Building a database on S3? That is the title of the article.

Odd M. Trondrud

November 2013

Abstract

A report about the article "Building a database on S3." It presents a brief summary of the article and briefly discusses security and client management concerns with the implementation proposed by the article. Reasons for why one might to do such a thing as build a database on top of S3 are discussed at the end.

Introduction

"Imagine, if you will, a worry-free world."

Traditional system administration involved maintaining the system's hardware as well as everything else. Some enjoy this while others would rather not crawl around in tight spaces and tinker with something physical in addition to their other duties. And assystems grow more complex and fields expand outwards, being a jack-of-all trades becomes an unsustainable venture for the individual. Will society progress towards a future in which each individual must specialize so thoroughly in one field that their usefulness in all other fields diminish to nothing? Who knows.

Either way, service providers (like those guys who started selling books on the internet) have figured out how to turn a profit by providing certain services at a sufficiently large scale. Take their Simple Storage System, for example: A near infinitely scalable and concurrent file storage service (for the consumers) that is discounted if the availability drops below 99.9%[2]. Such a service is typically referred to as a "cloud" or "utility" service. Possibly because there's always a cloud somewhere on this planet and iif all clouds were connected to the internet and stored the same information with a promise of *eventual concurrency* you'd be able to reach your information from wherever.

If one were to maintain such a system one would require a minimum of the maximum expected storage space for the system's data, and then enough extra storage for backups and mirroring in case of disk failures. One would have to pay for an internet connection capable of handling a way higher load than what's expected. And people would have to be kept on payroll to maintain the physical parts of the system as well as the non-physical parts like the software. To summarise it is a load of hassle.

However there exists *utility service providers* that provide for their customers such a service and run a by-use payment model. In other words, the customers' infrastructure becomes near infintely scalable and concurrent without the customer having to pay the overhead such an infrastructure would impose on them if they were to develop and maintain it themselves. Ah, the future! Isn't it great?

The article, "Building a database on S3", lays down an outline of how one could build a near infinitely scalable database system on top of Amazon's Simple Storage Service (S3). The proposed system exploits the high level of availability Amazon offers with its S3[1].

The Proposed Implementation

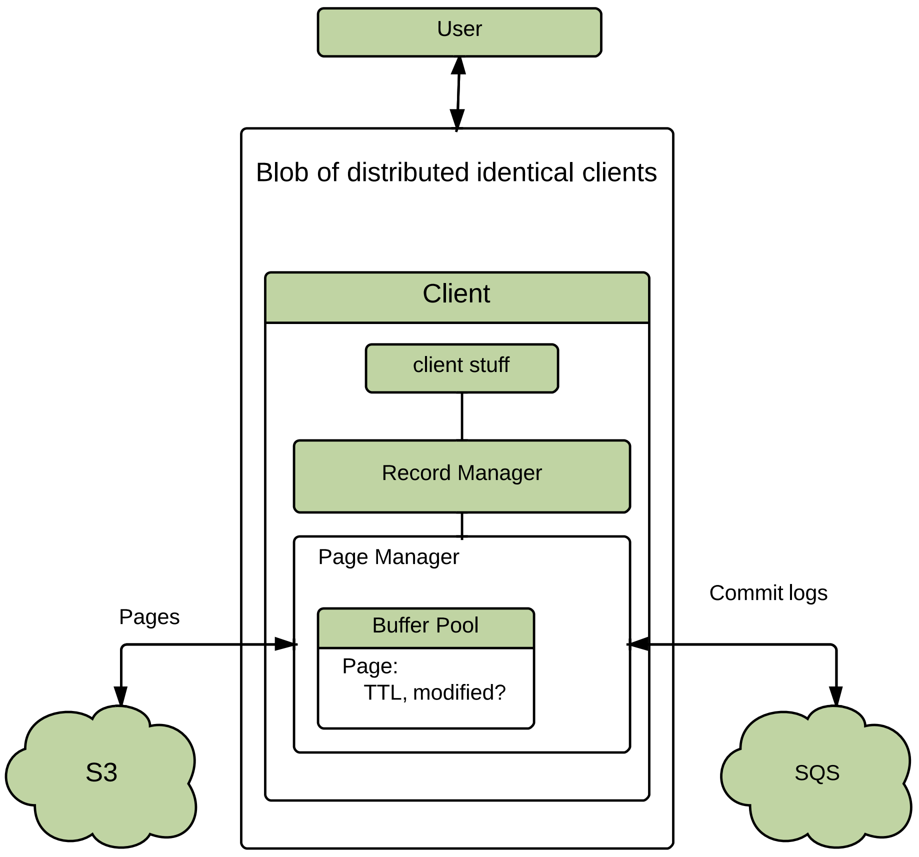

Figure 1: An overview of the suggested architecture in the article.

