Cloud Bursting: a First Experience

hpc-ch forum on Virtualization

Thursday, 20 May 2021

Ricardo Silva (SCITAS)

Outline

EPFL

SCITAS

Scientific IT and Application Support

- Systems Administrators and Applications experts team
- Providing compute resources, expertise and training to all EPFL communities
  - consulting: design, write, optimize, parallelize your code
  - training: Base (Linux, Clusters), Advanced (MPI, OpenMP, CUDA), Best Practices (Data Management)
  - code hosting: `https://c4science.ch`
- Projects: Square Kilometer Array, EuroFusion

EPFL

SCITAS Clusters

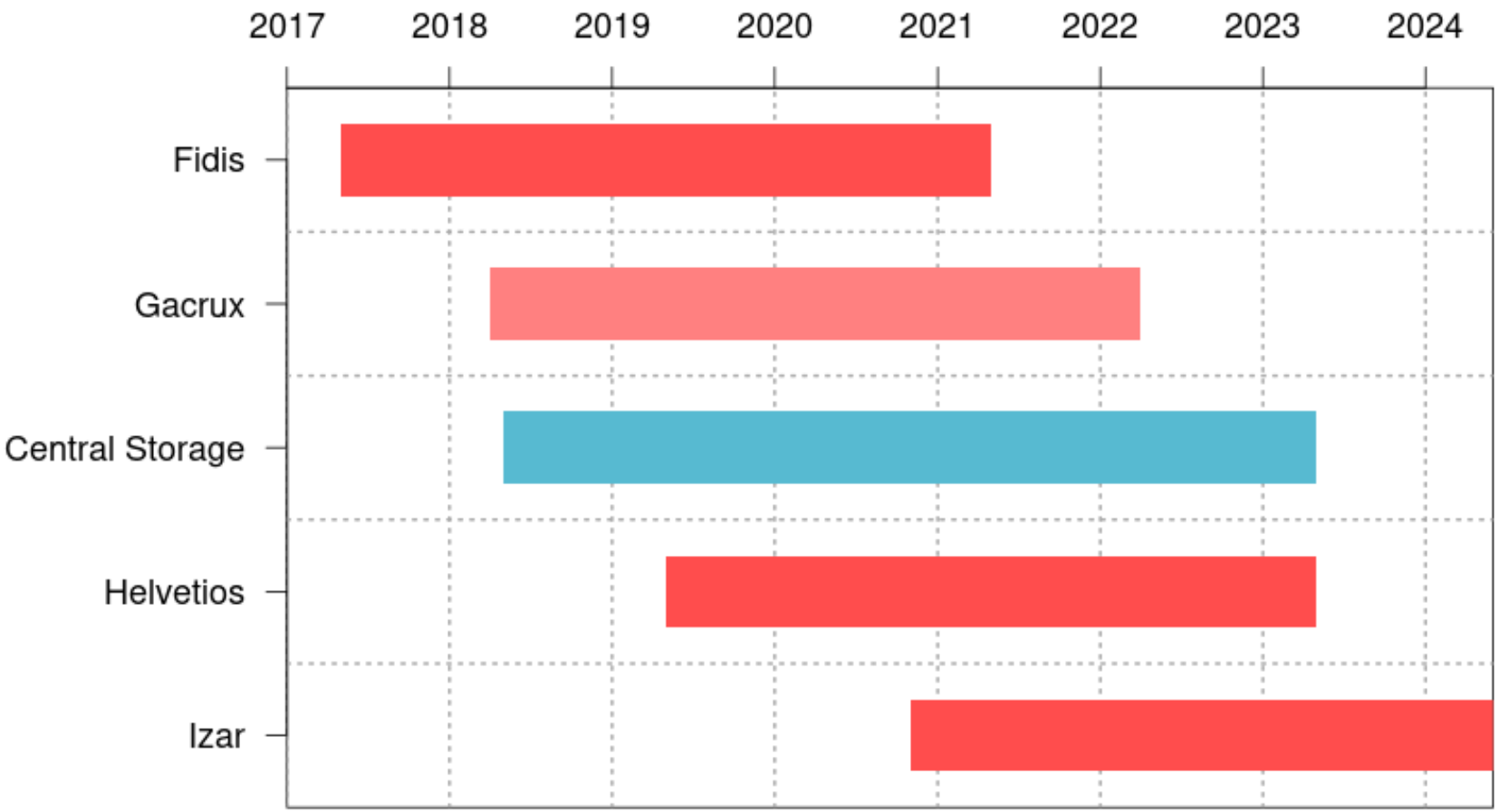
HPC Platform Compute Resources

- 30k cores in >900 nodes, across 3 clusters
  - >2.8 PFLOPs aggregate peak performance
- 6 PB shared storage (GPFS)
- optimized for HPC workloads
  - fast network interconnect (InfiniBand)
  - fast parallel filesystem (GPFS)
- >800 active users from >110 labs all over EPFL

