University Of Agder

DAT 304

Preliminary Report

NVIDIA DGX - 1

Author:
Øystein Andreassen
Bendik Egenes Dyrli

Supervisor:
Sigurd Kristian Brinch

May 10, 2018

UNIVERSITETET I AGDER

Contents

List of Tables

Chapter 1

Project Overview

With the advancements, focus and investments on artificial intelligence (AI) and the new Mechatronics Innovation Lab (MIL) at UiA there is a need to distribute the available resources as evenly and effectively as possible while reducing the overhead, and maintain visibility about usage of the GPU clusters.

1.1 Purpose

Over the course of this project the group is primarily aiming to find an effective solution that can automate the creation and monitoring of docker containers to consume resources at the request of students and faculty members, while striving to distribute the resources in as fair a way as possible while still maintaining an acceptable degree of performance. Secondly the group also aims to provide a management interface to track the resource utilization based on username or other identifiable information across multiple docker instances.

Chapter 2

Project Objective

2.1 Goal

The end goal is giving a simple and intuitive interface for students and faculty members to queue up a need for resources, which will based on proven queuing theory try to distribute the resources among the requesting parties based on their needs in a timely fashion. At the same time the group wants to provide the administrators with an interface that makes management and monitoring of the instances easy and intuitive, while providing enough information to make future investment predictions possible.

2.2 Existing products

At this point and time there is no existing products that do exactly what the thesis description states, which is a GPU[1] scheduling system. Though there are products that exist for the rest of the scope of the project which is containerization, and the management of these such as Kubernetes[2] and OpenStack[3].

Containerization has existed for years, but have never hit the market until later years as a replacement for full-scale virtualization across the board. Products such as Docker[4] which will be used in this project is one of the main reasons for the increased popularity.

[1] Graphical Processing Unit

[2] `https://kubernetes.io/`

[3] `https://www.openstack.org/`

[4] `https://www.docker.com/`

Chapter 3

Project Scope

3.1 Work Plan

- **Backend**
  - Management platform for docker
  - Queuing system
  - Monitoring system
  - APIs for frontend
- **Frontend**
  - Administration
    - Frontend system for management platform
    - Frontend system for queuing platform
    - Frontend system for monitoring system
  - User
    - Access to queuing platform
  - Integrate with FEIDE/ADFS system for authentication
  - Use HTTPS for communication

3.2 Out-of-Scope

Browser Support

The group won't have capacity to fully implement and test for all possible platforms, browsers and operating systems. The group will strive to use modern technologies that should make compatibility less of an issue.

Support after completion

Since this is a project and proof-of-concept, the group won't be able to maintain the application after end of project. The group will strive to make it easily readable, and well documented to be used as a basis for further work by other project groups if applicable.

3.3 Deliverables

The main delivery is the report, as well as

- A working proof-of-concept of the detailed system
- Source code
- Documentation of installation, configuration and management

Chapter 4

Project Organization

Due to the fact that the group only consists of two members makes it so that we'll split the task and either work on thing by our self or together as split a big task in to multiple small bites so it can be done in an efficient way. And that both team member are heavily involved with every task along the way in the project.

For collaboration the group will use the following tools:

- Atlassian Jira for kanban board for issue and time tracking.
- Google Docs for general document sharing and remote storage
- ShareLatex for report writing
- Discord for communication
- Bitbucket for source control

The group will have a meeting with the supervisor every week to oversee the progress of the project and guide us if we're going a bit off track. It's the responsibility of the group to keep reports from these meetings.

Chapter 5

Schedule

5.1 Deadlines

Deadline	Date
Preliminary report	February 20th
Main report	May 19th
Presentation	May 30/31th

Table 5.1: Hard deadlines set by the university

5.2 Milestones

Milestone	Date
Backend - Container API	February 19th
Backend - REST API	March 12th
Backend - Monitoring	March 19th
Frontend - Student Portal	April 2th
Frontend - Management Dashboard	April 16th

Table 5.2: Soft milestones set by the group members

Chapter 6

Assumptions and constraints

6.1 Constraints

- Need to borrow projects/people to simulate real workload to predict usability

6.2 Assumptions

- A user of the system needs to be technically competent
- Basic server infrastructure will be available by the time initial research is done

Chapter 7

Risks

7.1 Project risk

Tunnel vision

Focusing to much on a specific feature or technology might lead members or the group as a whole to spend to much time on a certain part and forget the overall goal. The group or its members can also be to stubborn to abandon a certain aspect that seems to be to time consuming to make it work.
With the help of the supervisor and discussions within the group, this should be minimized.

Delay and unfinished work

There's always a chance certain things might be delayed for numerous reasons, which in turn might force us to move items to out-of-scope during the duration of the project.

7.2 Product risk

External dependencies

Docker support for GPU pass-through and the support from available management systems are still in alpha or beta stages, thus a risk factor is the support and compatibility from these systems.
The group will follow the bug tracking systems where available, and follow up on updates during the process to mitigate these risks.

Bugs

Minor and major flaws in the system, which might be exploitable can be introduced without our knowledge.
Thus code reviews and testing/validation will be used to reduce the likelihood of bugs going unnoticed.

7.3 Risk analysis chart

	Minor	**Moderate**	**Major**
Frequent			
Likely		Bugs	
Remote		Tunnel vision, Delay	External

Table 7.1: Risk assessment chart

Chapter 8

Critical success criteria

A user must be able to...

- Queue new request for a docker instance
- Cancel pending requests
- Insert external data into the container for processing
- Retrieve results after processing

A admin must be able to...

- Monitor the status of the queue
- Monitor the status of the docker environment
- Identify the container's creator

A hacker won't be able to...

- break out of a docker container and get root access on the server.
- get access to sensitive information
- exploit the frontend