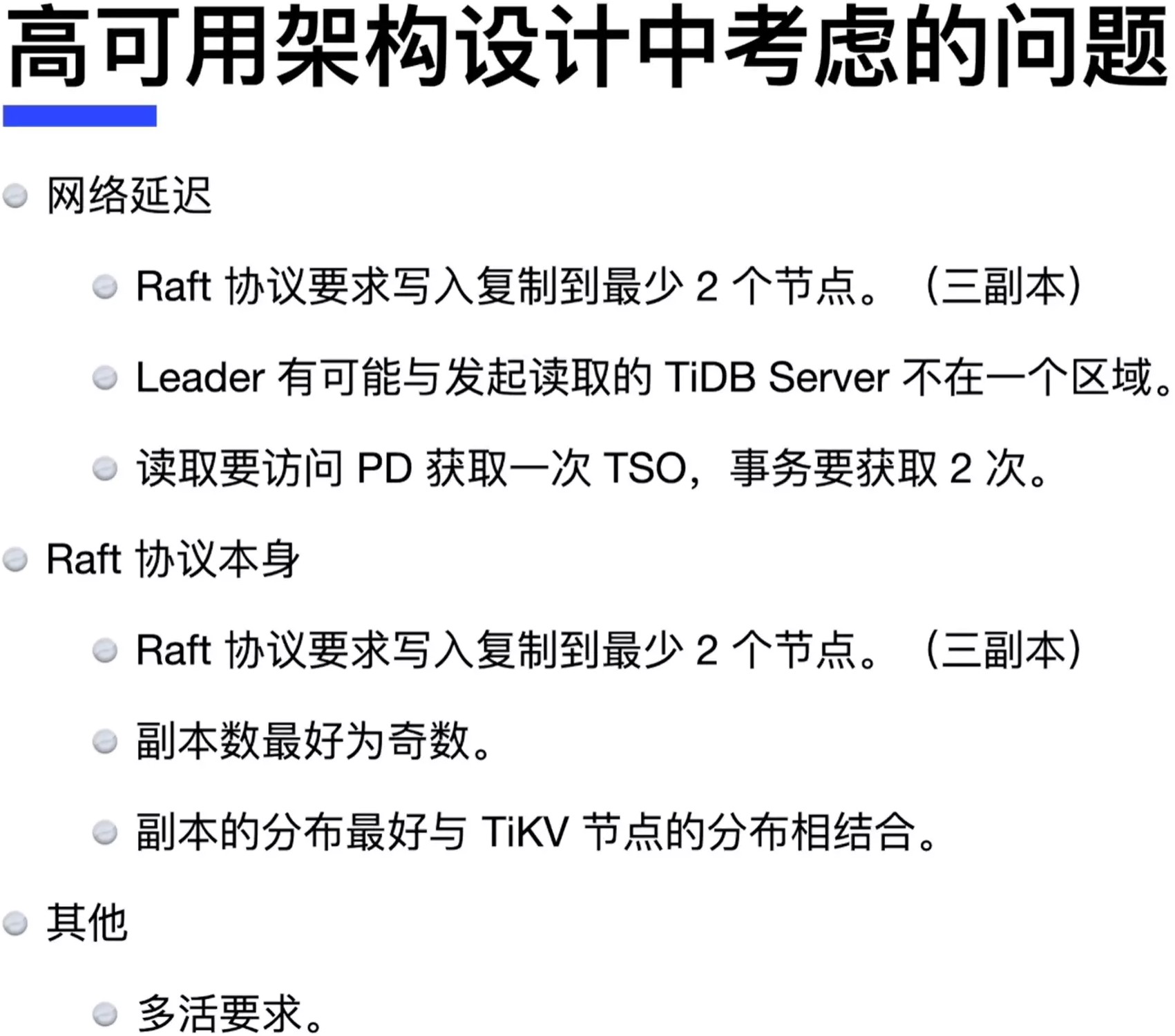


02、TiDB 数据库常⽤⾼可⽤架构

--2022-12-03 春雷





1、⾼可⽤架构设计中考虑的问题



2、架构概述



2.1、架构概述

 同城三中⼼架构：

 同区域有三中⼼部署 TiDB 集群

 同城两中⼼架构：

 同区域双中⼼部署TiDB集群

 两地三中⼼：

 双区域三中⼼部署TiDB集群



2.2、基本概念



2.2.1、Placement Rules 简介

是 PD 在 4.0 版本引⼊的⼀套副本规则系统，⽤于指导 PD 针对不同类型的数据⽣成对应的调度。通过组合不同的调度规则，⽤户可以精细地控制任何⼀段连续数据的副本数量、存放位

置、主机类型、是否参与 Raft 投票、是否可以担任 Raft leader 等属性。 Placement Rules 特性在 TiDB v5.0 及以上的版本中默认开启。

注意：

TiDB 5.2 及以上版本默认不⽀持label-property配置。若要设置副本策略，请使⽤Placement Rules。



2.2.2、TiKV Labels 简介

【简介】：

TiKV 是⼀个 Multi-Raft 系统，其数据按 Region（默认 96M）切分，每个 Region 的 3 个副本构成了⼀个 Raft Group。假设⼀个 3 副本 TiDB 集群，由于 Region 的副本数与 TiKV 实例数量⽆关，则⼀个 Region 的 3 个副本只会被调度到其中 3 个 TiKV 实例上，也就是说即使集群扩容 N 个 TiKV 实例，其本质仍是⼀个 3 副本集群。

