基于动态限速的云计算应用负载调度方法

刘永波[1],周 博[1],李亚琼[1],李守超[1],宋云奎[2]
(1. 江苏润和软件股份有限公司,江苏 南京 210012;
2. 中国科学院 软件研究所,北京 100190)

摘 要: 云计算环境下,应用负载规模巨大,云服务提供商为多个客户提供共享的计算、网络和存储资源以最大化资源利用率,降低总体能耗,从而减少数据中心的运营成本,同时需要为用户提供良好的性能保障。针对该问题,提出一种基于动态限速的云应用负载调度方法。面向长时间运行的云应用,根据负载历史记录生成 r-b 曲线以描述存储和网络利用率,基于动态规划为每类负载自动生成限速参数。在保障处理负载的性能满足 SLO 的约束下,通过对自动化设置存储和网络限速参数,调度并整合负载以最小化处理负载的服务器数量,从而提高资源利用率并减少能耗。最后,设计并实现了原型系统,实验结果表明,提出的方法能够保障云应用性能,减少运行服务器数量,并具有较好的可扩展性。

关键词: 动态限速; 负载调度; 云应用; 性能保障; 资源管理
中图分类号: TP393 **文献标识码**: A **文章编号**: 1673-629X(2019)03-0051-04
doi: 10.3969/j.issn.1673-629X.2019.03.010

Workloads Scheduling Approach for Cloud Computing Applications Based on Dynamic Rate Limit

LIU Yong-bo[1],ZHOU Bo[1],LI Ya-qiong[1],LI Shou-chao[1],SONG Yun-kui[2]
(1. Jiangsu Hoperun Software Company,Nanjing 210012,China;
2. Institute of Software,Chinese Academy of Sciences,Beijing 100190,China)

Abstract: In cloud computing environment,applications face a large scale of workloads. Cloud service providers provide shared computing,network,and storage resources to multiple customers to maximize resource utilization and reduce overall energy consumption,thereby reducing the operating costs of data centers and providing users with well performance assurance. The address this issue,we propose a dynamic rate limit-based workloads scheduling approach for cloud applications. For long-running cloud applications,r-b curves are generated based on load history to describe storage and network utilization,and speed limit parameters are automatically generated for each type of load based on dynamic programming. Under the constraint of SLO (service level object) to ensure the performance of the processing load,by setting storage and network speed limit parameters for automation,scheduling and integrating the load,the number of servers to process the load is minimized,thus improving resource utilization and reducing energy consumption. Finally,we have designed and implemented a prototype system. The experiment demonstrates that the approach proposed can guarantee the performance of cloud applications and reduce the number of running servers with well scalability.

Key words: dynamic rate limit; scheduling workloads; cloud computing applications; performance guarantee; resource management

0 引 言

在云计算环境中,云服务提供商为多个客户提供共享的计算、网络和存储资源以最大化资源利用率,降低总体能耗,从而减少数据中心的运营成本,同时保障良好性能,如满足请求处理延迟时间,以提升客户满意度。客户通常定义服务水平目标(service level object,SLO)以描述处理负载的性能要求,比如“80% 的请求必须在 100 ms 内完成”。那么,需要具有高效的负载调度方法,服务器在满足性能目标的前提下,能够处理多样化的负载。为了应对整合负载所带来的网络拥塞问题,云服务提供者和客户通常会达成限速协议,客户静态设定限速规则,服务提供商则进行相应的优化,以实现性能最大化。

当前的负载调度方法是,为客户预先保留一定数

收稿日期: 2018-04-23 **修回日期**: 2018-08-24 **网络出版时间**: 2018-12-20
基金项目: 国家自然科学基金(61602454);南京市高端人才团队引进计划(10072090)
作者简介: 刘永波(1982-),男,工程师,研究方向为分布式系统。
网络出版地址: http://kns.cnki.net/kcms/detail/61.1450.TP.20181219.1542.068.html

量的资源,或者以尽最大努力的方式处理负载。文献[1-2]提出了BCLL-Min-Min算法以满足带宽需求约束。文献[3]提出一种在线式负载调度算法以最小化数据中心的电费,建立数据中心的电费模型,形式化为随机优化问题,求解得到负载调度策略。文献[4]提出一种智能电网环境下家庭可控负载优化调度策略,对负载进行分类并划分优先级,对负载进行通断调度,以提高能源利用效率。文献[5]提出一种流媒体集群服务器的负载调度策略,运用多级模糊系统和人工神经网络对用户请求响应延时进行模糊预测。文献[6-7]动态评估作业在截止时间内完成所需要的Map和Reduce计算资源数量,动态地增加或减少独立虚拟机的方式来调整CPU资源。文献[8]建立了负载双层矩形域模型,基于贪婪策略给出了多项式时间近似算法,对数据传输阶段进行负载排序调度,使GPU性能达到全局最优或近似最优。文献[9]组建信息传输负载均衡性调度聚合时机的调度机制,完成对海量信息传输负载均衡性调度。文献[10]及时收集释放空转资源降低资源能耗浪费,实现了节点负载度量、任务度量、负载整合算法,并测算出节点自适应负载阈值。Silo[11]设计限速方案以保障网络延迟,pClock[12]设计限速方案以保障存储延迟,文献[13]采用性能预测的方法动态调整等待处理的负载。Pythia[14]监测运行过程中SLO是否发生冲突,动态学习适合的迁移行为。

