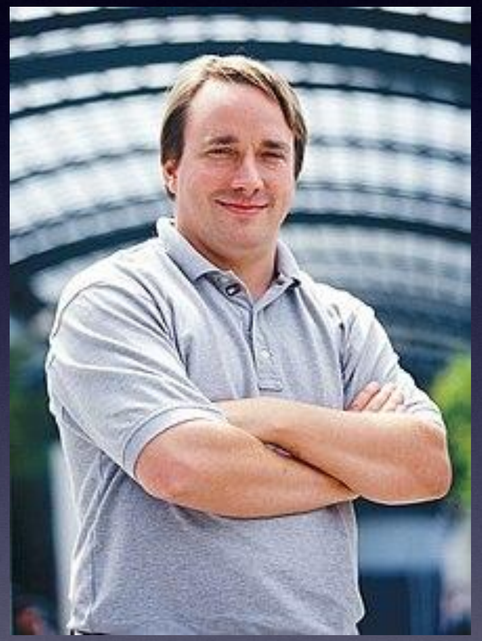
高可用分布式
Git 代码托管系统的构建

欧俊飞
Coding 资深架构师

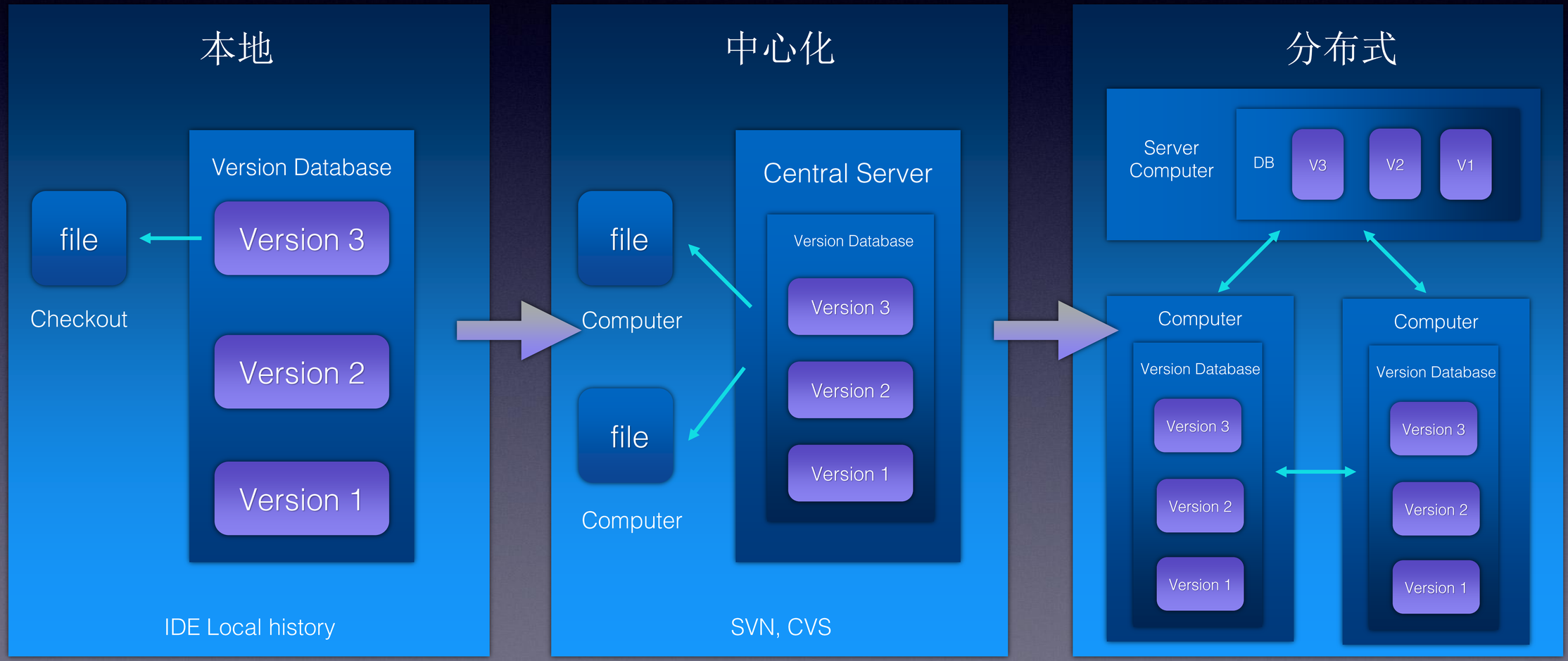
什么是 Git



Git 是一个分布式的版本控制软件，最初由 Linux 内核作者 Linus Torvalds 为管理 Linux 内核源代码而开发设计。

Git 具有强大的分支管理合并功能和天然的分布式特性，速度快效率高，这对于 Linux 内核这样的大型开源项目的管理来说非常重要。

版本控制系统演进


本地
Version Database
file
Version 3
Version 2
Version 1
Checkout
IDE Local history
中心化
Central Server
Version Database
Version 3
Version 2
Version 1
file
Computer
file
Computer
SVN, CVS
分布式
Server Computer
DB
V3
V2
V1
Computer
Version Database
Version 3
Version 2
Version 1
Computer
Version Database
Version 3
Version 2
Version 1

