title: HBase Monitoring Integration description: Sematext HBase monitoring captures all key HBase metrics with out of the box dashboards and charts. Monitor all key metrics and stats such as requests, locality, compactions, splits, flushes, read and write rates, and more. Heartbeat alerts, enabled by default, notify you when any of your nodes goes down and help you troubleshoot HBase database performance issues

Integration

- Instructions: https://apps.sematext.com/ui/howto/HBase/overview

Metrics

You can choose which of some 300 HBase metrics to collect by adjusting the HBase integration YML files once you install the HBase monitoring agent.

Metric Name Key *(Type) (Unit)*	Description
lifo mode switches**hbase.ipc.lifo.mode.switches** *(long counter)*	Total number of calls in general queue which were served from the tail of the queue
general dropped calls**hbase.ipc.general.dropped.calls** *(long counter)*	Total number of calls in general queue which were dropped by CoDel RPC executor
insecure auth fallbacks**hbase.ipc.authentication.fallbacks** *(long counter)*	Number of fallbacks to insecure authentication
ipc request exceptions**hbase.ipc.exceptions** *(long counter)*	Exceptions caused by requests
sanity check exceptions**hbase.ipc.exceptions.failed.sanity.check** *(long counter)*	Number of requests that resulted in FailedSanityCheckException
region busy exceptions**hbase.ipc.exceptions.region.too.busy** *(long counter)*	Number of requests that resulted in RegionTooBusyException
scanner reset exceptions**hbase.ipc.exceptions.scanner.reset** *(long counter)*	Number of requests that resulted in ScannerResetException
full queue exceptions**hbase.ipc.exceptions.call.queue.too.big** *(long counter)*	Call queue is full
not serving region exceptions**hbase.ipc.exceptions.not.serving.region** *(long counter)*	Number of requests that resulted in NotServingRegionException

Metric Name Key *(Type) (Unit)*	Description
order scanner next exceptions**hbase.ipc.exceptions.out.of.order.scanner.next** *(long counter)*	Number of requests that resulted in OutOfOrderScannerNextException
unknown scanner exceptions**hbase.ipc.exceptions.unknown.scanner** *(long counter)*	Number of requests that resulted in UnknownScannerException
large response exceptions**hbase.ipc.exceptions.multi.response.too.large** *(long counter)*	A response to a multi request was too large and the rest of the requests will have to be retried
region moved exceptions**hbase.ipc.exceptions.region.moved** *(long counter)*	Number of requests that resulted in RegionMovedException
ipc requests**hbase.ipc.requests** *(long counter)*	Number of requests
ipc request min size**hbase.ipc.request.size.min** *(long gauge) (bytes)*	Min Request size
ipc request max size**hbase.ipc.request.size.max** *(long gauge) (bytes)*	Max Request size
ipc requests size**hbase.ipc.requests.size** *(long counter) (bytes)*	Requests size
ipc responses**hbase.ipc.responses** *(long counter)*	Number of responses
ipc response min size**hbase.ipc.response.size.min** *(long gauge) (bytes)*	Min Response size
ipc response max size**hbase.ipc.response.size.max** *(long gauge) (bytes)*	Max Response size
ipc responses size**hbase.ipc.responses.size** *(long counter) (bytes)*	Responses size
ipc total calls**hbase.ipc.total.calls** *(long counter)*	Total calls
ipc total call min time**hbase.ipc.total.call.time.min** *(long gauge) (ms)*	Total call min time including both queued and processing time
ipc total call max time**hbase.ipc.total.call.time.max** *(long gauge) (ms)*	Total call max time including both queued and processing time

Metric Name Key *(Type) (Unit)*	Description
ipc total calls time**hbase.ipc.total.calls.time** *(long counter) (ms)*	Total calls time including both queued and processing time
ipc queue size**hbase.ipc.queue.bytes** *(long gauge) (bytes)*	Number of bytes in the call queues; request has been read and parsed and is waiting to run or is currently being executed
ipc general queue calls**hbase.ipc.queue.size** *(long gauge)*	Number of calls in the general call queue; parsed requests waiting in scheduler to be executed
ipc replication queue calls**hbase.ipc.queue.replication.size** *(long gauge)*	Number of calls in the replication call queue waiting to be run
ipc priority queue calls**hbase.ipc.queue.priority.size** *(long gauge)*	Number of calls in the priority call queue waiting to be run
ipc open connections**hbase.ipc.connections.open** *(long gauge)*	Number of open connections
ipc active handlers**hbase.ipc.handlers.active** *(long gauge)*	Total number of active rpc handlers
ipc queue calls**hbase.ipc.queue.calls** *(long counter)*	Queue Calls
ipc queue call min time**hbase.ipc.queue.call.time.min** *(long gauge) (ms)*	Queue Call Min Time
ipc queue call max time**hbase.ipc.queue.call.time.max** *(long gauge) (ms)*	Queue Call Max Time
ipc authentication failures**hbase.ipc.authentication.failures** *(long counter)*	Number of authentication failures
ipc authorization failures**hbase.ipc.authorization.failures** *(long counter)*	Number of authorization failures
ipc authentication successes**hbase.ipc.authentication.successes** *(long counter)*	Number of authentication successes
ipc authorization successes**hbase.ipc.authorization.successes** *(long counter)*	Number of authorization successes

