Performance Limitation of Sequence Alignment on the NUMA-based Architecture

Chen Shaolong*, Luo Zhenzhen, Gui Xiaolin, Yang Wei

*Software School, Jiangxi Normal University, Nanchang JX 330022, China

*schen@jxnu.edu.cn

Abstract. In recent years NUMA architecture based on multicore system leads a mainstream solution to tackle the rapidly expanding computation in bioinformatics analysis. Sequence alignment is the most computing cost in the workflow of variant analysis in bioinformatics. Although parallel coding in sequence alignment could be few difficulties, approaching a good performance in such a system is complicated. This research explores the architecture bottlenecks we usually neglect in the implementation of bioinformatics computing, NUMA effect for instance. We exploit the scalability of threads in the sequence aligners to illustrate the problem and the significance of NUMA architecture in the multicore system. The preliminary results in the experiments display that sequence aligners cannot take advantage of NUMA-based multicore architecture. The scalability of threads is deficient, even negative in time-cost at the case of large-scale genome data larger NUMA nodes.

Keywords: Bioinformatics, Variant analysis, Sequence alignment, Multicore system, NUMA architecture

1. Introduction

With the years growing, the genome data is increasing rapidly since the cost of sequencing largely decreases under the emerge of the next generation sequence technology [1]. The amount of genome data doubles every year, while the total number of transistors in one processor takes 18 months to redouble [2]. Therefore, efficient methods utilizing modern hardware infrastructure are necessary to propose when facing this challenge in bioinformatics analysis.

Sequence alignment, variant calling, and variant annotation constitute the essential workflows of genome variant analysis in bioinformatics [3]. Sequence alignment is the most time-cost and computing-consuming step which supplies significant results for the remaining two operations. Without the primary consequences in the sequence alignment, variant calling and variant annotation cannot achieve accurate results in the end.

In recent years mainstream heterogeneous system in the area of HPC(High Performance Computing) consists of multicore and manycore. NUMA(Non-Uniform Memory Access) architecture, such as Intel Xeon and AMD Opteron, dominates the research in the system of multicore. Although it is convenient to plug in coprocessors or accelerators that largely supply many cores available for calculation at a reasonable expense, these heterogeneous architectures have its bottleneck that we always ignore, especially in the system of multicore.
The contributions in this work we continue our previous works [4-5] and explore deeply the overlooked bottlenecks in the NUMA-based architecture, such as NUMA effect and memory policy, which play a significant impact on the overall performance of sequence alignment. Furthermore, we primary exploit sequence aligners in terms of scalability of the threads, to reveal how these behaviors are influenced by NUMA architecture. The preliminary results in the experiments display that scalability of the threads of sequence aligners in the current NUMA-based architecture is deficient and even has a negative impact on performance. These behaviors we observed should be noticed in the implementation for parallelization in NUMA architecture, for instance, multi-instance execution for any memory sensitive applications.

The remaining of this paper is structured as follows. Section 2 gives some background about sequence alignment and NUMA architecture. Followed by section 3 which shows performance evaluation and analysis in terms of scalability of the threads. Finally, section 4 concludes this research and enlighten our future work.

2. Background

2.1. Sequence alignment
Sequence alignment could be classified into multiple sequence alignment(MSA) and pairwise sequence alignment(PSA). PSA concentrates on comparing two sequences while MSA focuses on matching more than two [6]. MSA could be iteratively calculated by utilizing PSA. PSA(denoted as sequence alignment) is our target in this research which involves two datasets, query - short read and reference - genome data. Making a score based on the similarity and difference between query and reference, to determine the optimal positions according to three kinds of mapping results: match, mismatch, and gap(deletion and insertion). There are two outstanding algorithms, Needleman-Wunsch (NW) algorithm and Smith-Waterman (SW) algorithm, which represent two different types in the pairwise sequence alignment, global alignment (NW) and local alignment (SW) [7].

After decade years many applications have been proposed for calculation of sequence alignment in bioinformatics analysis. FM-based and hash table-based algorithm, which have displayed a small memory footprint and better performance when comparing to others. For instance, BWA, SOAP, and Bowtie. Aligner BWA [8] is a significant software we focus on, however, other sequence aligners are also involved in this research.

2.2. NUMA architecture
System architecture in the processors could make a huge difference in performance but we occasionally overlook. There are three groups in mainstream system architecture. SMP (Symmetric Multi-Processor), NUMA (Non-Uniform Memory Access) and MPP (Massive Parallel Processing). NUMA [9] dominates the key structure in the multicore system. In this research, we explore some overlooked behaviors [10] in this NUMA architecture when implementing bioinformatics software.

Two prevalent NUMA systems Intel and AMD are displayed in Fig.1. They possess 4 processors and 4 NUMA nodes in the system but differs in the interconnection fashion of the processors. Intel NUMA node belongs to net architecture and AMD NUMA node is ring architecture. As shown in Fig.1, according to a fixed fashion, each processor interconnects by system line. There is only one NUMA node in each processor. Each node has its local cores that are high-speed connected with the local memory bank.

Two NUMA nodes are illustrated in Fig.2 which interconnects by system line. They own local memory banks in each node. Access from the remote core to local memory bank (Remote access as shown in Fig.2) results in more delays than that of access from the local core (Local access). Two bottlenecks in such NUMA architecture we observed and concluded.
(1) When a considerable amount of threads consumes many operations in data IO, the memory contention issue becomes a key point in the link between the cores and local memory bank. 

