Installing Your Own Galaxy

usegalaxy.org/production

Please make sure you have the “WS5” VM

The Team

Galaxy runs out of the box!

- Simple download, setup, and install design:

```
%   hg clone ...
%   sh run.sh
```

- Great for development!
- Not designed to support multiple users in a production environment with default configuration

Development-oriented defaults

- SQLite database
- One process
- Built-in HTTP server
- Local job execution

Workshop Steps

- PostgreSQL
- nginx or Apache
- universe_wsgi.ini customization
- FTP and local data access
- Connect to SGE
- Load balance Galaxy servers
- Load balance Galaxy datasets
- Monitor Galaxy health and view reports

Start Fresh

- Don’t use an old Galaxy installation - check out a new copy
- Use a dedicated non-root user
- Start and stop with your OS’ system service method (e.g. init.d, service)
- Don’t share the database or database user
- Use a dedicated Python or virtualenv
- If you plan to use a cluster, put galaxy in a shared filesystem

PostgreSQL

- SQLite is serverless
- Galaxy is a heavy database consumer
- Locking will be an immediate issue
- Consumes Galaxy server process resources
- Migrating data is no fun

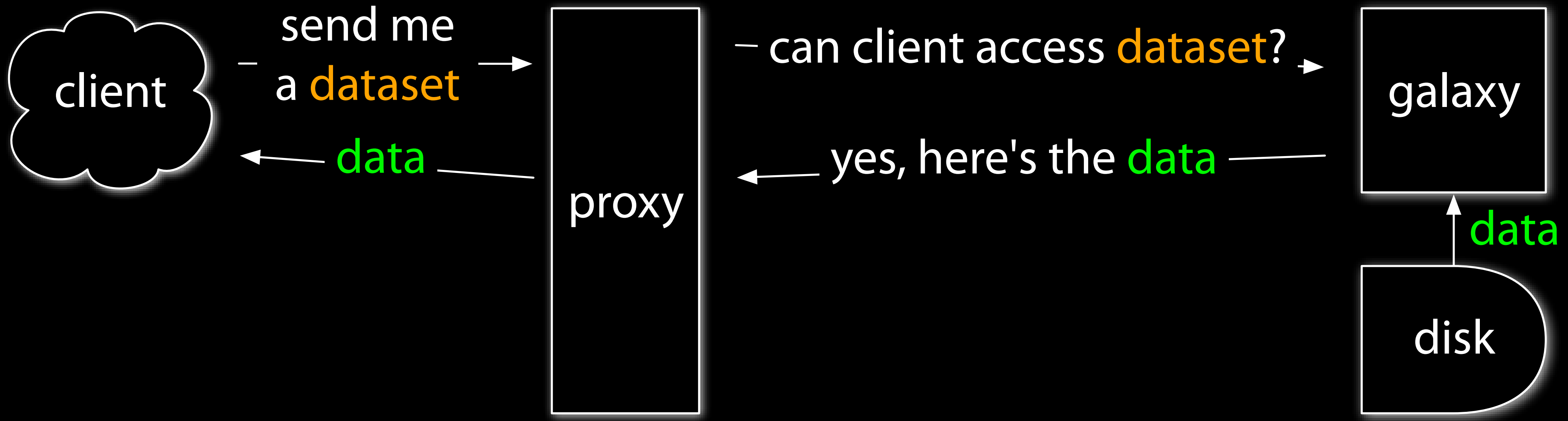
Proxy with nginx or Apache

- Directly serve static content faster than Galaxy's HTTP server
- Reduce load on the application
- Caching and compression
- Load balancing (more on that later)
- Hook your local authentication and authorization system

