Distributed Emergent Software Systems – Experiments Replication

September 2nd, 2016

1 Introduction

Welcome and thank you for using our distributed emergent software system framework prototype. This prototype was described in the paper *"Experiments with a Machine-centric Approach to Realise Distributed Emergent Software Systems"* submitted to **ARM workshop 2016**. This manual contains instructions to execute the framework with the available web server and load balancer. This document also describes the steps to replicate the results described in the paper.

Please, download and install Dana platform[1]. Feel free to drop us a line in the forum if you have any questions, comments or suggestions. Also, consult Dana's Programming Guide[2] for instructions to properly install Dana's toolchain. Dana can be executed in all supported Operating Systems (currently it supports Windows, Ubuntu, Mac OS X and Raspberry Pi). The framework, however, is suggested to be executed on Ubuntu.

2 Project Structure

The framework, the web server code and load balancer code are located in different folders. The web server main components are located in *request*, *http*, *cache* and *compression* inside of *web_server* folder. The main framework components (composer, monitor and learner) are presented in *composition*, *monitoring* and *learning* inside of *meta_systems* folder. Finally, the load balancer main components are located in *scheduler*, *special_scheduler* and *request* inside of *load_balancer* folder.

The folder structure of each project follows that of all Dana projects. The *resources* folder contains all of the interfaces for the components. The implementation of the interfaces are provided in the folders outside of *resources*. For example, the interface that defines Composer's functionality is in *resources/composition*, in `MetaComposer` interface. The components are located in the folders with the same name as the interface's folders outside *resources*, in this example *composition/MetaComposer.dn*.

[1] http://www.projectdana.com

[2] http://www.projectdana.com/dana/guide/doku.php

3 Executing Target Applications

The first step to execute the code is to compile. In order to compile the entire project execute *makeall* in the project's root folder. Please, make sure Dana's toolchain is properly installed and working.

To properly execute the prototype, some additional programs have to be installed. The information generated in the components are stored in a database, so make sure MySQL[3] database is installed. Also, to have access to the Perl scripts to generate HTML files for the variation pattern[4] and enable Dana components access to the database, please, install Perl's[5] toolchain.

3.1 Web Server

In order to execute only one configuration of the web server, without perception and learning, compile all components in *web_server* folder (using the command: `$ dnc . -sp ../meta_system/` in *web_server* folder) and execute `$ dana -sp ../meta_system/ WebServer` in a command prompt in the *web_server* folder. To verify if it is running, go to a web browser and connect to 'http://localhost:2012/'. The *index.html* file (located in *htdocs*) should be loaded and presented as result.

By executing an instance of the web server in this manner, the primary web server configuration will be executed. To change this configuration to any other web server configuration available go to *http* and change the .manifest file. This file contains, in a JSON syntax format, the component, in case an interface has multiple implementations, that Dana VM will load. To choose any other configuration, change the name of the component to any other component name present in the *http* folder. For example, if the `HTTPHandlerCGZ` is selected as primary and you want to change it to the component that only implements cache but not compression, change it to `HTTPHandlerC`, and so on.

All available web server's html pages are located in *htdocs* in *web_server* folder. Any file dropped in *htdocs* will be found and sent to the client upon request by any executing configuration of the web server.

3.2 Load Balancer

Similarly to the web server, the load balancer is located in $load_balancer$ folder. To only compile the load balancer enter load balancer's folder and execute the following command: `$ dnc . -sp "../meta_system/;../web_server/"`. In order to run the load balancer executes: `$ dana -sp "../meta_system/;../web_server/ LoadBalancer`. Please, make sure to execute at least two instances of the web server , and to configure *server_list*[6] configuration file with port and ip address of the running web servers.

[3] http://www.mysql.com/downloads/

[4] The variation pattern is an artificially created request pattern. We generated 30000 HTML files and for each request the client requests a different file from the 30000 files (more information on that, please refer to the paper).

[5] https://www.perl.org/get.html

[6] The configuration file is located in *load_balancer/scheduler*

Analogous to the web server, the load balancer will start running with the default configuration. In order to change the configuration, locate the .manifest files and replace the component's name to any available component. The .manifest file can be located in the following folders: *load_balancer/scheduler*, *load_balancer/special_scheduler* and in *load_balancer/request*.

4 Replicating Evaluation Experiments

The evaluation was conducted using three machines. The configuration of these machines are described below.

