Implement Non Relational Data Stores

Saturday, 25 July 2020 5:52 PM

Introduction to NoSQL

- Horizontal scaling
- Flexibility

Usage

- large amounts of data
- relationships isn't important
- data changes over time

SQL vs NoSQL

SQL	NoSQL
Relational database	Non-relational or distributed
Fixed schema	Dynamic
Designed for complex queries	Not for complex queries
SQL, MySQL, Oracle, Postgres	Mongo DB, Redis, HBase
Vertical scaling	Horizontal scaling

Use Cases

Product Orders	Internet of things
Student Management	Content Management

Types of NoSQL databases

Document databases

- Typically JSON
- Family of information with varying information.
- Cosmos DB

Key Value Stores

- highly optimized for simple lookup
- scalable across multiple nodes
- not ideal for non-key columns and values
- Cosmos DB tables API, Radis, Table storage

Graph databases

- comprised of nodes and edges
  - Nodes = entity
  - Edges = relationship between entities
  - Cosmos DB Graph API

Wide column stores

- Data organized into columns and rows
- Families of data separate from traditional RDBMS
- HBase, Cassandra

Azure Data Lake

Repository for storing massive amounts of structured and unstructured data.
Built on top of Blob storage

Does not support USQL

Cosmos DB Essentials

Globally distributed
Multi model
High availability

Usage
- IoT
- Marketing
- Web
- Mobile

Features
- Turnkey global distribution
  - transparent replication with touch of button
  - 5 consistency levels
    - strong
    - Bounded staleness
    - session
    - Consistent prefix
    - eventual
- Reliability
  - always on
  - 99.999 % SLA
  - < 10 ms latency
  - global
- Elastic
  - transparent multi-master replication
  - horizontal partitioning
- Multi-modal
  - SQL API
  - Cassandra API
  - Cosmos DB API for Mongo DB
  - Gremlin API
  - Table API

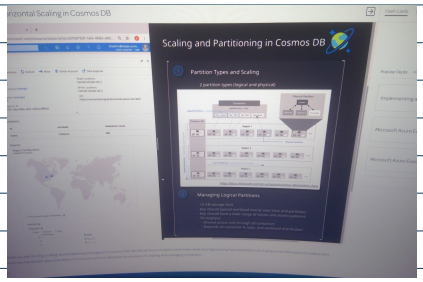
Automatic Indexing
- Modes (consistent, none)
- Includes and excludes may be set per container
- Composite Indexes (must be manually set)

Cosmos DB functions best with global data, low latency and big data

SQL API provides SQL query API over No-SQL store
Gremlin API provides graph based API.

Partitioning and Horizontal Scaling in Cosmos DB

Logical and Physical partitions



Logical partitions are converted into physical partitions using hashing algorithm.

Request Units (RU)

Throughput

Rate Limitations would kick in if the provisioned throughput is exceeded.

Managing Logical Partitions

- 10 GB storage limit
- Key should spread workload evenly over time and partitions
- Key should have a wide range of values and access patterns
- Throughput
  - shared across all containers
  - depends on container #, keys and workload distribution

Azure Table Storage

- NoSQL
- Key-value store
- used in Cosmos DB using Table API.

- High capacity, single region.
  Cosmos DB - Global

Common use cases

- storing large amounts of structured data
- storing joins that don't require complex joins or foreign keys
- less expensive option when global replication isn't needed

Entity Limits

1 MB for table storage per row
2 MB for Cosmos DB, max 252 properties per table

Implement Relational Data Stores

Sunday, 26 July 2020 11:46 AM

Data Warehouse

Cloud data Warehouse

- scalability
- Cost
- Time to market
- Performance

Data models and Fact tables
- Dimensions : Attributes
- Fact Table : Centre of Schema

SQL Data Warehouse

Massively Parallel Processing (MPP) Engine
- Data Warehouse units
- Separation of storage and compute

Distributions

- law of 60, compute, and control

- sharding patterns
  - Hash : Highest query performance for large tables
  - Round Robin : Easiest to create for staging tables
  - Replicate : Fastest query for small tables.

