Hbase For Enterprise

Agenda

- Wire compatibility through protobuf
- Unified RPC engine for insecure and secure Hbase
- Quick recovery after disaster
- Prefix-Trie data block encoding
- Snapshot / restore capability at table level
- Region Server Group based Assignment
- Support for running HBase on Windows server
- Support for JDK 1.7
- Q & A

About myself

- Graduated from Tsinghua University
- Have been working on Hbase for over two years
- Promoted Hbase committer / PMC member in June 2011
- Developer in Hadoop team at Ebay

Increasing Cross-version compatibility

- Umbrella issue is HBASE-5305
- Maintenance entails co-existence of clients with different RPC versions
- Serialization using protobuf format
  - New fields could be easily introduced
  - Self-describing format
- Automatically-generated serialization and deserialization code
  - Remote Procedure Call requests
  - Format for persistent data storage

Increasing Cross-version compatibility, cont'd

- ProtobufRpcEngine replaces WritableRpcEngine
- RPC requests/responses are native protobuf messages
- SecureRPCEngine removed, security-related logic moved to the core engine
- Better efficiency in terms of avoiding copies in the RPC layer (every PB message has a getSerializedSize method generated)
- In HBASE-2182 Netty based RPC was proposed, goal is to allow native clients to be written
- Whether VersionedProtocol should be replaced is under discussion. One choice is to add getServerCapabilities()

Table level snapshot

- Helps disaster recovery
- Facebook backup needs Hardlink; recovery replays WAL
- HBASE-6610 HFileLink: Hardlink alternative for snapshot restore
- HBASE-6055 is the master JIRA for snapshot
- Zero-copy of files
- Point-in-time semantics
- SLA enforcement: guaranteed max unavailability
- Built-in recovery

Snapshot Types

- Offline snapshot: table disabled
- Globally consistent snapshot

1. consistent across all servers
2. uses two-phase commit
3. No flushing
4. Unavailability SLA maintained per region

- Timestamp consistent snapshot: point-in-time according to each server

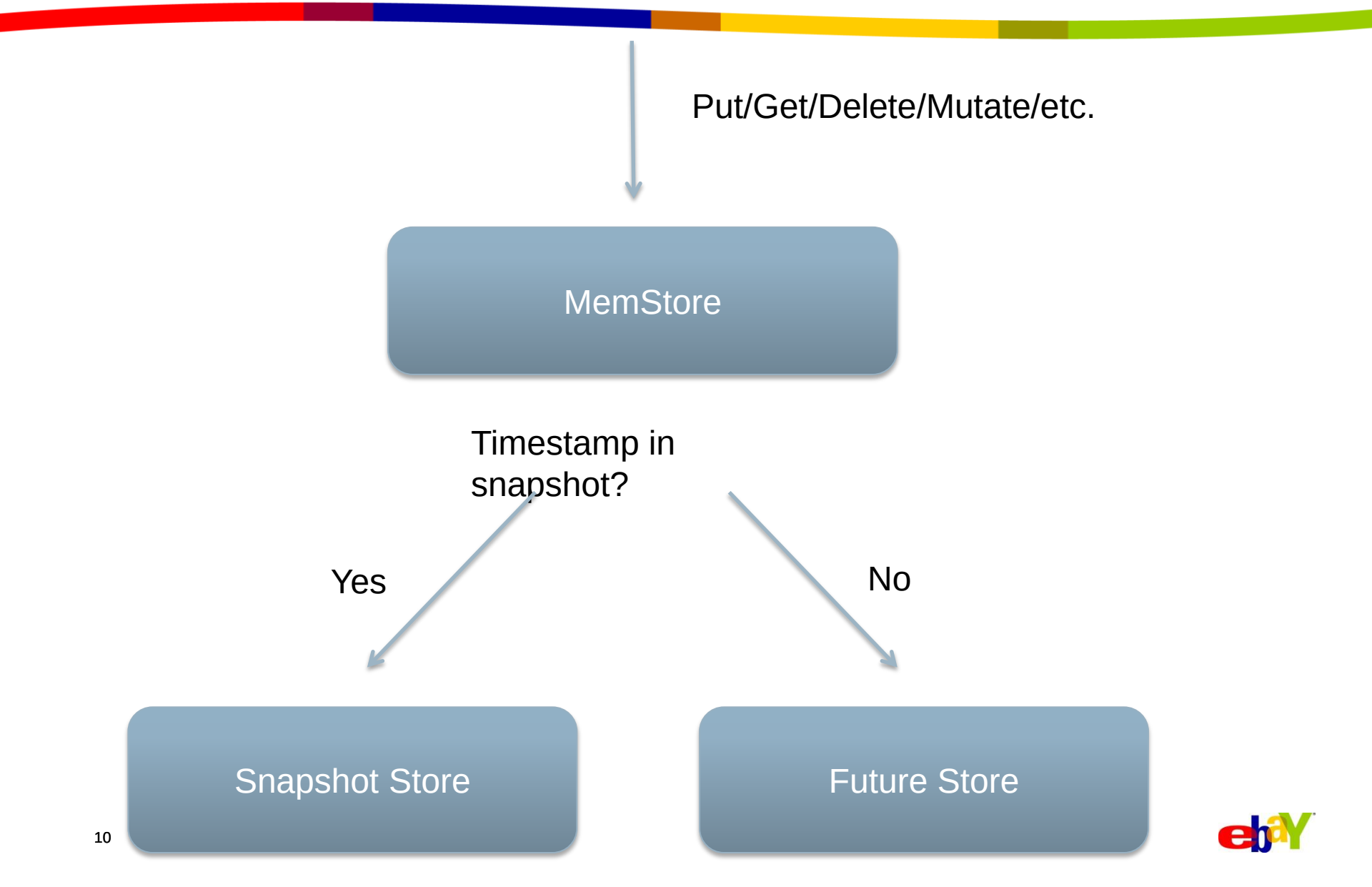
Potential issues with Globally consistent snapshot

