Name: Nivedita Mishra
Student Id: 36406272

Name: Chengxuan Du
Student Id: 61329991

Parallel and Distributed Computing: Homework 8

Structure of our program :

The program is designed to simulate load balancing in a distributed computing system. It is structured into several key components:

Global Definitions and Types:

- PID is a type alias for size_t, representing the ID of a processor.
- MAX_ITER is a global constant set to the maximum value for size_t, used to prevent infinite loops during simulation.

Utility Functions:

- copyLoadsList: Copies the current load values into a past load array for later comparison.
- isMaxDiff: Checks if the maximum difference in load across all processors is less than or equal to 1, a criterion for a balanced state.
- checkLoad: Compares current and past load arrays to determine if the system's load distribution has stabilized.
- initExecCyclesUniform: Initializes execution cycles for each processor with random values within a specified range.
- findLowestExecCycle: Determines the processor with the lowest execution cycle value.
- isExecCyclesOL: Checks if there is an overlap between current and past execution cycles.
- give: Redistributes load from one processor to another.
- balance: Balances the load among a processor and its immediate neighbors.
- printLoads and printIntervals: Utility functions for outputting processor loads and execution cycles.

Core Simulation Logic:

- iterateOverIntervals: The main loop of the program. It iteratively balances loads among processors and updates their execution cycles, checking for a steady state or balanced condition.
- The simulation is performed for different scenarios, each with varying numbers of processors.

Main Function:

- Sets up different test cases with varying numbers of processors.
- Initializes processor loads and execution cycles.
- Calls iterateOverIntervals to run the simulation.

- Outputs the initial and final load distributions, as well as the number of stages required to reach a balanced state.

Results:

Number of Processors in VM: 8

For all Cases:
Minimum Load Value: 10
Maximum Load Value: 1000
Minimum Interval Value: 100
Maximum Interval Value: 1000

Results Summary:
The simulation was run for various numbers of processors (3, 5, 10, 50, 100, 250), with initial loads and execution cycles randomly assigned within specified ranges. The key results are summarized below:

- Case 1 (3 Processors): Reached a steady balanced state in 8 stages.
- Case 2 (5 Processors): Reached a steady balanced state in 21 stages.
- Case 3 (10 Processors): Reached a steady balanced state in 116 stages.
- Case 4 (50 Processors): Reached a steady balanced state in 14,802 stages.
- Case 5 (100 Processors): Reached a steady balanced state in 85,460 stages.

- Case 6 (250 Processors): Reached a steady balanced state in 432,279 stages.

In each case, the final load distribution was very close to being perfectly balanced, with only minor differences between processors.

Case 1: Number of Processors: 3
Initial Loads::[330, 651, 184]
Initial Cycles::[348, 828, 230]
Number of stages to reach steady balanced state: 8
Final Loads::[388, 389, 388]
Final Cycles::[1069, 1437, 1043]

Case 2: Number of Processors: 5
Initial Loads::[984, 56, 472, 255, 90]
Initial Cycles::[562, 884, 583, 506, 350]
Number of stages to reach steady balanced state: 21
Final Loads::[372, 371, 371, 372, 371]
Final Cycles::[2480, 2147, 2113, 2343, 2474]

Case 3: Number of Processors: 10
Initial Loads::[950, 935, 329, 673, 494, 771, 596, 604, 841, 403]

Initial Cycles::[750, 111, 342, 332, 680, 492, 398, 528, 439, 800]
Number of stages to reach steady balanced state: 116
Final Loads:: [660, 660, 660, 660, 659, 659, 659, 660, 659, 660]
Final Cycles::[6178, 5873, 5908, 6038, 6271, 6232, 6032, 6296, 6402, 6383]

Case 4: Number of Processors: 50
Initial Loads::[558, 523, 503, 821, 567, 127, 284, 209, 170, 969, 782, 992, 34, 745, 512, 558, 28, 511, 548, 972, 519, 664, 74, 382, 446, 463, 806, 182, 822, 481, 728, 848, 994, 231, 147, 38, 817, 421, 707, 455, 858, 957, 447, 360, 702, 949, 387, 198, 928, 925]

Initial Cycles::[971, 128, 368, 136, 549, 681, 418, 235, 593, 306, 673, 943, 812, 817, 488, 903, 349, 738, 171, 608, 769, 417, 478, 433, 227, 374, 698, 278, 778, 110, 327, 748, 138, 595, 180, 885, 573, 795, 416, 165, 397, 990, 404, 209, 202, 189, 408, 748, 827, 480]
Number of stages to reach steady balanced state: 14802

Final Loads::[547, 546, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547, 547]

