ORCHESTRATING QUASI-REAL TIME DATA PROCESSING IN THE COMPUTING FARM OF THE ATLAS EXPERIMENT AT CERN

GIUSEPPE AVOLIO – CERN

OUTLINE

The ATLAS detector at the Large Hadron Collider (LHC)

Data volumes and rates

The Event Filtering (EF) online computing farm

Motivations and challenges

A Kubernetes journey

The departure

The adventure

The reward

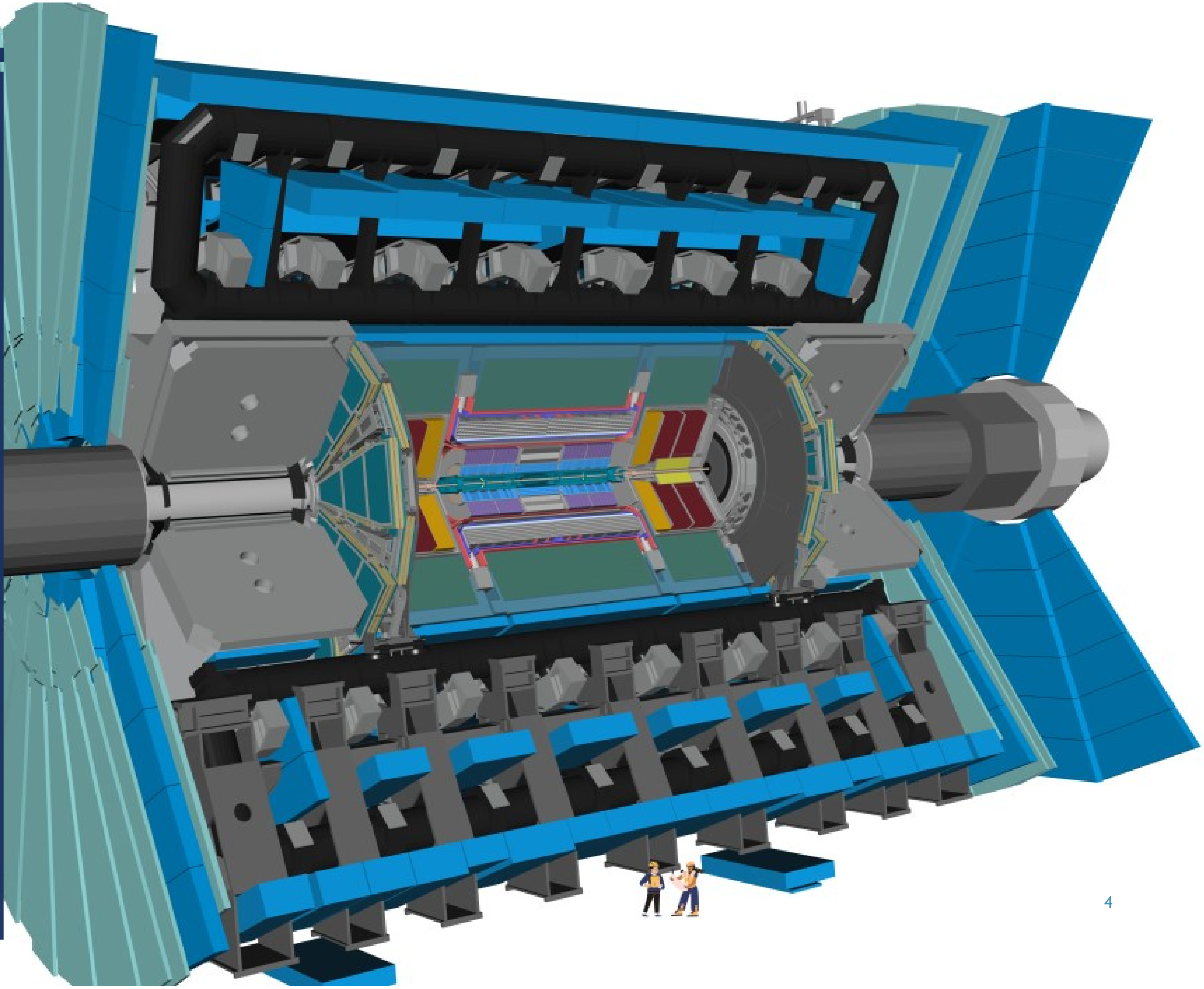
Conclusions and outlook

A glimpse of the future

LHC
ATLAS
FILTERING COMPUTING FARM

ATLAS

- **One of the four main particle detectors at the Large Hadron Collider (LHC)**
  - 44 m length
  - 25 m height
- **Collect data from proton collisions provided by the LHC**
  - 40 MHz collision rate
  - 14 TeV proton-proton energy

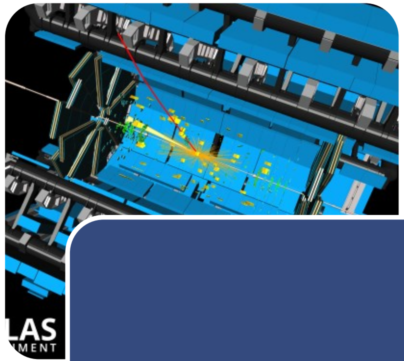


ATLAS – ONLINE

FROM COLLISIONS TO STORAGE (NOW)

	40 MHz	100 kHz	3 kHz
Event rate	40 MHz	100 kHz	3 kHz
Data throughput		300 GB/s	8 GB/s

LHC



ATLAS

Event Filtering Farm

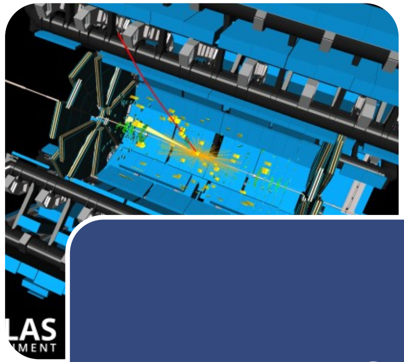
Storage

ATLAS – ONLINE

FROM COLLISIONS TO STORAGE (NOW AND **AFTER THE LHC UPGRADE: HIGH-LUMINOSITY LHC**)

Event rate	40 MHz	100 kHz (**1 MHz**)	3 kHz (**10 kHz**)
Data throughput		300 GB/s (**5 TB/s**)	8 GB/s (**50 GB/s**)

HL-LHC



ATLAS

Event Filtering Farm

Storage

ATLAS – ONLINE

THE EVENT FILTER COMPUTING FARM

Size

- About ~2000 servers (~60k CPU cores) today
- Up to 5000 servers (~500k CPU cores) for the LHC upgrade

High data volume, low latency

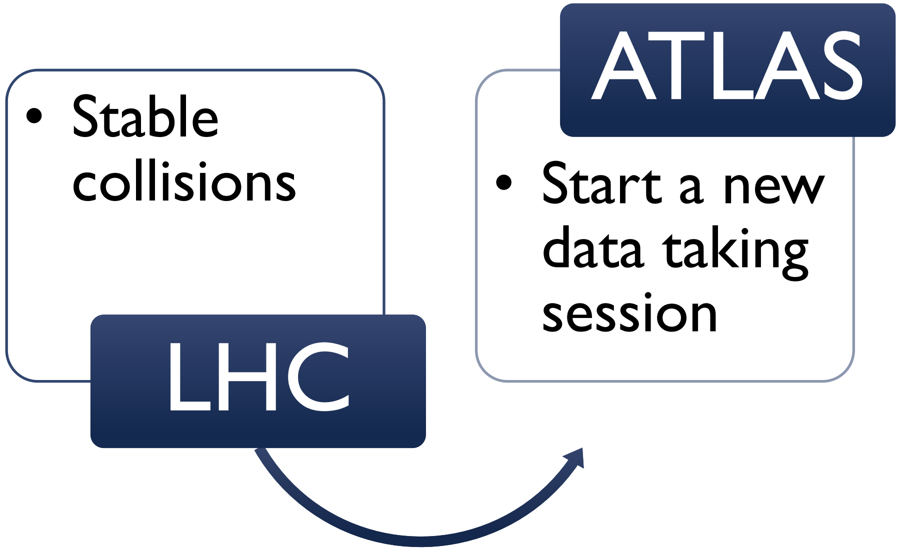
- The time budget to filter a single event is around one second

High number of processing applications

- Up to 65k for the LHC upgrade

Readiness

- Strict control on the application startup/stop times
- Not a start-and-forget scenario



ATLAS – ONLINE

ORCHESTRATING THE EVENT FILTER COMPUTING FARM

In-house custom system

- Process control
- Scheduling
- Error detection and management
- Bare processes
  - No containers

Kubernetes

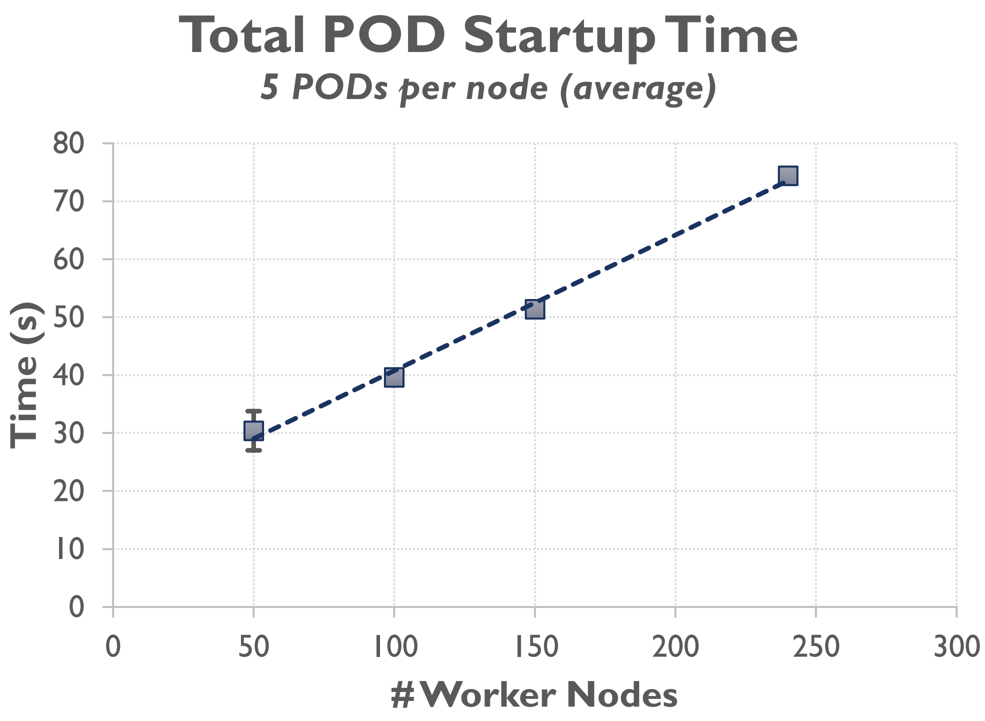
- Simplified operations and maintenance
- More dynamic scheduling
- Enhanced high availability and fault tolerance
- Exploit containerization

