计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220821
61(1): 86−97, 2024

神威超级计算机运行时故障定位方法

高剑刚[1] 郑 岩[1] 于 康[1] 彭达佳[1] 李宏亮[2] 刘 勇[1] 何王全[1] 陈德训[1] 王 飞[1]

[1](国家并行计算机工程技术研究中心 北京 100190)

[2](江南计算技术研究所 江苏无锡 214083)

(13701512205@139.com)

Runtime Fault Location Method for Sunway Supercomputer

Gao Jiangang[1], Zheng Yan[1], Yu Kang[1], Peng Dajia[1], Li Hongliang[2], Liu Yong[1], He Wangquan[1], Chen Dexun[1], and Wang Fei[1]

[1](*National Research Center of Parallel Computer Engineering and Technology*, *Beijing* 100190)

[2](*Jiangnan Institute of Computing Technology*, *Wuxi*, *Jiangsu* 214083)

Abstract In order to meet the needs of scientific research and engineering applications, the performance of high-performance computers has been continuously improved, the system scale has continued to increase, and the error rate of systems and applications has inevitably continued to increase. Quickly discovering and locating system and application-level errors and providing high-quality services to users have become issues that need to be considered urgently in the design and development of supercomputer systems. Hardware failures and exceptions, software program errors, etc. in supercomputer systems will cause users to hang up and exit large-scale parallel applications. How to quickly and accurately locate the fault site, so that administrators or users can view abnormal faults for high-precision and high-efficiency diagnosis based on this, is an important basis for maintaining the reliability of high-performance computing systems. According to the architecture of "New Generation Sunway Supercomputer" and the characteristics of SW26010-Pro many-core processor, a runtime fault location method is proposed, including fault correlation analysis based on message passing, online comprehensive analysis and diagnosis based on global aggregated information, The key technologies such as abnormal thread filtering method for Shenwei many-core processors are expounded on how to effectively detect, collect, and process abnormal information of a large number of system resources and parallel processes, so as to deal with the problem of efficient fault location system in ultra-large-scale high-performance computing in the future.

Key words exascale computer; reliability; fault location; runtime; many-core processor

摘 要 随着高性能计算机的性能不断提升、系统规模不断提高,系统和应用的错误率也不可避免地持续增多.快速发现和定位系统及应用级的错误、为用户提供高质量服务,成为了超级计算机系统设计开发过程中急需考虑的问题.超级计算机系统中硬件故障与异常、软件程序的错误等都会导致用户大规模并行应用的错误、挂死与退出.如何快速准确定位错误现场,让管理员或用户以此为基础查看异常发生的故障进行高精度、高效率的诊断,是维护高性能计算系统可靠性的重要基础.高性能计算机传统的故障定位主要通过硬件异常跟踪、系统日志分析和程序主动探测等方法,缺乏对无日志信息、无明显故障现象的程序挂死问题的定位手段,并且技术的扩展性也面临挑战.针对"新一代神威超级计算机"体系结构和SW26010-Pro众核处理器特点,提出一种运行时故障定位方法,包括基于消息传递的故障关联分析、基于全局聚合信息的在线综合分析诊断、面向申威众核处理器的异常线程过滤方法等关键技术,阐述了如何有效检测、收集、

收稿日期:2022-09-23;修回日期:2023-04-27

处理大量系统资源和并行进程的异常信息问题，为应对未来超大规模高性能计算中故障高效定位难题提供有效支撑.

关键词 E级计算机；可靠性；错误定位；运行时；众核处理器

中图法分类号 TP302

超级计算机是解决经济建设、社会发展、科学进步、国家安全等一系列重大挑战性问题的重要手段，已经成为信息时代世界各大国综合实力的体现. 目前，超级计算机的峰值速度已经达到E量级规模，根据2022年5月发布的Top500[1]，排名第一的超级计算机是部署在美国能源部下属橡树岭国家实验室的超级计算机前线（frontier），其峰值性能为1.686 EFLOPS，实测Linpack性能为1.102 EFLOPS，峰值计算能力为537 PFLOPS，Linpack持续性能为442.01PFLOPS. 中国的“神威太湖之光”超级计算机排名第六，峰值性能为125 PFLOPS，持续性能为93 PFLOPS. 综合分析以往Top500榜单，超级计算机的性能发展趋势是约每4年提高1个数量级，略高于摩尔定律.

随着高性能计算机的规模越来越大，系统的可靠性也面临巨大挑战. 通过分析超级计算机Blue Waters[2]的500万份高性能计算（high performance computing, HPC）作业，可以发现作业规模从10 000个节点增加到22 000个节点，故障概率也会从0.8%增加到16.2%，并且一旦节点发生故障，它很可能会衍生后续故障. IBM Blue Gene/L包含1 024个计算节点，峰值性能70.72 TFLOPS的平均无故障时间（mean time between failure, MTBF）为53 158 h[3-4]. 超级计算机CRAY XT4[5]包含23 016个处理器，峰值性能为101.7 TFLOPS，MTBF大约为772 h；超级计算机泰坦（Titan）有18 688个节点，每个节点有一个AMD 16核Opteron CPU（整个系统有299 008个内核）、一个NVIDIA Tesla K20 GPU和32 GB的主内存，MTBF大约为100 h[6]. 可以看出，随着超级计算机运算核心规模的增大，平均无故障时间也越来越短，这要求系统提高故障定位的效率.

现代超级计算机组装了大量的处理器、加速器、内存模块和更多部件，比如处理器计算核心的规模，在过去10年内从十万量级迅速上升为千万量级，高效故障定位分析对系统的可靠性越发重要，但也变得愈发困难[7-8]. 目前的故障定位技术需要对海量信息进行采集和分析，包括计算资源、I/O部件、网络环境以及操作系统等，并且多种故障之间具备很强的关联性，在大规模的并行环境下存在传递效应，某个根源性的故障可能引起多个故障的发生，例如表征为通信错误的课题，实则为某节点故障无法正常吞吐消息，采用重发并不能解除故障，采用回卷、迁移等操作则需要能定位错误节点. 如果通过并行故障定位工具来定位系统问题[9-10]，往往对明显的硬件故障有较好的效果，但对于软件挂起问题或者瞬时的系统异常，定位的精确度和效率都存在一定局限性.

当系统处理器、网络、存储器等硬件系统发生故障时，操作系统和维护系统可以通过定期探测、心跳检测、日志分析等手段来定位故障发生的原因，但很多系统的异常并不会通知操作系统或管理软件，也不会在日志系统中记录，比如结果错、单错、网络丢包等信息，但往往关联性的错误会对系统造成巨大影响. Titan是位于橡树岭领导计算设施（oak ridge leadership computing facility, OLCF）的Cray XK7系统，是最早使用CPU和GPU混合架构的超级计算机之一. Titan的峰值性能为17.59 PFLOPS，当它退役时，它在Top500排名中排名第九[11]. Titan上的每个异常事件都会自动注册到故障数据库中. 据统计，Titan在2014—2018年5年的故障事件总数为2 663 512. 经过一个过滤阶段，故障数据库中的事件数量减少了99.78%. 剩下的0.22%的事件对应于我们所描述的5 654个独特的故障，分别分布为565, 649, 1 824, 1 291, 1 325个事件. 虽然Titan系统通过筛选丢弃相关衍生故障，提高了错误定位精度，但通过单一的探测方法，依然无法对大规模并行计算系统高效地错误定位.

