Windows Azure

Architecting large scale Azure services

Conor Cunningham
Principal Architect
Azure Data Platform Group

Microsoft

Why Windows Azure SQL Database?

Windows Azure CAT
Customer Advisory Team

Why do customers choose to use WA SQL Database?

Reason	Description
Capacity On-Demand	You can have a database in just a few minutes from the moment you decide you need it.
Automatic Features Pre-Configured	Automatic High-Availability, Patching, Backups, and other labor-intensive features are handled by default – no work required
It's a Service	Microsoft manages it for you and deals with hardware, upgrades, uptime, etc.

WA SQL DB Architectural Implications

- Fixed Size Machines
  - Use Scale-Out , not Scale-Up
- Commodity Hardware
  - Things fail more often, implying unplanned failovers
- Automatic Logic for Backups
- Multi-Tenancy
  - Variable performance when others are busy on same machine
- On-demand capacity
  - Grow and shrink as you need

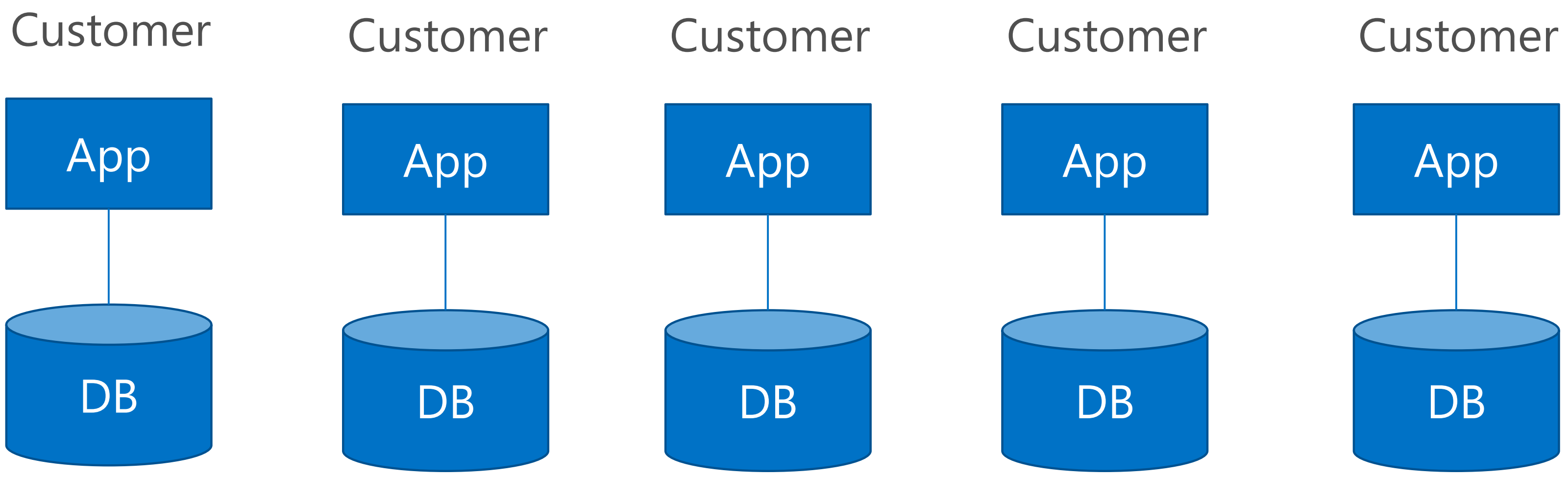
Additional Architectural Implications - Cost

Windows Azure CAT Customer Advisory Team

- Windows Azure has relatively cheap storage
- Often it makes more sense to store blobs or high-volume non-relational/non-transactional data on Windows Azure Storage
- Think through the choices
  - Blobs might just be easier to store in WA Storage
  - Logging data also makes sense
- This implies some differences in the application architecture

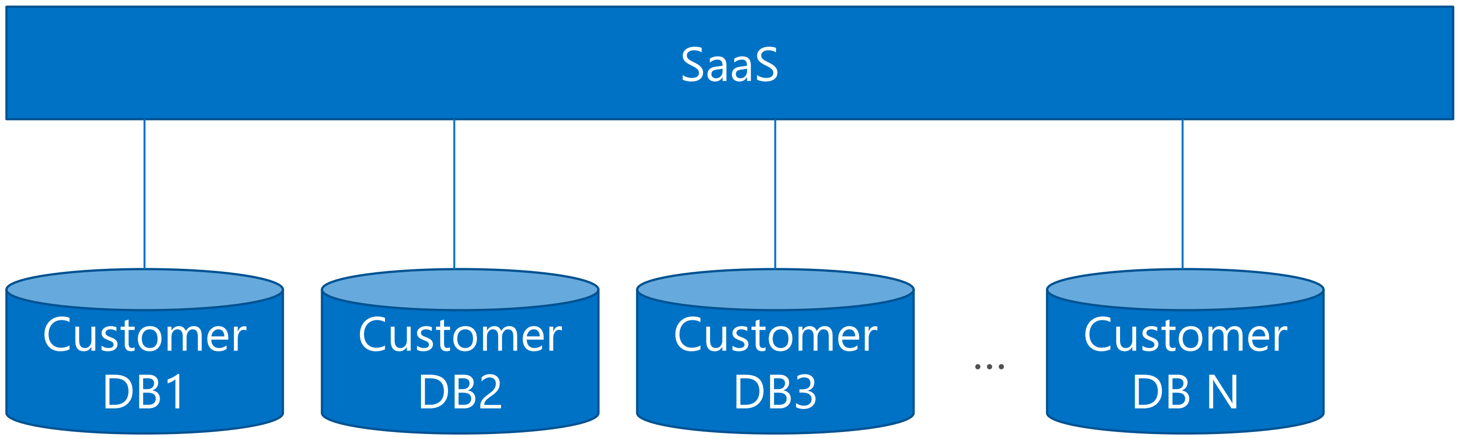
Common Scaling Patterns

- The largest WA SQL Database customers we have today are ISVs
- Build a DB App, sell it to a customer and they run it on their hardware
- Sell lots of copies, make lots of money



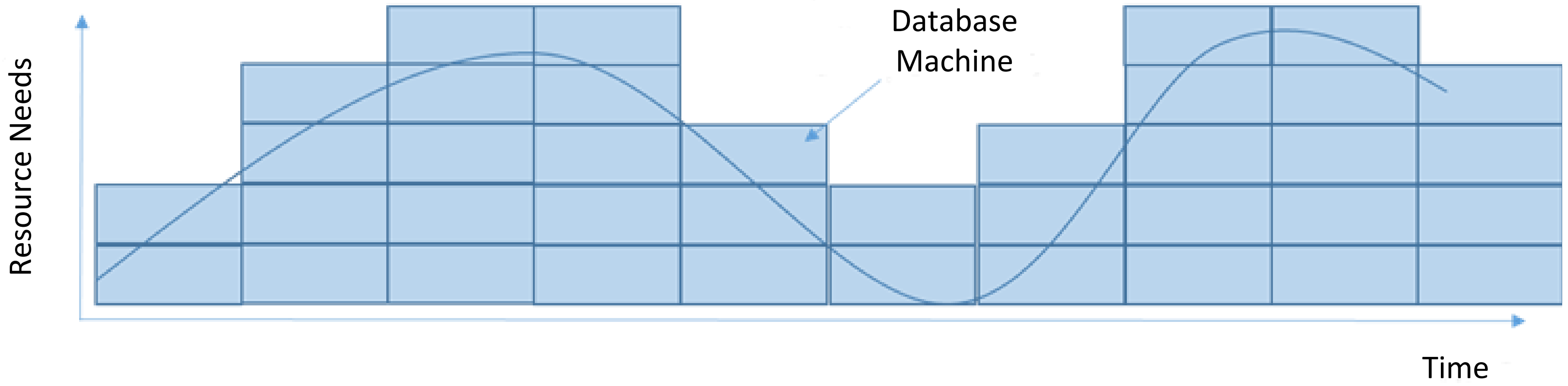
Software as a Service (SaaS) ISVs

- SaaS ISVs run their code as a layer
- Code is usually shared across many customers
- Sometimes databases are shared too
- When they have more customers, they often have more databases



Big Application Architectural Game Plan

- Find a way to get our database application to run over many machines
- Even better – make it adjust based on the resource needs for the database!
- Scale-out is natural in a Web Tier or App Tier
  - The same model can work in a Data Tier with a bit of work
- So - How can we split up a database and spread it over machines?



Data Model Sharding

- Typical OLTP databases look something like this picture
- Everything goes into a single database, but you usually only query for individual customers at a time (example: customer places an order)
- Reports run on the same database or are moved to a secondary replica to avoid contention on locks, resources

Customers

CustID	First Name	Last Name
1	Bob	Smith
2	Larry	Miller
3	Sue	Carter

Orders

OrderID	CustId	Date
1	1	1-Jan-2012
2	1	2-May-2012
3	1	5-May-2012
4	2	3-Feb-2012
5	3	21-Nov-2012
6	3	5-Dec-2012

