Supporting Partial Data Replication in Distributed Transactional Memory

João A. Silva, Tiago M. Vale, Ricardo J. Dias, Hervé Paulino, João M. Lourenço

CENTER FOR INFORMATICS AND INFORMATION TECHNOLOGIES

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

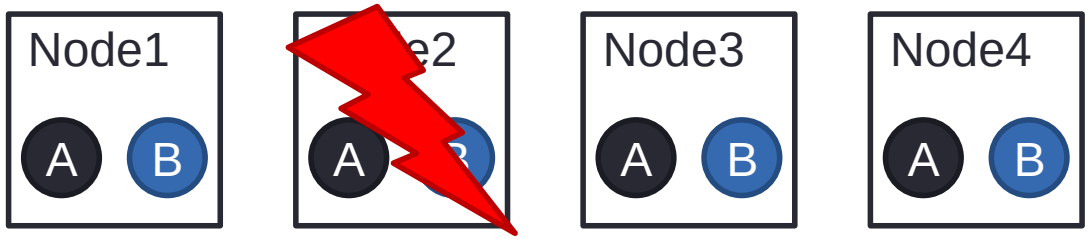
(Distributed) Transactional Memory

- Transactional Memory (TM)
  - Becoming mainstream
    - Deployed by CPU manufacturers...
    - And in reference compilers
- Scalability and Dependability
  - Combining TM with data replication & distribution
- Distributed Transactional Memory (DTM)
  - (Distributed) Concurrency control mechanism
  - Software Transactional Memory applied in the distributed context

Distributed Transactional Memory

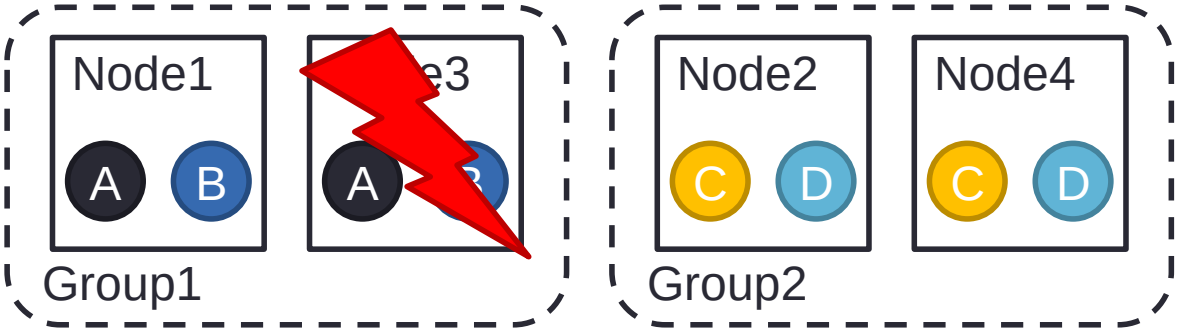
- Full Data Replication
  - Each node replicates the entire system's dataset

  + Data survival maximized
  + Workload distribution
  − Limited storage capacity
  − Coordination among all nodes



Partial Data Replication

- Each node replicates a subset of the system's dataset
  - + Limits coordination to "necessary" nodes
  - + Less storage capacity occupied
  - − Overhead in data partitioning and nodes management
  - − Latency when accessing remote data



