



基于 Kubernetes 的容器化数据库及其集群方案

严丽云¹, 何震苇¹, 杨新章¹, 张凌¹, 侯韶新²

(1. 中国电信股份有限公司广州研究院, 广东 广州 510630;

2. 中国移动通信集团南方基地, 广东 广州 510640)

摘要: 随着容器技术逐渐成熟, 越来越多的互联网企业和 IT 企业将核心业务平台迁移到容器环境中。Kubernetes 经过 3 年多的快速发展, 已成为业界首选的容器集群管理平台。从数据库容器化角度出发, 在分析基于 Kubernetes 实现数据库容器化承载的关键技术基础上, 给出了 MySQL 数据库及其集群的容器化方案。通过对多种数据库模型进行性能和可靠性等方面的评估比较, 给出了数据库容器化技术可能遇到的问题及不同场景下的数据库容器承载方案建议。

关键词: 数据库; 容器; Kubernetes; 集群

中图分类号: TP399

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2018311

Containerizing solution based on Kubernetes of database and its cluster

YAN Liyun¹, HE Zhenwei¹, YANG Xinzhang¹, ZHANG Ling¹, HOU Shaoxin²

1. Guangzhou Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China

2. China Mobile South Base, Guangzhou 510640, China

Abstract: With the maturity of container technology, more and more internet companies and IT companies have begun to migrate their core business platforms to the container environment. After more than three years of rapid development, Kubernetes has become the industry's preferred container cluster management platform. The containerization scheme of MySQL database and its cluster was given from the perspective of database containerization, based on the analysis of the key technology of database containerization on Kubernetes. Through the evaluation and comparison of performance and reliability of various database models, problems that database containerization technology may encounter and suggestion of database container hosting scheme under different scenarios were proposed.

Key words: database, container, Kubernetes, cluster

1 引言

随着 IT 架构的演进, 从传统的巨石型单体应用到服务化, 再到更细粒度的微服务化, 数据库

数量、容量和业务需求也发生了变化, 工作重点从数据库内核到自动化运维, 从精细化运维到集群化运维, 从关注个别库的性能到关注集群的可用性, 从依靠个人的能力到借助监控平台和自动

收稿日期: 2018-07-23; 修回日期: 2018-12-10



化运维脚本,数据库即服务(database-as-a-service, DBaaS or RDS)的需求越来越强烈,而面对虚拟化技术在实现分布式数据库上的短板,如计算密度难以提高、存储开销大、迭代成本高等,容器是资源利用率更高、整合效率更高的分布式数据库实现方式。

容器是一种基于系统内核的轻量级虚拟化技术,可以在单一宿主机上同时提供多个拥有独立进程、文件和网络空间的虚拟环境(即容器),是继虚拟机之后的更加轻量级的应用承载方式;同时,容器也是一种敏捷的应用交付技术,将应用依赖的软件栈整体打包,以统一的格式交付运行^[1]。

2 运营商数据库容器化驱动力

运营商随着业务量膨胀,业务节点大规模扩张,IT系统对数据库资源提出了更高的要求。目前,运营商的IT系统有大量中小规模数据库,存在资源利用率低、资源利用不充分、管理分散等问题,运营商希望借助容器改善现存资源的利用率问题,通过统一的容器集群管理平台实现大规模数据库容器集中管理,实现数据库容器随着业务需求快速、弹性的扩/缩容;对于大型数据库集群,目前存在部署/配置/扩展/维护困难、自动化程度低等问题,希望借助容器的应用交付技术以及自动化运维工具,实现数据库集群自动化部署、运维。对数据库服务的要求可总结如下:

- 所有的日常运维工作自动化;
- 高可靠,数据零丢失,安全的需求依然是刚需;
- 通过弹性扩展追求更高的资源利用率;
- 不再追求极致的高性能。

同时,随着运营商公有云上的业务增长,业务流量占比越来越大,数据库上云需求日益迫切,将数据库资源服务化,提供不同集群类型的部署容器模板,实现任何规模下可靠性能的数据库访问,并进一步实现无服务器数据库托管。行业内,

