Cloud and Parallel Computing

Study-Ready Notes

Compiled by Andrew Photinakis

November 15, 2025

Contents

1 Introduction to Cloud Computing

1.1 Motivation for Cloud Computing

- **Business Perspective:**
  - Cost reduction through shared infrastructure
  - Scalability to meet fluctuating demand
  - Access to enterprise-level computing resources
- **Research Perspective:**
  - Access to high-performance computing without capital investment
  - Ability to process large datasets
  - Collaborative research across institutions

[Summary: Cloud computing provides scalable, cost-effective computing resources for both business and research applications, eliminating the need for significant upfront hardware investments.]

1.2 Definition of Cloud Computing

- Distributed systems of virtualized computers
- Provides services to match providers and consumers
- Resources are abstracted through virtualization

2 Types of Cloud Computing

2.1 Deployment Models

- **Public Cloud:** Services available to general public over internet
- **Private Cloud:** Dedicated infrastructure for single organization
- **Edge Computing:** Processing at network edge near data sources

2.2 Service Models

- **Infrastructure as a Service (IaaS):** Virtual machines, storage, networking
- **Platform as a Service (PaaS):** Development platforms and tools
- **Software as a Service (SaaS):** Applications delivered over web

[Concept Map: Cloud Types → Deployment (Public/Private/Edge) × Service (IaaS/PaaS/SaaS)]

3 Benefits of Cloud Computing

3.1 Service Provider Benefits

- Dynamic scaling to match customer demand
- Cost advantages through resource sharing
- On-demand service delivery
- Information hiding through virtualization

3.2 Consumer Benefits

- Seemingly infinite resource pool
- Little additional capital expenses
- On-demand resource provisioning
- Robust to failures (no single point of failure)
- Cost reduction and security/privacy features

4 Cloud Service Models

4.1 Everything as a Service (XaaS)

- **Infrastructure as a Service (IaaS):**
  - Virtual machine instances
  - Storage services
  - Dynamic resource provisioning
- **Platform as a Service (PaaS):**
  - Development frameworks
  - Middleware services
  - Deployment platforms
- **Software as a Service (SaaS):**
  - Web-based applications
  - No local installation required
  - Automatic updates

[Summary: Cloud services provide the illusion of unique, secure infrastructure with dynamic scaling, security, and performance guarantees through various service models.]

4.2 Infrastructure as a Service (IaaS)

- Access to parallel computers (virtual machines)
- Dynamic provisioning of computing resources
- Suitable for parallelizable applications:
  - Weather modeling and prediction
  - Financial analysis and risk modeling
  - Scientific simulations

5 Parallel Computing in Cloud Environments

5.1 Cloud Processing Architecture

- Multiple processing units with many cores
- GPU acceleration support
- Efficient resource utilization through:
  - Task scheduling and load balancing
  - Performance optimization
  - Energy consumption management
  - Monetary cost optimization
- Access to distributed cloud storage
- MapReduce for large-scale data processing

5.2 Cost Models for Parallel Computing

5.2.1 Traditional Parallel Computing Cost

$$\text{Cost} = T_p \times P$$

Where:

- T_p = Execution time
- P = Number of processors

5.2.2 Cloud Computing Cost

$$\text{Cost} = T_p \times P \times C_x$$

Where:

- C_x = Service provider cost per processor (in dollars)
- C_x may vary with time of day or week
- Multi-objective optimization required:
  - Latency minimization
  - Monetary cost reduction
  - Energy consumption optimization

[Mnemonic: Cloud Cost = Time × Processors × Variable Rate]

6 Cloud Computing Models

6.1 MapReduce for Large Files

- Designed for read/write operations with large files
- Not optimized for inter-processor communications
- Suitable for batch processing of large datasets

6.2 MPI for Multi-Unit Systems

- Message Passing Interface support needed
- Enables communications among systems/processors
- Better for tightly-coupled parallel applications

7 Elasticity in Cloud Computing

7.1 Definition and Importance

- Ability to meet varying computational demands
- Dynamic resource allocation based on workload

7.2 Elastic Parallel Systems

- Control number of processing units at run-time
- Elastic controller for optimal resource utilization
- Number of processor units is time-dependent: $P(t)$
- Cost per processor may vary with time: $C_x(t)$
- Support for heterogeneous processing units

[Summary: Elasticity allows cloud systems to dynamically adjust computing resources in response to changing workloads, optimizing both performance and cost.]

7.3 Challenges in Elastic Parallel Computing

- Complex relationship between:
  - Number and duration of computing steps
  - Processor capacity and execution time
  - Elastic speedup and efficiency metrics

8 Designing Elastic Parallel Systems

8.1 Key Considerations

- **Application Analysis:**
  - Workload patterns of target problems
  - Resource intensity variations
- **Resource Mapping:**
  - Match workload patterns to available cloud resources
  - Cost-efficiency tradeoff analysis
- **Scaling Strategies:**
  - Horizontal vs. vertical scaling decisions
  - Evaluation of elastic parallel systems
- **Control Mechanisms:**
  - Understanding elasticity-related opportunities
  - Implementation of dynamic resource controllers

9 MapReduce Architecture

9.1 System Components

- **Input Data:** Large datasets divided into chunks
- **Map Phase:** Multiple worker nodes process data chunks
- **Master Node:** Coordinates overall process
- **Intermediary Results:** Temporary output from map phase
- **Reduce Phase:** Worker nodes combine intermediary results
- **Final Results:** Output written to file system

