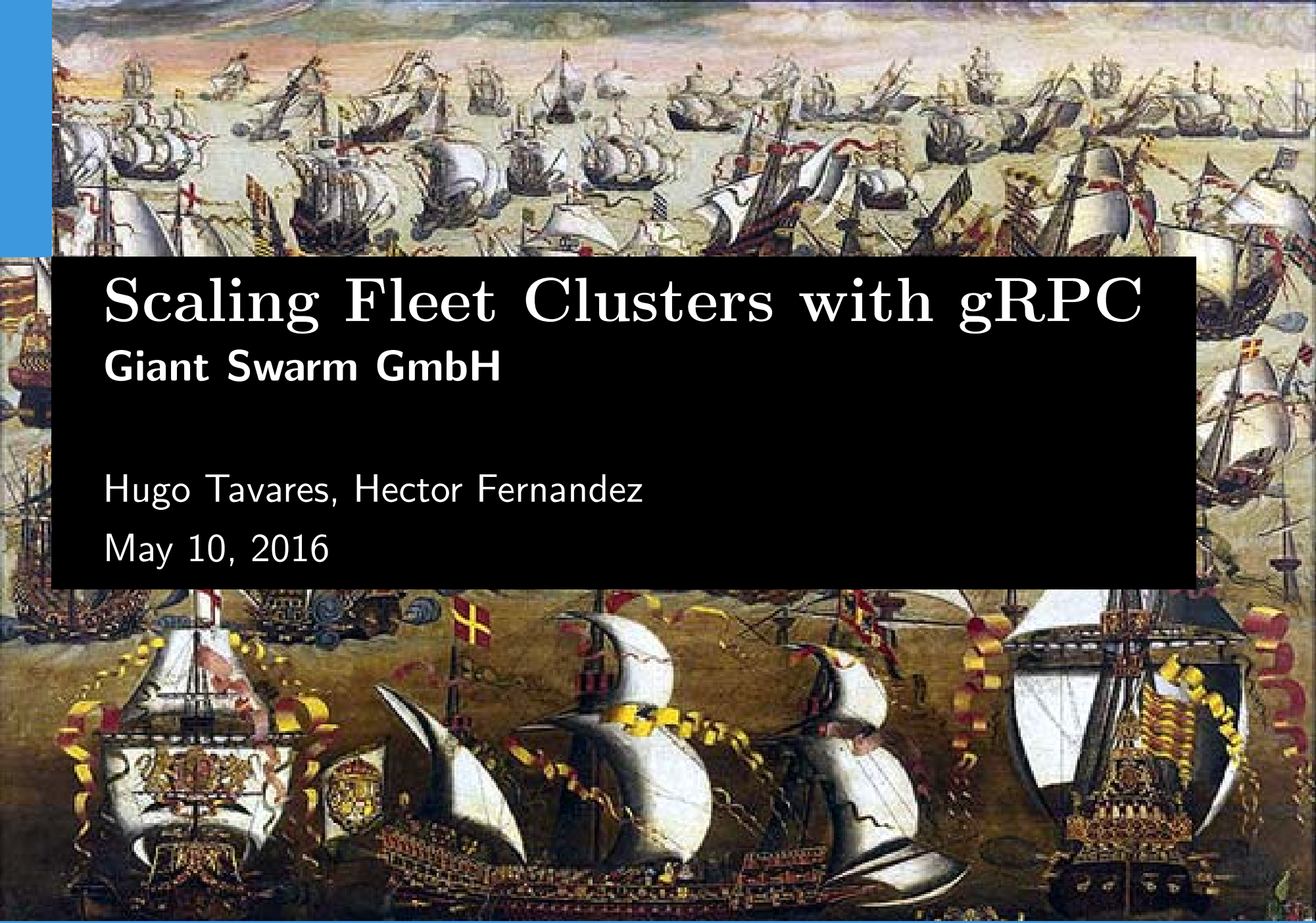


Scaling Fleet Clusters with gRPC

Giant Swarm GmbH

Hugo Tavares, Hector Fernandez

May 10, 2016

Outline

About me

- Infrastructure Engineer **@giantswarm**
- Love HPC, monitoring & performance :)
- Have fun with distributed systems
- Slow runner
- Spanish who DON'T like siesta

Fleet Today

init distributed systemd

- Manage systemd units in a cluster
- Talks to systemd via dbus
- Use etcd as storage and coordination system

Giant Swarm

Fleet Today

init distributed systemd

- Manage systemd units in a cluster
- Talks to systemd via dbus
- Use etcd as storage and coordination system

