RPC技术分享

headline

1 RPC是什么

2 RPC出现的场景

3 RPC协议

4 服务器模型

5 分布式RPC

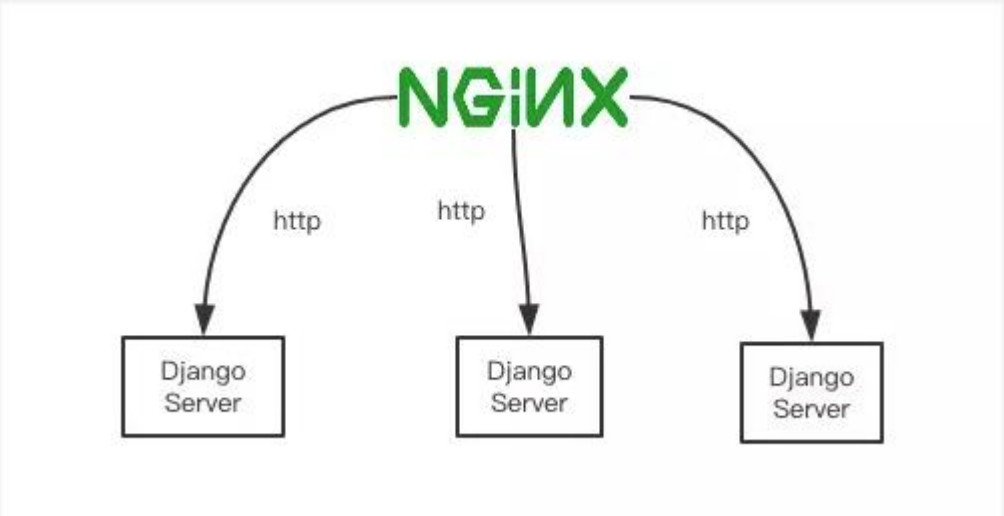
RPC是什么

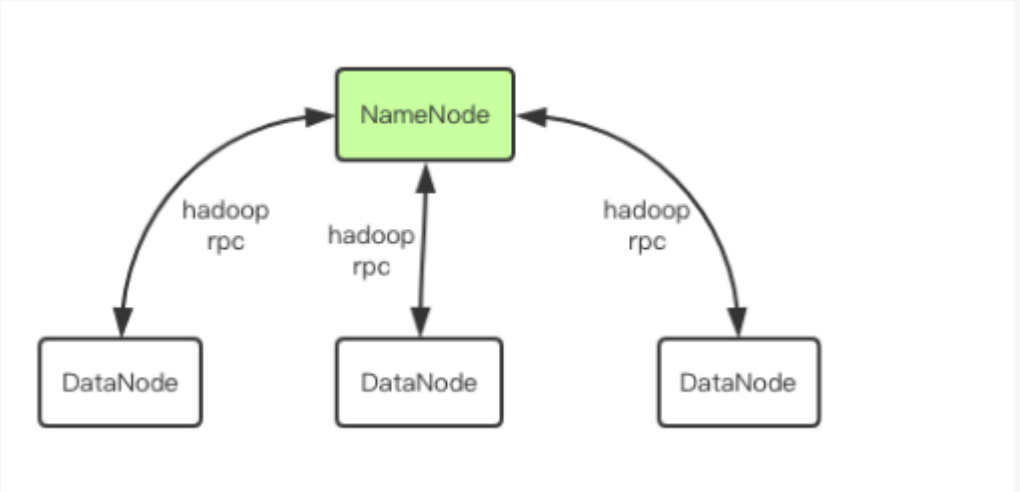
- 一句话解释:本地函数→远程函数
- 原因:大服务→小服务 单机—集群
- 物理隔离 常见的交互方案:
  - RPC
  - 分布式消息队列
  - HTTP请求
  - 数据库
  - 分布式缓存

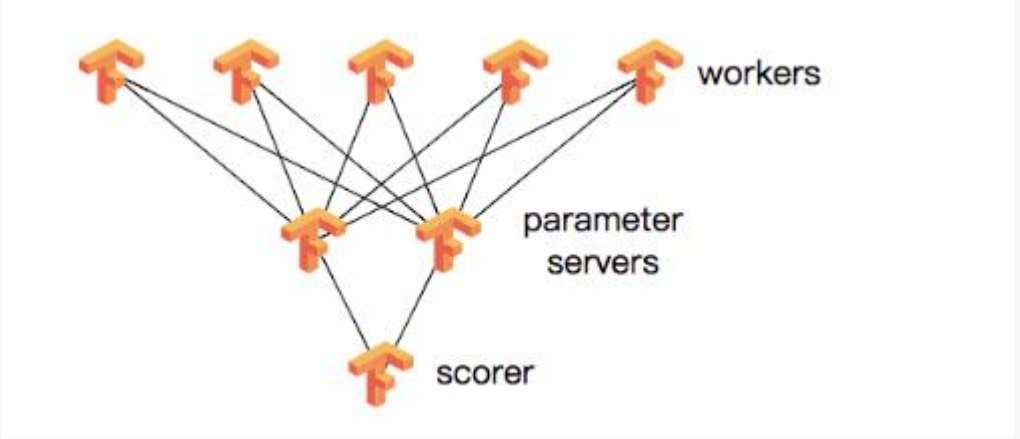
常见的RPC中间件

- dubbo(alibaba)
- dubboX(当当)
- Motan(新浪)
- Thrift(Facebook)
- gRpc(Google)

