TaskFarm hands-on

Kaveh Razavi Thilo Kielmann

Contrail summer school, July 25th 2012

1 Starting the TaskFarm service

For this hands-on session we will use the TaskFarm service of ConPaaS installed in the cluster computer of the Vrije Universiteit in Amsterdam. The cluster is heavily firewalled, so to access it you must configure your Web browser to use a SOCKS tunnel located at `130.37.30.92` on port 80. This can be configured in the chrome browser by pointing your browser to `chrome://settings/advanced` and then selecting the *Change Proxy Settings*.

You can then access ConPaaS by directing your browser to `10.100.123.45:9999`. Create an account and login to ConPaaS.

Create a TaskFarm service and wait for TaskFarm's manager VM to startup. Follow the instructions on the next section when the service is started.

2 Executing a bag-of-tasks with TaskFarm

We have prepared a high performance computing (HPC) problem for you to execute using the TaskFarm service. First, we are going to describe the problem. Then, we describe the general requirements for the TaskFarm service to execute a bag-of-tasks while taking a look at our HPC use-case. In the next section, we give you some technical details on how to prepare TaskFarm to execute our HPC use-case.

2.1 The DACH challenge [1, 2]

In the DACH challenge a large distributed database of scientific data, gathered by the Subaru telescope in Hawaii, has to be searched to find new and unknown supernova candidates. A supernova is a phenomenon in which a star explodes in a spectacular manner, causing a very large amount of light to be emitted. To find the candidates, two images of the sky taken at a time interval are compared.

2.2 The HPC program

The first step in executing HPC tasks is making sure that the agents are setup properly to execute the program code. This means that the required libraries and program code need to either:

1. Have already been installed
2. Be installed as part of agent contextualization
3. Read/executed on a remote filesystem

Allowing service-specific contextualization (option 2) is currently under development in ConPaaS. Thus, in the current version the program code needs to be either pre-installed or read/executed from a remote file system.

For the DACH challenge, we will put the program executables, which are compiled with static libraries, on a remote file system which is mounted on a pre-defined location as part of contextualization. TaskFarm service allows you to (optionally) specify the location of this remote file system as part of service configuration. More details to be followed in section 3.1.

2.3 The input/output data

HPC tasks usually execute on a set of given data. This data which depending on the program can vary in size needs to also be accessible by agents. There are two ways to make the data available to the agents:

1. Copy in the data to each agent
2. Mount a remote file system which holds the data

The future versions of ConPaaS will allow for copying the data inside the agents as part of service-specific agent contextualization. However, we are and will be using the second option for data input. You have a chance to let us know why you think we have made this choice at the end of this session.

HPC tasks also usually output the result of their computation on a set of files. As you have probably already guessed the similar methods can be used to handle data output. More details to be followed in section 3.1.

For the DACH challenge, the input data is located in a shared distributed filesystem which is accessible by TaskFarm agents. The output data is also written to this filesystem. You should provide the location of this remote file system as part of service configuration. More details to be followed in section 3.1.

3 Solving the DACH challenge with the TaskFarm service

In this section, we describe a step by step instruction to solve the DACH challenge with the TaskFarm service. This shows that the TaskFarm service is already a fine platform for executing HPC applications on the cloud.

3.1 Prepare the remote file system

The current version of the TaskFarm service uses XtreemFS [3] for remote file storage. XtreemFS is an open source distributed and replicated file system for the cloud. We are planning to support a variety of remote file systems using the SAGA API in the future releases of TaskFarm service.

We have prepared an XtreemFS volume for each group of students. Your XtreemFS location is `exp001.das4.cs.vu.nl/groupXY` where XY is the two digits of your assigned group number. We have pre-populated the input data and binaries for you to avoid wasting time on large uploads.

