New approaches in scientific computing: the design of a cloud-ready HPC infrastructure

Isac Pasianotto

Supervised by: prof. *Ruggero Lot*
Cosupervised by: prof. *Stefano Cozzini*

Graduation session: M.Sc. in *Data Science and Scientific Computing*

18th September 2024

University of Trieste

🤔 What is this thesis about?

Investigating the possibility of adopt a cloud-native approach based on container orchestration in a private HPC cluster to provide it in a SaaS fashion

📍 We are here

- 🧩 Background essentials in a nutshell
- 🏓 Communication & CNI plugins
- ⏱️ Network benchmarks
- 🐍 Dask: case of study for scientific computing in Kubernetes
- 📌 Final consideration

*“Cloud computing has been coined as an umbrella term to describe a category of sophisticated **on-demand computing services** initially offered by commercial providers”*

– R. Buyya et al. Cloud Computing: Principles and Paradigms

What is *"Cloud Computing"*

> *"Cloud computing has been coined as an umbrella term to describe a category of sophisticated **on-demand computing services** initially offered by commercial providers"*
>
> *– R. Buyya et al. Cloud Computing: Principles and Paradigms*

☁️ ***for us:***

*Cloud Computing is the result of the **evolution** of the computing concept **driven by the technology** improvements and by users' requirements.*

Virtualization

- Isolated environment with its own OS and resources
- **Overhead** for the hypervisor

🖥️ Virtualization

- Isolated environment with its own OS and resources
- **Overhead** for the hypervisor

📦 Containerization

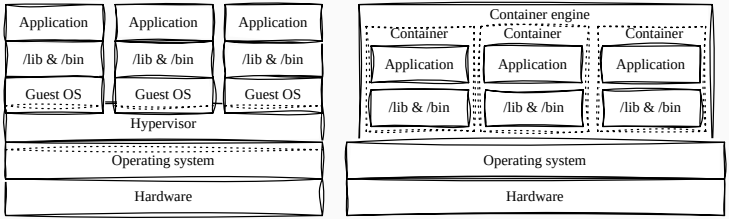
- Host's processes → they share the Kernel with the host
- Build over namespaces, cgroups, and capabilities

Virtualization

- Isolated environment with its own OS and resources
- **Overhead** for the hypervisor

Containerization

- Host's processes → they share the Kernel with the host
- Build over namespaces, cgroups, and capabilities



- The adoption of Cloud for production application was very immediate
- The idea of performing the same transition for HPC is not new
- Many effort in the early 2010s

- The adoption of Cloud for production application was very immediate
- The idea of performing the same transition for HPC is not new
- Many effort in the early 2010s

☠️ *The results were tragic!*

- Only considered Public Clouds
- Containerization was still in an embryonic state → VMs only
- *Multiple performance shortcoming* were reported

⏳ *Network latency, and I/O rapidly become the bottlenecks of the system*

🤨 *So why persist any longer?*

There are 3 main reasons for not giving up:

🔐 Private cloud

🤨 *So why persist any longer?*

There are 3 main reasons for not giving up:

🔐 Private cloud

⚙️ Technological progress

- Containers are the *de-facto standard*
- Computational drop performance completely negligible
- Many improvements and alternative solutions in for networking

🤨 *So why persist any longer?*

There are 3 main reasons for not giving up:

🔐 **Private cloud**

⚙️ **Technological progress**

- Containers are the *de-facto standard*
- Computational drop performance completely negligible
- Many improvements and alternative solutions in for networking

⚖️ **Performance/Usability trade-off**

- Simplifies the use and scalability of the cloud (SaaS)
- Container orchestration: simplify the interaction with the cluster for both the user and the administrator
  - Automatic scaling and load balancing
  - Availability, self healing and fault tolerance
  - Standardized API $\rightarrow$ portability
  - ...

📍 We are here

- 🧩 Background essentials in a nutshell
- 🏓 Communication & CNI plugins
- ⏱️ Network benchmarks
- 🐍 Dask: case of study for scientific computing in Kubernetes
- 📌 Final consideration

Def.