RPC使用场景


NGINX
http
http
http
Django Server
Django Server
Django Server


NameNode
hadoop rpc
hadoop rpc
hadoop rpc
DataNode
DataNode
DataNode


workers
parameter servers
scorer

RPC协议

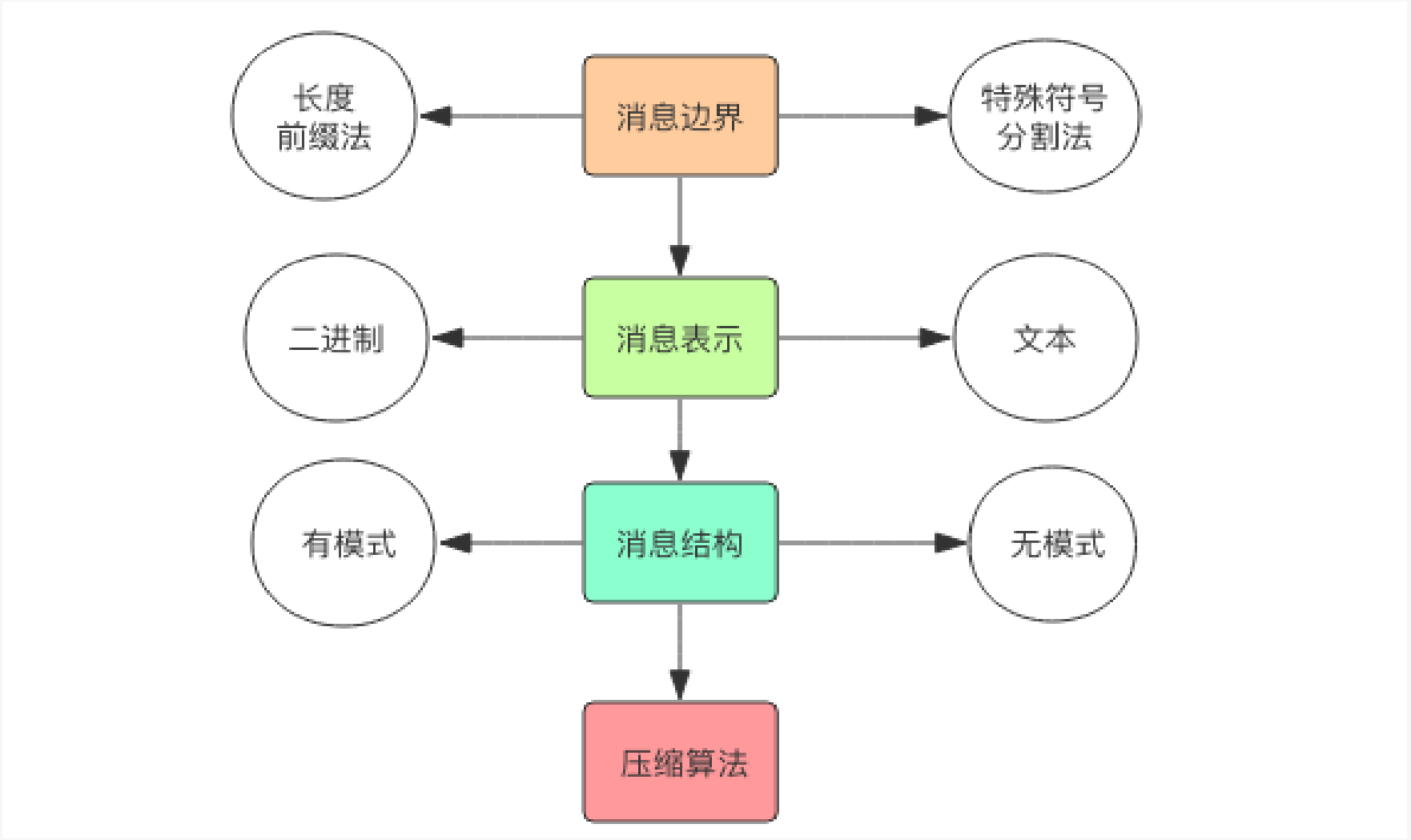
1 消息边界

2 消息表示

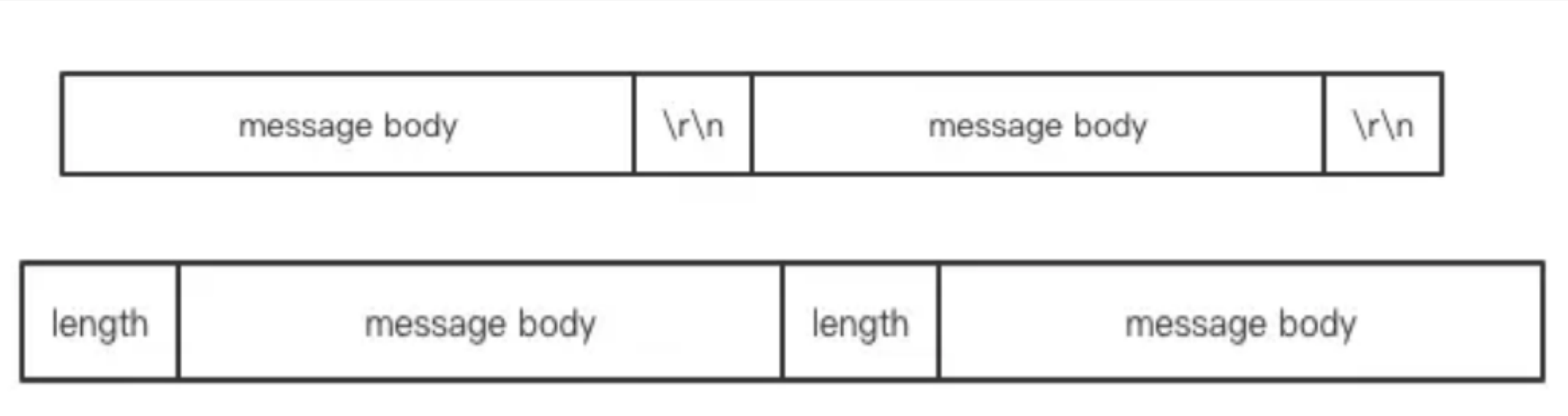
3 消息结构

4 消息压缩

RPC协议



RPC协议(消息边界)


message body
\r\n
message body
\r\n
length
message body
length
message body

RPC协议(消息表示)

