DOI：10.7652/xjtuxb201802003

异构系统中带可用性约束的性能优化调度算法

孙健，张兴军，董小社
（西安交通大学电子与信息工程学院，710049，西安）

摘要：针对可用性约束以及任务响应时间不同给异构系统中实时任务调度分配带来的处理机节点难选取的问题，提出了一种带可用性约束的性能优化调度算法(PO-SSAC)。该算法将异构系统内处理机节点、实时任务以及带可用性约束的实时任务调度过程进行数学建模，通过考虑可用性成本与任务平均响应时间的折中，引入系统综合开销的概念，根据实时任务的可用性需求为其分配系统综合开销最少的处理机节点调度执行，达到系统调度资源合理利用的目的。实验结果表明，在实时任务调度可用性方面，与SSAC算法相比，PO-SSAC算法提升了3.4%，与MinMin算法相比，提升了76.9%，与Sufferage算法相比，提升了76.5%；与Sufferage算法相比，PO-SSAC算法的系统综合开销减少了约30%，优化了异构系统的实时任务调度性能。

关键词：异构系统；可用性约束；实时任务调度
中图分类号：TP302.7 **文献标志码**：A **文章编号**：0253-987X(2018)02-0018-06

A Performance Optimization Scheduling Strategy with Availability Constraints for Heterogeneous Systems

SUN Jian, ZHANG Xingjun, DONG Xiaoshe
(School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: A performance optimization scheduling strategy with availability constraints (PO-SSAC) for heterogeneous systems is proposed to address the processor choosing difficulty brought by availability constraints and different task response times during real-time tasks scheduling in heterogeneous systems. The strategy models processors, real-time tasks as well as the process of real-time task scheduling with availability constraints of a heterogeneous system, and a concept of system comprehensive cost is introduced by considering the compromise of availability costs and average response times of tasks. Then tasks are allocated to processors with the least system comprehensive cost according to specific availability request of real-time tasks, which reaches the goal of reasonable utilization of scheduling resources within the heterogeneous system. Experimental results show that the real-time tasks scheduling availability of the proposed PO-SSAC is improved by 3.4%, 76.9% and 76.5% compared with the SSAC algorithm, MinMin and Sufferage algorithms, respectively, and that the proposed strategy reduces the system comprehensive cost by nearly 30% compared with the Sufferage. These results show that the proposed strategy optimizes the performance of real-time tasks scheduling in the heterogeneous

收稿日期：2017-06-17。 作者简介：孙健(1983—)，男，博士生；董小社(通信作者)，男，教授，博士生导师。 基金项目：国家重点研究发展计划资助项目(2016YFB1000303)；国家自然科学基金资助项目(61202041)；国家高技术研究发展计划资助项目(2008AA01A202)。
网络出版时间：2017-11-28 网络出版地址：http://kns.cnki.net/kcms/detail/61.1069.T.20171128.1540.006.html

systems.

Keywords: heterogeneous system; availability constraint; real-time tasks scheduling

可用性是衡量复杂计算机系统的关键指标之一，特别是近年来异构系统的日益发展，伴随系统规模以及实时应用范围的逐步扩大，研究异构系统中多实时任务的可用性需求问题具有非常重要的理论与实际意义。该研究领域内涌现出诸多以满足实时任务的具体可用性需求，并通过带可用性约束的调度策略来实现系统内实时任务合理调度分配的实时任务调度算法[1-2]。

早期实时任务调度理论中对任务调度算法的最基本假设是系统内所有用于任务分配执行的处理机节点均可用[3]。该假设仅在理想状态的多处理机系统实时任务调度中是合理的，但在复杂计算机系统特别是异构系统中，由于系统故障宕机、失效修复等情况时有发生常导致系统不可用，此时该假设便不再适用。另外，异构系统内各处理机节点的实时任务执行时间即任务响应时间也各不相同，为选取系统内任务分配的处理机节点增添了难度。为此，在设计实时任务调度算法时应充分考虑各实时任务的具体可用性需求即可用性约束，并在此基础上权衡系统调度分配过程中的任务响应时间，保证在任务可调度性以及可用性的前提下实现异构系统中实时任务的合理调度。

