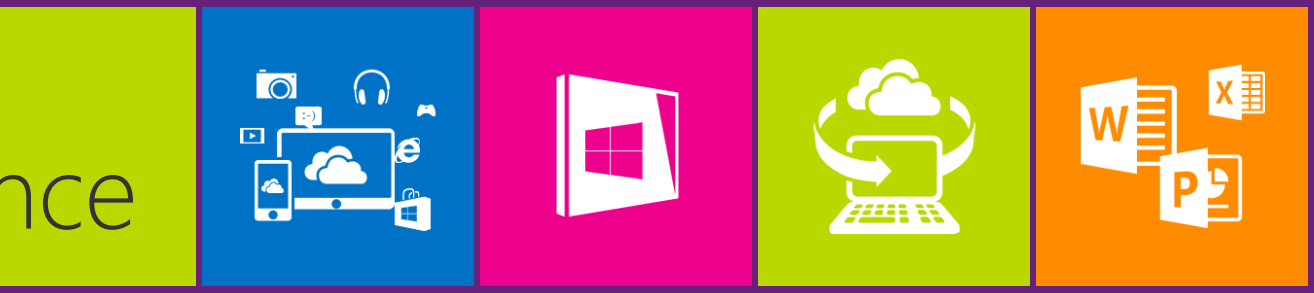
Microsoft

tech·days

Hong Kong | 2013

You make
the difference



How to scale SQL Azure databases

Edwin Cheung
Principal Program Manager
China Cloud Innovation Centre
Customer Advisory Team
Microsoft Asia-Pacific Research and Development Group

Agenda

Discussion Topics:

- PaaS vs. IaaS
- Application Scale/Performance fundamentals
- Scale-up vs. scale-out
- Data Placement & Data Access
- The Path to Scale

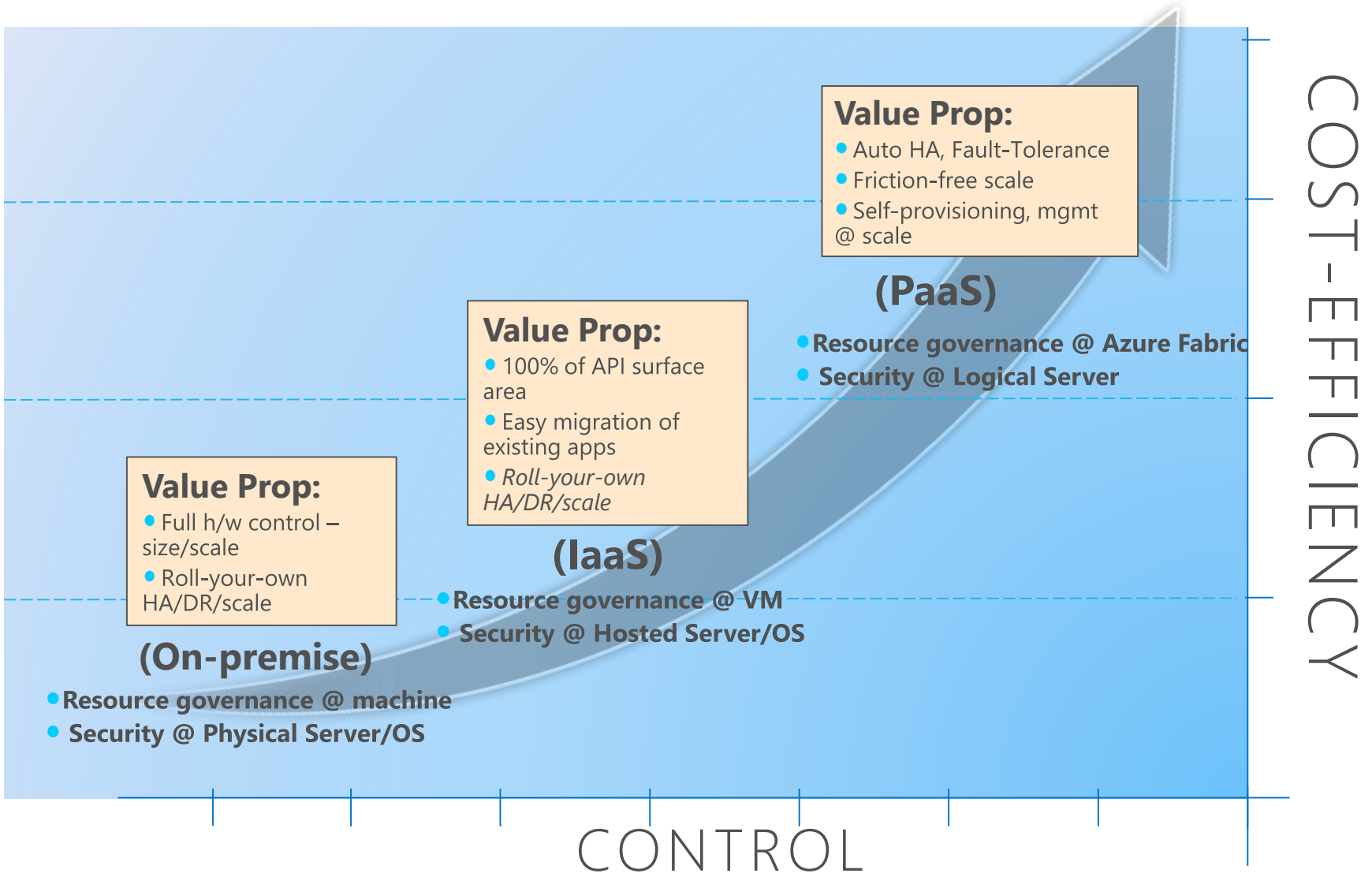
Real World Examples

Fit for Purpose – What makes a good cloud app?

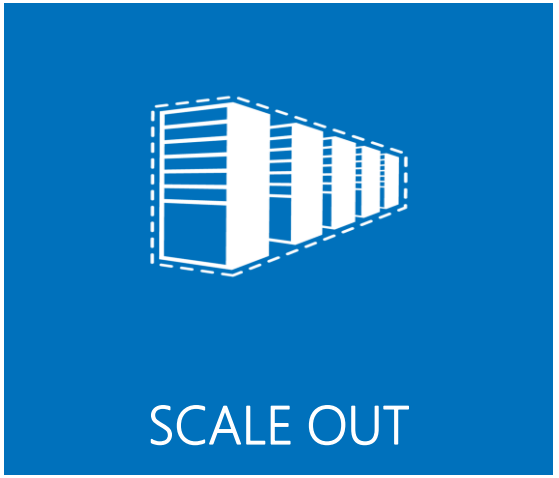
Box

Cloud

PaaS vs IaaS


Value Prop:
Auto HA, Fault-Tolerance
Friction-free scale
Self-provisioning, mgmt @ scale
(PaaS)
Resource governance @ Azure Fabric
Security @ Logical Server
Value Prop:
100% of API surface area
Easy migration of existing apps
Roll-your-own HA/DR/scale
(IaaS)
Resource governance @ VM
Security @ Hosted Server/OS
Value Prop:
Full h/w control – size/scale
Roll-your-own HA/DR/scale
(On-premise)
Resource governance @ machine
Security @ Physical Server/OS
COST-EFFICIENCY
CONTROL

Fit for purpose – good cloud app?



If your application is not suited for scale-out, it's **not suited for the cloud**

What is Large Scale?