在程序运行出错时，从用户视图看会有输出结果错误、输出报错和长时间无输出3种直观表现，其中输出结果错误是指输出没有达到预期，用户需要通过分析软件代码结合调试工具确定错误原因，这种错误只能由具备预期的用户去发现和定位；输出报错通常是用户程序或者运行环境捕获到运行时错误，将错误现象甚至原因反馈给用户，进而采取相应的错误处理措施和软硬件故障修复；而对用户来说，长时间无输出是程序运行时最难以发现和定位错误的情况，由于没有反馈无法判断程序当前是否运行正常，只能求助于系统管理员进行调试和维护检查.

在并行计算机系统中，运行时系统是指程序执行时支持其解释、运行、检查、调试、优化等功能的系统运行环境，运行时环境是拉近用户程序和系统平台之间鸿沟的一个重要层面，让程序运行在更好

的运行时环境下能够更好地发挥出应用程序的性能，减少运行时间从而增加系统效率[12]. 目前在超级计算机系统设计中，系统管理服务、操作系统、并行语言库等支撑程序运行的环境都可以称为运行时环境.

本文的主要贡献是针对程序挂死和无直接硬件异常的大规模应用，提出了一种用户层运行时故障定位方法，在不依赖系统管理服务的情况下，利用应用程序本身的并行特性，快速对软硬件异常进行筛查和定位，以适应超级计算机的好用性和可扩展性需求；该方法不仅针对特定故障类型，而且可以全面地对程序挂死、文件读写问题、硬件异常进行筛查，提高了故障定位的效率，丰富和完善了面向超大规模并行计算机系统错误定位的技术手段.

1 相关工作

在高性能计算中，故障定位用于在大规模并行程序中找到异常的原因，是系统维护和软件开发周期中最关键和最耗时的过程之一，其中基于运行时的故障定位方法也被广泛使用.

循环感知进度依赖分析工具 prodometer[13] 用于高性能计算故障定位和大规模并行应用程序调试. prodometer 需要在 MPI(message passing interface)程序中预先链接一个函数库，并在运行过程中为每个进程建立马尔可夫模型，用于分析软件栈的依赖关系. 通过推断 MPI 任务循环内的进度依赖性，可以显著减少识别故障起源任务的工作量，找到系统中进展最少的任务，从而精确地定位出现故障的 MPI 进程. prodometer 应用在 IBM Blue Gene/Q 高性能计算机上，成功定位了 32 768 个任务中的故障原因.

天河二号超级计算机上的故障原因解析器[14] 通过使用分布式监视器观察异常消息传递行为来检测程序故障，并利用事件驱动机制同时触发不同节点组之间的全局状态检查. 实验结果表明，故障原因解析器用于故障检测的延迟是可以接受的，开销可以忽略不计. 此外，该技术不需要管理权限，可以很容易地集成到现有的大型并行程序中.

基于故障定位树的检测框架 MPFL(message-passing based on fault localization)是一种基于消息传递的故障定位方法[15]，采用层次化和分布式的设计思想，将全局的故障定位任务分配给不同的节点，由各节点运行轻量级的故障定位进程，对局部范围内的多个节点进行故障的检测和分析.

Bolt [16] 利用运行时检查来检测软错误，并提出了一种编译器指导的软恢复方案，该方案提供细粒度和有保证的恢复，而不会产生过多的性能和硬件开销.RedMPI [17] 专注于检测和纠正 MPI 应用程序中的静默数据损坏，其中应用程序数据的损坏通过在副本之间产生不同的 MPI 消息来表现出来.

文献 [13−17] 所述的方法虽然通过分布式并行和 C 运行时方法提高了故障定位效率，但依然存在依赖消息通信和错误定位方法单一的问题.

2 面向神威超级计算机的运行时故障定位方法

2.1 运行时故障定位架构设计

在超级计算机的运行过程中，由于硬件故障和软件错误的表现特性并不一样，定位的方法和手段也有着很大的不同. 通过建立面向高性能计算的多策略协同的并行错误定位架构，可以快速对超大规模并行计算机系统进行错误溯源分析与处理，以实现系统故障的快速发现、分析、隔离与处理，支撑超级计算机容错和高可用需求，提高系统可用性.

本文面向新一代神威超级计算机，提出了运行时故障定位方法，针对程序出错时无硬件异常现象和操作系统日志信息问题，利用基于运行时库的错误探测分析、基于全局聚合信息的综合诊断、面向申威众核处理器的异常线程过滤等错误定位技术，同时进行软硬件异常处理，构建了基于用户层的软件探测分析的高效错误定位方法，并行运行时故障定位架构如图 1 所示.

1) 基于运行时库的错误探测分析技术

神威高性能网络接口芯片(Sunway high-performance network interface chip, SWHNI)采用硬件方式负责网络层传输，实现了处理器间高速通信，层功能实现了为计算机系统提供高带宽、低延迟的数据传输能力[18].

基于运行时库的错误探测分析技术通过网络接口芯片 SWHNI、神威消息库及网络管理软件，可以对大规模系统的局部资源错误进行快速定位，通过紧密耦合节点间消息实现机制，感知运行时错误并进行分析和定位，快速准确发现错误源头.

2) 基于全局聚合信息的综合诊断技术

SWHNI 消息引擎实现了软件定义的硬件集合通信机制，解决了超级计算机中受限的硬件资源和巨大的软件需求间的矛盾，大幅度提高了集合通信的处理能力、可扩展性和实用性，为全局聚合信息的分

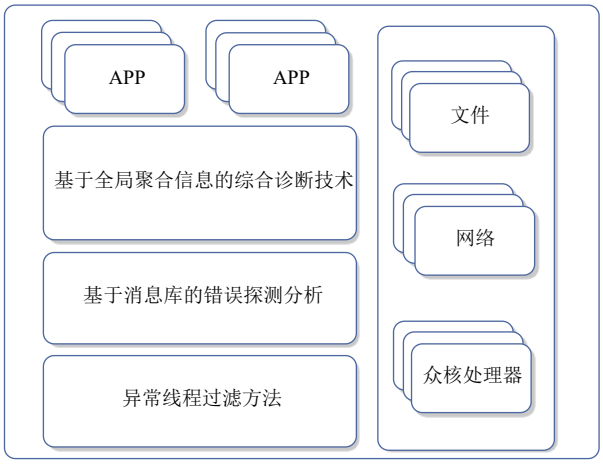


Fig. 1 Parallel runtime fault location architecture

图 1 并行运行时故障定位架构

析提供了支撑[18].

基于全局聚合信息的综合诊断通过硬件集合通信机制，可以实时开展全局计算与通信资源的状态诊断，针对大规模并行应用常见的面向全局资源调试难、诊断难的问题，通过在线状态收集和离线分析相结合的方法，实现错误状态的快速、准确诊断.

