CSS451: Cloud Computing

Midterm Mock Exam

curated by The Peanuts

Name ..... ID ..... Section ..... Seat No .....

Conditions: Closed Book

Directions:

1. This exam contains 16 pages (including this one). If yours has fewer, congratulations — you have discovered the distributed truncation problem.
2. Write your name and ID clearly at the top.
3. Show your reasoning for short and long answer questions. This is not a black box, your thought process matters.
4. Answers must be written in English. Pseudocode is acceptable where explicitly permitted. YAML, Dockerfile, and Scala will not be graded.
5. This is a **closed book** exam. You may not use phones, laptops, or your neighbor's cluster of neurons.
6. For Part V (Case Study), choose **exactly one** scenario. Attempting more than one will result in zero points for the entire part. Indecision is not fault-tolerant.
7. If a question is ambiguous, state your assumptions clearly and proceed. Partial credit is awarded based on reasoning quality.

For solution, ***click here****.*

Part I: True / False (10 Points, 1 pt each)

*Write **TRUE** or **FALSE** in the blank provided. If a statement is partially correct, it is **FALSE**.*

1.1 _____ Spark achieves fault tolerance primarily through data replication across multiple nodes, similar to HDFS's default replication factor of 3.

1.2 _____ In an RDD lineage graph, transformations are executed lazily — they are not computed until an action such as `count()` or `collect()` is called.

1.3 _____ A `join` operation on two RDDs of types `RDD[(K, V1)]` and `RDD[(K, V2)]` produces an output of type `RDD[(K, (V1, V2))]`.

1.4 _____ In Spark Streaming, computation is performed as a continuous event-driven data flow, not as small batch jobs.

1.5 _____ Apache Mesos uses a two-level scheduling model in which the Mesos master offers resources to framework schedulers, and the frameworks decide how to use those resources.

1.6 _____ Omega uses optimistic concurrency control, meaning multiple schedulers can read the shared cluster state simultaneously and propose resource allocations independently, with conflicts resolved at commit time.

1.7 _____ In Kubernetes, a Pod is the smallest deployable unit, and a single Pod can contain more than one container.

1.8 _____ The `etcd` component in Kubernetes is responsible for scheduling Pods onto available worker nodes.

1.9 _____ In GFS (Google File System), the master server is directly involved in transferring chunk data from a chunkserver to the client during a read operation.

1.10 _____ Para-virtualization requires modifications to the guest operating system, whereas full virtualization can run an unmodified guest OS.

Part II: Fill in the Blanks (10 Points, 1 pt each)

Fill in each blank with the most appropriate word or phrase. Choose from the word bank below where indicated, or write your own answer.

Word Bank (not all words will be used):

cogroup *lineage* *DStream* *etcd* *kubelet* *kube-proxy* *straggler* *combiner* *chunkserver* *master* *optimistic* *pessimistic* *immutable* *in-memory* *partitioned* *two-level* *shared-state*

2.1 In Spark, an RDD is a _____, _____ collection of objects that can be processed in parallel.

2.2 When a partition of an RDD is lost due to a node failure, Spark recovers it by re-executing the chain of transformations recorded in the RDD's _____ graph.

2.3 The _____ operation in Spark groups the values of two key-value RDDs by key, returning an RDD of type `(K, (Iterable[V1], Iterable[V2]))`.

2.4 In Spark Streaming, a discretized stream is represented as a _____ (abbreviation), which is a sequence of RDDs generated at regular time intervals.

2.5 In MapReduce, a _____ is a mini-reducer that runs on the mapper output locally, reducing the volume of data transferred across the network before the shuffle phase.

2.6 Mesos employs a _____ scheduling architecture, where the master exposes available resources to framework schedulers via *resource offers*.

2.7 Omega differs from Mesos primarily in its use of _____ concurrency control, allowing all schedulers to operate on a full copy of cluster state simultaneously.

