Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 2 April 2015
Applications Note

Sequence analysis

MSAProbs-MPI: Parallel Multiple Sequence Aligner for Distributed-Memory Systems

Jorge González-Domínguez[1,*], Yongchao Liu[2], Juan Touriño[1] and Bertil Schmidt[3]

[1]Grupo de Arquitectura de Computadores, Universidade da Coruña, Campus de Elviña, 15071 A Coruña, Spain
[2]School of Computational Science and Engineering, Georgia Institute of Technology, 266 Ferst Drive, 30332 Atlanta (GA), USA
[3]Institut für Informatik, Johannes Gutenberg Universität Mainz, Staudingerweg 9, 55128 Mainz, Germany

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Summary: *MSAProbs* is a state-of-the-art protein multiple sequence alignment tool based on hidden Markov models. It can achieve high alignment accuracy at the expense of relatively long runtimes for large-scale input datasets. In this work we present *MSAProbs-MPI*, a distributed-memory parallel version of the multithreaded *MSAProbs* tool that is able to reduce runtimes by exploiting the compute capabilities of common multicore CPU clusters. Our performance evaluation on a cluster with 32 nodes (each containing two Intel Haswell processors) shows reductions in execution time of over one order of magnitude for typical input datasets. Furthermore, *MSAProbs-MPI* using eight nodes is faster than the GPU-accelerated *QuickProbs* running on a Tesla K20. Another strong point is that *MSAProbs-MPI* can deal with large datasets for which *MSAProbs* and *QuickProbs* might fail due to time and memory constraints, respectively.
Availability and implementation: Source code in C++ and MPI running on Linux systems as well as a reference manual are available at http://msaprobs.sourceforge.net
Contact: jgonzalezd@udc.es

1 Introduction

MSAProbs (Liu *et al.*, 2010) is a cutting-edge tool to compute protein multiple sequence alignments (MSAs). A number of recent benchmark studies such as Sievers *et al.* (2011); Rivas and Eddy (2015); Katoh and Standley (2016) have shown its high accuracy. However, the runtime of *MSAProbs* can be relatively high for input datasets consisting of several hundreds or a few thousand sequences. Since molecular biologists frequently need to compute such multiple alignments, reducing runtimes is of high importance. The *QuickProbs* tool (Gudyś and Deorowicz, 2014) addressed this problem by adapting the *MSAProbs* approach to GPUs. Nevertheless, such specialized hardware accelerators have limited memory that can make *QuickProbs* fail for large input datasets.

In this paper we present *MSAProbs-MPI*, a message-passing parallelization of *MSAProbs* that returns the same alignment results but in much shorter times by exploiting the resources available on common multicore clusters. Acceleration of MSA on compute clusters has been recently discussed (Orobitg *et al.*, 2015). However, the accuracy of this parallelization is limited to that of *T-Coffee* (Notredame *et al.*, 2000). Our approach obtains the same accuracy as *MSAProbs* (higher than *T-Coffee*, as shown in the studies cited in the previous paragraph) and, as will be shown in Section 3, is able to significantly reduce alignment runtimes.

2 Implementation

MSAProbs-MPI uses a progressive alignment approach based on four stages: 1) Calculation of posterior probability matrices and pairwise distances; 2) Construction of a guide tree using all the pairwise distances, as well as weighting of the sequences; 3) Weighted consistency transformation of all posterior probability matrices; 4) Final alignment using the guide tree and the transformed probability matrices, followed by an iterative refinement. We refer to Liu *et al.* (2010) for more detailed information. All *MSAProbs-MPI* configuration parameters are specified in the command line. An explanation of all the arguments, as well as installation instructions, are included in the reference manual of *MSAProbs-MPI*. The Supplementary Table 1 shows the percentage of runtime spent by each stage during different *MSAProbs* executions on a single node. The first and third stages are responsible for most execution time as they work over all possible sequence pairs ($O(n^2)$). Consequently, we have focused on reducing the runtime of these two stages using a hybrid MPI/OpenMP approach.

In *MSAProbs-MPI* all processes start their execution by reading the whole input file in parallel with efficient MPI I/O routines. Subsequently, they calculate the posterior probability matrices and pairwise distances of different sequence pairs. All processes deal with the same number of pairs to balance their workload. Our tool includes a second level of parallelism in this first stage by maintaining the OpenMP directives of *MSAProbs* and

Table 1. Evaluation of *MSAProbs-MPI* for 8 and 32 nodes containing two Intel Haswell 2680 processors each (using one MPI process per node and 24 OpenMP threads per process). It is compared to the original *MSAProbs* (version 0.9.7) using 24 threads on a single node and to *QuickProbs* (version 2.0) executed on one NVIDIA K20 GPU. All alignments have been computed using five passes of consistency transformations and one pass of iterative refinement. Speedups are calculated over the *MSAProbs* runtimes and *b* indicates the employed number of blocks. Experiments marked as "-" failed due to memory constraints.

