15640 Project1 Report

Portable, Migratable Work

Name1: Jian Wang

AndrewID:jianw3

Name2: Yifan Li

AndrewID: yifanl

Part I: Design

We design the entire migration framework based on a manager-worker model(Figure 1). Manager is the pivotal of the system and only manager can accept user input. All worker nodes will listen to the commands from master and report it’s status periodically to let manager know it’s alive and the status of the process running on it. Worker node only communicate with manager and there is no connection between workers. We use this manager centralized model because it is easy to maintain the consistency of the whole system and simplify the synchronization of different workers. The whole system is divided into 4 packages including manager, process, worker and utility.

The manager package is responsible to collect user input, listen to the joining in request from new worker and communicate with the worker.

The worker package is responsible for handling the commands from the manager process and report the processes status and its own status to the manager every 5 seconds.

The processes package defines the MigratableProcess class which is utilized as the interface for processes to migrate from one worker to another. And we implement two MigratableProcess examples, one is to grep the word in a file and the other is to reverse the word of the every line in a file.

The utility package includes file I/O code and also message code which is used to define the interface between the manager and the wokers. The file I/O migration is implemented by using randomAccessFile to move position in file and keeping a variable – offset to record the reading and writing position in the file.

FAIGURE the Manager-Worker Model

ProcessManager

WorkerNode 1

WorkerNode N

……..

Message

Part II: Commands

We implemented following commands:

1. ls

list all the workers and their status on the whole system

1. ps

list all the processes and their status on the whole system.

1. start <workerId> <process name> <process arguments 1> <…>

start a process on the specified worker node

1. kill <process id>

kill the process specified by the process id

1. migrate <process id> <source worker id> <target worker id>

migrate the process from source worker to target worker

1. clear

clear all the process which is not FINISHED or FAILED

1. shutdown

shutdown the whole system. It will send command to every worker to let them shut down and then manager will shutdown.

1. help

display the help information

Part III: Features

1. Asynchronous status table update:

We use asynchronous model to achieve the status report of the worker. Worker will report it’s status and the process status on it every 5 seconds. Manager will check the status of the worker every 5 seconds. Through the every 5 seconds report from the workers, the manager is able to keep the status of each processes and workers. When any process has encountered failure or other problems, the worker will report the failure to the manager and update the status tables. Whenever the worker crashes due to some reason, manager will know it’s not alive.

1. Race condition free migration:

Since the update of the process status on the manager would happen every 5 seconds after receiving the report from the workers. There might be some situations that the process has already finished or failed but the manager still asks the worker to migrate that process. When this happens, we call it as a race condition. In our system, worker will check the process status when we are trying to migrate it, then this problem will be avoided. And worker will send the failure cause to manager and the process status will be updated.

1. Failure Tolerant

Our system will tolerate the failure on the workers. The cause of the failure will be sent to the manager.

1. Clean status table

We have implemented a command called “clear” to remove all the processes that have terminated due to any reason. In this manner, it is possible to main a clean status table.

1. Extra process status

Due to the fact that process will run on different machine than the manager, so we need more process status to show the accurate status of the process except running and finished. We add more status including STARTING, MIGRATING, FAILED, KILLED. The STARTING and MIGRATING status will be very useful to show the status of the process when there are some problems with the worker and can not finish the start or migrate command.

Part III: Deployment

Require: JRE Java 1.7, Linux, AFS file system

1. Copy source code to each machine
2. “cd” to the src/ directory
3. “make”
4. “cd” to the **MigratableProcess/bin** directory
5. For master machine:

Type in: **java manager.ProcessManager <port number>** in terminal

Example: **java manager.ProcessManager 12345**

1. For worker machine:

Type in: **java worker.WorkerNode <master IP address> <port number>**

Example: **java worker.WorkerNode 192.168.1.112 12345**

5. When the worker machine joins the master machine there will be a notice in the master machine: “One worker joins”

Part IV: Test Case

1. Example:
   1. GrepProcess

Usage: GrepProcess <queryString> <inputFile> <outputFile>

* **Start:** To assign this process to a worker, please use **start** command (make sure master and worker connected successfully)

Type in: **start** **<worker id> edu.cmu.ds15640.process.GrepProcess <queryString> <inputFile> <outputFile>**

Example: **start 0 processes.GrepProcess 15640 /afs/andrew.cmu.edu/course/15/440-f14/handin/lab1/jianw3-yifanl/grepInput.txt /afs/andrew.cmu.edu/course/15/440-f14/handin/lab1/jianw3-yifanl/grepOutput.txt**

* **Migrate:** To migrate a process from one worker to another, please use **migrate** command (make sure use **ps** command to get the process id and worker id first)

Type in: **migrate <process id> <source worker id> <target worker id>**

Example: **migrate 1 0 1**

* **Kill:** To kill a process, please use kill command (make sure use **ps** command to get the process id)

Type in: **kill <process id>**

Example: **kill 1**

* 1. ReverseWordProcess

Usage: ReverseWordProcess <inputFile> <outputFile>

For start command below is the example

Example: **start 0 processes.ReverseWordProcess /afs/andrew.cmu.edu/course/15/440-f14/handin/lab1/jianw3-yifanl/reverseInput.txt /afs/andrew.cmu.edu/course/15/440-f14/handin/lab1/jianw3-yifanl/reverseOutput.txt**

For **migrate** and **kill** command they are same as the GrepProcess one

1. Suggested Test Procedure
   1. Basic test

Launch one master machine and two worker machines. Type ls command to list all the workers. Start a process in worker 0 and migrate to work 1. Type ps to get process status. Migrate process from worker 0 to worker 1 and check the status again. Kill the process or wait for a work finishing the job.

* 1. Advanced test

Launch one master machine and five worker machines. Type ls command to list all the workers. Start 3 processes in worker 0 and migrate to work 2. Type ps to get process status. Migrate 3 processes from worker 1 to worker 2. Then type ps to see the process status. After that, then migrate processes from worker 2 to worker 3. Keep doing this migration operation until all 3 processes reaches the worker 5 or the processes have finished..