容器平台的存储性能评估方法研究

IAGS SSE CSE Storage Team

Tingjie Chen, Vivian Zhu

前言

可能你遇到过下面的场景：

1. 你们公司刚刚购买了英特尔傲腾存储硬件，你设计了一个加速业务的使用场景，觉得用起来还不错，但是需要有个全方位的性能评估报告。
2. 你负责维护一个大型的存储集群，引入了新的缓存架构对系统存储的瓶颈进行优化，你首先要要设立一个基准并搭建一套快速易用的评估工具，并且可以反映出当前主流的工作负载。
3. 你是一名运维工程师，为了提升效率，研发团队引入了 Kubernetes 集群取代原有的闭源部署方案，很多业务都开始向云原生转换，运维系统也需要将一些基准测试工具容器化。原有的存储系统如何跟容器化的业务和基准测试工具无缝集成？

希望本篇白皮书介绍的基于容器平台的存储性能的评估方法有助于您解决类似的问题。

典型场景

模拟复杂基准测试

在存储业界广泛使用的微基准测试程序比如 FIO、VdBench 等，这些微基准测试工具配置灵活，模式简单，可以快速在某个技术点验证结论，但是它们不能反映出复杂生产环境下的真实负载。模拟复杂的基准测试填补了这一缺憾，它定义了特定的工作负载模型，用操作数据库的方式模拟出各种基准场景，并且这些模拟的行为是可以重现的。

总的来说，微基准测试和模拟复杂基准测试系统可以互为补充，它们使用的条件和情况也有所不同。模拟复杂基准测试适合在下面几个场景下使用：

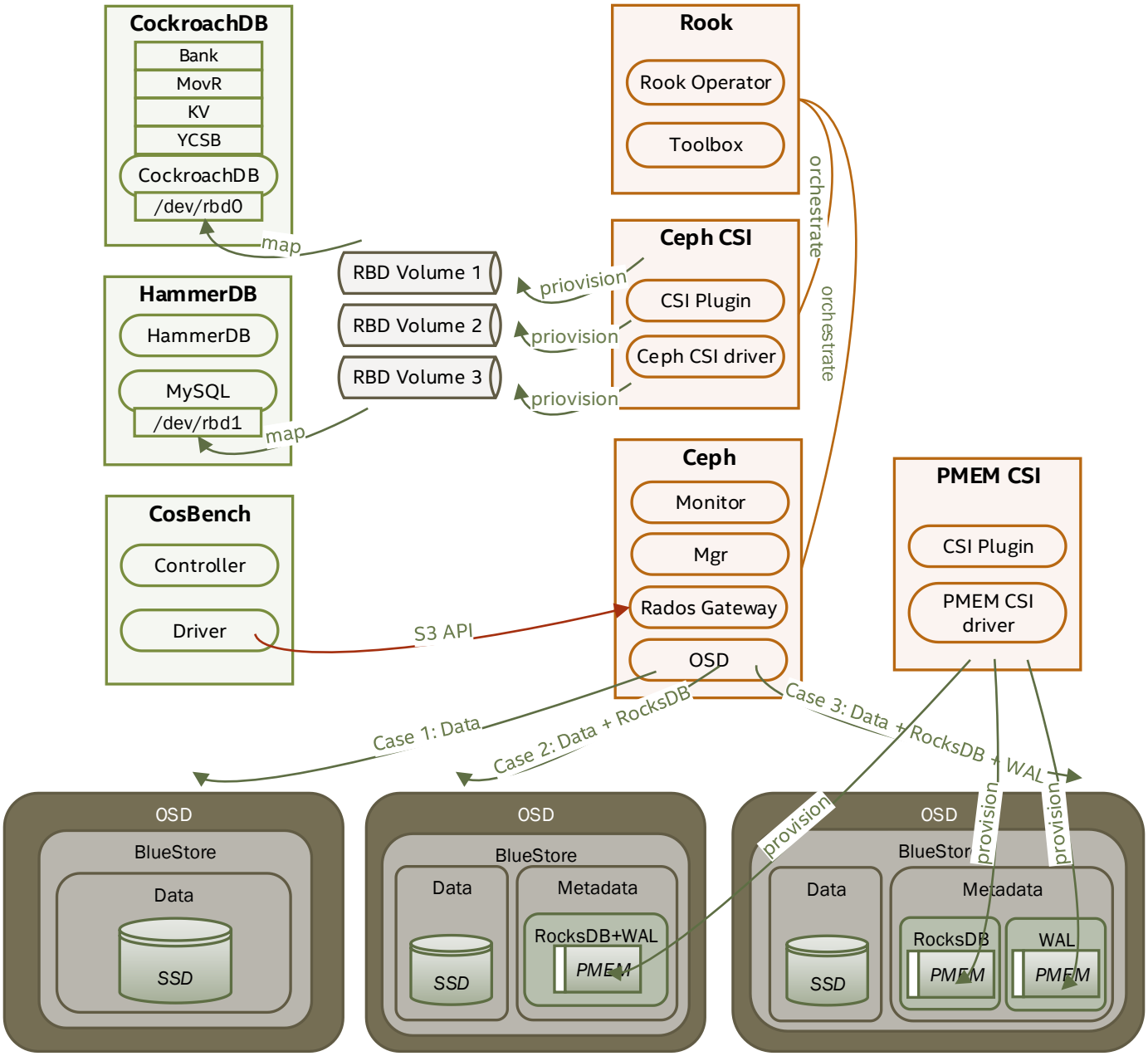
- 综合评估存储系统。评估系统的性能天花板，瓶颈以及查找可能存在的缺陷。这个场景相对复杂，本篇白皮书暂不讨论。
- 引入新解决方案或重要功能。比如更换新的硬件，存储系统的重大版本升级，引入新的解决方案架构或者将原来的系统切换到新的系统中。本篇白皮书会重点讨论其中的两个场景。
- 客户要求。客户有在线的业务场景，我们需要模拟这些场景来进行相关的优化。

