Bioinformatics, YYYY, 0–0
doi: 10.1093/bioinformatics/xxxxx
Advance Access Publication Date: DD Month YYYY
Applications Note

Genome Analysis

Aether: Leveraging Linear Programming For Optimal Cloud Computing In Genomics

Jacob M. Luber[1,2,3,4,‡], Braden T. Tierney[1,2,3,4,‡], Evan M. Cofer[1,2,4,5], Chirag J. Patel[3,*], and Aleksandar D. Kostic[1,2,4,*]

[1]Section on Pathophysiology and Molecular Pharmacology, Joslin Diabetes Center, Boston, 02215, USA, [2]Section on Islet Cell and Regenerative Biology, Joslin Diabetes Center, Boston, MA 02215, USA, [3]Department of Biomedical Informatics, Harvard Medical School, Boston, MA 02115, USA, [4]Department of Microbiology and Immunobiology, Harvard Medical School, Boston, MA 02115, USA, and [5]Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ 08544, USA

*To whom correspondence should be addressed. ‡These authors contributed equally to this work.

Associate Editor: XXXXXXX; Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Across biology we are seeing rapid developments in scale of data production without a corresponding increase in data analysis capabilities.

Results: Here, we present Aether (http://aether.kosticlab.org), an intuitive, easy-to-use, cost-effective, and scalable framework that uses linear programming (LP) to optimally bid on and deploy combinations of underutilized cloud computing resources. Our approach simultaneously minimizes the cost of data analysis and provides an easy transition from users' existing HPC pipelines.

Availability: Data utilized are available at https://pubs.broadinstitute.org/diabimmune and with EBI SRA accession ERP005989. Source code is available at (https://github.com/kosticlab/aether). Examples, documentation, and a tutorial are available at (http://aether.kosticlab.org).

Contact: chirag_patel@hms.harvard.edu and aleksandar.kostic@joslin.harvard.edu

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

1 Aether

Data accumulation is exceeding Moore's law, which only still progresses due to advances in parallel chip architecture.(Esmaeilzadeh et al., 2013) Fortunately, the shift away from in-house computing clusters to cloud infrastructure has yielded approaches to computational challenges in biology that both make science more reproducible and eliminate time lost in high-performance computing queues(Beaulieu-Jones and Greene, 2017; Garg et al., 2011); however, existing off-the-shelf tools built for cloud computing often remain inaccessible, cumbersome, and in some instances, costly.

Solutions to parallelizable compute problems in computational biology are increasingly necessary; however, batch job-oriented cloud computisystems, such as Amazon Web Services (AWS) Batch, Google preemptible Virtual Machines (VMs), Apache Spark, and MapReduce implementations are either closed source, restrictively licensed, or locked in their own ecosystems making them inaccessible to many bioinformatics labs.(Shvachko et al., 2010; Yang et al., 2007) Other approaches for bidding on cloud resources exist, but they neither provide implementations nor interface with a distributed batch job process with a backend implementation of all necessary networking.(Zheng et al., 2015; Andrzejak et al., 2010; Tordsson et al., 2012).

Our proposed tool, Aether, leverages a linear programming approach to minimize cloud compute cost while being constrained by user needs and cloud capacity, which are parameterized by the number of cores, RAM, and in-node solid-state drive space. Specifically, certain types of instances are allocated to large web service providers (e.g., Netflix) and auctioned on a secondary market when they are not fully utilized.(Zheng et al., 2015) Users bid amongst each other for use of this already purchased but unused compute time at extremely low rates (up to 90% off the listed price).(https://aws.amazon.com/ec2/pricing/) However, this market is not without its complexities. For instance, significant price fluctuations, up to an order of magnitude, could lead to early termination of multi-hour compute jobs (Figure 1A). Clearly, bidding strategies must be dynamic to overcome such hurdles.

Aether consists of bidder and batch job processing command line tools which query instance metadata from the vendor application programming interface (APIs) to formulate the linear programming problem. Linear programming is an optimization method that simultaneously solves a large system of equations to determine the best outcome of a scenario that can be described by linear relationships. The Aether bidder, described in detail in the supplementary methods, generates and solves a system of 140 inequalities using the simplex algorithm (Figure 1B). For the purposes of reproducibility, an implementation of the bidder using CPLEX is also provided as an optional command line flag.

