

面向天基计算的故障表征与容错机制

史冰冰^{1,2,3,4}, 李泓辛⁴, 刘垚圻^{1,2,3,4}, 周 浩^{1,2,4}, 谭海宁^{1,2}, 韩银和^{1,2}

(1. 中国科学院计算技术研究所处理器芯片全国重点实验室, 北京 100089;

2. 中国科学院计算技术研究所, 北京 100089;

3. 上海科技大学信息科学与技术学院, 上海 201210;

4. 北京中科天算科技有限公司, 北京 100094)

摘 要: 随着商业航天的快速发展, 天基计算系统正面临空间辐射导致的高故障率与任务规模扩张带来的算力需求激增的双重挑战。为提升商用器件的可靠性与算力效率, 构建一个包含物理层、故障层、任务层和评价层的四维系统模型, 并提出“有效平均算力”指标, 以揭示任务调度与资源分配之间的内在联系。在此基础上, 设计一种动态容错机制, 通过时空资源耦合建模并采用混合优化策略, 实现故障-算力-资源的协同平衡。通过蒙特卡洛仿真评估, 提出一种融合冗余配置、检查点优化与任务迁移的混合容错算法, 使系统平均有效算力提升约39.20%, 为天基计算系统的在轨可靠性保障提供理论支撑与可行技术路径。

关键词: 容错计算; 天基计算; 时空资源建模; 混合容错算法

中图分类号: TP302.7

文献标志码: A

doi: 10.11959/j.issn.2096-8930.2026010

Fault Characterization and Fault-Tolerant Mechanism for Space-Based Computing

Shi Bingbing^{1,2,3,4}, Li Hongxin⁴, Liu Yaoqi^{1,2,3,4}, Zhou Hao^{1,2,4}, Tan Haining^{1,2}, Han Yinhe^{1,2}

1. State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100089, China

2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100089, China

3. School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China

4. Comospace Technology Co., Ltd., Beijing 100094, China

Abstract: With the rapid development of commercial aerospace, space-based computing systems are confronted with dual challenges of high failure rates caused by space radiation and the surging demand for computing power resulting from the rapid increase in mission scale. To enhance the reliability and computing efficiency of commercial devices, a four-dimensional system model that includes the physical layer, fault layer, task layer and evaluation layer is constructed, and the "effective average computing power" index to reveal the intrinsic connection between task scheduling and resource allocation is proposed. On this basis, a dynamic fault-tolerant mechanism is designed. Through the coupling modeling of spatio-temporal resources and the adoption of a hybrid optimization strategy, the collaborative balance of faults - computing power - resources is achieved. Through Monte Carlo simulation evaluation, a hybrid fault-tolerant algorithm integrating redundant configuration, checkpoint optimization and task migration is proposed, which increased the average effective computing power of the system by approximately 39.20%, providing theoretical support and feasible technical paths for the in-orbit reliability guarantee of space-based computing systems.

Keywords: fault-tolerant computing, space-based computing, spatio-temporal resource modeling, hybrid fault-tolerant algorithm

收稿日期: 2025-07-29; 修回日期: 2025-11-24

基金项目: 国家自然科学基金 (No. 62025404); 北京市自然科学基金 (No. L241013, No. L258047)

Foundation Items: The National Natural Science Foundation of China(No. 62025404), The Beijing Natural Science Foundation(No. L241013, No. L258047)

0 引言

随着遥感等领域技术的快速进步,天基系统对高分辨率图像获取和快速信息处理的需求不断增强。其中,遥感图像分辨率的提升使得数据量急剧增长,依赖传统地面后处理方式已难以满足时效性要求,亟须提升星上智能处理能力^[1]。在卫星平台集成高性能计算模块可显著降低信息获取到决策应用的时延,但受限传统抗辐射器件的性能瓶颈,算力难以支撑复杂算法的高效运行。

商用现成器件凭借先进制程、高性能和低成本优势,逐渐成为构建新一代天基算力系统的重要选择^[2]。商用现货 (commercial off-the-shelf, COTS) 器件凭借先进制程、高计算密度和成本优势,已成为构建新一代星载计算系统的重要选择,并在多个航天任务中验证了其在轨可行性^[3]。然而, COTS 器件在空间辐射环境下易受单粒子翻转、闩锁效应和总剂量损伤等影响,进而导致故障率升高,且其中多数为可恢复的瞬态故障^[4-5]。因此,面向 COTS 器件的星上容错机制 (如运行状态监测、任务迁移与动态资源调度),成为保障系统可靠性与可用性的关键技术路径。

当前,针对容错机制的设计已提出多种方案。冗余架构 (如双模冗余策略、三模冗余策略) 可屏蔽部分硬件故障,但资源开销大^[6]。固定检查点间隔的回滚策略,其性能依赖于检查点间隔的设定:间隔过长会导致恢复时回退过多,间隔过短则带来频繁的输入输出 (input/output, I/O) 开销和资源消耗^[7]。分布式容错架构可实现任务动态迁移与负载均衡^[8]。然而,这些方法仍存在局限:其一是过度依赖硬件资源;其二是假设系统状态静态,忽略其动态变化特性;其三是缺乏协同优化机制,难以实现全局最优。在容错评估方面,现有研究多采用静态指标,如平均失效前时间 (mean time to failure, MTTF),但其无法准确反映器件在动态环境中的表现。由于多数故障具有瞬时性和可恢复性,仅依赖静态指标难以全面评估容错能力^[9]。为此,研究者们开始探索结合实时算力需求与故障概率的动态评估机制。传统方法的根本问题在于未充分考虑 COTS 器件的空间适应性特征^[10]:一方面,瞬时性故障使静态分析结果偏差较大;另一方面,故障恢复与任务执行之间存在强耦合关系,若不加以建模,将直接影响容错效果^[11-12]。

为应对 COTS 器件在空间辐射环境下的可靠性不足问题,本文提出一种面向天基计算的动态容错优化框架。该框架构建了涵盖物理层、故障层、任务层与评价层的

四维系统模型,统一表征硬件资源、辐射故障、任务需求与系统性能之间的耦合关系,并引入“有效平均算力”作为核心评价指标,以量化动态故障场景下的实际计算效能。在此基础上,设计了一种融合任务级多模冗余、检查点回滚与任务迁移的协同容错机制:通过多模冗余隔离单点故障,利用检查点减少故障恢复开销,并在节点失效时动态迁移任务以保障执行连续性。为实现资源约束下的全局优化,采用模拟退火算法对任务调度序列、冗余度及检查点间隔等关键参数进行联合寻优,以最大化理论有效平均算力。通过蒙特卡洛仿真,将所提方法与无容错调度、固定检查点策略及双模冗余方案进行对比验证。