A KUBERNETES JOURNEY

THE DEPARTURE

2018: THE KUBERNETES JOURNEY STARTS

PRELIMINARY TESTS



Goal

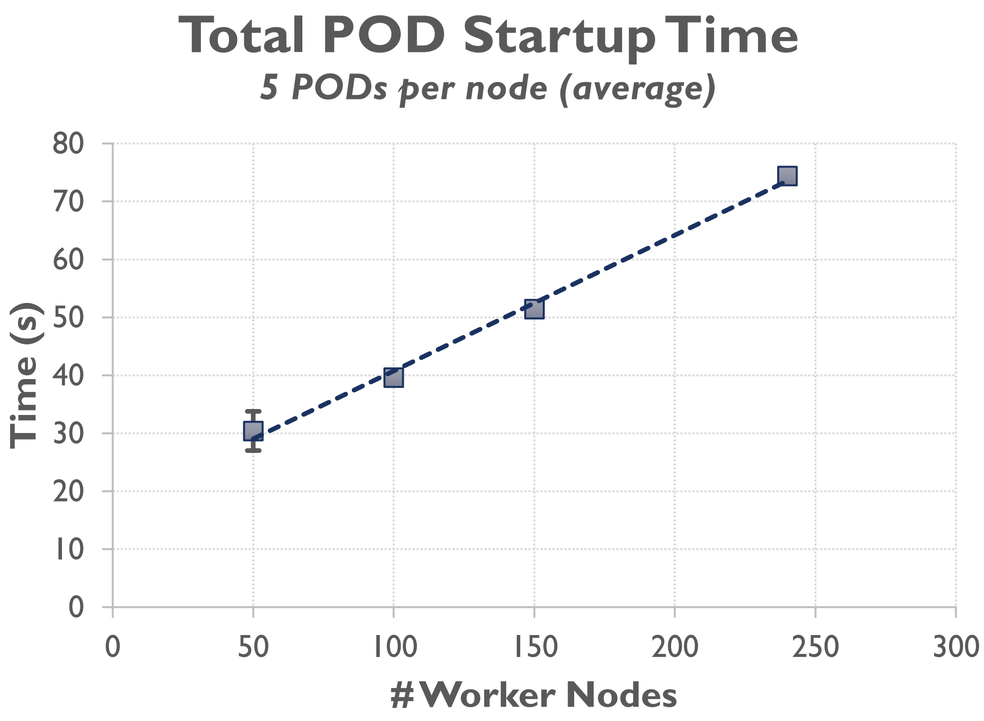
- How fast can we start PODs?
- What about the scaling with the number of working nodes?

System Under Test

- Kubernetes version **1.5**
- **Single** control plane node
- Up to **240 working nodes**
- Average of **5 PODs per node**
  - Single *pause* container (pre-loaded)
- All running in **Virtual Machines**

2018: THE KUBERNETES JOURNEY STARTS

PRELIMINARY TESTS

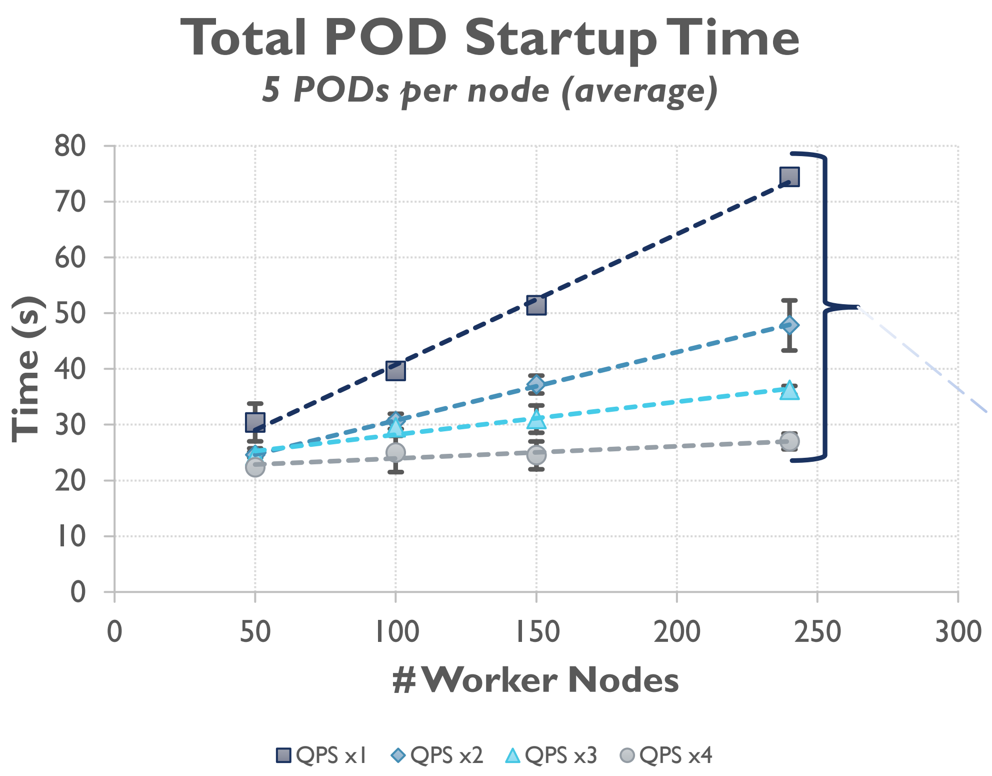


Good linear scaling

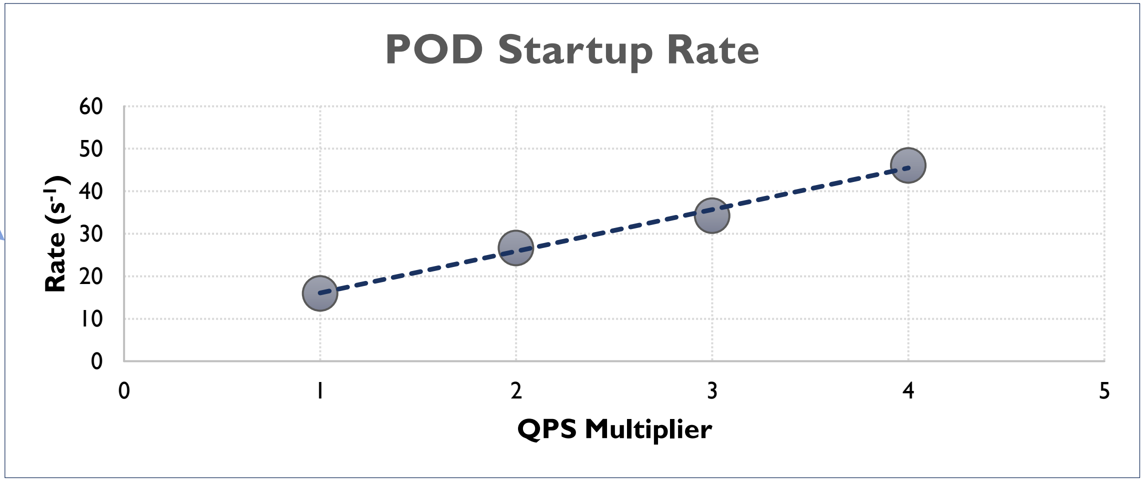
More than 70 seconds for 1200 PODs

2018: THE KUBERNETES JOURNEY STARTS

PRELIMINARY TESTS – QPS TO THE RESCUE



Component	QPS Parameters (def. values)
kube-controller-manager	kube-api-qps (20) kube-api-burst (30)
kube-scheduler	kube-api-qps (50) kube-api-burst (100)



2018: THE KUBERNETES JOURNEY STARTS

PRELIMINARY TESTS – EXECUTIVE SUMMARY

General behaviour

- Linear scaling
- Reproducibility
- Performance improvements with QPS tuning

Performance

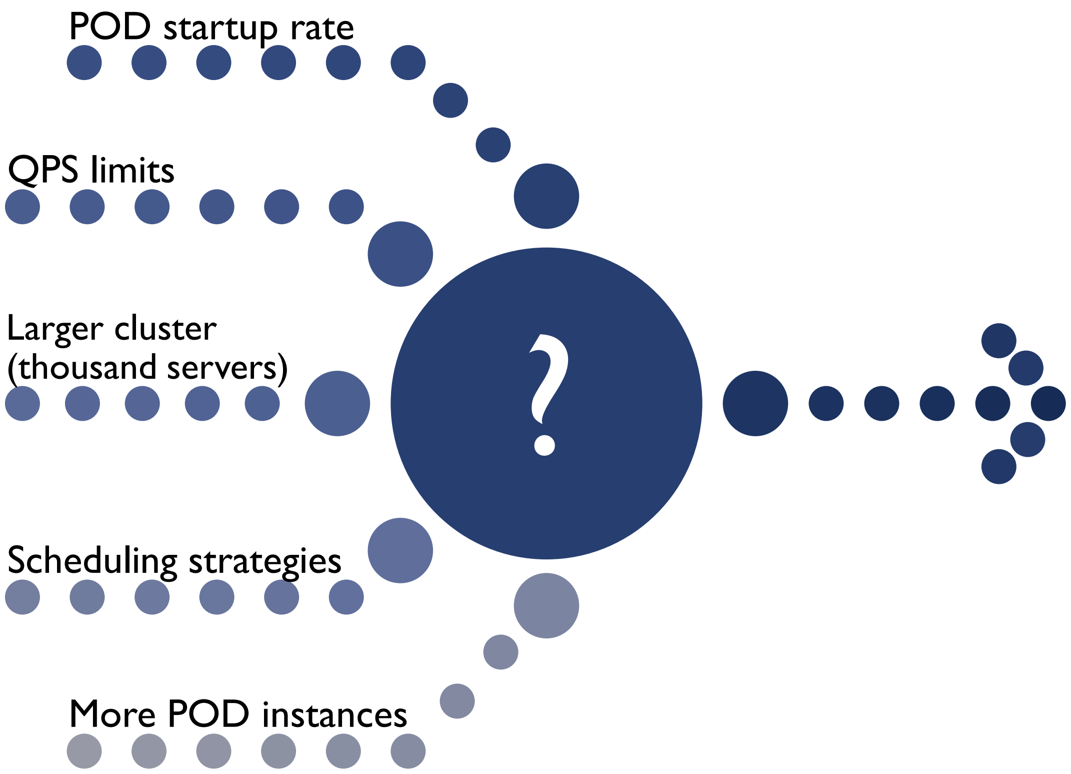
- POD startup rate too low

A KUBERNETES JOURNEY

THE ADVENTURE

THE KUBERNETES JOURNEY CONTINUES

A LOT OF QUESTIONS TO ANSWER AND A STRATEGY



Systematic tests using the currently available computing farm at the ATLAS experimental site

THE KUBERNETES JOURNEY CONTINUES

TESTING SETUP

Kubernetes Cluster

- 4 control plane servers
- 4 dedicated ETCD servers (one used to send *events*)
- Up to about 2600 worker nodes (bare metal)

Testing Plan

- Study POD startup/stop time scaling (*pause* container)
- Realistic scheduling
- Exploit *clusterloader2*
- Look at all the steps in the POD **startup phases**

