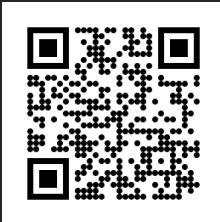


Understanding Fabric Capacities

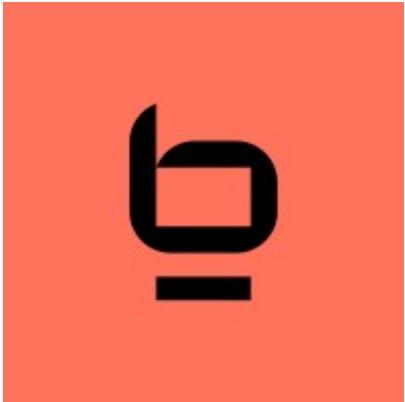
Benni De Jagere



Slides



Sponsors





Benni De Jagere

Senior Program Manager | Fabric Customer Advisory Team (FabricCAT)



Fabric CAT

.be Member

@BenniDeJagere

/bennidejagere

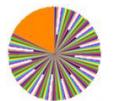


sessionize

/bennidejagere

/bennidejagere

#SayNoToPieCharts

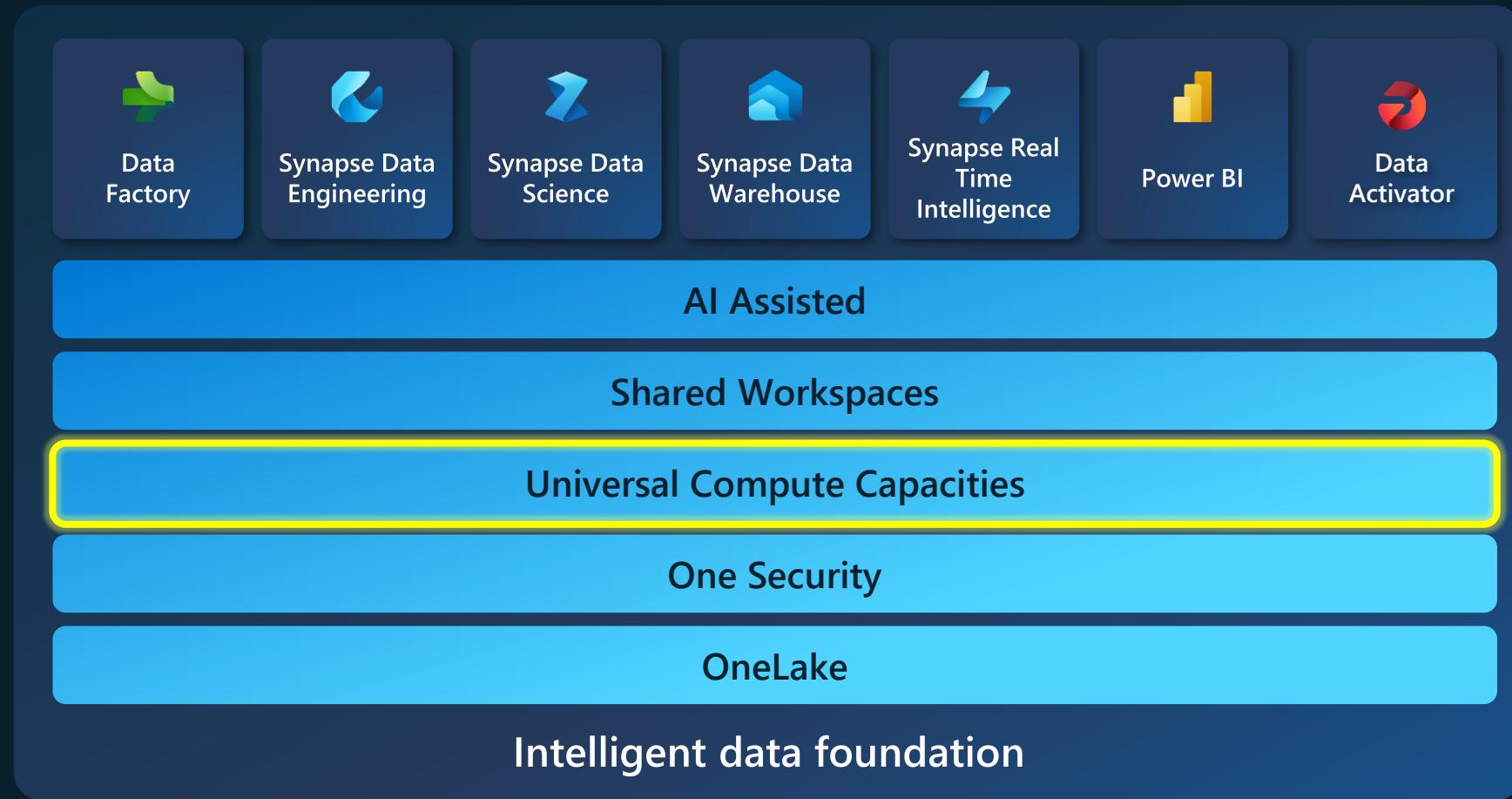




Fabric Capacities Introduction



Microsoft Fabric



Single...

- Onboarding and trials
- Sign-on
- Navigation model
- UX model
- Workspace organization
- Collaboration experience
- Data Lake
- Storage format
- Data copy for all engines
- Security model
