基于负载均衡的随机作业流密码服务调度算法

李莉[1,2]，史国振[3]，耿魁[4]，董秀则[2]，李凤华[3,4,5]

（1. 西安电子科技大学通信工程学院，陕西 西安 710071；2. 北京电子科技学院电子信息工程系，北京 100070；
3. 北京电子科技学院信息安全系，北京 100070；4. 中国科学院信息工程研究所信息安全国家重点实验室，北京 100093；
5. 中国科学院大学网络空间安全学院，北京 100049）

摘 要：针对安全领域业务流并行处理系统面临数据密码服务请求多样，串行工作模式和并行工作模式交叉，不同业务的数据流相互交叉的现状以及服务响应的高速、高可靠性需求的问题，为了提高多密码算法并行处理的效率，以负载均衡为调度目标，在基于业务标识的分层硬件调度方法 HHS-ACDID 基础上，综合考虑算法处理节点的存储容量和处理速度，设计一种同时支持非关联任务和关联任务的负载均衡作业调度算法，实现了高速的密码处理吞吐率。仿真结果表明，该算法能够完成对数据流系统的动态调度并且得到较优的负载均衡效果，与 HHS-ACDID 相比，执行效率提高 12%左右。

关键词：作业调度；并行处理；负载均衡；多密码算法；数据流

中图分类号：TP393.2

文献标识码：A

doi: 10.11959/j.issn.1000-436x.2018105

Scheduling algorithm for stochastic job stream cipher service based on load balancing

LI Li[1,2], SHI Guozhen[3], GENG Kui[4], DONG Xiuze[2], LI Fenghua[3,4,5]

1. College of Communication Engineering, Xidian University, Xi'an 710071, China
2.Department of Electronic and Information Engineering, Beijing Electronics Science and Technology Institute, Beijing 100070, China
3. Department of Information Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China
4.State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Science, Beijing 100093, China
5. School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: Business stream parallel processing system face the situation of the diversity of cipher service requests, the cross of serial mode and parallel mode, the intercross of different business data flow, and the demand of high speed and high reliability in security field. In order to improve the parallel processing efficiency of multi-cryptographic algorithm, load balancing was used as scheduling objective. Based on hierarchical hardware scheduling method (HHS-ACDID), considering the processing node's storage capacity and processing speed, a load balancing scheduling algorithm was approved to support non related tasks and related tasks at the same time, which achieves the high speed cipher processing throughput. Simulation results show that the algorithm can complete dynamic scheduling of data stream system and get better load balancing effect. Compared with HHS-ACDID, the efficiency of the algorithm is improved by about 12%.

Key words: job stream scheduling, parallel processing, load balancing, multi-cryptography, data stream

收稿日期：2017-10-30；**修回日期**：2018-03-22

通信作者：李凤华，lfh@iie.ac.cn

基金项目：国家重点研发计划基金资助项目（No.2016YFB0800304）；北京市自然科学基金资助项目（No.4152048）

Foundation Items: The National Key Research and Development Program of China (No.2016YFB0800304), The Natural Science Foundation of Beijing (No.4152048)

1 引言

天地一体化网络作为无处不在的网络服务的基础支撑，是国家信息化的关键。在安全形势日益严峻的今天，天地一体化网络的安全对于实现国家安全战略目标具有重要的意义。天地一体化网络的安全保障离不开相应的密码防护，通过配备高性能的密码服务器，采用密码计算资源的虚拟化、密码服务的高并发调度运行来保障数据安全。天地一体化信息网络由于跨陆、海、空、天多域及天基网络的特殊性，拓扑结构具有动态变化的特点，且传输链路具有高延时、大方差、间歇性的特点，异构互联也致使网络设备类型多、密码资源多、业务需求量大[1]。因此服务于天地一体化网络的密码服务器存在接收业务海量高并发且不同业务随机交叉的现象，如何保证来自不同网络应用端的不同密码服务请求得到有效、快速的解决，防止业务流阻塞，业务流的调度及并发运算起着至关重要的作用。文献[2]针对安全领域中海量业务安全需求多样性导致的多种密码算法运算随机交叉的现象，提出了具有关联判断的基于业务标识的分层硬件调度方法——HHS-ACDID，通过两级调度完成业务流与密码算法IP核间的映射以及随机交叉业务流中关联作业包的正确有序处理，从而实现了高速数据流下多对多通信中多密码算法、多数据流的随机交叉加解密问题。

由于多算法密码服务器属于典型的异构多核系统，各密码算法处理节点的处理能力和存储容量不同，业务流到达各处理节点的时间不一致，如果不能进行合理的作业包与处理节点间的映射，则可能导致处理节点负载轻重不一，影响整个处理系统的吞吐率。本文针对多算法密码服务器的特点，考虑实际处理需求，在文献[2]的基础上，抽象出与实际密切吻合的任务模型，在充分考虑处理节点运算速率和节点存储容量的基础上，以负载均衡为调度目标，将处理节点的实际处理能力作为静态参数，处理节点任务队列的剩余容量作为动态参数，采用静态参数和动态参数相结合的方法，设计了一种支持非关联任务和关联任务的两级负载均衡动态调度算法，在不额外增加系统硬件成本的条件下，实现了高速的密码处理吞吐率。

