University of Zurich[UZH]

S3IT

IT infrastructure for research: an ongoing journey

Riccardo Murri
Services and Support for Science IT, University of Zurich
riccardo.murri@gmail.com

2019-05-21

Two disclaimers

Opinions and views expressed here are mine only, and may not reflect the official stance of UZH, its IT services, or my colleagues.

Although I have tried to report on scientific research accurately, there can still be errors and inaccuracies. They are all my faults.

What is Research IT?

> "S3IT supports UZH researchers
> in using IT to empower their research,
> from consultancy to application support
> and access to cutting-edge cloud, cluster
> and supercomputing systems."
>
> (source: https://www.s3it.uzh.ch/)

What is Research IT?

From: some.one@uzh.ch
Subject: computing power

Dear Madam/Sir,

I have been invited to submit a revision of the attached paper. There are some missing numbers in Table 1, since **I did not have enough computing power on my office computer to carry out these computations.** A referee has asked us for them, **therefore I need access to a supercomputer.**

Many thanks,
Some One

Traditional options for scientific computing

- Personal workstations
- Large shared batch-queuing systems

Traditional options for scientific computing

- Personal workstations
  - Interactive use
  - Complete control over SW stack
    - . . . but then *you* have to manage it!
  - Limited: how much computing power can fit under your desk?
- Large shared batch-queuing systems

Traditional options for scientific computing

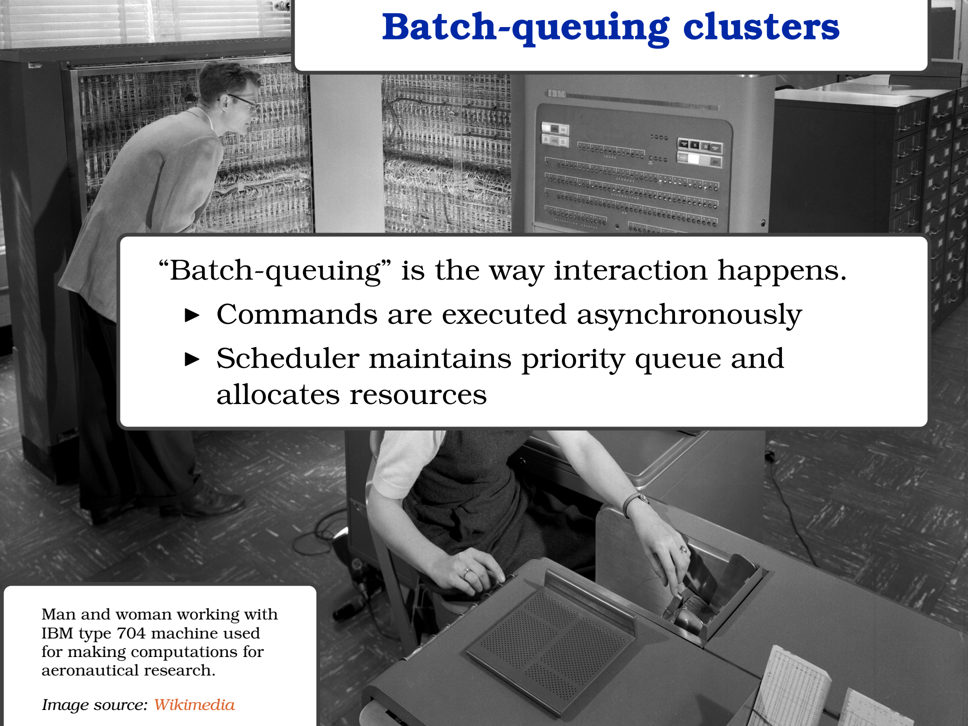
- Personal workstations
- Large shared batch-queuing systems
  - Centrally provided and administered
  - Typically a GNU/Linux cluster nowadays.

Batch-queuing clusters

Man and woman working with IBM type 704 machine used for making computations for aeronautical research.

Image source: Wikimedia

Batch-queuing clusters

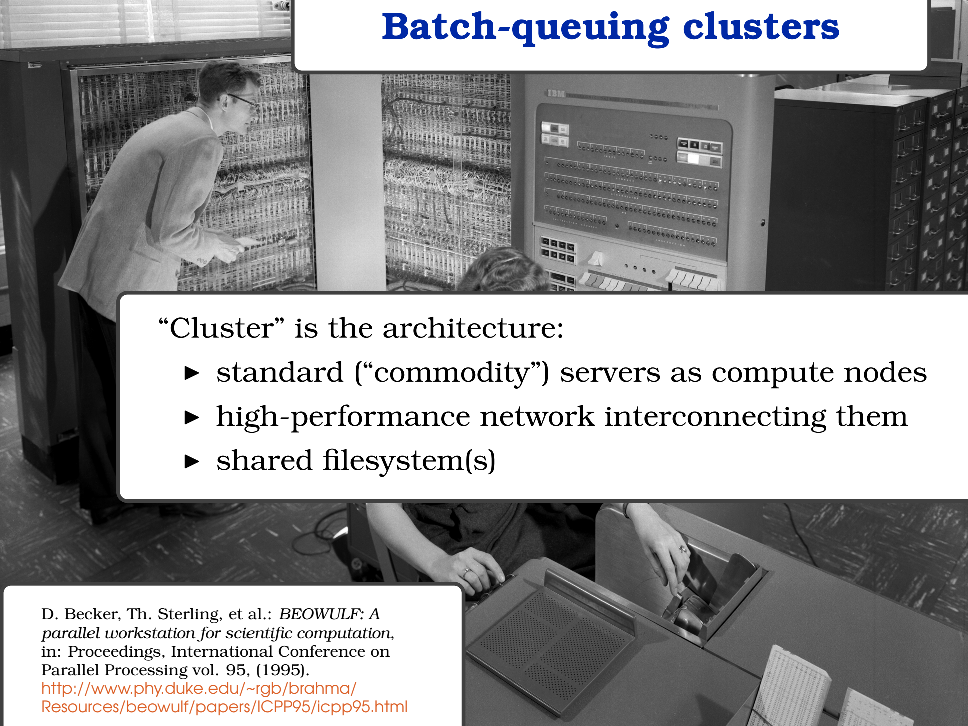


"Batch-queuing" is the way interaction happens.

