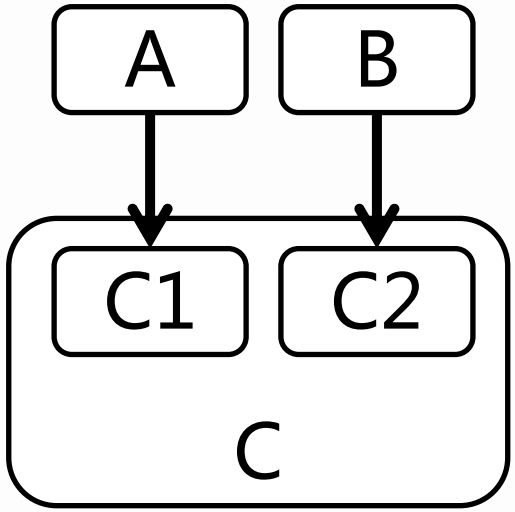

# 分布式系统稳定性模式

——淘宝-小邪

# 分布式系统稳定性模式-22条

1. 隔离
2. 内存溢出控制
3. 循环阀值
4. 超时
5. 限制请求阻塞的Size
6. 异步调用
7. 限流
8. 降级
9. 开关
10. 热点缓存
11. 缓存容灾
12. 包版本冲突
13. 内部调用优先
14. 依赖诊断和调试
15. 日志跟踪
16. 依赖识别
17. 依赖简化
18. Beta发布
19. 服务治理
20. 容量规划
21. 演练
22. 监控报警

- **1.隔离**
  - 通过把系统分割成独立模块，使得单系统的部分模块发生故障的时候可以保障大部分的功能依旧可用
  - 方法
    - 模块线程隔离
    - 虚拟机、容器、实例隔离
    - 服务器分组隔离
    - 机房隔离



- **2.内存溢出控制**
  - 在临时性单机缓存，务必确保内存使用size是可控的

```
Map.put(key,value);
List.add(value);
```

- **3.循环阀值**
  - 当循环的条件来自网络，则务必进行阀值限制

```
if (remoteNumber > MAX_COUNT)
        remoteNumber = MAX_COUNT;

for ( int i = 0; i < remoteNumber; i++ )  {
    doSomething();
}
```