针对上述问题，本文提出一种异构系统中带可用性约束的性能优化调度算法，构建系统内处理机节点、实时任务以及带可用性约束的实时任务调度模型，通过引入可用性成本和系统综合开销，考虑可用性成本与任务平均响应时间的折中，为实时任务分配系统综合开销最少的处理机节点，合理利用异构系统调度资源，在提升实时任务调度可用性的同时进一步优化异构系统的调度性能。

1　相关工作

目前，异构系统中可用性相关的实时任务调度算法大致可分为两类：一类是可用性受限的实时任务调度算法，即在满足实时任务可用性约束的前提下，对以往仅考虑任务执行时间的调度算法进行优化改进，典型算法如 QoS-SAC[4]、SSAC[5] 等；另一类是实时任务本身无可用性约束，而在算法设计中考虑可用性因素调度策略，旨在提升系统整体以及实时任务分配执行的可用性水平和性能，如 HM-SAS[6]、ADSS[7] 等。

多处理机异构环境下的实时任务最优分配一直是一个难以解决的 NP 完全问题[8]。考虑可用性受限的实时任务调度算法，Xie 等首次提出了异构系统中带可用性约束的多任务组调度策略(SSAC)[9]，该策略的核心思想是结合任务平均响应时间和可用性约束对实时任务进行数学建模，根据任务的具体可用性约束，为其选择系统内满足该约束的候选处理机集合，并将任务分配至集合中任务平均响应时间最短的处理机处理执行，在提升调度性能的同时提高系统可用性。此外，Tong 等考虑系统 QoS 以及可用性需求，在文献[5,9]的基础上，针对异构分布式系统设计可用性受限的实时任务调度算法(QoS-SAC)[4]，以缩短实时任务的完成时间同时提高系统可用性。

本文提出的异构系统中带可用性约束的性能优化调度算法(PO-SSAC)，在实时任务调度执行时考虑任务平均响应时间与可用性成本的折中，并为其分配系统综合开销最少的处理机节点，与 SSAC 以及其他现有算法相比，进一步提升了异构系统的实时任务调度可用性以及性能，实现了系统资源的合理利用。

2　实时任务调度模型

2.1　带可用性约束的实时任务调度框架结构

本文所需解决的具体问题是如何在可用性受限的情况下对异构系统中实时任务进行合理调度分配，并在提升系统任务调度可用性的基础上进一步优化调度性能。根据上述问题描述，异构系统中带可用性约束的实时任务调度框架结构如图 1 所示。

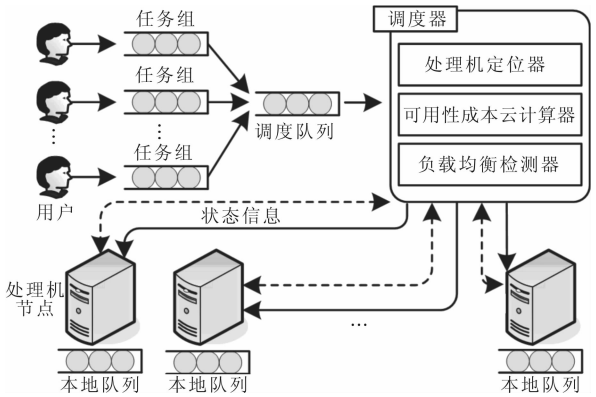


图 1　带可用性约束的实时任务调度框架结构

调度器是实时任务调度框架结构的核心，负责调度分配来自各不同用户的任务组队列，并实时监

测异构系统内所有处理机节点的运行状态，及时获取各处理机节点的运行状态信息。调度队列负责接收来自各用户的任务组，调度器按照先来先服务原则接收所有到达的实时任务，分配其至各处理机节点，并由处理机节点内本地队列实现对各实时任务的并行处理。

