

doi:10.3969/j.issn.1007-7162.2016.01.006

基于离散事件系统的云资源分配优化控制

胡 芹, 刘富春

(广东工业大学 计算机学院, 广东 广州 510006)

摘要: 如何对资源进行合理有效的分配一直是云计算领域的热点问题. 由于传统的云资源分配算法不能有效屏蔽底层硬件资源的异构化以及不同层的云服务类别, 本文提出一种多参数资源打包的方法, 构建出基于离散事件系统的云资源分配控制模型, 并给出了合理的资源分配算法. 算法通过计算服务器端各资源包容量参数与客户端资源需求量参数的贴程度来定义资源分配事件, 并对事件发生与否的状态实行分层控制, 最终使得整个资源分配系统到达可接受状态. 实验表明, 基于离散事件系统的云资源分配模型, 能够保证在可接受状态下, 不仅每个用户的资源请求能够得到合理的分配, 且能实现云资源利用率最大化.

关键词: 云资源分配; 离散事件系统; 资源包; 资源利用率

中图分类号: TP301.1

文献标志码: A

文章编号: 1007-7162(2016)01-0029-07

Optimization Control of Cloud Resource Allocation Based on DES

Hu Qin, Liu Fu-chun

(School of Computers, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: How to allocate resources reasonably and effectively has been an important problem in the field of cloud computing. For traditional cloud resource allocation algorithms can not effectively shield the isomerization of underlying hardware resources, and they also can not neglect the categories of different layers' cloud service, this paper proposes a multi-parameter resource packing method, and constructs a cloud resource allocation control model based on discrete event system. Besides, it gives a reasonable resource allocation algorithm. By calculating the close degree of capacity parameters from each resource package over the servers and the resource demands of the clients, the algorithm defines the resource allocation events, and hierarchically controls the states' occurrence or nonoccurrence. Finally, the entire resource allocation system will reach an acceptable state. The experiments show that the cloud resource allocation model based on the discrete event system can guarantee that under the acceptable state, not only each resource request can get a reasonable allocation, but also the resource can get a maximal utilization.

Key words: cloud resource allocation; discrete event systems(DES); resource packages; resource utilization rate

云计算^[1-2]为用户提供灵活的计算能力和高效的数据存储能力, 云资源分配将用户请求的虚拟机分配给各个底层物理机去处理^[3], 由分布在不同地方的物理机提供用户所需的资源, 满足用户请求的执行条件. 在云环境下, 资源分配的效率显得非常重要, 其对云平台系统的综合性能影响很大. 当前, 云计算下的资源分配已经进行了大量的研究, 已经投

入使用的著名的 MapReduce 调度模型^[4], 通过 3 种角色 (Master、Worker、User) 的分工协作, 能高效地为用户请求分配合适的计算资源; 文献[5]提出了基于粒子群优化算法的云资源分配, 采用进化算法原理, 依据用户需求大小来适当地分配资源量; 文献[6-9]提出了基于 Qos (Quality of service) 任务分配和蚁群算法的资源分配与任务调度算法, 先采取

收稿日期: 2015-01-15

基金项目: 国家自然科学基金资助项目(61273118); 广东省高校省级重大科研项目(2014KZDXM033); 广东省公益研究与能力建设专项资金项目(2015A030402006)

作者简介: 胡芹(1989-), 女, 硕士研究生, 主要研究方向为离散事件系统的监督与控制应用.

通信作者: 刘富春(1971-), 男, 教授, 主要研究方向为离散事件系统理论. E-mail: liufch@gdut.edu.cn.

Qos 算法,将不同的资源请求分类处理,再利用蚁群算法得到一组最优的计算资源.文献[10]分别研究了在分布式 DES(Discrete Event Systems)监控和协调器协调监控中的两类具体问题:作业调度问题 TAP(Task Allocation Problem)、最少干预问题 MIP(Minimal Intervention Problem);文献[11-14]研究了基于模拟退火的蚁群算法求解网格任务调度问题;文献[15]研究了云环境中面向随机任务的用户效用优化模型.然而以上研究都没有考虑具体的资源分配模型.本文将成熟的离散事件系统(DES)的控制理论应用到资源分配上,基于有穷自动机的资源分配的基础,建立了 DES 云资源控制模型.将所有资源分配事件都看作是可控事件,通过控制可控事件的发生与不发生,保证提高资源利用率的事件有效发生.将用户请求的虚拟机准确分配到具体的资源包,通过资源参数值与任务对应参数值的对比,调整资源分配,在满足资源利用率最大的同时,可以实现有效的资源分配.