- **4.超时**
  - 设置恰当的网络超时时间，可以有效的保障系统调用者不被临时性故障拖垮
  - 注意不要设置太长也不要设置太短，以满足正常流量为佳

    目标：在调用超时的情况下，正常QPS流量刚好消耗完可以接受的线程数量

    估算公式：正常QPS = 1000毫秒/超时平均响应时间 * 可以接受的最大线程数量

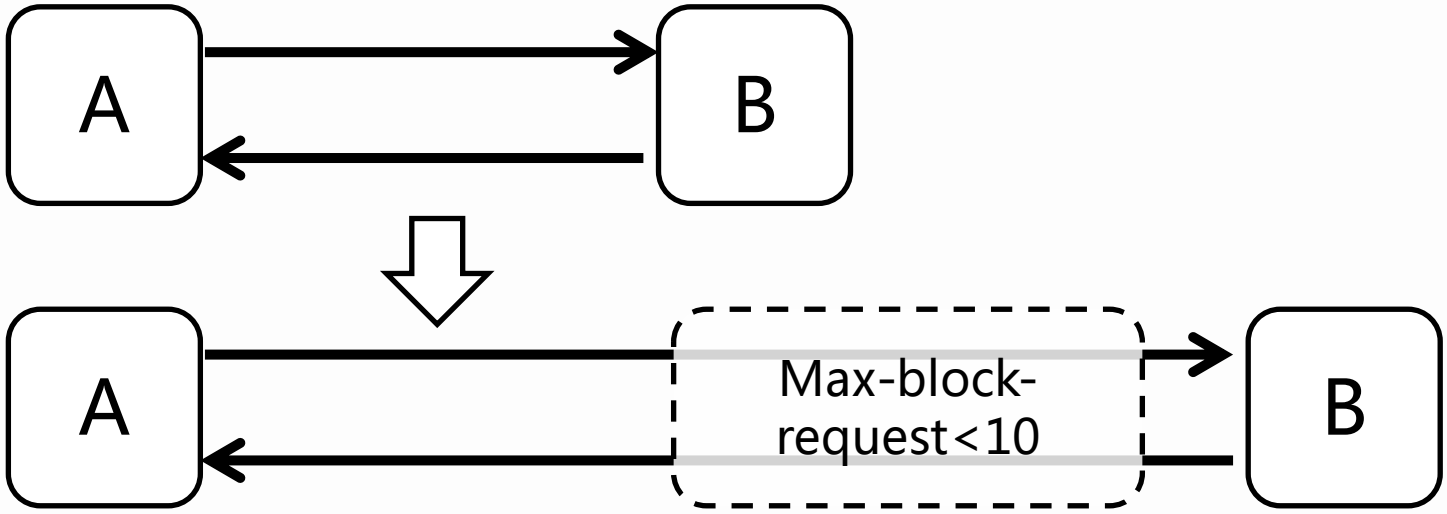
    如：

    超时平均响应时间= 1000*可以接受的最大线程数量/正常QPS

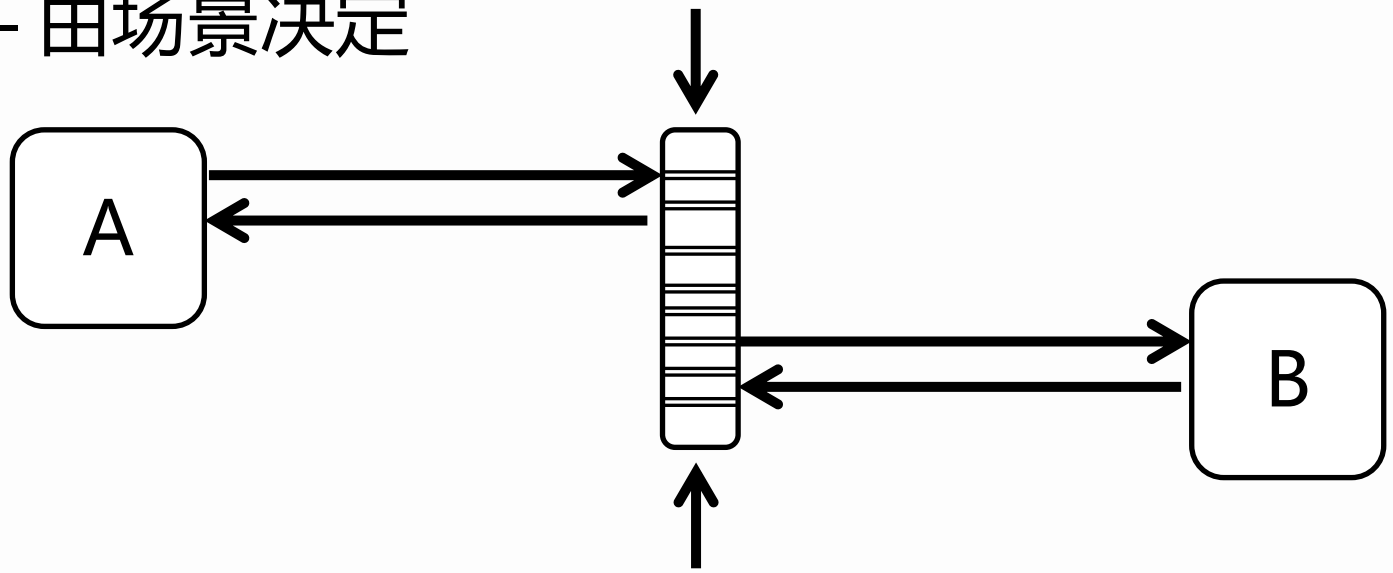
    超时平均响应时间= 1000*100/30 = 3333ms

    如果不知道怎么设置的时候，通常可以设置3秒

- **5.限制请求阻塞的Size**
  - 本质和超时类似，不过相比较超时而言对系统更加友好
    - 可以保证在系统所能承受的Max流量情况下工作依旧正常