3）面向申威众核处理器的异常线程过滤技术

神威超级计算机的维护处理机系统（baseboard management controller, BMC）为插件上的众核处理器提供基础维护及数据采集服务，由于插件上的处理器数量有限，全机的 BMC 系统可以进行并行状态检测[19].

面向申威众核处理器的异常线程过滤技术基于 BMC 系统，可以高效排查众核处理器的异常状态，可精准定位部分大规模并行应用程序因硬件失效、应用程序错误而导致的无现象异常挂死、消息死锁、集合操作挂死和 I/O 故障问题等，提高了并行程序调试的效率和好用性.

神威超级计算机运行过程中，软硬件的错误都可能导致系统发生挂死，最常见的问题往往出现在处理器单元、消息部件和全局共享系统 3 个运行时故障定位技术中. 这 3 个技术都具有运行时特性，通过技术融合，可以在不影响用户课题运行的情况下，基本覆盖常见的软硬件错误，诊断出错误的源头.

通过综合上述 3 个故障定位技术，设计一体化的并行时故障定位技术，为超级计算机系统错误定位提供了有力的技术支撑.

2.2 基于运行时库的错误探测分析

在高性能计算机中，应用级错误主要来自于操作系统的异常报告、网络管理的消息超时、作业管理的节点错误和状态查询中断，神威超级计算机中使用的运行时库主要包括通信库（MPI 库）和并行语言库，其提供的异常查询接口对并行应用运行过程中报告的错误进行定位，为上层并行语言提供错误定位支持，具体的软件栈如图 2 所示.

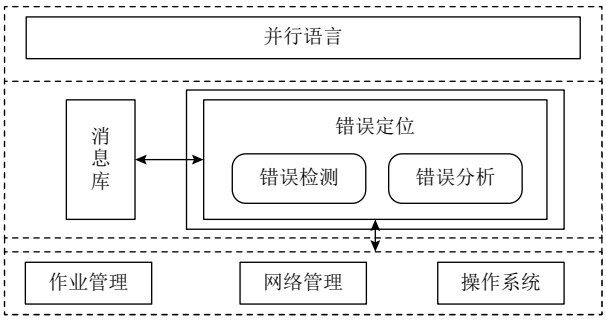


Fig. 2 Error detection analysis based on the runtime library

图 2 基于运行时库的错误探测分析

错误检测主要是通过基于定制的消息探测和关联故障综合分析相结合的方式进行，其中定制消息探测包括响应探测、状态探测和投票探测 3 种方法.

1）响应探测

要求目标节点立即返回响应消息，以判别该目标节点与发起节点的网络是否连通. 发起节点可以向周围节点群发送响应探测 *check_ping* 消息，收到该消息的节点需要立即回复无数据的应答消息 *reply_ping*，以帮助发起节点判断自身的通信环境.

2）状态探测

要求目标节点返回节点资源状态、通信状态或者消息日志内容等运行状态信息. 如图 3 所示，节点 P_1 向目标节点 P_2 发送状态探测消息 *check_status*，P_2 根据收到的消息内容回复 P_1 包含对应信息的 *reply_status* 消息. 状态探测应用在多种场景中，例如 P_1 等待接收 P_2 消息超时，可以通过状态探测取得 P_2 的消息日志，分析它是否已发送相关消息、当前处于什么消息状态等；在同步超时的情况中，同步发起节点可以主动探测未同步节点在该进程组内最近同步操作的状态. 状态探测消息可以根据具体的问题进行扩展，使得发起节点能够得到分析错误所需要的各种信息.

3）投票探测

发起节点要求节点组内所有节点都对目标节点进行响应探测或状态探测，并汇报结果进行投票. 如图 3 所示，节点 P_1 向目标节点 P_2 所在进程组内所有节点发起投票探测消息 *check_vote*；收到该消息的节点需要立即向 P_1 回复 *reply_vote* 消息告知 P_1 通信正常；同时向 P_2 发送响应探测消息 *check_ping*，收到回

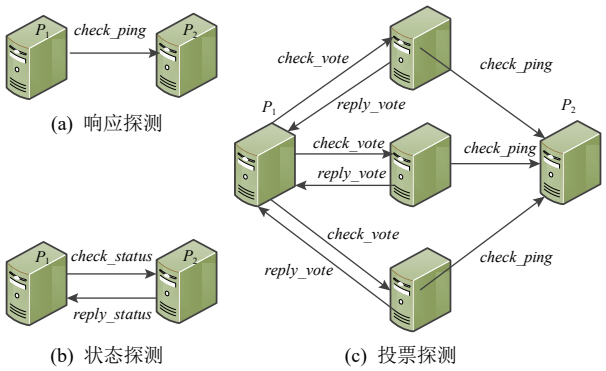


Fig. 3 Message-based error detection method

图 3 基于消息的错误探测方法

复或超时后向 P_1 发送对 P_2 运行状态的投票消息 *reply_vote*, 以帮助 P_1 确认 P_2 的运行状态.

运行时系统在初始化时为每个节点指定拓扑相关的错误检测负责节点, 以保证探测的效率. 错误被定义为五元组〈*src*, *dst*, *time*, *type*, *status*〉, 其中 *src* 为发现错误的节点, *dst* 为发生错误的节点, *time* 为对应发现错误的时间, *type* 为内部定义的错误类型, *status* 为存储当前探测进度. 定位流程包括:

1) 节点 *src* 感知到错误后, 将立即向 *dst* 发起响应探测, 并向 *dst* 对应的负责节点 *Pr* 报告探测结果;

2) 负责节点执行错误分析, *Pr* 查询消息日志, 通过 *time* 判断是否在一定时间内包含对 *dst* 的探测记录;

3) 检测 *type* 属性, 对于已完成相同探测的等待 *status* 标记完成后返回, 通过排重和关联合并错误, 不能合并的则执行新增错误操作;

4) 对于新增错误发送消息进行探测;

5) *Pr* 综合报错结果, 进行错误分析并报告.

错误检测模块将错误定位任务与底层消息机制紧密联系起来, 能够更好地适应高性能计算系统. 通过修改并行语言运行时库(MPI 库的消息实现), 在消息日志中记录每个正常消息发起时间并设置计时器, 计时器超时将触发错误分析模块(在 3.2 节的测试中将讨论由此带来的额外开销). 计时器的时间阈值由消息类型和通信距离共同决定, 经过排重和关联后, 依据规则库决定消息探测的类型. 响应探测通常针对 2 点之间的 1 对 1 消息超时, 状态探测是在确定对方存在响应后的进一步状态获取, 投票探测则是在关联发现多对 1 故障后的探测手段. 错误探测分析对用户透明, 仅在发现疑似问题后向用户通知错误信息. 错误探测的开销包括消息发起开销和消息响应开销 2 个部分, 我们通过限制探测消息的长度, 即将状态探测的数据控制在单包短消息数据长度内, 以保证探测流程尽量不影响系统的正常运行, 在神威系统中单包消息延迟为微秒级; 消息响应时间与探测触发时机、应用类型紧密相关, 视被探测方何时查看消息队列而定. 通常来说, 计算密集型课题消息处理间隔时间较长, 而消息密集型课题会更频繁地去主动获取网络消息, 对消息的响应时间更短, 因此更适合错误探测的方法, 在实际应用中错误探测和分析的总时间可以控制在秒级.