Create

Schedule

Run

Watch

POD STARTUP/STOP TIMES

KUBE-API-QPS AND KUBE-API-BURST: SCANNING

10k PODs across 2000 worker nodes
(HostNetwork, Deployment, kube-api-qps = kube-api-burst, K8s version 1.21)

Startup Time (s)		Scheduler (QPS)		
		200	*400*	*800*
Controller Manager (QPS)	*200*	52	52	52
	400	52	32	31
	800	52	31	32

Stop Time (s)		Scheduler (QPS)		
		200	*400*	*800*
Controller Manager (QPS)	*200*	115	114	115
	400	68	63	67
	800	42	42	42

Startup Time

Changes in Scheduler QPS only relevant for a Controller Manager QPS higher than 400	For a Scheduler QPS of 200 no gain in changing the Controller Manager QPS

Stop Time

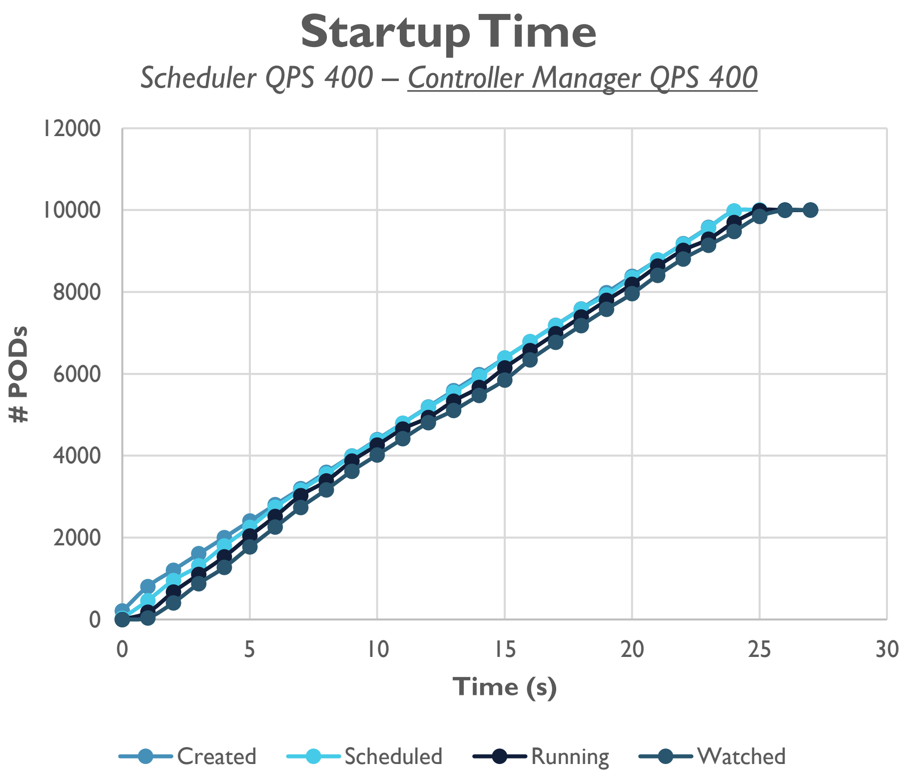
Independent of the Scheduler QPS	Better with higher QPS for the Controller Manager