- **6.异步调用**
  - 解决remoting故障时候对系统减少影响
  - 注意控制异步队列大小
  - 由场景决定

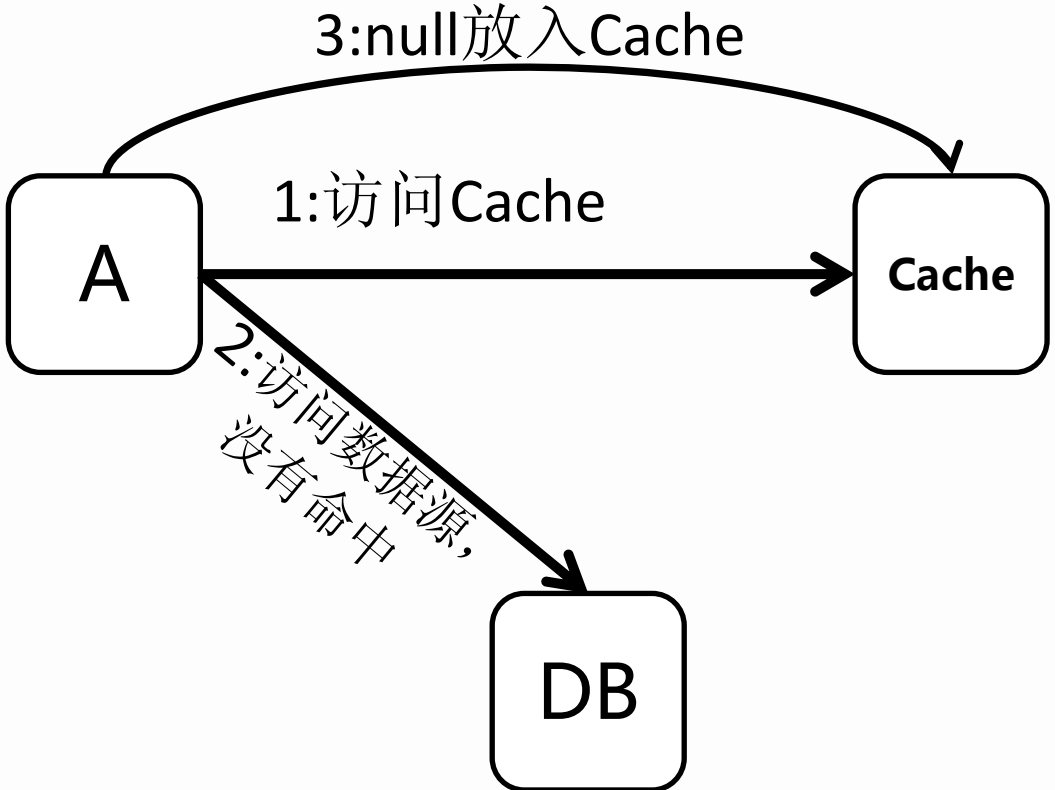


- **7.限流**
  - 当流量超过系统容量的时候，要能自动的拒绝超过的请求，并且告知调用方原因

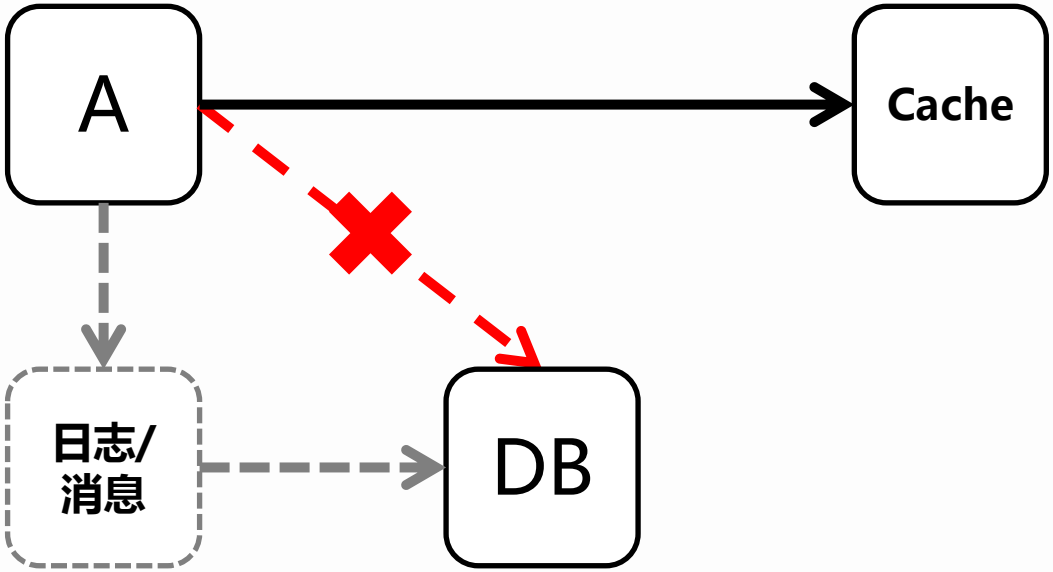
- **8.降级**
  - 当系统对很多系统进行依赖，如果有一个依赖系统发生故障，则系统本身需要对该故障系统进行降级，防止故障扩散
