# 基于 Web 服务组合的主从式失效检测器研究*

陈丽金,周 娅

(桂林电子科技大学 计算机与控制学院,广西 桂林 541004)

**摘 要**:针对 Web 服务组合环境对失效检测方法进行研究,设计主从式失效检测器和失效检测算法并对其进行分析,为 Web 服务组合监测提供有效的手段,实验模拟表明,该算法能有效降低错误率,提高系统可用性。

**关键词**:失效检测器;Web 服务组合;失效检测算法

**中图分类号**:TP393 **文献标识码**:A

## Research of primary-slave failure detection based on Web service composition

CHEN Li Jin, ZHOU Ya

(School of Computer Science and Control, Guilin University of Electronic Technology, Guilin 541004, China)

**Abstract**: The paper researches the failure detection methods based on the Web service composition, designs the primary-slave failure detector, analyses a failure detection algorithm, and then provides effective means to Web service composition monitoring. The experimental simulation show that this algorithm can effectively decrease the error and improve the system availability.

**Key words**: failure detector; Web services composition; failure detection algorithm

通过 Web 服务组合来动态生成新的应用系统,以满足企业的动态需求,已成为 Web 服务技术不断向前发展的技术动力和研究热点。但是在非可靠异构、分布式自治和快速变化发展的网络环境中,Web 服务组合运行可能会受到网络失效、通信模式变化、基础设施变更等多种问题的影响,这都将影响所提供的服务质量。复杂的网络环境、难以预测的各类故障以及无法确定的功能变化使得业务流程的功能与服务质量均难以得到保障,导致服务组合提供者可能由此失去客户的信任,因此,Web 服务组合的失效检测研究十分重要。失效检测器是软件系统可靠性保障的基本技术,其性能的好坏直接影响到系统容错的能力,而失效检测器的核心就是失效检测算法。Web 服务组合的失效检测是指 Web 服务组合在运行阶段的失效检测[1]。

当前服务组合的容错研究主要集中在标准协议的扩展、成分服务的容错以及 Web 服务组合容错平台等几个方面。参考文献[2]将监测及检测服务结合到面向服务架构 QoS 管理中,通过监测服务反馈的信息,诊断服务能够检测状态的变化,并利用一种基于图模型的方法对状态变化原因进行推断。参考文献[3]提出了一种基于模型的 Web 服务组合故障诊断方法,该方法向服务组合中的每个服务增加一个局部诊断器,并为整个组合服务设置一个全局诊断器,全局诊断器与局部诊断器进行协作,并推断 Web 服务组合发生故障的原因。参考文献[4]提出了一种基于反射技术的 Web 服务失效处理方法,通过在 Web 服务请求者和服务提供者配置反射层,使之检测到造成服务失效的各种状态,调整 Web 服务的内部结构和运行状态,使得 Web 服务适应环境的变化,以提高 Web 服务的适应性和健壮性。Marin[5]等人提出了适应网络状况变化的失效检测方法,适应性失效检测方法在一定程度上提高了失效检测器的检测准确度,但是由于没有考虑网络丢包的影响,不能有效地减少由于网络丢包而产生的误判,影响了适应性失效检测器的检测服务质量。

* 基金项目:广西省自然科学基金(0823101)

## 1 失效检测及其基本模型

失效检测(failure detection)是Web服务组合系统可靠性保障的基本技术,它对运行时系统的存活状态进行检测。提供失效检测功能的组件称为失效检测器FD(failure detector)[6],失效检测器是失效恢复、动态重启、可靠性通信、集群管理等功能的基础。在Web服务组合中需要提供良好的失效检测机制来及时检测服务器以及其中服务组件和应用组件的失效情况,从而提高系统的可靠性和可用性。

失效检测一般采用超时机制实现,即通过被检测对象在规定时间内有无应答来判断失效,有两种最基本的检测方法:心跳策略和轮询检测方法。下面分别对它们进行介绍。

心跳策略:由被检测进程按照一定的时间时隔定期地向失效检测器发送心跳信息"I am alive",通告它们依然存活。如果失效检测器超过某一期限没有收到心跳消息,失效检测器则认为其失效。该模式被检测对象向失效检测器"推"(push)失效事件,我们也称为"推"模式,其消息交换如图1所示。