由于 3 副本的 Raft Group 只能容忍 1 副本故障，当集群被扩容到 N 个 TiKV 实例时，这个集群依然只能容忍⼀个 TiKV 实例的故障。2 个 TiKV 实例的故障可能会导致某些 Region 丢失多个副本，整个集群的数据也不再完整，访问到这些 Region 上的数据的 SQL 请求将会失败。⽽ N 个 TiKV 实例中同时有两个发⽣故障的概率是远远⾼于 3 个 TiKV 中同时有两个发⽣故障的概率的，也就是说 Multi-Raft 系统集群扩容 TiKV 实例越多，其可⽤性是逐渐降低的。

正因为 Multi-Raft TiKV 系统局限性，Labels 标签应运⽽出，其主要⽤于描述 TiKV 的位置信息。Label 信息随着部署或滚动更新操作刷新到 TiKV 的启动配置⽂件中，启动后的 TiKV 会将

⾃⼰最新的 Label 信息上报给 PD，PD 根据⽤户登记的 Label 名称（也就是 Label 元信息），结合 TiKV 的拓扑进⾏ Region 副本的最优调度，从⽽提⾼系统可⽤性。



【TiKV Labels 样例规划】

针对 TiKV Labels 标签，你需要根据已有的物理资源、容灾能⼒容忍度等⽅⾯进⾏设计与规

划，进⽽提升系统的可⽤性和容灾能⼒。并根据已规划的拓扑架构，在集群初始化配置⽂件中进

⾏配置（此处省略其他⾮重点项）：

1

server\_configs: pd:

replication.location-labels: ["zone","az","rack","host"]

注：地域，机房，机架，服务器

tikv\_servers:

- host: 10.63.10.30

config:

server.labels: { zone: "z1", az: "az1", rack: "r1", host: "30" }

- host: 10.63.10.31

config:

server.labels: { zone: "z1", az: "az1", rack: "r1", host: "31" }

- host: 10.63.10.32

config:

server.labels: { zone: "z1", az: "az1", rack: "r2", host: "32" }

- host: 10.63.10.33

config:

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 18 | server.labels: { | zone: | "z1", | az: | "az1", | rack: | "r2", | host: | "33" | } |
| 19 |  |  |  |  |  |  |  |  |  |  |
| 20 | - host: 10.63.10.34 |  |  |  |  |  |  |  |  |  |
| 21 | config: |  |  |  |  |  |  |  |  |  |
| 22 | server.labels: { | zone: | "z2", | az: | "az2", | rack: | "r1", | host: | "34" | } |
| 23 | - host: 10.63.10.35 |  |  |  |  |  |  |  |  |  |
| 24 | config: |  |  |  |  |  |  |  |  |  |
| 25 | server.labels: { | zone: | "z2", | az: | "az2", | rack: | "r1", | host: | "35" | } |
| 26 | - host: 10.63.10.36 |  |  |  |  |  |  |  |  |  |
| 27 | config: |  |  |  |  |  |  |  |  |  |
| 28 | server.labels: { | zone: | "z2", | az: | "az2", | rack: | "r2", | host: | "36" | } |
| 29 | - host: 10.63.10.37 |  |  |  |  |  |  |  |  |  |
| 30 | config: |  |  |  |  |  |  |  |  |  |
| 31 | server.labels: { | zone: | "z2", | az: | "az2", | rack: | "r2", | host: | "37" | } |
| 32 |  |  |  |  |  |  |  |  |  |  |
| 33 | - host: 10.63.10.38 |  |  |  |  |  |  |  |  |  |
| 34 | config: |  |  |  |  |  |  |  |  |  |
| 35 | server.labels: { | zone: | "z3", | az: | "az3", | rack: | "r1", | host: | "38" | } |
| 36 | - host: 10.63.10.39 |  |  |  |  |  |  |  |  |  |
| 37 | config: |  |  |  |  |  |  |  |  |  |
| 38 | server.labels: { | zone: | "z3", | az: | "az3", | rack: | "r1", | host: | "39" | } |
| 39 | - host: 10.63.10.40 |  |  |  |  |  |  |  |  |  |
| 40 | config: |  |  |  |  |  |  |  |  |  |
| 41 | server.labels: { | zone: | "z3", | az: | "az3", | rack: | "r2", | host: | "40" | } |
| 42 | - host: 10.63.10.41 |  |  |  |  |  |  |  |  |  |
| 43 | config: |  |  |  |  |  |  |  |  |  |
| 44 | server.labels: { | zone: | "z3", | az: | "az3", | rack: | "r2", | host: | "41" | } |
| 45 |  |  |  |  |  |  |  |  |  |  |

