Neural Network based parallel computation in distributed system environment

Under the Supervision of:
Savitha S.

TEAM :
Rakesh Pavan (17IT154)
Yogesh Choubey (17IT252)
Dhruvik Navadiya (17IT225)
Neeraj (17IT226)

Abstract

Modern world technologies and problems require heavy computations and huge processing power to solve. Single system cannot be configured to such a degree of complexity. Thus people go for distributed and cluster systems and divide the problem (or load) into sub-problems and solve them in parallel. In this project we try to address an important problem involved in parallelizing a task, i.e to efficiently find an optimal distribution of the given computational load in a way that yields minimum processing time, making effective use of resources available. We use a neural network based approach trained on ground truth generated by a divide & conquer based ternary search algorithm, to find an optimal way to distribute the given load on the basis of dynamic system parameters fed as features. A comparative performance study against some common existing methods show that our approach performs better in terms of effectiveness and efficiency.

Literature Review:

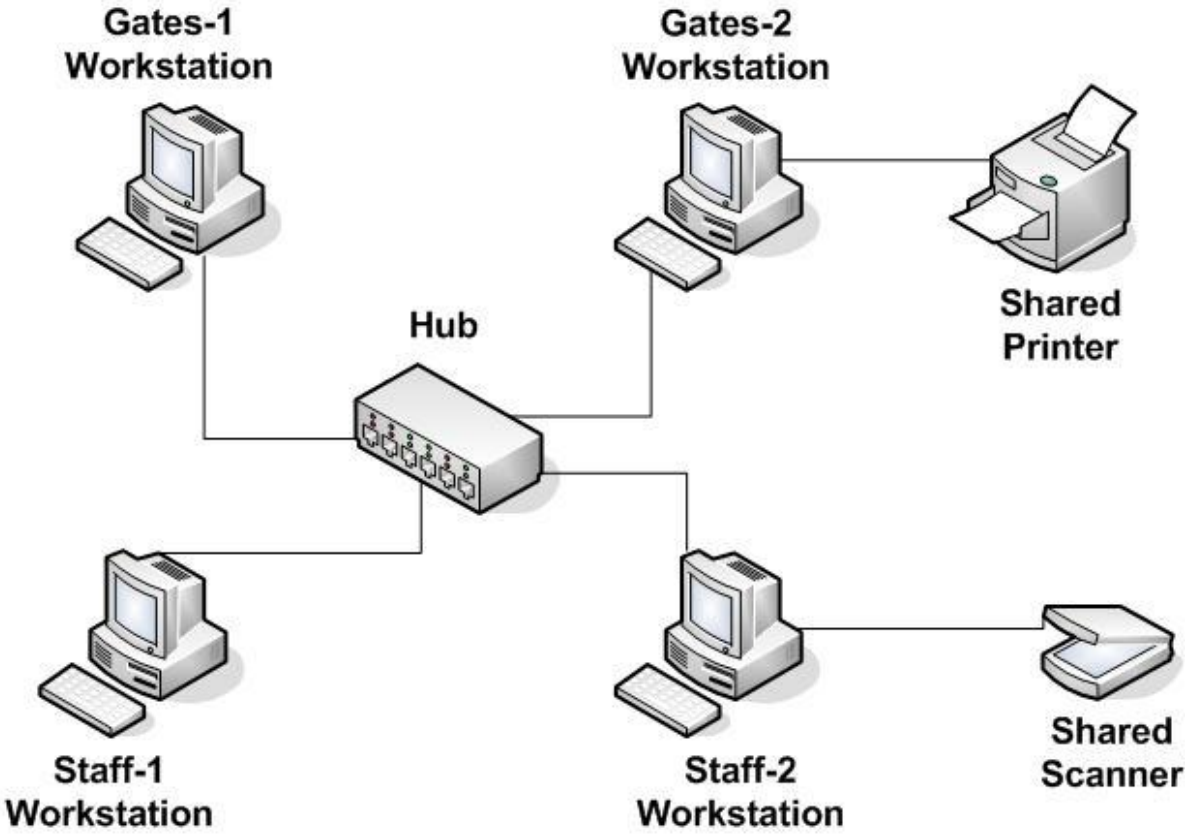
No.	Title	Year	Advantages	Shortcomings
1	“**Load balancing using process migration for linux based distributed system**” IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)	2014	-Technique to migrate process to another system on reaching a threshold load for a particular system	-Not taking into consideration the state of the other systems in the cluster - Initial distribution of the load is not defined
2	“**Effective Techniques for Message Reduction and Load Balancing in Distributed Graph Computation**” , ACM Proceedings of the 24th International Conference on World Wide Web	2015	-Effective and efficient graph based interpretation and algorithms for balancing and message reduction in social and comm. network	-Algorithm not specifically applied for distribution of computational loads in Client Server based architectures, or other common distributed systems
3	“**Optimizing Machine Learning Algorithms on Multi-Core and Many-Core Architectures Using Thread and Data Mapping** ” , 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)	2018	- Applied modern machine learning techniques combined with effective data mapping and threading	-Not applied for distributing load in network separated client server architectures or distributed systems. -Not considering CPU and Memory state for division of load.

