CumuloNimbo Final Review
Brussels 27/11/13

FP7-257993

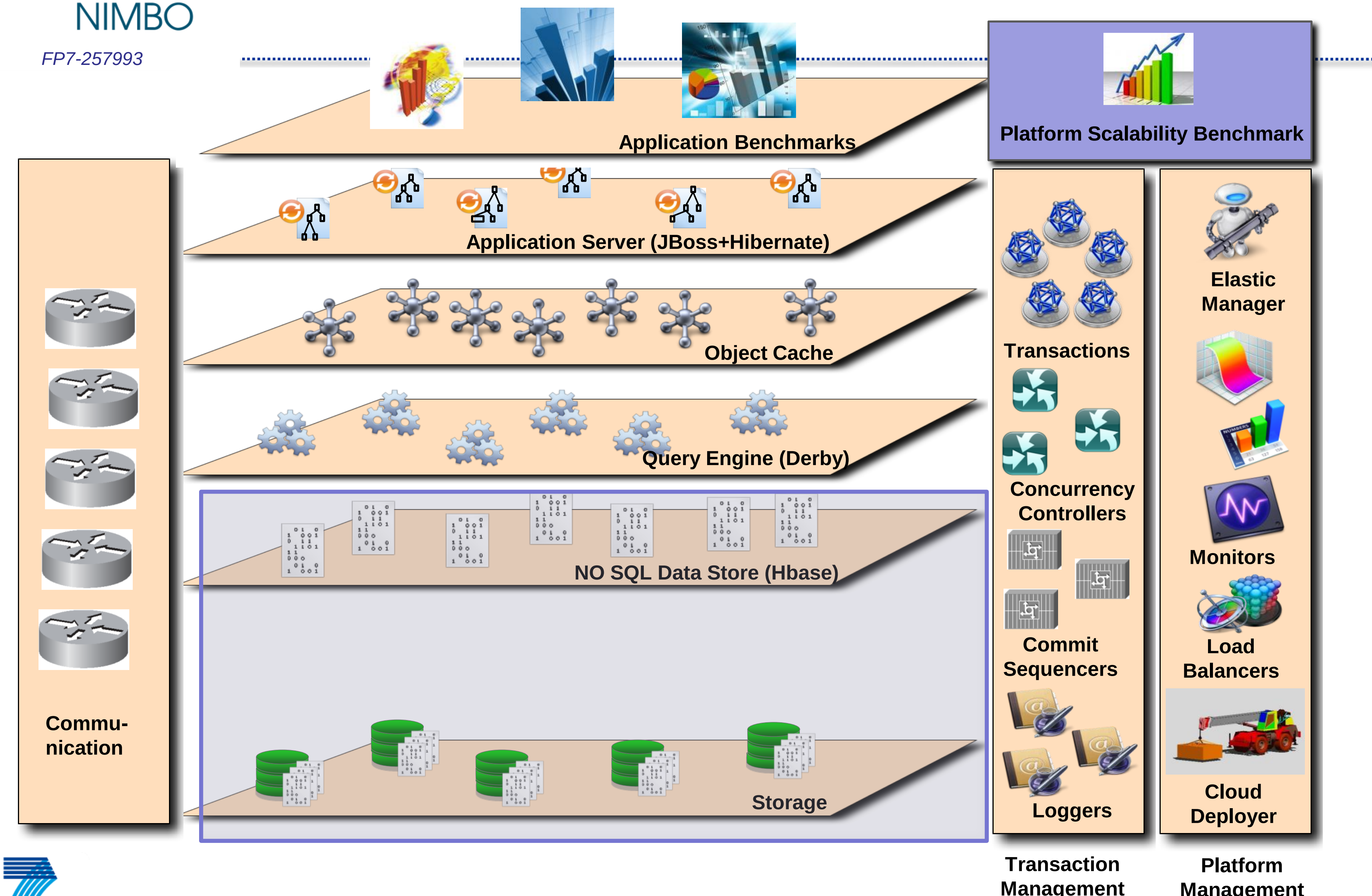
Cumulonimbo storage and communication infrastructure

Kostas Magoutis, ICS-FORTH

SEVENTH FRAMEWORK PROGRAMME

Positioning with Respect to the Project

FP7-257993



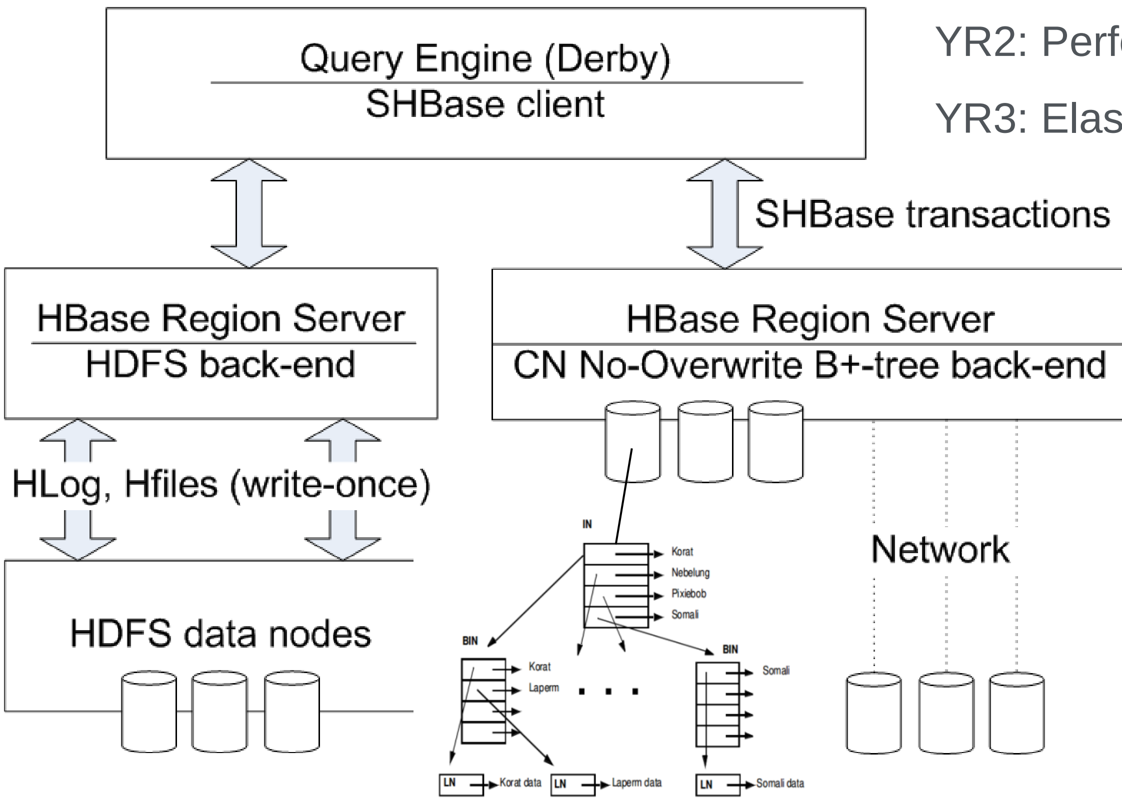
Advancement with respect to SOTA

- Cumulonimbo storage infrastructure
  - Novel B+-tree based key-value store (HBase-BDB) that outperforms HBase, especially in
    - Read/write workloads
    - Update intensive workloads
  - Effective support for elasticity, placing minimal impact on application performance

