计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20151043
54(3):597-608, 2017

面向微服务架构的容器级弹性资源供给方法

郝庭毅[1,2,3]　吴　恒[1]　吴国全[1,2]　张文博[1]

[1](中国科学院软件研究所软件工程技术中心　北京　100190)

[2](计算机科学国家重点实验室(中国科学院软件研究所)　北京　100190)

[3](中国科学院大学　北京　100049)

(haotingyi13@otcaix.iscas.ac.cn)

Elastic Resource Provisioning Approach for Container in Micro-Service Architecture

Hao Tingyi[1,2,3], Wu Heng[1], Wu Guoquan[1,2], and Zhang Wenbo[1]

[1](*Technology Center of Software Engineering*, *Institute of Software*, *Chinese Academy of Sciences*, *Beijing* 100190)

[2](*State Key Laboratory of Computer Science* (*Institute of Software*, *Chinese Academy of Sciences*), *Beijing* 100190)

[3](*University of Chinese Academy of Sciences*, *Beijing* 100049)

Abstract　As a logical abstraction of physical resources, container-based virtualization has been adopted widely in cloud computing environment for elastic resource provisioning, which is lower overhead and potentially better performance. Nowadays, more and more enterprises seek to move large-scale Internet-based applications with micro-service architecture into the container-based infrastructure, and focus on efficient resource management. Unfortunately, many existing approaches were restricted by physical machine or virtual environment, the resources are hard to be elastically or timely provisioning. Therefore, Internet-based applications may suffer from frequent service-level agreement(SLA) violations under flash-crowd conditions. To address this limitation, this thesis proposes a quality of service(QoS) sensitive resource provisioning approach for containers in micro-service architecture based on the feed-forward control. We employ a performance model based on queuing theory. Firstly, we capture the relationship among workload, resource utilization and response time. Secondly, we predict the response time with fuzzy federal adaptive Kalman filtering based on the feed-forward control, and if the prediction result is against pre-defined QoS, elastic resource scheduling process is triggered. Experimental results based on CloudStone show that the feed-forward algorithm converges quickly. The prediction result of the response time has only maximum error of 10%, and is more effective and accurate compared with existing approaches. Furthermore, our approach can effectively protect resource provisioning for flash-crowds workload.

Key words　container virtualization; fuzzy adaptive Kalman filtering; elastic resource provisioning; micro-service architecture; flash-crowds

摘　要　容器作为物理资源的逻辑抽象,具有资源占用少、资源供给快等特点,适合工作负载突变的互联网应用模式,特别是面向微服务架构的新型服务范型.已有工作受限于物理机和虚拟化环境,或资源难

收稿日期:2015-12-09;修回日期:2016-04-01

基金项目:国家自然科学基金项目(61472407,61363003)

This work was supported by the National Natural Science Foundation of China (61472407, 61363003).

通信作者:吴恒(wuheng09@otcaix.iscas.ac.cn)

以弹性供给或资源供给时效性较差,难以应对负载突变(flash-crowds)场景.针对此问题提出了一种服务质量(quality of service, QoS)敏感的、基于前馈的容器资源弹性供给方法,该方法采用排队论刻画工作负载、资源利用率和响应时间的关联关系,构建应用性能模型.其中,响应时间采用模糊自适应卡尔曼滤波进行预测(前馈控制器),预测结果违背 QoS 是触发资源弹性供给的依据.基于 CloudStone 基准的实验结果显示,前馈控制器具有快速收敛的特点,对响应时间的预测误差小于 10%.在 flash-crowds 场景下,相对于已有方法可有效保障应用的 QoS.

关键词 容器虚拟化; 模糊自适应式卡尔曼滤波; 弹性资源供给;微服务架构;突发性负载

中图法分类号 TP319

微服务架构(microservice)[1]体现了互联网应用的设计思想,其核心理念是细粒度模块划分、服务化接口封装、轻量级通信交互,具有 2 点优势:1)模块自治性强,能很好满足互联网应用诉求变化快、模块独立更新的需求;2)模块扩展性好,能很好满足互联网应用用户难预测、资源动态分配的需求.其中,前者涉及的是应用开发和设计范畴;后者强调的是应用运行和维护问题,也是本文关注的重点.根据 Gartner 报告,微服务本身具有良好的扩展性,正逐渐成为构造互联网应用的主流架构模式,但从运行和维护的视角来看,应对典型的互联网突变负载(flash-crowds)场景、保障应用服务质量[2](quality of service, QoS)依旧面临挑战(服务质量是指应用软件对时间要求的满足程度,响应时间是其重要的度量指标之一,比如用户服务质量为 5 s,即表示用户从请求发起到请求响应的时间间隔不应超过 5 s).

近年来,轻量级容器技术应运而生,其本质是模拟进程运行环境,具有资源占用少、应用启动快等特点[3],其正逐步成为支撑微服务运行的主流架构平台.容器具有秒级资源供给的特点,可很好地满足互联网应用负载突变对实时资源供给的需求.然而,已有方法或受限于物理机灵活性不够,资源难以弹性供给;或受限于虚拟机资源的供给分钟级时效性,通常只能适用于周期变化(time-of-day)的负载模式.如京东采用容器作为 2015 年“618”限时抢购活动,用户访问数增长率高于预期,导致部分应用构件出现“无响应”、“卡顿”等现象.其原因是应用性能模型的参数选择受限于训练集,缺少运行时自我调整的能力.其中,性能模型是评估应用资源需求、实现弹性供给的依据.

本文提出了一种基于前馈控制、面向负载突变场景的容器资源按需供给方法,该方法采用排队论刻画应用负载、资源消耗和响应时间的关联关系,构造应用性能模型.同时,使用模糊自适应卡尔曼滤波实现模型参数的运行时调整,进行应用响应时间预测.最后,以响应时间是否违约 QoS 作为资源弹性供给的依据.

本文的贡献主要有 3 点:

1) 提出一种基于容器的应用弹性供给框架,利用容器轻量级特点,提高了资源供给的实效性.

2) 提出一种基于预测的应用突变负载估算方法,利用模糊自适应卡尔曼滤波快速收敛的特点,提高了突变负载的预测准确性和资源供给的有效性.

3) 实验结果显示,本方法对响应时间预测的误差率小于 10%.在突发负载的场景下,相比已有方法,可以有效保障应用的服务质量.

