文章编号：2096-1618(2019)03-0223-05

基于 Kubernetes 的微服务业务拓扑发现及业务评价

高 宇， 陶宏才
(西南交通大学信息科学与技术学院，四川 成都 611756)

摘要：随着 Kubernetes 在各个云服务平台的广泛使用，在云上获取业务逻辑流向和可视化业务关系，对云上业务进行评估，有利于业务管理人员及时发现并解决问题。在以 Weave Scope conntrack 方式获取业务拓扑的基础上，使用 TCP 流量抓包方式获取业务拓扑，弥补了 conntrack 未记录某些转发信息的情形。将权重计算方法应用到 Kubernetes 中的微服务业务，结合业务中的资源并利用业务的繁忙度来评估业务。使用变权法对熵权法得到的常权重进行修正，以使业务繁忙度的计算更加准确。

关 键 词：Kubernetes；微服务；业务拓扑；业务评价；繁忙度
中图分类号：TP393.022 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2019.03.002

0 引言

随着云计算、虚拟化等技术的出现，单体架构已不能满足云端应用的部署，云端应用需要使用微服务架构来实现服务之间的松耦合。微服务架构中服务的细分演化及云端业务的丰富，使微服务业务系统变得愈发复杂[1]，同时也增加了运维的困难性。为有效管理微服务业务，Kubernetes(K8S)提供了一种将本来要捆绑在一起的服务进行拆分的机制，帮助微服务架构实现了落地。并且，Kubernetes 的自动化部署、自动扩缩容和负载均衡等优点[2]，使 Kubernetes 深受各云服务平台的青睐，云服务平台纷纷以 Kubernetes 作为构建容器化服务的平台方案。为了降低运维的困难性，则需要监控云上业务，使用可视化的方式反映云上业务关系，以帮助业务管理人员在复杂的系统中发现业务问题。

当前，能够获取云上业务状态的方式主要是应用性能管理系统(application performance management，APM)，国内外的 APM 主要有 Computerware、New Relic、听云和 OneAPM 等[3]。APM 以云平台应用为核心，主要是进行应用性能的瓶颈分析，但不对业务进行评估。其主要工作方式有探针和日志。探针方式主要针对 Java 应用，而目前云上业务中 Go 应用的占比较大，所以探针方式不能较全面地覆盖云上应用[4]。至于日志方式，因运维人员需要通过额外代码开发量增加各种各样的日志格式，而满足日志规范的日志才能为运维人员解析，故日志方式的实用性较差。

相比于 APM，文中注重站在用户的角度，获取云平台中的微服务业务逻辑流向并评估业务性能，主要方式是以现有插件 Weave Scope[5] conntrack 的方式，结合 TCP 流量走向，进行 Kubernetes 云上业务拓扑的发现。此外，对微服务业务评估的方法进行研究，然后结合业务中的资源对业务进行综合评估，以便业务管理人员及时采取主动措施，驱动业务改进。

收稿日期：2019-02-05
基金项目：国家自然科学基金资助项目(61505168)

1 相关技术方法

1.1 业务拓扑发现方法

Weave Scope 是 Kubernetes 的可视化监控工具，由两个组件构成，分别是 agent 和 APP。agent 组件是收集信息的探针，负责收集集群中各节点的拓扑信息和运行信息。APP 组件负责将 agent 组件中收集到的信息总结生成报告，提供给用户。agent 收集拓扑信息是基于 Linux 内核的连接跟踪模块 conntrack，该模块会记录防火墙所转发的连接信息，包括源 IP 地址、目的 IP 地址、源端口和目的端口等[6]。Agent 收集信息还会采取 eBPF 的方式。BPF(berkeley packet filter)是一种用于过滤网络报文的架构，eBPF(extended berkeley packet filter)的设计源于 BPF，它在内核追踪、应用性能监控和流量控制等方面提供了可靠的方式[7]。但是，eBPF 方式要求 Linux 内核版本大于4.4，而文中所用的 Linux 内核版本为3.10，所以不考虑 eBPF 方式。在实际操作过程中，发现 conntrack 模块存在没有记录某些实际转发信息的情况，使 agent 无法获取全部连接信息。为了弥补这一缺点，采用在 Kubernetes 环境上抓包的方式来获取流量转发路径，进而获取连接调用关系，生成完整的微服务拓扑信息。Tcpdump 是 Linux 中强大的网络数据采集工具，可以将网络上的数

