A1 – Thread/Process Benchmarker

Ben Soer

A00843110

Table of Contents

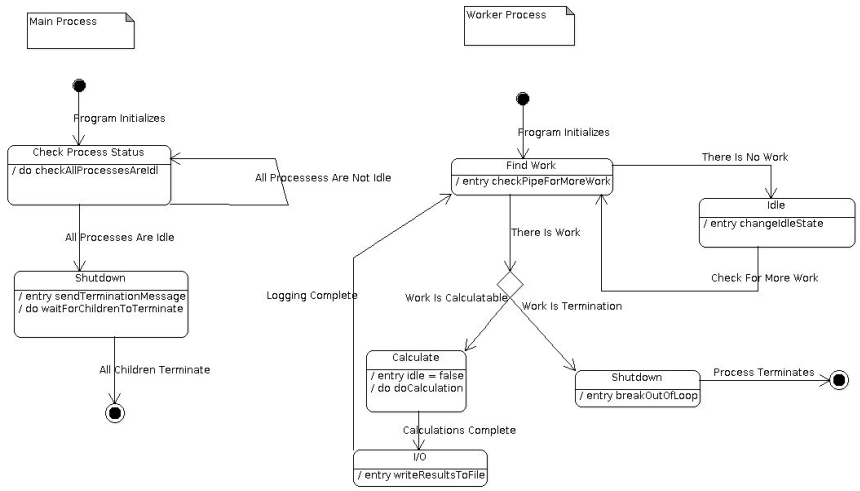
Finite State Machines

Diagram Notes:

- “Program Initialization” as labeled in the state diagrams is defined as all steps carried out in the pseudocode (listed below) up to step 6 in tmain.cpp (main method for threads) and up to step 7 in pmain.cpp (main method for processes)

