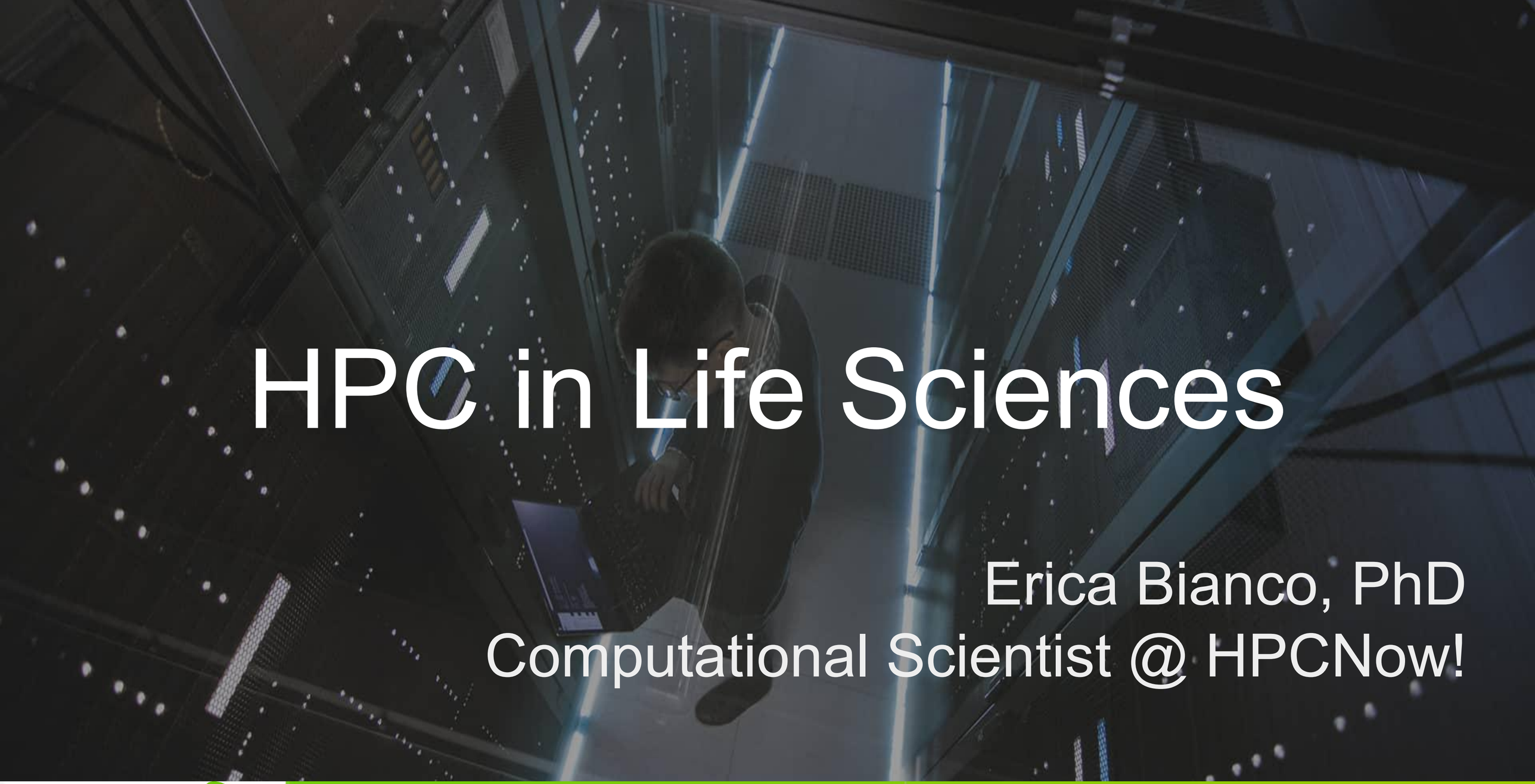


HPC in Life Sciences

Erica Bianco, PhD
Computational Scientist @ HPCNow!

Why use an HPC?

The key goal is to solve the problems faster.

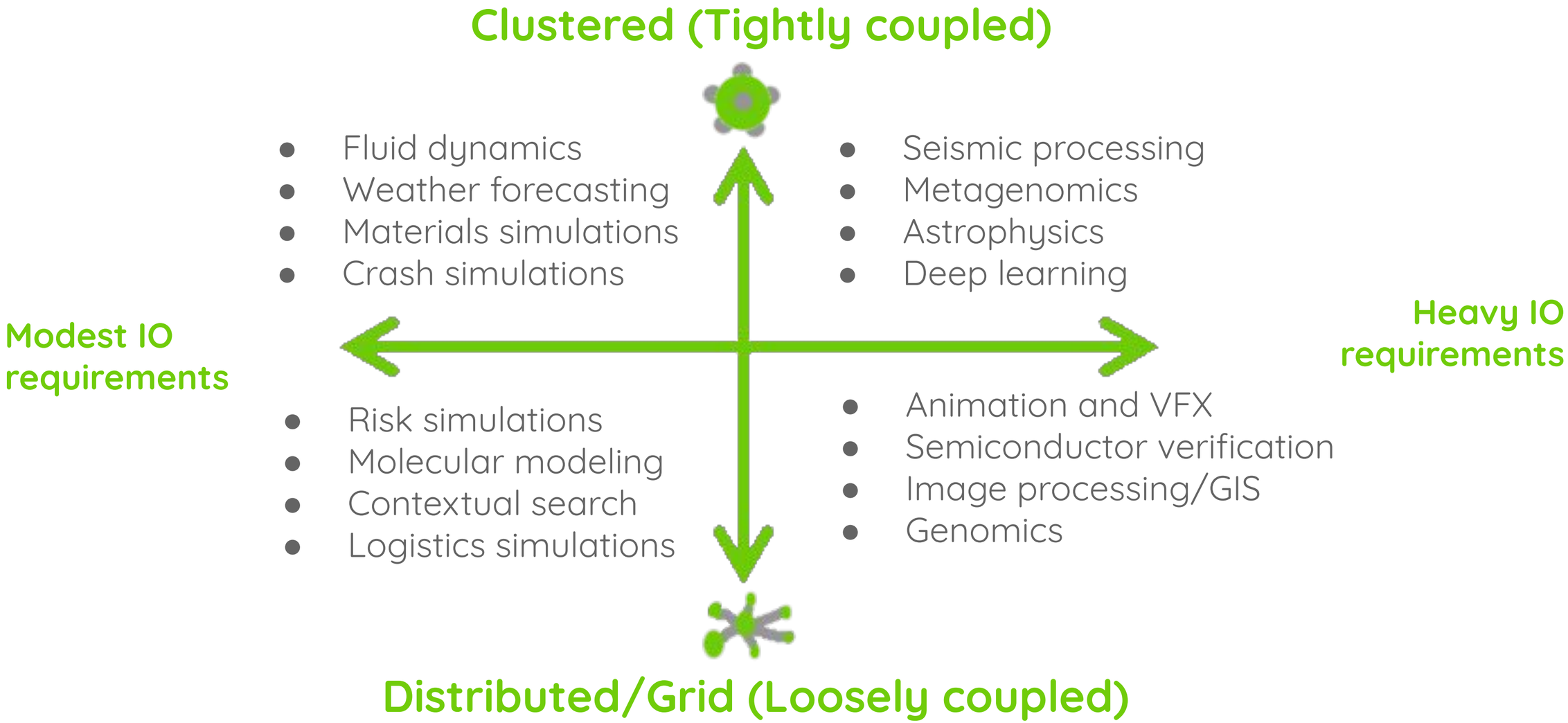
- Speed → Split the work between several processors : the program will run N-ish times faster by using N processors.
- Volume → large data analyses
- Cost → faster results, reduced wet lab analyses
- Efficiency and convenience → shared resources used 24/7, no need to use all resources of your own PC

Why use an HPC?