- Commands are executed asynchronously
- Scheduler maintains priority queue and allocates resources

Man and woman working with IBM type 704 machine used for making computations for aeronautical research.

Image source: Wikimedia



Batch-queuing clusters

"Cluster" is the architecture:

- standard ("commodity") servers as compute nodes
- high-performance network interconnecting them
- shared filesystem(s)

D. Becker, Th. Sterling, et al.: *BEOWULF: A parallel workstation for scientific computation*, in: Proceedings, International Conference on Parallel Processing vol. 95, (1995). http://www.phy.duke.edu/~rgb/brahma/Resources/beowulf/papers/ICPP95/icpp95.html

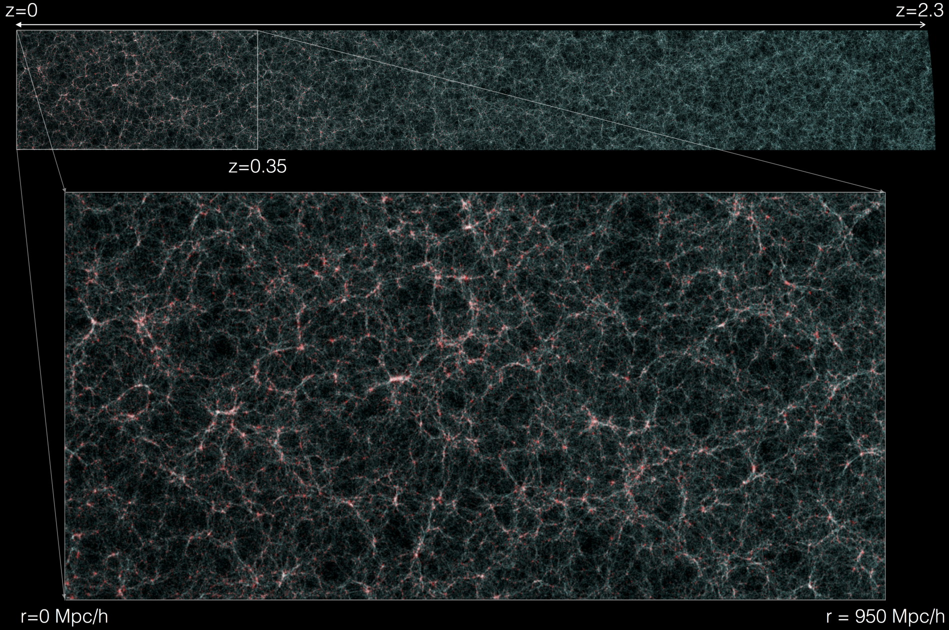
PKDGRAV3

Large N-body simulation code.

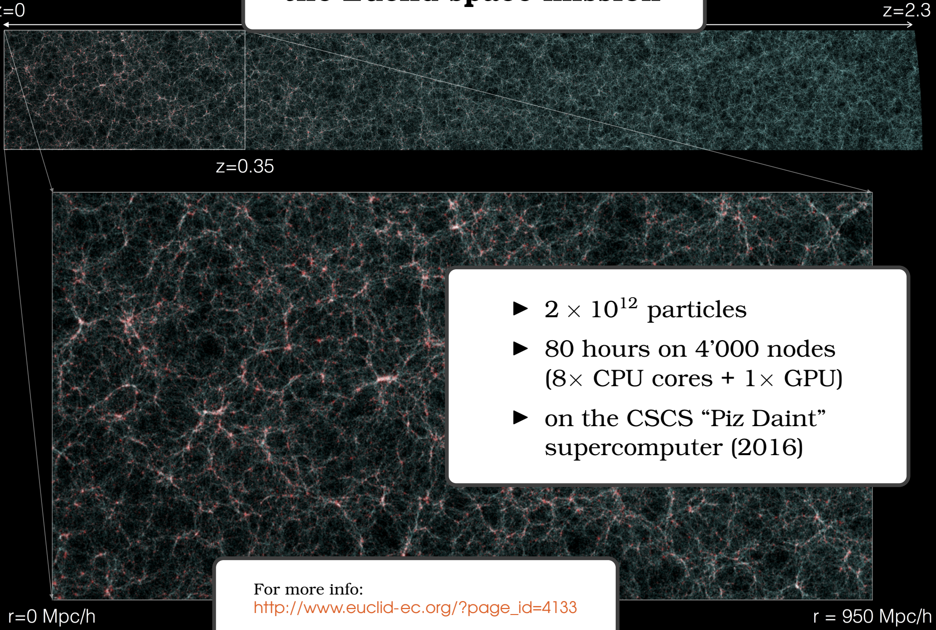
Written by Joachim Stadel, Doug Potter,
and collaborators at UZH.

PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys
D. Potter, J. Stadel, R. Teyssier - Computational Astrophysics and Cosmology, 2017

Flagship mock galaxy catalog



Create test dataset for the Euclid space mission



- 2×10^{12} particles
- 80 hours on 4’000 nodes (8× CPU cores + 1× GPU)
- on the CSCS “Piz Daint” supercomputer (2016)

For more info:
http://www.euclid-ec.org/?page_id=4133

PKDGRAV3: computation and communication

- Fast Multipole Method: $O(N)$
- Communication overlaps with computation
  - one CPU core dedicated to MPI communication
  - latency is more important than bandwidth!
  - supported by Cray's custom cluster interconnect

PKDGRAV3: checkpointing and filesystem I/O

- Light-cone: 240 TB total over 150’000 files.
  - “Final” output, post-processed in further steps of the pipeline
- Checkpoints: 20× 48 TB spread over 20× 28’000 files.
  - *Synchronous:* calculation must stop and wait until file is dumped
  - approx. 2GB per file
  - 1 file per computing thread

PKDGRAV3: checkpointing and filesystem I/O

- Light-cone: 240 TB total over 150’000 files.
  - “Final” output, post-processed in further steps of the pipeline
- Checkpoints: 20× 48 TB spread over 20× 28’000 files.
  - *Synchronous:* calculation must stop and wait until file is dumped
  - approx. 2GB per file
  - 1 file per computing thread

Checkpoints are *needed* to overcome the 24h max runtime policy!

