文章编号: 1001- 2486(2005) 01- 0035- 04

基于网格的分布式仿真系统容错机制*

刘云生, 张 童, 张传富, 查亚兵
(国防科技大学 机电工程与自动化学院, 湖南 长沙 410073)

摘 要: 针对分布式仿真的需求, 在网格的基础上构建了通用的分布式仿真容错系统。该系统由三部分组成: 仿真资源状态监控模块、数据保存模块及错误恢复模块。其中仿真资源状态监控基于网格的 MDS 实现; 数据保存(包括进程空间、进程间交互关系的保存) 及错误恢复基于检查点机制在用户空间实现。就所增加的容错机制跟仿真系统原有功能模块的关系进行了分析。最后, 基于网格及上述容错模块设计并实现了一个 C/S 模式的容错代理, 用来实现仿真系统的自动容错。

关键词: HLA; 容错; 网格
中图分类号: TP391. 9 文献标识码: A

The Fault-tolerance Mechanism in Grid-based Distributed Simulation System

LIU Yun-sheng, ZHANG Tong, ZHANG Chuan-fu, ZHA Ya-bing
(College of Mechatronics Engineering and Automation, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Aiming at the demand of the distributed simulation system, this paper has built a common grid-based fault tolerance system. The system consists of three parts: simulation resource monitoring module, data saving module, and error recovery module. The implementation of monitoring module is built on top of grid's MDS, while data saving module, including the saving of the process space and the iterative relationship between processes, and fault recovery are realized based on checkpoint mechanism in the user space. In addition, we analyze the relationship between these three modules and the existing function modules in simulation system. In the end, we design and implement a fault tolerance broker in Client/Sever mode to automate the fault tolerance.

Key words: HLA; fault-tolerance; grid

基于 HLA 的分布式仿真系统在部队训练、战略战术研究、决策制定方面发挥着越来越大的作用, 但是无论HLA 规范还是当前主要的HLA 产品都没有提供相应的容错机制[1], 系统在可靠性方面的缺陷在一定程度上制约了仿真作用的发挥。因为随着仿真规模的扩大、仿真时间的增长, 系统的故障率将越来越高, 如果某个关键 LP(Logical Process) 或计算节点崩溃, 将会导致整个仿真系统停止运行, 而仅仅重启崩溃的LP 或节点又会导致系统状态的不一致, 这时惟一的方法就是重启整个仿真系统, 这对人力、物力及时间造成了极大的浪费, 特别是在一些细粒度、大规模、长时间跨度的分布式仿真中, 这种矛盾尤为突出, 因此, 迫切需要解决系统的容错问题。鉴于目前国内外在这方面做的研究相对较少[2~ 4], 本文试图就解决分布式仿真系统的容错问题提出一个初步的、相对完整的解决方案。

1 分布式仿真容错系统设计

由于分布式仿真系统的构成千差万别, 错误类型又各不相同, 而我们的研究不可能涵盖各个方面, 选取具有代表性的仿真系统模型将会使研究更有普遍意义, 为此对仿真系统模型做如下假设:

- 可靠的网络连接。
- 系统中发生的错误类型为失败停(fail-stop) 。

* 收稿日期: 2004- 09- 06
基金项目: 国家部委基金资助项目(51404010403KG0155)
作者简介: 刘云生(1976—) , 男, 博士生。

ъ 系统所有的仿真节点在任何时候都可以通过网络访问一个稳定的存储介质。

该容错系统基于进程迁移[7]机制实现, 主要包括如下三个模块: 资源状态监控模块、数据保存模块以及错误恢复模块。

ъ 资源状态监控模块: 对仿真系统中的软硬件进行监控。这种监控的目的是: ①获取资源状态信息, 并在错误发生时选择错误恢复节点; ②进行错误检测与定位。

ъ 数据保存模块: 对仿真进程空间及进程间的交互关系进行保存。

ъ 错误恢复模块: 利用数据保存模块提供的崩溃进程的数据及资源状态监控模块提供的仿真进程恢复节点来进行错误恢复。

上述模块之间的关系如图 1 所示, 由图中可见: 资源状态监控模块是容错的前提, 数据保存模块是容错的基础, 错误恢复模块是容错的最终实现。如果没有资源状态监控模块提供的信息, 错误就无从定位, 错误恢复模块也无法获得合适的错误恢复目标节点; 如果缺少数据保存模块, 错误恢复模块就没有足够的信息来重新启动已经崩溃的进程。

