副本一致性对大数据系统性能测试的影响

清华大学软件学院
王建民

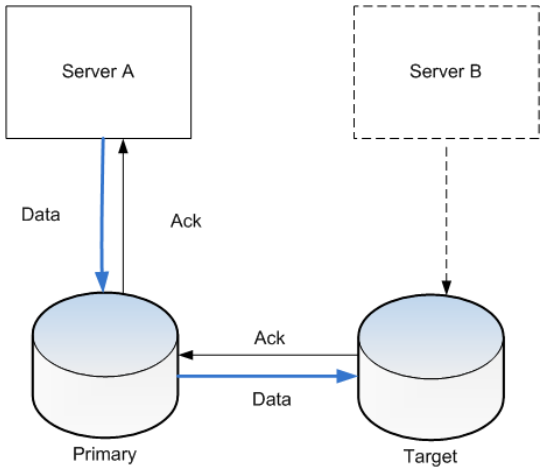
2015.4.28

数据副本

data replica-
the same data is stored on multiple storage devices

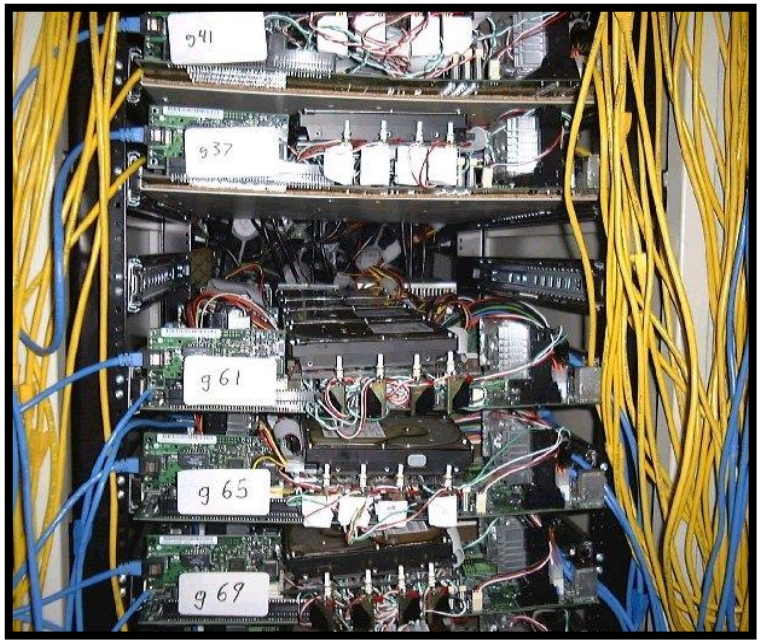
to guarantee

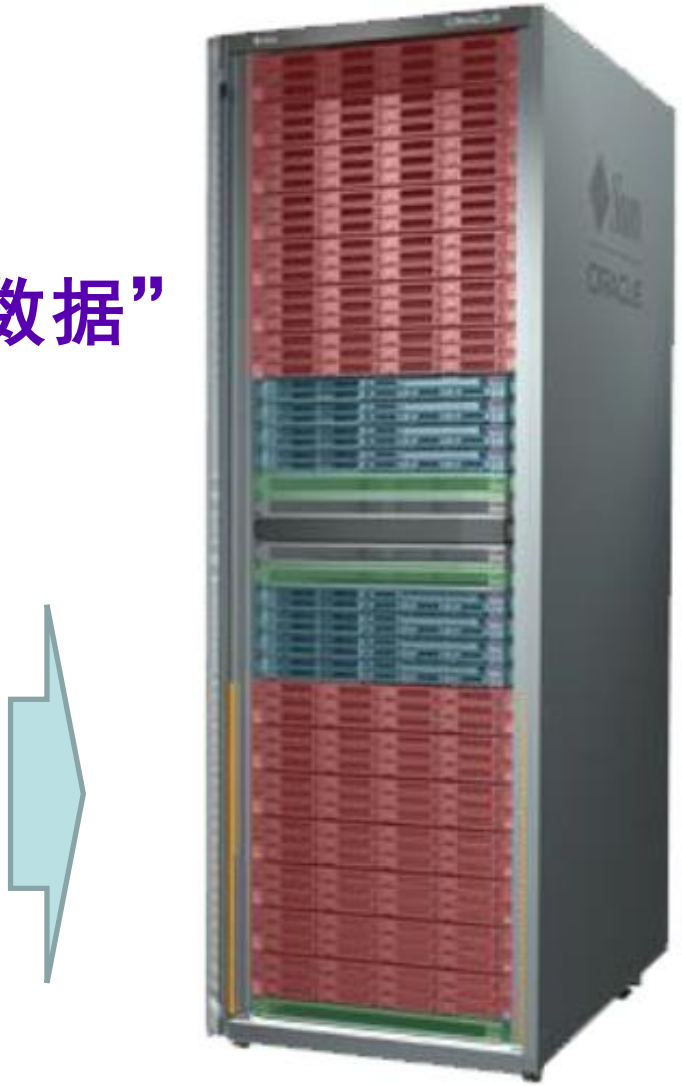
- the data reliability,
- fault-tolerance,
- accessibility for users



