**Slide 1**

Hi, I’m Rosie I am a summer student working with Stewart on the high level trigger.

**Slide 2**

I’ve been looking at whether the use of a different memory allocator and multithreading could improve the throughput. I focused on the throughput but also considered the memory usage by looking at the pss.

**Slide 3**

I ran the high level trigger on a sample of 100 events using the test trig program on the CERN test bed computers. I edited it to vary the number of forks and threads, up to 8 events and to preload different memory allocators. The overall time

**Slide 4**

Memory allocators assign memory to processes that request it and free it up when it is no longer in use. They are more efficient at this than if the code just accessed memory when it needed it. Each uses different algorithms to make these decisions.

I looked at 4 different allocators,

stdcmalloc is the one currently used it is the default on linux.

Tcmalloc is the default for Athena it was made by google, it focuses on threading with TC standing for thread caching.

Jemalloc is used in firefox. Stands for Jason Evans, which does not give much information, it was designed for multi threading and multiprocessing and tries to minimise overall memory usage

Mimalloc was developed by Microsoft, when developing it they conducted a study on cpu time that stated it would perform better than the other memory allocators.

**Slide 5**

I also looked at using multiple threads and forks, trying to find the optimal combination. This is an example of the phase space of threads and forks, it implied the most effective may be a combination of the 2.

I set the number of events processed by each thread to one so the number of events processed at the same time is forks multiplied by threads.

**Slide 6 + 7**

This is what I found for the throughput of the 4 allocators. They all increased in throughput as the number of events increased. I found the best combination was 2 forks and 4 threads for all of the allocators.

**Slide 8+9**

Now comparing the allocators for each number of forks, for all of them jemalloc and tcmalloc performed the best. They are both better than stdc, but there’s not much difference between them so looked at memory usage to see if one was more beneficial.

**Slide 10**

This is the peak pss used during each of these runs, jemalloc and tcmalloc use significantly less than stdcmalloc, with jemalloc using the least overall.

**Slide 11**

The black scaling line is equivalent to using N computers, all runs used significantly less than this so multithreading and multiforking are efficient in terms of memory usage.

**Slide 12**

Overall when looking at the memory allocators there is some evidence of the benefit of using something other than stdcmalloc, at the moment jemalloc seems to be performing best in terms of throughput and memory usage. I also found use of multiple threads and forks improved the throughput and didn’t use anywhere near as much memory as running it on N computers.