doi:10.11887/j.cn.201601016 http://journal.nudt.edu.cn

E级超级计算机故障预测的数据采集方法*

胡 维[1,2],蒋艳凰[1],刘光明[1,2],董文睿[1,2],崔新武[3]
(1.国防科技大学 计算机学院,湖南 长沙 410073;2.国家超级计算天津中心,天津 300457;
3.中国人民解放军95942部队,湖北 武汉 430313)

摘 要:面向未来E级超级计算机,提出用于故障预测的数据采集框架,能够全面采集与计算结点故障相关的状态数据。采用自适应多层分组数据汇集方法,有效解决随着系统规模增长数据汇集过程开销过大的问题。在TH-1A超级计算机上的实现和测试表明,该数据采集框架具有开销小、扩展性好的优点,能够满足未来大规模系统故障预测数据采集的需求。

关键词:超级计算机;故障预测;数据采集方法;数据汇集
中图分类号:TP311 **文献标志码**:A **文章编号**:1001-2486(2016)01-093-08

Data collection for failure prediction toward exascale supercomputers

HU Wei[1,2], *JIANG Yanhuang*[1], *LIU Guangming*[1,2], *DONG Wenrui*[1,2], *CUI Xinwu*[3]
(1. College of Computer, National University of Defense Technology, Changsha 410073, China;
2. National Supercomputer Centre in Tianjin, Tianjin 300457, China; 3. The PLA Unit 95942, Wuhan 430313, China)

Abstract: Aimed at an exascale supercomputer, an FPDC (failure prediction data collection framework) was introduced to fully collect the data related to the state of compute nodes' health. An adaptive multi-layer data aggregation method was presented for data aggregation with less overhead. Extensive experiments, by implementing FPDC on TH-1A, indicate that the FPDC has the advantage of high efficiency and good scalability.

Key words: supercomputer; failure prediction; data collection method; data aggregation

超级计算机的飞速发展面临许多挑战,可靠性问题成为影响系统性能发展的重要挑战之一。未来E级超级计算机由数十万个部件组成,系统平均无故障时间(Mean Time Between Failure, MTBF)将从小时级到分钟级[1]。消息传递接口(Message Passing Interface, MPI)是超级计算机应用的主要并行方式,若有一个进程出现故障,则整个应用都被迫停止并从头开始。检查点技术是目前超级计算机系统中最常用的容错方法。随着超级计算机规模不断扩大,MTBF时间逐渐缩短,保存检查点的时间间隔越来越短;而超级计算机I/O系统性能发展缓慢,保存和恢复检查点的开销越来越大,检查点技术将无法满足系统可靠性的需求。

高性能计算容错方式通常分为被动容错和主动容错两种。被动容错是在故障发生后再实施容错,典型的就是检查点技术。主动容错通过故障预测的方法提前预知故障的发生,在故障发生前预先采取进程迁移、进程复制等低开销保护性技术,保障并行应用持续运行。主动容错技术因开销小,成为解决未来E级超级计算机可靠性挑战最有希望的技术之一,其中故障预测的准确率直接决定着主动容错的有效性。现有用于主动容错的故障预测方法主要包括基于模型的故障预测和数据驱动的故障预测两类。

基于模型的故障预测方法将系统实际执行行为与模型描述的预期行为进行比较,通过发现明显行为差异来预测系统故障。该方法仅适用于小规模系统中某些类型的故障,对于复杂的大规模系统,难以用模型准确描述系统的故障特征。数据驱动的故障预测方法利用数据挖掘、机器学习等技术对历史数据进行学习,获取故障发生的规律,并利用学习结果对系统进行实时状态数据分析,预测是否有故障发生。这类方法的重要基础

* **收稿日期**:2015-04-09
基金项目:国家自然科学基金资助项目(61272141,61120106005);国家863计划资助项目(2012AA01A301)
作者简介:胡维(1982—),男,江西南昌人,博士研究生,E-mail:huwei@nscc-tj.gov.cn;
刘光明(通信作者),男,教授,硕士,博士生导师,E-mail:liugm@nscc-tj.gov.cn

是获取与系统故障相关的运行状态数据,这些数据直接影响着故障预测的准确性。

目前超级计算机系统故障预测研究中采用的数据主要包括两种:一种是可靠、可用和可维护性(Reliability, Availability, and Serviceability, RAS)日志数据;一种是硬件环境和结点运行状态数据。

RAS 数据通常是超级计算机监控系统定时对系统中各部件的运行状态进行扫描监测,将可能有用的数据(如异常事件)保存在日志中,其通常作用是在系统发生故障后,管理员通过查询日志内容,对故障进行人工诊断,现有的故障预测研究大多是基于 RAS 日志数据展开的[2-6]。由于 RAS 日志数据本身是对软硬件事件的记录,一方面信息记录不完全,只记录事件发生信息而没有软硬件随时间变化的状态信息,容易使故障预测出现漏报;另一方面,由于系统运行状态复杂,日志事件定义不可能完全准确,容易使预测产生误报。因此,基于 RAS 数据进行的故障预测研究,预测精度较低,学习结果的可理解性较低。