EPFL

SCITAS Clusters

HPC Platform Compute Resources Timeline



Cloud Bursting

Proof of Concept Objectives

- Analyze the state of the art and get to know the software solutions offered by Google and its partners
- Evaluate features from the perspective of system and infrastructure administrators
- Test the scalability

EPFL

Authors

Who did what?

- **Antonio J. Russo**: Architecture, Deployment, Configuration
- **Gilles Fourestey**: Benchmarks
- **Nicolas Richart**: Software stack and Benchmarks
- **Ricardo Silva**: Architecture Review

EPFL

Proof of concept

Compute nodes configuration

- Instance type: c2-standard-60 with hyperthreading disabled (30 cores) (in europe-west4-c)
- Image: GCP hpc-centos-7 image
- Additional system software: python-virtualenv, java-1.8.0-openjdk-devel, gcc-gfortran, gcc-c++ and sssd
- Storage: GCP Filestore to stock our software stack (built using Spack)

EPFL

Three Models

- External Elastic Computing
- Hybrid Elastic Computing
- External Compute Resources

EPFL

External Elastic Computing

Specifications

- A computing cluster entirely localized in the cloud
- Users can log in to the front end with their EPFL (LDAP) account
- During the first connection, the home directory is created automatically
- Data is persistent
- The same `modules` available in our local clusters are available in the cloud

