A Demonstration of Iterative Parallel Array Processing in Support of Telescope Image Analysis

Matthew Moyers[1], Emad Soroush[1], Spencer C Wallace[2], Simon Krughoff[1], Jake Vanderplas[1], Magdalena Balazinska[1], and Andrew Connolly[1]
[1]University of Washington [2]University of Arizona
{mmoyers, soroush, jakevdp, magda}@cs.washington.edu
{krughoff, ajc}@astro.washington.edu
spencerw@email.arizona.edu

Abstract

In this demonstration, we present AscotDB, a new tool for the analysis of telescope image data. AscotDB results from the integration of ASCOT, a Web-based tool for the collaborative analysis of telescope images and their metadata, and SciDB, a parallel array processing engine. We demonstrate the novel data exploration supported by this integrated tool on a 1 TB dataset comprising scientifically accurate, simulated telescope images. We also demonstrate novel iterative-processing features that we added to SciDB in order to support this use-case.

1. INTRODUCTION

Scientists today are able to generate data at an unprecedented scale and rate [4, 5]. Because scientific data often takes the form of multidimensional arrays (*e.g.*, 2D images or 3D environment simulations), many engines are being built to support this model natively [2, 7, 8]. SciDB [7] is one such engine. To handle today's large-scale datasets, arrays must also be partitioned and processed in a shared-nothing cluster [7]. In array-based systems, structural information is associated with each cell through its dimension values. Additionally, array dimensions provide a natural index for the data, which improves query performance.

Many data analysis tasks today require iterative processing [3]: machine learning, model fitting, pattern discovery, flow simulations, cluster extraction, and more. This need extends to analysis executed on multidimensionnal scientific arrays as we show in Section 3. While it is possible to perform iterative computations by iteratively invoking array queries from a script, this approach is highly inefficient. Instead, large-scale data management systems such as SciDB should support iterative computations as first-class citizens. We demonstrate new SciDB features for efficient iterative processing in the context of real data analysis from the astronomy domain.

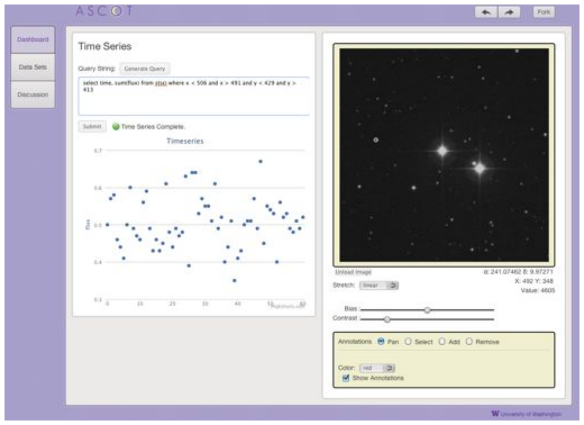
The AStronomical COllaborative Toolkit (ASCOT) [1] is a collection of Web-based gadgets that facilitate collaboration between astronomers. These gadgets can be assembled into a dashboard and communicate using a node.js server. Through the use of a customizable dashboard interface, users can easily visualize, manipulate, and share large data sets from many different sources.

Our demonstration makes two contributions: (1) First we demonstrate a scientifically useful integration of ASCOT and SciDB. We call the integrated tool AscotDB. We demonstrate the data exploration enabled by this integrated tool on a terabyte-sized dataset. (2) Second, we demonstrate basic iterative processing in SciDB and several optimizations that significantly improve performance.

2. WHAT THE USER WILL SEE AND DO?

We will take visitors to the demonstration through the following steps. The result of each task will appear in a new window and users will be able to go back and forth between windows. Figure 1 shows a sample screenshot from AscotDB.

