NoSQL

• Motivation:
  • Elastic (can change the deployment size) and scalable
  • Flexible schema

• Need to give up on typical RDBMS features, such as
  • Complex queries like joins
  • Transactions involving multiple objects
  • ACID
    • Instead, B(asically) A(vailable) S(oft state) E(ventually consistent)

https://courses.cs.washington.edu/courses/cse444/16sp
ACID vs BASE

• Three properties of a distributed system with replication
  • **Consistency**: every read gets most recent write or error *(different from RDBMS definition of consistency!)*
  • **Availability**: every request gets *some* response
  • **Partition tolerance**: system operates even when network messages getting dropped or servers go down

• **CAP theorem**: in the presence of a network partition, you can only have consistency or availability

• **ACID** (e.g., conventional RDBMS) pick C
• **BASE** systems (e.g., NoSQL systems) pick A

Example: Amazon’s Dynamo

- Highly available *key-value store*
- Designed to host the Amazon retail website
- Observation: many services just needed **primary-key access**

- Data model (key, value)
- Simple operations: insert/delete/lookup by key
- Data replication with eventual consistency

https://courses.cs.washington.edu/courses/cse444/16sp
Summary

• *Scale out* and replicate to serve read-only queries
• *Scale out* and shard to serve transactions that touch limited data (e.g., that of a single user)
• When transactions touch data on different servers (e.g., that of two users), need to provide distributed transactions. ACID is expensive.
• “NoSQL” systems give up on something to scale
CS4400: Database Systems
Parallel query evaluation

Brandon Myers
11/29/2016
Scalability scenario #4

- *MedReport* is doing so well that you hire an analyst whose job is to use the crowdsourced data to discover previously unknown side effects for every medication.

- They collect all the user reports to one server and write a SQL query to answer the question. Unfortunately, they find it will take 5 weeks to run. By then the data will be out of date.

- What do you do?
Parallelism for a single query

Think of motivation in one of two equivalent ways

a) Lower latency to run the query on a whole dataset
b) Higher throughput in terms of records or documents processed per second
Example: analyzing messages from newsgroups

newsgroup sci.space

1000 files, each containing the text of one thread
(not huge but let’s pretend it is)
Q1. Find documents whose subject contains the words “space station”
Q2. For each document with at least one “NASA”, count the occurrences of “NASA”

3 workers processing the data in parallel
countByKey worked fine without requiring any communication between the partitions

Is that always the case?
Q3. Count the number of documents containing the given word, for the words NASA, station, and satellite
distinct

Map swap key & value

countByKey
Send tuple \((K, V)\) to partition \(h(K)\) (i.e., shuffle)

Sum the counts for each key

\[\begin{align*}
\text{countByKey} & \quad (NASA, \text{sc.i.space/6000}) \\
& \quad (NASA, 1) \\
& \quad (NASA, 1) \\
& \quad (NASA, 3) \\
\text{countByKey} & \quad (NASA, \text{sc.i.space/60201}) \\
& \quad (NASA, 2) \\
& \quad (satellite, 1) \\
& \quad (satellite, 1) \\
& \quad (station, 1) \\
\text{countByKey} & \quad (station, \text{sc.i.space/60201}) \\
& \quad (station, 1) \\
\text{countByKey} & \quad (NASA, \text{sc.i.space/6040}) \\
& \quad (station, 1) \\
& \quad (station, 1) \\
\text{countByKey} & \quad (satellite, \text{sc.i.space/6060}) \\
& \quad (satellite, 1) \\
& \quad (satellite, 1) \\
\end{align*}\]

\(h\) is a hash function