**THE LOGIC BEHIND THE CODE (by Mihael Hategan)**

1. We need the ability to quickly plot large ranges of data (i.e., spanning 4 years). The original LIGO data samples instruments at specific intervals (most often at 1s or 1m). If we were to scan through all the data to plot something, that would take a lot of time (imagine how many seconds or minutes there are in 4 years).
2. We could just sample the data, but that has the potential to miss things. For example, if we want 1000 data points, over 4 years, we read a sample every 1.46 days, but that means we can miss earthquakes that happen between two samples, so that's not a good strategy. We want mean or RMS values over that whole 1.46 day period.
3. So **we store the sum of all values in a channel up to the current time.** So when we want an average from a to b, we read sum(b), sum(a) and then the average is (sum(b) - sum(a)) / (b - a). In other words, we don't integrate over a period, but we store the integral at all points, so then to calculate the integral from a to b we only need to look up the endpoints. As you can see, with this scheme we can plot our entire data span (or any other span) almost instantly.
4. **But this needs the data to be converted.** The original LIGO data is stored in some proprietary format that can be extracted using ligo-tools. It's a two-step process:
	1. **extract channel data using ligo-tools (this simply dumps binary data from ligo data) and**
	2. **convert data to our stream format.**
5. The stream format is made of records of the following format: [valid, timestamp, sum, sumSquares].
6. .rms files don't really exist. The code looks at the last part of the file name (we call it data path) and if **it's "mean" it returns (sum(b) -sum(a)) / (b - a),** and if it's **rms, it returns sqrt((sumSquares(b) - sumSquares(a))/(b - a)).**

**SERVER SETUP AND PROCESSES RUNNING:**

1. Files are in data4.i2u2.org:
	1. /disks/i2u2/ligo/data/streams
		1. This folder has the converted data
	2. /disks/i2u2/ligo/data/frames/trend
		1. This folder has frames that were rsync’ed from Caltech until 2011.
	3. /disks/i2u2/ligo/data/frames/trend\_after23April2013
		1. This folder has frames that have been rysnc’ed since April 2013.
2. The following cronjobs runs nightly:

# LIGO data import

0 0 \* \* \* rsync -a --password-file=/home/quarkcat/.rsyncpw i2u2data@terra.ligo.caltech.edu::ligo/trend/second-trend/ /disks/i2u2/ligo/data/frames/trend/second-trend

0 0 \* \* \* rsync -a --password-file=/home/quarkcat/.rsyncpw i2u2data@terra.ligo.caltech.edu::ligo/trend/minute-trend/ /disks/i2u2/ligo/data/frames/trend/minute-trend

# LIGO data conversion

30 0 \* \* \* /usr/local/ligotools/i2u2tools/bin/ImportData /disks/i2u2/ligo/data/frames/trend\_ after23April2013 /usr/local/ligotools/ligotools /disks/i2u2/ligo/data/streams

The two first instructions rsync the second-trend frames and the minute-trend frames from Caltech.

The last instruction converts the frames into our streams.

1. Our conversion code:
	1. ligo/src/java/gov/fnal/elab/ligo/data/convert
	2. If the code needs to be fixed or updated, then we need to install and compile this code in data4.i2u2.org. This code **does not** get deployed with a rollout.
	3. Login to data4.i2u2.org and go to /home/quarkcat/tools/ligo-install
	4. Run script **installdatatools.sh.** This script checks out the code from branches/3.0-test. Make sure your changes are in this branch before running the script.
	5. The cronjob will now run the compiled code that night.
2. How the streams are served to the website:
	1. By using a REST python server: DataServer.py (which is inside the streams folder).
	2. This python server needs to be running at all times.
	3. If LIGO doesn’t display any plots, then we need to check if the process is running at data4 (ps –ef | grep DataServer.py)
	4. Recently added a check in the cronjob. If the process is not running, it will start it. See /home/quarkcat/tools/ligo-check-dataserver/check-start-process.sh.