Seminar Assignments!

Grading, Homeworks, Etc.

- Two homeworks (50% of the grade):
  - All homeworks will be IPython notebooks
  - HW #1, handed out at the end of January, focused on software tools
  - HW #2, handed out at the end of February, focused on survey data analysis

- Paper presentations (25% of the grade):
  - Present a paper on a survey / dataset of your choosing, making sure to cover:
    - The goals of the survey
    - The data it contains (and any caveats about it)
    - What it may be useful for
    - The best ways to obtain it

- Final project (25% of the grade):
  - More about this in a week or so ☺.

Seminar Dates

- Next week:
  - Introducing SDSS and WISE
- The week after, we start with your survey seminars
  - Feb 9, 11, 16, 18, 23rd
- Please let me know which survey you'd want to cover (see next slide for some ideas)

Seminar Assignments

Survey	Presenter & Date
TESS	
Kepler	
DES	
GALEX	
VISTA	
MWA	
PTF	
CFHT-LS	
Fermi	
UKIDSS	
CRTS	
2MASS	
RAVE	
…	

Large Survey Database

The Trouble With Web Archives

- What do most astronomical web archives do for users?
  - Data storage
  - Cone searches
    - Give me all objects at position (α, δ) within radius r
  - Execution of complex queries against the database (e.g., you can in principle compute correlation functions from within an SQL query), a few at a time

- However, they are quite poor at supporting complex, whole-dataset, operations!

But that is where the science is!

- With large data sets, the name of the game is complex data mining – we need to go through the entire dataset (repeatedly), to discover interesting things.

- Corollary: we will either
  a) Want to push the computing to the data (LSST)
  b) Pull the data to your computing at home (now (and LSST!))

Examples

- Galactic structure: density/proper motion maps of the Galaxy
  - => forall stars, compute distance, bin, create 5D map
- Galactic structure: dust distribution
  - => forall stars, compute g-r color, bin, find blue tip edge, infer dust distribution
- Near-field cosmology: MW satellite searches
  - => forall stars, compute colors, convolve with spatial filters, report any satellite-like peaks
- Variability: Bayesian classification of transients and discovery of variables
  - => forall stars, get light curves, compute likelihoods, alert if interesting
- …

Large Survey Database

- A simple Python database for (large) astronomical catalogs
- http://lsddb.org
- Data access: SQL-like queries (Python)

- Current release:
  - SQL-like query language
  - Local caching of data
  - On-the-fly cross-matching of catalogs
  - ACID (Atomicity, Consistency, Isolation, Durability) transactions
  - MapReduce engine and Python API
  - Multi-core aware query engine
  - Distributed query engine (experimental)

Limitations

- A made-up, non-standard, language
  - Table definitions in YAML
  - SQL-like queries
- Limited ability to do joins
- May have strange, buggy, corner cases
- Only a few active developers

… but, it's one of the rare things that address our needs!

LSD Queries

LSD Query Syntax

1.) SQL-like, case-insensitive keywords

2.) Column specifications are Python expressions; free to call Python functions from within query clauses

```
SeLEct
    ra, dec, g, r, sdss.g as sg, sdss.r as sr,
    sg-sr as sgr, ffitskw(chip_hdr(chip_id), "ZPT_OBS") as zpt
FROM
    ps1_obj, ps1_det, ps1_exp, sdss(outer)
WHERE
    sr < 22.5
INTO
    mytable
```

4.) The WHERE clause is a Python expression, with column data given in NumPy arrays.

