High Performance Data Analytics for Numerical Simulations

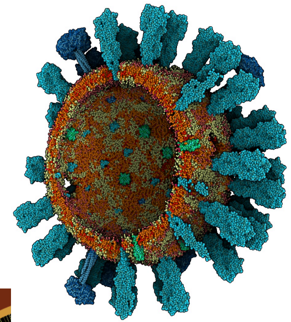
Bruno Raffin
DataMove

bruno.raffin@inria.fr

April 2016

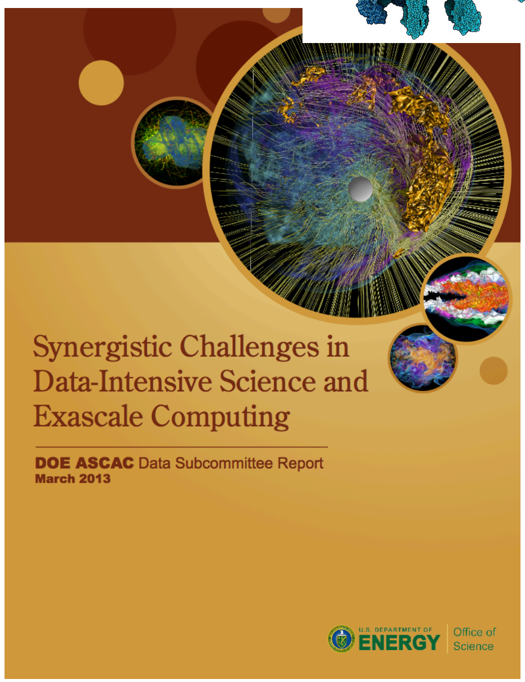
About this Talk

HPC for analyzing the results of large scale parallel numerical simulations (and not Big Data applications on HPC plateforms)

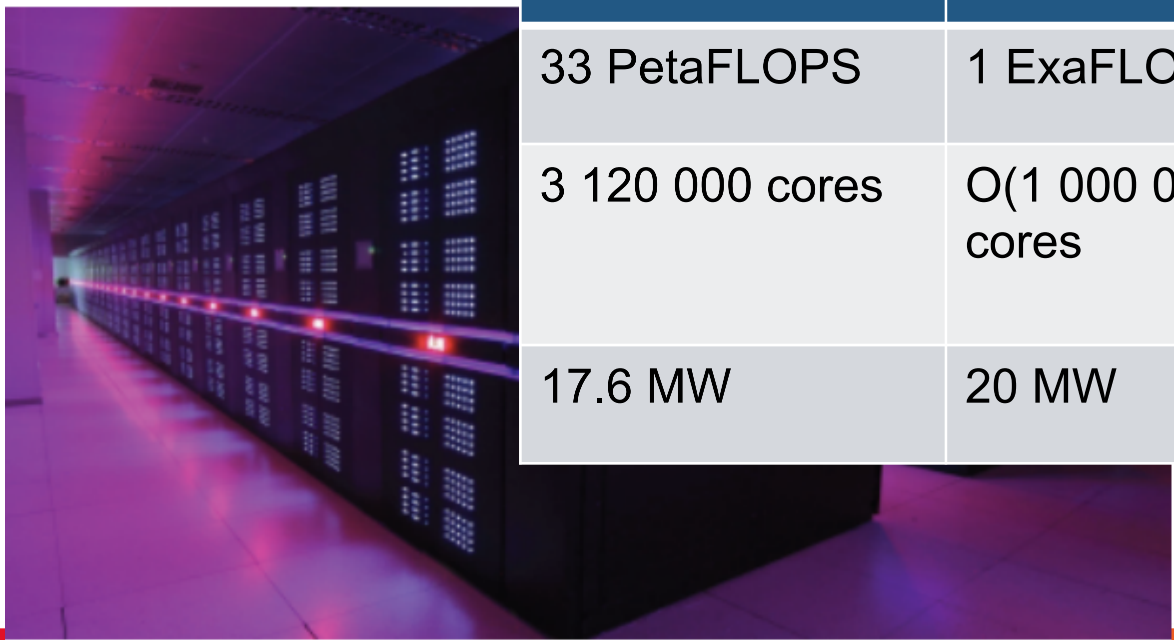


Most of my examples taken from molecular dynamics

Good overview document:
2013 DOE report on Synergistic Challenges in Data-Intensive Science and Exascale Computing



Exascale



2016 Tianhe-2 (China) #1 @ Top 500	2020 Exascale Machine
33 PetaFLOPS	1 ExaFLOPS
3 120 000 cores	O(1 000 000 000) cores
17.6 MW	20 MW

The Data Challenge

More compute capabilities -> larger simulations -> more data

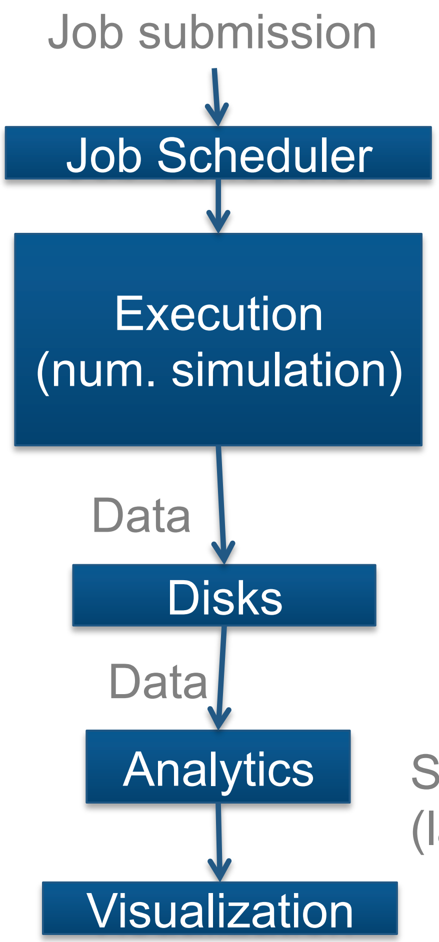
Usability Challenge:

