Distributed Information Processing

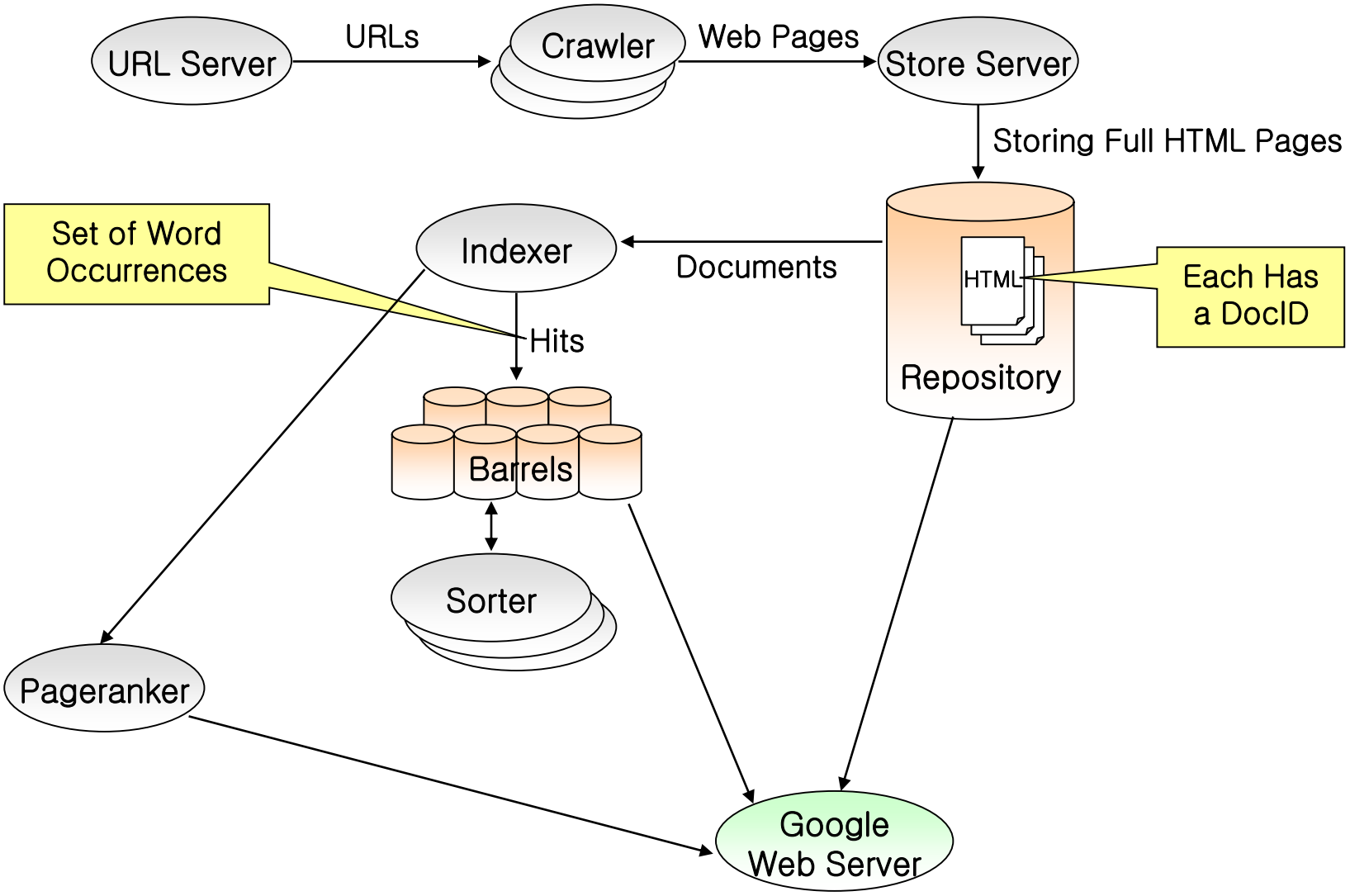
9th Lecture

Eom, Hyeonsang (엄현상)
Department of Computer Science & Engineering
Seoul National University

