Hypertable Goes Realtime at Baidu

Yang Dong
yangdong01@baidu.com
Sherlock__Yang(http://weibo.com/u/2624357843)

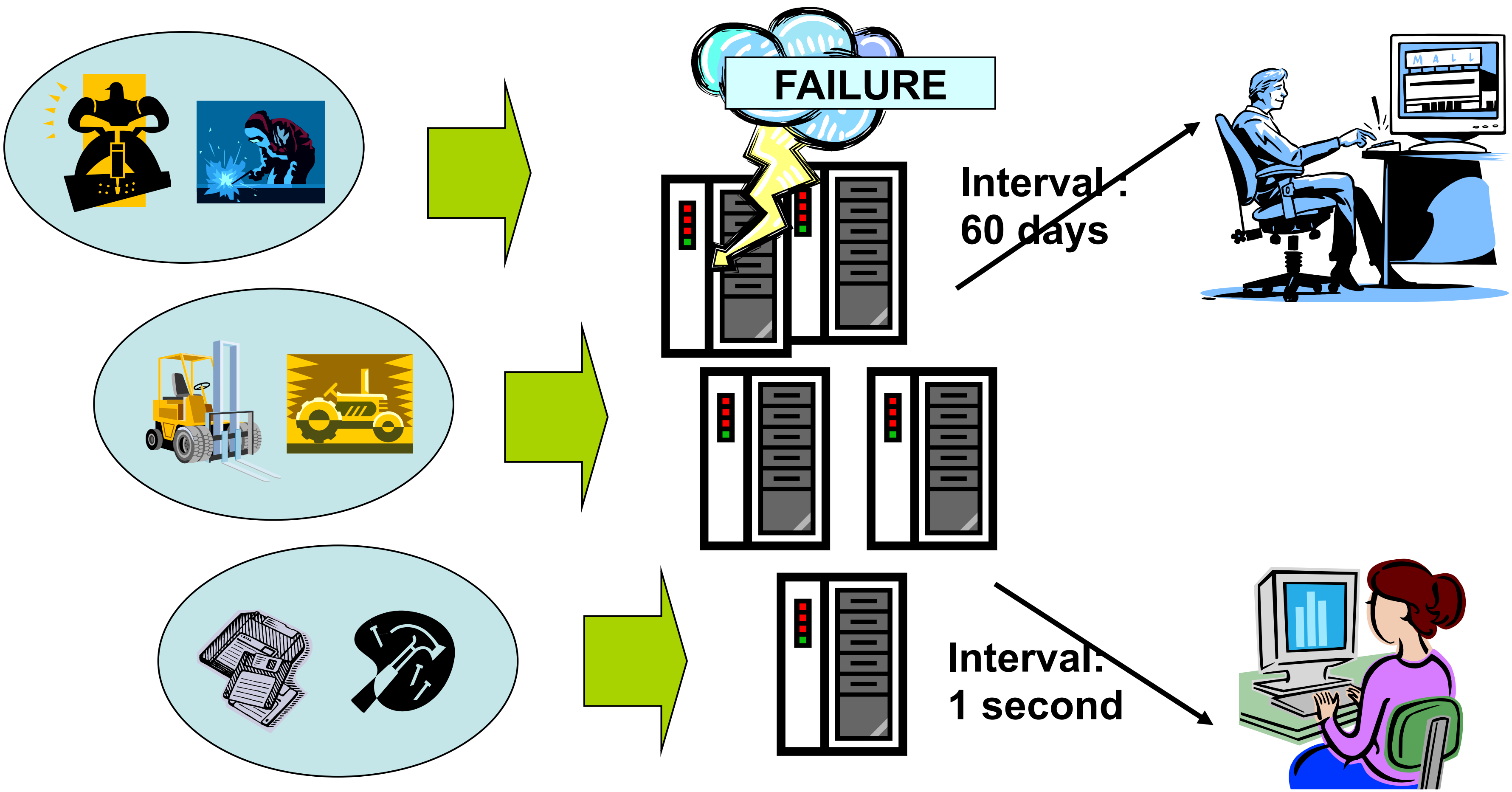
Agenda

- Motivation
- Related Work
- Model Design
- Evaluation
- Conclusion

Agenda

- Motivation
- Related Work
- Model Design
- Evaluation
- Conclusion

Motivation



What is Noah

- Noah System
  - Operational Data Store
    - System metrics (CPU, Memory, IO, Network)
    - Application metrics (Web, DB, Caches)
    - Baidu metrics (Usage, Revenue)
      - Easily visualize data over time
      - Supports complex aggregation, transformations, etc.
  - Component
    - **Monitoring sub-system**
    - Collection sub-system
    - ... ...

System Requirement

- Storage Capacity
  - 10TB~100TB
  - 100~1000 billion Records
- Automatic Sharding
  - Irregular data growth patterns
- Heavy Writes
  - 10000~30000 inserts/s
- Fast Reads of Recent Data
  - Recently written data should be available quickly
- Table Scans
  - e.g. The entire dataset will also be periodically scanned in order to perform time-based rollups

Problems of Existing Stack

- MySQL
  - Not inherently distributed
    - Difficult to scale
      - Irregular data growth patterns
      - Manually re-sharding data frequently
  - Table size limitation
  - Inflexible schema
- Hadoop
  - No support for random writes
  - Poor support for random reads

Hypertable Meets the Requirements

- Hypertable + Hadoop
  - Elasticity
  - High write throughput
  - High Availability and Disaster Recovery
  - Fault Isolation
  - Range Scans

Typical Applications of Hypertable

- Common Library
  - AsyncComm/Compression/HQL … in Hypertable
- Database (Query Engine)
  - Hypertable
- MapReduce (Batch Processing)
  - MapReduce + Hypertable
- OLAP (Online Analytical Processing)
  - MySQL + Hypertable