Proposal

Introduce support for

partial data replication in a DTM system

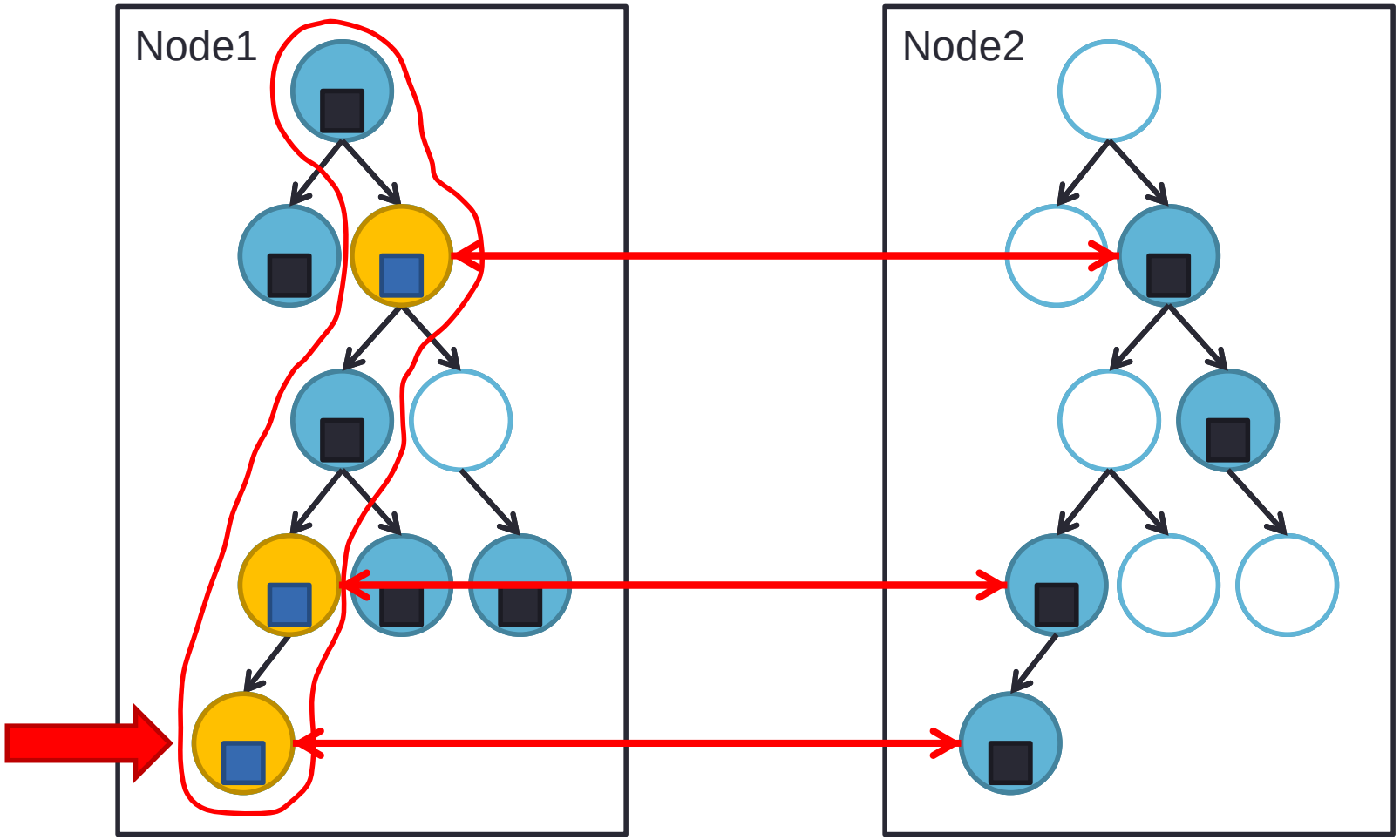
targeting a

general purpose programming language

Challenges

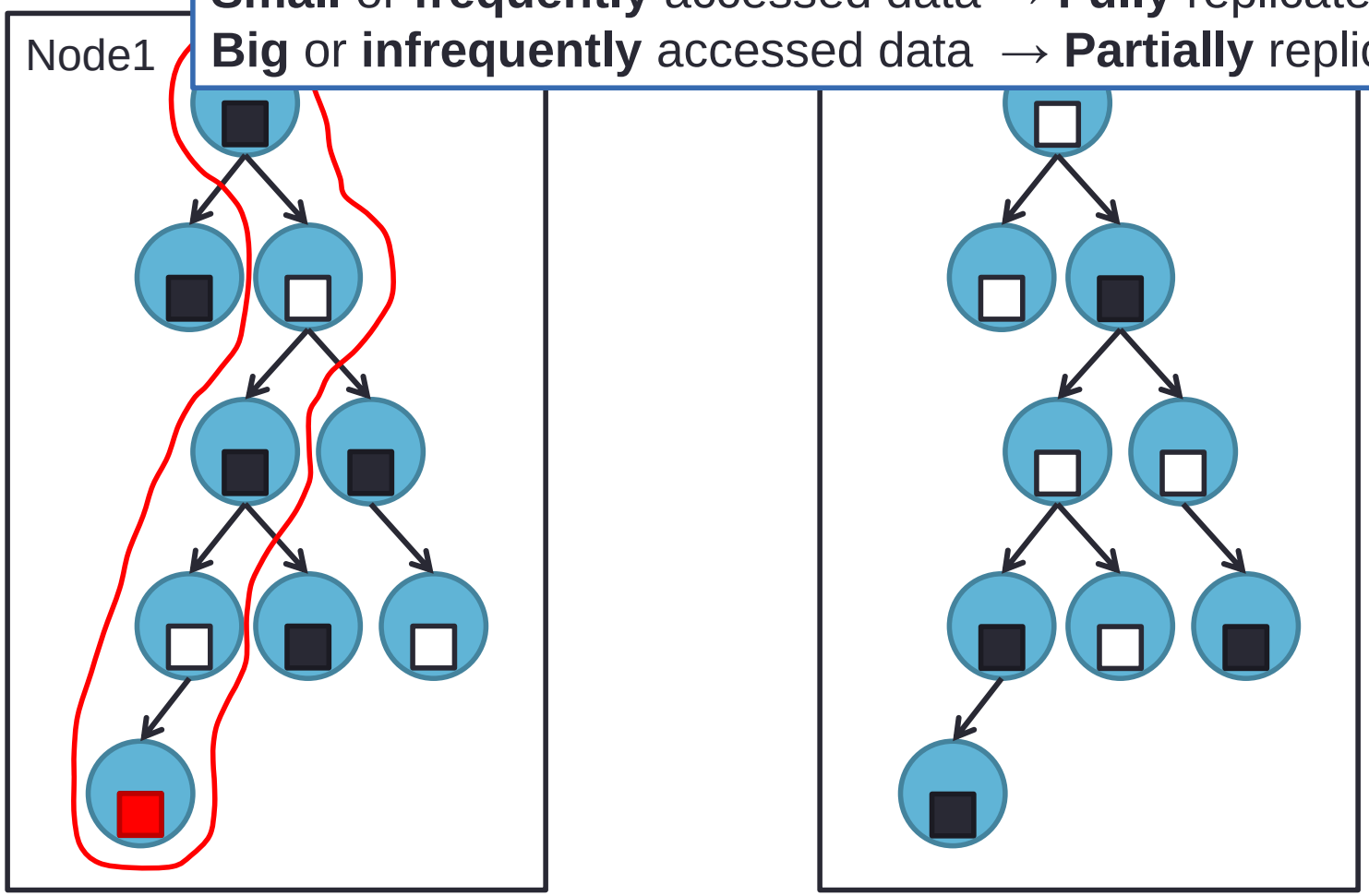
- What data should be partially replicated?
- How to express that replication level?
