一种适用于P2P存储系统的自反馈故障检测算法

万亚平[1,2] 冯 丹[1] 欧阳利军[2] 刘 立[1] 杨天明[1]

(华中科技大学计算机科学与技术学院 武汉 430074)[1] (南华大学计算机科学与技术学院 衡阳 421001)[2]

摘 要 在构建高可用性P2P存储系统的过程中,针对系统中节点的高度动态特征,设计了一种自反馈的心跳故障检测算法。它结合心跳策略和无偏灰色预测模型,根据应用需求和网络环境的变化动态地改变检测的质量,在保持一定检测时间的前提下,提高了故障检测的精度。实验表明,根据该算法实现的故障检测器具有较好的性能,提高了P2P存储系统的可用性。

关键词 可用性,自反馈,心跳,对等存储系统,故障检测

Self-feedback Fault Detection Algorithm for Peer-to-Peer Storage System

WAN Ya-ping[1,2] FENG Dan[1] OUYANG Li-jun[2] LIU Li[1] YANG Tian-ming[1]

(Computer Department, Huazhong University of Science and Technology, Wuhan 430074, China)[1]

(School of Computer Science and Technology, Nanhua University, Hengyang 421001, China)[2]

Abstract Peer-to-Peer(P2P) storage systems have a lot of attractive advantages, such as self-organizing, scalability and fault tolerance. Fault detection must be one of basic components to build a reliable P2P storage systems. To address the highly dynamic characteristics of system nodes, a self-tuning heartbeat fault detection algorithm, which can combine heartbeat strategy with unbiased grey prediction model, was designed to improve the fault detection quality of system (QoS) according to the application needs and network environment changes. The results show that, on the basis of the algorithm implementation, fault detectors have better performance.

Keywords Availability, Self-feedback, Heartbeat, P2P storage systems, Fault detection

在以数据为中心的信息时代,存储已成为IT基础设施的核心之一。数据存储已经成为继互联网热潮之后的第三次技术浪潮,它将网络带入了以数据为中心的时代。在P2P(对等结构)网络上构建存储系统有诸多的优势,它能更有效地利用节点的网络带宽、存储空间以及计算资源。P2P存储易于维护、可扩展性好、自配置功能强,特别是P2P存储可以和大量加盟的PC机和服务器中的存储器组合成存储系统,以提供高带宽的视频服务和其他共享服务。目前主要的研究机构包括Berkeley,MIT,MSR以及国内北京大学等都提出了自己的一些相应的P2P存储原型系统,如OceanStore,CFS[1],Bit-Vault[2,3],PAST[4]等,但是P2P存储系统的节点具有高度动态性和流动性,使得构造高可用的P2P存储系统成为研究的关键问题,且高效的故障检测方法是检测节点的存活性和快速功能切换的前提和保证。

故障检测在分布式计算系统中的应用很多[5]。对故障检测器的特性主要考虑准确性(accuracy)、完整性(completeness)、网络负载、自适应性(adaptive)以及灵活性(Flexibility)等。起初,研究者在进行故障检测研究时,针对的异步系统模型仅仅包括一对进程或者一组进程,系统规模较小,因此当时进行的研究也仅仅考虑到完整性和有效性。后来人们逐渐注意到分布式环境中网络状况的变化对故障检测的影响,于是提出了自适应网络状况变化的故障检测,并提出为分布式系统建立通用的故障检测服务来平衡上述几个特性。随着使用故障检测服务的分布式环境规模的不断扩大,这些环境的动态性使人们认识到故障检测应该具有良好的可扩展性。

本文针对P2P分布式存储系统的特点,基于不可靠故障检测(Unreliable Failure Detector)[6]的要求,结合无偏灰色预测模型[7]的预测值和修正值,实现了一种自反馈的心跳故障检测方法。在此基础上,实现了动态故障检测算法。实验和计算表明,该方法有效提高了故障检测的质量,在保证一定检测时间的前提下,提高了检测的精度,适用于P2P这种节点高动态变化的系统。

1 相关工作

在同步系统中,进程的执行时间和消息传递时间都有限制,系统可以实现"完美设计"的可靠故障检测器,但在异步系统中,如果检测器仅仅依靠消息传递是不可能真正达到完美的。Fischer等人证明:在一个异步系统中,即使是只有一个

到稿日期:2009-03-05 返修日期:2009-06-01 本文受国家973基础研究计划(2004CB318201)专项基金,教育部创新团队(No. IRT-0725),湖南省教育厅课题(06C728)资助。

万亚平(1973-),男,博士生,主要研究方向为网络存储、故障检测技术等,E-mail:wypzll@163.com;冯 丹(1970-),女,教授,博士生导师,主要研究方向为信息存储系统、计算机网络等;欧阳利军(1969-),男,副教授,主要研究方向为分布式计算等;刘 立(1971-),男,博士,主要研究方向为图像处理等;杨天明(1968-),男,博士生,主要研究方向为网络备份、重复数据删除等。

