



UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

Autumn Report

Docker containers deployed using Bluemix

Thomas Flynn

16117743

Sean McGrath

Information & Network Security MEng

2017

Electronic & Computer Engineering

Table of Contents

1 Introduction and report outline.....	4
1.1 Project description.....	4
1.2 Project aims and objectives.....	5
1.3 Report outline.....	6
1.4 Acronyms.....	6
2 Literature survey.....	7
2.1 Outline of research.....	7
2.2 Docker research.....	8
2.1 IBM Bluemix.....	10
IBM Bluemix DevOps Services.....	12
2.2 Microsoft Azure Container Service.....	13
2.3 Amazon Container Service.....	14
2.3 NoSQL Database.....	15
NoSQL features.....	15
Common types of NoSQL databases.....	15
2.5 Neo4j.....	16
2.6 Neo4j on IBM POWER8.....	19
2.7 Graph visualization.....	20
2.8 Programming languages.....	21
3 Theory.....	23
3.1 Analytic and theoretical aspects of the project.....	23

4 High level design.....	25
4.1 High level design.....	25
4.2 Software.....	26
4.3 Process related block 1.....	26
4.4 Process related block 2.....	26
4.5 Process related block 3.....	26
4.6 Process related block 4.....	26
5 Detailed action plan.....	27
5.1.1 Spring action plan.....	27
5.4.2 Summer action planning.....	29
Summer critical path map.....	30
6 Discussion.....	31
6.1 Progress to date.....	31
6.2 Challenges and risks.....	31
7 Requirements of facilities and materials.....	31
8 References.....	32

1 Introduction and report outline

1.1 Project description

This project aims to provide a bridge between the university's wireless sensor network (WSN) mobile application and the data mining opportunities of the sensor information obtained. This will comprise of a highly scalable database and a simple web application developed using Docker containers deployed on the cloud.

Docker containers are lightweight self contained Linux Operating systems. They will be used to handle the load balancing aspects of the WSN. The Global Position System coordinates transmitted from the WSN will be stored in a graph database for data mining purposes. The database will need to be a non relational database, allowing the database to scale more efficiently.

The mobile sensor IoT application at this point is still in early development by another student. The application will track the Global Positioning System coordinates of the user when they have stepped onto the university's campus. This project will allow for a potential survey feature to be implemented in the mobile application. This would provide great data mining capabilities when combined with other sensor data.

The web application will comprise of a simple user interface that will provide the ability to query the sensor data stored in the scalable database. The data will be used to display the current status of the sensors to the user . Graph databases will allow for additional database filtering features to be implemented on the website in the later stages of the development life cycle.

In summary this project aims to provide a bridge between WSN and data mining. The learning outcomes of this project will provide me with a foundational knowledge in the areas of security, development and operations in relation to designing highly scalable and secure cloud applications.

1.2 Project aims and objectives

Primary objectives

- 1.) Create an instance that will transmit pseudo sensor data.
- 2.) Create an instance that will read sensor data.
- 3.) Develop a database that can store the pseudo data in an organized manner.
- 4.) Develop a web application that will display information from the database.
- 5.) Develop a web server to handle web application database queries.

Secondary objectives

- 1.) Scale up number of instances.
- 2.) Test load balancing aspects of increased scale.
- 3.) End to end testing of application security.

Learning outcomes

- 1.) Discipline and minimizing procrastination.
- 2.) Strengthen programming skills.
- 2.) Development operations.
- 4.) Application security.

Project summary diagram

1.3 Report outline

Literature survey

Summary of the various aspects of the project such as Docker, Cloud container services vendors, database, and application security research.

Theory

Expands upon in detail the areas discussed in the literature survey.

High level design

Description and diagram of how the multiple project components interact with each other from a high level of abstraction.

Detailed action plan

Describes the project management strategy, critical path, and identified risks.

1.4 Acronyms

MS - Microservice

GPS - Global Positioning System

DB - Databases

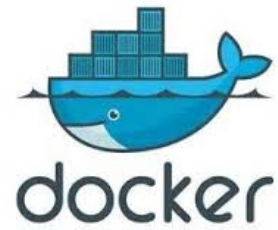
2 Literature survey

2.1 Outline of research

2.2 Docker research

Docker containers

Docker containers wrap a piece of software in a complete file system that contains everything needed to run: code, runtime, system tools, and system libraries. [7]



Lightweight

Containers running on a single machine share the same operating system kernel, they start instantly and use less RAM. Images are constructed from layered filesystems and share common files, making disk usage and image downloads much more efficient. [7]

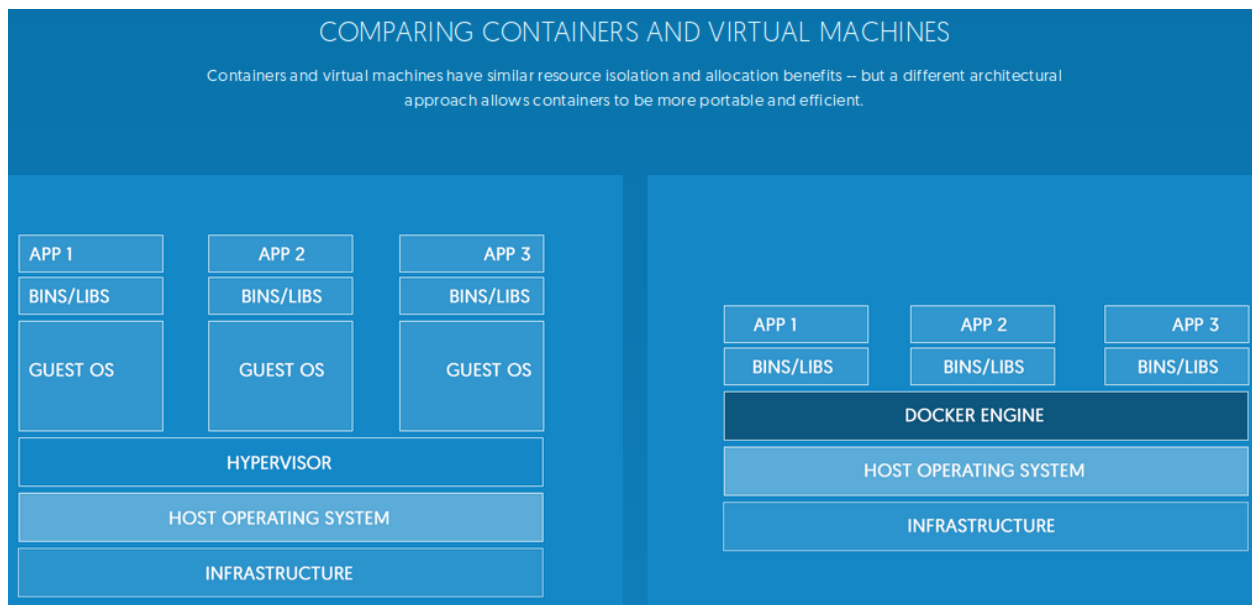
Open

Docker containers are based on open standards, enabling containers to run on all major Linux distributions and on Microsoft Windows -- and on top of any infrastructure. [7]