> Two rackmount servers with an Intel Xeon E3-1280 v2 Quad Core 3.60 GHz CPU, 16 GB of RAM, running Ubuntu 14.04. The load balancer was a desktop machine with an i7-4770 3.40 GHz CPU, 6 GB of RAM, running Ubuntu 14.04. The server and client reside on two different subnets of our campus network, in different buildings.

The paper presents two results, the first one referenced as 'Divergent Optimality', and the second, 'Learning Behaviour'. The steps to replicate the findings described in the paper is presented in the subsequent sub-sections.

Note that it is possible to limit the components that will be available for experimentations. In order to do so, copy *.ignore.example* file and rename it to *.ignore*, this file contains a list of components that will be ignored by the Assembly module (so, please, only use the .ignore file when trying to limit the number of architectures that will be exposed to the requests, otherwise, there is no need to create the *.ignore* file).

It is required, however, to limit the number of possible configurations to only 64 to replicate the paper's results. This is due to the amount of time it will take to run the experiments in case all possible configurations for the web servers are available. In this case, where all web servers configurations are available, there will be 7056 unique possible configurations for the entire distributed system (considering 42 for each web server and 4 for the load balancer). The Learning module has a 5 seconds window to expose and collect information for each configuration, thus taking ∼9 hours for the centralised learning strategy to locate the optimal configuration for each workload. Considering two unique workloads (Workload 1 and Workload 2, described in Sec. 4.1), the time the system will spend in the learning exploration phase is ∼18 hours (not considering the exploitation phase of the 'Learning Behaviour', and the 'Divergent Optimality' tests).

4.1 Workloads

To evaluate the system, two workloads were used. The first workload, as described in the paper, consists of one single html file located in *web_server/htdocs* named index.html. This is a text-only html file with 3.9KB size. The second workload consists of a total of 30000 copies of the index.html file with different names, the files will have the same characteristic (content, size and resource-type) with different name, which forces the

system to treat them as complete different files with the same characteristics. The 30000 variations of index.html are not distributed with this project due to size, instead all 30000 have to be created before running evaluation.

In order to generate the requests for the Workload 2 simply execute the Perl script `script_to_create_variations.pl` located inside *web_server/htdocs/* using the command `$ perl script_to_create_ variations.pl`. Note that if you want to delete all of the 30000 files, run the script `script_to_delete_ variation.pl` with the command `$ perl script_to_delete_variation.pl` also located in *web_server/htdocs/*.

4.2 Divergent Optimality

The Divergent Optimality section presented two bar graphs showing the average performance of all available architectures, when subjected to two different workloads. In order to replicate the results, executes two instances of the web server in two different machines with equivalent configurations, and one instance of the load balancer using the component *DistributedAdaptor.dn* located in *meta_system* folder executing the following command:

For the Web Server:

```
$ dana DistributedAdaptor ../web_server/WebServer
```

For the Load Balancer:

```
$ dana DistributedAdaptor -sp ../web_server/ ../load_balancer/LoadBalancer
```

After running the following commands, for two web servers and the load balancer, it is required to run the *ClientRemote.dn* component and the *DistributedMain.dn* component, this last component is responsible to coordinate the tests (it will coordinate the system's configuration and the client pattern). It is important to run the *DistributedMain.dn* on the same machine as the Load Balancer. Execute both by using the following commands:

For the Client:

```
$ dana ClientRemote
```

For the Coordinator:

```
$ dana DistributedMain
```

The client command has to be executed from the *ws_clients* folder, whereas the Coordinator must be executed from *meta_system* folder. Furthermore, it is necessary to run the database module that enables any Dana component access the database. Please run the database broker, only for the Coordinator (not necessary to run it for the web servers). For more information on how to set up the database and run the broker, see Sec. 5. After running all above processes, type "starttest" in the *DistributedMain.dn* executed process, this will trigger tests.

As a final note, it is necessary to make sure that i) all of the components have the right IP addresses to the components they need to exchange information/coordinate and ii) the manifest files for the learning modules are configured. To configure the IP address (i) first make sure that the *DistributedMain.dn* have the Client's machine IP address as well as the Web Servers' and the Load Balancer's addresses. In the *DistributedMain.dn* component in *meta_system* folder, search for comments containing the texts: 'change

IP for Client here', 'change IP for Web Servers and Load Balancer here'. Similarly, in *ClientRemote.dn* in *ws_clients* folder, search for comments: 'change Load Balancer IP here'. Also, make sure the web server's IP addresses are set in *server_list.config* in *load_balancer* folder and search for comments: 'change IP for Coordinator here' to configure *DistributedAdaptor.dn*. To configure the Learning module (ii) is required to edit .manifest file in *meta_system/learning* for all instances of the system running the web servers and load balancer and change (if necessary) the component to "SingleArch", the resulting string should look like the following:

```
{interface: "MetaLearner", component: "SingleArch"}
```

After the Coordinate runs, the results will be in the *meta_system* folder within the text files: *pattern*1.*dat* for Workload 1 and *pattern*2.*dat* for Workload 2; the architectures for both workloads will be in *archlist*1.*dat* and *archlist*2.*dat* respectively.

