clover

Clover: A NameNode Cluster Version of HDFS

马灿 中科院计算所 集成应用中心

HiC 2011 @ 12/2/2011

HDFS中NN的问题

- 现状
  - 单一NN
  - 冷启动过程缓慢，可用性低
- 元数据服务的扩展性
  - 受制于CPU、内存和网络
  - 随着系统规模扩展，单节点服务压力大
- 元数据服务的可用性
  - 单一故障点，需要外部HA组件

已有的NN改进方法

- HDFS Federation
  - ✓ 多个NN，分离独立的名字空间
  - ✓ 共享DN存储池
  - x 缺乏全局名字空间，互相共享数据困难
- HDFS-DNN
  - ✓ 使用HBase存储元数据
  - x 原型系统，无详细信息
- HDFS2
  - ✓ 分离名字空间与块管理
    - 分布式文件管理，集中式名字空间
  - x 名字空间服务的扩展性受限，不支持目录rename

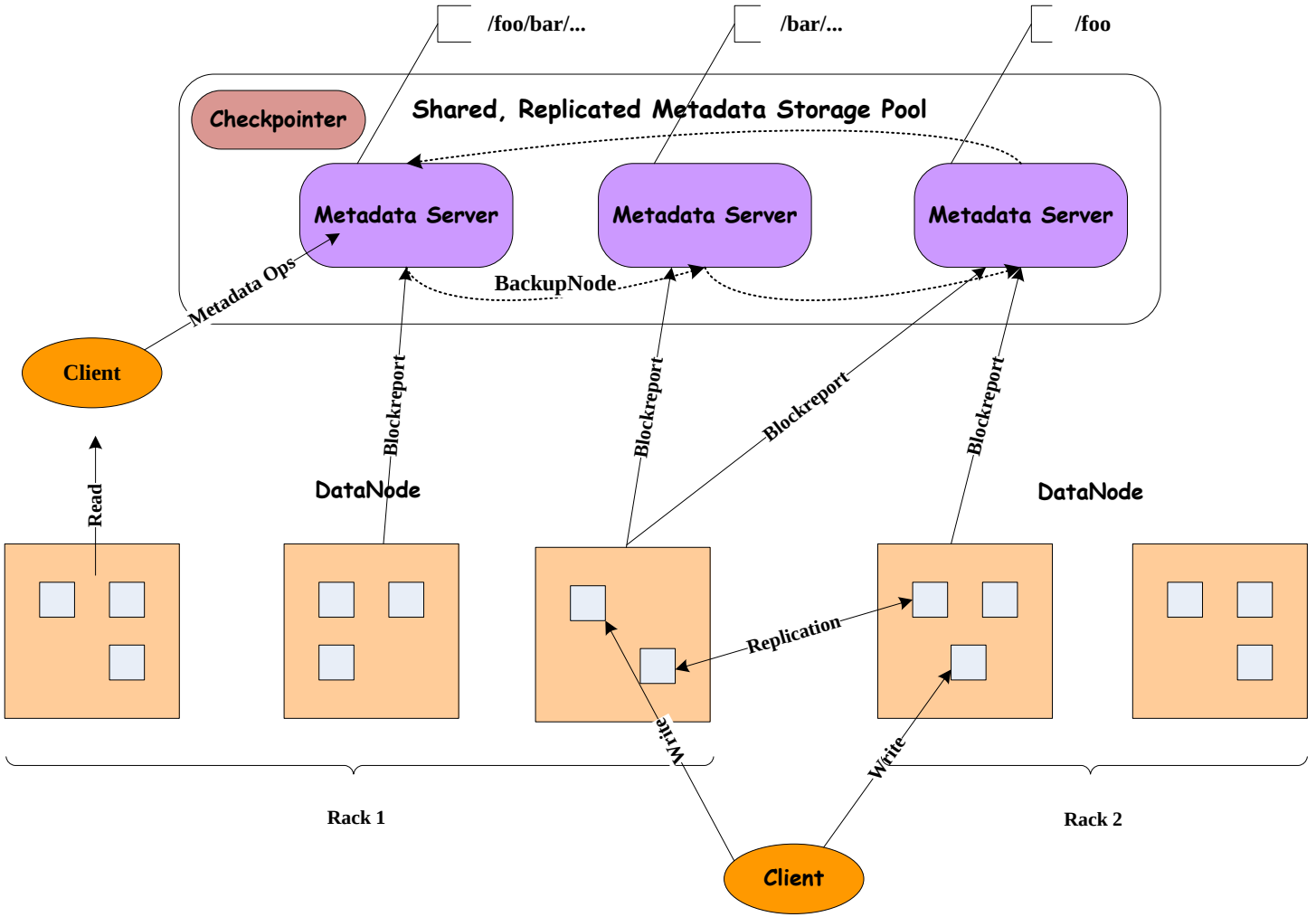
clover

Clover的NN Cluster方案

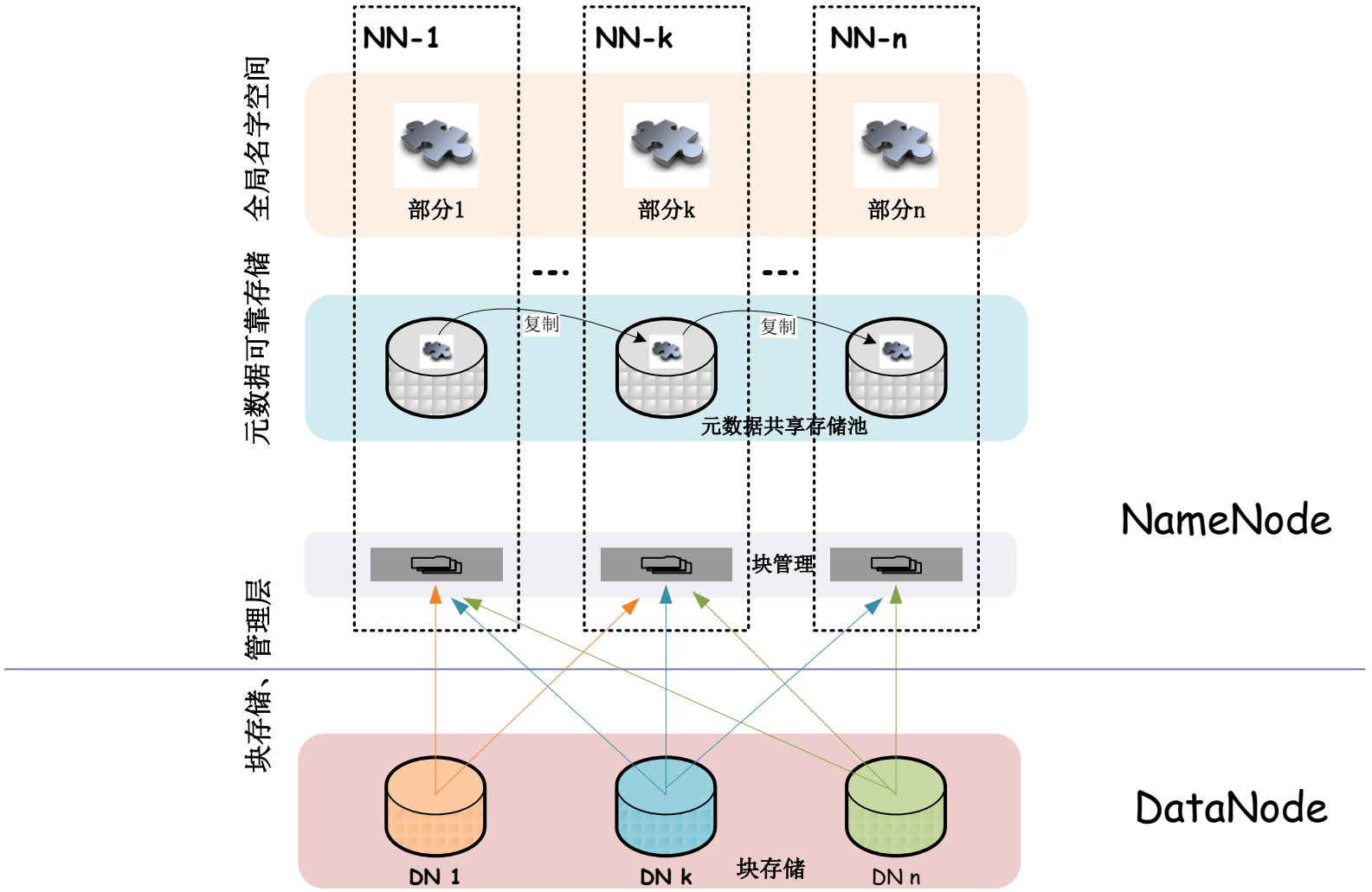
- 分析名字空间与块管理到集群上
  - 提供全局名字空间
  - 使用分布式日志确保mkdir, rename, delete等操作的一致性
  - DN块汇报分流
- 元数据共享存储池
  - 提供基于多副本的元数据高可靠存储
  - 在NN和BackupNode之间共享元数据文件
  - 支持分布式事务的日志
- 高可用方法
  - 元数据集群的多机互相热备
  - 全局检查点，缩短冷启动时间