本例中，zone 表示逻辑可⽤区层级，⽤于控制副本的隔离（当前集群 3 副本）。

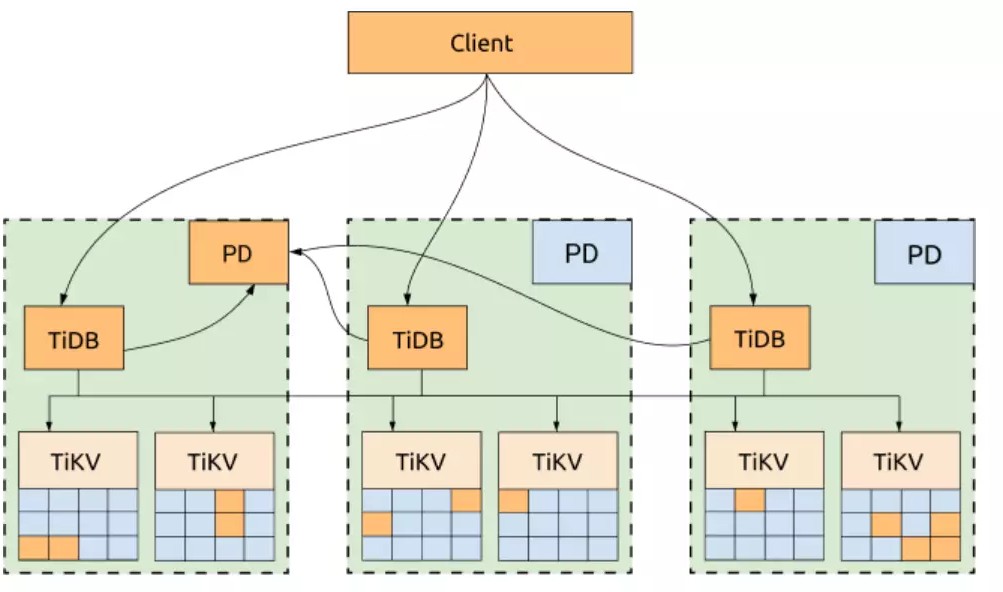


3、同城三中⼼架构

同区域三中⼼⽅案，即同区域有三个中⼼部署 TiDB 集群，中⼼间的数据在集群内部（通过 Raft 协议）进⾏同步。同区域三中⼼可同时对外进⾏读写服务，任意中⼼发⽣故障不影响数据⼀致性。

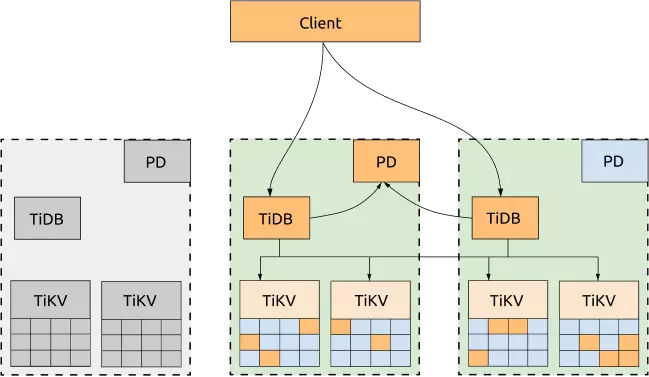
【架构图】：

集群 TiDB、TiKV 和 PD 组件分别部署在 3 个不同的中⼼，这是最常规且⾼可⽤性最⾼的⽅案



【优点】：

所有数据的副本分布在三个 AZ，具备⾼可⽤和容灾能⼒任何⼀个 AZ 失效后，不会产⽣任何数据丢失 (RPO = 0)

任何⼀个 AZ 失效后，其他两个 AZ 会⾃动发起 leader election，并在⼀定时间内（通常 20s 以内）⾃动恢复服务

【缺点】：

性能受⽹络延迟影响。具体影响如下：

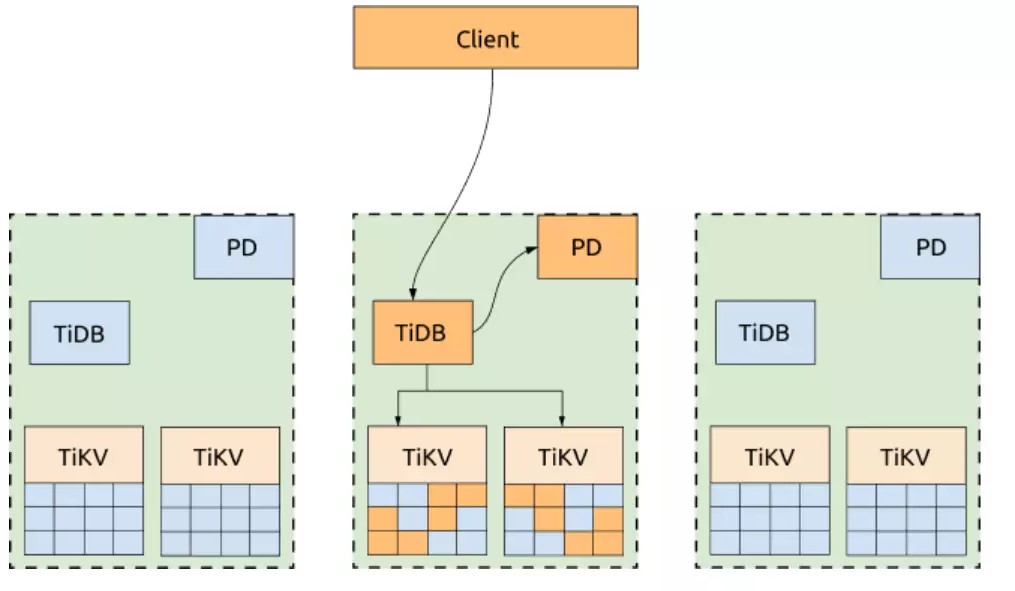
对于写⼊的场景，所有写⼊的数据需要同步复制到⾄少两个 AZ，由于 TiDB 写⼊过程使⽤两阶段提交，故写⼊延迟⾄少需要两倍 AZ 间的延迟。

对于读请求来说，如果数据 leader 与发起读取的 TiDB 节点不在同⼀个 AZ，也会受⽹络延迟影响。

TiDB 中的每个事务都需要向 PD leader 获取 TSO，当 TiDB 与 PD leader 不在同⼀个 AZ时，TiDB 上运⾏的事务也会因此受⽹络延迟影响，每个有写⼊的事务会获取两次 TSO。



【架构优化图】：

如果不需要每个 AZ 同时对外提供服务，可以将业务流量全部派发到⼀个 AZ，并通过调度策略把 Region leader 和 PD leader 都迁移到同⼀个 AZ。这样，不管是从 PD 获取 TSO，还是读取 Region，都不会受 AZ 间⽹络的影响。当该 AZ 失效时，PD leader 和 Region leader 会⾃动在其它 AZ 选出，只需要把业务流量转移⾄其他存活的 AZ 即可。

