Introduction to Docker containers

containerized UM-Bridge models/benchmarks

A. Reinarz[1] (Department of Computer Science)
December 2023

[1] anne.k.reinarz@durham.ac.uk

Agenda

Goal:

Introduce Docker containers and how they are used in UM-Bridge to provide access to models and benchmarks.

Agenda

Goal:

Introduce Docker containers and how they are used in UM-Bridge to provide access to models and benchmarks.

This Talk covers:

- Introduction to Docker Containers

Agenda

Goal:

Introduce Docker containers and how they are used in UM-Bridge to provide access to models and benchmarks.

This Talk covers:

- Introduction to Docker Containers
- How docker containers are used in UM-Bridge

Agenda

Goal:

Introduce Docker containers and how they are used in UM-Bridge to provide access to models and benchmarks.

This Talk covers:

- Introduction to Docker Containers
- How docker containers are used in UM-Bridge
- How to set them up

Agenda

Goal:

Introduce Docker containers and how they are used in UM-Bridge to provide access to models and benchmarks.

This Talk covers:

- Introduction to Docker Containers
- How docker containers are used in UM-Bridge
- How to set them up
- How to run them

Agenda

Goal:

Introduce Docker containers and how they are used in UM-Bridge to provide access to models and benchmarks.

This Talk covers:

- Introduction to Docker Containers
- How docker containers are used in UM-Bridge
- How to set them up
- How to run them
- Overview of UM-Bridge models and benchmarks library

Docker containers

UM-Bridge

Brief Summary

UM-Bridge is an abstract interface between UQ methods and models. It mimicks the mathematical "model interface" required by many UQ methods, essentially treating a model as a function mapping vectors onto vectors.

UM-Bridge

Brief Summary

UM-Bridge is an abstract interface between UQ methods and models. It mimicks the mathematical "model interface" required by many UQ methods, essentially treating a model as a function mapping vectors onto vectors.

UM-Bridge benchmark library

Models and benchmark implementations support the UM-Bridge interface and are provided as containers for portability and ease of use.

Why use docker?

- **Easy installation:** The installation process for any model or benchmark comes down to a single docker command.

Why use docker?

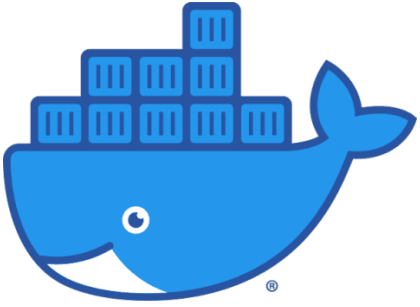
- **Easy installation:** The installation process for any model or benchmark comes down to a single docker command.
- **System independent:** The installation/running of a model/benchmark no longer relies on assumptions on the users OS or installed packages/software or versions.

Why use docker?

- **Easy installation:** The installation process for any model or benchmark comes down to a single docker command.
- **System independent:** The installation/running of a model/benchmark no longer relies on assumptions on the users OS or installed packages/software or versions.
- **Easy access to HPC resources:** UM-Bridge allows for easy access to HPC-scale performance in cloud computing environments. Model containers can be run on any kubernetes cluster without modification.

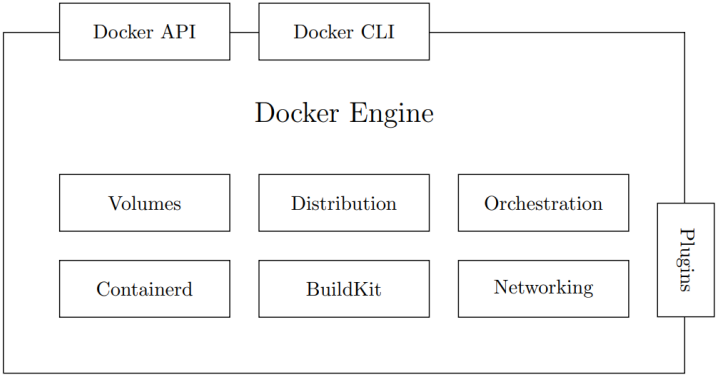
  UM-Bridge provides a reference kubernetes configuration which takes care of load balancing on large-scale clusters. The UQ software can then make use of parallel model instances simply by making concurrent model calls.