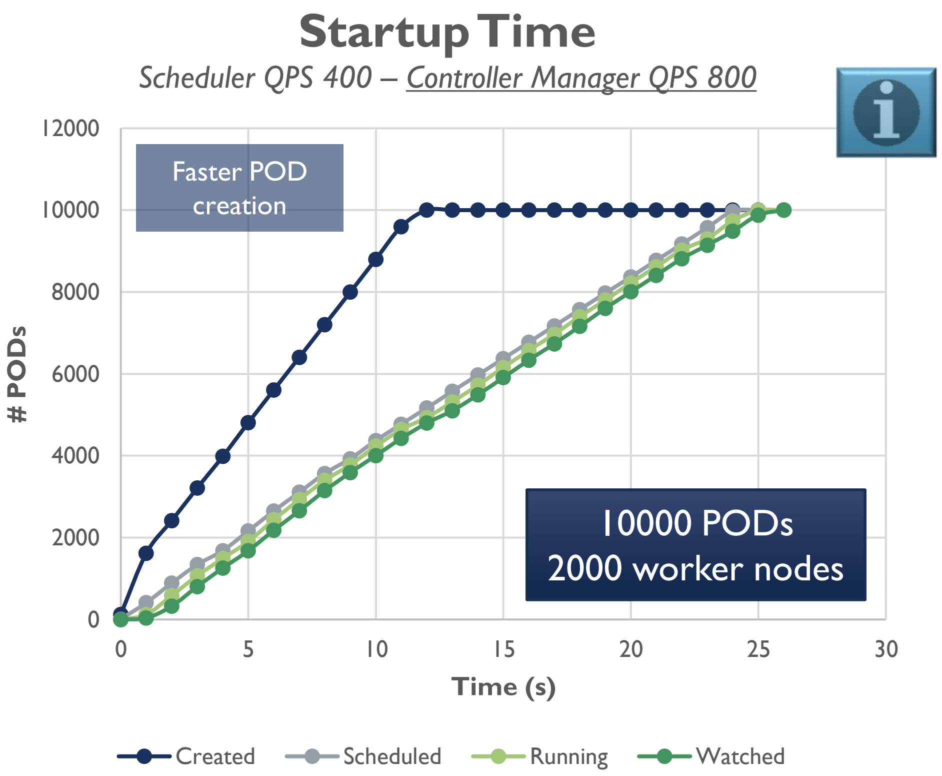
Set the Working Point

Scheduler QPS: 800	**Controller Manager QPS: 800**

POD STARTUP TIMES

KUBE-API-QPS AND KUBE-API-BURST: STARTUP PHASES ANALYSIS





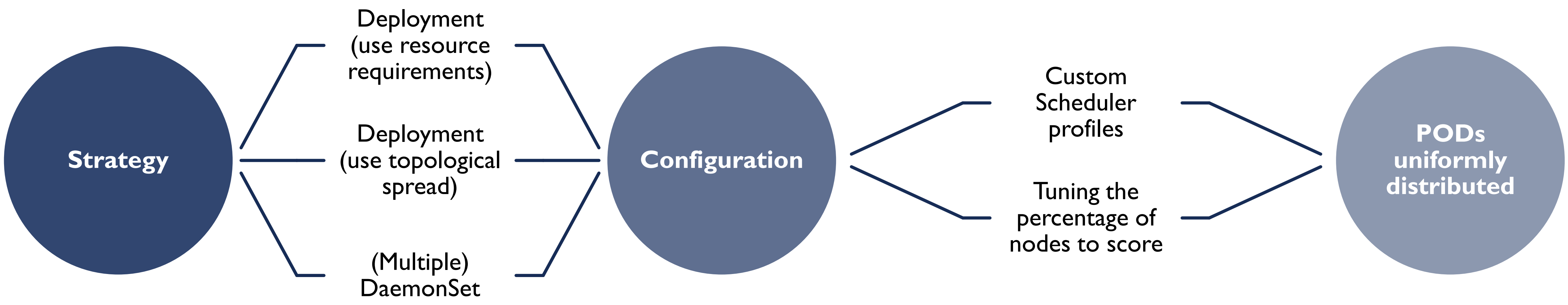
LESSONS LEARNT

1. Some QPS tuning is needed in large clusters to improve POD startup and stop times

2. Scheduling time may be a bottleneck for large deployments

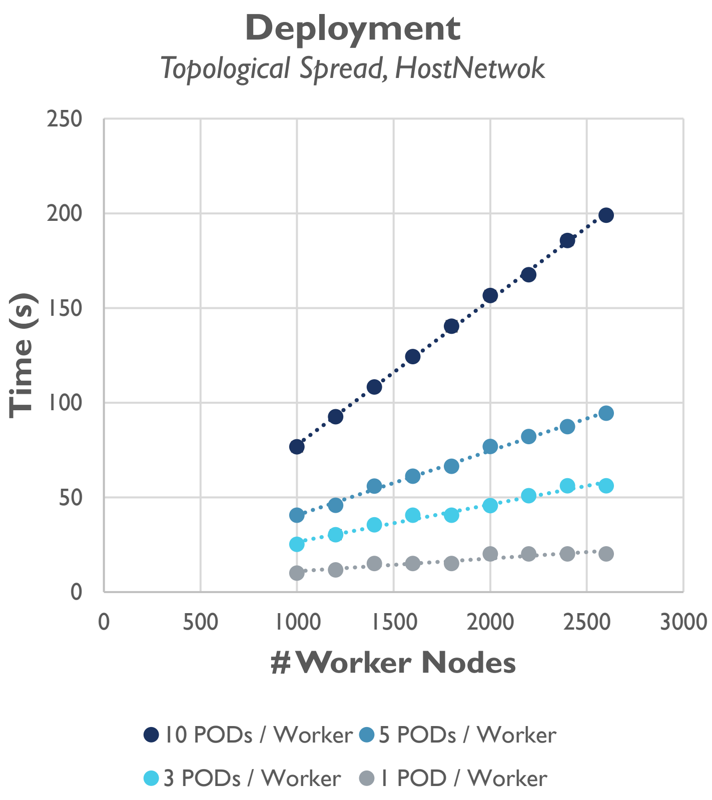
POD STARTUP TIMES

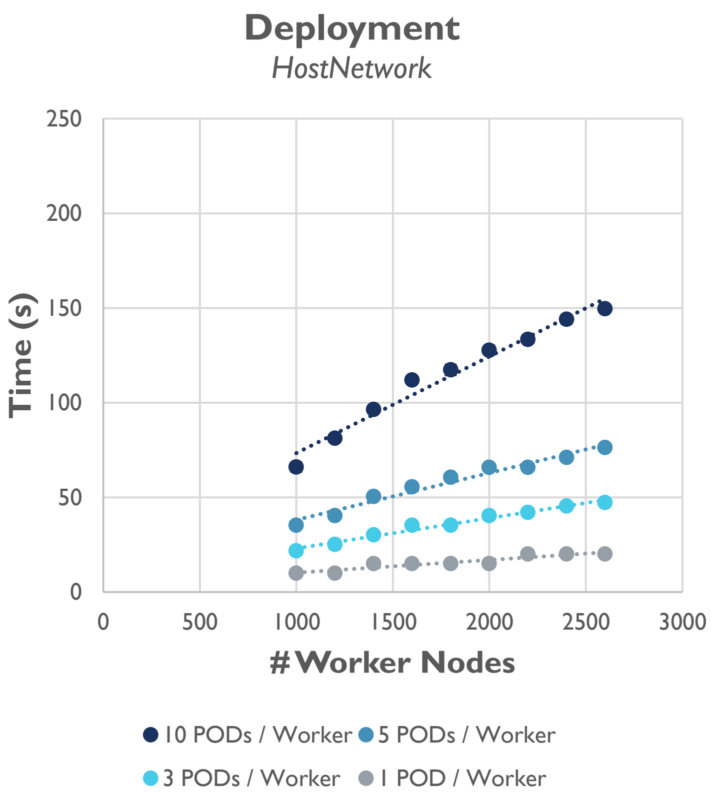
SCHEDULING MATTERS

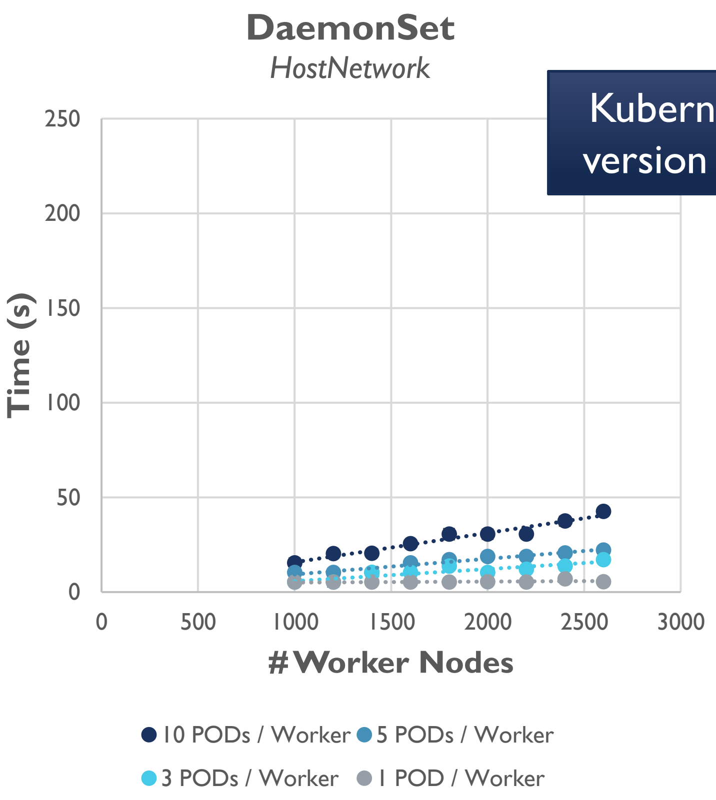


POD STARTUP TIMES

SCHEDULING STRATEGIES – DEFAULT SCHEDULER PROFILE



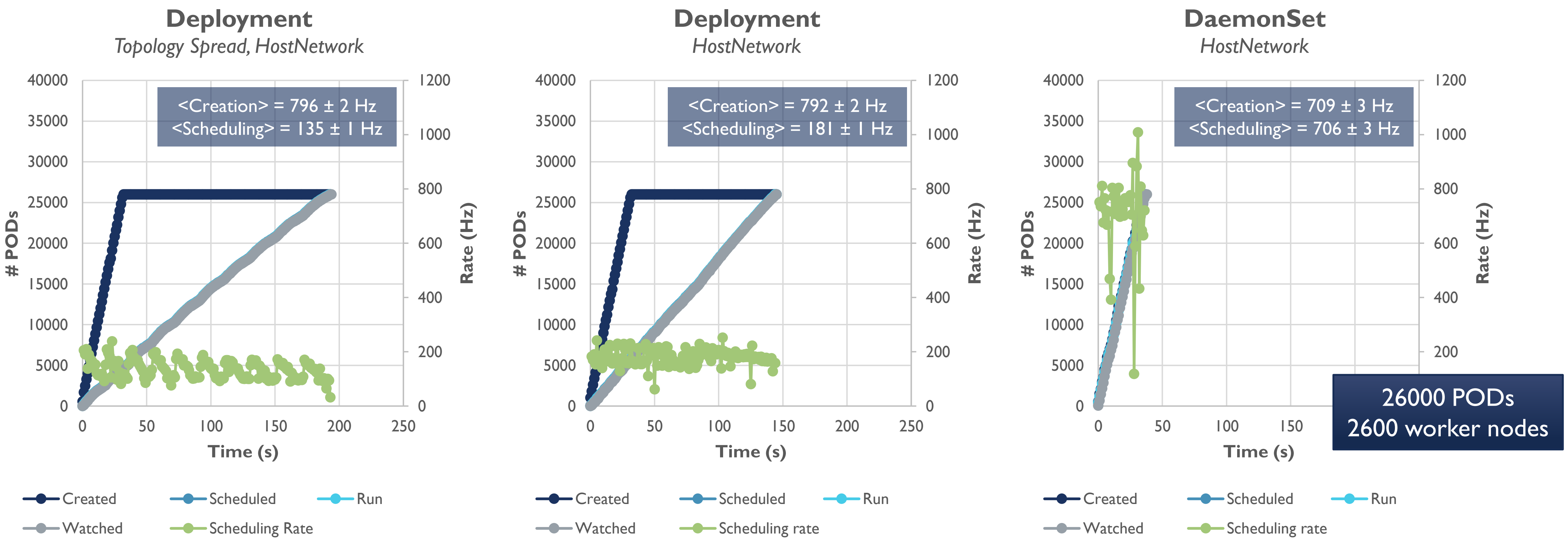




Kubernetes version 1.28

POD STARTUP TIMES

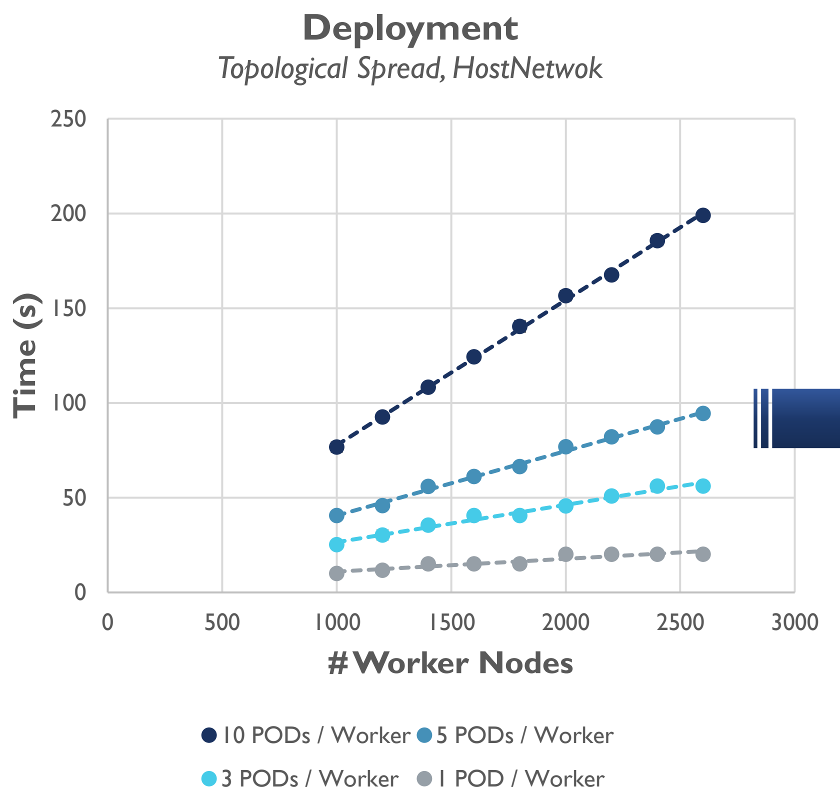
SCHEDULING STRATEGIES – DEFAULT SCHEDULER PROFILE – STARTUP PHASES

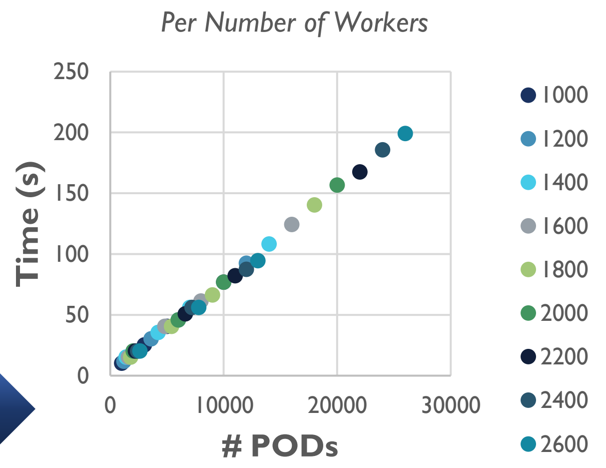


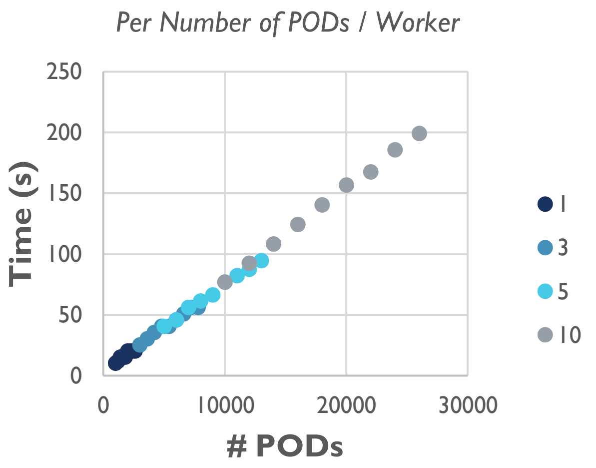
26000 PODs
2600 worker nodes

POD STARTUP TIMES

DEPENDENCY ON THE NUMBER OF PODS







Average POD Startup Rate	
Deployment (Topological)	133 ± 1 Hz
Deployment	167 ± 3 Hz
(Multiple) DaemonSet	680 ± 20 Hz

