Balancing Thread-level and Task-level Parallelism for Data-Intensive Workloads on Clusters and Clouds

Olivia Choudhury, Dinesh Rajan, Nicholas Hazekamp, Sandra Gesing, Douglas Thain, Scott Emrich
Department of Computer Science and Engineering
University of Notre Dame
{ochoudhu, dpandiar, nhazekam, sgesing, dthain, semrich}@nd.edu

***Abstract*—The runtime configuration of parallel and distributed applications remains a mysterious art. To tune an application on a particular system, the end-user must choose the number of machines, the number of cores per task, the data partitioning strategy, and so on, all of which result in a combinatorial explosion of choices. While one might try to exhaustively evaluate all choices in search of the optimal, the end user's goal is simply to run the application once with reasonable performance by avoiding terrible configurations. To address this problem, we present a hybrid technique based on regression models for tuning data intensive bioinformatics applications: the sequential computational kernel is characterized empirically and then incorporated into an *ab initio* model of the distributed system. We demonstrate this technique on the commonly-used applications BWA, Bowtie2, and BLASR and validate the accuracy of our proposed models on clouds and clusters.**

I. Introduction

Determining an optimal runtime configuration of parallel and distributed applications is a challenging task. To tune an application on a particular system, the end-user is confronted with a wide array of controls: number of machines, number of cores per task, partitioning the data, task scheduling strategies, and so forth. It is customary in academic papers to perform an exhaustive exploration of these parameters, and then select a configuration which is optimal under some limited set of circumstances. Any change to the machines, the network, or the workload itself could result in poor performance, often **orders of magnitude** worse than what is achievable. From the user's perspective, running an application multiple times to obtain a performance curve is a waste of time.

Therefore, when designing distributed applications, our objective should be to achieve **acceptable performance the first time** by avoiding extremely bad configurations. It is impossible to provide a comprehensive model for programs of arbitrary structure. (This is simply the halting problem.) Fortunately, users in a given area of study often use similar computational patterns over and over again, which gives us an opportunity to develop a model that is highly effective within a given application domain. For example, in the realm of bioinformatics applications, many tools have the form of a database search: a program performs a search for a query pattern within a database of genomic strings. These tools are typically transformed into parallel applications by partitioning the query and/or the database, and then running multiple instances of the local multithreaded program in parallel.

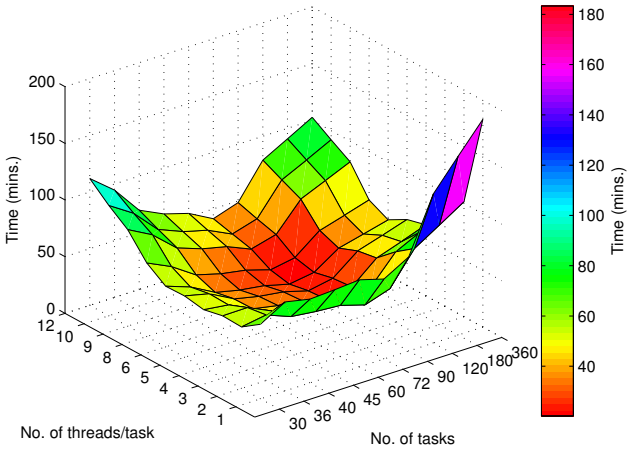

Fig. 1: Impact of multi-level concurrency (thread-level parallelism) and distributed computing (task-level parallelism) on the execution time of a data-intensive workload. Selecting a good runtime configuration can optimize resource utilization.

We present a regression model-based technique to infer optimal runtime configurations and predict execution time and resource consumption for a data-intensive workload. We perform black-box analysis of the multithreaded program (computational kernel) to determine its execution time and memory usage relative to sizes of query and database, and local concurrency. Once obtained, this model is embedded in an abstract model of the distributed system that incorporates data partitioning, data movement, and the tradeoff between local and distributed concurrency.

We show the effectiveness of our proposed approach in three commonly used bioinformatics applications BWA [1], Bowtie2 [2], and BLASR [3] that are widely used in production to study cancer [4] and variation-detection in general [5]. In all the cases, our proposed models accurately predict execution time and resource consumption. The optimal configurations identified by the models also enable cost-efficient

utilization of cloud-based resources.