- CI/CD
- Monitoring hub
- Governance & Capacity Metrics**
- Data Hub

Capacities are a shared resource

Shared across workloads

A single capacity is providing the compute power for all Fabric workloads.

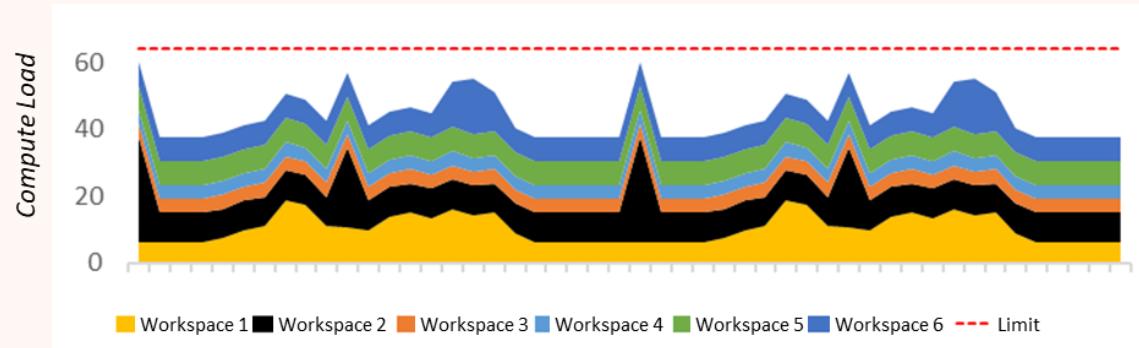
There is no need to allocate compute for each workload separately.



Shared Across Projects

A single capacity typically supports dozens of separate projects simultaneously, each managed in its own workspace.

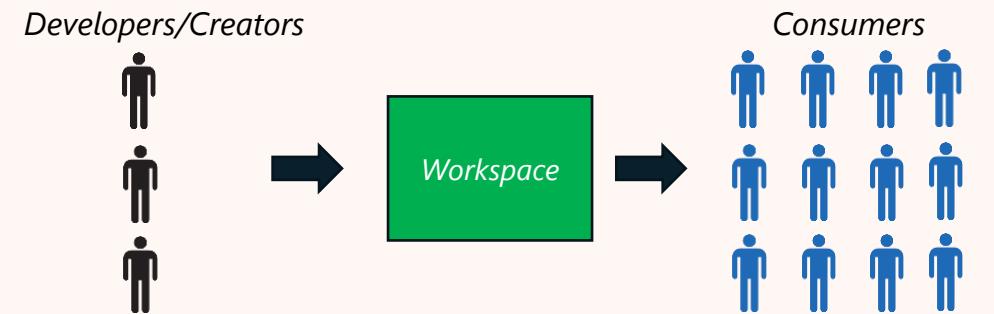
It is rare to have a capacity dedicated to a single project



Shared across users

For each project, many developers will share a workspace where collaborative development and consumption at scale is managed.