公有云巨头已实现数据库服务容器化,如AWS、Google、阿里云,此外数据库霸主Oracle也在GitHub上推出12C企业版Docker镜像。已证明了数据库可高效地运行在容器里。而如何管理大规模容器化数据库这一点还有待进一步验证。目前国内运营商纷纷搭建自有的容器云管理平台,且容器云管理平台大多采用基于Kubernetes的容器集群系统,基于Kubernetes的数据库容器化方案有待进一步验证,包括与物理机承载性能的比较、对可靠性的影响以及带来的部署、运维及架构的变化等。

3 容器技术的数据库应用分析

Kubernetes技术目前提供了若干可用于支撑数据库等有状态集群部署管理的机制,以简化数据库集群部署维护。

3.1 数据库代理与通信

数据库集群通常在应用与数据库节点之间引入一层数据库代理,通过数据库代理为应用提供数据库节点的透明访问,并通过代理实现数据库的负载均衡、读写分离和高可用。在Kubernetes中,数据库代理方案有以下几种。

(1) service

提供虚拟IP地址或域名访问,通过iptables的DNAT机制将虚拟IP地址映射成实际的Pod IP^[2],可支持简单的负载均衡和读写分离策略,性能高、时延短,缺点是不支持复杂的数据库分流策略,不支持超时重发,依赖于数据库Pod的状态监测。

(2) headless service

没有虚拟IP地址的service,只支持域名访问,通过ETCD实时查看Pod IP,与service的特点类似,由于不需要配置iptables,Pod映射规则更新时间比service更短,而由于需运行时进行PodIP查看,比service的本地DNAT产生更多时延。此外,headless service与有状态集结合,可以为有状

态集的 Pod 提供固定的网络标识, 这为数据库集群内部节点间通信提供了便利。

(3) 独立代理

通过独立部署的 MyCat、ProxySQL 等专业数据库代理实现数据库流量转发, 可提供诸如读写分离、数据分片、主从切换等高级数据库分流与控制功能, 通过可以结合 Kubernetes 的 service 机制实现 MyCat 等数据库代理的高可用。其缺点是由于增加了独立的代理层, 其性能比 service 代理方式要低一些。

(4) 旁挂代理

利用 Pod 的多容器部署特性, 将数据库代理以旁挂容器 (side-car container) 的形式与主数据库容器部署在同一个 Pod 中, 数据库代理与 Pod 内部的数据库直接使用高效的本地网络通信, Twitter 开源的 Vitess 集群和腾讯开源的 PhxSQL 集群均可采用旁挂式代理模式部署。

3.2 数据库集群编排

数据库集群涉及数据库服务器、数据库代理、数据库监控程序等诸多部件, Kubernetes 提供了多种容器编排模型可以简化数据库集群部署。

(1) 旁挂容器

利用 Pod 的多容器部署特性, 将数据库作为主容器, 将代理容器、监控容器作为旁挂容器打包成一个 Pod 部署, 数据库容器与旁挂容器之间共享网络栈与存储。

(2) 副本集 (replicaset)

无状态容器集, 用于管理一组完全等价的无状态 Pod, 由 Kubernetes 自动保证 Pod 的可用实例数符合预期, 可用于无服务状态的数据库集群组件^[3], 如数据库代理, 也可用于单实例数据库, 通过 RS 的实例数保证机制保证单实例数据库的高可用。

(3) 有状态集 (statefulset)

有状态容器集, 用于有序地调度一组有状态 Pod, 这些 Pod 采用“服务名+序号”的形式命名,

拥有稳定的网络标识, 按照严格的次序启动和停止^[3]。可用于数据库集群节点编排, 例如用于主从数据库可以保证主库 Pod 成功启动后再启动从库 Pod。

(4) 节点选择器

基于节点标签将 Pod 调度到满足特定配置要求的节点上^[2], 可为数据库容器实例绑定合适的 Kubernetes 节点。

(5) 守护集 (daemonSet)

将某个 Pod 部署到整个 Kubernetes 集群上或基于节点选择器绑定的一组节点上, 可简化数据库监控 Pod、日志采集 Pod 的部署^[3]。

3.3 数据存储

数据库的数据存储主要有本地存储和共享存储两种方案, 本地存储性能更高, 但难以管理, 共享存储可靠性更高, 但性能较差。

(1) 持久卷 (PV persistent volume)