II. Methods

In the realm of bioinformatics, millions of next generation sequencing (NGS) data, called reads, are often compared to the genome of a related species (reference) for further biological analysis. For the short read aligners BWA and Bowtie2, we use the genome of mosquito *Culex quinquefasciatus* [6] (562 MB in size) as reference and simulated sequences as query for these tools. Since BLASR aligns long reads to a reference genome, we consider the *Anopheles gambiae* S form genome [7] (230 MB in size) as reference and PacBio sequenced data of size 1 GB as query. We use the flexible master-worker framework WorkQueue [8] to develop distributed versions of the applications. We utilize shared computing resources from the Sun Grid Engine as workers, and an Intel x86 machine with 12-cores and 64 GB memory as the master server.

For the application-level model, we collect 343 data points from varying sizes (in GB) of reference (R) and query (Q) data, and number of threads (N) for each application. Similarly, for the system-level model we vary number of tasks or granularity value (K) and number of cores used by each task (N) to record execution time and memory consumption. We randomly select two-thirds of the data for training the regression models and the rest for testing. From the training set of 230 data points, we randomly select unique subsets of size 100 each to train the models and compute the optimal coefficients $\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\eta}^*$, and $\boldsymbol{\phi}^*$. We then use these coefficients to measure the accuracy of our model on the test data.

A. Application-level model for execution time

The time to execute a BWA, Bowtie2 or BLASR task is given by

$$T(R, Q, N) = \beta_1 \frac{RQ}{N} + \beta_2, \tag{1}$$

where $\boldsymbol{\beta} = \begin{bmatrix}\beta_1 & \beta_2\end{bmatrix}^\top$ are regression coefficients. The scalar β_2 signifies the fraction of the program that cannot be parallelized (discussed in section III-A). We compute the optimal regression coefficient $\boldsymbol{\beta}^*$ using the method discussed in [9].

B. Application-level model for memory usage

As the memory consumed is independent of query size [10], this model is not considerably impacted by the query data. Using previous notations, the model to estimate the memory usage of the applications is

$$M(R, N) = \gamma_1 R + \gamma_2 N, \tag{2}$$

where $\gamma = \begin{bmatrix}\gamma_1 & \gamma_2\end{bmatrix}^\top$ are regression coefficients. Similarly, we calculate the optimal regression coefficients $\boldsymbol{\gamma}^*$.

C. System-level model for execution time

Let T_S, T_{In}, T_{Out}, and T_J be the times to split a workload, transfer input files to workers, return output files to master, and join outputs, respectively. If T_{Tasks} is the time required to execute K tasks, then the total execution time, T_{total} is

$$T_{total} = \eta_1 T_S + \eta_2 T_{In} + \eta_3 T_{Tasks} + \eta_4 T_{Out} + \eta_5 T_J,$$

where $\boldsymbol{\eta} = \begin{bmatrix}\eta_1 & \dots & \eta_5\end{bmatrix}^\top$ are the regression coefficients. T_S, T_J depend on the data size, granularity value, and speed (D) of the disk, whereas T_{In}, T_{Out} depend on the data transferred and network bandwidth (B). We split the query into K subsets, align each ($\frac{Q}{K}$) to the reference, and then join the outputs (O) [11]. Let M and C be the number of available machines and cores per machine, respectively. For a finite number of usable cores P, if $P \leq MC$, there are more available resources than required. If $P > MC$, re-utilization of resources can limit the extent of parallelism. Thus, a key feature in minimizing T_{total} in

$$\begin{aligned} T_{total} = \eta_1 \frac{QK}{D} + \eta_2 (\frac{Q}{B} + \frac{RKN}{BC}) \\ + \eta_3 T(R, \frac{Q}{K}, N) \times \frac{KN}{MC} + \eta_4 \frac{O}{B} + \eta_5 \frac{OK}{D} \end{aligned} \tag{3}$$

is to determine optimal K and N. We calculate optimal regression coefficients $\boldsymbol{\eta}^*$.