Each creator can provision any artifact and run any job without the need for any pre-approval or planning

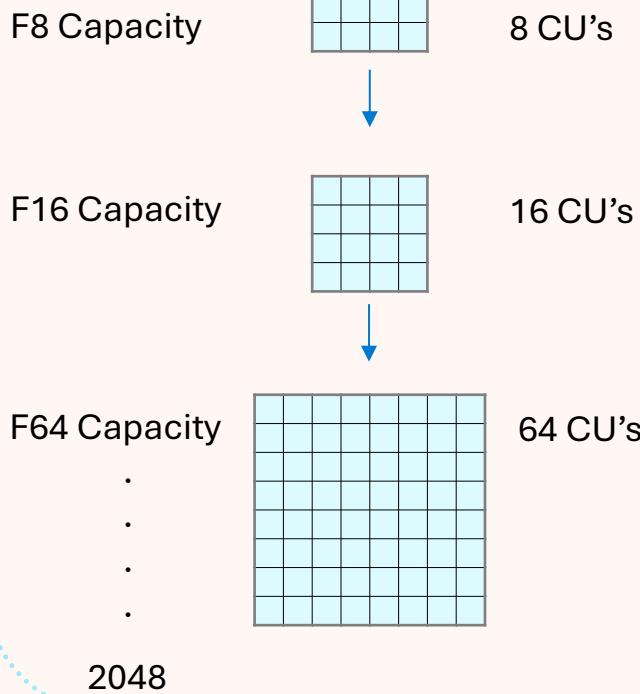


Capacities are flexible building blocks for growth

Capacities can be configured in endless ways to meet scale, usage and governance requirements while tuning to minimize TCO and performance goals

