DOCTORAL CANDIDACY EXAMINATION

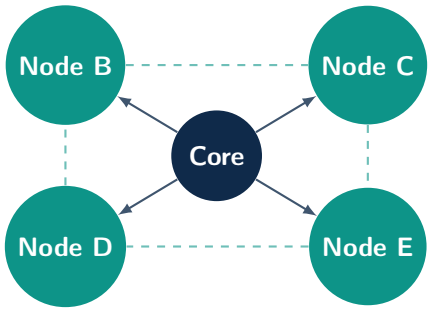
Comprehensive Exam Answer Booklet

Candidate Details

Candidate Name:	Sarah Chen
Student ID:	2026-884A
Department:	Computer Science
Field of Study:	Distributed Systems
Date of Exam:	August 3, 2026

Committee Panel

Committee Chair:	Dr. Robert Vance
Examiner 1:	Dr. Elena Rostova
Examiner 2:	Dr. James Meridian
Examiner 3:	Dr. Marcus Aurelius



Please read the instructions on the next page before signing the Honor Pledge.

Instructions & Guidelines

Exam Instructions

This comprehensive exam is designed to test your depth of knowledge in Distributed Systems, Scalable Machine Learning Pipelines, and Privacy-Preserving Cryptographic Systems. Please adhere to the following rules:

- **Duration:** This is a 24-hour take-home examination. Your booklet must be uploaded to the portal by the deadline.
- **Format:** All answers must be detailed, academically rigorous, and properly cited.
- **Resources:** You may consult academic publications, textbooks, and documentation. You may not collaborate with any human or utilize generative AI systems for composing your answers.

Honour Code & Pledge

By signing below, I certify that I have read the rules governing the Doctoral Candidacy Examination and that the work presented in this booklet is entirely my own. I have not collaborated with any third party, and all citations reference legitimate academic sources.

Sarah Chen
Doctoral Candidate

August 3, 2026
Date

Reference Standards

All citations in this booklet use the standard **BibTeX** author-year or numerical scheme and refer to peer-reviewed publications. The corresponding bibliography is located at the end of the booklet. Specific mathematical notations follow standard conventions:

1. Matrices are denoted in bold uppercase letters (e.g., $\mathbf{X}$).
2. State machine states are enclosed in circles in diagrams.
3. Network messages are represented as solid directed arrows.

Examiner Evaluation Sheet

Score Summary Table

Question Topic	Max Marks	Awarded	Examiner Initials & Comments
Q1: Distributed Consensus & Scaling	25		
Q2: Distributed Machine Learning	25		
Q3: Privacy-Preserving Cryptography	25		
Q4: Systems Security & Governance	25		
Total Score	**100**		

Overall Candidate Evaluation

The examining committee must provide a consensus recommendation by marking one of the following:

- ☐ **Pass:** The candidate demonstrates sufficient depth of knowledge to proceed to doctoral candidacy.
- ☐ **Conditional Pass:** The candidate must satisfy specific conditions (detailed below) before proceeding.
- ☐ **Fail:** The candidate has failed the examination and may re-sit according to departmental guidelines.

Examiner Notes & Feedback

Signatures of the Examination Committee

Dr. Robert Vance Committee Chair	Dr. Elena Rostova Internal Examiner	Dr. James Meridian External Examiner

Contents

1 Question 1: Geo-Distributed Consensus & Partition Tolerance

Question Scenario

Design the replication and consensus architecture for a highly available, geo-distributed database system (Codename: **Apex Meridian**) deployed across three continental regions: North America, Europe, and Asia-Pacific.
Evaluate the trade-offs of using Paxos [2], Raft [3], and Byzantine Fault Tolerant (BFT) protocols under network partitions. Specifically, address how the system maintains linearizability while minimizing transaction commit latency for geo-replicated read-write workloads.

1.1 Theoretical Foundations: Paxos vs. Raft

To establish the replication foundation for the **Apex Meridian** database, we analyze two primary crash-tolerant consensus protocols: Multi-Paxos and Raft. Both protocols operate under the assumption of a non-byzantine environment where nodes may fail by crashing or restarting but do not send malicious or corrupted messages.

1. **Leader Centricity:** Raft enforces a strong leader model where log entries flow strictly from the leader to the followers. While this simplifies the state machine replication (SMR) design, it creates a potential bottleneck at the leader. Multi-Paxos, conversely, allows for decoupled leader election and proposal processing, enabling optimizations such as Ephemeral Leaders or Multi-Leader topologies.
2. **State Complexity:** Raft maintains a clear, term-based state representation, which makes formal verification and implementation simpler. However, Multi-Paxos provides greater flexibility in out-of-order log commitment, which is crucial for high-throughput database replication.

