Scalability of replicated metadata services in distributed file systems

Dimokritos Stamatakis, Nikos Tsikoudis
Ourania Smyrnaki, Kostas Magoutis

2012

Wendel Muniz de Oliveira

22 de Abril 2015

Roteiro

- Contexto
- Introdução
- Projeto
- Implementação
- Análise de Desempenho
- Trabalhos Relacionados
- Conclusões
- Referências

Contexto - Paxos

- Proposto por Leslie Lamport em 98
  - The Part-Time Parliament
- Problema de consenso
  - Grupo de nodos acordar em um determinado valor
    - Perda de mensagens ou entregues fora de ordem
- Benefícios para a replicação
  - Garante uma maior consistência de dados e tolerância a falhas

Contexto – Berkeley DB (BDB)

- Sistema de BD Embarcado
  - Pode ser associado à uma aplicação qualquer.
- B+tree,Hash,Recno,Queue
- Suporta ACID
- Replicação Master/Slave
- Compatível com protocolos Paxos
- DB SQL API
  - introduzida pela Oracle
  - Compatível com SQLite

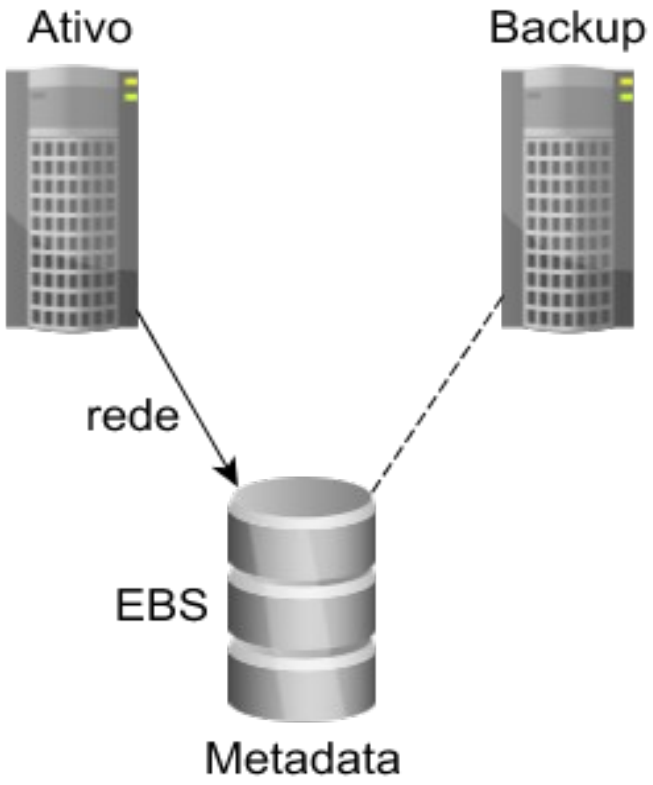
Contexto – Gerenciamento de metadados em SAD

- Sistemas como pNFS, PVFS e GoogleFS
  - Componente isolado
- Garantir que acesso aos metadados não obstrua o acesso aos dados
- Simplicidade de projeto
- Escalar partes independentes do sistema

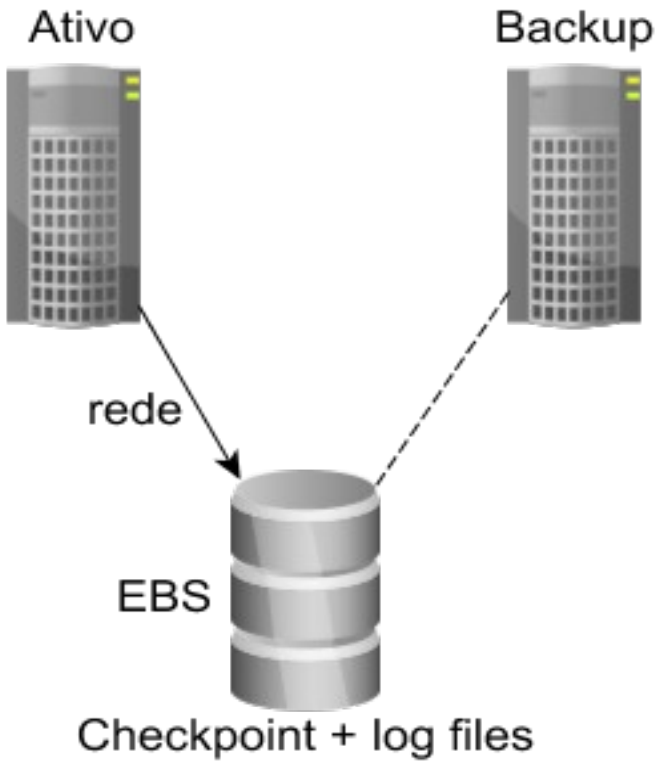
Introdução - Motivação

- Performance somente ao acesso aos dados
- Metadados tratados de forma isolada
  - muitas vezes não escalável
- Sistemas compatíveis com Paxos
  - não aplicado com frequência à cenários típicos de SAD
  - Podem ser aplicados para obter escalabilidade