Metric Name Key *(Type) (Unit)*	Description
ipc processing calls**hbase.ipc.process.calls** *(long counter)*	Processing calls
ipc processing call min time**hbase.ipc.process.call.time.min** *(long gauge) (ms)*	Processing call min time
ipc processing call max time**hbase.ipc.process.call.time.max** *(long gauge) (ms)*	Processing call max time
ipc sent bytes**hbase.ipc.bytes.sent** *(long counter) (bytes)*	Number of bytes sent
ipc received bytes**hbase.ipc.bytes.received** *(long counter) (bytes)*	Number of bytes received
ipc processing calls time**hbase.ipc.process.calls.time** *(long counter) (ms)*	Processing call time
ipc queue calls time**hbase.ipc.queue.calls.time** *(long counter) (ms)*	Queue Call Time
new threads**jvm.threads.new** *(long gauge)*	Current number of NEW threads
runnable threads**jvm.threads.runnable** *(long gauge)*	Current number of RUNNABLE threads
blocked threads**jvm.threads.blocked** *(long gauge)*	Current number of BLOCKED threads
waiting threads**jvm.threads.waiting** *(long gauge)*	Current number of WAITING threads
timed waiting threads**jvm.threads.waiting.timed** *(long gauge)*	Current number of TIMED_WAITING threads
terminated threads**jvm.threads.terminated** *(long gauge)*	Current number of TERMINATED threads
fatal logs**jvm.log.fatal** *(long counter)*	Total number of FATAL logs
error logs**jvm.log.error** *(long counter)*	Total number of ERROR logs
warn logs**jvm.log.warn** *(long counter)*	Total number of WARN logs
info logs**jvm.log.info** *(long counter)*	Total number of INFO logs

Metric Name Key *(Type) (Unit)*	Description
non-heap memory used**jvm.nonheap.used** *(long gauge) (bytes)*	Current non-heap memory used
non-heap memory committed**jvm.nonheap.committed** *(long gauge) (bytes)*	Current non-heap memory committed
max non-heap memory**jvm.nonheap.size.max** *(long gauge) (bytes)*	Max non-heap memory size
heap memory**jvm.heap.used** *(long gauge) (bytes)*	Current heap memory used
heap memory commited**jvm.heap.committed** *(long gauge) (bytes)*	Current heap memory committed
max heap memory**jvm.heap.size.max** *(long gauge) (bytes)*	Max heap memory size
max memory size**jvm.memory.size.max** *(long gauge) (bytes)*	Max memory size
successful logins**hbase.ugi.login.success** *(long counter)*	Successful kerberos logins
failed logins**hbase.ugi.login.failure** *(long counter)*	Failed kerberos logins
group resolutions**hbase.ugi.groups.gets** *(long counter)*	Total number of group resolutions
failed logins latency**hbase.ugi.login.failure.time** *(long counter) (ms)*	Failed kerberos logins latency
successful logins latency**hbase.ugi.login.success.time** *(long counter) (ms)*	Successful kerberos logins latency
group resolutions time**hbase.ugi.groups.gets.time** *(long counter) (ms)*	Time for group resolution
oldest regions in transition**hbase.master.rit.oldest** *(long gauge) (ms)*	Timestamp of the oldest Region In Transition
total duration regions in transition**hbase.master.rit.duration** *(double counter) (ms)*	Total durations in milliseconds for all Regions in Transition