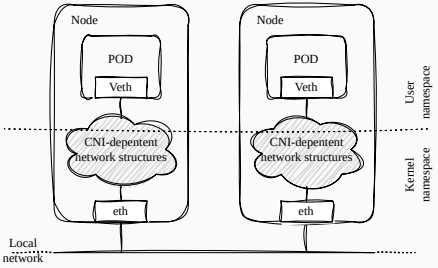
A CNI plugin is responsible for inserting a network interface into the container network namespace (e.g., one end of a virtual ethernet (veth) pair) and making any necessary changes on the host (e.g., attaching the other end of the veth into a bridge).

– Red Hat sysadmin's guide

Def.

A CNI plugin is responsible for inserting a network interface into the container network namespace (e.g., one end of a virtual ethernet (veth) pair) and making any necessary changes on the host (e.g., attaching the other end of the veth into a bridge).

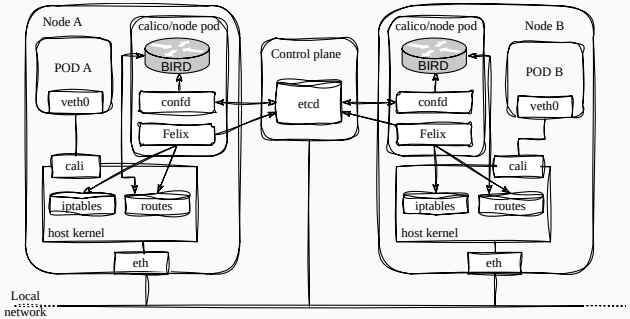
– Red Hat sysadmin's guide



👉 *Calico*

- Based on **BGP** routing protocol
- *Felix*: agent which configures the network
- *BIRD*: BGP client for routing updates
- *confd*: ensure the configuration is up-to-date

`Flannel`
`Cilium`



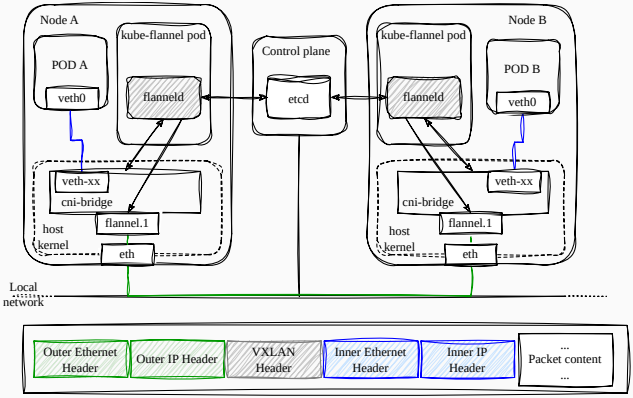
CALICO

`Calico`

👉 *Flannel*

- Based on **VXLAN** encapsulation
- Runs a flanneld binary agent responsible for managing the network configuration, IP address management and subnets

`Cilium`

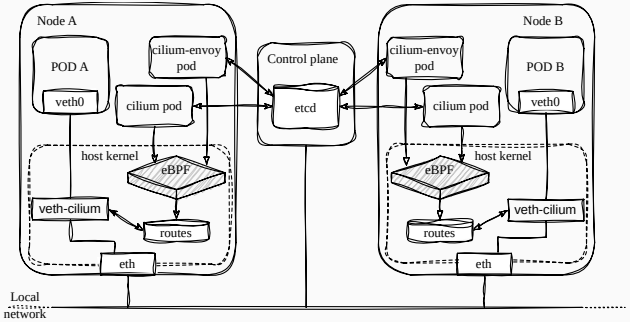


flannel

`Calico`
`Flannel`

👉 *Cilium*

- Based on **eBPF** (extended Berkeley Packet Filter)
- Runs a agent which configures the network
- similar to a JIT compilation in a sand-boxed environment inside the kernel space