2 相关研究

多处理器任务调度问题是一种NP-hard问题，其最优解往往与具体应用有关，需要针对实际问题进行建模，很多文献从不同角度（如负载均衡、吞吐量最大化、资源利用率最大化、减少功耗等方面）对其进行了研究。为获得资源的最大利用率和并行任务的最短执行时间，Liu等[3]根据历史调度信息，采用Apriori分类数据挖掘算法，通过建立任务和处理器之间的关联规则来进行多核处理器调度。该算法提高了内核利用率，但是没有考虑决定系统调度成败的历史调度信息的产生；另外，由于该算法必须考虑关联规则外任务的调度，使该算法实现复杂。文献[4]提出一种基于异构多核处理器的PIE（performance impact estimated）调度算法，该算法以进程在各个内核上能够获得性能的评估结果作为任务调度的参考依据，以求充分利用处理器资源，提高任务执行率。其中，性能采集模块以及评估参数寄存器的使用在一定程度上降低了该算法的易用性。

针对多处理器平台中公平调度问题，Li等[5]基于时间片轮转（RR，round robin）调度算法，分别提出了面向同构的多核处理器（DWRR，distributed weighted round robin）调度算法和面向异构的多核处理器（ADWRR，asymmetric distributed weighted round robin）调度算法[6]，实现核间的负载均衡，并保证了全局调度公平性。但并未考虑关联任务的调度问题。针对多核处理器平台上关联任务的调度问题，杨茂林等[7]提出一种资源敏感的实时任务调度算法。该算法采用任务分组策略将存在共享资源敏感的任务划分为若干个相关任务子集，将同一相关任务子集上的任务尽量分配到同一内核上；对于无法分配在内核上的相关任务子集采用任务拆分策略进行拆分。该算法可以减少核间任务阻塞。但是，当系统中任务数较少或较多时，可能会因为任务相关度评价耗时过长，造成任务的等待，降低系统吞吐率。文献[8]考虑到云计算环境处理节点的安全性和可用性以及任务之间的相互依赖关系，采用有向无环图（DAG）描述一个应用中相互依赖的任务，提出一种云计算环境下融合安全与可用性的DAG任务调度策略。文献[9]提出一种包含优先级和条件边的集成控制流信息，扩展并行有向无环图任务模型，提高对条件任务的解决效率，减少传统的零星DAG任务系统的调度分析时间。

在异构系统中，基于负载均衡的调度使用最多的是RR算法和启发式算法[10]。RR算法采用处理节点轮询的方式，对任务进行调度，算法简单、快

速，比较适用于硬件实现，但是没有考虑不同任务间的差异以及关联任务的执行。考虑 QoS 需求，已有不少学者基于多目标约束，对经典的启发式算法，如遗传算法[11]、蚁群算法[12-14]、Sufferage[15]等进行改进并提出新的调度方案。文献[16]提出一种双目标优化粒子群算法（DPSO, discrete particle swarm optimization），在满足任务安全需求的前提下，缩短系统对任务的响应时间。文献[17]基于完成时间最快和用户支出费用最少 2 个目标，面向商业云提出一种基于完成时间和支付费用权重函数的动态工作流调度算法，为任务选取合适的处理节点。文献[18]针对多核异构分布式系统中流程序的执行，提出了 2 种新的启发式分区算法：KLA（Kernighan-Linadapted）算法和 CA（congestion avoidance）算法来解决空间图分割流程序的划分问题，以提高系统的吞吐率。文献[19]提出采用统计的方法来进行线程的分配，给每个线程分配一个随机样本，通过减少需要测量的候选解决方案数量的统计方法来降低线程分配时间，获得最佳性能。文献[20]采用多目标优化为约束条件，提出了一种基于混沌蚁群的遗传算法（GA-CAS），获得了优于其他启发式算法的收敛速度。

由于在实际应用中具有许多时刻变化的因素，通过静态调度找出最优调度方案几乎不可能，容易造成调度和过程优化控制相互脱节，因此，大规模动态复杂调度始终是研究中的热点与难点。本文基于海量高并发随机交叉密码服务的应用场景，建立一个与实际应用更接近的任务模型，在文献[2]提出的两级调度的基础上，通过对不同运算需求业务流的感知，实现作业包在处理节点上的合理分配，充分发挥异构多核的优势，实现快速的业务流处理。

3　任务模型

3.1　任务集

在流数据处理系统中，数据需要以包的形式进行传输和处理。根据作业包的大小不同，一个作业需要拆分为若干个作业包。假设系统中有 n 个作业，作业集合记为 $J=\{J_1,J_2,\cdots,J_i,\cdots,J_n\}$，作业 J_i 划分为 m 个作业包，可以表示为 $J_i=\{J_{i1},J_{i2},\cdots,J_{id},\cdots,J_{im}\}$。根据文献[2]中的描述，作业 J_i 的第 a 个作业包 P_{ia} 表示为

$$P_{ia}=\{task_i,cmd_i,cyp_i,mode_i,No_i,long,d_{ia}\}$$

其中，$task_i$ 为作业包所属业务的编号，同一作业的不同作业包该字段相同，是作业的唯一标识，同时保证处理完作业包的正确返回。cmd_i 为处理命令，决定对数据的具体操作，如加密、解密、签名、验签等。cyp_i 为密码算法模块 id 号，决定数据的处理节点，$cyp_i=\{cluster\_id \,\|\, ip\_id\}$，其中，高位表示簇号，低位表示簇内的算法处理节点号，即在流数据打包的软件层就确定了作业包与算法 IP 核间的映射关系。$mode_i$ 为密码算法工作模式，决定对密码算法处理节点入口数据的处理，如对于 ECB 模式可以将数据直接送入算法处理节点，而对于 CBC 模式，则需要将数据与同一个业务的前一组数据的运算结果进行异或后，再送入算法处理节点。No_i 为作业包的序号，$long$ 为作业包中运算数据的长度，d_{ia} 为运算数据。由于不同业务作业包的大小不尽相同，且不同的工作模式、不同算法的运算速度不同，因此在软件层无法精确地预知处理节点的负载状况，从而导致处理节点负载不均衡，影响系统的整体吞吐率。为此，本文将 cyp_i 修改为 $cyp_i=\{cluster\_id\}$，即在软件调度层仅确定作业包所需送至的算法处理节点簇，相同的算法实现处于相同的算法节点簇中。作业包与算法 IP 核间的具体映射关系由硬件调度模块根据处理节点的工作能力和负载情况实时确定。