Introdução – Opções de Arquitetura Existentes

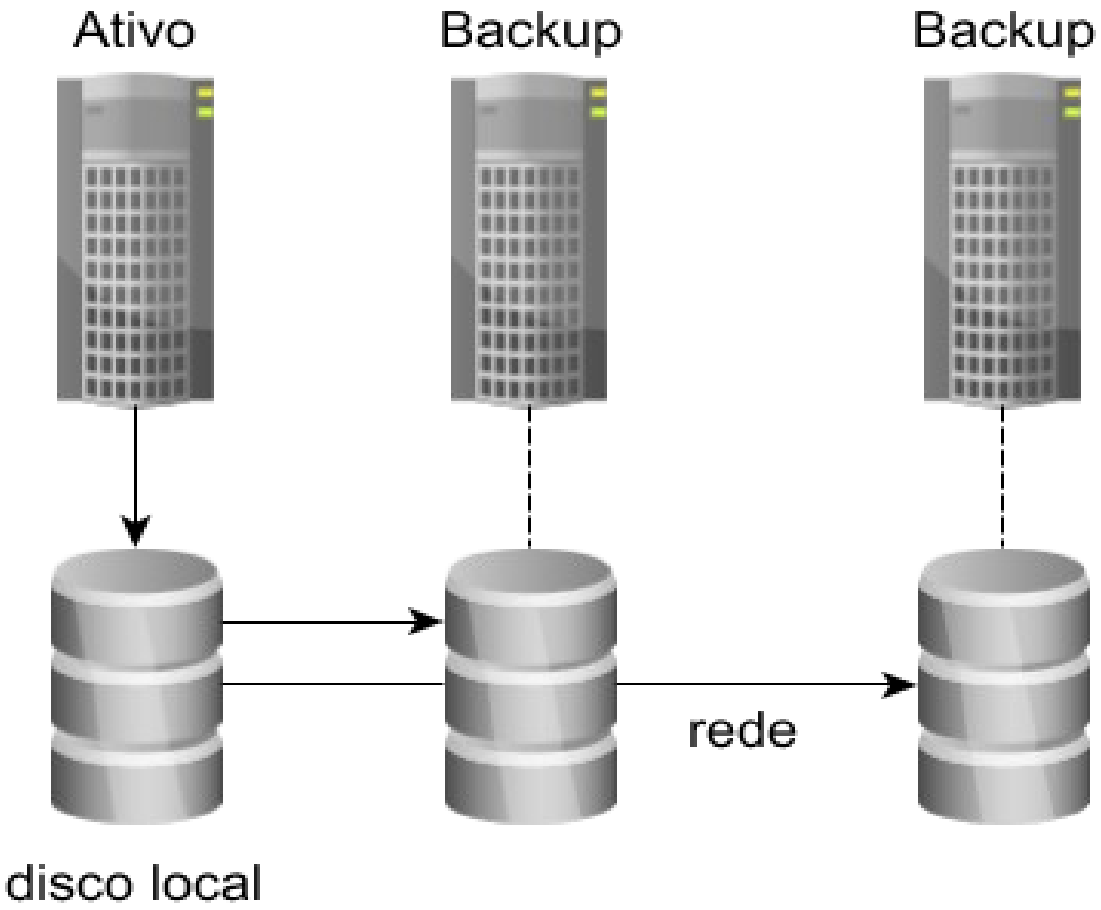


PVFS



HDFS

Introdução – Arquitetura Proposta



Introdução – Arquitetura Proposta

- Replicação Consistente no nível do SA
  - Repositório chave - valor
- De simples a replicado através de:
  - Camada de rede com auto disponibilidade
  - Replicas idênticas do servidor de metadados
  - Banco de dados replicado

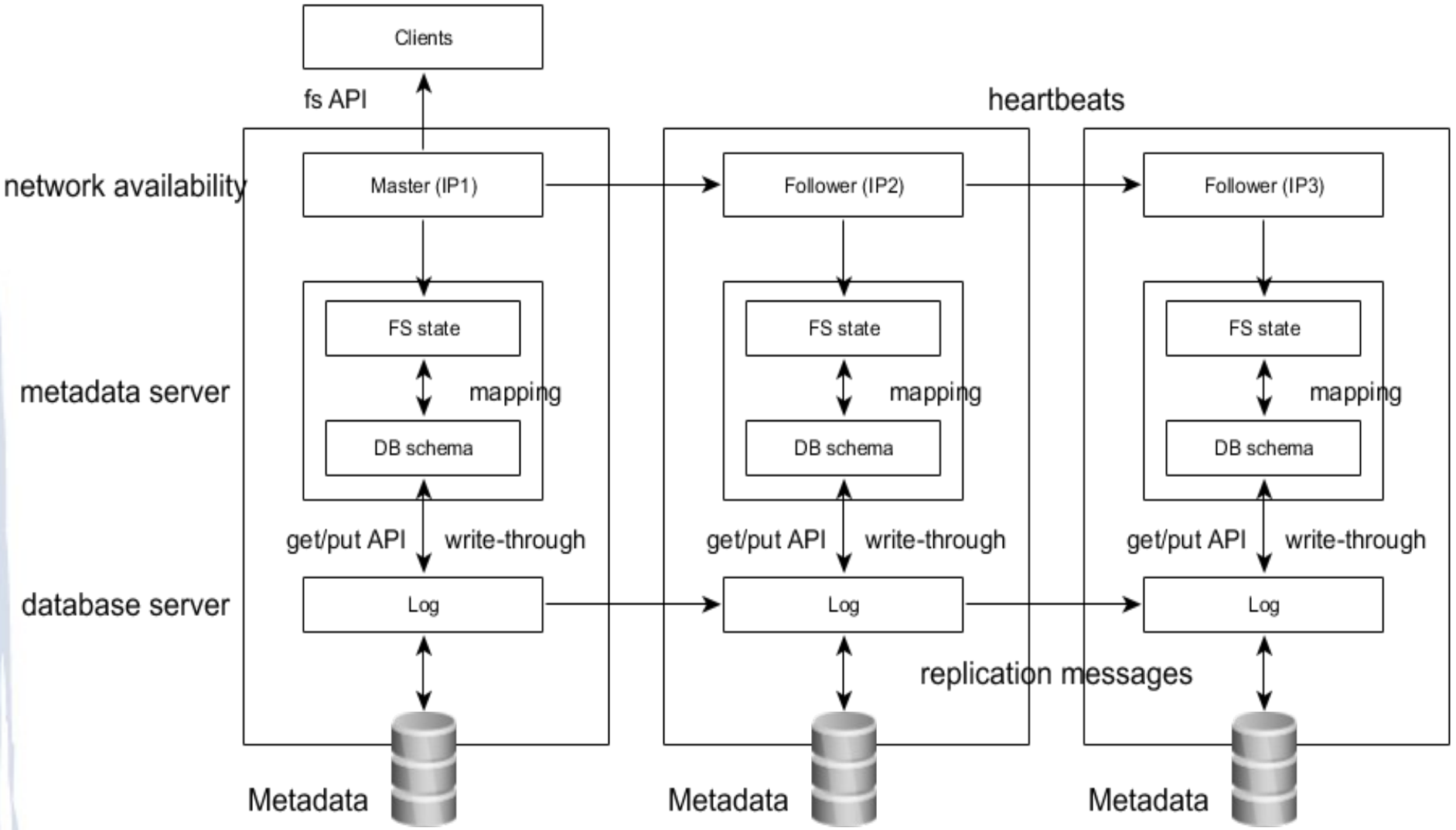
Introdução – Contribuições

- Metodologia genérica
  - Serviços de metadata com auto disponibilidade
- Servidores metadata replicados
  - Projeto e implementação
  - No contexto do PVFS e HDFS
- Avaliação da Arquitetura proposta
  - Amazon Web Services EC2 Cloud

