AXS: Making end-user petascale analyses possible, scalable, and usable

Petar Zečević - *University of Zagreb, Faculty of Electrical Engineering and Computing; Dirac Institute, Visiting Fellow*

Colin T. Slater - *University of Washington, Dirac Institute*

Mario Jurić - *University of Washington, Dirac Institute*

Sven Lončarić - *University of Zagreb, Faculty of Electrical Engineering and Computing*

CHARLES AND LISA SIMONYI FUND
FOR ARTS AND SCIENCES

Washington Research Foundation

Outline

- Problem description
- About Apache Spark
- AXS implementation details
- AXS performance testing results
- Future plans

Problem description

- Astronomical data is... well, astronomical
  - exponential growth
  - e.g. LSST is expected to produce about 80 PB of data

- The current *subset-download-analyze* paradigm may be too cumbersome for the next generation of datasets
  - SQL queries online
  - Download FITS files
  - process with custom Python programs

Problem description

We want a tool that:

- is scalable (can handle large datasets)
- is easy to use for a domain scientist
- is efficient (fast querying and cross-matching)
- natively handles time-series
- provides a simple and extendable interface for running analysis algorithms

Problem description

We want a tool that:

- Is built on industry-standard tools
  - industry is already dealing with the problems of this scale
  - automatically benefits from new developments in the industry
  - easier to maintain (easier to find expertise)

AXS
Astronomy eXtensions for Spark

AXS and Apache Spark

Based on **Apache Spark** because:

- Efficient in handling big, distributed data sets
- Easily scalable
- Resilient to individual worker failures
- Already provides Python interfaces
- Various connectors to third-party systems and databases
- Very large community (industry and academic)
- Actively developed

AXS in a nutshell

AXS = Spark extensions + Python library

- Minimally extends Spark with only two extensions to make cross-matching and processing fast:
  - Specific data partitioning scheme
  - Sort-merge join optimization
- Spark already provides a significant fraction of functionality needed
- AXS adds additional methods to make astronomers' lives easier
- Time-series aware

AXS - Python API examples

```
from axs import AxsCatalog, Constants

spark = SparkSession.getOrCreate()

db = AxsCatalog(spark)
sdss = db.load("sdss")
gaia = db.load("gaia")

gaia_sdss = gaia.
    crossmatch(sdss, 2*Constants.ONE_ASEC).
    select("ra", "dec", "psfMag_r", "mag_r").
    where("abs(psfMag_r - mag_r) > 1").
    count()
```