Metric Name Key *(Type) (Unit)*	Description
regions in transition**hbase.master.rit.count** *(long gauge)*	Current number of Regions In Transition
regions in transition long time**hbase.master.rit.count.overthreshold** *(long gauge)*	Current number of Regions In Transition over threshold time
bulk assigns**hbase.master.assigns.bulk** *(long counter)*	Number of bulk assign operations
bulk assign min time**hbase.master.assigns.bulk.time.min** *(long gauge) (ms)*	Min time for bulk assign operation
bulk assign max time**hbase.master.assigns.bulk.time.max** *(long gauge) (ms)*	Max time for bulk assign operation
master assigns**hbase.master.assigns** *(long counter)*	Number of assign operations
assign min time**hbase.master.assigns.time.min** *(long gauge) (ms)*	Min time for assign operation
assign max time**hbase.master.assigns.time.max** *(long gauge) (ms)*	Max time for assign operation
bulk assigns time**hbase.master.assigns.bulk.time** *(double counter) (ms)*	Time for bulk assign operations
assigns time**hbase.master.assigns.time** *(double counter) (ms)*	Time for assign operations
balancer ops**hbase.master.balancer.ops** *(long counter)*	Balancer invocations
balance min time**hbase.master.balancer.time.min** *(long gauge) (ms)*	Min time for balance operation
balance max time**hbase.master.balancer.time.max** *(long gauge) (ms)*	Max time for balance operation
balancer misc invocations**hbase.master.balancer.misc.invocations** *(long counter)*	Balancer misc invocations
balances time**hbase.master.balancer.time** *(long counter) (ms)*	Time for balance operations

Metric Name Key *(Type) (Unit)*	Description
wal splits**hbase.master.hlog.splits** *(long counter)*	Number of WAL files splits
wal split min time**hbase.master.hlog.split.time.min** *(long gauge) (ms)*	Min time it takes to finish WAL.splitLog()
wal split max time**hbase.master.hlog.split.time.max** *(long gauge) (ms)*	Max time it takes to finish WAL.splitLog()
meta wal splits**hbase.master.hlog.meta.splits** *(long counter)*	Meta WAL files splits
meta wal split min time**hbase.master.hlog.meta.split.time.min** *(long gauge) (ms)*	Min time it takes to finish splitMetaLog()
meta wal split max time**hbase.master.hlog.meta.split.time.max** *(long gauge) (ms)*	Max time it takes to finish splitMetaLog()
meta wal split min size**hbase.master.hlog.meta.split.size.min** *(long gauge) (bytes)*	Min size of hbase:meta WAL files being split
meta wal split max size**hbase.master.hlog.meta.split.size.max** *(long gauge) (bytes)*	Max size of hbase:meta WAL files being split
wal split min size**hbase.master.hlog.split.size.min** *(long gauge) (bytes)*	Min size of WAL files being split
wal split max size**hbase.master.hlog.split.size.max** *(long gauge) (bytes)*	Max size of WAL files being split
meta wal splits size**hbase.master.hlog.meta.splits.size** *(long counter) (bytes)*	Size of hbase:meta WAL files being split
meta wal splits time**hbase.master.hlog.meta.splits.time** *(long counter) (ms)*	Time it takes to finish splitMetaLog()
wal splits time**hbase.master.hlog.splits.time** *(long counter) (ms)*	Time it takes to finish WAL.splitLog()
wal splits size**hbase.master.hlog.splits.size** *(long counter) (bytes)*	Size of WAL files being split
plan splits**hbase.master.plan.splits** *(long gauge)*	Number of Region Split Plans executed

Metric Name Key *(Type) (Unit)*	Description
plan merges**hbase.master.plan.merges** *(long gauge)*	Number of Region Merge Plans executed
region servers**hbase.master.servers.region** *(long gauge)*	Number of RegionServers
dead region servers**hbase.master.servers.region.dead** *(long gauge)*	Number of dead RegionServers
requests**hbase.master.requests** *(long counter)*	Number of cluster requests
average load**hbase.master.load** *(double gauge)*	Average Load
snapshots restores**hbase.master.snapshots.restores** *(long counter)*	Number of restoreSnapshot() calls
snapshot restore min time**hbase.master.snapshots.restore.time.min** *(long gauge) (ms)*	Min time it takes to finish restoreSnapshot() call
snapshot restore max time**hbase.master.snapshots.restore.time.max** *(long gauge) (ms)*	Max time it takes to finish restoreSnapshot() call
snapshots clones**hbase.master.snapshots.clones** *(long counter)*	Number of cloneSnapshot() calls
snapshots clone min time**hbase.master.snapshots.clone.time.min** *(long gauge) (ms)*	Min time it takes to finish cloneSnapshot() call
snapshots clone max time**hbase.master.snapshots.clone.time.max** *(long gauge) (ms)*	Max time it takes to finish cloneSnapshot() call
snapshots**hbase.master.snapshots** *(long counter)*	Number of snapshot() calls
snapshot min time**hbase.master.snapshot.time.min** *(long gauge) (ms)*	Max time it takes to finish snapshot() call
snapshot max time**hbase.master.snapshot.time.max** *(long gauge) (ms)*	Max time it takes to finish snapshot() call
snapshots restores time**hbase.master.snapshots.restores.time** *(double counter) (ms)*	Time it takes to finish restoreSnapshot() calls