- How to extract meaningful information from this huge amount of data in a reasonable time
- Analysis tools have not been considered as first class citizen so far. They did not receive the same as simulation codes. Today analysis codes are either:
  - In the simulation codes
  - Scripts (with limited parallelism)
  - Rely on on scientific visualization tools like Paraview/VTK or Visit (reasonable parallelism support)

Performance Challenge:

- Moving data becomes the bottleneck for simulation as well as data analytics
- Compute capabilities increase faster than data transfer ones
- Data movements and storage consume 50%-70% of total energy (ScidacReview 1001)

Traditional Workflow



Not sustainable at Exascale !

Big Machine: Simulation codes may include some analysis

Small Machine (laptop): Limited support for parallelization

A Data Challenge Already Present

Scientists already spend a significant part of their efforts in the data analysis:

Computational Biology:

- 2013 Molecular Dynamics Simulation wit Gromacs: 21’000’000 CPU hours (Curie supercomputer)
- More than 5 TB of data
- Analysis (VMD, MDAnalysis) still on-going work

Material Science:

- Molecular Dynamics Simulation with Stamps: 700 million atoms on 4096 cores, 1 million iterations
- Output: 1 every 10000 iteration, 100GB each
- Analysis (in-simulation code, Paraview/VTK):
  about 30% CPU wall clock time of the simulation time wall clock time.

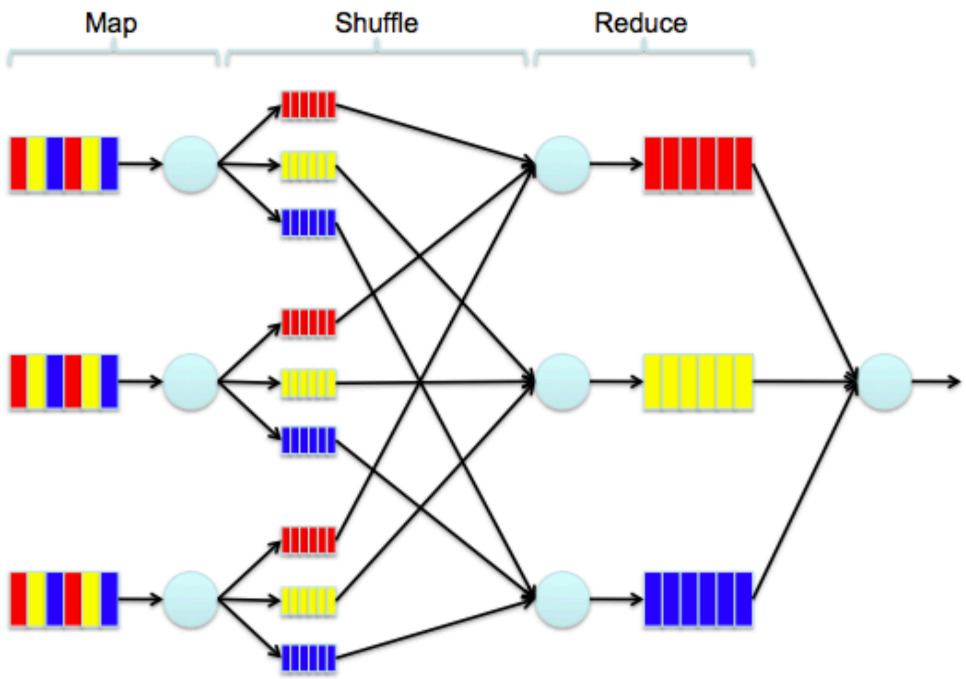
A simple but classical strategy to limit the impact of the data challenge:

Reduce output frequency

Big Data: Google Map/Reduce

Google Map/Reduce (2004):

- Two data parallel operators: map, reduce
- Values are indexed with a key (key/value model)
- Parallel execution on a cluster (distributed memory)
- Runtime takes care of tasks scheduling, load balancing and fault tolerance



Big Data: Beyond Map/Reduce

The original model has been extended in different ways (Spark, Flink) to support complex analysis plans:

- More operators (join, union,….)
- In-memory data store
- Iterative scripts
- Streaming (interactive scripts)

Augmented with specialization layers to support:

- SQL queries
- Large graph processing
- Machine learning

But tailored for:

- Running un cloud infrastructures (do not leverage supercomputers specifics)
- Process web data (web pages, tweets,…)

And Java based

HiMach [TU & al., HIMach, SC 2008]

A map/reduce like framework for analysing molecular dynamics trajectories

- Key/value store + map/reduce like operators
- Implementation:
  - Python + MPI
  - No fault tolerance
- Use VMD for some compute kernels
- Some analysis need only to keep one timestep at a time in memory (counting ion passing though a channel), other need a sliding window of timesteps (RMSD on a sliding window)

