doi:10.11887/j.cn.202006010 http://journal.nudt.edu.cn

超算环境科学工作流应用的容错*

李于锋[1,2],莫则尧[2],肖永浩[1],赵士操[1],段博文[1]
(1. 中国工程物理研究院 计算机应用研究所,四川 绵阳 621900;2. 北京应用物理与计算数学研究所,北京 100094)

摘 要:超算环境中科学工作流技术广泛应用于科学研究和工程仿真领域。复杂多物理过程数值模拟、多阶段数据处理等应用往往需要使用多种应用软件相互协作,构建业务流程自动执行来提升工作效率。然而在超算环境中执行科学工作流应用面临着资源失效、任务配置错误等异常情况,造成工作流执行中断,严重影响完成效率,故容错功能对超算工作流应用的稳定持续运行有重要意义。介绍了科学工作流的容错设计分类,并对典型工作流系统的容错设计进行分析评述;提出了基于决策树的事件-条件-动作容错模型,设计了非侵入式可扩展的容错架构,并针对自主研发的部署在超算环境下的科学工作流应用平台HSWAP,实现了运行时可配置的容错策略。在实际的工程仿真任务中,基于所提出模型和架构实现的容错机制为提高工作流执行效率发挥了重要作用。

关键词:容错;科学工作流;决策树模型;工作流引擎
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2020)06-082-08

Fault tolerance in HPC scientific workflow application

LI Yufeng[1,2], *MO Zeyao*[2], *XIAO Yonghao*[1], *ZHAO Shicao*[1], *DUAN Bowen*[1]
(1. Institute of Computer Application, Chinese Academy of Engineering Physics, Mianyang 621900, China;
2. Institute of Applied Physics and Computational Mathematics, Beijing 100094, China)

Abstract: Scientific workflow technologies in HPC are extensively applied in scientific research and engineering simulation domain. Application such as numerical simulation in complex multi-physics problems and multi-stages data process need software to compose an automatic executable workflow to increase the efficiency. There are lots of exceptions such as resource failure, task configurations errors which may cause the workflow execution to be ceased, therefore robust and continuous execution is important for workflow application. A taxonomy of fault tolerance in workflow was made and some fault tolerance techniques in typical workflow systems were reviewed. A decision-tree based event-condition-action fault tolerance model was proposed, and a non-intrusive extendable framework which was implemented in our HPC scientific workflow system HSWAP was designed. Runtime configurable error recovery strategies were also implemented in our fault tolerance software module. In order to validate our new model and framework, the fault tolerance functions were tested in real engineering simulation project. Results show that fault tolerance plays an important role in increasing workflow execution efficiency.

Keywords: fault tolerance; scientific workflow; decision tree model; workflow engine

在科学发现和工程仿真中,使用一系列相关软件完成数据收集、建模、模拟、分析成为普遍现象。各步骤间可能有数据依赖或控制依赖关系,这些软件相互协作才能获得最终结果。科学工作流管理系统对这些软件及其数据依赖关系进行组合,并控制各部分在时间、空间以及资源等约束条件下按序完成,已经成为复杂科学计算流程管理的必要手段,有效推动了科研进展[1]。面向大规模数据集和复杂计算流程的超算环境科学工作流技术提供了自动化、流程化的方法,并为用户屏蔽作业投递和数据传递转换等细节,极大地促进了超算环境和应用平台的协同发展。

在超算环境中,资源失效是常见现象,如计算结点死机或网络模块出现故障等问题造成无法正常工作等。据估计E级计算的平均无故障时间(Mean Time Between Failure, MTBF)已缩短为数分钟[2]。资源失效会造成任务和流程中断执行。另外,在复杂流程(通常表现为稠密大规模有向图)中组件数量大、配置烦琐,对系统环境变量及运行参数有严格要求,用户使用易出现

* **收稿日期**:2019-09-21
基金项目:国家重点研发计划资助项目(2018YFB0703903)
作者简介:李于锋(1982—),男,河南光山人,博士研究生,E-mail:liyf@caep.cn;
莫则尧(通信作者),男,研究员,博士,博士生导师,E-mail:zeyao_mo@iapcm.ac.cn

配置错误导致软件组件运行异常,也同样造成任务运行失败和流程中断执行。如何对任务运行失败进行预防或者失败后进行自动恢复,以保证流程整体自动持续运行,是工作流容错研究的重点问题。

容错作为科学工作流管理系统的重要组成部分,随着科学工作流运行环境从单机、集群、异构多集群到云的变迁,涌现出多种容错策略,在不同的运行模式下,为工作流平台的稳定运行提供了有力保障。特别是对于长时间运行的科学工作流,或者数据处理步骤繁多的流程,提供错误自动恢复机制尤为重要。

