JDEM DB Development Status

Igor Mandrichenko, Vladimir Podstavkov

7/8/2010

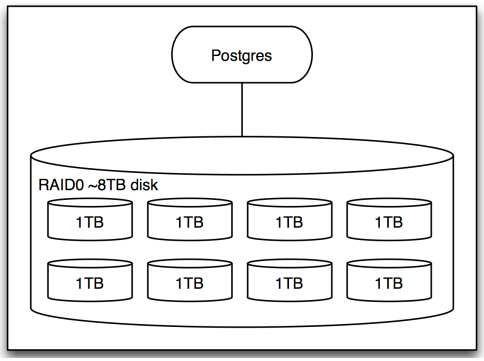
Where we are

- Work with Steve and Eric to implement the DB for slitless spectroscopy and other programs
- Prototyping JDEM use cases
  - Galaxy position reconstruction
  - Catalog cross-matching
- Comparing GreenPlum to plain Postgres

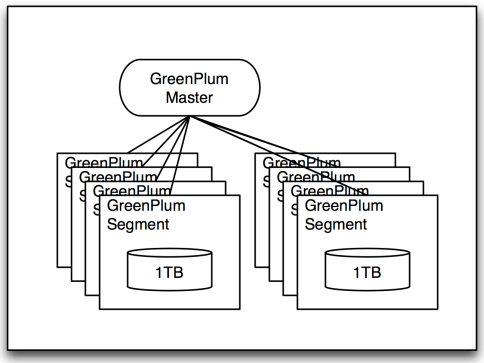
Slitless Spectroscopy Database

- 1 Day of work
  - 5,000,000 galaxies
  - 20,000,000 detections, ~10KB each
- 1 Year
  - 1.8 Billion galaxies

2 database configurations



jdemdb1: Postgres running on RAID-0 array of 8 disks, 7.2 TB in total



jdemdb2: 8-segment GreenPlum instance, each runs on its own physical disk of ~1TB

Plan: generation

- Generate 500 million galaxies as a “base catalog”. 500 million number comes from disk space limitation
  - Compare data ingestion performance of GreenPlum and Postgres (we pretty much know the result, GreenPlum is *much* faster due to the ability to ingest in parallel)

Use case 1: slitless spectroscopy reconstruction

- Generate 1 day worth of “detections”
  - 20million (5 million galaxies by 4 roll angles) biased detections from 1 x 30 degree area of the base catalog and reconstruct galaxies coordinates
  - Goal is to reconstruct all in less than 24 hours
  - Compare GreenPlum and Postgres performance and scaling
  - Compare performance with and without blobs (move detection data into separate table with 1-to-1 relationship with the coordinate table)
  - Compare 1 by 1 reconstruction algorithm with bulk reconstruction (say 500 detections at once)

Schema: Galaxy Catalog

```
Table "public.galcat"
  Column |       Type       |                        Modifiers
 --------+------------------+-------------------------------------
  galid  | bigint           | not null default nextval('galcat_galid_seq'::regclass)
  coord  | spoint           |
  row    | double precision |
  col    | double precision |
  mag    | double precision |
  radius | double precision |
  dcoord | spoint           |
 Indexes:
     "galcat_pkey" PRIMARY KEY, btree (galid)
     "galcat_coord_idx" gist (coord)
```

