CompSci-131: Homework 7

Due May 31, 2018, in pdf format on EEE.

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 $P_{(i-1)mod(k)}$. Each procesor P_i is loaded with $\|P_i\|$ never-ending load units. A Load Balancing strategy is implemented as follows:

- Time is divided into unit 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 an average number of load units each should have to equalize each other's load and will "give" load units to the neighboring processor(s) such that the difference in load units between neighboring processors is balanced. If by "giving" load units balancing is not possible, the processor does nothing.
- The definition of "balanced among neighbors" and "balanced state of the system" is left to be defined as appropriate.

The purpose of this exercise is to verify experimentally that the load balancing strategy described above works and estimate how long it takes to converge to a balanced workload among processors. For this purpose, computer simulation will be used.

1. Write a program that defines an array of k processors each with a load of $\|P_i\|$ load units.
2. Your program initializes $\|P_i\|$ using a random number generator with uniform distribution, but such that the system is "unbalanced."
3. Your program implements the load balancing strategy outlined above.
4. Scheduling load activity for each processor is also done using a random number generator with uniform distribution.
5. Your program runs for as long as needed to identify a steady state (load is balanced) and/or identify the fact that the load balancing strategy will never converge to a balanced system. Impose a time limit to terminate each run and avoid an infinite execution.
6. If the strategy as stated does not converge to a balanced system, can you suggest a variation that actually works?

Suggested numbers: Try $k = \{5, 10, 100\}$ with $\|P_i\|$ in the interval $[10, 1000]$. Scheduling load activity at each processor could be done at random time intervals in the range $[100, 1000]$. Tally the number of cycles it takes to reach a steady state for each case value k.