clover

Bird's-eye View

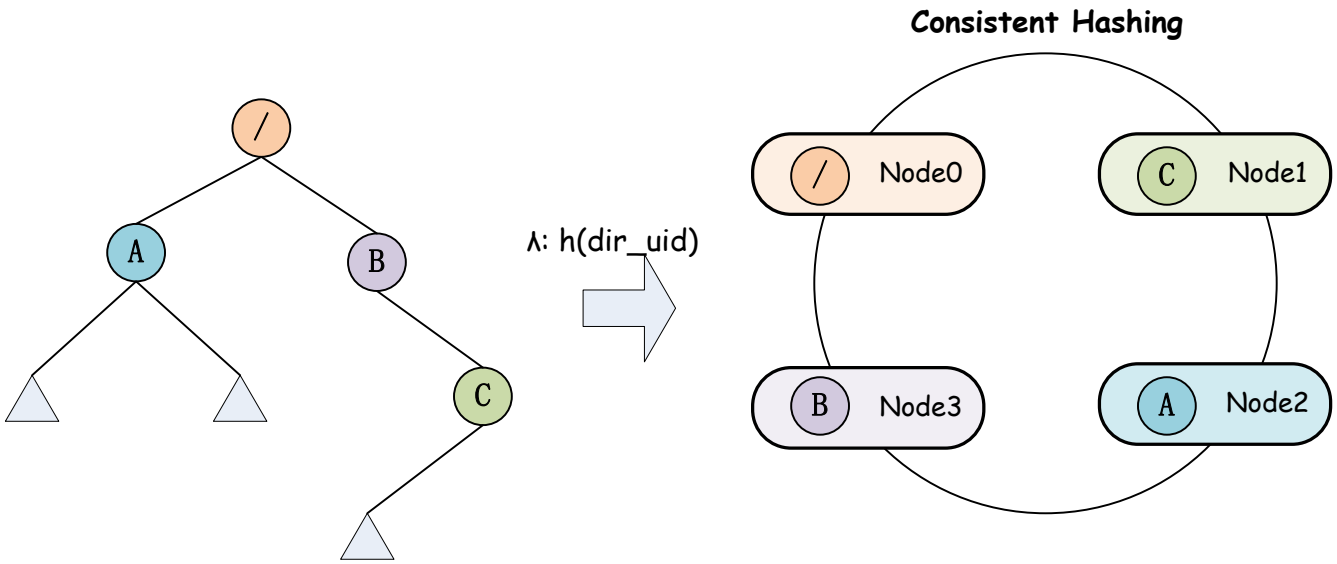


Clover NameNode Cluster



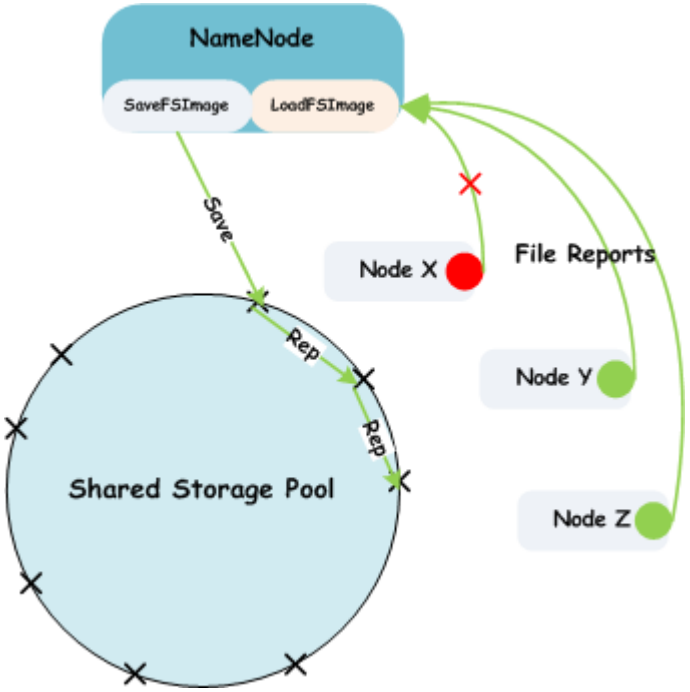
分布式名字空间

- 全局目录表（集中式or分布式）
  - 存储所有目录到唯一ID(dir_uid)的映射关系
- 基于目录的分布
  - dir_uid → 虚拟节点 → 物理节点
- 支持目录的rename

