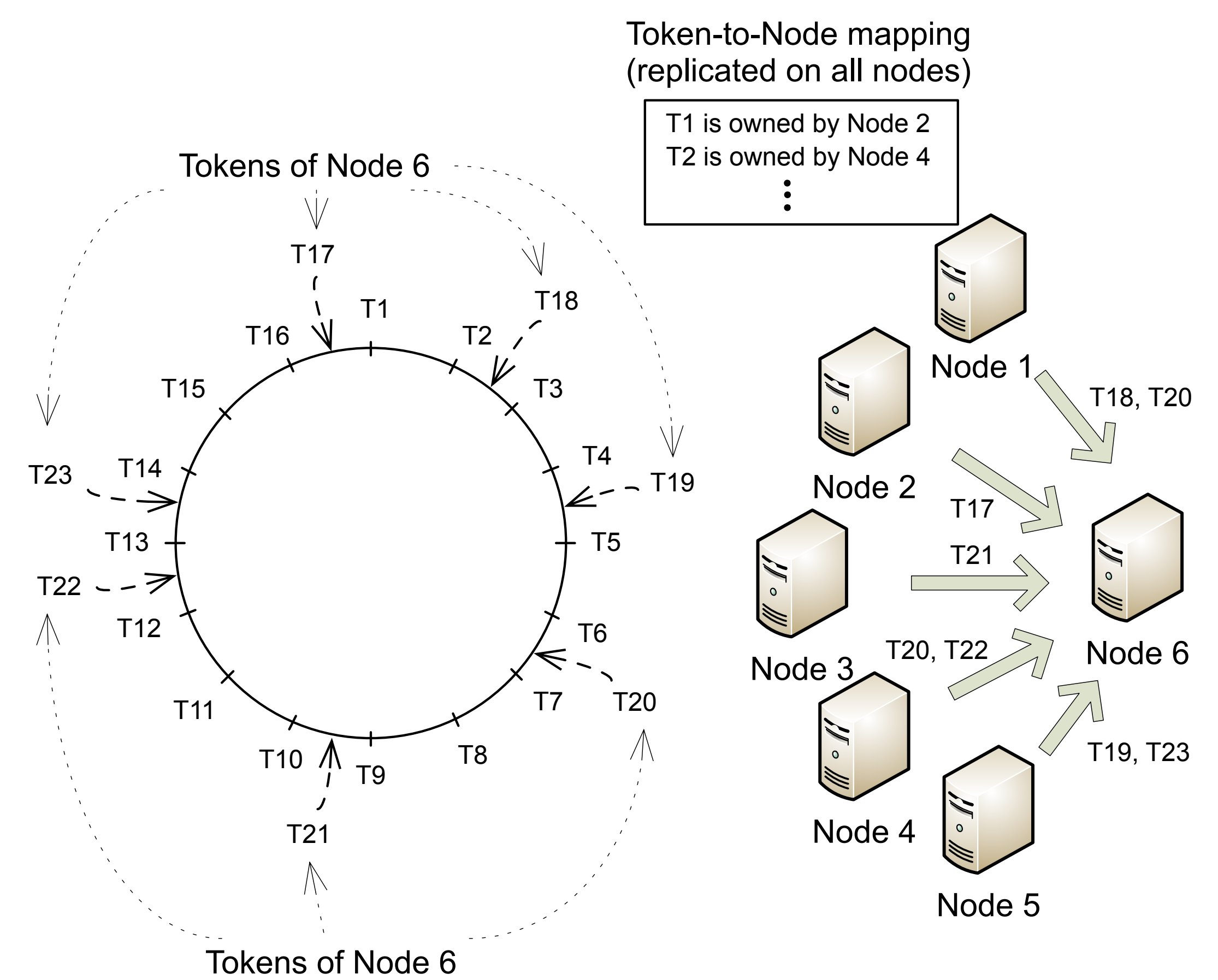


Motivation

Problem Statement

- Elasticity in NoSQL data stores is used to adapt to workload variations
- Most data stores typically perform parallel data transfers to new (joining) node
 - New node must receive data from all existing nodes for uniform data distribution
- All nodes are simultaneously reducing their processing capacity during data transfer
- New node capacity available only at the end of the elasticity action
- Significant I/O activity at the new node after data transfers complete
 - Multiple compactions, cache misses
- Many-to-one communication pattern known to be a cause of throughput collapse



