架构设计的一些思考

@林仕鼎
CSDN-TUP
2012/2/18

先从例子说起...

- 存储
- 分布式
- 服务架构
- 计算模型

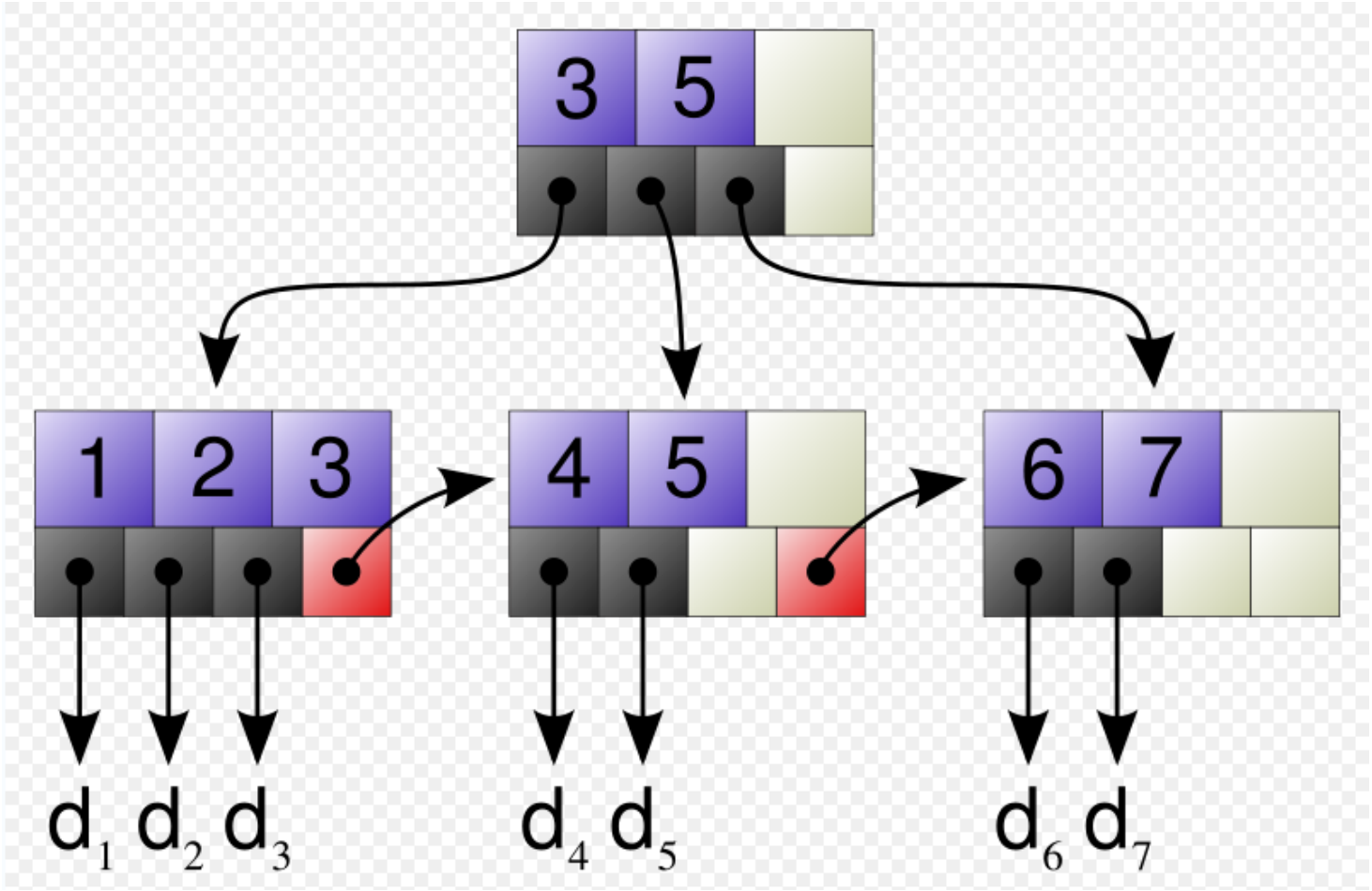
存储

- 结构
  - File
  - Object
  - Table
- 数据特点
  - mutable or not
  - size
  - data layout
- 访问模式
  - 实时读写
  - 批量写、实时读
  - 流式读
  - Scan / Range Query
- “实时性”
  - Realtimeness
  - Freshness
  - Consistency