1 云资源分配模型

云资源分配是一个 NP(Non-deterministic Polynomial)难问题,到目前为止还没有确切的资源分配模型,但是存在对资源分配的多种优化算法,一些启发式算法将云资源分配问题转换为 NP 难优化问题^[16-17].本文建立合理的 DES 资源分配模型,在该模型中,云计算数据中心是由大量的异构的计算资源构成的云资源池,这些海量的资源均是客户端请求执行所必须的,如 CPU、内存、存储空间、防护服务、服务器、带宽等.考虑到基本的云计算三层结构模型,在每一层中,云提供服务可以抽象为“XaaS”,如基本架构即服务“IaaS(Infrastructure as a Service)”,平台即服务“PaaS(Platform as a Service)”,软件即服务“SaaS(Software as a Service)”.为了屏蔽不同层的服务类别,本文将资源池的资源看作为容量不一的资源包,每个资源包中含有多种不同资源,将不同类别资源表征为该资源包具有的不同参数,如参数 1 为 CPU 数量,参数 2 为存储容量,参数 3 为带宽等等.同理,用户任务请求所需资源类别也进行参数化处理,如某一个任务,所需要资源配置的参数 1、参数 2、参数 3、…分别为完成该任务所需要的 CPU、内存、存储空间、…的资源量.如此,每一层的服务对所有用户都是透明的,用户只需要申请和使用所需的资源即可,不需要关心资源是如何分配的.

本文重点考虑为已经到达任务处理缓冲池中的用户请求分配资源,设资源池中的资源为 $R = \{R_1, R_2, \dots, R_n\}$, R_i 表示资源池中的第 i 类资源,资

源池中共有 n 种资源;用户请求的任务队列为 $T = \{T_1, T_2, \dots, T_n\}$,其中 T_x 表示第 x 个用户请求; $P = \{P_1, P_2, \dots, P_n\}$ 表示资源包队列,其中 P_y 表示第 y 个资源包; $C_{R_i}^{T_x}$ 表示第 x 个用户任务所需要的 R_i 类资源量, $C_{R_i}^{P_y}$ 表示第 y 个资源包所拥有的资源 R_i 的量.如此,资源包的资源分配情况可以形式化为 $P_i = \{C_{R_1}^{P_i}, C_{R_2}^{P_i}, \dots, C_{R_n}^{P_i}\}$,单个用户请求资源需求量可以形式化为 $T_i = \{C_{R_1}^{T_i}, C_{R_2}^{T_i}, \dots, C_{R_n}^{T_i}\}$,在资源包和请求中,若不存在此类资源,则将此类资源参数数量设置为 0.

资源包含有的资源以及任务请求资源情况均可以用一个 $n \times m$ 阶矩阵表示,具体表示如下:

$$\begin{aligned} & \begin{matrix} & R_1 & R_2 & \cdots & R_m \\ P_1 & \begin{bmatrix} C_{R_1}^{P_1} & C_{R_2}^{P_1} & \cdots & C_{R_m}^{P_1} \\ C_{R_1}^{P_2} & C_{R_2}^{P_2} & \cdots & C_{R_m}^{P_2} \\ \vdots & \vdots & & \vdots \\ C_{R_1}^{P_n} & C_{R_2}^{P_n} & \cdots & C_{R_m}^{P_n} \end{bmatrix} \\ P_2 & \\ \vdots & \\ P_n & \end{matrix} \\ & \begin{matrix} R_1 & R_2 & \cdots & R_m \\ T_1 & \begin{bmatrix} C_{R_1}^{T_1} & C_{R_2}^{T_1} & \cdots & C_{R_m}^{T_1} \\ C_{R_1}^{T_2} & C_{R_2}^{T_2} & \cdots & C_{R_m}^{T_2} \\ \vdots & \vdots & & \vdots \\ C_{R_1}^{T_n} & C_{R_2}^{T_n} & \cdots & C_{R_m}^{T_n} \end{bmatrix} \\ T_2 & \\ \vdots & \\ T_n & \end{matrix} \end{aligned} \quad (1)$$

在资源分配过程中,要求任务请求的各个参数资源总量不得超过资源包中各个参数资源的资源总量,且资源包各个参数资源总量不得超过资源池中该类资源的资源总量,即要求满足

$$\forall i \in N, \sum_{x=1}^n C_{R_i}^{T_x} < \sum_{y=1}^m C_{R_i}^{P_y} < R_i. \quad (2)$$

初始化资源分配调度也是一个 $n \times m$ 阶矩阵,为

$$\begin{aligned} & \begin{matrix} P_1 & P_2 & \cdots & P_m \\ T_1 & \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \\ T_2 & \\ \vdots & \\ T_n & \end{matrix} \end{aligned}$$

其中, $A[T_x, P_y] \in \{0, 1\}$, 值为 1 表示第 x 个任务获得了第 y 个资源包的资源,为 0 则表示没有获得.当为任务请求分配资源时,要求保证每个任务都可以分配一个确定的资源包,一个包可以被多个任务共享.因此,资源分配矩阵中的每一行有且只有 1,每一列存在一个或者多个 1.

