JULIA LANGUAGE DELIVERS PETASCALE HPC PERFORMANCE

November 28, 2017 Rob Farber

Written in the productivity language Julia, the Celeste project—which aims to catalogue all of the telescope data for the stars and galaxies in in the visible universe—demonstrated the first Julia application to exceed 1 PF/s of double-precision floating-point performance (specifically 1.54 PF/s).

The project took advantage of all 9300 Intel Xeon Phi Phase II nodes on the NERSC (National Energy Research Scientific Computing Center) Cori supercomputer.

Even in HPC terms, the Celeste project is big, as it created the first comprehensive catalog of visible objects in our universe by processing 178 terabytes of SDSS (Sloan Digital Sky Survey) data[1]. Remarkably, the combination of Cori supercomputer and Julia application was able to load and analyze the SDSS data set in only 15 minutes. Thus the Celeste team demonstrated that the Julia language can support both petascale compute and terascale big data analysis on a leadership HPC system plus scale to handle the seven petabytes of data expected to be produced by the Large Synoptic Survey Telescope (LSST) every year.[i]

"The Celeste project achieved a 100x increase over results previously reported in the literature" – Jeffrey Regier (postdoctoral fellow, UC Berkeley Department of Electrical Engineering and Computer Science and PI of the Celeste project)

The Celeste project statistician, Jeffrey Regier (postdoctoral fellow, UC Berkeley Department of Electrical Engineering and Computer Science), notes that the Julia-based Celeste application also set a new HPC and scientific milestone. Specifically, their Julia-based application performed an eight billion parameter Variational Bayesian Inference analysis that encompassed 188 million stars and galaxies. Thus, Regier reports that, "The Celeste project achieved a 100x increase over results previously reported in the literature". Further, the Celeste team notes that they created, "A statistically efficient scheme for decomposing astronomical optimization problems into sub-problems".[ii] This approximation runs in "nearly linear time in all relevant quantities: the number of light sources, the number of pixels, and the number of parameters".[iii] This method was revised to run in parallel and made use of the Julia threading model and runtime system to successfully utilize 650,000 Intel Xeon Phi 7200 cores and 1.3 million threads.[iv]

THE CELESTE PROJECT

The goal of the Celeste project, according to Prabhat (Group Leader, Data & Analytics Services, NERSC), "Is to take all telescope data that exists in the world and create one list of all stars and galaxies". The 178 TB SDSS run is a test case to demonstrate the Celeste project's highly scalable algorithm for constructing astronomical catalogues. Modern astronomical surveys produce vast amounts of complex data. It is expected that the Large Synoptic Survey Telescope (LSST) will have to process more than 30 TB of new data nightly. [v]

Both performance on parallel computational nodes and scaling across large numbers of nodes is critical to analyzing the expected tens to hundreds of petabytes of data the LSST will produce over its lifetime. The payback can be huge as it allows astronomers to better understand dark matter and energy, find hazardous asteroids, understand the formation and structure of the Milky Way,[vi] and even detect gravitational fields that are strong enough to affect light and hence act as an optical 'lens'.

A simple sketch of a gravitational lens is shown below.

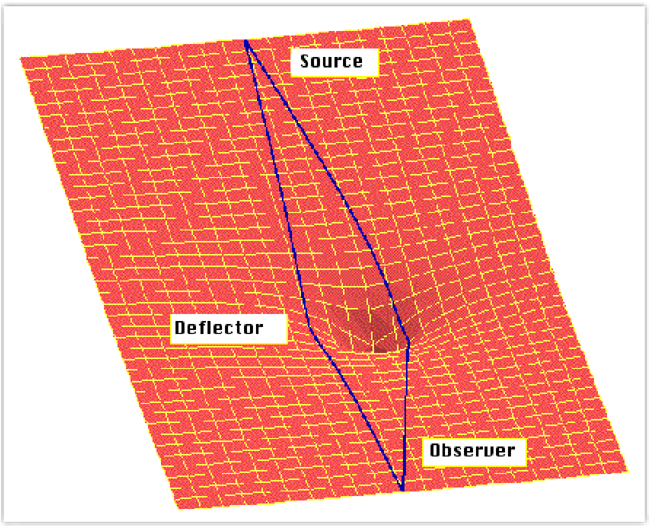

Figure 1: A sketch of a gravitational lens. The strong gravitational field is represented by the depression in the surface, while the lines show how light is affected by the field.
(http://w.astro.berkeley.edu/~jcohn/lens.html)