2 分布式仿真容错机制分析及实现

2.1 资源状态监控模块

2.1.1 资源状态信息获取

网格[5]为分布异构环境中的资源动态共享提供了一套完整的信息集成框架, 资源状态获取基于网格的信息服务(MDS[5])实现。在获取资源状态信息时, 我们感兴趣的信息主要有: CPU 类型、节点的平均负载、内存的使用情况、操作系统类型等。基于 MDS 提供的框架, 编写可获取上述信息的 information provider, 将其配置在本地的 GRIS[5]中, 并将该 GRIS 向 GIIS[5]注册, 然后就可以通过向 GRIS 或 GIIS 查询获得该节点的信息。图 2 就是利用信息查询 shell 命令 grid-info-research[5]向某个节点查询后返回的结果。显然, 这些信息可以帮助了解节点状态、选择错误恢复节点。

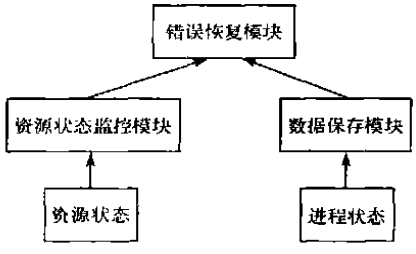

图 1 容错系统结构图

Fig. 1 Framework of fault tolerance system

objectClass:	MdsLibertySiteState
Mds-Known-GRIS-Node-CpuGeneral:	Intel®Pentium ®4CPU2.00GHz
Mds-Known-GRIS-Node-CpuLoad:	0.15
Mds-Known-GRIS-Node-MemFree:	346796032
Mds-Known-GRIS-Node-MemTotal:	526123008
Mds-Known-GRIS-Node-OS:	Linux2.4.20-8

图 2 仿真资源信息管理系统查询结果图

Fig. 2 The query result of simulation resource management system

2.1.2 错误监控与定位

错误的检测与定位通过“心跳”机制实现。在分布式仿真系统中, 仿真进程崩溃及网络连接故障对交互双方最终表现都为网络连接故障, 所以, 如果可以对网络故障进行检测, 就可以间接实现对错误的监控与定位。在本文中, 每个网络连接对应一个“心跳”, 当连接一方连续收不到规定个数的对方“心跳”信息时, 则认为对方已经崩溃, 此时容错系统会利用保存的进程间交互信息来重新恢复该网络连接。

2.2 数据保存模块

就分布式仿真而言, 因为仿真进程间存在交互, 所以为了恢复崩溃的仿真进程, 需要保存两种数据: 仿真进程空间数据以及进程间的交互关系。这是整个容错系统实现的重点和难点。

2.2.1 仿真进程空间保存

采用检查点机制来保存仿真进程。检查点机制可以在内核级、应用级、用户级实现。其中内核级的实现可以获取进程的全部信息, 因此可以实现最大程度上的透明, 但是最终获得的检查点文件比较大,

在传递时会增大系统的开销,此外,因为内核级的实现完全依赖于相应的内核版本,因此维护代价比较大;应用级、用户级的实现只能获得部分进程的信息,所以透明度不会很高而且功能受限,但是最终的进程数据文件较小,系统开销小,由于应用级、用户级的实现独立于内核,因此,维护代价比较小。综合考虑各种实现方式的利弊,本文中检查点机制选择在用户级实现。其实现思路为将实现检查点机制的相应代码编译为静态库,用户程序做少许修改(实际上,用户程序只需要修改一行即可),再跟该静态库重新编译、连接,就可以使该用户程序具有检查点功能。