流数据处理系统如图 1 所示，来自不同应用线程的业务通过作业提交模块将作业包发送至流数据处理系统的作业接收模块，作业接收模块按照时间顺序依次接收到各作业包。

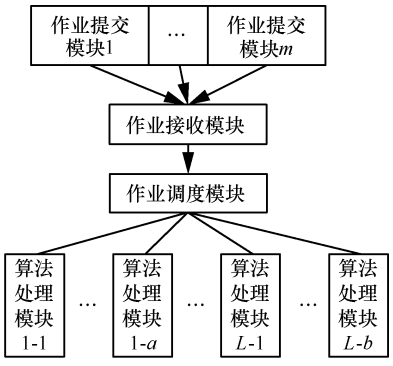


图 1　流数据处理系统

用 i 表示时间戳，则该时间戳的作业包为 T_i。多任务环境下，由于数据传输路径、速度等不同，不同业务请求的作业包到达作业调度系统的作业接收模块时间不同。所以对于同一个业务来说，虽然各作业提交模块按序提交各业务作业包，但是在作业接收模块中的作业包是乱序的，即存在不同业务间

的交叉接收。因此定义作业接收模块中的任务集为

$$T=\{T_i|T_i|=P_{mn},0<i<n_{\max}\}$$

其中，$n_{\max}$ 为一个作业窗口中作业接收模块能够接收的最大作业包的个数。

通过有向无环图（DAG，directed acyclic graph）对流数据任务集进行描述，可以清晰地表示流数据作业包的传递及作业包间的数据相关关系。图 2 为流数据任务集 DAG，共有 4 个任务：$JOB_1=\{T_1,T_2\}$，$JOB_2=\{T_3,T_5,T_9\}$，$JOB_3=\{T_4,T_7,T_8,T_{11}\}$，$JOB_4=\{T_6,T_{10}\}$。

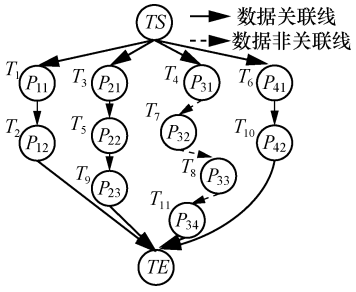


图 2 流数据任务集 DAG

其中，JOB_1、JOB_2、JOB_4 为依赖任务，JOB_3 为非依赖任务。以 *CBC* 加密模式为例介绍密码运算中的依赖任务：假设 $JOB_2.mode=CBC$，首先将明文分成 3 种固定长度的作业包，选取初始向量 ***IV***，对第一个作业包 T_3 进行加密，得到密文 T_3'；然后，将密文 T_3' 与下一个作业包 T_5 异或后作为初始向量加密，得到密文 T_5'；最后，将密文 T_5' 与下一个作业包 T_9 异或后加密，得到密文 T_9'。加密过程如图 3 所示。

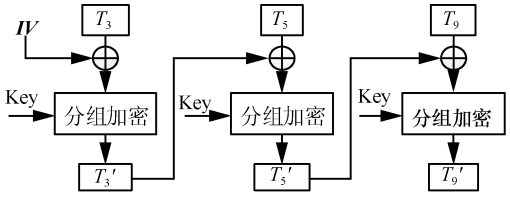


图 3 JOB_2 CBC 加密

图 2 中 *TS* 为任务集的入口节点，*TE* 为任务集的出口节点，实线箭头为数据关联线有序边。前序节点：数据关联线有序边的起点处的节点为终点处节点的前序节点。由图 3 可知，T_3 是 T_5 的前序节点，T_5 是 T_9 的前序节点。后继节点：数据关联线有序边的终点处的节点为起点处节点的后继节点。由图 3 可知，T_5 是 T_3 的后继节点，T_9 是 T_5 的后继节点。虚线箭头为数据非关联线有序边，其有序仅表现为时间传递上的有序性，并没有数据相关性。

通过流数据任务集的 DAG 模型可以看出，任务集有以下 2 种特点：同一业务的作业包在任务集中的排列是有序的；同一业务的作业包序列中穿插有其他业务的作业包。

3.2 作业窗口

作业调度系统根据作业接收模块中的任务集，对作业包进行调度。假设作业接收模块的存储容量为 m，数据流的传输速率为 s bit/s，作业包大小为 p，则作业窗口时间 t 必须满足

$$st\leqslant m$$

且在 t 作业窗口时间内，作业接收模块能够接收到的作业包的个数最大为

$$n_{\max}=t\frac{s}{p}$$

作业包的大小是可变的，由规则化数据结构中的数据长度决定，作业窗口的大小必须保证作业接收模块能够完整地接收作业提交系统传递过来的作业包。

4 作业调度算法

4.1 作业调度模型

作业调度模型如图 4 所示，包括 7 个部分：作业接收队列、作业分转模块、簇任务队列、负载均衡控制模块、处理节点任务队列、中间状态存储模块和等待任务队列。

作业接收队列接收输入的任务集；作业分转模块进行作业的第一级调度，按照 cyp_i 将作业包送至相应的簇任务队列；负载均衡控制模块进行作业的第二级调度，按照负载均衡策略，将作业包送至相应的处理节点任务队列；处理节点任务队列按照先来先服务的原则，将作业包送至处理节点进行算法处理；中间状态存储模块用于暂存依赖作业前序作业包的中间结果，为后继作业包提供中间数据；等待任务队列是为了保证数据流的连续性，防止由于某个簇任务队列满导致的数据流阻塞，用于暂存已满簇任务队列的作业包。