- Cumulonimbo communication infrastructure
  - Novel network storage protocol (Tyche) transparently multiplexes network traffic over several 10Gbps links
  - Tyche achieves excellent performance:
    - Reads: up to 6.2 GBytes/s (~7 max)
    - Writes: up to 6.7 GBytes/s (~7 max)

FP7-257993

HBase-BDB: B+ tree indexed regions

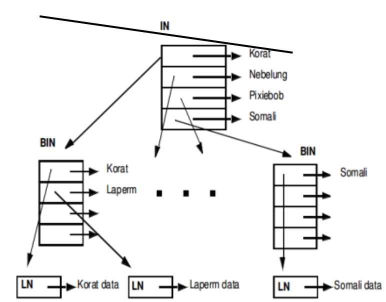


(a) Standard HBase

(b) CumuloNimbo architecture

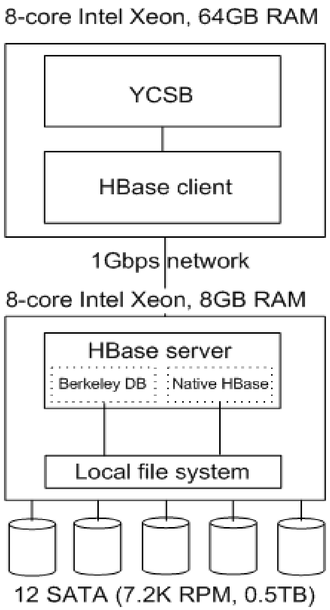
YR2: Performance evaluation, integration

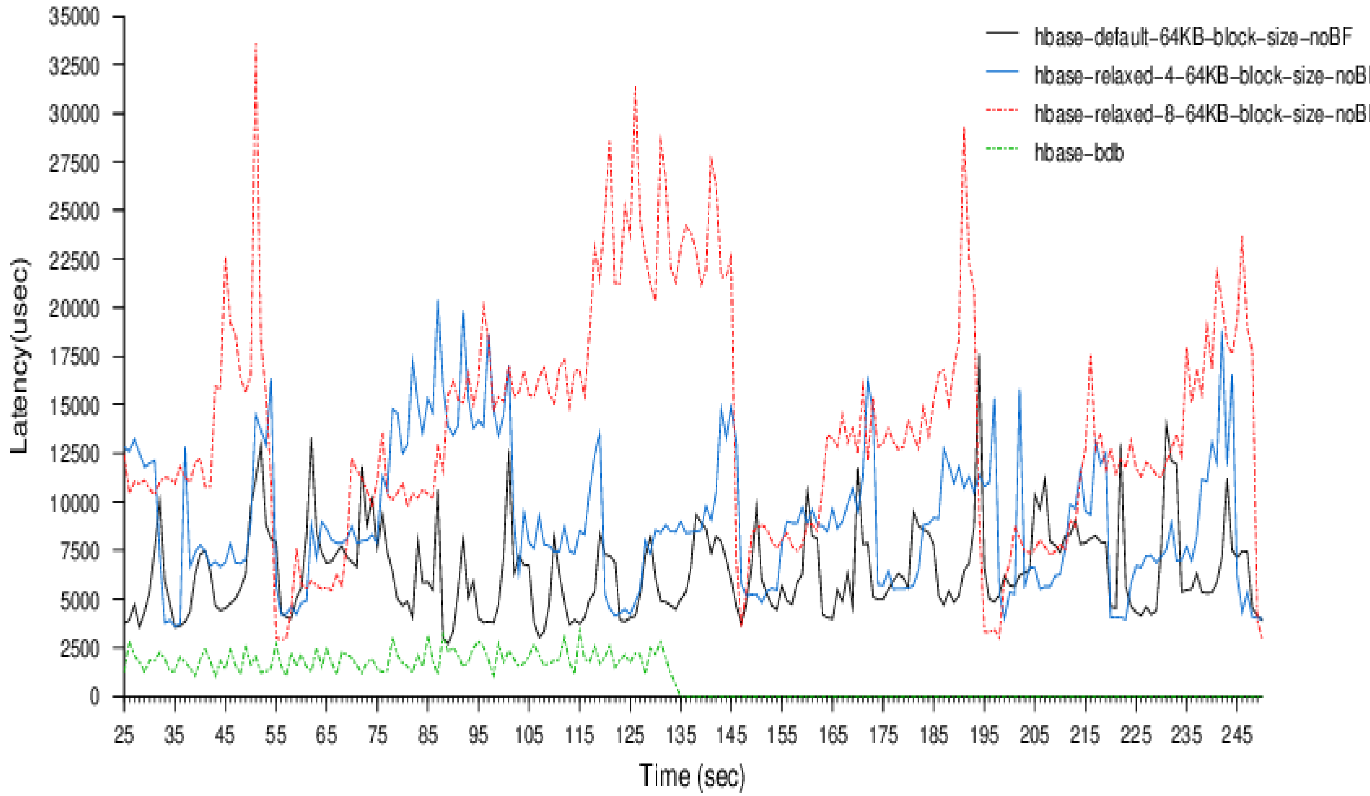
YR3: Elasticity support, more integration



