OMERO.biobank:
a flexible approach for managing data in experimental biology

Ciclo seminari interni CRS4 2012

23 maggio 2012

Luca Lianas

lianas@crs4.it

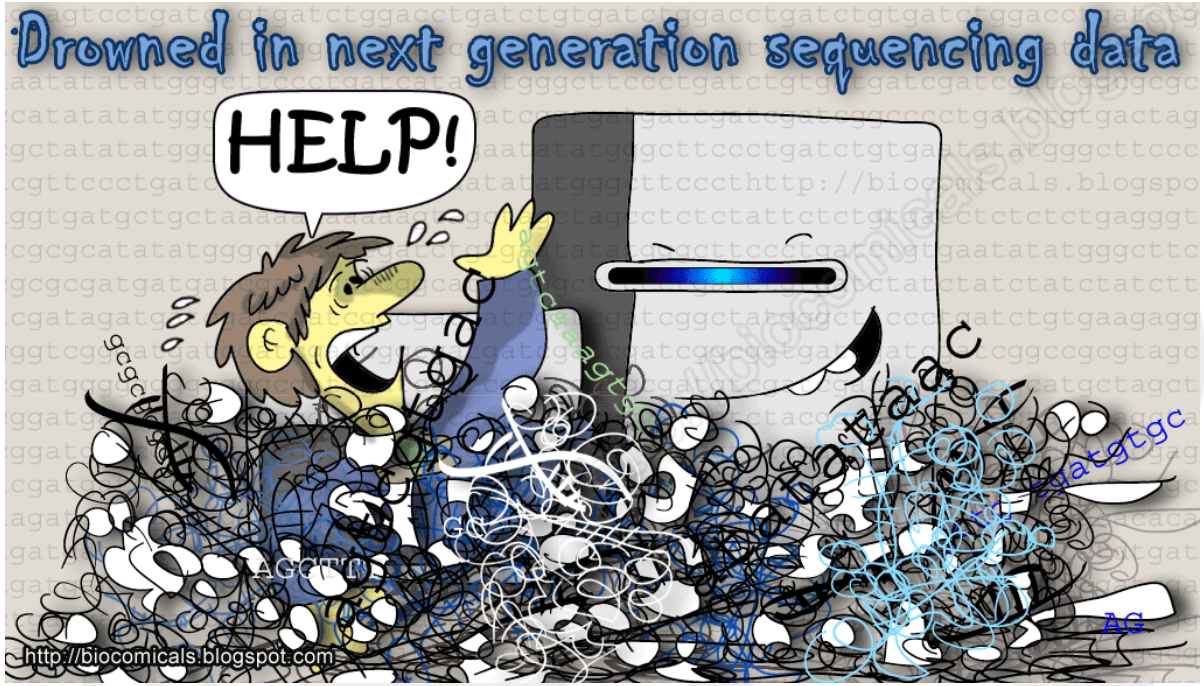
Agenda

- **Introduction**
- **OMERO**
  - What does it do
  - How does it do it
  - Why do we like OMERO
- **OMERO.biobank**
  - What is a “computable” biobank
  - OMERO.biobank overview
  - Code examples
- **Conclusions**

Agenda

- **Introduction**
- **OMERO**
  - What does it do
  - How does it do it
  - Why do we like OMERO
- **OMERO.biobank**
  - What is a “computable” biobank
  - OMERO.biobank overview
  - Code examples
- **Conclusions**

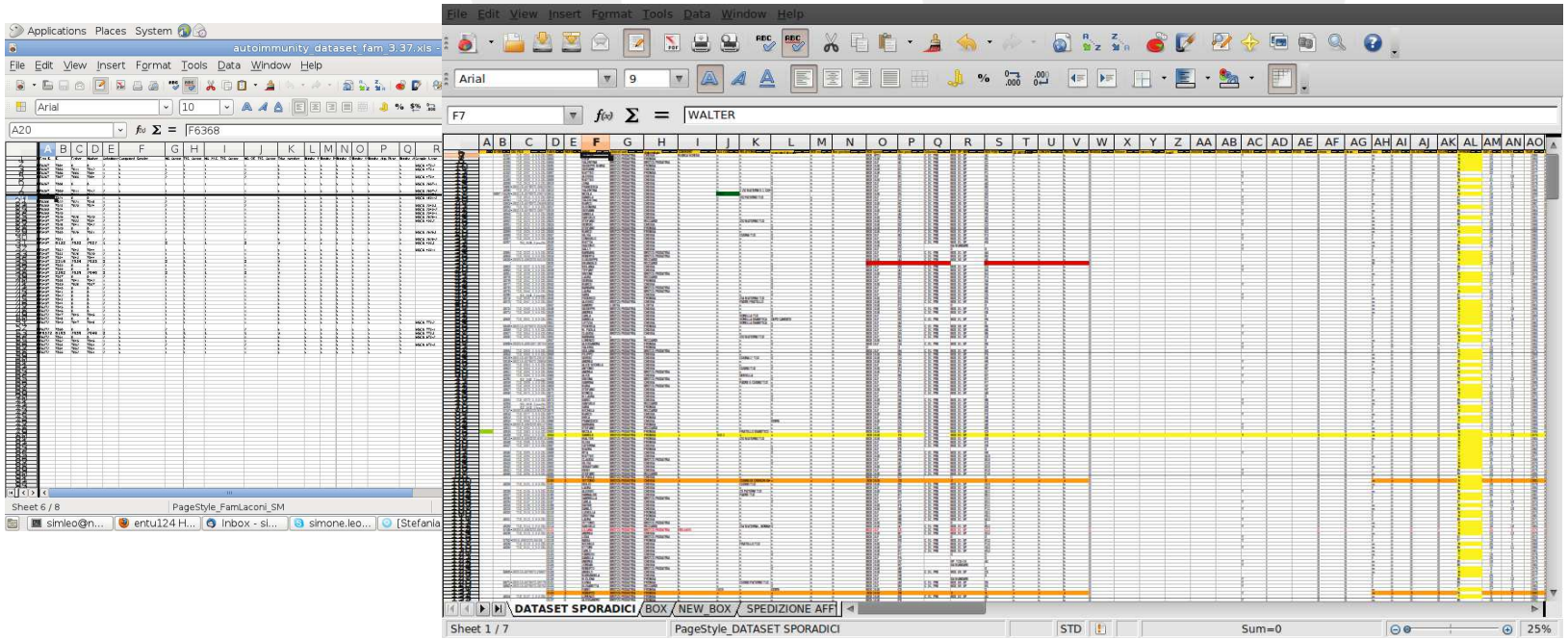
Coping with huge amounts of data



- High throughput technologies
- Heterogeneous sources
- Continuously evolving data acquisition technology