错误分析模块包含通过历史查错经验总结的规则库, 指定满足错误合并的条件, 例如其中有些重复的案例:

1) 多个节点报与同一节点消息超时或挂死;

2) 多个同步操作报挂死在同一节点;

3) 多个消息死锁在同一资源;

4) 短时间内节点上多次 I/O 错误;

5) 接收其他软件提交的错误信息等.

也有部分可以进行关联的案例, 例如:

1) 网络错误, 导致消息或同步挂死;

2) 节点硬件故障, 导致其他节点报消息超时或挂死;

3) 节点硬件故障, 导致同步挂死;

4) I/O 错误导致消息超时或挂死.

在错误分析模块中, 对错误检测发现的错误信息进行关联, 通过匹配错误类型、错误位置、错误时间区域进行快速筛选去重, 从而快速定位错误根源.

基于运行时库的错误探测方法是对用户透明的, 通过局部多进程消息的方式进行探测, 在代价较小的前提下能够较为准确地发现节点状态、消息部件故障以及部分软件问题引起的错误, 通过进一步的排重和关联, 能够有效避免错误报告杂乱无效的问题.

2.3 基于全局聚合信息的综合诊断技术

在大规模并行程序运行过程中, 由于任务分配不均匀造成的部分节点暂时无输出, 与节点、网络故障等硬件错误或者软件编写错误造成的程序挂死(程序在某一操作中等待触发条件以结束该操作, 然而触发条件无法达到)现象, 对用户来说难以准确分辨. 然而, 由于系统规模巨大, 特别是在无输出的条件下, 用户难以快速判断是否有错误发生, 以及进一步分析得到错误原因, 因此为提供用户发起的运行时程序状态检查机制十分必要.

当前常用的检查方法包括穷尽排查和在线调试等. 穷尽排查是指系统管理员通过维护管理工具依次排查系统硬件和运行环境状态, 查找是否有可能影响当前任务运行的故障出现, 从而推断出当前程

序无输出是否处于不正常状态，这种方法由于不具有针对性的查找，工作量大且最后结果并不能排除程序本身错误，因而存在不确定性，不适合大规模系统和应用. 在线调试是使用更多的一种方法，用户或者程序开发维护人员通过调试工具对计算系统中的单个节点依次进行重现式检查，查看节点是否陷入死循环等状态；更进一步的方法是开发人员在程序开发时记录程序运行状态，维护人员通过在线调试工具依次检查该状态以判断程序是否运行正常. 然而，在大规模环境下由于节点较多，用户不具备全局视图，并且部分运行时的实现对用户是透明的，依次检查耗时可能十分漫长，而且在调试过程中也可能改变程序运行状态.

在神威超级计算机软件系统中，通过在运行时库中加入接收用户信号的接口，由对程序运行状态最关心的用户发起检查，将部分节点作为叶子节点，采用递进的树型结构层层推进检查流程，最终向用户报告当前程序运行状态. 该技术在不影响程序运行状态的前提下，帮助程序运行用户了解当前程序运行状态，既能够避免程序运行用户由于等待时间太长而“杀掉”程序的误操作，又有助于实际发生错误时的进一步处理. 该技术由用户发起，减轻了系统维护人员的负担，利用节点之间的消息通信机制，对程序运行状态进行收集和比对，有针对性地得到程序运行状态，能够及时发现程序运行错误.

在并行程序中，进程最终都需要通过集合消息实现一致性协调，当部分节点在计算过程中发生软硬件故障时将无法到达集合点，或者在集合过程中发生通信故障，都会导致程序挂死. 基于全局聚合信息的综合诊断，分为在线收集和离线诊断 2 个阶段.

在线收集阶段负责一次性获取全局所有节点的软硬件状态信息，主要有 4 个步骤，如图 4 所示.

1）发起诊断. 用户向指定作业发起诊断，并准备好状态数据输出文件.

2）诊断类型分析. 判断是否处于集合消息过程中，以及是否处于点对点消息等待状态.

3）数据状态收集. 收集节点状态信息，即进程通过共享内存和消息通信聚合状态信息. 收集数据采用分区聚合的方法，由于部分节点或者通路故障导致的收集错误，将触发软件超时并报告，将对应的数据置为空.

4）状态输出. 全局主节点收集数据后进行分类和预处理，输出到对应文件.

离线诊断分为 5 个步骤，如图 5 所示.

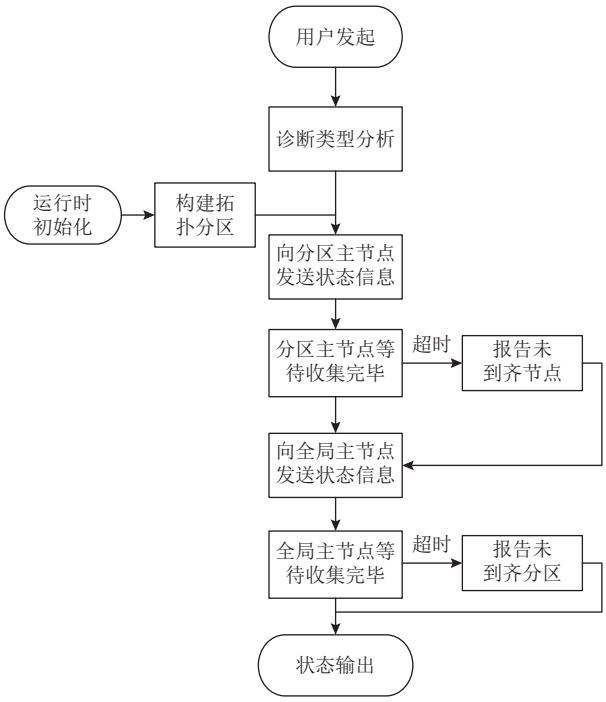


Fig. 4 Online collection of global status

图 4 全局状态在线收集

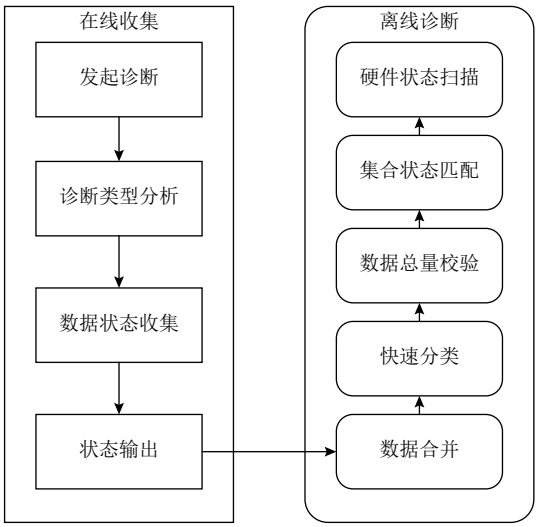


Fig. 5 Comprehensive diagnostic technique based on global aggregation information

图 5 基于全局聚合信息的综合诊断技术

1）数据合并. 将状态文件批量合并处理，并按照进程编号进行排序.