Latency of read operations
30%-read 70%-update 500K records

FP7-257993

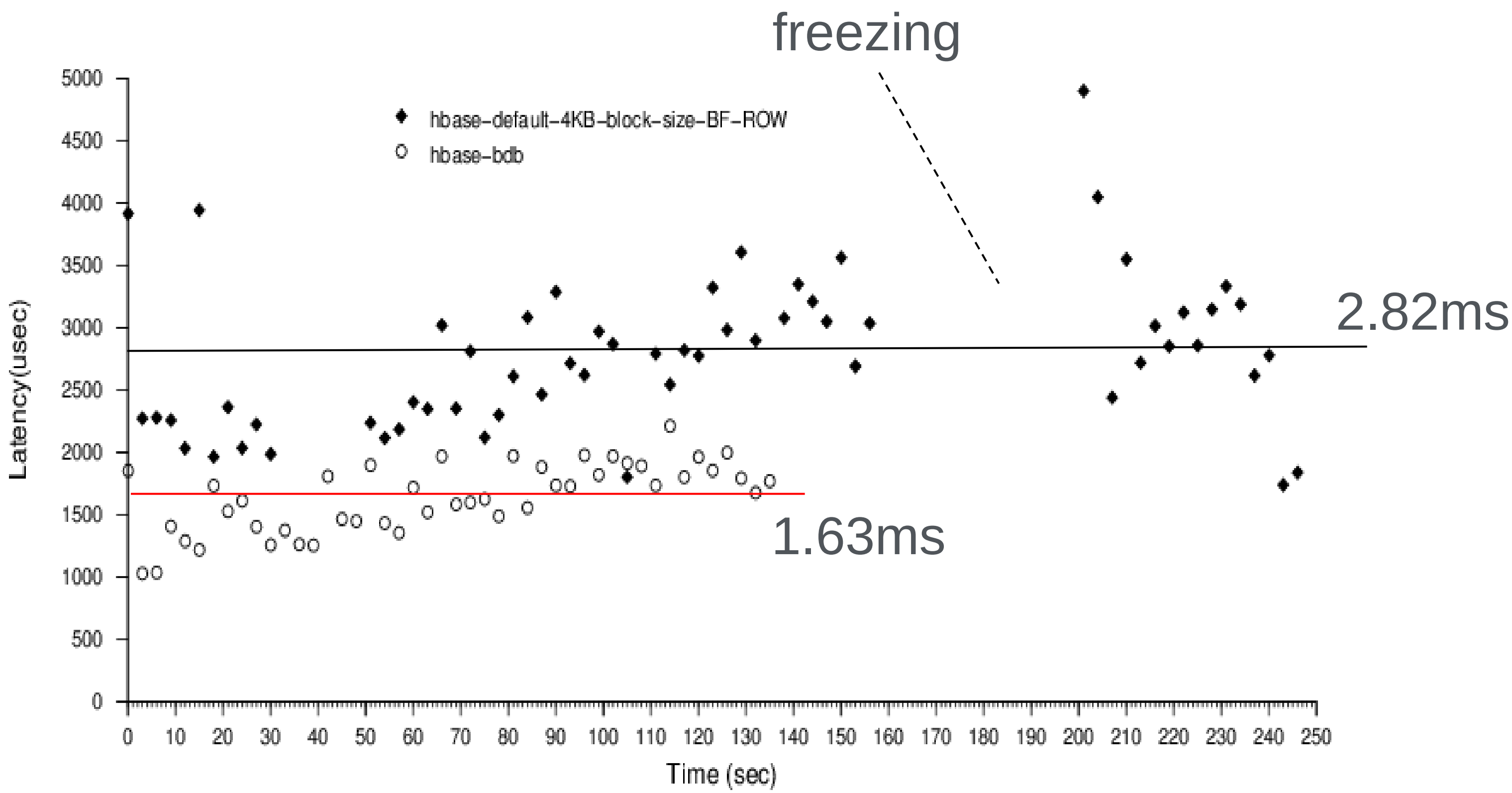




- Different compaction frequencies
- HBase block size: 64KB

Latency of read operations
30%-read 70%-update workload 500K records

FP7-257993



- HBase uses 4KB block size
- Bloom Filter on rows

Conclusions

- HBase-BDB outperforms native HBase
  - HBase-BDB has lower latency, more stable performance
- Important to correctly tune native HBase server
  - Block size
  - Bloom filters
  - Compaction and GC activity

CUMULO NIMBO

FP7-257993

Elasticity: What HBase does

- Regions reaching a maximum size are split
  - Initially virtually split, daughter regions map to parent HFiles
  - Future compactions create the new HFiles
- Load balancer migrates region to spread capacity
  - Move operation is easy, leverages distributed file system
  - Data movement is eventually performed by HDFS

Elasticity: HBase-BDB

- Region split: avoid eager physical copies
- Region move: At network speed, minimal blocking

HBase–BDB elasticity: Split

- Split copies region (A) into a daughter-region (B)
  - Region “mid-point” maintained throughout
  - Split command realized as BTRFS copy operation on A’s underlying BDB log files

- When done copying to B, rename parent to reflect split
  - Delete half of all keys (different halves) from two regions
  - Actual deletion happens at BDB cleaning time

HBase–BDB elasticity: Move

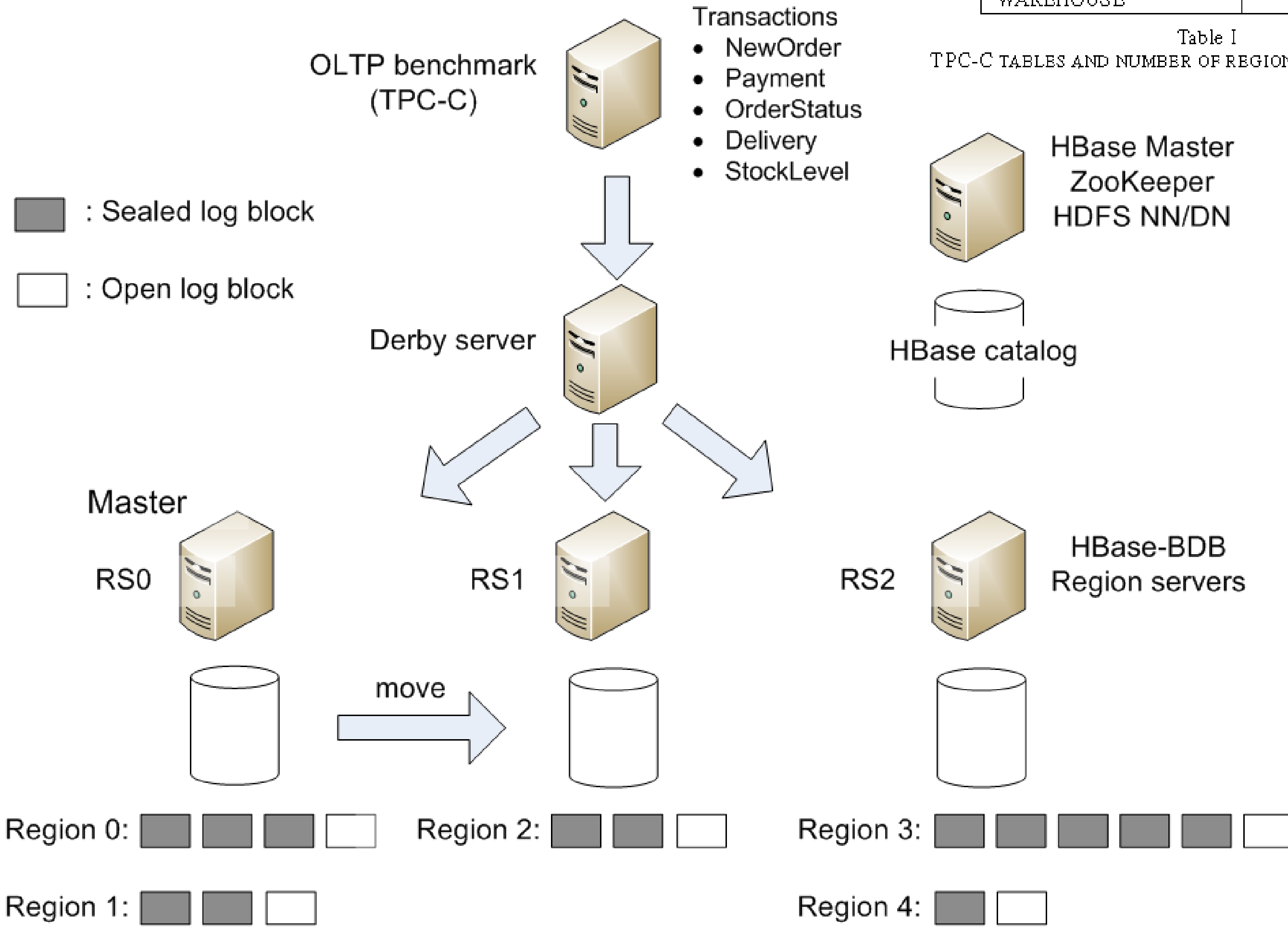
- Coordination between source and target region servers achieved via Zookeeper
- Multiple parallel TCP transfers for region's BDB log files
  - Utilize multiple cores, links
- Transfer immutable content first, without closing region
- Close region and transfer last (active) log
- When transfer complete, target region server opens region

