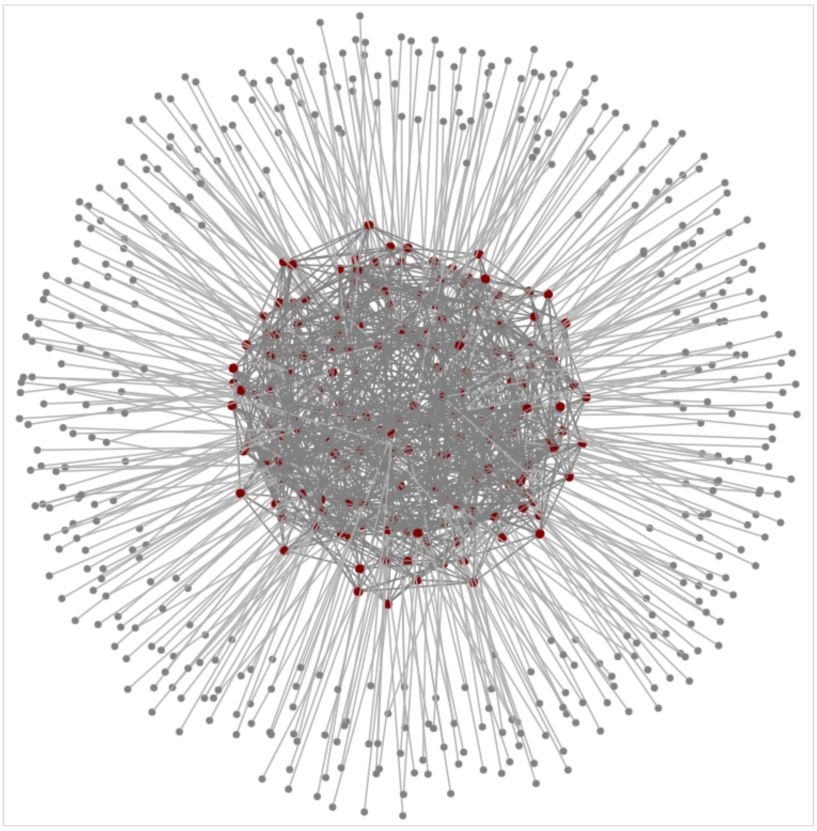
Data Center - Challenge 1

Group Abramson

April 2023

Group members

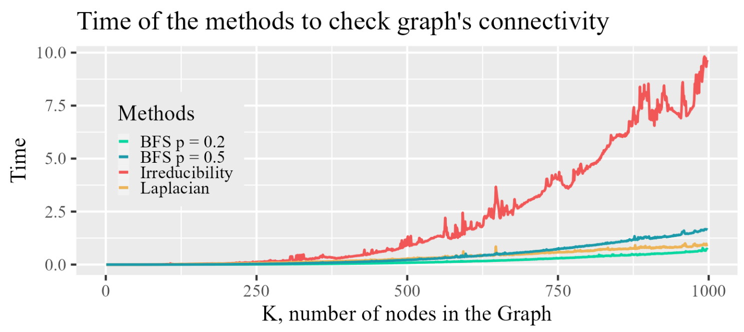
Susanna Bravi (1916681)
Simone Facchiano (1919922)
Maria Vittoria Vestini (1795724)



1 Challenge part 1

1.1 Plot 1: Comparisons of the time complexity of the three methods for checking the connectivity of the graph

Figure 1



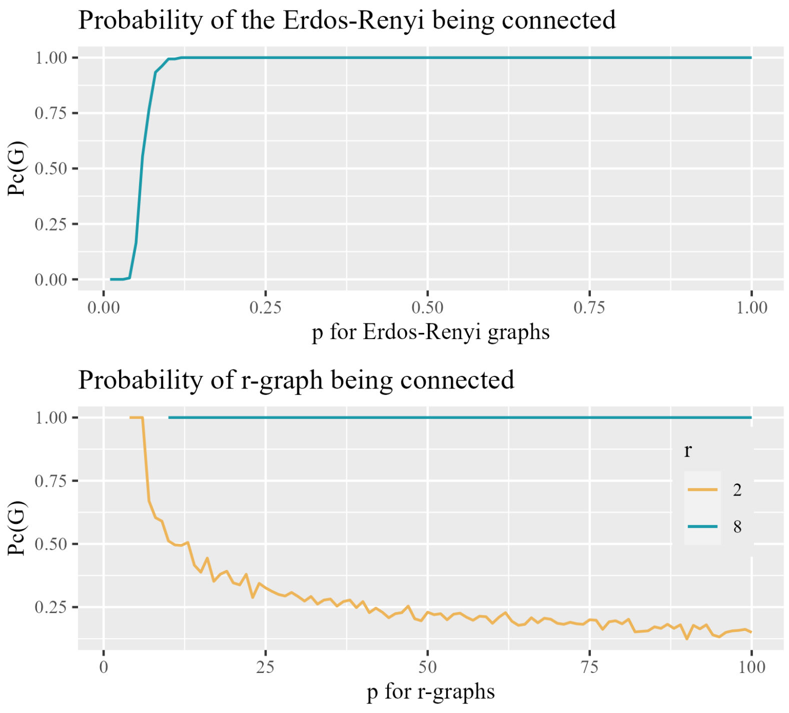
As can be seen from the graph, created using an ER graph, there is a substantial difference between the proposed methods. For **BFS**, two separate curves were used (for $p = 0.2$ and $p = 0.5$), since this method depends not only on the number of nodes but also on the number of edges. For the other two methods there was no need to make the same distinction since their execution time depend exclusively on the number of nodes.

