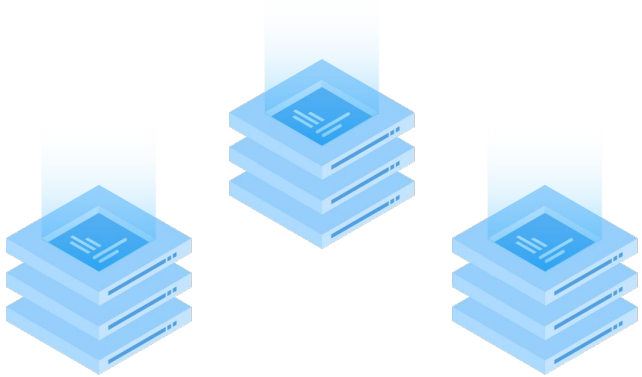
TiDB Benchmark

Presented by Liuwei

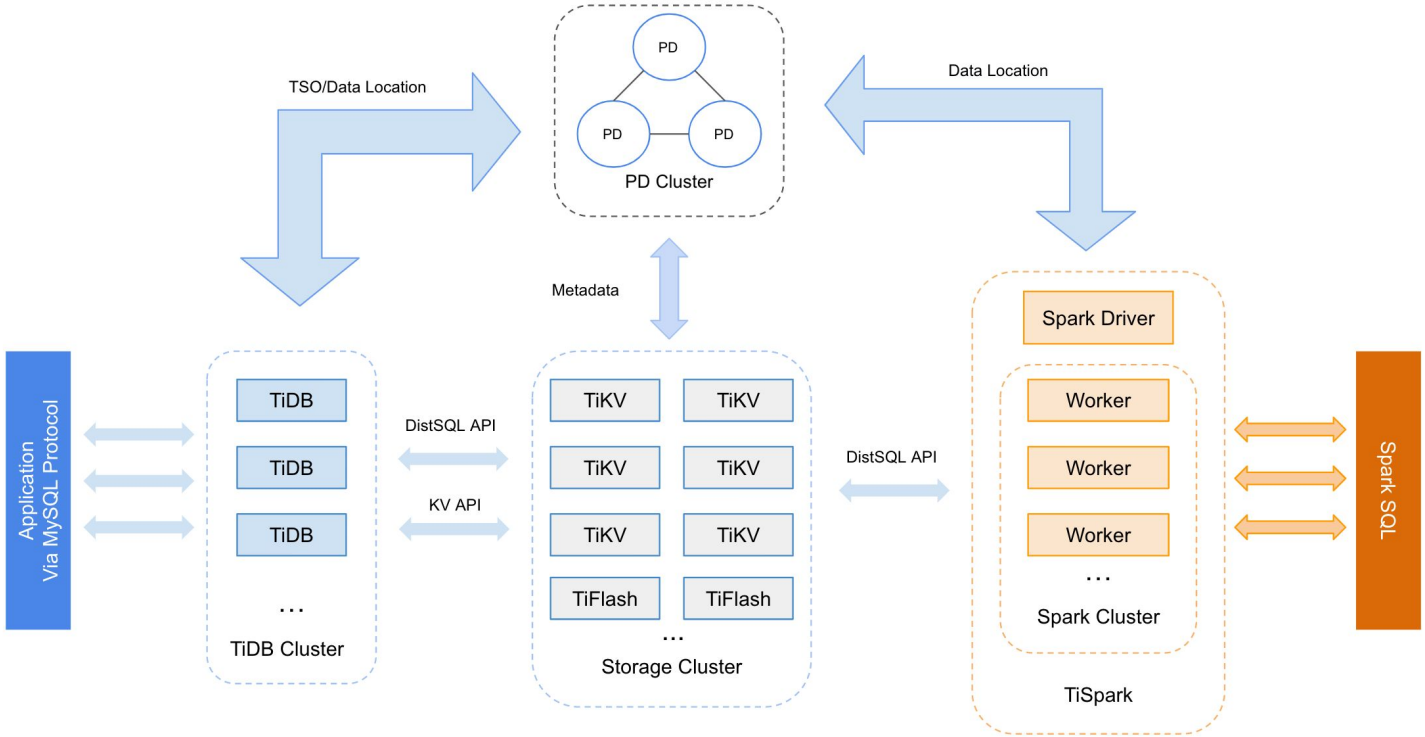
PingCAP TiDB



Before we begin

- Context
- Goal
- Outline
  - TiDB Architechture
  - Tiup && Dashboard
  - Sysbench
  - go-ycsb
  - go-tpc

