A Parallel Framework for Ab Initio Transcript-Clustering

Dhananjai M. Rao
Miami University
Oxford, Ohio
raodm@miamiOH.edu

ABSTRACT

Clustering is used to partition genomic data into disjoint subsets to streamline further processing. Since inputs can contain billions of nucleotides, performance is paramount. Consequently, clustering software is typically developed as a tightly coupled monolithic system which hinders software reusability, extensibility and introduction of new algorithms as well as data structures. Having experienced similar issues in our own clustering software, we have developed a flexible and extensible parallel framework called PEACE. The objective of the framework is to ease design, implementation, and use of various clustering methods without compromising performance. This paper presents the PEACE framework, its software architecture, parallel infrastructure, and distributed data structures along with a case study of developing a clustering algorithm. Case studies of developing filters, heuristics, and comparison algorithms are also discussed to illustrate modularity and extensibility of PEACE which enables software reuse in unique ways that may not have been foreseen when individual components were developed.

KEYWORDS

Clustering; Expressed Sequence Tag (EST); Object-Oriented Design Patterns; distributed caches, Minimum Spanning Tree (MST); MPI; parallel framework

ACM Reference Format:
Dhananjai M. Rao. 2018. A Parallel Framework for Ab Initio Transcript-Clustering. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3183440.3194995

1 INTRODUCTION

Next-generation sequencing (NGS) technologies produce genetic data in fragments that need to be suitably post-processed using bioinformatics software to obtain the full dataset [1]. A common approach used to enable tractable post-processing analysis of NGS data is called *clustering*, which is the process of segregating the reads according to the transcript from which they were derived [3]. Clustering permits further processing and analysis to focus on a related subset of data, thereby reducing computational complexities.

However, clustering has to deal with complete set of sequencing output data and therefore it is a computationally challenging problem, typically $O(n^2)$. Therefore, performance is paramount and sophisticated parallel clustering algorithms and distributed software tools are necessary to efficiently cluster today's data sets. Hence, clustering software tend to be monolithic and tightly coupled to extract maximum performance. Tight coupling combined with poor cohesion exacerbates extensibility and reusability of software modules. These issues with clustering software are indeed akin to a similar situation with other bioinformatics software as summarized in [2] — *"Historically, all these sequence assemblers – each representing many man-years of effort – appear to require complete and costly overhaul, with each introduction of a new short-read or long-range technology."*

The initial version of our clustering software system [3] also suffered from aforementioned shortcomings. These issues motivated redesign and implementation of reconfigurable clustering pipelines that are flexible, extensible, reusable, and importantly performant. This short paper summarizes key details on the new object-oriented framework of PEACE, its parallel infrastructure, and its distributed data structures. Note that the focus of this paper is on the design and effectiveness of the generic software infrastructure provided by PEACE and not on the specifics of a given clustering solution.

2 PEACE FRAMEWORK

The primary objective of the PEACE framework is to provide a modular, extensible, and performant infrastructure that eases incorporating new algorithms and data structures in various phases of clustering. Modularity and extensibility has been realized by leveraging object-oriented design patterns [4] and architecting the framework as a collection of loosely coupled subsystems. The subsystems have been designed and developed to ease parallelization by adopting the Single Program Multiple Data (SPMD) paradigm. PEACE uses the Message Passing Interface (MPI) to enable parallel and distributed computing on Supercomputing clusters.

An architectural overview of the subsystems constituting the core of the framework is shown in Figure 1. The top-level `PEACE` class performs the core task of controlling and coordinating instances of four independent subsystems, namely: the INPUT subsystem, the FILTERING subsystem, the CLUSTERING subsystem, and the OUTPUT subsystem. Life cycle management of the subsystems are accomplished via the `SubSystem` class which exposes a polymorphic Application Programming Interface (API) and serves as the base class for the concrete subsystem implementations. In addition, the design facilitates introduction of additional subsystems into the framework.

Subsystems in PEACE, have been designed using the façade pattern [4] to encapsulate and manage the collection of components constituting the subsystem. The `Component` class is the abstract leaf class of the subsystem hierarchy. It employs a composite pattern [4] to compose subsystems using hierarchies of components. A two-phase approach has been utilized in the main `PEACE` class

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194995

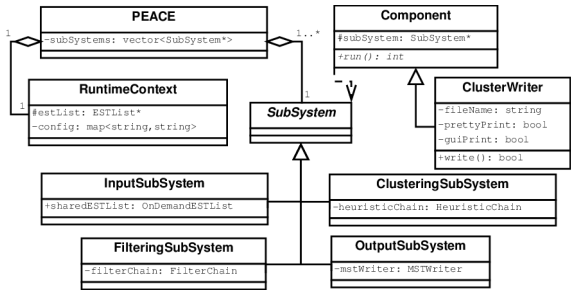
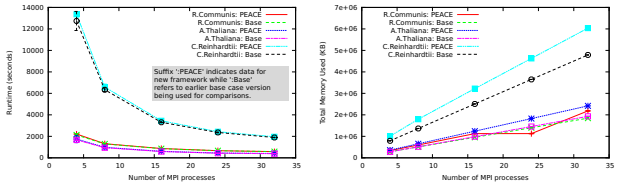