The **Irreducibility** method has a computational complexity of $O(n^4)$ and it is the least efficient in every case. Therefore, the best methods seem to be **BFS** and **Laplacian**, but it is not possible to arrive at a final result given the dependence of the **BFS** on the number of arcs.

1.2 Plot 2: Probability of a connected ER random graph as a function of p for K equal to 100 nodes

From Erdős-Rényi's theory of random graphs, it is known that when $p > \frac{(1-\epsilon)ln(n)}{n}$, where ϵ is an arbitrarily small constant, then the graph will be almost surely connected. Indeed, in this graph, one can observe the presence of a sharp threshold at $p = \frac{ln(n)}{n} = \frac{ln(100)}{100} = 0.046$. So if the probability of forming an edge between two nodes in the ER-graph is greater than 0.046, the graph will be almost surely connected.

Figure 2



1.3 Plot 3: Probability of a connected r-regular random graph as a function of K for r = 2 and r = 8

For the r-regular graph it is possible to see two different trends. In the case where r, that is the degree of each node, is equal to 8 the graph turns out to be always connected for $K \leq 100$, where K is the number of nodes. However for what it concerns $r = 2$, i.e. the case where there is usually a ring-shaped topology, connectivity is not ensured. For $K \geq 6$, indeed it can happen that more connected components are formed rather than a single one, and this tendency becomes more and more pronounced until the probability of having only one connected component approaches 0.

2 Challenge part 2

2.1 Equation and algorithm

Given a job that a server A needs to do, it can be done by server A locally in $T_0 + X$, where T_0 is a fixed **set-up time** and X is a random variable, or by splitting the task across K servers.
For each server, the response time is just the sum of two components: the **computational job time** $T_0 + X_i$ and the time to transfer the amount of data with an **overhead** (which is a fraction of the original data), $\left(\frac{L_f}{K} + L_{o,i}\right)(1+f)$, considering a **throughput** of $TH_i = C\frac{1/T_i}{\sum_{j=1}^{K} 1/T_j}$.
The throughput is the rate at which data are transferred, and it takes into account the distance (hops) between A and the server i, since $T_i = 2\tau h_i$.
So, in case of parallelizing the job across K servers, the final output will be available only when all the servers have sent back their final data $L_{o,i}$ to A. Therefore the overall response time is the highest response time between all the K servers:

$$R = \max_i \left(T_0 + X_i + \frac{\left(\frac{L_f}{K} + L_{o,i}\right)(1+f)}{TH_i} \right) \tag{1}$$

This quantity will be computed n times for each K, and then the average of the results will be taken, as a proxy of the Mean Response Time $E[R]$.

2.2 Plot of the mean response time

The mean response time was studied for both the **Fat-Tree** and **Jellyfish** topologies (keeping the number of servers unchanged) as a function of K for K ranging between 1 and 10000, and the results are shown in the plot below:

Figure 3

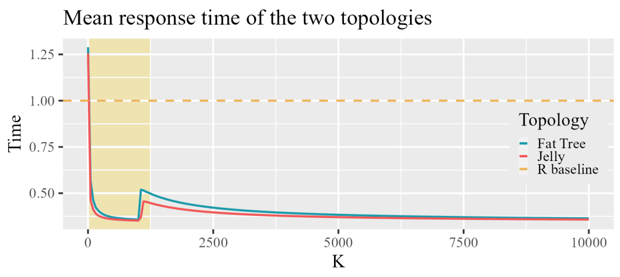
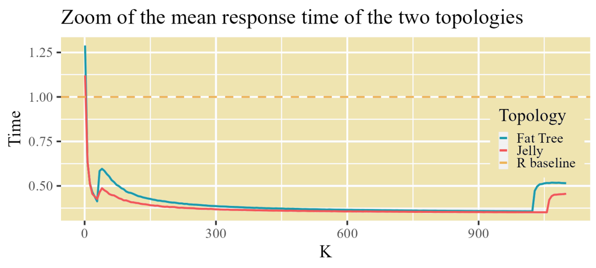


Figure 4



From the graph, it seems that parallelizing a task across multiple servers is convenient. In fact, both the Jellyfish and Fat-tree curves are below the baseline curve representing local execution time.

