Optimizing Massive Data Access for Large Scale Population Genomics Analysis Using HDF5

Junrong Yang[1#], Peihao Liu[2#], Guixin Guo[3], Hanquan Liang[3], Wei Yan[3], Bingqiang Wang[3*], Yunfei Du[3] , Shoubin Dong[1]

*#Equal contributor *Corresponding author*

1.South China University of Technology

2.National University of Defense Technology

3.National Supercomputer Center in Guangzhou, Sun Yat-sen University

Abstract

More and more DNA sequencing data are generated, which enables population scale modeling for both scientific and clinical purposes. The traditional plain organization and layout of these data don't fit well with large scale analysis. Genotype imputation needs to analyze the same genome region of all individuals, thus small partial data of a large amount of files are used. Such kind of data access brings significant pressure to the parallel file system. To tackle this, HDF5 file format is employed as kind of container for these raw data files. Naturally one single HDF5 file corresponds to a human chromosome, inside the HDF5 file two layouts are proposed and tested. The first one is one-dimensional, data distributed as different individuals/samples. The second one is two-dimensional, data distributed along both fixed size regions and different individuals/samples. Our experiment shows that both layouts gain significant improvement, 3.4x speedup is observed. And two-dimensional layout performs even better because the feasibility to locate a certain region. It is clear that our work solves the metadata congestion, thus improves in data access performance.

Background

Over the last decade, DNA sequencing cost is reduced by orders of magnitudes, more scientific researches in life sciences and clinical diagnosis are relying heavily on sequencing technology nowadays. Population genetics studies analyze a large group of samples together to decipher genomes with unprecedented resolution. Supercomputer is the right choice for such extreme scale analysis up to millions of individuals. Current population analysis, such as variant identification and imputation analysis, first perform pre-analysis on a subset for each sample, then pooled the results together for a secondary analysis.

However, population genomics analysis is data intensive computing featuring high throughput. There is a gap between current population analysis pipelines (software) and supercomputer systems (hardware) in terms of compute to data access ratio.

Problem

Data organization

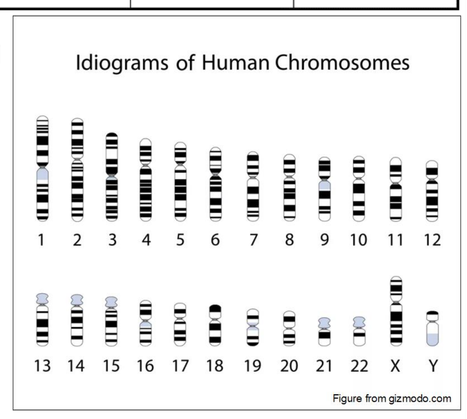
- One or several files per sample, storing data corresponding to chromosome 1 to Y, taking human genome as example.

File example stores all records of a sample

Various file formats (SAM/BAM/CRAM)				
Chr1	Chr2	…	ChrX	ChrY

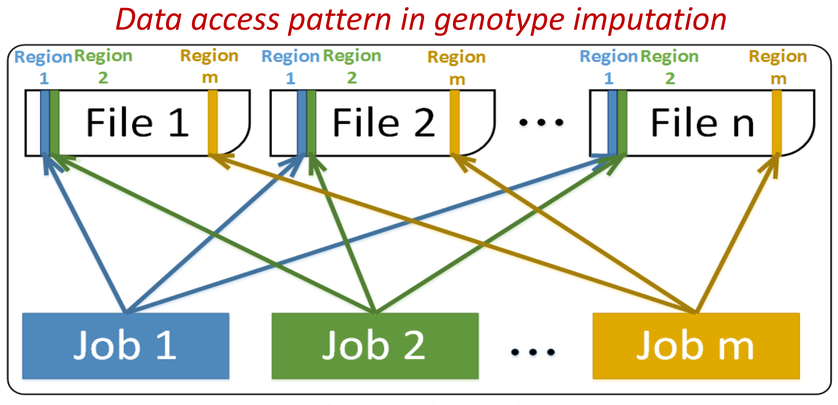
Access pattern

- Open and seek operations over all samples' relevant files per compute job
- Only a small region of data for an individual is loaded per compute job



In our genotype imputation case, every 5 million continuous genetic loci correspond to a region.