我们为此引入了下面一组面向生产环境的复杂基准测试集 CockroachDB 和 HammerDB，并将它们容器化部署在 Kubernetes 集群中。此外 CosBench 评估的对象存储需求日益旺盛，这是我们引入此基准的原因。根据 IDC 的数据显示，企业级数据中的 80%是非结构化数据，而 75%将会存放到对象存储上。

基准测试	类型	特点	工作负载	评估标准
CockroachDB	数据库内置基准	CockroachDB是一个可伸缩，跨区域复制，并支持事务，高可用性和高度一致性的分布式SQL数据库。它带有内置的负载生成器，用于模拟不同类型的客户端工作负载。	银行：使用货币平衡表为一组账户建模。 键值：在整个集群中均匀随机地读取和写入键值对。 MovR：模拟MovR示例应用程序的工作负载。MovR是一家虚构的车辆共享公司，旨在展示CockroachDB功能。 TPCC：使用多个丰富的模式模拟联机事务处理工作负载。 YCSB：使用自定义功能，模拟大量读，写或基于扫描的大规模键值工作负载。	Ops: 每秒处理的操作 平均等待时间
HammerDB	联机事务处理 联机分析处理	HammerDB是基准测试和负载测试套件，支持当前流行的Oracle数据库、SQL Server、IBM DB2、MySQL、MariaDB、PostgreSQL和Redis等。它支持基于TPC-C和TPC-H的工作负载。	TPC-C是衡量OLTP (联机事务处理)的系统工业标准，是行业中公认的最权威和最复杂的在线事务处理基准测试。它通过模拟仓库和订单管理系统，测试广泛的数据库性能。 TPC-H是决策支持基准，也叫OLAP (联机分析处理) 基准， 它由一套面向业务的临时查询和并发数据修改组成.	TPC-C: transactions-per-minute-C (tpmC) and associated price-per-tpmC($/tpmC) TPC-H: Composite Query-per-Hour Performance Metric (QphH@Size) and associated price-per-QphH($/QphH@Size)
CosBench	对象存储	CosBench是用于衡量云对象存储服务性能的分布式基准测试工具，它支持Amazon S3和OpenStack Swift等主流对象存储接口。	配置文件定义对象的读写等操作。	吞吐量，带宽和响应时间

