SciPy2015

Performance of PyCUDA (Python in GPU High Performance Computing)

Roberto Colistete Jr and Ramon Giostri Campos

UFES, Brazil

Universidade Federal do Espírito Santo

Introduction

High Performance Computing with GPU (Graphics Processing Unit) and CUDA (Compute Unified Device Architecture) can be made using Python via PyCUDA. Here we will compare PyCUDA versus C, CUDA, Python and Wolfram Mathematica in a calculation example from type Ia supernova cosmology.

Advances in computational techniques have allowed to explore issues related to science and technology that were previously unimaginable, one of them is the development of parallel computing using GPU.

We target the problem of calculating the distance modulus (μ_0) for type Ia supernovae (SNe Ia) for the case of flat standard model of cosmology[1], when it has $N = 3$ free parameters. Under these conditions the calculation is performed about 10^3 times for each supernova. But for supported models with N dimensions, it can vary between 10^N to 10^{2N}. Also worth remembering is that today the supernovae catalogs are of the order of a thousand supernovae and the estimate is that with new experiments it will have hundreds of thousands of SNe Ia in a few years[2]. Given the values shown we have a very intensive computational problem.

In this context, we perform the calculation of the distance modulus (μ_0) using C, C/CUDA, Python/Numpy, PyCUDA and Wolfram Mathematica (without and with CUDA), clearly showing the advantages and disadvantages of these approaches.

Physical Problem

A supernova is one of the possible final stages of a star, there are several types of supernovae[3] and we give particular attention to the type Ia supernovae (SNe Ia). The SNe Ia are standardizable candles and this feature was essential to prove that the universe expands with acceleration in 1998/1999[4,5], this pioneering work have won the Nobel prize in 2011.

Calculation of the theoretical distance modulus μ_0 is a central part of SNe Ia observational cosmology, as it is the amount statistically compared with the astronomical observable:

$$\mu_0(z,\Theta) = 25 + 5\,log_{10}\left(\frac{(1+z)c}{H_0}\int_0^z \frac{1}{h(z',\Theta)}dz'\right)$$

For flat cosmological models the changes are in the denominator of integrand, for non flat models, more significant changes are necessary[6].

Benefits of PyCUDA

- ▶ DP (Double Precision) as fast as C/CUDA
- ▶ Hundreds times faster than NumPy
- ▶ Faster than Wolfram Mathematica with CUDA, no issues with SP (Single Precision)
- ▶ Easier to write, less code than pure C/CUDA
- ▶ Allows metaprogramming
- ▶ Free and open source, while Wolfram Mathematica is not