云资源分配的最理想状况是任务群被分配的总资源量等价于资源池中的总资源量,即

$$\sum_{i=1}^k R_i - \sum_{x=1}^n \sum_{i=1}^k C_{R_i}^{T_x} = 0.$$

然而,这种理想的状态目前尚不能达到,因为存在部分资源不能完全被利用. 若要实现分配之后的各个资源包的资源参数能够最大限度地满足任务需求的各个参数的资源量,求出资源包的容量值与用户请求资源的容量欧氏距离

$$D(x, y) = \sqrt{(C_{R_1}^p - C_{R_1}^t)^2 + (C_{R_2}^p - C_{R_2}^t)^2 + \dots + (C_{R_k}^p - C_{R_k}^t)^2}. \quad (3)$$

距离越小,表明资源包容量与用户请求资源量的贴进度越大. 对于任意给定的 $x \in N$, 要得到最大的贴进度,即找出唯一 $y \in N$, 存在

$$D(x, y) = \min\{D(x, y) | x, y \in N\},$$

则返回资源分配结果,令

$$A[T_x, P_y] = 1; A[T_x, P_y] = 0.$$

为了达到资源的最优分配,要求满足所分配的资源包能够得到最大化利用,即利用率之和最大.

$$U = \sum_{y=1}^n A[T_x, P_y] \frac{\sum_{x=1}^n \sum_{i=1}^k C_{R_i}^{T_x}}{\sum_{i=1}^k C_{R_i}^{P_y}}. \quad (4)$$

2 构造 DES 控制模型

构造资源分配系统的 DES 控制模型,设系统 $\Phi = \{Q, \Sigma, \delta, q_0, q_a\}$, 状态集 $Q = \{q_0, q_1, \dots, q_k\}$, $q_i = \{\exists i \in N, P_1 \cup P_2 \dots \cup P_i\}$ 表示在该状态下,存在一个或多个资源包以供选择;事件集 Σ 为并行执行的任务 T 串,如 $\{T_i\}, \{T_i T_j\}, \{T_i T_j T_k\}, \dots$; 转移函数 $\delta: Q \times \Sigma^* \rightarrow Q$, 当为某个用户任务分配资源包之后,系统跳转到另一个状态; $q_0, q_a \in Q$, $q_0 = \{\forall i \in N, P_1 \cup P_2 \dots \cup P_i\}$ 表示系统的起始状态,在起始状态,任意的资源包都可供选择, q_a 为资源分配系统的接受状态,在该状态下,所有等待的用户请求都可以获得资源包,且资源包的利用率最大.

对于该资源分配系统,资源分配事件 $L \subseteq \Sigma^*$ 分为可满足事件 L_c 、不可满足事件 L_u 和可接受事件

L_a . 其中可满足事件

$$L_c = \{s \in L: \forall T_j \in s, \exists P_j, \forall i \in N (C_{R_i}^{T_j} \leq C_{R_i}^{P_j})\}$$

表示只有在当前状态中的某个确定的资源包参数能够满足该事件串中的所有事件的子串所对应的资源需求量,该事件才是可满足的;不可满足事件

$$L_u = \{\forall T_j \in s: \forall P_j, \exists i \in N (C_{R_i}^{T_j} > C_{R_i}^{P_j})\}$$

表明存在某个资源包的参数不能满足对应请求的资源参数,显然,以 L_u 为子串的所有事件串均为不可满足事件;可接受事件

$L_a = \{\forall T_j \in s: \max(U)\} \subseteq L_c$ 的发生能够满足资源包的最大化利用.

在该资源分配系统中,一个资源包可以同时满足多个用户请求,资源包和用户请求之间是一对多的关系,所有可满足事件的发生都可以是并行的. 由于在并行状态下的多个事件串的发生,不具有可观性;又由于每个用户请求只能被一个资源包服务,资源包与任务请求之间不具有交叉性,因此,在该分配模型中,可以考虑系统中资源包分配的串行执行情况,分别考虑当前状态中每一个资源包在各个事件串发生后的状态. 最后通过分析比较,确定可接受事件串,控制不可满足事件的不发生,以及控制可接受事件的发生来确保系统资源利用率最大化.

