Aswin Sampath - 53844684 - saswin@uci.edu Shreya Chetan Pawaskar - 12041645 - pawaskas@uci.edu

PDC HW8 REPORT

Group No: 18

1. *Aswin Sampath - 53844684 - saswin@uci.edu*
2. *Shreya Chetan Pawaskar - 12041645 - pawaskas@uci.edu*

Structure of the Program:

Variables:

- `LMIN`, `LMAX`, `DMIN`, & `DMAX` are constants defining the minimum & maximum values for load & time units respectively.
  - `LMIN=10`
  - `LMAX=1000`
  - `DMIN=100`
  - `DMAX=1000`
- `MAX_CYCLES= 2*DMIN*DMAX` represents the upper time threshold -> 200 cycles in total.
- `processorLoadUnits` & `processorTimeUnits` are used to store the load & time units for each processor, respectively and `hashTable` is used to efficiently index processors based on their time units.

Functions:

- `findNeighborsLoadUnits`: Finds the load units of neighboring processors
- `assignProcessorLoadUnits`: Assigns random initial load units to processors.
- `assignProcessorTimeUnits`: Assigns random time intervals for load balancing activity.
- `updateHashTable`: Updates the hash table based on the time intervals.
- `checkbalanced`: Checks if the system is balanced or not. It returns 1 if balanced, 0 otherwise.
- `simulate`: Simulates the load balancing strategy until a balanced state is achieved

Code Logic:

1. The program is run for 3 values of processors: 5, 10, & 100.
2. Initialize processors with random load units & time intervals in their range, for load units ->(10,1000) and for time units (100,1000).
3. Set a hash table for time intervals
4. Run simulation until steady balanced states:
   a. For each step:
      i. Identify processors scheduled for load balancing
      ii. For each such processor:
         1. Find load units of left & right neighbors
         2. Check load balancing
         3. If unbalanced, perform load balancing
   b. Check for balanced state & end if that is achieved
5. Output the no of cycles

Example Output for k = 5:

Processor Load Units -> *977 539 483 588 590*
Processor Time Units -> 972 844 378 423 209
Hash Table
209: 4
378: 2
423: 3
844: 1
972: 0
***Balanced State ->** 592 591 590 590 590*
Number of cycles taken to balance for 5 processors: 6

Analysis:

1. *Number of cycles taken to balance for 5 processors:* ***6***
2. *Number of cycles taken to balance for 10 processors:* ***9***
3. *Number of cycles taken to balance for 100 processors:* ***162***

1) What is your definition of “balanced among neighbors”?

When distributing a set of processing units among multiple processors, the approach involves sharing excess load with neighboring processors if their current load is lower. This sharing mechanism aims to equalize the load distribution, with the processors collectively computing an average and redistributing the load accordingly for more balanced processing.

2) **What is your definition of the system in “steady state”?**

A steady state is when the load balancing strategy no longer leads to significant changes in distribution. The `simulate` function runs until either a steady balanced state is achieved or a maximum number of cycles is reached. System reaches equilibrium.

3) **What is your definition of the system in “balanced state”?**

A balanced state is when all processors have load-units within a defined range & neighboring processors have similar load-units. The loads are distributed equally among the processors.

4) **Does the proposed strategy converge to a balanced state?**

While the proposed strategy generally converges to a balanced state (162 cycles for k=100 processors), challenges arise in cases of extreme initial load imbalances, strict load-sharing conditions, or unfavorable randomness in load assignments. The algorithm's success hinges on the interaction between processor loads and initial randomness. If load adjustments are restricted or the simulation cycle limit is reached prematurely, convergence may be hindered, preventing the achievement of a balanced state.

5) **Can you prove that the strategy will converge to a balanced state for any possible initial load-units distribution?**

Since the load of processors always gets distributed among its neighbors , the amount of load it takes is always less than what it is initially distributed. In each cycle , the load keeps getting smaller with the system eventually reaching a balance state.

Example below for k=5 processors

Cycle 1 977 539 483 588 590

Cycle 2 689 689 553 553 553

Cycle 3 613 613 643 553 553

Cycle 4 597 597 594 586 586

Cycle 5 592 592 594 588 588

Cycle 6 592 591 590 590 590

In 6 cycles , processors reached steady state

6) **What could be the “worst” possible initial load-units assignment?**

- The worst case is likely all load units assigned to a single processor, & 0 on other processors.
- If certain processors start with extremely high or low loads, it will be difficult for the strategy to equalize the loads among neighboring processors.
- In such a case, the system might not converge to a balanced state, & the load balancing may be ineffective.

References:

1. Lecture Slides by Prof. Isaac D. Scherson