国内外已经出现了很多工作流系统平台,这些工作流系统都提供了容错相关的功能。比如美国加州大学研究者基于 Ptolemy II[3] 系统开发的 Kepler[4] 系统,广泛应用于包括生态学、分子生物学、化学、计算机科学、电子工程和海洋学等领域;美国南加州大学信息科学研究所开发的系统 Pegasus[5],已应用于地震及灾害模拟,引力波探索(Laser Interferometer Gravitational-wave-Observatory, LIGO)等领域;英国曼切斯特大学计算机科学学院开发的 Taverna[6] 工作流套件,广泛应用于生命科学等领域;英国加地夫大学开发的 Triana[7] 网格集成计算环境,可用于天文学计算和网络数据处理等领域。这些工作流管理系统的容错功能将在本文第二部分进行阐述。

为了支持超算环境下科学和工程计算领域数值模拟流程的高效设计与自动执行,中国工程物理研究院计算机应用研究所自主研发了 HSWAP 工作流应用平台[8-9],以高性能应用软件作为组件封装的基本单位,配置简单,部署便捷,支持跨平台应用的协作。本文设计的容错模型和方法将以 HSWAP 系统为验证平台,基于日志分析,利用数据驱动的方式实现了容错功能,为错误自动恢复、工作流无间断运行提供了重要支撑。

1　科学工作流容错的分类

工作流系统的容错分为任务级和工作流级[10-11],其中任务级又可分为任务重试、检查点/重启动、替换资源、多副本运行等,工作流级可分为替换任务、冗余多任务、用户定义的异常处理、基于修复工作流等。具体分类如图 1 所示。任务重试是在任务出错后,简单地重新执行该任务,通常会设置一个最大重试次数,即超过该重试次数后若还没有恢复成功,则放弃重试。检查点重启动机制[12]需要任务本身支持检查点重启动功能,在任务运行过程中进行一定间隔的检查点信息保存,在任务异常出错后能够从最近的检查点恢复执行。替换资源方式是指在任务运行失败后,切换到其他资源继续运行该任务,以应对资源失效造成的异常。任务多副本运行是指同一个任务(同一实现)在多种资源上同时运行,确保至少有一个运行成功,这是一种预防失效的方式。工作流级容错的替换任务方式是指任务运行失败后,会执行任务的另一种实现,该实现同样完成原来任务的目标;冗余多任务是指会同时运行多个替换任务,也是预防失效的方式;用户定义的异常处理机制通常是在组件级别对出错情况进行处理。基于修复工作流是指工作流执行过程中异常发生后记录失败的任务信息,生成新的工作流以备后续提交重新运行。

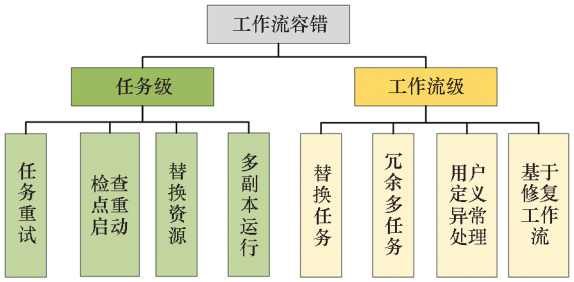


图 1　工作流容错的分类

Fig. 1　Taxonomy of fault tolerance

面向基于数据流的科学工作流管理系统,Ustun 等[13] 提出了多数据实例相关的容错框架。首先识别出数据处理管线(pipeline)中的错误模式,提出的恢复方法包括容许限定次数的失败项、以哑数据代替因错误数据而不能执行的任务输出、立即重执行错误任务(需考虑数据实例依赖关系)、依赖工作流实例的恢复(考虑中间数据的存储代价和生成代价)等。另外,文献中利用编程语言模型中的异常处理等技术[14]或研发新的异常处理语言[15]给容错研究带来了新技术路线,但由于和原工作流系统的紧耦合性,不易推广。

2　典型工作流系统的容错设计

Taverna 科学工作流管理系统中的容错采用了任务重试和替换任务两种策略,任务重试中工作流设计者可定义最大重试次数,该策略也可应用于子工作流容错;替换任务允许在任务重试达到最大次数后选择执行一个不同的任务。