  - 被降级的系统恢复之后，系统能自动恢复对其的调用

- **9.开关**
  - 各种有风险的业务或者模块上线的时候可以加上开关，关键时刻可以手工关闭功能
  - 开关当前状态要能被监控
  - 开关的使用频率可能比较低，所以要有比较好的传承方法

- **10.热点缓存**
  - 当缓存没有命中，则访问原始资源（database），如果原始资源也不存在，需要把null放入缓存



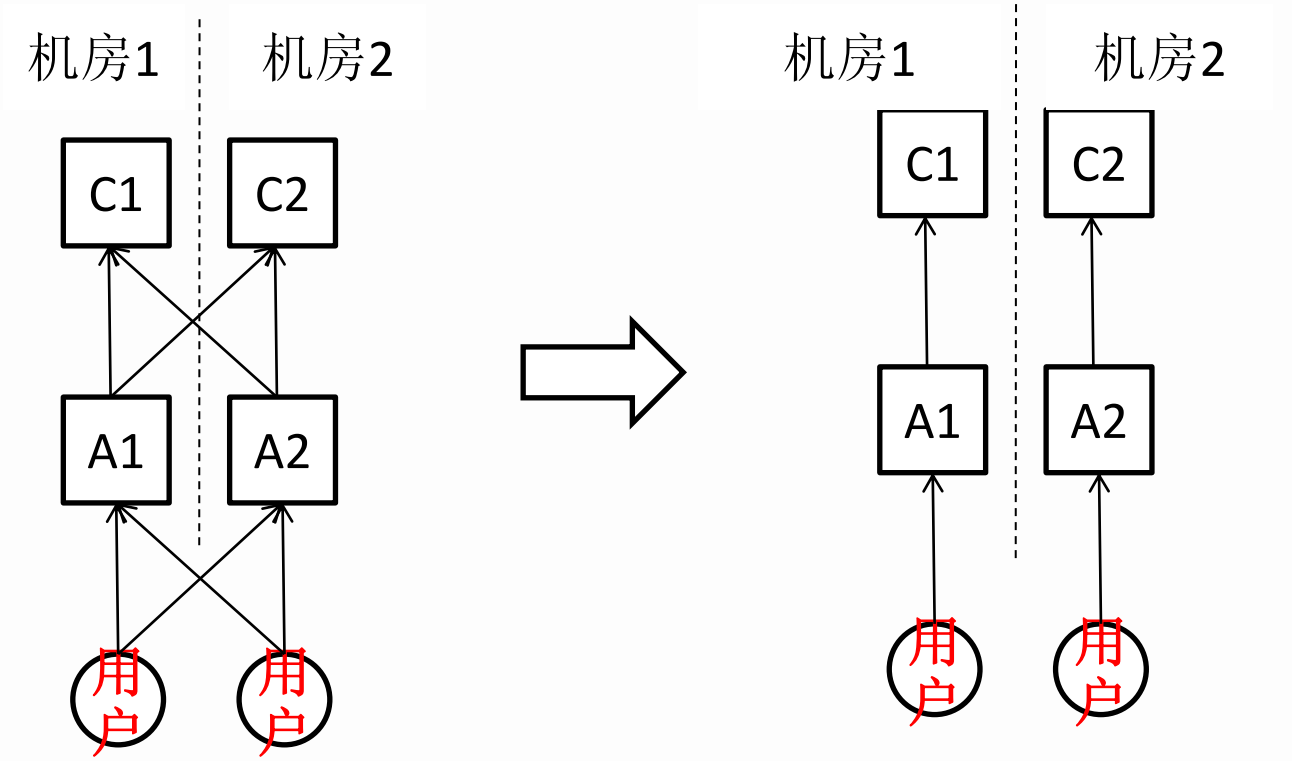
- **11.缓存容灾**
  - 缓存不仅仅是缓存，同时也可以作为数据源的容灾备案



