Virtualisierung und Container im Kontext kognitiver Workloads

Dr. Basil Moshous
Executive Client IT Architect
Member of the IBM Academy of Technology (AoT)
E-mail: moshous@de.ibm.com

IBM Systems for Cognitive Solutions, Ehningen July 11-12, 2017
Agenda

1. Build & Run of Services
   • Monolithic Services
   • Microservices
2. Virtualization Concepts
   • Bare Metal Systems and Virtualized Machines
   • Containers
   • Container Evolution and what makes Docker interesting
3. How to deploy cognitive workloads
   • Hybrid Cloud
   • Public APIs
The industry has changed!

**Systems of Record:**
- easy definition of functionality
- monolithic applications
- long development cycles
- slowly scaling up

**Systems of Engagement:**
- new functionality as business advantage
- decoupled services
- fast, iterative improvements
- quickly scaling out

**Cognitive Systems:**
- Leverages elements of both worlds
Build & Run of Services
Traditional Requests from the Business

"I want a XYZ system", $XYZ=\{\text{ERP, HR, SCM, ...}\}$

**Traditional Services:**
- Enterprise Resource Planning
- Human Resources Tools
- Supply Chain Management
- Custom
- ...
Monolithic Design, Service Bus

Traditional Services were build using monolithic Designs

- Three Tier Architectures are very popular
- All Application Logic is done in one layer, the monolithic Application Server
- Consistency is key for Data
- Predominant development method: Waterfall
  - One Releases a year
  - Program changes have to be tested very well
- Connectivity to other Services: APIs and/or Service Bus

Examples

Browser
JAVA
WebSphere
Netweaver
Tomcat
DB2
Oracle
Cloud deployment: Open Source LAMP

Web cache
- Squid
- Polipo
- Traffic server

Web server
- Apache
- Cherokee
- Lighttpd
- Nginx

CGI scripting
- Perl
- PHP
- Python

Database
- MariaDB
- MySQL
- Drizzle

Linux kernel
- AppArmor
- SELinux
- Smack
- TONOYO
- Process Scheduler
- Netfilter
- Linux network stack
- Network scheduler
- NIC device driver
- kmod-fo-ext4
- kmod-fo-btrfs
- Lustre

Hardware
- CPU & RAM

Networking hardware

Requests
- Serve

Responses
- Low latency

Storage
- SATA
- SAS
- RAID
- iSCSI
- NAS

Applications:
- WordPress
- Moodle
- MediaWiki
- Magento
- SugarCRM
- Bug Tracker

Source: https://en.wikipedia.org/wiki/LAMP_(software_bundle)
Issues with Waterfall development model

Changes to monolithic applications using a Waterfall development method tend to take a long time!

We finally adjusted the application to your needs. Sorry it took so long.
A microservice should only provide a single service. “Do only one thing and do it good.” A business process will be composed out of a series of microservices. A microservice should terminate after it performed its duty.
Decomposition of a process into Microservices

Monolithic Design

- Step 1
- Step 2
- Step 3
- Step 4
- Step 5

Microservices

- Service 1
- Service 2
- Service 3
- Service 4
- Service 5

TIME

IBM Systems for Cognitive Solutions, Ehningen July 11-12, 2017
Operating Microservices adds Complexity

Monolithic Design

Microservices

Be aware of the complexity you get 😊
Virtualization Concepts
Bare Metal and Virtualized Machines

![Diagram of Bare Metal and Virtualized Machines]

- **Bare Metal Machines**
  - Host OS
  - Server
  - Bins/Libs
  - App A

- **Virtualized Machines**
  - Hypervisor (Type 2)
  - Guest OS
  - Bins/Libs
  - App A
  - App A'
  - App B
  - VM

IBM Systems for Cognitive Solutions, Ehningen July 11-12, 2017
Linux Startup Process / Bare Metal

BIOS Phase
- performs startup tasks;
- executes the boot code

Boot loader Phase
- menu of possible boot options
- loads the kernel into memory
- gives it control kernel

Kernel Phase
- essential hardware and memory paging
- start_kernel() performs the majority of system setup
- starts up the idle process
- starts up the scheduler
- starts up the init process

Init process
- establishes and operates the entire user space
- Starts up all daemons configured
  - ssd
  - networkd
  - multipathd
  - ...

Application Phase

Startup Time: Minutes
Linux Startup Process / Virtual Machines

Startup Time (Once): Minutes

Startup Time: Minutes
Containers
Linux Kernel Level Virtualization

Cgroups:
confinement and prioritization of resources (CPU, memory, block I/O, network, etc.)

Namespaces:
Isolation of process trees, networking, user IDs and mounted file systems
Application Container

multiple isolated user-space instances - look and feel like a real server

Application Container

Virtualization

Host OS

Guest OS

Hypervisor (Type 2)

Server
Linux Startup Process / Application Container

Startup Time (Once): Minutes

Startup Time: ms, no resources dedicated
Memory Resources

Bare Metal

- Memory Dedicated to the Host OS
- All Applications share the available Memory

Virtualized Machines

- Memory Dedicated to the Guest OS
- All Applications in a particular guest share the available Memory in this guest

Container

- Memory Dedicated to the Host OS
- All Applications share the available Memory
Container Evolution
“Purpose of creating open industry standards around container formats and runtime”
What makes docker so attractive?