Triana 工作流系统的容错是面向用户的,如

错误发生时,会产生警告信息,并允许用户修改后继续执行;在工作流级支持轻量级的检查点/重启动和工作流服务的选择。在中间件级和任务级,所有预定义的异常会被 Triana 引擎感知(死锁、活锁和内存泄露除外);在最底层的资源失效方面,借助 GridLab GAT 工具可以识别资源失效,但是错误恢复机制还未完善。

Pegasus 的容错是基于 DAGMan 和 HTCondor 开展的。例如,在作业运行错误后进行重试或重新提交处理,通常可以在作业文件中设置作业重试次数。针对数据传输的可靠性问题,Pegasus 传输服务会首先尝试高性能的并行传输,失败后会进行更安全的低速单连接传输。如果重试次数已达最大,将生成修复工作流待后续重新运行。另外,Pegasus 还支持进行重新资源规划以重用失败时已生成的数据,并通过调度任务到不同资源来实现容错。

Kepler 科学工作流系统中的容错是通过一个称为 Checkpoint 的复合 actor 实现的。actor 是 Kepler 中的专有概念,表示一个执行组件。检查点复合组件包括一个子工作流和若干可替换的子工作流,当错误发生后,发出错误时间信息,检查点内所有工作流停止执行并处理错误,决定是否重运行该子工作流或者运行一个可替换的具有同样功能的子工作流。

已有工作流系统的容错存在的问题有:容错生命周期管理不健全,如 Taverna、Triana 等;容错处理与执行任务紧耦合,扩展方式不灵活,如 Kepler、Taverna 等;没有独立的容错机制,依赖其他(网格基础设施提供的)资源管理工具实现,如 Pegasus、Triana 等。

容错的设计需要考虑可灵活配置的错误恢复策略、与工作流执行引擎的解耦模型、容错处理的全生命周期管理等问题。如此设计方可将容错服务功能方便集成到多种科学工作流系统。HSWAP 是面向高性能计算领域的工作流系统平台,其以封装数值模拟软件和应用形成松耦合、粗粒度的工作流为特色,下文提出的容错模型和方法设计在该类平台上的实现十分便捷。

3 容错模型和方法设计

3.1 容错全生命周期管理

工作流执行异常出现后,对异常的发生、消息的流转以及针对异常的处理进行全面的分析,有助于给出全面系统的容错方案。容错的生命周期如图 2 所示,首先是错误发生,生成错误事件,然后错误被监控工具识别监测,最后进入错误处理流程,包括错误恢复策略以及相关的恢复动作。

图 2 容错全生命周期管理

Fig. 2 Full period management of fault tolerance

为支持容错,工作流系统在设计时应考虑异常状态管理以及错误信息的传递,比如错误发生后,流程执行中断进入待恢复状态,还要考虑错误事件的描述,事件消息的发出和相应的消息跟踪。容错处理流程的过程如图 3 所示,包括异常信息监测、错误信息识别、错误信息分类、错误恢复处理四种主要的功能步骤。

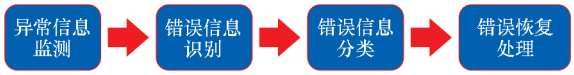


图 3 容错处理流程

Fig. 3 Fault handling process

异常信息监测是指异常事件消息发出后,容错模块能够感知和探测到异常;错误信息识别则是从工作流系统日志等记录中有效识别出错误信息;错误信息分类则是为了进一步的容错处理缩小处理方式范围;错误恢复处理则是采取相应的执行动作,以使得工作流从中断暂停状态转移至继续执行或等待用户救援状态。

3.2 基于决策树的 ECA 容错模型

异常处理可以采用基于事件 - 条件 - 动作(Event-Condition-Action, ECA)的模型[10]构建。ECA 模型如图 4 所示,针对不同的错误事件,在不同的条件下执行相应的动作。

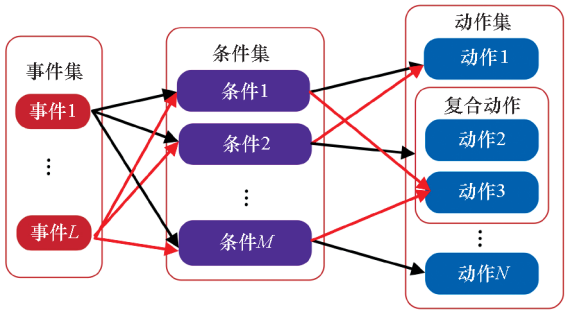


图 4 ECA 模型

Fig. 4 ECA model

在 ECA 模型中,事件集表示所有发生的事件,每类事件都有一个明确的标识,表明相应的

信息,比如任务运行失败、网络服务超时等异常信息。条件集里的条件表明当前工作流系统的运行状态和运行环境,比如工作流引擎是否停止响应、资源是否可用等。动作集里的动作表明实现错误恢复需要执行的命令,可以是单个命令,也可以是多步骤的多条命令组合成的复合动作。ECA 模型具有很好的灵活性,针对同一事件,对应不同的条件下,则可执行不同的动作指令;在相同条件下,不同事件也可对应不同的动作指令。