Subsequently, the replica nodes specified by the linear programming result are placed under the control of a primary node, which assigns batch processing jobs over Transmission Control Protocol (TCP), monitors for any failures, gathers all logs, sends all results to a specified cloud storage location, and terminates all compute nodes once processing is complete (Figure 1C). Additionally, Aether is able to distribute compute

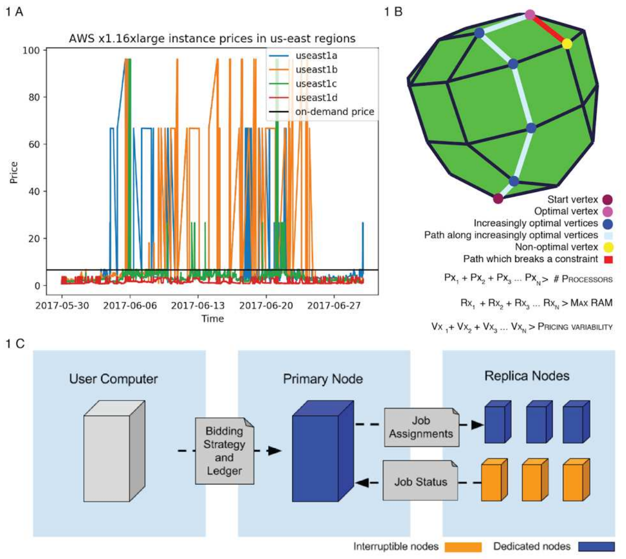

Fig. 1. Overview of Aether. A) Pricing history of an x1.16xlarge EC2 Instance showcasing variability of an order of magnitude, in both directions, for spot prices. B) Simplified example showing three constraints on a sample bidding approach minimizing an objective function cTx considering cost according to a system of constraints represented as inequalities. x_1, x_2, … x_n represent the number of specific types of compute nodes to solve for. Each inequality represents a constraint and adds another dimension to the space which the simplex algorithm needs to traverse vertices in to find ideal solution. The green line represents the optimal solution. C) General Overview of Aether.

across multiple cloud providers. Sample code for this is provided with the Aether implementation although it was not utilized in our reported tests due to cost feasibility. Our implementation runs on any Unix-like system; we ran our pipeline and cost analysis using AWS but have provided code to spin up compute nodes on either Microsoft Azure or on a user's local physical clusters.

To test our bidding approach and batch job pipeline at scale, we used our framework to de novo assemble and annotate 1572 metagenomic, longitudinal samples from the stool of 222 infants in Northern Europe (Supplementary Figure 1).(Bäckhed et al., 2015; Kostic et al., 2015; Vatanen et al., 2016; Yassour et al., 2016) The sequencing data within datasets from the DIABIMMUNE consortium ranged from 4,680 to 22,435,430 reads/sample with a median of 19,020,036 reads/sample. Assemblies were performed with MEGAHIT and annotations were done with PROKKA.(Li et al., 2015; Seemann, 2014)

Metagenomic data, typically shotgun DNA sequencing of microbial communities, is difficult to analyze because of the enormous amounts of compute required to naively assemble short sequence reads into large contiguous spans (contigs) of DNA. To accomplish our assemblies, our bidding algorithm suggested that the optimal strategy would be to spin up 30TB of RAM across underutilized compute nodes. Our networked batch job processing module utilized these nodes for 13 hours and yielded an assembly and annotation cost of ~$0.30 per sample (Supplementary Figure 2). Theoretically, the pipeline can complete in the time it takes for the longest sub-process (i.e. assembly in this case) to finish (~7 hours). Spinning up the same nodes for this long without a bidding approach would cost ~$1.60 per sample (Supplementary Figure 3). In order for on-site hardware to achieve the same cost efficiency as our pipeline, one would have to carry out on the order of 1 million assemblies over the lifespan of the servers, a practically insurmountable task (Supplementary Figure 2). Such efficiency in both time and cost at scale is unprecedented. In fact, due to resource paucity, computational costs have forced the field of metagenomics to rely on algorithmic approaches that utilize mapping back to reference genomes rather than de novo methods.(Truong et al., 2015)