1 基于时空资源的有效算力建模

传统算力指标每秒万亿次运算 (tera operations per second, TOPS) / 每秒浮点运算次数 (floating-point operations per second, FLOPS) 通常忽略故障影响,难以反映天基计算系统在高辐射环境下的实际效能。频繁的任务重启与进度丢失导致显著算力损耗,仅凭瞬时算力曲线无法量化此类损失 (如图 1 橙色区域所示)。为此,本文从任务执行视角出发,建立“故障-算力”耦合模型,以更准确地刻画故障对系统有效算力的影响。

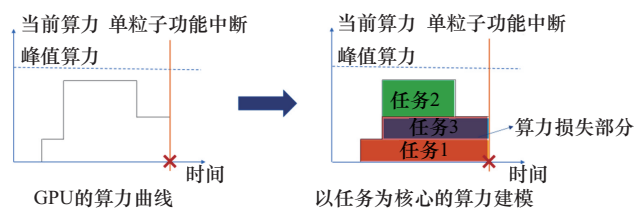


图 1 有效算力建模思路

1.1 物理资源模型

假设有数量为 NG 个的图形处理单元 (graphics processing unit, GPU) 节点: $G = \{G_1, \dots, G_{NG}\}$, 这些 GPU 节点以一定的拓扑关系挂载在 NC 个中央处理单元 (central processing unit, CPU) 节点上: $C = \{C_1, \dots, C_{NC}\}$ 。GPU 节点均具备一定的算力资源:

$$GR_i \in R^+, \forall i \in \{1, 2, \dots, NG\} \quad (1)$$

由于每个 GPU 节点与 CPU 节点均可能发生错误, 设其概率分布函数分别为 $PG_0(t)$ 和 $PC_0(t)$ 。

同时, 现有数量为 NT 个的任务: $T = \{T_1, \dots, T_{NT}\}$, 每个任务的参数如下。

每个任务的资源需求:

$$TR_i \in R^+, \forall i \in \{1, 2, \dots, NT\} \quad (2)$$

每个任务的执行时间：

$$TT_i \in R^+, \forall i \in \{1, 2, \dots, NT\} \quad (3)$$

同时，每个任务均具有一定的数据量，其影响任务传输至板卡上的传输时间：

$$TS_i \in R^+, \forall i \in \{1, 2, \dots, NT\} \quad (4)$$

根据任务的资源需求和执行时间，可以计算出该任务的资源量：

$$TA_i = TR_i \times TT_i, \forall i \in \{1, 2, \dots, NT\} \quad (5)$$

为直观建模并计算板卡与任务的关系，可将板卡视为一个底面积代表其可用计算资源的容器，任务视为一根底面积代表其所需计算资源、高度代表其执行时间的柱子。任务分配过程即为将这些“任务柱”有序放入“板卡盒”的过程。在任务分配过程中，要求同时执行的任务所占用的资源总和不能超过系统的可用资源，即：

$$\sum_{\substack{k, \text{如果任务} T_k \\ \text{正在GPU} \\ \text{节点} G_i \text{上运行}}} TR_k \leq GR_i \quad (6)$$

在该模型中，上述约束可以表示为：柱子之间不能重叠，且柱子总底面积不能超过盒子底面积，完整资源传输模型如图2所示。

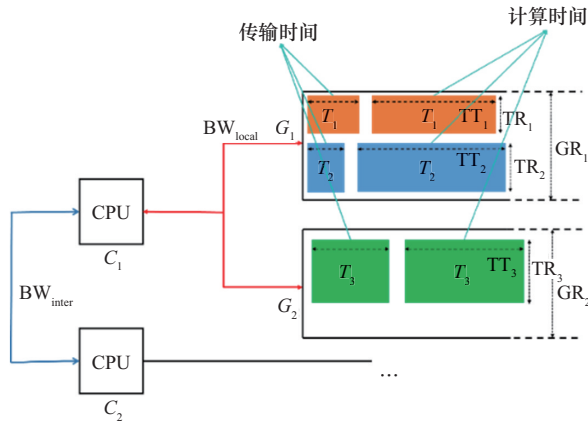


图2 资源传输模型

1.2 任务传输模型

如图2所示，任务数据需传输至GPU节点后才能进行计算，因此本文引入了通信时间。通信主要分为CPU域内通信和CPU间通信两类：CPU域内通信通过外设组件互连高速（peripheral component interconnect express, PCIe）标准传输，设带宽为 BW_{local} ，任务 T_i 需要的传输时间 t 为：

$$t = \frac{TA_i}{BW_{local}}, \forall i \in \{1, 2, \dots, NT\} \quad (7)$$

CPU间通信通常通过以太网、串行快速输入输出（serial rapid IO, SRIO）等方式传输，设其带宽为 BW_{inter} ，任务 T_i 需要的传输时间 t' 为：

$$t' = \frac{TA_i}{BW_{inter}}, \forall i \in \{1, 2, \dots, NT\} \quad (8)$$

1.3 故障模型

本文假设各计算节点的故障事件在物理层面相互独立。这是由于在空间辐射环境中，高能粒子（如质子、重离子）与电子器件的相互作用具有高度空间随机性，通常仅在局部区域沉积能量，进而引发单粒子事件效应（single event effect, SEE）。其中，单粒子翻转事件（single event upset, SEU）、单粒子功能中断事件（single event functional interrupt, SEFI）等瞬时故障多表现为仅影响单一器件或局部电路的单点故障。美国国家航空航天局（national aeronautics and space administration, NASA）与欧洲航天局（european space agency, ESA）的大量在轨统计数据显示，此类局部故障是空间电子系统中的主要失效模式，系统级级联失效则较为罕见^[13]。

本文在假设计算节点故障相互独立的基础上，进一步考虑了系统架构层面的故障传播。CPU作为核心控制单元，其故障将导致所辖PCIe总线、内存控制器等共享资源失效，进而使挂载的全部GPU节点丧失功能。为此，故障模型进行了精细化区分：GPU节点故障属于局部单点故障，可通过检查点或任务迁移恢复；CPU节点故障则被建模为域级失效，会引发其管理范围内所有GPU的连带失效，且需系统重启恢复，恢复时延显著增加。该建模方式更准确地反映了控制节点与计算资源间的依赖关系。假设GPU、CPU节点的失效率恒定，分别为 λ_G 和 λ_C ，则板卡的可靠度服从指数分布^[14]：

$$PG_0(t) = e^{-\lambda_G t} \quad (9)$$

$$PC_0(t) = e^{-\lambda_C t} \quad (10)$$

假设各节点之间发生故障互不影响，且概率服从独立同分布。如果GPU节点发生故障，则其运行的所有任务均需重新开始，且需要经过一定的恢复时间 GRT_0 才能继续工作；如果CPU节点发生故障，则该域内的所有GPU节点均视为故障，同时还需经过一定时间恢复（CPU恢复时间设为 CRT_0 ）。通常，CPU恢复需要经历等待重新上电、操作系统恢复等一系列过程，而GPU恢复可以通过单独上下电形式恢复，因此通常 $CRT_0 > GRT_0$ 。