对于每个来自用户的实时任务，处理机定位器为其筛选一个候选处理机集合，集合包含所有满足该任务可用性需求即可用性约束的处理机节点。若集合非空，调度器从集合内选择系统综合开销最少的处理机节点，并分配任务至该处理机节点处理执行；若集合为空，说明没有候选处理机节点满足当前任务的可用性约束，此时可用性成本计算器计算系统内能够满足该任务可用性约束的处理机节点，并将其添加至候选处理机集合。负载均衡检测器检测候选处理机节点是否超载，超载时实时任务将被分配至负载最轻的处理机节点，否则分配至该处理机节点处理执行。

2.2 处理机和实时任务模型

对异构系统内处理机和实时任务进行如下的形式化数学描述。①处理机模型。异构系统指由一定数量内部构造不同且相互独立的自治节点，通过高速互联网络相互连接所共同组成的高性能、高可用计算机系统，能够作为一个整体为用户提供所需的应用服务。设异构系统处理机集合 $H=\{N_1,N_2,\cdots,N_j,\cdots,N_n,j=1\sim n\}$，$n$ 为处理机节点数。H 中各处理机节点处理能力由每秒百万条指令(MIPS)来衡量，同时由于是异构系统，假设各处理机节点处理能力和可用性均不相同。②实时任务模型。设实时任务集合 $T_G=\{t_1,t_2,\cdots,t_i,\cdots,t_m,i=1\sim m\}$，$m$ 为来自用户的实时任务数。系统根据用户的可用性需求为实时任务设定可用性约束，范围为 0～1，实时任务必须分配至能够满足其可用性约束的处理机节点以确保得到成功处理。

定义实时任务 t_i 在处理机节点 N_j 上执行的平均响应时间

$$T_{ij}=E(s_j)+(\Lambda_j E(s_j^2))/(2(1-\phi_j)) \quad (1)$$

式中：s_j 为实时任务集合 T_G 在处理机节点 N_j 上的服务时间；$s_j{}^2$ 为 s_j 的二阶矩；$E(s_j)$ 为服务时间均值；$E(s_j{}^2)$ 为服务时间均方。处理机节点 N_j 的任务到达率为 Λ_j，服务率为 ϕ_j，假设异构系统内任务到达模式和服务率是先验的，则 Λ_j 与 ϕ_j 均可通过代码剖析和统计预测方法估算得出。

2.3 带可用性约束的实时任务调度模型

本文异构系统中可用性均为稳态可用性，是指在某个时间段内系统维持正常运行状态的概率。定义异构系统内处理机节点 N_j 的可用性为 ξ_j，表示任意时间段内处理机节点 N_j 可提供持续运算处理的概率，相应的异构系统内处理机节点 N_j 的不可用性为 θ_j。定义实时任务 t_i 的可用性约束为 a_i，代表任务 t_i 必须执行成功的概率，例如，当 $a_i=0.85$ 时，任务 t_i 执行失败的概率不得高于 0.15。

进一步定义 a_{ij} 为实时任务 t_i 在处理机节点 N_j 上的可用性成本

$$a_{ij}=p_{ij}\theta_j/\mu_{ij} \quad (2)$$

式中：p_{ij} 为实时任务 t_i 分配至处理机节点 N_j 的概率；μ_{ij} 为实时任务 t_i 分配至处理机节点 N_j 的服务率。

本文引入综合性能开销的概念，定义 C_{ij} 为实时任务 t_i 在处理机节点 N_j 上的系统综合开销，结合式(1)和式(2)，得到 C_{ij} 的计算公式为

$$C_{ij}=a_{ij}T_{ij}=p_{ij}\frac{\theta_j}{\mu_{ij}}\left[E(s_j)+\frac{\Lambda_j E(s_j^2)}{2(1-\phi_j)}\right] \quad (3)$$

结合 2.2 小节对处理机和实时任务模型的相关分析，将带可用性约束的性能优化调度问题抽象为异构系统可用性与实时任务平均响应时间两者之间的折中，则本文调度算法的设计目标可描述为：提高实时任务调度分配的可用性；合理利用资源减少系统综合开销，即保证较短的实时任务平均响应时间和较低的处理执行可用性成本，得到如下数学模型

$$\max A=\sum_{i=1}^{m}\left[\frac{\lambda_i}{m\lambda}\exp\left(-\sum_{j=1}^{n}p_{ij}\frac{\theta_j}{\mu_{ij}}\right)\right] \quad (4)$$