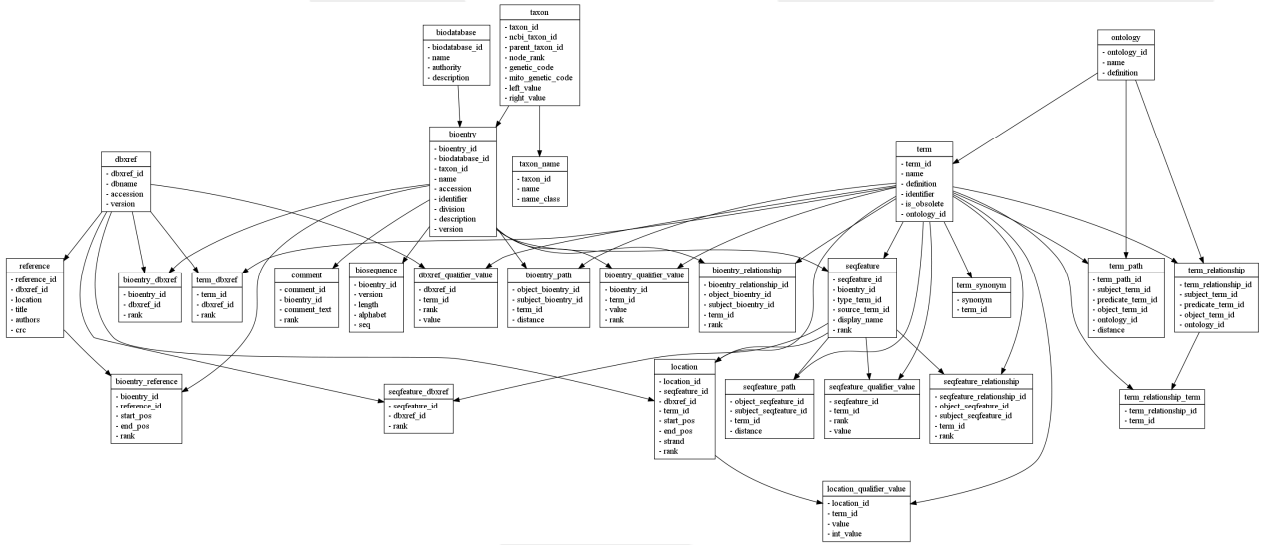
How to handle Data: the standard approach

- The “default” solution:
  - Excel files
    - Pros: instant-on, everybody knows how to handle Excel files
    - Cons: everybody **thinks** they know how to handle Excel files, brittle, uncontrollable, does not scale (in any sense)



How to handle Data: the standard approach

- The “do it from scratch” approach:
  - Custom databases
    - Pros: structured approach, can evolve, can scale in size
    - Cons: hard to maintain, need to develop middleware and interfaces, does not scale with project evolution



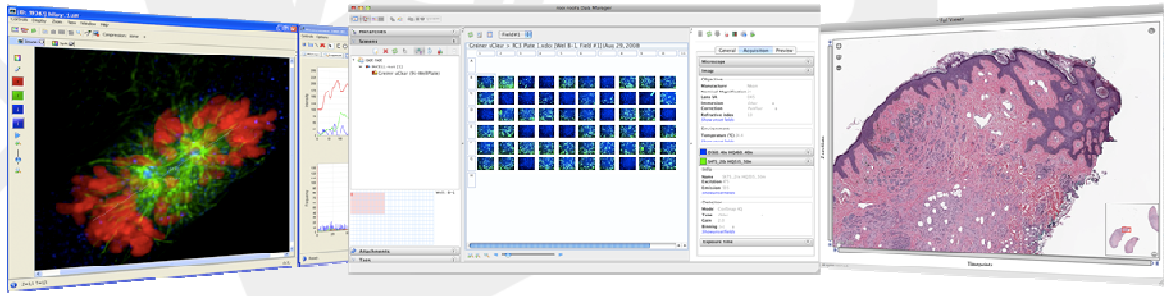
- Maybe another strategy is needed?

Agenda

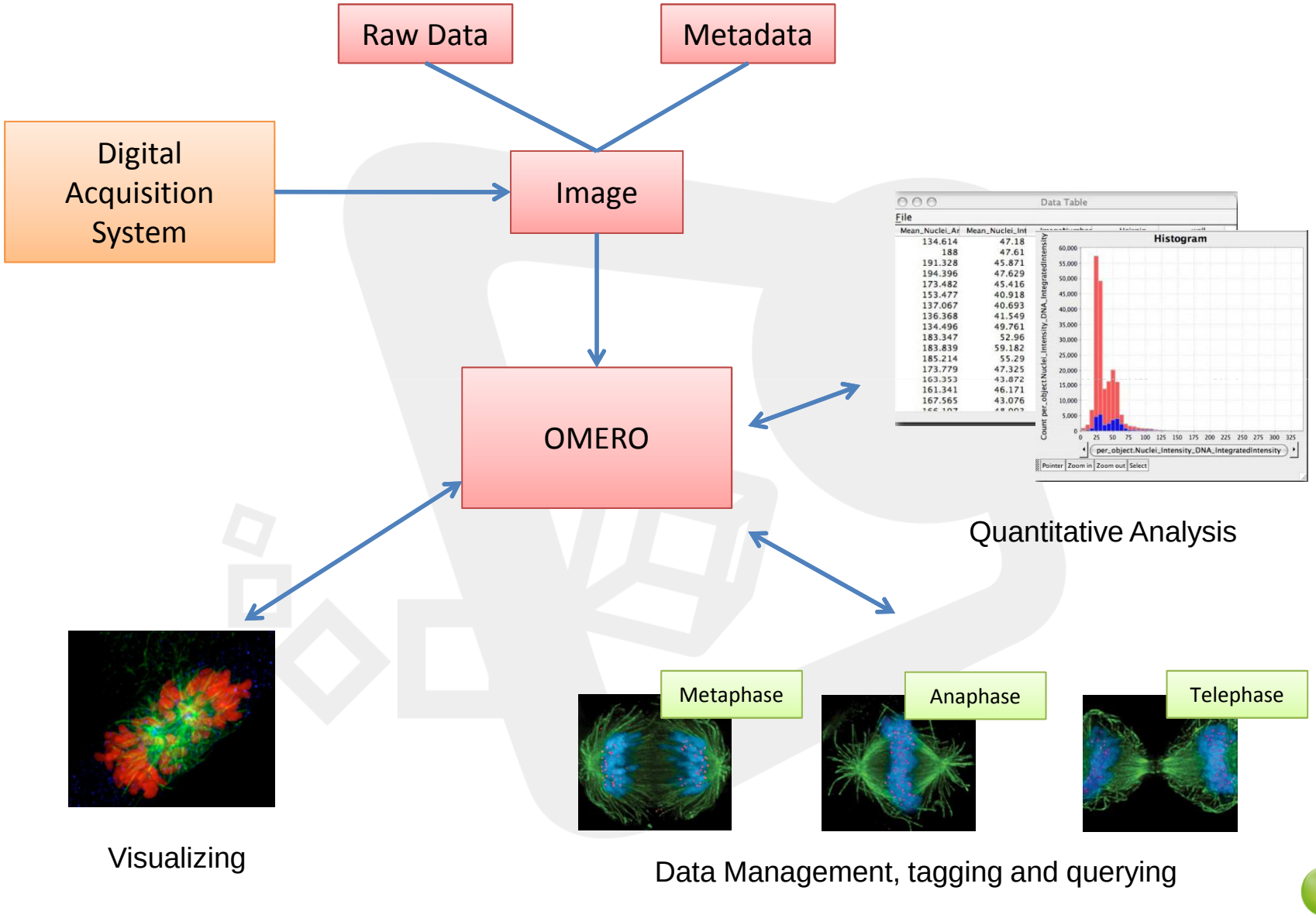
- Introduction
- **OMERO**
  - What does it do
  - How does it do it
  - Why do we like OMERO
- OMERO.biobank
  - What is a “computable” biobank
  - OMERO.biobank overview
  - Code examples
- Conclusions