持久化共享存储, 将数据库的数据保存在后端的网络存储中, 可供不同的数据库服务器共享, 利用存储集群自身机制保证数据的高可用性, 缺点是性能较差, 主要用于单实例数据库的可靠存储。

(2) 持久逻辑卷 (PLV persistent local volume)

持久化本地存储, 直接使用 Kubernetes 节点上本地存储保存数据库数据, Kubernetes 1.9 提供 LVM (local volume management) 机制可本地存储作为集群资源统一管理, 主要用于数据库集群的数据库数据高速存储。

4 容器化数据库集群方案

MySQL 数据库在中小型数据库应用场景中, 单实例/主备、主从集群和主主集群是最常见的 3 种部署架构, 3 种部署架构各有特点及其适用场景, 具体见表 1。本文针对这 3 种架构结合 Kubernetes 能力, 分别采用 MyCat、Kubernetes service 作为代理, 实现了 MySQL 数据库的容器化承载方案。



表 1 3 种数据库模型对比

	单实例数据库	主从数据库集群	主主数据库集群
部署难易程度	简单	较难	较难
读写性能	读性能低，写性能高	读性能高，写性能低	读写性能高
可靠性	低	中	高
适用范围	中小型数据库、可靠性要求低	大型数据库、可靠性要求高	大型数据库、业务可靠性要求高
实例数	1	≥2	≥2

4.1 MySQL 单实例

由单一的数据库实例提供服务，特点是部署简单、数据库读写性能相对较差（依赖于具体的存储后端）、可靠性较高。图 1 中数据库以副本集或有状态集形式部署均可，通过 Kubernetes service 提供稳定端口服务，数据可存储在 IPSAN、NFS 等共享存储上，通过共享存储保障单实例数据库的可用性。

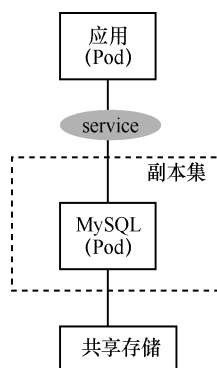


图 1 数据库单服务器

4.2 MySQL 主从集群

主从集群数据库的特点是其结构为一个主数据库与多个从数据库，主数据库支持读写操作，从数据库只支持读操作，主库与从库间的数据同步^[4]通过 MySQL 自身的 BinLog 复制机制实现。应用将访问代理层，由代理层进行负载均衡和分发。主从集群方案其代理层可以采用专用数据库代理，常见的代理为 MyCat，正如上文所述 Kubernetes service 也有负载均衡和分发作用，可以部分替代专用数据库代理的作用。为了对比两

类方案的差异，本文分别选取 MyCat 代理与 Kubernetes service 代理方式进行对比测试，其容器化部署方案分别设计如下。

(1) 基于 MyCat 代理

部署模型如图 2 所示，主数据库和从数据库均采用 Kubernetes 的有状态集模型部署，以 Pod 形式运行，主节点作为有状态集的 1 号 Pod，利用开源分布式数据库中间件 MyCat 实现主从数据库的读写分离和自动切换。MyCat 以副本集形式部署，绑定 Kubernetes service，向应用提供代理层虚拟 IP 地址，数据库主与从之间的通信采用 headless service。

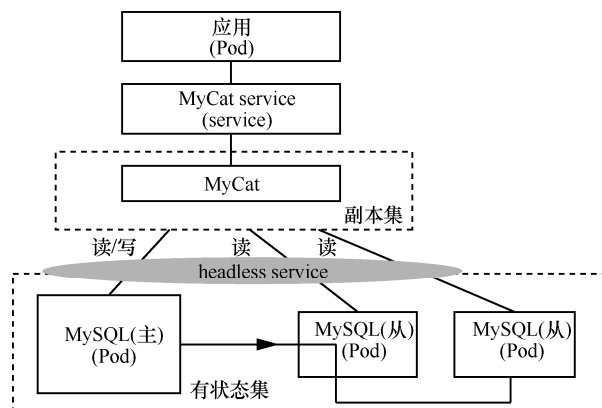


图 2 基于 MyCat 代理的主从集群

(2) 基于 Kubernetes service 代理

部署模型如图 3 所示，数据库采用有状态集模型部署，分别为主库和从库创建一个 Kubernetes service：主库服务作为主数据库的代理服务，支持数据库读/写操作；从库服务作为从数据库的代理服务，只支持数据库读操作。主库、从库切换