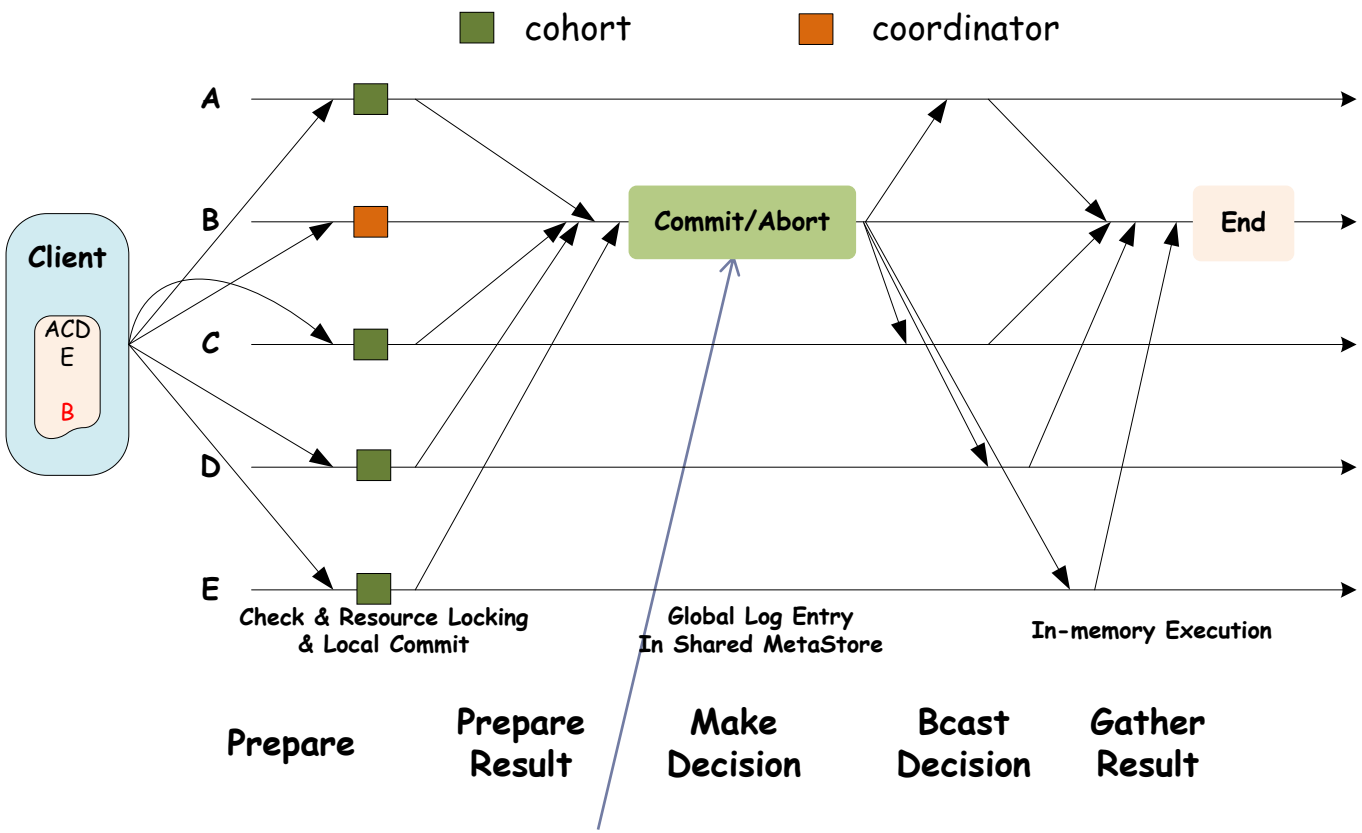


元数据文件的共享、可靠存储

- 由NN集群构建共享存储池
- 在文件级别自动复制
  - 一个local，多个remove
- 三种文件访问模式
  - FSImage → 大块顺序读写
  - Editlog → 小粒度的追加写、大块读
  - WAL → 小粒度并发追加写、并发读
- 性能要求
  - FSImage → 高读写带宽→Pipeline
  - Editlog/WAL → 低延迟→Sync to local disk + remote mem; Async to remote disk

