# MOB User Guide

Data comes in many sizes, and it is convenient to save the binary data like images, documents into the HBase. While HBase can handle binary objects with cells that are 1 byte to 10MB long, HBase's normal read and write paths are optimized for values smaller than 100KB in size. When HBase deals with large numbers of values > 100kb and up to ~10MB of data, it encounters performance degradations due to write amplification caused by splits and compactions. HBase 2.0+ has added support for better managing large numbers of \*Medium Objects\* (MOBs) that maintains the same high performance, strongly consistently characteristics with low operational overhead.

To enable the feature, one must enable and configure the mob components in each region server and enable the mob feature on particular column families during table creation or table alter. Also in the preview version of this feature, the admin must setup periodic processes that re-optimize the layout of mob data.

## Enable and configure the mob feature on region servers

Edit hbase-site.xml, change or add the following Mob file cache properties:

|  |
| --- |
| <property>  <name>hbase.mob.file.cache.size</name>  <value>1000</value>  <description>  Number of opened file handlers to cache.  A larger value will benefit reads by provinding more file handlers per mob  file cache and would reduce frequent file opening and closing.  However, if this is set too high, this could lead to a "too many opened file handers"  The default value is 1000.  </description>  </property>  <property>  <name>hbase.mob.cache.evict.period</name>  <value>3600</value>  <description>  The amount of time in seconds before the mob cache evicts cached mob files.  The default value is 3600 seconds.  </description>  </property>  <property>  <name>hbase.mob.cache.evict.remain.ratio</name>  <value>0.5f</value>  <description>  The ratio (between 0.0 and 1.0) of files that remains cached after an eviction  is triggered when the number of cached mob files exceeds the hbase.mob.file.cache.size.  The default value is 0.5f.  </description>  </property> |

## Mob management

The mob feature introduces a new read and write path to HBase and in its current incarnation requires external tools for housekeeping and optimization. There are two tools introduced -- the expiredMobFileCleaner for handling TTLs and time based expiry of data, and the sweep tool for coalescing small mob files or mob files with many deletions or updates.

1. Clean the expired MOB data(expiredMobFileCleaner)

|  |
| --- |
| org.apache.hadoop.hbase.mob.compactions.expiredMobFileCleaner tableName familyName |

Set the mob clean delay, the default is one hour.

|  |
| --- |
| <property>  <name>hbase.mob.cleaner.delay</name>  <value>60 \* 60 \* 1000</value>  </property> |

1. Sweep tool

|  |
| --- |
| org.apache.hadoop.hbase.mob.compactions.Sweeper tableName familyName |

The properties are set as follows:

|  |
| --- |
| <property>  <description>  If there're too many cells deleted in a mob file, it's regarded  as a invalid file and needs to be re-written/merged.  If (mobFileSize-existingCellsSize)/mobFileSize>=ratio, it's regarded  as a invalid file. The default value is 0.3f.  </description>  <name>hbase.mob.compaction.invalid.file.ratio</name>  <value>0.3f</value>  </property>  <property>  <description>  If the size of a mob is less than the threshold, it's regarded as a small  file and needs to be merged. The default value is 64MB.  </description>  <name>hbase.mob.compaction.small.file.threshold</name>  <value>67108864</value>  </property>  <property>  <description>  The flush size for the memstore used by sweep job. Each sweep reducer owns such a memstore.  The default value is 128MB.  </description>  <name>hbase.mob.compaction.memstore.flush.size</name>  <value>134217728</value>  </property> |

**WARN**: the worst case of using sweep tool is: the compaction of MOB files succeeds but the update of ref (PUT) fails. That means the new MOB files have been created but failed to put the new mob file paths to HBase, so these MOB files will not be referenced by HBase.

**Tips**: Please check the yarn-site.xml, add the HBase install dir: $HBASE\_HOME/\* and HBase lib dir: $HBASE\_HOME/lib/\* to yarn.application.classpath

|  |
| --- |
| <property>     <description>Classpath for typical applications.</description>      <name>yarn.application.classpath</name>      <value>  $HADOOP\_CONF\_DIR         $HADOOP\_COMMON\_HOME/\*,$HADOOP\_COMMON\_HOME/lib/\*          $HADOOP\_HDFS\_HOME/\*,$HADOOP\_HDFS\_HOME/lib/\*,          $HADOOP\_MAPRED\_HOME/\*,$HADOOP\_MAPRED\_HOME/lib/\*          $HADOOP\_YARN\_HOME/\*,$HADOOP\_YARN\_HOME/lib/\*  $HBASE\_HOME/\*, $HBASE\_HOME/lib/\*   </value>   </property> |

## Enable the mob feature on user tables

1. Set the column family be MOB

|  |
| --- |
| HColumnDescriptor hcd = new HColumnDescriptor(“f”);  hcd.setValue(MobConstants.IS\_MOB, Bytes.toBytes(Boolean.TRUE)); |

1. Set the MOB cell size threshold, the default is 102400

|  |
| --- |
| HColumnDescriptor hcd;  hcd.setValue(MobConstants.MOB\_THRESHOLD, Bytes.toBytes(102400L); |

To a client, mob cells actions just like normal cells.

1. Put the MOB value

|  |
| --- |
| KeyValue kv = new KeyValue(row1, family, qf1, ts, KeyValue.Type.Put, value );  Put put = new Put(row1);  put.add(kv);  region.put(put); |

1. Get the MOB value

|  |
| --- |
| Scan scan = new Scan();  InternalScanner scanner = (InternalScanner) region.getScanner(scan);  scanner.next(result, limit); |

There is a special scanner mode users can use to read the raw values

1. Get the MOB reference

|  |
| --- |
| Scan scan = new Scan();  scan.setAttribute(MobConstants.MOB\_SCAN\_RAW, Bytes.toBytes(Boolean.TRUE));  InternalScanner scanner = (InternalScanner) region.getScanner(scan);  scanner.next(result, limit); |

## Run Mob Integration Test

|  |
| --- |
| sudo -u hbase hbase org.apache.hadoop.hbase.IntegrationTestIngestMOB |