硬件环境状态数据包括系统硬件各部件的温度、电压、风扇和电源状态等信息。研究者[7-9]通过智能平台管理界面(Intelligent Platform Management Interface, IPMI)获取数据,进行故障预测研究。结点运行状态数据通常指超级计算机计算结点运行过程中,结点 CPU、内存、网络和 I/O 等系统的运行状态数据。由于大部分超级计算机计算结点具有同构性,运行应用具有相似性,所以结点运行过程中的系统状态信息能够反映结点的健康状况。Sahoo 等[10]结合日志记录和状态数据进行故障预测的研究。使用结点运行状态数据进行故障预测的研究较少,主要原因是数据采集困难。虽然现有集群系统监控工具 PARMON[11]、Ganglia[12]和 Ovis - 2[13]等同时具备数据采集功能,但采集数据属性少、开销大,无法满足故障预测的实际需要。

从上面可以看出,现有数据获取方法具有如下缺陷:一是所采集数据的属性少,无法反映出系统运行状态的变化;二是所采集数据的时间连续性差,不能满足故障预测精度的要求。针对以上不足,提出用于故障预测的数据采集框架(Failure Prediction Data Collection Framework, FPDC),以解决数据采集的全面性和有效性问题。由于文章篇幅限制,主要介绍面向故障预测数据采集面临的挑战和解决方法,对故障预测方法的研究另行论述。

1 FPDC 数据采集框架

超级计算机主动容错系统的核心是故障预测,而数据采集是故障预测的基础,故障预测的准确性不仅与故障预测模型有关,还与用于预测的数据密切相关。

FPDC 数据采集框架具有两方面功能:一是在初始故障学习阶段,累积一定时间段的数据形成初始训练集,用于学习产生初始的故障预测分类器;二是在故障预测阶段,实时获取系统状态数据用于实时故障预测,并对故障预测分类器进行在线学习更新。

1.1 FPDC 框架及其功能

图 1 为 FPDC 数据采集框架体系结构,FPDC 框架采用分布式结构,获取数据过程分为数据采集和汇集两个部分。

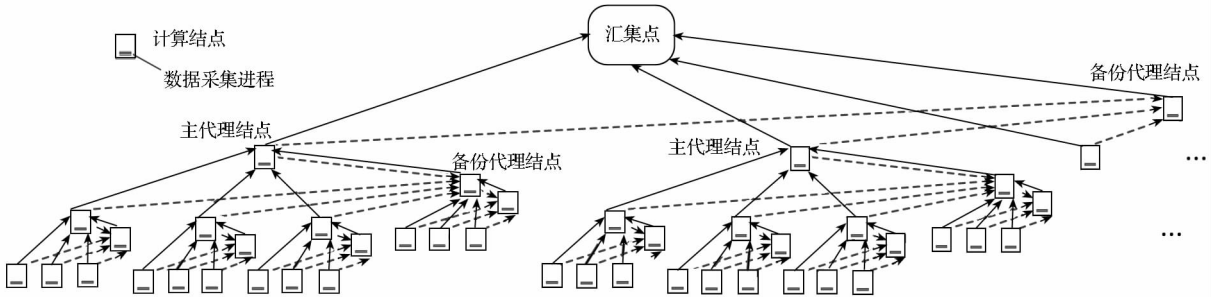

图 1 FPDC 数据采集框架体系结构

Fig. 1 FPDC architecture

数据采集部分将数据采集任务分布到每个计算结点上,结点运行轻量级数据采集进程,按照一定的系统配置要求,周期性采集结点状态数据,这种分布式采集方式能够全面获取与计算结点故障相关的状态数据,而采集开销较小。

数据汇集时,采用自适应多层分组数据汇集方法,结点通过分组形成层次式树形结构,数据获取后采用 Push 协议,结点主动向上一级发送数

据。采用层次式分组的方法,能够减少频繁小数据传输,从而减少通信和存储资源消耗,同时能够避免数据直接汇集时的单点瓶颈,有效利用高速互连带宽,减少共享存储系统 I/O 开销,提高数据汇集的可扩展性。FPDC 在每个结点上以一定频率进行周期性数据采集,数据获得后汇集到最终汇集点。其中,结点上数据采集进程的开销是固定的,数据汇集开销对整个数据采集的可扩展性起到决定性作用,而自适应多层分组数据汇集方法能够有效减少开销,提高可扩展性。

通过采集获取的结点状态数据具有空间性和时间性。空间性是指采集的数据内容能否覆盖所有可能发生的故障。计算结点出现故障可能由计算结点的硬件部件引起,也可能由软件错误引起。FPDC 数据采集的内容从硬件和软件两个方面出发,获取不同硬件部件和不同软件层次的各方面状态数据,提高对所有故障的覆盖率。时间性是指所采集的数据能否有效体现结点状态随时间变化的全过程,并满足故障预测提前性的要求。FPDC 采用分布式架构和自适应多层分组数据汇集方法,开销小,对大规模系统扩展性高,保证了数据采集时间性的需求。FPDC 框架采集数据的空间性和时间性是提高故障预测精度的基础。