EPFL

External Elastic Computing

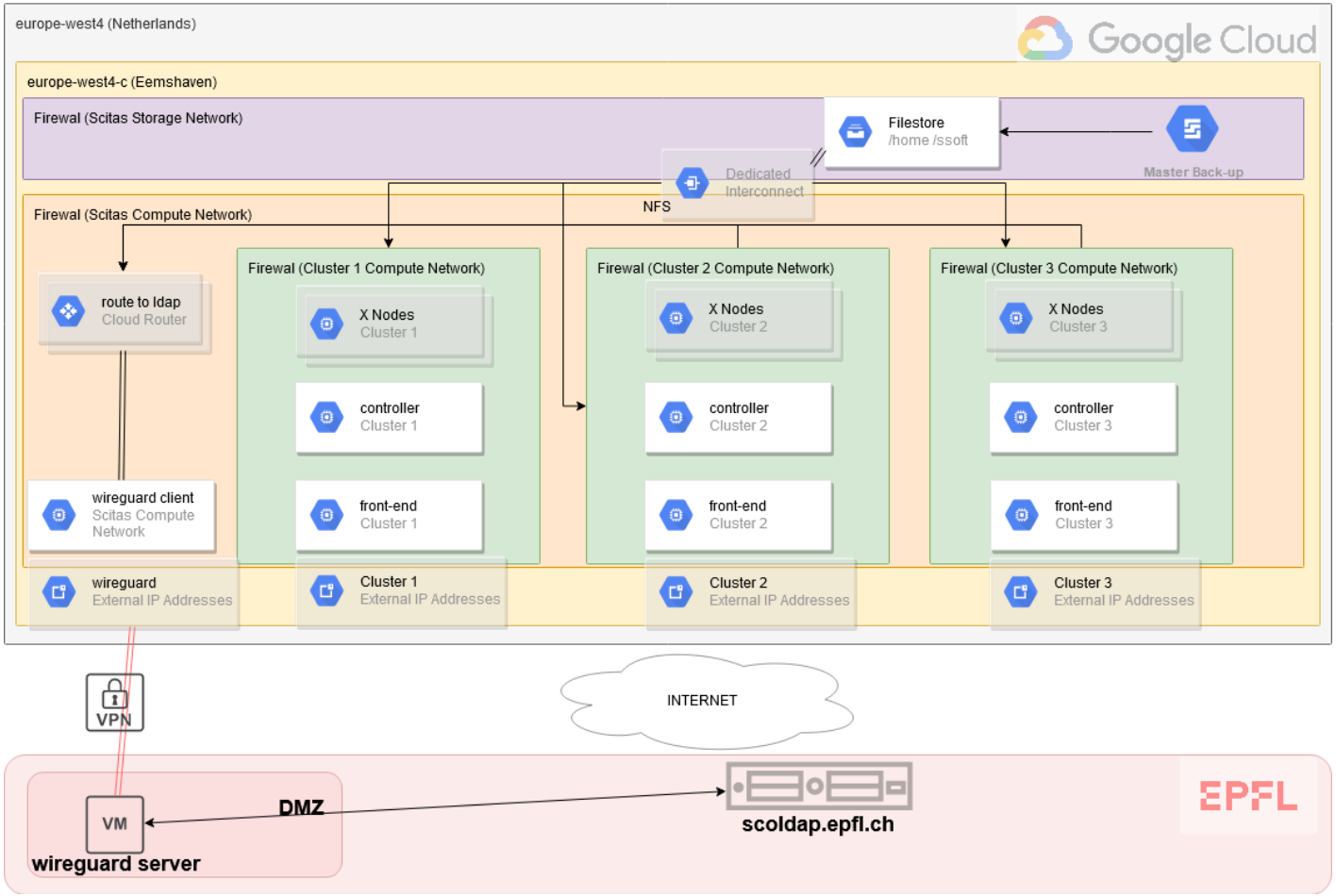
Deployment

- Terraform recipes provided by the slurm-gcp project
  `https://github.com/SchedMD/slurm-gcp`
- Deploying a functional cluster takes approximately 15 minutes (with a pre-built scientific software stack)
  - Dominated by the approach used to deploy slurm
  - A production solution would use a configuration management tool (puppet, ansible, ...) and package repositories (for system and scientific software).

EPFL

External Elastic Computing

Architecture



EPFL

Hybrid Elastic Computing

Specifications

- Compute nodes entirely localized in the cloud
- Services nodes inside the EPFL network
- Users can log in to the front end with their LDAP account
- `/home` directory available only on the front-end node
- `/soft` and `/scratch` deployed on the cloud and available on front-end and compute nodes (mounted via NFS through the VPN)

