基于网络感知的容错志愿计算

樊 沛 沈 锐
(国防科学技术大学计算机学院分布与并行处理国防科技重点实验室 长沙 410073)

摘 要 针对志愿计算系统中节点分布在不同地理位置的特性,分析了传统主-从计算模型在志愿计算系统中的缺陷,提出了基于网络感知的容错志愿计算模型,该模型考虑了节点的网络因素,将节点划分到不同的子集中,基于该模型能够处理由于网络因素造成的故障。另一方面对传统的覆盖容错策略进行了改进并将其应用到容错志愿计算中。实验结果表明,基于网络感知的模型和改进覆盖容错策略能够显著地提高志愿计算系统的可靠性和性能。

关键词 志愿计算,网络感知,容错,覆盖策略
中图法分类号 TP302 **文献标识码** A

Network-aware Based Fault Tolerant Volunteer Computing

FAN Pei SHEN Rui
(Key Laboratory of Science and Technology for National Defense of Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073, China)

Abstract The traditional Master-slave model of volunteer computing is limitation due to the compositional nature of volunteer computing. Considering the volunteer nodes are usually from different locations, a Network-aware based model for fault tolerant volunteer computing was proposed. This model considers the network of the volunteer nodes and partitions the nodes into different subsets, based on this model volunteer computing system can tolerant some fault caused by network delay or timeout. In addition, the overlapping strategy was improved and it was applied to the fault tolerant volunteer computing. The results of experiments show that the network-aware model and improved overlapping strategy can improve the reliability and performance of the volunteer computing.

Keywords Volunteer computing, Network-aware, Fault-tolerance, Overlapping strategy

1 引言

随着计算技术与网络技术的飞速发展和广泛使用,互联网已经演变成为现代社会的一个关键基础设施[1]。互联网上汇聚了大量分布在全球各地的海量资源,这些资源主要有:存储资源、CPU(计算)资源和带宽资源等。尽管人们在网络资源有效共享和利用方面的研究取得了很大的进展,但目前来说互联网上的资源仍然没有得到很好的利用。为了解决这个问题,一系列的技术被提了出来,志愿计算就是其中的一种[2]。志愿计算类似于传统的并行计算,它充分利用互联网用户的空闲资源构建了一个高性能的、提供无限计算能力的并行计算网络,以此满足科学计算的需要[3]。但是同传统的并行计算不同,参与志愿计算的节点分布在不同的地理位置上,并且这些节点一般都是异构的,具有不同的网络环境。正是这些特性志愿计算系统需要处理各种各样的故障,而这些故障一般来自于参与计算的志愿节点(节点加入/退出,节点遭受恶意攻击等)。为了提高系统的可靠性,容错技术被引入到志愿计算中来。

目前有一系列的容错技术被应用到志愿计算中,如冗余计算。但是这些技术都是基于软件容错的方法,它们并没有考虑到参与计算的节点所处的网络环境和需求。节点的网络环境主要包括延迟、负载等等,这些因素都会影响到节点之间的通信,因此对志愿计算系统的可靠性有很大的影响。例如:在亚洲有一个从节点,而主节点在美洲,由于这两个节点所处的网络环境不同,从节点有可能处于一个较差的网络连接的状态,这样从节点和主节点之间的连接很可能是不可靠的。当从节点向主节点发送计算结果时很有可能会出现延迟、掉包等造成的故障。为了解决这个问题,我们需要更多的基础理论、适宜的模型和有效的设计模式用于构建高度可靠的志愿计算系统。

2 相关工作

容错技术按照实现方法的不同可以分成两种类型:空间和时间冗余,主要包括设置硬件和软件副本、执行重复操作等[4,5]。基于这些冗余方法,一系列的容错技术被应用到志愿计算中。

Bayanihan[6]是用 Java 开发的一个志愿计算平台。为了提高结算结果的正确性,Bayanihan 采用了最大投票技术来消除故障带来的影响。最大投票技术的具体实现过程是:对返回的 n 个结果进行投票,当有至少 m 个相同结果时,则认为