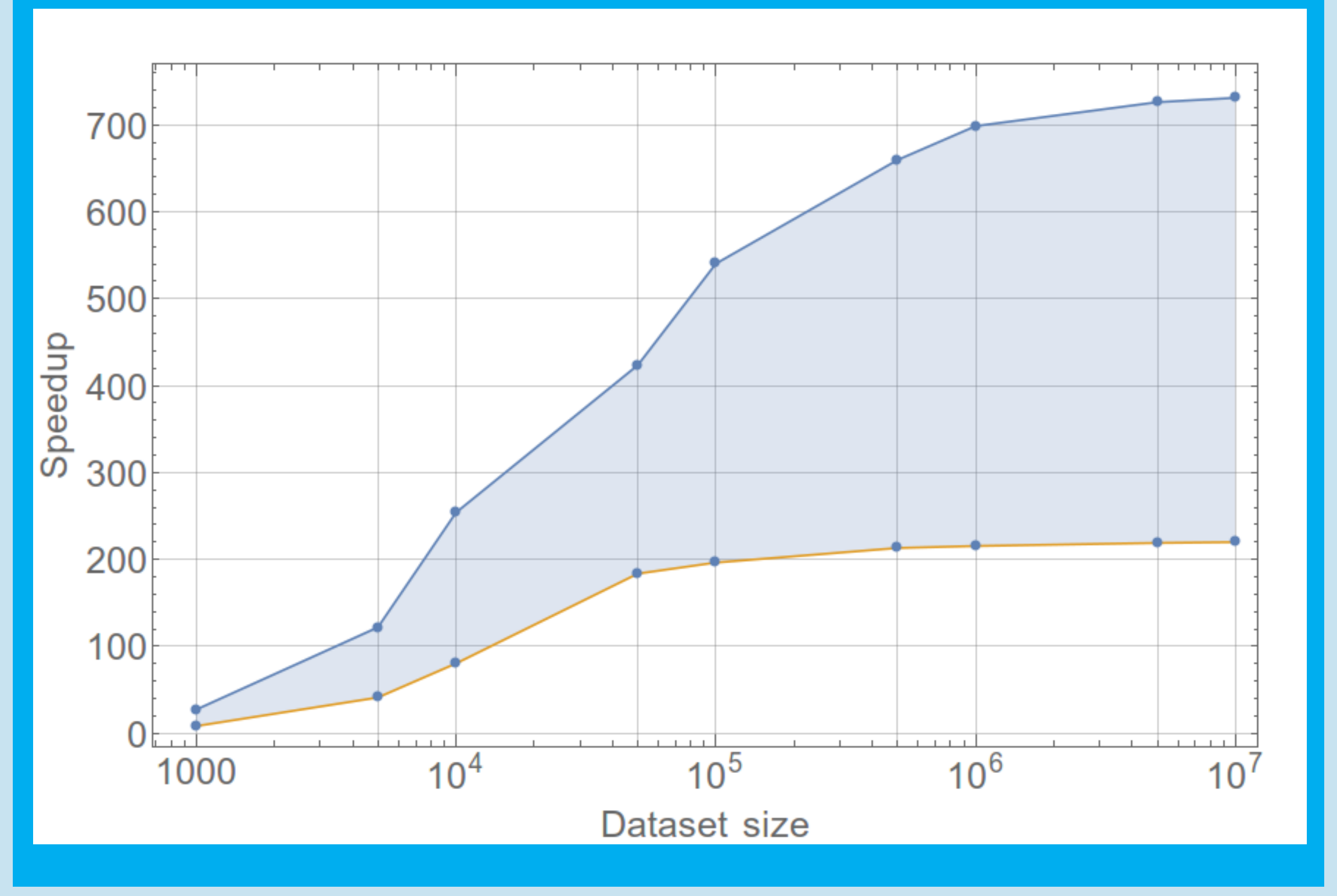
Drawbacks of PyCUDA

- ▶ SP (Single Precision) slower than C/CUDA
- ▶ Kernel is still written in C/CUDA
- ▶ Additional installation, sometimes with dependency issues

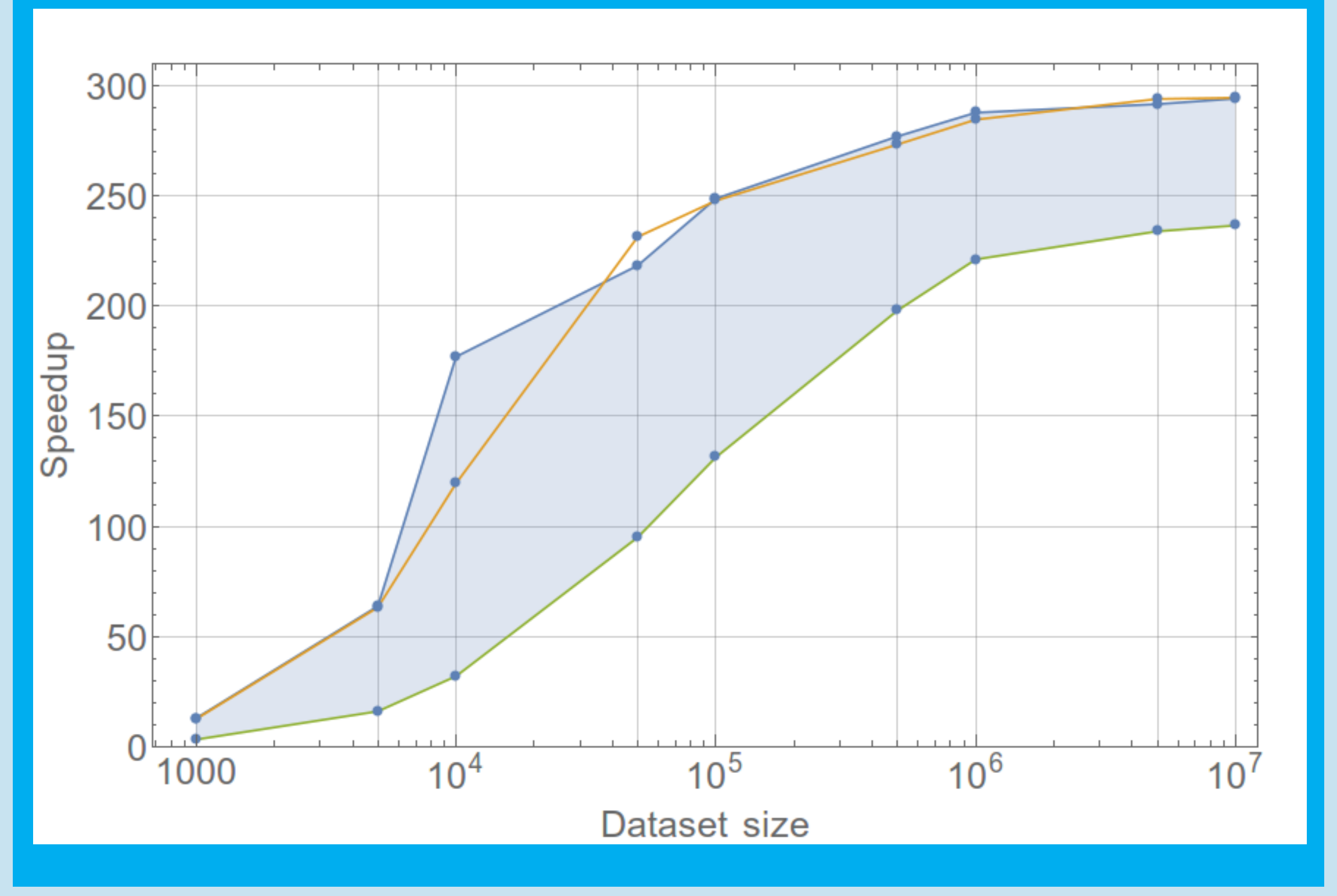
Performance of μ_0 Calculations in GPU and CPU

