面向SaaS应用的SMTDM可伸缩多租户数据管理框架①

尤晓青

(西安工程大学 计算机科学学院, 西安 710048)

摘 要: 云计算是一种可以弹性并按需提供资源的技术. 在多租户共享数据存储模式下, 如何实现数据的动态伸缩存储是云数据管理的关键. 针对 SaaS 应用如何随着租户数量及请求规模的变化而进行自适应伸缩的问题, 在分析数据存储层的伸缩性需求的基础上, 基于 Walraven 等人提出的多租户架构思想, 扩展典型的云应用架构, 设计了一个多租户数据管理框架, 实现存储资源的弹性. 基于该框架, 开发了一个面向网络管理领域的 SaaS 原型系统, 验证了其有效性和可用性.

关键词: 可伸缩; SaaS; 多租户数据管理框架; SMTDM; 存储资源弹性

Scalable Multi-Tenancy Data Management (SMTDM) Framework for SaaS Aplications

YOU Xiao-Qing

(School of Computer Science, Xi'an Polytechnic University, Xi'an 710048, China)

Abstract: Cloud computing is a technology that enables elastic, on-demand resource provisioning. In multi-tenant shared data storage model, how to achieve dynamically scalable data storage is the key of cloud data management. For the issue of how SaaS applications adaptively scale with the change of the quantity of tenants and request, on the basis of analyzing scalability requirements of the data storage layer, based on Walraven et al. proposed multi-tenant architecture, we extend the typical cloud application architecture, and design a multi-tenant data management framework, achieving scalability of storage resources. Based on the framework, a SaaS prototype system for network management is developed, verified its validity and availability.

Key words: scalability; SaaS; multi-tenancy data management framework; SMTDM; elasticity of storage resources

1 引言

SaaS 应用为多个租户提供服务, 他们共享同一应用实例, 通过提供租户定制化支持和数据隔离及性能隔离技术, 实现私有实例的效果. 云平台通过自动配置资源允许运行其上的应用自动适应负载变化. SaaS 应用根据当前负载运行在一个或多个应用服务器实例上. 如果当前负载过大, 必须增加额外的应用服务器实例. 同样, 租户数据可以被分配到多个数据库实例上. 如果当前负载过高, 必须增加额外的数据库服务器实例, 并且现有的租户数据必须被重新布局. 随着租户数量的增加, 对数据库容量的需求会呈几何级速度不断增长, 迫切需要一个灵活的、可伸缩的架构.

伸缩性是 SaaS 模式成熟度的最高级别, 指系统的可伸可缩, 主要强调的是性能和容量方面的可扩展. 最理想的情况是随着租户数量的增加, 系统架构不用改变, 而仅需要增加相应的硬件设备即可.

云环境下, SaaS 应用的可伸缩性主要与性能、负载、资源、费用等因素相关[1]. 文献[2]提出了一个类 OSGi 的 COSCA 框架, 通过监控不同应用组件在多个计算节点上的分布来决定何时进行伸缩, 自动管理云计算集群中基于组件的应用. 文献[3]建立了一个 CloudScale 系统, 面向多租户云计算基础设施进行自动化细粒度弹性资源扩展, 实现云中各种应用所需资源的自适应分配. 文献[4]提出了一个面向多租户的多

① 收稿时间:2016-03-12;收到修改稿时间:2016-04-11 [doi:10.15888/j.cnki.csa.005451]

层次可伸缩的 SaaS 软件架构, 分别针对业务层和数据层描述了可伸缩的实现方法. 然而, 当前研究更多关注云计算和应用层的可伸缩性, 对于数据层的可伸缩性研究相对较少.

文献[5]提出了一个多租户管理的架构, 实现了数据隔离、性能管理和租户定制. 本文基于此思想进行扩展, 从 SaaS 应用数据存储层的伸缩性需求出发, 构建了一个可伸缩的多租户数据管理 SMTDM(The Scalable Multi-Tenant Data Management)框架, 实现了高可伸缩性.

