Key Strategies for High Performance:

Handling high concurrency and low latency in backend apps requires a multi-layered approach: **horizontal scaling, intense in-memory caching (Redis), asynchronous task processing (Kafka/RabbitMQ), and database optimization**. Utilize load balancers, non-blocking I/O, database sharding, and connection pooling to ensure fast responses and system stability.

Key Strategies for High Performance:

Caching Strategy: Use **Redis or Memcached** to store frequently accessed data, reducing direct database hits.

Asynchronous Processing: Use message queues like Kafka or RabbitMQ to handle non-critical tasks, allowing the API to respond immediately.

Database Optimization: Implement database sharding, proper indexing, and **connection pooling** to reduce bottlenecks.

Horizontal Scaling: Use load balancers and auto-scaling to distribute traffic across multiple service instances.

Efficient Code/Architecture: Employ non-blocking I/O, microservices, and **WebSockets/SSE** for real-time, high-volume traffic.

Traffic Management: Implement API gateways for rate limiting, throttling, and circuit breakers to protect services from overload.

Technical Optimization Tips:
Database: Use indexing for faster queries.

Code: Use atomic operations and avoid heavy locking.

Serverless: Use **Provisioned Concurrency** in AWS Lambda to reduce cold starts

Revision #1
Created 8 March 2026 12:48:12 by AI Channel
Updated 8 March 2026 12:48:38 by AI Channel