2）快速分类. 对状态进行预分析，分别存储不同通信域的集合状态数据.

3）数据总量校验. 检查是否有无状态进程，判断是否有节点 *down* 或进程被杀死.

4）集合状态匹配. 根据集合状态，区分进程当前所在的通信域，检查是否有未到达进程；对未到达节

点进行关联分析,检查不同通信域之间是否存在依赖关联.

5) 硬件状态扫描. 当软件状态显示无未到达进程时,使用深度优先算法扫描集合树中进程的硬件状态,判断是否存在网络硬件错误;对于集合状态匹配发现的未到达进程,同样需要扫描该进程的硬件,判断是否存在错误.

基于全局聚合信息的综合诊断技术在运行时使用消息聚合的方法收集各节点状态信息,与运行时错误探测分析技术在触发机制、实现方法以及目标问题类型上都有所不同. 在触发机制上,运行时错误探测分析技术由消息计时器超时触发、对用户透明,基于全局聚合信息的综合诊断技术由用户判断,课题长时间无输出后主动发起. 在实现方法上,前者是完全基于运行时消息的探测方式,存在探测—分析—探测的迭代定位模式;后者使用消息仅是作为一次性收集信息的手段. 在问题类型上,前者主要针对局部问题,消息内容超过单包后将报警;后者的分析样本数据更大,因此采用离线分析的方式,更适合于较大规模系统的综合分析.

相较于穷尽排查的方法,本节所描述的技术能够获取程序本身的运行时信息,包括但不限于当前的通信对象和类型、计算的操作类型和位置等,因此更具有针对性. 这些信息可以反映运行节点之间当前联系状态,因此需要通过树型结构进行聚合,通过统一视角进行离线分析. 本文技术也可以作为在线调试的前置,在确定问题节点或者代码范围的基础上,通过在线调试手段进行更精确诊断.

2.4 面向申威众核处理器的异常线程过滤方法

在申威众核处理器中,所有线程的程序计数器可以被有效用于定位并行程序中异常挂起的线程. 当程序出现挂起异常时,一种比较常见的现象是:少量线程执行到某个同步或者通信语句时,由于硬件故障或编码错误,不满足条件停止执行,进入等待状态. 此时,其 PC(program counter) 值是稳定不变的;而其他绝大部分线程能够正常执行一段时间,直到出现同步或通信等待,此时,它们的 PC 值要么不断变化,要么不变,但其值与异常线程的 PC 值通常不同. 在这种情况下,我们可以数次采集所有线程的 PC,找出那些 PC 值一直不变的线程,并根据 PC 值不变量将线程分为不同的类,通常我们可以将所有线程分为少数几类,然后将这些类展示给用户,辅助定位异常进程、线程. 在明确异常进程、线程的情况下,根据 PC 值能够直接定位到错误程序挂住的代码行,有效提高了程序调试效率.

异常线程过滤方法的具体流程为 3 步:1)数次采集目标程序所有线程的程序计数器;2)过滤出 PC 值在多次采样中均为改变的线程;3)根据 PC 值,将异常线程分组. 对于挂起的异常线程,若其 PC 值不变,算法能够检测到并记录.

对于进入等待状态的正常线程,通常数量较多,可以通过数量与异常线程区分开来;对于仍在执行的正常线程,PC 值变化且 2 次采集通常不同,将不会记录.

该方法的处理时间主要由步骤 3)的异常线程分组构成,即对每一个线程,根据其 PC 值查找对应的分组,并将其插入分组. 因此,处理时间与线程的数量和不变程序计数器的类数相关. 假设作业中的线程数为 n,不变程序计数器的类数为 M,则方法的时间复杂度为 $O(M\times n)$.

增加采样次数,能够有效提升方法的准确性. 程序执行中存在一种情况:正执行一个很大迭代次数的循环线程,多次 PC 扫描的值相同,这将导致误判. 假设某个线程在执行一个 k 条指令的循环,为了简化分析,我们假设采样 PC 值时线程正在执行每条指令的概率相同(采集的时机是不确定的,可视为随机,但实际上,不同的指令需要不同的周期数,采集到某指令 PC 值的概率还与指令执行周期数有关),概率为 $1/k$,则 s 次采样 PC 值均相同的概率为

$$p=\frac{1}{k^{s-1}}. \tag{1}$$

对于 n 个独立的线程来说,s 次采样 PC 值均相同的线程数的期望为

$$E=np=\frac{n}{k^{s-1}}. \tag{2}$$

为了过滤这些正常执行的线程,增加采集次数,使得期望 $E<1$,此时

$$s>1+\log_k n. \tag{3}$$

假设程序中 $n=1\times10^7$, $k=250$,我们可以采集至少 4($s>3.91$)次,保证每次采集 PC 值均相同的正常执行的线程数的期望小于 1.由式(3)可见,k 值越高,意味着需要采集的次数 s 越小;相反地,一个较低的 k 值意味着循环执行时间越短,在数次采样中,线程执行的程序块通常不同,也意味着 PC 值不同. 在实际应用中,程序循环块的指令条数往往较高,即 $k>250$,因此仅需进行 3 或 4 次采集即可. 上述分析是统计学上的期望,实际应用中仍然存在在数次采集中 PC 值均相同的线程概率,但由于判定出错的线程数量极少,用户使用调试跟踪等手段可以轻易排除这类错误.

在采集的过程中，获取到的运算控制核心的 PC 值可能指向其他进程或者操作系统核心代码. 这种情况下，如果运算处理核心在执行加速核，那么其对应的运算控制核心在等待所有运算处理核心结束运算，其状态是确定的. 一般情况下，高性能计算应用占用了绝大部分 CPU 时间，BMC 采集到的 PC 值大概率属于目标的计算进程. 如果在多次采样中，采集到的运算控制核心的 PC 值均不属于目标进程，那么这个进程很有可能发生异常.

该方法能够精确定位大规模并行应用程序因硬件失效、应用程序错误而导致的无现象异常挂死、消息死锁、集合操作挂死和 I/O 故障问题. 对于部分程序错误导致线程 PC 值持续不断变化的错误，该方法存在一定局限性.

3 实验结果与分析

3.1 实验平台

本文采用的实验平台是新一代神威超级计算机，该机器采用 SW26010-Pro 众核处理器，处理器架构与“神威•太湖之光”系统的 SW26010 处理器[20] 架构类似，每个处理器包含 6 个核组，每个核组中有 65 个核心，总共 390 个核. 每个核组包含一个管理处理元素、一个计算处理元素集群和一个内存控制器，核组中运算核心以 8×8 阵列方式进行排布，运算核心之间以及运算核心与外部交互通过阵列内网络进行互连.SW26010-Pro 处理器支持大量线程和数据并行性，在并行工作负载上提供高性能，SW26010-Pro 处理器的架构如图 6 所示.

实验最大使用总共 32 768 个 SW26010-Pro 众核处理器，具备 12 779 520 核并行规模.

