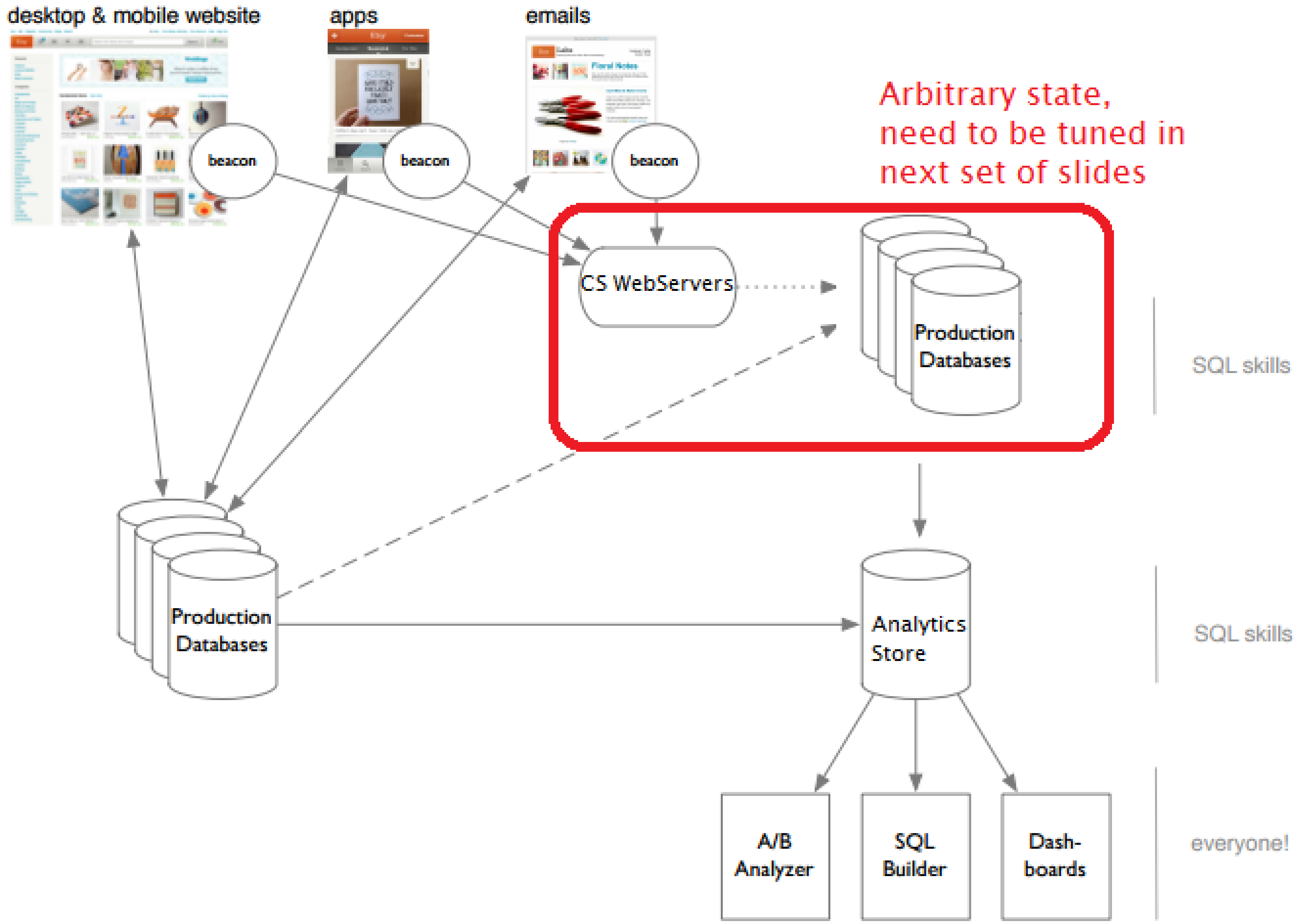
Use case – Clickstream Analytics

Recap - Usecase

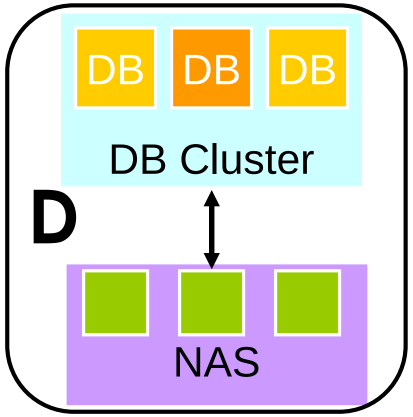
Use Case: Analyzing large volume of log data, without taxing the databases