2.8 In Kubernetes, the _____ is a distributed key-value store that persists the entire cluster configuration and state.

2.9 The _____ agent runs on each worker node in Kubernetes and is responsible for ensuring that the containers described in Pod specifications are running and healthy.

2.10 In GFS, the component that holds actual file data blocks and serves data directly to clients (after initial metadata lookup) is called the _____.

Part III: Multiple Choice (10 Points, 1 pt each)

*Choose the **single best** answer for each question. Circle the letter of your choice.*

3.1. Which of the following best explains why Spark is significantly faster than Hadoop MapReduce for iterative machine learning algorithms?

a) Spark uses a more efficient sorting algorithm during the shuffle phase.

b) Spark stores intermediate RDDs **in memory**, avoiding repeated disk I/O between iterations.

c) Spark parallelizes the reduce phase using a binary tree reduction strategy.

d) Spark compresses intermediate data using LZ4 before writing to HDFS.

3.2. Given `rdd1: RDD[(K, V1)]` and `rdd2: RDD[(K, V2)]`, what is the output type of `rdd1.join(rdd2)`?

a) `RDD[(K, V1, V2)]`

b) `RDD[(K, (Iterable[V1], Iterable[V2]))]`

c) `RDD[(K, (V1, V2))]`

d) `RDD[((K, V1), (K, V2))]`

3.3. In Spark Streaming, the fundamental processing model is best described as:

a) A continuous event-driven pipeline where each record is processed as it arrives.

b) A series of small, deterministic batch jobs over micro-batch intervals (DStreams).

c) A push-based model where data is sent directly to Spark executors by producers.

d) A persistent stateful loop that blocks until a defined window is full.

3.4. Which statement correctly distinguishes Omega from Apache Mesos?

a) Mesos uses shared-state scheduling; Omega uses two-level scheduling.

b) Omega uses optimistic concurrency control on a shared cluster state; Mesos uses pessimistic control by locking resources via offers.

c) Omega is a centralized monolithic scheduler; Mesos is fully decentralized.

d) Mesos allows all frameworks to simultaneously modify the cluster state; Omega serializes all scheduling decisions through a single master.

3.5. In Apache Mesos, the framework can _____ a resource offer from the Mesos master. Which word correctly completes the sentence, and what is the implication?

a) *Accept* — the framework must use all offered resources immediately.

b) *Reject* — the framework may decline resources that do not suit its constraints, enabling other frameworks to receive them.

c) *Preempt* — the framework can forcibly reclaim resources from other frameworks.

d) *Partition* — the framework can split offers among multiple tasks.

3.6. Which Kubernetes component acts as a load balancer for incoming traffic, routing requests to the correct Pod?

a) kubelet

b) etcd

c) Scheduler

d) kube-proxy

3.7. In HDFS, which node is responsible for managing file namespace, access control, and metadata, but does *not* store actual file data?

a) DataNode

b) NameNode

c) SecondaryNameNode

d) JobTracker

3.8. Which of the following correctly describes a DAG (Directed Acyclic Graph) in the context of Apache Spark?

a) A graph that represents the physical storage layout of data blocks across cluster nodes.

b) A logical execution plan representing the sequence of transformations from input to output, used by the Spark scheduler to optimize and stage tasks.

c) A replication topology used by the master to distribute tasks to executors.

d) A graph that models network communication between driver and worker nodes.

3.9. In the MapReduce programming model, data locality refers to:

a) Storing all output data on the same node as the master for fast retrieval.

b) Moving computation to the node where data resides, rather than moving data to computation.

c) Replicating data to all mapper nodes before processing begins.

d) Ensuring all reducers are co-located on the same rack to minimize network traffic.

