基于排队 Petri 网的服务系统性能建模与分析方法

顾 军[1),2)] 罗军舟[1)] 曹玖新[1)] 李 伟[1)]

[1)](东南大学计算机科学与工程学院 南京 211189)

[2)](中国矿业大学计算机科学与技术学院 江苏 徐州 221116)

摘 要 如何有效评估服务系统的性能表现是解决服务对动态网络环境适配性的关键.然而,基于互联网的服务系统的规模尺度和复杂程度不断提高,使得服务系统的性能建模和分析越来越困难.通过分析服务系统的执行过程,提出了一种基于排队 Petri 网的性能建模和分析方法.该方法将服务系统运行过程分为服务准备和服务提供两个阶段.服务准备阶段被建模为多层 Web 系统,刻画了浏览、选择、组合和注册 4 种行为的性能表现.服务提供阶段被建模为组合服务,重点讨论了交互关系、节点失效和恢复机制对组合服务执行性能的影响,并在此基础上建立组合服务在集中和分散两种执行方式下的性能模型.最后,采用 QPME 工具仿真和比较提出的模型在不同配置下的性能表现.结果表明,该方法在定量评估服务系统性能时具有一定的优越性.

关键词 排队 Petri 网;服务系统;组合服务;性能;失效

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.02435

Performance Modeling and Analysis of Service Systems Using Queueing Petri Nets

GU Jun[1),2)] LUO Jun-Zhou[1)] CAO Jiu-Xin[1)] LI Wei[1)]

[1)](*School of Computer Science & Engineering, Southeast University, Nanjing* 211189)

[2)](*School of Computer Science & Technology, China University of Mining and Technology, Xuzhou, Jiangsu* 221116)

Abstract How to evaluate the performance of service systems is the key for resolving the adaptation problem between services and dynamic network environment. Nevertheless, the increasing of size and complexity make it more difficult to model and analyze the performance of service systems. In this paper, an analytic model is proposed for evaluating performance of service systems using Queueing Petri Nets. The approach divides the execution process of service systems into two stages, service preparing and services providing. The service preparing stage is modeled as a multi-tier Web system, which can show the performance characteristics of four behaviors including browsing, selection, composition and register. The service providing stage is modeled as a composite service, which focuses on targeting the impact of interaction relations, node failure and node recovery on the system performance. Based on the interaction model and node model, two performance models of composite service with centralized and decentralized execution are given. Finally, QPME tool is used to simulate and compare the parameters under different configures. The simulation results show that this method can quantitatively analyze the performance of service systems.

Keywords queueing Petri nets; service systems; composite service; performance; failure

收稿日期:2011-06-23;最终修改稿收到日期:2011-11-02.本课题得到国家"九七三"重点基础研究发展规划项目基金(2010CB328104)、国家自然科学基金(60903161,60903162,61003257,61070161,61070158,61003311)、高校博士点专项基金(200802860031)、江苏省自然科学基金重点项目(BK2008030)、江苏省网络与信息安全重点实验室(BM2003201)和计算机网络与信息集成教育部重点实验室(93K-9)资助.**顾 军**,男,1977 年生,博士研究生,讲师,中国计算机学会(CCF)会员,主要研究方向为网络与服务计算.E-mail: jgu@seu.edu.cn.**罗军舟**,男,1960 年生,博士,教授,博士生导师,主要研究领域为下一代网络体系结构、协议工程、网络安全、网格与云计算、无线局域网.**曹玖新**,男,1967 年生,博士,教授,博士生导师,主要研究领域为网络安全、服务计算和数字版权管理.**李 伟**,男,1978 年生,博士,副教授,主要研究方向为下一代互联网、网络管理、服务计算.

1 引 言

面向服务的信息处理系统以服务为构造单元，强调软件的松耦合、可重用、易组合和动态优化[1]，能够有效解决互联网环境下的数据、资源和系统集成问题.随着面向服务应用的普及，服务系统开发、部署、运行和维护的网络环境逐渐从封闭、静态、可控走向开放、动态、难控.网络环境的动态性和不可预知性要求服务系统必须能够适应及处理各种变化，实现和增强服务对环境的适配能力，提升信息服务质量，即解决服务对环境的适配性问题.为此，首先必须对具有动态变化特性的服务系统的运行状态进行评估，通过建立服务系统的性能模型，预测环境变化对服务系统运行性能的影响，判定服务对环境能否适配，进而开展服务适配机制和方法的研究，增强服务的适配能力.

然而，面向服务的系统以可共享与可集成的自治网络资源为基础，往往包括众多协同工作的服务器或设备，加上用户需求的多样性和不确定性，使得服务系统不论在规模尺度还是复杂程度上都远远超过传统的软件系统，增加了系统性能建模和分析的难度[2].事实上，随着面向服务应用的普及，服务系统的边界、复杂性瓶颈和非功能性要求悄然发生了质的变化，规模的扩大化使得直接建模变得非常困难.

Dai 等人[3-4]的研究表明分析网格服务的性能和可靠性比针对整个网格更可行，也更有实际意义.我们认为服务系统的性能对外表现为向用户提供服务的能力，即服务的执行能力.为了更好地满足用户的需求，往往选择现有的一组服务按照一定的业务逻辑以组合服务的形式执行，而每类服务使用的资源规模和数量都是有限和可描述的.因此，只要能够从服务执行过程中抽象出具有共性的执行模型，就可以完成服务性能模型的建立和分析，并进而评估整个系统的性能.此外，互联网环境下的服务系统中，资源故障和组件失效的情况比较普遍.如果不考虑系统的失效行为，仅仅从纯性能的角度去分析会显得过于乐观.因此，本文将通过研究失效影响下的服务系统执行过程和性能表现，探索具有通用性的服务系统性能建模和分析方法.

为了预测各种性能指标，排队论、随机 Petri 网、随机进程代数等方法常被用作形式化建模和分析工具，建立反映系统行为和性质的数学模型[5].排队论(queuing theory)[6]，也称随机服务系统理论，是通过对服务对象到来及服务时间的统计研究，得出这些数量指标(等待时间、排队长度、忙期长短等)的统计规律，然后根据这些规律来改进服务系统的结构或重新组织被服务对象，使得服务系统既能满足服务对象的需要，又能使系统的费用最经济或某些指标最优.随机 Petri 网[7]以研究模型系统的组织结构和动态行为为目标，着眼于系统中可能发生的各种状态变化以及变化之间的关系.Petri 网作为一种图形化建模工具和一种具有丰富数学基础的形式化模型，可以广泛应用于描述和研究并发、异步和分布式特征的系统.因此，随机 Petri 网非常适合描述松耦合的分布式服务系统.排队 Petri 网(Queueing Petri Nets，QPN)[8]继承和发展了排队网模型和随机 Petri 网的优点，既能定量地建模资源的运行性能，也可以刻画多层系统之间的依赖关系.Kounev 等人[9-11]的研究工作表明排队 Petri 网具有很强的定量评价能力和行为描述能力，能很好地对分布式系统和网格系统进行建模和模拟，可用于系统的性能预测、能力规划和在线性能管理.因此，本文将在排队 Petri 网的系统模型框架上采用图形化的方式完成服务系统的性能建模.

本文第 2 节分析服务系统的运行过程，将其划分为服务准备和服务提供两个阶段，并给出组合服务的集中执行和分散执行两种方式；第 3 节给出服务系统的形式化描述，设计失效恢复影响下服务器节点的 QPN 模型，讨论满足时间约束和自动化要求的服务交互关系 Petri 网描述方法，并在此基础上对服务准备阶段、组合服务集中执行和组合服务分散执行 3 种情况下的性能进行建模；第 4 节给出仿真实验和数据分析；第 5 节介绍相关工作；最后，第 6 节总结全文并提出下一步工作.

2 服务系统运行过程分析

服务系统本质上是大量服务的集合，每个服务都驻留在一定的自治软硬件资源上，通过相关协议实现服务的发布、交互和组合，并协调多个服务提供者(providers)的行为，以便为消费者(consumers)提供增值服务.本文在面向服务架构(Service Oriented Architecture，SOA)[1]的基础上提出服务系统的一般性运行模式(如图 1 所示)，可以分为服务准备(service preparing)和服务提供(service providing)两个阶段.在服务准备阶段，服务门户(service portal)、

服务代理(Service Broker，SB)[12]和服务注册库(service registry)共同完成服务请求的接纳、服务的选择和组合、服务提供方案的确定、服务注册信息的登记和更新等. 为了提高处理能力，常常对这些功能部件的软硬件进行冗余处理. 在服务提供阶段，服务请求将按照已形成的组合服务方案，构建服务执行环境(service execution environment)，完成服务的调度、分配、绑定和执行，最后将服务结果反馈给消费者.

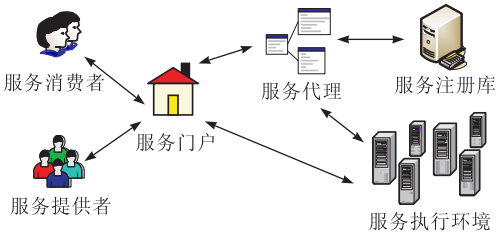


图 1　服务系统的一般性运行模式

服务准备阶段注册的服务信息、确定的服务方案都是信息服务提供阶段得以实施的前提和基础，而服务提供阶段的实际运行效果又是对服务准备阶段工作成效的评测与验证，两者组成了一个完整的服务系统，为服务消费者和服务提供者提供了交互的平台. 服务准备阶段的服务门户、服务代理和服务注册库等部件的性能会影响服务消费者和服务提供者的访问规模、对请求的应答时延和访问成功率等；而服务提供阶段的执行环境由分布的多个服务提供者的运行环境组成，单个服务的性能以及多个服务交互后的综合性能对系统性能影响很大. 服务准备阶段和服务提供阶段以服务为联系纽带，形成一种松耦合的结构，因此本文从服务准备和服务提供两个阶段对服务系统进行建模.

服务准备阶段的运行过程和系统所能提供的业务流程种类、每类业务包含的功能数量以及消费者的行为有关. 比如系统对外提供 n 项业务 $\{BP_1, BP_2, \cdots, BP_n\}$，其中业务 BP_i 包括 m_i 个功能. 每个功能的实现都可以由多个候选的服务提供者(service providers)负责，因此需要在服务注册库中为每个功能查找符合业务功能和非功能要求的服务 S，形成具有不同 QoS 属性的组合服务，并从中确定最终解决方案. 在实际执行过程中，系统并不需要为每个消费者都执行服务选择和组合. 此外，服务提供者也需要访问服务准备系统进行服务信息的注册和更新. 因此，我们将系统的处理行为分为 4 类：(1) 消费者只是一般性的浏览，并不提交服务请求；(2) 消费者提交服务请求，从系统推荐的组合服务方案中选择一个去执行；(3) 消费者提交服务请求，启动新组合服务方案的生成；(4) 服务提供者注册或更新服务信息. 每个处理行为都包括若干个操作，如图 2 所示，n_j 表示某个操作的执行次数. 比如，Browse 包括 (n_1+4) 个操作，Select 包括 (n_2+4) 个操作，Composite 包括 (m_i+5) 个操作，Register 包括 (n_3+4) 个操作. 虚线方框的操作属于服务提供阶段的处理工作.

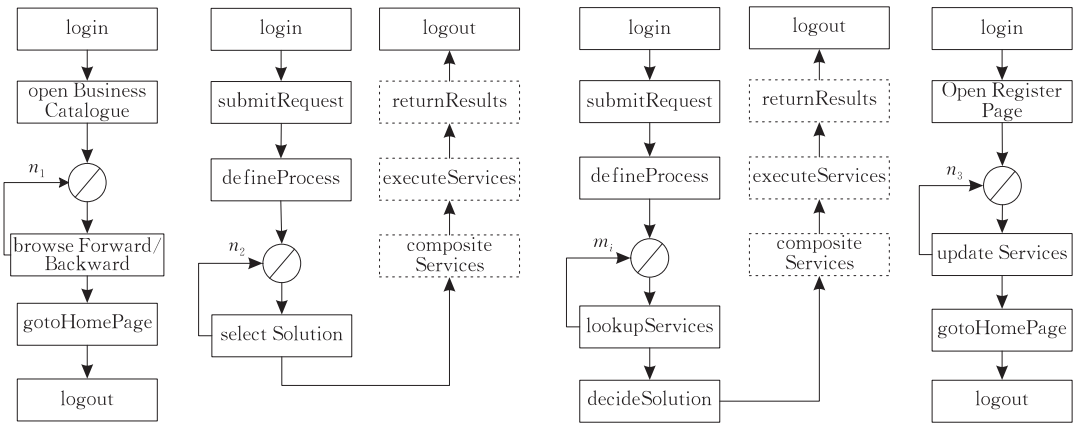


图 2　服务准备阶段的执行图

在服务提供阶段，单一服务或者组合服务在服务代理的调度下开始执行. 在服务执行环境 NE 中，子服务 S_i 驻留在节点 N_k 上执行. 服务与节点之间可以是一对一或者多对一的关系，即允许一个节点运行一个或者多个子服务. 在应用需求驱动下，服务环境 NE 中的部分服务提供者的节点和服务将被选择，并按服务组合方案形成组合服务协同工作以满足用户需求. 为此，本文引入服务覆盖网(service overlay network)的概念对组合服务执行环境进行描述，如图 3 所示. 服务链路表示节点之间的逻辑通道，可以是由多条物理链路组成.

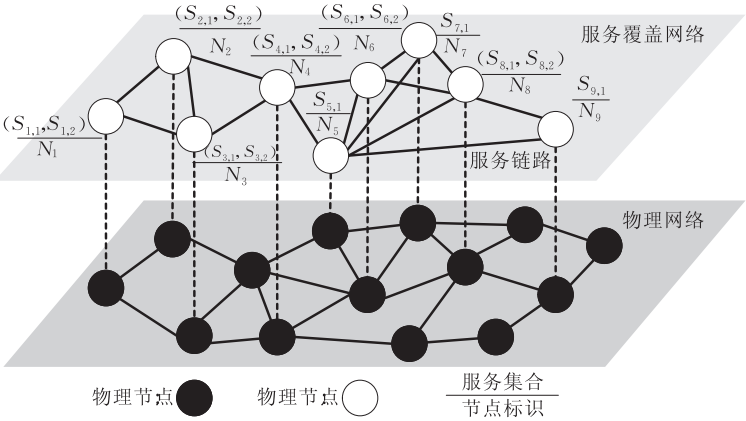


图 3 组合服务执行环境

多层次、具有高可扩展性的服务组合为用户和提供商提高运行效率提供了极大便利. 文献[13]的研究表明,引起系统复杂行为的主要原因不是元件的数量而是元件之间的交互,只要能保持系统元件之间交互的基本性质,那么即使对系统加以简化,系统的基本特性也不会改变. 为此,本文对组合服务的交互关系定义如下.

