SRB in the BioEmergences project

Dominique de Waleffe
dominique.dewaleffe@denali.be

Denali SA

CC-IN2P3 – Feb 2, 2009

Partners

- Framework Program 6 project
- Consortium:
  - CNRS – Centre De Recherche en Epistémologie Appliquée (CREA) (FRANCE)
  - Institut Curie (France)
  - Slovenska Technicka Univerzita V Bratislave (Slovakia)
  - Universidad de Málaga (Spain)
  - Denali Consulting S.A. (Belgium)
  - European Molecular Biology Laboratory (Germany)
  - University of Bologna (Italy)
  - CNRS – CC-IN2P3 (France)
- Project fact sheet on CORDIS:`http://tinyurl.com/5yc42k`

Project goals

What?

With the BioEMERGENCES project, we aim at providing an **experimental platform** to observe **in vivo** emergent patterns at various scales and **measure their variability between different individuals** of the same species. This is a strategy towards the measurement of the individual susceptibility to genetic diseases or response to treatments.

...

The main result expected from BioEMERGENCES is the **specification of a European platform to achieve high throughput measurement** of individual differences and screening of drugs combinations such as bi or tri-therapies.

Goals

Team

- Multi-disciplinary team : biologists, mathematicians, engineers, computer scientists

Goals

Team

- Multi-disciplinary team : biologists, mathematicians, engineers, computer scientists

Research

Goals

Team

- Multi-disciplinary team : biologists, mathematicians, engineers, computer scientists

Research

- **Observe:** Using high definition microscopes, capture 4D sets of images of living embryos (Zebra Fish, Sea Urchin,. . .)

Goals

Team

- Multi-disciplinary team : biologists, mathematicians, engineers, computer scientists

Research

- **Observe:** Using high definition microscopes, capture 4D sets of images of living embryos (Zebra Fish, Sea Urchin,. . .)
- **Transform:** Invent methods to go from images to symbolic representations (lineage trees, contours)

Goals

Team

- Multi-disciplinary team : biologists, mathematicians, engineers, computer scientists

Research

- **Observe:** Using high definition microscopes, capture 4D sets of images of living embryos (Zebra Fish, Sea Urchin,. . .)
- **Transform:** Invent methods to go from images to symbolic representations (lineage trees, contours)
- **Compare:** Invent methods for efficient and meaningful comparisons

Goals

Team

- Multi-disciplinary team : biologists, mathematicians, engineers, computer scientists

Research

- **Observe:** Using high definition microscopes, capture 4D sets of images of living embryos (Zebra Fish, Sea Urchin,. . .)
- **Transform:** Invent methods to go from images to symbolic representations (lineage trees, contours)
- **Compare:** Invent methods for efficient and meaningful comparisons

Industrialize

Goals

Team

- Multi-disciplinary team : biologists, mathematicians, engineers, computer scientists

Research

- **Observe:** Using high definition microscopes, capture 4D sets of images of living embryos (Zebra Fish, Sea Urchin,. . .)
- **Transform:** Invent methods to go from images to symbolic representations (lineage trees, contours)
- **Compare:** Invent methods for efficient and meaningful comparisons

Industrialize

- Platform for high throughput execution of the processes

Some details

Gather observations

- Biologists place an embryo under microscope for a number of hours

Some details

Gather observations

- Biologists place an embryo under microscope for a number of hours
- a stack of horizontal images of size $x * y$, separated in time by δt and space by δz are captured

Some details

Gather observations

- Biologists place an embryo under microscope for a number of hours
- a stack of horizontal images of size $x * y$, separated in time by δt and space by δz are captured
- a new stack is captured every ΔT

Some details

Gather observations

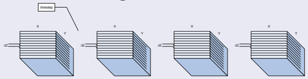
- Biologists place an embryo under microscope for a number of hours
- a stack of horizontal images of size $x * y$, separated in time by δt and space by δz are captured
- a new stack is captured every ΔT
- Repeated for many individuals under different conditions

Some details

Gather observations

- Biologists place an embryo under microscope for a number of hours
- a stack of horizontal images of size $x * y$, separated in time by δt and space by δz are captured
- a new stack is captured every ΔT
- Repeated for many individuals under different conditions

Output

- A large set of large files containing raw images: 