系统初始状态包含所有未被分配的资源包,考虑资源池中的每一个资源包 P_m , 假设某个事件是可满足事件,则考虑该可满足事件的扩充串 (enlarge 语言),直到找出在该资源包下最大的可满足事件 L_c , 并修改状态集 Q 中 P_m 的参数,其他资源包保持不变,若 T_i 是属于该可满足语言,则将 P_m 资源包分配给任务 T_i ,修改事件集 $\Sigma' = \Sigma - T_i$,使得已经获得资源包的用户请求退出缓冲池,接下来考虑包 P_n ($n \neq m$),转移函数 $\delta: Q \times \Sigma' \rightarrow Q$,直到所有任务都可以得到分配. 假设系统存在 2 个资源包 P_1, P_2 , 3 个任务 T_1, T_2, T_3 , 如图 1 所示,在 q_0 状态下考虑资源包 P_1 , 若任务事件 T_1 属于可满足事件,接下来考虑事件 $T_1 T_2, T_1 T_3, T_1 T_2 T_3$ 是否为可满足事件,即判断

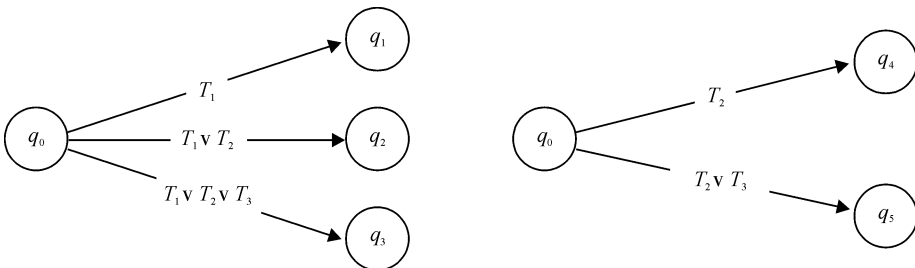


图1 某个资源包下的状态迁移

Fig. 1 State migration in a resource package

Σ^* 中所有包含可满足事件 T_1 的所有串是否仍为可满足事件,找出包含 T_1 的最大可满足串;接下来判断,若事件 T_2 不属于可满足事件,即状态 4 不存在,则不用考虑包含 T_2 的事件串,即不会出现状态 q_5 .

$$D(L_c, y) = \sqrt{(C_{R_1}^{P_y} - C_{R_1}^{T_1} - C_{R_1}^{T_2} - \cdots - C_{R_1}^{T_i})^2 + (C_{R_2}^{P_y} - C_{R_2}^{T_1} - C_{R_2}^{T_2} \cdots - C_{R_2}^{T_i})^2 + \cdots + (C_{R_k}^{P_y} - C_{R_k}^{T_1} - C_{R_k}^{T_2} - \cdots - C_{R_k}^{T_i})^2}.$$

(5)

3 算法实现

输入:任务请求队列 T 、资源池 P ;
输出:资源分配矩阵 A .
执行:
1) 初始化资源分配控制系统起始状态
根据式(1)初始化请求队列 T 、资源包 P ;
确定起始状态 q_0 ;
2) 定义可满足事件集
For 所有资源包 P_y ,
For 任意的事件集
 $L = \{T_1 \cup T_2 \cup \cdots \cup T_m\}$.
If 不等式(2)成立,
 L 属于 P_y 的可满足事件集 L_c ;
Else L 属于 P_y 的不可满足事件集 L_u ;
3) 计算可满足事件串资源与资源包的距离
对任意的资源包 P_y ,
For($y = 1; y < m; y++$),
For($\forall L_i \in L_c$)
根据等式(4),求得 P_y 与 L_i 的距离 D 值.
4) 根据距离 D ,确定分配方案
For 任意的 P_y ,存在 $\min(P_y, L_i)$,
If $T_x \subset L_i$,将 T_x 分配给 P_y ,令 $A[T_x, T_y] = 1, A[T_x, T_y] = 0$;
Else If $\exists L_k \subset L_c, T_x \subset L_k$ 存在且
 $\min(P_j, L_k) \{j \neq y, L_k \cap L_i = \phi\}$,
将 T_x 分配给 P_j ,令
 $A[T_x, T_j] = 1, A[T_x, T_j] = 0$;
5) 计算资源利用率,定义可接受事件
根据等式(3),计算资源利用率 U ,定义可接受事件为满足 $U_{\max}$ 的任务串;
6) 返回分配矩阵 A
控制可接受事件的有效发生,得到最佳分配矩阵 A .