TissueMAPS

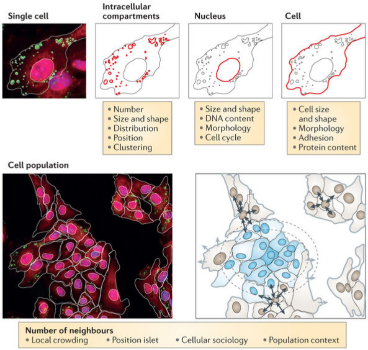
Scalable platform for image analysis of microscopy images.

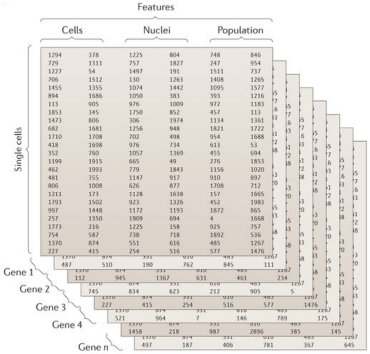
- Developed for image-based cell profiling
- Automated workflow for microscopy image processing
- Browser-based client to explore results and command further analysis

Reference: "Computational Methods and Tools for Reproducible and Scalable Bioimage Analysis"

— M. D. Herrmann, Ph.D. Thesis, Univ. of Zurich (2017).

Image-based Cell Profiling

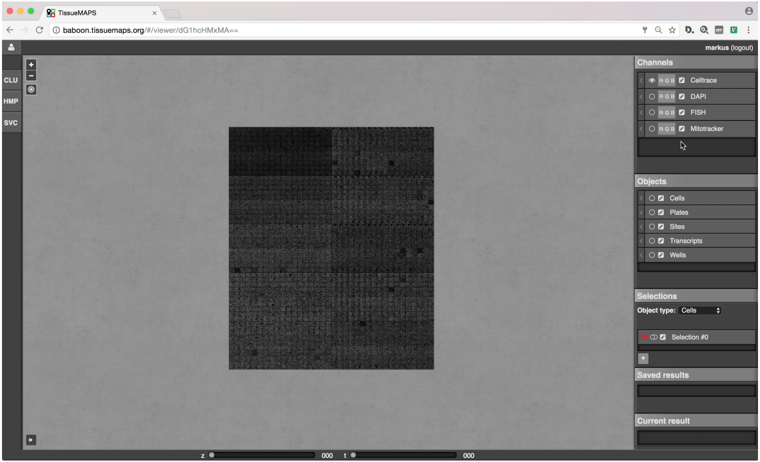




Reference: "Single-cell and multivariate approaches in genetic perturbation screens"

— P. Liberali, B. Snijder, L. Pelkmans, Nat. Rev. Genet., 16:18–32 (2015)

TissueMAPS: Demo of “Transcriptomics” data

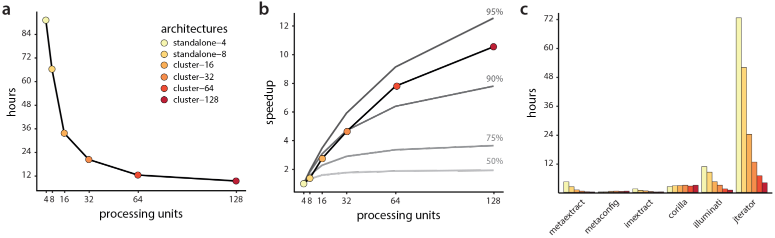


https://youtu.be/Qmqf0ysDrx0

TissueMAPS: Scalability

Time for processing 35’280 microscope images
on clusters of varying size.

- “Emabrassingly parallel”: almost perfectly scalable
  - see figure b — in gray, theoretical speedup for different levels of parallelization
- The “image analysis” step benefits the most from larger resources



TissueMAPS: storage requirements

For instance, in the “transcriptomics” data set:

- input microscope images: 352′800 images, a few MBs each
- pyramid tiles: 41′231′720, a few kB each
- DB table for object features: 650M rows

Conflicting requirements!

PKDGRAV3	TissueMAPS
Single large MPI job.	Huge swarm of short-lived jobs
Low-latency communication.	No communication across tasks.
10'000s of files, a few GBs each	100'000s of files, a few MBs each
Adapted to (high-end) cluster computing environment.	Requires setup of custom DB and web-service endpoints.

Large shared infrastructure

Centrally-administered clusters means larger budget for compute power, but. . .

Same OS and *same* set of installed software for all, *same* scheduler configuration for all, *same* filesystem(s) for all . . .

So, installed software and usage is subject to **policies**.

Conflict on Scheduling Policies

From: unhappy.user@uzh.ch
Subject: cluster priorities

Dear all,

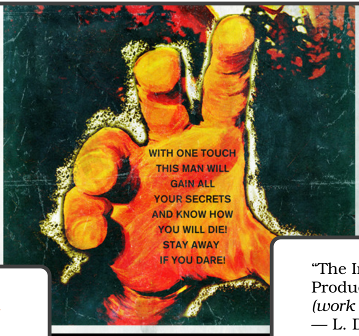
despite my occasional complaints, it has never been explained that group *X* has a default higher priority on the cluster.

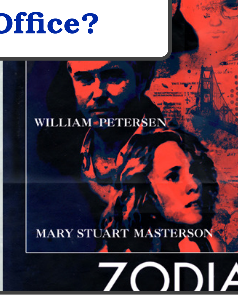
It leads to user *Y* being able to use 3120 cores at the time of writing with all(!) other users combining for 824 cores despite those users having eligible jobs in the queue.

My feeling is that the policy seems outdated and (nowadays) inappropriate.

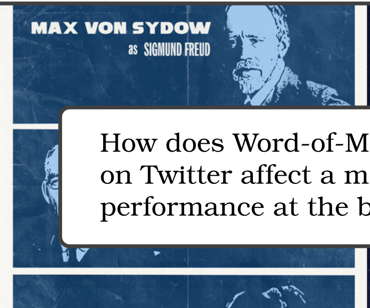
Cheers + thanks, *Z*.

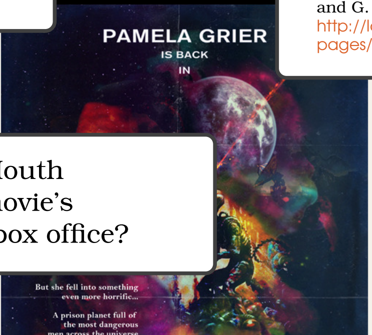
How do Tweets affect the Movie Box Office?





