Service Oriented Interactive 3D Digital Twin

GOALS: MEASURING PERFORMANCE OF INTERACTIVE I4.0

- Deployment and scaling analysis & measurements for service oriented interactive media engines
- Background: Driving paradigm changes in I4.0 (Industry 4.0), Retail & CPQ (Configure Price Quote)
- Cost efficient instant availability anytime anywhere (no loading times) instead of "lift & shift".

CHALLENGES: ARCHITECTURS FOR INTERACTIVE 3D

- Classical service oriented architectures are not suited for interactive media applications and 3d
  → requires service oriented interactive media (SOIM) architectures
- New challenges: Node GPU access, Windows platforms (including WMF, DirectX) , efficient deployment

EXPERIMENT SETUP: GPU NODES

- Virtual Wall of imec:
  gpunodes n083-01, n083-03 - n083-10
- Multi-user service oriented interactive media (SOIM) architecture
- Windows Server 2012 WM on Ubuntu 16.04
- Fully automated pipeline
  - File copying, VM image creation, upload, deployment, launching
- Deployment abstraction layers
  - To be ready for future different non-commercial & commercial clouds

LESSON LEARNED: VM & GPUS

Lesson learned: Containers intead of VMs

- Our service was running successfully within windows WM hosted by Linux WMs on the Virtual Wall.
- But no satisfactory access to the GPU of the gpunodes
  - Fall-back drivers, not suitable for high-performance interactive rendering and media
  - Seems to be a general challenge of VMs (not OS specific).
  - Important aspect: Performance fragility of service oriented interactive media architectures

RESULTS: PERFORMANCE OF AUTOMATIZATION AND DEPLOYMENT

- Automatization of packaging and deployment works very well: Python based & shell scripts, Virtual Wall and Fed4FIRE connection and tunnel management, Automated SSH. Multiple abstraction layers: Content, packaging, transfer, startup, logging
  - Essential for rapid workflow and heterogeneous application pool. But brute-force upload is very slow: An hour and more. Incremental updates hard to do on image base, but possible on content package base
- Deployment times outshadow startup times: Image upload: 1:20 - 1:30 hours
  - Node availability waiting times: Typically 12 - 25 minutes
  - WM startup times: 1:20 - 1:50 minutes
  - Service executable startup time: 12 - 14 seconds
  - Service content loading time (base & specific): 2:15 - 2:33 minutes

CONCLUSIONS

- Learned need for sharing data for elasticity of heterogeneous collection of multiple applications
- VMs not optimal: Choice between different images vs. separate content management → Lesson learned: Looking into Windows containers
- Tighter integration of platform with applications needed
- Looking into container layers now for better deployment performance

POST MORTEM

Experiments had significant impact on future technology strategy:

- Decision for containerization and layers
- Multi-layered deployment automatization architectures, smarter orchestration
- Decision for a new combination of own tech with Docker, Kubernetes and VPNs (for heterogenous infrastructures)