据包截取,供用户分析使用。在 Kubernetes 上,kube-proxy 组件可实现流量从 Pod 到 Service、Service 到 Pod 的转发,负责 TCP 和 UDP 数据包的路由转发。因此,使用 Tcpdump 以获取 Kubernetes 中流量的转发路径。

1.2 业务评价方法

评价业务性能是多方面的,如果某个业务的繁忙程度一直居高不下,则说明系统内部需要进行优化,以更好地支持系统工作。所以,使用业务繁忙度来评估业务,能够快速地发现业务问题。云计算平台中充斥着各种各样的资源,而这些资源支持着各个微服务业务提供的服务[8]。因此,评价业务繁忙度,需要结合业务中多方面的资源进行综合评估。

取自云计算平台中常用的资源,包括应用类资源(如 Tomcat 和 MySQL)和计算类资源(如 CVK 主机、VirtualMachine 虚拟机、K8S 节点和 K8S 容器)。为了评估上述资源,必须建立评估准则。根据微服务业务中各资源因素的影响,结合相关资料与实际指标测试,归纳筛选出具有代表性,完备性的资源繁忙度指标,指标集合见表 1。

表 1 微服务业务繁忙度评价指标体系

类别	资源	指标
中间件	Tomcat	Tomcat 每秒接收到的请求数
		当前繁忙线程占最大线程的比例
数据库	MySQL	MySQL 每秒执行的语句数
H3C CAS 云计算管理平台	CVK 主机	CVK 主机 CPU 利用率
		CVK 主机内存利用率
	VirtualMachine 虚拟机	虚拟机 CPU 利用率
		虚拟机内存利用率
Kubernetes	K8S 节点	节点 CPU 利用率
		节点内存利用率
	K8S 容器(Docker 容器)	容器对节点 CPU 的总利用率
		容器对节点内存总利用率

2 微服务业务拓扑发现

2.1 Kubernetes 中的微服务

Kubernetes 是自动化容器操作的平台,其中容器技术以 Docker 技术为代表,容器技术将容器变为资源分配和利用的最小单元,具有强大的可移植性和跨平台[9]。其中,Docker 镜像是生成 Docker 容器的基础,微服务会打包成 Docker 镜像。

Kubernetes 中各个微服务需要相互调用来支持整体应用,在基于 TCP 流量进行拓扑发现时,微服务之间的关系需要根据 Kubernetes 中的操作对象 Service 和 Pod 关系进行转化。由于 Kubernetes 中 Pod 的运行状态可以动态变化,Kubernetes 会重新为其分配 IP,Service 提供了一种直接访问 Service,通过 Service 将请求转发到 Pod,不需要关心 Pod 状态的机制[10]。Pod 是 Kubernetes 的最小管理单元,Pod 中可以包含一个或多个容器,容器共享 Pod 的 IP 地址和端口范围。

2.2 Weave Scope

由于在 Kubernetes 环境中提供微服务业务的 Pod 可能有多个副本,因此如果按照 Pod 粒度或容器粒度来进行业务拓扑发现,就会出现多条调用关系,而实际微服务之间只存在一条调用关系。此外,因为 Kubernetes 中的微服务会打包成镜像,所以根据微服务镜像来进行拓扑发现。

使用 Weave Scope 插件进行拓扑发现,先在需要捕获关系的 K8S 环境上部署 Weave Scope。部署成功后,即可使用其 API 获取相应的信息。这里需要使用两个 API,一个是/api,可获取该 K8S 环境的唯一标识,并将获取到的唯一标识存储到 Etcd 中,供 TCP 流量抓包方式获取相关业务拓扑保存链路数据使用。另一个是/api/topology/containers,可获取容器及其连接链路的信息。获取信息之后进行解析,解析为容器镜像之间的关系,持久化到数据库。在实际的 conntrack 记录中,某些转发信息可能没有在记录表中,所以使用 TCP 抓包方式作为辅助手段,补充链路发现。