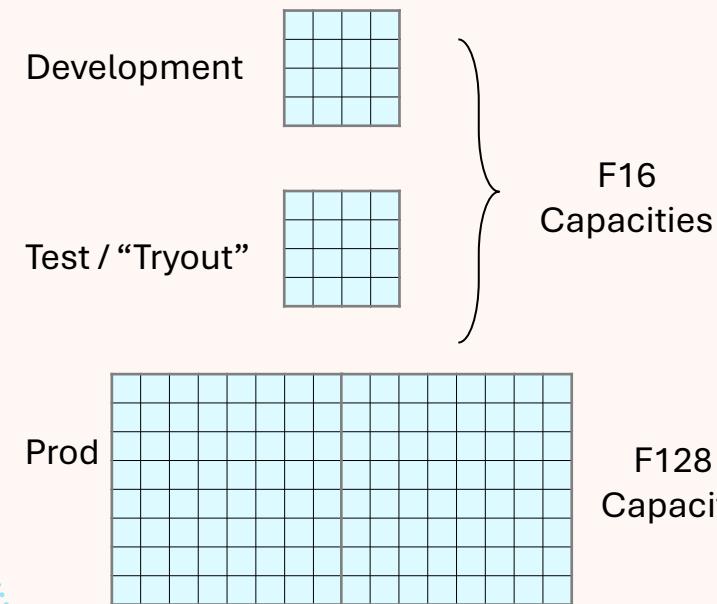
Scale Vertically

Increased capacity size provides more throughput



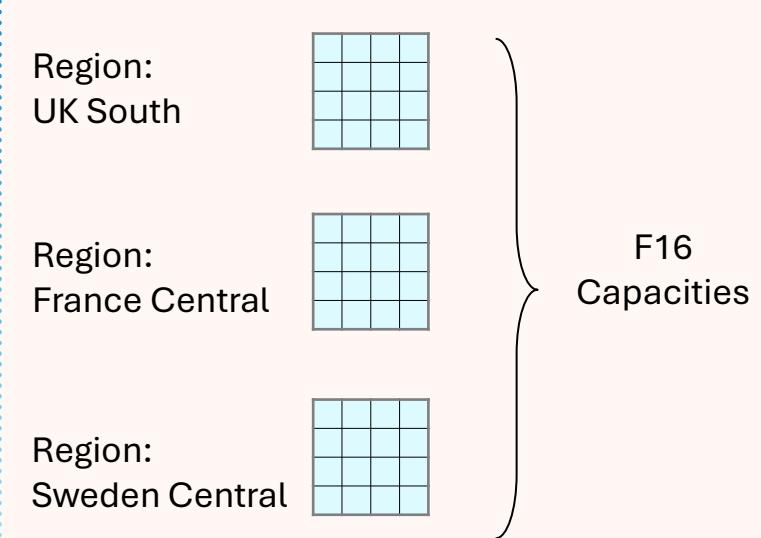
Scale Horizontally

Scale horizontally using the benefits of modular design for hardened isolation and governance



Regional Availability

Use different capacities for different regions to support GDPR / Data residency requirements



Provisioning and Deploying Capacities

Purchased in Azure

- **Purchased** either as a PAYG or RI resource
- **Provisioned with a certain amount of compute** units, analogous to CPU cores.
- The **more capacity units are provisioned, the more load** the capacity can support
 - Multiply SKU size by 30s to match platform evaluation in metrics app
- Capacities are **priced at a fixed hourly rate**, based on capacity units provisioned
- The RI commitment (1-year reserved instance) enjoys a **41% discount**

Universal Compute Capacities SKU Sizing

SKU	Capacity Units (CU)	CU's (per 30s)	Power BI SKU	Power BI V-cores
F2	2	60	-	0.25
F4	4	120	-	0.5
F8	8	240	A1	1
F16	16	480	A2	2
F32	32	960	A3	4
F64	64	1920	P1	8
F128	128	3840	P2	16
F256	256	7680	P3	32
F512	512	15360	P4	64
F1024	1024	30720	P5	128
F2048	2048	61440	-	256

Provisioning and Deploying Capacities

Deployed to Regions

- Each capacity **resides in a specific region of the buyers' choice** where both the data & compute reside
- **Workspaces are assigned to a capacity** that provides the compute and storage for all the workspace artifacts
- Multiple capacities can be purchased, deployed and managed by **different owners** residing in a single tenant allowing each business unit to pay for their own consumption

Tenant (Microsoft)

Capacity 1 (West Europe)

Workspace 1
(Data Science Team)

Item 1
Power BI
Semantic Model

Item 2
AI Function
Evaluation

Workspace 2
(Research)

Item 3
Spark Notebook

Item 4
ML Dataflow

Capacity 2 (Central India)

Capacity 3 (West US)



Bursting and
Smoothing

Smoothing intro and benefits

Load stabilization

Smoothing helps capacities self-stabilize by flattening large spiky loads into a smooth load profile, eliminating temporal spikes

Eliminates Scheduling contention

Large/scheduled Jobs usage (not execution) are smoothed over 24 hours, eliminating the need to decide the timing and order of job execution

Surge protection

Interactive operations smoothed over several minutes, preventing a single user with a very demanding query from hogging the entire capacity



What is Bursting?

Job acceleration

Bursting provides extra compute resources to jobs and queries to accelerate their completion

Go beyond

The extra resources of bursting allow jobs to **utilize far more resources than “face value”**

Instead of running a job on 64 CU and completing in 60 seconds, bursting could use 256 CUs to complete the job in 15 seconds.

Same amount of work, just completed faster

No hassle, No overload

Bursting is automatic when the system reasons it can accelerate the job by applying extra resources. No settings are required.