2 SaaS应用数据存储层的伸缩性需求

可伸缩性是 SaaS 应用的非功能性需求, 数据存储层的伸缩性是实现 SaaS 应用高性能和降低成本的重要方面. 在 SaaS 应用架构中, 实现数据层的伸缩性涉及多个方面的需求, 包括性能需求、存储容量的需求、负载均衡处理的需求等.

性能需求. 租户与 SaaS 应用提供商签订服务等级协议 SLA(service level agreement). SLA 中的性能要求, 尤其是租户对于响应时间的要求是本文考虑的重点. 不同的租户有不同的性能需求, 有些租户要求有较高的响应速度, 而有些租户则为了降低租用成本对响应速度要求较低, 够用就行. 这就要求 SaaS 应用能够根据租户的不同性能需求动态分配存储资源, 从而保证每个租户的性能需求.

存储容量的需求. SaaS 应用中, 由于多个租户共享相同的应用软件, 且每个租户拥有大量的用户, 因此 SaaS 模式具有用户数量多、数据吞吐量大等特点. 在一个可伸缩的 SaaS 应用架构中, 当租户数量或用户请求增加时, 存储资源的消耗会上升, 当租户数量或用户请求减少时, 存储资源的消耗就会减少, 这就要求 SaaS 应用的存储容量能够随着需求的变化而弹性伸缩.

负载均衡处理的需求. SaaS 模式下, 租户的数据分布存储在多个数据节点上, 通过并行处理能够在一定程度上提高系统的性能. 当进行数据操作时, 负载均衡器可以在多个数据节点上进行并行操作, 提高系统的执行效率.

3 SMTDM框架

3.1 系统整体架构

本部分针对上述关于 SaaS 应用数据存储层的伸缩性需求分析, 设计一个多租户数据管理框架, 获取存储资源的高可伸缩性, 系统整体架构如图 1 所示. 该架构在多租户应用平台层和数据库层之间引入一个抽象层-SMTDM 层. 多租户应用平台层包括租户感知层、业务逻辑层和数据访问层. 大多数 SaaS 应用使用数据库存储数据, 数据访问层为业务逻辑层提供一个访问持久化数据的接口. 这些数据库实例集群在一起形成数据库层. 数据库层依据应用的需求可由关系型数据库、NoSQL 数据库或两者的混合构成. 引入 SMTDM 层, 使数据库层的持久化数据的物理分布实现细节对数据访问层透明. 数据访问层向下提供一个通用的接口, 独立于使用的数据库系统和租户数据的物理分布.

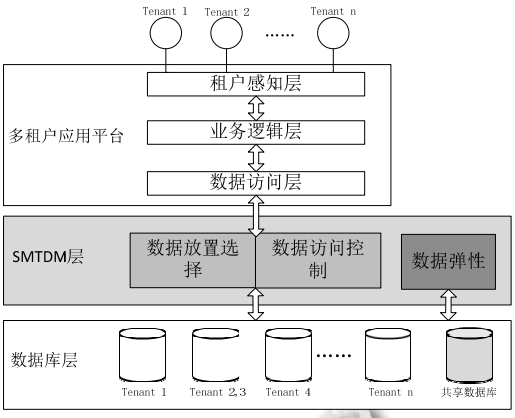

图 1 系统整体架构

图 1 中, 我们在 SMTDM 层内部定义了三个主要的组件: 数据放置选择组件、数据访问控制组件和数据弹性组件. 数据放置选择组件的主要功能是选择合适的数据库实例存放租户数据并能高效检索租户数据. 数据访问控制组件验证当前用户是否有权限读取或者修改所要访问的数据. 数据弹性组件的功能是, 基于当前数据库的负载情况并考虑数据保留策略等约束, 添加或者移除数据库实例和重新布局租户数据, 实现存储资源的高可伸缩性.

