Optimisation of Memory Allocators for the HLT (using AthenaMT)

Maruf Ali

Supervisor: Dr Stewart Martin-Haugh

Trigger Core Software Meeting, CERN

August 2, 2019

Science & Technology Facilities Council
Rutherford Appleton Laboratory

PROJECT

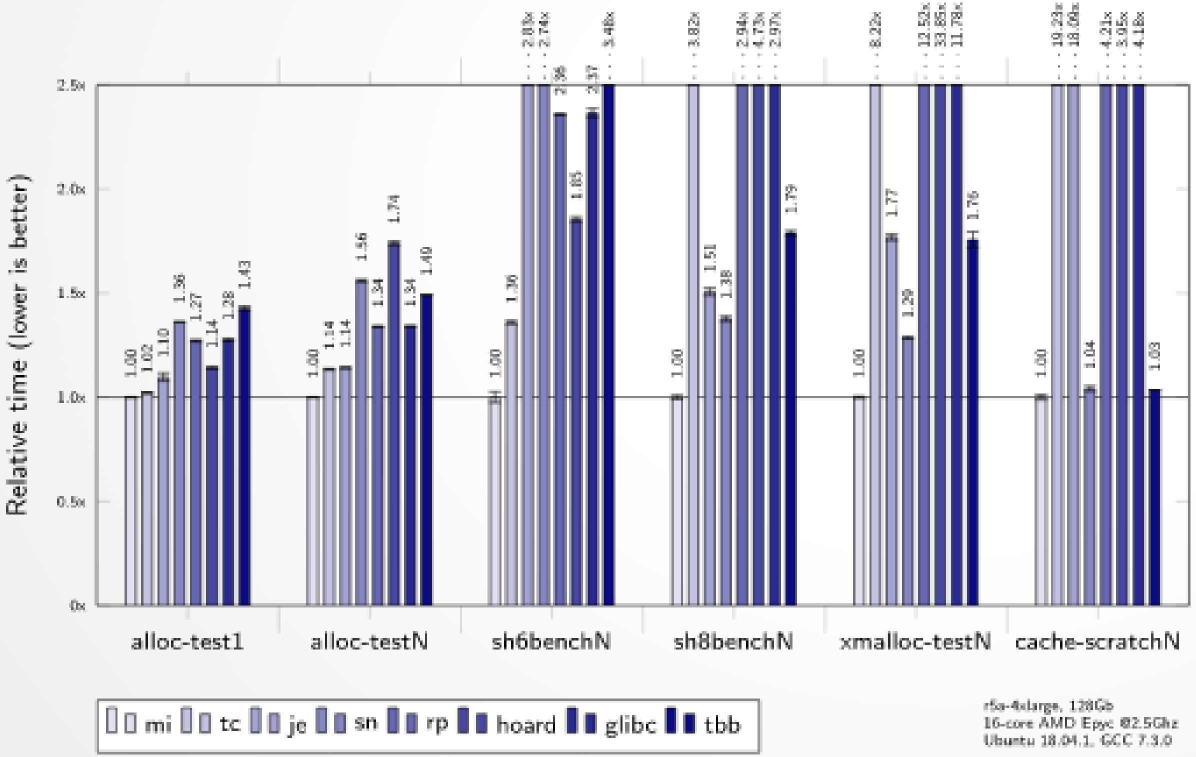
- ATLAS code allocates huge amounts of memory - using different memory allocators instead of glibc (stdc) can offer big (10,20%) performance improvements without changing any of our code.
- I am currently investigating the relative performance of 5 allocators on ATLAS HLT code.
- Single-Threaded Network → Multi-Threaded Network comparing processing speeds and memory usage.

MEMORY ALLOCATORS

- ‘jemalloc’ (Jason Evans) – Firefox & FreeBSD
- ‘mimalloc’ (Microsoft) – Just released (06/19)
- ‘tcmalloc’ (Thread-Caching) – Google Chrome
- ‘tbbmalloc’ (Intel Threading Building Blocks) – Designed to work with TBB
- ‘stdcmalloc’ (or ‘glibcmalloc’) – Linux (system default)

WHY?

- Preloads libraries to decrease CPU time



Histogram showing the times in different CPU tests for each type of memory allocator (from developers of mimalloc)

1. ‘mimalloc’
2. ‘jemalloc’
3. ‘tcmalloc’
4. ‘tbbmalloc’
5. ‘glibc’ (or stdc)

Decreasing performance

https://github.com/microsoft/mimalloc

https://www.microsoft.com/en-us/research/publication/mimalloc-free-list-sharding-in-action/

INSTRUCTIONS

I used the following commands to preload the libraries for each allocator with Athena

jemalloc:
‘ --preloadlib=/cvmfs/sft.cern.ch/lcg/releases/LCG_95/jemalloc/4.1.0/x86_64-slc6-gcc8-opt/lib/libjemalloc.so’