"The Impact of Twitter on New Product Performance"
(work in progress)
— L. Deer, P. Chintagunta, and G. S. Crawford,
http://lachlandeer.github.io/pages/research.html





How does Word-of-Mouth on Twitter affect a movie's performance at the box office?

How do Tweets affect the Movie Box Office?

Try and isolate mechanisms by which Twitter is influencing demand — a computational experiment.

- Get the data:
  - Twitter stream dump
    - 300 movies
    - ± 6 months from release date
  - Box Office performance
- Analyze & Model
  - 85% of Tweets are in the English language — **Filter** out the rest!
  - **Categorize** each Tweet
    - advertisement, buzz, review — each category may affect the dynamics differently
  - **Compute** sentiment score of tweets
  - **Correlate** to Box Office timeseries data

How do Tweets affect the Movie Box Office?

Try and isolate mechanisms by which Twitter is influencing demand — a computational experiment.

- Get the data:
  - Twitter stream dump
    - 300 movies
    - ± 6 months from release date
  - Box Office perfor
- Analyze & Model
  - 85% of Tweets ar
    — **Filter** out the
  - **Categorize** each Tweet
    - advertisement, buzz, review
      — each category may affect the dynamics differently
  - **Compute** sentiment score of tweets
  - **Correlate** to Box Office timeseries data

Classical data science workflow!

Spark/Hadoop are the go-to tools.

How do Tweets affect the Movie Box Office?

Try and isolate mechanisms by which Twitter is influencing demand — a computational experiment.

- Get the data:
  - Twitter stream dump
    - 300 movies
    - ± 6 months from release date
  - Box Office perfor
- Analyze & Model
  - 85% of Tweets ar
    — **Filter** out the
  - **Categorize** each
    - advertisemen
      — each catego
      differently
  - **Compute** sentiment score of tweets
  - **Correlate** to Box Office timeseries data

Classical data science workflow!

Spark/Hadoop are the go-to tools.

Oh, wait. . . Do we have a Spark/Hadoop cluster here?

Three issues with single shared batch clusters

Batch cluster computing is not the only paradigm in use in computational science!

- Policy turns technical issues into social ones.
- No “one size fits all”: Different frameworks (e.g., Spark/Hadoop, Kubernetes) may be required by different communities.
- Interactive environments (e.g., Jupyter, RStudio) and short feedback loop required for development and debugging.

*"Every problem can be solved
by adding one more layer of indirection."*
— *Fundamental Theorem of Software Engineering*

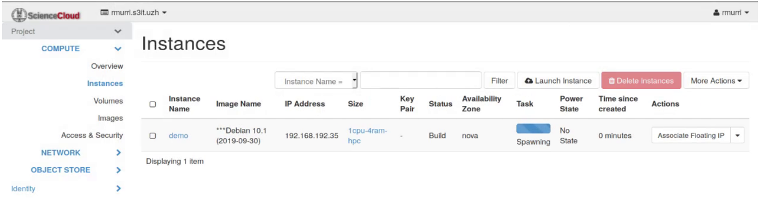
Abstract away the Infrastructure Layer!

Use *Infrastructure-as-a-Service* as a base for providing compute infrastructure.

We can create and setup ad-hoc computing infrastructures:

- *dedicated*: no sharing, exactly the software and policies you want
- *ephemeral*: create when idea comes, dispose when experiment is over

IaaS cloud computing



Demo: starting and stopping a VM on OpenStack

IaaS cloud computing

1. Provision *virtual* resources:
   - virtual machines (VM)
   - block and object storage
   - software-defined networking
2. Pay per use
   - No upfront investment in HW
3. Network-accessible API for control
   - allows *scripting* the set-up and tear-down of infrastructure
   - “infrastructure as code”

Advantages of “Infrastructure as Code”

1. Reproducibility
   - You can re-create the exact same infrastructure at a later time.
2. Version Control
3. Easy to clone/adapt

Advantages of “Infrastructure as Code”

1. Reproducibility
2. Version Control
   - can easily roll back changes!
   - precise log of how the infrastructure evolved over time
   - . . . plus all niceties that we have from coding environments
3. Easy to clone/adapt

Advantages of "Infrastructure as Code"

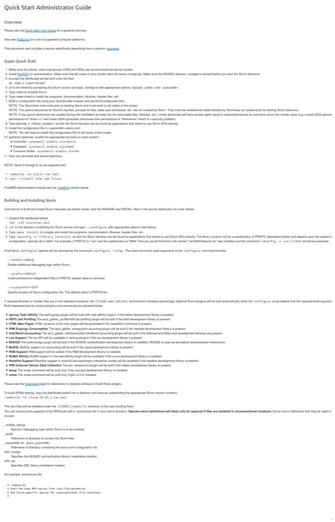
1. Reproducibility
2. Version Control
3. Easy to clone/adapt
   - It's just text files!
   - Good configuration/deployment tools have a programming languages: functions allow defining *"parametric infrastructure"*

"Software-defined Sysadmin"

However, there are infrastructure setup chores:

- e.g., software installation and configuration
- now you must do these yourself!

ElastiCluster is our solution for automation of basic sysadmin tasks: provisioning and initial setup of a computing infrastructure.



What is ElastiCluster

ElastiCluster provides a **command line tool** and a Python API to **create, set up and resize** computing clusters hosted on IaaS cloud infrastructures.

Main function is to get a compute cluster up and running with a single command.