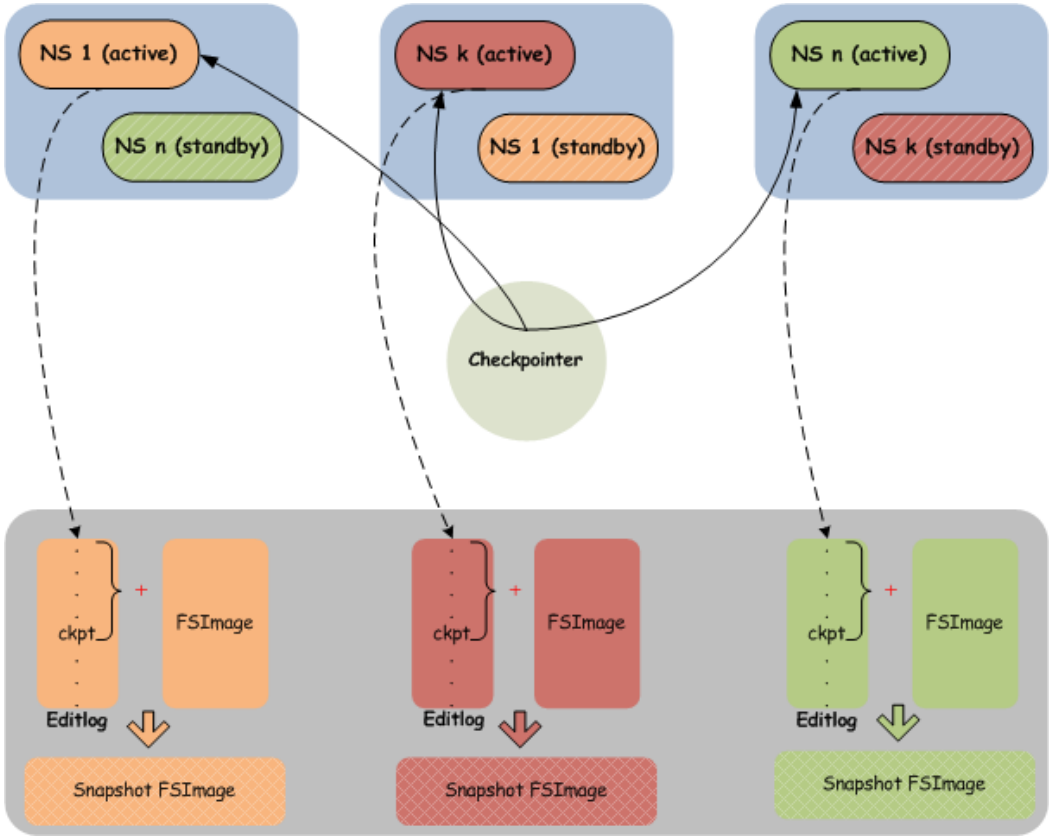


基于共享存储的快速分布式事务

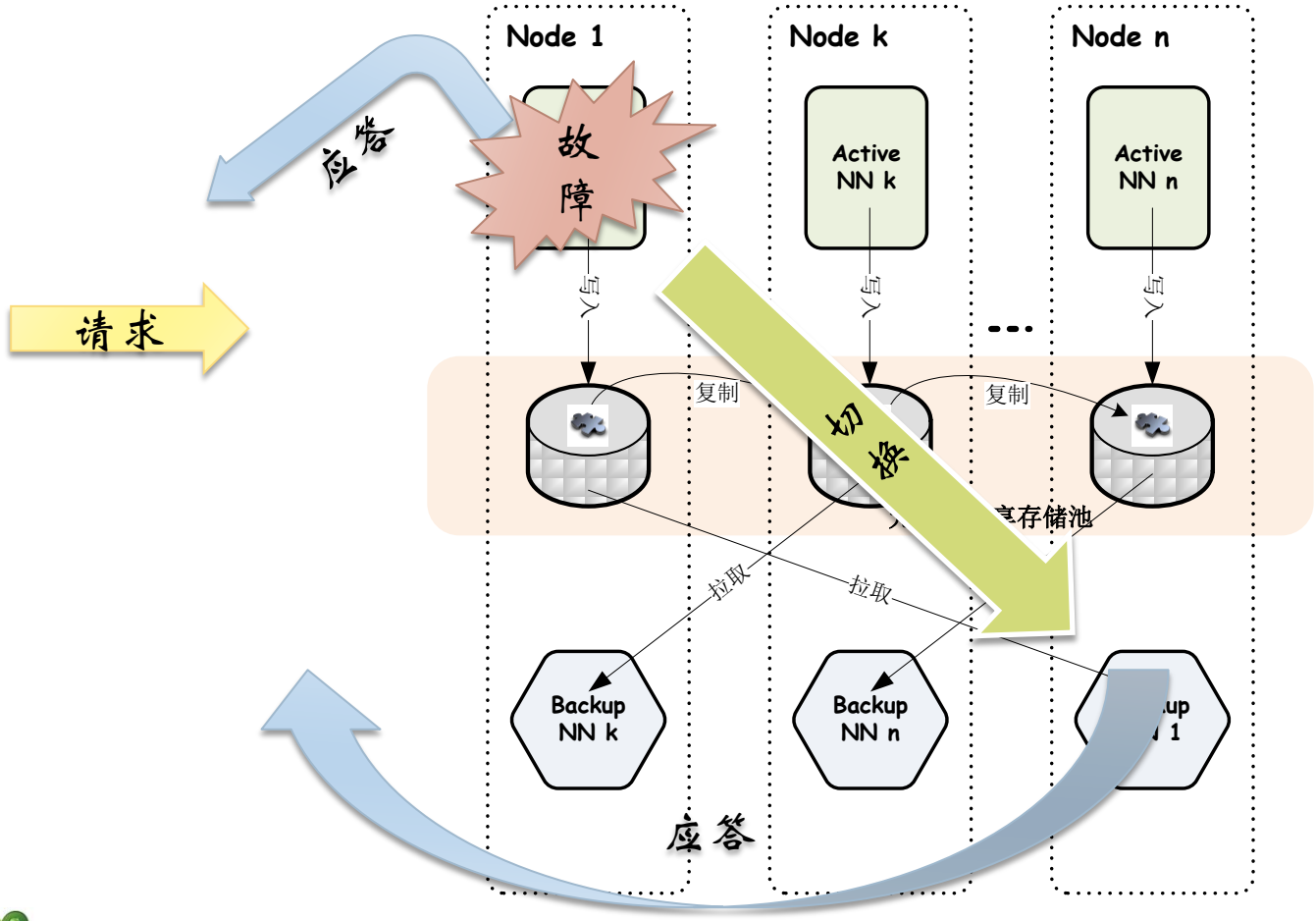


全局检查点

- 外部检查点工具触发
- 步骤
  - 全局Barrier同步
  - 所有的NN清空信道（Chandy-Lamport算法）
  - 所有的NN向Editlog写入Barrier标记
  - 外部检查点工具从共享存储池中读取每个NN的元数据文件合并FSImage和Editlog，直到遇到标记
    - 写回新的FSImage



多机间互相热备



基于社区的BackupNode方案

多机间互相热备

- 假设节点年化故障率为α，系统冷启动时间为T
- HDFS双机热备
  - 单机故障，热备生效→故障率：α；MTTR≈0
  - 双机故障，冷启动→故障率：α^2；MTTR=T
- Clover热备（N节点）
  - 单机故障，热备生效→故障率：$N\cdot\alpha$；MTTR≈0
  - 双机故障，冷启动→故障率：$N\cdot\alpha^2$；MTTR=T/N
  - 全系统故障→故障率：α^N；MTTR=T/N

通过热备提高了系统的可用性，同时降低了系统平均故障恢复时间（冷启动时间）

clover

Who are we?