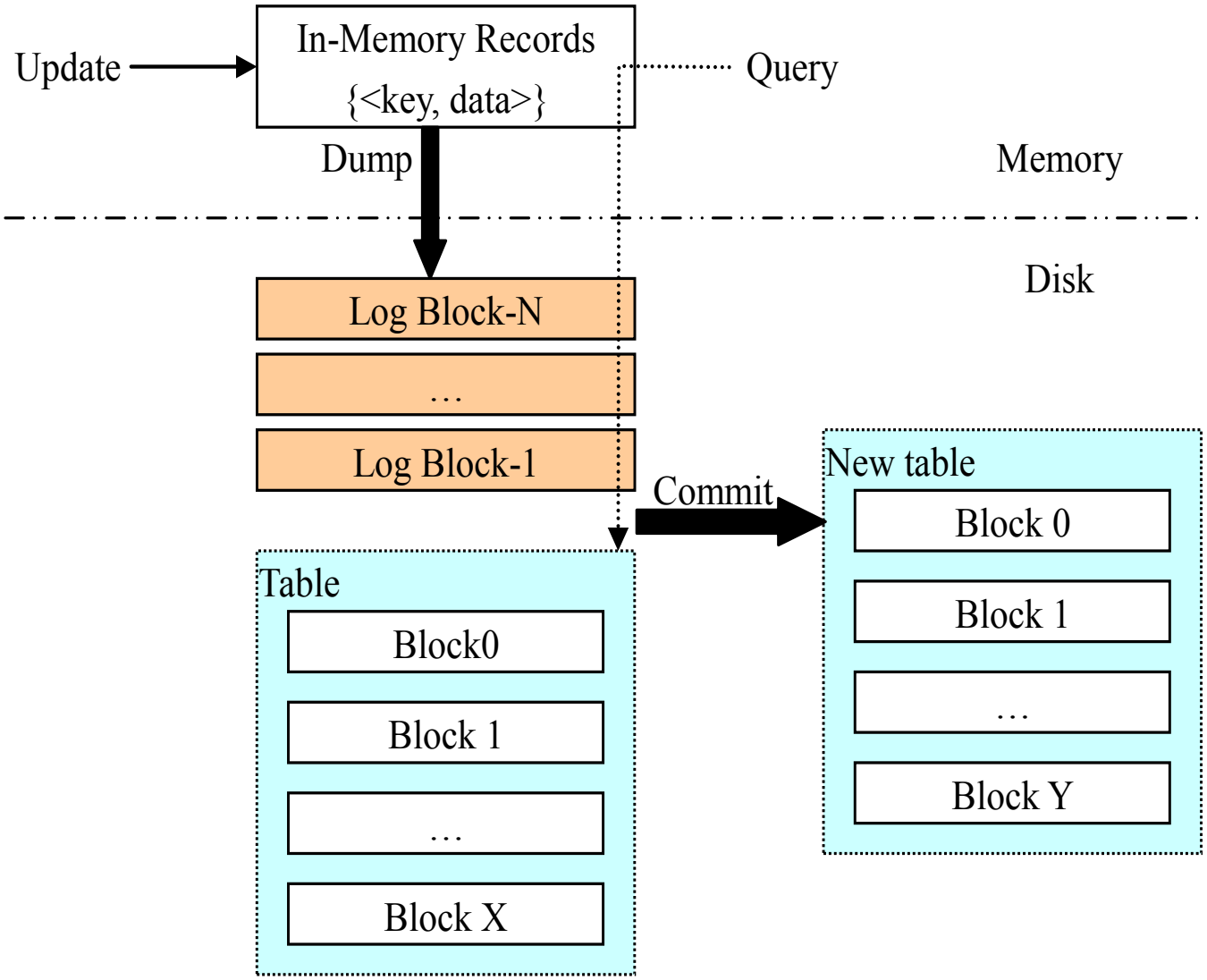
存储

- 矛盾
  - 延迟与吞吐
  - 随机与顺序
  - 规模与实时性
- 模型
  - B+ tree （实时、随机）
  - Log-based （批量、顺序）
- 化解矛盾
  - 弱化需求
  - 发掘局部性
  - 组合模型