EPFL

Hybrid Elastic Computing

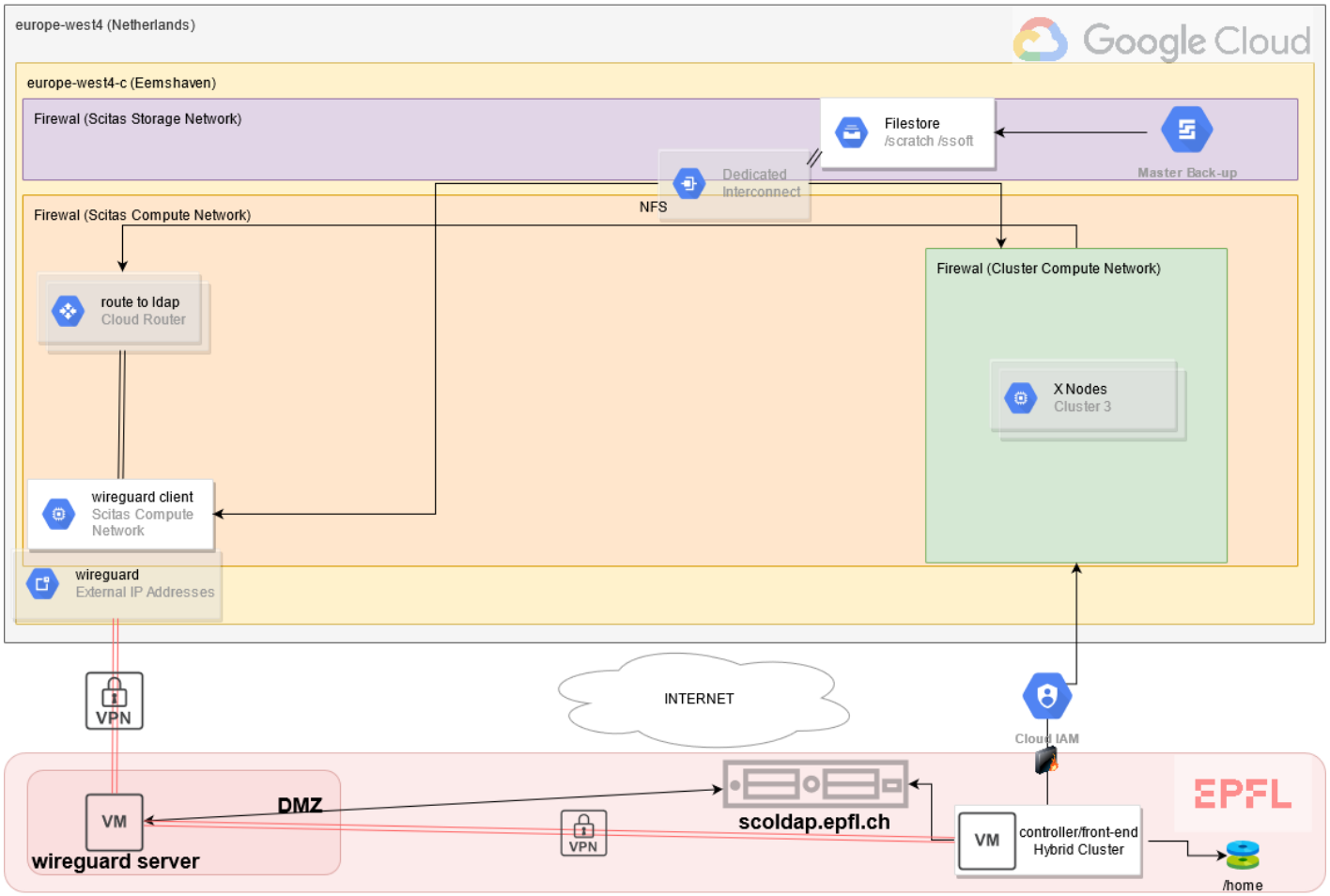
Deployment

- Instances created and destroyed by Slurm (using *suspend* and *resume* programs)
- Deploying a functional compute node on the cloud takes approximately 4 minutes (independent of the number of nodes)
- Front-end and Slurm controller nodes are Virtual Machines inside EPFL's network

EPFL

Hybrid Elastic Computing

Architecture



External Compute Resources

Specifications

- Compute resources entirely localized in the cloud (TPUs)
- No data storage device in the cloud
- Input data is processed in the front-end node and sent to the computing device on the cloud
- Test by a multiplication of two matrix on each core of the TPU node using Tensorflow