【优点】：

集群 TSO 获取能⼒以及读取性能有所提升。具体调度策略设置模板参照如下：

1

-- 其他 AZ 将 leader 驱逐至承载业务流量的 AZ

config set label-property reject-leader LabelName labelValue

-- 迁移 PD leader 并设置优先级 member leader transfer pdName1 member leader\_priority pdName1 5 member leader\_priority pdName2 4 member leader\_priority pdName3 3

2

3

4

5

6

7

8

9

【缺点】：

写⼊场景仍受 AZ ⽹络延迟影响，这是因为遵循 Raft 多数派协议，所有写⼊的数据需要同步复制到⾄少两个 AZ

TiDB Server 是 AZ 级别单点

业务流量纯⾛单 AZ，性能受限于单 AZ ⽹络带宽压⼒

TSO 获取能⼒以及读取性能受限于业务流量 AZ 集群 PD、TiKV 组件是否正常，否则仍受跨 AZ ⽹络交互影响



【⾼可⽤和容灾分析】：

采⽤区域多 AZ ⽅案，当任意⼀个 AZ 故障时，集群能⾃动恢复服务，不需要⼈⼯介⼊，并能保证数据⼀致性。注意，各种调度策略主要⽤于优化性能，当发⽣故障时，调度机制总是优先考虑可⽤性⽽不是性能。

RTO <30S RPO =0

4、同城双中⼼

单区域双 AZ 部署⽅案下，两个 AZ 通常位于同⼀个城市或两个相邻城市（例如北京和廊坊）， 相距 50 km 以内，AZ 间的⽹络连接延迟⼩于 1.5 ms，带宽⼤于 10 Gbps。

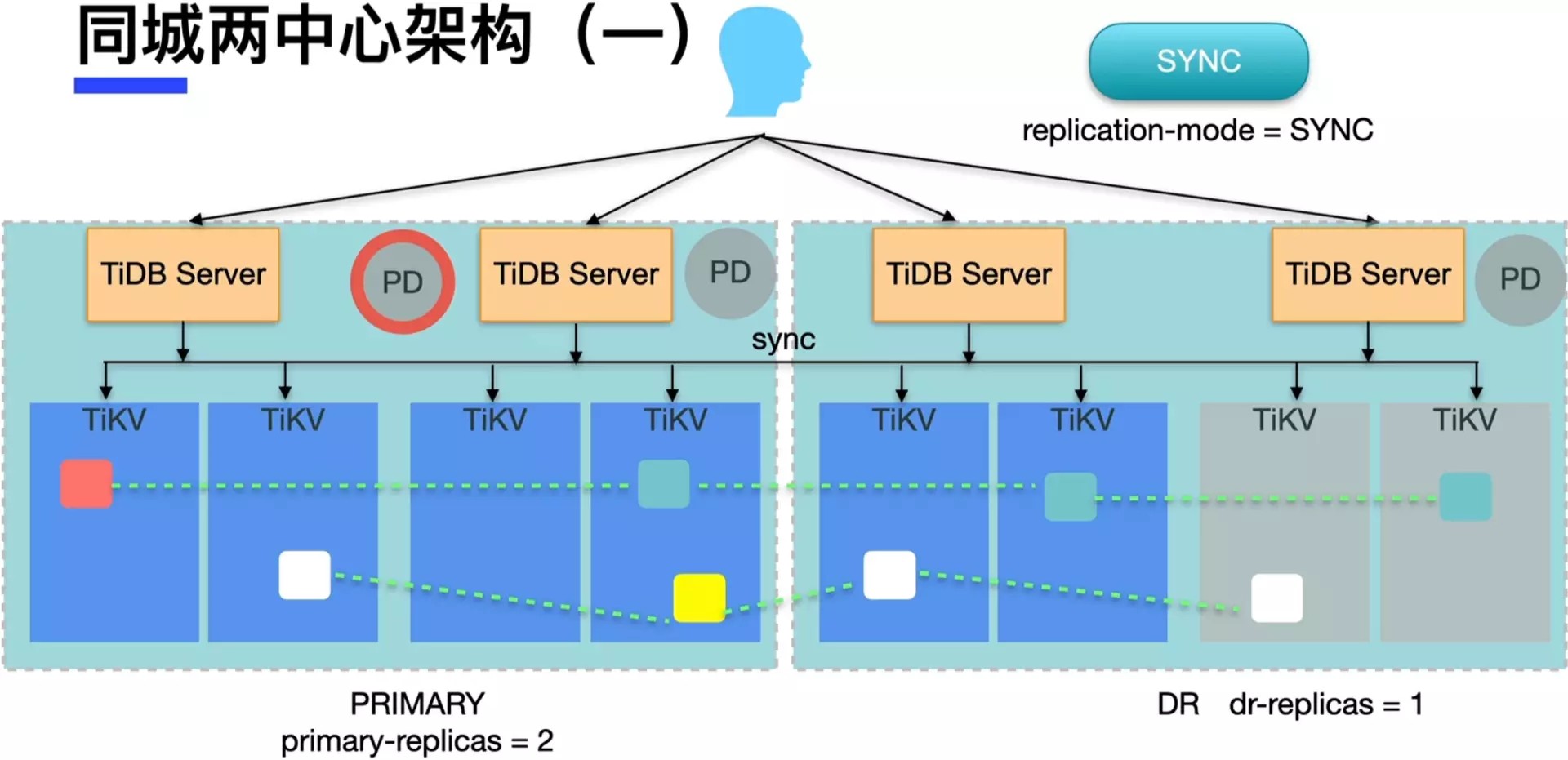
【架构图】：

副本数：4

主机房：2个Voter

从机房：1个Voter，1个Learner配置：

primary-replicas=2 dr-replicas=1





【Placement Rules的region副本的⻆⾊】：

Voter：参加选举，⼜能够被选择为leader Follower：参加选举，不会被选择为leader

Learner：只接受⽇志复制，不参加选举，不会被选择为leader



【Placement Rules 规划】：

为了按照规划的集群拓扑进⾏部署，你需要使⽤ Placement Rules来规划集群副本的放置位

置。以 6 副本（3 个 Voter 副本在主 AZ，2 个 Follower 副本和 1 个 Learner 副本在从 AZ）的部署⽅式为例，可使⽤ Placement Rules 进⾏如下副本配置：