对于数据放置选择和数据访问控制问题, 目前国内外已有较多研究成果. 如文献[6]中提出的基于租户树的映射机制和自底向上的检索方法解决数据的放置选择问题. 文献[7]和[8]中提出的基于属性的访问控制方法实现数据访问控制. 数据放置选择组件和数据访问控制组件向数据访问层定义了接口. 数据放置选择组件和数据访问控制组件可采用已有的研究成果实现,

本文重点考虑数据弹性组件的设计. 数据弹性组件向上提供一个管理接口用于监控和配置, 向下仅和数据库层通信, 不可从多租户应用内部访问. 数据弹性组件是框架中最重要和最复杂的组件, 本文接下来详细的描述这个组件的设计.

3.2 数据弹性组件

将传统应用迁移到云平台上最大好处就是可以增加它的弹性, 以自动配置的方式添加或移除资源来适应应用负载的变化. 这种弹性可以被云提供商或第三方作为一种服务提供. 如运行在 Amazon AWS 上的应用, Amazon 通过 CloudWatch 监控资源负载从而提供弹性功能[9]. 而对于运行在 Microsoft Azure 上的应用, 通过第三方的产品, 如 AzureWatch 提供弹性功能[10]. 云提供商也提供一个应用程序接口(API), 允许应用开发商实现一个定制的弹性组件. 然而, 这些解决方案主要针对计算资源的弹性, 很少关注存储资源的弹性.

我们在 SMTDM 层引入的数据弹性组件, 其主要功能是按需分配数据库实例数量(水平扩展), 从而获取存储资源的高可伸缩性. 通过评估当前所有数据库实例的负载, 并考虑数据保留策略等约束, 添加或移除额外的数据库实例和重新布局租户数据. 数据弹性组件的功能如图 2 所述. 数据弹性组件根据当前数据布局信息和负载监控信息, 判断是否超出预设的阈值范围, 从而生成数据布局调整策略, 应用于数据库节点. 数据弹性组件应该作为一个独立运行的进程实现, 不影响主应用的性能.

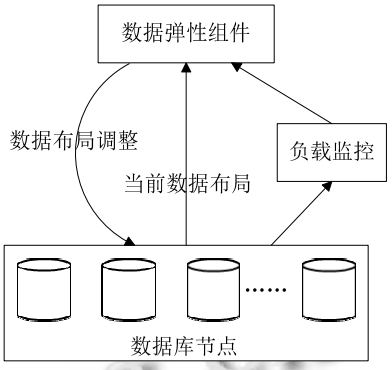

图 2 数据弹性组件的功能

系统可伸缩性的执行流程如图 3 所示. 系统通过在一个初始数量的数据库实例上分配租户数据开始, 然后进入控制循环. 在这个控制循环过程中, 不同数据库实例的当前负载被周期性地评估. 如果当前负载过高, 系统将会添加一个或多个额外的数据库实例, 并且重新布局当前租户数据, 之后控制循环继续执行. 类似的, 如果当前负载过低, 系统将会重新布局当前租户数据, 并移除空的数据库实例. 数据弹性组件使用数据布局调整算法来在数据库实例上找到一个可行的租户数据布局调整方案. 一定的事件可以触发对当前数据库实例的负载进行重新评估, 如: 当一个数据库实例的负载超出阈值范围(过载或者过轻)时; 当一个租户被添加或移除时.