EPFL

External Compute Resources

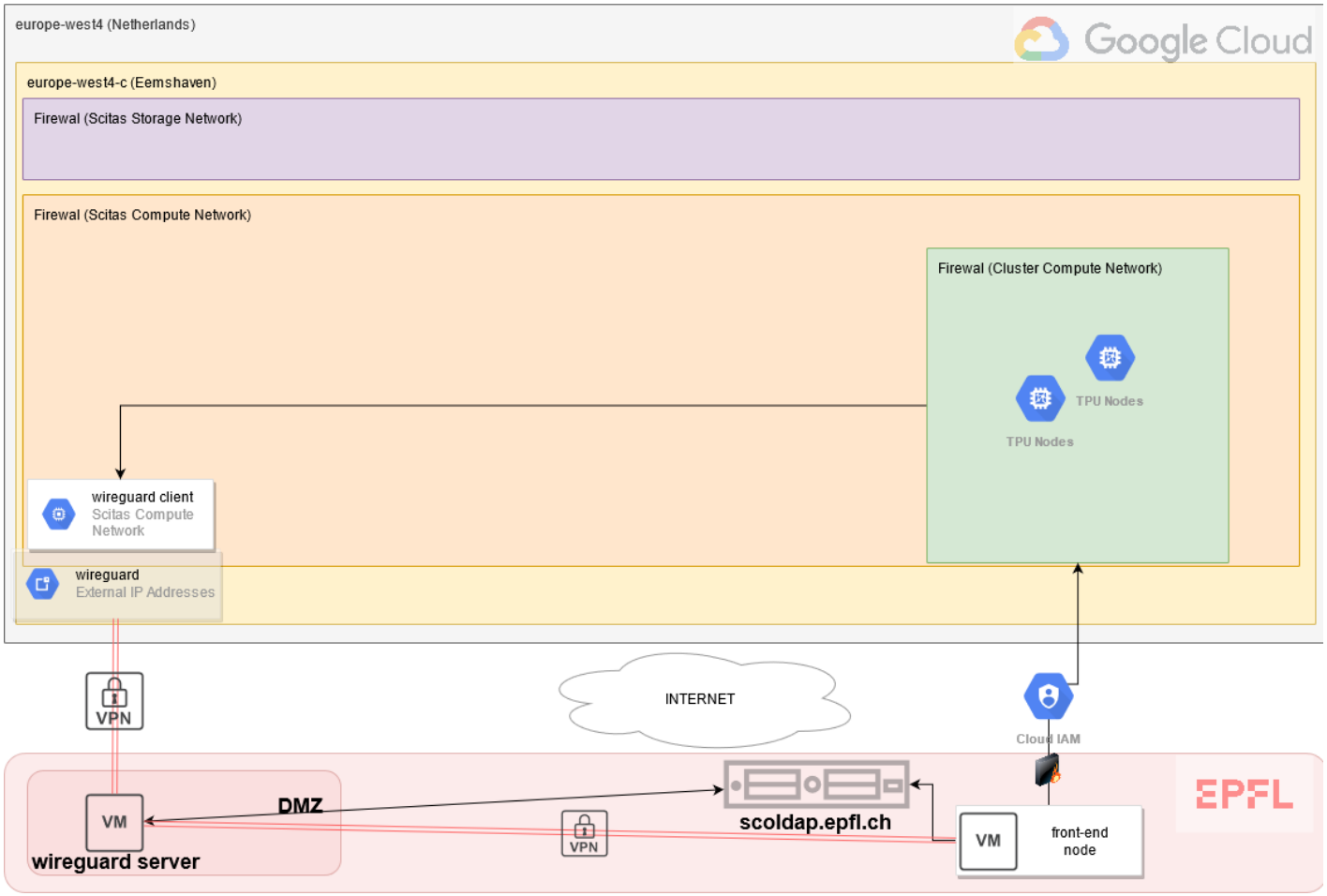
Deployment

- Instances are created by a GCP administrator manually (but could easily be triggered by Slurm prolog)
- Job submitted directly on the front-end without any scheduler
- To run a job, user must know the TPU IP address

EPFL

External Compute Resources

Architecture



Benchmarking

Description

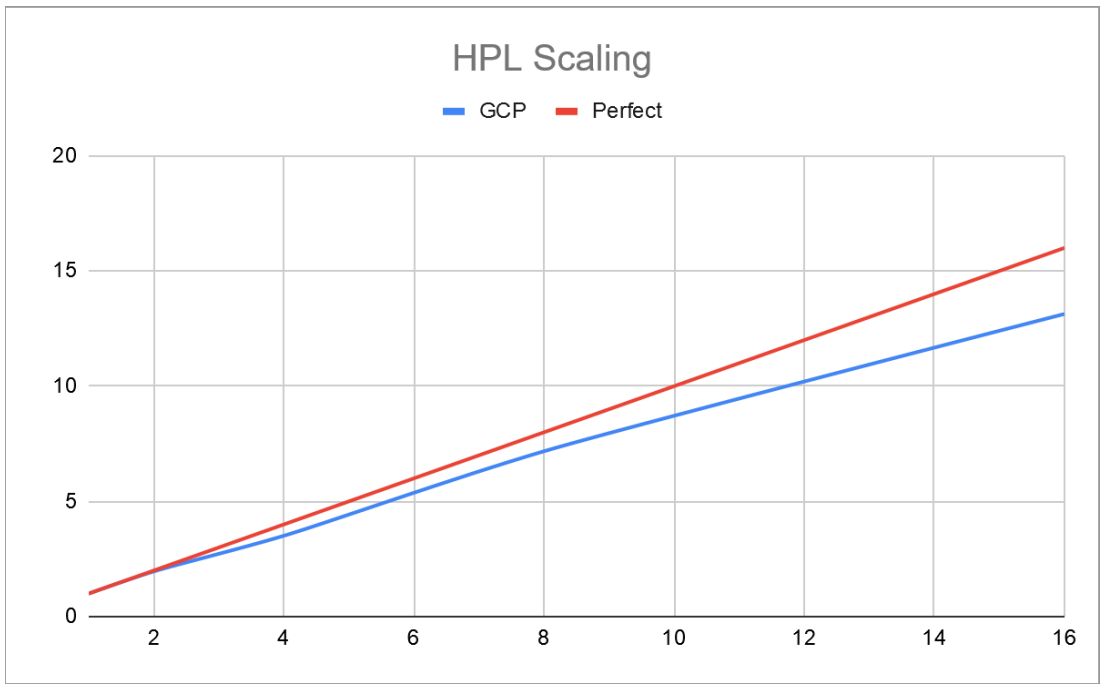
- **HPL**: HPL is a software package that solves a (random) dense linear system in double precision (64 bits)arithmetic on distributed-memory computers ;
- **FFTW**: The Fastest Fourier Transform in the West (FFTW) is a software library for computing discrete Fourier transforms (DFTs). It can compute transforms of real and complex-valued arrays of arbitrary size and dimension ;
- **GBS**: The Global Braginskii Solver (GBS) solves the drift-reduced Braginskii equations to study the plasma dynamics in edge configuration.