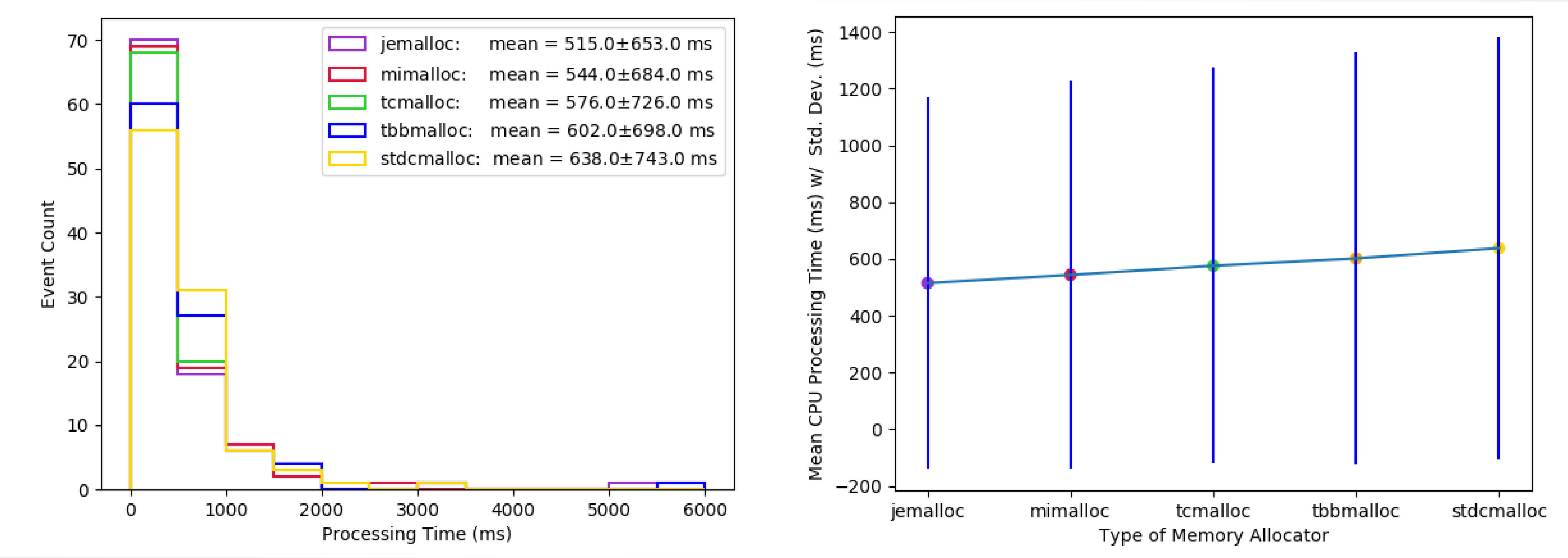
mimalloc (compiled from Github):
‘ --preloadlib=mimalloc/out/release/libmimalloc.so’

tcmalloc: Athena default
(‘/cvmfs/atlas-nightlies.cern.ch/repo/sw/master/sw/lcg/releases/LCG_95/gperftools/2.5/x86_64-slc6-gcc8-opt/lib/libtcmalloc_minimal.so’)

tbbmalloc:
‘ --preloadlib=/cvmfs/atlas-nightlies.cern.ch/repo/sw/master/sw/lcg/releases/LCG_95/tbb/2019_U1/x86_64-slc6-gcc8-opt/lib/libtbbmalloc.so’