Experimental setup

FP7-257993

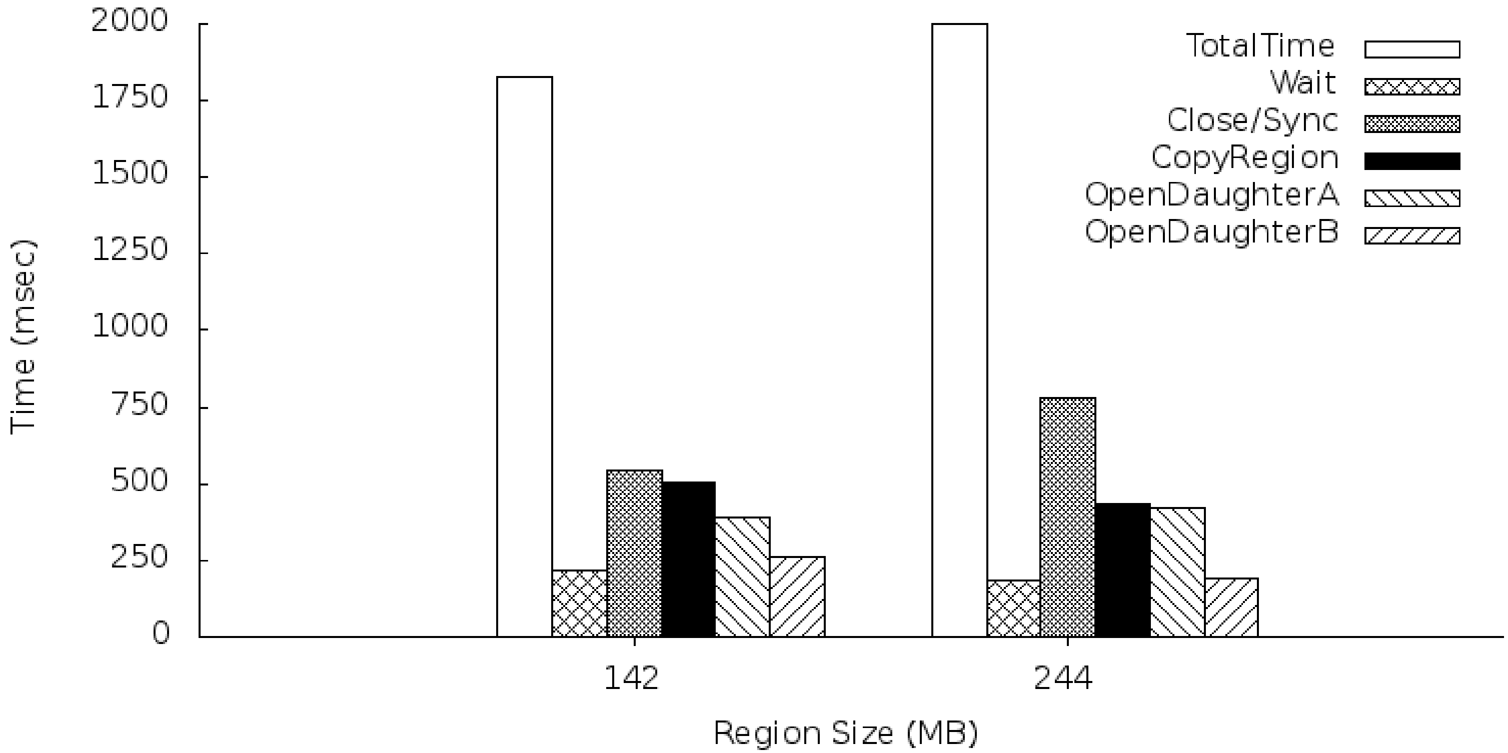
Table name	Number of regions
CUSTOMER	3
IDX_CUSTOMER_NAME	3
HISTORY	1
DISTRICT	1
ITEM	1
NEW_ORDER	1
OORDER	3
ORDER_LINE	25
STOCK	9
WAREHOUSE	1

Table I
TPC-C TABLES AND NUMBER OF REGIONS AFTER POPULATION



Cost of splits (during TPC-C population)

Database scale factor 8



- Splits last about 2 seconds (without any aggressive optimizations)

Cost of moves (no TPC-C client load)

Region-move stage	Time (sec)	Std. dev (sec)
Prefetch (144MB)	2.5	0.16
Open (15MB)	1.54	0.7