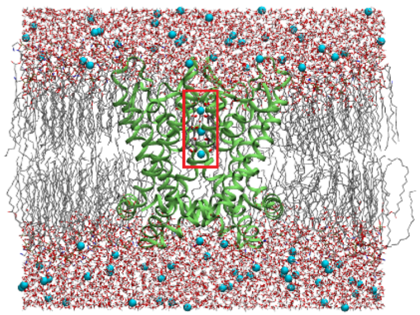
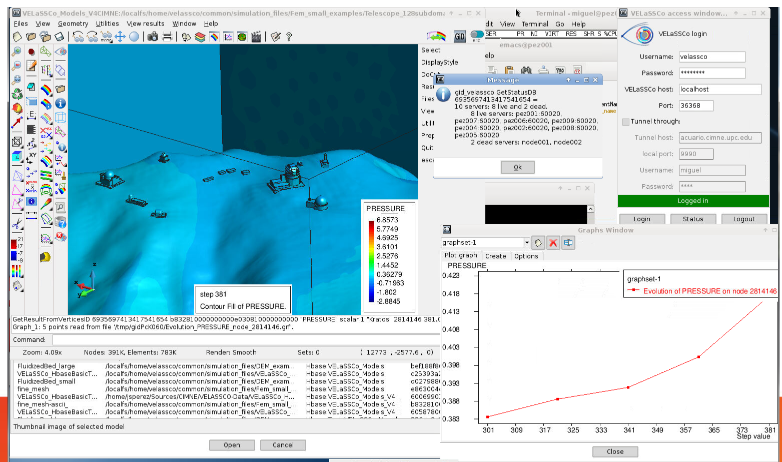


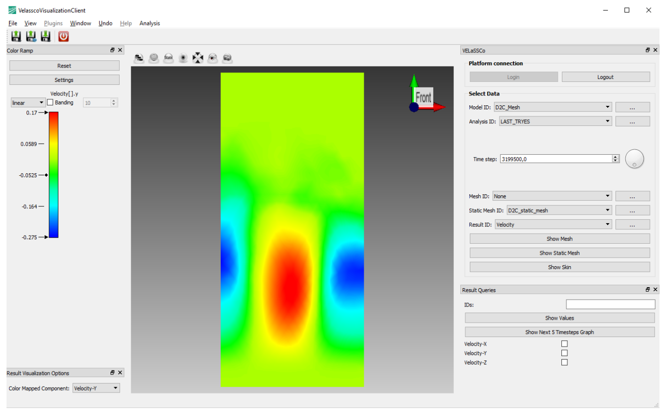
Figure 2. Ion permeation through a channel.

VelaSSco (FP7)

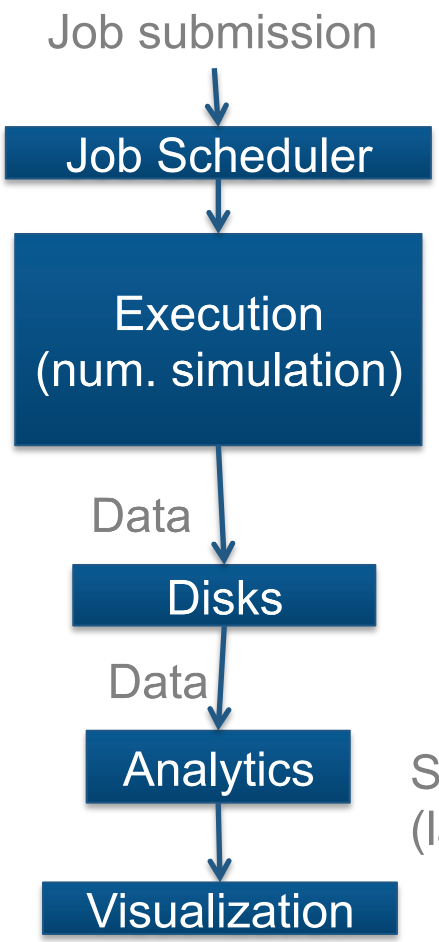
Query based Scientific Visualization

- FEM/DEM simulation data
- Hadoop software suite (MapReduce, HDFS, Hbase, Yarn, Thrift)
- Key/value: (timestep+rank-id, data)
- Scientist request some visualization (isosurface for a given timestep):
  - Vis client <-> front server <-> map/reduce job <-> HBASE





Traditional Workflow



Not sustainable at Exascale !

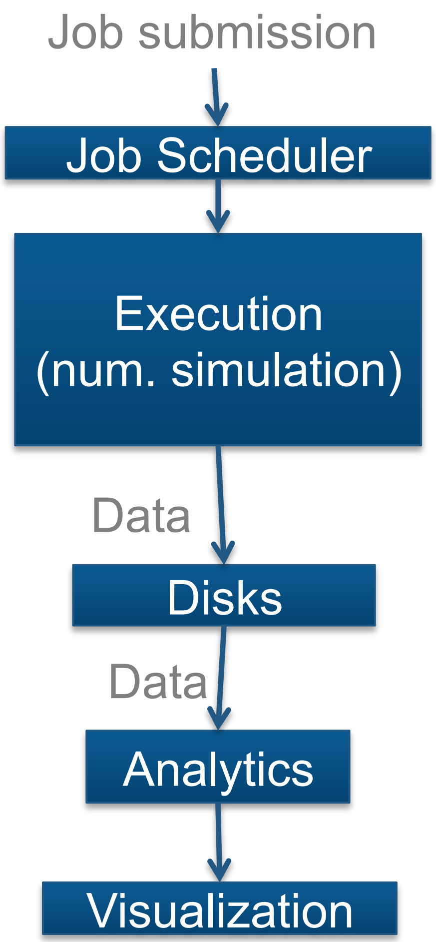
Big Machine

Simulation codes may include some analysis

Small Machine (laptop)