- Writes are blocked while waiting for all servers
- Limited by slowest region
- Requires WAL replay to restore table

Timestamp Consistent Snapshots

- All writes up to a TimeStamp are in the snapshot
- Leverages existing flush functionality
- Doesn’t block writes
- No WAL replay on recovery

Timestamp Consistent Snapshots, cont'd

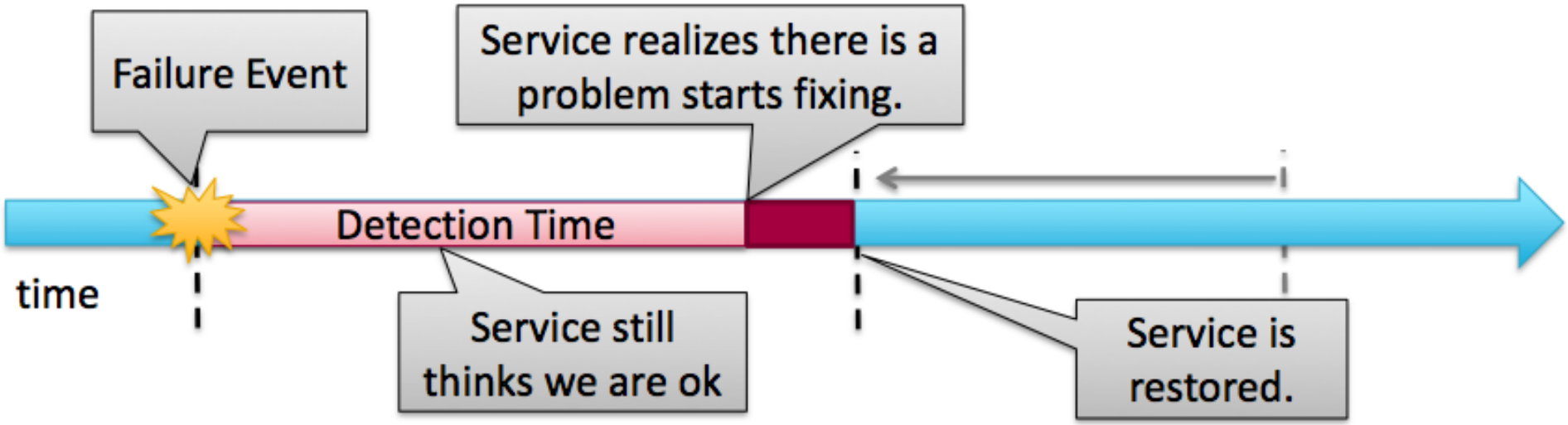


Snapshot Recovery Options

- Export snapshot
  - Send snapshot to another cluster
  - All required HFiles/Hlogs
- Clone snapshot
  - Create new table from snapshot
- Restore table
  - Rollback table to specific state
  - Handles region creation / deletion
  - Fixes META for you

