

Pipelines and Workflows

Adapting to HPC

Partners



Funding



Reusing this material



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.

Outline

- Example: bioinformatics pipelines and HPC
 - What's the problem?
- MapReduce programming model
- Hadoop – a MapReduce engine
- Halvade: distributing pipelines using Hadoop
- Workflow / Pipeline Languages, Tools & Frameworks
- Containers

Bioinformatics pipelines and HPC

what's the problem?

- Many bioinformatics tools parallelised using threading only
 - Best suited to shared-memory machines with a large amount of memory and modest numbers of cores (compared to HPC)
- On distributed-memory systems, restricted to single node
 - Indications are many multithreaded tools used in pipelines don't scale well to typical full number of single-node cores (low parallel efficiency)
- Few tools use MPI
 - Not the only way to scale, but an important / powerful one that would give instant ability to run on HPC machines

Bioinformatics pipelines and HPC

what's the problem?

Example:

```
user@machine:~> bwa mem -t $NSLOTS -M $BWA_INDEX_REF -R
"@RG\tID:$PU\tPL:illumina\tPU:$PU\tSM:$SAMPLE" $READS1
$READS2 | samblaster --splitterFile >(samtools view -hSu /dev/stdin |
samtools sort -@ $NSLOTS /dev/stdin > $SAMPLE.sr.bam) --
discordantFile >(samtools view -hSu /dev/stdin | samtools sort -@
$NSLOTS /dev/stdin > $SAMPLE.disc.bam) | samtools view -hSu
/dev/stdin | samtools sort -@ $NSLOTS /dev/stdin >
$SAMPLE.raw.bam
```

Bioinformatics pipelines and HPC

what's the problem?

Example simplified:

```
A | B --arg1 >( C | D > file1) --arg2 >( E | F > file2) | G | H > file3
```

- 8 executables (A – H)
 - 4 multithreaded
 - potentially using different threading standards (pthreads, OpenMP, ...)
- Efficiency of parallel executables unknown
 - No idea of optimal number of threads to assign to each (assuming we can)
- Linux pipes don't allow straightforward control over parallel execution
 - Mostly relying on operating system to do the right thing
 - Data flow through pipes and pipe buffers adds additional complications

Bioinformatics pipelines and HPC

what's the problem?

- Can take days to run
 - Difficult to determine which stages are taking the most time
 - Can't use many standard performance profiling tools
 - Difficult to understand, let alone improve runtime
 - Need to carefully tease apart performance of each step
 - Still leaves questions regarding dataflow between piped commands
- Linux piping *may* be very efficient in some circumstances
 - but don't have a good handle on this
- Risk is this approach became common because it was available, convenient (interactive), and sufficiently fast for smaller data sets
 - Possibly running into problems with more and more data
 - Issues surrounding robustness, checkpointing (can't run > 48 hours on ARCHER)
 - Need a solution engineered for purpose to tackle larger scale in a controlled manner

MapReduce

- Approach for processing big data sets in parallel on a distributed-memory cluster
- Similar to Task Farm, but with three types of workers:
 - Mappers (computers)
 - Shufflers (data redistributors) – can be done by master alone
 - Reducers (recombine results of mappers' computations)

Hadoop & Spark

- Hadoop is an implementation of the MapReduce pattern
- Engine to manage execution of many data processing tasks
- Includes a distributed filesystem
- Used as an engine to automatically and efficiently distribute data to where it's needed to perform data processing tasks and gather results
- Resilient (fault tolerant)
- Spark uses Hadoop's MapReduce and distributed storage, adapted to process streaming realtime data.

Halvade

- Example framework to run sequencing pipelines in parallel
 - based on Hadoop
 - multicore **AND** multi-node
 - alignment (BWA) → data preparation (Picard) → variant calling (GATK)
- Developers analysed
 - single-node threaded efficiency of each tool
 - used to allocate optimum number of threads for each tool in Halvade
 - Optimal task size (experimentally) - found granularity sweet spot
- Good parallel efficiency (~90%) running whole human genome on distributed-memory cluster (up to 16 nodes - 512 cores, 60GB memory per node)

Barriers

- Parallel pipeline performance of Halvade and similar that rely on Hadoop (or Spark) seem promising
- Barrier to deployment on especially the largest HPC systems is conflict with existing file systems, resource management, etc.
 - Potential disruption of environment finely tuned for performance of tightly-coupled traditional HPC codes?
 - Not enough demand or incentive?
 - Gap between (potential) users and service providers?
 - HPC hardware and service mainly funded by non-bioinformaticians?
 - ... ?
- Smaller HPC machines more flexible
 - EPCC HPC service Cirrus used for genomics analyses, has Hadoop, Spark

Workflow / Pipeline Languages, Tools & Frameworks

- Currently observe a diversity of workflows and pipeline frameworks
 - Varying approaches to parallelism
 - Future prospects of any given framework uncertain
 - Hampers adoption / traction, decreases incentive for continued development
- Similarities with earlier decades of parallel computing:
 - Early diversity of message-passing libraries
 - Later adoption of a common standard
 - Allowed applications to be written in parallel once and run everywhere
 - Single standard for parallelism enabled targeting of efforts to improve performance from software and hardware sides
 - Great success - led to portable applications that can run on many different and constantly newer HPC machines

Workflow / Pipeline Languages, Tools & Frameworks

- WL and pipeline frameworks and their users might benefit from similar
 - Single common framework could potentially improve uptake, deployment and usage on HPC machines, similar to historical emergence of MPI
- Common Workflow Language (CWL)
 - Provides valuable standardisation
- Software development efforts could focus on parallelisation of CWL execution layer(s)
 - Would allow separation of concerns, with relevant expertise directed at each level of software and hardware

<https://bioexcel.eu/software/workflows/>

Containers

- Lightweight alternative to virtual machine (VM)
- Provide a way to package up a particular (version of) an application and everything it requires
 - Described in terms of a simple recipe
- Example: Docker (Singularity for HPC)
- Provides a way to introduce more flexibility into usage of clusters / HPC machines, allowing easier transition from laptop/desktop/server

Summary

- Some bioinformatics pipelines / workflows are problematic from the point of view of HPC
 - Understanding and improving performance is challenging
- Frameworks for massively parallel data-centric computing appear promising
- There are some barriers to uptake / deployment on HPC systems
 - Situation is made more complicated by diversity of approaches without one unambiguous favourite