OVERVIEW OF GOOGLE SEARCH ENGINE

Xiannong Meng
Computer Science Department
Bucknell University
Lewisburg, PA 17837
U.S.A.

Hardware Architecture

- Very little is published on specific search engines
- Google clusters (early 2003 data) (Barroso *et.al.*)
  - 15,000 Off-the-shelf PCs
  - Ranging from single-processor 533-MHz Celeron to dual-processor 1.4 GHz PIII
  - One or more 80G IDE drives on each PC
  - Giga-bit switches
  - 2-3 years of life cycle (hardware)
  - Fault-tolerant software

Some Highlights

- On average, a single query reads hundreds of megabytes of data and consumes billions of CPU cycles
- Thousands of queries per second at peek time
- Index structure is partitioned
- Different queries can run on different processors

Design Considerations

- Use software to provide functionality, rather than using hardware
- Component-off-the-shelf (COTS) approach: commodity PCs are used with faulty tolerant back-bone network
- Reliability is achieved through replicating services across many different machines
- Price/performance consideration beats peek performance

Serving a Query

- When a query is received, it is mapped to a local google cluster that is close to the user geographically
- Google has clusters distributed across the world
- Each cluster has a few thousand machines
- A DNS-based hardware load-balancing scheme is used

Serving a Query (cont)

- Query execution consists of two major phases
- Phase one:
  - The index servers consult an inverted index that map each query word to a hit list
  - The index servers then determine a set of relevant documents by intersecting the hit lists of each query words
  - A relevance score is computed for each document
  - The result of this phase is an ordered list of docIds
  - The entire index is randomly divided into pieces (*index shards*)

Serving a Query (cont)

- Phase two:
  - The document servers take the docIds and compute the actual title and URL for each, along with a summary
  - Documents are randomly distributed into smaller shards
  - Multiple servers replicas responsible for handling each shard
  - Routing requests through a load balancer

Document Server Clusters

- Each must have access to an online, low-latency copy of the entire web
- Google stores dozens of copies of the web across its clusters
- Supporting services of a google web server (GWS) besides doc server and index server:
  - **Spell check**
  - **Ad serving (if any)**

Commodity Parts

- Google's racks consist of 40 to 80 x86-based servers mounted on either side of a custom made rack
- Each side of the rack contains 20 2-u or 40 1-u servers
- These servers are mid-range
- Several generations of CPU in active use, ranging from 533-MHz single processor to dual 1.4-GHz Pentium III servers

Commodity Parts (cont)

- Each server contains one or more IDE 80 G byte disk drive
- The servers on each side of a rack interconnect via a 100-Mbps Ethernet switch
- Each switch has one or two gigabit uplinks to a core gigabit switch that connect all racks together

Cost Estimate of an Example Rack

- Late 2002, a rack of 88 dual-CPU 2-GHz Xeon servers with 2 G bytes Ram and 80 G bytes of disk costs around $278,000
- This rack contains 176 2-GHz Xeon CPUs, 176 G bytes of RAM, 7 T bytes of disk
- A typical x86-based server contains 8 2-GHz Xeon CPUs, 64 G bytes of RAM, 8 T bytes of disk, costing about $758,000

Google File System

- **Google file system (GFS,Ghemawat 2003)**
  - **64-bit distributed file system**
  - **Single *master*, multiple *chunkservers***
  - **Chunk size 64 MB**
  - **The largest GFS cluster has over 1000 storage nodes and over 300 TB disk storage space**

Issues with File Systems

- Component failures are the norm rather than the exception
- Files are huge in google by traditional standards
- Most files are mutated by appending new data rather than overwriting existing data
- Co-design (GFS) with other applications makes it more efficient

Assumptions

- Built from many inexpensive commodity components that often fail
- Store a modest number of large files, on the order of a few million files, 100 MB or large for each
- Workload: large streaming reads and small random reads, e.g. streaming reads involve 100 KB or 1 MB per read

Assumptions (cont)

- Workloads may also have many large, sequential writes to append files
- Once written, revisions are rare
- Multiple clients, concurrent access (append)
- High sustained bandwidth is more important than low latency

