面向容错的对地观测卫星实时任务调度研究

朱晓敏,王建江,王 吉,刘 忠
(国防科学技术大学信息系统与管理学院,湖南长沙 410073)

摘 要: 提出一种面向容错的对地观测卫星任务调度模型,该模型采用主版本/副版本技术可以实现对任意时刻一颗卫星失效时的容错.在容错调度模型的基础上,提出了一种卫星容错调度算法 FTSS.FTSS 采用重叠技术,有效提高了卫星资源利用率.此外,FTSS 采用了任务合成策略可以有效减少实际执行任务的个数从而进一步提高系统的可调度性.为了验证 FTSS 算法的性能,本文通过模拟实验对 FTSS 与其它 3 个基准算法进行了比较分析.实验结果表明 FTSS 优于其它算法,适合卫星实时任务容错调度.

关键词: 对地观测卫星; 容错; 调度; 主/副版本; 重叠; 合成
中图分类号: 0436 **文献标识码:** A **文章编号:** 0372-2112 (2015)08-1471-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.08.002

Fault-Tolerance Oriented Real-Time Task Scheduling on Multiple Earth-Observing Satellites

ZHU Xiao-min, WANG Jian-jiang, WANG Ji, LIU Zhong
(*College of Information Systems and Management, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: This paper presents a novel dynamic fault-tolerant scheduling model for real-time tasks on multiple observation satellites. In this model, the primary/backup policy is employed to tolerate one satellite's permanent failure at one time instant. On the basis of the fault-tolerant model, we propose a novel fault-tolerant satellite scheduling algorithm called FTSS. To improve the resource utilization, the overlapping technology is adopted. According to the satellite feature with time windows, we analyze the overlapping mechanism on satellites and combine them into FTSS. In addition, the FTSS employs the task merging strategies that are used to decrease the task count needed to execute and thus further enhance the schedulability. To demonstrate the superiority of our FTSS, we conduct simulation experiments to compare FTSS with three baseline algorithms. The experimental results indicate that FTSS efficiently improves the scheduling quality of others and is suitable for satellite fault-tolerant scheduling.

Key words: earth-observation; satellite; fault-tolerance; scheduling; primary copy/backup copy; overlapping; merging

1 引言

对地观测卫星由于其具有覆盖范围广,连续监视时间长和不受空域国界限制等优势,已被广泛应用于地球资源勘测,战场态势侦察,自然灾害监视等领域[1].随着对地观测卫星数量和种类的不断增加以及观测空间分辨率的不断提升,对地观测卫星将成为应急条件下获取地面实时信息的重要手段[2].由于应急环境具有事件突发性、时间紧迫性、不确定性等特点,事件发生具有突然性,时间、地点、规模具有不确定性,为及时开展应急救援行动或军事行动,需要在几小时、甚至几十分钟内提供实时服务[3].

特别需要注意的是,应急条件下的实时任务,往往需要系统具有极高的可靠性,例如在作战过程中,卫星可能由于被攻击或失去控制导致已安排在其上的观测任务不能完成,从而无法在规定的时间内获取对地观测信息,影响作战决策和作战部署.因此,如何保障卫星在失效的情况下仍能满足任务的时效性需求是目前极为重要的研究课题.值得注意的是,随着卫星应用的发展,目前可以采用中继卫星等手段保持对地观测卫星与地面站的不间断连接,为实时任务的容错处理提供了时效保障.

众所周知,调度是提高卫星观测系统性能的有效手段[4,5].但是从目前的研究成果来看,卫星调度主要侧重于研究如何最大化任务的完成率,而忽略了对系统容错性的考虑,显然不适用于具有高可靠性要求的实时任

收稿日期:2014-04-02;修回日期:2014-09-18;责任编辑:蓝红杰
基金项目:国家自然科学基金(No.61104180,No.71271216);教育部创新团队发展计划(No.IRT13014)

务.与此同时,很多容错调度算法已在其它分布式系统,如集群系统、网格系统中(如文献[6,7])进行了较多研究.但是,据我们所知,目前还没有关于多星容错调度方面的研究.在实时容错调度中必须充分考虑卫星的时间窗口限制和任务的截止期限制,这极大增加了容错调度问题的复杂性.为了弥补在此方面的研究空白,本文设计并实现了一种面向容错的对地观测卫星实时任务调度方法.具体来讲,该方法针对非周期、独立、实时任务,采用主版本/副版本(Primary/Backup,PB)模型,在卫星时间窗口约束的基础上采用主版本-副版本重叠(Primary-Backup Overlapping)技术、副版本-副版本重叠(Backup-Backup Overlapping)技术以及任务合成技术,提高系统的资源利用率.

2 相关工作

由于调度在卫星管控系统中的重要性,目前已有许多关于卫星调度方面的研究.而研究较多的主要集中在静态调度方面.在静态调度中,任务调度时间固定,任务何时开始执行预先决定,通常执行周期性任务.例如,Bianchessi 提出了一种改进的禁忌搜索算法来解决多星、多轨道、多用户调度问题,并且采用 column generation 算法的上界程序来评估方案的质量[8].Frank 等人采用一种基于约束的语言,在启发式策略的基础上利用随机贪婪算法来寻求优化解[5].Zhang 等人采用蚁群优化算法进行多星资源调度[9].Zufferey 等人采用图着色技术来解决卫星调度问题并提出了两种算法:基于禁忌搜索的算法和自适应记忆算法[10].但是,需要注意的是上述提到的卫星调度算法均属于静态调度算法,具有明确的调度周期,调度决策一旦制定,不能被修改,缺乏立即、动态反应,不适合到达时间不确定的非周期实时任务调度.

在动态调度过程中,任务到达时间不确定.例如,Wang 等人提出了一种求解多星动态调度问题的启发式算法,该方法基于任务迭代修复思想,采用启发式规则进行任务替换[11].Billups 等人研究了静态和动态调度问题,并提出了几种解决方法,如贪婪动态调度方法、遗传算法、整数规划方法和基于图论的方法等等[12].Dilkina 和 Havens 研究了一种敏捷卫星调度方法,允许新达到的对地观测请求动态加入[23].该研究集成了变化搜索的优势并采用了约束繁衍方法.但是上面的动态调度方法并不针对实时任务,因此不能保证任务的时效性.