Effectively, a wrapper around **Ansible** which provides:

- idempotent configuration playbooks
- no-bootstrap remote actions via SSH

ElastiCluster

SLURM cluster
on Ubuntu 14.04

https://youtu.be/DDm6-QEnNsU

ElastiCluster features (1)

Computational clusters supported:

- Batch-queuing systems:
  - SLURM
  - GridEngine
  - PBSPro
  - HTCondor
- Kubernetes
- Spark / Hadoop

Distributed storage:

- CephFS
- GlusterFS
- HDFS

Optional add-ons:

- Ganglia
- JupyterHub
- EasyBuild

ElastiCluster features (2)

Run on multiple clouds:

- Amazon EC2
- Google Compute Engine
- OpenStack
- MS Azure
- ... and anything supported by LibCloud

Supports several distros as base OS:

- Debian 10.x *(buster)*, Debian 9.x *(stretch)*
- Ubuntu 18.04 *(bionic)*, 16.04 *(xenial)*
- CentOS / Scientific Linux 7.x

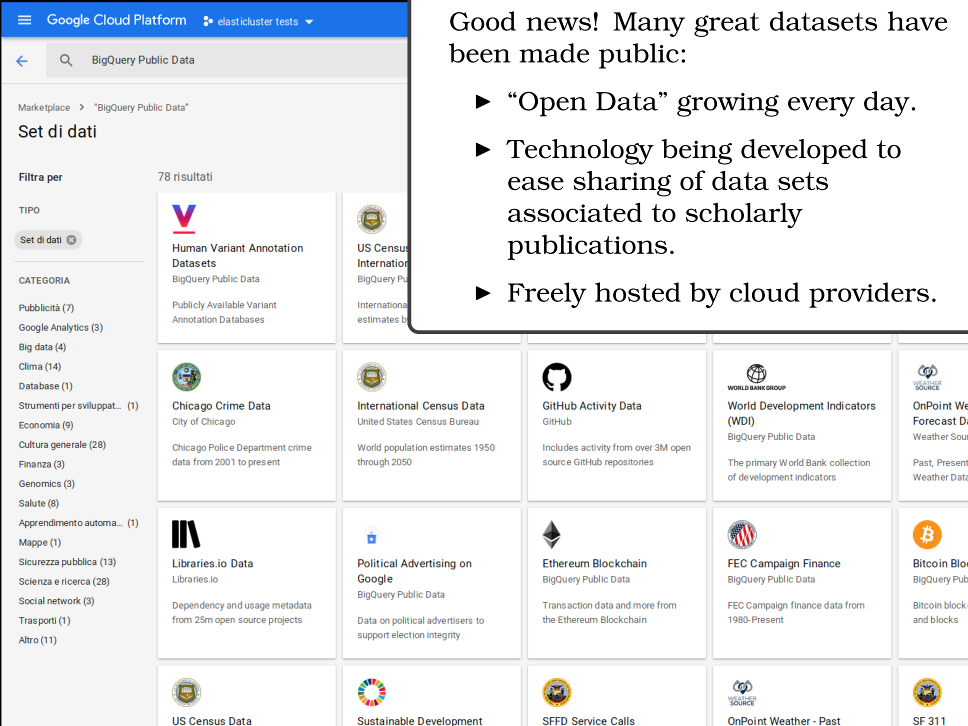
```
changed: [server001 -> localhost] => {"changed": true, "cmd": "echo 'done' > '/tmp/elasticluster.Q
lta": "0:00:00.001948", "end": "2018-11-06 16:21:50.888160", "rc": 0, "start": "2018-11-06 16:21:5
derr_lines": [], "stdout": "", "stdout_lines": []}
changed: [server002 -> localhost] => {"changed": true, "cmd": "echo 'done' > '/tmp/elasticluster.Q
lta": "0:00:00.001598", "end": "2018-11-06 16:21:50.912735", "rc": 0, "start": "2018-11-06 16:21:5
derr_lines": [], "stdout": "", "stdout_lines": []}
changed: [server003 -> localhost] => {"changed": true, "cmd": "echo 'done' > '/tmp/elasticluster.Q
lta": "0:00:00.001518", "end": "2018-11-06 16:21
derr_lines": [], "stdout": "", "stdout_lines":

PLAY RECAP ***********************************
*****************************************
client001                  : ok=60    changed=9
server001                  : ok=80    changed=15
server002                  : ok=74    changed=9
server003                  : ok=74    changed=9

2018-11-06 16:21:51 monia gc3.elasticluster[301

Your cluster `gluster-on-ubuntu` is ready!

Cluster name:     gluster-on-ubuntu
Cluster template: gluster-on-ubuntu
Default ssh to node: client001
- client nodes: 1
- server nodes: 3

To login on the frontend node, run the command:

    elasticluster ssh gluster-on-ubuntu

To upload or download files to the cluster, use the command:

    elasticluster sftp gluster-on-ubuntu

(elasticluster)
rmurri@monia: ~/w/elasticluster issues/#496 ⚡
$
```

- **On demand provisioning** of computational clusters
- Clusters/servers for **Teaching**
- **Testing** new software or configurations
- **Scaling** a permanent computing infrastructure

More on ElastiCluster

No Compute without Data

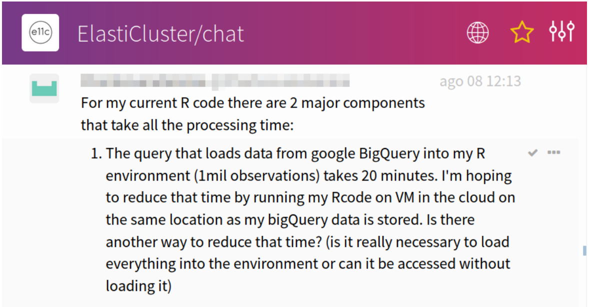
Unless you're modeling
from first principles,
you need data
to base your computations on.



Good news! Many great datasets have been made public:

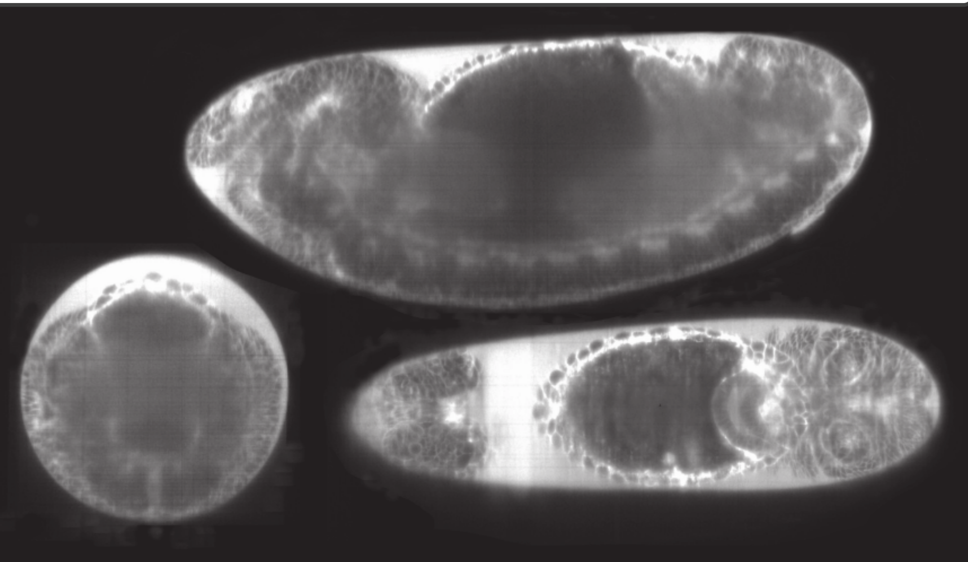
- "Open Data" growing every day.
- Technology being developed to ease sharing of data sets associated to scholarly publications.
- Freely hosted by cloud providers.