Limited support for parallelization

Workflow with Map/Reduce



Big Machine

Simulation codes may include some analysis

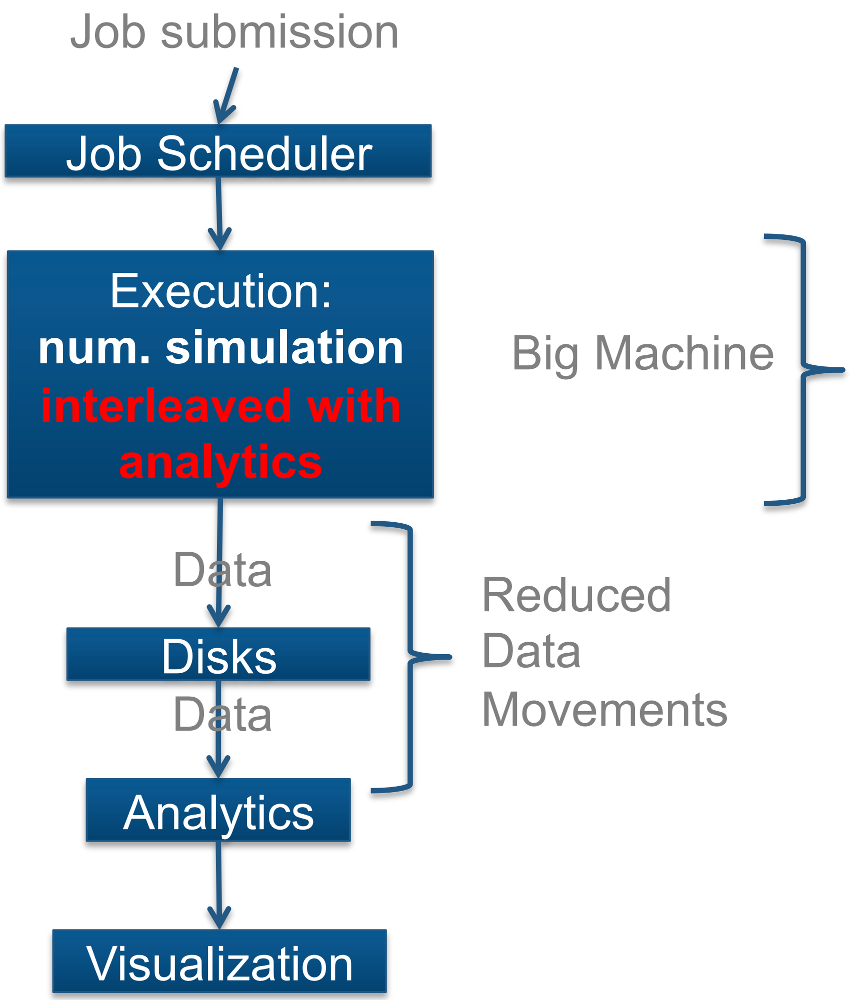
- Do not fix the data movement bottleneck

Cluster

Map/Reduce approach

+ High level parallel programming model

WorkFlow with In-situ Analytics



In-situ analytics:

- **Data reduction**
- **Large scale parallel analytics**
- **On-line monitoring**

In Situ Processing: What for ?

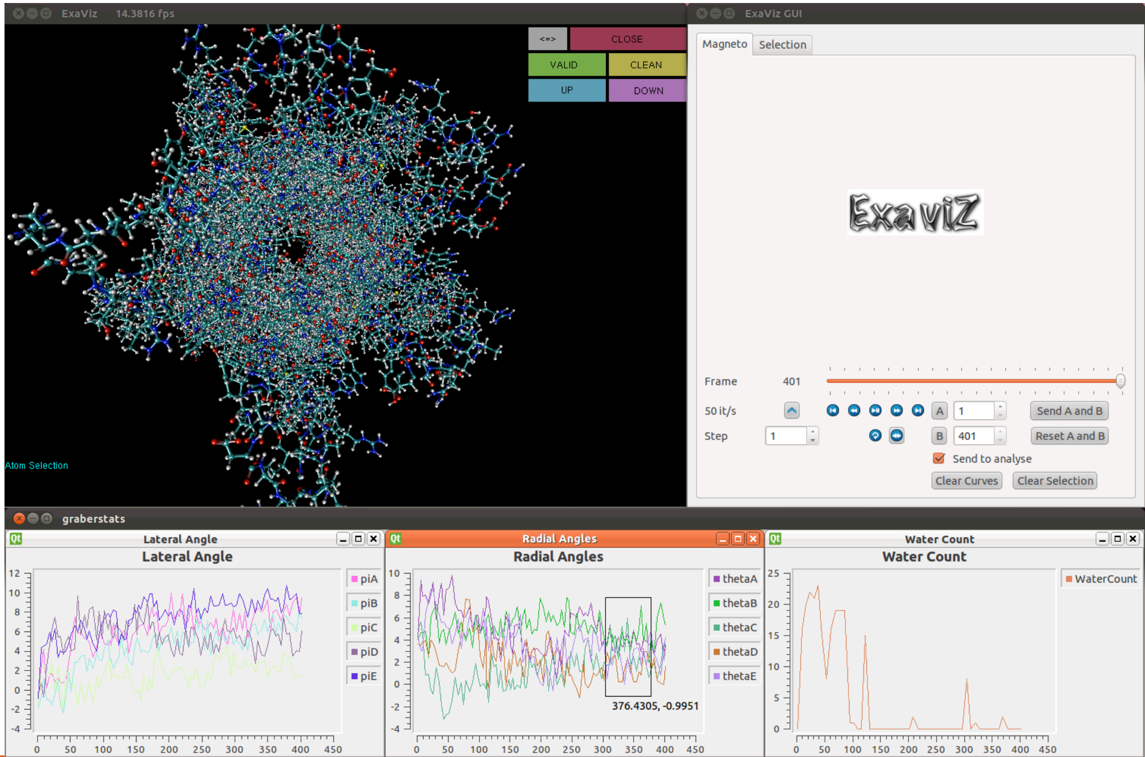
Data compression (Isabela [Lehmann & al. LDAV'14])