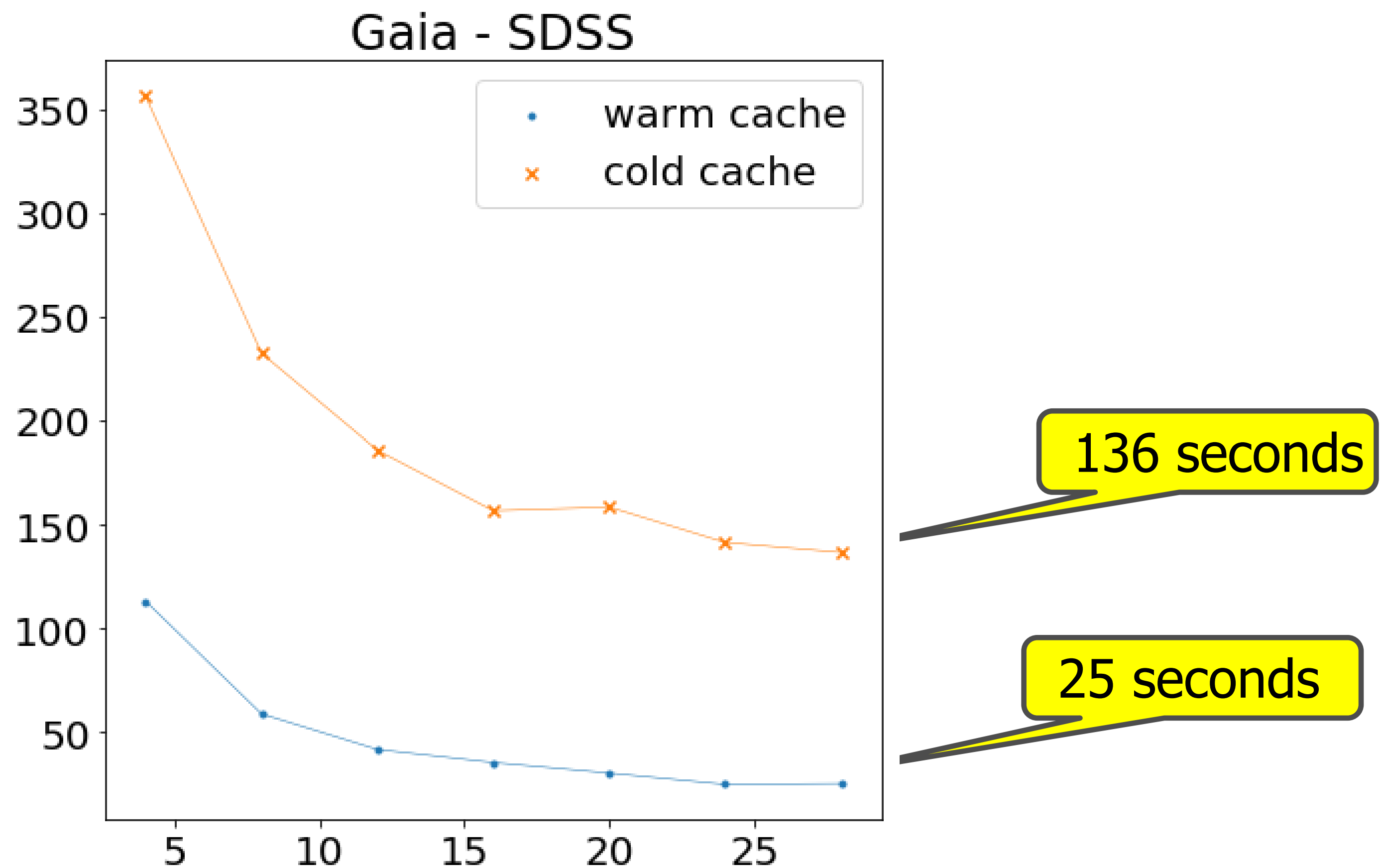
Initialize Spark

Load the catalog data

Do the crossmatch

Do further filtering and processing

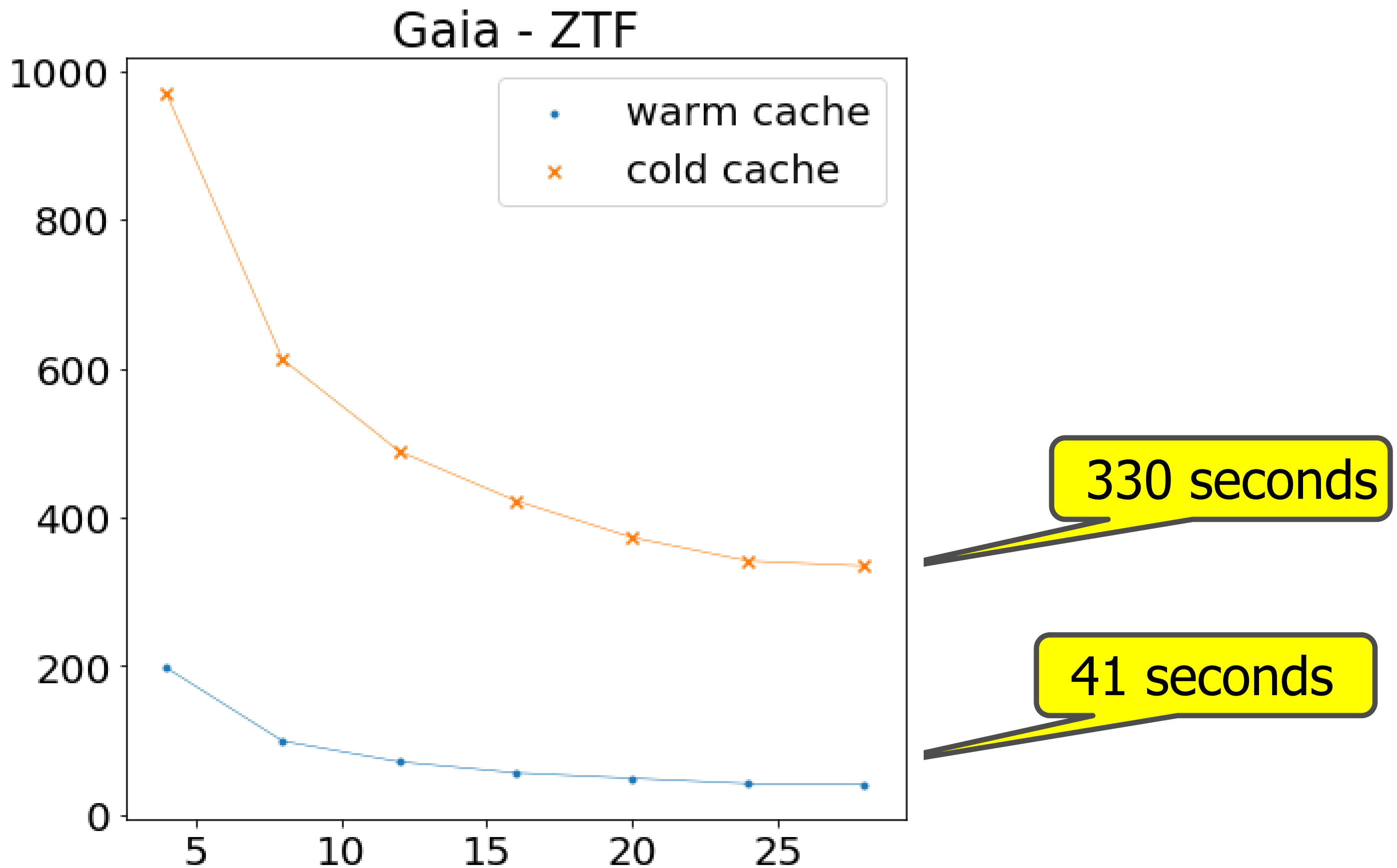
AXS - performance tests



Gaia DR2 - **1.7 billion objects** (425GB compressed)
SDSS - **710 million objects** (66GB compressed)

(one machine, 512GB RAM, 48 CPUs, fast disks)

AXS - performance tests



Gaia DR2 - **1.7 billion objects** (425GB compressed)
ZTF - **2.9 billion objects** (1.2TB compressed)

(one machine, 512GB RAM, 48 CPUs, fast disks)

AXS - API examples

Other AxsFrame methods:

- region (ra1, dec1, ra2, dec2)
- cone (ra, dec, r)
- histogram (condition, num_bins)
- histogram2d (cond1, cond2, num_bins1, num_bins2)
- add_primitive_column (colname, coltype, func, *in_col_names)
- add_column (colname, coltype, func, *in_col_names)

Ligh-curve handling:

- array_allpositions (column, value)
- array_select (column, indexes)

AXS inside - Data partitioning

- Based on the zones algorithm (Gray, Nieto-Santisteban, Szalay 2007), adapted for a distributed architecture
- Partitions the sky into horizontal strips (1 arc-min high, by default)
  - zone = (Dec+90) / NUM_ZONES
  - gives 10800 zones
- Physically stored into *buckets* - Parquet files
  - bucket = zone % NUM_BUCKETS
  - 500 buckets, which gives 21 zones per bucket, on avg, by default
- Data inside buckets sorted by *zone* and *ra* columns

AXS inside - Distributed x-matching

- Spark's sort-merge join with our epsilon-join implementation (Silva et al. 2010)
- Spark not able to optimize this query:

```
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;
```

- Epsilon-join uses a moving window
  - slides over right table's rows (sdss) as the left row changes (gaia)
  - reduces the number of rows considered
  - only one pass through the data is needed
  - uses minimal amount of memory

AXS - what's next?

- Currently using it to work with ZTF data (2.9 billion rows)
  - enable science!
- Performance testing and optimization
- Paper in preparation (submitting very soon)
- Making AXS widely available (Github, Conda, cloud, documentation & tutorials) and collecting initial feedback
  - make ZTF DR1 available on a cloud resource, ready for analysis

If you are interested in taking AXS for a spin, please contact us:

petar.zecevic@fer.hr
mjuric@uw.edu,
ctslater@uw.edu
sven.loncaric@fer.hr

Thank you!

CHARLES AND LISA SIMONYI FUND
FOR ARTS AND SCIENCES
Washington Research FOUNDATION