HBase Large Object Storage(LOB)

Motive

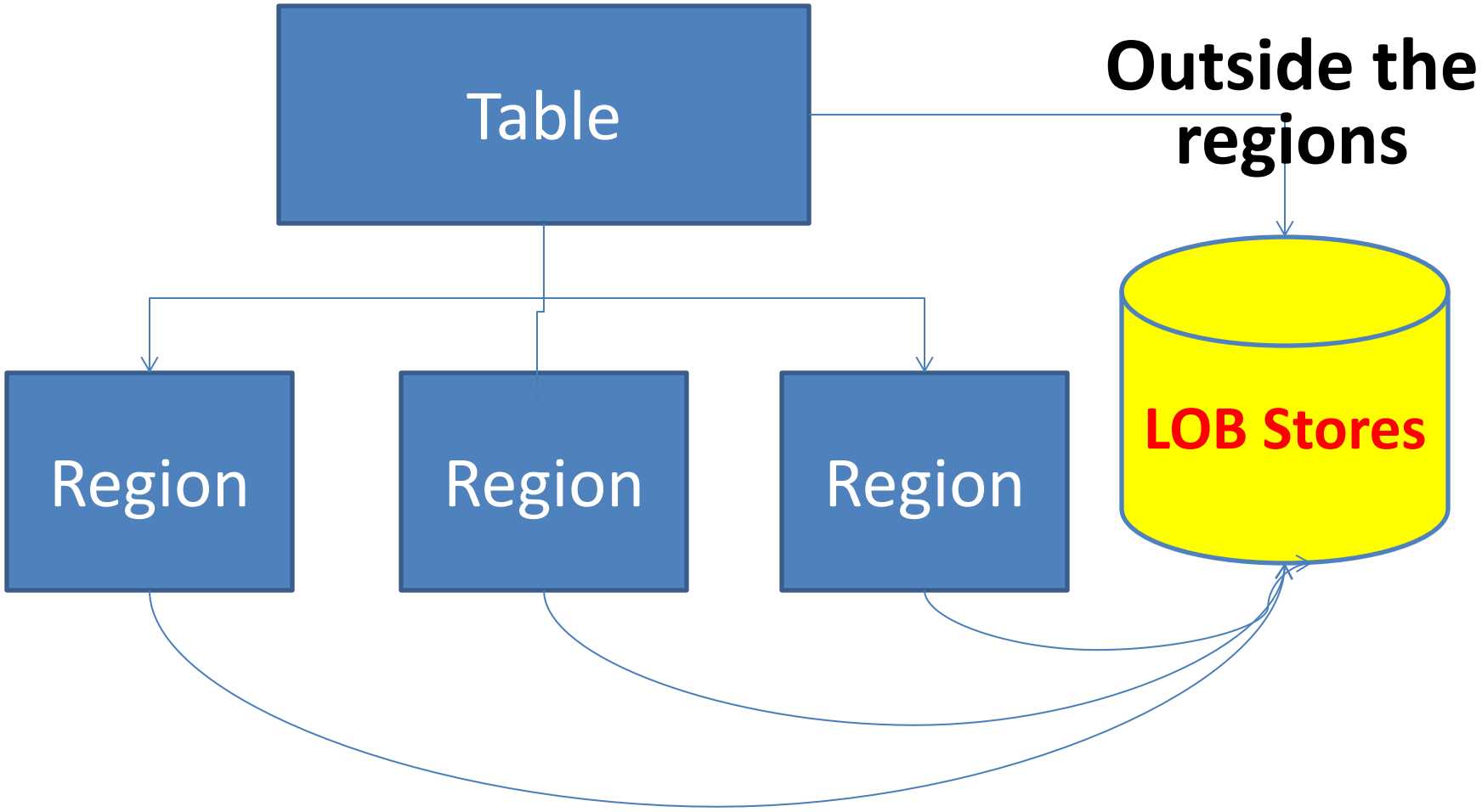
1. Reduce the performance impact of compaction, and split.
2. HBase act as a file system to store large binary data and scale infinitely.

Three possible approaches

1. Scale by adding more horizontal regions. (bottleneck: memory, Hbase mater)
2. Scale by storing thousands of store file in single region. (Investigate in another doc) (bottleneck: memory, cpu)
    1) Need customize the compaction algorithm
    2) Bloom filter will be a bottle neck when there are large number of store files.
    3) Can scale to thousands of store file in single region.
3. Scale by saving binary data into the LOB Storage which is independent of the region, thus not impacted by the compaction.
    1) No limit on the scale ability