Metric Name Key *(Type) (Unit)*	Description
snapshots clones time**hbase.master.snapshots.clones.time** *(double counter) (ms)*	Time it takes to finish cloneSnapshot() calls
snapshots time**hbase.master.snapshots.time** *(double counter) (ms)*	Time it takes to finish snapshot() calls
completed logs**hbase.rs.replication.completed.logs** *(long gauge)*	Source completed logs
repeated log files size**hbase.rs.replication.repeated.log.file.size** *(long gauge) (bytes)*	Source repeated log files size
restarted load readings**hbase.rs.replication.restarted.log.reads** *(long gauge)*	Source restarted load readings
closed logs**hbase.rs.replication.closed.logs.with.unknown.file.length** *(long gauge)*	Source closed logs with unknows file length
uncleanly closed logs**hbase.rs.replication.uncleanly.closed.logs** *(long gauge)*	Source uncleanly closed logs
ignored uncleanly closed logs size**hbase.rs.replication.ignored.uncleanly.closed.log.content.size** *(long gauge) (bytes)*	Source ignored uncleanly closed logs
log queue**hbase.rs.replication.log.queue** *(long gauge)*	Source log queue
log edits read**hbase.rs.replication.log.edits.read** *(long counter)*	Source log edits read
log edits filtered**hbase.rs.replication.log.edits.filtered** *(long counter)*	Source log edits filtered
shipped batches**hbase.rs.replication.batches.shipped** *(long counter)*	Source shipped batches
shipped operations**hbase.rs.replication.ops.shipped** *(long counter)*	Source shipped operations
shipped size**hbase.rs.replication.batches.shipped.size** *(long counter) (bytes)*	Source shipped size

Metric Name Key *(Type) (Unit)*	Description
log read size**hbase.rs.replication.log.edits.read.bytes** *(long counter) (bytes)*	Source log read size
rs tables**hbase.rs.tables** *(long gauge)*	Number of tables in the metrics system
rs read requests**hbase.rs.table.read.requests** *(long counter)*	Number of read requests
rs write requests**hbase.rs.table.write.requests** *(long counter)*	Number of write requests
rs memstore size**hbase.rs.table.memstore.size** *(long gauge) (bytes)*	The size of memory stores
rs store files size**hbase.rs.table.store.files.size** *(long gauge) (bytes)*	The size of store files size
rs table size**hbase.rs.table.size** *(long gauge) (bytes)*	Total size of the table in the region server
compacted in size**hbase.rs.compacted.in.size** *(long counter) (bytes)*	Total number of bytes that is read for compaction both major and minor
major compacted out size**hbase.rs.major.compacted.out.bytes** *(long counter)*	
flushed memstore size**hbase.rs.flushed.memostore.size** *(long counter) (bytes)*	Total number of bytes of cells in memstore from flush
compacted out size**hbase.rs.compacted.out.size** *(long counter) (bytes)*	Total number of bytes that is output from compaction major only
splits requests**hbase.rs.split.requests** *(long counter)*	Number of splits requested
flushed out size**hbase.rs.flushed.out.size** *(long counter) (bytes)*	Total number of bytes written from flush
cache failed insertions**hbase.rs.cache.block.failed.insertions** *(long counter)*	Number of times that a block cache insertion failed. Usually due to size restrictions
cache hits rate**hbase.rs.cache.block.hits.rate** *(double gauge)*	Percent of block cache requests that are hits

