Cambridge-1: An Approach to Dynamic Multitenancy

Simon Traill, NGC Site Reliability Engineering | Lustre User Group@CIUK / 30th November, 2022

Cambridge-1: An Approach To Dynamic Multitenancy

...Using NodeMaps and GSSAPI

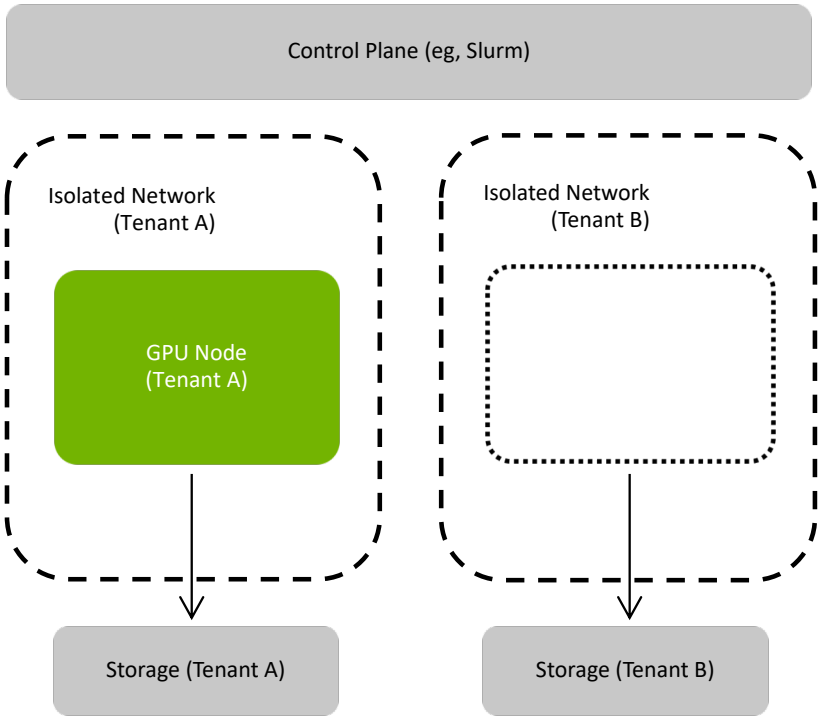
On July 6, 2021, NVIDIA opened the Cambridge-1 supercomputer for use by life sciences providers.

- Slurm cluster consisting of 80 DGX A100 systems.
- Multiple tenants, each needing:
  - Isolated compute
  - **Isolated storage**

To support the datacentre, we were asked to create a dynamic storage platform using Lustre on DDN Exascaler.

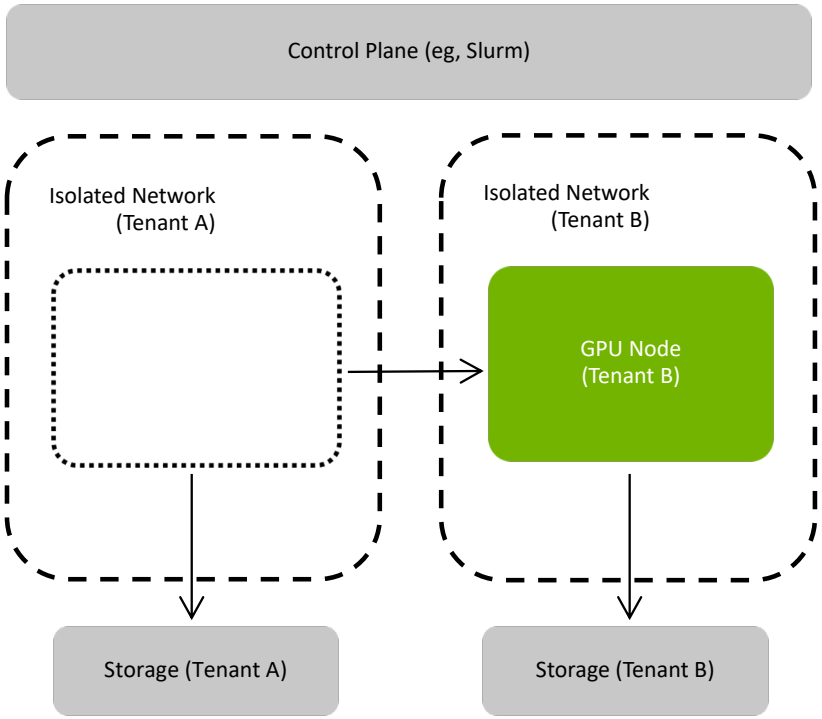
(Dynamic) Multitenancy in a Supercomputer

- **Isolated Partitions:**
  Tenants typically isolated at the compute, network and storage (Lustre) layers.



