

CS 417 – DISTRIBUTED SYSTEMS

Week 12: Infrastructure  
[Original] Google Cluster Architecture



Paul Krzyzanowski

© 2022 Paul Krzyzanowski. No part of this content may be reproduced or reposted in whole or in part in any manner without the permission of the copyright owner.

# A note about relevancy

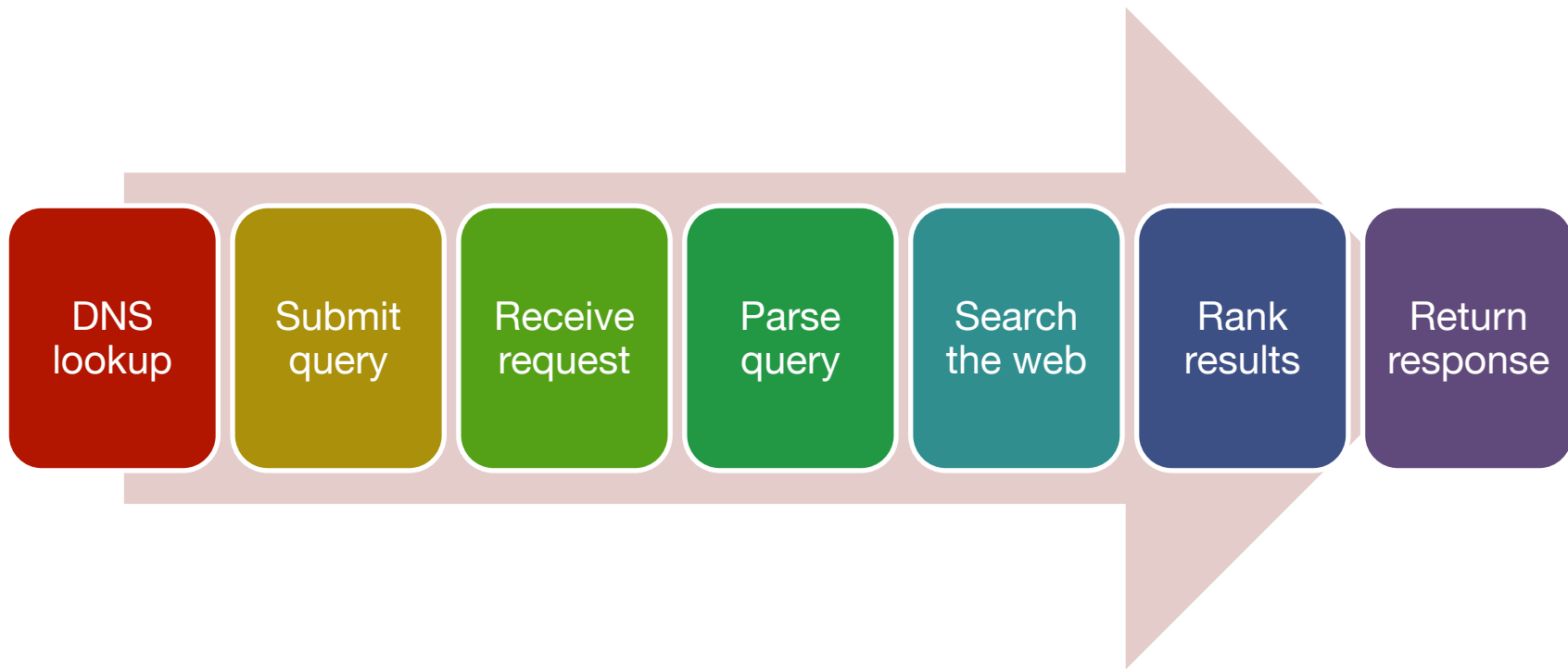
This describes the Google search cluster architecture in the mid 2000s. The search infrastructure was overhauled in 2010.

Nevertheless, the lessons are still valid, and this demonstrates how incredible scalability has been achieved using commodity computers by exploiting parallelism.



# Search flow

What needs to happen when you do a search?



# Some statistics

- 3.5 billion searches/day – trillions per year
- Volume grows ~10% per year – ability to scale is crucial
- 16-20% of searches have never been issued before
  - Caching won't help much
- Average user session < 1 minute
- Hundreds of billions of web pages indexed
  - Index > 100 million gigabytes ( $10^{17}$  bytes)
- 60% of searches are done via a mobile device



Query sizes

# What is needed?

- A single Google search query
  - Reads 10s-100s of terabytes of data
  - Uses tens of billions of CPU cycles
- Environment needs to support tens of thousands of queries per second
- Environment must be
  - Fault tolerant
  - Economical (price-performance ratio matters)
  - Energy efficient (this affects price; **watts per unit of performance** matters)
- Parallelize the workload
  - CPU performance matters less than price/performance ratio

# Best Practices?

**IBM**

IT Infrastructure Servers Storage Software Solutions

## Secure your IT infrastructure with large enterprise servers

Unlock the full potential of your data with the servers designed to meet mission-critical workloads while maintaining privacy across your entire hybrid cloud environment.

