The Google Cluster Architecture

Technology behind the Scenes

張峰誠

References

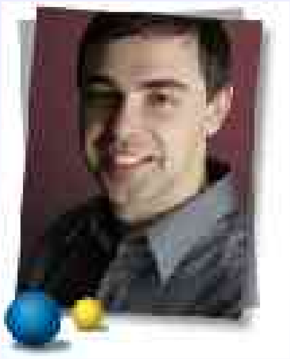
- L.A. Barroso, J. Dean, and U. H?lzle, "Web Search for a Planet: The Google Cluster Architecture," IEEE Micro, March-April 2003, pp.22-28.
- GoogleRank
  http://www.googlerank.com/ranking/pagerank.html
  - The Anatomy of a Large-Scale Hypertextual Web Search Engine
  - PageRank Explained

Outlines

- Architecture
  - Searching
  - Indexing
- Concerns
  - Load balance
  - Fault Tolerance
  - Cost/Performance
  - Power Problem
- Conclusions

Introduction

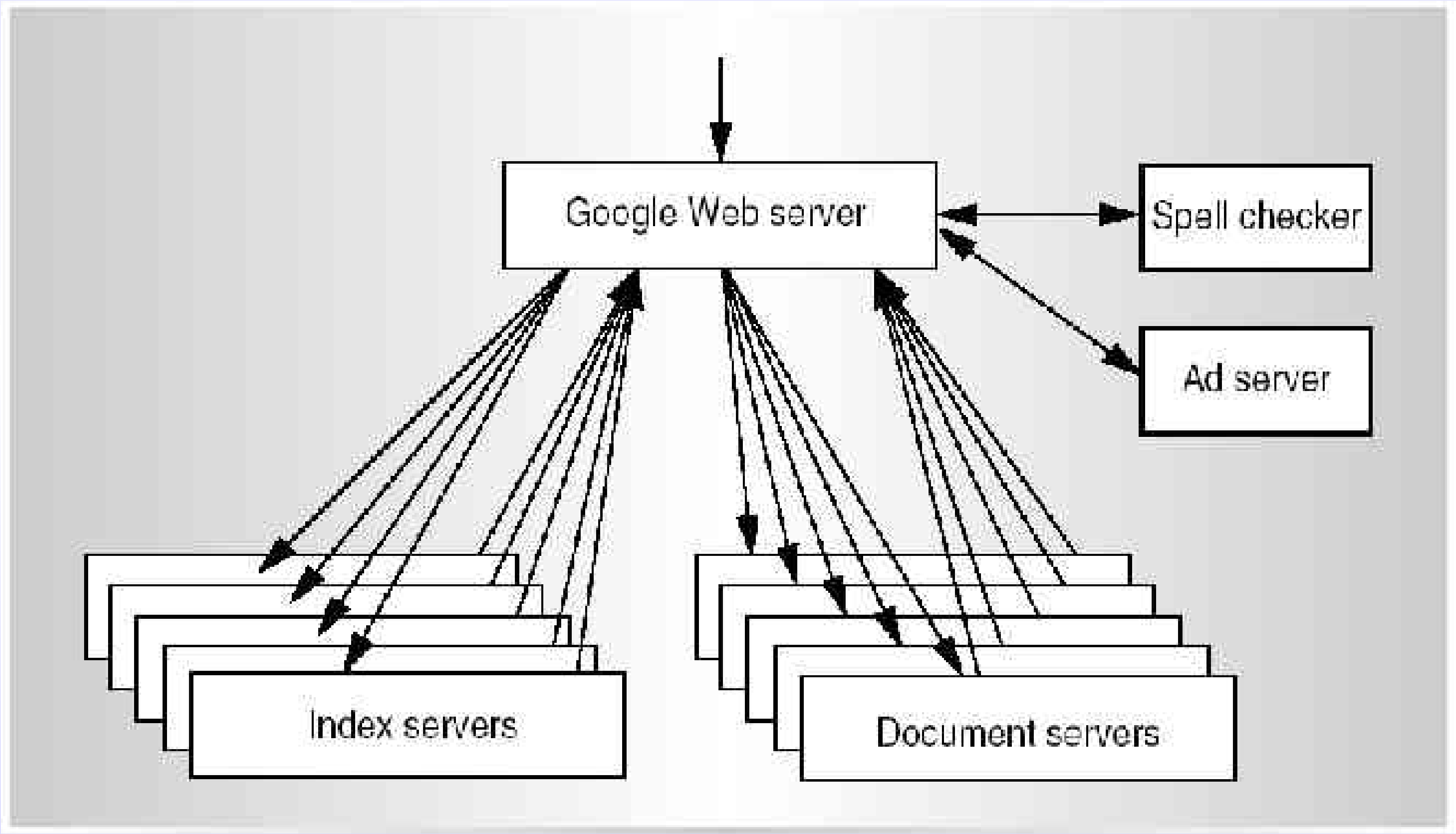
- Web search engines scale up
- Google: scaling with the web
  - Eliminate junk results
  - PageRank
    - Developed by Larry Page and Sergey Brin
  - 15,000+ commodity-class PCs



