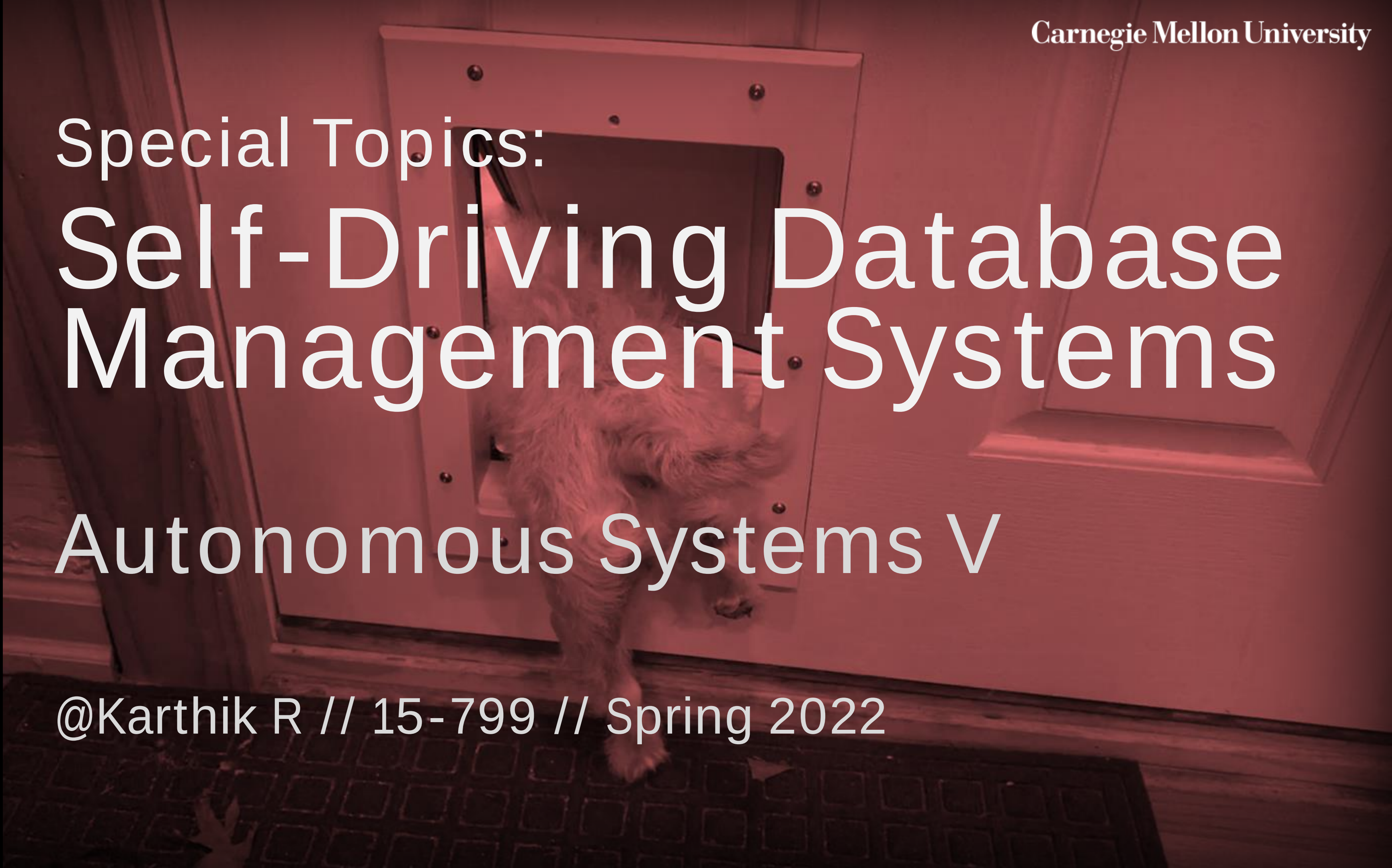
Lecture #22

Carnegie Mellon University



Special Topics:

Self-Driving Database Management Systems

Autonomous Systems V

@Karthik R // 15-799 // Spring 2022

LAST CLASS

Series on End-to-End Autonomous Systems

- UDO
- openGauss
- Database Learning
- Transferable Model for Learning

TODAY'S AGENDA

- Database Offerings at Microsoft
- Auto-Indexing Architecture
- Index Recommendations
- Experiments
- Parting Thoughts

AZURE SQL PRODUCT-LINE

SQL Server on Azure Virtual Machine

- Vanilla DB-on-VM offering
- Pay for virtual machine + software license
- Access to underlying OS