For the purposes of this presentation

Millions of users

Hundreds of thousands of operations per second

Thousands of cores

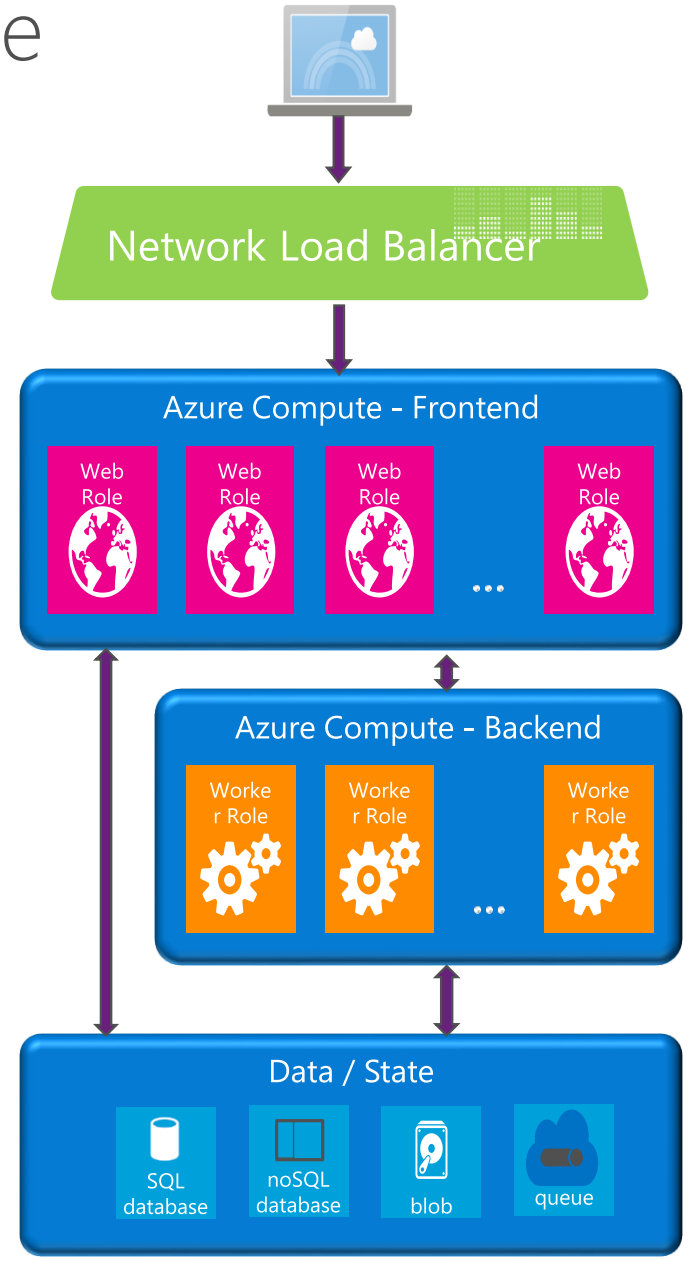
Hundreds of databases

Top statistics for 1 app

Category	Metric
Largest sharded SQL database	20 TB
Largest number of databases	11,000
Most worker instances	6,000
Most number of cores	24,000
Largest Microsoft App	1 Exabyte
Largest Customer App	50 PB

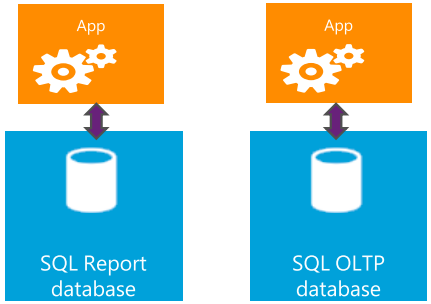
Application Scale/Performance Fundamentals

- Workload driven design
- Fan-out the application tier to support the workload
- Loosely couple: asynchronous, but keep close and process fast
- Use cache for data and content
- Batch together storage/database operations if possible
- Partition Azure storage objects across multiple storage accounts if necessary
- Size appropriate (instance size and number of instances)

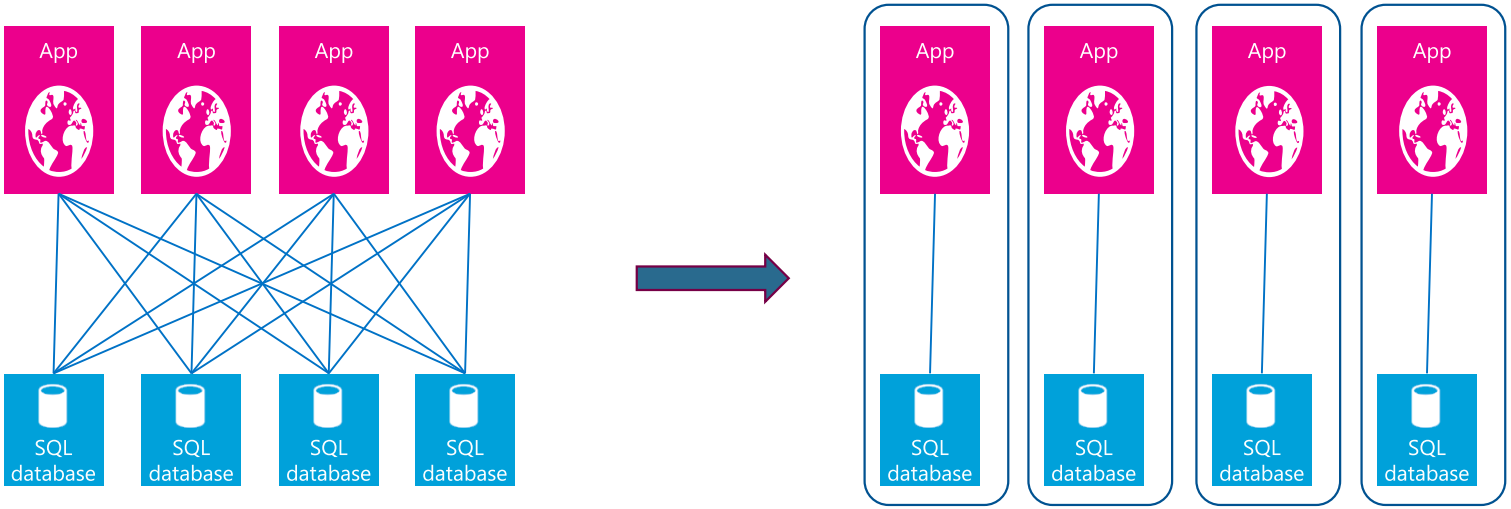


Architect Application Based on Workload

- Read-only data separated from read-write data

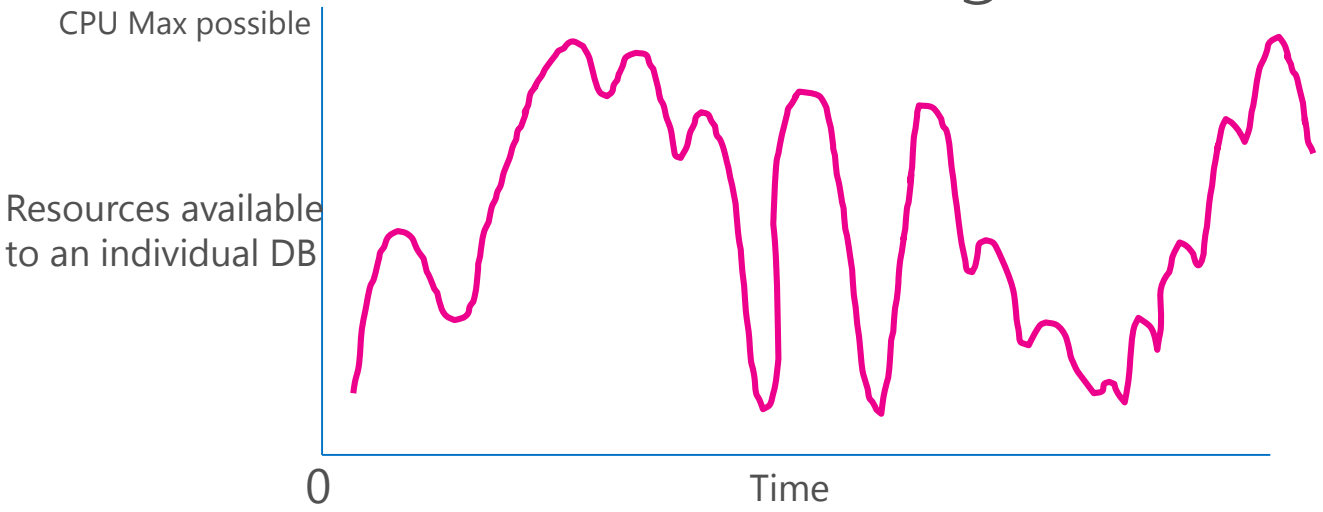


- Affinitize application and database as scale unit

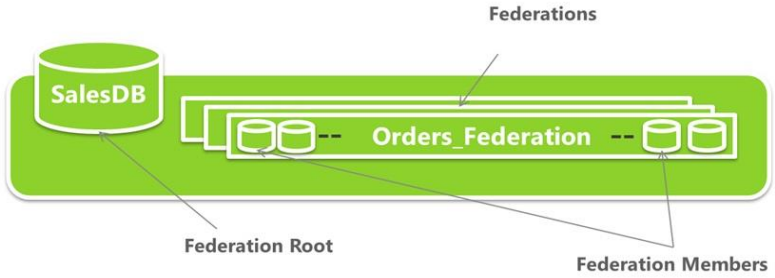


SQL Azure Scale-out

Multi-tenant resource sharing environment



SQL Azure Federation



Data Placement in the Cloud

1. Structured or un-structured data
2. Persisted or temporary data
3. Secure or insecure data
4. Other Factors –
   - Client connectivity
   - Performance
   - Availability
   - Manageability

