DATABASE

LECTURE-9

A Benchmark to Evaluate Interactive Social Networking Actions.

BG contributions

Design, abstraction & implementation of a novel benchmark (BG) that can evaluate different classes of data stores used to manage social graphs.

- Resembles a social networking system.
- Generate meaningful requests.
- Produces repeatable & explainable results, meaning that if you run this BG over & over, you should see the same results coming out of it.
- Runs in a reasonable amount of time.

Used extensively to explore the behavior of alternative design decisions. Quantitative analysis & comparison of different social networks & demonstration of their trade-offs.

Benchmark helps you with comparison between 2 systems (Dynamo VS Foundation)

How are data store vendors to identify inefficiencies in their systems & try to improve it?

Benchmark

It plays an important role in the application developer's decision to utilize a data store.

— Compare various data stores with one another.

Facebook, will look at their workload, the requirements & what a data store has to offer & how well they match one another. It'll be pruned down to a collection of data stores.

— Identify tradeoffs.

They're going to start to want to compare these data stores with one another to understand the trade offs associated with those data stores.

— Understand how suitable a solution is.

Understand how suitable the solution is for their workload. How well does it take care of the customers?

Database Management System

→ As you insert data into it, if it runs out of storage, it brings up an error message.

→ when retrieving the data, if it says, it doesn't have it, it mean you never gave it to it.

data

Cache management system (memcached)

→ If you insert data into it such that its storage is exhausted, it stays quiet.

→ It kicks a data item out & replaces it with the data you just gave it.

If it does have the data then, it evicted it. It got rid of the data because the storage was full.

Motivation

— Be aware of the functionalities you want.

→ Is it caching?

→ Is it database?

enchance the performance of the system.

— Why do systems perform as they do?

— Which component of a data store result in it becoming the bottleneck?

If you know the bottleneck resource is, say CPU, then you should be adding CPU & not disk drivers, Benchmark helps identify the components that are becoming bottlenecks

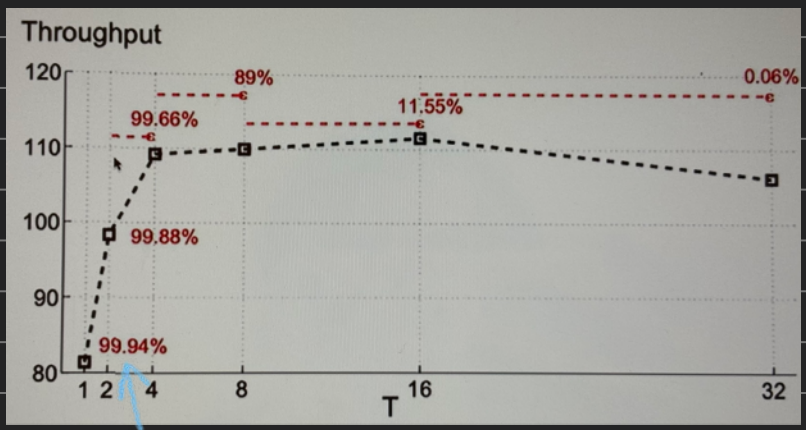
— How accurate is the information being offered by the vendors of data stores?

A benchmark is a good sanity check that lets you evaluate the performance of the system to verify, vendors claim.

— How do we compare one data store with another in a social networking application?

Throughput & Response time.

The relation between system response time & throughput is complex.



percentage of requests processed faster than 100 milliseconds. (typical human threshold.)

Why is the response time plummeting?

QUEUES

Because of queues, there is delay & as a monolithic commercial database Management System, it has finite amount of resources.

CPU DISK MEMORY

Monolithic Systems are multithreaded; multiple threads to process different requests concurrently.

You need Semaphores & latches to protect critical data structures.

↑ eg: Buffer Managers.

These threads come acquire the semaphores & then they block one another; which in turn results in the queuing delays that causes the response time to go down the tube.

Slowest section of the software becomes bottleneck; eg: If buffer pool is implemented poorly all the thread queue on that buffer pool because it's the slowest piece of code.

What does NoSQL mean?

→ Forget 1st normal form

→ NoSQL front end

→ Scale

→ Replicate & distribute data across many servers.

→ Uses Basically Available, Soft State, Eventually consistent. BASE.

→ Ability to dynamically add new attribute to data records.

CAP Theorem

— Consistency:

A read request observes the most recent write or an error.

— Availability:

A request observes a non-error response, even when node fails.

