HPC Cluster Deployment in the Cloud: Strategies for Scalability, Speed, and Self-Service

hpc-ch forum on Business Continuity / Crises Management & Energy Savings and Efficiency

Pablo Llopis, Edita Kizinevic, George Ioannidis, Carolina LIndqvist
And all other SCITAS colleagues.

May 23rd, 2024

EPFL

SCITAS @ EPFL

We provide a full-fledged HPC service for university staff and students.

3-4 HPC clusters, each between 200 and 420 nodes.

....provide researchers at EPFL access to scientific equipment and service expertise in high-performance computing that is efficient and customer-oriented.

... advancing the technology of the platform for the benefit of its users and the influence of EPFL

Agenda

Introduction

Experiences bridging on-premise HPC infrastructure with the cloud

- Cloud bursting models: a brief overview
- How this benefited our science community
- Technical implementation

Slurm power saving

- How this relates to cloud bursting

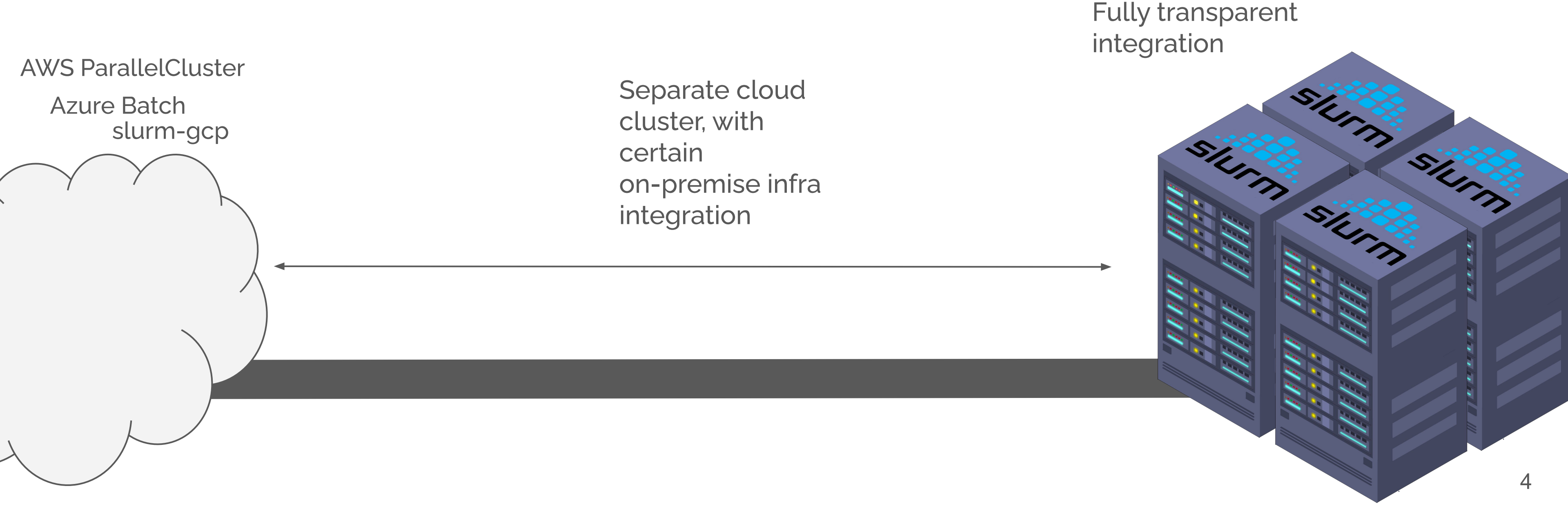
Looking towards the future (sorry for the buzzwords)

- Cloud-native solutions
- AI-assisted ops

Conclusions

Cloud bursting models

There's a full **spectrum** of how much integration you can achieve with your organisation's infrastructure.



Maintenance: Big Bang Interventions

Big service interventions can be **scary** (to the HPC staff), and **disruptive** (to the HPC users).

SCITAS recently experienced such a big maintenance in February 2024.

- Network replacement (new hardware, new net topology, new uplinks)
- New central storage system
- OS upgrade RHEL8 -> RHEL9
- New software stack
- PDU recall

Maintenance: Big Bang Interventions

A few user communities had **urgent needs** to keep running (conference deadlines, HPC training courses scheduled, etc)

We experimented with our first **cloud proof-of-concept deployment** to provide alternative HPC resources during on-premise downtime and reduced availability of resources.