[Hear from a customer \(01:59\)](#)

### Meet the IBM large enterprise servers

Ensure that your infrastructure is ready to unlock new business opportunities with high-performance servers that ensure data privacy and security across your hybrid cloud.

- IBM z15™**  
The newest IBM Z® multi-frame system extends next-level data privacy, resiliency and agility to hybrid cloud infrastructures.
- IBM LinuxONE III™**  
Gain the ultimate in uptime and economics, extend data privacy across hybrid cloud, and deliver cloud-native applications faster.
- IBM Power System E950**  
Designed for private and cognitive workload E950 offers a unique blend of enterprise-class capability in a 4-socket 4U form factor.

**Considering a purchase?**  
Chat now with an IBM Sales Representative who can assist you in finding the right products and services to meet your needs.

## Engineered Storage Products

[Zero Data Loss Recovery Appliance](#) [Oracle ZFS Storage Appliance](#) [StorageTek Tape Automation](#)

### Zero Data Loss Recovery Appliance

**Purpose-built by the Oracle Database team as a data-protection extension to the database**

Oracle's Zero Data Loss Recovery Appliance is the only data protection solution designed to eliminate data loss, ensure that backups are recoverable, repair or restore quickly, and easily recover Oracle Databases to any point in time.

[Explore Zero Data Loss Recovery Appliance](#)

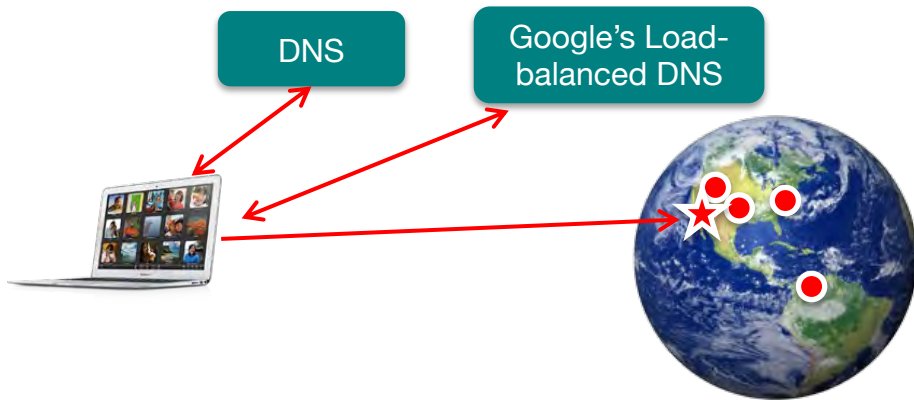
“Enterprise-grade” components

# Key design principles

- Have reliability reside in software, not hardware
  - Use low-cost (unreliable) commodity PCs to build a high-end cluster
  - Replicate services across machines & detect failures
- Design for best total throughput, not peak server response time
  - Response time can be controlled by parallelizing requests
  - Rely on replication: this helps with availability too
- Price/performance ratio more important than peak performance

# Life of a query – step 1: DNS

- User's browser must map *google.com* to an *IP address*
- “google.com” comprises multiple clusters distributed worldwide
  - Each cluster contains thousands of machines
- DNS-based load balancing
  - Select cluster by taking user's geographic & network proximity into account
  - Load balance across clusters