Agenda

- Motivation
- Related Work
- Model Design
- Evaluation
- Conclusion

Bai du 百度

Related Work

- **Apache Hadoop Goes Realtime at Facebook**
  - *Titan* (Facebook Messages)
  - *Puma* (Facebook Insights)
  - *ODS* (Facebook Internal Metrics)

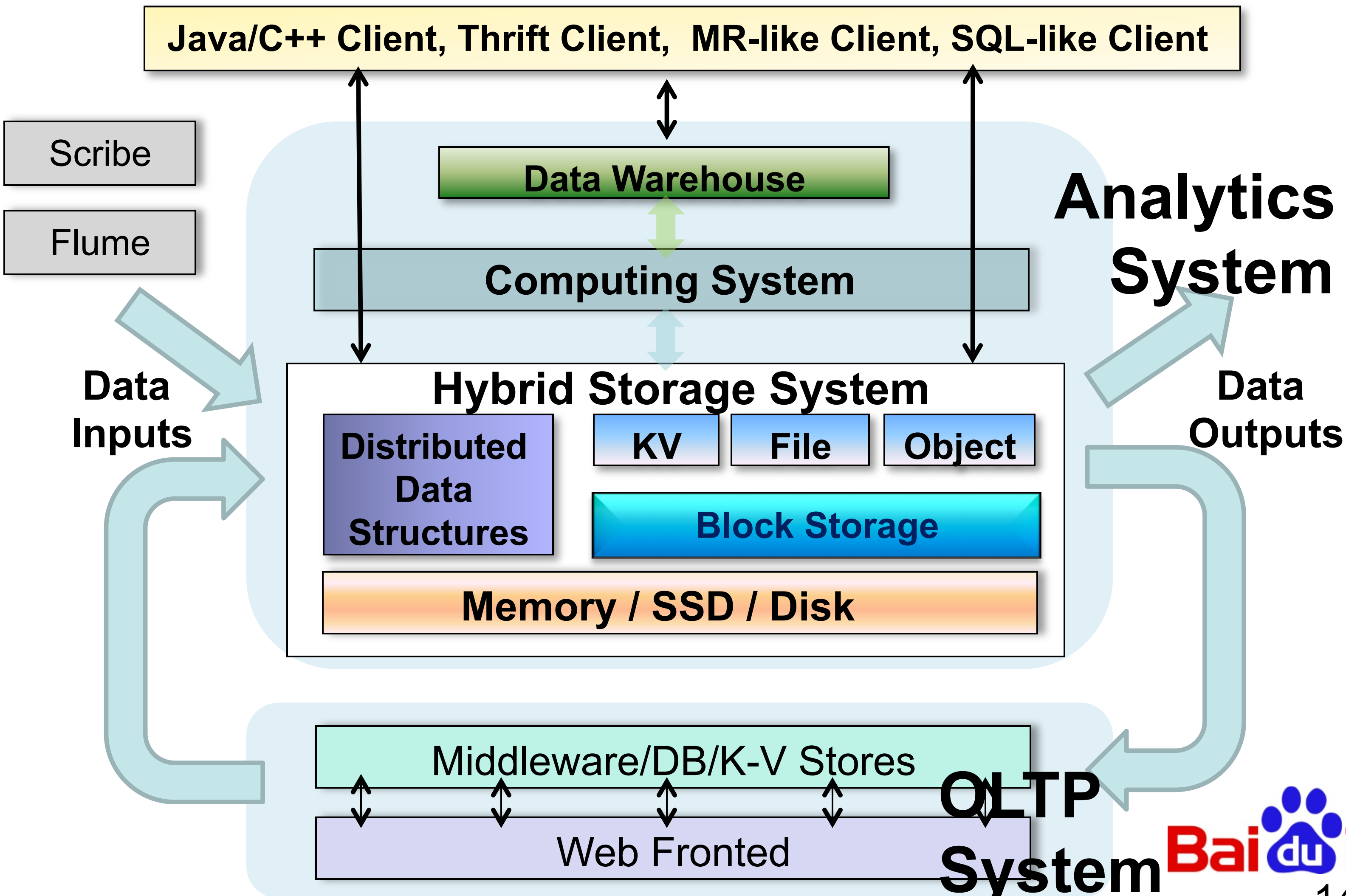
Agenda

- Motivation
- Related Work
- Model Design
- Evaluation
- Conclusion

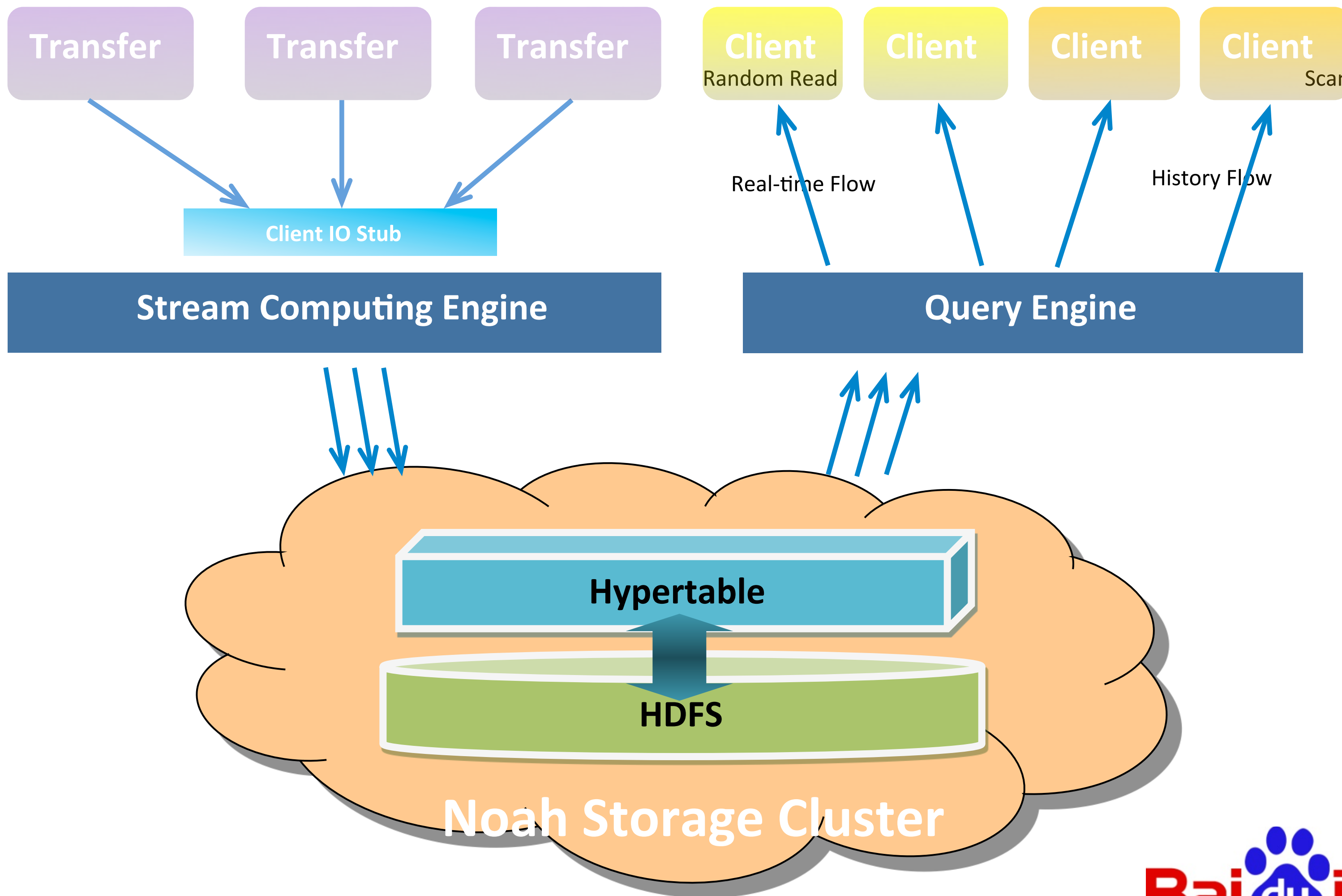
Model Design

- Storage System Architecture
- Functional Model
- Processing Model
- Data Model

Storage System Architecture



Functional Model



Processing Model

- Query Interface

  InfoQuery <table> <rowkey_range_start> <rowkey_range_end> <column_family> <to_file> <human|machine> [max_lines]

  - table: table name
  - rowkey_range_start & end: row key string range
  - column_family: monitor_data_status is ”value”, other tables are “value_pack”
  - to_file: file by result written into
  - human|machine: text or binary
  - max_lines: default < 2w

Processing Model

- Query Mode
  - Normal Query
    - Hostname + MonitorItemIDs + Interval -> Results Graph
    - Hostnames + MonitorItemIDs -> Sorted Latest Results
    - Hostnames -> Status Results
  - Special Query
    - Many Hosts + MonitorItemIDs + Interval: Multiple queries
    - Long Interval: default <= 2w Recods
    - Low Latency: No guarantee
    - Batch queries continuously: Influence to system

Data Model

- Tradeoff
  - Query efficiency
  - Record volume

Tables	Row Key	Column	Version	TTL
History Tables	Hostname + Timestamp	MonitorItem +MonitorItem*	1	0.5 ~ 24 months
Latest Table	Hostname + MonitorItemID	MonitorItem +MonitorItem*	1	
Status Table	Hostname	MonitorItem	10	24 months