What is docker?



- A Docker container image is a lightweight, standalone, executable package of software.
- The container packages code, runtime, system tools, system libraries and settings.

What is docker?



Useful Links

- For installation guidelines for most operating system look at: https://docs.docker.com/engine/install/ubuntu/
- Docker has an extensive documentation at: https://docs.docker.com/get-started/
- The tutorials at https://diveintodocker.com/ may also be interesting

Hello World

Let's test it

```
docker run hello-world
```

Hello World

Let's test it

```
docker run hello-world
```

- If the `hello-world` image does not exist locally it will be pulled from dockerhub
- Allows you to test your docker installation

A simple container

Let's try it for another simple example

```
docker run -it ubuntu
```

- This is running new container, we are simply pulling the latest ubuntu image from dockerhub

A simple container

Let's try it for another simple example

```
docker run -it ubuntu
```

- This is running new container, we are simply pulling the latest ubuntu image from dockerhub
- `-it` is shorthand for `-i -t`

A simple container

Let's try it for another simple example

```
docker run -it ubuntu
```

- This is running new container, we are simply pulling the latest ubuntu image from dockerhub
- `-it` is shorthand for `-i -t`
- `-i` or `--interactive` starts an interactive docker container. We connect to the container's `stdin`.

A simple container

Let's try it for another simple example

```
docker run -it ubuntu
```

- This is running new container, we are simply pulling the latest ubuntu image from dockerhub
- `-it` is shorthand for `-i -t`
- `-i` or `--interactive` starts an interactive docker container. We connect to the container's `stdin`.
- `-t` gives us access to the terminal in the docker container

A simple container

Inside the container we can run ubuntu commands:

```
apt update && apt install -y figlet
figlet hello world
```

- We have installed the package necessary to run the hello world command from the previous container and

A simple container

Inside the container we can run ubuntu commands:

```
apt update && apt install -y figlet
figlet hello world
```

- We have installed the package necessary to run the hello world command from the previous container and
- run the same command ourselves

The Dockerfile

The corresponding Dockerfile would look like this:

Dockerfile

```
FROM ubuntu

RUN apt update && apt install -y figlet

CMD figlet hello world
```

Running the new container

To build an image from a Dockerfile running

```
docker build -t image-name .
```

- `-t` means tag and gives the image a name rather than meaning terminal as in the run command

Other basic Docker CLI commands

To see current images

```
docker image ls
```

Other basic Docker CLI commands

To see current images

```
docker image ls
```

To remove specific images

```
docker image rm name
```

Creating your own Dockerfile

Some commonly used docker commands:

- `FROM` sets the base image
- `RUN` executes linux commands while building
- `CMD` executes linux commands while running
- `WORKDIR` sets the working directory
- `ENV` sets environment variables
- `COPY` copies data into the image

UM-Bridge Workflow

To create an UM-Bridge server, we

- Create container image with our application
- Create an UM-Bridge server for communication and place it into the Docker container

UM-Bridge Workflow

To create an UM-Bridge server, we

- Create container image with our application
- Create an UM-Bridge server for communication and place it into the Docker container

To run our new server

- Run the container with that image (we will need to specify ports)
- Send requests for model evaluations via UM-Bridge
- If we need to tweak the model, update the Dockerfile and rebuild

UM-Bridge: Client

Connect to model

```
import umbridge
model = umbridge.HTTPModel("http://localhost:4242", "forward")
```

UM-Bridge: Client

Connect to model

```
import umbridge
model = umbridge.HTTPModel("http://localhost:4242", "forward")
```

Display input / output dimensions

```
print(model.get_input_sizes())
print(model.get_output_sizes())
```