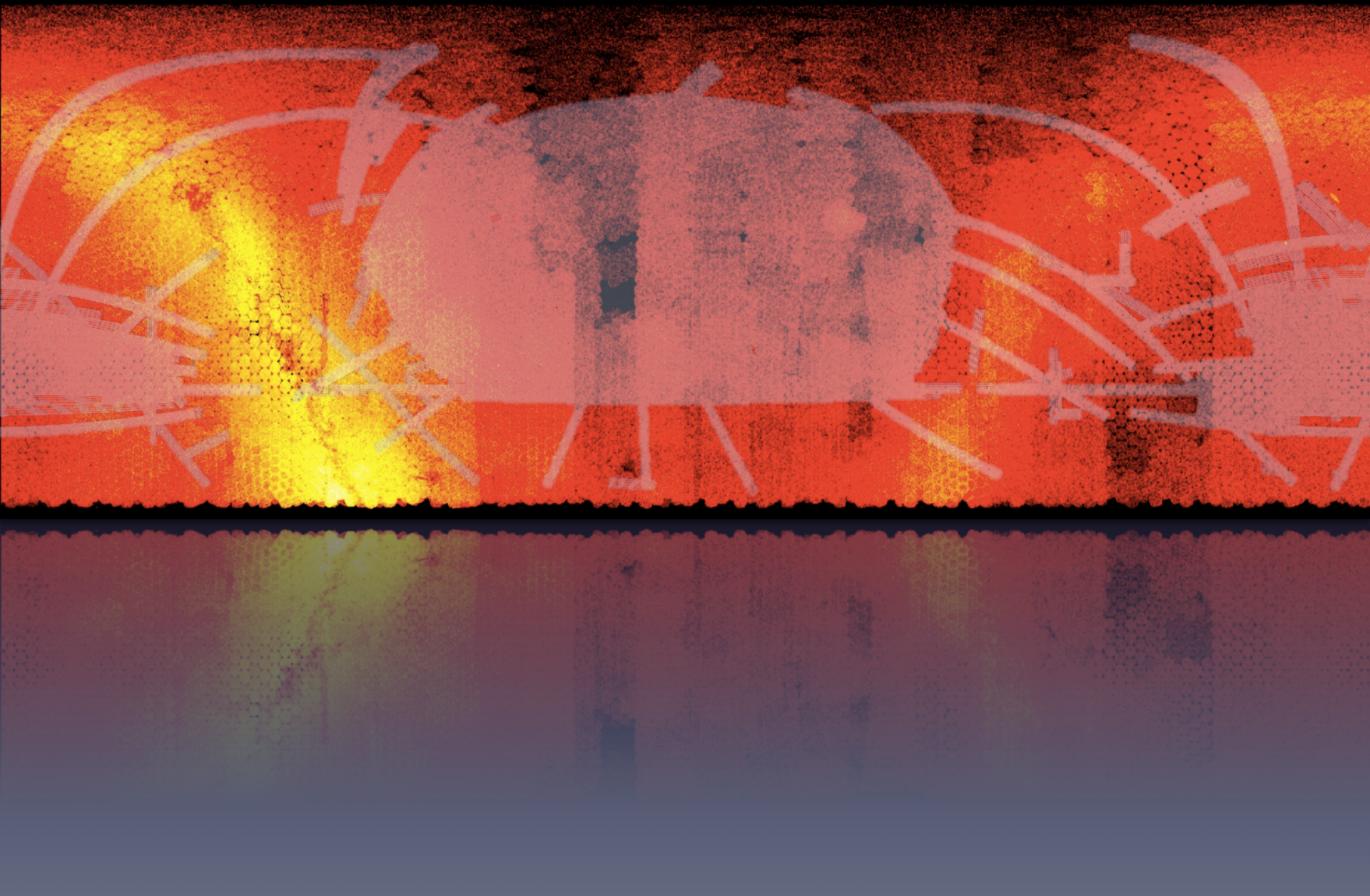
3.) Implicit natural JOINs between tables, outer JOINs supported

Details: *http://research.majuric.org/trac/wiki/LargeSurveyDatabase#QuerySyntax*

What and Where

- Catalogs:
  - Pan-STARRS PS1 (~14B rows)
  - SDSS (~300M rows)
  - GALEX (~40M rows)
  - LSST mocks (13B rows)
  - PTF
  - …

14 billion detections of PS1



LSD @ ASTR 597b

- We will primarily be using LSD to extract subsets from large data sets (mostly SDSS)
- We will use it to do catalog cross-matching
- We will be importing catalogs to Survey Science Group's LSD server
  - This will be your final project: writing the table schemas and code to load a survey into LSD
  - Note: still procuring the machine, should be here by the end of the quarter. We'll use magneto until it arrives.
  - Result: one of the more capable survey archives!