Agenda

- Motivation
- Related Work
- Model Design
- Evaluation
- Conclusion

Bai du 百度

Evaluation

- High Availability
- Memory Usage
- Read/Write Performance

High Availability

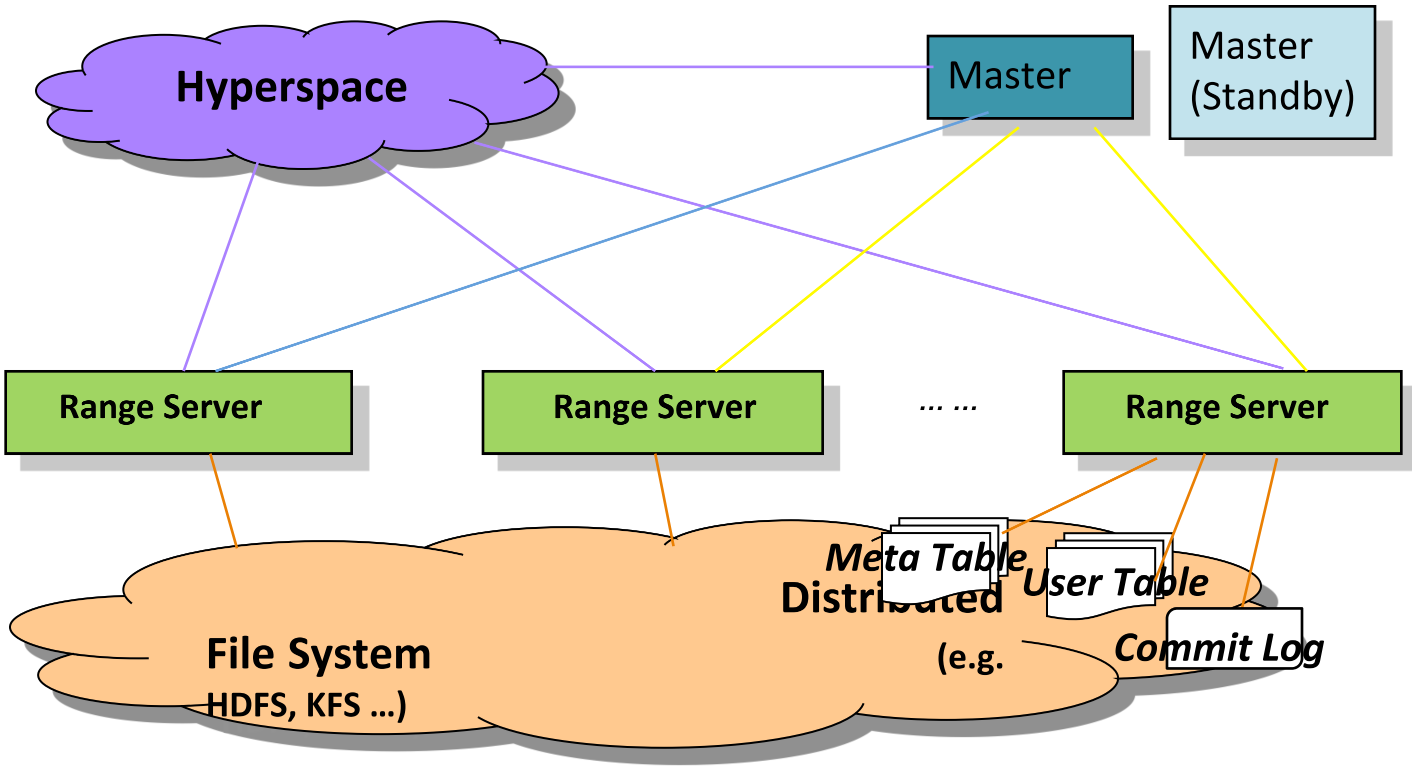
- Challenge
  - Recovery
    - No support sync for append operation in Hadoop-v2 (based on 0.18.2)
  - Load Balance
    - No implementation in Hypertable-B2 (based on 0.9.2.0)

High Availability

- Improvements
  - Recovery
    - Data in HDFS, Log in LocalFS
      - Local-Recovery
    - Centralized Meta Ranges
      - 2048PB User-Data index by 16TB Meta-Data
    - Master Standby
    - Application Remove-duplicates
  - Load Balance
    - Split-driven
    - Manual online and offline operations

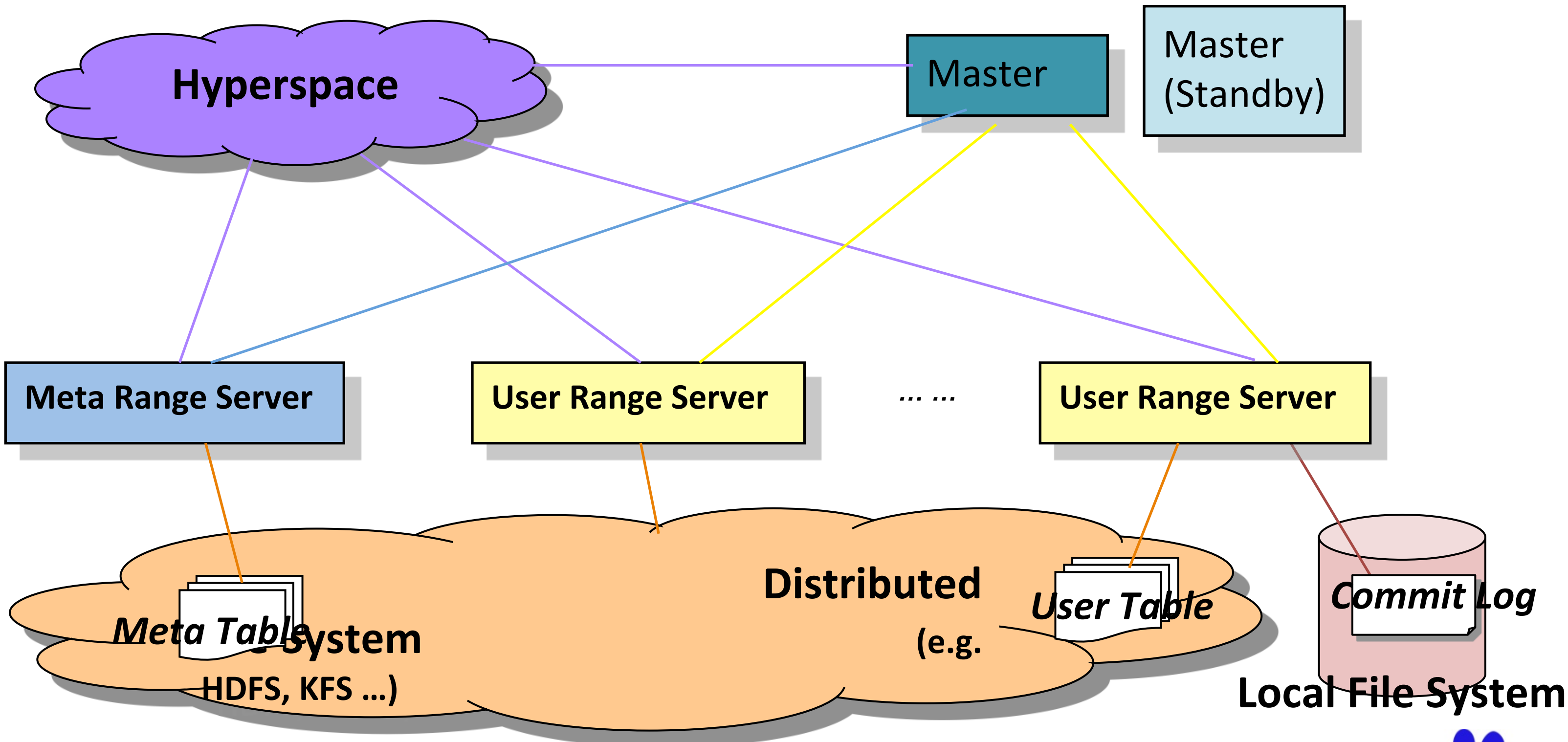
High Availability

- Deployment Model (before improvement)