进程出现崩溃故障，也没有算法能够解决一致性问题(Consensus Problem)。因为在一个异步系统中，进程可以随机发出响应的信息，所以没有办法分辨一个进程是速度很慢还是已经崩溃。他们的证明显示了进程的执行总是由于延迟的原因而很难达成一致性认识。这个结论被称为 FLP(Fischer-Lynch-Paterson)不可能性结论[8]。

Chandra 和 Toueg 第一次提出有关在异步系统中使用不可靠的故障检测器(Unreliable Failure Detector，UFD)的问题[6]。一个不可靠故障检测器可以产生以下两个值之一：Trusted 和 Suspected。这两种结果都只是提示，这种提示可能精确地反映了进程的故障情况，也有可能是不精确的。提示值 Trusted 表示检测器信任，认为被监控进程无故障，例如，最近从该进程收到一个消息。提示值 Suspected 表示检测器怀疑，有迹象表明进程可能已经出故障了，例如，在故障检测时限内没有收到来自进程的消息。不可靠故障检测器在达到共识时，允许被怀疑的进程正确行动，而不是排除出去。Chandra 和 Toueg 为了解决异步系统中的共识，分析了一个故障检测器必须拥有的属性。他们证明了，即使是使用不可靠故障检测器，只要通信是可靠的且崩溃进程不超过进程总数的 1/2，异步系统的共识问题是可以解决的。

随着故障检测应用范围不断扩大，人们逐渐注意到分布式环境中网络状况的变化对故障检测的影响。在异步系统中，网络状况是不断变化的。网络负载过重或进程处理速度变化都会影响检测消息的发送和到达时间。因此，研究者提出了自适应网络状况变化的故障检测方法，并进一步扩展了故障检测器的评估标准，提出了故障检测器服务质量(Quality of Service，QoS)的需求。Chen 等人在文献[9]中研究了故障检测器的服务质量问题，第一次系统地给出了 6 个评价故障检测服务质量的指标，为故障检测器的应用提供了更详细的理论指导。

自适应故障检测的研究内容主要是利用收集到的心跳消息动态调整故障检测时间，根据历史窗口大小的心跳到达时间数据来预测下一个心跳应该到达的时间[10]。此外，自适应性故障检测还能够适应不同的应用，根据故障检测服务质量的要求来配置自适应故障检测器。目前，异步系统中的适应性故障检测算法主要有两种：一种是 Chen 在文献[9]提出的自适应故障检测算法，另一种是由 Bertier 在文献[11]中提出的自适应故障检测算法。Bertier 的算法是在 Chen 的算法基础上进行改进的算法。在 Chen 的预测算法中，由于修正值取常数的方法导致故障检测时间长、速度慢，因此，在 Bertier 的算法里，提出结合 Jacobson 算法[12]来估计修正值。

上面所述方法或者是基于静态心跳机制，或是采用动态心跳机制，但是都需要大量的样本信息或者需要样本空间满足某种概率分布要求。比如 Hayashibara 在文献[14] 中提出了一种 ϕ 故障检测器，并且假定心跳到达时间符合正态分布。然而实际上在 P2P 存储系统中，并不能确定心跳到达时间是否满足假定的概率分布条件。

文献[15] 提出了一种所谓“懒散”的故障检测机制，其结合心跳消息和应用消息做心跳的发送，当没有应用消息发送时，才启用心跳消息进行监控。此法一定程度上降低了系统因为发送心跳消息而导致的负载，但故障检测的准确度和系统的可用性却有所降低，并且由于两种消息的切换导致需要更多的检测时间。

文献[13] 将心跳策略和“小样本”灰色预测模型结合起来，实现了一种网格环境下的动态故障检测算法。但其具有灰色预测模型的固有偏差，应用范围不广，对存在随机扰动的系统精度不高，并且没有对修正值进行估计，不能满足 P2P 存储系统的要求。

本文基于心跳机制，结合无偏灰色预测模型设计了一种自整定的动态故障检测方法，其动态调整修正值，减少了消息窗口大小对检测精度的影响。

2 故障检测器模型

Chandra 和 Toueg 最早给出了一系列故障检测器的形式化定义[6]，指出无法完全解决故障检测的准确性，只能在一定程度上改善。假设一个有 n 个进程组成的系统，记作 $F=\{P_1,P_2,P_3,\cdots,P_n\}$。假设系统存在一个独立的全局时钟，根据时钟报时信号得到的值域 T 作为自然记数。将故障检测器定义为一个故障检测模块的集合，每一个模块对应于一个进程，其输出为一系列此进程所怀疑发生故障的进程集合。故障检测有两种最基本的特性：完整性(Completeness)和准确性(Accuracy)。完整性设定了对故障进程的检测条件，准确性限定了故障检测器所能犯的错误。Chandra 和 Toueg 对这两个特性又进行了更细致的划分，分为 8 类[6]。

