Pthread_cond_t

Condition Variables were implemented to do the following
On SHM: alert MM servers are booted, and to let Servers know when to change performance mode, a server alert the dispatcher that it's available to receive tasks, alert the monitor when a new task is scheduled or dispatched.
On Servers: alert that MM told the server maintenance is over, to block the High performance thread until server is on high performance mode and while a vCPU is waiting for a new task.
On Task Mngr: To alert scheduler when a new tasks arrived
On MM / Monitor / System Mngr / Task Mngr: to know when to shutdown
Note: each condition variable has a corresponding mutex

Sem_t

Semaphores were implemented to do the following
On SHM: Mutexes for accessing logger and access the multiple SHM areas (config, servers, statistics and tasks)
On MM: To limit the servers in maintenance to (server number - 1), so at least one server is always working

System Architecture

System Manager

- Blocks all signals, starts all shared mutexes
- Loads configuration file (and validates it)
- Creates SHM and populates default values on all vars and initialises pthread_mutex_t / pthread_cond_t that are in shared memory
- Creates named pipe
- Launches *Task Manager*, *Monitor* and *Maintenance Manager* processes
- Initialises signals used (SIGINT, SIGTSTP, SIGUSR1, SIGUSR2)
- Wait on condition variable for program end

Task Manager

- Allocates memory space for unnamed pipes and creates them dynamically
- Starts Edge Server Processes and Creates Threads Scheduler and Dispatcher
- Goes into while loop to read from named pipe

  Scheduler
  - Waits for new task to arrive (condition variable), Iterates all tasks and updates all priorities. If a task is already due, it's removed from queue and logged
  - Broadcasts to Monitor a task was scheduled

  Dispatcher
  - Checks if there's any available server, and if there isn't, waits on a condition variable for a server to signal it
  - Checks and selects the first server available for processing
  - Selects the most urgent task to be processed (highest priority)
  - Checks if the server can process the time in the required time and if it can send the task to server, else checks if there's another available server which can do it in the required time. If no server can do it, drop the task and log it
  - Broadcasts to monitor that a task was dispatched

Monitor

- Launch thread that checks performance mode
- Wait on condition variable for program end

  Monitor Thread:
  - Check current performance mode:
  - "*Regular Performance*"
    - When a new task is scheduled (condition variable), check if the task queue is more than 80% full and if the time required to process a task is greater than the limit specified in the config file. If so, change performance mode to High Performance
  - "*High Performance*"
    - When a task is dispatched (condition variable), check if the task queue is less than 20% full and if it is, change the simulator back to Regular Performance

Maintenance Manager

- Wait for all servers to boot (variable in SHM, MM signalled by conditional variable)
- Create one thread per server to manage maintenance operations

  Maintenance Thread:
  - Randomise time for which the server doesn't need maintenance and maintenance time for next operation
  - Check if server can go into maintenance
  - Message Server to go into maintenance
  - Wait for signal that the server is stopped
  - Do the maintenance time (nanosleep())
  - Message server that it's maintained and can go back to work

Edge Server

- Creates the two vCPU threads
- Create another two threads: performance checker and pipe reader
- After booting, updates "booted servers" var on SHM and signals MM cond var to check if all servers are fully booted
- Main Thread: Reads from message queue, handles vCPU stopping/resuming.
- Pipe Reader - Reads unnamed pipe to handle incoming tasks and assigns them to the vcpu that's free and/or can handle the task
- Performance Checker - waits for a signal from the Monitor to activate the second vCPU or to disable it if going back to regular performance
- vCPU regular performance - Waits for a new task and when the task arrives processes it for the required time. If the server needs to go into maintenance, finish the tasks being processed and go into maintenance, otherwise if the server is waiting for a task, go straight for the maintenance
- vCPU high performance - While server is in regular performance, it's waiting for a signal to be activated when the simulator goes into high performance mode, besides that it's the same as the regular vCPU