1. ~ame two possible uses for MapReduce besides WordCount. Do not

list problems that calculate a specific attribute of the words in a

document corpus, e.g., DistributedGrep or WhitespaceCount. In your

example, what are the map and reduce functions (including inputs and

outputs)?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

Use cases of mapreduce

nalytics, data analysis, recommendation engines, fraud detection, and user behavior analysis, among others, for which MapReduce is used as a solution

2. What kind of constraints does MapReduce place on its problem

domain. Said m another way, what applications would *not* work in

MapReduce? e.g., could you use MapReduce's for Amazon's shopping

cart?

3. Besides ~apReduce, what is another possible strategy for parallelizing

computation over a large document corpus? What are the

strengths and

weakness~s of that system relative to MapReduce?

Hive pig and spark

**MapReduce Alternative 1: Pig**

  
*The folks at Apache have had a porking good time naming components of Pig. PigLatin and Pig Engine are just two of the oink-inducing monikers.*

Pig was [originally a development by Yahoo!](http://www.dbta.com/Editorial/Think-About-It/Pig-Offers-Easy-Alternative-to-MapReduce-87653.aspx), where teams needed a language that could maximize productivity and accommodate a complex procedural data flow. Pig eventually became an Apache project, and has characteristics that resemble both scripting languages (like Python and Pearl) and SQL. In fact, many of the operations look like SQL: load, sort, aggregate, group, join, etc. It just isn’t as limited as SQL. Pig allows for input from multiple databases and output into a single data set.

**MapReduce Alternative 2: Hive**

  
*From porkers to buzzers, the world of Hadoop is never lacking in creative names. But if MapReduce is stinging, Hive can sweeten it up like honey.*

Hive also looks a lot like SQL at first glance. It accepts SQL-like statements and uses those statements to output Java MapReduce code. It requires little in the way of actual programming, so it’s a useful tool for teams that don’t have high-level Java skills or have fewer programmers with which to produce code. Initially developed by the folks at Facebook, [Hive is now an Apache project](https://en.wikipedia.org/wiki/Apache_Hive).

**MapReduce Alternative 3: Spark**



Perhaps the most momentum has been [achieved with Spark](https://vision.cloudera.com/mapreduce-spark/), which has widely been hailed as the end of MapReduce. Spark was born in the AMPLab at the University of California in Berkley.

Unlike Pig and Hive, which are merely programming interfaces for the execution framework, Spark replaces the execution framework of MapReduce entirely. One of the most celebrated qualities of Spark is that it’s super smart about memory and resource usage. It’s a solid general-purpose engine that allows you to run more Hadoop workloads and to run them faster. Spark also packs an impressive list of features, including stream processing, data transfer, fast fault recovery, optimized scheduling, and a lot more.

While each alternative to hand coding Java comes with pros and cons of its own, all are easier to manage than MapReduce, unless you are the proud owner of a team of Java experts. Of course, some organizations decide to leverage 3rd-party tools that help users to avoid hand coding all together. [Syncsort’s DMX-h](http://www.syncsort.com/en/Products/BigData/DMXh?utm_source=Blog-Post&utm_medium=Blog-In-Text&utm_title=3%20Alternatives%20to%20MapReduce%20Programming" \t "_blank)is one popular “no-coding” choice to simplify the entire data pipeline, whether you are using the MapReduce or Spark execution framework – because it runs on both.

Using its simple GUI, you can access data from across your enterprise (including [hard to manage mainframe sources](http://www.syncsort.com/en/Solutions/Hadoop-Solutions/Mainframe-Offload-to-Hadoop?utm_source=Blog-Post&utm_medium=Blog-In-Text&utm_title=3%20Alternatives%20to%20MapReduce%20Programming)), bring it into Hadoop, and then leverage it – instead of Pig or Hive – for processing the data on the cluster. Organizations leveraging DMX-h say they are up and running faster – and can make changes quicker – compared to hand coding.

For more talk about the future of Big Data, including more on Spark, read our eBook, [**Bringing Big Data to Life: What the Experts Say**](http://www.syncsort.com/en/Resource-Center/BigData/eBooks/Bringing-Big-Data-to-Life-What-the-Experts-Say?utm_source=Blog-Post&utm_medium=Blog-In-Text).

[APACHE SPARK](http://blog.syncsort.com/tag/apache-spark/)[HIVE](http://blog.syncsort.com/tag/hive/)[MAPREDUCE](http://blog.syncsort.com/tag/mapreduce/)[PIG](http://blog.syncsort.com/tag/pig/)

4. Whe?" des1~g a distributed system from the ground up, what

cons1der~t1ons would you make to improve the reliability of the

networking component?

5. How does MapReduce improve the reliability of distributed systems?