MXAN V1.0
A technical introduction to MXAN code and running environment

Nico Sanna
Sept 25th, 2019

MXAN: Three Dimensional Structures for Metal Sites in Condensed Phases and in Catalysts
Summary

- MXAN workflow
- Intro to docker technology
- Docker runtime installation
- How is made? Details about our MXAN V1.0
- Test runs
- Q & A
MXAN workflow

- A new F90 code
- Based on Intel compiler and libraries (MKL)
- Uses MINUIT from CERN
- Parallel w/ OpenMP
- Pre-release
- Molecular Dynamics
MXAN workflow

Exp. X spectrum → INPUT → Theoretical Spectrum & Best Fit (MINUIT) → Experimental spectrum → Molecular geometry
HPC+BDA+AI model

**Tier 0**
- The “Things”
  - Primarily analog data sources
  - Devices, gadgets, machines, tools, people, cars, animals, clothes, etc.

**Tier 1**
- Sensors/Actuators (Wired, Wireless)

**Tier 2**
- Internet Gateway, Data Acquisition Systems
  - (Data Aggregation, Analog to Digital Conversion)

**Tier 3**
- Edge IT
  - (Analytics, Pre-processing)

**Tier 4**
- Data Center / Cloud
  - (Analytics, Management, Archive)

**4 Tier End-to-End IoT Solutions Architecture**

---

**“Spectrum of Insight”**
- Trade-off: Faster Insight vs. Deeper Insight
- Compute and Analytics shifts to the left
HPC+BDA+AI model

CONTAINER 1

Applications
CUDA Toolkit
Container OS User Space

Docker Engine
CUDA Driver
Host OS

NVIDIA GPUs
Server

CONTAINER N
Case study: Docker

- Lightweight, open and secure container-based virtualization
  - Containers include the application and all of its dependencies, but share the kernel with other containers
  - Containers run as an isolated process in userspace on the host operating system
  - Containers are also not tied to any specific infrastructure
Docker engine

- **Docker Engine**: client-server application composed by:
  - A server, called daemon process
  - A REST API which specifies interfaces that programs can use to control and interact with the daemon
  - A command line interface (CLI) client
Docker architecture

- Docker uses a client-server architecture
  - The Docker *client* talks to the Docker *daemon*, which builds, runs, and distributes Docker containers
  - Client and daemon communicate via sockets or REST API
Docker image

- Layered image
  - Each image consists of a *series of layers*
  - Docker uses *union file systems* to combine these layers into a single unified view
    - Layers are stacked on top of each other to form a base for a container’s root file system
    - Based on the *copy-on-write* (COW) principle

- Layering pros
  - Enable layer reuse, installing common layers only once and saving bandwidth and storage space
  - Manage dependencies and separate concerns
  - Facilitate software specializations
How to? Let’s start and install DOCKER runtime on your PC or Mac

- Go to Docker Hub
  - www.docker.com
- Register
- Download installer
- Install it
- Setup
- READY!
How to? Let’s start and install DOCKER runtime on your PC or Mac

- Go to Docker Hub
  www.docker.com
- Register
- Download installer
- Install it
- Setup
- READY!
How to? Let’s start and install DOCKER runtime on your PC or Mac

- Go to Docker Hub
  - www.docker.com
- Register
- Download installer
- Install it
- Setup
- READY!

NOTE: Set Intel VT to ON on your BIOS. Hyper-V virtualization will be set ON (Windows 10 - all versions but home - needed).
How to? Let’s start and install DOCKER runtime on your PC or Mac

▶ Go to Docker Hub
www.docker.com
▶ Register
▶ Download installer
▶ Install it
▶ Setup
▶ READY!

NOTE: When installing select LINUX containers.
How to? Let’s start and install DOCKER runtime on your PC or Mac

- Go to Docker Hub [www.docker.com](http://www.docker.com)
- Register
- Download installer
- Install it
- Setup
- READY!
How to? Let’s start and install DOCKER runtime on your PC or Mac

- Go to Docker Hub [www.docker.com](http://www.docker.com)
- Register
- Download installer
- Install it
- Setup
- READY!

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
```bash
docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
How to run MXAN container on your PC or Mac

docker pull 193.205.145.87:4043/mxan2018:b2

**Windows**

docker run --rm -v"E:\work":/work 193.205.145.87:4043/mxan2018:b2 mxan_run Ni 2

**Linux or Mac**

docker run --rm -v$PWD:/work 193.205.145.87:4043/mxan2018:b2 mxan_run Ni 2
PS E:\work> docker run --rm -v "E:\work":/work 193.205.145.87:4043/mxan2018:b2 mxan_run Ni 2
... MXAN2018 JOB START @ Tue Sep 24 15:38:48 UTC 2019
==================================================================================================
Running your job with inputs from /work/Ni
  total 28
  drwxrwxrwx 2 root root 4096 Sep 24 15:29 ..
  -rwxr-xr-x 1 root root 4682 Sep 2 13:29 COMMAND.MIN
  -rwxr-xr-x 1 root root 844 Sep 2 13:29 DATA.Ni22
  -rwxr-xr-x 1 root root 5852 Sep 2 13:29 ni_exp.dat
==================================================================================================
Node     : a88e01189002
Job name : Ni
Scratch dir :
Output in : Ni.out.a88e01189002
No. threads : 2

MXAN2018 script is run with this command line:

/MXAN_2018R3105/mxan2018_intel

real    30m6.695s
user    28m41.770s
sys     0m12.580s
... MXAN2018 JOB STOP @ Tue Sep 24 16:08:55 UTC 2019
==================================================================================================
Docker CLI useful commands at runtime

**Docker process status**

```
docker ps -a
```

**STOP a running container**

```
docker stop <CONTAINER ID>
```

**REMOVE a stopped container**

```
docker rm <CONTAINER ID>
```

**LOGIN into a running container**

```
docker exec -ti <CONTAINER ID> /bin/bash
```
Thank you!
Grazie!

Maurizio Benfatto
Elisabetta Pace
Nico Sanna
Giovanni Chillemi
Cristiano Padrin