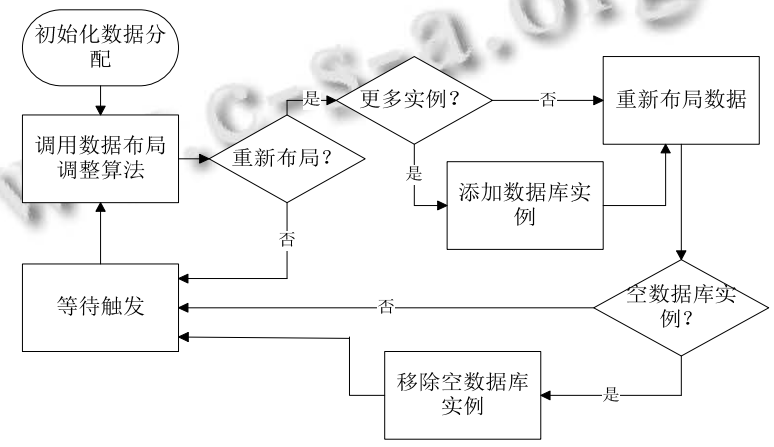

图 3 系统可伸缩性的执行流程

3.3 数据库层

应用程序的数据持久化既可以使用传统的提供 SQL 接口的关系型数据库系统, 也可以使用 NoSQL 数据库系统. NoSQL 数据库, 如开源的 Apache

Cassandra、MongoDB、Redis 等, 在某些场景下比关系型数据库性能更好. 文献[11]中比较了 SQL 和 NoSQL 数据库的性能, NoSQL 数据库并不总是比 SQL 数据库性能更好. 本文设计的框架是独立于底层的数据库系统的, 既可以使用SQL数据库实现也可以使用NoSQL数据库实现, 或者二者结合实现.

数据库层由多个数据库实例组成, 根据租户所要求的服务级别, 租户可以使用一个独有的实例或者和其它租户共享同一个实例. 对于有大量数据的租户可以使用数据分片技术将数据存储在多个数据库实例上. 数据库层包含一个共享的数据库实例, 这个实例存储所有租户的配置信息, 如租户指定的配置参数, 计费信息和数据保留策略等. 也存储着每个租户的当前数据分配映射信息. 数据放置选择组件访问这个共享的数据库, 以选择正确的租户数据库实例读取数据. 这个共享的数据库实例不应该成为系统的瓶颈, 当这个数据库实例负载过大时, 可以采用复制或者分区技术进行负载均衡. 数据弹性组件不应该被使用在这个共享数据库上, 这个组件仅仅用来处理其它租户数据库实例的可伸缩性.

4 原型系统实现

在研究可伸缩的 SaaS 多租户数据管理框架的基础上, 开发了一个面向网络管理领域的多租户 SaaS 原型系统. 原型系统包括数据采集层、应用服务器层、数据访问和可伸缩层、数据库层等几个部分. 其中数据采集层从租户单位的被管网元 NE(Network Element)采集网管数据传送给应用服务器层, 并通过数据访问层写入到相应的数据库中.