The key goal is to solv

- Speed → Split the w es faster by using N processors.
- Volume → large data
- Cost → faster result
- Efficiency and conve rces of your own PC

Main Use Cases for HPC



Main Use Cases for HPC

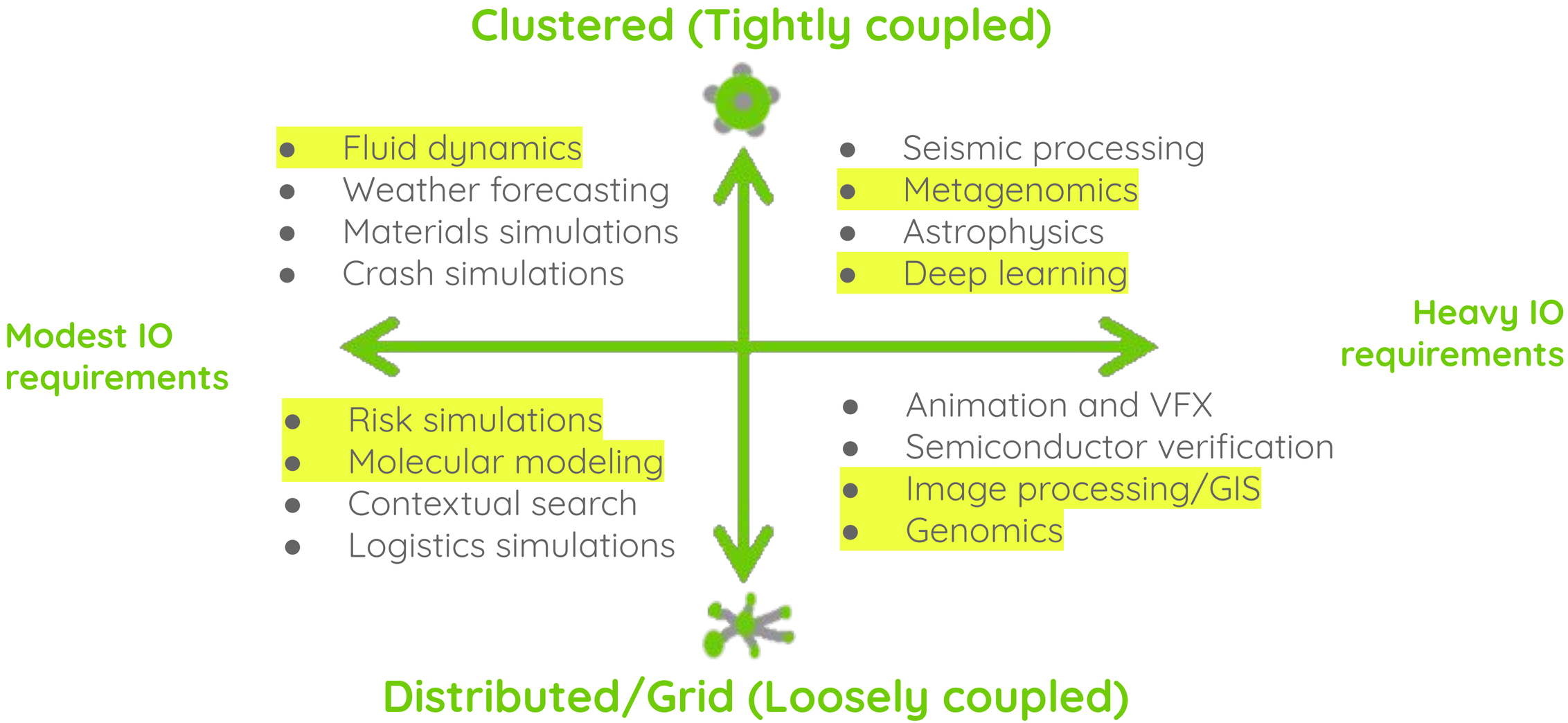


Table 1 High-performance computing architectures: advantages and drawbacks

HPC type	Architecture	Advantages	Drawbacks	Computing paradigm
Computer cluster	Set of interconnected computers controlled by a centralized scheduler	Require minimal changes to the existing source code of CPU programs, with the exception of possible modifications necessary for message passing	Expensive, characterized by relevant energy consumption and requires maintenance	Multiple Instruction Multiple Data
Grid computing	Set of geographically distributed and logically organized (heterogeneous) computing resources	Require minimal changes to the existing source code of CPU programs, with the exception of possible modifications necessary for message passing	Generally based on 'volunteering': computer owners donate resources (e.g. computing power, storage) to a specific project; no guarantee about the availability of remote computers: some allocated tasks could never be processed and need to be reassigned; remote computers might not be completely trustworthy	Multiple Instruction Multiple Data
Cloud computing	Pool of computation resources (e.g. computers, storage) offered by private companies, attainable on demand and ubiquitously over the Internet	Mitigate some problems like the costs of the infrastructure and its maintenance	Data are stored on servers owned by private companies; issues of privacy, potential piracy, espionage, international legal conflicts, continuity of the service (e.g. owing to some malfunctioning, DDoS attacks, or Internet connection problems)	Multiple Instruction Multiple Data
GPU - Graphics Processing Units	Dedicated parallel co-processor, formerly devoted to real-time rendering of computer graphics, nowadays present in every common computer	High number of programmable computing units allow the execution of thousands simultaneous threads. Availability of high-performance local memories	Based on a modified SIMD computing paradigm: conditional branches imply serialization of threads' execution. GPU's peculiar architecture generally requires code rewriting and algorithms redesign	Same Instruction Multiple Data (although temporary divergence is allowed)
MIC - Many Integrated Cores	Dedicated parallel co-processor installable in common desktop computers, workstations and servers	Similar to GPUs but based on the conventional ×86 instructions set: existing CPU code, in principle, might be ported without any modification. All cores are independent	Fewer cores with respect to latest GPUs. To achieve GPU-like performances, modification of existing CPU code to exploit vector instructions are required	Multiple Instruction Multiple Data
FPGA - Field Programmable Gates Arrays	Integrated circuits containing an array of programmable logic blocks	Able to implement a digital circuit, which directly performs purpose-specific tasks (unlike general-purpose software tools). Such tasks are executed on a dedicated hardware without any computational overhead (e.g. those related to the operating system)	Generally programmed using a descriptive language (e.g. VHDL, Verilog [18]), which can be cumbersome. Debugging using digital circuits simulators might be complicated and not realistic. Experience with circuit design optimization might be necessary to execute tasks using the highest clock frequency	Dedicated hardware

Nobile, et al., 2017

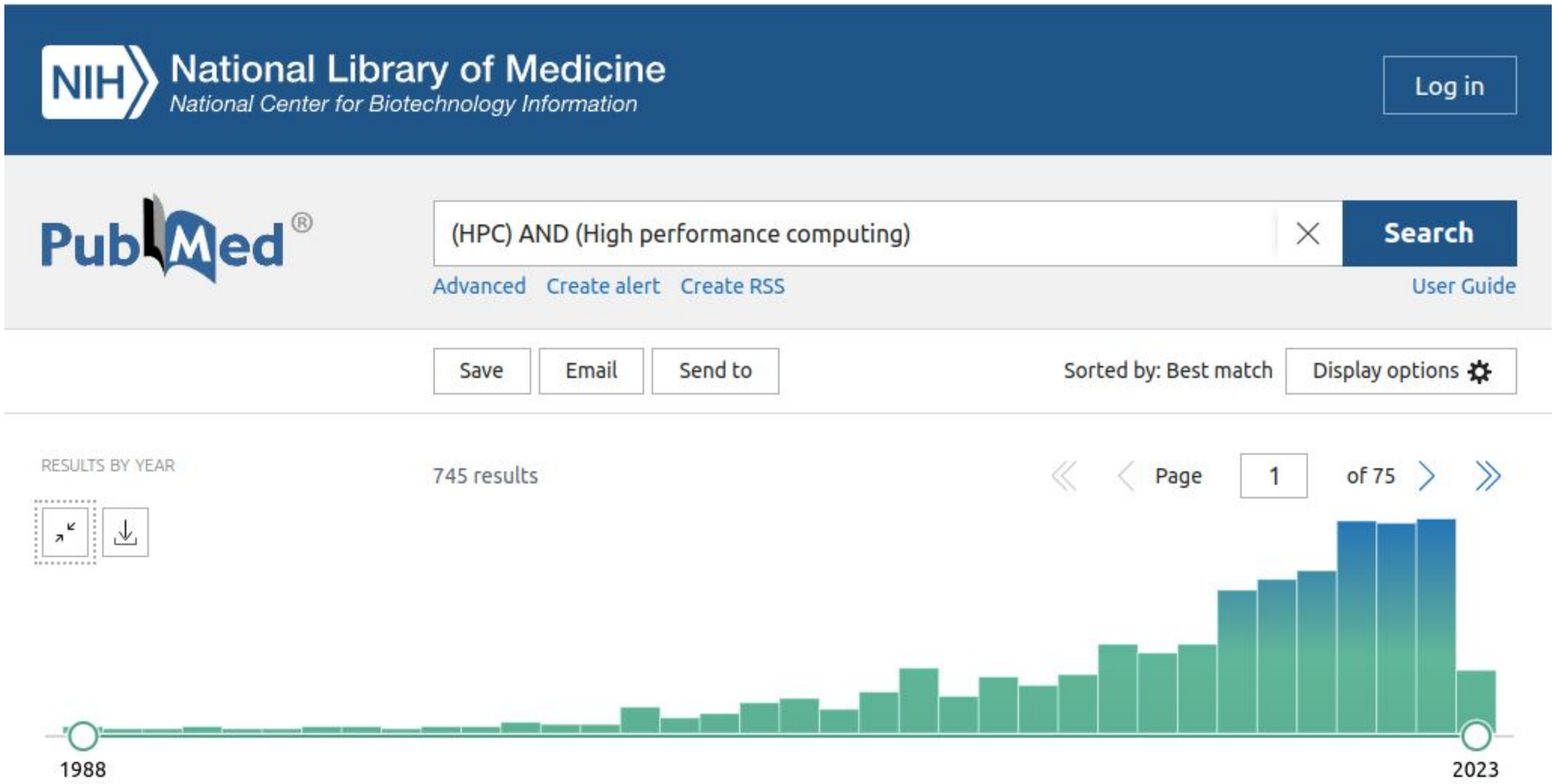
Table 1 High-performance computing architectures: advantages and drawbacks