以 Kubernetes 为代表的容器环境越来越广泛的部署在生产环境中，本白皮书提供了容器环境下的两个典型场景，基于新硬件 -- 英特尔傲腾非易失性内存硬件和新的解决方案 -- Open CAS 缓存框架对分布式存储系统 Ceph 及上层业务的加速。

场景一：用傲腾非易失性内存加速 Ceph 及上层业务

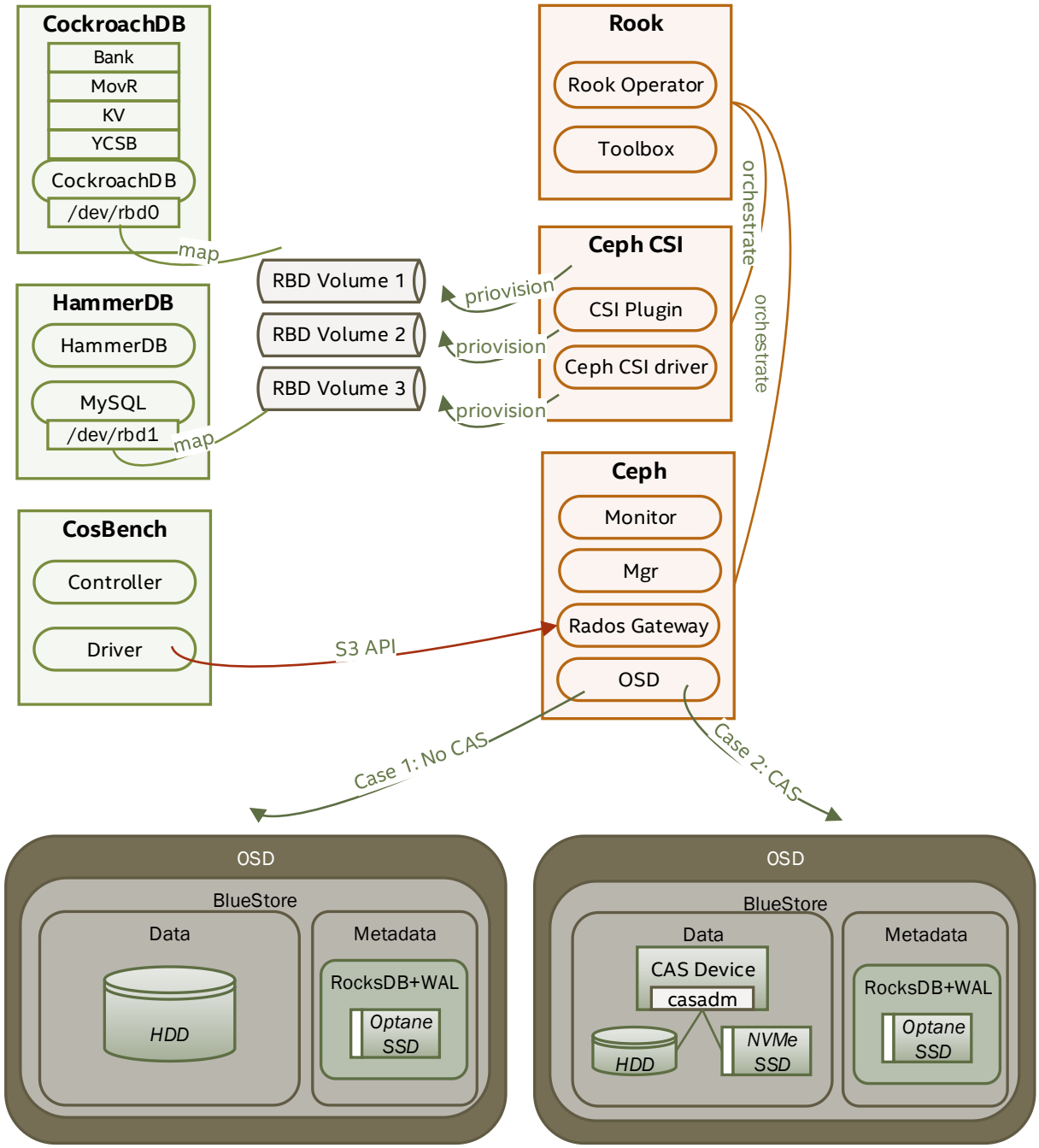


容器化的环境下，通过 Rook 部署 Ceph 集群，Ceph 集群的存储服务 OSD 有三种部署方案：

- SSD 直接作为存储引擎 BlueStore 的单独分区。
- SSD 作为 BlueStore 的主数据分区，而元数据 (RocksDB 和 WAL) 部署在英特尔傲腾非易失性内存上。
- SSD 作为 BlueStore 的主数据分区，元数据分 (RocksDB 和 WAL) 开并单独部署在英特尔傲腾非易失性内存上。

