doi:10.11959/j.issn.1000-436x.2015162

面向分布式环境的信号驱动任务调度算法

辛宇[1]，杨静[1]，谢志强[2]
（1. 哈尔滨工程大学 计算机科学与技术学院，黑龙江 哈尔滨 150001；2. 哈尔滨理工大学 计算机科学与技术学院，黑龙江 哈尔滨 150001）

摘 要：为优化 IaaS 服务的执行效率，提出面向 IaaS 的信号驱动任务调度算法，该算法根据 IaaS 模型的结构特征建立控制子系统和节点子系统，根据任务的结构特征建立任务的 DAG(directed acyclic graph)调度模型，并建立各任务分片的状态转化机制及控制子系统和节点子系统间的信号通信机制。以系统间信号交互的方式驱动任务分片的状态改变，并在每一调度时刻来临时利用并行优化选择策略分配任务分片。由于本算法采用了模拟 IaaS 模型的双系统控制方式，使本算法与 IaaS 模型的分布式体系相兼容且复杂度较低。最后通过实验验证了所提算法的有效性和实用性。
关键词：IaaS；云计算；任务调度；信号驱动；并行优化
中图分类号：TP311 **文献标识码**：A

Task scheduling algorithm for distributed environment based on signal-driven

XIN Yu[1], YANG Jing[1], XIE Zhi-qiang[2]
(1. College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China;
2. College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150001, China)

Abstract: In order to optimize the performance of user services in IaaS, the task scheduling algorithm for IaaS based on signal-driven is proposed, by which CS(control subsystem) and NS(inquiry nodes subsystem) based on the structural characteristics of the IaaS is established, and the DAG scheduling model based on the structural characteristics of the inquiry task is created. Then the conversion mechanism for the task partitions is created, constructing the signal communication mechanism for CS and NS, changing the status of the task partitions by signal-driven between the CS and NS, completing the task partitions allocation by POSS (parallel optimization selective strategy) in the scheduling time. This algorithm with low complexity is compatible with the distributed architecture of IaaS, because of utilizing dual system control mode. The effectiveness and practicality of this algorithm is verified by experiment.
Key words: IaaS; cloud computing; task scheduling; signal-driven; parallel optimization

1 引言

随着云计算技术的迅猛发展，云计算技术可将计算、存储、软件、服务等资源从分散的个人计算机或服务器移植到互联网环境中，以集中管理大规模高性能计算机、个人计算机、虚拟计算机，从而方便用户使用云资源。从层次上云计算平台可以分为以下 3 种服务模型：软件即服务(SaaS, software as a service)，平台即服务(PaaS, platform as a service)，基础架构即服务(IaaS, infrastructure as a service)。

目前 IaaS 模型是当前最重要的云平台模型，IaaS 的意义在于，用户可通过云接口利用 IaaS 云服务提供商所提供的 IT 设备资源运行所需应用，且无需考虑硬件的维护及更新[1]。IaaS 的技术核心是硬件虚拟化，通过利用虚拟化技术，把物理主机的各种硬件整合起来，屏蔽其硬件的物理细节，以资

收稿日期：2014-08-19；**修回日期**：2014-11-28
基金项目：国家自然科学基金资助项目（61370083, 61370086）；高等学校博士学科点基金资助项目（20122304110012）；黑龙江省自然科学基金资助项目（F201101）
Foundation Items: The National Natural Science Foundation of China (61370083, 61370086); The Research Fund for the Doctoral Program of Higher Education of China (20122304110012); The National Natural Science Foundation of Heilongjiang Province(F201101)

源池的概念统一代表物理硬件，保证资源的可定制能力和统一分配能力[2]。目前，有很多企业和科研机构推出了自己的 IaaS 云计算平台，面向用户提供计算资源和存储资源。最具代表性的是亚马逊(Amazon)的弹性计算云(EC2, elastic compute cloud)[3]，它通过 Web 服务的方式让用户弹性地运行自己的 Amazon 机器映像，用户可以在自己申请的虚拟机镜像上运行任何自己所需的软件或应用程序。同时，还有一些开源 IaaS 云计算平台：与 EC2 兼容的云平台 Eucalyptus (elastic utility computing architecture for linking your programs to useful systems)[4]，美国国家航空航天局和 Rackspace 合作研发的 OpenStack[5]以及中科院的 LingCloud[6]等。

IaaS 通过虚拟设备实现用户的服务，IaaS 在进行虚拟设备管理时，将一台物理设备划分为多台独立的虚拟设备，各个虚拟设备之间能进行有效的资源隔离和数据隔离；由于多个虚拟设备共同享一台物理设备的物理资源，能够充分复用物理设备的计算资源，提高资源利用率。IaaS 将云环境中的所有物理设备资源转化为对用户透明的统一资源池，并能按照用户需求分配不同性能的虚拟设备。IaaS 模型的框架体系如图 1 所示，其中用户的服务请求通过 Scheduler 模型进行虚拟设备的任务分配。

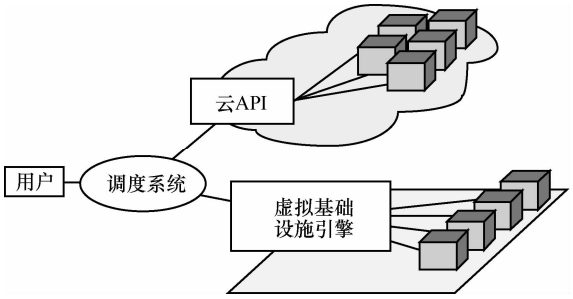


图 1　IaaS 的拓扑结构

为提高资源分配及数据处理的效率，近些年来面向 IaaS 的虚拟设备资源调度问题逐渐成为学术研究的重点，其研究内容主要包括任务分配调度和数据传输调度。在任务分配调度方面，主要研究内容是处理用户的任务分解及分配，对此 Marcos 等对 IaaS 模型的调度模块设计了 6 种面向弹性计算云(EC2)的兼容策略，并提出了 cloud schedule 和 sub-schedule 两级调度模式，分别从宏观和微观角度进行任务分配优化[7,8]；在数据传输调度方面，主要研究内容是根据数据的分布式存储特性建立以通信损耗化及数据时效优化的调度策略，对此 Nan 等[9]提出了 cost model 及 queuing model 模型，构建了多媒体数据的传播评价方法，并设计了以多媒体数据传输 QoS 优化为导向的云资源分布调度算法，Kllapi 等[10]建立了流式数据的 split、compute 和 merge 过程，实现了分布式数据流的传输调度。为统一 IaaS 分配和数据传输的调度模型，Lin 等[11]根据虚拟机的处理能力及网络的数据传输能力，将功能和位置相近的虚拟机划归为 cluster 单元，从而将 IaaS 模型的调度问题转化为 DAG(directed acyclic graph)调度优化问题，统一了 IaaS 任务调度模型。