存储模型：B+ tree

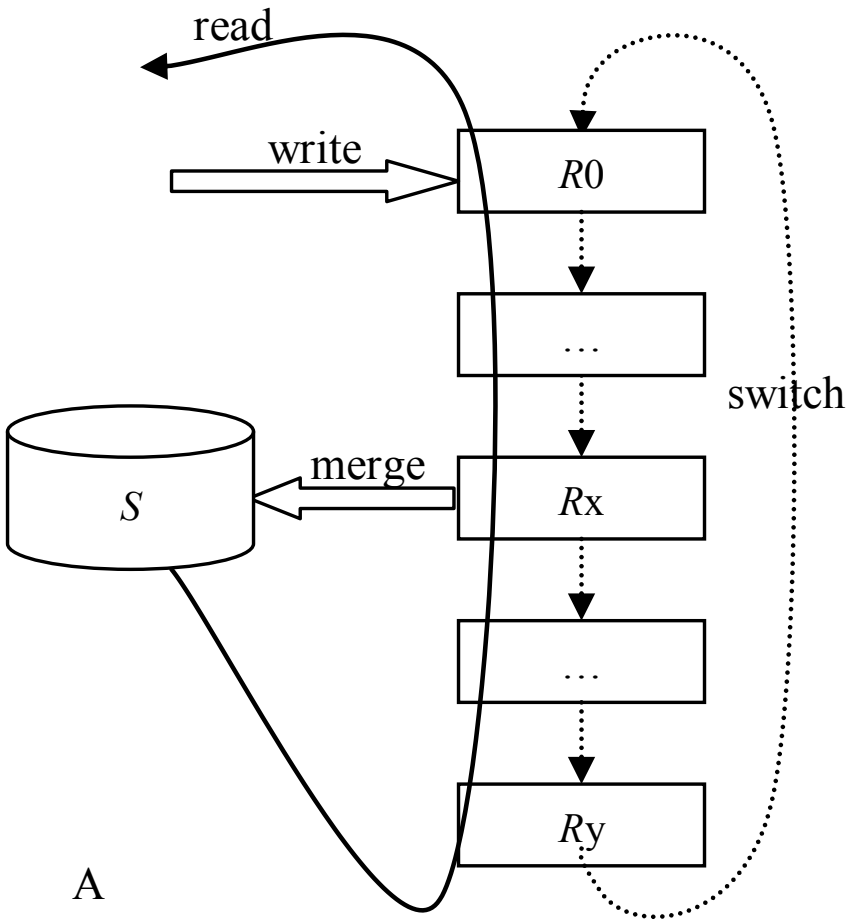


存储模型：Log-based structure

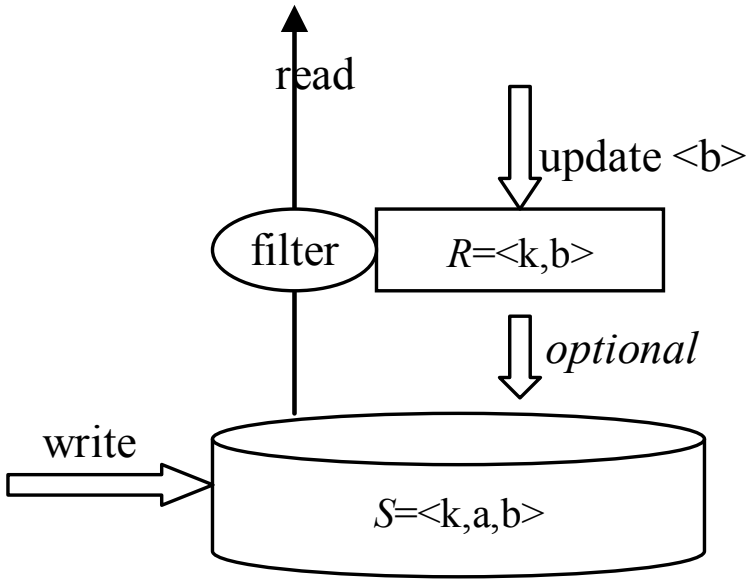


存储：组合模型

S: 顺序性能好，容量大
R: 随机性能好，容量小



A

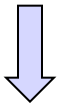


B

分布式

- 目标
  - 扩容：scalability
  - 容错：availability
- 方法
  - Partition
  - Replication
- 难点
  - 协议设计
  - 调试

