Towards a Provenance Framework for Sub-image Processing for Astronomical Data

Johnson Mwebaze
Kapteyn Astronomical Institute
The Netherlands
jmwebaze@cit.ac.ug

John McFarland
Kapteyn Astronomical Institute
The Netherlands
mcfarland@astro.rug.nl

Danny Booxhorn
Kapteyn Astronomical Institute
The Netherlands
danny@astro.rug.nl

Edwin Valentijn
Kapteyn Astronomical Institute
The Netherlands
valentine@astro.rug.nl

ABSTRACT

While there has been advances in observational equipment that generate huge high quality images, the processing of these images remains a major bottleneck. We show that provenance data collected during the processing of data can be reused to perform selective processing of data and support network collaboration without clogging distribution networks. We introduce the idea of sub-image processing (SIMP) in the context of processing a subset of pixels of an image and the use of provenance data to assemble pipelines and to select processing metadata for SIMP. We describe an implementation of SIMP in Astro-WISE[1].

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Distributed Systems; H.4 [**Information Systems Applications**]: Workflow Management; J.2 [**Physcial Sciences and Engineering**]: Astronomy

General Terms

Design, Experimentation

1. INTRODUCTION

Much of modern and scientific research such as virtual astronomical observations, bioinformatics and high-energy physics involves the accumulation of huge amount of digitized data and requires global participation for processing and visualization. Such observations provide image data with catalogues of millions of objects, each object with hundreds of associated parameters. Raw data is often collected from shared observational instruments, while end-users of this data perform analysis and/or visualization at local stations in their research centers. Because of the distributed nature of data, people and the processes; processing of these huge datasets is becoming problematic [9].

End-users can no longer visualize and manipulate these large data on their local workstations and over networks in real time because of the high pixel images coupled with the narrow network bandwidth. The processing is also increasing in complexity requiring laboriously sophisticated techniques and high end distributed computing resources. Such processing requires trying different datasets and processing techniques, tweaking parameters, verifying data quality, repeating this data derivation and customization of the results. Although most of the processes are run in a separate dataflow, they have a certain amount of overlap. (e.g. share some input and sometimes intermediate data). Its clearly wasteful to run the same processes repeatedly without reference to (or the use of) already existing processed data.

Astronomical systems have been built to work with and process data based on the telescope's detector or chip (full-image) [2, 6]. All metadata and processing parameters are based on an instrument with fixed detector properties (e.g., image size, calibration frames, overscan regions, etc.). An image from an observation may have a catalogue of millions of sources, each source with hundreds of associated parameters. By source we mean a celestial objects such as stars, planets, comets, nebulae, star clusters and galaxies. However, most often users are interested in a source that lies on a few pixels of an image. The current approach allows the processing of a full-image even if the source of interest exists on a few pixels of an image. Accordingly, out of millions of images in a survey, it is nearly impossible and wasteful to process the whole data volume. Instead of processing the whole dataset, a user should only select, retrieve and process only relevant pixels on an image where the source exists. These pixels are extracted and processed as a sub-image.

Based on these observations, we motivate the need to record fine-grained provenance and also the need to use provenance to enable sub-image processing (SIMP). Data provenance in scientific experiments has been used to understand results, the relationships between data products and the programs that were used during processing by examining the sequence of steps that led to a result [3, 4]. By using provenance, we can simply the processing of data by examining previous reruns and relating them to current runs [8].

[1] www.astro-wise.org

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

In this context, since all pipelines for astronomical image processing have been written for full-images, and all metadata and processing parameters are based on an instrument (or full-image), we use provenance data to match and retrieve existing pre-processed information in the system from which we build pipelines and select input data for the SIMP. In the same line, we enable operations that are very difficult (or impossible) to execute at sub-image level.

This paper presents our approach to providing these fundamental services in a distributed computing environment. This papers focus on the use of provenance to support SIMP. We specifically show how we trace fine-grained provenance (at pixel level) and how we use this provenance data to perform SIMP. With SIMP the user controls the downloading process by selecting, extracting and processing pixels with their objects of interest. This schemes drastically reduces the amount of data transferred over wide-area-networks.

To the best of our knowledge, this is the first work that leverages provenance to support SIMP. The rest of the paper is organized as follows; In Section 2, we briefly describe our provenance model in Astro-WISE. We present the SIMP framework in Section 3 and we review related work in Section 5. We conclude in Section 6 where we outline directions for future work.