在实际应用中，系统通过硬件错误校正码（error-

correcting code, ECC)、周期性“心跳信号”及任务检查点校验等多层级手段,实现对 SEU 等瞬时故障的快速检测。一旦检测到 GPU 节点故障,系统将根据检查点状态选择回滚恢复或任务迁移;若 CPU 节点故障,则重启后由调度器重新分配其域内所有任务^[15]。

1.4 算力评价模型

与功耗、重量、体积和算力等可直接度量的物理指标不同,容错能力难以直接量化。由于硬件错误在空间环境中具有随机性和概率性特征,因此,其评估需引入概率建模。为综合衡量系统在故障扰动下的持续服务能力,本文从系统整体性能出发,并考虑以下指标进行计算。

(1) 实际平均算力 (actual average computing power, AAC): 将故障与恢复的时间纳入范围,定义为完成所有任务的计算量与其所需完成时间的比率,用于表示系统的整体性能:

$$AAC = \frac{\sum_{i=1}^{NT} TC_i}{T_{total}} \quad (11)$$

其中, TC_i 表示任务 T_i 所需的计算量, T_{total} 表示完成所有任务总共花费的时间。当任务固定后,执行任务的实际总耗时越小,计算得出的实际平均算力越高。因此实际平均算力代表了系统在实际工作中发挥的算力,是本文需要评价的主要目标。

由于故障的发生具有随机性,实际平均算力需要通过任务执行过程才能准确计算,在任务真正执行前进行分析时,可以采用理论有效平均算力 (theoretical valid average computing power, TAC) 来替代:

$$TAC = \frac{\sum_{i=1}^{NT} TC_i \times P_i}{T_{theo_total}} \quad (12)$$

其中, P_i 表示任务 T_i 完成的概率, T_{theo_total} 表示无故障发生时,完成所有任务所需的时间。

(2) 平均任务完成时间 (average task complete time, ATT): 即每个任务的实际完成时间的平均值。

$$ATT = \frac{\sum_{i=1}^{NT} ATT_i}{NT} \quad (13)$$

其中, ATT_i 表示任务 T_i 从开始到完成所花费的时间,包括故障、恢复等一系列时间开销。

2 容错机制建模

2.1 检查点回滚机制

检查点回滚机制通过在任务执行过程中周期性地保

存中间状态,将大规模计算任务划分为若干个连续子阶段,从而有效减少故障导致的算力资源浪费。其核心在于故障发生后可从最近的有效检查点恢复任务执行,避免全程重算。当系统检测到计算节点故障时,首先会判定受影响任务的检查点状态:对于已完成检查点保存的任务,待节点恢复后将从最近检查点继续执行;对于尚未建立检查点的任务,则需重新启动。该机制在恢复开销与状态保存频率之间存在权衡,需结合故障率与任务特性进行优化配置^[16]。传统故障模型和检查点回滚模型对比如图 3 所示。其中, G_1 代表一个 GPU 处理器能够执行的任务量,算作一个算力资源池。

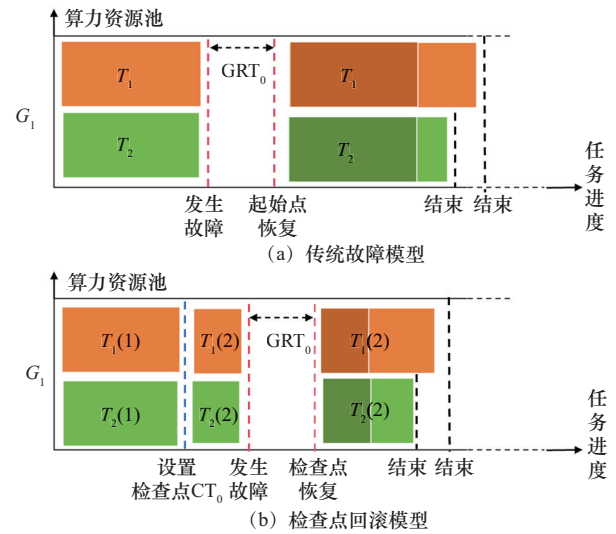


图 3 传统故障模型与检查点回滚模型对比

如图 3 (a) 的传统故障模型所示,当节点发生故障时,若未提前设置检查点,任务 T_1 和 T_2 需从头开始重新执行,导致大量计算资源浪费,图中阴影部分即为重复计算区间;而如图 3 (b) 的检查点回滚模型所示,若在故障发生前已设置检查点,系统可在故障恢复后从最近的检查点继续执行任务,相较于传统故障模型,该模型下的重复计算部分占比较小。这表明,检查点机制有效保留了故障发生前已完成的计算进度,从而显著缩短任务恢复后的总执行时间。因此,在实际系统中引入检查点策略,能够有效减少故障导致的重复计算开销,提升整体计算效率。假设该任务的基础完成时间为 TT_0 ,任务完成的概率为 P_0 ,在任务执行过程中设置 $k-1$ 个检查点,相当于将原任务划分为 k 个子任务。接下来主要考虑任务的实际执行时间。该任务有 P_0 的概率直接完成,总耗时为 TT_0 ;有 $(1-P_0) \cdot P_0$ 的概率经过一次故障重启恢

复后完成，由于故障发生前任务平均执行了一半，因此总耗时为 $TT_0 + TT_0/2$ ；类似地，该任务有 $(1 - P_0)^j \cdot P_0$ 的概率经过 j 次重启恢复后完成，每次故障发生前任务平均执行了一半，对应总耗时为 $TT_0 + TT_0/2 \times j$ ，因此：

$$E(ATT_0) = \sum_{j=0}^{\infty} (1 - P_0)^j \cdot P_0 \cdot \left(TT_0 + \frac{TT_0}{2} \times j \right) \quad (14)$$

$$E(ATT_0) = TT_0 + \frac{(1 - P_0)}{2P_0} \cdot TT_0 \quad (15)$$

同样地，将任务划分为 k 个子任务后，再加上任务检查点所需要的时间，可以计算出：

$$E(ATT_0) = TT_0 + \frac{1 - P_0}{k - (1 - P_0)} \cdot \frac{TT_0}{2} + (k - 1) \cdot CT_0 \quad (16)$$