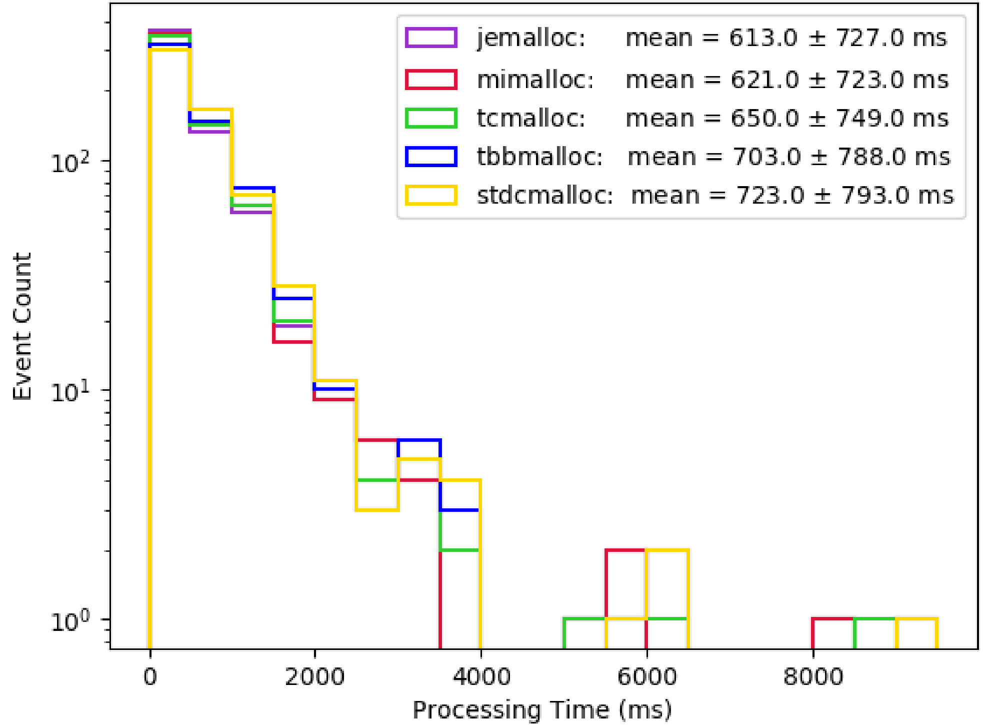
stdcmalloc: From OS

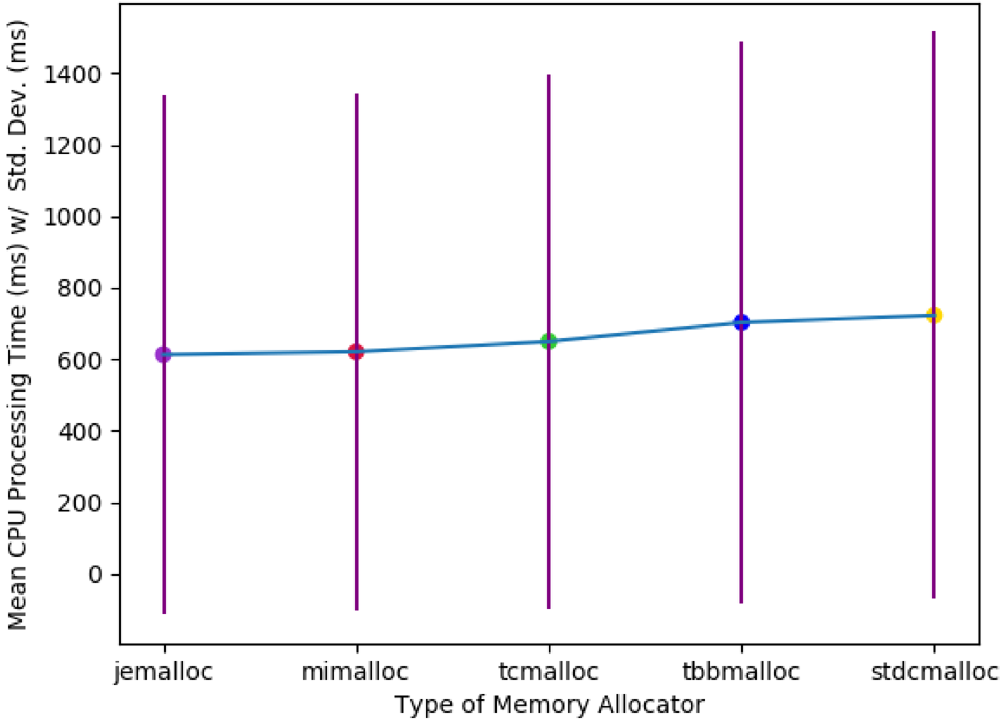
SINGLE THREADS (100 EVENTS)



‘asetup master,r2019-07-01T2127,Athena’
‘test_trigUpgr_full_menu_build.sh’ (with different preloads)

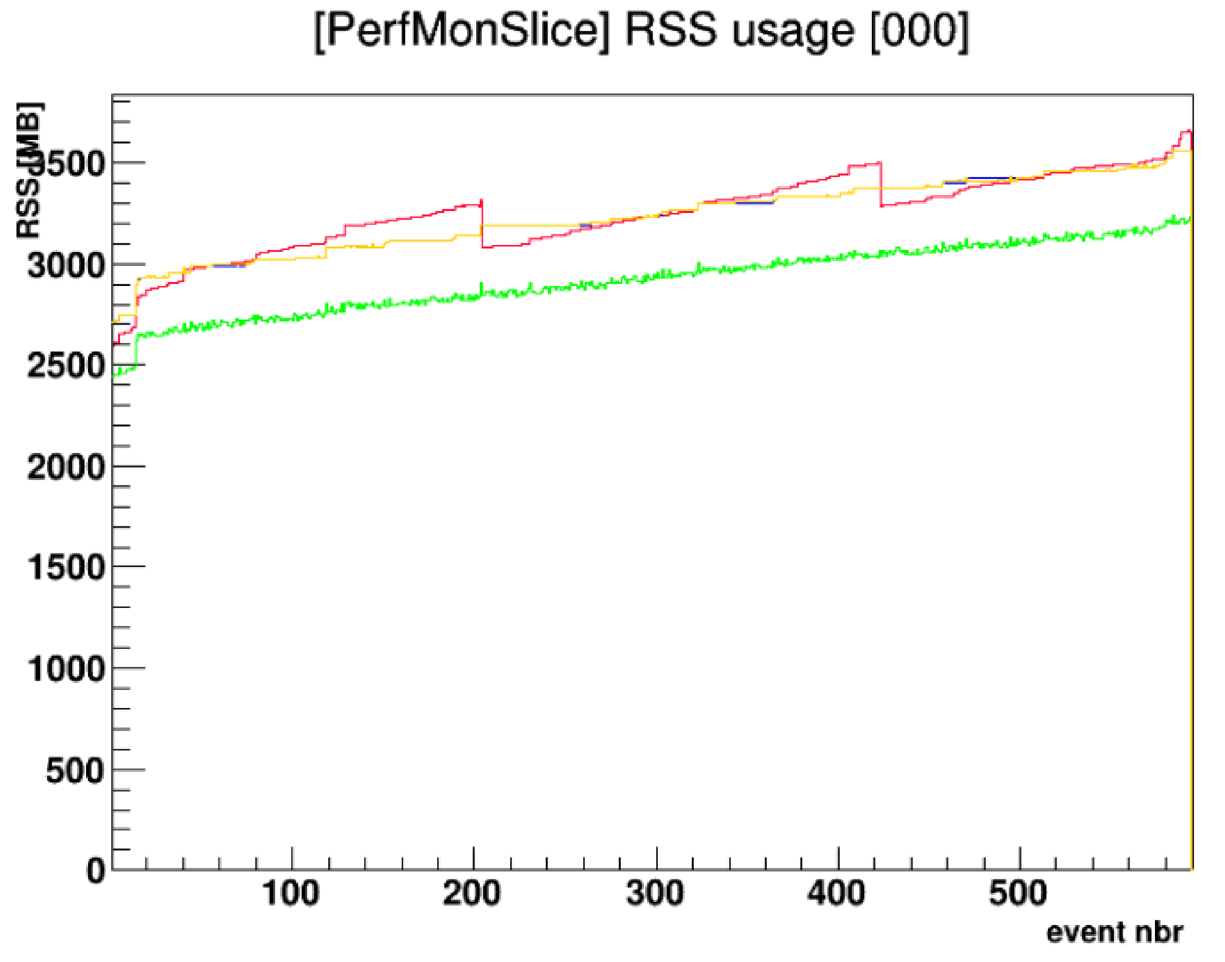
SINGLE THREADS (~600 EVENTS)