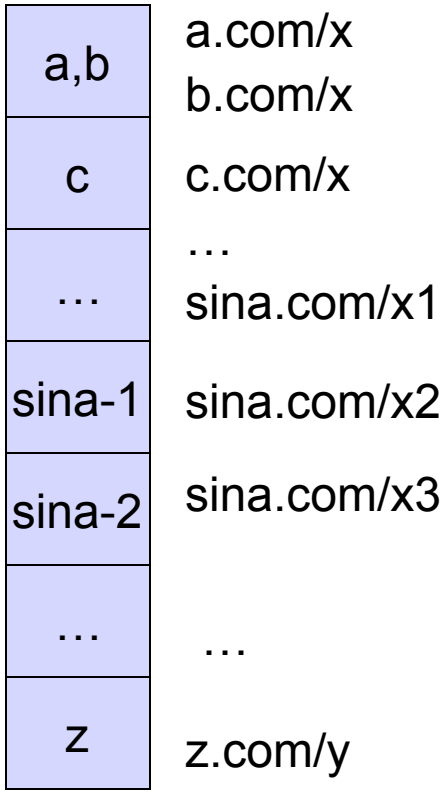
$$P = p^k$$



$$P = 1 - (1-p)^k$$

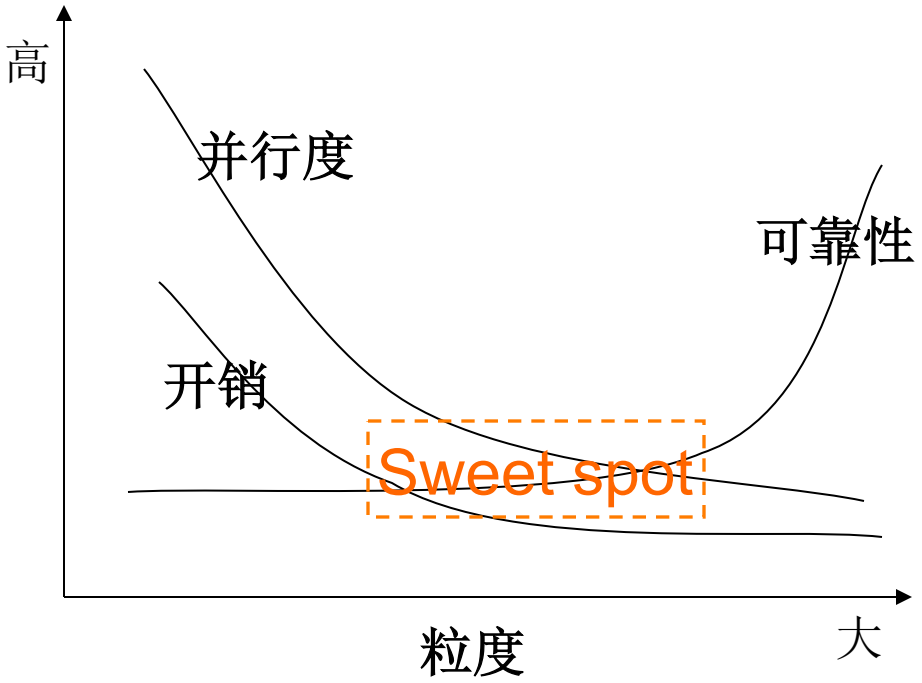
分布式：Partition

- Static hashing
  - 无法调整
- Consistent hashing
  - 影响K/n数据
- Mapping