Architecture

- A single master cluster
- Multiple chunk-servers
- Each accessed by multiple clients
- Files are divided into fixed-size chunks (64 Mbytes)
- Each chunk is identified by a unique 64-bit chunk-handle
- Chunk-servers store chunks on local disks as linux files

Metadata

- The master stores three major types of metadata:
  - File and chunk namespaces
  - Mapping from files to chunks
  - Locations of each chunk's replicas
- All metadata have a copy in memory
- Namespaces and file-to-chunk mapping are also kept persistent storage (local disk and remote replica)

Consistent Model

- **What is a consistent model? – How Google (or anyone) guarantees the integrity of the data**
- **Guarantee by GFE**
  - **File namespace mutations are atomic**
  - **State of a file region after a data mutation depends on the type of operations, success or failure, and whether or not concurrent updates occured**

Consistent Model (cont)

- A file region is consistent if all clients will always see the same data
- GFS guarantees the consistency by
  - Applying mutations to a chunk in the same order on all its replicas
  - Using chunk version number to detect any replica that has become stale because it has missed mutations while its chunkserver was down

System Interactions
--- Lease

- **Lease: when each mutation is performed, the master grants a chunk least to one of the replicas, *primary***
- **The primary picks up a serial order for all mutations to the chunk**
- **A lease has an initial amount of timeout of 60 seconds**
- **As long as the chunk is being updated, the primary can request and will get extension**

System Interactions
--- Data Flow

- Data is pushed linearly along a chain of chunkservers
- Each machine's full outbound bandwidth is used to transfer data
- Each machine forwards the data to the "closest" machine that has not received the data
- Data transfer is "pipelined" over TCP connection – as soon as a chunkserver starts receiving data, it forwards the data immediately

System Interactions
--- Atomic Record Appends

- Clients only specify the data
- GFS appends the data the file at least once atomically
- Returns the location to the client
- If the append would cause the chunk to exceed its size limit (64M), the current chunk is padded, a new chunk starts with the record in operation
- If an append fails at any replica, the client retries the operation

System Interactions
--- Snapshot

- The snapshot operation makes a copy of a file or a directory tree
- Copy-on-write is used to implement snapshots
- When a request is received, the master revokes any outstanding leases, ensuring data integrity
- Subsequent write would have to interact with the master again

Fault Tolerance

- **High availability**
  - **Both master and chunkservers restart in seconds**
  - **Each chunk is replicated on multiple chunkservers on different racks**
  - **A read-only “shadow” master replica, may lag the primary slightly**

Fault Tolerance (cont)

- **Data integrity**
  - A chunk is broken into 64 K blocks, each of which has a 32 bit checksum
  - Upon a read request, if checksum fails, the reader may read from another chunk replica
  - Checksum computation is heavily optimized for appending operation

Fault Tolerance (cont)

- Diagnostic tools
  - Extensive and detailed logs are kept to help in problem isolation, debugging, performance analysis

Overall System Architecture

- **Google overall architecture based on their 1998 paper, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Sergey Brin and Lawrance Page, 1998 WWW conference, Brisbane, Australia, April 14-28 1998**
  - **http://citeseer.ist.psu.edu/brin98anatomy.html**