1. The user will start by selecting a region of the sky and a timestep. AscotDB will display a composite telescope image that is representative of the selected region at the chosen time.
2. The user will then have the option to select a subregion of the sky within the image by drawing a bounding box on the original pixel image. She will be able to further manipulate her selection by either zooming in or zooming out.
3. For the selected region, the user will be able to run an image analysis task called "co-addition" where images in the database that overlap the selected location are added together, enabling faint sources to become visible.
4. The result from the previous step is noisy. We will show how to launch an iterative outlier removal algorithm, called "sigma-clipping", before the co-addition to achieve a much cleaner result.
5. An important component of this demonstration is the exploration of SciDB's new iterative processing features. For this, the user will be able to switch between



select time, sum(flux) from stsci where x < 506 and x > 491 and y < 429 and y > 413

Figure 1: Demonstration: A sample screenshot from AscotDB with a pixel image and a time-series chart.

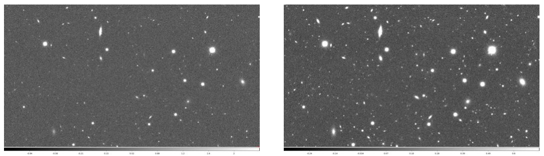
naïve iterative processing and optimized in-engine processing. To test the selected configuration, the user will re-run step 4.

6. The demonstration will also explore optimizations that speed-up re-executions of the same iterative analysis. To test this new feature, the user will be able to change the threshold k in the "sigma-clipping" algorithm (Algorithm 1) and re-run the computation. Similarly, the user will be able to generate a time-series graph of aggregate pixel values for a given location on the sky as shown in Figure 1 and use that graph to focus on or to exclude certain timesteps from the analysis before re-executing it. In both cases, we will demonstrate that AscotDB executes these subsequent re-runs faster thanks to iteration-specific caching features.

3. USE-CASE

The Large Synoptic Survey Telescope (LSST [5]) is a large-scale, multi-organization initiative to build a new telescope and use it to continuously survey the visible sky. The LSST will generate tens of TB of telescope images every night. The planned survey will cover more sky with more visits than any survey before. The novelty of the project means that no current dataset can exercise the full complexity of the data expected from the LSST. For this reason, before the telescope produces its first images, astronomers are testing their data analysis pipelines, storage techniques, and data exploration using realistic but simulated images. In this demonstration, we use a 1TB dataset of such simulated images and present a simple but fundamental type of processing that needs to be performed on those images.

ASCOT [1] is a tool used in the astronomy domain that enables scientists to visualize sky survey data and to share their visualizations with others for discussion. ASCOT is a collection of Web-based gadgets running in communication with a node.js server. The gadgets can be assembled into a dashboard. The current set of gadgets is geared toward exploratory analysis including viewing images, querying catalogs, and plotting the results. ASCOT's query capabilities are currently limited to querying only the metadata about telescope images, such as selecting and displaying the sources detected in the images. However, significantly more



(a) Single Image (b) Co-added image

Figure 2: Illustrative comparison of one *single* image and its corresponding *co-added* image. There are many faint objects that show up in the co-added image but not in the single image

Listing 1 Pseudocode for sigma-clipping and co-addition

```
Input: Array A with pixels from all the x-y images over time.
//Part 1: Iterative sigma-clipping
While(some pixel changes)
 For each (x,y) location
  Compute mean/stddev of all pixel values at (x,y).
  Filter any pixel value that is k
  standard deviations away from the mean
//Part 2: Image co-addition
 Sum all non-null pixel values grouped by x-y
```

interesting science can be derived from querying the raw pixel data directly. AscotDB is the extended version of ASCOT with the ability to query raw pixel images using SciDB. The challenge is that some of the required processing tasks are iterative in nature. In this section, we present the details of one of those tasks and later we present the novel SciDB features necessary to support it.

The integrated AscotDB tool enables several new types of analysis. We focus on two specific tasks here, one of which is iterative:

Sigma-Clipping and Co-Addition in LSST images. When analyzing telescope images, some sources (a "source" can be a galaxy, a star, etc.) are too faint to be detected in one image but can be detected by stacking multiple images from the same location on the sky. The pixel value (`flux` value) summation over all images is called image *co-addition*. Figure 2 shows a *single* image and the corresponding *co-added* image. Before the co-addition is applied, astronomers often run a "sigma-clipping" noise-reduction algorithm. The analysis in this case has two steps: (1) outlier filtering with "sigma-clipping" and then (2) image co-addition. Listing 1 shows the pseudocode for both steps.