Projeto – Visão Geral

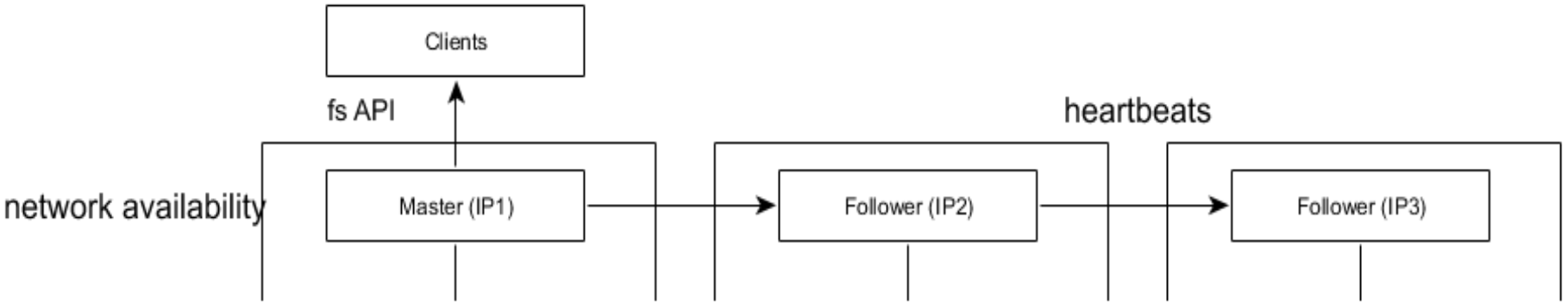
- N nodos servindo metadados
  - 1 master, n - 1 seguidores
  - Clientes acessam o master para leitura e escrita
- Cada nodo possui três camadas de sw
  - Auto disponibilidade de rede
    - Receber requisições dos clientes
  - Servidor de metadados
    - Mapeamento fs → esquema chave-valor
  - Servidor de Banco de Dados

Projeto - Arquitetura



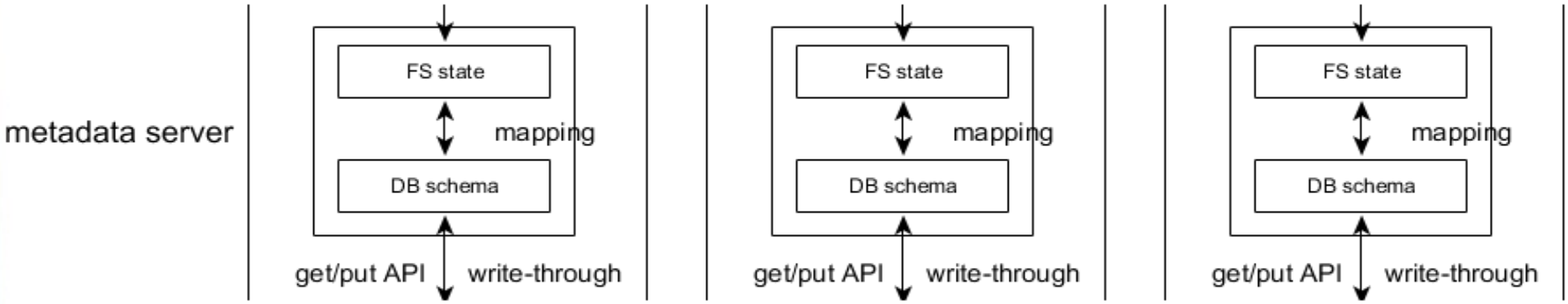
Projeto – *network availability*

- Mapear dinamicamente um endereço IP para o nodo eleito correntemente como Master



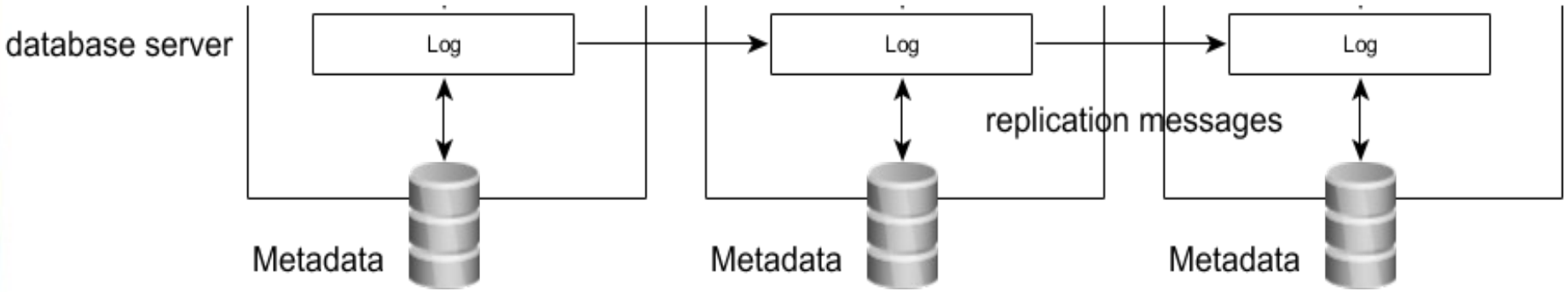
Projeto – *metadata server*

- Mapeamento entre o sistema de arquivos e o esquema chave – valor
- *Commit* transações / Sincronização com o disco
  - Política de Confirmação (*Ack policy)*
    - *all,one,majority,without waiting*
  - Trading off durabilidade e peformance