Best performance from jemalloc
Worst performance from stdcmalloc

MEMORY USAGE (ST)

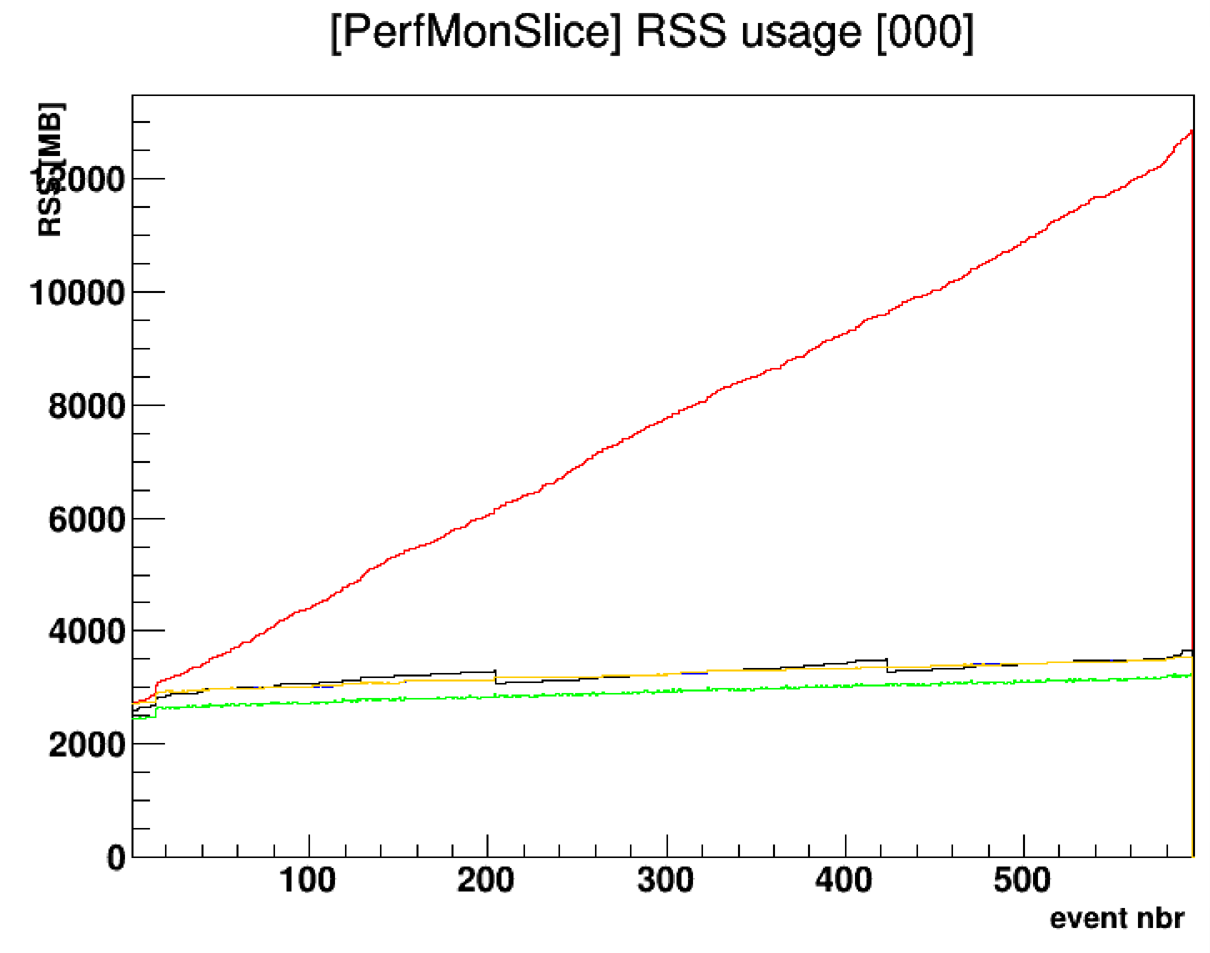


Taken from ROOT; to show memory usage for each allocator.
Key:
Red = jemalloc
Yellow = stdcmalloc (glibc)
Blue = tbbmalloc
Green = tcmalloc