需要手工更新 service 配置。

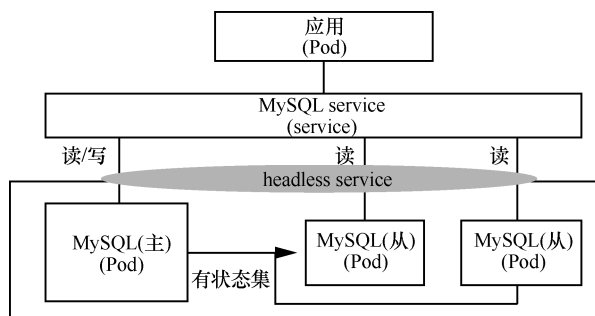


图3 基于 Kubernetes service 代理的主从集群

4.3 MySQL 主主集群

MySQL 主主集群通过 Galera 实现，每个 MySQL 实例可被业务系统读写访问，MySQL 数据实例之间的数据同步由 Galera 的同步写机制保证。本次实验同样将对比以 MyCat 这类专用数据库代理作为代理层和以 Kubernetes 自身 service 部件为代理层，对数据库性能和可靠性的影响，其容器化部署方式分别如下。

(1) 基于 MyCat 代理

部署模型如图4所示，MySQL 主主集群采用 MyCat 作为数据库代理，数据库与 Galera 集群模块打包到 Pod 运行，采用有状态集方式部署。

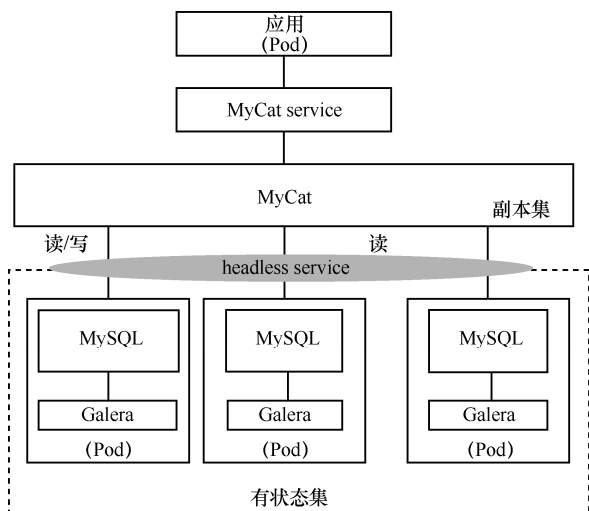


图4 基于 MyCat 代理的主主集群

(2) 基于 Kubernetes service 代理

部署模型如图5所示，MySQL 主主集群采用

Kubernetes 的有状态集部署，数据库与 Galera 集群模块 1:1 打包到 Pod 运行，通过 Kubernetes 的标准 service 机制实现数据库访问的负载均衡。