针对卫星动态容错调度问题,据我们所知,目前尚没有此方面的研究成果.但在一些相关领域,如集群计算、网格计算以及其它多处理器系统中,容错调度问题已有较多研究.其中,采用主版本/副版本模型(PB 模型)是最为流行且有效的容错调度手段.在 PB 模型中,一个任务的两个版本被分配到两个不同的节点上,并通过接收测试来验证调度的正确性[14].例如,Ghosh 等人提出了撤销和重叠两种技术以降低系统开销,在容错调度的同时提高了系统的可调度性.Manimaran 等人通过将处理器划分为多个不同的组来实现在同一时刻对多个失效节点的容错[15].此外,Al-Omari 等人研究了一种基于主版本-副版本重叠的技术,即一个任务的主版本可以和其它任务的副版本重叠,从而提高了系统的可调度性[16].值得注意的是,上述这些容错调度方法属于被动副版本模式,即一个任务的副版本只有在其对应主版本所在节点失效时才执行.此外,假设任务裕度至少为任务执行时间的两倍.与被动副版本模式相对应的为主动副版本模式.在主动副版本模式中,允许一个任务的主版本与其副版本同时执行,也就是说任务的副版本可以在其主版本完成前开始执行.通常情况下,主动副版本模式用于任务具有较小裕度的情况.例如,Tsuchiya 提出了一种重叠方法,每个任务的两个版本可以在不同的开始时间执行[14].Yang 等人研究了一种容错调度算法,允许一个任务的两个版本同时执行来提高可调度性[17].在我们前期的研究工作中,提出了一些集群系统中容错调度算法[18,19],充分考虑了任务的时间限制、QoS 需求和系统可靠性.但是,由于卫星对地观测与其它分布式计算的较大差别,上述提到的算法并不适用于多星实时容错调度.

本文重点研究多星动态容错调度策略,用于调度非周期、独立、实时任务.其中,详细分析了时间窗口限制下的 PB 重叠和 BB 重叠技术,并采用任务合成方法提高卫星资源利用率.

3 系统模型

本节介绍本文采用的模型、概念和术语.

3.1 任务模型

本文考虑的对地观测任务为点目标任务,即卫星传感器可以对其一次完成成像.点目标任务可以表示为一组独立、非周期、实时任务集合,即 $T=\{t_1,t_2,\cdots,t_n\}$.由于在容错调度中采用主版本/副版本模型(PB 模型),对于任意一个任务 t_i 的主版本和副版本可分别表示为 t_i^P 和 t_i^B.t_i^P 和 t_i^B 执行在不同的卫星上以实现容错.任务 t_i 可表示为 $t_i=(a_i,d_i,rs_i)$,其中 a_i、d_i 和 rs_i 分别表示任务 t_i 的到达时间、截止期和观测分辨率需求.

卫星资源集合为 $R=\{r_1,r_2,\cdots,r_m\}$.文中所述的对地观测资源是指卫星上的用来进行目标成像的传感器资源.任意资源 r_j 可表示为 $r_j=(du_j,\sigma_j,s_j,b_j,o_j,as_j,msg_j,bor_j)$,其中 du_j、σ_j、s_j、b_j、o_j、as_j、msg_j 和 bor_j 分别是

资源 r_j 的任务执行时间、视场角、侧摆速率、启动时间、关机滞留时间、姿态稳定时间、最大侧摆角度和最佳地面观测分辨率[20].点目标能被传感器单个视场覆盖,大小可忽略不计,所以 r_j 上所有任务执行时间相同,记为 d_j.

令 f_i^P 和 f_i^B 分别表示任务 t_i 主版本 t_i^P 和副版本 t_i^B 的完成时间. $\boldsymbol{ST}^P=(st_{ij}^P)_{n\times m}$ 为主版本 t_i^P 的开始时间矩阵,其中元素 st_{ij}^P 表示主版本 t_i^P 在资源 r_j 上的开始时间.类似地, $\boldsymbol{ST}^B=(st_{ij}^B)_{n\times m}$ 为副版本 t_i^B 的开始时间矩阵,其中元素 st_{ij}^B 表示副版本 t_i^B 在资源 r_j 上的开始时间.相应地,令 $\boldsymbol{FT}^P=(ft_{ij}^P)_{n\times m}$ 和 $\boldsymbol{FT}^B=(ft_{ij}^B)_{n\times m}$ 分别表示主版本 t_i^P 和副版本 t_i^B 的完成时间矩阵. $r(t_i^P)$ 和 $r(t_i^B)$ 分别表示 t_i^P 和 t_i^B 被分配到的资源.

$AO_{ij}^P=\{ao_{ij1}^P,ao_{ij2}^P,\cdots,ao_{ijK_{ij}}^P\}$ 为主版本 t_i^P 在资源 r_j 上的观测机会集合. AO_{ij}^P 表示 t_i^P 在所有资源上的观测机会集合, $AO_i^P=\bigcup_{r_j\in R}AO_{ij}^P$. 对于任意一个观测机会 $ao_{ijk}^P\in AO_i^P$, $ao_{ijk}^P=\{[ws_{ijk}^P,we_{ijk}^P],\theta_{ijk}^P\}$,其中 $[ws_{ijk}^P,we_{ijk}^P]$ 为观测机会 ao_{ijk}^P 的可观测时间窗口, θ_{ijk}^P 为理想侧摆角,如图 1 所示.

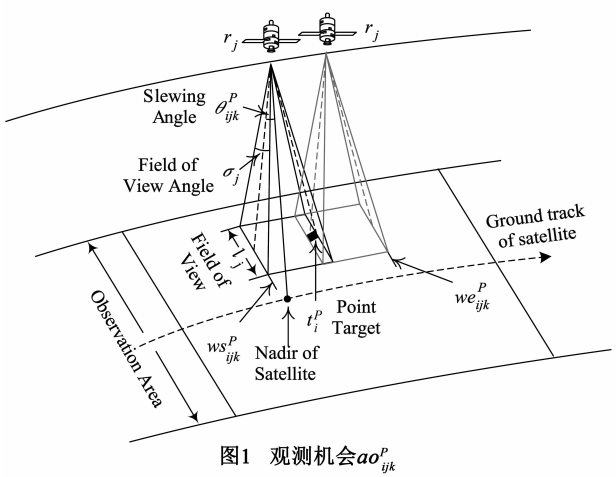

图1　观测机会 ao_{ijk}^P

相应地有 $AO_{ip}^B=\{ao_{ip1}^B,ao_{ip2}^B,\cdots,ao_{ipQ_{ip}}^B\}$,其中 $p\neq j$, $AO_i^B=\bigcup_{r_p\in R}AO_{ip}^B$, $ao_{ipq}^B=\{[ws_{ipq}^B,we_{ipq}^B],\theta_{ipq}^B\}$. ao_{ijk}^P 和 ao_{ipq}^B 为有效的观测机会必须满足:

(1) $we_{ijk}^P>a_i$, $we_{ipq}^B>a_i$.

(2) $ws_{ijk}^P<d_i$, $ws_{ipq}^B<d_i$.

(3) $\theta_{ijk}^P\in[-msg_j,msg_j]$, $\theta_{ipq}^B\in[-msg_j,msg_j]$.