  - 拆分-合并

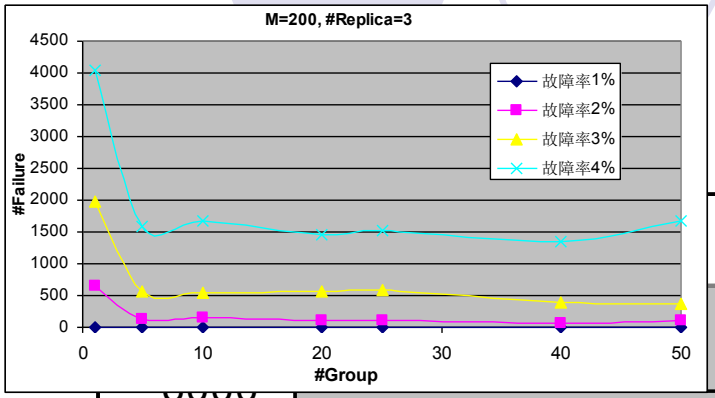


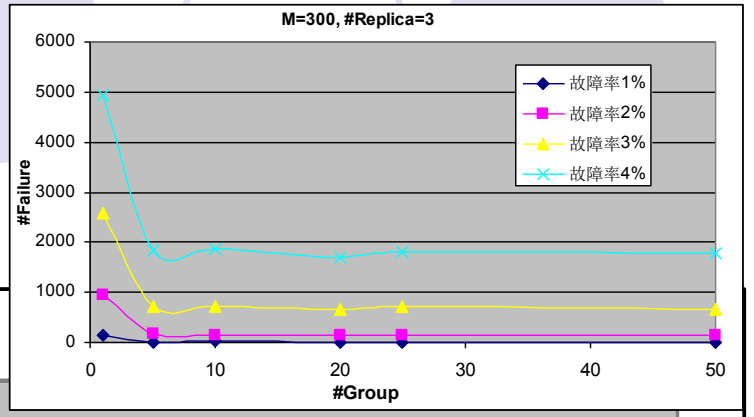
分布式：Replication

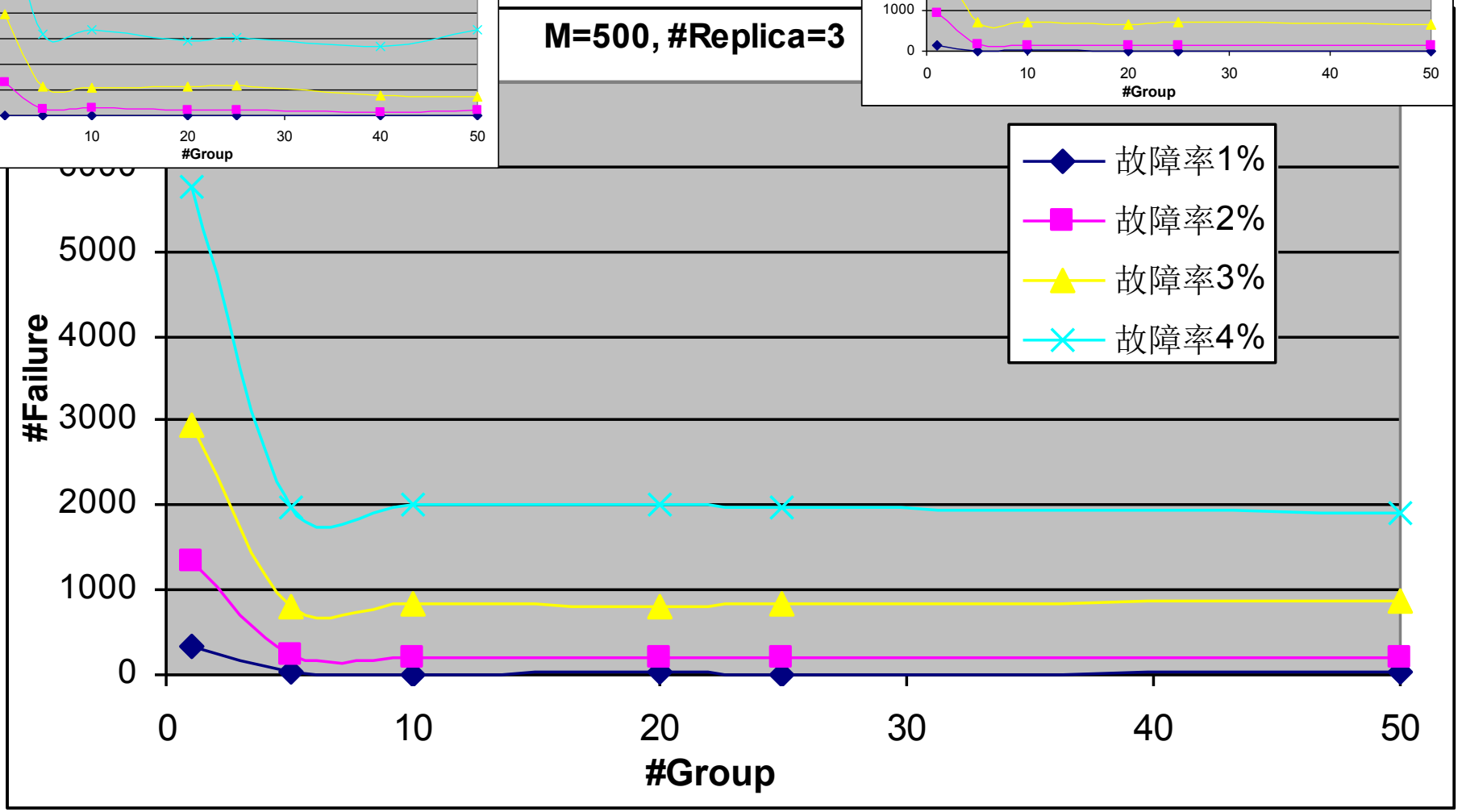
- 粒度
  - Machine
  - Record
  - **Group**



