HBase Large Object (LOB) Storage

Xue, Wei

wei.xue@intel.com

Agenda

- The Large Object (LOB) Use Case
  - Design Goals
  - Potential Solutions
- Considering Implementation on Apache HBase
- Performance results

What is Large Object (LOB)?

- *Usually refers to BLOB (Binary Large Object) and CLOB (Character Large Object); can be PDF documents, word documents, images, multimedia objects, etc.*

- *Unlike structured or text records, LOBs can typically be several hundred KB to tens or hundreds of MB in size.*

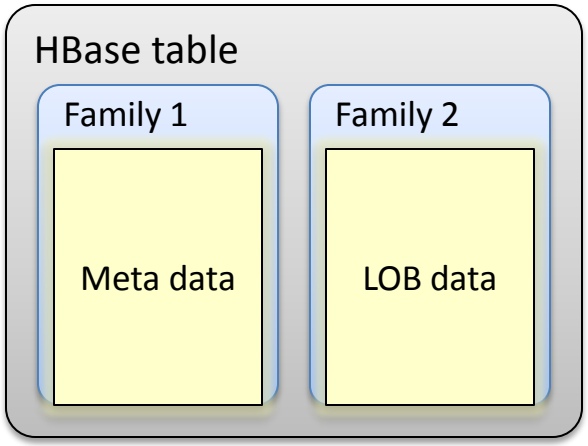
Large Object (LOB) Applications

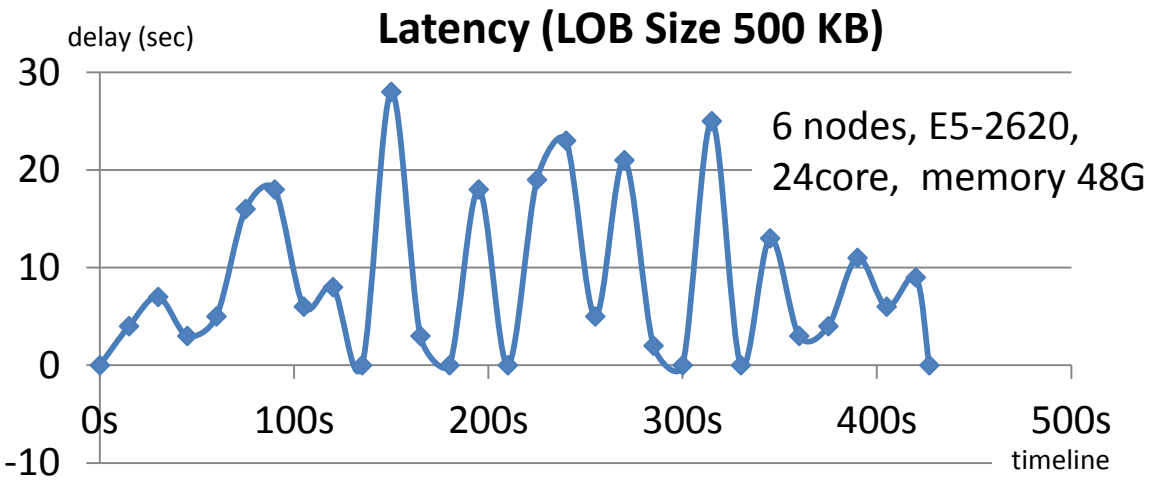
- *Case1: Online apps are now looking to show real-time photos of the traffic, together with maps and live traffic status.*
- *Case2: Bank XXX's customers can use their hand-held devices to query their transaction history and the photo copies of the original bills.*

Design Goals

- Avoid too many small files
- Minimized impact on write performance: stable low latency and high throughput
- Low read latency and good concurrent read performance
- Consistency and transparency