Scalability and bottlenecks

While the article claims that the proposed implementation would be able to support a very large number of concurrent clients, effectively eliminating the DBMS as a potential bottleneck, it does not discuss how the number of clients could be managed. An ideal scalable system should scale itself automatically, the number of active clients at any given time, growing and shrinking with the number of active users. A "master" client that monitors traffic and load on the system and spawns and terminates clients as needed could be introduced, however this could result in the master client becoming a new bottleneck. Alternatively each client could be allowed to spawn new clients whenever the load passes over some threshold and diverting traffic to other clients (new clients could be given a copy of the spawning client's buffer pool to reduce

[1]99.9% uptime as of November 2013

the amount of requests to S3). Clients could terminate themselves when the load passes below some threshold, however if there is only one active client it should not be allowed to terminate itself. This introduces the question of how traffic should be diverted between clients, which depends on the application's nature. Research has probably been done on this and could be worth looking into.

Security

The article barely touches on security concerns related to such a truly distributed system which it proposes. It states that read and/or write privileges to a collection (i.e. a *bucket* in S3 terminology) can be restricted on a per-client basis by the client that owns the collection, and that a security infrastructure *can* be implemented on top of S3 however it does not present an explanation of *how*. This is somewhat understandable as the definition of "client" provided by the article[2] is rather vague and implementing a proper security infrastructure for a system one knows very little about can be difficult.

However, guesstimation is possible and there are certain general security-related issues that arise when dealing with distributed systems.

Data Encryption

If your data is proper sensitive like you probably shouldn't store it on a service you don't control. Although Amazon offers server-side encryption of the data stored on S3, recent revelations regarding certain companies willingness to comply with certain intelligence agencies' requests for access to their data[1] should serve as deterrent for everyone from trusting anyone but themselves. In other words, client side encryption of the data would be adviseable. This would increase the complexity of the implementation, as every client should be able to decrypt and encrypt the data independently.

Client Authentication

To avoid (or at least reduce the risk of) an attacker snooping up the URIs of pages and buckets and writing junk data to these, the system should require some sort of client authentication. Amazon allows you to create users on an account that can be granted access to specific buckets or objects[3]. One possible authentication scheme could be that each client corresponds to a user on the S3 account. However this would require that users be created on the fly as new clients are spawned. In addition, this solution would require that the clients are somehow informed of their user credentials when spawned. If it is not possible to create and delete users through the S3 API then the maximum number of active clients would be limited by the number of available users on the account. Some type of special client that the other clients request credentials from could be introduced. The request would have to be verified by the system somehow, to prevent attackers from obtaining credentials by simply requesting them. For example, the credential granting client could require that the request be signed by the client that spawned the new client. Alternatively the job of actually spawning new clients could be delegated to the credential granting client. Both of these solutions introduces a potential bottleneck in the special client.

[2]"

...

refer to software artifacts that retrieve pages from S3 and write pages back to S3."

Summary

Would one want to build a database on S3? Well, there's nothing wrong with that kind of scalability but it depends on what the bottleneck of your web service is going to be. If your service, for some reason, is incompatible with any sort of caching or if it's built around users somehow submitting information to be stored in the database then... maybe? One should price shop: look at alternatives (to both infrastructure and the service's structure).

When developing a service one should try and think about such things as "what is the maximum amount of users this service needs to be able to serve at once?" and "what kind of traffic/activity is the average user going to generate?" and "how high does the service's availability need to be?". There'll probably be questions or cases that are more specific to particular services, but unless the answer to these three questions are "infinity", "all of it" and "$99.\bar{9}\%$" then the service probably does not need all that "building a database on S3" (or similar solutions, like SimpleDB) offers. However it does help, but there's probably alternatives out there to building a database on S3 and *you should try and look for them.*

References

[1] 2013 global surveillance disclosures, wikipedia. `http://en.wikipedia.org/wiki/2013_global_surveillance_disclosures`.

[2] Amazon s3 service level agreement. `http://aws.amazon.com/s3-sla/`.

[3] Amazon s3 access control. `http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingAuthAccess.html`.