Examples of exam question types

Databases for Big Data

- NoSQL data stores and techniques
  - Explain the main reasons for why NoSQL data stores appeared.
  - List and describe the main characteristics of NoSQL data stores.
  - Explain the difference between ACID and BASE properties.
  - Discuss the trade-off between consistency and availability in a distribute data store setting.
  - Discuss different consistency models and why they are needed.
  - Explain how consistency between replicas is achieved in a distributed data store.
  - Explain the CAP theorem.
  - Explain the differences between vertical and horizontal scalability.
  - Explain how consistent hashing works and what are the problems it addresses.
  - Explain how vector clocks work and what are the problems they address.
  - List and describe dimensions that can be used to classify NoSQL data stores.
  - List and describe the main characteristics and applications of NoSQL data stores according to their data models.
- HDFS
  - Explain what HDFS is and for what types of applications it is (not) good for.
  - Explain the organization of HDFS.
  - Explain the process of reading and writing to HDFS.
  - Explain how high availability is achieved in HDFS.
- Dynamo
  - Explain the data model and list main applications of Dynamo.
  - Explain the Dynamo design considerations and what are the advantages of Dynamo in comparison to RDBMSs.
  - Explain how basic NoSQL techniques are applied in Dynamo.
  - Explain versioning and semantic reconciliation in Dynamo.
- HBase
  - Explain the difference between column-oriented and row-oriented storage.
  - Explain the data model of HBase.
  - Give example applications of HBase.
- Hive and Shark/SparkSQL
  - Explain the problem that Hive and Shark/SparkSQL address.
  - Explain the data model of Hive and Shark/SparkSQL.
  - Discuss the trade-off between schema-on-read and schema-on-write approaches.
  - Explain the difference between OLAP and OLTP.
  - Explain the main differences between Hive and Shark and what are the advantages they lead to.
  - Explain how fault tolerance is achieved in Shark/SparkSQL.

Parallel Computing

- PAR-Q1: Define the following technical terms:
  (Be thorough and general. An example is not a definition.)

    - Cluster (in high-performance resp. big-data computing)
    - Parallel work (of a parallel algorithm)
    - Parallel speed-up
    - Communication latency (for sending a message from node Pi to node Pj)
    - Temporal data locality
    - Dynamic task scheduling
- PAR-Q2: Explain the following parallel algorithmic paradigm: Parallel Divide-and-Conquer.
- PAR-Q3: Discuss the performance effects of using large vs. small packet sizes in streaming.
- PAR-Q4: Why should servers (cluster nodes) in datacenters that are running I/O-intensive tasks (such as file/database accesses) get (many) more tasks to run than they have cores?
- PAR-Q5: In skeleton programming, which skeleton will you need to use for computing the maximum element in a large array? Sketch the resulting pseudocode (explain your code).
- PAR-Q6: Describe the advantages/strengths and the drawbacks/limitations of high-level parallel programming using algorithmic skeletons.
- PAR-Q7: Derive Amdahl's Law and give its interpretation.
- PAR-Q8: What is the difference between relative and absolute parallel speed-up? Which of these is expected to be higher?
- PAR-Q9: The PRAM (Parallel Random Access Machine) computation model has the simplest-possible parallel cost model. Which aspects of a real-world parallel computer does it represent, and which aspects does it abstract from?
- PAR-Q10: Which property of streaming computations makes it possible to overlap computation with data transfer?

MapReduce

- MR-Q1: A MapReduce computation should process 12.8 TB of data in a distributed file with block (shard) size 64MB. How many mapper tasks will be created, by default? (Hint: 1 TB (Terabyte) = 10^12 byte)
- MR-Q2: Discuss the design decision to offer just one MapReduce construct that covers both mapping, shuffle+sort and reducing. Wouldn't it be easier to provide one separate construct for each phase instead? What would be the performance implications of such a design operating on distributed files?
- MR-Q3: Reformulate the wordcount example program to use no Combiner.
- MR-Q4: Consider the local reduction performed by a Combiner: Why should the user-defined Reduce function be associative and commutative? Give examples for reduce functions that are associative and commutative, and such that are not.
- MR-Q5: Extend the wordcount program to discard words shorter than 4 characters.
- MR-Q6: Write a wordcount program to only count all words of odd and of even length. (There are several possibilities.)
- MR-Q7: Show how to calculate a database join with MapReduce.
- MR-Q8: Sometimes, workers might be temporarily slowed down (e.g. repeated disk read errors) without being broken. Such workers could delay the completion of an entire MapReduce computation considerably. How could the master speed up the overall MapReduce processing if it observes that some worker is late?
- Spark-Q1: Why can MapReduce emulate any distributed computation?

- Spark-Q2: For a Spark program consisting of 2 subsequent Map computations, show how Spark execution differs from Hadoop/Mapreduce execution.
- Spark-Q3: Given is a text file containing integer numbers. Write a Spark program that adds them up.
- Spark-Q4: Write a wordcount program for Spark. (Solution proposal: see last slide in lecture 8.)
- Spark-Q5: Modify the wordcount program by only considering words with at least 4 characters.

Cluster Resource Management

- YARN-Q1: Why is it reasonable that Application Masters can request and return resources dynamically from/to the Resource Manager (within the maximum lease initially granted to their job by the RM), instead of requesting their maximum lease on all nodes immediately and keeping it throughout the job's lifetime? Contrast this mechanism to the resource allocation performed by batch queuing systems for clusters.
- YARN-Q2: Explain why the Node Manager's tasks are better performed in a daemon process controlled by the RM and not under the control of the framework-specific application.

Machine Learning for Big Data

- Implement in MapReduce or Spark a machine learning algorithm, e.g. logistic regression, k-means, EM algorithm, support vector machines, neural nets, etc. (The pseudo-code of the algorithm will be provided in the exam.)