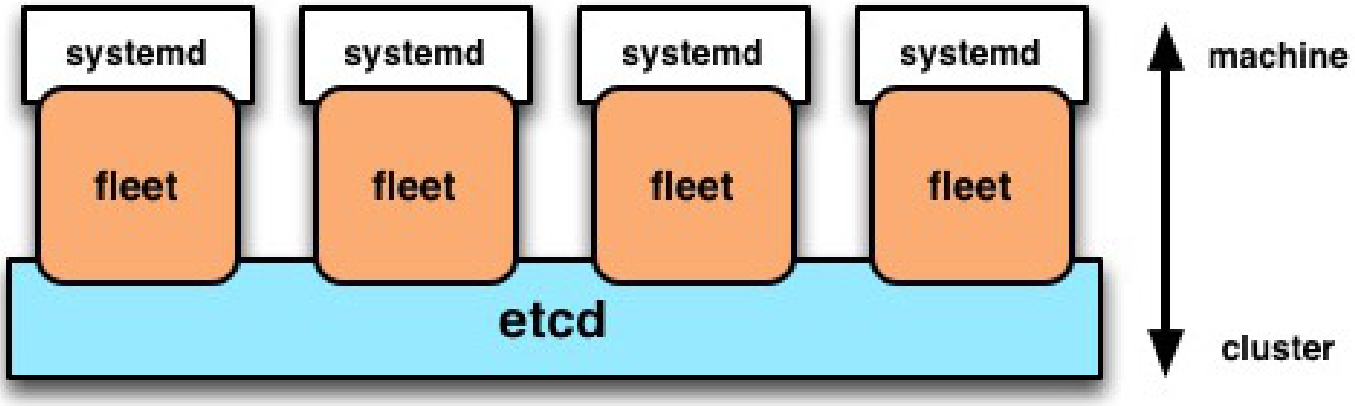


Figure: fleet overview

Giant Swarm

Fleet Today

The architecture is composed of two main building blocks:

- Engine (master)
- Agent (workers)