代码托管系统

代码托管系统是为方便存储和分享版本控制系统数据而开发的服务器端软件平台，现代化的代码托管平台已经发展为开源软件社区的集散地。

知名代码托管平台：

公有云代码托管挑战

公有云系统各方面的量级都会比传统系统提升一个档次，代码托管系统会面临很强的波峰波谷，频繁的 IO 操作，以及密集的 CPU 计算。

在设计公有云系统时，我们需要考虑以下几点：

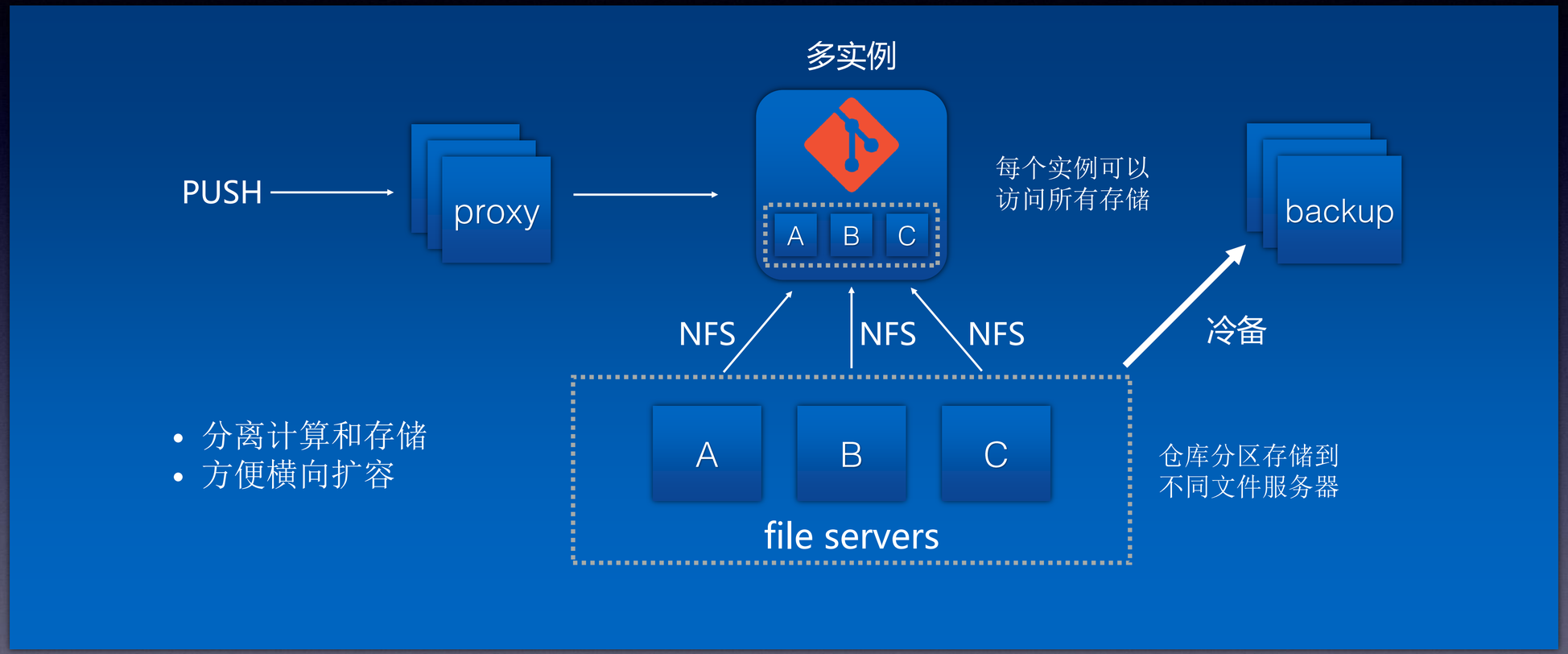
代码托管系统架构演进