然而,面对云计算环境下应用的大量负载,难以实现通过预留资源来满足处理延迟,尤其是,短期突发性负载会对处理延迟产生显著影响。同时,难以为不同的负载类型设置合理的限速参数。针对该问题,面向长时间运行的云应用,提出一种基于动态限速的云应用负载调度方法,在保障处理负载的性能满足SLO的约束下,将负载调度并整合到目标服务器,以最小化处理负载的服务器数量。

1 基于动态限速的云应用负载调度方法

文中刻画负载,描述其对处理延迟的影响,通过对存储和网络进行限速,并设置负载的优先级,在满足处理延迟的条件下,自动化地减少服务器数量。

(1) $r-b$ 曲线生成。

生成 $r-b$ 曲线以描述负载处理速率 r 与令牌桶容积 b 之间的关系。当请求到达时,令牌添加到令牌桶中,如果令牌桶中有足够的空间来添加令牌,即不超过令牌桶大小为 b,则允许继续处理请求。否则,请求就会排队等待,直到令牌桶中有足够的空间。令牌以速率 r 不断地从桶中流出,空间逐渐变得可用。对于给定的 r 值,通过重放具有速率 r 以及无限大小令牌桶在任意时间点的执行轨迹,计算得到不需要排队的请求数量 b。输入 r 与输出 b 构成 $<r, b>$ 元组作为点,连接形成分段 $r-b$ 曲线。对 r 值进行标准化处理(例如,网络流量除以网络带宽),那么,$r=1.0$ 表示负载占用了所有带宽资源。传输的数据量取决于请求类型(例如,读/写),分别生成不同的 $r-b$ 曲线。

(2) 限速参数设定。

使用网络微积分方程进行计算,由于在服务器上排队而导致的处理延迟,对于优先级 p 的负载,处理延迟的上限为:

$$\frac{\sum_{p_j \geqslant p} b_j}{1-\sum_{p_j \geqslant p} r_j} \leqslant \mathrm{SLO}_p \tag{1}$$

其中,$<r_j, b_j>$ 是负载 j 的限速设置,b_j 是负载 j 的令牌桶大小,r_j 是负载 j 的处理速率;p_j 是负载 j 的优先级,其高于或等于 p;SLO_p 是与优先级 p 关联的SLO。

进而可以得到:

$$\sum_{p_j \geqslant p} b_j + \sum_{p_j \geqslant p} r_j \times \mathrm{SLO}_p \leqslant \mathrm{SLO}_p \tag{2}$$

使用分段线性凸函数 $r-b$ 曲线,可以将 b_i 表示为 r_i 的函数,进而利用线性规划方法求解得到每个限速元组 $<r_j, b_j>$,满足约束条件:

$$\sum_{p_j \geqslant p} r_j \leqslant 1 \tag{3}$$

每个负载关联 $r-b$ 曲线,当新的负载调度到该服务器时,动态重新计算现有负载共享该服务器的限速设置。

(3) 服务器选择。

通过线性规划求解,将负载分发给SLOs能够满足的服务器,采用首先匹配的策略。在通常情况下,大多数服务器几乎都是满载的,所以新负载不能分发给几乎满负载的服务器。因此,文中提出了快速首次匹配方法,跟踪每个服务器上配置的速率总和,跳过将负载放到接近满负载的服务器,避免了不必要的运行线性规划计算的过程。

提出的方法能够根据服务的实际资源使用状态,动态调整限速参数,即负载处理速度 r 和令牌桶容积 b;在保障处理负载的性能满足SLO的约束下,将负载调度并整合到目标服务器,以最小化处理负载的服务器数量;能够满足在同一台服务器上,不同类型负载对于处理延迟的要求。

2 负载调度系统设计与实现

如图1所示,文中提出的负载调度系统由五个主要组件构成。

(1) $r-b$ 曲线生成器:根据处理负载的历史记录生成 $r-b$ 曲线,描述负载的存储和网络利用率,并根

据客户需求定义 SLO;

(2) 部署器: 标识可以分发负载的候选服务器;

