Volume Title
ASP Conference Series, Vol. **Volume Number**
Author
© **Copyright Year** Astronomical Society of the Pacific

Astronomical Image Processing with Hadoop

Keith Wiley[1], Andrew Connolly[1], Simon Krughoff[1], Jeff Gardner[2], Magdalena Balazinska[3], Bill Howe[3], YongChul Kwon[3], and Yingyi Bu[3]

[1]*University of Washington Department of Astronomy*

[2]*University of Washington Department of Physics*

[3]*University of Washington Department of Computer Science*

Abstract. In the coming decade astronomical surveys of the sky will generate tens of terabytes of images and detect hundreds of millions of sources every night. With a requirement that these images be analyzed in real time to identify moving sources such as potentially hazardous asteroids or transient objects such as supernovae, these data streams present many computational challenges. In the commercial world, new techniques that utilize cloud computing have been developed to handle massive data streams. In this paper we describe how cloud computing, and in particular the map-reduce paradigm, can be used in astronomical data processing. We will focus on our experience implementing a scalable image-processing pipeline for the SDSS database using Hadoop (http://hadoop.apache.org/). This multi-terabyte imaging dataset approximates future surveys such as those which will be conducted with the LSST. Our pipeline performs image coaddition in which multiple partially overlapping images are registered, integrated and stitched into a single overarching image. We will first present our initial implementation, then describe several critical optimizations that have enabled us to achieve high performance, and finally describe how we are incorporating a large in-house existing image processing library into our Hadoop system. The optimizations involve prefiltering of the input to remove irrelevant images from consideration, grouping individual FITS files into larger, more efficient indexed files, and a hybrid system in which a relational database is used to determine the input images relevant to the task. The incorporation of an existing image processing library, written in C++, presented difficult challenges since Hadoop is programmed primarily in Java. We will describe how we achieved this integration and the sophisticated image processing routines that were made feasible as a result. We will end by briefly describing the longer term goals of our work, namely detection and classification of transient objects and automated object classification.

1. Introduction

Future astronomical surveys will generate data in quantities which cannot be processed by single computers. One potential solution to this problem is to harness large clusters of computers by using *cloud computing*. In this paper we describe the development of a cloud computing based image coaddition system using the Hadoop MapReduce cluster framework. We describe our system, then show an example of a coadded mosaic and analyze its improved detection threshold.

2. Experimental Setup

This research was performed on the *CluE* cluster (see Acknowledgements). At the time, the cluster had 700 nodes, each with 4x2.8GHz cores, 8GB ram, and 800GB storage for a total cluster storage capacity of 560TBs.

We chose as our dataset *Sloan Digital Sky Survey* (SDSS) Stripe 82 (Abazajian 2009; SDSS). The SDSS camera has 30 CCDs (2048x1489 pixels, 6MB FITS) in 5 bandpass filters which capture 6 parallel strips of sky at a time. Stripe 82 is a 30TB, 4 million image dataset gathered near the equatorial plane ($+/-1.25^\circ$ declination) with an average coverage of ~75. In theory, coaddition at such coverage should yield a SNR improvement of ~8.7x or an improved limiting magnitude of ~2.3 mags.

Our research has focused on the development of a massively parallel *image coaddition* system. Given the variety of uses of the term, we define image coaddition as the process of background-subtracting, warping, PSF-matching, registering, and per-pixel averaging a set of partially overlapping images into a final image called a *mosaic*. Much of this process can be trivially parallelized since many of the steps are performed on the input images prior to their incorporation into the mosaic.

3. Massively Parallel Data Processing

In recent years, a new approach to massively parallel data processing called *cloud computing* has gained popularity. A cloud consists of a large network (1000s) of relatively cheap commodity computers which is then made accessible over the internet. This economical construction and internet-based access permit clouds to be offered as a generic service wherein users program and submit their own jobs remotely and as third party customers. One popular example of such a general-purpose cloud is Amazon's EC2.

MapReduce is a framework for designing cloud-computing programs (Dean & Ghemawat 2004) which encapsulates the cluster-related aspects of parallel computing, namely intra-network communication, resiliency to task/node failure, etc.. This design alleviates much of the complexity that parallel programing would otherwise impose. A MapReduce program is performed in two sequential stages. The *mapper* stage performs a parallel computation on the input data. The results are distributed to the *reducer* stage which conglomerates the mapper outputs into the final job output. *Hadoop* is an open-source implementation of MapReduce (Apache; White 2009) which has quickly grown in popularity in large part due to is relatively easy learning curve and the large and active online community of support.

Hadoop is programmed in Java. However, our research group has already developed a sophisticated C++ image-processing library. In order to access this library from Hadoop, we use the *Java Native Interface* (JNI). Using JNI, our Java-based mapper and reducer serve primarily to interface with the Hadoop framework and distributed file system, but delegate most of the computational demands (image coaddition) to C++.

While many form of data-processing require processing the entire input dataset, image coaddition does not. A *query* (a bounds on the sky within which to generate a mosaic) only covers the small subset of the input images. Therefore, we use a front-end relational database containing metadata about the input images, including their sky bounds. Our Hadoop job first performs a SQL query to retrieve the filenames of the images which are relevant to the coaddition process. Those filenames then represent the input to our MapReduce image coaddition system.