1 有模式

```
[
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
]
```

2 无模式

```
@Override
    public void writeImpl() {
        writeByte((byte) this.platformId);
        writeLong(deviceId);
        writeStr(productId);
        writeStr(channelId);
        writeStr(versionId);
        writeStr(phoneModel);
    }...

@Override
    public void readImpl() {
        this.platformId = readByte();
        this.deviceId = readLong();
        this.productId = readStr();
        this.channelId = readStr();
        this.versionId = readStr();
        this.phoneModel = readStr();
    }...
```

RPC协议(压缩和编码)

压缩算法:

平衡CPU和网络带宽

编码:

1.变长编码

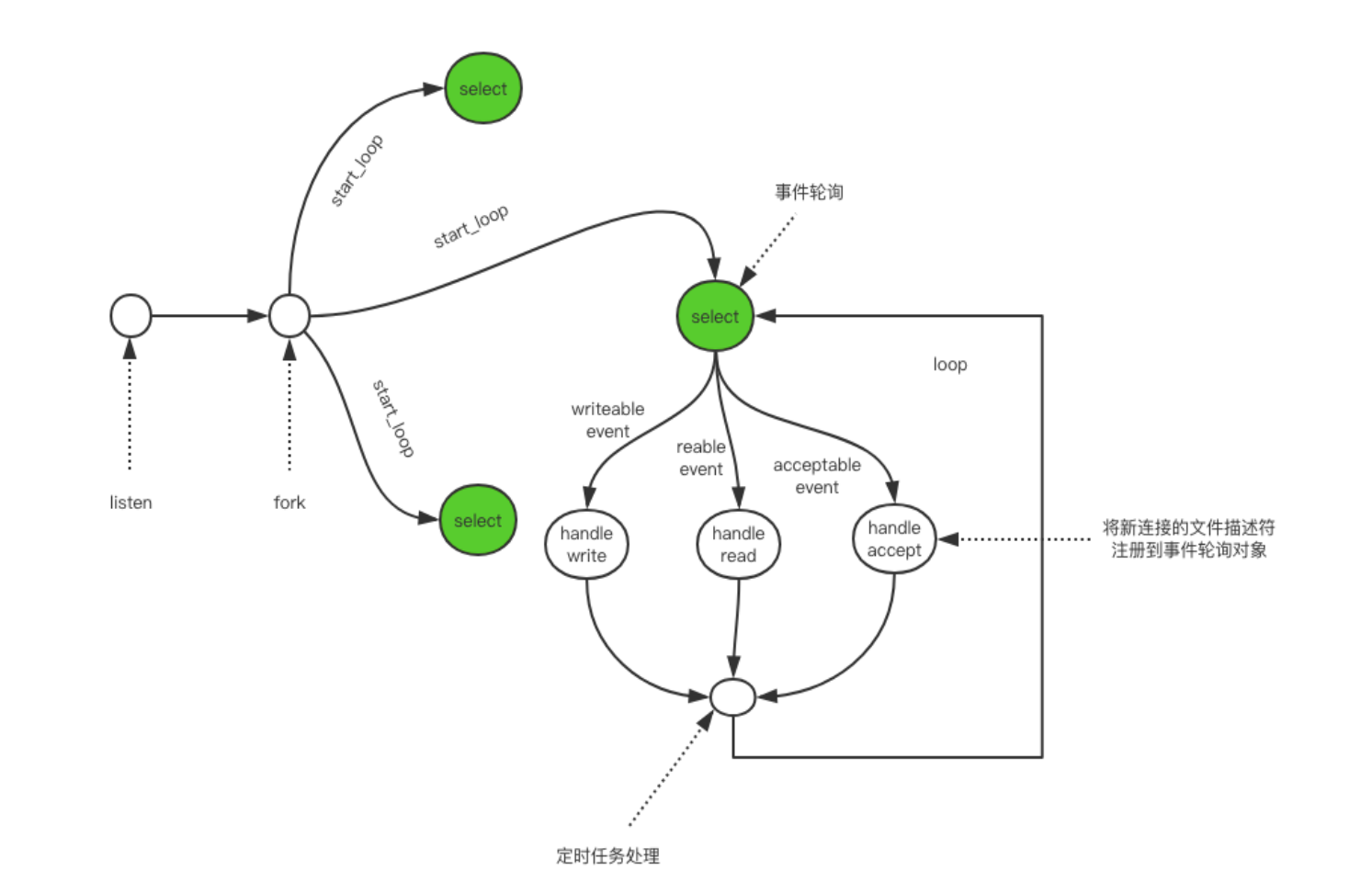
2.Zigzag编码

```
0 => 0
-1 => 1
1 => 2
-2 => 3
2 => 4
-3 => 5
3 => 6
```