Dataset	Num. Seqs.	Avg. Length	*MSAProbs*	*MSAProbs-MPI* (8 nodes)			*MSAProbs-MPI* (32 nodes)			*QuickProbs*	
			Time(m)	Time(m)	Speedup	*b*	Time(m)	Speedup	b	Time(m)	Speedup
PF02330	708	266	11.70	2.58	4.53	1	1.13	10.35	1	3.57	3.28
PF05103	968	219	18.80	4.35	4.32	1	1.90	9.89	1	5.92	3.18
PF04777	1022	348	38.82	7.97	4.87	1	3.32	11.69	1	11.57	3.36
PF07085	975	512	64.88	13.67	4.75	1	5.58	11.63	1	23.02	2.82
PF10150	1635	691	336.18	68.92	4.88	10	23.83	14.11	5	-	-
PF08699	1543	885	495.92	106.02	4.68	10	31.42	15.78	5	-	-

thus creating several threads per MPI process. These threads work with different sequence pairs in parallel but each thread can only compute the probability matrices and pairwise distances of sequence pairs associated to its corresponding MPI process. At the end of the first stage the posterior probability matrices are saved in memory (each process only stores the matrices of its associated pairs) and the pairwise distances are gathered into Process 0. Process 0 then sequentially constructs the guide tree and broadcasts the sequence weights. Parallelizing this second stage is not necessary as it is computationally negligible (see Supplementary Table 1).

In Stage 3 each process transforms the posterior probability matrices that have been saved in its local memory. This transformation is performed by several matrix-matrix products, and matrices stored in the memory of other processes are required. We divided the transformation stage into $P-1$ steps, P being the total number of processes, and we employed a ring communication pattern. In Step i Process p receives in a buffer the matrices stored in Process $(p+i)\ mod\ P$ and performs the associated matrix products. For very large datasets this buffer can cause memory problems. Consequently, we developed a block-based approach where the matrices associated to each process are divided into b smaller blocks (b is specified by the user through command line). Instead of one message per step, each process sends and receives b messages and performs the matrix products with the corresponding data. Hence, using more blocks increases the number of messages and the communication performance overhead, but the advantage is that the required size of the buffer is reduced. Nonblocking MPI routines are used to to overlap the matrix products and the communications in order to reduce the mentioned performance overhead. Note that, due to the required buffers to asynchronously send and receive matrices and additional data structures (dense matrices) necessary to save partial results of the products between communication steps, the memory requirements of this phase are higher than in *MSAProbs*. Similar to Stage 1, each process can create several threads that calculate the different matrix products in parallel.

Finally, the transformed matrices are gathered by Process 0, which also completes the final alignment and writes the output file. *MSAProbs-MPI* follows the approach of *QuickProbs* and provides OpenMP parallelism to the final alignment (Stage 4) on Process 0.

3 Results

A cluster consisting of 32 nodes (connected through InfiniBand FDR) containing two Intel Haswell 2680 processors each (24 cores and 128GB of RAM per node), and six different real protein families datasets downloaded from the Pfam database were used for performance evaluation. Table 1 shows the results of the original *MSAProbs* and our hybrid MPI/OpenMP implementation. Both tools were compiled with Intel ICC/MPI version 16.0.2 and full optimization options (-O3). The table also includes the runtime of *QuickProbs* on a K20 GPU, compiled with the OpenCL support of GCC version 4.9.2 and full optimization options. We do not include a comparison with other available distributed-memory-based MSA tools such as *MTA-TCoffee* as they have been shown to be less accurate for several benchmarks (Orobitg *et al.*, 2015).

We have also verified that the alignments provided by *MSAProbs* and *MSAProbs-MPI* are exactly the same. Consequently, we can assess that our parallel approach maintains the accuracy of the original tool. Table 1 shows that the use of our hybrid MPI/OpenMP implementation on a multicore cluster significantly reduces runtimes for the selected datasets with varying number of input sequences and average sequence lengths. For instance, *MSAProbs* needs more than eight hours to align the large *PF08699* dataset, even when exploiting the 24 cores of the node, while our 32-node distributed-memory version reduces this runtime to only around half an hour (speedup of 15.78). The Supplementary Tables 2-4 and Figure 1 provide information about the performance of the different stages of the algorithm, showing that both the calculation of the posterior probability matrices and the weighted consistency transformation are accelerated. Among them, the first stage obtains higher speedups as no communication overhead is involved. Finally, the experimental evaluation shows that eight nodes of our cluster are enough for *MSAProbs-MPI* to outperform *QuickProbs* on specialized accelerator hardware. Furthermore, our tool is able to complete the alignments of the largest datasets while *QuickProbs* fails due to GPU memory constraints.

References

Gudyś, A. and Deorowicz, S. (2014). QuickProbs: a fast multiple sequence alignment algorithm designed for graphics processors. *PLOS One*, **9**(2).

Katoh, K. and Standley, D. M. (2016). A simple method to control over-alignment in the MAFFT multiple sequence alignment program. *Bioinformatics*, **Online**.

Liu, Y. *et al.* (2010). MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition posterior probabilities. *Bioinformatics*, **26**(16), 1958–1964.

Notredame, C. *et al.* (2000). T-Coffee: a novel method for fast and accurate multiple sequence alignment. *J. of Molecular Biology*, **32**(1), 205–217.

Orobitg, M. *et al.* (2015). High performance computing improvements on bioinformatics consistency-based multiple sequence alignment tools. *Parallel Computing*, **42**, 18–34.

Rivas, E. and Eddy, S. R. (2015). Parameterizing sequence alignment with an explicit evolutionary model. *BMC Bioinformatics*, **16**(406).

Sievers, F. *et al.* (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, **7**(539).