定义 1. 顺序关系(sequential). 服务 S_i 执行完毕后,S_j 才开始执行,记作 $S_i > S_j$.

定义 2. 重复关系(iterative). 服务 S_i 重复执行 k 次,记作 $\mu S_i(k)$.

定义 3. 并发关系(parallel). 服务 S_i 和 S_j 相互独立执行,记作 $S_i \mid S_j$. 并发后又有合并和不合并两种情况.

定义 4. 容错关系(fault-tolerant). 服务 S_i 和 S_j 提供相同的服务应用,在工作服务发生失效且不可恢复时,可以用备份服务来代替而使应用能够维持正常工作,记作 $S_i \mid S_j$.

定义 5. 条件关系(conditional). 前驱服务执行完毕后,按照一定的条件概率选择 S_i 和 S_j 中的一个开始执行,记作 $S_i(p)+S_j(1-p)$.

假设图 3 执行环境中有服务流程如图 4 所示. $S_{1.1}$ 称为($S_{2.1}+S_{3.2}$)的前驱服务,($S_{2.1}+S_{3.2}$)称为 $S_{1.1}$ 的后继服务(集). 起始服务 $S_{1.1}$ 执行后按照一定的条件选择 $S_{2.1}$ 和 $S_{3.2}$ 中的一个来执行. 其中,$S_{2.1}$ 被前驱服务(集)选中执行的概率为 p,$S_{3.2}$ 被选中的概率为$(1-p)$. 服务 $S_{4.1}$ 以重复方式连续执行 k 次后进入容错服务集($S_{5.1} \mid S_{6.1}$). $S_{5.1}$ 和 $S_{6.1}$ 同时启动后各自独立执行,只要有一个完成就可以调用后继服务(集). 并发执行的 $S_{7.1}$ 和 $S_{8.2}$ 必须先合并后再调用后继服务 $S_{1.2}$. 因此,业务流程可以形式化为

$$“S_{1.1} > (S_{2.1}(p)+S_{3.2}(1-p)) > \mu S_{4.1}(k) > (S_{5.1} \mid S_{6.1}) > (S_{7.1} \mid S_{8.2}) > S_{1.2}”.$$

组合服务的执行主要有两种方式[14-15]. 集中执行(centralized execution)方式需要控制中心完成所有服务的调用和数据传输,服务之间不直接交互,实现简单但存在性能瓶颈,只适合小规模的服务系统. 分散执行(decentralized execution)方式允许服务之间按照业务流程需要直接交互和传输数据,有利于减少通信成本、提高组合服务的吞吐量,适合构建大规模的服务系统. 图 4 的组合服务业务流程示例在不同执行方式下的结构如图 5 所示.

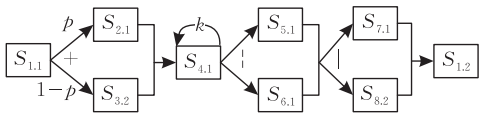


图 4 组合服务业务流程示例

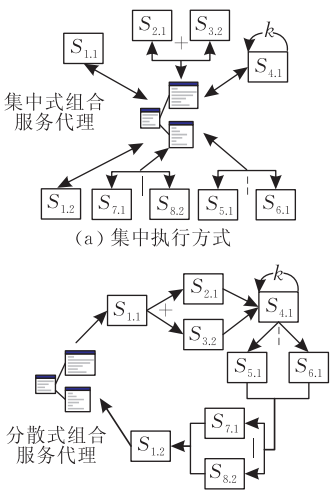


图 5 集中和分散的执行方式

3 服务系统的性能建模

3.1 服务系统的形式化描述

排队 Petri 网(QPN)结合了排队网络、有色 Petri

网(Colored Petri Nets,CPNs)、随机 Petri 网(Generalized Stochastic Petri Nets,GSPNs)的功能特性.在这里我们假定读者具有随机 Petri 网、排队网络和系统性能评价的基本知识.对此不熟悉的读者可以参阅文献[7].下面给出排队 Petri 网 QPN 的形式化定义,其余概念见文献[8-9,16].

定义 6.　QPN 是一个 8 元组(P,T,C,I^-,I^+,M_0,Q,W),其中

① P 是非空有限库所(places)集合$(p_1,p_2,\cdots,p_{|P|})$;

② T 是非空有限变迁(transitions)集合$(t_1,t_2,\cdots,t_{|T|})$;

③ $P\cap T=\varnothing$;

④ C 为颜色函数,$C:P\cup T\to\sum C$ 把每个库所 p 都映射到一个颜色集 $C(p)$,把每个变迁 t 都映射到一个颜色集 $C(t)$,也就是说 p 和 t 中的每个托肯(token)都属于颜色类型;

⑤ I^-,I^+ 分别是 $P\times T$ 上的后向和前向关联函数,使得对所有$(p,t)\in P\times T$ 有

$$I^-,(p,t),I^+,(p,t):C(t)\to C(p)_{MS};$$

⑥ M_0 称为 CPN 的初始标识,对于所有 $p\in P$ 都有 $M_0(p)\in C(p)_{MS}$.

⑦ $Q=(\tilde{Q}_1,\tilde{Q}_2,(q_1,q_2,\cdots,q_{|P|}))$,其中

$\tilde{Q}_1\subseteq P$ 表示时间队列库所;

$\tilde{Q}_2\subseteq P$ 表示瞬时队列库所,且 $\tilde{Q}_1\cap\tilde{Q}_2=\varnothing$;

如果库所 p_i 是队列库所,那么 q_i 表示具有 $C(p_i)$中所有颜色的队列;如果库所 p_i 是普通库所,那么 q_i 等价于关键字“null”.

⑧ $W=(\widetilde{W}_1,\widetilde{W}_2,(w_1,w_2,\cdots,w_{|T|}))$,其中

$\widetilde{W}_1\subseteq T$ 表示时间变迁;

$\widetilde{W}_2\subseteq T$ 表示瞬时变迁,且 $T=\widetilde{W}_1\cup\widetilde{W}_2,\widetilde{W}_1\cap\widetilde{W}_2=\varnothing$;

$w_i\in[C(t_i)\mapsto\Re^+]$使得对于所有的 $c\in C(t_i)$有:如果 t_i是时间变迁,那么 $w_i(c)$描述了与颜色 c 有关的变迁触发时延的概率分布函数;如果 t_i是瞬时变迁,那么 $w_i(c)$描述了与颜色 c 有关的变迁触发频率的权值.

由此可见,排队 Petri 网模型的结构元素包括队列库所(queueing place)、普通库所(ordinary place)、时间变迁(timed transition)、瞬时变迁(immediate transition)、弧(arc)和颜色托肯(color token).队列库所用于描述具有随机服务特征的业务处理行为,包括队列(queue)和贮存库(depository)两部分;普通库所用于描述可能的系统局部状态(条件或状况);时间变迁用于描述具有时延特征的改变系统状态的事件;瞬时变迁用于描述系统状态改变的事件;弧用于规定局部状态和事件之间的关系;颜色托肯用于描述不同类型的服务实例.

从软硬件配置的角度看,服务准备阶段可以建模为一个多层系统(multi-tier system),包括负载均衡器(load balancer machine)、应用服务器集群(application server clusters)和数据库服务器(database servers).其中应用服务器处理 Web 业务逻辑和服务组合,数据库服务器处理 Web 数据访问和服务注册库处理.服务准备阶段的形式化定义如下.

定义 7.　服务准备阶段是一个四元组,即$MS=(MQ,V,P,SH)$,其中

① $MQ=\{MQ_1,MQ_2,\cdots,MQ_M\}$是多层服务器集,其中 MQ_i是第 i 层服务器集,可以包含多个并发运行的服务器,每个服务器都可以建模为一个队列库所;

② $V=\{V_1,V_2,\cdots,V_M\}$是服务器访问速率集,其中 V_i是第 i 层服务器集的访问比率(visit ratio),即到达 MQ_i的平均访问数目,可用于计算队列的到达率 λ;

③ $P=\{p_1,p_2,\cdots,p_M\}$是多层服务器之间的变迁概率集,其中由 MQ_i返回 MQ_{i-1}的概率是 p_i,由 MQ_i到 MQ_{i+1}的概率是$(1-p_i)$;

④ $SH=\{SH_1,SH_2,\cdots,SH_M\}$是每层服务器集内部的调度策略(schedule displine),如随机调度法、负载均衡法、优先级调度法等.

在服务提供阶段,由于服务系统的分布性和异构性,失效的发生不可避免.所谓服务失效是服务节点软硬件运行行为对消费者需求的偏离,是服务系统的一个动态特征.服务节点发生失效后将会停止工作直到失效被恢复,即当服务器节点遇到大量的突发请求、用户误操作、CPU 资源短时衰竭、网络瞬时堵塞、临时中断、恶意的入侵攻击或网络错误等可修复故障时,通过恢复策略解决服务遇到失效而被迫终止的问题.因此,服务提供阶段的形式化定义如下.

定义 8.　服务提供阶段是一个七元组,即$DS=(N,S,CL,IS,NS,RN,RL)$:

① N,S,CL 分别是有限节点集、可用服务集和服务链路集,其中服务链路是对跨地域网络基础设施的虚拟化,屏蔽了实际的网络拓扑和通信细节;

② $IS:N\times N\to\{>,\mu,|,\dagger,+\}$是节点间的交

互关系函数，其中$>$，μ，$|$，$\mathop{|}$，$+$分别表示顺序、重复、并发、容错和条件关系；

③ $NS:N\rightarrow S^*$ 是节点的可用服务函数，$NS(N_i)=S_i=\{S_{i,1},S_{i,2},\cdots,S_{i,m}\}$表示节点 N_i的可用服务集，其中 $S_{i,j}$是节点N_i的第 j 个可用服务；

④ $RN:N\rightarrow(0,1)$是节点成功运行不失效的概率，$RL:CL\rightarrow(0,1)$是链路成功通信不失效的概率，根据实际情况，节点和链路的成功概率不为 1.

在不确定的、复杂的各种外部因素的影响下，服务系统不可能永远稳定、可靠地运行. 分析服务系统的性能先要建立系统的性能模型，而建立服务系统的性能模型可以从以下几个方面入手：首先要建立系统节点的性能模型，它描述了理想无失效发生时节点的结构和性能情况；二是对可能出现失效的节点资源建立失效模型和恢复模型，它用来描述节点失效的过程及失效后的反应. 这其中引起节点失效的原因很多，可能是节点自身设计的问题、管理或操作中的问题，也可能是由于外来恶性攻击而造成的；三是将前面的两者结合建立考虑失效恢复的节点性能模型；最后建立包含多种交互关系和不同执行方式的系统性能模型，它描述了在一个特定的业务流程结构中某些节点失效后对整个系统所带来的影响，根据性能模型计算各种性能参数，就可综合分析一个服务系统的性能状态.

3.2 建模服务准备阶段

服务准备阶段的 QPN 图形化模型如图 6 所示. 其中，黑色方框图代表瞬时变迁，圆圈表示普通库所，带有竖线的圆圈表示队列库所.

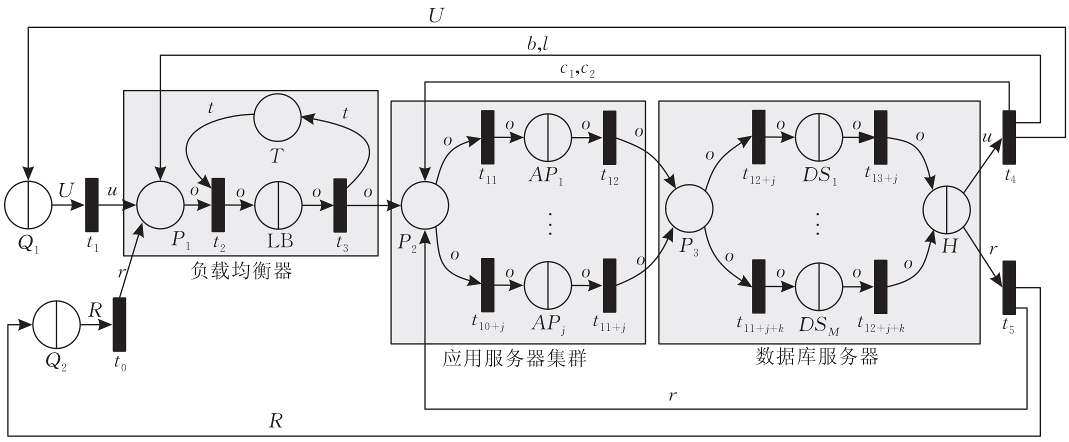


图 6 服务准备阶段的 QPN 表示

为了方便讨论建模方法，本文假设服务准备阶段只提供两项业务$\{BP_1,BP_2\}$，包括的功能个数分别为 m_1和 m_2. 为此，服务准备阶段执行时包含 5 类处理行为，且每类都包含若干操作. 其中，托肯“B”表示 Browse 行为，由 n_1个操作“b”组成；托肯“L”表示 Select 行为，由 n_2个操作“l”组成；托肯“C_1”表示业务 BP_1 的 Composite 行为，由 m_1 个操作“c_1”组成；托肯“C_2”表示业务 BP_2 的 Composite 行为，由 m_2个操作“c_2”组成；托肯“R”表示 Register 行为，由 n_3个操作“r”组成. 为了简化模型表示的复杂，对符号含义规定如下：

(1) 符号“U”描述托肯“B”，“L”，“C_1”或“C_2”，表示消费者的一个行为.

(2) 符号“u”描述托肯“b”，“l”，“c_1”或“c_2”，表示消费者行为的一个操作.

(3) 符号“o”描述托肯“r”，“b”，“l”，“c_1”或“c_2”，表示消费者或提供者的一个操作.

服务准备阶段模型的执行过程如下：

(1) 库所 Q_1($G/M/\infty/IS$ 队列)建模消费者请求的发生，托肯“U”(“B”，“L”，“C_1”或“C_2”)的初始数量反映了并发请求的规模，间隔时间满足指数分布. 库所 Q_2($G/M/\infty/IS$ 队列)建模提供者请求的发生，托肯“R”的初始数量反映了请求的规模，间隔时间满足指数分布.