Azure SQL Database

SQL Database functionality

- Relational database managed service (RDBMS)
  * Built on SQL database engine
  * Vertically scaling

- In memory OLTP

- In memory OLTP

- Extensive monitoring and alerting
  * Azure storage
  * Azure Event Hub
  * Azure Monitor Logs

- Platform as a Service (PaaS)
  * Single database
  * Elastic pool
  * Managed Instance

- Scalability
  * Vertical scalability
    - CPU power, memory, IO throughput, and storage
    - DTU and vCore models to scale
    - Dynamic scalability

SQL Purchasing models

Virtual Core (vCore)

- Independently choose compute and storage
  * scale compute, storage and IO
  * Azure Hybrid benefits for SQL server
  * Cost for both compute and storage
- Provides flexibility, control and transparency

Database Transaction Unit (DTU)

- choose bundled compute, storage and IO options
- Provides simple, pre-configured options
  * if you use over 300 DTU, vCore is better choice
  * 3 tiers : Basic, Standard and Premium

Criteria for choosing

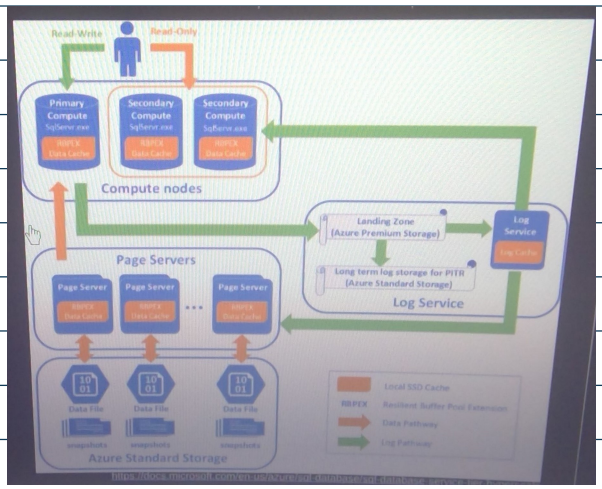
- vCore is better for high-cost databases or if you need to closely monitor and trim costs.
- DTU is best when running inexpensive databases or need simplicity

SQL Database Hyperscale

Only accessible via vCore option
Removes the max data size limitations

Ideal for
- modernizing large on-premise databases
- cloud customers hitting max database restrictions (1-4 TB)

100 TB max size
Near instantaneous backup with no impact on compute
Billed for use with scale up and "scale out" capabilities



Elastic Pools and Elastic Jobs

Elastic Pools

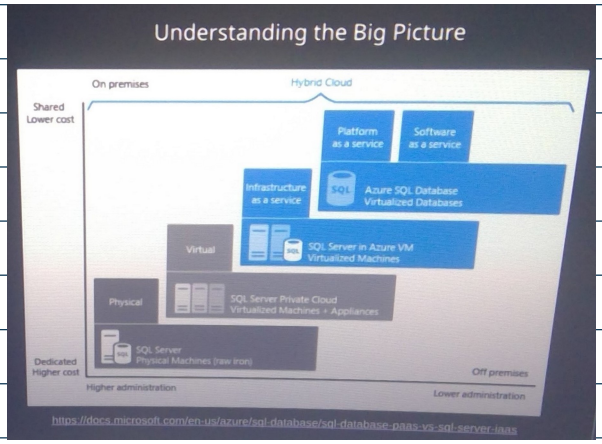
- Designed for unpredictable demands
- Shared pool resources
  * Manage and scale multiple databases
- 2 pricing options

* Manage and scale multiple databases
- 2 Pricing options
  * Database transaction units (DTU)
  * Virtual Core (vCore)

Elastic Job

- scheduled jobs running TSQL or queries
- 2 types
  * SQL Agent jobs (Managed)
  * Elastic jobs (Elastic)
- Retries using logged queries
- Can run against multiple servers, pools, or databases

SQL DB Managed Instance



Managed Instance is all about moving from Iaas or on-premise environments
Paas - no hardware management, auto patching, 99.999 % uptime
Native vNet integration
Automated backups

Could take upto 6 hours to provision a managed instance.

Automated Backup Strategies:

Automated Backups

- Read-access geo-redundant storage (RA-GRS)
- Full weekly / differential every 12 hours / transaction every 5-10 mins

Usage

- restore an existing database to point in time
- restore deleted database
- restore DB to another geographical region
- restore DB from a specific long-term backup.

defaults

DTU: Basic 1 week / standard and premium 5 weeks
vCore: 1 week by default

Long-Term Retention (LTR)

- Leverages PITR - blob for long term storage
- Geo-replication

Managing Data Security

Monday, 27 July 2020 11:00 PM

Implement Data Masking

Dynamic Data Masking

- Real-time data masking
- Limit sensitive data exposure by masking it to non-privileged users
- Masking logic
  * Default → complete mask
  * Credit card → partial data mask
  * Email → first letter visible, rest all letters masked
  * Random → Random masks
  * Custom → define custom mask.

