DOI: 10.14529/jsfi150104

Neo-hetergeneous Programming and Parallelized Optimization of a Human Genome Re-sequencing Analysis Software Pipeline on TH-2 Supercomputer

Xiangke Liao[1]***, Shaoliang Peng***[1]***, Yutong Lu***[1]***, Chengkun Wu***[1]***, Yingbo Cui***[1]***, Heng Wang***[1]***, Jiajun Wen***[1]

The growing velocity of biological big data is the way beyond Moore's Law of compute power growth. The amount of genomic data has been explosively accumulating, which calls for an enormous amount of computing power, while current computation methods cannot scale out with the data explosion. In this paper, we try to utilize huge computing resources to solve the big dataproblems of genome processing on TH-2 supercomputer. TH-2 supercomputer adopts neo-heterogeneous architecture and owns 16,000 compute nodes: 32000 Intel Xeon CPUs + 48000 Xeon Phi MICs. The heterogeneity, scalability, and parallel efficiency pose great challenges for the deployment of the genome analysis software pipeline on TH-2. Runtime profiling shows that SOAP3-dp and SOAPsnp are the most time-consuming parts (up to 70% of total runtime) in the whole pipeline, which need parallelized optimization deeply and large-scale deployment. To address this issue, we first design a series of new parallel algorithms for SOAP3-dp and SOAPsnp, respectively, to eliminatethe spatial-temporal redundancy. Then we propose a CPU/MIC collaborated parallel computing method in one node to fully fill the CPU/MIC time slots. We also propose a series of scalable parallel algorithms and large scale programming methods to reduce the amount of communications between different nodes. Moreover, we deploy and evaluate our works on the TH-2 supercomputer in different scales. At the most large scale, the whole process takes 8.37 hours using 8192 nodes to finish the analysis of a 300TB dataset of whole genome sequences from 2,000 human beings, which can take as long as 8 months on a commodity server. The speedup is about 700x.

Keywords: biological big data; parallelized optimization; TH-2; sequence alignment; SNP detection; whole genome re-sequencing.

Introduction

Whole genome re-sequencing refers to the procedure of genome sequencing of different individuals of species with a reference genome and the execution of divergence analysis on individual or colony. Using whole genome re-sequencing, researchers can obtain plentiful information of variations like single nucleotide polymorphisms (SNP), copy number variation (CNV), insertion/deletion (INDEL) and structure variation (SV). Its application areas include clinical pharmaceuticals (cancer, Ebola, AIDS, leukemia etc.), population genetics, correlation analysis, evolution analysis and so on. With the development of personalized medicine and sequencing technologies, the cost of sequencing has been decreasing (currently the sequencing of one human genome costs less than 1000$). Consequently, the scale of sequences is growing rapidly. However, if the accumulated data could not be analyzed efficiently, a large amount of useful information would be left unused or discarded.

BGI, one of the top three gene sequencing institutions in the world, produces 6TBs sequences every day. Their current analysis pipeline and solutions need two months to accomplish the comparison and mutation detection of those data with one single server. The DNA sequences of 2000 people constitute a dataset as large as 300TB. The ongoing million people genome project

[1]National University of Defense Technology, Changsha, China

needs to analyze 500PB of DNA sequences. Given the current analyzing efficiency, it would be a mission impossible.

Our analysis shows that sequencing alignment and mutation detection tools occupy 70% of total time. These two applications and the output data are essential to the whole pipeline; therefore, it would be beneficial for the whole pipeline if we can explore the parallelism and improve the computing scale and parallel efficiency of those two components using supercomputer.

The TH-2 supercomputer, developed by the National University of Defense Technology(NUDT), is the outstanding achievement of the National 863 Program. Now the TH-2 is installed in the Guangzhou Supercomputing Center. TH-2 is co-designed with international collaboration and adopts a new neo-heterogeneous architecture. The compute array has 16,000 compute nodes. Each node contains and 2 multi-core Xeon CPUs and 3 many-core Xeon Phi MICs.It is an innovative architecture which significantly expands the application domains of the TH-2. It is a big challenge to solve the problems of genome big data processing and high throughput biological applications on TH-2.