- A set of metadata describing the experiment

Some details

Reconstruct cell lineage tree

- Invent different algorithms to:
  - filter images (remove noise)
  - detect centers of cell nuclei ((x, y, z) position)
  - determine membrane contours (set of 3-D polygons)
  - determine nucleus contours (set of 3-D polygons)
  - identify mytosis (cell divisions)
  - track individual cell from step T_i to step T_{i+1} and build lineage tree
  - compare lineage trees , infer new results

Some details

Reconstruct cell lineage tree

- Invent different algorithms to:
  - filter images (remove noise)
  - detect centers of cell nuclei ((x, y, z) position)
  - determine membrane contours (set of 3-D polygons)
  - determine nucleus contours (set of 3-D polygons)
  - identify mytosis (cell divisions)
  - track individual cell from step T_i to step T_{i+1} and build lineage tree
  - compare lineage trees , infer new results
- visualize reconstructions

Some details

Reconstruct cell lineage tree

- Invent different algorithms to:
  - filter images (remove noise)
  - detect centers of cell nuclei ((x, y, z) position)
  - determine membrane contours (set of 3-D polygons)
  - determine nucleus contours (set of 3-D polygons)
  - identify mytosis (cell divisions)
  - track individual cell from step T_i to step T_{i+1} and build lineage tree
  - compare lineage trees , infer new results
- visualize reconstructions
- correct and annotate datasets

Some figures

Image sizes: $512 * 512 * 8$ to $1024 * 1024 * 8$ pixels, $0.5\mu < \delta x, \delta y < 1.5\mu$, but soon: $2048 * 2048 * 24$,

Number of images in stack: between 50 and 200,

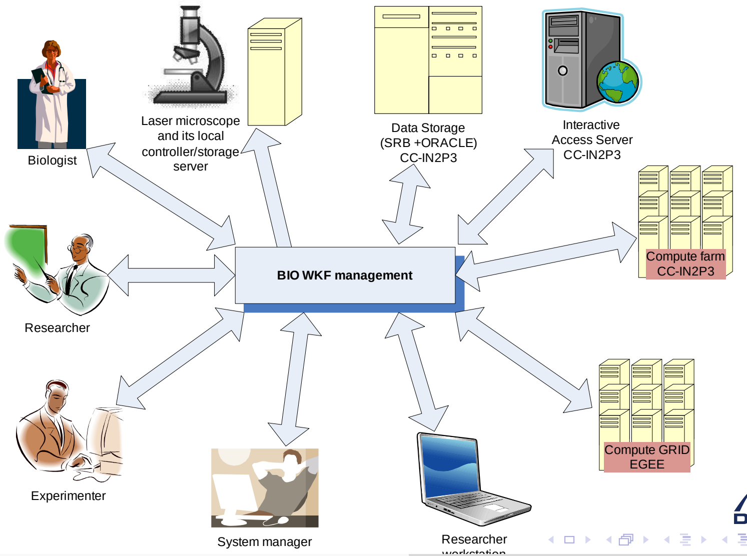
Number of time steps: ΔT typically between 1 and 10 minutes, a few tens to a few hundreds of time intervals captured.

Raw data volumes: 50 to 60 *Gigabytes* of raw image files per experiment (size: 512) but will soon be $1/2$ *Terabytes* with new microscope.

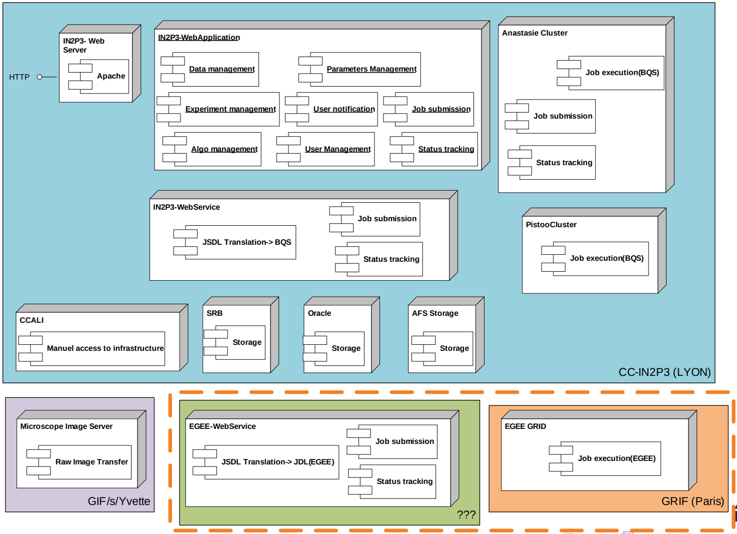
Number of cells: lineage trees contains several million cells.

