UK Atomic Energy Authority

Running HTC & HPC applications opportunistically across private, academic and public clouds

Andrew Lahiff, Shaun de Witt, Miguel Caballer, Giuseppe La Roca, Stanislas Pamela, David Coster

EOSCpilot
The European Open Science Cloud for Research Pilot Project

CCFE
CULHAM CENTRE FOR FUSION ENERGY

Overview

Motivation & aims

Software portability & reproducibility

PROMINENCE

- Architecture
- Resource placement
- Data handling & storage

Fusion use cases

Summary & future work

Computing in fusion

UK Atomic Energy Authority

Fusion

- Recreating the power of the sun in a small volume
- Eventual aim is to supply power to the electrical grids

EUROfusion research community today

- Distributed computing does not exist
- Isolated “islands” of data & compute resources (HPC clusters)
- Difficult to access data & resources at other sites
- Software is not portable

Challenges of ITER

- Largest fusion experiment ever built
- ~2 PB data will be generated per day
- Data may need to be distributed globally
- Need for access to more computing resources

Aims

We aim to address *part* of the fusion computing problem:

- How to get the same result wherever you run
- How to make use of academic and public clouds for fusion HPC applications
- How to extend small scale on-premises facilities to cope with bursty workloads

Work initially carried out within the Fusion Science Demonstrator in EOSCpilot

EOSCpilot
The European Open Science Cloud for Research Pilot Project

The European Open Science Cloud for Research pilot project is funded by the European Commission, DG Research & Innovation under contract no. 739563

Software in the fusion community

Applications make use of:

- FORTRAN, C, C++, Python, Perl, Bash, IDL, Matlab, …
- Commercial software, e.g. NAG libraries, IDL, Matlab, …
- Different compilers, including GNU, Intel, PGI

Designed to be built & run on only a few specific clusters

- Extensive use of env modules & pre-installed software
- Makefiles for different HPC clusters

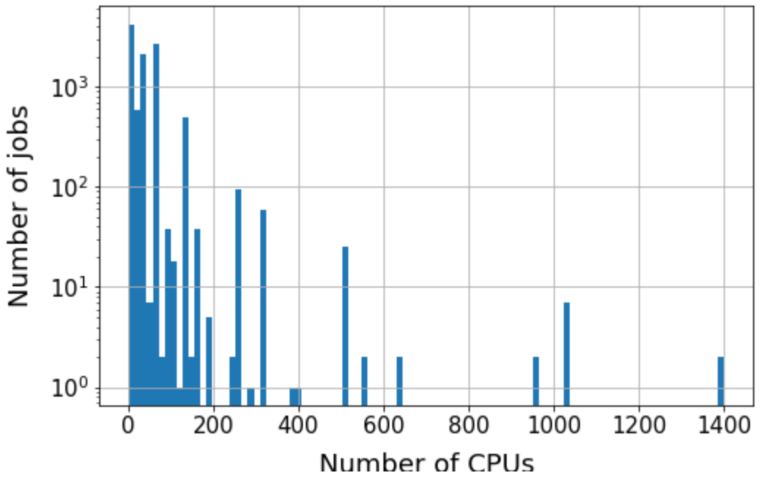
Wide variety of applications

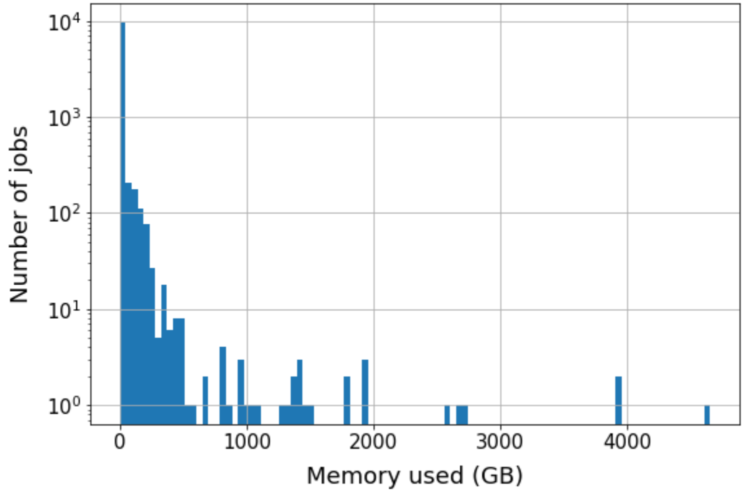
- Plasma modelling, data processing, engineering, materials research, rendering, uncertainty quantification, ML, DL, …

Wide range of resource requirements

- From 1 core to thousands of cores

Many applications are not open source





Recent jobs run on a HPC system at CCFE

Portability & reproducibility

UK Atomic Energy Authority

How to make software available on many different clouds?

- Create container images with each application and all dependencies

We're using two unprivileged container runtimes

- Singularity
- udocker

Images always created using Docker

Storing images

- Docker Hub for images which can be public
- S3 (Ceph) for private images
- In future maybe Harbor as a private registry

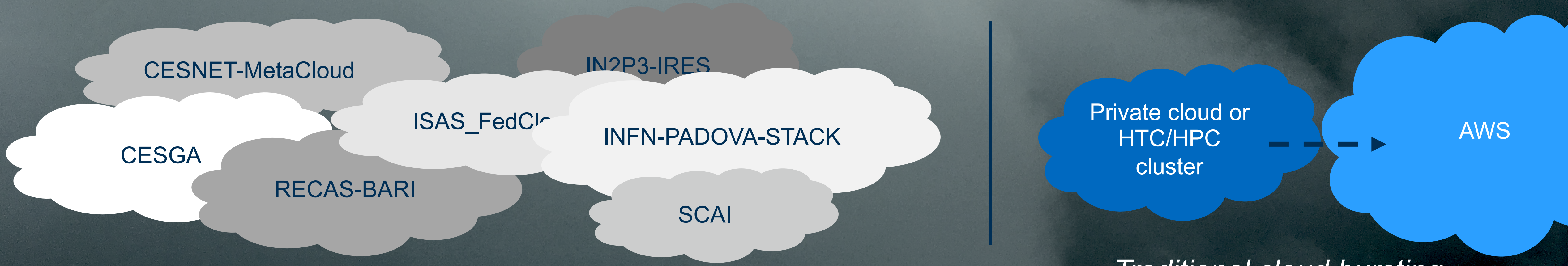
Time to pull & run a 1 GB container

	Singularity	udocker
Docker Hub	110s	52s
Tarball on S3	-	90s
Singularity image on S3	2s	-