Procedure followed

1. Creating the Client-Server program in Java using multi threading and socket programming for interprocess communication.
2. Generating the ground truth optimal load distribution using ternary search.
3. Finding the time stamps for understanding the nature of the load. And also calculating the dummy computation for normalization troughout the clients.
4. Gathering information about the client’s current system state using bash script.
5. Training the ML model using the generated data of step 2 and step 3 and outputting the 4 parameters for distribution.
6. Comparing the results by equally distributing load over the 4 system.

Creating the Client Server Architecture

Peer-to-Peer Network



1. Socket programming was used to establish connection between different nodes in the system.
2. Client Server Communication was carried by TCP IP protocol.
3. Use of Java Multithreading enabled handling of multiple client on a single server.

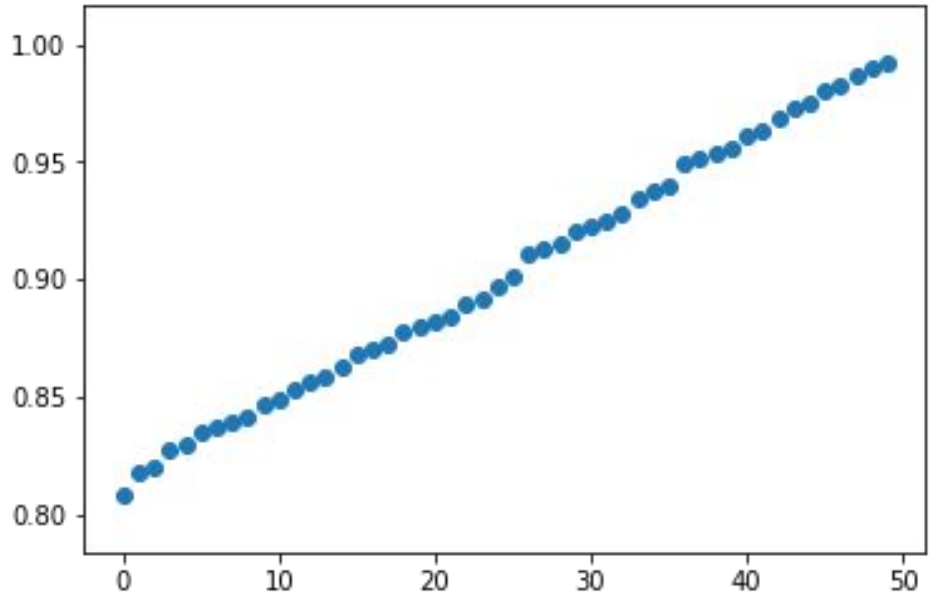
About the Load :

For our study we used a linear load, i.e calculating factorial of a number under modular arithmetic.