根据任务到达时间和截止期,可以进一步决定任务的有效观测机会.

$$ao_{ijk}^P=\begin{cases}\{[a_i,we_{ijk}^P],\theta_{ijk}^P\}, & \text{if } ws_{ijk}^P<a_i,a_i+du_j\leqslant we_{ijk}^P\\ \{[ws_{ijk}^P,d_i],\theta_{ijk}^P\}, & \text{if } we_{ijk}^P>d_i,d_i-du_j\geqslant ws_{ijk}^P\end{cases}\tag{1}$$

副版本的有效观测机会也可采用类似式(1)的计算方法.

矩阵 $\boldsymbol{X}^P=\{x_{ijk}^P\}_{n\times m\times K_{ij}}$ 表示主版本 t_i^P 的调度决定, $x_{ijk}^P=1$ 表示 t_i^P 被分配到资源 r_j 上的第 k 个观测机会,否则 $x_{ijk}^P=0$.同样, $\boldsymbol{X}^B=\{x_{ipq}^B\}_{n\times m\times Q_{ipq}}$, $x_{ipq}^B=1$ 表示 t_i^B 被分配到资源 r_p 上的第 q 个观测机会,否则 $x_{ipq}^B=0$.因此,可知 $r(t_i^P)=j\Leftrightarrow x_{ijk}^P=1$, $r(t_i^B)=p\Leftrightarrow x_{ipq}^B=1$. Z_i^P 和 Z_i^B 表示主版本 t_i^P 和副版本 t_i^B 的所有可能的调度方案. $z_i^P\in Z_i^P$ 表示 t_i^P 的调度选择, $z_i^B\in Z_i^B$ 表示 t_i^B 的调度选择. t_i^P 采用调度选择 z_i^P 的观测分辨率 $or(z_i^P)$ 可由公式(2)计算:

$$or(z_i^P)=\frac{bor_j\cdot L_{ijk}^P}{H_j}\tag{2}$$

$L_{ijk}^P=(H_j+R)\cos\theta_{ijk}^P-\sqrt{R^2-(H_j+R)\sin^2\theta_{ijk}^P}$,其中, R 表示地球半径, θ_{ijk}^P 表示 t_i^P 在资源 r_j 的第 k 个时间窗口的观测角度, H_j 表示资源 r_j 所在的轨道高度. $or(z_i^B)$ 可采用类似公式(2)的计算得到. Z_i^P 和 Z_i^B 是可行的调度必须满足:(1) $f_i^P\leqslant d_i$, $f_i^B\leqslant d_i$;(2) $or(z_i^P)\leqslant rs_i$, $or(z_i^B)\leqslant rs_i$.

3.2　失效模型

本文研究的卫星失效模型基于以下假设:

(1)由于两颗卫星同时失效的可能性很小,假设某一时刻系统中只有一个卫星失效,在下一个卫星失效前,那些因前一故障而执行失败的任务主版本通过运行任务副版本而正确结束.

(2)在对任务进行容错调度时,系统能及时诊断并报告失效卫星,调度时不会将任务再分配给失效的卫星.

(3)一颗卫星失效不会导致其它卫星失效,即系统提供故障隔离机制.

3.3　调度目标

本文研究的主要目标是在采用 PB 容错调度时尽可能接收更多的实时任务.由于一个任务的主版本被成功分配决定了任务的完成情况,因此有如下表达式:

$$AN(X^P)=\max_{t_i^P\in T,r_j\in R}\left\{\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{K_{ij}}x_{ijk}^P\right\}\tag{3}$$

另外一个调度目标是为了获取尽量好的观测分辨率,即在时间约束范围内最小化所有任务的分辨率之和:

$$OR(X^P,X^B)=\min_{t_i^P\in T,t_i^B\in T,r_j\in R}\left\{\frac{ORP+ORB}{CPB}\right\}\tag{4}$$

其中, $ORP=\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{K_{ij}}x_{ijk}^P or(z_i^P)$,

$$ORB=\sum_{p=1}^{m}\sum_{i=1}^{n}\sum_{q=1}^{Q_{ip}}x_{ipq}^B or(z_i^B),$$

$$CPB = \sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{K_{ij}} x_{ijk}^{P} + \sum_{p=1}^{m}\sum_{i=1}^{n}\sum_{q=1}^{Q_{ip}} x_{ipq}^{B}.$$

4 重叠设计与分析

由于卫星资源出错的概率较小,通常情况下大量的副版本仅占用资源却不被执行,因此,为了有效提高卫星资源利用率,进而提高任务调度成功率,本文在任务调度过程中采用两种重叠技术,即副版本-副版本(BB)重叠和主版本-副版本(PB)重叠.

命题 1 任务调度可实现对一颗卫星资源容错当且仅当任务的两个版本被分配到不同的卫星资源上,即:

$$\forall\, t_i \in T, r(t_i^P) \neq r(t_i^B) \tag{5}$$

如前所述,任务的副版本可以采用主动执行模式也可以采用被动执行模式.但是,在我们的容错方案中,仅采用被动执行模式.这是因为在卫星实际飞行中,两颗卫星同时对一个区域进行观测的概率很小.因此,给出如下命题:

命题 2 任务副版本 t_i^B 的开始时间晚于其对应主版本 t_i^P 的完成时间,即:

$$\forall\, t_i \in T, st_{ip}^{B} > ft_{ij}^{P}, j \neq p \tag{6}$$

命题 3 如果主版本 t_i^P 可以在截止期内完成,那么 t_i^P 执行完成后,其对应副版本 t_i^B 所占用的时间槽将被立即释放,t_i^B 不需执行.

命题 4 采用重叠技术当且仅当任务的主版本和副版本可以在观测机会内执行,即:

$$\forall\, t_i^P \in T, r_j \in R, ao_{ijk}^{P} \in AO_{ij}^{P},$$
$$st_{ij}^{P} \in [ws_{ijk}^{P}, we_{ijk}^{P} - du_j], \theta_{ijk}^{P} \in [-msg_j, msg_j] \tag{7}$$
$$\forall\, t_i^B \in T, r_p \in R, ao_{ipq}^{B} \in AO_{ip}^{B},$$
$$st_{ip}^{B} \in [ws_{ipq}^{B}, we_{ipq}^{B} - du_j], \theta_{ipq}^{B} \in [-msg_p, msg_p] \tag{8}$$

命题 5 一个任务可以被分配当且仅当其观测分辨率小于或等于用户需求分辨率,即:

$$\forall\, t_i^P \in T, t_i^B \in T, or(z_i^P) \leqslant rs_i \wedge or(z_i^B) \leqslant rs_i \tag{9}$$

4.1 BB 重叠

为了提高卫星资源利用率,我们在容错设计中首先采用 BB 重叠技术,即一个任务的副版本可以与其它任务的副版本重叠.图 2 给出了一个 BB 重叠的例子.