Scattered clouds

In EOSCpilot we had access to the numerous clouds in EGI FedCloud

- Opportunistic access only
    - Different numbers of cores here and there
- We wanted to aggregate all these resources and make them useful



Traditional cloud bursting

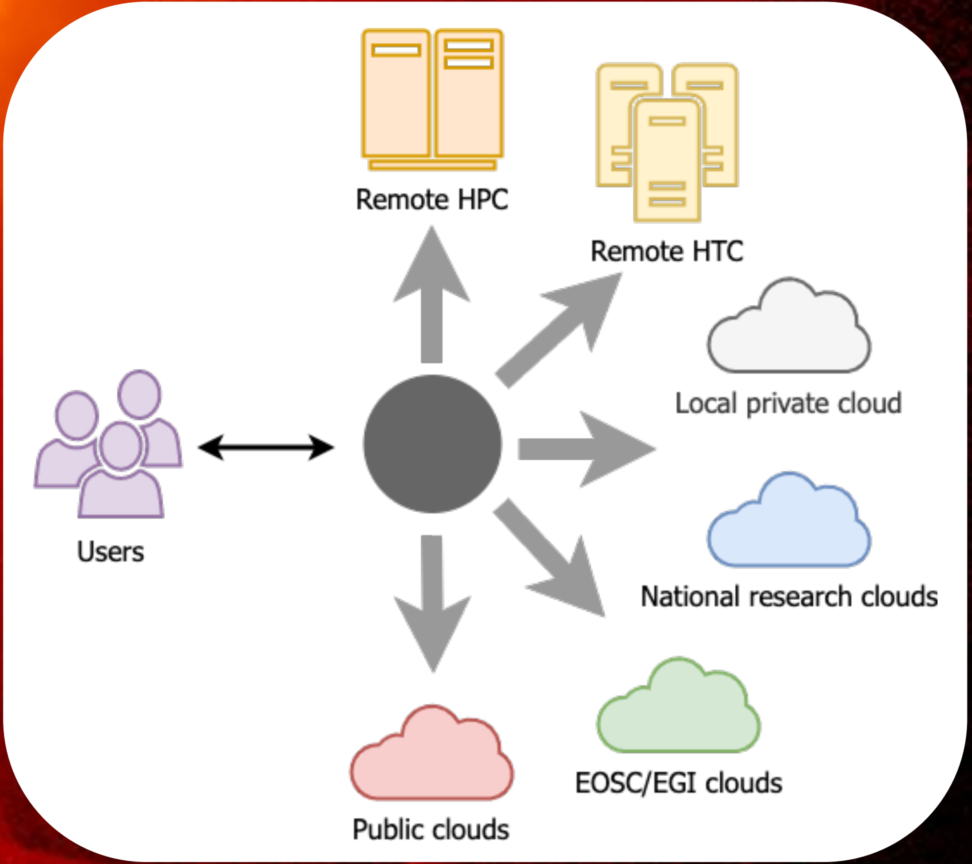
This is not a situation that is normally considered

- More common to assume one or a couple of static clouds with ~“infinite” resources

Introducing PROMINENCE

Platform developed in the EOSCpilot Fusion Science Demonstrator

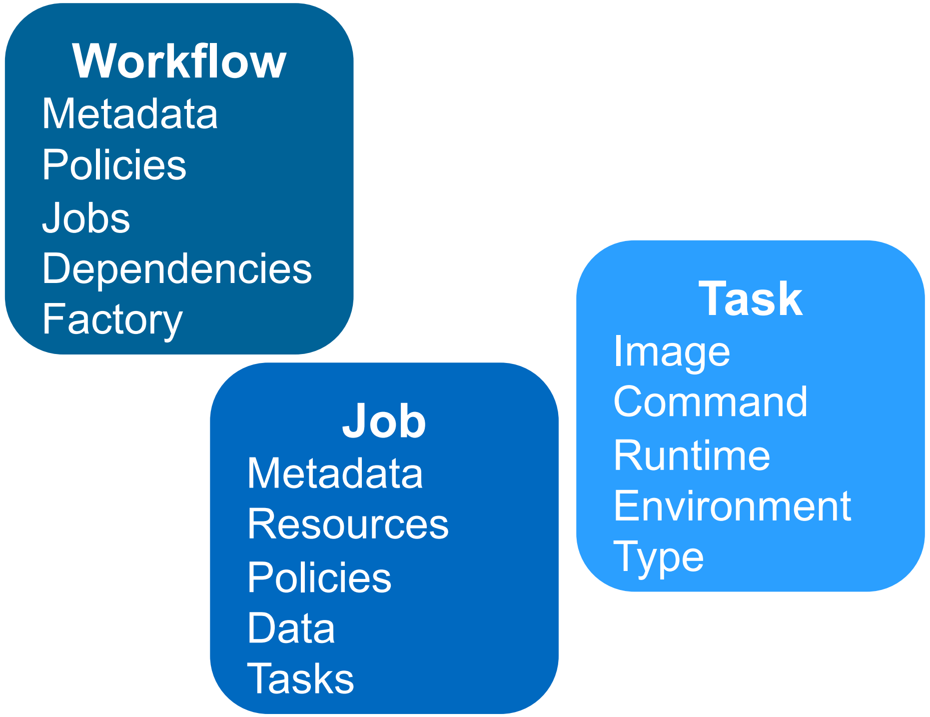
- Users have the same experience as using a traditional batch system
- Jobs run on any number of clouds in addition to HTC and HPC batch systems
- Users don't need to worry about where their jobs are running
- Users don't need to worry about provisioning infrastructure on clouds
    - Or even know what a cloud is



