Introducing AXS: A framework for large-scale analysis of astronomical data

Petar Zečević PETAR.ZECEVIC@FER.HR
Faculty of Electrical Engineering
University of Zagreb
Unska 3, Zagreb, Croatia

Colin T. Slater, Ph.D. CTSLATER@UW.EDU
Astronomy Department
University of Washington
Box 351580, Seattle, WA 98195, USA

Sven Lončarić, Ph.D. SVEN.LONCARIC@FER.HR
Faculty of Electrical Engineering
University of Zagreb
Unska 3, Zagreb, Croatia

Mario Jurić, Ph.D. MJURIC@UW.EDU
Astronomy Department
University of Washington
Box 351580, Seattle, WA 98195, USA

Abstract

Astronomy eXtensions for Spark, or AXS, is a framework for large-scale astronomical data analysis based on Apache Spark, cutting-edge open-source engine for processing large amounts of data. The ever expanding scale of today's astronomical surveys demand scalable and stable tools to help extract scientific information from the resulting data sets. However, astronomical software support is lacking in this regard. AXS aims to fill this void by providing easy-to-use Spark-based APIs with features such as on-line cross-matching and spatial selection. AXS has so far been used at University of Washington's DIRAC Institute for analysis of Zwicky Transient Facility (ZTF) and other datasets. AXS is capable of cross-matching Gaia DR2 (1.7 billion rows) and ZTF (2.9 billion rows) in 25 seconds (with data cached in filesystem) while users can further analyze results with custom Python code, within the same framework. AXS' long-term goal is to become a preferred tool for individual researchers or groups when they need to analyze astronomical datasets, whether small or at petabyte scales.

Keywords: astronomy, cross-matching, framework, astronomical data analysis

1. Introduction

Astronomy today requires processing of huge amounts of data. Each new astronomical survey produces larger amounts of data than those that came before it. For example, Data Release 14 (the latest release) of the Sloan Digital Sky Survey (SDSS), whose mission started back in 2003, contains about 150 GB of data and 1.2 billion objects. Large Synoptic Survey Telescope (LSST), starting in 2022, will observe approximately 20 billion objects, 1000 times each, and will generate about 50 PB of data in the 10 years of its planned activity period.

However, software tools commonly used in astronomy aren't ready for such large volumes of data. Astronomers typically use SQL interfaces to select, preprocess and download smaller data sets as FITS files and then use custom programs and scripts, often written in

Python, to further analyze those data. One of the operations astronomers often need to do is a cross-match, where objects from one catalog are matched to the corresponding objects from a different catalog, based on their coordinates in the sky. This is a computationally very demanding operation when the two catalogs contain large numbers of objects.

Astronomical data is often organized in the so-called *light-curves*, corresponding to multiple measurements (observations) of a single object. LSST will, for example, observe each object about 1000 times on average, in the 10 years of its planned operation period (Collaboration et al., 2009). For ground-based surveys these observations are taken with different filters, but astronomers are mostly interested in analyzing light-curves belonging to the same filter. Therefore, any tool for processing astronomical data has to allow for efficient light-curve data queries on a per-filter basis.

The goal of AXS is to provide astronomers with a simple, easy-to-use, efficient and scalable tool for cross-matching, processing and analyzing large amounts of astronomical data.

AXS is based on Apache Spark (Zaharia et al., 2016), a general-purpose framework for large-scale data processing. Spark offers Scala, Java, Python, R and SQL interfaces to its rich APIs comprising unstructured and structured data processing, streams processing, graph analytics and machine learning. It is also extremely scalable, tolerant to failures of individual components and implements advanced performance optimizations. Because of its general applicability, Spark is widely used in projects in various industries and academic settings. It is actively developed and enjoys a large user base.

This all makes Spark an excellent base for building a tool for analyzing and processing data from large astronomical catalogs.

2. Using AXS

AXS API is meant to make manipulating and cross-matching astronomical catalogs straightforward and easy to use; flexible so that astronomers can extend it with their custom functions; and sufficiently high-level so that users don't have to deal with the underlying data partitioning details.