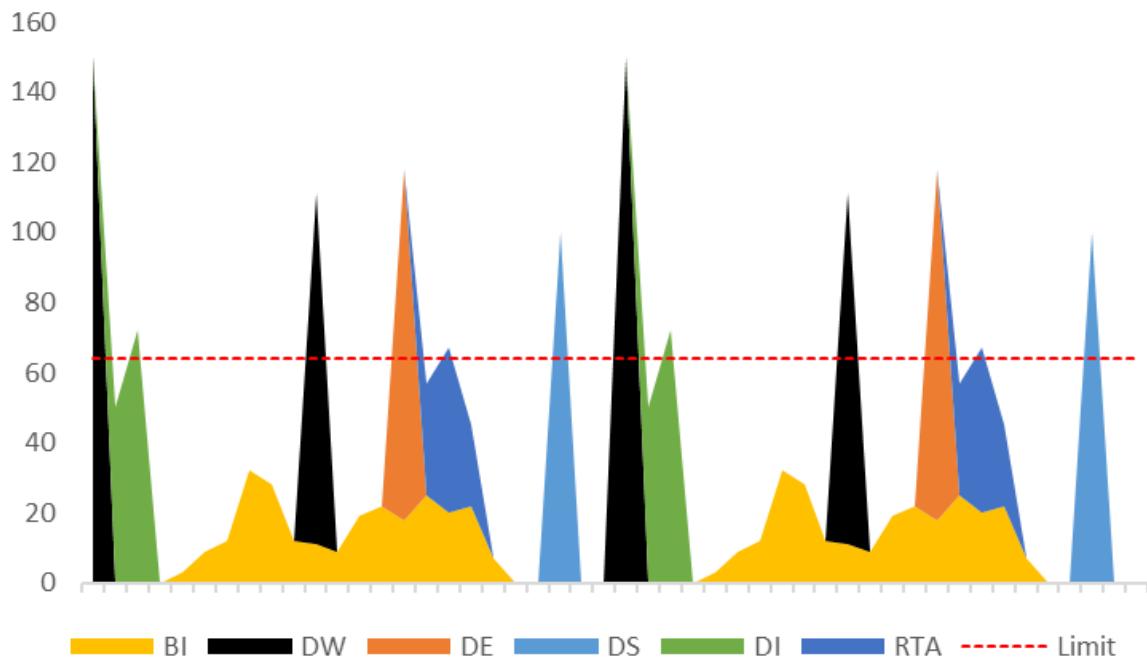
Bursting prevents an overload as the *smoothing* mechanism will always flatten the resource burst

Bursting and smoothing | before and after

Looking at an example of a 64 CU capacity, running multiple workloads over a couple of days...

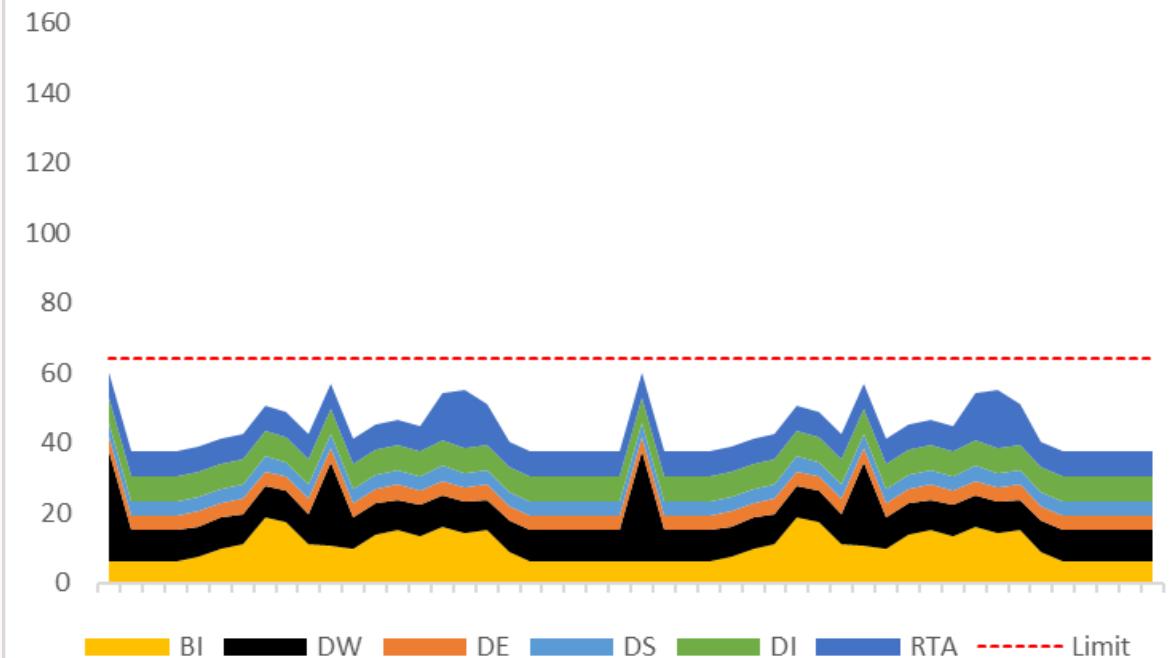
Before Smoothing

- Actual load as executed on the capacity before smoothing
- *Bursting* accelerates jobs execution by resource boosting
- The capacity could be overloaded 25% of the time
- Some of the overloads are more than 2x the limit
- There are periods of no/low usage