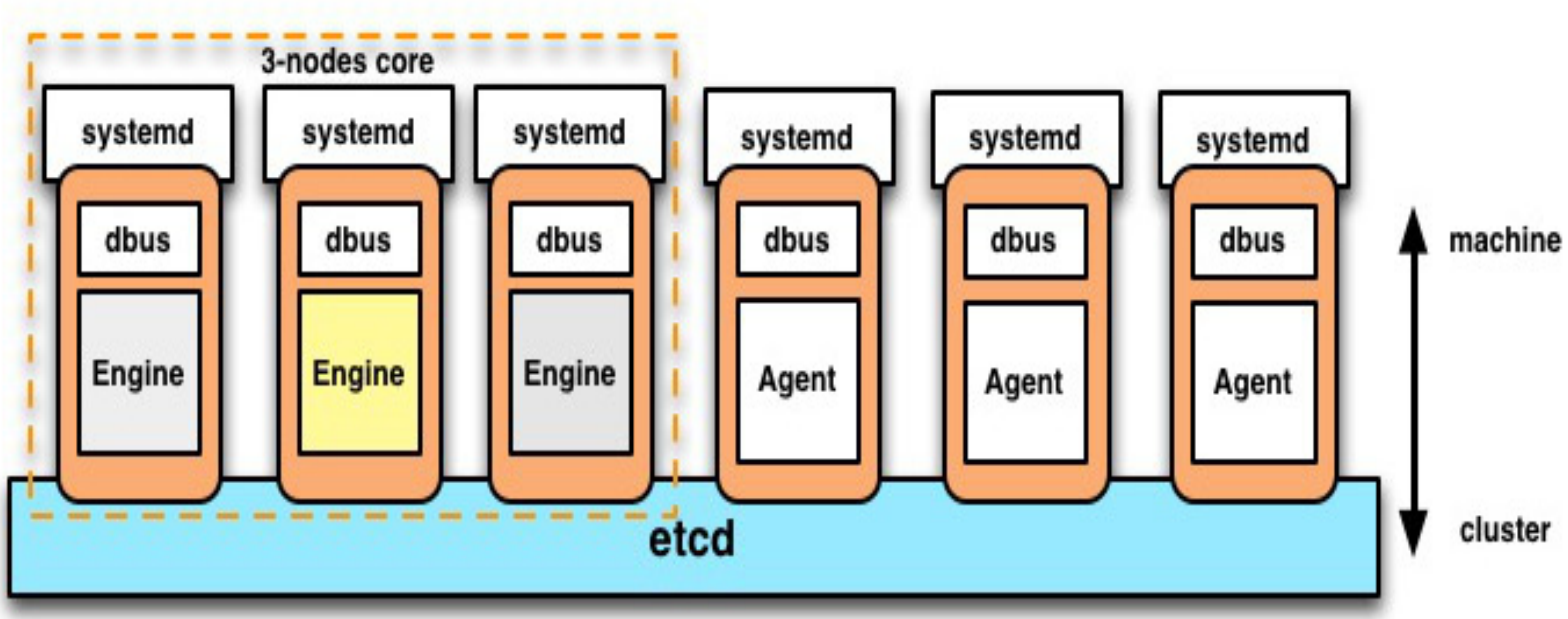


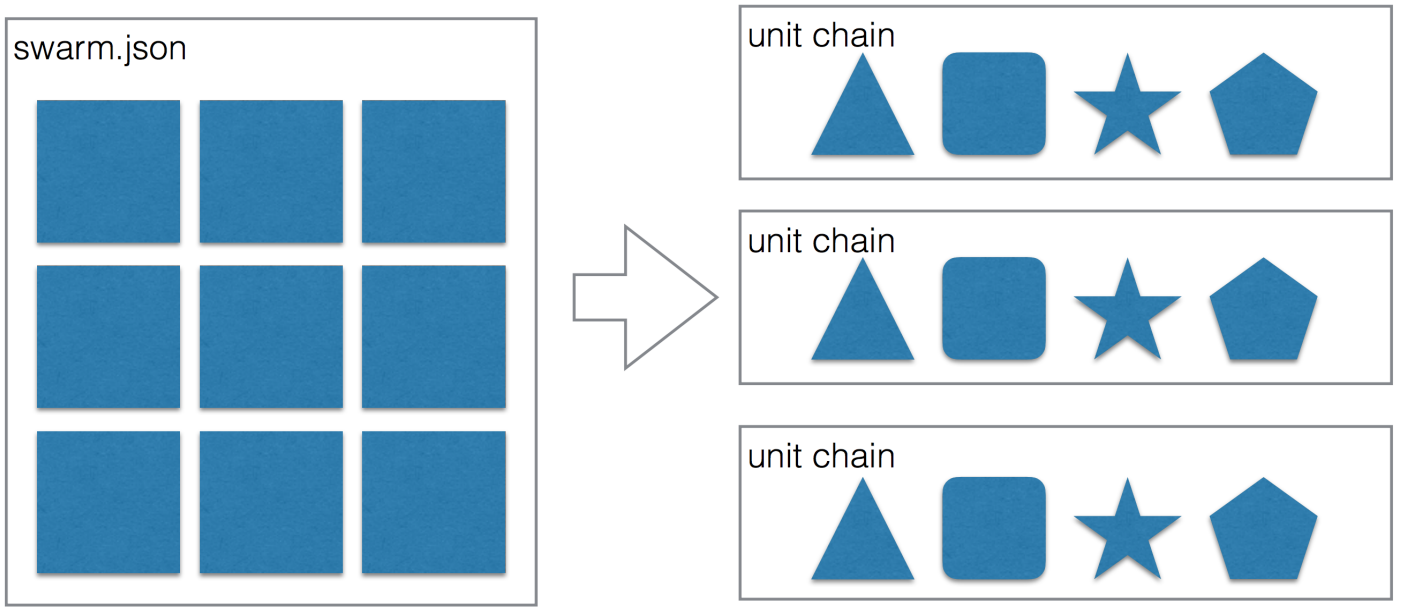
Figure: fleet: Engine and Agents

Giant Swarm

How do "we" use Fleet?

Our services are deployed as unit chains

- Service definition is done using a JSON file, *'swarm.json'*



Giant Swarm


HOWEVER :/ ...

Limitations of Fleet

- Performance degradation over the time
  - actions over units taking forever
  - *etcdctl set* command can take 40ms or 1100ms
- Fleet out of sync
  - unresponsive agents
  - incoherent fleet 'list-units'
- Limited scalability
- Excessive **etcd** dependencies: persistent and transient storage
  - *"If* **etcd** *is sad. fleet is sad. We are sad."*
- Fail recovery
  - If **etcd** goes down, everything falls apart

Giant Swarm


Why Fleet ?

Why Fleet ?

Giant Swarm intensively use **fleet** as init system of our infrastructure

Giant Swarm

Why Fleet ?

Giant Swarm intensively use **fleet** as init system of our infrastructure

What do we love ?

- Robustness
- Intuitive app definition
- Distributed systemd

Giant Swarm

Why Fleet ?

How do you manage your Kubernetes systemd units?

- Fleet could help you to control Kubernetes

```
UNIT                                MACHINE
k8s-master-api-server.service       0fc90277.../192.168.50.33
k8s-master-controller-manager.service 0fc90277.../192.168.50.33
k8s-master-scheduler.service        0fc90277.../192.168.50.33
k8s-network-setup-env.service       0fc90277.../192.168.50.33
k8s-network-setup-env.service       d077df29.../192.168.50.32
k8s-node-kubelet@348.service        d077df29.../192.168.50.32
k8s-node-kubelet@436.service        0fc90277.../192.168.50.33
k8s-node-proxy@348.service          d077df29.../192.168.50.32
k8s-node-proxy@436.service          0fc90277.../192.168.50.33
```

Giant Swarm

Why Fleet ?

How do you manage your Kubernetes systemd units?