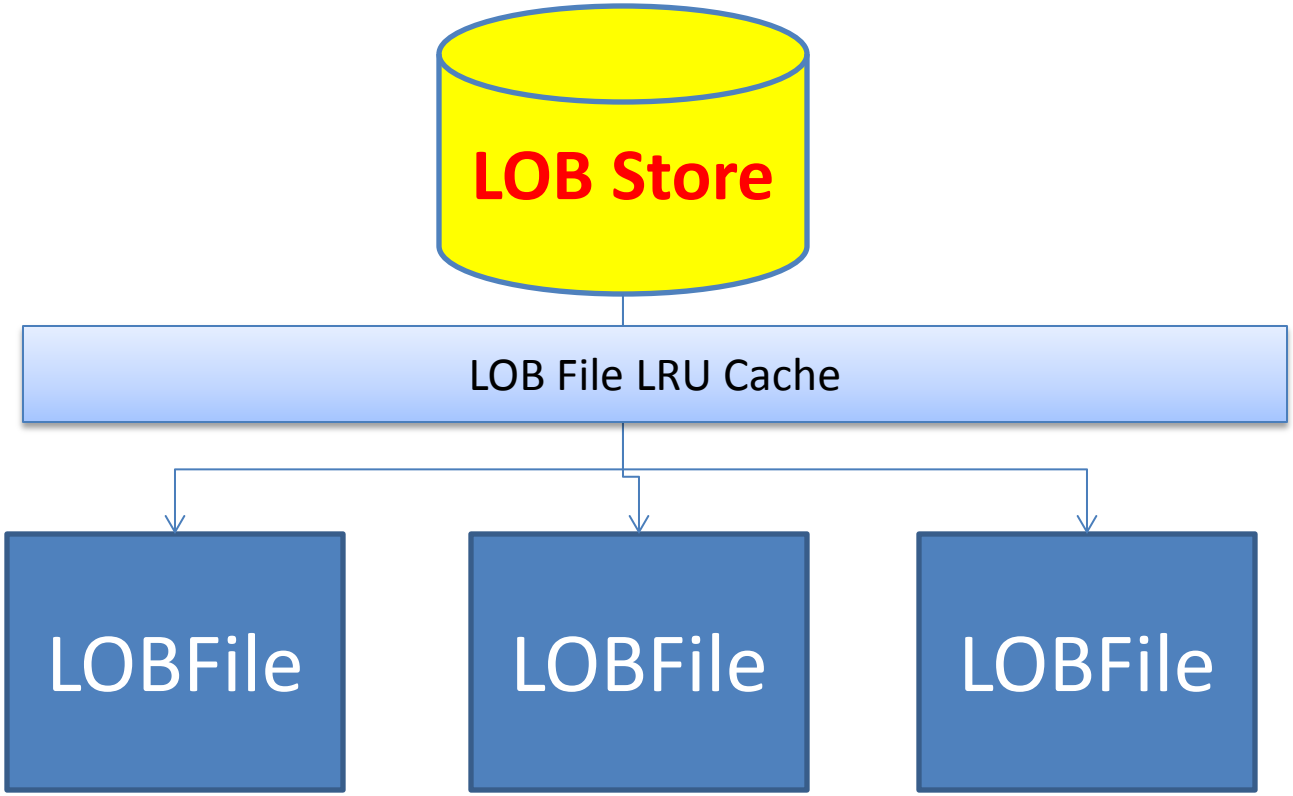
Approach 1 and 2 will meet bottleneck when scaling to some level. We will focus approach 3.

Overall Design



LOB Store

- LOB Store is at column family level. Each LOB Store can contains millions of LOB Files.



LOB's Benefit

1. **Scale infinitely** to store big record. Small CPU and memory needed.
2. **Fast Insertion.** Insertion Performance is **200%.**
3. **Reduce Insertion Delay by 90%.**
4. **Improved availability**

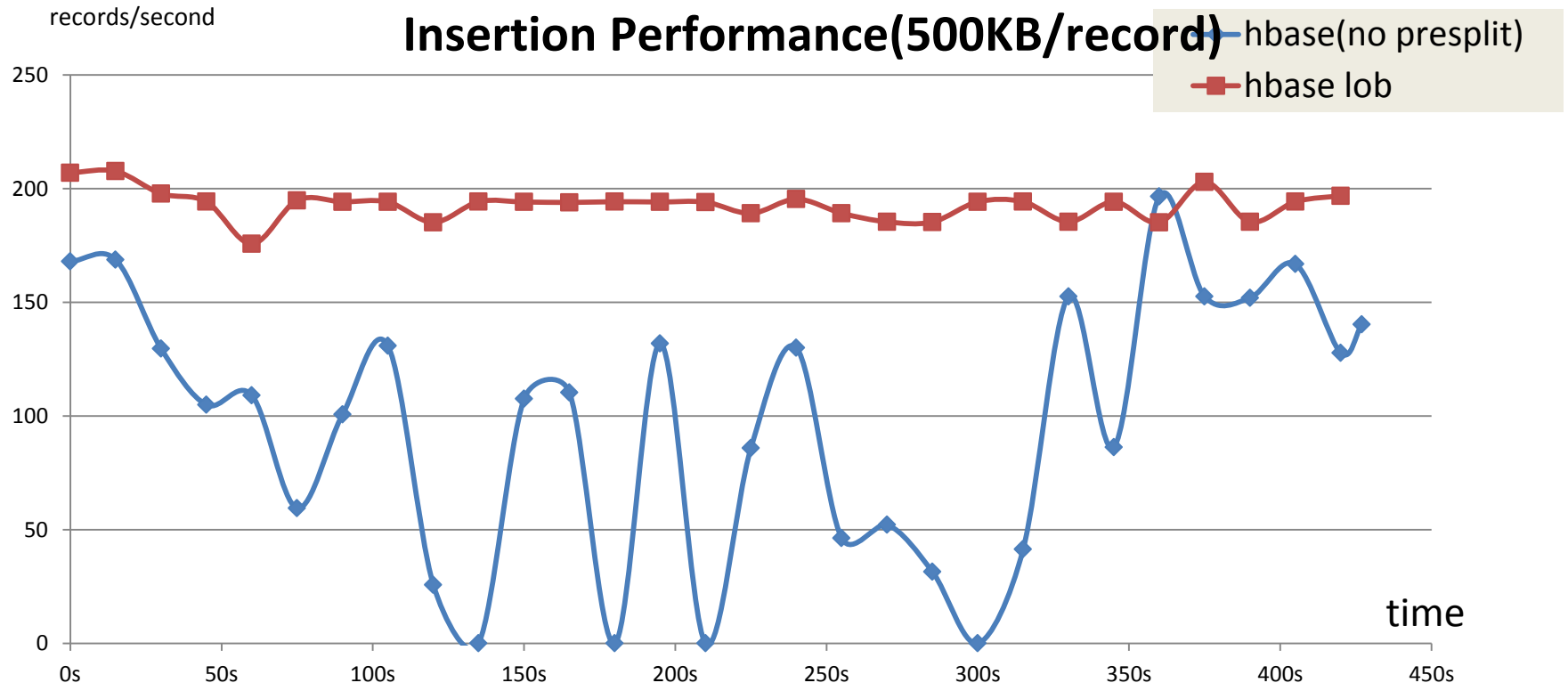
   Eliminate timeout exception during insertion because of compaction, split or load balancing. **Scale infinitely** for big record storage.
5. **200%** random get performance.
6. Faster sequential scan, **130%** performance**.**
7. **Transparent to user. Retain all HBase functions.**