到稿日期:2010-07-27 返修日期:2010-11-11 本文受国家重点基础研究发展计划(973)(2005CB321800,2005CB321802)资助。
樊 沛(1984—),男,博士生,主要研究方向为容错计算、分布式系统,E-mail:peifan@nudt.edu.cn;沈 锐(1979—),男,博士生,主要研究方向为程序语言设计、分布式系统。

该次计算获得了正确结果，并将获得最多投票的结果作为正确结果存储到数据库中。最大投票技术在运行时需要同时调用多个副本并对最后的结果进行投票，所以该方法的开销较大。

XtermWeb[7]是一个实验性的志愿计算平台。在 XtermWeb 中采用了两种容错策略：(1)当故障发生时重新执行相应的任务；(2)在服务器出错的情况下仍然确保上传结果的正确性。

BOINC[8]是应用最为广泛的志愿计算平台，目前有很多著名的项目都是利用 BOINC 创建的，如 SETI@home 和 Predictor@home 等。为了处理故障，BOINC 提供了冗余计算机制来确定和排除不正确的结果。该机制类似于 Bayanihan 中应用的技术，它也是采用最大投票技术来确定最终的正确结果。

DISC(Data-Intensive Super Computing)[9]是目前的研究热点，与志愿计算不同，DISC 主要关注大规模数据的处理，一般部署在大规模的集群上，采用并行的方式对大规模数据集进行处理。虽然 DISC 和志愿计算有所不同，但是 DISC 中的容错技术也可以应用到志愿计算中。目前常用的 DISC 系统有 google 的 MapReudce 和 Yahoo 的 Hadoop。为了提高系统的可靠性，这两种系统都采用了重新执行的方式来进行容错计算。

覆盖策略(overlapping strategy)[10]在实时系统中得到了一定的应用，但在分布式系统中覆盖策略的应用却有一定的局限性，这是因为在实时系统中每个任务的起始时间和执行时间是已知的。而在分布式系统中，由于节点是异构的，因此每个任务的执行时间是不一样的，这就给志愿计算系统采用覆盖容错策略带来了一定的难度。

从以上的相关工作可以看出：

1. 虽然目前有众多的容错技术用于志愿计算，但是这些技术都是基于软件容错的思想而提出来的，如重新执行和最大投票技术等。作为一个新的计算平台，志愿计算由分布在不同地理位置、具有不同网络属性的节点组成，这些因素为提高系统的可靠性带来了难度。为了解决这个问题，在志愿计算领域我们需要新的方法和模型来提高整个系统的可靠性。

2. 志愿计算系统的分散性和异构性给覆盖策略的应用带来了很大的难度。如何对覆盖策略进行改进，使其能够用于志愿计算以提高系统的可靠性，还需要做更多的研究。

为了提高志愿计算的可靠性，本文提出了基于网络感知的分组模型，根据从节点和主节点之间的网络延迟对节点集合分组，使得从节点和主节点的延迟最小，从而达到提高系统可靠性的目的。另一方面对覆盖策略进行了改进，并应用到了志愿计算中。

3 模型

目前志愿计算平台大多数使用主-从结构，并且从节点之间相互独立地运行。例如：在 BOINC 平台中的应用都是主-从模式的应用，每一个工作节点(从节点)从主服务器上下载相应的任务并完成它，最后将计算结果传回到服务器中。从 BOINC 应用的执行过程可以看出，志愿计算类似于传统的集群计算，只是参与计算的节点分布在不同的地理位置上。由于传统的主-从模型没有考虑节点的网络环境，因此更有可能会面临由于网络因素造成的故障，比如：在一个志愿计算系统中规定如果计算节点 10s 之内没有返回计算结果，则认为该次计算是无效的，即系统认为发生了故障，但是很有可能会出现由于计算节点所处的网络环境较差，其与服务器之间的延迟要大于 10s，这样就会造成该节点的每次计算都是无效的现象。为了解决这个问题，本节提出了基于网络感知的志愿计算模型。由于节点的网络因素很多，如延迟、负载等，在本文中重点关注是节点的延迟性能。