cilium

📍 We are here

🧩 Background essentials in a nutshell

🏓 Communication & CNI plugins

⏱️ Network benchmarks

🐍 Dask: case of study for scientific computing in Kubernetes

📌 Final consideration

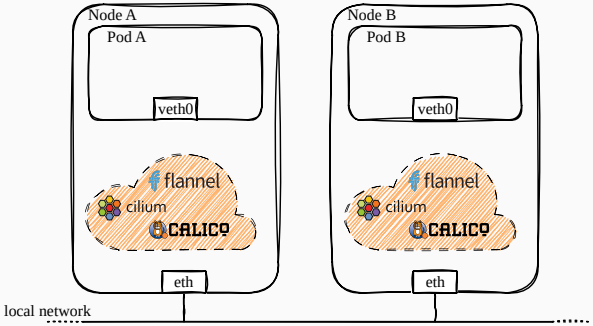
- Many CNI plugins, with many different strategies and implementations
- CNI plugin is the main source of *overhead* in any Kubernetes cluster
- This is a **very critical choice**...

- Many CNI plugins, with many different strategies and implementations
- CNI plugin is the main source of *overhead* in any Kubernetes cluster
- This is a **very critical choice**...

🤔 *How to choose the best solution?*

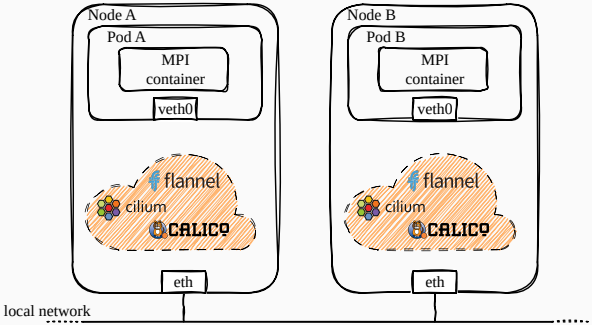
💡 *The idea:*

- Spawn 2 pods in 2 (different) nodes



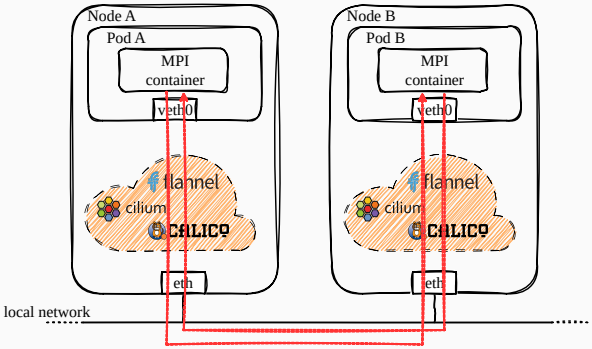
The idea:

- Spawn 2 pods in 2 (different) nodes
- Each pod will have a container running an MPI process

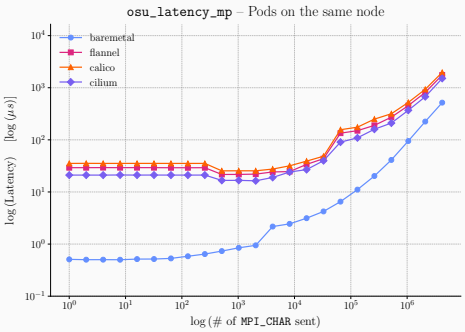


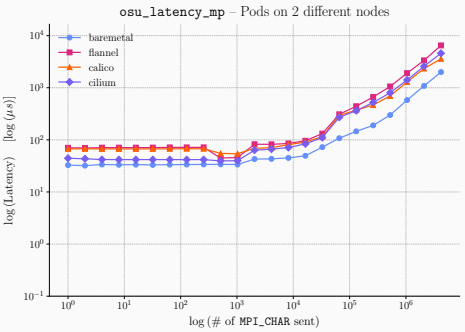
💡 *The idea:*

- Spawn 2 pods in 2 (different) nodes
- Each pod will have a container running an MPI process
- Perform ping-pong tests between the 2 pods 🏓



