The key is combination of a row key, a column key and a timestamp. We have to provide three parameters to get the value: row key, column key, and timestamp.

(2) “Clients can exploit this property by selecting their row keys so that they get good locality for their data accesses.” How would clients select keys to get good locality? What possible advantages could a client receive by having the locality?

Good locality is obtained by placing access–related data together. It means that in order to get good locality, the data that is often accessed together should be placed near each other.

Now in Bigtable, since the data is sorted by row key, clients can select the row keys for data in the way so that they can be grouped together in contiguous rows. For example, if there are two row keys: apple and banana. In order to group this two key together, I can revise these two keys as fruit.apple, fruit.banana.

In this way, these two keys can be grouped together, even though they may not exactly next to each other.

What possible advantages could a client receive by having the locality?

I think the possible advantages include getting a lot of useful data quickly by only one read request, reducing the number of read requests, and experiencing high-speed of read operation because the read operation involves a few tablet servers.

(3) “Bigtable uses the distributed Google File System (GFS) to store log and data files.” To ensure high data reliability, does BigTable need to maintain multiple replicas for each of its data items?

Answer: No. the authors do not consider the possibility of multiple copies of the same data. the data items of Bigtable are either in commit log files or in SSTables. Both kinds of files are stored in GFS.

And GFS already provides high data reliability by having multiple replicas across different storage devices. In other words, Bigtable provides high data reliability by using GFS as its basic infrastructure to store data items.

(4) “The Google SSTable file format is used internally to store Bigtable data. An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings.” What does it mean by “immutable”? Why is this feature required?

Being immutable means that you are allowed to read but not allowed to modify any data in an SSTable.

This feature is required, because it has many benefits:

One benefit is that many parts of the Bigtable system can simplified.
For example, tablet server has two types of caching. Scan cache that caches the KV pairs and block cache that caches SSTable blocks. Due to the immutability feature, there is no inconsistency issue and no need to write them back to disk. This makes the cache maintaining very easy.

Another example, we do not need any synchronization of accesses to the file system when reading from SSTables. As a result, the concurrency control over rows can be implemented very efficiently.

Another benefit is that the immutability of SSTables enables us to split tablet quickly. We do not need to generate a new set of SSTables for each child tablet, we just let the child tablets share the SSTables of the parent tablet.

In addition, immutability can restore data to the current value from old version.

(5) “A block index (stored at the end of the SSTable) is used to locate blocks; the index is loaded into memory when the SSTable is opened. A lookup can be performed with a single disk seek: … ” Describe how a KV item is retrieved from an SSTable and why only one disk access is required for a lookup? Also explain why KV items are sorted according to the keys. [Hint: assume each block in an SSTable is 4KB, the disk access unit.]

Assume that the KV item is in an SSTable. System searches key starting from the memtable, then goes to the SSTables from low level to high level until finds it.

Here are the steps:
First, go to the in-memory index.

Then find the appropriate block using binary search.

After this, check the corresponding Bloom Filter to see if the Key is there or not?
Suppose the key is there, then go to the disk to read the block and get the value.
Otherwise, move to the next SSTable and repeat the above steps until find the value.

Then why only one disk access is required to a lookup? This is quite straightforward. The system only goes to the disk to read the block when it finds the block in memory. Otherwise, it just claims that the KV item is not existent. So there is only one disk access.

But it is possible that there are two or more disk accesses. But this possibility is very low. This happens when the Bloom filter gives the false positive.

Why KV items are sorted according to the keys? This is because sorted KV items have two major merits.

First, they can support range search. (this is obvious)

Second: they can reduce the size of block index at the end of SSTables.

(6) “Of these updates, the recently committed ones are stored in memory in a sorted buffer called a memtable; the older updates are stored in a sequence of SSTables.”. Why do older updates exist and possibly exist in multiple SSTables?

So why do older updates exist?
One reason is that the size of memtable is limited. the extra data has to be dumped to disk.
Another reason is that SSTables are immutable, you can not do any modification in them.
I think the third reason is that the value is allowed to have multiple versions identified by timestamp.

Why do older updates possibly exist in multiple SSTables? SSTables from different levels can have overlap ranges.

(7) “A merging compaction that rewrites all SSTables into exactly one SSTable is called a major compaction.” What is minor compaction, and what is major compaction? Why is major compaction needed? How is a KV item deleted?

Minor compaction means that when the memtable is full, the memtable is converted into an SSTable and written to GFS.

Major compaction means to rewrites many SSTables into exactly one SSTable. The major compaction produces an SSTable that contains no deletion information or deleted data.

Why is major compaction is needed?

1. major compactions allow bigtable to reclaim resources used by deleted data, and also allow it to ensure that deleted data disappears from the system in a timely fashion, which is important for services that store sensitive data.

2. major compactions can bound the number of SSTables.

3. remove overlapped ranges to support ranch search.

How to delete a KV item?

First, write a special deletion record to commit log, insert the record to the memtable. Through minor compaction, the special deletion record is transferred into an SSTable. Finally, when major compaction occurs, the data needed to be deleted and the special deletion record are deleted totally. This is how to delete a KV item.