Evaluating Kubernetes as an Orchestrator of the Event Filter Farm of the Trigger and Data Acquisition System of the ATLAS Experiment at the LHC

GIUSEPPE AVOLIO – CERN

MATTIA CADEDDU – CERN (ON LEAVE)

REINER HAUSER – MICHIGAN STATE UNIVERSITY

Outline

The ATLAS Trigger and Data Acquisition (TDAQ) system for the High Luminosity LHC (HL-LHC) era

Why Kubernetes?

Kubernetes functionality and features

Evaluating Kubernetes as an orchestrator for the Event Filter (EF) farm

- Running EF processes in Docker container
- Scaling tests

Conclusions

Roadmap to HL-LHC

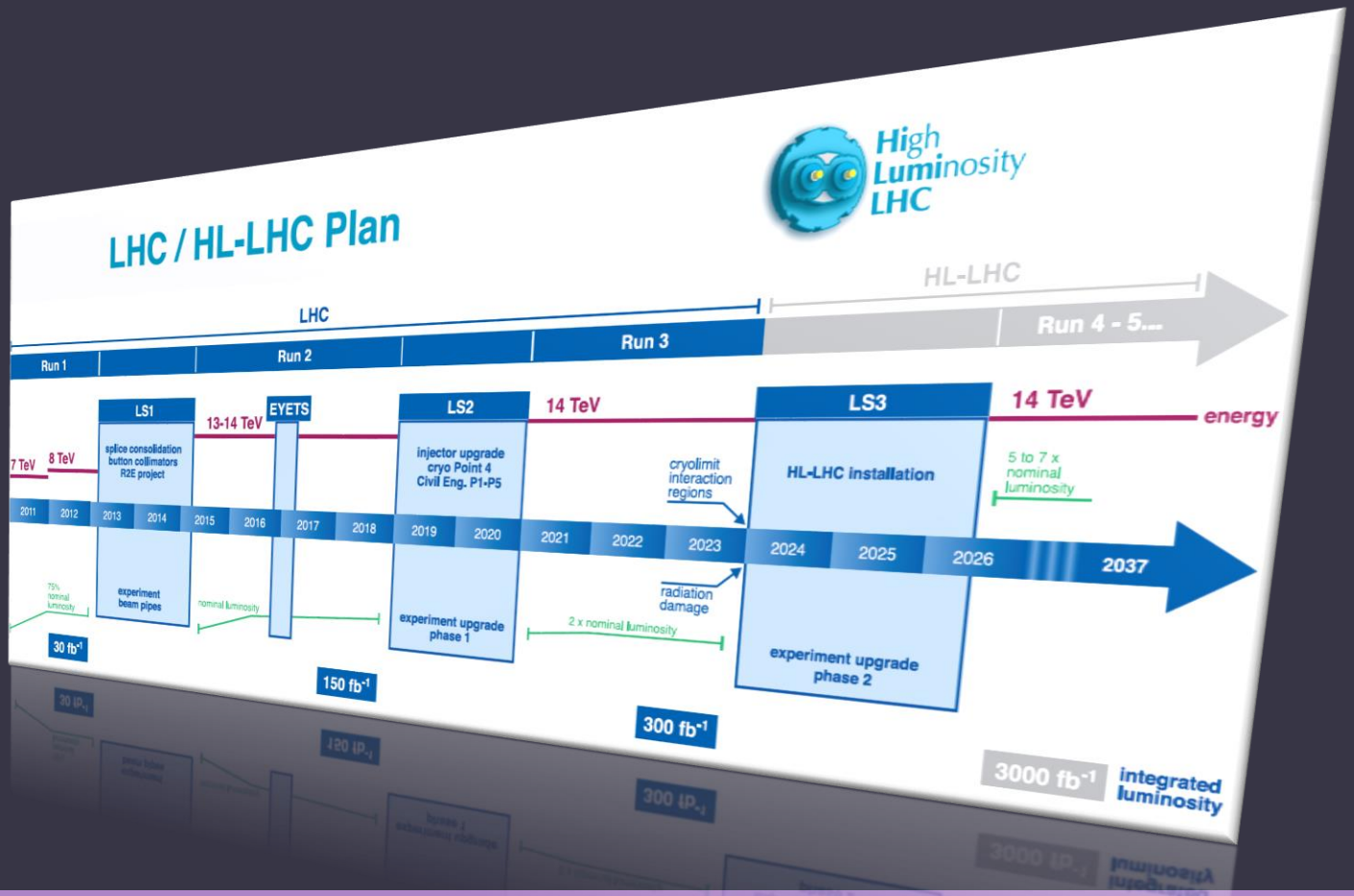
LHC is now in the last year of Run 2 operations