Azure SQL Managed Instance

- Fully "managed" instance
- Pay for (compute, storage)
- Auto-scales
- No access to underlying box
- Auto-upgrades

Source: Azure Documentation

AZURE SQL PRODUCT-LINE

Azure SQL Database

- Flagship offering
- All the features of SQL Managed Instances + “Intelligence”
- Currently available for storage < 100 TB
- Server-ful and serverless variants

Azure SQL Edge

- ~~Streaming, Time-series and in-database Machine Learning capabilities~~

Source: Azure Documentation

CHALLENGES AT SCALE

Even if you could recommend the "best" indexes, how do you:

- Identify a representative workload?
- Analyze prod instances ensuring QoS compliance?
- Implement the indexes (the WHEN)?
- Ensure no regressions?
- Detect drifts in workload and data distributions?

. . . . while still being profitable?

AUTO-INDEXING ARCHITECTURE

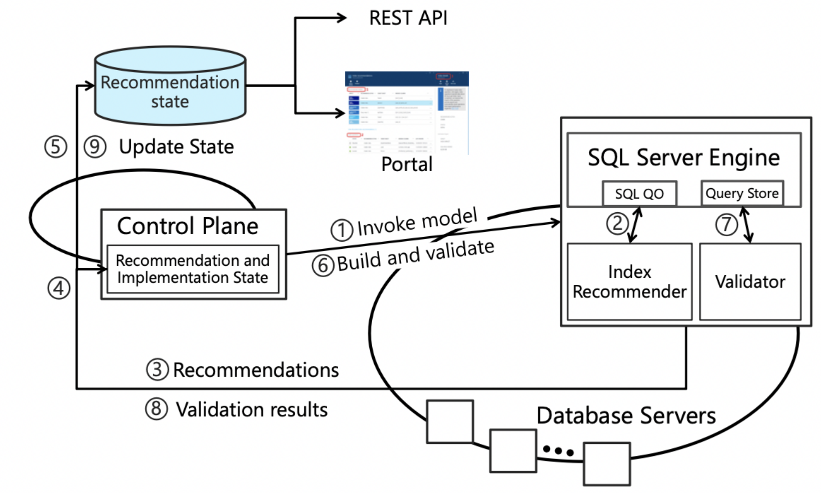


Figure 4: Conceptual architecture of auto-indexing service within a single Azure region.

Moving Parts:

Central:

- Control Plane aka "The Brain"

Per Database Instance:

- Index Recommender
- Validator

Within SQL Server:

- Query Optimizer
- Query Store

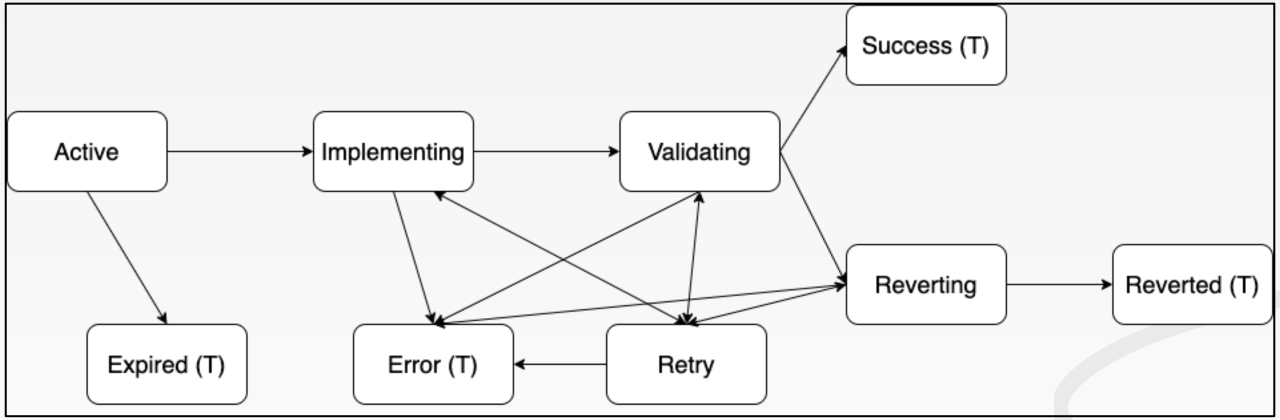
Everywhere:

- Telemetry

CONTROL PLANE

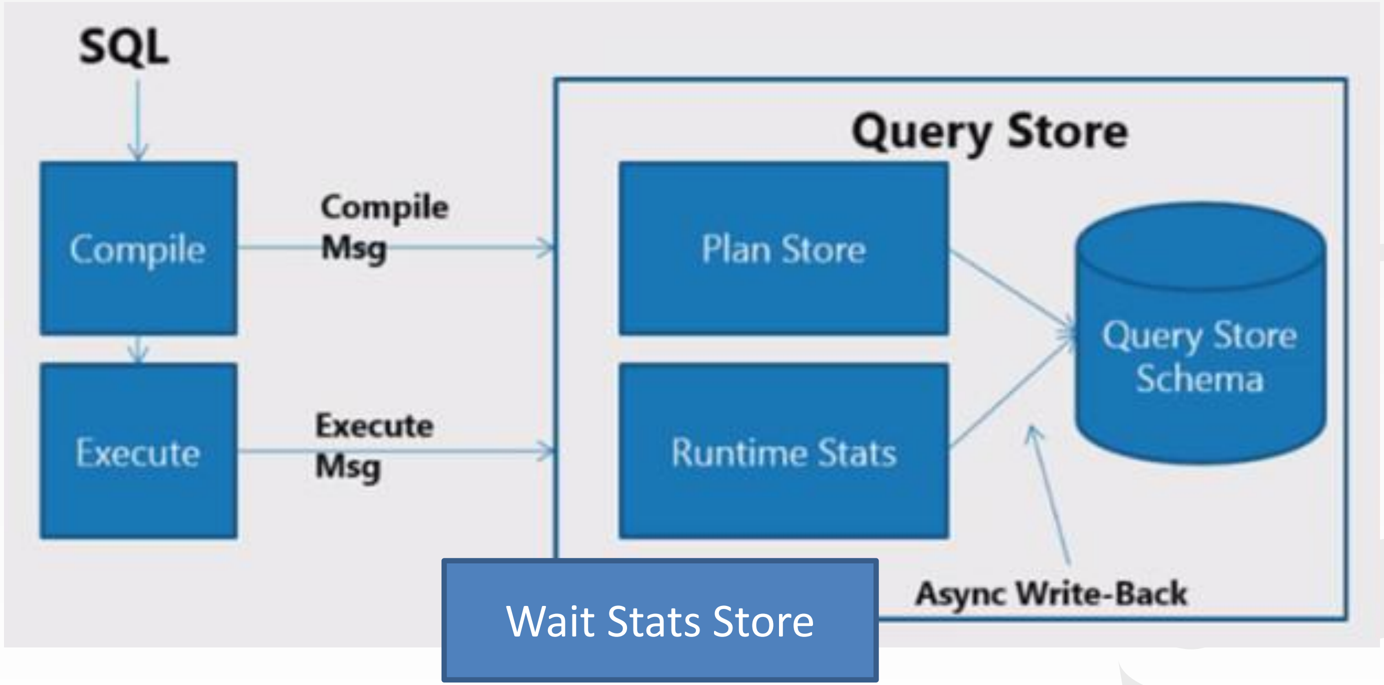
- Highly Available and Fault Tolerant service
- One auto-indexing service per Azure region
- Manages a persistent, highly available data store
- Collection of micro-services
  - Invoke Index Recommendation service
  - Implement recommendations
  - Validate recommendations
  - Detect anomalies in the system

CONTROL PLANE



Every index recommendation is persisted as a state machine.