(2) Remote access is growing rapidly when more and more threads are applying in the process since there is a limited amount of threads in local cores. This phenomenon is frequently observed in the current system since the memory policy in default is called first-touch in most cases of OS setting. Memory congestion occupies the interconnect line between two NUMA nodes.

![Fig.1 NUMA architecture: Intel Xeon(left) vs AMD Opteron(right).](image)

This behavior reveals thread allocation and memory distribution are significant elements in the performance of NUMA system. Thread allocation is determined by thread affinity and memory policy decides how memory allocates. Local policy, or denoted as first-touch, means the allocation of memory only occurs in the memory bank of the local node where the program is currently initial. Local policy is the default memory allocation policy when the most OS is running. This policy in memory could make worse in such architecture which we observed in the experiments.

3. Performance evaluation and analysis

No data dependency in the mapping procedure ensures multi-threading ability working in the sequence alignment. However, it does not guarantee processors are utilized efficiently when multi-threading parallelism is applying. In the experiments, obviously, we observed that the efficiency of the threads decreases when the thread amount grows large enough.

These results achieved from sequence aligners, BWA, SOAP, and Bowtie, are shown in Fig.3-5, when two query datasets(17.6GB and 4.3GB) map against human genome which comes from the 1000 Genome Project. Each case in the experiments are implemented on two separated NUMA-based systems which differ in the number of processors. One equips with 2-NUMA 24-thread and the other holds 4-NUMA 64-thread. Each processor owns one NUMA node in the system.

As shown in Fig.3, obviously time-consuming of sequence aligners are reducing with a continuous increasing of the number of the threads. Degraded performance is paid attention when half thread amount and more(12+) are executed since two NUMA nodes are allocated with the threads. Worse scalability is illustrated in a larger NUMA node system. As shown in Fig.4, it is observed a negative
impact in performance that time-consuming of sequence aligners are rising when more than 32 threads are implemented. System architecture becomes a crucial influence in performance of sequence aligners which we neglect in the current execution. As shown in Fig.5, a smaller query data case shows a normal phenomenon that less time taken when more threads were applied.

This unusual behavior in these experiments could be interpreted for two reasons. (1) Remote access is growing in NUMA nodes when a large number of threads are applying which comes out a result in
memory congestion in the interconnect link. (2) Memory contention is growing where the genome data is allocated in local memory bank when an increasing thread access occurs.

4. Conclusion
In this paper, we explore some bottlenecks in the NUMA-based multicore architecture when applying in the sequence alignment of bioinformatics genome variant analysis. NUMA architecture leads various distances among cores and memory banks. Local access and remote access could produce a very different bandwidth and latency among the cores. Additional, such architecture determines thread allocation and memory policy could be significant elements of performance. According to the discussion we made in this architecture, we implemented experiments in the scalability of threads with three sequence aligners to reveal their weakness of performance. Afterwards, the results we got in the experiments have shown that thread allocation and memory policy are two considerable factors on the performance we cannot neglect. In the future, we would stretch our work to other applications in the workflows of variant analysis, and give a deeper exploration and evaluation on NUMA-based multicore architecture.

Acknowledgments
This research is partially supported by Youth Doctor Initial Project of Jiangxi Normal University (No.12020361), Natural Science Foundation of Jiangxi Province (No.20192BAB217007), and National Nature Science Project (No.61967010).

References
[1] Goodwin S, McPherson JD and McCombie WR 2016 Coming of age: ten years of next-generation sequencing technologies 1 J.Nat Rev Genet 7 pp33-51
[2] Prabhu C S R, Sreewallabh C, Aditya M, Rohit G and Jenila L 2019 Big data analytics in bioinformatics J.Big Data Analytics: Systems, Algorithms, Applications pp275-286
[3] Lappalainen T, Scott AJ, Brandt M and Hall IM 2019 Genomic analysis in the age of human genome sequencing 177(1) pp70-84
[4] Shaolong C and Senar MA 2019 Exploring efficient data parallelism for genome read mapping on multicore and manycore architectures J.Parallel Computing 87 pp11-24
[5] Shaolong C and Senar MA 2018 Improving performance of genomic aligners on Intel Xeon Phi-based architectures IEEE Int. Parallel and Distributed Processing Symposium Workshops (IPDPSW)
[6] Lladós J, Fernando C and Fernando G 2019 Optimization of consistency-based multiple sequence alignment using big data technologies The Journal of Supercomputing pp1310-1322
[7] Hakim A, Kashzwari A, Tiwari R and Sharma J 2019 Performance analysis of DNA sequencing using smith-waterman algorithm on FPGA J.Journal of VLSI Design Tools & Technology pp44-50
[8] Li H and Durbin R 2009 Fast and accurate short read alignment with burrows–wheeler transform J.Bioinformatics 25 pp1754–1760
[9] Entezari M, Reza, Younghyun C and Bernhard E 2020 Evaluation of memory performance in NUMA architectures using stochastic reward nets J.Journal of Parallel and Distributed Computing
[10] ZhengMing Y, YiPing Y 2020 A scalable lock on NUMA multicore J.Concurrency and Computation: Practice and Experience