Metric Name Key *(Type) (Unit)*	Description
cache primary evictions**hbase.rs.cache.block.primary.evictions** *(long counter)*	Count of the number of blocks evicted from primary replica in the block cache
cache primary misses**hbase.rs.cache.block.primary.misses** *(long counter)*	Number of requests for a block of primary replica that missed the block cache
cache primary hits**hbase.rs.cache.block.primary.hits** *(long counter)*	Count of hit on primary replica in the block cache
large compaction queue**hbase.rs.large.compaction.queue** *(long gauge)*	Length of the queue for compactions with input size larger than throttle threshold (2.5GB by default)
small compactions queue**hbase.rs.small.compactions.queue** *(long gauge)*	Length of the queue for compactions
splits queue**hbase.rs.splits.queue** *(long gauge)*	Length of the queue for splits
secondary regions local files rate**hbase.rs.files.local.rate.secondary.regions** *(double gauge)*	The percent of HFiles used by secondary regions that are stored on the local hdfs data node
rpc mutation requests**hbase.rs.rpc.mutate.requests** *(long counter)*	Number of rpc mutation requests this RegionServer has answered
rpc multi requests**hbase.rs.rpc.multi.requests** *(long counter)*	Number of rpc multi requests this RegionServer has answered
rpc scan requests**hbase.rs.rpc.scan.requests** *(long counter)*	Number of rpc scan requests this RegionServer has answered
rpc get requests**hbase.rs.rpc.get.requests** *(long counter)*	Number of rpc get requests this RegionServer has answered
avg rs region size**hbase.rs.region.size.avg** *(long gauge) (bytes)*	Average region size over the RegionServer including memstore and storefile sizes
reference files**hbase.rs.reference.files** *(long gauge)*	Number of reference file on this RegionServer
blocked requests**hbase.rs.blocked.requests** *(long counter)*	The number of blocked requests because of memstore size is larger than blockingMemStoreSize
cache trailer hits**hbase.rs.cache.block.trailer.hits** *(long counter)*	Block cache trailer hits

Metric Name Key *(Type) (Unit)*	Description
cache delete family bloom hits *(long counter)*	Block cache delete family bloom hits **hbase.rs.cache.block.delete.family.bloom.hits**
cache general bloom meta hits *(long counter)*	Block cache general bloom meta hits **hbase.rs.cache.block.general.bloom.meta.hits**
cache file info hits *(long counter)*	Block cache file info hits **hbase.rs.cache.block.file.info.hits**
cache intermediate index hits *(long counter)*	Block cache intermediate index hits **hbase.rs.cache.block.intermediate.index.hits**
cache root index hits *(long counter)*	Block cache root index hits **hbase.rs.cache.block.root.index.hits**
cache meta hits *(long counter)*	Block cache meta hits **hbase.rs.cache.block.meta.hits**
cache bloom chunk hits *(long counter)*	Block cache bloom chunk hits count **hbase.rs.cache.block.bloom.chunk.hits**
cache leaf index hits *(long counter)*	Block cache leaf index hits **hbase.rs.cache.block.leaf.index.hits**
cache data hits *(long counter)*	Block cache data hits **hbase.rs.cache.block.data.hits**
cache trailer misses *(long counter)*	Block cache trailer misses **hbase.rs.cache.block.trailer.misses**
cache delete family bloom misses *(long counter)*	Block cache delete family bloom misses **hbase.rs.cache.block.delete.family.bloom.misses**
cache general bloom meta misses *(long counter)*	Block cache general bloom meta misses **hbase.rs.cache.block.general.bloom.meta.misses**
cache file info misses *(long counter)*	Block cache file info misses **hbase.rs.cache.block.file.info.misses**
cache intermediate index misses *(long counter)*	Block cache intermediate index misses **hbase.rs.cache.block.intermediate.index.misses**

Metric Name Key *(Type) (Unit)*	Description
cache root index misses**hbase.rs.cache.block.root.index.misses** *(long counter)*	Block cache root index misses
cache meta misses**hbase.rs.cache.block.meta.misses** *(long counter)*	Block cache meta misses
cache bloom chunk misses**hbase.rs.cache.block.bloom.chunk.misses** *(long counter)*	Block cache bloom chunk misses
cache leaf index misses**hbase.rs.cache.block.leaf.index.misses** *(long counter)*	Block cache leaf index misses
cache data misses**hbase.rs.cache.block.data.misses** *(long counter)*	Block cache data misses
success splits**hbase.rs.success.splits** *(long counter)*	Number of successfully executed splits
rs regions**hbase.rs.regions** *(long gauge)*	Number of regions
rs stores**hbase.rs.stores** *(long gauge)*	Number of Stores
hlog files**hbase.rs.files.hlog** *(long gauge)*	Number of WAL Files
hlog files size**hbase.rs.files.hlog.size** *(long gauge) (bytes)*	Size of all WAL Files
stores files**hbase.rs.stores.files** *(long gauge)*	Number of Store Files
memstore size**hbase.rs.memstore.size** *(long gauge) (bytes)*	Size of the memstore
stores files size**hbase.rs.stores.files.size** *(long gauge) (bytes)*	Size of storefiles being served
total requests**hbase.rs.total.requests** *(long counter)*	Total number of requests this RegionServer has answered; increments the count once for EVERY access whether an admin operation
rs read requests**hbase.rs.requests.read** *(long counter)*	Number of read requests with non-empty Results that this RegionServer has answered
rs write requests**hbase.rs.requests.write** *(long counter)*	Number of mutation requests this RegionServer has answered