目前可面向 IaaS 模型的 DAG 调度算法主要分为以下 5 类。

1) 权值选择算法。如 HEFT[12]、ACPM 算法[13]等，此类算法是早期 DAG 调度的主要算法，其实现原理是根据 DAG 的结构关系预先计算各个任务分片的 EFT(earliest finish time)，以确定任务分片的先后关系。

2) 层次优先选择算法。如 PCH(path clustering heuristic)[14]、优先工序集调度[15]等，此类算法以最大化同一层任务分片的并行处理为手段，将任务分片的层数作为优先级，以确定任务分片的调度次序。

3) 堆栈式回退算法。如 HEFT-lookahead[16]、回退抢占算法[17]等，此类算法考虑了后续调度中空闲时段被占用的情况，当出现预先计算的 EFT 与实际的结束时间偏差较大时，采用重定向的方式回退前续调度任务分片。

4) 路径聚合算法。如 HH(hybrid heuristic)[18]、动态关键路径法[19]等，此类算法考虑优先调度关键路径上的任务分片，增加关键路径上任务分片的优先级，从纵向角度优化调度结果。

5) 启发式优化算法。如遗传算法[20]，粒子群算法[21,22]等，此类算法通过建立代价函数对调度结果进行局部优化，建立并利用循环选择策略逐步保留局部优化结果，从而实现全局调度优化。

为实现高效处理用户 IaaS 任务请求，本文设计了一种基于信号驱动的 DAG 调度算法，由于采用了信号驱动的形式，可使算法结合权值选择算法、层次优先选择算法和路径聚合算法的特点，即在每一调度时刻根据任务分片的纵向权值进行横向并行优化选择，从而实现了 DAG 任务分片的纵横双向筛选，使调度过程更加合理化。另外，该算法由于回避了堆栈式回退算法的重定向操作，以及层次优先选择算法和

路径聚合算法的预调度操作，因此，算法的复杂度较低(约为 $O(n\log(n))$)且优于传统 DAG 调度算法，更适于应对云计算环境中的海量任务请求问题。

2 IaaS 模型的 DAG 调度问题分析

根据 IaaS 模型的分布式特性和虚拟设备的异构特性，IaaS 任务以分片的形式进行调度且各任务分片之间满足以下约束[7,20,21]。

1) IaaS 任务具有唯一的起始任务分片（DAG 入口节点）和唯一的终止任务分片（DAG 出口节点）。

2) 某一任务分片仅在满足其要求的虚拟设备中执行，且满足同一任务分片要求的虚拟设备不唯一，任务分片在各虚拟设备中的执行时间已知。

3) 任一虚拟设备在同一时刻只能执行同一任务的一个任务分片。

4) 任务分片 v_i 的紧后任务分片 v_j 需要等待 v_i 执行完毕后的通信数据，若 v_i 与 v_j 在同一虚拟设备中执行，则通信时间为 0。

5) 每一任务分片必须在其前任务分片均执行完毕，并收集到所有前序任务分片的通信数据时才可开始执行。

可根据 IaaS 模型的任务约束建立 DAG 调度模型，其符号描述如下。

1) DAG 模型以图 $G=(V, E)$表示，G 表示某一任务，$V=\{v_1, v_2, \cdots, v_n\}$表示任务 G 的任务分片集合，其中 $n=|V|$为任务分片的个数。

2) $D=\{d_1, d_2, \cdots, d_m\}$表示虚拟设备的集合，其中 $m=|D|$表示 IaaS 模型中虚拟设备的个数。

3) $c_{i,j}$ 表示虚拟设备 d_i 到虚拟设备 d_j 的通信代价。

图 2 为某任务的 DAG 结构模型，其中节点表示任务分片，节点 1 和节点 8 为起始任务分片和终止任务分片，有向边表示任务分片的有序关系，其权值代表任务分片间的通信代价，表 1 为任务分片在各虚拟设备中的执行时间。

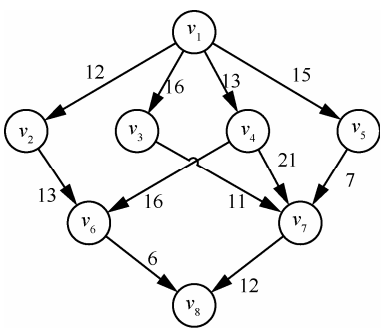


图 2 DAG 结构模型

表 1 任务分片执行时间

分片	d_1	d_2	d_3
v_1	17	19	21
v_2	22	27	23
v_3	15	5	9
v_4	17	9	20
v_5	30	27	18
v_6	49	49	46
v_7	17	16	15
v_8	13	10	9

3 IaaS 调度系统设计

3.1 系统结构设计

定义 1 任务完毕信号(Sig_F)：NS 在任务分片执行完毕时向 CS 发送的反馈信号。

定义 2 数据传递信号(Sig_T)：CS 在分配任务分片时向 NS 发送的通知信号，该信号包含通信数据的源虚拟设备 ID 和目标虚拟设备 ID。

定义 3 任务分配信号(Sig_A)：CS 在分配任务分片时向 NS 发送的通知信号，该信号包含被分配的任务分片 ID 及虚拟设备 ID。

定义 4 就绪信号(Sig_R)：NS 在某一任务分片集齐所有所需的通信数据时向 CS 发送的信号。

图 3 为 CS 和 NS 的信号交互过程示例，图中 CS 包含了 3 个任务分片(v_i, v_j, v_k)，NS 包含了 3 个虚拟设备(d_1, d_2, d_3)，v_i、v_j、v_k 分别在 d_1、d_3、d_2 中执行，实箭头和虚箭头分别表示信号交互和数据传递，序号标识了 CS 与 NS 的信号先后次序，分别表示为：①任务分片 v_j 在 d_3 中执行完毕时，NS 向 CS 反馈信号；②任务分片 v_i 在 d_1 中执行完毕时，NS 向 CS 反馈信号；③CS 通知 d_1 和 d_3 向 d_2 传递数据；④NS 内部的数据传递；⑤d_2 在集齐所需的数据时向 CS 反馈信号；⑥CS 通知 NS 在 d_2 中执行 v_k；⑦任务分片 v_k 在 d_2 中执行完毕时，NS 向 CS 反馈信号。