- Peak luminosity ~ **2 x 10^{34}** cm^{-2} s^{-1}
- More than **60** interactions per bunch crossing

HL-LHC will push the limit much higher

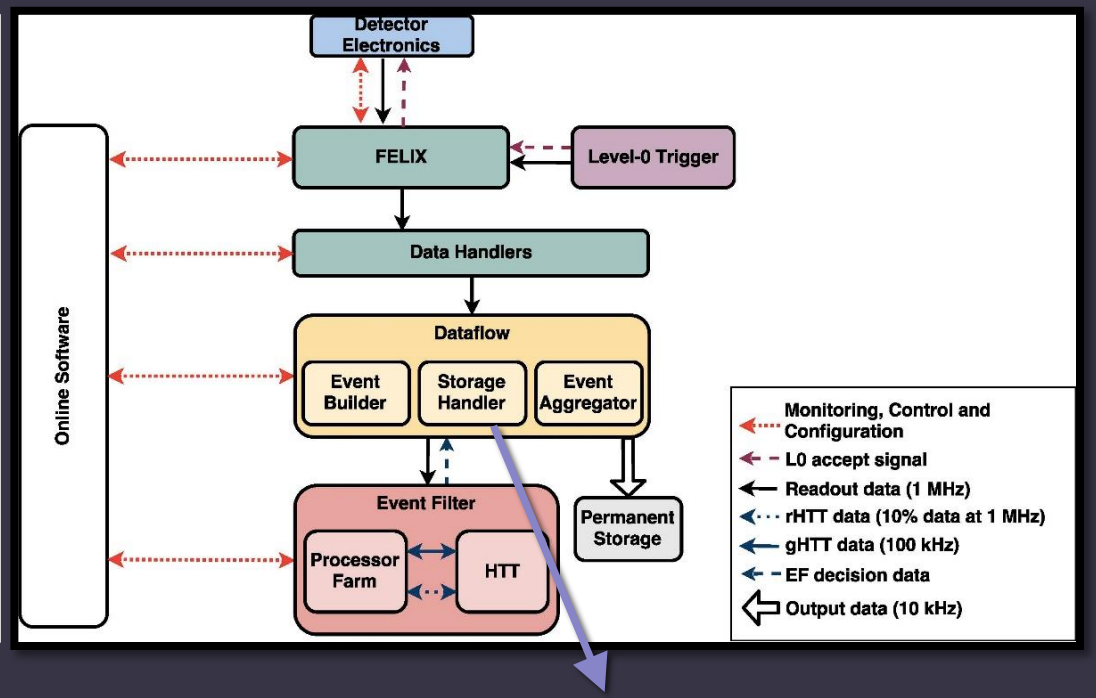
- Luminosity up to **7 x 10^{34}** cm^{-2} s^{-1}
- More than **200** interactions per bunch crossing

The data acquisition system has to cope with the higher luminosity



The ATLAS TDAQ System for HL-LHC

See **plenary S4**: *"ATLAS and CMS Trigger and Data Acquisition Upgrades for the High Luminosity LHC"*



The system has to sustain high rates

- Input data rate is **1 MHz**
  - 10 times more than Run 2
- Event size is about **5 MB**
  - 4 times with respect to Run 2

Highly distributed system

- Tens of thousands of applications to supervise

Large IT infrastructure

- The Event Filter farm only will consist of more than **3000** computing nodes

The **Storage Handler** buffers data received from the read-out system to **decouple** the read-out and the **Event Filter** (more than one hour of event buffering)

Why an Orchestrator?