UM-Bridge: Client

Connect to model

```
import umbridge
model = umbridge.HTTPModel("http://localhost:4242", "forward")
```

Display input / output dimensions

```
print(model.get_input_sizes())
print(model.get_output_sizes())
```

Evaluate model

```
print(model([[0.0, 10.0]]))
```

UM-Bridge: Client

Connect to model

```
import umbridge
model = umbridge.HTTPModel("http://localhost:4242", "forward")
```

Display input / output dimensions

```
print(model.get_input_sizes())
print(model.get_output_sizes())
```

Evaluate model

```
print(model([[0.0, 10.0]]))
```

Optionally, pass configuration options

```
print(model([[0.0, 10.0 ]] , {"level": 0}))
```

UM-Bridge: Server

Defining a model class

```
TestModel(umbridge.Model):
    def get_input_sizes(self):
        # Number and dimensions of input vectors
        return [1]
    def get_output_sizes(self):
        # Number and dimensions of output vectors
        return [1]
    def call(self, parameters, config = {}):
        output = parameters[0][0]*2 # Do something with the input
        return [[output]]
    def supports_evaluate(self):
        return True
```

UM-Bridge: Server

Defining a model class

```
TestModel(umbridge.Model):
    def get_input_sizes(self):
        # Number and dimensions of input vectors
        return [1]
    def get_output_sizes(self):
        # Number and dimensions of output vectors
        return [1]
    def call(self, parameters, config = {}):
        output = parameters[0][0]*2 # Do something with the input
        return [[output]]
    def supports_evaluate(self):
        return True
```

Serve model via HTTP:

```
testmodel = TestModel()
umbridge.servemodel(testmodel, 4242)
```

Simple Example with UM-Bridge

- Grab the minimal server and copy it into the Dockerfile

Simple Example with UM-Bridge

- Grab the minimal server and copy it into the Dockerfile
- Run the UM-Bridge server instead of our figlet command

Simple Example with UM-Bridge

- Grab the minimal server and copy it into the Dockerfile
- Run the UM-Bridge server instead of our figlet command
- When running we need to specify the ports with `-p 4242:4242`

Simple Example with UM-Bridge

- Grab the minimal server and copy it into the Dockerfile
- Run the UM-Bridge server instead of our figlet command
- When running we need to specify the ports with `-p 4242:4242`
- Try this yourself in the tutorial

UM-Bridge Models/benchmarks

UM-Bridge Models & Benchmarks Library

Goal

Maintain a curated set of ready-to-run benchmark problems and models which can be used to reproducibly compare UQ methods.

UM-Bridge Models & Benchmarks Library

Goal

Maintain a curated set of ready-to-run benchmark problems and models which can be used to reproducibly compare UQ methods.

- **Automated testing** and building via Github actions.
- Partially-automated **documentation**.
- **Ease of use:** Each model or benchmark can be accessed from any supported language through a simple function call

UM-Bridge Models

Mathematically, an UM-Bridge model is a function mapping parameter vectors onto model output vectors, supporting some of the following: evaluation, gradient evaluation, Jacobian action, or Hessian action.

UM-Bridge models may be used in multiple benchmarks, e.g. there may be an infererence problem and a propagation problem in the benchmarks library using the same model.

UM-Bridge Models

Name	Short description
Euler-Bernoulli beam	Deformation of an Euler-Bernoulli beam with a spatially variable stiffness parameter
L2-Sea	Total resistance estimation of the DTMB 5415 destroyer-type vessel at model scale by potential flow
Tsunami	Propagation of the 2011 Tohoku tsunami modeled by solving the shallow water equations.
Composite material	Elastic deformation of an L-shaped composite part with random wrinkles.
Tritium Desorption	Microscopic transport of tritium through fusion reactor materials using the Foster-McNabb equations.
Agent based disease transmission	Transmission of disease in a heterogenous population using EMOD, a stochastic agent based disease transmission model.
Membrane model	Determines the deformation of a membrane for a fixed right hand side given the stiffness values on an 8×8 grid that makes up the membrane.

