机学院,黑龙江 哈尔滨 150080;
2.哈尔滨工程大学 计算机学院,黑龙江 哈尔滨 150001)

摘 要:针对目前大多数多核处理器任务分配优化算法没有考虑关键路径上节点对任务完成时间的重要影响,导致任务完成总时间延迟的问题,提出了基于关键路径和任务复制(CPTD)的单任务调度算法。CPTD算法通过复制任务图中fork节点的方式将任务图转化为与之相对应的产品加工树;再在生成的产品加工树中找到关键路径,并采取使关键路径上节点的紧前节点尽早调度的方式,使关键路径上节点尽早开始执行,进而使产品加工树中节点完成时间得以提前,达到缩短任务执行总时间的目的。理论分析表明,CPTD算法能够实现应用程序在多核上充分并行处理,并能缩短任务完成时间。

关键词:单任务;任务复制;关键路径;产品加工树;多核
中图分类号:TP316 **文献标志码**:A **文章编号**:1001-2486(2014)01-0172-06

A scheduling algorithm for multi-core based on critical path and task duplication

XIE Zhiqiang[1], *HAN Yingjie*[1], *QI Yonghong*[1], *YANG Jing*[2]
(1. College of Computer, Harbin University of Science and Technology, Harbin 150080, China;
2. College of Computer, Harbin Engineering University, Harbin 150001, China)

Abstract: Aiming at the problem of current scheduling algorithm for multi-core which fails to consider that the nodes on the critical path have a major impact on the ending time of tasks, leading to the delay of the task completion time; a scheduling algorithm based on critical path and task duplication (CPTD) is proposed. Firstly, the fork-nodes were duplicated to change the task graph into products processing tree, then the critical path in the processing tree were found, and the father nodes of the nodes on critical path were made to work at the earliest time. These operations can advance the start time of nodes on critical path. The purpose of the above operation is to shorten the implementation of the mandate of the total time. Theoretical analysis shows that the algorithm can achieve a single task fully parallel processing on multi-core, and also can shorten the completion time of the tasks.

Key words: single-task; task duplication; critical path; processing flow chart; multi-core

片上多核处理器(Chip Multi-Processor, CMP)是指将多个计算内核集成在一个处理器芯片中,从而提高计算能力。按内核对等与否,CMP可分为同构多核和异构多核;按处理器内核互联方式又可分为共享L2cache类型和基于片上互连类型等[1]。如何通过优化调度算法使处理器上应用程序合理分配,使多核处理器性能优势得以充分发挥,是多核研究的主要方向。

多核处理器任务调度算法已经被证明是NP完全的[2],只能尽量求近优解。目前,多核处理器任务调度主要有任务分配优化、增加L2cache命中率的任务共享存储优化[1]和任务负载均衡优化[3]三个方向的研究。本文只对任务分配优化方面进行分析,其他两个方向暂不进行讨论。任务分配优化方面较为著名的传统调度算法主要有TDMSCL[4],IREA[5],PPA[6],ETDS[7],TDS[8],CPFD[9],PY[10]等。虽然这些算法采用了任务复制(Task Duplication Base,TDB)的方式,通过减少内核间不必要的通信开销,缩短了任务完成总时间,但是这些算法在设计过程中均没有考虑到任务图可以转换为产品加工树。

事实上,当产品生产综合调度优先考虑产品加工树中的关键路径时,可以优化调度结果[11-12]。主要原因是关键路径是产品加工树中执行时间最长的路径,其上节点都对任务完成时间起决定性作用。而多核调度中,任务图通过任

* 收稿日期:2013-04-25
基金项目:国家自然科学基金资助项目(60873019)
作者简介:谢志强(1962—),男,教授,博士后,E-mail:xiezhiqiang@hrbust.edu.cn

务复制方式转换成的产品加工树在节点承接关系及加工树组成上与综合调度中产品加工树模型类同。因此,考虑采用文献[11]和文献[12]中的优先调度关键路径的拟关键路径法,在通过任务复制的方式减少不必要通信开销基础上,进一步缩短任务完成总时间。

