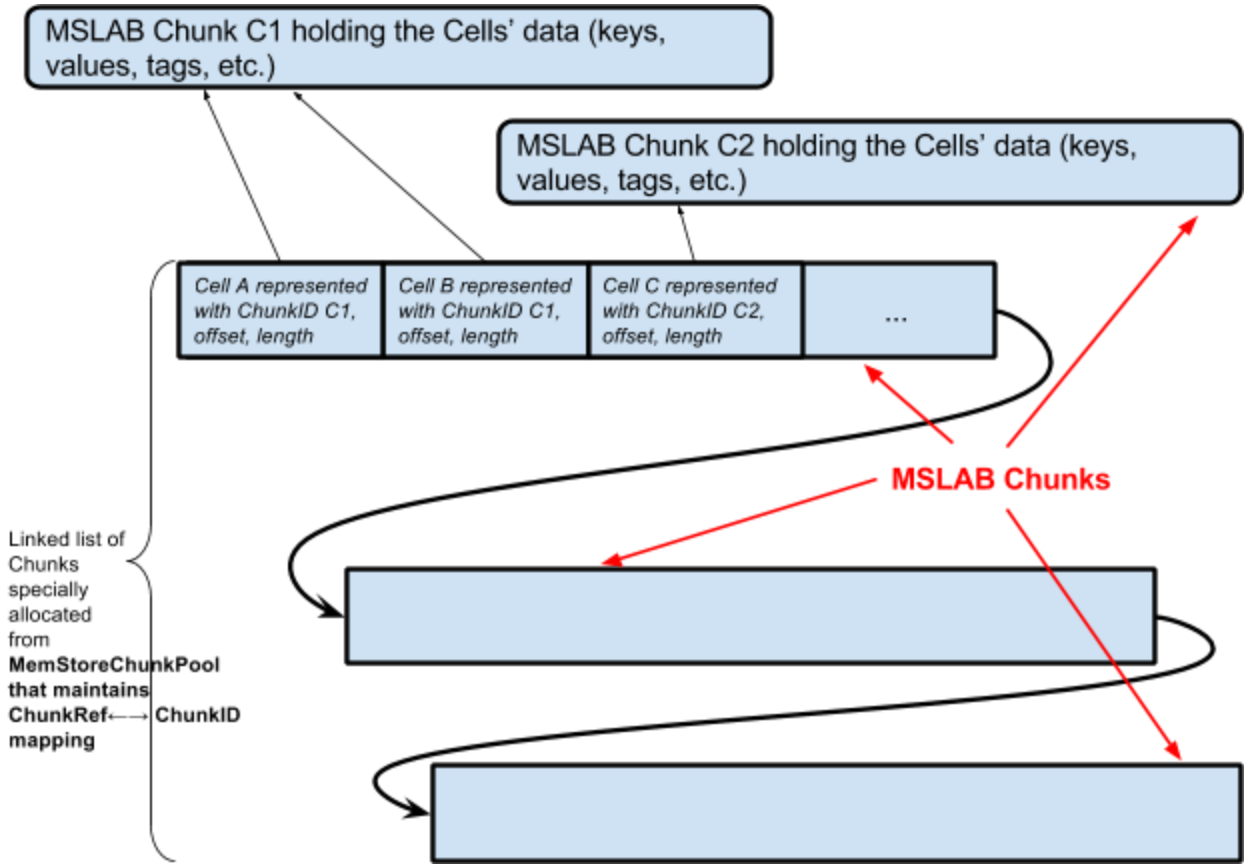
CellChunkMap Revived

Current Implementation Status:

CompactingMemStore now has only CellArrayMap as an ImmutableSegment's index (CellSet). We want to add to the CellSet of ImmutableSegment an option to be also CellChunkMap. Basic CellChunkMap implementation code that we already have is as following:



Memory Management:

Because Java doesn't maintain any pointers we need to be able to access any chunk using its ID. The single entity managing the ChunkIDs should be the one that allocates the chunks. From here the MemStoreChunkPool:

1. allocates every chunk whether it is from pool or not
2. assigns every chunk its ID upon allocation
3. keeps the mapping from IDs to chunks and translates chunk IDs to chunk references

Conclusions:

1. CellChunkMap works only when MSLAB is turned on [*According to Stack: MSLAB is on always in master branch, right? If not, let's make it so by fiat in 2.0.*]
2. CellChunkMap works only when MemStoreChunkPool is on

Flattening:

Flattening of a segment *S* is a process of replacing *S*'s CellSet from memory consuming ConcurrentSkipListMap (CSLM) to flat (space efficient) CellArrayMap or CellChunkMap. The CSLM index is scanned and each Cell is referenced in the new index, the new index is created from scratch. However, this is not working for CellChunkMap. Currently, the Cell is not aware whether it is allocated on MSLAB or from JVM heap. The Cell is not aware on which MSLAB Chunk it is allocated. Therefore when CSLM index is scanned we do not know each cell's chunk ID in order to build the CellChunkMap.

Conclusions:

1. Cell need to be aware on which chunk it is allocated, either by having chunk ID as a cell's field or by knowing to read chunk ID from the chunk going to chunk's start using offset.

Super Large Cells:

One of the questions raised, what to do with the Cells bigger than MSLAB Chunk size. Firstly, when working with CellChunkMap, it is assumed that MSLAB Chunk size is going to be bigger than what it is now. However, however bigger will the bound be there still can come a cell bigger than that. But still we assume super-large-cells to be a rare case and not the common case.

We suggest to resolve the issue by having special variable-size chunks allocated (as all other chunks) through the MemStoreChunkPool, but not being reused via the pool. Meaning those variable-size chunks are going to be allocated per super-large-cell (one or more) and released when cell (cells) is (are) no longer needed to be held in memory.

Required Refactoring (software engineering):

1. Take Chunk class out of HeapMemStoreLAB class. Refine MemStoreLAB interface and let everyone work with MemStoreLAB instead of HeapMemStoreLAB (related to MemStoreChunkPool, CellChunkMap, MemStoreLAB, ImmutableSegment, etc.)
2. Ensure MemStoreChunkPool is always not null with CellChunkMap being in use. Or probably split MemStoreChunkPool into MemStoreChunkPool and MemStoreChunkAllocator (where the last always exists when MSLAB is on).
3. Create three subclasses of ImmutableSegment according to the type

The issues raised in the sub-tasks of HBASE-16421:

1. Cell bigger than chunk
2. When a cell is upserted (append/increment op), we will not copy that cell to MSLAB.
    a. Anoop and Ram are suggesting to keep a boolean that whether a Segment contains a cell which is not copied to MSLAB and then intermix CellChunkMap and CellArrayMap
3. For scans we need to create instances of cells every time we recreate the cell from the index. Time consuming/GC issues
4. Whether to keep in the CellChunkMap ChunkID+offset+length (12bytes) or only ChunkID+offset (8bytes)

TODO PLAN:

1. Take the old CellChunkMap implementation and port it with needed augmentation, including needed refactoring. Dead code not connected to the CompactingMemStore.
2. Resolve the issue of Cells being aware of ChunkID.
3. Arrange connection to the CompactingMemStore (meaning flattening to the CellChunkMap). Still no support for super-large-cells.
4. Take care for the super-large-cells case, when Cells are bigger than Chunks (current workaround use only CellArrayMap in this case)
5. Improve testing of the CellChunkMap