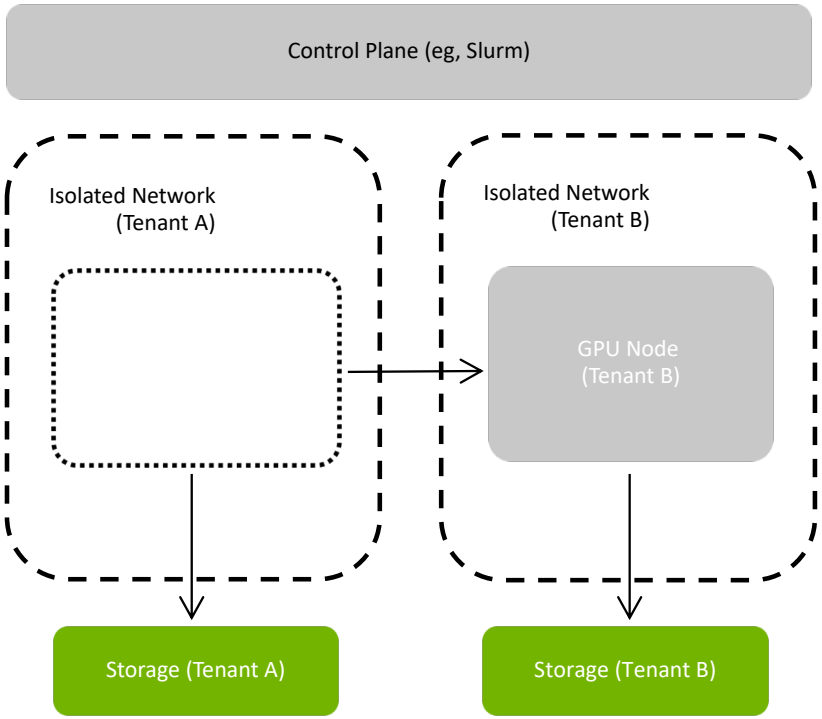
(Dynamic) Multitenancy in a Supercomputer

- **Isolated Partitions:**
  Tenants typically isolated at the compute, network and storage (Lustre) layers.
- **Nodes are more dynamic:**
  Nodes need to switch partitions more frequently than tenants as we allocate resources.



(Dynamic) Multitenancy in a Supercomputer

- **Isolated Partitions:**
  Tenants isolated at the compute, network and storage (Lustre) layers.
- **Nodes might be dynamic:**
  Nodes can switch partitions.
- **How should we handle Storage?**
  - Multiple Lustre filesystems, Isolated Storage Networks?
    - Need to automate (more) network provisioning, be it ethernet, IB, LNET
    - Might need multiple storage appliances (sorry, DDN).
  - Secure user level authentication, eg Kerberos?
    - Need to install and operate Kerberos!



Objectives

Infrastructure is hard enough already

- **Encapsulated:**
  As few dependencies on external infrastructure as possible.
- **Reusable:**
  Deploy our multitenancy system again and again amongst differing surrounding infrastructure.
- **Testable:**
  Test our system, ideally like software, without needing a spare supercomputer.
- **Defined Public Interface:**
  Hide complexity, and reveal only a simple (REST, in this case) API

Tenant lifecycle

```
GET      /tenants
POST     /tenants/<tenant>
DELETE   /tenants/<tenant>
```

Per-tenant **node** lifecycle

```
GET      /tenants/<tenant>/<nids>
POST     /tenants/<tenant>/<nids>/<nid>
DELETE   /tenants/<tenant>/<nids>/<nid>
```

Objectives

Infrastructure is hard enough already

- **Encapsulated:**
  As few dependencies on external infrastructure as possible.
- **Reusable:**
  Deploy our multitenancy system again and again amongst differing surrounding infrastructure.
- **Testable:**
  Test our system, ideally like software, without needing a spare supercomputer.
- **Defined Public Interface:**
  Hide complexity, and reveal only a simple (REST, in this case) API

Tenant lifecycle

```
GET      /tenants
POST     /tenants/<tenant>
DELETE   /tenants/<tenant>
```

