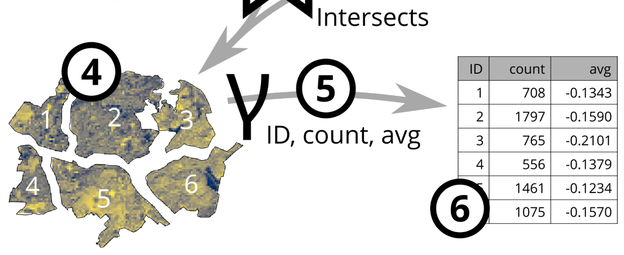
Multi-Backend Zonal Statistics Execution with Raven

Gereon Dusella, Haralampos Gavriilidis, Laert Nuhu, Volker Markl, Eleni Tzirita Zacharatou

MULTIMODAL DATA

Zonal Statistics

Given a vector ① and a raster dataset ② a **Zonal Statistics (ZS) Query** joins ③ each vector feature with the underlying raster pixels ④, calculates aggregates ⑤ of all raster pixels contained in each vector feature, and returns a results table ⑥.



ID	count	avg
1	708	-0.1343
2	1797	-0.1590
3	765	-0.2101
4	556	-0.1379
5	1461	-0.1234
6	1075	-0.1570

Zonal Statistics Query Example

INTERFACES

Linguistic Clutter

All ZS-capable systems implement their own language for describing ZS queries. This inconsistency harms standardization and thus increases lock-ins.

```
def main(args): Unit = {
  // Setup Spark + Beast
  // load raster and vector datasets
  val join = raster.raptorJoin(vector)

  val rdd_raw = join.map(v => (v.feature
      .getAs[String]("id"), v.m))
  spark.createDataset(rdd_raw)
    .toDF(Seq("id", "value"): _*)
    .createOrReplaceTempView("raptorjr")

  val zonal_stats = spark.sql("""
  SELECT rj.id, COUNT(rj.value) AS count,
    AVG(rj.value) AS avg
  FROM raptorjr rj GROUP BY rj.id
  """)
  // Return result
}
```

ZS Query in Beast

```
for p in (sentinel)
  return avg( clip( c,
    Polygon((14.0061 52.7301, 14.0404 52.7300,
             14.0341 52.7420, 14.0464 52.7522,
             14.0131 52.7526, 14.0061 52.7301)),
    "http://loc:80/rasdaman/def/crs/EPSG/0/3587"
  ))
```

Execute $|features|$ times

ZS Query in RasDaMan

```
SELECT v.id,
  SUM(t.count) AS count,
  SUM(t.count * t.value) / SUM(t.count) AS mean,
FROM plots v JOIN sentinel r
  ON ST_Intersects(raster.rast, vector.geom),
  ST_ValueCount(
  ST_Clip(raster.rast, vector.geom), 1) AS t
GROUP BY v.id
```

ZS Query in PostGIS

Load Datasets | Calculate overlap | Select Vector Features | Aggregation Functions

EFFORTLESS ZONAL STATISTICS

Raven: A Unified System

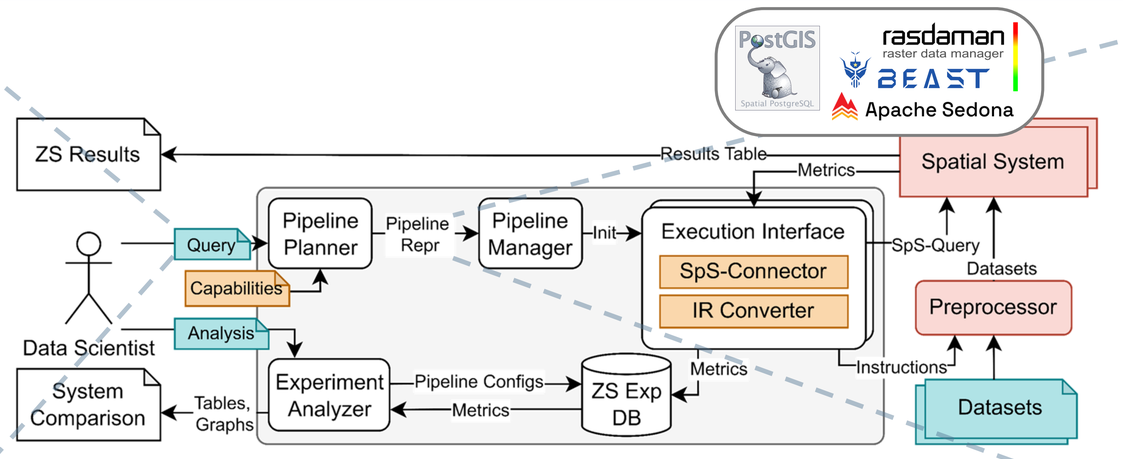
ZS Queries in Raven

```
# Datasets definition
zs_result = ZSGen.build(
        raster="/data/s2a_ndvi_20m",
        vector="/data/ALKIS_nutz_MOL")
# Aggregation operations
  .group("oid")
  .summarize({"max": ZSAgg.MAX,
              "avg": ZSAgg.AVG})
  .filter('nutzart="Landwirtschaft"')
  .join_using(ZSJoin.INTERSECT)
# Systems
  .system(System.PostGIS())
# Parameter settings (optional)
  .vectorize_type(VecType.POINTS)
  .align_to_crs(DataType.RASTER)
  .vector_filter_at(Stage.EXECUTION)
  .raster_clip(True)
```

