SAPIENZA
UNIVERSITÀ DI ROMA

Networking for Big Data

Data Center Module

Gray

Challenge 1: Topology design

Professor:
Andrea Baiocchi

Himel Ghosh - 2102750
Francesco Lazzari - 1917922
Giorgio Bertone - 2112729
Marina Zanoni - 1964213

Academic year 2023/2024

1 Random Graphs

1.1 Complexity measure

In order to measure the complexity as a function of the number of nodes (K) of the three methods, we generated the p-ER Graph and then ran the 3 connectivity checking algorithms: a) irreducibility, b)eigenvalue of the Laplacian matrix, and c) breadth-first searc. Then, we used the run-time of these algorithms as a proxy for the complexity measure. As shown in **Fig.1**, method a) has, as expected, the worst performance since its complexity is $O(K^3 \cdot log(K))$. While, the best performing one is method c) that has computational complexity $O(K + E)$, where E is the number of edges in the graph.

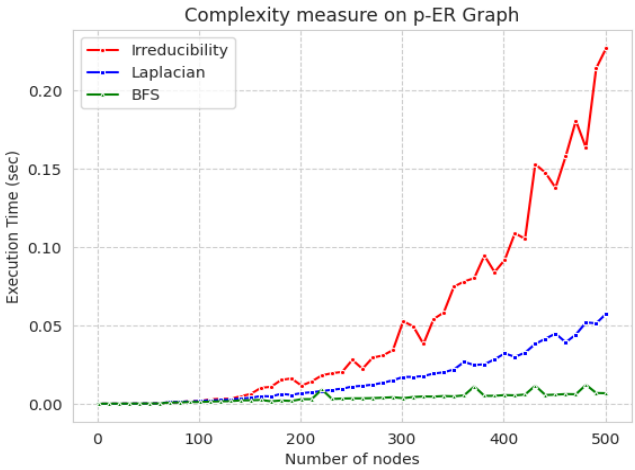

Figure 1: *Complexity Measure: ER Graph with p=0.2*

1.2 Connectivity Probability

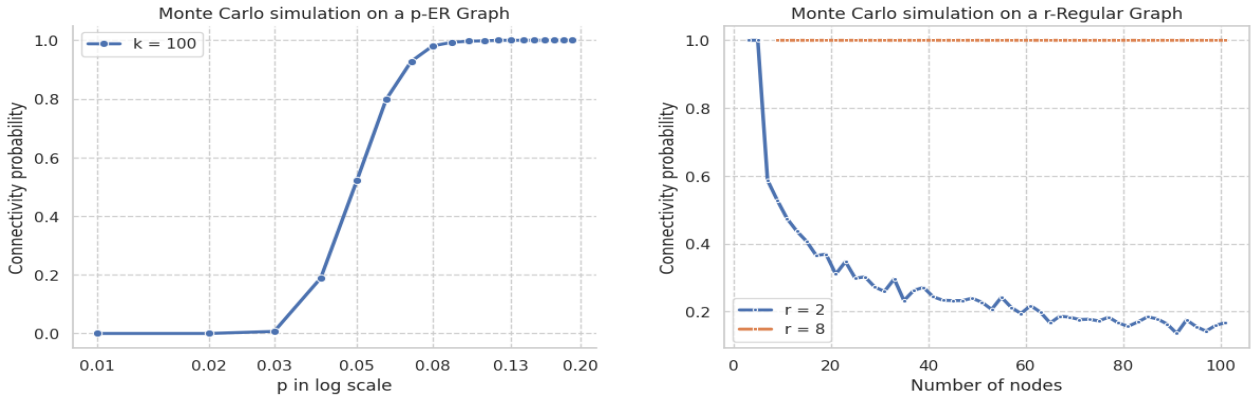

Figure 2: *Connectivity Probability: ER vs r-Regular Graph*

We used Monte-Carlo simulations to analyze connectivity probabilities in both p-Erdős-Rényi and r-Regular graphs with varying numbers of nodes (up to 100). In the p-Erdős-Rényi graph, connectivity probability increases gradually from nearly 0 to 1 as p ranges from 0.03 to 0.08, marking a transition from sparse to dense connectivity. At the critical point of $p = \frac{ln(k)}{k} = \frac{ln(100)}{100} = 0.046$, a phase transition occurs. In the r-Regular graph with r=2, connectivity probability decreases as the number of nodes increases, while for r=8, the graph remains almost surely connected, with a constant probability of 1. See [**Fig.2**].