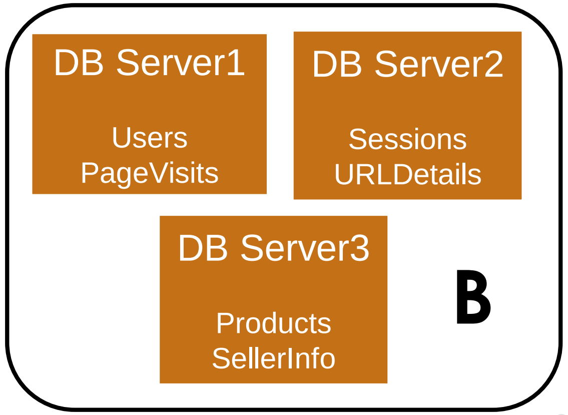
An online store, wanting to do Clickstream Analytics

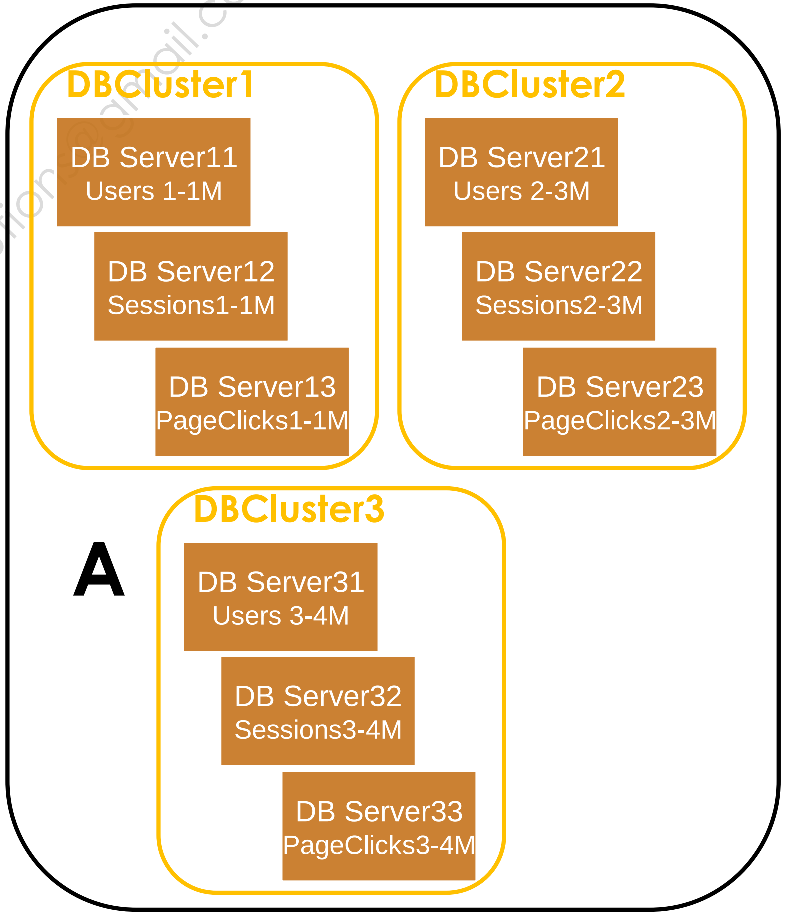


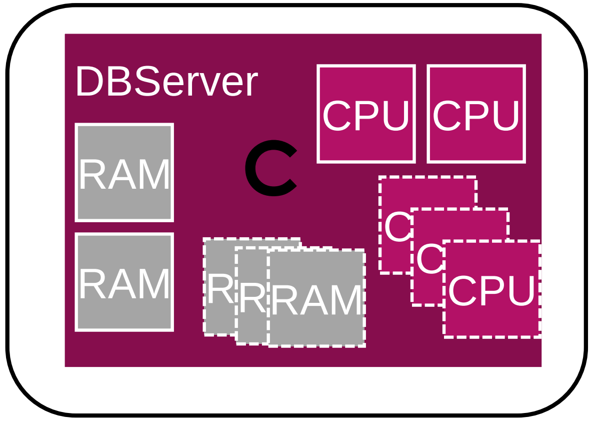
Arbitrary state, need to be tuned in next set of slides