- **12.避免依赖包的版本冲突**
  - 务必确保系统使用的依赖包（jar包）的版本在系统里是唯一的

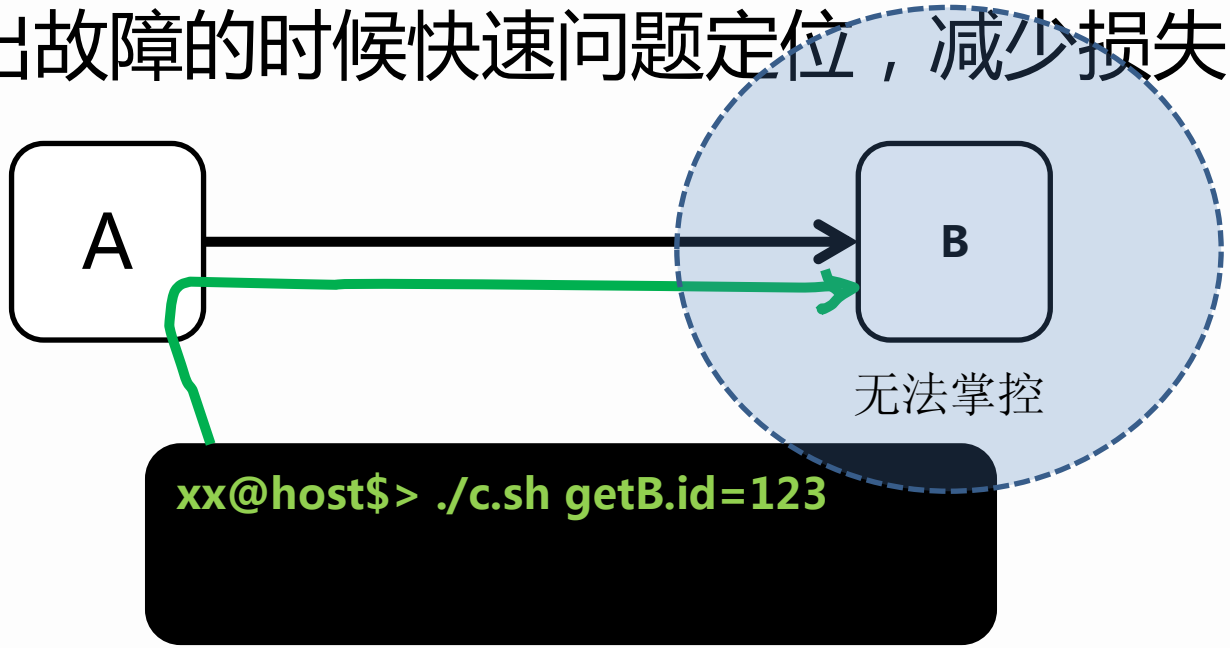
    java.lang.NoSuchMethodException

- **13.内部调用优先**

