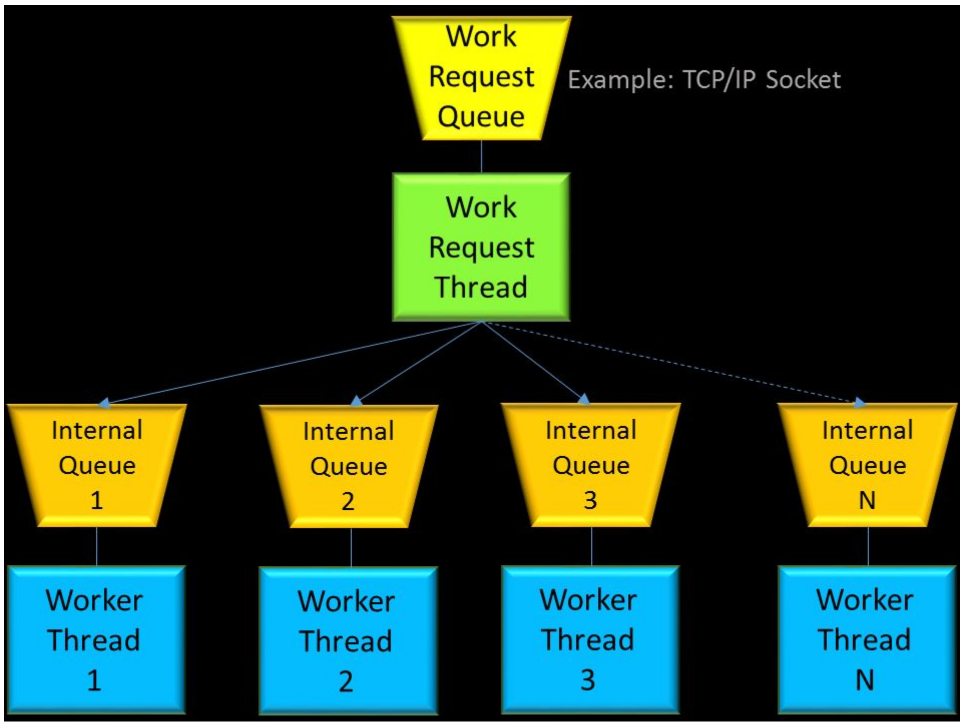
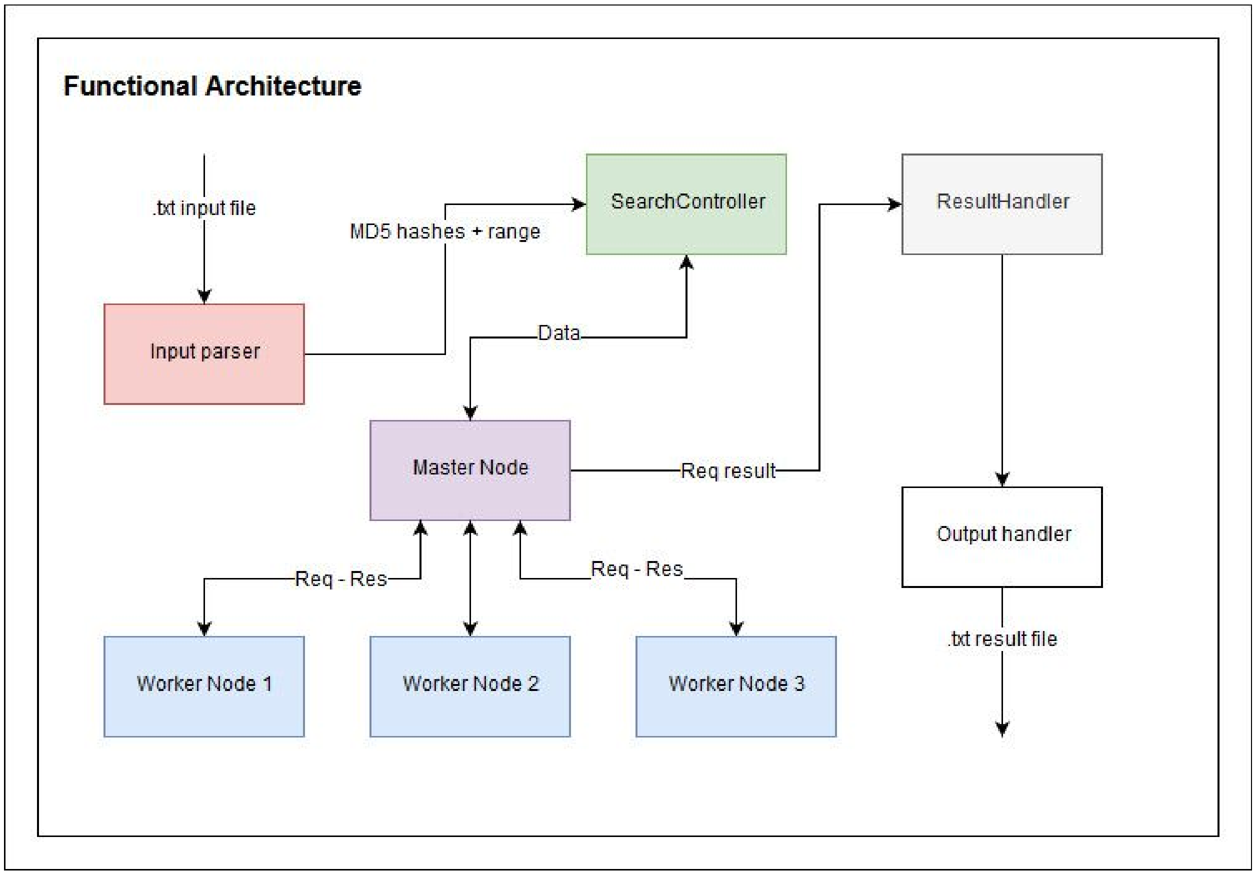


