文章编号:1001 - 9081(2008)07 - 1702 - 03

多引擎服务组合的执行优化

杨 林,林建素
(中国科学院 成都计算机应用研究所,成都 610041)510006
(yanglin05@mails.gucas.ac.cn)

摘 要:提出了多引擎的 Web 服务管理系统,解决了系统中服务组合的执行优化问题。分析了该多引擎系统中服务组合的调度执行,提出了动态规划算法。在分派和执行 Web 服务前,生成流水执行的子序列方案,使调度执行 Web 服务的引擎的最大负载最小。实验结果表明,与随机算法相比,该算法使性能显著提高。

关键词:Web 服务; 执行优化; 多引擎; 动态规划
中图分类号: TP302 **文献标志码**:A

Execution optimization for composite services through multiple engines

YANG Lin, LIN Jian-su
(*Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu Sichuan* 610041, *China*)

Abstract: A Web service Management System with Multiple Engines (WSMSME) was proposed to solve the problem of execution optimization for composite services in the system. The scheduler execution of composite services in system with multiple engines was analyzed, and a dynamic programming algorithm was put forward, which optimally minimized the heaviest load of engines by segmenting a pipelined execution plan into subsequences before they were dispatched and executed. Experiment with an initial prototype indicates that the algorithm can lead to significant performance improvement than the random algorithm.

Key words: Web services; execution optimization; multiple engines; dynamic programming

0 引言

近年来 Web 服务技术得到了快速的发展和应用,它是基于网络的、分布的、自描述的模块化组件,是解决业务过程执行和应用集成的一种有效手段。但 Web 服务与其他分布式计算技术(如 CORBA、Java RMI、DCOM)的实现相比较,延迟及吞吐率具有一定的差距[1],所以优化 Web 服务执行,改善 Web 服务质量成为了目前研究的一大热点。

Web 服务的应用包括服务提供者使用服务描述语言发布服务,服务使用者采用某种策略发现所需服务并绑定调用执行服务。目前,针对 Web 服务应用模式及其优化有大量的研究工作,如对 Web 服务进行语义扩展(如 OWL-S、WSDL-S、WSML 等[2]),建立先进的服务发现体系结构(如 P2P 结构[3]),采用更好的服务匹配算法(如文献[4]中基于本体论的匹配算法),在服务组合方面有基于服务请求分类的多主体服务组合优化[5]等。这些措施在某种程度上确实提高了服务性能,但没有意识到多引擎架构和服务流水分段执行的优点,或者意识到了而没有提出有效的措施。基于此本文提出多引擎的 Web 服务管理系统(WSMSME),并形成成本模型,优化 Web 服务组合的调度执行,提高服务性能。

1 WSMSME 系统简介

1.1 系统原理

WSMSME 系统是一个通用的多引擎 Web 服务管理系统。如图 1 所示,WSMSME 系统主要由 3 部分组成:

1)元数据组件(Metadata Component):处理元数据的管理,新的 Web 服务的注册,以及方案映射,并将整合的视图展示给客户,接受客户发送的请求。

2)执行进程和优化组件(Query Processing and Optimization Component):处理 Web 服务请求的优化和调度执行。例如,调用几个相关的 Web 服务,形成执行方案。

3)配置和统计组件(Profiling and Statistics Component):配置 Web 服务的反应时间,空间成本以及维护相关的统计和日志等,为执行优化器提供参考数据。

WSMSME 系统与分布式数据整合系统的中间者[6-7]类似。它接受客户的请求,查找相应的 Web 服务组合,并将服务组合分段,动态分派适当的执行引擎去调用执行 Web 服务,并将最终结果返回给客户。执行进程和优化组件是系统的核心,它根据调度策略,动态调度执行引擎执行不同的 Web 服务。其中查询优化器(Query Optimizer)实现了本文描述的优化算法。

1.2 系统优点

与单引擎架构比较,该系统具有以下优点。

1)提高了系统的可靠性、响应的及时性。单引擎可能不能完成整个进程的执行,甚至死锁崩溃等,特别是计算机在计算能力和资源不十分充裕的环境下;同时单引擎处理比较耗时的 Web 服务时,来自其他的客户请求便得不到处理,使之陷入长时期的等待之中。而多引擎能够提供可靠性保障,当一个引擎出现故障后,组合的服务可以通过另外 $k-1$ 个引擎调度执行。同时,提高了系统响应的及时性。

2)提高并行性。通过优化算法,可以把 Web 服务组合分段,使之流水执行,减小反应时间,提高效率。

3)分布式。多引擎可以分布在不同的系统上,分布的查询可以获得更多的资源,更有利于 Web 服务组合的执行。

收稿日期:2007 - 12 - 18;**修回日期**:2008 - 02 - 03。

作者简介:杨林(1980 -),男,四川南充人,硕士研究生,主要研究方向:嵌入式系统; 林建素(1982 -),男,浙江温州人,硕士研究生,主要研究方向:软件过程与技术。