- Which are the architectural and functional requirements for a runtime system support partial data replication?

What should be partially replicated?



What should be partially replicated?

Small or **frequently** accessed data → **Fully** replicated
Big or **infrequently** accessed data → **Partially** replicated



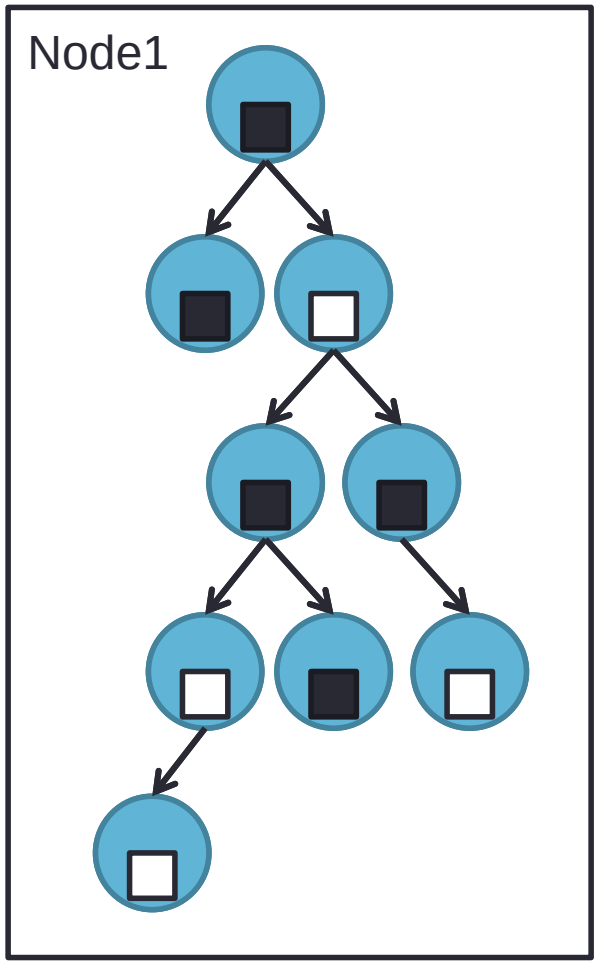
How to express the replication level?