Secure by default

Containers isolate applications from one another and the underlying infrastructure, while providing an added layer of protection for the application. [7]

Comparing container and virtual machines



Virtual machines

Virtual machines include the application, the necessary binaries and libraries, and an entire guest operating system -- all of which can amount to tens of Gbs. [7]

Eliminate environment inconsistencies

Packaging an application in a container with its configurations and dependencies guarantees that the application will always work as designed in any environment: locally, on another machine, in test or production. [7]

Quickly scale

Docker containers spin up and down in seconds, making it easy to scale application services to satisfy peak customer demand, and then reduce running containers when demand ebbs. [7]

2.1 IBM Bluemix

IBM and Docker integrated container solutions

IBM and Docker offer integrated container solutions that can meet the diverse needs of enterprises. Supporting the creation and deployment of



multi-platform, multi-container workloads across hybrid infrastructures, IBM and Docker accelerate application delivery and enable application lifecycle management for Dockerized containers. [1]

Create scalable and portable apps and services

Having a hosted container service eliminates the need for users to install, operate and scale container infrastructure meaning they can create containerized applications quickly and easily. Since containers maintain all application configuration settings and dependencies, there is no need to change or reconfigure applications for different environments. Applications can move seamlessly between development, test and production environments. Operations teams can spend less time doing manual installations and troubleshooting issues. [2]

Easily package and deploy applications that will run in hybrid environments

IBM Containers on Bluemix makes it possible to more quickly and easily package and deploy applications that can run across hybrid environments. This technology also makes it possible to run many more applications on the same servers, providing improved resource utilization. [2]

Improve productivity and increase resource utilization

IBM Container users can spin up and tear down standardized environments across development and operations groups faster while increasing application density per server. [2]

Achieve maximum application portability

IBM Containerized applications are seamlessly portable across environments and computing platforms, letting users move them freely to, from and between Linux-based cloud environments. IBM Containers does not add any proprietary code to the Docker engine or registry. [2]

Vulnerability advisor

Discover vulnerabilities and compliance policy problems in Docker images and learn new ways to improve images to meet best practices and upgrades to known industry fixes, regardless of the image source.

Integrated container monitoring

Log visibility and performance insight to the CPU, memory, and network utilization per container.

IBM Bluemix security features

The Bluemix platform secures data-in-transit by securing the end-user access to the application by using SSL through the network until the data reaches IBM DataPower Gateway at the boundary of the Bluemix internal network. IBM DataPower Gateway acts as a reverse proxy and provides SSL termination. From there to the application, IPSEC is used to secure the data as it travels from the IBM DataPower Gateway to the application.

Security for both data-in-use and data-at-rest is your responsibility as you develop your application. You can take advantage of several data-related services available in the Bluemix Catalog to help with these concerns.

Security of Bluemix applications

As an application developer, you must enable the security configurations, including application data protection, for your applications that run on Bluemix.

You can use security capabilities that are provided by several Bluemix services to secure your applications. All Bluemix services that are produced by IBM follow IBM secure engineering development practices.

SSO service

IBM Single Sign On for Bluemix is a policy-based authentication service that provides an easy to embed single sign-on capability for Node.js or Liberty for Java™ applications. To enable an application developer to embed single sign-on capability into an application, the administrator creates service instances and adds identity sources. The Single Sign On service supports several identity sources where your users' credentials are stored:

IBM UrbanCode plug-in for application security testing

The IBM Application Security Testing for Bluemix plug-in enables you to run security scans on your web or Android apps that are hosted on Bluemix. This plug-in is developed and supported by the IBM UrbanCode™ Deploy Community on the IBM Bluemix DevOps Services platform

IBM Bluemix DevOps Services

Connect with your GitHub repository

It's easy to link work items with GitHub code changes: use your tools to code, and manage your project with IBM® Bluemix® DevOps Services. You can reference a work item in GitHub comments before you push code changes, and the work item is updated with the new commit information.



Your work in one place

At a glance, see the projects that you own, belong to, and like. You can also see requests to join your project and invitations for you to join other projects. Just another way to help you stay organized.

Built-in source code management

Easily deliver code and seamlessly track changes from all developers. Each project gets a Git repository or a Jazz source code management repository and workspace where project members check in changes, associate code changes with work items, and view a history of recent updates.

Integrated source code editor

The Web IDE provides features to support these tasks:

- Import your source code.
- Copy a file by dragging it to a new directory.
- Edit code quickly. The editor includes several features:
 - Content assist for CSS, HTML, and JavaScript
 - Syntax highlighting for over 20 popular programming languages
 - Syntax highlighting, validation, and content assist for Cloud Foundry manifest files
 - Code validation
 - Keyboard shortcuts
 - Bracket and block-comment auto-completion
 - Key-bindings for vi and emacs users
 - Quick fixes to automatically solve common programming problems in JavaScript

Work items to track and plan project activities

Track who's doing what, and when it will get done. Need a design? Open a task and assign it to the person who has wireframing tools. Is something not working? Open a defect. Expect to feel great when you complete a work item and change the status to Done.

2.2 Microsoft Azure Container Service

Docker for Azure

An integrated, easy-to-deploy environment for building, assembling, and shipping applications on Microsoft Azure, Docker for Azure is a native Azure application optimized to take optimal advantage of the underlying Azure IaaS services while giving you a modern Docker platform that you can use to deploy portable apps. Docker for Azure installs a Swarm of Docker Engines secured end to end with TLS by default, and is integrated with Azure VM Scale Sets for autoscaling, Azure Load Balancer and Azure Storage. [3]



Create an optimized container hosting solution

Azure Container Service optimizes the configuration of popular open source tools and technologies specifically for Azure. You get an open solution that offers portability for both your containers and your application configuration. You select the size, the number of hosts, and choice of orchestrator tools, and Container Service handles everything else. [4]

Scale and orchestrate using DC/OS or Docker Swarm

Choose the tools and solution that best suits your needs for Docker container orchestration and scale operations. Use the Mesos-based DC/OS or use Docker Swarm and Compose for a pure Docker experience. [4]

Manage container applications using familiar tools

There's no need to change your existing management practices to move container workloads to the cloud. Use the application management tools you're already familiar with and connect via the standard API endpoints for the orchestrator of your choice. [4]

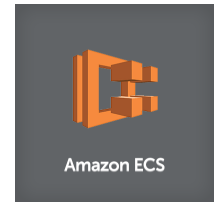
Scale and orchestrate using DC/OS or Docker Swarm

Choose the tools and solution that best suits your needs for Docker container orchestration and scale operations. Use the Mesos-based DC/OS or use Docker Swarm and Compose for a pure Docker experience. [4]