To prepare your volume follow these steps:

- Mount your XtreemFS volume on a local folder

```
sudo mkdir /root/mount; sudo mount.xtreemfs exp001.das4.cs.vu.nl/groupXY /root/mount;
```

- Check that the input data is already populated. The input files are in the *input* folder and are a set of images in *fits* [5] format.

```
sudo ls /root/mount/input
```

- Check that the input data is already populated. The input files are in the *bin* folder

```
sudo ls /root/mount/bin
```

- Create the output folder

```
sudo mkdir /mnt/xtreemfs/output
```

3.2 Prepare the bag-of-tasks file

A bag-of-tasks file is a list of independent tasks to be executed by the TaskFarm service. It contains the actual program name along with necessary arguments that possibly include the location of input and output files. For your convenience, we have created a simple script for you which takes the XtreemFS mount-point and outputs a bag-of-tasks file. Assuming your XtreemFS volume is going to be mounted on

the `/mnt/xtreemfs`, the following command will create your bag-of-tasks file (/home/contrail/dach.bot):

```
cd; sudo ./generate_bag.sh /root/mount /root/mount /home/contrail/dach.bot; sudo chown contrail /home/contrail/dach.bot
```

Feel free to look into the bag generation script as well as the generated bag-of-tasks file.

3.3 Configuring TaskFarm's frontend

In TaskFarm's frontend webpage, select the bag-of-tasks file that you just generated and type in your XtreemFS location. Congratulations! You now have the service configured. You can simply start the sampling phase and choose your desired schedule right after it. Sampling should take about ten minutes. The total execution-time should take about thirty minutes if you select the most expensive/fastest schedule (which we suggest you do!). You can use this time to answer the questions of section 4.

3.4 The final solutions

Once the service has finished the execution, you can see the final cost of the execution. Terminate the service. We now want to take a look at the results of the computation:

```
sudo ls /root/mount
```

The result of the comparison of each two images are in the folder */root/mount/output/IMG1.IMG2*. The possible candidates are in the *result* file and a mask of the original image with this result is in the *fits* file. You can visualize this file with a NASA provided software for viewing *fits* files [6]. You can try doing so if you have some extra time left after you are done with all the questions.

Do not forget to unmount the XtreemFS volume once your done:

```
sudo umount /root/mount
```

4 Questions

(a) ConPaaS is realized around a standard Linux image with the required applications for each service pre-installed. For example, the PHP service has the PHP language runtime already available.

Question: Why is TaskFarm a different service compared to other ConPaaS services in this manner?

(b) We mentioned that the TaskFarm service needs the required libraries and program code to be installed before executing the given tasks.

Question: In section 2.2, we discussed three possible ways to get the HPC program into the agents. What are the pros and cons of each method? Can you think of other possibilities?

(c) Imagine you are a ConPaaS administrator and you are given a ConPaaS *incompatible* VM image with the program code installed.

Question: Can you think of a way (ways) to use the provided VM image without completely porting ConPaaS to it?

(d) In section 2.3, we mentioned that the TaskFarm service needs the input data of the HPC tasks to be present to its agents and the outputs of task executions need to be delivered to the user.

Question: We discussed two possible methods to handle data input/output. Can you point out the pros/cons of each? What do you think was the main reason that we opted for using the remote file

system approach?

References

[1] The First International Data Analysis Challenge for Finding Supernovae. http://www.cluster2008.org/challenge/

[2] Ibis wins first prize at DACH 2008. http://www.cs.vu.nl/ibis/awards.html

[3] F. Hupfeld, T. Cortes, B. Kolbeck, E. Focht, M. Hess, J. Malo, J. Marti, J. Stender, E. Cesario. "XtreemFS - a case for object-based storage in Grid data management". VLDB Workshop on Data Management in Grids. In: Proceedings of 33th International Conference on Very Large Data Bases (VLDB) Workshops, 2007.

[4] Simple API for Grid Applications. http://www.saga-project.org/

[5] FITS Data Format. http://heasarc.nasa.gov/docs/heasarc/fits.html

[6] Fv: The Interactive FITS File Editor. http://heasarc.nasa.gov/docs/software/ftools/fv/