Comparing systems for analyzing big neuroscience imaging data

Parmita Mehta, Sven Dorkenwald, Dongfang Zhao, Magda Balazinska, Alvin Cheung & Ariel Rokem

Computer Science and Engineering and the eScience Institute, University of Washington

Contact: arokem@uw.edu

Introduction

The analysis of image data has been a central part of neuroscience research since its very beginning, but with the more recent accelerated development of methods to image and record the brain at many different scales, large collections of digital image data have become available. The size, diversity and complexity of these data have thrust neuroscience researchers into the era of *big data*.

There are many software systems that can be harnessed for the analysis of large image data sets, but comparing these systems is a complex task that requires a high degree of expertise. This makes the selection of a system for specific analysis tasks, and for the creation of an analysis pipeline a daunting task.

We compared several big data software systems and benchmarked their performance in a state-of-the-art data processing and analysis pipeline with human neuroimaging data.

Apache Spark

(http://spark.apache.org/)

- Open-source parallel processing framework that enables users to run large-scale data analytics applications across clustered computers.
- Dataflow-based execution system that provides a functional, collection-oriented API.
- Parallel tasks are described as a directed acyclic graph (DAG) – resilience against transient failures by tracking computational lineage
- Optimizes for data locality when scheduling work.
- Programming interfaces in Scala, Java, and Python.

Myria

(http://myria.cs.washington.edu/)

- Distributed, "shared-nothing" big data management system from the University of Washington.
- Derives requirements from real users and complex workflows, especially in science.
- Provides a programming model that extends relational algebra with iteration that affords rich, iteration-aware optimization without sacrificing expressive power.
- Supports user defined functions (UDFs) in Python.
- Also provided as a cloud service that users can access directly through their browser or through a Jupyter notebook.

Rasdaman

(http://www.rasdaman.org/)

- An array database management system designed for storing and querying high-dimensional arrays (such as images).
- In addition to conventional SQL queries, it allows users to manipulate arrays directly in its own query language called Rasdaman Query Language (Rasql).
- Also supports other language bindings such as C++, Java, and Python (only available in the Enterprise version).
- Redirects the data to the underlying database systems (e.g., PostgreSQL, SQLite) or file systems.
- Users interact with Rasdaman mostly via queries, either through Rasql or programming interfaces; as a consequence, migrating an application to Rasdaman usually requires development effort to translate application logic to queries, and to glue different stages of the pipeline

TensorFlow

(https://www.tensorflow.org/)

- A
- B
- C

Materials and Methods

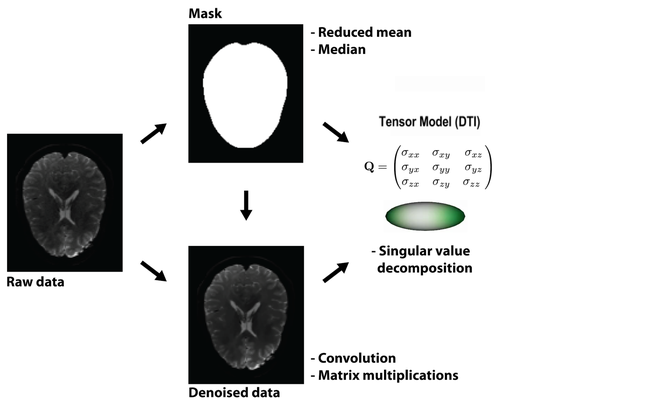
The human connectome can be assessed *in vivo* using MRI. Diffusion MRI (dMRI) is used to evaluate the microstructure of white matter, and the local orientation of nerve fibers in each voxel.

Data

Measurements were obtained from the Human Connectome Project (https://www.humanconnectome.org/). Measurements in 288 directions of diffusion were obtained at a 1.25 x 1.25 x 1.25 mm^3 resolution.

A typical pipeline of MRI analysis might include several steps. We focus here on:

- Segmentation: the part of the image containing the brain is identified using the median Otsu algorithm [3].
- Denoising: the image within this brain mask is denoise using the non-local means algorithm [2].
- Model fitting: a model is fit in every voxel of the measurement. We used the classic DTI model [1].



Data analysis was based on functions implemented in the open-source Dipy software (http://dipy.org).

Computational experiments

We used AWS r3.2xlarge instance type, optimized for memory-intensive applications (lower price per GiB of RAM): each instance has 8 vCPU, 61 GiB Memory and 160 GB SSD storage.

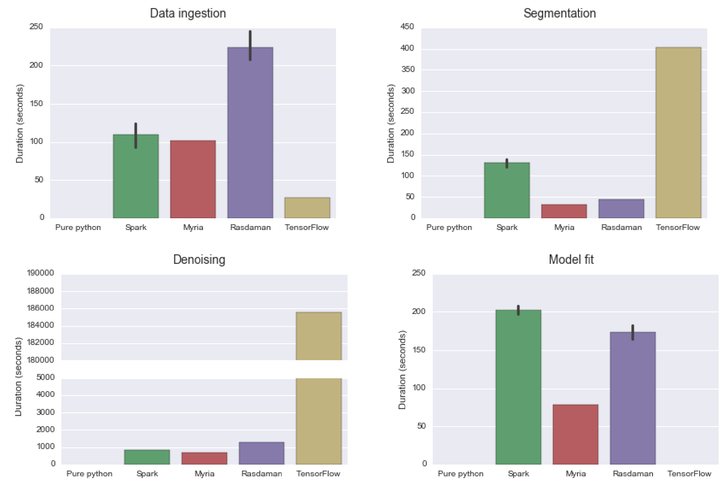
Pure Python: The reference implementation was run on a single instance.

Myria: Four compute nodes and one master node. Each compute node ran four workers making for a total of sixteen workers. For this image processing pipeline, Myria was setup with PostgreSQL as the per-node, local storage layer (default configuration).

Spark: Four compute nodes and one master node, Each compute node ran eight workers; spark default is one worker per CPU, for a total of thirty-two workers. We used spark-1.6.1-bin-hadoop1, in standalone mode with HDFS as the storage layer. We also used thunder 0.6.0 (http://thunder-project.org/) to set up spark.

Rasdaman: Four compute nodes were used and the default configuration: nine Rasdaman server processes on each node. We made reasonable effort to push certain application operations into Rasdamans query engine.

Results



Conclusions

- Myria performs faster than other systems for most tasks. This is probably due to reduced serialization cost, which other systems incur.
- Rasdaman requires significant rewriting of algorithms to be used.
- Spark error handling can be cryptic, because it uses the JVM.
- TensorFlow is still under heavy development, and might be much more performant in the future.

Future directions

Future developments include the development of user-defined aggregations in Myria.

Myria will also serve as the computational backend for web-based image processing pipelines we will provide as a service.

References

[1] P J Basser, J Mattiello, and D LeBihan. Estimation of the effective self-diffusion tensor from the NMR spin echo. *J. Magn. Reson. B*, 103(3):247–254, March 1994.

[2] P Coupe, P Yger, S Prima, P Hellier, C Kervrann, and C Barillot. An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images. *IEEE Trans. Med. Imaging*, 27(4):425–441, April 2008.

[3] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

Acknowledgements

This research was supported through a grant from the Gordon & Betty Moore Foundation and the Alfred P. Sloan Foundation to the University of Washington eScience Institute and through grants from the National Science Foundation