In this paper, we try to utilize huge computing resources to solve the big dataproblems of genome processing on TH-2 supercomputer. We aim to address the above biological big data problem by carrying out parallelization and scalability on the aforementioned key components and implementing the optimized pipeline on the TH-2. To address these issues of genome big data alignment and SNP detection, we propose a set of algorithms and parallel strategies of intra-node and inter-node.

Intra-node: We partition genome analysis tasks and data into each node of TH-2 evenly and fully. A series of methods are proposed in order to fully use 3 MICs and 2 CPUs on one node, such as three-channel IO latency hiding, elimination of computation redundancy, spatial-temporal compression, vectorization of 512 Bit wide SIMD command, CPU/MIC collaborated parallel computing method, and so on. Our programming and parallelized optimization also aim at one computation node on TH-2 in order to make our algorithms scalable and extendible when the data scale up.

Inter-node: A series ofscalable parallel algorithms and large scale programming methods are also proposed to reduce the amount of communications between different nodes according to the genome sequence data suffix and characteristics, such as fast windows Iteration and scalable multilevel parallelism. Moreover, experiments and evaluations in different scale from 512 to 8192 nodes are analyzed and shown on TH-2. And some inspiring results are generated: we can use 8192 nodes to finish the analysis a 300TB dataset of whole genome sequences from 2,000 human beings within 8.37 hours, which can take as long as 8 months on a commodity server.

1. Background

1.1. Pipeline of Human Genome Re-sequencing Analysis

The pipeline of human genome re-sequencing is composed of several components. The first step is to perform quality control of original reads in order to reduce noise. Sequence alignment and single nucleotide polymorphisms (SNP) detection are then conducted consecutively. Sequence alignment is the procedure of comparing screened sequences to the reference genome, confirming its occurring frequency and appearing location in the reference genome. The output will be stored in either SAM or BAM format. SNP calling is the procedure of detecting bases deletion, insertion, transition and perversion in DNA sequences. It takes alignment results, the

reference sequence, and curated SNP databases as input and detects SNPs. These two steps are essential for consequent analysis including correlation analysis, functional annotation, and population genomics analysis [6]. We performed runtime profiling and found that short-read alignment and variation detection are the most time-consuming parts (up to 70% of total runtime) in the whole pipeline. So we are devoted to accelerating the most worldwide used software for each part, namely SOAP3-dp and SOAPsnp correspondingly. We also ported other parts of the whole pipeline to TH-2 in order to fully utilize the computational resources of different nodes on TH-2.

1.2. Neo-heterogeneous Architecture of TH-2 supercomputer

Capable of a peak performance of 54.9PFlops, the TH-2 achieves a sustained performance of 33.9PFlops on Linpack with a performance-per-watt of 1.9GFlops/W. The compute array has 16,000 compute nodes: 32000 Intel Xeon CPUs + 48000 Xeon Phi MICs. Each node contains and two multi-core Xeon CPUs and three many-core Xeon Phi MICs. It is an innovative architecture, which significantly expands the application domains of the TH-2. We named it: Neo-Heterogeneous Architecture Compute Array, which significantly improves the compatibility, the flexibility and the usability of the applications. Here the phrase neo-heterogeneous refers to the fact that compared with CPU+GPU architecture, CPU+MIC is less heterogeneous as MIC is x-86 compatible.