(2) 消费者发出请求，触发变迁 t_1销毁 Q_1中的一个托肯“U”，同时在库所 P_1中创建一个托肯“u”，表明消费者行为的第 1 个操作准备执行. 提供者发出请求，触发变迁 t_0销毁 Q_2中的一个托肯“R”，同时在库所 P_1中创建一个托肯“r”，表明提供者行为的第一个操作准备执行.

(3) 库所 LB($G/M/1/PS$ 队列)建模负载均衡器，普通库所 T 建模负载均衡器线程池. 当请求“o”被 LB 调度时，将从 T 中分配一个托肯“t”；当请求“o”成功调度后将把托肯“t”返回给 T. 如果 T 中没有可用托肯“t”时，请求“o”只能在 P_1中等待新的线程托肯“t”的出现. T 中托肯“t”的初始数量决定了

可用线程的数目.

(4) 变迁 t_{11}，t_{12}，…，t_{10+j} 的触发概率相同，因此被 LB 成功调度的请求“o”将等概率地从库所 P_2 移入 j 个应用服务器 CPU 库所中的一个，如 AP_i($G/M/1/PS$ 队列).

(5) 托肯“o”从 AP_i 进入库所 P_3，接着被等概率地移入 k 个数据库服务器 CPU 库所中的一个，如 DS_i($G/M/1/PS$ 队列).

(6) 库所 DS_i 处理完的托肯“o”将被移入数据库磁盘库所 H($G/M/1/PS$ 队列). 完成磁盘访问后，托肯“o”的处理有 5 种情况：

(6.1) 变迁 t_4 将托肯“b”或“l”移入库所 P_1，开始下一个 Web 访问操作.

(6.2) 变迁 t_4 将托肯“c_1”或“c_2”移入库所 P_2，开始下一个服务注册库访问操作.

(6.3) 消费者行为的所有操作都完成后，从 Q_1 中移出的托肯“U”返回到 Q_1 中.

(6.4) 变迁 t_5 将托肯 r 移入库所 P_2，开始下一个服务注册操作.

(6.5) 提供者 Register 行为的所有操作完成后，从 Q_2 中移出的托肯“R”返回到 Q_2 中.

图 6 中除 t_4 和 t_5 之外的变迁描述如表 1 所示. 符号“$A\{x\}\rightarrow B\{y\}$”表示模型中变迁的触发模式(firing mode)，托肯“x”在一次变迁中由库所 A 迁出后以“y”的形式存放在库所 B 中. In 和 Out 表示变迁的输入和输出库所. 每个变迁的触发权值(firing weight)都相同，设为 1.

表 1　服务准备阶段中 t_4 和 t_5 以外的变迁触发模式

变迁	触发模式	变迁	触发模式
t_0	1：$In\{R\}\rightarrow Out\{r\}$	t_3	1：$LB\{b\}\rightarrow P_2\{b\}+T\{t\}$
t_1	1：$In\{B\}\rightarrow Out\{b\}$		2：$LB\{l\}\rightarrow P_2\{l\}+T\{t\}$
	2：$In\{L\}\rightarrow Out\{l\}$		3：$LB\{c_1\}\rightarrow P_2\{c_1\}+T\{t\}$
	3：$In\{C_1\}\rightarrow Out\{c_1\}$		4：$LB\{c_2\}\rightarrow P_2\{c_2\}+T\{t\}$
	4：$In\{C_2\}\rightarrow Out\{c_2\}$		5：$LB\{r\}\rightarrow P_2\{r\}+T\{t\}$
t_2	1：$P_1\{b\}+T\{t\}\rightarrow LB\{b\}$	t_{11}，t_{12}，…，t_{11+N}，t_{12+N}，t_{13+N}，…，t_{12+N+M}	1：$In\{B\}\rightarrow Out\{b\}$
	2：$P_1\{l\}+T\{t\}\rightarrow LB\{l\}$		2：$In\{L\}\rightarrow Out\{l\}$
	3：$P_1\{c_1\}+T\{t\}\rightarrow LB\{c_1\}$		3：$In\{C_1\}\rightarrow Out\{c_1\}$
	4：$P_1\{c_2\}+T\{t\}\rightarrow LB\{c_2\}$		4：$In\{C_2\}\rightarrow Out\{c_2\}$
	5：$P_1\{r\}+T\{t\}\rightarrow LB\{r\}$		5：$In\{r\}\rightarrow Out\{r\}$

变迁 t_4 和 t_5 的变迁模式如表 2 所示. 为了对各类处理行为进行准确评估，必须合理设置这些变迁模式的触发权值，以保证只有当所有操作完成后才能结束这类处理. 对于变迁 t_5 来讲，根据图 2(d)的描述，Register 包括(n_3+4)个操作，即每完成(n_3+4)个操作就完成一个处理行为. 为此，可以设置变迁 t_5 的模式 1 和 2 的触发权值分别为$(n_3+3)/(n_3+4)$和$1/(n_3+4)$. 这样，当有(n_3+4)个托肯“r”进入库所 H 后，变迁 t_5 的模式 1 平均触发(n_3+3)次，而模式 2 平均触发 1 次. 虽然不能保证其触发顺序，但资源消耗和队列行为方面与真实系统非常接近.

表 2　服务准备阶段中 t_4 和 t_5 的变迁触发模式

变迁	触发模式	过程描述
t_4	1：$H\{b\}\rightarrow P_1\{b\}$	完成一个 Browse 操作，但是整个处理行为没有结束.
	2：$H\{b\}\rightarrow Q_1\{B\}$	完成一个 Browse 操作，整个处理行为结束.
	3：$H\{l\}\rightarrow P_1\{l\}$	完成一个 Select 操作，但是整个处理行为没有结束.
	4：$H\{l\}\rightarrow Q_1\{L\}$	完成一个 Select 操作，整个处理行为结束.
	5：$H\{c_1\}\rightarrow P_2\{c_1\}$	完成一个 BP_1 的 Composite 操作，但是整个处理行为没有结束.
	6：$H\{c_1\}\rightarrow Q_1\{C_1\}$	完成一个 BP_1 的 Composite 操作，整个处理行为结束.
	7：$H\{c_2\}\rightarrow P_2\{c_2\}$	完成一个 BP_2 的 Composite 操作，但是整个处理行为没有结束.
	8：$H\{c_2\}\rightarrow Q_1\{C_2\}$	完成一个 BP_2 的 Composite 操作，整个处理行为结束.
t_5	1：$H\{r\}\rightarrow P_2\{r\}$	完成一个 Register 操作，但是整个处理行为没有结束.
	2：$H\{r\}\rightarrow Q_2\{R\}$	完成一个 Register 操作，整个处理行为结束.

变迁 t_4 的情况要复杂一些. 根据图 2 的描述，Browse 包括(n_1+4)个操作，Select 包括(n_2+4)个操作，BP_1 的 Composite 包括(m_1+5)个操作，BP_2 的 Composite 包括(m_2+5)个操作. 令变迁 t_4 的 8 个模式的触发权值为 $w(i)$，其中 $i=1,2,\cdots,8$. 权值的设置步骤如下：

(1) 令 $w(1)+w(2)=w(3)+w(4)=w(5)+w(6)=w(7)+w(8)$,从而保证 4 个消费者处理行为的发生概率相等.

(2) 令 $mw=\max\{n_1+4,n_2+4,m_1+5,m_2+5\}$,即取操作数目的最大值.

(3) 按照以下公式计算各个模式的权值.

$$w(1)=mw\times\frac{n_1+3}{n_1+4} \tag{1}$$

$$w(2)=mw\times\frac{1}{n_1+4} \tag{2}$$

$$w(3)=mw\times\frac{n_2+3}{n_2+4} \tag{3}$$

$$w(4)=mw\times\frac{1}{n_2+4} \tag{4}$$

$$w(5)=mw\times\frac{m_1+4}{m_1+5} \tag{5}$$

$$w(6)=mw\times\frac{1}{m_1+5} \tag{6}$$

$$w(7)=mw\times\frac{m_2+4}{m_2+5} \tag{7}$$

$$w(8)=mw\times\frac{1}{m_2+5} \tag{8}$$

3.3 建模服务提供阶段

3.3.1 考虑失效恢复的服务节点

根据失效对服务节点影响的不同,可以分为通信失效和计算失效两大类. 通信失效使得节点停止接收新的任务请求,同时停止将完成的任务输出,等到失效修复后将恢复到正常的任务输入和输出状态. 计算失效使得服务任务在节点的生命周期分为有效执行和失效恢复 2 个阶段. 在有效执行阶段,服务任务持续进入节点,完成后进入下一个执行环节. 在失效恢复阶段,节点启动服务恢复程序进行处理. 当然,失效恢复具有一定的概率,即失效可以恢复时,恢复后的服务从失效断点继续执行,直到遇到不可恢复的失效或者服务 S_i 执行完毕;反之,失效不可恢复,S_i 被迫终止,在该节点的任务执行宣告失败[15]. 因此,在建模服务系统的性能模型时需要考虑服务节点的失效和恢复对服务性能的影响.

为了更好地描述服务节点的性能表现,本文的分析满足以下假设:

(1) 用户任务请求的到达率与节点的状态是独立的且服从泊松分布,这个假设在已有的关于失效对性能影响的研究工作中被广泛使用[17-18];

(2) 通信失效和计算失效独立发生,失效发生率服从泊松分布;

(3) 每个服务节点都存在一个有限长度的队列来处理所有的服务请求;

(4) 所有节点被调用后立即执行,并且其无失效执行时间服从指数分布;

(5) 由于失效恢复主要通过执行恢复程序完成,所以假设失效恢复时间服从指数分布;

(6) 因通信失效而丢失的服务任务将会继续发送给服务节点.

服务提供阶段的服务节点的 QPN 模型如图 7 所示. 队列库所 $F_1(G/M/\infty/IS)$代表通信失效的发生,按照一定失效发生率 λ_{f_1} 输出托肯"f_1",失效间隔时间服从 λ_{f_1} 的指数分布. 通信失效发生后进入修复阶段,用队列库所 $R(G/M/1/PS)$建模失效恢复时间. NP 建模服务节点处理环节,包含一个 $G/M/n/PS$ 队列,可以模拟具有 n 个 CPU 的服务节点工作负载. 队列库所 $F_2(G/M/\infty/IS)$代表计算失效的发生,按照一定失效发生率 λ_{f_2} 输出托肯"f_2",失效间隔时间服从 λ_{f_2} 的指数分布. 发生计算失效的服务将以托肯"s'"或"w'"的形式回送到库所 NP 中,启动恢复程序,进入计算失效恢复处理阶段.

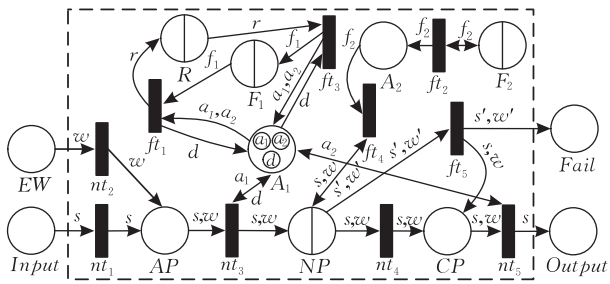


图 7 服务节点的 QPN 表示

由于服务的共享性与重用性,服务提供者的服务节点上往往运行着多类服务任务,这些额外负载会影响服务任务的性能,而失效对这些额外负载的影响也是一样的. 代表前驱服务(集)的库所 $Input$ 向服务节点 NP 输入任务托肯"s",库所 EW 产生服务任务"s"之外的工作负载"w". 库所 A_1 中包含 3 个控制托肯("a_1","a_2"和"d"),其取值在 0 和 1 之间切换,0 表示托肯不存在,禁止变迁触发,1 表示托肯存在,允许变迁触发. 在未发生通信失效,即变迁 ft_1 未触发时,库所 A_1 中只包含托肯"a_1"和"a_2"各一个,其中"a_1"用于控制服务节点的输入,即变迁 nt_3 的触发,"a_2"用于控制服务节点的输出,即变迁 nt_5 的触发.

如果新任务到达时没有发生通信失效,那么变迁 nt_3 触发模式 1 和模式 2 将新到的托肯"s"和"w"移入队列库所 NP 中,任务完成后将输出到下一个

环节，即触发变迁 nt_5 的模式 1 和模式 2. 如果发生通信失效，即变迁 ft_1 触发，将托肯“a_1”和“a_2”从库所 A_1 中移出，同时在 A_1 中创建托肯“d”，在队列库所 R 中创建托肯“r”. 托肯“a_1”的移出使得变迁 nt_3 的触发模式 1 和模式 2 被禁止；托肯“a_2”的移出使得变迁 nt_5 的触发模式 1 和模式 2 被禁止；托肯“d”的创建使得变迁 nt_3 的模式 3 和模式 4 被触发，销毁新到的托肯“s”和“w”；托肯“r”的创建使得通信环节进入修复阶段，等到修复成功，即变迁 ft_3 触发，将会在库所 A_1 中创建托肯“a_1”和“a_2”各一个，在队列库所 F_1 中创建托肯“f_1”，同时销毁库所 A_1 中的托肯“d”，服务节点进入新的失效间隔阶段.

计算失效和通信失效独立发生，且对服务任务的影响是个体性的，即计算失效的一次发生只会使当前从库所 NP 中输出的任务托肯“s”或“w”转变为托肯“s'”或“w'”，然后回送到库所 NP 中. 如果失效恢复成功，那么托肯“s'”移入库所 CP 后直接输出到库所 $Output$. 如果失效恢复不成功，那么托肯“s'”将被移入库所 $Fail$ 中. 如果服务有冗余容错处理环节，那么托肯“s'”将被送入备份工作部件中，否则被销毁，即服务失败终止. 变迁 ft_5 属于自由选择冲突模型，选择哪一个变迁模式实施并不依赖于库所中的标识，而取决于变迁模式的触发权值(firing weight)，即发生的概率. 令变迁 ft_5 的 4 个变迁模式的触发权值为 $w(i)$，$i=1,2,3,4$. 服务任务“s'”的失效恢复成功概率为 φ_s，服务任务“w'”的失效恢复成功概率为 φ_w，那么它们之间的关系如下：

$$\varphi_s=\frac{w(1)}{w(1)+w(2)} \tag{9}$$

$$\varphi_w=\frac{w(3)}{w(3)+w(4)} \tag{10}$$

图 7 中的变迁描述如表 3 所示. 符号“$A\{x\}\rightarrow\{\}$”表示“x”在变迁中由库所 A 迁出后不留在任何库所中，而是被销毁(destroy).