Access speed still an issue?



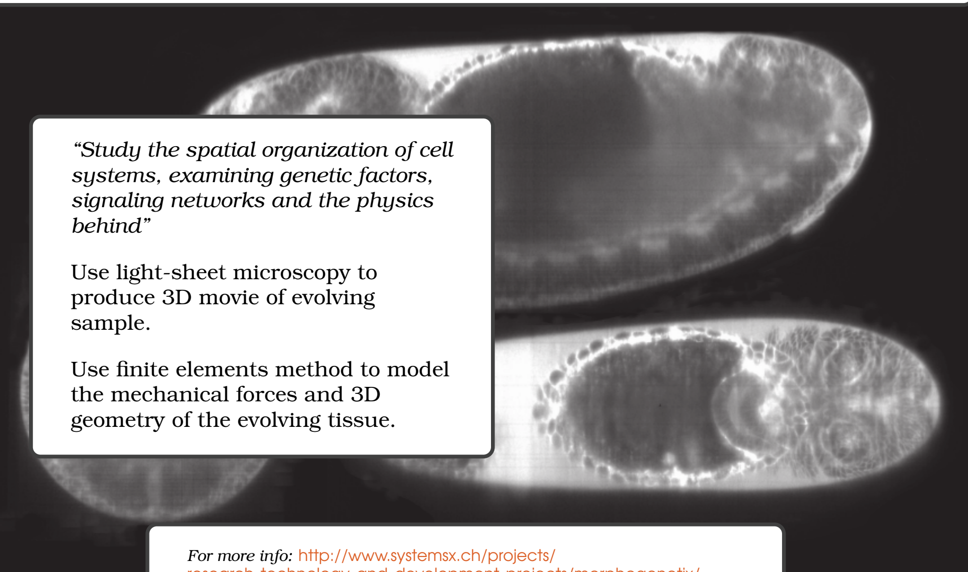
"The query that loads data from BigQuery into my R environment *[. . .]* takes 20 minutes."

MorphogenetiX: Modelling 3-D Shaping of Tissues



Single sections of a fly embryo imaged in 3D with light sheet microscopy. Clockwise from top: side view, top view, frontal view. © Damian Brunner

MorphogenetiX: Modelling 3-D Shaping of Tissues



"Study the spatial organization of cell systems, examining genetic factors, signaling networks and the physics behind"

Use light-sheet microscopy to produce 3D movie of evolving sample.

Use finite elements method to model the mechanical forces and 3D geometry of the evolving tissue.

For more info: http://www.systemsx.ch/projects/research-technology-and-development-projects/morphogenetix/

Single sect… microscopy.
Clockwise from top: side view, top view, frontal view. © Damian Brunner

MorphogenetiX: Modelling 3-D Shaping of Tissues

Use light-sheet microscopy to produce 3D movie of evolving sample.

- Up to 8TB of data every 4 hours.

Post-process images to generate discretized model and connectivity information.

From FEM model to run simulation of embryo development.

- Again, large production of data.

MorphogenetiX: Modelling 3-D Shaping of Tissues

Use light-sheet microscopy to produce 3D movie of evolving sample.

- Up to 8TB of data every 4 hours.

Post-process i
connectivity ir

From FEM mc
development.

- Again, lar

Bandwidth to data center $\approx$ 1Gbit/s.

16 hours to copy 8TB.

$4\times$ times more than to produce it!

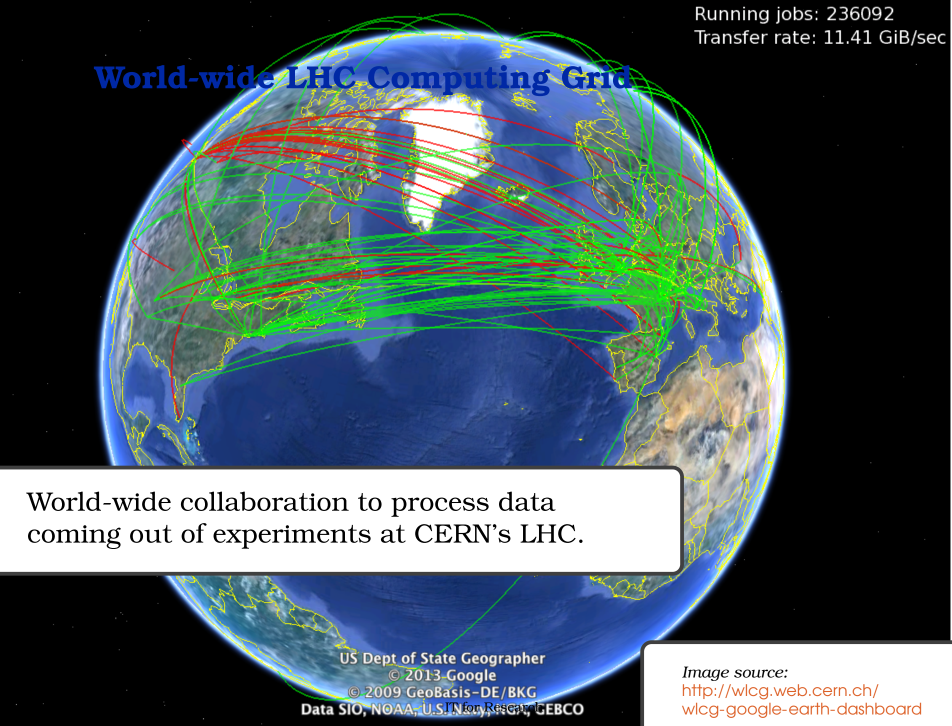
Need to filter data at source.