When LOB is applicable?

- When the record size is bigger than 100KB, and less than 5MB.

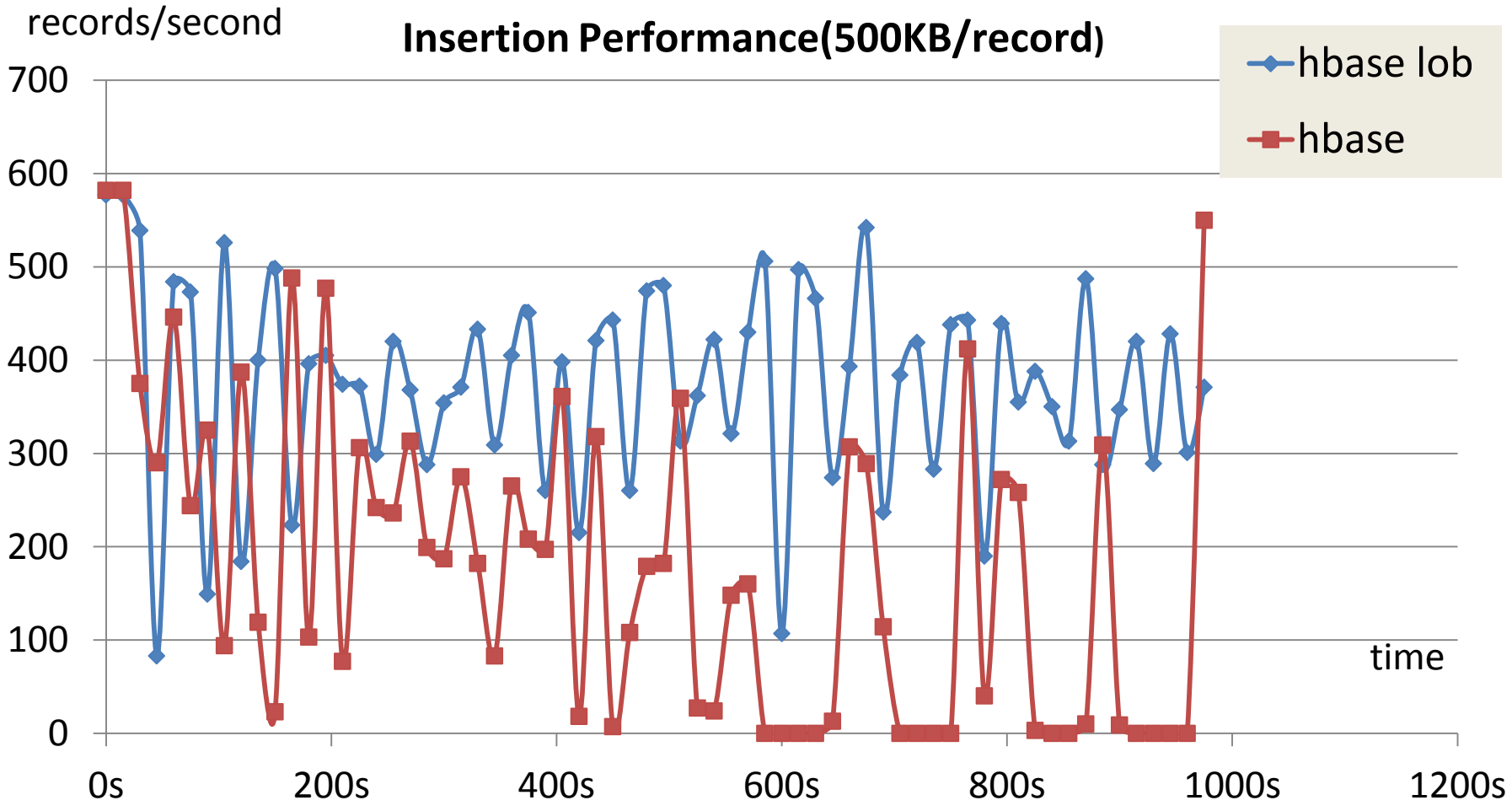
Faster Insertion

Test1: Insertion Performance(no pre-split, low concurrency)



Test setup: (intel-01 cluster, 6 machines, E5-2620, 24core, 48G memory). No client cache. No WAL. For hbase(no split), after insertion, the region count is 20. Single client(8 thread) . 3 replications.

