CS 580K Advanced Topics in Cloud Computing: Written Assignment 3

Ketan Deshpande
B00816854
kdeshpa5@binghamton.edu

1. What is the relationship between MapReduce and Hadoop?

Hadoop uses MapReduce model to process data which is stored on HDFS. Hadoop has a Master-Slave architecture for data storage and distributed data processing using MapReduce and HDFS methods. The master node receives job submission from clients and allow them to conduct parallel processing of data using Hadoop MapReduce. Tasktracker runs the actual code of job on the data blocks of input file within the master node using MapReduce technique. If the submitted job is failed during the map operation, it automatically selects the healthy node and re-run the map function. Therefore, Hadoop is also known as self-healing system.

2. One of the key ideas of Hadoop is to move computation closer to data. How is this idea materialized on Hadoop?

The concept of moving computation closer to data is materialized in Hadoop by using Hadoop Distributed File System i.e. HDFS. It is responsible for storing large data on the cluster. It works with small number of large data files. Basically, the data files are split into blocks and distributed across the nodes in the cluster. When a MapReduce job is submitted, it is divided into map jobs and reduce jobs. A Map job is assigned to a datanode according to the availability of the data, i.e. it assigns the task to a datanode which is closer to or stores the data on its local disk. Data locality refers the process of placing computation near to data, which helps in high throughput and faster execution of data.

3. By design, intermediate data generated by Map functions are stored in storage (disk drives instead of memory). What is the purpose of this design?

Hadoop distributes data across multiple nodes within a cluster of commodity servers. The distributed storage approach is crucial to big data since it allows vast datasets to be stored across innumerable hard drives, thus saving organizations the cost of maintaining a single expensive hardware. In other words, Hadoop has its own file storage system called Hadoop Distributed File System (HDFS). HDFS takes in data, breaks it into segments, and stores it in the various nodes in the system. HDFS supports MapReduce effectively along with the other tools in the Hadoop ecosystem to ensure that they can all be integrated easily. MapReduce is more effective when handling large amounts of data. Spark requires a lot of memory; if it runs with resource-demanding services or if the memory proves to be insufficient for data, there could be major performance issues. On the other hand, MapReduce kills its processes once a task is done so it easily runs in parallel with other services. When it comes to iterative computations on dedicated clusters, Spark has the upper hand but when it comes to ETL-like jobs, MapReduce is the superior alternative.

4. What is the key idea behind serverless computing? How do you think it will benefit cloud end users?

Serverless computing makes app development & operations dramatically faster, cheaper, easier. It drives infrastructure cost savings due to its more fine-grained pricing model which is pay as execution. Another key idea behind serverless computing is that its use of other providers' services so you can easily integrate services from multiple providers into your computing service applications. By doing this it allows vendor lock-in issue which is you can leverage services from multiple service providers. All these key ideas behind serverless computing will benefit cloud end users.

5. What are the big drivers for edge computing? Will edge computing and cloud computing co-exist in the future or edge computing replace cloud? Please state your opinion.

One big driver for edge computing is latency issue. The latency is the time taken to deliver the output requested by the user from very long distance. For e.g., the time taken to show webpage from the moment you clicked on the link of that webpage. Sometimes latency is tolerable like showing a webpage. But sometimes it is not tolerable, and this

impacts user negatively. Unfortunately cloud computing does not guarantee low latency which is important in certain type of applications. The latency problem becomes more severe with the exponential growth of IoT devices because the cloud network traffic increases as more IoT devices try to connect with cloud to send/receive data. Edge computing and cloud computing can co-exist in the future; indeed, we need an in-between solution between IoT devices and cloud computing. Because as the definition of edge suggests that edge computing is done at or near the source of the data, instead of relying on the cloud. But even if we implement this kind of architecture, we need a platform through which the data can transferred between these devices and the servers. And this can be done using cloud computing. For e.g. in the Amazon echo devices, the voice request is accepted on the device, sent to the cloud and it processed there. A web service API is also called if needed while processing. However, if more computing is done locally, then it will reduce the latency as per the definition of edge computing expects.

6. Why is the micro-kernel based “container” solution more secure than docker containers?

Micro-kernel architecture excludes system services from kernel space and includes only system call-based architecture results in much smaller code base and is a promising way to make the system more stable and secure. The micro-kernel-based containers contain in addition to the application code and dependencies we also package up the required kernel components into the unikernel container image. The unikernel container can run on top of micro-kernel providing high security properties. In the micro-kernel all the system services reside in the user space in the form of normal processes such as memory management, file servers and network stack. By excluding these systems services, it promises more stable and secure system.