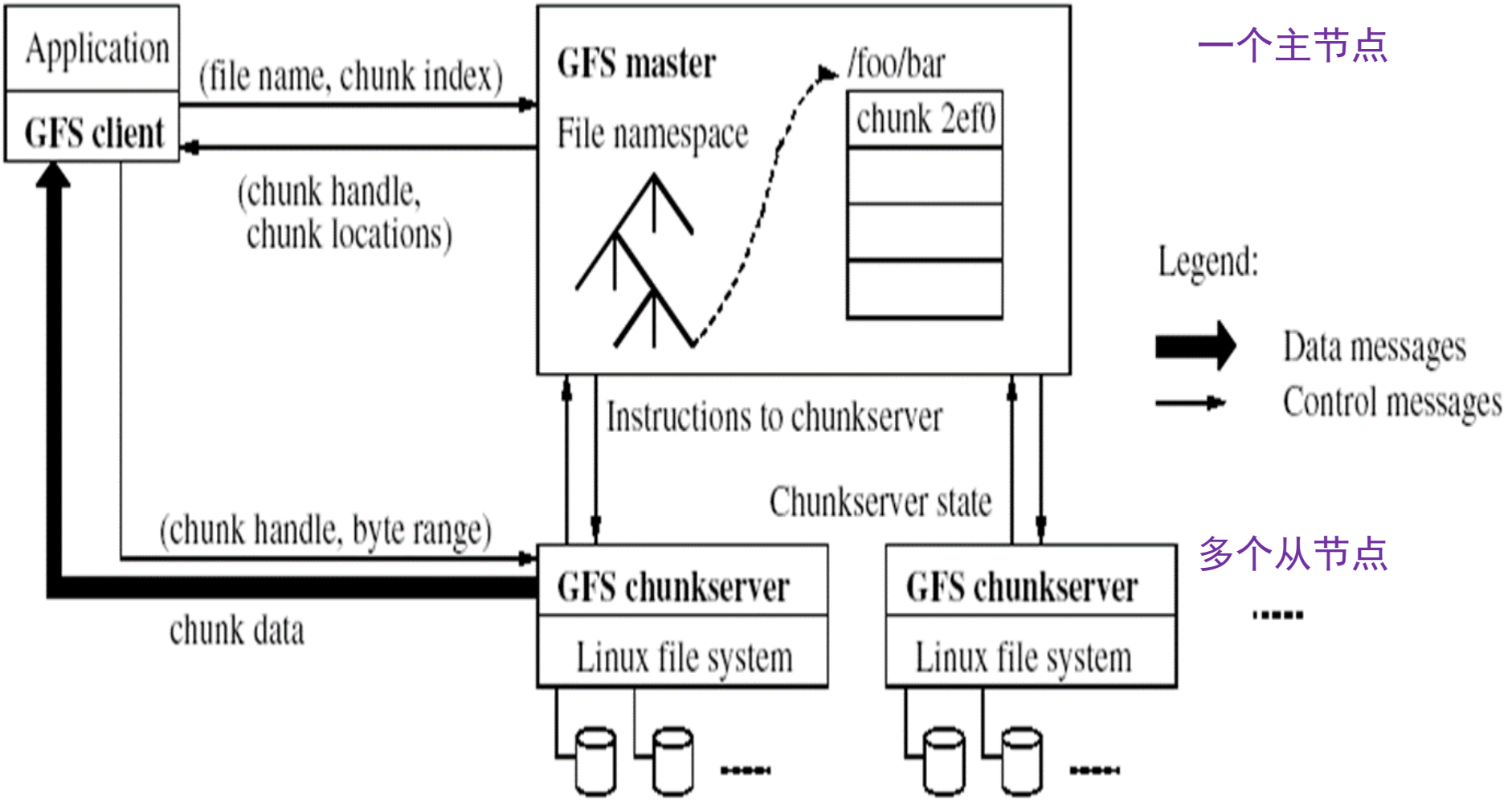
1：GFS - Google文件系统

- “小机器”拼成“大系统”
- “差机器”变成“好系统”
- 为“并行计算”准备“分布数据”