- Fleet could help you to control Kubernetes

```
UNIT                                MACHINE
k8s-master-api-server.service       0fc90277.../192.168.50.33
k8s-master-controller-manager.service 0fc90277.../192.168.50.33
k8s-master-scheduler.service        0fc90277.../192.168.50.33
k8s-network-setup-env.service       0fc90277.../192.168.50.33
k8s-network-setup-env.service       d077df29.../192.168.50.32
k8s-node-kubelet@348.service        d077df29.../192.168.50.32
k8s-node-kubelet@436.service        0fc90277.../192.168.50.33
k8s-node-proxy@348.service          d077df29.../192.168.50.32
k8s-node-proxy@436.service          0fc90277.../192.168.50.33
```

How do you manage other INFRA services?

- Maybe Kubernetes is not what you want to use, e.g. cleanup tasks, monitoring

Giant Swarm

What's Next ?

What's next ?

We started investigated how to:

- Reduce the etcd dependencies
- Obtain better throughput (units/sec and amount of units)
- Improve fail recovery mechanisms
- Mitigate the scalability problems

Giant Swarm

What's next ?

We started investigated how to:

- Reduce the etcd dependencies
- Obtain better throughput (units/sec and amount of units)
- Improve fail recovery mechanisms
- Mitigate the scalability problems

Maybe we can use gRPC instead of etcd !

Giant Swarm

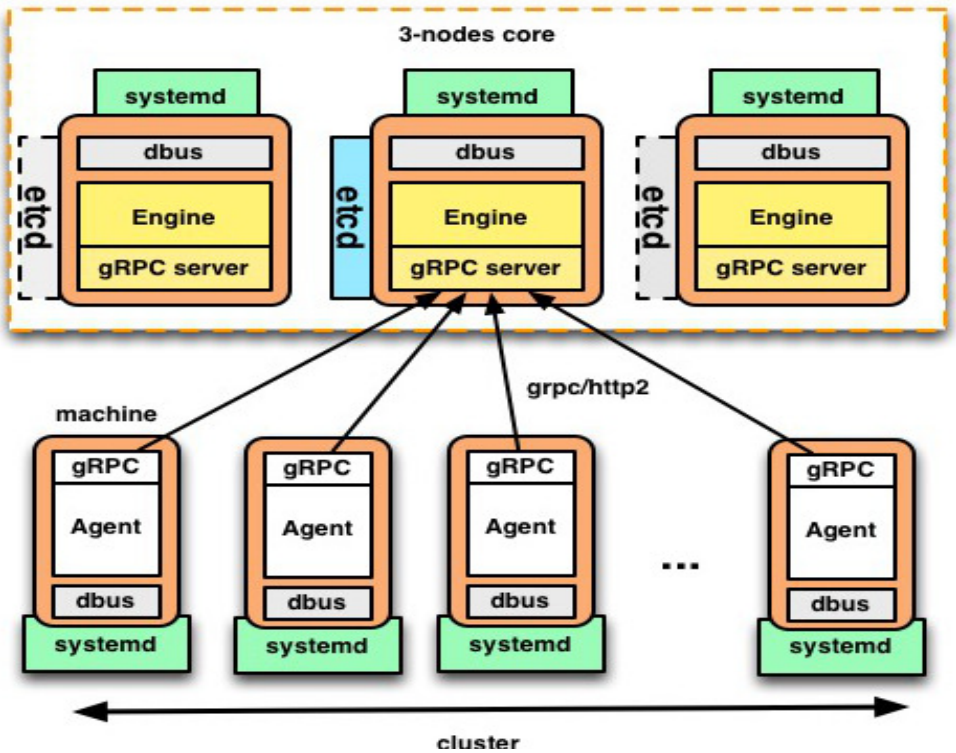
Fleet gRPC

- **gRPC** is the new communication layer between: Engine & Agents
- **etcd** is *ONLY* used for *PERSISTENT* storage
- *TRANSIENT* data is stored **in-memory**

Giant Swarm

Fleet gRPC

- **gRPC** is the new communication layer between: Engine & Agents
- **etcd** is *ONLY* used for *PERSISTENT* storage
- *TRANSIENT* data is stored **in-memory**



Giant Swarm

Fleet gRPC Benefits

Fail recovery:

- Connection retry
- Resistant to network hiccups
- Fallback mechanism (e.g. firewall)
- Agent healtchecks
- If **etcd** goes DOWN, we are still operative (on-going)

Giant Swarm

Fleet gRPC Benefits

Smooth Rolling Updates

- Engine/Agents detect if there are **etcd** nodes in the cluster
  - Tested in our production INFRA (hybrid fleet cluster)

Giant Swarm

Fleet gRPC Benefits

Smooth Rolling Updates

- Engine/Agents detect if there are **etcd** nodes in the cluster
  - Tested in our production INFRA (hybrid fleet cluster)

Other benefits:

- Hybrid etcd/gRPC cluster (etcd and grpc nodes)
- New leader election algorithm
- TLS communication

Giant Swarm

Let's validate those benefits!