1

cat rule.json [

{

"group\_id": "pd", "group\_index": 0, "group\_override": false, "rules": [

{

"group\_id": "pd",

"id": "az-east",

2

3

4

5

6

7

8

9

10

1. "start\_key": "",
2. "end\_key": "",
3. "role": "voter",
4. "count": 3,
5. "label\_constraints": [

16 {

17 "key": "az",

18 "op": "in",

1. "values": [
2. "east"

21 ]

22 }

23 ],

24 "location\_labels": [

25 "az",

1. "rack",
2. "host"

28 ]

29 },

30 {

1. "group\_id": "pd",
2. "id": "az-west",
3. "start\_key": "",
4. "end\_key": "",
5. "role": "follower",
6. "count": 2,
7. "label\_constraints": [

38 {

39 "key": "az",

40 "op": "in",

1. "values": [
2. "west"

43 ]

44 }

45 ],

46 "location\_labels": [

47 "az",

1. "rack",
2. "host"

50 ]

51 },

52 {

1. "group\_id": "pd",
2. "id": "az-west",
3. "start\_key": "",
4. "end\_key": "",
5. "role": "learner",
6. "count": 1,
7. "label\_constraints": [

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 60 | { | | | | |  | |
| 61 |  | | | | | "key": "az", | |
| 62 |  | | | | | "op": "in", | |
| 63 |  | | | | | "values": [ | |
| 64 |  | | | | | "west" | |
| 65 |  | | | | | ] | |
| 66 | } | | | | |  | |
| 67 |  |  |  |  | ], | |  |
| 68 |  |  |  |  | "location\_labels": | | [ |
| 69 |  |  |  |  | "az", | |  |
| 70 |  |  |  |  | "rack", | |  |
| 71 |  |  |  |  | "host" | |  |
| 72 |  |  |  |  | ] | |  |
| 73 |  |  |  | } |  | |  |
| 74 |  |  | ] |  |  | |  |
| 75 |  | } |  |  |  | |  |
| 76 | ] |  |  |  |  | |  |

【三种状态】：

该部署⽅案定义了三种状态来控制和标识集群的同步状态，该状态约束了 TiKV 的同步⽅式。集群的复制模式可以⾃动在三种状态之间⾃适应切换。

sync：同步复制模式，此时从 AZ (DR) ⾄少有⼀个副本与主 AZ (PRIMARY) 进⾏同步， Raft 算法保证每条⽇志按 Label 同步复制到 DR。

async：异步复制模式，此时不保证 DR 与 PRIMARY 完全同步，Raft 算法使⽤经典的 majority ⽅式复制⽇志。

sync-recover：恢复同步，此时不保证 DR 与 PRIMARY 完全同步，Raft 逐步切换成 Label 复制，切换成功后汇报给 PD。

【启⽤⾃适应同步模式】：

副本的复制模式由 PD 节点控制。如果要使⽤ DR Auto-sync ⾃适应同步模式，需要按照以下任⼀⽅法修改 PD 的配置。

⽅法⼀：先配置 PD 的配置⽂件，然后部署集群。

1

[replication-mode]

replication-mode = "dr-auto-sync" [replication-mode.dr-auto-sync] label-key = "az"

primary = "east" dr = "west"

primary-replicas = 3

dr-replicas = 2

wait-store-timeout = "1m" wait-sync-timeout = "1m"

2

3

4

5

6

7

8

9

10

11

⽅法⼆：如果已经部署了集群，则使⽤ pd-ctl 命令修改 PD 的配置。

config set replication-mode dr-auto-sync

1

2

config set replication-mode dr-auto-sync label-key az config set replication-mode dr-auto-sync primary east config set replication-mode dr-auto-sync dr west

config set replication-mode dr-auto-sync primary-replicas 3 config set replication-mode dr-auto-sync dr-replicas 2

3

4

5

6

7



【配置项说明】：

replication-mode 为待启⽤的复制模式，以上示例中设置为 dr-auto-sync。默认使⽤ majority 算法。

label-key ⽤于区分不同的 AZ，需要和 Placement Rules 相匹配。其中主 AZ 为 "east"，从 AZ 为 "west"。

primary-replicas 是主 AZ 上 Voter 副本的数量。 dr-replicas 是从 AZ 上 Voter 副本的数量。

wait-store-timeout 是当出现⽹络隔离或者故障时，切换到异步复制模式的等待时间。如果超过这个时间还没恢复，则⾃动切换到异步复制模式。默认时间为 60 秒。



【状态转换】：

简单来讲，集群的复制模式可以⾃动在三种状态之间⾃适应的切换：

当集群⼀切正常时，会进⼊同步复制模式来最⼤化地保障灾备 AZ 的数据完整性。

当 AZ ⽹络断连或灾备 AZ 发⽣整体故障时，在经过⼀段提前设置好的保护窗⼝之后，集群会进⼊异步复制状态，来保障业务的可⽤性。

当 AZ ⽹络重连或灾备 AZ 整体恢复之后，灾备 AZ 的 TiKV 节点会重新加⼊到集群，逐步



同步数据并最终转为同步复制模式。



【状态转换的细节过程如下】：

1. 初始化：集群在初次启动时处于 sync（同步复制）模式，PD 会下发信息给 TiKV，所有 TiKV 节点会严格按照 sync 模式的要求进⾏⼯作。
2. 同步切异步：PD 通过定时检查 TiKV 的⼼跳信息来判断 TiKV 是否宕机或断连。如果宕机数超过 Primary 和 DR 各⾃副本的数量 primary-replicas 和 dr-replicas，意味着⽆法完成同步复制，需要切换状态。当宕机时间超过了wait-store-timeout 设定的时间，PD 将集群状态切换成 async（异步复制）模式。然后 PD 再将 async 状态下发到所有 TiKV 节点，