Current storage used (SRB): in excess of 8 TB.

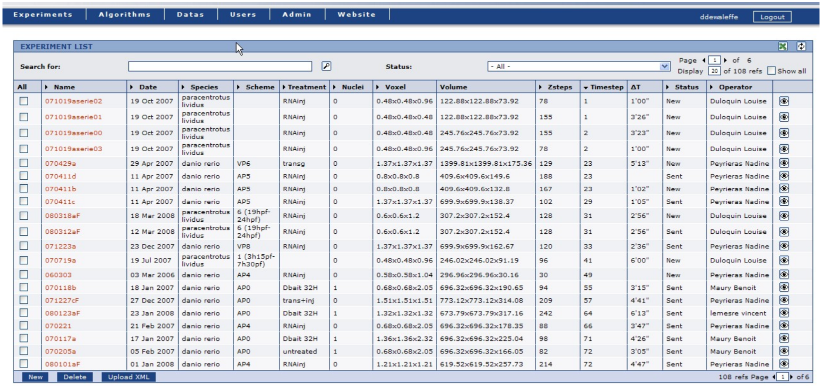
Context diagram


Biologist
Laser microscope and its local controller/storage server
Data Storage (SRB +ORACLE) CC-IN2P3
Interactive Access Server CC-IN2P3
Compute farm CC-IN2P3
BIO WKF management
Researcher
Experimenter
System manager
Researcher workstation
Compute GRID EGEE

Deployment viewpoint



Application: experiment list



EXPERIMENT LIST

Search for:

Status: - All -

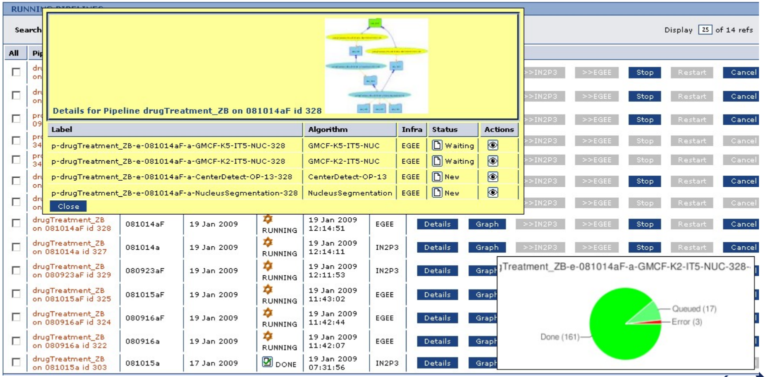
Page 1 of 6
Display 20 of 108 refs ☐ Show all

