BIOINFORMATICS APPLICATIONS NOTE Vol. 27 no. 21 2011, pages 3072–3073
doi:10.1093/bioinformatics/btr523

Systems biology

Advance Access publication September 16, 2011

MapReduce implementation of a hybrid spectral library-database search method for large-scale peptide identification

Ananth Kalyanaraman[1,*], William R. Cannon[2,*], Benjamin Latt[1] and Douglas J. Baxter[3]

[1]School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, [2]Computational Biology and Bioinformatics Group and [3]Molecular Sciences Computing Facility, Environmental Molecular Sciences Laboratory, Pacific Northwest National Laboratory, Richland, WA 99352, USA

Associate Editor: Alex Bateman

ABSTRACT

Summary: A MapReduce-based implementation called *MR-MSPolygraph* for parallelizing peptide identification from mass spectrometry data is presented. The underlying serial method, *MSPolygraph*, uses a novel hybrid approach to match an experimental spectrum against a combination of a protein sequence database and a spectral library. Our MapReduce implementation can run on any Hadoop cluster environment. Experimental results demonstrate that, relative to the serial version, *MR-MSPolygraph* reduces the time to solution from weeks to hours, for processing tens of thousands of experimental spectra. Speedup and other related performance studies are also reported on a 400-core Hadoop cluster using spectral datasets from environmental microbial communities as inputs.

Availability: The source code along with user documentation are available on http://compbio.eecs.wsu.edu/MR-MSPolygraph.

Contact: ananth@eecs.wsu.edu; william.cannon@pnnl.gov

Supplementary Information: Supplementary data are available at *Bioinformatics* online.

Received on February 4, 2011; revised on July 18, 2011; accepted on September 1, 2011.

1 INTRODUCTION

Identifying the sequence composition of peptides is of fundamental importance to systems biology research. High-throughput proteomic technologies using mass spectroscopy are capable of generating millions of peptide mass spectra in a matter of days. One of the most effective ways to annotate these spectra is to compare the experimental spectra against a database of known protein sequences. The main idea here is to generate candidate peptide sequences from the genome of the organism under study and then to use models of peptide fragmentation to generate model spectra that can be compared against each experimental spectrum. However, as samples become richer in diversity (e.g. from environmental microbial communities), the number of candidate comparisons could increase by orders of magnitude (Supplementary Figure S1). An increase in the number of candidates also increases the probability of finding high-scoring, random matches. It is therefore essential to implement a peptide identification method that is both accurate and scalable to large sizes of spectral collections and sequence databases. The prediction accuracy of peptide identification can be improved if experimental spectra are also compared against spectral libraries, although this would only exacerbate the computational demands.

Recently, Cannon *et al*. (2011) developed a novel hybrid statistical method within the *MSPolygraph* framework, which combines the use of highly accurate spectral libraries, when available, along with on-the-fly generation of model spectra when spectral libraries are not available. This method demonstrated increases of 57–147% in the number of confidently identified peptides at controlled false discovery rates. This effort to enrich quality of prediction, however, comes at an increased computational cost. While a parallel MPI version of the code exists, most users do not have access to large-scale parallel clusters. Whereas, open-source science cloud installations and commercial vendors such as Amazon provide access to MapReduce clusters on an on-demand basis.

In this article, we present a MapReduce implementation of *MSPolygraph* called *MR-MSPolygraph*. MapReduce (Dean and Ghemawat, 2008) is an emerging parallel paradigm for data intensive applications, and is becoming a *de facto* standard in cloud installations. One of the popular open-source implementations for MapReduce is the Hadoop framework. *MR-MSPolygraph* uses MapReduce to efficiently distribute the matching of a large spectral collection on a Hadoop cluster. Previously, Halligan *et al*. (2009) ported peptide identifications tools that use the database search approach onto the Amazon EC2 cloud environment. Our work incorporates the statistics of the hybrid search method in *MSPolygraph* to any cluster running the open-source Hadoop environment.

2 METHODS

MR-MSPolygraph is designed to achieve parallelism across the number of experimental spectra to be matched. The MapReduce framework requires developers to define two functions: *mapper* and *reducer*. In our case, since the processing of each spectrum is independent of one another, we take advantage of the inherent data parallelism by splitting the input experimental spectra across map tasks. More specifically, the user inputs: (i) (*queries*) a set of experimental spectra to be matched; (ii) (*database*) a fasta file containing known protein/peptide sequences; (iii) (*spectral library*) a set of peptides to be used as the spectral library (required only when the software is run in the 'hybrid' mode); and (iv) a file with quality control and output parameters. In addition, the user specifies a desired number of map tasks. The algorithm executes as follows: first, the queries are automatically partitioned into roughly equal sized chunks and supplied as input to each map task. The chunk size can be controlled either by altering the number of map tasks and/or the *min.split.size* parameter within Hadoop. Each map task then runs a modified implementation of the serial *MSPolygraph* code, which matches the

*To whom correspondence should be addressed.