对计算结点硬件和软件采集的数据分别对应结点硬件环境状态数据和系统运行状态数据。如图 2 所示,FPDC 主要由硬件环境数据采集模块、运行状态数据采集模块和数据汇集模块组成。

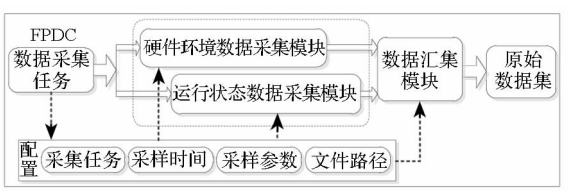

图 2　FPDC 组成模块和功能

Fig. 2　FPDC modules and functions

硬件环境数据采集模块用于采集与计算结点硬件环境状态相关的数据,主要包括结点硬件各个部件的温度、电压,风扇和电源状态等数据,能够反映硬件部件的实时物理状态。运行状态数据采集模块用于采集与结点系统运行相关的状态数据,即结点操作系统活动报告(System Activity Report, SAR)数据,这些数据包括 CPU、内存、网络和 I/O 等系统的状态数据和统计数据。数据汇集模块用于完成数据采集后向最终汇集点的数据传输工作。

1.2　自适应多层分组数据汇集方法

现有数据汇集方法主要包括直接汇集法和分组汇集法,这些方法存在数据汇集开销大[12]、结点发生故障时关键状态数据丢失的问题。自适应多层分组数据汇集方法,能保证关键状态数据不丢失,并有效降低传输开销,提供良好可扩展性。

设计思想:如图 1 所示,自适应多层分组数据汇集方法借鉴多叉树结构,根结点是最终汇集点(管理结点或共享存储),其余的每个结点代表一个计算结点,非叶子结点代表代理结点。其本质是利用计算结点间的高速互连带宽优势,通过分组将数据分层收集和压缩,对于最终汇集点是管理结点的结构,能够有效缓解多对一汇集时的单点瓶颈,对于最终汇集点是共享存储的结构,可以有效节省共享存储系统的 I/O 资源,减少并行文件系统开销。后续论述汇集点以共享存储为例。

通过对结点本身计算和网络负载的分析,选择低负载结点作为代理结点,并针对结点负载的变化,定期自适应地改变代理结点和备份结点,能够减少对结点上应用的影响。同时,备份代理结点能够在主代理结点故障时,保存其状态数据,保证重要状态数据不丢失。

算法描述:自适应多层分组数据汇集方法类似于多叉树结构,根结点是共享存储,其余每个结点代表一个计算结点,分为两类:一是叶子结点,仅采集自身数据,并将数据发送给父主代理结点和父备份代理结点;二是非叶子结点,即代理结点,在每个组中,均包含一个主代理结点和一个备份代理结点,代理结点不仅要获取本结点数据还要汇总子结点数据,而后向父结点传输。计算结点简称 cn(compute node),每组中选出主代理结点 cna(compute node agent)和备份代理结点 bcna(backup of compute node agent),bcna 用于存储组内 cn 和 cna 数据的备份,收集和压缩数据后并不向父结点传输,只有在 cna 出现故障时接替 cna 将数据向父结点传输。各组根据结点负载选举 cna 和 bcna 流程如下:

1)每个分组中 cn 计算自身总空闲率(总空闲率 idle% = cpu 空闲率 + 内存空闲率 + 网络带宽空闲率 + I/O 带宽空闲率),而后向组中广播由自身结点名和空闲率组成的二元组(name, idle%);

2)每个 cn 收到其他结点广播的结点名和空闲率二元组,依据空闲率对二元组进行降序排序,确定排在第 1 位的 cn 为 cna,排在第 2 位 cn 为 bcna,空闲率相等的以结点编号数字小的排在前

面,并依次向 cna 和 bcna 进行投票,即向排序位于 1 和 2 的结点发送 cna 和 bcna 投票信息;

3)收到 cna 和 bcna 投票信息的结点各自进行计数,若结点获得投票数达到组内结点数一半以上,向所有组员和管理结点注册 cna 和 bcna;