Indexing (FastBit, Dirac [Lakshminarasimhan & al. HPDC'13])

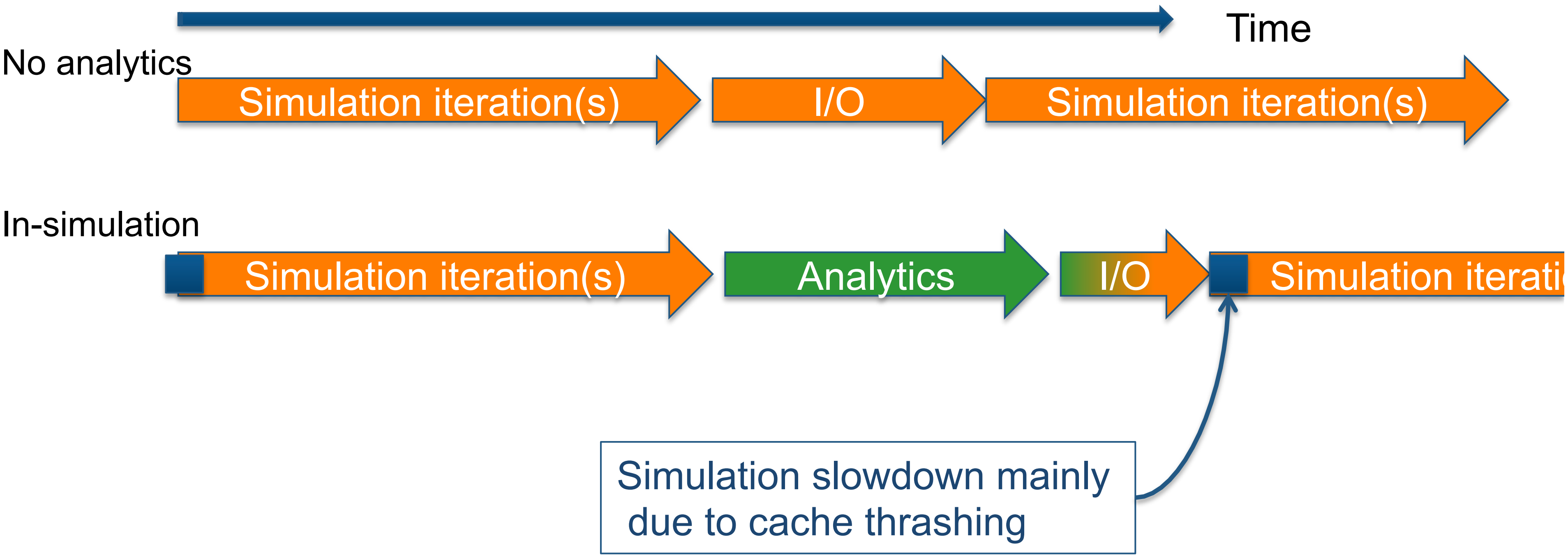
Storage (DataSpaces [Docan & al. Cluster Computing 12])

Analytics (1D, 2D, 3D descriptor computing)

[Dreher & al.
Faraday Discussion'14]