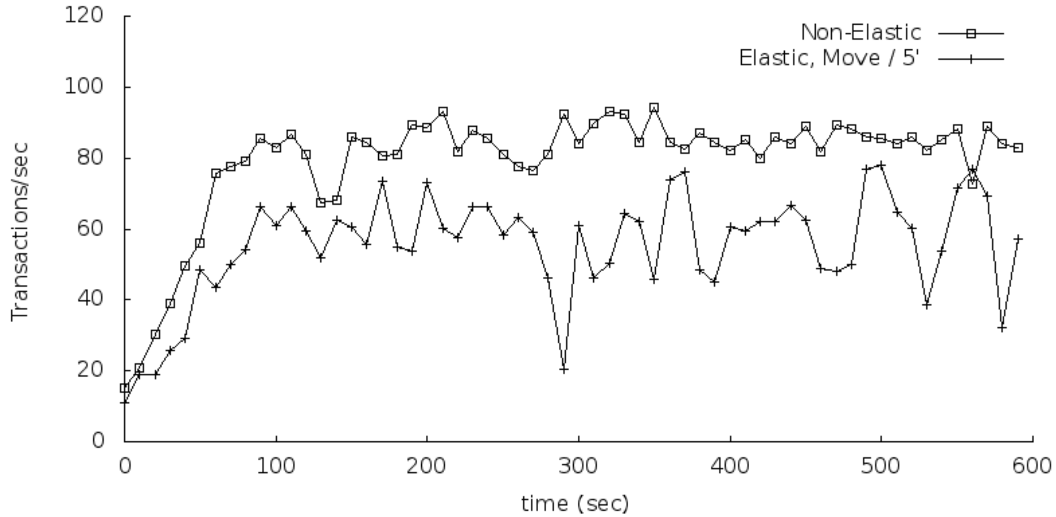
Table II
MOVE OPERATION TIME BREAKDOWN

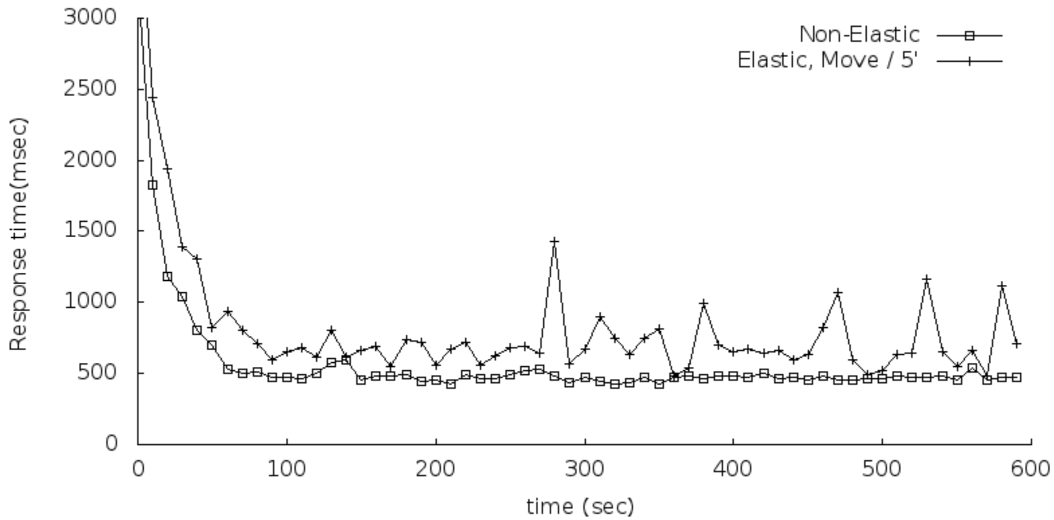
- Moves last about 4 seconds (without any aggressive optimizations)

- ~60MB/s out of ~70MB/s possible (iperf) during prefetch stage

Cost of region moves

FP7-257993





- *40 client threads*
- *Moving regions of CUSTOMER table (hottest)*

TPC-C throughput

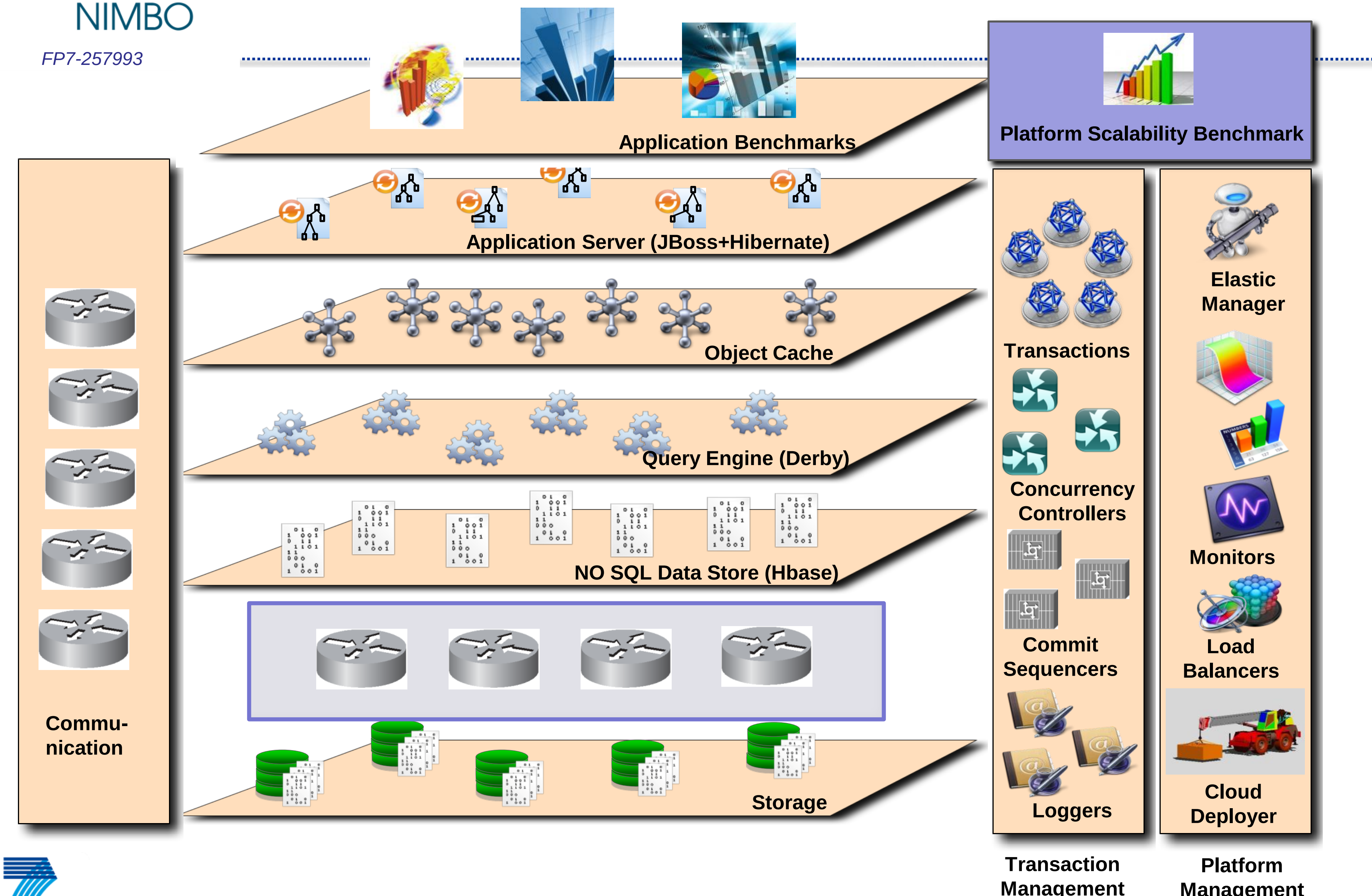
Transaction mix
- 45% NewOrder
- 43% Payment
- 4% StockLevel
- 4% OrderStatus
- 4% Delivery