与 P2P 存储系统对应，定义故障检测系统是有限 n 个故障检测进程的集合，记作 $GF=\{P_1,P_2,P_3,\cdots,P_n\}$。任意两进程之间都有固定的链路连接，通道可靠，消息在传输的过程中不会发生丢失、更改和复制。

由以上给出一个◇P 类的故障检测器，定义如下。

1)强完整性(strong completeness)：在任意一次运行中，每一个发生故障的进程最终都会被所有正确的进程永远判定为故障。

2)最终强准确性(eventual strong accuracy)：在某个时间点后，正确的进程都不会被怀疑故障。

在实际系统中，应用程序主要从以下两方面评价一个故障检测器所提供的 QoS。

1) 故障检测器对真实发生的故障的检测速度，即一个故障检测器需要多少时间发现故障进程；

2) 故障检测器在一定的检测速度下发生错误输出的概率，即对一个正确进程产生怀疑的概率。

为了准确评价故障检测器的 QoS，参考 Chen 等人提出的定量的 QoS 度量体系[9]，其提出了以下 3 个评价指标。

1)故障检测时间(TD)：指从被监控进程发生故障开始到监控进程开始怀疑它发生故障之间的一段时间；

2)故障检测精度(PA)：指故障检测方法正确判断进程故障的几率；

3)故障检测器 CPU 占用率(Δ)：指运行故障检测器占用的 CPU 负载。

3 无偏灰色预测模型及实时心跳设计

3.1 基本设计思想

进程 p 周期性地向监控进程 q 发送心跳消息，发送之前记录每次发送心跳信息的序号且序号递增增加。定义时间窗大小为 n，根据建立的无偏灰色预测模型预测第 $n+1$ 次心跳

消息到达的时间。如果在定义的时限内没有收到 p 发送的心跳消息，则 q 怀疑 p 发生故障。

灰色预测是灰色系统理论的重要组成部分。在灰色预测中，应用最广泛的是提出的 GM(1,1)模型[16]，在此将其称为传统 GM(1,1)模型。由于传统 GM(1,1)模型本身的缺陷，使其仅能适用于原始数据序列按指数规律变化且变化速度不是很快的场合[17]。本文使用改善后的无偏灰色预测模型并将其应用于 P2P 存储系统的动态节点的存活状态的检测。

3.2 传统 GM(1,1)模型

设有由故障检测的心跳到达时间组成的原始数据序列：$t^{(0)}(1),t^{(0)}(2),\cdots,t^{(0)}(n)$，满足 $t^{(0)}(k)\geqslant 0,k=1,2,\cdots,n$。利用该数据序列建立 GM(1,1)模型的步骤如下。

(1)对原始序列做一阶累加形成生成数据序列

$$t^{(1)}(k)=\sum_{m=1}^{k}t^{(0)}(m)\quad t=1,2,\cdots,n \tag{1}$$

(2)确定数据矩阵 A 和 X_n

$$A=\begin{pmatrix} -\frac{1}{2}[t^{(1)}(1)+t^{(1)}(2)] & 1\\ -\frac{1}{2}[t^{(1)}(2)+t^{(1)}(3)] & 1\\ \vdots & \vdots\\ -\frac{1}{2}[t^{(1)}(n-1)+t^{(1)}(n)] & 1 \end{pmatrix}$$

$$X_n=\begin{pmatrix} t^{(0)}(2)\\ t^{(0)}(3)\\ \vdots\\ t^{(0)}(n) \end{pmatrix}$$

(3)求参数列

$$[\hat{a}\quad \hat{u}]^T=(A^TA)^{-1}A^TX_n \tag{2}$$

(4)插入生成数据序列模型

$$\hat{t}^{(1)}(k+1)=(t^{(0)}(1)-\frac{\hat{u}}{\hat{a}})e^{-\hat{a}k}+\frac{\hat{u}}{\hat{a}}\quad k=1,2,\cdots \tag{3}$$