表 3　服务节点中的变迁触发模式

变迁	触发模式	变迁	触发模式
nt_1	1：$Input\{s\}\rightarrow AP\{s\}$	nt_2	1：$EW\{w\}\rightarrow AP\{w\}$
nt_3	1：$AP\{s\}+A_1\{a_1\}\rightarrow NP\{s\}+A_1\{a_1\}$ 2：$AP\{w\}+A_1\{a_1\}\rightarrow NP\{w\}+A_1\{a_1\}$ 3：$AP\{s\}+A_1\{d\}\rightarrow A_1\{d\}$ 4：$AP\{w\}+A_1\{d\}\rightarrow A_1\{d\}$	nt_4	1：$NP\{s\}\rightarrow CP\{s\}$ 2：$NP\{w\}\rightarrow CP\{w\}$
		nt_5	1：$CP\{s\}+A_1\{a_2\}\rightarrow Output\{s\}+A_1\{a_2\}$ 2：$CP\{w\}+A_1\{a_2\}\rightarrow A_1\{a_2\}$
ft_1	1：$F_1\{f_1\}+A_1\{a_1\}+A_1\{a_2\}\rightarrow R\{r\}+A_1\{d\}$		
ft_2	1：$F_2\{f_2\}\rightarrow F_2\{f_2\}+A_2\{f_2\}$		
ft_3	1：$R\{r\}+A_1\{d\}\rightarrow F_1\{f_1\}+A_1\{a_1\}+A_1\{a_2\}$		
ft_4	1：$A_2\{f_2\}+NP\{s\}\rightarrow NP\{s'\}$ 2：$A_2\{f_2\}+NP\{w\}\rightarrow NP\{w'\}$	ft_5	1：$NP\{s'\}\rightarrow CP\{s\}$ 2：$NP\{s'\}\rightarrow Fail\{s'\}$ 3：$NP\{w'\}\rightarrow CP\{w\}$ 4：$NP\{w'\}\rightarrow Fail\{w'\}$

3.3.2　服务交互的 Petri 网表示

通过 Petri 网的变迁触发机制的设计可以描述服务组合中存在的多种交互关系，如图 8 所示. 然而这只是对交互关系的静态描述，并没有考虑实际执行过程的时间约束、失效影响和自动化要求. 顺序和条件交互关系下的任务执行是独立和一次性的，即只要业务流程中规定的前驱服务(集)能够完成，随后就会启动这两类交互，其触发条件是确定的，因此在系统建模中的描述方法和图 8 一致.

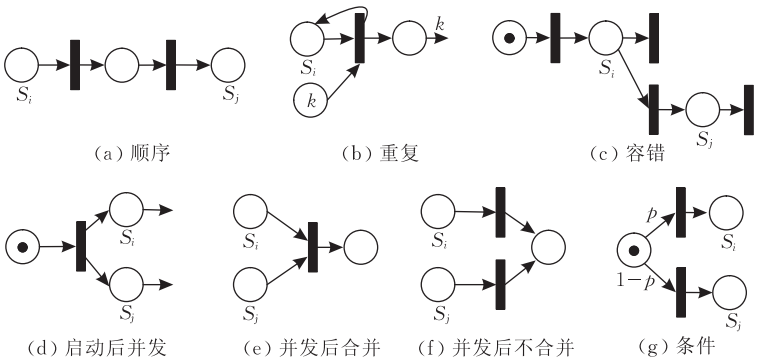
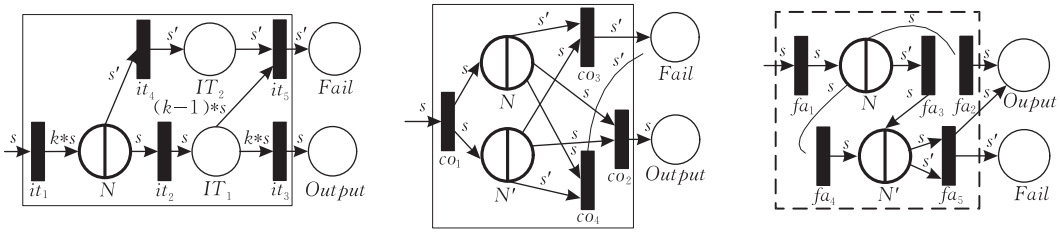


图 8　服务交互关系的 Petri 网表示

重复关系需要预设重复次数 k，确保在重复执行次数满足后才进入到后继服务(集)，并且自动设置下一次重复次数. 从运行过程分析，重复交互可以看作具有 k 个操作的处理过程. 但是，从失效影响来

看,对于通信失效,按照假设(6)可以通过任务的重发来解决;对于计算失效,一个操作的失效就意味着整个重复关系的失败,如图 9(a)所示. 当有一个服务任务"s'"恢复不成功时,将会触发变迁 it,销毁 $(k-1)$个托肯"s",节点 N 的内部细节如图 7 所示.



(a) 重复关系节点的QPN模型 (b) 并发关系节点的QPN模型 (c) 容错节点的QPN模型

图 9 重复、并发和容错关系节点的 QPN 模型

容错关系是在对失效服务采取修复措施外,增加冗余备份资源以提高系统可用性最常用的策略之一. 这样可以使系统在出现故障的时候仍能维持正常功能,当工作部件发生不可恢复的故障后,可以用备份部件来代替有故障的部件而使系统能够维持正常工作. 按备份部件所处状态的不同,冗余备份系统主要有 3 种形式:冷备份、温备份和热备份. 冷备份是指备用部件处于完全不工作状态,并假设它的失效率为零. 温备份系统指备用部件与主部件处于完全相同的工作状态下,但备用部件相对于主部件处于轻载荷工作状态,其失效率较主部件的小,而在热备份系统中,两者的失效率相同[19]. 在实际的服务系统中可根据服务提供者管理策略的不同设置不同的冗余容错结构,如图 9(c)所示. 主节点 N 中没有失效或者失效修复成功的服务任务将通过变迁 fu_1 输出到下一个环节 $Output$,由于通信失效而被主节点 N 丢失的服务任务将通过变迁 fu_3 输送到备用节点 N'中,由于计算失效而失效且修复不成功的服务任务将通过变迁 fu_2输送到备用节点 N'中. 备用节点 N'中的服务任务按照图 7 的方式工作. 容错关系对服务任务的响应时间影响不大,但可以提高服务可用性.

此外,在开放的网络环境中,服务交互关系既和业务逻辑有关,也和服务驻留的节点位置有关. 当服务与节点之间是一对一关系,即一个节点只运行一类服务时,服务交互和节点交互是一致的,描述也比较简单. 当服务与节点之间是多对一关系,即一个节点运行多类服务时,同样的两个节点在业务流程的不同执行阶段可能按照不同的方式进行交互. 图 3 的组合服务执行环境中有 9 个服务节点提供了 15 类服务,其中有 3 个节点属于单类服务队列,6 个节点属于两类服务队列. 为此,我们用"s_j"作为节点 N_i的第 j 个服务 $S_{i,j}$的托肯符号,以便描述节点队列的处理行为.

3.3.3 组合服务集中执行的性能模型

业务流程"$S_{1.1}>(S_{2.1}(p)+S_{3.2}(1-p))>\mu S_{4.1}(k)>(S_{5.1}\mid S_{6.1})>(S_{7.1}\mid S_{8.2})>S_{1.2}$"虽然包含多种交互关系,但可以分解为 6 个顺序执行的服务(集). 用托肯"S"表示这个组合服务,"z_1"表示 $S_{1.1}$,"z_2"表示 $S_{2.1}(p)+S_{3.2}(1-p)$,"z_3"表示 $\mu S_{4.1}(k)$,"z_4"表示$(S_{5.1}\mid S_{6.1})$,"z_5"表示$(S_{7.1}\mid S_{8.2})$,"z_6"表示 $S_{1.2}$,"z_7"表示处理组合服务的执行结果. 当组合服务"S"以集中方式执行时,服务代理 SB 将从"z_1"开始依次调用服务(集)"z_i". 当托肯"z_i"回到 SB 时,SB 将开始调用"z_{i+1}"表示的服务(集),直到"z_7"完成为止. 为了简化模型表示,规定符号"z"表示托肯"z_1","z_2",…,"z_6",或"z_7". 组合服务"S"集中执行场景对应的 QPN 模型如图 10 所示.

执行过程如下:

(1)队列库所 $CQ(G/M/\infty/IS)$建模组合服务执行请求的到达,托肯"S"的到达速率反映了服务请求的频度. 队列库所 $CB(G/M/1/PS)$建模服务代理,完成服务调用和最终结果处理. 变迁 ct_1触发后将销毁库所 CS 中的一个托肯"S",同时在 CB 中创建托肯"z_1",准备调用"z_1"表示的服务 $S_{1.1}$.

(2)变迁 ct_2触发后在库所 SE 中创建托肯"s_1",调用节点 N_1的服务 $S_{1.1}$. 变迁 ct_1销毁 N_1输出的托肯"s_1",在库所 CB 中创建一个托肯"z_2",准备调用"z_2"表示的服务集 $S_{2.1}(p)+S_{3.2}(1-p)$.

(3)服务 $S_{2.1}$和 $S_{3.2}$的选择概率可以通过变迁 cd_1和 cd_2的触发权值来实现. 变迁 cd_1的触发将在 N_2中创建托肯"s_1"调用服务 $S_{2.1}$,变迁 cd_2的触发将在 N_3中创建托肯"s_2"调用服务 $S_{3.2}$. 无论是哪个服务被调用,完成后将触发变迁 ct_1在库所 CB 中创建一个托肯"z_3",准备调用"z_3"表示的服务集 $\mu S_{4.1}(k)$.

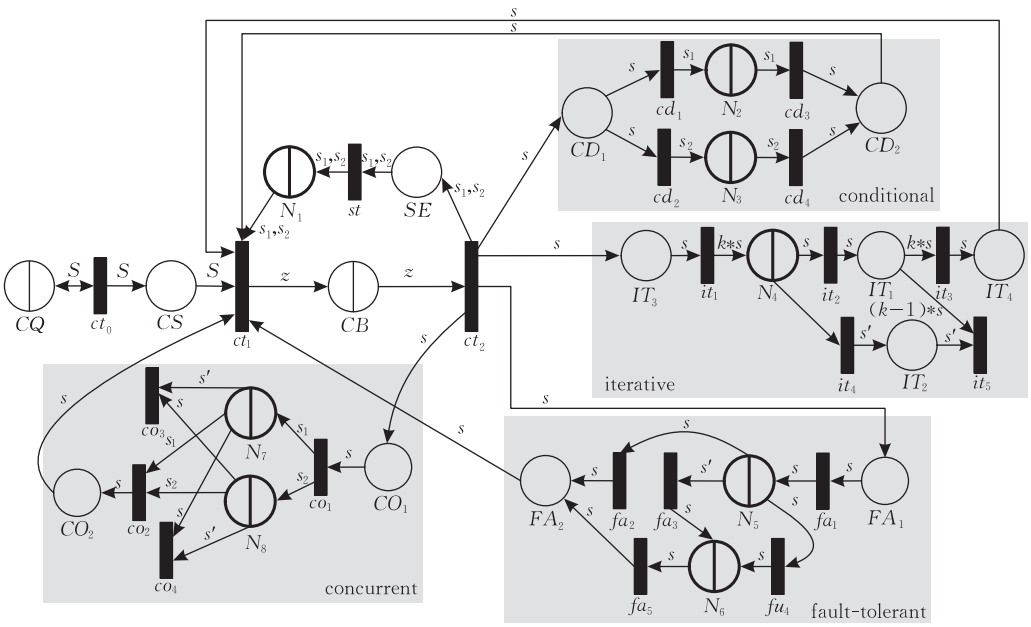


图 10　组合服务集中执行的 QPN 表示

(4) 变迁 ct_2 在库所 IT_3 中创建托肯“s”，调用节点 N_4 的服务 $S_{4.1}$. 变迁 it_1 的触发在队列库所 N_4 中生成 k 个任务托肯“s”，变迁 it_3 的触发将销毁库所 N_4 中的 k 个任务托肯“s”，同时输出托肯“s”到库所 IT_4，变迁 ct_1 得到托肯“s”后将在库所 CB 中创建一个托肯“z_4”，准备调用后继服务集 $(S_{5.1} \mid S_{6.1})$.

(5) 变迁 ct_2 在库所 FA_1 中创建托肯“s”，调用节点 N_5 的服务 $S_{5.1}$. 库所 FA_1 中成功执行的任务托肯“s”输出到库所 FA_2 中，触发变迁 ct_1 在库所 CB 中创建一个托肯“z_5”，准备调用“z_5”表示的服务集 $(S_{7.1} \mid S_{8.2})$. 库所 FA_1 中不能成功执行的任务托肯“s”将通过变迁 fa_3 输出到容错备用节点 N_6 中. 如果能够成功执行，就通过变迁 fa_4 输出到库所 FA_2 中，否则就销毁，表明任务失败终止.

(6) 变迁 ct_2 在库所 CO_1 中创建托肯“s”后，变迁 co_1 同时调用 N_7 中的服务 $S_{7.1}$ 和 N_8 中的服务 $S_{8.2}$. 等到两个服务都完成后触发变迁 co_2 在库所 CO_2 中创建托肯“s”. 变迁 ct_1 触发后在库所 CB 中创建一个托肯“z_6”，准备调用“z_6”表示的服务集 $S_{1.2}$.

(7) 变迁 ct_2 触发后在库所 SE 中创建托肯“s_2”，调用节点 N_1 的服务 $S_{1.2}$. 完成后，变迁 ct_1 销毁 N_1 输出托肯“s_2”，在库所 CB 中创建托肯“z_7”，准备处理组合服务的最终结果.

(8) 库所 CB 输出托肯“z_7”后触发变迁 ct_2 将托肯“z_7”销毁，表明组合服务执行完成.

图 10 中的变迁描述如表 4 所示.