Architecture



Deploy

- 测试 TiDB 集群的性能最好选取不低于三台物理机(或者虚拟机)节点部署 TiKV
- TiDB 与 TiKV 不要部署在同一节点上。
- 单节点部署 TiKV 时 CPU 最好在 16～24 之间，如果一台机器的 CPU 资源较多，可以考虑部署 2 个 TiKV 节点。TiKV 需要一部分内存用于磁盘的缓存，建议内存总大小不要低于数据总量的 10%。
- 单节点部署了多个 TiKV 实例时需要调整一些线程池配比。
  - 详细调整方式参见
    :https://github.com/pingcap-incubator/tidb-in-action/blob/master/session4/chapter8/threadpool-optimize.md

Tiup

https://tiup.io

```
curl --proto '=https' --tlsv1.2 -sSf https://tiup-mirrors.pingcap.com/install.sh | sh
```

```
source .bash_profile
```

详细使用方法见： https://docs.pingcap.com/zh/tidb/stable/production-deployment-using-tiup

Dashboard

- 使用 tiup 部署好集群后，可以打开 TiDB dashboard
- http://{{host}}:{{port}}/dashboard/#/overview
- 可通过 tiup cluster display {{cluster-name}} 来查看 dashboard 的地址
  - 如右图中标记了 Up|UI 的那一行PD
  - 节点所在的地址，即 为 Dashboard
  - 地址