OrderDetails

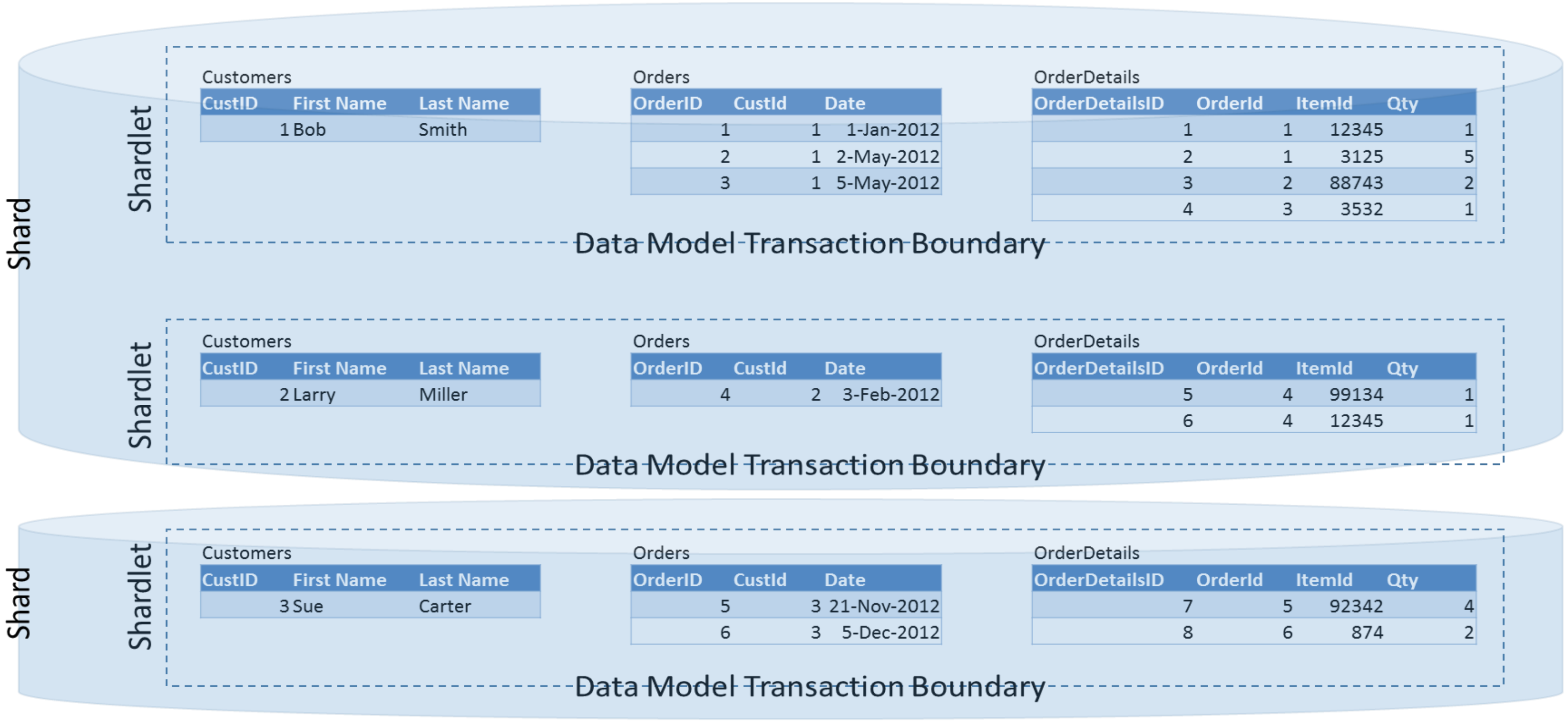
OrderDetailsID	OrderId	ItemId	Qty
1	1	12345	1
2	1	3125	5
3	2	88743	2
4	3	3532	1
5	4	99134	1
6	4	12345	1
7	5	92342	4
8	6	874	2

Data Model Transaction Boundary

Windows Azure CAT
Customer Advisory Team

Sharded Model

- Sharded Models split the data across multiple databases that each have the same schema
- All data about one customer is located within a single database – OLTP operations work fine
- Cross-database operations do not work at all (without manual work)
- Data is automatically spread across many machines in a cluster, not just one



Customers

CustID	First Name	Last Name
1	Bob	Smith

Orders

OrderID	CustId	Date
1	1	1-Jan-2012
2	1	2-May-2012
3	1	5-May-2012

OrderDetails

OrderDetailsID	OrderId	ItemId	Qty
1	1	12345	1
2	1	3125	5
3	2	88743	2
4	3	3532	1

Customers

CustID	First Name	Last Name
2	Larry	Miller

Orders

OrderID	CustId	Date
4	2	3-Feb-2012

OrderDetails

OrderDetailsID	OrderId	ItemId	Qty
5	4	99134	1
6	4	12345	1

Customers

CustID	First Name	Last Name
3	Sue	Carter

Orders

OrderID	CustId	Date
5	3	21-Nov-2012
6	3	5-Dec-2012

OrderDetails

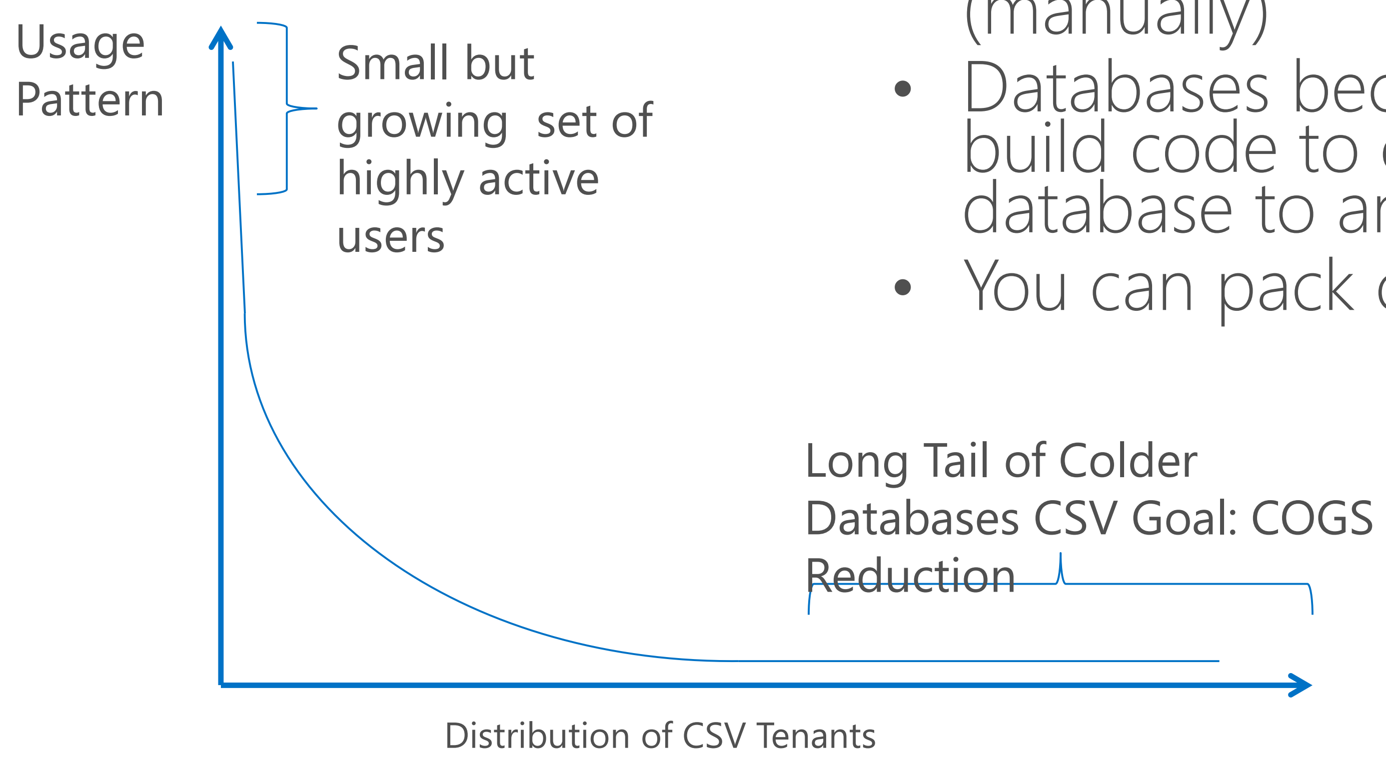
OrderDetailsID	OrderId	ItemId	Qty
7	5	92342	4
8	6	874	2

Capacity Balancing

- SQL Azure will
  - Auto-place different databases on different machines in a cluster
  - Adjust to load over time and move databases around
- We can add/remove machines under the cover as well and this pattern still works just fine
- This capability is difficult to do on normal SQL Server
  - You can try to do readable secondaries, but you still have only one master and that machine must be at least big enough for your write load
  - You can try to move files around, but then you need to set up the HA once you move around databases to adjust for load
- We do this over hundreds of machines and thousands of databases all day, every day.

Multi-Tenancy

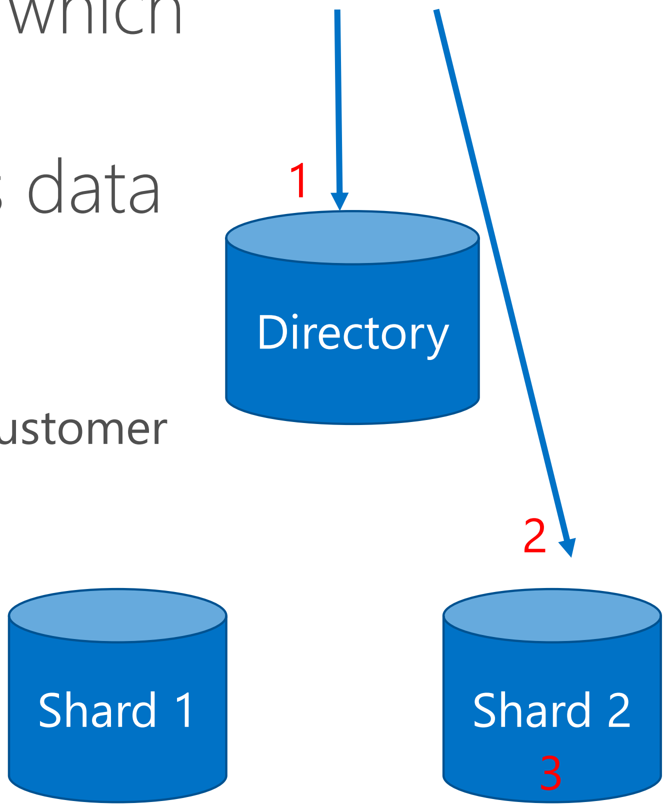
- Single-tenancy case (1 customer = 1 database) is easy
- It is also possible to do multi-tenant cases (manually)
- Databases become containers and you build code to copy customer data from one database to another
- You can pack cold tenants to save money