2 Computation job: local versus distributed run

2.1 Formal Statement: Evaluation of Mean Response Time

Algorithm 1 Pseudocode for Simulating Mean Response Time in Network Topologies

function CALCULATE_THROUGHPUT(N, τ, C)
    $rtt \leftarrow$ array of size N initialized with zeros
    for $i \leftarrow 1$ **to** N **do**
        $rtt(i) \leftarrow 4\tau 1_{[1,31]}(i) + 8\tau 1_{[32,1024]}(i) + 12\tau 1_{[1025,N]}(i)$
        $th \leftarrow C \cdot \frac{1/rtt}{\sum_{i=1}^{N} 1/rtt}$
    end for
    return th
end function
$L\_oi \leftarrow$ array of size N with $L_{oi} \sim Unif\left(0, \frac{2*Lo}{N}\right)$
$th \leftarrow$ CALCULATE_THROUGHPUT(N)
function CALCULATE_RESPONSE_TIME(N, Ex)
    $R1 \leftarrow T0 + X_i$ where $X_i \sim exp(\frac{E_X}{N})$
    $R2 \leftarrow \frac{L_f(f+1)}{N \cdot th}$
    $R3 \leftarrow \frac{(L_{oi}+f \cdot L_{oi})}{th}$
    $R \leftarrow R1 + R2 + R3$
    return R
end function
function SIMULATION(M, N)
    $R \leftarrow$ array of size M initialized with zeros
    for $i \leftarrow 0$ **to** $M - 1$ **do**
        $R[i] \leftarrow$ CALCULATE_RESPONSE_TIME(N)
    end for
    return $E[R]$
end function

For the evaluation of the mean response time, we designed a Monte Carlo Simulation strategy as stated in the pseudocode above. In the distributed network, the data of size L_f is distributed from the server A to the N nearest servers and then back to A, it is necessary to identify the bottlenecks in both cases. Both $R2$ and $R3$ are the response times in the respective cases. $R1$ is the time to run the job in each of the N servers. Then, cumulatively, the response time is $R = R1+R2+R3$. However, $R2$ & $R3$ TCP transmission times depend on the throughput of the link which further depends on the distance (hops h_i) of the servers from each other. The number of hops is deterministic based on the total number of servers and ports considerations. We leverage the structure of the network topologies, to determine the hops. Then we use this data

Topology	2hops	3hps	4hops	6hops	
Fat-Tree	0-32	–	32-1024	1024+	servers
Jelly-Fish	0-32	32-1024	1024+	–	servers

to formulate the algorithm to simulate the Fat Tree & Jellyfish topology. Furthermore, running the Monte-Carlo simulation, we obtain the mean response times for the number of servers ranging from 1 to 10000. When run on the non-distributed scenario, we have the baseline value given, which we use to normalise the estimates we retrieved from the Monte-Carlo simulations.