OMERO: a flexible approach for managing data in experimental biology

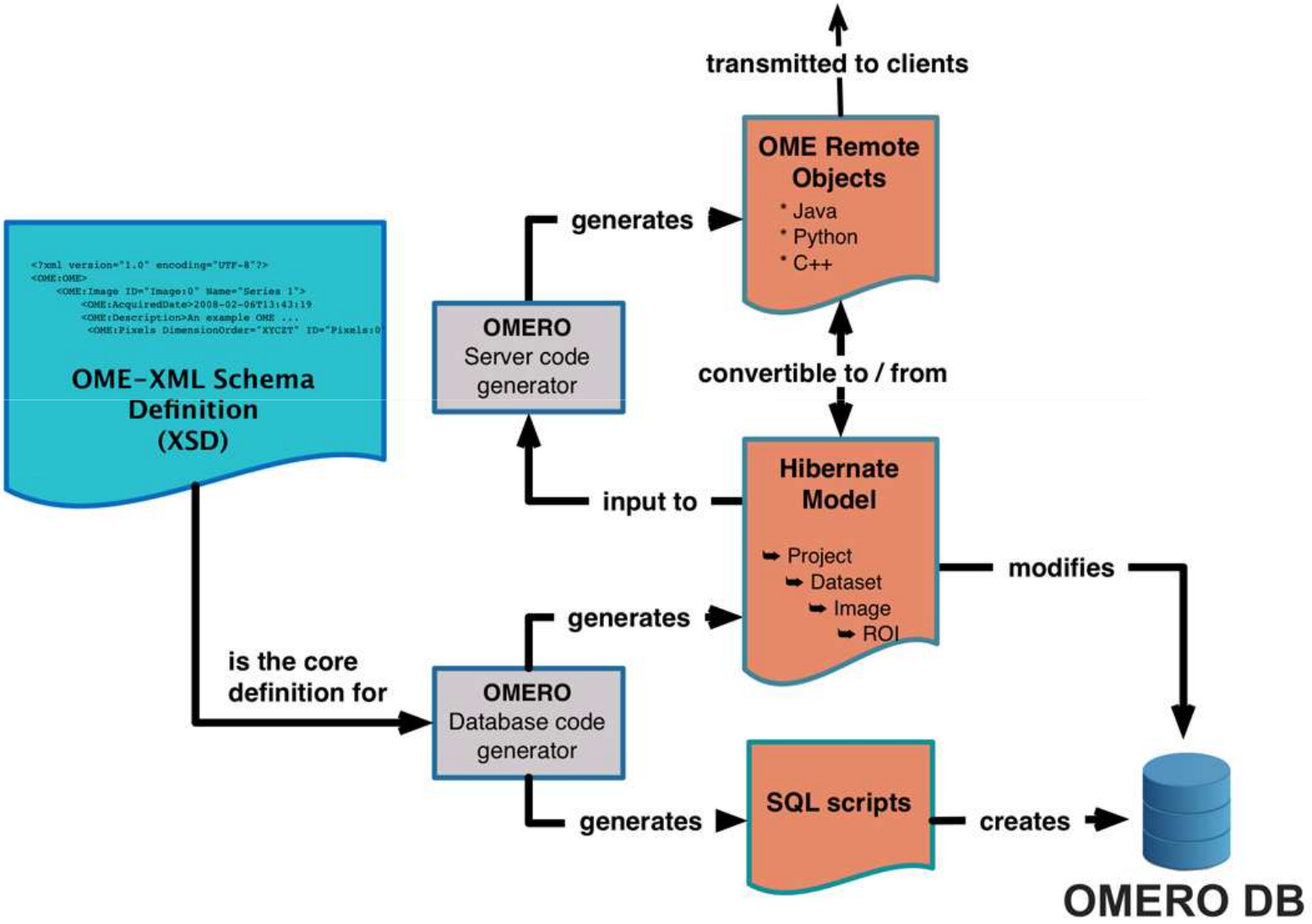
- Client-server software for visualization, management and analysis of biological microscope images (http://www.openmicroscopy.org/site)
- Developed by the Open Microscopy Environment Consortium (University of Dundee, Glencoe Software, Harvard Medical School, LOCI)



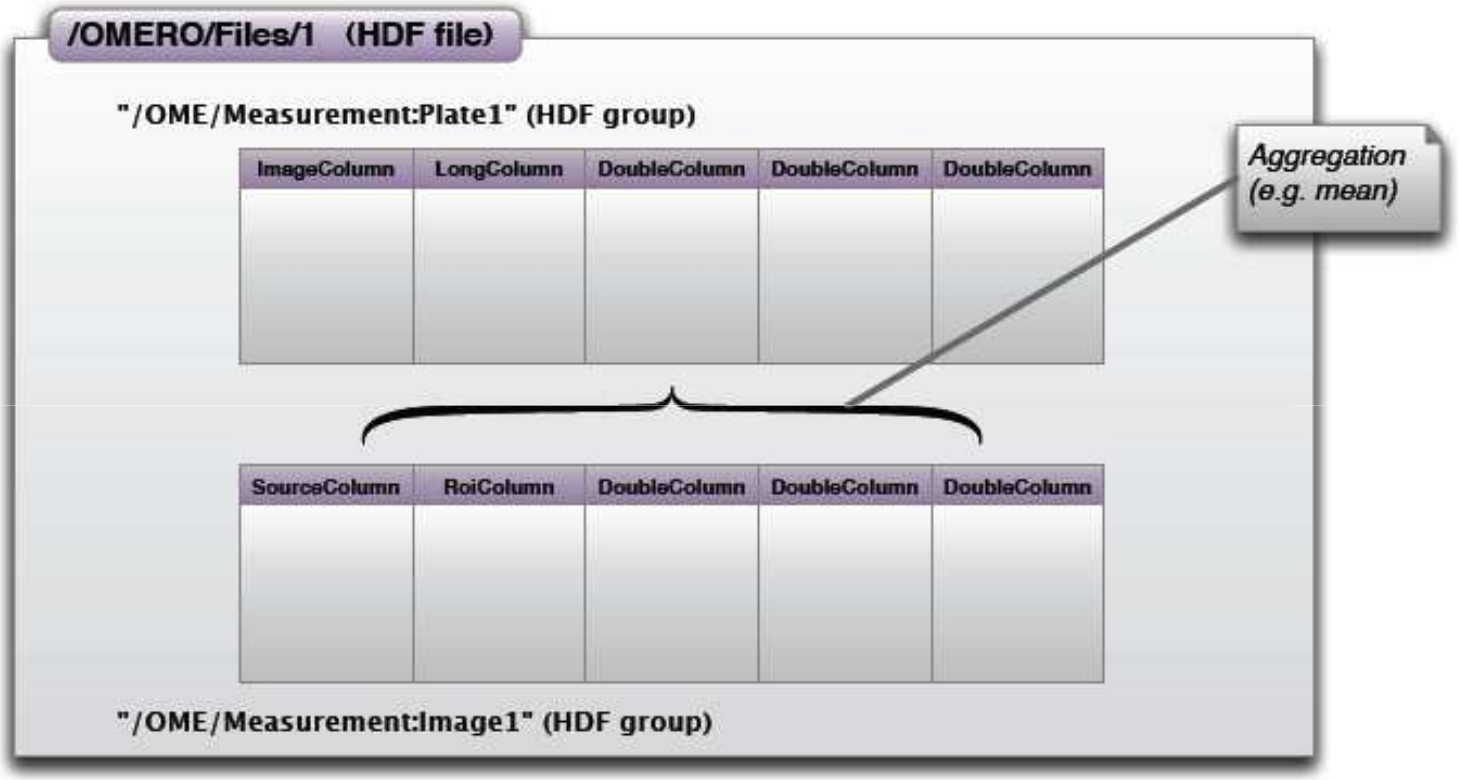
What does OMERO do?