Encrypt data at rest and in motion

Encrypting data at rest

Available for

- SaaS
- PaaS
- IaaS

Transparent data encryption

- Key services
  * SQL Server
  * Azure SQL Database
  * Azure SQL Data Warehouse
- Works
  * AES and Tripple Data Encryption Standard
  * Database encryption key

Cosmos DB database encryption

- Encryption by default
- Uses secure key storage system, encrypted networks

- Encryption by default
- Uses secure key storage system, encrypted networks and cryptographic keys

Datalake encryption
- on by default
- Customer managed keys
- 3 keys
  * Master Encryption Key (MEK)
  * Data Encryption Key (DEK)
  * Block Encryption key (BEK)

Encrypting Data in motion
- TLS / SSL
  * Transport Layer Security -> between cloud and customer
  * Strong authentication
  * Message privacy
  * Integrity
  * Perfect forward secrecy (PFS)
    => Protects data between customers client systems and cloud

- Shared Access Signature
  * Delegate access to Azure storage objects

- Data Lake
  * Data in transit is always encrypted in Data Lake store
  * HTTPS is the only protocol available for REST interface

Developing Batch Processing Solutions

Monday, 3 August 2020 9:46 PM

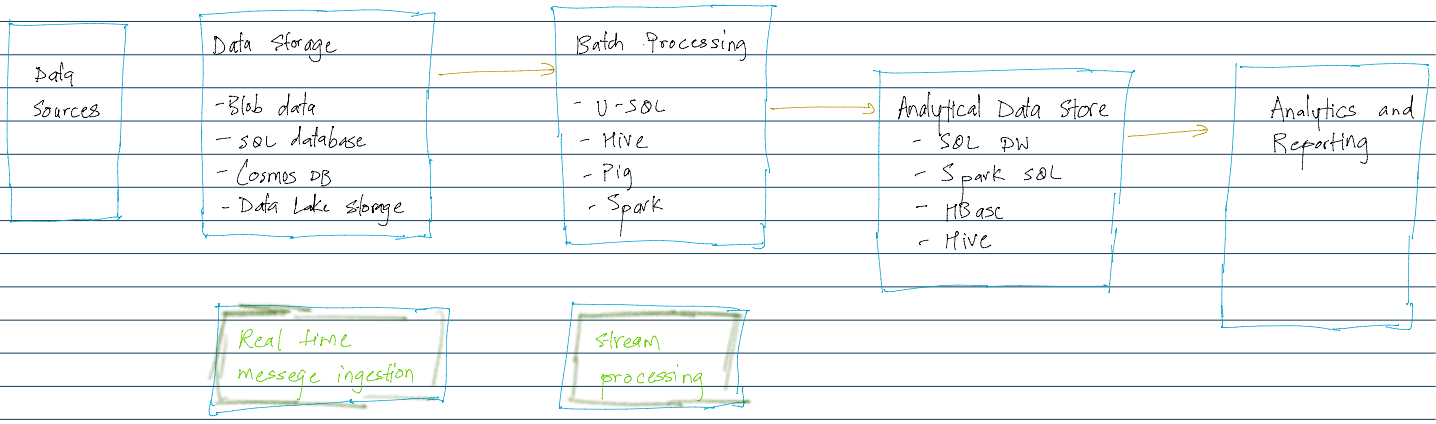
Batch Processing Overview

Batch processing

- Blocks of data over time
- Challenges
  - * Data format and encoding
  - * Orchestration
- Stream processing
  - Real time processing

Batch Processing Architecture

- Orchestration
  - * Data factory as a driver
- Batch Processing
  - * Read source file
  - * Process
  - * Write output
- Analytical Data Stores
  - * Optional
  - * Necessary for structured data needs
- Analysis
  - * Driving critical business insights



