Module 9: CPU Scheduling

In this group activity, we will continue to practice problem solving with the ADJ approach.[1]

- Length: ~35 minutes

Introduction

As previously discussed, we use the following steps for solving a design problem in SER334: Analysis, Design, and Justification.

- **Analysis:** a translation of the original ill-defined problem into concrete requirements that can be evaluated.
- **Design:** a solution to the problem being solved.
- **Justification:** an argument which supports your design as being superior to other potential designs which also satisfy the analysis requirements.

In previous activities, we focused on the base notion of the ADJ method, as well as took a more focused look at the analysis stage and the notation of K to represent knowledge. This time, we will look at a single problem involving *justification* for a given design choice. For the time being, we neglect the design aspect of the ADJ method. This is for a simple reason: design of a technical solution is the primary aspect of most of the coursework you have already completed.

K =

(DON'T PANIC, EVERYTHING IS FINE. YOU DON'T EVEN NEED THIS TO DO THE PROBLEM. -RA)

- Let the total process load of a system be defined as a set called $\mathbb{P}$. Let the next burst of each process n be represented as P_n, an ordered pair of the form $(id,\ t)$, where id is a identifier for a process and t is the burst time. The constraints are that id is unique, and that $t > 0$. As an example, consider a set of ordered pairs of processes and burst time: $P = \{(P0,\ 10),\ (P1,\ 5),\ (P2,\ 8)\}$.
- Let $\mathbb{S}$ be an ordered set that represents a schedule of bursts in $\mathbb{P}$.
- Let the five scheduling criteria be called: *CPU utilization*, *throughput*, *turnaround time*, *waiting time*, and *response time*.
  - CPU utilization (cpu_{util}): At a given time t in a system, we say that $cpu_{util} = \frac{cpu_{used}}{cpu_{capacity}}$.
  - Throughput (*throughout*): For a given time interval T, we say that *throughput* is the sum of the number of process bursts that complete within T.
  - Waiting time ($time_{waiting}$): For some $\mathbb{S}$, $time_{wait} = \Sigma_{p_n \in \mathbb{P}}\,(start(P_n))$ where $start(P_n)$ is the time at which a process starts in $\mathbb{S}$.[2]
  - Turnaround time ($time_{turnaround}$): For some $\mathbb{S}$, $time_{turnaround} = time_{waiting}\,(\mathbb{P}) + \Sigma_{(id,\,t)\in\mathbb{S}} t$.
  - Response time ($time_{response}$): For some P_n, the time t at which the system produces a result for burst P_n.

[1] As we're sure you've noticed, this document is written in Comic Sans. For the advanced problem analyzer: How does it feel to read this document written in Comic Sans? Are there any cognitive side effects?

[2] Just to cover myself: this definition is a little on the simple side and won't quite work for round robin. I'm leaving it simple just to illustrate stuff. -RA

Task 1 (5 minutes): Getting Started

Start by forming a group of 6 ± 1 people, and reading over this document. Begin by discussing the prompt listed in the appendix, and at a high level why you could argue that the given design prompt is correct.

Task 2 (15 minutes): Justifying a Design Choice

See the appendix for the problem that you will be working with. There is already work on the analysis and design, left for you is the task of justifying the design against alternative options.

Justifying this problem lends itself to *elimination* (see below) as a tool to justify the design choice. Given the five criteria that you might optimize for, it would be helpful to go through each and explain why or why not each criterion is or is not the best design choice.

Elimination is a simple problem solving tactic, or we might even call it a ***pattern***. You have certainly seen it before - perhaps the only new idea is how it is structured and written below.[3] For us, a pattern will mean a generic way to solve a problem, that is valid for a certain context. This may sound familiar - it is similar to the idea of design patterns in software engineering.

Elimination

- **Context:** Asked to make and support a choice from a set of options.
- **Requirements:** Set of choices must be small and discrete. Must have a metric to evaluate options, where results for that metric can be ordered.
- **Methodology:** Evaluate each option again metric. Select one as optimal.
- **Justification:** Follows immediately from metric producing an order, having a metric value for each choice, and being able to order the choices based on the metric.
- **Related Patterns:** Optimization (for continuous sets).
- **Difficulty of Application:** Low (requires only mechanical computation once metric is defined).
- **Time to Apply:** Long (requires brute force analysis of all possible alternatives).

Task 3 (15 minutes): Group Presentations

All groups will be asked to provide a ~2 minute presentation on their justification for the provided design.

[3] And then perhaps we might ask ourselves: could we do this for other problem solving techniques we know? Would it even be valuable to do such a thing?

Appendix: Analysis & Design for CPU scheduling

- **Problem Statement**

  Consider a render farm used to render animated scenes for various applications (e.g. physics simulations, movies, shows, etc.). The owners of the render farm are primarily concerned with optimizing their profit for a given render (charging the highest amount that's reasonable while minimizing the actual computation time).

  Therefore, it is important to schedule the jobs in such a way that they can maximize their profits. Which of the five scheduling criteria is most important to optimize for in this situation? Analyze the problem, design a choice, and **justify** the choice.

- **Analysis & Assumptions**

  - Analysis
    - Since the render farm's primary goal is to maximize the profits from each render, the renders must be ordered in a way that the customer receives their final product in the least amount of time as possible. Profit is a direct result of the amount the customer paid minus the server cost (utilities, rent, misc. overhead). The amount the customer paid is not adjustable after the fact, therefore the existing jobs in the queue must be done in a manner that provides the minimum time (therefore power consumption & overhead cost) to get through that job after being added to the queue. In short, the time that a job finishes is the most sensitive criteria to maximizing profits.
  - Assumptions:
    - At a given instance, the render farm might have 5-10 renders in the queue waiting for CPU time.
    - Approx. job size is known ahead of time.
    - Job complexity isn't known ahead of time.

- **The Goal**

  A potential *design* for this problem would be the potential criteria to schedule our jobs in order to maximize profits. Our choices are: *CPU utilization, throughput, turnaround time, waiting time, and response time.*

- **Design**

  After analyzing the jobs on the render farm, it was found that **turnaround time** is the most important criteria to optimize for in this circumstance.

- **Justification**

  (Your group's awesome answer here!)