MapReduce: Simplified Data Processing on Large Clusters

1. What is the problem that the paper wants to solve? Why is it difficult (related works)?
   - Parallel programming for distributed systems is difficult as the programmer needs to take care of partitioning data, scheduling jobs, inter-machine communication etc.
   - Programmers without distributed computing experience can't use large resources
2. What is the solution? What is the main idea?
   - Idea: provide a high-level abstraction that can be used by programmers to formulate their algorithm in a way that can automatically be distributed to a large cluster of machines
   - Allows for easy and scalable implementation of many algorithms utilizing all resources
   - Algorithm is divided in a *map* function that applies an operation on each logical record and a *reduce* function which combines data records that have the same key
3. What is the result?
   - Tested for grep and sort algorithm: scales very well, performance similar to best reported result on TeraSort benchmark
   - System is tolerant to process failures (presumably machine failures too but this is not tested)
4. What is the main novelty that enabled the solution?
   - MapReduce is an abstraction based on regular distributed programming, parallel programming enabled the solution
   - Availability of cheap hardware and gigabit ethernet allowed realization of idea in 2003
5. What are the good aspects of the paper? Did you learn something from the paper?
   - Authors explain how it has been used successfully in practice in a broad range of problems
6. What is the impact of the paper?
   - Solved a real-world problem and has been tested extensively internally at Google prior to publication, has become very popular and been applied practically many times since publication
   - Enables programmers with no experience in distributed programming to exploit large resources
   - Fault tolerance considerations have been made as machines in the real-world often fail
7. Are there weaknesses/missing parts in the paper? How can you improve it?
   - No direct comparisons or benchmarks to show the speed of MapReduce compared to regular parallel programming
   - It is not clear if there is any overhead of the abstraction (compared to parallel programming) and how big it is
   - Limitations as to which programs can not be expressed in the MapReduce framework are not shown
8. How can you extend the paper?
   - Explore limitations of MapReduce: which programs can or can't be written as a MapReduce program?
   - Reduce overhead for machine communication to achieve higher overall throughput
9. How can you apply the technique to other data/problems?
   - Create abstraction of a complicated method in a different field to enable programmers without experience in this particular domain to write programs exploting its benefits