由于多核任务调度与综合调度在节点间有无通信开销的问题上有本质的区别,为了充分发挥拟关键路径法的优点,本文提出基于关键路径和任务复制的 CPTD 算法。该算法首先利用任务复制的方式将任务图转换为可减少通信开销的加工树,再通过提出的紧前节点组尽早完成策略,使关键路径上任务尽早开始执行,进而缩短任务完成时间。通过理论分析,CPTD 算法能够明显地缩短任务完成时间。

1　任务调度模型

多核处理器静态任务调度中,一个程序即为一个任务的集合[2]。这些任务之间大都存在依赖关系,而且这种依赖关系在程序开始执行之前就已经存在。加权有向无环图(Directed Acyclic Graph, DAG)可以较好的表示任务之间的这种关系[13]。

DAG 图用 $G=(T,E,t,c)$ 表示,其中 $T=\{T_1,T_2,\cdots,T_i,\cdots,T_n\}$ 表示节点(对应应用程序中的任务)集合,T_i 表示第 i 个节点,n 表示节点总数;$t=\{t_1,t_2,\cdots,t_n\}$ 为图中节点的权值(任务执行时间)的集合;E 为有向边的集合$\{E_{ij}\}$,表示任务 T_i 与 T_j 之间存在通信关系,否则不能直接通信;c 为有向边的权值(任务间通信时间)集合$\{c_{ij}\}$。同一个内核上的任务之间通信开销远小于内核之间的通信开销,因而可忽略不计[14],即 $c_{ij}=0$;不同内核上的任务之间通信时间则不可忽略,可由下式定义[15]:

$$c_{ij}=b+\frac{D(T_i,T_j)}{r} \tag{1}$$

其中,b 为任务通信启动时间,$D(T_i,T_j)$ 为节点 T_i 与 T_j 间数据通信量,r 为内核间的数据传输率。

如 DAG 图中包含 N 个节点,T_N 为终止节点,则程序完成时间 F 可表示为:

$$F=\min(\max(B_N+t_N)) \tag{2}$$

其中,B_N 为节点 T_N 开始时间,t_N 为节点 T_N 执行时间。

本文在算法设计过程中假定处理器内核充足,以 $P=\{P_1,\cdots,P_i,\cdots,P_m\}$ 表示处理器内核集合,P_i 为第 i 个内核,m 为内核总数。

2　调度算法设计

2.1　算法思想

应用程序中的任务之间存在的数据依赖关系,使得前驱任务与当前任务不在同一内核上执行时,会产生通信开销,使任务的开始时间延迟。为最大限度消除通信开销,CPTD 算法中 DAG 任务图的处理方式为:复制任务图中 fork 节点,形成只含 join 节点的任务图,join 图可以转化成一种综合调度中常用的任务图——产品加工树。

本文采用如图 1 所示的产品加工树模型进行算法的进一步设计与分析。图中矩形框内描述分别为:任务序号/任务执行时间,有向边的权值表示两个任务不在同一内核上执行时的通信开销,T_8 为根节点(对应 DAG 图中终止节点),T_1,T_2,T_4 为叶节点(对应 DAG 图中初始节点)。在产品加工树中,每个节点都能虚拟为以自身为根节点的一棵相对独立的产品加工树,便于分析处理。

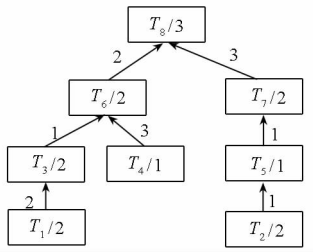

图 1　产品加工树

Fig. 1　Process flow chart

CPTD 算法在对产品加工树调度过程中采用查找并优先调度关键路径上节点的方式。查找关键路径的方法为:计算出各路径的长度(路径上各节点加工时间和通信时间总和),如果最长路径不唯一,则选择节点数最多的路径为关键路径。