All	Name	Date	Species	Scheme	Treatment	Nuclei	Voxel	Volume	Zsteps	Timestep	ΔT	Status	Operator	
☐	071019aserie02	19 Oct 2007	paracentrotus lividus		RNAinj	0	0.48x0.48x0.96	122.88x122.88x73.92	78	1	1'00"	New	Duloquin Louise	👁
☐	071019aserie01	19 Oct 2007	paracentrotus lividus		RNAinj	0	0.48x0.48x0.48	122.88x122.88x73.92	155	1	3'26"	New	Duloquin Louise	👁
☐	071019aserie00	19 Oct 2007	paracentrotus lividus		RNAinj	0	0.48x0.48x0.48	245.76x245.76x73.92	155	2	3'23"	New	Duloquin Louise	👁
☐	071019aserie03	19 Oct 2007	paracentrotus lividus		RNAinj	0	0.48x0.48x0.96	245.76x245.76x73.92	78	2	1'00"	New	Duloquin Louise	👁
☐	070429a	29 Apr 2007	danio rerio	VP6	transg	0	1.37x1.37x1.37	1399.81x1399.81x175.36	129	23	5'13"	New	Peyrieras Nadine	👁
☐	070411d	11 Apr 2007	danio rerio	AP5	RNAinj	0	0.8x0.8x0.8	409.6x409.6x149.6	188	23		Sent	Peyrieras Nadine	👁
☐	070411b	11 Apr 2007	danio rerio	AP5	RNAinj	0	0.8x0.8x0.8	409.6x409.6x132.8	167	23	1'02"	New	Peyrieras Nadine	👁
☐	070411c	11 Apr 2007	danio rerio	AP5	RNAinj	0	1.37x1.37x1.37	699.9x699.9x138.37	102	29	1'05"	Sent	Peyrieras Nadine	👁
☐	080318aF	18 Mar 2008	paracentrotus lividus	6 (19hpf-24hpf)	RNAinj	0	0.6x0.6x1.2	307.2x307.2x152.4	128	31	2'56"	New	Duloquin Louise	👁
☐	080312aF	12 Mar 2008	paracentrotus lividus	6 (19hpf-24hpf)	RNAinj	0	0.6x0.6x1.2	307.2x307.2x152.4	128	31	2'56"	Sent	Duloquin Louise	👁
☐	071223a	23 Dec 2007	danio rerio	VP8	RNAinj	0	1.37x1.37x1.37	699.9x699.9x162.67	120	33	2'36"	Sent	Peyrieras Nadine	👁
☐	070719a	19 Jul 2007	paracentrotus lividus	1 (3h15pf-7h30pf)		0	0.48x0.48x0.96	246.02x246.02x91.19	96	41	6'00"	New	Duloquin Louise	👁
☐	060303	03 Mar 2006	danio rerio	AP4	RNAinj	0	0.58x0.58x1.04	296.96x296.96x30.16	30	49		New	Peyrieras Nadine	👁
☐	070118b	18 Jan 2007	danio rerio	AP0	Dbait 32H	1	0.68x0.68x2.05	696.32x696.32x190.65	94	55	3'15"	Sent	Maury Benoit	👁
☐	071227cF	27 Dec 2007	danio rerio	AP0	trans+inj	0	1.51x1.51x1.51	773.12x773.12x314.08	209	57	4'41"	Sent	Peyrieras Nadine	👁
☐	080123aF	23 Jan 2008	danio rerio	AP0	Dbait 32H	1	1.32x1.32x1.32	673.79x673.79x317.16	242	64	6'13"	Sent	lemesre vincent	👁
☐	070221	21 Feb 2007	danio rerio	AP4	RNAinj	0	0.68x0.68x2.05	696.32x696.32x178.35	88	66	3'47"	Sent	Peyrieras Nadine	👁
☐	070117a	17 Jan 2007	danio rerio	AP0	Dbait 32H	1	1.36x1.36x2.32	696.32x696.32x225.04	98	71	4'26"	Sent	Maury Benoit	👁
☐	070205a	05 Feb 2007	danio rerio	AP0	untreated	1	0.68x0.68x2.05	696.32x696.32x166.05	82	72	3'05"	Sent	Maury Benoit	👁
☐	080101aF	01 Jan 2008	danio rerio	AP4	RNAinj	0	1.21x1.21x1.21	619.52x619.52x257.73	214	72	4'47"	Sent	Peyrieras Nadine	👁

108 refs Page 1 of 6

Application: processing pipelines

Details button: brings view below:



Details for Pipeline drugTreatment_ZB on 081014aF id 328

Label	Algorithm	Infra	Status	Actions
p-drugTreatment_ZB-e-081014aF-a-GMCF-K5-IT5-NUC-328	GMCF-K5-IT5-NUC	EGEE	Waiting	👁
p-drugTreatment_ZB-e-081014aF-a-GMCF-K2-IT5-NUC-328	GMCF-K2-IT5-NUC	EGEE	Waiting	👁
p-drugTreatment_ZB-e-081014aF-a-CenterDetect-OP-13-328	CenterDetect-OP-13	EGEE	New	👁
p-drugTreatment_ZB-e-081014aF-a-NucleusSegmentation-328	NucleusSegmentation	EGEE	New	👁

drugTreatment_ZB on 081014aF id 328	081014aF	19 Jan 2009	RUNNING	19 Jan 2009 12:14:51	EGEE
drugTreatment_ZB on 081014a id 327	081014a	19 Jan 2009	RUNNING	19 Jan 2009 12:14:11	IN2P3
drugTreatment_ZB on 080923aF id 329	080923aF	19 Jan 2009	RUNNING	19 Jan 2009 12:11:53	IN2P3
drugTreatment_ZB on 081015aF id 325	081015aF	19 Jan 2009	RUNNING	19 Jan 2009 11:43:02	EGEE
drugTreatment_ZB on 080916aF id 324	080916aF	19 Jan 2009	RUNNING	19 Jan 2009 11:42:44	EGEE
drugTreatment_ZB on 080916a id 322	080916a	19 Jan 2009	RUNNING	19 Jan 2009 11:42:07	EGEE
drugTreatment_ZB on 081015a id 303	081015a	17 Jan 2009	DONE	19 Jan 2009 07:31:56	IN2P3