Potential Solutions: LOB in HBase tables



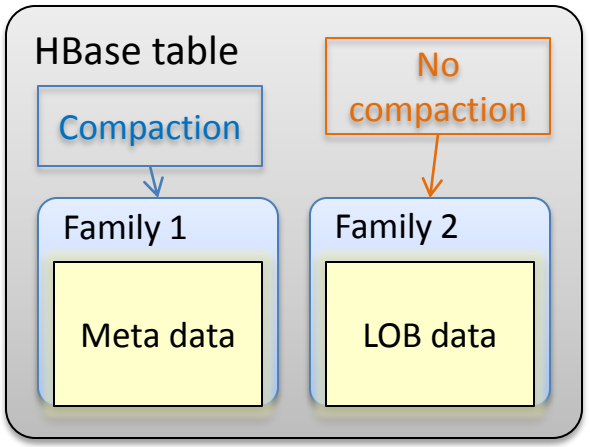


- *Region split and compaction → heavy I/O*
- *Slow compaction → flush delayed → blocking updates*
- *High latency → socket timeout → unnecessary retries*

Potential Solutions:
LOB in HBase + customized compaction

Customized compaction policy:

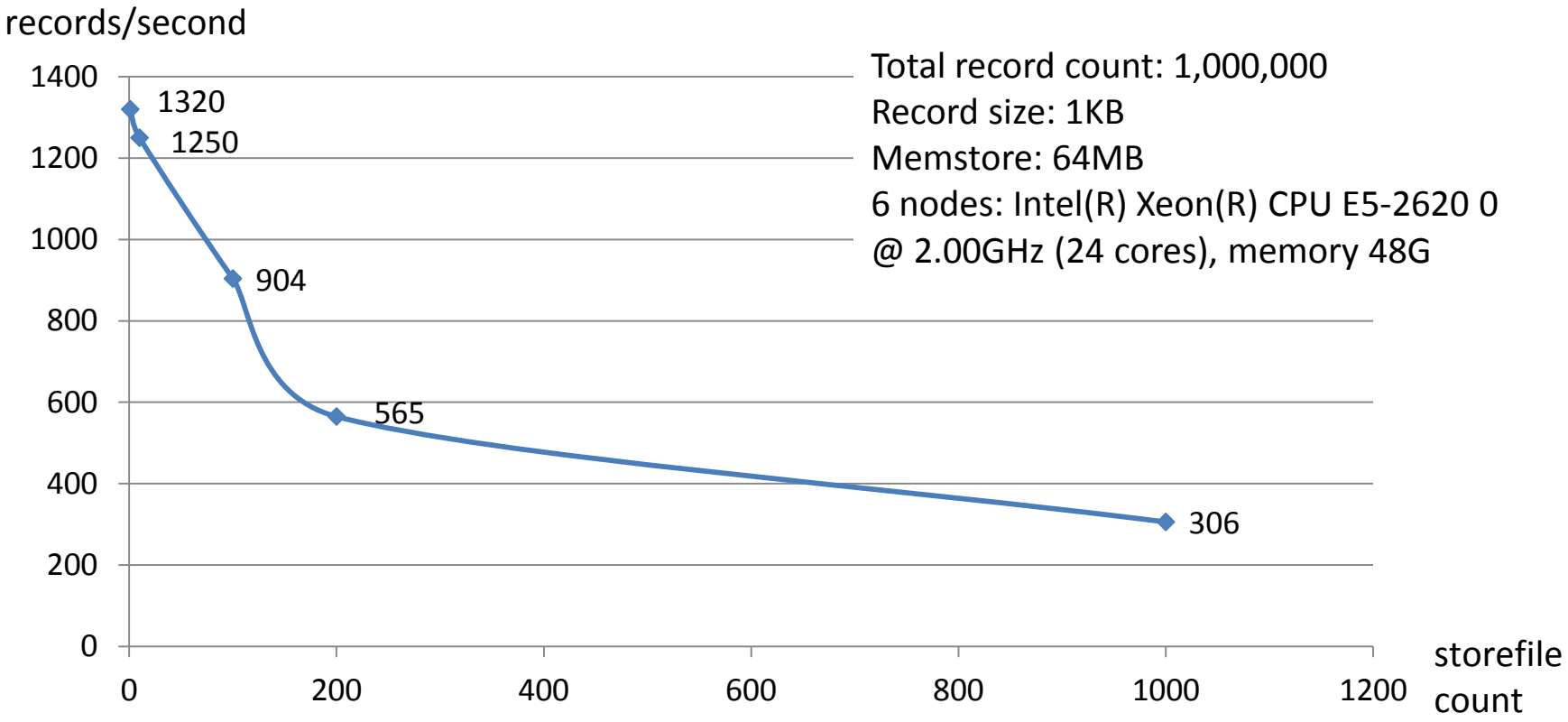
skip compaction for LOB data



- Expensive I/O when splitting LOB files