QUERY STORE



Source: RED-GATE

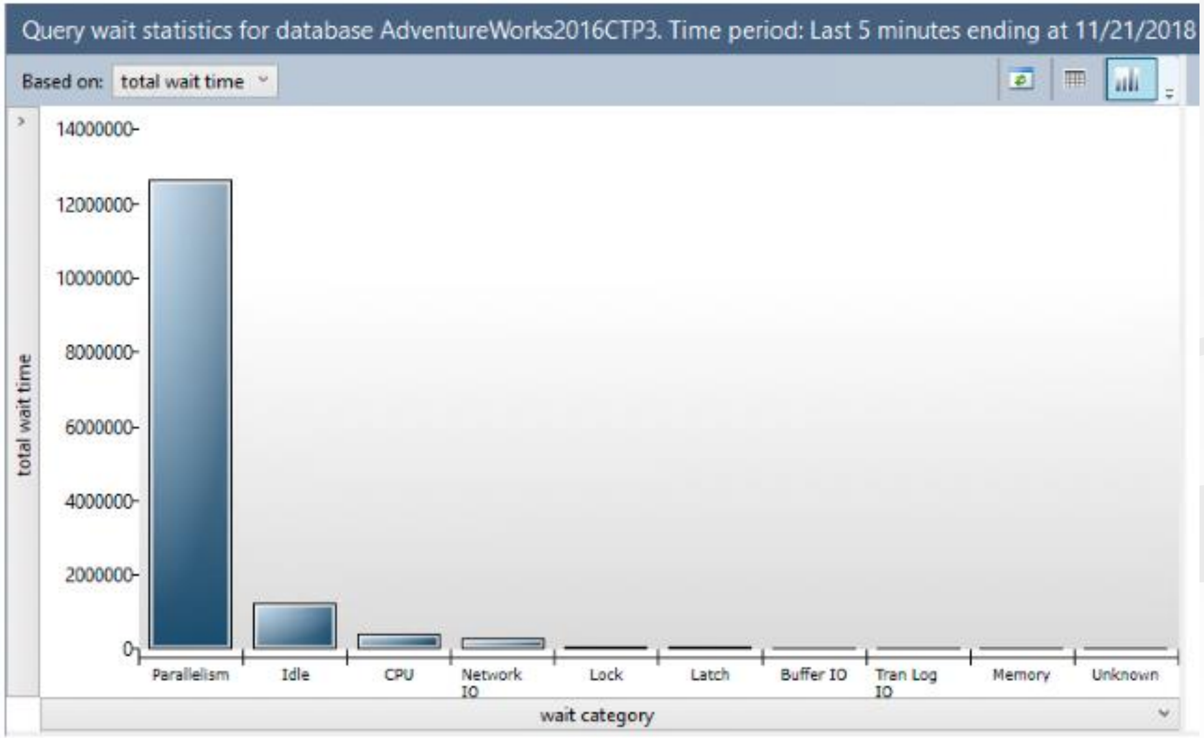
QUERY STORE

Stores the following:

- Plans: Execution plans for a given query
  - Useful for forcing query plans and detecting impact of indexes
- Runtime stats: Resource consumption per query
  - Can answer questions like "Top N queries for resource A in the last X hrs"
- General stats: audit information
- Wait stats: Locking, resource limits, system degradation

Source: Microsoft documentation

QUERY STORE


Query wait statistics for database AdventureWorks2016CTP3. Time period: Last 5 minutes ending at 11/21/2018

Based on: total wait time

Fig: Query wait categories

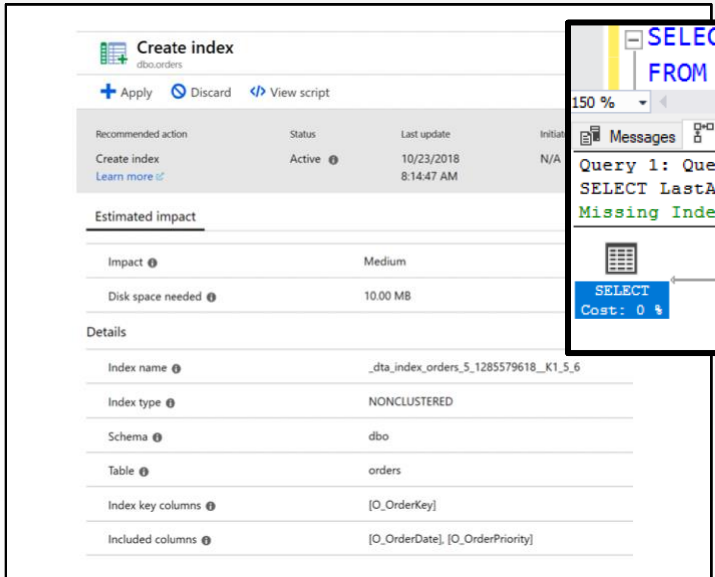
Source: Microsoft documentation

HISTORY OF INDEX RECOMMENDATIONS

97-99	• Index Recommendations (Selection, What-If Analzser)
00-04	• Automated Physical Design Recommendation (MV, Partitioning)
04-11	• Database Tuning Advisor, Auto-Admin
13-16	• Databases on Cloud (SQLVM, Resource Sharing)
18-19	• DTA for Cloud (Hybrid physical designs, Auto-indexing)

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

INDEX RECOMMENDER



Create index
dbo.orders

Apply · Discard · View script

Recommended action	Status	Last update	Initiat
Create index Learn more	Active	10/23/2018 8:14:47 AM	N/A