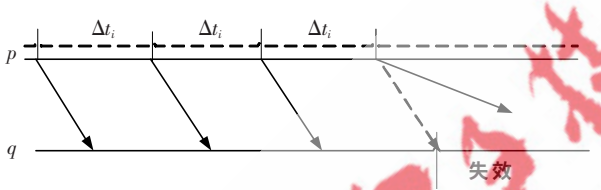


图1 push模型的监控消息

轮询检测:失效检测器$q$定时发送查询消息"Are you alive?",检查被检测对象$p$的状态,被检测对象$p$收到查询消息后返回应答消息"I am alive!",如果失效检测器超过一定时间间隔没有收到应答消息,则认为被检测对象失效。该模式失效检测器从被检测对象中"拉"(pull)失效事件,我们又称之为"拉"模式,其消息交换如图2所示。

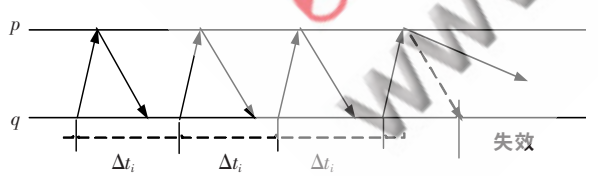


图2 pull模型的监控消息

两种检测模型各有优劣。pull模型需要失效检测器发送查询消息,并等待被检测对象返回应答,这样导致失效检测时间较长;同时,由于失效检测器发送消息,被检测对象返回消息导致同样的性能需要两倍数量的消息数目。然而,pull模型是一种主动检测方式,可以只在需要的时候才发起检测,而且,使用pull模型可以保证每个检测消息及其延时对检测结果的影响是独立的。

## 2 主从式失效检测算法

算法基本思想:在主失效检测器中,每个被检测进程周期性地向主失效检测进程发送不同序号的心跳消息,心跳消息的序号总是递增的。主失效检测器对收到的心跳消息的历史纪录作统计、分析,并计算出下一个心跳消息到达的时间上限,如果超过等待时限未收到心跳消息,系统自动启动从失效检测器。从失效检测进程主动发送问询消息给被检测进程,如果收到对被检测进程的应答消息,则认为被检测进程正常,返回主失效检测器;否则认为被检测进程失效。

### 2.1 主失效检测算法基本思想

主失效检测算法中,采用心跳策略。被检测进程$p$每间隔$t_i$周期性地向推失效检测器进程$q$发送消息$mp_i$($i$为检测消息的序号);进程$q$负责接收消息。进程$q$收到消息之后,首先将消息$mp_i$的接收消息时间($T_{current}-f$)保存到滑动窗口$W$中。当进程$q$超过$E(T_D)$没有收到进程$p$的发送消息$mp_i$时,即认为该系统资源失效,则启动从失效检测器进行确认。主失效检测器算法主要步骤如下:

```
For process p:
  for all i>0, at time △t_i* i,
  send mp_i to q,
  For process q:
   f=-1,                                  //心跳新鲜点
   upon receive mp_i from p at time T_current do;
   if f==-1
         then f=T_current;
   else {
         △T=T_current-f,
           f=T_current,
   append △T to W endif;
     }
   end if;
  if (△T>E(T_D))
   {
    execute S_transition,
    SlaveFD( ),                           //调用从失效检测器
   }
   else
   i=i+1;
 end if;
```

### 2.2 从失效检测算法基本思想

从失效检测器算法现实比较简单,在主失效检测器调用时才启用。算法采用轮询策略,从失效检测器主要由进程$q$发送查询信息$mq_i$给进程$p$,$p$在收到该消息后发回应答消息$ma_i$,以表明自己处于正常状态;如果进程$q$没有收到应答消息$ma_i$则认为进程$p$失效。从失效

检测算法主要步骤如下所示：

```
For process q:
  send mq_i to p,
  if(upon receive ma_i from p and T_wait<T_ado)
      execute T_transiton,
  else
      execute S_transition
end if;
```

