POSTER: Asynchrony versus Bulk-Synchrony for a Generalized N-body Problem from Genomics

Marquita Ellis, Aydın Buluç, Katherine Yelick
The University of California at Berkeley & Lawrence Berkeley National Laboratory
Berkeley, California, U.S.A.
{mme,abuluc,yelick}@berkeley.edu

Abstract

This work examines a data-intensive irregular application from genomics, a long-read to long-read alignment problem, which represents a kind of Generalized N-Body problem, one of the "seven giants" of the NRC Big Data motifs [5]. In this problem, computations (genomic alignments) are performed on sparse and data-dependent pairs of inputs, with variable cost computation and variable datum sizes. In particular, there is no inherent locality in the pairwise interactions, unlike simulation-based N-Body problems, and the interaction sparsity depends on particular parameters of the input, which can also affect the quality of the output. We examine two extremes to distributed memory parallelization for this problem, bulk-synchrony and asynchrony, with real workloads. Our bulk-synchronous implementation, uses collective communication in MPI, while our asynchronous implementation uses cross-node RPCs in UPC++. We show that the asynchronous version effectively hides communication costs, with a memory footprint that is typically much lower than the bulk-synchronous version. Our application, while simple enough to be a kind of proxy for genomics or data analytics applications more broadly, is also part of a real application pipeline. It shows good scaling on real input problems, and at the same time, reveals some of the programming and architectural challenges for scaling this type of data-intensive irregular application.

CCS Concepts: • **Computing methodologies** → **Parallel computing methodologies**; **Distributed computing methodologies**; • **Applied computing** → *Computational genomics*; *Bioinformatics*; *Molecular sequence analysis*; Computational proteomics.

Keywords: HPC, genomics, bioinformatics, big data

PPoPP '21, February 27-March 3, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8294-6/21/02.
https://doi.org/10.1145/3437801.3441580

ACM Reference Format:
Marquita Ellis, Aydın Buluç, Katherine Yelick. 2021. POSTER: Asynchrony versus Bulk-Synchrony for a Generalized N-body Problem from Genomics. In *26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '21), February 27-March 3, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3437801.3441580

1 Extended Abstract

Per the 2013 National Academies report on Big Data Frontiers, "Generalized N-body problems (Gray and Moore, 2001) include virtually any problem involving distances, kernels, or other similarities between (all or many) pairs (or higher-order n-tuples) of points". This describes classic, well-understood N-Body problems like particle simulation. It also describes currently challenging problems with non-Euclidean or higher-dimensional similarity metrics, proliferating from bioinformatics, computer vision, computational chemistry, and other Big Data domains [3, 5, 15]. Long-read to long-read alignment, a genomics problem, is representative of this challenging subclass.

To achieve acceptable fidelity for genomic analysis in the presence of sequencing errors, *pairwise alignment* is used to compute similarity or "distance" between reads (strings representing DNA fragments) from a set of input reads. For a read pair with maximum length n, pairwise alignment is $O(n^2)$ in general (Needleman-Wunsch 1970, Smith-Waterman 1981) [10, 12]. However, average-case $O(n)$ methods, such as X-drop [16] can achieve good accuracy for long-read to long-read alignment in particular [7]. Yet, individual long reads vary widely in length, roughly $n \in (10^2, 10^5)$, and while algorithms like X-drop offer significantly lower expected complexity, they also entail much more irregularity due to their dynamic adaptation and early-termination heuristics. The overall problem can be solved by computing the pairwise alignment of each input read to each other input read in $O(N^2 \times n^2)$. However, the size of the input read sets vary with the size of the underlying genomes and the sequencing depth; in practice, the all-to-all approach is intractable for all but the smallest workloads. Domain-specific analysis [2, 4, 7–9, 14] can effectively reduce the number of interactions (alignments). However, these analyses also typically yield unstructured sparse (data-dependent) interaction

patterns across the input reads, over which the many-to-many pairwise alignment must be performed. This poses challenges for parallel load balancing and communication cost minimization, and remains the dominant scalability bottleneck of practical pipelines [6]. Similar challenges appear in other computational biology problems as well, such as protein alignment [11] and protein searches in massive data sets [13].

This work examines practical approaches, near the bulk-synchronous and asynchronous extremes, for many-to-many long read alignment. Our bulk-synchronous approach in MPI 2.0 uses collective communication for the many-to-many read exchange, supporting independently parallel computation of many alignments in subsequent compute steps. It utilizes all available memory to minimize the number of supersteps and maximize bandwidth-utilization. The data-intensive nature of the application and the fact that there is no perfect partitioning strategy (in general) as we illustrate, motivates alternative one-sided or asynchronous approaches. Our asynchronous approach, retrieves reads required for sets of local alignment tasks asynchronously, and overlaps retrievals with unrelated computations. Given its performance and programmability for the target use-case, the asynchronous version is implemented in UPC++ [1] with RPCs. From the outset, it is unclear whether the latency of many fine-grained messages, with asynchronous framework overheads, can compete with a small number of bisection-bandwidth-bound exchanges. Our analysis considers both communication performance and memory footprint for fixed workloads.