2.3 基于 TCP 流量的拓扑发现

在 Kubernetes 环境中,对象的增减(称缩放)以 Pod 为单位。于是,所有在 Pod 中的容器,无论是否需要,也都会随之被缩放,但这样会造成资源的浪费和成本的增加[11]。所以,Tcpdump 抓包时,采用 Pod 单容器抓包方式。

```
docker tun-i-name tcpdump $1-net
container:$1 tcpdump:test
tcpdump "(tcp[tcpflags]&(tcp-syn)! =0)
and(tcp[tcpflags]&(tcp-ack)= =0)"-c
$acqCount -q -nn >>cap/$1.cap
2>/dev/null
```

容器抓包的过程是:先启动抓包容器 tcpdump $1,其与被抓包容器 container:$1 处于同一网络;然后,抓取指定数量$ acqCount 的 TCP 3 次握手 SYN 请求包。不过,如果抓取全部的数据包(如 ACK、SYN、FIN、PUSH 等),则无法确定数据包方向。因此只抓取 SYN 请求包。

利用抓包方法生成业务拓扑的步骤如下:

Step1:根据配置文件获取 Pod IP 和 Service IP 前缀,并从 Etcd 中获取该 Kubernetes 环境标识。Step2:根据该 K8S 环境标识,从远程数据库(使用 MySQL 数据库存储数据)获取该环境的业务信息,保存到文件。Step3:查询该环境中运行的容器 id,并过滤基础容器等。Step4:遍历容器,将该容器的 Pod IP 和 Service IP 转化为镜像 id 的映射。Step5:遍历容器,如果该容器可以抓包,则启动抓包容器,抓取与该容器在同一个网络下指定数量的 TCP 请求包,保存抓包数据到 cap 文件,等待抓包进程结束。若超时,则强制结束抓包进程。若未超时,且抓包进程已经抓到指定数据包,则抓包进程自动结束。抓包结束后,删除该抓包容器。Step6: 遍历容器,根据 2.1 中提到的 Pod 和 Service 关系,对抓包文件进行解析得出源业务 id 和目的业务 id,判断数据库中是否存在以该源业务和目的业务为起始端点和目的端点的链路,不存在则插入该链路到数据库。

2.4　业务拓扑发现实际应用

使用 Weave Scope 可获取微服务业务拓扑,如图 1 所示。不过,在 K8S 环境下,实际存在微服务 alert-manager 到微服务 alert-collector 的调用关系,而 Weave Scope 不能获取到该关系。经笔者实际调试,发现 conntrack 记录表中一直不存在这条记录。为此,使用 TCP 流量发现的方法来获取 alertmanager 到 alert-collector 的调用关系。于是,可获得实际业务完整的拓扑图,如图 2 所示。