Incremental Elasticity Approach

- A new mechanism for scheduling data transfers during elasticity
- Senders take turns sending data to the new node
- As soon as a transfer is over, data become available for access on new node
- Processing capacity of new node (and overall system) gradually increasing

Incremental Elasticity Benefits

- Processing capacity increases in a stepwise incremental fashion
- Results to smoother elasticity action
 - Fewer nodes involved in network transfer at any time
 - New node contributes to the cluster capacity during elasticity
- Overhead of streaming nodes may be masked by other replicas

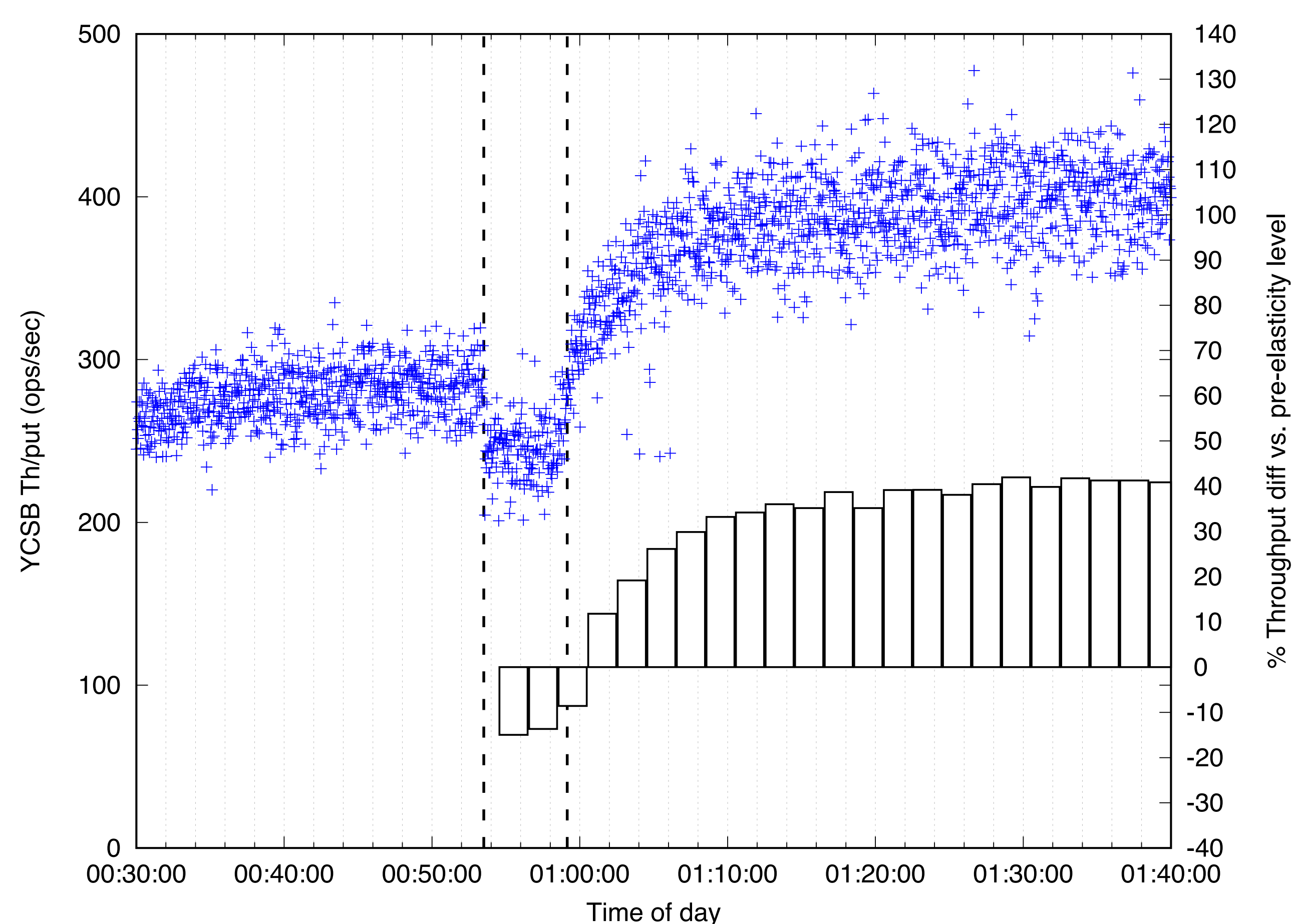
Evaluation

Testbed

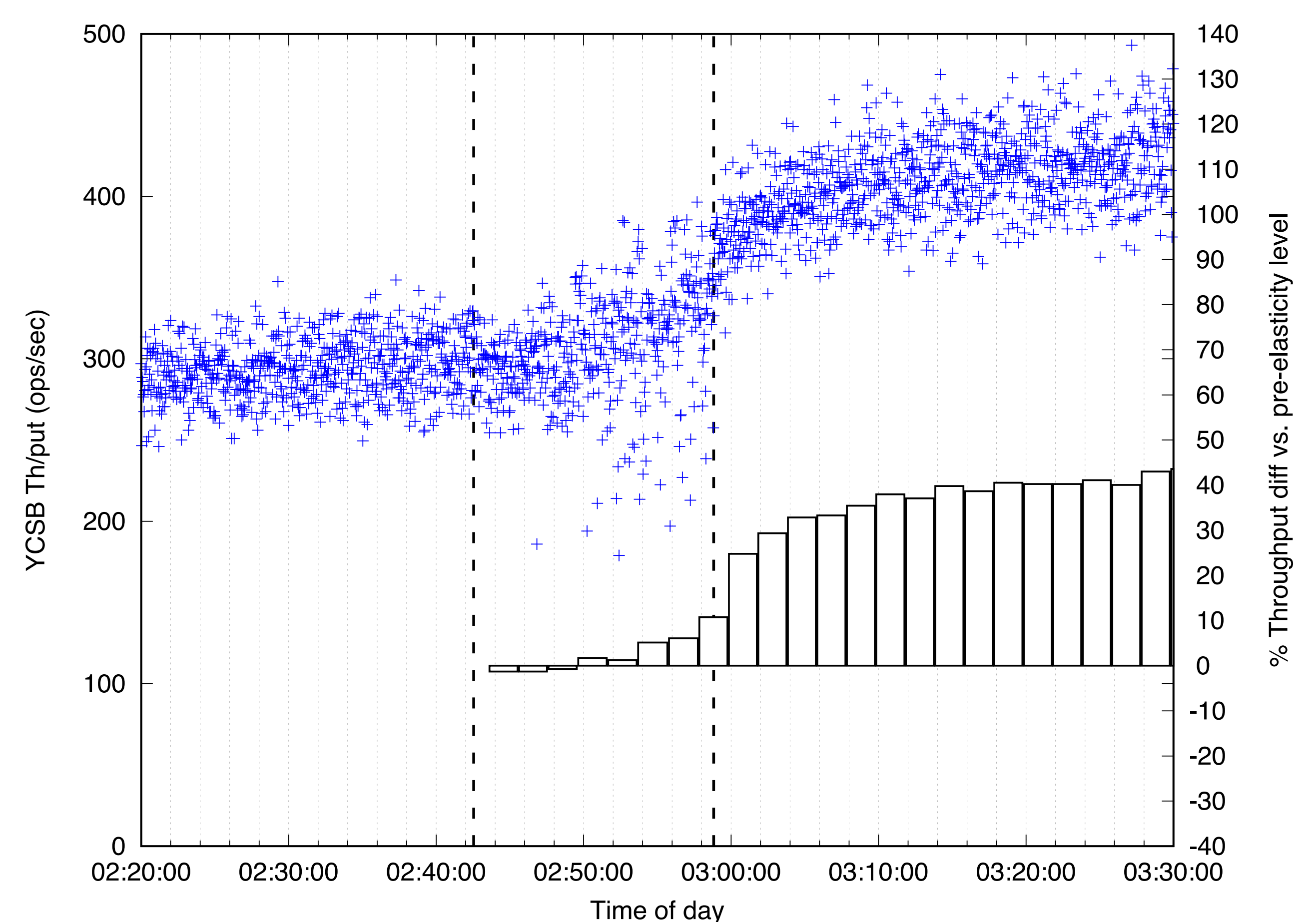
- 7 AMD dual-core Opteron 275 servers, 12 GB memory
- Cassandra cluster starts with 5 servers, 6th server joins during elasticity action
- 1 client on dedicated server