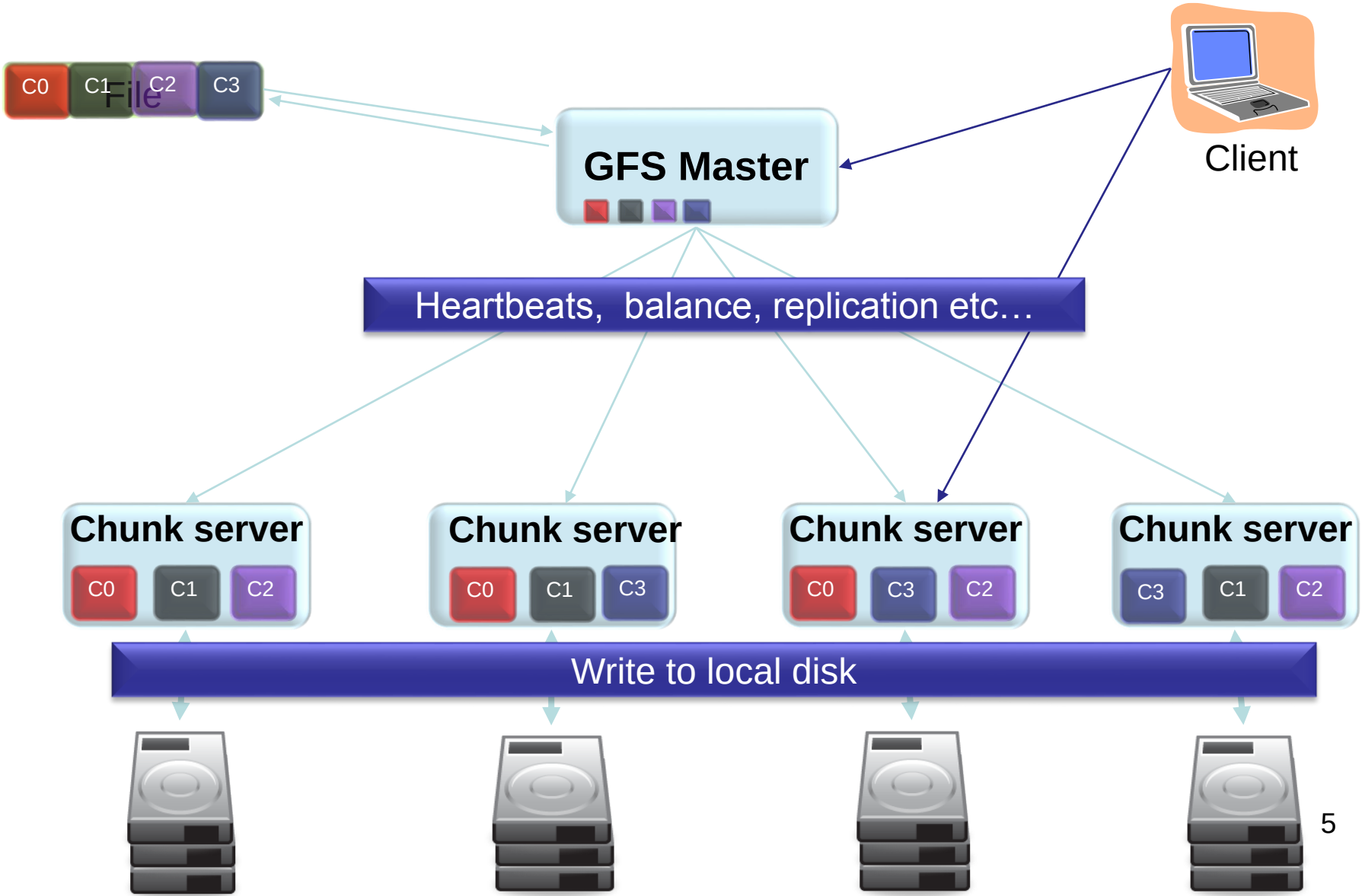


GFS采用主从架构

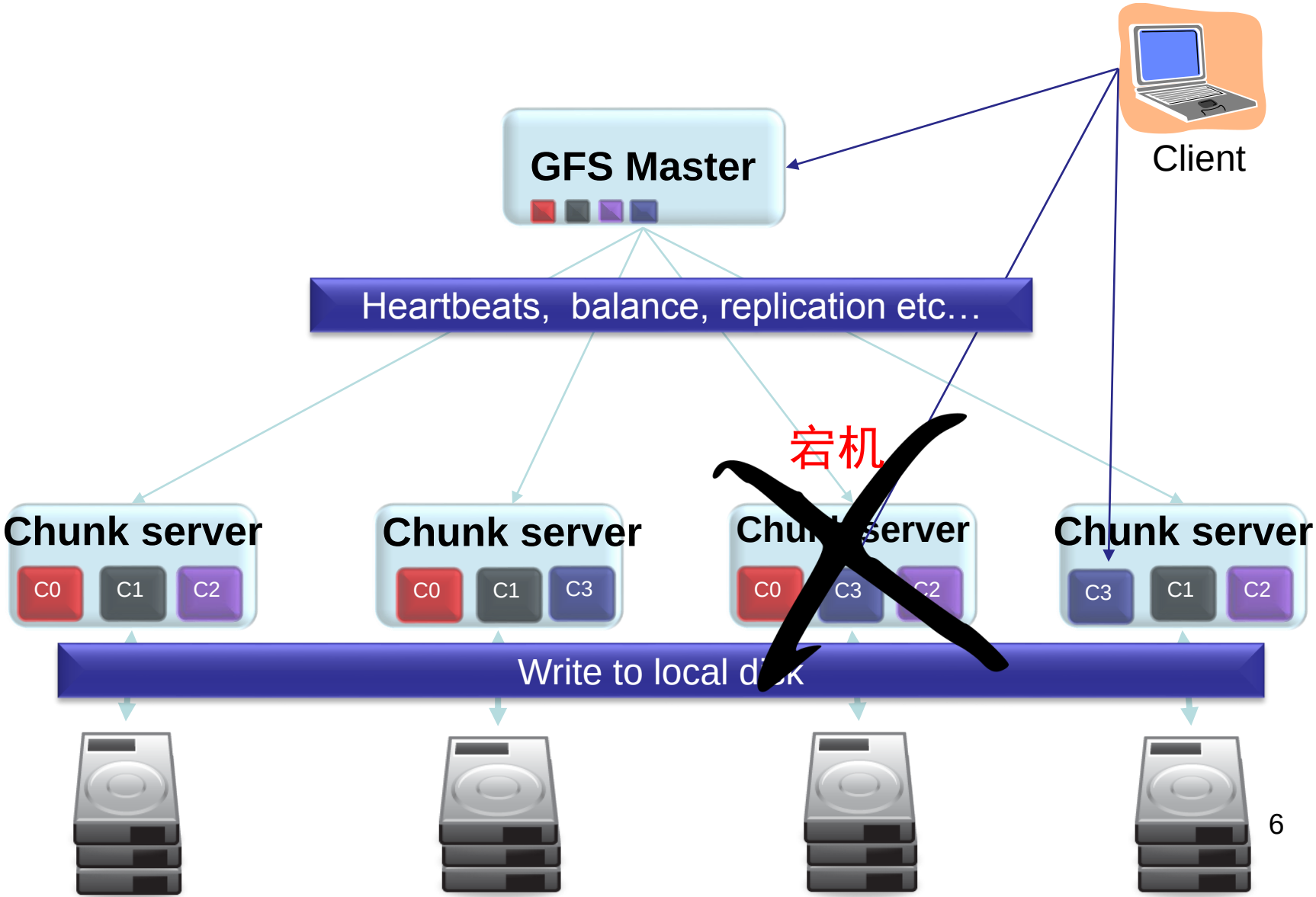


Ghemawat, S.; Gobioff, H.; Leung, S. T. "The Google file system". SOSP 2003

GFS文件的分片与副本



GFS副本：“读”的收益


Client
GFS Master
Heartbeats, balance, replication etc…
宕机
Chunk server
C0
C1
C2
Chunk server
C0
C1
C3
Chunk server
C0
C3
C2
Chunk server
C3
C1
C2
Write to local disk