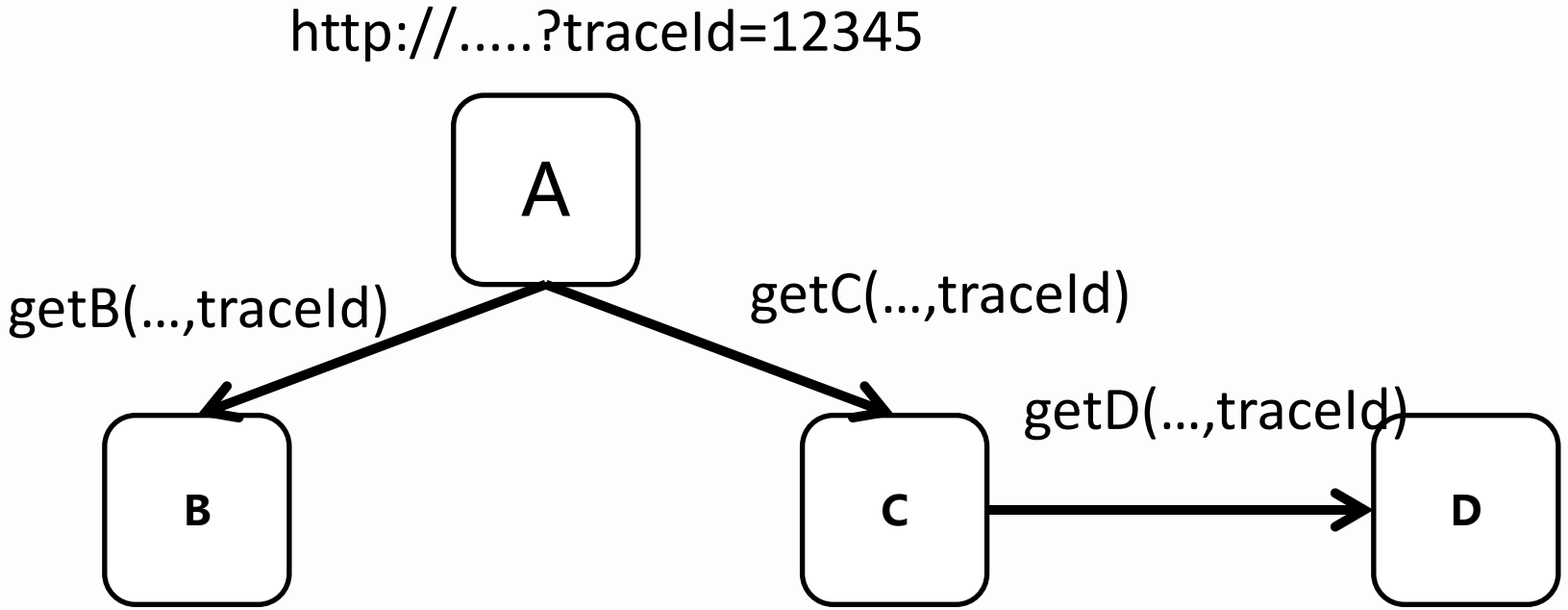


- **13.内部调用优先**
  - 如果要调用的服务在本地服务器，或者同机房的服务器，则优先调用，当本机或者同机房的集群发生故障的比例达到一定程度的时候，才进行跨机房调用，这个将大大减少因为机房间网络故障带来的损失

