一种基于 iSCSI 的自适应故障检测器的研究*)

杨 光 周敬利 刘 钢
(华中科技大学国家光电实验室 武汉 430074)

摘 要 故障检测器是构建可靠的 iSCSI 存储系统所必需的基础组件。本文实现了一种 iSCSI 系统中自适应故障检测器 iAFD (adaptive failure detector for iSCSI)。根据心跳(heartbeat)策略,设计了一种自适应心跳机制。故障检测器通过估计预期到达时间来提供一个探测时间,并动态地估算心跳消息超时时限,以适应系统状态的变化,减少故障检测服务的错误。实验表明,该方法与其它的故障检测方法相比,故障检测出错次数较少,检测时间较短,并能够适应高可靠计算系统状况的变化,在侦测的实时性和正确性上提供较好的平衡。

关键词 故障检测,iSCSI 系统,心跳算法,自适应性

Research of an Adaptive Failure Detector on iSCSI

YANG Guang ZHOU Jing-li LIU Gang
(National Op-to Electrical Laboratory, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract Failure detector is one of the fundamental building blocks to build dependable communication applications over iSCSI systems subject to faults. In this paper, we propose a new implementation of an adaptive failure detector-iAFD (adaptive failure detector for iSCSI). This implementation is a variant of the heartbeat failure detector which is adaptable. It dynamically estimates the heartbeat detection timeout and transmission delay of the system. It adapts to the change of the system state so as to reduce false detections. The experimental results show that the failure detector has less false detections and shorter detection time compared with normal failure detector. It can adapt the change of the state of high reliable computing system, and achieve a compromise between a good detection time and the need of avoiding false detections.

Keywords Failure detection, iSCSI system, Heartbeat, Adaptability

1 引言

故障检测器是构建可靠的 iSCSI 系统所必需的基础组件,其应用极为广泛:网络通讯协议[1]、组成员管理[2,3]、集群计算机管理[4]等多个相关领域。众所周知,在存在失效进程的系统中,由于无法正确区分失效进程和反应很慢的进程,导致很多重要的基础性问题(例如一致性问题)无法解决[10]。

为了解决故障检测的不可靠问题,Chandra 与 Toueg 在文献[5]中最先提出了不可靠故障检测器(unreliable failure detector)的概念。不可靠故障检测器的任务是提供系统中故障节点的信息,这些信息主要以被怀疑节点列表的形式记录在各节点中。因为节点发生故障以后,故障监测器需要花费一定的时间开始怀疑故障节点,并且故障监测器也可能错误地怀疑一个正确的节点,所以故障监测器不能保证被怀疑节点列表中的节点一定发生故障或者列表包含了所有故障节点。

Chandra 与 Toueg 之后,人们在故障检测方面进行了大量研究。这些工作大部分是基于部分同步(partially synchronous)系统模型,利用超时判断节点是否失效。Chen[6]首先提出了定量的 QoS 故障检测评价参数,用来衡量故障检测器的有效性。Chen 之后,Nunes[9], Bertier[11]等人继续研究了基于 QoS 的故障检测器。根据 QoS 需求与心跳消息到达时间,基于 QoS 的故障检测器可以动态调节自身参数,使得检测结果能够达到速度与精度的要求。

为此,本文提出了一种新的失效检测器——iAFD (adaptive failure detector for iSCSI)。在 iSCSI 系统中,每个 Initiator 模块上的故障检测器监控与它相连的 Target 模块的状态,并维持一个包含被怀疑已失效的 Targets 模块的列表。本文中我们将研究在 iSCSI 系统中如何实现动态调整故障检测。我们结合心跳(heartbeat)策略,设计了一种自适应心跳机制。在系统中每个 Target 模块会定时发送"I am alive"消息给它所连接的 Initiators 模块。为了获得小的检测时延,系统自动调整检测时间。iAFD 能通过自动调节超时时间来适应不断变化的网络状况。

实验表明,该方法与其它的故障侦测方法相比,故障检测出错次数较少,检测时间较短,并能够适应高可靠分布计算系统状况的变化,在检测的实时性和正确性上提供较好的平衡。

2 不可靠故障检测

我们简单地介绍故障检测器和几个评价性能的方面。每个 Initiator 模块都有一个故障检测器的接收信息部分,且都维持一个被怀疑已失效的 Targets 列表。由于故障检测器不