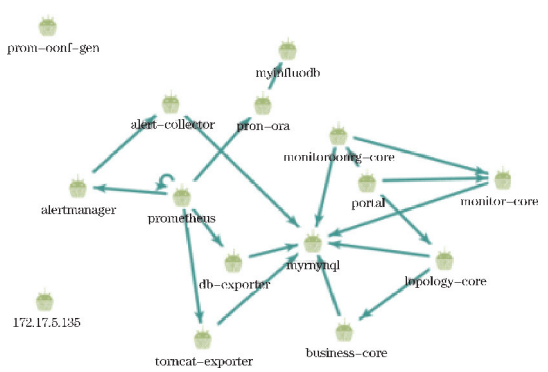

图 1　Weave Scope 获取的微服务业务拓扑

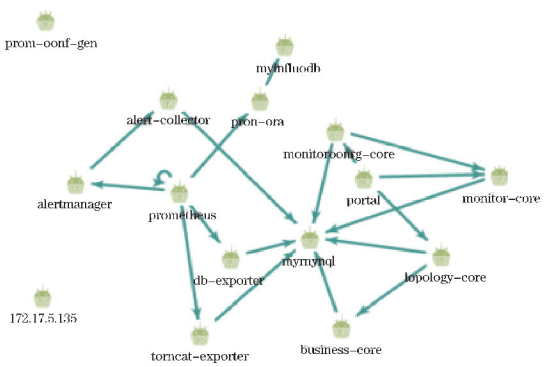

图 2　微服务业务拓扑

3　微服务业务评价

3.1　业务评价等级

业务的繁忙度得分,基于业务内各个资源的基础监控指标数据计算得出,表示业务运转的繁忙程度,此得分越高业务越繁忙,业务繁忙等级如表 2 所示。

表 2　业务繁忙度分值与等级关系表

业务繁忙度分值	业务繁忙等级
0 ~ 25	较空闲
26 ~ 74	较繁忙
75 ~ 100	很繁忙

3.2　资源繁忙度的计算

3.2.1　指标量纲统一化

对选取指标进行权重计算和繁忙度计算之前,需要先对指标量纲进行统一化处理,选取 Min-max 归一化[12]方法。这种归一化比较适合处理数值比较集中的情况,实际使用中可以用经验常量来代替 max 和 min 值。计算繁忙度时,选取的都是对繁忙度有贡献的指标(即正向指标),指标值越大说明对繁忙度的贡献越大。暂未考虑负向指标。正向指标归一化方式为

$$y_i = \frac{x_i - \min x_i}{\max x_i - \min x_i} \tag{1}$$

其中 x_i 为实际指标值,y_i 为归一化后的指标值。

将表 1 中 Tomcat 资源每秒接收到的请求数指标量纲统一化为百分制。并且,遵照通行做法,将指标 Tomcat 每秒接收到的请求数的最小值定为 0,最大值定为 3000。

3.2.2　熵权法与变权法结合生成权重

确定资源各指标常权重的方法为客观赋权法,即熵权法。熵权法是一种根据各项指标观测值所提供的信息量的大小来确定指标权重的方法。在信息系统中,如某属性信息熵越大,则信息无序度越高;而信息量越小,则其在评价中的权重越小。因此,信息熵可用来确定评估系统中属性权重的大小[13]。

常权反映了在理想状态下,各评价指标之间的相对重要程度,因其具有一定的合理性而被广泛使用。变权法则是在常权的基础上,对常权向量进行相应的调整,从而满足对加权综合结果的不同偏好设计要求。当某个指标较为严重时,若直接使用常权进行评估,则该指标的严重程度可能被其他指标中和,导致评估出来的繁忙度结果失真。而运用变权法则可以根据各指标的严重程度,对各指标的常权进行调整,从而使结果更加符合实际情况。故采取常权结合变权的方式,建

立信息熵—变权模型,确定各指标的权重。

在变权理论中,状态变权向量可分为惩罚型、激励型和混合型。因为使用正向指标对繁忙度进行评估,所以采用激励型状态变权向量。方法如下:

设 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 为因素状态向量,$\boldsymbol{W}=(w_1,w_2,\cdots,w_n)$ 为因素常权向量,$\boldsymbol{S}(\boldsymbol{X})=(S_1(X),S_2(X),\cdots,S_n(X))$ 为状态变权向量,则变权向量 $\boldsymbol{W}(\boldsymbol{X})=(w_1(X),w_2(X),\cdots,w_n(X))$ 计算公式为[14]

$$w_j(X)=w_jS_j(X)/\sum_{k=1}^{n}w_kS_k(X) \tag{2}$$

借助指数型状态变权向量构造方法,状态变权向量 $\boldsymbol{S}(\boldsymbol{X})=(S_1(X),S_2(X),\cdots,S_n(x))$ 按公式(3)[14]构造:

$$S_i(X)=\begin{cases}1 & ,0\leqslant X_i\leqslant b\\ e^{a(X_i-b)} & ,b<X_i\leqslant 1\end{cases},i=1,2,\cdots,n \tag{3}$$

其中,$\boldsymbol{S}(\boldsymbol{X})$ 为激励型状态变权向量,a 为奖励水平。a 越大,则激励效果越明显。b 为指标的阈值,一般取经验值0.85。即指标值超过85% 时,须修正相应的繁忙度权重,对权重进行增强。