Storage Offering	Purpose	Maximum Size
Local Storage	Per-instance temporary storage	250GB to 2TB
Windows Azure Storage	Durable storage for.....	
Blob	Large binary objects such as video or audio	200GB or 1TB
Table	Kay/value pairs	100TB
Queue	Inter-process Messages	100TB
SQL Database	Relational Database Management System	150GB

Persisted VM role data placement

- System Drive (C:\)
- Temporary Storage (D:\)
- Attached Disks
  - Keep write caching setting off

Virtual Machine Sizes and SQL Server Editions					
VM Size	**CPU Cores**	**Memory**	**Bandwidth**	**# Data Disks**	**SQL Server Edition**
Extra small	Shared	768 MB	5 (Mbps)	1	Express
Small	1	1.75 GB	100 (Mbps)	2	Any
Medium	2	3.5 GB	200 (Mbps)	4	Any
Large	4	7 GB	400 (Mbps)	8	Any
Extra large	8	14 GB	800 (Mbps)	16	Any
Each persistent data disk can be up to 1 TB					

SQL Azure Data Access in the Cloud

Data access implementation in Windows Azure and SQL Azure introduces additional complexity around two main areas

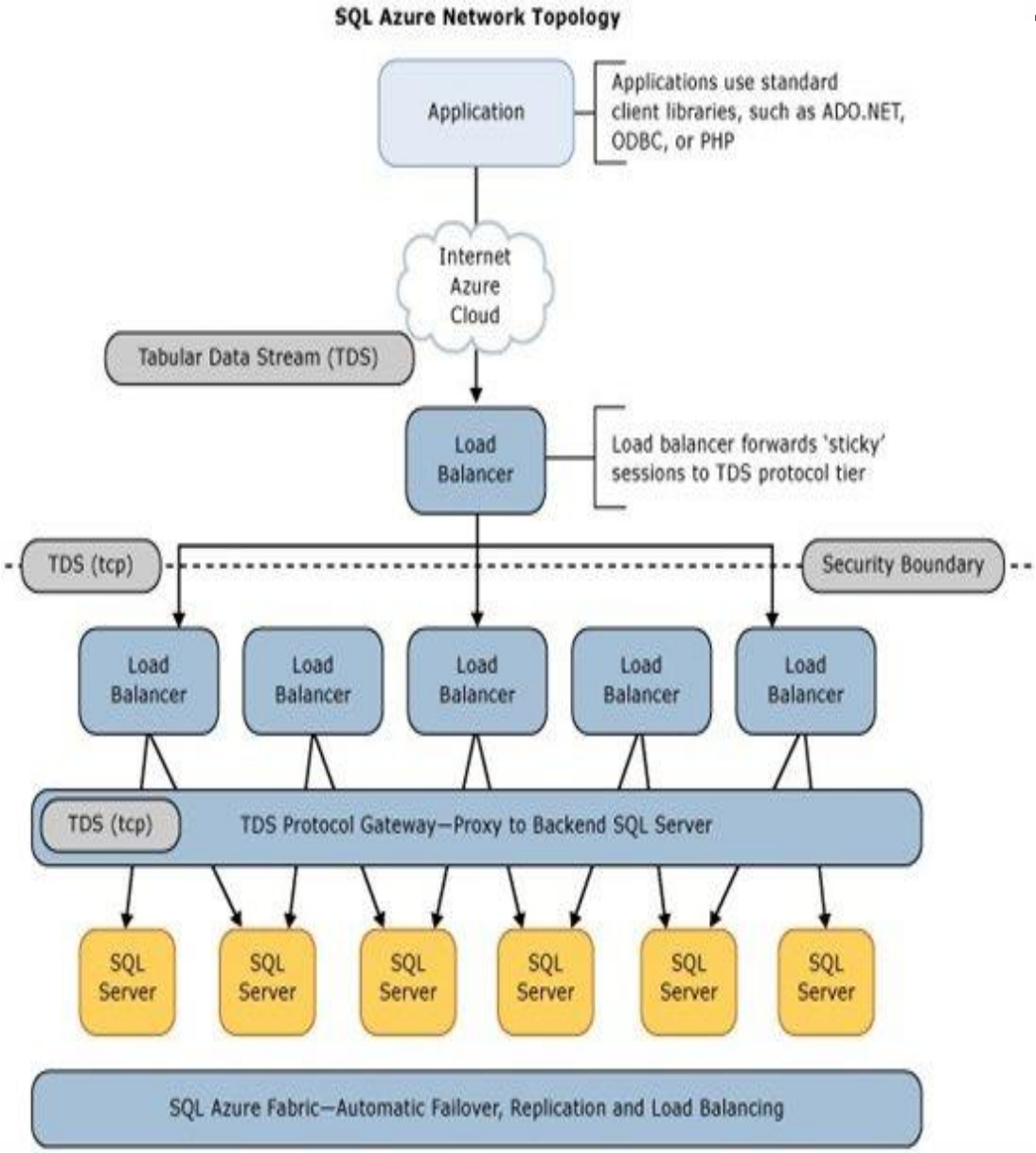
Connection management

- Retry logic necessary to implement reliable communications between application and database server

Latency between app tier and database tier on Azure is higher compared to an on premises deployment

- Firewalls, load balancers, gateways
- This amplifies the impact of chatty application behaviors

SQL Azure Network Topology



Connection Management

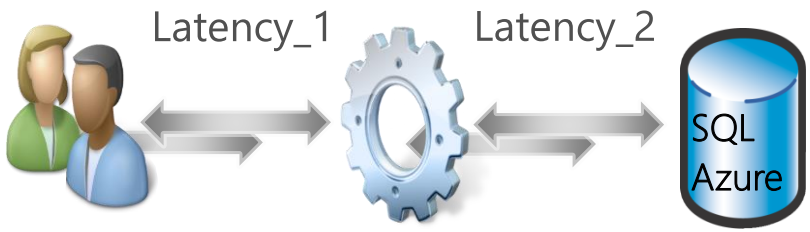
- Connection loss
- Idle connections idle for 30 minutes or longer will be terminated
- Transaction termination: kills all transactions after they run for 24 hours.
- Denial of Service attacks: high number of login failures from a particular IP address, SQL Azure will block the connections from that IP address for a period of time
- Failover
- Load balancing
  - Microsoft Enterprise Library Transient Fault Handling Application Block

- Throttling
- Protect a machine from sustained high usage of system resources
- Evaluate actual resource usage vs. safe thresholds real-time
- Throttle the busiest DBs first (soft throttle)
- Throttle every DB if necessary (hard throttle)

Network Latency

- Network Latency:

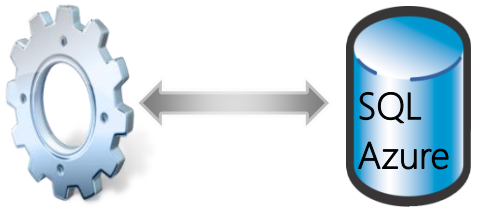
1. Between **User** and **Application**
2. Between **Application** and **SQL Azure DB**



- Perceived performance:
  - **Response Time** = 2x(Latency_1 + Latency_2) + App_Time + SQL_Time

- Optimization
  - Minimize latency 1: select data center closest to majority of your users
  - Minimize latency 2: co-locate with Windows Azure application
  - Minimize network round trips

Measure the Latency of SQL Azure



Blog Post: [Testing *Client* Latency to SQL Azure]

Results | Messages | Client Statistics

	Trial 1	Average
Client Execution Time	09:21:29	
Query Profile Statistics		
Number of INSERT, DELETE and UPDATE statements	0	→ 0.0000
Rows affected by INSERT, DELETE, or UPDATE statem...	0	→ 0.0000
Number of SELECT statements	1	→ 1.0000
Rows returned by SELECT statements	1	→ 1.0000
Number of transactions	0	→ 0.0000
Network Statistics		
Number of server roundtrips	1	→ 1.0000
TDS packets sent from client	1	→ 1.0000
TDS packets received from server	1	→ 1.0000
Bytes sent from client	46	→ 46.0000
Bytes received from server	37	→ 37.0000
Time Statistics		
Client processing time	0	→ 0.0000
Total execution time	122	→ 122.0000
Wait time on server replies	122	→ 122.0000

Measure the Latency of SQL Azure Cont.