`AxsCatalog` is the starting point for loading data with AXS API. It is a replacement for Spark's `Catalog` object, but a Spark session needs to be initialized before constructing an `AxsCatalog` instance because `AxsCatalog` relies on Spark for reading and writing table data. So, a `SparkSession` object needs to be provided for `AxsCatalog` object construction:

```
# create a SparkSession object
spark = SparkSession.getOrCreate()
# create an AxsCatalog object
from axs import AxsCatalog
AxsCatalog = AxsCatalog(spark)
```

An `AxsCatalog` instance is then used for loading data from tables, like in the following example:

```
AxsCatalog.list_tables()
... list of tables ...
sdss = AxsCatalog.load("sdss")
gaia = AxsCatalog.load("gaia")
```

Existing Spark or AXS tables can be imported into an AXS catalog using method `import_existing_table`. A partial data increment containing additional data with the

same schema can be added to an existing table using method `add_increment(table_name, increment_dataframe)`.

Table data in AXS are represented by the `AXSFrame` class, which extends Spark's `DataFrame` functionalities adding several functions useful to astronomers. One of those is the `crossmatch` method for cross-matching two `AXSFrame`s:

```
from axs import Constants
gaia_sdss_cm = gaia.crossmatch(sdss, 2*Constants.ONE_ASEC)
gaia_sdss_cm.select("ra", "dec", "r").save_axs_table("gaia_sdss_r")
```

The last command (`save_axs_table`) saves the cross-matched data on disk and partitions the data so that future queries and cross-matching operations run quickly.

`region` and `cone` methods allow users to select subsets of data based on a rectangle defined by two points, and by a circle defined by the central point and a radius (respectively). `histogram` and `histogram2d` methods allow for binning data in one or two dimensions based on custom criteria. `add_column` and `add_primitive_column` aim to make it easier for users to define new columns in an `AXSFrame` using Spark's `udf` and `pandas_udf` mechanisms.

Light-curve data in AXS are stored as array columns. To enable all operations on light-curve data columns, we extend Spark's API with two additional functions:

`array_allpositions`, which returns an array of indexes of all occurrences of an element

`array_select`, which returns all elements indexed by an array of indices.

These two functions, together with the rest of AXS and Spark APIs, enable full range of light-curve data operations.

3. Implementation details

Efficiency of AXS in data cross-matching and querying is based on its special data partitioning scheme, the underlying Spark API and its sort-merge join optimizations. We will not go into details of Spark API here, but describe the most important features that make AXS fast.

3.1 AXS data partitioning

AXS data partitioning is based on the *zones algorithm* (Gray et al., 2007) well-known in the astronomy community, but adapted for a distributed, shared-nothing architecture. AXS partitions the sky into horizontal stripes called *zones* of a fixed height (one arc-minute by default, which gives 10800 zones). All objects within the same *Dec* (declination) coordinate range get assigned the same zone, according to the following simple formula (where Z is zone height and *dec* is the declination coordinate in range $[-90, 90]$):

$$zone = \lfloor (dec + 90)/Z \rfloor$$

AXS then places zones in *buckets*, which are implemented as Parquet files on a distributed filesystem, so that each subsequent zone gets placed in a different bucket. In other words: $bucket = zone\%N$. Default number of buckets is 500. Figure 1 shows this concept graphically (for a much smaller number of buckets and zones than what is used in reality).