Metric Name Key *(Type) (Unit)*	Description
failed mutates**hbase.rs.ops.mutates.failed** *(long counter)*	Number of Check and Mutate calls that failed the checks
passed mutates**hbase.rs.ops.mutates.passed** *(long counter)*	Number of Check and Mutate calls that passed the checks
store files indexes size**hbase.rs.stores.index.size** *(long gauge) (bytes)*	Size of indexes in storefiles on disk
static indices size**hbase.rs.static.index.size** *(long gauge) (bytes)*	Uncompressed size of the static indices
static bloom filters size**hbase.rs.static.bloom.size** *(long gauge) (bytes)*	Uncompressed size of the static bloom filters
mutations without wal**hbase.rs.ops.mutates.nowal** *(long counter)*	Number of mutations that have been sent by clients with the write ahead logging turned off
mutations size without wal**hbase.rs.ops.mutates.nowal.size** *(long counter) (bytes)*	Size of data that has been sent by clients with the write ahead logging turned off
local files rate**hbase.rs.files.local.rate** *(long gauge)*	The percent of HFiles that are stored on the local hdfs data node
compaction queue**hbase.rs.compaction.queue** *(long gauge)*	Length of the queue for compactions
flush queue**hbase.rs.flush.queue** *(long gauge)*	Length of the queue for region flushes
rs cache free size**hbase.rs.cache.block.free.size** *(long gauge) (bytes)*	
cache blocks**hbase.rs.cache.block.count** *(long gauge)*	Number of block in the block cache
rs cache size**hbase.rs.cache.block.size** *(long gauge) (bytes)*	Size of the block cache
rs cache hits**hbase.rs.cache.block.hits** *(long counter)*	Count of the hit on the block cache
rs cache misses**hbase.rs.cache.block.misses** *(long counter)*	Number of requests for a block that missed the block cache

Metric Name Key *(Type) (Unit)*	Description
rs cache evictions**hbase.rs.cache.block.evictions** *(long counter)*	Count of the number of blocks evicted from the block cache (Not including blocks evicted because of HFile removal)
rs cache express hits rate**hbase.rs.cache.block.hits.express.rate** *(long gauge)*	The percent of the time that requests with the cache turned on hit the cache
blocked updates**hbase.rs.updates.blocked.time** *(long counter)*	Number of MS updates have been blocked so that the memstore can be flushed
flushed cells**hbase.rs.flushed.cells** *(long counter)*	The number of cells flushed to disk
compaction cells**hbase.rs.compaction.cells** *(long counter)*	The number of cells processed during minor compactions
major compaction cells**hbase.rs.compaction.major.cells** *(long counter)*	The number of cells processed during major compactions
flushed cells size**hbase.rs.flushed.cells.size** *(long counter) (bytes)*	The total amount of mob cells flushed to disk
compaction cells size**hbase.rs.compaction.cells.size** *(long counter) (bytes)*	The total amount of data processed during major compactions
major compaction cells size**hbase.rs.compaction.major.cells.size** *(long counter) (bytes)*	The total amount of data processed during major compactions
hedged reads**hbase.rs.reads.hedged** *(long counter)*	The number of times we started a hedged read
hedged reads wins**hbase.rs.reads.hedged.wins** *(long counter)*	The number of times we started a hedged read and a hedged read won
mob cached files**hbase.rs.mob.cache.files** *(long gauge)*	The count of cached mob files
mob cache files accesses**hbase.rs.mob.cache.files.accesses** *(long counter)*	The count of accesses to the mob file cache
mob cache files misses**hbase.rs.mob.cache.files.misses** *(long counter)*	The count of misses to the mob file cache
mob cache files evictions**hbase.rs.mob.cache.files.evictions** *(long counter)*	The number of items evicted from the mob file cache