单机

所有仓库存储在
本地超大磁盘里

- 单点故障
- 无法扩容
- 冷备恢复

多机 NFS



- 分离计算和存储
- 方便横向扩容

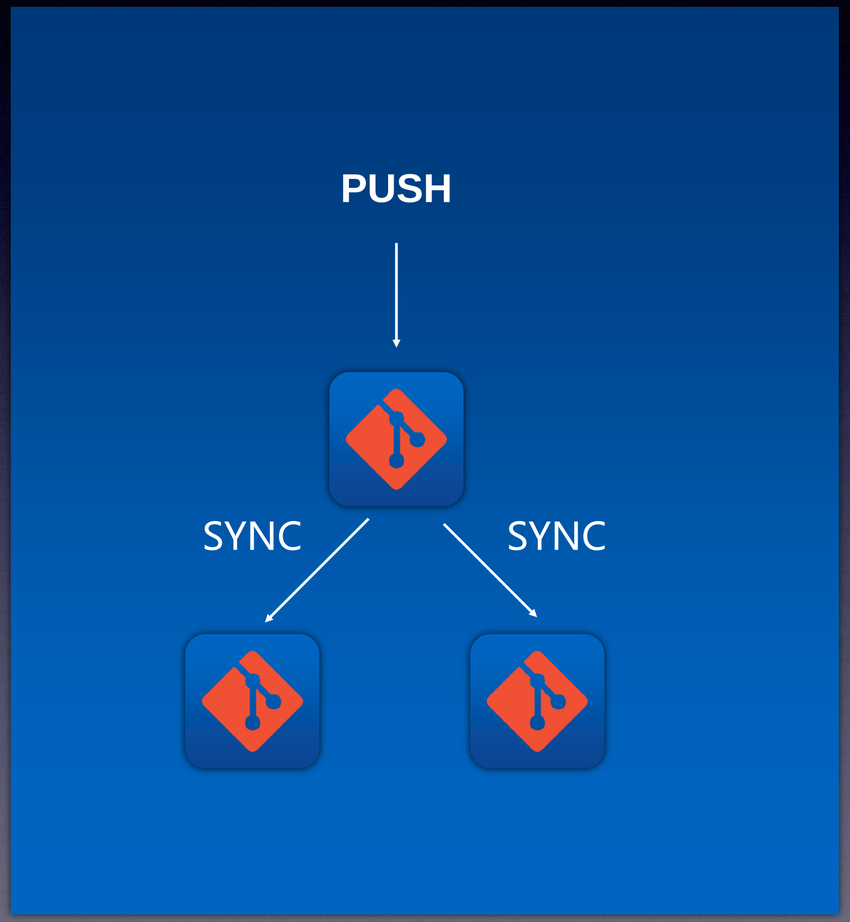
问题

- 性能损失
  - >网络传输增加延时
  - >需做大量缓存提速
- 安全性差
  - >仓库只有一份热拷贝
  - >冷备恢复可能丢失数据
- 运维困难
  - >仓库迁移需要停机
  - >故障只能人工处理