由式(16)可以看出，该函数是一个与 k 相关的对勾函数，通过求极值可以计算出最优 k 值：

$$\tilde{k} = 1 - P_0 + \sqrt{\frac{(1 - P_0)TT_0}{2CT_0}} \quad (17)$$

进一步，由于 $1 - P_0$ 表示错误概率，根据前文的故障假设（错误概率与任务运行的时间呈正比），其可以表示为 $1 - P_0 = \lambda \cdot TT_0$ ，因此：

$$\tilde{k} = \left(\lambda + \sqrt{\frac{\lambda}{2CT_0}} \right) \cdot TT_0 \quad (18)$$

由此可见，检查点设置个数与硬件故障率、检查点设置所需时间和任务所需时长相关。其中，最佳检查点个数与任务所需时长关系最为密切，且与任务所需时长呈正比。

2.2 任务级多模冗余容错

任务级多模冗余技术通过生成多个功能等效的任务副本，并将其并行部署于物理隔离的计算单元上，结合多数表决机制实现高可靠性，以空间冗余的方式有效隔离单点故障，保障关键任务的连续性与结果的正确性。当某一计算节点发生故障时，系统仅采纳正常副本的输出结果，通过表决确定最终结果，屏蔽并丢弃故障节点的异常输出，从而在多数副本正常运行前提下实现故障的实时、无缝容错^[17]。

与传统依赖硬件同构冗余的 N 模冗余技术不同，该方法基于任务级复制机制，允许多块板卡并行执行不同任务，同时调度部分资源动态承担冗余角色。如图4所示，

板卡 B_3 可为板卡 B_1 和板卡 B_2 的关键任务提供冗余支持，由此形成跨板卡的任务级“1.5模”冗余架构。

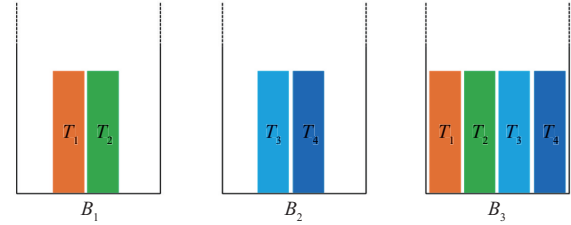


图4 任务级多模冗余模型

其中，板卡 B_3 计算资源足够用于 B_1 和 B_2 任务的部署。因此，任务级多模冗余的效果与板卡的剩余资源量相关。理论上，若假设板卡资源无限，则每个任务都可以实现冗余分配，因此理论有效平均算力可表示为：

$$TAC = \sum_{i=1}^{NT} TR_i \times \left(1 - (1 - P_i)^k \right) \quad (19)$$

其中， k 表示冗余度，也就是任务被分配的次数， P_i 表示任务 T_i 完成的概率。

在实际应用场景中，板卡资源往往存在严格约束，若任务已满足板卡全部算力，则对任务实施 k 模冗余相当于将板卡资源划分为 k 份，即理论有效平均算力仅为：

$$TAC = \frac{\sum_{i=1}^{NB} BR_i}{k} \quad (20)$$

因此，任务级多模冗余的实施需要评估板卡资源约束情况，以确定合理的冗余度。

2.3 基于混合策略模拟退火的容错机制

检查点回滚机制虽能够提高有效算力，但在设置额外检查点时会产生开销，从而在一定条件下降低平均任务完成时间，其关键影响因素包括任务时长、检查点设置时间和故障率。任务级多模冗余机制在特定情况下也能提升有效算力，且可以降低平均任务完成时间，其主要受任务资源、任务完成概率和板卡资源的影响。

为解决上述难点，本文设计了基于增强型混合容错策略的模拟退火算法。虽然模拟退火是一种经典优化算法，但当其直接应用于本问题时存在明显局限：传统算法通常仅对任务调度序列进行优化，而本文需同时优化任务分配、冗余度配置、检查点间隔（checkpoint interval, CI）设置等多个异构决策变量，且需考虑故障恢复后的动态再分配。

该问题求解过程包含两个核心环节：一是为任务选择恰当的容错策略（如冗余度、检查点间隔等）；二是将

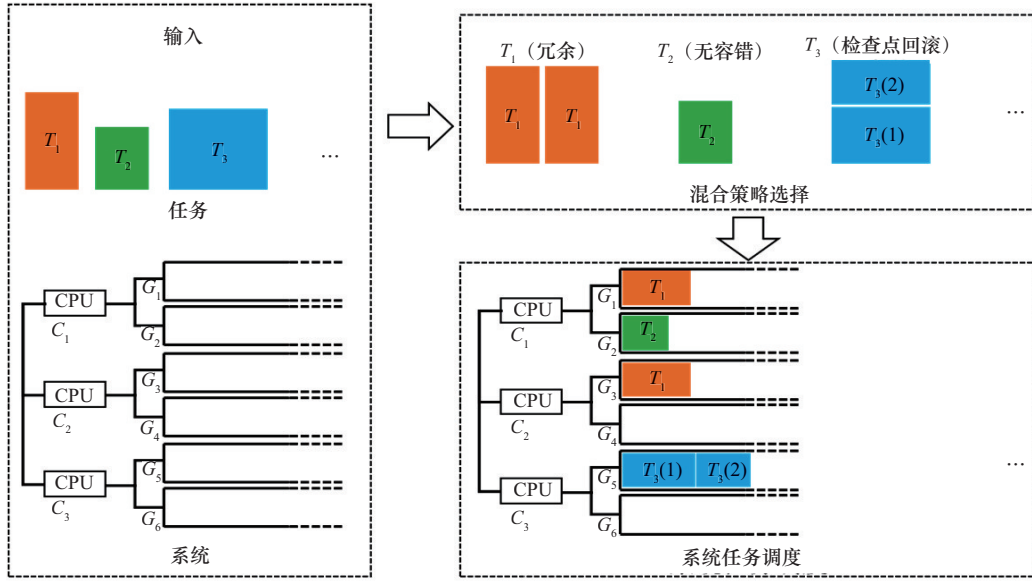


图5 混合容错策略问题描述

任务按序调度至合适的计算节点执行。本文所研究的混合容错策略问题描述如图5所示,采用由3块CPU组成的集群,且每块CPU挂载2个GPU,从而构成典型的异构计算环境。

(1) 三维解空间重构:突破传统调度算法仅考虑“板卡-任务”二维分配的局限,将解空间扩展为“板卡×任务×时间”的三维结构,如图6所示,并在其中对任务执行序列、冗余策略(k 值)和检查点间隔进行编码,实现多维决策的统一表示。