Not getting better short-term

"a 10x *(network)* speed increase over 15 years
is far slower than the 2x speed per 1.5 years
typically cited for Moore's law."
— https://en.wikipedia.org/wiki/100_Gigabit_Ethernet

"Recent growth in *(genome)* sequencing technology
eclipses Moore"
— https://blog.acolyer.org/dna-storage-fig-1/

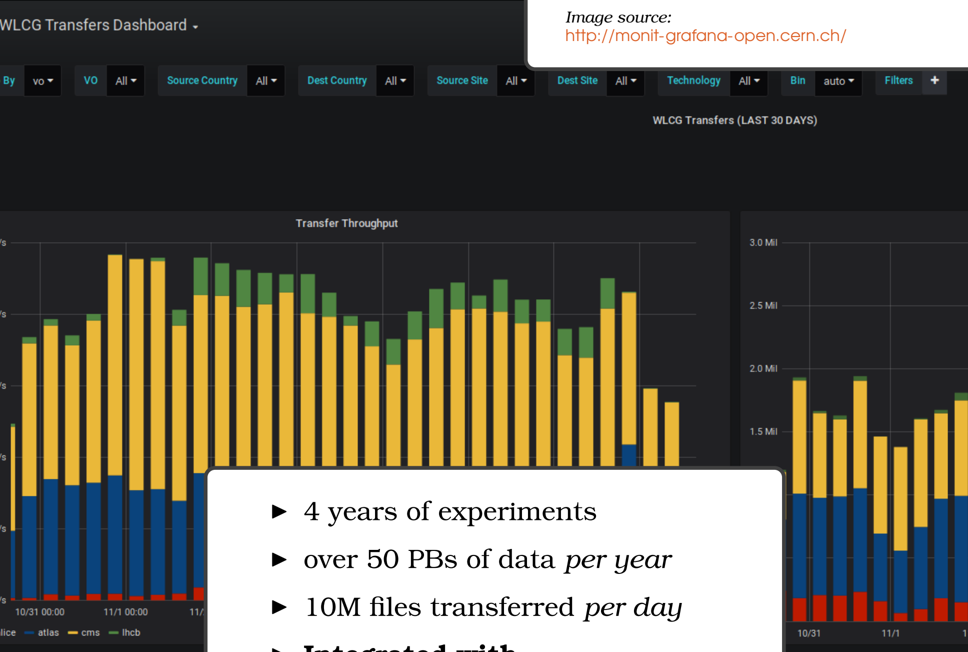
World-wide LHC Computing Grid



World-wide collaboration to process data coming out of experiments at CERN's LHC.

Image source:
http://wlcg.web.cern.ch/wlcg-google-earth-dashboard

Image source:
http://monit-grafana-open.cern.ch/



- 4 years of experiments
- over 50 PBs of data *per year*
- 10M files transferred *per day*
- **Integrated with computing grid!**

Data is *the* problem for experimental science

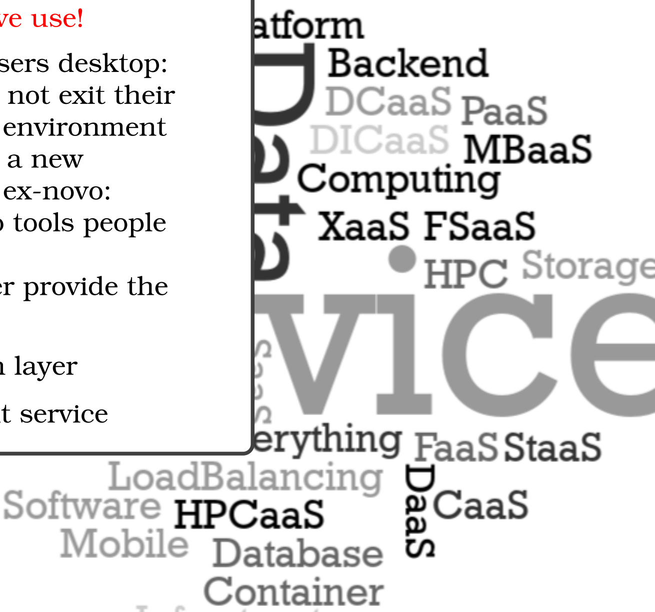
- Data can be produced faster than it can be moved.
  - HEP model: few large experiment sites, well-connected to high-speed Internet backbone.
- Some data comes with strict legal requirements attached!

What will a Science Cloud look like?

- Support interactive use!
- Flexible execution layer
- Data management service

What will a Science Cloud look like?

- Support interactive use!
  - Extends to users desktop: users should not exit their development environment
  - Cannot write a new environment ex-novo: integrate into tools people already use
  - Can container provide the bridge?
- Flexible execution layer
- Data management service



What will a Science Cloud look like?

- Support interactive use!
- Flexible execution layer
  - Need to scale!
  - Reconfigurable mix of batch-queueing and other paradigms
- Data management service

What will a Science Cloud look like?

- Support interactive use!
- Flexible execution layer
- Data management service
  - Manages data life cycle: from production, to consumption, to archival
  - Not necessarily a filesystem
  - Data format aware: can slice, filter, pre-process . . .

. . . but in the end, it's a people's thing

- IT support moving closer to researchers and away from infrastructure
- Interdisciplinary teams will be key

Special thanks go to . . .

. . . to the ElastiCluster fellow devs:
Antonio Messina, Nicolas Bär

. . . to my colleagues at GC3/S3IT:
Sergio Maffioletti, Tyanko Aleksiev

. . . to the Scientists who contributed:
Lachlan Deer, David Dreher,
Markus D. Herrmann, Franz Liem, Lucas Pelkmans,
Doug Potter, Joachim Stadel

Thanks!

(Any questions?)

Appendix

ElastiCluster

On-demand provisioning of compute clusters

TissueMAPS

- Deploy on cloud: compute cluster + parallel DB + web front-end

WLCG

- Deploy compute cluster with SL6.*x*

"Twitter Effect on Movies" experiment

- Deploy Spark + JupyterHub

PKDGRAV3

- Still need a real HPC cluster!