LESSONS LEARNT

1. Some QPS tuning is needed in large clusters to improve POD startup and stop times

2. Scheduling time may be a bottleneck for large deployments

3. Scheduling strategies do impact the POD startup times

4. Global POD startup times seem to depend on the total number of PODs only

POD STARTUP TIMES

CUSTOM SCHEDULER PROFILE

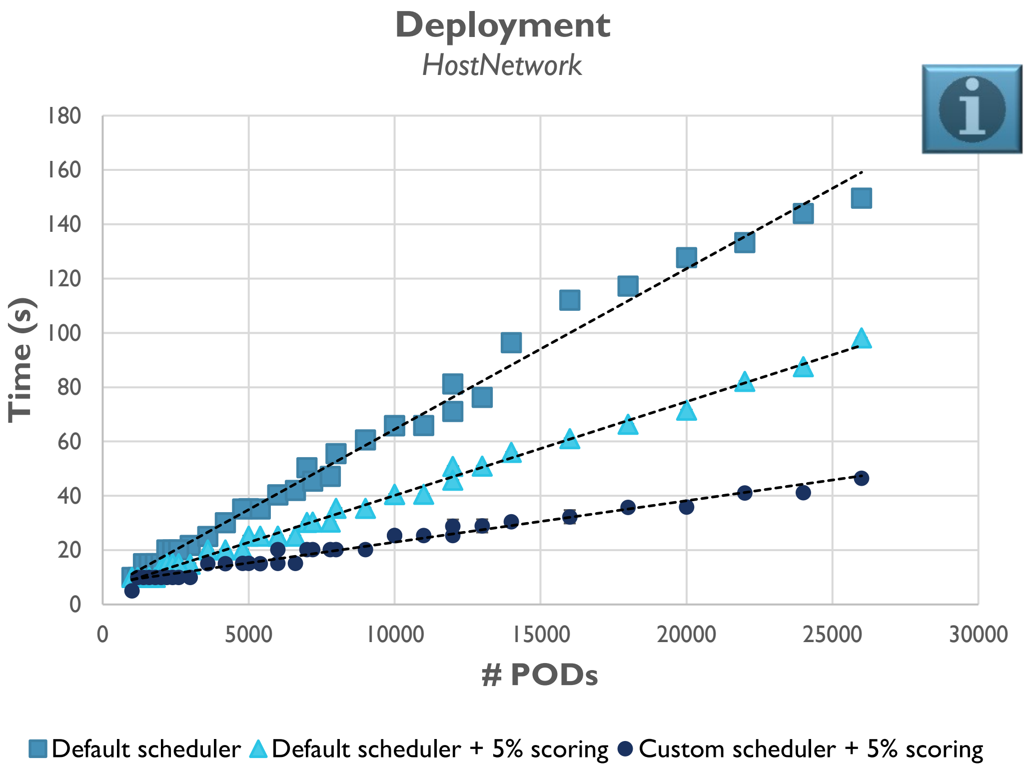
no-scoring-scheduler

- Disable all the plug-ins in the *preScore* and *score* phases
- Optionally, tune the *percentageOfNodesToScore*

```yaml
- schedulerName: no-scoring-scheduler
  percentageOfNodesToScore: 5
  plugins:
      preScore:
          disabled:
          - name: '*'
      score:
          disabled:
           - name: '*'
```

POD STARTUP TIMES

CUSTOM SCHEDULER PROFILE



Time to Start 26000 PODs		
	Default Scheduler	**Custom Scheduler + 5% Scoring**
Deployment (Topological)	199 ± 3 s	122 ± 1 s
Deployment	150 ± 1 s	47 ± 1 s
(Multiple) DaemonSet	43 ± 2 s	44 ± 2 s

Average POD Startup Rate			
	Default Scheduler	**Custom Scheduler + 5% Scoring**	**Throughput Increase**
Deployment (Topological)	133 ± 1 Hz	226 ± 3 Hz	70%
Deployment	167 ± 3 Hz	640 ± 20 Hz	280%

POD STARTUP TIMES

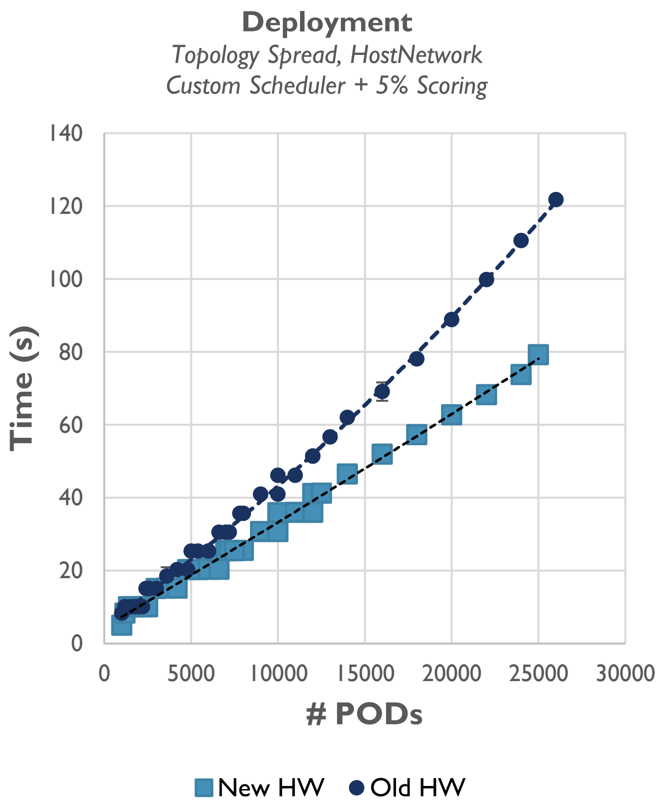
WHAT ABOUT THE CONTROL PLANE HARDWARE?

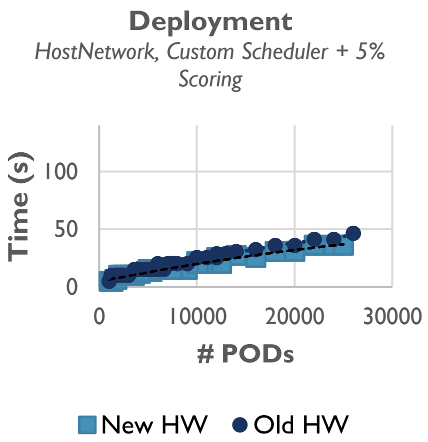
Old Control Plane HW

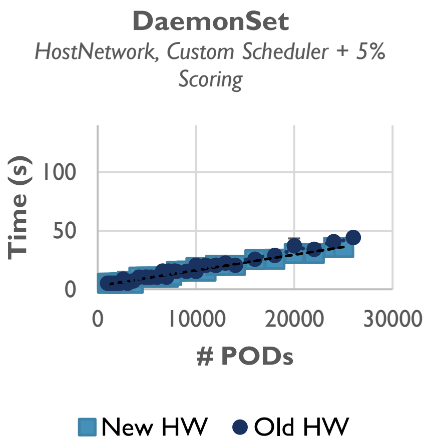
- 4 servers
- 2 x Intel Xeon E5-2620 v3 6C/12T
- 1 Gbps connectivity

New Control Plane HW

- 4 servers
- 2 x AMD EPYC 7313 16C/32T
- 10 Gbps connectivity
- Same ETCD cluster





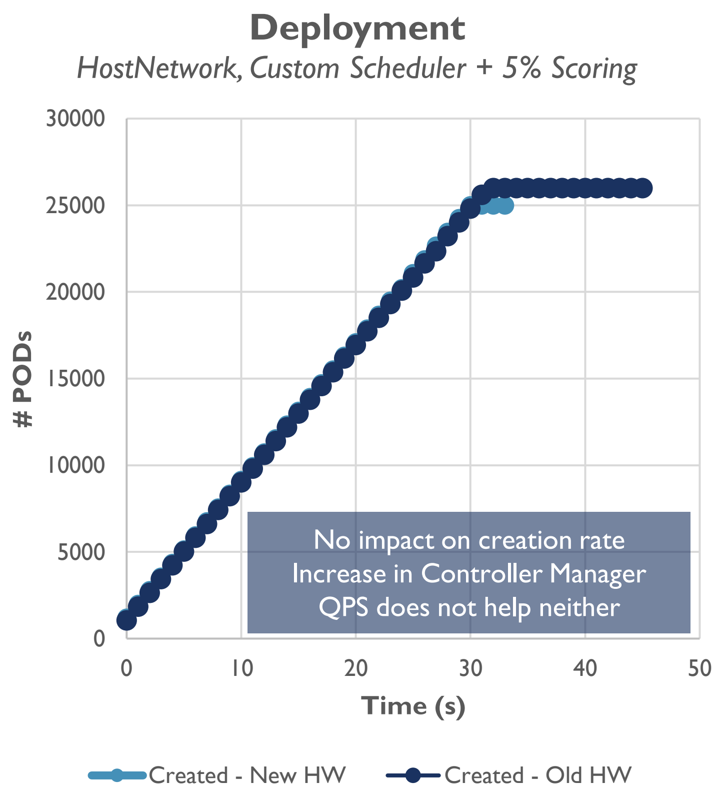


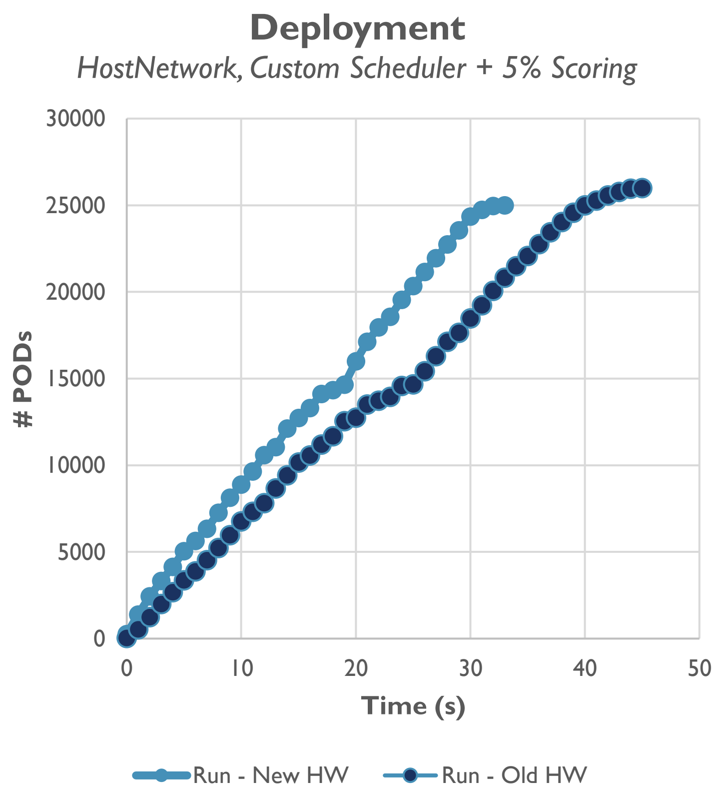
Average POD Startup Rate

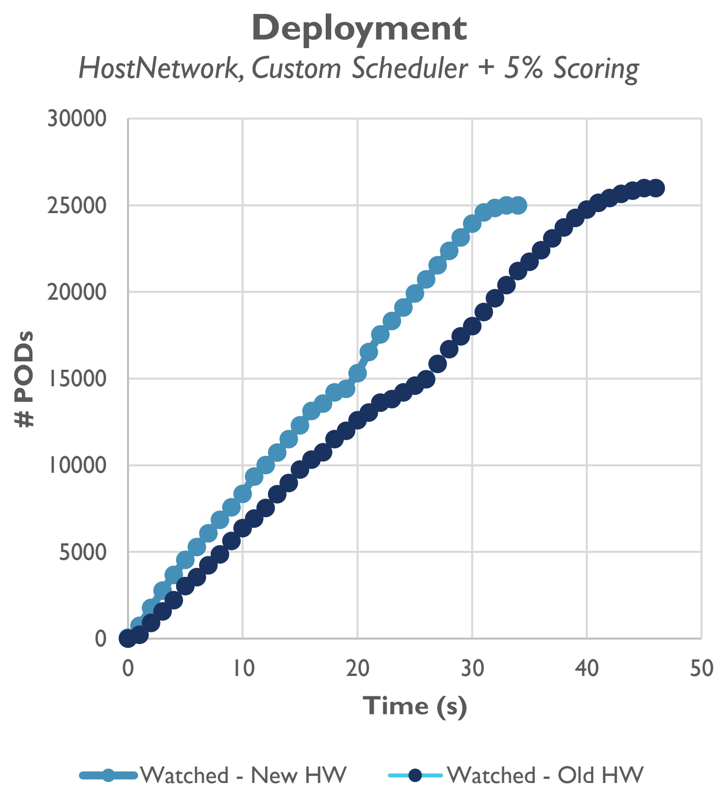
	Old HW	New HW	Throughput Increase
Deployment (Topological)	226 ± 3 Hz	340 ± 4 Hz	50%
Deployment	640 ± 20 Hz	740 ± 20 Hz	16%
(Multiple) DaemonSet	630 ± 20 Hz	730 ± 20 Hz	16%