(5)建立原始数据序列模型

$$\hat{t}^{(0)}(1)=\hat{t}^{(0)}(1) \tag{4a}$$

$$\hat{t}^{(0)}(k)=\hat{t}^{(1)}(k)-\hat{t}^{(1)}(k-1)=(1-e^{\hat{a}})(x^{(0)}(1)-\frac{\hat{u}}{\hat{a}})e^{-\hat{a}(k-1)}$$

$$k=2,3\cdots \tag{4b}$$

其中，$\hat{t}^{(0)}(k),k>n$ 表示原始序列的预测值。

设原始数据序列为严格的指数序列，即

$$x^{(0)}(k)=Be^{a(k-1)}\quad k=1,2,3,\cdots N \tag{5}$$

其一次累加生成序列为

$$x^{(1)}(k)=B\frac{1-e^{ak}}{1-e^{a}}\quad k=1,2,3,\cdots N \tag{6}$$

用传统 GM(1,1) 建模方法建模可得

$$A=\begin{bmatrix} -\frac{1}{2}B\frac{2-e^{a}-e^{2a}}{1-e^{a}} & 1\\ -\frac{1}{2}B\frac{2-e^{2a}-e^{3a}}{1-e^{a}} & 1\\ \vdots & \vdots\\ -\frac{1}{2}B\frac{2-e^{(N-1)a}-e^{Na}}{1-e^{a}} & 1 \end{bmatrix},$$

$$X_n=\begin{bmatrix} Be^{a}\\ Be^{2a}\\ \vdots\\ Be^{(N-1)a} \end{bmatrix}$$

由式(2)求得参数列

$$[\hat{a}\quad \hat{u}]^T=(A^TA)^{-1}A^TX_n=\begin{bmatrix} \frac{2(1-e^{a})}{1+e^{a}}\\ \frac{2A}{1+e^{a}} \end{bmatrix} \tag{7}$$

最终拟和结果为

$$\hat{t}^{(0)}(1)=B \tag{8a}$$

$$\hat{t}^{(0)}(k)=\frac{-Be^{a}(1-e^{\hat{a}})}{1-e^{a}}e^{-\hat{a}(k-1)}\quad k=2,3,\cdots,N \tag{8b}$$

比较式(5)和式(8)发现，传统 GM(1,1)模型在进行指数序列预测时存在较大的偏差，相较于存储系统的故障检测要求存在性能的不稳定特征，需要对其加以修复。

3.3 无偏 GM(1,1)模型

由式(7)可得 $a=\ln\frac{2-\hat{a}}{2+\hat{a}},B=\frac{2\hat{u}}{2+\hat{a}}$，即可用传统 GM(1,1)模型的参数 $\hat{a},\hat{u}$ 表示原始数据序列的参数 a 和 B。

假设对指数序列所建立的模型为：

$$\hat{t}^{(0)}(k)=\hat{B}e^{\hat{a}'(k-1)}\quad k=1,2,3,\cdots,N \tag{9}$$

令 $\hat{a}'=\ln\frac{2-\hat{a}}{2+\hat{a}},\hat{B}=\frac{2\hat{u}}{2+\hat{a}}$，则有 $\hat{a}'=a,\hat{B}=B$。

由此建立无偏 GM(1,1)模型，其建模步骤为：

步骤(1)－步骤(3)　同传统 GM(1,1)模型；

步骤(4)　求无偏 GM(1,1)模型参数

$$\hat{a}'=\ln\frac{2-\hat{a}}{2+\hat{a}},\hat{B}=\frac{2\hat{u}}{2+\hat{a}};$$

步骤(5)　建立原始数据序列

$$\hat{t}^{(0)}(1)=t^{(0)}(1)$$

$$\hat{t}^{(0)}(k)=\hat{B}e^{\hat{a}'(k-1)}\quad k=2,3,\cdots$$

由上可见，与传统的 GM(1,1)模型相比，无偏 GM(1,1)模型不存在 GM(1,1)模型固有的偏差，消除了原始序列增长率较大时发生故障的现象，并且无须进行累减还原，简化了建模步骤，降低了计算时间和系统的负载。

4 自整定的故障检测机制

基于 P2P 存储系统的高度动态性特征，本文设计了一种自整定的故障检测机制，它能适应实际系统由于环境的变化而存在的不稳定性，并且能够根据性能需求的变化重新配置故障检测的修正值，以满足故障检测的 QoS 评价指标要求。

不失一般性，任选系统中的两个故障检测进程 p 和 q。所述信息监控进程 p 将 q 发送过来的心跳到达时间放入时间窗中，达到设定的数量以后，利用无偏灰色预测模型，预测下一次心跳消息到达时间，并结合心跳到达时间修正值设定判断故障的时限；若在判断故障的时限内没有收到对方发送过来的心跳消息，则认为对方故障。

由无偏灰色预测模型产生的心跳到达时间预测值，结合心跳到达时间修正值 σ，生成 $N+1$ 次心跳消息到达时间预测

值作为判断故障的时限(Timeout)。

$$Timeout=\hat{t}^{(0)}(k)+\sigma,k=N+1$$

心跳到达时间修正值 σ 为

$$\sigma=i\times\frac{\sum_{k=1}^{N}(t^{(0)}(k)-\hat{t}^{(0)}(k))}{N}$$

其中,i 为权值且 $0<i<2$,$t^{(0)}(k)$ 为每次原始心跳到达时间,$\hat{t}^{(0)}(k)$ 为每次预测的心跳到达时间,N 为时间窗口的大小。