Intel Many Integrated Core (MIC) architecture is also known as Intel Xeon Phi coprocessor. A MIC card is equipped with 50+ cores clocked at 1 GHz and 6 GB or more on-card memory; each core supports 4 hardware threads and owns a 512-bit wide SIMD vector process unit (VPU). Each MIC card can achieve a double precision arithmetic performance of 1TFlops per second, which makes it a competitive type of accelerator. MIC architecture is x86-compatible, which alleviates the efforts required to port applications to Xeon Phi compared to its counterpart GPUs. Some simple applications can even run directly on Xeon Phi simply after re-compilation. There are two major modes to employ Xeon Phi in an application:

1) *native* mode, where Xeon Phi has a whole copy of the application and runs the applications, just like a normal compute node.

2) *offload* mode, where the application runs as a master thread on CPU and offloads some selected part to Xeon Phi, which treats Xeon Phi as a coprocessor.

2. Related Work

Short Oligonucleotide Analysis Package (SOAP [3]) is an integrated software package includes a lot of software such as SOAPsnp, SOAP3-dp, SOAPfusion and so on. SOAP is dedicated to next generation sequencing data analysis.

SOAP3-dp is known as a fast, accurate, sensitive short read aligner [2]. It is designed by BGI and belongs to SOAP (Short Oligonucleotide Analysis Package) series analysis tools. It comes from SOAP3, a GPU-based software for aligning sort reads to reference sequence[2]. So as to adapt to the increasing throughput of next-generation sequencing, SOAP3-dp was proposed to increase the performance of SOAP3 [5] in terms of speed and sensitivity. For instance, it takes less than 10 seconds for SOAP3-dp to align length-100 single-end reads with the human genome per million reads, while SOAP3, tens of seconds. The main improvement is that SOAP3-dp can tackle lots of reads in parallel by means of exploiting compressed indexing and memory-

optimizing dynamic programming on GPU. As a result, the gapped alignments can be examined extensively and compared to other tools; SOAP3-dp has the advantages of high speed, accuracy and sensitivity [3]. To our best of knowledge, there is no MIC version for SOAP3-dp so it cannot utilize the compute power provided by MIC cards equipped on TH-2.

SOAPsnp is a popular SNP detection tool developed by BGI as a member of its SOAP (Short Oligonucleotide Analysis Package) series analysis tools. The software adopts the Bayesian model to call consensus genotype by carefully considering the data quality, alignment and recurring experimental errors. All these information is integrated into a single quality score for each base to measure the calling accuracy. SOAPsnp usually costs several days or even more to analyze a human genome, which accounts for more than 30% time of normal genome analysis pipeline. The low efficiency calls for a performance boost by advanced computing technologies [6].

3. Programming and Optimization on TH-2 supercomputer

3.1. MICA: A MIC version of SOAP3-dp

To utilize the compute power provided by TH-2, we developed MICA, a tool that can make full use of both CPU and MIC to execute alignment of short reads from high throughput sequencing. It took less than 1 hour to complete comparison of 17T short sequence data using 932 compute nodes when MICA was tested on TH-2, which would take about 3 months on a commodity server. The MICA is designed and developed by National University of Defense Technology (NUDT) and BGI, and it is open online to all biologists [7].

3.1.1. Three-channel IO Latency Hiding

MICA is optimized for the configuration of both software and hardware on Tian-he2. Considering that each compute node has three MICs, we allocate three input buffers and three output buffers to hide the delay produced by data transfer, as depicted in fig. 1.