RPC服务器模型

- 01 单线程同步
- 02 多线程同步
- 03 单线程异步
- 04 多线程异步
- 05 preforking

RPC服务器模型- - 同步&异步



RPC服务器模型- - 同步&异步

1 Select & poll

2 Kqueue & epoll

分布式RPC

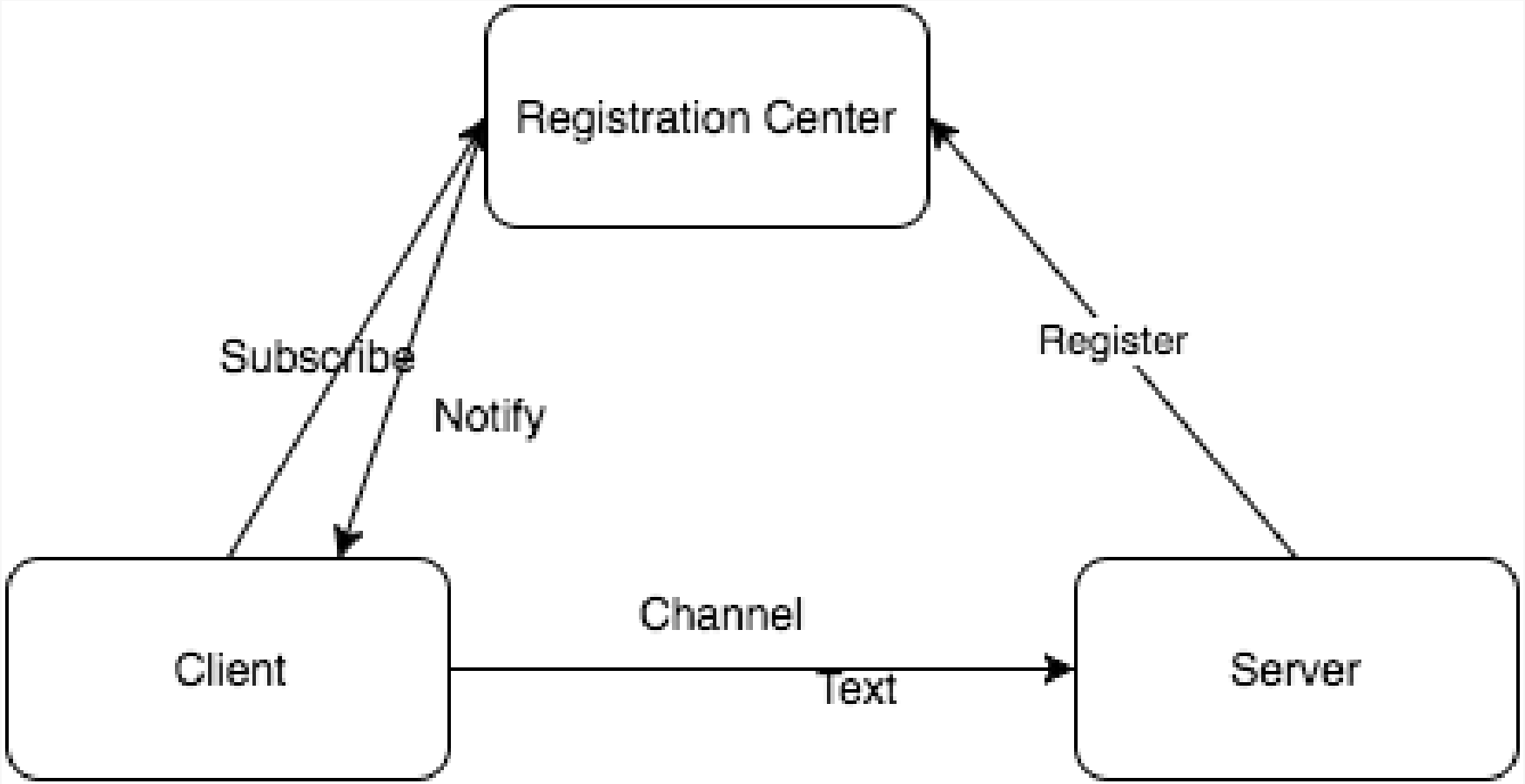
distribution

1 服务注册与发现

2 高可用

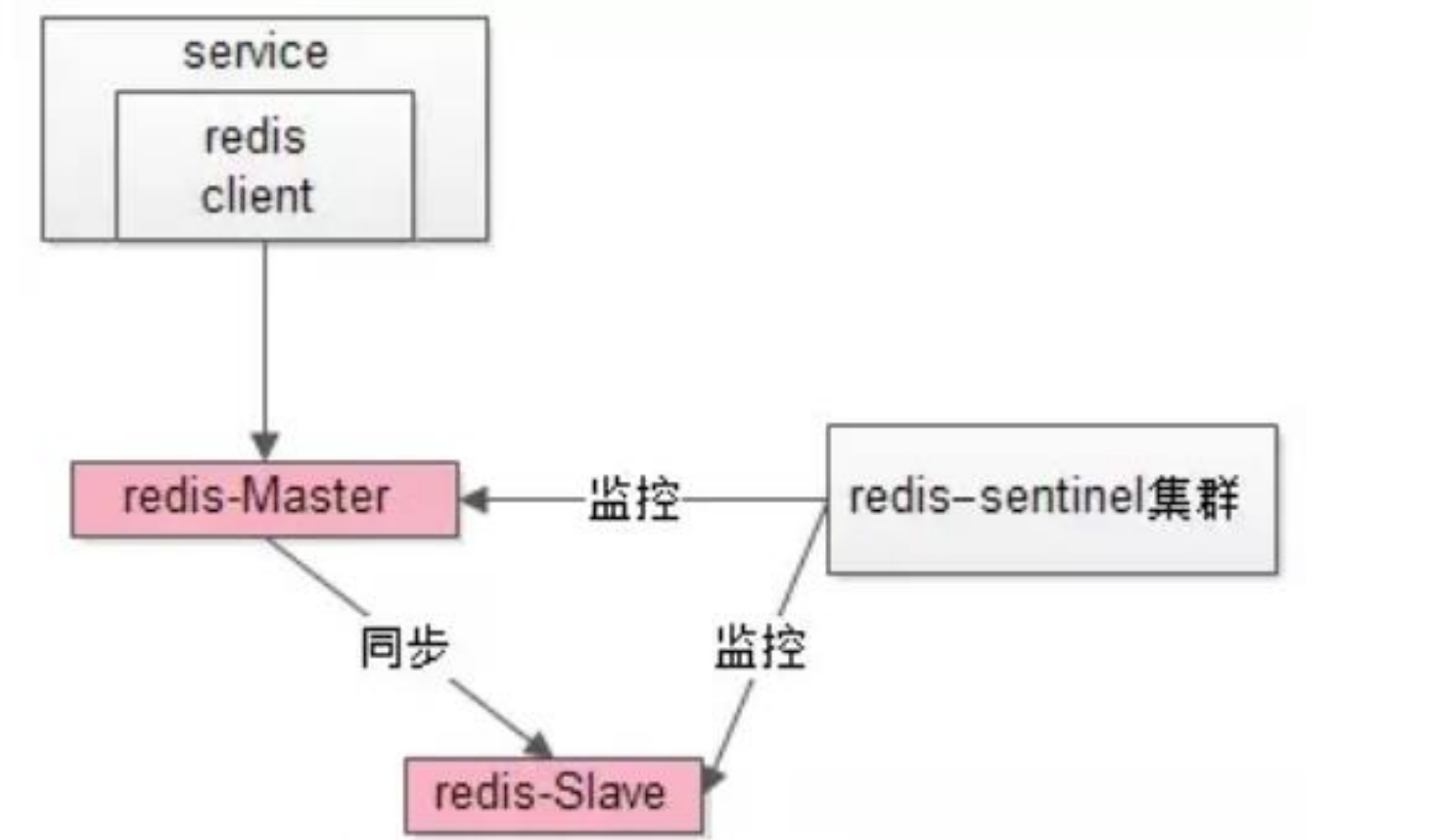
3 负载均衡

分布式RPC- - 服务注册与发现



分布式RPC- - 高可用

冗余
故障转移



分布式RPC- - 负载均衡

1. RoundRobin
2. WeightedLeastConnection
3. WeightedRoundRobin
4. DAWRRRLB
5. Random
6. ConsistentHash

负载均衡- - 加权最少连接法

$$P_i = \min\left(\frac{C(N_i)}{W(N_i)}\right) \tag{1}$$

负载均衡- - WeightedRoundRobin

1.

W(weight)

E(effectiveWeight)

C(currentWeight)

T(totalWeight)

2.每次轮询:C = C + E