另外,为了反映系统的实时性,达到指定的时间窗口大小的数据后,最新的心跳到达时间数据总是作为最后一个数据加入到时间窗口中,并且把第一个数据从时间窗口中排除出去。

4.1 基于自整定的故障检测算法

根据上面所述,给出一个自整定的故障检测算法如下:

算法 1

```
1.  For Process q:
2.  send heartbeat meassage to p every Δi
3.  For Process p:
4.  N //max size of time_window(e. g. 1000)
5.  S=nil // S is time_window of tr
6.  S'=nil //S' is time_window of tp
7.  tr // tr is real heartbeat arrival time every Δi
8.  tp // tp is prediction heartbeat arrival time every Δi
9.  i // i is weight of correction value
10.  upon receive heartbeat message mj at time tr from q
11.  if j<N then add tr into S and tp = tr into S'
12.       else if j=N then
13.  t^(0)(k) = B^ e^a'(k-1) //predict (the K+1)th heartbeat arrvial time using unbiased grey-forecasting model
14.  σ=i×(Σ_{k=1}^{j}(t^(0)(k)-t^(0)(k)))/j // correction value
15.  Timeout=t^(0)(k)+σ
16.  tp =Timeout //tp is (N+1)th prediction heartbeat arrival time
17.       else remove head of S endif
18.  endif
19.  At time Timeout
20.  if not receive mj from q
21.  then Suspected
22.  else Trusted
23.  At time t>timeout
24.  if p receive m from q
25.  then remove q from suspected
```

算法描述如下:

被监视进程 q 周期性地向故障检测进程发送心跳信息。p 收到心跳消息后,按照到达时间的先后放入实际心跳到达时间窗 S 中。在心跳到达时间的数量小于时间窗大小 N 时,每一次预测的心跳达到时间等于实际的心跳到达时间,并且把这个时间放入预测心跳达到时间窗 S' 中。当心跳到达时间的数量等于时间窗大小 N 时,即采用无偏灰色预测模型预测第 $N+1$ 次心跳消息到达时间。此值结合心跳到达时间修正值生成判断故障的超时时限 Timeout。修正值的计算方法为:当心跳到达时间数量小于等于 N 时,修正值为 0;当大于 N 时,修正值等于实际心跳到达时间和预测的心跳到达时间的差值的加权平均值。

4.2 算法证明

在这一节,按照 Chandra 和 Toueg 对故障检测器的分类,将证明在部分同步模型下,本文所提出的故障检测算法为应用程序所提供的故障检测服务可以等价于一个◇P 类故障检测器。Chandra 和 Toueg 所描述的部分同步模型是一种常用的、与实际系统更为相似的分布式系统模型。在此模型中,假设在到达某个未知的时间,即全局稳定时间(global stabilization time,简称 GST)之后,进程的处理速度和消息的传输时间都是有界的,但界限是未知的。

定理 1 算法 1 所示的故障检测算法满足不可靠故障检测的强完整性要求,即每一个发生故障的进程最终都会被所有正确的进程判定为故障。

证明:假设一个正确的进程 p 使用算法 1 来检测进程 q。如果 q 发生故障,那么存在某一时刻 tl,对于 $\forall t>tl$,一个应用程序查询的结果都将判定 q 故障。

$\forall$ p,q;假设 q 在第 n 次发送心跳消息的时间为 $tlast$,且在此之后的 tc 时刻发生故障,则在 $tlast+\Delta i$ 时刻 q 不再向 p 发送心跳消息。所以在 p 预测的时间 timeout 内显然不能收到 q 的第 $n+1$ 次消息,由此证毕。

定理 2(最终强准确性,eventual strong accuracy) 在某个时间点后,正确的进程都不会被怀疑故障。

证明:某个时刻 t 之后,应用程序将不会错误地认为 q 发生故障,因为此属性具有最终(eventual)性。

假设 q 在最终稳定时间之后在时刻 ts 发出第一个检测消息。在 GST 之后,假设忽略消息的处理时间,而消息的传递时间有上界,记为 Δttr。因此对于任意的被检测进程 q 发送的第 $n+1$ 次心跳消息,除非检测进程 p 发生故障,否则该消息一定会在 Δttr 后到达 p 处。若 $tp=\hat{t}^{(0)}(k)+\sigma$ 为第 $n+1$ 次心跳消息到达时间的预测值,tr 为第 $n+1$ 次心跳消息实际的时间,如果 $\Delta ttr<tp$,则 $tr\leqslant\Delta ttr<tp$,由算法 1(19－22 行)来看,p 不会怀疑 q 发生故障。

如果 $\Delta ttr>tp$,则 $tr>tp$ 的情况下,p 怀疑 q 发生故障,但由于 $tp\leqslant\Delta ttr$,由算法 1(23－25 行)来看,p 最后会将 q 从其 suspected 中删除。得证。