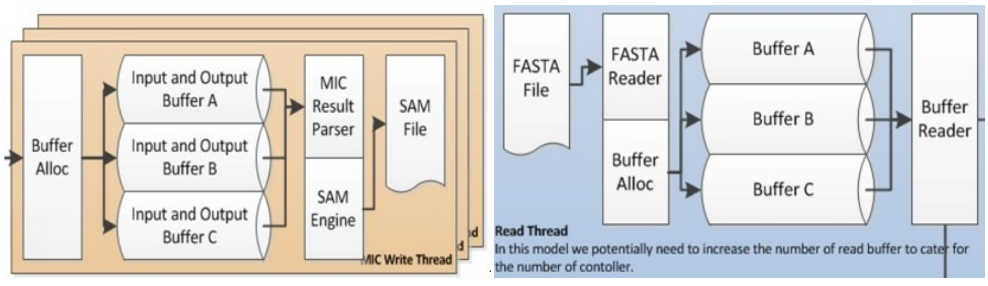

Figure 1. Three Channel IO Design

3.1.2. Vectorization of 512 Bit Wide SIMD Command

Each MIC core owns one 512-bit wide VPU, so we implement the core functions 512 Bit Wide SIMD Command to make full use of vector units of MIC. Core code can be seen as below (fig. 2).

```
mA = _mm512_add_epi32(
    rA,
    _mm512_mask_mov_epi32(dpw->rMismatch, _mm512_cmpeq_epi32_mask(_mm512_srli_epi32(rcodeB,4),rcodeC), dpw->rMa
);
mB = _mm512_add_epi32(
    rB,
    _mm512_mask_mov_epi32(dpw->rGapOpen,_mm512_cmpeq_epi32_mask(rtraceB, dpw->rMaskTraceB),dpw->rGapExtend)
);
mC = _mm512_add_epi32(
    rC,
    _mm512_mask_mov_epi32(dpw->rGapOpen,_mm512_cmpeq_epi32_mask(rtraceC, dpw->rMaskTraceC),dpw->rGapExtend)
);
// r1 stores the max score among A, B, C
r1 = _mm512_mask_mov_epi32(mA,_mm512_cmpge_epi32_mask(mB,mA),mB);
r1 = _mm512_mask_mov_epi32(r1,_mm512_cmpge_epi32_mask(mC,r1),mC);
r2 = _mm512_setzero_epi32();
r2 = _mm512_mask_add_epi32(r2, _mm512_cmpeq_epi32_mask(r1,mA), r2, _mm512_set1_epi32(0x00000001));
r2 = _mm512_mask_add_epi32(r2, _mm512_cmpeq_epi32_mask(r1,mB), r2, dpw->rMaskTraceB);
r2 = _mm512_mask_add_epi32(r2, _mm512_cmpeq_epi32_mask(r1,mC), r2, dpw->rMaskTraceC);
r1 = _mm512_and_epi32(r1,_mm512_set1_epi32(0xffff0000));
r1 = _mm512_add_epi32(r1,rcodeB);
r1 = _mm512_add_epi32(r1,rcodeC);
r1 = _mm512_add_epi32(r1,r2);
_mm512_store_epi32(dpw->matrix+coord(i,j),r1);
```

Figure 2. 512 Bit Wide Vector SIMD Code

3.1.3. Parallelized Construction and Vectorization of Smith-Waterman Dynamic Matrix

DNA sequence imprecise alignment adopts Smith-Waterman Dynamic Programming algorithm. Procedure of filling Smith-Waterman Dynamic Matrix is one of the hotspots of the whole software. The Smith-Waterman Dynamic Score Matrix is filled as below:

$$H(i,0) = 0,\ 0 \le i \le m$$
$$H(0,j) = 0,\ 0 \le j \le n$$
$$H(i,j) = \max \left\{ \begin{array}{ll} 0 & \\ H(i-1,j-1) + s(a_i,b_j) & \text{Match/Mismatch} \\ \max_{k \ge 1}\{H(i-k,j) + W_k\} & \text{Deletion} \\ \max_{l \ge 1}\{H(i,j-l) + W_l\} & \text{Insertion} \end{array} \right\},\ 1 \le i \le m, 1 \le j \le n$$

Here we calculate the values in the matrix diagonal wise, and use 512 bit wide vectorization to rewrite the original function. The new order of our filling matrix algorithm is from left to right and from top to bottom, and the Smith-Waterman Matrix can be parallelized built along diagonal, as is shown in fig. 3:

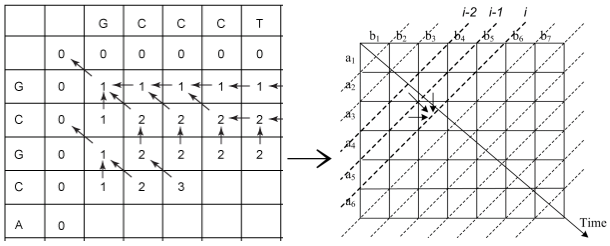

Figure 3. Parallel Constructing Smith-Waterman Matrixalong Diagonal

3.1.4. Using Inline Function to Decrease Overhead of Function Call

When it comes to the processing of large sequencing dataset, the overhead of function call cannot be ignored. For the processing of 17T sequence data, there will be 1.2P times calling of the function. So we set the core function to be inline function or macro to increase the efficiency of function execution.

3.1.5. Perfecting of Index Data

The BWT (BurrowsWheeler Transform) index array is a core data structure in SOAPsnp. It will be frequently visited by a number of threads. To avoid memory overhead, increase efficiency of CPU, we prefetching the data for next loop.

3.2. Optimization of SOAPsnp

Hotspots of SOAPsnp are two modules: likelihood and recycle, as shown in the tab. 1.

Table 1. Time breakdown of SOAPsnp

Module	*likelihood*	*recycle*	**Other modules**	**Total**
Time/s	1478.36	752.98	40.59	2552.18
Percentage	57.93%	29.50%	12.57%	1

We analyzed the characteristics of the original implementation of SOAPsnp and developed high performance version - mSOAPsnp. We devised a number of optimization methods to achieve an optimized performance assisted by MIC.

3.2.1. Compression of 4-D Sparse Matrix

The main function of the likelihood module is to traverse the base_info matrix, which is a 4-D sparse matrix with 0.08% non-zero elements. The matrix saves the information of short sequence, and occupies 1MB space. Frequent access to the base_info matrix leads to low efficiency due to unnecessary memory accesses. So we compressed the 4-D sparse matrix, only saving non-zero elements of the matrix; we also modified the original algorithm from 4 levels of loops to only one loop, which reduces the memory overhead down to 5%. By this improvement, the program gets a speedup of 2.3x.

3.2.2. Fast Windows Iteration

The recycle module handles sequences that are located at overlapped area of two windows in the phase of windows iteration, copying SNP information (100 SNPs) from one window to the next one. Therefore, the recycle module has a large amount of duplicated variables and memory copying operations. We found that copying the base_info matrix introduces the most overhead. The size of base_info matrix is 1MB, every SNP has one base_info matrix, so in each iteration of shifting the window, we need to copy 100 base_info matrixes, which is 100MB in total. The whole program needs to iterate over 50000 windows, which results in a total amount of 5TBs memory copying. Such a big overhead is not acceptable for efficient processing.

With sparse matrix compression, we decrease the memory usage down to 800B, every windows changing-over just needs 80KB, it is 875.6x faster than the original way.

3.2.3. Elimination of Redundancy via Building a Fast Table

When calculating the SNP probability, the likelihood module reads revising probability from p_matrix and stores the results to type_likely after calculation. The results has 262144 probable values, while there are 500 trillion calculation in one pass execution of the program. So that

most of the calculations are duplicated and can be avoided. We built a fast access table for computed values to eliminate redundant computation. We pre-calculate all probable results and save them into the table. Although it introduces additional 64MBs memory overhead, repeated calculation is prevented. For results storing, we save the data based on spatial locality of SNP probability calculation, so that we can increase the cache-hit rate. The likelihood module is 5x faster when employing this method.

3.2.4. Consistency Sort of the Gradient

Bases need to be sorted in likelihood the module to ensure the access order is in accordance with the calculation model. The original sorting algorithm consists of three rounds of multidimensional mixed gradient algorithm. For improvement, we carefully analyzed the characteristics of base data, and adjusted the original data storage pattern, making the original three rounds of hybrid gradient consistency gradient sorting sorted into one round.