— Partition Tolerance:

The system continues to operate despite the nodes not being able to communicate with one another.

eg: node failures, loss of an arbitary number of messages between nodes for an arbitrary duration of time.

If you have failure in your system:
you must choose between consistency & availability.
And if you want to have consistency & availability you cannot have failure in your system.

CAP Example.

Consisting of 2 nodes → primary
→ Secondary

→ Every time primary executes a write transaction it forwards it to the secondary & waits for the secondary to apply.

→ If the secondary node fails & primary detects it; it's going to continue to process the writes & advances the LSN, while secondary was done.

→ At some point when the primary also fails, the service is now unavailable, both the primary & secondary are down.

→ When the secondary comes back up & running but it's LSN doesn't have the lastest.

Network partition

— Here, primary & secondary are available but the synchronous replication cannot go over the network, both are isolated & looks as though secondary is unavailable.

Eventually Consistent Reads.

Eventually consistent is the default read consistent model for all read operations. When issuing eventually consistent reads to a DynamoDB table or an index, the responses may not reflect the results of a recently completed write operation. If you repeat your read request after a short time, the response should eventually return the more recent item. Eventually consistent reads are supported on tables, local secondary indexes, and global secondary indexes. Also note that all reads from a DynamoDB stream are also eventually consistent.

look at the system → look at the requirements;

If you need the concept of transaction, you need to pay more for the consistent reads, otherwise, if you go for eventual reads, its not going to support the transactions & the consistency requirements needed.

☆ Response time.

A percentile indicates the response time below which a given percentage of response time falls.

95th percentile = <95%

Online Request: Fast Response Time.

Processed by the system as soon as it is issued.

eg: Interactive users requests.

Offline Requests: Favors throughput.

Buffered & processed at a later time.

eg: write of information / cookie-settings.

BG Benchmark

- Compare & contrast different data stores.
- Horizontal & vertical scalability behavior of data stores.

It is a standard by which something can be measured or judged. It quantifies measures of performance. Identify gaps & inefficiencies in the system for improvement.

Designed to mimic a particular type of workload on a component / system.

— Synthetic workload:

Used for stress testing individual components.

— Application workload:

Resembles the real world application workload. Gives a better measure of how the system will work in real world.

Challenges of developing a Benchmark.

— Focus on metrics

Are you targeting the response time / throughput.

— Applicable to a broad spectrum of hardware/architecture

It shouldn't care whether the database management system is disaggregated or it's monolithic.

— Gather accurate data.

— Benchmark cannot be the bottleneck.

When you come up with a Benchmark, make sure you capture the actions that the Benchmark should be generating.

BG Architecture

At the bottom you could have the data store servers. eg: DynamoDB, Oracle.

The Benchmark is scalable because it has multiple BG clients that can issue requests against the data stores. There could be potentially 'n' nodes that are issuing requests againts the underlying database management system.

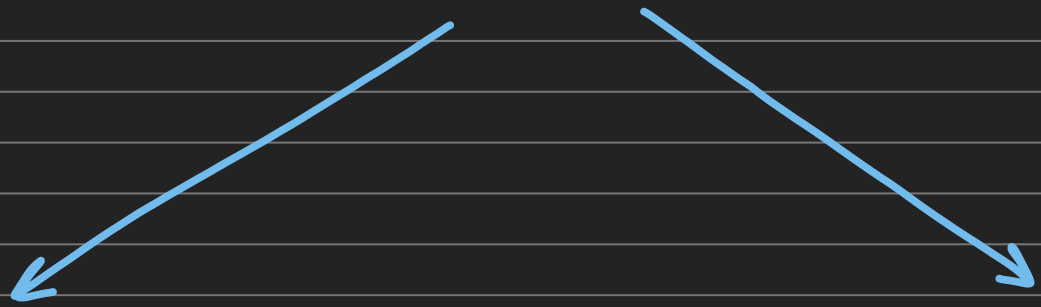
At the very top we have a specification of the workload that emulates the user's behaviors, including a specification of a service level agreement.(SLA).

no. of concurrent requests at a certain response time.

The user also has a visualization tool to see the execution of the benchmark. The benchmark has a coordinator that initiates the clients & monitors how much unpredictable data is produced by the underlying DBMS.

There is a data analyzer that gathers the results from the client & it adjusts the duration of the experiments that are running, for the rating of the underlying data store.