Central Metadata Databases

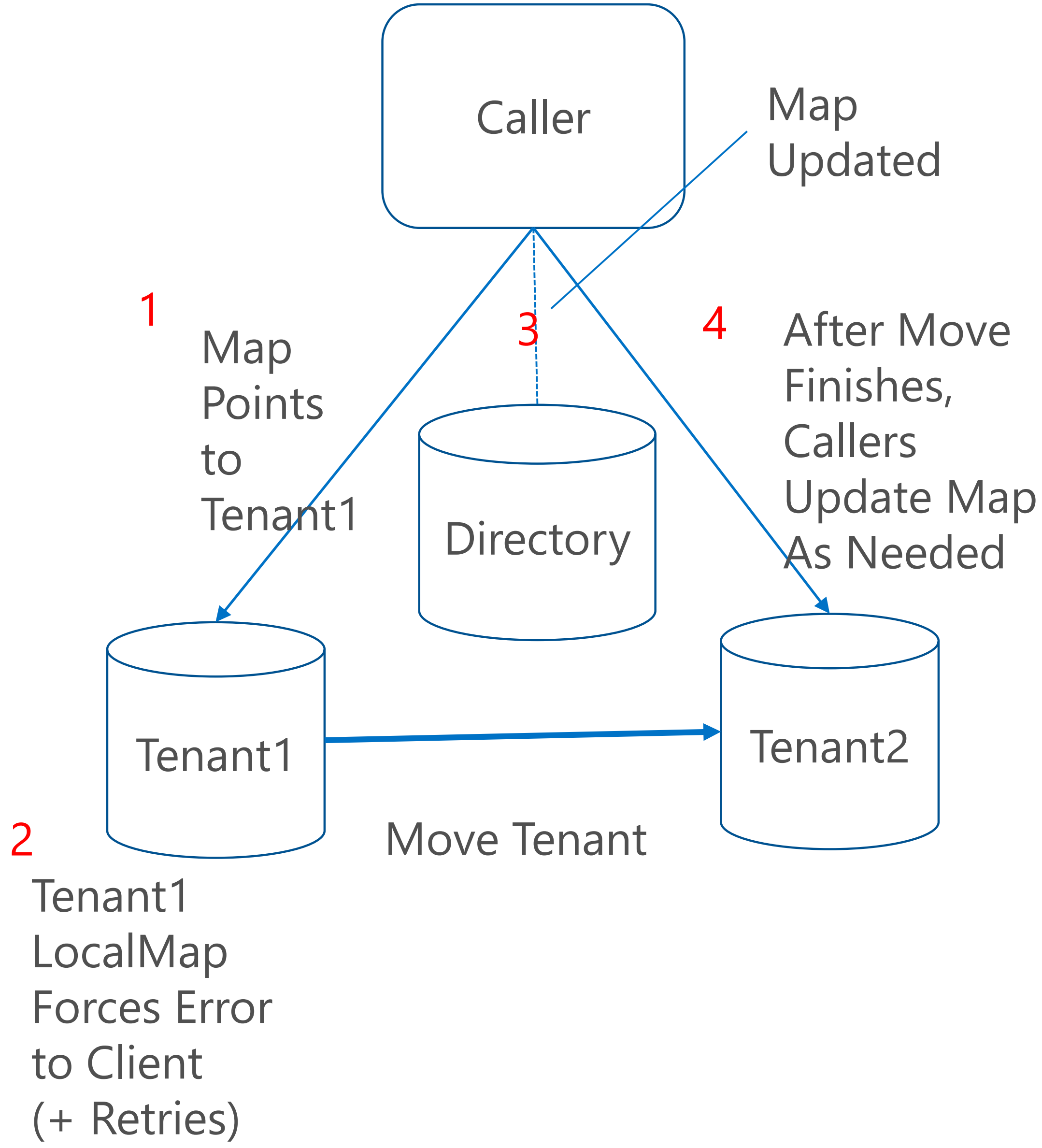
Windows Azure CAT
Customer Advisory Team

- If you have a whole bunch of databases, you need a directory to keep track of which Customers are in each database
- A "Directory" database stores this data
- General Login Path
    1. Connect to Root Database, find tenant
    2. Connect to Right Client Database for this Customer
    3. Perform Work on per-customer data



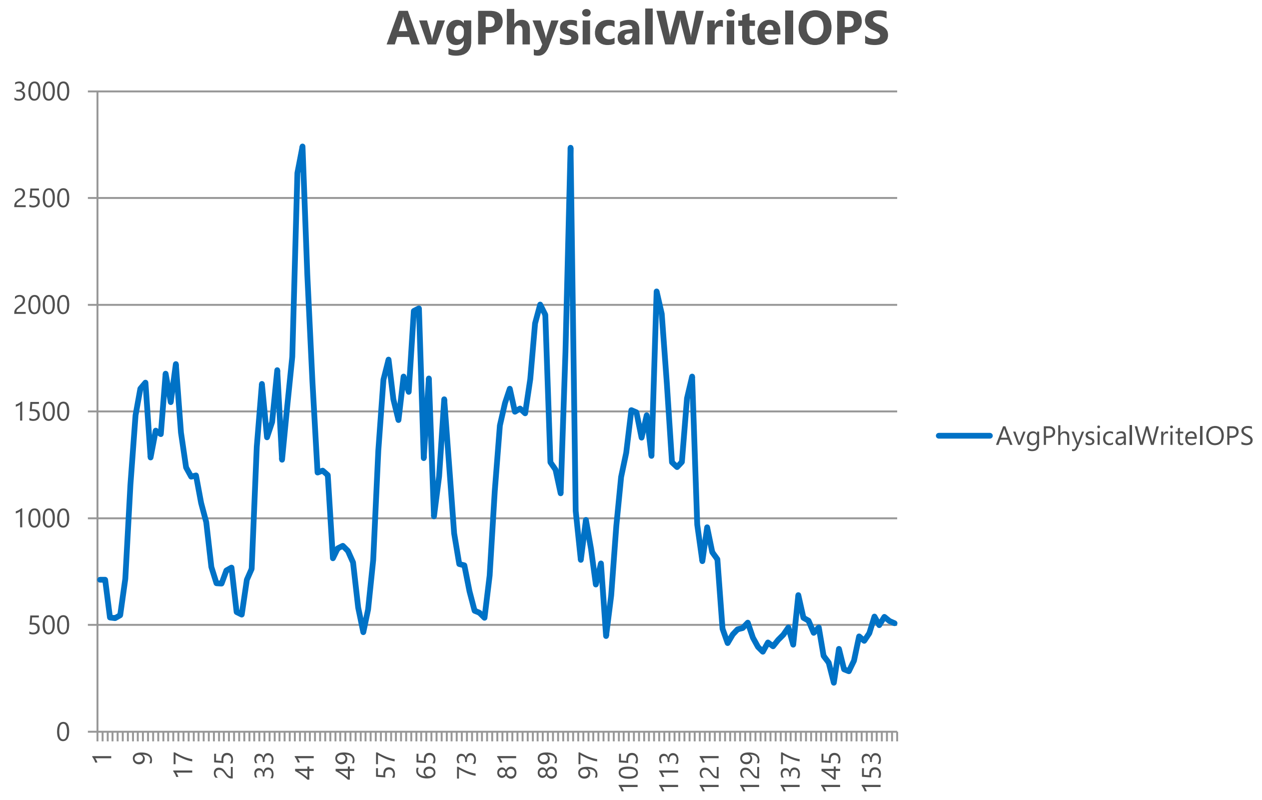
Offline Move Tenant

- You can move tenants to adjust for load/space/capacity in a SaaS ISV
- Mark per-tenant data offline in directory (often both global and local copies)
- Then copy from src->dest
- Then we mark it online again
- Client code must check for availability when connecting each time
- Online Tenant Move is possible but requires careful app planning



Telemetry

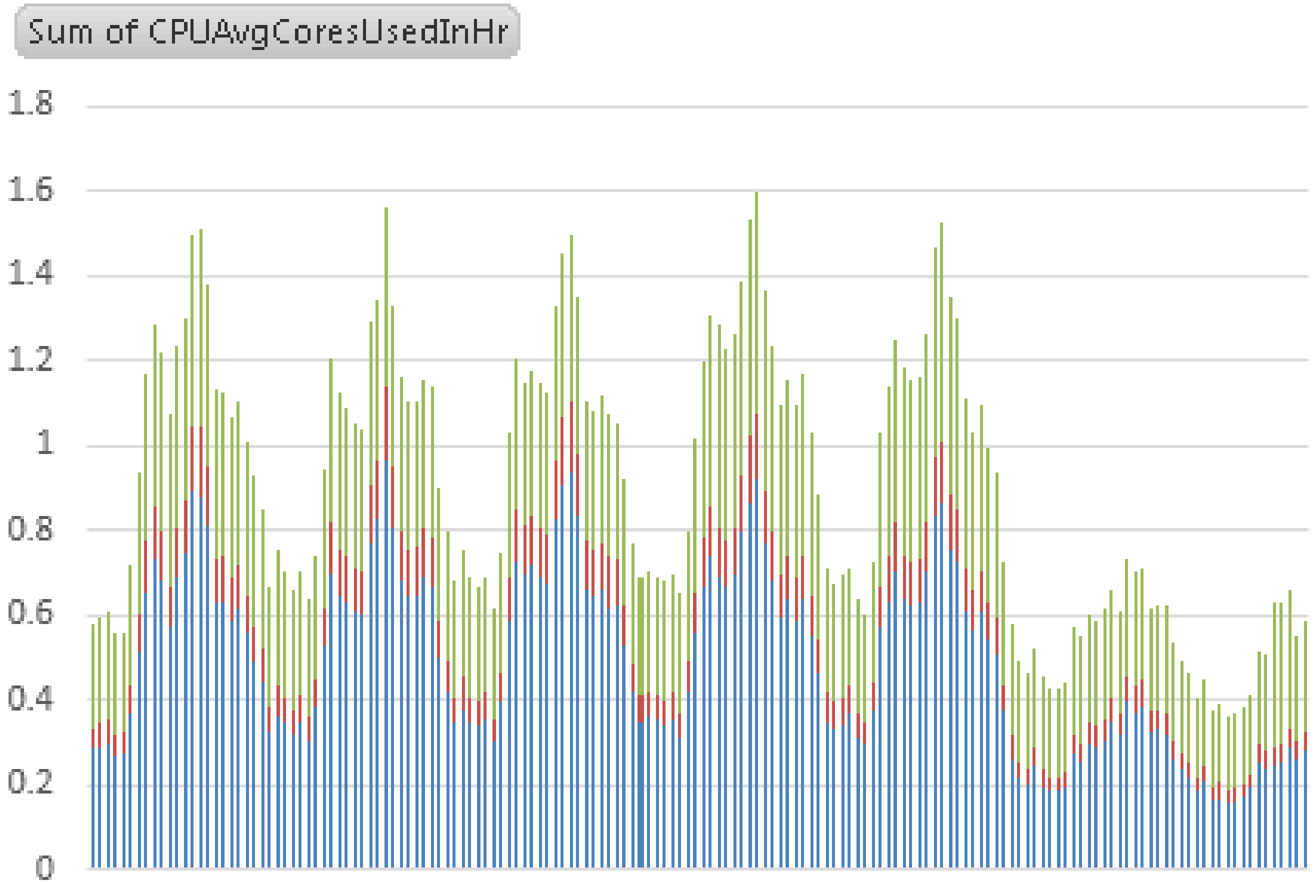
- This is an example from one SaaS ISV
- Weekly data on Physical Writes
- Thousands of Databases
- Regular weekly growth
- We spend time tuning the various jobs to even out the load and avoid spikes