Since the gravitational lens effect is visible to the eye, the Hubble telescope has been able to detect some strong gravitational lensing examples such as the smiling face image shown below. [vii]

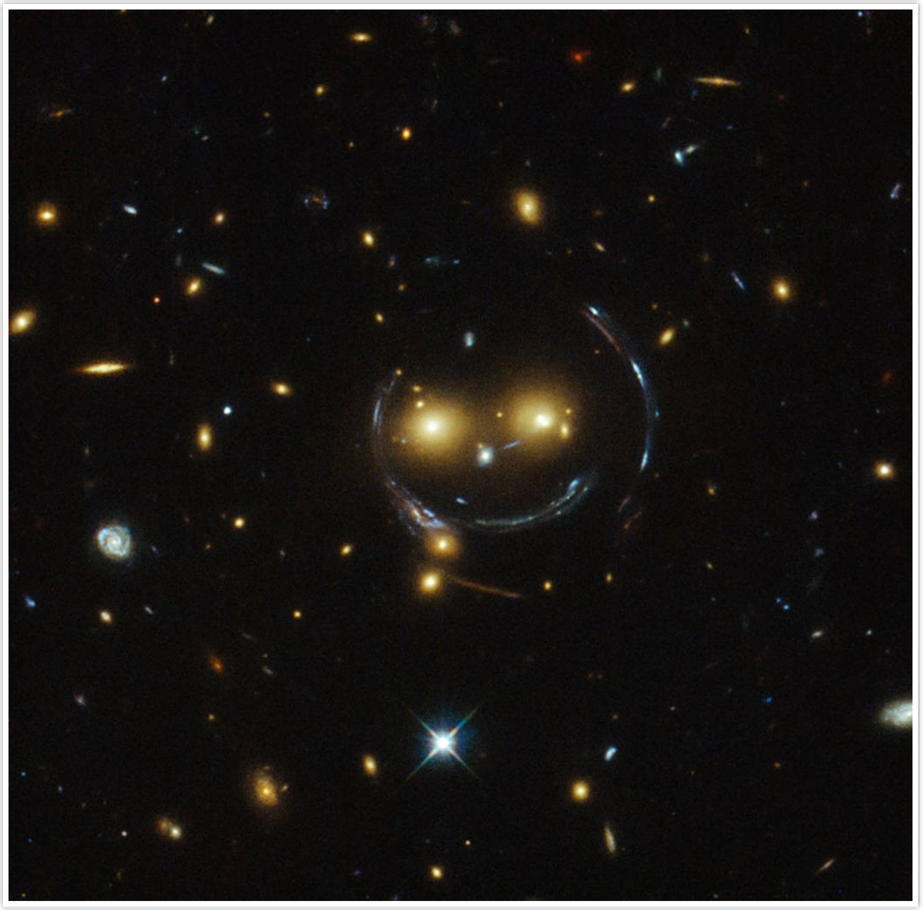

Figure 2: In the case of this "happy face", the two eyes are very bright galaxies and the misleading smile lines are actually arcs caused by an effect known as strong gravitational lensing. (Image courtesy NASA)

PROCESSING DATA IN CELESTE

The Celeste application starts off by preprocessing the data into roughly equivalent amounts of work. Of course this is difficult as our view of the light sources throughout the universe vary greatly across the sky. In particular, the Celeste load balancing preprocessing is complicated as data dependencies vary greatly depending on if contiguous regions of the sky have to be partitioned. For example, discontinuous regions generate fewer dependencies between computational tasks compared to other more contiguous regions.

However the load balancing works in practice. As seen below, the strong scaling results by the Celeste team shows that they achieve a 65% scaling efficiency from 2,000 to 4,000 Intel Xeon Phi nodes, and a 50% efficiency from 2,000 to 8,000 nodes. Load balancing becomes ever more important as the number of nodes increases as shown in the figure below. Note the log scale on the x-axis.