EPFL

Benchmarking

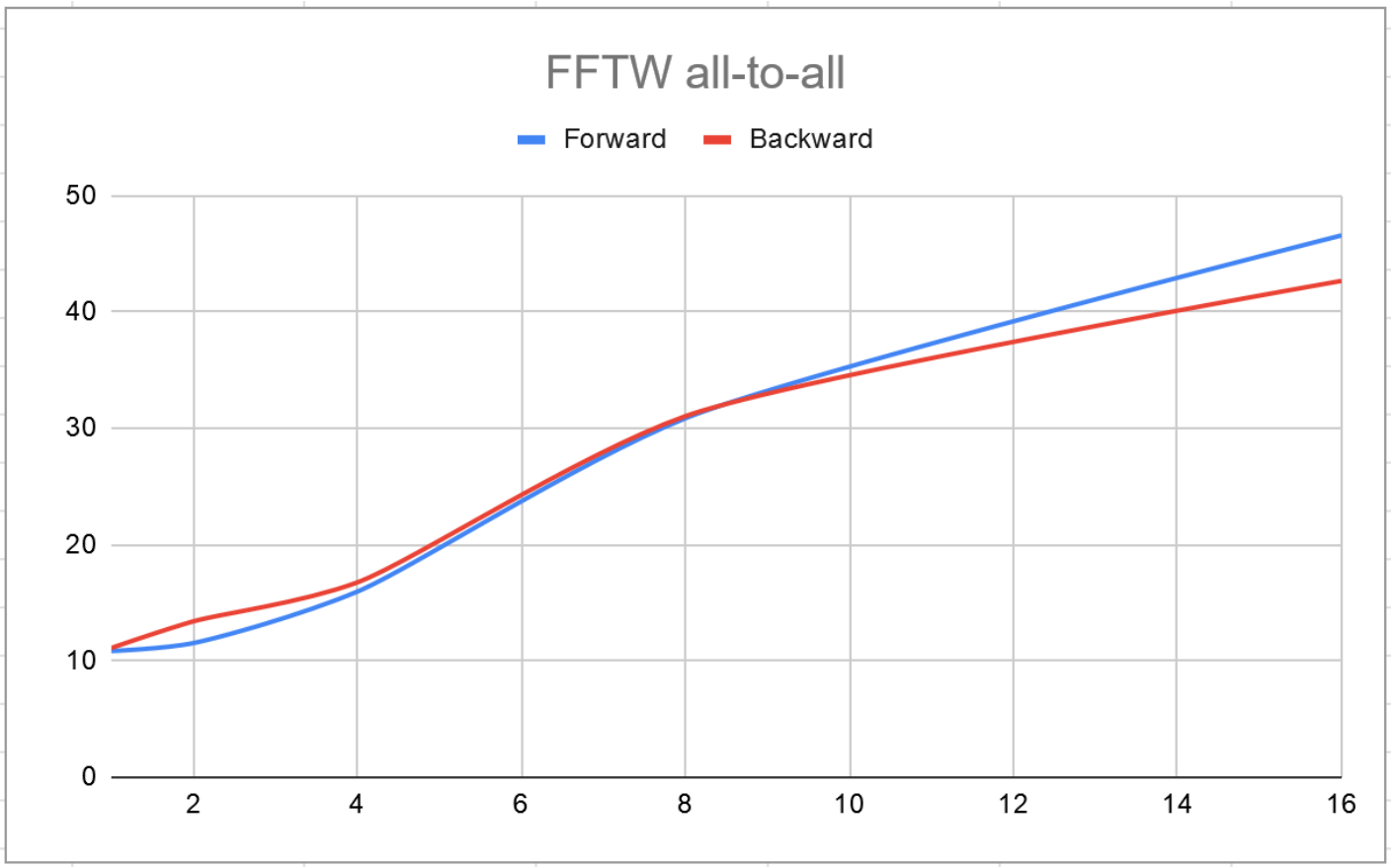
HPL

Nodes	Tflop/s
1	1,72
2	3,36
4	6,02
8	12,30