2. LINEAGE IN Astro-WISE

In our previous work [7], we proposed a model that uniformly captures provenance during the course of data processing. From that model, we explicitly assume that every output depends on every input and parameters passed to the function. Therefore, such proveance accounts for an output product produced during the course of a dataflow execution by displaying a connected graph of input, intermediate, output data and also parameters and attributes used during the processing. This level of granularity is in-sufficient for sub-image processing. We need to trace fine-grained lineage to include transformations at pixel/byte level. The challenge is to determine which pixels of which input images were used in the construction of the composite pixels in the output image.

2.1 Pixel Lineage

We define $F : \mathcal{V} \rightarrow \mathcal{V}$ as a function on the space of astronomical image processing, and $\delta : \mathcal{V} \times \mathcal{V} \rightarrow F$ as a function that takes an input pixels p_a and outputs p_b, then for brevity let $\delta_{ab}(p_a) = p_b$. If there exists δ_{ba} such that $\delta_{ba}\delta_{ab} = e$, where e is an identity matrix, then there exists a 1-1 mapping between pixels of the iinput image to pixels to an output image. Then it is possible to a find a mapping between pixels of related images or connect all pixels from raw images intermediate to the final objects. However, we can achieve this if our sequence of operations consists of invertible atomic operations. Specifically, suppose $\delta_{ab} = f_n \circ \cdots \circ f_1$ then each f_i must have a well-defined inverse. For example, if f_i is the operation of applying a distortion correction then, f_i^{-1} is the operation of removing the distortion correction.

Because of the nature of the source data as well as the nature of the image processing and regridding algorithms, invertibility is a very hard problem or impossible. For such non-reversible transformation, pixel lineage is computed using numerical techniques. Thus pixel lineage becomes probabilistic and different precision levels can be achieved . We have implemented and tested such numerical techniques for

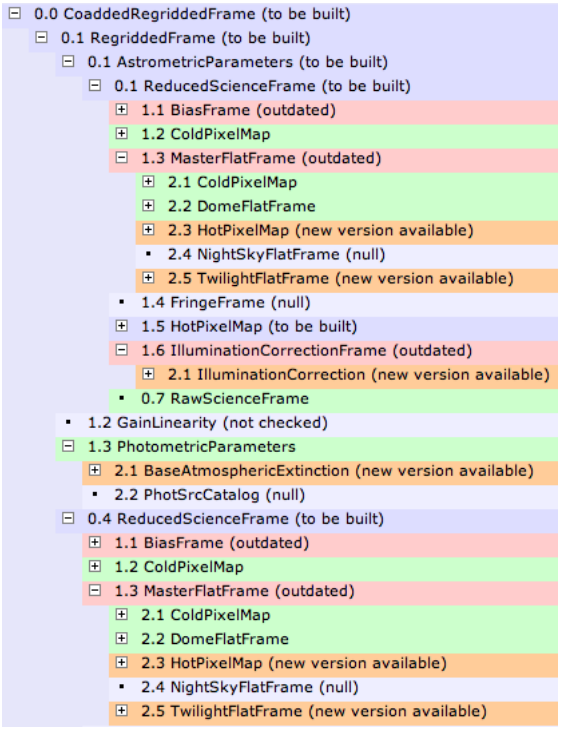

Figure 1: A tree view is given of the dependencies This tree view gives an overview of the target dependencies. Green dependencies are up-to-date, red dependencies are out-of-date and for orange dependencies indicate a newer version exists.

.

tracing pixel lineage. Early results are promising. A positional accuracy of approximately 5 orders of magnitude can be achieved. The algorithm can be tuned to achieve any precision, but its a trade-off between performance and accuracy. With this framework, we can trace and link all pixels that have been processed in the system.

3. SUB-IMAGE PROCESSING

The ability to store and mine provenance data is required to enable SIMP. This is because pipelines have been written for full-images and designed for instruments with fixed detector properties. No metadata and processing parameters currently exist for SIMP. These parameters have to be copied from provenance data, modified and new pipelines for SIMP assembled. SIMP involves 4 basic steps;

3.0.1 Pipeline Matching

The basic idea behind our algorithm is to search for a graph representation of the dataflow that was used in earlier runs to process target. The starting point is a target and a set of preconditions. E.g. consider the basic query *find the derivation path for a source 'S'* The answer to this query may consists of the several paths rooted at the same Raw-Data. It is possible that the same object can be computed using different variations of method or parameters. To select the required pipeline, we add more constraints to the query. Using these constraints, the system matches and selects the pipeline from all candidate pipelines by pairing nodes and computing similarity between two adjacent nodes. From the selected pipeline, the system builds a directed graph representing the data dependencies with nodes representing ob-

jects and edges representing all dependencies attached to object (Refer to Fig. 1). The graph begins with the topmost node, which is the target to be made. New edges are added starting at this trigger and expanding outward, using the dependency logic derived from provenance data. The dependency graph is built and checked recursively till the last dependency (in this case raw data from the telescope).