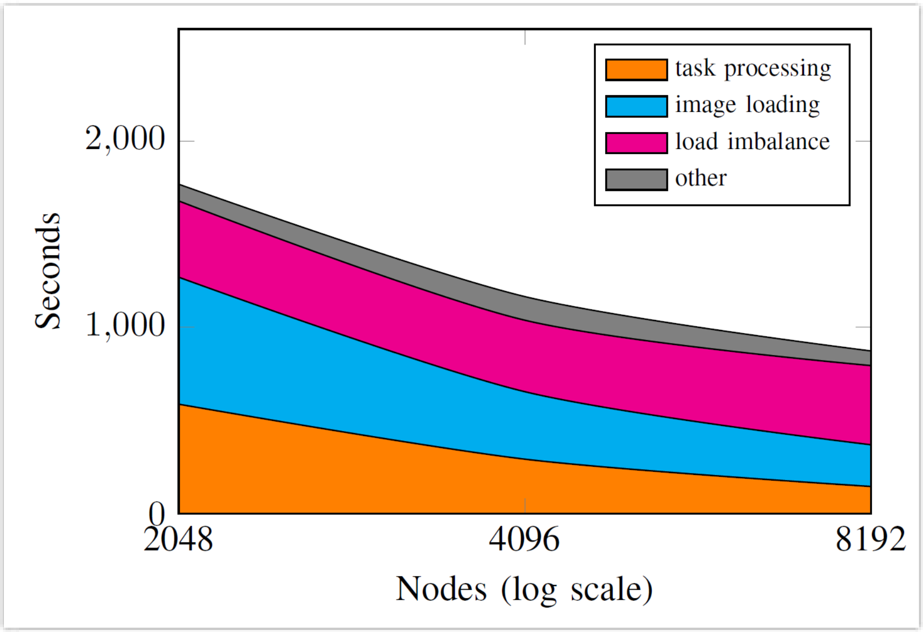

Figure 3: Strong scaling results

Even so, the team reports they found optimal parameters for 188,107,571 light sources in the SDSS data set.[viii] Putting the LSST data generation in perspective, in four nights the LSST will generate as much imaging data as SDSS did in over a decade.[ix] Thus, the performance and scalability of the Celeste software is an important demonstration that the LSST data can be put into form that is accessible to astronomers – namely an astronomical database. Once catalogued, the astronomical database can be managed by conventional relational database tools.

The performance and scalability of the Celeste software on the SDSS data represents an important demonstration of capability as the LSST will generate as much imaging data in four nights as SDSS generated in over a decade.

As to the accuracy of that database, the Celeste team utilized a region of the sky that has been imaged approximately 80 times in the SDSS database. The combined exposures are able to produce a very high signal-to-noise image from which ground truth parameters can be estimated and compared to the Celeste results. The team reports, "Improved accuracy on nearly all estimated parameters, usually by a substantial margin, improvements that are both statistically significant and of practical significance to astronomers".[x]

THE JULIA PRODUCTIVITY LANGUAGE

Julia is a relatively new productivity language compared to industry stalwarts like Python. While still new and relatively unknown by many in the HPC community, Julia is rapidly growing in popularity. Created in 2009, the Julia language has been downloaded over 1.2 million times as of September 2017.[xi] According to the Julia literature, it "enables rapid, interactive prototyping while achieving performance competitive with C, C++, and Fortran" and without forcing the programmer to use third-party "accelerators" (e.g. Numba, PyPy) or requiring that heavily used kernels be written it a low-level language. [xii] The ability to express all of the application in one language is one of the main reasons why Regier chose to use Julia for the Celeste project.

The timing for Julia appears to be good as productivity languages such as Python are now being viewed as acceptable for high performance HPC applications, due in particular due to recent 2016 Gordon Bell finalist applications such as PyFR, which demonstrated that Python application performance can compete head-to-head against native language applications written in C/C++ and Fortran on the world's largest supercomputers.

Unlike Python, the Celeste project team was able to attain their performance using only Julia source code and the Julia threading model. As a result, they were able to fully utilize the many-core Intel Xeon Phi processors. A per-thread performance breakdown is shown below.[xiii] Overall, the Julia code on each node delivered 82.3% of the total 1.54 PF/s of performance when operating on the 8-wide AVX512 vector registers.