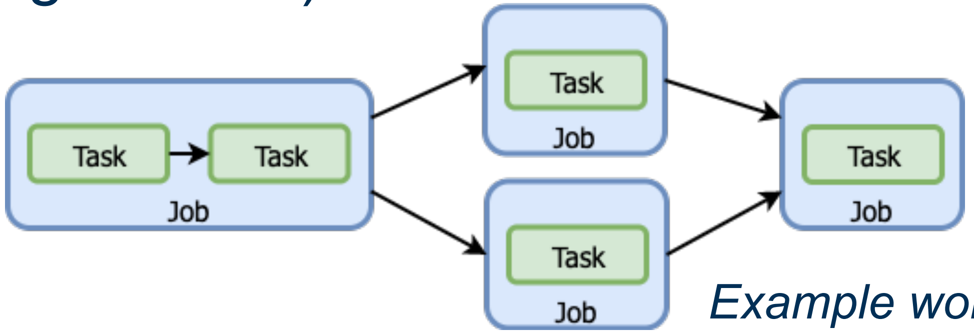
Interacting with PROMINENCE

Based on a RESTful API and JSON

- Much more flexible than ssh into a login node
- Token-based authn/authz (OpenID Connect)
- Simple Python-based command line interface
  - Provides a batch system like experience
- Easy to run jobs programmatically
- Easy to run jobs from any Jupyter notebook (e.g. Google Colab)



JSON workflow & job descriptions



Example workflow, jobs & tasks

Architecture overview

Identity provided by an OpenID Connect server

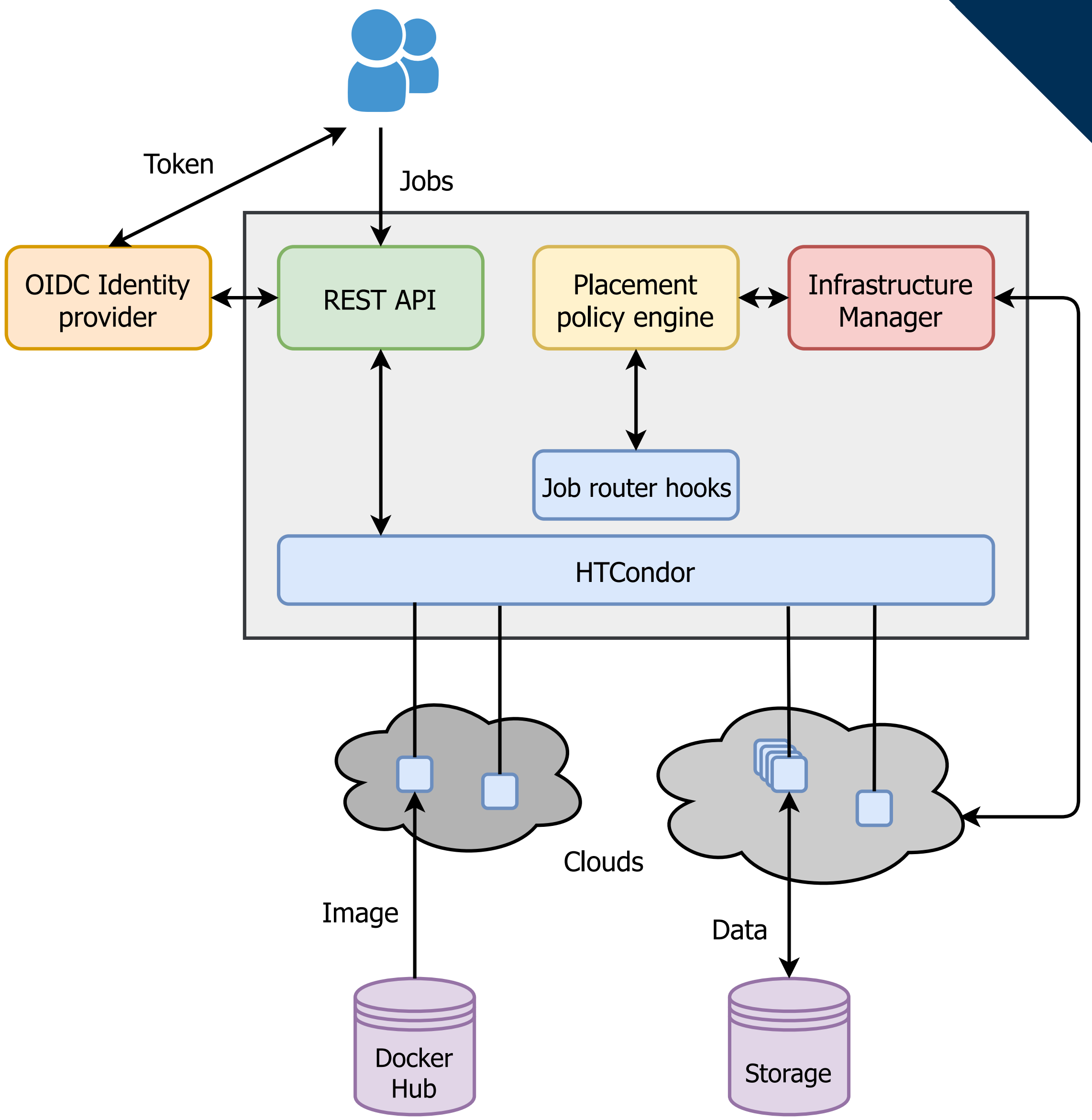
- INDIGO IAM, EGI Check-in

Infrastructure Manager

- Deploys virtual infrastructure on a wide variety of clouds
  - including OpenStack, GCP, Azure, AWS

HTCondor provides

- A transactional database for jobs
- Secure & reliable execution of remote jobs
- Triggers deployment & deletion of infrastructure
- Streaming stdout/err back to users



