Using Containers for HPC Workloads

HEPiX – Apr 21, 2016

Fritz Ferstl – CTO, Univa

Who is Univa

We solve the hard problems of shared mission-critical and scaled high-performance data centers

100
Use Cases

1,000
Applications

10,000
Deployments

grid engine

UNIVA

Proven Technology: 500+ Customers

Container Use Cases

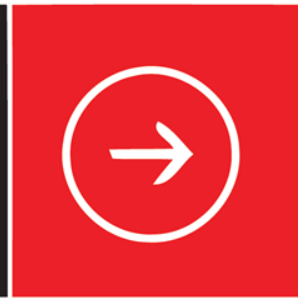
Use Case	Description
Sandboxing	Embed OS, SW stack and configuration dependencies with application
Resource Isolation	• Restrict resource usage within container • Limit interference of applications on same node
App Packing	Pack more applications on a node than possible with VMs
Portability / Cloud-Native	• Run same container on modern versions of Linux without changes • Including on-premise or public clouds
Microservices	Stateless single app containers working together to provide a scalable, highly available and portable service
DevOps	• Application-centric versus machine/server-centric design • Support for automatic container builds • Built-in version tracking • Reusable components • Public registry for sharing containers • A growing tools ecosystem from the published API

HPC Use Case Considerations

- What works well:
  - New applications architected with containers in mind
  - Sequential or multi-threaded
  - Simple IO and networking dependencies
- What can work but with challenges:
  - Distributed memory apps (MPI, etc) run on single host
  - Strong reliance on network access to POSIX filesystems (NFS)
  - GPU integration (issues are socket affinity, for instance)
- What doesn't really work (yet):
  - Cross-host distributed memory apps
  - Strong reliance on network access to POSIX filesystems (NFS)
- ➠ Choices
  - Re-architect your software
  - Look towards HPC-specific container technology (Shifter, Singularity)
  - Wait for mainstream container innovation (Docker has announced pure ipvlan support)
  - Focus adoption on new software

UNIVA

Bio-Tech Use Case



Use Case Description

- Centre for Genomic Regulation (CRG)
  - Barcelona, Spain
  - http://www.crg.eu
  - Life-sciences research
- Deployment of scientific data analysis pipelines on distributed clusters
  - Genome sequence discovery
  - Massive data analysis → large cluster
  - Life-science analysis pipelines have standardized on *Grid Engine*

Why Containers?

- Large amount of different versions
  - Life-science analysis pipelines are open source and constantly evolving
  - Scientists make their own modifications
- Large matrix of SW stack dependencies
  - Operating system
  - Workflow consist of many steps
  - SW stack layers with diverse version
- Need for reproducibility and versioning
  - This is **life**-sciences research, after all
- ➔ **Key Challenge**: maintain many versions of production environment over a long time ➔ *Sandboxing*

Use Case Environment

- Computing environment:
  - ~3,000 cores, heterogeneous systems
  - Managed with Univa Grid Engine
  - Non-trivial networking and shared storage
- Cluster is a shared resource
  - Many users
  - Advanced policies, e.g. fair-sharing, back-filling and dependable resource controls
  - Advanced job types, e.g. array jobs
  - Detailed accounting and billing for resource consumption
- High performance environment
  - Can't loose performance

Challenges and Solution

Challenges

- Sanboxing – maintain many production environments for a long time
- At minimal or no performance impact:
  - From running applications in a container
  - From network and shared file system access from within a container
  - From starting the same containers over and over on nodes
    - Avoid to reload images

Solution

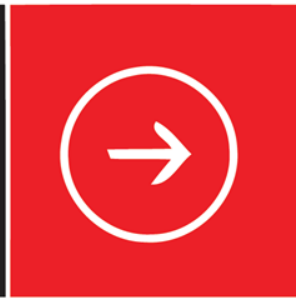
- CRG Nextflow workflow management
- Integrated with Univa Grid Engine
- And integrated with Docker
- Make Univa Grid Engine Docker-aware
  - Enable Docker jobs
  - Container image cache-aware scheduling

Results

- 4% increase of Docker application run-time vs native run-time with cached images
- 12,5% increase with container bootstraping, i.e. downloading from image repository
  - Image-cache aware scheduling has solid benefit on utilization and throughput
- Cost is considered low vs benefit by CRG
  - Use case requirements really can't be satisfied without containers

Univa Grid Engine

Container Edition



Docker with Univa Grid Engine

- Launch Docker Container on best machine in cluster
  - Minimize layer downloads *(not in early access version)*
  - Reduces the time wasted (it can be minutes) waiting for the Docker image to download from the Docker registry. Container runs faster increasing throughput in the cluster.

- Run Docker Containers in a Univa Grid Engine Cluster
  - Business Critical containers are prioritized over other containers. Increases efficiency of the overall organization.

- Job Control and Limits for Docker Containers
  - Provides user and administrator control over containers running on Grid Engine Hosts.

➔ **Currently in Early Access; Release in May 2016**

Docker Integration with Univa

- Accounting for Docker Containers
  - Keeps track of containers. Share policies require accounting.

- Data file Management for Docker Containers
  - Transparent access to input, output and error files. Simplifies the management of input and output files for Docker Containers and ensures any output or error files are moved to a location where the user can access them.

- Interactive Docker Containers
  - Good for debugging when containers don't work correctly!

➔ **Currently in Early Access; Release in May 2016**

Managing cloud-native and container-only micro-service frameworks

Based on Googles Kubernetes

Univa Container Solutions

LAUNCH

Easy installation, preconfigured solution including pre-integration with cloud services.

Build a container cluster on premise or in the cloud.

The fastest way to build a container cluster!!

COMMAND

Respond Quickly: Easy to resize, adapt, dynamic provisioning

Orchestrate and Optimize: Best use of resources and keep track of containers

The most advanced container orchestration!!

Solving the set up problem – in minutes

“Implementing Docker and Kubernetes is complicated.

It's even more complicated for enterprises that have diverse workloads, various app stacks, heterogeneous infrastructures, and limited resources.”

InfoWorld May 2015

navops

Rapidly deploy your container-ready infrastructure with pre-packaged, certified components:

docker + kubernetes + ATOMIC + UNIVA

LAUNCH

“The next step is large scale orchestration and scale”.

451 Research

Before

Containers and other workloads need resources

COMMAND **Navops orchestration solution**

- Run containers at scale
- Blend containers with other workloads
- Maximize resources / use of cloud

After

On Premises Servers / VMs

Cloud

UNIVA

Conclusions

- Containers are well suited to address use cases like sandboxing
- A bit of a paradigm shift in SW development, networking, storage, security, provisioning, etc
- Production use requires enterprise-ready container orchestration
- Univa Grid Engine Container Edition for
  - Mixed Workloads
  - "Fat" Containers (no microservice)
  - "Jobs"
- Navops for container-only, microservices-based SW architectures