2.3 Amazon Container Service

Docker For AWS

An integrated, easy-to-deploy environment for building, assembling, and shipping applications on AWS, Docker for AWS is a native AWS application optimized to take optimal advantage of the underlying AWS IaaS services while giving you a modern Docker platform that you can use to deploy portable apps. Docker for AWS does not require any software installed. [5]



Amazon EC2 Container Service

Amazon EC2 Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.[6]

Flexible Container Placement

Amazon ECS is a shared state, optimistic concurrency system that supports multiple schedulers on the same cluster for each business or application-specific requirement.[6]

Performance at Scale

Amazon EC2 Container Service is built on technology developed from many years of experience running highly scalable services. You can launch tens or tens of thousands of Docker containers in seconds using Amazon ECS with no additional complexity.[6]

Secure

Amazon EC2 Container Service launches your containers on your own EC2 instances. No compute resources are shared with other customers. Your clusters run in a VPC allowing you to use your own VPC security groups and network ACLs. [6]

Extensible

Amazon EC2 Container Service can easily be integrated or extended through simple APIs. Amazon ECS provides complete visibility and control into your AWS resources, so you can easily integrate and use your own schedulers or connect Amazon ECS into your existing software delivery process.

2.3 NoSQL Database

A NoSQL database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases.

NoSQL features

1. The ability to horizontally scale “simple operation” throughput over many servers,
2. The ability to replicate and to distribute (partition) data over many servers.
3. Simple call level interface or protocol (in contrast to SQL binding),
4. Efficient use of distributed indexes and RAM for data storage
5. The ability to dynamically add new attributes to data records.

Common NoSQL database types

Document store

Documents are addressed in the database via a unique *key* that represents that document. One of the other defining characteristics of a document-oriented database is that in addition to the key lookup performed by a key-value store, the database offers an API or query language that retrieves documents based on their contents.

Key value store

Data is represented as a collection of key-value pairs, such that each possible key appears at most once in the collection. The key-value model is one of the simplest non-trivial data models, and richer data models are often implemented as an extension of it. The key-value model can be extended to a discretely ordered model that maintains keys in lexicographic order.

Graph

This kind of database is designed for data whose relations are well represented as a graph consisting of elements interconnected with a finite number of relations between them. The type of data could be social relations, public transport links, road maps or network topologies.

2.5 Neo4j

Graphs are the most efficient and intuitive way of working with data, mimicking the interconnectedness of ideas in the human mind. Neo4j is built from the ground up to harness the power of graphs for real-time, bottom-line insights.

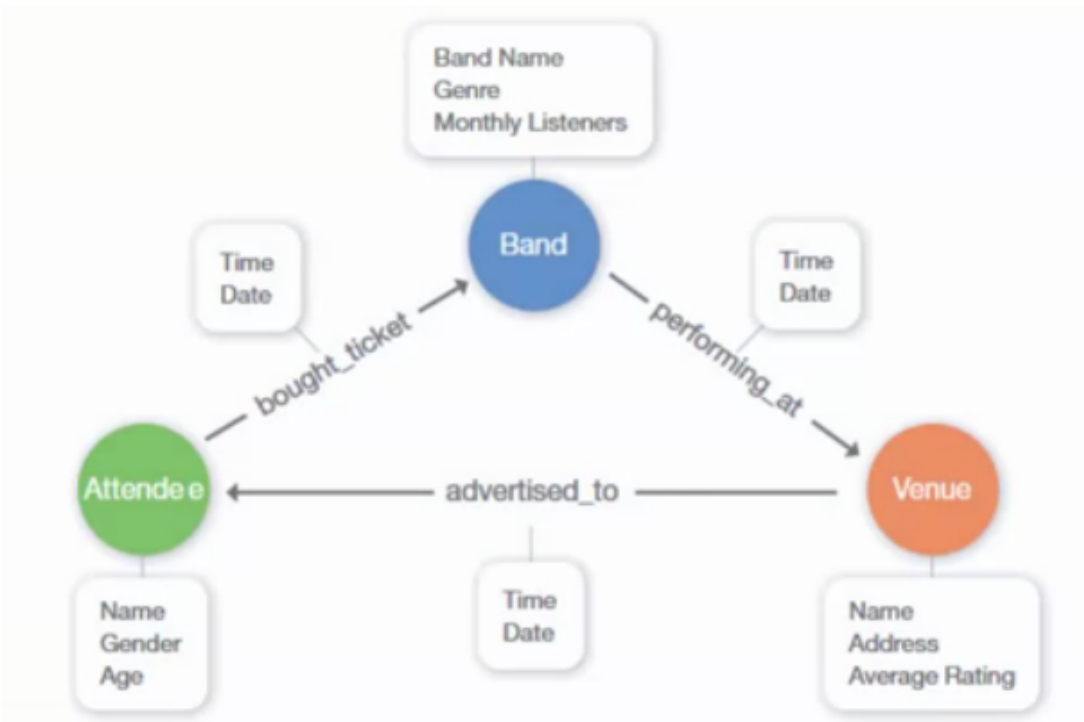


Graph databases

Graph databases are based on graph theory. Graph databases employ nodes, edges and properties.

- **Nodes** represent entities such as people, businesses, accounts, or any other item you might want to keep track of. They are roughly the equivalent of the *record*, *relation* or *row* in a relational database.
- **Edges**, also known as *graphs* or *relationships*, are the lines that connect nodes to other nodes; they represent the relationship between them.
- **Properties** are pertinent information that relate to nodes.

Neo4j



Neo4j vs Relational Databases

Category	Relational Database	Neo4j, Native Graph Database
Data Storage	Storage in fixed, pre-defined tables with rows and columns with connected data often disjointed between tables, crippling query efficiency.	Graph storage structure with index-free adjacency results in faster transactions and processing for data relationships.
Data Modeling	Database model must be developed with modelers and translated from a logical model to a physical one. Since data types and sources must be known ahead of time, any changes require weeks of downtime for implementation.	Flexible, "whiteboard-friendly" data model with no mismatch between logical and physical model. Data types and sources can be added or changed at any time, leading to dramatically shorter development times and true agile iteration.
Query Performance	Data processing performance suffers with the number and depth of JOINS (or relationships queried).	Graph processing ensures zero latency and real-time performance, regardless of the number or depth of relationships.
Query Language	SQL: A query language that increases in complexity with the number of JOINS needed for connected data queries.	Cypher: A native graph query language that provides the most efficient and expressive way to describe relationship queries.
Transaction Support	ACID transaction support required by enterprise applications for consistent and reliable data.	Retains ACID transactions for fully consistent and reliable data around the clock – perfect for always-on global enterprise applications.
Processing at Scale	Scales out through replication and scale up architecture is possible but costly. Complex data relationships are not harvested at scale.	Graph model inherently scales for pattern-based queries. Scale out architecture maintains data integrity via replication. <u>Massive scale up possibilities with IBM POWER8 and CAPI Flash systems.</u>
Data Center Efficiency	Server consolidation is possible but costly for scale up architecture. Scale out architecture is expensive in terms of purchase, energy use and management time.	Data and relationships are stored natively together with performance improving as complexity and scale grow. This leads to server consolidation and incredibly efficient use of hardw

Native Graph Storage

Neo4j uses native graph storage that is specifically designed to store and manage interconnected data. Each piece of data in Neo4j has an explicit connection to every related entity, meaning database queries can ignore anything that's not connected rather than crawl the entire dataset. The result is unparalleled speed and scale.