Outline

- Distributed File Systems
  - Google Architecture
- Q&A

Google Overall Architecture



Google Basic Architecture

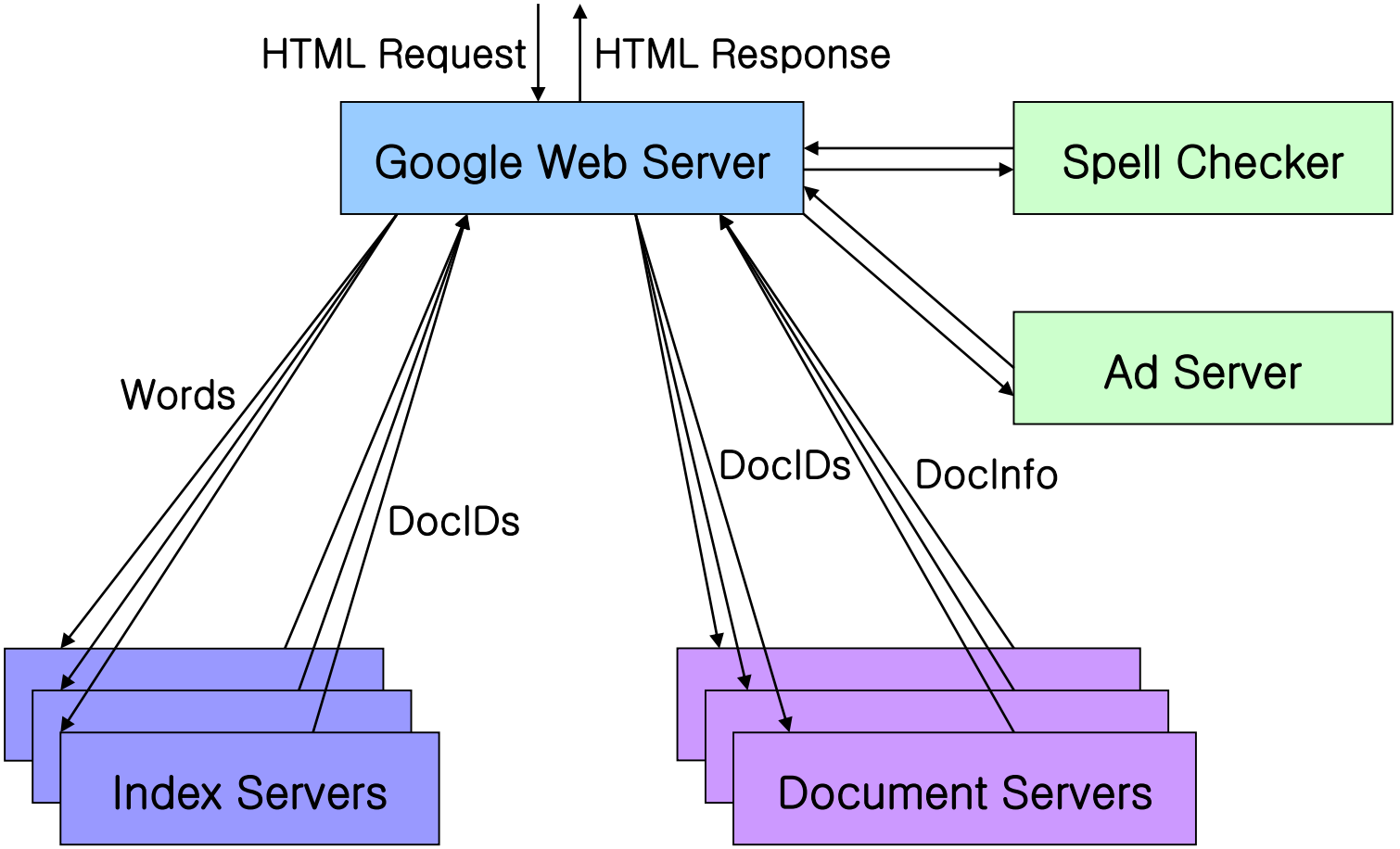
- Cluster Architecture with Unreliable PCs
  - Reliability in Software
    - Using commodity PCs to build a high-end computing cluster
      - Replicating services across machines
      - Detecting and handling failures automatically
  - Aggregate Request Throughput
    - Parallelizing Individual Requests

Google Service Architecture

- Multiple Clusters Distributed Worldwide with Each Cluster Having Thousands of Machines
  - DNS-Based System
    - Selecting a cluster based on the user's geographic proximity
- Multiple Google Web Servers (GWSs) in a Cluster
  - Hardware-Based Load Balancer
    - Selecting a GWS

Load Balancing

Google Query-Serving Architecture



Serving a Query

- Index Servers
  - Consulting an Inverted Index Mapping Each Query Word to a List of Docs (Hit List)
    - Search in parallel
      - Each query is served by one machine or machines within a pool for each index piece (shard)
  - Determining Relevant Docs by Intersecting the Resulting Hit Lists
  - Computing the Relevance Score for Each Doc
  - Producing an Ordered List of Docids

Randomly Chosen; Replicated for Uninterrupted Service

Determining the Order of Results

Serving a Query (Cont'd)

- Doc Servers
  - Computing the Actual Title, URI, & Doc Summary for the Doc By Fetching the Doc with Each Docid
    - Fetch in parallel
      - Each computation is performed by a server within a pool (of multiple replicas) for each shard

Chosen by a Load Balancer

Randomly Chosen Docs

- Spell Checker
- Ad Server
  - Generating Relevant Advertisements

Key Observations

- Most Accesses Being Read-Only
- Relatively Infrequent Updates
- Safely Divertible Queries at Updates
  - Sidestepping many of consistency issues
- Much Inherent Parallelism in the Application
  - Parallelizing the Search over Many Machines
  - Parallelizing the Service
  - Adding Machines for Capacity Increase and Index Growth

We Can Divide Computation Across More CPUs and Disks to Answer a Query Fast.

Quiz: Can We Increase the Number of Shards to Accommodate Slower CPUs?

Key Design Principles

- Software Reliability
  - Tolerating Failures in Software
- Replication for Performance and Availability
  - Replicating Services across Machines
- Low Price/Performance Rather than Peak Performance
  - Using PCs with the Best Performance per Unit Price
- Low Cost Computation
  - Using Commodity PCs

Large Cluster vs Shared Memory (SM) Machines

Condition	Cluster	SM Machines
Low Comp-to-Comm Ratio		○
Dynamic Comm Patterns or Data Partitioning		○
High Management or S/W Licensing Cost		○
Balanceable Index Data and Computation	○	
Frequent Component Failures	○	
Extensive Automation or Monitoring Using In-House Software	○	