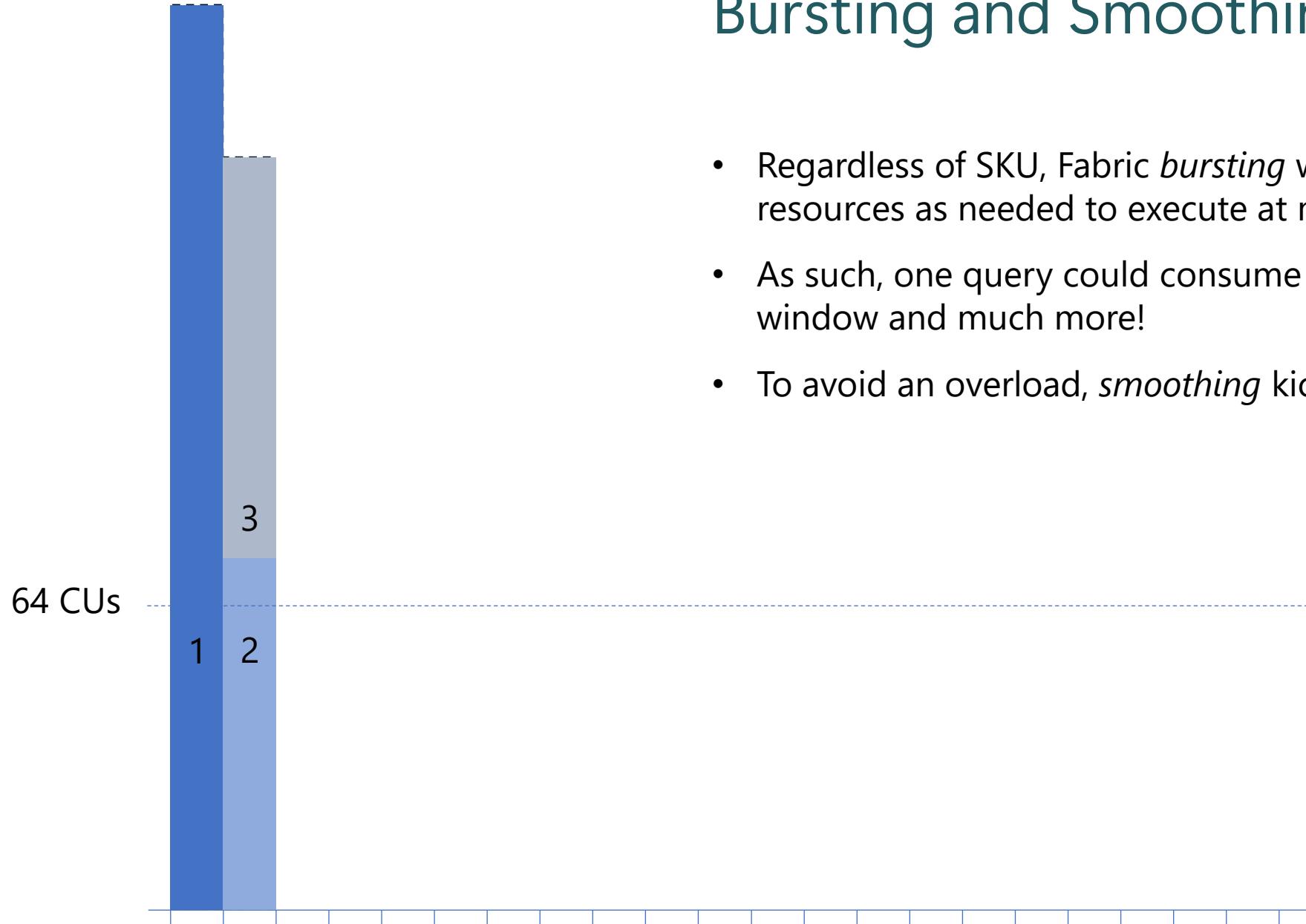


After Smoothing

- Shows the reported load (not runtime execution) against the capacity limits
- There is NO overload, and consumption is more stable
- The smoothing of usage fills in gaps