Data access pattern in genotype imputation



- Assuming ***n*** samples and ***m*** jobs
- ➢ ***O(n)*** file accesses per job (open/close, seek and read)
- ➢ ***O(mn)*** file accesses total for ***m*** jobs
- In our real case, typically hundreds of jobs (*m*=100) and hundreds of thousands of samples (*n*=100,000)
- ➢ Massive file operations per job (orders of 100,000)
- ➢ A burst of metadata requests (orders of 10,000,000 in a short time)

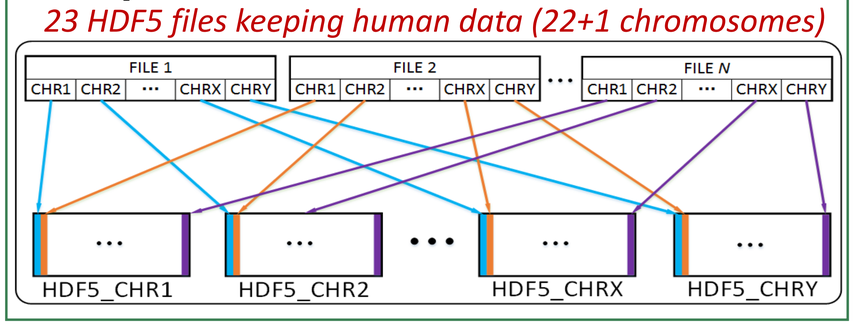
Proposed Solution

Key ideas

- Using HDF5 as container to reduce total number of files, and supporting parallel access
- Fully exploit built-in hierarchical data storage infrastructure: blocking/indexing and compression
- Carefully designed data layout schema to minimize overhead and enable efficient access

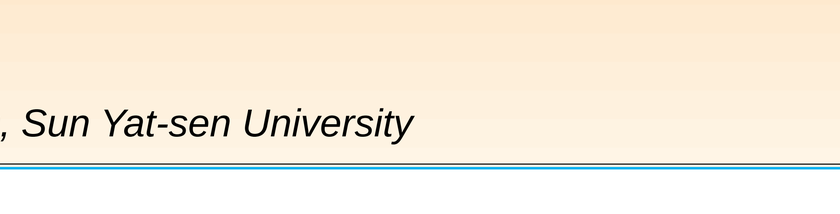
Improved access

23 HDF5 files keeping human data (22+1 chromosomes)



- Tier-0 partitioning of data is based on chromosome. Naturally a single HDF5 file keeps all data belonging to same chromosome of all samples, totally 23 HDF5 files for human data.

Improved access pattern with HDF5 as container



- ➢ ***O(1)*** file accesses per job
- ➢ ***O(m)*** file accesses total for ***m*** jobs

Tier-0 partitioning reduces unnecessary metadata access and pressure on the metadata server. Proper layout of data can further reduce internal access complexity inside HDF5 infrastructure.

Layout Design

- **Multi-level partitioning**
- ➢ Layout1: One dimension: ***samples***
  Data is kept in a single dataset in order of samples.
- ➢ Layout2: Two dimensions: ***regions*** + ***samples***
  Data of a specific region of all samples is kept in a standalone dataset in order of samples.

Layout1: One dimension: samples

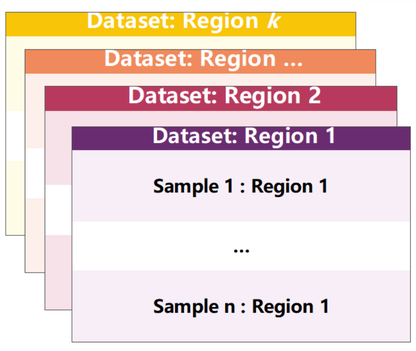
Layout1 : Chromosome 1 data stored as a whole in a single HDF5 file

Dataset:All regions	
Sample 1	region 1 region … region *k*
…	
Sample n	region 1 region … region *k*

Index		
Sample 1		
Block 1	..	Block *u*
…		
Sample n		
Block 1	…	Block *v*

Layout2: Two dimensions: regions and samples

Layout2 : Chromosome 1 data stored as sliced regions in a single HDF5 file



Index		
Sample 1		
Region 1	…	Region *p*
…		
Sample n		
Region 1	…	Region *q*

- **Blocking and indexing**

For Layout1 a block is defined as dataset for an individual sample. For Layout2, a block is defined as dataset for a given region at an individual sample, each block contains multiple records corresponding to 5M genetic loci region.