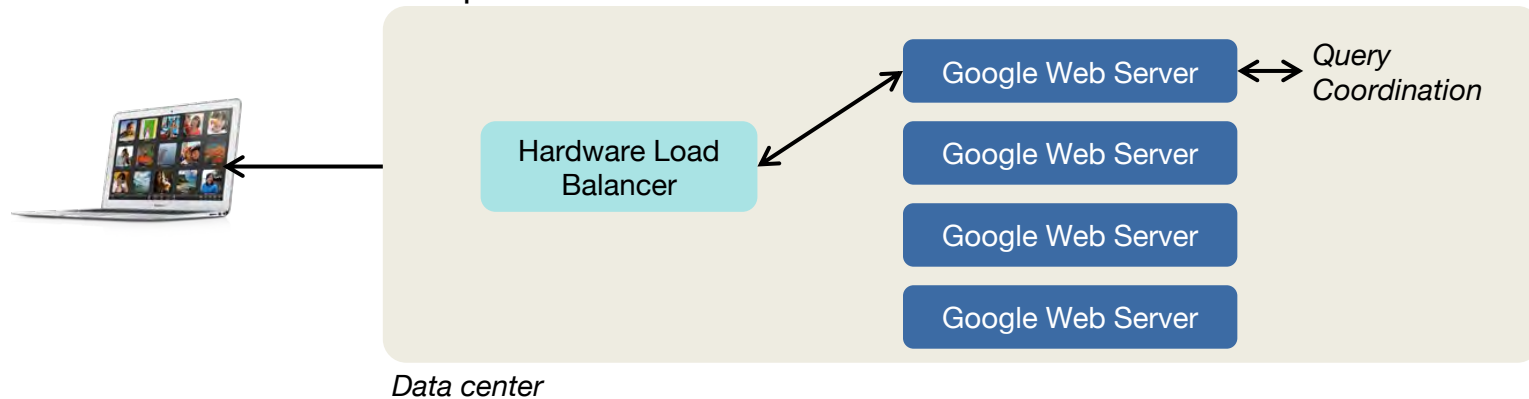


1. Contact DNS server(s) to find the DNS server responsible for google.com
2. Google's DNS server returns addresses based on location of request
3. Contact the appropriate cluster



# Life of a query – step 2: Send HTTP request

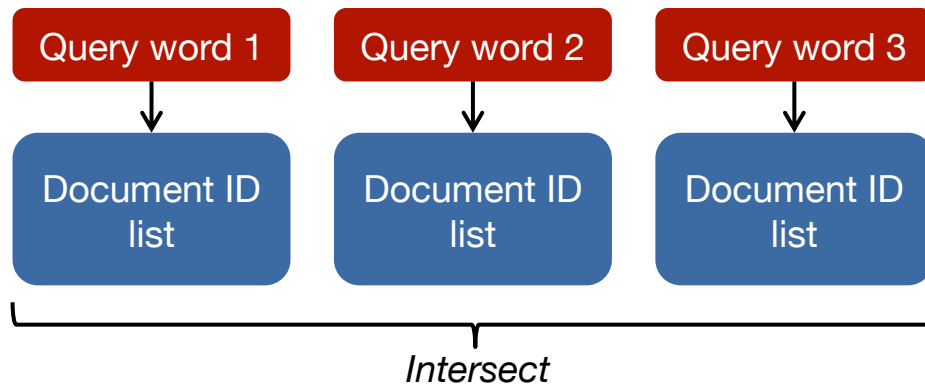
- IP address corresponds to a load balancer within a cluster
- **Load balancer**
  - Monitors the set of **Google Web Servers** (GWS)
  - Performs local load balancing of requests among available servers
- GWS machine receives the query
  - Coordinates the execution of the query
  - Formats results into an HTML response to the user



# Step 3. Find documents via inverted index

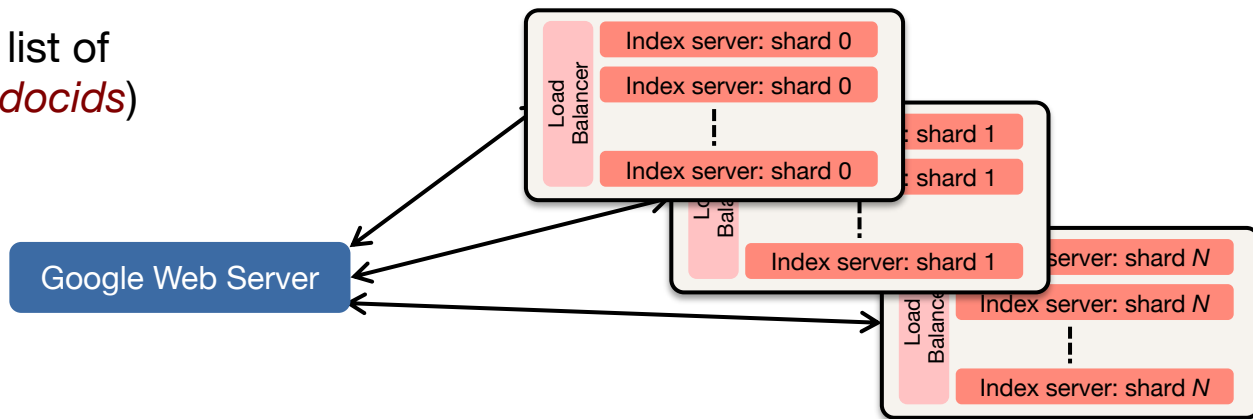
## Index Servers

- Map each query word  $\rightarrow$  {list of document IDs} (this is the **hit list**)
  - Inverted index generated from web crawlers via **MapReduce**
- Intersect the hit lists of each per-word query
  - Compute the relevance score for each document
  - Determine the set of documents
  - Sort by relevance score