4.2 计算资源

处理节点集 $IP=\{IP_{i1},IP_{i2},\cdots,IP_{im},\cdots,IP_{p1},\cdots,IP_{pn}\}$ 为任务所能获得的计算资源，每个处理节点具有各自的适用领域 *field*、运算速度 s 和存储容量 *cap*。其中，适用领域描述该处理节点的功能，与

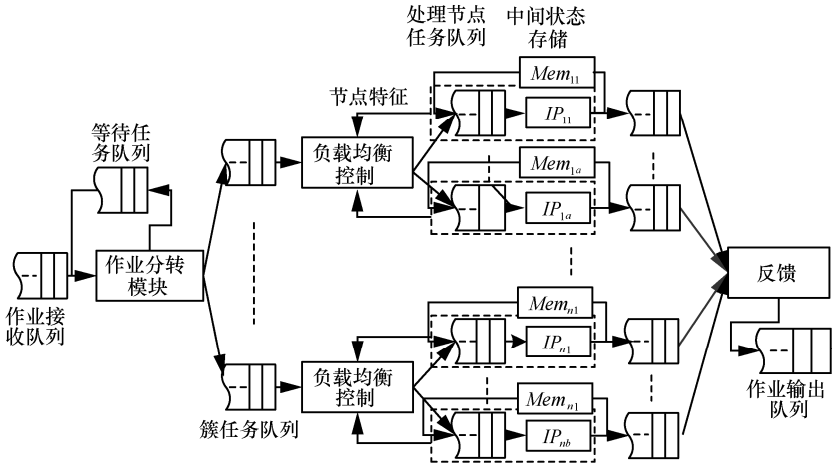


图 4　作业调度模型

任务属性中的 cyp_i 对应；运算速度取决于处理节点本身的设计，以密码算法 IP 核为例，包括密码算法设计中的信号流水线处理、密钥扩展的同步处理以及算法实现的工作频率。存储容量取决于处理节点任务队列的大小，存储容量必须为单元作业包的整数倍，否则会造成数据丢失。即

$$IP_{ij}=\{field,s,cap\},cyp_{id}\in field$$

4.3　作业调度策略

本文所研究的基于负载均衡的动态作业调度算法采用了静态参数与动态参数结合的方法。以处理节点的实际处理能力为静态参数，处理节点任务队列的剩余容量为动态参数；按照可调节的比例得到一个权重来决定作业包的优先级。

1) 处理节点状态表

每个算法簇的负载均衡控制模块维护一个处理节点状态表：$S_{ip}=\{t_{\text{unit}},cap,n\}$。负载均衡调度算法根据此状态表进行处理节点权重的计算，进行作业包调度的动态调整。其中，t_{unit} 为单元作业包 P_{unit} 的处理时间，cap 为处理节点的任务队列存储容量，n 为任务队列中已有的作业包的个数。处理节点任务队列中每接收一个作业包，则 n+1，每输出一个作业包，则 n−1。

由于每个处理节点上可以执行的操作类型不同，如加密、解密、签名、验签，而不同的操作消耗的时间不同，因此单元作业包在不同处理节点上的处理时间为

$$t_{\text{tunit\_}i}=\{t_{i1},t_{i2},t_{i3},t_{i4}\}$$

其中，t_{i1} 表示第 i 个处理节点加密单元作业包的时间，t_{i2} 表示第 i 个处理节点解密单元作业包的时间，t_{i3} 表示第 i 个处理节点对单元作业包签名需要的时间，t_{i4} 表示第 i 个处理节点对单元作业包验签需要的时间。因此，具有 m 个算法 IP 的算法 i 的处理节点状态矩阵为

$$\boldsymbol{s}=\begin{bmatrix}t_{11} & t_{12} & t_{13} & t_{14} & cap_1 & n_1\\ t_{m1} & t_{m2} & t_{m3} & t_{m4} & cap_m & n_m\end{bmatrix}$$

2) 处理节点权重

处理节点的权重与处理节点的剩余容量以及处理速度有关。由于处理节点的速度和容量的数量级不同，数据尺度的不统一对处理节点的选择影响很大，必须进行调整。将节点的处理速度归一化至[0,1]区间。

$$W_s=\frac{t_{\text{unit}}-t_{\min}}{t_{\max}-t_{\min}}$$

其中，$t_{\min}$ 为簇上运算最快的节点处理单位作业包所需的时间，$t_{\max}$ 为簇上运算最慢的节点处理单位作业包所需的时间。以具有 m 个算法 IP 的算法 i 的簇为例，有

