CEL72, DSL, Monsoon 2020

Lab 5: Implementation of Linear and Fibonacci Load Balancing

Batch A, Student ID: 2017130011

**Objective**

Implementation of Linear and Fibonacci Load balancing.

**Introduction**

The main purpose of load balancing is to efficiently distribute the total load among the individual nodes so that resource can be effectively utilized and response time of a job can be reduced. Load balancing tries to eliminate any possible condition in which some of the nodes have to carry load that are beyond its capacity while some others nodes are still underutilized. Any algorithm for balancing the load must be dynamic in nature which must not consider the any previous state or behaviour of the system i.e. Load balancing algorithms are only depended upon the present state and behaviour of the system. The most essential part of any load balancing algorithm is to contemplate the following methodology:

1. how to estimate the load.

2. How to analyse of load?

3. Procedure to measure the stability of divergent system.

4. How to measure the performance of system?

5. How different nodes interact among themselves?

6. Selection of target nodes and many others.

The parameters of load can be any one or combination of CPU load, delay, network load or amount of memory used etc.

**Problem Statement**

To implement Linear and Fibonacci Load balancing schemes and compare the performance.

\*\* The evaluation is done on the basis of the average waiting time for a process(task).

**Conceptual architecture of the distributed system you have considered**

Number of processors: 9

Processors (processing power in GBs): [1,1,2,4,6,8,10,10,12]

Processor ranking (on the basis of processing power): [9,8,7,6,5,4,3,2,1]

Linear sequence approach for process allocation to the processors: [1,2,3,4,5,6,7,8,9]

Fibonacci sequence approach for process allocation to the processors: [1,1,2,3,5,8,13,21,34]

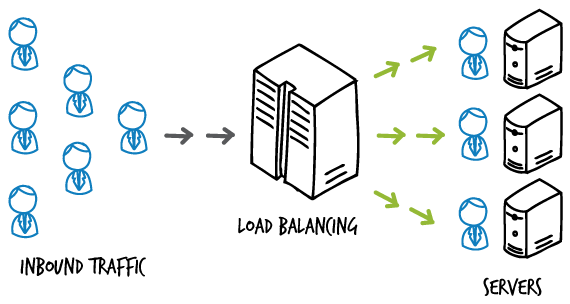


Fig.1 Load Balancer conceptual diagram

**Show process/event call steps for algorithm that you will implement**

**Source Code:**

#!/usr/bin/env python

# coding: utf-8

from collections import defaultdict

processors = [1,1,2,4,6,8,10,10,12]

ranks = [9,8,7,6,5,4,3,2,1]

linear = [1,2,3,4,5,6,7,8,9]

fibo = [1,1,2,3,5,8,13,21,34]

def LinearLoadBalance(processors,linear,timeslots,new\_tasks):

taskpool = []

processlists = defaultdict(list)

waittime = defaultdict(lambda :0)

for i in range(new\_tasks):

taskpool.append(i)

while len(taskpool) > 0:

'''

Assign tasks

'''

counter = 0

for i in range(len(linear)):

for j in range(linear[i]):

if counter < len(taskpool):

processlists[i].append(taskpool[counter])

counter += 1

taskpool = taskpool[counter:]

'''

Processing tasks

'''

for i in range(len(linear)):

prcs\_count = 0

while len(processlists[i]) > 0 and prcs\_count < processors[i]:

processlists[i].pop(0)

prcs\_count += 1

for i in range(len(linear)):

for j in processlists[i]:

waittime[j] += 1

if len(taskpool) == 0:

break

if timeslots > 1:

timeslots -= 1

last = taskpool[-1]

for i in range(new\_tasks):

taskpool.append(last+i)

return waittime

def FiboLoadBalance(processors,fbo,timeslots,new\_tasks):

taskpool = []

processlists = defaultdict(list)

fibo\_waittime = defaultdict(lambda :0)

for i in range(new\_tasks):

taskpool.append(i)

while len(taskpool) > 0:

'''

Assign tasks

'''

counter = 0

for i in range(len(fibo)):

for j in range(fibo[i]):

if counter < len(taskpool):

processlists[i].append(taskpool[counter])

counter += 1

taskpool = taskpool[counter:]

'''

Process tasks

'''

for i in range(len(fibo)):

prcs\_count = 0

while len(processlists[i]) > 0 and prcs\_count < processors[i]:

processlists[i].pop(0)

prcs\_count += 1

for i in range(len(fibo)):

for j in processlists[i]:

fibo\_waittime[j] += 1

if len(taskpool) == 0:

break

if timeslots > 1:

timeslots -= 1

last = taskpool[-1]

for i in range(new\_tasks):

taskpool.append(last+i)

return fibo\_waittime

avg\_wt\_fibo = []

for i in range(1,100):

wtf = FiboLoadBalance(processors,fibo,i,100)

avg\_wt\_fibo.append(sum(wtf.values())/len(wtf))

avg\_wt\_linear = []

for i in range(1,100):

wtl = LinearLoadBalance(processors,linear,i,100)

avg\_wt\_linear.append(sum(wtl.values())/len(wtl))

for i in range(1,100):

print("100 new processed added to the pool {} times \nWait time: {}".format(i,avg\_wt\_linear[i-1]))

print("\n")

for i in range(1,100):

print("100 new processed added to the pool {} times \nWait time: {}".format(i,avg\_wt\_fibo[i-1]))

print("\n")

**Sample Output**

Linear load balancing:

100 new processed added to the pool 1 times

Wait time: 1.0909090909090908

100 new processed added to the pool 2 times

Wait time: 1.4285714285714286

100 new processed added to the pool 3 times

Wait time: 1.8064516129032258

100 new processed added to the pool 4 times

Wait time: 2.1951219512195124

100 new processed added to the pool 5 times

Wait time: 2.8181818181818183

100 new processed added to the pool 6 times

Wait time: 3.230769230769231

100 new processed added to the pool 7 times

Wait time: 3.6266666666666665

100 new processed added to the pool 8 times

Wait time: 4.023529411764706

Fibonacci load balancing:

100 new processed added to the pool 1 times

Wait time: 1.3055555555555556

100 new processed added to the pool 2 times

Wait time: 1.5670103092783505

100 new processed added to the pool 3 times

Wait time: 1.75

100 new processed added to the pool 4 times

Wait time: 1.9607843137254901

100 new processed added to the pool 5 times

Wait time: 2.179640718562874

100 new processed added to the pool 6 times

Wait time: 2.401937046004843

100 new processed added to the pool 7 times

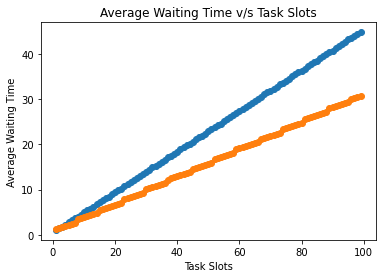
Wait time: 2.6260162601626016

100 new processed added to the pool 8 times

Wait time: 3.3960923623445827

**Test Procedure for objective validation:**

Average waiting time for a processes.

****

**Orange Points:** Linear load balancing scheme.

**Blue Points:** Fibonacci load balancing scheme.

**Conclusion:**

The Fibonacci load balancing scheme leads to much more efficient allocation of processes to processors with respect to Linear load balancing as the waiting time for the processes is reduced.

**Evaluation**

Grading is performed for this assignment.

|  |  |
| --- | --- |
| **Election Algo laboratory work grading details Points 5 + Instructor Signature** | **Requirements** |
| 5 | * Fibonacci and Linear load balancing with suitable system model (1M) * Performance Analysis w.r.t. given questions: (2M) * Documentation (2M) |

|  |  |
| --- | --- |
| **Feedback** | **Yes/No** |
| * Contents in this write up has been useful to perform experiment? * Level of understanding DS has improved? | Yes  Yes |