关键路径上的节点通常有多个紧前节点序列,这些序列如果不能得到合理的调度,则有可能使关键路径上节点开始时间产生延迟。为了避免这种情况发生,本文提出一种紧前节点组尽早完成策略,该策略的主要思想为:判断当前节点的多个紧前节点序列是否可以合并到同一序列上执行,如合并后使得完成时间延迟则将这些紧前节点序列分别形成单独的调度序列,并分配到独立的内核上执行;否则,将这些紧前节点序列合并到同一序列上执行。

2.2　算法描述

设计 CPTD 算法具体执行步骤如下:

步骤 1:遍历 DAG 任务图,复制 fork 节点,形

成 join 图。

步骤 2:将 join 图转换为产品加工树。

步骤 3:在产品加工树中查找关键路径,并将关键路径上节点依次加入队列。

步骤 4:判断队列是否为空,如不为空,继续向下执行,否则跳转至步骤 13。

步骤 5:取出并删除队列中第一个节点,判断该节点入度是否大于 1。如节点入度不大于 1,继续向下执行;否则,跳至步骤 7。

步骤 6:与紧前节点序列形成调度序列,跳转到步骤 4。

步骤 7:判断不在关键路径上的紧前节点序列是否已形成调度序列。如已形成调度序列则跳转至步骤 10,否则继续向下执行。

步骤 8:判断不在关键路径上的序列中是否含入度大于 1 的节点。如果有入度大于 1 的节点则继续向下执行,否则跳转步骤 10。

步骤 9:将节点序列虚拟为一棵产品加工树,跳转至步骤 3。

步骤 10:拟合并紧前节点序列。如果合并后使得该节点开始时间延迟,继续向下执行;否则跳转至步骤 12。

步骤 11:将紧前节点序列分别形成调度序列,并将当前节点分配到使其最早开始执行的序列上,跳转至步骤 4。

步骤 12:合并紧前节点序列,形成调度序列,跳转至步骤 4。

步骤 13:判断产品加工树中是否有未调度节点。如果有,将该节点序列虚拟为产品加工树,跳转至步骤 3;否则继续向下执行。

步骤 14:为每个调度序列分配内核。

步骤 15:输出调度结果甘特图。

算法流程图如图 2 所示。

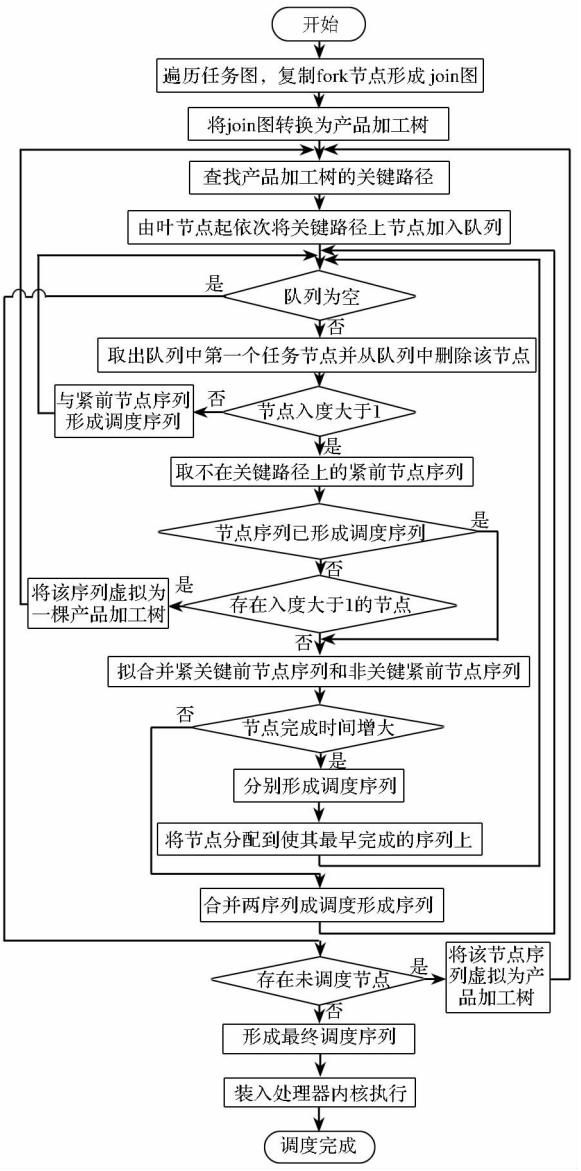

图 2 算法流程图

Fig. 2 Flow diagram of the algorithm

2.3 算法复杂度分析

设 DAG 任务图有 n 个节点。遍历任务图,将任务图转换成产品加工树时,最多需要每个节点变换 1 次,所以任务图转换成产品加工树的时间复杂度为 $O(n)$。