Operating the EF Farm

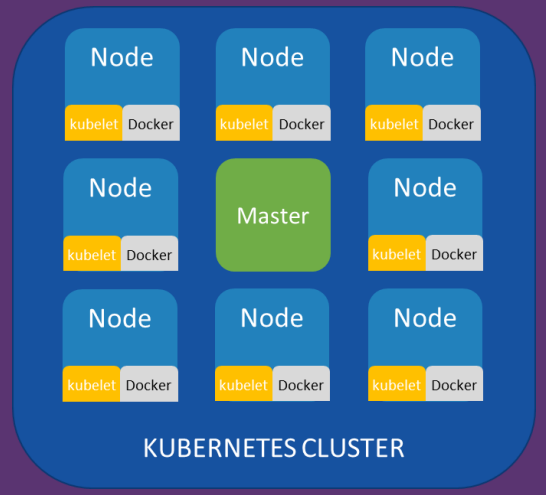
- The presence of the **Storage Handler** allows to operate the EF farm in different manners
  - **Decoupled** or not from the LHC cycles
  - **Prompt** or **delayed** processing
  - **Mixed** work-loads (*e.g.*, Monte Carlo production)
- A **robust** and **reliable** mechanism for the management of all processes running in the EF farm is a requirement to guarantee **stable** and **effective** execution of the EF service

Kubernetes

- Support to different application **life-cycles**
- Flexible **scheduling** and easy **scaling** of applications
- **Dynamic** handling of cluster resources
- Scaling to **thousands** of hosts
- Support for several **storage** back-ends
- **Containerized** applications

Kubernetes

An open-source system for automating deployment, scaling, and management of containerized applications



https://kubernetes.io/

Evaluating Kubernetes

Configuration

- Kubernetes version **1.5**
- CERN IT virtual infrastructure
  - Cluster with **1000** virtual cores
  - **1 master** node
    - 32 cores
    - 60 GB RAM
  - **240 slave** nodes
    - 4 cores
    - 8 GB RAM

Performed Tests

- Execute **EF processing units** in **containers** with Kubernetes
- Measure the time needed to **fully populate** the cluster for
  - different cluster size
  - different number of per-host instances of the same container
- Study the impact of the Kubernetes ***QPS*** (Query per Second) parameter set

The QPS Parameter Set

Several Kubernetes modules expose some configurable QPS parameters

- Mainly configuring the interaction with the API server

QPS tuning not really documented

- Some sparse information from few sources available on the web
- Digging into command line parameters of Kubernetes components

Approach

- Scale default values with some fixed multipliers

Component	Parameters (def. values)
kubelet	*event-qps* (5) *kube-api-qps* (5) *event-burst* (10) *kube-api-burst* (10)
kube-controller-manager	*kube-api-qps* (20) *kube-api-burst* (30)
kube-proxy	*kube-api-qps* (5)
kube-scheduler	*kube-api-qps* (50) *kube-api-burst* (100)

EF Processing Units in Kubernetes

Emulating EF processing units with the offline version of today's filtering software (AthenaHLT)

Docker container

- Base Scientific Linux CERN 6 (SLC6) OS image
- Few additional packages installed

Software retrieved from the CERN VM File System (CVMFS) repository

- Storage volume technology abstracted via a *FlexVolume* driver developed at CERN

Simulating data processing

- Input storage area with data files
- Output storage area with processing results

Kubernetes Volume Abstraction

```
volumes:
  - name: atlas
    flexVolume:
      driver: "cern/cvmfs"
      options:
        repository: "atlas.cern.ch"
  - name: atlas-condb
    flexVolume:
      driver: "cern/cvmfs"
      options:
        repository: "atlas-condb.cern.ch"
  - name: sft
    flexVolume:
      driver: "cern/cvmfs"
      options:
        repository: "sft.cern.ch"
```

Mounts in Containers

```
volumeMounts:
  - name: atlas
    mountPath: /cvmfs/atlas.cern.ch
  - name: atlas-condb
    mountPath: /cvmfs/atlas-condb.cern.ch/
  - name: sft
    mountPath: /cvmfs/sft.cern.ch
```