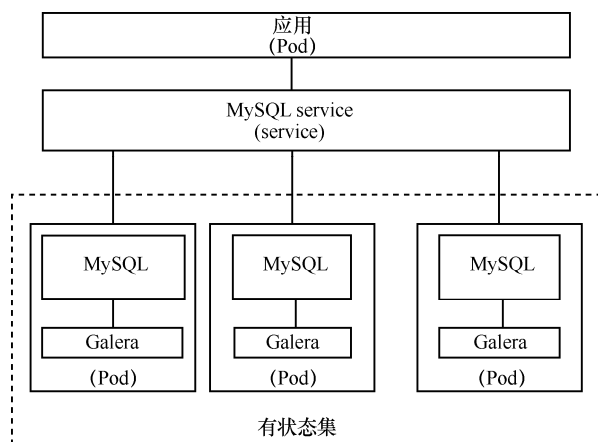


图5 基于 Kubernetes service 代理的主主集群

5 测试过程及结果分析

5.1 性能测试

性能测试以 SysBench 对 MySQL 数据库作基准测试，SysBench 是跨平台的基准测试工具，支持多线程，支持多种数据库^[5]。采用混合读写模式、纯读和纯写这3种模式，测试数据量为单表2 000万，共8个表，每个基准测试时间为600 s，分别对应基准测试脚本为 oltp.lua、select.lua、update.lua，其中对混合读写基准测试脚本进行修改，采用1写4读的方式。基于上述 Kubernetes 容器化承载方案，分别对不同场景下的 MySQL 数据库从物理机承载和容器承载两方面进行了性能对比评测。测试步骤如下：

步骤1 启动创建 Kubernetes 资源的 shell 脚本，例如 ./startGalera.sh；

步骤2 在 sysbench 容器执行初始化数据的 shell 脚本，例如 ./stat_init.sh；

步骤3 数据初始化完成后，运行基准测试的 shell 脚本，例如 ./stat_run.sh；

步骤4 测试完成后，需要清理初始化数据，就执行 ./stat_clean.sh。



测试部署架构如图 6 所示, 基于 Kubernetes 的容器环境分别涵盖了单数据库服务器模式、集群（主主、主从）数据库模式、service 代理、MyCat 代理模式, 测试软件环境见表 2, 硬件环境见表 3。

表 2 软件环境

用途	软件项目	版本
操作系统	CentOS 7	1511
文件系统	XFS	
容器引擎	Docker	1.12.6
容器集群管理	Kubernetes	1.6.2
网络方案	Flannel(host-gw)	0.5.5
数据库软件	MySQL	5.7
测试工具	SysBench	1.1.0、0.5
数据库集群	主从集群、Galera	5.7
数据库代理	MyCat、service	1.4.3

表 3 硬件环境

硬件名称	硬件信息
CPU	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40 GHz (32 核)
内存	128 GB
内核	Linux 3.10.0-327.el7.x86_64
网卡信息	10GE
硬盘	Intel DC S3510 2.5" 800 GB SATA III SSD HP Vx0800geyjt, 800.1 GB

(1) 单实例性能测试结果与分析

测试结果方面, 纯写性能如图 7 所示, 混合读写性能如图 8 所示, 纯读性能如图 9 所示, 单实例性能测试结论如下:

- 在写密集型场景（纯写、混合读写）中, 物理机和容器性能大致相当;
- 在读密集型场景中, 小并发读容器性能略优于物理机, 大并发读物理机优于容器;