3.0.2 Parameter and Attribute Selection

In this step we retrieve input data, processing parameters and intermediate data products from the provenance store based on the objects in the graph generated in section 3.0.1. Each node in the graph is associated to an object. Each object is identified with a unique ID which is used when searching the database for provenance data related to a particular object. Part of the provenance includes input and output data, processing parameters and attributes, methods/modules that were used to process the object. If part of provenance is an image, a cutout is made of the pixels of interest from this image and used as input to the module. The pixels extracted as a cutout are determined through pixel lineage. We also identified basic operations that are useful for common querying tasks over provenance store that simplify the query syntax. Some examples are listed below;

- `get_inverse_properties(obj)` returns all objects that used *obj*
- `get_dependencies(obj)` returns all attributes of *obj*
- `get_onthefly_dependencies(obj)` returns all dependencies of *obj*
- `info(obj)` displays a lineage tree for *obj*
- `retrieve()` makes a cutout from an image of a size specified by the sub-image attribute

3.0.3 Pipeline Changes

In this frame-work there are two types of changes. Firstly, pipelines must be modified to process sub-images and secondly a user might need to change processing parameters or to switch to a different algorithm to improve a given result while carrying out a detailed analysis or a computation to a specific region on an image. In any case we build on already existing data from previous runs to enable such changes. All changes required for this framework are included in the system, as a new module, attribute or parameter without changing the overall data model. These are automatically loaded whenever a user requests to process a sub-image. However, the complete account on what changes are made to the pipelines is beyond the scope of this paper.

3.0.4 Pipeline Building

Using the dependency graph generated in section 3.0.1, we apply all changes as identified in section 3.0.3 and attributes and parameters from section 3.0.2, and we build the pipeline for SIMP. We apply all the changes (or additions) and analyze the dependency graph. Only the modules, intermediate data or a dependency affected by the change will be processed. If new versions exist of the classes that created the dependencies, then these modules will also be rerun using the new versions of the classes (refer to the caption of Fig 1).

After assembling the pipeline and collecting all necessary input data, processing of sub-image is then done. Source extraction is then run on the sub-image resulting in a new catalog of sky positions, and/or any other user specific processing is done on the sources extracted.

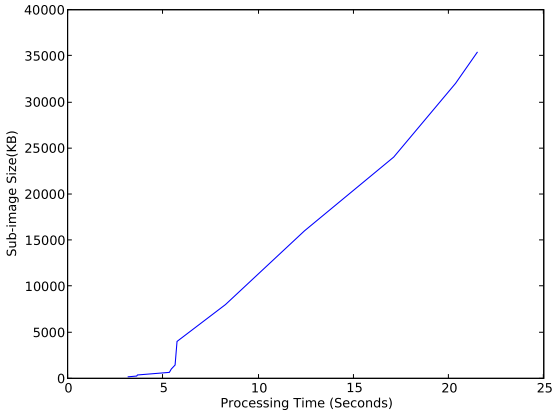

Figure 2: A plot showing the time it takes to process sub-images of different sizes

4. USE CASE: FINDING DROPOUTS

Initially images are reduced, processed and source extraction carried out. After source extraction, it happens that sources were detected in some frames(images) derived from data taken in certain filters, we call 'master filters', and not detected in other frames derived from data taken in some other filters, we call 'drop-out filters'. The task is to find drop-out candidates, that is, those sources that were detected in frames observed using the master filters, but not appearing in the frames observed using dropout filters. Such sources might be extremely high-redshift quasars. Precise fluxes need to be determined in the master filters and flux upper-limits in the drop-out filters. After identifying the drop-out candidates, all the images need to be reprocessed using new source parameters and new detection thresholds. Rather than re-processing full-images, we extract and process only the pixels surrounding the dropout candidates. By doing this we save on computation and data transfer time. Below, we present two tests to to demonstrate the power of SIMP.

Processing Time: We create the experiment as explained above. We process a full image using new source parameters and new detection thresholds. We repeat the same test using sub-images of various sizes. The results of this processing are shown in Fig.2. The time it took to process each different sized cutout was recorded. Notice from Fig.2 that the processing time increases as the size of the cutout increases. This therefore shows that there is significant saving in computation time (and resources) while processing sub-images compared to processing full-images.