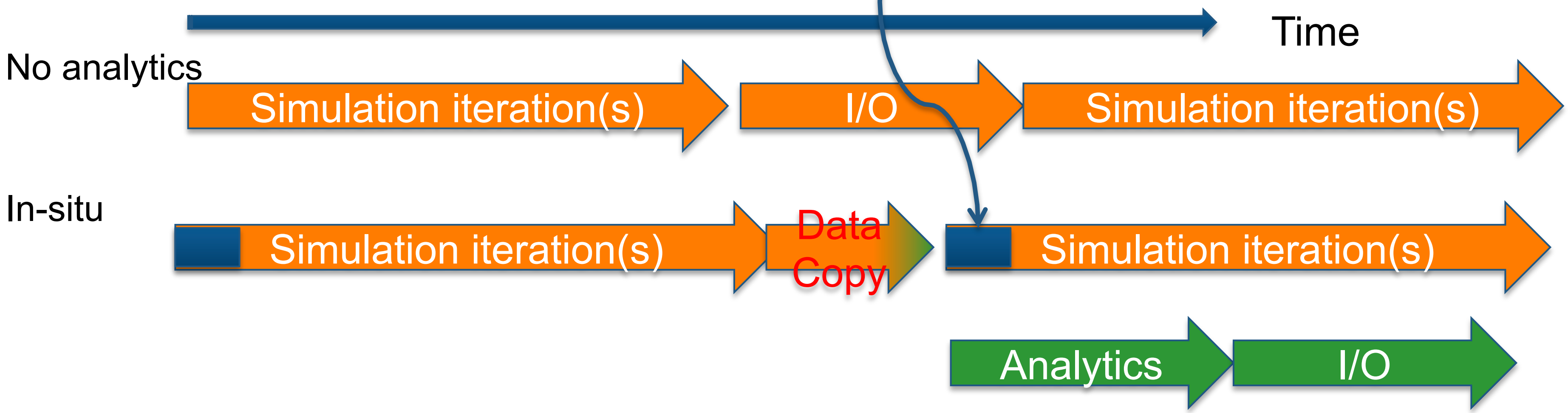


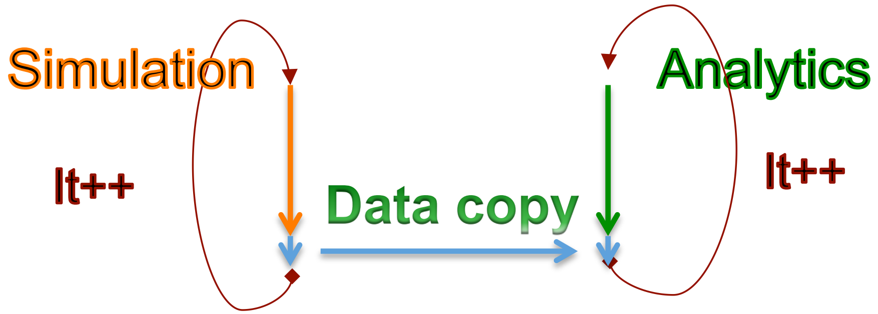
In-simulation Processing



In-situ Processing

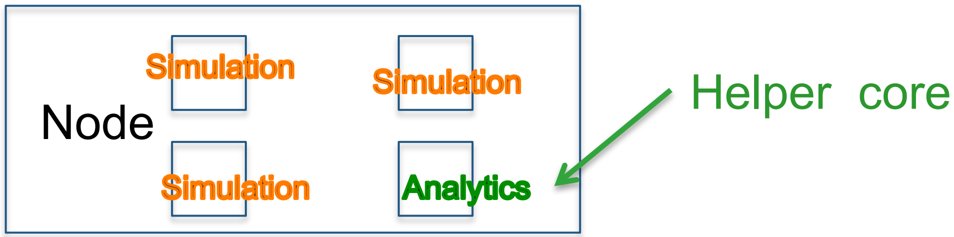
Simulation slowdown
due to concurrent use of some
resources with analytics and I/Os



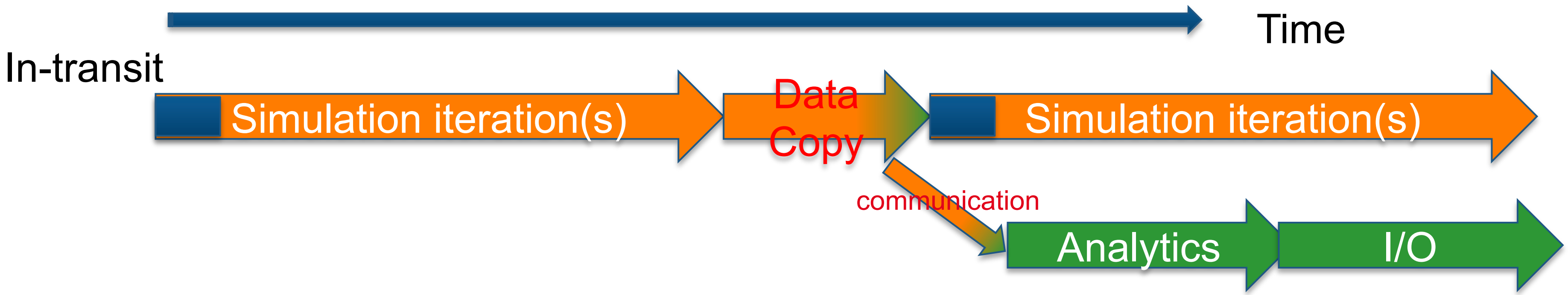


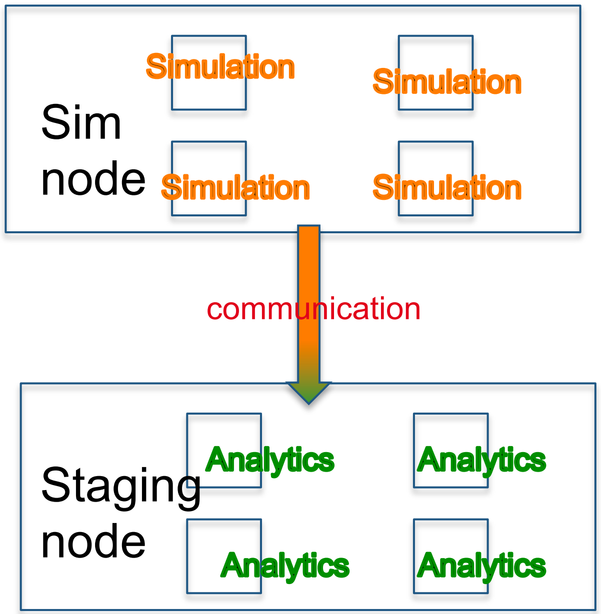
In-situ:
simulation and analytics share the same nodes

Resource allocation strategies:
time sharing or space sharing
(dedicated helper core)