16	22,54



Benchmarking

FFTW



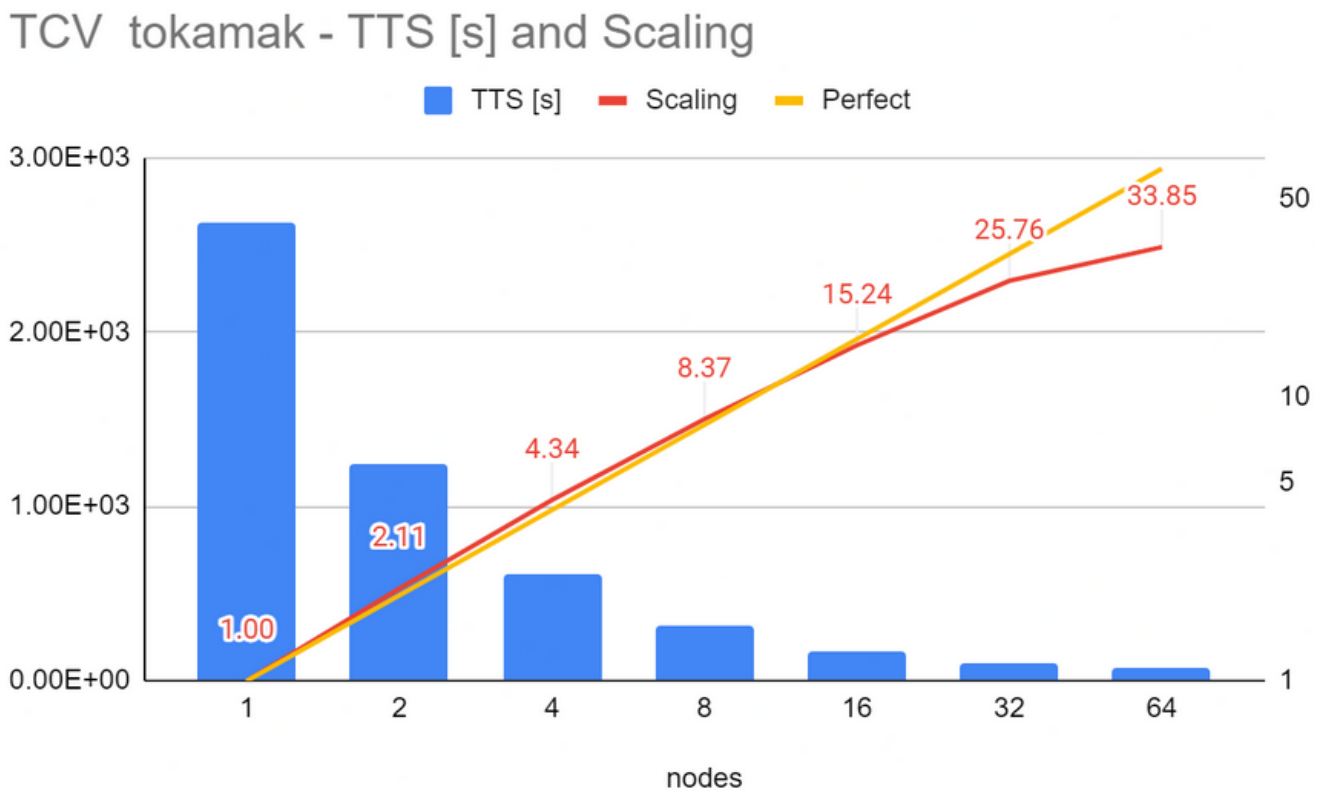
- Very sensitive to interconnect performance