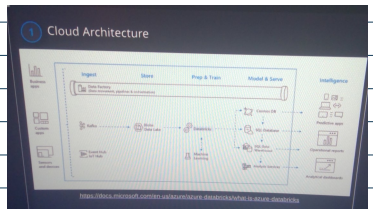
Apache Spark Overview :-

Introduction to Apache Spark

- Performs batch processing and stream processing
- Spark Applications
  - * Driver process
    - Executes program commands
  - * Executor process
  - * Cluster manager
- Resilient Distributed Datasets (RDD)
  - * fundamental data structure of spark
  - * Resilient - fault tolerant
  - * Distributed - Data resides on multiple nodes
  - * Datasets - data that you work with
- Transformations and Actions
  - * Lazy evaluations
    - Data is not loaded until it is necessary
    - Transformations get executed only when driver requests information
  - * Actions
    - returns values

Databricks Introduction

Cloud Architecture



Terminology

Cluster: Group of computer resources
Workspace: The filing cabinet for databricks work
Notebooks: Folders in the cabinet that contain cells
Cells: Individual pieces of code used to execute commands
Libraries: Packages or modules that provide additional functionality
Tables: stores structured data

Key features

- Interactive workspace for exploration and visualization
- Core API: R, SQL, Python, Scala, and Java
- Streaming (HDFS, Flume, Kafka) and batch processing
- Dynamic scaling and secure integrations

PolyBase Introduction

Import / Export data from Data Warehouse and ADLS

Allows to bypass traditional ETL in fav of ELT.

- performs transformations in the data warehouse
- Better distribution of transformations across multiple databases

Basic steps

- Extract source data into text files
- Land data into Azure Blob storage or Azure Data Lake Store
- Prepare data for loading
- Load data into SQL Data Warehouse staging tables using PolyBase
- Transform the data
- Insert data into production tables.

Develop Streaming Solutions

Monday, 3 August 2020 11:00 PM

Introduction to Azure Stream Analytics

Batch Processing vs. Streaming

- Batch processing : Collect a batch of information, then send it in chunks for processing
- Streaming : Information is constantly fed into the system in realtime.

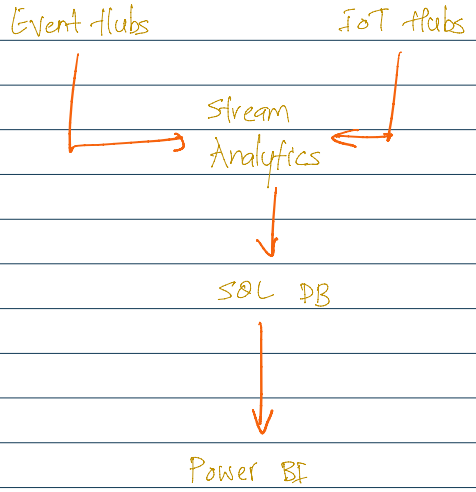
Streaming overview:

- Key for analytics results in real time
  - What is the end result of data,
  - What will data be used for?
  - How to handle complexity?
  - Reliability and fault tolerance?
  - cost factor?
- Data refresh should be driven from actionable business decisions

Use Cases :

- Security or fraud detection
- Traffic sensors
- Health care

Introduction to Azure Stream Analytics



Azure Stream Analytics

Recommended scenarios
- Dashboards for data visualization
- Real-time alerts
  * Running
  * Stopped
  * Degraded
  * Failed
  * Percentage
- Extract, transform, load
- Event sourcing pattern
- IoT Edge

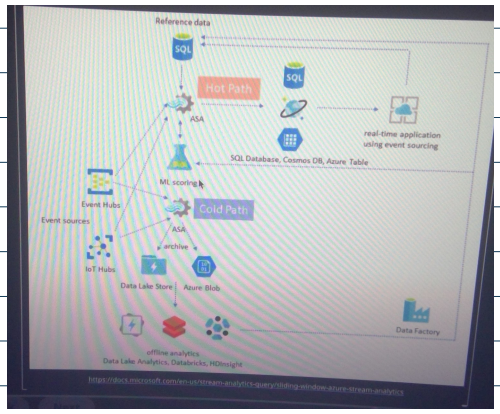
Azure Stream Analytics Job:-

- Pricing
  * Streaming units consumed
  * Amount of data processed
- Input, query and output
  * Inputs
    - IoT Hubs
    - Event Hubs
    - Blob storage
  * Outputs
    - Store and save results
  * Query language
    - Based on SQL
    - Windowing
      - sliding
      - Tumbling

