Using Onasys to Supervise On-line Data Analysis

Kipp Cannon, University of Wisconsin Milwaukee

March 26, 2005

http://www.lsc-group.phys.uwm.edu/lscdatagrid/LSCGridCamp/index.html
Condor-based LIGO Data Analysis

- Stand-alone processing tasks are submitted to compute clusters using Condor.
- Complex analysis pipelines are sequenced using DAGMan.
- Has traditionally been performed “off-line”.
- Latencies of several months.

We Want “On-line” Data Analysis

- Low(er) latency for detector characterization.
- Continuous analysis for uninterrupted instrument operation.

On-Line Software Goals

- Latencies of hours or (hopefully) minutes.
- Minimize the developer cost in converting an off-line analysis to an on-line analysis.
- Be robust against software and hardware failures.
- Provide status reports and error notification.
How we do it

- Retain DAGMan as the work flow sequencer.
- User provides a script to generate a DAG that will analyze an interval of data given the start and end time of that data.
- A daemon watches for data, and calls the user’s script as each new data segment becomes available.
- A job monitoring process runs alongside each DAG, periodically updating a MySQL database with the DAG’s state.
- Scripts query the database, and provide a live view of the status of all analyses via a web interface.

So...

- Your first step towards performing an on-line analysis using Onasys is to migrate the execution of your pipeline to DAGMan.
A Caveat

• The software tools being presented here are undergoing rapid development.
• We are using them right now (no kidding), and you can too. However...
  – Things change in response to user needs.
  – Things change in response to facilities needs.
• This is a living software package, not an established static system.
• Please be willing to learn, and maybe re-learn as improvements are made.
The (First) Problem

- DAGMan does not permit dynamic DAGs — a DAG cannot change once its execution begins.
- We do not know in advance what data we want to analyze.

The Solution

- Wait for data to become available.
- Periodically construct DAGs to analyze it in portions.
The Hard Problem

- How do we decide what boundaries to choose for each DAG’s segment?
- There are constraints. Pipelines require the segment:
  - to be longer than some minimum length,
  - to be a multiple of some length,
  - to overlap neighbouring segments by some amount,
  - to be in co-incidence with multiple instruments,
  - and we cannot see into the future.
- But wait, there’s more:
  - Data can arrive “late”.
  - Data can arrive out of order (probably not true for frame files any more).
A Real Example

- A pipeline:
  - requires a minimum of 2048 seconds of data,
  - but it can analyze any quantity of data longer than that in increments of 128 seconds,
  - and adjacent segments should overlap each other by 64 seconds.

- The instrument is currently in science mode, and has accumulated 2560 seconds of data ($= 2048 + 4 \times 128$). Should we launch a DAG? Where should its boundaries be placed?
  - If the instrument loses lock 256 seconds later, it will have proven better to wait because we can’t analyze that 256 seconds by itself but we can if we add it to the end of the data that we already have.
  - But if the instrument stays locked for another 2048 seconds, then not analyzing all the data that we have now will have needlessly delayed the results by an additional 40 minutes.
  - We could choose to analyze just the first 2048 in order to maximize the chances of filling up another 2048 while still getting early results; but this also maximizes the amount that will go unanalyzed if we don’t get another 2048...
The Solution (So Far)

- A daemon wakes up at a configurable frequency, and checks for new segments.
- For each segment that is longer than a configurable minimum, and that has not been previously analyzed, a user-supplied DAG generation script is called.
- The DAG generation script produces a DAG to analyze the segment whose bounds it is given.
- The DAG generation script can tell the daemon how much of the segment it chose to analyze.
- The DAG is launched accompanied by a job monitoring tool that reports back to a database the state of the DAG.
Potential Problems

- “A DAG every ten minutes” might bury us under log files — reliable machine-assisted job monitoring and diagnostics are a must.
- Long-term scalability. Existing setup survived the S4 science run. The next science run can probably be accommodated as well. Will need to develop automated house-keeping tool(s) for continuous running.
Start

Get segment list

segment database

Compute unused segments

used segments

Pause

No

Segments remain?