3.2.5. Scalable Multilevel Parallelism

The original single CPU program is weak in scalability. For better performance, we exploit multilevel parallelism, which employs multi-threading for sites within the window and MPI based parallel processing across different compute nodes. A nearly linear scalability was achieved in our test.

3.2.6. Optimization of Utilizing MIC

After all above improvements, we found that the overall performance of the program is still not as good as expected and we discovered that the program has a low utilization rate of CPU and MIC wide vector unit. After a thorough analysis, we found that the original program is exhibiting poor locality of data accesses in the core algorithm.

To address this problem, we used loop expansion and space padding to improve spatial locality of data and the amount of calculation in loops, as well as making full use of CPU's 256-bit VPU and MIC's 512-bit VPU. Our test demonstrated that the utilization rate of CPU and MIC VPUs are up by more than 30%.

3.2.7. CPU/MIC Collaborated Parallel Computing Method

We used the offload mode to utilize the computing power of MIC. However, in a "naive" offload mode, CPU will be in an idle state after launching the offload part, and will only continue to work after the computation on MIC side completes and return the results back to the host CPU. Consequently, many CPU cycles are wasted.

To fully utilize all CPU cycles, we design the CPU/MIC neo-heterogeneous parallel computing framework, in which the CPU can do computation currently with the offload part running on MIC. This can be achieved by using one thread on CPU to communicate with MIC and using other threads for computation.

4. Evaluations on TH-2 supercomputer

4.1. Fast Windows Iteration

We evaluated the performance of MICA and compared it with SOAP3-dp and BWA in the following configuration: MICA: Compiled with Intel C/C++ Compiler version 13.1; one to three 57-core MIC cards (8G, ECC enabled). SOAP3-dp: version r176. Compiled with GCC 4.7.2 and CUDA SDK 5.5; one nVidia GeForce GTX680 (4G); 4 CPU cores; serial BAM output. BWA-MEM: version 0.7.5a. Compiled with GCC 4.7.2; 6 CPU cores; SAM output. Host machine: Intel i7-3730k, 6-core @3.2GHz; 64G memory. The performance comparison is listed in tab. 2.

Table 2. Performance of MICA

Data set		Yh150
MICA	1 card	22751(6.32h)
	2 cards	11479(3.19h)
	2 cards scale-up	1.98X
	3 cards	7728(2.15h)
	3 cards scale-up	2.94X
	Properly paired	95.48%
SOAP3-dp	1 card	22751(6.32h)
	Properly paired	95.01%
BWA-MEM	6 cores	101466(28.19h)
	Properly paired	92.32%

To note, there is no dependence among the sequences, so the sequence comparison has inherent parallelism, the parallel efficiency will not decrease when the number of nodes increases, as illustrated in tab. 3. The code of MICA is open-source online and evaluations in detail can be seen in [7].

Table 3. Speedup of MICA

Threads	Time elapsed	Speedup	Efficiency
56	58.3137	1.0000	1.0000
112	37.7260%	1.5457%	0.7729
224	21.8856%	1.6645%	0.6661

4.2. Evaluations of SOAPsnp

4.2.1. Evaluation Environment

Program optimization effects on a single compute node were tested in the Lv-Liang Supercomputing Center of China; the large-scale test was performed in TH-2 located in the National Supercomputing Center of Guangzhou. Compute nodes in two supercomputer centers share the same configuration, as shown in tab. 4.

Table 4. Data size and computation scale

Short sequence	Reference sequence	SNP database	Output results	Computation scale
2.2 GB	49 MB	9.7 MB	3.1 GB	50million sites

4.2.2. Effects of Algorithm Optimization

We have adopted a series of strategies to optimize our algorithms. Our strategies reduce the complexity of the algorithm, improve the efficiency of the algorithm of time and space, and further improve the program inherent parallelism, which has laid a solid foundation for heterogeneous parallel optimization. The effects of our optimization are illustrated in fig. 7.