4 仿真实验

本文给出一个云资源分配的子系统,该系统中

确定各个资源包的可满足事件串,若 $T_i, T_j, \cdots, T_l \subset L_c$ 根据式(3)可得 L_c 与资源包 P_y 的参数贴近度,资源参数距离 $D(L_c, y)$ 越小,表明参数贴近度越大.

资源池中有 5 种资源 R_1, R_2, R_3, R_4, R_5 , 分别是 CPU、存储空间、内存、服务器、带宽的容量. 存在 3 个资源包 P_1, P_2, P_3 , 5 个用户请求 T_1, T_2, T_3, T_4, T_5 . 资源包参数分配情况如表 1 所示,用户任务请求资源参数分配情况如表 2 所示.

表 1 资源包参数分配情况

Tab. 1 Parameters allocation of resource packets

资源包	R_1	R_2	R_3	R_4	R_5
P_1	7	10	5	13	15
P_2	2	5	7	8	10
P_3	9	8	10	6	4

表 2 用户请求资源参数分配情况

Tab. 2 Parameters allocation of user requested resource

用户请求	R_1	R_2	R_3	R_4	R_5
T_1	4	2	1	2	2
T_2	1	2	1	4	3
T_3	5	3	4	1	4
T_4	2	3	0	1	5
T_5	1	3	3	0	4

首先,考虑在先来先服务(First Come First Service, FCFS)算法下的云资源分配情况,假设任务请求的顺序是 $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5$, 根据 FCFS 算法, T_1 将最先获得资源分配,根据先后顺序类推, T_5 最后获得资源分配. 当 T_1 到达时,满足各个资源参数条件的资源包有 P_1, P_3 , 将 T_1 分配给 P_1 ; 当 T_2 到达时,将其分配给 P_2 ,同理得出分配矩阵

$$A_{\text{FCFS}} = \begin{matrix} & \begin{matrix} P_1 & P_2 & P_3 \end{matrix} \\ \begin{matrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{matrix},$$

在此示例中,先来先服务算法,虽然可以实现资源分配,但是先来先服务算法存在一个很大的弊端,其没有从全局的角度出发,先到达的任务很有可能占用了一部分资源,使得缩减之后的资源包不能满

足后到达的任务,从而导致一些请求始终得不到资源分配. 接着,考虑本文的基于离散事件系统的云资源分配算法,在 q_0 状态,依次考虑事件集 Σ 中最小子串在各个资源包下是否为可满足事件,考虑资源包 P_1 下,事件 T_1 、 T_2 、 T_3 、 T_4 、 T_5 均为可满足事件;在资源包 P_2 下,可满足事件为 T_2 、 T_5 ;在资源包 P_3 下,可满足事件为 T_1 、 T_2 、 T_3 、 T_5 . 如图2~图4所示.

