

**School of Computing Science and Engineering**

**DECLARATION**

I/We hereby declare that the project entitled **“Druid Monitoring Dashboard”** submitted by me/us to the School of Computing Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by me/us under the supervision of **Sharmila Banu K, Assistant Professor.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

**Kushagra (11BCE0068)**



**School of Computing Science and Engineering**

**CERTIFICATE**

The project report entitled “**Druid Monitoring Dashboard**” is prepared and submitted by **Candidates Kushagra (Register No: 11BCE0068).** Ithas been found satisfactory in terms of scope, quality and presentation as partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** in VIT University, India.

**(Name & Signature of the Internal Guide)**

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

Give your acknowledgement here in the following format specification. Times new roman, 12 point scale, Normal, Justified with 1.5 line spacing. Do not sign in the acknowledgment page.

Follow the hierarchy

Guide & Co guide

Programme Chair

Dean, SCSE

VIT University

**CONTENTS**

**Chapter Title Page**

Title Page i

Declaration ii

Certificate iii

Acknowledgement iv

Table of Contents v

List of Tables vi

List of Figures vii

List of Abbreviations viii

Abstract ix

1 Introduction

1.1 General

1.2 Motivation

1.3 Aim & Objective – Problem Description

1.4 Related Work

1.5 Considered Design Constraints (State at least two constraints)

1.5.1 Economic

1.5.2 Environmental

1.5.3 Social

1.5.4 Political

1.5.5 Ethical

1.5.6 Health and Safety

* + 1. Sustainability
    2. Legality
    3. Inspectability

1.6 Engineering Standards likely to be used

1.7 System Requirements

1.7.1 H/W Requirements

1.7.2 S/W Requirements

1.8 Report Organization

2 Overview of the Proposed System

2.1 Introduction of problem and its related concepts

2.2. Proposed solution

3 Analysis and Design

3.1 Brief Introduction

3.2 Requirement Analysis

3.2.1 Functional Requirements

3.2.1.1 Product Perspective

3.2.1.2 Product features

3.2.1.3 User characteristics

3.2.1.4 Operating requirements

3.2.1.5 Assumption & Dependencies

3.2.1.6 Domain Requirements

3.2.1.7 User Requirements

3.2.2 Non Functional Requirements

3.2.2.1 Performance requirements

3.2.2.2 Safety requirements

3.2.2.3 Security requirements

3.3 Design of the proposed system

3.3.1 Design

3.3.1.1 Design Overview

3.3.1.2 Requirements Traceability Matrix

3.3.2 System Architectural Design

3.2.1.1 Chosen System Architecture

3.2.1.2 Discussion of Alternative Designs

3.3.3 System Interface Description

3.3.3.1 Detailed Description of Components

3.3.3.2 *n* Component-*n*

3.3.4 Considered Design Constraints

3.3.4.1 Economic

3.3.4.2 Environmental

3.3.4.3 Social

3.3.4.4 Political

3.3.4.5 Ethical

3.3.4.6 Health and Safety

3.3.4.7 Sustainability

3.3.4.8 Legality

* + - 1. Inspectability
    1. Engineering Standards used

\* Explanation should be given for the applicable constraints to your project

References

Appendix - I

**LIST OF TABLES**

**Title Page**

Table 1.1

Table 2.1

Table 3.1

Table 4.1

**LIST OF FIGURES**

**Title Page**

Fig 1.

Fig 1.1

Fig 2.

Fig 2.2

Fig 3.

Fig 4.

**LIST OF ABBREVIATIONS**

**Abbreviation Expansion**

WWW World Wide Web

Term 2 Expansion 2

Term n Expansion n

**ABSTRACT**

Druid is an open-source projects that promises high availability, scalability and highly optimized performance for OLAP queries. For data that is rarely edited, Druid performs better than most other distributed systems. Druid services can be configured to emit their logs to an http endpoint. The http endpoint can itself be a druid realtime service, which would ingest all incoming log metrics. Queries can be fired against the realtime node to analyze logs to assess the performance of queries and systems.

For a particular use case, a new query has to written as a JSON object following Druid’s querying syntax and sent as the body of an http POST request sent via a curl command or tools such as Postmam. This is cumbersome and error prone.

Thus it is required to create a robust dashboard system with an intuitive graphical interface that would abstract away the process of writing druid queries and provide easy drill down operations. The results could be visualized through appropriate charts that would immediately make them more readable. The system should also allow easily sharing the rendered information which would help teams tackle issues.