- Too many LOB storefiles:
  - Slow scan
  - Slow random read

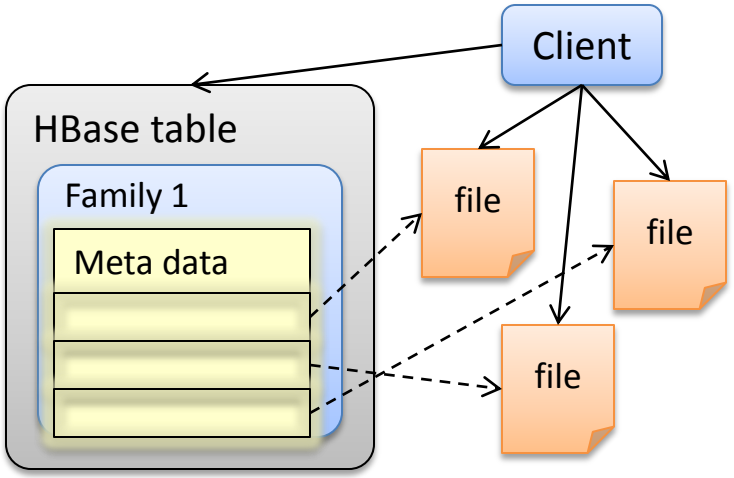
Potential Solutions:
LOB in HBase + customized compaction

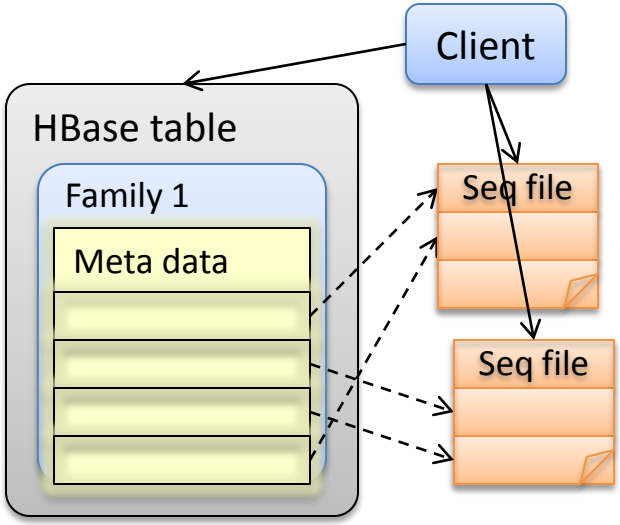
Random get (with bloom-filter) performance decline with storefile count increase



Potential Solutions:
LOB in HDFS + meta data in HBase

- *Approach 1:*
  HDFS – one file per LOB entry
  HBase – meta data and LOB file path
- Too many small files