1 相关工作

已有工作主要面向物理机和虚拟机场景实现资源的弹性供给.文献[4-10]针对物理资源难以快速供给的特点,或采用容量规划,以服务质量作为约束条件,估算应用的峰值资源需求;或采用准入控制机制,根据资源供给量反推应用可承受的峰值负载,通过拒绝服务策略来保障应用的服务质量.如 Cherkasova 等人[9]提出的基于准入控制机制,建立会话与吞吐量的损失模型,推演资源的供给需求.又如 Robertsson 等人[10]提出的基于线性模型的资源需求方程组,在保障应用质量的前提下估算资源需求的峰值,从而达到资源按需供给的目的.文献[11-17]考虑虚拟机性能开销因素,采用模型驱动的方法刻画虚拟化环境下应用的资源需求变化,并以此作为资源提供的依据.这些方法通常采用增强学习(reinforcement learning)、统计学习(statistical learning)等机制进行模型参数训练.如 Karlsson 等人[15]提出基于性能隔离的方法分析每个服务实例的资源需求,分别对每种服务构建性能变化模型,从而进行自适应资源供给;文献[18-21]考虑虚拟机之间性能相互干扰问题,采用统

计机器学习(statistical machine learning)、模糊控制(fuzzy control)、概率论(probability theory)等方法刻画出虚拟机之间性能相互干扰对应用资源供给的影响,并以此作为应用资源供给的依据.如 Bodık 等人[13]提出的基于机器学习的方式,通过历史数据集合对应用模型的性能参数进行规则训练,并根据得到的资源供给规则进行资源动态调整.然而,由于虚拟机资源供给是分钟级的时效性,因此上述方法通常只适用于负载随时间周期性变化的应用场景.

综上所述,已有方法或受限于物理机灵活性不够,资源难以弹性供给;或受限于虚拟机资源的供给时效性,难以满足互联网应用负载突变的场景.因此,本文引入容器作为互联网应用的载体,利用其秒级资源供给的特点,满足负载突变的需求.具体而言,本文提出了一种面向微服务架构的容器级弹性供给方法,基于卡尔曼滤波收敛快、无需保存历史数据的特点,对服务响应时间进行预测,并根据预测结果的实现资源的弹性供给,弥补已有研究工作难以应对突发负载的不足.

2 方法总体框架

总体框架如图 1 所示,本方法以每个应用构件的资源使用率及负载等参数构建应用性能模型,利用自适应卡尔曼滤波器对服务响应时间进行预测,并通过模糊逻辑对预测模型进行实时调整,最终以服务质量是否违约作为容器调度标准,达到资源弹性供给的目的.

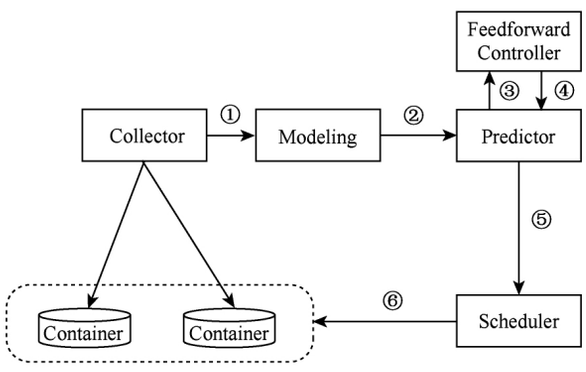

Fig. 1 Approach overview

图 1 方法总体框架及流程

系统方法流程可描述为:

① 数据采集器采集每个容器的负载以及 CPU、内存等系统资源使用率.

② 性能建模器基于排队论,以步骤①中资源使用率作为基准,构建应用性能模型,刻画负载和响应时间的关联关系,详见 3.1 节.

③ 响应时间预测器通过卡尔曼滤波器对性能模型参数进行运行时估算,估算是以满足预测和实测响应误差预期为收殓条件,详见 3.2.1 节.

④ 前馈控制器通过模糊控制器分析残差均值和方差,得到卡尔曼滤波器控制参数的前馈调整值,详见 3.2.2 节.

⑤ 容器调度器判断响应时间的预测值是否违背了应用服务质量,并根据调度算法进行调度,详见第 4 节.

⑥ 执行容器扩展、收缩或者迁移后,继续执行步骤①,形成方法闭环.

3 响应时间预测模型

负载是影响应用资源需求的主要因素,本节通过 Jackson 网络排队模型构建负载与 QoS 的关联关系,并以 QoS 是否违约作为资源供给的依据.具体而言,本节首先利用 Jackson 网络排队论构建负载、资源使用率与响应时间的性能模型,然后采用卡尔曼滤波算法对模型中的未知参数进行预测,并通过模糊逻辑(前馈控制器)校正滤波器中的控制参数,以达到提高响应时间预测准确性,保障 QoS 的目的.在本节,QoS 特指应用的响应时间.

3.1 基于 Jackson 网络排队的应用性能模型

Jackson 开环网络是适合微服务架构的应用性能模型,其原因包括[22]:

1) 微服务架构中应用构件是相互独立的,模块之间通过消息总线进行通信,没有状态信息存在,满足 Jackson 网络排队模型下节点(应用构件即节点)相互独立、满足指数分布的约束;

2) 微服务架构下应用构件之间通过消息进行交互,满足 Jackson 网络是开环、节点输入符合泊松分布的假设;

3) 应用构件在处理请求后,可选择进入下一个节点或者离开网络.

用户请求会在节点中跳转,经过相关应用构件的处理,最后响应给用户.当某个应用构件存在多个实例时候,采用轮循调度(round-robin scheduling)策略.如图 2 所示,为了区分同一应用构件的不同实例,本文定义:f 为用户请求流,j 是应用构件,i 是

应用构件 j 的第 i 个实例. f,j,i,三者的关系可描述为:1 个用户请求 f 将会流经多个应用构件 j_1, $j_2,\cdots,j_n$,每个应用构件 j 都含有多个实例,如 $j(i_k)$ $1\leqslant k\leqslant m$ 表示应用构件 j 的第 k 个实例,共有 m 个实例,每个实例都运行在容器中. 由于应用构件的资源偏好不同(如 CPU 密集集型、I/O 密集集型),导致容器出现的偏好资源不同,定义偏好资源为容器 CPU、内存、磁盘 I/O 中使用率最高的资源;$u_j\in[0,1)$是应用构件 j 的偏好资源使用率;u_{0j}是指当应用构件 j 在无负载情况下的偏好资源使用率;γ_{ji}是指应用构件 j 的第 i 个容器的并发数,即每秒到达的请求数,满足泊松到达过程;T_{ji}是指应用构件 j 的第 i 个容器的服务处理时间;T_j 是指应用构件 j 的平均服务处理时间;d 是指用户请求流 f 的总网络传输时间;B 是指服务流 f 的响应时间;τ_j 是服务 j 的负载与资源使用率的相关系数. 根据 Jackson 网络流量方程及网络性能方程有[23]:

$$u_j=\sum_i(u_{0j}+t_j\times g_{ji}\times T_{ji}),\tag{1}$$

$$B=d+\sum_j\frac{T_j}{1-u_j/m_j},\tag{2}$$

其中,u_j,u_{0j},γ_{ji},B 是通过监测获取的,τ_j 是根据历史数据给出的经验值,T_{ji},d 是难以监测的,需要通过预测进行估算. 所谓弹性供给是指响应时间 B 在相对固定的取值区间前提下,应用的资源需求. 可见,T_{ji},d 是进行自适应资源供给的关键要素.