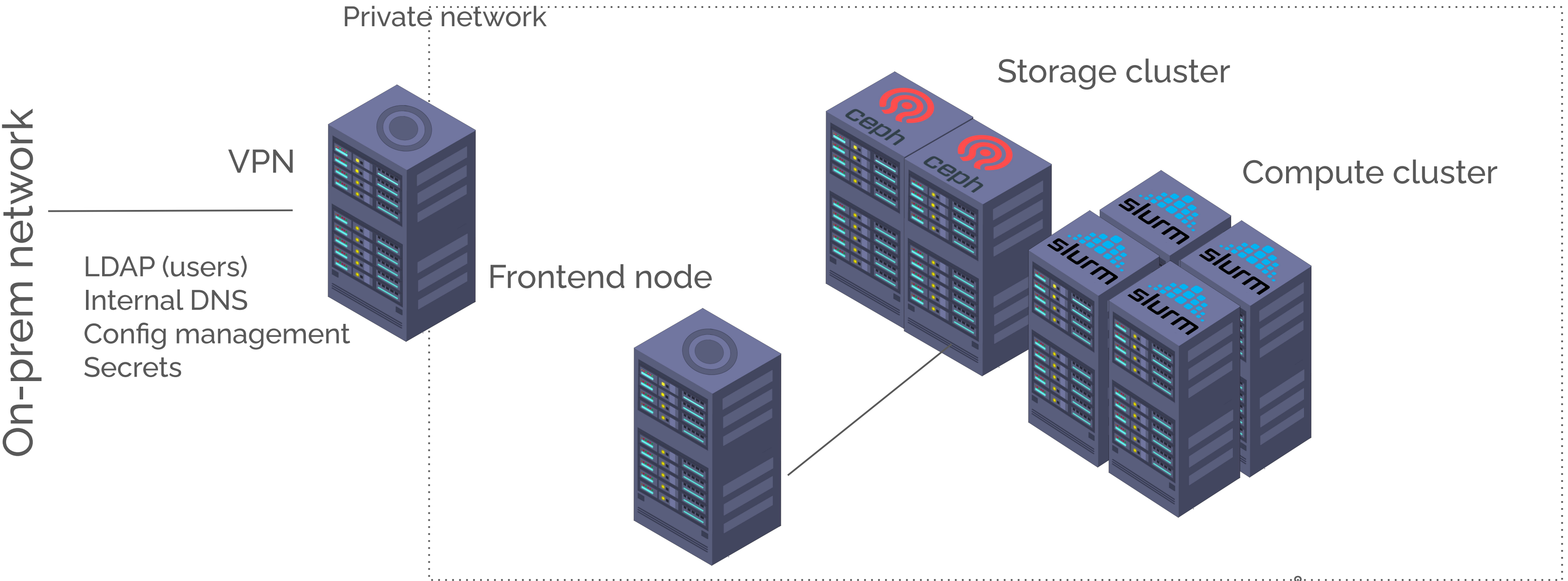
Dedicated cloud cluster:
Independence from on-prem
Redundancy

Cluster in the cloud proof-of-concept

Swiss cloud provider: **Exoscale**
Cluster size: 100 nodes (CPU only)

Goal: Provide **same "look and feel" as our on-premise cluster**.

Cluster in the cloud proof-of-concept



Technical implementation: cloud images

We build images for the cloud that are almost identical to those on-premise: allows **reuse of a common codebase**, and helps give the cloud cluster the same **look and feel** for end-users.

Images are **built automatically** via a pipeline using Hashicorp **Packer**, and also automatically uploaded to the cloud.

We build images for **frontend**, **controller**/admin, and **compute**.

Each of these images can be reused for several **cluster instances**.

HashiCorp Packer

Technical implementation: Cluster creation

Cluster resources are created with **Terraform**.
Infrastructure as Code all the way.
References the cloud images to spin up the cluster resources.

But this is not enough! Automatic creation of a cluster involves more than just creating the cluster resources. There are unique things about each cluster instance since they're meant to be isolated clusters for different user communities.

Orchestration of a cluster creation workflow.

HashiCorp Terraform

OpenTofu

Technical implementation: Orchestration of a cluster creation workflow

Automatic creation of a cluster involves orchestrating multiple steps in a workflow.

- Creation of cluster resources.
- Setting up VPN connection between cloud and on-premise.
  - Per-cluster instance keys for encrypted connections
  - Per-cluster instance private ssh keys for admin access
- Restricting access to target user groups
- Creation of Slurm accounts
- Verification that the cluster instance is working correctly.

ANSIBLE

There’s more! Dynamic resource bursting

Often there’s no reason to create a cluster on the cloud of a static size.
Well-known fact: Cluster utilisation is rarely 100%. Clouds can be expensive!

Solution: On-demand bursting.

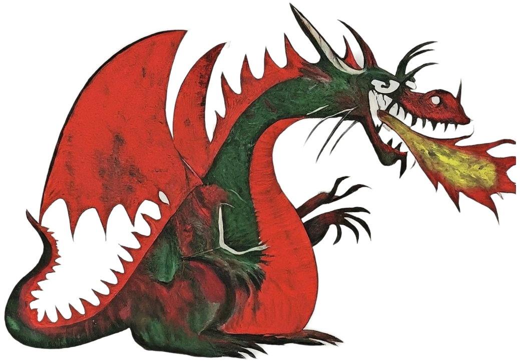
Leverages Slurm’s **power saving mechanism**. When nodes have been **idle** for X amount of time, they are **stopped or destroyed**. From Slurm’s perspective, they’re “*idle*” and “*powered off*”.

When jobs are **pending**, powered off nodes are “powered up”. In the cloud case: VMs are **instantiated**.

Beware! There be dragons

Slurm is known to mess up the **node state machine**.
Can end up with nodes “stuck” in many incompatible states.
Requires manual intervention, not well understood.

Paradigm clash:
Terraform is highly declarative.
Slurm’s power saving mechanism is event based.
Difficult to reconcile Terraform’s method with Slurm’s method.



Slurm-burst project aims to provide a generic multi-cloud bursting solution. Can also work for on-premise power save!
Work in progress: not open source yet, but will be.

Future work

Our end goal is that our user community can create clusters via **self-service** interface. Lots of automation needed!

Automation with high reliability and resiliency: we don't want a bad user experience where a cluster is created ($$) which does not work.

Evaluate AI chatbot integration that is not just able to do question answering, but also interact with our systems via function calling.
One interface to rule them all? (e.g. create a cluster, look up my consumption, ask about planned interventions, ...)