From\To (ms)	North-central US	South-central US	North Europe	West Europe	East Asia	South-East Asia
North-central US	**5**	47	101.3	112.1	198.8	227.2
South-central US	39.6	**13.7**	121	116.9	181.1	209.6
North Europe	104.5	122.8	**4.5**	28.9	287.7	317.3
West Europe	119.7	127.2	29.8	**5.8**	310.8	341.4
East Asia	202.7	196.2	290.3	317.8	**1.9**	33.5
South-East Asia	230.8	230	318	350.2	32.8	**2.2**
Average Latency (ms)	5.0					
Fastest Latency (ms)	1.9					
Slowest Latency (ms)	350.2					

Chatty Application 1 - Server Cursor vs Client Cursor

- Server cursors cache results at server side and waits for clients to fetch and process data.
- Client cursors are implemented by caching all the result set rows on the client.
- Azure data center latency magnifies server cursor slow performance.
- The query below uses a server-side cursor to iterate through and retrieve 16,900 records (the entire table). 5 secs (on-premise) vs 50 secs (Azure)

  - *SELECT t.ID, t. RECORD FROM Customer_MVMT t*

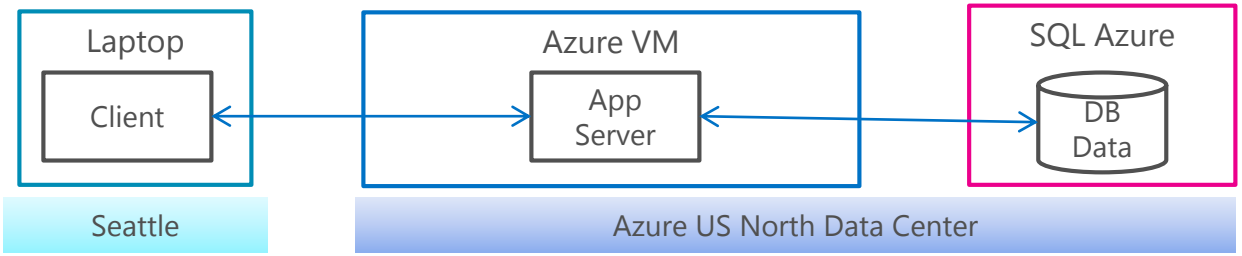
  - --Switch to Client side cursor and/or increase fetch batch size

Chatty Application 2 – Single Value Retrieval Access

- Uses 3rd-party framework for data access – single threaded
- Chatty behavior – Retrieve single value multiple times for one line item entry
- Moving to VM to mitigate the latency
- **App server & SQL server are closer**
- **VM local disk cache**
- In the long term implement client caching via EF (Entity Framework)

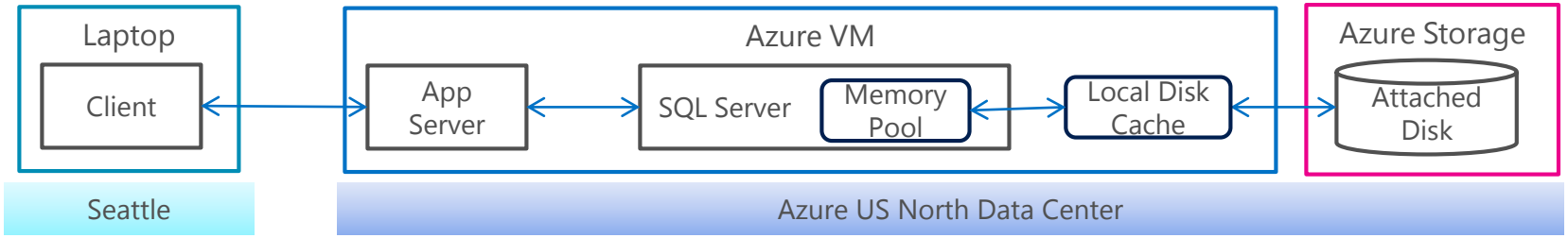
Chatty Application 2 test result - 20 Line Items Processed

1



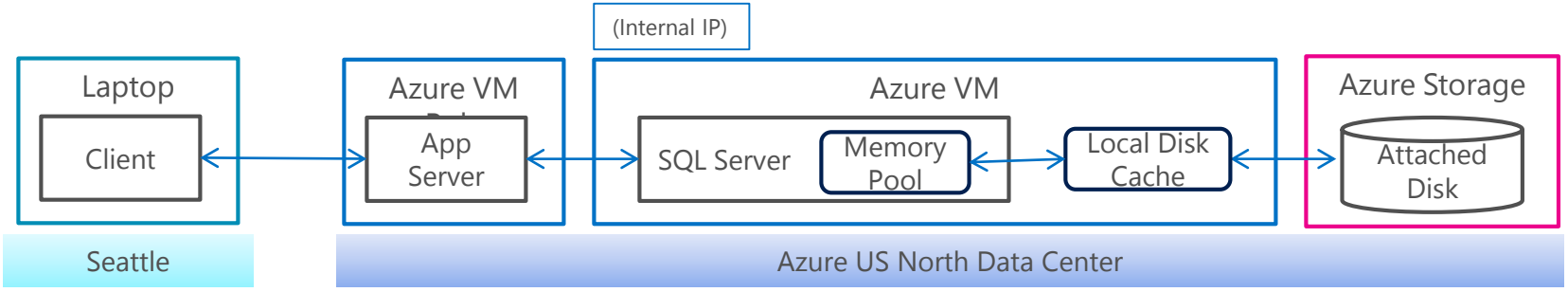
- DB Data is SQL Azure..
- Test Result: **240 seconds (avg)**

2



- App Server and DB Server are in the same box.
- Test Result: **110 seconds (avg)**

3



- AppServer and DB Server are in two separate VMs.
- Test Result: **150 seconds (avg)**

Chatty Application 3 – Ad-Hoc Queries

- Architecture produces ad-hoc queries to maintain a platform neutral database and a flexible framework.
- Tens of thousands of individual SQL queries may be issued in the process of opening a single 3D object for viewing or editing
- Recommendation: Batching up Ad-hoc queries

The Path to Scale

Capacity
Partition application, add additional scale-out capacity to meet demand

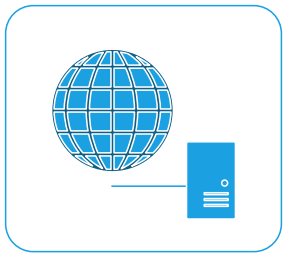
Optimize
Improve application density through optimum resource usage

Shift
Trade durability, queryability, and consistency for throughput, latency

Add Capacity – Partitioning and Scale Out

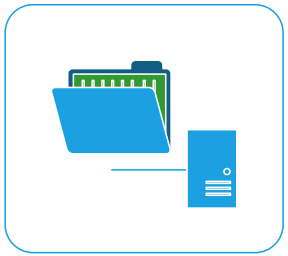
Azure architecture is based on scale-out

Composing multiple scale units to build large systems



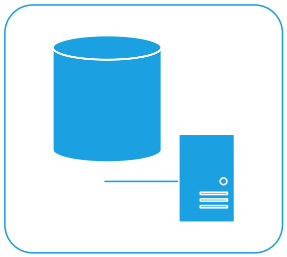
Azure Compute
(Web, Worker, IaaS)