Latency results

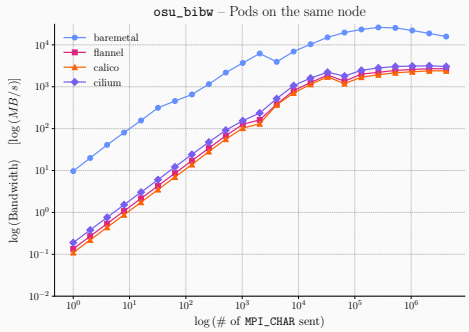


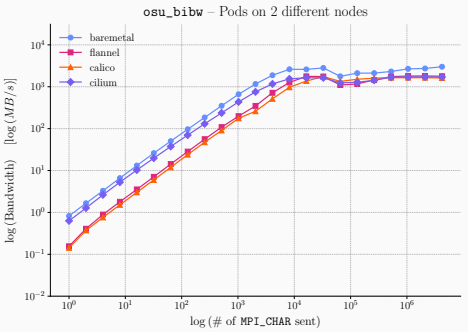


[μs]	Bare metal	Calico	Flannel	Cilium
Min	0.5	35.3	29.2	21.1
Max	0.5e3	2.0e3	1.8e3	1.5e3

[μs]	Bare metal	Calico	Flannel	Cilium
Min	32.7	67.5	70.4	44.4
Max	2.0e3	3.6e3	6.5e3	4.6e3

Bandwidth results





$[MB/s]$	Bare metal	Calico	Flannel	Cilium
Min	9.69	0.11	0.14	0.19
Max	1.58e3	2.40e3	2.69e3	3.04e3

$[MB/s]$	Bare metal	Calico	Flannel	Cilium
Min	0.82	0.14	0.16	0.63
Max	3.016e3	1.62e3	1.78e3	1.71e3

🥇 And the Winner Is...

- There is not a absolute winner
- Some CNI pluigns are better in some scenarios, and worse in others
- We want a general purpose CNI plugin:

- There is not a absolute winner
- Some CNI pluigns are better in some scenarios, and worse in others
- We want a general purpose CNI plugin:

cilium

🧩 Background essentials in a nutshell

🏓 Communication & CNI plugins

⏱️ Network benchmarks

🐍 Dask: case of study for scientific computing in Kubernetes

📌 Final consideration

What is Dask?

- Flexible library for parallel computing in Python
- Allow to *distribute the computation* **without managing the parallelism**
- Has a same interface of *NumPy*, *Pandas* and the most popular libraries

What is Dask?

- Flexible library for parallel computing in Python
- Allow to *distribute the computation* **without managing the parallelism**
- Has a same interface of *NumPy*, *Pandas* and the most popular libraries
- *Allows the user to interact with the cluster directly from the code!*

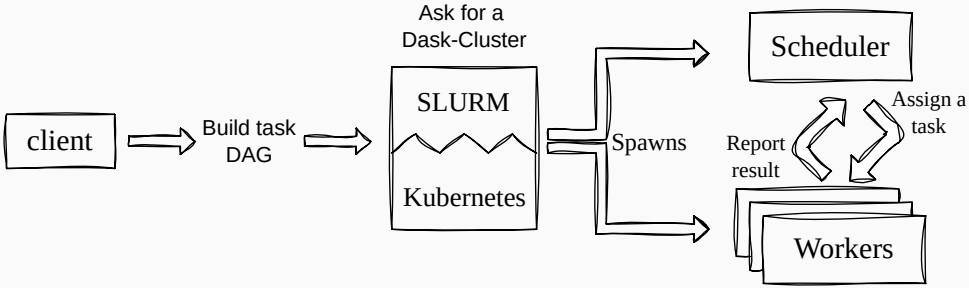
What is Dask?