$$t_{\min}=\{t_{\min_1},t_{\min_2},t_{\min_3},t_{\min_4}\}$$

$$t_{\max}=\{t_{\max_1},t_{\max_2},t_{\max_3},t_{\max_4}\}$$

$$t_{\min_i}=\min_{1\leqslant j\leqslant m}t_{ji},1\leqslant i\leqslant 4$$

$$t_{\max_i}=\max_{1\leqslant j\leqslant m}t_{ji},1\leqslant i\leqslant 4$$

因此，对于每一个处理节点来说，其速度权重为一固定向量。处理节点的剩余容量归一化至[0, 1]区间。

$$W_c=\frac{cap_r}{cap}=\frac{cap-np}{cap}$$

其中，cap 为处理节点的任务队列容量，cap_r 为处理节点的任务队列剩余容量，可通过队列的容量减去已

经使用的存储偏移量获得，p 为作业包的大小，n 表示任务队列中已有 n 个作业包，n 可以通过处理节点任务队列计数器实现。则处理节点的权重函数为

$$W_{ip} = -\alpha w_s + \beta w_c, \alpha + \beta = 1$$

其中，α 为速度权重系数，β 为剩余容量权重系数。为便于后期处理节点权重的计算，将处理节点状态矩阵替换为

$$\boldsymbol{s}' = \begin{bmatrix} w_{s1\_1} & w_{s2\_1} & w_{s3\_1} & w_{s4\_1} & cap_1 & n_1 \\ w_{s1\_m} & w_{s2\_m} & w_{s3\_m} & w_{s4\_m} & cap_m & n_m \end{bmatrix}$$

3) 选择关系表

选择关系表用于描述进入处理节点任务队列的依赖任务 ID 和处理节点 ID 间的对应关系，为一个二元表：$choose\_list = \{task_i, ip\_id\}$。选择关系表的长度 *choose_list.length* 取决于算法簇处理节点的个数，每个算法簇维护一个选择关系表。若某算法簇中的处理节点为 n 个，则

$$choose\_list.length = n$$

4.4 调度算法

1) 第一级分转调度

根据 cyp_i 进行调度，按照先来先服务的原则将不同算法处理需求的作业包放入对应算法簇任务队列中；当任务集中某一算法处理需求的作业包大量流入导致相应的簇任务队列满时，将当前作业包放入等待任务队列中，并置 mask 为 1，标识此单元被占用，继续进行下一个作业包的调度；当簇任务队列有数据移出时，则查询等待任务队列，将先进入的同一算法需求的作业包取出放入簇任务队列中，如图 5 所示。

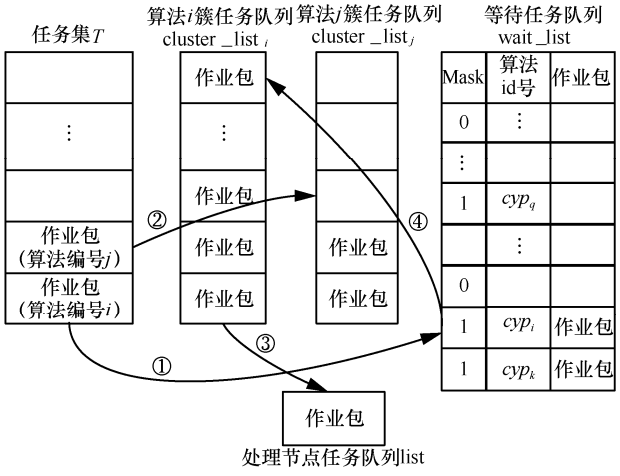


图 5 第一级分转调度模型

等待任务队列能够使任务集中的作业包不断地流入，避免了因某个算法簇的任务执行速度慢导致队列满，进而导致数据流的停滞。

2）第二级负载均衡调度

策略 1 为保证信号的快速处理，优先考虑算法的存储容量，当存储容量不能满足时，就没有必要再考虑运算速度。当存储容量满足时，选取权重大的处理节点进行作业包的处理。

策略 2 当某依赖任务的前序作业包在处理节点 i 上运算时，其后继作业包也必须在此处理节点上运算。此策略可以免除数据迁移带来的传输时延，获得较好的数据本地化。

具体作业调度算法步骤如下所示。

Step1 根据工作模式 $mode_i$，判断依赖性。

若 $mode_i$ 为 CBC|OFB|CFB，则此作业属于依赖作业。首先查询选择关系表，若选择关系表中存在此作业包的 $task_i$ 表项，则将此作业包放置于此表项对应的处理节点 *ip_id* 的任务队列中，进入 Step7；若选择关系表中不存在此作业包的 $task_i$ 表项，则将此 $task_i$ 加入选择关系表中，进入 Step2。

若作业为非依赖作业，进入 Step2。

Step2 根据处理节点状态表，获取本簇处理节点的剩余容量 $cap_r = cap - np$。

Step3 根据处理节点状态表和处理算法类型，获取剩余容量非空处理节点的速度权重。

Step4 计算剩余容量非空处理节点的权重，选取权重最大的处理节点。

Step5 若作业包属于依赖作业，将此处理节点 *ip_id* 加入 $task_i$ 对应的表项；否则，进入 Step6。

Step6 将作业包放入处理节点 *ip_id* 的任务队列中。

Step7 若作业包为依赖作业包，在作业包处理完成时，判断此作业包是否为作业的最后一个分片，若是则从选择关系表中将此作业包 $task_i$ 对应的表项删除，并将运算结果直接返回业务请求端；否则将此作业包的运算结果作为中间状态保存至中间状态存储模块。若作业包不属于依赖作业包，则将运算结果直接返回业务请求端。

5 分析及验证

本文对任务调度负载均衡性和执行效率进行了仿真，并与基于业务标识的分层硬件调度方法 HHS-ACDID 进行对比分析。仿真在 Windows 环境下，通过 C++语言实现算法。由于每一个簇的调度