Fig. 4 shows the acceleration effect of algorithm redesigning, "o0" represents the original algorithm, "o1" represents acceleration effect of 4-d matrix dimension reduction, it gets a speedup of around 2.3X, "o1+o2" represents acceleration effect using elimination of redundancy and building fast table on the foundation of "o1", the speedup is around 1.7X, "o1+o2+o3" represents acceleration effect of consistent gradient sorting on the foundation of "o1+o2", the speed up is about 1.6X.

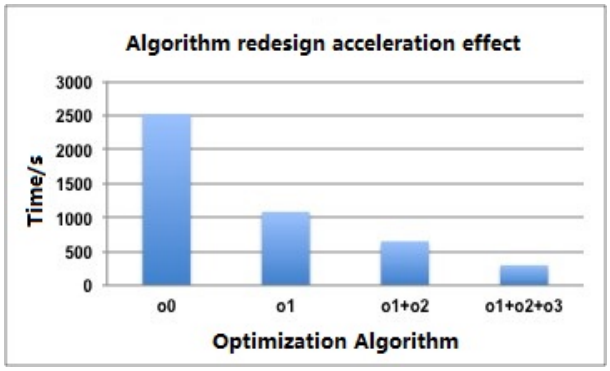

Figure 4. Algorithm Acceleration Effect of Mixed Methods

4.2.3. Overall Performance Boost via Parallelization on a Single Node

We utilized a CPU-MIC collaborated way to exploit the maximum of parallelism provided by each compute node. Fig. 5 demonstrates the speedups test on a single compute node of TH-2, using different number of MICs. It can be seen that one MIC card can achieve a performance equivalent to the CPU performance.

4.2.4. Hotspot Optimization Effect

Running time of the optimized program module is shown in Table V. Compared with the numbers before optimization (tab. 5), two hot module likelihood and recycle achieves a speedup of 85.4x and 875.6x respectively.

4.2.5. Overall Performance Boost via Parallelization and Scalability

The parallelism on a single node is mainly implemented via OpenMP while the parallelism across different compute nodes is achieve via MPI. We tested the scalability of our program on

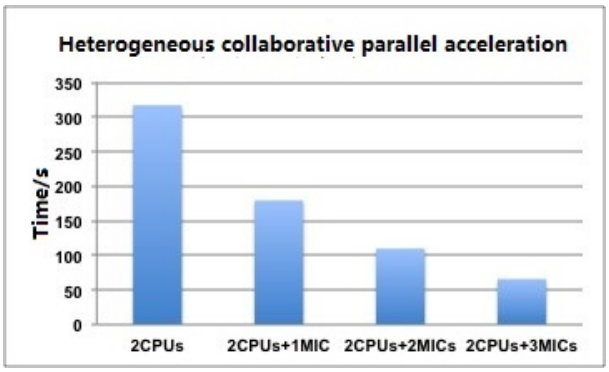

Figure 5. Heterogeneous Collaborative Parallel Accerleration

Table 5. Data size and computation scale

Module	likelyhood	recycle	Other modules	Total
Time/s	17.32	0.86	278.56	296.74
Percentage	5.84%	0.29%	93.87%	1

the TH-2 using at maximum 512 compute nodes. The scalability is depicted in fig. 6. Good scalability can be gained when we do large scale test using more human genome data (2000 human beings) on TH-2 supercomputer.

