MapReduce improves performance by exploiting parallelism but benefits of parallelism depend on good load balancing.

How does MapReduce achieve load balancing?

> Map reduce achieves load balancing by splitting the input data into many different pieces, and then passing that data to "worker" machines that map it, and then other worker machines that reduce it. Essentially, the data is first split into lots of equally large pieces, and then those pieces are then handed off to a lot of worker machines. This is where the first aspect of load balancing comes in: the master process. This process is responsible for making sure that tasks are distributed evenly, and assigns tasks to machines based on their workload. Those worker machines then process that data by calling the map function on it until it is fully mapped (to local memory), and then tell the master process that they are done. There is a second aspect of load balancing here: even though the map workers don't actually call the reduce function, they may call a combine function. This somewhat reduces the data, but does not write to a final output file. The master process will then tell the reduce workers where the output of the map workers is, and they will call the reduce function on that data until it is fully reduced and the desired results are achieved. The third aspect of load balancing has to do with backup tasks, where if a task is near completion, additional machines will be assigned that task by the master, and only the output of the one that finishes first will be accepted. This allows other machines to pick up the work of machines that are struggling and may be taking a long time to process the data.