Another Example

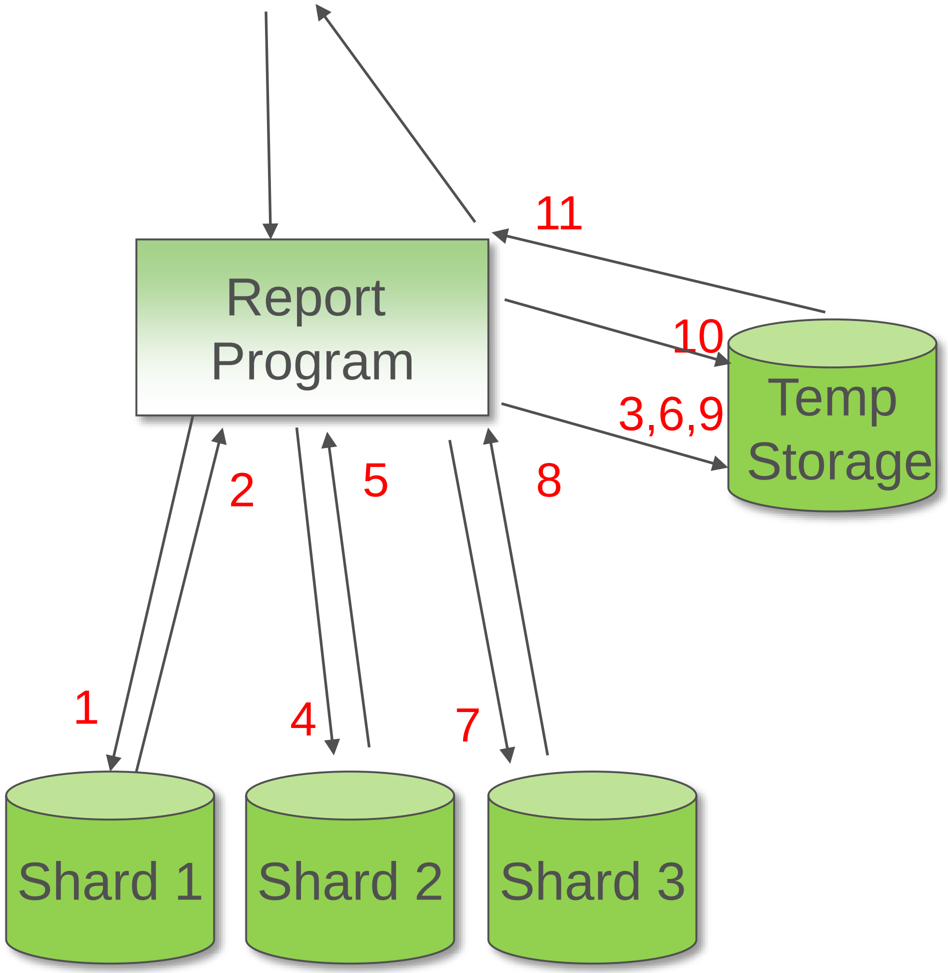
Windows Azure CAT
Customer Advisory Team

- Weekly CPU Data over 3 special databases
- We are helping this customer split out their load from a monolithic single "directory" database
- Once we finish we'll be able to grow their load 10x-100x without further changes



Reporting "Queries"

- How do you run a monthly report over all customers?
  - Iterate over each DB
  - Collect intermediate results in a single database
  - Finish query over intermediate results
- Key Details
  - Not transactionally consistent
  - Intermediate results needs to fit in one database (150GB limit)
  - Some operations may fail; re-run pieces that fail
  - On huge systems, it can take hours to run



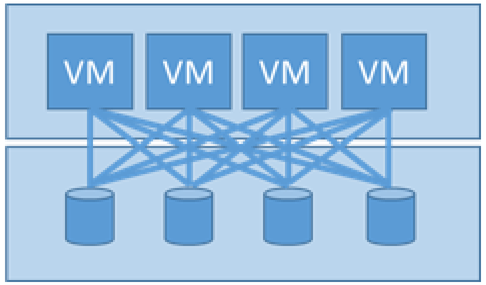
SaaS ISV Scaling

Windows Azure CAT
Customer Advisory Team

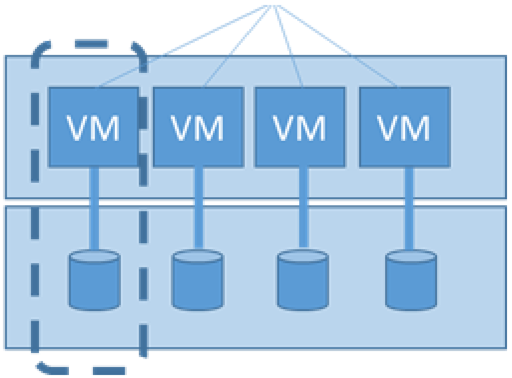
- So, what happens when you keep adding connections to a regular SQL Server instance???
  - Eventually you hit the 32K sessions cap per SQL Server instance
- WA SQL Database limits connections at far lower levels (hundreds)
- WA lets you scale each service somewhat independently
- This usually works, but it has some trouble with state-full services

State-full Scale-Out Applications on Azure

- When you Scale-Out Multiple Tiers with SQL,..
  - You can cause connections to grow at N*M instead of linearly



- Scale-Out Should *Align* Data Access Across Tiers to avoid this issue
  Front-End Routing Web Role that understands partitioning
  Separate App Tier Deployments
  Affinity with Databases



Availability

Windows Azure CAT
Customer Advisory Team

- WA SQL DB SLA is 3 "9s" or 99.9% availability of each database
- WASD also performs automatic mechanisms that fail over databases if
  - Load is too high on one machine (load balancing across cluster)
  - A machine dies
  - We are rolling out a patch to the service
- These usually manifest as small outages when SQL fails over from one replica to another
- Like SQL Server failovers, this can cause small outages (several seconds, usually) as we move the database from one copy to another as primary

Programming for Availability Issues

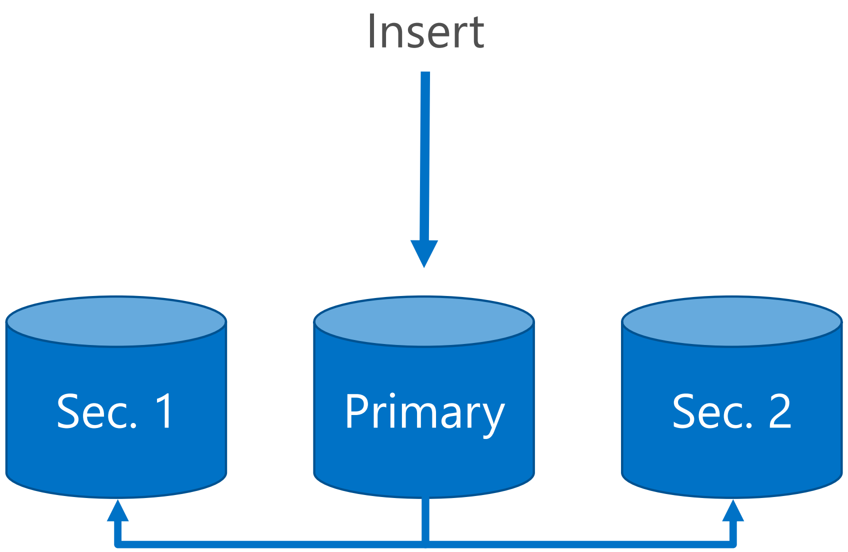
Windows Azure CAT
Customer Advisory Team

- Basic Guidance
  - Make each operation atomic (avoid lots of session state) so you can reconnect and continue
  - Make 1 Batch == 1 Transaction
  - Understand Idempotency Needs

Disaster Recovery

Windows Azure CAT
Customer Advisory Team

- WA SQL Database maintains 3 copies of each database in the system
- When a database is changed, the transaction commits only if a quorum "ack"s the change
- This protects you against individual machine failures and rack failures
- It does not protect you against
  - Data Center failures
  - Yourself ("Oops" scenarios)
- Failover times are usually well under 30 seconds



Changes automatically pushed before transaction commit completes

Throttling

- WA SQL DB has defined limits on various resources you can use (connections, worker threads, memory, etc.)
  - These are smaller than a dedicated SQL Server Instance (service is multi-tenant)
- WA SQL DB also defines min and max capabilities for each resource
  - You are guaranteed to get the minimum
  - You are guaranteed not to get more than the maximum
- We also measure the average experience and are improving user isolation over time (in each service release, we improve performance isolation)
- There are times when everyone on a node gets busy at once…
- WA SQL Database will sometimes throttle connections until it can rebalance the system across the cluster
  - Soft throttling will just tell you to retry
  - Hard throttling will kill your connection and force you to reconnect
  - More details in "Windows Azure SQL Database Performance and Elasticity Guide"

Designing for Unavailability

- Program “defensively” to allow for outages
- Cache data more aggressively instead of always calling DB
- Especially important for key databases (Central Metadata DB)
- Separate each part of Internet-facing Services
  - Core OLTP separated from Billing and Account Management and …
  - This allows each part to be offline separately
  - It also allows for you to upgrade each part separately
- Key Lesson: At larger scales, everything fails – you can determine how it fails in your design to minimize customer pain

Telemetry Pipelines/Dashboards

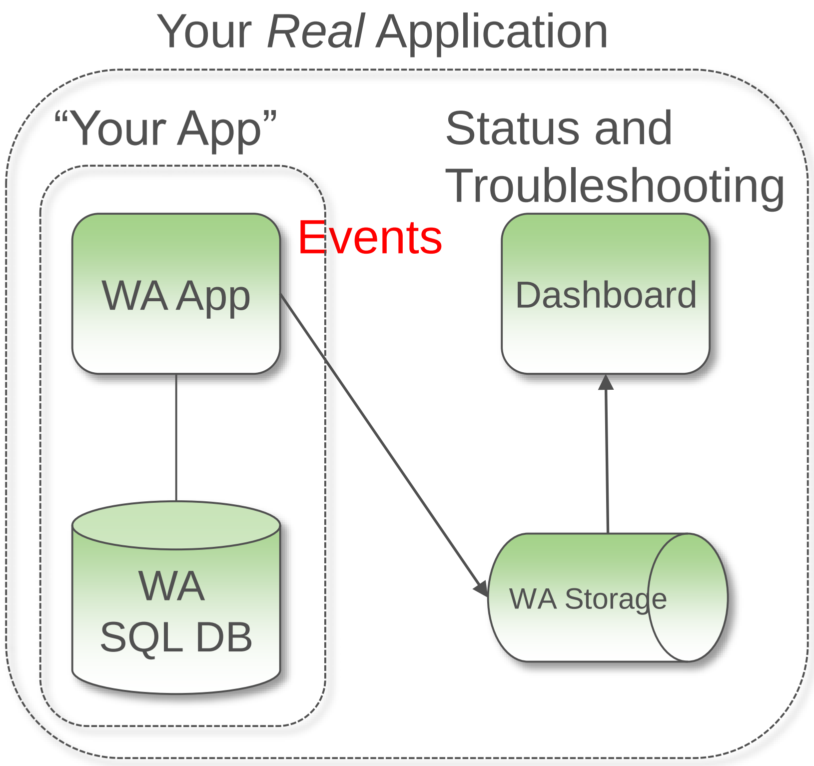
- Traditional SQL Server debugging is a hands-on affair
  - Ad-hoc tracing
  - Querying DMVs
  - Performance Counters
  - …
- With 1 Server, this is ok
  - With 5 Servers, it starts to be painful
  - With thousands of servers, it is impossible
- Debugging in Windows Azure is driven by Logging systems
- It is also critical to separate Time-To-Resolution from Root-Cause-Analysis
  - Engineers like RCA
  - Customers like Time-to-Resolution first

Telemetry Pipelines/Dashboards

- Traditional SQL Server debugging is a hands-on affair
  - Ad-hoc tracing
  - Querying DMVs
  - Performance Counters
  - …
- With 1 Server, this is ok
  - With 5 Servers, it starts to be painful
  - With thousands of servers, it is impossible
- Debugging in Windows Azure is driven by Logging systems
- It is also critical to separate Time-To-Resolution from Root-Cause-Analysis
  - Engineers like RCA
  - Customers like Time-to-Resolution first

Common Telemetry Architectures

- Most "Big" Apps write events from each scaled-out instance
- Multi-Tenant Systems can not be taken "offline" to do debugging
- WA Blob/Table Storage gives higher write throughput
- Automated Log Processing used to
  - Visualize Trends
  - Spot common patterns
  - Build Alerting Systems
- Goal: Continuously automate common patterns + solutions, avoid labor costs, achieve massive scales



Upgrades?

