PaperTrader Protocol Specification

altffour

May 11, 2020

Contents

1 Introduction

This is the document for the specification of PaperTrader. PaperTrader is an application for 'fake' trading assets, to practice investing. The document contains explainations on how to implement the papertrader application. It should be noted that the document isn't 'production-ready' until this sentence is removed. The document will go over the roles of the master server, and the worker servers, how they interact with eachother and the communication protocol, and finally, suggestions on server side implementations.

2 Overview

This section contains the required terminology and modelling of the PaperTrader infrastructure.

2.1 Terminology

2.1.1 Inner World

This is Master server, and all worker servers. This should be kept under high lockdown. Meaning, critical data should be kept secure.

2.1.2 Outer World

This is the frontend, including the desktop cleint, mobile client, or the website client. The data here is controlled by the authorization of the account.

2.1.3 Critical Data

Cirtical Data are all data types that shouldn't be tampered with without authorization. For example, accounts, personal information, messages, and in this context user's portfolios.

2.1.4 User/Client

In this context it is the frontend, which is either the desktop client, mobile client, or the website client.

2.1.5 User/Client Data

This is the data of the user. The meaning depends on the specific conext. It could mean the personal information, credentials, etc. Most of the time it means data that is attached to a data transfer to identify client (IP?).

2.1.6 Master Server

This is the main server that MUST be run when deploying the application. Contains critical data, it would only interact to the outside world by the worker servers.
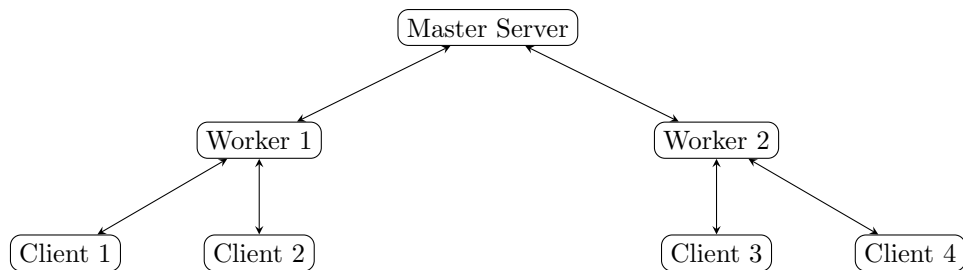
2.1.7 Worker Servers

These are servers that contact the outer world. Worker Servers will interact with the Master Server acting like a 'cache' servers. Data should be routed through worker servers to the master server. The main job for worker server is to add timestamps onto commands sent from the user. The data sent to the main server must contain the data of the client/user. There MUST be ATLEAST one instance running to have a functional infrastructure.

2.1.8 User Accounts

This is the account that abstractly is a the data structure that contains information about hte user and their account.
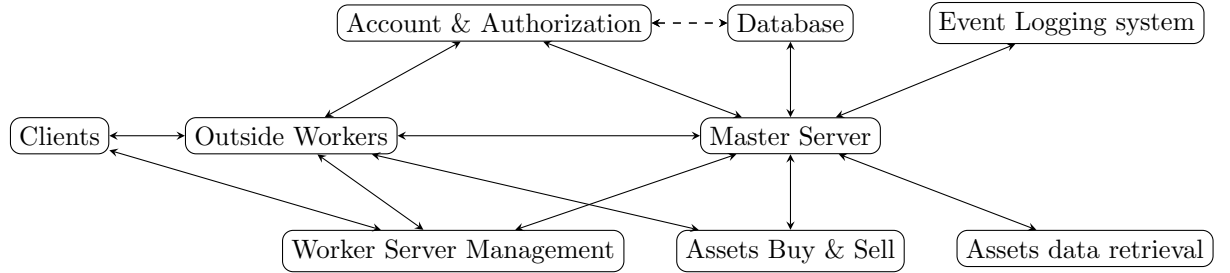
2.2 Infrastructure Model

A fully deployed infrastructure cotains *ONE* master server, *ATLEAST* one worker server, theoretically across the world to maintain speed and reliabilty. An overview diagram of the infrastructure:
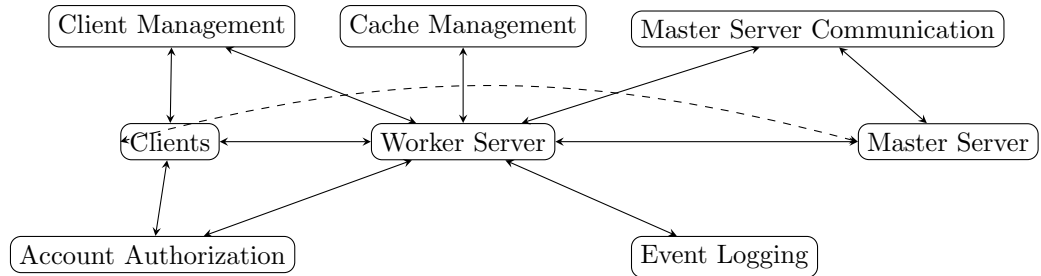
2.2.1 Master Server Infrastructure Model

The master can be defined into modules as demonstrated in the following diagram:

2.2.2 Worker Servers Infrastructure Model

The worker servers can be defined into modules as demonstrated in the following diagram:

2.3 Global Deployment Variables

This section contains an overview of the global deployment variables.

2.3.1 Number of Workers

This is the number of workers deployed with the master server. It must be atleast one. The worker server preferably should be deployed regionally.

2.3.2 Memory Size of Log system

This is a technical variable, this is the size of the log in memory before it being flushed to harddisk. Generally the smaller this is the more disk speed is required. And the larger it is the more RAM the instance needs and the faster it is.

2.3.3 Stock Data Update Interval

This is the interaval of the stock data retrieval. The more this is the faster the transactions that can occur in a minute. This should be planned perfectly so that it can maintain the userbase with the API calls.

3   A more Technical Overview

This is the section that describes the functioning parts of the project in detail. We will start with modules, including master server modules, and worker server modules.

3.1   Master Server

The master server has multiple modules:

- Main Module, i.e the driver.
- Database Management.
- Account Management.
- Event Logging System.
- Authorization.
- Worker Management.
- Assets Data Retrieval.
- Assets Transaction Management.

3.1.1   Main Module

The main module should be able to do the following things:

- Start the authorization thread.
- Start the Worker Management thread.
- Start the assets data retrieval thread.
- Start the assets transaction thread.
- Be able to parse commands from the worker threads.
- Be able to route the commands to the correct thread.

The main module's functionality in relation of the deployment and running stage is as follows, The binary containing the master server is run -¿ initializes states required to operate the server -¿ start the threads -¿ start listening to workers -¿ parse it -¿ pass it to the appropriate thread. This is usually the set of functions that the main function would call. Putting the workings of the main module on a seperate is advised, since it gives the ability to crash the server and dump the logs from the memory of the event log system. Refer to 3.1.1 for insight on why this is required.