LSD Concepts: Catalogs == Tables

- **LSD is designed to work with simple catalogs; one catalog is one table.**
  - Catalogs with multiple observations are an exception and usually have three tables (one for objects, one for observations, and a "join" table connecting the two)
  - Not optimal, but generally works

Importing new catalogs

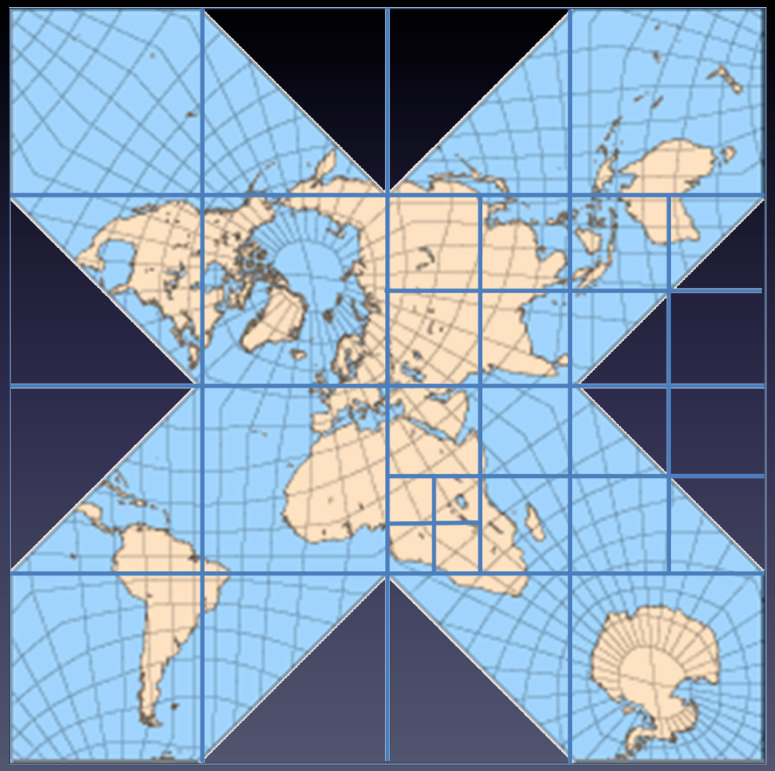
- Write a schema in YAML
- Create a table using 'lsd-admin create table'
- Use the lsd-import utility to populate the table

```
    I    ./src/schemas/usnob.yaml
# Schema for USNO-B table
filters: {complevel: 5, complib: blosc}
schema:
  common:
    primary_key: usnob_id
    spatial_keys: [ra, dec]
    columns:
    - [usnob_id, u8]
    - [ra, f8]
    - [dec, f8]
    - [sra, i2]
    - [sde, i2]
    - [epoch, f4]
    - [muRa, i2]
    - [muDec, i2]
    - [muprob, u1]
    - [muflag, u1]
    - [smura, i2]
    - [smude, i2]
    - [sfitra, i2]
    - [sfitde, i2]
    - [nfitpt, u1]
    - [mag, 5f4]
    - [fldid, 5i2]
    - [sg, 5u1]
    - [fldepoch, 5f4]
```