HPC type	Architecture	Advantages	Drawbacks	Computing paradigm
Computer cluster	Set of interconnected computers controlled by a centralized scheduler	Require minimal changes to the existing source code of CPU programs, with the exception of possible modifications necessary for message passing	Expensive, characterized by relevant energy consumption and requires maintenance	Multiple Instruction Multiple Data
Grid computing	Set of geographically distributed and logically organized (heterogeneous) computing resources	Require minimal changes to the existing source code of CPU programs, with the exception of possible modifications necessary for message passing	Generally based on 'volunteering': computer owners donate resources (e.g. computing power, storage) to a specific project; no guarantee about the availability of remote computers: some allocated tasks could never be processed and need to be reassigned; remote computers might not be completely trustworthy	Multiple Instruction Multiple Data
Cloud computing	Pool of computation resources (e.g. computers, storage) offered by private companies, attainable on demand and ubiquitously over the Internet	Mitigate some problems like the costs of the infrastructure and its maintenance	Data are stored on servers owned by private companies; issues of privacy, potential piracy, espionage, international legal conflicts, continuity of the service (e.g. owing to some malfunctioning, DDoS attacks, or Internet connection problems)	Multiple Instruction Multiple Data
GPU - Graphics Processing Units	Dedicated parallel co-processor, formerly devoted to real-time rendering of computer graphics, nowadays present in every common computer	High number of programmable computing units allow the execution of thousands simultaneous threads. Availability of high-performance local memories	Based on a modified SIMD computing paradigm: conditional branches imply serialization of threads' execution. GPU's peculiar architecture generally requires code rewriting and algorithms redesign	Same Instruction Multiple Data (although temporary divergence is allowed)
MIC - Many Integrated Cores	Dedicated parallel co-processor installable in common desktop computers, workstations and servers	Similar to GPUs but based on the conventional ×86 instructions set: existing CPU code, in principle, might be ported without any modification. All cores are independent	Fewer cores with respect to latest GPUs. To achieve GPU-like performances, modification of existing CPU code to exploit vector instructions are required	Multiple Instruction Multiple Data
FPGA - Field Programmable Gates Arrays	Integrated circuits containing an array of programmable logic blocks	Able to implement a digital circuit, which directly performs purpose-specific tasks (unlike general-purpose software tools). Such tasks are executed on a dedicated hardware without any computational overhead (e.g. those related to the operating system)	Generally programmed using a descriptive language (e.g. VHDL, Verilog [18]), which can be cumbersome. Debugging using digital circuits simulators might be complicated and not realistic. Experience with circuit design optimization might be necessary to execute tasks using the highest clock frequency	Dedicated hardware

Nobile, et al., 2017

HPC in Life Sciences recent papers :

Queried NCBI PubMed on April 11, 2023



Some HPC in Life Sciences papers

www.hpcnow.com

OXFORD JOURNALS | Briefings in Bioinformatics

Brief Bioinform. 2017 Sep; 18(5): 870–885.
Published online 2016 Jul 7. doi: 10.1093/bib/bbw058

PMCID: PMC
PMID: 2

Graphics processing units in bioinformatics, computational biology and systems biol

Editorial

Is high performance computing a requirement for novel drug discovery and how will this impact academic efforts?

Savíns Puertas-Martín, Antonio J. Banegas-Lun
Juana L. Redondo, Pilar M. Ortigosa, Ol'ha
Pages 981-985 | Received 16 Oct 2019, Accepted 17 Apr 202

Download citation | https://doi.org/10.1080/174

Editorial | Published: 18 July 2018

Special Issue on High Performance Computing in Bio-medical Informatics

Luping Zhou, Islem Rekik, Chenggang Yan & Guorong Wu

Neuroinformatics **16**, 283 (2018) | Cite this article

2370 Accesses | **2** Citations | Metrics

Open Access | Article

High Performance Computing PP-Distance Algorithms to Generate X-ray Spectra from 3D Models

by César González [1,*], Simone Balocco [2], Jaume Bosch [3], Juan Miguel de Haro [3], Maurizio Paolini [4], Antonio Filgueras [3], Carlos Álvarez [3] and Ramon Pons [1,*]

1 Institut de Química Avançada de Catalunya (IQAC-CSIC), 08034 Barcelona, Spain
2 Department of Mathematics and Informatics, Universitat de Barcelona, 08007 Barcelona, Spain
3 Barcelona Supercomputing Center (BSC), 08034 Barcelona, Spain
4 INTEL, 20090 Assago, Italy
* Authors to whom correspondence should be addressed.

Research | Open Access | Published: 16 December 2022

Profiling the BLAST bioinformatics application for load balancing on high-performance computing clusters

Trinity Cheng, Pei-Ju Chin, Kenny Cha, Nicholas Petrick & Mike Mikailov

BMC Bioinformatics **23**, Article number: 544 (2022) | Cite this article

987 Accesses | **4** Altmetric | Metrics

BIOLOGY Methods & Protocols | OXFORD UNIVERSITY PRESS

Biol Methods Protoc. 2022; 7(1): bpac032.
Published online 2022 Nov 15. doi: 10.1093/biomethods/bpac032

PMCID: PMC9767868
PMID: 36561335

Teaching computational genomics and bioinformatics on a high performance computing cluster—a primer

Arun Sethuraman

Immune Repertoire Analysis on High-Performance Computing Using VDJServer V1: A Method by the AIRR Community

Scott Christley, Ulrik Stervbo & Lindsay G. Cowell on beha

Protocol | Open Access | First Online: 28 May 2022

4292 Accesses

Part of the Methods in Molecular Biology book series (MIMB

tps://doi.org/10.3390/ijms231911408

20 September 2022 / Accepted: 22 September 2022 / Published: 27 September 2022

Exscalate4CoV: Innovative High Performing Computing (HPC) Strategies to Tackle Pandemic Crisis

by Andrea R. Beccari [1] and Giulio Vistoli [2,*]

1 EXSCALATE, Dompé Farmaceutici S.p.A., Via Tommaso De Amicis 95, I-80131 Napoli, Italy
2 Dipartimento di Scienze Farmaceutiche, Università degli Studi di Milano, Via Mangiagalli 25, I-20133 Milano, Italy
* Author to whom correspondence should be addressed.

igh-Performance Computing in Cardiovascular Medicine of Multi-Dimensional Data Analysis-

re, Shinichi Goto

Open Access | Published: 03 May 2018

Towards Portable Large-Sc High-Performance Computing

Yuankai Huo, Justin Blaber, Stephen M. Damon, Brian D. Boyd, Shun
Parvathaneni, Camilo Bermudez Noguera, Shikha Chaganti, Vishwesh N
Greer, Ilwoo Lyu, William R. French, Allen T. Newton, Baxter P. Rogers,
Landman

J Med Libr Assoc. 2018 Oct; 106(4): 494–495.
Published online 2018 Oct 1. doi: 10.5195/jmla.2018.512

Journal o

High-performance computing service for bioinformatic

Jean-Paul Courneya

Bioinformationist, Health Sciences and Human Services Library, Univ

Alexa Mayo, AHIP

Associate Director for Services, Health Sciences and Human Service
21201

Hyperparameter Tuning with I Learning for Imbalanced Alzheimer's Disease Data

Int. J. Mol. Sci. **2022**, *23*(19), 11576; https://doi.org/10.3390/ijms231911576

by Fan Zhang [1,2,*], Melissa Petersen [1,2], Leigh Johnson [1,3], James Hall [1,2] and Sid E. O'Bryant [1,2]

1 Institute for Translational Research, University of North Texas Health Science Center, Fort Worth, TX 76107, USA
2 Department of Family Medicine, University of North Texas Health Science Center, Fort Worth, TX 76107, USA
3 Department of Pharmacology and Neuroscience, University of North Texas Health Science Center, Fort Worth, TX 76107, USA
* Author to whom correspondence should be addressed.