在异常事件和环境条件可监测、可探察的前提下,灵活配置错误恢复动作是 ECA 模型的特色。本文扩展了简单的 ECA 模型,使得事件发生后,可以结合多个条件来最终选择相应的动作执行。具体地,引入决策树算法,基于决策树高效实现事件 - 条件集合到动作集合的映射。基于决策树的错误恢复动作选择示例如图 5所示。

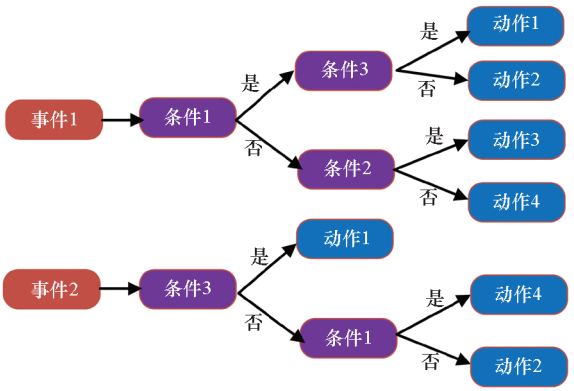


图 5　基于决策树的错误恢复

Fig. 5　Failure recovery based on decision tree model

基于决策树模型设计的错误恢复针对每个事件都有一颗不同的决策树。允许不同的事件有不同的处理逻辑,比如图 5 中事件 1 的处理逻辑必须判断两个条件才能给出最终的恢复动作,而事件 2 在条件 3 满足的前提下只需判断一个条件就可以决定恢复动作。另外,不同事件的决策顺序也不相同,图 5 中事件 1 需要先判断条件 1,然后判断条件 2 或条件 3;而事件 2 则先判断条件 3,然后视结果可能判断条件 1。如此设计有助于减少环境探查,快速决定错误恢复动作。具体的执行动作决策如算法 1 所示。

算法 1 中决策变量是指某一个条件的布尔值,即对任一条件,都有两个状态表示是否满足。比如计算结点网络故障不可达(无法连接)、计算结点可连接两个状态,每个状态下都有后续不同的处理逻辑。

算法 1　容错动作决策

Alg. 1　Decide recovery actions

输入:ECA 规则,异常事件

输出:处理动作 action

```
1. 加载 ECA 规则
2. cond_list = 事件的发生环境或条件集
3. curr_decision = 当前决策点
4. While (curr_decision 无法决定动作){
5.     cond, other = 当前决策点的下一步决策变量
6.     If cond in cond_list:
7.     curr_decision = 在当前决策点引入 cond 后进入下
       一级决策点
8.     Else:
9.     curr_decision = 在当前决策点引入 other 后进入下
       一级决策点
10. }
11. action = curr_decision 决策点确定的动作
```

3.3　错误信息和恢复动作

针对超算环境科学工作流系统的执行特征,设计了错误事件信息的要素,如图 6 所示。错误信息包括错误信息的发出位置、出错模块、引入环境、产生时间、发生频率、严重等级、造成影响等七个维度。

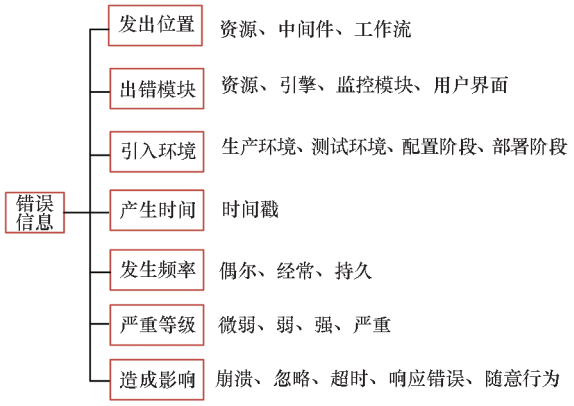


图 6　错误信息要素

Fig. 6　Elements of fault information

虽然错误信息包含的要素较多,但在实际工作流执行过程中能够获得的信息却是有限的,需要建立日志系统来存储和分析历史错误信息,补充相对完整的错误信息,方便后续的错误恢复策略选择决策。

