Reproducible and automated processing in high throughput NGS facilities

Gianmauro Cuccuru, Giorgio Fotia, Josh Moore, Luca Lianas, Luca Pireddu, Jason Swedlow, Gianluigi Zanetti

CRS4
Wellcome Trust Centre for Gene Regulation and Expression – University of Dundee

July 1st, 2013
Motivation
A public multi-disciplinary research center in Italy
Focuses on applied computational sciences
Within top 5 Italian computation facilities

Currently the largest sequencing center in Italy
Has enabled a number of studies on the Sardinian population

**Sequencing Equipment:** 3 Illumina HiSeq2000, plus older sequencers
**Sequencing Capacity:** about 5 Tbases/month
Since Sept. 2010 we’ve sequenced about...

- over 2000 whole-genome samples (low-pass, high-coverage)
  - some cancer genomes as well
- 800 total RNA samples
- 100 exomes
- a handful (30) of ChIP-Seq samples
Scalability problems

- The number of samples and the amount of data to handle presented significant difficulties
  - difficulties scaling computational throughput
  - difficulties tracking the process
  - ample opportunities for inefficiencies
The number of samples and the amount of data to handle presented significant difficulties
- difficulties scaling computational throughput
- difficulties tracking the process
- ample opportunities for inefficiencies

**Wishlist**

We wanted to improve our process in several ways:
- automated processing
  - hands off from when the sequencer is started to deliverable data
- trace all data processing activities
- effectively manage file storage
- computational scalability
Our solution
Our solution

Automated processing and tracking platform

To satisfy those requirements, we implemented a solution based on five core components:

- Galaxy
- the “Automator”
- OMERO.biobank
- iRODS
- Hadoop
**Requirements**

**Requirement**

Automated processing

- Monitor sequencers, detect when new data is ready
- Automatically run data through standard pipelines
- Notifications to operator for data ready and for errors
- Automatically track these data sets and how they were generated

In part implemented with Galaxy; in part with custom software
Since you’re at GCC, I can probably assume you know what Galaxy is!

Key features for our application

**Workflows:** give a way to define automated analysis “recipes”

**Histories:** saves sequence of tool invocations that produced a data set
- A convenient way to trace reproducible actions performed on data

**REST API:** provides some degree of programmatic access
- e.g., launch workflows, retrieve results

**Familiarity:** Galaxy was a desirable and familiar tool for our users

One tool for both automation and downstream analysis
In our system, we use two Galaxy instances

**Public instance**

- Collect sample data and flowcell configuration from wet lab and/or client
- Return processed samples to users through data set libraries (WIP)
  - Possibly through integration with iRODS
- Currently runs a customized version of the nglims Galaxy fork by Brad Chapman (Harvard School of Public Health Bioinformatics)
Private instance

- Manages execution of our standard processing workflows
- Accumulates processing history for each flowcell and sample
  - We fetch this information through the Galaxy REST API (using the bioblend Python module by Afgan et al.)
- The private instance is a standard version of Galaxy
Requirements

**Requirement**

**Automated processing**

- We found Galaxy alone to be insufficient for full automation
- Clumsy for housekeeping tasks
  - e.g., move/rename files, interact with other services
- Workflows sometimes aren’t sufficiently expressive
  - e.g., linking to variable number of outputs
  - no “if” operator
Our custom Automation package

- Part interfaces to programmatically control other components
  - i.e., Galaxy, iRODS, OMERO
- Part distributed event-dispatching daemon based on RabbitMQ
- Part custom-made event handlers
  - These use the aforementioned interfaces and anything else they need to implement actions, that may emit new events

Task division

- Galaxy handles all operations that transform or create datasets
  - allows us to easily create a history
- The Automator does all other operations, including driving Galaxy
Requirements

Requirement

Trace all data processing activities

- Essential for reproducibility
- For any data set generated, track how it was created
  - Actions on data sets
- Track relations between data sets
- Database should support appropriate queries; e.g.,
  - From what flowcell was the dataset derived?
  - Through which operations/parameters?
  - With which other samples was it normalized?
  - What other data sets came from the same batch?
OMERO is a “model-driven data management platform for experimental biology” (Allan, et al.; Nature Methods, 2012)

Stores a graph structure where data set nodes are connected by actions

Nodes and actions are tagged with model-dependant information

We extended OMERO to handle data types produced in sequencing and microarray experiments

OMERO.biobank – will be included in official OMERO releases

In our model, we store information about samples, data paths, data format, and the data set’s *entire processing history*
OMERO

Fig: An example of an OMERO graph
Requirements

Requirement

Effectively manage file storage

- We generate lots of files
- Incremental growth → multiple file systems
- Geographically dispersed collaborations
A file cataloguing system

- We have used it to create a single go-to place to find data files
  - Simplifies accessing data on complex storage architecture

- Allows us to tag files with attributes (e.g., run id, sample id, etc.) and use the tags in queries

- Optimized file transfers
Requirements

**Requirement**

**Computational scalability**

- Large data generation rate from sequencers
- Interest in minimizing turn-around time
- Need to scale out computation over many nodes
Hadoop

- The system that enables many data-centered companies
  - e.g., Twitter, Facebook, Yahoo, ...
- Automatically handles:
  - distribution of computation and data
  - node and task failures
- To leverage Hadoop in this context, we adopted:
  - Seal: toolkit for Hadoop-based sequencing data processing
    - demultiplexing, alignment (based on BWA, sorting, etc.)
  - Pydoop: Python API for Hadoop
    - A dependency for Seal, but also used for custom tools and scripts
  - SeqPig: SQL-like scripting for Hadoop with sequencing-specific functionality
We’ve implemented a thin integration between Galaxy and Hadoop
Can run Hadoop-based programs via Galaxy
Big Hadoop datasets referenced by Galaxy though a “pointer” type, *pathset*
  - a file containing a list of paths
Hadoop jobs executed through a wrapper that knows how to interpret pathset files and passes the correct arguments to the Hadoop job
Galaxy dataset clean-up program also has to learn about pathsets (WIP)
Automation example: new run registered

Bringing it all together...

Hadoop

Galaxy

OME biobank

iRODS Integrated Rule-Oriented Data System
Conclusion
Currently...

- The system is currently being used to process the sequences produced by our center.
  - is able to process flowcells in complete automation
- Through the stored histories, we ensure reproducibility
- Development is ongoing to improve it
- Is proving to be a good solution to our original problem
Future work

- Better management and monitoring
- Better error handling and restart
Future work

- Better management and monitoring
- Better error handling and restart

Open source?

- Some parts already are: https://github.com/crs4
  - Omero.biobank
  - Seal, Pydoop
- Hopefully by the fall we’ll also release:
  - the automator
  - Hadoop-Galaxy integration
A shameless plug...  
- Public Galaxy instance for NGS microbiology data by CRS4  
- If you’re interested, go see the poster!

http://orione.crs4.it
A shameless plug... 

- Public Galaxy instance for NGS microbiology data by CRS4 
- If you’re interested, go see the poster!

http://orione.crs4.it

Thanks for your attention!