Appl. Sci. **2022**, *12*(13), 6670; https://doi.org/10.3390/app12136670

Biocomputing 2023, pp. 541-545 (2022)

HIGH-PERFORMANCE COMPUTING MEETS HIGH-PERFORMANCE MEDICINE

Anurag Verma, Jennifer Huffman, Ali Torkamani, and Ravi Madduri

https://doi.org/10.1142/9789811270611_0050 | **Cited by:** 0

ar dinamics, Artificial intelligence, Neural network, Machine learning

High-Performance Computing in Bayesian Phylogenetics and Phylodynamics Using BEAGLE

Guy Baele, Daniel L. Ayres, Andrew Rambaut, Marc A. Suchard & Philippe Lemey

Open Access | First Online: 06 July 2019

sses | **4** Citations | **8** Altmetric

e Methods in Molecular Biology book series (MIMB,volume 1910)

Some HPC+AI+ML in Life Sciences papers

Advancing biomolecular simulation through exascale HPC, AI and quantum computing

Edward O Pyzer-Knapp [1], Alessandro Curioni [2]

Affiliations + expand

PMID: 38733863 DOI: 10.1016/j.sbi.2024.102826

Free article

AI/ML-Based Medical Image Processing and Analysis

Jaafar Alghazo [1], Ghazanfar Latif [2]

Affiliations + expand

PMID: 38132255 PMCID: PMC10742629 DOI: 10.3390/diagnostics13243671

Precision medicine in the era of artificial intelligence: implications in chronic disease management

Murugan Subramanian [1 2], Anne Wojtusciszyn [3], Lucie Favre [3], Sabri Boughorbel [4], Jingxuan Shan [2 5], Khaled B Letaief [6], Nelly Pitteloud [7], Lotfi Chouchane [8 9 10]

Affiliations + expand

PMID: 33298113 PMCID: PMC7725219 DOI: 10.1186/s12967-020-02658-5

HPC+ in the medical field: Overview and current examples

Miriam Koch [1], Claudio Arlandini [2], Gregory Antonopoulos [3], Alessia Baretta [4], Pierre Beaujean [5], Geert Jan Bex [6], Marco Evangelos Biancolini [7], Simona Celi [8], Emiliano Costa [9], Lukas Drescher [10], Vasileios Eleftheriadis [11], Nur A Fadel [10], Andreas Fink [10], Federica Galbiati [9], Ilias Hatzakis [12], Georgios Hompis [3], Natalie Lewandowski [1], Antonio Memmolo [2], Carl Mensch [13], Dominik Obrist [14], Valentina Paneta [11], Panagiotis Papadimitroulas [11], Konstantinos Petropoulos [3], Stefano Porziani [7], Georgios Savvidis [11], Khyati Sethia [15], Petr Strakos [15], Petra Svobodova [15], Emanuele Vignali [8]

Affiliations + expand

PMID: 36641699 PMCID: PMC10357138 DOI: 10.3233/THC-229015

Developing Edge AI Computer Vision for Smart Poultry Farms Using Deep Learning and HPC

Stevan Cakic [1 2], Tomo Popovic [1 2], Srdjan Krco [3], Daliborka Nedic [3], Dejan Babic [1], Ivan Jovovic [1]

Affiliations + expand

PMID: 36991712 PMCID: PMC10055782 DOI: 10.3390/s23063002

Transforming medical sciences with high-performance computing, high-performance data analytics and AI

Natalie Lewandowski, Bastian Koller

PMID: 37355917 DOI: 10.3233/THC-237000

Biomolecular applications optimized for HPC

ACEMD	A molecular dynamics software particularly optimized for NVIDIA GPUs.	Optimized for GPUs, primarily NVIDIA.	[11]
AMBER	A suite of biomolecular simulation programs emphasizing on molecular dynamics.	Supports both CPU multi-core parallelism (via MPI) and GPU acceleration.	[12]
ADF	A quantum chemistry program focused on molecular properties and electronic structure.	Supports multi-core parallelism.	[13]
CHARMM	A molecular simulation program with extensive force fields for biomolecular systems.	Supports CPU multi-node parallelism via MPI.	[14]
CP2K	A quantum mechanics and molecular dynamics program for atomistic simulations.	Supports CPU multi-node parallelism via MPI and GPU acceleration.	[15]
CPMD	The CPMD code is a parallelized plane wave/pseudopotential implementation of Density Functional Theory, particularly designed for ab-initio molecular dynamics.	Supports CPU multi-node parallelism via MPI and GPU acceleration.	[16]
Desmond	A molecular dynamics software developed by D.E. Shaw Research.	Optimized for both multi-core CPUs and GPUs, and scales well on supercomputers.	[17]
GROMACS	A versatile package to perform molecular dynamics and energy minimization.	Well-known for CPU and GPU scalability, can run efficiently on both supercomputers and workstation clusters.	[18]
GENESIS	A software package for scalable MD simulations focused on parallelisation	Well-known for CPU and GPU scalability	[19]
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator for molecular dynamics.	Highly scalable on supercomputers with support for multi-node CPU parallelism and some GPU acceleration.	[20]
OpenMM	A toolkit for high-performance molecular simulations.	Primarily optimized for GPU acceleration, also supports multi-core CPUs.	[21]
NAMD	NAMD is a molecular dynamics program	NAMD is known for its scalability on supercomputers, supporting both CPU multi-node parallelism and GPU acceleration.	[22]
TeraChem	A quantum chemistry software optimized for GPU acceleration.	Optimized primarily for GPUs.	[23]

Pyzer-Knapp and Curioni, 2024

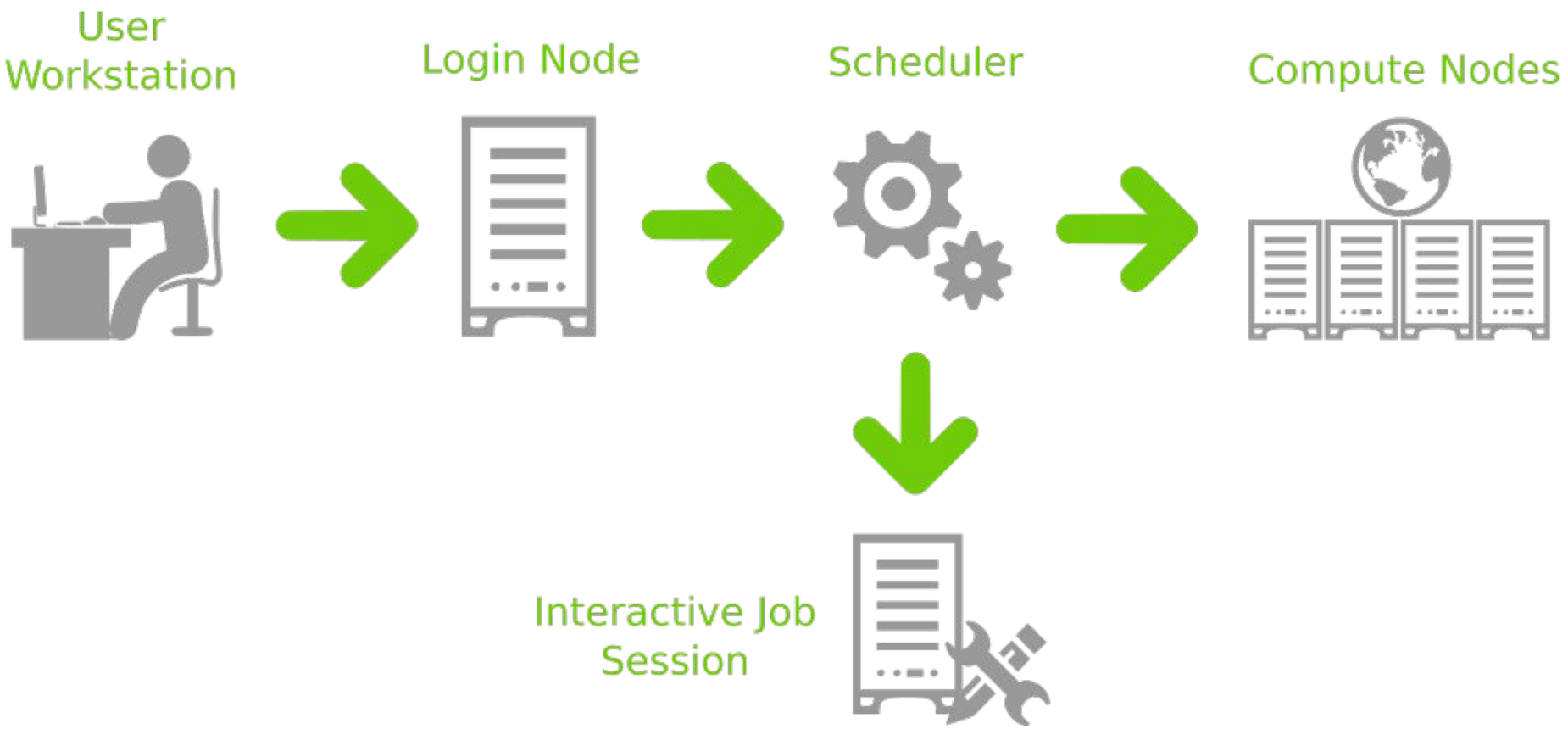
Some topics covered and tools used

- protein folding
- drugs design
- genomics
  - WGS
  - RNAseq
  - proteome
  - metabolome
- ecology
- large datasets
- precision medicine
- ML / AI
- simulations