*)本文得到国家自然科学基金项目"支持内容感知的 IP 存储网络系统研究"(60673001)资助。杨 光 博士生,主要研究方向为计算机网络存储;周敬利 教授,博士生导师,主要研究方向为计算机多媒体网络通信、高性能网络接口和计算机网络存储;刘 钢 博士生,主要研究方向为计算机网络存储。

是永远可靠的，它可能错误地将一个还在运行的 Target 也加入到列表中。但如果故障检测器后来发现这个 Target 还在运行，它会将这个 Target 从列表上移除。故障检测器应满足完整性(completeness)和准确性(accuracy)。完整性表示故障检测器能永远怀疑失效的 Target，准确性表示故障检测器不会怀疑正确的 Target。

强完整性(Strong completeness)：每个失效的 Target 模块在一定时间之后会被所有正确的 Initiator 模块永远判定为失效。

最终强精确性(Eventual strong accuracy)：在一定时间之后所有正确的 Target 模块都不会被正确的 Initiator 模块认为发生失效。

文献[6]提出了几个方面去评价一个故障检测器的 QoS (Quality of Service)。

故障侦测时间(T_D)：T_D 是从 Target 失效的时刻到 Initiator 模块开始永远怀疑它的时间。

错误间隔时间(T_{MR})：T_{MR} 是两次发生错误输出之间的时间间隔。

错误持续时间(T_M)：T_M 是故障检测器修正一次错误怀疑所用的时间。

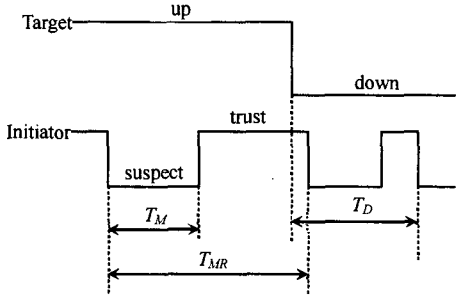

图 1　T_D，T_{MR} 和 T_M

3　故障检测策略

本节中介绍我们的系统模型，介绍两种经典的故障检测的实现。

3.1　Push 策略

Push 策略是实现故障检测最普通的方式之一。每个 Target p 周期地发送“I am alive”给 Initiator。若一个 Initiator q 在一定的时延之后没有收到从 Target p 发送过来的消息，则会开始怀疑 Target p 并将 p 加入进怀疑列表。若 Initiator q 在一段时间之后收到 Target p 发送过来的消息“I am alive”，则 q 知道 p 还是正常的，就将 p 从列表中删除。

Push 周期(Δ_i)：Δ_i 是两次发送“I am alive”消息之间的时间间隔。

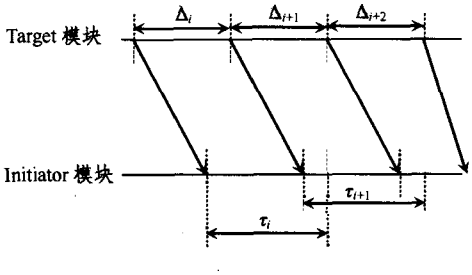

图 2　Push 模式

超时(Δ_{to})：Δ_{to} 是 Initiator p 最后一次收到 Target p 发送的“I am alive”消息的时刻到 Initiator q 开始怀疑 p 的时刻之间的时间间隔。

这种方式的优点是探测时间与最后一个 heartbeat 消息相互独立。这个修改增加了准确性，因为这样避免了设置过小的超时值。

3.2　Pull 策略

Initiator 模块 q 通过定期向 Target 模块 p 发送“Are you alive?”消息来监测 Target 的状态。当 Target 接受到这些消息后，会发送“I am alive”消息给 Initiator 模块。若 Initiator 在一定时间内没有收到 Target 发出的回执信息，它会将 Target p 加入到怀疑列表中。如果 Initiator q 在一段时间后接收到从 Target p 发出的回执信息，则会将 p 从怀疑列表中删除。

询问周期(Δ_i)：Δ_i 是两次发送“Are you alive?”消息之间的时间间隔。

超时(Δ_{to})：Δ_{to} 是 Initiator q 向 Target p 发送“Are you alive?”消息到最后一次收到 Initiator q 开始怀疑 p 的时刻之间的时间间隔。