3.访问成功:C = C – T

4.请求异常:E = E - 1

负载均衡- - WeightedRoundRobin

请求序号	current_weight before selected	select peer	current_weight after selected
1	{ 4, 2, 1 }	a	{ -3, 2, 1 }
2	{ 1, 4, 2 }	b	{ 1, -3, 2 }
3	{ 5, -1, 3 }	a	{-2, -1, 3 }
4	{ 2, 1, 4 }	c	{ 2, 1, -3 }
5	{ 6, 3, -2 }	a	{ -1, 3, -2 }
6	{ 3, 5, -1 }	b	{ 3, -2, -1 }
7	{ 7, 0, 0 }	a	{ 0, 0, 0 }

负载均衡- -DAWRRRLB(动态自适应权重轮询)

C(i) 单节点cpu占用率

M(i)单节点内存占用率

D(i)单节点IO占用率

B(i)单节点带宽占用率

$$S_c(\text{Total}) = \sum_{i=1}^{n} C(N_i) \quad (2)$$

$$S_m(\text{Total}) = \sum_{i=1}^{n} M(N_i) \quad (3)$$

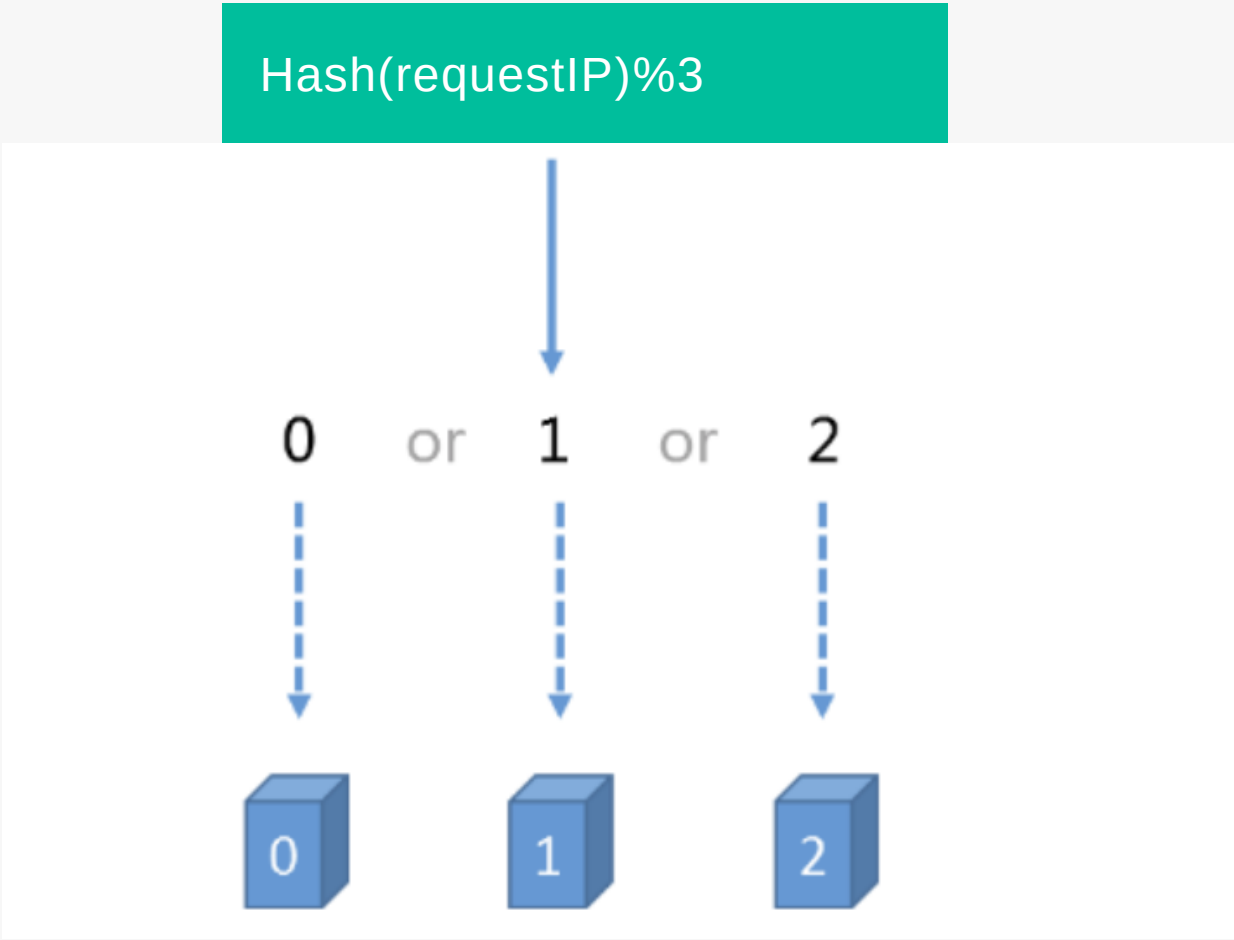
$$S_d(\text{Total}) = \sum_{i=1}^{n} D(N_i) \quad (4)$$