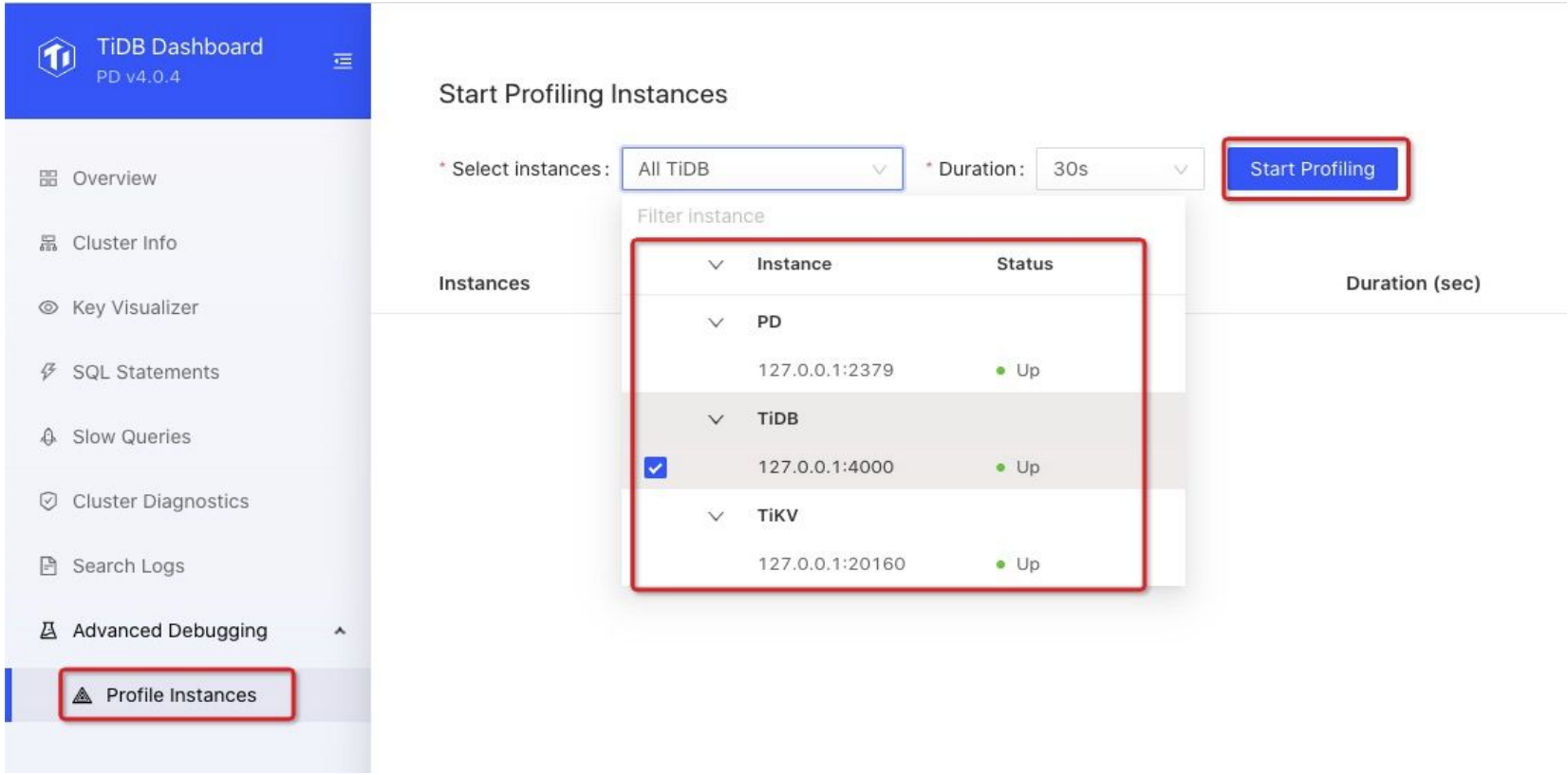
```
[pingcap@ip-172-16-5-40 go-tpc]$ tiup cluster display
Starting component `cluster`: /home/pingcap/.tiup/compone          v1.0.9/tiup-cluster
tidb Cluster:
tidb Version: nightly

ID                  Role          Host         Ports        OS/Arch       Status  Data Dir
--                  ----          ----         -----        -------       ------  --------
172.16.5.39:9098    alertmanager  172.16.5.39  9098/9099    linux/x86_64  Up      /nvme1n1
172.16.5.38:3005    grafana       172.16.5.38  3005         linux/x86_64  Up      -
172.16.5.37:2389    pd            172.16.5.37  2389/2390    linux/x86_64  Up|L    /nvme1n1
172.16.5.38:2389    pd            172.16.5.38  2389/2390    linux/x86_64  Up      /nvme1n1
172.16.5.39:2389    pd            172.16.5.39  2389/2390    linux/x86_64  Up|UI   /nvme1n1
172.16.5.39:9095    prometheus    172.16.5.39  9095         linux/x86_64  Up      /nvme1n1
172.16.5.38:4005    tidb          172.16.5.38  4005/10085   linux/x86_64  Up      -
172.16.5.39:4005    tidb          172.16.5.39  4005/10085   linux/x86_64  Up      -
172.16.5.37:20165   tikv          172.16.5.37  20165/20185  linux/x86_64  Up      /nvme1n1
172.16.5.38:20165   tikv          172.16.5.38  20165/20185  linux/x86_64  Up      /nvme1n1
172.16.5.39:20165   tikv          172.16.5.39  20165/20185  linux/x86_64  Up      /nvme1n1
[pingcap@ip-172-16-5-40 go-tpc]$
```

Dashboard

- 在运行测试的过程中，可以通过 Dashboard 来对集群的 SQL 执行延迟，以及热点分布进行初步的判断。

TiDB CPU Profile



Part I: Sysbench

- Sysbench 是一个采用 lua 编写的比较简单的测试数据库负载的程序
- 主要包括随机点查询（point select），简单的点更新（update where id=）和范围查询。
- 用户可以根据自己的业务场景，改写 sysbench 的 .lua 脚本来进行测试

Install

- 源码编译安装 https://github.com/akopytov/sysbench
- Ubuntu：
  - `curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.deb.sh | sudo bash`
  - `sudo apt -y install sysbench`
- CentOS
  - `curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.rpm.sh | sudo bash`
  - `sudo yum -y install sysbench`

Run

- 编辑配置文件 config
  - `mysql-host=172.16.30.33`
  - `mysql-port=4000`
  - `mysql-user=root`
  - `mysql-password=password`
  - `mysql-db=sbtest`
  - `time=600`
  - `threads=16`
  - `report-interval=10`
  - `db-driver=mysql`