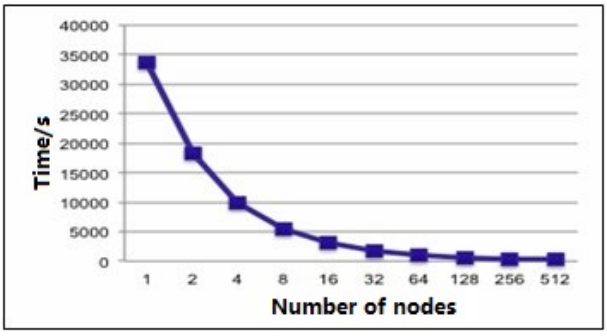

Figure 6. Scalability of the program

Conclusions

We analyzed the human genome resequencing software pipeline, and conducted in-depth parallel optimizations of two core components: SOAP3-dp and SOAP-snp. For SOAP3-dp, we improve the algorithm from a number of aspects, three-channel IO Latency hiding, vectorization of 512 Bit wide SIMD command, and parallelized construction of Smith-Waterman Dynamic matrix Etc. to accelerate the alignment procedure. We used 932 nodes on TH-2 to process 17.4TB whole human genome sequencing data, finished the work in one hour, getting a speedup of 2160x. For SOAPsnp, we propose 4-D matrix compression, fast windows iteration, scalable multilevel parallelism, CPU/MIC collaborated parallelism, and vectorization Etc. methods. We optimized the SOAPsnp in both algorithm and parallelism. The resulting program can achieve a 50x speedup on one single compute node, with a parallel efficiency over 73.6%. Tests on 512 nodes on TH-2 showed a speedup of 483.6x. It can finish the processing within one hour that would

take one week originally. At last, we analyze 300TB (2000 human beings whole genome) big data set using different scale nodes from 512 to 8192 nodes to analyze the scalability. At most, 8192 nodes on TH-2 are used to speed up the human genome resequencing analysis software pipeline deeply, which has generated some inspiring results. Using the improved pipeline, we performed an analysis of the 2000 peoples sequencing data and finished it in 8.37 hours, which would take 8 months using the original pipeline. In general, our optimized version is 700x faster. For future work, we are aiming for large-scale and long term deployment of our optimized pipeline on TH-2 supercomputer and we will perform analysis of genomics data from a much larger population of people, say 1 million human genome data.

We would like to thank Dr. Bingqiang Wang from BGI for providing the source code of SOAPsnp and Dr. Ruibang Luo and Lin Fang from BGI for providing related large scale test data. This work is supported by NSFC Grant 61272056, U1435222, 61133005, 61120106005, 91430218, and 61303191.

References

1. Marx V. Biology: The big challenges of big data[J]. Nature, 2013, 498(7453): 255-260.
2. Luo R, Wong T, Zhu J, et al. SOAP3-dp: fast, accurate and sensitive GPU-based short read aligner[J]. PloS one, 2013, 8(5): e65632. DOI: 10.1371/journal.pone.0065632.
3. Li R, Li Y, Kristiansen K, et al. SOAP: short oligonucleotide alignment program[J]. Bioinformatics, 2008, 24(5): 713-714. DOI: 10.1093/bioinformatics/btn025.
4. Li R, Yu C, Li Y, et al. SOAP2: an improved ultrafast tool for short read alignment[J]. Bioinformatics, 2009, 25(15): 1966-1967. DOI: 10.1093/bioinformatics/btp336.
5. Liu C M, Wong T, Wu E, et al. SOAP3: ultra-fast GPU-based parallel alignment tool for short reads[J]. Bioinformatics, 2012, 28(6): 878-879. DOI: 10.1093/bioinformatics/bts061.
6. Li R, Li Y, Fang X, et al. SNP detection for massively parallel whole-genome resequencing[J]. Genome research, 2009, 19(6): 1124-1132. DOI: 10.1101/gr.088013.108.
7. Chan S H, Cheung J, Wu E, et al. MICA: A fast short-read aligner that takes full advantage of Intel Many Integrated Core Architecture (MIC)[J]. arXiv preprint arXiv:1402.4876, 2014.
8. Luo R, Liu B, Xie Y, et al. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler[J]. Gigascience, 2012, 1(1): 18.
9. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM[J]. arXiv preprint arXiv:1303.3997, 2013. DOI: 10.1186/2047-217x-1-18.

Received February 15, 2015.