$$S_b(\text{Total}) = \sum_{i=1}^{n} B(N_i) \quad (5)$$

负载均衡- -DAWRRRLB(动态自适应权重轮询)

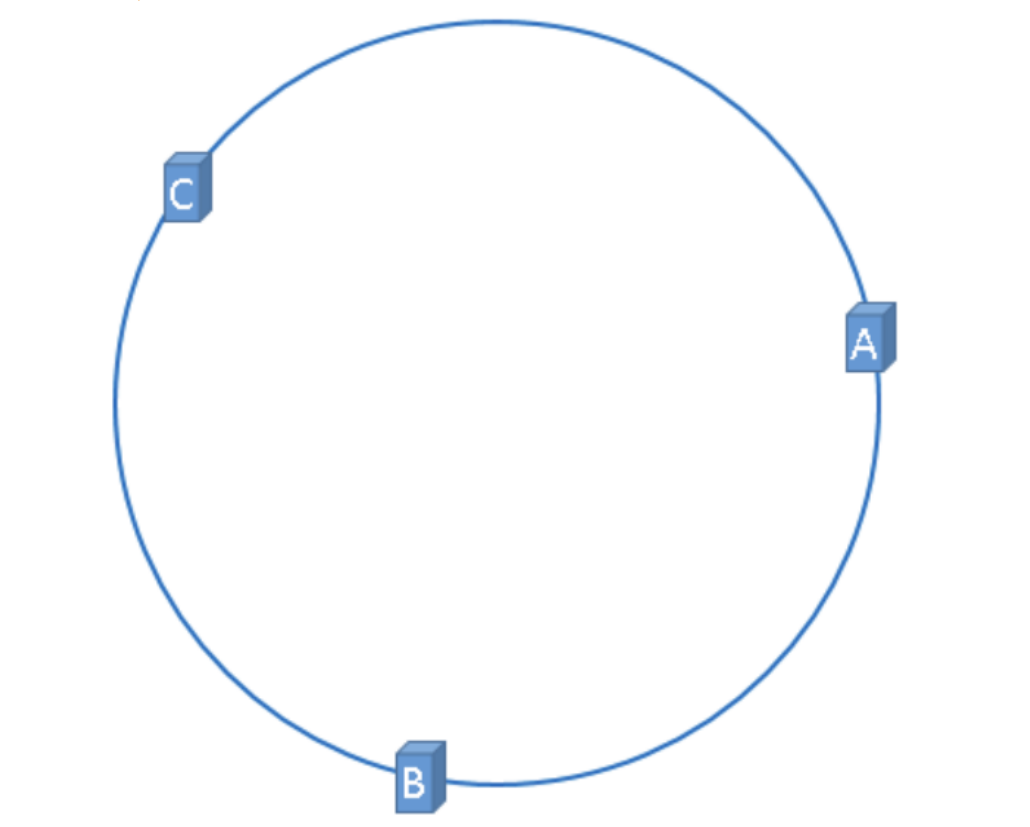
$$
\begin{aligned}
W_p(N_i) = {} & K_c * (C(N_i) / S_c(\text{Total})) + \\
& K_m * (M(N_i) / S_m(\text{Total})) + \\
& K_d * (D(N_i) / S_d(\text{Total})) + \\
& K_b * (B(N_i) / S_b(\text{Total}))
\end{aligned}
\qquad (7)
$$