Final Cycles::[171018, 171544, 171446, 171609, 170980, 171319, 171634, 170808, 170808, 170888, 171581, 171055, 170802, 170817, 171168, 171472, 170836, 170827, 171175, 170957, 171668, 170854, 170774, 170948, 171507, 170907, 171191, 170983, 171144, 171045, 171105, 170842, 171049, 170783, 171419, 170876, 171692, 171333, 170939, 171397, 170922, 171161, 171596, 170932, 171026, 171014, 170769, 170919, 170898, 171274]

Case 5: Number of Processors: 100
Initial Loads::[926, 875, 195, 486, 667, 936, 483, 199, 25, 245, 532, 666, 108, 236, 707, 895, 404, 871, 93, 36, 565, 788, 977, 199, 68, 994, 819, 478, 898, 554, 201, 823, 897, 386, 777, 42, 790, 260, 231, 283, 964, 231, 939, 540, 457, 645, 435, 330, 515, 987, 356, 548, 252, 333, 737, 310, 795, 555, 257, 170, 578, 448, 983, 943, 303, 238, 975, 92, 488, 675, 835, 920, 896, 773, 459, 822, 887, 362, 620, 870, 348, 966, 418, 591, 767, 623, 369, 39, 647, 94, 199, 224, 533, 651, 635, 304, 879, 88, 386, 835]

Initial Cycles::[818, 885, 982, 429, 872, 844, 640, 800, 782, 895, 941, 861, 405, 703, 865, 843, 354, 355, 996, 876, 737, 510, 205, 566, 357, 229, 432, 534, 863, 535, 454, 977, 716, 732, 406, 884, 576, 342, 683, 654, 237, 624, 514, 839, 623, 676, 681, 877, 931, 973, 752, 667, 483, 857, 530, 136, 383, 258, 571, 245, 693, 321, 519, 706, 954, 221, 886, 529, 463, 569, 479, 600, 489, 893, 735, 111, 568, 416, 889, 795, 388, 937, 462, 167, 794, 892, 204, 473, 149, 972, 915, 139, 292, 433, 141, 245, 554, 927, 971, 918]

Number of stages to reach steady balanced state: 85460

Final Loads::[537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537, 538, 537, 538, 538, 537, 538, 537, 538, 538, 537, 538, 537, 538, 538, 538, 537, 538, 538, 537, 538, 538, 538, 538, 537, 538, 537, 538, 538, 537, 538, 538, 538, 538, 537, 538, 538, 537, 538, 537, 538, 538, 537, 538, 537, 538, 537, 538, 537, 538, 537, 538, 537, 538, 537, 538, 537, 537, 537, 537, 537, 537, 537, 537, 537, 537]

Final Cycles::[512509, 511819, 512292, 512275, 511964, 512152, 512010, 512096, 512185, 512145, 512128, 512162, 511876, 511887, 512009, 511775, 512564, 512287, 511973, 511933, 511831, 511846, 511870, 511797, 512094, 512445, 512087, 511767, 511801, 511841, 511872, 512517, 512248, 511766, 511985, 511823, 512429, 512021, 512235, 512196, 512359, 511831, 512379, 511973, 512000, 512190, 511982, 511804, 512535, 511795, 511936, 511771, 512642, 511874, 512065, 512107, 512043, 512351, 512128, 512310, 511943, 511901, 511943, 512187, 511831, 512492, 512327, 512095, 511814, 512205, 511895, 512187, 512404, 512037, 512140, 512246, 512103, 511944, 512105, 512236, 511998, 512258, 511865, 512056, 512074, 512231, 511813, 511995, 512321, 511945, 512046, 512496, 511982, 512153, 511807, 511846, 511870, 511810, 512376, 511912]

Case 6: Number of Processors: 250
Number of stages to reach steady balanced state: 432279

Final Loads::[495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 496, 496, 495, 496, 496, 495, 496, 496, 496, 495, 496, 496, 496, 496, 495, 496, 495, 496, 496, 495, 496, 495, 496, 496, 496, 495, 496, 496, 495, 496, 496, 495, 496, 495, 496, 495, 496, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 496, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495]