粗实框表示主版本 t_1^P 的观测机会,粗虚框表示副版本 t_1^B 的观测机会,细实框表示主版本 t_2^P 的观测机会,细虚框表示副版本 t_2^B 的观测机会.

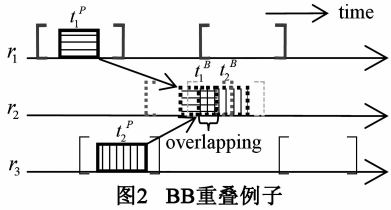

图2 BB重叠例子

从图 2 可以发现,t_1^B 和 t_2^B 可以重叠,其原因可从以下 4 个方面分析:

(1)如果资源 r_1、r_2 和 r_3 没有失效,t_1^P 和 t_2^P 可成功完成,t_1^B 和 t_2^B 不必执行,因此可以重叠;

(2)如果资源 r_1 失效,t_1^P 不能成功完成,所以 t_1^B 不得不执行.根据 3.2 节的假设,r_2 和 r_3 将正常运行,因此 t_2^P 可以成功完成.由于 $st_{22}^{B} > ft_{23}^{P}$,当 t_2^P 成功完成后,被 t_2^B 所占用的时间槽将被释放,不会导致 t_1^B 和 t_2^B 在执行时间上的冲突,因此 t_1^B 和 t_2^B 可以重叠;

(3)如果资源 r_3 失效,t_2^P 不能成功完成,类似上一种情况的分析,t_1^B 和 t_2^B 不存在执行时间冲突,可以进行重叠;

(4)如果资源 r_2 失效,t_1^B 和 t_2^B 不能被成功完成.基于假设,资源 r_1 和 r_3 可正常运行,t_1^P 和 t_2^P 可以被成功完成,因此 t_1^B 和 t_2^B 重叠不影响容错.

需要注意的是,t_1^P 和 t_2^P 不能被分配到同一个卫星资源上.一个不可行的分配例子如图 3 所示.

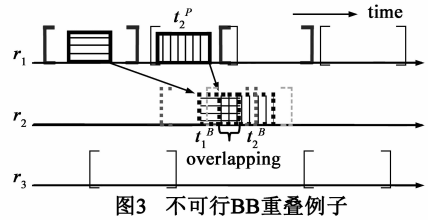

图3 不可行BB重叠例子

从图 3 可以看出,如果资源 r_1 失效,t_1^B 和 t_2^B 必须执行.但是由于 t_1^B 和 t_2^B 重叠,导致 t_1^B 和 t_2^B 执行时间发生冲突.

命题 6 对于主版本 $t_i^P, t_j^P, \cdots, t_k^P$,如果它们的副版本 $t_i^B, t_j^B, \cdots, t_k^B$ 重叠,则 $t_i^P, t_j^P, \cdots, t_k^P$ 不能被分配到同一个卫星资源上.

$$r(t_i^P) \cap r(t_j^P) \cap \cdots \cap r(t_k^P) = \varnothing \tag{10}$$

根据卫星对地观测的特点,如果一些邻近目标同时在卫星的可见区域内,则可同时观测[17].因此,可以在重叠过程中通过合成方式,减少观测任务数量,提高系统的可调度性.图 4 给出了一个例子.

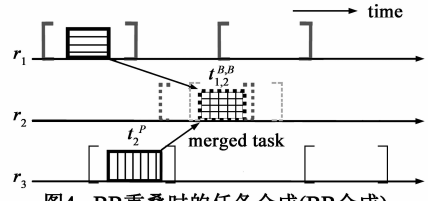

图4 BB重叠时的任务合成(BB合成)

在图 4 中,$t_{1,2}^{B,B}$表示由 t_1^B 和 t_2^B 合成的任务.容易得到无论 r_1、r_2 和 r_3 任何一个资源失效,都不会影响容错,即 t_1 和 t_2 可被成功完成.关于合成约束将在 4.4 节进行介绍.

此外,如果一些副版本被合成,它们对应的主版本不能被分配到同一个资源上.

4.2 PB 重叠

PB 重叠即一个任务的主版本可和其它任务的副版本重叠.图 5 给出了一个 PB 重叠例子.

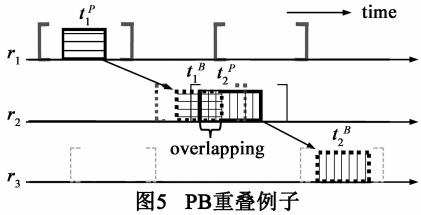

图5　PB重叠例子

在图 5 的例子中,t_1^B 可以和 t_2^P 重叠,其原因可以从以下 4 个方面分析:

(1)如果资源 r_1、r_2 和 r_3 没有失效,在 t_1^P 和 t_2^P 被成功完成后,t_1^B 和 t_2^B 不需要执行,t_1^B 和 t_2^B 所占用的时间槽将被释放.所以 t_1^B 和 t_2^P 没有机会同时执行,进而 t_1^B 和 t_2^P 可以重叠;

(2)如果资源 r_1 失效,t_1^B 必须执行.根据假设,r_2 和 r_3 正常运行.由于 t_1^B 和 t_2^P 存在重叠,为了使 t_1^B 成功完成,t_2^B 必须选择执行,同时放弃 t_2^P 的执行.通过此操作,t_1^B 和 t_2^P 可重叠;

(3)如果资源 r_2 失效,在我们假设的基础上 t_1^P 和 t_2^B 可以被成功完成,因此 t_1^B 和 t_2^P 可以重叠;

(4)如果资源 r_3 失效,t_1^P 和 t_2^P 可以被成功完成.由于在 t_1^P 成功完成后 t_1^B 所占用资源被释放,因此,t_1^P 和 t_2^B 不存在执行时间上的冲突,t_1^B 和 t_2^P 可重叠.

但是,如果 t_i^P 的开始时间早于 t_i^B 的开始时间,t_i^P 不能和 t_i^B 重叠.图 6 给出了一个 *PB* 重叠不可行的例子.

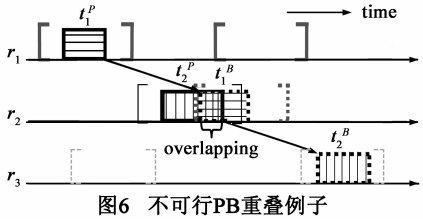

图6　不可行PB重叠例子

假设资源 r_1 失效,t_1^B 必须执行,但是 t_2^P 已在 t_1^B 之前运行,且不能被终止,因此 t_1^B 和 t_2^P 存在时间冲突.命题 7 给出了主版本和副版本重叠的一个约束.