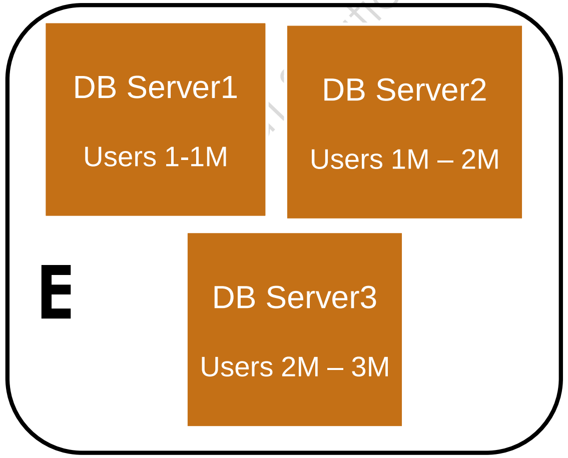
Recap – DB Scalability







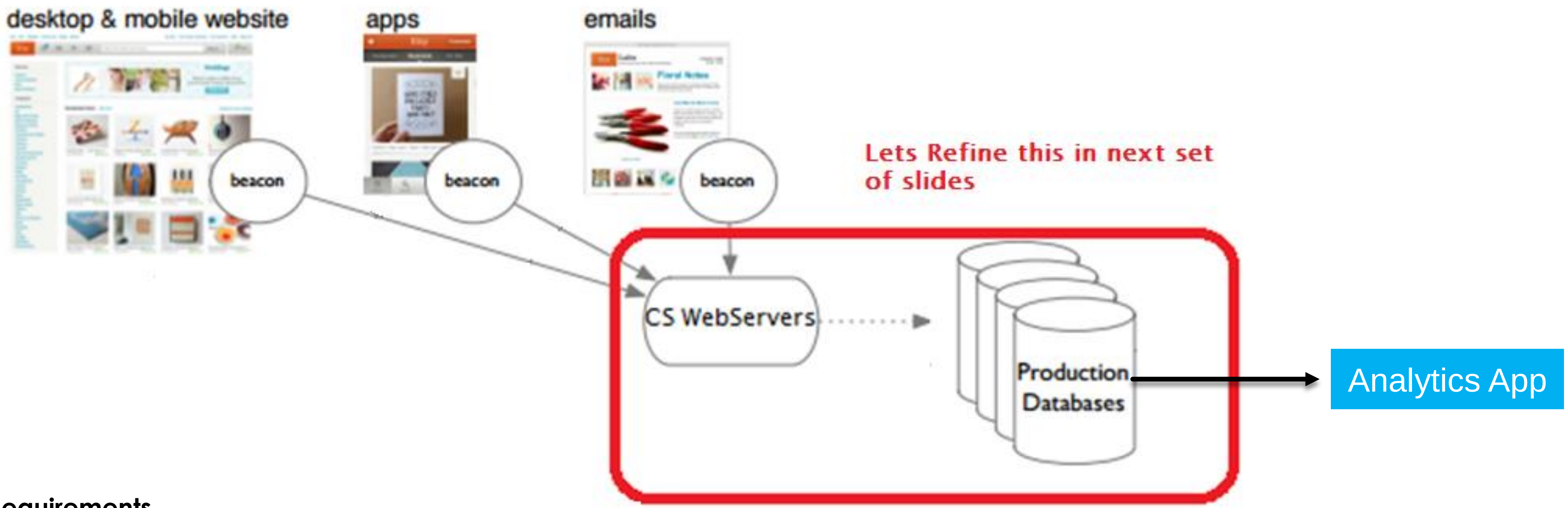




Match the logical architectures with the provided names

1. **Vertical Scaling**
2. **Horizontal Partitioning(Sharding)**
3. **Horizontal Scaling**
4. **Vertical Partitioning**
5. **Hybrid (y & z of AFK cube)**

System architecture (TBE)



Lets Refine this in next set of slides

Business Requirements

Mgmt wants an Analytics application which can let the Business promotors and Decision makers know about the below –

- Insights into which product is being viewed the most?
- What are the top 10 URL's (~products) that being viewed by users?
- What other URL's (~products) are being viewed by the users who have accessed any of the above top 10 URL's?