Per-tenant **node** lifecycle

```
GET      /tenants/<tenant>/<nids>
POST     /tenants/<tenant>/<nids>/<nid>
DELETE   /tenants/<tenant>/<nids>/<nid>
```

Our system needs to handle **Authentication, Authorization** and **Isolation.**

Authentication

With **Nodemaps**, **Filesets** and **SSK Keys**

On a single lustre filesystem:

- Create one **nodemap** per tenant.
  *Each NodeMap contains a list of **NIDs** (client nodes).*

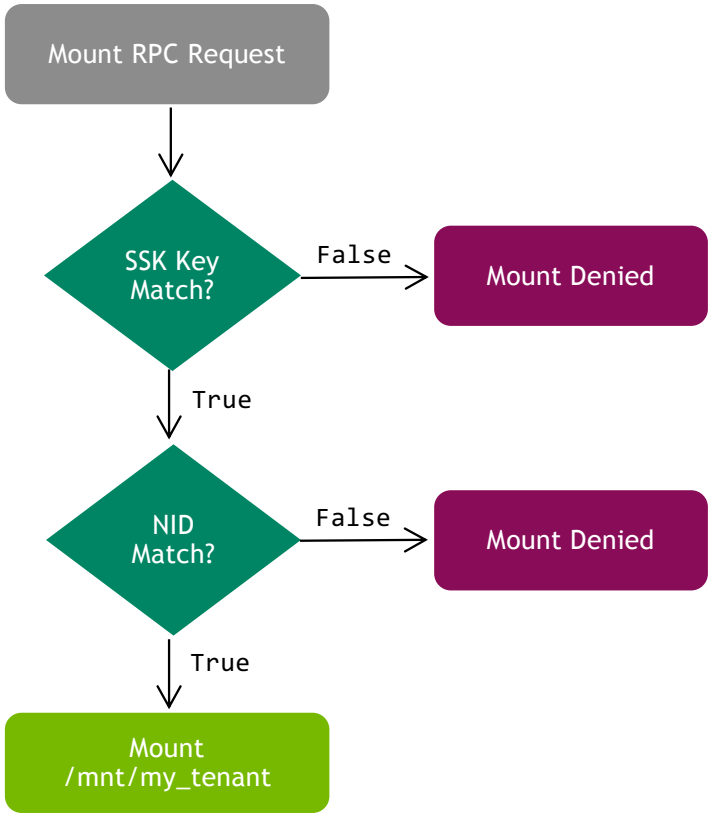
```
lctl nodemap_add my_tenant
```

- Create one **fileset** for each tenant.
  *Each **Fileset** allow access only to a top level directory.*

```
lctl nodemap_set_fileset --name my_tenant \
  --fileset /my_tenant
```

- Create one **SSK Key** for each tenant.
  *Clients must possess the **SSK Key** to mount the **fileset**.*

```
lgss_sk -t client -f my_fs \
  -n my_tenant \
  -w my_secret_key
```



Isolation

Data in flight needs to be private

Encryption?

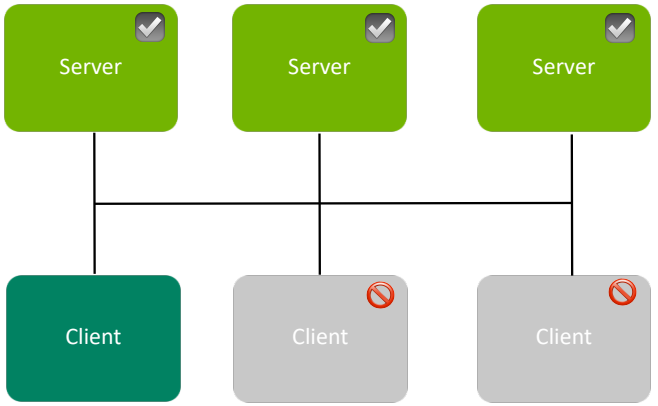
- **Encryption brings a 30%-70% throughput penalty for IOPs**
  - Perhaps not.

Infiniband allows for "Limited" or "Full" partition membership.

- **Full** network peers can talk to anyone
- **Limited** network members can only talk to **full** members

Solution:

- **Lustre serving** nodes use **full** IB membership.
- **Lustre client** nodes use **limited** IB membership.
- Fewer moving parts.
- We do allow encryption using (GSSAPI skpi) to be enabled as a config setting for ethernet users...
- ...who, generally, care less about performance.



Automation

How can we expose this as an API?

Lustre has a high level API... but not for the constructs we're using.