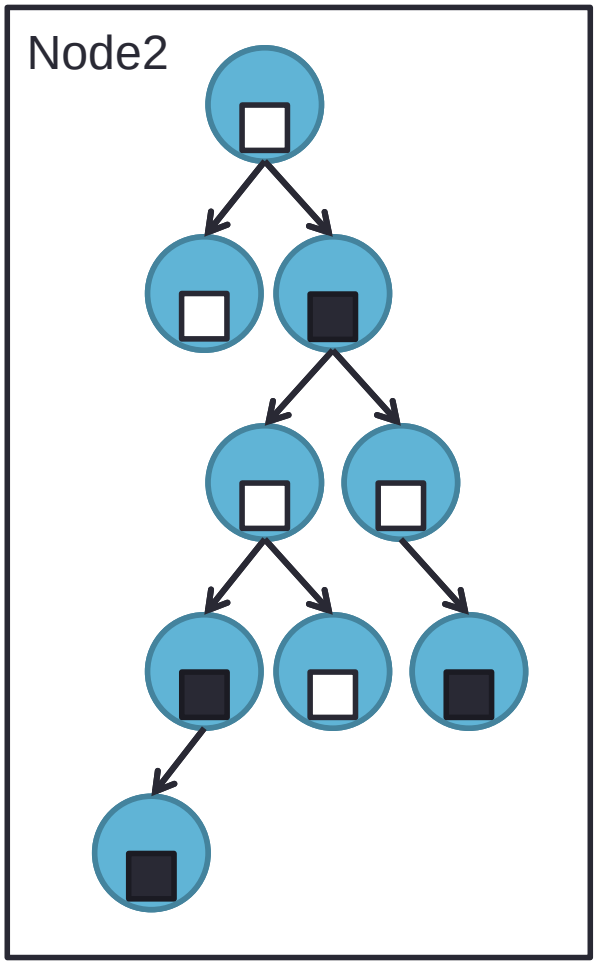
- Delegate the decision to the programmer
  - + Expressiveness
  - + Generality
  - − Reasoning about the impact of the replication model in application's performance

- Concern can be confined to the implementation of data structures

How to express the replication level?

```
class Node {
  int key;
  @Partial
  Object value;
  Node l;
  Node r;
  ...
}
```



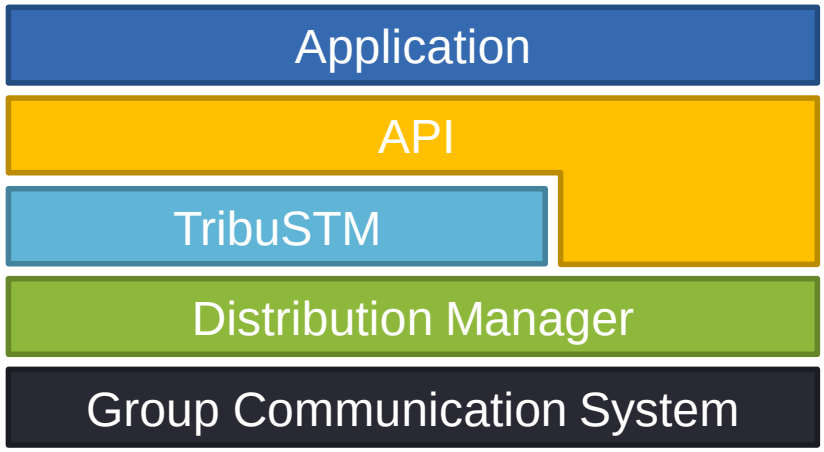


Our Starting Point

- TribuSTM [DVL12]
  - Extension to DeuceSTM [KSF10]

- TribuDSTM [VDL12]
  - Modular
  - Non-Intrusive

  - Local Concurrency Control
  - Distribution Manager
  - Communication System
  - Full Data Replication



Runtime System Support

- Groups partitioning
- Data partitioning
- Remote objects location and fetching
- Transactions' certification in partial replication environments
- Processing of the `@Partial` annotation

Experimental Results

- Evaluation goals:
  - Does it make an efficient use of memory?
  - How does it compare with full replication in different workloads?
    - Read/Write dominant workloads
    - Workloads that modify mainly fully replicated data
    - Workloads that modify mainly partially replicated data

