1) Not sure how good my answers are so I added them in review mode. If someone agrees with them pls accept the suggested change.

a)

i) A sloppy Quorum is a type of quorum that doesn’t enforce strong consistency. It allows N read and write operations to be performed rather than R + W > N. This is used in dynamo’s consistency model.

[https://ebrary.net/64728/computer\_science/sloppy\_quorums\_hinted\_handoff](https://ebrary.net/64728/computer_scieznce/sloppy_quorums_hinted_handoff)

A sloppy Quorum is a type of quorum that doesn’t enforce strong consistency. Like quorums, it sends read/write requests to N nodes, and requires R reads and W write requests to be performed for the request to succeed. However, it can write to node’s outside the N designated nodes if they are unavailable.

ii) Immutability is the concept of having data structures that can’t be modified; only created or deleted. This is used in many scalable systems (Such as SSTables in BigTable) so that consistency issues don’t have to be dealt with.

iii) What is this? Not Examinable?

Energy proportionality is where energy consumption grows proportional to the growth of load. Essentially, it ensures that we are optimally making use of our hardware available by providing energy to run the hardware only when it is doing useful work. This is required in scalable systems as we want to keep our costs low and only provide energy when stuff is happening.

iv) A virtual node is similar in concept to a virtual machine, but it acts as a node in a distributed system. It is much more versatile than a physical node when it comes to load balancing and it is used in Dynamo for this purpose.

b)

i) BigTable is a semi-relational multi-version column store. As mentioned in the question, it stores data in a map with the following schema:  
  


The row is essentially the “unique” identifier for the object (usually, the reversed URL in the webtable case).

Column can be specified by the user. The initial paper demonstrates anchors on a web pages as column values as well as the HTML content of a webpage itself.

The time field allows for the multi-version aspect and it is at the developer’s discretion as to the number of versions that are retained.

In practice, the data model of BigTable is broken up into:  
 (index: int64, column: string, time: int64) and (row: string, index: int64)  
with the former stored before the latter in the SSTable, for better indexing.

ii)

Data locality is important because in a consumer focused world, profits are driven by the number of users and poor performing systems results in less users. Achieving good locality of data that is frequently accessed together gives better performance and will lead to more users and profits.

I think the focus on this question is the data-intensive bit. So maybe we can talk about how compression is better when there is good data locality- aka rows with alot of common words in them are compressed and stored together. This saves more storage space, an issue in data-intensive contexts.

I think data-locality is also about doing computations on nodes where the data already exists. This way we prevent network congestion and increase overall throughput.

iii)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Row | Columns | | | | Time |  |
| uk.ac.imperial.www | “contents” | “/study” | “/study/pg” | “/study/ug” | ... |  |
| uk.ac.ic.doc.www | “contents” | | | | ... |  |

The uk.ac domains would be kept in somewhat close locality. The developer is able to group “/study”, “/study/pg” and “/study/ug” into column families in order to ensure that the data stored under them is kept close in a locality group.

iv)

You could maintain a list of mappings of aliases in order to further keep sites on the same domains closer together in terms of locality. For example, in this case, we would map ic to imperial so both domains would fall under uk.ac.imperial and be closer together.

2)

a)

i) A distributed file system is a system for storing data across multiple machines, that is able to process requests for the data and serve it from the machine it is stored from in a distributed fashion. Two examples of this are GFS and HDFS, which are by google and hadoop respectively.

ii) A nearline system is one that incorporates aspects of both a batch and online system. Thus, it is required to support workloads that process large groups of data but also to be able to process individual transactions and behave at a low latency.

iii) A coordinator is the part of a system responsible for maintaining concurrency between processes of a system. It may also have other tasks and is usually a centralized component that’s purpose is to keep the system running. An example of this are coordinator nodes in Dynamo, that maintain correctness of read/write requests across all nodes in the quorum.

iv) A session is when a client connects to an application and exchanges multiple requests before disconnecting. An example of this is in BigTable, when tablet servers open a Chubby session with a Chubby service to acquire an exclusive lock over a tablet. Disconnecting from the session loses the lock.

b)

i)

RDDS: input, cleandata, pointdata (All of type RDD[Number], not sure if Float or Int) pointdata is not a number anymore I guess

Transformations: filter (Number -> Bool), Map (Number -> Number)

(Note, the reduce is not a transformation, it is an action, and the reading of the input file is neither)

ii)

Input ---filter(n>10)---> cleandata ---map(n\*n)--->pointdata---reduce---> result

iii)

Failure at line 3 would cause a new, healthy worker to identify the persisted data in memory from input that was involved in the failed computation. It would then proceed to recompute the filter and map operations, (learning them from inspecting the lineage graphs of other nodes in the cluster) in order to allow the reduce to be performed.

iv)

A worst case program would have a long lineage chain, with no calls to persist (so data is not checkpointed in memory) and it would have a lot of wide data dependencies with pipeline-breaking operations such as joins. This would mean a failure would most likely need to start the computation again from scratch.