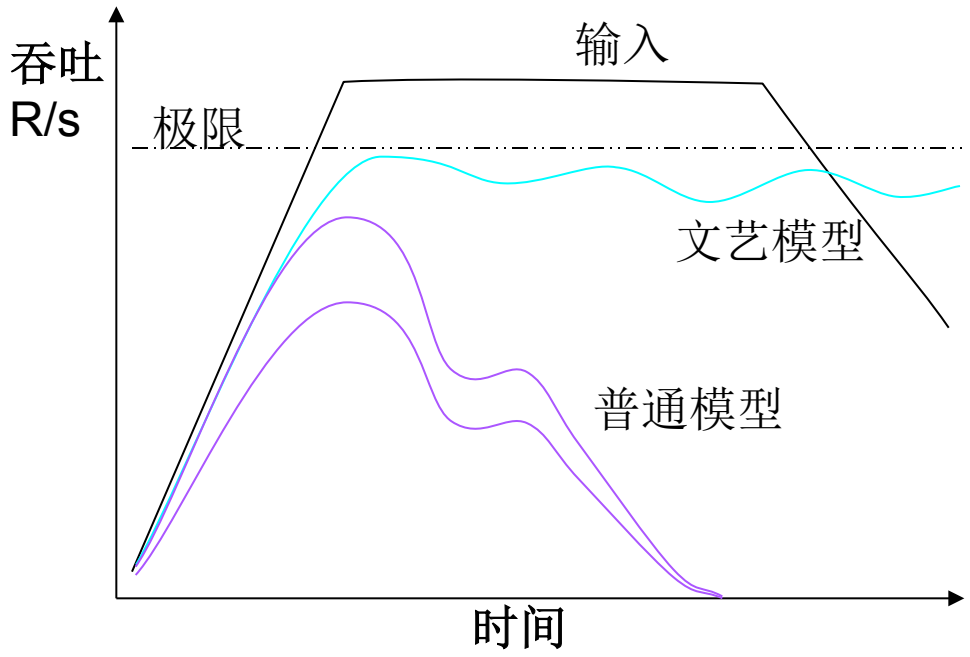
Replication不是万能的







服务架构



服务架构

- 目标
  - 高吞吐
  - 极限压力下稳定输出
- 模型
  - 基本：threadpool + queue
  - 复杂：event-driven
- 稳定性保证
  - 减小资源分配粒度，主动调度
  - Flow Control
    - 负载反馈，Throttling
    - 延迟截断，分级队列