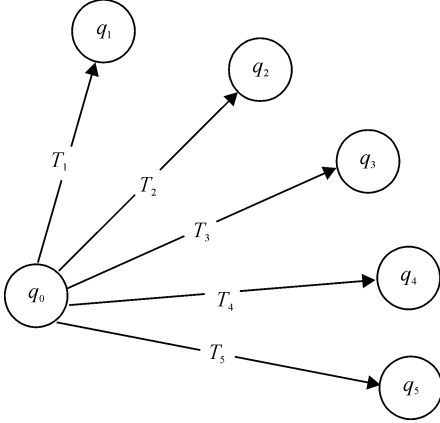


图2 在资源包 P_1 下的可满足最小事件

Fig. 2 Satisfiable minimal event under Resource Packet P_1

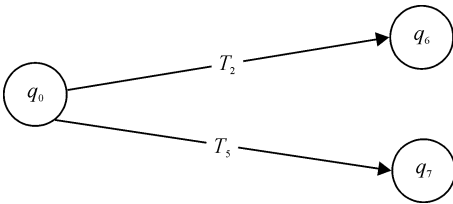


图3 在资源包 P_2 下的可满足最小事件

Fig. 3 Satisfiable minimal event under Resource Packet P_2

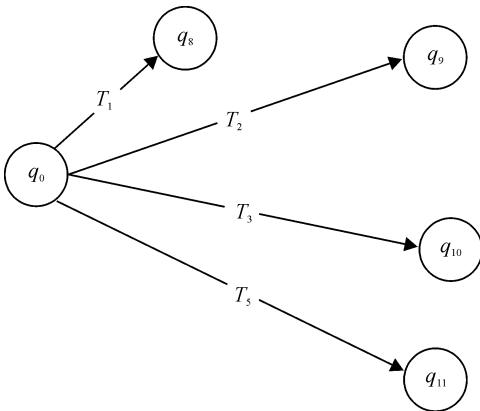


图4 在资源包 P_3 下的可满足最小事件

Fig. 4 Satisfiable minimal event under Resource Packet P_3

依次考虑每个可满足事件的扩充(enlarge)串,找出在各个资源包下分配多个任务的可满足事件

串,若资源包 P_i 不能同时满足 n 个任务串,则不用再考虑其是否满足 $(n+1)$ 个任务串,资源分配情况如图5~图7所示.

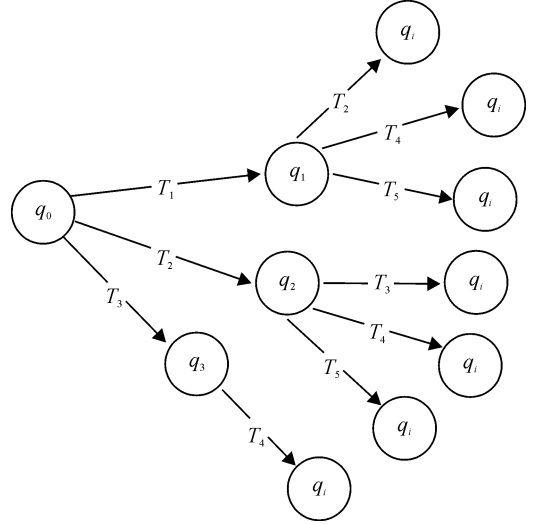


图5 在资源包 P_1 下分配2个任务的可满足事件

Fig. 5 Satisfiable event of allocating two tasks under Resource Packet P_1

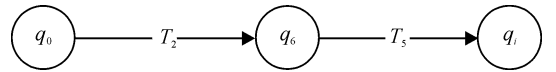


图6 在资源包 P_2 下分配2个任务的可满足事件

Fig. 6 Satisfiable event of allocating two tasks under Resource Packet P_2

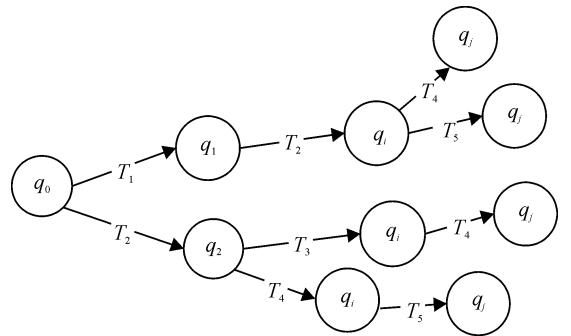


图7 在资源包 P_1 下分配3个任务的可满足事件

Fig. 7 Satisfiable event of allocating three tasks under Resource Packet P_1

在资源包 P_2 下已经不存在可以分配3个任务的可满足事件,则不用再考虑. 本文列出所有可满足事件,并根据式(5)求出距离 D 的值,如表3所示.