3.2 基于运行时库的错误探测分析

基于运行时库的错误探测分析需要通过消息库的探测和分析，在不影响用户程序的同时，不断对系统错误进行迭代分析. 由于该技术需要在系统运行时库核心中增加错误定位与分析模块，并对用户发起的探测行为进行响应处理，记录与分析消息日志、节点状态等信息，会对课题运行带来一定的开销. 信息记录的动作由消息和定位触发，因此开销也和通信密度成正比，相对来说，通信密集型课题开销会比计算密集型课题开销更大.

我们使用 NAS 并行基准（NAS parallel benchmark, NPB）测试程序集（版本 3.3.1）标准测试程序，包含 IS, FT, EP, CG, LU, MG, BT, SP 等程序，在这些程序中都包含常用的通信操作. 我们设置问题规模为 E 规模，在 NPB 测试程序中 E 规模下不支持 IS，因此在本实验中忽略该程序.

在 4 096 进程规模下测试关闭和开启错误探测分析带来的时间开销百分比为

$$rate_cost = 100 \times (t_fl - t_ori)/t_ori, \tag{4}$$

其中 t_fl 为开启错误探测分析后的运行时间，t_ori 为测试程序在原始环境下的运行时间，测试结果如表 1 所示.

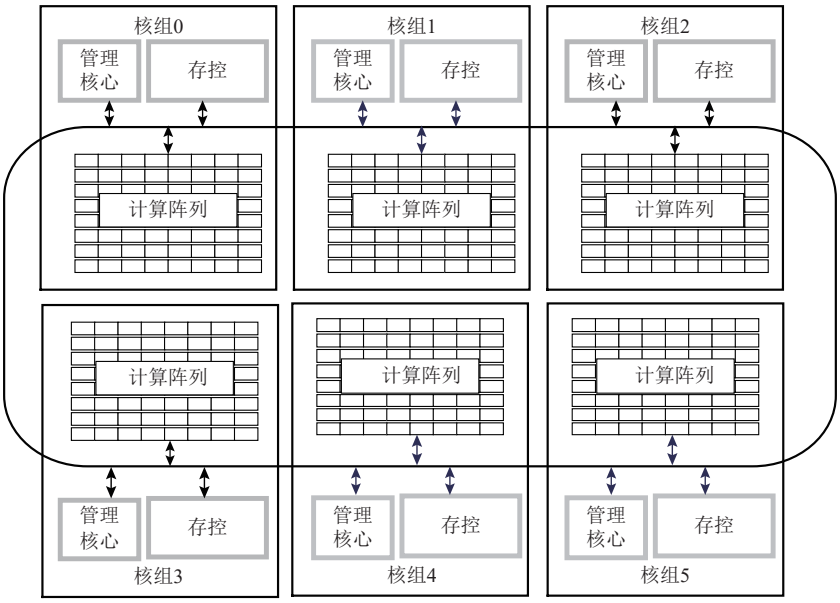


Fig. 6 Architecture of SW26010-Pro many-core processor

图 6 SW26010-Pro 众核处理器架构

Table 1 Percentage of NPB3.3.1 Assembly Time Overhead

表 1 NPB3.3.1 程序集时间开销百分比

程序	时间开销百分比/%
FT	1.02
EP	0.06
CG	0.51
LU	0.38
MG	0.21
BT	1.27
SP	1.39

测试课题额外增加的时间开销百分比最大为1.39%，最小仅为0.38%，结果表明，基于运行时库的错误探测分析方法对程序本身的运行几乎不会产生影响.

3.3 基于全局聚合信息的综合诊断

基于全局聚合信息的综合诊断过程分为在线和离线2个阶段，在线数据信息收集阶段的时间开销同课题运行进程规模、节点的拓扑结构相关，离线分析阶段的时间开销与终端处理状态文件的性能和错误类型相关. 由于该方法可以不影响程序运行状态的前提下帮助程序运行用户了解当前程序运行状态，并且采用离线分析系统规模较大时可能存在可扩展性问题. 为测试诊断方法的可扩展性，在新一代神威超级计算机中，将一个已知故障节点放入测试环境作为错误类型样本，测试题在做集合操作时故障节点会导致课题挂死. 在此模拟错误环境下，测试不同进程规模的错误定位总时间，每次测试取10次平均值，系统规模为4~32 768节点，其中每个节点上运行6个进程，结果如图7所示.

从趋势上看，时间开销在到达32 768节点规模上仍然保持在秒级，并且时间开销与进程规模的比值反而在下降，因此具有良好的可扩展性.

3.4 异常线程过滤方法

我们分别测量2个超节点，并收集不同数量的计算节点的程序计数器的时间，图8显示了耗时. 可以观察到时间随着计算节点数量的线性增长趋势.

在测试中，当测量从 n 个计算节点收集数据的性能时，向 n 个连续节点提交作业，这意味着这些节点位于尽可能少的计算板上并且采集数据使用的维护处理器单元的数量是最少的. 该作业提交方案与资源管理系统尽可能将连续计算节点分配给资源队列的策略以及作业管理系统向连续的计算节点提交并行作业的策略相一致. 在一个确定的规模，我们进行了10次所有线程的程序计数器的采集，并报告了测

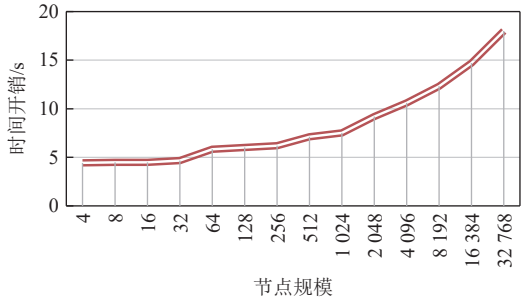


Fig. 7 Error location time at different node scales

图7 不同节点规模下的错误定位时间

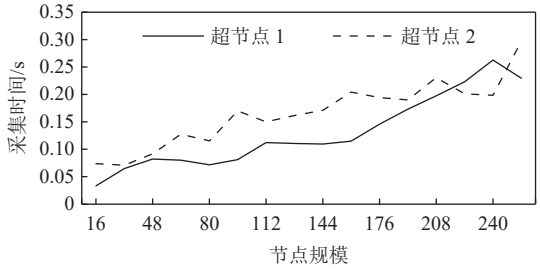


Fig. 8 Time of data collection on different super nodes

图8 不同超节点上采集数据的时间

量结果的平均值. 对于使用多个超节点资源的作业，可以通过并行采集进一步提高效率.

采集数据后的主要工作是将程序计数器不变的线程根据值分类. 我们在范围2~128的 M 下，在范围从1 024个节点(总计399 360个核)到32 768个节点(总计12 779 520个核)的节点规模下测量处理时间. 我们将每个线程控制到指定的挂起位置，使其属于 M 类之一，因此处理耗费的时间最长.

图9显示了处理时间. 可以看出，在每个 M 下处理时间随着应用规模的增加而线性增加. 处理时间随着 M 在每个尺度上的增长而增加，并且在最大规模和 M=128下该方法可以在0.7 s内完成处理.