TiKV 的复制模式由双 AZ 同步⽅式转为原⽣的 (majority)。



Raft ⼤多数落实⽅式

1. 异步切同步：PD 通过定时检查 TiKV 的⼼跳信息来判断 TiKV 是否恢复连接，如果宕机数

⼩于 Primary 和 DR 各⾃副本的数量，意味着可以切回同步。PD 会将集群复制状态先切换

⾄ sync-recover，再将该状态下发给所有 TiKV 节点。TiKV 的所有 Region 逐步切换成双 AZ 同步复制模式，切换成功后通过⼼跳将状态同步信息给 PD。PD 记录 TiKV 上 Region的状态并统计恢复进度。当 TiKV 的所有 Region 都完成了同步复制模式的切换，PD 将集群复制状态切换为 sync。



【灾难恢复】：

单区域双 AZ 部署提供的容灾恢复⽅案。

当处于同步复制状态的集群发⽣了灾难，可进⾏ RPO = 0 的数据恢复：

如果主 AZ 发⽣故障，丢失了⼤多数 Voter 副本，但是从 AZ 有完整的数据，可在从 AZ 恢复数据。此时需要⼈⼯介⼊，通过专业⼯具恢复（恢复⽅式请联系 TiDB 团队）。

如果从 AZ 发⽣故障，丢失了少数 Voter 副本，能⾃动切换成 async 异步复制模式。当不处于同步复制状态的集群发⽣了灾难，不能保证满⾜ RPO = 0 进⾏数据恢复：

如果丢失了⼤多数 Voter 副本，需要⼈⼯介⼊，通过专业⼯具恢复（恢复⽅式请联系 TiDB团队）。

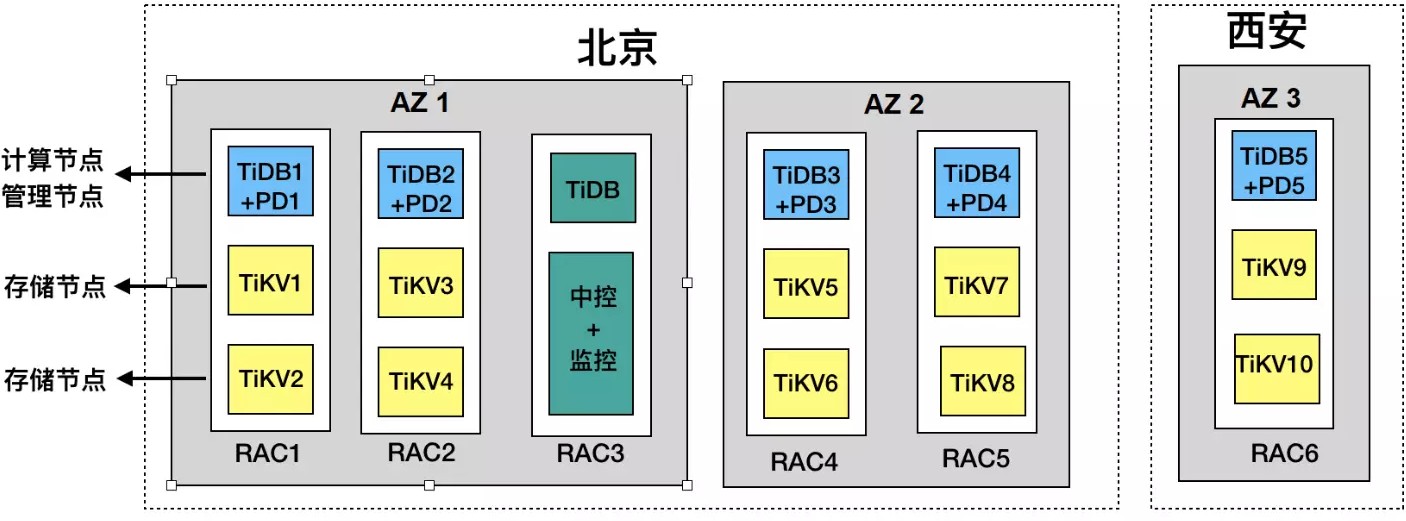
4、两地三中⼼

假设北京有两个 AZ，AZ1 和 AZ2，⻄安有⼀个 AZ，AZ3。北京同区域两 AZ 之间⽹络延迟低于 3 ms，北京与⻄安之间的⽹络使⽤ ISP 专线，延迟约为 20 ms。

集群采⽤双区域三 AZ 部署⽅式，分别为北京 AZ1，北京 AZ2，⻄安 AZ3。

集群采⽤ 5 副本模式，其中 AZ1 和 AZ2 分别放 2 份副本，AZ3 放 1 份副本；TiKV 按机柜设置 Label，即每个机柜上有 1 份副本。

副本间通过 Raft 协议保证数据的⼀致性和⾼可⽤，对⽤户完全透明。



该架构具备⾼可⽤能⼒，同时通过 PD 调度保证 Region Leader 只出现在同区域的两个 AZ。相⽐于三 AZ，即 Region Leader 分布不受限制的⽅案，双区域三 AZ ⽅案有以下优缺点：