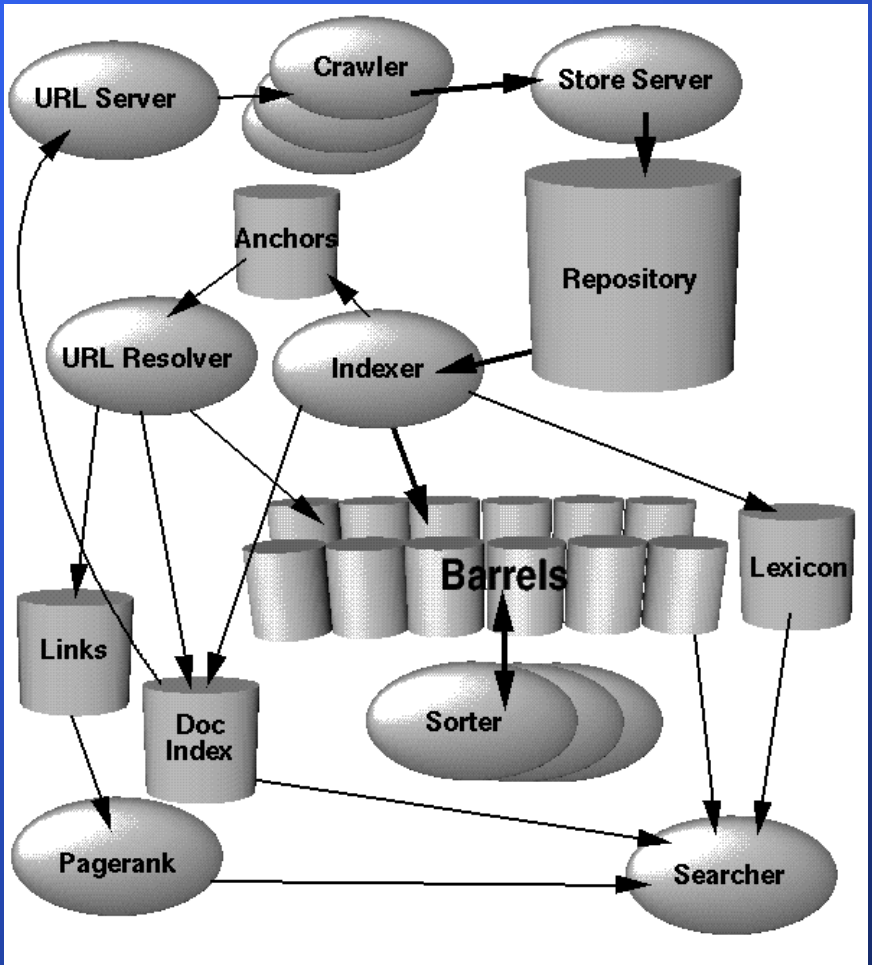
Some Basics

- Google: a common spelling of 10^{100}
- Issues to address: accurate, high quality search results
- Quick history:
  - 1994 WWW Worm had an index of 110,000 web pages
  - 1998 AltaVista had about 140 million pages indexed
  - About 20 million queries per day in November of 1997 to AltaVista

Google Features

- **PageRank:**
  - **use the frequency of citations to a page and the citations the page used to point to other pages to measure importance**
- **Anchor text:**
  - **Provide more accurate description than the page itself**
  - **Exist for documents that cannot be indexed by a text-based search engine (e.g. image, db)**

Google Architecture



Components of Google

- Distributed web crawling
- URLserver sends lists of URLs to be fetched to the crawlers
- The web pages fetched are sent to the Storeserver which compresses and stores them
- A unique docID is assigned for each page
- The indexing function is performed by the indexer and the sorter

Components of Google (cont)

- **Indexer:**
  - Each doc is converted into a set of word occurrences called *hits*
  - The hits record the word, its position in the doc, font size, and capitalization
  - The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index
  - The indexer also parses out all the links in every web page and stores important information about them (points to and from, text of the link) in an anchor file

Components of Google (cont)

- The URLresolver reads the anchors file and converts relative URLs into absolute URLs, and docIDs
  - Put the anchor text into the forward index, associated with the docID
  - Generate a database of links which are pairs of docIDs (the host page, and the page pointed by this host page)
  - The links database is used to compute PageRanks for all documents

Major Data Structures

- BigFiles: virtual files spanning multiple file systems, addressed in 64 bit integers
- Repository: full HTML of every web page (then) current size: 150 G
- Document Index: information about each doc, it is a fixed width ISAM (index sequential access mode) index, ordered by docID
- Hit list: occurrences of a particular word in a particular document including position, font, capitalization
- Forward index: from doc to words, in a number of barrels
- Inverted index: from word to doc

Crawling the Web

- Multiple crawler (typically three)
- Each crawler keeps roughly 300 connections open at once
- At peak speed, crawl over 100 web pages per second with four crawlers (about 600 K per second)
- A major performance stress is DNS look-up, each crawler keeps its own DNS cache to improve performance

References

- Luiz Andre Barroso, Jeffrey Dean, and Urs Holzle, “Web Search For A Planet: The Google Cluster Architecture”, *IEEE Micro*, March-April 2003, pp. 22-28
- Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google File System”, *Proceedings of SOSP’03*, pp. 29-43
- Sergey Brin and Lawrence Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, *Proceedings of the Seventh World Wide Web Conference*, 1998, pp. 107-117