Projeto – *database server*

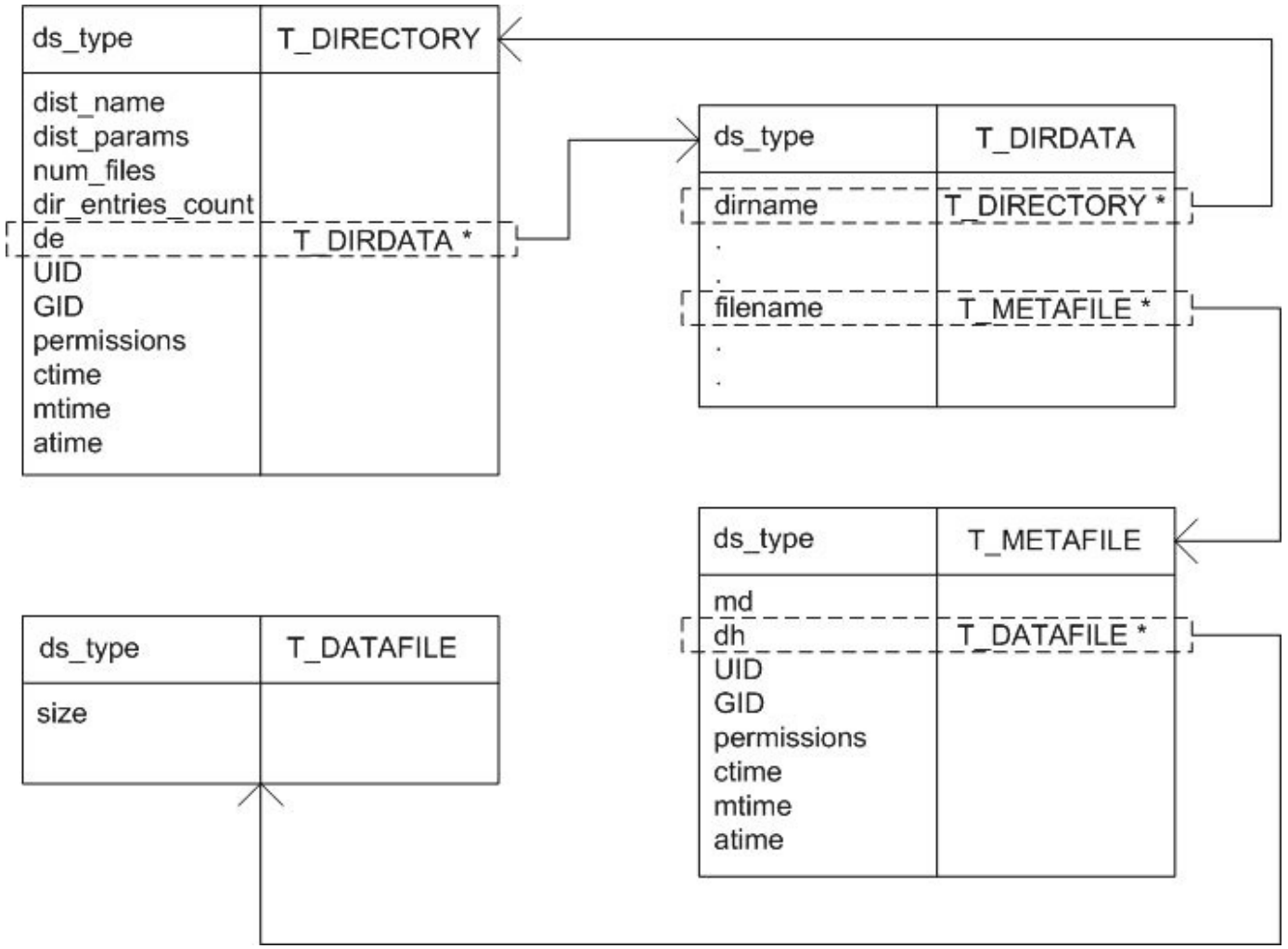
- Replicação consistente
- Política de Confirmação *commits* (*ack policy*)
  - Protocolo de replicação distribuída do BDB
- Coordenação entre o BDB e a camada de rede para eleição do master



Implementação - PVFS

- BDB em modo *single-node*
- Esquema chave-valor não foi modificado
- Sem controle transacional
- Alterações
  - BDB iniciado em modo replicado
  - Foi modificado para ser transacional
  - Master *leases* para leituras consistentes
  - BDB elege um master e comunica ao PVFS
    - PVFS notifica a camada de rede

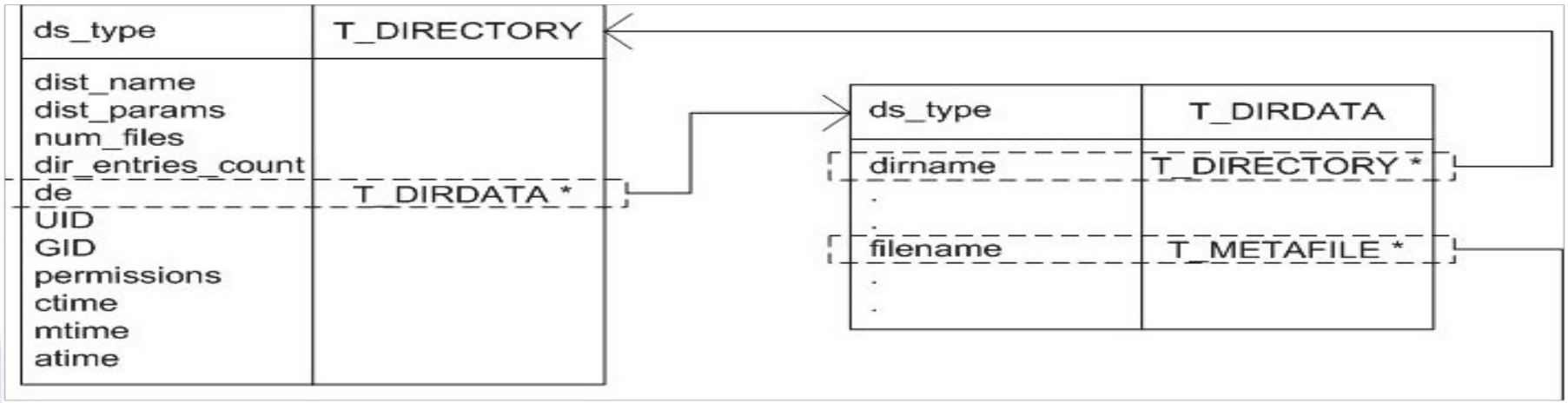
Implementação - PVFS



Implementação - PVFS

mkdir /dir1/:

1. get *"de"* from handle 1 ⟶ handle 2.
2. get *"dist_name"*, *"dist_params"*, *"num_dfiles"* from handle 1.
3. create new directory object with handle 3.
4. put T_DIRECTORY, id, gid, permissions, {a,c,m}time into *"ds_type"*, *"uid"*, *"gid"*, *"permissions"*, *"{a,c,m}time"* in handle 3.
5. create new directory data object with handle 4.
6. put T_DIRDATA into *"ds_type"* in handle 4
7. put handle 4 into *"de"* of handle 3
8. get *"de"* of handle 1 ⟶ handle 2.
9. put handle 3 under key *"dir1"* in handle 2.
10. get *"directory_entries_count"* of handle 2 ⟶ X.
11. put *"X+1"* into *directory_entries_count* of handle 2.

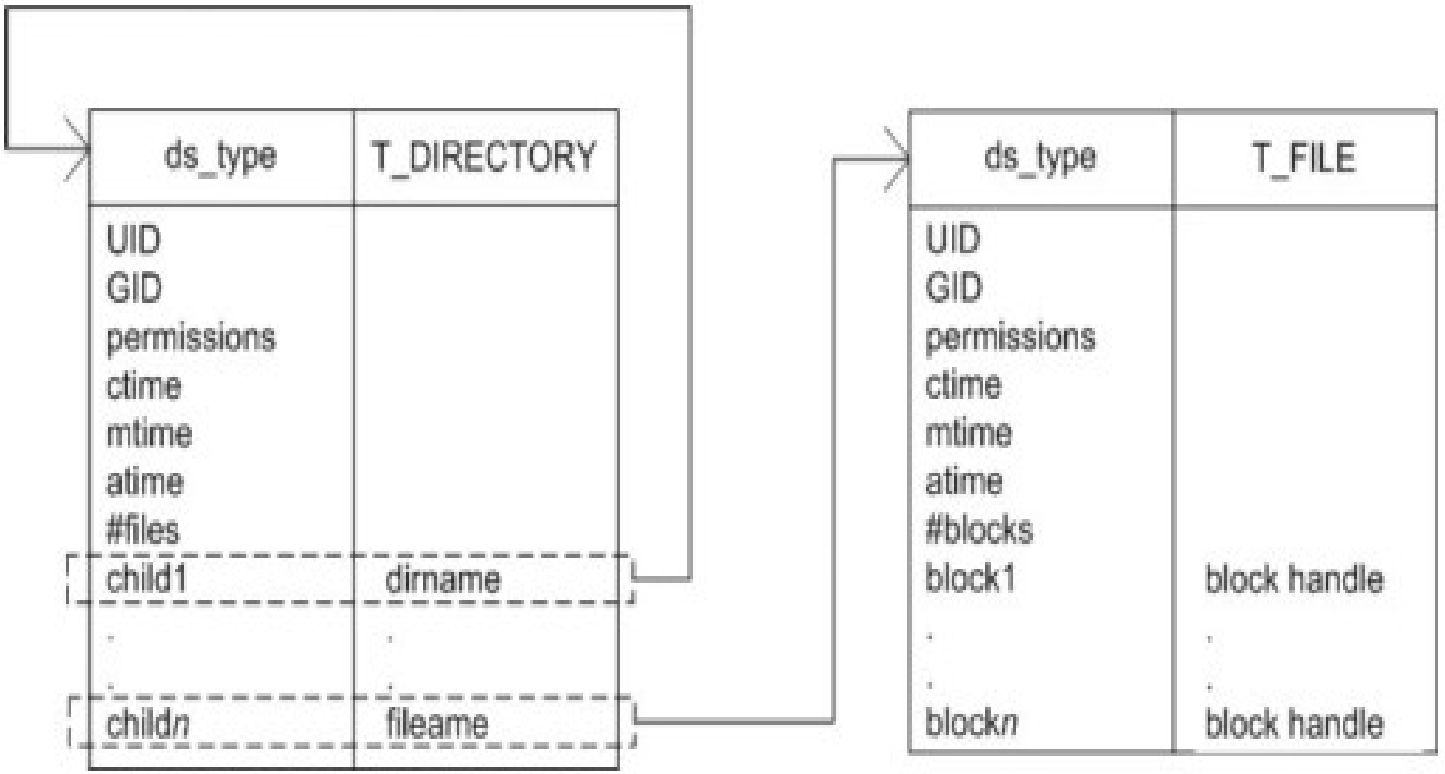


Implementação - HDFS

- Banco de dados em memória
  - *NameNode*
  - Armazena log em uma estrutura chamada *journal*
- Modificações
  - BDB como *back-end* para o NameNode
  - persistir no BDB
  - NameNode como *cache*
  - BDB substitui o *journal* na recuperação de dados
    - Método de *recovery* padrão substituído

Implementação - HDFS

- esquema de BD é uma representação da imagem do *NameNode*



Implementação – Durabilidade/ Performance *tradeoffs*

- Para obter durabilidade através do BDB
  - escritas em disco síncronas
  - Protocolo de replicação distribuída
- Combinar os dois meios
  - performance levemente prejudicada com durabilidade forte

Avaliação

- Configuração
  - Amazon EC2
  - 1.7GB de memória - 168GB disco virtual local
  - PVFS 2.8.1 - HDFS 0.20.205.0
  - BDB 5.30 512MB de cache
- Parâmetros
  - Qtd. replicas
  - Ack policy (*ONE, QUORUM, ALL*)
  - Commits síncronos ou assíncronos