Experimental Results

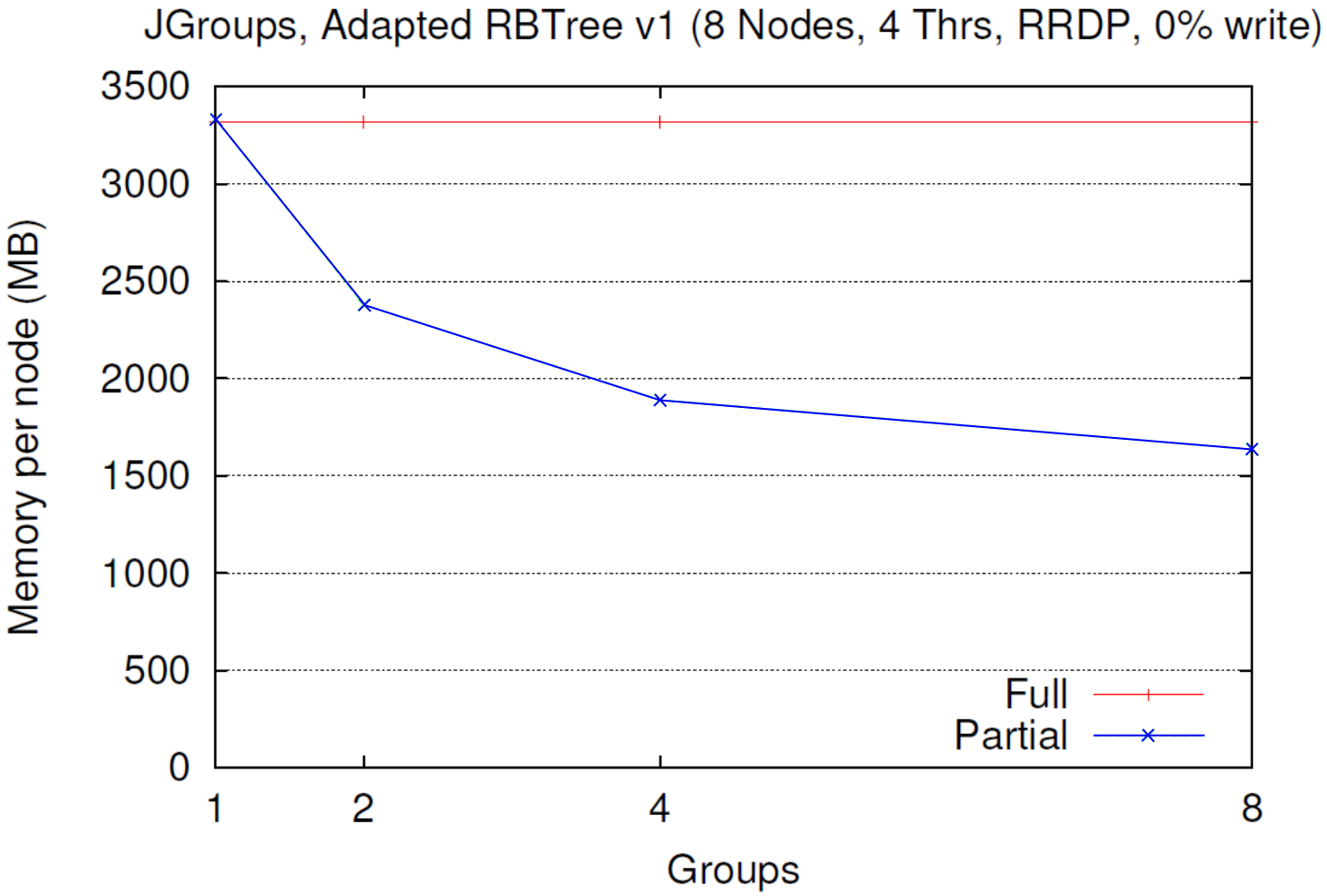
- Cluster@DI
  - Heterogeneous
  - 8 nodes
  - Ethernet Gigabit

- Benchmarks
  - Red-Black Tree
  - TPC-W

- Full Rep. configuration
  - TL2 & Non-Voting
- Partial Rep. configuration
  - MVCC & SCORe [PRQ12]
- JGroups