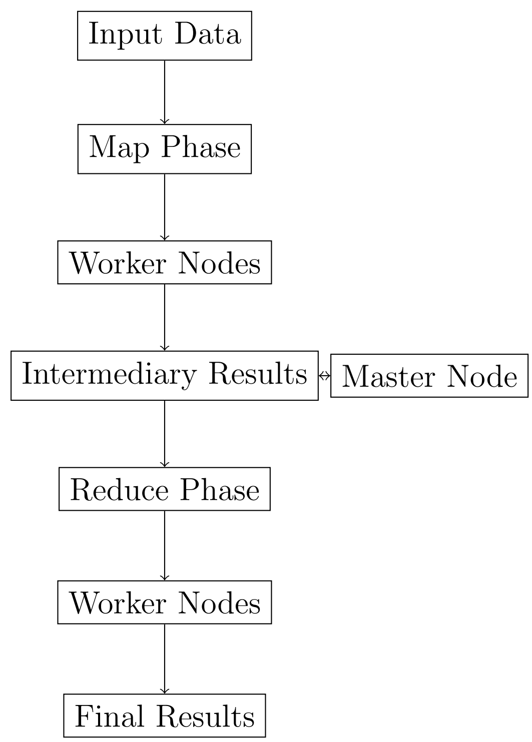

Figure 1: MapReduce Architecture Diagram

9.2 MapReduce Process Flow

1. **Input Partitioning:**
   - Break input data into small chunks
   - Data typically stored in distributed file system (e.g., Google File System)
2. **Map Phase:**

- Initial processing on data chunks
- Produce intermediary key-value pairs
- Results buffered in distributed storage

3. **Reduce Phase:**

- Combine intermediary results
- Produce final output
- Results stored in output files

10 MapReduce Examples and Applications

10.1 Common Use Cases

Function	**Map Phase**	**Intermediate Step**	**Reduce Phase**
Word Count	For each word occurrence, emit ⟨word, 1⟩	Merge/sort by key	Count 1s for each word
Grep	Output lines matching pattern	Identity operation	Concatenate results
Sort	Emit key-value pairs for sorting	Sort by key	Identity operation
Inverted Index	Emit ⟨word, document ID⟩ pairs	Group by word	Produce sorted document lists

Table 1: MapReduce Examples and Their Processing Steps

[Mnemonic: Map = Process chunks, Reduce = Combine results]

10.2 Main Processing Steps

- **Data Partitioning:**
  - Key-value operations over different data chunks
  - Parallelism achieved through input data division
- **Intermediate Processing:**
  - Compute key-value/associated value pairs
  - Parallel computation of intermediate data
- **Reduction Phase:**
  - Merge and/or group pairs
  - Separate reduction per group
  - Parallel reduction over groups

11 MapReduce Advantages and Limitations

11.1 Positive Aspects

- **Simplicity and Ease of Use:**
  - Programmer doesn't specify job distribution
  - Number of map tasks depends on input blocks, not processors
- **Flexibility:**
  - Handles irregular and unstructured data
  - Adaptable to various data formats
- **Fault Tolerance:**
  - Resilient to worker failures
  - Master pings workers and reschedules if necessary
- **Scalability:**
  - Highly scalable across many nodes
  - Linear scaling with data size

11.2 Challenges and Limitations

- **Low Efficiency:**
  - Poor I/O efficiency due to disk operations
  - Map and Reduce are blocking operations
- **Synchronization Overhead:**
  - No transition to next stage until current stage completes
  - Difficult to implement pipelined operations
- **High Latency:**
  - Batch processing nature introduces delays
  - Not suitable for real-time processing

[Summary: MapReduce provides simple, fault-tolerant data processing but suffers from I/O inefficiency and high latency due to its batch-oriented, synchronized design.]

12 Advanced Example: Minimum Spanning Tree with MapReduce

12.1 Problem Formulation

Given graph $G(V, E)$ where:

- V = set of vertices
- E = set of edges
- N = number of vertices

12.2 MapReduce Algorithm Steps

1. **Random Partitioning:**
    - Partition vertices into k equal-sized subsets
    - $V_1 \cup V_2 \cup V_3 \cup \cdots \cup V_k = V$
    - $V_i \cap V_j = \emptyset$ for $i \neq j$
    - Each subset contains N/k vertices
2. **Edge Set Definition:**
    - $E_{i,j} \subseteq E$ is the set of edges induced by $V_i \cup V_j$
    - $E_{i,j} = \{(u, v) \in E \mid u, v \in V_i \cup V_j\}$
    - $G_{i,j} = (V_i \cup V_j, E_{i,j})$
    - Total of $\binom{k}{2}$ subgraphs $G_{i,j}$ ("Chunks")
3. **Map Phase:**
    - Compute unique minimum spanning forest $M_{i,j}$ for each $G_{i,j}$
4. **Aggregation:**
    - Compute $H = (V, \bigcup_{i,j} M_{i,j})$
5. **Reduce Phase:**
    - Compute M = minimum spanning tree of H

12.3 Mathematical Formulation

$$\text{Partition: } V = \bigcup_{i=1}^{k} V_i, \quad V_i \cap V_j = \emptyset \text{ for } i \neq j$$

$$\text{Subgraphs: } G_{i,j} = (V_i \cup V_j, E_{i,j}), \quad \text{where } E_{i,j} = \{(u,v) \in E \mid u,v \in V_i \cup V_j\}$$

$$\text{Map: } M_{i,j} = \text{MST}(G_{i,j})$$

$$\text{Aggregate: } H = \left(V, \bigcup_{1 \leq i < j \leq k} M_{i,j} \right)$$

$$\text{Reduce: } M = \text{MST}(H)$$

[Concept Map: MST with MapReduce → Partition → Local MSTs (Map) → Combine (Aggregate) → Global MST (Reduce)]

Reference: Karloff, Howard, Siddharth Suri, and Sergei Vassilvitskii. "A model of computation for MapReduce." In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, pp. 938-948. Society for Industrial and Applied Mathematics, 2010.