Design & Programming Team: Alexandru Stanisor
994777434

ECE419 - Lab 4 Design Document

JobTracker: At startup the JobTracker will try to create a ZNode with the path: /job_tracker with the node type Ephemeral and set its IP and port as the node data. This way if more than one JobTracker are present only one will create the folder and store their address. If the creator of that node dies, since the node is Ephemeral it will get deleted and the next JobTracker will become the leader. This methodology also applies to the FileServer. JobTracker also creates a node "/jobs" where all jobs will be added.

JobTracker also listens on a port and creates JobTrackerConnections (each in their own thread) for incoming connections from ClientDrivers. Through these connections ClientDrivers can submit commands: add job, status job, remove job, and receive responses.

Add: When adding a job, the JobTrackerConnection checks if it is already existent, and if it's not it will create it as a child of the node "/jobs", with the path equal to the job hash. It will then create 136 child nodes (tasks) for this job node, each with a path equal to its index. If it dies during this creation, at startup JobTrackers, check all jobs to see if they have exactly 136 tasks. If not they create the remaining nodes.

Status: When checking for status, the JobTrackerConnection retrieves all the task nodes from the job and checks if the number is 136. If not it returns in progress. If the number is proper, it then goes through all tasks and checks to see if they have any children. A task being processed by a Worker will have an Ephemeral node "/in_progress" attached to it. If the task is complete it will have a Persistent child node "/result" attached. The data of this "/result" node determines if the job is complete. If the data is not null then JobTrackerConnection returns it as the final result of the job to the ClientDriver. If it is null, it continues to the next task and redoes the checks. If it goes through all of the tasks and all have "/result" nodes will null data fields, then the job did not find an answer and this result is transmitted to the ClientDriver. If any tasks have no children nodes then the job is in progress.

FileServer: As mentioned above, the file Server also uses a dedicated Ephemeral node for leader election. At startup it loads the dictionary and creates a list of 136 partitions, and a list of 1956 words for each partition. JobTracker also listens on a port and creates FileServerConnections which receive a pointer to this list of partitions. When a Worker requests a partition, the FileServerConnections just retrieves the List of words with the given index and returns it.

ClientDriver: This class handles the user input and transmits commands to the JobTracker. It first performs a lookup for the node "/job_tracker" from ZooKeeper, and retrieves the IP/port pair, which it then uses to establish a connection to the JobTracker. Given the user input it then creates messages and transmits them, and then awaits for the reply. The reply is then displayed to the user. If the TCP connection with the current JobTrackerConnection is terminated, it then performs a new lookup using the "/job_tracker" path. It keeps doing this until server is found.

Design & Programming Team: Alexandru Stanisor
994777434

Worker: Each worker creates a Persistent node "/worker_pool" at startup (it will fail if already present) and also an Ephemeral_Sequential node for itself. It then gets all the jobs under "/jobs" and for each job it gets all the tasks of that job. For each Task it then checks if it has any children. If not it creates an Ephemeral node "/in_progress" and adds this node to a list of tasks which it will process later. When it has gathered enough tasks (based on the argument #of tasks per cycle, provided at start time) it will then go through each of them and it will request the partition with the same index as the path of the task, from the FileServer. The response will be a list of words in that partition. The Worker then runs each word through a hash function and compares the result with the job path (which is the job hash). If a match is found it creates a Persistent node "/result" with its data filed set to the currently processed word, else if it goes through all tasks and does not find a match it creates a Persistent node "/result" with null as data.

The pair of nodes: the Ephemeral "/in_progress" and Persistent "/result" allows for recovery from lost JobTrackers and Workers. JobTrackers can use the presence of Persistent nodes upon startup to determine the state of the job, while Workers can use the Ephemeral in_progress nodes to determine which tasks to tackle. If a Worker dies while it has list of Ephemeral nodes due for processing, those nodes will be deleted, and since the Worker has not had a chance to create the Persistent "/result" node, the tasks will be reprocessed by another Worker. If no tasks are found each Worker places a watch on the children of "/jobs" and also on the children of "/worker_pool". If any of them receive or loose nodes the workers are awakened.