How does OMERO do it?



Support for tabular data



- Tabular format data storage
- Like Excel, but with muscles and brain (each table can handle about 1TB of data)

OMERO is not limited to Bio Images

- Omero is agnostic
  - Configurable, distributed platform that deals with collections of objects
  - Agnostic with respect to object models
  - Agnostic with respect to programming languages (client side)
- Omero can grow
  - Meta class description of objects
  - Omero Tables
- Easy and fast deployment
  - Minimal down-time for model set extension

Agenda

- Introduction
- OMERO
  - What does it do
  - How does it do it
  - Why do we like OMERO
- **OMERO.biobank**
  - What is a “computable” biobank
  - OMERO.biobank overview
  - Code examples
- Conclusions

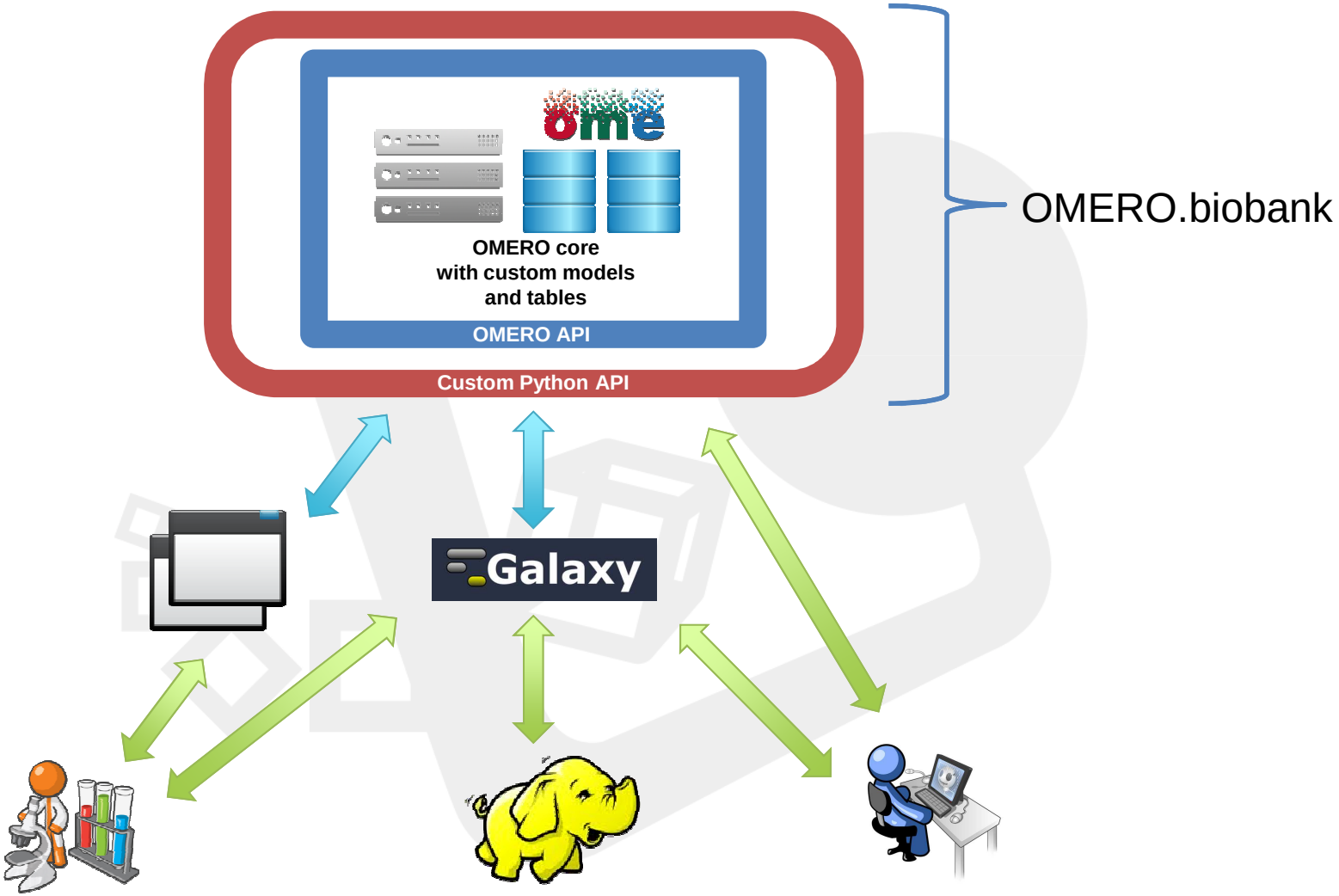
“Computable” biobank

- Computable data repository
  - Uniform formalism for bio-medical data (and operations) description
  - Scalable, distributed technologies
- Driver for data intensive computing
  - MapReduce applications
  - GPU-based applications

Capturing statics and dynamics

- Electronic Health Records
  - Multiple, heterogeneous sources
  - Implementation specific details
- Samples
  - Biological and synthetic
  - Chain of custody
- Description of operations
  - “Experimental” and “digital” operations
- Computation-driven inference process
  - Uniform access to data