在图10中，我们可以更清楚地看到处理时间随 M 线性增长的趋势. M 越高，该方法耗时越长. 但是，越高的 M 意味着越多线程挂起，这时用户可能难以判断异常线程. 根据经验，当程序挂起时，M 相对较低，因此处理时间也较短.

采集到作业中所有线程的PC值后，使用异常线程过滤方法可以定位到挂起线程，并生成其详细信息. 图11展示了HPL(high-performance linpack benchmark)运行中的一个诊断输出，当执行函数 *slave_dgemm* 时，10 358节点1号核组上的41 413号任务的8个线程(编号为56~63的从核)挂在了PC为0x4ff0410bb0的指令处，同时，在相应主核上的控制线程挂在了函数 *athread_join*. 根据PC值，可以判断程序正在执行的

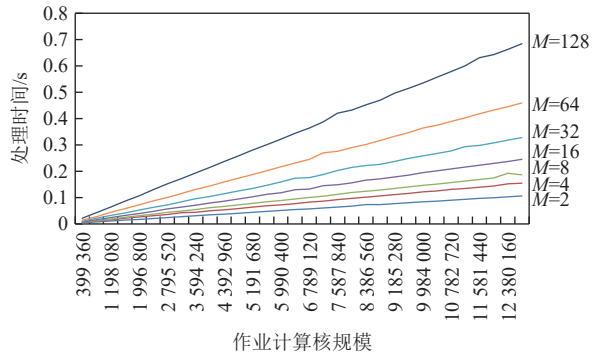


Fig. 9 Data processing time at different scales

图 9 不同规模下的数据处理时间

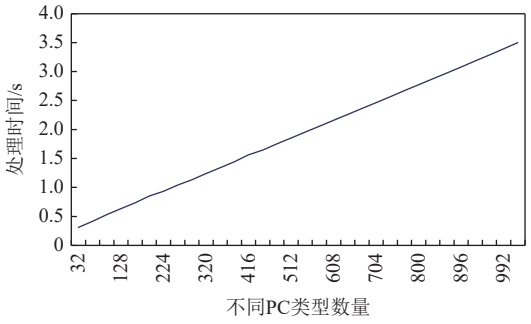


Fig. 10 Data processing time of different PC types at 32 768 node

图 10 32 768 节点时不同 PC 类型的数据处理时间

语句为 RMA(remote message access)，挂起可能是由于计算核间的通信故障导致.

3.5 错误定位综合测试

我们在 32 768 节点环境下分别采用运行时故障定位架构和传统错误定位方法开展综合测试. 其中基于运行时故障定位方法通过并行执行 3 种错误定位技术进行系统错误定位，传统错误定位主要包括操作系统分析、维护系统分析和 MPFL 故障分析.

```
PC 信息条目: (进程号 节点号 核组号 核心号)
=============================================
0x4ff0410bb0 slave_dgemm 8: (41413 10358 1 56) (41413 10358 1 57)
(41413 10358 1 58) (41413 10358 1 59) (41413 10358 1 60) (41413
10358 1 61) (41413 10358 1 62) (41413 10358 1 63)
0x4ff0418b70 athread_join 1: (41413 10358 1)
```

Fig. 11 A diagnosis output in a HPL run

图 11 HPL 运行中的诊断输出

测试设计了 2 种测试集，分别是有明显故障信息 MsgSet1 测试集和无明显故障信息 MsgSet2 测试集.MsgSet1 主要包括处理器管理核心、处理器计算核心、存储器、网络接口、操作系统等错误；MsgSet2 主要包括程序挂死、消息性能异常和消息丢包等无故障信息的错误现象. 通过基于运行时库的错误定位和传统的故障定位方法，比较时间开销.

表 2 显示了传统错误定位和运行时故障定位技术的时间开销对比. 可以发现，由于 MsgSet1 测试集产生的错误在操作系统日志和维护管理系统中有相关信息，定位效率较为高效.MsgSet2 测试过程中，由于该类错误没有操作系统日志信息和硬件故障信息，导致操作系统日志分析和维护系统分析无法正常定位，只能通过运行时错误定位技术进行错误诊断. 其中基于运行时库的错误探测分析技术可以对结果错和消息丢包课题进行有效判断；基于全局聚合信息的综合诊断由于依赖错误信息分析，无法进行程序挂死和结果错的故障定位；异常线程过滤方法弥补了上述 2 个定位技术的缺项，通过核心 PC 值的判断，较好地定位了程序挂死问题.

通过测试，运行时错误定位和传统的故障分析

Table 2 Comparison of Error Location Time

表 2 错误定位时间对比 s

测试集	传统错误定位				运行时故障定位		
	故障类型	操作系统分析	维护系统分析	MPFL 故障分析	基于运行时库的错误探测分析	基于全局聚合信息的综合诊断	异常线程过滤方法
MsgSet1	管理核心	1.7	2.5			18.4	2.4
	计算核心	2.3	3.2			18.4	3.3
	存储器	1.8	1.8			18.9	2.7
	网络接口	2.6	3.5	2.6	1.6	19.3	
	操作系统	1.6	2.8			18.7	2.4
MsgSet2	程序挂死						2.3
	结果错				1.7		
	性能异常					18.7	
	消息丢包			2.8	1.8	19.5	

方法相比,针对无明显故障信息的程序挂死等错误现象有较好的检测及诊断效果,并通过技术融合,比较准确全面地分析出了发生错误的现场.

4 结 论

当前,并行计算技术已经迈入E级计算时代,超级计算机的规模和复杂度越来越大,系统可靠性面临前所未有的挑战.故障定位作为可靠性管理的核心技术,对实现超级计算机的高可用有重要意义,但当前的故障定位技术存在手段单一、对无明显错误现象的故障缺乏有效定位方法的问题.

本文针对高性能计算系统中故障定位难度高且实时性差的问题,面向"新一代神威超级计算机"提出了一种运行时故障定位方法,包括基于消息传递的故障关联分析、基于全局聚合信息的在线综合分析诊断、面向申威众核处理器的异常线程过滤方法等关键技术.通过试验可以发现,并行运行时故障定位技术具有较好的可扩展性,针对无日志信息、无明显故障现象挂死等疑难问题有较好的适用性,可有效解决超大规模系统下的即时精准定位,提升系统平均无故障时间,为新一代超级计算的可靠性设计提供技术支撑.