git 分布式存储
解决方案探索

- ketch 集群同步系统
- libgit2 存储后端
- 分布式文件系统
- DGit 分布式系统

ketch



集群同步系统

Google 开源作品，纯 java 实现，支持 master-salve，master-master 架构，使用 Raft 实现一致性保证。

- 开源出来的实现不完整，需要大量工作
- 需要 hack jgit 源码维护自己的分支
- 同步机制效率低下，实测延时太高
- 缺乏活跃维护（2016 年之后再无更新）
- 无生产环境成功案例（google 自己也还没开始用）

https://git.eclipse.org/r/64206

实测结果

传统本地文件存储

Test Summary

39	0	0	9.806s
tests	failures	ignored	duration

100%
successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
net.coding.git.repo	39	0	0	9.806s	100%

ketch 多存储同步

Test Summary

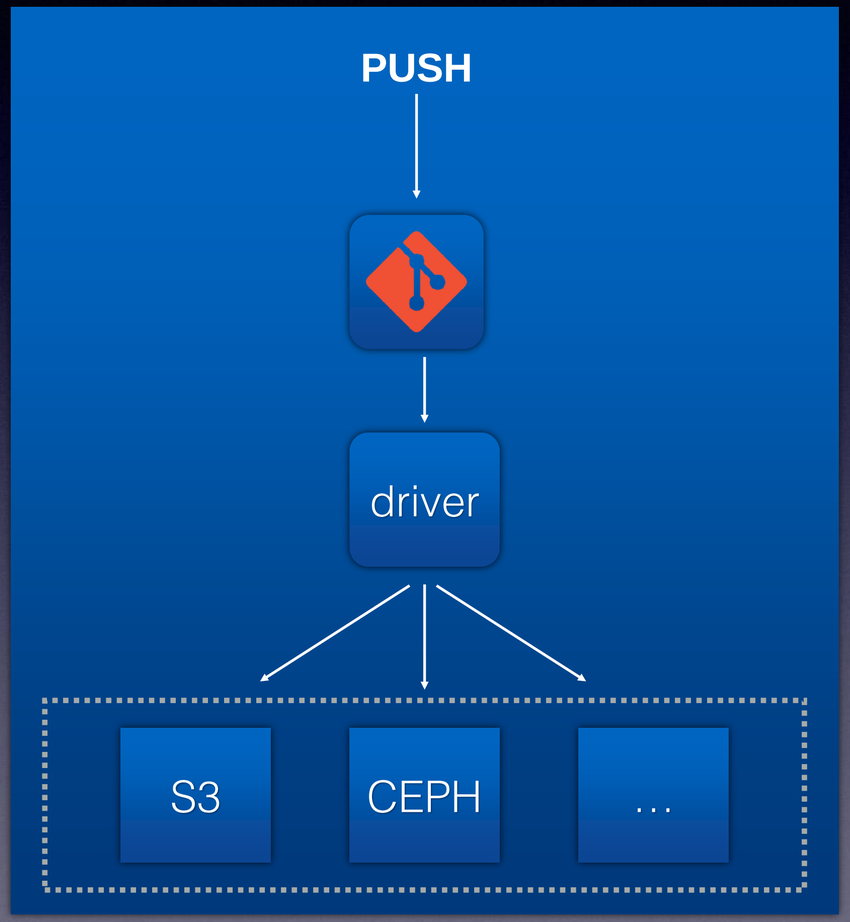
39	0	0	1m18.21s
tests	failures	ignored	duration