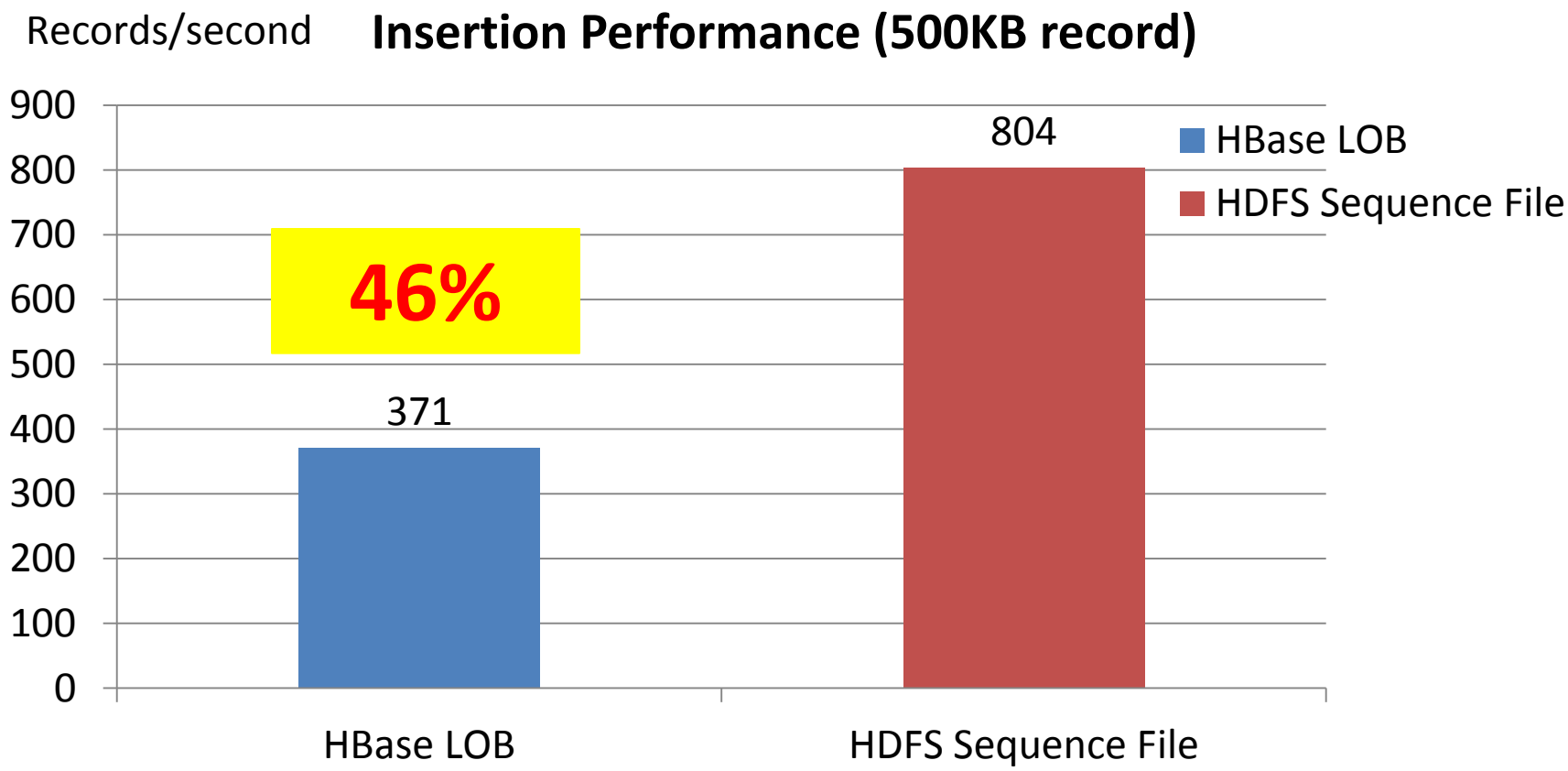
Test2: Insertion Performance(Pre-split 32, high concurrency)



Test setup: (intel-01 cluster, 6 machines, E5-2620, 24core, 48G memory). No client cache. No WAL. Pre-split 32 regions.