容错设计考虑了在发生异常事件后可配置不同的处理方式。支持的处理方式如图 7 所示,有重试、替换、重启动、错误传播、忽略、标注和用户介入七类。重试又包括任务重试和子流程重试,任务重试表示仅重运行出错任务,子流

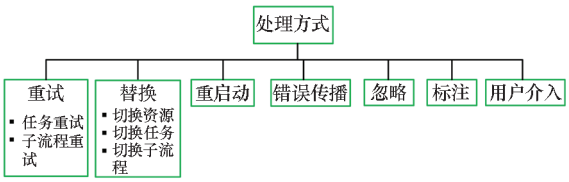


图 7 错误恢复处理方式

Fig. 7 Failure recovery methods

程重试则重运行出错任务所在的子流程(工作流的一个子结构)。替换类的处理包括替换资源、替换任务和替换子流程。替换资源表示同一任务在不同的资源上重运行;替换任务表示选择执行另一个具有同样功能但不同实现的任务;替换子工作流则表示选择执行另一个具有相同功能的子工作流执行。重启动(或称检查点 - 重启动)方式一般需要任务实现支持,在任务运行过程中,会以一定间隔保留后续恢复时所需的数据信息(检查点信息记录),在错误恢复时,会从检查点处重新加载信息继续执行而不会从头开始计算。错误传播是指将错误信息从出错任务所在的执行模块传递到工作流系统引擎、用户界面以及日志系统(将来可能通过网络远程传输给客户端或其他系统)。对于一些不影响流程主要功能完成度的异常信息,可采取忽略的处理措施。对于未知异常,可采取标注的方式记录异常出现的场景及其造成的影响等信息。当任务配置参数出现错误引发异常时,通常需要用户介入,以用户的专业知识修正任务配置参数,才能够达到错误恢复的目的,这可通过科学工作流的"人在回路"(human in the loop)等技术实现。

3.4 容错与科学工作流系统的关系

异常信息和处理方法、处理策略设计好之后,应该考虑如何将容错与科学工作流系统进行有效融合,既能有效发挥出容错的重要作用,又不影响原有工作流系统的设计。本文提出如图 8 所示的设计架构来确定各子系统间的关系。容错服务和科学工作流引擎独立开发和部署,基于远程过程调用(Remote Procedure Call,RPC)和日志服务及消息来进行交互。

异常事件发生后,由监控模块探测到错误信息,生成错误事件传递给科学工作流引擎,进而形成错误事件信息,并将该信息发送至日志服务器(可基于 Elastic Search-Logstash 构建)。容错服务会间隔轮询日志服务获得所有出错事件,并用预先配置好的错误恢复准则进行错误分析,然后

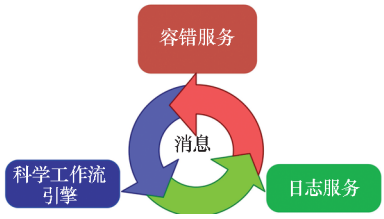


图 8 容错与科学工作流系统的关系

Fig. 8 Fault tolerance architecture in HSWAP

触发错误恢复动作,完成错误恢复。

如此的容错架构设计有三方面的优点:一是模块化,将容错服务与工作流管理系统和日志系统解耦,实现了松耦合以及容错模块可插拔、可替换的目的;二是服务化,将所有错误统一集中管理,并实现了高并发的处理逻辑,可同时处理不同用户、不同工作流实例的错误恢复;三是单向消息机制,数据传输代价小、效率高、逻辑清晰,在实际系统开发中简单实用。

4 容错在 HSWAP 系统的实现与验证

4.1 HSWAP 简介

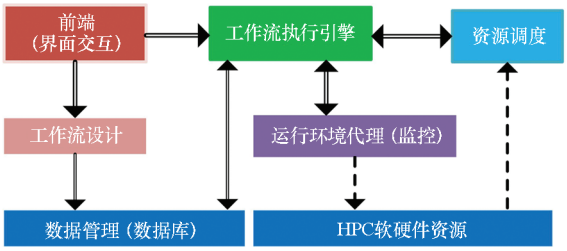
HSWAP 是中国工程物理研究院计算机应用研究所开发的超算连贯计算引擎[8-9],旨在 HPC 环境中使用科学工作流技术提供集成的超算服务模式助力科研人员提高工作效率。基于 HSWAP 开发的石油地震勘探平台以及材料高通量计算平台等行业计算平台,已在实际项目中得到应用并发挥了重要作用。

HSWAP 的主要特色是为超算用户屏蔽使用超算系统的复杂性,以计算软件为基本封装单位形成可复用组件,进而实现灵活可定制业务流程的功能。流程以有向无环图(Directed Acyclic Graph, DAG)表达,实现流程中结点间依赖管理和数据自动传递和转换功能。平台的架构和相关模型如图 9 所示。