设选举间隔时间计数器为 T_e,采样间隔时间为 T_{col},则自适应多层分组数据汇集算法流程见图 3。

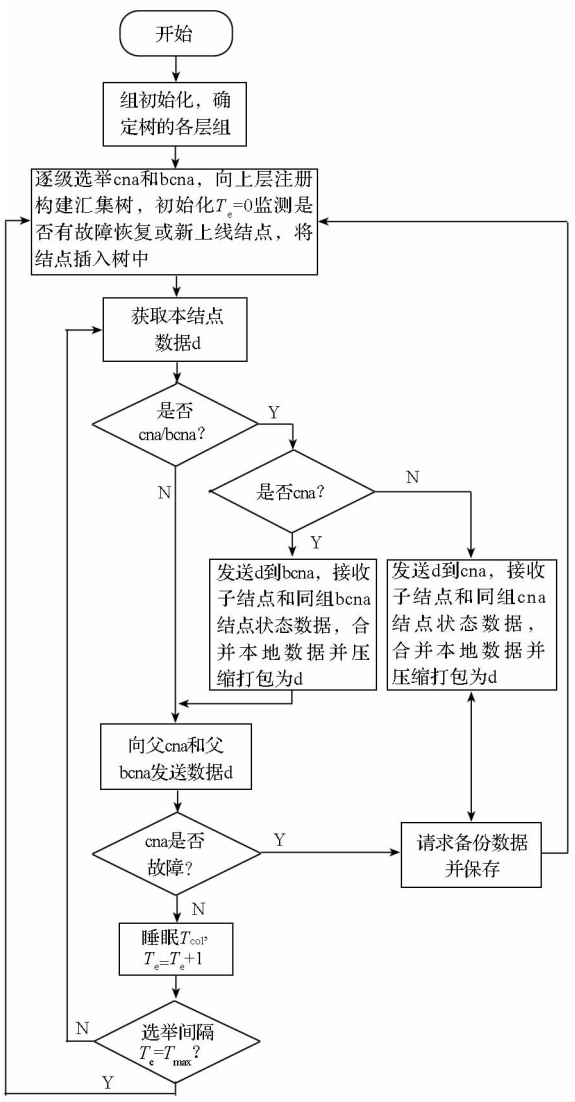

图 3 自适应多层分组数据汇集流程图

Fig. 3 Flow chart of adaptive multi-layer data aggregation algorithm

算法分析:自适应多层分组数据汇集方法,能够有效利用超级计算机高速互连的优势,将大量分散的小数据整合压缩,以通信资源换 I/O 资源,减少 I/O 开销,提高效率。但随着系统规模增大,如果数据汇集过程中层次太多,会造成数据多次重复传输,增加数据传输开销,所以需要针对系统规模具体实际,有效平衡多叉树的深度和宽度,以较小开销获得较好的数据采集性能。同时,可利用超级计算机自身的网络拓扑特点进行分组,能够显著减少网络开销。

从上述分析得出,自适应多层分组数据汇集方法具有一定优势:1)数据汇集路径没有交叉,可同步进行;2)数据网络传输趋于分散平衡,能够减少网络拥塞;3)数据分组传输过程中设置备份代理结点,能够保证故障结点关键状态数据不丢失;4)能够自适应选择代理和备份结点,减少对负载较重计算结点上应用的影响;5)对采集数据进行压缩,能够减少网络传输开销。

2 基于 TH-1A 系统的数据采集设计与实现

2.1 TH-1A 超级计算机及其数据采集框架

图 4 为国防科学技术大学研制的 TH-1A 系统,采用 CPU 和 GPU 结合的异构融合体系结构,全系统包含 7168 个计算结点,每个计算结点含 2 路英特尔 CPU 和 1 路英伟达 GPU。采用自主设计互连通信系统,实现光电混合胖树结构高阶路由网络[14],采用麒麟 Linux 系统。FPDC-TH 数据采集框架是 FPDC 在 TH-1A 上的具体实现。FPDC-TH 包括硬件环境数据采集模块 SMCcolmanager、运行状态数据采集模块 noderuninfomanager 和数据汇集模块 dataaggregation。

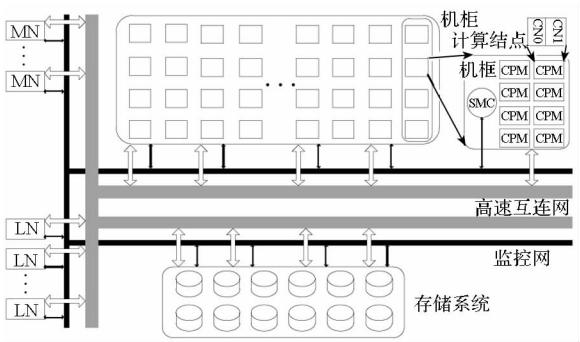

图 4 TH-1A 系统结构

Fig. 4 TH-1A architecture

2.2 硬件环境数据采集

TH-1A 计算机系统每个计算机柜包括 4 个计算机框,分别由 4 块系统管理控制器(System Management Controller, SMC)进行监控;通信机柜包括 2 个通信机框,分别由 2 块 SMC 板进行监控,如图 4 所示。由于计算结点没有单独映射 SMC,不能通过计算结点操作系统访问 SMC 接口,但 SMC 提供了网络访问接口,可通过提供 IP 地址实现远程监测控制 1 个机框的 16 个计算结点。基于维护控制网络和 SMC 系统,设计实现了 SMCcolmanager 硬件环境数据采集模块。