Partitioning

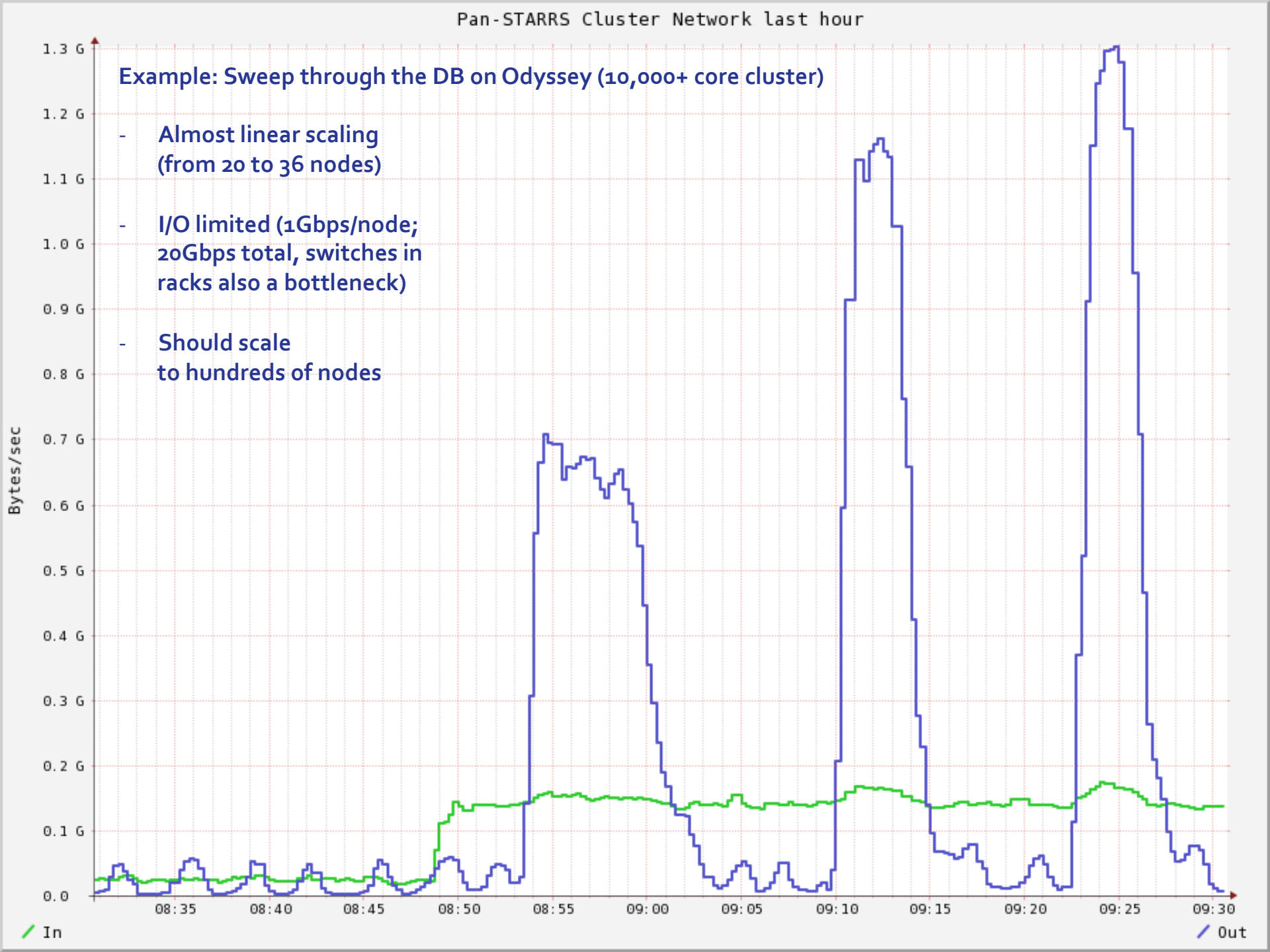
- **Each table is spatially (and, possibly temporally) partitioned**
  - The sky is subdivided in cells in space and time
  - Each cell is stored in its own directory, with multiple files containing the data (see below)
  - This is known as *horizontal partitioning* or *sharding* in database terminology
- **Each table is partitioned into a number of "column groups" – groups of columns that are frequently used together**
  - Each column group is stored in its own file within a cell
  - This is known as *vertical partitioning*

Healpix Projected, Spatially Partitioned Data



Performance

- **This design results in better performance than with traditional databases:**
  - Only the data that's needed for the query will be loaded from disk (i.e., infrequently used column groups won't be read unless referred to)
  - A query can be subdivided into pieces that run in parallel on multiple cells
  - The parallelization can span multiple machines