傲腾非易失性内存无法直接作为块设备使用，需要通过 PMEM CSI（一种容器存储接口驱动）提供面向 Kubernetes 容器的存储卷类型，PMEM 存储卷作为 Ceph BlueStore 的元数据分区。同样，Ceph 通过 Ceph CSI（Ceph 的容器存储接口驱动）提供 RBD 块设备给上层容器应用，基准测试程序以 Kubernetes 应用服务的方式部署，并挂载 RBD 块设备，通过执行基准的工作负载评估 Ceph RBD 块存储及 Ceph Rados Gateway 对象存储的性能。

场景二：基于 Open CAS 缓存架构加速 Ceph 及上层业务



Open CAS 作为新的方案应用于 Ceph OSD 上，两种方案的对比：

- HDD 作为 BlueStore 的主数据分区，而元数据 (RocksDB 和 WAL) 部署在英特尔傲腾 SSD 上。
- HDD 和 NVMe SSD 通过 Open CAS 软件组合成新的 CAS 设备作为 BlueStore 的主数据分区，元数据 (RocksDB 和 WAL) 部署在英特尔傲腾 SSD 上。

Open CAS 是英特尔开发的缓存加速方案，通过加载内核的方式，将高速介质盘作为缓存，和慢速盘 “融合” 为一块盘使用，从而提高系统整体的磁盘读写性能。

性能评估环境

软硬件环境

针对上面的两种典型的场景，我们搭建一套兼容性的环境，共享硬件和软件配置。在服务器端，以 Rook Ceph 为核心的存储集群提供块存储和对象存储服务；在客户端部署基准测试程序，基于块存储和对象存储之上运行工作负载。

硬件配置		
	- **CPU**	3 * Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
	- **内存**	3 * 8 * 16 GB
	- **网络交换机**	100 Gb
	- **傲腾非易失性内存**	2 * 512 GB
	- **傲腾 SSD**	2 * Intel P4800X 750 GB
	- **NVMe SSD**	6 * Intel P4510 1 TB
	- **HDD**	8 * 1.2 TB
软件配置		
	- **操作系统**	Ubuntu 18.04 LTS
	- **内核版本**	Linux version 4.15.0-109-generic
	- **Docker 版本**	19.03.8
	- **Kubernetes 版本**	1.18.8
	- **Ceph 版本**	Octopus 15.2.2
	- **Ceph 集群**	2 服务节点，1 客户端
	- **Rook 版本**	1.3.3
	- **Open-CAS 版本**	20.06.00.00000703
	- **CockroachDB 版本**	20.1.1
	- **HammerDB 版本**	3.3
	- **MySQL 版本**	8.0.19
	- **CosBench 版本**	0.4.2
场景一		
	• **每个 Ceph 节点**	
	- **傲腾非易失性内存**	512 GB
	- **NVMe SSD**	3 * 1.0 TB
	- **OSD 个数**	6 (SSD:OSD=1:2)
场景二		
	• **每个 Ceph 节点**	
	- **傲腾 SSD**	3 * 750 GB
	- **NVMe SSD**	3 * 1.0 TB
	- **HDD**	4 * 1.2 TB
	- **OSD 个数**	4 (HDD:OSD = 1:1)