4.3 Learning Behaviour

The 'Learning Behaviour' presents the results of running the learning approach in two different modes: i) the decentralised (selfsih) mode and ii) the centralised mode. For both modes, however, the .manifest file inside of *learning* folder in *meta_system* has to be reconfigured. To configure the learning for both learning behaviours, edit the .manifest file and change the component from "SingleArch" to "MetaLearner" in all instances of the system running the web servers and the load balancer. The .manifest file should look like the following:

```
{interface: "MetaLearner", component: "MetaLearner"}
```

4.3.1 Centralised Learning Approach

In order to execute the Centralised Learning approach, run the web servers using the *DistributedAdaptor.dn* with the following command:

```
$ dana DistributedAdaptor ../web_server/WebServer
```

Then, execute the database broker (see 5)and the *DistributedLearner.dn* in the same machine. Note that the database broker has to be executed before the the *DistributedLearner.dn*, in order to execute the learner use the following command:

```
$ dana DistributedLearner -sp ../web_server/ ../load_balancer/LoadBalancer
```

After executing the learner module (*DistributedLearner.dn*), executes the clients. You can execute either the Workload 1, or the Workload 2 in any order. However, to replicate the results in the paper, execute the Workload 1, then wait for the system to converge to the optimal configuration, then stop Workload 1 and execute Workload 2. To execute the workloads, use the following commands:

For Workload 1:

```
$ dana Client /
```

For Workload 2:

```
$ dana Client2
```

To collect the results generated by the learning process, use the SQL scripts *monitoring.sql* located in *meta_system* folder, using the following command:

```
$ mysql -u root -p webserver < monitoring.sql > coordinated_learning.dat
```

To collect the architecture's description use the script *architectures.sql* also located in *meta_system* using the command:

```
$ mysql -u root -p webserver < architectures.sql > clearning_archs.dat
```

4.3.2 Decentralised Learning Approach

The Decentralised Learning is to execute one instance of local learning for all target applications, i.e. for the two web servers, and the load balancer. In order to do that, first, execute a database broker (see Sec. 5) for all the instances of the web servers and load balancer (have each of them running in a different machine). To execute a local learning component use the following command:

For the Web Servers:

```
$ dana Main ../web_server/WebServer
```

For the Load Balancer:

```
$ dana Main -sp ../web_server/ ../load_balancer/LoadBalancer
```

After executing the above commands for the Web Servers and the Load Balancer, run one instance of the client, again, this can be the Workload 1 or Workload 2. Use the same commands described in the Centralised Learning approach to run the clients each client. Also, in order to collect the results, use the same SQL scripts described in the previous section. Finally, it is important to point out that for every time each of the tests are run, please make sure to clean the database, dropping the entire 'webserver' database. Otherwise, the data that remains after executing a test will interfere with the learning process and alter the observed results.

5 Database Configuration

To configure the database you have to create a database named `webserver` in a MySQL database on the server machine, using the command `mysql> CREATE DATABASE webserver;` After that, create the database schema by executing the file named `webserver_schema.sql`, located at `database_external/database_schema` folder using the following command: `$ mysql -u root -p webserver < database_external/database_schema.sql`. Note that the previous command to set the database scheme, can be used to delete all of the database data (this can be used every time before starting the tests described in this manual). Also, remember to set the user name and password to access the database

in the `database.config` file located in `database_external` directory. To enable Dana components access to the database, open a new command prompt and execute the Perl script `database_broker.pl`, also in `database_external` folder. To execute the script run the command `$ perl database_broker.pl` in *meta_system/ database_external* folder. It is imperative that `database_broker.pl` script remains running while all tests are running.

6 Final Considerations

Thank you for your interest in our research. We hope you find this manual helpful to get you started with our current version of the distributed emergent software framework. All questions, comments and feedback is greatly appreciated. Contact us on Project Dana's forum (http://www.projectdana.com/fora) or over email (r.rodriguesfilho@lancaster.ac.uk).