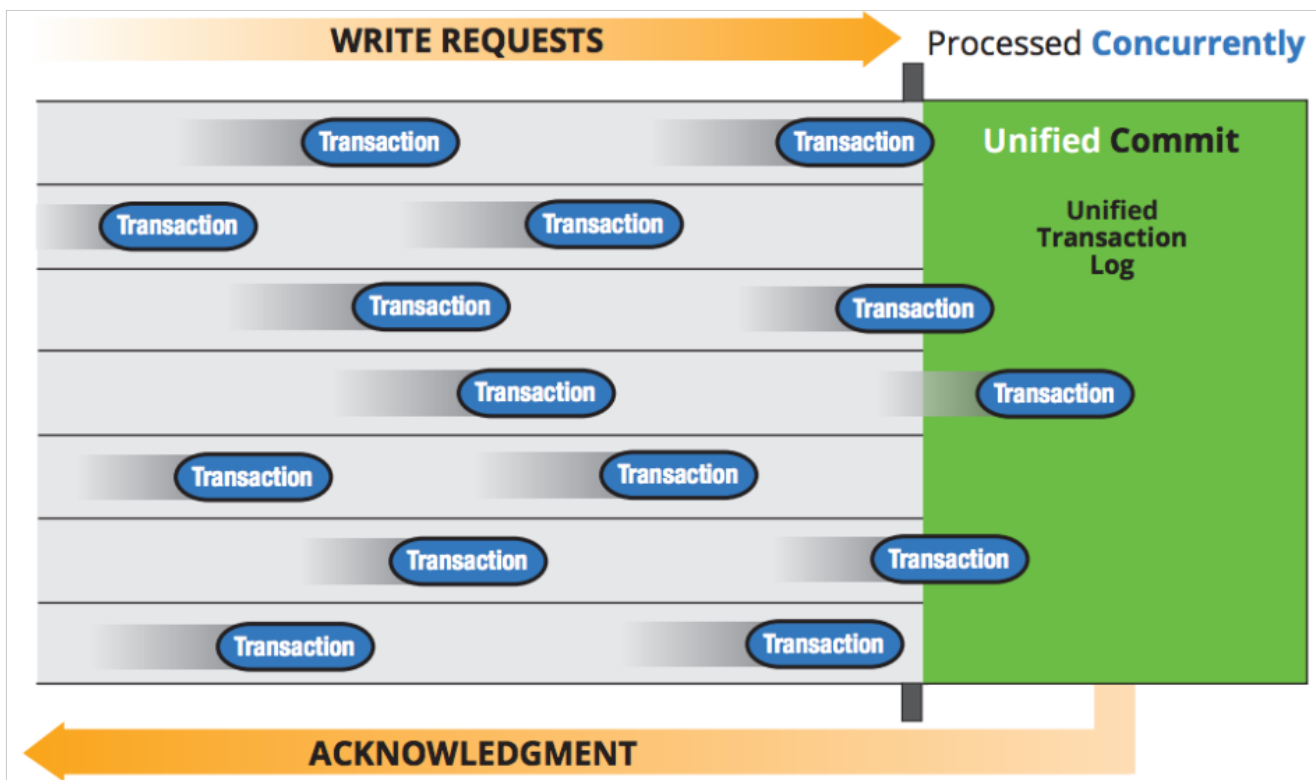
Non-native graph databases use relational or object-oriented databases for data storage instead, which becomes much more latent as data volume and query complexity grow.

Native Graph Processing

Neo4j's native graph processing is the most efficient means of processing graph data because connected nodes physically “point” to each other in the database. This means Neo4j can evaluate results at a rate of millions of hops per second delivering constant-time performance regardless of the size of the dataset.

Non-native graph processing often defaults to expensive index lookups which result in reduced performance.

Concurrent processing of write requests



2.6 Neo4j on IBM POWER8



Neo4j on IBM POWER8 is the result of a joint effort between Neo4j and IBM engineering to provide the world's most scalable graph database platform capable of storing and processing graphs of extremely large size all in-memory – shattering all previous real-time scalability limits.

IBM Power Systems Built with POWER8

IBM Power Systems are specifically designed to capture and manage data from a variety of sources and put that data to work in your enterprise – including real-time graph processing.

Highlights of the POWER8 advantage:

- **Processors:**
 - 4x more hardware threads per core (versus an x86 core)
 - 96 threads on a 12-core chip
 - Up to 1536 threads per system
 - Better Simultaneous Multi-threading (SMT) performance than Hyper-threading (HT)
- **Memory:**
 - 4x more memory bandwidth (versus an x86 core)
 - Up to 16 TB of DRAM
 - 192 GBs of sustained memory bandwidth per scale-out socket
 - Up to 230 GBs per second of memory bandwidth for enterprise-class server
- **I/O Bandwidth:**
 - 96 GBs per second of peak bandwidth
 - 4x more bandwidth than previous generation (POWER7)

For Business

The Neo4j Enterprise Edition offers incredible power and flexibility, with enterprise-grade availability, management and scale-up & scale-out capabilities.

For Individuals

Ideal for learning and smaller do-it-yourself projects that do not require high levels of scaling. Excludes professional services and support.

2.7 Graph visualization

Alchemy.js Open Source Graph Visualization

As you've just seen, preparing, converting and rendering graph data with plain D3 force layout involves a bit of work. The open-source Alchemy.js library replaces this effort with a few lines of configuration. It uses the GraphJSON format for interaction which is easier to handle than the raw D3 JSON format. Alchemy.js is mostly configuration based but can also be customized with javascript functions. It supports custom styling, highlighting, hover effects, clustering and interactive exploration.

Linkurious Graph Visualization

Linkurious uses Neo4j's graph database technology to offer you an easy solution to store, search and visualize graphs. You can start navigating inside your graph database. Search for properties, inspect nodes, and explore their relationships visually in your web browser. find any node in your database easily thanks to our built-in search engine. You can simply modify, add and remove nodes or relationships. You can also customize what is displayed, how the data is indexed.

Keylines Graph visualization

KeyLines is an out-of-the-box JavaScript solution for visualizing networks. How and where you get the data is up to you. KeyLines does the job of rendering it and responding to user interactions like clicking, touching, moving nodes, and more. You bind to these events to customize what happens, and, most importantly, your data stays under your control at all times: KeyLines is self-contained and needs no external connections.