3.2 系统状态分析

每一任务分片的运行过程中均会经历 4 种状态，各状态的定义如下。

定义 5 接收数据状态：某一任务分片的任一紧前任务分片执行完毕时所处的状态。

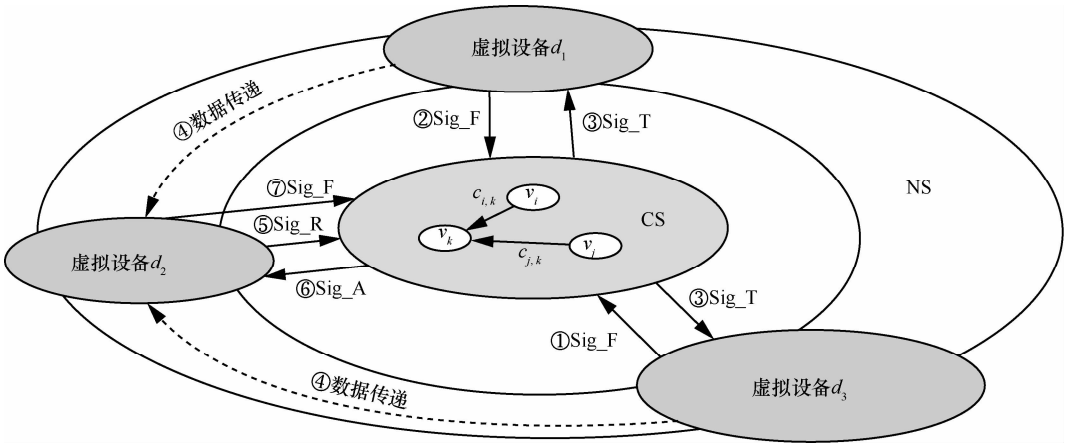


图 3　CS 和 NS 信号交互

定义 6　就绪状态：某一任务分片在接集齐所有所需的通信数据时所处的状态。

定义 7　执行状态：某一任务分片在开始执行时所处的状态。

定义 8　完毕状态：某一任务分片执行完毕时所处的状态。

虚拟设备会经历 2 种状态，其定义如下。

定义 9　忙碌状态：某一虚拟设备执行任务分片时所处的状态。

定义 10　空闲状态：某一虚拟设备无任务分片执行时所处的状态。

图 4 为任务分片和虚拟设备的状态变化示例，该示例直观表示了任务分片和虚拟设备在运行过程中的状态变化关系，图中右(左)侧虚框表示某一任务分片 v_i 在某一虚拟设备 d_j 中执行(完毕)时，v_i 和 d_j 的状态同时变化。其中，信号起到信息传递及驱动状态转化的作用。

3.3　调度系统优化分析

由于各任务分片的执行时间和各虚拟设备的数据传递时间均为已知，因此任务的寻优过程是静态过程，可向所有的虚拟设备发送 Sig_A 作寻优假设，具体的优化方案如下。

1) 当某一任务分片 v_i 的任一紧前任务分片执行完毕时(此时变为接收数据状态)，CS 向 NS 发送 m 个不同目标虚拟设备 ID 的信号 Sig_T，为 v_i 预分配所有 m 个虚拟设备。

2) CS 建立任务分片就绪表(RT)，记录任务分片在不同虚拟设备中变为就绪状态的时刻。

3) NS 建立虚拟设备空闲表(DT)，记录任一虚拟设备下一次变为空闲状态的时刻。

4) 由于 RT 和 DT 会随 CS 和 NS 的状态变化而变化，当 RT 或 DT 发生变化时利用 current 记录当前时刻，并利用下一节描述的并行优化选择策略进行任务分片的选择。

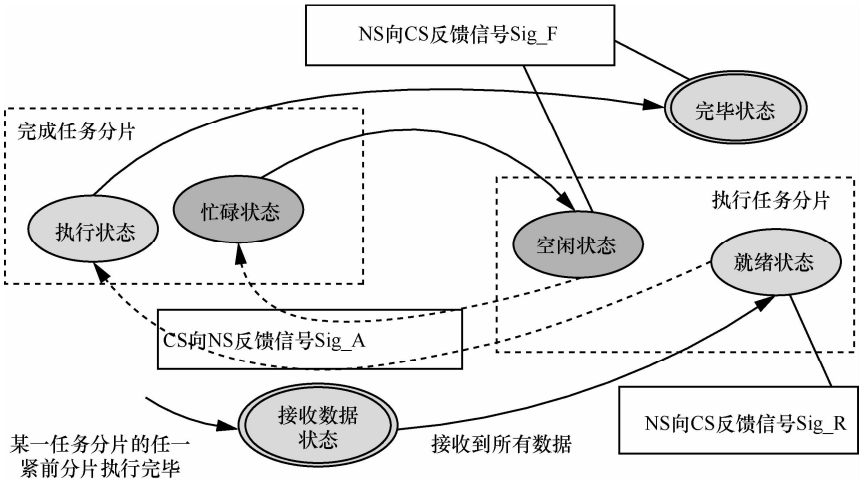


图 4　任务分片和虚拟设备的状态转化

4 并行优化选择策略

任务分片的执行需具备 2 个条件：1) 任务分片处于就绪状态；2) 虚拟设备处于空闲状态。为此本文所提出的并行优化选择策略以 RT 或 DT 的更新作为一次任务分片分配的时刻，为处于就绪状态的任务分片分配空闲虚拟设备，将该时刻定义为调度时刻。

由于当某一任务分片变为接收数据状态时，CS 会向 NS 中所有的虚拟设备发送 Sig_A，因此当调度时刻来临时，就绪的任务分片与空闲的虚拟设备间会存在多对多的关系，为实现任务分片与虚拟设备的一对一执行关系，本文在调度时刻利用如下策略为空闲的虚拟设备分配执行任务。

1) 利用 HEFT 算法[12]计算就绪任务分片的 *rank* 值，*rank* 值最大者优先选择空闲虚拟设备，HEFT 的计算公式为

$$rank_i = w_i + \max_{v_j \in succ(v_i)} (c_{i,j} + rank_j)$$

其中，$succ(v_i)$为任务分片 v_i 的紧后任务分片的集合，w_i 为 v_i 在各虚拟设备中执行时间的均值，当 v_i 为终止任务分片时 $rank_i=w_i$，表 2 为图 2 实例的 *rank* 值。

表 2　　图 2 示例的 *rank* 值

分片	*rank*
v_1	132.67
v_2	101.67
v_3	59.33
v_4	96.00
v_5	70.67
v_6	64.67
v_7	38.67
v_8	10.67

2) 计算就绪任务分片在每个虚拟设备的执行完毕时刻，选择执行完毕时刻最小者作为该任务分片的执行虚拟设备。

3) RT 和 DT 中的最小值为下一个调度时刻。

表 3 记录了图 2 示例的调度优化过程，表中各行记录了各调度时刻 RT 与 DT 的变化过程，2 列和 3 列为调度之前 RT 与 DT 的内容，4 列和 5 列为调度之后 RT 与 DT 的内容，第 6 列为预算出的下一调度时刻，如 2 列和 3 行可表示为：当 *current* 为 17 时，d_1、d_2、d_3 均空闲且 v_2、v_3、v_4、v_5 均只能在 d_1 中执行，此时利用优化选择策略选择在 d_1 中执行 v_2，RT 和 DT 中的最小值为 30，即为下一个调度时刻；当 *current* 为 30 时，仅 d_2, d_3 空闲且 v_3 和 v_5 只能在 d_1 中执行，因此无法调度 v_3 和 v_5，又由于此时 v_4 可在 d_2, d_3 中执行，此时利用优化选择策略选择在 d_2 中执行 v_4，并计算下一调度时刻为 33。在 RT 表格中 R 表示在当前时刻任务分片 v_i 可在 d_j 中执行，即 d_j 在当前时刻已集齐了 v_i 所需的数据，DT 记录了虚拟设备所执行的任务分片及执行结束时间。任务分片的调度结果如图 5 所示。