100%
successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
net.coding.git.repo	39	0	0	1m18.21s	100%

libgit2 存储后端



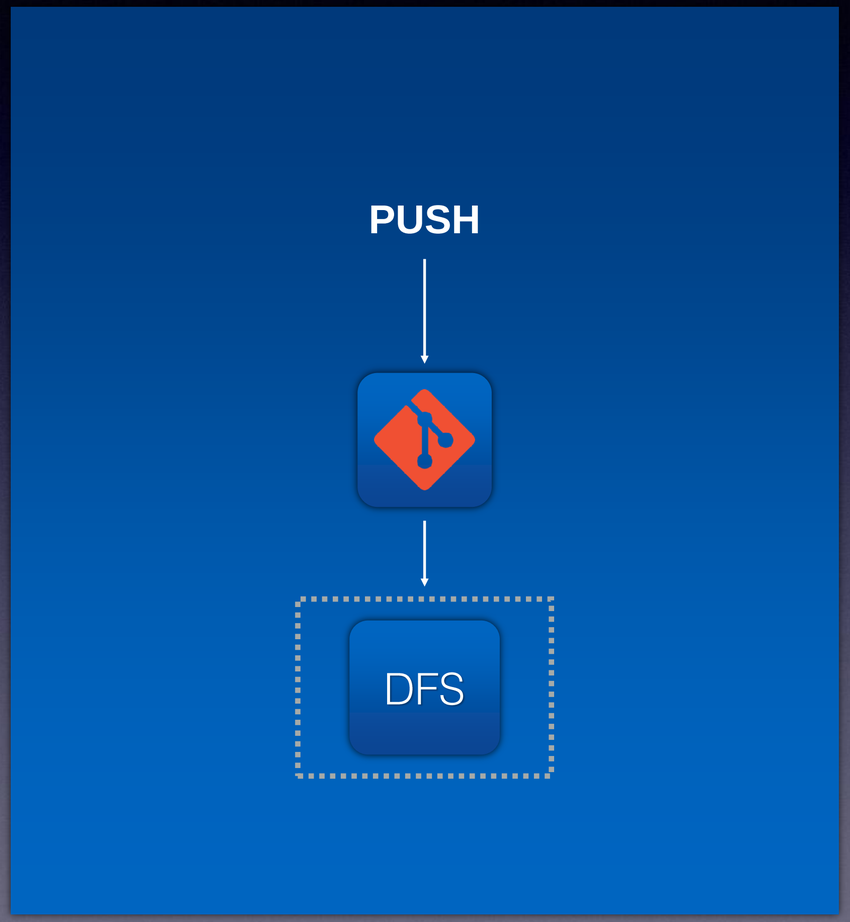
libgit2 提供了一套存储接口，使用者可以自行编写存储驱动把第三方存储系统嫁接到 git 上。