3.2 Fast distributed catalog cross-matching

For fast cross-matching AXS uses Spark's sort-merge join implementation with a contribution of an epsilon-join optimization. Epsilon-join (Silva et al., 2010) uses range conditions

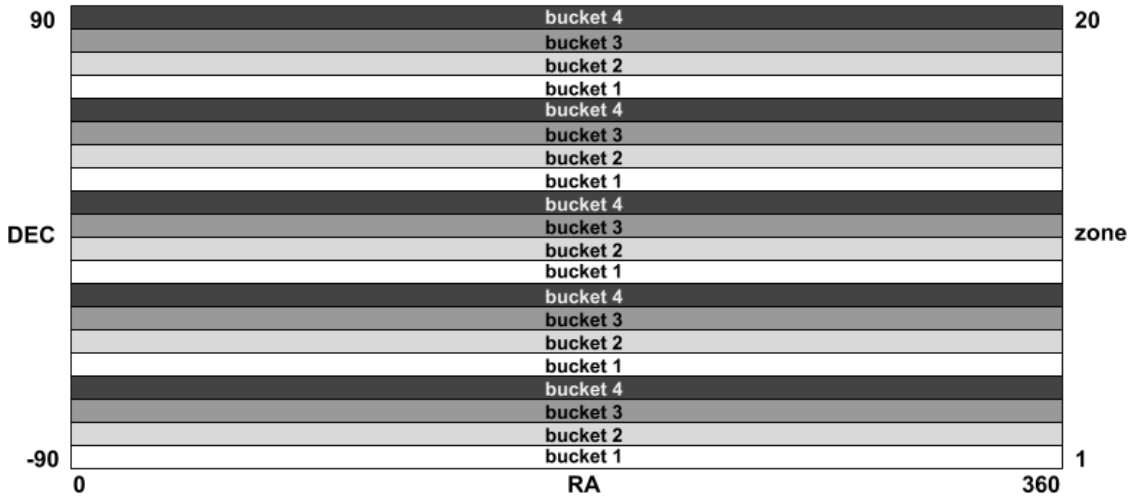

Figure 1: Partition of sky into 20 zones and placement of zones into 4 buckets. In reality, a much larger numbers are used.

on secondary columns to restrict the number of rows considered during the sort-merge join. This results in a moving window going across rows of the right table as the left row changes.

Concretely for AXS, data inside buckets are sorted by `zone` and `ra` columns and a query such as the following one can take advantage of these optimizations.

```
select * from gaia, sdss where gaia.zone = sdss.zone AND
  gaia.ra BETWEEN (sdss.ra + DELTA, sdss.ra - DELTA) AND
  distance(gaia.ra, gaia.dec, sdss.ra, sdss.dec) < DELTA;
```

The preceding query results in one task per bucket (the number of tasks executed in parallel depends on Spark configuration). Each task goes sequentially through rows in a `gaia` table's bucket and maintains a list of matching rows from the `sdss` table's bucket that satisfy the condition `gaia.zone = sdss.zone` and whose `ra` columns fall into the specified range. In this way, the `distance` function is evaluated only for those rows in the moving window, the data need to be read only once, and the amount of memory used in the process is minimal.

3.3 Data skew

Astronomical data are often highly skewed because some parts of the sky are much more densely populated than the others and because many surveys particularly concentrate on specific regions. Skewed data can pose a problem for distributed data processing because some tasks might take much longer time to complete than the others. Data partitioning scheme presented previously elegantly solves data skew because it evenly distributes narrow strips of data across many buckets. This results in evenly-sized buckets.

4. Performance evaluation

We have tested cross-matching Gaia DR2 catalog (containing 1.7 billion objects) against ZTF catalog (containing 2.9 billion objects) on a single machine. The cross-match algorithm used for tests only considered distance based on RA and Dec coordinates. A more complex distance function can also be used (calculating cross-match likelihood, for example). The

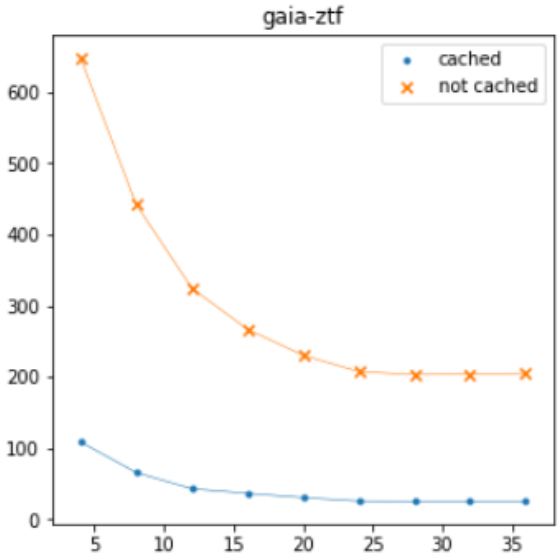

Figure 2: Cross-matching Gaia DR2 and ZTF catalogs with varying numbers of Spark executors. The results on y-axis is time in seconds. The two curves correspond to data obtained with filesystem caching turned on or off. Each data point is an average of three tests.