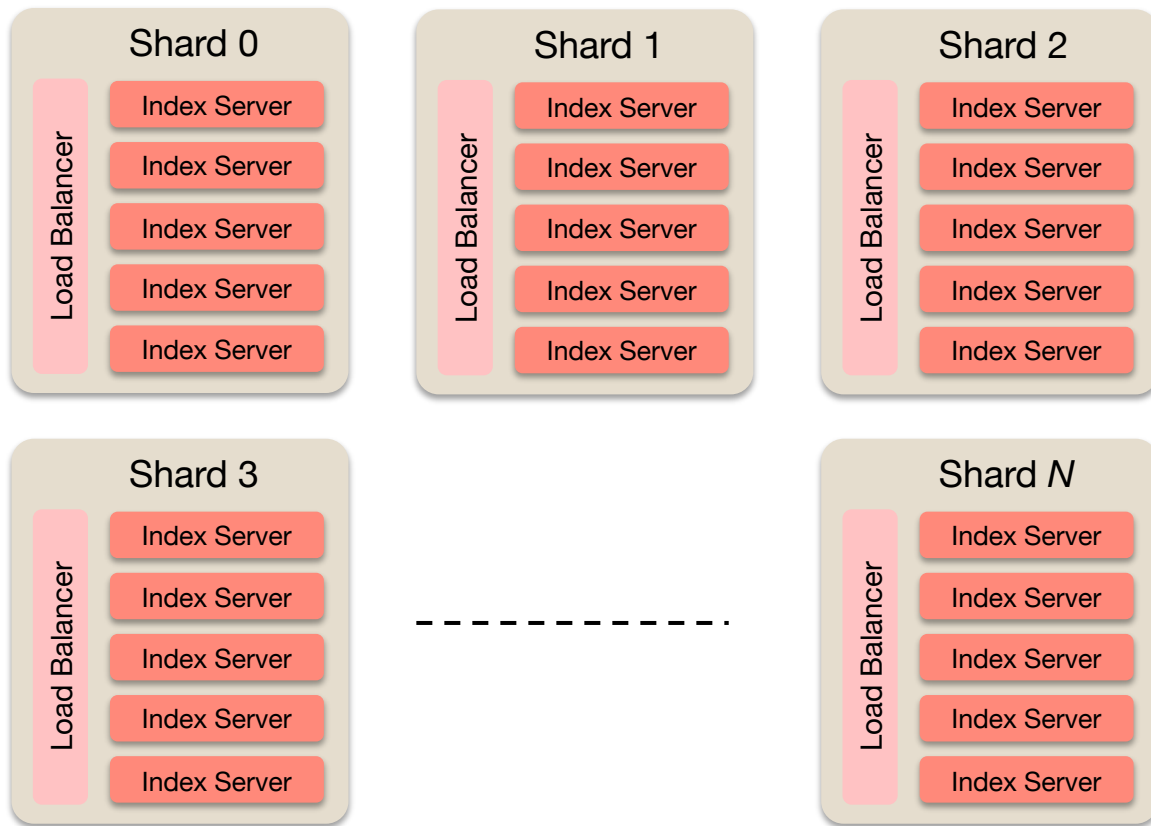


# Parallel search through an inverted index

- Inverted index is 10s of terabytes
- Search is parallelized
  - Index is divided into *index shards*
    - Each index shard is built from a randomly chosen subset of documents
    - Pool of machines serves requests for each shard
    - Pools are load balanced
  - Query goes to one machine per pool responsible for a shard
- Final result is ordered list of *document identifiers (docids)*