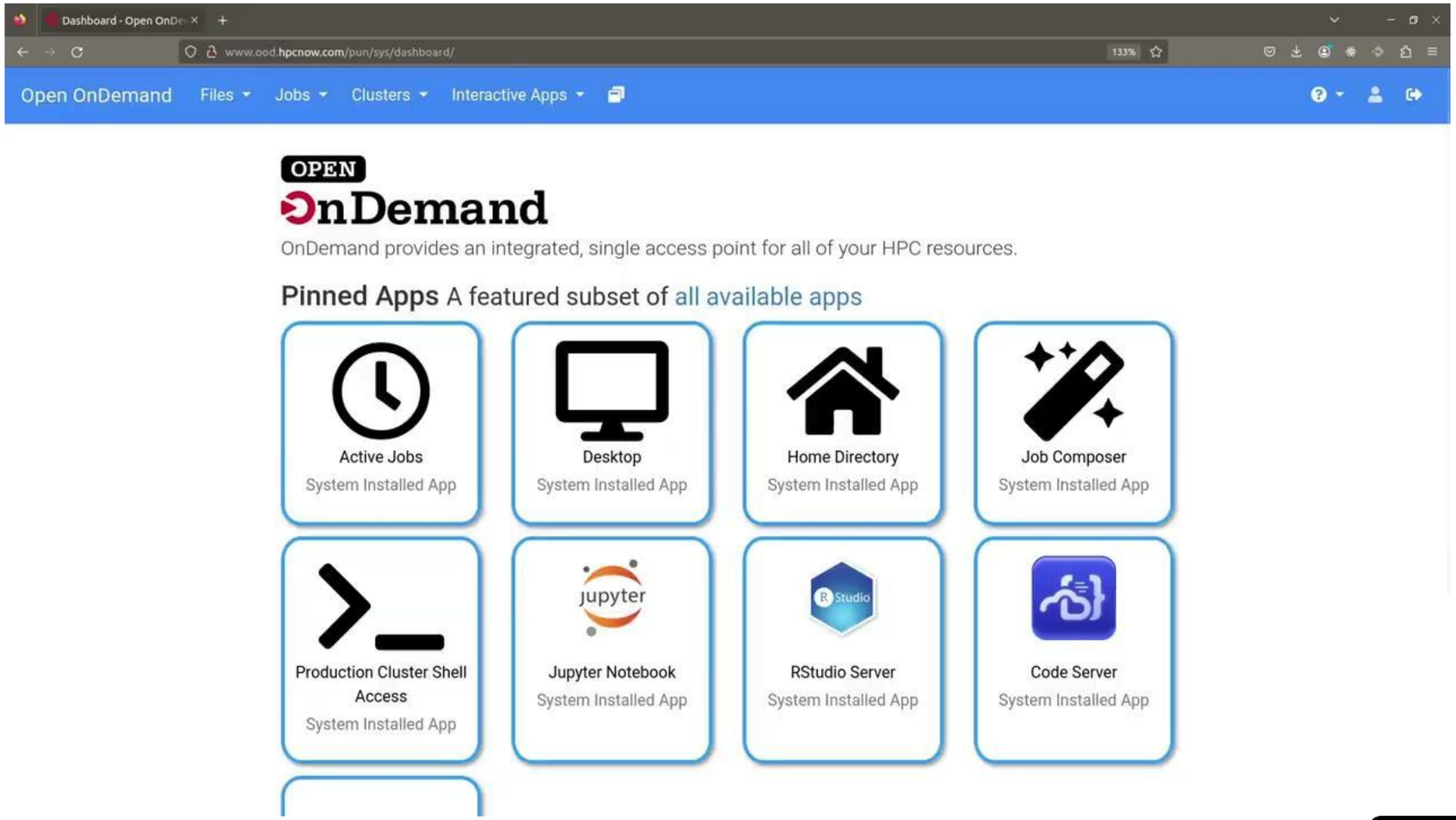
- AlphaFold
- Nextflow - workflow managers
- CI/CD
- containers

- Quantum computing
- AI
- cloud vs on-prem
- GPU
- Future directions in HPC

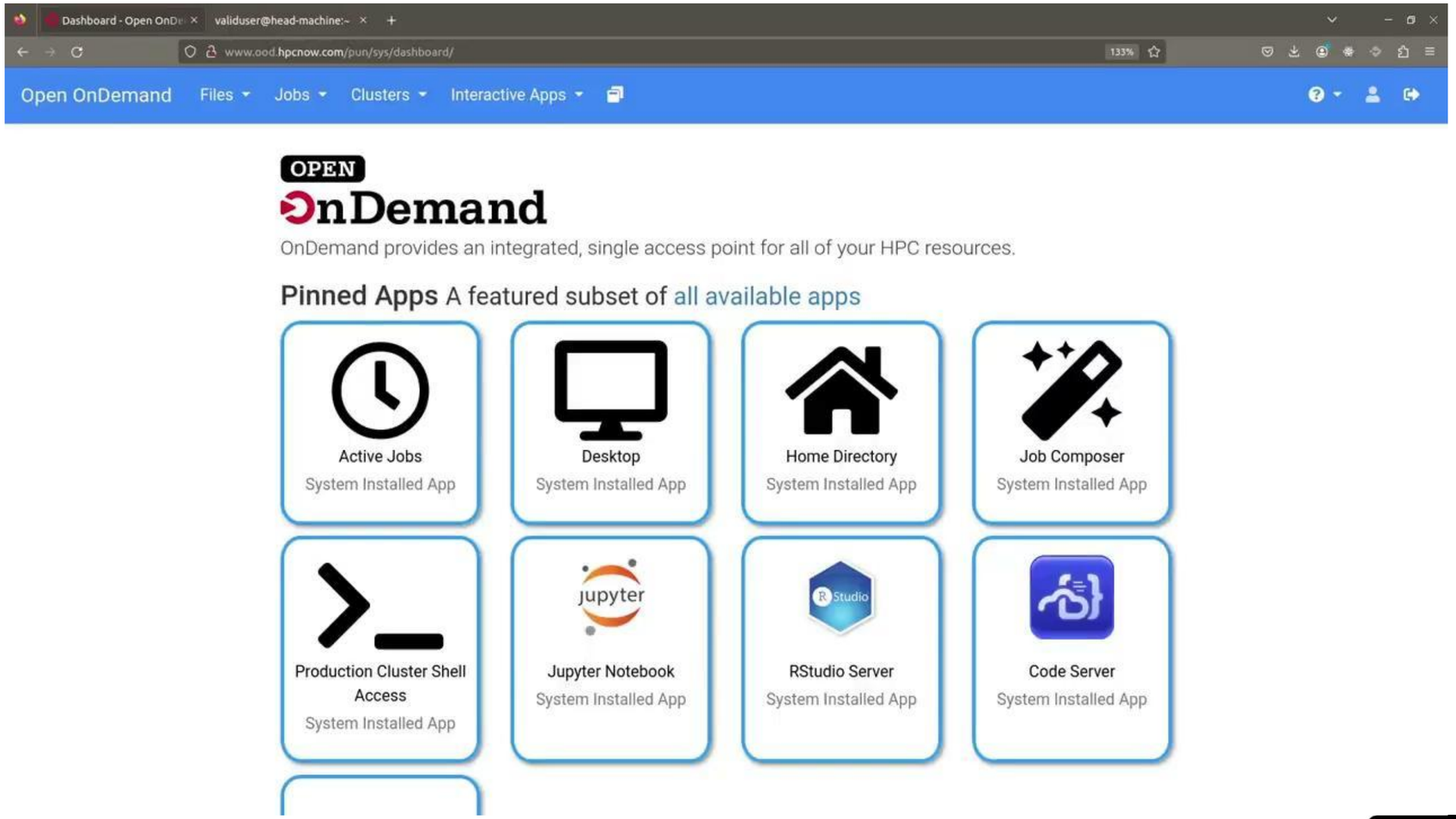
User vs. HPC



User vs. HPC



User vs. HPC



How to run 4*100k simulations in few days

And what happens when you accidentally delete ¼ of your temporary results

https://doi.org/10.1093/molbev/msaa156

HPC Now!