3.10. A framework running on Mesos holds onto acquired resources without using them, preventing other frameworks from accessing those resources. This behavior is best described as:

a) Starvation

b) Thrashing

c) Resource Hoarding

d) Contention

Part IV: Short and Long Answer (20 Points)

Question 4.1 (4 Points)

Consider the following two RDDs:

```
orders: RDD[(CustomerID, OrderAmount)]
customers: RDD[(CustomerID, CustomerName)]
```

With the following data:

orders	customers
(101, 250.0)	(101, "Alice")
(102, 80.0)	(102, "Bob")
(101, 430.0)	(103, "Carol")
(104, 150.0)	

(a) Perform a `join` operation on `orders` and `customers` by `CustomerID`. Write the complete output as a set of key-value pairs. **(2 pts)**

(b) Perform a `cogroup` operation on the same two RDDs. Write the complete output, clearly showing each key and its grouped iterables for both RDDs. Include keys that appear in *only one* of the RDDs. **(2 pts)**

Question 4.2 (3 Points)

A company wishes to count the number of purchases made per product category from a large transaction log. Each line of the log contains: `<TransactionID> <Category> <Amount>`

(a) Describe (in words) what the **Map** function should output for each input record. **(1 pt)**

(b) Describe (in words) what the **Reduce** function should do. **(1 pt)**

(c) Could a **Combiner** be used here? Justify your answer briefly. **(1 pt)**

Question 4.3 (5 Points)

Draw and label a diagram of a Kubernetes cluster that includes the following components and clearly shows how they interact:

- Control Plane: API Server, Scheduler, Controller Manager, `etcd`
- Worker Node(s): kubelet, kube-proxy, Pod(s) with container(s)
- An external user/client issuing a `kubectl` command

Indicate with **arrows** the direction of communication between components (e.g., user → API Server → Scheduler → kubelet). Brief labels on arrows (e.g., "schedules Pod", "watch loop") are encouraged.

Question 4.4 (4 Points)

An engineering team is considering whether to adopt Mesos or Omega as the scheduling backbone for a mixed workload cluster (batch analytics + real-time services + ML training).

(a) State **two key limitations** of Apache Mesos that motivated the design of Omega. **(2 pts)**

(b) Explain how Omega's **optimistic concurrency control** addresses those limitations. What is the main risk or trade-off introduced by this approach? **(2 pts)**

Question 4.5 (4 Points)

(a) In GFS, the master is **not** involved in one particular step of a read operation. Identify that step and explain why the architecture is designed this way. **(2 pts)**

(b) HDFS uses a **NameNode** and multiple **DataNodes**. What is the key single point of failure risk in HDFS, and what mechanism does HDFS use to mitigate it? **(2 pts)**

Part V: Case Study (20 Points)

INSTRUCTION: Choose **exactly ONE** scenario from the five options below. Attempting more than one will result in **zero points** for this entire part.

Circle your chosen scenario number before you begin writing.

Your answer will be graded on: *correctness of concepts* (10 pts), *justification and reasoning* (6 pts), and *clarity of presentation* (4 pts).

Scenario A — Kubernetes Deployment Under Failure

Context: A fintech startup deploys a payment processing microservice on a Kubernetes cluster consisting of 1 Control Plane node and 4 Worker nodes. The payment service runs as a Deployment with `replicas: 3`.
The following sequence of events occurs:

1. A developer runs `kubectl scale deployment payment-service --replicas=5`.
2. Worker Node 2, which hosted 2 running Pods, suddenly crashes.
3. A new Worker Node 5 joins the cluster.

Your Task:

(a) For each event (i–iii), identify **which Kubernetes component(s)** detect the change, make the scheduling decision, and take corrective action. Explain the role of each component in that specific event.

(b) Draw a simplified diagram of the cluster state **after all three events**, showing which Pods run on which nodes (you may use boxes and labels).

(c) The Control Plane node also fails shortly after. What is the impact on the *currently running* Pods? Can the cluster still serve traffic? Explain your answer in terms of Kubernetes architecture.

Scenario B — Mesos vs. Omega: Cluster Redesign

Context: You manage a shared cluster used by four teams:

- **Team Alpha:** Latency-critical web services (must respond within 50ms)
- **Team Beta:** Batch analytics jobs (Spark, runtime hours to days)
- **Team Gamma:** GPU-based ML training (long-running, resource-intensive)
- **Team Delta:** CI/CD pipelines (short-lived, bursty, unpredictable)

Current problems:

- Static quotas lead to resource waste when teams are idle.
- Batch jobs starve latency-critical services during peak hours.
- The centralized scheduler is a bottleneck — scheduling latency is increasing.