SMCcolmanager 采用 Client-Server 的结构,可一次获取每个机框 SMC 板服务器维护的机框内 16 个计算结点的硬件环境状态数据记录。利用 TCP/IP 套接字,采用多线程的方式并行采集多机框 SMC 数据,减少在多结点数据采集时对管理结点的资源占用,避免了管理结点瓶颈问题。由于维护控制网络是专用以太网络,这种远程访问 SMC 采集数据的方式对结点应用性能没有影响。

SMCcolmanager 模块能够全面采集结点硬件环境状态数据。表 1 为该模块在 TH - 1A 上获得与计算结点相关的硬件环境状态数据,包括机框内风扇转速、网络路由芯片(Network Pouting, NR)状态、网络接口芯片(Network Interface, NI)状态、计算结点电源状态、SMC 监控板状态和计算结点状态数据。

表 1　硬件环境状态数据

Tab. 1　Hardware status data

状态类别	硬件状态属性数目					
	风扇	NR	NI	电源	SMC 板	计算结点
转速	6			2		
温度		1	1	1	4	
电压		3	5	2	4	12
电流				2		3
合计	6	4	6	7	8	15

2.3　运行状态数据采集

FPDC-TH 采用分布式结构采集运行状态数据,每个计算结点运行数据采集进程,通过多层分组数据汇集方法将数据汇集至代理结点。

noderuninfomanager 数据采集模块,采用分析/proc 文件系统的方法获取结点运行状态数据。/proc 虚拟文件系统是 linux 内核的一部分,提供用户动态查看内核运行状态的接口,包括当前系统中进程、硬件、内存等相关信息。通过分析/proc 中 cpuinfo、meminfo、slabinfo、uptime、net/、sys/、scsi/等文件或文件夹中相关文件,能够获得包括 CPU、内存、网络和 I/O 等系统运行的信息。为了提高数据分析和采集的效率,采用先并行读取/proc 相关文件,后整合结点运行状态数据的方法,这种方式开销小,采样间隔可达到毫秒级。

noderuninfomanager 模块通过对 TH - 1A 中/proc文件系统的分析,选择采集与结点运行状态密切相关的 136 个数据,见表 2,主要包括下面四部分。

1)CPU 相关:CPU 使用负载情况,任务创建和系统切换活动,中断统计,队列深度等;

2)内存相关:内存使用情况,包括内存的利用率,内存页替换和缓存的速率等;

3)网络相关:网络参数统计,包括包速率、带宽、网络设备错误、socket 统计、IP 网络通信量和错误统计、ICMP 网络通信量和错误统计、TCP 网络通信量和错误统计、UDP 网络通信量等;

4)I/O 相关:详细的物理设备传输速率统计,文件系统统计和 Lustre 客户端访问速率统计等。

表 2　结点运行状态数据

Tab. 2　Running status data

不同类别数据数目			
CPU 相关	内存相关	网络相关	I/O 相关
21	22	78	15

2.4　数据汇集

结合 TH - 1A 计算机系统的实际结构,自适应多层分组数据汇集方法适用于结点运行状态数据的汇集。首先将最终汇集点确定为 TH - 1A 共享存储系统 Lustre,然后根据计算结点规模和高速互连拓扑结构特点确定具体的分组方法。

数据汇集模块 dataaggregation 在 TH - 1A 上实现时,分组层次选择为 3 层,从下至上依次为叶结点层、代理层和 Lustre 存储层,如图 5 所示。以机框为单位(16 个结点),每个机框选出主代理结点 cna 和备份代理结点 bcna,其中叶子结点(14 个结点)为叶结点层,代理结点 cna 和 bcna 为叶结点的父结点,系统中所有的这些代理结点组成代理层,最终数据汇集到 Lustre 共享存储。

由于最终汇集点是 Lustre 共享存储,该层不再设置备份结点。

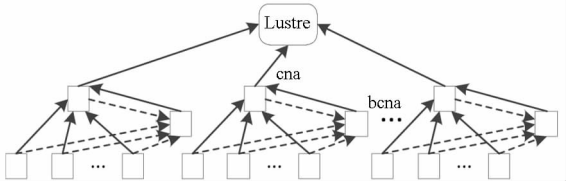

图 5　FPDC - TH 数据汇集示意图

Fig. 5　Data aggregation workflow of FPDC - TH

dataaggregation 模块采用如图 5 所示的 3 层分组数据汇集方法基于如下考虑:

1)TH - 1A 超级计算机全系统共有 7168 个计算结点,计算结点数不超过 10^4,采用 3 层的分组方式,能够有效缩减单位时间内计算结点直接访问共享存储的访问数,使存储系统开销减少到