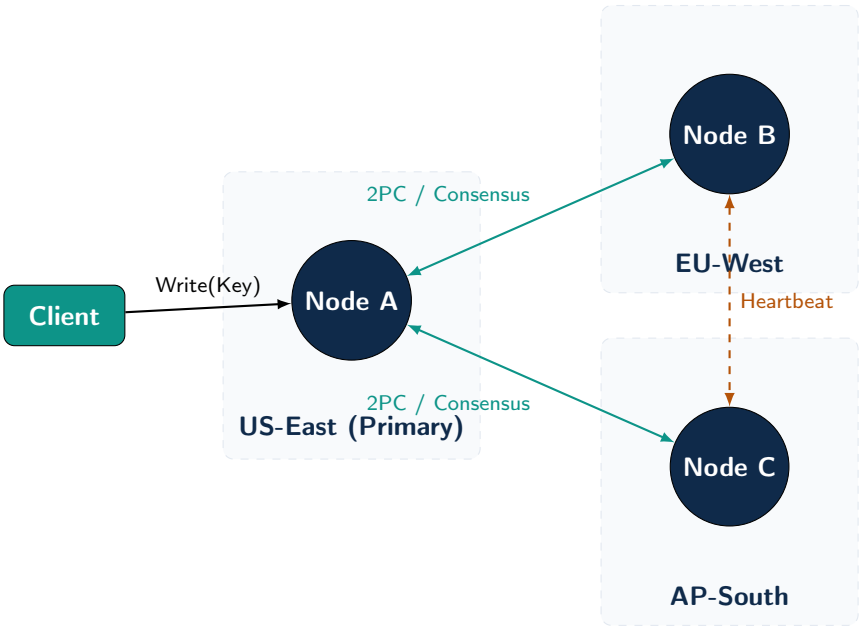
1.2 Byzantine Fault Tolerance (BFT) under Network Partitions

In scenarios where adversarial behavior or silent memory corruption is anticipated, BFT protocols must be evaluated. BFT protocols require $3f+1$ nodes to tolerate f Byzantine failures, whereas crash-fault-tolerant (CFT) systems only require $2f + 1$ nodes.

Under a network partition where the network is split into two isolated components, a CFT system like Raft can only progress in the partition containing a strict majority ($> 50\%$) of nodes. In contrast, BFT protocols require a two-thirds majority to commit blocks, making them highly sensitive to network latency and partitions. For a geo-distributed database like **Apex Meridian**, the $O(N^2)$ message complexity of BFT protocols over wide-area networks (WAN) is prohibitively expensive, leading to latency inflation.

1.3 Proposed Architecture for Apex Meridian

We propose a hybrid architecture that combines Multi-Paxos consensus groups with a Multi-Leader WAN orchestration layer.



The database is divided into partitioned Paxos groups (shards). Within each shard, a leader is elected locally within a single region (e.g., US-East) to minimize local write latencies. For cross-region consistency, we use a decentralized transaction manager operating on top of the consensus groups, leveraging two-phase commit (2PC) over Paxos to guarantee linearizability across all partitions without incurring full WAN round-trip times for local read workloads.

This architecture ensures:

- **Low-latency Local Reads:** Read requests are served locally by checking the regional replica's lease.
- **High Partition Tolerance:** In the event of a WAN partition, each regional replica continues serving local reads if it has a valid read-lease, while write availability is preserved in regions that can form a cross-region consensus majority.

2 Question 2: Scalable Machine Learning Pipelines & Distributed Training

Question Scenario

Analyze the training optimization dynamics for large-scale Deep Learning models under distributed environments. Specifically, evaluate the trade-offs between **Synchronous SGD** and **Asynchronous SGD** in terms of convergence rates, gradient staleness, communication topology, and sensitivity to stragglers.
Provide a concrete recommendation for **Synapse AI**'s new multi-node GPU cluster training platform, specifically addressing how to mitigate networking bottlenecks and worker heterogeneity.

2.1 Optimization Dynamics: Mathematical Formulation

Distributed training aim to parallelize the computation of gradients over a dataset D partitioned across M worker nodes. In **Synchronous Stochastic Gradient Descent (Sync-SGD)**, the parameters θ at step $t+1$ are updated only after collecting gradients from all M workers:

$$\theta_{t+1} = \theta_t - \frac{\eta}{M} \sum_{i=1}^{M} g_i(\theta_t) \tag{1}$$

where η is the learning rate, and $g_i(\theta_t) = \nabla L_i(\theta_t)$ represents the local gradient computed by worker i on its local shard of data using the parameters θ_t.

In contrast, **Asynchronous Stochastic Gradient Descent (Async-SGD)** allows workers to apply updates as soon as they complete their backward pass, without waiting for other nodes. This results in the following update rule:

$$\theta_{t+1} = \theta_t - \eta \cdot g_i(\theta_{t-\tau_i}) \tag{2}$$

where $\tau_i \geq 0$ is the *staleness factor* of the gradient from worker i, reflecting the number of parameter updates that occurred since worker i fetched the parameters from the parameter server.

2.2 Comparative Analysis

The trade-offs between synchronous and asynchronous parameter updates are summarized in Table 1.