High Availability

- Deployment Model (after improvement)



Weaknesses

- Range data managed by a single range server
  - Though no data loss, can cause periods of unavailability
  - Can be mitigated by client-side cache or memcached
  - Can minimize recovery time by active standby of cluster

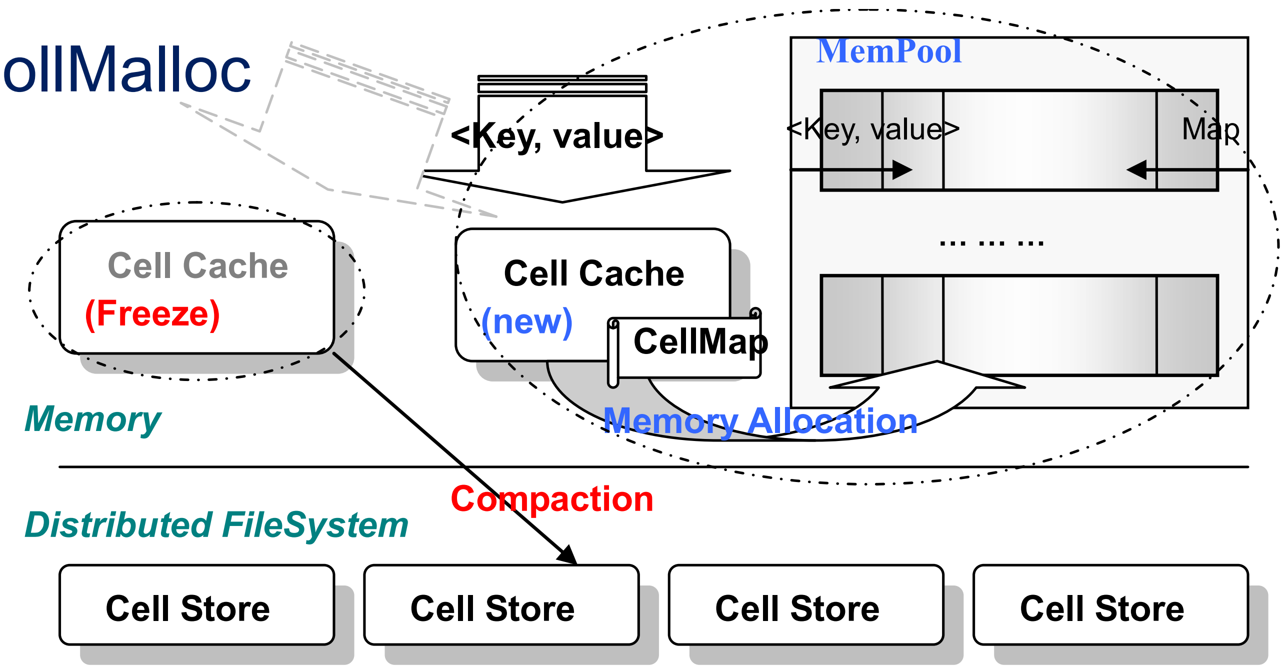
Memory Usage

- Challenge
  - Hypertable is memory intensive

Function (during execution)	Memory Usage
Hypertable::CellCache::add	75.6%
__gnu_cxx::new_allocator::allocate	18.8%
Hypertable::DynamicBuffer::grow	4.1%
Hypertable::IOHandlerData::handle_event	1.0%
Hypertable::BlockCompressionCodecLzo ::BlockCompressionCodecLzo	0.5%