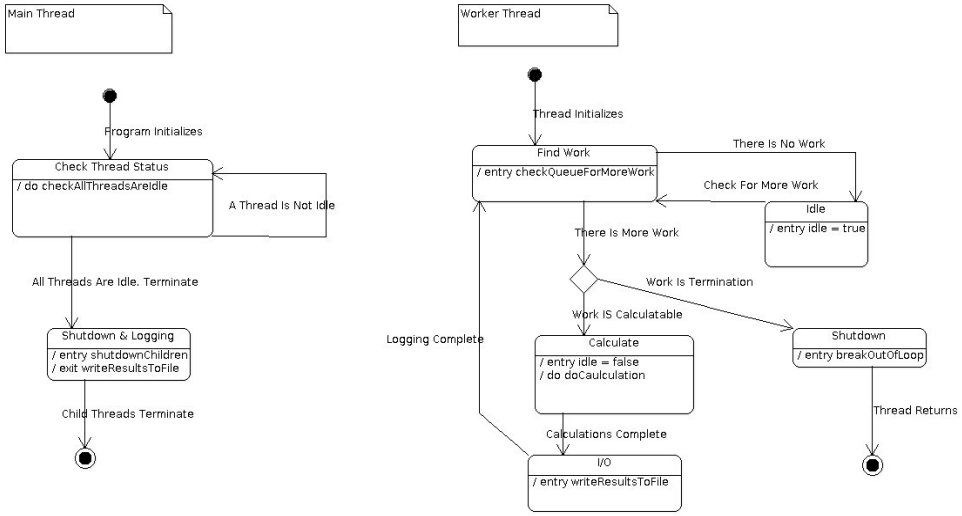
Processess

See ProcessStateMachine.jpg for seperate image



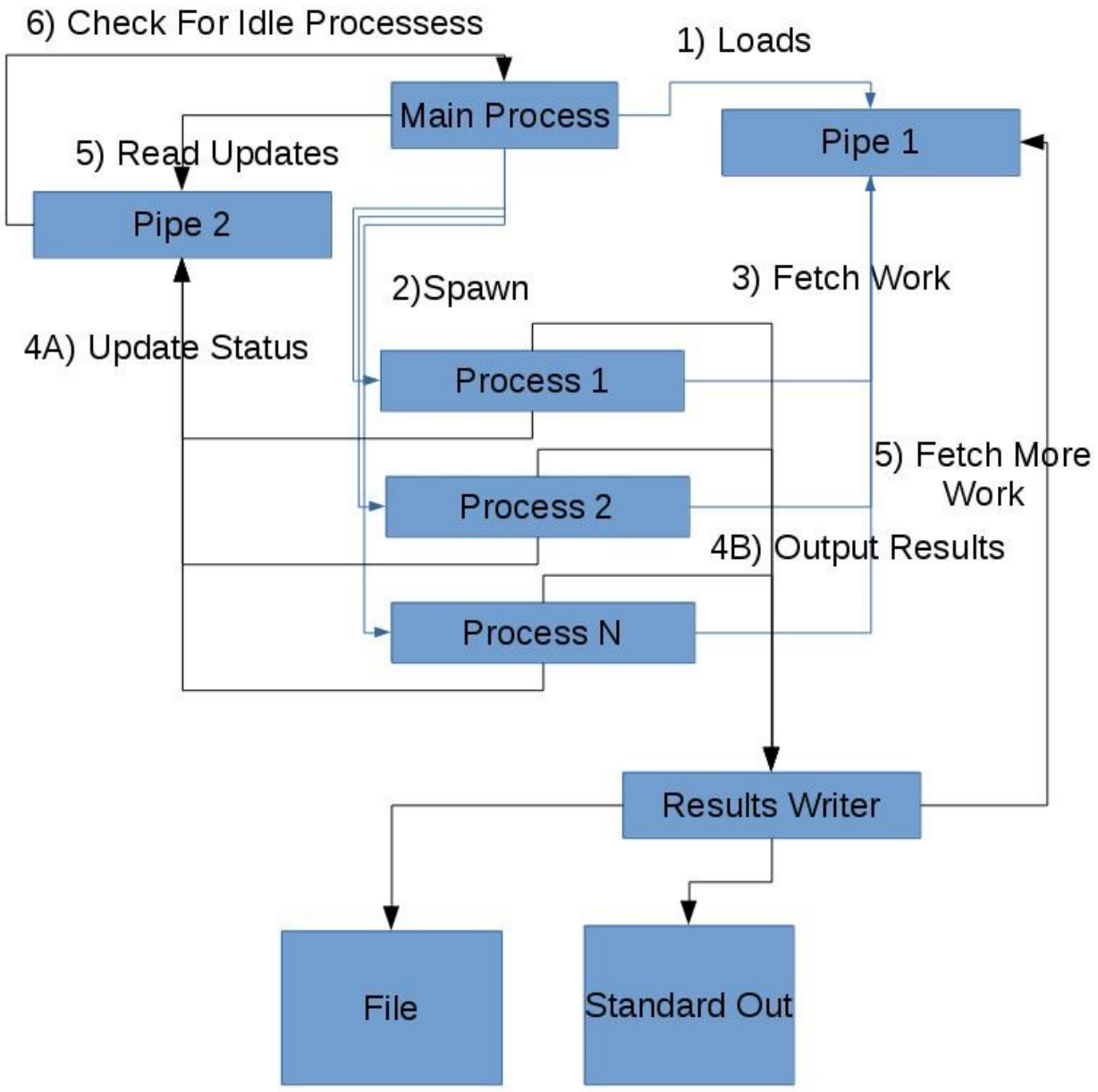
Threads

See ThreadStateMachine.jpg for seperate image

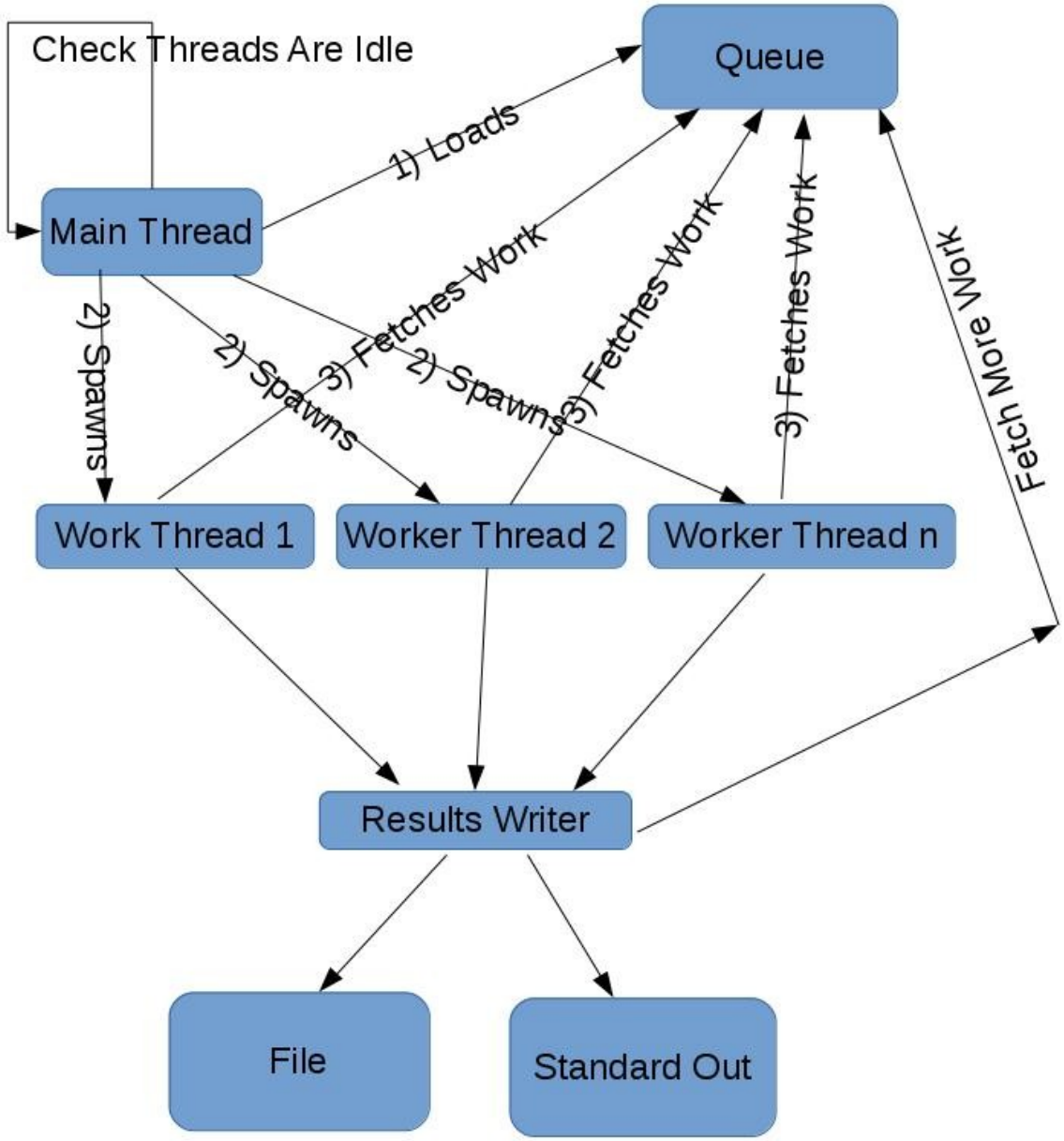


Data Flow Diagrams

Processess



Threads


Check Threads Are Idle
Queue
1) Loads
Main Thread
2) Spawns
2) Spawns
2) Spawns
3) Fetches Work
3) Fetches Work
3) Fetches Work
Fetch More Work
Work Thread 1
Worker Thread 2
Worker Thread n
Results Writer
File
Standard Out

Pseudocode

Threads

tmain.cpp (Main Thread)

main()

1. Read In Arguments
2. Create TaskManager object wrapping a Queue to be used as a shared resource for the worker threads. This will contain all tasks needed by the worker threads
3. Get Time
4. Create 5 POSIX Worker Threads
5. Generate all work for worker threads and load into shared resource queue
6. Start POSIX Worker Thrreads
7. Check for Idle Threads
8. Join all worker threads to main thread
9. Write additional logging information to file
10. Cleanup

bootstrapper(WorkerThread)

1. Instantiate Worker Thread
2. Call Start on Worker Threaded

WorkerThread.cpp (Worker Thread)

start()

1. checkForMoreWork()
2. Execute Caclualtion of new Task

checkForMoreWork()

1. Check for new work in TaskManager
2. While there is no new valid task, set Thread to idle, and then check again
3. Set current task to the fetched task

stop()

1. Set continueRunning to false

isIdle()

1. Return value of idle attribute

Processess

pmain.cpp (Main Process)

main()

1. Read in Arguments
2. Setup Pipes for IPC
3. Load Pipe with Initial Tasks
4. Get Time
5. Create 5 Child Worker Processes
6. While All Worker Processes Are Not Idle, Check All Worker Process Are Idle
7. Wait For Child Processes To Terminate
8. Cleanup

WorkerProcess.cpp (Worker Process)

parseValueForWork()

1. Parse substring of N value and D value, seperated by a '.'
2. cast to long and assign to object attributes

start()

1. checkForMoreWork()
2. if can continue working, parseValueForWork()
3. make calculation
4. log values and send notification messages to parent or more tasks to work pipe
5. else break

stop()

1. Set continueWorking to false to stop allow start loop from continueing

setIdleState(state)

1. If state is different from current state, update state and then send message through pipe to parent
2. else state is the same, do nothing

checkForMoreWork()

1. Check pipe for more work
2. If there is more work, setIdleState(false) and set work as currentWork
3. else setIdleState(true) and release CPU