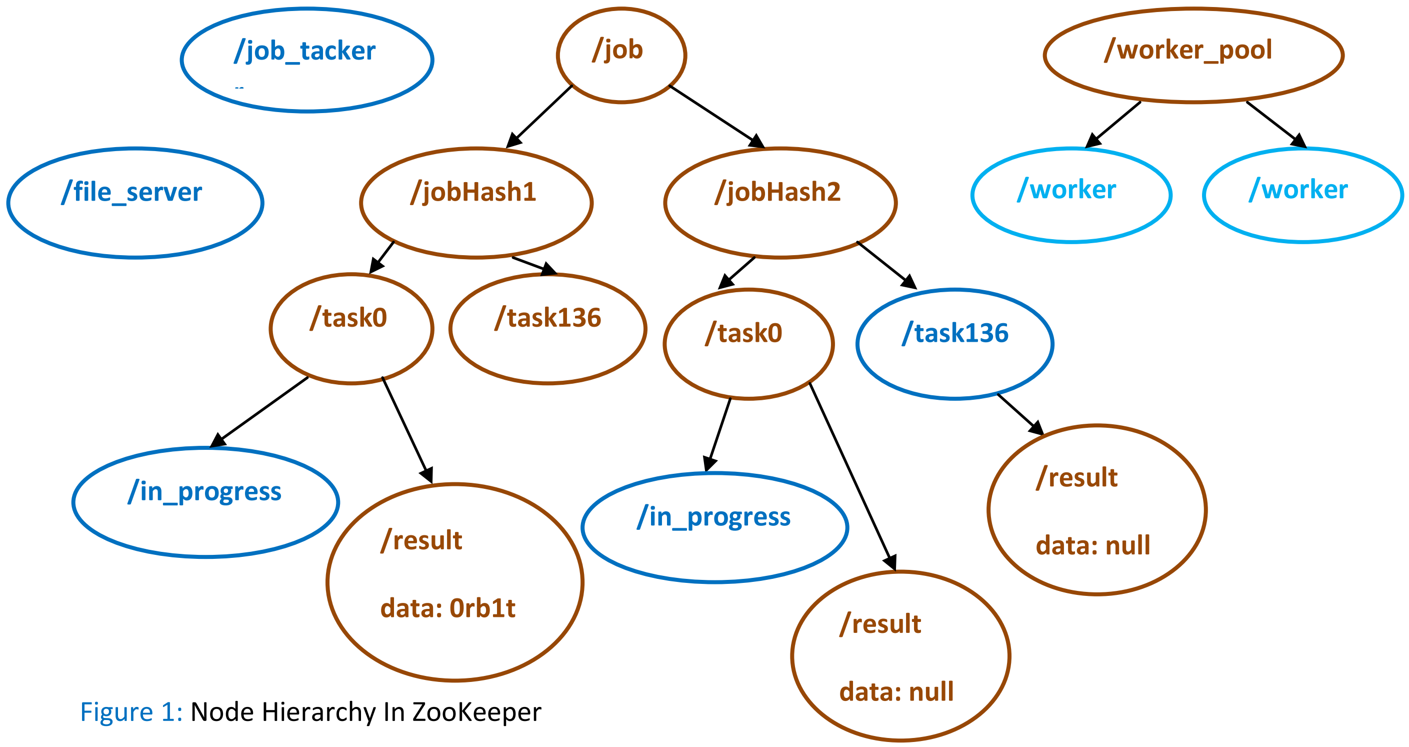

Figure 1: Node Hierarchy In ZooKeeper

Legend: 🟫 Persistent, 🟦 Ephemeral, 🟦 Ephemeral_Sequential