Run 4*100k simulations in few days

- when to make the job start (waiting in the queue)
- "auto"launching launcher and set a waiting time
- Keep working on the cluster while simulating and fairplay
- Control in real time how much memory your job is using

wflow_fsc_cluster.sh (ebianco on marvin.s.upf.edu /gpfs42/robbyfs/homes/users/ebianco/SCRIPTS) - gedit

Open Save

```
#!/bin/bash
#mar 17 abr 2018 10:43:18 CEST
#script to:
# (1) run fastsimcoal simulation
# (2) send the script to convert apr file to vcf individually
#      (or in blocks defined by variable $rep, default is rep=10 )
#to launch it
# sbatch -t 30-00 -x mr-01-[01-02] -J fsc.$model ~/SCRIPTS/wflow_fsc_cluster.sh $model
# squeue -o "%.18i %.9P %.8j %.8T %.10M " -u ebianco

#SBATCH -o fsc.%j.out
#SBATCH -e fsc.%j.err
#SBATCH --job-name=fsc
#SBATCH --cpus-per-task=12
#SBATCH --partition=normal
#SBATCH --mem-per-cpu=4g
#SBATCH --mail-type=all
# SBATCH --mail-type=fail

module load fastsimcoal2/2.6.0.3

model=$1

SCRIPTF="/homes/users/ebianco/SCRIPTS"

#check if variables are set correctly
if [[ ! $model ]]
then
echo -e "ERROR: model is not specified. \nScript must be run as \nsbatch wflow_fsc.sh \$model \$numsim \$numsamp \$rep[optional]"
exit 1
fi

#run fastsimcoal after checking if model files are present in running folder
# if [[ -a $model.tpl && -a $model.est ]]
# then
time fsc26 -t $model.tpl -n 15 -e $model.est -E 25000 -q -c 12

rm -f $model/sust.*.out
# else
# echo "ERROR: no $model.tpl or $model.est file in running folder."
# exit 1
# fi

exit 0
```

sh Tab Width: 8 Ln 38, Col 24 INS

When to make a job start (waiting in the queue)

sbatch --dependency=after:$j launcher_sust.sh $model $m

When to make a job start (waiting in the queue)

sbatch --dependency=after:$j launcher_sust.sh $model $m

-d, **--dependency**=<*dependency_list*>

Defer the start of this job until the specified dependencies have been satisfied completed. <*dependency_list*> is of the form <*type:job_id[:job_id][,type:job_id[:job_id]]*> or <*type:job_id[:job_id][?type:job_id[:job_id]]*>. All dependencies must be satisfied if the "," separator is used. Any dependency may be satisfied if the "?" separator is used. Many jobs can share the same dependency and these jobs may even belong to different users. The value may be changed after job submission using the scontrol command. Once a job dependency fails due to the termination state of a preceding job, the dependent job will never be run, even if the preceding job is requeued and has a different termination state in a subsequent execution.

after:job_id[:jobid...]
This job can begin execution after the specified jobs have begun execution.

afterany:job_id[:jobid...]
This job can begin execution after the specified jobs have terminated.

afterburstbuffer:job_id[:jobid...]
This job can begin execution after the specified jobs have terminated and any associated burst buffer stage out operations have completed.

aftercorr:job_id[:jobid...]
A task of this job array can begin execution after the corresponding task ID in the specified job has completed successfully (ran to completion with an exit code of zero).

afternotok:job_id[:jobid...]
This job can begin execution after the specified jobs have terminated in some failed state (non-zero exit code, node failure, timed out, etc).

afterok:job_id[:jobid...]
This job can begin execution after the specified jobs have successfully executed (ran to completion with an exit code of zero).

expand:job_id
Resources allocated to this job should be used to expand the specified job. The job to expand must share the same QOS (Quality of Service) and partition. Gang scheduling of resources in the partition is also not supported.

singleton
This job can begin execution after any previously launched jobs sharing the same job name and user have terminated.

"Auto"launching launcher and set a waiting time

sbatch --begin=now+5minutes launcher_sust.sh $model $m

"Auto"launching launcher and set a waiting time

sbatch --begin=now+5minutes launcher_sust.sh $model $m

--begin=<*time*>

Submit the batch script to the Slurm controller immediately, like normal, but tell the controller to defer the allocation of the job until the specified time.

Time may be of the form *HH:MM:SS* to run a job at a specific time of day (seconds are optional). (If that time is already past, the next day is assumed.) You may also specify *midnight*, *noon*, *fika* (3 PM) or *teatime* (4 PM) and you can have a time-of-day suffixed with *AM* or *PM* for running in the morning or the evening. You can also say what day the job will be run, by specifying a date of the form *MMDDYY* or *MM/DD/YY YYYY-MM-DD*. Combine date and time using the following format *YYYY-MM-DD[THH:MM[:SS]]*. You can also give times like *now* + *count time-units*, where the time-units can be *seconds* (default), *minutes*, *hours*, *days*, or *weeks* and you can tell Slurm to run the job today with the keyword *today* and to run the job tomorrow with the keyword *tomorrow*. The value may be changed after job submission using the **scontrol** command. For example:

```
--begin=16:00
--begin=now+1hour
--begin=now+60           (seconds by default)
--begin=2010-01-20T12:34:00
```

Notes on date/time specifications:

- Although the 'seconds' field of the HH:MM:SS time specification is allowed by the code, note that the poll time of the Slurm scheduler is not precise enough to guarantee dispatch of the job on the exact second. The job will be eligible to start on the next poll following the specified time. The exact poll interval depends on the Slurm scheduler (e.g., 60 seconds with the default sched/builtin).
- If no time (HH:MM:SS) is specified, the default is (00:00:00).
- If a date is specified without a year (e.g., MM/DD) then the current year is assumed, unless the combination of MM/DD and HH:MM:SS has already passed for that year, in which case the next year is used.

Keep working on the cluster while simulating and fairplay

```
sbatch -x mr-01-[01-02] --time=30 --array=0-9 -J sust.${m}.${model} \

~/SCRIPTS/wflow_sust_cluster.sh  $model $mult
```

Keep working on the cluster while simulating and fairplay

sbatch -x mr-01-[01-02] --time=30 --array=0-9 -J sust.${m}.${model} \

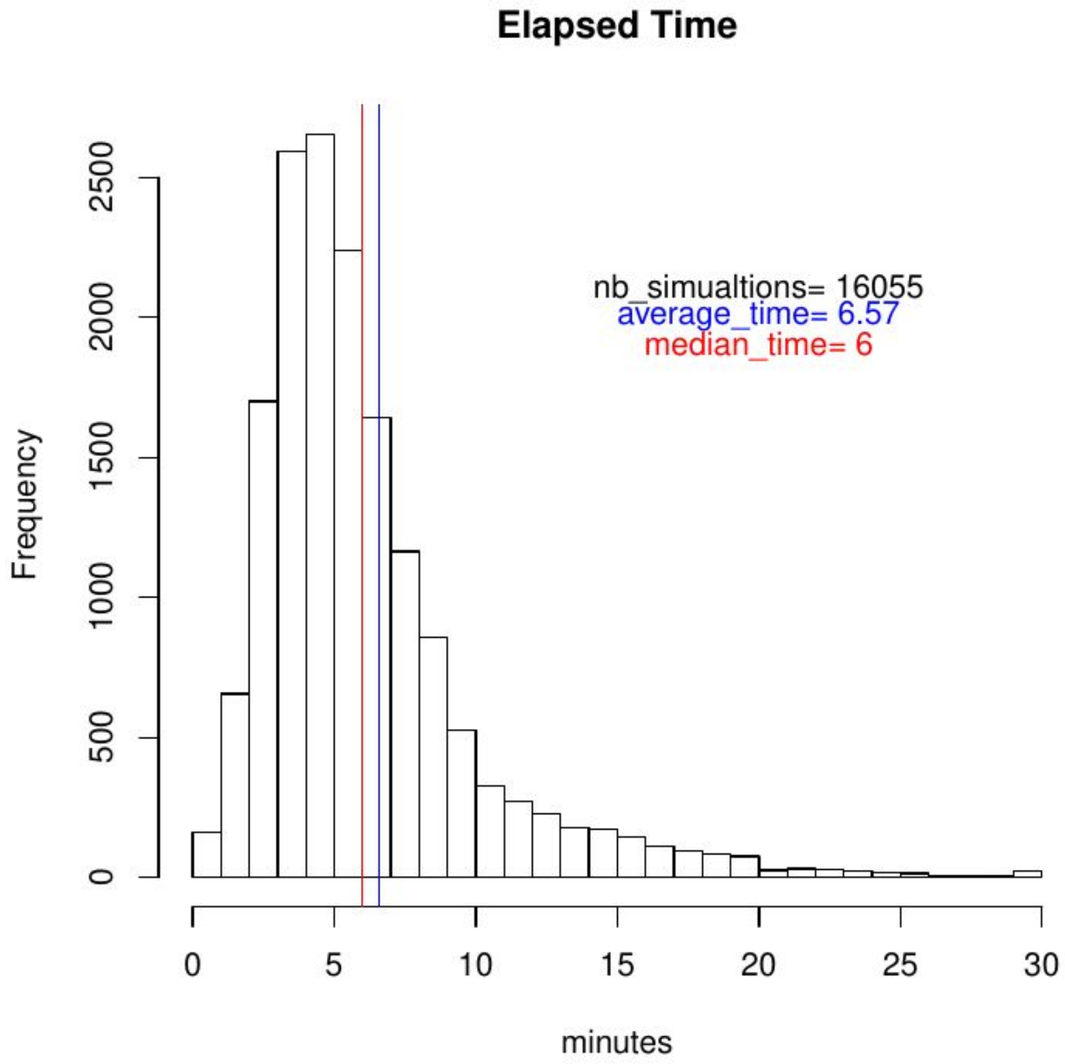
~/SCRIPTS/wflow_sust_cluster.sh $model $mult