TPC-C response time

Conclusions

- Designed, implemented, integrated, and evaluated B+-tree based storage backend to HBase
  - Improved, more stable performance over standard HBase
- Designed, implemented, integrated, and evaluated elasticity architecture of HBase-BDB
  - Effective elasticity with minimal impact on performance

Positioning with Respect to the Project



CUMULO NIMBO

FP7-257993

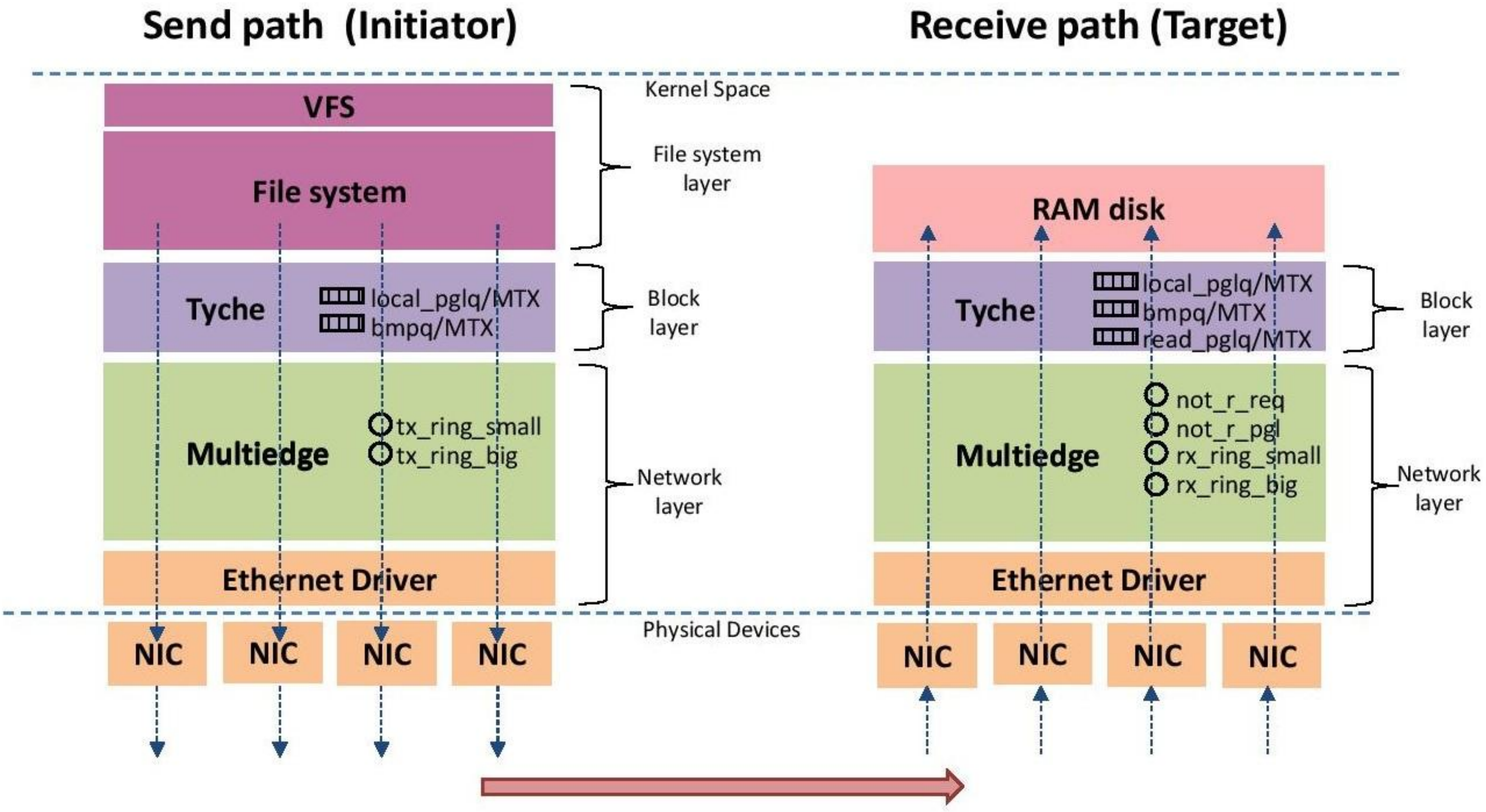
Motivation

- In Cloud environments, application/middleware servers are typically connected to storage over storage-area networks
- Targeted workloads are I/O intensive
  - I/O likely to become bottleneck
  - Need efficient network storage protocols
  - Current protocols (iSCSI, NBD) do not scale to multi-GB/s hardware speeds

Advancement with respect to SOTA

- Novel networked storage protocol: Tyche
  - Transparently use multiple NICs and many logical connections
  - Addressed contention, memory mgmt, and network ordering
  - Considered elasticity aspects and NUMA affinity
- Achieve scalable throughput close to hardware limits
  - Reads: up to 6.2 GBytes/s (~7 max)
  - Writes: up to 6.7 GBytes/s (~7 max)
- Significantly outperform NBD and the vanilla TCP/IP sockets

Communication subsystem



Contributions

- Independent of the file system and storage device
- Allow concurrency and elasticity
  - Several NICs simultaneously (tested up to 6 NICs), adaptively
- Reduce synchronization
  - Optimize each lock for the specific purpose it is used: 3x
- Memory management overhead
  - Avoid all dynamic memory operations in the common path
- Efficiently map I/O operations to network messages
  - Use storage protocol semantics to reduce packet overhead
  - Reduce copies to minimal for commodity Ethernet: 2x
- Perform NUMA affinity
  - Achieve perfect affinity in I/O path: 2x