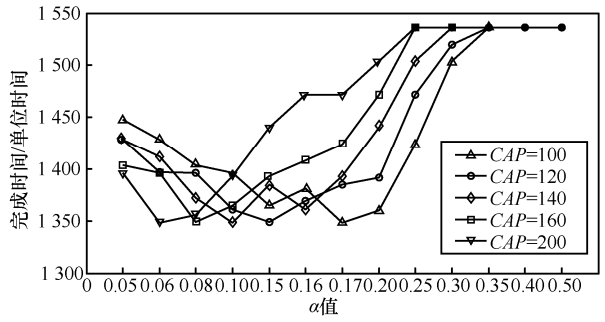
过程是相似的，选取一个簇下的调度进行仿真。

任务集信息参数设置如表 1 所示。其中，$task_i$ 为作业编号；No_i 为不同作业包编号；*flag* 为依赖作业标识；*arrivalTime* 为作业包进入作业调度模块的时刻；*long* 为作业包大小。

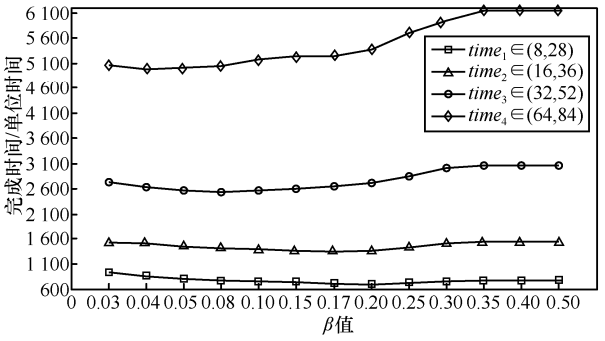
表 1　　任务集的参数设置

参数类型	值
$task_i$	随机生成，$task_i \in [1,50]$
No_i	随机生成，$No_i \in [1,7]$
flag	随机生成，$flag \in [0,1]$
arrivalTime	0,1,…
long	单位作业包长度

仿真实验 1　权重系数 α、β 对完成时间的影响。假设当 $\alpha = \alpha_i$ 时，系统获得最小完成时间。IP 核数=4，随机生成 4 组处理速度 $time_{ij} \in [16,36], i,j \in [1,4]$。改变处理节点队列存储容量初始值 *CAP*，观察 *CAP* 与 α_i 之间的关系如图 6(a)所示，可见在其他参数相同的情况下，取得最小完成时间的 α_i 的取值随 *CAP* 的增大而减小。IP 核数=4，单位作业包完成时间分别为 $time_1 \in (8,28)$，$time_2 \in (16,36)$，$time_3 \in (32,52)$，$time_4 \in (64,84)$，$CAP = 200$，仿真结果如图 6(b)所示，取得最小完成时间的 α_i 的变化趋势与图 6(a)基本一致。



(a) *CAP*对α_i的影响



(b) IP核处理时间对α_i的影响

图 6　α、β 与完成时间关系

仿真实验 2　权重系数对负载均衡性和执行效率的影响。采用相同的任务集，作业包总数=600，IP 核数=4，$time_{ij} \in [16,36], i,j \in [1,4]$，改变 $\alpha \in [0,1]$，$\beta \in [0,1]$ 的取值，负载分布和完成时间如图 7 所示。

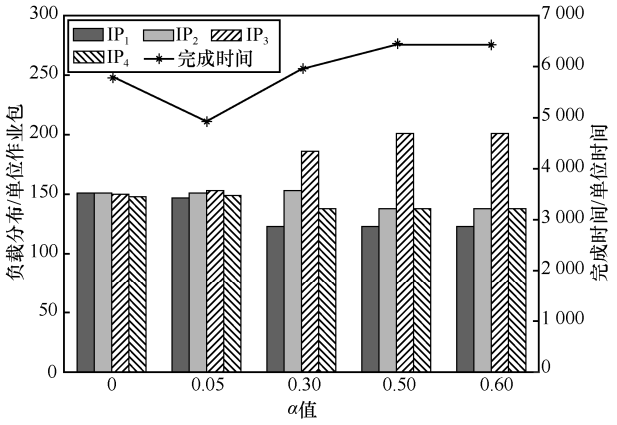


图 7　α、β 对调度结果的影响

从图 7 可以看出，随着处理节点速度权重系数 α 增大，IP 核负载分布趋向于不均衡；系统完成时间先减小后增大，最后趋于不变。其中，当 α=0.05 时，完成时间取得最小值，负载分布较均衡。由此可知，α、β 的取值影响任务调度过程中 IP 核的负载分布和完成时间，以下实验都将在完成时间最小的前提下讨论。

仿真实验 3　算法的负载均衡性。处理节点性能不变，$time_{ij} \in [16,36], i,j \in [1,4]$，IP 核数=4，作业包总数分别为 100、200、300、400、500、600、700，负载分布情况如图 8 所示。

