INSTRUMENTING HUMAN VARIANT CALLING WORKFLOW

Allocation: Illinois/0.60 Mnh
PI: L. S. Mainzer[1,2,3]
Collaborators: Arjun Athreya[3], Subho Banerjee[3], Ravishankar K. Iyer[3], Victor C. Jongeneel[1,2,3], Volodymyr Kindratenko[1,3], Zachary Stephens[3]

[1]National Center for Supercomputing Applications
[2]Institute for Genomic Biology, University of Illinois at Urbana–Champaign
[3]University of Illinois at Urbana–Champaign

EXECUTIVE SUMMARY:

If whole-genome sequencing and analysis become part of the standard of care within the next few years, human genetic variant calling will need to be performed on hundreds of individuals on any given day. For example, genotyping every baby born in Illinois would require analysis of ~500 genomes per day. At this scale, the standard workflow widely accepted in the research and medical community will use thousands of nodes at a time and have I/O bottlenecks that could affect performance even on a world class petascale system like Blue Waters. We identified and documented the bottlenecks associated with the large number of small files created by the workflow, saturated I/O bandwidth for part of the workflow, and potential for unbalanced data load on the file system. Now we are designing and testing tools and methods to overcome these problems.

INTRODUCTION

Human variant calling is becoming the computational tool of choice to help diagnose intractable diseases and cancers. This bioinformatics tool searches for differences between a patient's genome and that of a reference, or average, of a human population. It is likely that in the not-too-distant future, every state and major metropolitan area would produce enough human genetic sequencing data to require their own HPC facility to analyze those data. What kinds of challenges would such a computational facility face, and what preparations would be necessary to ensure good performance with sustained throughput?

METHODS & RESULTS

We set up the standard GATK-based workflow [1] and tested a number of alternative tools at every step in the computation in an effort to shorten the computation wall time per genome. In addition, we benchmarked the CPU, RAM, and I/O system across the workflow using Perfsuite [2], Cray Profiler [3], OVIS [4], Valgrind [5], and some of the software we developed. This work generated close collaborations with Cray, the Blue Waters support team, and groups on campus, and identified several performance issues, most of which have to do with data I/O.

Variant calling is a big-data workflow when used in a sustained fashion on hundreds of samples per day. The required disk space is at the petascale on a daily basis. Even if the intermediary files are removed after the workflow is complete, they still need to be created, stored, and managed for the duration, and also in case one of the steps fails. The vast majority of these data are generated in the form of small files, which creates several concerns.

Creating, reading, and writing large numbers of small files can create an I/O bottleneck if the files are not placed uniformly across the file system. We found that the Lustre file system on Blue Waters places files relatively evenly across disks, according to our measurements. We continue to explore situations that could create unbalanced file placement and introduce tools into the workflow to prevent that from happening.

Handling large numbers of files can strain the metadata servers and result in uneven node performance, which affects both variant calling itself and other users on the system. We are investigating methods to do bookkeeping for deep collections of directories to lessen the load on the metadata servers and make data handling faster.

This workflow may benefit from storage pools. Usually, HPC jobs are MPI based and require fast inter-node communication, which means they benefit from being placed onto adjacent nodes. However, the variant calling workflow consists of jobs that run independent computational tasks on their own nodes. The I/O happens between the compute nodes and storage disk, not between compute nodes themselves. Thus, we are testing the use of a wide, sparse distribution of nodes assigned to the same job across the system.

The sheer size of the data footprint can create an infrastructure bottleneck. We are automating the output data archiving and ingestion of input streams, and offloading the output streams onto the target destinations to emulate the real-world case of genome data being streamed from sequencing facilities back to the medical centers.

Re-sorting aligned files by read position or by location along the genome is a common task that has to be done in a number of places along the variant calling workflow. The fastest algorithm, to our knowledge, is Novosort [5], which involves two phases. The first phase sorts as much data as possible in memory and then writes segments of sorted records to temporary disk files. The second phase merges the sorted fragments to produce the final sorted file. This algorithm is so efficient that it saturated the peak node injection bandwidth on Blue Waters (which is 9.6 GB/sec). This could create bottlenecks at scale by saturating the network routers on the system and interfering with functioning of other users. We are now measuring the extent of this potential problem and investigating a workaround by using data disk pools and/or a staggered version of the workflow to keep the sort–merge processes from overlapping.

WHY BLUE WATERS?

This project is timely because hundreds of sequencing facilities have already opened across the nation and the deluge of human genomics data is a reality today. The HPC facilities of tomorrow need to be prepared to handle the incoming load of genomic data. We hope that our project will inform the hardware and software designers about the requirements imposed by genomics on the next generation of computing facilities. The great advantage of using Blue Waters for this work is that it combines both a state-of-the-art data system and a large number of nodes to even make these experiments possible. The Blue Waters support staff have been instrumental in helping us figure out and eliminate issues with computational performance.