# Sharded & Replicated Index Servers



## Step 4. Get title & URL for each docid

For each *docid*, the GWS looks up the *docid* to get

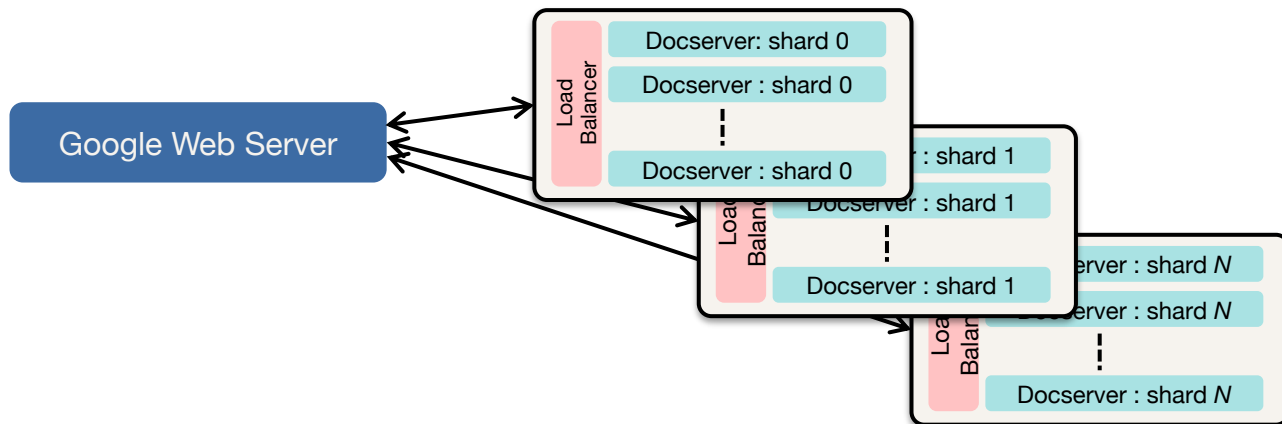
- Page title
- URL
- Relevant text: document summary specific to the query

This is handled by document servers (**docservers**)

# Parallelizing document lookup

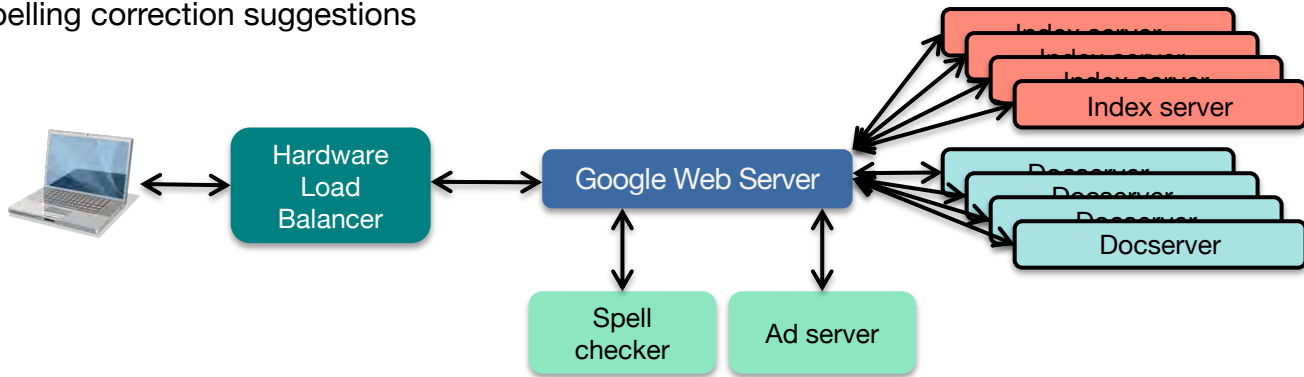
- Like index lookup, document lookup is partitioned & parallelized
- Documents distributed into smaller shards
  - Each shard = subset of documents
- Pool of load-balanced servers responsible for processing each shard

*Together, document servers access a cached copy of the entire web!*



# Additional operations

- In parallel with search:
  - Send query to a spell-checking system
  - Send query to an ad-serving system to generate ads
- When all the results are in, GWS generates HTML output:
  - Sorted query results with
    - Page titles, summaries, and URLs
    - Ads
    - Spelling correction suggestions



# Lesson: exploit parallelism

- Instead of looking up matching documents in a large index
  - Do many lookups for documents in smaller indices
  - Merge results together: merging is simple & inexpensive
- Divide the stream of incoming queries
  - Among geographically-distributed clusters
  - Load balance among query servers within a cluster
- **Achieve linear performance improvement with more machines**
  - Shards don't need to communicate with each other
  - Increase # of shards across more machines to improve performance



# Updating & scaling are easy

## Updates

- Updates infrequent compared to reads
- Load balancers make updating easy
  - Take the system out of the load balancer during the update
  - No need to worry about data integrity and locks
- Shards don't need to communicate with each other

## Scaling

- Add more shards as # of documents grows
- Add more replicas if a throughput increase is needed

# Summary

- Use software to achieve reliability
- Use replication for high throughput
- Price-performance is more important than peak CPU
- Use commodity hardware

The End

The End