IOWA STATE UNIVERSITY
College of Liberal Arts and Sciences

Graduate and Professional Students' Research Conference
April 12, 2017

Nathan T. Weeks[1,2], Glenn R. Luecke[2]

1. Department of Computer Science
2. Department of Mathematics

Optimization of SAMtools Sorting Using OpenMP Tasks

Introduction

SAMtools is a widely-used bioinformatics application for post-processing high-throughput sequence alignment data. Such sequence alignment data are commonly sorted to make downstream analysis more efficient.

Problem

Sorting can be a bottleneck in genomics workflows: high-throughput sequence alignment files in the de facto standard Binary Alignment/Map (BAM) format can be many gigabytes in size, and may need to be decompressed before sorting and compressed afterwards.

Objectives

To analyze and improve the performance of SAMtools for the purpose of sorting large BAM files.

Optimizations

Input / Decode

- ➢ **Problem:** BGZF blocks are read and decoded sequentially.
- ➢ **Solution:** Concurrently decode each BGZF block in an OpenMP task.
- ➢ **Result:** Over 4X speedup.

Sort

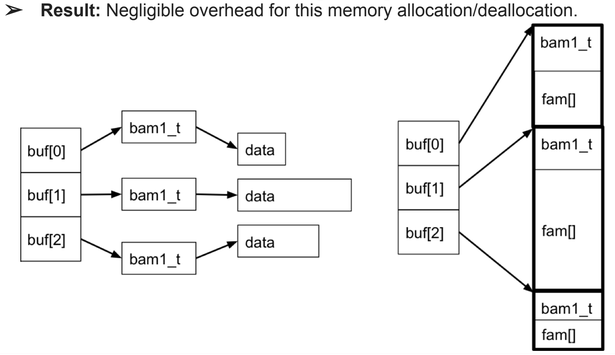
- ➢ **Problem:** The internal sort is not parallelized (only the external sort is).
- ➢ **Solution:** Generate a sort task every 2^{20} BAM records. Merge sorted sublists before output.
- ➢ **Result:** Sorting is parallelized and overlapped with input/decode, utilizing otherwise-idle processor cores.

Encode

- ➢ **Problem:** Multi-threaded compression didn't fully utilize processor cores (see Fig. 2, "before optimizations").
- ➢ **Solution:** Ticket-lock-like algorithm implemented using OpenMP tasks and circular buffers reduces memory copies & increases processor utilization.
- ➢ **Result:** Encoding run time cut by ~⅓.

Memory allocation / deallocation

- ➢ **Problem:** > 10% run time spent allocating 2 buffers for each of the ~207 million BAM records. ~13.5 % of run time spent deallocating the memory at the end of execution.
- ➢ **Solution:** Allocate/deallocate a single contiguous-memory buffer.
- ➢ **Result:** Negligible overhead for this memory allocation/deallocation.



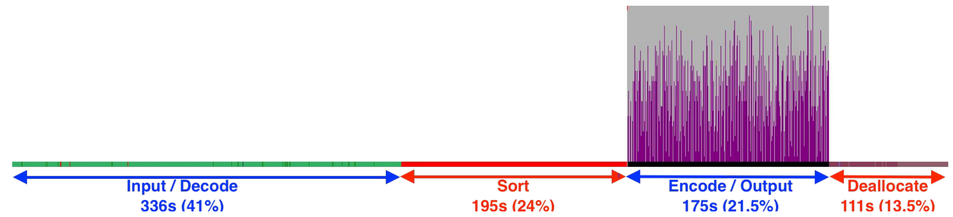
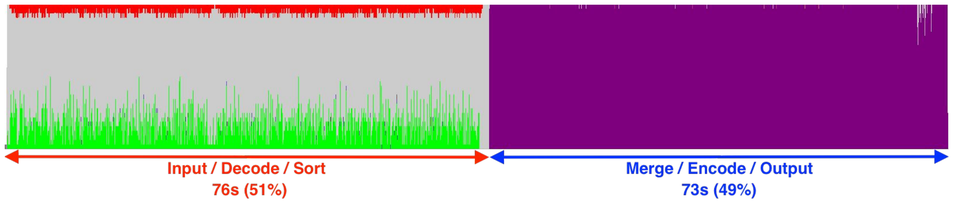


Figure 2. (Above: before optimzations; Below: after optimizations) HPCToolkit performance summary for the internal sort using 32 threads. The color represents the procedure a thread is executing: *green* – read+decode (primarily zlib `inflate()`), *red* – sort, *purple* – encode (primarily zlib `deflate()`), *gray* – thread waiting, white – no thread exists.



