Running experiments using Singularity containers

Ionut Moraru

King's College of London

October 24, 2018

Overview

1. Singularity
   - What it is?
   - Example
2. Experiments tool
   - How to use
   - Running experiments
3. Running on cluster

What is Singularity?

- It is a software container that makes sharing your software a lot easier (no more need to install all the missing dependencies);
- Singularity was made for use in scientific purposes, having a very small overhead;

What is Singularity?

- It is a software container that makes sharing your software a lot easier (no more need to install all the missing dependencies);
- Singularity was made for use in scientific purposes, having a very small overhead;
- It offers seamless (debatable) integration for a complex project;
- It creates an image that can run on any machine that has Singularity installed;
- Singularity also allows you to leverage the resources of whatever host you are on - lab machine or HPC.

How it works?

Let's say you have a planner installed on you machine. To use singularity you need to create in the Planners folder a Singularity file - intuitively named Singulary, which is split in:

- Setup - the bootsrap of an empty container (for our example its an Ubuntu machine);
- Post - now after the *empty* container has been created, it will install all the dependencies and will build the planner;
- Runscript - this part is being called each time the container is called to solve something, like when you execute the planner;
- Label - this part is to add different metadata for the container.

How it works?

Let's say you have a planner installed on you machine. To use singularity you need to create in the Planners folder a Singularity file - intuitively named Singulary, which is split in:

- Setup - the bootsrap of an empty container (for our example its an Ubuntu machine);
- Post - now after the *empty* container has been created, it will install all the dependencies and will build the planner;
- Runscript - this part is being called each time the container is called to solve something, like when you execute the planner;
- Label - this part is to add different metadata for the container.

An example of this type of file is in *demoPlanner/Singularity*

Coding it!

Let's now create an image file of the demoPlanner (it's an instance of FastDownward running LMCut).

```
sudo singularity build planner.img \
    demoPlanner/Singularity
```

This command will create the singularity image[1]. This should take a bit...

[1] If you are using a Mac, you should run the command inside of the vagrant that you have at the end of the tutorial sent in the installation email)

Coding it!

Awesome, now we *should* have an image of our planner. Next we will make it execute the planner:

Coding it!

Awesome, now we *should* have an image of our planner. Next we will make it execute the planner:

```
RUNDIR="$(pwd)/rundir"
DOMAIN="$RUNDIR/domain.pddl"
PROBLEM="$RUNDIR/instance-1.pddl"
PLANFILE="$RUNDIR/sas_plan"

singularity run -C -H $RUNDIR planner.img \
    $DOMAIN $PROBLEM $PLANFILE
```

Coding it!

Awesome, now we *should* have an image of our planner. Next we will make it execute the planner:

```
RUNDIR="$(pwd)/rundir"
DOMAIN="$RUNDIR/domain.pddl"
PROBLEM="$RUNDIR/instance-1.pddl"
PLANFILE="$RUNDIR/sas_plan"

singularity run -C -H $RUNDIR planner.img \
    $DOMAIN $PROBLEM $PLANFILE
```

Right now you should have just executed the planner inside the Singularity container.

How to take advantage of this?

The runPlanningTool

What we implemented was a short program in which we can automatically run planners with a set of benchmarks. Feature list:

- Easy way to run multiple planners and domains with very little hassle;
- It validates all the plans using VAL at the end of executing one;
- At the end of executing all the plans it has a results scrapper;

The runPlanningTool

What we implemented was a short program in which we can automatically run planners with a set of benchmarks. Feature list:

- Easy way to run multiple planners and domains with very little hassle;
- It validates all the plans using VAL at the end of executing one;
- At the end of executing all the plans it has a results scrapper;
- It has only a couple of bugs - but they always appear at the beginning of the execution so that's good.

How to use it 1/2

Adding a planner[2]:

- Add a planner inside of *runPlanningTool/planners* with a Singularity file inside of it
- Now specify the planner inside of the *runPlanningTool/files* folder

```
# Planner ID | repo url | planner folder
  OPTIC-Base ,           , OPTIC-Base
```

[2] All of this is inside of the runPlanningTool folder

How to use it 2/2

Adding domain and problem files:

- Add which domains and problems you want to run the experiments with in */benchmarks*
- Add them similar to the planner inside of the *runPlanningTool/files*

```
# DomainID |  folder  | domain file | problem file | domain folder | problem folder | lb | up | b
  AGRICOLA , agricola , domain.pddl , p01.pddl     ,               ,                , 0  , 0  , 0
```

Executing

For our tutorial please run the following command[3]:

```
./run_benchmarks.py -ipc2018
```

[3] From the /runPlanningTool folder

Executing

For our tutorial please run the following command[3]:

```
./run_benchmarks.py -ipc2018
```

Now your computer should executing all the IPC 2018 benchmarks on our planner PlanningPDBs (4 problems away from winning btw).

[3] From the /runPlanningTool folder

Running experiments on a cluster

Demo

Ganxie nin de guanzhu!