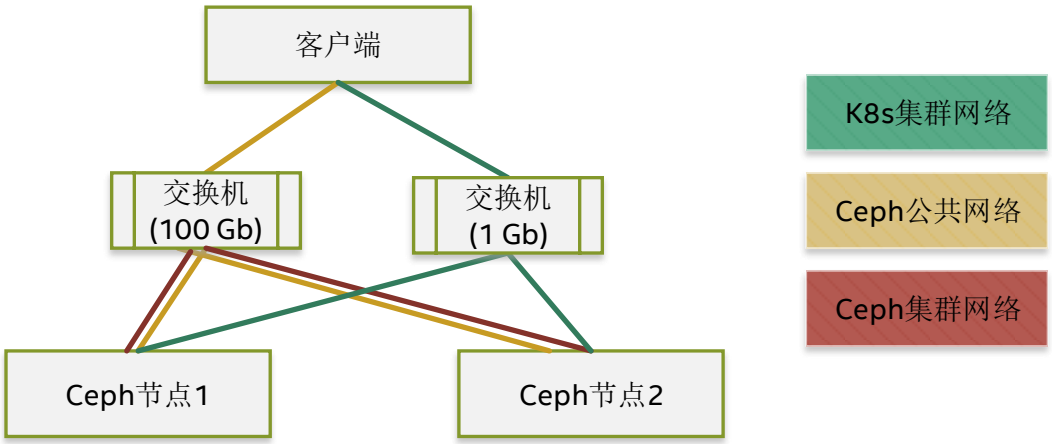
Kubernetes 和 Rook Ceph 存储集群

我们通过 kubeadm 工具部署三个节点的 Kubernetes 集群，其中两个节点是 Ceph 的服务节点，一个节点是客户端。

Rook 部署基于 Kubernetes 集群之上，通过 Rook Operator 将 Ceph 的模块以 Pod 的形式部署在两个 Ceph 节点上，因为是两个节点，我们部署两个 Ceph Monitor，设置 Replication 为 2，实现了数据冗余和性能的平衡。

Rook 也支持 Ceph 部署在两个网络上，公共网络和集群网络 (使用主机网络) 的配置可以通过 configMap override 的方式，也可以通过定义 Ceph 集群的 yaml 网络配置 (功能不太成熟稳定)，我们采用第一种方式。

客户端上以 StatefulSet Pod 的形式部署三种基准测试程序。对于块存储，通过在 yaml 里定义 PersistentVolumeClaim 向 Ceph CSI 申请；对于对象存储，以 S3 API 的方式向 Ceph RGW 获取对象。



需要注意的是 Rook 集群的 OSD 的配置有两种方式：

- 直接定义每个节点上的设备路径。这种方式简单，适合轻量部署的场合，但扩展性不太好，而且不支持远端的公有云存储。
- 通过 PVC (PersistentVolumeClaim) 的方式，需要定义 PVC 模板，然后填充每个 OSD 类型所需的 StorageClass 即可。这种方式扩展性好，支持本地和远端的各种存储。

以第二种 PVC 的方式为例，配置文件如下：

```
volumeClaimTemplates:
- metadata:
    name: data
  spec:
    resources:
      requests:
        storage: 64Gi
    # IMPORTANT: Change the storage class depending on your environment (e.g. local-storage, gp2)
    storageClassName: local-ssd
    volumeMode: Block
    accessModes:
      - ReadWriteOnce
- metadata:
    name: metadata
  spec:
```

```
    resources:
      requests:
        storage: 5Gi
    # IMPORTANT: Change the storage class depending on your environment (e.g. local-storage, io1)
    storageClassName: local-pmem
    volumeMode: Block
    accessModes:
      - ReadWriteOnce
- metadata:
    name: wal
  spec:
    resources:
      requests:
        storage: 5Gi
    # IMPORTANT: Change the storage class depending on your environment (e.g. local-storage, io1)
    storageClassName: local-pmem
    volumeMode: Block
    accessModes:
      - ReadWriteOnce
```

我们定义了 BlueStore 的三个分区的 PVC 模板，分别对应 data, block.db 和 block.wal 分区，每个分区对应一种类型的 StorageClass，如果通过 PVC 能申请到相应大小的块设备，就创建成功，组建一个新的 OSD 组件服务。本样例中，我们通过 local-storage 内置类型创建了多个 PV (Persistent Volume)，然后 PVC 选取符合大小的 PV 就可以绑定成功。

