Algorithm development and evaluation with virtual environments

Dipl.-Inf. Robert Manthey
• 2008: University degree in computer science at Technical University of Chemnitz

• Since 2008: Different software projects at database management, ergonomics, facility-and-school safety, automotive embedded system testing, media processing and image understanding

• 2017: Guest research at NIST
Solving a problem needs an Algorithm to process the Data and handle the Constraints in a given Environment.

BUT
Solving a problem needs an Algorithm to process the Data and handle the Constraints in the given Environment.

**BUT**

Verifications, Evaluations and Competitions need similar elements.
Often only Algorithm, Data and related informations being provided.

BUT, no informations of

- Environment-dependencies to external programs or libraries, environment variable settings, functions from older versions of source code etc.
- Constraints to specific libraries, certain hardware/GPU, parameter settings etc.

→ Much “Debugging” needed
Virtual Machines provide:

- Predefined setup of external programs, environment variables, libraries and parameters
- Different levels of hardware and system abstraction
- Easy to provide and to share
- Increase reproducibility
- Hypervisor between hardware and OS
- Slight overhead
- Good isolation of host and guests
- Each guest with own hardware driver possible
- Complex configuration
- Consumer GPU/CUDA limitations
• Hypervisor between host OS and guests
• Slightly more overhead
• Good isolation of host and guests
• Each guest with own paravirtual hardware driver possible
• Easier and flexible configuration
• Consumer GPU/CUDA limitations
- Hypervisor between host control application and guests processes
- Slightly more overhead
- Isolation of processes, libraries and user environments
- Guests share kernel, drivers and host resources (→ Linux only)
- Simple and flexible configuration
- Guest creation possible with:
  - Version control
  - Inheritance from existing image
  - Easy setup of guest configuration and environment
  - Reproducability
- On Linux, GPU access through host driver on Linux with *nvidia-docker*[^5]
FROM nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04
MAINTAINER robert.manthey@informatik.tu-chemnitz.de

# Args
ARG PackageFile="Packages.txt"
ARG CPUNumbers="8"

# Copy data to image
ADD $(PackageFile) /$(PackageFile)

# Install common packages
RUN apt-get update & & apt-get upgrade -y & & apt-get install -y $(cat /$(PackageFile))

# Upgrade pip
RUN pip install --upgrade pip

# Install Caffe, Models & OpenPose
RUN cd /opt/
   
   git clone https://github.com/CMU-PercyBloor-Lab/openpose.git
   cd /opt/openpose/3rdparty/caffe/;
   cp Makefile.config.Ubuntu16_cuda8.example Makefile.config;
   make all -j$[CPUNumbers] & & make distribute -j$[CPUNumbers];
   cd /opt/openpose/models/;
   .getModels.sh;
   cd /opt/openpose/;
   cp ubuntu/Makesfile.config.Ubuntu16_cuda8.example Makefile.config;
   make all -j$[CPUNumbers]

# Config
#RUN

# Create user
RUN useradd -r -u -u 1000 openpose & echo "openpose ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/openpose

# Clean up
RUN rm /$(PackageFile)
RUN rm -rf /var/lib/apt/lists/*

# Run when the container launches
RUN ./opt

CMD ["/bin/bash"]

## Build
# docker build -t openpose/openpose .
VM - Docker (IV)

Image Definition → Developed Code → Evaluation + Debug → Requirements → Adaptions → Result Image

Result Image → Evaluation Settings → Evaluation → Modifikations → Results
Example - Openpose [6]
• Virtual machines provide different granularity of isolation and abstraction from other guests and hardware (GPU)

• Virtual machines provide clean, easy to share, reproducible, scalable and fast environments for development, verification and evaluation of algorithm and systems

• Can prevent unneeded debugging

Thank you for your attention.
Any questions?
References

2. https://www.xenproject.org/
4. https://www.docker.com/