Avaliação – *single metadata server*

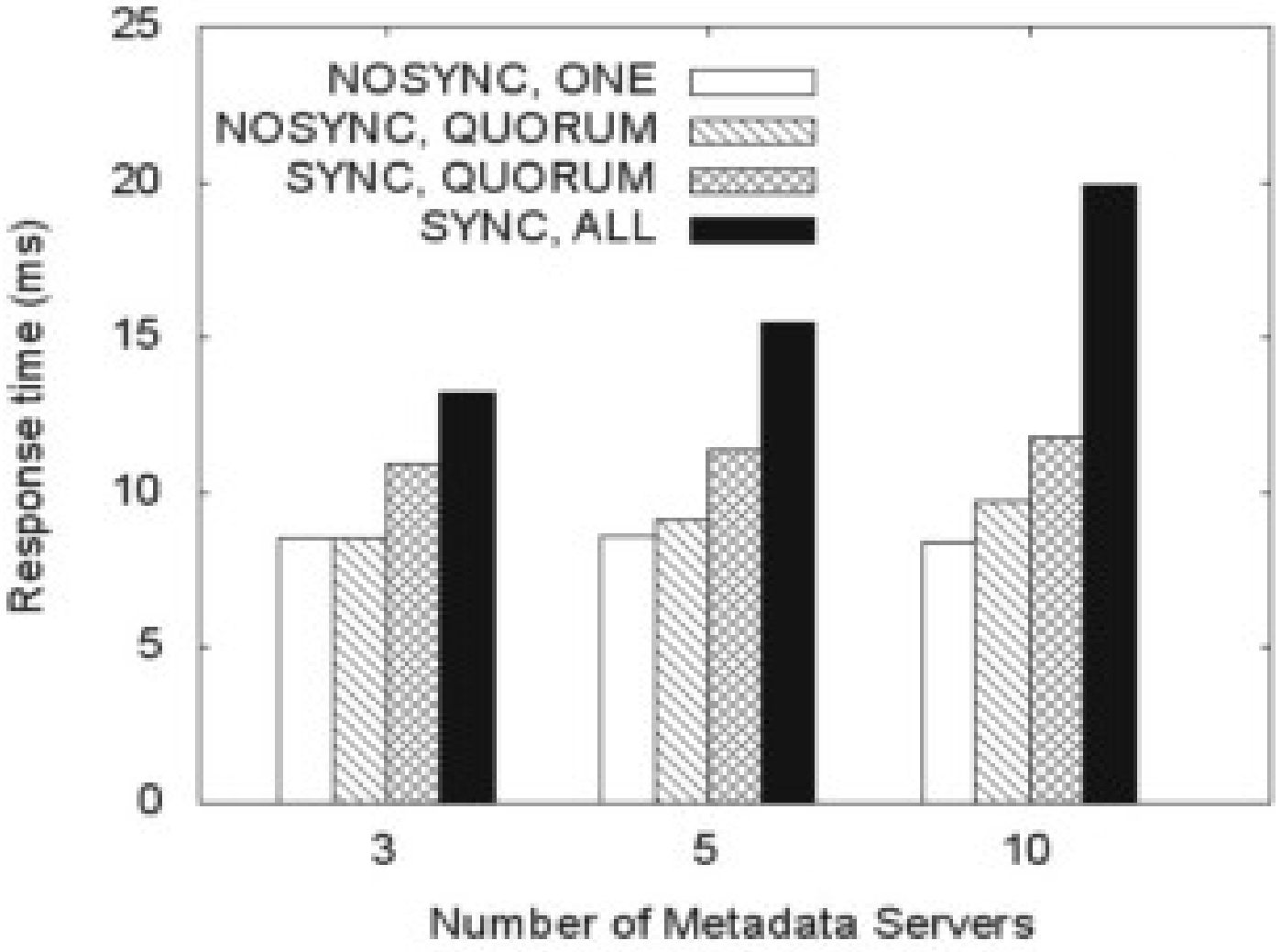
- Baseado no *response time*
  - 2000 comandos mkdir
  - Commits síncronos e assíncronos
  - Impl. do PVFS 15%-20% mais lento que original
  - PVFS possui resp. time mais alto devido ao esquema
  - Impl. HDFS mais lento com múltiplos arquivos

	PVFS		HDFS	
	SYNC	NOSYNC	SYNC	NOSYNC
Original	4.5	4.3	-	3.0
1-repl single file	5.1	5.1	3.7	3.0
1-repl one file per table	-	-	10.9	10.9