参 考 文 献

[1] Zhang Yunquan, Yuan Liang, Yuan Guoxing , et al. Present situation and prospect of high performance computers in China in 2022[J] . Frontiers of Data & Computing, 2022, 4(6): 3−12 (in Chinese)
(张云泉, 袁良, 袁国兴, 等. 2022年中国高性能计算机发展现状分析与展望 [J]. 数据与计算发展前沿, 2022, 4(6): 3−12)

[2] Liu Qing, Logan J, Tian Yuan, et al. Hello ADIOS: The challenges and lessons of developing leadership class I/O frameworks[J]. Concurrency and Computation: Practice and Experience, 2014, 26(7): 1453−1473

[3] Liang Yinglung, Zhang Yanyong, Sivasubramaniam A, et al. BlueGene/L failure analysis and prediction models[C/OL]//Proc of the 36th Int Conf on Dependable Systems and Networks. Piscataway, NJ: IEEE, 2006[2022-10-08].https://ieeexplore.ieee.org/document/1633531

[4] Liang Yinglung, Zhang Yanyong, Xiong Hui, et al. Failure prediction in IBM BlueGene/L event logs[C]//Proc of the 7th Int Conf on Data Mining. Piscataway, NJ: IEEE, 2007: 583−588

[5] Yu Li, Zheng Ziming, Lan Zhiling , et al. Practical online failure prediction for BlueGene/P: Period-based vs event-driven[C]// Proc of the 41st IEEE/IFIP Int Conf on Dependable Systems & Networks Workshops. Piscataway, NJ: IEEE, 2011: 259−264

[6] Ostrouchov G , Maxwell D E , Ashraf R A , et al. GPU lifetimes on Titan supercomputer: Survival analysis and reliability[C]// Proc of the 39th Int Conf for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2020: 412−421

[7] Han Qiqi, Liu xin, Zeng Yunhui, et al. Design and implementation of ocean numerical operation management system[J]. Compute Applications and Software, 2020, 37(4): 6−11 (in Chinese)
(韩琦琦, 刘鑫, 曾云辉, 等. 海洋数值模式运行管理系统的设计与实现 [J]. 计算机应用与软件, 2020, 37(4): 6−11)

[8] Snir M , Wisniewski R W , Abraham J A , et al. Addressing failures in exascale computing[J]. International Journal of High Performance Computing Applications, 2014, 28(2): 129−173

[9] Dong H A , Supinski B , Laguna I , et al. Scalable temporal order analysis for large scale debugging[C]//Proc of the 28th Int Conf for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2009: 44−54

[10] Laguna I, Dong H A, Supinski B, et al. Probabilistic diagnosis of performance faults in large-scale parallel applications[C]// Proc of the 21st Int Conf on Parallel Architectures and Compilation Techniques. Piscataway, NJ: IEEE, 2012: 213−222

[11] Meuer H W. The TOP500 Project: Looking Back over 15 Years of Supercomputing Experience[J]. PIK−Praxis der Information-sverarbeitung und Kommunikation, 2008, 31(3): 203−222

[12] Cao Zongyan. Configuration optimization of high performance computing cluster runtime[J]. E-science Technology & Application, 2011, 2(6): 52−61 (in Chinese)
(曹宗雁. 高性能计算集群运行时环境的配置优化 [J]. 科研信息化技术与应用, 2011, 2(6): 52−61)

[13] Mitra S , Laguna I, Ahn D H , et al. Accurate application progress analysis for large-scale parallel debugging[C]// Proc of the 35th ACM SIGPLAN Conf on Programming Language Design and Implementation. New York: ACM, 2014: 193−203

[14] Zhang Guozhen, Liu Yi , Yang Hailong , et al. A lightweight and flexible tool for distinguishing between hardware malfunctions and program bugs in debugging large-scale programs[J]. IEEE Journal Article, 2018, 6(4): 71892−71905

[15] Gao Jian, Yu Kang, Qing Peng, et al. Distributed fault location framework for high performance computing[J]. Journal of Computer Applications, 2018, 38(1): 44−49
(高剑, 于康, 卿鹏, 等. 面向高性能计算的分布式故障定位框架 [J]. 计算机应用, 2018, 38(1): 44−49)

[16] Liu Qingrui , Jung C , Lee D , et al. Compiler-directed lightweight checkpointing for fine-grained guaranteed soft error recovery[C]// Proc of the 35th Int Conf for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2016: 228−239

[17] Fiala D, Mueller F, Engelmann C, et al. Detection and correction of silent data corruption for large-scale high-performance computing[C]// Proc of the 25th IEEE Int Symp on Parallel and Distributed Processing. Piscataway, NJ: IEEE, 2011: 2069−2072

[18] Gao Jiangang, Lu Hongsheng, He Wangquan, et al. Interconnection

network and message mechanism of Sunway E-class prototype[J]. Chinese Journal of Computers, 2021, 44(1): 222−234 (in Chinese)
(高剑刚, 卢宏生, 何王全, 等. 神威 E 级原型机互连网络和消息机制 [J]. 计算机学报, 2021, 44(1): 222−234)

[19] Jian Lantao, Ren Xiujiang, Zhang Zhen, et al. Research on maintenance fault diagnosis system of E-class high performance computer[J]. Computer Engineering, 2022, 48(12): 24−37 (in Chinese)
(建澜涛, 任秀江, 张祯, 等. E 级高性能计算机的维护故障诊断系统研究 [J]. 计算机工程, 2022, 48(12): 24−37)

[20] Hong Wenjie, Li Kenli, Quan Zhe, et al. PETSc's heterogeneous parallel algorithm design and performance optimization on the Sunway TaihuLight system[J]. Chinese Journal of Computers, 2017, 40(9): 2057−2069 (in Chinese)
(洪文杰, 李肯立, 全哲, 等. 面向神威·太湖之光的 PETSc 可扩展异构并行算法及其性能优化 [J]. 计算机学报, 2017, 40(9): 2057−2069)



Gao Jiangang, born in 1963. Master, professor. Senior member of CCF. His main research interests include high performance computing and computer architecture.
高剑刚, 1963 年生. 硕士, 正高级工程师. CCF 高级会员. 主要研究方向为高性能计算、计算机体系结构.



Zheng Yan, born in 1977. Bachelor, professor. His main research interests include high performance computing and operating system.
郑　岩, 1977 年生. 学士, 正高级工程师. 主要研究方向为高性能计算和操作系统.



Yu Kang, born in 1987. PhD, assistant professor. His main research interests include parallel computing and runtime system.
于　康, 1987 年生.博士, 助理研究员.主要研究方向为并行计算、运行时系统.



Peng Dajia, born in 1992. PhD candidate, engineer. His main research interests include parallel computing and debugging.
彭达佳, 1992 年生. 博士研究生, 工程师. 主要研究方向为并行计算、调试.



Li Hongliang, born in 1975. PhD, professor. His main research interests include high performance computing and computer architecture.
李宏亮, 1975 年生. 博士, 研究员. 主要研究方向为高性能计算、计算机体系结构.



Liu Yong, born in 1981. PhD, associate professor. His main research interest includes parallel algorithms.
刘　勇, 1981 年生. 博士, 副研究员. 主要研究方向为并行算法.



He Wangquan, born in 1981. PhD, professor. His main research interests include high performance computing software and architecture.
何王全, 1975 年生. 博士, 研究员. 主要研究方向为高性能计算软件、体系结构.



Chen Dexun, born in 1973. PhD, professor. His main research interests include high performance computing software and architecture.
陈德训, 1973 年生. 博士, 研究员. 主要研究方向为高性能计算软件、体系结构.



Wang Fei, born in 1981. Master, professor. His main research interests include high performance computing and compiler.
王　飞, 1981 年生. 硕士, 研究员. 主要研究方向为高性能计算、编译器.