Yes

Generate DAG for segment

Submit DAG to compute cluster

LIGO-XXXXXXX-XX-X
Notes

- This is the “Analyze all that you’ve got right now” choice.
- You can run nearly *any DAG you want*.
- You can generate the DAG *any way you want*.
- This will all make more sense when you try it out.

On that note, let’s begin...
Download Onasys

- Go to http://www.lsc-group.phys.uwm.edu/daswg/projects/onasys.html
- Download onasys-0.1.1.tar.gz and BossConfig.clad.LL0

Initialize a Proxy Certificate

$ grid-proxy-init
Your identity: /DC=org/DC=doegrids/OU=People/CN=Kipp Cannon 334325
Enter GRID pass phrase for this identity:
Creating proxy ............................................... Done
Your proxy is valid until: Sat Mar 26 11:57:32 2005

Copy Onasys to ldas-grid.ligo-la

$ gsiscp onasys-0.1.1.tar.gz BossConfig.clad.LL0 ldas-grid.ligo-la.caltech.edu:
onasys-0.1.1.tar.gz 100% 2398KB 342.5KB/s 00:07
BossConfig.clad.LL0 100% 1044 1.0KB/s 00:00
Copy your .globus directory to ldas-grid.ligo-la

$ gsiscp -r ~/.globus ldas-grid.ligo-la.caltech.edu:

Log in to ldas-grid.ligo-la

$ gsissh ldas-grid.ligo-la.caltech.edu

• It will be helpful to have two terminals, and log into ldas-grid.ligo-la in both.
Initialize a Local Proxy

$ unset X509_USER_PROXY
$ grid-proxy-init
Your identity: /DC=org/DC=doegrids/OU=People/CN=Kipp Cannon 334325
Enter GRID pass phrase for this identity:
Creating proxy ............................................... Done
Your proxy is valid until: Sat Mar 26 11:57:32 2005

Create BossConfig.clad Link

$ ln -s BossConfig.clad.LLO ~/BossConfig.clad

- The job management tools within the Onasys software package use a MySQL database to track the job states. BossConfig.clad.LLO contains the configuration information for connecting to the job database at LLO. This configuration must be in a file named BossConfig.clad, located in your home directory (or the directory from which jobs will be submitted, but don’t try this).
Unpack, Build, and Install Onasys

$ cd ${HOME}/src
$ tar -xzf ~/onasys-0.1.1.tar.gz
$ cd onasys-0.1.1
$ unset CC CFLAGS CPP CPPFLAGS CXX CXXFLAGS LIBS LDFLAGS
$ ./configure --prefix=${HOME}
$ make && make pdf
$ make install

Install lalapps_power

• lalapps_power is normally distributed as part of LALApps. We are just doing a simple demo here, so...

$ cp /tmp/lalapps_power ${HOME}/bin/

Adjust Your Environment

$ export PATH="${HOME}/bin:${PATH}"
$ export PYTHONPATH="${HOME}/lib/python2.3/site-packages:${PYTHONPATH}"
Test Your Installation

$ boss showSchedulers
dagman
$ echo "import onasys.onasysd" | python
$ which lalapps_power
~/bin/lalapps_power

- Note that the second command should generate no output.

Setup an On-Line Analysis

- We will run the excess power burst search example that ships with the release:

  $ mkdir online.test
  $ cd online.test
  $ cp ${HOME}/share/onasys-0.1.1/examples/power/* .
Examine The Files

- The `onasysd` daemon allows the user to define a “data directory” in which files required by the DAG generation script can be placed.
- `${HOME}/online.test` is our data directory.
- We will generate DAGs from a “template DAG”: a pre-constructed DAG will have the start and stop times of each segment pasted into it.
- The files for the template DAG can be found in this directory:
  - `datafind.sub` describes an `LSCdataFind` job,
  - `power.sub` describes a `lalapps_power` job,
  - and `pipeline.dag` describes the order in which they must be run.
- The DAG generation script, `mkdag.sh`, can be found in this directory.
- The `onasysd` daemon’s configuration file, `onasysd.ini`, can be found in this directory.
The Template DAG