Windows Azure CAT
Customer Advisory Team

- Downtime is not usually an option for most services
- Upgrades should be split into small pieces
- Example
  - Roll out schema changes first (no code behavior changes)
  - Then roll out side-by-side stored procedures (old, new)
  - Roll out Application Tier changes (draining old, starting to use new)
  - Eventually, remove “old” stored procedures in clean up step
- Functionally Partition Service into different components
  - Do upgrades for each service independently (if needed)
- Primary Goal: Avoid ALL downtime

Conclusion

- We have customers running at large scales (thousands to tens of thousands of databases) across multiple geographies
  - Thousands to hundreds of thousands of end-users.
- This pattern enables building Internet-facing web sites and services
  - …that can scale almost arbitrarily
- Over time, we hope to make it easier and faster for customers to make applications in this model.

Windows Azure

Windows Azure CAT
Customer Advisory Team

Designing data tiers for the cloud

Silvano Coriani
Principal Program Manager
Windows Azure Customer Advisory Team

Microsoft

Agenda

- Windows Azure Storage Options
  - Windows Azure Storage
  - SQL Server in Azure VMs
  - Azure SQL DB
- Azure SQL DB Performance/Scaling
  - Sizing databases
  - Cost Optimization
  - Techniques to minimize peak load
- Functional Partitioning
- Sharding Trigger Points
- Application Tier Caching

Azure Storage Options

Platform as a Service

- Azure SQL Database (managed databases)

- It's not SQL Server!
- Publish and run
- Shared environment

Infrastructure as a Service

- SQL Server running in a Windows Azure VM

- It's SQL Server!
- Full control / insight
- More administrative effort

Azure Storage

- Tables
- Blobs
- Queues

- No relational
- Cheap storage
- Optimized for density

Azure Storage Tables

Windows Azure CAT
Customer Advisory Team

Schema-less / NoSQL abstraction on Azure Storage (video – slides - paper)
No relational capabilities, limited query-ability
Works great for append-only workloads, range (partition key) lookups

Density
• Use appropriate partition keys to co-locate data
• Use appropriate partition keys to break data up into more partitions
• Implement retry logic with back-off for 503 (service unavailable) errors
• Avoid use of table storage for applications requiring non-trivial aggregation or function projection
Scale
• Leverage partitioning multiple storage accounts (not multiple tables) to increase operations/second

Table Limits	Limit
Max operations / second per partition	2,000
Max row size (names + data)	1 MB
Max column size (byte[] or string)	64 KB
Maximum number of rows	N/A (up to storage account size limit)
Scale-out unit	Table partition
Scale-out impedance	Low

SQL Server in Windows Azure VMs

Windows Azure CAT
Customer Advisory Team

More transparency, control and tuning options

- Application compatibility, works with any traditional workload
- Storage layout flexibility, trace flags, etc.
- Windows Azure Blobs exposed as NTFS volumes

Multiple VM sizes available

- Up to 8 cores, 56 Gb of RAM and up to 16 Data Disks (up to 1TB each, 16 x 500 IOPS)

Can be deployed in HA/DR configurations leveraging AlwaysON

- Require a proper maintenance plan and cloud ‘infrastructure design’ (no single VM deployment in production)

Can scale up to the biggest size available

- Require adequate planning for scale out scenario (can leveage AlwaysON readable secondaries)
- Require additional efforts if full elasticity is needed (e.g. optimize running costs based on peaks management)
- With careful design (DIP communication) can minimize latency, but still exceeds co-located on-premise

Azure SQL Database

Windows Azure CAT
Customer Advisory Team

Multi-tenant environment with shared resources

- Worker threads, Memory, Log, IO subsystem
- Resource governance
- HA configuration out of the box (3-copies replica set)
- Self provisioning, limited maintenance required

Works great with OLTP workloads

- Point lookup, fully relational
- Less for DW and analytical queries, no parallelism
- Cloud client access approaches (batch, minimize round trips) to improve latency and avoid resource throttling

Scale out approach (not just db size but computing resources)

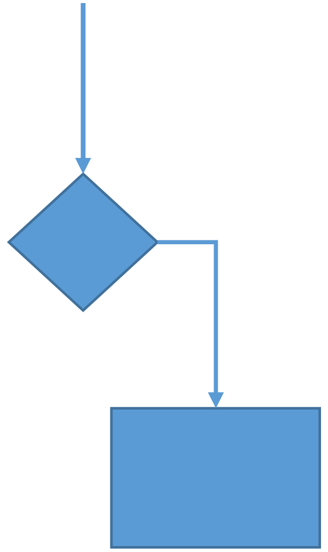
- Adding more databases gets you more performance
- Microsoft balances load over time

Soft and hard throttling to protect system stability

- Requires retry logic and restartability

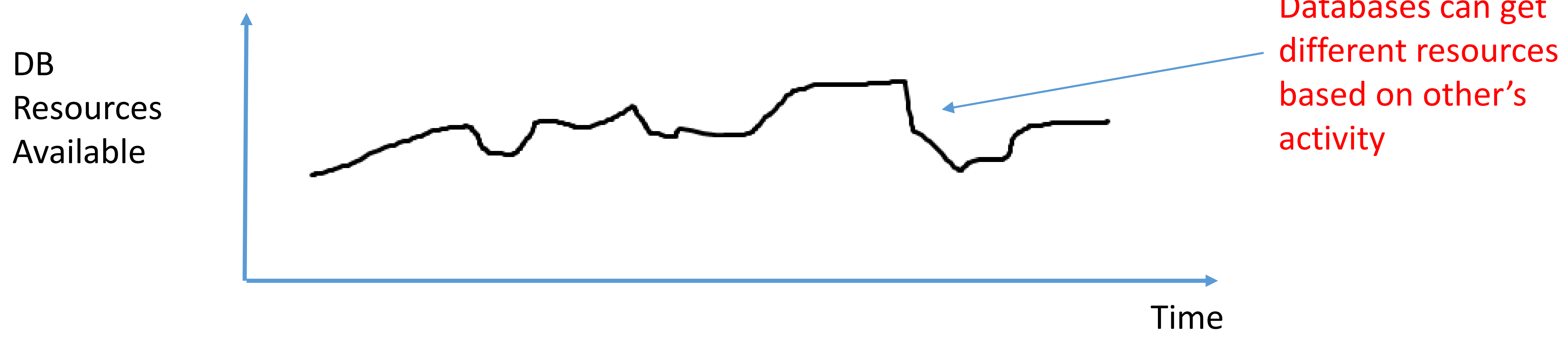
Decision Points

- Common Data going to WA Storage
  - Telemetry Logs
  - Blobs for WA SQL DB (lower costs, reduce DB size under 150GB limit)
  - Things where cost is the driving factor
- Commonly going to SQL Server in VM
  - Existing SQL Server applications (they get hosting)
  - Applications needing more performance (IOPS usually) – light DW workloads
  - Applications needing features not in SQL DB (example: Fulltext)
- Commonly going to SQL DB
  - Applications who do not want to manage their databases
  - Applications that need massive horizontal scale (Internet-facing SaaS ISVs)
  - SQL Server applications where the customer is willing to do some rearchitecting



SQL DB Web/Business Performance Variance

- Web/Business Editions provide no performance guarantees
- We host hundreds of customers on a single machine
- Performance of your DB will vary based on what others are doing
- SQL DB contains logic to move DBs around to balance load across each cluster to maximize average resources



Premium Edition

- Not all applications work well in shared environments
- Premium Edition was introduced for customers who need dedicated resources.
- Common customer attributes:
  - High throughput requirements
  - Low latency requirements
  - Low performance variance requirements
- Premium Edition details
  - Dedicated resources (min=max) to avoid performance variance
  - Different sizes (P1-P2) allow adjustment based on resource needs
  - Currently in Public Preview

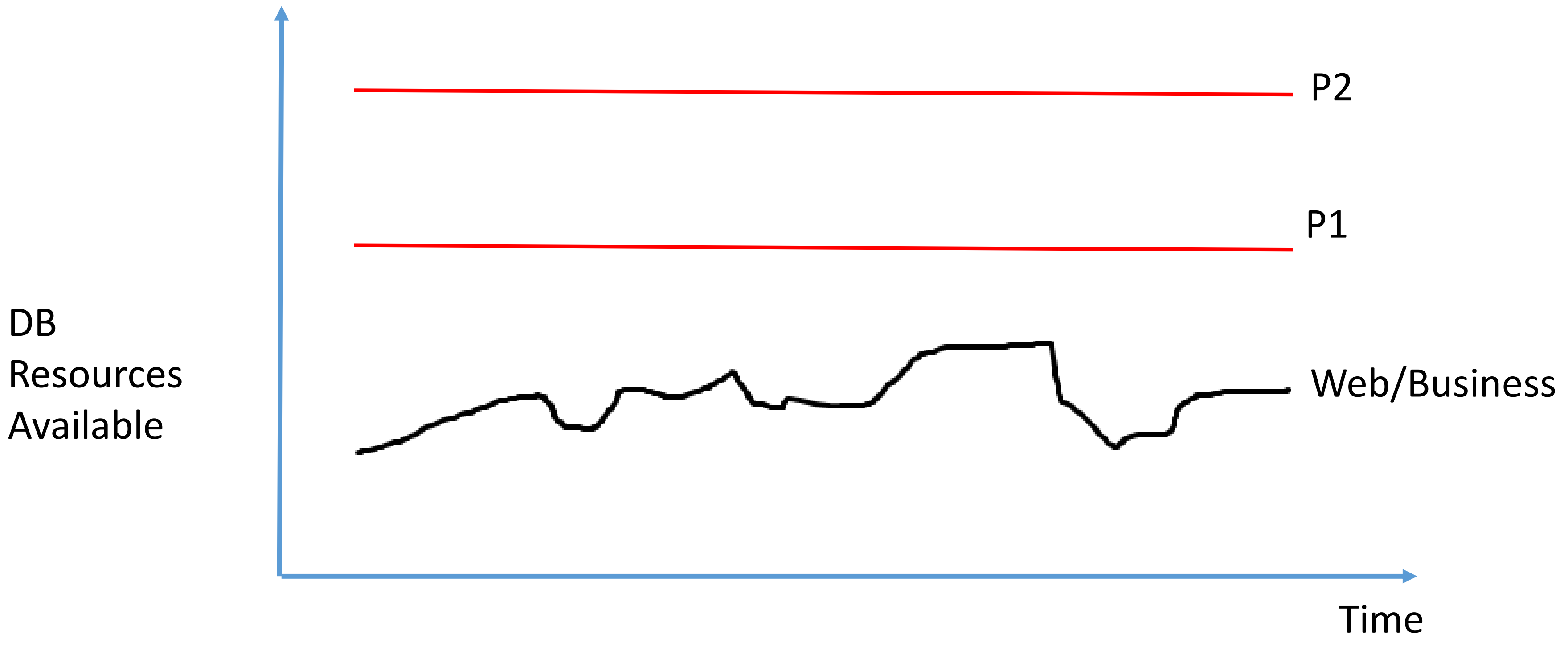
Premium Edition Reservation Sizes

- Reservations are done separately for each database
  - Capacity is limited during public preview
  - Customers can get 1-2 reservations based on availability
- Monthly Price is USD $930 for P1 at GA. P2 is 2x

