Parallel Computing in the Cloud for All Levels

Steven Bogaerts
Department of Computer Science
DePauw University
Greencastle, IN, U.S.A.
stevenbogaerts@depauw.edu

***Abstract*—This document describes a project, adaptable for all undergraduate levels, in which students access cloud computing resources and leverage them in a parallel application.**

***Index Terms*—parallelism; cloud computing; data mining**

Undergraduate projects in parallelism often use a multicore processor on a local machine, but many real-world applications of parallelism use cloud-computing resources. Thus, this adaptable project is designed to give students a first experience in parallel computing in the cloud. This project was applied in an upper-level undergraduate data mining course that uses Python with pandas and scikit-learn. However, the provided materials are also appropriate for lower-level courses, including CS1, and a CS1-level exercise set is included. The project is also highly adaptable in time spent, being suitable as a standalone assignment of a couple hours, or as a component of a much broader project.

While there exist many distinct high-level interfaces for cloud-computing resources, this project focuses on the more uniform command-line interface. Introductory exercises guide students to connect via ssh and scp, manage files, edit code via Emacs, and run code at the command line. Students then work through another tutorial on accessing the cloud computing resource itself – in this case, Jetstream via XSEDE [1], available for educational use through a simple application process. The tutorial explains the idea of a virtual machine, management of a limited computing allocation, setting up public key infrastructure for the transfer of files, configuring the virtual machine, and running code. While some technical details are specific to XSEDE, the concepts are universal. All of the above materials are suitable for all undergraduate levels.

The remainder of the project depends on the desired time allocation and the level of the students. For a short assignment in CS1, students can run the code with different parameters by following the provided exercises, and report results about running time according to Amdahl's Law. For upper-level courses, instructors can give a short assignment by providing the same code as in CS1, and instructing students in conducting a full hyperparameter grid search. In a longer project, students can develop their own code for not just the grid search itself, but also data transformation and algorithm implementation, in a wide range of application domains.

This work uses the Extreme Science and Engineering Discovery Environment (XSEDE) Jetstream at Indiana University and the Texas Advanced Computing Center, through allocation TG-CIE180012, which is supported by National Science Foundation grant number ACI-1548562.

In the data mining course, we use a Kaggle.com dataset for predicting the selling price of homes given several dozen attributes. While the dataset is a modest size, the range of useful experiments and algorithm tuning efforts is quite extensive. As such, students must use parallelism on a cloud-computing resource in order to conduct a detailed analysis.

Having established the procedures for accessing cloud computing resources, students are ready to apply these resources to obtain significant speedup in their data mining projects – starting with serial code either provided by the instructor or developed themselves. Students quickly learn that simply running this serial code on a highly-parallel virtual machine does not give automatic speedup. Rather, their code must be adapted to make use of the parallel resources. Simple adaptations include the use of scikit-learn operations with an `n_jobs` parameter, by which the programmer can specify the number of parallel processes to run simultaneously to complete that operation. For example, the function `cross_val_score` computes a k-fold cross validation score, while `GridSearchCV` conducts a grid search for hyperparameter tuning of a chosen machine learning algorithm. Both of these functions include an `n_jobs` parameter. Of course this is only the beginning, as students are also ready at this point to create parallel processes via the Python multiprocessing module. Much data processing work is embarrassingly parallel, and can be sped up significantly through this module. Through this process, students also see Amdahl's Law in action. While they have access to up to 44 virtual CPUs in a virtual machine on XSEDE, students can observe how much speedup is actually possible, based on the proportion of their code that is parallel.

To summarize, a couple hours of background is required before students are ready for parallel computing in the cloud. On the one hand, this delays the students' entry into the fun of parallel programming. On the other hand, these barriers, if never surpassed, may prevent students from ever running parallel code on anything beyond a local multicore machine. One major contribution of this "peachy assignment", then, is the adaptable exercises to get students up to speed quickly in the prerequisite concepts, with no assumption of prior experience. The materials submitted here include the introductory tutorials and exercises, the CS1-level final exercises, and the advice for upper-level students for effectively using cloud computing resources in their project. While some technical details may vary (e.g. a different cloud computing resource), these documents describe primarily universal concepts.

REFERENCES

[1] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr, "XSEDE: Accelerating Scientific Discovery," *Computing in Science & Engineering*, vol. 16, no. 5, pp. 62–74, Sept.-Oct. 2014. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MCSE.2014.80