优点

Region Leader 都在同区域 AZ，延迟低，数据写⼊速度更优。双 AZ 可同时对外提供服务，资源利⽤率更⾼。

任⼀ AZ 失效后，另⼀ AZ 接管服务，业务可⽤并且不发⽣数据丢失。



缺点

因为数据⼀致性是基于 Raft 算法实现，当同区域两个 AZ 同时失效时，因为远程灾备 AZ 只剩下⼀份副本，不满⾜ Raft 算法⼤多数副本存活的要求。最终将导致集群暂时不可⽤，需要从⼀副本恢复集群，丢失少部分还没同步的热数据。这种情况出现概率较

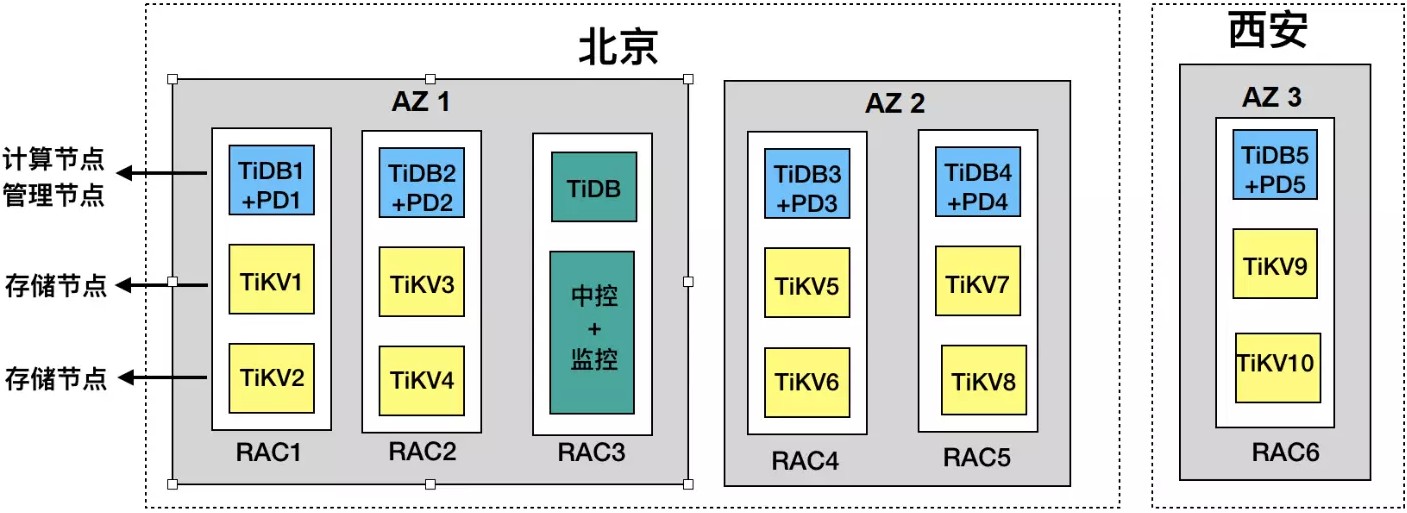
⼩。

由于使⽤了⽹络专线，该架构下⽹络设施成本较⾼。

双区域三 AZ 需设置 5 副本，数据冗余度增加，空间成本攀升。

【详细示例】：

北京、⻄安双区域三 AZ 配置详解：



如上图所示，北京有两个可⽤区 AZ1 和 AZ2，可⽤区 AZ1 有三套机架 rac1、rac2 和 rac3，可⽤区 AZ2 有两套机架 rac4 和 rac5；⻄安可⽤区 AZ3 有⼀套机架 rac6。

AZ1 的 rac1 机架中，⼀台服务器部署了 TiDB 和 PD 服务，另外两台服务器部署了 TiKV 服务，其中，每台 TiKV 服务器部署了两个 TiKV 实例 (tikv-server)，rac2、rac4、rac5 和 rac6 类似。

机架 rac3 上部署了 TiDB Server、中控及监控服务器。TiDB Server ⽤于⽇常管理维护和备份。中控和监控服务器上部署了 Prometheus、Grafana 以及恢复⼯具。

另可增加备份服务器，其上部署 Drainer，Drainer 以输出 ﬁle ⽂件的⽅式将 binlog 数据保存到指定位置，实现增量备份的⽬的。

【配置示例】：

以下为⼀个tiup topology.yaml ⽂件示例：

1

# # Global variables are applied to all deployments and used as the defaul t value of

# # the deployments if a specific deployment value is missing. global:

user: "tidb" ssh\_port: 22

deploy\_dir: "/data/tidb\_cluster/tidb-deploy"

data\_dir: "/data/tidb\_cluster/tidb-data"

server\_configs: tikv:

server.grpc-compression-type: gzip pd:

replication.location-labels: ["dc","replication zone","rack","host"]

pd\_servers:

- host: 10.63.10.10

name: "pd-10"

- host: 10.63.10.11

name: "pd-11"

- host: 10.63.10.12

name: "pd-12"

- host: 10.63.10.13

name: "pd-13"

- host: 10.63.10.14

name: "pd-14"

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

|  |  |  |
| --- | --- | --- |
| 27 | tidb\_servers: | |
| 28 | - host: | 10.63.10.10 |
| 29 | - host: | 10.63.10.11 |
| 30 | - host: | 10.63.10.12 |
| 31 | - host: | 10.63.10.13 |
| 32 | - host: | 10.63.10.14 |
| 33 |  |  |

34 tikv\_servers:

35 - host: 10.63.10.30

1. config:
2. server.labels: { az: "1", replication zone: "1", rack: "1", host: "3

0" }

38 - host: 10.63.10.31

1. config:
2. server.labels: { az: "1", replication zone: "2", rack: "2", host: "3