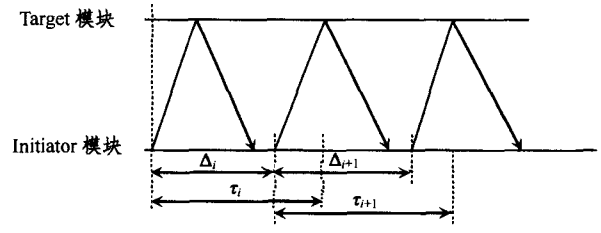

图 3　Pull 模式

3.3　策略比较

Push 模式的优点是：

(1)与 Pull 模式相比，Push 模式得到相同性能所需的消息数目仅是 Pull 模式的一半。

(2)Push 模式只需要预测传输“I am alive”信息的延时，而 Pull 模式需要预测传输“Are you alive?”信息的延时、反应延时和传输“I am alive”信息的延时。因此对 Push 模式消息的预测要比对 Pull 模式消息的预测简单。

3.4　系统模型

我们假设一个 iSCSI 系统有 Initiator 模块集合 Ω 和 Target 模块集合 Φ。Initiator 和 Target 分布在不可靠的网络中，并假设网络存在比较大的时延，而且通讯链路会丢包。在这个 iSCSI 系统中，每个 Initiator 模块中都有一个 iAFD 故障检测器的接收信息部分，从 fd_1 到 fd_n。而每个 Target 模块中都有一个 iAFD 故障检测器的发送信息部分。这里 fd_q 对应的是一个 Initiator 模块 $q(q\in\Omega)$。iAFD 采用的是 Push 方式的监控策略[8]。

每个 fd_q 维持一个包含已失效 Target($Suspect_q$)的列表。因此，在某个时刻 t，一个 Initiator 模块 $q(q\in\Omega)$怀疑一个 Target 模块 $p(p\in\Phi)$，Target p 应该被记入 fd_q 所保持的那个列表中。由于故障检测器是基于事件的时延来做出判断的，因此 fd_q 可能会把正确的 Target p 错误地当成已失效。当 fd_q 发现它的判断是错误的时候，fd_q 会将 Target p 从列表中删除。

4　故障检测器

4.1　预测到达时间

本节中我们研究如何预计 heartbeat 消息的到达时间。

Δ_i 和 Δ_{to} 参数影响了探测的 QoS。Δ_{to} 非常重要，因为它确定了探测时间。使用每个 Initiator 模块拥有的本地信息来预计 Δ_i。这些信息会受到 heartbeat 信息到达时间和 Δ_i 周期的限制。也就是说，heartbeat 信息的到达时间随着网络负载和用户负载的改变而改变。

在 iAFD 设计中，故障检测器分为两层：第一层通过准确的估计来优化探测时间；第二层调整仲裁超时间，以提高检测的正确性。iAFD 能在不增加错误探测数量的情况下尽可能准确预测到达时间。因此我们比较两种故障检测器：第一种是文献[6]中提出的 NFD-U 故障检测器，第二种是 iAFD 故障检测器。

4.1.1 Chen 的 NFD-U 故障检测器

文献[6]中预测 heartbeat 信息到达时间(EA)，并增加一个固定的安全时间余量。期望达到时间的计算将在下面显示，而安全时间余量是根据 QoS 的需求初步计算得出的。

进程 q 只考虑最近收到的 n 个 heartbeat 信息 $m_1, m_2, \cdots, m_n$。而 $A_1, A_2, \cdots, A_n$ 表示按照 q 的本地时钟这些信息接收的时间。当 n 条信息接收后，$EA_{(k+1)}$ 可以计算为：

$$EA_{(k+1)} \approx \frac{1}{n}\left(\sum_{i=k-n}^{k} A_i - \Delta_i \times i\right) + (k+1) \times \Delta_i$$

下一个超时 Δ_{to}(到达下一个时间点 $\tau_{(k+1)}$)由 EA 和固定安全时间余量 α 组成。EA 代表理论上的到达时间。安全时间余量是用来避免由于传输时延或处理器过载产生的错误探测。

$$\tau_{(k+1)} = \alpha_{(k+1)} + EA_{(k+1)}$$

这种方法提供了一种对下次到达时间较准确的预测。然而，它使用了一个固定的安全时间余量。

4.1.2 iAFD 故障检测器

根据 Chen 的方法我们可以预测出到达时间，而且可以动态地计算出安全时间余量。Chen 对 $EA_{(i+1)}$ 的预测是通过对最近接收的 n 个消息到达时间计算平均值来实现的。现在我们将它转换为一个递归公式：直到 Initiator 模块接收到从 Target 模块发送过来的 n 条信息，则 Initiator 能预测下一条信息的到达时间：