- Our DAG consists of two nodes: an LSCdataFind job followed by a lalapps_power job.
- Examine datafind.sub, and notice that:
  - We are locating frame files of type “R”. These are “raw” frame files — level 0 data fresh from the instrument.
  - We are searching for files located in the directory /frames on localhost.
- Examine pipeline.dag, and notice that:
  - the start and stop times appear as the strings GPSSTARTTIME and GPSSENDTIME respectively. Our DAG generation script will replace these with the real start and stop times of the segment.
The DAG Generator Script

- Examine `mkdag.sh`, and notice that:
  - It is a simple bash script.
  - Information about the segment to be analyzed is passed as command-line arguments.
  - All it really does is copy all `.sub` files and the `.dag` file, doing a search-and-replace using `sed` on the `.dag` file.
- The daemon will create a fresh directory for each DAG, and the DAG generator script is invoked in that directory.
- The parts of the template DAG are found using the “data directory” path, passed to the DAG generator as a command line argument.
The Daemon Configuration File

- Examine `onasysd.ini`, and notice that:
  - A start time, or initial epoch, is specified.
  - An update interval is specified.
  - You can choose the parameters to pass to LSCsegFind.
  - The DAG generator is identified.
  - A “base name” is set — this is widely used by the daemon as a tag for this search, so it should be set to something specific and descriptive.
  - We can choose a minimum segment length, and an overlap.
Generate a DAG by Hand

$ mkdir -p onasysd.jobs/test
$ cd onasysd.jobs/test
$ ${HOME}/online.test/mkdag.sh -s 795708612 -e 795708700 -a 795708612 -b 795708700 -f power_A4L1.795708612_795708700.dag -t ${HOME}/online.test

Launch the DAG by Hand

$ boss_submit_dag power_A4L1.795708612_795708700.dag
$ boss_q

• NOTE: the DAG is submitted using boss_submit_dag rather than condor_submit_dag.
• The “Boss” utilities are a set of tools that wrap the Condor utilities, and present a view of your jobs at the DAG level rather than at a job-by-job level.
• Why use these tools to submit your DAG? Go to

https://ldas.ligo-la.caltech.edu/onasysd/onasysd.cgi
### Onasys Monitor — Hanford

#### Today's Jobs

<table>
<thead>
<tr>
<th>Date</th>
<th>Count</th>
<th>Successful</th>
<th>Failed</th>
<th>Running</th>
<th>Unscheduled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 13</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 12</td>
<td>345</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 11</td>
<td>345</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 10</td>
<td>362</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 09</td>
<td>258</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 08</td>
<td>268</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 07</td>
<td>382</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 06</td>
<td>346</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 05</td>
<td>344</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 04</td>
<td>362</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 03</td>
<td>287</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 02</td>
<td>259</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 01</td>
<td>229</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 28</td>
<td>334</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 27</td>
<td>329</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 26</td>
<td>354</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 25</td>
<td>341</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 24</td>
<td>248</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 23</td>
<td>141</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 22</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Average

- Total Jobs: 286.0
- Successful: 237
- Failed: 15
- Running: 4
- Unscheduled: 1

---

The Onasys Monitor is used to track the status of jobs over a specified period, showing the number of successful, failed, running, and unscheduled jobs. The dashboard provides a detailed summary for the last 20 days, allowing for monitoring and analysis of job execution patterns at the Hanford site.
Job Summary for: Sun Mar 13, 2005

<table>
<thead>
<tr>
<th>ID</th>
<th>SID</th>
<th>Host</th>
<th>User</th>
<th>Path</th>
<th>Submit Time</th>
<th>Start Time</th>
<th>Stop Time</th>
<th>Exec</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5733</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp/2445779379475567344</td>
<td>17:46:40</td>
<td>17:46:45</td>
<td>0</td>
<td>power_54SH2.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5732</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
<tr>
<td>5731</td>
<td>174506</td>
<td>ligo-grid</td>
<td>kipp</td>
<td>/dso-test/kipp1/2445779379475567344</td>
<td>17:46:33</td>
<td>17:46:33</td>
<td>0</td>
<td>power_54SHL.794799037_794799744.dag</td>
<td>R</td>
</tr>
<tr>
<td>5730</td>
<td>174506</td>
<td>ligo-grid</td>
<td>dbrown</td>
<td>/dso-test/dbrown/projects/dso47924539_2379470680</td>
<td>17:45:15</td>
<td>0</td>
<td>0</td>
<td>ligo_51SHL.7947970928_794799980.dag  </td>
<td>R</td>
</tr>
</tbody>
</table>
Onasys Monitor — Hanford