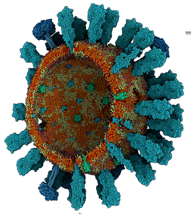
In-transit Processing

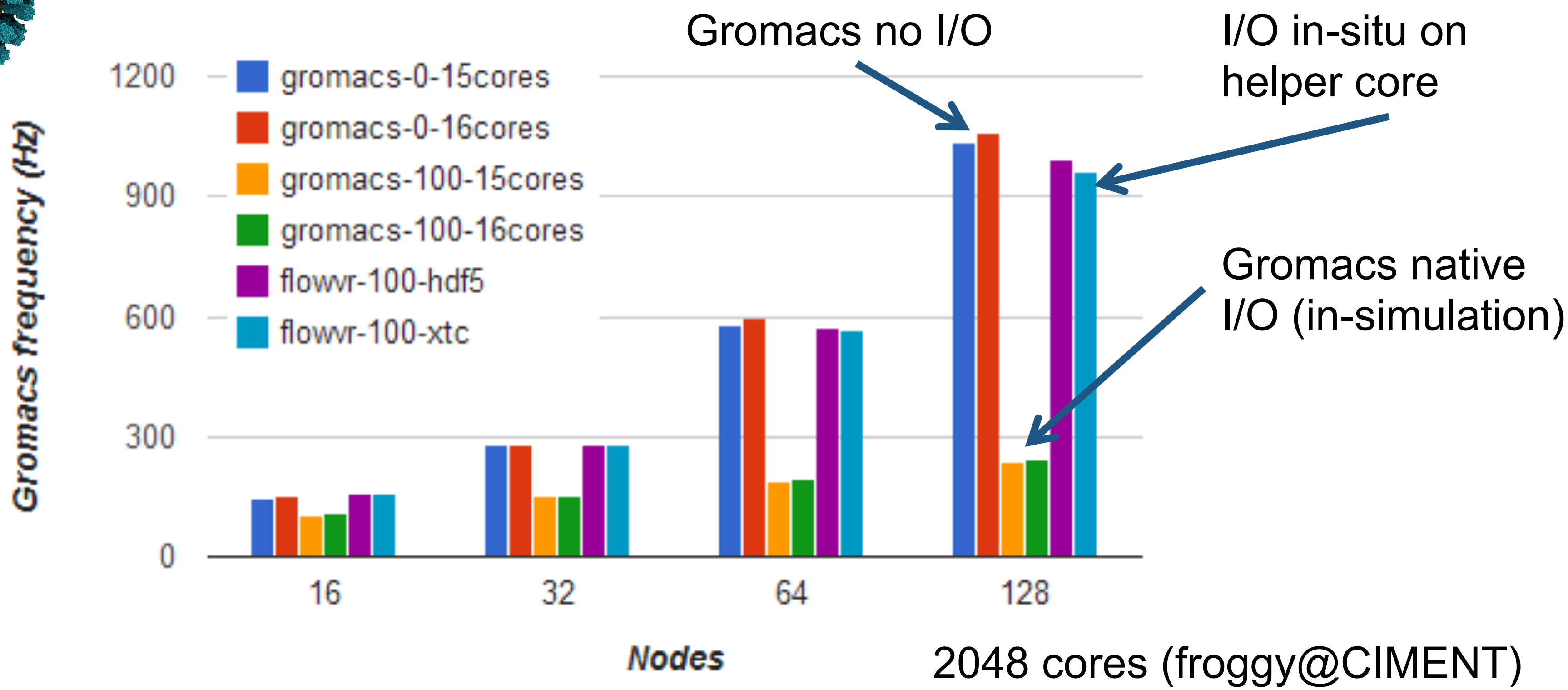




In-transit: simulation and Analytics run on different nodes (staging nodes)

In-Sim vs. In-Situ I/O [Dreher,CCGRID'14]

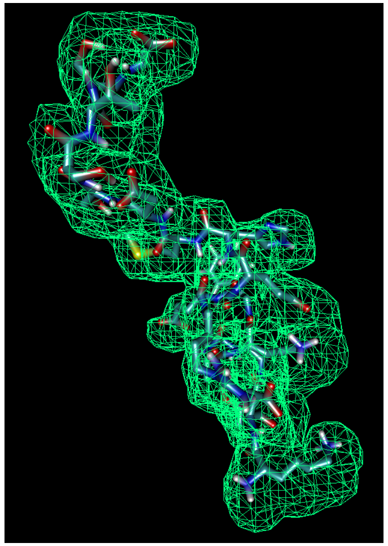




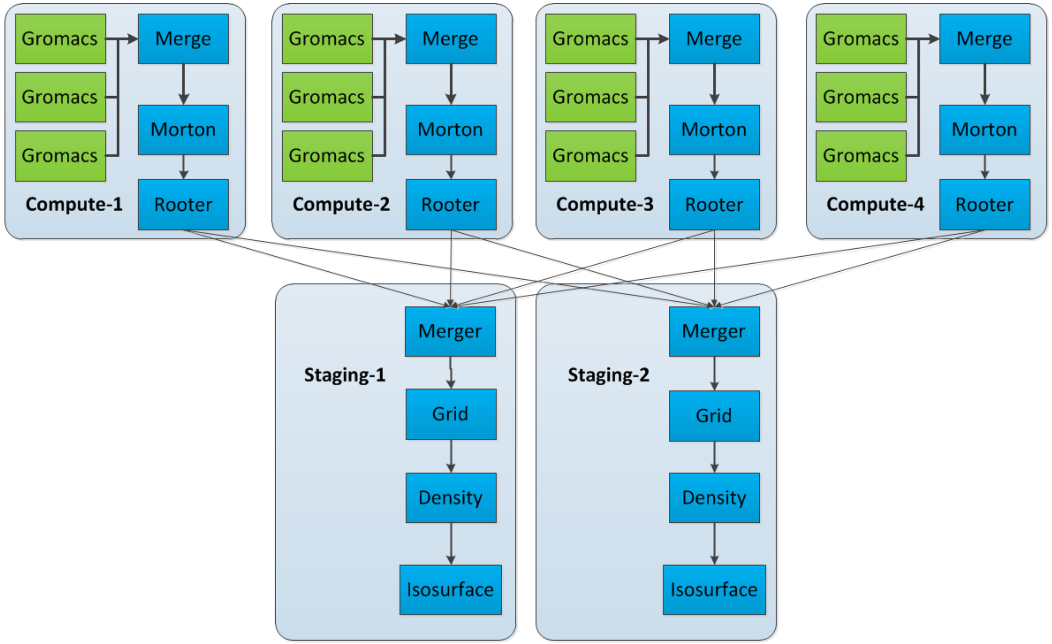
2048 cores (froggy@CIMENT)