- Windowing
  - sliding
  - Tumbling
  - Hopping

Azure Stream Windowing functions

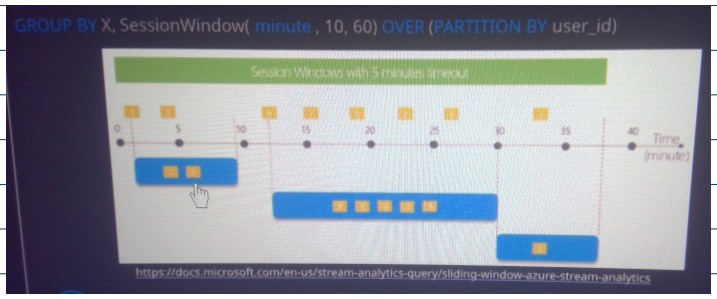
- Allows for functions to be run on sets of streaming data
- All outputs results at the end of the window
- Can be combined with highly complex solutions



Session Windows

- Group events that arrive at similar times, filters out periods of times where there is no data
- Windows are constantly re-evaluated
  * based on when events enter or exit the window
  - Max window size is 7 days

Group by x, SessionWindow (minute, 10, 60) over (partition by user_id)



Tumbling Windows :

- Repeating
- Non-overlapping
- Events cannot belong to more than one tumbling window

Group by x, TumblingWindow (Duration (minute, 10), Offset (milliseconds, -1))

Hopping Windows

- Hop forward in time by a fixed period
- Windows that can overlap

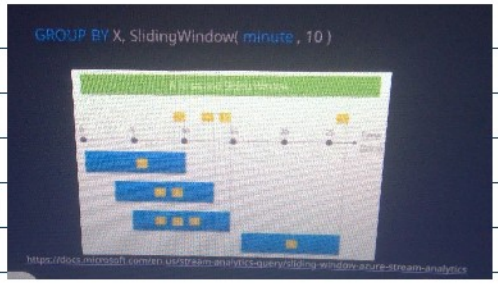
Max Window size is 7 days

Group by x, HoppingWindow (Duration (minute, 10), Hop (minute, 10), offset (milliseconds, -1))

Sliding Windows

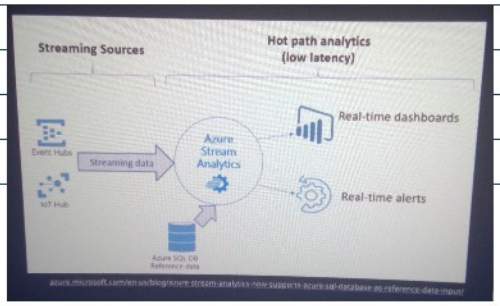
- Output produced only when an event occurs
- Every window has atleast one event
- Window continuously move forward by an epsilon

Group by x, SlidingWindow (minute, 10)



Using Reference Data Lookups in Stream Analytics

- Also known as lookup table
- Finite dataset that is static or slowly changing
  * Reference data has maximum size of [redacted]
- Can be stored in memory for low-latency stream processing
- Azure SQL Database and Azure Blob storage



Azure SQLDB and Azure Blob storage

Azure SQLDB -
stored as snapshot in memory for processing

Azure Blob storage
- modelled as a sequence of blobs in ascending order
  - defined in input configuration

Develop Integration Solutions

Tuesday, 4 August 2020 10:03 PM

Azure Data Factory

Cloud-based data integration service
Create data-driven workflows in the cloud
Orchestrate and automate data movement and transformation

Core concepts

- Pipeline
  * Logical grouping of activities
  * Activities perform a task

- Activity
  * Processing step in a pipeline
  * 3 types of activities
    - Data movement
    - Data transformation
    - Control

- Datasets
  * Data structures within the data stores

- Linked services
  * Connection string needed to connect to data

Data Factory Triggers

Types of triggers
- scheduled
  * Wall-clock schedule

- scheduled
  * Wall-clock schedule
  * Can only trigger future-dated loads
  * Many-to-many relationships

- Tumbling windows
  * Periodic intervals
  * Fixed size, non-overlapping, contigious time intervals
  * Can trigger past and future dated loads
  * One-to-one relationship

- Event based
  - Responds to an event in Azure Blob storage

Monitor Data Storage

Tuesday, 4 August 2020 10:20 PM

Monitor Storage Account in Azure Portal

Configuration

- Azure Storage Analytics
  * Provides metrics for
    - storage services
    - Logs for blobs, queues, tables
  * Useful for
    - Tracking requests
    - Analyzing usage trends
    - Diagnosing issues with storage
  * Must be enabled for each service
    - Portal, client library or REST API
  * Aggregated data is collected and stored in well-known blobs (logging) or well-known tables (metrics)
  * 20 TB capacity