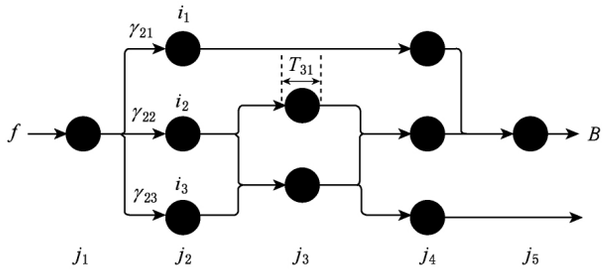

Fig. 2 Micro-service architecture network queuing model

图 2 微服务架构网络排队模型

3.2 响应时间 T 的预测值

卡尔曼滤波算法是由 Kalman[24] 在 1960 年提出的一种最优线性状态估计方法,常被用于轨迹跟踪预测领域. 其优点是采用递归的方法来解决线性滤波问题,只需当前的测量值和前一个采样周期的估计值就能够进行状态估计. 其缺点主要有 2 点[25]:

1) 需要定义精确的状态转移矩阵(将目标从 $k-1$ 状态转移到 k 状态),这个矩阵是符合预测目标轨迹变化规律的数学模型,这个模型的精确程度将影响预测的准确性,偏离严重时可能会导致滤波发散.

2) 由于递归的特性,其对过去所有观测值都给予以均一的权值,即对新老数据给予相同的置信度,这样随着时间的推移,采集到的数据越来越多,使得算法失去修正能力,即数据饱和.

由于每个应用构件的负载是非线性无规律的,很难根据卡尔曼滤波的要求定义一个精确的状态转移矩阵,另一个方面当负载突发性变化时,由于历史数据的影响会导致滤波因“惯性”发散,所以,传统卡尔曼滤波无法适应时变结构的微服务架构. 而基于模糊逻辑的自适应卡尔曼滤波算法是采用前馈控制的方式,根据滤波预测值实时对滤波模型参数进行调整,其方法原理如图 3 所示,多篇文献论述了该算法在实时轨迹追踪系统中的有效性[26-27].

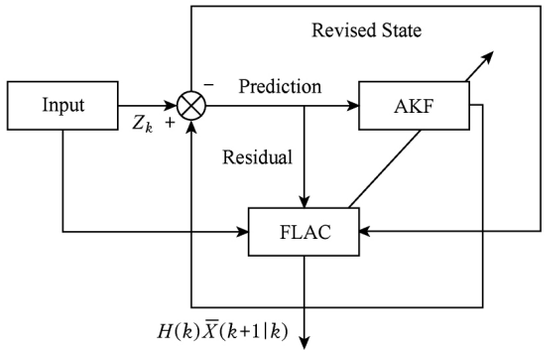

Fig. 3 Fuzzy adaptive Kalman filter

图 3 模糊自适应卡尔曼滤波器原理图

3.2.1 自适应卡尔曼滤波算法对响应时间的预测

自适应卡尔曼滤波(adaptive Kalman filtering, AFK)原始方程如下:

$$\boldsymbol{X}(k+1)=\boldsymbol{F}(k)\boldsymbol{X}(k)+\boldsymbol{Q}(k),\tag{3}$$

$$\boldsymbol{Z}(k)=\boldsymbol{H}(k)\boldsymbol{X}(k)+\boldsymbol{R}(k),\tag{4}$$

其中,$\boldsymbol{X}(k)$是预测矩阵,其值为$(T_j,d)^{\mathrm{T}}\ \forall j$,代表服务处理时间与总时延的矩阵;$\boldsymbol{Z}(k)$为 $\boldsymbol{X}(k)$的状态矩阵;$\boldsymbol{H}(k)$负责将可观测值的多维向量转换到待预测值的多维向量,其值为$(u_j,u_{0j},\gamma_{ji},B)^{\mathrm{T}}\ \forall i$的状态描述向量,是由应用实例的资源利用率、负载与响应时间的所构成的矩阵;$\boldsymbol{Q}(k)$是过程激励噪声协方差矩阵,其符合 $\boldsymbol{Q}(k)\sim N(0,\boldsymbol{Q})$的高斯分布;$\boldsymbol{R}(k)$是测量噪声协方差矩阵,其符合 $\boldsymbol{R}(k)\sim N(0,\boldsymbol{R})$的高斯分布,一般认为应将这 2 个噪声矩阵设置为零均值白噪声[25],但是负载变化往往是不确定的,如负载突变场景,所以为了使系统的弹性资源供给具有实

时性,过程激励噪声协方差矩阵以及测量噪声协方差矩阵应该随时间自适应调整,现将噪声矩阵设为

$$\boldsymbol{Q}(k)=T\boldsymbol{Q}, \tag{5}$$

$$\boldsymbol{R}(k)=U\boldsymbol{R}, \tag{6}$$

其中,T 和 U 为时变的调整值,可得到预测方程如下:

$$\overline{X}(k+1|k)=\boldsymbol{F}(k+1|k)\overline{X}(k|k), \tag{7}$$

$$\overline{Z}(k+1|k+1)=\boldsymbol{H}(k)\overline{X}(k+1|k), \tag{8}$$

其中,$\overline{X}(k+1|k)$ 是各个服务处理时间与请求总延迟时间的预测值,$\overline{Z}(k)$是预测值的状态估计值,可以将 $\overline{X}(k+1|k)$的预测值带入式(1)(2),得出滤波器对响应时间的预测值.

修正的状态估计方程如下:

$$\overline{X}(k+1|k+1)=\overline{X}(k+1|k)+K(k+1)\times(Z(k+1|k+1)-\overline{Z}(k+1|k+1)), \tag{9}$$

$$K(k+1)=P(k+1|k)\boldsymbol{H}(k)^{\mathrm{T}}(\boldsymbol{H}(k)\times P(k+1|k))\boldsymbol{H}(k)^{\mathrm{T}}+T(k+1)\boldsymbol{R})^{-1}, \tag{10}$$

$$P(k+1|k+1)=(\boldsymbol{I}_n-K(k+1)\boldsymbol{H}(k))P(k+1|k), \tag{11}$$

$$P(k+1|k)=\boldsymbol{F}(k+1)P(k|k)\boldsymbol{F}(k+1|k)^{\mathrm{T}}+U(k+1)\boldsymbol{Q}, \tag{12}$$

其中,式(9)代表预测值的修正值,代表滤波器认为的各个应用构件的服务处理时间与请求总延迟的真实值.本文通过反馈控制实时调整噪声和过程激励矩阵,以达到自适应修正滤波参数的目的.式(9)中 $Z(k)-\overline{Z}(k)$定义为残差 r,代表系统模型依赖测量值的程度,其值越大则系统模型对测量值的依赖越大,这时说明系统负载可能处于突发性变化的状态,滤波器无法对服务处理时间和总延迟进行准确预测,需要修正滤波参数.

3.2.2 前馈控制器对预测模型的调整

基于 3.2.1 节所述,判断滤波器是否需要更新的依据就是监测残差,理想情况下残差为零均值白噪声,即滤波器可以完美自适应,如果残差不为零均值白噪声,则说明滤波器预测出现误差.由于残差方差、残差均值与 $\boldsymbol{Q}$ 和 $\boldsymbol{R}$ 相关,可以通过估计残差方差与均值,然后进行模糊推理,最后调整 U 和 T 的值,已达到使卡尔曼滤波算法适应时变结构的目的.残差方差计算方程如下:

$$P(r)=\boldsymbol{F}(k+1|k+1)(\boldsymbol{H}(K)P(k|k)\boldsymbol{H}(k)^{\mathrm{T}}+\boldsymbol{Q})\boldsymbol{H}(k)^{\mathrm{T}}+\boldsymbol{R}, \tag{13}$$

所以,设计一个模糊函数来不断监视残差方差 $P(r)$和均值的变化,然后根据模糊规则调整 T 和 U,以改变噪声矩阵,从而对卡尔曼滤波器的方误差矩阵见式(11)进行调整,使其一直执行最优估计,以满足时变需求.