Experimental Validation

Configuration:

- CoreOS cluster with 3 machines as master/engine and 4 workers/agents
- systemd v219
- etcd v2.1.2
- fleet v0.11.7
- nomi v0.1.0

Tested versions:

- Scenario 1: 3k units **fleet-etcd** vs **fleet-grpc**
- Scenario 2: 20k units with **fleet-grpc**
- Scenario 3: 1k units **fleet-etcd** with **etcdv2.2** vs **etcdv3**

Giant Swarm

systemd vs fleet

How much time does it take to create 500 systemd units in 1 node?

Giant Swarm

systemd vs fleet

How much time does it take to create 500 systemd units in 1 node?

- With **fleet-grpc**: 58.8 secs
- With **fleet-etcd (current upstream version)**: 74.4 secs
- With **systemd**: 162 secs (sequential)

Giant Swarm

systemd vs fleet

How much time does it take to create 500 systemd units in 1 node?

Why ?

- **'systemd linking'** operation is very EXPENSIVE

Giant Swarm

Fleet gRPC - Throughput

How do they perform to **load/start 3k units** ?

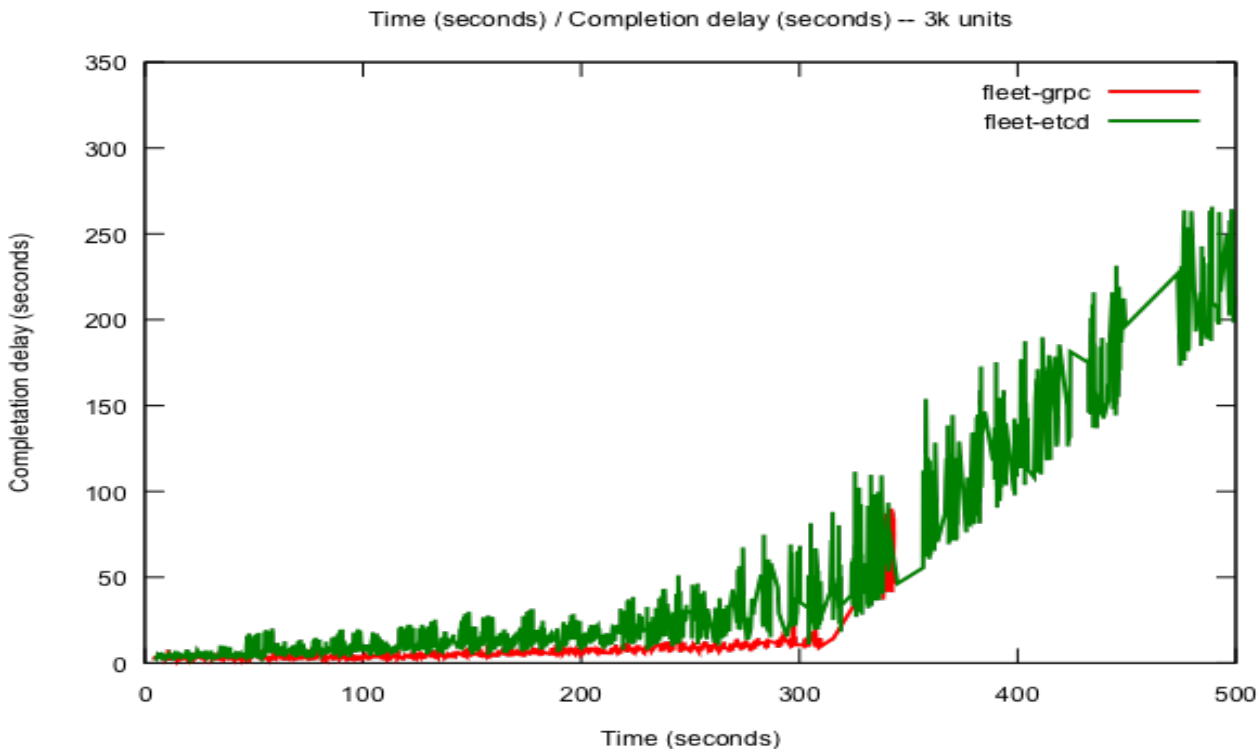
- **fleet-etcd**:
  - 75 % of units started in between 1.729 - 28.14 secs
  - Max: 265.88 secs
- **fleet-grpc**:
  - 85 % of units started in between 1.004 - 9.907 secs
  - Max: 90.0 secs

Giant Swarm

Fleet gRPC - Throughput

How do they perform to **load/start 3k units** ?

- **fleet-etcd**: 75 % of units started in between 1.729 - 28.14 secs
- **fleet-grpc**: 85 % of units started in between 1.004 - 9.907 secs



Giant Swarm

Fleet gRPC - Threshold

How many units can we handle in the cluster ?