UM-Bridge Benchmarks

Uncertainty quantification benchmarks are identical to models regarding implementation of both server and client. They only differ in that they fully define a UQ problem rather than just a forward model.

They are based on forward models of the previous section.

- The inference benchmarks define a complete Bayesian posterior density, including prior, likelihood and observations. The goal is then to determine the posterior distribution or quantities derived from it.
- The propagation benchmarks that are defined by a model and a specific distribution probability distribution. The goal is then to find the corresponding distribution of the model output, or a quantity derived therefrom.
- The optimization benchmark defines an optimization problem with a range of parameters under certain constraints.

UM-Bridge Inference Benchmarks

Name	Short description
Analytic densities	Infers the PDF of various analytic function.
Membrane model	Infers the PDF of stiffness values of a membrane from measured deformation data.
Tritium Diffusion Posterior	Computes the (unnormalised) posterior density of the input parameters given by experimental data.
Disease transmission model	Agent based model of disease transmission in an entirely susceptible population with correlation between disease acquisition and transmission. Provided problem is to infer the disease parameters that result in 40% of the population infected at the end of the outbreak.
Deconvolution problem	Defines a posterior distribution for a 1D deconvolution problem, with a Gaussian likelihood and four different choices of prior distributions with configurable parameter.
Computed Tomography	Defines a posterior distribution for a 2D X-ray CT image reconstruction problem problem, with a Gaussian noise distribution.

UM-Bridge Inference Benchmarks

Name	Short description
Heat inverse problem	defines a posterior distribution for a 1D heat inverse problem, with a Gaussian likelihood and Karhunen–Loève (KL) parameterization of the uncertain parameters.
Beam Inference	Bayesian inverse problem for characterizing the stiffness in an Euler-Bernoulli beam given observations of the beam displacement with a prescribed load.
Tsunami Source	Infer parameters describing the initial displacements leading to the tsunami from the data of two available buoys located near the Japanese coast
Poisson	Estimates a spatially varying diffusion coefficient in an elliptic PDE given limited noisy observations of the PDE solution.
p-Poisson	Estimates a two dimensional flux boundary condition on the bottom of a three dimensional domain with nonlinear p-Poisson PDE.

UM-Bridge Propagation Benchmarks

Name	Short description
Euler-Bernoulli beam	Models the effect of uncertain material parameters on the displacement of an Euler-Bernoulli beam with a prescribed load.
Genz	Multi-dimensional functions for testing integration and surrogate methods.
L2-Sea UQ	Forward UQ of the total resistance in calm-water conditional to operational and geometrical uncertain parameters.

UM-Bridge Optimisation Benchmarks

Name	**Short description**
L2-Sea UQ	Constraint deterministic optimization problem for the total resistance reduction in calm-water at fixed speed.

Contributing to the benchmark library

- The benchmark library should not be a static resource, it is a community building project and we welcome contributions!
- If you have a model/benchmark with features not found in existing benchmarks please add it
- We're happy to discuss/help
- Finally, to add it simply make a pull request on the repository

Contributing to the benchmark library

Models consist of at least:

- A Dockerfile containing the UM-Bridge server
- A README following a pre-defined structure (we provide a script that auto-generates this file so that you only need to fill in the description)
- A corresponding benchmark which uses this forward model

Contributing to the benchmark library

Models consist of at least:

- A Dockerfile containing the UM-Bridge server
- A README following a pre-defined structure (we provide a script that auto-generates this file so that you only need to fill in the description)
- A corresponding benchmark which uses this forward model

They may also have

- Additional required files, e.g. code, input data
- More than one benchmark, e.g. a corresponding inference and propagation benchmark have been defined

Questions?

Tutorial

- We will continue where we left off with the tutorial
  https://um-bridge-benchmarks.readthedocs.io/en/docs/tutorial.html
- Go to Section 5 and try to create your own model server
- If you don't have any application you want to wrap in a container you can use the minimal server you created in the earlier session.