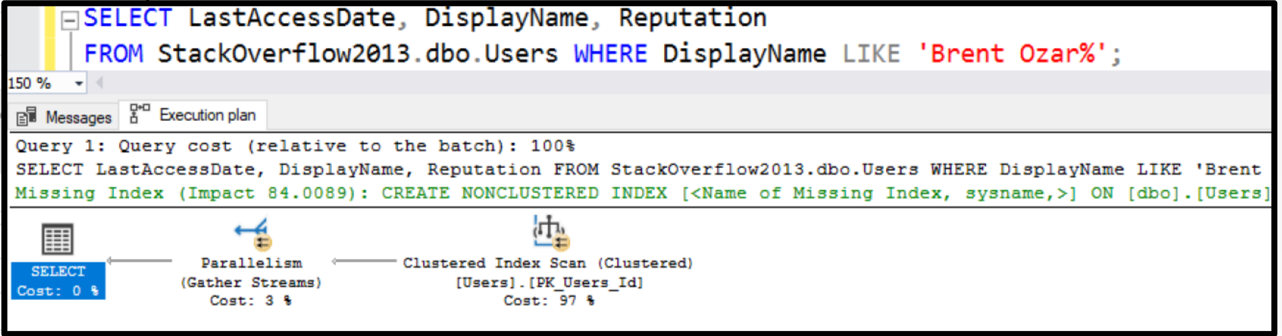
Estimated impact

Impact	Medium
Disk space needed	10.00 MB

Details

Index name	_dta_index_orders_5_1285579618__K1_5_6
Index type	NONCLUSTERED
Schema	dbo
Table	orders
Index key columns	[O_OrderKey]
Included columns	[O_OrderDate], [O_OrderPriority]

Figure 3: A snapshot of the UI where customers see the recommendation details.



```
SELECT LastAccessDate, DisplayName, Reputation
FROM StackOverflow2013.dbo.Users WHERE DisplayName LIKE 'Brent Ozar%';
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT LastAccessDate, DisplayName, Reputation FROM StackOverflow2013.dbo.Users WHERE DisplayName LIKE 'Brent
Missing Index (Impact 84.0089): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users]
```

Two components:

- Missing Index Recommender
- Database Tuning Advisor

AUTOMATICALLY INDEXING MILLIONS OF DATABASES IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

INDEX RECOMMENDER

Missing Indexes (MI) Recommendation

- Lightweight, always ON tool
- Analyzes the leaf nodes of plans (query-level)
- Suggests "good-to-have" indexes

```
SELECT column_a, column_b
FROM table_1
WHERE column_a < 10000
AND column_b < 10000
ORDER BY column_b, column_a
```

- Doesn't benefit JOINs, ORDER/GROUP BYs
- Does not account for index maintenance overheads
- Primarily memory resident

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

MISSING INDEXES RECOMMENDATION

For each query:

- Step 1: Identify candidate columns
- Step 2: Leverage statistics to determine optimizer-estimated benefit on a per query/statement basis
- Step 3: Filter out candidates with few query executions
- Step 4: Assess the impact of candidates over a period of time
- Step 5: Merge of index candidates

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

INDEX RECOMMENDER

Database Tuning Advisor (DTA)

"DTA is a comprehensive physical design tool that given a workload *W*, finds the physical design that minimizes the optimizer estimated cost of *W*."

Problem : How to identify a representative workload *W* automatically?

Leaving workload-capture ON 24*7 is resource intensive!

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

IDENTIFYING WORKLOADS

How about leveraging the Query Store?

- Query are sampled before logging in Query Store (QS)
- Conditional, Loops, Variables etc. are not logged
- A single query isn't always the right granularity

Solution:

- Query Text in QS is augmented with system metadata
- Batch definitions are pulled from plan cache of QS
- Queries incompatible with "What-If" API are rewritten

DATABASE TUNING ADVISOR (DTA)

Problem: How to ensure that database is always online?