OMERO.biobank



Objects handled

- INDIVIDUALS
- STUDIES
- ENROLLMENTS

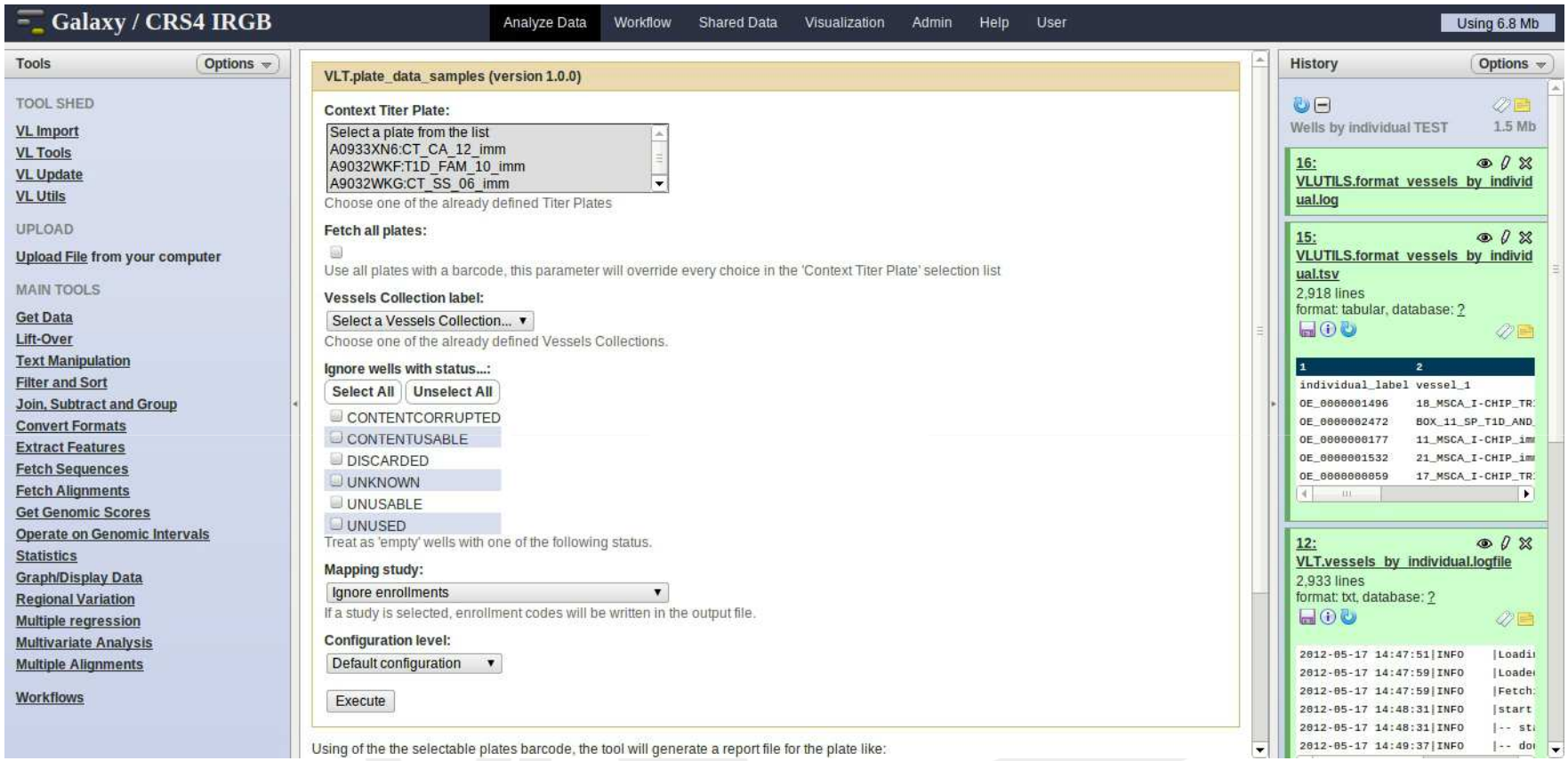
- DATA SAMPLES
  - MICROARRAY MEASURES
    - AFFYMETRIX CEL FILES
    - ILLUMINA BEAD CHIP ARRAY
  - GENOTYPE DATA SAMPLES
- SNP MARKER SETS

- VESSELS
  - TUBES
  - WELLS
- CONTAINERS
  - TITER PLATES
- VESSEL COLLECTIONS

- MARKER DEFINITIONS
- MARKER ALIGNMENTS
- GENOTYPE CALLS

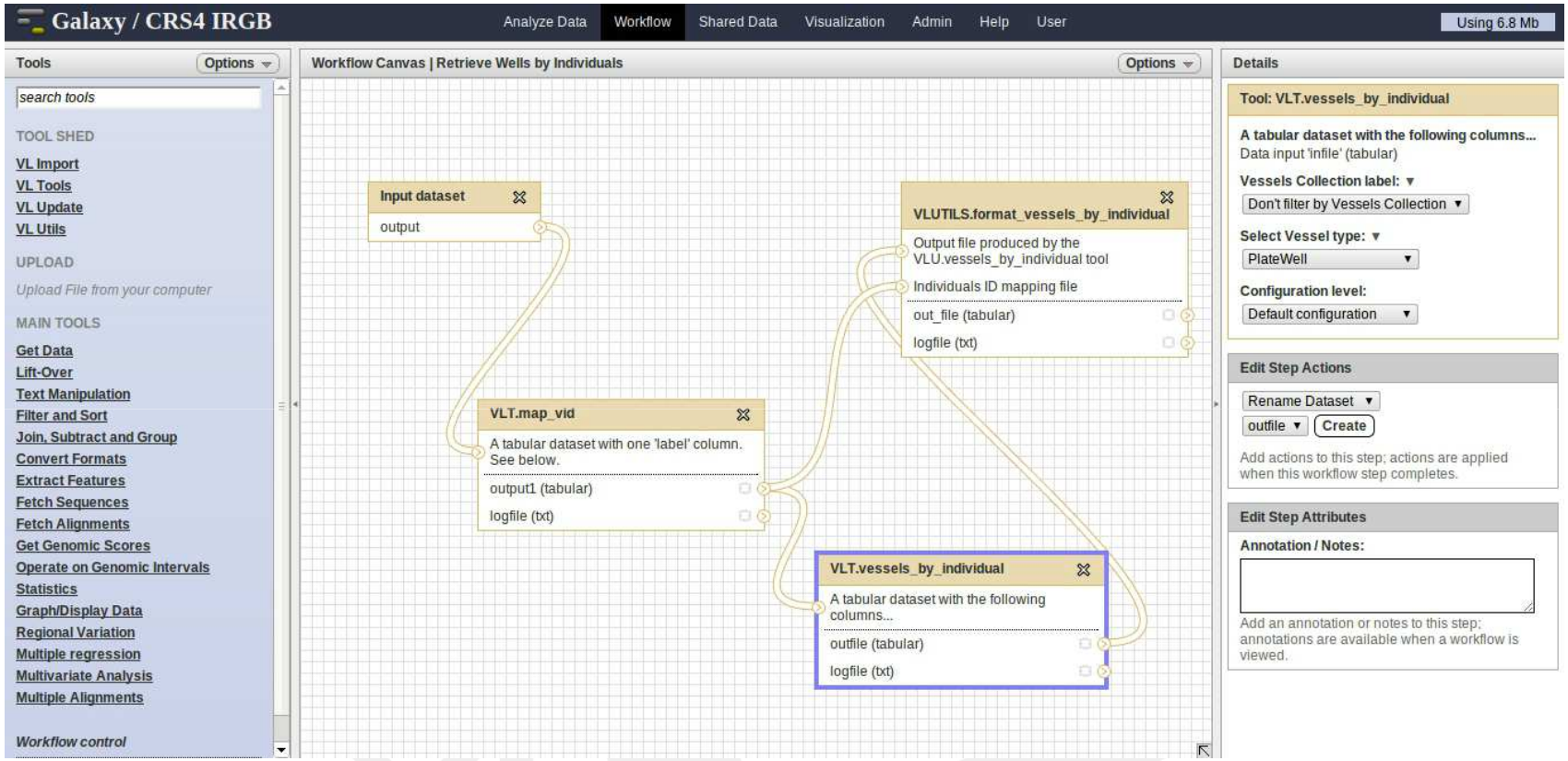
- ANONYMIZED PATIENTS RECORDS
  - OpenEHR-based