GFS副本：“写”的代价

原则：

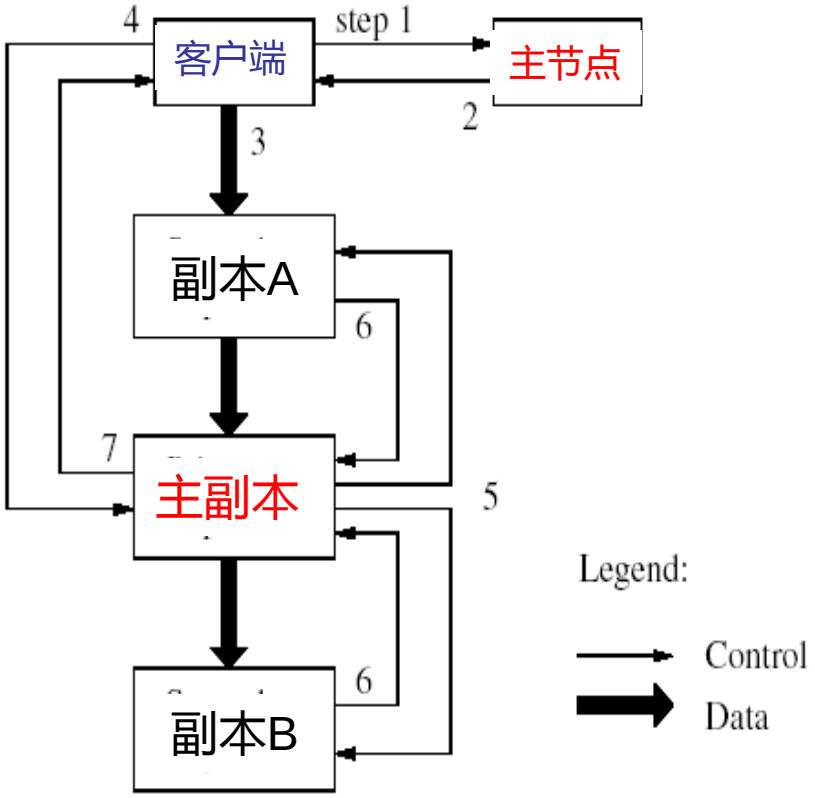
“写”必须覆盖所有副本

主(Master)节点负载最小化

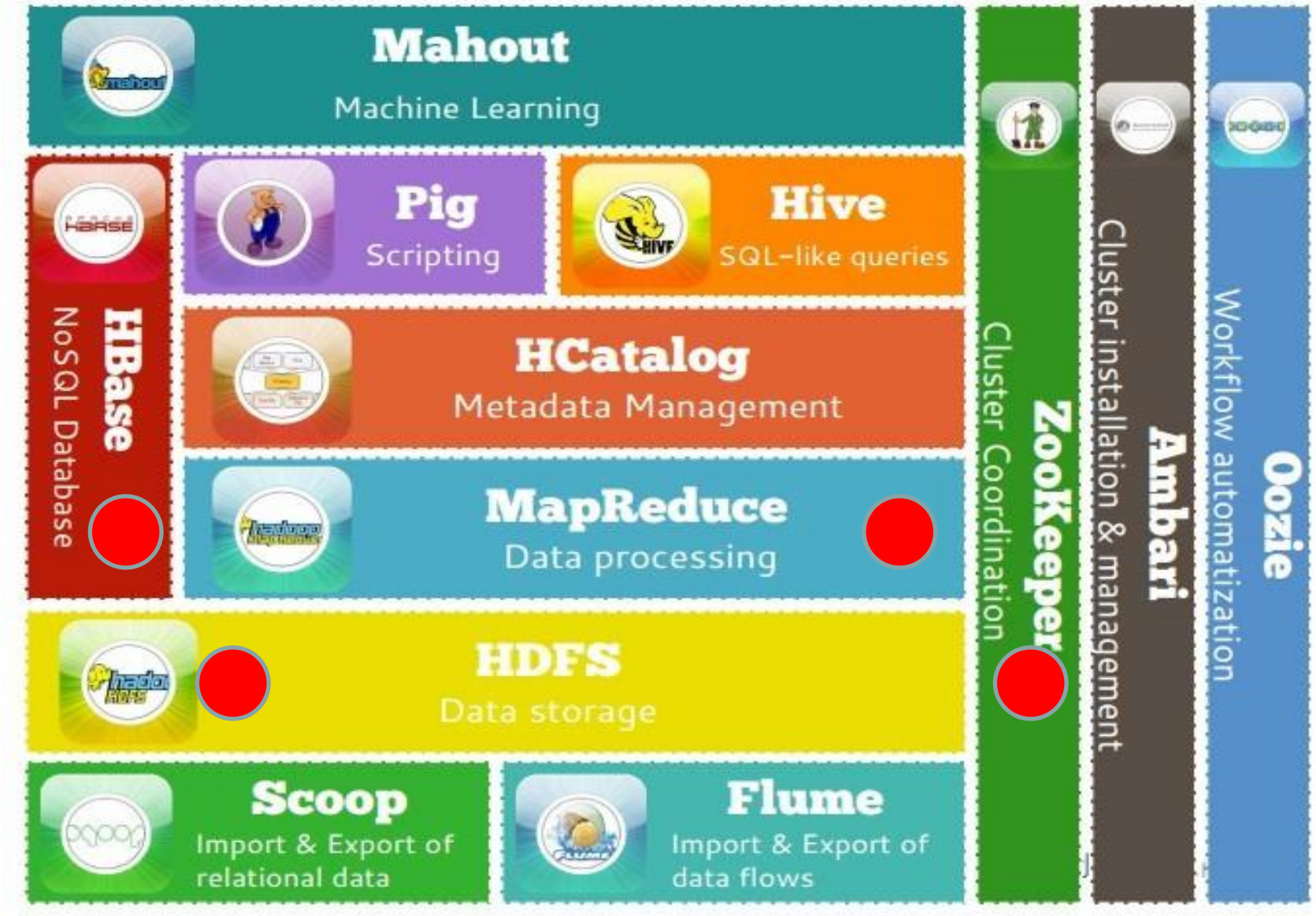
机制：

- 主节点指定主副本
- 主副本获得“写”锁
- 主副本确定“传播链”
- 所有副本遵循同一顺序

数据流和控制流解耦



Hadoop生态系统


Mahout
Machine Learning
Pig
Scripting
Hive
SQL-like queries
HBase
NoSQL Database
HCatalog
Metadata Management
MapReduce
Data processing
HDFS
Data storage
Scoop
Import & Export of relational data
Flume
Import & Export of data flows
ZooKeeper
Cluster Coordination
Ambari
Cluster installation & management
Oozie
Workflow automatization

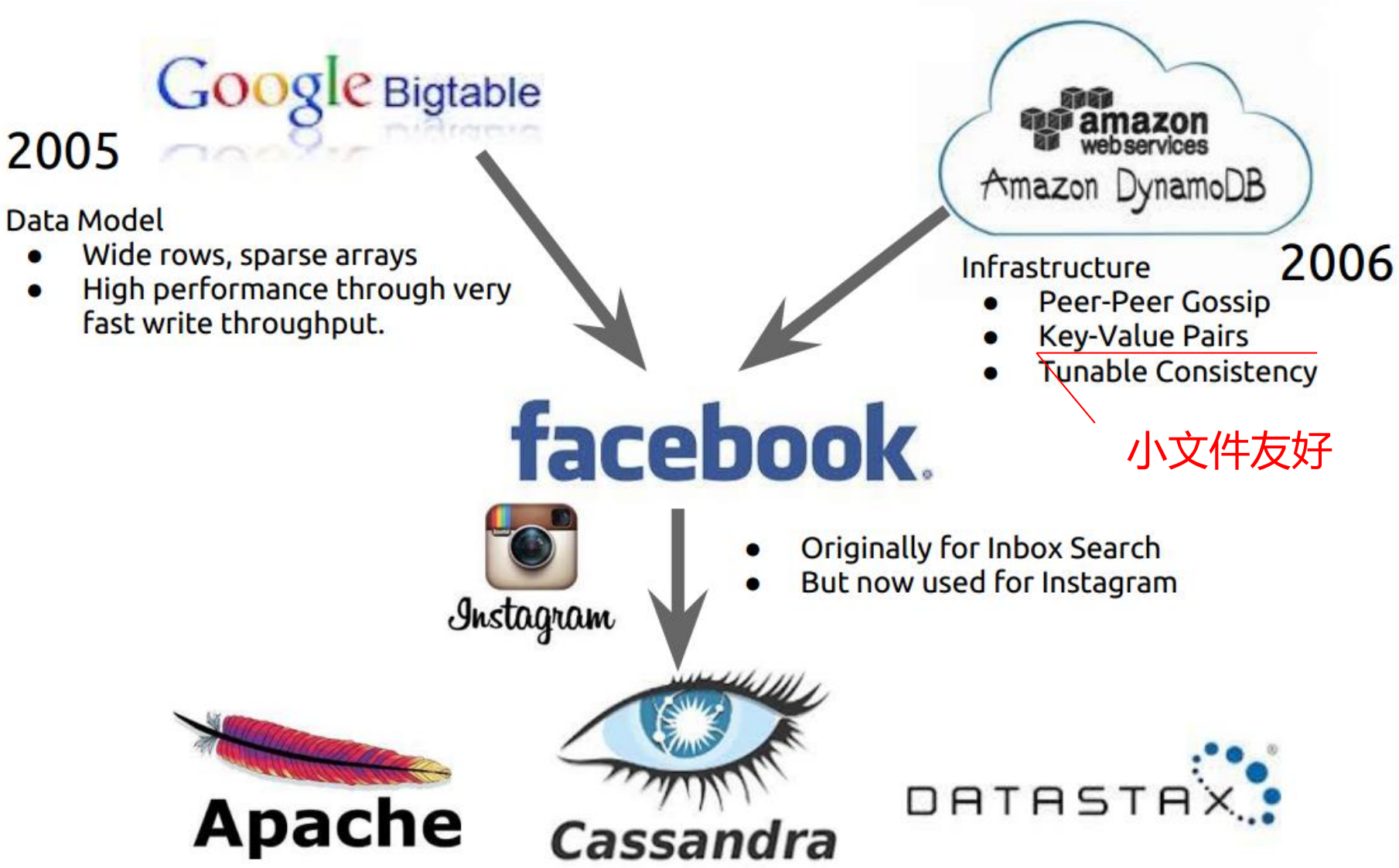
You Say, “tomato…”