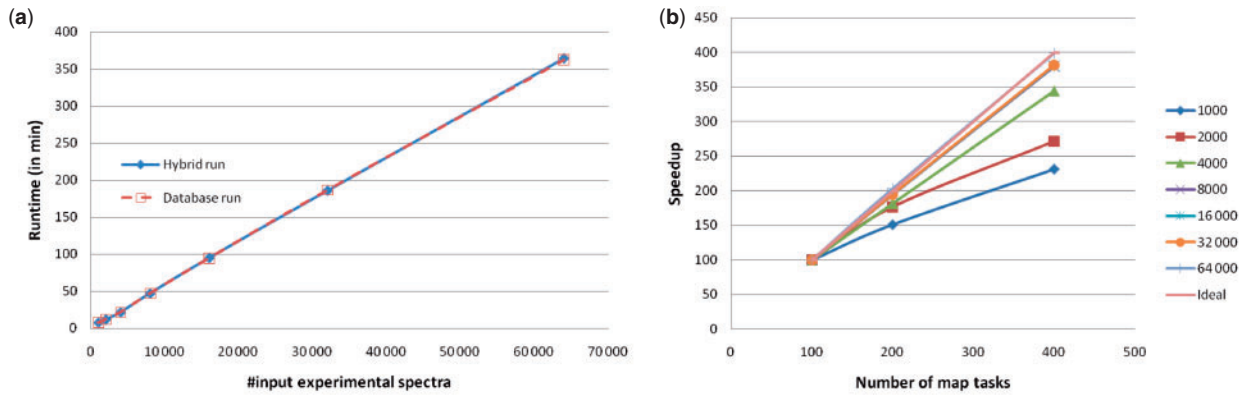


Fig. 1. Performance of MR-MSPolygraph: (**a**) Runtime as a function of the input number of spectra, keeping the number of map tasks fixed at 400; and (**b**) speedup of the hybrid version relative to 100 map tasks, for varying input sizes. The number of map tasks is generally equal to the number of cores used, although that could slightly vary as determined by Hadoop at runtime.

local batch of queries against the entire database, and also against the spectral library if run on the hybrid mode. The map tasks then output, in one file per task, a list of hits (sorted by statistical significance) for each of their queries. The algorithm has a worst-case complexity of $O(q(n_1+n_2)/p)$, where q is the number of experimental spectra, p is the number of mappers and n_1 and n_2 are the sizes of the database and spectral library, respectively. Since the mappers' output cover different subsets of queries, the reducer functionality is not used. However, if it is desired to have all the hits reported in one output file, then it can be achieved using a single reducer. More usage details and parameter descriptions can be found at the software web site.

3 RESULTS

MR-MSPolygraph was tested on the Magellan Hadoop cluster at National Energy Research Scientific Computing Center (NERSC). The cluster has 75 nodes with a total of 600 cores dedicated for Hadoop, where each node has 2 quad cores Intel Nehalem 2.67 GHz processors and 24 GB DDR3 1333 MHz RAM. These nodes run Cloudera's distribution for Hadoop 0.20.2 + 228. In our experiments, we used the following datasets: (i) a collection of 64 000 experimental spectra obtained from *Synechococcus* sp. PCC 7002; (ii) a database containing 2.65 million microbial protein sequences downloaded from NCBI GenBank; and (iii) a spectral library containing a set of 1752 *S.Oneidensis* MR-1 spectra.

Figure 1a shows the runtime of *MR-MSPolygraph* as a function of input number of spectra (from 1K to 64K). Both modes of the software, hybrid and database only, were tested. As expected, the runtime grows linearly with the input number of spectra. Furthermore, both the hybrid and database-only versions take almost identical times, indicating that the additional cost of matching against the spectral library is negligible for this input. It can be expected that this cost grows gradually with the size of spectral library used.