Galaxy



- Web interface for CLI tools
- Keep trace of operations

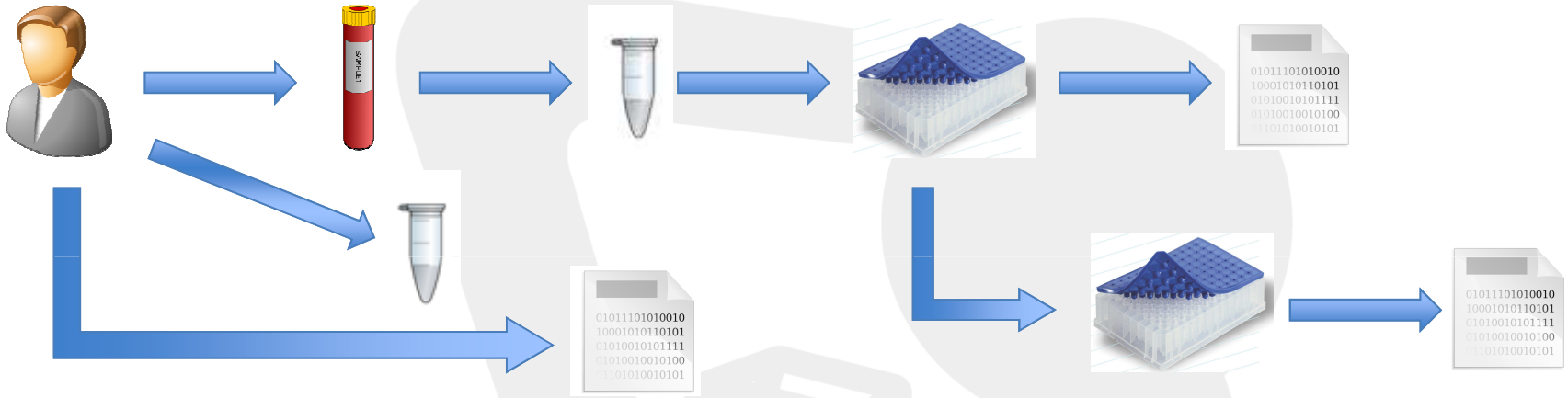
Galaxy



- Powerful workflow editor

Object transitions

- We want to model the “chain of custody” from the biological sample to the synthetic results (i.e., genotype calls)



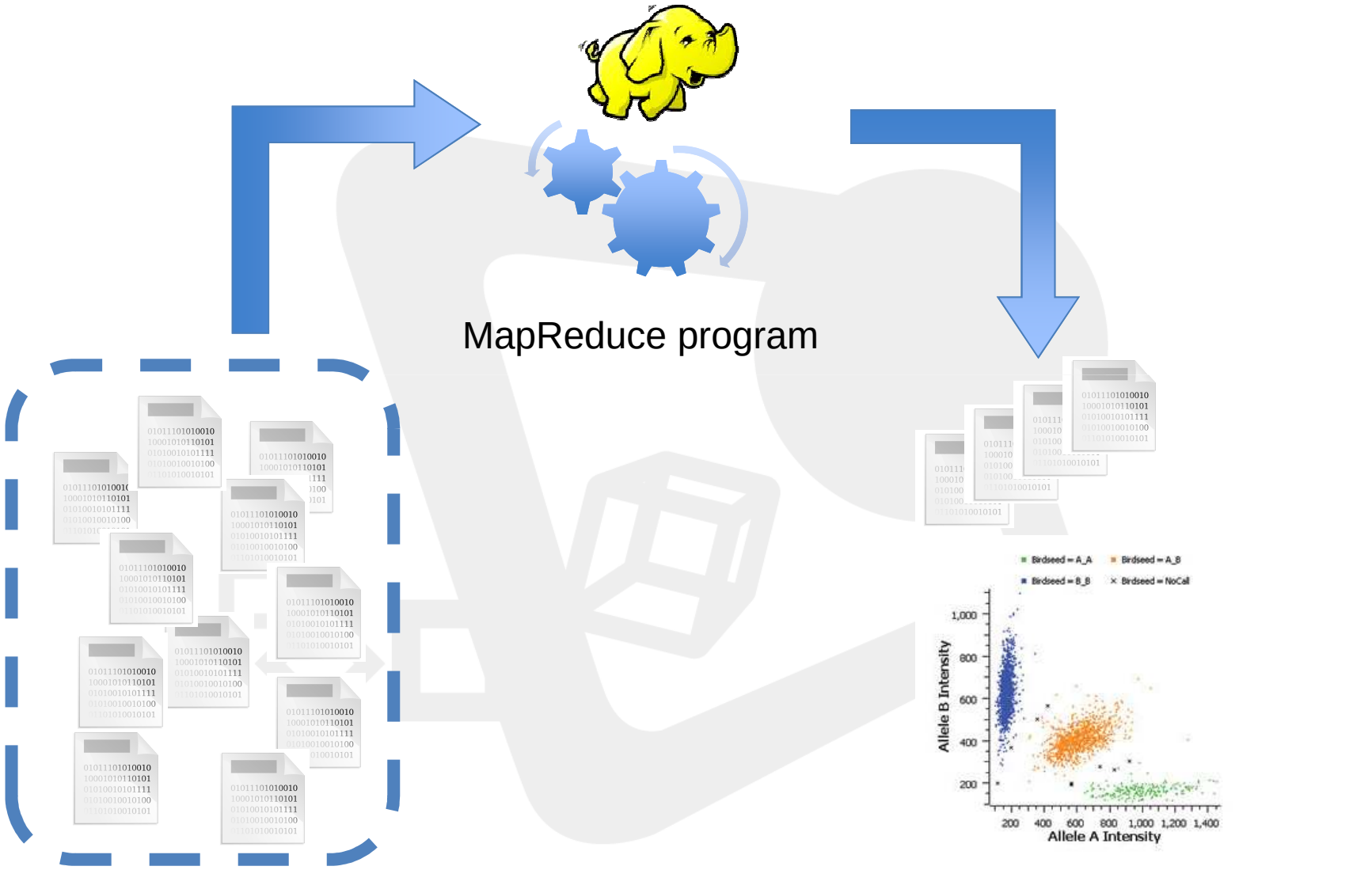
- It is impossible to represent data using a static system, so we introduce the concept of “action”

If object X is the result of an event that occurred on object Y, the action has a reference to it