Google calls it:	**Hadoop** equivalent:
MapReduce	Hadoop
GFS	HDFS
Bigtable	HBase
Chubby	Zookeeper

Google/Hadoop的技术弱点

- 主从架构
- 单点故障
- 系统弹性
- 大文件友好
- ……

2：Cassandra=Bigtable+P2P



一致性哈希 - 改善系统弹性

口原理 –把数据对象的Key映射到一个足够大的环上

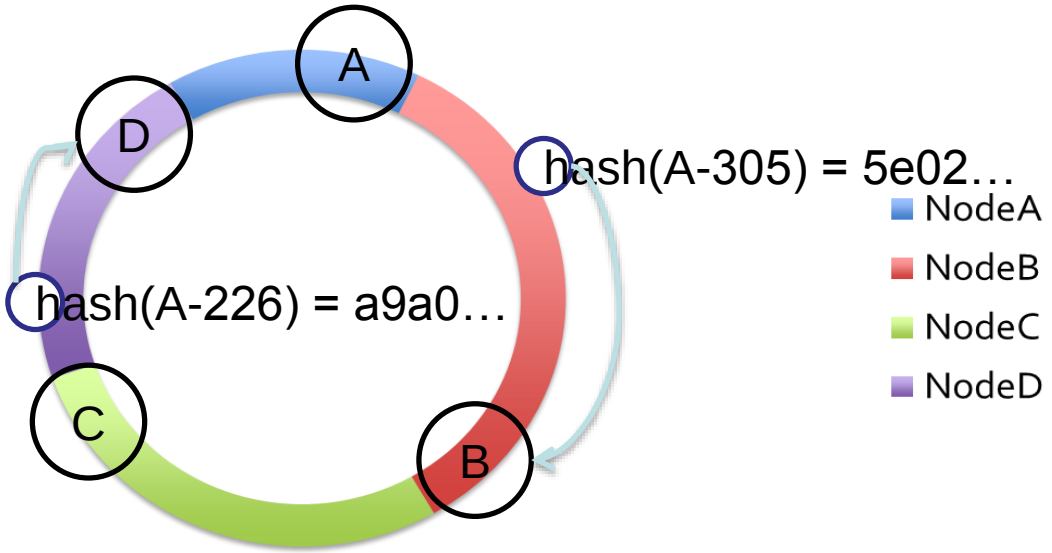
同时把每个机器也映射到环上的一个点

每个机器负责环上的一段数据

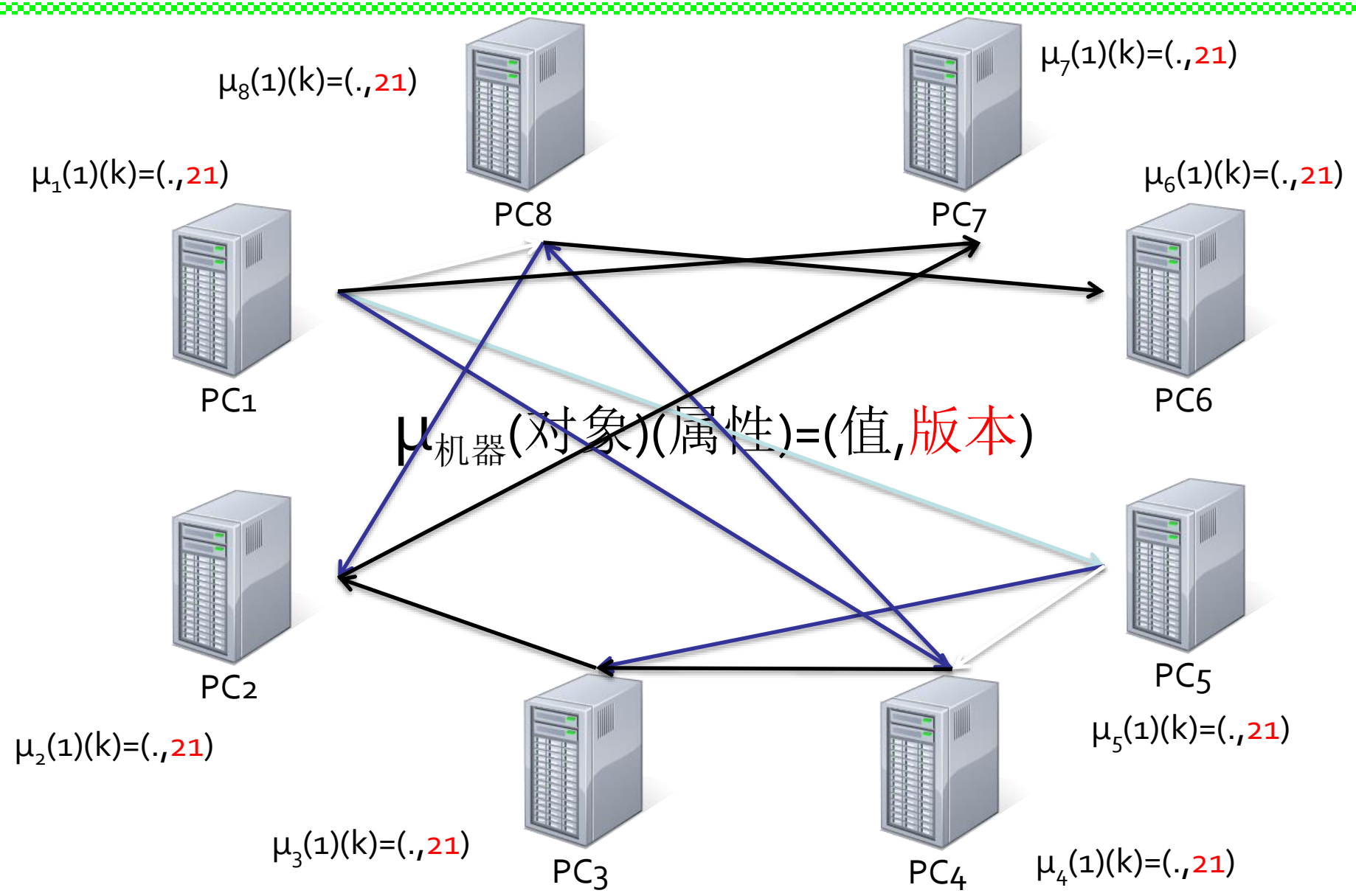
口优点 – 系统弹性好

机器进入或离开时只影响有限个数据段

机器	开始	结束
A	e590…c	12a3…e
B	12a3…f	7310…3
C	7310…4	a331…1
D	a331…2	e590…b



P2P Gossip 协议 – 副本同步



工业大数据系统性能测试

- 数据模型
- 负载模型
- 副本一致性参数配置

数据模型

- 数据模式
  - 构造时模式定制
  - 运行时模式演化
  - 多版本模式并存
- 数据特征
  - 工况类型特征

（开关量、离散值、连续值、… …）

  - 时序参数特征

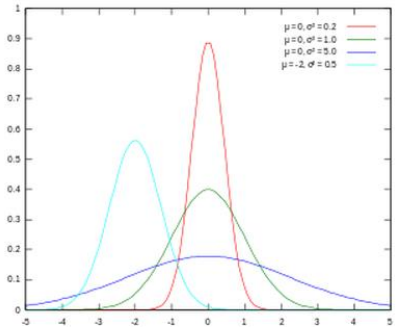
（采样时间、采样频率、精确度、… …）

  - 数据分布规律