cross-match algorithm in this case results in 270 million rows. The resulting rows were only counted during the test and no further processing on them was performed.

We varied the number of Spark executors (degree of parallelism) and obtained the results shown in Figure 2. The two lines in the graph in the figure show tests with filesystem cache turned on and off. Each point is an average of three tests. Spark executors had 12 GB of memory.

The best result we obtained, running on a single machine, is 25 seconds when the data was cached in the filesystem, and 205 seconds when data was not cached. It is apparent from the tests that adding more than 22 executors cannot speed-up the computation any further.

5. Conclusion and future work

In this paper we introduced AXS, a flexible and fast system for cross-matching and analyzing data from astronomical catalogs based on Spark and offering Python APIs. The goal of AXS is to be easy to use for an average astronomer and to enable them to perform most of their data analysis within a single framework, with various catalogs at their disposal. We performed basic usability and performance testing. In the near future we plan to further refine the AXS API, perform more extensive performance tests and further optimizations, and make AXS available to the wider astronomy community.

References

LSST Science Collaboration, Paul A. Abell, Julius Allison, Scott F. Anderson, John R. Andrew, J. Roger P. Angel, Lee Armus, David Arnett, S. J. Asztalos, Tim S. Axelrod, Stephen Bailey, D. R. Ballantyne, Justin R. Bankert, Wayne A. Barkhouse, Jef-

