Parallel Computing and GPU Processing Approaches to GAIL Routines

Fabio Araujo da Silva Renan Guarese Fred J. Hickernell

Illinois Institute of Technology, Chicago, IL

Introduction & Objective

- GAIL **(Guaranteed Automatic Integration Library)** MATLAB library:
  - Set of algorithms for integration problems in n-dimensions, using Monte Carlo and Quasi Monte Carlo methods.
- Improving the performance of GAIL routines through solutions in **Parallel Computing** and **GPU Processing**.

Methodology

- Matlab's **Parallel Computing Toolbox (PCT)**: both CPU and GPU processing;
- Integrating Java classes to Matlab: parallel calculations running in Java;
- Julia's performance vs. Matlab's

Materials and Methods

- All tests were done using an 8-Core I7 2.6GHz CPU, with a NVIDIA GeForce GTX 965M GPU (1024 cores), on Windows 10.
- Among the functions adapted, *MeanMC_CLT* was chosen to be displayed in this poster.
- ***MeanMC_CLT***: Monte Carlo method to estimate the mean of a random variable.

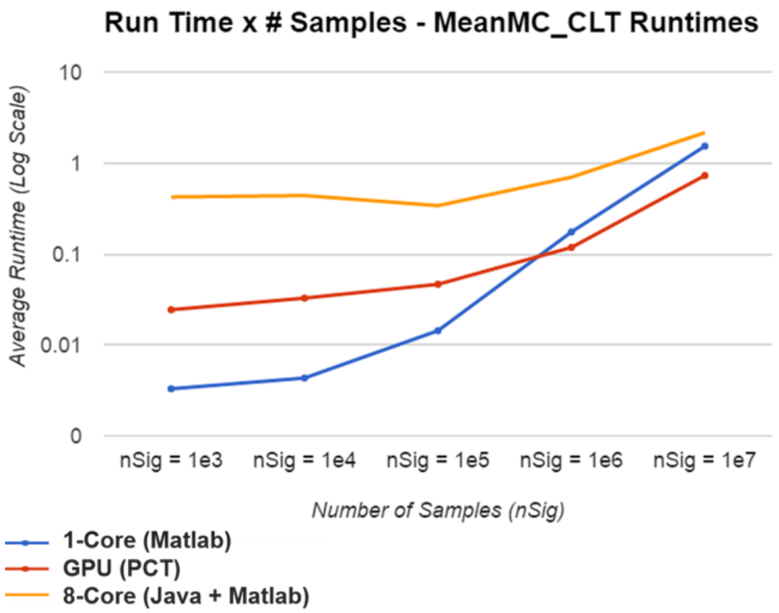
References

HICKERNELL, Fred J. ***et al.*** *GAIL: Guaranteed Automatic Integration Library* 2011. `http://gailgithub.github.io/GAIL_Dev/`

ALTMAN, Yair *Accelerating MATLAB Performance* 2014: CRC Press.

REESE, Jill; ZARANEK, Sarah *GPU Programming in MATLAB* 2012. `http://www.mathworks.com/`

First Results



Discussion

These results led us to pursue a different course: GPU Processing. We adapted this function to increase its computational cost by replacing the simple random numbers generation with the European Call Option, given by:

$$Max(S(T),0) - K, \quad where$$

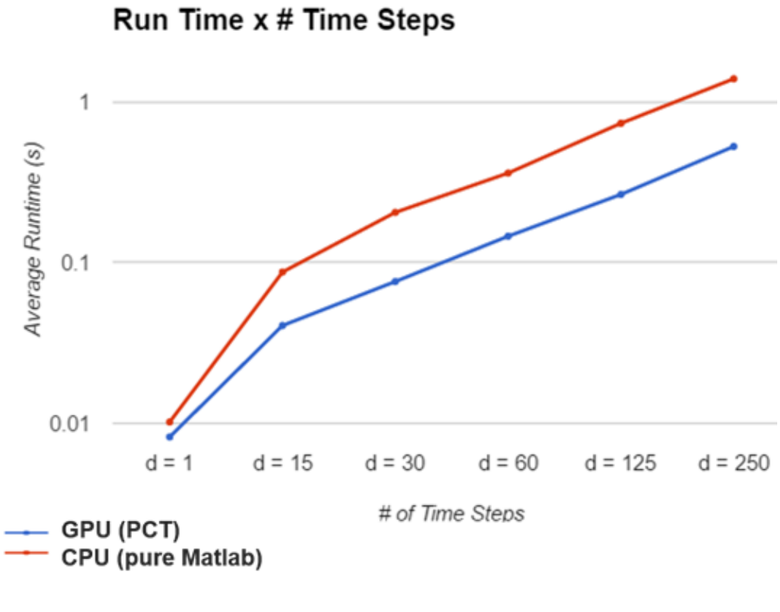
$$\mathrm{S(T)} = \mathrm{S_0} \times e^{(\frac{-\sigma^2}{2} \times T + \sigma \times \sqrt{\frac{T}{d}} \times \sum_{j=1}^{d} Z_j)}$$

$$Z_j = IID \quad N(0,1)$$

Future Work

- Perform tests on a GPU cluster;
- Look for new effective parallel approaches on CPU.

Final Results



Conclusions

- Preliminary results showed that PCT's overhead is too high on CPU;
- Tests in Julia language have proven to run quite fast on 1-core, however there's no gain on parallel computing;
- Java presented satisfactory results, but the data conversion (Java to Matlab) takes too much time, nullifying its gain;
- GPU has proven to run faster as the parameters get bigger.

Acknowledgements

CIÊNCIA SEM FRONTEIRAS CAPES INSTITUTE OF INTERNATIONAL EDUCATION
ILLINOIS INSTITUTE OF TECHNOLOGY UFRGS UFMT

fabio-ricardo@outlook.com renanghp@gmail.com hickernell@iit.edu