（地域、时间、产品、工况、… …）

年份	工况种类
2008	1560
2009	3545
2010	3665
2011	4970
2012	5363

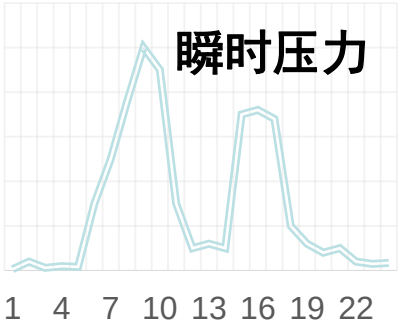
工况种类变化

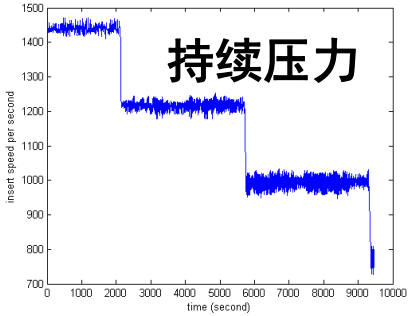


$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$?

负载模型

- 瞬时压力与持续压力并存
  - 瞬时压力模型
  - 持续压力模型





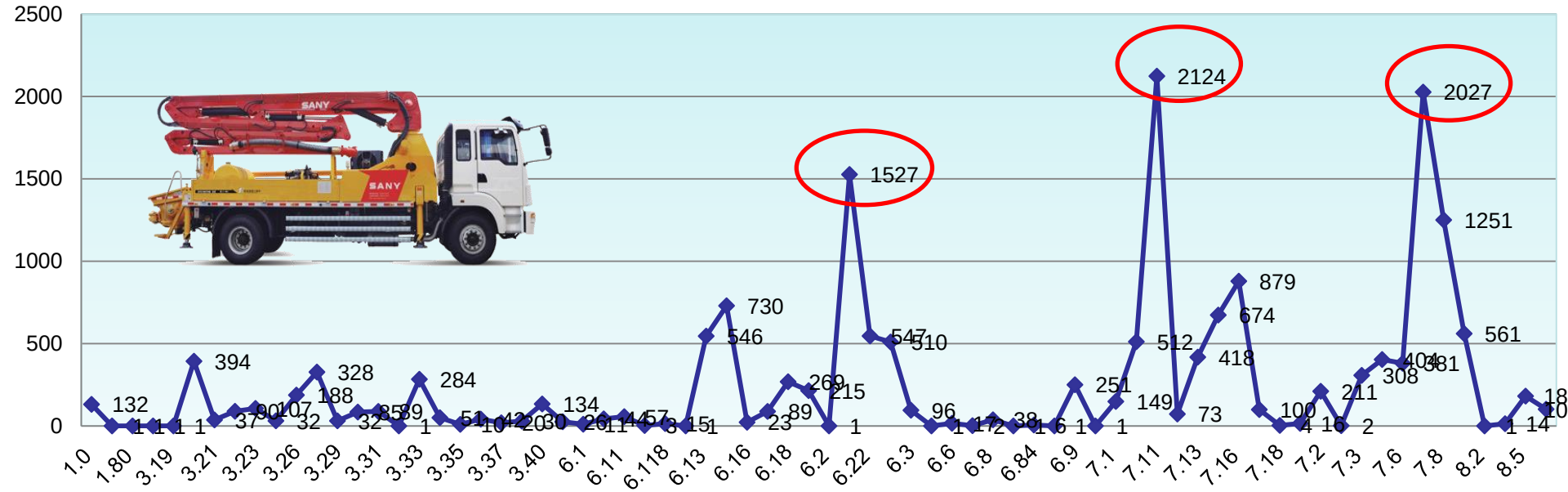
- OLTP和OLAP操作并存
  - 突发工况更新
  - 全部数据Online
  - 时序模式匹配
  - 交互统计分析

编号	参数名称	参数说明
1	在线数据量D	单位为天，即D天的工况数据
2	主机编号E	随机选取
3	工况编号S	随机选取
4	时间段T	可以人为设定也可以随机选取
5	重复次数N	人为设定
6	并发度P	人为设定

模式动态多变、多版本共存

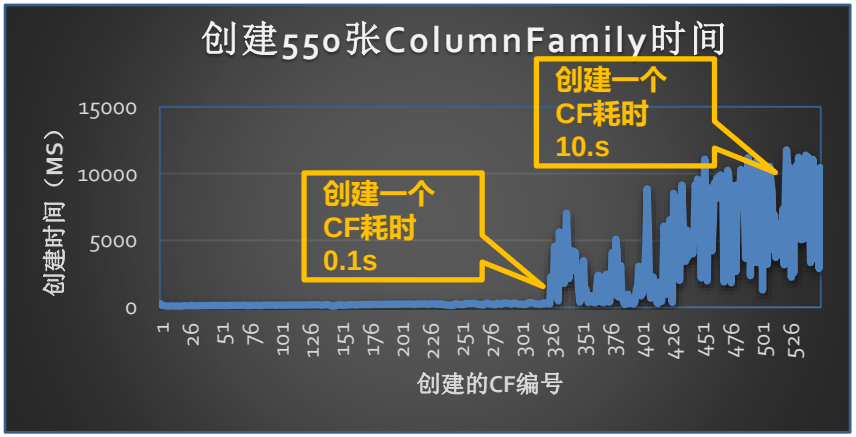
以泵车控制系统为例

- 从2008年后升级到控制器版本1.0，共有17511辆泵车
- 目前主控程序有72个版本，其中6.21版(1527台)、7.11版(2214台)和7.8版(2027台)
- 每个版本都意味着工况数量的变化，**平均240种工况**