2.2 Performance Insights

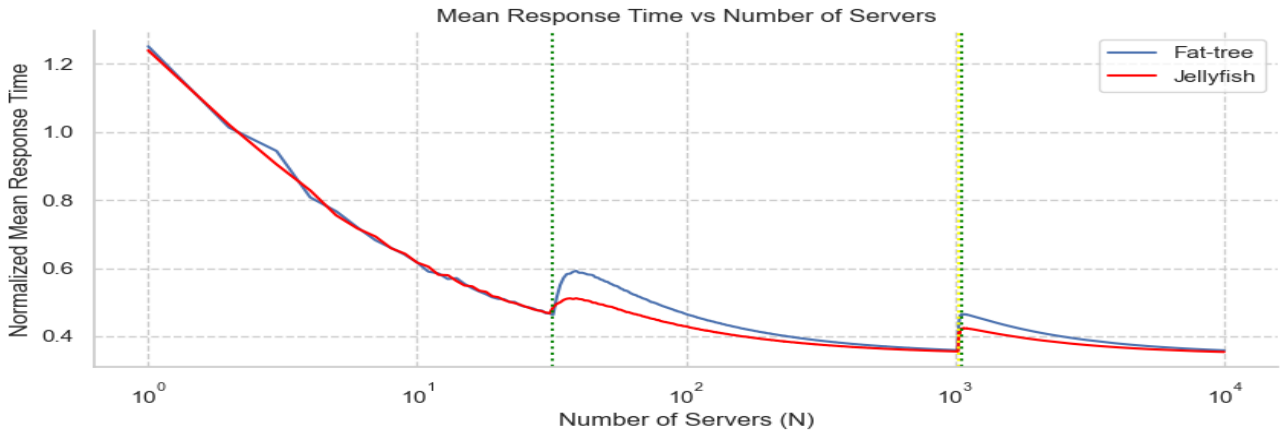

Figure 3: *Mean Response Time vs N*

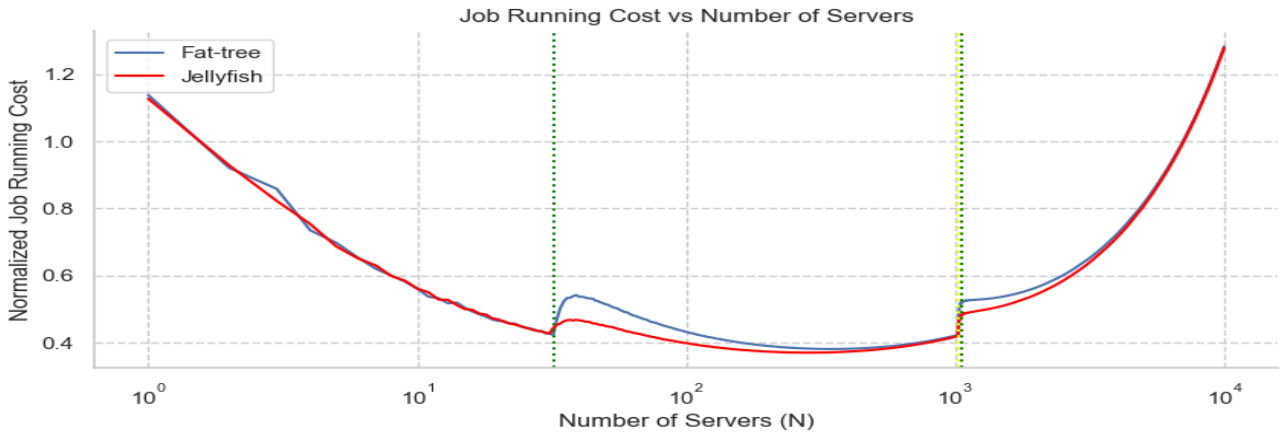

Figure 4: *Job Running Cost vs N*

The optimal number of servers that minimises the job running cost is 333 for Fat-Tree and 270 for Jelly-Fish Topology.

2.3 Results & Analysis

In the **Fig.3**, the Mean Response Time exponentially decreases with the increase in the number of servers because the execution times have a negative exponential distribution with mean $\frac{Ex}{N}$, but we observe spikes in the curve at two points, which is essentially contributed by the increasing distance between the servers of the different racks. In the same rack, both Fat-Tree and Jelly-Fish topology network have a similar behaviour, but in different racks, the distance between the servers is higher for Fat-Tree giving a higher spike compared to that of Jelly-Fish. Hence, parallel distribution of the job among N closest server renders a lesser time consumption with a higher value of N upto a certain extent. **Fig.4** clearly indicates that the Job Running Cost incurred, rather increases with a larger value of N, after an initial decrease. Because, a very high number of server means quite a large amount of setup time for each server in all the racks cumulatively dominates after an optimal value. Upto 333 servers in Fat-Tree, upto 270 servers in Jelly-Fish topology can be used optimally. This trade-off exists between the mean response time and the cost of usage of all the resources for the distribution among the N servers.