We also studied the performance by measuring the parallel runtime as a function of the number of map tasks used. Supplementary Table S1 shows the runtimes and Figure 1b shows the corresponding speedup up to 400 map tasks, calculated relative to the corresponding 100 mapper run. As can be observed, the runtime roughly halves with doubling of the number of map tasks and the speedup becomes linear for larger inputs (e.g. 398× on 400 map tasks for 64K spectra). This can be expected as for smaller inputs; the overhead of loading the database and spectral library is likely to dominate in larger processor sizes. Perhaps the merits of Hadoop parallelism become more evident upon comparing its performance against a serial implementation. For instance, to match the entire collection of 64 000 spectra in hybrid mode, the *MSPolygraph's* serial implementation can be estimated to take >2000 CPU hours using a state-of-the-art desktop computer; whereas, our Hadoop implementation finishes this task in ∼6 h using 400 cores. We also studied the effect of changing task granularity for each map task and the results are summarized under Supplementary Material.

ACKNOWLEDGEMENTS

We thank Dr Ramakrishnan at NERSC for offering extensive help with the set up of Hadoop environment. And, the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory.

Funding: This work was supported by the National Science Foundation (IIS 0916463 to A.K. and W.R.C.) and Department of Energy's Office of Biological and Environmental Research and Office of Advanced Scientific Computing Research under contracts (57271 and 54976 to W.R.C.).

Conflict of Interest: none declared.

REFERENCES

Cannon,W.R. *et al.* (2011) Large improvements in MS/MS based peptide identification rates using a hybrid analysis. *J Proteome Res.*, **10**, 2306–2317.

Dean,J. and Ghemawat,S. (2008) MapReduce: simplified data processing on large clusters. *Commun. ACM,* **51**, 107–113.

Halligan,B.D. *et al.* (2009) Low-cost, scalable proteomics data analysis using Amazon's cloud computing services and open source search algorithms. *J. Proteome Res.*, **8**, 3148–3153.

MapReduce Implementation of a Hybrid Spectral Library-Database Search Method for Large-scale Peptide Identification

Ananth Kalyanaraman, William R. Cannon, Benjamin Latt and Douglas J. Baxter

SUPPLEMENTARY INFORMATION

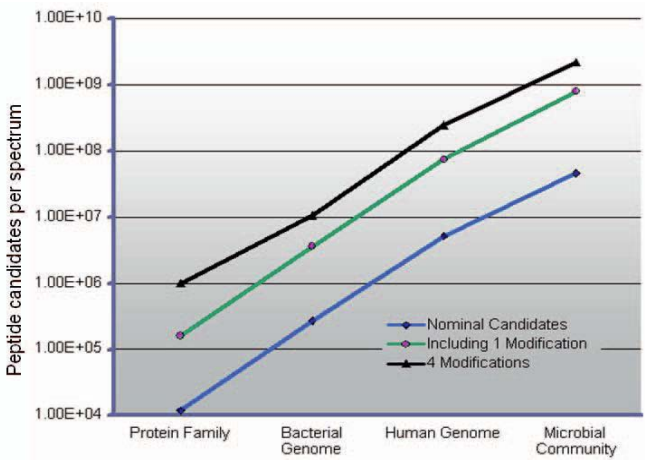


Fig. S1. Number of candidates to be evaluated against one spectrum for different data classes. The chart shows that with increasing complexity of the biological system, the number of candidates to be matched also increases.

Table S1. Parallel runtime (in minutes) as a function of the input number of experimental spectra. All runs were performed by matching the queries against the entire input database (2.65 million sequences) and spectral library (1,752 peptides). The results show that for larger inputs, it is possible to roughly half the runtime by doubling the number of map tasks. They also show that it is possible to achieve weak scaling ((i.e., increasing system size allows bigger problems to be solved in the same amount of time). This is evident from the table, where the diagonal entries show mostly comparable runtimes.

Number of experimental spectra \ Number of map tasks	100	200	400
1,000	18.63	12.32	8.05
2,000	33.33	18.83	12.27
4,000	75.28	41.45	21.90
8,000	179.85	92.55	47.43
16,000	361.78	185.92	95.12
32,000	708.72	363.37	185.80
64,000	1451.67	714.63	364.05