2.8 Programming languages

NodeJS

As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications. Upon each connection the callback is fired, but if there is no work to be done Node is sleeping. This is in contrast to today's more common concurrency model where OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node are free from worries of dead-locking the process, since there are no locks. Almost no function in Node directly performs I/O, so the process never blocks. Because nothing blocks, scalable systems are very reasonable to develop in Node.



Angular JS

AngularJS lets you write client-side web applications as if you had a smarter browser. It lets you use good old HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. It automatically synchronizes data from your UI (view) with your JavaScript objects (model) through 2-way data binding. To help you structure your application better and make it easy to test, AngularJS teaches the browser how to do dependency injection and inversion of control.



Java

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.



Perl

Perl 5 is a highly capable, feature-rich programming language with over 29 years of development. Perl 5 runs on over 100 platforms from portables to mainframes and is suitable for both rapid prototyping and large scale development projects.



PHP

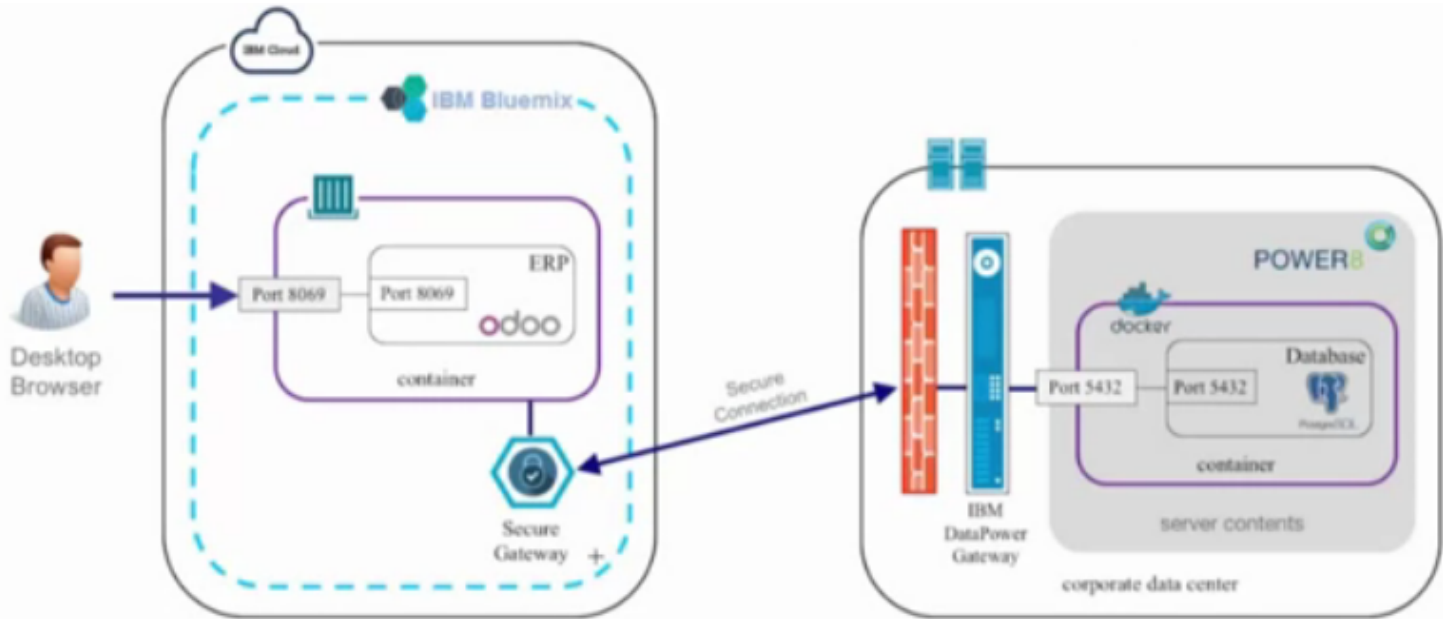
What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.



3 Theory

3.1 Analytic and theoretical aspects of the project

IBM Bluemix and POWER8 configuration example



The above figure demonstrates an example of an IBM container running within IBM Bluemix (left hand side) connecting to a database running within a Docker container that is deployed with IBM's POWER8 (right hand side). The IBM container is linked with the secure gateway service which helps provide a secure connection to the database.

To connect to the database you have to go through 2 layers. The first is a firewall. The second is the IBM DataPower gateway which will make a connection with the secure gateway on the Bluemix side. Behind these 2 layers is the POWER8 system which is hosting the Docker environment.