- No C developers; limited time.
- Instead, we implemented an API over `lctl` itself.
- Bind the concepts of **nodemaps**, **filesets** and **ssk keys** together into a **tenant.**
- Allow client node **NIDs** to be added and removed from each **tenant**.

Tenant lifecycle

```
GET       /tenants
POST      /tenants/<tenant>
DELETE    /tenants/<tenant>
```

Per-tenant **node** lifecycle

```
GET       /tenants/<tenant>/<nids>
POST      /tenants/<tenant>/<nids>/<nid>
DELETE    /tenants/<tenant>/<nids>/<nid>
```

Automating Tenants

An API to CRUD Lustre "tenants"

Tenant provisioning is time consuming.

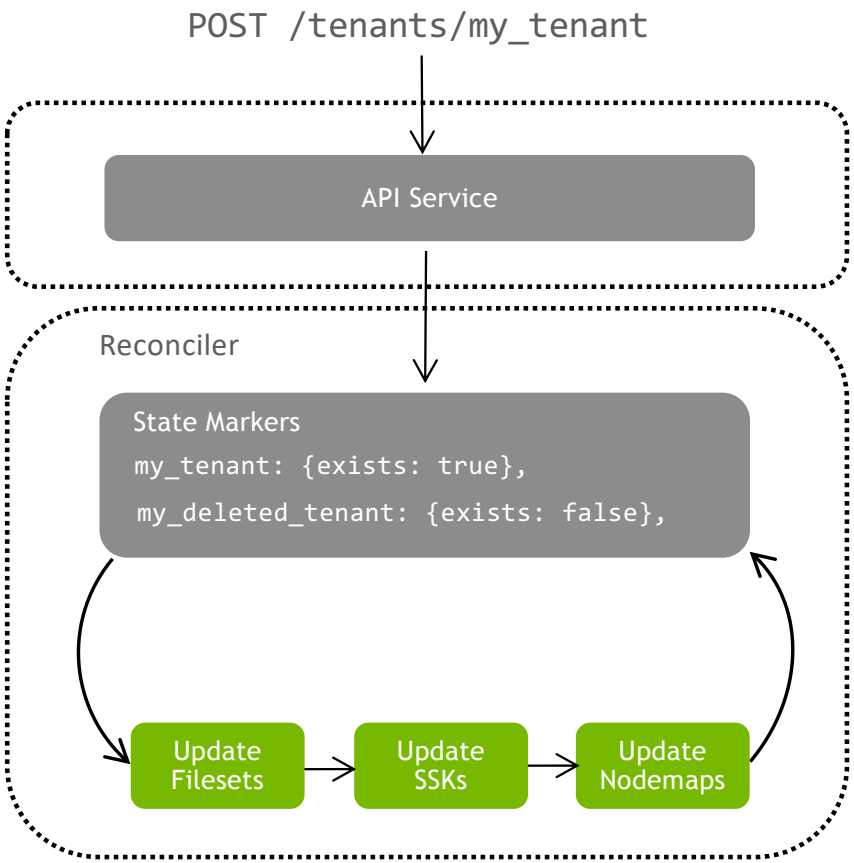
- Nodemap, fileset and SSK key creation take a while (minutes).
- `POST /tenants/<tenant>`:
  - Creates a state marker stating that "<tenant> should be created".
  - Returns a pollable job id.

```
$ curl -X POST http://lustre-api-endpoint/v1/tenants/my_tenant
{"message": "Changes were queued for processing", "jobid":
"9e3d6fd3-70e9-4174-ab80-56d8d29e3428"}
```

Tenants are reconciled asynchronously.

An asynchronous reconciliation process constantly ensures that:

- A **fileset** exists for each **tenant**.
- An **SSK** has been created for each tenant.
- A **nodemap** exists for each tenant.
- The above are removed when a tenant has been **deleted**.
- The reconciler invokes `lctl` (and other commands) on the MGS.



Automating Client Nodes

Node API updates must be (somewhat) quicker

Node provisioning changes happen far more frequently than those for tenants.

- `POST /tenants/<tenant>/nids/<nid>:`
  - Enqueues a nodemap change operation; alters client nodes to be granted or denied access to a Nodemap.
  - Returns a pollable job id.
  - Can be synchronous.
  - Needs to be quicker than tenant setup.

```
$ time curl -X POST http://lustre-api-endpoint/v1/tenants/my_tenant/nids?sync --data "{"nids": ["192.168.0.13"]}
real    0m2.569s
```

Tenant lifecycles can be slow; node allocations must be fast(ish).