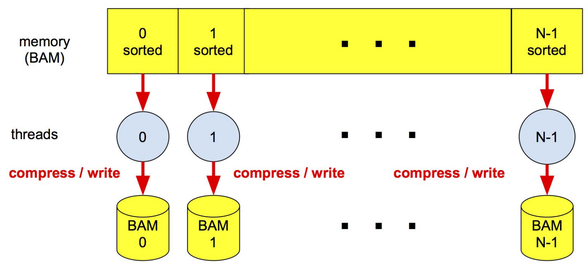
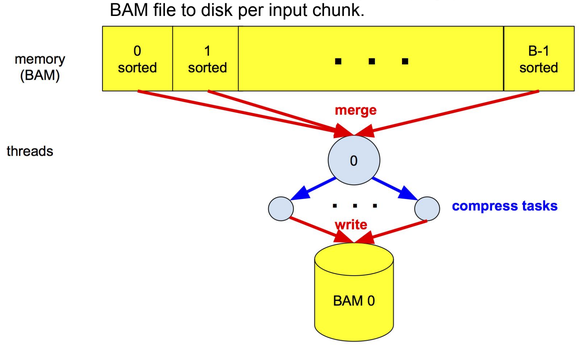


Figure 3. (Left) *(External sort)* If the input chunk fills memory, SAMtools 1.3.1 partitions memory into *N* partitions (where *N* is the number of threads). Each thread sorts its partition and writes it to secondary storage. After all input chunks have been sorted, the intermediate files are merged to produce the final sorted output BAM file.

Figure 1. (Far left) SAMtools 1.3.1 uses two dynamically-allocated buffers to store each BAM record, one for fixed-length fields (`bam1_t` struct) and one for variable-length (`data`). The array of pointers (`buf[]`) to these BAM records is sorted during the sort phase.

(Left) SAMtools OpenMP stores both fixed- and variable-length BAM record data in a single contiguous memory region, facilitated by the use of a flexible array member (fam[]) for the variable-length data.

Figure 4. *(Below)* SAMtools OpenMP merges in-memory sorted BAM records, writing only one (large) intermediate BAM file to disk per input chunk.



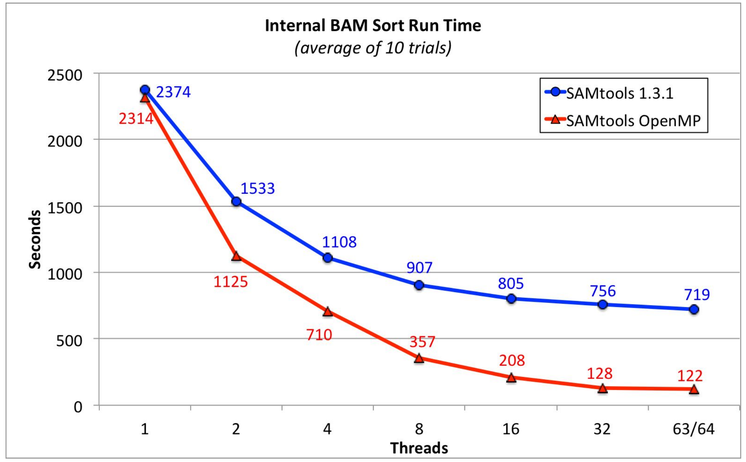


Figure 5. Benchmark results for an internal (in-memory) sort of 24.6 GiB BAM file containing alignments for 207 million 100 bp Illumina reads). Compute nodes had 128 GiB of memory and 32 processor cores; 63 threads (SAMtools OpenMP) and 64 threads (SAMtools 1.3.1) results use *Hyper-Threading* (2 threads per processor core). Input was staged to the NERSC Cori *Burst Buffer*, which provides fast I/O via SSDs. The Cray IOBUF library buffered reads from the Burst Buffer, avoiding a severe performance regression that would otherwise result from SAMtools small (32K) read transfer size.

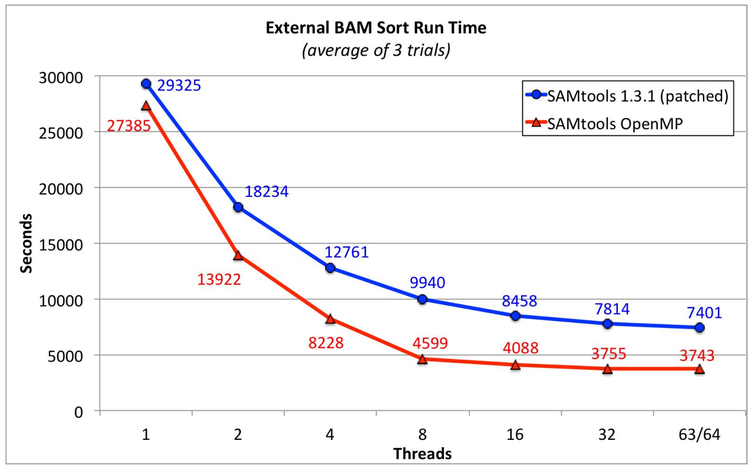


Figure 6. Benchmark results for an external (out-of-core) sort of a 271.4 GiB BAM file containing alignments for ~570 million 250 bp Illumina reads. The Burst Buffer stored input, output, and intermediate files. SAMtools 1.3.1 was patched to allow buffered reads from temporary files, but avoiding buffered writes, as the Cray IOBUF library is not thread safe.

Conclusions

The presented optimizations resulted in up to a 5.9X speedup versus SAMtools 1.3.1 for an internal (in-memory) sort of a 24.6 GiB BAM file, and up to a 1.98X speedup for an external (out-of-core) sort of a much larger BAM file (271.4 GiB) that didn't fit into memory.

Acknowledgements This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.