Index facilitates locating required set of data records. Layout1: two-dimensional indices built at an interval of 16k records per sample. Layout2: similar to Layout1, but indices built according to block border.

- **Compression**

Data kept in text format without redundant fields (fields filtered out comparing with original dataset). Fast blosc compression is employed to reduce data volume.

Experiment & Results

Experiment

- Dataset

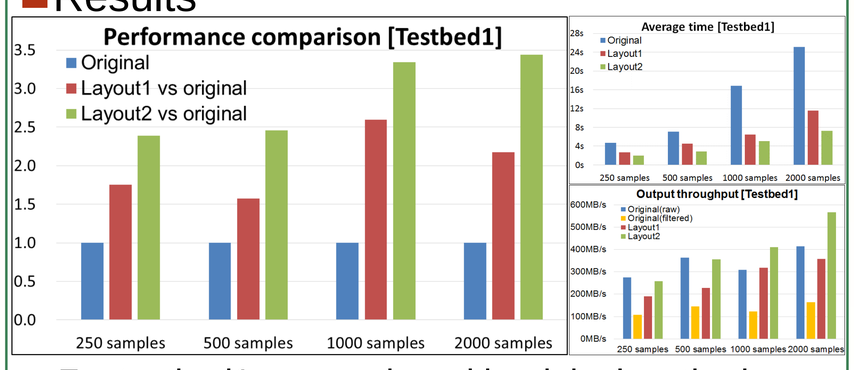
Only Chromosome 1 is selected for testing.

- ➢ Original data for all chromosomes are kept in 2,000 CRAM files with a total size of 560 GB (approximately 44GB for Chromosome 1).
- ➢ One HDF5 file containing Chromosome 1 only, with a size of 122 GB.

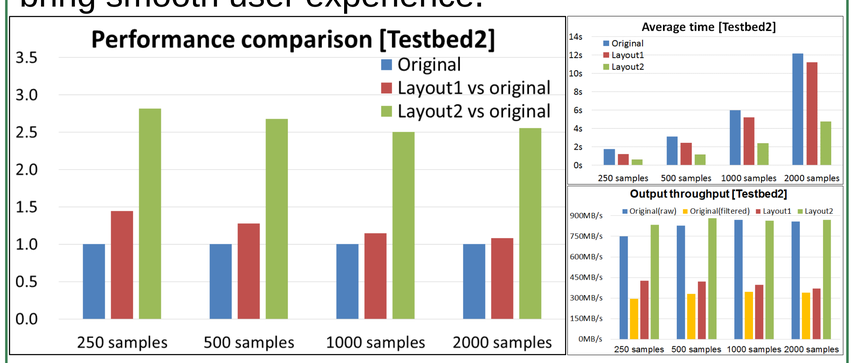
- Methodology
- ➢ Each job reads data of a specific region for a given number of samples. Timing is only for data access, without extra processing.

- Testbed
- ➢ Testbed1: 50 compute nodes from Tianhe-2 supercomputer system, each with 24 cores CPU, 64GB RAM, Lustre file system. 50 jobs runs in parallel, each with 24 threads to load data.
- ➢ Testbed2: Fat node with 128 cores, 6TB RAM, memory file system. 50 jobs are submitted, but only 5 jobs each with 24 threads are executed in parallel at the same time.

Results



For testbed1, comparing with original method, Layout1 and Layout2 perform very well and have higher throughput when the number of samples increases. Layout2 has sustainable advantages increasing more samples, and works slightly better than Layout1 because it can reach block borders directly while Layout1 needs extra search. Original method has relatively high throughput due to data of redundant fields. Both Layout1 and Layout2 reduce metadata congestion for parallel file system, and bring smooth user experience.



For testbed2, results reflect the software overhead. Layout2 has higher speedup and throughput, which reveals the benefits of our new approach. Layout2 performs better than Layout1 as the reasons mentioned above. Layout1 shows slightly lower speedup and throughput when the number of samples increases, which is relevant to higher software overhead.

Conclusion

The results show that for parallel file system like Lustre, our new approaches, especially two-dimensional layout, enable scalable analysis for thousands of samples, and preserve the potential towards massive scale analysis. By designing new layouts optimized for such kind of analysis, significant performance improvement is achieved in data access stage, which is critical for such data intensive applications.

Acknowledgements

This work is supported by National Natural Science Foundation of China, Grant No. U1611261.