Made on a workstation with Intel Core i7 4770K (3.5-3.9 GHz, 8MB cache, 4 cores, 8 threads), 16GB RAM, GeForce GTX Titan (2,688 cores, 6GB GDDR5 RAM), Ubuntu 14.04.2 64 bits, Linux kernel 3.16.0, gcc 4.8.4, Python 2.7.6, NumPy 1.8.2, CUDA 7.0-28, PyCUDA v2015.1, Mathematica 10.1 :

Comparison of CUDA, PyCUDA in SP × C



Comparison of CUDA, PyCUDA, WMC in DP × C



Speedup with respect C/C++ (in 1 CPU core) vs. dataset size (number of supernovae) in SP (Single Precision) and DP (Double Precision). WMC : Wolfram Mathematica 10.1 using CUDA, which doesn't work in SP.

μ_0 calculation time in DP for 10^5 SNe Ia

Method	CUDA	PyCUDA	WMC	C/C++	Python	NumPy	WM
Total time (s)	0.002941	0.002952	0.006366	0.7327	46.13	1.499	2.634
Kernel time (s)	0.002576	0.002450	0.003632	—	—	—	—

μ_0 calculation time in DP (Double Precision) for 10^5 SNe Ia. WMC : Wolfram Mathematica 10.1 using CUDA. WM : Wolfram Mathematica 10.1 using CPU. Python : pure Python. NumPy : Python with NumPy.

Performance Analysis

In SP (Single Precision), C/CUDA is 1-2 times faster than PyCUDA because PyCUDA has overhead in dealing with SP data. While Wolfram Mathematica versions 9/10 don't calculate at all when running kernels in SP. In DP (Double Precision), PyCUDA is approximately as fast as C/CUDA. Both are faster than Wolfram Mathematica with CUDA, which is not efficient in CPU-GPU communication.

Conclusion

PyCUDA let us use CUDA (Compute Unified Device Architecture) from Python, making High Performance Computing (HPC) with GPU (Graphics Processing Unit) easier with respect to C/C++ with CUDA, while being robust and giving good performance.

Wolfram Mathematica, which has an interpreted language, can also call CUDA since version 8, so it (in version 10) was compared here.

Performance wise in SP (Single Precision), C/CUDA speedup w.r.t. C/C++ (CPU using 1 core) is up to 730 times. PyCUDA speedup w.r.t. C/C++ is up 220 times, w.r.t. Python/NumPy is up to 450 times. So C/CUDA is 1-2 times faster than PyCUDA in SP.

In DP (Double Precision), PyCUDA is approximately as fast as C/CUDA : C/CUDA speedup w.r.t. C/C++ is up to 294 times. PyCUDA speedup w.r.t. C/C++ is up 294 times, w.r.t. Python/NumPy is up to 601 times. Wolfram Mathematica with CUDA speedup w.r.t. C/C++ is up to 237, w.r.t. to Mathematica (using CPU) is up to 852 times.

So PyCUDA compares very well, being : as fast as C/CUDA in Double Precision (DP); faster and more robust than Wolfram Mathematica with CUDA; hundreds times faster than C/C++, pure Python, Python with NumPy, and Wolfram Mathematica calculated in CPU.

We conclude that PyCUDA has many advantages, so it is the logical step for Python developers coming from CPU calculations with Python/NumPy and going to use GPU programming.

Acknowledgments

R. Colistete Jr. thanks the SciPy 2015 Conference organization for the hospitality and financial support.

References

1 Wang et al., Dark Energy, Wiley, Weinheim, Germany, 2010.
2 Weinberg et al., Observational Probes of Cosmic Acceleration, 2012.
3 Filippenko, OPTICAL SPECTRA OF SUPERNOVAE, 1997.
4 Riess et al., Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant, 1998.
5 Permutter et al., Measurements of Ω and Λ from 42 HighRedshift Supernovae, 1999.
6 Persis et al. Type Ia supernovae, evolution, and the cosmological constant, 1999.