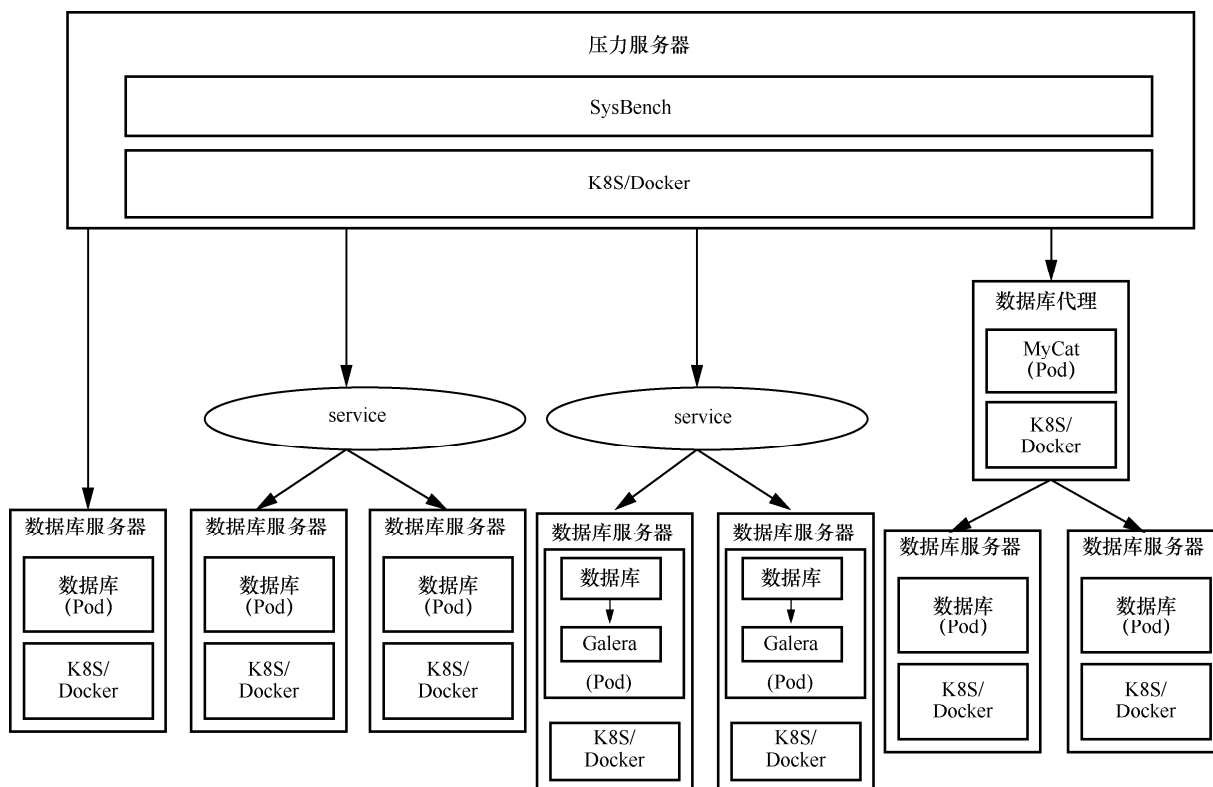


图 6 部署架构

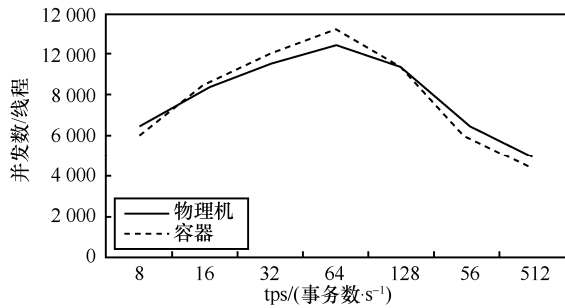


图7 纯写性能

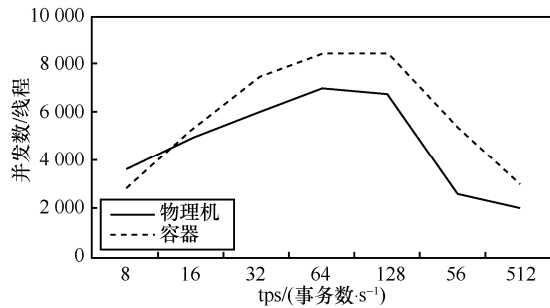


图8 混合读写性能

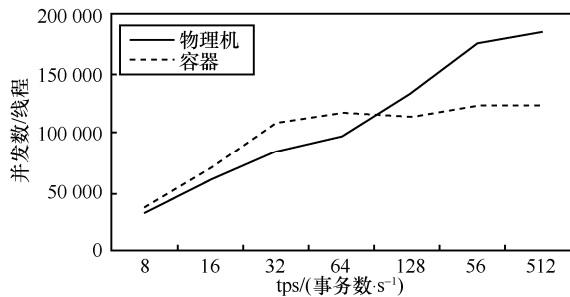


图9 纯读性能

- 写密集型场景的主要资源瓶颈为磁盘 I/O，读密集型场景的主要资源瓶颈为 CPU；
- 写密集型数据库与读密集型数据库存在混合部署可能。

(2) 双实例性能测试结果与分析

(3) 测试结果方面，混合读写性能如图 10 所示，纯读性能如图 11 所示，纯写性能如图 12 所示。

双实例性能测试结论如下：

- 在写密集型场景下，容器承载与物理机承载的同类数据库集群性能大致相当；
- 在读密集型场景下，小并发容器承载性能与物理机大致相当，大并发物理机承载性能优于容器性能 10%~20%；