较小的程度。

2)利用机框进行分组可以有效利用超级计算机互连网络拓扑结构的优势。TH－1A 互连网络为光电混合层次式胖树结构,第一层为机框内部电互连,机框通信交换板和 16 个结点之间通过背板电互连;第二层为机柜内部机框之间的光连接,采用 Mesh 结构;第三层由 11 个 384 口交换机组成,机框交换板和 384 口交换机采用光互连,组成胖树结构。这种层次式胖树结构结点之间互连传输速度决定于点对点之间的跳步数和通信距离。同一机框内部结点之间通信跳步数为 1 或 3;而在机框之间,不同机柜间的结点跳步数逐渐增多,最多为 11 跳,同时,传输距离不断增加。因此,以一个机框为基本分组单位,结点间数据传输速度最快、开销最小。

3 性能测试与分析

基于 TH－1A 超级计算机对 FPDC-TH 数据采集系统进行测试,该系统计算结点包括 2 个英特尔至强 X5670 处理器(2.93GHz,6 核),24G 内存,Linux 内核版本为 2.6.32。实验过程中默认数据采集间隔为 10 s。

3.1 硬件环境数据采集开销分析

SMCcolmanager 模块通过专用以太网访问 SMC 获取数据,对计算结点性能没有影响,仅测试对管理结点性能影响。通过多次运行 ps 命令和 vmstat 命令求平均值的方法获得数据采集开销。图 6 所示为管理结点分别从 112,224,336 和 448 个 SMC 服务器(全系统共 448 个 SMC)获取数据开销的比较。从测试可知,虚拟内存和物理内存的使用随着采集 SMC 服务器数目的增加而增加,主要是由于 SMCcolmanager 模块访问每个 SMC 服务器需要一个线程。但实际内存开销不大,小于总内存的 0.02%。同时处理器开销较小,低于 0.4%;I/O 开销低于 3.8 MB/s。另外,从图 6 可知,随着数据采集规模成倍增加,采集开销增长缓慢,具有良好的扩展性,为后续部署到 TH－2 或更大规模系统提供了较好的依据。总体来看,硬件环境数据采集对管理结点开销较小,同时并不影响系统实际应用的性能。

3.2 结点运行状态数据采集开销分析

表 3 所示为 noderuninfomanager 模块在单一叶计算结点上采集运行状态数据的开销。Valgrind 是 Linux 环境下对应用程序的内存分析工具集。其包含的 Massif 内存剖析工具能够检测

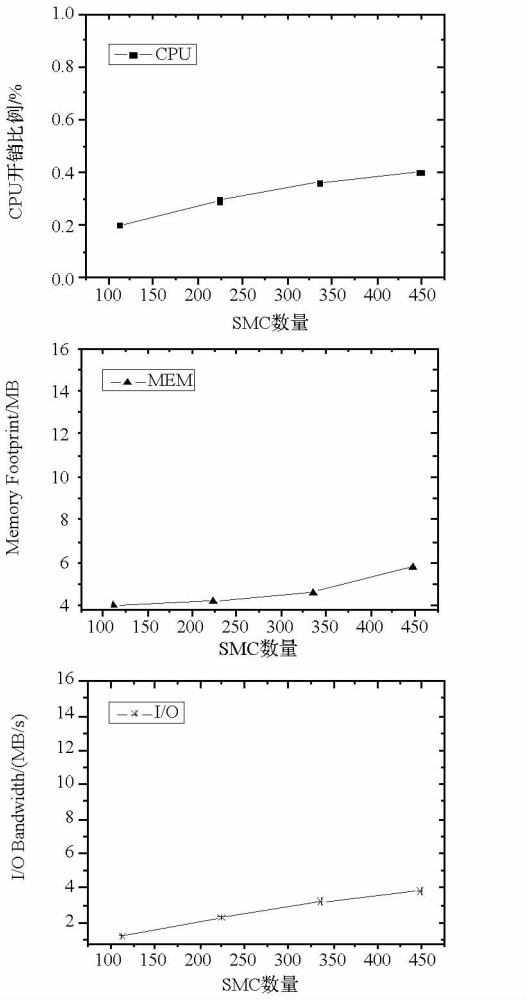

图 6 硬件环境数据采集模块可扩展性测试图

Fig. 6 Scalability test of SMCcolmanager

程序运行过程中堆内存(heap memory)和全部占用内存。利用/usr/bin/time 命令多次测试求平均值的方法获得时间和 CPU 开销,利用 Massif 工具多次测试求平均值的方法获取内存占用开销。从数据可知,计算结点上的数据采集进程占用的 CPU、内存和带宽等开销较小。叶子结点将数据传输到主代理结点和备份代理结点时,并不产生 I/O 开销,数据传输时间小于 1 ms。将传输数据量除以数据采集间隔(10 s)作为数据传输的带宽,则占用数据带宽为 0.14 KB/s,该带宽值忽略了传输协议的开销,可以认为是占用带宽的最小值,可以反映带宽开销水平。