Zonal Statistics queries in Raven are split into four parts:

- Dataset
- Aggregation
- System
- Parameters

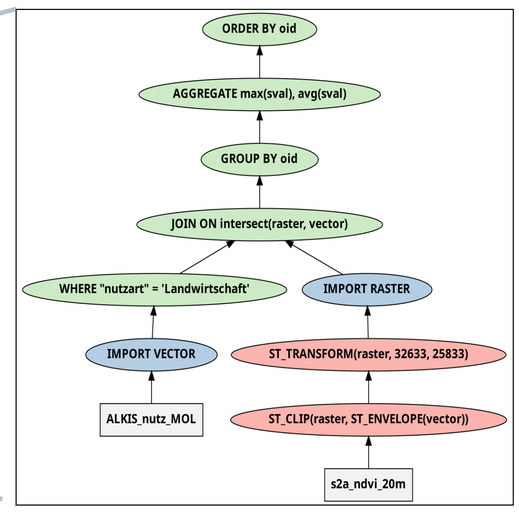
Architecture of Raven



Raven (**Ra**ster **Ve**ctor Joi**n**) enables users to always choose the best system for their ZS queries by providing:

- Domain-specific Language for describing ZS queries
- Many configuration parameters to optimize latency
- Automatic system-specific compliance checks
- Benchmarking mode for comparisons

Internal Representation

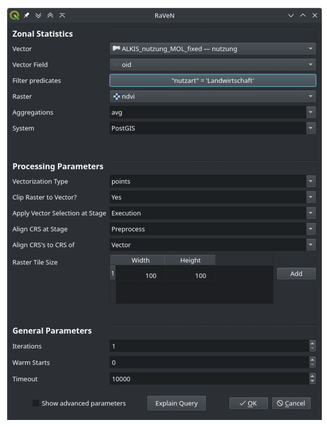


To improve performance, Raven splits operations into a system-agnostic preprocessing phase and a system-specific ingestion and execution phase.

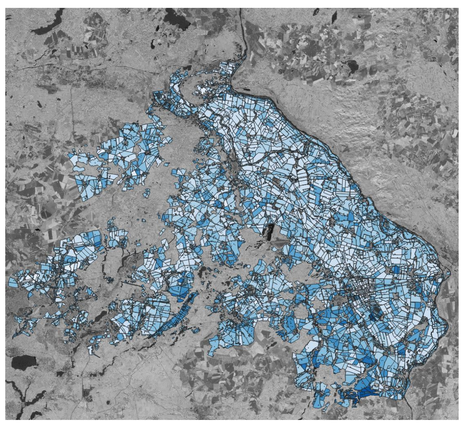
INTERACT

Raven in Action!

Raven **plugs into QGIS** and enables scientists to create analyses in their well-established working environment.



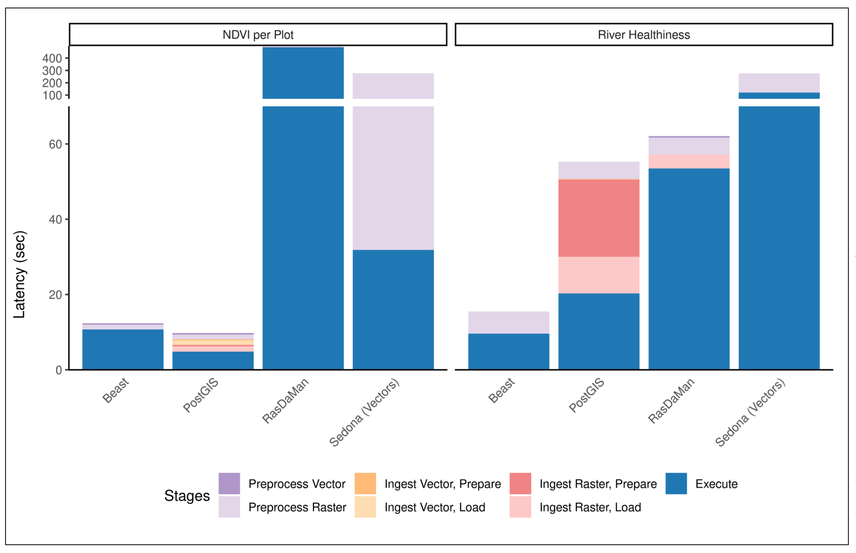
Raven UI in QGIS



ZS Query Result

PERFORMANCE

Results



NDVI per Plot

- Cadastre (78819 polygons)
- NDVI (30 Mio pixels)

River Healthiness

- Waterways (208 lines)
- Reflectance (120 Mio pixels)

OUR CODE

Try It!

polydbms/RaVeN

Raven is part of the PolyDB project for DBMS interoperability.
www.polydbms.org

TECHNISCHE UNIVERSITÄT BERLIN | Database Systems and Information Management | BIFOLD | DASYA Data-Intensive Systems and Applications | IT UNIVERSITY OF CPH