对于客户端块设备的获取，标准方式是通过存储容器接口 (CSI) 的方式，Rook 的部署目前支持 Ceph CSI，正常的加载过程需要确保两个条件：

- CSI 服务，包括插件和驱动正常运行，默认 Rook 会启动 CSI 服务
- 创建对应的 StorageClass: rook-ceph-block，其中 provisioner 配置项是: rook-ceph.rbd.csi.ceph.com

CockroachDB 基准

CockroachDB 的工作负载内置在数据库安装包里，基准测试的使用也非常简单，对于它的每一个负载，有两个典型的步骤：

步骤	功能	配置
init	加载工作负载schema	(--drop) 删除当前的数据库。
run	运行工作负载	(--duration) 运行的持续时间，带有时间单位后缀。

CockroachDB 的基准每种类型的工作负载都有一些参数可以定义，比如数据集大小，并发度等参数，像本示例一样保持默认参数也是一种选择。

HammerDB 基准

HammerDB 基准分成两个部分：HammerDB 测试套件和数据库 (本例中使用 MySQL)。我们定义了一个 TCL 脚本来自动执行命令行配置和相应工作负载的运行：

```
#!/usr/bin/tclsh
proc runtimer { seconds } {
set x 0
set timerstop 0
while {!$timerstop} {
```

```
 incr x
 after 1000
  if { ![ expr {$x % 60} ] } {
  set y [ expr $x / 60 ]
  puts "Timer: $y minutes elapsed"
  }
 update
 if {  [ vucomplete ] || $x eq $seconds } { set timerstop 1 }
   }
return
}
puts "SETTING CONFIGURATION"
dbset db mysql
dbset bm tpc-c
diset tpcc mysql_driver timed
diset tpcc mysql_rampup 0
diset tpcc mysql_duration 1
vuset logtotemp 1
loadscript
puts "SEQUENCE STARTED"
foreach z { 1 2 4 8 } {
puts "$z VU TEST"
vuset vu $z
vucreate
vurun
runtimer 120
vudestroy
after 5000
}
puts "TEST SEQUENCE COMPLETE"
```

CosBench 基准

CosBench 基准基于 Rook Ceph 的 Rados Gateway，至于在 Rook 下详细设置 Gateway 的方式参考：https://rook.io/docs/rook/v1.3/ceph-object.html

CosBench 支持 Web UI 的方式，直接将 xml 格式的负载配置文件上传即可自动加载运行。

性能数据和对比分析

场景一

<数据图表>

<结果分析>

场景二

<数据图表>

<结果分析>

总结和展望

复杂基准测试是常规微基准测试的有效补充，在几种特定场景下可以模拟出真实生产环境中的负载，有效全面的评估系统的性能。而作为工业标准的几种测试基准 (TPC-C、TPC-H 等)，可以作为重要的评估依据。

<继续展望>

附录

基准测试与工作负载

存储的工作负载

对于一个存储系统来说，评价性能好坏的有几个标准单位。

- IOPS：每秒钟的处理 IO 请求数。
- 吞吐量（MB/s）：单位时间内的读写数据量。
- 响应时间/等待时间：处理一个请求的时延，包括从发送请求到接到响应的时间。

而对于底层的存储介质来说，主要的操作就是读和写，还有一些文件系统元数据的操作。而读写根据地址空间的访问模式分成随机访问和顺序访问。随机访问的逻辑地址/物理地址是不可预测的，而顺序访问是根据当前的逻辑地址依次访问。

读写请求的块大小也对系统的性能有影响。一般来说，小块的请求更看重 IOPS 或响应时间，大块的请求以吞吐量为考量。

底层存储还有个重要因素就是队列深度，就是在端口队列中等待服务的 I/O 请求数量。适当增大队列深度，可以测出磁盘的峰值，但 IOPS 不会随着队列深度的增加而一直增加，达到一定峰值后会有所下降。对于 IO 请求的等待时间，随着队列深度的增加而增大。