## 3 算法性能评估

### 3.1 服务质量指标

为了准确评价失效检测器的 QoS，Chen 等人提出了定量的 QoS 度量体系[7]。用 T 表示失效检测器认为进程处于正常状态，S 表示失效检测器认为进程处于失效状态，S-transition 表示失效检测器输出由 T 变为 S，T-transition 表示失效检测器的输出 S 变为 T。下面三个基本评价指标分别为：

检测时间($T_D$)：这一指标保障失效检测器输出的完整性要求，是指从进程发生失效的时刻到它开始被怀疑的时间，即到发生 S-transition 之间的时间。这一指标保障失效检测器输出的完整性要求。

错误间隔时间($T_{MR}$)：两次发生错误输出之间的时间间隔的度量，即连续两个 S-transition 之间的时间间隔，描述失效检测的准确度。

错误持续时间($T_M$)：检测器修正一次错误怀疑所需要的时间，即从 S-transition 到下一个 T-transition 之间的时间间隔，描述失效检测的准确度。

错误率($\lambda_M$)：指失效检测器产生怀疑的比率，是在一次运行中出现的频度就是一个常用的体现失效检测系统准确性的指标。

### 3.2 计算阈值 $T_D^U$

$T_D^U$ 是一个失效检测器与被检测对象之间的服务质量的基本评价指标之一，$T_D^U$ 是检测时间的上界，限定了检测速度，保证了失效检测的完整性级别，是一个十分重要的评价指标。

图 3 介绍了主从式失效检测器的执行过程。进程 $p$ 定时向进程 $q$ 发送心跳消息，进程 $q$ 分别在 $T_0$、$T_1$、$T_2$ 时刻收到前 3 个心跳消息，并预测下一个心跳消息将在 $T_3$ 时刻到达。而实际上进程 $p$ 在 $T_6$ 时刻失效，对于进程 $q$ 的时钟而言是 $T_5$ 时刻，则进程 $q$ 在 $T_3$ 时刻未收到应该到达的心跳消息，开始怀疑进程 $p$。则实际检测时间为：$T_3-T_5$，即 $T_3-T_4+\eta$，$\eta$ 为心跳消息的时间延迟。通常 $T_D^U=\Delta t_i+E(\eta)$，$E(\eta)$为平均心跳消息延迟。进程 $q$ 超过 $T_D^U$ 时间没有收到进程 $p$ 的心跳消息，则主动发送问询消息，查看进程 $p$ 是否失效。如果进程 $q$ 在发送问询消息后，超过 $T_7-T_3+T_D^U$ 时间没有收到应答消息，则确认进程 $p$ 失效。

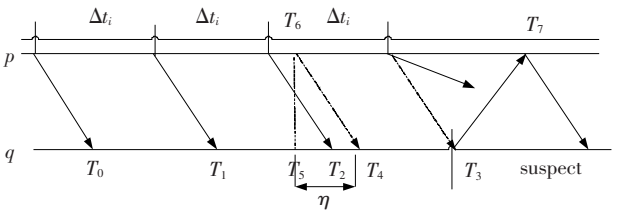


图 3 失效检测执行过程

### 3.3 算法证明

在 Web 服务组合失效检测中，失效检测器为应用程序提供失效检测等价于◇P 类的失效检测器，即失效检测器需达到强完整性和强准确性级别。通过以下分析说明本失效检测器等价于◇P 类失效检测器。

(1)强完整性。在任意一次运行中，每一个发生失效的进程最终都会被所有正确的进程永远判定为失效。

证明：假设在一个正确的进程 $q$ 检测进程 $p$。需要证明：如果 $p$ 发生失效，那么存在某一时刻对于任一的 $T>t$，失效检测的结果都将判定 $p$ 失效。在 $p$ 失效之后，将不会发出任何消息，也就是说，进程 $q$ 此后将收不到任何消息，因此，$E(T_D)$将不会发生变化。假设 $mp_i$ 是 $q$ 在时刻 $t_{last}$ 从 $p$ 收到的最后一个应答消息，则 $i$ 值将永远一个常数。此后，在任何查询时刻 $t_{current}>t_{last}$，都有 $\Delta T=T_{current}-f$，随着 $t_{current}$ 的增长，$\Delta T$ 将严格单调递增。因此，对于应用程序 $A$，在任意时刻 $t_{current}>t_{last}$ 查询，都会将 $p$ 判定为失效进程。