SRB usage: What

- Main repository: cc-in2p3 Lyon.

SRB usage: What

- Main repository: cc-in2p3 Lyon.
- Raw data storage. Data captured in Paris, format standardized, then copied to SRB. `Srsync` is used.

SRB usage: What

- Main repository: cc-in2p3 Lyon.
- Raw data storage. Data captured in Paris, format standardized, then copied to SRB. `Srsync` is used.
- Derived data.

SRB usage: What

- Main repository: cc-in2p3 Lyon.
- Raw data storage. Data captured in Paris, format standardized, then copied to SRB. `Srsync` is used.
- Derived data.
  - Kept for reuse in further processing.

SRB usage: What

- Main repository: cc-in2p3 Lyon.
- Raw data storage. Data captured in Paris, format standardized, then copied to SRB. `Srsync` is used.
- Derived data.
  - Kept for reuse in further processing.
  - Data history kept in application's DB.

SRB usage: What

- Main repository: cc-in2p3 Lyon.
- Raw data storage. Data captured in Paris, format standardized, then copied to SRB. `Srsync` is used.
- Derived data.
  - Kept for reuse in further processing.
  - Data history kept in application's DB.
- Some additionnal files (movies of reconstructions,...)

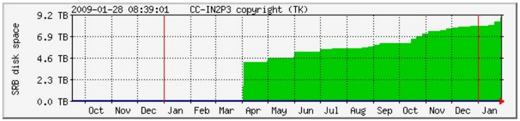
SRB usage: What

- Main repository: cc-in2p3 Lyon.
- Raw data storage. Data captured in Paris, format standardized, then copied to SRB. `Srsync` is used.
- Derived data.
    - Kept for reuse in further processing.
    - Data history kept in application's DB.
- Some additionnal files (movies of reconstructions,...)
- Stores algorithms (scripts, sources, builds procedures, executables)

SRB usage: What

- Main repository: cc-in2p3 Lyon.
- Raw data storage. Data captured in Paris, format standardized, then copied to SRB. `Srsync` is used.
- Derived data.
  - Kept for reuse in further processing.
  - Data history kept in application's DB.
- Some additionnal files (movies of reconstructions,...)
- Stores algorithms (scripts, sources, builds procedures, executables)

Yearly Graph (1 Day Average)



	Max	Average	Current
● space used:	9059.0 GB	3747.0 GB	9059.0 GB

SRB usage: how

- Used in identical manner from both farms.

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system
- Command line: `Smkdir`, `Scd`, `Sls`, `Sput`, `Sget`, `Srm`, `Srsync`

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system
- Command line: `Smkdir`, `Scd`, `Sls`, `Sput`, `Sget`, `Srm`, `Srsync`
- Jargon library used for

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system
- Command line: `Smkdir`, `Scd`, `Sls`, `Sput`, `Sget`, `Srm`, `Srsync`
- Jargon library used for
    - displaying raw data file lists

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system
- Command line: `Smkdir`, `Scd`, `Sls`, `Sput`, `Sget`, `Srm`, `Srsync`
- Jargon library used for
  - displaying raw data file lists
  - Streaming movies [todo]

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system
- Command line: `Smkdir`, `Scd`, `Sls`, `Sput`, `Sget`, `Srm`, `Srsync`
- Jargon library used for
  - displaying raw data file lists
  - Streaming movies [todo]
- Not used in the project:

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system
- Command line: `Smkdir`, `Scd`, `Sls`, `Sput`, `Sget`, `Srm`, `Srsync`
- Jargon library used for
  - displaying raw data file lists
  - Streaming movies [todo]
- Not used in the project:
  - user meta data

SRB usage: how

- Used in identical manner from both farms.
- Mostly used as a file system
- Command line: `Smkdir`, `Scd`, `Sls`, `Sput`, `Sget`, `Srm`, `Srsync`
- Jargon library used for
  - displaying raw data file lists
  - Streaming movies [todo]
- Not used in the project:
  - user meta data
  - web based browser

SRB usage: issues

- Slow access when doing operations requiring catalog access (e.g. `Sls -l`).