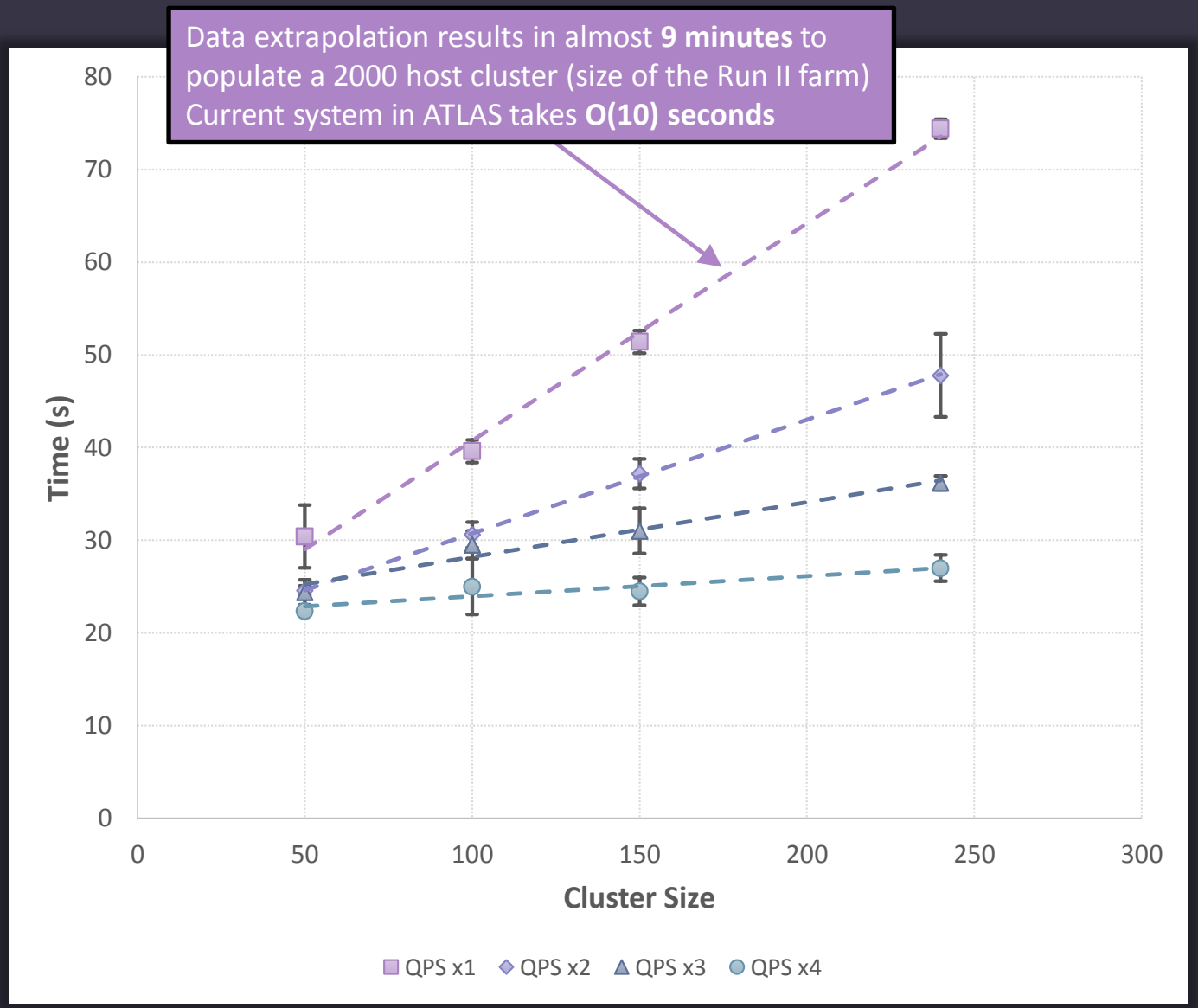
Kubernetes Tests

Time to fully populate the cluster
(using the *Google pause* container)