5 算法设计及复杂性分析

5.1 调度算法设计

通过以上分析，在以下 2 种情况发生的时刻，可作为调度时刻：①NS 在任务分片执行完毕时，通过改变 DT 内容的同时驱动 CS 改变 RT 的内容；②当某一任务分片变为就绪状态时，CS 改变 RT 的内容。本节通过分析 CS 和 NS 内部运行机制，从全局角度设计调度算法。

1) 信号设计。根据 CS 和 NS 的运行机制，设计了信号 Sig_A、Sig_T、Sig_R 和 Sig_F 进行数据通信与执行任务分片的交互，其中，Sig_T 和 Sig_R 用于数据通信的交互，Sig_A 和 Sig_F 用于执行任务分片的交互，4 种信号的说明如表 4 所示。

2) NS 内部运行算法。

①NS 等待 CS 传来的信号；

②若 CS 传来的信号为信号 Sig_T 则转③，若为信号 Sig_A 则转④；

③此时说明 CS 需求 Sig_T->D_ID 接受数据，NS 驱动虚拟设备 Sig_T->S_ID 向虚拟设备 D_ID 传递数据，并转⑤；

④此时说明 CS 需求 Sig_T->D_ID 执行任务分片，任务分片 Sig_A->partition 在 Sig_A->DB_ID 中执行，转⑥；

⑤当任务分片 v_i 在虚拟设备 d_j 中变为就绪状态时(即此时 d_j 已接收到所有所需的数据)，向 CS 反馈信号 Sig_R(Sig_R-> partition=v_i, Sig_R->DB_ID=d_j)，此时利用 Sig_R 向 CS 报告虚拟设备 d_j 可执行调度任务，转①；

⑥当任务分片 v_i 在虚拟设备 d_j 中执行完毕时，向 CS 反馈信号 Sig_F(Sig_F->partition=v_i，Sig_F->DB_ID=d_j)并更新 DT，若 v_i 为终止任务分片则转⑦否则转①；

⑦结束。

表 3　　**图 2 示例的 *rank* 值**

current	RT(before)	DT(before)	RT(after)	DT(after)	next *current*
0	分片 d_1 d_2 d_3 $\boldsymbol{v}_1$ R R R	d_1:Idle d_2:Idle d_3:Idle		d_1:v_1:17 d_2:Idle d_3:Idle	17
17	分片 d_1 d_2 d_3 v_2 R 29 29 v_3 R 33 33 v_4 R 30 30 v_5 R 32 32	d_1:Idle d_2:Idle d_3:Idle	分片 d_1 d_2 d_3 v_3 R 33 33 v_4 R 30 30 v_5 R 32 32	d_1: v_2:39 d_2:Idle d_3:Idle	30
30	分片 d_1 d_2 d_3 v_3 R 33 33 v_4 R R R v_5 R 32 32	d_1:v_2:39 d_2:Idle d_3:Idle	分片 d_1 d_2 d_3 v_3 R 33 33 v_5 R 32 32	d_1:v_2:39 d_2: v_4: 39 d_3:Idle	32
32	分片 d_1 d_2 d_3 v_3 R 33 33 v_5 R R R	d_1:v_2:39 d_2: v_4:39 d_3:Idle	分片 d_1 d_2 d_3 v_3 R 33 33	d_1:v_2:39 d_2: v_4: 39 d_3: v_5:50	33
33	分片 d_1 d_2 d_3 v_3 R R R	d_1:v_2:39 d_2: v_4:39 d_3: v_5:50	分片 d_1 d_2 d_3 v_3 R R R	d_1:v_2:39 d_2: v_4: 39 d_3: v_5:50	39
39	分片 d_1 d_2 d_3 v_3 R R R v_6 55 52 55	d_1:Idle d_2:Idle d_3: v_5:50	分片 d_1 d_2 d_3 v_6 55 52 55	d_1:Idle d_2: v_3: 44 d_3: v_5:50	44
44	分片 d_1 d_2 d_3 v_6 55 52 55	d_1:Idle d_2:Idle d_3: v_5:50	分片 d_1 d_2 d_3 v_6 55 52 55	d_1:Idle d_2: Idle d_3: v_5:50	50
50	分片 d_1 d_2 d_3 v_6 55 52 55 v_7 60 57 60	d_1:Idle d_2:Idle d_3: Idle	分片 d_1 d_2 d_3 v_6 55 52 55 v_7 60 57 60	d_1:Idle d_2: Idle d_3: Idle	52
52	分片 d_1 d_2 d_3 v_6 55 R 55 v_7 60 57 60	d_1:Idle d_2:Idle d_3:Idle	分片 d_1 d_2 d_3 v_7 60 57 60	d_1:Idle d_2: v_6:101 d_3: Idle	57
57	分片 d_1 d_2 d_3 v_7 60 R 60	d_1:Idle d_2:v_6:101 d_3:Idle	分片 d_1 d_2 d_3 v_7 60 R 60	d_1:Idle d_2: v_6:101 d_3:Idle	60
60	分片 d_1 d_2 d_3 v_7 R R R	d_1:Idle d_2:v_6:101 d_3:Idle		d_1:Idle d_2: v_6:101 d_3: v_7:75	75

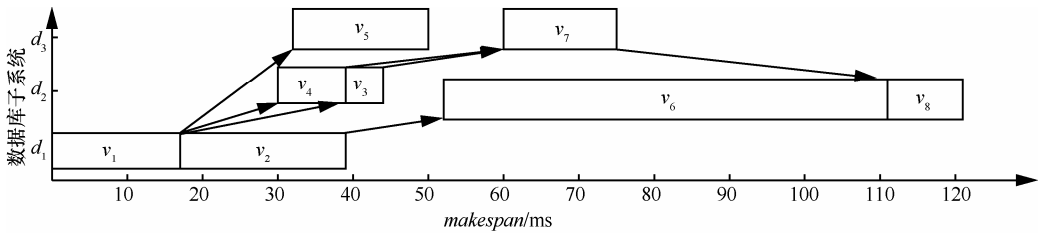


图 5　图 2 示例的调度结果

表 4 信号说明

Sig_A	Sig_T	Sig_R	Sig_F
作用：CS 向 NS 分配任务分片执行任务 分片：分配执行的任务分片 DB_ID：执行任务分片的虚拟设备 ID	作用：CS 向 NS 分配数据传输任务 S_ID：传递数据的虚拟设备 ID D_ID：接受数据的虚拟设备 ID 分片：需要传递数据的任务分片	作用：NS 向 CS 反馈已收到所有所需数据 分片：就绪的任务分片 DB_ID：就绪任务分片所在的虚拟设备 ID	作用：NS 向 CS 反馈已执行完毕的任务分片 分片：执行完毕的任务分片 DB_ID：执行完毕的虚拟设备 ID

3) CS 内部运行算法。CS 内部在 RT 和 DT 的内容变化时(收到 Sig_R 或 Sig_F 时)，利用并行优化选择策略对就绪任务分片进行分配，其运行机制如下：