Solution:

- Resource governance for "system tasks" in a multi-tenant setup
- Low-priority locks for modifying hypothetical indexes
- Change to DTA to reduce number of invocations

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

RESOURCE SHARING

In a multi-tenant environment:

- Resource guarantees must be absolute
- Services must be cost-effective for provider
- Independent accountability from provider
- No assumptions about workloads, usage timelines
- Support for policies such as elasticity

CPU SHARING TECHNIQUES FOR PERFORMANCE ISOLATION
IN A MULTI-TENANT RELATION DATABASE-AS-A-SERVICE
VLDB 2014

RESOURCE SHARING

Solution

- Concept of an SQLVM
  - A logical construct with a promise of a set of resources for a tenant inside a database server (ie. VM)
  - Resources: CPU, I/O, Memory, (potentially network bandwidth, etc)
  - Isolates system/mgmt tasks from the tenant's resource reservation
- Independent metering
- Scheduling based on Largest Deficit First (LDF)
  - Can support elasticity, and heuristics such as max-ing provider's revenue

CPU SHARING TECHNIQUES FOR PERFORMANCE ISOLATION
IN A MULTI-TENANT RELATION DATABASE-AS-A-SERVICE
VLDB 2014

DATABASE TUNING ADVISOR (DTA)

Putting it all together,

- Service-ification – invoked by the control plane
- Analyses the execution stats of top k queries over last N hours by resource X
- Recommends ONLY non-clustered B+ tree indexes

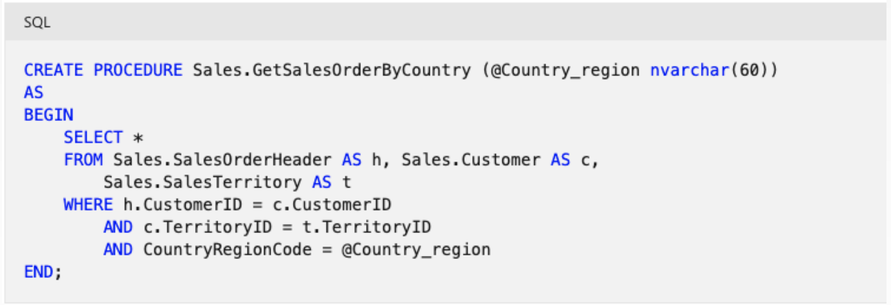
(Remember that DTA also supports clustered indexes, MV, partitions)

- Considers JOINs, ORDER BY, GROUP BY columns
- Works under imposed constraints (such as storage budget)

DATABASE TUNING ADVISOR (DTA)

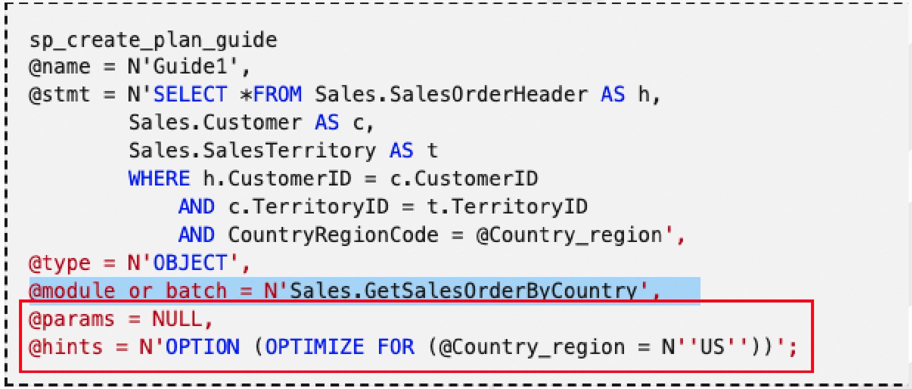
Dropping Indexes:

- Manually created indexes for infrequent but imp. queries
- Query hints and forcing query plans
- Duplicate indexes; which one to drop?



```
SQL

CREATE PROCEDURE Sales.GetSalesOrderByCountry (@Country_region nvarchar(60))
AS
BEGIN
    SELECT *
    FROM Sales.SalesOrderHeader AS h, Sales.Customer AS c,
        Sales.SalesTerritory AS t
    WHERE h.CustomerID = c.CustomerID
        AND c.TerritoryID = t.TerritoryID
        AND CountryRegionCode = @Country_region
END;
```



```
sp_create_plan_guide
@name = N'Guide1',
@stmt = N'SELECT *FROM Sales.SalesOrderHeader AS h,
        Sales.Customer AS c,
        Sales.SalesTerritory AS t
        WHERE h.CustomerID = c.CustomerID
            AND c.TerritoryID = t.TerritoryID
            AND CountryRegionCode = @Country_region',
@type = N'OBJECT',
@module_or_batch = N'Sales.GetSalesOrderByCountry',
@params = NULL,
@hints = N'OPTION (OPTIMIZE FOR (@Country_region = N''US''))';
```

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

INDEX IMPLEMENTATION

Problem: How to schedule indexing operations?

- Index creation is resource hungry
- Impacts concurrently running workloads

Solution:

- Resource governance for “system tasks” in a multi-tenant setup
- Forecasting periods of low activity

INDEX VALIDATION

Problem: How to detect implemented indexes are *actually* good?

