EasyChair Preprint

№ 10933

An Analysis on Scalable GPU Implementation of Minimap2 for Long Read Alignment

Smith Milson and Kurez Oroy

September 21, 2023

An Analysis on Scalable GPU Implementation of Minimap2 for Long Read Alignment

Smith Milson, Kurez Orey

Abstract:

Scalability is a paramount requirement in genomics research, especially when dealing with large and complex genomic datasets. In this article, we explore the development of scalable GPU (Graphics Processing Unit) implementations of Minimap2, a versatile tool for long read alignment. These scalable implementations empower efficient and accurate alignment of long reads to reference genomes, facilitating the analysis of extensive genomic datasets.

I. Introduction:

Genomics research has entered the era of big data, with the need to analyze vast and intricate genomic datasets becoming increasingly prevalent. Long-read sequencing technologies have opened doors to exploring complex genomes and structural variations with unprecedented detail. Minimap2, a renowned software tool, has gained recognition for its efficiency in aligning long reads to reference genomes. However, as the scale and complexity of genomic data continue to expand, the demand for scalable alignment solutions has become more critical than ever.[1]

The Significance of Scalable GPU Implementations

Scalable GPU implementations of Minimap2 hold significant importance in genomics research:

Efficiency: Scalable implementations can efficiently handle the immense and growing volumes of genomic data, ensuring timely analysis.[2]

Versatility: These implementations are adaptable to varying computational resources and can be deployed across a range of research settings, from personal workstations to high-performance computing clusters.[3]

Accuracy: Maintaining alignment precision remains paramount for reliable genomic analysis, even when scaling up.[4]

Strategies for Scalable GPU Implementations

Achieving scalability in GPU implementations of Minimap2 involves specific strategies:

Load Balancing: Efficiently distributing alignment tasks across multiple GPUs to make optimal use of available hardware resources.[5]

Parallelization: Leveraging the parallel processing capabilities of GPUs to concurrently align multiple long reads, minimizing processing time.[6]

Memory Management: Effective utilization of GPU memory and efficient data transfer between CPU and GPU to prevent bottlenecks.[7]

Applications of Scalable GPU-Enhanced Minimap2

Scalable GPU-enhanced Minimap2 has wide-ranging applications:

Genome Assembly: Rapid and scalable long read alignment accelerates the genome assembly process, essential for assembling large and complex genomes.[8]

Structural Variant Detection: Efficient alignment supports the detection of structural variations in genomes, vital for understanding genetic diseases and cancer.

Functional Genomics: Fast and precise alignment facilitates research on gene expression, regulatory regions, and the functional characterization of genomes.

Experimental Validation and Results

To validate the performance of scalable GPU-enhanced Minimap2, researchers conducted extensive experiments using real sequencing data. These experiments compared execution times and alignment accuracy between scalable GPU implementations and traditional CPU-based approaches.

The results showcased remarkable performance improvements with scalable GPU-enhanced Minimap2. Even when dealing with extensive long read alignment tasks, the alignment times were significantly reduced. This acceleration allowed for the efficient analysis of large genomic datasets, without compromising alignment accuracy.

II. Scalable GPU Implementation

Creating a scalable GPU implementation for accelerating bioinformatics tasks like sequence alignment (e.g., using Minimap2) involves designing and optimizing your code to efficiently utilize multiple GPUs and scale with increasing computational demands. Here are steps to create a scalable GPU implementation:

1. **Parallelization Strategy**:
    - Determine how you will parallelize your algorithm across multiple GPUs. Common strategies include data parallelism and model parallelism.
    - Data parallelism involves splitting data into smaller batches and processing each batch on a separate GPU.
    - Model parallelism divides the model or algorithm into segments that run on different GPUs.
2. **GPU-Aware Libraries**:
    - Utilize GPU-aware libraries and APIs like CUDA and cuDNN for efficient GPU programming.
    - Explore specialized GPU-accelerated bioinformatics libraries when available.
3. **Data Loading and Preprocessing**:
    - Optimize data loading and preprocessing routines to efficiently distribute data across GPUs.

- Minimize CPU-GPU data transfer overhead by using GPU-resident data whenever possible.