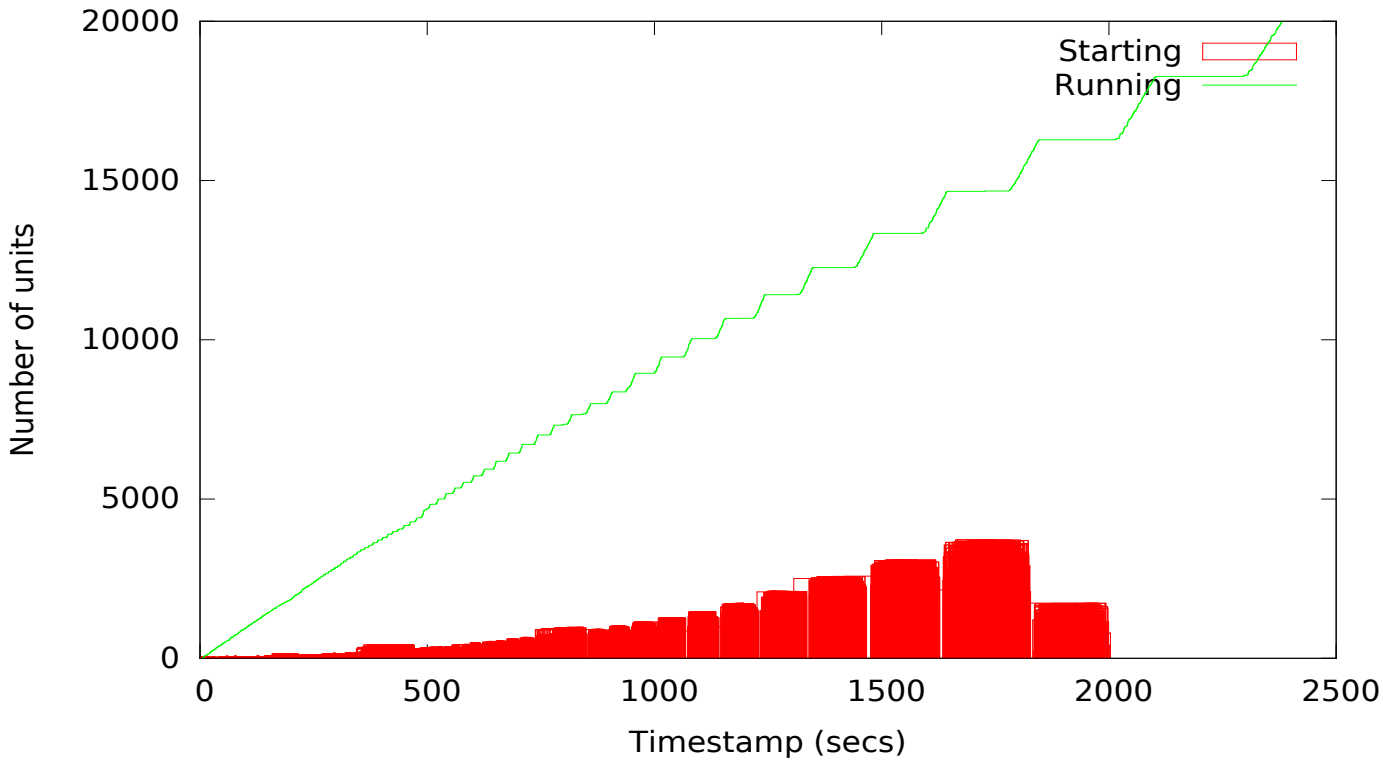
- **fleet-etcd**: no more than 3k
- **fleet-grpc**: up to 20k and beyond..

Giant Swarm

Fleet gRPC - Threshold

How many units can we handle in the cluster ?

- **fleet-etcd**: no more than 3k
- **fleet-grpc**: up to 20k and beyond..



Giant Swarm

Fleet gRPC - CPU usage

CPU usage of **fleet** & **etcd** when using **fleet-etcd** ?