在工艺图中查找关键路径时,需遍历加工树的所有的边和节点,计算 n 个节点和 $n-1$ 条边所形成路径的长度。由于部分边和节点需重复计算,最坏情况下被重复计算 n 次,即计算路径长度的加法次数为 $(2n-1)n$ 次,故确定关键路径的时间复杂度为 $O(n^2)$。

判断关键路径上节点入度数,需计算关键路径上各节点紧前节点数。由于关键路径上最多有 n 个节点,每个节点最多有 $n-1$ 个紧前节点,因此,计算关键路径上各节点的紧前节点数最多计算 $(n-1)n$ 次,即判断关键路径上节点入度数的时间复杂度为 $O(n^2)$。

判断不在关键路径上节点入度的时间复杂度与为判断关键路径上节点入度数类似,为 $O(n^2)$。

判断是否合并紧前工序关键路径上节点序列与非关键路径上节点序列,最坏情况下有 $n-2$ 个工序需要判断,每次判断需比较 2 次,故此步骤需比较 $2(n-2)$ 次。因此,判断合并紧前工序关键路径上节点序列与非关键路径上节点序列的时间复杂度 $O(n)$。

综上所述,CPTD 算法时间复杂度为:$2O(n)+3O(n^2)=O(n^2)$。

3　算法实例分析

3.1　算法调度实例

以上算法分析与设计没有考虑具体调度实例,因此 CPTD 算法不受具体例子约束,即算法具有普遍意义。为方便读者理解本算法,以图 3 所示随机生成的 DAG 图为例,说明 CPTD 算法调度过程。

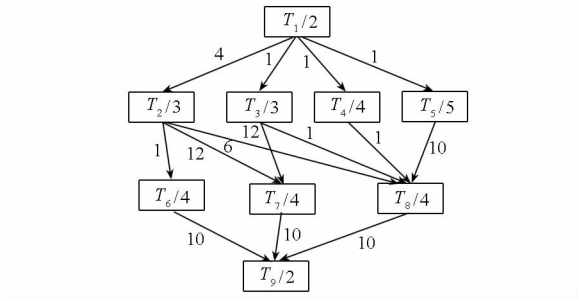

图 3　DAG 图

Fig. 3　DAG graph

首先,采用任务复制的方式将图 3 中 fork 节点分别复制到各自的后继节点上,通过这种节点复制的方式把 DAG 图转换成为图 4 所示的产品加工树。

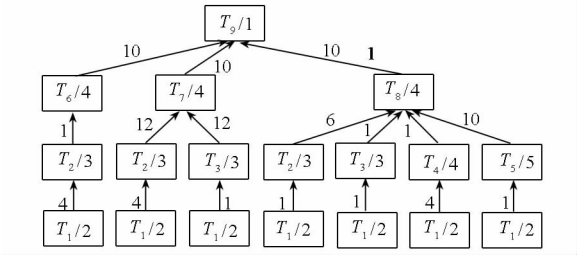

图 4　图 3 对应的产品加工树

Fig. 4　Process flow chart to fig. 3

对得到的产品加工树进行调度,具体过程为:分别计算由叶节点到根节点的各条路径长度,以找到关键路径,如表 1 所示。

表 1　路径列表

Tab. 1　Table for paths

路径	路径长度
(T_1,T_2,T_6,T_9)	25
(T_1,T_2,T_7,T_9)	36
(T_1,T_3,T_7,T_9)	33
(T_1,T_2,T_8,T_9)	27
(T_1,T_3,T_8,T_9)	22
(T_1,T_4,T_8,T_9)	26
(T_1,T_5,T_8,T_9)	33