BUILD, SHIP, RUN
Docker is a **software containerization platform**

- **Build** the container anywhere (for example on your notebook!)
- **Ship** containers with all their dependencies into a corporate repository
  - Break image into layers
  - Only ship layers that have changed
  - Save disk, network, memory usage
  - Deploy from that corporate repository
- **Run** anywhere
Docker Build, Ship, Run

Development Environment

- Pull Image
- Build
- Container

On or Off Premise Cloud Deployment

- Search
- Pull

Prod Hosts: OS (Linux)

Container A

Container B

Container C

Company Docker Repository

Development Environment

- Dockerfile For A
- Source Code Repository
- Dev Host: OS (Linux, Mac OS, Windows)
Managing Containers

Serverless

PaaS

Container Orchestration

Container Engine

Event Driven
Milliseconds to deploy
Live for seconds

Swarm

IBM Spectrum Conductor for Containers
How to deploy cognitive workloads
## What did we learn for the infrastructure

<table>
<thead>
<tr>
<th></th>
<th>Bare Metal</th>
<th>Virtualized</th>
<th>Application Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Start Time</td>
<td>++</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Separation</td>
<td>--</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Agile Application Deployment</td>
<td>--</td>
<td>--</td>
<td>++</td>
</tr>
<tr>
<td>Data Persistency</td>
<td>++</td>
<td>++</td>
<td>--</td>
</tr>
<tr>
<td>High IO Performance</td>
<td>++</td>
<td>++</td>
<td>--</td>
</tr>
</tbody>
</table>
What is important if dealing with monolithic apps

<table>
<thead>
<tr>
<th></th>
<th>Bare Metal</th>
<th>Virtualized</th>
<th>Application Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Start Time</td>
<td>++</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Separation</td>
<td>--</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Agile Application Deployment</td>
<td>--</td>
<td>--</td>
<td>++</td>
</tr>
<tr>
<td>Data Persistency</td>
<td>++</td>
<td>++</td>
<td>--</td>
</tr>
<tr>
<td>High IO Performance</td>
<td>++</td>
<td>++</td>
<td>--</td>
</tr>
</tbody>
</table>
What is important if dealing with microservices

<table>
<thead>
<tr>
<th></th>
<th>Bare Metal</th>
<th>Virtualized</th>
<th>Application Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Start Time</td>
<td>++</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Separation</td>
<td>--</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Agile Application Deployment</td>
<td>--</td>
<td>--</td>
<td>++</td>
</tr>
<tr>
<td>Data Persistency</td>
<td>++</td>
<td>++</td>
<td>--</td>
</tr>
<tr>
<td>High IO Performance</td>
<td>++</td>
<td>++</td>
<td>--</td>
</tr>
</tbody>
</table>
So how to bridge the different requirements?
Hybrid Cloud: Systems of Record & Engagement

Systems of Record
- IBM
- SAP
- ORACLE

Systems of Engagement
- μ
- Container ship
IBM Systems for Cognitive Solutions, Ehningen July 11-12, 2017

Hybrid Cloud: Infrastructure View

- Bluemix
- Integration Services
- On-Premise API Enablement
- API Management
- DataPower GW appliance
- DMZ
- z Systems
- z/OS Connect
- Power Systems
- Cloud – Mobile App Enablement
- Cloud - Integration Services
- Enterprise APIs
- Web Services or REST based services
- Enterprise Transaction Processing
- IBM Worklight

Off Premise or On Premise (p,z)
IBM Power Systems Container Solution

IBM Hardware

IBM Spectrum Software

IBM Spectrum Scale (+ Ubiquity !)

IBM Spectrum Conductor for Containers

IBM Spectrum Cluster Foundation
IBM LinuxONE Container Solution

• IBM LinuxONE gives you the ability to run industry leading enterprise containers.
• IBM LinuxONE offer integrated container solutions that can meet the diverse needs of enterprises.
• Supporting the creation and deployment of multi-platform, multi-container workloads across hybrid infrastructures.
• IBM LinuxONE and containers accelerate application delivery and enable application lifecycle management for Dockerized containers.
IBM Systems + IBM Cloud - Build your apps, your way

A Full Spectrum from Bare Metal to Event Driven

*Use a combination of the most prominent open-source compute technologies to power your apps.*

- **OpenWhisk**
  Event-driven apps, deployed in a serverless environment.

- **Instant Runtimes**
  App-centric runtime environments based on Cloud Foundry.

- **IBM Containers**
  Portable and consistent delivery of your app without having to manage an OS.

- **Virtual Machines**
  Get the most flexibility and control over your environment with VMs.

- **Bare Metal**
  For the ultimate performance and scale

Ease of getting started  Full stack Control
Thank You