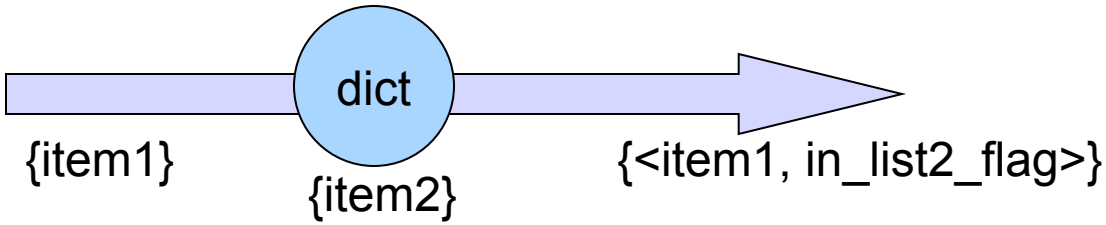
计算

- 数据密集型
  - MapReduce
  - Scan-Filter
- 计算密集型
  - seti@home
- 通讯密集型（传统HPC）
  - 机器学习系统
  - 矩阵计算

Scan-Filter

- 例子：两个list求交集
- Input: list1 + list2，|list1|>>|list2|
- Output: {<item1, in_list2_flag>}

- MapReduce
  - Sort + Partition + Reduce
- Scan-Filter



怎么做存储系统？
怎么做高性能服务？
怎么做数据仓库？

架构是什么？
架构师需要什么？

架构师三板斧

- **看清需求**
  - Tradeoff
    - 无法满足所有需求
    - 无须同等对待所有需求
  - 发现根本需求
    - 分解、抽象、降维
    - 定义primitives和组合规则
  - 了解需求随时间的变化
- 选择方法
- 把握节奏