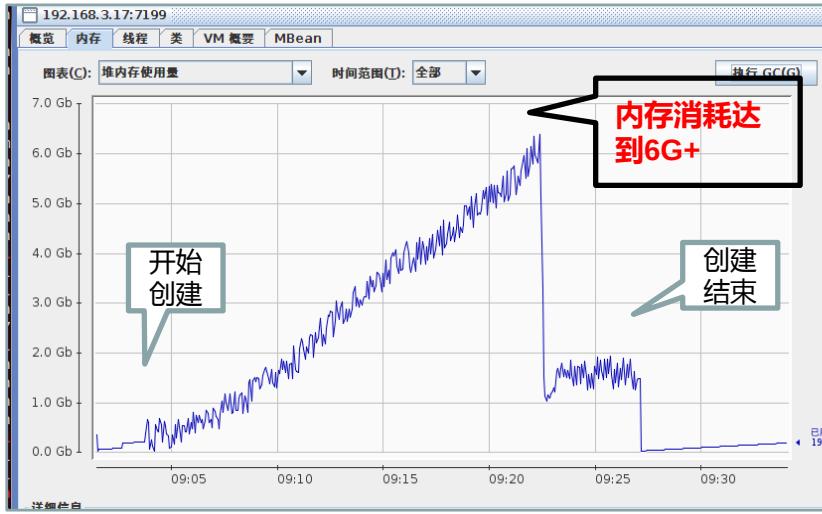


副本一致性参数配置

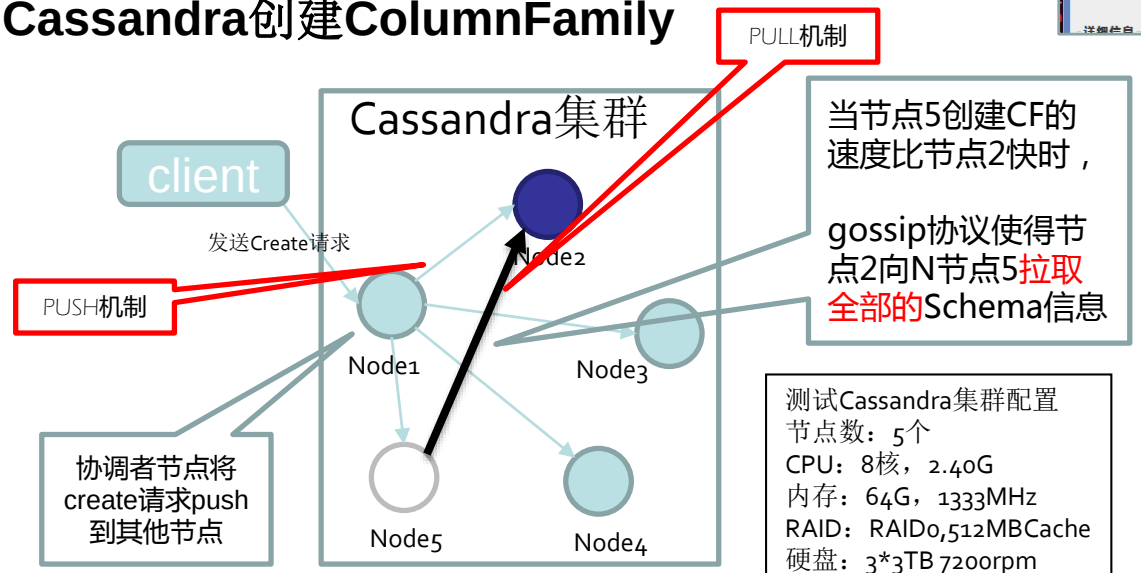
创建CF耗时急剧上升



创建CF内存消耗剧烈



Cassandra创建ColumnFamily



当节点5创建CF的速度比节点2快时，

gossip协议使得节点2向N节点5拉取全部的Schema信息

测试Cassandra集群配置
节点数：5个
CPU：8核，2.40G
内存：64G，1333MHz
RAID：RAID0,512MBCache
硬盘：3*3TB 7200rpm

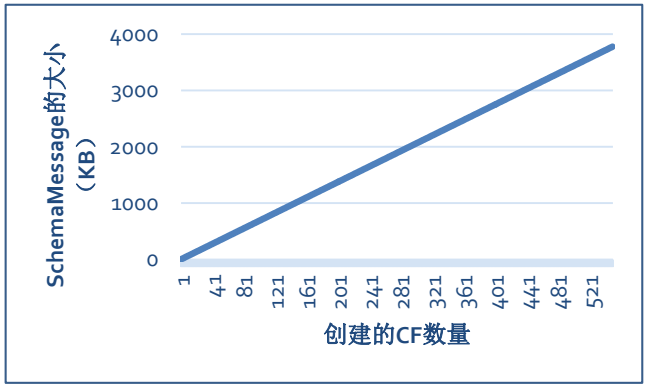
问题分析：
1、建表过程中，各节点的Schema状态出现短暂的不一致；
2、Gossip发现节点间Schema不一致进而不断触发节点间的Schema传输；
3、导致节点间不断发送大量不必要的信息。

Gossip协议带来的内存和速度损耗

节点收发Gossip次数

	SendSchema	ReceiveSchema	Total
Pc1	1326	1339	2665
Pc2	1399	1353	2752
Pc3	1371	1350	2721
Pc4	1570	1551	3121
Pc5	2162	2356	4518

Gossip消息大小的变化



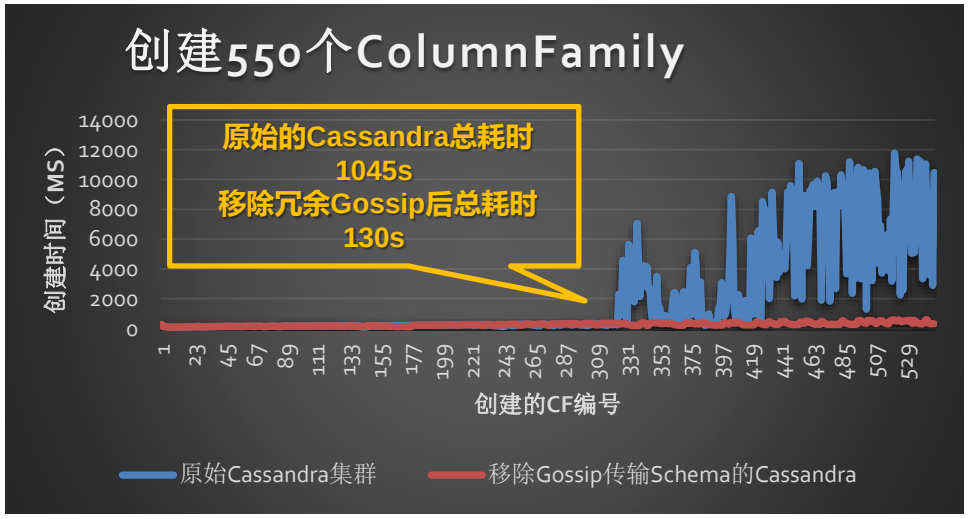
Gossip引发的内存消耗

	ReceiveSchema Message Memory Cost	SendSchema Message Memory Cost	Total
Pc1	4.465G	4.236G	8.70G
Pc2	4.308G	4.907G	9.21G
Pc3	4.236G	4.024G	8.26G
Pc4	4.808G	4.387G	9.19G
pc5	6.111G	6.373G	12.48G