关键路径为(T_1,T_2,T_7,T_9),路径长度为 36。由关键路径上第一个节点开始将关键路径上节点依次加入待调度队列,进行调度。关键路径队列中节点变化过程,以及调度序列形成过程如表 2 所示。

调度结果采用图 5 所示的甘特图表示,因其可以较为直观地反映出节点在处理器内核上的分布情况,以及处理器内核之间的通信关系。图中阴影部分表示不在同一内核上的节点之间的通信开销。本例中任务执行总时间 $F=22$,占用处理器内核 3 个。

3.2　对比分析

采用文献[2]中算法(简称算法 1)对图 3 所示的任务图进行调度,执行总时间 $F=23$,占用处理器内核个数为 5,调度结果如图 6 所示。

表 2　CPTD 算法调度步骤说明

Tab. 2　Scheduling process statistics for CPTD

	队列元素	当前节点	节点入度	紧前序列	是否合并	调度序列	执行时间
1	T_1,T_2,T_7,T_9	T_1	0	–	否	(T_1)	2
2	T_2,T_7,T_9	T_2	1	(T_1)	否	(T_1,T_2)	5
3	T_7,T_9	T_7	2	$(T_1,T_2),(T_1,T_3)$	是	(T_1,T_2,T_3)	12
4	T_9	T_6	1	(T_1,T_3)	否	(T_1,T_3,T_6)	9
5	T_9	T_8	4	$(T_1,T_2),(T_1,T_3),$ $(T_1,T_4),(T_1,T_5)$	是	$\{(T_1,T_3),(T_1,T_4),$ $(T_1,T_5,T_2,T_8)\}$	14
6	T_9	T_9	3	$(T_1,T_2,T_3),(T_1,T_3,T_6),$ $\{(T_1,T_3),(T_1,T_4),(T_1,T_5,T_2,T_8)\}$	是	$\{(T_1,T_2,T_6),(T_1,T_4),$ $(T_1,T_2,T_3,T_7,T_5,T_8,T_9)\}$	22

对比图5 与图6 可以看出,CPTD 算法先查找关键路径,后采用紧前节点组尽早完成策略,将节点 T_7 和 T_8 的调度序列合并,使得 T_9 的开始执行时间得以提前,通信时间得到较好地控制,进而使

任务执行总时间得以缩短，并且占用处理器核数较少。而算法 1 则采用先将 T_9 节点分配到 T_8 的执行序列上后判断是否合并序列的调度方式，在该方式下，T_7 节点的调度序列将不能合并到 T_8 节点的调度序列上来，因合并后会使的 T_9 的开始时间进一步延迟。