Infrastructure deployment

Open Policy Agent used to

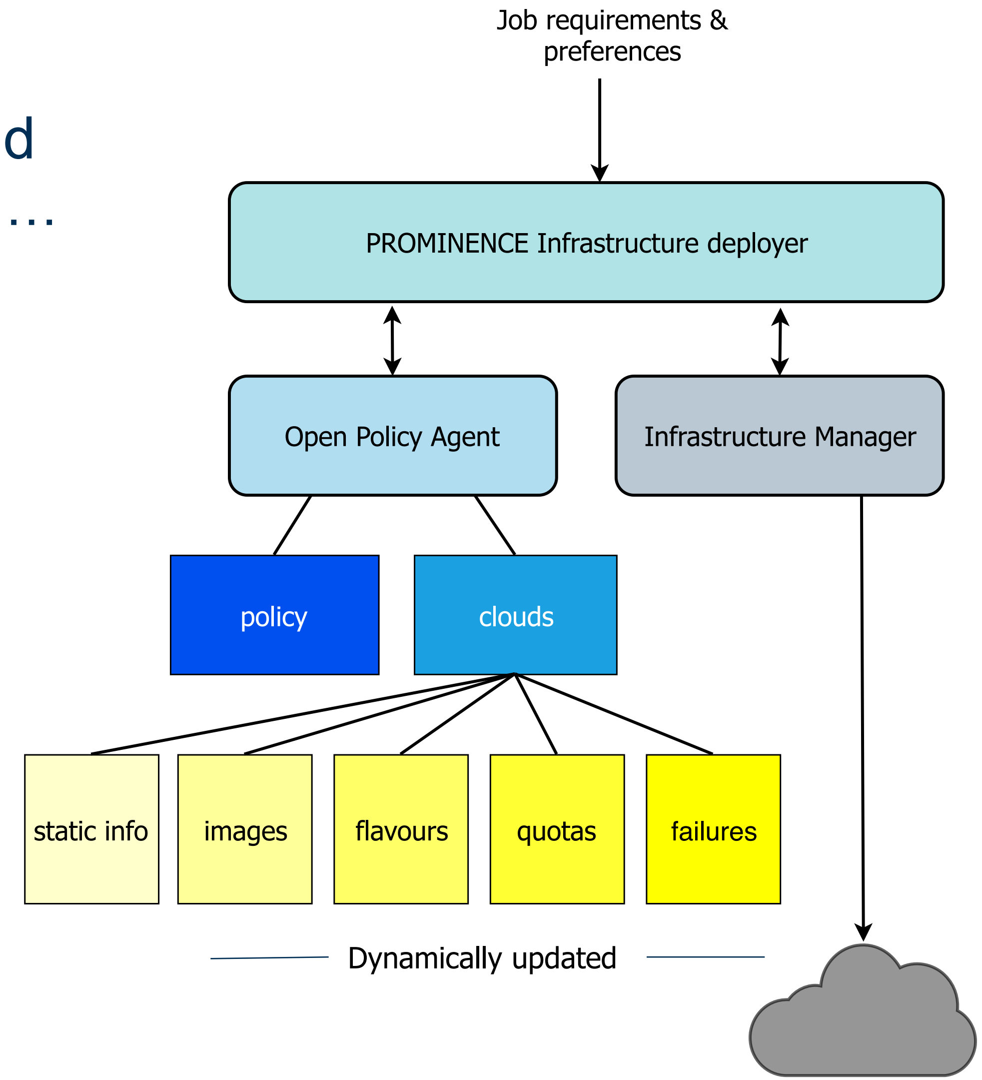
- Store static & dynamic information about each cloud
  - e.g. current quota, recent failures, images, flavours, …
- Determine what clouds meet job requirements
- Rank clouds based on job preferences

Failure handling is essential

- Time-outs
- Retries on the same cloud
- Retries on other clouds meeting requirements
- Back-off from clouds which fail

Also handles cloud credentials

- E.g. refreshing access tokens if necessary



Resource placement

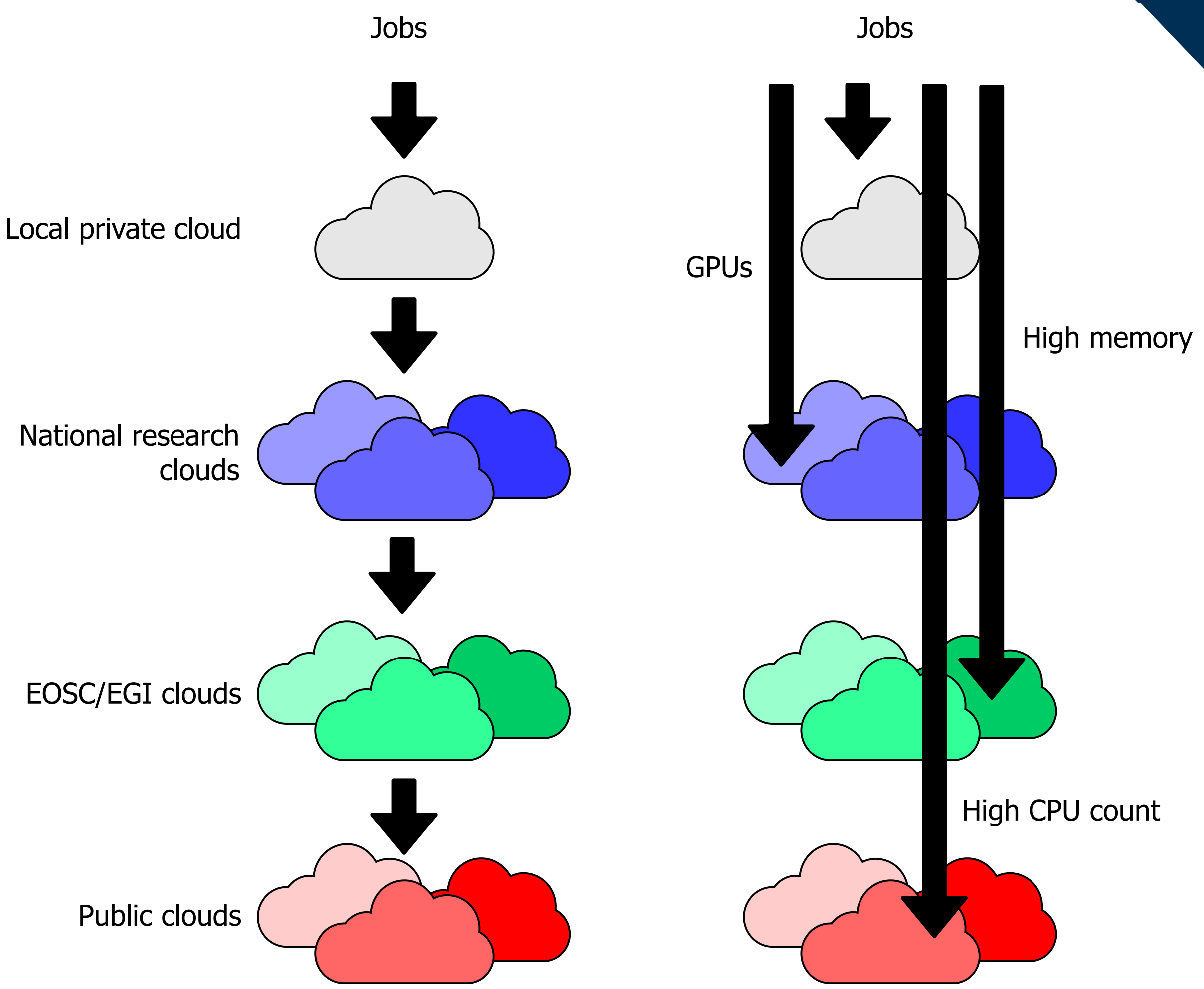
Can preferentially use a local cloud, but

