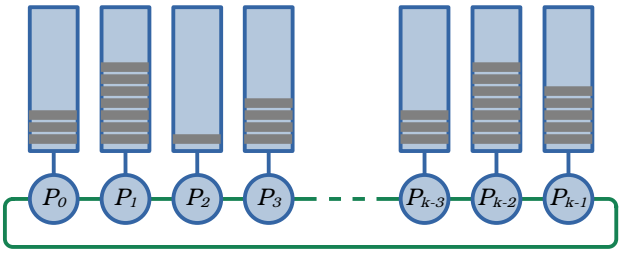
Parallel and Distributed Computing: Homework 8

Objectives: Verify experimentally through computer simulation that a given local-knowledge-based load balancing strategy works and estimate how long it takes to converge to a workload balanced among processors.

Consider a distributed computing system with k physical processors connected on a ring network. Each processor P_i connects to two neighbors $P_{(i-1) \mod k}$ and $P_{(i+1) \mod k}$.



processor P_i is initially loaded with $\|P_i\| \in \mathbb{N}^{+0}$ never-ending load-units. A Load Balancing strategy is implemented as follows:

- Time is divided into time-intervals.
- Each processor schedules load_ balancing activity at random time-intervals.
- Each processor whose load_balancing activity is current will:
  - Look at its two neighbors.
  - Compute the average number of load-units each should have to equalize their load-units.
  - *Give* load units to the neighboring processor(s) such that the load is **balanced among neighbors**.
  - If by *giving* load units balancing is not possible, the processor does nothing. *i.e.* the processor cannot *take* load units from a neighbor.

The described strategy is expected to eventually lead the system to a **balanced state**. The definition of **balanced among neighbors** and **balanced state** of the system, is left to be defined as appropriate.

Assignment

1. Write a program that defines an array of k processors each with $\|P_i\| \in \mathbb{N}^{+0}$ load-units.
2. Your program initializes each processor's load-units with a discrete uniform random variable $\|P_i\| \sim \mathcal{U}(L_{min}, L_{max})$, where L_{min} and L_{max} are the minimum and maximum load-units a processor can be initially assigned.
3. Choose distant values for L_{min} and L_{max} such that the system starts "unbalanced."
4. Your program implements the load balancing strategy previously outlined.
5. Each processor schedules load_balancing activity at discrete uniform random time-intervals $B_t \sim \mathcal{U}(D_{min}, D_{max})$, where D_{min} and D_{max} are the minimum and maximum number of time-intervals a processor can stay without performing load_balancing activity.
6. Your program runs for as long as needed to identify a **steady state** and/or identify the fact that the load balancing strategy will never achieve a steady state. If a steady state is achieved, check whether the system exhibits the defined **balanced state** or not.
7. Impose a time limit to terminate each run and avoid an infinite execution.
8. If the strategy as stated does not converge to a steady balanced state of the system, can you suggest a variation that actually works?

Suggested numbers: It is suggested that you experiment with at least the following configuration: number of processors $k \in \{5, 10, 100\}$; load-units distribution $\|P_i\| \sim \mathcal{U}(10, 1000)$; and load_balancing activity distribution $B_s \sim \mathcal{U}(100, 1000)$. Tally the number of cycles it takes to reach a steady balanced state for each value k.

Report

Please write a report that inlcudes your name and student ID number on top of the first page. Explain the structure of your program, and discuss the results obtained. Also discuss and give answers to the following 6 questions. Please be concise but complete. **The use of Chat GPT is strongly discouraged**:

- What is your definition of "balanced among neighbors"?
- What is your definition of the system in "steady state"?
- What is your definition of the system in "balanced state"?
- Does the proposed strategy converge to a balanced state?
- Can you prove that the strategy will converge to a balanced state for any possible initial load-units distribution?
- What could be the "worst" possible initial load-units assignment?

Submission

Submit one file named hw8.zip, containing exactly 3 files. Make sure the zip file does not include sub-directories (as Mac's default compression tool does) or extra files:

1. balance.c: The code of your simulator. (No template code provided this time.)
2. report.pdf: Your report explaining your design, results, and addressing the aforementioned questions.
3. team.txt: Text file with UCINetID and name of each student in your team. Each line will have this format:
   `<UCINetID>, <firstName>, <lastName>`