An action is a generic event that produces data that is recorded into the system

Every object knows the action that generated it

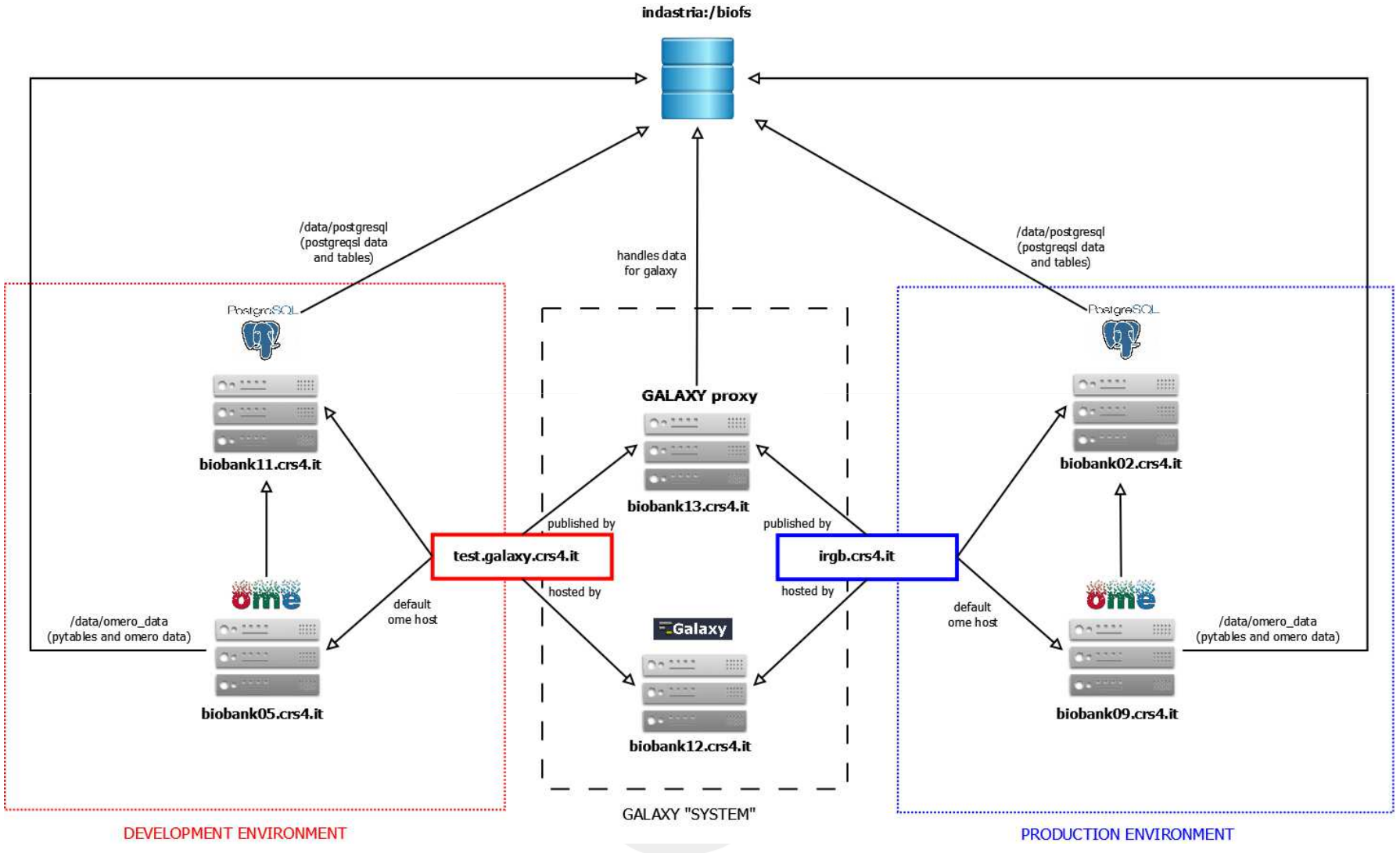
From metadata to computation



Programming interface

```
def main():
    kb = KB(driver='omero')('biobank.crs4.it', 'omero', 'secret')
    maker, model = 'crs4-bl', 'taqman-foo'
    mset = kb.get_markers_set('AFFY_GW6')
    s = gkb.get_gdo_iterator(mset)
    counts = algo.count_homozygotes(s)
    mafs = algo.maf(counts)
    hwe = algo.hwe(counts)
```

System Overview



Data Overview

- Data for autoimmune disease studies (CNR-IRGB / CRS4)
- Used from CRS4, Polaris building 5 lab, Monserrato, Lanusei, Tramariglio
- Currently handling
  - ~16.500 individuals (with parental relationships in order to build families)
  - ~28.200 vessels
  - ~330 Titer Plates
  - 2 Genotyping technologies
    - Illumina Immunochip
      - ~196.000 markers, ~10.000 genotypes
    - Affymetrix Genome-Wide Human SNP Array 6.0
      - ~935.000 markers, ~7.000 genotypes
  - 26.800 clinical records
- We are going to acquire ASAP
  - Illumina Human OmniExpress
    - ~730.000 markers, ~3.000 genotypes
  - Illumina Human Exome
    - ~ 240.000 markers, ~5.000 genotypes

Examples

```
<type id="ome.model.vl.Vessel">
  <properties>
    <required name="activationDate" type="timestamp"/>
    <optional name="destructionDate" type="timestamp"/>
    <required name="currentVolume" type="float"/>
    <required name="initialVolume" type="float"/>
    <required name="content"
                type="ome.model.vl.VesselContent"/>
    <required name="status"
                type="ome.model.vl.VesselStatus"/>
  </properties>
</type>
```

```
<type id="ome.model.vl.Tube"
        superclass="ome.model.vl.Vessel">
  <required name="label" type="string"
              unique="true"/>
  <optional name="barcode" type="string"
              unique="true"/>
</type>
```

```
>>> omero.model.VesselI()
object #0 (::omero::model::Vessel)
{
   _id = <nil>
   _details = object #1 (::omero::model::Details)
   {
      ....
   }
   _loaded = True
   _version = <nil>
   _activationDate = <nil>
   _destructionDate = <nil>
   _currentVolume = <nil>
   _initialVolume = <nil>
   _content = <nil>
   _status = <nil>
   }
```

```
>>> omero.model.TubeI()
object #0 (::omero::model::Tube)
{
   _id = <nil>
   _details = object #1 (::omero::model::Details)
   {
      ....
   }
   _loaded = True
   _version = <nil>
   _activationDate = <nil>
   _destructionDate = <nil>
   _currentVolume = <nil>
   _initialVolume = <nil>
   _content = <nil>
   _status = <nil>
   _label = <nil>
   _barcode = <nil>
}
```

Examples

```
import omero
import omero.model as om
import omero.rtypes as ort


tube = om.TubeI()
tb.label = ort.rstring('TEST_TUBE')
tb.barcode = ort.rstring('XXYYZZ')
tb.activationDate = ort.rtime(time.time())
tb.initialVolume = ort.rfloat(15.0)
tb.currentVolume = ort.rfloat(10.0)
tube_status = om.VesselStatusI()
tube_status.value = ort.rstring('CONTENTUSABLE')
tb.status = tube_status
tube_content = om.VesselContentI()
tube_content.value = ort.rstring('DNA')
tb.content = tube_content


c = omero.client()
s = c.createSession('biobank.crs4.it', 'omero', 'secret')
us = s.getUpdateService()
tube = us.saveAndReturnObject(tube)
c.closeSession()
```

Examples

```
from bl.vl.kb import KnowledgeBase as KB

kb = KB(driver='omero')('biobank.crs4.it', 'omero', 'secret')

tube = kb.factory.create(kb.Tube, {'activationDate' : time.time(),
                                   'initialVolume' : 15.0,
                                   'currentVolume' : 10.0,
                                   'content' : kb.VesselContent.DNA,
                                   'status' : kb.VesselStatus.CONTENTUSABLE,
                                   'label' : 'TEST_TUBE',
                                   'barcode' : 'XXYYZZ',
                                    'action' : kb.create_an_action(kb.get_study('FOOBAR')})

kb.save(tube)
```

```
class Vessel(wp.OmeroWrapper):
    OME_TABLE = 'Vessel'
    __fields__ = [('activationDate', wp.TIMESTAMP, wp.REQUIRED),
                  ('destructionDate', wp.TIMESTAMP, wp.OPTIONAL),
                  ('currentVolume', wp.FLOAT, wp.REQUIRED),
                  ('initialVolume', wp.FLOAT, wp.REQUIRED),
                  ('content', VesselContent, wp.REQUIRED),
                  ('status', VesselStatus, wp.REQUIRED),
                  ('action', Action, wp.REQUIRED),
                  ('lastUpdate', Action, wp.OPTIONAL)]
```

```
class Tube(Vessel):
    OME_TABLE = 'Tube'
    __fields__ = [('label', wp.STRING, wp.REQUIRED),
                  ('barcode', wp.STRING, wp.OPTIONAL)]
```