6 Client in total, each client has 8 threads. 3 replications.

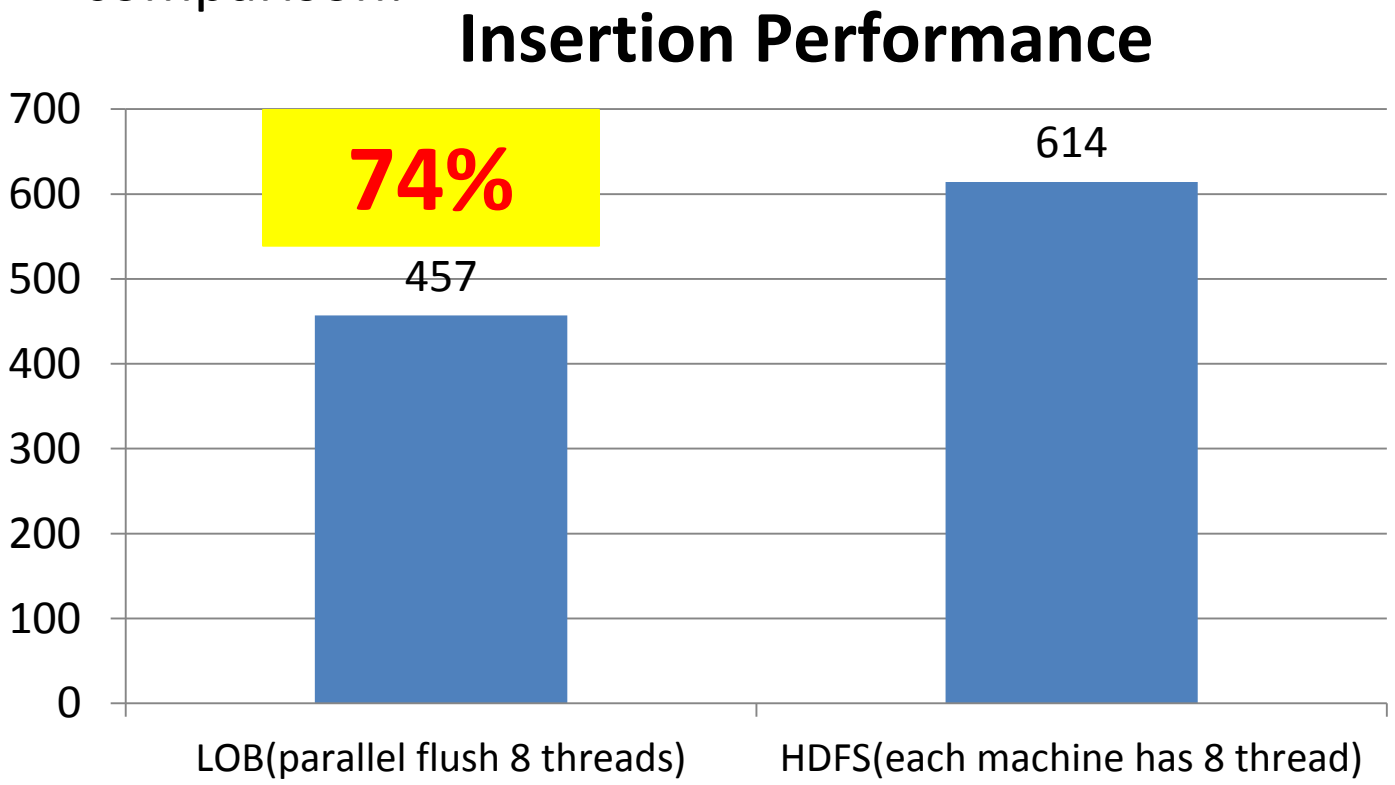
Test3: Insertion Performance HBase LOB vs. HDFS Sequence File (turn **off** parallel flush)



Test setup: (intel-01 cluster, 6 machines, E5-2620, 24core, 48G memory)
For HDFS Sequence File, we used 6 clients, each client have 8 threads.
For Hbase LOB, each region server have ONE thread to flush the memstore to disk.

Test4: Insertion Performance HBase LOB vs. HDFS Sequence File (turn **on** parallel flush)

- If we use **8 parallel** flush thread, the performance comparison:



Test setup: (intel-01 cluster, **5** machines, E5-2620, 24core, 48G memory)

For HDFS Sequence File, we used 5 clients, each client have 8 threads. Inserted 90GB data.

For Hbase LOB, each region server have 8 thread to flush the memstore to disk. Inserted 80GB data.

HBase LOB only has **26% performance drop** compared to writing to HDFS directly.

Insertion Performance is **doubled**

For Single Client (500KB, no presplit, 8 threads, no WAL, 3 replications, auto-split to 20 regions):

Insertion Test: No table pre-split	Insertion Performance(For whole cluster)	
Hbase	95 records/second	
HBase LOB(large object storage)	190 records/second	

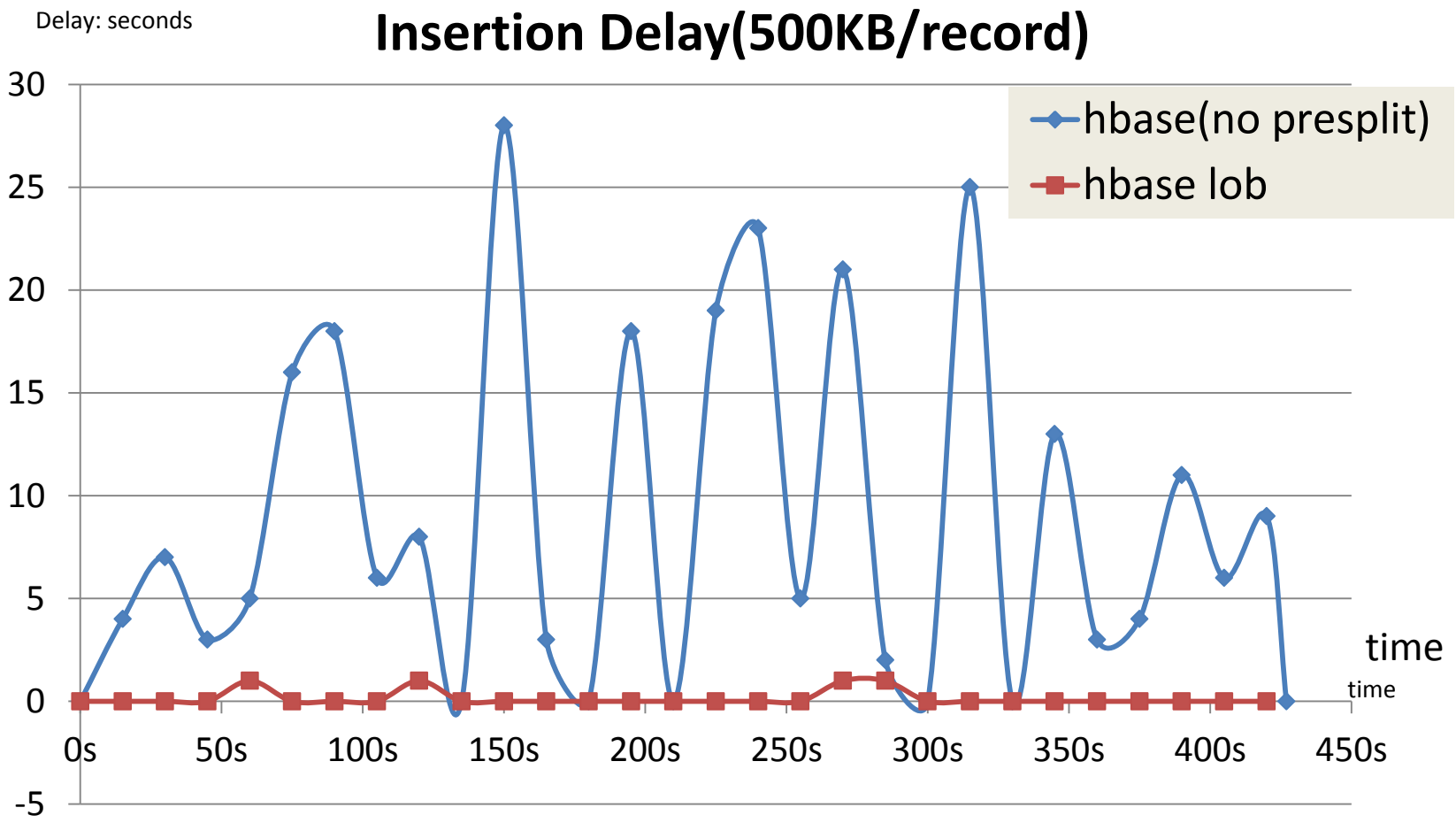
For 6 Clients (500KB, no presplit, 8 threads for each client, , no WAL, 3 replications, pre-split