$$U_{(i+1)} = \frac{A_i}{i+1} + \frac{k \times U_i}{i+1} \text{(到达时间的平均值)}$$

$$EA_{(i+1)} = U_{(i+1)} + \frac{i+1}{2} \times \Delta_i \quad (U_{(1)} = A_0)$$

若 Initiator 接收的信息不止 n 条，则

$$EA_{(i+1)} = EA_{(i)} + \frac{1}{n}(A_i - A_{(i-n-1)})$$

我们假设时间余量不是恒定的，每次收到一个信息后都会调整安全时间余量。在系统中，我们使用基于错误的余量预测值($\alpha_{(i+1)}$)。考虑到网络流量是一个很重要的因素，$\alpha_{(i+1)}$ 会自我调整到预测值，以满足系统的需要。只要预测值发生变化，则要重新计算时间余量。

与 Jacobson 的方法[7]相似，时间余量 $\alpha_{(i+1)}$ 每次根据故障监测器收到的信息和网络负载状态的变化而改变。在 t 时刻接收到一个新信息，则新的时间余量为

$$\alpha_{i+1} = \alpha_i + c(|EA_I - A_i| - \alpha_i)$$

其中 c 是平滑系数，本文中设置 $c=0.25$。

下一次超时的预测值 $\Delta_{to(i+1)}$：

$$\tau_{(i+1)} = EA_{(i+1)} + \alpha_{(i+1)}$$

4.2 探测时间的动态调整

作为一个服务组件，应用程序的 QoS 需求是设计失效检测器的主要依据。故障检测器大部分采用了基于超时的心跳检测策略，它通过周期性地发送检测消息来检测对方的状态。在这种机制下，心跳发送周期 Δ_i 和检测超时值 Δ_{to} 可以决定检测器的行为，但是随着网络状况(如流量、传输延迟等)的不断变化，却无法确保应用程序得到的 QoS 能够一直满足要求[6,16]。iAFD 通过自动调节参数 $\Delta_I(T)$ 来适应不断变化的网络状况。

5 故障检测算法

5.1 算法

本节介绍 iAFD 故障检测器的算法实现。iAFD 根据探测时间对下一个 heartbeat 消息到达时间的预测来实现基本的故障检测服务。这个数据预测到达时间包括期望达到时间和动态时间余量。iAFD 的目的不是完全避免错误探测，而是达到错误探测次数与探测时间的准确率上的平衡。图 4 显示了故障检测器的算法。

```
初始化：
Suspect_I ← ∅
for all Target
  Δ_I(T)=0{Δ_I(T)仲裁探测时间}
  τ_0(T)=0 {最初 Initiator 模块怀疑所有的 Target 模块}
  EA_0(T)=U_0(T)=0
  delay_0(T)=initia value
  α_0=0
  A_0(T)=0
  k(T)=-1 {k(T)保存 Initiator 模块收到的来自 Target 模块信息
           的最大序列号}
算法：
  Target 模块：
  在时刻 i·Δ_i，发送 heartbeat 信息 m_i 给 Initiator 模块。{Δ_i是检测
  周期}
  Initiator 模块：
  在时刻 t 收到来自 Target 模块的信息 m_j
  if j>k(T) then
    k(T) ← j
    α_{i+1}=α_i+c(|EA_I-A_i|-α_i)
    if j < n then
      U_{i+1}=[t/(i+1)]×[i/(i+1)]U_i
      EA_{i+1}=U_{i+1}+[(i+1)/2]Δ_i
    else
      EA_{i+1}=EA_i+(1/i)×(t-A_{i-n-1}(T))
    end if
    A_i(T) ← t{A(T) 是一个保存最近 n 个来自 Target 的数据的队
      列}
    τ_{(i+1)}(T)=EA_{i+1}(T)+α_{i+1}(T)
    if Target q∈Suspect_I then
      Suspect_I ← Suspect_I-{q} {信任 Target q}
      State_q=S→T {Target q 的状态由失效变为有效}
      Δ_I(T) ← Δ_I(T)+1 {增加仲裁超时}
    end if
任务：
  upon τ_{(i+1)}(T)=当前时间 {此时 Initiator 模块没有收到 Target 模
    块发送的信息}
  在仲裁超时 Δ_I(T)内继续等待
  if 没有收到来自 Target q 的信息
  Suspect_I ← Suspect_I ∪ {q} {开始怀疑 Target q }
  State_q=T→S { Target q 的状态由有效变为失效}
```