SRB usage: issues

- Slow access when doing operations requiring catalog access (e.g. `Sls -l`).
- Bizarre error messages, or no exit codes (makes it difficult in scripts)

SRB usage: issues

- Slow access when doing operations requiring catalog access (e.g. `Sls -l`).
- Bizarre error messages, or no exit codes (makes it difficult in scripts)
- Locking bugs (multiple e.g. Smkdir X) which impact the whole group!

SRB usage: issues

- Slow access when doing operations requiring catalog access (e.g. `Sls -l`).
- Bizarre error messages, or no exit codes (makes it difficult in scripts)
- Locking bugs (multiple e.g. Smkdir X) which impact the whole group!
- deleted stuff is not always fully deleted

Why iRODS?

- Post-processing on ingestion. Could trigger raw data format changes on upload.

Why iRODS?

- Post-processing on ingestion. Could trigger raw data format changes on upload.
- Workflows. Probably redundant with what we already have or can have.

Why iRODS?

- Post-processing on ingestion. Could trigger raw data format changes on upload.
- Workflows. Probably redundant with what we already have or can have.
- Looks like iRODS would be a good fit but:

Why iRODS?

- Post-processing on ingestion. Could trigger raw data format changes on upload.
- Workflows. Probably redundant with what we already have or can have.
- Looks like iRODS would be a good fit but:
    - we have no extra budget in current project

Why iRODS?

- Post-processing on ingestion. Could trigger raw data format changes on upload.
- Workflows. Probably redundant with what we already have or can have.
- Looks like iRODS would be a good fit but:
  - we have no extra budget in current project
  - team is currently looking at using ROOT as storage framework

Why iRODS?

- Post-processing on ingestion. Could trigger raw data format changes on upload.
- Workflows. Probably redundant with what we already have or can have.
- Looks like iRODS would be a good fit but:
  - we have no extra budget in current project
  - team is currently looking at using ROOT as storage framework
  - taking advantage of iRODS imply large re-architecting effort

Why iRODS?

- Post-processing on ingestion. Could trigger raw data format changes on upload.
- Workflows. Probably redundant with what we already have or can have.
- Looks like iRODS would be a good fit but:
  - we have no extra budget in current project
  - team is currently looking at using ROOT as storage framework
  - taking advantage of iRODS imply large re-architecting effort
- sources available.

Why iRODS , some risks ?

- maintainability of complex rule base?

Why iRODS , some risks ?

- maintainability of complex rule base?
  - rule syntax (one liners, readability, choices of operators, comments?)

```
myRule|foo==1|action1(...);action2(...);...|action3(...);action4(...);...
```

Why iRODS , some risks ?

- maintainability of complex rule base?
  - rule syntax (one liners, readability, choices of operators, comments?)

    ```
    myRule|foo==1|action1(...);action2(...);...|action3(...);action4(...);...
    ```
  - Why not sligthly more verbose

```
rule myRule { //this rule is triggered when foo and does bar
   when ( foo == 1)
   do {
     /* watch that this action has side-effects */
     action1 (...);
     action2 (...);...
   }
   on failure {
     action3 (...);
     action4 (...);...
   }
}
```

Why iRODS , some risks ?

- maintainability of complex rule base?
  - rule syntax (one liners, readability, choices of operators, comments?)

```
myRule|foo==1|action1(...);action2(...);...|action3(...);action4(...);...
```

  - Why not sligthly more verbose

```
rule myRule { //this rule is triggered when foo and does bar
   when ( foo == 1)
   do {
     /* watch that this action has side-effects */
     action1 (...);
     action2 (...);...
   }
   on failure {
     action3 (...);
     action4 (...);...
   }
}
```

- can I define new microServices as complex jobs (e.g submit job(s) to farm) without going to C programming?

Conclusion

- BioEmergences has complex distributed data/processing needs

Conclusion

- BioEmergences has complex distributed data/processing needs
- Could make use of iRODS if risks are shown to be a non issue

Questions?