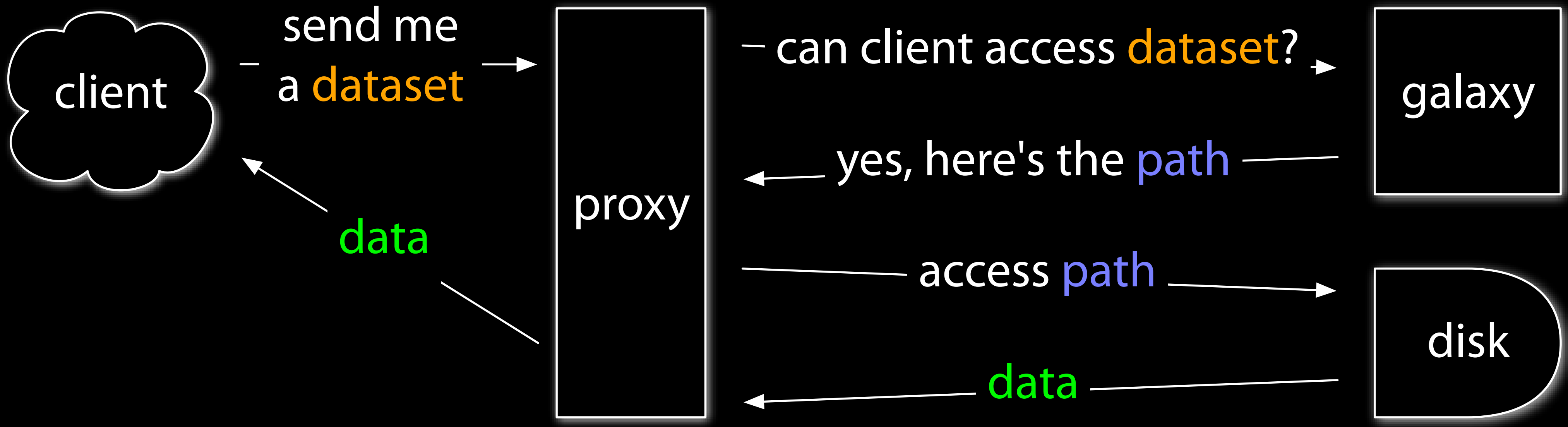
NGINX

APACHE
HTTP SERVER

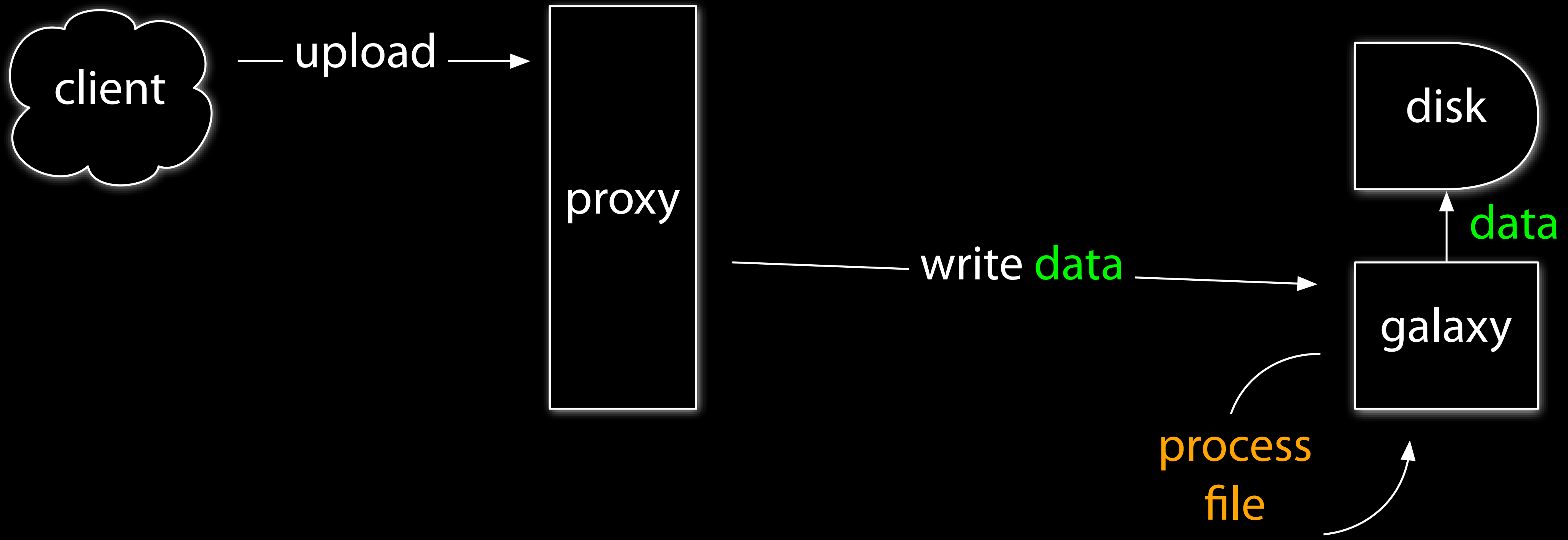
Downloading



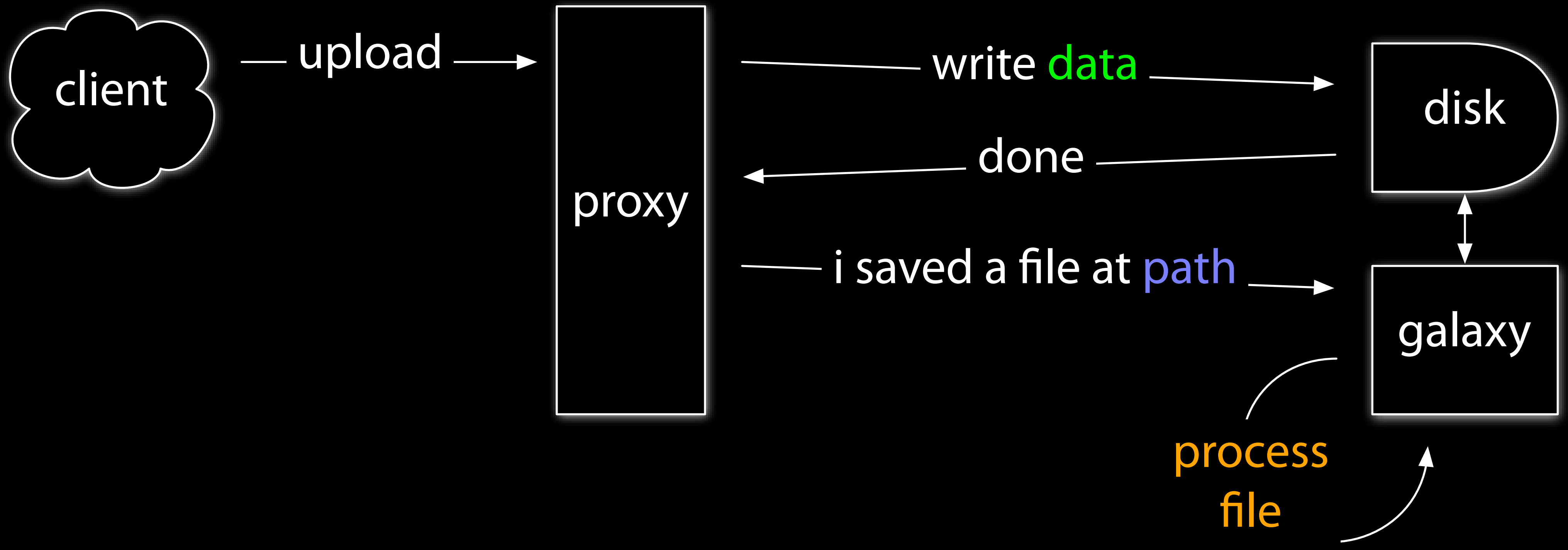
Downloading



Uploading


client
upload
proxy
write data
disk
data
galaxy
process file

Uploading


client
upload
proxy
write data
done
i saved a file at path
disk
galaxy
process file

Alternative uploads

- From the local filesystem to data libraries
- Via FTP (or sftp, scp, cp, ...) to histories