系统综合开销约束条件为

$$\forall\, 1\leqslant i\leqslant m,1\leqslant j\leqslant n,a_i\leqslant\xi_j:\min C_{ij}$$

在该数学模型中，A 为异构系统实时任务调度的系统可用性；λ_i 为实时任务 t_i 的到达率且符合泊松过程；λ 为实时任务集合 T_G 在异构系统中的总平均到达率。

3 带可用性约束的性能优化调度算法

实时任务平均响应时间在很大程度上依赖于异构系统中各处理机节点所采用的具体排序策略，因此本文所提 PO-SSAC 算法采用文献[10]中给出的最优排序策略以最大程度减少异构系统中实时任务集合的平均响应时间，并作如下命题。

命题 1 已知 m 组实时任务队列和由 n 个处理机节点构成的异构系统，根据最优排序策略，在处理机节

点 N_j 上，当 $\mu_{ij} \geqslant \mu_{kj}$ 时，实时任务 t_i 的优先级高于 t_k。

上述命题说明高服务率的实时任务在调度分配时必须拥有相对较高的优先级，以达到缩短任务平均响应时间的目的。为简化描述进一步作如下假设。

假设 1　假设异构系统内实时任务集合 T_G 按照服务率从高到低排序，即

$$\rho_1 \geqslant \rho_2 \geqslant \cdots \geqslant \rho_i \cdots \geqslant \rho_m$$

式中：ρ_i 为实时任务 t_i 分配至 H 的总服务率。根据该假设，调度策略可以在执行最初对所有实时任务按照服务率从高到低顺序进行重新排列，以方便之后的调度分配。

在 PO-SSAC 算法中，执行程序首先按照命题 1 和假设 1 为高服务率的实时任务分配高优先级并按照优先级进行排序，之后进入对实时任务集合的循环处理。循环处理是 PO-SSAC 算法的核心部分，首先判断候选处理机子集是否为空，若子集非空，说明子集内至少包含 1 个处理机节点能够满足实时任务 t_i 的可用性约束，进而循环计算候选处理机子集中各候选处理机节点的系统综合开销 C_{ij}，选取子集中系统综合开销最少的处理机节点，并将该处理机节点选为任务调度的执行节点。

讨论调度的特殊情况，若候选处理机子集为空，说明此时异构系统中所有处理机节点均无法满足实时任务 t_i 的可用性约束。在该情况下 PO-SSAC 算法将采取相应的措施尽可能对 t_i 的调度执行可用性进行提升。循环计算 t_i 在异构系统内处理机节点的可用性成本 a_{ij}，并为 t_i 分配系统内可用性成本最低的处理机节点。另外，如果系统内存在 2 个或 2 个以上可用性成本取值相同且最低的处理机节点，则 t_i 将分配至其中平均响应时间相对较短的处理机节点上。

当某候选处理机节点 N_s 选定以后，PO-SSAC 主函数将调用 loadBalance()函数进行负载均衡检测并最终分配实时任务 t_i 至相应的处理机节点处理执行。loadBalance()函数首先估算 H 内所有处理机节点的负载指标并找到其中负载最轻的处理机节点 $N_{m\min}$，令该节点负载指标为 $L_{\min}$。对于所选取的处理机节点 N_s，如果 N_s 没有超载，系统将分配 t_i 至该节点处理执行；如果 N_s 超载，系统将分配 t_i 至 H 内负载最轻的处理机节点 $N_{m\min}$ 处理执行，负载阈值为 L_T。

对于 PO-SSAC 主函数，T_G 按照服务率从高到低排序的时间复杂度为 $O(m\mathrm{lb}m)$，处理机节点选取的时间复杂度为 $O(n)$，则对于实时任务集合 T_G，PO-SSAC 主函数在最差执行情况下的时间复杂度为 $O(mn)$。进一步分析 loadBalance()函数，循环计算异构系统内所有处理机节点负载指标的时间复杂度为 $O(n)$，假设单次任务调度分配中，PO-SSAC 主函数和 loadBalance()函数中其他步骤执行时间复杂度为 $O(1)$，则对于实时任务集合 T_G，PO-SSAC 算法任务分配的时间复杂度为 $O(m(n+1))$。综合上述分析，PO-SSAC 算法在最差执行情况下的整体时间复杂度为 $O(m\mathrm{lb}m)+O(mn)+O(m(n+1))\approx O(2mn)$。