Memory freed up for jemalloc – sawtooth-like behaviour

Best performance - tcmalloc

MEMORY USAGE (ST cont.)

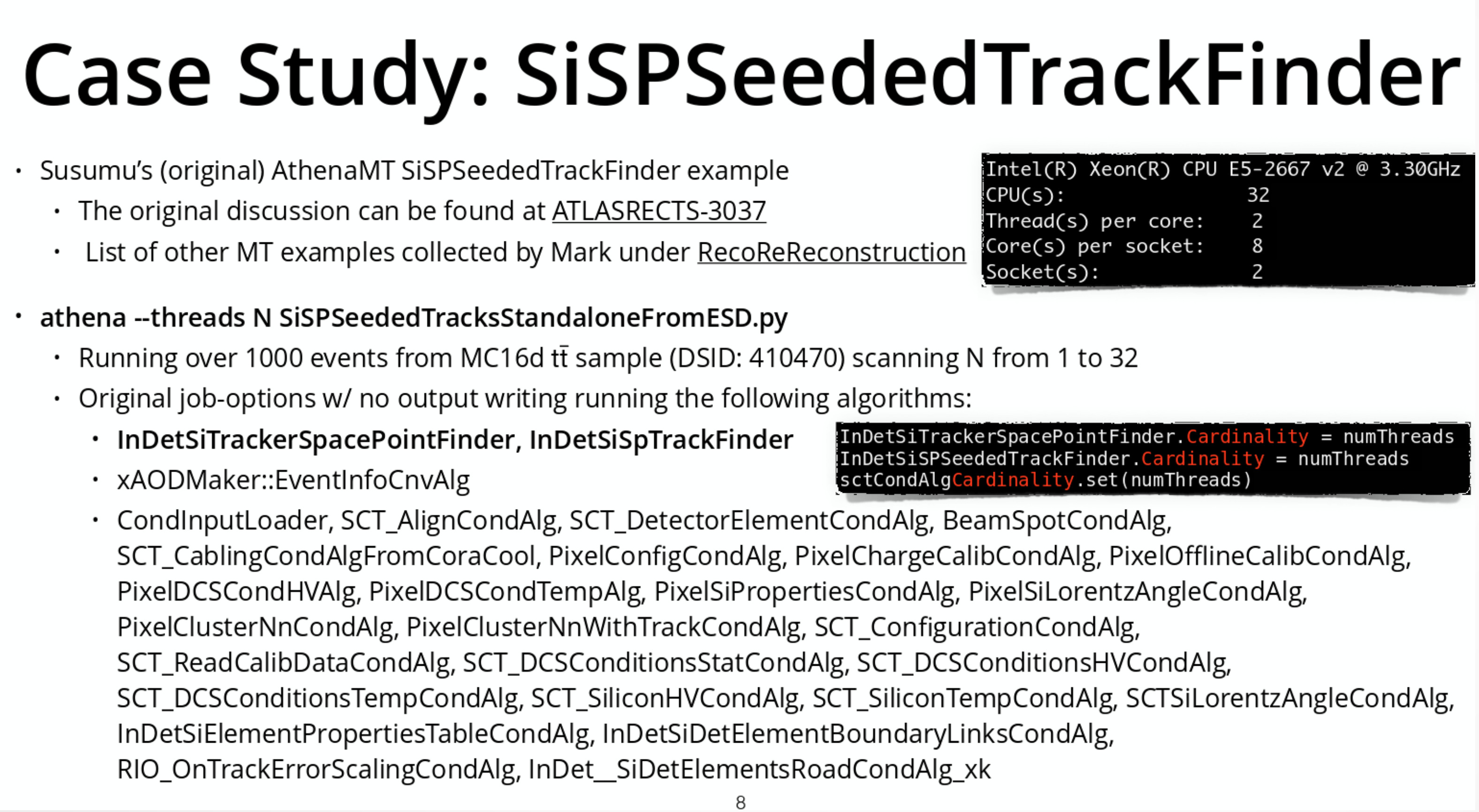


Taken from ROOT; to show memory usage for each allocator.