Fn = (n*Fn-1)%mod, where mod = 1e9+7

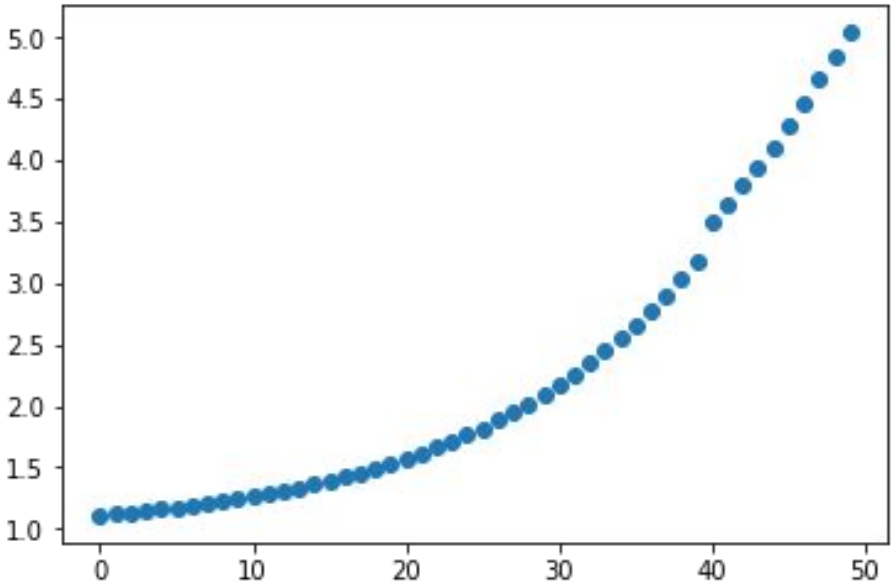
Load Characteristic

Linear Load



Load Characteristic

Quadratic Load



Generating the ground truth

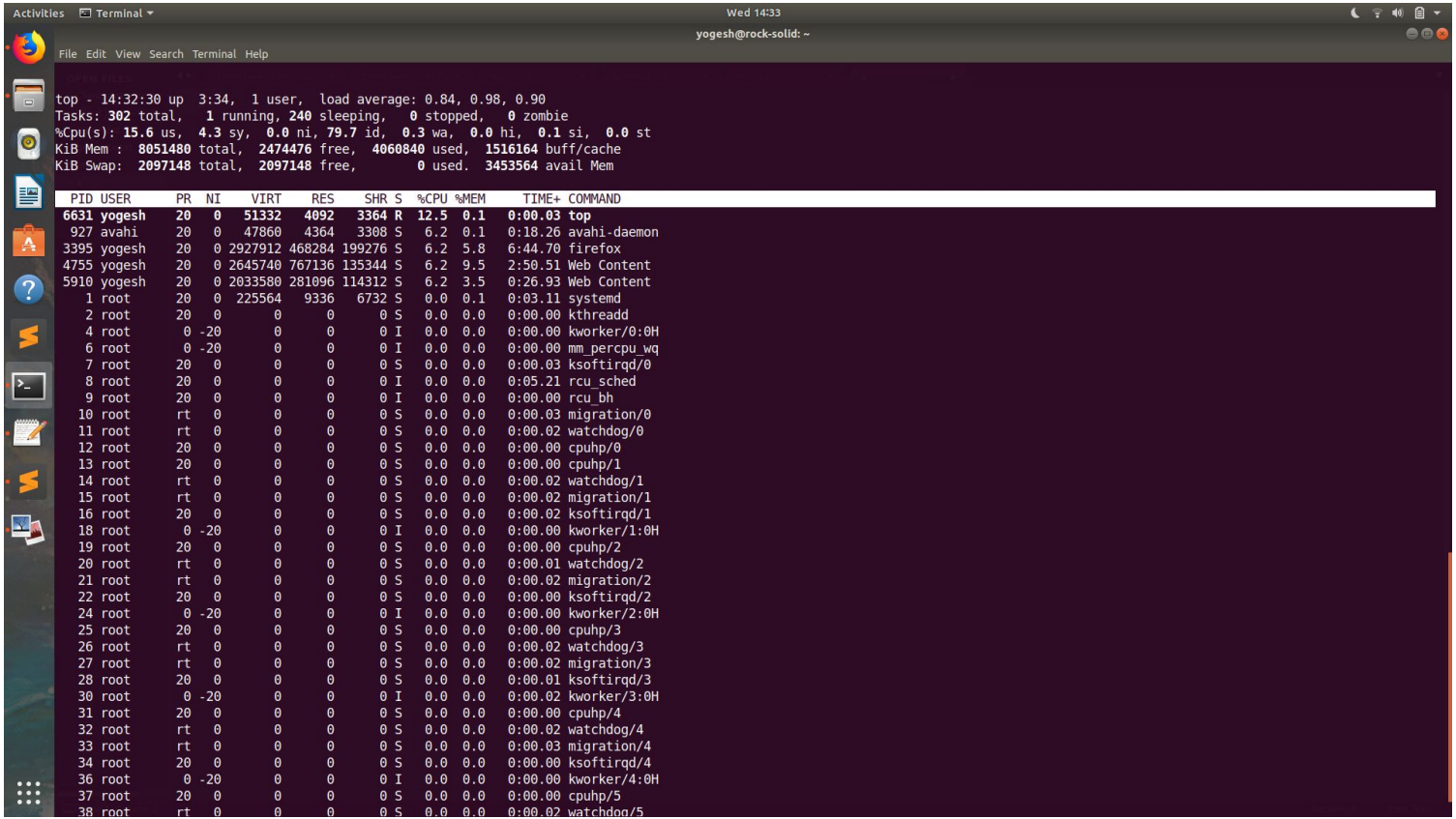
Divide & Conquer based Ternary search was used to find out the most optimal load distribution (load distribution that minimizes the overall time for execution of the complete program on parallel processor).

```
getOptimalTime()
    minTime = INF
    for x = 0 to x = 100
        for y = x to y = 100
            low = y, high = 100, mid1 = low + (high-low)/3, mid2 = low + 2*(high-low)/3
            while(mid2 - mid1 < 1)
                t1 = getTime(x, y-x, mid1-y)
                t2 = getTime(x, y-x, mid2-y)
                if(t1 > t2)
                    low = mid1
                else
                    high = mid2

            minTime = min(minTime, t1)

    return minTime
```