假设一个志愿计算系统包含有 n 个节点，可以表示为 $c=\{s_i\}_{i=1}^{n}$。该集合的具体介绍如下：

1. c 表示所有节点的集合，在该集合中根据不同的任务属性，每个节点可以定义成元组的形式 $s=\langle D,A_1,A_2,\cdots,A_m\rangle$，在这个元组中 D 和 A_i 是节点的属性，D 代表的是节点的网络属性(本文重点关注节点的延迟属性，即从节点和主节点的延迟属性)。属性 A_i 根据不同的计算任务代表着不同的含义。

2. 根据不同的属性，D 可以将节点划分到不同的子集中：$\{u_i\}_{i=1}^{k}$，这里 $u_i\cap u_j=\phi$，$1<i,j<k$，于是集合 c 可以表示成为 $c=\bigcup_{i=1}^{k}u_i$。

根据上面的介绍可以得到基于网络感知的节点分组过程：每一个节点(计算节点)加入后，都会从主服务器上下载一个列表文件，该列表文件记录了所有的主节点的信息，计算节点会和每个主节点进行连接测试，以确定它们之间的延迟性能，并选择其中最小的延迟将计算节点和主节点划分到同一个子集中。对每一个参与计算的节点都执行该操作，最后可以将所有的节点划分成多个子集，每个子集都有一个主节点和多个计算节点。每个计算节点从主节点下载计算任务，并将结果传回主节点，最后由主结点统一传回主服务器。

与传统的主-从模型相比，本文提出的基于网络感知的分组模型具有以下几个优点：

1. 可扩展性：传统的主-从模型是两层结构，而基于网络感知的模型是三层结构，基于该模型所有的节点被划分到不同的子集中，这样更利于节点的管理，中央服务器只需要和主节点通信，每个主节点负责管理其所在子集中的计算节点，中央服务器不再需要管理所有的节点，这样降低了中央服务器的开销。

2. 可靠性：新的模型较传统的模型具有更大的可靠性，这是因为每个计算节点都被划分到和其延迟最小的主节点同一个集合中。在该模型中，计算节点和主节点的连接状态是最好的，通信延迟是最小的。因此，相比原来的模型，新的模型能够减少由于网络延迟带来的故障。

图 1 是基于网络感知模型的示意图。

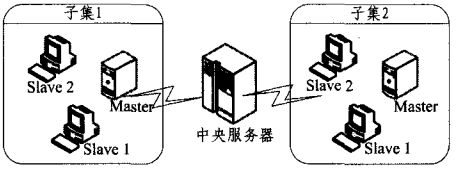

图 1 基于网络感知模型示意图

4 容错策略

4.1 基本容错策略

为了提高志愿计算的可靠性，可以采用空间和时间冗余

方法来进行容错处理。重复执行(Retry)[11]和恢复块技术(Recover Block)[12]是两种主要的时间冗余策略。NVP(N Version Programming)[13]是最主要的空间冗余方法。下面将介绍这几种基本的容错策略。