命题 7　如果主版本 t_i^P 可以和副版本 t_j^B 重叠,t_i^P 的开始时间必须晚于 t_j^B 的开始时间,即:

$$st_{ip}^P > st_{jp}^B \tag{11}$$

与 BB 重叠类似,t_1^P 和 t_2^B 不能被分配到同一资源上,因此有如下命题:

命题 8　如果主版本 t_i^P 与副版本 $t_j^B,\cdots,t_k^B$ 重叠,t_i^B 和 $t_j^P,\cdots,t_k^P$ 不能被分配到同一资源上,即:

$$r(t_i^B)\cap r(t_j^P)\cap\cdots\cap r(t_k^P)=\varnothing \tag{12}$$

在图 5 的基础上,如果 t_1^B 和 t_2^P 可以合成,t_1^P 和 t_2^B 可以被分配到同一资源,如图 7 所示.

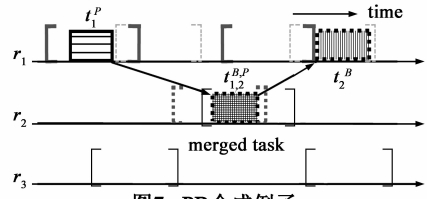

图7　PB合成例子

4.3 PP 合成

从上面重叠技术的分析可知,两个主版本之间不能重叠,否则不能实现容错.但是除了在 PB 重叠时的 PB 合成以及 BB 重叠时的 BB 合成之外,还可以进行主版本-主版本合成,即 PP 合成.在图 8 中,本文给出了 3 种可进行 PP 合成的例子.

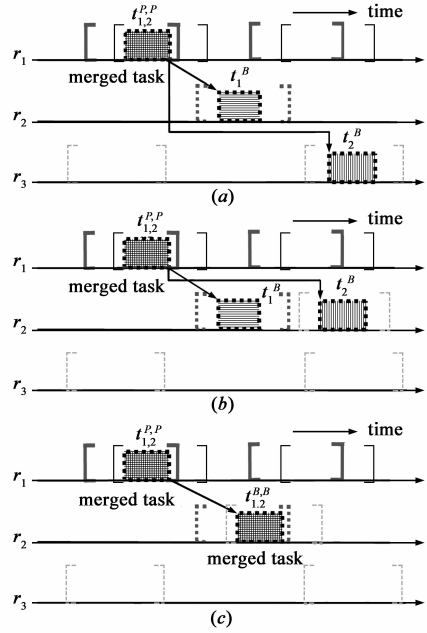

图8　可行PP合成例子

图 8(*a*)显示如果 t_1^P 和 t_2^P 合成,则其对应的副版本 t_1^B 和 t_2^B 须分配到不同的资源上.也就是说,如果 t_1^P 和 t_2^P 进行合成操作,则 t_1^B 和 t_2^B 在进行分配时,被分配到除 $r(t_{12}^{PP})$之外的不同资源上是可行的.从图 8(*b*)中可以看出,如果资源 r_1 失效,任务 t_1^B 和 t_2^B 可以成功完成;

如果资源 r_2 失效,合成任务 $t_{1,2}^{P,P}$可成功完成.因此可知,如果 t_1^P 和 t_2^P 进行合成操作,则 t_1^B 和 t_2^B 在进行分配时,被分配到除 $r(t_{1,2}^{P,P})$同一个资源且不重叠是可行的.从图 8(c)不难看出,t_1^P 和 t_2^P 进行合成,其对应的副版本 t_1^B 和 t_2^B 可被分配到同一个资源上且 t_1^B 和 t_2^B 也需进行合成.

但是如果 t_1^B 和 t_2^B 采用重叠方式,则任务 t_1 和 t_2 的版本分配方式是可不行的,即不能实现容错.图 9 给出了一个不可行的 PP 合成例子.

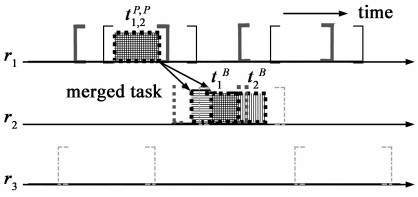

图9 不可行PP合成例子

从图 9 中可以发现,如果资源 r_1 失效,t_1^B 和 t_2^B 不得不执行,但是由于 t_1^B 和 t_2^B 存在执行时间冲突,因此,不能实现容错.

4.4 合成约束

令 t_a^X 和 t_b^Y 表示任意两个待合成的任务,其中 X 和 Y 表示任务版本,即 X 和 Y 可以表示主版本 P 或副版本 B.

第一,t_a^X 和 t_b^Y 必须有重叠的时间窗口,即:

$$\forall\, t_a^X \in T, t_b^Y \in T, r_j \in R, ao_{ajk}^X \in AO_{aj}^X, ao_{bjq}^Y \in AO_{bj}^Y:$$
$$we_{ajk}^X > ws_{bjq}^Y, \text{if } we_{bjq}^Y > we_{ajk}^X,$$
$$we_{bjq}^Y > ws_{ajk}^X, \text{if } we_{ajk}^X > we_{bjq}^Y. \tag{13}$$

第二,重叠的时间窗口大小必须大于等于 t_a^X 和 t_b^Y 的执行时间,即:

$$\min\{we_{ajk}^X, we_{bjq}^Y\} - \max\{ws_{ajk}^X, ws_{bjq}^Y\} \geq du_j \tag{14}$$

第三,资源 r_j 必须能够通过调整观测角度同时覆盖 t_a^X 和 t_b^Y 所指定的观测目标,即:

$$|\theta_{ajk}^X - \theta_{bjq}^Y| \leqslant \sigma_j \tag{15}$$

第四,合成任务 t_{ab}^{XY}的观测分辨率必须等于或优于每个任务的分辨率要求,即:

$$or(z_{ab}^{XY}) \leqslant rs_a, or(z_{ab}^{XY}) \leqslant rs_b \tag{16}$$

如果 t_a^X 与 t_b^Y 合成,则合成任务 t_{ab}^{XY}的时间窗口变为:

$$tw_{ab,j,kq}^{XY} = |\max\{ws_{ajk}^X, ws_{bjq}^Y\}, \min\{we_{ajk}^X, we_{bjq}^Y\}| \tag{17}$$

此外,合成任务 t_{ab}^{XY}的理想侧摆角变为:

$$\theta_{ab,j,kq}^{XY} = \frac{\theta_{a,j,k}^X + \theta_{b,j,q}^Y}{2} \tag{18}$$

因此,t_{ab}^{XY}的观测机会为:

$$ao_{ab,j,kq}^{XY} = \{tw_{ab,j,kq}^{XY}, \theta_{ab,j,kq}^{XY}\} \tag{19}$$

显然,t_{ab}^{XY}的观测机会数目为 t_a^X 与 t_b^Y 可合成的个数.因此,t_{ab}^{XY}可有多个观测机会,可为容错调度提供多个优化选择.

5 容错调度算法

本节介绍多星容错调度算法 FTSS(Fault-Tolerant Satellite Scheduling).