- 1-8 CPU cores
- 2-14 GB RAM
- 5-800 Mbps network



Azure Storage

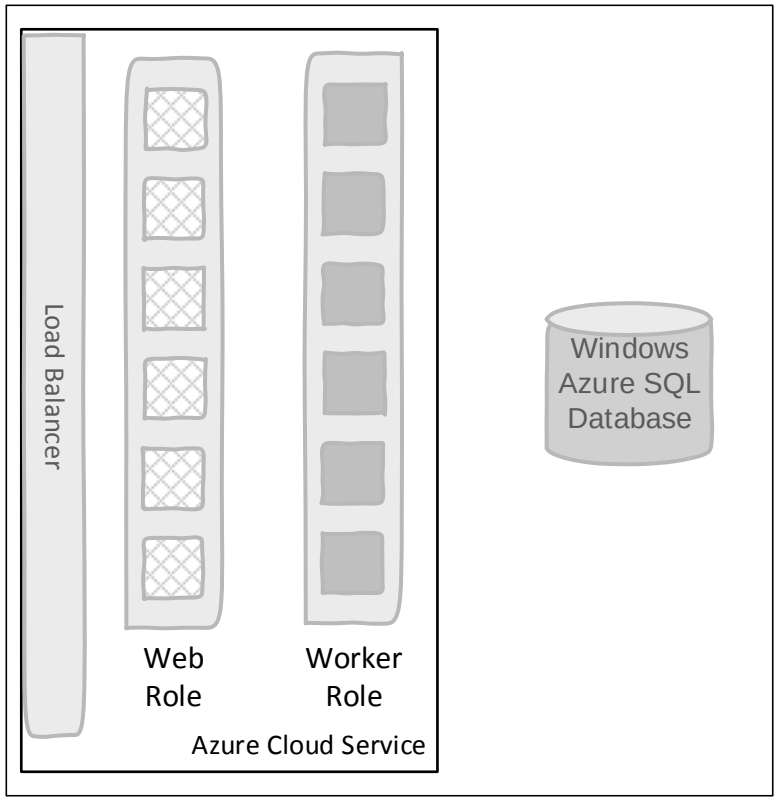
- 100 TB storage (max)
- 5000 operations / sec
- 3 Gbps



Azure SQL Database

- 150 GB
- 305 threads
- 400 concurrent reqs

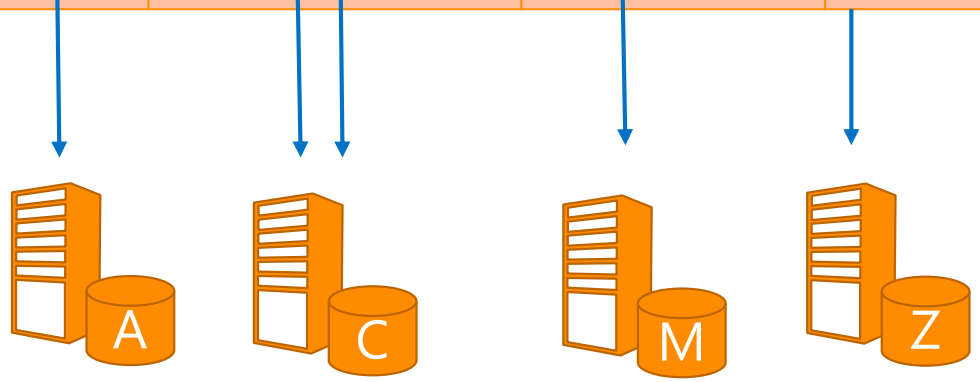
Evaluating Scale and Bottlenecks



Aspect	Partitioning	Capacity
Web role	Low state Automatic (via load balancer) Round-robin	Add more instances (easy)
SQL Database	High state Manual (via app code) Choose partitioning function	Add more databases

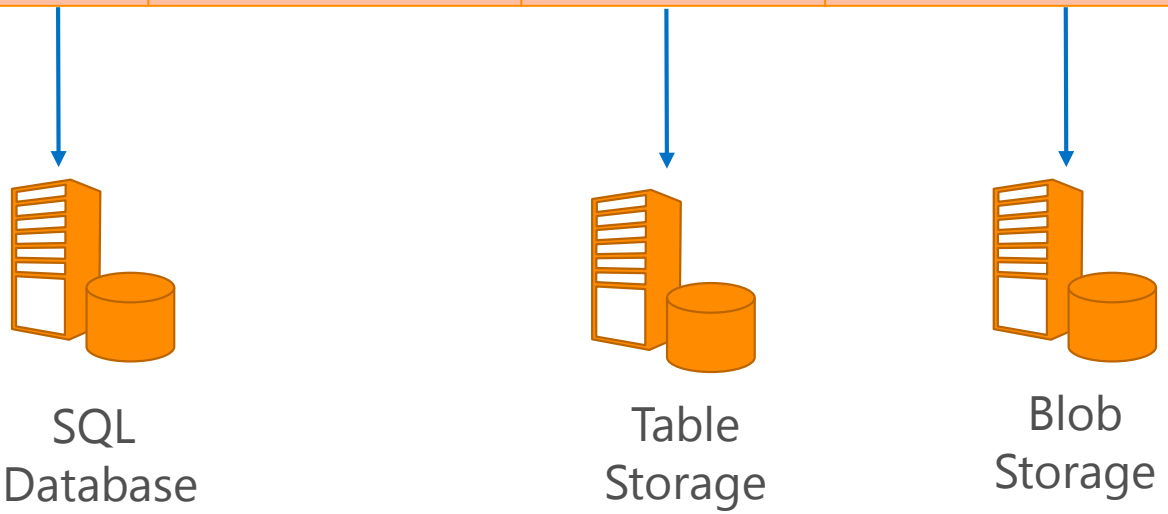
Horizontal Partitioning

First Name	Last Name	Email	Thumbnail	Photo
David	Alexander	davida@contoso.com	3kb	3MB
Jarred	Carlson	jaredc@contosco.com	3kb	3MB
Sue	Charles	suec@contosco.com	3kb	3MB
Simon	Mitchel	simonm@contoso.com	3kb	3MB
Richard	Zeng	richard@contosco.com	3kb	3MB



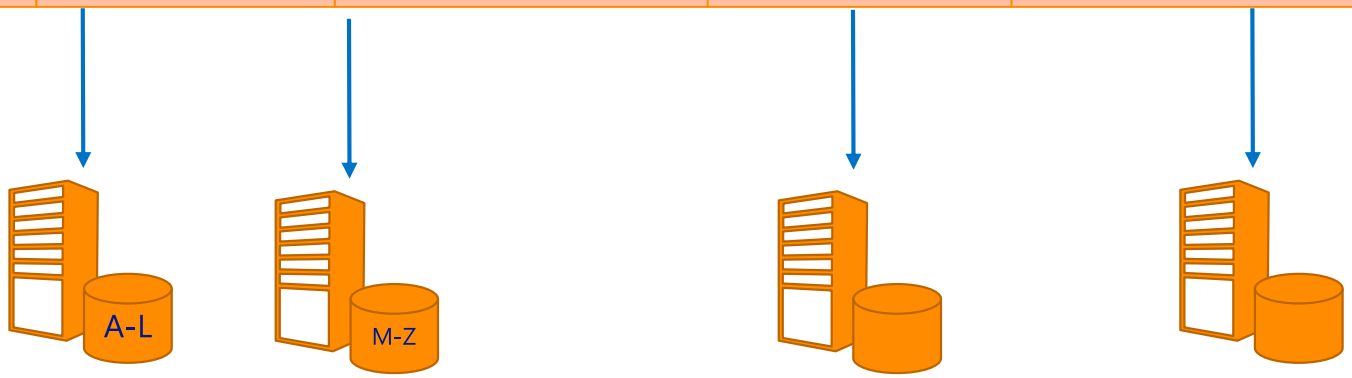
Vertical Partitioning

First Name	Last Name	Email	Thumbnail	Photo
David	Alexander	davida@contoso.com	3kb	3MB
Jarred	Carlson	jaredc@contosco.com	3kb	3MB
Sue	Charles	suec@contosco.com	3kb	3MB
Simon	Mitchel	simonm@contoso.com	3kb	3MB
Richard	Zeng	richard@contosco.com	3kb	3MB

