In this study, two models for latency prediction of the **broadcast** and **reduce** collective operations have been built and compared: the first one based on point-to-point communication, while the second based on a linear model. The two models have been trained and tested on the data measured on the EPYC nodes for different processes allocations as allowed by the `map-by` policy of the MPI library.
As results highligted, the linear model outperformed the model based on point-to-point communications in practically all scenarios. This is not only visually confirmed by the plots but also by the values obtained for the adjusted R^2 of the two models. This is mostly due to the high capability of the linear model to fit the data. However, despite the great results, it's important to remark that the parameters obtained for the linear model are not only platform-dependent, but algorithm-dependent as well. This means that the model has to be trained for every eventual new algorithm for which latency prediction is needed, which reduces it ability to generalize beyond the measured data. Moreover, the connection between the architecture on which the collective operations run and the parameter of the model are less clear than in the case of the point-to-point model. This is also a point of concern for the linear model, as the model might not have been able to reproduce similar predictions had the benchmarks been run on a different infrastructure.
The results obtained with the model based on point-to-point trasmissions, instead, fail in predicting the measured data in many of the observed cases. Leaving aside the anomalous results obtained for the linear algorithm of the reduce operation, this inability of the model to capture the true behaviour of the measurements could be due to multiple factors. In contrast to the linear fit, this model is based on single point-to-point communications, hence the first source of error might be the false assumption that the implicit order of the communication channels based on point-to-point latencies explained by equation **??** is always respected in all situations. Indeed, the results presented above refer to messages of size 1 B for which this condition is valid, however it has been observed that for message sizes higher than 1024 B, socket-to-socket transmissions can actually take more time than node-to-node transmissions, violating the channels order previously introduced. Moreover, this model does not takes into account resources contention among the different channels, hence the possible dependency between the point-to-point transmission and the total number of processes allocated. A second source of error might reside in the way that NBFT using multiple communication channels have been modelled. In fact, it's possible that the sobstitution of groups of lower order point-to-point communication into single higher order communications done in equation **??** is an exceeding simplification of the real process. Perhaps, a different modelling of NBFT using multiple channels could return more accurate results. A third assumption that could not alway reveal to be true might be the fact that the stages of the chain and binary tree algorithm are perfectly serial. Especially in situations where the model overestimates the true latency, possible parallelization of these stages might actually have been the cause of the discrepancy. On the other hand, when the model underestimates the true latency, it's also possible that there exist a limit to the total number of processes communicating at any given time that the model does not take into account. This might also be a possible explanation for the non-logarithmic trends observed in the final part of the binary tree algorithm plots.
Having taken these limitations into account however, the point-to-point model has showed convincing results at least in some situations. Despite a non-optimal performance, the clear advantage of this model compared to the linear fit is that its parameters are platform-dependednt but algorithm-independent, therefore the model offers the possibility of being extended for the prediction of latencies of new algorithms without the need of retraining. This includes the possibility of also simulating the latency of algorithms that have not being implemented yet, before running them on the actual infrastructure.