Metric Name Key *(Type) (Unit)*	Description
mob flushes**hbase.rs.mob.flushes** *(long counter)*	The number of the flushes in mob-enabled stores
flushed cells**hbase.rs.mob.flushed.cells** *(long counter)*	The number of mob cells flushed to disk
mob flushed cells size**hbase.rs.mob.flushed.cells.size** *(long counter) (bytes)*	The total amount of mob cells flushed to disk
scanned cells**hbase.rs.mob.scan.cells** *(long counter)*	The number of scanned mob cells
scanned cells size**hbase.rs.mob.scan.cells.size** *(long counter) (bytes)*	The total amount of scanned mob cells
mob cache files hit rate**hbase.rs.mob.cache.files.hits.rate** *(long gauge)*	The hit percent to the mob file cache
rs appends**hbase.rs.ops.appends** *(long counter)*	The number of batches containing puts
rs deletes**hbase.rs.ops.deletes** *(long counter)*	The number of batches containing delete(s)
rs mutates**hbase.rs.ops.mutates** *(long counter)*	The number of Mutates
rs gets**hbase.rs.ops.gets** *(long counter)*	The number of Gets
rs replays**hbase.rs.ops.replays** *(long counter)*	The numbers of Replays
rs increments**hbase.rs.ops.increments** *(long counter)*	The number of Increments
rs slow appends**hbase.rs.ops.appends.slow** *(long counter)*	The number of batches containing puts that took over 1000ms to complete
rs slow deletes**hbase.rs.ops.deletes.slow** *(long counter)*	The number of batches containing delete(s) that took over 1000ms to complete
rs slow increments**hbase.rs.ops.increments.slow** *(long counter)*	The number of Increments that took over 1000ms to complete
rs slow gets**hbase.rs.ops.gets.slow** *(long counter)*	The number of Gets that took over 1000ms to complete
rs slow puts**hbase.rs.ops.puts.slow** *(long counter)*	The number of batches containing puts that took over 1000ms to complete

Metric Name Key *(Type) (Unit)*	Description
rs scan min size**hbase.rs.ops.scan.size.min** *(long gauge) (bytes)*	Min scan size
rs scan max size**hbase.rs.ops.scan.size.max** *(long gauge) (bytes)*	Max scan size
rs flushes**hbase.rs.ops.flushes** *(long counter)*	Number of flushes
rs flush output min size**hbase.rs.ops.flushes.out.size.min** *(long gauge) (bytes)*	Min number of bytes in the resulting file for a flush
rs flush output max size**hbase.rs.ops.flushes.out.size.max** *(long gauge) (bytes)*	Max number of bytes in the resulting file for a flush
rs compaction input min size**hbase.rs.ops.major.compaction.in.size.min** *(long gauge) (bytes)*	Compaction min total input file size major only
rs compaction input max size**hbase.rs.ops.major.compaction.in.size.max** *(long gauge) (bytes)*	Compaction max total input file size major only
rs compactions**hbase.rs.ops.compactions** *(long counter) (bytes)*	Compactions both major and minor
rs compactions input min size**hbase.rs.ops.compactions.in.size.min** *(long gauge) (bytes)*	Min compaction total input file size both major and minor
rs compactions input max size**hbase.rs.ops.compactions.in.size.max** *(long gauge) (bytes)*	Max compaction total input file size both major and minor
rs flush min time**hbase.rs.ops.flushes.time.min** *(long gauge) (ms)*	Min time for memstore flush
rs flush max time**hbase.rs.ops.flushes.time.max** *(long gauge) (ms)*	Max time for memstore flush
rs compactions output min size**hbase.rs.ops.compactions.out.size.min** *(long gauge) (bytes)*	Min compaction total output file size
rs compactions output max size**hbase.rs.ops.compactions.out.size.max** *(long gauge) (bytes)*	Max compaction total output file size both major and minor
rs splits**hbase.rs.ops.splits** *(long counter)*	The number of Splits

Metric Name Key *(Type) (Unit)*	Description
rs split min time**hbase.rs.ops.split.time.min** *(long gauge) (ms)*	Min split time
rs split max time**hbase.rs.ops.split.time.max** *(long gauge) (ms)*	Max split time
rs flush memstore min size**hbase.rs.ops.flushes.memstore.size.min** *(long gauge) (bytes)*	Min number of bytes in the memstore for a flush
rs flush memstore max size**hbase.rs.ops.flushes.memstore.size.max** *(long gauge) (bytes)*	Max number of bytes in the memstore for a flush
rs scans**hbase.rs.ops.scans** *(long counter)*	The number of Scans
rs scan min time**hbase.rs.ops.scan.time.min** *(long gauge) (ms)*	Min scan time
rs scan max time**hbase.rs.ops.scan.time.max** *(long gauge) (ms)*	Max scan time
rs major compactions**hbase.rs.ops.major.compactions** *(long counter)*	Compactions major only
rs major compaction min time**hbase.rs.ops.major.compaction.time.min** *(long gauge) (ms)*	Min time for compaction major only
rs major compaction max time**hbase.rs.ops.major.compaction.time.max** *(long gauge) (ms)*	Max time for compaction major only
rs major compactions time**hbase.rs.ops.major.compactions.time** *(long counter) (ms)*	Time for compactions major only
rs scans time**hbase.rs.ops.scans.time** *(long counter) (ms)*	Scans time
rs flushes memstore size**hbase.rs.ops.flushes.memstore.size** *(long counter) (bytes)*	Number of bytes in the memstore for a flushes
rs major compactions input files**hbase.rs.ops.major.compactions.in.files** *(long counter)*	Compactions input number of files major only
rs compactions input files**hbase.rs.ops.compactions.in.files** *(long counter)*	Compactions input number of files both major and minor