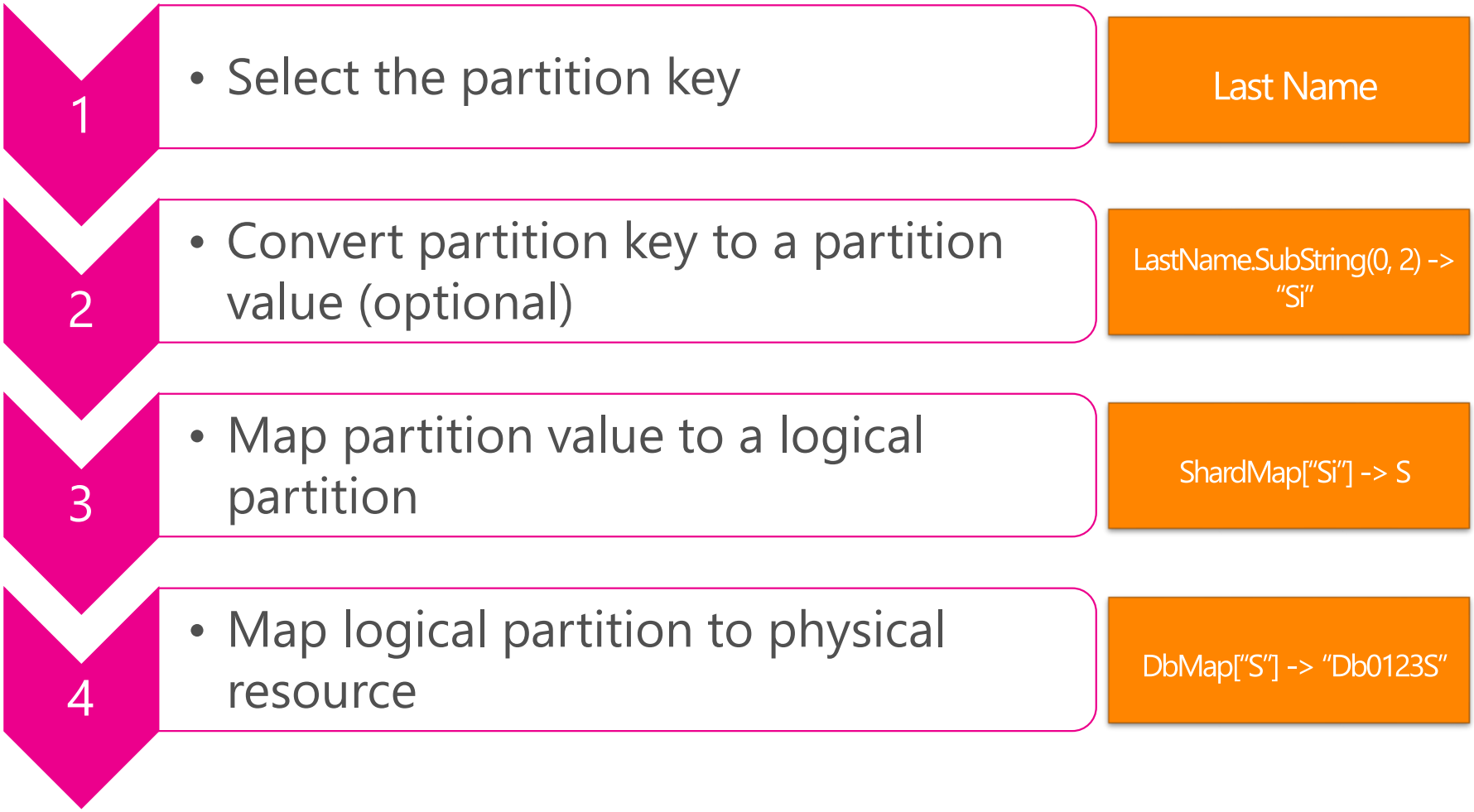


Hybrid Partitioning

First Name	Last Name	Email	Thumbnail	Photo
David	Alexander	davida@contoso.com	3kb	3MB
Jarred	Carlson	jaredc@contosco.com	3kb	3MB
Sue	Charles	suec@contosco.com	3kb	3MB
Simon	Mitchel	simonm@contoso.com	3kb	3MB
Richard	Zeng	richard@contosco.com	3kb	3MB



Partitioning



Partitioning (Range Based)

1. The user (user ID) is a natural partitioning key; all workloads are user-centric

   "MaSimms"

2. Use a non-cryptographic hash to convert the user ID to an integer value

   639837447

3. Map a range of integers to a logical "shard"

   ShardMap.FirstOrDefault(e => e.IsInRange(639837447))

4. Map logical "shard" to physical resource (database)

   DbMap[Shard].ConnectionString

Partitioning Approaches

Range Based

Split and merge the partition range into segments

Logical Buckets

Assign data to a logical bucket, then map to a physical resource

Lookup Assignment

Lookup table to map to physical resource segment

Isolation Options (Logical->Physical)

Database as a container

Logical shard occupies a single database.

Pros: high isolation, simple mapping (isolated to connection string),

Cons: density (for small tenants) and capacity (very large tenants), rebalancing (1:1 map)

Schema as a container

Logical shard occupies a distinct schema in a database.

Pros: medium isolation, density (multiple tenants per database)

Cons: data layer mapping (need to dynamically generate and inject schema context)

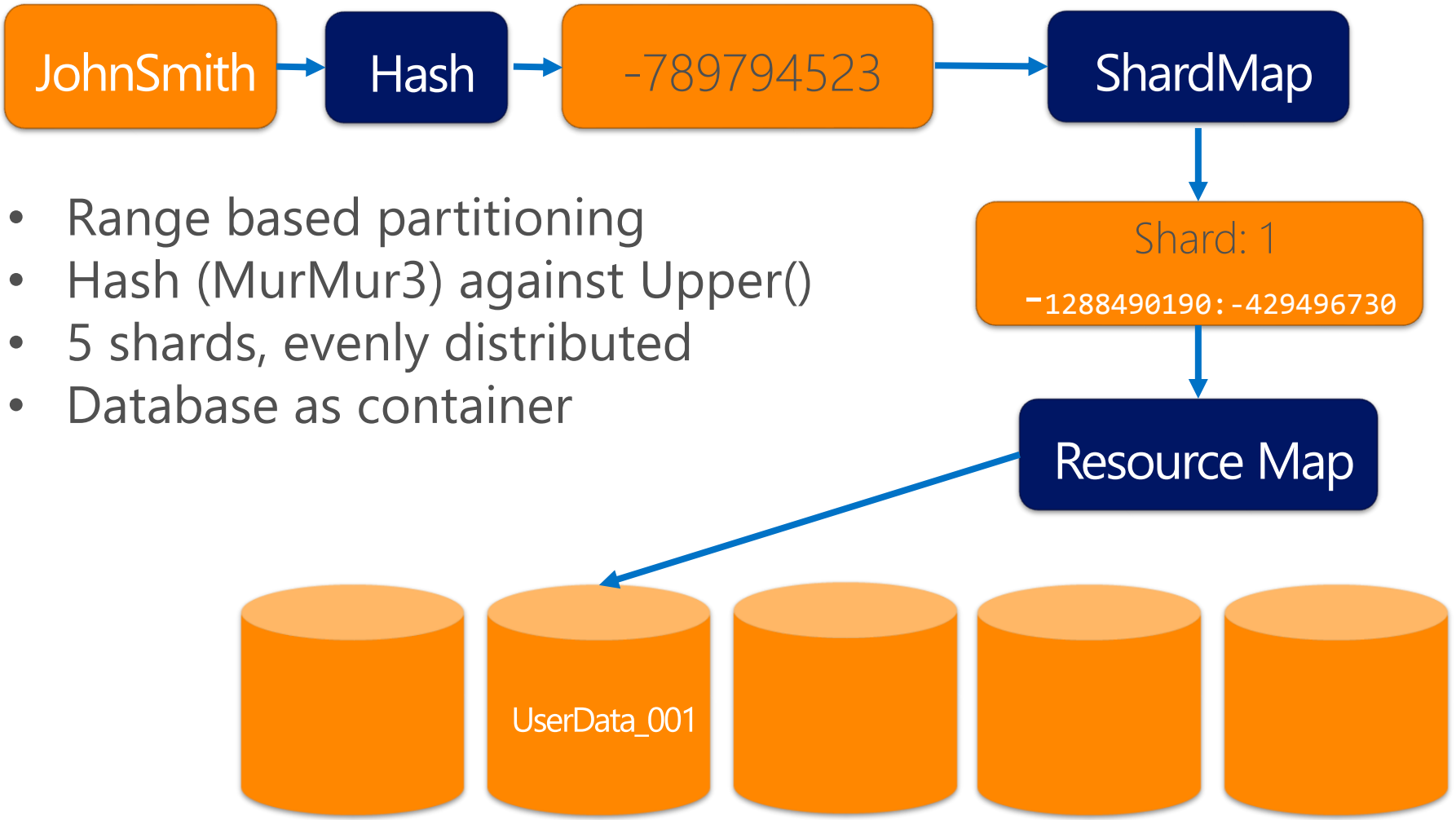
Table as a container ("no container")