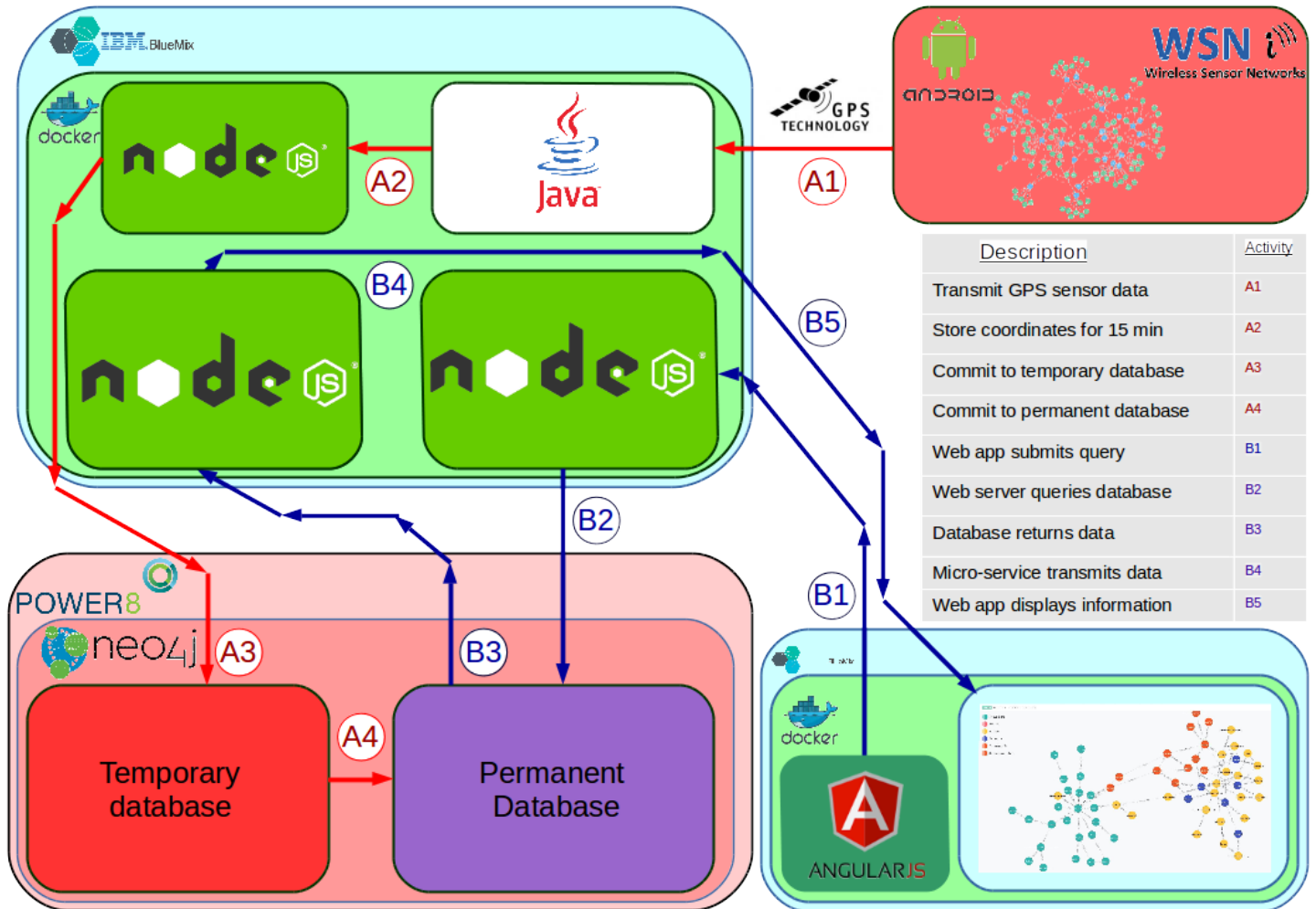
IBM DataPower Gateway

IBM Secure Gateway Service

4 High level design

4.1 High level design

High Level Design Diagram



4.2 Software

4.3 Process related block 1

4.4 Process related block 2

4.5 Process related block 3

4.6 Process related block 4

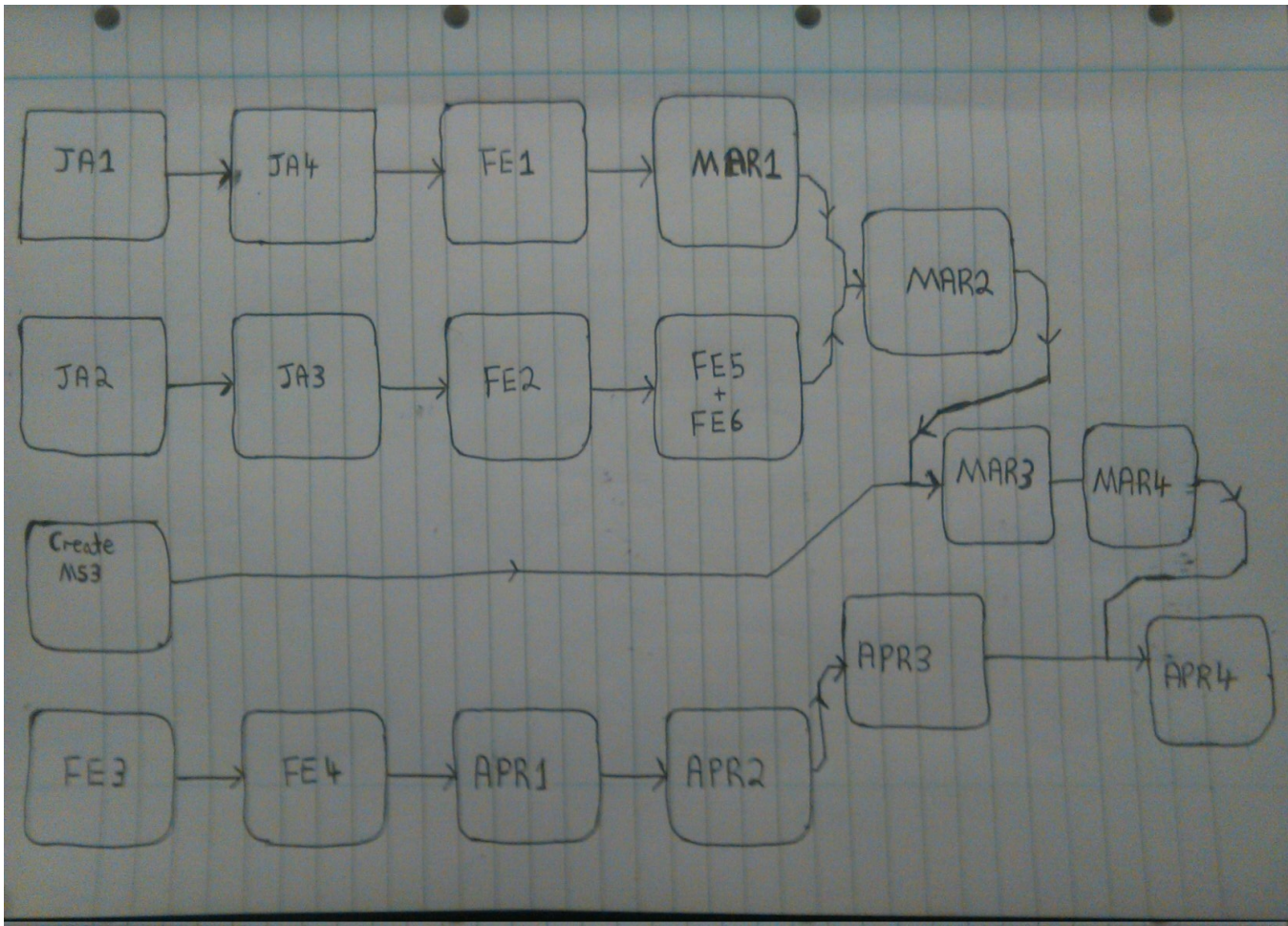
5 Detailed action plan

5.1.1 Spring action plan

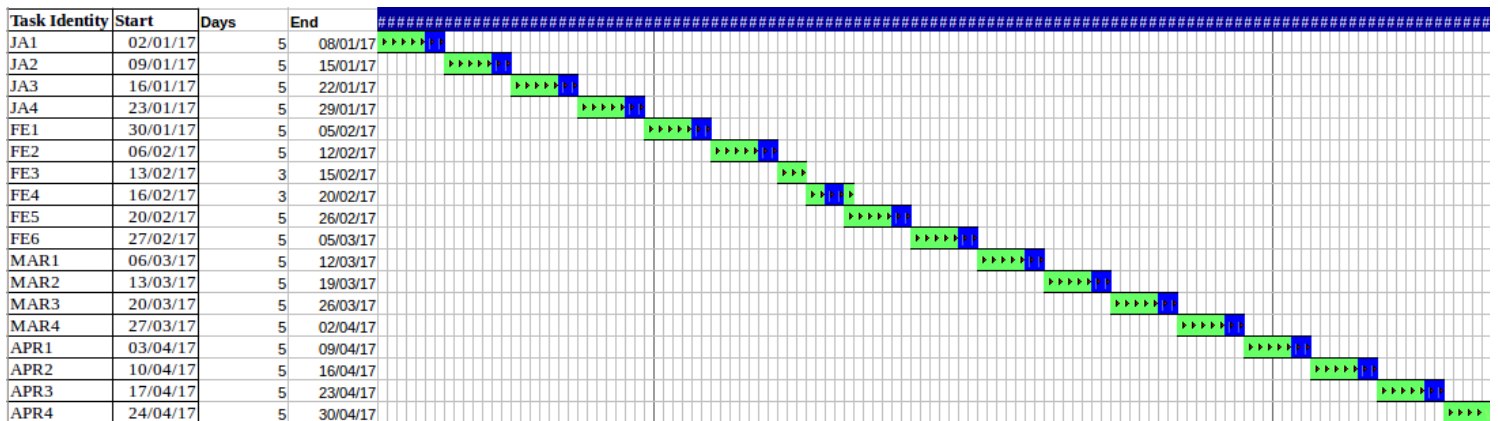
Table of tasks for Spring semester

Task Description	Task Identity	Duration (days)	Preceding Task
Research similar microservices	JA1	5	
Research Neo4J and Cypher	JA2	5	
Research POWER8 graph database examples	JA3	5	JA2
Deploy and run MS1 that creates and transmits pseudo sensor data	JA4	5	JA1
Deploy and run MS2 that handles incoming data	FE1	5	JA4
Deploy a basic Neo4j graph database on POWER8 with suitable pseudo data	FE2	5	JA3
Research Bluemix website deployment examples	FE3	3	
Deploy a basic website on Bluemix	FE4	3	FE3
Create permanent graph DB on POWER8	FE5	5	FE4
Create temporary DB on POWER8	FE6	5	FE5
Develop code for MS2 to store incoming data for a short period of time	MAR1	5	FE1
MS2 writes to temporary database every 15 minutes	MAR2	5	MAR1 & FE6
MS3 reads data from temporary database	MAR3	5	
MS3 writes to permanent POWER8 database	MAR4	5	MAR3
Develop map user interface for website	APR1	5	FE4
Identify important UL GPS coordinates for map user interface	APR2	5	APR1
Develop code for website to submit a simple database request	APR3	5	APR2
MS4 handles a simple website client database request	APR4	5	APR3 & MAR4