4.2 HSWAP 的容错实现和验证

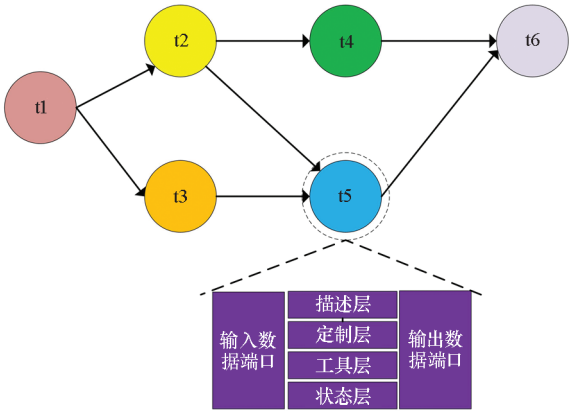
HSWAP 平台提供了日志系统,这为实现上文提出的容错架构提供了方便。引擎执行过程中,会通过 Logstash 服务将运行时相关信息写入 Elastic Search 数据库,相关信息在容错模块可以用来进行容错动作决策。

在 HSWAP 中提供了基于数据完整性校验的两类错误识别和恢复方法。在超算系统中,任务作业退出后,是否正常完成任务目标需要多方面的考虑,其中数据完整性是最常见的判别标准之一,特别是对于数值模拟仿真类任务,生成完整而



(a) HSWAP 的模块架构

(a) Architecture of HSWAP



(b) HSWAP 的流程和组件模型

(b) Workflow and component models of HSWAP

图 9　HSWAP 平台的架构和业务模型

Fig. 9　Architecture and models of HSWAP

正确的数据文件几乎是唯一的标准。基于数据判别的容错流程如图 10 所示,分为三个步骤,即出错标识、信息收集、错误恢复。

图 10　基于数据完整性校验的容错流程

Fig. 10　Fault tolerance based on integrity of data

基于数据完整性校验的出错标识在 HSWAP 工作流引擎中完成,主要是利用工作流组件的自定义配置功能,对不同的任务配置不同的数据完整性校验策略,比如数据体量、数据结束标识监测等,当数据完整性不达标时,发出错误信息。日志信息会收集所有任务运行上下文信息,以便容错模块对错误进行分析定位。容错服务根据出错事件,基于可配置的 ECA 规则(如图 11 所示),执行错误恢复逻辑,最后通过 HSWAP 引擎接口调用自动恢复流程执行。

面向单个数值模拟任务的检查点 - 重启动错误恢复方式在 HSWAP 的实现流程如图 12 所示。流程执行监控、结果文件正确性校验、重启动参数配置和输入文件准备、重投递运行等一系列过程自动化执行,无须人工干预。实际使用中结合重

```
### ECA.yml
### event: {condition: [actions], not_condition: [actions]}
### nested "if-else" binary tree syntax
### when parsing app's failure conditions,
### check xxx_failure first, then the other
run_failure:
  resource_failure:
    - alternate_resource
    - task_retry
  resource_success:
    process_failure:
      - task_retry
    process_success:
      data_failure:
        - task_restart
      data_success:
        - task_retry

data_failure:
    - task_restart
...
```