负载均衡- -ConsistentHash

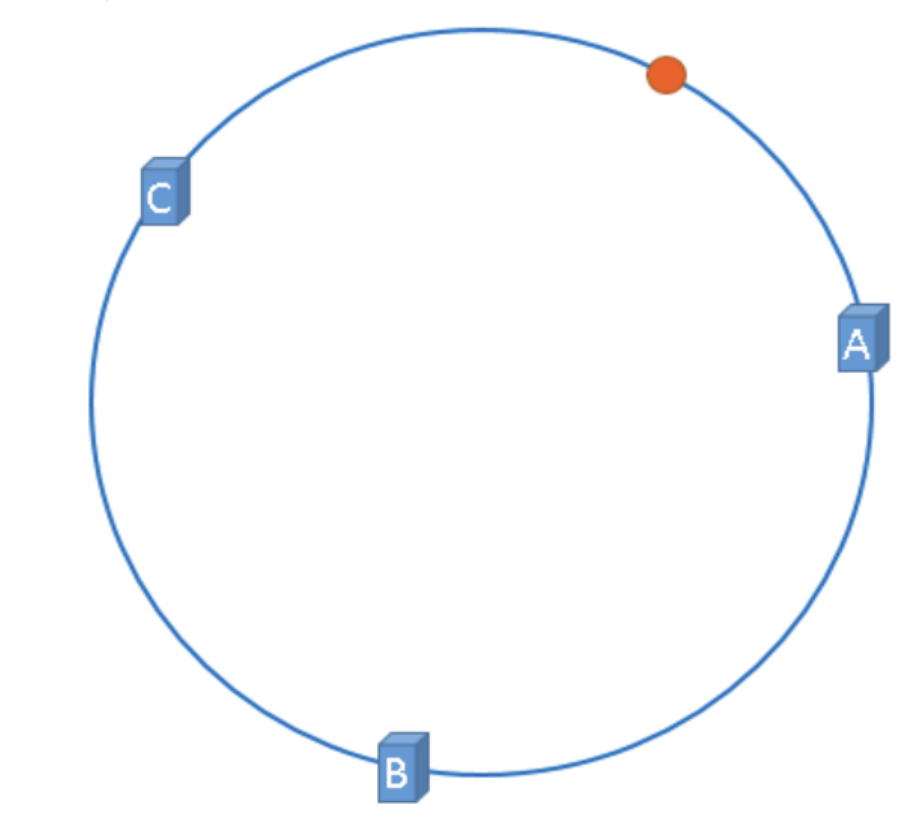


负载均衡- -ConsistentHash

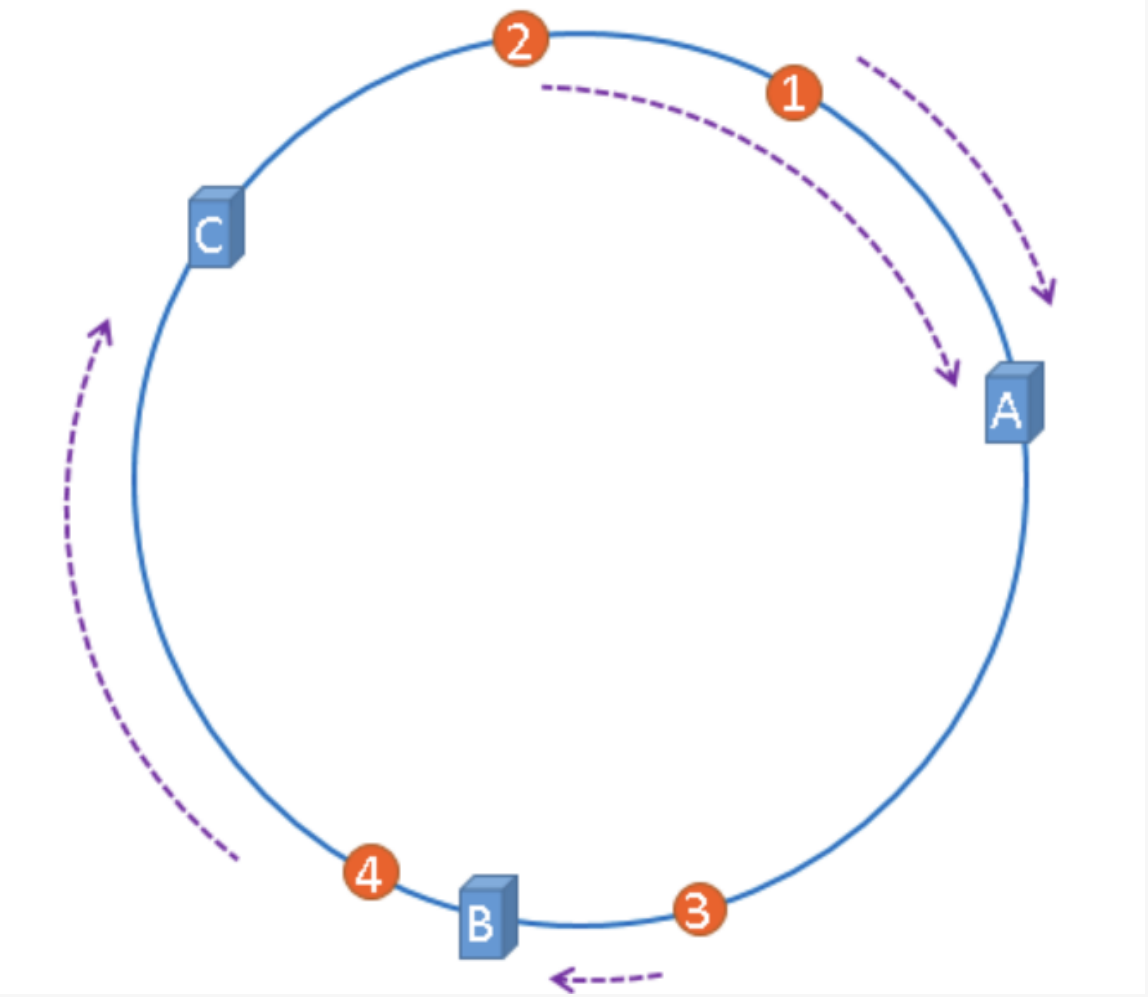
Hash(server IP)%(2^32)