Additional testing of Aether showed marginally better relative cost savings (compared to the assembly example) when tasked with aligning braw reads to the previously assembled genomes with BWA-MEM(Li, H., 2013;http://arxiv.org/abs/1303.3997); this is not surprising as shorter computational tasks are less sensitive to the risks of early spot instance termination. Additionally, in simulated runs of the bidder incorporating pricing history from periods where ask prices were approximately an order of magnitude higher than normal on the east coast of the United States (Figure 1A), Aether suggested utilization of different instance types that would have resulted in similar cost and time to completion as our actual run. To allow users to make optimal usage of these benefits, the ability to simulate bidding for different timeframes is included as a feature. By not having to potentially re-run analysis pipelines (due to being outbid on compute during runtime), we claim that utilizing Aether leads to a reduction of market inefficiencies. We have both qualitatively and empirically compared Aether to existing AWS tools such as AWS Batch and Spot Fleet Pricing (Supplemental Figure 1.3 and Supplementary Methods). Additionally, where empirical validation of benefit was possible, we have iterated on previous work and incorporated strategies such as basing a subset constrains on service level agreements (Andrzejak et al., 2010). Future directions include training the bidding algorithm to predict its own effect on pricing variability when being utilized at massive scale as well as distributing compute nodes across datacenters when enough resources are being spun up to strongly influence the market.

To our knowledge, this is the first implementation of a bidding algorithm for cloud compute resources that is tied both to an easy-to-use front-end as well as a distributed backend that allows for spinning up purchased compute nodes across multiple providers. Conceivably, this tool can be applied to any number of disciplines, bringing cost-effective cloud computing into the hands of scientists in fields beyond biology.

Acknowledgements

We thank Thomas Lane and Chengwei Luo for their feedback and review of the manuscript. We thank Tommi Vatanen for helping us with data access.

Funding

This work was funded by NIH/ NHGRI T32 HG002295, PI: Park, Peter J (J.M.L.), an AWS Research Credits for Education Grant (J.M.L. and A.D.K.), a Microsoft Azure for Research Grant (B.T.T. and C.J.P.), NIH NIEHS R00 ES023504 (C.J.P.), NIEHS R21 ES025052 (C.J.P.), NSF Big Data Spoke grant (C.J.P.), a Smith Family Foundation Award for Excellence in Biomedical Research (A.D.K.), and an ADA Pathway to Stop Diabetes Initiator Award (A.D.K.).

Conflict of Interest: none declared.

References

Andrzejak,A. et al. (2010) Decision Model for Cloud Computing under SLA Constraints. In, 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. ieeexplore.ieee.org, pp. 257–266.

Bäckhed,F. et al. (2015) Dynamics and Stabilization of the Human Gut Microbiome during the First Year of Life. Cell Host Microbe, 17, 852.

Beaulieu-Jones,B.K. and Greene,C.S. (2017) Reproducibility of computational workflows is automated using continuous analysis. Nat. Biotechnol., 35, 342–346.

Esmaeilzadeh,H. et al. (2013) Power Challenges May End the Multicore Era. Commun. ACM, 56, 93–102.

Garg,S.K. et al. (2011) Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers. J. Parallel Distrib. Comput., 71, 732–749.

Kostic,A.D. et al. (2015) The Dynamics of the Human Infant Gut Microbiome in Development and in Progression toward Type 1 Diabetes. Cell Host Microbe, 17, 260–273.

Li,D. et al. (2015) MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. Bioinformatics, 31, 1674–1676.

Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv [q-bio.GN].

Seemann,T. (2014) Prokka: rapid prokaryotic genome annotation. Bioinformatics, 30, 2068–2069.

Shvachko,K. et al. (2010) The Hadoop Distributed File System. In, 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). ieeexplore.ieee.org, pp. 1–10.

Tordsson,J. et al. (2012) Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Gener. Comput. Syst., 28, 358–367.

Truong,D.T. et al. (2015) MetaPhlAn2 for enhanced metagenomic taxonomic profiling. Nat. Methods, 12, 902–903.

Vatanen,T. et al. (2016) Variation in Microbiome LPS Immunogenicity Contributes to Autoimmunity in Humans. Cell, 165, 1551.

Yang,H.-C. et al. (2007) Map-reduce-merge: Simplified Relational Data Processing on Large Clusters. In, Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD '07. ACM, New York, NY, USA, pp. 1029–1040.

Yassour,M. et al. (2016) Natural history of the infant gut microbiome and impact of antibiotic treatment on bacterial strain diversity and stability. Sci. Transl. Med., 8, 343ra81.

Zheng,L. et al. (2015) How to bid the cloud. ACM SIGCOMM Computer Communication Review, 45, 71–84.