# Introduction

## General

Druid Monitoring Dashboard is web-based application that allows monitoring Druid clusters by analyzing their log metrics emitted vi http. It provides an intuitive user interface and appropriate visualizations to quickly focus on the required information. It provides a generic drill down capability that would help in isolating issues that may be leading to performance bottlenecks while executing querries.

## Motivation

Druid provides cutting edge performance for querying humongous datasets. However a large druid cluster is inevitably complex and micro-management of a large cluster becomes difficult. Also a running query would typically touch a number of host machines taking different amounts of time.

It is possible that due to some unknown reason which may be memory shortage, unavailability of segments, low performing disk or slow network resources to name a few, a particular query might take a much longer time than anticipated. In this case it is hard to find out what went wrong (if something did go wrong in the first place).

Thus the need for a dashboard through which information from logs can be easily obtained and visualized to identify the possible issues.

## Aim & Objective – Problem Description

The requirement is for an application that can be employed to analyze log metrics using an intuitive graphical user interface. Visualization charts are required to facilitate comparison of measures so that potential bottlenecks can be easily identified.

## Related Work

Druid is owned by Metamarkets, a U.S based company that offers analytics solutions. Druid is an open-source project that was conceived at Metamarkets. Metamarkets also owns a dashboard application that analyzes Druid logs, however the project is proprietary.

## Considered Design Constraints (State at least two constraints)

Druid logs contain limited information. As such the application is limited by the information that logs contain. For example, there is no way to know of the state of the system(memory, CPU, disk usage etc) at the time of log emission.

## Engineering Standards likely to be used

## System Requirements

### H/W Requirements

Multiple computing nodes with an interconnecting network

### S/W Requirements

Backend Druid cluster:

* Red Hat Enterprise Linux OS
* Java version > 1.6
* Druid
* MySQL

Hosting machine:

* NodeJS
* Bower
* Grunt
* Yeoman

## Report Organization

# Overview of the Proposed System

## Introduction of problem and its related concepts

Monitoring Druid clusters by analyzing log metrics. Specifically the use case of primary interest is that given a particular query ID, we want to be able to find out how long the query took across different services and across different host machines.

## Proposed solution

An web-based dashboard application is required that can internally talk to Druid log metrics ingestion endpoint. The application should provide a graphical user interface, which would essentially abstract the underlying Druid querying logic from the user. Also it provide visualizations that would be much more readable and informative than reading raw JSON data.

# Analysis and Design

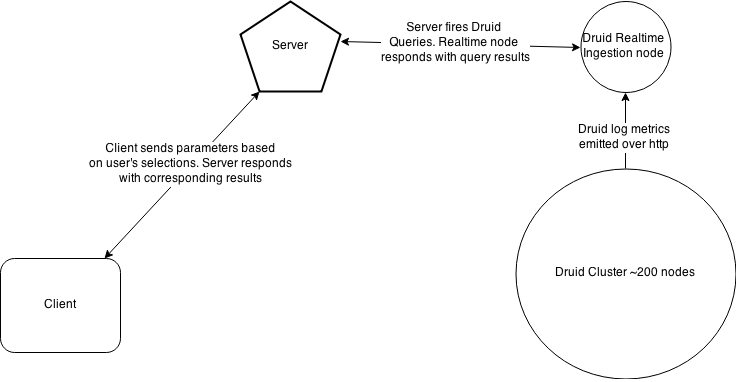
## Brief Introduction

The application will have two components: Server and Client. Server is responsible for communicating with Druid services via http. Client renders the application content view on the user’s screen and routes user’s selection as parameters to the server, which in turn maps into a runnable Druid query.

## Requirement Analysis

### Functional Requirements

#### Product Perspective



#### Product features

* Visualizations for measures count, sum and average of metrics.
* UI-based selection of time interval, granularity and threshold.
* Switching between time-based graph and host-based graph.
* Widgets representing each of the metric types that can be filtered on.
* Updating of widget content and graphs based on selection.
* Easy sharing by copy-pasting URL.

#### User characteristics

Druid system supervisors with a thorough understanding of Druid architecture and Druid Queries.

#### Operating requirements

#### Assumption & Dependencies

#### Domain Requirements

#### User Requirements

### Non Functional Requirements

#### Performance requirements

#### Safety requirements

#### Security requirements

## Design of the proposed system

### Design

#### Design Overview

#### Requirements Traceability Matrix

### System Architectural Design

#### Chosen System Architecture

#### Discussion of Alternative Designs

### System Interface Description

#### Detailed Description of Components

#### *n* Component-*n*

### Considered Design Constraints

#### Economic

#### Environmental

#### Social

#### Political

#### Ethical

#### Health and Safety

#### Sustainability

#### Legality

#### Inspectability

### Engineering Standards used

# References