图 4 iAFD 实现

每个 Target 模块周期地向它所连的 Initiator 模块发送“I am alive”消息。当 Initiator 模块接收到从 Target 模块发送的“I am alive”消息后，会计算下一条从 Target 模块发送的 heartbeat 信息的期望到达时间 EA_i 和安全时间余量 α。然后 Initiator 模块会得出 Target 模块下一个信息到达时间 $\tau(i)$。如果 Initiator 模块现在怀疑 Target 模块失效，后来 Initiator 模块发现自己错误，Target 模块还在运行。这时 Initiator 模块会适当地增加仲裁超时 $\Delta_I(T)$，因为 Initiator 模块认为以前对 Target 设定的超时值过小。

当 Initiator 模块在下一个信息到达时间点 $\tau(T)$ 之前没有收到 Target 模块发送的“I am alive”信息时，任务开始运

行。Initiator 模块会继续等待 Target 模块发送的消息，直到仲裁超时 $\Delta_I(T)$。如果 Initiator 模块依然没有收到 Target 模块发送的消息，则 Initiator 模块会认为 Target 模块失效。

5.2 证明

按照 Chandra 和 Toueg 对于失效检测器的分类，在部分同步的分布式环境中，iAFD 故障检测器等效于一个$\diamond p$失效检测器($\diamond p$(Eventually Perfect failure detector)：即最终完全故障检测器，满足强完全性与最终强精确性)。因为$\diamond p$失效检测器满足强完全性与最终强精确性，本节将给出 iAFD 故障检测器满足强完全性及最终强精确性的证明。

定理 1(强完整性) 每个失效的 Target 模块在一定时间之后会被所有正确的 Initiator 模块永远地判定为失效。

$\exists t_0: \forall t \geqslant t_0, \forall p \in correct(I), \forall q \in crashed, q \in suspect_p(I)$

定理 2(最终强精确性) 在一定时间之后所有正确的 Target 模块都不会被正确的 Initiator 模块认为发生失效。

$\exists t_{bound}, \forall t \geqslant t_{bound}, \forall p, q \in correct(I), q \notin suspect_p(I)$

强完整性：

若引理 1,2 被证明，则定理 1 也被证明。也就是说存在一个 t_{mute} 时间，在此之后没有一个正确的 Initiator 模块收到已失效的 Target 模块发送的 heartbeat 信息。还存在一个时间 $t_{timeout}$，在此之后所有正确的 Initiator 模块会永久地认为 Target 模块已失效。

引理 1 如果在时刻 t_{carsch} Target 模块失效，则存在一个时间 t_{mute}，在此之后 Initiator 模块不会接收到 Target 模块发送的信息。

$t_{mute} \leqslant t_{crash} + \Delta_{msg}$

证明：因为系统中各模块运行时间与消息传递时间的上限仅存在于一个未知(但有限)的 T_{GST} 时间之后，假设 T_{GST} 之前发送的心跳消息已经被传递并被处理，那么对于 T_{GST} 之后发送的任意消息 i：

$\exists t_{GST}: \forall m_i \mid t_{si} \geqslant t_{GST}: (t_{ri} - t_{si}) < \Delta_{msg}$

t_{si} 是 Target 模块发送消息 i 的时刻，t_{ri} 是 Initiator 模块接收消息 i 的时刻。

假设 Target 模块在时刻 t_{crash} 时失效，则 Target 模块会停止发送“I am alive”消息。

$\nexists m_i \mid t_{si} \geqslant t_{crash}$

存在一个时刻 $t_{ri} + \Delta_{msg}$，在此之后 Initiator 模块不会接收到从 Target 模块发送过来的消息 i，因此 Initiator 模块从时刻 $t_{crash} + \Delta_{msg}$ 之后不会再接收到 Target 传送过来的任何消息。

引理 2 假设 Initiator 模块从 Target 模块接收了 i 个消息的心跳序列，当 Initiator 模块不再从 Target 模块接收任何心跳消息时，存在一个时刻 τ_i，τ_i 后 Initiator 模块开始怀疑 Target 模块。