Size	CPU Cores	Worker Threads	Active Sessions	Disk IO (IOPS)	Memory (GB)
P1	1	200	2000	150	8
P2	2	400	4000	300	16

Edition Comparison

- Premium has reserved resources
- You can upgrade or downgrade a database
- You should decide sizing based on your resource needs



Scale up comes with a cost

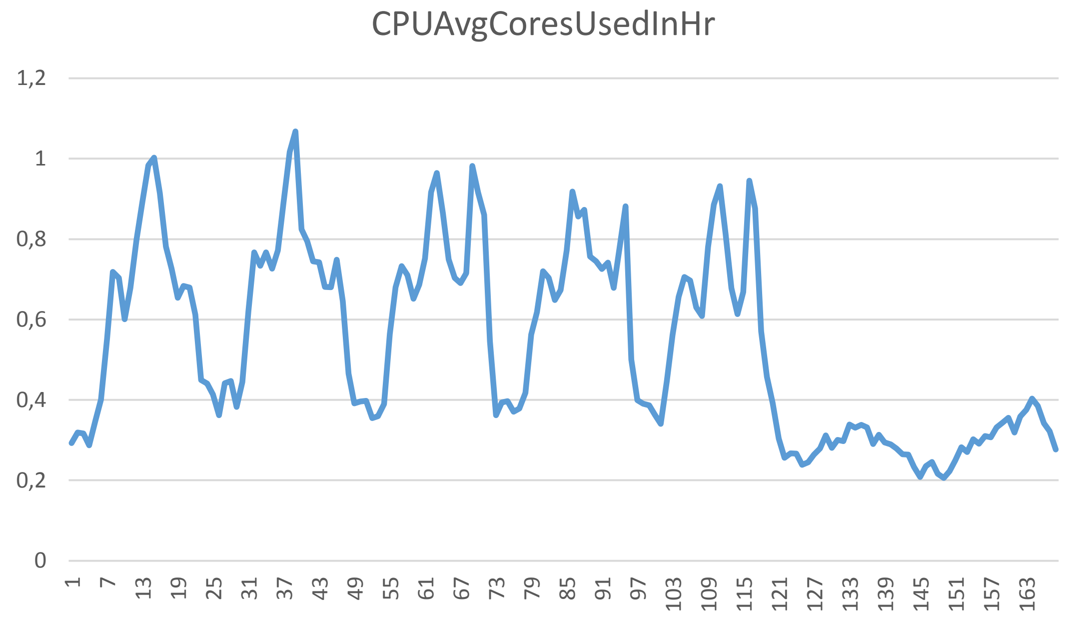
Windows Azure CAT
Customer Advisory Team

SQL Premium DB Size	SQL Premium GA Monthly Cost	SQL VM Monthly Cost	SQL VM Size (Enterprise Edition)
P1 (M) 1 CPU Core 8GB RAM 150 IOPS	$930	$1,629	S (A1) 1 CPU Core 1.75GB RAM 2x500 IOPS
P2 (L) 2 CPU Cores 16GB RAM 300 IOPS	$1860	$1696	M (A2) 2 CPU Cores 3.5GB RAM 4x500 IOPS
		$1,830	L (A3) 4 CPU Cores 7GB RAM 8x500 IOPS
		$2,321	A6 4 CPU Cores 28GB RAM 8x500 IOPS
		$3,660	XL (A4) 8 CPU Cores 14GB RAM 16x500 IOPS
		$4,642	A7 8 CPU Cores 56GB RAM 16x500 IOPS

- SQL DB resources are dedicated to that database workload only, SQL VM resources will be used to run the entire VM (OS + SQL + all databases)
- A single SQL VM does not provide any availability SLA. You'll need 2 instances and AlwaysOn
- Premium will offer more than just performance predictability

Sizing Databases

- For a SINGLE database…
  - Find largest resource consumer
  - Measure peak load over time period
  - Choose appropriate reservation size to handle peak load
- Workload Type matters
  - Batch processing – aim to achieve avg throughput over time (not size for peak)
  - Interactive applications need to size for the peak to preserve response times



Capacity planning

- Use *sys.resource_stats* in master db to determine your application resource needs:

```sql
SELECT * FROM
sys.resource_stats
WHERE  database_name =
'MyTestDB' AND start_time >
DATEADD(day, -7, GETDATE())
```

	start_time	end_time	database_name	sku	usage_in_seconds	storage_in_megabytes	avg_cpu_cores_used	avg_physical_read_iops	avg_physical_write_iops	active_memory_used_kb	active_session_count	active_worker_count
1	2013-09-27 09:10:00.00	2013-09-27 09:15:00.00	MyTestDB	Web	0	1.57	0.0000000	0.0000	0.0867	0	0	0
2	2013-09-27 09:15:00.00	2013-09-27 09:20:00.00	MyTestDB	Web	0	1.57	0.0005733	0.0033	0.0033	0	2	0
3	2013-09-27 09:20:00.00	2013-09-27 09:25:00.00	MyTestDB	Web	0	1.59	0.0000000	0.0000	0.0033	0	2	0
4	2013-09-27 09:25:00.00	2013-09-27 09:30:00.00	MyTestDB	Web	0	1.59	0.0000000	0.0000	0.0033	0	2	0
5	2013-09-27 09:30:00.00	2013-09-27 09:35:00.00	MyTestDB	Web	25	9.46	0.0844000	0.0267	1.1933	0	102	98
6	2013-09-27 09:35:00.00	2013-09-27 09:40:00.00	MyTestDB	Web	23	9.46	0.0789833	0.0000	1.1933	0	102	100
7	2013-09-27 09:40:00.00	2013-09-27 09:45:00.00	MyTestDB	Web	7	20.15	0.0245567	0.0000	0.4600	0	2	0
8	2013-09-27 09:45:00.00	2013-09-27 09:50:00.00	MyTestDB	Web	0	20.15	0.0000000	0.0000	0.0033	0	1	0

Windows Azure CAT
Customer Advisory Team

Investigating resource usage

Avg and Max resource usage

```
SELECT
    avg(avg_cpu_cores_used) AS 'Average CPU Cores Used',
    max(avg_cpu_cores_used) AS 'Maximum CPU Cores Used',
    avg(avg_physical_read_iops + avg_physical_write_iops)
AS 'Average Physical IOPS',
    max(avg_physical_read_iops + avg_physical_write_iops)
AS 'Maximum Physical IOPS',
    avg(active_memory_used_kb / (1024.0 * 1024.0)) AS
'Average Memory Used in GB',
    max(active_memory_used_kb / (1024.0 * 1024.0)) AS
'Maximum Memory Used in GB',
    avg(active_session_count) AS 'Average # of Sessions',
    max(active_session_count) AS 'Maximum # of Sessions',
    avg(active_worker_count) AS 'Average # of Workers',
    max(active_worker_count) AS 'Maximum # of Workers'
FROM sys.resource_stats
WHERE  database_name = 'MyTestDB' AND start_time >
DATEADD(day, -7, GETDATE())
```

Percentage of time using more than 1 core

```
SELECT
(SELECT
    SUM(DATEDIFF(minute, start_time, end_time))
    FROM sys.resource_stats
    WHERE database_name = 'MyTestDB' AND
          start_time > DATEADD(day, -7, GETDATE()) AND
          avg_cpu_cores_used > 1.0) * 1.0 /
SUM(DATEDIFF(minute, start_time, end_time)
) AS percenage_more_than_1_core
FROM sys.resource_stats
WHERE database_name = 'MyTestDB' AND start_time > DATEADD(day,
-7, GETDATE())
```

	percenage_more_than_1_core
1	NULL

	Average CPU Cores Used	Maximum CPU Cores Used	Average Physical IOPS	Maximum Physical IOPS	Average Memory Used in GB	Maximum Memory Used in GB	Average # of Sessions	Maximum # of Sessions	Average # of Workers	Maximum # of Workers
1	0.0235641	0.0844000	0.372062	1.2200	0.000000000000	0.000000000000	26	102	24	100

Cost Optimization

- Two paths to improve your cloud service
  - Spend more money (purchase more capacity)
  - Optimize/Tune (more operations in capacity you have)
- The Cloud model let you choose
  - If you have development resources available, you might choose to ‘tune’
  - If you are on a time deadline, you might just choose to spend more instead
- This model also works great for seasonal demand changes
  - Example: Add capacity before the holiday sales season, remove after