本文采用 TS 模糊逻辑系统,对残差方差及均值建立三角形隶属度函数及模糊规则,如图 4 所示.例如,如果残差方差越来越大,均值也渐渐远离 Zero,则应该减小过程激励噪声 T 并增加测量噪声 U.从而建立模糊逻辑规则表,如表 1 所示,表 1 中的 Zero 代表 T 和 U 不需要变化,Small 代表增加 T 减小 U,Large 代表减小 T 增加 U,Medium 代表同时增加 T 和 U.根据以上原则,通过 MATLAB 仿真出一些列系统的误差曲线,并将其和常规卡尔曼滤波器的误差曲线进行横向对比,从而推断该组数据的可行性及有效性.再将每次实验结果与前一次实验结果进行纵向对比,以确定效果更好的线性组合.最后经过 100 组仿真实验,确定了模糊自适应控制器(fuzzy logic adaptive controller, FLAC)的输出规则,这里仅列举 2 个重要的 FLAC 规则:

1) 当且仅当残差方差为 Small、残差均值为 Zero 时,$T=P(r)\times 0.3+0.8$,$U=-P(r)\times 0.2+1.9$;

2) 当且仅当残差方差为 Large、残差均值为 Small 时,$T=-P(r)\times 0.5+0.6$,$U=P(r)\times 0.1+1.4$.

根据上述 FLAC 规则动态调整滤波器的参数以保障预测结果的有效性.

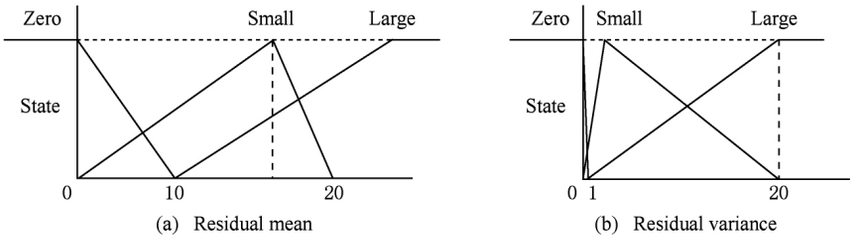

Fig. 4 Residual mean and residual variance membership function

图 4 残差均值和残差方差隶属度函数

Table 1 Fuzzy Logic Rule

表 1 模糊逻辑规则表

Residual	Mean Zero	Mean Small	Mean Large
Variance Zero	Small	Zero	Zero
Variance Small	Small	Zero	Large
Variance Large	Large	Large	Medium

4 容器调度策略

通过第 3 节对方法的分析,我们可以根据负载及资源使用情况,对容器进行实时调度,以保障平稳的输出响应时间. 如算法 1 行①～⑧所示,定义卡尔曼滤波算法中的噪声参变量为全局变量 T 和 U(行①);定义服务实例中资源使用量的最大值、最小值及最大响应时间(行②);定义卡尔曼滤波函数 AKF,用于预测响应时间(行③);定义模糊函数 $FLAC$,用于对 T 和 U 发送前馈控制信息,自适应调节 AKF 的模型参数(行④);定义函数 $ResponseTime$,以 AKF 的预测输出值为参数,计算响应时间(行⑤);定容器迁移函数 $Migrate$、容器扩展函数 $Expand$、容器收缩函数 $Contract$(行⑥～⑧). 系统的输入如 3.1 节中给出的 2 个未知量及 4 个已知量(行⑨);行⑩～⑬首先根据函数 AKF 计算响应时间预测值及 3.2.2 节中定义的残差均值及方差,然后将这 2 个值带入函数 $FLAC$,对函数 AKF 参数进行前馈调节,最后计算出应用的平均响应时间. 算法 1 行⑭～⑲描述了容器调度策略,主要有 3 种:

1) 容器迁移. 其产生的主要原因是容器本身的资源使用并没有达到资源限制上限,而不是因为宿主机的资源总量将要到达限制上限,此时并不需要对容器进行扩展(算法 1 行⑭～⑮). 控制器只需要将服务名称汇报给调度器,调度器将根据资源类型及集群资源使用情况对容器进行迁移. 容器迁移过程大致如下,首先控制器将容器持久化成镜像,这一步将保存当前的应用状态,然后调度器发出调度命令,收到调度命令的控制器将根据调度命令,从镜像中产生 1 个新的容器,然后反馈给服务注册模块,以完成服务发现. 原有的容器将会被控制器删除,同样也汇报给服务注册模块,完成服务注销.

2) 容器扩展. 其产生的主要原因是容器本身某项资源率使用已到限制器的限制值,此时进行容器迁移并不能解决问题,为保障该容器的平均响应时间,必须对该容器进行扩展(算法 1 行⑯～⑰). 容器扩展的过程大致如下,首先调度器根据控制器的汇报信息,在当前集群中选择合适的节点发出创建容器命令,并将容器的配置信息传输给节点中的控制器,控制器创建新容器后将根据容器的网络地址及端口号自动进行服务注册.

3) 容器收缩. 其产生的主要原因是应用的各个实例的资源使用情况都低于估计值,此时需要裁剪该应用的实例数量(算法 1 行⑱～⑲). 容器收缩的过程大致如下,首先调度器根据控制器的汇报信息,判断该应用是否需要进行收缩,并将判断结果反馈给控制器,控制器收到调度命令后自动进行服务注销.

算法 1. 容器调度算法.

① global var T,U;

② const var $RESOURCE\_MAX\_LIMIT$,
$RESOURCE\_MIN\_LIMIT$,
$TIME\_MAX\_LIMIT$;

③ define func $AKF(X(i),Z(i))(Variance(P(r)),Average(P(r)),X(i))$;

④ define func $FLAC(Variance(P(r)),Average(P(r)))(T,U)$;

⑤ define func $ResponseTime(X(i))B$;

⑥ define func $Migrate(Service)$;

⑦ define func $Expand(Service)$;

⑧ define func $Contract(Service)$;

⑨ Input: $X(i)=(T_i,d)$, $Z(i)=(u_i,u_{bi},r_i,B)$;

⑩ var $E(P(r))$, $Avg(P(r))$, $\overline{X}_i:=AKF(X(i),Z(i))$;

⑪ var $U_0,T_0:=FLAC(E(P(r)),Avg(P(r)))$;

⑫ $T,U=U_0,T_0$;

⑬ var $B:=ResponseTime(\overline{X}_i)$;

⑭ if $\sum u_i>RESOURCE\_MAX\_LIMIT$ && $B<TIME\_MAX\_LIMIT$:
$Migrate(Service(i))$;

⑮ end if

⑯ if $B>TIME\_MAX\_LIMIT$:
$Expand(Service(i))$;

⑰ end if

⑱ if $u_i<RESOURCE\_MIN\_LIMIT$:
$Contract(Service(i))$;

⑲ end if

5 系统实现

系统架构如图 5 所示,分为 3 个部分:自动部署模块、资源弹性调度模块、服务注册与发现模块. 主要工作流程如下:

系统采用主从结构,用户在模板仓库中选择或创建应用所需配置,然后将一组配置模板传给应用部署器,应用部署器对配置模板进行容错验证、组合之后将应用配置信息传给全局调度器,全局调度器根据当前各个子节点的资源使用情况选择合适的子节点进行服务部署,并将服务信息持久化存储,子节点的控制器收到调度命令后,进行具体调度,另一方面,控制器也会实时收集服务的信息汇报给主控节点的调度器,当有新服务产生时,会自动向主控节点的存储器进行服务注册.