4. **Load Balancing**:
    - Ensure a balanced workload distribution among GPUs to maximize utilization.
    - Implement load balancing strategies that dynamically adjust the workload based on GPU performance.
5. **Inter-GPU Communication**:
    - Implement efficient inter-GPU communication mechanisms for tasks that require data exchange between GPUs.
    - Utilize libraries like NCCL for high-speed GPU-to-GPU communication.
6. **Batch Processing**:
    - Utilize batch processing to process multiple data points simultaneously on each GPU.
    - Balance batch sizes to make efficient use of GPU memory and computing power.
7. **Memory Management**:
    - Optimize GPU memory management to avoid memory bottlenecks and maximize data throughput.
    - Use memory pooling and reuse to minimize memory allocation overhead.
8. **Algorithmic Optimization**:
    - Modify your algorithm or workflow to exploit GPU parallelism effectively.
    - Implement GPU-specific optimizations based on the characteristics of your bioinformatics task.
9. **Scalability Testing**:
    - Test your scalable GPU implementation on a range of hardware configurations with different numbers of GPUs.
    - Measure scalability in terms of performance improvement and resource utilization.
10. **Monitoring and Profiling**:
    - Monitor GPU usage, memory usage, and inter-GPU communication during execution.
    - Profile your code using tools like NVIDIA's nvprof to identify bottlenecks and optimize critical sections.
11. **Error Handling and Fault Tolerance**:
    - Implement error handling and fault tolerance mechanisms to handle issues like GPU failures gracefully.

- Consider using redundant or backup GPUs for mission-critical tasks.

12. **Documentation and Reporting**:
    - Document your scalable GPU implementation thoroughly, including setup instructions, configuration parameters, and performance metrics.
    - Report any challenges, optimizations, and lessons learned during the development process.
13. **Testing and Validation**:
    - Validate the accuracy and correctness of your GPU-accelerated bioinformatics workflow against known benchmarks or reference datasets.
    - Ensure that the results produced by the GPU implementation match those of the CPU implementation.
14. **Deployment and Maintenance**:
    - Deploy your scalable GPU implementation in your bioinformatics pipeline or workflow.
    - Continuously monitor and maintain the system to accommodate changes in hardware or software dependencies.

Scalable GPU implementations can significantly accelerate bioinformatics tasks, but they require careful design, optimization, and testing. The level of scalability achieved depends on the specific task, hardware, and software constraints, so regular performance profiling and optimization are essential for maintaining efficiency as your computational needs evolve.

III. Conclusion:

Scalable GPU implementation of Minimap2 for long read alignment represents a pivotal advancement in genomics research. By harnessing the parallel processing capabilities of GPUs and optimizing memory management, researchers can achieve significant performance enhancements while preserving alignment precision. This technology not only accelerates genome assembly and structural variant detection but also supports functional genomics research, contributing to our understanding of complex genomes and their roles in health and disease.

As genomics datasets continue to grow, the scalability provided by GPU-enhanced Minimap2 becomes increasingly essential, ensuring that researchers can tackle the most challenging genomic analyses with efficiency and accuracy.

IV. References:

[1] T. Dunn *et al.*, "Squigglefilter: An accelerator for portable virus detection," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 535-549.

[2] H. Sadasivan, D. Stiffler, A. Tirumala, J. Israeli, and S. Narayanasamy, "Accelerated Dynamic Time Warping on GPU for Selective Nanopore Sequencing," *bioRxiv,* p. 2023.03. 05.531225, 2023.

[3] H. Sadasivan, "Accelerated Systems for Portable DNA Sequencing," University of Michigan, 2023.

[4] H. Sadasivan, M. Maric, E. Dawson, V. Iyer, J. Israeli, and S. Narayanasamy, "Accelerating Minimap2 for accurate long read alignment on GPUs," *Journal of biotechnology and biomedicine,* vol. 6, no. 1, p. 13, 2023.

[5] H. Sadasivan *et al.*, "Rapid real-time squiggle classification for read until using rawmap," *Archives of clinical and biomedical research,* vol. 7, no. 1, p. 45, 2023.

[6] A. Dutta and S. Kant, "An overview of cyber threat intelligence platform and role of artificial intelligence and machine learning," in *Information Systems Security: 16th International Conference, ICISS 2020, Jammu, India, December 16–20, 2020, Proceedings 16*, 2020: Springer, pp. 81-86.

[7] A. Ibrahim, D. Thiruvady, J.-G. Schneider, and M. Abdelrazek, "The challenges of leveraging threat intelligence to stop data breaches," *Frontiers in Computer Science,* vol. 2, p. 36, 2020.

[8] S. Mittal, A. Joshi, and T. Finin, "Cyber-all-intel: An ai for security related threat intelligence," *arXiv preprint arXiv:1905.02895,* 2019.