3.2.3 指标权重计算

根据表 1 中的资源指标,以 K8S 容器为例,选取多个容器对象,实际采集其指标数据,构造出的样本矩阵如下。其中,行表示容器对象,列表示指标。

$$\begin{bmatrix}0.12469 & 12.1889\\ 0.0346 & 0.496776\\ 0.11597 & 2.3475\\ 2.8578 & 2.433657\\ 0.015465 & 0.1238726\\ 0.09906 & 0.5265\\ 0.434792 & 2.114435\\ 0.0822648 & 1.1365\end{bmatrix}$$

根据熵权法[13]计算出 K8S 容器的常权为[0.631 0.369]。同理,构造其他资源的样本矩阵,计算这些资源的繁忙度指标权重,分别为,K8S 节点:[0.503 0.497],CVK 主机:[0.703 0.297],虚拟机:[0.705 0.295],Tomcat:[0.350 0.650]。

在对常权修正时,选取不同的指标数据对不同的奖励水平进行测试。测试结果发现:当指标值相差较大时,奖励水平 a 取 4,5,8,10,指标权重转移就超过 0.1,说明指标权重转移过大,就会出现权重失真的情况。而当 a 取 1 时,调权程度不明显。在 a 取 2,3 时,两者权重转移差异不明显,为使调权程度适中,取其中间值,即奖励水平 a 的值取2.5。

综上,为使随着指标值增大时,繁忙度增长速度加快,奖励水平 a 取2.5。即当指标值超过85% 时,需要对常权重进行修正,修正后的权重由 $\boldsymbol{S}(\boldsymbol{X})$ 确定。综合权重得到后,就可以计算出各资源的繁忙度,结合资料综合得出资源繁忙度为

$$C=\sum_{j=1}^{n}v_j(X)x_j \tag{4}$$

其中,$v(X)$ 为使用上述方法对常权修正后的各指标权重。

3.3 业务繁忙度的计算

3.3.1 专家咨询法

专家咨询法是专家根据自己的经验和理解,对待评价的指标进行初步打分。然后,研究多位专家打分的偏移程度,进行综合,确定出指标权重的一种方法[15]。采用专家咨询法确定微服务业务中各资源权重,进而根据资源权重,进行业务繁忙度的计算。根据实际业务需求,将 Kubernetes 中权重的计算划分为 4 种情形。其中:情形 1 包括 K8S 节点、K8S 容器、主机和虚拟机;情形 2 包括 K8S 节点、K8S 容器、主机、虚拟机和 MySQL;情形 3 包括 K8S 节点、K8S 容器、主机、虚拟机和 Tomcat;情形 4 包括 K8S 节点、K8S 容器、主机、虚拟机、MySQL 和 Tomcat。

选取专家(系统设计者)和系统可能的使用者按照上述中的各种情况进行打分,以情形 1 为例,得到如下打分矩阵。其中,行表示专家,列表示资源。

$$\begin{bmatrix}0.3 & 0.4 & 0.1 & 0.2\\ 0.3 & 0.3 & 0.2 & 0.2\\ 0.4 & 0.1 & 0.3 & 0.2\\ 0.2 & 0.1 & 0.4 & 0.3\\ 0.3 & 0.3 & 0.2 & 0.2\end{bmatrix}$$

根据专家咨询法[15]计算出情形 1 的权重为[0.303 0.270 0.225 0.202],同理计算出其他情形下各资源的权重分别为:

情形 2 [0.303 0.236 0.195 0.158 0.108]

情形 3 [0.307 0.226 0.198 0.160 0.109]

情形 4 [0.229 0.229 0.197 0.115 0.115 0.115]

3.3.2 计算业务繁忙度

计算业务繁忙度,首先需要确定业务中的资源,结合 3.2 中确定出来的资源繁忙度和 3.3.1 中使用专家咨询法确定的资源繁忙度权重,即可得到业务的繁忙度,通过参考相关文献资料综合得出业务繁忙度公式为

$$F=\frac{\sum_{j=1}^{m}u_j(X)C_j}{\sum_{j=1}^{m}u_j(X)} \tag{5}$$

其中 $u(X)$ 为专家咨询法确定的资源权重。

例如,针对某 Kubernetes 环境中的某业务使用上述方法计算业务繁忙度,首先对该业务添加如情形 1 类型

的资源,获取各资源如表 1 的繁忙度指标值,其中 K8S 节点的指标值为:[23　32],K8S 容器的指标值为:[3.45　11.34],CVK 主机的指标值:[56.66　48.63],虚拟机的指标值:[30.84　100]。按上述计算方法,得出业务繁忙度值为33.77。对照表 2,即可知道该业务处于较繁忙程度。