图 2 给出了 PO-SSAC 调度策略的一个抽象实例，假设异构系统由 8 个处理机节点（$N_1 \sim N_8$）组成，其中各处理机节点可用性取值范围为 0.68～0.97，任务平均响应时间取值范围为 42～204 s，可用性成本取值范围为 0.031～0.297。假设实时任务可用性约束为 0.80，则符合该可用性约束的候选处理机集合为 N_3、N_4、N_6、N_7 和 N_8，各候选处理机的系统综合开销取值范围为 2.94～21.42。按照 PO-SSAC 调度策略，调度器将最终分配实时任务至系统综合开销最少的处理机节点 N_3 处理执行。

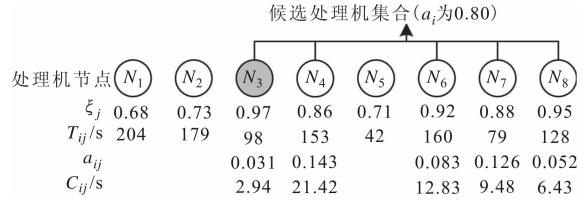


图 2　PO-SSAC 调度策略实例

4　实验分析

本文面向异构系统设计了带可用性约束的性能优化调度算法 PO-SSAC，并通过随机实时任务集在异构系统内的调度分配对现有实时任务调度算法 SSAC、MinMin、Sufferage 以及 PO-SSAC 进行对比仿真实验。异构系统选用 GridSim 仿真工具进行构建，代码用 Java 通过 eclipse 编译实现，仿真环境为 Red Hat Enterprise Linux 7.0 操作系统，CPU 为 Inter Core i7-6700k@4.00 GHz 四核，内存为 16 GB，硬盘为希捷 ST3000DM001-1ER166(3 TB)。

4.1　实验用例

实验主要从任务总平均到达率和处理机节点数的变化情况对异构系统内 PO-SSAC 算法以及对比算法的调度执行情况进行分析。实验运行 200 次并记录实验结果，去掉其中 5 次最大和最小结果，取剩余结果均值作为最终实验结果。实验参数和取值情况见表 1，参数或者参照文献[5,9]中实验部分的参

数取值,或者取自异构系统的实际评测经验值。

表 1 实验参数和取值

系统参数	固定值	取值
n	16	16,32,64,128
λ	1.0	0.2,0.4,0.6,0.8,1.0
T_{ij}/s		1～500
ξ_j		0.1～1.0
a_i		0.1～1.0
L_T	1.25	

异构系统处理机节点数为 n,实时任务集合 T_G 通过 GridSim 仿真工具随机生成,任务平均响应时间 T_{ij} 为 1～500 s 内随机整数。根据对实时任务调度模型的分析可知,任务总平均到达率 λ 服从泊松分布,任务平均响应时间 T_{ij}、处理机节点可用性 ξ_j 以及任务可用性约束 a_i 均服从均匀分布,负载阈值 L_T 取固定经验值。

实验选取与 PO-SSAC 算法相近似的另外 3 个调度算法进行对比,包括 SSAC、MinMin[11] 以及 Sufferage 算法[12],上述算法均适用于异构系统中的实时任务调度分配,同时也适用于分布式或同构系统的任务调度情况。

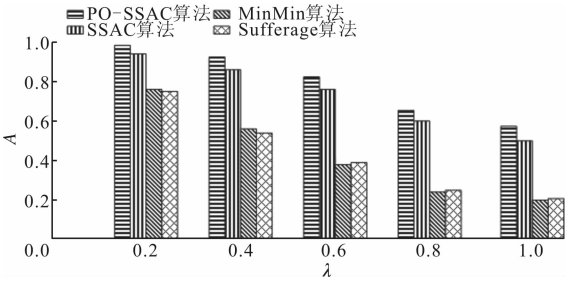
实验主要评价指标包括异构系统实时任务调度系统可用性、系统总平均响应时间以及系统综合开销。

