Laboratoire d'Informatique de Grenoble
UMR 5217 - CNRS, INPG, INRIA, UJF, UPMF

Phone : (+33) 4 57 42 15 22
E-mail address : mailto:arnaud.legrand@imag.fr
https://team.inria.fr/polaris/members/arnaud-legrand/

Adresse : LIG - Bâtiment IMAG
CS 40700
38058 GRENOBLE CEDEX 9
FRANCE

Arnaud Legrand, CNRS researcher
INRIA **POLARIS** project, LIG Laboratory

Internship proposal

Modeling and Performance Prediction of Linear Algebra Functions

Advisors: Arnaud Legrand (CNRS/LIG, Grenoble), Tom Cornebize (UGA/LIG, Grenoble)
Required Skills:

- Python and C programming, shell, ssh, git
- Basis of experiment analysis (with Python or R) is a plus
- Basis of statistics is a plus (linear regression)

1 Context

There is a continued need for higher compute performance: scientific grand challenges, engineering, geophysics, bioinformatics, etc. Indeed, the technological advances driven by the home PC market have contributed to achieving high performance in commodity components. For decades, computer performance had doubled every 18 months merely by increasing the clock frequency of the processors. This trend stopped last decade for reason of electricity consumption and heat. Indeed, the computational power of a computer increases nearly sub-linearly with clock frequency while the energy consumption increases more than quadratically.

As an answer to the power and heat challenges, processor constructors have increased the amount of computing units (or cores) per processor. Modern High Performance Computing (HPC) systems comprise thousands of nodes, each of them holding several multi-core processors. Most processors also have vector extensions, multiple levels of cache, dynamic voltage and frequency scaling (DVFS) and hyperthreading.

This increased hardware complexity gives a gain of performance, but also a higher variability: predicting the duration of a function call is no longer an easy task. This makes performance evaluation even more difficult and raises many software and *devops* challenges. Why do we get such performance? Is it normal or not?

2 Environment

The members of the POLARIS team focus their research on performance evaluation and optimization of large scale systems and parallel applications. They have a strong expertise regarding parallel applications and environment for parallel programming, performance evaluation of large scale distributed systems.

Some of POLARIS members are involved in the Joint Laboratory for Petascale Computing[1] between University of Illinois at Urbana-Champaign[2] Inria[3], Argonne National Laboratory[4], Illinois' Center for Extreme-Scale Computation[5], the National Center for Supercomputing Applications[6], the Barcelona Supercomputer Center[7], Julich and the Riken. Some of their members are also involved in the European Mont-Blanc[8] (European scalable and power efficient HPC platform based on low-power embedded technology) that aims at prototyping HPC supercomputers from from today's energy-efficient solutions used in embedded and mobile devices (e.g., ARM processors). Some of the POLARIS members are also in contact with research & development groups from Bull/ATOS that strongly influence the design and capacity planning of supercomputers.

[1] http://jointlab.ncsa.illinois.edu/
[2] http://illinois.edu/
[3] http://www.inria.fr/en/
[4] http://www.anl.gov/
[5] http://iacat.illinois.edu/
[6] http://www.ncsa.illinois.edu/
[7] http://www.bsc.es/
[8] http://www.montblanc-project.eu/

3 Goal

A lot of the HPC applications rely on dense linear algebra operations, using a standard interface called *Basic Linear Algebra Subprograms* (BLAS[9]). Several state-of-the-art implementations, such as OpenBLAS[10], Atlas[11] and MKL[12], can efficiently take advantage of the features of recent processors like multithreading, vectorization, cache and accelerators.

Modelling accurately the performance of the BLAS operations is an important task. A possible end-goal is to implement **performance non-regression tests** (most continuous integration tools only address functional non-regression tests). With a good model, we could regularly measure the duration of the operations and detect any significant but subtle changes or anomalies (possibly introduced by a modification in the software stack).

This internship can be summarized in a few steps:

1. Collect and parse the specification files (`*.h`) of the BLAS library.
2. Find the list of parameter names and types for each function. For instance, the function that computes the product of two matrices has three `int` parameters (the sizes) and three `double*` parameters (the matrices).
3. Choose a possible model for the function. There is a trade-off between the complexity of the model and its accuracy. For instance, in a first approximation, we can consider that the product of two $M \times N$ and $N \times K$ matrices takes a time $t = \alpha MNK + \beta + \varepsilon$, where α, β are two parameters that should be determined empirically and ε is a random variable accounting for measurement variability. This approximation may not be satisfying for all cases, especially for small matrices where lower terms of the polynomial (e.g., MN, NK, or N) may be non-negligible (e.g., because of cache effects). Bur including all terms of the polynomial in the model may also be detrimental as over-fitting induces uncertainty that precludes effective extrapolation.
4. Design and run an experiment to evaluate the model. For instance, which matrix sizes should we test? In which order?
5. Analyze the results of the experiment. This will include data visualization, linear regressions and bayesian modeling (e.g., using STAN[13] and/or PyMC3[14]).
6. Go back to step 3 if necessary.
7. Implement performance non-regression tests using the right model.

Prototypes of steps 3 to 5 have already been developed. The goal is to automate all these steps as much as possible. We also need to track the provenance of the results (e.g., when and how was the experiment done, what were the settings of the machine, etc.) and control the variability (e.g., disable the hyperthreading and turboboost features, pin the threads to the CPU cores, etc.), hence this internship will follow the reproducible research[15] good practices. The student will use a notebook (Jupyter[16] and/or Org mode[17]) and all the experiments will be done using Grid5000[18] platform.

This internship will allow you to sharpen your experimental skills both on a practical and theoretical side. We expect the intern to be curious, rigorous, and autonomous but both advisors will happily provide all the help needed on the software, hardware, theory, experimental methodology.

[9] http://www.netlib.org/blas/
[10] https://github.com/xianyi/OpenBLAS
[11] http://math-atlas.sourceforge.net/
[12] https://software.intel.com/en-us/mkl
[13] http://mc-stan.org/
[14] https://docs.pymc.io/
[15] https://en.wikipedia.org/wiki/Reproducibility
[16] http://jupyter.org/
[17] https://orgmode.org/
[18] https://www.grid5000.fr