DISTRIBUTED RESOURCE MONITORING SYSTEM

C.C. Lee | H.W. Lin | S.S. Jiang, supervised by Asst. Prof. Shih-Wen Ke

Overview

It is known to all that having a balance between theories and practices is more than tough. We have studied a wide variety of knowledge related to distributed systems. Therefore, it is whether there exists a distributed system that we are able to construct in a limited time that deserves our attention.

For the sake of providing a useful utility to those who are responsible for keeping computers running regardless of circumstances, Distributed Resource Monitoring System is born. Monitoring with ease is the primary goal of this system.

The concept of Software Engineering is well considered by us during the period of development. We adapt the aspect of the object-oriented programming as well as the incremental development model.

Survey

Design patterns for Distributed System

Given that the realm of distributed computing is too wide to understand in a limited time, we concentrate our study on dissolving the problem of the synchronization of the data. We assume scenarios to examine design patterns acknowledged by us.

Anaconda Cloud

Anaconda Cloud gives us a concept of cluster management tools. Anaconda Cloud is a cluster managing tool. For instance, by using it, the user is able to install an application on every machines at once.

HDFS, Hadoop Distributed File System

HDFS provides us an approach to manage data in a distributed means.

Apache Spark

Apache Spark offers us a brief understanding of Distributed - Computing firmware.

Task Manager

Both the task manager of Windows system and the user interface of AWS, Amazon Web Services, show an example of a monitoring GUI.

Objectives

Managing Monitored Machines

There exists a tremendous amount of information to describes the status of a computer. Therefore, we decide to monitor on four sorts of information, the hardware information, the OS information, and the processes running on each slave machine.

Simple Interface

We realize that the task manager of Windows systems and the user interface of AWS are too complicated. Therefore, we adapt a few of their idiosyncrasies. We implement our GUI with a concept of simplicity and clearness.

Future Work

Reinforcement of HCI

For instance, the information can be displayed in a graphic way.

Extension of functions of the Slave Application

A function that makes slaves can monitoring themselves for the sake of the detecting OS failure of slave machines and reduction of the reliability of the master application.

Reinforcement of the System Architecture

Improvement of data reliability as well as the stability of the system.

Integrity of Embedded System, IoT

This system is cross-platform applications, due to Python.

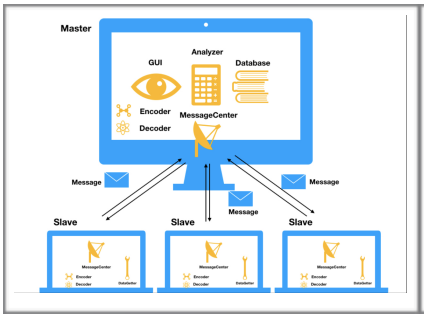
Cluster Management Tool

Such as Anaconda Cloud, by improving the functions of the master application, this plan is highly possible.

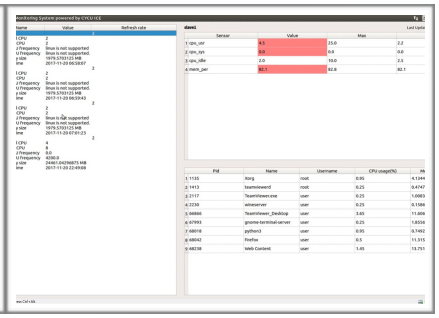
Distributed-Computing Platform

A distributed task management tool is essential, which is likely to be the next goal of this system.

Architecture Overview

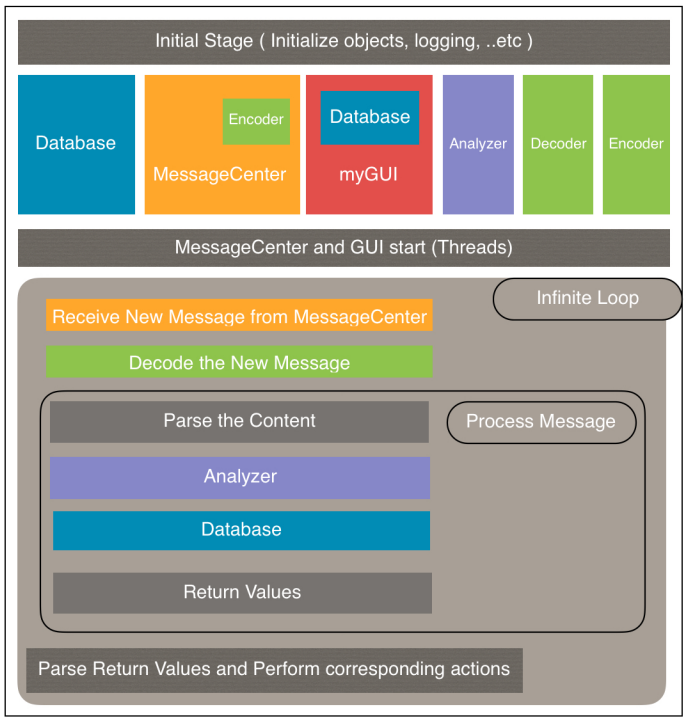


Main GUI



System Architecture

The Architecture Of Master Application



The Architecture Of Slave Application