Retry:如图 2(a)所示,当副本失效时计算节点将会重新执行相应的任务。

RB:如图 2(b)所示,如果主副本失效,则其他的副本将会按照顺序执行的方式运行。

NVP:如图 2(c)所示,采用 NVP 策略时所有的副本将会在同一时间启动,最终的结果采用前面介绍的最大投票技术来确定。

限于篇幅,这几种基本策略详细的细节可以参考文献[14]。

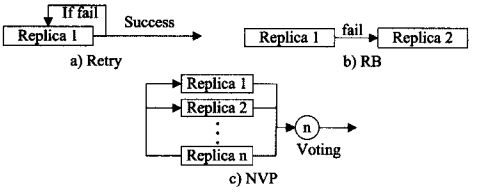

图 2 基本容错策略示意图

从图 2 可以发现,空间冗余策略(NVP)的开销非常高,这是因为所有的副本都在同一时间启动,并且最后的结果需要投票才能够确定。该方法引入了过多的计算时间,对系统的性能影响较大。另一方面,时间冗余策略虽然一次只调用一个副本,其开销要小于空间冗余策略,但是可能会有较高的失效率,这是因为时间冗余策略不能够处理逻辑。为了解决这个问题,本文综合了时间和空间冗余的方法,采用覆盖策略(Overlapping)来进行容错计算。

4.2 覆盖策略(Overlapping Strategy)

覆盖策略如图 3 所示,备份副本在主副本执行一段时间后才开始执行,当主副本执行成功时备份副本将会终止。基于这个执行过程,覆盖策略不仅考虑了时间的开销,也考虑了资源的开销。覆盖策略的可靠性和响应时间如式(1),式(2)所示。我们采用 RTT(Round-Trip-Time)来表示发送一个计算请求到接受到节点返回的计算结果之间的时间。在式(1),式(2)中,n 表示副本的个数,r_i 表示副本的可靠性,t_i 是第 i 次请求的 RTT 时间。o_t 表示在覆盖策略中,间隔多少时间后开始启动下一个副本,当 o_t 等于任务的计算时间时,覆盖策略等于 RB 策略,当 $o_t=0$ 时,覆盖策略类似 NVP 策略。

$$r=1-\prod_{i=1}^{n}(1-r_i) \tag{1}$$

$$t=t_1+\sum_{i=2}^{n}(t_i-o_t)\prod_{k=1}^{i=1}(1-r_k) \tag{2}$$

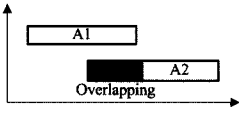

图 3 覆盖容错策略示意图

如前所述,要使用覆盖策略,首先要知道每个任务的执行时间。在志愿计算系统中,由于其每个节点都是异构的,网络环境是不同的,因此每个节点上副本的执行时间是不相同的,是无法确定的,这样就使得覆盖策略在志愿计算中的使用具有一定的局限性。为了解决这个问题,本文针对志愿计算的特点对覆盖策略做了改进,采用轮询时间作为每个任务的执行时间。轮询是指子集中的主节点每隔一定的时间就会对子集中所有计算节点的状态进行检查,两次检查之间的时间间隔称为轮询时间。采用轮询时间代替任务执行时间解决了任务执行时间不一致的问题,使得覆盖策略可以应用到容错志愿计算中。

4.3 容错策略的性能

在对容错策略进行比较时不能只考虑出错数,更要考虑容错策略的开销和性能。对于容错策略的性能,可以采用式(3)来计算。式中,ω_i ($1<i<3$),用户定义的权重,t_{max},f_{max},r_{max}均由用户指定。根据式(3)就可以选择性能最好的策略。

$$pref=\omega_1\frac{t_1}{t_{max}}+\omega_2\frac{f_1}{f_{max}}+\omega_3\frac{r_1}{r_{max}} \tag{3}$$

5 实验

本节进行了一系列的实验来对本文提出的模型和策略进行评估。所有的实验都是在 Planet-Lab[15]上进行的,从该平台上选取了来自美洲、欧洲和亚洲 3 个区域的节点,表 1 列出了每个区域的节点个数。在试验中分别对基于网络感知模型和主-从模型下各种容错策略的性能进行了比较。

表 1 试验使用的节点

	USA	EUR	ASIA
Num	61	33	18

本文的实验由 Owlet[16]和 JDK6.0 实现,Owlet 是一个基于发布/订阅机制的交互式程序设计语言,详细的 Owlet 语言介绍可以参考文献[16]。在试验中,副本都是部署在不同的节点之上,采用故障注入技术来模拟运行时可能出现的故障,表 2 定义了实验中的各种设置参数。

