Galore: Memory Manager

Goals

- Reuse all the objects in the galore. As all the action happens in the real time streaming in galore, expecting the java garbage collector doing the right thing is too much to ask for.
- Minimize locks while using the memory objects.
- simple accounting of memory objects
- decent memory usage reports, total picture and also per thread statistics.
- should do lot better than java garbage collector.

How we do it in Galore

Firstly, we divide the total memory into memory blocks. All the memory allocation or freeing should happen in terms of memory blocks as opposed to individual memory objects there by cutting the unnecessary allocation/freeing per object.
we should make sure that at any time only thread handles(either read/writes) a memory block. This would ensure no locking necessary while sharing the objects.
There would mainly be two type of memory blocks, one for byte buffers and another for the object pool. However, some variable inside the object may need a reference to a byte buffer, in which case the object pool memory block is linked to a byte buffer memory block.
All the freed memory blocks are maintained in a lock free stack. (implemented using java atomics). It is required to have separate lock free stacks for each object type.
For each thread we maintain per thread statistics in a bit map index. This helps in getting a report of per thread memory block usage at any point in time.
When a thread is killed or exited, all the memory blocks owned by the thread should be added to the free list.

Implementation notes

```
/**
 * Implementation Notes of Memory Manager.
 * THREAD_POOL, THREAD ID AS INDEX.
 * ____________________________
 * |thr |thr |
 * |id  |id  |
 * |0_|1__|_____________________
 * \    \
 * \       \
 * \          \
 * \              \
 * \                 \
 *  0 1 2 3 4 5 6 7 8
 * ___________________(MemoryIndexForThrd: per thread)
 * |_|_|_|_|_|_|_|_|_|_|_|_|        (per thread bit map)
 *  |   \  \
 *  \    \   \
 *   \     \    \
 *    \      \    \
 *     \      \     \
 *      \      \      \
 * m   \_______\_______ \___________________________________
 * B    |Memory |Memory |                      |
 * l    |Block  |Block  |         ............|
 * o    |Index:0|Index:1|         ............|
 * c    |_______|_______|______________________|_____________
 * k                      /     \
 * s                     /       \
 *            MemoryBlock       MemoryBlock
 *            ByteBuffer        Object Pool
 */
```

MemoryBlockByteBuffer

Memory given in a array of bytes, mainly useful to store variable size data like strings. The current used space is indicated by an offset variable.

MemoryBlockObjectPool

Objects are created in the memory block and will be reused when possible. This avoids objects creating and destroying after use, which makes the java garbage collector painfully slow.

MemoryBlockGroup

list of memory blocks. It is necessary to have a class for memory block group for this simple case; if any object in the MemoryBlockObjectPool contains variable string, the data is stored in a MemoryBlockByteBuffer. This demands that a MemoryBlockObjectPool should always be paired with a MemoryBlockByteBuffer.