Cluster

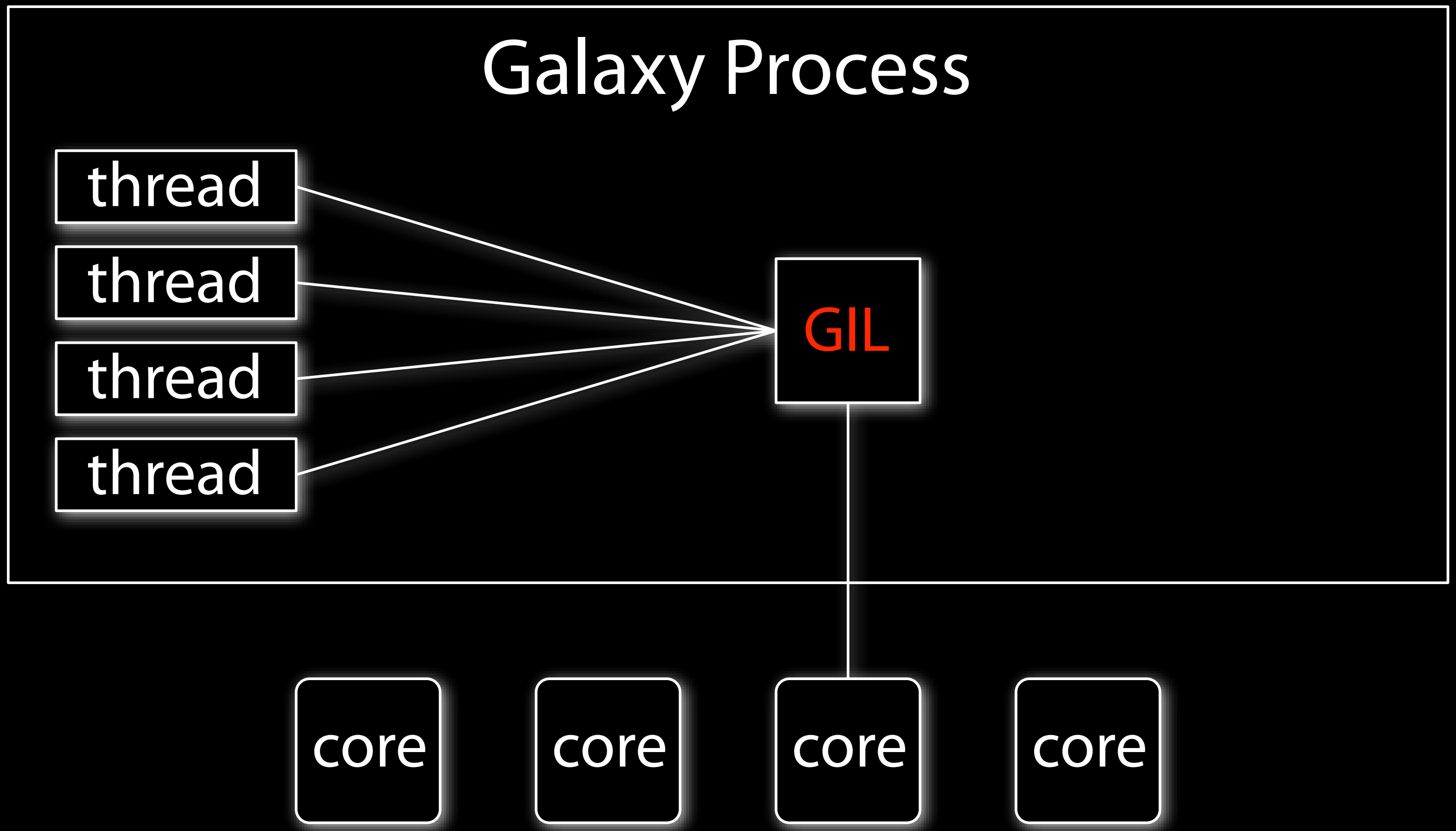
- Move intensive processing (tool execution) to other hosts
- Utilize existing resources
- No job interruption upon restart
- Per-tool cluster options
- DRMAA supports most other DRMs