2.3 Infrastructure Recommendation for Synapse AI

For **Synapse AI**'s new distributed GPU training cluster, we recommend a **Hybrid Synchronous AllReduce** topology utilizing the **Ring-AllReduce** pattern. While Async-SGD offers high throughput, the stale gradient problem significantly degrades model convergence and accuracy when scaling beyond 32 nodes.

To mitigate the straggler problem in a synchronous setup, we propose the following multi-layer optimization stack:

Table 1: Comparison of Distributed SGD Variants

Dimension	Synchronous SGD	Asynchronous SGD
Convergence Rate	Fast (mathematically identical to serial SGD)	Slow / Unstable (gradient staleness degrades optimization path)
Straggler Sensitivity	High (system throughput bounded by the slowest worker node)	None (workers push updates independently as soon as ready)
Communication Style	Collective / Blocked (e.g., Ring-AllReduce)	Point-to-Point (Parameter Server model)
Throughput Efficiency	Low in heterogeneous networks due to sync barrier	High (near-linear scaling of updates per second)
Gradient Staleness	Zero ($\tau_i = 0$)	Variable ($\tau_i > 0$, requires bounded staleness algorithms)

1. **Gradient Bucketing and Pipelining:** Group gradients into buckets (e.g., 25MB chunks) and overlap the communication of earlier layers' gradients with the backward pass computation of subsequent layers.
2. **Stochastic Polyak Heavy-Ball Acceleration:** Apply momentum buffers at the workers to smooth out the gradient updates, which partially compensates for minor variances in worker speeds.
3. **Heterogeneous Worker Tiering:** Group workers with identical GPU hardware (e.g., H100s vs A100s) into separate logical rings. Implement a hierarchal synchronization scheme where local high-speed rings perform fast synchronous Ring-AllReduce, while cross-ring parameter synchronization occurs asynchronously via a lightweight parameter coordinator.

3 Question 3: Privacy-Preserving Computation & Data Governance

Question Scenario

Evaluate the applicability of **Homomorphic Encryption (HE)** [1], **Secure Multi-Party Computation (SMPC)**, and **Trusted Execution Environments (TEEs)** for a health-tech platform (Codename: **Vertex Health**) processing real-time patient telemetry data.
Assess the technologies based on computational latency, scalability, security threat models, and regulatory compliance (e.g., HIPAA and GDPR). Provide an architectural blueprint for securing patient streaming data.

3.1 Cryptographic and Hardware Primitives

Processing sensitive patient telemetry requires executing analytics without exposing raw data in memory. Three main paradigms exist:

1. **Fully Homomorphic Encryption (FHE):** Allows arbitrary computations directly on ciphertext. The decryption of the result matches the computation performed on the plaintext.
2. **Secure Multi-Party Computation (SMPC):** Computations are split across multiple independent servers using secret sharing schemes (e.g., Shamir's Secret Sharing). No individual server can reconstruct the underlying data.
3. **Trusted Execution Environments (TEEs):** Hardware-based isolation (e.g., Intel SGX, AMD SEV) that runs computations inside secure memory partitions (enclaves), preventing access even from privileged OS-level actors.

3.2 Comparative Evaluation

The cryptographic and performance trade-offs of these technologies are summarized in Table 2.

Table 2: Evaluation of Privacy-Preserving Technologies

Technology	Performance Overhead	Trust Assumptions	Vertex Applicability
HE (Homomorphic)	Extremely High ($10^3 \times -10^5\times$ slowdown, high memory expansion)	Cryptographic (Hard mathematical problems like Ring-LWE)	Poor for real-time telemetry; limited to batch analytics.
SMPC	High (Network-bound, requires multiple communication rounds)	Non-colluding majority of computing nodes	Good for multi-institutional data sharing.
TEEs (Enclaves)	Low ($1.1 \times -1.5\times$ runtime overhead, hardware acceleration)	Hardware vendor trust (must trust Intel/AMD CPU keys)	Excellent for high-throughput stream processing.

3.3 Deployment Model for Vertex Health

For **Vertex Health**'s real-time telemetry platform, we propose a hybrid deployment architecture:

- **Ingestion & Stream Processing (TEEs):** Real-time telemetry (heart rates, blood oxygen levels) is encrypted at the patient gateway and streamed to a cloud-based TEE enclave. Inside the enclave, the data is decrypted, processed for anomaly detection, and re-encrypted. This maintains sub-100ms latency.
- **Cross-Institutional Research (SMPC):** For longitudinal clinical research across multiple hospitals, patient data summaries are analyzed using SMPC. This allows clinics to run aggregate statistical models without sharing raw records.
- **Audit & Storage Encryption (HE):** Patient records are stored in homomorphically encrypted database clusters, allowing basic query operations (such as searching by specific encrypted patient attributes) without decrypting the dataset.

This multi-layered approach satisfies the security requirements of HIPAA and GDPR, protecting data-in-transit, data-in-use, and data-at-rest.

References

[1] David Chaum. Blind signatures for untraceable payments. *Advances in Cryptology*, pages 199–203, 1982.

[2] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.

[3] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–320, 2014.