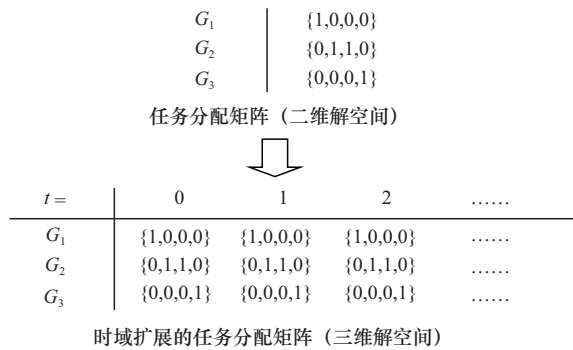


图6 解空间构建图

基于以上思考,设解为: $x = \{ \text{tasklists}, \text{CIs} \}$ 。

$\text{tasklists} = \{ \text{tasklist}_1, \dots, \text{tasklist}_{\text{NB}} \}, \text{CIs} = \text{CI}_1, \dots, \text{CI}_{\text{NT}}$

其中, tasklist_i 表示在板卡 B_i 上执行的任务列表, CI_j 表示任务 T_j 的检查点间隔时间。

(2) 复合邻域操作设计: 设计了多粒度邻域搜索操

作,包括任务在板卡间的移动、任务的复制(增加冗余)、任务的删除(减少冗余)以及检查点间隔的动态调整。这种复合邻域操作使算法能够在冗余、检查点、迁移三大容错机制间自由切换,实现解空间的全域探索。

(3) 理论驱动的能量函数: 采用“理论有效平均算力”作为能量函数,该函数融合了任务完成概率、检查点开销和资源约束,能够更准确地预测解的长期性能,有效避免单次仿真随机性导致的评价偏差,其表达式为:

$$f(x) = \text{TAC} = \frac{\sum_{i=1}^{\text{NT}} \text{TC}_i \times P_i}{T_{\text{theo_total}}} \quad (21)$$

其中, P_i 表示任务完成的概率,由以下公式计算可得:

$$P_i = 1 - \left(1 - (1 - \lambda)^{\text{TT}_i} \right)^{k_i} \quad (22)$$

其中, k_i 为任务 T_i 的冗余度,也就是任务被分配的板卡数量。

(4) 时间敏感的动态再分配机制: 当板卡发生故障时,若其上运行的任务已完成任意检查点状态存储,则继续在该板卡上执行,否则放入待分配任务列表。随后依次判断待分配列表中的任务在不同板卡的执行耗时,并将任务分配至执行耗时最短的板卡上。

该算法的求解过程如算法1所示。

算法1 模拟退火求解算法

输入:

任务集合 $T = \{T_1, T_2, \dots, T_{NT}\}$, 包含资源需求 TR_i 、
执行时间 TT_i 、数据量 TS_i

计算节点集合 $G = \{G_1, \dots, G_{NG}\}$, 包含算力资源 GR_i

故障参数: λ_G 、 λ_C 、 GRT_0 、 CRT_0

通信参数: BW_{local} 、 BW_{inter}

检查点参数: CT_0

模拟退火参数: $initial_temperature$ 、 $cooling_rate$ 、
 $stopping_temperature$

输出: 最优解 x_{best}

1: 基于输入参数生成满足所有约束条件的初始解 x_0

2: 使用式(21)和故障参数计算初始解 x_0 的目标函数
 $f(x_0)$

3: 最优解 $x_{best} \leftarrow x_0$, 当前解 $x_{current} \leftarrow x_0$, $tempera-$
 $ture \leftarrow initial_temperature$

4: **while** $temperature > stopping_temperature$ **do**

5: 生成邻域解 x_{new} : 在以下 4 种操作中随机选择
一种执行:

6: (1) 将随机任务移动到可能的计算节点 (考
虑 GR_i 和 TR_i 资源约束)

7: (2) 将随机任务复制到可能的计算节点 (考
虑冗余度约束)

8: (3) 将随机任务从重复的计算节点中删除

9: (4) 修改随机任务的检查点间隔 (基于 CT_0
和故障率)

10: 使用式(21)和故障参数计算邻域解 x_{new} 的目标
函数值 $f(x_{new})$

11: **if** $f(x_{new}) > f(x_{current})$ **then**

12: 接受新解 $x_{current} \leftarrow x_{new}$

13: **else**

14: 以概率 P 接受新解:

15: $P = \exp\left(\left(f(x_{new}) - f(x_{current})\right) / temperature\right)$

16: **end if**

17: 更新全局最优解:

18: $x_{best} \leftarrow f(x_{current}) > f(x_{best}) ? x_{current} : x_{best}$

19: 降温: $temperature \leftarrow temperature \times cooling_rate$

20: **end while**

21: **return** x_{best}

通过算法 1, 可以得到最优解 x_{best} , 对该算法的模拟
及后续执行过程如算法 2 所示。

算法 2 模拟退火执行与再分配算法

输入:

任务集合 $T = \{T_1, T_2, \dots, T_{NT}\}$, 包含资源需求 TR_i 、
执行时间 TT_i 、数据量 TS_i

计算节点集合 $G = \{G_1, \dots, G_{NG}\}$, 包含算力资源 GR_i

故障参数: λ_G 、 λ_C 、 GRT_0 、 CRT_0

通信参数: BW_{local} 、 BW_{inter}

检查点参数: CT_0

算法 1 输出的最优解 x_{best}

初始时间 $time = 0$

输出: 应用模拟退火算法后的实际平均算力

1: 根据最优解 x_{best} 中的 $tasklists$ 和 CIs , 初始化任务
分配

2: $time = 0$

3: **repeat**

4: // 执行当前时间片的任务

5: **for** G_i in G **do**

6: **if** G_i 正处于故障恢复状态 **then**

7: 恢复倒计时-1, 基于 GRT_0 或 CRT_0 判
断 G_i 是否恢复工作

8: **else**

9: **for** T_j 在 G_i 上执行的任务 **do**

10: 如果 T_j 正在设置检查点, 则继续执行
设置, 直到完成 (耗时累计 CT_0)

11: 否则任务倒计时-1, 若倒计时归 0, 则
完成任务, 释放资源

12: **end for**

13: **end if**

14: **end for**

15: // 故障检测与处理

16: **for** G_i in G **do**

17: 根据故障率 λ_G 、 λ_C 判断 G_i 是否发生故障

18: 若发生故障, 则进入故障恢复状态, 恢复时间
设为 GRT_0 或 CRT_0

19: 对于故障板卡中正在执行的任务, 若已完成检
查点状态存储则继续执行, 否则重新分配

20: **end for**

21: // 重新分配任务

22: 调用任务再分配函数, 结合 TR_i 、 GR_i 、
 BW_{local} 、 BW_{inter} 重新分配任务

23: $time = time + 1$

24: **until** 所有任务均已完成

25: 根据式(11)计算 AAC

26: return AAC

3 实验仿真模拟

为评估容错机制在动态故障环境下的实际效能, 本文设计了蒙特卡洛概率仿真器, 以模拟任务在异构板卡上的执行过程。具体来说, 本文通过二维数组表征任务到计算资源的映射关系, 并随机生成各任务的资源需求、执行时间及基础算力消耗, 模拟真实调度与故障扰动场景。仿真实验聚焦不同容错策略对系统性能的相对影响, 实际平均算力为无量纲指标, 通过对比所提算法与基准策略的相对性能, 验证其在提升系统鲁棒性方面的有效性。