Memory Usage

- Improvement
  - TCMalloc
  - SimplePollMalloc

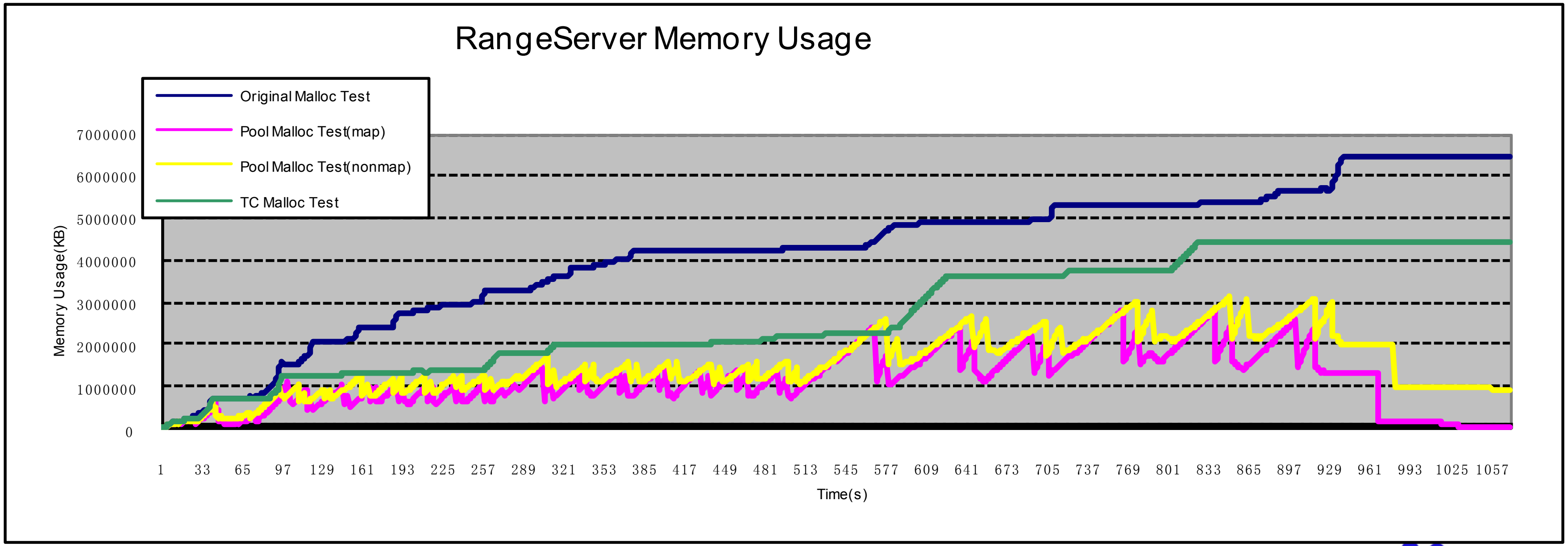


Memory Usage

Function (during execution)	Mem Usage	Mem Usage	Function (during execution)
CellCachePool::get_memory	**94.3%**	**75.6%**	Hypertable ::CellCache::add
		18.8%	__gnu_cxx ::new_allocator::allocate
Hypertable:: DynamicBuffer::grow	3.8%	4.1%	Hypertable ::DynamicBuffer::grow
Hypertable ::BlockCompressionCodecLzo ::BlockCompressionCodecLzo	1.1%	1.0%	Hypertable ::IOHandlerData ::handle_event
Hypertable ::IOHandlerData ::handle_event	0.5%	0.5%	Hypertable ::BlockCompressionCodecLzo ::BlockCompressionCodecLzo

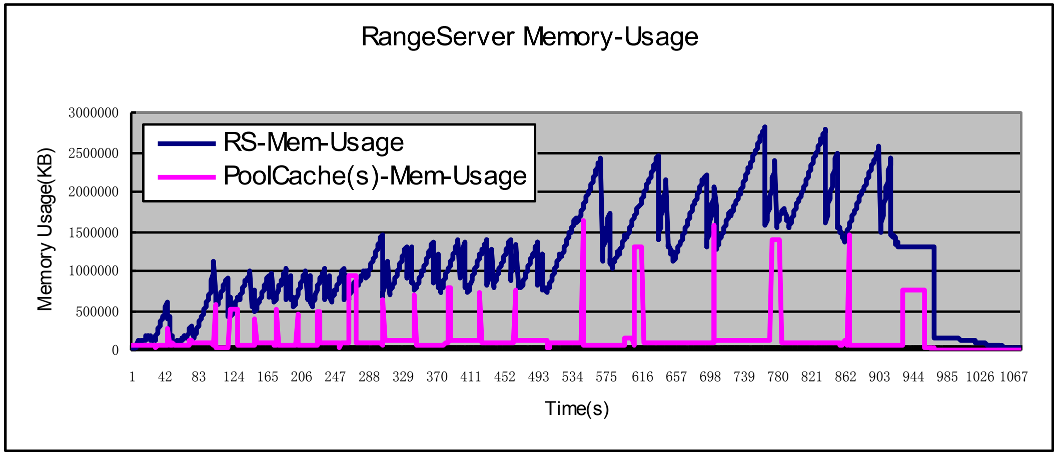
Memory Usage

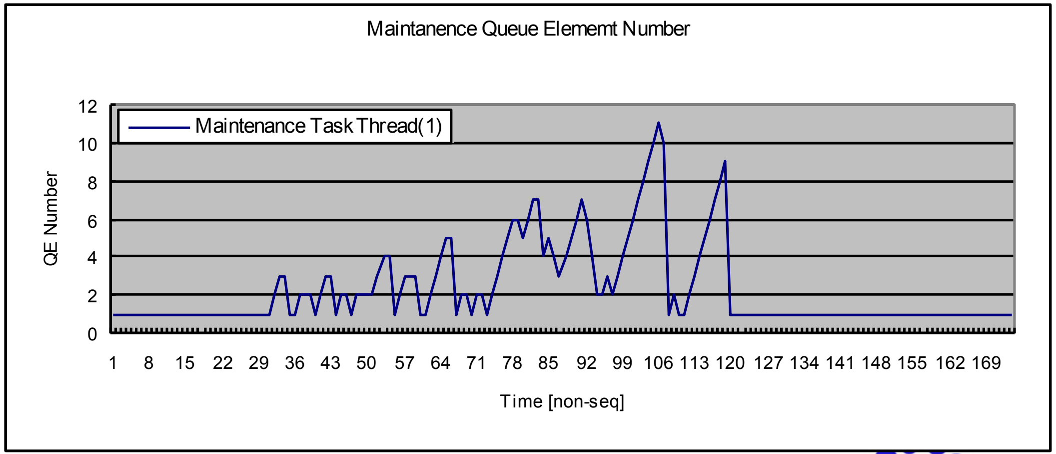
- New/Delete vs. TCMalloc vs. PollMalloc vs. PollMalloc (with map)