具体的实现方式为:(1) 保存进程数据段起始地址,堆的结束地址,栈的结束地址;(2) 利用setjmp()保存与进程相关的寄存器信息;(3) 保存进程的栈空间;(4) 保存进程利用 mmap()所分配的大块内存空间;(5) 保存进程的堆空间及数据段。

上述信息被保存到一个文件中,这个文件就是所谓的检查点文件。用户可以通过命令行来指定程序执行检查点功能的周期。

为提高数据保存效率,该检查点机制还实现了 forked 检查点(forked checkpointing[7])、增量式检查点(incremental checkpointing[7])技术。上述两项技术的实现显著地降低了由数据保存带来的系统开销。

2.2.2 *仿真进程间交互关系保存*

如前所述,该容错系统基于进程迁移机制构建。在进程迁移后,其宿主的 IP 地址、端口号必然会发生改变,为了透明地恢复进程间的交互关系,本文采用的方法是在用户空间通过采用"LD _ PRELOAD[9]"这一动态库特性来拦截用户的每一个网络操作,然后用自己开发的动态库来替换这些网络系统调用,并在这些调用中保存进程间的这种交互关系。

交互关系的保存主要包括如下两个方面:交互双方的 IP 地址、端口号;内核中 Send Buffer、Receive Buffer。其中 IP 地址、端口号的保存相对简单;对 Send Buffer、Receive Buffer 通过拦截诸如 write, read 等系统调用实现:在用户空间开辟一块 buffer,数据在被真正写到 Send Buffer 前先保存在该 buffer 中,然后通过该 buffer 模拟 Send Buffer、Receive Buffer 的行为,这样就可间接实现对内核中 Send Buffer、Receive Buffer 的保存。当然,由于在用户空间无法获得内核中网络连接的所有状态,所以其应用有一定的限制。

为防止在错误恢复时发生多米诺效应[6],在进行进程状态保存时采用如下数据保存逻辑:协同检查点(coordinated checkpoint)[6]加基于发送者的悲观消息日志(sender-based pessimistic logging)[6]。

因本地节点可能会崩溃,所以本地硬盘并不可靠,为了能够随时恢复崩溃的进程,必须将获得的数据保存在系统中的稳定存储介质上。具体的解决方法是在仿真系统中添加一个"仿真进程数据存储服务器",用来保存进程检查点文件以及进程间交互的数据。数据的传输通过 GASS Server[5]实现。

2.3 错误恢复模块

该部分的作用是利用保存的进程检查点文件及进程间的交互关系恢复该进程及相应的交互,选择错误恢复节点所需信息由资源状态监控模块实现。其恢复过程具体就是:利用 Linux 的系统调用函数 mmap()和 read()将检查点文件重新映射到系统的虚拟内存空间,而后调用 longjmp()恢复进程的寄存器信息,从而使进程从最近一个检查点开始重新运行;利用保存的进程交互关系来恢复仿真进程间的网络通讯,并重放(replay)进程间传递的消息。该机制也是在用户级实现,原因同上。

由图 1 可见,为实现容错而增加的三个模块自成体系,构成一个相对独立的系统。实际上分布式仿真中的容错主要包括两个方面:RTI 容错、联邦成员容错。因为上述容错系统的容错等级为进程级,而 RTI 或联邦成员最终的表现形式也是进程,所以该容错系统既可适用于 RTI 层的容错,又适合于仿真联邦的容错。即这种容错系统可以跟具体的容错对象相脱离,从而增加了这种容错模式的可移植性。