3.1 约束条件和核心指标

本实验设立以下约束条件实现模拟过程。

(1) 资源约束: 即每个 GPU 节点的分配资源总和不能超过其总资源, 即:

$$\sum_{j=1}^{NT} TR_j \cdot x_{ij} \leq GR_p, \forall i \in \{1, 2, \dots, NG\} \quad (23)$$

(2) 故障约束: 当计算节点发生故障时, 进入重启状态, 当重启完成后, 才能重新执行故障发生时的任务。

(3) 任务完成约束: 所有任务均需要完成。

为了评价容错性能, 本文采用 AAC 作为评判标准, 计算式同式(11)。

3.2 模拟算法与模拟过程

为评估所提混合容错算法在天基计算环境下的性能, 本文基于蒙特卡洛方法构建仿真场景, 模拟低轨卫星上的异构计算系统。该系统由 3 个 CPU 节点及其挂载的 6 个 GPU 加速器组成, 形成“CPU-GPU”协同架构, 且任务通过 PCIe 总线传输并由 GPU 执行。

仿真重点考察不同容错策略在动态故障环境下, 对系统实际平均算力和任务完成时间的影响。实验中, 算法随机生成任务资源需求与执行时间, 初始化任务分配后进入时间步进循环, 具体流程如下。

(1) 更新正在执行或恢复中的任务状态。

(2) 根据故障率判断节点是否发生故障。

(3) 若故障发生, 则触发恢复机制, 并将未完成任务重新调度。

(4) 通过再分配函数优化任务执行路径。

仿真过程持续至所有任务执行完成, 最终以总计算量与总耗时的比值计算 AAC。本实验以不同容错策略为变量, 其他参数均保持固定, 具体设置见表 1。

表 1 检查点回滚模拟实验参数

参数/变量名	含义	值	单位
NT	任务的总数量	20	个
NG	GPU 节点的总数量	6	个
NC	CPU 节点的总数量	3	个
$TR_{\min}$	任务的最小所需资源量	2	GB
$TR_{\max}$	任务的最大所需资源量	4	GB
$TT_{\min}$	任务的最小所需时间	100	s
$TR_{\max}$	任务的最大所需时间	1 000	s
GR	每张板卡的可用计算资源量	8	GB
$\lambda_G \in [0,1)$	GPU 节点的故障率	0.8%	—
$GST_0 \in R^+$	GPU 节点的故障恢复时间	100	s
$\lambda_C \in [0,1)$	CPU 节点的故障率	0.2%	—
$CST_0 \in R^+$	CPU 节点的故障恢复时间	100	s
BW_{local}	传输速率(CPU 与 GPU、GPU 间)	1 000	Gbit/s
BW_{inter}	传输速率(CPU、数据源之间)	10	Gbit/s
CI_0	任务检查点间隔时间	变量	s
CT_0	任务检查点设置时间消耗	50	s
k	冗余算法的冗余度	2	—

3.3 容错机制模拟测试

在以上模拟器的基础上, 可以通过对分配函数、故障执行函数、再分配函数等模块进行修改, 从而适配不同场景, 对容错模型进行进一步分析, 并通过蒙特卡洛模拟实验验证模型的真实应用情况。

(1) 检查点回滚机制

为了降低故障概率导致的偶然因素影响, 所有实验结果均取 10 次独立测试的平均值, 本文简单算法 (simple) 指未引入任何额外容错机制的算法, 即无检查点回滚、任务级多模冗余等容错策略的基线算法版本, 算法测试结果如图 7 所示。

由实验结果可以看出, 检查点回滚机制具有较优容错效果, 可以避免因任务重新计算产生的额外计算量。当节点发生故障时, 系统可回滚至故障前最近的一个检查点继续计算任务, 有效提高实际平均算力, 但也会缩短平均任务完成时间。当检查点时间间隔设为 200 单位时间时, 实际平均算力可达 12.949; 当检查时间间隔设为 300 单位时间时, 平均任务完成时间最低可达 876.41 s。

(2) 任务级多模冗余容错机制

在算法 1 的基础上, 多模冗余算法仅需要对初始任务分配方式进行修改, 即按顺序遍历算法为板卡分配任务, 在满足资源充足、冗余度合规且任务未重复部署的条件下循环执行分配操作, 直至无可分配任务。本实验参数与表 1 相同, 设冗余度为变量 k 。共进行了 10 次独立实验, 算法测试结果分别如图 8、图 9 所示。