4)更具有商业价值。多引擎分布在不同系统上,可以按引擎的执行来收费,或者在执行结果返回的 XML 中嵌入广告代码。

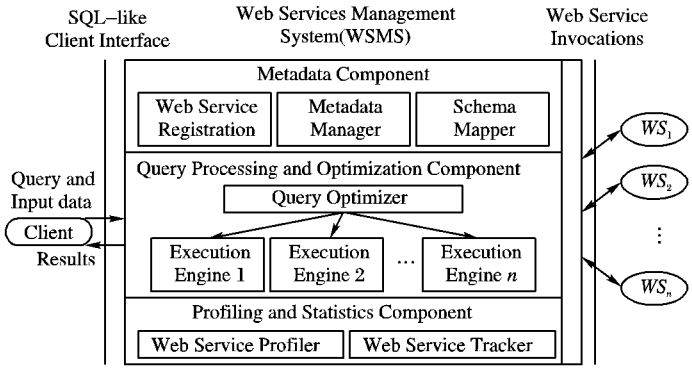

图 1　WSMSME 系统

2　执行优化

如图 1 所示,WSMSME 系统采用了多引擎架构,当接收到客户请求时,可以采用某种策略去分派引擎调度执行 Web 服务,例如随机调度策略,随机调用引擎处理 Web 服务,或将服务组合随机分成子序列执行。但这些策略有可能导致一些引擎负载过重,执行成本过高,系统响应缓慢,甚至崩溃,而其他一些引擎却长期处于空闲状态。为了避免这种现象的发生,我们采用动态规划算法来优化服务组合的调度执行,使各引擎的最大负载最小,实现负载均衡。

2.1　组合模式

在 WSMSME 中,Web 服务之间通过元数据组件进行通信,假设服务组合 $WS_1 \cdots WS_n$ 是顺序的,如图 2 所示,WS_{i+1} 的输入来自 WS_i 的输出($0 < i < n$),所有的 Web 服务按顺序执行,WS_i 执行完之后 WS_{i+1} 才能执行,并且保证每个 Web 服务的输入都能准确获得。目前我们只考虑顺序执行的 Web 服务组合模式。

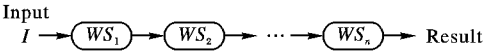

图 2　Web 服务组合模式

2.2　问题定义

假设用 M 个引擎来调度执行 K 个 Web 服务,用属性 C 表示调度执行 Web 服务的成本,包括时间、内存等,$WS_i(C_i)$ 则表示 Web 服务 WS_i 的调度执行成本是 C_i。引擎的负载即为该引擎调度执行的 Web 服务的成本之和。当然单用一个 C 来表示执行成本过于简单,将来应考虑使用更复杂的机制去表述 Web 服务的调度执行成本。

例 1:设 Web 服务的调度执行成本和可用的引擎数如表 1 所示。

表 1　Web 服务的成本和引擎数

K	$WS_K(C_K)$	M
1	4	
2	2	2
3	3	

由于是顺序组合模式,有 3 个分派方案:1)分配引擎 M_1 执行 WS_1 和 WS_2,引擎 M_2 执行 WS_3;2)分配引擎 M_1 执行 WS_1,引擎 M_2 执行 WS_2 和 WS_3;3)让其中一引擎执行所有 Web 服务。很显然,方案 3 应首先排除的,因为它使一个引擎承担所有任务,另一个却一直空闲,背离了负载均衡的原则。方案 1 和 2 中,很容易区分出 2 较好,因为 M_1 和 M_2 的最大负载是 5。

为了衡量不同分派方案的优劣,我们引入不满意度指数 DI 来表示方案的引擎最大负载。DI 值越小,该方案越好。采用 M 个引擎(用 $EE_1, EE_2, \cdots, EE_M$ 表示),那么有:

$$DI = \max\{\sum_{EE_i 执行 WS_K} C_K \mid i \in [1, M]\}$$

对于例 1,我们得到:

$DI(a) = \max\{(4+2), 3\} = 6$

$DI(b) = \max\{4, (2+3)\} = 5$

$DI(a) > DI(b)$,由此可见,方案 2 较好。

2.3　优化算法

$D(i,j)$ 表示用 i 个引擎调度执行 j 个 Web 服务时的不满意度指数,即 i 个引擎中的最大负载。当 $i \geqslant j$ 时,最大的引擎负载显然是 $\max(C_1, C_2, \cdots, C_i)$,由于引擎数比 Web 服务数还大,系统不会出现问题,所以主要优化 $i < j$ 的情形。