(3) 优化器: 为每个负载配置<r, b>限速参数,并决定在哪个服务器上放置负载来满足处理延迟要求;

(4) 延迟检查器: 确定负载的候选位置和<r, b>元组是否能够满足负载的 SLO 要求;

(5) 实施器: 配置适当的存储和网络限速,并将负载分配给服务器。

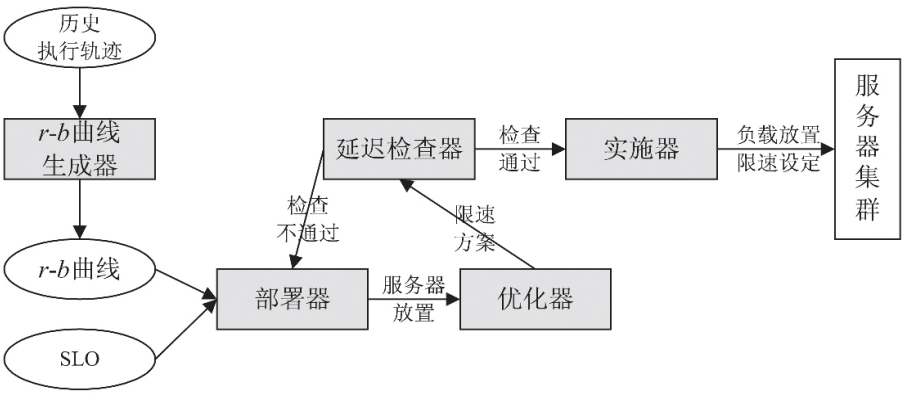

图 1　负载调度系统设计

文中提出的负载调度方法的执行流程如下:

(1) $r-b$ 曲线生成器根据负载类型的历史执行记录生成网络或内存的 $r-b$ 曲线(即线性分段函数 $b=f(r)$),并将用户定义的负载 SLO 要求,一同发送给部署器;

(2) 部署器选择可以分发负载的服务器,并生成候选服务器列表,即存在较充足资源的服务器;

(3) 优化器使用线性规划方法计算服务器上共存的各类负载的<r, b>元组参数,并将计算结果发送给延迟检查器:

(4) 延迟检查器检测在候选服务器以及<r, b>元组是否能够满足用户所定义的 SLO 要求,如果能满足则将候选服务器以及<r, b>元组信息发送给实施器,否则重新发回部署器以生成新的方案;

(5) 实施器将负载发送给选定的候选服务器,并设置<r, b>元组参数。

3　实验结果与评价

3.1　实验环境

文中搭建由 11 台服务器构建的集群,其中包括 1 台部署客户端和 10 台部署服务端,每台服务器配置了两个英特尔 Xeon E5-2680 处理器,64 GB 的 DRAM,以及一个 300 GB 的 Intel 710 SSD,通过 1 Gbps 网络连接。每台机器运行 64 位 Ubuntu 14.04,并通过标准 kvm 包(qemu-kvm-1.0)运行虚拟机,使用标准的 NFSv3 服务器和客户端,提供远程存储访问。在服务器端存储文件,客户端发送请求访问文件,访问文件的模式分为读取、增加、删除、修改文件等几种,访问文件的大小随机产生,这样在服务器端产生各种访问文件的执行轨迹。随机选取一半执行轨迹用于训练产生 $r-b$ 曲线,另一半执行轨迹在服务器端重放以验证所提出的负载调度方法可以保证响应时间,并减少服务器数量。

3.2　减少服务器数量

文中提出的方法旨在通过设置每个负载的限速参数,实现在满足负载处理延迟的情况下,减少服务器数量。提出的限速方法与当前限速方法比较如下:

(1) 扩展平均带宽: 统计计算历史带宽的平均值,将 r 值设置为该平均值的 1.5 倍;

(2) $r-b$ 曲线拐点: 观察选取 $r-b$ 曲线拐点,作为选取的参数以最小化 r 值与 b 值之和。

实验中设置的负载处理延迟分别为: 100 ms、150 ms、250 ms、500 ms、1 000 ms,计算使用的服务器数量。如图 2 所示,与现有方法相比,文中提出的方法需要更少的服务器数量。