File Retrievals; One common characteristic of all dataflow programming frameworks is the requirement of locally staged data for processing. In this test, we estimate how much time is required to download 24 sub-images of 544k from a data server to local PC compared to downloading the 24 full-images of 384MB. we run this test over two links of different capacities. The table below shows the results;

Link Size	No of Images	Image Size(KB)	Time (s)	Transfer Rate
1Gbit/s	24	544	0.4	1.22 MB/s
		544	0.4	1.21 MB/s
		544	0. 4	1.22 MB/s
		544	0.7	737 KB/s
		393216	22	17.7 MB/s
		393216	21	17.9 MB/s
		393216	23	17.1 MB/s
		393216	23	16.6 MB/s
6Mbit/s	24	544	1.3	423 KB/s
		393216	676	5822 KB/s

The time in seconds is the total time that the actual transfer took, from the first byte to the last byte of each file. We notice a significant difference when transferring sub-images compared to full-images on a 1Gbit/s link. As you notice downloading the sub-images on a 6Mbit/s takes almost the same amount required as on a 1Gbit/s. However, the results for transfer full-images 6Mbit/s are shocking and might render the system unusable.

5. RELATED WORK

We could not find anywhere in literature where SIMP has been done and therefore we have no comparative analysis to evaluate if there could be advantages or disadvantages of our approach compared to any other approach. However, distributed data sharing systems e.g. grid systems [10] have been developed to address performance and dataset concerns. Such systems like [2], provide a scalable infrastructure for running image pipelines in a distributed way. However even with such grid system, transferring of the data to the processing node still suffers from delays due to congestion on WAN links.

Provenance-aware scientific workflow systems [5] have also been considered as the paradigm for representing and managing complex distributed scientific computations. Systems such as those surveyed in [4] have enabled scientists to carry out complex scientific computations while capturing provenance. Despite these developments, little or no support exists in current systems to recored provenance data and at pixel level and most models do not allow end-users to use lineage data in scientifically meaningful way in particular to improve scientific processes. Our provenance model introduced in this paper does trace lineage at the finest detail (e.g., a pixel transformation process). This lineage captured is then used for various scientific processes but most specifically as introduced in this paper, this lineage at pixel level has been used for SIMP.

The use of provenance we describe in this paper is analogous to how other authors have used provenance to solve some use-cases. For example in Kepler [1] provenance has been used to enable smart 'reruns'. Kepler takes data dependencies into account and only execute those parts of the workflow affected by the parameter change. In [8] provenance has been used for interactive design of workflows.

6. DISCUSSION AND CONCLUSIONS

We have described a framework that leverages provenance to aid in selective retrieval and processing of data. All of the discussed functionality has been implemented, however due to the page size limitation, most of the interesting features have not been mentioned. We plan to continue working on the data models and make it available as a web-service in the near future. This paper does not focus on the changes made to the modules/pipelines process sub-images, but rather how provenance can be used to support SIMP.

We can safely say that we have accomplished our design goals of supporting network collaboration because users at remote research centers could comfortably run and process data, without limitations of huge data transfers and limitations of resources on local clients. However our approach is not foolproof, and there are cases where it may fail to produce the results a user expects. For example, if a user applies the methodology to a processing that involves neighboring pixels to determine a result of a pixel, the pipelines is likely to fail. However, when such a processing fails the user can initially process the full image and extract all parameters/needed to aid in processing of other sub-images derived from the same image. The effect shall be slower performance for the start, which improves significantly while processing other sub-images.

Although we reduced the domain from a full frame to a sub-image, we intend to further transform scientific systems to process pixels rather than images. We are currently investigating how we can use databases to aid in such processing.

7. REFERENCES

[1] I. Altintas, B. Ludaescher, S. Klasky, and M. A. Vouk. Introduction to scientific workflow management and the kepler system. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 205, New York, NY, USA, 2006. ACM.

[2] K. G. Begeman, A. N. Belikov, D. R. Boxhoorn, F. Dijkstra, E. A. Valentijn, W.-J. Vriend, and Z. Zhao. Merging grid technologies. *Journal of Grid Computing*, 8:199– 221, 2010.

[3] J. Freire, D. Koop, and L. Moreau, editors. *Provenance and Annotation of Data and Processes: Second International Provenance and Annotation Workshop*. Springer-Verlag, Berlin, Heidelberg, 2008.

[4] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21, 2008.

[5] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, 2007.

[6] P. Greenfield. Reaching for the stars with python. *Computing in Science and Engg.*, 9(3):38–40, 2007.

[7] J. Mwebaze, D. Boxhoorn, and E. Valentijn. Astro-wise: Tracing and using lineage for scientific data processing. *Network-Based Information Systems, International Conference on*, pages 475–480, 2009.

[8] C. Scheidegger, H. Vo, D. Koop, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1560 –1567, nov.-dec. 2007.

[9] A. S. Szalay. The sloan digital sky survey and beyond. *SIGMOD Rec.*, 37(2):61–66, 2008.

[10] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3):44–49, 2005.