4.2 实验结果分析

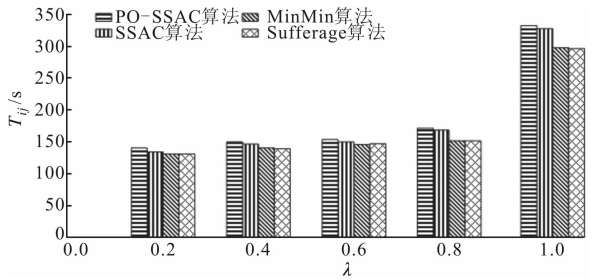
首先分析任务总平均到达率 λ 变化情况下异构系统实时任务调度的各评价指标。λ 取值范围为 0.2～1.0,增量为 0.2,处理机节点数 $n=16$。实时任务调度分配实验结果如图 3 所示。

由图 3a 可以看出,采用可用性约束调度策略的 PO-SSAC 与 SSAC 可用性提升明显,与 MinMin 算法相比,可用性提升约 76.9%,与 Sufferage 算法相比,可用性提升约 76.5%,同时 PO-SSAC 更优于 SSAC,可用性提升约 3.4%,原因在于 PO-SSAC 在实时任务处理机节点选取时考虑了实时任务可用性成本与平均响应时间的折中,为其分配系统综合开销最少的处理机节点调度执行,该策略使实时任务调度分配的可用性能够得到进一步的提升。图 3b、图 3c 中实验结果表明,PO-SSAC 在获取较高系统可用性的同时增加了异构系统内实时任务的调度执行时间,有效降低了系统综合开销,与系统综合开销最多的 Sufferage 算法相比,减少近 30%,优化了异构系统的性能。

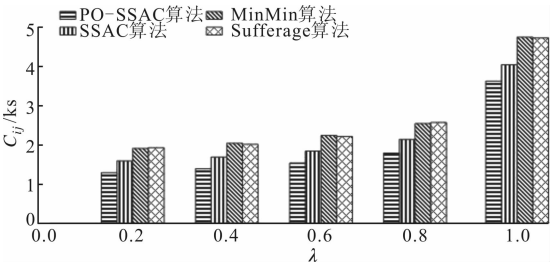
进一步分析处理机节点数 n 变化时异构系统实



(a)系统可用性



(b)系统平均响应时间

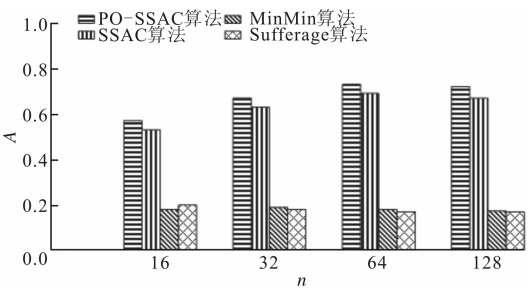


(c)系统综合性能开销

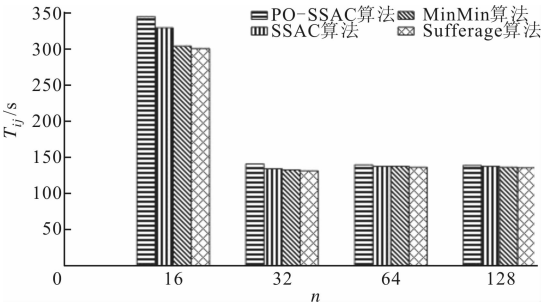
图 3 任务总平均到达率不同时系统的各项性能实验结果

时任务调度的各评价指标。n 取值为 16、32、64、128 时,任务总平均到达率 $\lambda=1$。实时任务调度分配实验结果如图 4 所示。

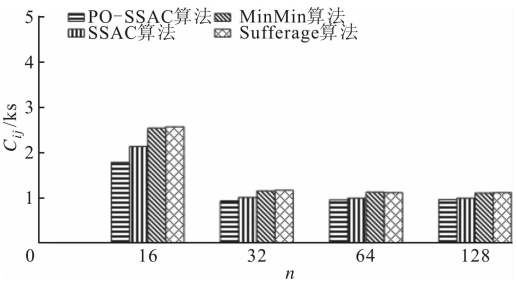
与任务到达率变化实验结果类似,与其他 3 个算法相比,PO-SSAC 系统可用性有所提升,但由于考虑可用性约束,增加了异构系统的实时任务调度执行时间。由图 4c 中实验结果可以看出,与其他 3 个算法相比,PO-SSAC 系统综合开销最少。另外,本实验结果也说明当异构系统规模扩大,即处理机