表 4　集中执行模型的变迁触发模式

变迁	触发模式	变迁	触发模式	变迁	触发模式
ct_0	1：$CQ\{S\}\rightarrow CS\{S\}$	ct_1	1：$CS\{S\}\rightarrow CB\{z_1\}$ 2：$N_1\{s_1\}\rightarrow CB\{z_2\}$ 3：$CD_2\{s\}\rightarrow CB\{z_3\}$ 4：$IT_4\{s\}\rightarrow CB\{z_4\}$ 5：$FA_2\{s\}\rightarrow CB\{z_5\}$ 6：$CO_2\{s\}\rightarrow CB\{z_6\}$ 7：$N_1\{s_2\}\rightarrow CB\{z_7\}$	fa_1	1：$FA_1\{s\}\rightarrow N_5\{s\}$
ct_2	1：$CB\{z_1\}\rightarrow SE\{s_1\}$ 2：$CB\{z_2\}\rightarrow CD_1\{s\}$ 3：$CB\{z_3\}\rightarrow IT_3\{s\}$ 4：$CB\{z_4\}\rightarrow FA_1\{s\}$ 5：$CB\{z_5\}\rightarrow CO_1\{s\}$ 6：$CB\{z_6\}\rightarrow SE\{s_2\}$ 7：$CB\{z_7\}\rightarrow\{\}$			fa_2	1：$N_5\{s\}\rightarrow FA_2\{s\}$
				fa_3	1：$N_5\{s'\}\rightarrow N_6\{s\}$
				fa_4	1：$N_5\{s\}\rightarrow N_6\{s\}$
				fa_5	1：$N_6\{s\}\rightarrow FA_2\{s\}$
				co_1	1：$CO_1\{s\}\rightarrow N_7\{s_1\}+N_8\{s_2\}$
				co_2	1：$N_7\{s_1\}+N_8\{s_2\}\rightarrow CO_2\{s\}$
		it_1	1：$IT_3\{s\}\rightarrow N_4\{k\times s\}$	co_3	1：$N_7\{s'\}+N_8\{s\}\rightarrow\{\}$
cd_1	1：$CD_1\{s\}\rightarrow N_2\{s_1\}$	it_2	1：$N_4\{s\}\rightarrow IT_1\{s\}$	co_4	1：$N_7\{s\}+N_8\{s'\}\rightarrow\{\}$
cd_2	1：$CD_1\{s\}\rightarrow N_3\{s_2\}$	it_3	1：$IT_1\{k\times s\}\rightarrow IT_4\{s\}$	st	1：$SE\{s_1\}\rightarrow N_1\{s_1\}$ 2：$SE\{s_2\}\rightarrow N_1\{s_2\}$
cd_3	1：$N_2\{s_1\}\rightarrow CD_2\{s\}$	it_4	1：$N_4\{s'\}\rightarrow IT_2\{s'\}$		
cd_4	1：$N_3\{s_2\}\rightarrow CD_2\{s\}$	it_5	1：$IT_1\{(k-1)\times s\}+IT_2\{s'\}\rightarrow\{\}$		

3.3.4　组合服务分散执行的性能模型

组合服务“$S_{1.1}>(S_{2.1}(p)+S_{3.2}(1-p))>\mu S_{4.1}(k)>(S_{5.1}\mid S_{6.1})>(S_{7.1}\mid S_{8.2})>S_{1.2}$”分散执行的 QPN 模型如图 11 所示. 队列库所 $DQ(G/M/\infty/IS)$ 建模组合服务执行请求“S”的到达，队列库所 $DB(G/M/1/PS)$ 建模组合服务代理. 变迁 dt_2 触

发模式 1 后在库所 SE 中创建托肯“s_1”后开始调用第一个服务 $S_{1.1}$，随后不再干涉组合服务的执行过程. 服务之间按照预定的执行方案直接交互，由最后一个完成的服务 $S_{1.2}$ 通过变迁 dt_9 将最终结果“S'”返回给服务代理 DB，处理后通过变迁 dt_2 的触发模式 2 将托肯“S'”销毁，表明整个组合服务执行完成.

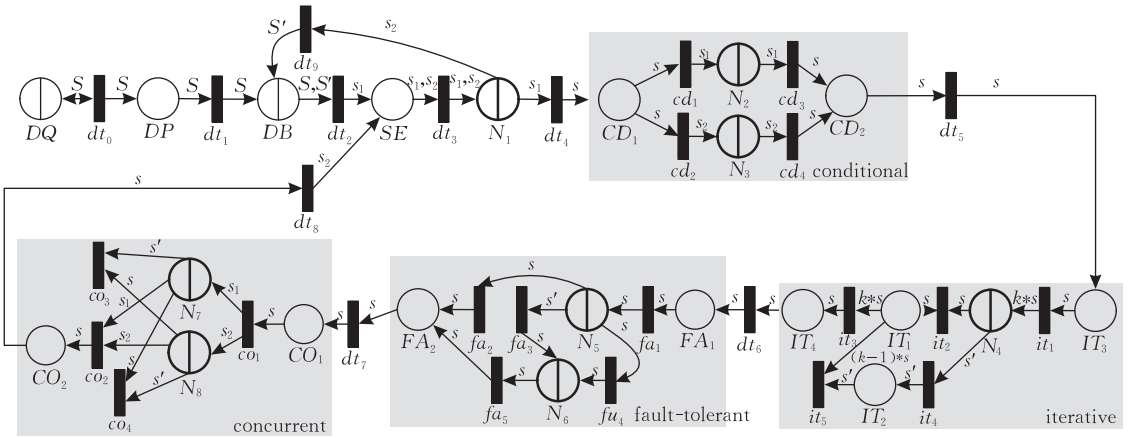


图 11 组合服务分散执行的 QPN 表示

4 实验与分析

4.1 仿真工具 QPME

按照上述建模思路可以建立起基于排队 Petri 网的服务系统性能模型，但是在实际应用中仍存在不少局限性，主要是状态空间爆炸问题. 排队 Petri 网模型的状态空间大小会随模型中库所、变迁、颜色托肯等建模元素数目的增长呈指数级增长，导致所谓状态空间爆炸问题或维数灾难问题，不仅关系到数值方法求解的可行性，而且影响到仿真方法分析的精确度，严重限制着此类建模机制在现实中的应用.

QPME(Queueing Petri net Modeling Environment)软件包[16,20]是一款基于排队 Petri 网的性能建模工具，具有友好的图形界面，包括 QPN 编辑器(QPE)和仿真器(SimQPN)两部分. SimQPN 能够规避 QPN 的状态空间爆炸问题，支持构建层次化模型 HQPNs(Hierarchically-Combined QPNs)，有助于理解系统各部件之间的交互关系，还可以更简洁地描述系统. Kounev 等人[10]采用 SimQPN 对大规模复杂分布式组件系统进行了建模，结果证明其对 QPN 模型有等价的形式化表达能力，能够提供准确和稳定的性能度量值，可用于分析具有现实规模和复杂性的 QPN 模型. Kounev 等人[9-10]使用 QPME 对 J2EE 软硬件平台性能测量基准程序 SPECjAppServer2004 进行建模和仿真，实验结果和实际系统测量值非常接近，总体的结果误差在 ±5% 以内，表明 QPME 能够很好反映系统的性能变化，预测系统性能度量值.

因此，本文在 Kounev 等人的研究成果基础上建立服务系统的 QPN 性能模型，重点关注服务系统中的部件能力、交互关系、执行方式等的图形化描述；采用 QPME 作为仿真工具，既可以利用 SimQPN 工具的形式化建模能力为性能模型的形式化验证提供保证，减少不必要的繁琐证明，也可以通过实验分析阐述所提出的服务系统性能模型的可行性.

4.2 仿真服务准备阶段

服务准备阶段的实验仿真场景如图 6 所示，其中负载均衡器包含 1 个 CPU，线程池 T 的数量为 50，应用服务器包含 5 个 CPU，数据库服务器包含 2 个 CPU 和 1 个磁盘系统. 相应的队列库所针对服务准备阶段的 5 类操作行为的参数 p_1 设置如表 5 所示.

表 5 服务准备阶段的队列库所的参数配置

处理行为	Q_1	Q_2	LB	AP_i	DS_i	H
Browse	3.2E-4	—	0.1	0.005	0.016	0.06
Select	3.2E-4	—	0.1	0.010	0.050	0.050
Register	—	1.75E-4	0.1	0.025	0.045	0.052
Composite1	3.2E-4	—	0.1	0.012	0.032	0.08
Composite2	3.2E-4	—	0.1	0.015	0.035	0.092

为了分析服务准备阶段的性能表现，设计了两个具有不同访问规模的应用场景，如表 6 所示.

表 6 服务准备阶段的实验场景

处理行为	托肯	操作数	场景 1	场景 2
Browse	B	$n_1=10$	25	200
Select	L	$n_2=7$	10	70
Register	R	$n_3=4$	10	50
Composite1	C_1	$m_1=6$	15	85
Composite2	C_2	$m_2=8$	18	68

根据图 2 的描述，Browse 行为包含 14 个操作，即当托肯“b”在库所 H 中出现 14 次时，变迁 t_4 的模式 1 平均触发 13 次，而模式 2 平均触发 1 次；Select 行为包含 11 个操作，即当托肯“l”在库所 H 中出现 11 次时，变迁 t_4 的模式 3 平均触发 10 次，而模式 4 平均触发 1 次；Composite 1 行为包含 11 个操作，即当托肯“c_1”在库所 H 中出现 11 次时，变迁 t_4 的模式 5 平均触发 10 次，而模式 6 平均触发 1 次；Composite 2 行为包含 13 个操作，即当托肯“c_2”在库所 H 中出现 13 次时，变迁 t_4 的模式 7 平均触发 12 次，而模式 8 平均触发 1 次. 为此，变迁 t_4 的 8 个触发模式的权值的一种可能的设置方案如下：

$$w(1)=13,\ w(2)=1,\ w(3)=12.73,$$
$$w(4)=1.27,\ w(5)=12.73,\ w(6)=1.27,$$
$$w(7)=12.92,\ w(8)=1.08.$$

Register 行为包含 8 个操作，即当托肯“r”在库所 H 中出现 8 次时，变迁 t_5 的模式 1 平均触发 7 次，而模式 2 平均触发 1 次. 为此，变迁 t_5 的 2 个触发模式的权值的一种可能的设置方案如下：

$$w(1)=7,\ w(2)=1.$$

服务准备阶段的性能度量属性包括稳态时每种行为的吞吐量（throughput）X_i 和响应时间（response time）T_i，服务器利用率（service utilization）U_i 包括负载均衡器利用率 U_{LB}、应用服务器利用率 U_{AS} 以及数据库服务器利用率 U_{DB}，结果数据的置信区间（c. i.）为 95%. 仿真结果如表 7 所示.

表 7　服务准备阶段的仿真结果

X_i	场景 1	场景 2	T_i	场景 1(95% c. i.)	场景 2(95% c. i.)	U_i	场景 1/%	场景 2/%
X_B	21.12	194.35	T_B	16997.491(+/−30.800)	107408.501(+/−142.771)	U_{LB}	28.6	37.1
X_L	7.69	67.37	T_L	10404.323(+/−23.478)	80178.498(+/−163.445)	U_{AS}	63.5	76.9
X_{C_1}	11.63	81.83	T_{C_1}	10795.878(+/−21.523)	80749.312(+/−160.700)	U_{DB}	52.5	61.7
X_{C_2}	14.13	65.80	T_{C_2}	11418.780(+/−21.938)	93412.132(+/−207.386)			
X_R	5.75	45.56	T_R	7735.456(+/−17.099)	58583.748(+/−119.557)			

分析实验结果可知，性能模型可以描述不同访问规模的服务系统，且当系统的访问规模增大时，系统中各种操作行为的吞吐量 X_i 随之增加，使得系统中各类服务器的利用率 U_i 也得到很大提高. 但是负载量的增加，也使得系统的处理效率下降，操作行为的响应时间 T_i 变得更长.

4.3　仿真服务提供节点

仿真实验分析失效率、恢复时间、额外负载和失效恢复成功率对服务节点模型的性能影响，度量属性包括稳态时任务“s”的丢失率 L（loss ratio）、吞吐率 x（throughput rate）、平均响应时间 τ（mean response time）和服务节点利用率 U（service node utilization），结果数据的置信区间（c. i.）为 95%. 丢失率 L 指任务托肯“s”到达时因为失效而没有被接受的任务数量占到达任务数目的比率. 实验仿真场景如图 12 所示. 采用开放式队列模型模拟一定速率 λ 到达的服务请求，队列库所 $Q(G/M/\infty/IS)$ 模拟稳定的任务产生源，队列库所 $Q'(G/M/\infty/IS)$ 模拟稳定的额外负载产生源. 库所 SC 用来跟踪和描述服务节点的执行状况，托肯“c_1”统计丢失率 L 和吞吐率 x，托肯“c_2”统计服务任务的平均响应时间 τ.

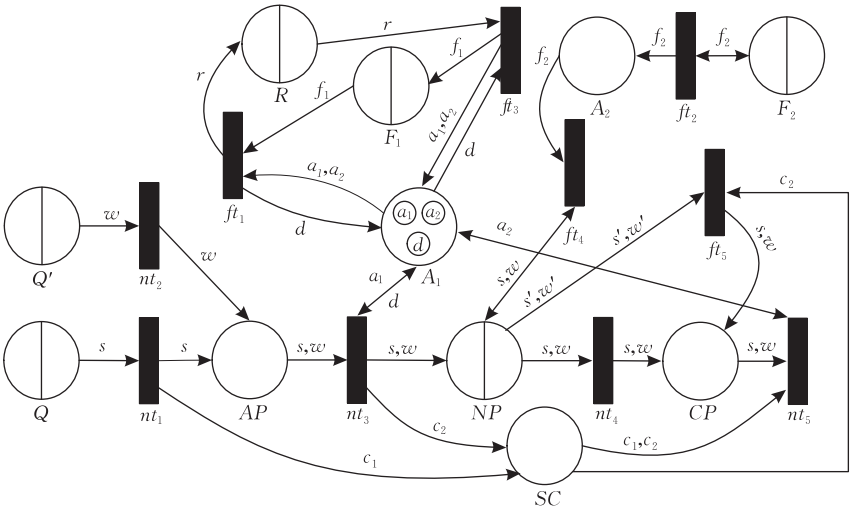


图 12　服务节点的仿真

对于可恢复的服务,其可靠性的主要参数是平均失效间隔时间(Mean Time Between Fail,MTBF)和平均恢复时间(Mean Time To Recovery,MTTR).服务节点模型 N 中队列库所的初始设置如表 8 所示,其中任务到达服从泊松分布,服务处理时间和失效恢复时间服从 λ 的指数分布,服务失效恢复的成功率为 99.9%.