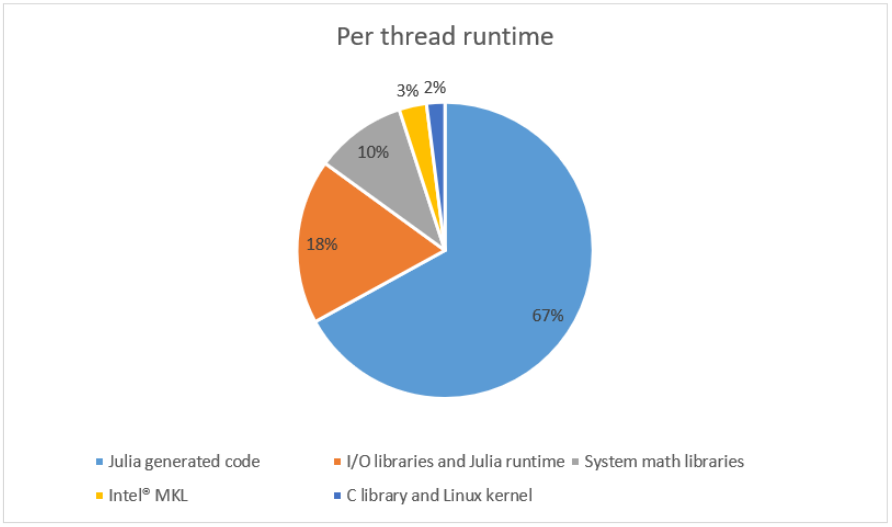

Figure 4: Per-thread runtime performance breakdown

Keno Fischer (CTO, Julia Computing) states that, "Scientists can now take the prototypes they have developed on their laptops and run them on the biggest supercomputers without having to switch languages or completely rewrite their code".[xiv]

"Scientists can now take the prototypes they have developed on their laptops and run them on the biggest supercomputers without having to switch languages or completely rewrite their code" – Keno Fischer (CTO, Julia Computing)

While Julia provides many performance oriented features, the Celeste project particularly used multi-dispatch and the Julia dynamic type system.

As is usual in the HPC world, data layout is key to performance as is the elimination of memory allocations. The flexibility of dynamic types meant that the Celeste developers were able to transition their data structure layout with a one line change from AoS (Array of Structures) to a SoA (Structure of Arrays) data layout to achieve a significant performance increase.

SUMMARY

The Celeste project provides a concrete demonstration that the Julia productivity language can deliver petascale performance on today's leadership class supercomputers. The proof point from the Celeste project is that the Julia code is efficient enough to deliver 1.54 PF/s of double-precision performance, even when working on large data sets, and when implementing a decently scalable (between 50% and 65%) algorithm.

This is antithetical to the current HPC view that high-level languages are only suitable for acting as glue-code that then calls high-performance C/C++ or Fortran code. Instead, Julia has proven to support many-core parallelism, efficient use of vector capabilities and high-performance network fabrics to deliver performant scalable applications. Remarkably, such scalable Julia applications can be written and tested on a laptop and then literally be moved onto an HPC system.

Rob Farber is a global technology consultant and author with an extensive background in HPC and in developing machine learning technology that he applies at national labs and commercial *organizations. Rob can be reached at* info@techenablement.com

[1] While the SDSS dataset is 55 TB, the team had to load and process parts of it multiple times.

[i] https://arxiv.org/ftp/cs/papers/0604/0604112.pdf

[ii] https://arxiv.org/pdf/1611.03404.pdf

[iii] ibid

[iv] https://www.youtube.com/watch?v=uecdcADM3hY

[v] Large Synoptic Survey Telescope Consortium, http://www.lsst.org/about.

[vi] https://www.lsst.org/about

[vii] https://www.nasa.gov/content/hubble-sees-a-smiling-lens

[viii] Cataloging the Visible Universe through Bayesian Inference at Petascale, to be published at IDF'17

[ix] ibid

[x] ibid

[xi] https://en.wikipedia.org/wiki/Julia_(programming_language)

[xii] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," SIAM Review, vol. 59, no. 1, pp. 65–98, 2017.

[xiii] Cataloging the Visible Universe through Bayesian Inference at Petascale, to be published at IDF'17

[xiv] https://juliacomputing.com/case-studies/celeste.html

SHARE THIS:

Reddit Facebook 33 LinkedIn 50 Twitter Google Email

SIMILAR VEIN

Dirt Simple HPC: Making the Case for Julia

NERSC Preps for Next Generation "Cori" Supercomputer

CPU Based Exascale Supercomputing Without Accelerators

Optimization Tests Confirm Knights Landing Performance Projections

First Burst Buffer Use at Scale Bolsters Application Performance

FPGAs, OpenHMC Push SKA HPC Processing Capabilities

Categories: Code, HPC

Tags: Cori, Julia

Intel Stacks Up Xeons Against AMD Epyc Systems

Debating The Role Of Commodity Chips In Exascale

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

PAGES

About
Contact
Contributors
Newsletter

RECENT POSTS

Cramming The Cosmos Into A Shared Memory Supercomputer
The Ecosystem Expands For AMD Epyc Servers
Debating The Role Of Commodity Chips In Exascale
Julia Language Delivers Petascale HPC Performance
Intel Stacks Up Xeons Against AMD Epyc Systems