图 11　错误恢复策略配置文件示例

Fig. 11　Example for fault tolerance strategy

试、重启动两种方式,解决了资源不稳定等问题引起的执行中断问题。

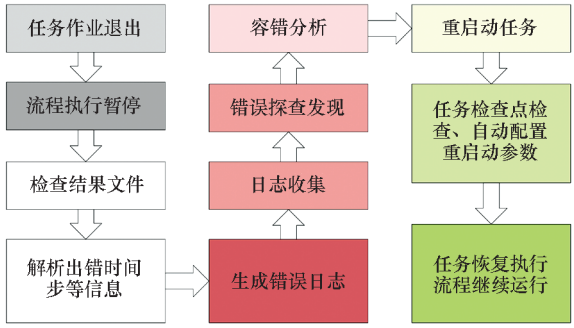


图 12　基于检查点 - 重启动的容错

Fig. 12　Fault tolerance based on checkpoint-restart

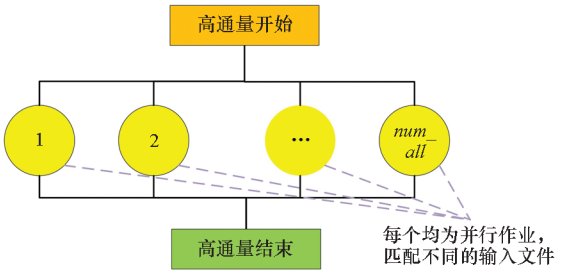
HSWAP 平台针对材料计算等领域高通量计算模式提供了特有支持,功能包括数百上千并行任务的并发投递、监控和容错。高通量计算模式如图 13(a)所示,一般表现为大量并发执行的相似任务,用于材料分子筛选等参数扫描类计算或大数据分析等领域。在容错设计上,用户可定义高通量计算出错的判别标准,如以计算完成百分比作为失败阈值(fail_threshold)。当高通量并发任务失败比例(fail_ratio)超过此阈值,标识任务失败,并重启动失败部分的任务计算,完成容错恢复执行。高通量计算的失败比例定义为式(1),其中 *num_all* 为该高通量所并发执行的所有子任务(或称计算实例)数,*num_failed* 为其中运行失败的子任务数。

$$fail\_ratio = \frac{num\_failed}{num\_all} \tag{1}$$

例如,若对某高通量计算任务,需要并发执行 100 次不同的通量计算(可能为不同参数运行同一软件组件),设失败阈值为 20%,若 100 个计算实例中失败数为 20 以下,则认为该高通量计算成功运行,不进入容错处理;若失败数为 30,此时失

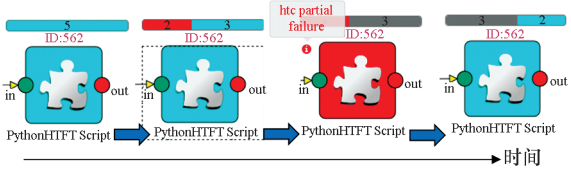
败比例大于失败阈值，进入容错处理过程，只需要重试或重启动运行失败的 30 个计算子任务，运行完毕重复容错过程，直到满足高通量计算任务的成功运行阈值标准，或者达到最大重试次数并报告错误信息。

HSWAP 平台针对高通量计算的容错过程如图 13(b)所示，其中失败阈值设为 20%，每个图标表示同一高通量计算任务在不同时刻的状态，图标上部表示子任务数目及状态，蓝色为正在运行，红色为失败，灰色为成功结束。实测结果表明容错模块能够正确识别错误，并按要求重运行失败的任务，满足自动恢复运行的需求。



(a) 高通量计算模式

(a) High throughput computing pattern



(b) 高通量计算的容错

(b) Fault tolerance of high throughput computing

图 13 高通量计算及其容错

Fig. 13 High throughput computing and fault tolerance

复杂超算应用具有计算规模大、运行时间长等特征，高通量计算等复杂流程也愈加普遍。由于机器故障，程序参数配置错误等异常造成运行中断现象常常出现。依靠人工查看、修改配置、重新投递作业的方式进行错误恢复，会严重影响执行效率。在某百万亿次超算平台上，某工程项目中冲击波计算程序的实际运行情况统计如表 1 所示。

表 1 中人工重启动间隔是指在任务失败后开始计时，直到人工发现错误并投递作业，再次投递排队后继续运行的时间。由于作业可能在深夜或凌晨中断运行而用户无法及时发现，加之重投作业造成再次排队等待时间，实际应用的完成时间就会大幅增长，自动容错技术能够缩减人工重启动间隔时间，显著缩短工程仿真或其他科学研究领域的计算周期。

表 1 某冲击波计算任务的容错效率

Tab. 1 Performance and fault tolerance of some shock computing software

计算模型	CPU 核数	网格数/亿	实际执行时间/h	重启次数	人工重启动间隔/h	容错节约时间占比
1	1540	2.24	5.8	1	7.5	0.56
2	1400	12.6	35	3	5.5, 12.8, 4.5	0.39
3	2128	32.6	94	4	5.5, 48, 48, 104	0.68

5 结论

本文针对超算环境中工作流应用的容错机制展开讨论，调研了容错在典型科学工作流系统的实现方式以及容错的分类。提出了完整的容错生命周期模型；在事件－条件－动作的处理逻辑基础上，提出了可配置的基于决策树的错误恢复模型；设计并实现了模块化、可扩展的科学工作流系统容错架构。本文提出的容错模型已在自主研发的超算环境工作流管理系统 HSWAP 中实现。面向单个任务和高通量任务应用场景分别给出了容错实现策略，并通过实际算例在超算平台上验证了容错对于提高流程执行效率的作用。目前实现的工作流系统和容错模块并没有系统软件的权限，作为应用级系统，无须系统管理员权限就可方便部署，助力加速科研和工程实践过程；但是未能进行超算平台软硬件基础设施的健康信息探查，提供给错误信息分析模块的信息还不够全面，仅从应用的视角实现了错误日志分析功能。下一步的工作将包括全面探测底层软硬件系统和应用各模块的运行信息，给出更准确的出错原因分析。另外，基于可靠性感知的调度方案设计以及云上容错[16－18]也是值得深入研究的技术方向。