Managing Peak Load

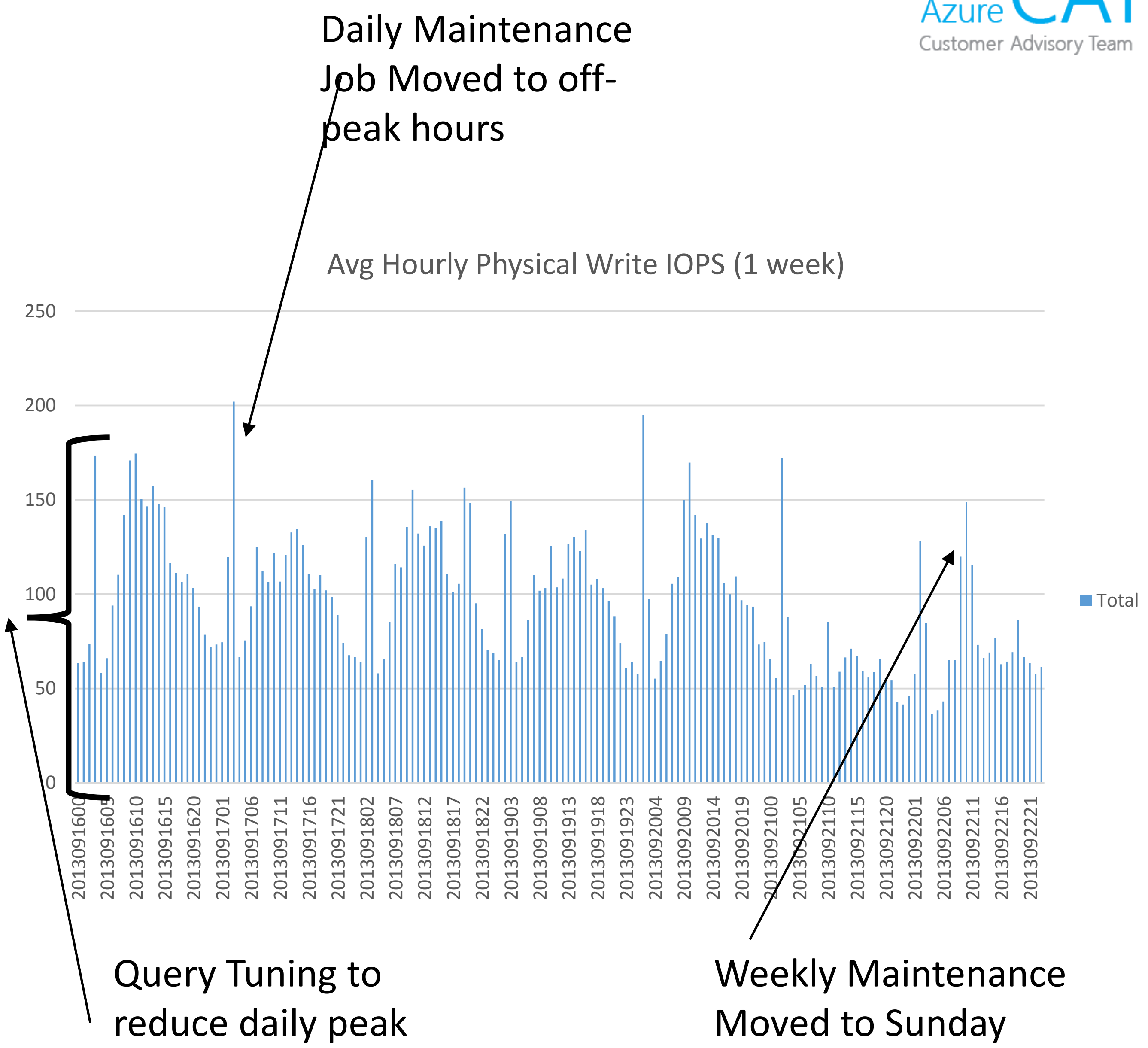
- Normal Box Capacity Planning happens rarely (usually once)
  - You buy a box bigger than you hope you will ever need
  - You tune as needed to make things work well in that box
  - If things are really bad, buy a bigger box
- Cloud planning is more-or-less continuous
- When you want to not spend more money, you tune
- Some tuning is just like normal SQL Server (tune queries, etc.)
- Other tuning is completely different…

Managing DB Resource Growth

- Database size/resource growth need careful management in SQL DB
  - There are size limits on things and you can be broken if you hit them
- Assuming your application resources grow over time, you need a plan to deal with that growth
- Two architectural approaches to manage
  - "Scale-up" (limited): Web/Business -> P1 -> P2
  - "Scale-out": use more databases
- Partitioning data by function or by tenant allows you to adjust as needed to growth in resource usage
- Plan on actively monitoring/alerting telemetry about the resource use so you can adjust to growth before something breaks…

Peak Load Example

- Weekly IO chart of a large customer on WA SQL DB
- We **actively** work on the load each week
  - Query tuning
  - Moving maintenance jobs to off-peak hours
- We also do aggressive things
  - Split different functions out into different databases
  - Rate-meter background jobs to not impact core workloads



Scale-Up vs. Scale-Out

- P1-P2 supported during public preview period
- Additional sizes can be introduced by GA
- With a scale up approach you may lose some flexibility
  - E.g. require planning for worst case / peaks
  - Premium let you scale up/down between P1 and P2 max one time a day
- Scale up may not fit all costs/business models
  - Unpredictable workloads
  - Multiple database deployments

Windows Azure CAT
Customer Advisory Team

Easyjet Seat Selection System

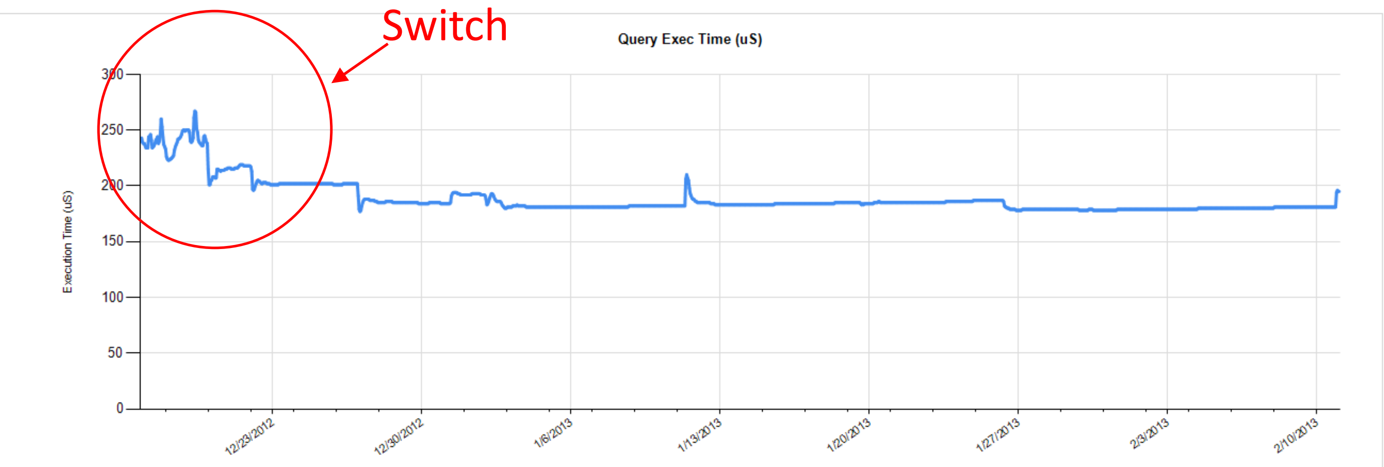
- 70/30 R/W workload, very efficient workload (<200mS max exec time)
- Majority of queries benefitted from switching
- Reduced and more stable response times for both reads and writes

SQL Database: Hourly Execution Time Stats

Return To Main Dashboard

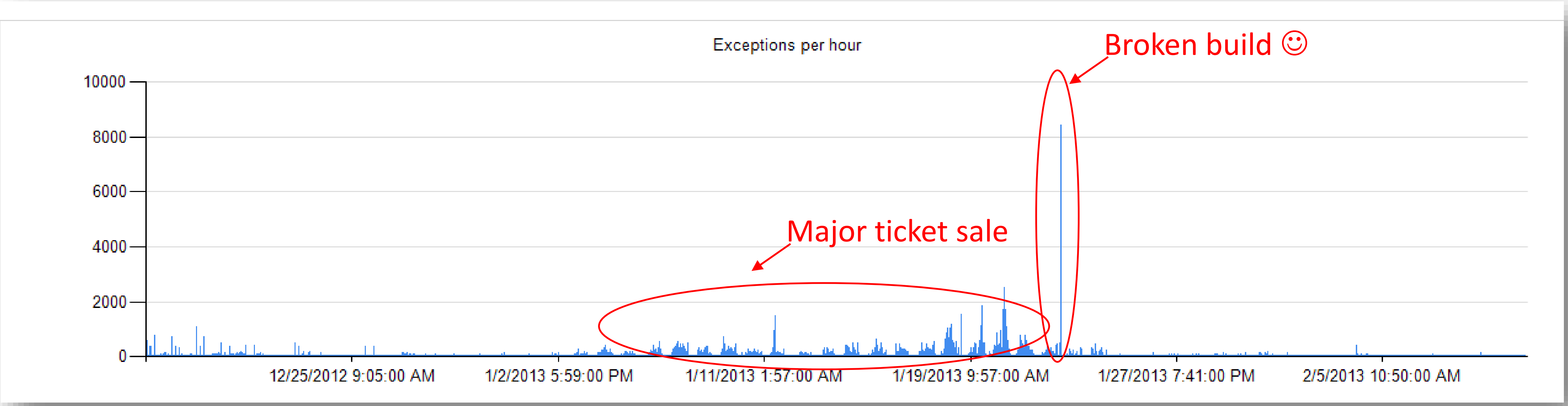
From: 12/1/2012 4:58:30 AM To: 2/11/2013 4:58:30 AM

Query text:	Avg Exec Time (uS):	Avg Elapsed (uS):	Avg Reads:	Avg Writes:	Count:	Last Execution Time:
SELECT s.FlightId, s.RowNumber, s.Letter, s.BlockNumber, s.SeatStatusId as Status, s.WithInfant, s.ReservedUntil, s.ReservationToken, s.ItineraryId, s.SeatAtGate, s.SsrNonPreferredSeat, s.[Timestamp] FROM dbo.Seat s WHERE s.FlightId = @FlightId ORDER BY s.rowNumber, s.Letter	181	227	5	0	24556022	2/11/2013 2:01 AM