证明：当 Initiator 模块从 Target 模块接收到一个信息，Initiator 模块会计算一个新的 τ_i，τ_i 之后 Initiator 模块开始怀疑 Target 模块。那么，当 τ_i 收敛时，肯定存在 Initiator 模块开始怀疑 Target 模块的时刻。我们必须证明 τ_i 是收敛的。

$\tau_i = (EA_i + \alpha) + \Delta_I(T)$

若 $i \leqslant n$，

$$EA_{i+1} = EA_i + \frac{1}{i}(A_i - A_0)$$

……

$$EA_{i+1} = (A_1 - A_0) + \frac{1}{2}(A_2 - A_0) + \cdots + \frac{1}{k}(A_i - A_0)$$

A_i，i 是收敛的，因此 EA_i 也是收敛的。

若 $i > n$，

$$EA_i = EA_{i-1} + \frac{1}{n}(A_{i-1} - A_{i-n-1})$$

$$EA_i = EA_{i-2} + \frac{1}{n}[(A_{i-1} - A_{i-n-1}) + (A_{i-2} - A_{i-n-2})]$$

……

$$EA_i = EA_{n-1} + \frac{1}{n}[(A_{i-1} + A_{i-2} + \cdots + A_{i-n}) - (A_{n-1} + A_{n-2} + \cdots + A_0)]$$

$$EA_i < EA_{n-1} + \frac{1}{n}(nA_{i-1} - nA_0) = EA_{n-1} + A_{i-1} - A_0$$

因为心跳消息到达 Initiator 模块的时刻大于 Target 模块发出该心跳消息的时刻，如果用 t_{si} 表示第 i 个消息发出的时刻，则 $t_{s0} < A_0$，并且由引理 1 可知 $A_{i-1} < t_{s(i-1)} + \Delta_{msg}$，则

$EA_i < EA_{n-1} + A_{i-1} - t_{s0} < EA_{n-1} + t_{s(i-1)} + \Delta_{msg} - t_{s0}$

第 i 个心跳消息发送的时刻与第 0 个心跳消息发送时刻之间相隔一系列的发送周期，即 $\Delta_1, \Delta_2, \cdots, \Delta_{i-1}$。

$EA_i < EA_{n-1} + \Delta_{msg} + (\Delta_{i-1} + \Delta_{i-2} + \cdots + \Delta_1)$

可以得出 $EA_{n-1} = A_0 + \Delta_{n-1}$，因此

$EA_i < EA_{n-1} + \Delta_{msg} + (i-1)\Delta_{\max}$

所以 EA_i 是收敛的。从文献[6]，我们计算出，$\Delta_{i+1} + \alpha_{i+1} < T_D^U$，即 $0 < \alpha_i \leqslant T_D^U$，因此 α_i 是收敛的。

当每次 Initiator 模块都在一定时间内知道 Target 模块正常工作时，则增加 $\Delta_I(T)$。存在一个时刻 t_{bound}，当 $\Delta_I(T)$等于 t_{bound} 时会避免错误探测，而 $\Delta_I(T)$也不会再增加。当 $\Delta_I(T)$大于 Δ_{msg} 后错误探测不会再发生。

$\exists t_{bound}, \forall t \geqslant t_{bound}, \Delta_I(T)(t) \geqslant \Delta_{msg}$

$\Delta_I(T)(t) = \Delta_I(T)(t+1)$

从引理 1 和引理 2，定理 1 得到证明。当 Initiator 模块没有接收到从 Target 模块发送的消息后，存在一个时间，在此之后 Initiator 模块会怀疑 Target 模块已失效。

最终强精确性：

如果 Initiator 模块的 $\tau_i(T)$足够大而能避免 Initiator 模块错误地判断有效的 Target 模块失效，则定理 2 就被验证了，在我们的模型中，如果引理 3 被证明了，则定理 2 也可以推论出。

引理 3 假设 Initiator 模块从 Target 模块接收心跳消息，经过某个时刻以后，Initiator 模块设置的时刻 τ_i 满足条件 $\tau_i > t_{si} + \Delta_{msg}$。

$\exists t_{bound}, \forall t \geqslant t_{bound}, \tau_i \geqslant t_{si} + \Delta_{msg}$

证明：从定理 1 的证明可知：

$$\forall m_i \begin{cases} t_{si} < EA_i \\ 0 \leqslant \alpha_i \end{cases}$$

$\exists t_{bound}, \forall t > t_{bound}, \Delta_I(T)(t) \geqslant \Delta_{msg}$