D. System-level model for memory usage

The memory required for splitting the workload and joining the outputs depend on the data sizes. As these operations run locally on the master server, we define the memory needed at the master server

$$M_{Master}(R, Q) = \phi_1 R + \phi_2 Q, \tag{4}$$

where $\boldsymbol{\phi} = \begin{bmatrix}\phi_1 & \phi_2\end{bmatrix}^\top$ are regression coefficients.

We evaluate the optimal coefficients $\boldsymbol{\phi}^*$. Recall that the memory required by each worker is computed using (2).

III. Results

For training the models, from 230 data points, we randomly select 10^4 unique subsets, each of size 100. The Central Limit Theorem ensures that the distribution of 10^4 subsets is arbitrarily close to a normal/Gaussian distribution. For each subset, we train the regression models and evaluate optimal coefficients $\boldsymbol{\beta}^*$, $\boldsymbol{\gamma}^*$, $\boldsymbol{\eta}^*$, and $\boldsymbol{\phi}^*$ which we use to estimate the execution time and memory usage of the models using the testing data. We compare the estimated and empirical values in Table I for different parameter configurations. The low mean absolute percentage error (MAPE) indicates that our regression models predicts the execution time and memory with high accuracy. Figures depicting the behavior of the models can be found in the technical report (https://curate.nd.edu/concern/generic_files/0r967368133).

Model	Application	Configuration	MAPE(%)
Application-level Model for Time	BWA	Vary R, Fix Q,N	3.4
		Vary Q, Fix R,N	3.8
		Vary N, Fix R,Q	2.6
	Bowtie2	Vary R, Fix Q,N	1.6
		Vary Q, Fix R,N	2.2
		Vary N, Fix R,Q	1.3
	BLASR	Vary R, Fix Q,N	4.3
		Vary Q, Fix R,N	5.1
		Vary N, Fix R,Q	3.6
Application-level Model for Memory	BWA	Vary R, Fix N	3.9
		Vary N, Fix R	3.3
	Bowtie2	Vary R, Fix N	2.6
		Vary N, Fix R	1.9
	BLASR	Vary R, Fix N	4.7
		Vary N, Fix R	4.2
System-level Model for Time		Vary K, Fix R,Q,P	2.1
		Vary N, Fix R,Q,P	2.7
System-level Model for Memory		Vary R, Fix Q	2.5
		Vary Q, Fix R	3.3

TABLE I: Mean absolute percentage error (MAPE) of application-level and system-level models for time and memory on the test data for various parameter configurations. We compare empirical data for time and memory with model-predicted values to measure error. For BWA and Bowtie2, the sizes of R varied between 50 MB and 562 MB and Q varied between 500 MB and 14 GB. For BLASR, R varied between 50 MB and 230 MB, whereas Q varied between 100 MB and 1 GB. For all the tools N varied between 1 and 16.

A. Multi-level Concurrency

The execution time of a computation-heavy workload reduces with multiple computing threads, although the speedup is not proportionally improved beyond a certain number of threads, as discussed in [12]. In (1), $\frac{RQ}{N}$ denotes the fraction of the program that can be parallelized with more threads whereas β_2 signifies the fraction of the program that cannot be parallelized, including the overhead of using multiple threads.

B. Task-level Parallelism

Data-intensive applications often split data-parallel workloads into smaller, independent tasks that are executed in parallel. As the number of tasks increases, the corresponding execution time can decrease while sufficient resources are available. It is important to determine an optimal granularity value that would maximize the overall performance of the application while factoring in the additional overhead of splitting the workload, transferring data across the network, and carefully merging the individual outputs. These overheads are considered in the formulation of (3).

C. Balancing Multi-level Concurrency and Task-level Parallelism

While decomposing a workload into multi-core tasks, it is crucial to find an optimal number of tasks and threads which maximize parallelism and efficiency, respectively. Although the use of more threads reduces the overall execution time of an application, the efficiency is not considerably improved. For a given configuration of instances (number of cores and RAM), there exists an optimal split of the workload to run on a distributed system. Following the discussion in section III-B, for a given number of resources P, such that $K_i N_i = P$, where K_i is the number of tasks and N_i is the number of cores used by each task for the i^{th} measurement, we select K and N to optimize the overall execution time, inherent cost of splitting tasks, transferring data, and merging outputs.