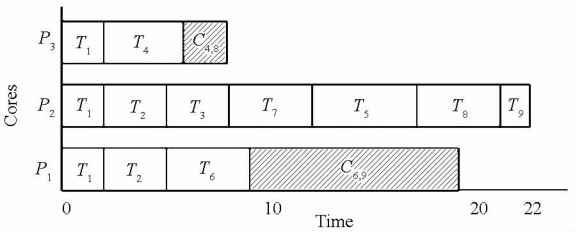

图 5 CPTD 算法对图 3 任务图的调度结果

Fig. 5 Scheduling result of DAG in fig. 3 by CPTD

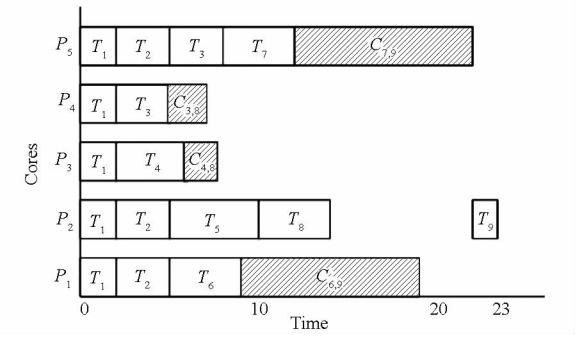

图 6 算法 1 对图 3 的调度结果

Fig. 6 Scheduling result of DAG in fig. 3 by algorithm 1

由文献[4,7,16]可知，TDMSCL、ETDS 和 PPA 算法是引言中所提到的 7 种传统调度算法中理论上最优且最具有代表性的算法，于是将选取相应参考文献中的 DAG 任务图图例，与这 3 种调度算法进行对比分析，以说明 CPTD 算法的优越性。虽然用来对比的任务图不多，但是这些任务图大都是实际应用例子或带有各自算法特征的图例，具有一定代表性。对比结果如表 3 所示。

表 3 调度性能比较

Tab. 3 Performance comparison among algorithms

图例	算法	完成时间	时间复杂度	内核占用
文献[2]图 7	算法 1[2]	16	n^2	3
	CPTD	15	n^2	3
文献[4]图 5－7	TDMSCL[4]	15	n^3	3
	CPTD	14	n^2	3
文献[7]图 5	ETDS[7]	22	n^3	3
	CPTD	22	n^2	3
文献[16]图 3	PPA[6]	20	n^2	5
	CPTD	20	n^2	3

通过以上对比分析可以看出，CPTD 算法在完成时间上比算法 1 及 TDMSCL 算法少，且在调度某些类型的任务图时比算法 1 及 PPA 算法节省 40% 的内核消耗；比 TDMSCL 算法及 ETDS 算法复杂度减少一个数量级。

4 结论

本文针对多核调度问题中存在的任务完成时间延迟的情况，提出了基于任务复制和综合调度中关键路径的 CPTD 算法，从调度原理上解决了此类问题。因此，本文研究可对多核调度算法的进一步研究有促进作用。

(1)在任务图处理中，引入综合调度产品加工树思想，使任务图中节点都能虚拟为一棵单独的加工树，便于对每个节点单独分析处理。

(2)采用任务复制和综合调度中关键路径法，可以在二次多项式内，较为有效地减少基于片上互连的同构多核处理器内核之间通信开销，提前关键路径上节点的开始时间，达到缩短程序执行总时间的目的，因此 CPTD 算法简便、可行。

参考文献(References)

[1] 陈芳园，张冬松，刘聪，等. 基于取指执行时序范畴的多核共享 Cache 干扰分析[J]. 计算机研究与发展，2013，50(1)：206－217.
CHEN Fangyuan，ZHANG Dongsong，LIU Cong，et al. Analysis of inter-thread interference on shared cache multi-core architectures based on instruction fetch timing frame[J]. Journal of Computer Research and Development，2013，50(1)：206－217.(in Chinese)

[2] Xie Z Q，Zhao L，Xin Y，et al. A scheduling optimization algorithm based on task duplication for multi-core processor[J]. Energy Procedia，2011，13：6145－6154.

[3] 陈莉丽，姚益平，蔡潞. 多核环境下负载均衡的并行离散事件全局调度机制[J]. 国防科技大学学报，2012，34(4)：108－113.
CHEN Lili，YAO Yiping，CAI Lu. A global schedule mechanism for PDES on multi-core environments[J]. Journal of National University of Defense Technology，2012，34(4)：108－113.(in Chinese)

[4] 吴佳骏. 多核多线程处理器上任务调度技术研究[D]. 北京：中国科学院研究生院，2006.
WU Jiajun. Task scheduling for multi-core and multi-thread processor[D]. Beijing：Graduate University of Chinese Academy of Sciences，2006.(in Chinese)