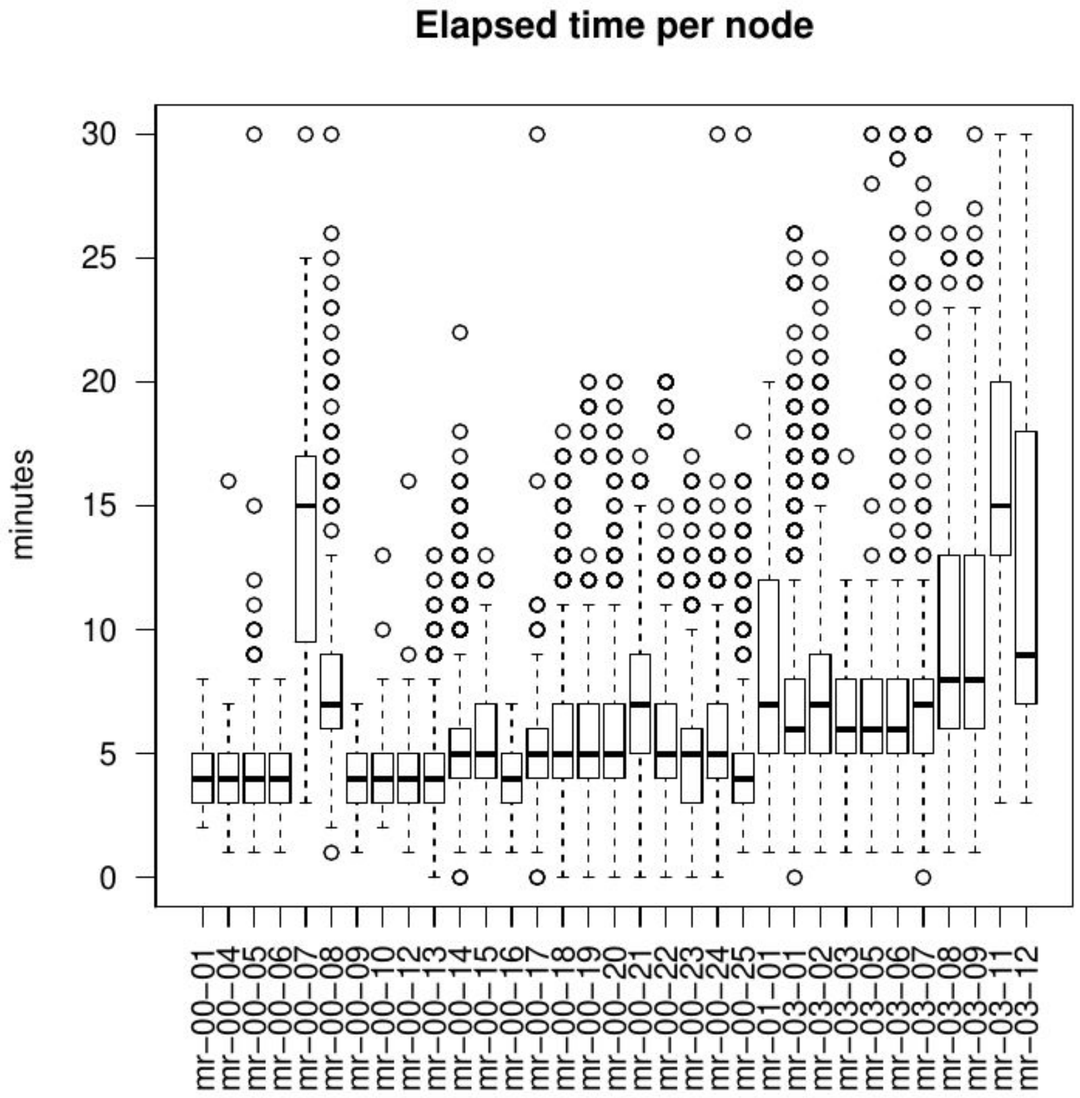
-t, **--time**=<*time*>

Set a limit on the total run time of the job allocation. If the requested time limit exceeds the partition's time limit, the job will be left in a PENDING state (possibly indefinitely). The default time limit is the partition's default time limit. When the time limit is reached, each task in each job step is sent SIGTERM followed by SIGKILL. The interval between signals is specified by the Slurm configuration parameter **KillWait**. The **OverTimeLimit** configuration parameter may permit the job to run longer than scheduled. Time resolution is one minute and second values are rounded up to the next minute.

A time limit of zero requests that no time limit be imposed. Acceptable time formats include "minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-hours:minutes" and "days-hours:minutes:seconds".

Homemade benchmark





Keep working on the cluster while simulating and fairplay

sbatch -x mr-01-[01-02] --time=30 --array=0-9 -J sust.${m}.${model} \

~/SCRIPTS/wflow_sust_cluster.sh $model $mult

-x, **--exclude**=<*node name list*>
Explicitly exclude certain nodes from the resources granted to the job.

-w, **--nodelist**=<*node name list*>
Request a specific list of hosts. The job will contain *all* of these hosts and possibly additional hosts as needed to satisfy resource requirements. The list may be specified as a comma-separated list of hosts, a range of hosts (host[1-5,7,...] for example), or a filename. The host list will be assumed to be a filename if it contains a "/" character. If you specify a minimum node or processor count larger than can be satisfied by the supplied host list, additional resources will be allocated on other nodes as needed. Duplicate node names in the list will be ignored. The order of the node names in the list is not important; the node names will be sorted by Slurm.

Job array: Keep working on the cluster while simulating and fairplay

sbatch -x mr-01-[01-02] --time=30 --array=0-9 -J sust.${m}.${model} \

~/SCRIPTS/wflow_sust_cluster.sh $model $mult

-a, **--array**=<*indexes*>
Submit a job array, multiple jobs to be executed with identical parameters. The *indexes* specification identifies what array index values should be used. Multiple values may be specified using a comma separated list and/or a range of values with a "-" separator. For example, "--array=0-15" or "--array=0,6,16-32". A step function can also be specified with a suffix containing a colon and number. For example, "--array=0-15:4" is equivalent to "--array=0,4,8,12". A maximum number of simultaneously running tasks from the job array may be specified using a "%" separator. For example "--array=0-15%4" will limit the number of simultaneously running tasks from this job array to 4. The minimum index value is 0. the maximum value is one less than the configuration parameter MaxArraySize. NOTE: currently, federated job arrays only run on the local cluster.