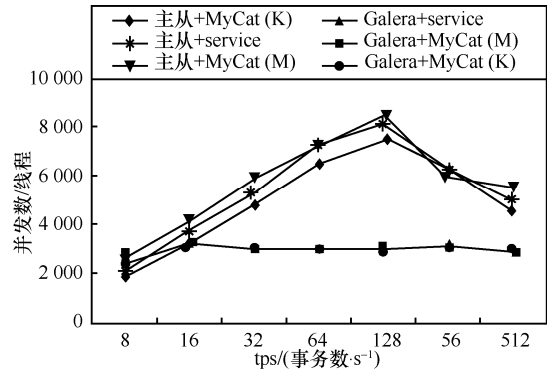


图10 混合读写性能

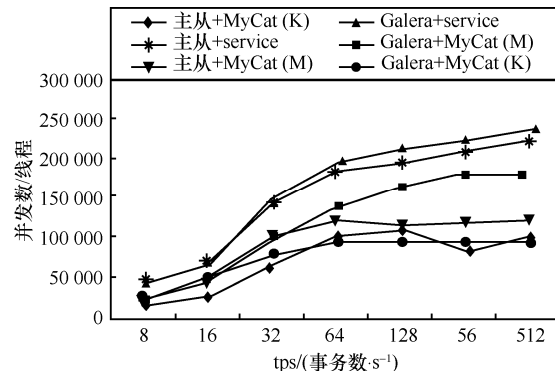


图11 纯读性能

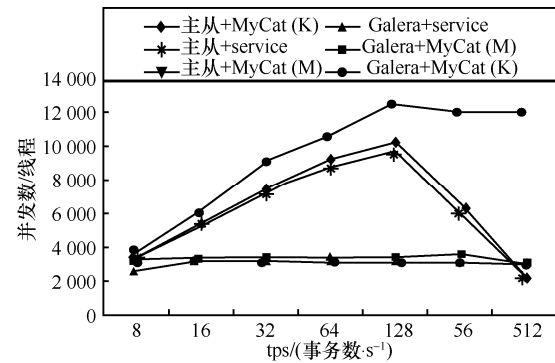


图12 纯写性能

- Kubernetes service 的性能在写密集场景下与物理机大致相当，在读密集场景下比 MyCat 快 20%~30%。

5.2 数据库高可用测试

主要对主备切换、单实例容器恢复、service 代理主主切换、MyCat 代理主主切换所产生的业务中断时间和集群恢复时间进行测试，测试客户端每隔 10 ms 向数据库插入和查询 1 条记录，业务中断时长为查询和插入均中断的时长，业务降



级时间为插入中断而查询正常的时长，集群恢复时间为集群节点均恢复正常访问的时间。

(1) 主备切换

主库、备库共享存储，通过 Keepalive 保持同步，主库宕机后，备库通过 Keepalive 检测到主库失效自动切换成主库。

(2) 单实例容器恢复

数据库以副本集形式部署，数据保存在共享存储中，通过容器健康检查机制判断数据库容器失效后通过副本控制器自动将数据库容器调度到新的节点上。

(3) 主主切换（service 代理）

采用 Kubernetes service 代理模式部署 Galera 集群，通过容器健康检查机制识别数据库容器失效更新容器状态，service 代理将数据库请求路由到状态为就绪（ready）的数据库容器上。

(4) 主主切换（MyCat 代理）

采用 MyCat 代理方案部署 Galera 集群，通过 MyCat 自动检测数据库容器状态，MyCat 代理将数据库请求路由到正常的数据库容器上。

测试结果见表 3，数据库可靠性测试结论如下：

表 3 可靠性测试结果

故障场景	业务中断 时间/s	业务降级 时间/s	集群恢复 时间/s
主备切换（物理机）	0.5~2	0	30~120
单实例容器恢复	4~7	0	1~3
主主切换(service 代理)	1~3	0	30~120
主主切换(MyCat 代理)	0	0	30~120