Bursting and Smoothing



- Regardless of SKU, Fabric *bursting* will automatically allocate resources as needed to execute at maximum performance
- As such, one query could consume all the quota of a single time window and much more!
- To avoid an overload, *smoothing* kicks in

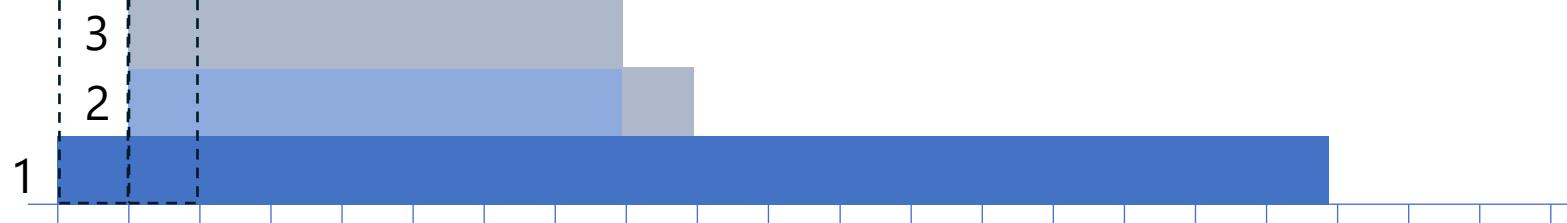
Bursting and Smoothing

Actual execution

- No one (or few) queries can trigger an overload
- Instead of allowing runaway queries to create a local overload, Fabric smooths the queries reported usage to future time windows
- Kind of “Buy now, pay in the future” installment plan

64 CUs

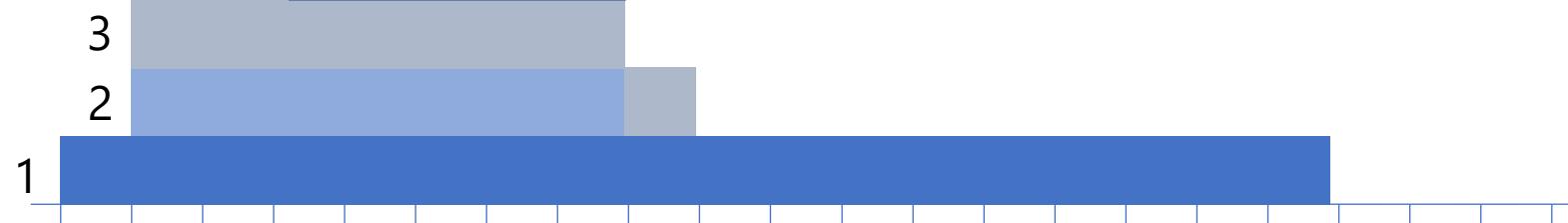
Reported load on CUs



Bursting and Smoothing

4 – large batch job

64 CUs



- Large batch processes traditionally were a threat to interactive queries as they would overload the compute resource
- DBAs traditionally had to carefully schedule these jobs to off-hours to avoid interference with interactive user experiences

Bursting and Smoothing

4 – large batch job

Actual execution

64 CUs



- A similar “installment plan” logic is applied for batch jobs
- But for batch jobs the smoothing is applied uniformly for the next 24 hours
- This completely liberates the DBA from any consideration of job scheduling. The load will be uniform regardless of the schedule.
- Most importantly, regardless of when batch jobs are scheduled, there will not be any degradation on interactive query performance