根据上述分析,增加的模块和现有仿真系统的关系可用图 3 来表示。

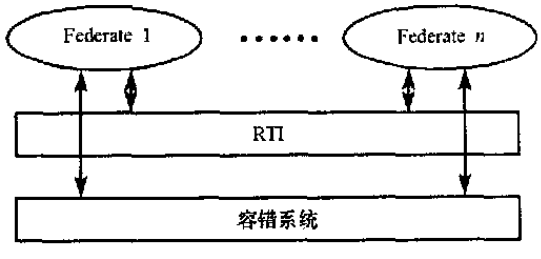

图 3 容错系统、RTI 关系图

Fig. 3 The relationship between fault tolerance system and RTI

2.4 分布式仿真容错代理

为实现透明的容错, 本文在上述容错模块及网格的基础上开发实现了一个容错代理。它主要由运行在仿真数据保存服务器端的 serverBroker() 和运行在每个仿真节点上的 clientBroker() 构成。

serverBroker() 的作用: ①向每个 clientBroker() 发送检查点请求并收集每个节点的检查点文件; ②通过向 MDS 查询获得节点资源信息, 当错误发生时采取措施进行错误恢复。其运行过程用伪码表示如下:

```
while(1){getProcessCheckpointedFile();    //向每个仿真节点发送数据保存指令, 获得进程的检查点文
                                          //件及进程间传递的数据文件并将其保存。
    watchProcessState();                  //利用 PULL 模式, 向 MDS 查询节点及进程状态
    if(nodeFailed||processFailed);        //查询结果表明节点或仿真进程崩溃
    {searchSuitableNode();                //向 MDS 查询, 获得合适的错误恢复节点
        pingGottenNode();                 //确定该节点可用
        startFileTransfer();              //将保存的进程数据文件传送到该节点
        restartProcess()};                //在该节点上恢复该崩溃进程
      sleep(interval_time)};              //每隔 interval_time 秒查询一次, 可以根据实际需要更改
```

clientBroker() 是运行在每个仿真节点上的监听进程, 当它收到 serverBroker 发来的进程数据保存指令时, 就通知数据保存模块进行进程数据保存, 当保存完成后, 调用 GASS Server 将该数据文件传递到仿真数据保存服务器中去; 如果接收到错误恢复指令, 则调用错误恢复模块, 利用从仿真数据保存服务器传来的进程检查点文件以及进程间交互的数据文件来进行错误恢复。

显然通过资源监控模块、数据保存模块、错误恢复模块以及容错代理的组合使用, 可实现容错的透明。

3 结 论

通过前期的测试得知, 上述容错系统可对比较简单的仿真系统进行容错, 但是对关系复杂的仿真进程则难以实现。主要原因是上述容错系统在用户空间实现, 难以获得进程及进程间的全部信息; 此外, MDS 的查询效率太低。本文后续工作将围绕寻找更完善的方式获取进程信息及提高 MDS 的查询效率展开。

参 考 文 献:

[1] Dahmann J S. The High Level Architecture and Beyond: Technology Challenges[A]. In Proceedings of 13th Workshop on Parallel and Distributed Simulation[C], 1999.

[2] Kiesling T. Fault-tolerant Distributed Simulation: A Position Paper[R]. 2003.

[3] 金士尧, 马民. HLA 分布式仿真中容错机制研究[A]. 第十届全国容错计算学术会议, 2003.

[4] L thi J, Berchtold C. Concepts for Dependable Distributed Discrete Event Simulation[A]. In Proceedings of the International European Simulation Multi-conference[C], 2000.

[5] 刘鹏, 等. 网格计算[M]. 北京: 清华大学出版社, 2003.

[6] Elnozahy M, Alvisi L, Wang Y M, et al. A Survey of Rollback-recovery Protocols in Message-passing Systems[R]. Technical Report CMU-CS-99-148, School of Computer Science, Carnegie Mellon University, June 1999.

[7] Milojicic D, Douglis F, Zhou S, et al. Process Migration[R]. HP Labs, AT&T Labs-Research, TOG Research Institute, EMC, and University of Toronto and Platform Computing, Feb, 1999.

[8] Stelling P, Foster I, Kesselman C. A Fault Detection Service for Wide Area Distributed Computations[J]. 0-8186-8579-4/98, IEEE, 1998.

[9] Goerzen J. Linux 编程宝典[M]. 北京: 电子工业出版社, 2000.