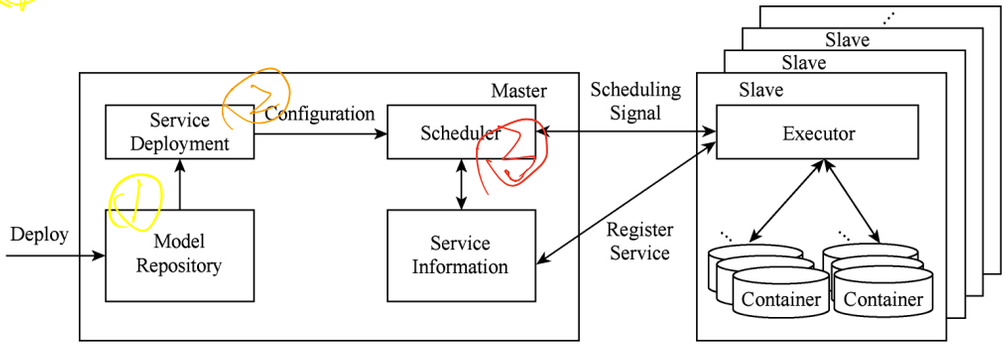

Fig. 5 System architecture

图 5 系统架构图

5.1 自动部署模块

自动部署模块由模板仓库与应用部署器组成,模板仓库中提供了大量可参考的部署模板及应用服务,用户可以根据服务需求选择具体应用及其版本号,当模板仓库不能满足用户服务需求时,用户可以根据模板格式自主创建模板.

应用部署器根据模板仓库传来的一系列模型信息,对服务所需的组件间的关联关系进行分析,并将分析结果持久化到配置文件中,这些信息将会决定各个服务的部署及启动顺序,也是服务注册与发现的基本依赖. 另一方面,系统会分析测试生成的配置文件,进行容错性验证的同时,也会生成相应的 Jackson 网络排队模型,该模型是系统进行弹性资源调度的关键所在.

5.2 资源弹性调度模块

资源弹性调度模块由主控节点的调度器及子节点的控制器构成. 调度器会根据用户和控制器传来的调度请求进行全局调度分配,调度器的主要调度原则有:保障服务运行的优先级、将 CPU 密集型服务与 I/O 密集型服务混布、保障各服务公平的使用资源等规则.

由于容器与传统虚拟机不同,创建容器的时候可以不对其做资源限制,让其根据服务的运行需求自行申请或释放资源,这样可以最大化地利用宿主机的物理资源,但是采取这种方式会同时引入资源竞争的问题. 如当宿主机内多个容器的内存使用量同时突升时,可能会导致操作系统强制停止容器(out of memory kill);当某个容器的网络传输量很大时,可能会导致主控节点无法接收到子节点的心跳包,导致系统调度停滞等问题. 所以,需要在控制器中添加 1 个资源限制模块,保证系统进程的基本资源. 另外,当宿主机中各个服务的资源使用量都很高,而且总使用量已达到理论限制值时,是对服务进行迁移,还是对服务进行扩展?服务迁移会有一定的延迟及网络开销,服务扩展虽然开销较小,但是很难保障原有服务会持续保持高资源利用率的状态,所以,有可能在服务扩展后,又需要进行服务收缩,导致系统频繁进行调度. 为避免此情况,控制器会实时收集各个服务的负载、资源使用、性能等数据通过卡尔曼滤波对系统的输出响应时间进行实时预测,然后通过模糊逻辑理论进行分类与推理,最终得出服务是否需要进行服务迁移或者进行服务伸缩.

5.3 服务注册与发现模块

服务注册及发现模块采用分布式的一致性键值存储系统实现,内部采用层级树状存储结构,可以很高效地存储服务之间的关联关系. 服务注册及发现的过程可描述为,当子节点有新的服务生成时,控制器会根据该服务的配置信息,将服务的网络地址及端口号注册给主控节点的服务信息存储器,存储器会分析服务的配置信息对服务自动进行分类与组合.

6 实 验

本文所述方法已在基于 Docker 的容器虚拟化环境下进行了原型实现. 本节将通过实验与已有方法进行对比,验证系统模型在突发性负载场景下预测的准确性,以及在负载周期性变化场景下的有效性.

6.1 实验部署环境

实验选取 6 台相同配置的服务器,Intel Core i7 CPU 3.40 GHz,8 GB 内存以及千兆网卡,选取其中 1 台作为 Master 节点,4 台作为 Slave 节点,分别为 Slave1～Slave4,所有节点都安装有 CentOS7 的 minimal 系统以及 Docker 1.7. 另外 1 台用于客户端的负载发生器,装有 Windows 系统和 JMeter 压力测试程序.

实验用例选取 QCon San Franciso 2014 会议上的微服务架构最佳实践——Event Sourcing + CQRS 作为 Cloudsuite 中 CloudStone 的基础应用,该应用分为 4 个自包含的应用构件,每个应用构件都可以单独部署,服务之间相互独立.

下述实验中的误差是指真实值与估计值的偏差大于 5%的点.

6.2 实验设计

实验分为 4 个部分:

1) 通过 MATLAB 模拟系统运行情况,比较模糊自适应式卡尔曼滤波器对系统响应时间的预测值与模拟值的拟合情况,验证本文算法的准确性. 另一方面将调整前馈控制器中的模糊规则,并再次进行实验验证本文模糊规则的适用性.

2) 通过 JMeter 负载生成器对真实系统进行压力测试,并观察系统输出响应时间的变化曲线. 实验负载将模拟负载周期性变化、负载平稳、负载平稳上升、负载急剧上升、负载急剧下降等情况,然后记录应用响应时间,观察各应用的容器数量变化情况,从而验证系统实现的有效性.

3) 通过与已有工作中的资源供给算法进行对比,使用 MATLAB 模拟生成负载-使用率的数据集,作为系统输入,分别对增强学习、模糊控制、准入控制方式进行实验,观察各个算法对负载变化的响应情况,验证已有工作不适用于本文所述的问题场景.

4) 通过模拟负载随时间规律变化的情况,对比本文方法与已有工作在系统输入负载的预测准确率. 从而验证本文方法在负载周期性变化场景下的有效性.