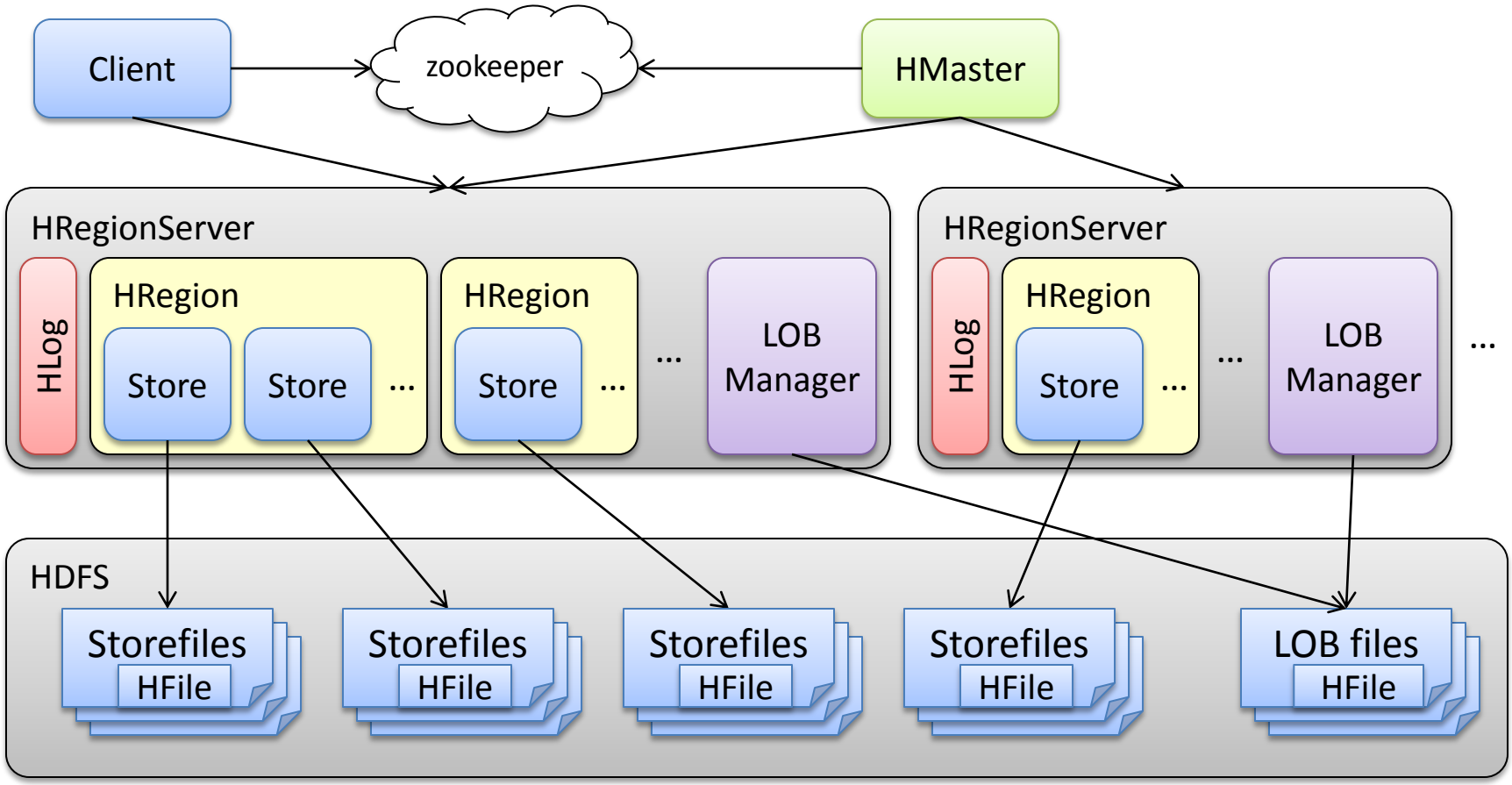




- *Approach 2:*
  HDFS – many LOBs in one sequence file
  HBase – meta data and sequence file path + LOB offset
- No consistency guarantee

- Poor manageability: unused or out-dated LOBs?