(2)强准确性。在任意一次运行中，在某个时刻 $t$ 之后，每个正确的进程都不会被错误认为发生失效。

证明：同上面的证明进行同样的假设，需要证明：在某个时刻 t 之后，应用程序 A 将不会错误地认为 $p$ 发生失效。因为此属性具有最终(eventual)性。假设 $mp_i$ 是 $p$ 发出的第 $i$ 个消息，发送时刻为 $t_s$，进程处理消息时间和消息的传递时间都有上界，分别记为 $t_p$ 和 $t_t$。因此，对于任意检测消息 $mp_j(j\geqslant i)$，其检测时间 $T_{Dt}\leqslant t_p+t_t$。那么，对于任意的查询时刻 $t_{current}>t_s$，$\Delta T$ 存在一个上界。只要 $T_D^U>T_{Dt}$，那么，对于任意的查询时刻 $t_{current}>t_s+t_p+t_t$，A 都不会怀疑进程 $p$。

### 3.4 性能分析

本文对主从式失效检测算法进行了编码方式仿真，实验操作系统为 Microsoft Windows XP，配置 1.6 GHz 和 1 GB 内存，在 Microsoft Visual Studio.NET 环境中，使用 C# 开发了对失效检测算法进行有效性验证的模拟程序。心跳周期 $\Delta t_i$ 设定为 1 000 ms，进行了算法准确性测试。

为了测试系统的准确性，与 Marin 等人提出适应网络状况变化的失效检测方法作比较测试，比较结果如图 4 所示。从图 4 可以看出，本失效检测较 Marin 有较低的误判率，当被检测对象数目超过 300 时候表现尤其明显，差距也越来越大，原因是这时候系统负载过高，系统错误数提高，误判率就随之上升。

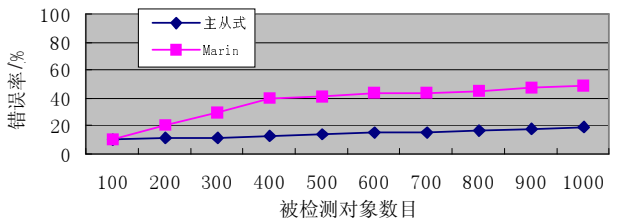


图 4 两种失效检测方法的比较

为了适应复杂的网络生态环境，增强了系统的可用性，本文设计了主从式失效检测算法，对相应的主从式失效检测器进行性能测试和分析。新算法减少了失效检测器在网络延时出现的误判情况，进一步适应应用环境，实验证明，在相同情况下，主从式失效检测算法准确性有明显提高，优化了失效检测器的检测服务质量，增强了系统的可用性，为流程执行时业务功能与服务质量的保障提供了一种有效的机制。

参考文献

[1] 付晓东，邹平.一种规则驱动的 Web 服务组合例外处理方法[J].计算机应用，2007，27(8)：1984－1990.

[2] WANG G J, WANG C Z, CHEN A, et al. Service level-management using qos monitoring, diagnostics, and adaptation for networked enterprise systems [C]//Proceedings of the 9th IEEE International EDOC Enterprise Computing Conference. Washington, DC: IEEE ComputerSociety, 2005: 239-250.

[3] ARDISSONO L, CONSOLE L, GOY A, et al. Enhancing Web services with diagnostic capabilities [C]//Proceedings of the 3 IEEE European Conference on Web Services. Washington: IEEE Computer Society, 2005:182-191.

[4] 徐新卫，周良，丁秋林. Web 服务失效处理的反射中间件技术应用与实现[J].系统工程与电子技术，2007，29(8)：1371－1376.

[5] BERTIER M, MARIN O, SENS P. Implementation and Performance Evaluation of an Adaptable Failure Detector [C]. Proc. IEEE Int'l Conf. Dependable Systems and Networks(DSN'2002), 2002:354-363.

[6] 陈宁江，魏峻，杨波，黄涛.Web 应用服务器的适应性失效检测[J].软件学报，2005，16(11)：1929－1938.

[7] CHEN W, TOUEG S, AGUILERA M K. On the Quality of Service of Failure Detectors. IEEE Tran on Computers, 2002,51(5):561-580.

(收稿日期：2009－03－06)