POD STARTUP TIMES

WHAT ABOUT THE CONTROL PLANE HARDWARE? STARTUP PHASES

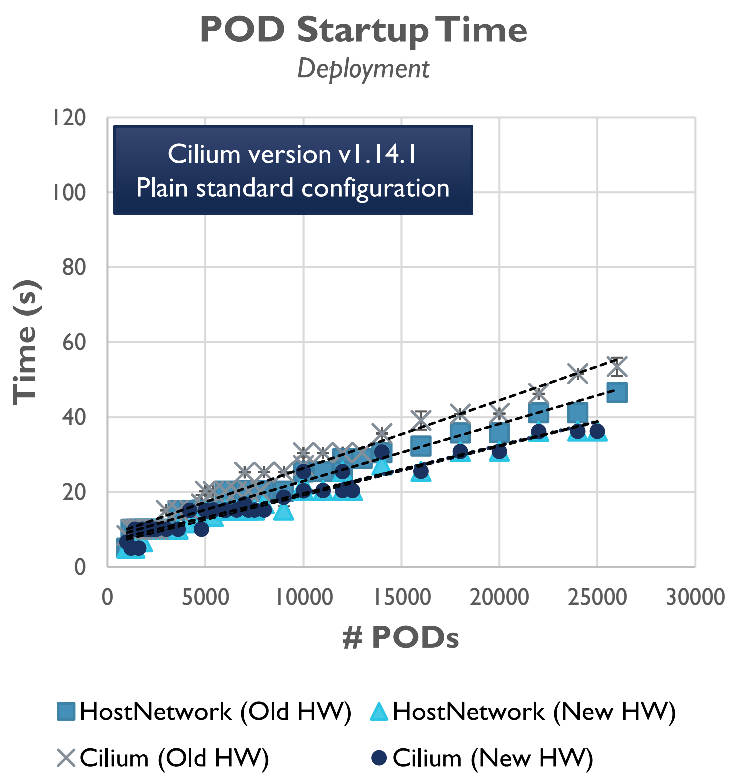


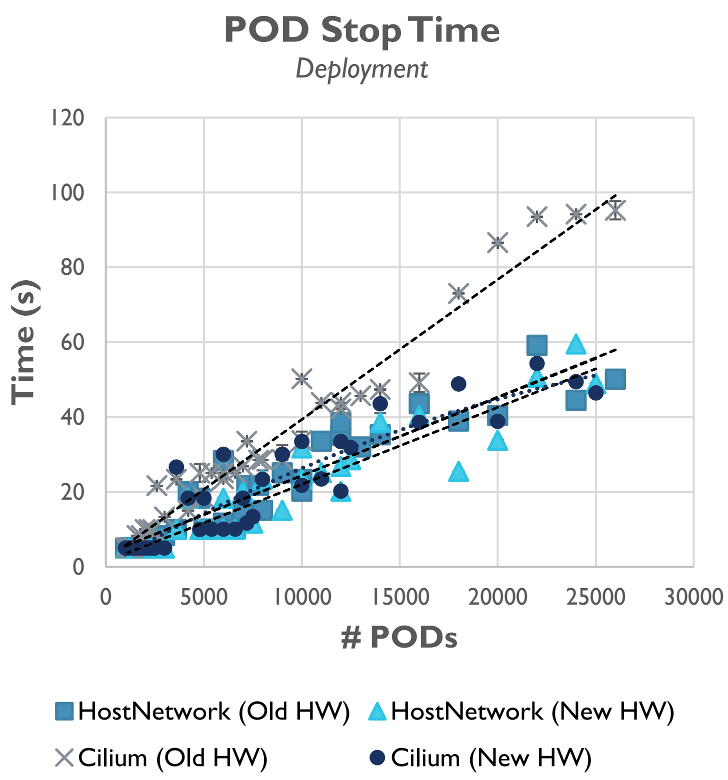




POD STARTUP (AND STOP) TIMES

WHAT ABOUT THE CNI?





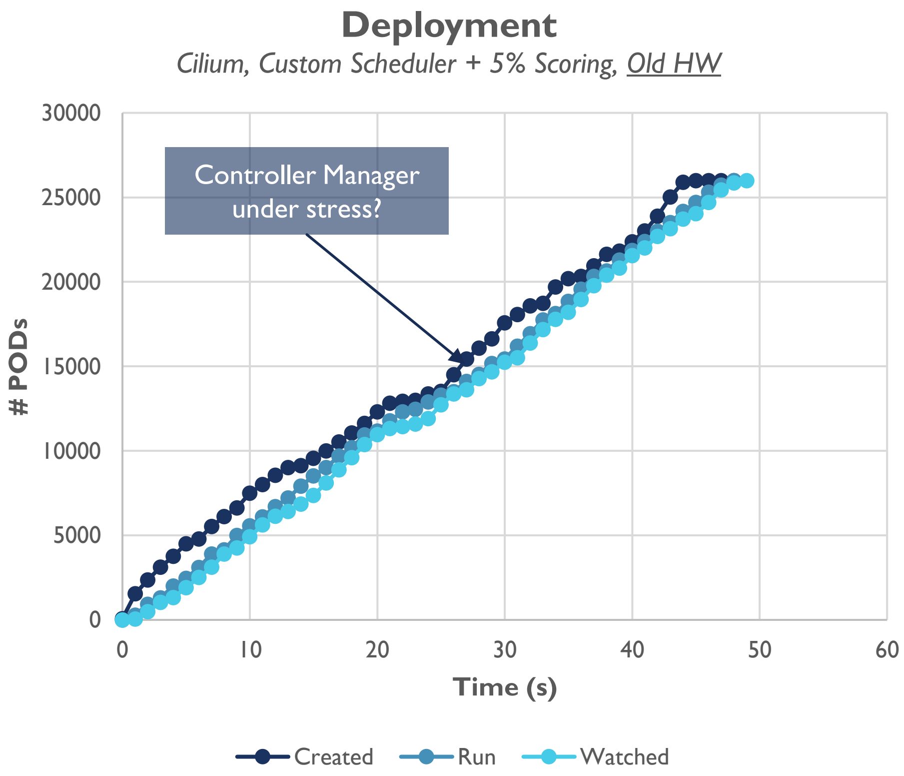
General observation

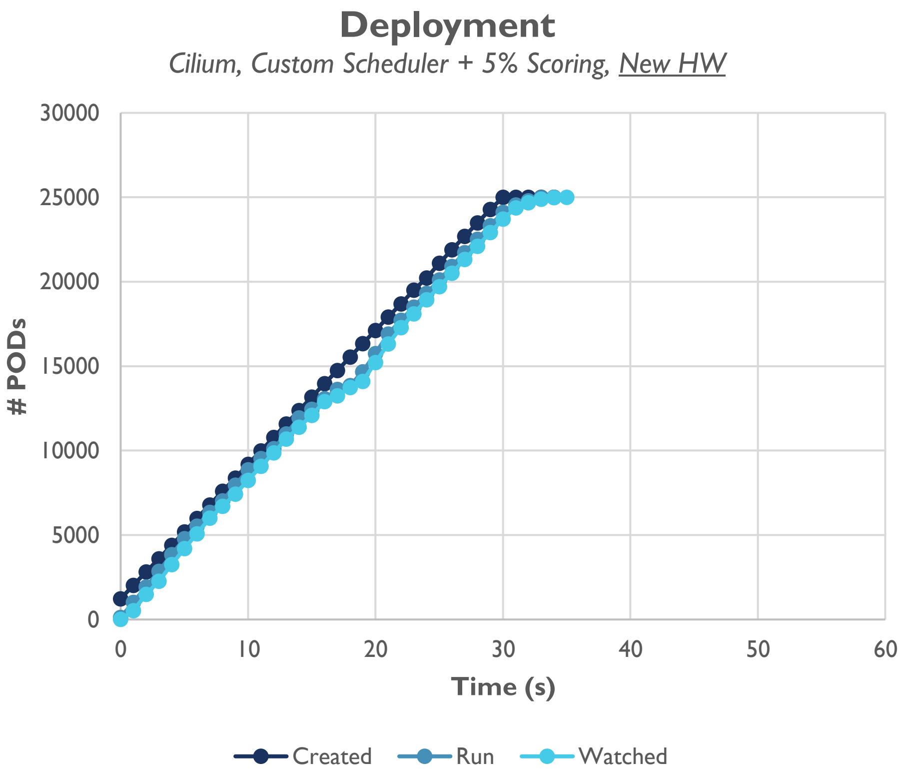
- Startup and stop times higher than the *HostNetwork* case

Differences go away with the improved (new) HW

POD STARTUP TIMES

WHAT ABOUT THE CNI?





LESSONS LEARNT

1. Some QPS tuning is needed in large clusters to improve POD startup and stop times

2. Scheduling time may be a bottleneck for large deployments

3. Scheduling strategies do impact the POD startup times

4. Global POD startup times seem to depend on the total number of PODs only

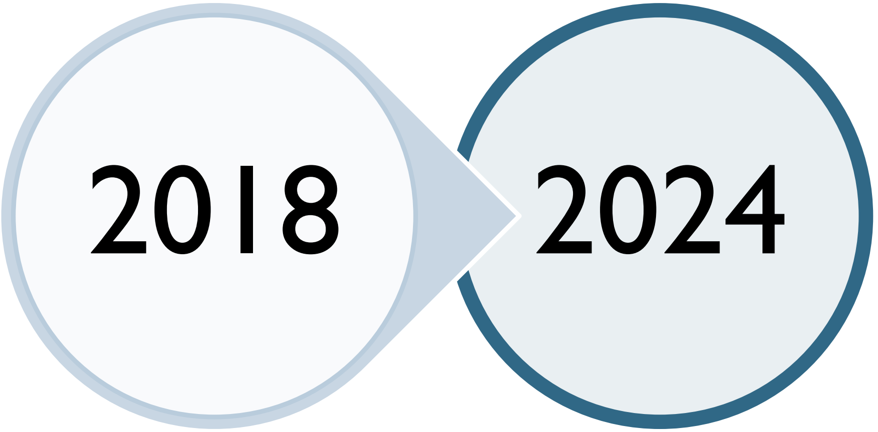
5. Custom Scheduler profiles may greatly increase its throughput