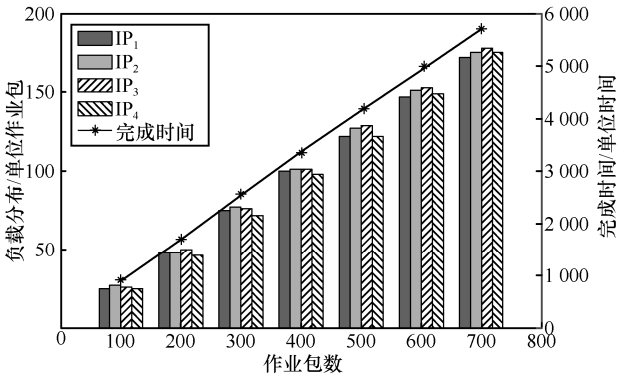


图 8　4 个 IP 核的负载分布

IP 核数为 10，作业包数分别为 500、700、1000、1100、1300、1500，负载分布情况如图 9 所示。

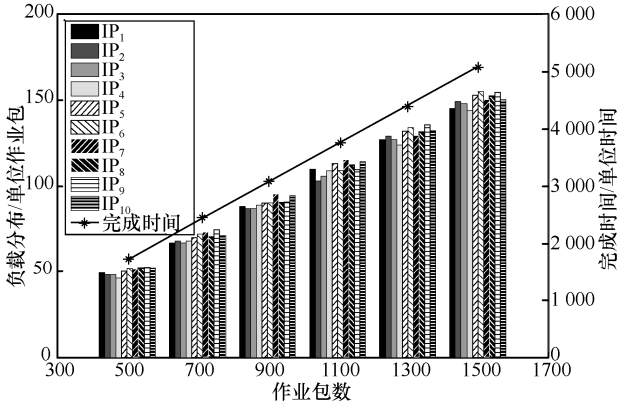


图 9 10 个 IP 核的负载分布

当处理节点数分别为 4 和 10 时，反映出相同的规律：随着作业包数的增加，调度结果保持良好的负载均衡性。

仿真实验 4 从 2 个方面比较本文算法与分层硬件调度方法 HHS-ACDID 的执行效率：固定 IP 核数，改变作业包数；固定作业包数，改变 IP 核数。

1) 固定 IP 核数为 4，作业包总数变化（100~700）；固定 IP 核数为 10，作业包数变化（500~1500），本文算法与 HHS-ACDID 算法的执行时间比较分别如图 10 和图 11 所示。

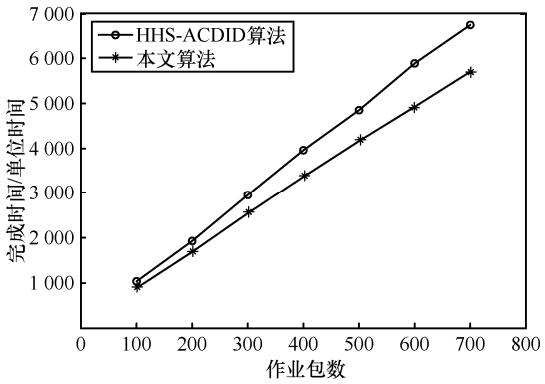


图 10 4 个 IP 核的完成时间

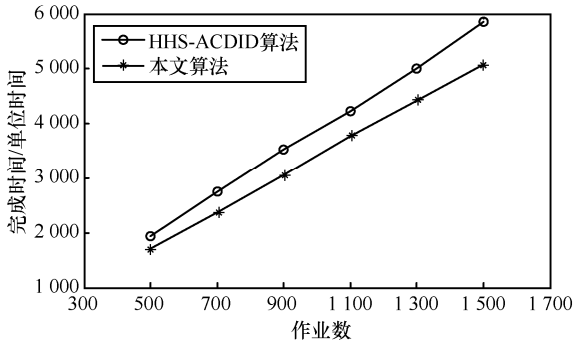


图 11 10 个 IP 核的完成时间

由图 10 和图 11 可知，在相同的实验环境下，本文算法的执行效率明显优于 HHS-ACDID 算法，执行时间比 HHS-ACDID 算法减少约 10.6%~15.3%，并且随着作业包数的增多，其优势更加明显。这主要是因为与 HHS-ACDID 算法相比，本文算法同时考虑不同处理节点的处理性能和当前负载情况，保证了处理节点的负载均衡和任务的处理速度，使高性能的处理节点得到更多的使用率。

2) 固定任务数=1 000，IP 核数变化（4~10 个 IP 核）时，本文算法与 HHS-ACDID 算法执行时间如图 12 所示。

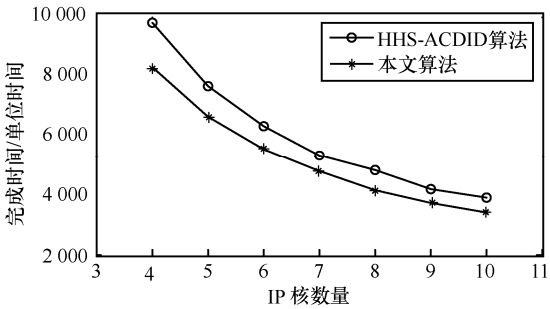


图 12 不同 IP 核下的任务完成时间对比

由图 12 可知，对于相同的任务集，本文算法的执行时间比 HHS-ACDID 算法少 10.3%~15.3%。IP 核数量较少时优势明显；随着 IP 数量的增加，其优势有所降低。

可见，通过硬件实时监控处理节点任务队列的剩余容量，在任务数较大时，提高速度权重系数，减小剩余容量权重系数；任务数较小时，提高剩余容量权重系数，减小速度权重系数，能获得较佳的负载均衡效果。根据测试结果，在作业量增大的趋势下，本文算法展现出更大的优势。

6 结束语

本文针对多算法密码流系统模型，提出了一种基于负载均衡的两级作业调度算法，首先根据作业包的算法标识 cyp_i 把任务递交到相应的簇；然后根据作业包的工作模式 $mode_i$ 以及处理节点状态表进行二级均衡调度。该算法适用于关联任务和非关联任务以及混合情况，实现上采用三级存储层次的流水架构，通过作业分转模块、负载均衡控制模块以及等待任务队列，保证了数据流传输的连续性，不存在作业的重分配以及因作业需求类型不同导致的某个作业长期占用计算资源，而其他作业得不到

响应的情况，也起到了隐藏传输时延、改善带宽效应的作用。仿真结果表明，该算法在保证系统负载均衡的基础上，执行效率比 HHS-ACDID 算法高12%左右。

参考文献：

[1] 李风华,殷丽华,吴巍,等. 天地一体化信息网络安全保障技术研究进展及发展趋势[J]. 通信学报,2016,37(11):156-168.
LI F H, YIN L H, WU W, et al. Research status and development trends of security assurance for space-ground integration information network[J]. Journal on Communications, 2016,37(11):156-168.