Summary of DAG for Job #5733

<table>
<thead>
<tr>
<th>Name</th>
<th>Submit Time</th>
<th>Start Time</th>
<th>Stop Time</th>
<th>Status</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>lalapps_power</td>
<td>2005-03-13 17:46:59.00</td>
<td>2005-03-13 17:47:17.00</td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>LSCdataFind</td>
<td>2005-03-13 17:46:47.00</td>
<td>2005-03-13 17:46:47.00</td>
<td>2005-03-13 17:46:48.00</td>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>publish</td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
</tr>
</tbody>
</table>

LSCdataFind → lalapps_power → publish

No ImageMap. Sorry. Graphviz version is too old. Or something.
Onasys Monitor — Hanford

Node lalappp_power / Job # 5733

- Job ID: 2733
- Name: lalappp_power
- Submit Time: 2005-03-13 17:46:59.00
- Start Time: 2005-03-13 17:47:17.00
- Stop Time: None
- Return Code: None
- Submit File: power_pipe.power.34.sub
- Std Output File: logs@power.out
- Std Error File: logs@power.err

Parent Nodes
- LSC-lalappp

Child Nodes
- publish
Onasy Monitor — Hanford

Node dc0502318bcdf83a789809a184eb26 / Job # 5630

Job ID: 5630
Name: dc0502318bcdf83a789809a184eb26
Submit Time: None
Start Time: None
Stop Time: None
Return Code: None
Submit File: online.sinae.sub
Std Output File: logsinae-$(macropostime)-$(macropsetid)-$(cluster)-$(process).out
Std Error File: logsinae-$(macropostime)-$(macropsetid)-$(cluster)-$(process).err

Parent Nodes

Child Nodes

Done
Job Monitoring Flow-chart
Job Monitoring Flow-chart

- `dagman`
- `pipeline.dag.condor.sub`
- `datafind.sub`
- `LSCdataFind`
- `pipeline.dag`
- `lalapps.power`
- `power.sub`
Job Monitoring Flow-chart

- dbUpdater.sub
- dagdbUpdater
- datafind.sub
- LSCdataFind
- pipeline.dag
- lalapps.power
- pipeline.dag.condor.sub
- power.sub
- dagman
- pipeline.log
Job Monitoring Flow-chart

- Database
- dbUpdater.sub
- dagdbUpdater
- dagman
- pipeline.dag
- power.sub
- lalapps.power
- pipeline.dag.condor.sub
- datafind.sub
- LSCdataFind
- pipeline.log
Job Monitoring Flow-chart

- Database
  - SuperDag.dag
    - dbUpdator.sub
  - DataMan
    - pipeline.dag
    - lalapps.power
    - power.sub
    - datafind.sub
    - SuperDag.log
- SuperDag.dag
  - LSCdataFind
  - pipeline.log
  - pipeline.dag
  - dagdbUpdater
  - dagman
  - dagdbUpdater
  - dbUpdator.sub
  - pipeline.dag.condor.sub
  - datafind.sub
  - power.sub
  - pipeline.log
Now Automate It

- Return to the main directory
  $ cd ~/online.test

- Last-minute check of the onasysd daemon’s config file
  $ vi onasysd.ini

- Set the initial_epoch to the output of
  $ tconvert now

- Set dryrun to False.

- Start the daemon.
  $ onasyd

What’s Going On?

- It is useful to refer to the flow chart on page 10.
How to terminate the Daemon

- The daemon can be terminated by identifying the process ID of the daemon and sending it SIGTERM:
  
  ```
  $ kill -s SIGTERM 18916
  ```

- The process ID is printed in parantheses near the start of each line of log output.

Where to Find More Information

- The Onasys project’s homepage is at
  
What’s Ahead?

- Different segmentation algorithm: favour completeness over low latency.
- Secondary analysis: provide one analysis’ output for use as input to another.
- Automatic house-cleaning: delete old job directories, and old database entries.
- Switch to Quill and, later, CondorDB.