表 8 服务节点模型 N 的队列库所属性表

库所	托肯	初始值	λ	库所	托肯	初始值	λ
Q	$\{s\}$	$\{1\}$	0.0005	Q'	$\{w\}$	$\{1\}$	0.0005
F_1	$\{f_1\}$	$\{1\}$	1.2E−7	F_2	$\{f_2\}$	$\{1\}$	1.4E−8
R	$\{r\}$	$\{0\}$	1.5E−5	NP	$\{s,w,s',w'\}$	$\{0,0,0,0\}$	1.5E−3, 1.5E−3, 1.2E−4, 1.2E−4

实验 1 分析通信失效率对服务节点模型的性能影响,仿真结果如图 13 所示.分析实验结果可知,随着通信失效率降低,即通信失效的 MTBF 变长,服务节点的吞吐率逐渐增加,任务丢失率逐渐下降,从而增加了服务节点的利用率,也使得节点负载变得更高.当通信失效间隔时间 MTBF 超过 1.0E+06 ms 后,吞吐率、丢失率和利用率的变化趋于稳定.服务的平均响应时间的变化分为 3 个阶段:(1)初期时呈现下降趋势,分析其原因主要是通信失效率的下降使得通信失效恢复的执行频率降低,减少了服务响应过程中由于通信失效恢复而产生的等待时间.因为通信失效恢复时间相对节点的处理时间要长,所以实际的服务平均响应时间反而下降了.(2)当通信的 MTBF 超过 1.0E+06 ms 后,响应时间开始了增加趋势,分析其原因主要是失效恢复时间在平均响应时间中所占的比重逐渐下降,而由于节点负载增加造成的等待时间增加的影响增强,导致服务平均响应时间开始增加.(3)当通信的 MTBF 超过 2.0E+09 ms 后,服务平均响应时间下降并保持稳定,分析其原因是此时的通信 MTBF 时间接近初始设置的仿真实验时间范围,使得通信失效发生的概率几乎为零,其平均响应时间只与相对稳定的节点负载和处理能力有关.

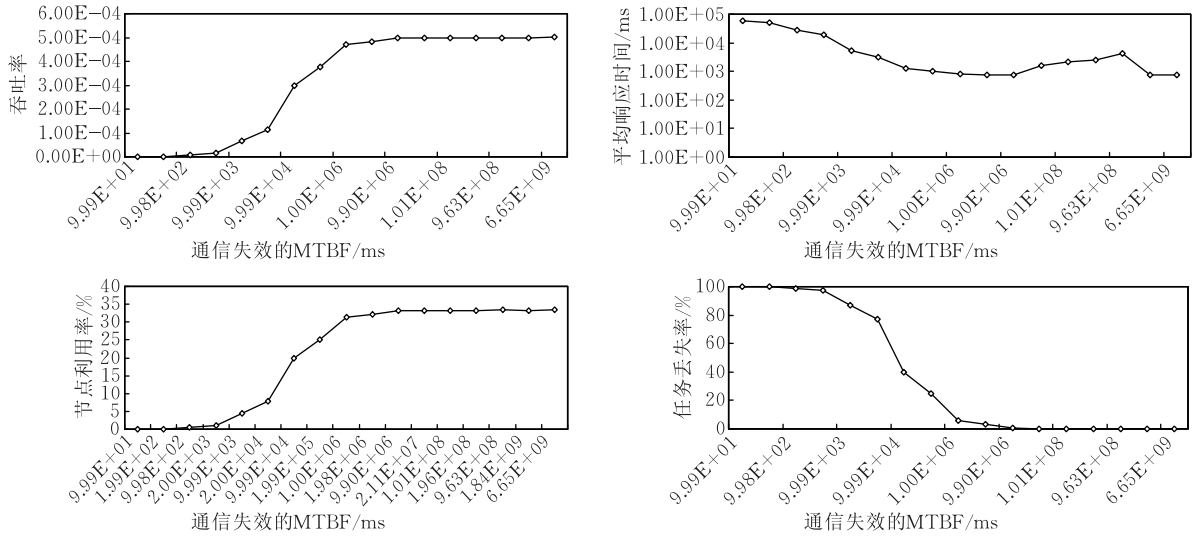


图 13 实验 1 的仿真结果

实验 2 分析通信失效恢复时间对服务节点模型的性能影响,仿真结果如图 14 所示. 分析实验结果可知,通信失效恢复时间的延长,使得服务节点的有效执行时间变短,节点吞吐率和利用率下降,任务丢失率上升.这个变化趋势在初期是比较平缓的,当通信失效恢复时间 MTTR 超过 1.2E+06 ms 后,变化幅度逐渐增大.服务平均响应时间的变化分为 3 个阶段:(1)初始时趋于平稳,主要原因是通信失效恢复时间的变长虽然增加了任务的平均等待时间,但是也在一定程度上减少了进入节点的任务的数量,降低了节点负载,减少了任务在节点中的平均逗留时间,因此综合影响后的服务平均响应时间变化平缓.(2)当通信 MTTR 超过 1.2E+06 ms 后,较长的失效恢复时间的影响力变大,导致服务平均响应时间变长.(3)当通信 MTTR 超过 5E+09 ms 后,由于接近初始设置的仿真实验时间范围,使得通信失效发生的概率几乎为零,其平均响应时间只与相对稳定的节点负载和处理能力有关.比较实验 1 和实验 2 的仿真结果,可以发现其对服务节点模型的性能分析结果是一致的.

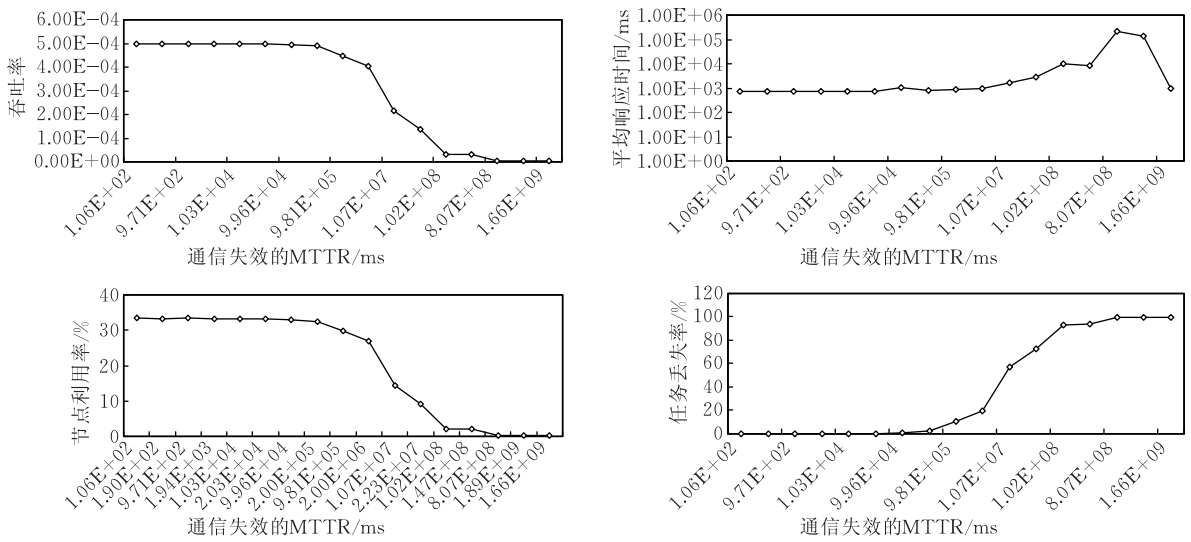


图 14　实验 2 的仿真结果

实验 3 分析计算失效率对服务节点模型的性能影响，仿真结果如图 15 所示. 分析实验结果可知，随着计算失效率降低，计算失效的 MTBF 变长. 初始时，计算失效率很高，对应的 MTBF 很大，服务节点的吞吐率保持在一个较低的水平，平均响应时间很大，节点处于满负载工作状态，任务丢失率接近 40%. 当 MTBF 上升到 5.0E+3 ms 左右时，节点吞吐率快速上升，节点利用率和任务丢失率快速下降，平均响应时间快速下降，随后又保持相对平稳的变化趋势. 分析其原因，主要是当失效率较高时，服务节点需要执行大量的恢复处理程序，使得节点处于满负荷工作状态，因此所有性能指标都处于低谷；当时失效率降低到一个门限值时，节点负载减轻，各个性能指标都得到恢复；但是当失效发生间隔 MTBF 超过了平均响应时间时，其影响变小，因此性能指标的变化又趋于平稳.

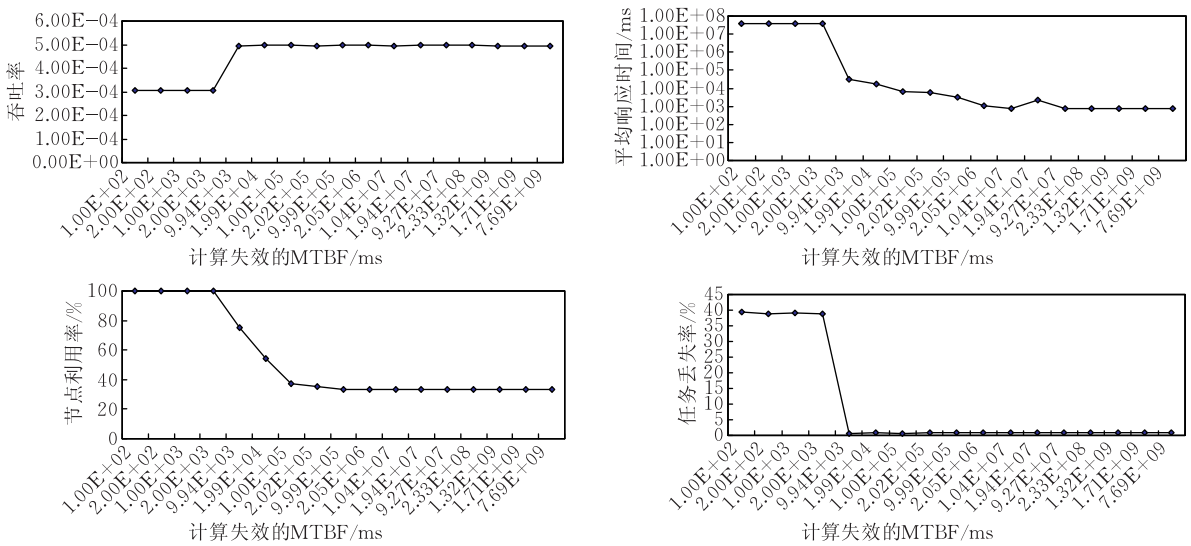


图 15　实验 3 的仿真结果

实验 4 分析计算失效恢复时间对服务节点模型的性能影响，仿真结果如图 16 所示. 分析实验结果可知，随着计算失效恢复时间 MTTR 变长，服务任务在节点中的逗留时间变长，因此增加了节点的利用率和负载，造成了平均响应时间的增加. 由于计算失效和计算失效恢复之间是独立发生的，因此失效恢复时间 MTTR 对吞吐率和丢失率的影响是间接的，其表现为在一定范围内的上下波动.

实验 5 分析额外负载“w”对服务节点模型性能的影响，仿真结果如图 17 所示. 分析实验结果可知，只有当额外负载“w”到达率很高时，节点利用率和负载处于较高水平，使得任务“s”的吞吐率很低，平均响应时间很长，任务丢失率很高. 但是当“w”到达时间间隔增加 500 ms 以后，即“s”到达时间间隔(2000 ms)

的 1/4,额外负载“w”对“s”的影响就变得很小了.

实验 6 分析计算失效恢复成功率对服务节点模型的性能影响,仿真结果如图 18 所示. 分析实验结果可知,随着计算失效恢复成功率的不断提高,任务丢失率逐渐下降. 而其它性能参数基本不受其影响,只是在一个较小范围内上下波动.