①等待 NS 传来的信号；

②若为 Sig_F 则转⑥，若为信号 Sig_R 转③；

③根据 Sig_R->partition 和 Sig_R->DB_ID 更新 RT；

④利用并行优化选择策略对处于就绪状态的任务分片进行调度；

⑤任务分片 v_i 在虚拟设备 d_j 中执行，向 NS 反馈信号 Sig_A(Sig_A->partition=v_i，Sig_A-> DB_ID=d_j)，驱动 NS 执行任务 v_i，若 v_i 为终止任务分片则转⑦否则转①；

⑥向 NS 反馈信号 Sig_T(Sig_T->S_ID =Sig_F-> DB_ID，Sig_A-> DB_ID=所有虚拟设备 ID)，对执行完毕的任务进行后续执行的数据传递，转①；

⑦结束。

图 6 为 CS 与 NS 算法流程。其中，实线表示单一任务分片的运行过程，虚线表示多个任务分片的实时等待过程。

5.2 复杂度分析

本算法的复杂之处在于某一调度时刻到来时，利用并行优化选择策略，实现任务分片与虚拟设备一对一的选择，设任务分片总数为 n，虚拟设备总数为 m，总体复杂度分析如下。

1) NS 算法复杂度分析

NS 只接收信号 Sig_T 和 Sig_A，在调度过程中 CS 为每一任务分片发送 m 个信号 Sig_T 和 1 个信号 Sig_A，NS 共需处理 $n(m+1)$个信号，且 Sig_T 和 Sig_A 的处理过程仅为常数复杂度，因此 NS 的复杂度为 $O(mn)$。

2) CS 算法复杂度分析

① 根据文献[12]计算每个任务分片 *rank* 值的

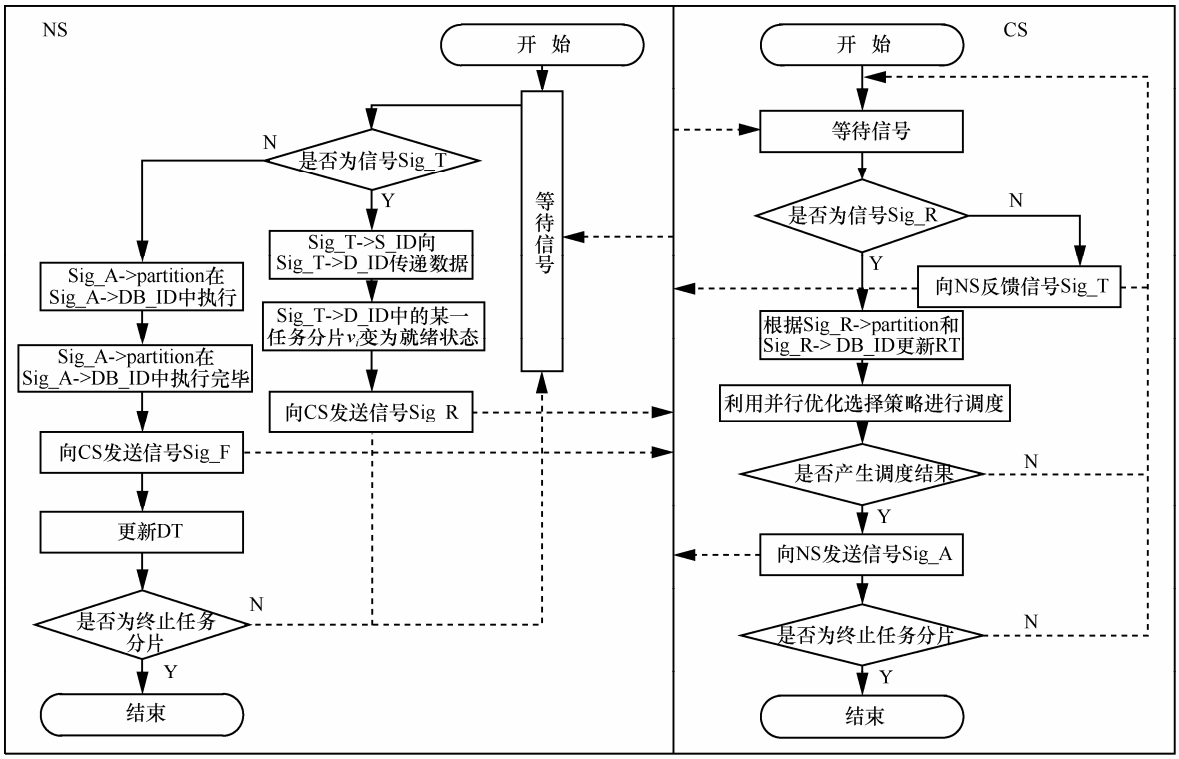


图 6 CS 与 NS 算法流程

时间复杂度为 $O(mn)$；

② n 个任务分片按 *rank* 值进行排序的时间复杂度为 $O(n\log(n))$；

③ 每一调度时刻，处于空闲状态的虚拟设备个数最大为 m，因此，就绪任务分片对虚拟设备的选择次数据最大为 m，总选择次数为 mn，其时间复杂度为 $O(mn)$。

综上所述，本文算法的时间复杂度为 $O(mn+n\log(n))$，由于在现实环境下虚拟设备总数为较小的常数，因此本文算法的复杂度可近似为 $O(n\log(n))$，文献[12~19]的复杂度分别为 $O(n^2)$、$O(n^2\log(n))$、$O(n^2\log(n))$、$O(n^2\log(n))$、$O(n^3)$、$O(n^2\log(n))$、$O(n^2\log(n))$、$O(n^3)$均高于本文算法。

6　实验分析

为验证本文算法的有效性，本实验以文献[23]的 DAG 数据 NSL(normalized schedule length)为标准随机生成任务，以模拟 IaaS 云计算平台的 DAG 任务分片结构，并以 HEFT[12]、PCH[14]、HEFT-lookahead[16]和 HH 算法[18]作为对比算法（本文算法用 SD 表示）。本实验平台软件：Win7 32 位，Matlab2012；硬件：intel i3 处理器，4 GB 内存，所设计的 4 组实验如下。

1) 资源占用时间分析实验。本组实验通过计算 100 组随机任务的调度结果，分析任务对虚拟设备的占用时间

$$makespan = \sum_{i=1}^{i=|V|} makespan_{v_i}$$

其中，$makespan_{v_i}$ 为任务分片 v_i 的执行时间，*makespan* 越小说明运行结果对虚拟设备的占用率越低，算法有效性越高。由于 100 组随机任务的结构差异较大，为直观显示实现各算法的对比结果，分别按各算法的 *makespan* 对 100 组结果进行排序，其结果如图 7 所示，该图分别以各算法为视角，对比各算法的资源占用时间，其中曲线越靠近底部说明算法有效性越高，从图 7 可直观看出，在资源占用时间方面 lookahead 和本文算法优于其他算法且 HEFT 算法的效果最差。为消除各组随机任务的结构差异，本实验采用对每组结果进行[0,1]区间映射的方法整理各组数据，并以各算法结果之和作为各