frey D. Barr, L. Felipe Barrientos, Aaron J. Barth, James G. Bartlett, Andrew C. Becker, Jacek Becla, Timothy C. Beers, Joseph P. Bernstein, Rahul Biswas, Michael R. Blanton, Joshua S. Bloom, John J. Bochanski, Pat Boeshaar, Kirk D. Borne, Marusa Bradac, W. N. Brandt, Carrie R. Bridge, Michael E. Brown, Robert J. Brunner, James S. Bullock, Adam J. Burgasser, James H. Burge, David L. Burke, Phillip A. Cargile, Srinivasan Chandrasekharan, George Chartas, Steven R. Chesley, You-Hua Chu, David Cinabro, Mark W. Claire, Charles F. Claver, Douglas Clowe, A. J. Connolly, Kem H. Cook, Jeff Cooke, Asantha Cooray, Kevin R. Covey, Christopher S. Culliton, Roelof de Jong, Willem H. de Vries, Victor P. Debattista, Francisco Delgado, Ian P. Dell'Antonio, Saurav Dhital, Rosanne Di Stefano, Mark Dickinson, Benjamin Dilday, S. G. Djorgovski, Gregory Dobler, Ciro Donalek, Gregory Dubois-Felsmann, Josef Durech, Ardis Eliasdottir, Michael Eracleous, Laurent Eyer, Emilio E. Falco, Xiaohui Fan, Christopher D. Fassnacht, Harry C. Ferguson, Yanga R. Fernandez, Brian D. Fields, Douglas Finkbeiner, Eduardo E. Figueroa, Derek B. Fox, Harold Francke, James S. Frank, Josh Frieman, Sebastien Fromenteau, Muhammad Furqan, Gaspar Galaz, A. Gal-Yam, Peter Garnavich, Eric Gawiser, John Geary, Perry Gee, Robert R. Gibson, Kirk Gilmore, Emily A. Grace, Richard F. Green, William J. Gressler, Carl J. Grillmair, Salman Habib, J. S. Haggerty, Mario Hamuy, Alan W. Harris, Suzanne L. Hawley, Alan F. Heavens, Leslie Hebb, Todd J. Henry, Edward Hileman, Eric J. Hilton, Keri Hoadley, J. B. Holberg, Matt J. Holman, Steve B. Howell, Leopoldo Infante, Zeljko Ivezic, Suzanne H. Jacoby, Bhuvnesh Jain, R, Jedicke, M. James Jee, J. Garrett Jernigan, Saurabh W. Jha, Kathryn V. Johnston, R. Lynne Jones, Mario Juric, Mikko Kaasalainen, Styliani, Kafka, Steven M. Kahn, Nathan A. Kaib, Jason Kalirai, Jeff Kantor, Mansi M. Kasliwal, Charles R. Keeton, Richard Kessler, Zoran Knezevic, Adam Kowalski, Victor L. Krabbendam, K. Simon Krughoff, Shrinivas Kulkarni, Stephen Kuhlman, Mark Lacy, Sebastien Lepine, Ming Liang, Amy Lien, Paulina Lira, Knox S. Long, Suzanne Lorenz, Jennifer M. Lotz, R. H. Lupton, Julie Lutz, Lucas M. Macri, Ashish A. Mahabal, Rachel Mandelbaum, Phil Marshall, Morgan May, Peregrine M. McGehee, Brian T. Meadows, Alan Meert, Andrea Milani, Christopher J. Miller, Michelle Miller, David Mills, Dante Minniti, David Monet, Anjum S. Mukadam, Ehud Nakar, Douglas R. Neill, Jeffrey A. Newman, Sergei Nikolaev, Martin Nordby, Paul O'Connor, Masamune Oguri, John Oliver, Scot S. Olivier, Julia K. Olsen, Knut Olsen, Edward W. Olszewski, Hakeem Oluseyi, Nelson D. Padilla, Alex Parker, Joshua Pepper, John R. Peterson, Catherine Petry, Philip A. Pinto, James L. Pizagno, Bogdan Popescu, Andrej Prsa, Veljko Radcka, M. Jordan Raddick, Andrew Rasmussen, Arne Rau, Jeonghee Rho, James E. Rhoads, Gordon T. Richards, Stephen T. Ridgway, Brant E. Robertson, Rok Roskar, Abhijit Saha, Ata Sarajedini, Evan Scannapieco, Terry Schalk, Rafe Schindler, Samuel Schmidt, Sarah Schmidt, Donald P. Schneider, German Schumacher, Ryan Scranton, Jacques Sebag, Lynn G. Seppala, Ohad Shemmer, Joshua D. Simon, M. Sivertz, Howard A. Smith, J. Allyn Smith, Nathan Smith, Anna H. Spitz, Adam Stanford, Keivan G. Stassun, Jay Strader, Michael A. Strauss, Christopher W. Stubbs, Donald W. Sweeney, Alex Szalay, Paula Szkody, Masahiro Takada, Paul Thorman, David E. Trilling, Virginia Trimble, Anthony Tyson, Richard Van Berg, Daniel Vanden Berk, Jake VanderPlas, Licia Verde, Bojan Vrsnak, Lucianne M. Walkowicz, Benjamin D. Wandelt, Sheng Wang, Yun Wang, Michael Warner, Risa H. Wechsler, Andrew A. West, Oliver Wiecha, Benjamin F. Williams, Beth Willman, David Wittman, Sidney C. Wolff, W. Michael Wood-Vasey, Przemek Wozniak, Patrick Young, Andrew Zentner, and Hu Zhan. Lsst science book, version 2.0, 2009.

Jim Gray, Maria A. Nieto-Santisteban, and Alexander Szaay. The zones algorithm for finding points-near-a-point or cross-matching spatial datasets. 02 2007.

Y. N. Silva, W. G. Aref, and M. H. Ali. The similarity join database operator. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 892–903, March 2010. doi: 10.1109/ICDE.2010.5447873.

Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, October 2016. ISSN 0001-0782. doi: 10.1145/2934664. URL `http://doi.acm.org/10.1145/2934664`.