Job array: Keep working on the cluster while simulating and fairplay

sbatch -x mr-01-[01-02] --time=30 --array=0-9 -J sust.${m}.${model} \

~/SCRIPTS/wflow_sust_cluster.sh $model $mult

-a, **--array**=<*indexes*>

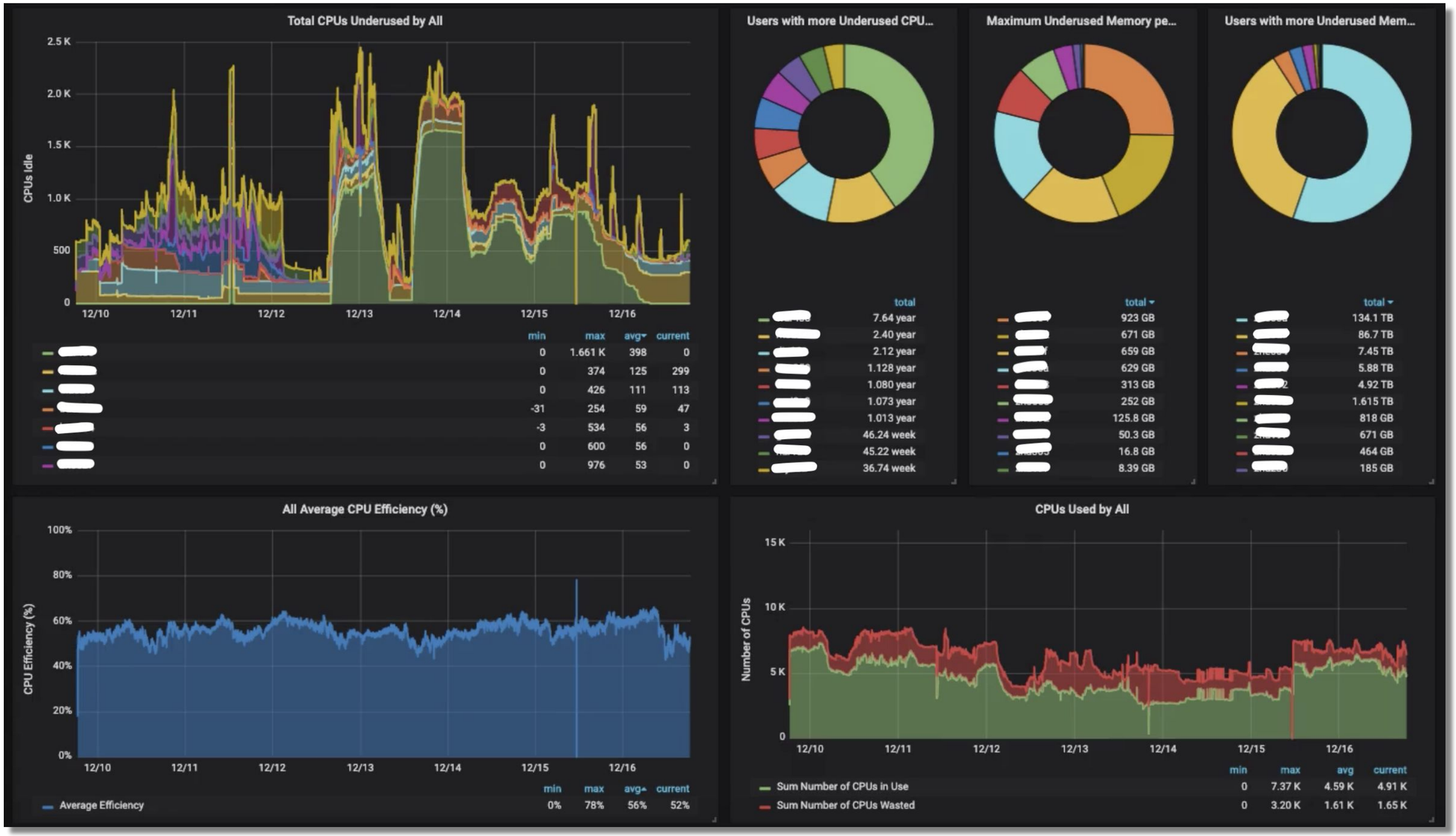
Submit a job array, multiple jobs to be executed with identical parameters. The *indexes* specification identifies what array index values should be used. Multiple values may be specified using a comma separated list and/or a range of values with a "-" separator. For example, "--array=0-15" or "--array=0,6,16-32". A step function can also be specified with a suffix containing a colon and number. For example, "--array=0-15:4" is equivalent to "--array=0,4,8,12". A maximum number of simultaneously running tasks from the job array may be specified using a "%" separator. For example "--array=0-15%4" will limit the number of simultaneously running tasks from this job array to 4. The minimum index value is 0. the maximum value is one less than the configuration parameter MaxArraySize. NOTE: currently, federated job arrays only run on the local cluster.

Keep working on the cluster while simulating

sbatch -x mr-01-[01-02] --time=30 --array=0-9 -J sust.${m}.${model} \

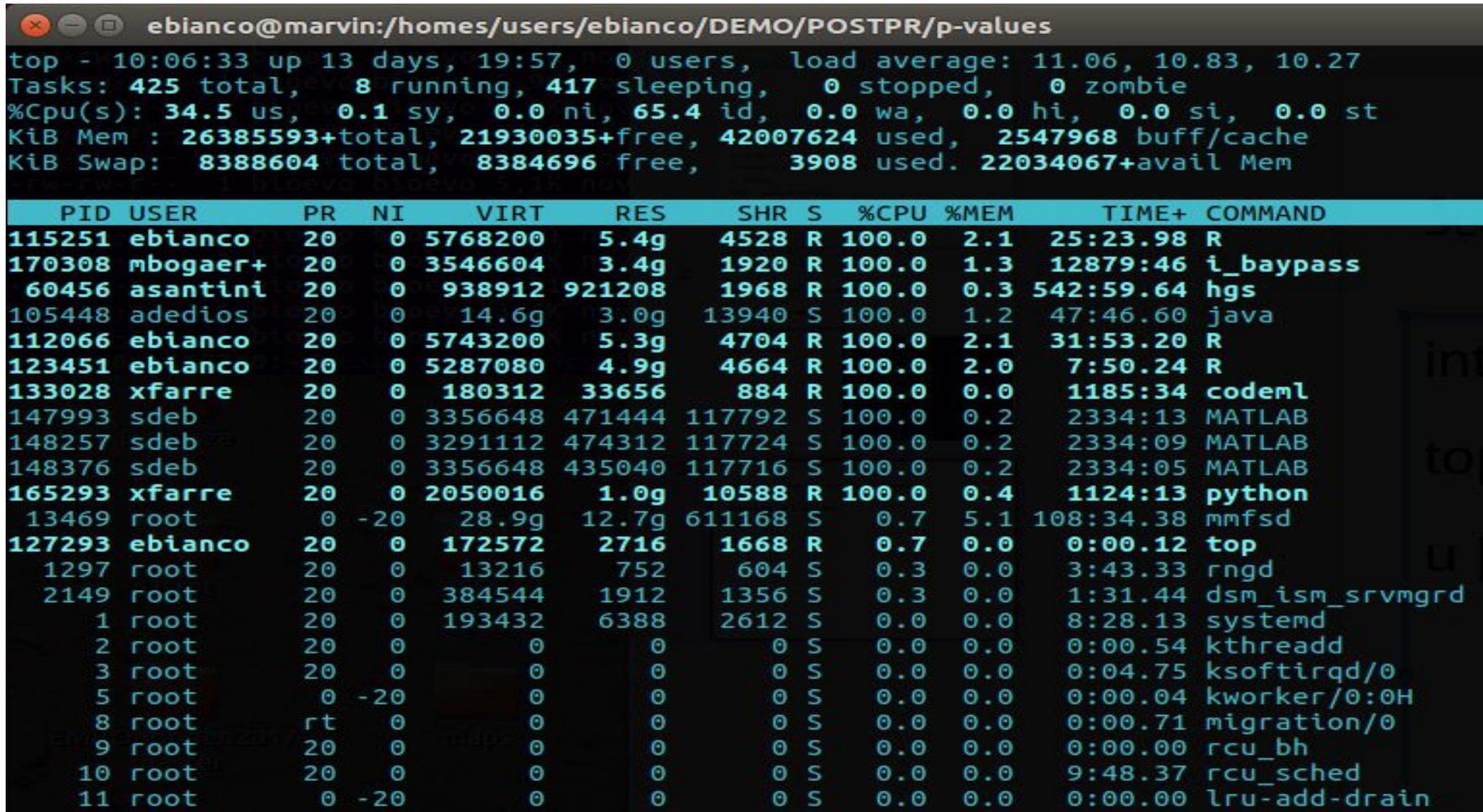
~/SCRIPTS/wflow_sust_cluster.sh $model $mult

Control in real time how much memory your job is using - job monitoring tool



Control in real time how much memory your job is using - without job monitoring tool

- scontrol show jobid $JOBID
- squeue -t R -u $USER → to see in which node your job is running
- interactive -m 1 -w <node>
- top



In conclusion:

- All disciplines in Life Science can benefit from HPC
- Work smarter: user your resources at their best performance
  - reduce analysis time,
  - reduce costs,
  - reduce wetlab tests
- Be fair, and FAIR (Findability, Accessibility, Interoperability, and Reuse)
  - document and comment all your codes! (It's for your own benefit)

Questions?

erica.bianco@hpcnow.com
www.hpcnow.com