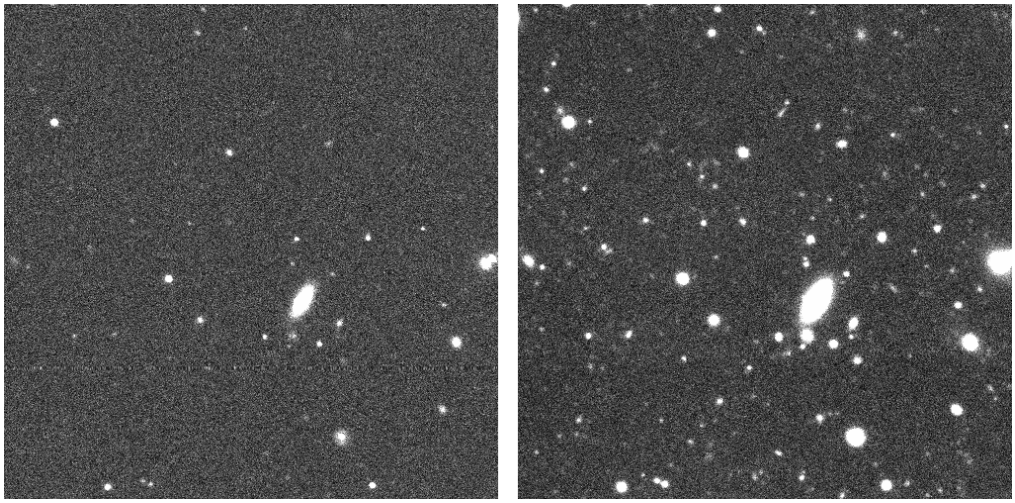

Figure 1. This figure shows a single r-band frame on the left and a mosaic of 96 frames on the right (with a max coverage of ~75). The mosaic reveals more faint sources as a result of coaddition.

4. Image Coaddition in Hadoop

In order to adapt image coaddition to Hadoop, we perform the initial processing on each input image in a highly parallelized mapper stage. This processing includes background-subtraction, warping to the final coordinate system, and PSF-matching. The results are then sent to a single reducer which performs the per-pixel average and generates the final mosaic. The serialized nature of the reducer is acceptable since the overall computational demands are dominated by the steps performed in the mappers.

5. Results

Fig. 1 shows an example of image coaddition. A single r-band frame is shown on the left and a mosaic of 96 frames is shown on the right. We would expect the point source detection threshold for such a mosaic to be improved by ~2 mags over the single frame and in Fig. 2 we observe that the expected improvement was achieved. On the CluE cluster, our system was able to generate this mosaic in ~34 minutes. However, many factors can influence this result: Hadoop restarts failed tasks, we did not enable compiler optimizations, and our image-processing routines are still under development. We estimate that when properly configured, this job time may drop well below 13 minutes, which corresponds to a per-image (mapper) processing time of <8 minutes.

6. Future Work

In the near future we hope to improve our coaddition system in many ways. We would like to improve the overall algorithm by parallelizing the reducer, implementing better memory management, and continuing to improve our image-processing routines.

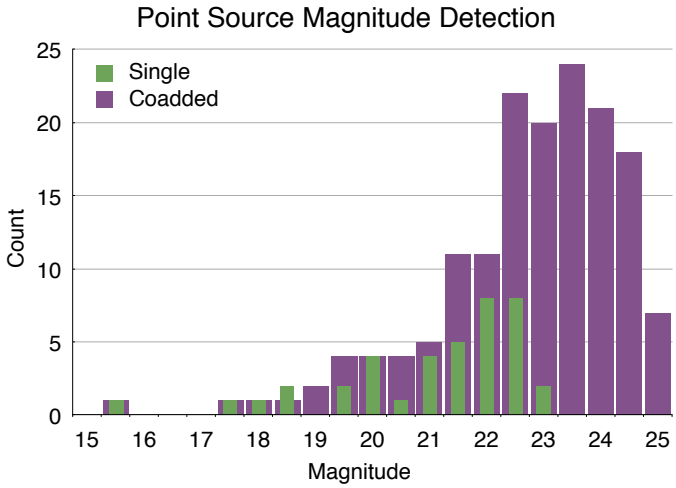

Figure 2. This plot shows the point source magnitude detections achieved by the single image and the mosaic shown in Fig. 1. We observe that the mosaic's point source detection threshold is improved by ~2 mags, as expected.

We intend to extend the query description to include time-bounded queries and to ultimately perform automated object detection and classification on the mosaics. Finally, we intend to wrap our system in more accessible scripting languages and perhaps to offer it through a web-based graphic user interface (GUI).

7. Conclusions

This research demonstrates a massively-parallel cloud-computing based image coaddition system. We described Hadoop and our image coaddition system within Hadoop. We then showed an example mosaic generated from SDSS Stripe 82 and demonstrated that it achieved the expected improvement in point source detection threshold.

Acknowledgments. This work is funded by the NSF Cluster Exploratory (CluE) grant (IIS-0844580) and NASA grant 08-AISR08-0081. The cluster is maintained by IBM and Google. We thank them for their continued support. We further wish to thank both the LSST group in the astronomy department and the database research group in the computer science department at the University of Washington.

References

Abazajian, et. al. 2009, The Astrophysical Journal Supplement, Vol. 182, pp. 543-558.
Apache, Apache Hadoop. http://hadoop.apache.org/, 2007.
Dean, J., & Ghemawat, S. 2004, in Sixth Symposium on Operating System Design and Implementation (San Francisco, CA, USA), OSDI'04.
SDSS, SDSS Stripe 82. http://www.sdss.org/legacy/stripe82.html, http://www.sdss.org/dr7/coverage/sndr7.html, 2007.
White, T., Hadoop The Definitive Guide (1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media Inc.), 1st ed., 2009.