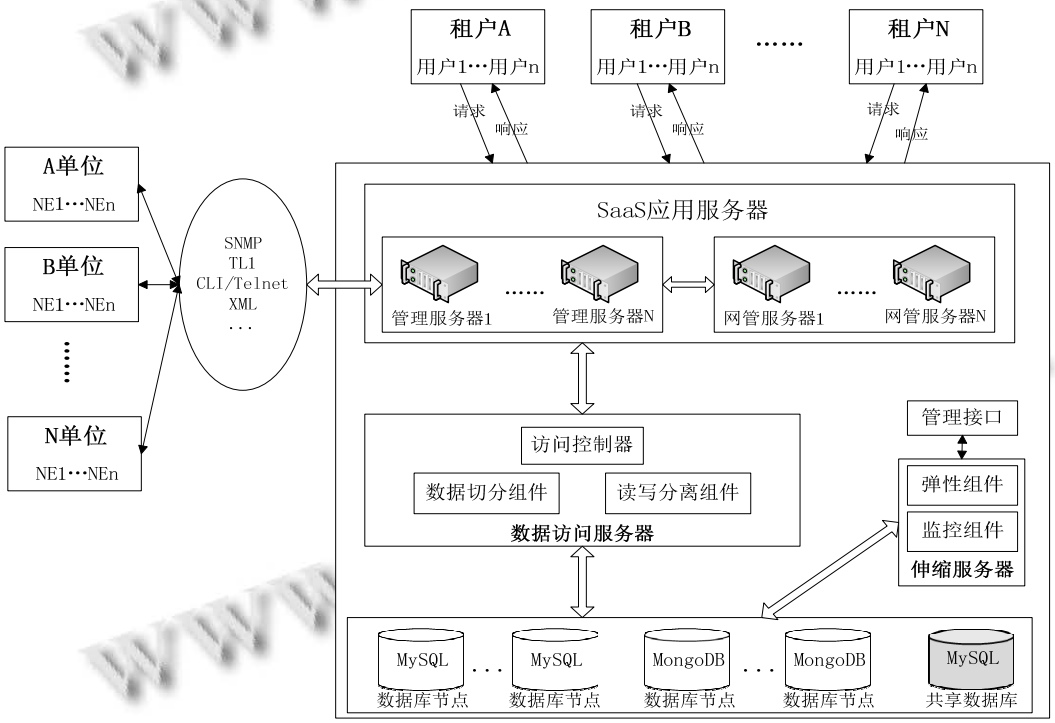

图 4 原型系统实现

图 4 展示的是原型系统的一个部署和运行架构. 由 SaaS 应用服务器、数据访问服务器、伸缩服务器和数据库服务器等几个部分构成. 首先在各租户单位安装采集器, 用来采集被管网元数据. 管理服务器通过不同的网管协议(如 SNMP 协议、TL1 协议、Telnet 协议和 XML 等), 与被管网元交互, 处理从各租户单位被管网元采集到的数据, 发送给网管服务器或通过数据访问服务器存入数据库. 数据访问服务器包含访问控制器组件、数据切分组件和读写分离组件. 访问控制器处理不同租户的数据访问权限, 数据切分组件对租户数据进行水平切分实现数据的水平扩展, 读写分离组件可以增加系统的性能. 伸缩服务器包含弹性组

件和监控组件, 并提供管理接口用于监控和配置. 监控组件监控各数据库实例的运行状态, 弹性组件根据数据库实例的负载实现数据库节点的扩展和收缩.

根据网络管理系统的数据特点, 数据库节点可由关系型数据库 MySQL 和非关系型数据库 MongoDB 两者混合构成. 其中 MogoDB 用来存储具有时间序列特征的网络设备性能数据.

未来的工作就是完善和细化可伸缩的 SaaS 多租户数据管理框架中的内容, 深入研究数据布局调整算法的设计和实现以及 SaaS 应用层的可伸缩性等问题, 并在此基础上完善原型系统的开发.

参考文献

1 张涛,燕静,徐照淼,等.云计算环境下 SaaS 服务可伸缩性评估方法研究.西北工业大学学报,2014(6):998–1000.

2 Chele S, Hauck FJ. Component-based scalability for cloud applications. Proc. of the 3rd International Workshop on Cloud Data and Platforms. ACM. 2013. 19–24.

3 Shen Z, Subbiah S, Gu X, et al. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. Proc. of the 2nd ACM Symposium on Cloud Computing. ACM. 2011. 5.

4 周学权,战德臣,聂兰顺,等.面向多租户的多层次可伸缩 SaaS 软件架构研究.华中科技大学学报(自然科学版), 2013(S2):131–136.

5 Walraven S, Truyen E, Joosen W. A middleware layer for flexible and cost-efficient multi-tenant applications. Proc. of the 12th International Middleware Conference. International Federation for Information Processing. 2011. 370–389.

6 Maenhaut PJ, Moens H, Decat M, et al. Characterizing the performance of tenant data management in multi-tenant cloud authorization systems. Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014: 1–8.

7 Decat M, Lagaisse B, Van Landuyt D, et al. Federated authorization for software-as-a-service applications. On the Move to Meaningful Internet Systems: OTM 2013 Conferences. Springer Berlin Heidelberg. 2013. 342–359.

8 Decat M, Lagaisse B, Joosen W. Middleware for efficient and confidentiality-aware federation of access control policies. Journal of Internet Services and Applications, 2014, 5(1): 1–15.

9 Amazon Cloudwatch - Cloud and Network Monitoring Services. Available: http://aws.amazon.com/cloudwatch/.

10 Autoscaling and Monitoring for Windows Azure applications. Available: http://www.paraleap.com/azurewatch.

11 Li Y, Manoharan S. A performance comparison of SQL and nosql databases. 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM). IEEE. 2013. 15–19.