Pan-STARRS Cluster Network last hour
Example: Sweep through the DB on Odyssey (10,000+ core cluster)
- Almost linear scaling (from 20 to 36 nodes)
- I/O limited (1Gbps/node; 20Gbps total, switches in racks also a bottleneck)
- Should scale to hundreds of nodes
Bytes/sec
1.3 G
1.2 G
1.1 G
1.0 G
0.9 G
0.8 G
0.7 G
0.6 G
0.5 G
0.4 G
0.3 G
0.2 G
0.1 G
0.0
08:35
08:40
08:45
08:50
08:55
09:00
09:05
09:10
09:15
09:20
09:25
09:30
In
Out

Hands-on

1. Installing LSD
2. Creating tables and importing data
3. Connecting to remote tables
4. Querying from the command line
5. Catalog cross-matching with LSD
6. Using LSD from Python

In class git repo @ lectures/2016-01-28-lsd/lsd.ipynb

Important!

- LSD is *very* greedy when it comes to CPU and memory. It will try to use everything it can get its hands on. It will also create huge temp files to speed things up.

- This is not great on a shared machine (like magneto), but it can be controlled using environmental variables. So, on magneto, make sure to set:
  - export NWORKERS=8 # Number of CPU cores to use
  - export LSD_TEMPDIR=~/temp-dir # Where to keep temporary files
  - export LSD_CACHEDIR=~/temp-dir # Where to keep temporary files

- And also make sure that ~/temp-dir exists.

Heavy parallelism

```
top - 12:25:23 up 3 days,  2:55, 11 users,  load average: 6.91, 7.38, 5.04
Tasks: 603 total,  18 running, 585 sleeping,   0 stopped,   0 zombie
Cpu(s): 49.0%us,  1.5%sy,  3.0%ni, 46.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  264508168k total, 224618492k used, 39889676k free,   801192k buffers
Swap: 32767996k total,     8472k used, 32759524k free, 185778252k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
20258 mjuric    20   0 2086m 405m 4328 R 107.9  0.2   0:23.25 lsd-import
20268 mjuric    20   0 2185m 1.8g 1908 R 100.0  0.7   0:23.01 lsd-import
20254 mjuric    20   0 3264m 1.8g 4224 R 99.9  0.7   0:23.01 lsd-import
20255 mjuric    20   0 2487m 860m 4260 R 99.9  0.3   0:23.30 lsd-import
20257 mjuric    20   0 3335m 1.7g 4356 R 99.9  0.7   0:22.84 lsd-import
20259 mjuric    20   0 2173m 1.8g 1908 R 99.9  0.7   0:23.01 lsd-import
20261 mjuric    20   0 1932m 1.5g 1932 R 99.9  0.6   0:23.01 lsd-import
20262 mjuric    20   0 3514m 1.8g 4364 R 99.9  0.7   0:23.06 lsd-import
20263 mjuric    20   0 2167m 1.8g 1908 R 99.9  0.7   0:23.00 lsd-import
20266 mjuric    20   0 1905m 1.5g 1908 R 99.9  0.6   0:23.00 lsd-import
20256 mjuric    20   0 1901m 1.5g 1932 R 99.6  0.6   0:23.00 lsd-import
20260 mjuric    20   0 1918m 1.5g 1932 R 99.6  0.6   0:23.00 lsd-import
20264 mjuric    20   0 2156m 1.7g 1908 R 99.6  0.7   0:22.95 lsd-import
20265 mjuric    20   0 1955m 1.5g 1932 R 99.6  0.6   0:22.96 lsd-import
20267 mjuric    20   0 2010m 1.6g 1908 R 99.6  0.6   0:23.00 lsd-import
20269 mjuric    20   0 2349m 1.9g 1908 R 99.6  0.8   0:23.00 lsd-import
```

When to consider using LSD?

- Storing simple (but large) catalogs
- Fast cross-matching (two or more) catalogs
- Research requiring (potentially numerous) sweeps through catalogs
- Simple sharing of catalogs over the web, and access to catalogs others have shared

- *"When you need a simple catalog database, but SQLite won't do"*