Purely logical isolation (tenant / customer ID column)

Pros: high density

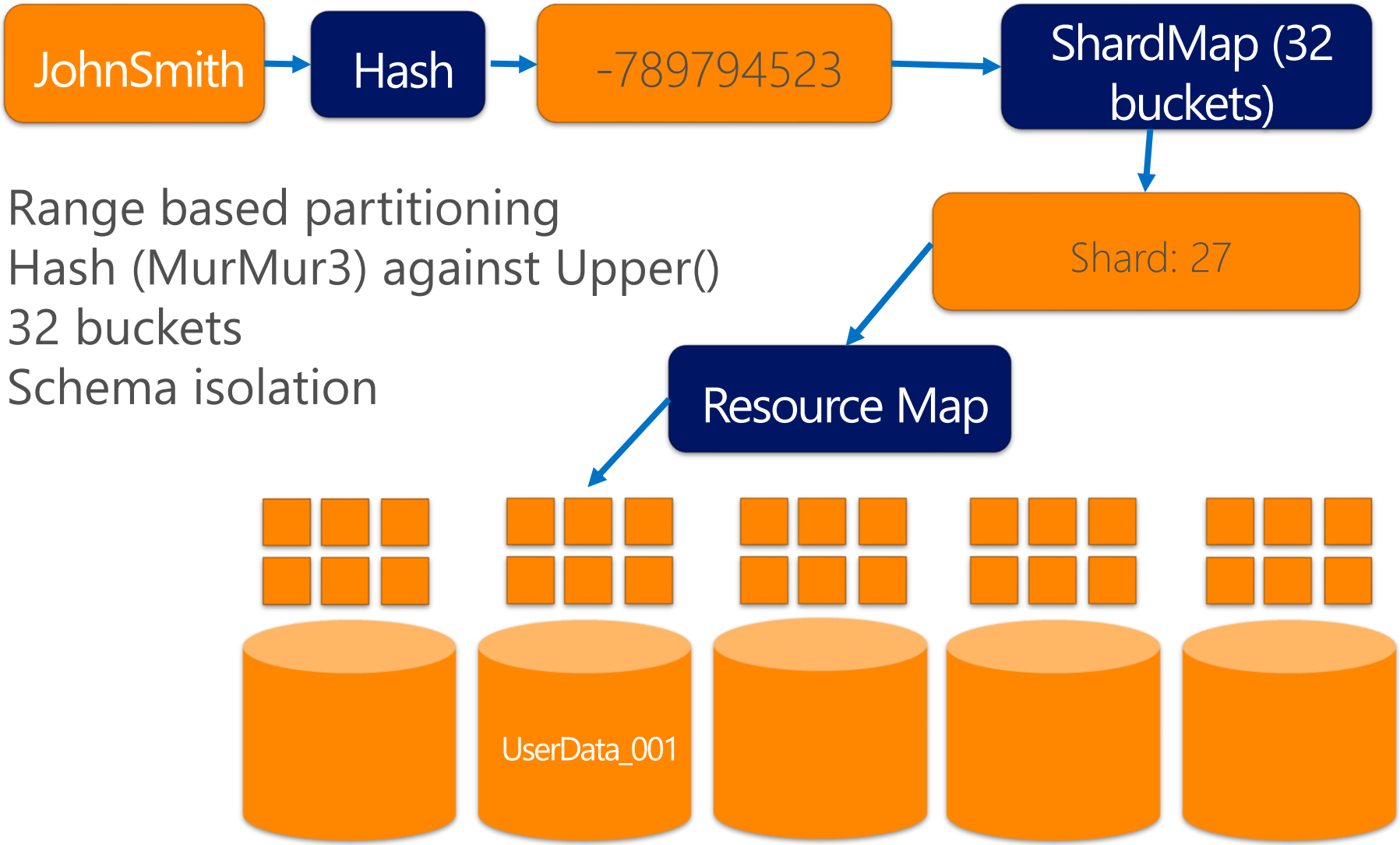
Cons: data layer mapping (need to associate tenant ID with all per-tenant usage)

Partitioning: Ranges



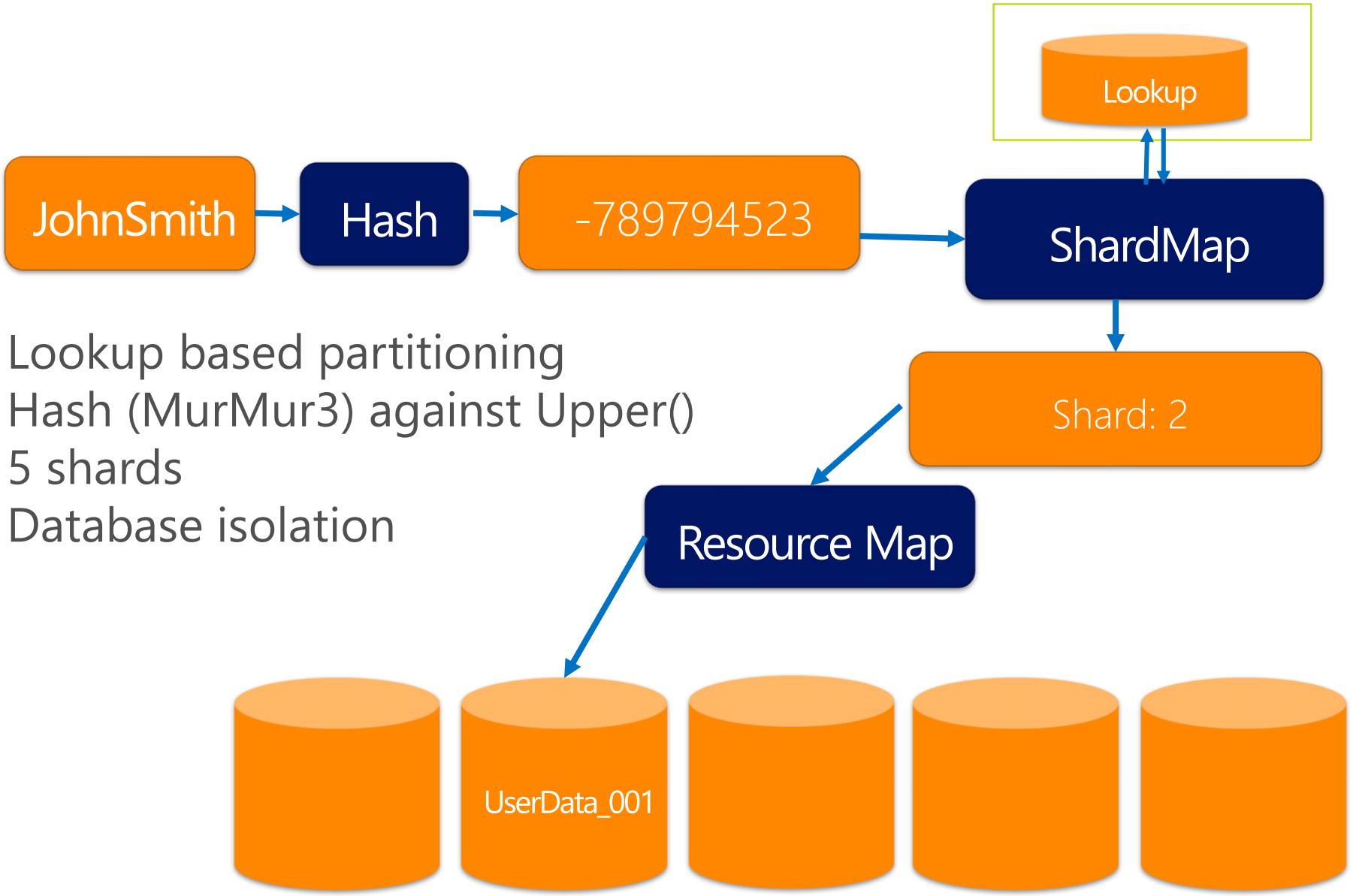
- Range based partitioning
- Hash (MurMur3) against Upper()
- 5 shards, evenly distributed
- Database as container