Gromacs without I/O: 15 cores/node 3% slower than 16 cores/node
(- 6% if scalability would have been perfect)

Parallel In-Situ Isosurface Extraction [Dreher,CCGRID'14]

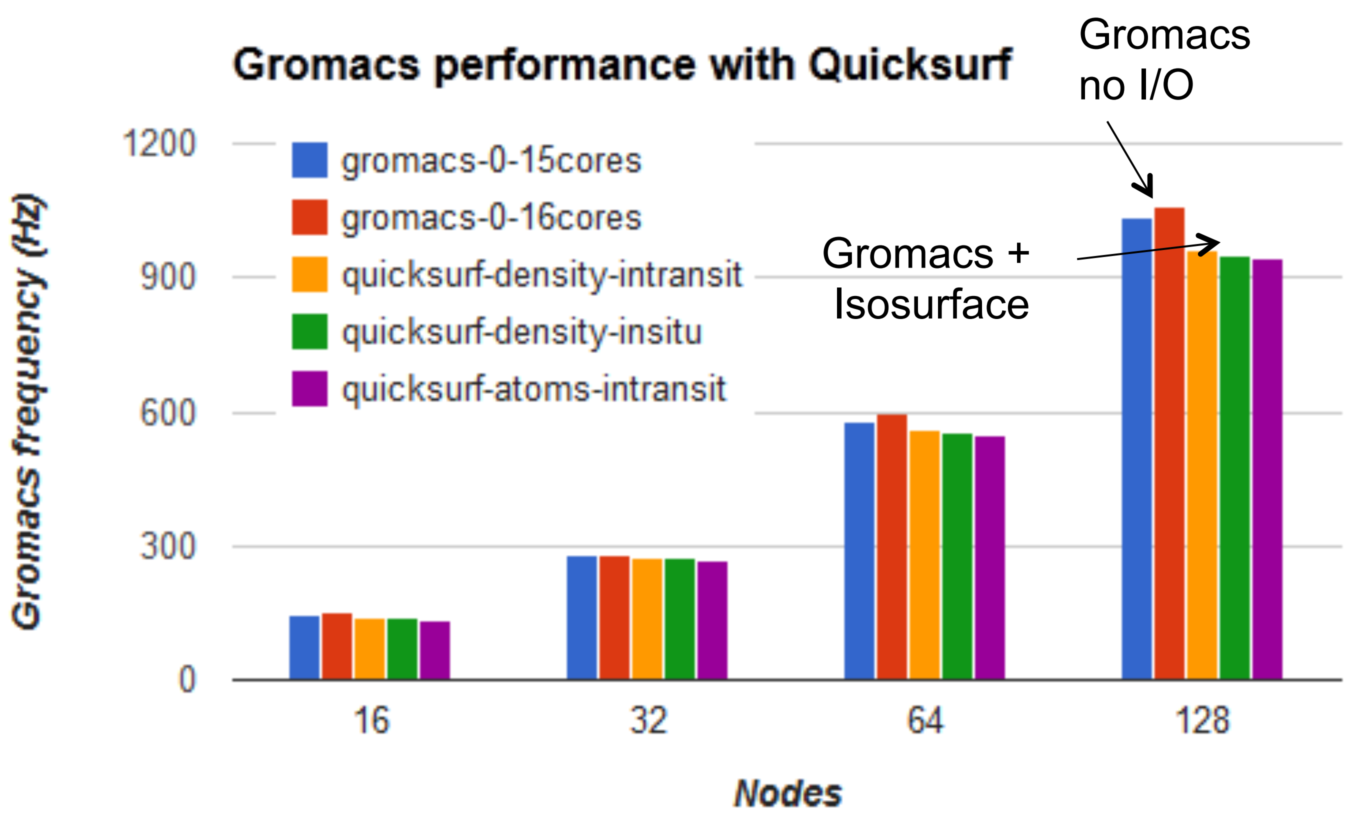


Compute a molecule surface based on atom density



Tested different distributions of processing steps to in-situ and in-transit nodes.

Performance [Dreher,CCGRID'14]



(froggy@CIMENT)

- In transit: 1 staging node every 64 compute nodes
- Density-intransit: costs 7% comp. to gromacs 15 cores
- Density-insitu costs 8% but use 1.5% less nodes than density-intransit
- Atoms-intransit costs 8.6% but enables other in-transit analytics (3x more data to move on stagging nodes than Density-intransit)

In-Situ Analytics Status

- Paraview and Visit: support in-simulation data processing
- Advanced prototypes supporting in-situ and in-transit:
  - FlexIO (IPDPS'13),
  - Damaris (Cluster'12),
  - FlowVR (CCGrid'14)
- In-memory data storage on staging nodes: DataSpace
- Programming model:
  - MPI level (Damaris)
  - In I/O library (ADIOS)
  - Data-flow (FlowVR)

No Standard Yet

Conclusion and Discussion

Map/Reduce model: successful in Big Data why not in HPC

- High level programming model, “efficient” executions

In-situ Analytics: a paradigm shift

- An opportunity to rethink the use of the I/O budget

In-situ versus post-mortem analysis:

- Different tools or same one ?
- Interface between the two words with an in-memory database (à la DataSpace) ?
- Programming model: Data flow oriented (à la Map/Reduce) or a more classical HPC appraoch (à la MPI) ?
- Reusing Big Data software stacks or need to develop HPC specific ones ?