

# Map Reducer Architecture Document

---

Phase 4

**Group 5:** Abi Parekh, Emmanuel Sevieux, Stephen O'Connor

**Date:** 12/11/22

## **TABLE OF CONTENTS**

<b>1 Introduction</b>	<b>2</b>
1.1 Document overview	2
<b>2 Software Architecture overview</b>	<b>2</b>
<b>3 Software design description</b>	<b>3</b>
3.1 Workflow	3
3.1.1 Component interfaces	3
3.1.2 Component design description	3
3.2 Map Class	4
3.2.1 Component interfaces	4
Component design description	5
3.2.2 Workflows and algorithms	5
3.2.3 Software requirements mapping	6
3.3 Reduce Class	7
3.3.1 Component interfaces	7
3.3.2 Component design description	7
3.3.3 Workflows and algorithms	8
3.3.4 Software requirements mapping	8
3.4 Final Class	8
3.4.1 Component interfaces	8
3.4.2 Component design description	9
3.4.3 Workflows and algorithms	10
3.4.4 Software requirements mapping	11

# 1 Introduction

All the Architecture information will be captured within this Architecture Document. The design information includes but is not limited to Software Architecture and Process Flow.

## 2 Software Architecture overview

The Phase-4 Map Reducer Software is a Microsoft Visual Studios based Program that is executed from the Command Line. Below Figure 1 describes the different stages of the program. The Next figure describes how the subsystems fit together while Figure 2, specifically how the Data is passed around the system. The Phase 4 Software can be broken into 4 different subsystem with additional common utilities (see below)

1. **MapReducer Controller:** Top Level Class which validates the input configuration, Requests Map and Reducer threads created, processes heartbeat status of Map and Reducer Class, and creates Final Output.
2. **Stub:** Sits and listens for messages (using sockets). The stub will get a message from the controller to tell it to create child processes (either map or reduce).
3. **Mapper DLL:** Extracts individual words (tokens) from files and exports results to intermediary directory
4. **Reducer DLL:** Arranges tokenized words into alphanumeric sorted order and sums the total occurrence of each word.
5. **Utilities:**
  - a. **Socket Common Functions:** Set of Common Classes that create and listens to Sockets.
  - b. **File Common Functions:** Set of Common Functions for Reading and Writing from File IO that is used by all of the software components.

## Design of Map Reducer software

The Figure below shows the End to End Flow Diagram for the program. The Program is broken into 3 different stages: Introduction Stage, Processing Stage, and Finalization Stage. The Introduction Stage is where the Configuration File is validated and the Controller is constructed. The Processing Stage is where all the files are parsed and shared between the different threads which results in a set of Reduced Files. The last stage is the Finalization Stage which merges all the Reduced Files into a Single Output File. Additionally, all the Controller class is destructed and the Socket Server is shutdown.