6. Better HW for the control plane seems to improve the more scheduling-demanding scenarios

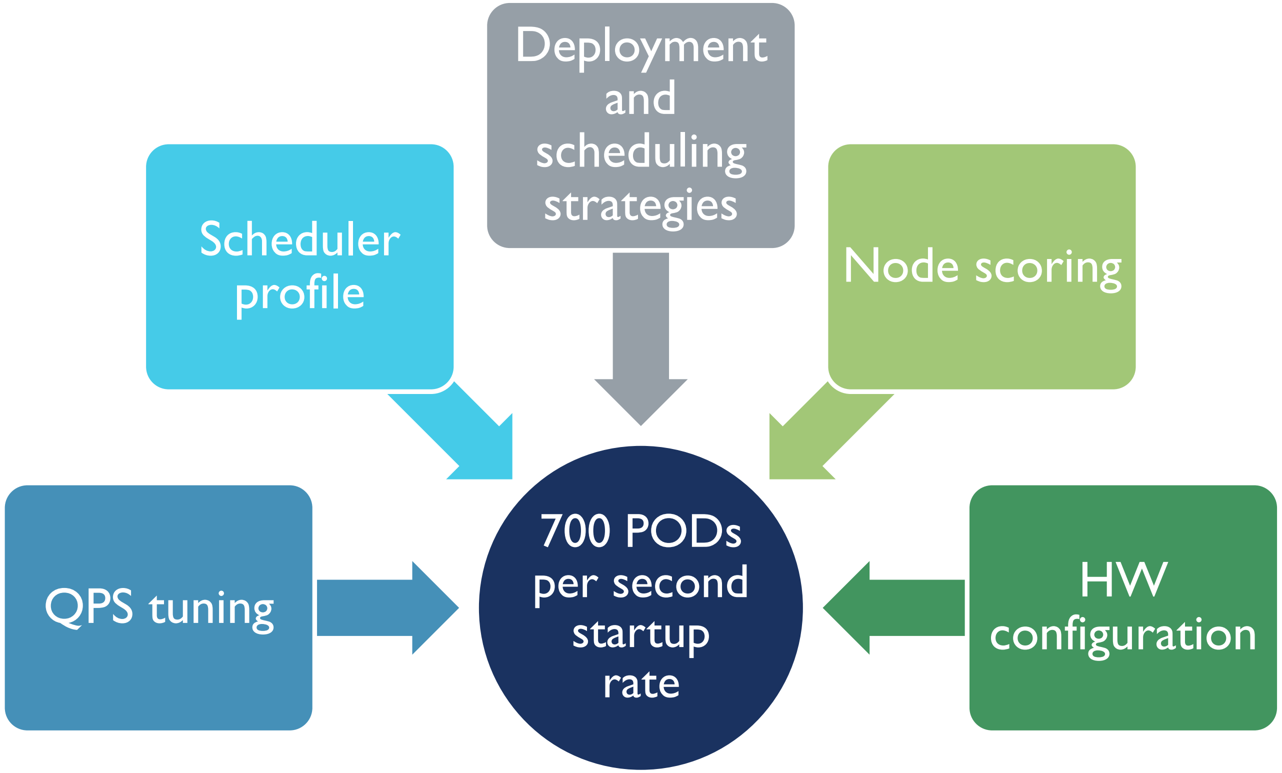
A KUBERNETES JOURNEY

THE REWARD

THE END OF (THIS) JOURNEY



- 20 PODs/s
- 700 PODs/s



CONCLUSIONS & OUTLOOK

Today

A lot of experienced gained in operating a large Kubernetes cluster

Great performance improvements in terms of POD startup times

Very flexible scheduling

Enhanced monitoring and operability

Predictable behaviour across several Kubernetes releases

Tomorrow

Scale to even more PODs

- In the worst-case scenario, about 65k data filtering applications replicas have to be deployed

Evaluate node extended resources for simplified scheduling

Consider more containers per POD if POD startup times get too high

Exploit Kubernetes for different workloads

- Dynamically mix online filtering and "offline" simulation jobs

What about *kwok*?

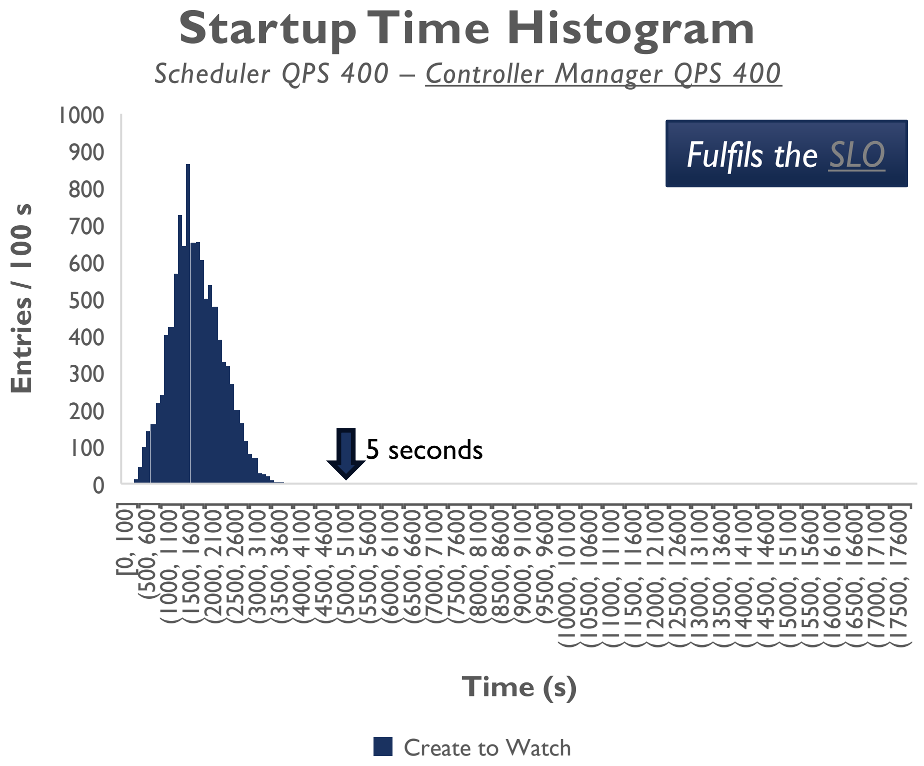
- The computing farm is available for Kubernetes tests only for very short periods during the year

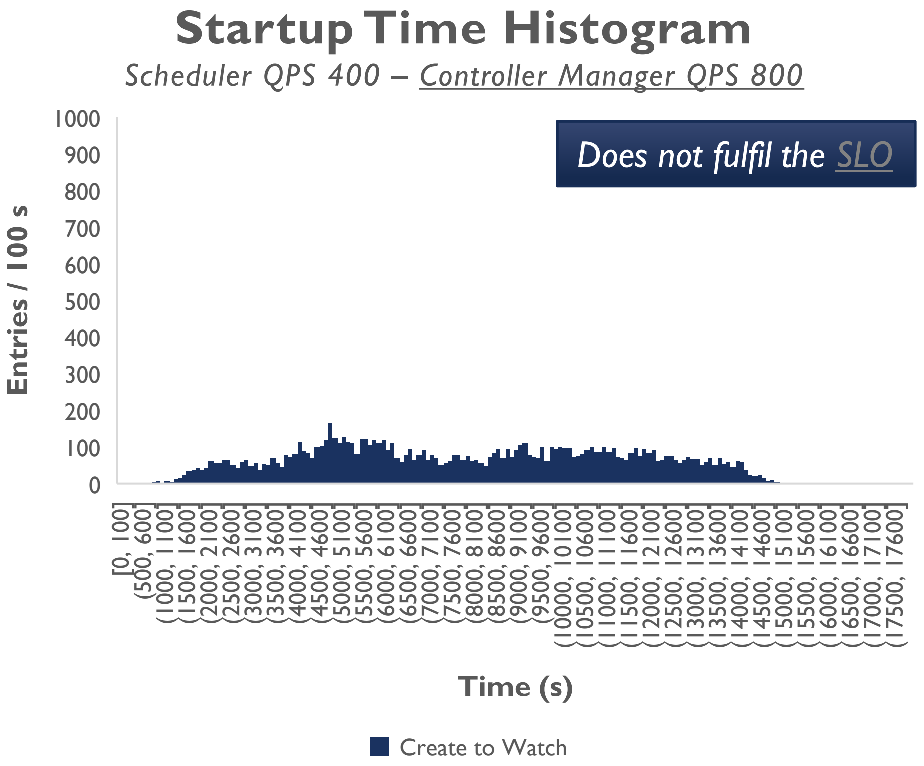
THANKS FOR YOUR FEEDBACK

BACK-UP SLIDES

POD STARTUP TIMES

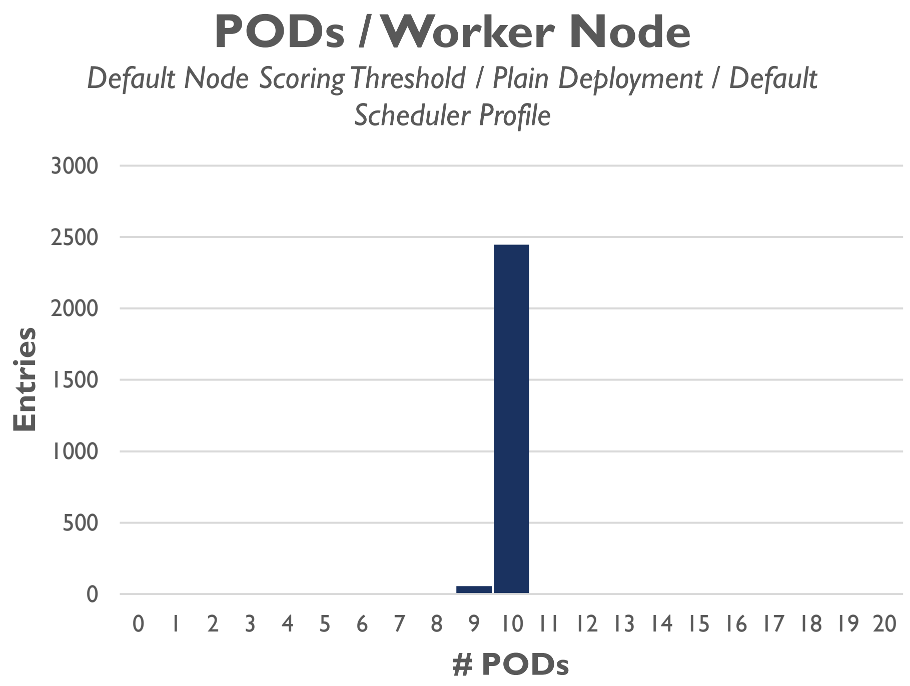
KUBE-API-QPS AND KUBE-API-BURST: A NOTE ON SLI/SLO