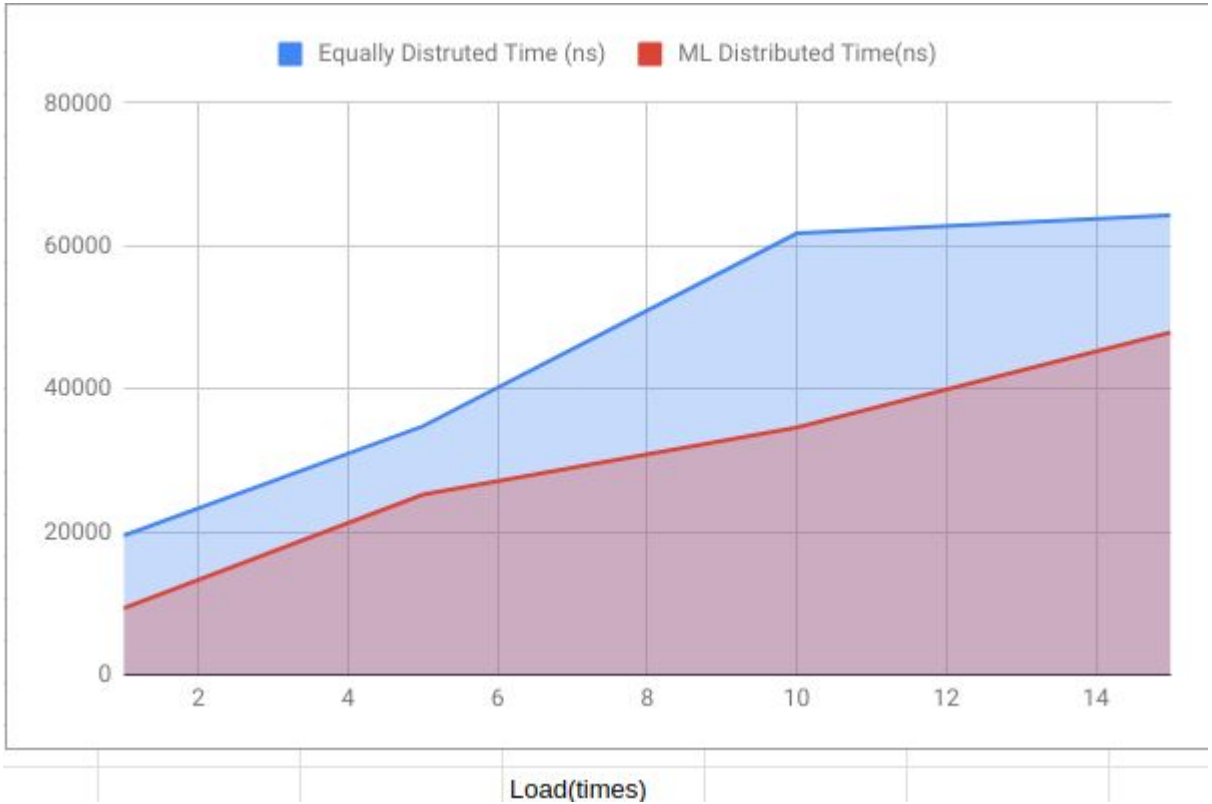
Gathering information using bash script



Training the ML model

We have used a neural network having a single hidden layer with 4 neurons, and activation set as relu activation. The input layer is a feature vector of 149 features, and the output layer is a softmax distribution of size 4 which gives the optimal distribution for the given feature input.

Comparing the results



Comparing the results

Load	Computation Time with our Method (ns)	With Existing Method (ns)	Percentage Improvement
1x	9289	19466	52.280903%
5x	25188	34777	27.572823%
10x	34574	61745	44.005184%
15x	47876	64277	25.516125%

Few Other References :

1. https://www.sciencedirect.com/science/article/pii/S2212017313005318
2. http://www.cs.nuim.ie/~dkelly/CS402-06/Process%20Load%20Distribution.htm
3. http://vega.cs.kent.edu/~mikhail/classes/aos.f03/l15load.PDF
4. https://ieeexplore.ieee.org/abstract/document/8421895
5. https://dl.acm.org/citation.cfm?id=3230906
6. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7342686

Thank You