(a)系统可用性



(b)系统平均响应时间



(c)系统综合开销

图 4　处理机节点数不同时系统的各项性能实验结果

节点数增多时，系统综合开销会减少，实时任务调度可用性和系统性能将得到提升。

5　结　论

本文提出了一种异构系统中带可用性约束的性能优化调度算法，该算法对异构系统内处理机节点、实时任务以及带可用性约束的实时任务调度进行数学建模，引入系统综合开销概念并考虑可用性成本与任务平均响应时间的折中，将实时任务分配给系统综合开销最少的处理机节点，以达到系统调度资源合理利用的目的。实验结果表明，与现有算法相比，PO-SSAC 算法提升了异构系统实时任务调度的系统可用性，系统调度性能也得到了进一步优化。

参考文献：

[1] FAN J, LU X W, LIU P H. Integrated scheduling of production and delivery on a single machine with availability constraint [J]. Theoretical Computer Science, 2015, 562: 581-589.

[2] BELMABROUK M, MARRAKCHI M. Optimal parallel scheduling for resolution a triangular system with availability constraints [C]// Proceedings of the International Conference on Computer Systems and Applications. Piscataway, NJ, USA: IEEE, 2015: 1-7.

[3] SALFNER F, WOLTER K. A Petri net model for service availability in redundant computing systems [C]// Proceedings of the 2009 Winter Simulation Conference. Piscataway, NJ, USA: IEEE, 2009: 819-826.

[4] TONG Z, LI K L, XIAO Z, et al. A Qos scheduling scheme with availability constraint in distributed systems [C]// Proceedings of the 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. Piscataway, NJ, USA: IEEE, 2012: 481-486.

[5] QIN X, XIE T. An availability-aware task scheduling for heterogeneous systems [J]. IEEE Transactions on Computers, 2008, 57(2): 188-199.

[6] KHOUDI A, BERRICHI A, YALAOUI F. Heuristics to maximize system availability on parallel machine scheduling problem [C]// Proceedings of the International Symposium on Programming and Systems. Piscataway, NJ, USA: IEEE, 2015: 1-6.

[7] ZHU M, GUO W, XIAO S L, et al. Availability-driven scheduling for real-time directed acyclic graph applications in optical grids [J]. Journal of Optical Communications and Networking, 2010, 2(7): 469-480.

[8] 李智勇，陈少淼，杨波，等. 异构云环境多目标 Memetic 优化任务调度方法 [J]. 计算机学报，2016，39(2)：377-390.
LI Zhiyong, CHEN Shaomiao, YANG Bo, et al. Multi-object memetic algorithm for task scheduling on heterogeneous cloud [J]. Chinese Journal of Computers, 2016, 39(2): 377-390.

[9] XIE T, QIN X. Stochastic scheduling with availability constraints in heterogeneous clusters [C]// Proceedings of the International Conference on Cluster Computing. Piscataway, NJ, USA: IEEE, 2006: 1-10.

[10] SETHURAMAN J, SQUILLANTE M S. Optimal stochastic scheduling in multiclass parallel queues [J]. ACM Sigmetrics Performance Evaluation Review, 1999, 27(1): 93-102.

[11] TAN M, SIEGEL H J, ANTONIO J K, et al. Minimizing the application execution time through scheduling of subtasks and communication traffic in a heterogeneous computing system [J]. IEEE Transactions on Parallel and Distributed Systems, 1997, 8(8): 857-871.

[12] SONG S S, HWANG K, KWOK Y K. Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling [J]. IEEE Transactions on Computers, 2006, 55(6): 703-719.

(编辑　武红江)