Multithreaded GoLang MD5 Hasher - Component interfaces & component design

June 2019

Yvo Elling
Github repo here

Functional Architecture



Interfaces in functional architecture are as follows:

Nr.	Side 1	Side 2
1	.txt file input	input parser
2	input parser	searchController
3	searchcontroller	masterNode
4	MasterNode	Worker Node(s)
5	MasterNode	ResultHandler
6	ResultHandler	OutputHandler
7	OutputHandler	.txt result file

Software components

The software components of this document have been designed to adhere to certain design principles. In particular, these components will be designed according to the SRP (Single Responsibility Principle) and KISS (Keep it Simple Stupid).

Component I: Input parser node

Input:	A .txt files as defined in the interface
Output:	An array of MD5
Method:	The function of this component is to parse the .txt input file into a slice. The first six lines of the input file that are not marked as a comment (#) will be stored explicity in a different object to keep track of metadata of the input file. The remaining data are considered to be MD5 hash values. These values will be stored in an array of MD5 objects and will be forwarded to the searchController
Implementation	This component requires the following implementation: - *.txt to (array) slice file parser - MD5 object - Metadata object - Forward + Ack function to searchController

Component II: searchController

Input:	An array of MD5 objects and a metadata object for the output file
Output:	An array of MD5-Wrapper Object with additional data fields and an generatorData class with ascii range, amount of workers and the maximum length of the string
Method:	The searchController receives an array of MD5 object. Each MD5 value

	will be retrieved from the array and stored in a wrapper object that contains additional data fields, namely an empty array(tbd) of strings. Besides, it will also create an object of a struct that contains the maximum and the minimum ASCII value to iterate over. This is used for the string generator that will be converted to MD5 hash values. The component will forward both objects to the Master Node. The master node will acknowledge the arrival by returning.
Implementation	The component requires the following implementation: - MD5 hash wrapper class - Create Wrapper objects function - generatorData class - A function to forward the data to the master node.

Component III: MasterNode

Input (1)	Will receive data from the searchController in the form of two objects: a generatorData object and an MD5 wrapper object.
Input (2)	Will receive feedback from the worker nodes in the form of an integer representing the amount of matches it has found.
Output (1)	Will forward the wrapper to the resultHandler once it has received message -1 from all its children
Output (2)	Will forward to its children the following: - The string that needs to be evaluated by the workerNode - A reference to the wrapper array so that it can add the string to the field in the array. - A bool whether or not this was the final Job and the slavenode may die

Method:	The masterNode will receive a wrapper array from the searchController. The masterNode will start a string generator based on the generatorData object sent to it by the searchController. It will create all the possible strings of a specified length in the generatorData object. It will create workerNodes based on the metadata provided that is also provided in the generatorData object. For each string that is generated, the masterNode will sent this string, combined with a reference to the wrapper class (protected with a mutex) and a bool whether or not this was the final job. The masterNode will accept return integers from the worker nodes once they have finished. These integer return values have the following meaning: - X = 0 : No Matches found - X > 0 : X matches found - X = -1 : Child is terminating If no worker node is available, then the masterNode will wait for the first node to be available and pass the job to this node. Once the masterNode has received AMOUNT_OF_WORKERS -1 integer return values, the masterNode will forward the wrapper array to the result handler.
Implementation	The implementation requires special attention for: - A string generator function - Passing a reference to the worker nodes of an wrapper object from the array - Mutex lockers and unlockers

Component IV: WorkerNodes

Input:	Will receive the following data from the masterNode: - The string that needs to be evaluated by the workerNode - A reference to the wrapper array so that it can add the string to the field in the array. - A bool whether or not this was the final Job and the slavenode may die
Output:	Will return an integer to the masterNode. This integer is specified as