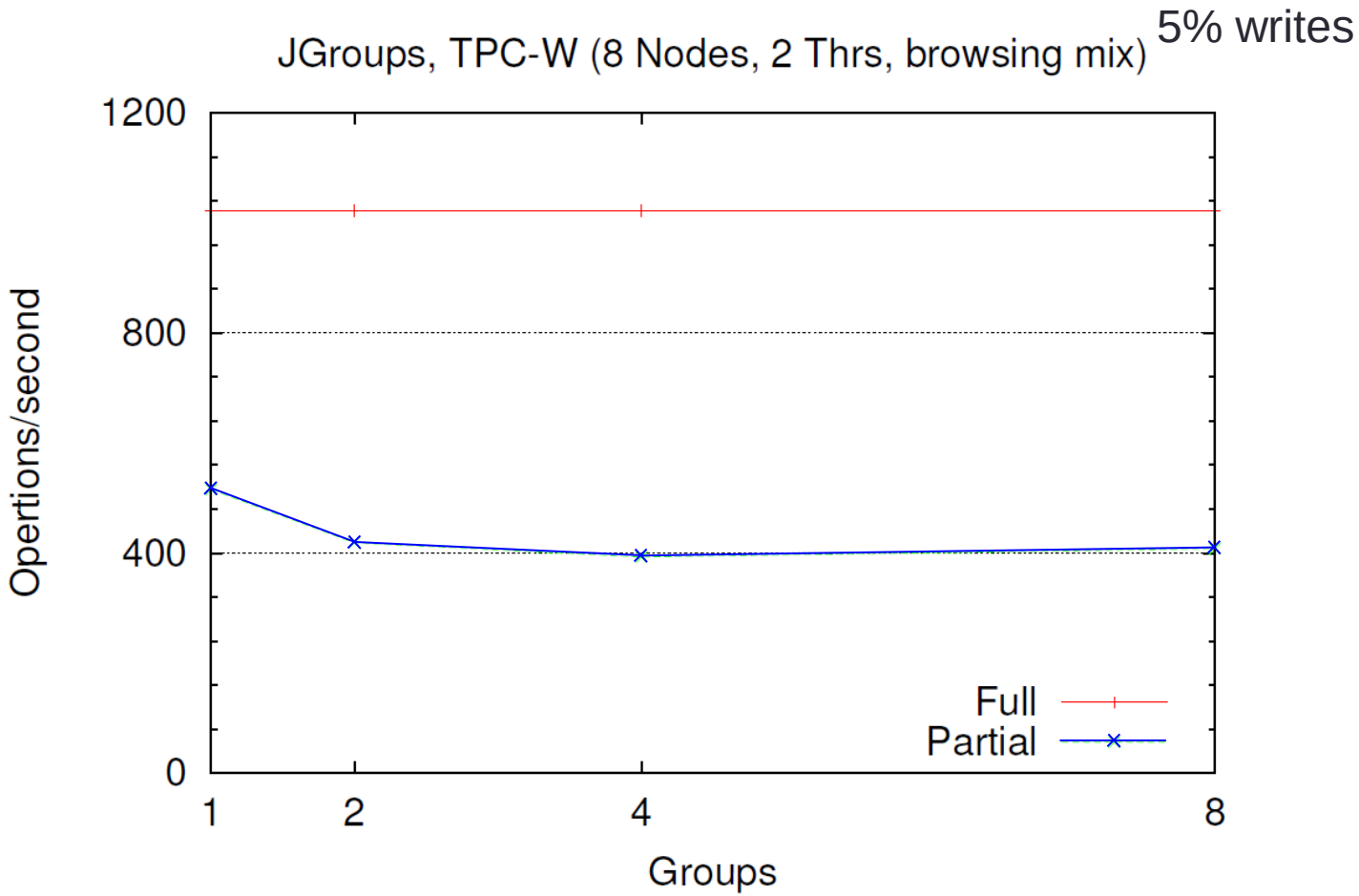
Experimental Results

- Does it make an efficient use of memory?