我们可知：

$\exists t_{bound}, \forall t > t_{bound}, \tau_i > t_{si} + \Delta_{msg}$

如果 $\tau_i > t_{si} + \Delta_{msg}$ Initiator 模块不会认为有效的 Target 模块失效，定理 2 得证。

6 实验结果及分析

6.1 恒定负载性能比较

恒定的负载意味着 heartbeat 信息的发送周期是不规律的。本节显示 iAFD 采用正确的探测时间来避免错误探测。

这个实验显示 iAFD 跟 Chen 的 NFD-U 相比能避免更多的错误探测，而且在相同时间内能维持一个更好的探测时间。

实验结果如表1所示。

表1 恒定负载实验结果

	iAFD	NFD-U
错误探测次数	17	19
平均错误持续时间(ms)	56	51
平均探测时间(ms)	5016	5089

6.2 服务质量分析

本小节验证iAFD是否可以满足用户故障检测服务质量的要求。设置一个Initiator模块与一个Target模块，Initiator模块利用iAFD故障检测器探测Target模块节点是否故障。

表2验证在相同 T_{MR}^{U} 与 T_{M}^{U} 条件下，T_{D}^{U} 变化时，iAFD的平均故障检测时间是否小于 T_{D}^{U}。假设 T_{MR}^{U} 为10000ms，T_{M}^{U} 为1500ms，表2中是 T_{D}^{U} 分别取值4000ms，4500ms，…，6000ms时经过仿真得到的 T_D。

表2 T_{D}^{U} 不同取值得到的 T_D(ms)

T_{D}^{U}	4000	4500	5000	5500	6000
T_D	2213	2567	2938	3306	3752

由表2可知，平均 T_D 在 T_{D}^{U} 的范围内。

图5验证在相同 T_{D}^{U} 与 T_{M}^{U} 条件下，T_{MR}^{L} 变化时，iAFD的平均错误间隔时间(T_{MR})是否大于 T_{MR}^{L}。假设 T_{D}^{U} 为5000ms，T_{M}^{U} 分别为1200ms与1500ms。图6验证在相同 T_{D}^{U} 与 T_{MR}^{L} 条件下，T_{M}^{U} 变化时，iAFD的平均错误持续时间(T_M)是否小于 T_{MR}^{L}。假设 T_{D}^{U} 为5000ms，T_{MR}^{L} 分别为7000ms与10000ms。

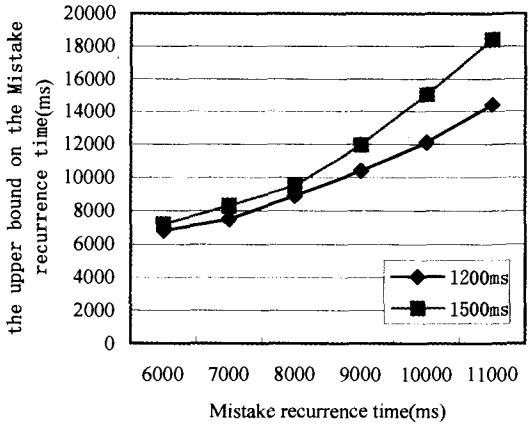

图5 错误间隔时间

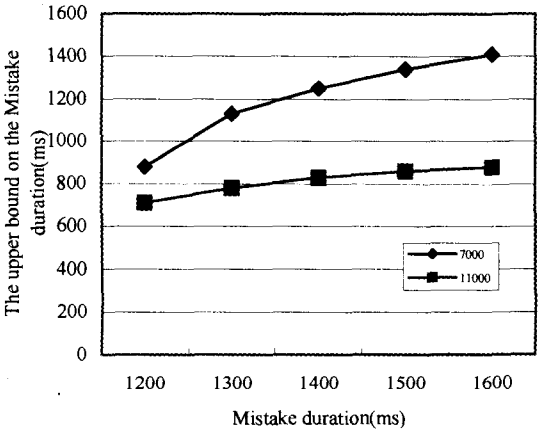

图6 错误持续时间

由图5可知，平均 T_{MR} 大于相应的 T_{MR}^{L}，当 T_{MR}^{L} 增大时，T_{MR} 也随之增大，并且，T_{M}^{U} 越大，则 T_{MR} 的增幅也会更大。而由图6可知，平均 T_M 小于相应的 T_{M}^{U}，当 T_{M}^{U} 增大时，T_M 也随时增大。但是，当 T_{MR}^{L} 增大时，T_M 的增幅会减小。