根据请求块大小，顺序随机访问读写模式的不同，存储负载可以可以分成几个典型的类型：

- 数据库 (OLTP)：也叫联机事务处理，表示事务性非常高的系统，以小的事务以及小的查询为主。负载类型以小块的随机（随机约占 80%）写（读 25%，写 75%）为主，主要评估响应时间。
- 数据仓库 (OLAP)：也叫联机分析处理，支持复杂的分析操作，侧重决策分析，并且提供直观易懂的查询结果。负载类型以大块顺序（顺序约占 80%）读（读 75%，写 25%）为主，主要评估吞吐量。
- 视频数据采集：负载类型以大块顺序写为主。
- 虚拟桌面基础架构：负载类型以小块的随机读写（读 50%，写 50%）为主。
- 物联网：负载类型以混合大小块的顺序写为主。

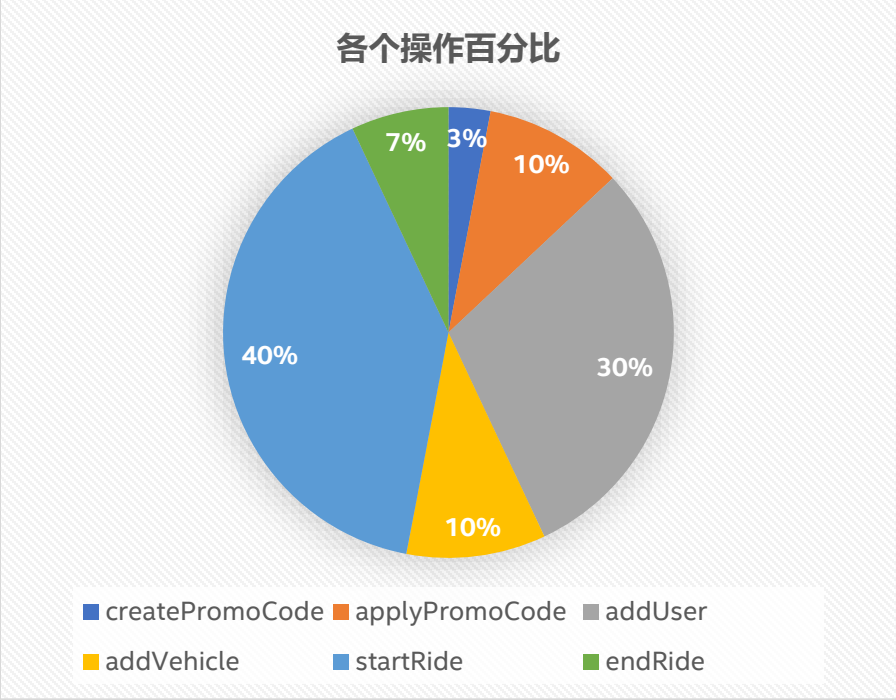
现实世界的负载要比这些典型的负载模型复杂的多，我们选取复杂基准测试是为了更真实的模拟生产环境中的负载，来指导我们对系统的评估和改造。

综合复杂负载 -- CockroachDB

CockroachDB 是一个可伸缩，跨区域复制，并支持事务，高可用性和高度一致性的分布式 SQL 数据库。它带有内置的负载生成器，用于模拟不同类型的客户端工作负载。下面列出几个常用的负载。

- Bank (银行)：使用货币平衡表为一组账户建模。
- KV (键值)：在整个集群中均匀随机地读取和写入键值对。

- MovR：模拟 MovR 示例应用程序的工作负载。MovR 是一家虚构的车辆共享公司，旨在展示 CockroachDB 功能。这个负载的数据集包含 6 个数据库表，模拟的操作按照比例如下图所示：