Experimental environment

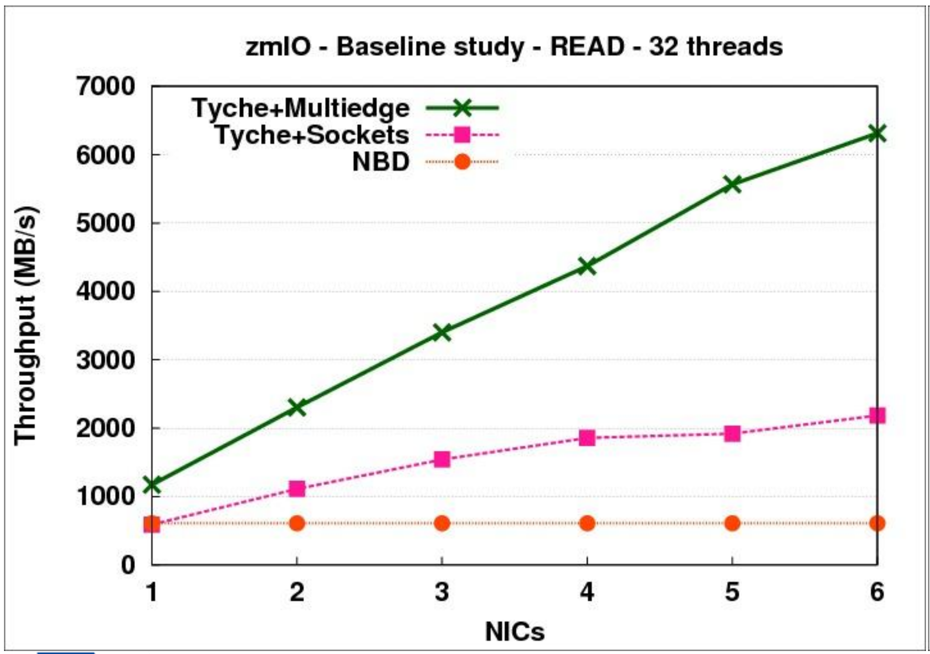
- Two systems back to back with 6x Myrinet 10GE cards
  - 8-core/16-thread Intel Xeon E5520 @2.7GHz
  - Initiator: 12 GB RAM
  - Target: 48 GB RAM, 36 GB used as ramdisk
  - OS: CentOS 6.3, Linux kernel 2.6.32 + XFS
- Benchmarks
  - zmIO, Hbase+BDB+YCSB, Indexer, Blast, TPCC
- Tyche (CumuloNimbo) compared to:
  - Linux Network Block Device – NBD (today)
  - Tyche + sockets

FP7-257993

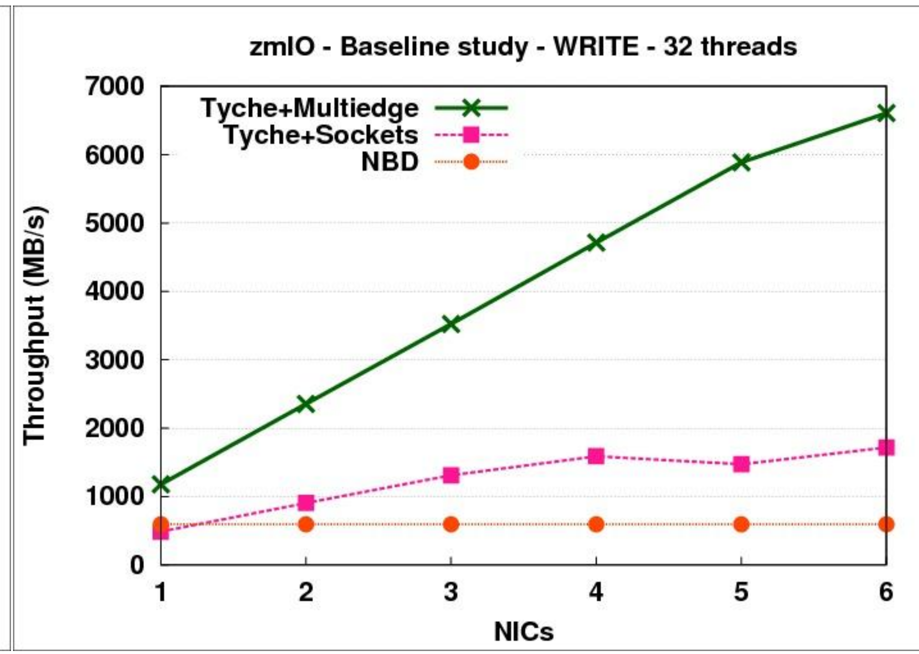
zmIO: Throughput at the Initiator node

- 32 threads, raw device (no file system), 1MB request size
- Tyche outperforms NBD by up to 10x
- Tyche outperforms the version with TPC/IP Sockets up to 3.8x

