# Systems Architecture Question

REST

Web Server

REST

Web Server

OAUTH

Server

SQL Database

The above diagram is of a simple OAUTH wrapped REST style web service that gives access to a SQL Database. The Database may have queries and operations performed against it from other in systems in the business directly (i.e. batch and data loads). This is currently a small Colo based service provided to external fee paying clients with associated SLA’s, its effectively version 1.0 of the service that was thrown up in a hurry.

The REST endpoints have the following characteristics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Endpoint Group | Distribution | Write to DB? | Results can be Cached? | Complexity |
| A | 10% | Yes | No | Low |
| B | 20% | No | No | High |
| C | 20% | No | Yes | High |
| D | 50% | No | Yes | Low |

Complexity refers to the processing or query overhead incurred enacting the given endpoint. If the results can be cached, it’s okay for the results to be stale.

*The above system will encounter a number of requirements changes given below in order, for each describe the technical transformation you would logically apply to the prior iteration and why what instance(s) of real world technology you would use to actually implement said transformation, do not ‘skip’ in the evolution of the system. Credit will be given for mentioning appropriate algorithmic and architectural patterns.*

**You can assume that cost and availability of technical & engineering resources will always be a factor to consider; i.e. the costlier and more complex the solution is to implement the less likely or harder it will be to get approved – in short, you need to pick your battles.**

## Step#1 – 10x increase in endpoint usage

Sales have signed up a new client (XYZ Inc) who will increase equally across all endpoints the loading on the system 10 fold, this is greater than the current system can cope with. You have heard down the grapevine that Sales have two other similar sized clients in the pipeline…

*As the story tells, the required additional resources are simply not available in current system, and take into consideration that another two similarly sized clients are also in the pipeline, upgrade of current system resource is not avoidable.*

*To upgrade the current system that would need to grow at such a fast pace, a*

*scalable solution is desirable.*

*The monitoring statistics of the endpoint group shows that, loading on different group are different, the consumption pattern of these groups are also possible to change over time. Seperating these endpoint into different deployment environments should be beneficial, doing so would enable us to distribute the resource according to the demand change.*

*Possible enhancements:*

*Hosting planning:*

*Transition from current Colo based hosting to Could based hosting, such as Google Cloud*

*Load balancing planning on following:*

*Network bandwidth*

*Procecssing units*

Data I/O

## Step#2 – Those 2 new clients signed up

The 2 new clients (Blah Inc and Wazoo Holdings) did indeed get signed up, although both seem to be intent on using the endpoints with 85% in the D endpoint group with all the rest proportionally reduced. Accounts are complaining about the cost of the system, they just saw the bill and they want your blood in trade, what can you do to reduce costs and support the new client’s usage patterns?

*As the billing increase is due to the overall loading increase, a breakdown of charge increase should be analysed.*

*At hosting level:*

*The heavily used services should be upgrade to a better cpu/memory/bandwidth combination instance. Identify the under used resources. For example, if there are stand-by redundancy servers running idle, they could be added to load balancing group to scale up the system and still serving the purpose of fall-back backup.*

*At software application level:*

*The database should be profiled and hot spot should be identified. Database optimisation such as indexing, stored procedure optimisation, better backup scheduling should be applied accordingly.*

*The endpoints server should also be reviewed, examine the current caching strategy and improve the coding quality of components that involves the computing intensive operations.*

## Step#3 – They feel the need for Speed…

Wazoo Holdings have a new requirement for their B group endpoint calls to go twice as fast or they will go elsewhere. Accounts won’t approve a doubling of system ‘grunt’ to do the job. On analysis you find the requests spend a third of their time waiting on the OAUTH service, and that of the data returned per call two thirds of it never changes. What could to change or add to take advantage of this information?

*The OAUTH token expiration time may need to be increased if it was set too short. The multiple endpoint groups could also share the same OAUTH credential. If OAUTH is being provided by a third party, other providers should be considered according to their geo-location relative to the clients to reduce network latency.*

*The endpoint server could be implemented in a way that the continuous client calls in a time interval can be bundled into bags of calls before been transferred. And endpoints calls should be gzipped, so the single call request/response size can be reduce to improve traffic.*

*If the current loading is hitting the ceiling of network bandwidth limit, a better bandwidth plan should be considered to save round-trip time of the service calls and also saves overall billing.*

## Step#4 – The End is Nigh..

Blah Inc management have just ‘discovered’ BCP and have put down an edict that all the systems they interface with must be fully BCP compliant and operate Hot-Hot 24x7. They are prepared to pay, so finance will permit a doubling of systems. What changes would you make to the system to enable Hot-Hot and how would you ensure everything is indeed in sync and correct?

*As finance permits a doubling of systems, the read-only group like heavily used B group, should be reading from a mirrored databased that is dedicated to read operations, this will leave room for better I/O and better database optimisation.*

*To ensure things are in sync and correct comparing the original site, aside from the synchronisation strategy provided by the hot sites hosting provider complying with BCP(Business Continuity Planning), the original site should also implement periodical synchronisation checking and validation as a separated daemon thread. For example, this can be a file level checksum comparison between original and hot sites.*

## Step#5 – Hackers at the gates…

Someone told the hackers you have Peta Bytes of privacy rich information for the picking, so you are suddenly faced with an avalanche of dictionary & scripted attacks from various dark corners of the Internet – you need to cost effectively lock them out without negatively impacting your existing customer base performance. What tech do you need to add to ensure your whole system locks the hackers out in lock step?

*The security flaws can happened at different levels of the running business. The technologies can be utilised may include:*

* *Review firewall policies, close all the ports that are not used, and enforce whitelisting access control for identified clients*
* *Use TSL encryption on all the public message transport*
* *Use Cloud-based DDOS protection*
* *Randomised token expiration and update secrets*
* *Limit crawler by setting proper site robot.txt*
* *Use secured VPN tunnelling*

## Step#6 – Never cross the privacy streams…

Management want to know what specific features of the system actively stop one client from ever seeing another client’s data. What technology or cast iron techniques do you employ to ensure to them that this is never possible?

*The access control to individual client should be applied:*

* *Individual database user account should be assigned to different clients.*
* *Finer grained database access control such as Row-level permission access control to different user accounts.*
* *Encrypt the client private data with a dedicated secret for each of them*

## Step#7 – Web hooks all the way down

XYZ Inc have a requirement that when certain datums get updated they need the B endpoint ran and its results sent to them via a web hook, which takes an absolute age to complete and often drops out for no apparent reason. How can you set this up to be performant, secure and reliable?

* After B endpoint ran, the results should be encrypted and stored in a posting quet.
* Set a timeout for resending failed POST, and re-queue it when maximum number of re-try is reached.
* The POST queue should be implemented in a non-blocking process, for example, node.js’ use of libuv
* Use TSL secured channel for making POST request and encrypt data with a dedicated secret always