- Optimizer costs can be inaccurate
- Collected data can be noisy due to concurrency
- If the index is bad, what do you do?

Solution:

- Leverage ONLY logical metrics for goodness of query execution
- Consider only executions with a plan change
- Use of statistical measures to determine regression

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

EXPERIMENTATION AT SCALE

Problem: How are changes to the system tested?

- Changes cannot be run in production
- What constitutes a representative set of instances?

Solution: "B-instances" to the rescue!

- Not a secondary replica
- An entirely different instance with a snapshot of the database
- Asynchronous to the primary, a best-effort clone
- Allows running different binaries, having different resources

AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

EXPERIMENTATION AT SCALE

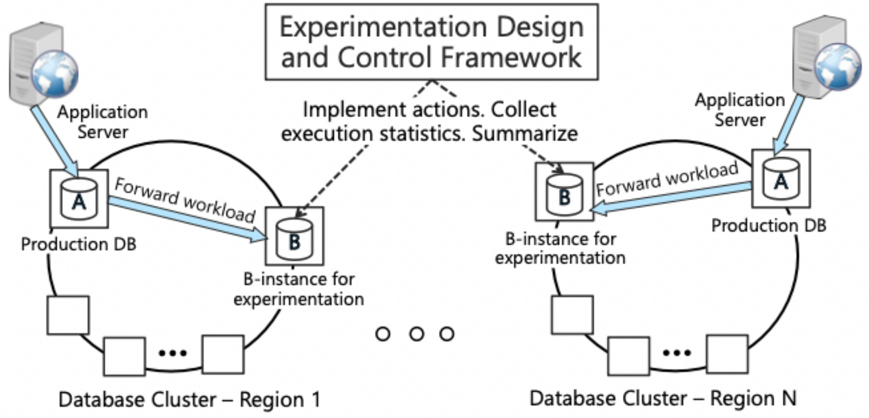

Experimentation Design and Control Framework
Implement actions. Collect execution statistics. Summarize
Application Server
Forward workload
A
Production DB
B
B-instance for experimentation
Database Cluster – Region 1
Application Server
Forward workload
A
Production DB
B
B-instance for experimentation
Database Cluster – Region N

Figure 5: A conceptual architecture for experimentation using B-instances in Azure SQL Database.

EXPERIMENTATION AT SCALE

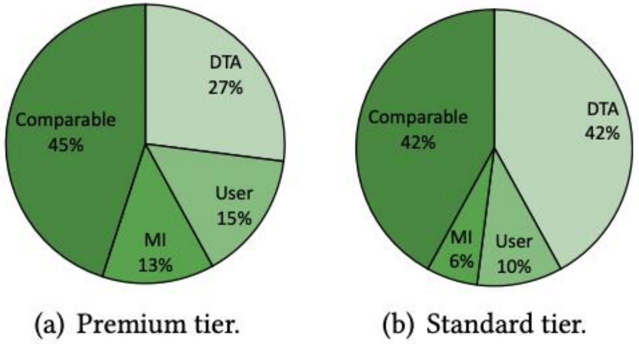
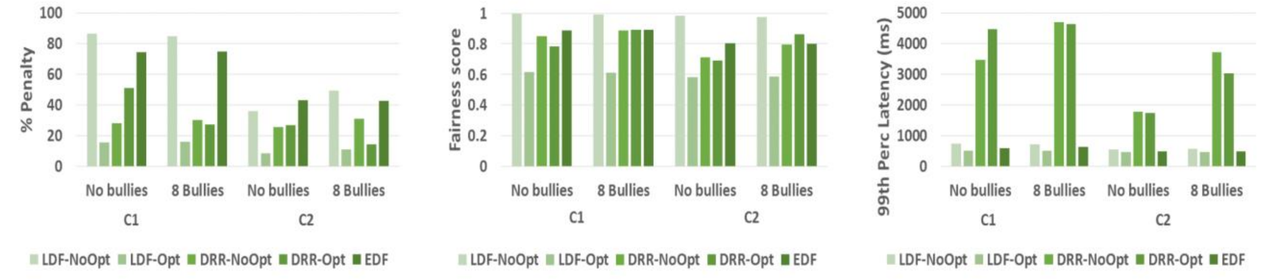


Figure 6: Experimentation at scale with production databases in the premium and standard tier.

WHAT ABOUT PERF ISOLATION?



