In addition to the point-to-point communication model presented in the previous section, a linear model has been also build to attempt the latency prediction by fitting the collected data with a simple linear regression approach. Contrairly to the previous model, this linear model is independent from the single point-to-point latency measurements and only takes into account the data collected from the benchmarks of the different algorithms.
The main assumption is that, for every possible processes `map-by` allocation and any possible message size, the latency can be described as a linear function of the number of processes involved in the collective operation. This is a reasonable assumption since the execution time of either a **broadcast** or a **reduce** operation is expected to increase with the number of processes involved. Driven by the measured data however, different fitting functions have been used depending on the specific mapping policy fixed in the test.
For data collected with the `map-by core` policy, a categorical variable[1] `z` has been added to the model in order to identify the active number of sockets during the test. This is done since the `core` allocation fills the sockets of the 2 nodes in a sequential order. Hence, each time a new socket starts being populated with processes, the overall latency is expected to increase with a much higher pace due to the added communications between the sockets. Formally, for any message size m, the linear model can be explained by the following equation:

$$T^{m}_{\text{latency}}(P) = \beta_0 + \beta_1 P + \sum_{i=1}^{3} (\beta_{2,i} z_i + \beta_{3,i}(P \cdot z_i)) \qquad (1)$$

where z_i is the i^{th} binary dummy variable produced from the categorical variable `z` to identify wheter or not the i^{th} socket has been involved in the test[2]. Mathematically, the effect of this variable is to change the intercept and the slope of the linear fit, depending on the number of sockets involved. Therefore, this fit produces a segmented curve in the latency plot, where the slope of the curve changes at the points where a new socket starts being populated. While few can be said about the intercept, the i^{th} slope $\beta_1 + \beta_{3,i}$ of this fit can be interpreted as the increase in latency of the specific algorithm being tested, due to the addition of one process in the i^{th} socket. Hence, contrairly to the previous model, paramenters of this model are both platform and algorithm-dependent.
A similar fit has then been done for the `map-by socket` policy, but using a categorical variable `z` that identifies the active number of nodes in each test. Therefore a similar equation to 1 has been used, but having only two dummy variables. The slope in the segmented curve of this fit therefore changes only when a new node is being used in the test. The `map-by node` instead, did not require any categorical variable since in this case all the sockets of the 2 nodes are populated in a round-robin fashion. Also in these two cases the fit parameters assume the same meaning an caracteristics as the previous one.
As described by equation 1 the fits have been done with respect to the number of processes P. This is true in the case of the **linear** and **chain** algorithm, according to the expectation that the overall execution time is in some way linearly related to the number of processes. For the **binary tree** algorithm however, the execution time is expected to be logarithmically related to the number of processes, hence the $\log_2(P)$ has instead been used as the independent variable in the fit.
The data collected for the **default** algorithms have also been fitted with a linear model using the same approach and keeping P as the independent variable. However, since as explained in section **??** the algorithm for this case in not known a priori, the objective here has just been to check to which extent the linear model well represents the data.

[1]Then converted to 3 dummy variables to perform the actual fit.
[2]i ranges from 1 to 3 since indexing is zero-based and socket 0 is always involved in any test.