表 3 noderuninfomanager 在叶结点上的数据采集开销

Tab. 3 Leaf node overhead of noderuninfomanager

CPU/%		最大堆内存/KB	最大内存/MB	数据传输/ms	带宽/(KB/s)
采集时	采集间隔				
<0.6	0	86.97	10.57	<1	0.14

3.3　自适应分组数据汇集开销分析

dataaggregation 数据汇集模块采用自适应 3 层分组数据汇集方法,以机框为基本分组单位,利用 TH－1A 互连网络光电混合层次式胖树拓扑结构特点,使得较多的数据传输发生在跳数较少和距离较近的结点之间,有效减少了数据分组传输过程中的通信开销。如图 7 所示,利用 Glex 接口编程采用乒乓(ping-pong)测试方法,在机框内部、机柜内部和机柜之间分别选择跳步数为 1,5 和 10 的结点对,传输数据从 8 B 增加到 4 KB。由图 7 中测试结果可知,随着通信跳步数的增多和通信距离的增加,通信延迟变长,带宽变小。因此,将较频繁的数据传输控制在机框内,数据可通过背板直接传输,传输速率高、开销小。

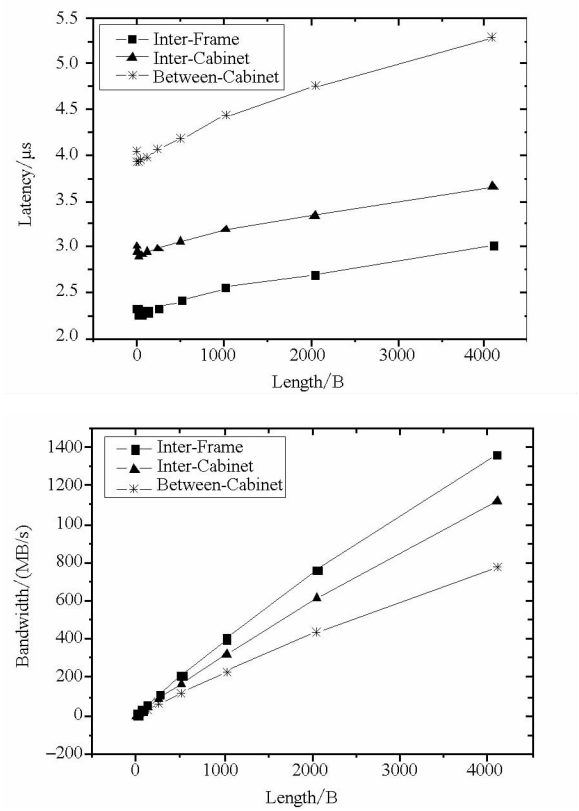

图 7　数据汇集网络测试

Fig. 7　Network test of data aggregation

表 4 和表 5 分别是 dataaggregation 模块在主代理结点数据压缩和数据传输的开销,压缩采用 zlib 库,实验采用/usr/bin/time 和 Massif 多次测试求平均值的方法分别获得时间、CPU 和内存开销。表 4 分别列出了主代理结点收集 16,8 和 4 个结点的数据进行压缩时的开销,可知 3 种情况下压缩开销差别很小,但以 16 个结点进行分组,能够有效减少 I/O 和通信开销。表 4 中 16 结点组 CPU 开销小是因为这里 CPU 开销指压缩操作耗时内的 CPU 开销,由于 16 个结点数据压缩耗时相对较长,所以 CPU 占用率相对较低。同时观察到,压缩过程具有一定的内存开销,但内存开销仅存在压缩操作运行的这个较短时间内(<4 ms),从时间角度来看,开销也不大。

表 4　主代理结点数据压缩开销

Tab. 4　Master agent node's data compression overhead

分组规模	压缩耗时/s	压缩时间段内 CPU/%	占用最大堆内存量/KB	占用最大内存/MB
16	0.004	6.25	35.23	117.5
8	0.003	10.16	35.02	117.5
4	0.003	10.16	35.02	117.5

表 5 中分别列出了不同分组方式数据压缩后向 Lustre 存储系统传输时主代理结点的开销,由于采集数据本身的特点,具有较高压缩比,单一结点采集数据原始大小约 700 ~ 750 B。以 16,8 和 4 个结点进行分组压缩数据后,数据传输时间仅相差 0.000 7 s。在数据传输过程的时间段内 16 个结点组压缩数据的 CPU 开销最大,是由于传输时间增长不多的情况下,传输较多数据占用了相对较多的 CPU 时间;但对于较短的传输时间,这个开销是可以接受的。对于不压缩的情况,数据传输时间明显增加,从表 5 中可知,由于压缩时间加上传输时间仍明显小于不压缩传输的时间,因此数据传输采用压缩方式有明显的时间收益。

图 8 为采用不同的分组方式对共享存储系统的开销情况。从图 8 可得,通过分组层次式汇集数据,能够迅速减缓数据汇集过程中的 I/O 请求数随结点规模增长的速度,可有效提高数据采集系统的可扩展性。数据存储到共享存储系统后,针对实际需求,进行数据处理工作,保存有效数据,降低存储空间开销。