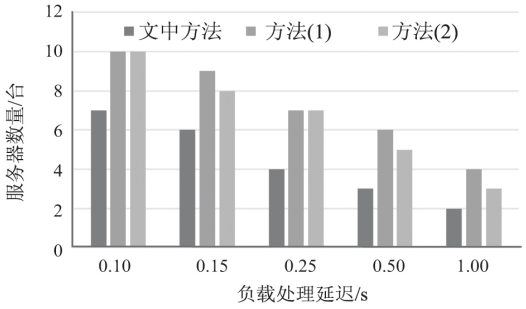

图 2　服务器数量变化

3.3　计算的可伸缩性

随着集群规模的增长,所提出方法计算能力的可伸缩性如图 3 所示。提出的快速首先满足策略由于跳过几乎满负载的服务器,比典型的首先满足策略具有更好的效果。

3.4　负载离开的影响

以上实验中,假定负载随时可以到达,而在实际中,负载也会随时离开系统,留下一些空白来放置将来

的负载。为了模拟这种情况,在实验中,负载随机到达并离开系统。如图 4 所示,提出的方法能够更好地处理负载,减少了 50% 服务器。

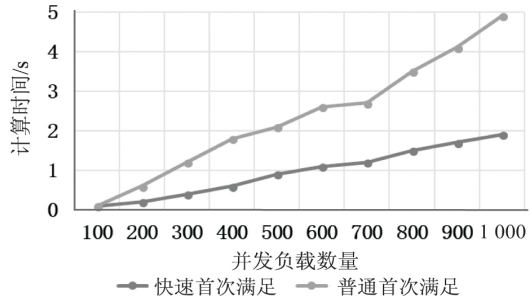

图 3 计算时间比较

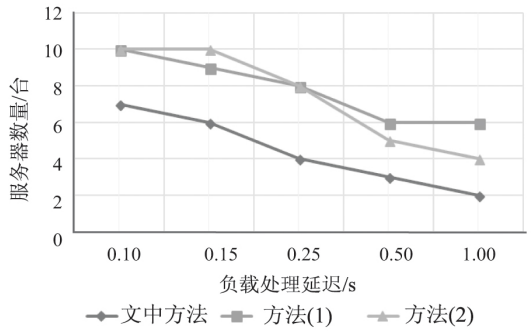

图 4 负载离开的影响

4 结束语

云计算环境下,应用负载规模巨大,云服务提供商为多个客户提供共享的计算、网络和存储资源以最大化资源利用率,降低总体能耗,从而减少数据中心的运营成本,同时需要为用户提供良好的性能保障。针对该问题,提出一种基于动态限速的云应用负载调度方法。实验结果表明,该方法能够保障云应用性能,减少运行服务器数量,具有较好的可扩展性。

参考文献:

[1] 郑 卉,郭 平,李 琪,等. 基于带宽约束的云计算负载调度算法[J]. 西南师范大学学报: 自然科学版,2014,39(7):121-128.

[2] 陈 斌,甘茂林,李 娟. 一种基于负载均衡的云资源调度方法[J]. 计算机技术与发展,2017,27(6):51-55.

[3] 窦 晖,齐 勇,王培健,等. 一种最小化绿色数据中心电费的负载调度算法[J]. 软件学报,2014,25(7):1448-1458.

[4] 常海松,刘家豪,刘天晓. 基于优先级的可控负载调度算法[J]. 通信电源技术,2017,34(1):64-65.

[5] 王 芳,汪 伟. 基于响应延时预测的流媒体集群服务器负载调度策略[J]. 计算机与现代化,2013(5):108-111.

[6] 秦 军,童 毅,戴新华,等. 基于 MapReduce 数据密集型负载调度策略研究[J]. 计算机技术与发展,2015,25(4):48-52.

[7] 刘立帮,黄 刚. 一种多层网络下动态负载均衡算法[J]. 计算机技术与发展,2017,27(2):51-55.

[8] 孙景昊,邓庆绪,孟亚坤. GPU 上两阶段负载调度问题的建模与近似算法[J]. 软件学报,2014,25(2):298-313.

[9] 王丽珍. 网络资源信息传输负载均衡性调度仿真[J]. 计算机仿真,2017,34(12):366-369.

[10] 张宝婷,芮建武,周 鹏,等. 基于 CoreOS 面向负载整合的集群调度研究[J]. 计算机系统应用,2017,26(11):67-75.

[11] JANG K,SHERRY J,BALLANI H,et al. Silo: predictable message latency in the cloud [C]//Proceedings of ACM SIGCOMM. [s. l.]: ACM,2015:435-448.

[12] GULATI A,MERCHANT A,VARMAN P J. pClock: an arrival curve based approach for QoS guarantees in shared storage systems [C]//Proceedings of ACM SIGMETRICS international conference on measurement and modeling of computer systems. [s. l.]: ACM,2017:13-24.

[13] PARK N,AHMAD I,LILJA D J. Romano: autonomous storage management using performance prediction in multi-tenant datacenters [C]//Proceedings of the third ACM symposium on cloud computing. San Jose,CA,USA: ACM,2012:14-21.

[14] ELMORE A J,DAS S,PUCHER A,et al. Characterizing tenant behavior for placement and crisis mitigation in multitenant DBMSs [C]//Proceedings of ACM SIGMOD international conference on management of data. [s. l.]: ACM,2013:517-528.