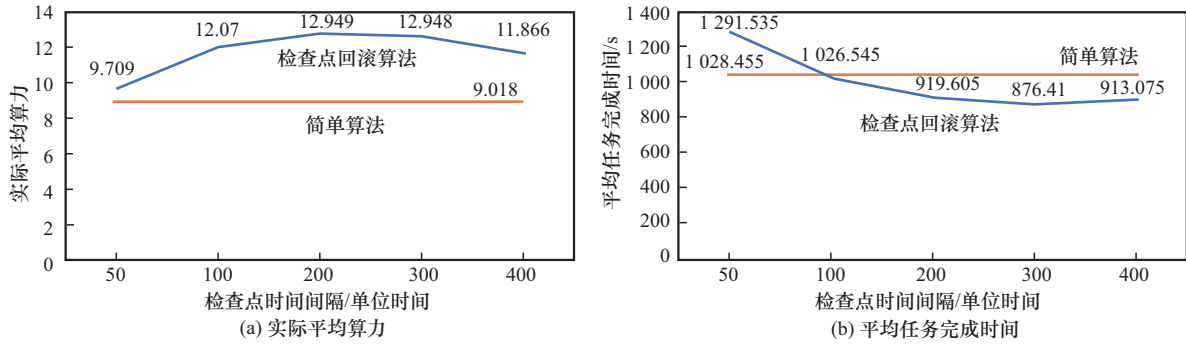


图7 检查点回滚算法实际平均算力与平均任务完成时间测试结果

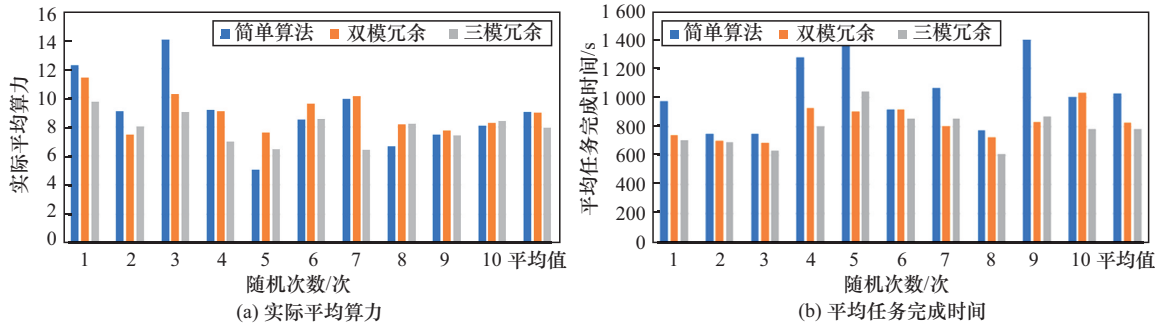


图8 不同算法实际平均算力与平均任务完成时间测试结果

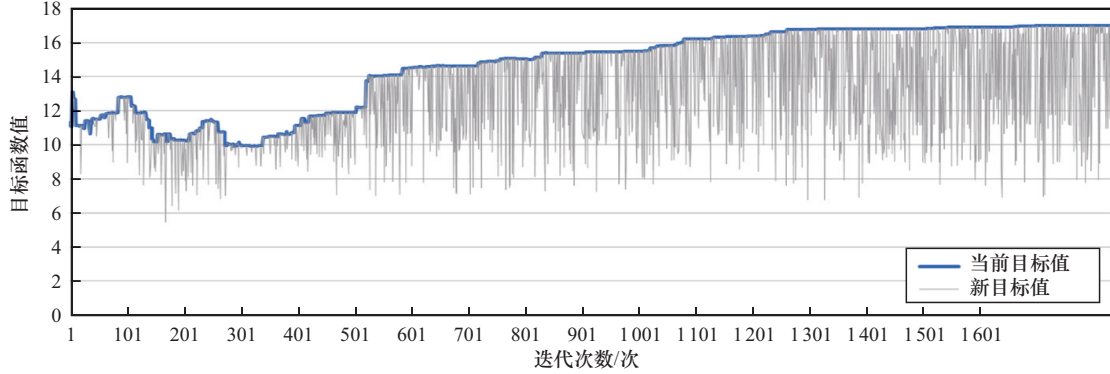


图9 目标函数值随迭代次数的变化曲线

图8展示了3种容错策略在10次独立实验中的性能对比,包括实际平均算力与平均任务完成时间。实验结果表明,简单算法虽具备最高理论算力,但因缺乏故障恢复机制,任务执行过程易受中断影响,导致任务完成时间波动大且延迟显著。相比之下,双模冗余策略与三模冗余策略理论算力虽略微降低,但通过提升执行可靠性,能够显著缩短平均任务完成时间并减小性能波动。这说明适度冗余可在资源开销与系统鲁棒性之间取得有效平衡,增强对动态故障的适应能力。

(3) 基于混合策略的模拟退火优化策略

为求解该优化问题,本文采用基于混合策略的模拟

退火优化策略。策略以随机生成的初始解为起点,通过迭代执行任务迁移、检查点调整、冗余复制等邻域操作进行搜索。图9展示了目标函数值随迭代次数的变化过程:迭代初期因温度较高,算法接受劣解的概率大,目标函数值波动明显;随着温度下降,搜索过程趋于收敛,目标函数值逐步提升,并在约1000次迭代后趋于稳定。

最终优化结果显示,在20个任务中,1个关键任务(T_{13})被配置冗余,其余任务均启用检查点机制,验证了算法对多种容错策略的协同优化能力。实验数据表明,该算法在1957 s内完成总计算量30 847.4 TFLOPS,实际平均算力达15.76,平均任务完成时间为1 127.75 s。

为验证基于混合策略的模拟退火优化策略的效果, 本文将其与其他算法进行对比实验模拟。实验参数同表1, 其中 $CI_0 = 100$ s。需说明的是, 每次邻域搜索操作均需要对能量函数进行计算; 而计算能量函数时需要计算各节点的任务耗时, 这一过程需要遍历每个节点上的任务, 因此算法的时间消耗为 $O(NT)$, 实际时间复杂度为:

$$O\left(NT \times NB \times \frac{\log \frac{\text{initial_temperature}}{\text{stopping_temperature}}}{\text{cooling_rate}}\right)$$

该算法执行时间为1.240 408 s, 处于可接收范围内。简单算法、检查点回滚算法和任务级多模冗余算法的时间复杂度均为线性复杂度 $O(NT \times NB)$ 。模拟退火算法的复杂度和迭代次数正相关, 虽具有最高理论复杂度, 但单次优化收敛时间仅为1.24 s, 满足星载系统秒级调度需求。

为进一步验证算法性能, 本文选取简单算法、固定时间间隔检查点回滚策略及双模冗余策略3种基准策略开展对比实验, 实验结果如图10所示。

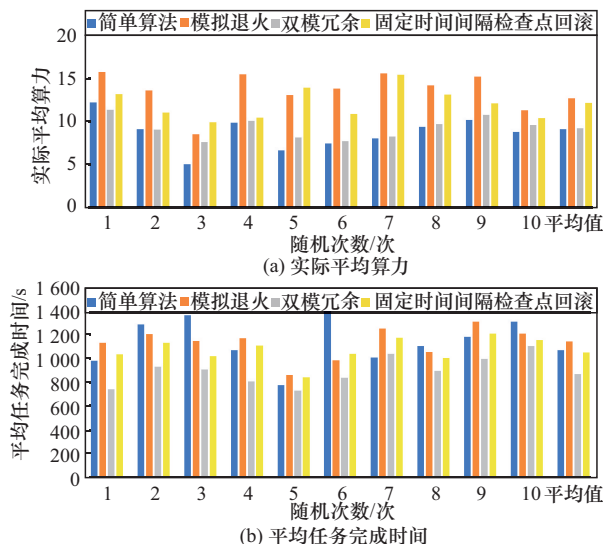


图10 实际平均算力与平均任务完成时间实验结果

图10展示了4种不同容错策略在10次独立随机实验中的性能对比, 包括实际平均算力和平均任务完成时间两个指标。从图10中可以看出, 本文提出的基于模拟退火的优化策略在综合算力表现上明显优于其他3种对比策略。具体而言, 在实际平均算力方面, 基于模拟退火的优化策略不仅整体数值最高, 而且波动较小, 展现出良好的稳定性。简单算法、基于模拟退火的优化策略、双模冗余策略、固定时间间隔检查点回滚策略的实际平均