Container replicas: 5 per host

QPS: variable

Cluster size: variable



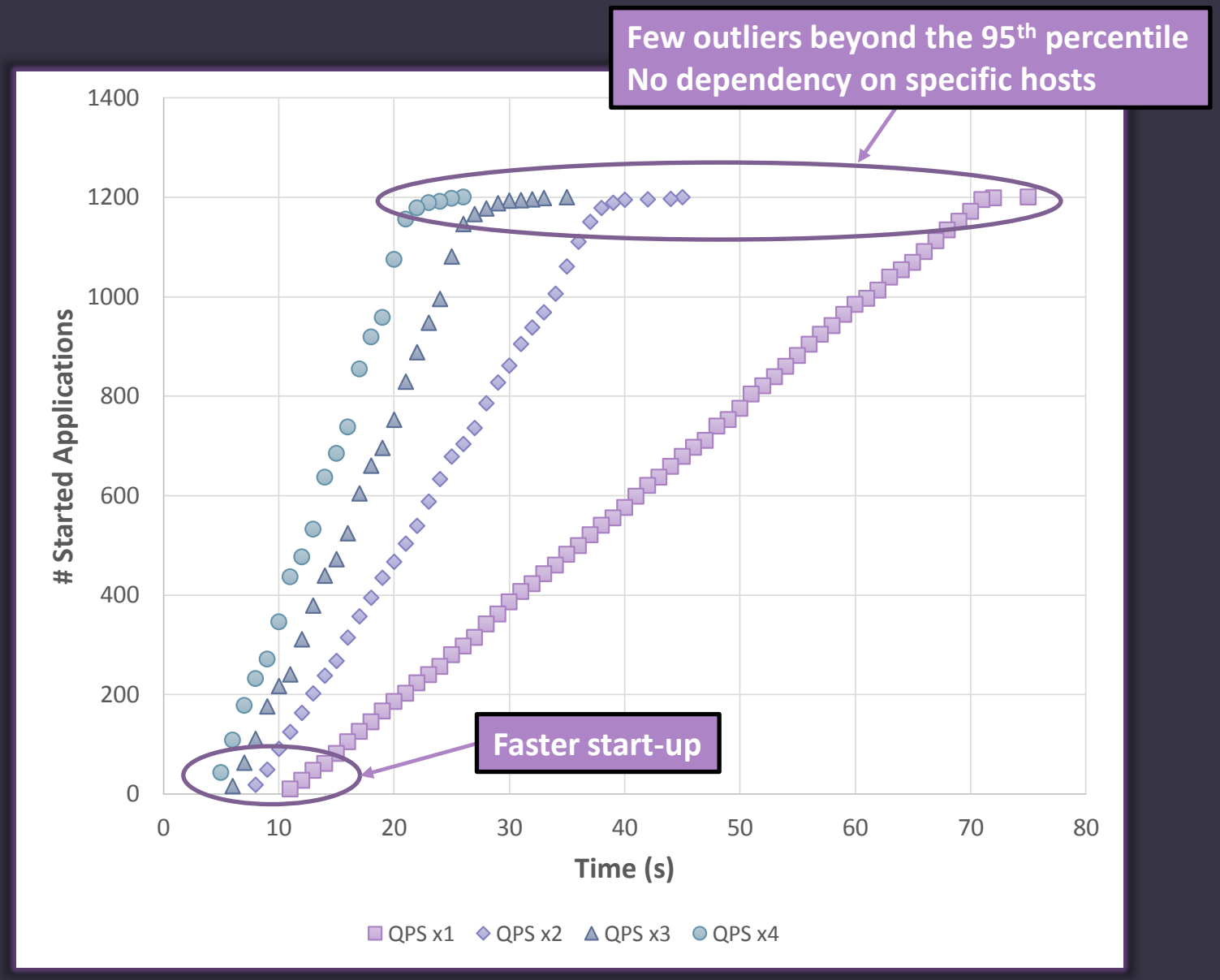
Kubernetes Tests

Time profile of started containers

Container replicas: 5 per host

QPS: variable

Cluster size: 240



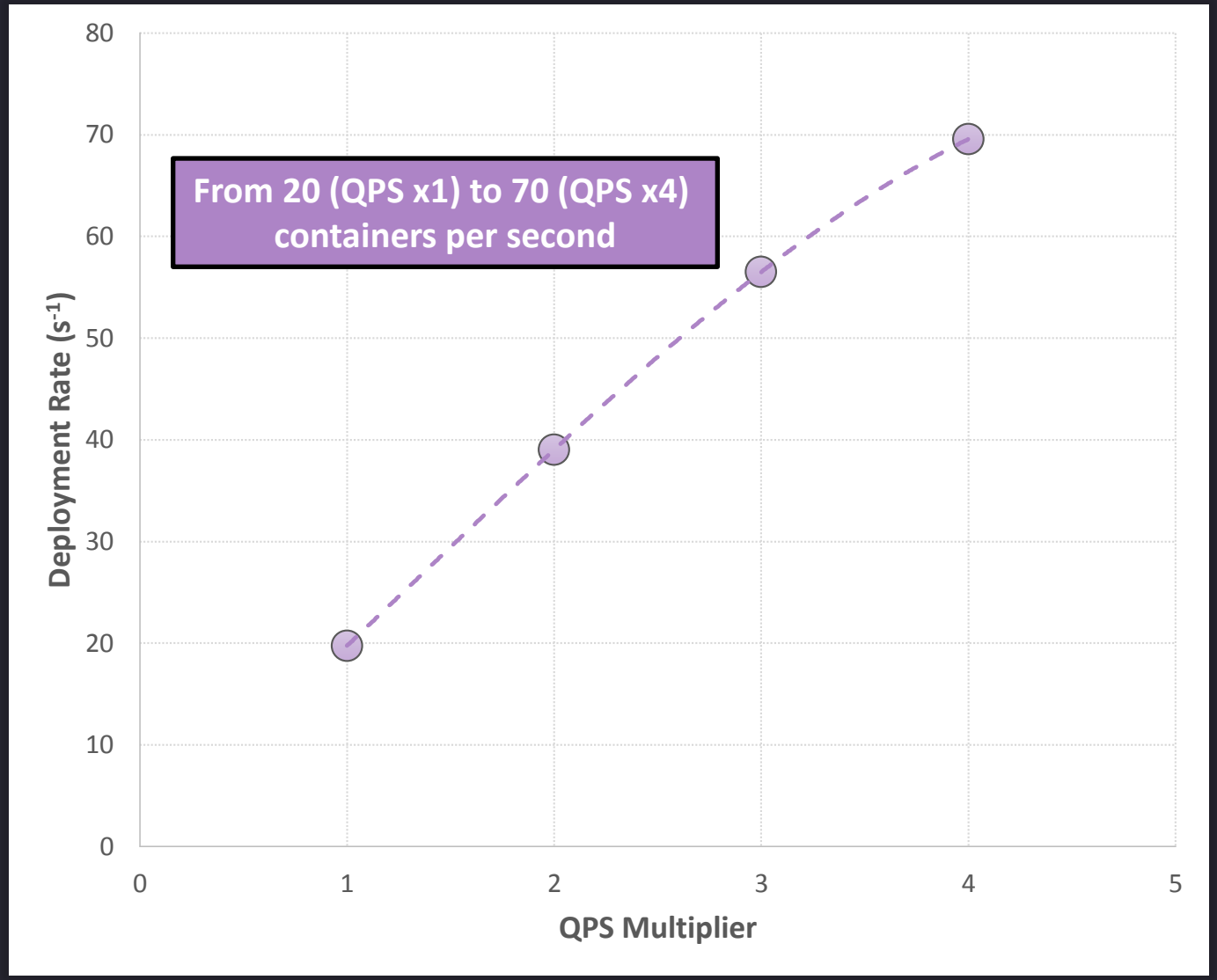
Kubernetes Tests

Container deployment sustained rate

Container replicas: 5 per host

QPS: variable

Cluster size: 240



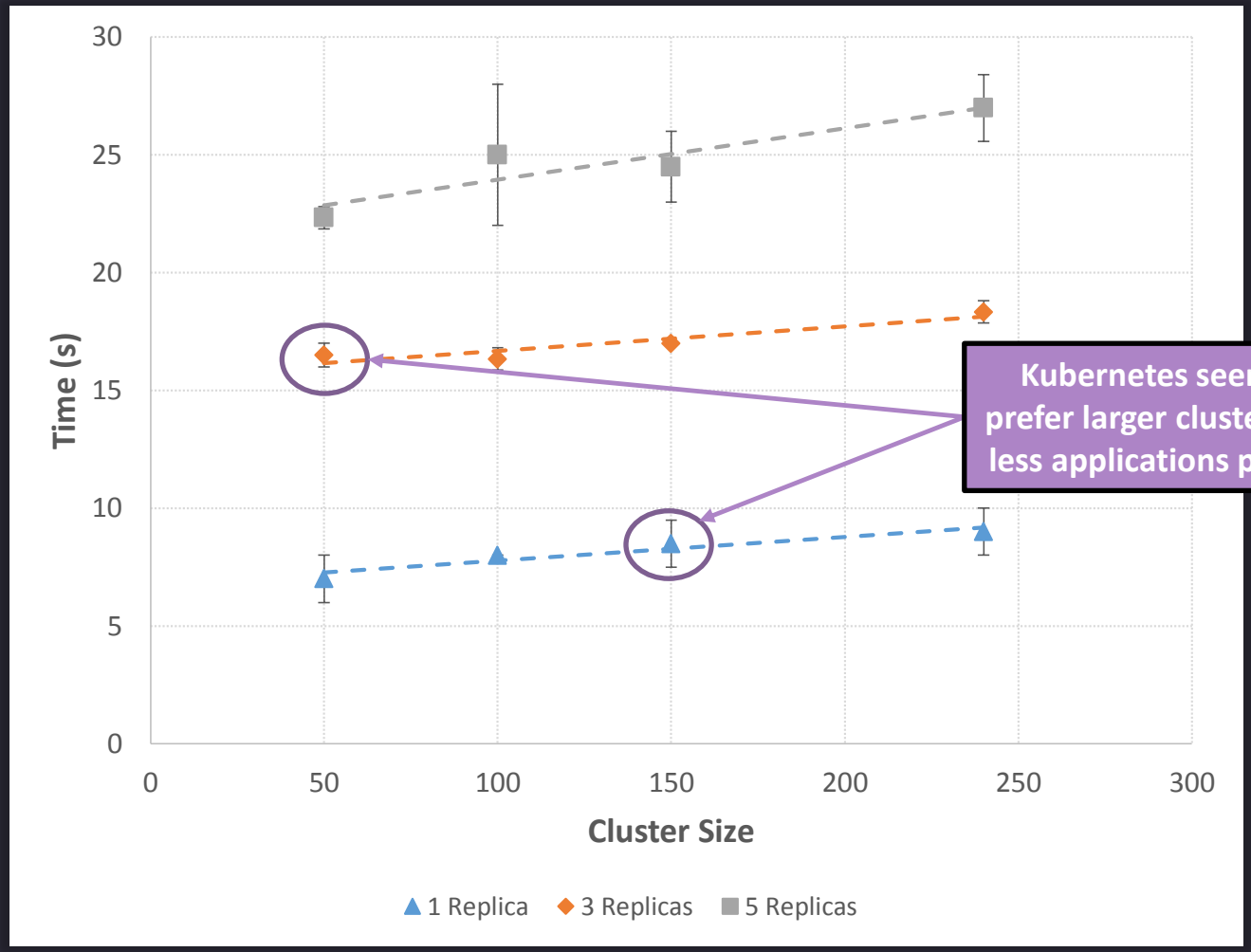
Kubernetes Tests

Time to fully populate the cluster
(using *Google pause* container)

Container replicas: variable

QPS: x4

Cluster size: variable



Extrapolating the obtained results to the Phase-II scenario (and excluding higher order effects for larger cluster – Kubernetes officially supports 5000 host clusters), the EF cluster (3000 hosts) will **be fully populated** with one processing unit instance on each host in about **35 seconds**

Conclusions & Outlook

Kubernetes provides a reach feature set…

- Easy scaling
- Flexible scheduling
- Native support to several storage back-ends

…and sufficient performances to be used as an orchestrator of the EF computing farm

- Fully populating a 3000 host cluster in about 35 seconds
- Performance is highly dominated by the *QPS parameter* set tuning
  - Several parameters in various Kubernetes modules (kubelet, controller manager, proxy, scheduler)
- Reached a sustained container deployment rate much higher than the out-of-the-box configuration
  - From 20 to 70 containers per second for QPS values four times bigger than the default configuration

Keep monitoring upcoming Kubernetes releases in order to track and verify evolving performance figures and new introduced features