Key:
Red = mimalloc
Black = jemalloc
Yellow = stdcmalloc (glibc)
Blue = tbbmalloc
Green = tcmalloc

Significant memory leak for mimalloc (origin uncertain)

RECONSTUCTION SOFTWARE



Case Study: SiSPSeededTrackFinder

- Susumu's (original) AthenaMT SiSPSeededTrackFinder example
  - The original discussion can be found at ATLASRECTS-3037
  - List of other MT examples collected by Mark under RecoReReconstruction

```
Intel(R) Xeon(R) CPU E5-2667 v2 @ 3.30GHz
CPU(s):                32
Thread(s) per core:     2
Core(s) per socket:     8
Socket(s):              2
```

- **athena --threads N SiSPSeededTracksStandaloneFromESD.py**
  - Running over 1000 events from MC16d $t\bar{t}$ sample (DSID: 410470) scanning N from 1 to 32
  - Original job-options w/ no output writing running the following algorithms:
    - **InDetSiTrackerSpacePointFinder, InDetSiSpTrackFinder**
    - xAODMaker::EventInfoCnvAlg
    - CondInputLoader, SCT_AlignCondAlg, SCT_DetectorElementCondAlg, BeamSpotCondAlg, SCT_CablingCondAlgFromCoraCool, PixelConfigCondAlg, PixelChargeCalibCondAlg, PixelOfflineCalibCondAlg, PixelDCSCondHVAlg, PixelDCSCondTempAlg, PixelSiPropertiesCondAlg, PixelSiLorentzAngleCondAlg, PixelClusterNnCondAlg, PixelClusterNnWithTrackCondAlg, SCT_ConfigurationCondAlg, SCT_ReadCalibDataCondAlg, SCT_DCSConditionsStatCondAlg, SCT_DCSConditionsHVCondAlg, SCT_DCSConditionsTempCondAlg, SCT_SiliconHVCondAlg, SCT_SiliconTempCondAlg, SCTSiLorentzAngleCondAlg, InDetSiElementPropertiesTableCondAlg, InDetSiDetElementBoundaryLinksCondAlg, RIO_OnTrackErrorScalingCondAlg, InDet__SiDetElementsRoadCondAlg_xk

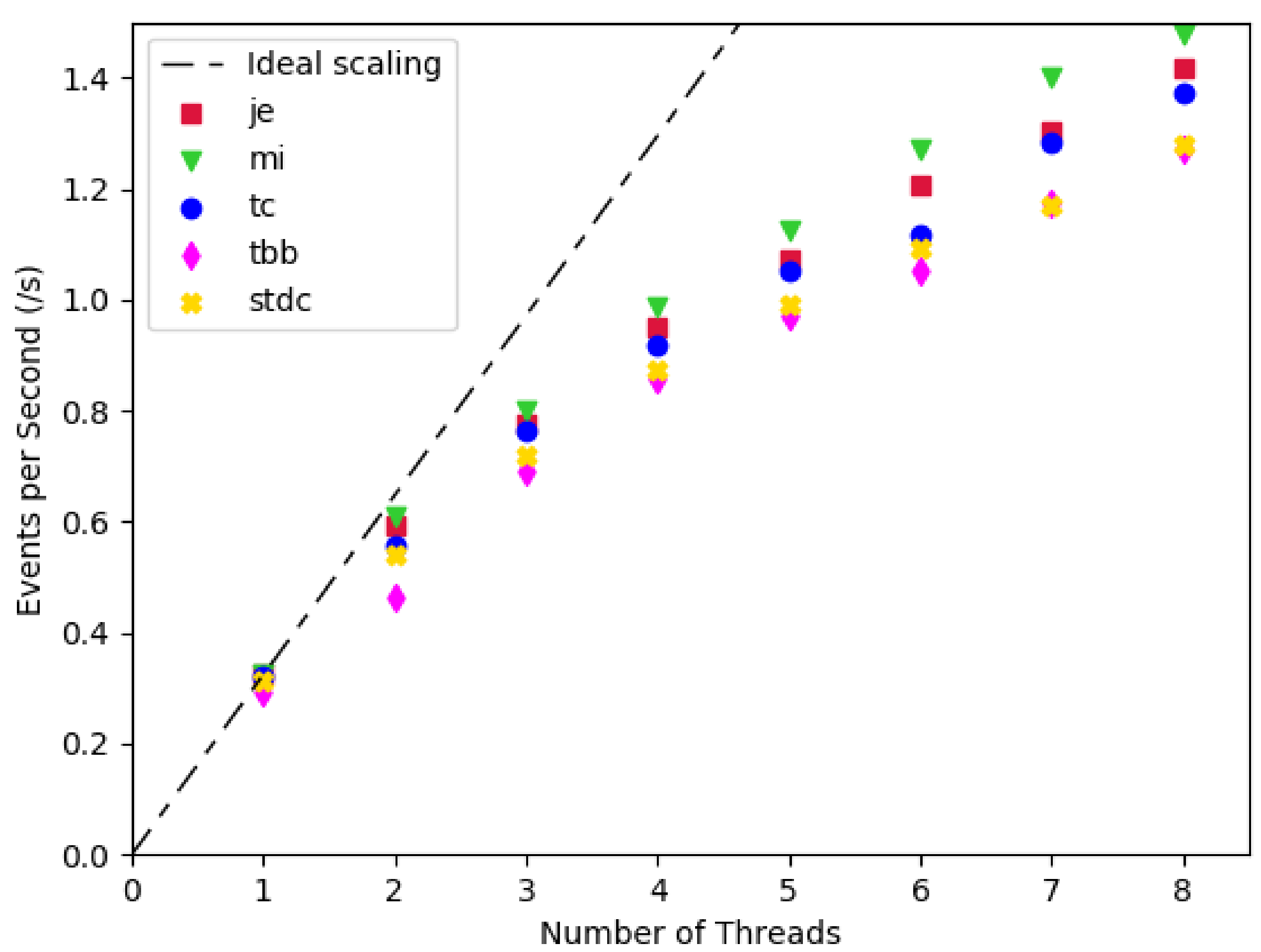
```
InDetSiTrackerSpacePointFinder.Cardinality = numThreads
InDetSiSPSeededTrackFinder.Cardinality = numThreads
sctCondAlgCardinality.set(numThreads)
```