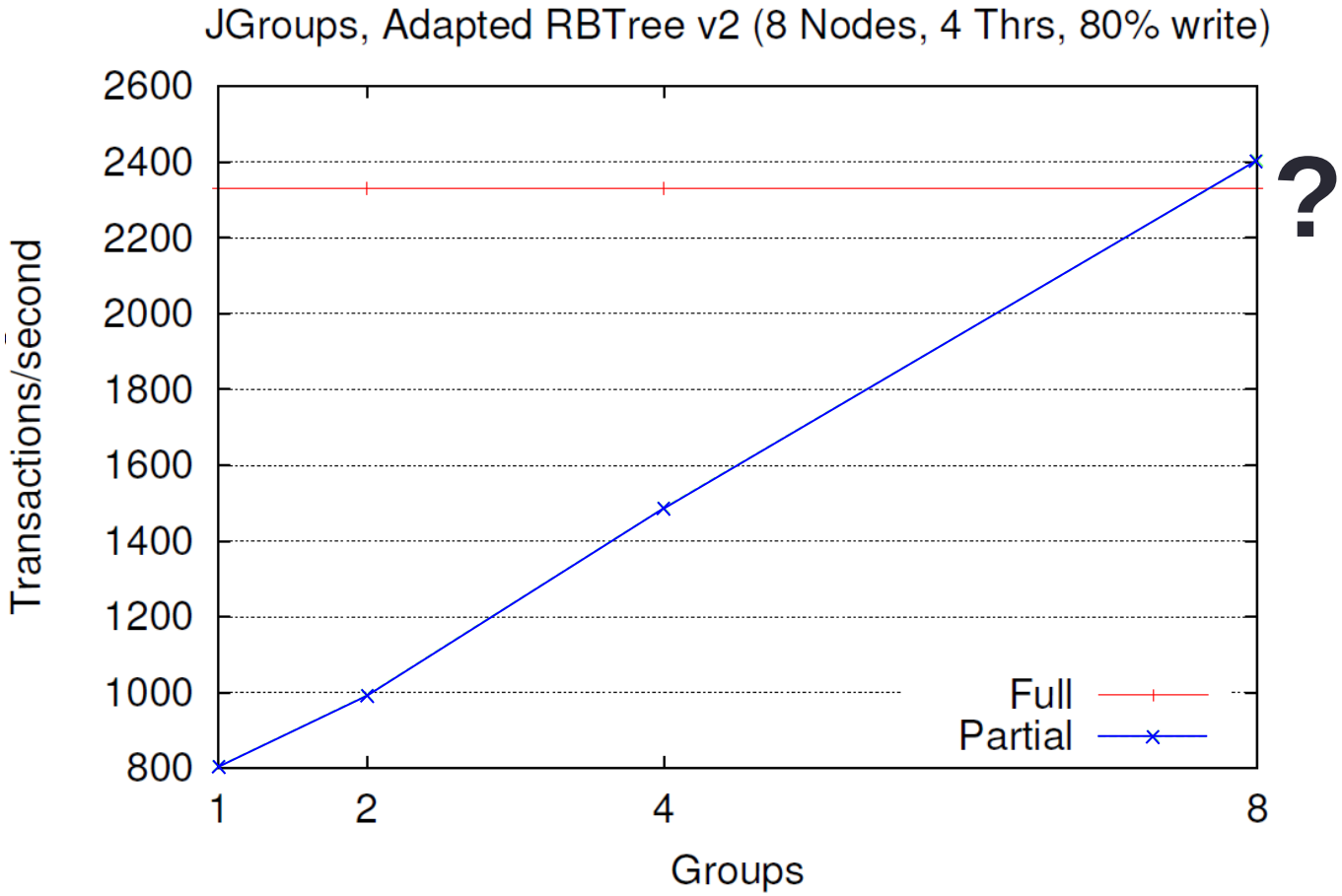
Experimental Results

- How does it compare with full replication?