Critical path map



Spring Gantt chart

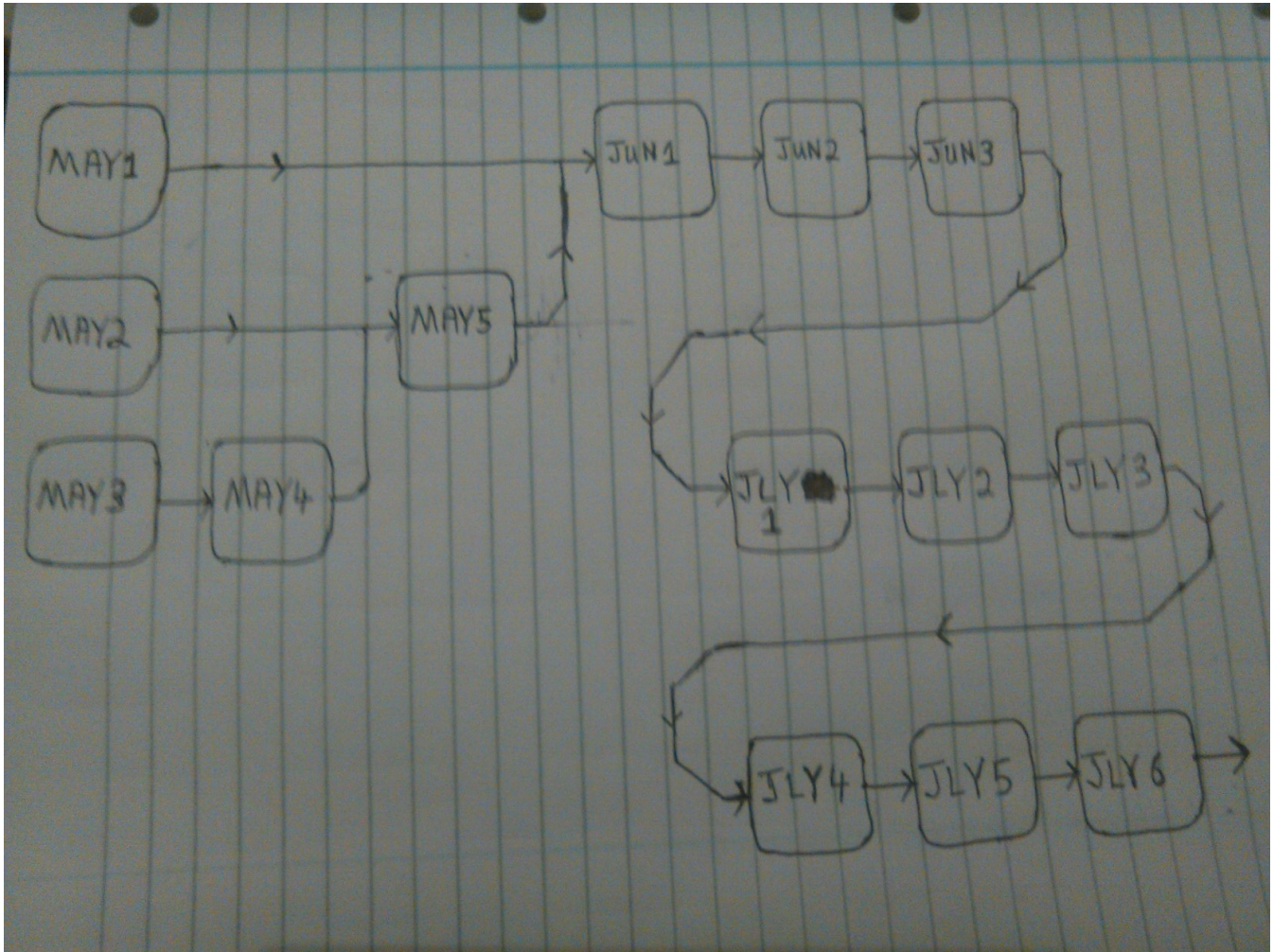


5.4.2 Summer action planning

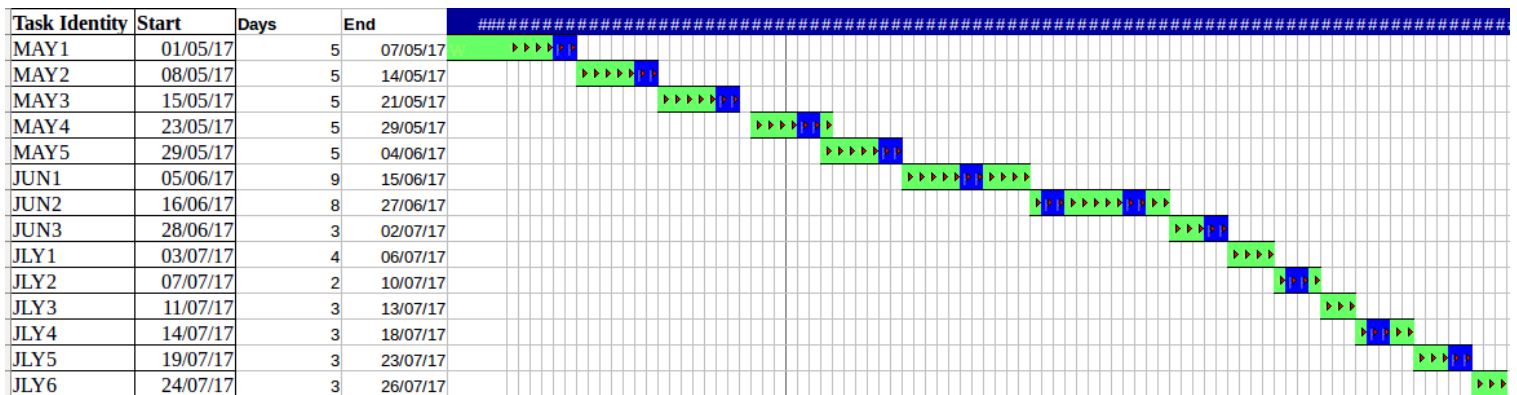
Table of tasks for Summer semester

Task Description	Task Identity	Duration	Preceding Task
Website displays DB readings on website map user interface	MAY1	5	APR4
Load balancing and scalability research	MAY2	5	
Increase funding	MAY3	5	
Increase MS1 instances	MAY4	5	MAY3
MS2 Automatically handles increase/decrease of incoming sensor data from MS1	MAY5	5	MAY2 & MAY 4
Test scalability of MS2 writing to temporary database	JUN1	9	MAY1 & MAY 5
Test scalability of MS3 reading from temporary database and writing to permanent database	JUN2	8	JUN1
End to end testing of website and permanent database under increased scale	JUN3	3	JUN2
End to end testing of the whole system	JLY1	4	JUN3
Collect results	JLY2	2	JLY1
Analyze results	JLY3	3	JLY2
Discuss results with supervisor	JLY4	3	JLY3
Write up initial report and submit to supervisor for review	JLY5	3	JLY4
Edit initial report	JLY6	3	JLY5

Summer critical path map



Summer Gantt chart



6 Discussion

6.1 Progress to date

6.2 Challenges and risks

7 Requirements of facilities and materials

8 References

- [1]"IBM", *Docker*, 2015. [Online]. Available: <https://www.docker.com/IBM>. [Accessed: 14- Oct- 2016].
- [2]"IBM Bluemix - Containers", *Ibm.com*, 2016. [Online]. Available: <http://www.ibm.com/cloud-computing/bluemix/containers/>. [Accessed: 14- Oct- 2016].
- [3]"Microsoft", *Docker*, 2015. [Online]. Available: <https://www.docker.com/microsoft>. [Accessed: 14- Oct- 2016].
- [4]"Azure Container Service | Microsoft Azure", *Azure.microsoft.com*, 2016. [Online]. Available: <https://azure.microsoft.com/en-us/services/container-service/>. [Accessed: 14- Oct- 2016].
- [5]"Get Started with Docker and AWS", *Docker*, 2015. [Online]. Available: <http://www.docker.com/aws>. [Accessed: 14- Oct- 2016].
- [6]"Amazon EC2 Container Service – Docker Management – AWS", *Amazon Web Services, Inc.*, 2016. [Online]. Available: <https://aws.amazon.com/ecs/>. [Accessed: 14- Oct- 2016].
- [7]"What is Docker?", *Docker*, 2015. [Online]. Available: <https://www.docker.com/what-docker>. [Accessed: 14- Oct- 2016].
- [8]"IBM Bluemix Docs", *Console.ng.bluemix.net*, 2016. [Online]. Available: <https://console.ng.bluemix.net/docs/services/graphdb/index.html>. [Accessed: 15- Oct- 2016].
- [9]"IBM Graph - API", *Ibm-graph-docs.ng.bluemix.net*, 2016. [Online]. Available: <https://ibm-graph-docs.ng.bluemix.net/api.html>. [Accessed: 15- Oct- 2016].
- [10]"GremlinDocs", *Gremlindocs.spmallette.documentup.com*, 2016. [Online]. Available: <http://gremlindocs.spmallette.documentup.com/>. [Accessed: 15- Oct- 2016].
- [11]A. TinkerPop, "Apache TinkerPop", *Tinkerpop.apache.org*, 2016. [Online]. Available: <https://tinkerpop.apache.org/>. [Accessed: 15- Oct- 2016].
- [12]"Neo4j on IBM POWER8® - Neo4j Graph Database", *Neo4j Graph Database*, 2016. [Online]. Available: <https://neo4j.com/neo4j-on-ibm-power8/>. [Accessed: 15- Oct- 2016].

- [13]"Neo4j Graph Database: Unlock the Value of Data Relationships", *Neo4j Graph Database*, 2016. [Online]. Available: <https://neo4j.com/product/>. [Accessed: 15- Oct- 2016].
- [14]"Implementing a Containers-based Hybrid Cloud ERP environment using IBM Bluemix", *YouTube*, 2016. [Online]. Available: <https://www.youtube.com/watch?v=9XbzUqCvI3I>. [Accessed: 15- Oct- 2016].
- [15]"Neo4j Spatial: Finding Things Close to Other Things - Neo4j Graph Database", *Neo4j Graph Database*, 2011. [Online]. Available: <https://neo4j.com/blog/neo4j-spatial-part1-finding-things-close-to-other-things/>. [Accessed: 15- Oct- 2016].
- [16]R. Cattell, "Scalable SQL and NoSQL Data Stores", 2016. [Online]. Available: <http://www.cattell.net/datastores/Datastores.pdf>. [Accessed: 16- Oct- 2016].
- [17]N. Staff, "Neo4j 2.2.0 – Massive Write & Read Scalability, Faster Cypher Query Performance, Improved Developer Productivity - Neo4j Graph Database", *Neo4j Graph Database*, 2015. [Online]. Available: <https://neo4j.com/blog/neo4j-2-2-0-scalability-performance/>. [Accessed: 16- Oct- 2016].
- [20]T. Shields, "The Forrester Wave™: Application Security, Q4 2014", *Forrester.com*, 2016. [Online]. Available: <http://www.forrester.com/pimages/rws/reprints/document/109101/oid/1-ZHIZ47>. [Accessed: 20- Oct- 2016].