5.1 主版本调度

为了使任务主版本和其对应的副版本被成功分配,主版本应该被尽早执行以为副版本留有足够的时间槽使其可以在截止期内完成.

为了得到主版本的最早开始时间,需考虑任务的准备时间和就绪时间.

定义 1 准备时间 $p_{i-1,i,j}^{XP}$是指从任务主版本 t_i^P 的前一个任务 t_{i-1}^X到开始 t_i^P 所需要的时间.

$$p_{i-1,i,j}^{XP} = o_j + \frac{|\theta_{i-1,j,k}^X - \theta_{i,j,q}^P|}{s_j} + as_j + b_j \tag{20}$$

定义 2 就绪时间 r_{ij}^P是指任务主版本 t_i^P 可以开始的时间.

$$r_{ij}^P = \max\{p_{i-1,i,j}^{XP} + ft_{i-1}, a_i\} \tag{21}$$

命题 9 一个任务主版本 t_i^P 在某一观测机会 ao_{ijk}^P 内执行可以采用以下 3 种方法:

(1)t_i^P 可以和某一任务(或合成任务)合成;

(2)t_i^P 可以和某一任务(或合成任务)重叠;

(3)ao_{ijk}^P存在空闲时间槽可以容纳 t_i^P.

无论上面那一种方法,t_i^P 的完成时间必须早于任务 t_i 的截止期,即:

$$f_i^P < d_i \tag{22}$$

在第一种方法中,必须满足约束(13)至(16)以保证 t_i^P 可以和某一任务(或合成任务)合成.t_i^P 的最早开始时间 est_{ijk}^P等于与之合成的任务 t_i^X 的开始时间,即:

$$est_{ijk}^P = st_{aj}^X \tag{23}$$

对于第二种方法,t_i^P 可以和一个副版本重叠或者和一个不包含任务主版本的合成任务重叠,即 t_i^P 可以和一个合成任务 $t_{i_1,i_2,\cdots,i_k}^{X_1,X_2,\cdots,X_k}$重叠,必须满足:

$$X_1 \neq P \bigcap X_2 \neq P \bigcap \cdots \bigcap X_k \neq P \tag{24}$$

由于 t_i^P 只能与副版本重叠或者与只含有副版本的合成任务重叠,因此需要寻找第一个任务 t_a^X(t_a^X 为副版本或与只含有副版本的合成任务).t_i^P 的最早开始时间 est_{ijk}^P可被计算为:

$$est_{ijk}^P = \max\{r_{ij}^P, st_{aj}^X + \varepsilon\} \tag{25}$$

其中,ε 表示如果 t_a^X 必须执行时取消执行 t_i^P 的时间.

对于第三种方法,同样需要计算 t_i^P 的最早开始时间 est_{ijk}^P.为不失一般性,在计算 est_{ijk}^P之前,假设任务 t_{i_1}, $t_{i_2},\cdots,t_{i_q}$已经被分配到资源 r_j 的第 k 个观测机会.为简化模型,我们假设这些任务既可以是主版本也可以是副版本.此外,它们也可以是单个任务或是合成任务.观测机会 ao_{ijk}^P的空闲时间槽可以表示为$[ws_{ijk}^P, st_{i_1}]$, $[ft_{i_1}, st_{i_2}],\cdots,[ft_{i_{q-1}}, st_{i_q}],[ft_{i_1}, we_{ijk}^P]$.为获得 est_{ijk}^P,所有这些空闲时间槽按时间升序扫描,选择第一个满足下面约束的空闲时间槽$[ft_{i_k}, st_{i_{k+1}}]$,

$$\max\{r_{ij}^P, ft_{i_k} + p_{i_k,i,j}^{XP}\} + du_j \leqslant st_{i_{k+1}} \tag{26}$$

因此,est_{ijk}^P可由下式计算:

$$est_{ijk}^P = \max\{r_{ij}^P, ft_{i_k} + p_{i_k,i,j}^{XP}\} \tag{27}$$

在调度算法 FTSS 中,令主版本的开始时间为其最早开始时间以尽早执行.

5.2 副版本调度

对于主版本已经被分配的副版本,应该被尽可能晚的执行以留出较早的时间槽保障较晚到达的任务能在其截止期内完成.此外,应该选择较小的空闲时间槽以为其它任务留出时间,提高任务调度成功率.

命题 10　一个任务副版本 t_i^B 在某一观测机会 ao_{ipq}^B 内执行可以采用以下 3 种方法:

(1)t_i^B 可以和某一任务(或合成任务)合成;

(2)t_i^B 可以和某一任务(或合成任务)重叠;

(3)ao_{ipq}^B存在空闲时间槽可以容纳 t_i^B.

在第一种方法中,约束(13)至(16)必须满足以保证 t_i^B 可以和某一任务(或合成任务)合成.t_i^B 的最晚开始时间 lst_{ipq}^B等于与 t_i^B 待合成任务 t_X^B 的开始时间,即:

$$lst_{ipq}^B = st_{bp}^X \tag{28}$$

下面,我们讨论在第二种和第三种情况下任务副版本的最晚开始时间.对任意一个副版本 t_i^B,其最晚开始时间 lst_{ipq}^B可以通过从右向左扫描时间窗口获得:

$$lst_{ipq}^B = \begin{cases} \min\{[we_{ipq}^B], d_i\} - du_j, & \text{if } lst = idle \vee ol(t_i^B), \\ st_{bp}^X - p_{ibp}^{BX} - du_j, & \text{if } ts = idle \vee ol(t_i^B). \end{cases} \tag{29}$$

其中,lst 表示最晚的时间槽,$ol(t_i^B)$表示 t_i^B 可以在最晚时间槽内与其它任务(或合成任务)合成.st_{bp}^X表示任务 t_b^X 的开始时间(t_b^X 为不与 t_i^B 重叠且紧接 t_i^B 执行的任务).ts 表示被 t_b^X 占用的时间槽.p_{ibp}^{BX}表示在 t_i^B 完成后,t_b^X 开始的准备时间.

在 FTSS 中,副版本的开始时间被设定为其最晚开始时间.

6 性能评估

本文通过模拟实验测试 FTSS 算法的性能,将其与其它 3 个 baseline 算法进行比较,即不考虑合成容错调度算法 NMFTSS,不考虑重叠容错调度算法 NOFTSS 和既不考虑合成又不考虑重叠容错调度算法 NMNOFTSS.

6.1 模拟方法和参数

为了验证 FTSS 算法的性能,目标在纬度 $-30°\sim60°$,经度 $0°\sim150°$的区域中均匀随机分布.目标规模分别为 200,400,600,800,1000,1200.本文考虑 10 种不同类型卫星传感器资源.表 1 给出了传感器相关参数,其中卫星轨道模型来自卫星仿真工具 STK,带 * 号的参数是根据文献设计的,其它参数为卫星和传感器的真实数据.