定理 3 在部分同步系统中,使用算法 1 的故障检测器可以得到等价于◇P 类故障检测器的输出结果。

证明:根据定理 1 和定理 2,定理 3 得证。

5 实验结果及其分析

通过实验,对算法 1 实现的故障检测器的性能进行了验证,实验环境由一台位于湖南衡阳南华大学和一台位于湖北华中科技大学的两台计算机组成。其中,位于南华大学的被作为接收机,位于华中科技大学的假设作为 P2P 存储系统节点,对外发送消息保持存活状态,所有消息通过 UDP 协议传输,检测周期设定为 1000ms。

实验 1 故障检测时间 TD 的比较

从图 1 所示的实验结果可以看出,Chen 的故障检测时间基本保持在一个稳定的值域空间,只是由于网络延时的影响有轻微的抖动。算法 1 的故障检测时间从总体上看明显要低于 Chen 的检测时间,但在某些时刻由于受网络状况的影响要略高于 Chen 的。这证明算法 1 实现的故障检测方法弥补了 Chen 的自适应故障检测器的不足,具有更好的检测服务质量。

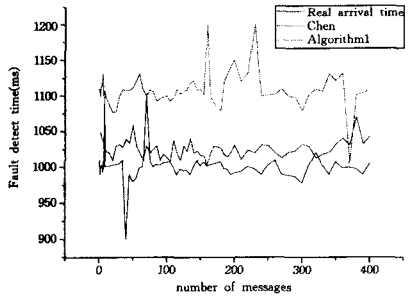

图 1　Chen 和算法 1 的故障检测时间的比较

实验 2　故障检测精度 P_A 的比较

为了检验算法 1 实现的故障检测器实际运行的性能，将其与另外一种有代表性的基于 QoS 的故障检测器（Chen 的 NFD-E 检测器）进行了实验对比。NFD-E 检测算法是 Chen 提出的一系列算法中性能最好的，而且系统假设相近（没有对同步时钟和延迟分布的任何假设）。选定时间窗口大小为 300，比较算法 1 和 Chen 的 NFD-E 的故障检测精度 P_A。检测时间设定为 1 个小时。

从图 2 可以看出，本文提出的故障检测算法 1 故障检测精度要明显高于 Chen 的算法，并且随着心跳到达时间窗的增大可以获得更高的故障检测精度。

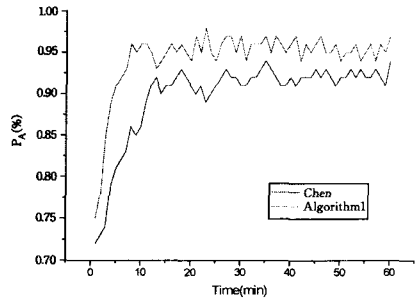

图 2　Chen 和算法 1 的故障检测精度的比较

实验 3　故障检测器 CPU 占用率（Δ）的影响

P2P 存储系统提供高可用的存储服务，实验表明，故障检测时间是影响到系统高可用性的重要因素。检测时间越长，系统服务中断的时间也越长，因此要求更加快速准确的故障检测。对于系统中监测的应用服务而言，可以通过更短的故障检测周期来完成，这样可以更快速地发现系统中的错误，但付出的代价是系统运行高可用性软件的开销增大。

CPU 占用率是系统性能的主要参考指标，如果在其他条件相同的情况下，CPU 占用率越低，系统的性能越好，反之就越差。由于系统中故障检测的时间间隔对 CPU 的利用率影响较大，因此，为了降低 CPU 占用率，作了下面的测试：修改 usleep()的时间，通过 top 命令来查看各项数据。测试数据如图 3 所示。

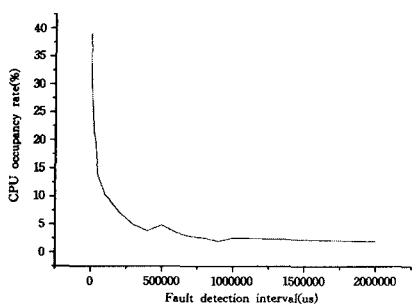

图 3　故障检测周期对 CPU 占用率的影响

综合故障检测周期与性能的平衡，在系统中给定的周期取值范围为 0.1s～0.4s。在这个范围内，CPU 占用率维持在 2%～10%，这一范围是较合理的。如果周期低于 0.1s，那么故障检测的实时性会得到进一步提高，但 CPU 占用率会显著提高，这对系统造成过于大的负担，反而使得整个系统性能下降；如果周期高于 0.4s，则系统的实时性会严重下降，并且当周期大于 0.4s 之后，基本 CPU 占用率维持在 3%左右，即周期的延长并不能显著地降低 CPU 的占用率。