BG METRICS



Measured Metrics

- Throughput.
- Response Time & confidence
- Amount of unpredictable data
- Freshness confidence for a fixed duration.

Rating metrics

- SoAR ← not Similar
- Socialites

* SoAR (Social Action Rating)

Highest number of completed actions per second satisfying an SLA requirement, thats where the throughput comes into the picture satisfying that SLA.

Socialites

Highest number of simultaneous threads or user requests that satisfy a given SLA requirement.

These are the 2 metrics to describe the behavior of a system given an SLA.

BG characteristics.

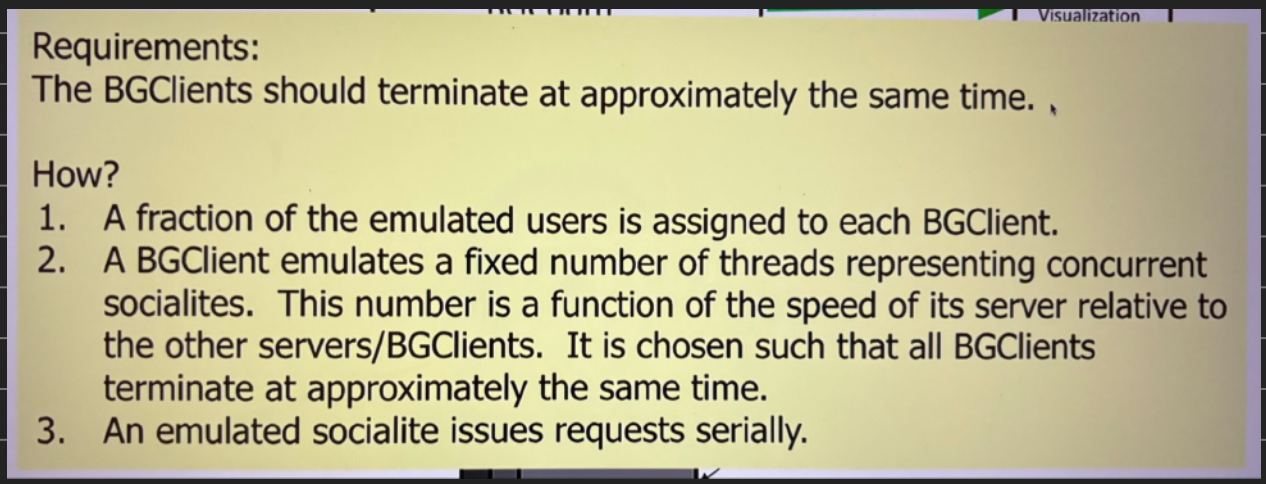
It's trying to emulate users with unique login sessions & different activity levels. And then it emulates the user behavior issuing social actions. Social session is a sequence of actions for the same users. It has the concept of think time & arrival time. Theres also a conceptual schema pertaining to a social network.

Since conceptual schema is high level description & that's why BG is netural to a DBMS.

BG Software Architecture.

It has a schema creation, tell it to create the database schema & write that piece of code for it. These are interfaces to be implemented. The load popul-

ates the database with data & the benchmark initiates a workload & measure the performance metrics such as response time, throughput. The system is designed to be extensible, so you can come up with new social actions.


Visualization

Requirements:
The BGClients should terminate at approximately the same time.

How?
1. A fraction of the emulated users is assigned to each BGClient.
2. A BGClient emulates a fixed number of threads representing concurrent socialites. This number is a function of the speed of its server relative to the other servers/BGClients. It is chosen such that all BGClients terminate at approximately the same time.
3. An emulated socialite issues requests serially.

One BG client is not sufficient & the coordinator will flag it & inform that your client is a bottleneck & it can do so is because it's getting information such as the CPU utilization or the network utilization.

* Unpredictable Data: Validation Phase.

- Is either stale, inconsistent or simply invalid data produced by DBMS.
- BG is aware of the initial state of data.
- BG issues read & write actions to the DBMS & hence knows the time stamp of the read & write actions.

- It comes up with all the possible ways of serializing it & if it cannot then flags it as unpredictable data.

How does BG rate a data store?

- The throughput of the system is either a square root of a function or a concave function of the no. of threads that are issuing requests to the system.
- The response time, the percentile is assuming that the response time increases beyond a certain number of threads.

Quick Rating.

→ Heuristic: increment the number of threads by one.

→ Two Agile data loading: loading the data store between the rounds of experiment with workloads that include writes.