LOB Implementation on HBase: Design Overview

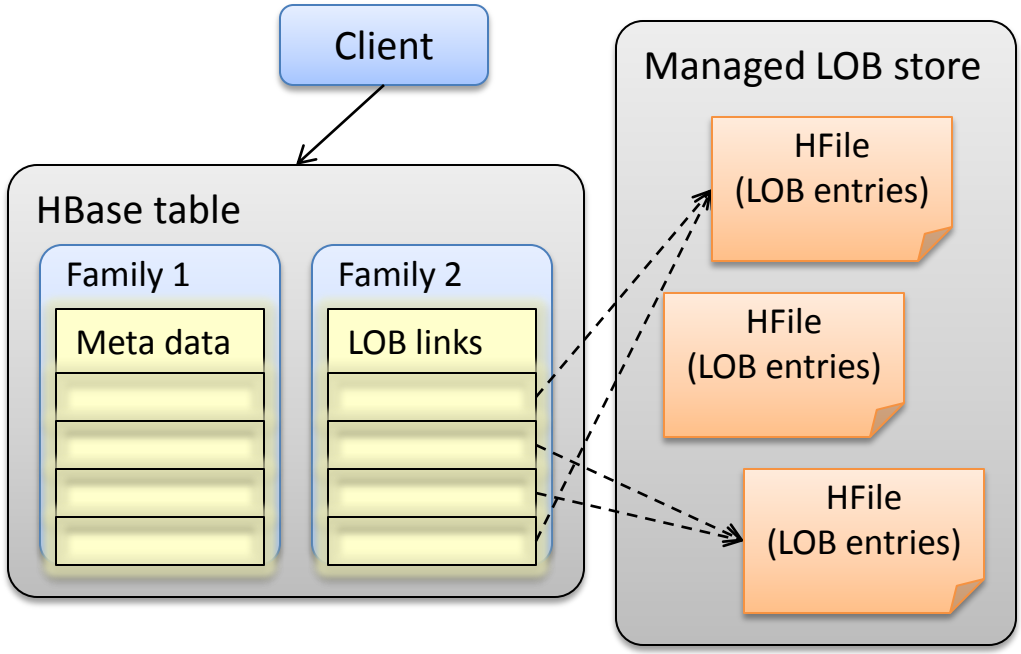


LOB Implementation on HBase: Design Overview