Your Task:

(a) Choose **either Mesos or Omega** as the new scheduling backbone. Justify your choice based on the workload characteristics of the four teams. Explicitly argue why the other system would be *less suitable*.

(b) Explain concretely how your chosen system addresses: (i) resource waste from static quotas, (ii) starvation of high-priority services, and (iii) scheduler scalability/bottleneck.

(c) Identify **one remaining limitation** of your chosen system that this cluster design does *not* solve, and propose a mitigation strategy.

Scenario C — Spark Streaming Pipeline Design

Context: A logistics company wants to build a real-time shipment tracking system. Sensors on delivery trucks emit GPS coordinates every 5 seconds. The data is ingested into Apache Kafka and must be processed to:

1. Detect trucks that have been stationary for more than 10 minutes (possible breakdowns).
2. Compute the average speed per truck over a sliding window of 15 minutes.
3. Alert a monitoring dashboard within 30 seconds of anomaly detection.

The engineering team proposes using **Spark Streaming**.
Your Task:

(a) Describe the Spark Streaming processing model (DStream, micro-batch), and explain why it is suitable or unsuitable for the 30-second latency requirement in requirement (iii). Would standard Spark Streaming or Structured Streaming be more appropriate? Justify.

(b) For requirement (ii), describe how you would configure a **windowed operation** in Spark Streaming. Define the window duration, slide interval, and what aggregation would be performed. Draw the window timeline for the first 30 minutes of data.

(c) If a Spark executor node crashes mid-window, how does Spark ensure the window computation is eventually correct? Reference the fault tolerance mechanism.

Scenario D — RDD Lineage and Fault Recovery

Context: A data science team builds the following Spark pipeline to analyze customer purchase data:

1. Load raw transaction records from HDFS into an RDD.
2. Filter out records with amount < 10 (noise).
3. Map each record to `(customerID, amount)` pairs.
4. Join with a customer profile RDD `(customerID, region)`.
5. Group by region and compute total spend per region.
6. Cache the result for downstream reporting queries.

Your Task:

(a) Draw the complete **RDD lineage graph** (DAG) for this pipeline. Label each node with the RDD name and each edge with the transformation applied. Clearly mark which RDD is cached.

(b) Suppose step 4's joined RDD loses 2 of its 8 partitions due to a worker failure. Explain step-by-step how Spark recovers *only those 2 partitions*. Which prior transformations must be re-executed? Which can be skipped?

(c) The team notices the pipeline is slow due to the join operation causing a large shuffle. Propose **two** optimizations they could apply to reduce shuffle cost, and explain the principle behind each.

Scenario E — Multi-Framework Cluster: Mesos in Practice

Context: A media company runs the following workloads on a single Mesos cluster (200 nodes):

Framework	Type	Characteristic
Hadoop MapReduce	Batch	Long-running, fault-tolerant
Apache Storm	Real-time streaming	Short tasks, low latency
Spark	Iterative ML	Memory-intensive, iterative

During peak hours, resource contention spikes. The ops team observes:

- Storm tasks are delayed because Hadoop is holding large resource slots.
- Spark is hoarding memory even between iterations.
- Mesos master scheduling latency has risen to 2 seconds per offer cycle.

Your Task:

(a) Identify which of the three problems above is caused by a **framework behavior** and which is caused by **Mesos architecture limitations**. Explain your reasoning for each.

(b) Mesos supports fine-grained and coarse-grained resource sharing modes. Explain the difference, and argue which mode would better address the Storm latency problem. What trade-off does your choice introduce?

(c) Suppose the team considers migrating from Mesos to Omega to solve the scheduler latency problem. Explain how Omega's architecture would reduce scheduling latency in this scenario. Would it introduce any new risks? Be specific.