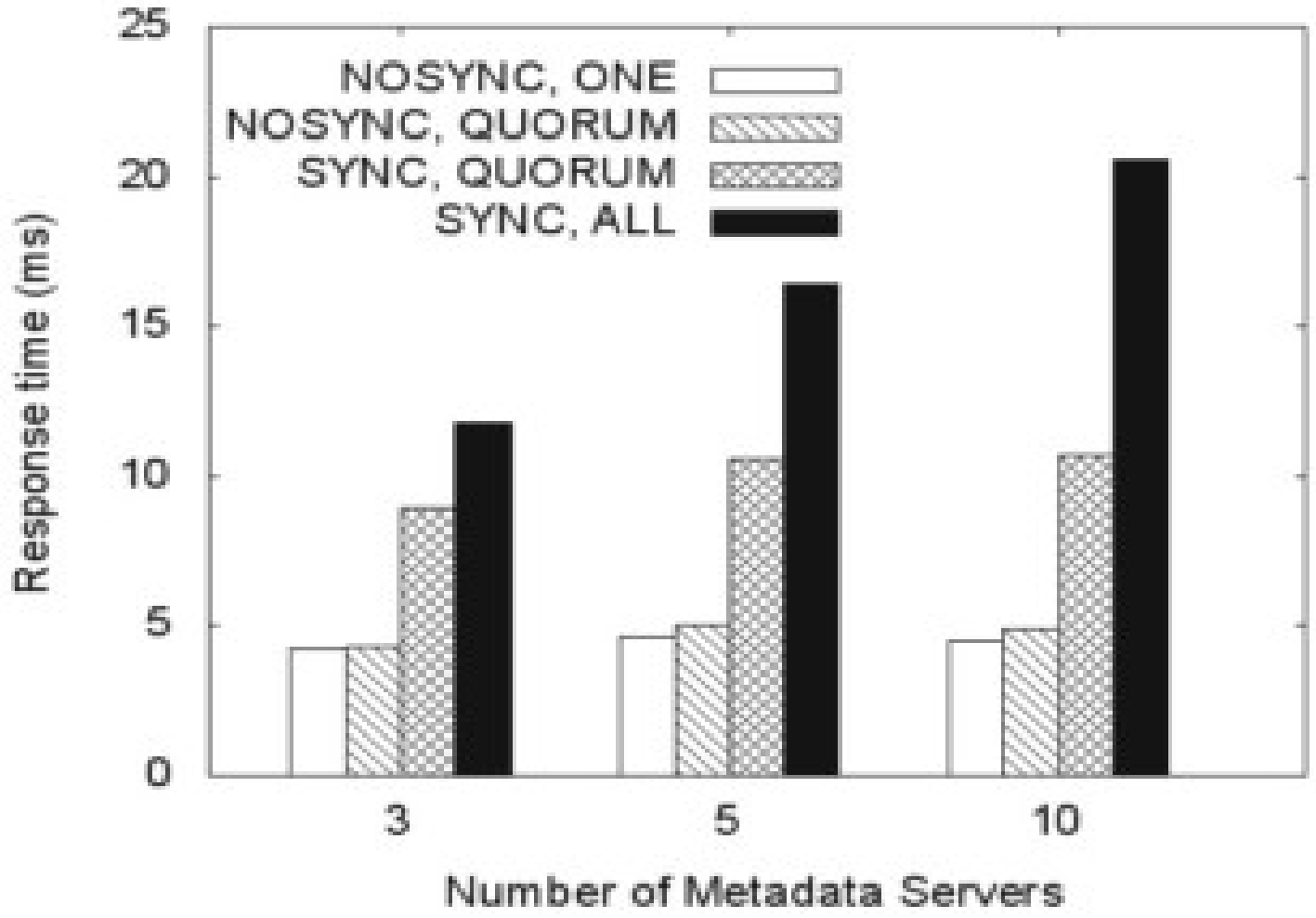
Avaliação – 3, 5 e 10 replica

- Baseado no *response time*
  - 2000 comandos mkdir
  - Quatro configurações
    - Mais estrita (*SYN, ALL*)
    - Paxos-Like (*SYNC,QUORUM*)
    - Performance/reliability (*NOSYNC, QUORUM*)
    - Mais relaxado (*NOSYNC, ONE*)

