**SUMMARY:**

Jeffrey Dean and Sanjay Ghemawat.

**MapReduce: Simplified Data Processing on Large Clusters**

**DATE: 3 January 2018**

This paper presents MapReduce, a simplified and abstract programming model for processing large clusters of data in a distributed computing system. MapReduce contains two functions; a map function that processes each key/value pair and a reduce function that processes

MapReduce is a library that helps people write distributed programs that operate on large clusters of data. The library spreads the computations across a lot of machines, run all the computations, deal with failures, collect all the results and then save them away.

"If we hadn't had to deal with failures, if we had a perfectly reliable set of computers to run this on, we probably never would have implemented Map Reduce, because without having to deal with failures, the rest of the support code just isn't that complicated." Sanjay Ghemawat

2 main reasons: - deal with failure, abstract all support stuff for similar distributed batch programming

So the failures, with a higher likelihood of occurring since they were using commodity hardware to begin with, may have been the factor that spawned the original building of a particular system around the two map/reduce components.  Parallelization, and its abstraction from the end user, were likely going to be a component of ANY successful solution.

 At that point, we started looking at the various operations in our indexing system and tried to come up with a general interface that would allow us to implement each of those operations, and would also allow us to have a number of different optimizations underneath the covers of that interface that would make things robust and scalable.

Another thing here is that the scale of the problem made the robustness critical - not a surprising thing, of course.  Starting the computations over, despite their "simplicity" for normal-sized systems, was not "tenable" for their time-frame needs.  This actually does seem to coincide with Ghemawat's "failure" assertion, though it is not conveyed as forcefully.