Algorithm 1 illustrates the pseudocode in `AQL`, the SQL-equivalent language used by SciDB. The 3D input array `Array A <float data>[x,y,t]` comprises all x-y images over time t.

4. USER INTERACTION WITH AscotDB

We now describe the AscotDB capabilities in more detail. There are two fundamental phases as the user interacts with AscotDB: the *Analysis phase* and the *Exploration phase.*

Analysis Phase. In this phase, the user can run and observe the results of the "co-addition" and iterative "sigma-clipping" tasks as described in Section 3. The user selects the region of the sky that forms the input to the analysis.

In the context of the demonstration, the user also has the choice to run the "sigma-clipping" algorithm in a naïve way

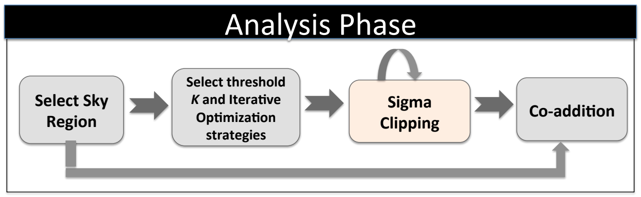
Algorithm 1 Sigma-clipping iterative algorithm followed by image co-addition

```
1: Input: Array $A$ <float $data$>[$x$,$y$,$t$]
2: Input: $k$ a constant parameter.
3: while (some pixels $A[x,y,t]$ are filtered) do
4:     $T[x,y]$ = SELECT $AVG(A.data)$ AS $\mu$, $STDV(A.data)$ AS $\sigma$ FROM $A$ GROUP BY $x,y$
5:     $S[x,y,t]$ = SELECT $A.data$, $T.\mu$, $T.\sigma$ FROM $T$ join $A$ on $T.x = A.x$ AND $T.y = A.y$
6:     $A[x,y,t]$ = SELECT $S.data$ FROM $S$ WHERE $S.data \geq S.\mu - k \times S.\sigma$ AND $S.data \leq S.\mu + k \times S.\sigma$
7: end while
8: $R[x,y]$ = SELECT SUM($A.data$) AS $coadd$ from $A$ GROUP BY $x,y$
```

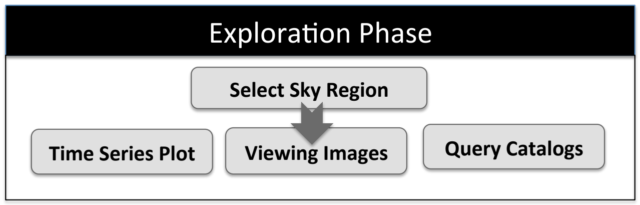
or to enable various optimizations (described in Section 5) and observe the performance differences. The user can also tune parameters, such as the threshold k, to observe its effect on the number of iterations until convergence. The sequence of tasks in this phase is illustrated in Figure 3(a).

Exploration Phase. AscotDB supports multiple exploratory tasks such as viewing images, querying catalogs, and plotting time-series data. This phase is illustrated in Figure 3(b). In this demonstration, we focus on the time-series capability: the user can select an arbitrary region on the sky and generate a time-series for that region as shown in Figure 1. The time-series plot that AscotDB supports shows the value of one of the attributes (*e.g.* flux value) over time. The value shown is an aggregate value computed over the entire selected region.

Each point in the time-series plot corresponds to one timestep in our original pixel images. AscotDB gives the option to the user to select *interesting* points in the time series, filter out the rest, and redo the analysis (*i.e.*, sigma-clipping and co-addition) on a subset of pixel images corresponding to the interesting points in the time series. As Figure 4 illustrates, the output of the exploration phase is fed back as refined input to the next analysis phase. Repetitions of the analysis tasks as a result of the exploration phase open up opportunities for optimizations as we discuss in the next section.