- **14.依赖诊断和调试**
  - 系统对依赖的远程系统的数据能非常方便的通过某种途径（工具）进行查询，这个将帮助系统出故障的时候快速问题定位，减少损失

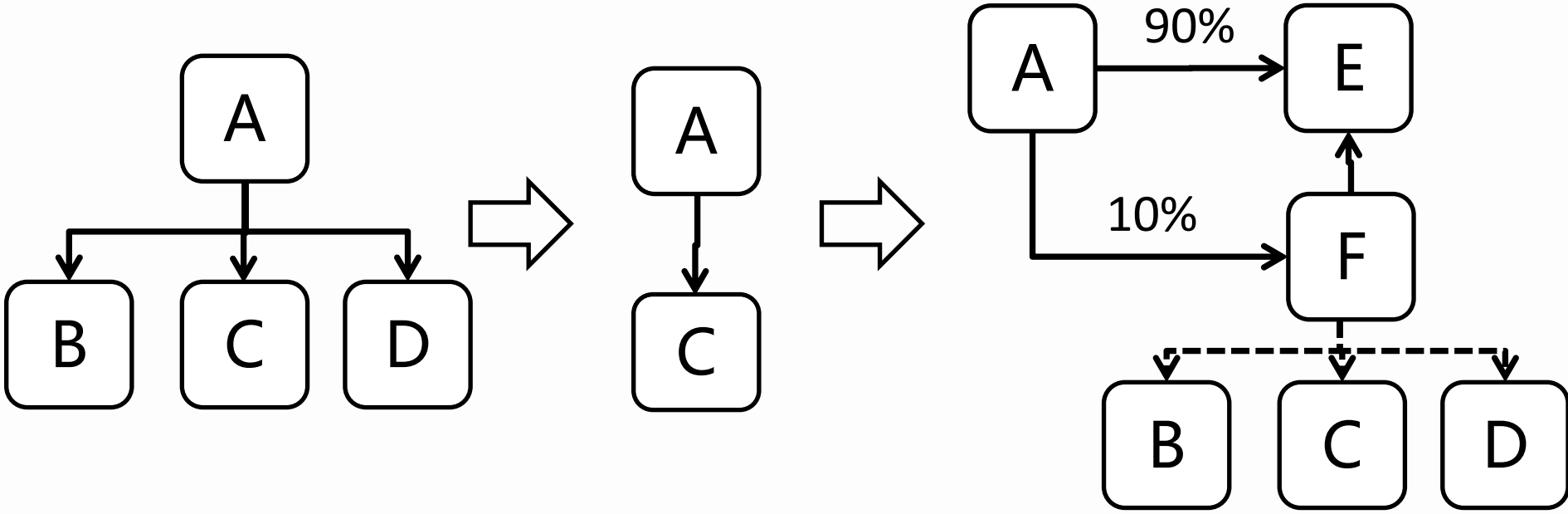


- **15.日志跟踪**
  - 分布式系统的日志通过traceID进行串联，让系统间的每次调用都非常清晰，对系统日常运维优化，对排查和调试问题帮助都很大
  - 注意这类接口不能对外暴露



- **16.依赖识别**
  - 分布式系统的依赖繁杂，每个系统的维护者必须非常清楚自己依赖了哪些系统，以及依赖的强弱，若发现依赖强弱不符合预期，则需要尽快改进

- **17.依赖简化**
  - 依赖越少系统越稳定
  - 采用客户端异步依赖



- **18.日常/预发/Beta发布**
  - 新功能上线先进行线上beta，如果发生故障，则可控
  - 业务预期测试可以采用灰度发布

- **19.服务治理**
  - 如果系统的服务提供给其他系统调用，则系统本身务必有比较强大的自我控制能力，主要包括：限流，降级，调用白名单，账号密码，调用量监控等

- **20.容量规划**
  - 务必清楚重要系统的容量，至少每周评估一次，大促之前需要额外评估
  - 解决大促容量不足，发现性能瓶颈风险，硬件计划增加

- **21.演练**
  - 定时进行系统风险演练，可以及时发现潜在的风险，以及帮助我们在发生故障的时候快速执行应急措施，减少故障时间

- **22.监控报警**
  - 对关键路径的系统进行详细的监控，发现问题自动报警
  - 包括
    - 硬件状态，网络状态（流量），操作系统，虚拟机
    - 应用层面：来源PV，依赖调用PV，响应时间
    - 以及各种业务数据

# 讨论时间