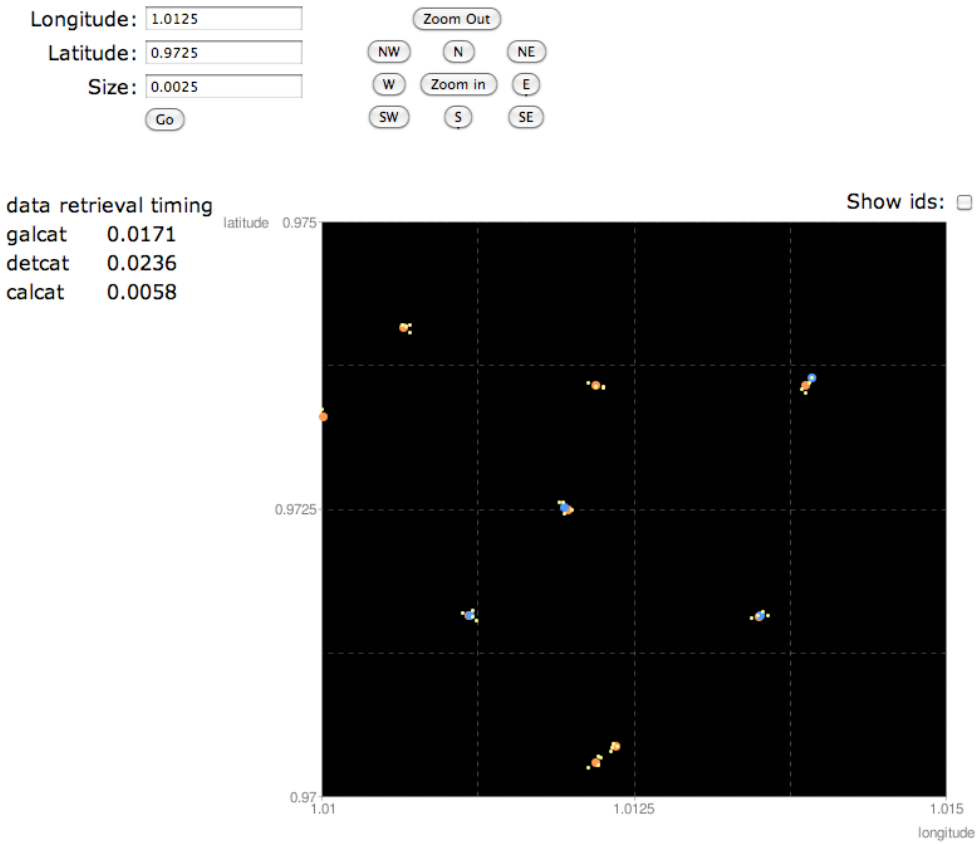
Schema: Detections Catalog

```
       Table "public.detcat"
 Column |        Type        |                        Modifiers
--------+--------------------+-----------------------------------------------------
 detid  | bigint             | not null default nextval('detcat_detid_seq'::regclass)
 galid  | bigint             |
 coord  | spoint             |
 ftab   | double precision[] |
 fbin   | bytea              |
 cgalid | bigint             |
Indexes:
    "detcat_pkey" PRIMARY KEY, btree (detid)
    "detcat_cgalid_idx" btree (cgalid)
    "detcat_coord_idx" gist (coord)
    "detcat_galid_idx" btree (galid)
Foreign-key constraints:
    "detcat_galid_fkey" FOREIGN KEY (galid) REFERENCES galcat(galid) ON UPDATE CASCADE
```

Schema: “Reconstructed” Galaxies

```
 Table "public.calcat"
 Column |  Type  |                        Modifiers
--------+--------+-----------------------------------------------------------
 galid  | bigint | not null default nextval('calcat_galid_seq'::regclass)
 coord  | spoint |
Indexes:
    "calcat_pkey" PRIMARY KEY, btree (galid)
    "calcat_coord_idx" gist (coord)
```

Database Viewer



Use Case 2: Catalog Crossmatching

- Take 1.0 by 0.5 degree area (1 exposure) of the base catalog and match all galaxies there back to the catalog, measure time.
  - Goal is < 350 seconds – to sustain 246 exposures/ day rate
  - Compare Postgres and GreenPlum performance, scaling

FITS-Database tools

- sql2fits – dump data from arbitrary query into a fits file – done

```
sql2fits –d <database> -o <fits file> … “select lon(coord), lat(coord) from
   galcat where coord @ sbox ‘(...”
```

- fits2sql – store data from a fits file into one of more database tables.

```
fits2sql –d <database> -i <fits file> … “galcat.coord=spoint(lon, lat);…”
```

- Both tools will recognize and handle pgSphere data types
  - spoint – array[2]
  - scircle – array[3]
  - sbox – array[4]

Questions

- 4 data management components
  - Astronomy Database
  - Operations Database
  - FITS files
  - Image storage
- Which part of the data model goes where ?
- How to map data model to database ?
- What technology to pick for each component ?