- 成本低，不需要对应用做大改
- 可直接利用第三方分布式系统

市面上没有适合 git 使用场景的分布式存储系统，只能作为一个过渡方案，不适合大规模使用。

https://github.com/libgit2/libgit2-backends

分布式文件系统

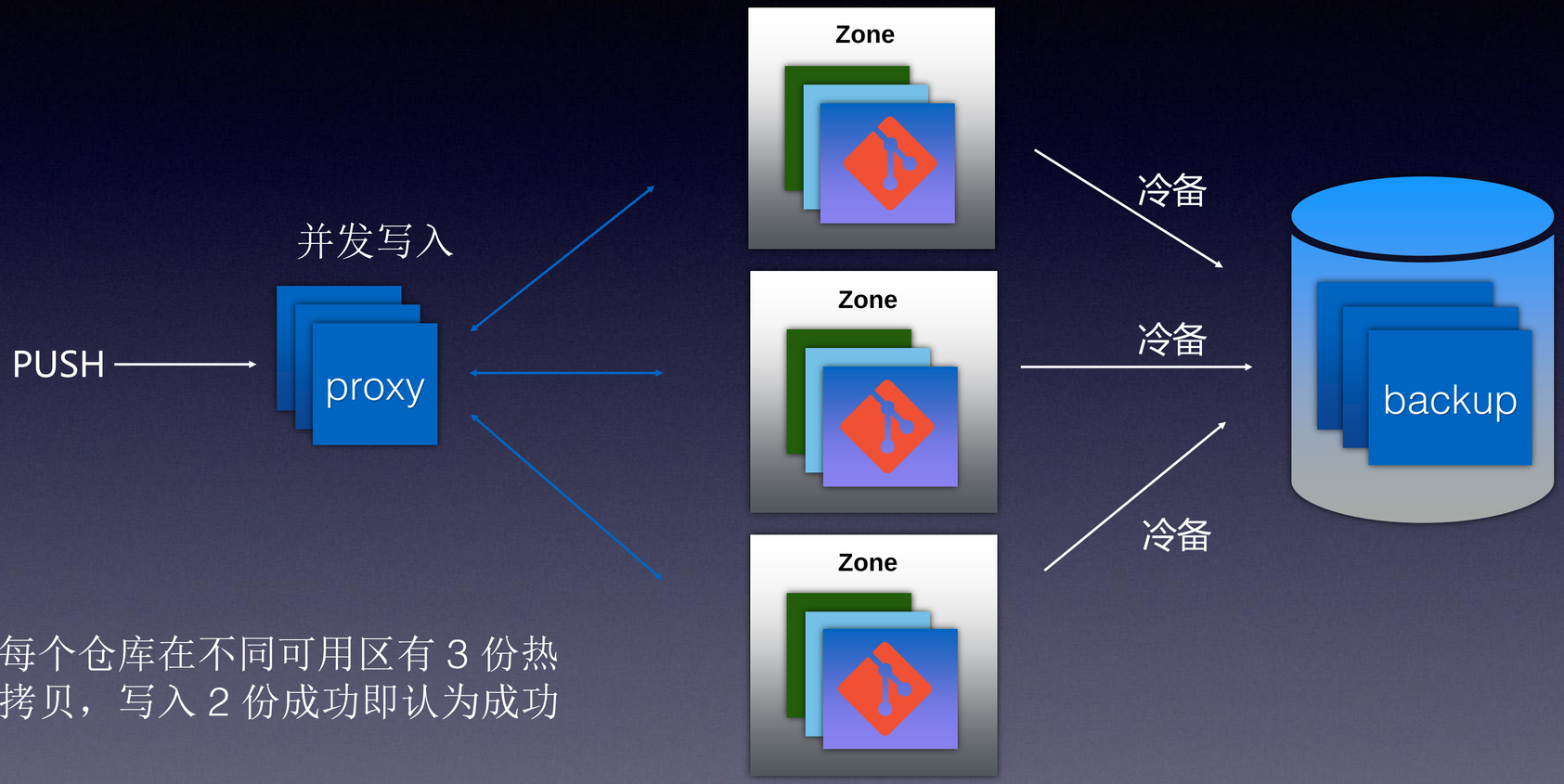


- 成本低，甚至可以不修改应用
- 性能指标无法满足 git 使用场景

git 对操作延时很敏感，一个简单的 git log 命令可能会需要遍历成百上千的对象（objects），每个遍历的延时变化加起来可能对整个命令的执行时间产生很大的影响，而且这个影响是无法预估的。