Results were collected on a Cray XC40, Cori KNL at NERSC, using three real workloads. Strong scaling results up to 32K cores are shown, with runtime breakdowns and memory footprints. Within a node, both codes effectively reduce the runtime from just under 1 hour on a single core to just under 1 minute on 64 cores. For another larger workload, both codes reduce the runtime from an estimated $\approx$ 7 hours on a single core to under 10 seconds on 4K cores (64 nodes).

While both versions achieve good scalability, the asynchronous version effectively hides communication costs, despite the irregularity of remote-lookups and alignment computations, resulting in up to 20% better efficiency. The asynchronous version also maintains a typically lower memory footprint over the bulk-synchronous version. Across workloads, implementations, and architectures, we expect that bandwidth for many-to-many exchanges supporting bulk-synchronous approaches, versus the (balance of) alignment computation to one-sided or asynchronous message latency supporting asynchronous approaches, will determine the relative performance and scalability of each approach.

Acknowledgments

This work was supported in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration as part of the ExaBiome project at Lawrence Berkeley National Laboratory and by the National Science Foundation as part of the SPX program under Award number 1823034 at UC Berkeley.

References

[1] John Bachan, Scott Baden, Steven Hofmeyr, Mathias Jacquelin, Amir Kamil, Dan Bonachea, Paul Hargrove, and Hadia Ahmed. 2019. UPC++: A High-Performance Communication Framework for Asynchronous Computation. 963–973. https://doi.org/10.1109/IPDPS.2019.00104

[2] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. 2015. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology* 33, 6 (2015), 623–630.

[3] Nicolas Bock, Matt Challacombe, and Laxmikant V Kalé. 2016. Solvers for O(N) Electronic Structure in the Strong Scaling Limit. *SIAM Journal on Scientific Computing* 38, 1 (2016), C1–C21.

[4] Mark J. Chaisson and Glenn Tesler. 2012. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics* 13, 1 (2012), 238.

[5] National Research Council et al. 2013. *Frontiers in massive data analysis*. National Academies Press.

[6] Marquita Ellis, Giulia Guidi, Aydin Buluç, Leonid Oliker, and Katherine Yelick. 2019. DiBELLA: Distributed Long Read to Long Read Alignment. In *48th International Conference on Parallel Processing (ICPP 2019)* (Kyoto, Japan). https://doi.org/10.1145/3337821.3337919

[7] Giulia Guidi, Marquita Ellis, Daniel Rokhsar, Katherine Yelick, and Aydın Buluç. 2018. BELLA: Berkeley efficient long-read to long-read aligner and overlapper. *bioRxiv* (2018), 464420.

[8] Heng Li. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 34, 18 (2018), 3094–3100.

[9] Gene Myers. 2014. Efficient Local Alignment Discovery amongst Noisy Long Reads. In *Algorithms in Bioinformatics*, Dan Brown and Burkhard Morgenstern (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 52–67.

[10] Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.

[11] Oguz Selvitopi, Saliya Ekanayake, Giulia Guidi, Georgios Pavlopoulos, Ariful Azad, and Aydın Buluç. 2020. Distributed many-to-many protein sequence alignment using sparse matrices. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–14.

[12] Temple F. Smith and Michael S. Waterman. 1981. Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147, 1 (1981), 195–197.

[13] Martin Steinegger and Johannes Söding. 2017. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology* 35, 11 (2017), 1026–1028.

[14] Chuan-Le Xiao, Ying Chen, Shang-Qian Xie, Kai-Ning Chen, Yan Wang, Yue Han, Feng Luo, and Zhi Xie. 2017. MECAT: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads. *Nature Methods* 14, 11 (2017), 1072.

[15] Katherine Yelick, Aydın Buluç, Muaaz Awan, Ariful Azad, Benjamin Brock, Rob Egan, Saliya Ekanayake, Marquita Ellis, Evangelos Georganas, Giulia Guidi, et al. 2020. The parallelism motifs of genomic data analysis. *Philosophical Transactions of the Royal Society A* 378, 2166 (2020), 20190394.

[16] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. 2000. A greedy algorithm for aligning DNA sequences. *Journal of Computational biology* 7, 1-2 (2000), 203–214.