Architecture & Design Goals

Storage Systems (GFS, Bigtable, DynamoDB)
- GFS: Single master with multiple chunkservers design, optimized for large sequential reads/writes[1]
- Bigtable: Distributed storage system for structured data, built on GFS, uses tablet servers for data management[2]
- DynamoDB: Fully managed multi-tenant architecture with partition-based data distribution and global access control[8]

Processing Systems (MapReduce, Kafka)
- MapReduce: Master-worker architecture for parallel data processing using map and reduce functions[7]
- Kafka: Distributed messaging system with brokers handling partitioned topics and producers/consumers[4]

Resource Management
- Borg: Centralized cluster management system handling 10K+ machines per cell through Borgmaster[3]

Key Features

Data Model
- GFS: File-based with chunks as basic units[1]
- Bigtable: Sparse, distributed multi-dimensional sorted map[2]
- DynamoDB: Key-value and document model with flexible schemas[8]
- Kafka: Topic-based publish-subscribe with partitioned message streams[4]

Scalability
- GFS: Scales through chunk distribution and replication[1]
- Bigtable: Horizontal scaling through tablet splitting[2]
- Borg: Scales through cell-based architecture and resource allocation[3]
- DynamoDB: Boundless table scaling with elastic partition management[8]

Fault Tolerance

Replication Strategies
- GFS: 3-way chunk replication[1]
- DynamoDB: Uses log replicas across availability zones[6]
- Kafka: Topic partition replication across brokers[4]

Failure Handling
- MapReduce: Automatic task re-execution on failures[5]
- Borg: Worker health monitoring and task rescheduling[3]
- DynamoDB: Automatic failover and recovery mechanisms[8]

Performance Optimization

Data Locality
- GFS: Places chunks near computation[1]
- MapReduce: Schedules tasks near data location[7]
- Kafka: Sequential disk access and zero-copy data transfer[4]

Throughput vs Latency
- GFS: Optimized for throughput over latency[1]
- DynamoDB: Focuses on predictable low-latency performance[8]
- Kafka: High-throughput message processing with sequential I/O[4]

Consistency Model

Consistency Guarantees
- GFS: Relaxed consistency with atomic file namespace operations[1]
- Bigtable: Strong consistency within tablets[2]
- DynamoDB: Configurable consistency levels (eventual or strong)[8]
- Kafka: Per-partition ordering with at-least-once delivery[4]

This comparison shows how each system was designed for specific use cases while sharing common distributed systems principles for scalability, fault tolerance, and performance optimization.