- TPCC：使用多个丰富的图表模拟联机事务处理工作负载。
- YCSB：使用其他自定义功能，模拟大量读，写或基于扫描的大规模键值工作负载。

CockroachDB 的负载还在不断的增加中，这些内置的负载操作简单、模式丰富，是快速验证的综合复杂基准测试的最佳选择之一。

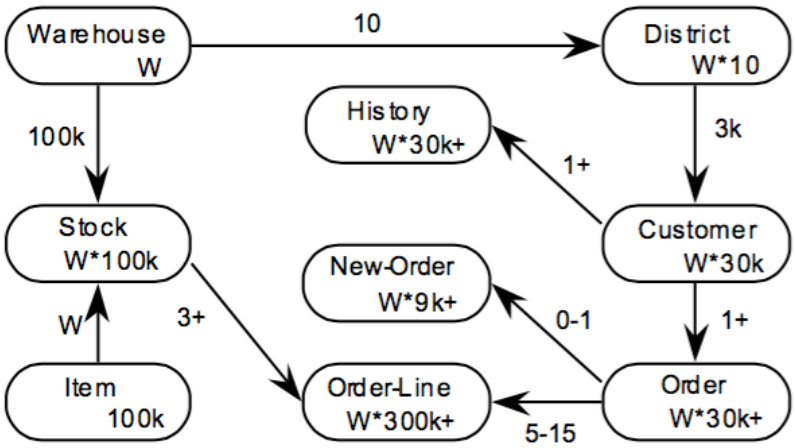
联机事务处理与联机分析处理 -- HammerDB

HammerDB 是基准测试和负载测试套件，支持当前流行的 Oracle 数据库、SQL Server、IBM DB2、MySQL、MariaDB、PostgreSQL 和 Redis 等。它支持基于 TPC-C 和 TPC-H 的工作负载，但是 HammerDB 支持的 TPC-C/TPC-H 和标准的 TPC-C/TPC-H 测试数据集略有不同，它更灵活轻量。

TPC 被称为事务处理性能委员会，负责定义诸如 TPC-C、TPC-H、TPC-W 等基准测试之类的事务处理与数据库性能基准测试，并依据这些基准测试想昂木发布客观的性能数据。

TPC-C 是衡量 OLTP（联机事务处理）的系统工业标准，是行业中公认的最权威和最复杂的在线事务处理基准测试。它通过模拟仓库和订单管理系统，测试广泛的数据库性能，包括查询、更新和队列是小批量业务。TPC-C 基准测试的评价是一种模拟订单录入每分钟商业事务 (tpmC)吞吐量。

TPC-C 的测试模型是一个大型批发销售公司，在地里分布的多个区域有业务，并且使用仓库管理。当业务扩展的时候，公司将添加新的仓库，每个仓库负责 10 个区域的供货，每个区域 3000 个客户服务，每个仓库维护 10 万种商品的库存记录。



它包含五种不同类型和复杂性的并发事务的混合体，这些事务可以在线执行或排队等待延迟执行。这五种事务包括：

- 新订单：客户输入新订单交易，占比 45%。
- 付款：更新账户余额以反映其付款状态，占比 43%。
- 交货：批量交易，占比 4%。
- 订单状态：查询客户最近交易的状态，占比 4%。
- 库存级别：查询仓库的库存状态，以便及时补货，占比 4%。

TPC-H 是决策支持基准，也叫 OLAP（联机分析处理）基准，它是在 TPC-D 上发展起来并取代了 TPC-D，它模拟决策支持系统中的数据库操作，测试数据库系统复杂查询的响应时间，以每小时执行的查询数(TPC-H QphH@Siz)作为度量指标。

在 TPC-H 模型中，定义了 8 张表，22 个复杂查询(SELECT)和 2 个更新(带有 INSERT 和 DELETE 的程序段)操作，被测试数据库的数据量从 1GB ~ 10000GB 有 8 个级别供用户选择。测试时，将 22 个查询随机组成查询流，2 个更新操作组成一个更新流，查询流和更新流并发执行数据路访问，查询流数目随数据量增加而增加。

数据库的应用一般有两种：以 TPC-C 为代表的联机事务处理和以 TPC-H 为代表的数据挖掘/联机分析处理。TPC-C 的结果对于数据库系统有一定的参考价值，银行、证券、税务报税系统、电子商务网站、电信业务都是比较典型的联机事务处理应用。TPC-H 的结果针对决策分析，也具有普遍的商业实用意义，目前在银行信贷分析和信用卡分析、电信运行分析、税收分析、烟草行业决策分析中都有广泛的应用。

参考资料

Rook 官网及 github：https://rook.io/ https://github.com/rook/rook

CockroachDB 工作负载：https://www.cockroachlabs.com/docs/stable/cockroach-workload.html

TPC 官网：http://www.tpc.org/

HammerDB 官网：https://hammerdb.com/

CosBench：https://github.com/intel-cloud/cosbench