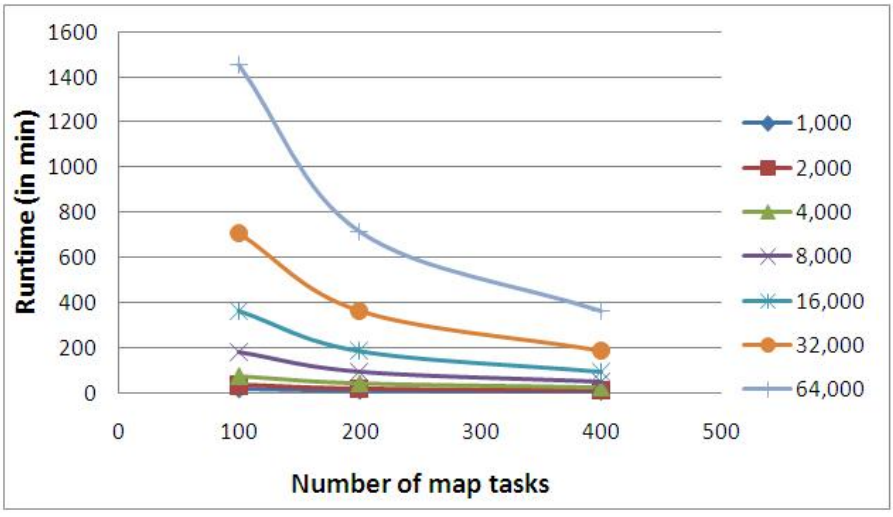
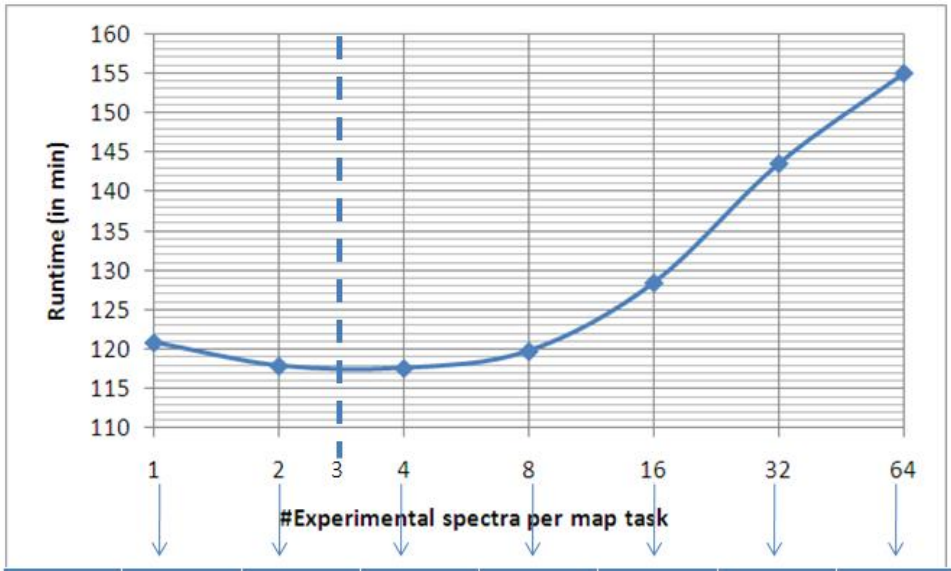


Fig S2. Parallel runtime (in minutes) as a function of the input number of experimental spectra. All runs were performed by matching the queries against the entire input database (2.65 million sequences) and spectral library (1,752 peptides).

Study showing the effect of changing granularity

The Hadoop environment provides a feature to specify an arbitrary number of map tasks and this number could be set larger in practice than the number of available cores. Intuitively, more tasks imply a smaller number of spectra per task, and hence a higher degree of parallelism and better scope for load balancing among these tasks. However, since each task loads the entire database and spectral library upon instantiation, there should also be an associated overhead. To evaluate this tradeoff, we performed hybrid runs of the 32,000 spectral collection, varying only the number of map tasks from 500 to 32,000. The number of cores in all these runs was fixed at 480, which is the maximum allowed number of map cores in the cluster. Note that, by varying the number of map tasks gradually from 500 to 32,000, we also automatically decrease the number of experimental spectra assigned to every task from 64 down to 1. Fig. S3 shows the results of these runs. As can be observed (from right to left), the runtime gradually reduces as the number of spectra per task is decreased until ~3 spectra per task, below which database initialization starts to negatively impact performance.

In other words, this curve representation is showing us a way to determine the appropriate number of map tasks for a given input. For example, the optimal number of map tasks to use for the 32,000 spectral input is roughly 10,666. Therefore, if one were to double the input, say to 64,000 spectra, the optimal number of map tasks to use can be analytically expected (not guaranteed though) to be at ~21,332. Alternatively, a more precise estimate of the optimal granularity setting can be determined empirically for other data sets *a priori* as shown in this study.



#map tasks	32000	16000	8000	4000	2000	1000	500
Runtime (in min)	120.85	117.88	117.58	119.76	128.41	143.55	155.00

Fig. S3. Effect of changing the number of spectra per map task on performance. The total number of spectra used in all runs is 32,000. Each run represents a different setting for task granularity (i.e., number of spectra assigned per map task). Empirical optimality for this data set is shown to be achieved for ~3 spectra per map task (indicated by the dotted line).