(a) Analysis phase: The user selects a region on the sky, tunes parameters, and runs the "sigma-clipping" algorithm followed by image "co-addition". The user also has the option to perform "co-addition" on the original images directly.



(b) Exploration phase: AscotDB supports multiple exploratory tasks such as viewing images, querying catalogs, and plotting time-series data.

Figure 3: "Analysis" and "Exploration" phases

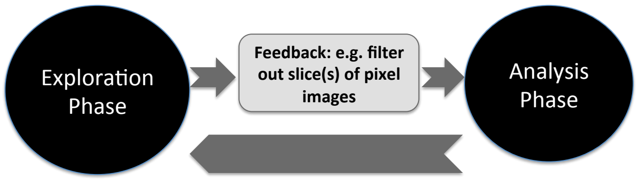


Figure 4: The user interacts with AscotDB by alternating between exploration and analysis, which enables optimizations that leverage earlier computations.

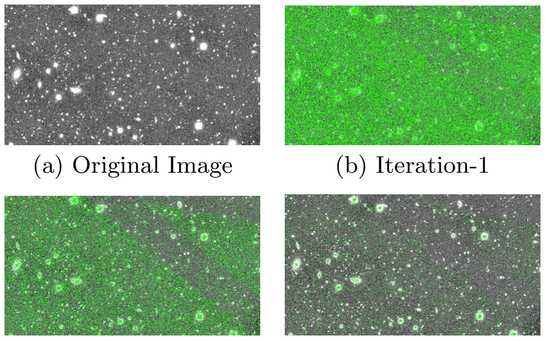
5. ITERATIVE OPTIMIZATION TECHNIQUES

To execute an iterative computation in SciDB, a naïve approach is to simply recursively invoke array queries from a script. This approach, however, prevents (or at least complicates) the automated optimization of iterative computations. Instead, we extend SciDB with two types of optimizations: incremental iterative processing and caching Intermediate results and statistics to speed-up re-executions of iterative computations in the face of changes in the input data or input parameters.

Incremental Iterative Processing. In a wide range of iterative algorithms, the output at each iteration differs only partly from the output at the previous iteration. Performance can thus significantly improve if the system computes, at each iteration, only the part of the output that changes rather than re-computing the entire result every time. This method, called *incremental iterative processing* [3], has been shown to significantly improve performance in relational and graph systems. In AscotDB, we show how this optimization can be applied to arrays. Our key observation is that increments between iterations translate into updates to array cells and can thus be captured with two auxiliary arrays: a *positive delta array* and a *negative delta array*. At each iteration, the positive delta array ΔA^+ records the *new* values of updated cells and the negative delta array ΔA^- keeps track of the *old* values of updated cells. These two arrays can *automatically* be computed by the system directly at the storage manager level. To achieve high performance, the storage manager keeps chunks of the result array A together on disk with the corresponding chunks from the auxiliary ΔA^+ and ΔA^- arrays. Furthermore, we extend the `Scan()` and `Store()` operators to read and write partial arrays ΔA^+ and ΔA^-, respectively. The user does not need to explicitly write a user-defined `diff()` function in order to extract ΔA^+ and ΔA^- from the output of the last iteration nor to write an explicit `merge()` function to combine

Algorithm 2 The incremental version of "sigma-clipping" iterative algorithm followed by image "co-addition"

```
1: Input: Array $A$ <float $data$>[$x$,$y$,$t$], $k$ a constant parameter.
2: Initialize $\Delta A^- = A$
3: while ($\Delta A^-$ is not empty) do
4:     $T_1[x, y]$ = SELECT COUNT($A.data$) AS $c$, SUM($A.data$) AS $s$, SUM($data^2$) AS $s^2$ FROM $\Delta A^-$ GROUP BY $x, y$
5:     $T[x, y]$ = SELECT $\frac{T_1.s}{T_1.c}$ AS $\mu$, $\sqrt[2]{\frac{T_1.s^2}{T_1.c} - (\frac{T_1.s}{T_1.c})^2}$ AS $\sigma$ FROM $T_1$
6:     $S[x, y, t]$ = SELECT $A.data$, $\Delta T^+.\mu$, $\Delta T^+.\sigma$ FROM $\Delta T^+$ join $A$ on $\Delta T^+.x = A.x$ AND $\Delta T^+.y = A.y$
7:     $A[x, y, t]$ = SELECT $S.data$ FROM $S$ WHERE $S.data \geq S.\mu - k \times S.\sigma$ AND $S.data \leq S.\mu + k \times S.\sigma$
8: end while
9: $R[x, y]$ = SELECT SUM($A.data$) AS $coadd$ from $A$ GROUP BY $x, y$
```