DB Schema

CS_ID	User_ID	URL	PageViews
111111111	U1	http://.../headset	1
111111112	U1	http://.../samsungn5	1
111111113	U2	http://.../iphone	2
111111114	U3	http://.../iphone	1
111111115	U1	http://.../iphone	3

Lets say user1 views sony xperia T3
and
user3 again visits the iphone page, the db is updated as follows

CS_ID	User_ID	URL	PageViews
111111111	U1	http://.../headset	1
111111112	U1	http://.../samsungn5	1
111111113	U2	http://.../iphone	2
111111114	U3	http://.../iphone	**2**
111111115	U1	http://.../iphone	3
111111116	**U1**	**http://.../sonyt3**	**1**

Refining by issues - 1

The system has gone to production, and after a month you see disk space issues, you see **no space left** errors for new click stream data.

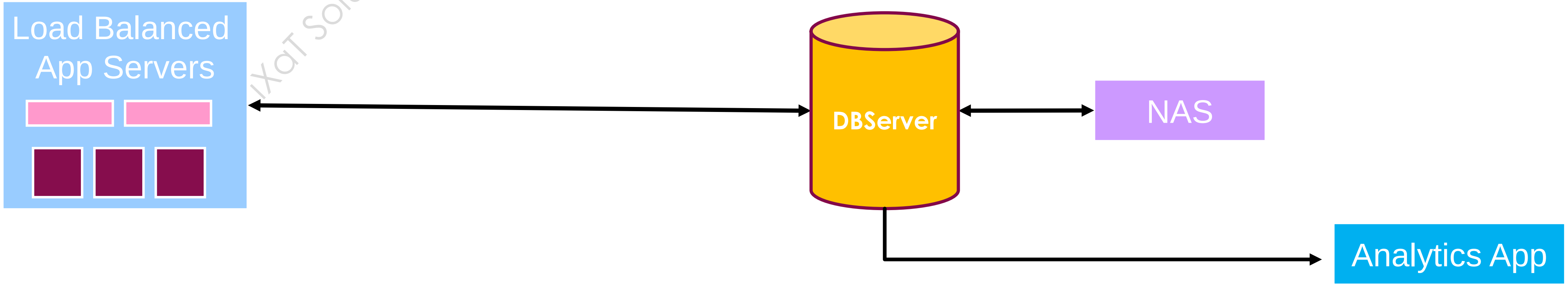
How do you mitigate the issue?

Refining by issues - 1

Solution – Add disk?, how about a month later? There is a limit to #of disks on a server

Or may be purchase/loan out NAS and mitigate the risk

Better option is to augment storage via h/w partitioning (NAS/SAN)



Refining by issues - 2

After a couple of months, you see weird error’s in inserting the data.

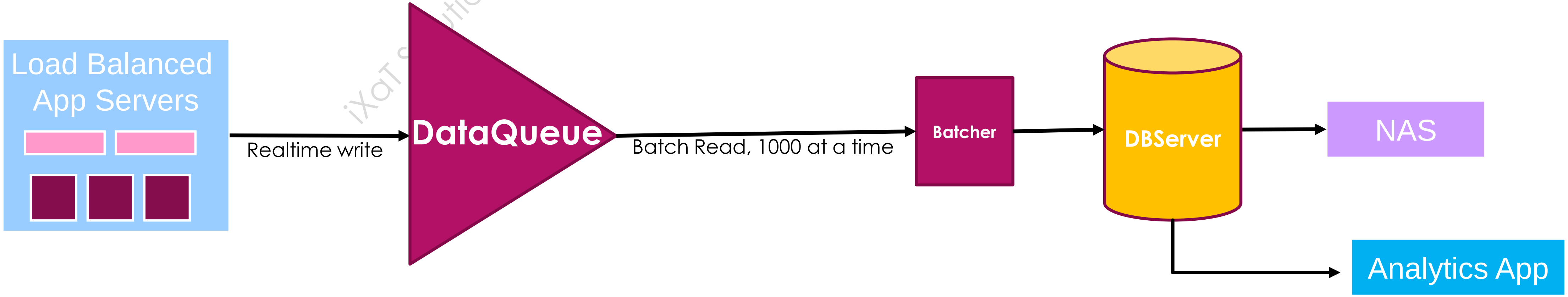
Logs reveal sporadic **timeout** errors in Insert and Updates, Analytic applications have not reported any error. i.e. updates & inserts are failing but selects are working good.

Refining by issues - 2

Quick Solution – Add a backing store and implement batching on top of it.