On-demand provisioning of compute clusters

:-) TissueMAPS
- Deploy on cloud: compute cluster + parallel DB + web front-end

WLCG
- Deploy compute cluster with SL6.*x*

"Twitter Effect on Movies" experiment
- Deploy Spark + JupyterHub

PKDGRAV3
- Still need a real HPC cluster!

On-demand provisioning of compute clusters

- :-) TissueMAPS
  - Deploy on cloud: compute cluster + parallel DB + web front-end
- :-) WLCG
  - Deploy compute cluster with SL6.*x*
- "Twitter Effect on Movies" experiment
  - Deploy Spark + JupyterHub
- PKDGRAV3
  - Still need a real HPC cluster!

On-demand provisioning of compute clusters

:-) TissueMAPS

- Deploy on cloud: compute cluster + parallel DB + web front-end

:-) WLCG

- Deploy compute cluster with SL6.x

:-) "Twitter Effect on Movies" experiment

- Deploy Spark + JupyterHub

PKDGRAV3

- Still need a real HPC cluster!

On-demand provisioning of compute clusters

- :-) TissueMAPS
  - Deploy on cloud: compute cluster + parallel DB + web front-end
- :-) WLCG
  - Deploy compute cluster with SL6.*x*
- :-) "Twitter Effect on Movies" experiment
  - Deploy Spark + JupyterHub
- :-(PKDGRAV3
  - Still need a real HPC cluster!

Clusters for teaching

Example: JupyterHub+Spark clusters

- for teaching courses (e.g., data science), or
- for short-lived events (e.g., workshops).

Key ingredient is the ability to apply custom Ansible playbooks on top of the standard ones, to make per-event customizations.

Scaling permanent clusters

Example: additional WLCG cluster for ATLAS analysis hosted on SWITCHengines

Processes: Grid Local

Country	Site	CPUs	Load (processes: Grid+local)	Queueing
Switzerland	ATLAS BOINC	98139	7894+6083	1571+4063
	ATLAS BOINC 3	98139	5815+8163	1253+4371
	ATLAS BOINC TEST	644	0+0	0+0
	Bern ce01 (UNIBE-LHEP)	1513	1048+0	156+0
	Bern ce02 (UNIBE-LHEP)	770	624+0	159+0
	Bern ce04 (UNIBE-LHEP>	304	304+0	192+0
	Bern UBELIX T3	4472	385+2822	208+2450
	CSCS BRISI Cray XC40	1500	576+0	154+0
	Geneva (UNIGE-DPNC)	720	168+349	169+0
	Lugano PHOENIX T2 arc>	1920	1526+4040	411+14
	Lugano PHOENIX T2 arc>	2240	2065+3504	391+4
	Lugano PHOENIX T2 arc>	2048	1864+3704	407+1
TOTAL	***12 sites***	***212409***	***22269 + 28665***	***5071 + 10903***

Reference: S. Haug and G. F. Sciacca,

"ATLAS computing on Swiss Cloud SWITCHengines", CHEP 2016

Scaling permanent clusters

Example: additional WLCG cluster for ATLAS analysis hosted on SWITCHengines

> *"A 304 virtual CPU core Slurm cluster was then started with one command on the command line. This process took about one hour. A few post-launch steps were needed before the cluster was production ready. However, a skilled system administrator can setup a 1000 core elastic Slurm cluster on the SWITCHengines within half a day.* ***As a result the cluster becomes a transient or non-critical component. In case of failure one can just start a new one, within the time it would take to get a hard disk exchanged.****"*

Reference: S. Haug and G. F. Sciacca,

"ATLAS computing on Swiss Cloud SWITCHengines", CHEP 2016

Example: SLURM cluster

Cluster definition is done in a INI-format text file.

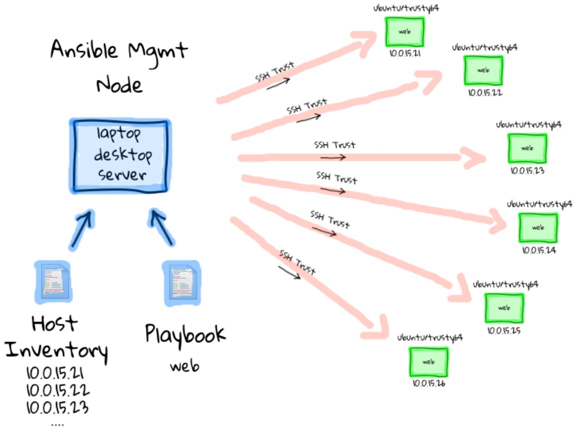
```
[cluster/slurm]
cloud=openstack
login=ubuntu
setup=slurm
frontend_nodes=1
compute_nodes=4
ssh_to=frontend
security_group=default
image_id=...
flavor=4cpu-16ram-hpc

[setup/slurm]
frontend_groups=slurm_master
compute_groups=slurm_worker
```

```
[cloud/openstack]
provider=openstack
auth_url=http://...
username=***
password=***
project_name=***

[login/ubuntu]
image_user=ubuntu
image_user_sudo=root
image_sudo=yes
user_key_name=elasticluster
user_key_private=
    ~/.ssh/id_rsa
user_key_public=
    ~/.ssh/id_rsa.pub
```

More examples: https://github.com/gc3-uzh-ch/elasticluster/tree/master/examples

Ansible

Ansible for Software Setup (1)



https://sysadmincasts.com/episodes/43-19-minutes-with-ansible-part-1-4

Ansible runs on a single node, and connects to all hosts under control via SSH.

No preparation is necessary on the target host, except for SSH access and Python 2.4+

Ansible for Software Setup (2)

Each *playbook* is a sequence of tasks.

All tasks are idempotent, hence all playbooks are idempotent.

Looping and conditional constructs allow (some) flexibility.

```
- name: Install required packages
  package:
    name: '{{item}}'
    state: 'latest'
  become: yes
  with_items:
   - auctex
   - emacs
   - evince
   - git

- name: Enable hibernation
  template:
    src: files/90-hibernate.conf
    dest: /etc/polkit-1/localauthori

- name: Make `apt-file` cache
  command: |
    apt-file update
  become: yes
```

Back to "What is ElastiCluster"