[5] 何琨，赵勇，陈阳. 分布式环境下多任务调度问题的分析与求解[J]. 系统工程理论与实践，2007，27(5)：119－125.
HE Kun，ZHAO Yong，CHEN Yang. Analysis and solutions for multitasks scheduling in distributed environments[J]. Systems Engineering-Theory and Practice，2007，27(5)：119－125.(in Chinese)

[6] 周双娥，袁由光，熊兵周，等. 基于任务复制的处理器预

分配算法[J]. 计算机学报, 2004, 27(2): 216 – 223.
ZHOU Shuange, YUAN Youguang, XIONG Bingzhong, et al. An algorithm of processor pre-allocation based on task duplication[J]. Chinese Journal of Computers, 2004, 27(2): 216 – 223. (in Chinese)

[7] Park C I, Choe T Y. An optimal scheduling algorithm based on task duplication[J]. IEEE Transactions on Computers, 2002, 51(4): 444 – 448.

[8] Darbha S, Agrawal D P. Optimal scheduling algorithm for distributed-memory machines [J]. IEEE Transactions on Parallel and Distributed Systems, 1998, 9(1): 87 – 95.

[9] Ahmad I, Kwok Y K. On exploiting task duplication in parallel program scheduling[J]. IEEE Transactions on Parallel and Distributed Systems, 1998, 9(9): 872 – 892.

[10] Papadimitriou C H, Yannakakis M. Towards an architecture independent analysis of parallel algorithms[J]. SIAM journal on computing, 1990, 19(2): 322 – 328.

[11] 谢志强, 辛宇, 杨静. 基于设备空闲事件驱动的综合调度算法[J]. 机械工程学报, 2011, 47(11): 139 – 147.
XIE Zhiqiang, XIN Yu, YANG Jing. Integrated scheduling algorithm based on event-driven by machines' idle [J]. Journal of Mechanical Engineering, 2011, 47(11): 139 – 147. (in Chinese)

[12] 谢志强, 杨静, 周勇, 等. 基于工序集的动态关键路径多产品制造调度算法[J]. 计算机学报, 2011, 34(2): 406 – 412.
XIE Zhiqiang, YANG Jing, ZHOU Yong, et al. Dynamic critical path muti-product manufacturing scheduling algorithm based on operation set[J]. Journal of Computers, 2011, 34(2): 406 – 412. (in Chinese)

[13] 邸楠, 王韬, 李晓明. Lily Task 任务并行环境中基于任务关系的初始任务分配算法[J]. 计算机学报, 2005, 28(5): 892 – 895.
DI Nan, WANG Tao, LI Xiaoming. A static task distribute algorithm based on task relation in Lily Task[J]. Journal of Computers, 2005, 28(5): 892 – 895. (in Chinese)

[14] 孟宪福, 闫玲玲, 刘伟伟. 基于动态任务优先级的网格任务调度算法研究[J]. 大连理工大学学报, 2012, 52(2): 277 – 284.
MENG Xianfu, YAN Lingling, LIU Weiwei. Research on task scheduling algorithm in grid computing systems based on dynamic task priority[J]. Journal of Dalian University of Technology, 2012, 52(2): 277 – 284. (in Chinese)

[15] 李静梅, 李静, 丁楠. 基于异构多核处理器的高效任务调度算法[J]. 高技术通讯, 2012, 22(3): 225 – 230.
LI Jingmei, LI Jing, DING Nan. An efficient task scheduling algorithm for heterogeneous multi-core processors [J]. Chinese High Technology Letters, 2012, 22(3): 225 – 230. (in Chinese)

[16] 徐成, 赵林祥, 杨志邦. 一种基于多处理器任务复制的分簇调度算法[J]. 计算机应用研究, 2012, 29(8): 2931 – 2934.
XU Cheng, ZHAO Linxiang, YANG Zhibang. Scheduling algorithm of multi-processor based on task clustering and duplication[J]. Application Research of Computers, 2012, 29(8): 2931 – 2934. (in Chinese)