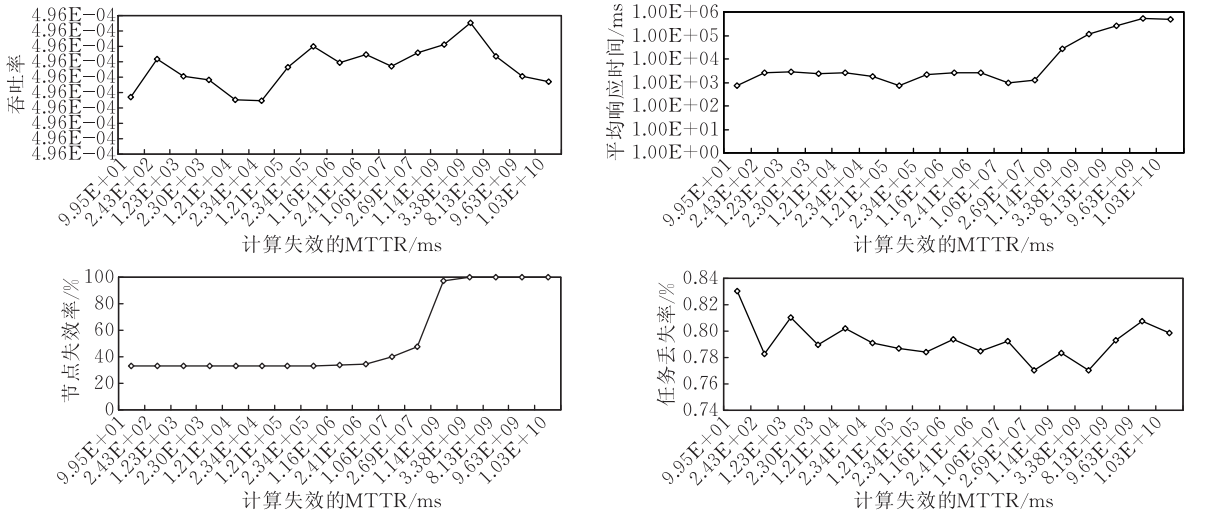


图 16 实验 4 的仿真结果

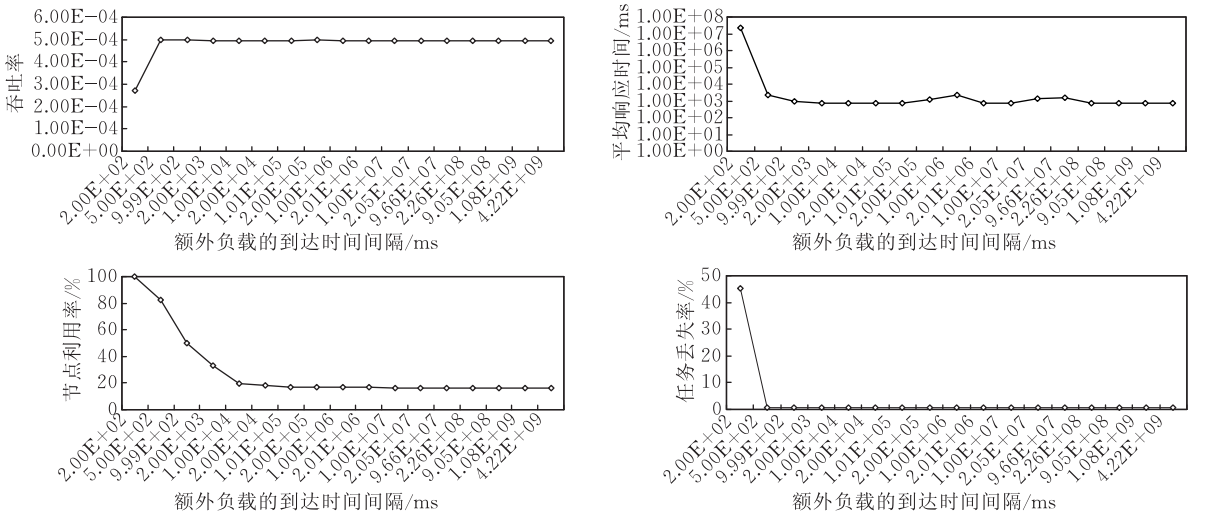


图 17 实验 5 的仿真结果

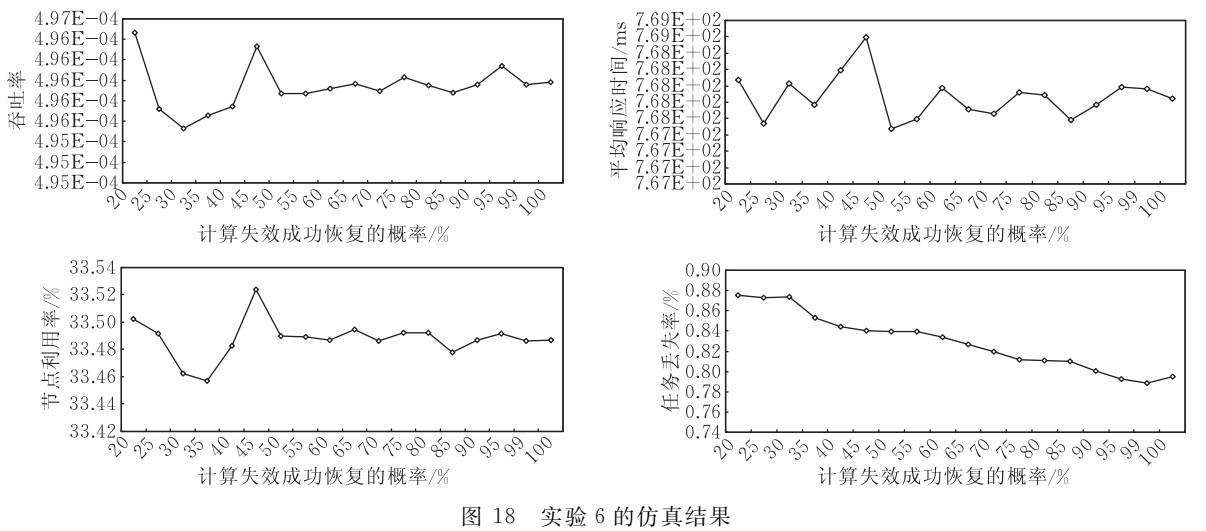


图 18 实验 6 的仿真结果

4.4　仿真服务交互关系

不同交互关系下的性能与可靠性指标的度量方式是不同的.根据性质的不同,可分为可加性度量、可乘性度量、最小性度量和最大性度量 4 类.比如,顺序关系下的响应时间具有可加性,可靠性具有可乘性;并发关系下的响应时间具有最大性,可靠性具有可乘性;重复关系可以看成是连续多个顺序结构;条件关系下的选择概率对性能和可靠性指标的影响很大;而容错关系下的不同结构使得度量属性的计算更加复杂.因此,当组合服务的规模和交互分支增加时,很难直接推导出通用的组合服务性能属性和可靠性的度量公式,更不用说量化考虑失效和恢复影响的服务性能属性.

为了分析不同交互关系下的服务节点模型的性能表现,引入一个新的服务节点模型 N',其中的队列库所的初始设置如表 9 所示,其中任务到达服从泊松分布,服务处理时间和失效恢复时间服从 λ 的指数分布,服务失效恢复的成功概率为 99.9%.条件交互下 N 和 N'的选择概率分别为 70%和 30%.容错关系下的 N'具有和 N 一样的配置.重复关系的重复次数 $k=5$.

表 9　服务节点模型 N'的队列库所属性表

库所	托肯	初始值	λ	库所	托肯	初始值	λ
F_1	$\{f_1\}$	{1}	1.2E−8	F_2	$\{f_2\}$	{1}	1.4E−8
R	$\{r\}$	{0}	1.5E−5	NP	$\{s,w,s',w'\}$	{0,0,0,0}	3.5E−3, 3.5E−3, 1.5E−4, 1.5E−4

不同交互关系下吞吐率的比较结果如图 19 所示.分析实验结果可知:(1)在并发关系下,虽然 N 和 N'的处理能力不同,但两者的吞吐率基本相同,而"$N|N'$"的吞吐率与较低的 x_N 更接近.(2)在条件关系下,选择概率高的 N 的吞吐率要高于 N',而"$N+N'$"的吞吐率是两者吞吐率之和.(3)在容错关系下,只有当 N 出现失效时,服务任务才会进入到 N'中,因此 N 的吞吐率比 N'的高出差不多两个数量级,而"$N|N'$"的吞吐率近似为两者吞吐率之和.(4)在重复关系下,因为需要在 N 中重复执行 $k=5$ 次,因此"μN"的吞吐率差不多是 N 的吞吐率的五分之一.

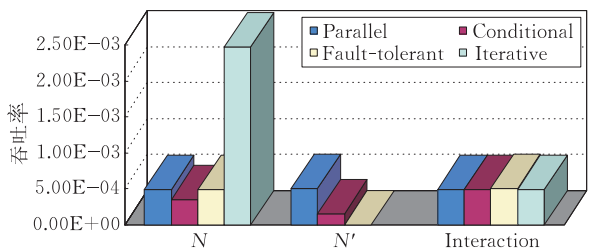


图 19　不同交互关系下吞吐率的比较

不同交互关系下平均响应时间的比较结果如图 20 所示.分析实验结果可知:(1)在并发关系下,由于 N'的处理能力比 N 强,因此 N'的平均响应时间要比 N 短;但是"$N|N'$"的平均响应时间比 N 还要长,分析其原因是,N 和 N'的通信失效发生率不同步,使得并发关系下的关联失效放大了失效恢复时间的影响.(2)在条件关系下,选择概率高的 N 的吞吐率要高于 N',而"$N+N'$"的吞吐率是两者吞吐率之和.(3)在容错关系下,虽然 N 和 N'的基本配置一致,但是实际进入到 N'中的任务量较少,负载较轻,因此 N'的平均响应时间要比 N 的短,而"$N|N'$"的平均响应时间要比 N 的略高.(4)在重复关系下,因为需要在 N 中重复执行 $k=5$ 次,因此"μN"的平均响应时间几乎是 N 的平均响应时间的 5 倍.

不同交互关系下丢失率的比较结果如图 21 所示.

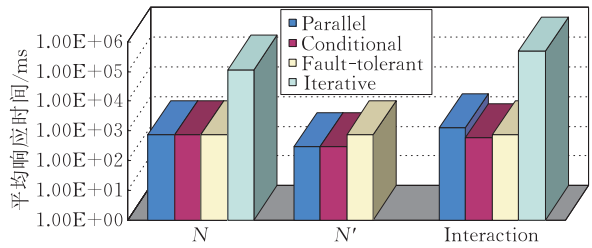


图 20　不同交互关系下平均响应时间的比较

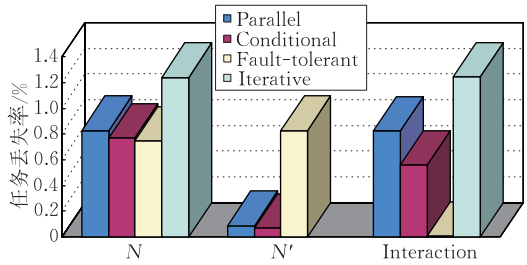


图 21　不同交互关系下丢失率的比较

分析实验结果可知:(1)在并发关系下,由于 N'的通信失效发生率低于 N,因此 N'中丢失的任务数目较少,所以 N'的丢失率比 N 的低不少,而"$N|N'$"的丢失率与较高的 N 的丢失率接近.(2)在条件关系下,N'具有较低的通信失效发生率,同时

被选择的概率也比较低，因此 N' 的丢失率要明显低于 N'；而“$N+N'$”的丢失率近似满足($L_N\times 70\%+L_{N'}\times 30\%$)的约束关系.(3)在容错关系下，虽然实际进入到 N' 中的任务量较少，但是 N' 和 N 具有相同的失效发生率，所以 N' 的丢失率和 N 的丢失率差不多；由于 N' 对 N 中的失效任务进行恢复执行，因此“$N|N'$”的丢失率降低到 0.0062%.(4)在重复关系下，N 和“μN”对失效的处理是同步的，因此两者具有一致的丢失率.

4.5 仿真服务提供阶段

服务提供阶段的性能度量属性包括稳态时的子服务 $S_{i,j}$ 和组合服务 S 的吞吐率 x、响应时间 T 和任务丢失率 L，结果数据的置信区间(c.i.)为 95%.对组合服务“$S_{1.1}>S_{2.1}(p)+S_{3.2}(1-p)>\mu S_{4.1}(k)>(S_{5.1}|S_{6.1})>(S_{7.1}|S_{8.2})>S_{1.2}$”分别按照集中和分散方式进行仿真的结果如表 10 所示.

表 10 服务提供阶段的分析结果

子服务	集中方式			分散方式		
	x	T(95% c.i.)	L/%	x	T(95% c.i.)	L/%
$S_{1.1}$	4.96E−04	878.95(+/−0.76)	0.81	4.96E−04	878.87(+/−0.73)	0.79
$S_{2.1}$	4.10E−04	741.13(+/−0.60)	0.79	4.10E−04	740.99(+/−0.57)	0.81
$S_{3.2}$	8.19E−05	698.61(+/−1.11)	0.79	8.20E−05	699.06(+/−1.10)	0.80
$S_{2.1}+S_{3.2}$	4.92E−04	734.05(+/−0.53)	0.79	4.92E−04	734.00(+/−0.51)	0.81
$S_{4.1}$	2.44E−03	72853.64(+/−2033.18)	0.80	2.44E−03	73031.26(+/−1648.66)	0.80
$\mu S_{4.1}(5)$	4.88E−04	374629.08(+/−3518.50)	0.80	4.88E−04	376140.54(+/−1834.21)	0.80
$S_{5.1}$	4.84E−04	729.60(+/−1.38)	0.78	4.84E−04	735.42(+/−1.89)	0.77
$S_{6.1}$	3.80E−06	721.55(+/−11.62)	0.83	3.72E−06	729.03(+/−14.04)	0.87
$S_{5.1}\|S_{6.1}$	4.88E−04	729.54(+/−1.37)	0.01	4.88E−04	735.37(+/−1.89)	0.01
$S_{7.1}$	4.84E−04	728.58(+/−0.92)	0.80	4.84E−04	722.95(+/−0.67)	0.81
$S_{8.2}$	4.84E−04	291.49(+/−0.26)	0.76	4.84E−04	291.09(+/−0.26)	0.80
$S_{7.1}\|S_{8.2}$	4.84E−04	810.78(+/−131.68)	0.80	4.84E−04	1500.39(+/−618.68)	0.81
$S_{1.2}$	4.82E−04	847.06(+/−1.11)	0.42	4.82E−04	842.75(+/−0.91)	0.42
CoS	4.82E−04	379643.75	17.98	4.80E−04	381051.55	3.58

组合服务的集中执行过程中，需要多次和服务代理库所 CB 进行交互，完成对托肯“z_1”，“z_2”，…，“z_6”和“z_7”的处理.令对应的平均处理时间表示为 $\tau(z_i)$，其中 $i=1,2,\cdots,7$.本例的仿真结果为

$\tau(z_1)=149.72\,\text{ms}$，$\tau(z_2)=150.14\,\text{ms}$，

$\tau(z_3)=150.30\,\text{ms}$，$\tau(z_4)=143.74\,\text{ms}$，

$\tau(z_5)=138.75\,\text{ms}$，$\tau(z_6)=139.12\,\text{ms}$，

$\tau(z_7)=142.52\,\text{ms}$.

那么可以近似计算出集中执行的组合服务 CoS 的平均响应时间为

$$\tau(CoS)=\tau(S_{1.1})+\tau(S_{2.1}+S_{3.2})+\tau(\mu S_{4.1}(5))+\tau(S_{5.1}|S_{6.1})+\tau(S_{7.1}|S_{8.2})+\tau(S_{1.2})+\sum\tau(z_i)=379643.75\,\text{ms}.$$

组合服务的分散执行过程中，只是在初始的任务调用和最后的结果汇总时才会和服务代理 DB 进行交互，分别用托肯“S”和“S'”表示.本例的仿真结果为

$\tau(S)=110.75\,\text{ms}$，$\tau(S')=108.87\,\text{ms}$.

那么可以近似计算出分散执行的组合服务 CoS 的平均响应时间为

$$\tau(CoS)=\tau(S_{1.1})+\tau(S_{2.1}+S_{3.2})+\tau(\mu S_{4.1}(5))+\tau(S_{5.1}|S_{6.1})+\tau(S_{7.1}|S_{8.2})+\tau(S_{1.2})+\tau(S)+\tau(S')=381051.55\,\text{ms}.$$

一般来讲，吞吐率通常用于度量小规模、耗时短的小型软件系统，或者单个软件模块的性能，而响应则适合度量规模较大、执行过程较复杂的系统的性能.比较相同配置下组合服务的两种执行方式可知，它们的吞吐率相同，集中方式下的任务丢失率较高，而分散执行方式下的平均响应时间较长.分析其原因，主要是在稳定的外部任务源输入的情况下，丢失率低表明实际处理的任务数量多，因此服务处理时间有所增加.