1" }

41 - host: 10.63.10.32

1. config:
2. server.labels: { az: "2", replication zone: "3", rack: "3", host: "3

2" }

44 - host: 10.63.10.33

1. config:
2. server.labels: { az: "2", replication zone: "4", rack: "4", host: "3

3" }

47 - host: 10.63.10.34

48 config:

49 server.labels: { az: "3", replication zone: "5", rack: "5", host: "3

4" }

50 raftstore.raft-min-election-timeout-ticks: 1000

51 raftstore.raft-max-election-timeout-ticks: 1200

52

53 monitoring\_servers:

54 - host: 10.63.10.60

55

56 grafana\_servers:

57 - host: 10.63.10.60

58

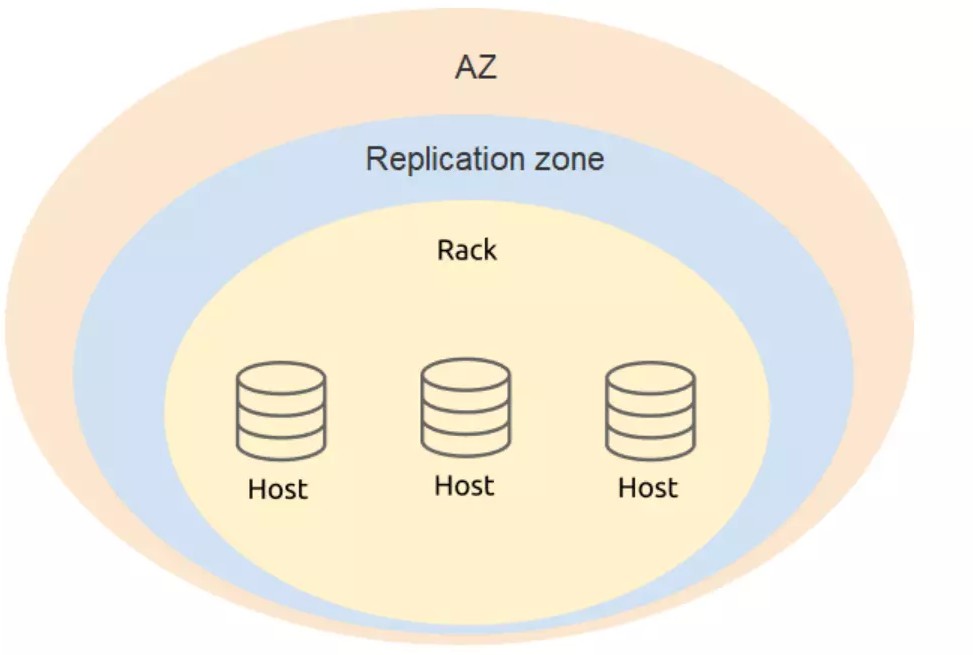
59 alertmanager\_servers:

60 - host: 10.63.10.60



【Labels 设计】：

在双区域三 AZ 部署⽅式下，对于 Labels 的设计需要充分考虑到系统的可⽤性和容灾能⼒，建议根据部署的物理结构来定义 AZ、replication zone、rack 和 host 四个等级。



PD 设置中添加 TiKV label 的等级配置。

1

server\_configs: pd:

replication.location-labels: ["az","replication zone","rack","host"]

2

3

4

tikv\_servers 设置基于 TiKV 真实物理部署位置的 Label 信息，⽅便 PD 进⾏全局管理和调度。

1

tikv\_servers:

- host: 10.63.10.30

config:

server.labels: { az: "1", replication zone: "1", rack: "1", host: "3

0" }

- host: 10.63.10.31

config:

server.labels: { az: "1", replication zone: "2", rack: "2", host: "3

1" }

- host: 10.63.10.32

config:

server.labels: { az: "2", replication zone: "3", rack: "3", host: "3

2" }

- host: 10.63.10.33

config:

server.labels: { az: "2", replication zone: "4", rack: "4", host: "3

3" }

- host: 10.63.10.34

config:

server.labels: { az: "3", replication zone: "5", rack: "5", host: "3

4" }

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17



【参数配置优化】：

在双区域三 AZ 的架构部署中，从性能优化的⻆度，除了常规参数配置外，还需要对集群中相关组件参数进⾏调整。

启⽤ TiKV gRPC 消息压缩。由于需要在⽹络中传输集群数据，可开启 gRPC 消息压缩，降低

⽹络流量。

1

server.grpc-compression-type: gzip

优化跨区域 AZ3 的TiKV 节点⽹络，修改 TiKV 的如下参数，拉⻓跨区域副本参与选举的时间，避免跨区域 TiKV 中的副本参与 Raft 选举。

1

raftstore.raft-min-election-timeout-ticks: 1000

raftstore.raft-max-election-timeout-ticks: 1200

2

调度设置。在集群启动后，通过 tiup ctl:<cluster-version> pd ⼯具进⾏调度策略修改。修改 TiKV Raft 副本数按照安装时规划好的副本数进⾏设置，在本例中为 5 副本。

1

config set max-replicas 5

禁⽌向跨区域 AZ 调度 Raft Leader，当 Raft Leader 在跨区域 AZ 时，会造成不必要的本区域 AZ 与远程 AZ 间的⽹络消耗，同时，⽹络带宽和延迟也会对 TiDB 的集群性能产⽣影响。

1

config set label-property reject-leader dc 3

设置 PD 的优先级，为了避免出现跨区域 AZ 的 PD 成为 Leader，可以将本区域 AZ 的 PD 优先级调⾼（数字越⼤，优先级越⾼），将跨区域的 PD 优先级调低。

1

member leader\_priority PD-10 5 member leader\_priority PD-11 5 member leader\_priority PD-12 5 member leader\_priority PD-13 5 member leader\_priority PD-14 1

2

3

4

5