表 2 试验参数设置

	参数名称	设置
1	请求频率	90000 ms
2	轮询频率	30000 ms
3	o_t	5000 ms
4	副本个数	3
5	Timeout	90000 ms
6	网络故障概率	0.11
7	逻辑故障概率	0.11
8	f_{max}	0.1
9	t_{max}	90000 ms
10	r_{max}	5
11	$\omega_1-\omega_3$	1/3

5.1 实验结果

实验结果如表 3 和表 4 所列,表 3 是采用基于网络感知模型的结果,表 4 是采用传统的主-从模型的结果。

在每种模型下分别对 Retry,RB,NVP 和 Overlapping 4 种容错策略进行了实验。表 3,表 4 中策略一栏代表所用的容错策略,All 代表请求的次数,RTT 代表 RTT 时间,Fail 代表出错数,Res 代表所使用的资源数,pref 代表每种策略的性能。

表 3 是基于网络感知模型下各种容错策略的结果,从表 3 中可以看出 NVP 策略的 RTT 性能最差,Overlapping 最好。在 Fail 一栏中显示了各种策略的出错数,Retry 和 RB 的出错数最多,NVP 的出错数最少。这是因为 NVP 不仅可以处理网络错误,还可以处理逻辑错误。在 Res 一栏中 Retry

(下转第 63 页)

Conference on Communication Systems and Networks. Mallorca, Spain, Aug. 2006: 7-12
[5] Füβler H, Widmer J, Käsemann M, et al. Contention-based Forwarding for Mobile Ad-Hoc Networks [J]. Elsevier's Ad-hoc Networks, 2003, 1(4): 351-369
[6] Lochert C, Mauve M, Füβler H, et al. Geographic Routing in City Scenarios [J]. ACM SIGMOBILE Mobile Computing and Communications Review (MC2R), 2005, 9(1): 69-72
[7] Kosch T. Local Danger Warning based on Vehicle Ad-hoc Networks: Prototype and Simulation [C]// 1st International Workshop in Intelligent Transportation (WIT 2004). Hamburg, Germany, March, 2004: 43-47
[8] Wischhof L. Adaptive Broadcast for Travel and Traffic Information Distribution Based on Inter-Vehicle Communication [C]// Proceedings of IEEE Intelligent Vehicles Symposium (IV03). Columbus, Ohio, USC. June 2003: 6-11
[9] Nadeem T, Shankar P, Iftode L. A Comparative Study of Data Dissemination Models for VANETs[C]// 3rd Annual International Conference on Mobile and Ubiquitous Systems Workshops. San Jose, CA, 2006: 1-10
[10] OMNet++ Community Website [EB/OL]. http://www.omnetpp.org

(上接第 40 页)

和 RB 都等于 1，因为这两种策略每次只运行一个副本，而 NVP 启动的副本数最多。在主要的 pref 性能一栏中可以看出，Overlapping 策略的性能最好，Retry 性能最差(数值越小性能越好)。

表 3 基于网络感知模型实验结果

策略	All	RTT	Fail	Res	Perf
Retry	2800	17843	324	1	0.518
RB	2800	17171	316	1	0.506
NVP	2800	25289	174	3	0.501
Overlapping	2800	14379	244	1.68	0.455

表 4 是在传统主-从模型下各种容错策略的实验结果，其结果类似于表 3，但是从表 3 和表 4 中可以得出以下结论：

1. 在基于网络感知模型中所有容错策略的各项结果均要好于传统的主-从模型中的结果，如在表 3 中所有策略的 RTT 时间要小于表 4 中的 RTT 时间，表 3 中的各种策略的出错数要小于表 4 中的，由此可以看出基于网络感知模型比传统的主-从模型能够处理更多的由于网络延迟带来的故障。

2. 在所有的策略中，覆盖策略的性能是最佳的，在表 3 中，覆盖策略的性能是 0.455，表 4 中性能是 0.499。从图 4 可以看出，这两个值均要小于其他的策略。