- 中科院计算所 集成应用中心 并行数据组
- 王伟平 (wpwang@ncic.ac.cn) 副研究员
- 马灿
  - Twitter: @nkmacana
  - Email: ml.macana@gmail.com (G+: +Can Ma)
  - Blog: http://macan.posterous.com
  - Github: http://github.com/macan
- 周江 (zhoujiang@ncic.ac.cn)
- 王有为 (wangyouwei@ncic.ac.cn)

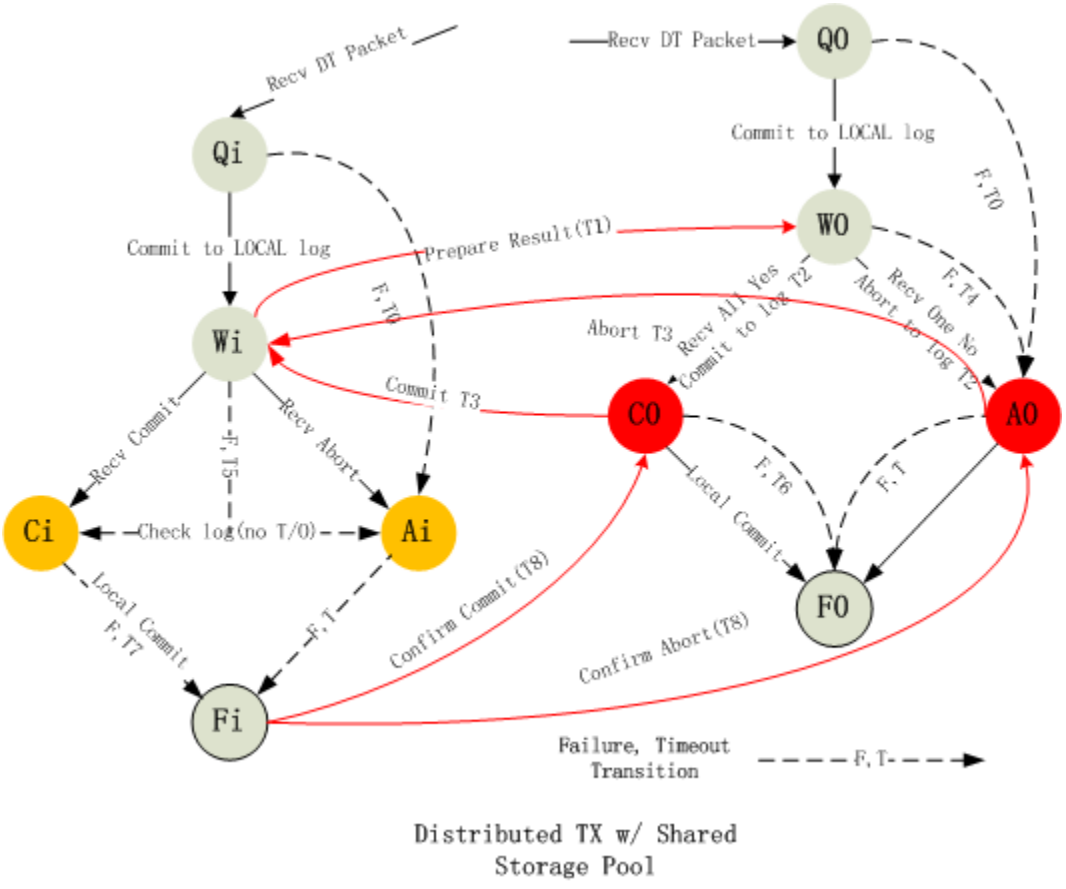
clover

Thanks!
Q & A

How to lookup a file?

- Input
  - File's pathname: F (e.g. /usr/macan/hello/all)
- Client Steps
  - Get dirname and basename: D(F)=/usr/macan/hello, B(F)=all
  - Get dir_uuid: Send request to $H_1(D(F))$ w/ D(F)
  - Get metadata: Send request to H_2(dir_uuid) w/ B(F)
- Pros and Cons
  - Rename friendly
  - 2RTT
    - D(F) → dir_uuid can be cached!

How complex operations are always right?



Distributed TX w/ Shared Storage Pool