**CSE 291 Homework #2**

**Spring 2016 (Kesden)**

**Student Name: Kaiwen Sun**

**Student ID: A53091621**

**Data-Intensive Scalable Computing (DISC)**

1. HDFS is implemented as a user-level file system vs an in-kernel file-system. (a) What is the advantage of this in the context of Hadoop?

First, it is safer to run in user mode than in kernel mode. If the user-mode HDFS crashes, then other unrelated applications will not be affected.

Second, it is more portable for HDFS to use the local filesystem provided by the operating system. The filesystem may be FAT32 in one host, and EXT4 on the other. HDFS doesn’t have to know about the implementation of local filesystem.

2. The output of a Mapper is written into the local filesystem instead of the global filesystem. Why? Your answer should explain both why writing into the global file system would be undesirable as well as why it would be of minimal benefit.

The result of mapping usually is not the final result. Usually the result of mapping will be processed locally before being merged with data from other hosts. So writing it to the global filesystem is not necessary. However, writing it to the global filesystem can cause heavy communication (called “shuffling”) which takes a lot of unnecessary time.

3. Why does Hadoop sort records en route to a Reducer? How would it affect things if these records were processed by the Reducer in the order in which they were received from the various Mappers?

(1) Human prefer to read data in sorted order.

(2) Sometimes the records are to be searched using techniques like binary sort, which benefits from sorted data.

(3) During the reduce step, a very common operation is to reduce-by-key. If data are sorted by key, then it will be more efficient because data with same key are close to each other, and are very likely to be on the same host.

As an interesting aside, since the keys are sorted en route to a reducer, it is possible to write a massively distributed sort via MapReduce. And, this is pretty cool.

4. How is the failure of a Mapper or Reduce managed?

If a Worker fails, it is marked as bad, and its work is rescheduled to another Worker. Furthermore, any Reduce that was scheduled to get results from the old Worker is told to begin getting them from the new Worker, instead.

If a mapper dies, since the related data result is stored on its local filesystem, the result is lost. So the new worker should recompute it from scratch.

In contrast, if a reducer dies, since the related data result is stored on the global filesystem, it is still available.

5. In a typical Map-Reduce graph algorithm, what data structure is used to represent the graph? Why?

Adjacency list.

Because using the information of very few nodes’ adjacency list, one worker can

– Perform local computations in mapper

– Pass along partial results via outlinks, keyed by destination node

– Perform aggregation in reducer on inlinks to a node

And they are parallel!

6. In a typical Map-Reduce graph algorithm, how many Map-Reduce phases are typically necessary before the graph can be traversed? Why?

Before the graph can be traversed, there is one Map-Reduce phase to build the graph. The Map parses the input to the adjacent list of every vertex. The Reduce is an identical function which does nothing.

**Processor Allocation and Migration**

7. If processor allocation is optimal, is it possible that migration will subsequently improve system performance? If not, why not? If so, how?

It is possible to improve system performance, because performance is also affected by communication. For example, sometimes it is better to have processes that require a great deal of IPC on the same host, so that the latency associated with the IPC is minimized. For example, we have 2 hosts and 4 processes: A, A’, B, B’.Each process takes 50% CPU poser. A and A’ have many IPCs. B and B’ have may IPCs. Allocating A,B on one host, and A’,B’ on the other, is an optimal allocation in terms of processor. However, we may want to swap A and A’.

8. Why are periodic broadcast advertisements often considered to be a poor way of communicating information about resource availability? What is the risk?

Broadcast advertisements may cause “thundering herds”. The previously idle processor suddenly is heavily loaded. Then, another processor, perhaps one that recently offloaded work becomes idle, and broadcasts. Then all workload flush back…… Tasks are being migrated again and again, without much progress.

9. Please explain two commonly used alternatives to the advertisements mentioned above and the relative costs and benefits.

1. Receiver initiated “ask around”. As soon as an idle processor finds work (within a certain number of asking), it stops asking. This approach avoids thundering herds, but it leads to heavy communications overhead when the processors are mostly idle.
2. Sender initiated “ask around”. As soon as a host realizes its workload is over a threshold, it asks around for help. If it finds an idle processor (within a certain number of asking), it stop asking. This approach avoids thundering herds, but it leads to heavy communications overhead when the processors are mostly busy.

**Distributed File Systems**

10. In class we observed that AFS and NFS manage consistency differently. AFS issues callbacks upon updates. NFS validates the client cache periodically.

(a) Do either of these mechanisms eliminate the window of vulnerability? If so, how? If not, is possible to eliminate the window of vulnerability? Why or why not?   
  
No, they can’t eliminate the window of vulnerability. For NFS, after the file is updated at server side and before a client validates its cache, the local file is stale. For AFS, after a file is updated at server side and before the server callbacks to registered clients (due to processing and network latency), the local files are stale.  
  
It is impossible to completely eliminate the window. Between the server side update and the client side update, there is always at least an unavoidable network communication latency.

(b) Which mechanism will result in less network traffic in the event that many dozens of clients have the same file open for high-frequency random-access reads?  
  
AFS results in less network traffic. Since there is much more reads than writes, a read in AFS will not cause callback communication. However, NFS’s communication doesn’t depend on read/write operations, so there will be still a considerable amount of traffic.

**Security**

11. Consider *Onion Routing* and the case of a compromised router. In this worst case, will it know the source of the message, the destination of the message, both? Why?  
  
Suppose the IP address of source and destination is not in the plaintext of the message.  
Then the exit node, if compromised, can see the destination, otherwise it would not be able to deliver the message to the destination.  
The entry node, if compromised, can see the source of the message, because it directly communicates with the source.  
No router can know both the source and destination, unless the router is both entry and exit node. Usually there will be more than one onion router.

12. Consider *Onion Routing,* why is the path chosen in advance by an agent of the client, rather than the network hop-by-hop?  
  
The Tor user wants to avoid any (except the exit node) from knowing the destination of the communication, so it has to determine the path in advance. Otherwise the intermediate router will not know where to deliver the message. Also, the initiate sender needs to use the key of every router to encrypt the message layer by layer, so it needs to decide the path in advance.

13. Kerberos enables a client to communicate credentials to a server. What guarantees that the server will be able to trust these credentials? (Covering this on Monday)  
  
If the server can correctly decrypt the message sent by client, it means the client has been granted by the TGS.  
If the client is granted by the TGS, it means the client has got the correct TGT from Authentication Service.  
If the client has got the correct TGT from AS, it means the client can correctly decrypt the message encrypted by AS using the symmetric key of whom the client claims to be.  
So AS can believe the client is whom it claims to be.  
The app server trusts the AS, so it trusts the client.

14. Kerberos uses *symmetric/secret key* cryptogrophy, rather than *asymmetric/public key* cryptopgraphy. Why? (Covering this on Monday)  
  
Asymmetric cryptosystem has some drawbacks. First, asymmetric usually runs more slowly than symmetric cryptosystem. Secondly, many asymmetric usually can only encrypt/decrypt a particular form of data (such as RSA, Elliptic curve, etc.).  
Also, deploying public keys doesn’t bring much more benefits than deploying symmetric keys.  
  
~~What’s more, suppose the system is using asymmetric cryptosystem, then for example, the GTS should has a fixed private key. Then both client A and client B request GTS’s public key from AS. Note that the public keys held by A and B are identical, because they are both corresponding to GTS’s fixed private key. Now A can forge B’s message, and vice versa.~~