当只有 1 个引擎时,所有 Web 服务都由它来完成,最大负载即为要执行的 Web 服务成本之和,所以有:

$$D(1,i) = \mathrm{sum}(C_1, C_2, \cdots, C_i)$$

当引擎数 $i > 1$ 时,可以利用已经生成的用 $i-1$ 个引擎的执行 Web 服务的优化方案来生成用 i 个引擎的执行方案,由于是顺序组合服务模式,我们可以让第 i 个引擎执行第 t 到第 j 个 Web 服务 ($i \leqslant t \leqslant j$),从而达到分段执行的目的,同时这些方案中,得到的最小的最大引擎负载的方案最佳。于是 $D(i,j)$ 为:

$$\min\{\max\{D(i-1, t-1), \sum_{k=t}^{j} C_k\} \mid i \leqslant t \leqslant j\}$$

由此,得到以下算法:

```
1)  k ← 可用的执行引擎数, n ← Web 服务数
2)  for (i = 1; i ≤ n; i ++)
3)    D(1,i) = sum (C1, C2, ···, Ci)
4)    for (i = 2; i ≤ k; i ++)
5)      for (j = i; j ≤ n; j ++)
6)        for (t = i; t ≤ j; t ++)
7)          tm = max {D(i-1, t-1), Σ_{k=t}^{j} Ck}
8)          if (tm < D(i,j))
9)            D(i,j) ← tm
10)   Return D(k,n)
```

该算法的时间复杂度为 $O(kn^2)$(n 为 Web 服务数,k 为引擎数)。由于将 Web 服务组合分段执行,当有多个请求时,可以流水执行,提高了并行性,减小反应时间。

3　实验

3.1　WSMSME 原型和实验设置

实验设置主要有两部分:由 WSMSME 组成的客户端和 Web 服务的服务端。

WSMSME 原型是一个用 Java 语言编写的实现了本文中动态规划算法的多线程系统。在原型中执行引擎是虚拟的,它只是一个普通的多线程对象,可以创建和删除特定的引擎。我们使用 Code XFire 工具自动生成 Web 服务语言描述的 Web 服务类,通过类的方法来调用 Web 服务,并使用 SOAP 协议进行 Web 服务间通信,同时在执行引擎中实现了调用 Web 服务执行功能。

在服务端,采用 Apache Tomcat 作为应用服务器,同时运用 Codehaus XFire 工具来部署 Web 服务。每一个 Web 服务运行在不同的机器上,并在 MySQL 数据库中有一张表 T(int a, int b, a 为主键)与之对应。对于属性 a, Web 服务可通过 SQL 查询取得相应的 b 值。

实验中,使用延迟来模拟每个Web服务具有的不同执行成本。WSMSME系统和Web服务运行在不同的机器上。另外每个Web服务组合由一系列的顺序的Web服务组成(其他的Web服务组合模式暂时不考虑)。

3.2 实验与结果分析

在3.1节原型系统中还实现了随机算法(RANDOM),用它和本文中动态规划算法(表示为OPTIMIZER)进行比较。随机算法采用的是将Web服务组合随机分成 个子序列进行执行的策略。

我们开发了19个Web服务,每个服务的执行成本在30到160之间,并从1到15动态增加引擎数。实验结果如图3所示,随着引擎数的增加,两种算法都能够使最大负载不断减小,但OPTIMIZER算法的性能始终比随机算法要好。

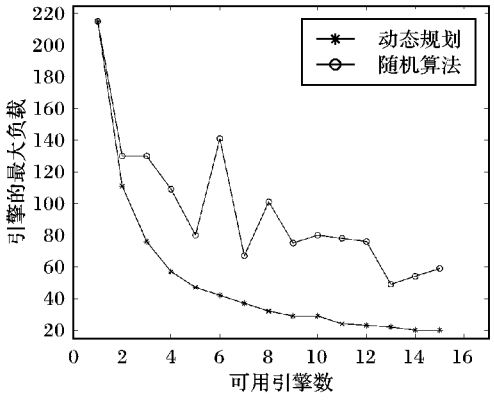

图3 引擎的最大负载

4 结语

基于多引擎的Web服务管理系统,弥补了单引擎架构的缺点,同时解决了WSMSME的调度执行优化问题。在Web服务分派和执行前,将其分成流水执行的子序列来最小化引擎的最大负载。然而这只是对WSMSME系统最基本的优化,并且实验的设计趋于理想化,下一步的工作应将算法扩展到其他更复杂的Web服务组合模式,并考虑使用更复杂的机制来表述Web服务的执行成本,尝试在Web服务组合语言如BPEL4WS方面做更多的努力。

参考文献:

[1] 马晓轩,林学练. Web服务性能优化的研究[J]. 计算机工程与应用,2005,41(8):19-20.