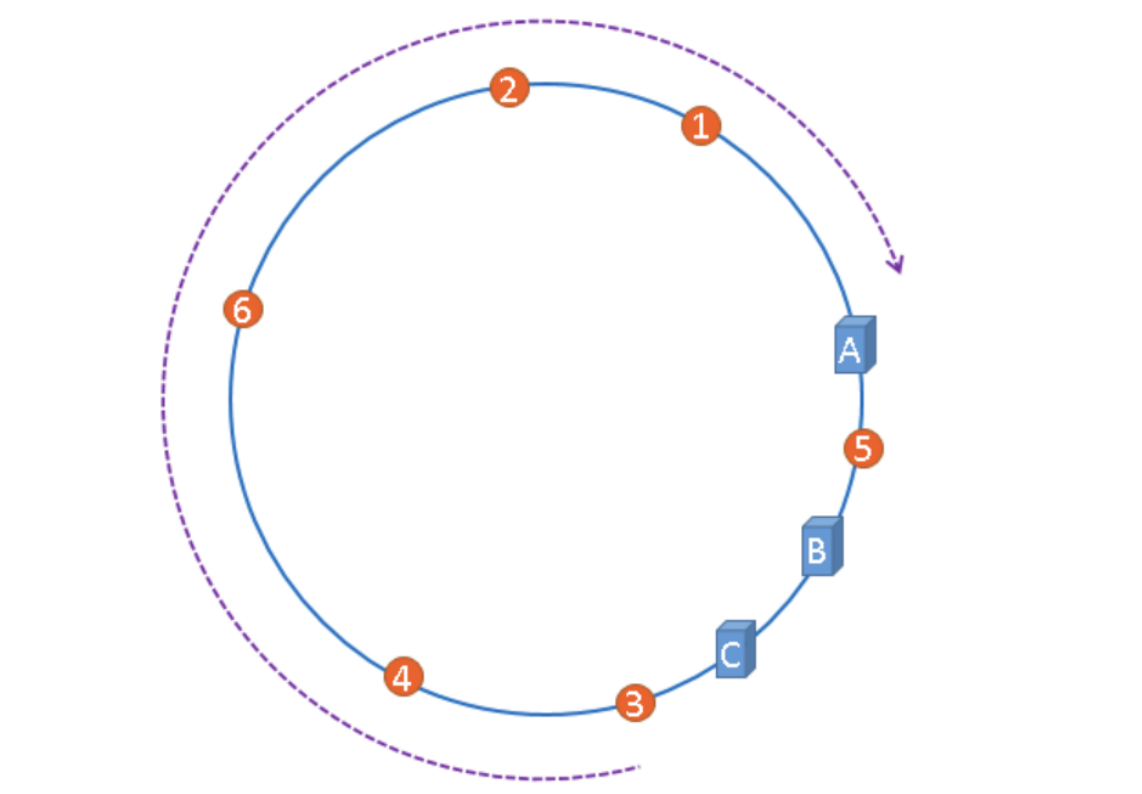
Hash(request IP)%(2^32)

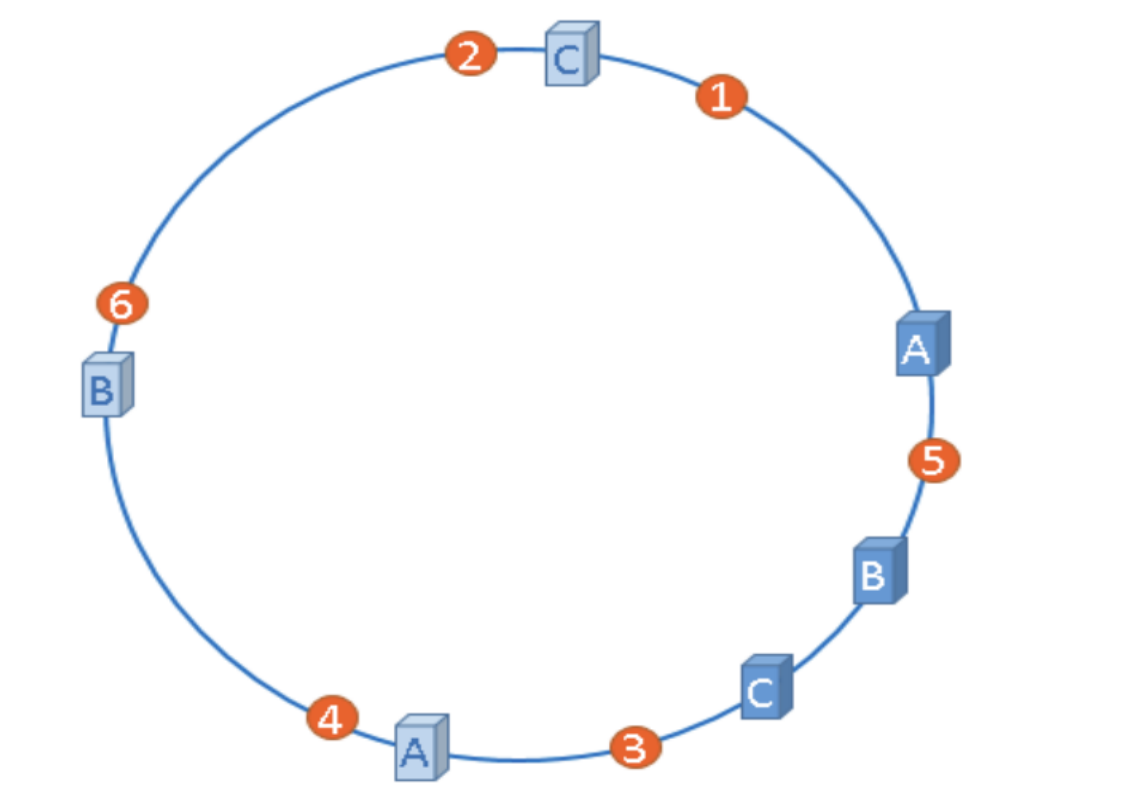


负载均衡- -ConsistentHash



负载均衡- -ConsistentHash





Question

1.写出一个你认为会使用类RPC技术的场景

2.随便写出两种刚刚讲过的负载均衡方法

3.一串数字用zigzag编码后是6556,那么原来要表达的数字串是什么