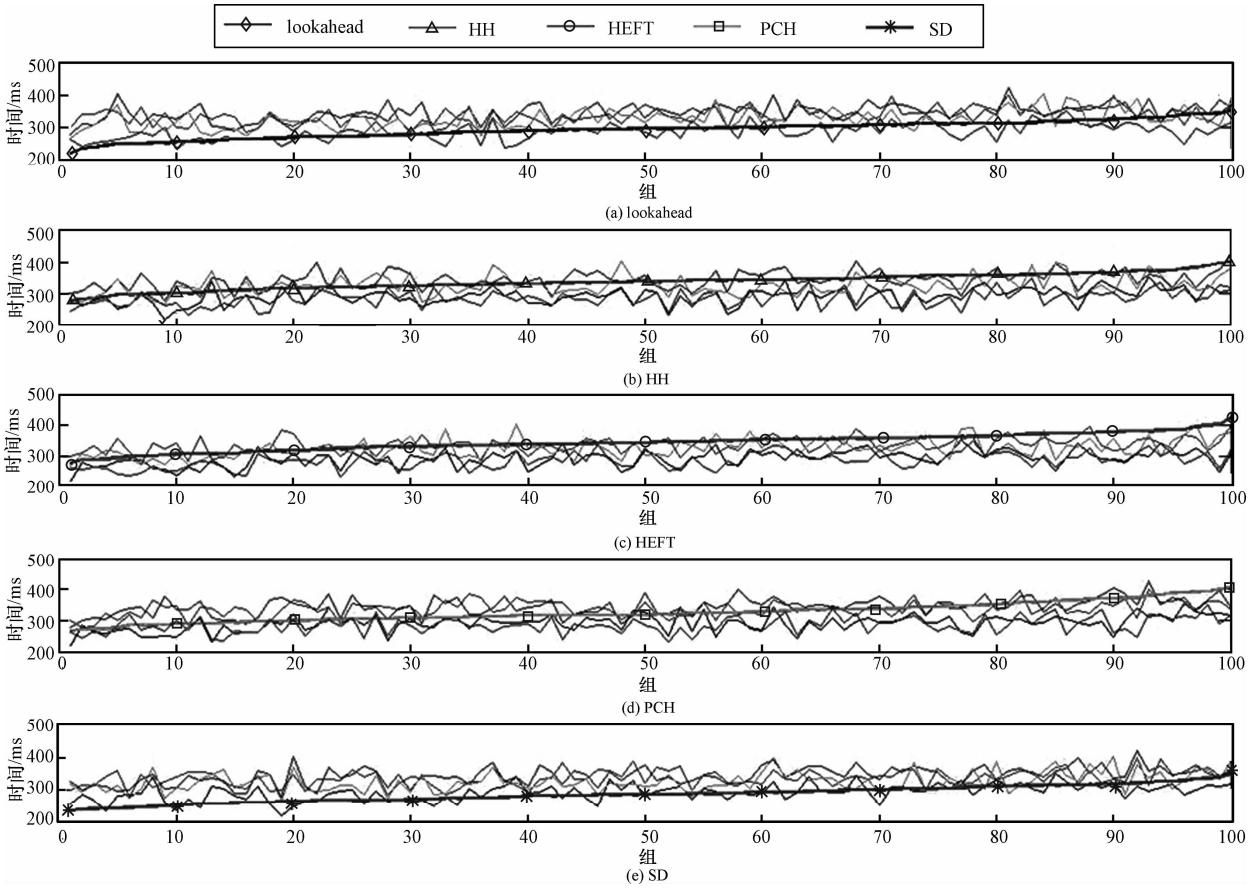


图 7　资源占用时间对比

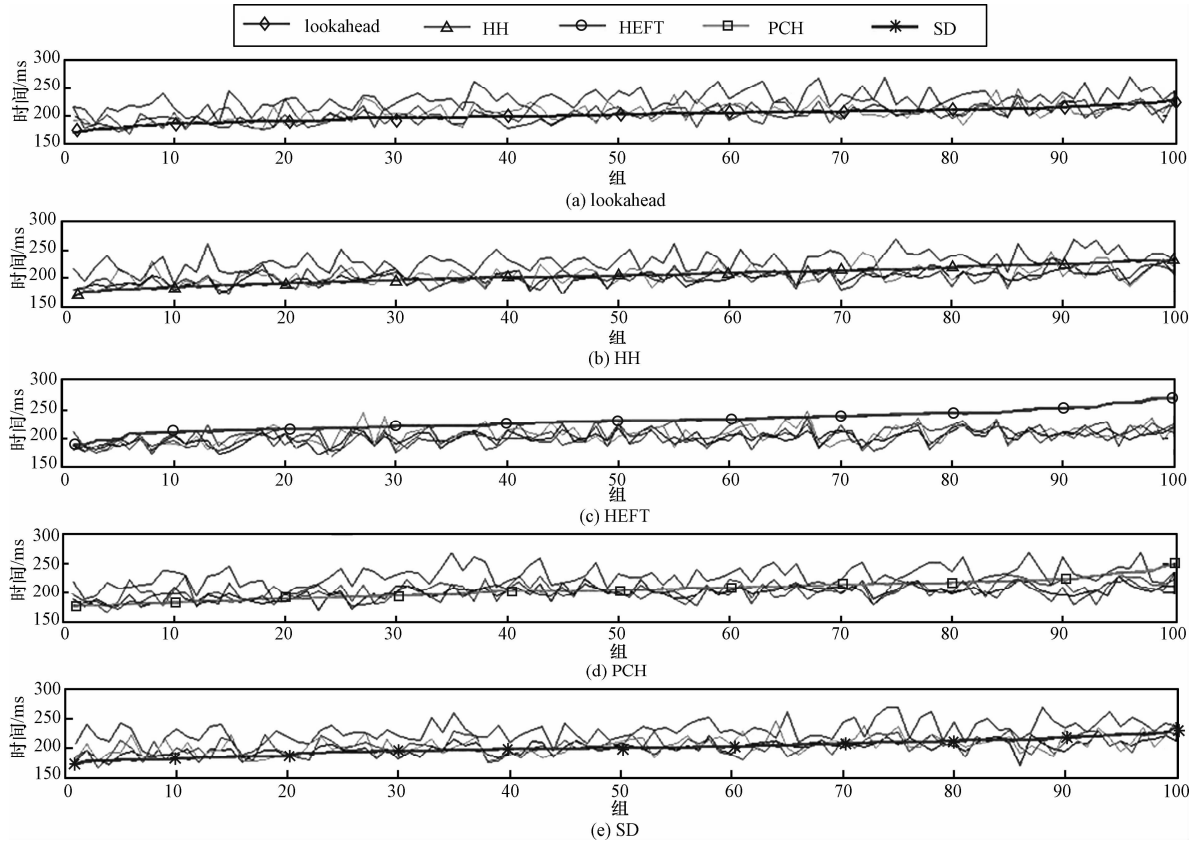


图 8　总执行时间对比

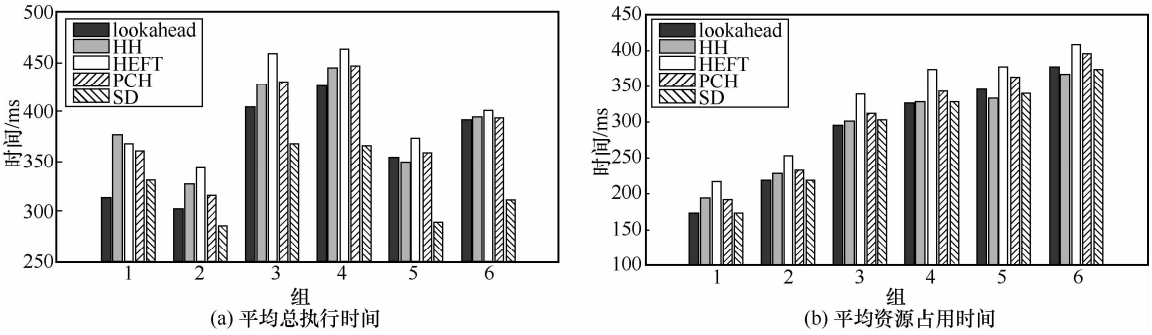
算法的数值评价，本实验中各算法的计算结果分别为：84.54，76.00，18.09，60.32，10.99，数值越小表示算法有效性越高，因此在本实验中各算法的有效性排序为：SD（本文算法）、lookahead、PCH、HH、HEFT。

2) 任务总执行时间分析。本实验以实验 1 的结果为数据集分析各算法的执行时间，其执行时间越短，说明调度算法的有效性越高，本实验采用与实验 1 相同的方式，分别按各算法的执行时间对 100 组结果进行排序，其结果如图 8 所示，其中曲线越靠近底部说明算法有效性越高，从图 8 中可直观看出，HEFT 算法的效果最差，其他算法性能相近。本实验采用实验 1 的[0,1]区间映射求和方法进行计算，结果分别为：96.57，31.59，21.98，34.63，23.39，其中数值越小表示算法有效性越高，因此各算法的有效性排序为：lookahead、SD、HH、PCH、HEFT。

3) CCR(communication computation ratios)分析实验。CCR 分析的目的在于通过改变随机任务的通信时间以验证调度算法的稳定性，本实验以[5,25]为随机区间生成各任务分片的执行时间，以[6,26] *CCR*(*CCR*=0.5~3)为随机区间生成通信时间，创建 6 组 DAG 任务样本(每组 100 个样本)作为实验数据。图 9 为各算法在执行 6 组数据后的平均总执行时间和平均资源占用时间对比，图 9 的结果说明了本文算法在不同 CCR 条件下，相较于其他算法在资源占用方面的优越性较高。