6.3 模糊自适应卡尔曼滤波器的预测效果

本实验的目的是验证模糊自适应式卡尔曼滤波算法(fuzzy adaptive Kalman filtering, FAKF)对系统输出响应时间趋势的预测能力. 实验每隔 4 s 进行 1 次预测,共迭代进行实验 400 余次. 实验结果如图 6 所示,滤波器初始收敛时间一般在 10 个周期左右,即小于 40 s,当出现负载突变的情况,虽然滤波器需要 3～5 个周期,即 20 s 左右的时间才能收敛,但是滤波器对整体变化趋势的判断较好,可以为资源供给提供出有效的判断信息. 该实验证明了滤波算法可以在突发负载场景下进行自我调整,并得到有效预测.

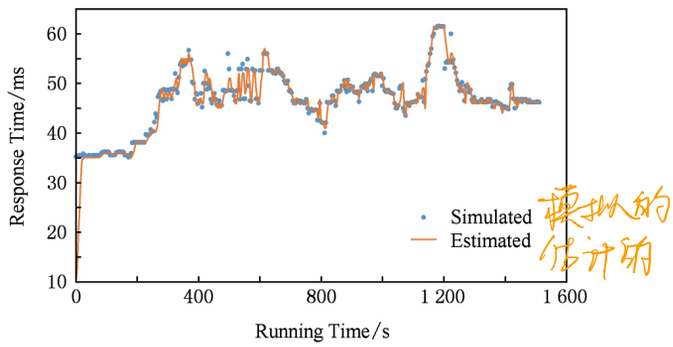

Fig. 6 Fuzzy adaptive Kalman filtering prediction experiment

图 6 模糊自适应卡尔曼滤波预测实验

系统初始化参数及模糊规则中的控制参数随机赋值,再次进行实验. 如图 7 所示,观察到系统方法预测结果误差率到达 30%左右,并且在 200～400 s 等多个时间段出现不收敛的情况. 说明初始参数及模糊规则中的控制参数对预测值有较大影响,应该参考本文 3.2.2 节所述方法进行赋值.

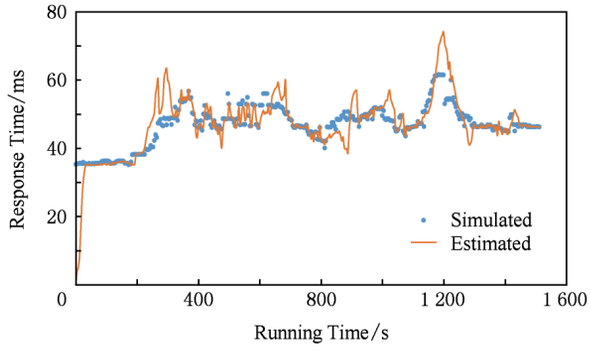

Fig. 7 Random control parameter experiment

图 7 随机控制参数实验

6.4 模型有效性及准确性分析实验

本实验的目的是验证系统模型在真实使用场景下,应对频繁变化的负载,系统的输出响应时间的平

稳性. 实验通过 JMeter 负载发生工具,模拟负载突发性变化场景. 系统负载变化如图 8 所示,系统输出响应时间如图 9 所示. 系统初始化时,将 2 个服务模块创建在 Slave1 上,其余 1 个服务创建在 Slave2 上,Slave3 和 Slave4 为空. 观察负载变化,其中 0～200 s 模拟 30 访问数/s 的稳定负载的情况,以充分保障滤波器预热收敛;200～300 s 模拟 30～60 访问数/s 的负载变化,观察到系统的输出响应时间也呈上升趋势,此时由于各个服务的资源使用情况尚未达到限制值,所以并没有发生服务扩展,但是由于 Slave1 上容器使用资源总量达到限制值,所以,将 Slave1 上的 2 个服务迁移到了 Slave3 和 Slave4 上,可以观测到迁移过程中系统输出响应时间存在小幅震荡;300～400 s 模拟 50 访问数/s 的平稳负载,可以观测到系统输出响应时间也处于平稳状态;600～700 s 模拟 50～100 访问数/s 的负载突增,可以观测到系统在 650 s 左右时,对 Slave1 和 Slave2 的某些服务进行了扩展,导致系统输出响应时间平缓下降;700～800 s 模拟 100～30 访问数/s 的负载突降情况,可以观测到系统在 750 s 左右,对服务进行了收缩. 800～1 600 s 又再次模拟了这个负载变化过程,系统输出响应时间基本保持前 800 s 的变化规律. 但是前 800s的负载突增是缓慢的增加趋势,然后突增至

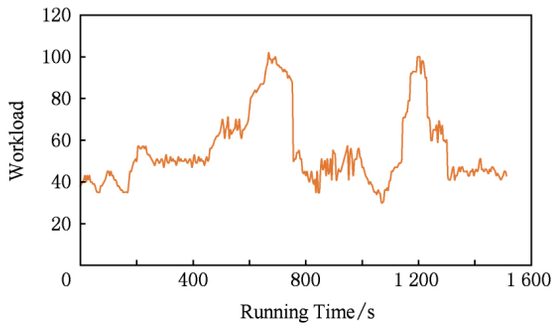

Fig. 8 Workload curve

图 8 系统负载曲线

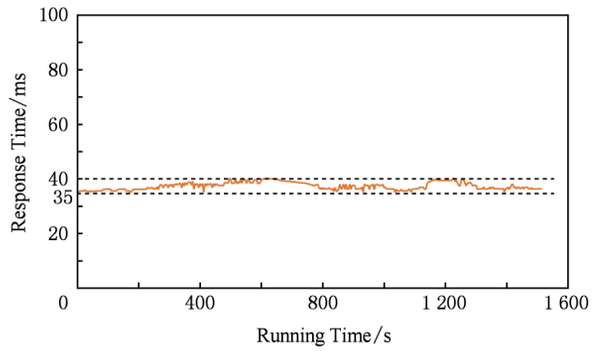

Fig. 9 Output responses time curve

图 9 系统输出响应时间曲线

“峰值”,而后 800 s 的负载突增是模拟的突降至“谷底”后又突增至“峰值”,以此证明系统可以应对实时性突发负载. 此实验中虽然系统输出响应时间随负载变化有变化趋势,但是整体保持在 30～40 ms 的平稳曲线上,以此证明了本模型的有效性及可用性.

6.5 与已有模型的对比实验

为说明已有方法在本文所述场景中的不适用性及验证本文方法的有效性,本节将与 3 种比较经典的资源供给算法进行对比. 本节实验都 MATLAB 模拟输入负载作为相同的基准测试集数据,首先收集 CloudStone 中性能数据与输入负载等数据形成训练集,然后依赖该训练集数据,形成基准测试集数据. 由于已有方法多为对负载进行预测,因此,通过本文方法预测得到响应时间后,根据式(1)(2)转化为相应的负载,实验结果如图 10 所示,预测误差率小于 5%,负载变化趋势预测准确.