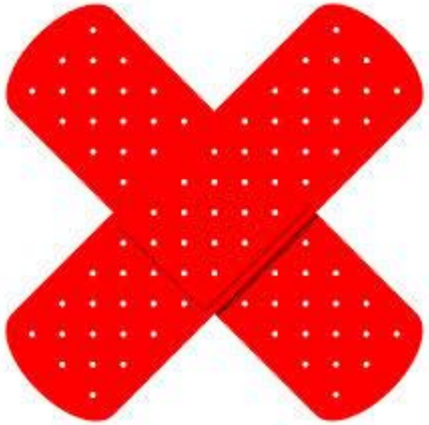
A backing store using a queue suites better here because of the one way pattern.

Also, we do not loose much data if any issues on DB.



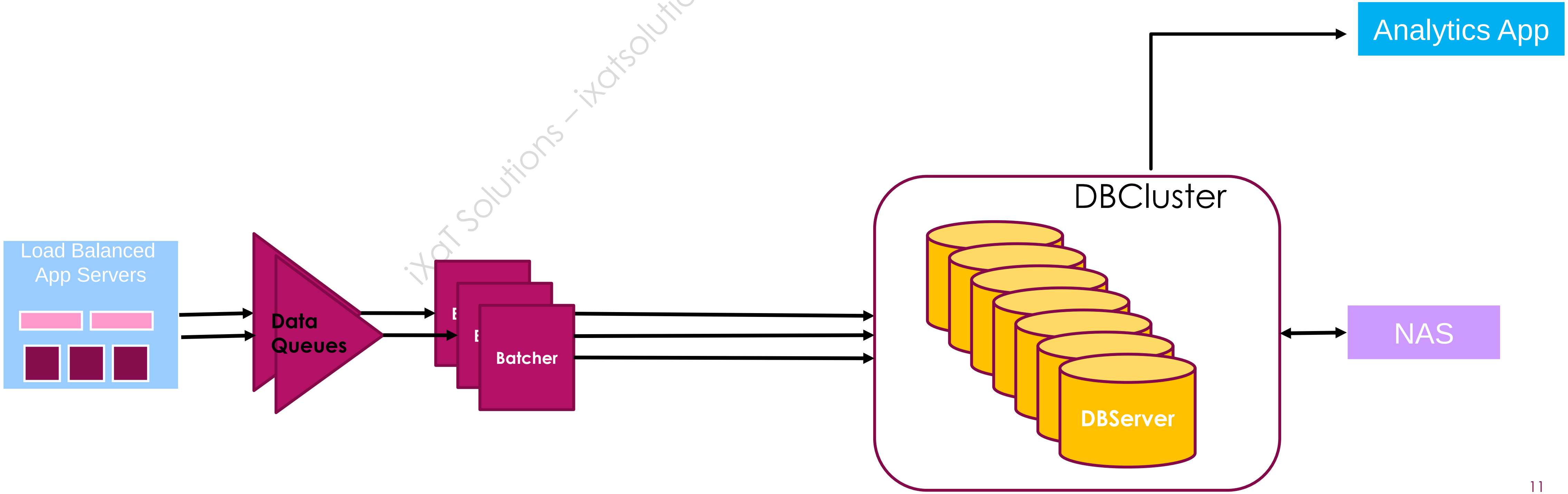
Refining by issues - 3

After a while your application has started getting huge volumes of data, you now have more batch updates to DB, thus overwhelming them and causing them to fail again, earlier solution of implementing a queue was a band-aid solution. You now need a more robust solution



Refining by issues - 3

Implement Sharding, split writes to the DB's.



Refining by issues -3 DB Sharding

Sharding is a technique to scale DB writes by balancing out write loads across DB servers, you would also choose shared nothing architecture to balance load at storage (NAS).

Points to note:

- We need to decide upon the key which would be used to move the rows across. If CS_ID is choosen to be the key, then, we could use modulo operator on CS_ID and number of shards. Hashing might be a good function in combination with a modulo.
- We need to write a utility to map and migrate the data from one DB server to all others in the cluster.
- Also the batcher need to be modified to understand the sharding logic, and the number of shards.
- We need to shutdown the batcher jobs when the utility runs, this should not cause any data loss nor business impact because the queue holds "on the fly" data.
- Also, the query which fetches top 10 URL's need to be modified.