4　结束语

针对云服务平台中 Kubernetes 环境,使用插件 Weave Scope 进行微服务业务拓扑发现,并使用 TCP 流量抓包解析的方法作为辅助手段,弥补了 Weave Scope conntrack 未记录某些转发信息的缺点。同时,分析能够反映各资源繁忙度的指标体系,研究权重计算方法如何与业务繁忙度相结合。使用熵权法对资源各指标的权重进行确定,为了更加准确地计算出业务的繁忙度,使用变权法对指标过于严重时的常权进行调整。结合专家咨询法,对资源在不同情况下进行打分,得到各资源在业务中的权重,进而通过计算业务繁忙度,来评估出微服务业务的繁忙程度。

参考文献:

[1]　田兵,王玮,苏琦. 基于微服务架构的应用性能监控平台研究[J]. 信息技术与信息化. 2018,(1):125-128.

[2]　Kubernetes 是什么[EB/OL]. http://docs. kubernetes. org. cn/227. html,2018-6-12.

[3]　潘昆豪. 应用性能管理中应用性能分类评估系统的研究与实现[D]. 北京:北京邮电大学,2014.

[4]　梁伟,杨明川,冯明. 应用性能管理技术的研发与应用[J]. 电信技术,2017(6):42-45.

[5]　Introducing Weave Scope[EB/OL]. https://www.weave. works/docs/scope/latest/introducing/, 2018-12-10.

[6]　严宏,孟晓东. 状态检测防火墙中流量采集的研究与实现[J]. 福建电脑,2008(9):167-168.

[7]　eBPF简史[EB/OL]. https://www. ibm. com/developerworks/cn/linux/l-lo-eBPF-history/, 2018-12-15.

[8]　肖扬,于艳华. 基于 Iaas 云平台的应用性能管理研究与应用[J]. 软件,2013(12):241-245.

[9]　Merkel D. Docker:lightweight Linux containers for consistent development and deployment[J]. 2014(239).

[10]　于泽萍. 面向微服务架构的容器云平台设计与实现[D]. 哈尔滨:哈尔滨工业大学. 2018.

[11]　龚正. Kubernetes 权威指南[M]. 北京:电子工业出版社,2017.

[12]　艾鹏. 网络攻击效果评估技术研究与实现[D]. 长沙:国防科学技术大学,2015.

[13]　朱怡安,周延年,夏平. 基于熵权的嵌入式计算机性能灰估计[J]. 西北工业大学学报,2012,30(5):647-651.

[14]　杨雪莹. 基于改进层次分析法的配电网规划综合评价方法[D]. 武汉:华中科技大学,2015.

[15]　张恺伦. 电力系统中典型扰动性负荷的电能质量影响研究[D]. 杭州:浙江大学,2013.

Topology Discovery and Service Evaluation of Microservice Service based on Kubernetes

GAO Yu,　TAO Hongcai

(School of Information Science & Technology, Southwest Jiaotong University, Chengdu 611756, China)

Abstract: With the wide use of Kubernetes in various cloud service platforms, it is beneficial for business managers to find and solve problems in time by acquiring business logic flow, visualizing business relationships and evaluating cloud services. Based on the Weave Scope conntrack method to obtain the service topology, this paper uses the TCP traffic capture method to obtain the service topology, which makes up for the fact that conntrack does not record some forwarding information. And the weight calculation method is applied to the micro-service in Kubernetes, combining the resources in the service, using the busy degree of the service to evaluate the service. The constant weight obtained by entropy weight method is modified by variable weight method to make the calculation of busy degree more accurate.

Keywords: Kubernetes; micro-services; business topology; business evaluation; busy degree