However, increasing the number of servers does not necessarily correspond to a smaller Mean Response Time.
In fact, as can be seen from Figure 3, there are some points at which the Mean Response Time suddenly increases. This is because for a number of servers equal to 1024 and 1055, for the Fat-tree and Jellyfish respectively, servers at distance h_1 ($h = 3$ for the Fat-tree and $h = 4$ for the Jellyfish) from A end and those at distance h_2 (with $h_1 < h_2$) begin to be taken.
The same thing happens for K = 31, but with a less pronounced effect (as can be seen from Fig. 4).
After these two jumps, the Mean Response Time decreases as K increases. This behavior is due especially to the decreasing execution time of individual machines, which is distributed according to a negative exponential with mean $\frac{E[X]}{K}$. The response time will have to take into account the maximum of these times X_i, which has a trend described by $\frac{log(K)}{K}$, where $log(K)$ is derived from higher sample size and $\frac{1}{K}$ from decreasing mean.
However, this trend appears slower and slower, until it settles down almost completely. This occurs

because although there are multiple servers that can share the computational work, there is always a fixed component of time T_0 that remains the same for each server.
Therefore we can state that for a very large number of servers parallelization stops being convenient because the cost of the physical resources required is not justified by actual time gain.

From the plot we can also observe how the Jellyfish topology is more time-efficient than the Fat-tree topology. Although from the graph the difference seems negligible, this is due to the fact that we are reasoning in relative terms given the normalization: in absolute terms the difference is on the order of hundreds of seconds.
Another advantage of the Jellyfish structure is also related to greater efficiency in resource allocation, since it requires fewer switches than the Fat-tree.

2.3 Plot of the Job running cost

The Job Running Cost S is defined as $S = E[R] + \xi E[\Theta]$.
$E[R]$ is the mean response time, while Θ is the sum of the times that all K servers need to complete their respective tasks. Therefore, if we do the math we discover that:

$$\begin{aligned} E[\Theta] &= E[K \cdot (T_0 + X_i)] = E[KT_0] + E[KX_i] \\ &= KT_0 + KE[X_i] = KT_0 + E[X] \end{aligned} \tag{2}$$

So, this $E[\Theta]$ quantity grows linearly with K, and therefore the trend of the Job Running time will look like this:

Figure 5

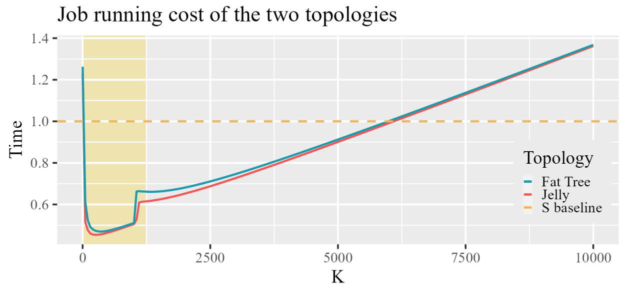
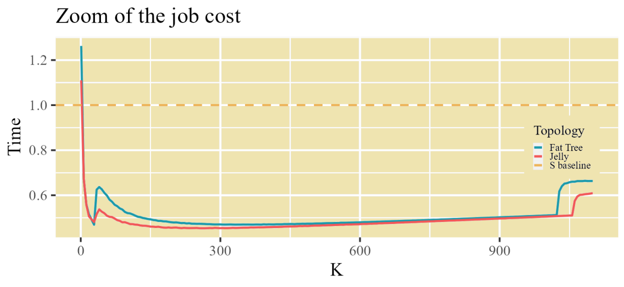


Figure 6



From these graphs it is possible to recognize the two components, $E[R]$ and $E[\Theta]$.
In fact, in the first part of the plot the trend is the same as that of Mean Response Time, while in the second part the linear nature of Job Running Cost is very clear.
Furthermore from the graph it can be seen that when K is over 6000, the "total time" of the K servers exceeds that of the single server A.

2.4 Main results and takeaways

The results worth noting are the following:

1. A large number of servers is not always a good idea. In fact, both from a time and economic point of view, from a certain point on splitting the job over more servers is not convenient anymore.
2. Mean Response Time and Job Running Cost provide different information. The former is used to analyze the time it takes for all servers to complete their task, while the latter puts more emphasis on the absolute amount of computation performed, and thus grows as the number of running machines increases.
3. The optimal number of servers for the Mean Response Time is $K = 1054$ for the Jellyfish, and $K = 1023$ for the Fat-tree.
   For the Job Running Cost, the optimal value is $K = 272$ for the Jellyfish topology and $K = 31$ for the Fat-tree.