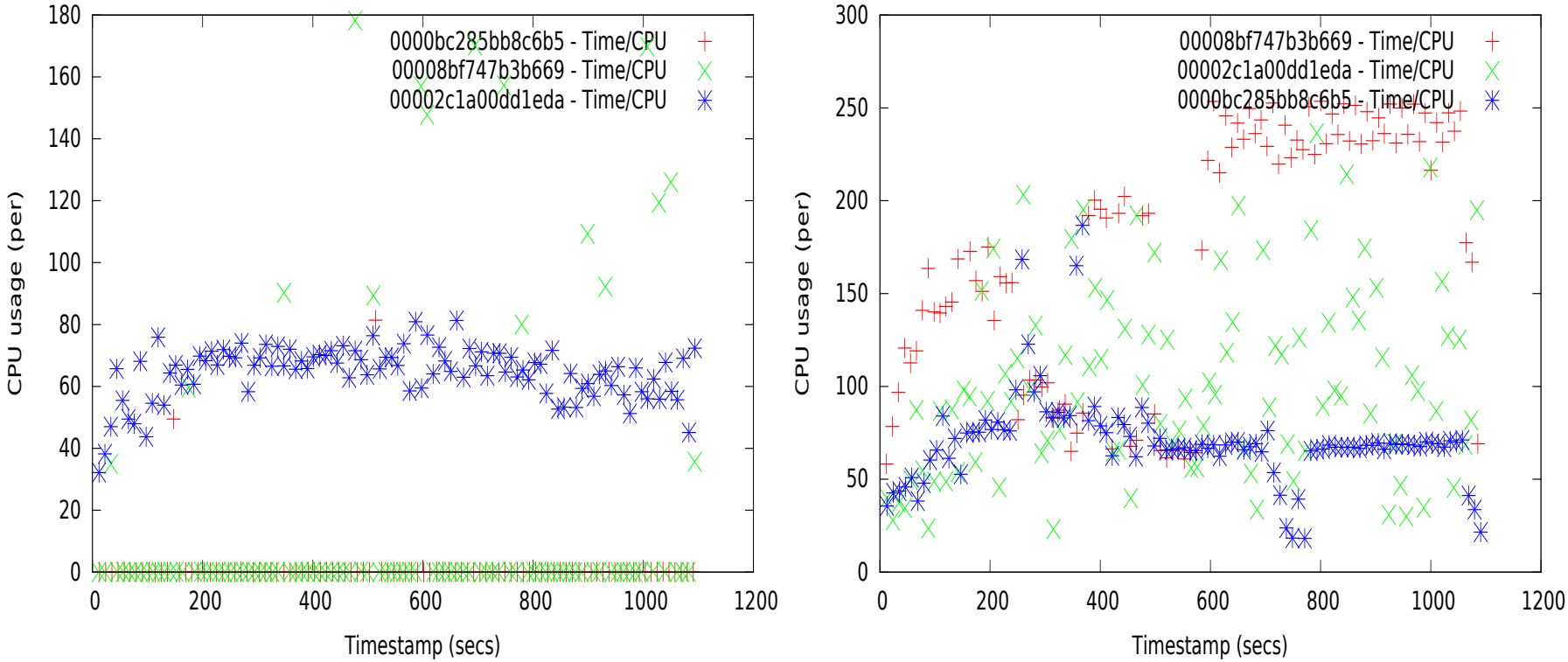


Figure: etcd / fleet – CPU usage

Giant Swarm

Fleet gRPC - CPU usage

CPU usage of **fleet** & **etcd** with the **fleet-gRPC** ?