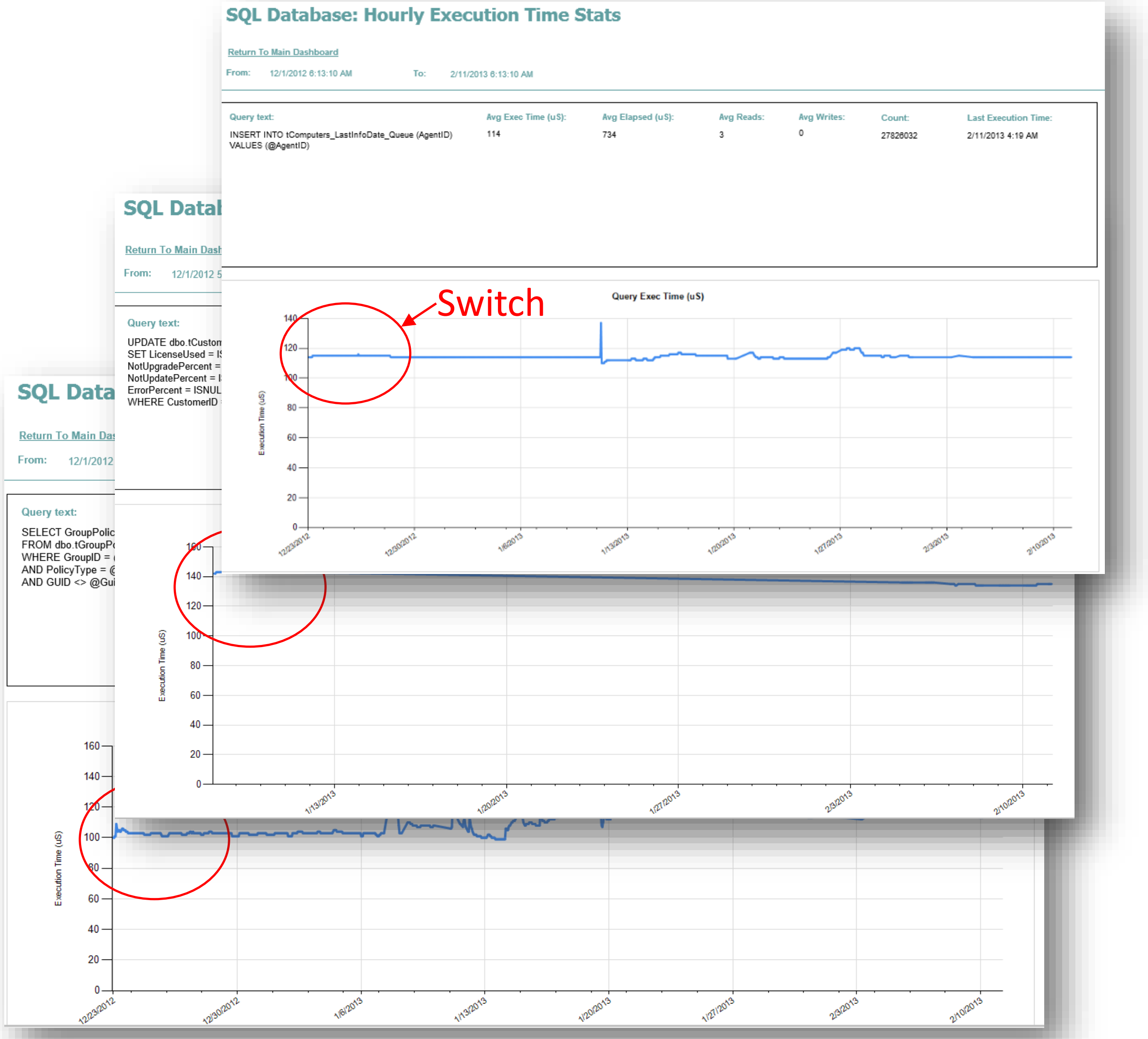
Customer experience: Easyjet

- Reduced impact of 40501, 10928 and 10929 errors
- Remaining exceptions have been mostly due to application issues



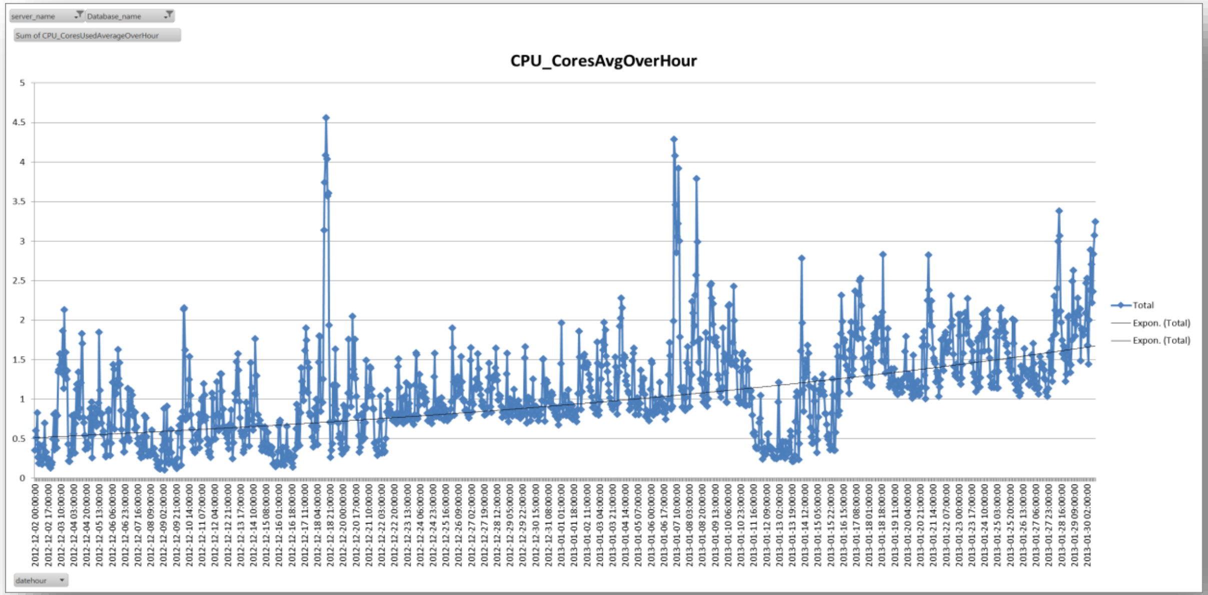
Customer experience: Panda Security

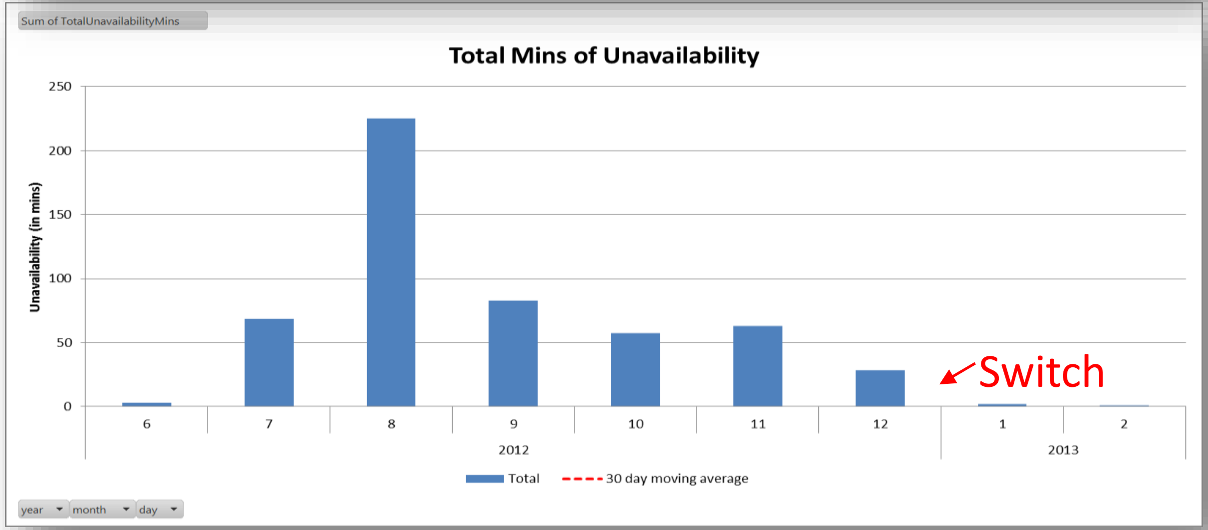
- 60/40 W/R workload
- Queue-like usage
- Have peaks of 50 active requests
- Many sub-optimized queries (indexing, blocking, etc)
- Most of the queries are stable, no big benefits after switch



Customer experience: Panda Security

- Availability has greatly improved after the switch (less than 2min x month)
- Growing trend in CPU usage
  - Around 2 on average, with spikes up to 5
- No major errors related to resource issues
- Sporadic throttling for High Log IO waits





Windows Azure CAT
Customer Advisory Team

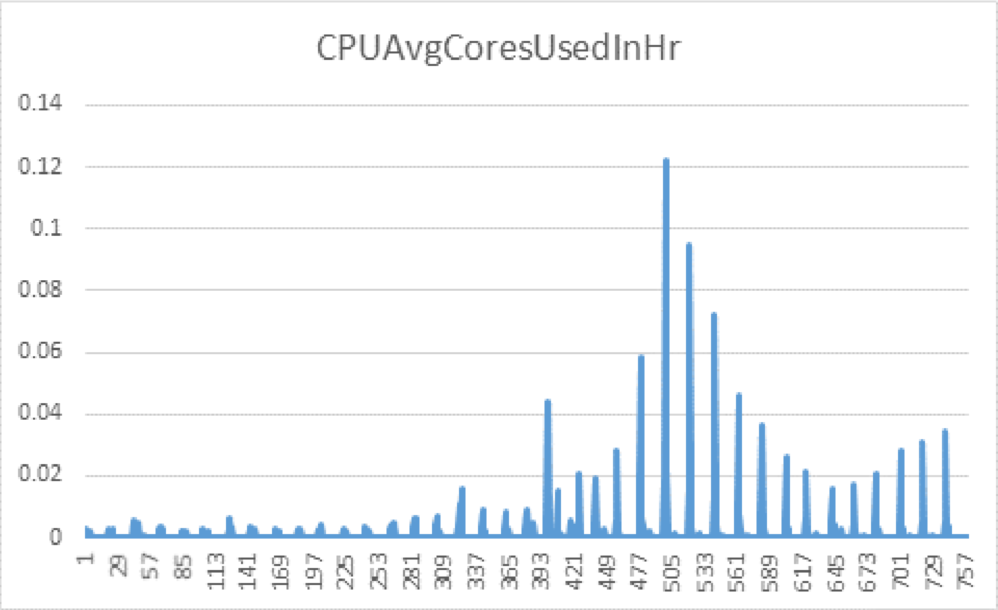
Application-Tier Caching

- App-tier caching is a very effective way to reduce data-tier load
- Azure has a several caching solutions available to you
- For load spikes, this can often significantly reduce peak load
- Example: Azure SQL DB was used in the last US Presidential Election
  - Few writes, massive reads all at once
  - Exponential decay distribution
  - App tier caching used to remove reads from the database

CPU graph for the core reporting DB

- 1st 10 seconds – 44K page views/second (est. ~450K DB calls/sec)
- Next 20 seconds – 10K page views/sec (est. ~100K DB calls/sec)

(DB calls mostly removed due to caching)



	Azure SQL DB	Azure SQL VM	Azure Storage Tables
Target Organizations	• Startup-mode dev organizations • Elasticity and flexibility, low friction	• Application compatibility • Existing enterprise customers • Traditional workloads	• Dev-oriented organizations • “Store and forget” • Limited query needs
Operations + Management	• Mostly auto-administered • Reduced surface for Insights	• Great single database • Requires effort for multi-db	• Storage analytics • 3rd party tooling
Availability	• 99.9% availability SLA • Requires development efforts if you need more	• (Almost) all the option of the box • Requires infrastructural effort for full HA	• Multiple copies of data • No “oops” recovery
Performance + Scalability	• Shared Resources • Requires scaling out	• Predictable performance • May be limited by hw resources	• Provided by partitioning • Throttling
Migration effort from on premises	• Limited application compatibility issues • May require effort in case of scaling out	• Mostly transparent • Can be more challenging to scale out if a single node is not enough	• May require rewrite of both data model and processing capabilities if coming from a relational app
Costs	• Optimized for low costs to buy and manage	• Licensing and VM costs could be not trivial	• Relatively cheap • May require additional computation resources

Resources

- Premium Preview for SQL Database Guidance (http://msdn.microsoft.com/en-us/library/jj853352.aspx)
- Azure SQL Database and SQL Server -- Performance and Scalability Compared and Contrasted (http://msdn.microsoft.com/en-us/library/windowsazure/jj879332.aspx)

Questions?

Windows Azure CAT
Customer Advisory Team

Microsoft