根据资源控制分配算法,在所有可满足事件串中,找出符合条件的可接受事件串. 在资源包 P_1 下

表 3 该资源分配系统中所有可满足事件

Tab. 3 All satisfiable events of resource allocation system

Packages	Events	<i>D</i>	Events	<i>D</i>	Events	<i>D</i>	Events	<i>D</i>	Events	<i>D</i>
<i>P</i> ₁	<i>T</i> ₁	19.467 9	<i>T</i> ₂	18.466 2	<i>T</i> ₃	17.860 5	<i>T</i> ₄	18.520 2	<i>T</i> ₅	19.467 9
	<i>T</i> ₁ ∨ <i>T</i> ₂	14.071 2	<i>T</i> ₂ ∨ <i>T</i> ₃	12.409 7	<i>T</i> ₃ ∨ <i>T</i> ₄	13.190 9	<i>T</i> ₄ ∨ <i>T</i> ₅	Unsatisfied		
	<i>T</i> ₁ ∨ <i>T</i> ₃	14.491 3	<i>T</i> ₂ ∨ <i>T</i> ₄	13.038 4	<i>T</i> ₃ ∨ <i>T</i> ₅	14.628 7				
	<i>T</i> ₁ ∨ <i>T</i> ₄	14.352 7	<i>T</i> ₂ ∨ <i>T</i> ₅	14	<i>T</i> ₃ ∨ <i>T</i> ₄ ∨ <i>T</i> ₅	11.445 5				
	<i>T</i> ₁ ∨ <i>T</i> ₅	15.231 5	<i>T</i> ₂ ∨ <i>T</i> ₃ ∨ <i>T</i> ₄	7.937 2						
	<i>T</i> ₁ ∨ <i>T</i> ₂ ∨ <i>T</i> ₃	9.539 3	<i>T</i> ₂ ∨ <i>T</i> ₃ ∨ <i>T</i> ₅	9.643 6						
	<i>T</i> ₁ ∨ <i>T</i> ₂ ∨ <i>T</i> ₄	8.888 1	<i>T</i> ₂ ∨ <i>T</i> ₄ ∨ <i>T</i> ₅	9.327 3						
	<i>T</i> ₁ ∨ <i>T</i> ₂ ∨ <i>T</i> ₅	9.746 7								
	<i>T</i> ₁ ∨ <i>T</i> ₃ ∨ <i>T</i> ₄	10.816 6								
	<i>T</i> ₁ ∨ <i>T</i> ₃ ∨ <i>T</i> ₅	12.124 3								
	<i>T</i> ₁ ∨ <i>T</i> ₄ ∨ <i>T</i> ₅	11								
<i>P</i> ₂	<i>T</i> ₁	12.206 5	<i>T</i> ₂	10.535 7	<i>T</i> ₃	10.344 0	<i>T</i> ₄	11.269 4	<i>T</i> ₅	11
			<i>T</i> ₂ ∨ <i>T</i> ₅	5.830 95						
<i>P</i> ₃	<i>T</i> ₁	12.727 9	<i>T</i> ₂	13.638 2	<i>T</i> ₃	10.099 5	<i>T</i> ₄	14.142 1	<i>T</i> ₅	13.190 9
	<i>T</i> ₁ ∨ <i>T</i> ₂	9.848 8	<i>T</i> ₂ ∨ <i>T</i> ₃	Unsatisfied	<i>T</i> ₃ ∨ <i>T</i> ₅	Unsatisfied				
	<i>T</i> ₁ ∨ <i>T</i> ₃	6.855 6	<i>T</i> ₂ ∨ <i>T</i> ₅	Unsatisfied						
	<i>T</i> ₁ ∨ <i>T</i> ₅	Unsatisfied	<i>T</i> ₂ ∨ <i>T</i> ₃ ∨ <i>T</i> ₅	Unsatisfied						
	<i>T</i> ₁ ∨ <i>T</i> ₂ ∨ <i>T</i> ₅	Unsatisfied								
	<i>T</i> ₁ ∨ <i>T</i> ₃ ∨ <i>T</i> ₅	Unsatisfied								

存在最小 *D* 值的可满足串是 *T*₁*T*₂*T*₄,假设将 *P*₁ 分配给 *T*₁*T*₂*T*₄,在 *P*₂ 中将不再考虑 *T*₁*T*₂*T*₄,此时存在最小 *D* 值的可满足事件串为 *T*₅,在 *P*₃ 不考虑任务 *T*₁、*T*₂、*T*₄、*T*₅,存在最小 *D* 值的可满足串为 *T*₃,假定方案一为将 *P*₂ 分配给 *T*₅,*P*₃ 分配给 *T*₃.同理,考虑资源包 *P*₂、*P*₃、*P*₄、*P*₅ 的可满足串,最终可得出两个分配矩阵

*P*₁

*P*₂

*P*₃

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

和

*P*₁

*P*₂

*P*₃

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\boldsymbol{A}_1 =$$

根据式(4)求出资源利用率 $U_{A_1} = 1.463\ 2$, $U_{A_2} = 1.317\ 2$,由于 $U_{A_1} > U_{A_2}$,分配矩阵 *A*₁ 的资源利用率较大,取 *A*₁ 为最优分配.定义 *P*₁ 下的事件串 *T*₁*T*₂*T*₄ 为可接受事件,同理,*P*₂ 下的 *T*₅ 以及 *P*₃ 下的 *T*₃ 也定义为可接受事件,控制可接受事件的有效发生,能实现资源的最优分配.

本文基于离散事件系统的云资源分配控制算法,从全局的角度定义出所有资源分配的可满足事件和可接受事件,通过对可满足事件的梳理以及控制可接受事件串的有效发生,不仅有效避开了 FCFS

算法陷入僵局的弊端,也保证了资源的利用率,是一个可行且有效的算法.