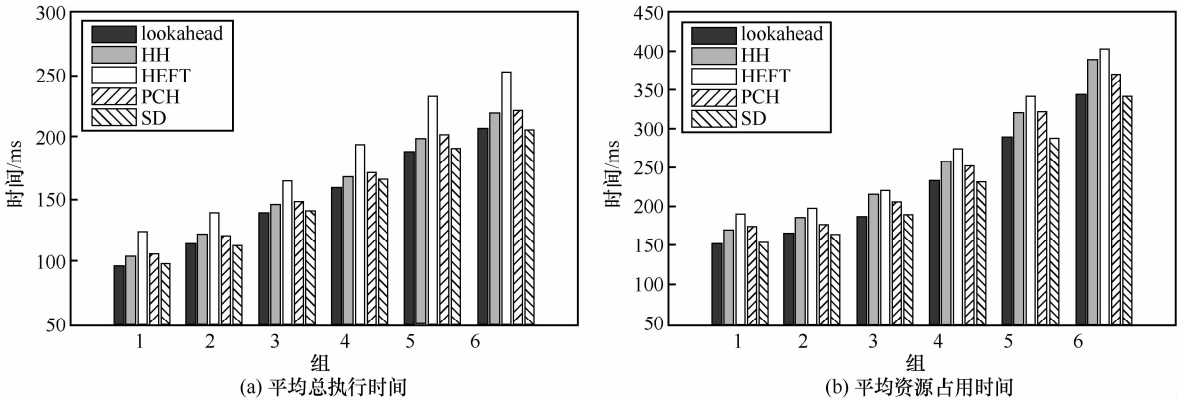
4) DAG 任务顽健性实验。调度算法的顽健性表现在处理异构 DAG 任务时的稳定性，本实验随机生成 6 组(每组 100 个样本)层数为 5~10 的异构 DAG 模型，统计各算法执行每组数据后的平均总执行时间和平均资源占用时间，其对比结果如图 10 所示，从图中本文算法与其他算法的性能对比变化可知，异构 DAG 任务对 SD 算法影响较小。

5) 虚拟设备个数稳定性实验。本实验生成了 6 组(每组 100 个样本)虚拟设备个数不同(3~8)的任务，统计各算法执行每组数据后的平均执行时间和平均资源占用时间，其对比结果如图 11 所示，从图中本文算法与其他算法的性能对比变化可知，虚拟设备的个数变化对 SD 算法的影响较小。



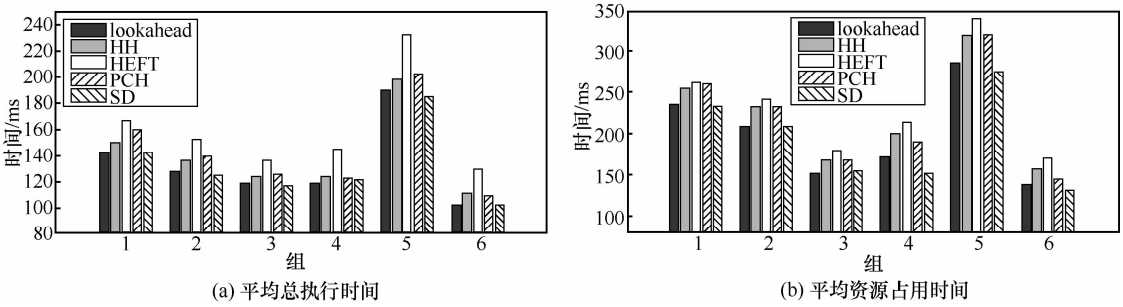
(a) 平均总执行时间　　(b) 平均资源占用时间

图 9　CCR 分片对比



(a) 平均总执行时间　　(b) 平均资源占用时间

图 10　DAG 模型顽健性对比



(a) 平均总执行时间　　(b) 平均资源占用时间

图 11　虚拟设备个数稳定性对比

6) 实验结论。本实验分别从 5 个方面实证了算法的有效性，在 CCR 分析实验、DAG 任务顽健性实验和虚拟设备个数稳定性实验方面，通过与其他算法结果的对比，验证了本文算法的稳定性，说明本文算法面对各类 DAG 任务调度问题时可保持其有效性；在任务总执行时间分析实验方面，本文算法执行结果近于同类算法最优值，在资源占用时间方面优于其他算法。