根据以上实验可知，iAFD故障监测器能够满足服务质量的要求。

结束语 本文设计了iSCSI系统中一种新的故障检测器iAFD。iAFD分为两层：第一层提供了对超时(Δ_{to})的一个基本预测；第二层调整仲裁超时间 $\Delta_I(T)$，以提高检测的正确性。iAFD达到了在探测时间和避免错误探测之间较好的平衡。

iAFD的主要特性是动态自适应性。它的探测时延(Δ_{to})由短期的动态安全时间余量和长期的动态期望到达时间组成。本文分析了衡量故障检测服务质量有3个参数，即检测时间、错误间隔时间、错误持续时间。iAFD故障检测器通过自适应调整预期心跳消息到达时间与安全时间余量保证服务质量的参数在一定范围内变化。本文证明了iAFD故障检测器满足强完全性与最终强精确性，理论上保证了iAFD的可靠。最后，实验说明iAFD故障检测器可以满足服务质量的要求。

参考文献

[1] Requirement for Internet Hosts-Communication Layers. Braden R, ed. RFC 1122, Oct. 1989

[2] Amir Y, Dolev D, Kramer S, et al. Transis: A Communications Sub-System for High Availability// proc 22nd Ann. Int'l Symp. Fault-Tolerant Computing. July 1992:76-84

[3] Birman K P, van Renesse R, et al. Reliable Distributed Computing with the Isis Toolkit. IEEE CS Press, 1993

[4] Pfister G F. In Search of Clusters. second ed. Prentice Hall, 1998

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

[6] Chen W, Toueg S, Aguilera M K. On the quality of service of failure detectors// Proc. of the First Int'l Conf. on Dependable Systems and Networks. 2002

[7] Jacobson V. Congestion Avoidance and Control// Proceedings of the ACM Symposium on Communications, Architectures and Protocols (ACM SIGCOMM). Aug. 1988:314-329

[8] Dolev D, Friedman R, Keidar I, et al. Failure detectors in omission failure environments// Symp. on Principles of Distributed Computing. 1997:286

[9] Nunes R C, Jansch-Poto I. QoS of Timeout-based Self-tuned Failure Detectors: the Effects of the Communication Delay Predictor and the Safety Margin// Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04). 2004

[10] Fischer M J, Lynch N A, Paterson M S. Impossibility of distributed consensus with one faulty process. Journal of the ACM, 1985, 32(2):374-382

[11] Bertier M, Marin O, Sens P. Implementation and Performance Evaluation of an Adaptable Failure Detector // Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing. 2004

[12] Shi X H, Jin H, Han Z F, et al. ALTER: Adaptive failure detection services for grid// Cantarella J D, ed. Proc. of the IEEE Int'l Conf. on Services Computing. Orlando: IEEE CS Press, 2005: 355-358

[13] Falai L, Bonvadalli A. Experimental evaluation of the QoS of failure detectors on wide area network// Tsuchiya T, ed. Proc. of the Int'l Conf. on Dependable Systems and Networks (DSN 2005). Yokohama: IEEE CS Press, 2005:624-633

[14] Aguilera M K, Delporte-Gallet C, Fauconnier H, et al. On implementing omega with weak reliability and synchrony assumptions// Borowsky E, ed. Proc. of the 22nd ACM Symp. on Principles of Distributed Computing. Boston: ACM Press, 2003: 306-314

[15] Muller M. Performance evaluation of a failure detector using SNMP. Technical Report, LSR-REPORT-2004-034. Lausanne: école Polytechnique Fédérale de Lausanne, 2004:12-24

[16] Chen N J, Wei J, Yang B, et al. Adaptive failure detection in Web application server. Journal of Software, 2005, 16(11): 1929-1938 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1929.htm

[17] Hayashibara N, Défago X, Yared R, et al. The φ-accrual failure detector// Titsworth F M, ed. IEEE Int'l Symp. on Reliable Distributed Systems (SRDS 2004). Florianopolis: IEEE Computer Society Press, 2004: 66-78

[18] Sotoma I, Madeira E R M. Adaptation-Algorithms to adaptive fault monitoring and their implementation on CORBA// Blair G, Schmidt D, Tari Z, eds. Int'l Symp. on Distributed-objects and Applications (DOA 2001). Rome: IEEE Computer Society Press, 2001: 219-228