DB Server1

CS_ID	User_ID	URL	PageViews
111111111	U1	http://…	299
111111112	U1	http://…	223

DB Server2

CS_ID	User_ID	URL	PageViews
111111113	U2	http://…	334
111111114	U2	http://…	342

DB Server3

CS_ID	User_ID	URL	PageViews
111111115	U2	http://…	11
111111116	U1	http://…	33

DB Server4

CS_ID	User_ID	URL	PageViews
111111117	U1	http://…	45
111111118	U2	http://…	211

DB Server5

CS_ID	User_ID	URL	PageViews
111111119	U3	http://…	4445
111111120	U3	http://…	222

Refining by issues -4 DB Sharding

What if the number of shards that we have are not scaling to our need after a while?

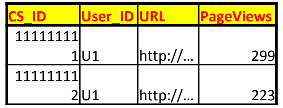
Add more shards.
Run the mapper utility agan.

Points to note:
What if the number of shards become too many? The mapper script would run for ages. What do you do now?

What if because of a bug the new shard# is not picked by the batchers?

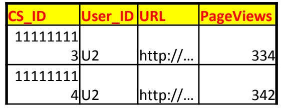
One more not infrequent requirement (non-functional) –
What you do when a shard is down?

Server1



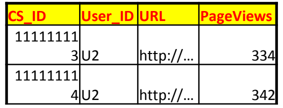
CS_ID	User_ID	URL	PageViews
111111111	U1	http://...	299
111111112	U1	http://...	223

Server2



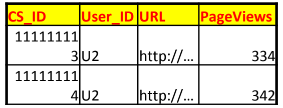
CS_ID	User_ID	URL	PageViews
111111113	U2	http://...	334
111111114	U2	http://...	342

Server3



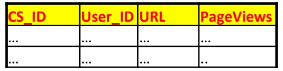
CS_ID	User_ID	URL	PageViews
111111113	U2	http://...	334
111111114	U2	http://...	342

Servern



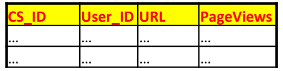
CS_ID	User_ID	URL	PageViews
111111113	U2	http://...	334
111111114	U2	http://...	342

Servern+1



CS_ID	User_ID	URL	PageViews
...	...	...	...
...	...	...	..