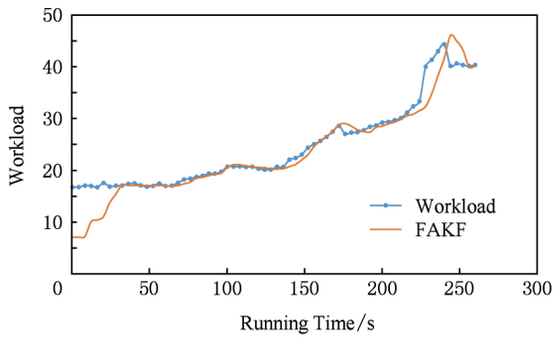

Fig. 10 FAKF algorithm fitting curve

图 10 模糊自适应卡尔曼滤波算法负载拟合曲线

6.5.1 基于增强学习的资源供给方法

由于增强学习(reinforcement learning, RL)是一种离线测试后在线调整的算法,本实验首先收集 CloudStone 的负载与系统各个资源使用率的数据,形成负载-使用率数据对,作为训练集. 然后,根据 Martinez 等人[11]的 RL 算法训练出资源供给规则,带入基准测试集中的资源使用率数据,并观察输出负载与真实负载的拟合情况.

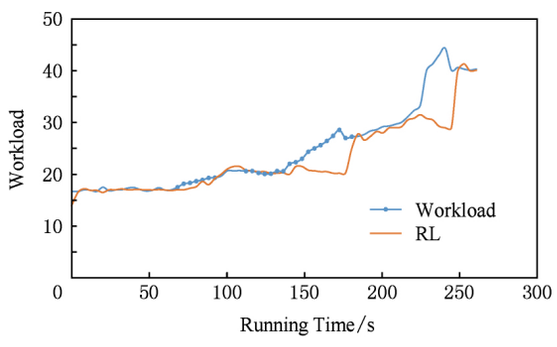

Fig. 11 RL algorithm load fitting curve

图 11 RL 算法负载拟合曲线

如图 11 所示,RL 算法误差率约为 25%左右,通过实验观察发现 RL 算法在负载突发性变化时收敛较慢,需要 30～50 个周期的时间才能完全收敛,如 150～200 s 时间段,甚至有可能出现不收敛的情况,如 230～270 s 时间段.微服务变更频繁是导致以历史数据为基准的 RL 算法准确度较低的一个主要原因.

6.5.2 基于模糊控制的资源供给方法

本实验以 Lama 等人[19]提出的适用于 3 层 Web 架构的模糊逻辑资源供给算法作为验证.将基准测试集中的资源使用率数据带入模糊规则,并观察输出负载与真实负载的对比情况.

如图 12 所示,模糊逻辑算法误差率约为 75%左右,通过实验观察发现基于模糊逻辑的资源供给算法对模糊函数的准确度要求较高,但是模糊函数的设计上缺乏理论指导,一般都是人为经验给出的,所以并不适用于结构复杂的微服务架构.另一方面,模糊函数对负载变化规律要求较高,当出现负载变化不规律,如 50～100 s 和 150～300 s 时间段这种负载突发性变化时,基于模糊规则的资源供给方式准确率较低,无法保障应用服务质量.

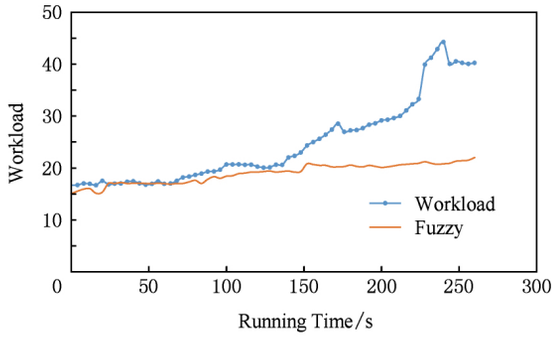

Fig. 12 Fuzzy logic algorithm load fitting curve

图 12 模糊逻辑算法负载拟合曲线

6.5.3 基于反馈控制的资源供给方法

反馈控制方法对比 RL 算法和模糊控制算法,不仅减少了训练规则的复杂度,而且提高了系统运行的稳定程度,但是控制参数的设计却需要大量领域经验,所以一般适用于负载随时间规律变化的场景.本实验以 Lu 等人[4]的反馈控制算法作为验证,将基准测试集带入反馈控制算法,观察预测负载与真实负载的对比情况.

如图 13 所示,反馈控制算法误差率约为 20%左右,通过实验观察发现误差点多集中在初始化阶段及负载突发性变化阶段,如 70～100 s 时间段和 200～260 s 时间段,反馈控制算法的收敛时间较长.其原因在于控制参数是根据领域经验给出的固定的值,不能自适应调整,无法应对负载无规律变化的场景.

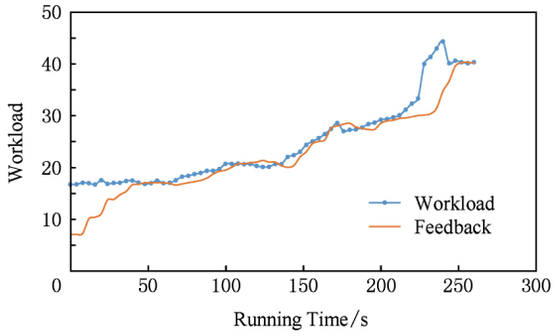

Fig. 13 Feedback control algorithm load

图 13 反馈控制算法负载拟合曲线

6.5.4 总 结

通过对比实验发现,已有工作在基于微服务且负载突发性变化的场景下,对负载的拟合度相对较低,甚至出现发散的情况,而本文方法误差率较低,且误差点多集中在初始化阶段,可准确预测负载变化趋势.

6.6 周期性负载变化场景下模型有效性验证

本节实验对比本文方法与 Lama 等人[28]提出的基于 RAMA 反馈控制方法对负载情况的预测.通过 MATLAB 模拟负载随时间规律性变化的应用场景,将负载使用率参数对作为测试集,根据 RAMA 算法得到对负载的预测数据集.如图 14 所示,通过实验观察发现 RAMA 反馈控制算法在负载随时间规律性变化的场景下拟合度较高,误差率小于 5%.

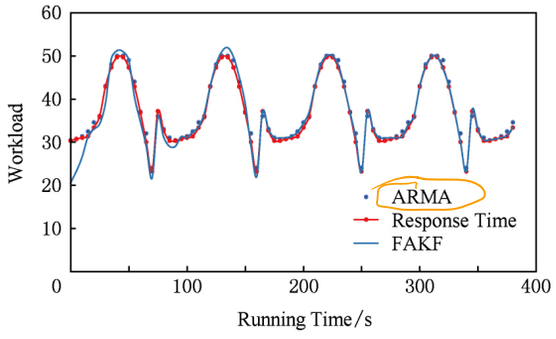

Fig. 14 Workload comparison with RAMA algorithm

图 14 本文方法与 RAMA 算法的负载预测对比实验

本文方法误差率为 5%左右,略高于 RAMA 算法,但是,由于采用自适应调整机制,虽然前 2 个周期内预测准确率要小于 RAMA 算法,但是本文方法收敛快速,在经过 2～3 个周期调整后,参数趋于

稳定,预测准确率逐步提升,在第 4 个周期时,准确率要高于 RAMA 算法.