Final Cycles::[1181740, 1181767, 1181966, 1181575, 1181481, 1181306, 1181931, 1181652, 1181438, 1181325, 1181963, 1181336, 1181394, 1181696, 1181467, 1181632, 1182141, 1181856, 1181418, 1181384, 1181393, 1181895, 1182202, 1182039, 1182148, 1181470, 1181925, 1181739, 1181511, 1182167, 1181316, 1181448, 1181693, 1181786, 1182253, 1181340, 1181387, 1181509, 1181476, 1181613, 1182139, 1182102, 1181910, 1182073, 1181434, 1181919, 1181956, 1181572, 1181806, 1182175, 1181422, 1181336, 1181818, 1182159, 1181443, 1181459, 1181338, 1181732, 1181719, 1181844, 1181524, 1181817, 1181455, 1181460, 1181480, 1181730, 1181935, 1181608, 1181552, 1182167, 1181922, 1181810, 1181693, 1181328, 1181669, 1181849, 1181319, 1181306, 1181868, 1181878, 1181588, 1181904, 1181704, 1181702, 1181676, 1181530, 1181585, 1181671, 1181864, 1182114, 1181471, 1181630, 1181370, 1181745, 1181796, 1181847, 1181675, 1181524, 1181813, 1181684, 1181758, 1181335, 1181379, 1181334, 1181422, 1181913, 1181666, 1181555, 1181753, 1181677, 1182177, 1181375, 1181759, 1181565, 1182163, 1181376, 1181437, 1181401, 1181438, 1181344, 1181508, 1181439, 1181694, 1181610, 1181839, 1181463, 1181881, 1181804, 1181610, 1181582, 1182177, 1181673, 1181744, 1181782, 1181860, 1181544, 1181947, 1182051, 1181863, 1181541, 1181919, 1181704, 1181511, 1181373, 1181490, 1181553, 1181325, 1181745, 1181652, 1181542, 1181429, 1181347, 1181564, 1181372, 1181429, 1182230, 1181485, 1181468, 1181432, 1181454, 1181677, 1181499, 1181827, 1181538, 1182234, 1181553, 1181358, 1182131, 1181380, 1181418, 1182007, 1181473, 1181401, 1182210, 1181496, 1181860, 1181443, 1181415, 1181706, 1181657, 1181672, 1181351, 1181545, 1181627, 1181913, 1181711, 1181443, 1181587, 1181348, 1181549, 1181337, 1181337, 1181758, 1181310, 1181458, 1181507, 1181581, 1182016, 1181467, 1181877, 1181557, 1181766, 1181335, 1181643, 1181969, 1181436, 1181410, 1181384, 1181856, 1181922, 1181340, 1181315, 1182151, 1181682, 1181910, 1181473, 1181371, 1181326, 1181374, 1181947, 1181886, 1181481, 1181452, 1181810, 1181750, 1181618, 1181695, 1182103, 1181626, 1181515, 1181497, 1181495, 1181677, 1181526, 1181612, 1181470, 1181590, 1181811, 1181556, 1181532, 1182065, 1181903, 1181390, 1181766, 1181669, 1181500, 1181457, 1181465, 1181663, 1181342]

What is your definition of “balanced among neighbors”?
In this program, a processor is considered "balanced among neighbors" when it successfully redistributes its load with its immediate neighbors (left and right) to achieve a load distribution where each processor has a load close to the average load of these three processors.

What is your definition of the system in “steady state”?
The system is in a "steady state" when there are no significant changes in the load distribution over time. In the program, this is checked using the checkLoad function, which compares the current and past loads of processors. If the loads remain unchanged, or if the maximum difference in loads between any processors does not change significantly (as checked by isMaxDiff), the system is considered to be in a steady state.

What is your definition of the system in “balanced state”?
The program defines the system as being in a "balanced state" when the maximum load difference between any two processors is less than or equal to 1. This is determined by the isMaxDiff function. It implies that the load is as evenly distributed as possible across all processors in the system.

Does the proposed strategy converge to a balanced state?
- Observation from Results: Yes, the proposed strategy does converge to a balanced state. In all cases (ranging from 3 to 250 processors), the system reached a state where the load difference between processors was minimal (less than or equal to 1).
- Effectiveness of Strategy: The gradual reduction in load difference across processors, as observed in the final loads of each case, indicates that the strategy is effective in redistributing the load evenly among processors.

Can you prove that the strategy will con- verge to a balanced state for any possible initial load-units distribution?
- Empirical Evidence: The results from the simulations suggest that the strategy works for a wide range of initial conditions, as it successfully reached a balanced state in all tested scenarios.
- Theoretical Proof: Providing a mathematical proof for convergence under any possible initial load distribution is challenging. Such a proof would require a comprehensive analysis considering all potential initial states and their evolution, which is complex given the dynamic nature of load balancing in distributed systems. The code and results provide strong empirical evidence but do not constitute a formal proof of convergence for all possible initial conditions.

What could be the “worst” possible initial load-units assignment?
- Based on Simulation Results: The "worst" initial load assignment would likely be one where the discrepancy between the highest and lowest loads is at its maximum. This

would mean one or more processors are overloaded (near the maximum load value), while others have minimal loads.

- Challenging Scenarios: Larger numbers of processors and greater variances in initial load distributions (as seen in cases with more processors) increase the number of stages required to achieve balance, indicating that these scenarios are more challenging for the load balancing strategy.
- Specific Example: In a practical sense, an initial state where loads are alternately distributed at maximum and minimum values across the processor ring would likely be very challenging and could represent a "worst-case" scenario, requiring a significant number of iterations to balance.