Figure 1: Overview of core sub-systems in PEACE



(a) Runtime comparison (b) Memory comparison

Figure 2: Comparison of PEACE vs. monolithic implementation

to dynamically compose and configure a subsystem using information from command-line arguments. The independent subsystems are loosely data-coupled via a shared, singleton [4] object called the `RuntimeContext` shown in Figure 1 that uses an implicit publish-subscribe style interactions. The architecture also permits the components to be used in a ad hoc manner via the framework's interactive console.

The input subsystem provides a consistent interface to read different file formats. It provides features to prefetch data to improve performance. The filtering subsystem permits different filters to be connected to form sophisticated pipelines. The clustering subsystem streamlines development of various algorithms. A key component in the framework is a distributed caching system. The framework provides various utility methods to manage distributed caches along with interfaces to streamline inter-component interactions. Caches provide interfaces to streamline management of frequently used reads to improve performance. The framework also provides interfaces to prune caches as fragments are added to the MST to minimize memory footprint. The output subsystem enables generating results in a variety of file formats.

3 EXPERIMENTS

The motivation for this research is to design a flexible and performant framework. We have implemented various heuristics, filtering, and clustering algorithms to enable an "apples-to-apples" comparison. The algorithms to be implemented were chosen based on their popularity and availability of software tools for comparison. The algorithms were suitably incorporated into PEACE and their functionality has been verified using a variety of data sets. We have used the following three large biological data sets with nucleotide (NT) sequences in FASTA file format — ❶ *R. Communis*: 57,690 reads, 709±199 NT/read, ❷ *A. Thaliana*: 76,941 reads, 427±128 NT/read, and ❸ *C. Reinhardtii*: 189,975 reads, 553±183 NT/read.

Having verified correct functionality, we compared performance of our modular framework against its monolithic counterpart. The time for clustering and the peak memory usage is shown in Figure 2(a) and Figure 2(b) respectively. The experiments were conducted on our cluster which consists of 36 compute nodes interconnected by Myrinet. Each node has dual quad-core 2.26 GHz Intel Xeon E5520 CPUs and 24 GB of memory. These experiments indicated that the object-oriented implementation using the PEACE framework provided comparable performance with < 5% degradation. Detailed profiling indicated that the degradation was primarily due to overhead of polymorphic calls used in the C++ implementation. The slight degradation is the cost of achieving increased interoperability and extensibility of the object-oriented subsystems. However, profile guided optimizations can be used to further narrow this performance gap. The memory usage of the new framework is about 20% to 25% higher as shown in Figure 2(b). This is due to additional data maintained by PEACE to support a broad range of functionality. The charts in Figure 2 also establishes that the framework did not introduce changes to the overall scalability or add overheads to the communication calls performed by the clustering algorithms.

4 CONCLUSION

This paper proposed and assessed a modular, parallel framework (and not a specific clustering algorithm) that loosely couples components in various subsystems to construct a suitable software pipeline for clustering. The paper presented the design rationale and patterns used in core framework and its subsystems. The framework has been used to incorporate a variety of bioinformatics algorithms to construct different software pipelines. The comparative experiments against its monolithic predecessor, performed using different real world data sets, were presented to highlight that significant enhancements in overall quality of the software can be achieved without sacrificing much performance through effective object-oriented design. The experiments indicate that the performance degradation was less than 5% and the increase in memory footprint is about 25%. We envision PEACE to serve as a framework and testbed to ease study, design, implementation, interoperability, and use of clustering algorithms that further lead to innovations and discoveries in biology and medicine.

Software download

An extended version of this paper and the PEACE framework can be downloaded from http://pc2lab.cec.miamiOH.edu/peace.

REFERENCES

[1] E. R. Mardis. A decadeâĂŹs perspective on DNA sequencing technology. *Nature*, 470:198–203, Feb. 2011.
[2] G. Narzisi and B. Mishra. Comparing de novo genome assembly: The long and short of it. *PLoS ONE*, 6(4):e19175, 04 2011.
[3] D. M. Rao, J. C. Moler, Y. Z. Mufit Ozden, C. Liang, and J. E. Karro. PEACE: Parallel Environment for Assembly and Clustering of Gene Expression. *Nucleic Acids Research*, 38(2):W737–W742, jun 2010.
[4] J. M. Smith. *Elemental Design Patterns.* Addison-Wesley Professional, Reading, MA, 1 edition, 2012.