Insert: Table pre-split to 32 regions	Insertion Performance(For whole cluster)	
Hbase	168 records/second	
HBase LOB(large object storage)	**371 records/second**	

Compared to Sequence file, LOB remains **74%** insertion Performance. Only have **26%** overhead.

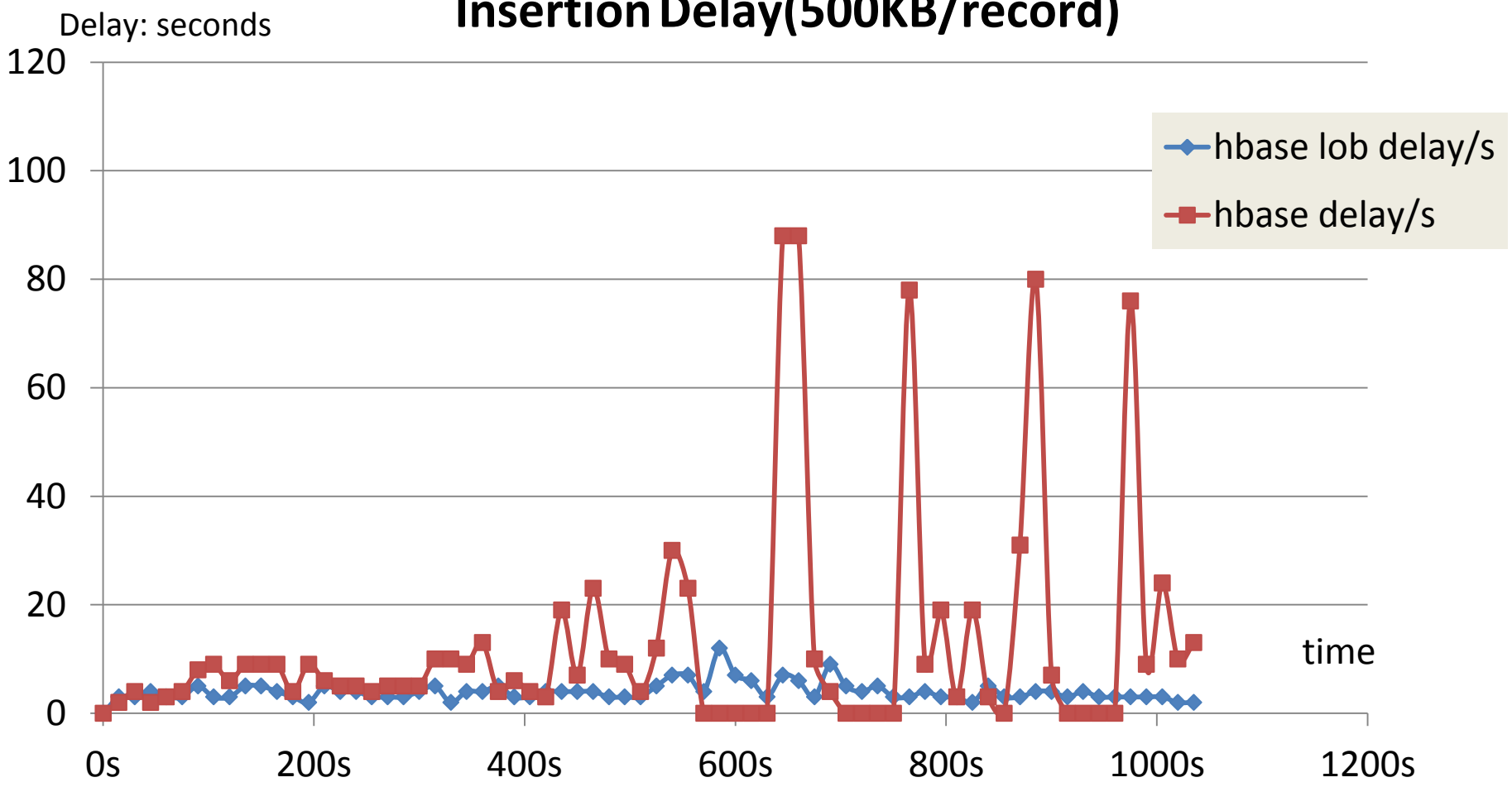
Low Insertion Delay

Test5: Insertion Delay(No pre-split, low concurrency)



Test setup: (intel-01 cluster, 6 machines, E5-2620, 24core, 48G memory). No client cache. No WAL. For hbase(no split), after insertion, the region count is 20. Single client(8 thread) . 3 replications.