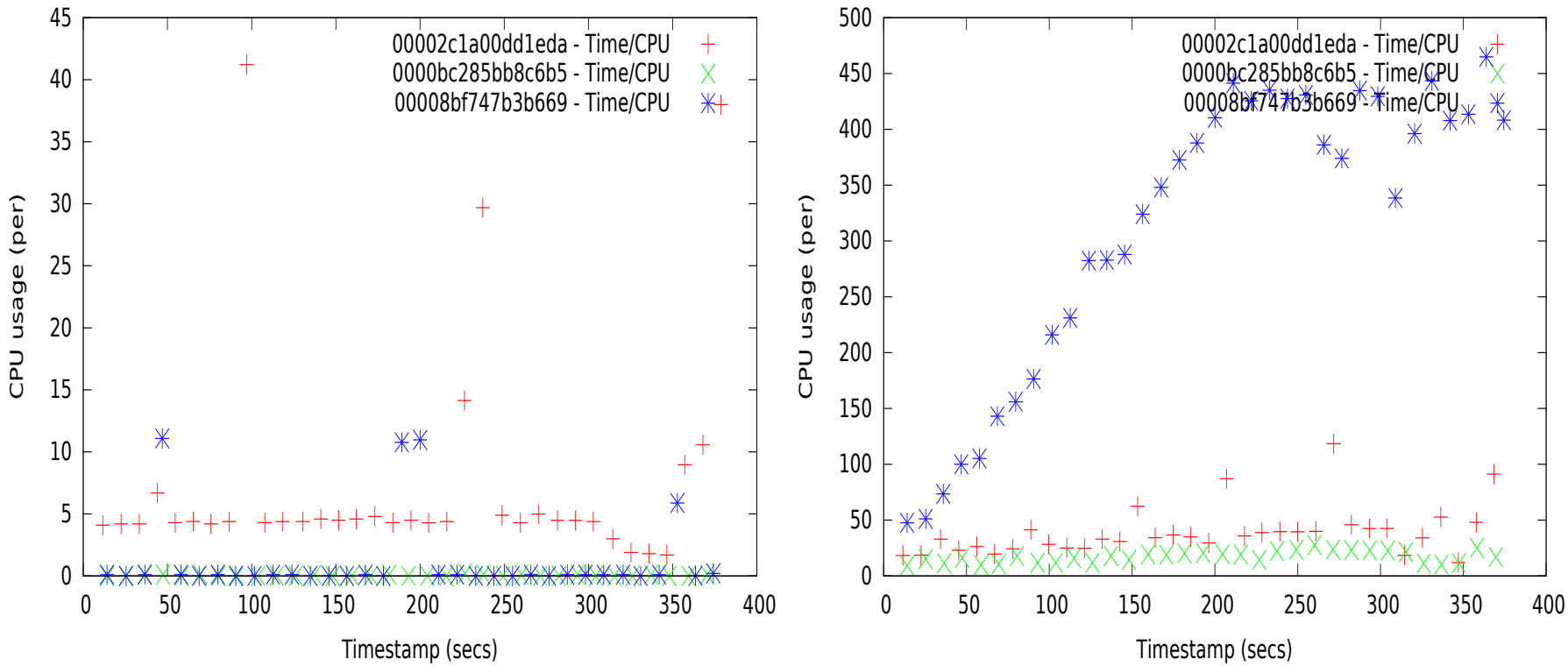


Figure: etcd / fleet – CPU usage

Giant Swarm

Fleet-gRPC - What about fleet-etcdv3 ?

Experiment:

- Evaluate the **DELAY** when starting (load/start) 1k units

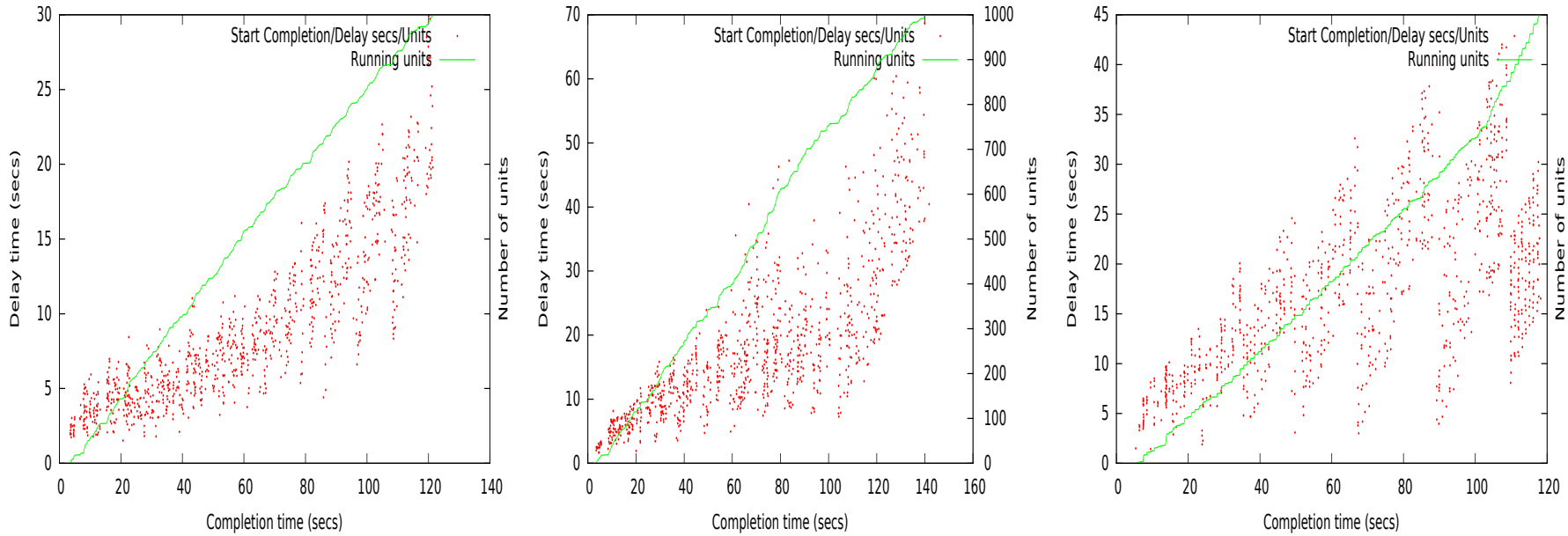


Figure: fleet-etcd **v3 experimental** / fleet-etcd **v2.3** / fleet-grpc with etcdv2.2

Giant Swarm

Conclusion

Fleet with gRPC offers:

- Better throughput
- Less etcd friction
- Lower CPU usage in the fleet agents
- Secure communication
- Improved scalability
- No need of frequent cleanup tasks

Giant Swarm

Future Improvements

We plan to:

- Use **etcdv3**
- Add TLS to secure the communication
- Improve the unit states transmission between agents-engine, (send what it has changed)
- Change the reconciliation loop mechanism
- Look at the usage of **systemd –user** to gain in performance
  - **systemd linking** is a quite expensive operation.

Giant Swarm

Future Improvements

On-going work (PRs/Issues)

- Our fleet repo ‘github.com/giantswarm/fleet‘
- Our benchmarking tool ‘github.com/giantswarm/nomi‘

Issues

- #1426
- #1418
- #3142
- #150
- #3182
- #1393

Giant Swarm


WE ARE HIRING !


THANKS ! Time for ...