Partitioning: Logical Bucket



Range based partitioning
Hash (MurMur3) against Upper()
32 buckets
Schema isolation

Partitioning: Lookup (full Map)

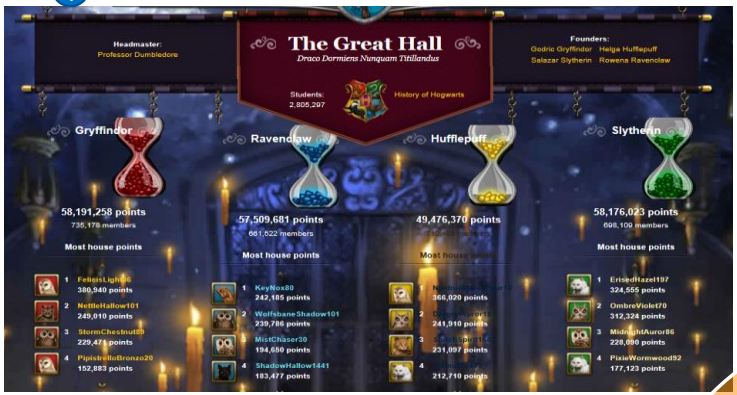


Lookup based partitioning
Hash (MurMur3) against Upper()
5 shards
Database isolation

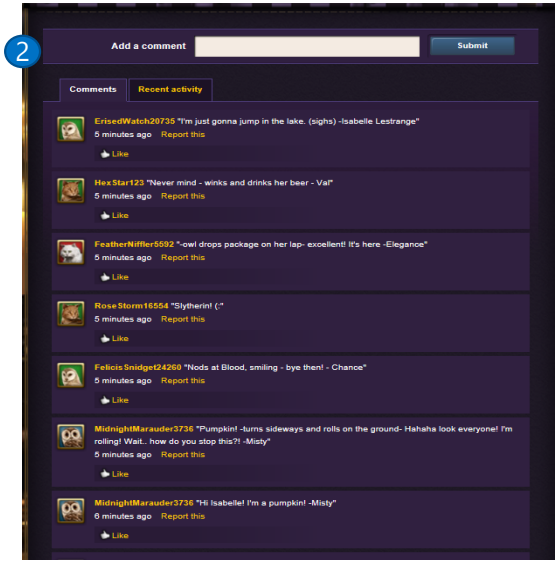
Real World Examples

What is Pottermore?

1



2



Reflect and Share your experience

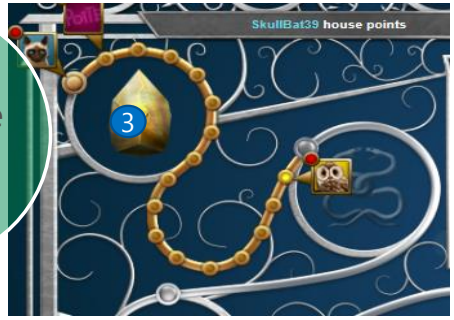
Read, Explore and Re-discover the Story

Buy and give the Story

Play and participate in the Story

4

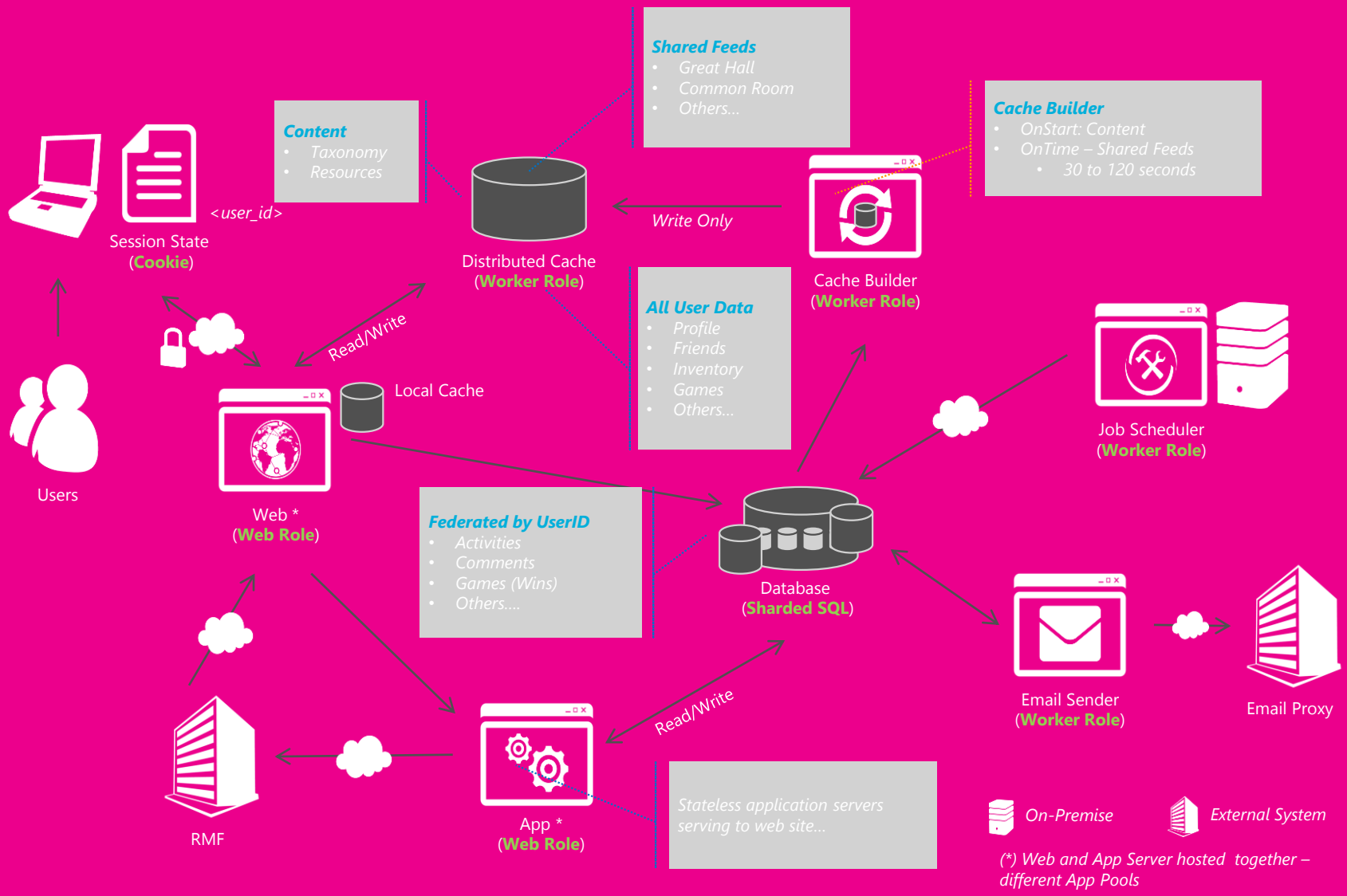
3



Pottermore Background

- Harry Potter Online Experience
- Why Cloud?
  - Original Beta Launch July 2011 on-premises solution
  - 4.2M Page views < 2 minutes
  - 1M Registrations < 10 hours
  - Could not scale easily
- Target User Base
  - Unknown popularity, estimated ~20M users
- Since port to Azure - Open to all
  - 1 billion page views in first 2 weeks
  - ~15K Registrations/day
  - >5M signups
  - Peak ~84K concurrent users, now 25K
  - Silent Launch: on April 14, 2012 and for 2nd book in July, 2012
- Pottermore = Very Happy CTO "Overwhelmed by support from Microsoft"

Pottermore Architecture



Pottermore - Lessons Learned

- Azure Platform
  - Enables scalability of Azure services through online provisioning
  - High Availability through application partitioning
  - Compose services for scale-out
  - Caching to enable fast response & resiliency
- Telemetry
  - Leverage the platform Windows Azure Diagnostics
  - Service Health: built-in logging and diagnostics roll-up Scale-out SQL Databases
  - Sharding is key
  - Partition data for different uses: User Profile, Activities and Email Tasks

Q&A