5 结语

本文提出基于 DES 的云资源分配优化控制策略,通过将资源参数化,形式化描述资源的优化分配过程,有效将离散事件系统控制理论应用到云资源分配模型中,保证提高资源利用率的事件有效发生,满足资源利用率最大的同时实现最快最有效的分配.

参考文献:

[1] MELL P, GRANCE T. The NIST definition of cloud computing 800-145[S]. Gaithersburg, MD: National Institute of Science and Technology,2010,53(6):50-50.

[2] 刘鹏. 云计算[M]. 北京:电子工业出版社,2010.

[3] 王笑宇,程良伦. 云计算下的多源信息资源云体系及服务模型研究[J]. 计算机应用研究,2014,31(3):784-788.

WANG X Y, CHENG L L. Study of multi-source information resources cloud systems and cloud services model on cloud computing[J]. Application Research of Computers, 2014,31(3):784-788.

[4] XIONG K Q, HE Y X. Power-efficient Resource Allocation

- in MapReduce Clusters[C]//Integrated Network Management(IM 2013),IFIP/IEEE International Symposium. Ghent;IEEE,2013:603-608.
- [5] CHEN Y M, TSAI Shin-Ying. Optimal provisioning of resource in a cloud service[J]. IJCSI International Journal of Computer Science Issue,2010,7(6):95-99.
- [6] CALHEIROS R N, RANJAN R, BUYYA R. Virtual machine provisioning based on analytical performance and QoS in cloud computing environments[C]//Parallel Processing (ICPP), International Conference. Taipei City: IEEE, 2011:295-304.
- [7] 华夏渝,郑骏,胡文心. 基于云计算环境的蚁群优化计算资源分配算法[J]. 华东师范大学学报(自然科学版), 2010,1(1):127-134.
- HUA X Y, ZHENG J, Hu W X. Ant colony optimization algorithm for computing resource allocation based on cloud computing environment[J]. Journal of East China Normal University:Natural Science Edition,2010,1(1):127-134.
- [8] 王静宇,谭跃生. 云计算环境下资源分配与任务调度研究[J]. 中南大学学报,2010,41(S):47-49.
- WANG J Y, TAN Y S. Research of resource allocation and job scheduling in cloud computing[J]. Journal of Central South University,2010,41(S):47-49.
- [9] 刘东,常静,魏文红,等. 基于MPI的并行蚁群算法的研究与实现[J]. 广东工业大学学报,2008,25(1):38-42.
- LIU D,CHANG J,WEI W H, et al. Research and Implementation of parallel ant colony optimization algorithm based on MPI[J] Journal of Guangdong University of Technology,2008,25(1):38-42.
- [10] SCHMIDT K, BREINDL C. Maximally permissive hierarchical control of decentralized discrete event systems[J]. Automatic Control, IEEE Transactions,2010,56(4):723-737.
- [11] 张浩荣. 云环境下基于蚁群算法的资源调度策略研究[D]. 广州:广东工业大学计算机学院,2014.
- [12] 姜晓涛. 基于模拟退火的蚁群算法求解网格任务调度问题[D]. 合肥:安徽大学计算机科学与技术学院, 2012.
- [13] NURMI D, WOLSKI R. The eucalyptus open-source cloud-computing system[C]//Cluster Computing and the Grid,9th IEEE/ACM International Symposium. Shanghai: IEEE2009:124-131.
- [14] 张浩荣,陈平华,熊建斌. 基于蚁群模拟退火算法的云环境任务调度[J]. 广东工业大学学报,2014,31(3):77-82.
- ZHANG H R, CHEN P H, XIONG J B. Task scheduling algorithm based on simulated annealing ant colony algorithm in cloud computing environment[J]. Journal of Guangdong University of Technology,2014,31(3):77-82.
- [15] 郑湃,崔立真,王海洋,等. 云计算环境下面向数据密集型应用的数据布局策略与方法[J]. 计算机学报, 2010,33(8):1472-1480.
- ZHENG P, CUI L Z, WANG H Y, et al. A data placement strategy for data intensive applications in cloud[J]. Chinese Journal of Computers, 2010,33(8):1472-1480.
- [16] SAN M. Cloud computing gives emerging markets a lift[J]. IT Professional,2011,13(6):60-62.
- [17] 唐卓,朱敏,杨黎. 云环境中面向随机任务的用户效用优化模型[J]. 计算机研究与发展,2014,51(5):1120-1128.
- TANG Z, ZHU M, YANG L. Random task-oriented user utility optimization model in the cloud environment[J]. Journal of Computer Research and Development,2014,51(5):1120-1128.