EPFL

Benchmarking

TCV Tokamak

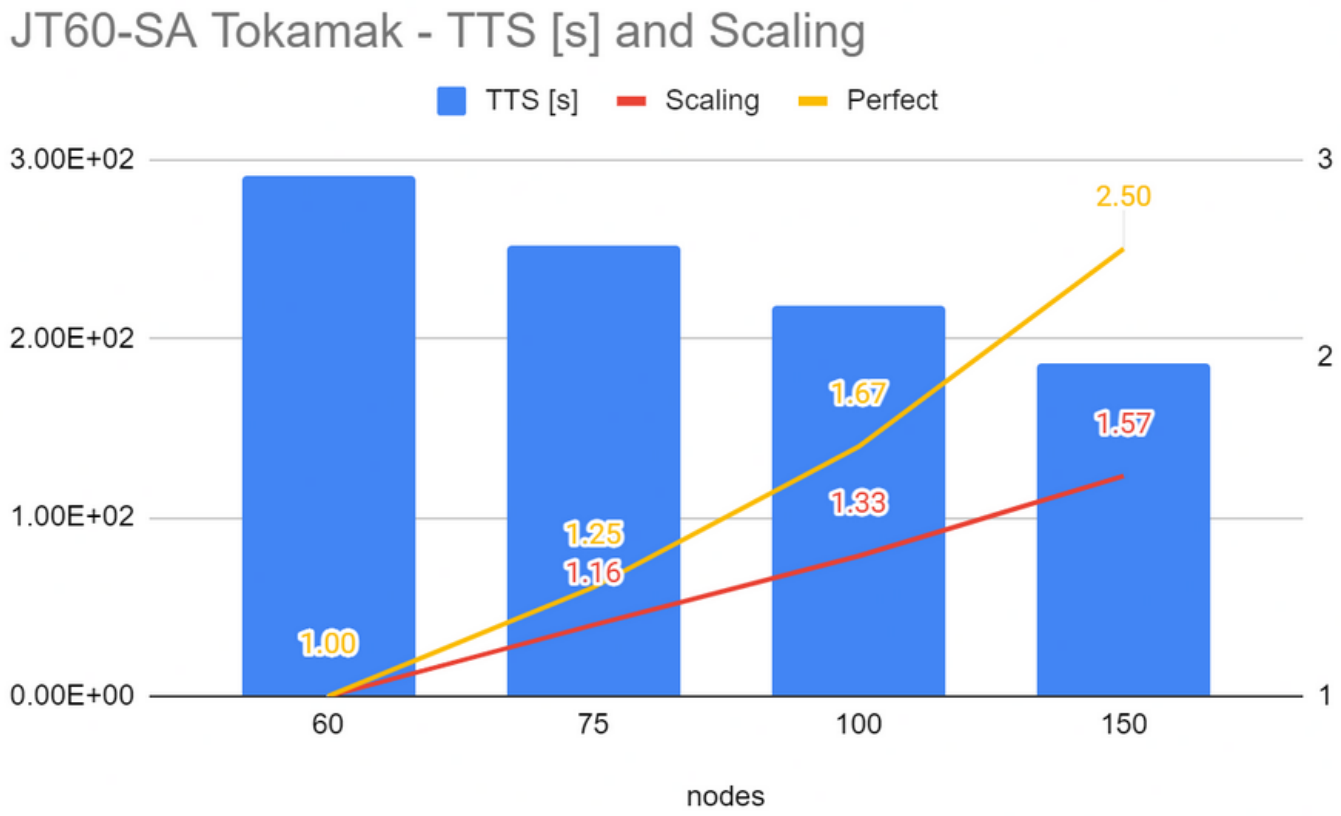
Simulating a plasma turbulence on the edge of the TCV tokamak (medium size):



Benchmarking

JT60-SA Tokamak

Simulating a plasma turbulence on the edge of the JT60-SA tokamak (large size):



Benchmarking

JT60-SA Tokamak: GCP vs SuperMUC-NG

Nodes	64	75
Configuration	TCV	JT60-SA
Time to solution per time step SuperMUC-NG [s]	0.44	9.1
Time to solution per time step GCP HPC VM [s]	0.77	25

- Comparable results using a relatively large number of nodes. (Note that SuperMUC-NG and GCP have different number of cores per nodes (48 vs 30)**!**)
- Very impressive considering the flexibility GCP is offering

EPFL

Problems Encountered

External/Hybrid Elastic Computing

- The availability of preemptible instances (prices are 60-80 % off)
- The price of regular compute instances
- Network bandwidth and latency between EPFL's LAN and GCP Infrastructure
  - Heavily impacts data sharing between the cloud-hosted storage space and the front-end node (in the Hybrid architecture)
- Compliance with our data management policies

EPFL

Problems Encountered

External Compute Resources

- The job walltime: without data storage device in the cloud the data transfer happens during the job
- We only ran a small example, left some open questions:
  - Is it possible to solve a real problem using this approach?
  - What is the size limit of the input dataset?

EPFL

Advantages

Expected

- Flexibility
- Infrastructure as Code management
- Excelent tooling / APIs / Documentation

EPFL

Advantages

Unexpected

- Scalability of all benchmarks (network is surprisingly good)
- Only limit was the availability of preemptible instances (up to 250 instances)
- Some use cases could be moved today to a cloud resource (courses, training, scalability testing, infrastructure validation)
- Access to compute resources (TPU) via an IP address
- Better foundation than expected (Slurm integration)

EPFL

Resources

Links

- Terraform recipes provided by the slurm-gcp project
  `https://github.com/SchedMD/slurm-gcp`
- Terraform modules and our patches on top of slurm-gcp
  `https://c4science.ch/diffusion/10965/`
- Spack stack definition
  `https://github.com/epfl-scitas/spack-packagelist/tree/releases/arvine`

EPFL

Questions?

Questions?

EPFL