Sequential reads



Sequential writes

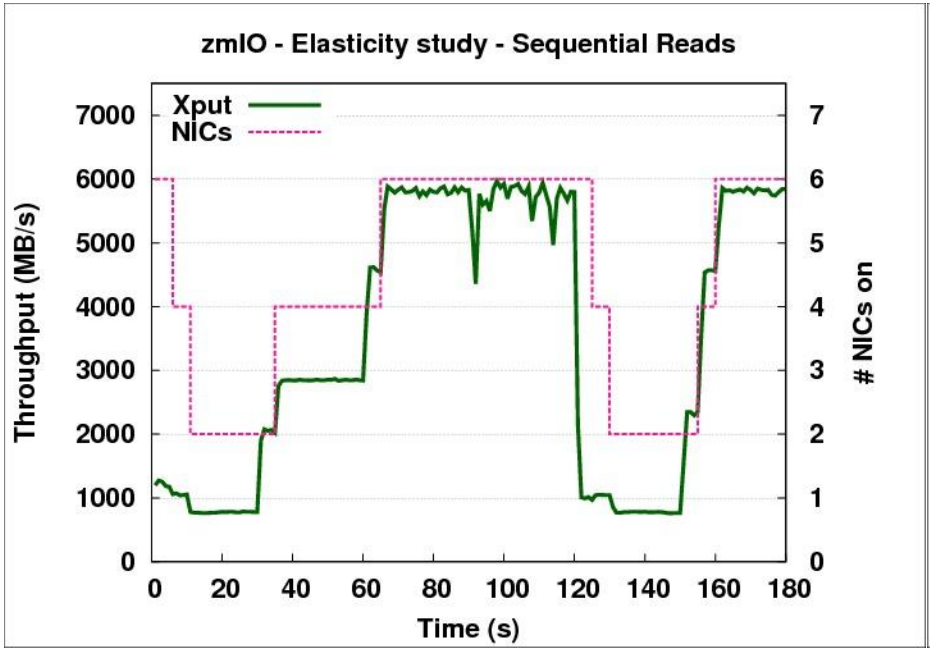


Elasticity behavior: Throughput at Initiator

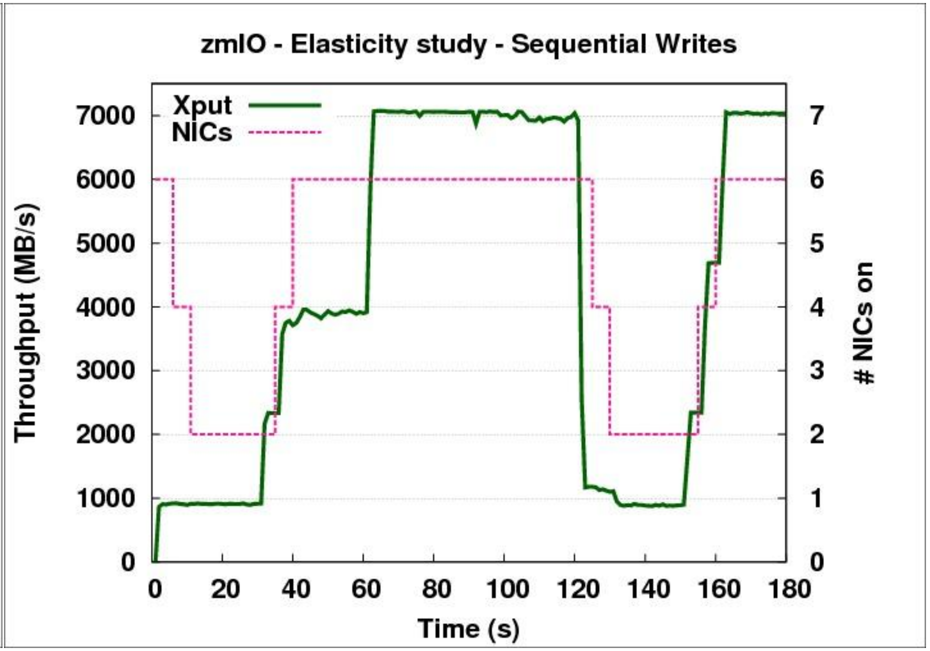
FP7-257993

- zmIO, 32 threads, raw device
- Request sizes: 4kB, 16kB, 1MB, 64kB, 4kB and 1MB
- Initially 6 NICs on, depending on throughput they are turned off/on

Sequential reads



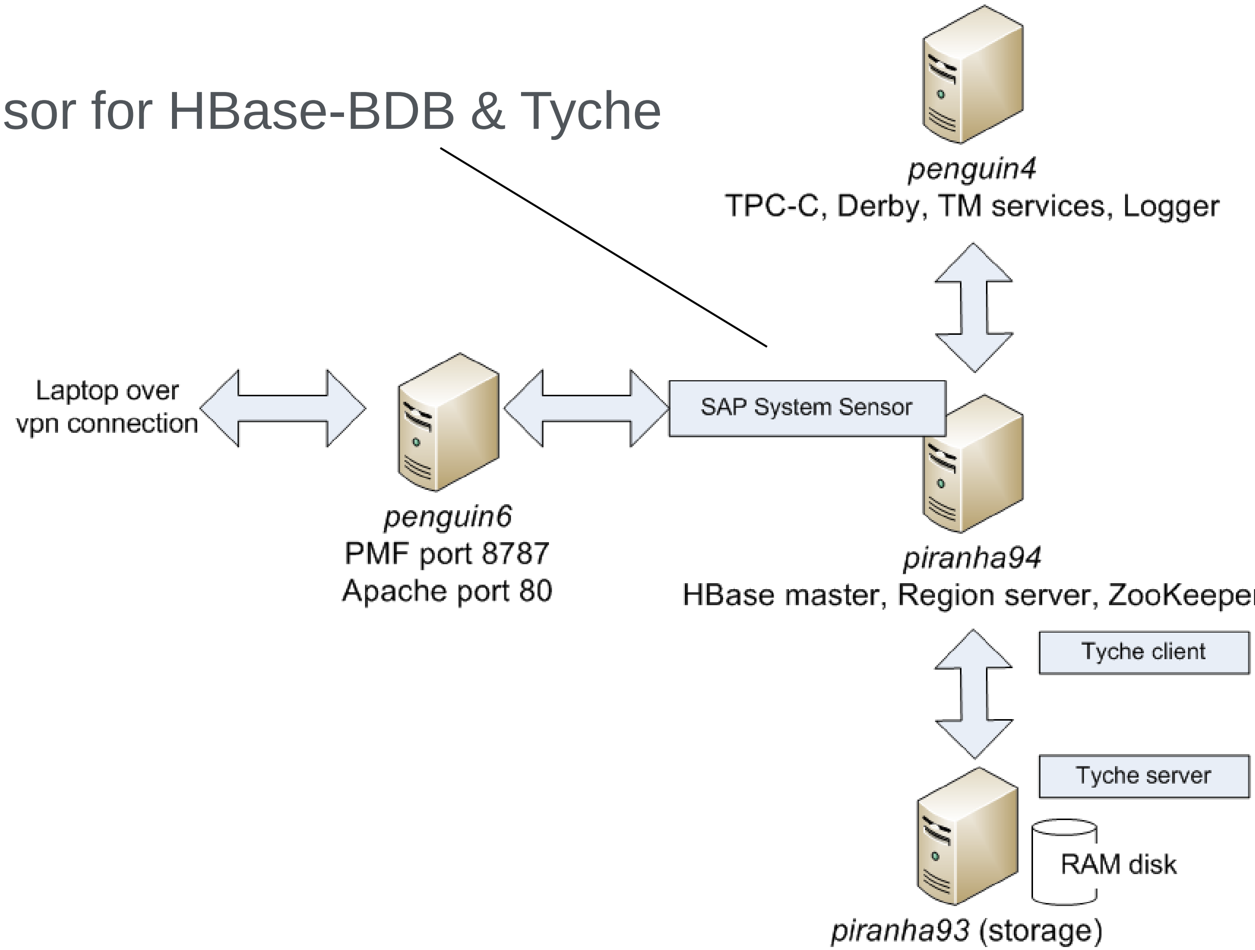
Sequential writes



Integration with PMF

FP7-257993

SAP sensor for HBase-BDB & Tyche



Summary

- Tyche: Novel networked storage protocol
- Transparently use multiple NICs and many logical connections
- Address contention, memory mgmt, and network ordering
- Consider elasticity aspects and NUMA affinity
- Achieve scalable throughput
  - Reads: up to 6.2 GBytes/s (~7 max)
  - Writes: up to 6.7 GBytes/s (~7 max)
- Significantly outperform NBD and the vanilla TCP/IP sockets
- Data-centres will need to use similar technology for improving efficiency, especially with trends towards converged storage