Cores	Tasks	MP Time	Speedup	AEE Cost	MAE Cost
1	360	70	6.6	50.4	64.8
2	180	38	12.3	25.2	32.4
4	90	24	19.5	18.9	32.4
8	45	27	17.3	18.9	32.4

TABLE II: Comparison of performance for different configurations (K and N) and their corresponding cost estimation based on Amazon EC2 and Microsoft Azure-based pricing. MP = Model predicted, AEE = Amazon EC2 estimated, MAE = Microsoft Azure estimated. The system-level model for time proves to be cost-efficient. Using optimized number of tasks and cores per task can save the cost incurred in harnessing commercial cloud-based resources.

D. Using Optimal Number of Instances

For considerably large number of tasks, it might be appealing to use as many nodes or instances as possible to reduce execution time. Assuming a single instance is associated with each task, if the number of tasks is lower than the number of available instances, all the tasks can be executed in a single round after transferring. On the other hand, if the number of tasks is higher than the number of available instances, the workload now requires multiple rounds, although the total data transferred can be reduced if shared data between tasks is cached, as made possible by our implementation.

E. Reducing Cost of Operation

Many cloud computing services allows users to select resources with differing base costs. Users must ensure utilization of less resources as well as for a shorter duration to minimize cost. For example, although the price of an 8-core instance per hour may be higher than a 4-core instance, the speedup obtained from using an 8-core instance may prove to be more cost-efficient. We also consider the case where customers pay for the entire instance hour, although the resources were used for a fraction of the time bought. Determining an optimal runtime configuration leads to cost saving. In Table II, we compare and contrast the computation cost of the test workload using our model-estimated optimized parameters (Cores=4, Tasks=90) and set of default parameters for popular utility services like Amazon EC2 and Microsoft Azure [13]. Our model is demonstrated to perform at the lowest overall cost.

IV. Related work

Several frameworks exist for the development and execution of large-scale applications. Hadoop [14], an open-source implementation of the MapReduce programming paradigm [15] is widely used for data-intensive applications. Dryad [16] provides a programming framework and distributed execution

engine for DAG-based workloads. CIEL [17] extends the programming and execution model to support dynamic data dependencies and arbitrary data-dependent control flows. Contrary to this, we develop a hybrid technique that determines optimal runtime configurations to minimize utilization of cloud or cluster-based resources. Some scheduling techniques [18]–[20] address the challenge of reducing resource utilization in a distributed system while maintaining fairness at the underlying hardware level. We optimize resource utilization and cost by exploring the trade-off between parallelism achieved from multi-level concurrency and distributed computing. Models designed for performance prediction plays a significant role in efficient management and operation of workloads on a distributed system [21]. As noted in [22], [23], it is important to strike a balance between the number of computational resources used and the duration of their usage. Although earlier studies [24]–[26] have implemented different Machine Learning algorithms to predict resource utilization, we develop a two-tiered hybrid model using holdout sampling and linear regression to detect the sweet spot between thread-level and task-level parallelism.

V. Conclusion

We have discussed and addressed the problem of balancing thread-level parallelism and task-level parallelism for data-intensive workloads on clusters and clouds. We observe that utilizing more computational threads does not necessarily improve speedup and efficiency. Similarly, splitting a large workload into multiple parallel tasks incurs high overhead costs which decelerates computation. Moreover, poor utilization of individual resources may prove prohibitive. In this paper, we developed a predictive modeling methodology that considers relevant constraints while solving the multi-variable optimization problem of determining runtime configuration. We test our models on arbitrarily selected data and demonstrate its predictive capability at high accuracy.

VI. Acknowledgement

This work was supported in part by National Institutes of Health/National Institute for Allergy and Infectious Diseases (grant number HHSN272200900039C to SJE) and National Science Foundation SI2-SSE grant number 1148330 to DT).