7　结束语

本文根据 IaaS 模型中各虚拟设备离散分布的特点，建立控制子系统 CS 和节点子系统 NS 及 IaaS 模型的 DAG 任务调度模型，并利用信号驱动的方式处理 DAG 任务调度问题。由于该算法以并行计算的 DAG 任务调度模型为基础使该算法具有普适性，可解决一般并行计算优化问题，本文算法的创新性和意义如下：所设计的利用信号交互机制进行驱动式调度，是面向 IaaS 的 DAG 任务调度的一次创新；所设计的并行优化选择策略，从全局角度考虑了 DAG 结构，使调度结果更加合理化；算法的复杂度为 $O(n\log(n))$，低于传统 DAG 调度算法，使得该算法更加简单适用；算法的稳定性高于其他算法，适用于各类异构任务；相较于其他算法，本文算法的执行总时间近于最优，且对虚拟设备的占用

时间最低。

因此，本文提出的算法不仅可以解决 IaaS 模型的调度问题，且对深入研究并行调度优化问题具有一定的理论和实际意义。

参考文献：

[1] ZHANG J X, GU Z M, ZHENG C. Survey of research progress on cloud computing[J]. Application Research of Computers, 2010, 27(2): 429-433.

[2] 谢亚龙，丁丽萍，林渝淇等. ICFF：一种 IaaS 模式下的云取证框架[J]. 通信学报, 2013, 34(5): 200-206.
XIE Y L, DING L P, LIN Y Q. ICFF: a cloud forensics framework under the IaaS model[J]. Journal on Communications, 2013, 34(5): 200-206.

[3] AMAZON Inc. Amazon elastic compute cloud (Amazon EC2)[EB/OL]. http://aws.amazon.com/ec2.

[4] NURMI D, WOLSKI R, GRZEGORCZYK C, *et al*. The eucalyptus open-source cloud-computing system[A]. Proceedings of the 9th IEEE/ACM Conference on International Computing and the Grid[C]. Tsukuba, Japan, 2009.124-131.

[5] COMPUTING R C. Openstack open source cloud computing software[EB/OL]. http://openstaek.org.

[6] LU X Y, LIN J, ZHA L, *et al*. Vega LingCloud: a resource single leasing point system to support heterogeneous application modes on shared infrastructure[A]. Proceedings of the 9th IEEE Conference on Parallel and Distributed Processing with Applications[C]. Busan, Korea, 2011. 99-106.

[7] MARCOS D, COSTANZO A, BUYYA R. A cost-benefit analysis of using cloud computing to extend the capacity of clusters[J]. Cluster Computing, 2010, 13(3): 335-347.

[8] MARCOS D , COSTANZO A, BUYYA R. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters[A]. Proceedings of the 18th ACM international Conference on High Performance Distributed Computing[C]. Munich, Germany, 2009.141-150.

[9] NAN X, HE Y, GUAN L. Optimal resource allocation for multimedia cloud based on queuing model[A]. Proceedings of the 13th IEEE Conference on Multimedia Signal[C]. Hangzhou, China, 2011.1-6.

[10] KLLAPI H, SITARIDI E, TSANGARIS M M, *et al*. Schedule optimization for data processing flows on the cloud[A]. Proceedings of the 2011 ACM SIGMOD Conference on Management of Data[C]. Athens, Greece, 2011. 289-300.

[11] LIN C, LU S. Scheduling scientific workflows elastically for cloud computing[A]. Proceedings of the 2011 IEEE Conference on Cloud Computing[C]. Washington DC, USA, 2011. 746-747.

[12] TOPCUOGLU H, HARIRI S. Performance-effective and low- complexity task scheduling for heterogeneous computing[J]. Parallel and Distributed Systems, 2002(13):260-274.

[13] 谢志强, 刘胜辉, 乔佩利. 基于 ACPM 和 BFSM 的动态 Job-Shop 调度算法[J]. 计算研究与发展, 2003, 40(7): 977-983.
XIE Z Q, LIU S H, QIAO P L. Dynamic job-shop scheduling algorithm based on ACPM and BFSM[J]. Journal of Computer Research and Development, 2003, 40(7): 977-983.

[14] LUIZ F B, EDMUNDO R M. Towards the scheduling of multiple workflows on computational grids[J]. Journal of Grid Computing, 2010, 8(3): 419-441.

[15] 谢志强，杨静，杨光. 可动态生成具有优先级工序集的动态 Job-Shop 调度算法[J]. 计算机学报, 2008, 31(3): 502-508.
XIE Z Q, YANG J, YANG G. Dynamic Job-Shop scheduling algorithm with dynamic set of operation having priority[J]. Chinese Journal of Computers, 2008, 31(3): 502-508.

[16] LUIZ F B, RIZOS S, EDMUNDO R M. DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm[A]. Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing[C]. Pisa, Italy, 2010.27-34.

[17] 谢志强, 辛宇, 杨静. 可回退抢占的设备驱动综合调度算法[J]. 自动化学报, 2011, 37(11): 1332-1343.
XIE Z Q, XIN Y, YANG J. Machine-driven integrated scheduling algorithm with rollback-preemptive[J]. Acta Automatica Sinica, 2011, 37(11): 1332-1343.

[18] RIZOS S, HENAN Z. A hybrid heuristic for DAG scheduling on heterogeneous systems[A]. Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS)[C]. Santa Fe, USA, 2004.266-286.

[19] 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 Z Q, YANG J, ZHOU Y, *et al*. Dynamic critical paths multi-product manufacturing scheduling algorithm based on operation set[J]. Chinese Journal of Computers, 2011, 34(2): 406-412.

[20] TAYAL S. Tasks scheduling optimization for the cloud computing systems[J]. International Journal of Advanced Engineering Sciences And Technologies, 2011, 5(2): 111-115.

[21] PANDEY S, WU L, GURU S M, *et al*. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments[A]. Proceedings of the 24th IEEE Conference on Advanced Information Networking and Applications[C]. Perth, Australia, 2010.400-407.

[22] WU Z, NI Z, GU L, *et al*. A revised discrete particle swarm optimization for cloud workflow scheduling[A]. Proceedings of the 2010 IEEE Conference on Computational Intelligence and Security[C]. Guangzhou, China, 2010.184-188.

[23] LUIZ F B, EDMUNDO R M. Towards the scheduling of multiple workflows on computational grids[J]. Journal of Grid Computing. 2010, 8(3) : 419-441.

作者简介：

辛宇（1987-），男，黑龙江哈尔滨人，哈尔滨工程大学博士生，主要研究方向为企业智能计算、数据库与知识工程。

杨静（1962-），女，黑龙江哈尔滨人，哈尔滨工程大学教授、博士生导师，主要研究方向为企业智能计算、数据库与知识工程。

谢志强（1962-），男，黑龙江哈尔滨人，博士，哈尔滨理工大学教授，主要研究方向为企业智能计算、数据库与知识工程。