[2] MCILRAITH S A, MARTIN D L. Bringing semantics to Web services [J]. IEEE Intelligent Systems, 2003, 18(1): 90-93.

[3] Lü WEI-FENG, YU JIAN-JUN. pService: Peer to Peer based Web services discovery and matching [C]// Proceedings of 2nd International Conference on System and NetWorks Communications (ICSNC 2007). Washington, DC. IEEE Press, 2007: 54-54.

[4] SHU GAO, RANA O F, NICK J, *et al*. Ontology-based semantic matchmaking approach [J]. Advances in Engineering Software, 2007, 38(1): 59-67.

[5] 蒋运承,史忠植. 多主体服务组合的优化策略[J]. 计算机工程与应用,2004,40(6):1-3.

[6] HONG W, STONEBRAKER M. Optimization of parallel query execution plans in XPRS [C]// Proceedings of the First International Conference on Parallel and Distributed Information Systems. Washington, DC. IEEE Press, 1991: 218-225.

[7] VIGLAS S, NAUGHTON J F, BURGER J. Maximizing the output rate of multi-join queries over streaming information sources [C]// Proceedings of the 2003 International Conference on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 2003: 285-296.

(上接第1688页)

统计本文的测试结果,在消歧过程中,词语搭配规则起了巨大作用,涵盖了67.3.%的歧义字段切分,其正确率很高,对于"学会"达到100%。

此外,不使用语法规则,单独使用Naïve Bayes切分的正确率为71%。如果先采用语法规则切分,符合切分条件的有50句,正确率达到92%,再对剩余的5句基于Naïve Bayes切分,正确率为60%。综合而言,单独采用Naïve Bayes正确率为71%,语法规则和Naïve Bayes相结合的方式正确率为89%,提高了18%。由此可见,语法规则的引入优于单纯的Naïve Bayes方法。

4.2 模型分析

同样都是利用上下文信息进行切分,文献[1-2]只考虑了歧义字段前后文中词的出现频率,而忽略了词的搭配和词性信息对于切分的影响。本文算法通过提取词语搭配规则、语法规则和Naïve Bayes方法充分挖掘并利用上下文信息,在歧义字段的切分上拥有更全面和准确的信息和依据。在通用性上,本文模型从一般情况出发,模拟人脑判断歧义字段的思维过程,首先考虑词的搭配,再从词性出发,理论依据充分。

5 结语

组合型歧义字段切分是中文自动分词的难点之一,长期以来一直没有完全解决,成为提高分词准确率,从而进一步推动中文信息处理应用的瓶颈。本文在对现有方法进行深入分析的基础上,提出了一种基于规则挖掘和Naïve Bayes方法的组合型歧义字段切分算法,该算法自动从训练语料中挖掘词语搭配规则和语法规则,然后基于这些规则和Naïve Bayes模型综合决策,对组合型歧义字段进行切分消歧。充分的实验结果表明,相对于文献中的研究结果,本文算法对组合型歧义字段切分的准确率提高了大约8%。

作为下一步工作,将考虑扩大前后文窗口的范围,使上下文的信息量更大,进一步提高切分的正确率。当然,随着前后文窗口的扩大,噪声也会增加,如何在扩大前后文的情况下减小噪声的影响是需要重点考虑的问题。

参考文献:

[1] 曲维光,吉根林,穗志方,等. 基于语境信息的组合型分词歧义消解方法[J]. 计算机工程, 2006,32(17):74-76.

[2] 肖云,孙茂松,邹嘉彦. 利用上下文信息解决汉语自动分词中的组合型歧义[J]. 计算机工程与应用,2001,37(19):87-89.

[3] 李静梅,孙丽华,张巧荣,等. 一种文本处理中的朴素贝叶斯分类器[J]. 哈尔滨工程大学学报,2003,24(1):71-74.

[4] 廉竹钧. 汉语组合型切分歧义字段消歧方法研究[C]//第一届学生计算语言学研讨会. 北京:北京语言文化大学,2002:65-70.

[5] 郑家恒,吴芳芳. 多义型歧义字段切分研究[M]. 北京:清华大学出版社, 1999:129-134.

[6] 曾华琳, 李堂秋. 一种基于提取上下文信息的分词算法[J]. 计算机应用,2005,25(9):2025-2027.

[7] 冯素琴,陈惠明. 一种自组织的汉语组合型歧义消歧方法[J]. 计算机工程与设计,2007,28(3):737-739.