Memory Usage

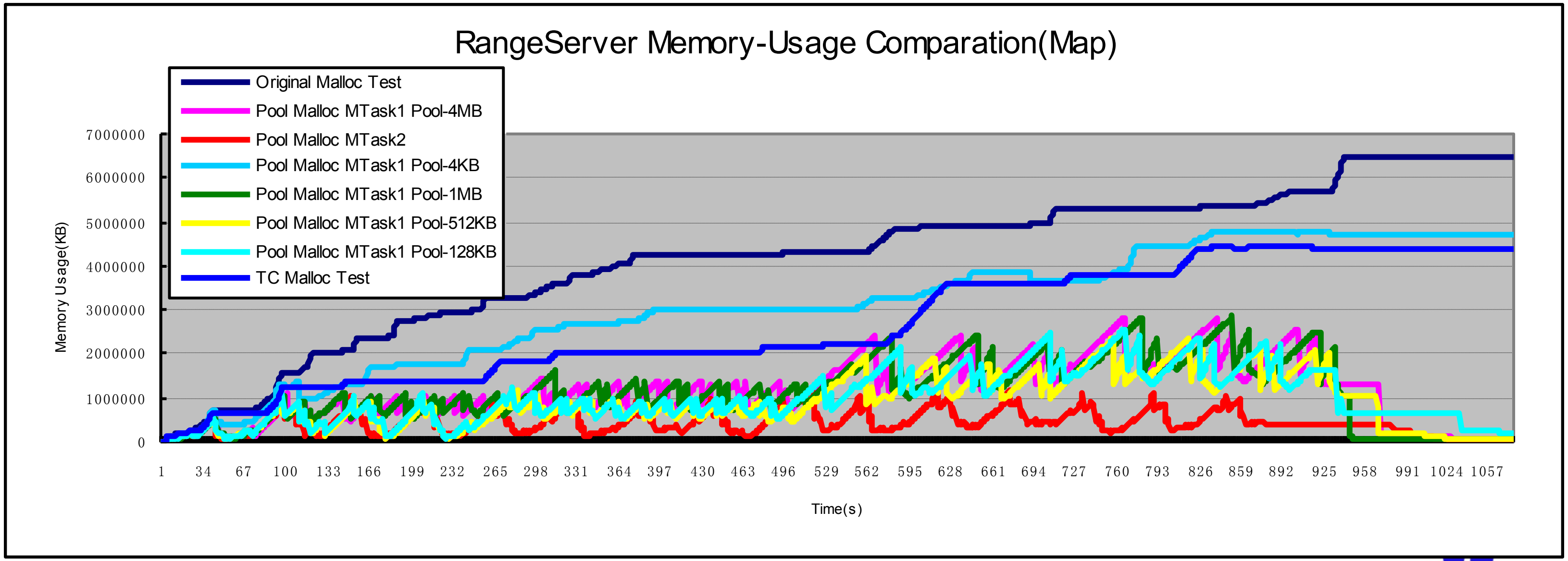
- Opportunity
  - Compaction Efficiency





Memory Usage

- Improvement
  - Compaction Optimization Factor
    - Cell Cache Buffer Size
    - Maintenance Thread Number



Read Performance

- Challenge
  - Noah generates lots of random read ops

Machine	Sequential Write (KB/s)	Random Read (KB/s)	Random Write (KB/s)
X1	**284191**	**1932**	**79834**
X2	165264	973	1627
X3	93919	863	7256
X4	63565	471	23149
X5	55598	424	18768
X6	60858	336	23188

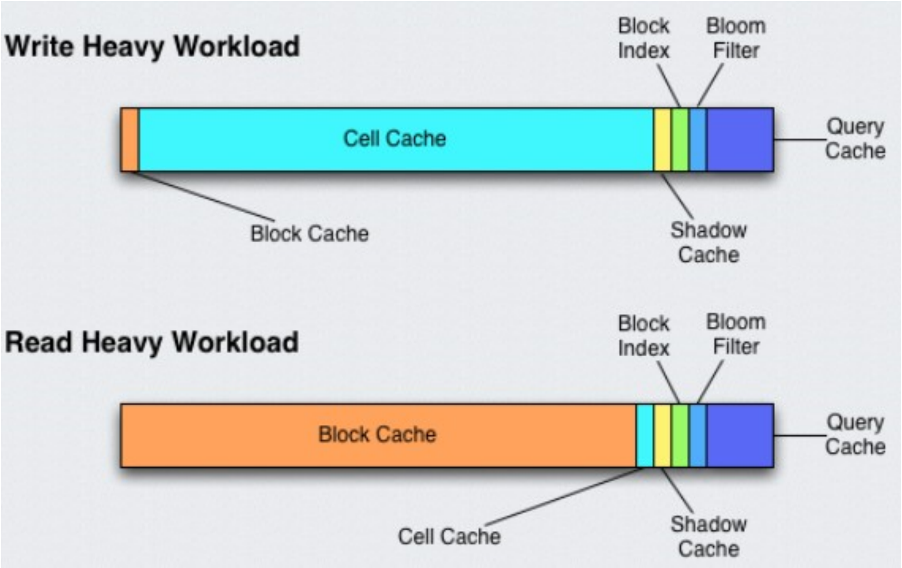
<./iozone -f t3 -s 10g -c -e -w -+n -i 0 ./iozone -f t3 -s 10g -c -e -w -+n -i 2 >

Read Performance

- Memory Read
  - Latency
    - ms level
  - Memory usage
    - Sparse Table: value + 40Bytes(keys + map)
    - MVCC: Old record is deleted when cell cache compaction

Read Performance

- Cache Read
  - Block Cache
    - Cellstore blocks
    - Uncompressed
  - Query Cache
    - Query results



Read Performance

- Disk Read
  - CellStore
    - 5 CSs/Range, 5*216=950 RR, 950*2=1900 RR; 500 iops, ioutil 100%
  - Bloomfilter