(1)任务批次到达时间可表示为 $a_i = a_{i-1}$ + intervalTime,其中 intervalTime 为非负的均匀分布随机数,$a_0 = 0$;

(2)任务 t_i 的截止期为 $d_i = a_i$ + Deadline,其中 Deadline 为正态分布随机数,Deadline ~ N(baseDeadline, baseDeadline/10).

表 1　卫星传感器参数

Sensor	Satellite	bs	msg	FOV	du^*	s^*	b^*	o^*	as^*	Height
Sensor1	IKONOS-2	4	45	0.931	1	0.5	5	5	10	681
Sensor2	QuickBird-2	2.44	25	2.1	1	0.5	5	5	10	450
Sensor3	SPOT-4	10	27	4.2	1	0.5	5	5	10	832
Sensor4	SPOT-5	10	27	2.09	1	0.5	5	5	10	822
Sensor5	ORBVIEW-03	1	27	1.0	1	0.5	5	5	10	470
Sensor6	JB-3A	3	32	5.72	1	0.5	5	5	10	300
Sensor7	EROS-A01	1.9	45	1.6	1	0.5	5	5	10	500
Sensor8	CBERS-1	19.5	32	8.32	1	0.5	5	5	10	778
Sensor9	CBERS-2	10	32	8.32	1	0.5	5	5	10	778
Sensor10	ERS-2	10	25	7.34	1	0.5	5	5	10	780

表 2 列出了相关实验参数及其取值范围.

表 2 实验参数

Parameters	Value(Fixed) - (Varied)
Task Number	(1000) - (200,400,600,800,1000,1200)
Resource Number	(10)
intervalTime(h)	([20,40]) - [0,20],[20,40],[40,60],[40,60],[60,80],[80,100])
u(%)	(50)
baseDeadline(h)	(30)

6.2 任务数量对算法的影响

本组实验测试任务数量对算法性能的影响.实验结果如图 10 所示.

图 10(a)显示随着任务数量的增加,所有算法调度成功率减小.这是因为在资源数量不变的情况下,任务数量增加,使得系统负载增加,因此任务的调度成功率减小.此外,图 10(a)显示 NMNOFTSS 算法的任务调度成功率最低,这是因为 NMNOFTSS 在主副版本分配过程中既没有采用重叠技术也没有采用合成技术,未能较好利用系统资源.相反,FTSS 的调度成功率总体上最高,这是因为在容错调度过程中,FTSS 考虑了多种重叠技术,提高了系统资源利用率.同时,通过合成技术减少了实际需要执行任务的个数.因此调度成功率最高.此外,NMFTSS 的调度成功率优于 NOFTSS 的调度成功率,这是因为合成技术比重叠技术的约束条件更为严格,因此 NOFTSS 的调度成功率略低.当任务数量变化时,FTSS 的调度成功率相比 NMFTSS、NOFTSS、NMNOFTSS 分别提高了 1.2%,3.4%,6.3%.

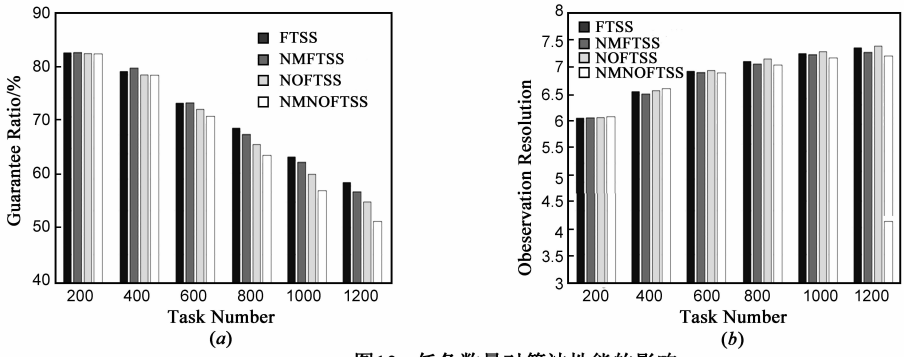

图10 任务数量对算法性能的影响

图 10(b)显示,随着任务数量的增加,所有算法的任务观测分辨率随之变大.这是因为,在资源数量不变情况下,任务数量增加,增加了系统的负载,系统在优先考虑接收更多任务时,不得不牺牲一定的任务分辨率(所有接收任务的分辨率均满足用户需求).从图 10(b)中可以看出,在系统负载较轻时(如任务数小于 400),FTSS 的任务分辨率小于 NMNOFTSS 的任务分辨率,这说明 FTSS 在接收任务的同时尽量优化任务分辨率,为用户提供更好的服务;而当系统负载较重时(如任务数大于 600),FTSS 的任务分辨率大于 NMNOFTSS 的任务分辨率,这是因为 FTSS 优先考虑接收更多的任务,因此会牺牲一定的任务分辨率.此外,NMFTSS 的任务分辨率优于 NOFTSS 的任务分辨率,这是因为在任务合成过程中为能够在同一次成像过程中完成多个任务,会进行传感器的侧摆角调整,降低了任务的分辨率,而在 NMFTSS 中,不需要为兼顾完成其它任务调整传感器侧摆角,因此任务分辨率略高.

6.3 任务到达率对算法的影响

本组实验评估任务到达率对算法性能的影响.实验结果如图 11 所示.

图 11(a)显示,当参数 intervalTime 的取值从[0,20]增大到[20,40]时,所有算法的任务调度成功率均有所下降,其原因主要有以下 2 个方面:第一,参数 intervalTime 的取值在[20,40]之间时,系统的负载仍然很重,此时在新任务到达时,前面到达的任务大多还未完成,但是由于到达速率有所降低后,前面一些任务可能被接收,而导致后面较多任务没有完成,从而调度成功率有所下降;第二,在任务到达较为密集时,任务有较多机会可以进行合成与重叠,即满足任务之间有时间交叠的机会较多,而当任务达到相对稀疏时(此时系统负载依然很重),合成和重叠的机会较少,因此,调度成功率有所下降.而当 intervalTime 的取值从[20,40]变化到[80,100]的过程中,所有算法的调度成功率随之提高.这是因为系统负载逐渐变轻,在新任务到达时,在其前面到达的某些任务已经执行完成,有更多的机会可以在截止期内完成.在参数 intervalTime 的变化过程中,FTSS 的调度成功率始终优于其它算法,相比 NMFTSS、NOFTSS、NMNOFTSS 分别提高了 2.0%,5.3%,5.9%.这是因为 FTSS 采用了多个版本重叠技术和任务合成技术,因此具有最好的性能.