Test6: Insertion Delay(Pre-split 32 regions, High concurrency)



Test setup: (intel-01 cluster, 6 machines, E5-2620, 24core, 48G memory). No client cache. No WAL. Pre-split 32 regions.
6 Client, each client has 8 threads. 3 replications.

Insertion delay cut by **90%**

Under low stress

For Single Client (500KB, no presplit, 8 threads, no WAL, 3 replications, auto-split to 20 regions):

Insertion Test: No table pre-split		Max Delay
Hbase		27 s
HBase LOB(large object storage)		2 s

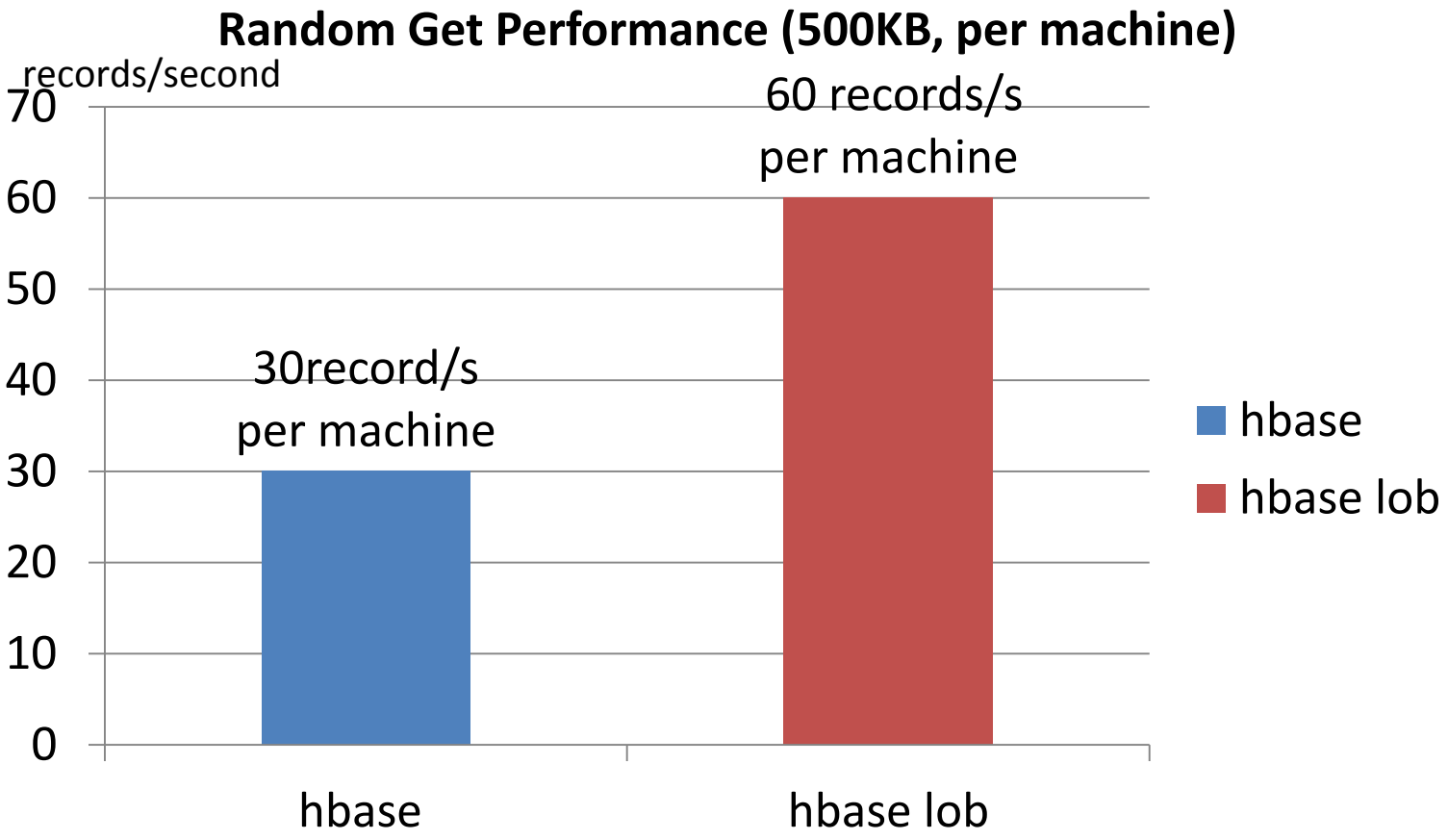
Under high stress

For 6 Clients (500KB, no presplit, 8 threads for each client, , no WAL, 3 replications, pre-split):

Insert: Table pre-split to 32 regions		Max Delay
Hbase		88 s
HBase LOB(large object storage)		**12 s**