	follows: - X = 0 : No Matches found - X > 0 : X matches found - X = -1 : Child is terminating
Method:	The workerNode will send the ‘ready for job’ message to the masterNode once it has started or is ready with a job. A workerNode can be in three states, namely ‘Active’ , ‘WaitingForJob’ and ‘Terminating’. The node is ‘Active’ when it is busy with a job and the other enum states are self-explanatory. A workerNode receives a generated string from the masterNode. The workerNode will compute the MD5 value of this string. The workerNode will subsquently iterate over the reference to the wrapper array to compare the computed MD5 value to the provided values. If no match is found, then the integer 0 is returned to the masterNode together with a ‘ready for job’ signal and workerNode will go into ‘WaitingForJob’ state. If a match is found, then this string is added to the string array data field in the object. Since this is a multithreaded program, the workerNode will lock the mutex to avoid race conditions in writing to the wrapper object. The worker will locally keep track of the amount of hits it made and return the total value at the end of its control loop. When the worker node receives the bool ‘last’ set to true, then the node will go from ‘Active’ state into the ‘terminating’ state instead of ‘WaitingForJob’ when it is done processing the string. The node will terminate after returning the integer.
Implementation	Implementation requires: - A local worker node state machine keeping track of the state it is in. - Returning to a state where the node accepts new job - An Wrapper object iterator to access data fields of all objects in the array. - A local counter keeping track of the amount of matches.

Component V: ResultHandler

input:	Will receive the wrapper array from the masterNode
Output:	Will forward an object with the MD5 hash value and all strings that hash to the same value. It will also forward a bool whether this is the last value or not.
Method:	The ResultHandler will iterate over the wrapperarray and store the MD5 hash value and all the corresponding strings in a new object. It will create an array of these objects, which will then be sent to the outputHandler on request basis. That is, the ResultHandler will not forward a new object before it has received the request for a new object.
Implementation:	Requires: - An Output object that stores the MD5 hash values and the corresponding strings - A send-and-wait function to forward output objects to the OutputHandler.

Component VI: OutputHandler.

Input:	Will receive an object with the MD5 hash value with all the strings that hash to the same value and a bool whether it is the last value.
Output:	A well-defined .txt files that it generates
Method:	The outputhandler will take the data from the metadata object and place this in the header of the output file. The struct can be discarded afterwards The outputhandler will also take data from the MD5 object that is forwarded. It will print the value with the corresponding strings underneath. It will then send a message to the resultHandler and request a new object. This process is iterated until no more message arrive.
Implementation	Requires: - .txt file generator - A print-and-request function.

Software component interfaces

Interface I: .txt input file & Input Parser

The input parser will receive a .txt file. In this file, the data that is required for the operation is specified. The data is specified as follows in the .txt file:

Line number	**What?**	**Purpose**
1	Date	To print date in result file
2	Name	To print name in result file
3	number of working threads	To specify amount of workers
4	Output file name	Output name to be specified
5	Minimal ASCII value to take	Used for range in generating MD5 values
6	Maximal ASCII value to take	Used for range in generating MD5 values
ANY	# = comment	These lines should be ignored
ANY > 6	MD5 value	

Values 1 through 6 will be stored in an object that is passed to the output handler. Any MD5 value that is found will be stored in an MD5 object. Together with all other objects it will be stored in an array of MD5 objects.

Interface II: input Parser & searchController

The input parser will pass the following data to the search controller:

- Array of MD5 Objects
- The object that contains values 1 through 6

The searchController will acknowledge that it has received the array.

Interface III: searchController & MasterNode

The search controller will sent the Minimal and the maximal ASCII value to take into consideration to the MasterNode. The SearchController will store the values from the MD5 object in a wrapper, together with fields 'handled' and an array of strings (for strings that map to the same MD5 values).

searchController sends the following data:

- An array of wrappers of the MD5 objects, with two new data fields
- The minimal and maximal values for the ASCII range to search in

The MasterNode will acknowledge the successful arrival of the data.

Interface IV: masterNode & workerNodes

The masterNode sends the following data to the workerNodes:

- The string that needs to be evaluated by the workerNode
- A reference to the wrapper array so that it can add the string to the field in the array.
- A bool whether or not this was the final Job and the slavenode may die

The WorkerNode will send the following data:

- Will respond with an integer representing how many matches it has found. Integer -1 will mean that the node is terminating after Job.

Interface V: Masternode & Resulthandler

The Masternode will forward the wrapper once it has received message -1 from all its children. This implies that all children has terminated and memory has been cleared.

MasterNode will send the following data:

- The wrapper array with all matches found

The ResultHandler will acknowledge that it has received the message.

Interface VI: ResultHandler & Outputhandler

The resultHandler will remove redundant data from the wrapper (handled field) and sent the data object by object to the outputhandler for writing.

ResultHandler will send the following data:

- A newobject with MD5 hash corresponding strings found

OutputHandler will send the following data:

- Response that it has processed the first object and expects a new object.

The data must be sent per object as a requirements.

Interface VII: outputhandler & .txt result file

The outputhandler will create an output .txt file. Based on the data from the ResultHandler the outputhandler will create the following format:

//////////////////////////////////RESULT FILE ///

Date - Name - NROF WORKERS: X - Output File Name - Min ASCII <-> Max ASCII

MD5 Value 1: [WRAPPER.MD5OBJECT.VALUE]

[result 1]

[result 2]

…

amount of results : X

MD5 Value 2: [WRAPPER.MD5OBJECT.VALUE]

[result 1]

[result 2]

…

amount of results : X

…

//////////////////////////////////END OF RESULT FILE //////////////////////////////////