从图 11(*b*)中可以发现,所有算法的任务调度分辨率差别不大.其中的差别可从以下方面分析.在系统负载较轻时(如 intervalTime 的取值在[80,100])所有算法的任务分辨率减小,这是因为在这种情况下,更多的任务可以被接收,同时可以为其提供较小的任务分辨率.此外,在 4 个算法中,NMNOFTSS 的任务分辨率相对最小,但是其接收的任务个数最少(见图 11(*a*)).在应急情况下,系统需要尽量多地为用户在规定时间内观测更多的任务同时满足观测分辨率要求,因此尽管 NMNOFTSS 的分辨率最小,但却牺牲了执行更多任务的机会.相反,FTSS 由于采用了合成与重叠技术,尽量提高任务调度成功率,而同时对任务分辨率的影响不大,与 NMNOFTSS 的平均分辨率之差只有 0.065867.

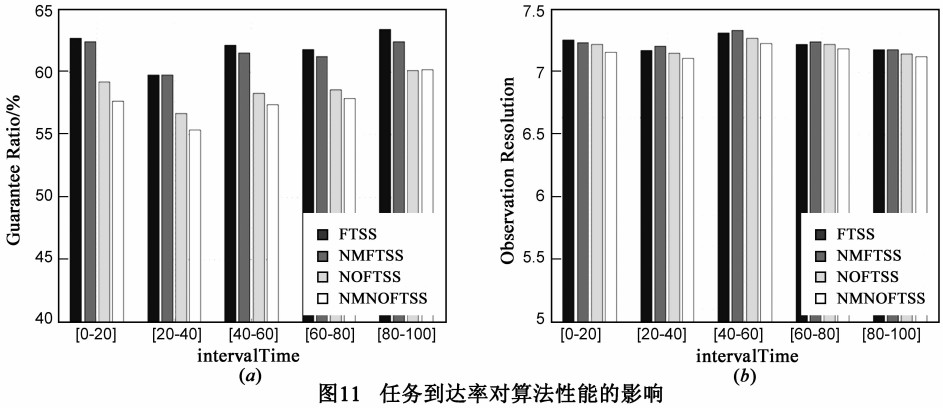

图11　任务到达率对算法性能的影响

7　结论

本文研究了面向多星容错的实时任务调度问题,通过采用主版本/副版本(PB)技术进行软件容错,以应对资源失效问题.本文首先建立了基于 PB 技术容错调度模型,详细分析了多版本间可重叠的条件,以及进行任务合成的相关约束.在此基础上提出了一种面向容错的多星实时任务调度算法 FTSS,该算法充分考虑了任务的合成与重叠,可在实现容错同时有效提高系统的资源利用率,从而提高任务完成率和分辨率.为了评估 FTSS 算法的性能,本文通过模拟实验对其进行了测试.实验结果表明 FTSS 相比其它算法具有更好的调度性能,适合动态环境下多星实时任务容错调度.

参考文献

[1] 文江平.卫星军事应用技术[M].北京:国防工业出版社,2007.1－5.
Wen Jiangping. Satellite Military Application[M]. Beijing: National Defense Industry Press, 2007. 1－5. (in Chinese)

[2] 王钧,等.一种应急条件对地观测卫星成像调度方法[J].电子学报,2008,36(9):1715－1722.
Wang Jun, et al. A multi-objective imaging scheduling approach of earth observation satellite for emergent conditions[J]. Acta Electronica Sinica, 2008, 36(9): 1715－1722. (in Chinese)

[3] Charlotte N C. ORS-1 Satellite, designed and built for combatant command operations, launches aboard a Minotaur 1 Rocket [EB/OL]. http://www.militaryaerospace.com/articles/2011/07/ors-1-satellite-designed.html, 2011-07-01.

[4] Lin W C. Hybrid algorithms for satellite imaging scheduling [A]. Proceedings of the International Conference on Systems, Man and Cybernetics[C]. New York: IEEE Press, 2005. 2518－2523.

[5] Frank J. Planning and scheduling for fleets of earth observing satellites[A]. Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics, Automation & Space[C]. Quebec: Canadian Space Agency, 2001, 1－8.

[6] Qin X, et al. A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters[J]. Journal of Parallel and Distributed Computing, 2005, 65(8): 885－900.

[7] Luo W. Boosting reliability in fault-tolerant heterogeneous distributed systems through dynamic scheduling[A]. Proceedings of 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing[C]. New York: IEEE Press, 2007, 640－645.

[8] Bianchessi N, et al. A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites [J]. European Journal of Operational Research, 2007, 177(2): 750－762.

[9] Zhang Z. Multi-satellite control resource scheduling based on ant colony optimization[J]. Expert Systems with Applications, 2014, 41(6): 2816－2823.

[10] Zufferey N, et al. Graph coloring approaches for a satellite range scheduling problem[J]. Journal of Scheduling, 2008, 11(4): 263－277.

[11] Wang J M. Study on heuristic algorithm for dynamic schedul-

ing problem of earth observation satellites[A]. Feng W. Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing[C]. New York: IEEE Press, 2007, 9 – 14.

[12] Billups S C. Satellite mission scheduling with dynamic tasking [R], Denver: University of Colorado, 2005.

[13] Dilkina B. Agile satellite scheduling via permutation search with constraint propagation[EB/OL]. http://www.cs.sfu.ca/CourseCentral/827/havens/papers/topic% 2312 (SatelliteScheduling)/SatelliteSched.pdf, 2005-03-31.

[14] Tsuchiya T. A new fault-tolerant scheduling technique for real-time multiprocessor systems[A]. Tokoro M. Proceeding of the 2nd International Workshop on Real-Time Computing Systems and Applications[C]. New York: IEEE Press, 1995, 197 – 202.

[15] Manimaran G, et al. A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis[J]. IEEE Transactions on Parallel and Distributed Systems, 1998, 9(11): 1137 – 1152.

[16] Al-Omari R, et al. Efficient overloading technique for primary-backup scheduling in real-time systems[J]. Journal of Parallel and Distributed Computing, 2004, 64(5): 629 – 648.

[17] Yang C H, et al. Fault-tolerant scheduling for real-time embedded control systems[J]. Journal of Computer Science and Technology, 2004, 19(2): 191 – 202.

[18] Zhu X, et al. QoS-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters[J]. IEEE Transactions on Computers, 2011, 60(6): 800 – 812.

[19] Zhu X, et al. Boosting adaptivity of fault-tolerant scheduling for real-time tasks with service requirements on clusters[J]. Journal of Systems and Software, 2011, 84(10): 1708 – 1716.

作者简介

朱晓敏 男, 1979 年生于辽宁盘锦. 国防科学技术大学信息系统与管理学院副教授. 研究方向为信息资源管理、任务调度、实时系统.
E-mail: xmzhu@nudt.edu.cn

王建江 男, 1986 年生于新疆乌鲁木齐. 博士研究生, 研究方向为卫星任务规划.