Run

- 创建数据库
  - `> create database sbtest;`
- 导入数据之前先设置为乐观事务模式，导入结束后再设置回悲观模式
  - `> set global tidb_disable_txn_auto_retry = off;`
  - `> set global tidb_txn_mode="optimistic";`
- 导入数据
  - ```
    sysbench --config-file=config oltp_point_select --tables=32
    --table-size=10000000 prepare
    ```
    ```

Run

- 调整回悲观事务
  - `> set global tidb_txn_mode="pessimistic" ;`
- Point select 测试命令
  - `sysbench --config-file=config oltp_point_select --threads=128 --tables=32 --table-size=5000000 run`
- Update index 测试命令
  - `sysbench --config-file=config oltp_update_index --threads=128 --tables=32 --table-size=5000000 run`
- Read-only 测试命令
  - `sysbench --config-file=config oltp_read_only --threads=128 --tables=32 --table-size=5000000 run`

Part II: go-ycsb

- go-ycsb 是 PingCAP 在雅虎开源的 ycsb 基础上用 golang 开发的数据库性能测试工具
- 下载地址 : https://github.com/pingcap/go-ycsb
- 下载后执行 make 即可完成安装

Run

- load & run
  - ./bin/go-ycsb load mysql -P workloads/workloada -p recordcount=10000000 -p mysql.host={{host}} -p mysql.port={{port}} --threads 256

  - ./bin/go-ycsb run mysql -P workloads/workloada -p operationcount=10000000 -p mysql.host={{host}}-p mysql.port={{host}} --threads 256

- 负载类型
  - workloada ~ workloadf

Other

ycsb 可以通过调整 workloads 下的各个配置文件来

- 调整 requestdistribution 来更改随机分布的方式，默 认是zipfan
- 调整 fieldcount 来修改 column 的数量
- 更多配置见：https://github.com/brianfrankcooper/YCSB/wiki/Core-Properties

Part II: go-tpc

- TPC-C 是专门针对联机交易处理系统（OLTP 系统）的规范，一般情况下我们也把这类系统称为业务处理系统。1992年7月发布。
- TPC-H 是国际上认定的分析性数据 库测试的标准
- PingCAP 使用 golang 编写了 TPC-C 和 TPC-H 的测试程序，下载链接和使用说明见：https://github.com/pingcap/go-tpc

TPC-C

- 安装方式
  - `git clone https://github.com/pingcap/go-tpc.git`
  - `make build`
- 准备数据
  - `./bin/go-tpc tpcc -H {{host}} -P {{port}} -D tpcc --warehouses 1000 prepare`
- 运行测试
  - `./bin/go-tpc tpcc -H {{host}} -P {{port}} -D tpcc --warehouses 1000 run`

TPC-H

- 导入数据
  - `./bin/go-tpc tpch prepare -H {{host}} -P {{port}} -D tpch --sf 50 --analyze`
- 运行测试
  - `./bin/go-tpc tpch run -H {{host}} -P {{port}} -D tpch --sf 50`

作业

课程作业

分值：300

题目描述：

使用 sysbench、go-ycsb 和 go-tpc 分别对 TiDB 进行测试并且产出测试报告。

测试报告需要包括以下内容

- 部署环境的机器配置（CPU、内存、磁盘规格型号），拓扑结构（TiDB、TiKV 各部署于哪些节点）
- 调整过后的 TiDB 和 TiKV 配置
- 测试输出结果
- 关键指标的监控截图
    - TiDB Query Summary 中的 qps 与 duration
    - TiKV Details 面板中 Cluster 中各 server 的 CPU 以及 QPS 指标
    - TiKV Details 面板中 grpc 的 qps 以及 duration

输出：写出你对该配置与拓扑环境和 workload 下 TiDB 集群负载的分析，提出你认为的 TiDB 的性能的瓶颈所在（能提出大致在哪个模块即可）

截止时间：下周二（8.25）24:00:00（逾期提交不给分）

作业提交方式

- 作业提交方式：提交至个人 GitHub，发送邮件至 talent-plan@tidb.io
- 邮件主题建议：HP - Lesson N-【GitHub ID】
  - N 代表课程章节周数，例如第一 节课程的作业邮件主题为 HP- Lesson 01-【GitHub ID】）
- 邮件正文建议：
  - 作业提交文档内容包含两部分：
    - Github 链接
    - 问题 / 意见反馈
- 友情提示：请不要直接在邮件中添加压缩包（GitHub 链接为唯一格式）

课程答疑与学习反馈

扫描左侧二维码填写报名信息，加入课程学习交流群，课程讲师在线答疑，学习效果 up up！

Thank you!