Searching

- Browser --> DNS
- Browser --> Google Web Server (GWS)
- GWS --> Index Server (send keywords)
- Index Server --> GWS (send docIDs)
- GWS --> Document Server (extract docs)
- GWS format results in HTML
- GWS --> Browser

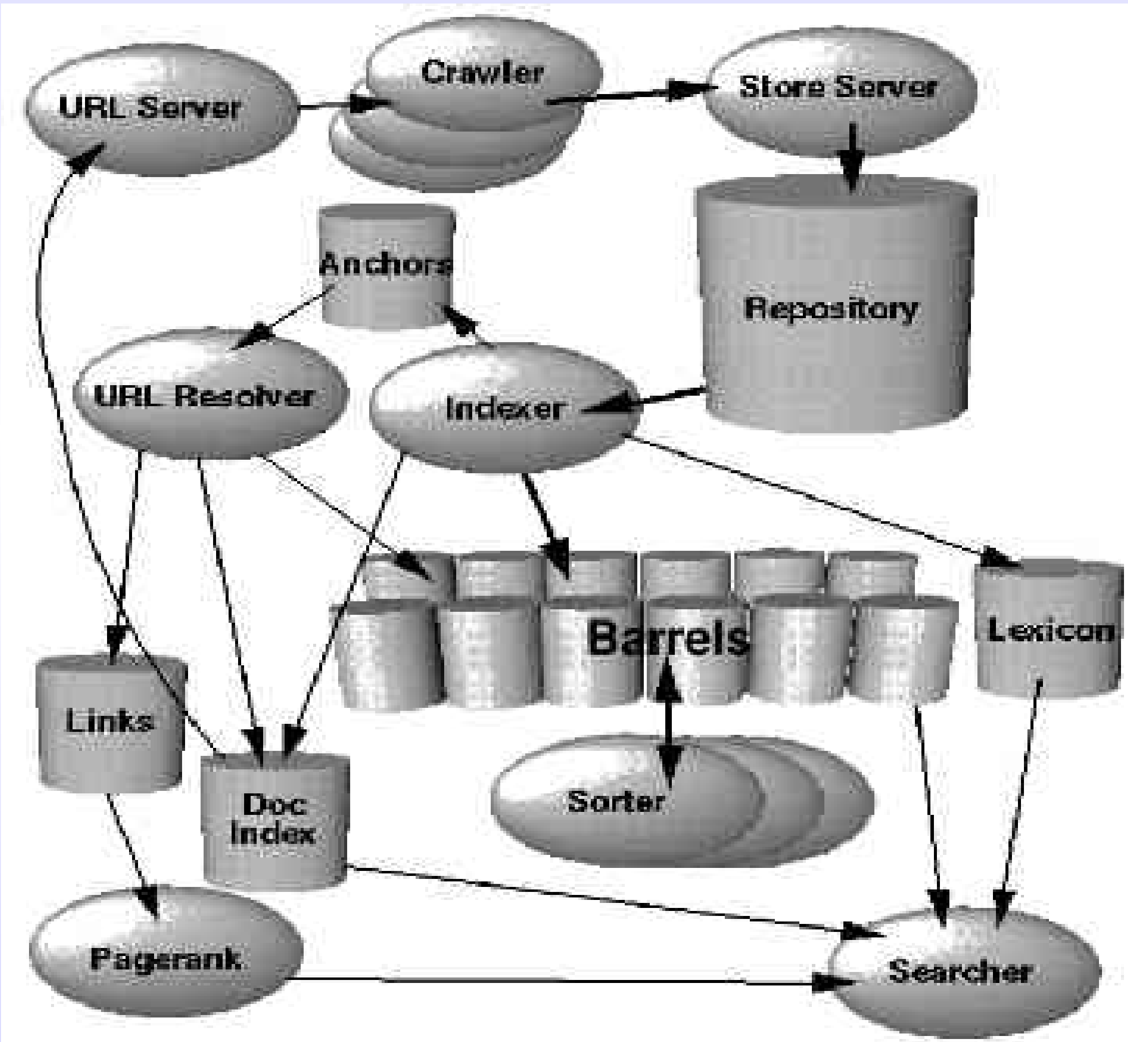
Searching Architecture



Indexing

- URL server and crawler fetches docs
- Indexer generates forward index (doc->word)
- Distribute index into different barrels
- Sorter generates backward index (word->doc)

