Uniprocessor Scheduling Algorithms

- **Highest Response Ratio**:
    - Aims to minimize *average turn-around time*.
    - Calculate *Normalized Turnaround Time*:
        -
        - *w* is the total waiting time, *s* is the predicted service time.
        - Algorithm for predicting *s* value is required.
    - Make priority-queue, priortize by smallest *R* value.
    - Prevents incoming newer processes from starving long, old processes
    - No one starves - since increasing *w* value insures *R* eventually reaches a sufficiently high value.
- **Multi-level Queue (Feedback)**:
    - Two types of task:
        - **Foreground/Interactive:** faster turn-around times are neccessary
        - **Background/Batch:** faster turn-around times are preferred
    - Uses multiple **FIFO queues** with different priority, and each has a different time-slice.
        - Higher the priority, fewer the time-slices.
        - Uncompleted higher-priority task pushed to lower-priority queue with more time-slices
    - Process can move down, but they *cannot* move up.
    - Long processes can starve.
- **Guaranteed Scheduling:**
    - Aim is to ensure that a system with *m* processes gives *1/m* portion of CPU time to each process.
    - *s* is the actual service time of a process.
    - Run process with *lower score* to ensure *s/m* approaches 1.0
- **Lottery:**
    - Each process assinged *Lottery* tickets.
    - Every decision requires choosing a winner randomly.
        - Higher the priority, more the tickets
    - Co-operating processes might *temporarily* exchange tickets
        - Process A request resource from Process B
        - To reduce delay, Process A gives all its tickets to B
        - Process B returns the tickets after receiving resource.
- **Additional Notes:**
    - **Idle Taks:** When nothing scheduled to run, *idle task runs.*
        - Often does *'household'* chores *e.g.* processing statistics, defragmenting disk etc.
    - **Bumping Priority:** High-priorty processes often wait for resources being used by low-priority proceses.
        - Scheduler can *temporarily* increase priority of the low-priority tasks to minimize high-priority tasks' wait.

Multiprocessor Scheduling Algorithms

Multiprocessor Scheduling: Three types of multiprocessor systems:

- **Distributed:** Relatively autonomous systems that interact.
- **Functionally Specialized:** Specialized devices working on specific domain.
- **Tightly-coupled:** Set of processor sharing main-memory, and being controlled by the OS.
    - **Asymmetric Processing:** Only 'Boss' processor can modify the *Process Control Block.*
        - **Load Balancing:** 'Boss' should aim to uniformly distribute work-load across processors
        - This is at odds with *Processor Affinity*
            - Load balancing *must* consider cost of continuting a *blocked/suspended* process on

non-affinite processor.

    - **Symmetric Processing;** All processors can modify *Process Control Block* after using a *Mutex.*

Processor Affinity:

- Processes typically have their own L1, L2, L3, etc. caches.
- Processors possible have *faster* access to certain sections of Main Memory *(NUMA: Non-uniform Memory Access)*
    - Preferable to keep program code in the *same* memory-section
- Desirable to run previously blocked/suspended processes to continue on some processor.

Multi-core Processor: Multiple processors mounter atop same chip.

- Minimizes processor-to-processor communication latency

Hyper-threading

Compute Cycle: Process is actively utilizing processor.
Memory Stall: Process is waiting for a resource (memory access).
Hyper-threading: Run two processes on the *same* processor core.

- Run *'Compute Cycle A'* during *'Memory Stall B'*
- Run *'Compute Cycle B'* during *'Memory Stall A'*