(1) 单实例数据库的容器自动恢复时间需要重启数据库容器，恢复时间比主备集群慢 3~5 s；

(2) Kubernetes service 依赖于数据库容器健康检查和状态更新，且不具备自动重发机制，即使采用集群数据库部署也会产生 1~3 s 的业务中断；

(3) 专业数据库代理具备自动故障检测能力和超时自动重发机制，基本不会出现业务中断

情况；

(4) 主从集群在主从切换期间会导致业务降级（即可读不可写），而主主集群不会出现业务降级，因此主主集群的可靠性更高。

5.3 结论分析

基于 Kubernetes 的数据库容器集群承载方案已经较为成熟。容器集群承载方案与物理机承载方案在性能、可靠性、资源损耗方面大致相当，但是在部署、维护、扩张性方面，前者比后者有优势。具体结论如下：

- 在性能方面，容器化数据库性能在大部分场景下与物理机大致相当，仅在某些大并发读场景下弱于物理机；
- 在资源损耗方面，容器和物理机承载方式几乎无差别；
- 在可靠性方面，容器集群自身容错机制能为小型数据库提供一定程度的可靠性保障，大型数据库主要依靠数据库集群自身机制保障高可用，目前 Galera 多主集群的可靠性最高；
- 在部署方面，通过容器集群自带的编排、模板和升级机制能够简化容器部署；
- 在数据库架构方面，Kubernetes 自带的 service 可替代简单的数据库代理，提升数据库集群性能。

6 结束语

在数据库容器化承载过程中，有以下建议可供参考：容器集群自身容错机制能为小型数据库提供一定程度的可靠性保障，大型数据库主要依靠数据库集群自身机制保障高可用。采用 Kubernetes service 代理的方式较专用代理方式性能优且简化部署架构，但主从切换时延较长，适用于对业务中断不敏感、中小型数据库场景，可提升数据库集群性能；在对业务连续性要求高、大型数据库场景中，建议采用专用代理的容器化

方案;多主集群的可靠性最高,可以进一步运用在可靠性要求高的数据库模型中。

参考文献:

- [1] 何宇锋,林奕琳,张琳峰.容器技术在移动核心网的应用[J].移动通信,2018,42(3): 27-32.
HE Y F, LIN Y L, ZHANG L F. The application of container technology in the mobile core network [J]. Mobile Communications, 2018, 42(3): 27-32.
- [2] 龚正,吴治辉,叶伙荣,等. Kubernetes 权威指南(第一版)[M]. 北京:电子工业出版社,2016.
GONG Z, WU Z H, YE B R, et al. Kubernetes authoritative guide: 1st edition[M]. Beijing: Publishing House of Electronics Industry, 2016.
- [3] 张勋,张呈宇,魏进武.一种基于容器编排技术的资源运维系统的设计与实现[J].电信科学,2018,34(6): 162-170.
ZHANG X, ZHANG C Y, WEI J W. Design and implementation of a resource operation and maintenance system based on container arrangement technology[J]. Telecommunications Science, 2018, 34(6): 162-170.
- [4] 斯维特·斯米尔诺娃. MySQL 排错指南[M]. 北京:人民邮电出版社,2015.
SMIRNOVA S. MySQL troubleshooting guide[M]. Beijing: Posts & Telecom Press, 2015.
- [5] 邱红飞,李先绪,黄春光.云计算数据库的测试技术研究与应用[J].电信科学,2013,29(10): 16-20.

QIU H F, LI X X, HUANG C G. Research and application of testing technology of cloud computing database[J]. Telecommunications Science, 2013, 29(10): 16-20.

[作者简介]



严丽云(1985-),女,中国电信股份有限公司广州研究院工程师,主要从事云计算虚拟化、容器技术研发方面的工作。

杨新章(1972-),男,博士,中国电信股份有限公司广州研究院工程师,主要从事云计算、物联网技术研发方面的工作。

何震苇(1976-),男,中国电信股份有限公司广州研究院工程师,主要从事终端应用和云计算平台研发方面的工作。

张凌(1978-),女,中国电信股份有限公司广州研究院工程师,主要从事云计算网络、容器技术、IDC研究方面的工作。

侯韶新(1987-),男,现就职于中国移动通信集团南方基地,主要从事IDC运维、研究方面的工作。