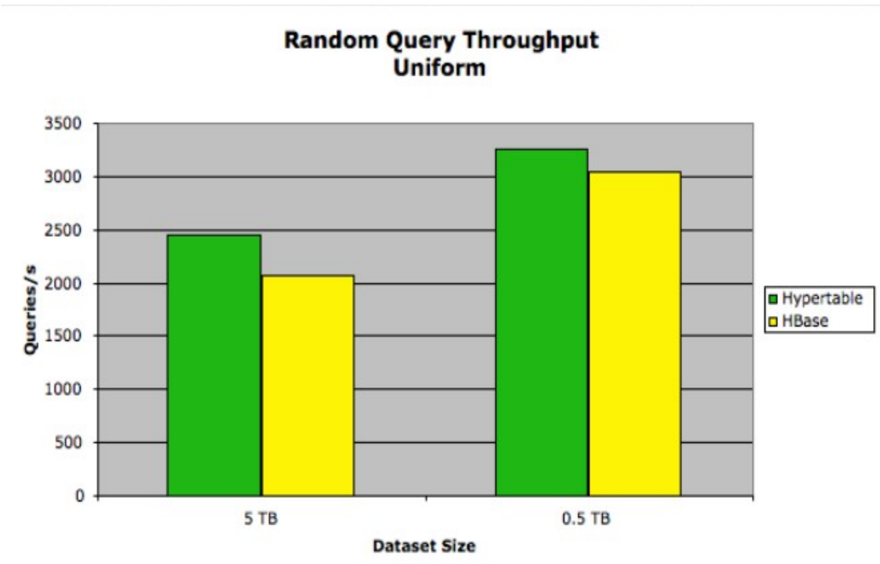
	CS 10			CS 5			CS 1		
Threads	1	16	32	1	16	32	1	16	32
Queries/s	3.8	3.8	2.4	9.8	13.5	5	23.1	52.6	25.6
Aggregate Queries/s	3.8	61	77	9.8	216	160	23.1	1684	1641
Average Latency(ms)	260	260	420	102	74	200	43	19	39

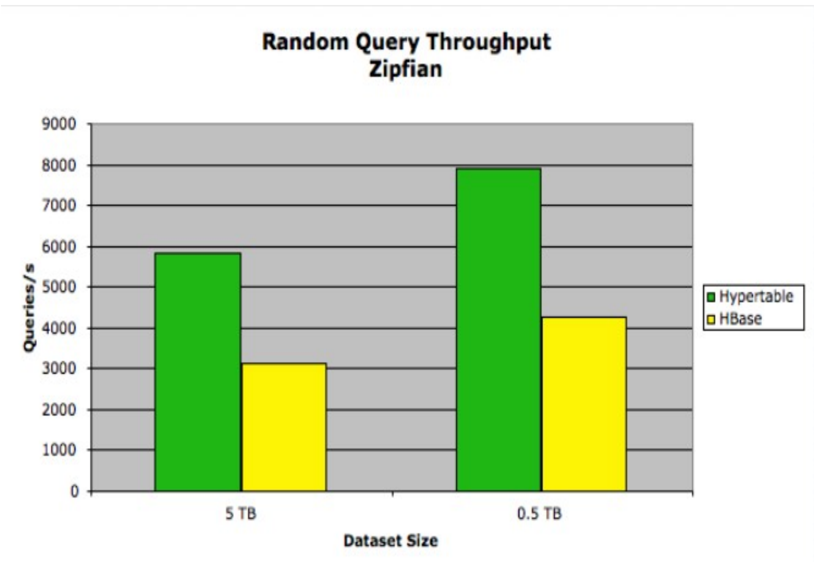
Read Performance

- Disk Read
  - Compression
    - Lzo/Quicklz/Gzip/Snappy
  - SSD/SATA/SAS
    - Read Access Time/Capacity/Price per GB/Idle or Full power/MTBF
  - Concurrent W/R
    - Resource isolation
    - Manual control

Read Performance

- Hypertable vs. Hbase





Write Performance

- Challenge
  - HDFS operation pressure
  - RangeServer writing pressure
- Improvement
  - Compaction frequency adjusting
  - Row key reversal

Agenda

- Motivation
- Related Work
- Model Design
- Evaluation
- Conclusion

Conclusion

- Application level
  - Table Design
  - Load-in Strategy
  - Duplicate Removing
- High Availability
  - Metadata Centralization
  - Master Standby
  - Log/Data Separation
  - Load Balance
  - System Active Standby
- Memory Usage
  - Memory Pool
  - Compaction Strategy
- Read/Write Performance
  - Mem/SSD/SAS/SATA
  - Block/Query Caching
  - Compression Strategy
  - Resource Isolation
  - Compaction Strategy

Hypertable versus HBase

versus	Hypertable	HBase
Community	Hypertable	Apache
Company	Hypertable Inc	Cloudera, HortonWorks
Implementation Language	Boost C++	Java
Memory Management	Explicit Memory Management	Garbage Collection
Cache Managemnt	Dynamic cache management	Java heap for caching
Performance	High	Normal
Compiler Optimization	Easy	Hard
Compression	Direct native compression	JNI-based compression

Thanks for your Attention!

Questions?

百度技术沙龙

畅想 交流 争鸣 聚会

关注我们：t.baidu-tech.com

资料下载和详细介绍：infoq.com/cn/zones/baidu-salon

“畅想•交流•争鸣•聚会”是百度技术沙龙的宗旨。 百度技术沙龙是由百度与InfoQ中文站定期组织的线下技术交流活动。目的是让中高端技术人员有一个相对自由的思想交流和交友沟通的的平台。主要分讲师分享和OpenSpace两个关键环节，每期只关注一个焦点话题。

讲师分享和现场Q&A让大家了解百度和其他知名网站技术支持的先进实践经验，OpenSpace环节是百度技术沙龙主题的升华和展开，提供一个自由交流的平台。针对当期主题，参与者人人都可以发起话题，展开讨论。

InfoQ 策划·组织·实施

关注我们：weibo.com/infoqchina