结束语　P2P 存储系统具有节点高度动态异构的特点，在这样一种环境中部署故障检测系统以及节点的管理都是一种挑战。在故障检测时间和故障检测精度之间需要作出较好的权衡。较长的故障检测时间可以获得较好的故障检测准确率，而较短的故障检测时间可以获得好的系统可用性，但是付出的是增加网络负载和高错误故障检测结果。本文设计的动态心跳机制，不要求心跳消息的到达时间满足何种概率分布，且设计的修正值可以较好地反映系统的实时特征。经仿真实验表明：在实验确定的心跳消息到达时间序列维数下，与采用自适应的故障检测算法相比，其出错率大大减小，故障检测时间更短，可用于分布式计算环境下的故障检测。下一步研究工作是通过理论分析和实验验证，深入分析 P2P 存储系统的节点管理策略。

参 考 文 献

[1] Dabek F, Kaashoek M, Karger D, et al. Wide-area cooperative storage with CFS[C]// 18th ACM Symposium on Operating Systems Principles (SOSP'01). October 2001

[2] Zhang Z, Lian Q, Lin S, et al. BitVault : a Highly Reliable Distributed Data Retention Platform [R]. MSR-TR-2005-179. Microsoft Research, 2005

[3] Zhang Z, Lin S, Lian Q, et al. RepStore : a self - managing and self - tuning storage backend with smart bricks[C]// Proc. of International Conference on Autonomic Computing. 2004:122-129

[4] Rowstron A, Druschel P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility[C]// Proceedings of the eighteenth ACM symposium on Operating systems principles. 2001:188-201

[5] 陈宁江，魏峻，杨波，等. Web 应用服务器的适应性故障检测[J]. 软件学报，2005，16(11)：1929-1938

[6] Chandra T D, Toueg S. Unreliable failure detectors for reliable distributed system[J]. Journal of the ACM, 1996, 43(2): 225-267

[7] 吉培荣，黄巍松，胡翔勇. 无偏灰色预测模型[J]. 系统工程与电子技术，2000，22(6)：6-7

[8] Huang Zunguo, Lu Xicheng, Wang Huaimin. A Diversified DynamicRedundancy Method Exploiting the Intrusion Tolerance [C]// ISW 2000 Proceedings. 2000:217-221

[9] Chen W, Toueg W, Aguilera M K. On the quality of service of failure detectors[J]. IEEE Transactions on Computers, 2002, 51 (1):561-580

[10] Hayashibara N, Cherif A, Katayama T. Failure detectors for large-scale distributed systems [C] // Proceedings 21st IEEE Symposium on Reliable Distributed Systems. New Jersey: IEEE, 2002:404-409

[11] Bertier M, Marin O, Sens P. Implementation and performance evaluation of an adaptable failure detector[C]// Proceedings of the International Conference on Dependable Systems and Networks. New Jersey: IEEE, 2002:354-363

（下转第 77 页）

行操作首先需要登录系统。然而，节点崩溃、网络断开的时间太久或者网络断开时用户退出系统，都将导致用户登录 session 失效。当这些情况发生时，无论是 MS 还是 SS 都无法对操作进行回退。为了解决这一问题，系统引入了回退队列来暂存这些需要回退的操作，队列结构如图 5 所示。

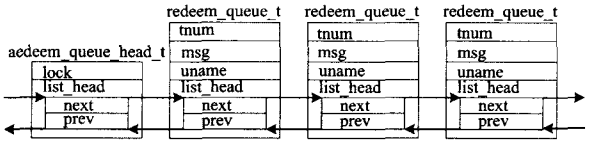

图 5　回退队列结构

用户重新登录系统时，认证模块在用户身份认证完成后激活对回退队列的处理。回退队列处理例程从队列头开始遍历队列，查找与当前登录用户名匹配的项。一旦找到匹配项，就依据 msg 域记录的操作类型、参数等信息进行回退处理。队列先进后出特性保证了用户的后一个操作先于前一个操作被回退，因此从队列头开始的回退处理保证了元数据能够依次回退到原始的状态。

3　2PC-MP 故障恢复

在恢复中假设故障只有节点失效和网络异常的情况，日志记录写入的原子性由同步日志方式保证，元数据修改操作的原子性由节点本身的事务机制保证。由于采用 undo 日志记录的设计方式，回退操作复原的始终是元数据的原始属性，因此回退操作是幂等的，在恢复过程中允许系统再次异常。

1）MS 发送元数据处理请求时超时

MS 关闭和 SS 之间的连接，终止协议的执行。收到元数据处理请求的 SS 在执行完请求发送处理结果后将发现连接已经断开，于是它对操作进行回退，同时在 log 中写入 Abort 记录，最后终止参与此协议。由于 MS 没有对元数据做任何处理，也没有记录任何日志，收到请求的 SS 会对元数据的修改自动进行回退，同时写入 Abort 日志记录。因此，这一情况下日志记录是完整的，元数据是一致的。