References

[1] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows–Wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[2] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with Bowtie 2," *Nature methods*, vol. 9, no. 4, pp. 357–359, 2012.

[3] M. J. Chaisson and G. Tesler, "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory," *BMC bioinformatics*, vol. 13, no. 1, p. 238, 2012.

[4] D. Wu, C. M. Rice, and X. Wang, "Cancer bioinformatics: A new approach to systems clinical medicine," *BMC bioinformatics*, vol. 13, no. 1, p. 71, 2012.

[5] R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen, and J. Wang, "SNP detection for massively parallel whole-genome resequencing," *Genome research*, vol. 19, no. 6, pp. 1124–1132, 2009.

[6] K. Megy, S. J. Emrich, D. Lawson, D. Campbell, E. Dialynas, D. S. Hughes, G. Koscielny, C. Louis, R. M. MacCallum, S. N. Redmond, *et al.*, "VectorBase: improvements to a bioinformatics resource for invertebrate vector genomics," *Nucleic acids research*, p. gkr1089, 2011.

[7] M. Lawniczak, S. Emrich, A. Holloway, A. Regier, M. Olson, B. White, S. Redmond, L. Fulton, E. Appelbaum, J. Godfrey, *et al.*, "Widespread divergence between incipient Anopheles gambiae species revealed by whole genome sequences," *Science*, vol. 330, no. 6003, pp. 512–514, 2010.

[8] P. Bui, D. Rajan, B. Abdul-Wahid, J. Izaguirre, and D. Thain, "Work Queue+ Python: A Framework For Scalable Scientific Ensemble Applications," in *Workshop on Python for High Performance and Scientific Computing at SC11*, 2011.

[9] E. K. Chong and S. H. Zak, *An introduction to optimization*, vol. 76. John Wiley & Sons, 2013.

[10] H. Li and R. Durbin, "Fast and accurate long-read alignment with Burrows–Wheeler transform," *Bioinformatics*, vol. 26, no. 5, pp. 589–595, 2010.

[11] O. Choudhury, N. Hazekamp, D. Thain, and S. Emrich, "Accelerating Comparative Genomics Workflows in a Distributed Environment with Optimized Data Partitioning," in *C4Bio at CCGrid*.

[12] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 483–485, ACM, 1967.

[13] "Windows Azure Cloud Platform." http://www.windowsazure.com. Accessed: 2013-12-21.

[14] Hadoop. http://hadoop.apache.org/, 2007.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Symposium on Operating System Design and Implementation (OSDI)*, pp. 137–150, 2004.

[16] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data parallel programs from sequential building blocks," in *Proceedings of EuroSys*, March 2007.

[17] D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand, "CIEL: a universal execution engine for distributed data-flow computing," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, NSDI'11, (Berkeley, CA, USA), p. 9, USENIX Association, 2011.

[18] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 455–466, ACM, 2014.

[19] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications," in *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pp. 352–358, IEEE, 2002.

[20] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[21] S. A. Jarvis, D. P. Spooner, H. N. Lim Choi Keung, J. Cao, S. Saini, and G. R. Nudd, "Performance prediction and its use in parallel and distributed computing systems," *Future Generation Computer Systems*, vol. 22, no. 7, pp. 745–754, 2006.

[22] S. Ibrahim, B. He, and H. Jin, "Towards pay-as-you-consume cloud computing," in *Services Computing (SCC), 2011 IEEE International Conference on*, pp. 370–377, IEEE, 2011.

[23] R. L. Grossman, "The case for cloud computing," *IT professional*, vol. 11, no. 2, pp. 23–27, 2009.

[24] R. Albers, E. Suijs, and P. de With, "Triple-C: Resource-usage prediction for semi-automatic parallelization of groups of dynamic image-processing tasks," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1–8, IEEE, 2009.

[25] R. Duan, F. Nadeem, J. Wang, Y. Zhang, R. Prodan, and T. Fahringer, "A hybrid intelligent method for performance modeling and prediction of workflow activities in grids," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 339–347, IEEE Computer Society, 2009.

[26] A. Matsunaga and J. A. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 495–504, IEEE Computer Society, 2010.