[2] 李莉,史国振,耿魁, 等. 密码芯片的多算法随机作业流调度方法[J]. 通信学报,2016,37(12):86-94.
LI L, SHI G Z, GENG K,et al. Stochastic job stream scheduling method for cipher chip with multi-cryptography[J]. Journal on Communications, 2016,37(12):86-94.

[3] LIU L D, QI D Y, CHEN Q, et al. Efficient scheduling mechanism for performance-heterogeneous multi-core processor[C]//2014 5th International Conference on Digital Home (ICDH). 2014:342-346.

[4] VAN C K, JALEEL A, EECKHOUT L, et al. Scheduling heterogeneous multi-cores through performance impact estimation (PIE)[J]. ACM Sigarch Computer Architecture News, 2012, 40(3):213-224.

[5] LI T, BAUMBERGER D, HAHN S. Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin[C]//The 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel programming. 2009: 65-74.

[6] LI T, BRETT P, KNAUERHASE R, et al. Operating system support for overlapping-ISA heterogeneous multi-core architectures[C]//2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA). 2010:1-12.

[7] 杨茂林, 雷航, 廖勇. 一种共享资源敏感的实时任务分配算法[J]. 计算机学报, 2014, 37(7):1455-1465.
YANG M L , LEI H , LIAO Y . A shared resource-aware real-time task allocation algorithm[J]. Chinese Journal of Computers, 2014, 37(7): 1455-1465.

[8] 刘亚秋, 邵洪润, 景维鹏. 云环境下融合安全与可用性的 DAG 任务调度[J]. 计算机工程, 2014(12):12-18.
LIU Y Q, SHAO H R, JING W P. DAG task scheduling integrating with security and availability in cloud environment[J]. Computer Engineering, 2014(12):12-18.

[9] MELANI A, BERTOGNA M, BONIFACI V A. Schedulability analysis of conditional parallel task graphs in multicore systems[J]. IEEE Transactions on Computers, 2017,66(2) :339-353.

[10] ZOMAYA A Y, THE Y H. Observations on using genetic algorithms for dynamic load-balancing[J]. IEEE Transactions on Parallel and Distributed Systems, 2001,12(9): 899-911.

[11] XU Y, LI K, HU J, et al. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues[J]. Information Sciences, 2014, 270(6):255-287.

[12] TASGETIREN M F, PAN Q K, SUGANTHAN P N, et al. A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion[J]. Applied Mathematical Modelling, 2013, 37(s10-11):6758-6779.

[13] JIN H , RAN L. A fair-rank ant colony algorithm in distributed mass storage system[J] . Canadian Journal of Electrical and Computer Engineering, 2015,38(4): 338-345.

[14] ALI A, BELAL M A, AL-ZOUBI M B. Load balancing of distributed systems based on multiple ant colonies optimization[J]. American Journal of Applied Sciences, 2010, 7(3): 433-438.

[15] TABAK E K, CAMBAZOGLU B B, AYKANAT C. Improving the performance of independenttask assignment heuristics minmin, maxmin and sufferage[J]. IEEE Transactions on Parallel & Distributed Systems, 2014, 25(5):1244-1256.

[16] 朱海, 王宇平. 融合安全的网格依赖任务调度双目标优化模型及算法[J]. 软件学报, 2011, 22(11):2729-2748.
ZHU H, WANG Y P. Integration of security grid dependent tasks scheduling double-objective optimization model and algorithm[J]. Journal of Software,2011,22(11): 2729-2748.

[17] MOHAMMADI F H, PRODAN R, FAHRINGER T. A truthful dynamic workflow scheduling mechanism for commercial multicloud environments[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(6):1203-1212.

[18] NGUYEN V T N, KIRNER R.Throughput-driven partitioning of stream programs on heterogeneous distributed systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2016,27(3): 913-926.

[19] RADOJKOVIC P, CARPENTER P, MORETO M, et al. Thread assignment in multicore/multithreaded processors: a statistical approach[J]. IEEE Transactions on Computers, 2015, 65(1):256-269..

[20] CUI H, LI Y, LIU X, et al. Cloud service reliability modelling and optimal task scheduling[J]. IET Communications, 2017, 11(2): 161-167.

[作者简介]

李莉（1974-），女，山东青岛人，西安电子科技大学博士生，北京电子科技学院副教授、硕士生导师，主要研究方向为网络与系统安全、嵌入式系统安全应用。

史国振（1974-），男，河南济源人，博士，北京电子科技学院副教授、硕士生导师，主要研究方向为网络与系统安全、嵌入式安全。

耿魁（1989-），男，湖北红安人，博士，中国科学院信息工程研究所助理研究员，主要研究方向为网络安全。

董秀则（1976-），男，山东莒县人，北京电子科技学院讲师，主要研究方向为信息安全、密码工程实现。

李凤华（1966-），男，湖北浠水人，博士，中国科学院信息工程研究所副总工、研究员、博士生导师，主要研究方向为网络与系统安全、可信计算。