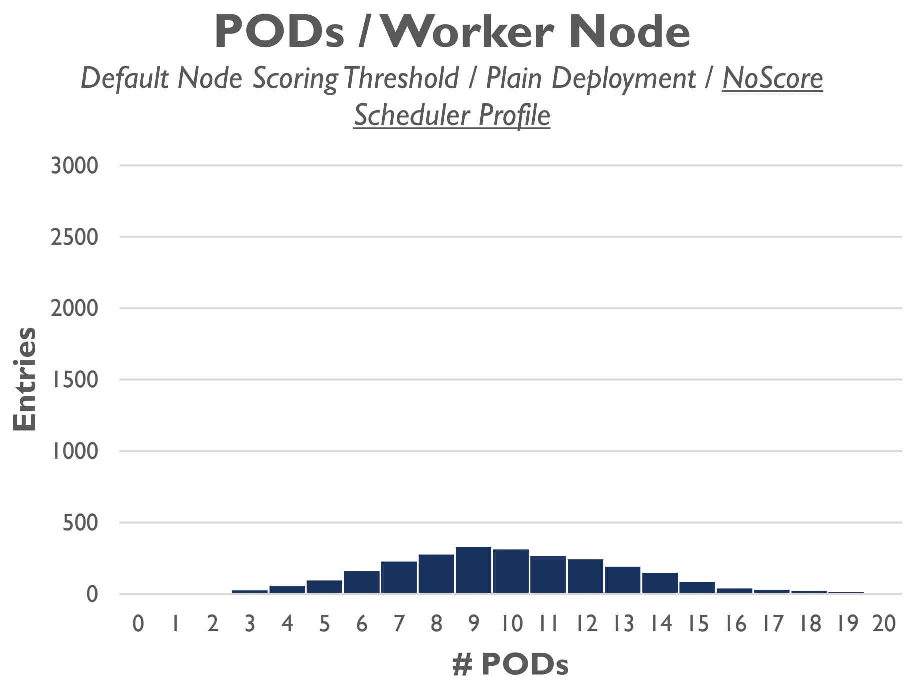




CUSTOM SCHEDULER EFFECT

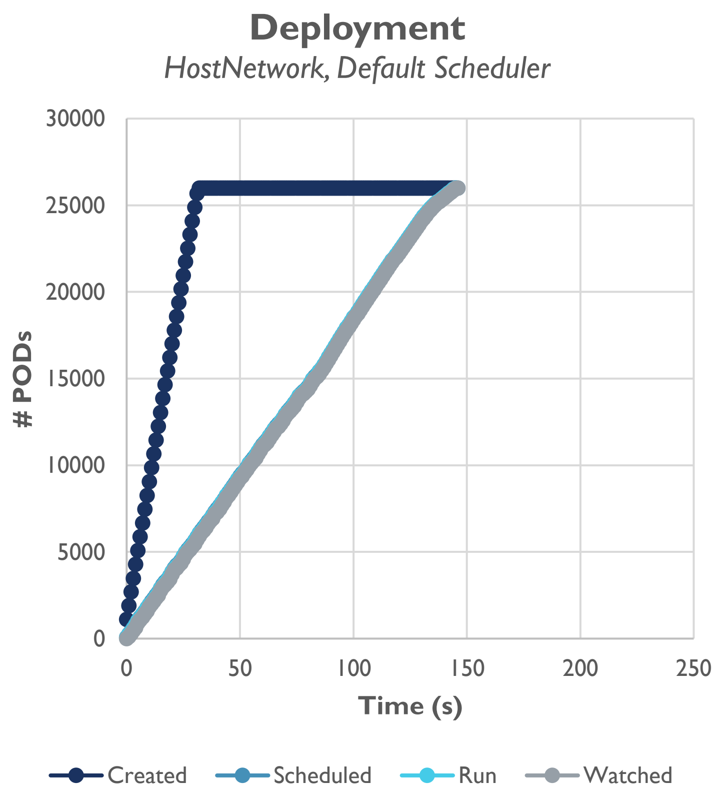
PLAIN DEPLOYMENT – NO CONSTRAINTS ON POD LOCATIONS

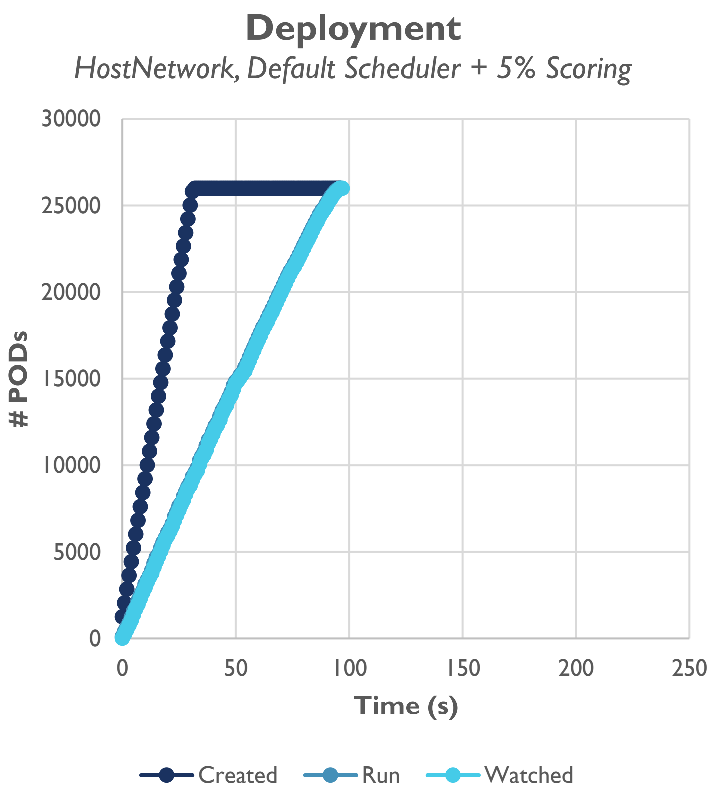


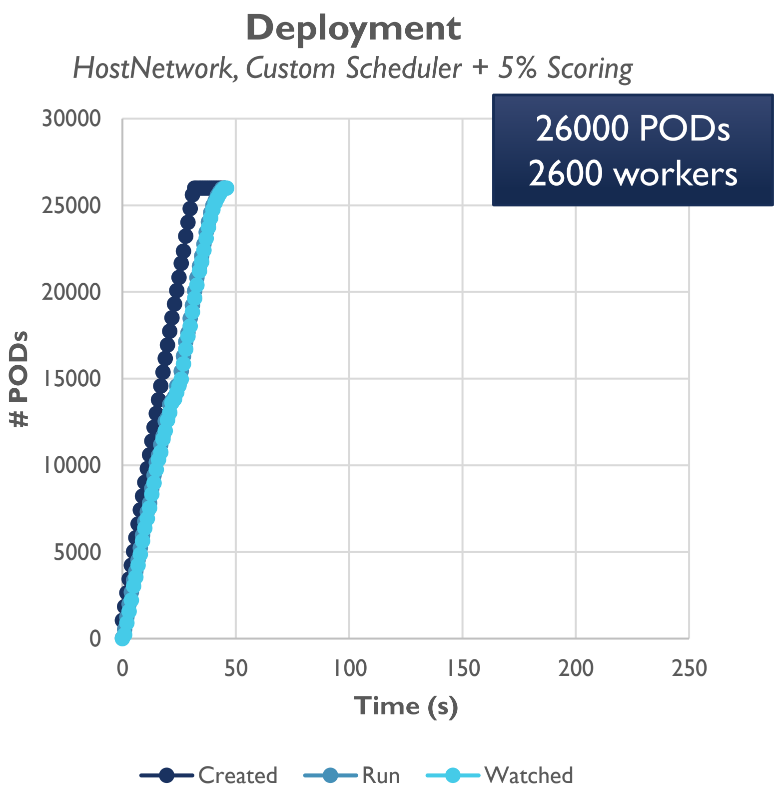


POD STARTUP TIMES

CUSTOM SCHEDULER PROFILE – STARTUP PHASES

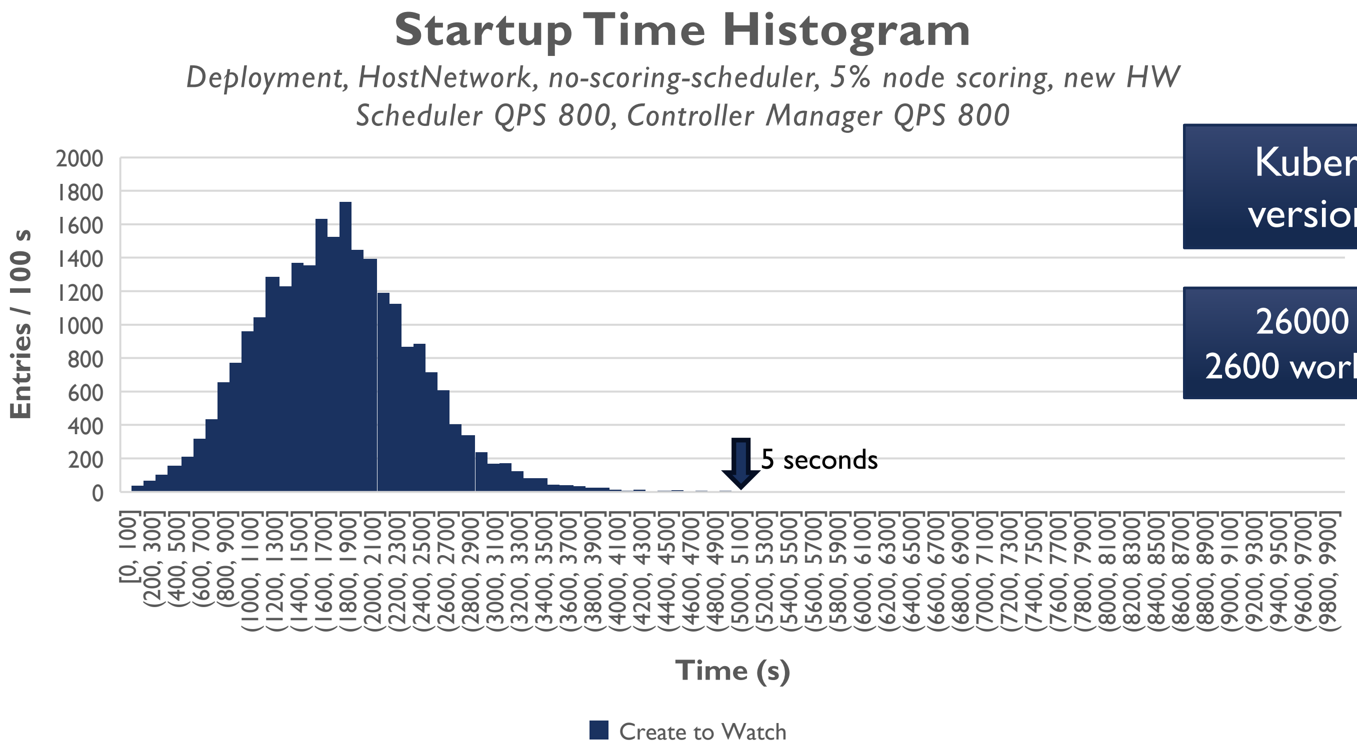






POD STARTUP TIMES

CREATE-TO-WATCH DISTRIBUTION (FINAL)



Kubernetes version 1.28

26000 PODs
2600 worker nodes

SENDING "EVENTS" TO A SEPARATE ETCD INSTANCE