Reduce downtime by speeding up recovery

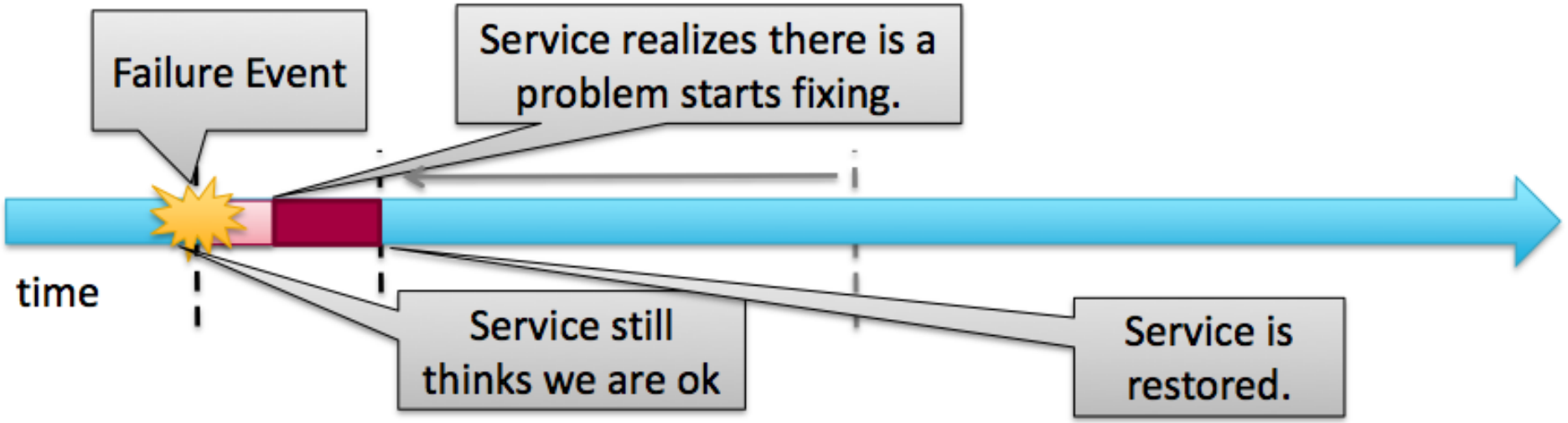
- HBASE-5843 Improve HBase MTTR - Mean Time To Recover



Improve HBase MTTR - Mean Time To Recover

- HDFS-3703 Decrease the datanode failure detection time, by default turned off
- HBASE-6752 On region server failure, serve writes and time-ranged reads during the log split
- HBASE-6435 Reading WAL files after a recovery leads to time lost in HDFS timeouts when using dead datanodes
- HBASE-6109 Improve RIT performances during assignment on large clusters
- HBASE-5970 Improve AM#updateTimer and speed up handling opened event

Reducing downtime by speeding up detection



- Proactively notify to recover from process failures quickly

0.92/0.94	All Master Failure Detection 180s	Some Region Server Failure detection 180s
0.96	Master process failure detection 0-1s	Region Server Failure detection 0-1s