Automating Client Nodes

More haste, less speed

Lctl (or rather, the MGS) does not guarantee the order of rapid node updates.

Ordering of updates is not guaranteed (probably by design):

If you add a node to a nodemap, remove it, then add it again – quickly – the end state is unknown.

- The node might have been added, or it might not.
- Fine for a CLI; less good for a REST API.

However (after *rather a lot* of testing):

- Waiting for each NID change to be reflected via `lctl get_param` preserves ordering (but is slower).
- Updates to *unrelated* NID ranges – different nodes - can be made in parallel without waiting.

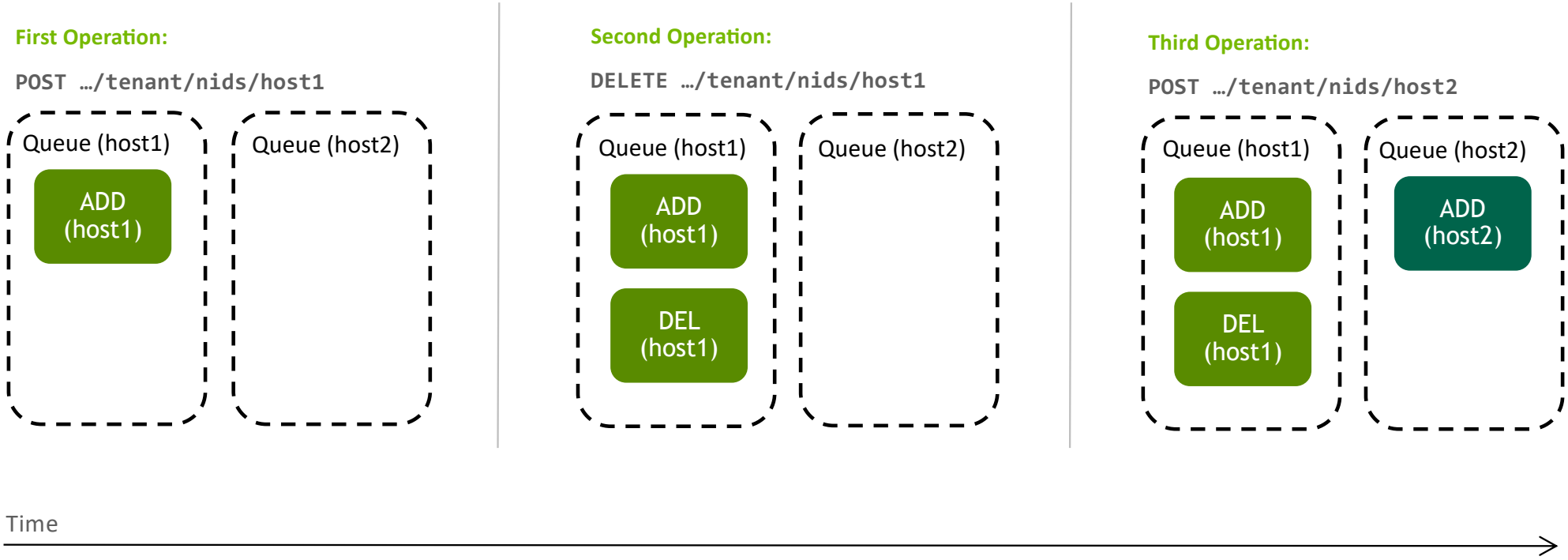
...plus, in general, swamping `lctl` is probably not a good idea.

Automating Client Nodes

Solution: enqueue NID updates per-node

Node (NID) updates therefore use a **per-node queueing system.**

- Slow(er) for individual nodes; but
- Fast(er) across many.