8

No MT trigger code

So running offline reconstruction code (from Serhan's recipe)

THROUGHPUT (SiSPSeededTracker)



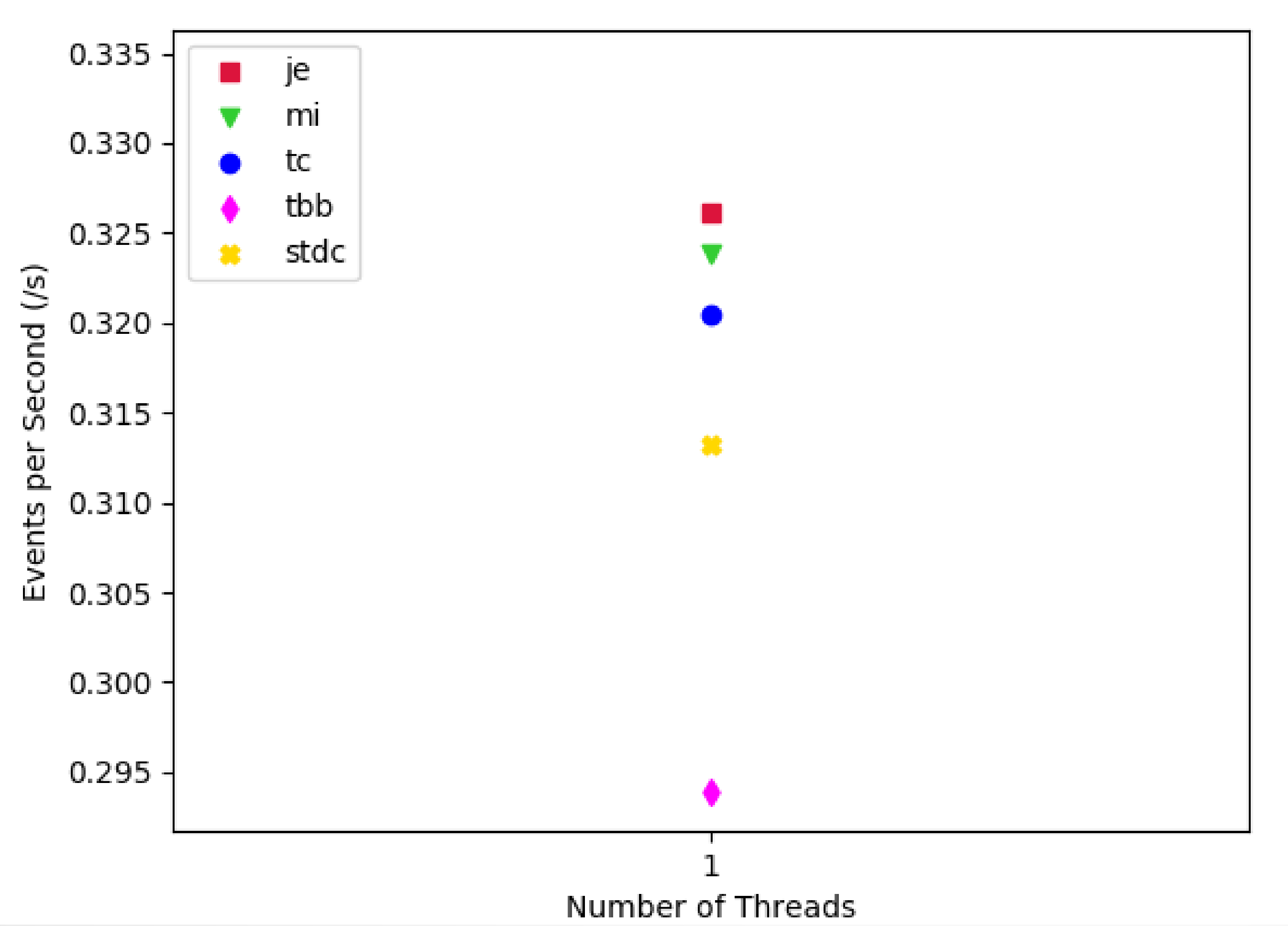
Comparison of all 5 memory allocator performance.

For 1 thread, jemalloc processes more events per second than mimalloc.

However, for more than 1 thread it is dominated by mimalloc.

We can deduce that mimalloc is the best performer with increasing number of threads.