Indexing Architecture



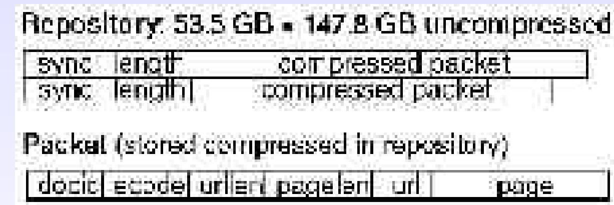


Figure 2. Repository Data Structure

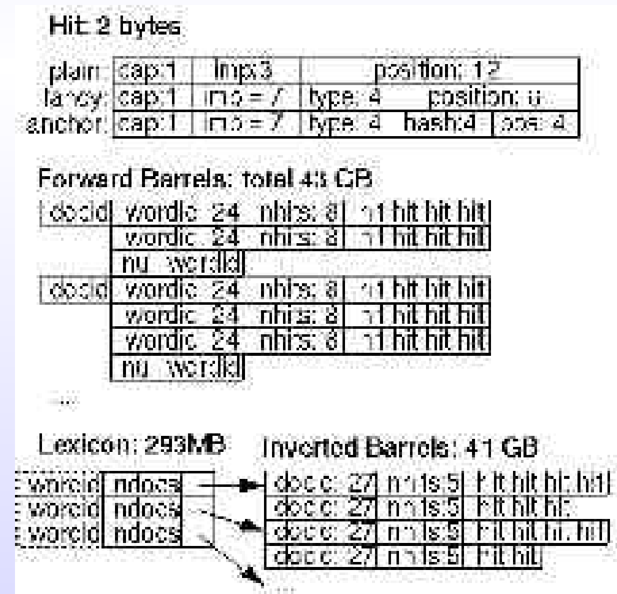


Figure 3. Forward and Reverse Indexes and the Lexicon

Query Distribution

- DNS map to near cluster (DNS load balancing)
- Request route to an available GWS (hardware load balancing)
  - The GWS coordinates query execution and formats collected results
  - Optionally performs value-added tasks

Query Execution

- Phase 1
  - Keywords --> relevant documents and hits
  - Relevant score for each document
- Phase 2
  - DocIDs --> titles, contexts, and URLs

Phase 1 Challenge

- Large amount of index data
  - Doc: 53.5GB Compressed, 147.8 GB uncompressed
  - Forward index: 43GB
  - Lexicon+Backward index: 293MB+41GB
- Resolution
  - Dividing index into pieces (index shards)
  - Duplicate each shard to machines
  - A pool of machines handles requests for each shard

Phase 2 Challenge

- Large amount of document data
  - Title, URL, keyword, and summery information are generated on-line
- Resolution
  - Document shards
  - Pool of machines
  - Cached pages

Replication and Faults

- Most accesses to Google are read-only
- Updates are infrequent
- Diverting queries away from updating services
- Fault tolerance:
  - Hardware failure --> decrease service capacity
  - Provide non-stop service

Parallelism

- Lookup of matching docs in a large index
  --> many lookups in a set of smaller indexes
  followed by a merge step
- A query stream
  --> multiple streams
  (each handled by a cluster)
- Adding machines to a pool increases serving capacity

PageRank

- PageRank™ is the core technology to measure the importance of a page
- Google's theory
  - If page A links to page B
    - Page B is important
    - The link text is irrelevant
  - If many important links point to page A
    - Links from page A are also important

Design Principles

- Software reliability
- Use replication for better request throughput and availability
- Price/performance beats peak performance
- Using commodity PCs reduces the cost of computation

Cost/Performance

- Criterion: cost per query
- Cost
  - Capital expense with depreciation
    - Hardware lasts two to three years
  - Operating costs
    - Hosting
    - Administration
    - Repair

The Power Problem

- High density of machines (racks)
  - High power consumption 400-700 W/ft^2
    - Typical data center provides 70-150 W/ft^2
    - Energy costs
  - Heating
    - Cooling system costs
- Reducing power
  - Reduce performance (c/p may not reduce!)
  - Faster hardware depreciation (cost up!)

Hardware Level Consideration

- Instruction level parallelism does not help
- Multiple simple, in-order, short-pipeline core
- Thread level parallelism
- Memory system with moderate sized L2 cache is enough
- Large shared-memory machines are not required to boost the performance

Conclusions

- For a large scale web service system like Google
  - Design the algorithm which can be easily parallelized
  - Design the architecture using replication to achieve distributed computing/storage and fault tolerance
  - Be aware of the power problem which significantly restricts the use of parallelism