- Burst onto national research clouds when needed
- Burst onto EGI/EOSC clouds when needed
- Burst onto public clouds when needed

Use an external cloud immediately if the local cloud doesn't meet requirements

- E.g. access to GPUs, low-latency interconnects, …

Rank clouds based on network bandwidth, cost, …



Data handling & storage

Data can be staged-in & staged-out from Ceph via the S3 API

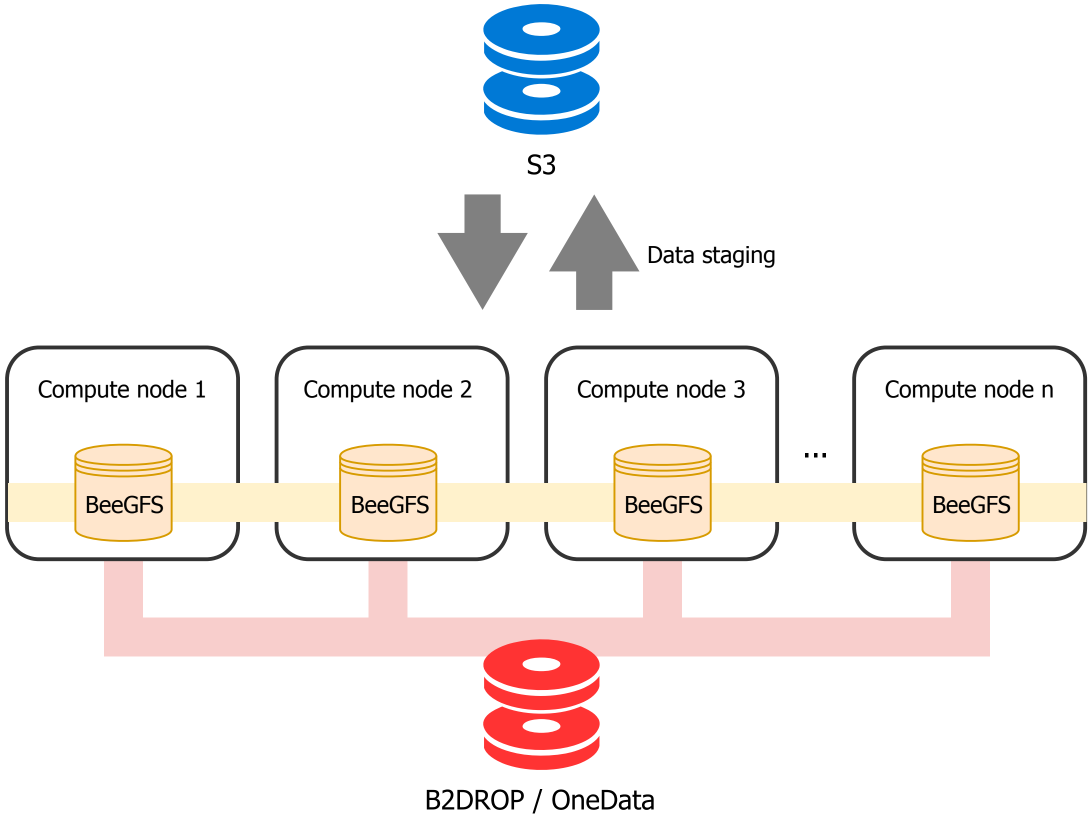
- PROMINENCE generates pre-signed URLs & provides tools to enable users to upload & download data

Multi-node jobs have converged shared storage

- Uses BeeGFS, unique to each job
- Aggregates the performance & capacity of the nodes

Also options for POSIX-like access to storage

- EUDAT B2DROP (based on Nextcloud)
- OneData



PROMINENCE user experience

Example MPI job submission:

```
prominence create --cpus 32 --memory 64 --nodes 8 \
                  --intelmpi --input in.lj \
                  alahiff/lammps-intel-avx512-2018 \
                  “lmp_intel_cpu_intelmpi -in in.lj”
```

List current jobs:

```
prominence list
```

Look at the standard output in real time:

```
prominence stdout <job id>
```

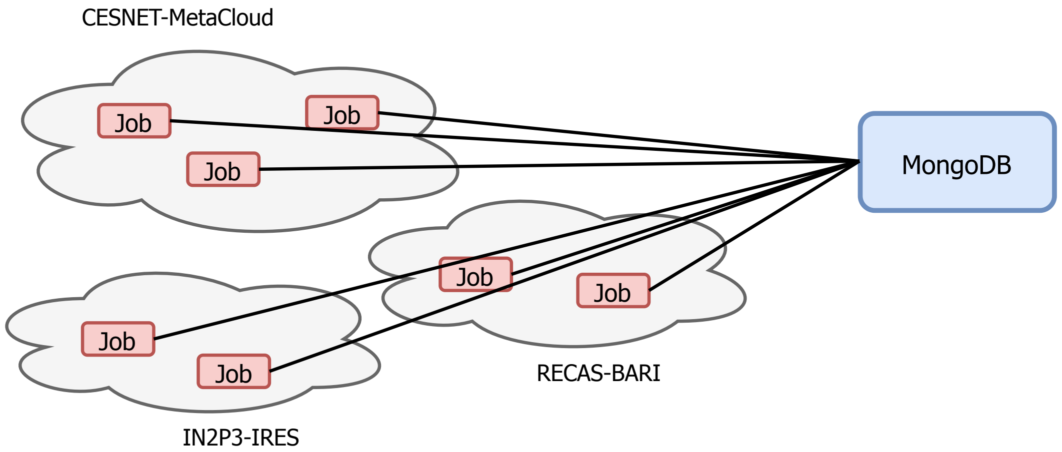
Many alternative platforms require lots of human involvement:

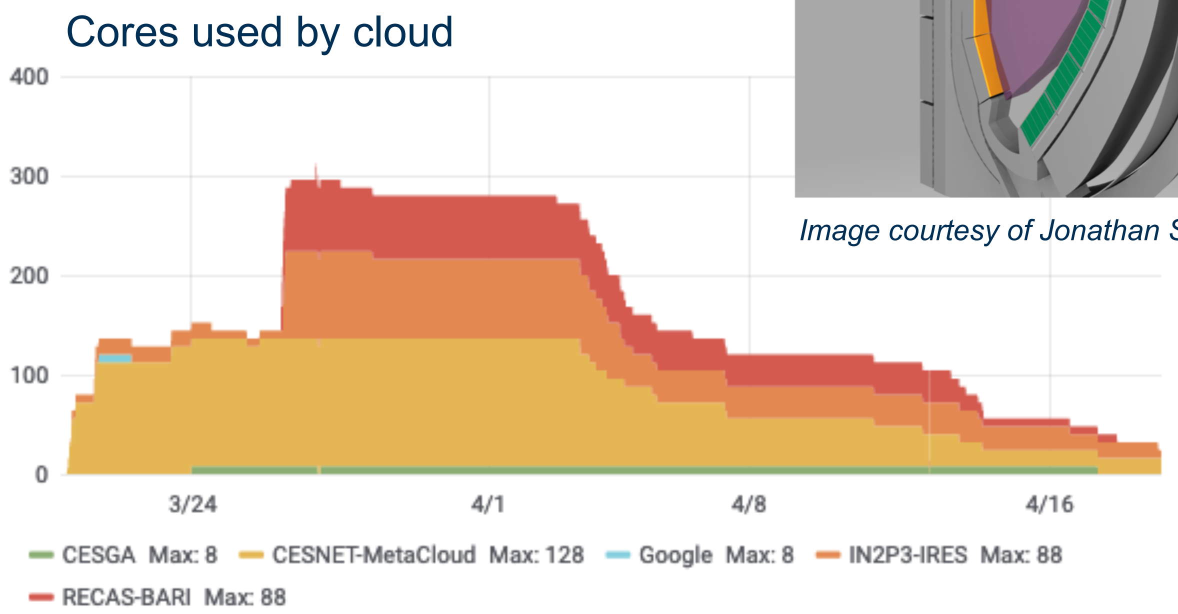
- Decide what VM flavour(s) to use
- Manually run a command to deploy a cluster
- scp data/scripts/… to a transient login node
- ssh into the login node
  - Submit jobs
- Manually run a command to destroy the cluster

Use case: breeder blanket design

Optimising tritium production using 3D CAD-based neutronics models

- Ran across 4 EGI FedCloud sites
- Self-organising HTC jobs using MongoDB for coordination
- Output data written back to MongoDB