表 4 传统主-从模型实验结果

策略	All	RTT	Fail	Res	Perf
Retry	2800	18117	395	1	0.604
RB	2800	18679	360	1	0.564
NVP	2800	25740	221	3	0.588
Overlapping	2800	15798	280	1.62	0.499

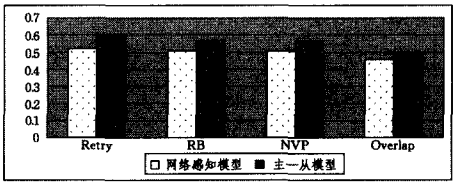

图 4 4 种策略的性能对比图

结束语 志愿计算是提高网络资源利用率的一个有效应用方式。由于志愿计算自身的特性，如何做到容错计算是目前的研究热点。本文在分析了传统了主-从模型和基本容错策略的基础上，提出了基于网络感知模型和改进的覆盖容错策略，相对比传统的模型，基于网络感知模型能够处理由于网络因素带来的故障，使得志愿计算系统的可靠性得到了一定的提高。

在未来工作中我们将考虑更多的网络因素，如负载、带宽等等，并分析这些因素对志愿计算系统可靠性的影响，进一步完善本文提出的模型和策略。

参 考 文 献

[1] 卢锡城，王怀民，王戟. 虚拟计算环境 iVCE：概念与体系结构 [J]. 中国科学 E 辑信息科学，2006，36(10)：1081-1099
[2] Anderson D P, McLeod J. Local scheduling for volunteer computing[A]// 21th International Parallel and Distributed Processing Symposium, 2007[C]. Long Beach: IEEE, 2007: 1-8
[3] 王宇，王志坚. 志愿计算模型形式化方法[J]. 软件学报，2008，19(5)：1125-1133
[4] Chan P P-W, Lyu M R, Malek M. Making Services Fault Tolerant[A]// Third International Service Availability Symposium, ISAS 2006[C]. Helsinki: Springer, 2006: 41-63
[5] 杜云飞，唐玉华，杨学军. 容错并行算法的性能分析[J]. 计算机科学，2009，36(9)：248-251
[6] Sarmenta L F G, Hirano S. Bayanihan: building and study web-based volunteer computing systems using java [J]. Future Generation Comp. Syst., 1999, 15(5/6): 675-686
[7] Malecot P, Kondo D, Fedak G. XtremLab: A system for Characterizing Internet Desktop Grids[A]// 15th IEEE International Symposium on High Performance Distributed Computing. 2006 [C]. Paris: IEEE, 2006: 357-358
[8] Aderson D P. BOINC: A System for Public-Resource Computing and Storage[A]// 5th International Workshop on Grid Computing, 2004[C]. Pittsburgh: IEEE, 2004: 4-10
[9] Lammel R. Google's MapReduce programming model-Revisited [J]. Sci. Computer. Program, 2008, 70(1): 1-30
[10] 王建，孙建伶，王新宇，等. 容错多处理机中一种高效的实时调度算法[J]. 软件学报，2009，20(10)：2628-2636
[11] Chan P P-W, Lyu M R, Malek M. ReliableWeb Services: Methodology, Experiment and Modeling[A]// 2007 IEEE International Conference on Web Services 2007[C]. USA, IEEE, 2007: 679-686
[12] Randel J X B. The evolution of the recovery block concept[M]// Lyu M R, ed. Software fault tolerance. Wiley, Chichester, 1995: 1-21
[13] Avizienis A. The methodology of n-version programming[M]// Lyu M R, ed. Software fault tolerance. Wiley Chichester, 1995: 23-46
[14] Leu E L D, Bastani F B. The effect of statically and dynamically replicated components on system reliability [J]. IEEE Trans Reliab, 1999, 39(2): 209-16
[15] Planet-Lab[EB/OL]. http://www.planet-lab.org
[16] Owlet[EB/OL]. http://owlet-code.sourceforge.net