Avaliação – 3, 5 e 10 replica - PVFS



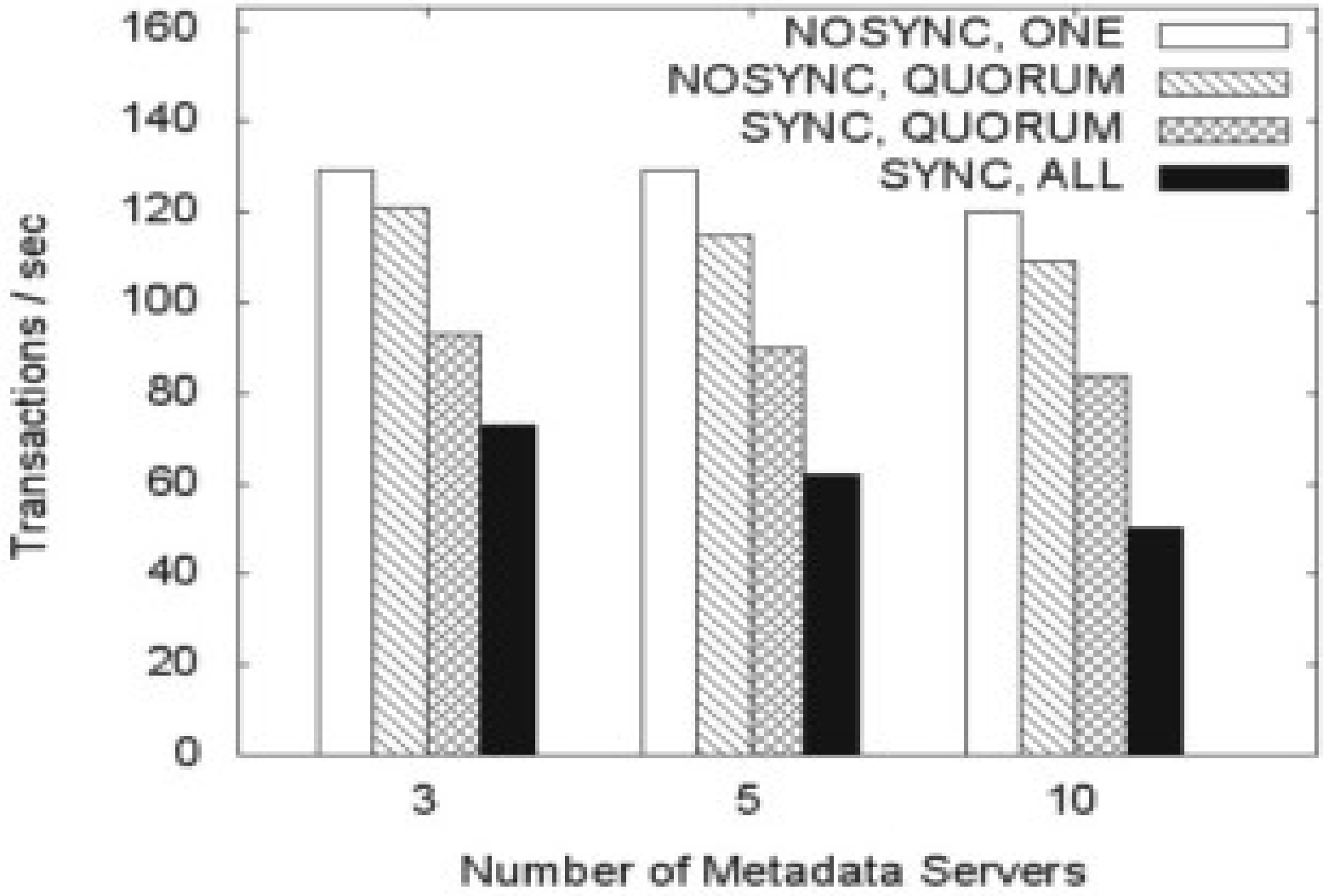
Avaliação – 3, 5 e 10 replica - HDFS



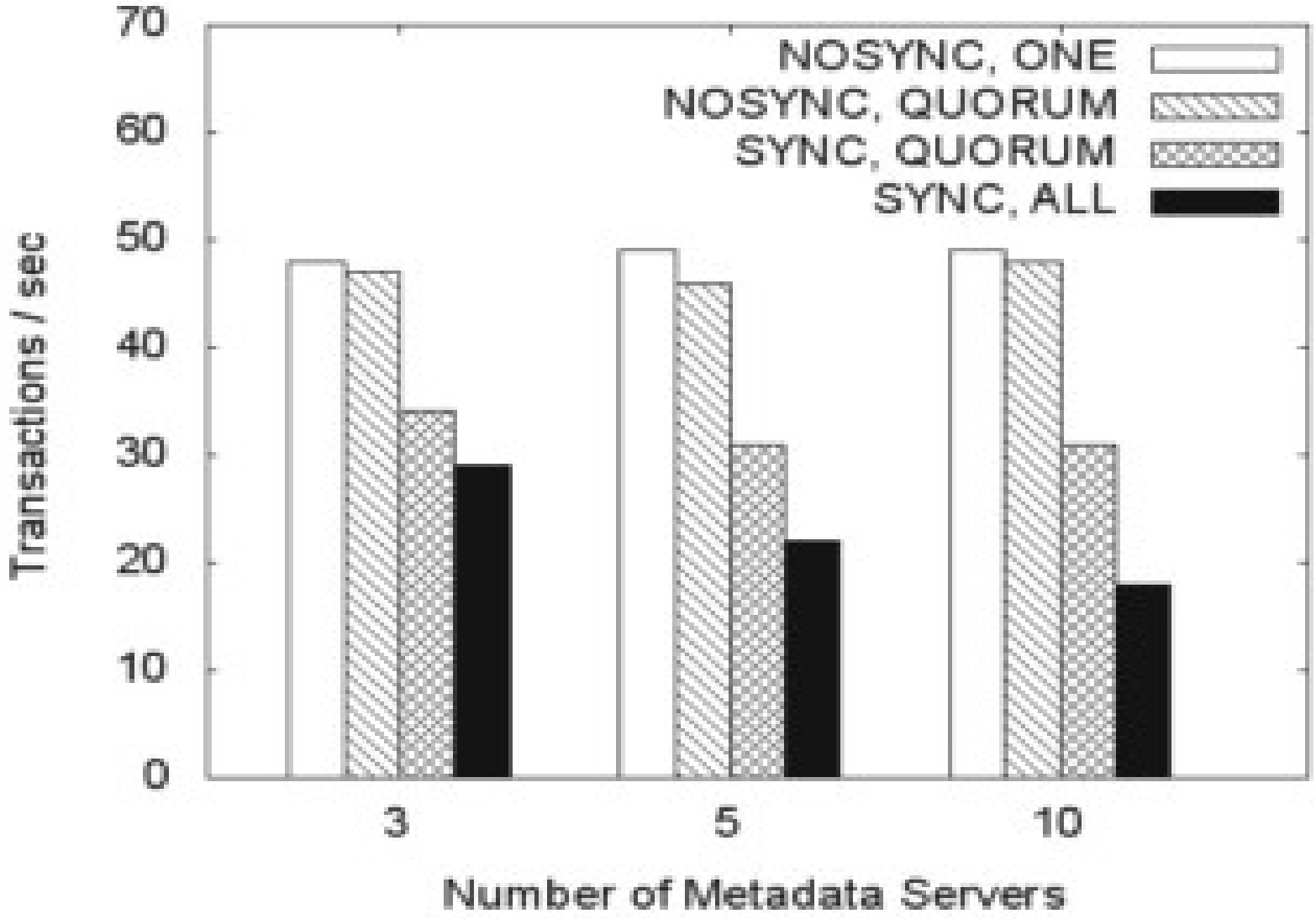
Avaliação – *Postmark*

- Benchmark sintético
  - Carga de Trab. Típica de provedores internet
  - Operações diversas
- Dinâmica
  - Criação de 500 arquivos com tamanhos entre 8-32KB
  - Após executa 2000 transações
    - criação de arquivos intercaladas com remoção

Avaliação – *Postmark* - PVFS



Avaliação – *Postmark* - HDFS



Avaliação – Recuperação de dados

- Comparar versão padrão do HDFS com a versão replicada

	Network/disk failover	State reconstruction	Total client outage
HDFS (8, 1)	38+35	7.3	80.3
HDFS (0.8, 1)	38+35	1.9	74.9
HDFS 3-replicas	38	0	38

Trabalhos Relacionados

- Mecanismos replicação
  - *State machine*
    - Paxos
      - Petal
      - Niobe
      - Chubby
      - BDB
  - *Process-pairs*
  - *Quorum systems*

Trabalhos Relacionados

- Localização Metadados
  - Centralizado
    - NFS
  - Distribuir simetricamente
    - Petal
    - Frangipani
    - XFS
  - Separar metadados dos dados
    - HDFS
    - PVFS
    - PNFS

Conclusões

- Alta disponibilidade e durabilidade com mínima penalidade de performance
- *commits* síncronos + (*QUORUM,ALL*)
  - Performance visivelmente afetada
- *commits* assíncronos não sacrificaram a durabilidade
- aceitável para o HDFS

Referências

Stamatakis, D., Tsikoudis, N., Smyrnaki, O., & Magoutis, K. (2012). Scalability of replicated metadata services in distributed file systems. Lecture Notes in Computer Science

Rao, J., Shekita, E. J., & Tata, S. (2011). Using Paxos to Build a Scalable, Consistent, and Highly Available Datastore. Proceedings of the VLDB Endowment, 243–254.

Perl, S. E. (2006). Data management for internet-scale single-sign-on. WORLDS06 Proceedings of the 3rd Conference on USENIX Workshop on Real Large Distributed Systems

Seltzer, M., & Corporation, O. (n.d.). Berkeley DB : A Retrospective 2 A Brief History of DB, 1–8.

Olson, M. a, Bostic, K., & Seltzer, M. (1999). Berkeley {DB}. Proceedings of the Annual Conference on {USENIX} Annual Technical Conference