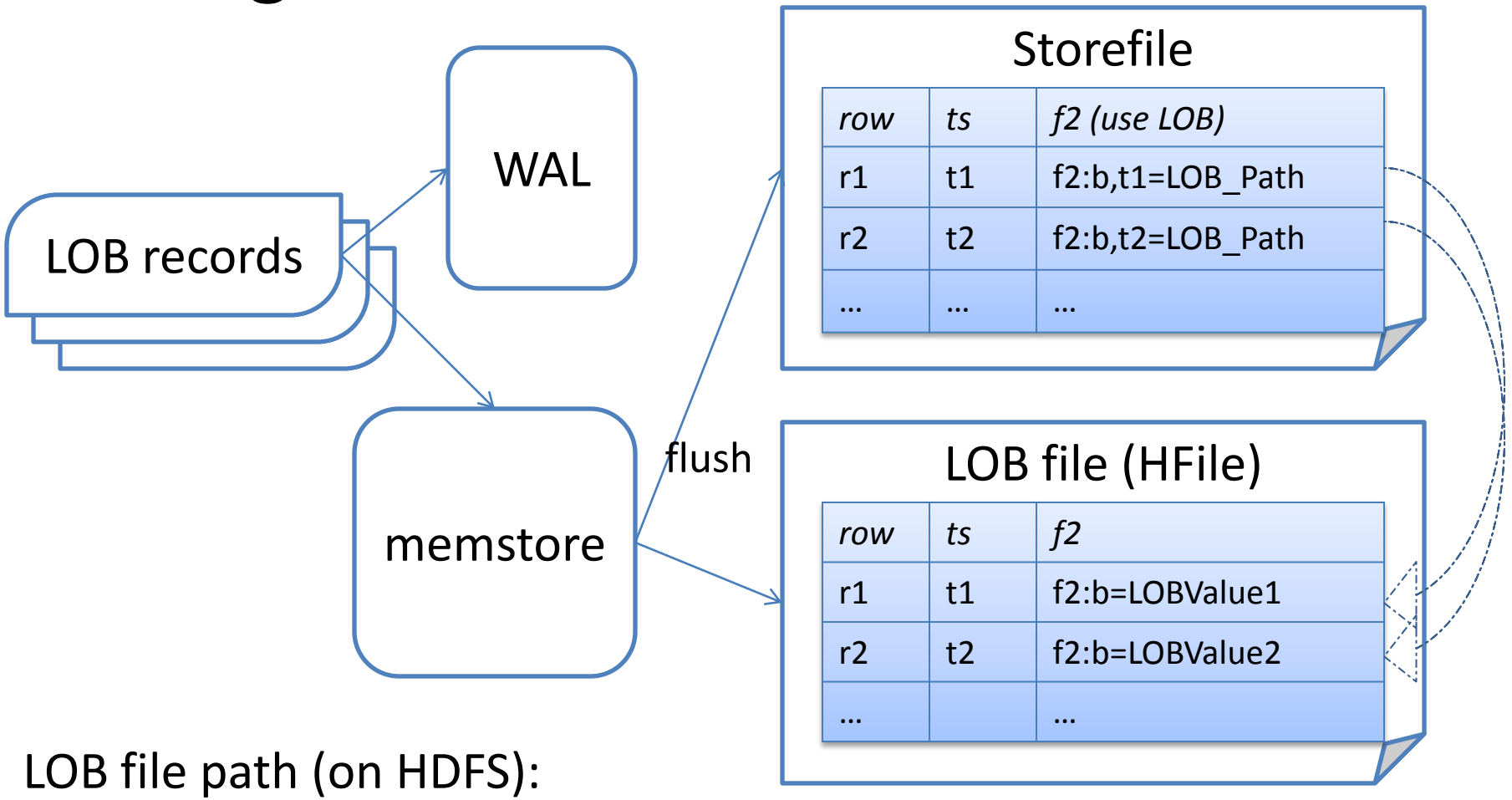
- Meta and LOB data in separate families
- Customize flush() for the LOB family:

  flush LOB data into a LOB file;

  on storefile flushing, replace LOB values with the LOB file name
- Customize major compaction for LOB family



LOB Implementation on HBase: Writing LOB records



LOB file path (on HDFS):

${LOB_ROOT}/${table}/${lob_family}/${startkey}/${uuid}_${lob_count}

LOB Implementation on HBase: Reading LOB records

1. Retrieve desired records from the HBase table

r1	f1:a,t1=v1	f2:b,t1="lobs/table1/f2/r0/a4ba509c-587b-4cc9-9c12-1ffb9b537ee2_52"

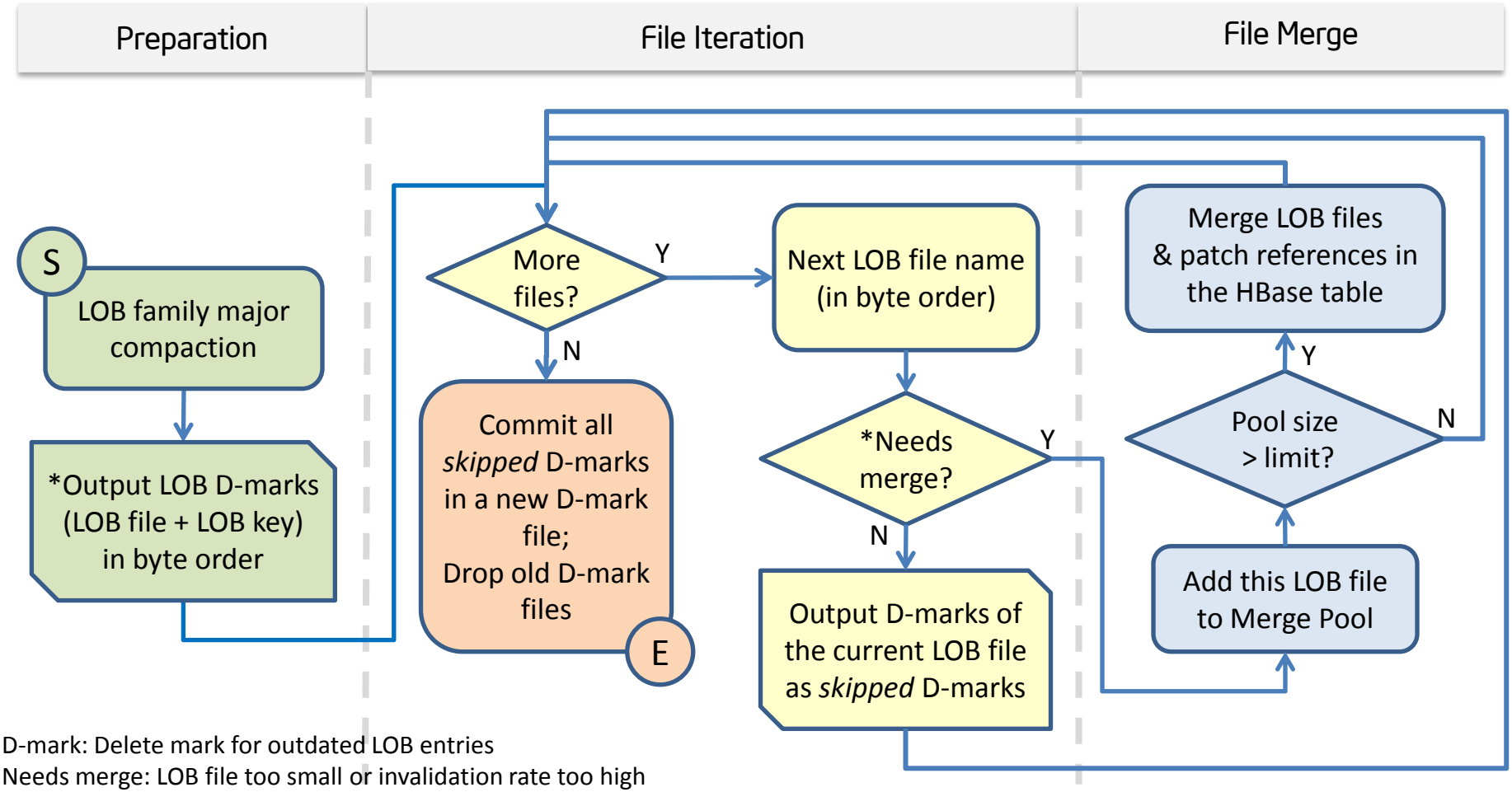
2. Search for the LOB keys ("r1, f2:b, t1") in the corresponding LOB files ("hdfs://lobs/table1/f2/r0/a4ba509c-587b-4cc9-9c12-1ffb9b537ee2_52")

r1	f2:b,t1=(LOB value)

3. Replace LOB file paths with LOB values

r1	f1:a,t1=v1	f2:b,t1=(LOB value)

LOB Implementation on HBase: LOB compaction



*D-mark: Delete mark for outdated LOB entries
*Needs merge: LOB file too small or invalidation rate too high
*The invalidation rate is inferred from D-mark count and entry count of a LOB file, and the entry count is included in the LOB file name.

LOB Implementation on HBase: Benefits

- Quick indexing for LOB data
- Multiple LOB entries combined in one HFile
- No I/O overhead in minor compactions
- No I/O overhead in region splitting
- Write LOB during flush: consistency guaranteed
- Leverage major compaction to assure LOB manageability and improve LOB locality: do clean-up, merge and balance in one pass