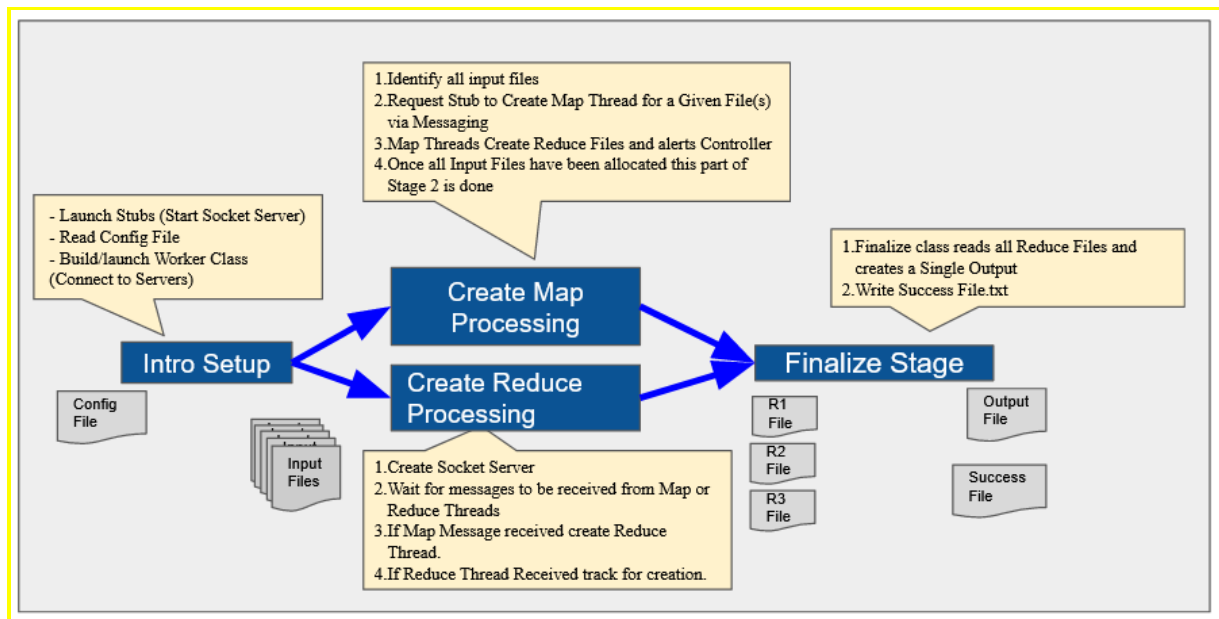


Fig. 1: End to End Flow Diagram

## Design of Map Reducer software

Additionally, Figure 2 identifies how data is passed between the Threads/Processes via Sockets. Prior to the start of the program there should be a set of STUB processes that have socket servers waiting for the controller to connect to. At startup the Controller will create a Server Socket for the Map and Reduce Threads to communicate with. The IP Address and Port for the controller and stub sockets will be provided to the Controller via the Configuration File. The Controller will send a message to one of the Stubs to create a map or reduce thread. The created mapper and reducer threads will send messages back to the controller indicating its status as well as a message indicating the file it created.

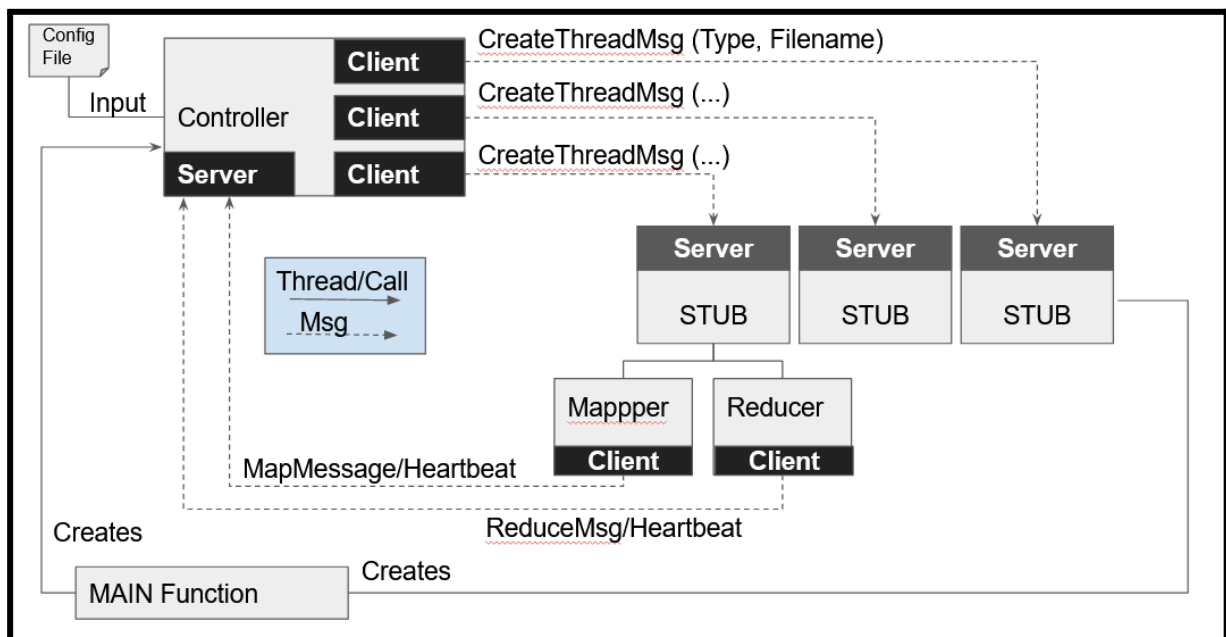


Fig 2: Subsystem Block Diagram