- Yahoo! Cloud Serving Benchmark (YCSB)
- 10 threads
- Workload: 95% reads / 5% writes, uniform key distribution

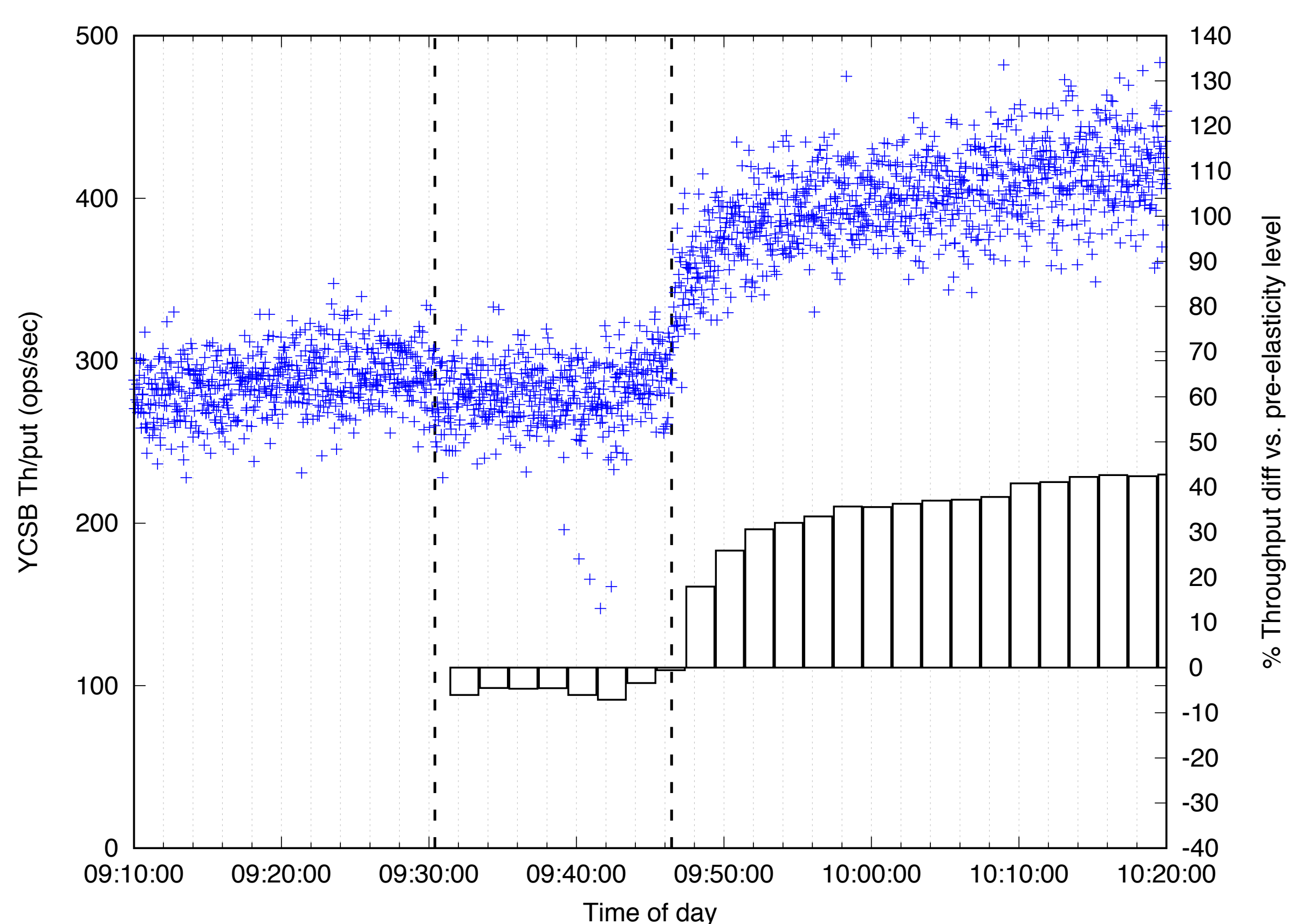
Parallel streaming



Incremental streaming



Parallel streaming throttled (36 Mbps)



Throughput of joining node