对不合理需求 **say NO!**
但给他**end-to-end**解决方案。

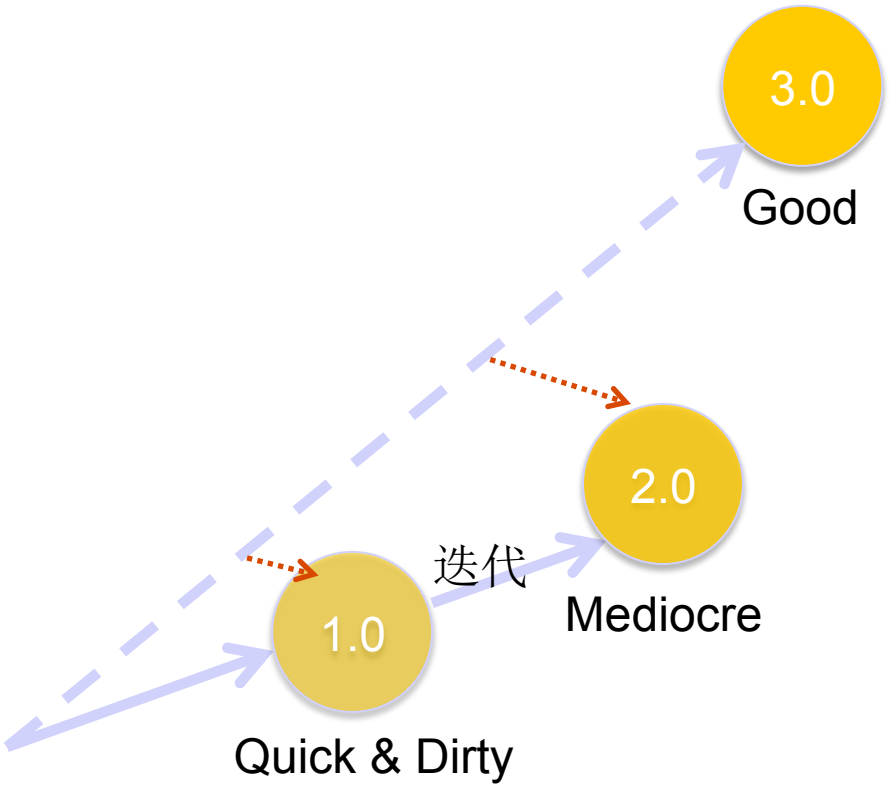
架构师三板斧

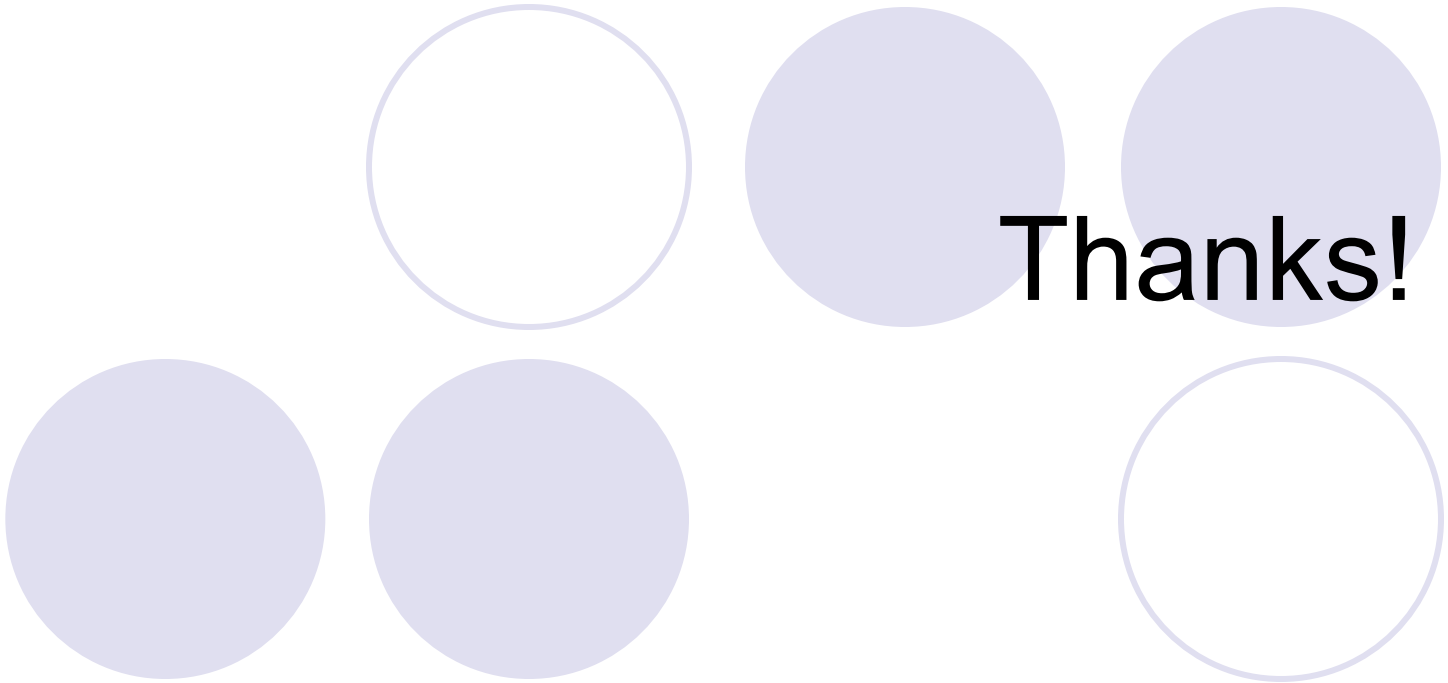
- 看清需求
- **选择方法**
  - 测算 -> 模拟 -> 实现
  - 分解 vs 迭代
  - 设计模式
- 把握节奏

Back-of-Envelop Calculation
Monte-Carlo Simulation
Discrete Event Simulation
Emulation

架构师三板斧

- 看清需求
- 选择方法
- **把握节奏**
  - 规划可达路径
  - 定期产出




Thanks!