**Title**:

Leveraging redundancy and coding techniques for speeding up distributed computing and securing distributed learning

**Abstract**:

Distributed computing is extensively used nowadays to facilitate the processing of large amounts of data for a wide range of applications, including, but not limited to, machine learning, large-scale sorting, matrix computations, and database queries. Executing jobs of large magnitude requires an increased amount of resources that are often unavailable in one machine; instead, a cluster of servers is utilized. In the first part of this work, we will focus on a popular framework that facilitates large-scale computations, known as MapReduce, to demonstrate methods that significantly speed up the overall execution. In MapReduce, a job is partitioned into multiple smaller tasks executed on different servers. The servers need to exchange intermediate information to complete the computation. There is ample experimental evidence that this so-called Shuffle phase can present a bottleneck to the algorithm and slow down the entire framework. In our work, we utilize structures from design theory known as resolvable designs to apply coding theoretic ideas in setups executing one or multiple jobs in parallel; these designs are used to decide the tasks each server runs. We have extensively tested our schemes on Amazon EC2 clusters and demonstrate speedups of up to 4.69x compared to existing approaches. In the latter parts of this work, we turn our attention to the problem of erroneous behavior of servers in distributed computing and focus specifically on training models on distributed clusters. The training is a critical component of most machine learning pipelines, and it can be made to fail if some machines return corrupted computations to the central machine that coordinates the protocol, also called the parameter server (PS). We propose a method that leverages Latin squares and bipartite expander graphs to assign gradient computations to the servers redundantly. We show that our method enjoys a 36% reduction in the fraction of corrupted gradients and an average of 20% advantage in CIFAR-10 classification accuracy. In the last part of this work, we are concerned with detecting the adversarial servers to exclude them from the training process altogether. We use constructions from design theory to assign the tasks in this case as well. Our algorithm can effectively detect all adversaries in an average of 3 iterations while achieving an accuracy benefit of at least 20% compared to the state-of-the-art on the CIFAR-10 data set.