# Large Survey Database, Continued

# But First

- Homework #1 assignment is in the homeworks/hw1 subdirectory of ast597b repository on GitHub!

**http://nbviewer.ipython.org/github/mjuric/ast597b/tree/master/homeworks/hw1/**

# *Large Survey Database*

- A distributed Python framework for storing, querying, and distributing computation on large datasets
- 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
- Limited ability to do joins
- May have strange, buggy, corner cases
- Only a few active developers

# LSD @ ASTR 597b

- We will primarily be using LSD to extract subsets from large data sets (mostly SDSS, WISE)
- 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.
  - Result: one of the more capable survey archives!

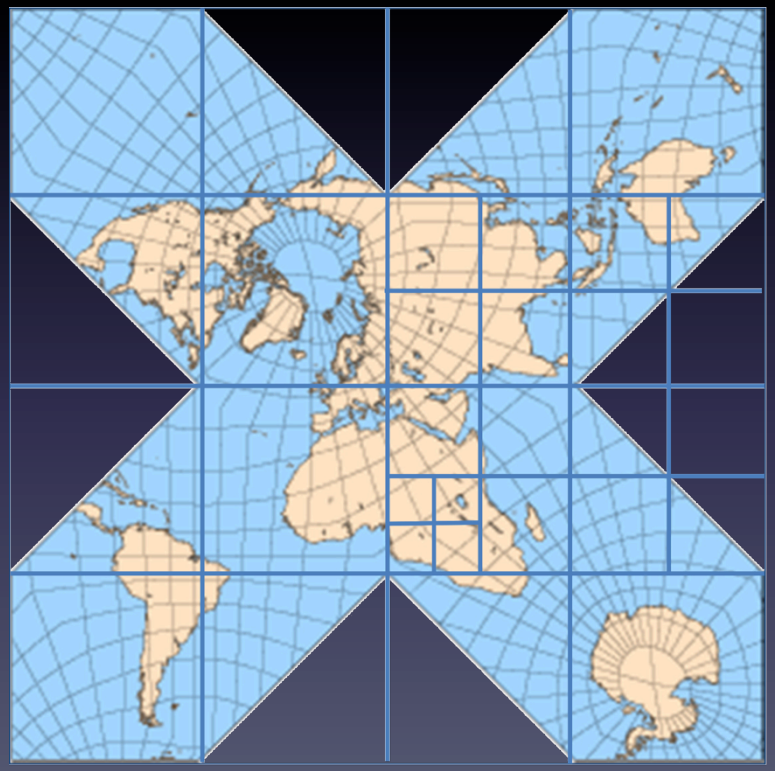
# Architecture & Performance

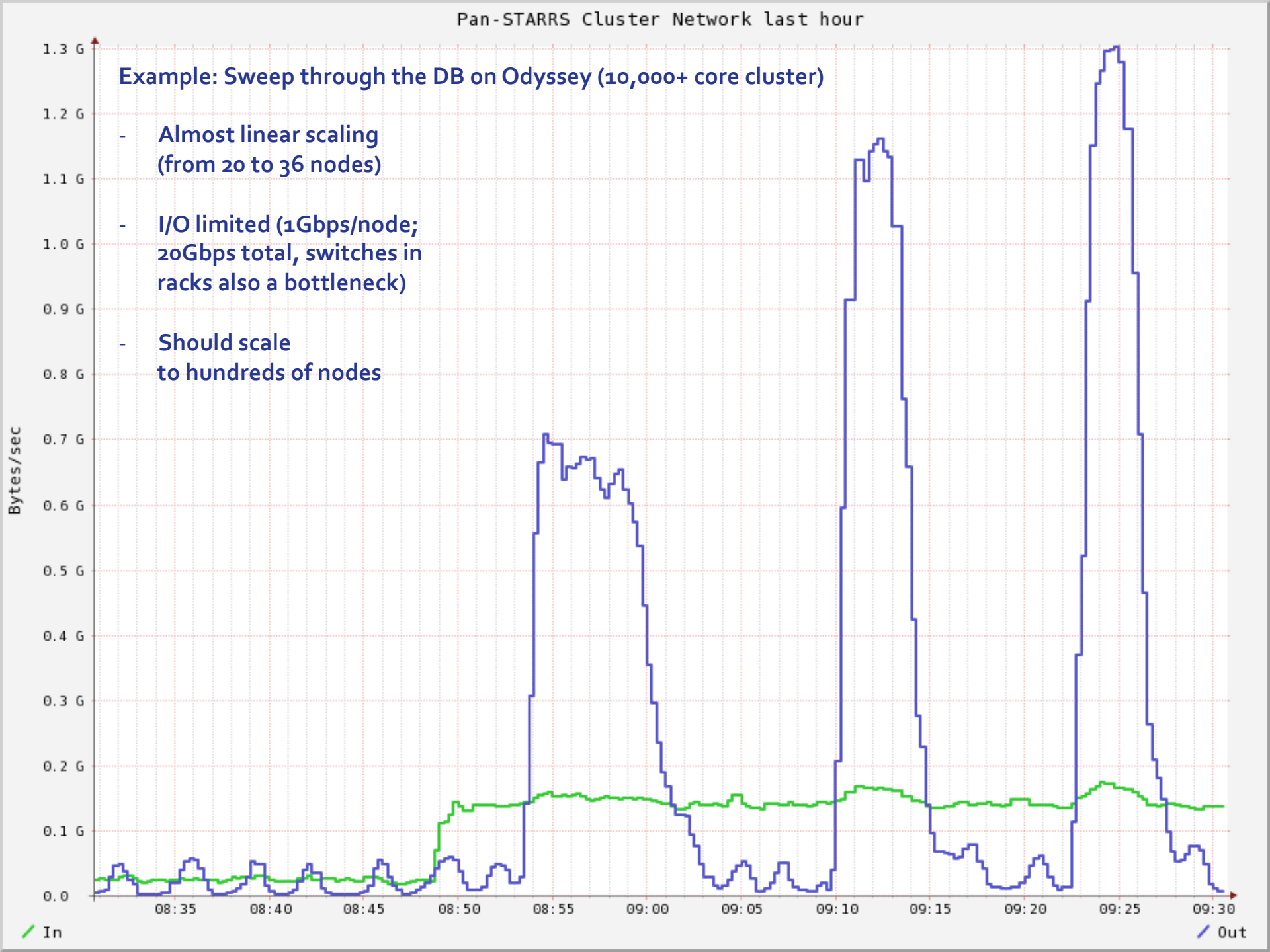
- **LSD is designed to work with simple catalogs; one catalog is one table.**
  - Catalogs with multiple observations are an exception and usually have three tables (one for objects, one for observations, and a "join" table connecting the two)
  - Not optimal, but generally works

- **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*

- **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

# Healpix Projected, Spatially Partitioned Data




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

# Topics

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/2015-02-02-lsd/lecture5.ipynb**

# 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
```

# 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*