(a) Penalty as % of subscription. (b) Fairness (Jain's index). (c) 99^{th} percentile latency of T_8.

Figure 5: Differentiated service and optimizing for provider revenue.

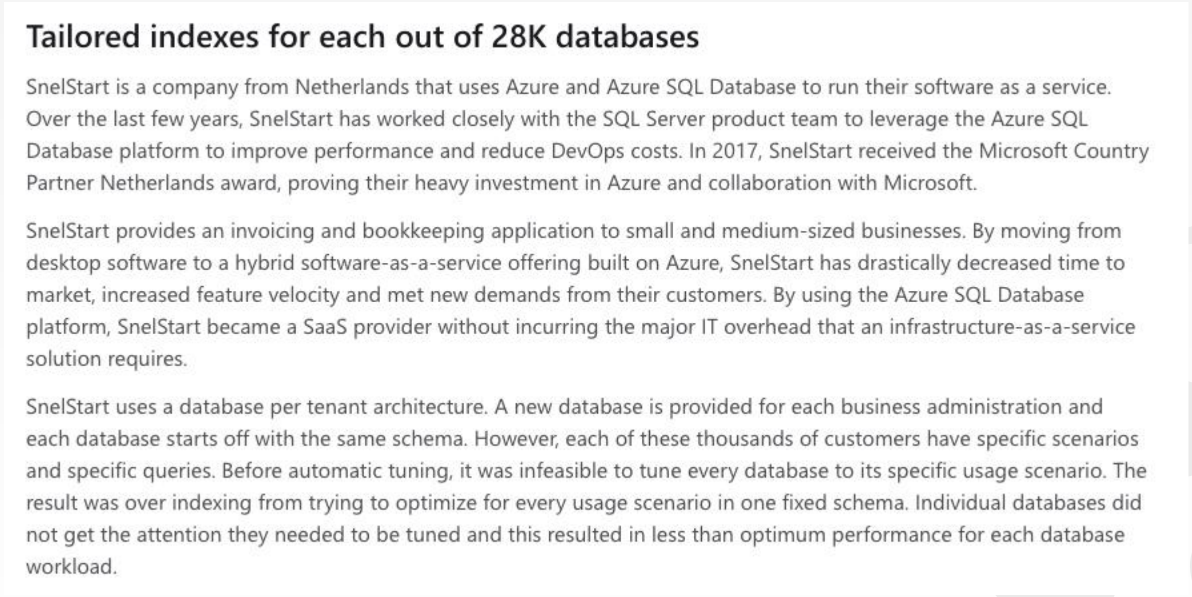
AUTOMATICALLY INDEXING MILLIONS OF DATABASES
IN MICROSOFT AZURE SQL DATABASE
SIGMOD 2019

SOME NUMBERS…

As of October 2018,

- 250K recommendations for CREATE INDEX
- 3.4M recommendations to DROP INDEX
- Most databases reach a stead state
- DTA results in ~82% CPU time improvement, MI ~72%
- 11% regression rate

USE CASE



Tailored indexes for each out of 28K databases

SnelStart is a company from Netherlands that uses Azure and Azure SQL Database to run their software as a service. Over the last few years, SnelStart has worked closely with the SQL Server product team to leverage the Azure SQL Database platform to improve performance and reduce DevOps costs. In 2017, SnelStart received the Microsoft Country Partner Netherlands award, proving their heavy investment in Azure and collaboration with Microsoft.

SnelStart provides an invoicing and bookkeeping application to small and medium-sized businesses. By moving from desktop software to a hybrid software-as-a-service offering built on Azure, SnelStart has drastically decreased time to market, increased feature velocity and met new demands from their customers. By using the Azure SQL Database platform, SnelStart became a SaaS provider without incurring the major IT overhead that an infrastructure-as-a-service solution requires.

SnelStart uses a database per tenant architecture. A new database is provided for each business administration and each database starts off with the same schema. However, each of these thousands of customers have specific scenarios and specific queries. Before automatic tuning, it was infeasible to tune every database to its specific usage scenario. The result was over indexing from trying to optimize for every usage scenario in one fixed schema. Individual databases did not get the attention they needed to be tuned and this resulted in less than optimum performance for each database workload.

PARTING THOUGHTS

Contributions of the paper:

- Identification of a representative workload
- Guaranteeing resource isolation for tenant

What's missing?

- Decision-making framework

What's next for Microsoft?

- Physical design automation (knobs maybe?)

NEXT CLASS

- Project Presentations
- Guest speakers next week!

"That's all Folks!"