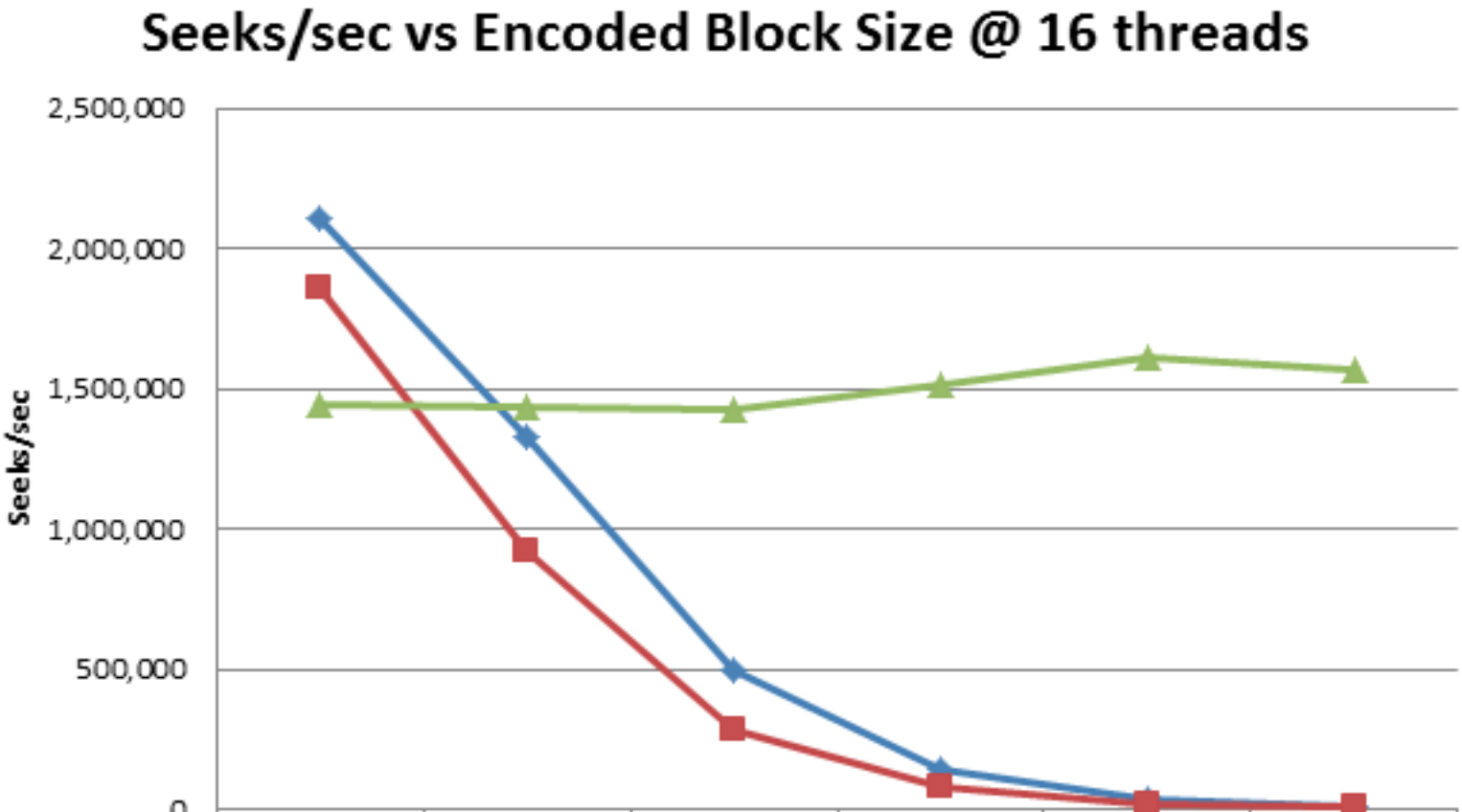
Modularizing HBase

- There are two builds for 0.92 and 0.94, one for insecure Hbase, one for Hbase with security
- HBASE-4336 modularized maven build
- Multiple pom.xml, one for each module: common, integration tests, server
- There are separate jar files for different modules: common, server, hadoop-compat, prefix-tree

Prefix-Trie Block Encoding

- Introduced in 0.96 as a new module
- Cell interface is introduced
- KeyValue class implements Cell interface
- Supports the random access Searcher methods which jump to any Cell in the block at roughly the same cost
- Trie format for row key encoding
- Column family stored once at the beginning of block
- Qualifiers in the block are stored in their own trie format
- Minimum timestamp stored at the beginning of the block

Prefix-Trie Block Encoding, cont'd



	256B	1KB	4KB	16KB	64KB	256KB
NONE	2,109,706	1,331,488	493,733	138,557	38,036	9,726
PREFIX	1,863,481	923,550	283,914	77,858	20,630	5,248
TRIE	1,446,817	1,429,824	1,425,903	1,515,525	1,614,555	1,567,727

Prefix-Trie Block Encoding, cont'd

- Each cell is fixed-width to enable binary searching
- Further (suffix, etc) compression on the trie nodes
- Using all VInts instead of FInts with sacrifice on random seek speed

raw Block Size	num blocks	avg Raw Block Bytes	avg Cells Per Block	# of Cells	avg Raw Key Bytes	approx Input MB
256KB	1310	262019	4172	5465516	52	327.34
Encoding	**Encoding MB/s**	**seeks/s**	**Avg Encoded Block Bytes**	**Avg Encoded Key Bytes**	**Key Compress Ratio**	**compression ratio**
None	661.54	12955	?	52	1	1
Prefix	232.9	8197	108462	24	2.17	2.42
Trie	66.62	127043	45618	9	5.78	5.74

Region Server Group based Assignment

- Early work in HBASE-4120 started with prioritization
- HBASE-5169 First implementation of region server groups, allowing table and group level balancing
- HBASE-6721 resumes region server group development

Running Hbase on Windows Server

- HBASE-6817 Get HBase tests working under Windows
- HBASE-6818 Catalog table .META. violates the naming convention in Window OS
- HBASE-6832 Tests should use explicit timestamp for Puts, and not rely on implicit RS timing
- System.currentTimeMilis() may not provide satisfactory update frequency on Windows

Q & A