创建CF过程中Schema传输严重影响了系统的性能。

由于PUSH机制的存在，在没有异常发生的情况下，各节点Schema最终是一致的。

在创建CF过程中的PULL机制是多余的



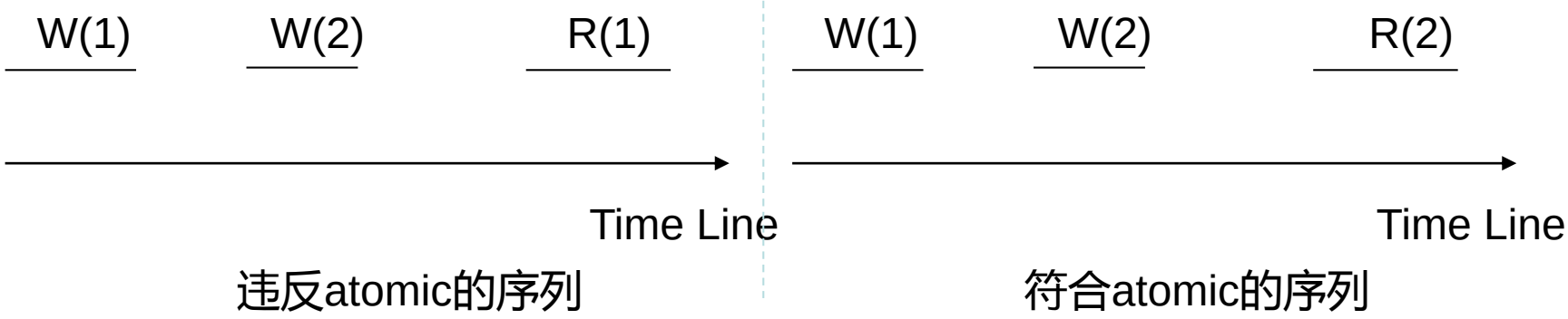
副本一致性的度量

- Data staleness metric
  - T-visibility & K-version
    - The probability of reading the write value at most *t* time-units after the write operation finished
    - The probability of reading at most *k* versions older than the latest one
- Operations order metric
  - Safety, regularity and atomicity
    - Check the percentage of disordered data operations
  - Consistency Anomalies
    - Check the percentage of the system schedules satisfy the causal consistency constraint

一致性度量: Δ-atomicity

- Δ-atomicity

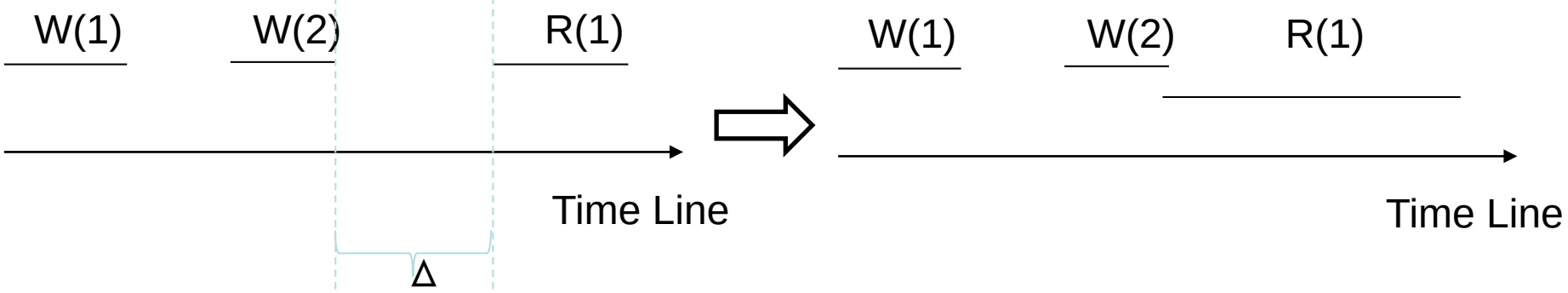
  atomic ：一个读写序列满足atomic，当且仅当所有的读都读到最新的写



一致性度量:Δ-atomicity

- Δ-atomicity

一个读写序列S满足Δ-atomicity，当且仅当将这条序列上的所有读操作的开始时间减小Δ形成新序列S_{Δ}，S_{Δ}这个序列是atomic的



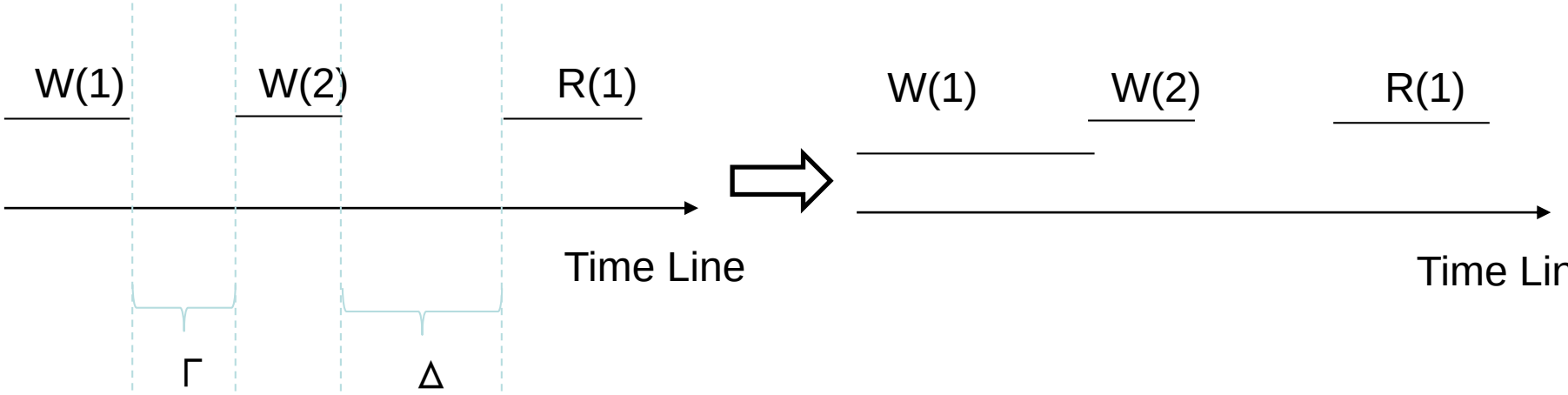
Δ-atomicity说明了一条序列违反atomic的程度；并反映了写的生效时间

一致性度量:Γ（Gamma）

- Γ（Gamma）

  减小Δ-atomicity的噪音

  ●不同客户端时钟倾斜



结论

- 副本一致性影响系统性能，有时甚至是十分显著的
- 在相同/相当副本一致性水平下，比较系统性能才公平
- 副本一致性度量方法还在路上

谢谢您的聆听