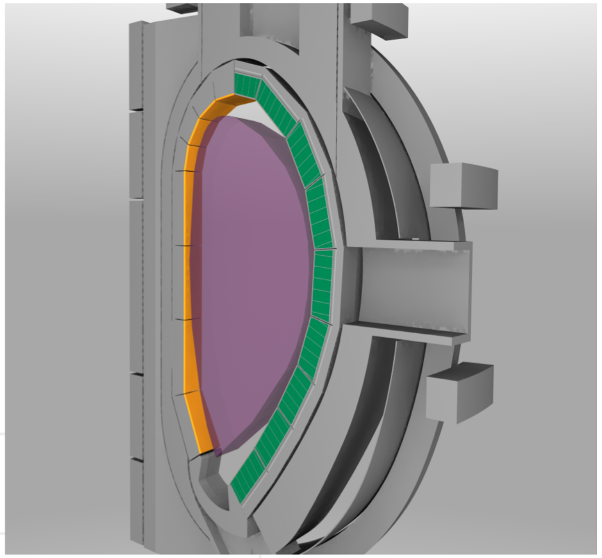


Image courtesy of Jonathan Shimwell

UK Atomic Energy Authority

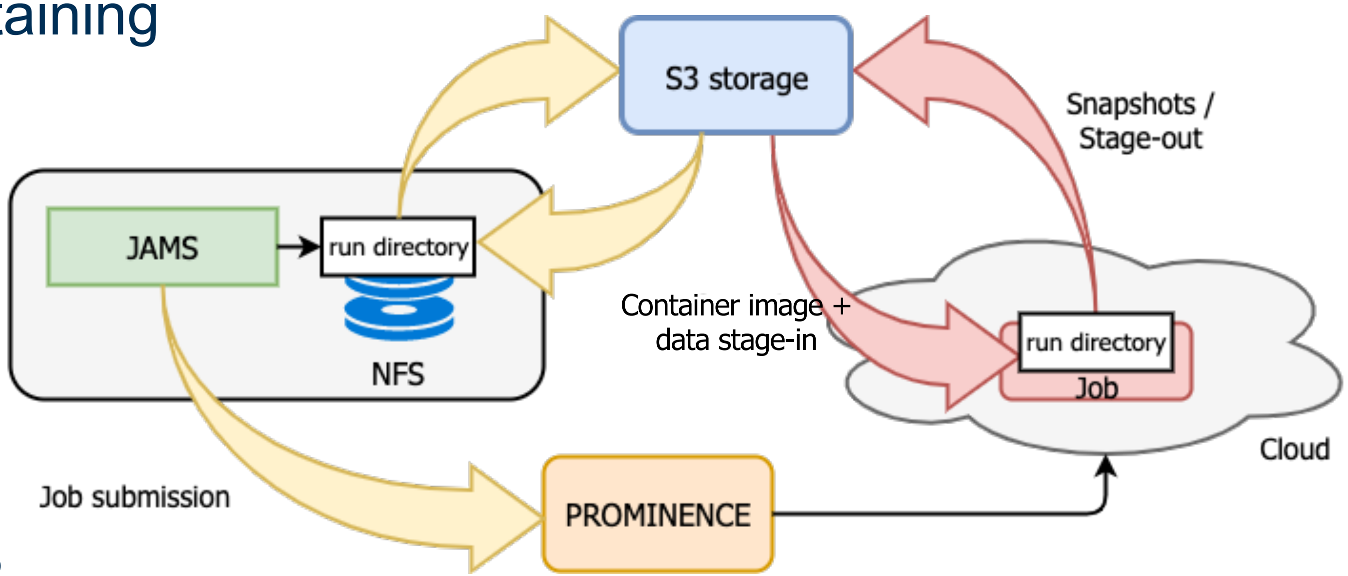
Use case: JAMS

JET **A**pplication **M**anagement **S**ystem

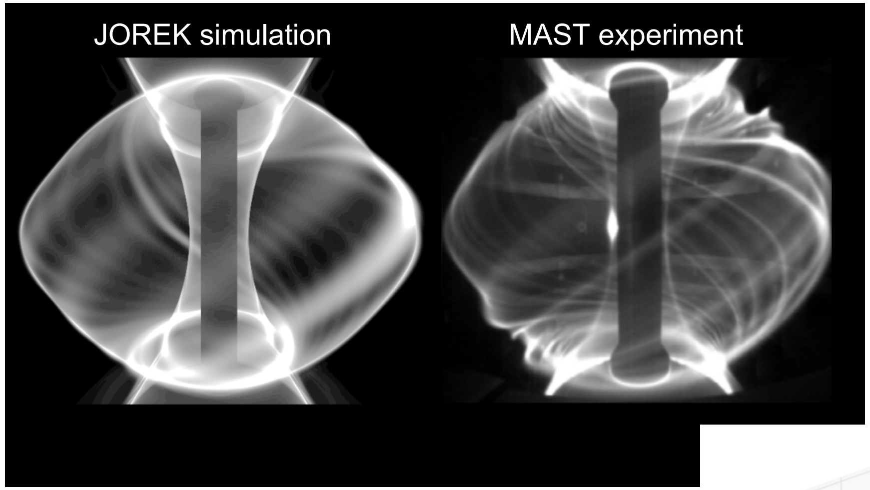
- User interface for a variety of transport & MHD codes

Migrated from a local HPC cluster to PROMINENCE using the REST API

- Application generates run directory containing config & data files
- Tarball of directory uploaded to S3
- Job submitted to PROMINENCE
- Job runs on a cloud somewhere
  - Open MPI (single node)
  - Uses the (commercial) NAG libraries
- Tarball of directory downloaded from S3
  - Users can trigger & download "snapshots", allowing interactive analysis of output files while jobs are still running



Use case: JOREK

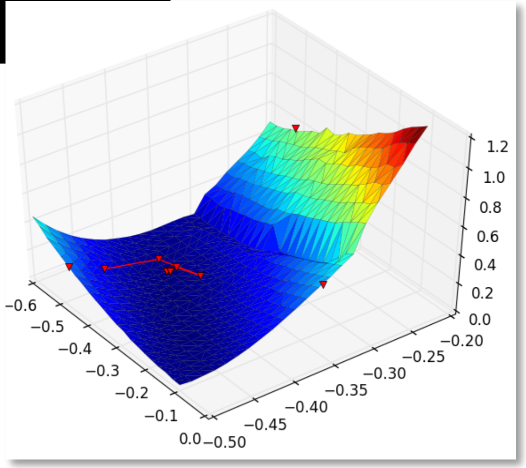


Simulations of MHD instabilities at the edge of tokamak plasmas

- Filamentary structures ejected from the edge
- Dynamics highly dependent on edge plasma pressure

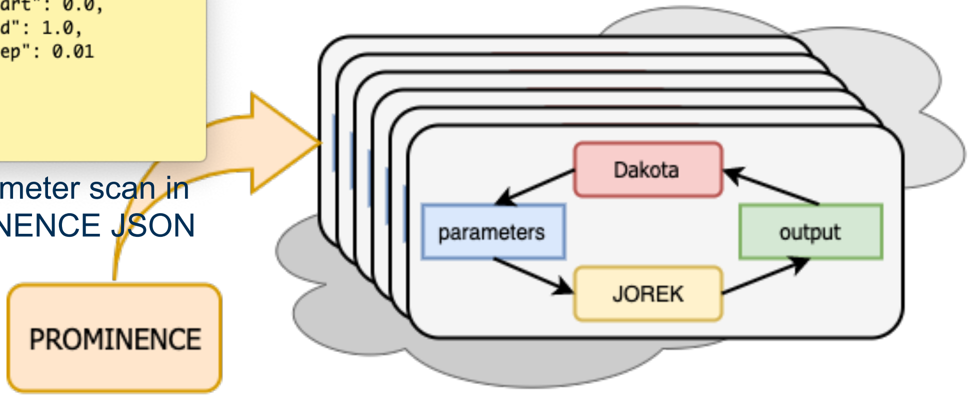
Scanning edge pressure in JOREK simulations

- Changing pressure changes magnetic equilibrium
- Each case requires a new (unknown) plasma current
- Optimise input to satisfy (known) experimental current




```
"factory": {
  "type": "parametricSweep",
  "parameterSets": [
    {
      "name": "x1",
      "start": 0.0,
      "end": 1.0,
      "step": 0.01
    },
    {
      "name": "x2",
      "start": 0.0,
      "end": 1.0,
      "step": 0.01
    }
  ]
}
```

2D parameter scan in PROMINENCE JSON



Use case: HPC applications

UK Atomic Energy Authority

Many multi-node MPI applications used in fusion, including:

- LAMMPS: molecular dynamics simulator
- VASP: atomic scale materials modelling
- BOUT++: plasma and fluid simulations in curvilinear geometry
- ASCOT: accelerated simulation of charged particle orbits in a tokamak

Questions:

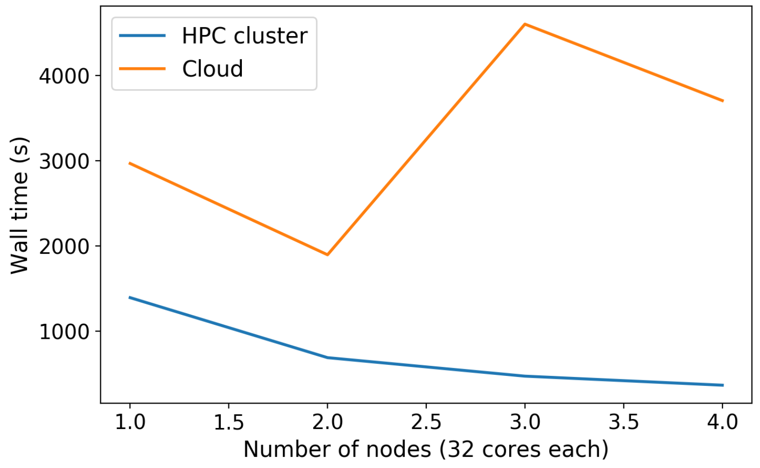
- Is it possible to usefully run any of these on clouds without low-latency interconnects?
- Effect of containers on performance for multi-node MPI jobs?
- Optimisation for each resource vs portability?

Use case: HPC applications

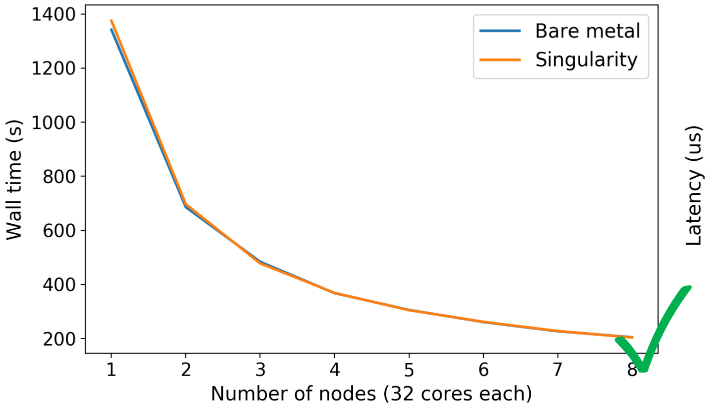
The container image which gives the best performance on some CPUs doesn't work everywhere

LAMMPS wall time	GNU	Intel, AVX512	Intel, AVX2
Intel Skylake	00:41:59	00:11:12	00:14:40
Intel Broadwell	00:47:24	-	00:14:53
AMD EPYC	00:44:28	-	-

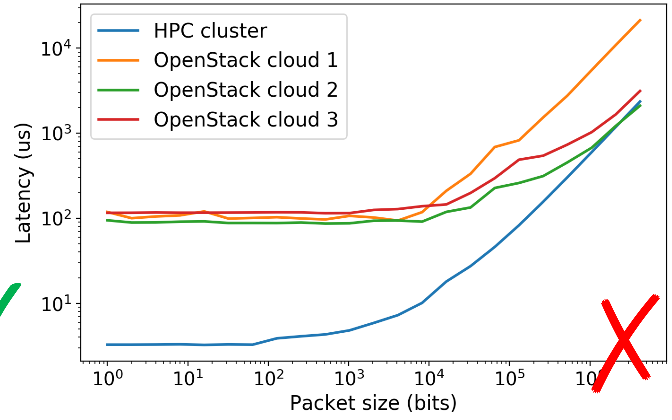
Significant communication between nodes - LAMMPS



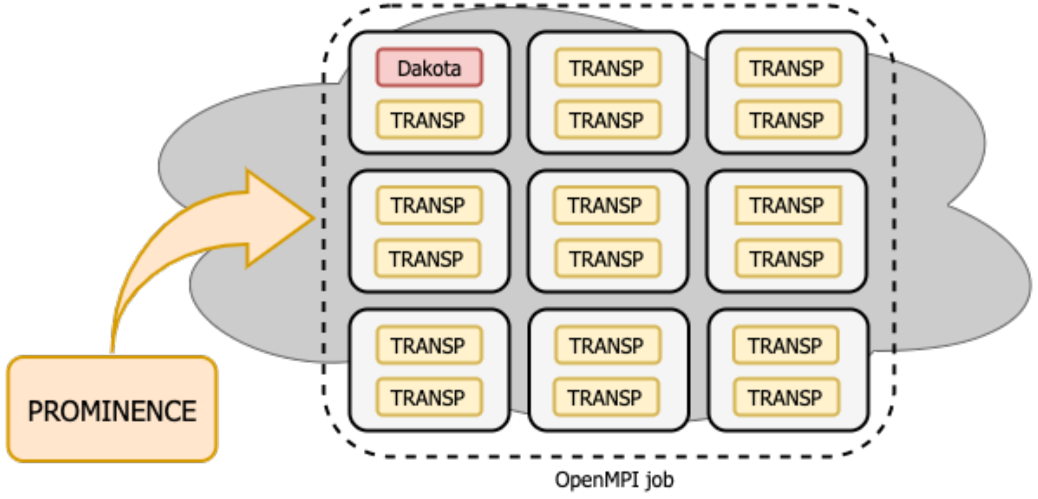
LAMMPS: bare metal vs Singularity



OSU Micro-benchmarks: latency



Very little communication between nodes - Dakota



Multi-node MPI example not requiring low latencies: parallel Dakota running function evaluations in each MPI rank

Summary & future work

Developed a platform enabling users to transparently exploit cloud resources for running both single-node HTC & multi-node MPI jobs

- Can use many clouds opportunistically in a dynamic way
- Not fusion specific

Work in progress & future work

- An EGI/EOSC PROMINENCE instance has been deployed for the long tail of science, uses EGI Check-in for AAI & EGI FedCloud resources
- Testing HPC applications on clouds with low-latency interconnects
- More use cases…

Thank you!