Pricing

- Billable events
  - Requests to create blobs for logging
  - Requests to create table entities for metrics
- Logged operations and status messages

Azure Storage Metrics in Azure Monitor

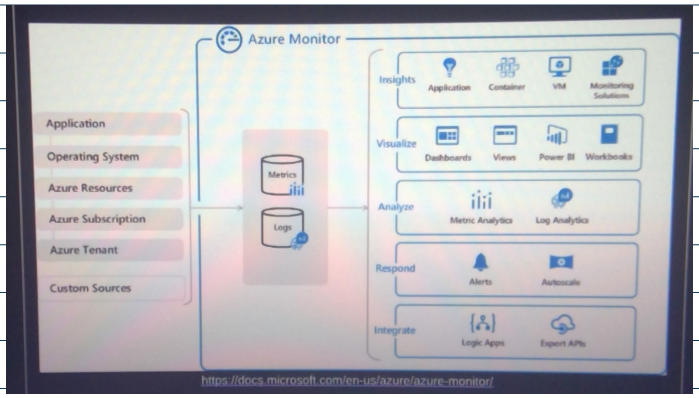
Azure Monitor

- Centralized, consolidated monitoring for all Azure resources
- Assists with troubleshooting issues
- Diagnose, trace, and debug failures in near real time
- Optimize performance, reduce bottlenecks and reduce costs

Types of data collected / Tiers

- Application monitoring data - performance and functionality of code
- Guest OS monitoring data - operating system
- Azure resource monitoring data - operations of an Azure resource
- Azure subscription - overall health and operation of Azure and your subscription
- Azure tenant monitoring data - tenant-level Azure services monitoring

- Azure subscription - overall health and operation of Azure and your subscription
- Azure tenant monitoring data - tenant-level Azure services monitoring



Logs

Metrics

Alerts in Azure Monitor

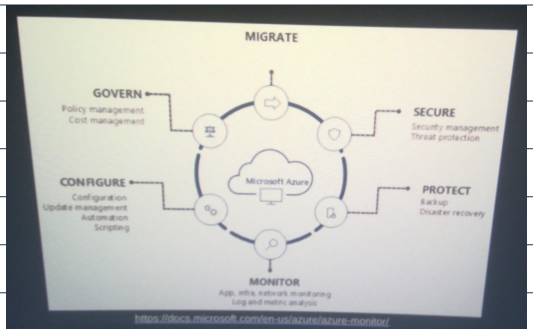
- Proactive notifications for critical conditions
- Attempt to take corrective action

Action Groups

- Collection of notification preferences
- Over 2000 action groups possible in a subscription
- Notifications by email or sms
- Core properties
  * Name
  * Action type
  * Details

Autoscale

Autoscale based on user-defined rules using Azure Monitor metrics



Monitoring SQL Database and SQL Data Warehouse

Monitoring database performance

Key metrics to be monitored

Monitoring database performance

Key metrics to be monitored
- CPU usage
- Wait statistics - why are queries waiting on resources?
- I/O usage - I/O limits of underlying storage
- Memory usage - memory available proportional to vCore number

Azure SQL Database includes tools and resources to monitor, troubleshoot, and fix potential performance issues

Tools:

Metrics Chart:
For DTU consumption and resources approaching max utilization

SQL Server Management Studio:
Performance dashboard to monitor top resource-consuming queries

Query Performance Insights:
Identify queries that spend the most resources (single and elastic)

SQL Database Advisor:
View recommendations for creating and dropping indexes, parameterizing queries, and fixing schema issues

Azure SQL Intelligent Insights:
Automatic monitoring of database performance

→ Query Performance Insights works with single and elastic pool types

SQL Data Warehouse

Most of the alerts are derived through Azure Monitor

Cosmos DB Monitoring

3 ways to gather metrics
- Performance metrics on the metrics page
- Performance metrics using azure monitoring
- Performance metrics on accounts page

Setting up alerts in portal are configured through Cosmos DB alerts rules

Setting up alerts in portal are configured through Cosmos DB alerts rules

Monitor Data Processing

Wednesday, 5 August 2020 11:31 pm

Monitoring Stream Analytics

Monitoring

Jobs can be monitored
- Portal
- PowerShell
- .Net SDK
- Visual Studio

Metrics available for Monitoring
- A variety of metrics can be monitoring
- Microsoft lists 4 sample scenarios
  * Streaming Units (SU) % utilization
  * Runtime errors
  * Watermark delay → 5 seconds delay is default for watermark
  * Input deserialization error
    ↳ Eg. malformed input in JSON document.