Metric Name Key *(Type) (Unit)*	Description
rs splits time**hbase.rs.ops.splits.time** *(long counter) (ms)*	Splits time
rs compactions output size**hbase.rs.ops.compactions.out.size** *(long counter) (bytes)*	Compaction total output file sizes both major and minor
rs major compactions.output size**hbase.rs.ops.major.compactions.out.size** *(long counter) (bytes)*	Compactions total output file sizes major only
rs compactions output files**hbase.rs.ops.compactions.out.files** *(long counter) (bytes)*	Compactions output number of files both major and minor
rs flushes time**hbase.rs.ops.flushes.time** *(long counter) (ms)*	Time for memstore flushes
rs major compactions output files**hbase.rs.ops.major.compactions.out.files** *(long counter)*	Compactions output number of files major only
rs compactions input size**hbase.rs.ops.compactions.in.size** *(long counter) (bytes)*	Compactions total input file sizes both major and minor
rs major compactions input size**hbase.rs.ops.major.compactions.in.size** *(long counter) (bytes)*	Compactions total input file sizes major only
rs flushes output size**hbase.rs.ops.flushes.out.size** *(long counter) (bytes)*	Number of bytes in the resulting file for a flushes
rs scans size**hbase.rs.ops.scans.size** *(long counter) (bytes)*	Scans size
wal roll requests**hbase.rs.wal.roll.requests** *(long counter)*	How many times a log roll has been requested total
wal written size**hbase.rs.wal.written.size** *(long counter) (bytes)*	Size of the data written to the WAL
wal low replica roll requests**hbase.rs.wal.low.replica.roll.requests** *(long counter)*	How many times a log roll was requested due to too few DN's in the write pipeline
wal syncs**hbase.rs.wal.syncs** *(long counter)*	The number of syncs the WAL to HDFS
wal sync min time**hbase.rs.wal.sync.time.min** *(long gauge) (ms)*	Min time it took to sync the WAL to HDFS

Metric Name Key *(Type) (Unit)*	Description
wal sync max time**hbase.rs.wal.sync.time.max** *(long gauge) (ms)*	Max time it took to sync the WAL to HDFS
wal append min size**hbase.rs.wal.append.size.min** *(long gauge) (bytes)*	Min size of the data appended to the WAL
wal append max size**hbase.rs.wal.append.size.max** *(long gauge) (bytes)*	Max size of the data appended to the WAL
wal append min time**hbase.rs.wal.append.time.min** *(long gauge) (ms)*	Min time an append to the WAL took
wal append max time**hbase.rs.wal.append.time.max** *(long gauge) (ms)*	Max time an append to the WAL took
wal slow appends**hbase.rs.wal.appends.slow** *(long counter)*	Number of appends that were slow
wal appends**hbase.rs.wal.appends** *(long counter)*	Number of appends to the write ahead log
wal syncs time**hbase.rs.wal.syncs.time** *(long counter) (ms)*	The time it took to syncs the WAL to HDFS
wal appends size**hbase.rs.wal.appends.size** *(long counter) (bytes)*	Size of the data appended to the WAL
wal appends time**hbase.rs.wal.appends.time** *(long counter) (ms)*	Time an appends to the WAL took
applied replication batches**hbase.rs.replication.batches.applied** *(long counter)*	Applied replication batches
applied replication ops**hbase.rs.replication.ops.applied** *(long counter)*	Applied replication ops
applied replication hfiles**hbase.rs.replication.hfiles.applied** *(long counter)*	Applied replication hfiles

FAQ

** How do I enable JMX in HBase **

Please see HBase Metrics page for instructions.

** Do I need to add a separate Monitoring App for each HBase server/node I want to monitor **

No, one App is enough. To monitor N HBase servers that belong to the same cluster create just a single Monitoring App and use its Token in the agent configuration file on all HBase servers that are a part of the same cluster. See App Guide for more info.

** Why don't some HBase metrics graphs have any data **

There could be 2 possible reasons:

1. Some metrics are for RegionServers (HBase slaves), some for HBase Master. Thus, if you select the Master node in the UI, graphs that contain Slave-specific metrics will be blank and vice-versa.
2. Different versions of HBase provide different metrics. Thus, if you have an older version of HBase, it may not be providing all metrics that Sematext Monitoring collects and graphs.