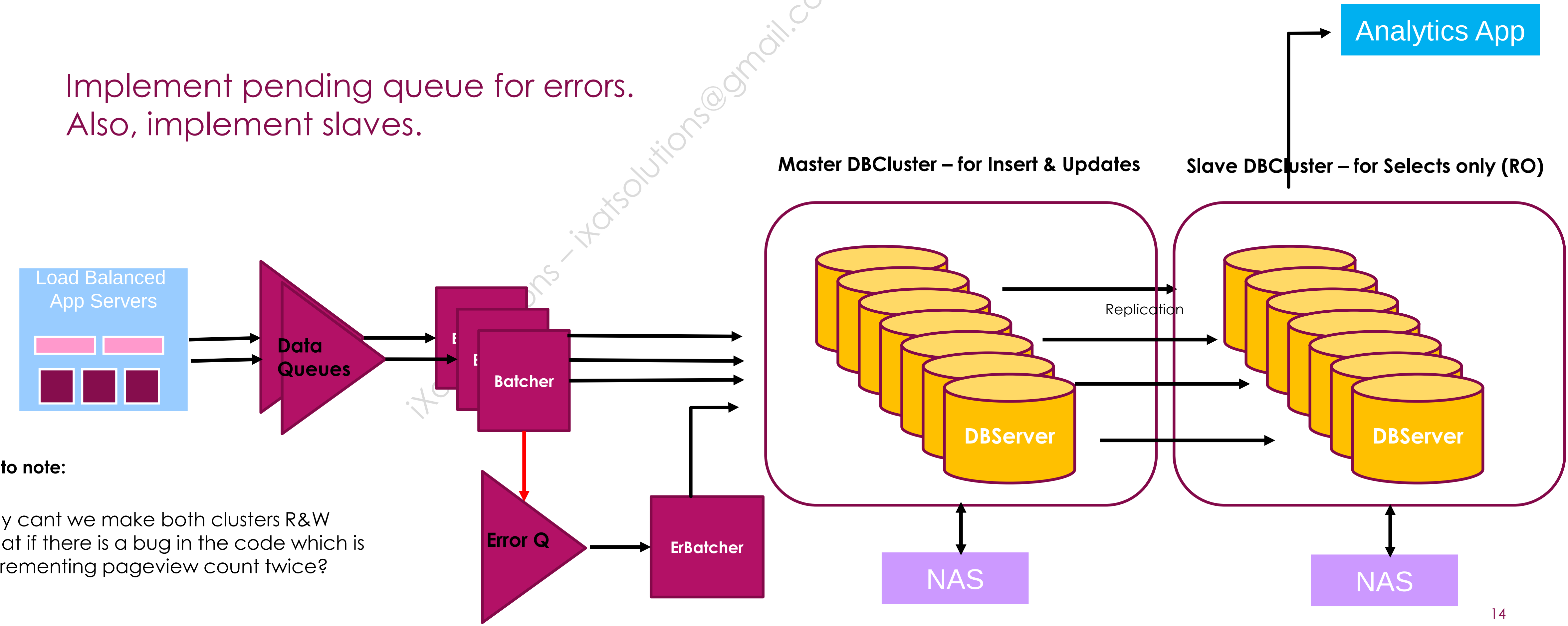
Serverx



CS_ID	User_ID	URL	PageViews
...	...	...	...
...	...	...	...

Refining by issues – 5, Slave DB’s

Implement pending queue for errors.
Also, implement slaves.


Analytics App
Master DBCluster – for Insert & Updates
Slave DBCluster – for Selects only (RO)
Load Balanced App Servers
Data Queues
B
B
Batcher
Replication
DBServer
DBServer
Error Q
ErBatcher
NAS
NAS

Points to note:

- Why cant we make both clusters R&W
- What if there is a bug in the code which is incrementing pageview count twice?

Why so many iterations?

- DB is not self-aware of its Distributed Nature, hence the application need to implement the sharding/re-sharding scripts
- The DB is also not aware of Distributed Queries. Hence the app need to take care of query split-merge.
- Bugs are inevitable, the system should be self healing even if there is a human mistake.
- The system has become complex, and that means there can be many more bugs and the system should be deigned to be resilient to mistakes.

What if?

The Database schema is re-architected and made immutable? i.e remove the necessity to do an update.

Earlier

CS_ID	User_ID	URL	PageViews
111111111	U1	http://…/headset	1
111111112	U1	http://…/samsungn5	1
111111113	U2	http://…/iphone	2
111111114	U3	http://…/iphone	1
111111115	U1	http://…/iphone	3

User_ID	UserName
U1	Sam
U2	Joseph
U3	Ramu

Now

User	URL	Timestamp
Sam	http://…/headset	xxx
Sam	http://…/samsungn5	xxx
Joseph	http://…/iphone	xxx
Joseph	http://…/iphone	xxx
Ramu	http://…/iphone	xxx
Sam	http://…/iphone	xxxx
Sam	http://…/iphone	xxxx
Sam	http://…/iphone	xxxx

Many Bigdata and Data Warehouse cases fall in this category

Points to note:

- We can easily change this system to use immutable model because of the fact that the data is historical and **append** only.
- If required this data could be de-normalized so that we can avoid joins (thus avoid cs_id colum as well)

What if?

- What if we have a system/framework which understands the distributed nature of data and takes care of partitioning, balancing, re-balancing, replication, data pipelining etc… on this own? **Distributed**
- What if we have a system that is aware that hardware failures are ineitable and thus takes that into consideration always – i.e **fault tolerant**?
- What if there is a system which can be used for any type of use cases which fit in the model that we discussed? **Generic**
- What if there is a system that can be extended with capabilities if the business demands them? **Extensible**
- What if there is a system that be easily managed and have smooth running of business? **Manageable.**
- What if there is a system which is cost effective and has good support? **Cost effectiveness**
- What if we have a system/framework which is elastic in scale and can handle huge loads of data? **Scalabale and Robust**.

Congratulations – enter Hadoop!

Homework

- Read the below stuff
  - Computer Hardware - https://web.stanford.edu/class/cs101/hardware-1.html
  - Blade Server - https://en.wikipedia.org/wiki/Blade_server
  - Storage - http://explainingcomputers.com/storage.html
  - IPC - https://en.wikipedia.org/wiki/Inter-process_communication

- Install Ubuntu on your laptops
  - http://www.krizna.com/windows-7/install-ubuntu-windows-7-virtualbox/
  - Practice some Linux - http://freeengineer.org/learnUNIXin10minutes.html