Setting up alerts in the portal

- Alerts configuration on job page
- Alerts are configured by job
- Notifications are customizable

Monitor Azure Data Factory

Monitor visually in Data factory

- Monitor pipeline runs
  * No code required
  * Monitor activity name, type, run start, duration status, activity inputs, activity outputs, and any errors.
  * Configure alerts by defining logic, severity, and notification type
  * Only 45 days of stored run data

Monitoring with Azure Monitor

- Persist data to storage account for data beyond 45 days
- Enabled through portal or REST APIs

Use Azure Monitor for

- Complex queries
- Monitoring across multiple data factories

Monitoring SDKs

Monitor Azure Databrics

Ganglia Monitoring System

- scalable monitoring system built into Databrics by default
- default collection every 15 mins
- can view snapshot or Live data

Azure Monitor

- One of the best solution for managing log data
- No native support for sending log data
  - ↳ Install Dropwizard metrics library
- Utilize Grafana dashboards
  - * Open source visualization platform
  - * Configure Azure Log Analytics workspace
  - * Deploy and configure Grafana

Manage Optimization

Thursday, 6 August 2020 12:05 am

Optimize SQL Data Warehouse Performance:

- check Advisor recommendations
- Polybase
  - use Polybase for loading and exporting data
  - use CREATE TABLE AS SELECT
- HASH - distribute large tables
  - choose correct distribution type
- Rule of 60
  - Don't over-partition → Experiment for partitioning to determine workload
- maximize throughput by breaking gzip into 60+ files
- shrink it down
  - minimize transaction sizes
  - minimize column sizes

Optimize Data Lake Performance

Configure toolset for maximum parallelization

Powershell - PerfileThreadCount, ConcurrentThreadCount
AdlCopy - Azure Data Lake Analytics units
Distcp - m (mapper)
Azure Data Factory - parallel copies
Sqoop - fs.azure.block.size, -m (mapper)

Dataset structure

- Keep filesize between 256 MB and 100 GB whenever possible
- Organize file/folder structure to optimize for larger file sizes

Optimize Azure Stream Analytics

Streaming Units (SU)

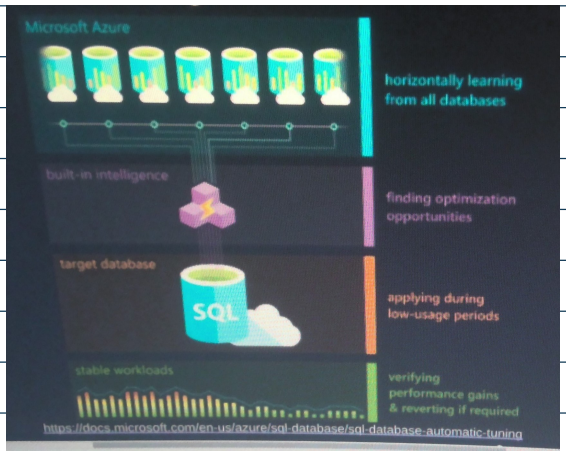
- Computing resources allocated to job
- Capacity is key
- Review SU% metric, 80% should be red line
- Start with 6 SUs for queries not using PARTITION BY

Parallelization

- Partitioning is key for optimization
  - Inputs are already partitioned
  - Outputs need to be partitioned
- Events should go to the same partition of the input
- PARTITION BY should always be used in all steps
- Input partition must equal output partition

Embarrassingly parallel jobs

Optimize SQL Database Performance



⇐ Automatic tuning

Performance overview

- summary of database performance and first stop for optimizations

Performance Recommendations

- Indexes to create or drop
- Database level schema issues
- Queries that can benefit from being converted into parameterized queries

Query Performance Insights

- Insights into DTU consumption
- CPU consuming queries
- Drill down into details of query results

Manage Data Lifecycle

Blob storage Archive

Hot, cool & Archive tiers

Provides durability, security and global reach

Lifecycle Management

- Transition blobs to optimize costs
- Delete blobs at the end of their lifecycle
- Define governance rules
- Apply rules to containers