git 在设计上是为本地磁盘的访问而优化的，在操作本地磁盘时非常高效，我们要利用这一优势。

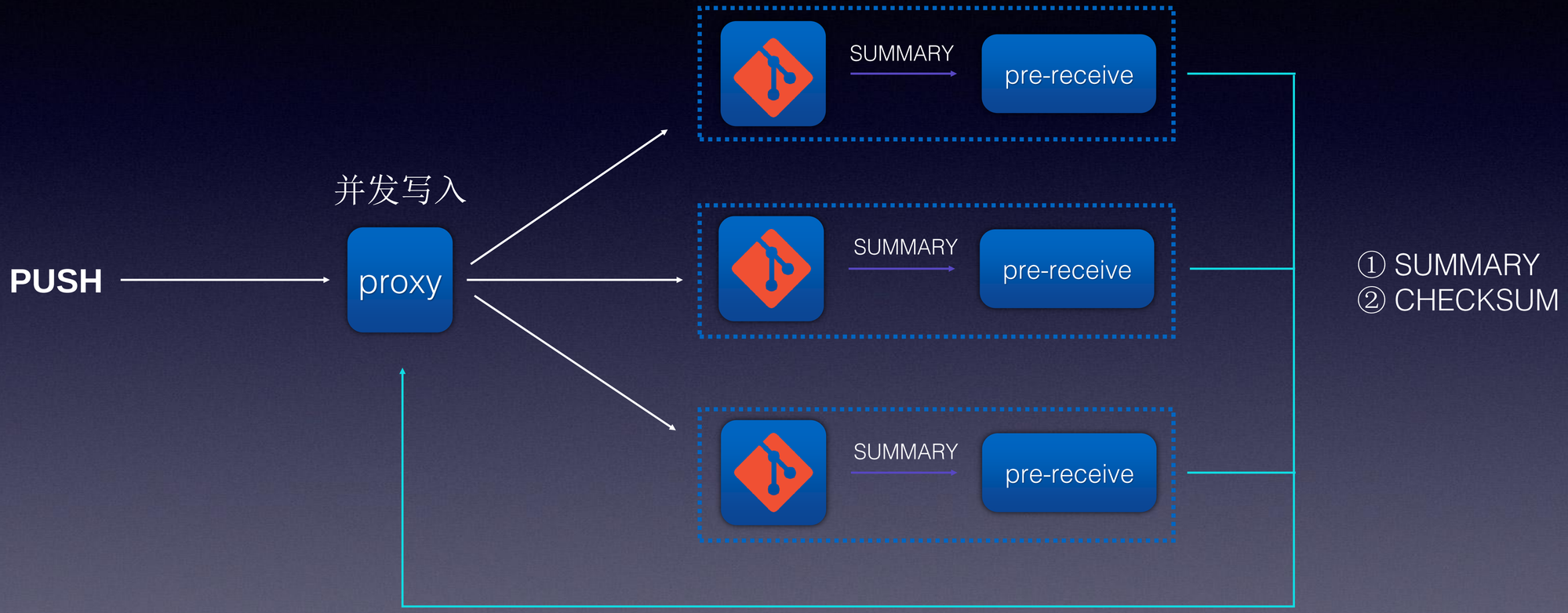
DGit 分布式系统



每个仓库在不同可用区有 3 份热拷贝，写入 2 份成功即认为成功

仓库存储在本地盘，无需通过网络访问

一致性保证



① 更新前：检查要更新的内容是否一致
② 更新后：对每个存储的更新结果进行校验

错误处理与恢复

- 自动屏蔽出错节点
- 自动恢复节点状态
- 最坏情况人工介入

优势

- 性能优秀
  - >充分利用本地高性能磁盘
  - >并发读取性能成倍提升（x3）
- 可靠性高
  - >多份热备 + 冷备，安全
  - >消除单点故障，可用性提升
- 运维效率提升
  - >仓库自动在线漂移
  - >最大化利用物理资源

性能对比

大仓库测试：2.58w commits, 12w+ objects, 386M

	git	dgit
git push	22.02s	30.64s
git clone	15.9s	15.3s
get tree	1.14s	1.09s
edit file	688ms	823ms

结果：写入性能有所下降，但是在可接受范围内，读取性能基本没有影响

Thank You

Coding
让开发更简单

代码托管 · 项目管理 · WebIDE