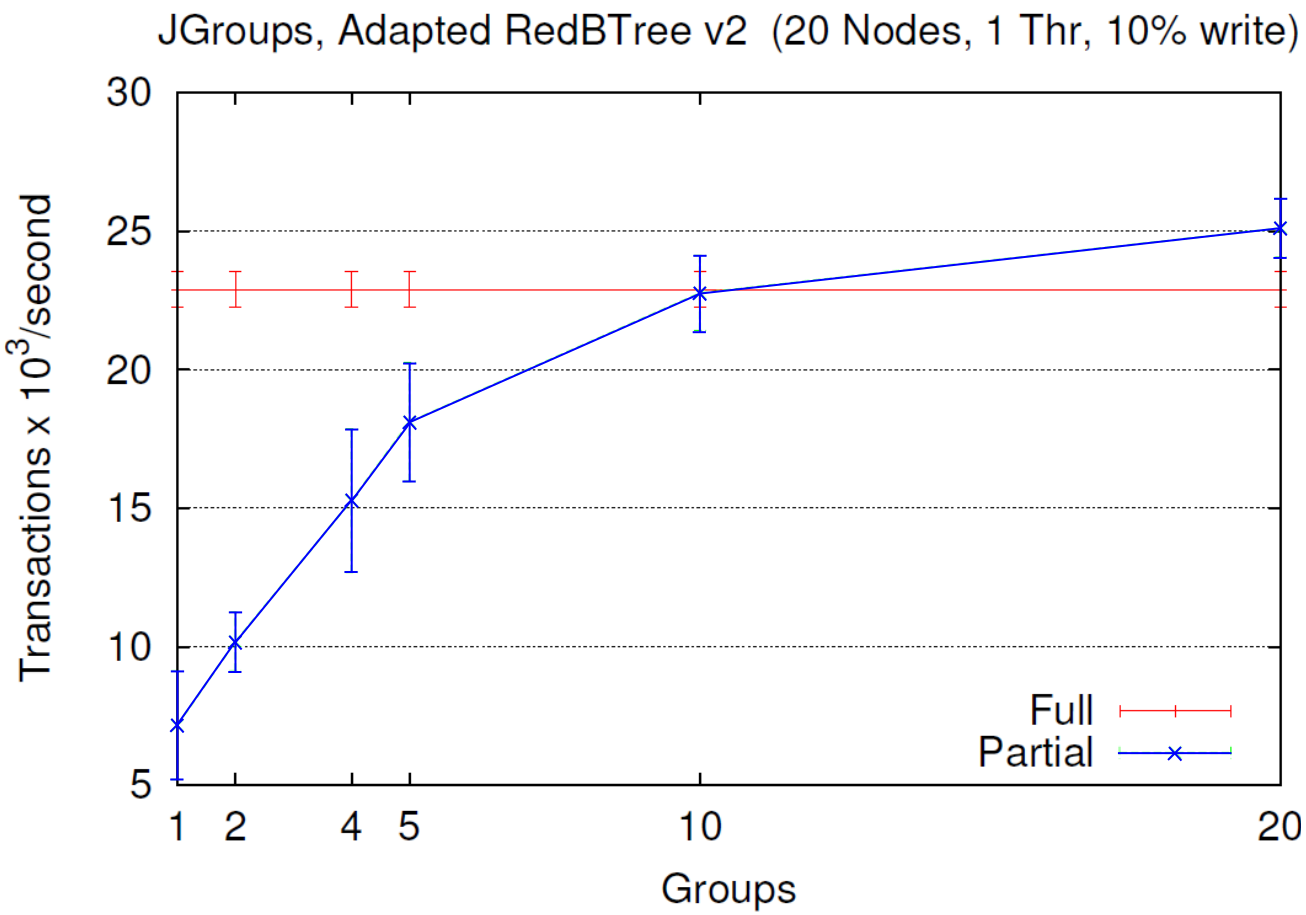
Experimental Results

- How does it compare with full replication?



Experimental Results

- How does it compare with full replication?



Conclusions

- Modular DTM framework with support for a combination of full and partial data replication
  - Flexible programming model
  - Allows different implementations of its components
- Applicability of partial data replication in DTM
  - Reduces memory usage
  - Gives evidence of scalability with the number of nodes
  - Better fit for transactions that modify partially replicated data

Future/Ongoing Work

- Extensive comprehensive evaluation
- Cache of remote objects
- Framework optimizations
- Protocol for seamless combination of partial and full data replication

Thank you for your attention

References

[CRR11] N. Carvalho, P. Romano, and L. Rodrigues. “A Generic Framework for Replicated Software Transactional Memories”. In: IEEE NCA. 2011.

[DVL12] R. J. Dias, T. M. Vale and J. M. Lourenço. “Efficient support for in-place metadata in transactional memory”. In: Euro-Par. 2012.

[KSF10] G. Korland, N. Shavit and P. Felber. “Deuce: Noninvasive software transactional memory in java”. In: Transactions on HiPEAC 2010.

[PRQ12] S. Peluso, P. Romano, and F. Quaglia. “SCORe: A Scalable One-Copy Serializable Partial Replication Protocol”. In: Middleware. 2012.

[Pel+12] S. Peluso, P. Ruivo, P. Romano, F. Quaglia, and L. Rodrigues. “When Scalability Meets Consistency: Genuine Multiversion Update-Serializable Partial Data Replication”. In: ICDCS. 2012.

[SR11] M. Saad and B. Ravindran. “HyFlow: a high performance distributed software transactional memory framework”. In: HPDC. 2011.

References

[SSP06] N. Schiper, R. Schmidt, and F. Pedone. “Optimistic Algorithms for Partial Database Replication”. In: Principles of Distributed Systems. 2006.

[SSP10] N. Schiper, P. Sutra, and F. Pedone. “P-Store: Genuine Partial Replication in Wide Area Networks”. In: IEEE SRDS. 2010.

[Ser+07] D. Serrano, M. Patiño-Martinez, R. Jimenez-Peris, and B. Kemme. “Boosting Database Replication Scalability through Partial Replication and 1-Copy-Snapshot-Isolation”. In: IEEE PRDC. 2007.

[Sou+01] A. Sousa, R. Oliveira, F. Moura, and F. Pedone. “Partial Replication in the Database State Machine”. In. IEEE NCA. 2001.

[VDL12] T. Vale, R. Dias, and J. Lourenço. “Uma Infraestrutura para Suporte de Memória Transacional Distribuída”. In: INForum. 2012.