算力平均值分别为: 9.126、12.703、9.208、12.163, 其中, 基于模拟退火的优化策略的实际平均算力值较简单算法提升了约39.20%。这得益于该策略能够智能调整任务分配、冗余度配置及检查点设置, 在计算效率与系统可靠性之间取得了最优平衡, 从而有效提升系统的整体算力利用率。尽管基于模拟退火的优化策略在有效算力方面表现优异, 但其平均任务完成时间相对偏长。如图10所示, 该策略的平均任务完成时间在多数实验中均高于对比方案, 主要源于在故障恢复机制的设计中, 任务发生故障后不立即重启, 而是返回调度池等待资源重新分配, 导致恢复时延累积。

4 结束语

本文研究了天基计算系统的容错问题。针对空间辐射导致器件易发生故障的问题, 提出了一种新型容错方法。该方法结合了任务冗余、检查点和任务迁移3种技术, 并采用模拟退火算法进行优化, 确保系统在故障场景下仍能高效运行。实验结果表明, 所提策略的性能优于传统简单算法、固定时间间隔检查点回滚策略、固定冗余策略等经典算法: 相比无容错机制的简单算法, 系统实际平均算力提升了约39.20%; 相比固定冗余策略或固定时间间隔检查点回滚策略, 资源利用更合理, 算力输出更高。这说明动态协同优化策略较单一固定策略更有效。需要说明的是, 本文策略对比的基准是星载系统常用的经典简单算法, 而对其他先进智能算法的比较尚不充分; 同时, 模型假设故障独立且恢复时间固定, 未充分考虑复杂耦合故障和老化效应, 且验证基于仿真实验, 尚未在真实星载平台部署, 后续将进一步拓展故障模型并开展在轨测试工作。

参考文献

- [1] Hiblot G, Serbulova K, Hellings G, et al. TCAD study of latch-up sensitivity to wafer thinning below 500 nm[C]//Proceedings of the 2021 International Semiconductor Conference (CAS). Piscataway: IEEE Press, 2021: 121-124.
- [2] 虞志刚, 冯旭, 戴天, 等. 空间边缘计算: 需求、架构及关键技术[J]. 电子与信息学报, 2022, 44(12): 4416-4425.
Yu Z G, Feng X, Dai T, et al. Space edge computing: requirement, architecture and key technique[J]. Journal of Electronics & Information Technology, 2022, 44(12): 4416-4425.
- [3] Rawlins F, Baker R, Martinovic I. Death by a thousand COTS: disrupting satellite communications using low earth orbit constellations[EB]. 2022.
- [4] Kobayashi D. Scaling trends of digital single-event effects: a sur-

vey of SEU and SET parameters and comparison with transistor performance[J]. IEEE Transactions on Nuclear Science, 2021, 68(2): 124-148.

- [5] 周嘉丹, 闫朝鑫, 姚钢, 等. 空间辐射环境对航天器分布式电力系统关键部件的影响及应对策略[J]. 电工技术学报, 2022, 37(6): 1365-1380.

Zhou L D, Yan C X, Yao G, et al. Influence of space radiation environment on critical components of spacecraft distributed power system and countermeasures[J]. Transactions of China Electrotechnical Society, 2022, 37(6): 1365-1380.

- [6] Zhang Y, Zheng Y M, Yang M, et al. Design and implementation of the highly-reliable, low-cost housekeeping system in the ZDPS-1A pico-satellite[J]. Journal of Zhejiang University SCIENCE C, 2012, 13(2): 83-89.

- [7] Roffe S, George A D. Evaluation of algorithm-based fault tolerance for machine learning and computer vision under neutron radiation[C]//Proceedings of the 2020 IEEE Aerospace Conference. Piscataway: IEEE Press, 2020: 1-9.

- [8] Lamport L, Shostak R, Pease M. The Byzantine generals problem [M]//Concurrency: The works of Leslie Lamport. Cambridge: MIT Press, 2019: 203-226.

- [9] Fayyaz M, Vladimirova T. Fault-tolerant distributed approach to satellite on-board computer design[C]//Proceedings of the 2014 IEEE Aerospace Conference. Piscataway: IEEE Press, 2014: 1-12.

- [12] 高吉星, 徐志平, 丁文慧, 等. 卫星智能计算平台自主管控架构设想[J]. 天地一体化信息网络, 2024, 5(4): 85-94.

Gao J X, Xu Z P, Ding W H, et al. Autonomous control architecture of satellite intelligent computing platform[J]. Space-Integrated-Ground Information Networks, 2024, 5(4): 85-94.

- [13] 郑宏超, 王亮, 李哲, 等. 面向设计加固的航天集成电路辐射效应评估方法研究与实践[J]. 核技术, 2023, 46(8): 080007.

Zheng H C, Wang L, Li Z, et al. Development and application of a radiation effect evaluation method of aerospace integrated circuits for radiation hardened by design[J]. Nuclear Techniques, 2023, 46(8): 080007.

- [14] Wang F X, Ramamritham K, Stankovic J A. Determining redundancy levels for fault tolerant real-time systems[J]. IEEE Transactions on Computers, 1995, 44(2): 292-301.

- [15] Yang M F, Hua G X, Feng Y J, et al. Fault-tolerance techniques for spacecraft control computers[M]. Hoboken: Wiley, 2017.

- [16] 魏晓辉, 童诗雨, 孙忠奥, 等. 为高性能计算系统构建一种程序错误弹性感知的检查点机制[J]. 计算机科学技术学报, 2025, 40(3): 671-685.

Wei X H, TONG S Y, Sun Z A, et al. Building an error-resilient checkpointing mechanism aware of program errors for high-performance computing systems[J]. Journal of Computer Science and Technology, 2025, 40(3): 671-685.

- [17] Liu Y Q, Han Y H, Li H X, et al. Computing over space: status, challenges, and opportunities[J]. Engineering, 2025, 54: 20-25.

[作者简介]



史冰冰 (2001-), 女, 上海科技大学硕士生、中国科学院计算技术研究所客座学生, 主要研究方向为天基容错计算。



李泓辛 (2000-), 男, 硕士, 北京中科天算有限公司战略发展部部长, 主要研究方向为天基容错计算。



刘垚圻 (1995-), 男, 博士, 中国科学院计算技术研究所副研究员, 主要研究方向为计算机体系结构、天基容错计算、通导遥算一体化。



周浩 (1997-), 男, 中国科学院计算技术研究所博士生, 主要研究方向为天基计算系统设计及应用。



谭海宁 (1991-), 男, 博士, 中国科学院计算技术研究所副研究员, 主要研究方向为计算机体系结构和智能计算系统优化。



韩银和 (1980-), 男, 博士, 中国科学院计算技术研究所研究员, 主要研究方向为计算机体系结构和芯片、智能机器人、智能硬件、数据计算芯片和计算系统。