The performance of the native Windows 10 Docker application with integrated Unikernel libraries

Large Installation Administration
Project Proposal

Student: Andrey Afanasyev
Andrey.Afanasyev@os3.nl

Supervisors:
Jaap van Ginkel
J.A.vanGinkel@uva.nl

Arno Bakker
Arno.Bakker@os3.nl

Mick Pouw
Mick.Pouw@os3.nl

March 5, 2017

Document Log

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-02-2017</td>
<td>0.1</td>
<td>Draft</td>
</tr>
<tr>
<td>5-03-2017</td>
<td>1.0</td>
<td>Major corrections</td>
</tr>
</tbody>
</table>

Abstract

In May 2016, a beta version of the native Windows 10 Docker application with integrated MirageOS unikernel libraries was presented[1]. A year earlier a deployment of Docker in unikernels environment was publicly demonstrated for the first time. This research will compare performance of the full web application stack deployed on both environments. Both environments will use the equivalent hardware to correctly compare aspects like boot-time and network latencies, memory and storage usage of running web application in the Docker container technology.
Introduction

Docker is an open source containerization platform which allows automatically deployments of applications inside software containers. Since initial release in March 2013, the usage of the Docker’s containerisation technology has explosively grew and gained its popularity in development and production environment[2]. Originally, the Docker was developed for use on Linux based distributions and supports only Linux based applications.

In containerization, which is also known as OS level virtualization, the host and the guests are using the same OS kernel. Each container is running as an isolated process in user space. A containerization engine which is on top of the host operating system manages the containers. Such approach reduces ineffective use of the resources due to the fact that each container only holds an application and its related libraries and binaries.

Decreasing an operating system overhead is a main advantage of using containerization. This advantage provides a huge positive impact on resilience of the applications using containerization technology as a platform in comparison with classical virtual machines.

Due to fact that containers share the same kernel it may create serious security issues. Further, components of an application are difficult to tightly couple in order not to create overhead. The extra layer of abstraction between OS kernel and application components may use more resources then by a hypervisor environment. A set of isolated containers per VM with own operating system does not solve overhead issue. To improve aforementioned robustness issues and spread Docker across the platforms, the collaboration between unikernels and Docker was presented.

Unikernels compiles source code as a specialized single-address-space virtual machine image using operating system libraries, components and functionalities according the requirements of an application [3][4]. A combination of Docker platform and unikernels machine images provide a stronger container isolation. Further, unikernels and Docker in such combination have extremely small size and low-latency boot-time what is perfect to use for single task specialized microservices. Finally, this combination can be managed by a Docker in aspects like image management, networking and storage size[5].

The Docker Toolbox solution is a tool that uses the Oracle Virtual Box to run Docker on Microsoft Windows and Apple Mac OS operating systems. Due to the fact that Virtual Box is not a native OS application, such solution has several drawbacks like deployment and boot speed of a container. To improve this situation in May 2016 the new components based on unikernel libraries were presented in Docker for Mac and Windows deployment packages.

Recently, four components with integrated unikernel libraries were realised. HyperKit is a toolkit for embedding hypervisor capabilities in an Docker application running on MacOS X. DataKit is a toolkit to coordinate processes with a git-compatible filesystem interface in MacOS X or Windows. VPNKit is a networking library based on the MirageOS TCP/IP unikernel stack that translates between raw Ethernet network traffic and their equivalent socket calls in MacOS X or Windows. InfraKit is a Linux-based toolkit for creating and managing declarative, self-healing infrastructure.

This research will compare performance of the native Windows 10 Docker application against an unikernel based Docker application. The comparison is focused on aspects like boot-time and network latencies, memory and storage usage on the equivalent hardware. A full web stack application with equal configuration will be used in both environments. Finally, the gathered results of experiments will be concluded.
Research question

Based on the aforementioned, the following main research question is defined:

What are the performance advantages and disadvantages of having the web full stack Docker application running on the Windows 10 Docker application or the unikernel Docker environment?

To answer the main question, the following sub-questions were deducted:

- What are boot-time latencies in both environments?
- What are memory and storage usage in both environments?
- How does the network perform in both environments?

Related work

Due to gained popularity of the Docker containerization technology it is actively in development. Features like robustness and resilience in cloud environments are the in focus of most researches[6]. The hardware utilization and higher performance computing in scientific applications were deeply researched [7][8]. The common factor in all these researches is that the Docker runs in a Linux environment with a shared kernel.

Unikernels are a promising alternative for application deployment in cloud platforms[9][3][4]. The usage of a container inside a unikernel is a new level of specialization and isolation of containers with clear advantages[5]. Docker technology ported to other platforms (Windows, MacOS X) with an additional level of abstraction that performs less in comparison to the Linux implementation of Docker. To improve this issue a Docker with implemented unikernels libraries for Windows 10 and MacOS X was presented.

This project will focus on finding performance advantages and disadvantages of having the web full stack Docker application running on the native Windows 10 Docker application in comparison with unikernel environment.

Scope

This project will be limited only to the hardware available in SNE labs (B1.23 & C2.154). The set of tests will run on the most recent stable release of Docker for Windows and MirageOS Unikernel OS using XenServer. The Windows 10 will use Hyper-V as native hypervisor for the first test environment and XenServer as hypervisor for MirageOS unikernel test environment.

Method

The research is divided into four main stages.

First stage, mainly focuses on preparation. Searching for information if needed. Selecting a single machine which can be used for installation of different OSes and choosing a set of tools that can measure boot-time, network latency, memory and storage usage.

In second stage the XenServer with Docker container running in MirageOS and full web stack application will be deployed. Wordpress will be used as a full web stack application in Docker container. Measurement tools will be executed to gather statistics.

The third stage is the same as a previous one only Windows 10 will be installed with native Docker application on board. The same full web stack application will be used. Using the same set of measurement tools, statistical data will be gathered.
In the fourth and final stage all conducted results will be analysed, plotted and compared. On the basis of results conclusions will be reported.

Requirements

This research will require a machine which complies with recommended Windows 10 and XenServer hardware requirements. Further, research will require a Python to organize statistical data, provide plots and make some conclusions.

Planning

This project will be executed from Monday 27 of February 2017 till Friday 31 of March 2017.

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>27-02-2017/3-03-2016</td>
<td>Defining topic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gathering related research</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Draft Project proposal</td>
</tr>
<tr>
<td>Week 2</td>
<td>6-03-2017/10-03-2017</td>
<td>Fixing and finalising the project proposal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Creating draft report and presentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gathering related research</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setting up environment to do experiments</td>
</tr>
<tr>
<td>Week 3</td>
<td>13-03-2017/17-03-2017</td>
<td>Execution of the experiment and gathering statistical data.</td>
</tr>
<tr>
<td>Week 4</td>
<td>20-03-2016/24-03-2017</td>
<td>Continue execution of the experiment and gathering data if needed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Analysing gathered data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Plotting data</td>
</tr>
<tr>
<td>Week 5</td>
<td>27-03-2017/31-03-2017</td>
<td>Analysing and comparing the final results</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interpreting and deriving conclusions of acquired results</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Presentation and Finalizing report</td>
</tr>
</tbody>
</table>

Table 1: Planning.

Ethical concerns

All tests will be performed in a controlled and closed test environment. No personal data of third parties will be collected during this research and no active tests targeting back-end tests will be performed. The test data will be generated for the purpose of the experiment. Ethical issues are not expected during this project.

References