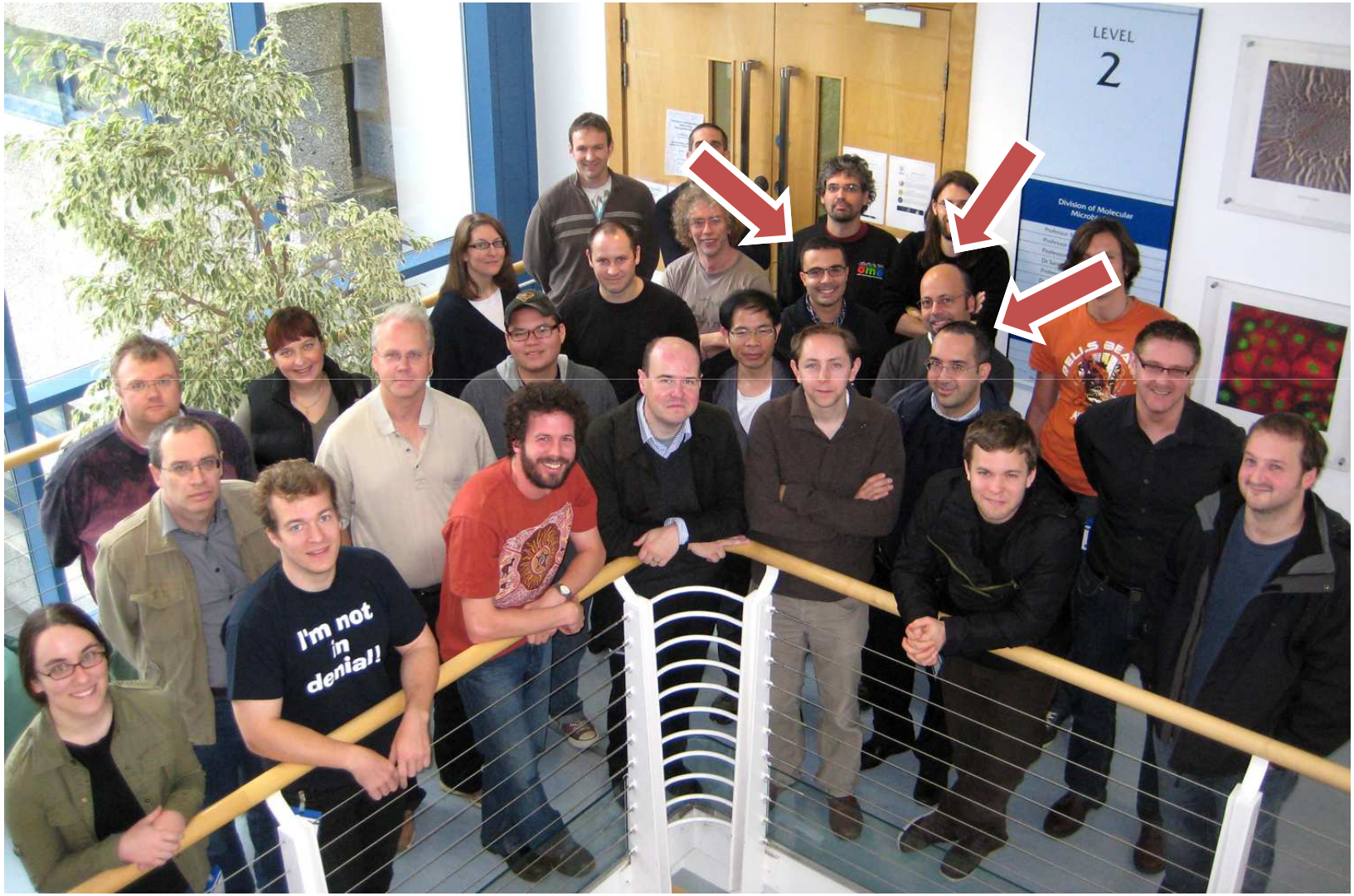
Agenda

- Introduction
- OMERO
  - What does it do
  - How does it do it
  - Why do we like OMERO
- OMERO.biobank
  - What is a “computable” biobank
  - OMERO.biobank overview
  - Code examples
- **Conclusions**

Publications

- **Computable biobanks**, *Bridging the Gap in Biomedical Genetics, Hinxton October 2010*
- **COBIK, a platform for uniform computational approach to integrated clinical and experimental data**, *IFHRO, Milan 2010*
- **Scalable data management and computable framework for large scale longitudinal studies**, *ICHG, Montreal 2011*
- **OMERO: flexible, model-driven data management for experimental biology**, *Allan et al., Nature Methods 9, 245–253 (2012)*

OMERO developers meeting

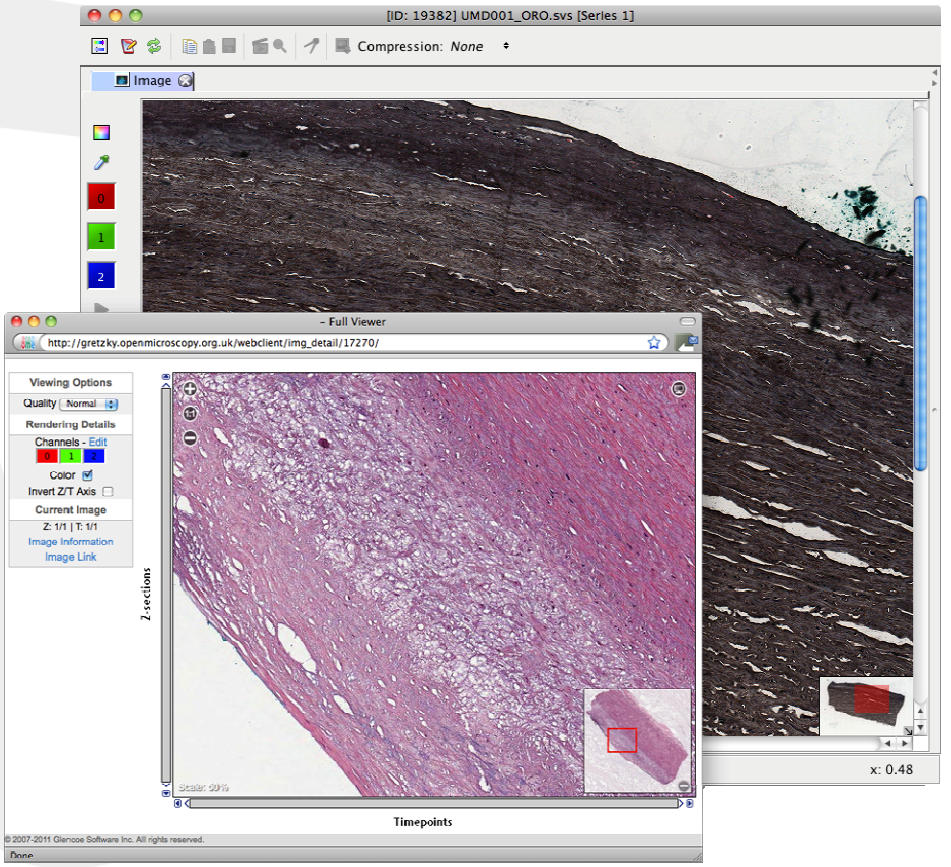


Conclusions

- OMERO is an useful and flexible technology
- If you have a problem related to biomedical data management, OMERO.biobank can be a good solution
- If you have a Big Data management issue you can extend OMERO in order to satisfy your needs

Conclusions

And of course, don't forget that OMERO handles images ☺



… and, of course, thanks for your attention