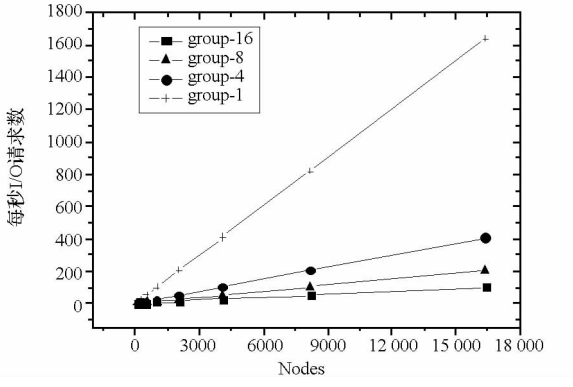

图 8　分组数据汇集 I/O 操作比较

Fig. 8　Comparison of I/O operation in data aggregation

表 5 主代理结点数据传输开销

Tab. 5 Master agent node's data transfer overhead

分组	数据压缩后大小/B	压缩比	代理结点向 Lustre 传输时间/s	压缩时间/s + 传输时间/s	传输数据时间段内 CPU 开销/%	使用最大堆内存量/KB	占用最大内存/MB
16 个结点为一组压缩	2755	4.21	0.006 3	0.010 3	3.41	0	5.875
8 个结点为一组压缩	1457	3.98	0.005 6	0.008 6	1.93	0	5.875
4 个结点为一组压缩	775	3.75	0.005 6	0.008 6	1.48	0	5.875
16 个结点为一组未压缩	11 610	1	0.066	0.066	1.33	0	5.875

4 总结

基于高性能计算故障预测数据采集的需要，提出数据采集框架 FPDC，能够获取与故障相关的结点软硬件状态数据，其分布式架构和自适应多层分组数据汇集方法有效解决了随着系统规模增长数据采集开销过大的问题。在 TH－1A 超级计算机上完成 FPDC 的实现，实验结果显示，FPDC 开销小，扩展性好，能够适应未来大规模系统故障预测数据采集的需要。

参考文献(References)

[1] Philp I R. Software failures and the road to a petaflop machine[C]// Proceedings of the 11th International Symposium on High Performance Computer Architecture, San Francisco, CA, USA, IEEE Computer Society, 2005.

[2] Liang Y, Zhang Y, Xiong H, et al. Failure prediction in IBM BlueGene/L event logs[C]//Proceedings of Seventh IEEE International Conference on Data Mining Omaha, Nebraska, USA, IEEE Computer Society, 2007:583－588 .

[3] Lan Z L, Gu J X, Zheng Z M, et al. A study of dynamic meta-learning for failure prediction in large-scale systems[J]. Journal of Parallel and Distributed Computing, 2010, 70(6): 630－643.

[4] Oliner A, Ganapathi A, Xu W. Advances and challenges in log analysis[J]. Communications of the ACM , 2012, 55(2): 55－61.

[5] Xu W, Huang L, Fox A, et al. Detecting large-scale system problems by mining console logs[C]//Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, New York, NY, USA: ACM, 2009.

[6] Gainaru A, Cappello F, Snir M, et al. Fault prediction under the microscope: a closer look into HPC systems[C]// Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Los Alamitos, CA, USA, IEEE Computer Society Press, 2012.

[7] Scott S L, Engelmann C, Vallée G R, et al. A tunable holistic resiliency approach for high-performance computing systems[C]//Proceedings of the 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, New York, NY, USA, ACM, 2009.

[8] Nagarajan A B, Mueller F, Engelmann C, et al. Proactive fault tolerance for HPC with Xen virtualization[C]// Proceedings of the 21st Annual International Conference on Supercomputing, New York, NY, USA, ACM, 2007: 23－32.

[9] Rajachandrasekar R, Besseron X, Panda D K. Monitoring and predicting hardware failures in HPC clusters with FTB-IPMI[C]//Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012: 1136－1143.

[10] Sahoo R K, Oliner A J, Rish I, et al. Critical event prediction for proactive management in large-scale computer clusters[C]//Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM, 2003:426－435.

[11] Buyya R. PARMON: a portable and scalable monitoring system for clusters[J]. Software-Practice Experience, 2000, 30(7): 723－739.

[12] Massie M L, Chun B N, Culler D E. The ganglia distributed monitoring system: design, implementation, and experience[J]. Parallel Computing. 2004, 30(7): 817－840.

[13] Brandt J M, Debusschere B J, Gentile A C, et al. Ovis－2: a robust distributed architecture for scalable RAS[C]// Proceedings of IEEE International Symposium on Parallel & Distributed Processing, IEEE Computer Society, 2008:1－8.

[14] Xie M, Lu Y T, Wang K F, et al. Tianhe－1A interconnect and message-passing services[J]. IEEE Micro, 2012, 32(1): 8－20.