200% Random Get performance

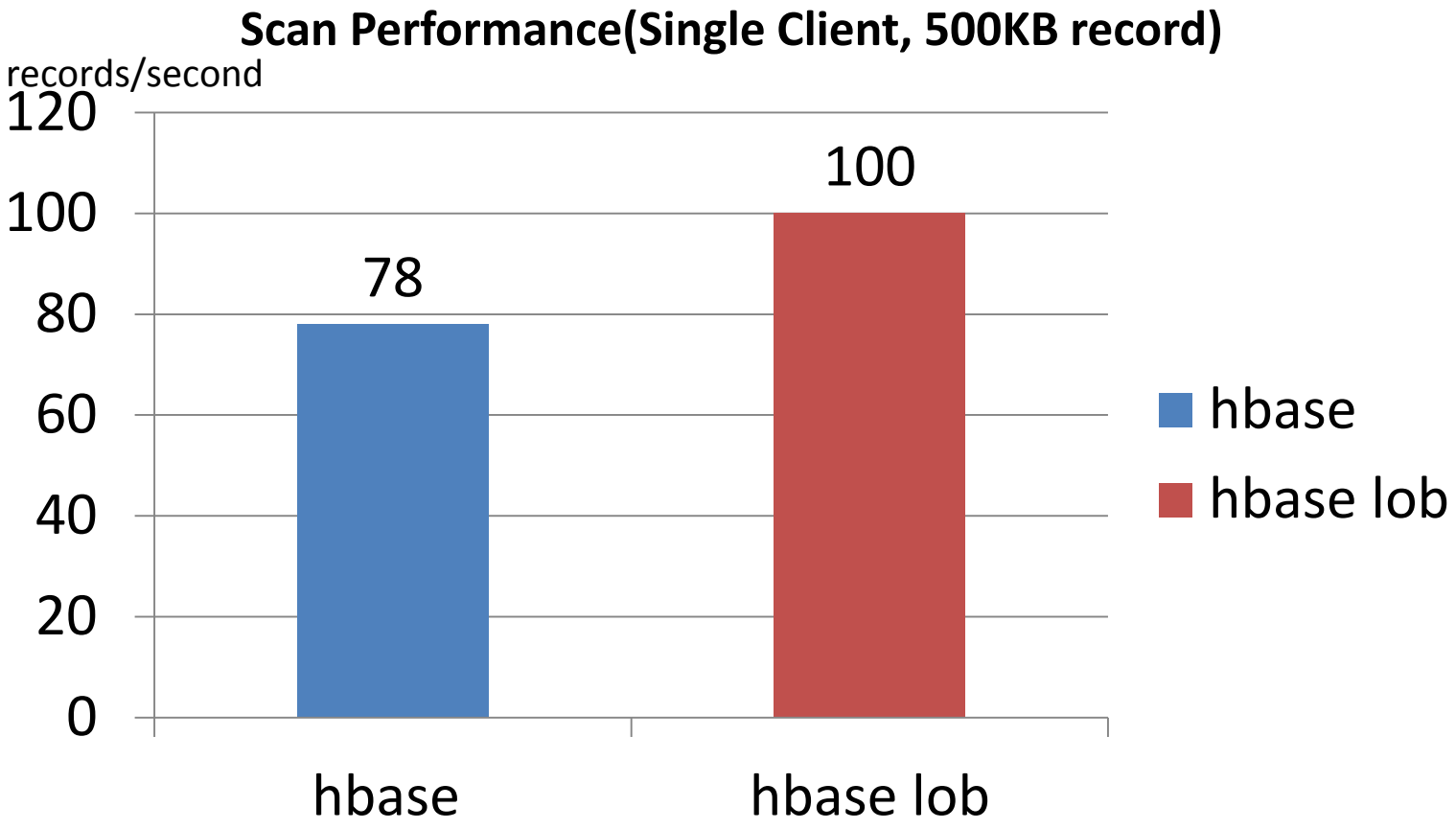
Test7: Random Get



Test setup: (intel-01 cluster, 6 machines, E5-2620, 24core, 48G memory). Pre-splitted to 32 regions. 6 Client, each client has 1 threads, each client will get 1000 records.

130% Sequential Read Performance

Test8: Sequential scan performance



Test setup: (intel-01 cluster, 6 machines, E5-2620, 24core, 48G memory). Pre-splitted to 32 regions. single Client, the client has 1 threads, Scan 10000 500KB records.

Less CPU and Memory

LOB can scale infinitely

- LOB can store PB level data without too much stress on CPU or memory.
- The performance will not drop as data scale increase.
- The memory & CPU usage will increase very slowly to keep some index data.
- High availability

  Eliminate timeout exception during insertion because of compaction, split or load balancing. **Scale infinitely** for big record storage

Transparent to user

Transparent to user

- No client code change needed.
- Same insertion process, same scan api.
- All HBase functions remains functional.

  - WAL

  - multi-version consistency control.

LOB Directory layout in HDFS

/hbase
  /blobstore
    /table
      /familyName
        /date(example: 20120101)
          /LOB file
        /LOB file

LOB File Name Schema: region_id_hash + record count + uuid

Examples

Enable LOB for special column family

LOB can only be enabled during creation of table.

```
HTableDescriptor htd = new HTableDescriptor(“lobtest");
HColumnDescriptor column = new HColumnDescriptor("f");
column.setTimeToLive(24 * 3600); //1 day
column.setLobStoreEnabled(true);
htd.addFamily(column);
admin.createTable(htd);

//All records which is in this column family will be stored in LOB.
```

How to Scan

- The LOB is totally transparent to user. There is no need to do anything to scan the LOB.

```
Scan scan = new Scan();
 ResultScanner scanner = table.getScanner(scan);
Result[] results = scanner.next(limit);
```

How to clean the data

sudo -u hbase hbase org.apache.hadoop.hbase.blobstore.compactions.Sweeper

Will check and sweep all tables in the LOB when it is obsolete.

sudo -u hbase hbase org.apache.hadoop.hbase.blobstore.compactions.Sweeper [TableName] [LobColumnFamily]

Will check and sweep the specified column family.