Considering Implementation on Apache HBase
Required changes to HBase core

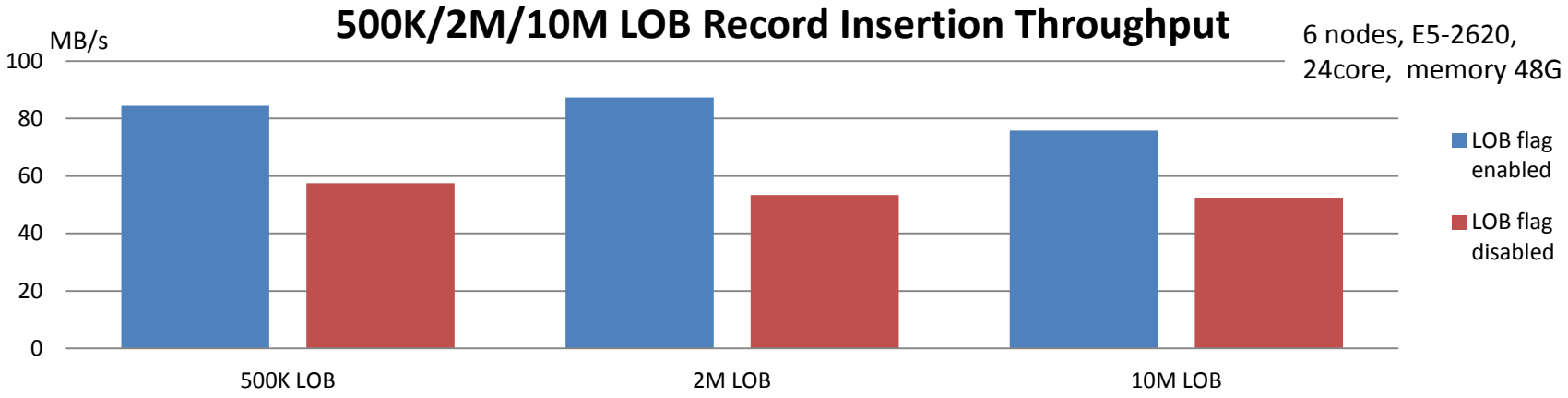
- Pluggable compaction policy
  - No changes here, thanks to HBASE-7516 (available in 0.96)

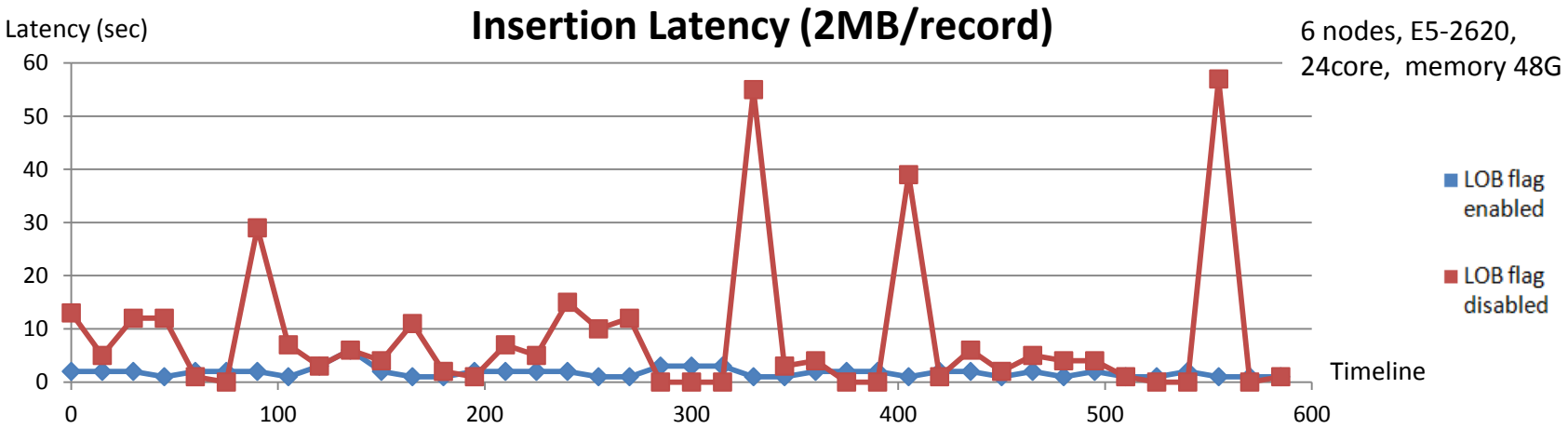
- Pluggable flush policy:
  - Make it possible to override flush implementation

  Going to propose

Performance Results:

Single HBase Client





Performance Results

Stress test with multiple HBase Client machines (to reach network bottleneck)