5 相关工作

目前，国内外关于服务系统的性能分析问题的研究还处于初步阶段.文献[21]提出了对组合的 Web 服务的性能和可靠性瓶颈进行闭环分析的随机模型方法.文献[22]使用扩展的广义随机 Petri 网(Extended Generalized Stochastic Petri Net，EGSPN)建模服务组合系统的性能，并且设计了模型化简规则和算法以解决状态空间爆炸问题.文

献[23]使用一组模糊微分函数描述服务组合及其性能，并通过平行算法来完成公式的计算. 文献[24]提出了一种云计算平台性能评价的框架 C-Meter，并在亚马逊弹性计算云上进行了初步的测试. 文献[25]给出了云计算性能预测的计算方法，但是缺乏相关的实验数据作为验证标准. 文献[26]分析了提供多任务科学计算的云计算服务的性能. 文献[27]提出了一种轻量级的企业服务总线测试框架. 文献[28]使用队列网络模型预测企业服务总线的响应时间，但该模型无法涵盖一些经典的企业服务总线应用模式(如消息的分裂和聚合等). 文献[29]研究了 QoS 感知的面向服务体系结构的性能问题. 文献[30]通过随机 Petri 网评价面向服务体系结构的性能，但是准确性会随着负载的增加而降低. 由此可见，已有研究有些是针对特定的封闭式系统，有些使用的模型相对简单，不能充分反映大规模复杂服务系统的特性.

排队网模型是一种确定性的网络建模工具[31]. Das 等人[32]使用分层排队网对多层服务系统的性能和可靠性进行了建模，并给出了基于马尔科夫链(Markov Chain)的分析方法. 但是由于存在信息量小、无结构化机制的弊端，排队模型只被应用于较为简单的情况. 文献[17]采用排队网(Queue network)分析了失效影响下的多层 Web 服务的性能，但是基于乘积网络(product-form network)的假设限制了该方法的适用范围. 随机 Petri 网[7]易于对系统状态进行全面有效的描述，精确刻画系统的不确定行为以及行为组合关系，便于计算各种分析指标，非常适合描述 Web 服务这种松耦合的分布式系统. Bernardi 等人[33]对基于概率模型的可信赖性评价方法进行了研究，并给出了利用随机 Petri 网对复杂系统进行建模和评价的方法. 林闯等人[19]研究了随机 Petri 网对网络系统可信赖性建模分析的方法和步骤，着重研究了随机 Petri 网描述系统的服务失效模型和容错模型，并给出了网络可信赖性分析中主要指标的计算方法. 排队 Petri 网继承和发展了排队网模型和随机 Petri 网的优点，能够更好地描述复杂的服务系统的性能表现[9-11].

因此，本文从通用的角度出发，通过分析服务系统的运行过程，研究并提出了一种基于排队 Petri 网的性能建模和分析方法.

6　结　论

随着信息服务的不断演化，以面向服务为基础的服务系统呈现出规模扩大化、结构复杂化、状态动态化的趋势，常用的性能评价形式化工具逐渐显出能力上的差异. 本文在分析服务系统运行过程和交互关系的基础上，提出服务准备和服务提供两个阶段. 服务准备阶段的运行环境由负载均衡器、应用服务器集群和数据库服务器构成，其建模方法类似多层 Web 系统. 服务准备阶段中的消费者和服务提供者的访问行为被归纳为 Browse，Select，Composite 和 Register 4 类，每类都由若干操作组成. 通过对各种行为在服务准备阶段的刻画，完成了服务准备阶段的 QPN 建模. 服务提供阶段由多个服务节点按照业务流程要求以组合方式向消费者提供服务. 对服务提供阶段的建模，除了考虑到不同交互关系(顺序、重复、并发、容错、条件)外，还要考虑时间约束、自动化要求、驻留节点的物理位置和节点失效恢复的影响. 在此基础上，应用 QPME 软件包对服务准备阶段性能模型和服务提供性能阶段模型进行了仿真. 模型的使用过程中可以设置不同的访问规模、更复杂的服务流程、更多的环境参数，构建出反映不同应用需求和环境配置的服务系统应用场景. 实验结果表明所提出的建模和分析方法具有很好的有效性和可扩展性.

与现有的服务系统性能分析方法相比，本文所提方法对服务系统性能的建模更全面，不但考虑到服务准备阶段的浏览、注册和组合行为，而且考虑到服务提供阶段中组合服务的不同执行方式(集中和分散)，尤其是将失效行为和恢复机制的影响引入到性能建模中，明显改善了性能量化的准确性. 采用的分析建模工具排队 Petri 网综合了排队网模型和随机 Petri 网的优势，能够刻画系统的执行细节和性能表现，极大地提高了建模和分析方法的实用性，为解决大粒度服务的复杂系统建模和服务质量评价问题提供了很好的解决方案. 在下一步的工作中，我们将继续优化服务系统分析性模型和评价方法，并在此基础上研究动态网络环境下的服务系统适配机制，增强服务对环境的适配能力.

致　谢　在此衷心感谢 Samuel Kounev 博士提供的排队 Petri 网仿真软件 QPME 1.01 对本文研究工作的帮助！衷心感谢评阅人对本文的工作提出的宝贵意见和建议！

参　考　文　献

[1] Papazoglou M P, Georgakopoulos D. Service oriented computing. Communications of the ACM, 2003, 46(10): 24-28

[2] Yin Jian-Wei, Chen Han-Wei, Deng Shui-Guang. Performance analysis of large-scale and complex service computing system. Communications of the China Computer Federation, 2010, 6(9): 32-36(in Chinese)
(尹建伟，陈韩玮，邓水光. 大规模复杂服务计算系统性能分析. 中国计算机学会通讯，2010，6(9)：32-36)

[3] Dai Yuan-Shun, Gregory Levitin. Reliability and performance of tree-structured grid services. IEEE Transactions on Reliability, 2006, 55(2): 337-349

[4] Dai Yuan-Shun, Pan Yi, Zou Xu-Kai. A hierarchical modeling and analysis for grid service reliability. IEEE Transactions on Computers, 2007, 56(5): 681-691

[5] Wang Yuan-Zhuo, Lin Chuang, Yang Yang, Shan Zhi-Guang. Research on manageability of grid service model method and management strategies. Chinese Journal of Computers, 2008, 31(10): 1716-1726(in Chinese)
(王元卓，林闯，杨扬，单志广. 网格服务可管理性模型及策略研究. 计算机学报，2008，31(10)：1716-1726)

[6] Lu Chuan-Lai. Queuing Theory (Version 2). Beijing: Beijing University of Posts and Telecommunications Press, 2009(in Chinese)
(陆传赉. 排队论(第 2 版). 北京：北京邮电大学出版社，2009)

[7] Lin Chuang. Stochastic Petri Nets and System Performance Evaluation (Version 2). Beijing: Tsinghua University Press, 2005(in Chinese)
(林闯. 随机 Petri 网和系统性能评价(第 2 版). 北京：清华大学出版社，2005)

[8] Bause F. Queueing Petri nets — A formalism for the combined qualitative and quantitative analysis of systems//Proceedings of the 5th International Workshop on Petri Nets and Performance Models. Toulouse, France, 1993: 14-23

[9] Kounev Samuel, Buchmann Alejandro. Performance modeling of distributed e-business applications using queueing Petri nets//Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software. Austin, Texas, 2003: 143-155

[10] Kounev S. Performance modeling and evaluation of distributed component-based systems using queueing Petri nets. IEEE Transactions on Software Engineering, 2006, 32(7): 486-502

[11] Kounev S, Nou R, Torres J. Autonomic QoS-aware resource management in grid computing using online performance models//Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS-2007). Nantes, France, 2007: 1-10

[12] Cardellini V, Casalicchio E, Grassi V et al. Flow based service selection for web service composition supporting multiple QoS classes//Proceedings of the IEEE International Conference on Web Services (ICWS' 07). Salt Lake City, UT, 2007: 743-750

[13] Dai Ru-Wei, Cao Long-Bing. Internet —— A open complex giant system. Science in China(Series E), 2003, 33(4): 289-296(in Chinese)
(戴汝为，操龙兵. Internet——一个开放的复杂巨系统. 中国科学(E 辑)，2003，33(4)：289-296)

[14] Nanda M G, Chandra S, Sarkar V. Decentralizing execution of composite web services//Proceedings of the OOPSLA'04. Vancouver, British Columbia, Canada, 2004: 170-187

[15] Guo Su-Chang, Yang Bo, Huang Hong-Zhong. Modeling and analysis for grid service reliability considering node recovery. Journal of Xi'an Jiaotong University, 2008, 42(6): 693-697, 790(in Chinese)
(郭夙昌，杨波，黄洪钟. 考虑节点失效恢复能力的网格服务可靠性建模与分析. 西安交通大学学报，2008，42(6)：693-697，790)

[16] Kounev S, Buchmann A. SimQPN: A tool and methodology for analyzing queueing Petri net models by means of simulation. Performance Evaluation, 2006, 63(4-5): 364-394

[17] Xie Bing, Chen Yong, Sun Xian-He et al. Performance under failure of multi-tier Web services//Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems. Shenzhen, China, 2009: 776-781

[18] Wu M, Sun X-H, Jin H. Performance under failure of high-end computing//Proceedings of the 2007 ACM/IEEE Conference on Supercomputing. Reno, Nevada, USA, 2007: 1-11

[19] Lin Chuang, Wang Yuan-zhuo, Yang Yang, Qu Yang. Research on network dependability analysis methods based on stochastic Petri net. Acta Electronica Sinica, 2006, 34(2): 322-332(in Chinese)
(林闯，王元卓，杨扬，曲扬. 基于随机 Petri 网的网络可信赖性分析方法研究. 电子学报，2006，34(2)：322-332)

[20] Kounev S, Dutz C, Buchmann A. QPME: Queueing Petri net modeling environment//Proceedings of the 3rd International Conference on Quantitative Evaluation of SysTems (QEST). Riverside, CA, 2006: 115-116

[21] Sato N, Trivedi K S. Stochastic modeling of composite Web services for closed-form analysis of their performance and reliability bottlenecks//Proceedings of the 5th International Conference on Service Oriented Computing (ICSOC 2007). LNCS 4749. 2007: 107-118

[22] Yang Huaizhou, Li Zengzhi. Extended GSPN modeling and reduction algorithms for rapid performance analysis of service composition system//Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 2009). Shanghai, 2009: 180-185

[23] Ding Zuohua, Shen Hui, Kandel Abraham. Performance analysis of service composition based on fuzzy differential equations. IEEE Transactions on Fuzzy Systems, 2011, 19(1): 164-178

[24] Yigitbasi N, Iosup A, Epema D, Ostermann S. C-Meter: A framework for performance analysis of computing clouds//Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. Shanghai, China, 2009: 472-477

[25] Xiong K, Perros H. Service performance and analysis in cloud computing//Proceedings of the 2009 Congress on Services-I. Los Angeles, CA, 2009: 693-700

[26] Iosup Alexandru, Ostermann Simon, Yigitbasi M Nezih et al. Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(6): 931-945

[27] Ueno K, Tatsubori M. Early capacity testing of an enterprise service bus//Proceedings of the 2006 IEEE International Conference on Web Services. Chicago, Illinois, USA, 2006: 709-716

[28] Liu Y, Gorton I, Zhu L. Performance prediction of service-oriented applications based on an enterprise service bus//Proceedings of the 31st Annual International Computer Software and Applications Conference. Beijing, China, 2007: 327-334

[29] Estrella J C, Toyohara R K T, Kuehne B T et al. A performance evaluation for a QoS-aware service oriented architecture//Proceedings of the 2010 6th World Congress on Services. Miami, FL, 2010: 260-267

[30] Teixeira M, Lima R, Oliveira C et al. Performance evaluation of service-oriented architecture through stochastic Petri nets//Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC). San Antonio, TX, 2009: 2831-2836

[31] Bolch G, Greiner S, de Meer H, Trivedi K S. Queueing Networks and Markov Chains. New York: John Wiley & Sons, 1998

[32] Das Olivia, Woodside C Murray. Dependable-LQNS: A performability modeling tool for layered systems//Proceedings of the 13th International Conference on Computer Performance Evaluation, Modeling Techniques and Tools (TOOLS 2003). Urbana, Illinois, USA, 2003: 672

[33] Bernardi Simona, Bobbio Andrea, Donatelli Susanna. Petri nets and dependability//Lectures on Concurrency and Petri Nets. 2003: 125-179

GU Jun, born in 1977, Ph. D. candidate, lecturer. His main research interests include network and service computing.

LUO Jun-Zhou, born in 1960, Ph.D., professor, Ph. D. supervisor. His research interests include next generation network architecture, protocol engineering, network security, grid and cloud computing, wireless local area network (WLAN).

CAO Jiu-Xin, born in 1967, Ph. D., professor, Ph. D. supervisor. His research interests include network security, service computing and digital right management (DRM).

LI Wei, born in 1978, Ph. D., associate professor. His research interests include next generation network, network management and service computing.

Background

This work is supported by the National Basic Research Program of China (973 Program) under Grant No. 2010CB328104, the National Nature Science Foundation of China under Grant Nos. 60903161, 60903162, 61003257, 61070161, 61070158, 61003311, Research Fund for the Doctoral Program of Higher Education of China under Grant No. 200802860031, Jiangsu Provincial Natural Science Foundation of China (Key program) under Grant No. BK2008030, the Jiangsu Provincial Key Laboratory of Network and Information Security under Grant No. BM2003201, the Key Laboratory of Computer Network and Information Integration of the Ministry of Education under Grant No. 93K-9.

Adaptation technologies provide effective approaches for acquiring high quality services from dynamic network environment. How to evaluate the performance of service systems is the key for resolving the adaption problem. Nevertheless, performance problems are ever more likely to occur in these environments, due to the dynamic interactions between a potentially large number of geographically and administratively distributed system components. While performance models of parallel and distributed systems are well investigated and there exist practical solutions, in most cases these techniques cannot be transferred directly to service systems. This paper focuses on the general performance modeling and analysis method of large scale of service systems. An analytic model is proposed for evaluating performance of service systems using Queueing Petri Net through analyzing its execution process. The simulation results show that this method can quantitatively analyze the performance of service systems.