THROUGHPUT (SiSP cont.)

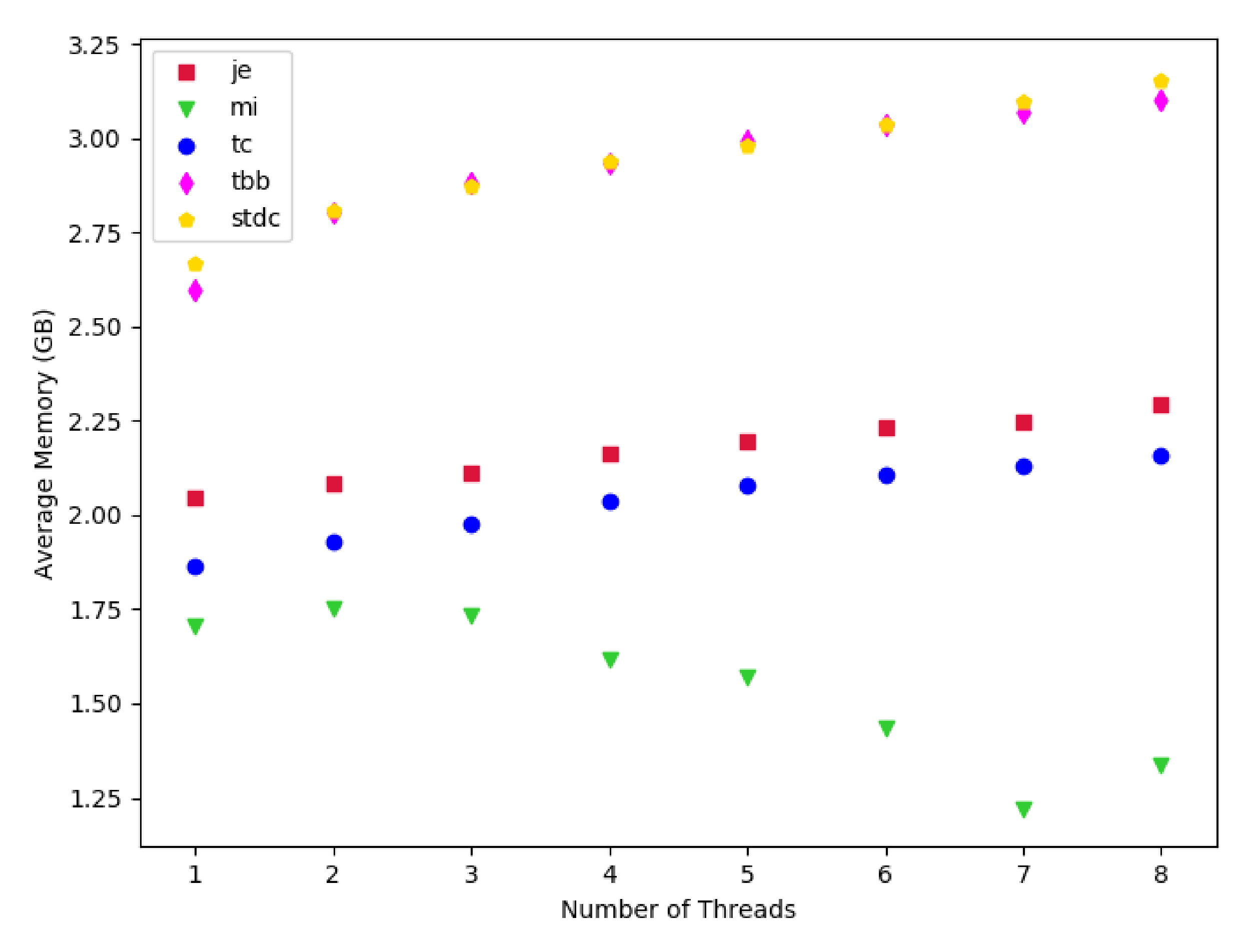


Throughput for 1 thread using a MT.

Notice that jemalloc has a greater value than mimalloc.

Also notice that the top 3 allocators are bunched up, similarly to the ST.

MEMORY USAGE (SiSP)



All but ‘mimalloc’ has the expected relationship when comparing the mean rss.

Notice that mimalloc rises with 2 threads then decreases until 7 threads where it then rises again.

Uncertain of mimalloc behaviour, perhaps it is due to a bug.

SUMMARY

- We have preloaded libraries for 5 different types of memory allocators to see if we can increase efficiency without actually modifying the contents of the ATHENA code.

From Github:
1. ‘mimalloc’
2. ‘jemalloc’
3. ‘tcmalloc’
4. ‘tbbmalloc’
5. ‘stdcmalloc’

From ST:
1. ‘jemalloc’
2. ‘mimalloc’
3. ‘tcmalloc’
4. ‘tbbmalloc’
5. ‘stdcmalloc’

From MT:
1. ‘mimalloc’
2. ‘jemalloc’
3. ‘tcmalloc’ ← Currently using
4. ‘stdcmalloc’
5. ‘tbbmalloc’

To conclude, both mimalloc & jemalloc better perform than the default tcmalloc to a certain extent (depending on the balance between throughput and memory usage)