2）MS 等待投票消息时超时

MS 对操作进行回退，在 log 中写入 Abort 记录，向返回 Success 消息的 SS 发送 Abort 消息，关闭和 SS 之间的连接，终止协议的执行。发送 Success 消息的 SS 将收到 Abort 消息，然后对操作进行回退，同时在 log 中写入 Abort 记录，最后终止参与此协议。其他的 SS 在执行完请求发送处理结果后将发现连接已经断开，于是将对操作进行回退，同时在 log 中写入 Abort 记录，最后终止参与此协议。由于 MS 和 SS 均会对元数据的修改进行回退并写入 Abort 日志记录，因此这一情况下日志记录是完整的，元数据也是一致的。

3）SS 等待 Submit 或者 Abort 消息时超时

由于 SS 不知道 MS 是决定 Submit 还是 Abort，因此当这种情况发生时，SS 进入不确定的状态。此时，SS 将消息等待转移到悬挂队列中，然后退出协议的执行。

4）MS 等待 Finish 消息时超时

MS 将消息等待转移到悬挂队列中，继续等待接受其他 SS 发出的 Finish 消息。

5）MS 崩溃

当 MS 崩溃后重新启动时，它将由后向前搜索其日志。

（1）搜索到的是 Abort 或者 Finish 记录，则在匹配数组中添加其事务序号。

（2）搜索到的是 Submit 记录。如果事务序号不在匹配数组中，那么它必定处于协议的第 2 阶段。此时，MS 根据 Submit 记录中的 SS 列表生成 Finish 消息等待，并将它们加入悬挂队列中，然后在匹配数组中添加此事务序号。其他情况不进行处理。

（3）找到的是 Ready 记录。如果事务序号不在匹配数组中，那么它必定处于协议的第 1 阶段。此时，MS 根据 Ready 记录中的操作类型和参数生成对应的回退操作，将其加入到回退队列中。其他情况不进行处理。

6）SS 崩溃

当 SS 崩溃后重新启动时，它将由后向前搜索其日志。

（1）搜索到的是 Abort 或者 Submit 记录，则在匹配数组中添加其事务序号。

（2）找到的是 Ready 记录。如果事务序号不在匹配数组中，那么 SS 在崩溃发生时可能在等待 MS 的 Submit 或者 Abort 消息，于是 SS 生成 Ready 消息等待，将其加入到悬挂队列中。其他情况不进行处理。

结束语　分布式加密存储技术集成了加解密技术和分布存储技术，是当前信息安全领域新的研究方向和热点。KESS 采用元数据集中管理、文件数据分散存储的带外管理机制，满足了系统对高性能、大容量的要求。由于加解密的原因，部分元数据同时需要存储在 SS 上。为了解决这些元数据处理时引发的一致性问题，提出了 2PC-MP 协议。另外，通过模拟各种网络异常和节点故障对协议进行了测试。结果表明，协议在节点失效或网络异常恢复后，能够保证元数据的一致性，提高分布式元数据处理的性能。

参 考 文 献

[1] Tanenbaum A S, van Steen M. Distribute Systems Principles and Paradiams[M]. Beijing: TsingHua University Press, 2004: 307-312

[2] Garcia-Molina H, Ullman J D, Widom J. Database System Implementation[M]. Beijing: China Machine Press, 2001: 310-318

[3] Lewis P M, Bernstein A, Kifer M. Database and Transaction Processing[M]. Beijing: China Machine Press, 2005: 655-661

[4] Gray J, Reuter A. Transaction Processing : Concepts and Techniques[M]. Beijing: China Machine Press, 2004: 357-378

（上接第 52 页）

[12] Ma L, Barner K E, Arce G R. Satistical Analysis of TCP's Retransmission Timeout Algorithm[J]. IEEE/ACM Transactions on Networking, 2006, 14(2): 383-396

[13] 田东，陈蜀宇，陈峰. 一种网格环境下的动态故障检测算法[J]. 计算机研究与发展，2006，43(11)：1870-1875

[14] Hayashibara N, D'efago X, Yared R, et al. The f accrual failure detector[C]// SRDS IEEE Computer Society. 2004: 66-78

[15] Satzger B, Pietzowski A, Trumler W, et al. A Lazy Monitoring Approach for Heartbeat-Style Failure Detectors[C]// Proceedings of the Third International Conference on Availability, Reliability and Security. 2008: 404-409

[16] 邓聚龙. 灰色预测与决策[M]. 武汉：华中理工大学出版社，1988

[17] 吉培荣，胡翔勇，熊冬青. 对灰色预测模型的分析与评价[J]. 水电能源科学，1999(2)：42-441