(a) Original Image (b) Iteration-1

(c) Iteration-2 (d) Iteration-3

Figure 5: Snapshots from the first 3 iterations of the "sigma-clipping" algorithm with incremental optimization on the LSST dataset. Green-colored points are the ones that change across iterations. As the iterative computation proceeds, the number of green-colored points drops dramatically.

the output of the last iteration with the current result of the iteration. Those functions are pushed into the storage manager and are automatically handled by the system with some hints from the user.

In the current extension of SciDB, the engine calculates the *algebraic* average function `AVG()` based on the combination of `SUM()` and `COUNT()` to leverage incremental iteration. The incremental version of "sigma-clipping" algorithm is shown in Algorithm 2.

Multiple snapshots from applying the incremental sigma-clipping algorithm to a subset of the `LSST` dataset are shown in Figure 5.

Caching Intermediate Results and Statistics. In AscotDB, the user alternates between the "analysis phase" and the "exploration phase". During the "exploration phase", the user can leverage the time-series view to focus her interest on a subset of timesteps. She can then re-execute the analysis on this slice of the input data array. Similarly, the user can change input parameters, such as the "sigma-clipping" parameter k, and re-execute the analysis. In this demonstration, we explore two techniques to speed-up such re-runs of iterative computations. In the first technique, AscotDB caches intermediate results from the *first* and *last* iterations of each computation and leverages these intermediate results to speed-up subsequent re-runs of the analysis when either the input array or the input parameters change. Unlike Naiad [6], which includes extensive caching of intermediate results, we focus on the first and last iterations only to ensure benefit from caching at a lower cost. In the second technique, AscotDB computes extra statistics on the data during the first run of the iterative analysis to speed-up subsequent runs. This technique also speeds-up the body of iterations within the same execution of the analysis. In the demonstrated use-case, the statistics capture the minimum and maximum value of all cells in an array chunk, which enables AscotDB to skip over chunks where no values need to be filtered out during one sigma-clipping iteration.

6. CONCLUSION

In summary, we demonstrate new capabilities in SciDB for efficient iterative processing. While these capabilities are generally applicable, we demonstrate them in the context of the analysis of a 1 TB set of astronomy telescope images.

7. ACKNOWLEDGMENTS

This work is partially supported by grants from the LSST corporation, NASA grant NNX09AK84G, DOE award DESC0002607, NSF grant IIS-1110370 and the Intel Science and Technology Center for Big Data.

8. REFERENCES

[1] Ascot:the astronomical collaborative toolkit. `http://ascot.github.com`.

[2] B. et. al. The multidimensional database system RasDaMan. In *Proc. of the SIGMOD Conf.*, pages 575–577, 1998.

[3] S. Ewen et al. Spinning fast iterative data flows. In *Proc. of the 38th Int. Conf. on Very Large DataBases (VLDB)*, pages 1268–1279, 2012.

[4] Hey et. al., editor. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, 2009.

[5] Large Synoptic Survey Telescope. `http://www.lsst.org/`.

[6] F. McSherry, D. Murray, R. Isaacs, and M. Isard. Differential dataow. In *Proc. of the Sixth CIDR Conf.*, 2013.

[7] J. Rogers et al. Overview of SciDB: Large scale array storage, processing and analysis. In *Proc. of the SIGMOD Conf.*, 2010.

[8] Zhang et. al. RIOT: I/O-efficient numerical computing without SQL. In *Proc. of the Fourth CIDR Conf.*, 2009.