Job users on the cluster

- By default, jobs run as the user Galaxy is started as
- If your Galaxy users and cluster system users are identical, you may wish to run jobs on the cluster as the actual user
- Galaxy uses sudo to change ownership of relevant files and submit the job to the cluster as the correct system user
- Configurable for your specific environment

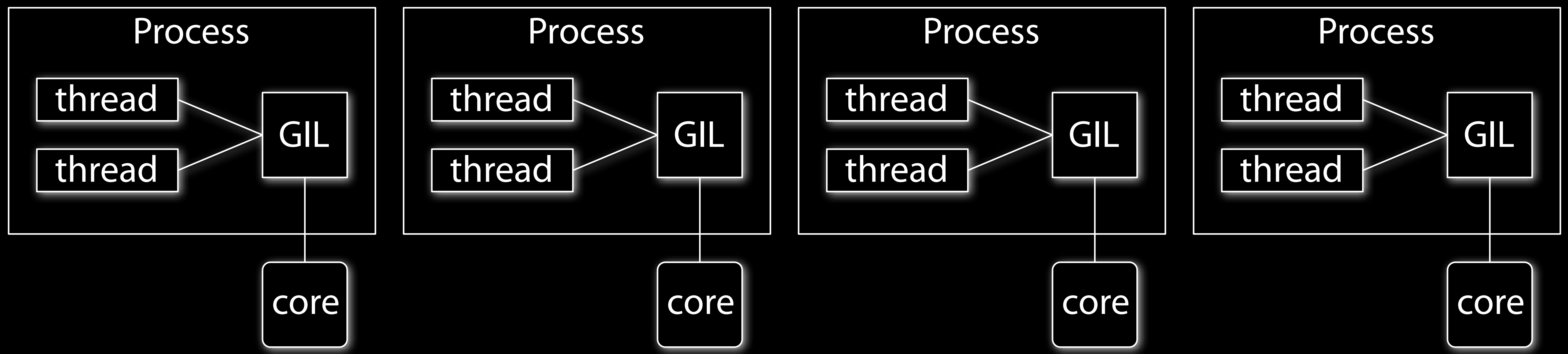
Python and threading

- Galaxy is multi-threaded. No problem, right?
- Problem... Enter the Global Interpreter Lock



- Guido says: "run multiple processes instead of threads!"

Opening the bottleneck



- One job manager - responsible for dispatching jobs to handlers
- Many job handlers - responsible for preparing and finishing jobs, monitoring cluster queue(s)
- Many web servers

Data Management

- The Galaxy philosophy
  - Data is never overwritten
  - Data is never deleted

Data Management

filesystem choices

- Storage can easily be the bottleneck
  - Your storage must scale with your cluster
- Transparent compression and deduplication can reduce usage drastically
- Suggestions
  - ZFS: usegalaxy.org relies on ZFS on Solaris
  - ZFS on FreeBSD stable, Native ZFS on Linux coming
  - Btrfs may be viable soon

Data Management

creating data

- By default, all Galaxy history and library datasets are assigned an ID and stored in `galaxy-dist/database/files/`
- Single directory = single massive filesystem
- Galaxy has a dataset abstraction layer to decouple from a single local filesystem: Object Store
    - Disk backend: single filesystem
    - Distributed backend: multiple filesystems
    - Amazon S3 backend in development

Data Management

cleaning data

- Data is never removed from disk unless
    - `allow_user_dataset_purge = True`
    - users click "delete permanently"
- Solution: cleanup_datasets.py
    - Run from cron to remove data from disk that has been deleted by the user (but not "deleted permanently")
    - Configurable deletion policy allows removal after data has been deleted for a specified number of days

Monitoring

- Monitor Galaxy
  - Provided methods:
    - With cron/email using `galaxy-dist/cron/check_galaxy.sh`
    - With Nagios using `galaxy-dist/contrib/nagios`
  - The provided scripts upload files and run jobs

Collect Statistics with Galaxy Reports

Caching data locally

- Sequences and associated indexes are needed for many tools
- Avoid duplication and wasted time repeatedly building indexes on the same sequences
- Galaxy admin user interface for locally cached data

usegalaxy.org/production