- Flexible library for parallel computing in Python
- Allow to *distribute the computation* **without managing the parallelism**
- Has a same interface of *NumPy*, *Pandas* and the most popular libraries
- *Allows the user to interact with the cluster directly from the code!*

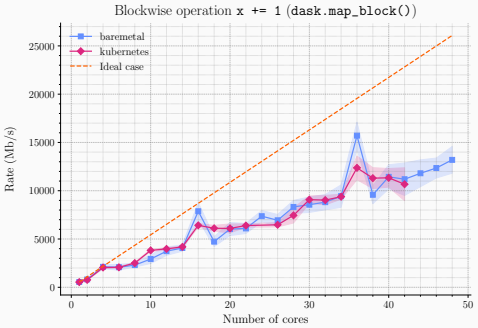
```python
from dask_jobqueue import SLURMCluster
cluster = SLURMCluster(
    job_cpu=8,
    job_mem='32GB',
    account='project',
    queue='partition',
    walltime='00:30:00',
)
cluster.adapt(
  minimum_jobs=2,
  maximum_jobs=10)
```

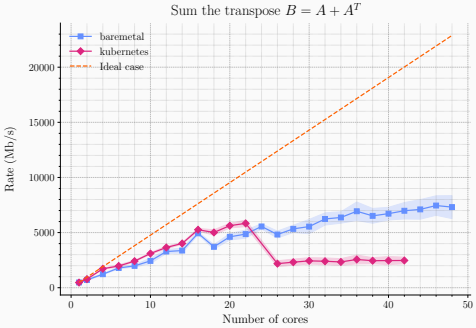
```python
from dask_kubernetes.operator import KubeCluster
cluster = KubeCluster(
    name='my-cluster',
    resources={ 'requests': {
        'cpu': '4', 'memory': '16Gi'
      },
                'limits': {
        'cpu': '8', 'memory': '326Gi'
      }
    }
)
cluster.adapt(minimum=2, maximum=10)
```

🧐 How we are going to use Dask?



Results


Blockwise operation x += 1 (dask.map_block())
baremetal
kubernetes
Ideal case
Rate (Mb/s)
Number of cores
0
5000
10000
15000
20000
25000
0
10
20
30
40
50


Sum the transpose $B = A + A^T$
baremetal
kubernetes
Ideal case
Rate (Mb/s)
Number of cores
0
5000
10000
15000
20000
0
10
20
30
40
50

🧩 Background essentials in a nutshell

🏓 Communication & CNI plugins

⏱️ Network benchmarks

🐍 Dask: case of study for scientific computing in Kubernetes

📌 Final consideration

🤔 Can we use Kubernetes to deploy and manage HPC applications?

🤔 Can we use Kubernetes to deploy and manage HPC applications?

- 🥳 *Yes!*, but with *some limitation.*

🤔 Can we use Kubernetes to deploy and manage HPC applications?

- 🥳 *Yes!*, but with *some limitation.*
- 🤦‍♂️ Times are **not** mature yet for a full migration
- ⚠️ In the **communication bounded** application, the performance gap is **not acceptable**
- 🤏 In all the other cases, the overhead is negligible

🤔 Can we use Kubernetes to deploy and manage HPC applications?

- 🥳 *Yes!*, but with *some limitation.*
- 🤦‍♂️ Times are **not** mature yet for a full migration
- ⚠️ In the **communication bounded** application, the performance gap is **not acceptable**
- 🤏 In all the other cases, the overhead is negligible
- 📊 **Kubernetes** comes with a lot of benefits that improve the user experience
- 🤞 The technology evolution of last years make us be optimistic about the future

Mellanox

- Use Inifniband instead of Ethernet
- Should mitigate the communication problems

Mellanox

- Use Inifniband instead of Ethernet
- Should mitigate the communication problems

Multus

- Attach **multiple network interfaces** to a pod
- **Split traffic** among different interfaces (e.g. data on Infiniband, checks on Ethernet)
- Allow to choose the desired CNI plugin

Thank you for your attention