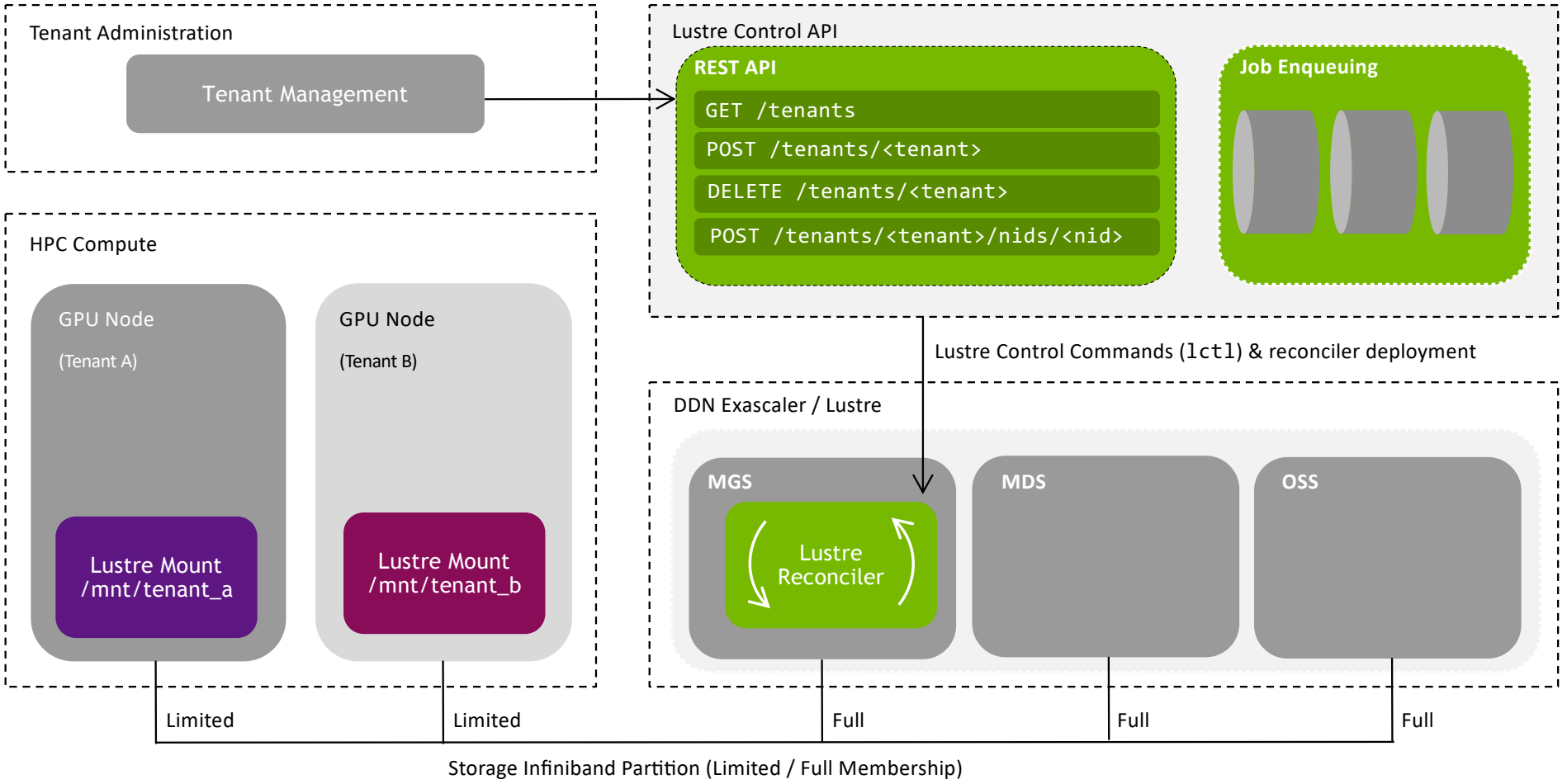


Put It All Together

- Tenant lifecycles handled using **nodemaps**, **filesets**, **SSK Keys**, and a **reconciler**
- Node isolation handled by adding or removing **NIDs** from **nodemaps,** throttled by **per-node queues**
- Interactions happen via REST API

Final Design

Drop in containers, run adjacent to a vanilla DDN / Lustre



Thoughts

Nodemap authN probably shouldn't depend on node NIDs if other GSSAPI methods are present.

- NIDs can trivially be spoofed if a client's `root` account is compromised.
- The SSK key is important to authN a client node. The NID is not.
- This would allow us to remove the NID update queueing system.

Your milage with GSSAPI (SSK Keys and Kerberos) may vary.

- We've seen various context related race conditions which can hang mounts on occasion.
- More usage of GSSAPI in the wild might help!

It would be great if Lustre had a high level (REST? GraphQL?) API over constructs like nodemaps.

- Although then you would have missed out on this presentation.

The DDN virtual appliance was extremely helpful:

- ...thanks to DDN in general and Rich Mansfield in particular for providing us with the lovely lightweight-vagrant-exascaler.


NVIDIA
Thank you!
Any questions? straill@nvidia.com