7 结束语

本文提出了一种基于模糊自适应式卡尔曼滤波算法的弹性资源供给模型,通过实时预测系统的输出响应时间,对系统中各个服务进行迁移或者伸缩,保证了系统的可靠运行.虽然本文在微服务场景下对该模型进行了验证,但是本文的预测算法同样适合于所有满足 Jackson 网络排队模型的传统架构的应用.

参 考 文 献

[1] Savchenko D I, Radchenko G I, Taipale O. Microservices validation: Mjolnirr platform case study [C] //Proc of the 38th Int Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Piscataway, NJ: IEEE, 2015: 235-240

[2] Ferguson P, Huston G. Quality of service: Delivering QoS on the Internet and in corporate networks [J]. Computer Communications, 1999, 22(10): 980-981

[3] Soltesz S, Pötzl H, Fiuczynski M E, et al. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors [J]. ACM SIGOPS Operating Systems Review, 2007, 41(3): 275-287

[4] Lu Chenyang, Abdelzaher T F, Stankovic J A, et al. Feedback control architecture and design methodology for service delay guarantees in Web servers [J]. IEEE Trans on Parallel and Distributed Systems, 2006, 17(9): 1014-1027

[5] Kundu S, Rangaswami R, Dutta K, et al. Application performance modeling in a virtualized environment [C] // Proc of the 16th Int Symp on High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2010: 307-318

[6] Lama P, Zhou Xiaobo. Efficient server provisioning with control for end-to-end response time guarantee on multitier clusters [J]. IEEE Trans on Parallel and Distributed Systems, 2012, 23(1): 78-86

[7] Cao Junwei, Zhang Wen, Tan Wei. Dynamic control of data streaming and processing in a virtualized environment [J]. IEEE Trans on Automation Science and Engineering, 2012, 9(2): 365-376

[8] Lu Ying, Abdelzaher T F, Lu Chenyang, et al. Feedback control with queueing-theoretic prediction for relative delay guarantees in Web servers [C] //Proc of the 9th Real-Time and Embedded Technology and Applications Symp. Piscataway, NJ: IEEE, 2003: 208-217

[9] Cherkasova L, Phaal P. Session-based admission control: A mechanism for peak load management of commercial Web sites [J]. IEEE Trans on Computers, 2002, 51(6): 669-685

[10] Robertsson A, Wittenmark B, Kihl M, et al. Design and evaluation of load control in Web server systems [C] //Proc of the 8th American Control Conf. Piscataway, NJ: IEEE, 2004: 1980-1985

[11] Martinez J F, Ipek E. Dynamic multicore resource management: A machine learning approach [J]. Micro, 2009, 29(5): 8-17

[12] Tesauro G, Jong N K, Das R, et al. A hybrid reinforcement learning approach to autonomic resource allocation [C] // Proc of the 3rd Int Conf on Autonomic Computing (ICAC). Piscataway, NJ: IEEE, 2006: 65-73

[13] Bodık P, Griffith R, Sutton C, et al. Statistical machine learning makes automatic control practical for internet datacenters [C] //Proc of the 1st Conf on Hot Topics in Cloud Computing. Berkeley, CA: USENIX Association, 2009: 12-17

[14] Xu Chengzhong, Rao Jia, Bu Xiangping. URL: A unified reinforcement learning approach for autonomic cloud management [J]. Journal of Parallel and Distributed Computing, 2012, 72(2): 95-105

[15] Karlsson M, Karamanolis C, Zhu Xiaoyun. Triage: Performance isolation and differentiation for storage systems [C] //Proc of the 12th Int Workshop on Quality of Service (IWQOS). Piscataway, NJ: IEEE, 2004: 67-74

[16] Wu Heng, Zhang Wenbo, Zhang Jianhua, et al. Benefit-Aware on-demand provisioning approach for virtual resources [J]. Journal of Software, 2013, 24(8): 1963-1980 (in Chinese)
(吴恒,张文博,张建华,等.一种收益敏感的虚拟资源按需提供方法[J].软件学报,2013,24(8):1963-1980)

[17] Wang Kai, Hou Zifeng. An adaptive scheduling method of weight parameter adjustment on virtual machines [J]. Journal of Computer Research and Development, 2011, 48(11): 2094-2102 (in Chinese)
(王凯,侯紫峰.自适应调整虚拟机权重参数的调度方法[J].计算机研究与发展,2011,48(11):2094-2102)

[18] Diao Y, Hellerstein J L, Parekh S. Optimizing quality of service using fuzzy control [G] //Management Technologies for E-Commerce and E-Business Applications. Berlin: Springer, 2002: 42-53

[19] Lama P, Zhou Xiaobo. Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee [C] //Proc of the 18th Int Symp on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS). Piscataway, NJ: IEEE, 2010: 151-160

[20] Lama P, Zhou Xiaobo. Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee [J]. ACM Trans on Autonomous and Adaptive Systems, 2013, 8(2): 9-40

[21] Wang Sa, Zhang Wenbo, Wu Heng, et al. Approach of quantifying virtual machine performance interference based on hardware performance counter [J]. Journal of Software, 2015, 26(8): 2074-2090 (in Chinese)
(王卅, 张文博, 吴恒, 等. 一种基于硬件计数器的虚拟机性能干扰估算方法[J]. 软件学报, 2015, 26(8):2074-2090)

[22] Shanthikumar J G, Buzacott J A. Open queueing network models of dynamic job shops [J]. International Journal of Production Research, 1981, 19(3): 255-266

[23] Gandhi A, Dube P, Karve A, et al. Adaptive, model-driven autoscaling for cloud applications [C] //Proc of the 11th Int Conf on Autonomic Computing (ICAC'14). Berkeley, CA: USENIX Association, 2014: 57-64

[24] Kalman R E. A new approach to linear filtering and prediction problems [J]. Journal of Basic Engineering, 1960, 82(1): 35-45

[25] Brown R G, Hwang P Y C. Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions [M]. New York: Wiley Heyden Ltd, 1997

[26] Sinopoli B, Schenato L, Franceschetti M, et al. Kalman filtering with intermittent observations [J]. IEEE Trans on Automatic Control, 2004, 49(9): 1453-1464

[27] Frühwirth R. Application of Kalman filtering to track and vertex fitting [J]. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 1987, 262(2): 444-450

[28] Lama P, Guo Yanfei, Zhou Xiaobo. Autonomic performance and power control for co-located Web applications on virtualized servers [C] //Proc of the 21st Int Symp on Quality of Service (IWQoS). Piscataway, NJ: IEEE, 2013: 1-10

Hao Tingyi, born in 1989. Master. His main research interests include network distributed computing and software engineering.

Wu Heng, born in 1983. Assistant researcher. His main research interests include network distributed computing and software engineering, etc.

Wu Guoquan, born in 1979. PhD. Associate professor of the Institute of Software, Chinese Academy of Sciences. Member of CCF. His main research interests include network distributed computing.

Zhang Wenbo, born in 1976. PhD. Professor and PhD supervisor. His main research interests include network distributed computing and software engineering, etc.