参考文献(References)

[1] 张卫民，刘灿灿，骆志刚. 科学工作流技术研究综述[J]. 国防科技大学学报，2011，33(3)：56－65.
ZHANG Weimin，LIU Cancan，LUO Zhigang. A review on scientific workflows[J]. Journal of National University of Defense Technology，2011，33(3)：56－65.(in Chinese)

[2] Dauwe D，Pasricha S，Maciejewski A，et al. An analysis of resilience techniques for exascale computing platforms[C]// Proceedings of IEEE International Parallel & Distributed Processing Symposium：Workshops，2017.

[3] Zhao Y，Xiong Y H，Lee E A，et al. The design and application of structured types in ptolemy II[J]. International

Journal of Intelligent Systems, 2010, 25(2): 118 – 136.

[4] Ludäscher B, Altintas I, Berkley C, et al. Scientific workflow management and the Kepler system[J]. Concurrency and Computation: Practice & Experience, 2006, 18: 1039 – 1065.

[5] Deelman E, Singh G, Su M H, et al. Pegasus: a framework for mapping complex scientific workflows onto distributed systems[J]. Scientific Programming, 2005, 13(3): 219 – 237.

[6] Wolstencroft K, Haines R, Fellows D, et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud[J]. Nucleic Acids Research, 2013, 41: 1 – 5.

[7] David C, Gabor G, Andrew H, et al. Programming scientific and distributed workflow with Triana services[J]. Concurrency and Computation Practice and Experience, 2006, 18(10): 1021 – 1037.

[8] 赵士操, 肖永浩, 段博文, 等. HSWAP: 适用于高性能计算环境的数值模拟工作流管理平台[J]. 计算机应用, 2019, 39(6): 1569 – 1576.
ZHAO Shicao, XIAO Yonghao, DUAN Bowen, et al. HSWAP: numerical simulation workflow management platform suitable for high performance computing environment[J]. Journal of Computer Applications, 2019, 39(6): 1569 – 1576. (in Chinese)

[9] 李于锋, 莫则尧, 肖永浩, 等. 超算环境科学工作流平台的引擎设计和资源调度[J]. 计算机应用研究, 2019, 36(6): 1723 – 1726.
LI Yufeng, MO Zeyao, XIAO Yonghao, et al. Engine design and resource scheduling of scientific workflow application platform in supercomputing[J]. Application Research of Computers, 2019, 36(6): 1723 – 1726. (in Chinese)

[10] Lackovic M, Talia D, Tolosana-Calasanz R, et al. A taxonomy for the analysis of scientific workflow faults[C]// Proceedings of 13th IEEE International Conference on Computational Science and Engineering, 2010.

[11] Yu J, Buyya R. A taxonomy of workflow management systems for grid computing[J]. Journal of Grid Computing, 2006(3): 171 – 200.

[12] Han L, Canon L C, Casanova H, et al. Checkpointing workflows for fail-stop errors[J]. IEEE Transactions on Computers, 2018, 67(8): 1105 – 1119.

[13] Ustun Y, Pierre M, Mladen V, et al. Fault-tolerance in dataflow-based scientific workflow management[C]// Proceedings of 6th IEEE World Congress on Services, 2010: 336 – 343.

[14] Tolosana-Calasanz R, Bañares J Á, Rana O F, et al. Adaptive exception handling for scientific workflows[J]. Concurrency and Computation Practice and Experience, 2010, 22: 617 – 642.

[15] Dong R. Models, languages, and algorithms for scientific workflow monitoring and exception handling[D]. Wayne State University, 2018.

[16] Zhu X M, Wang J, Guo H, et al. Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(12): 3501 – 3517.

[17] Vinay K, Kumar S M D, Raghavendra S, et al. Cost and fault-tolerant aware resource management for scientific workflows using hybrid instances on clouds[J]. Multimedia Tools and Applications, 2018(77): 10171 – 10193.

[18] Costa F, de Oliveira D, Ocaña K A C S, et al. Handling failures in parallel scientific workflows using clouds[C]// Proceedings of SC Companion: High Performance Computing, Networking Storage and Analysis, 2012: 129 – 139.