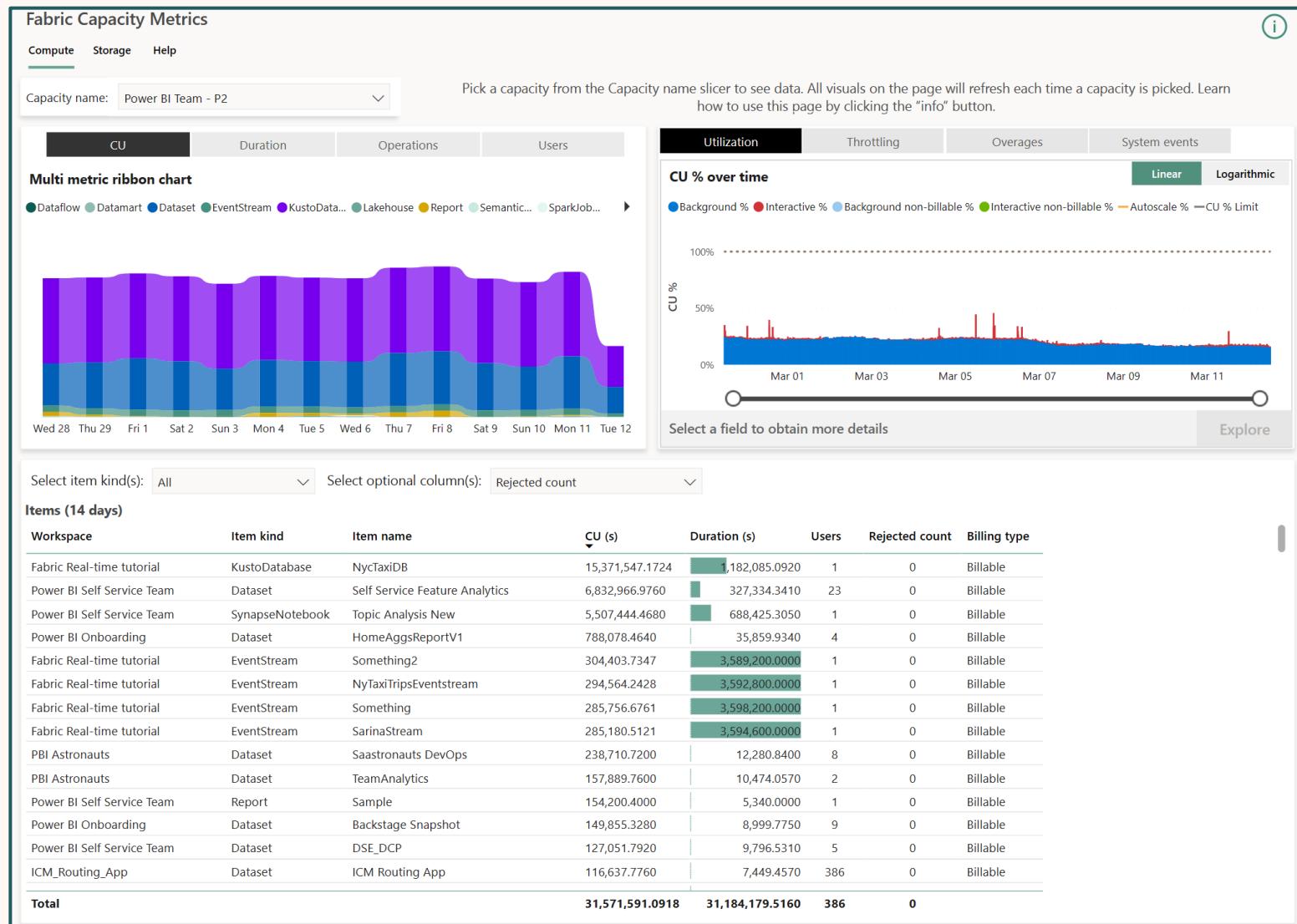


Monitoring with Capacity Metrics

Capacity Metrics

Monitor Capacities and Plan capacity scale-up with confidence

- Tenant wide visibility into capacity usage for all Fabric experiences
- Identify resource usage trends and their impact to autoscale & throttling
- View preview workload usage alongside production workloads to make data-driven capacity sizing decisions

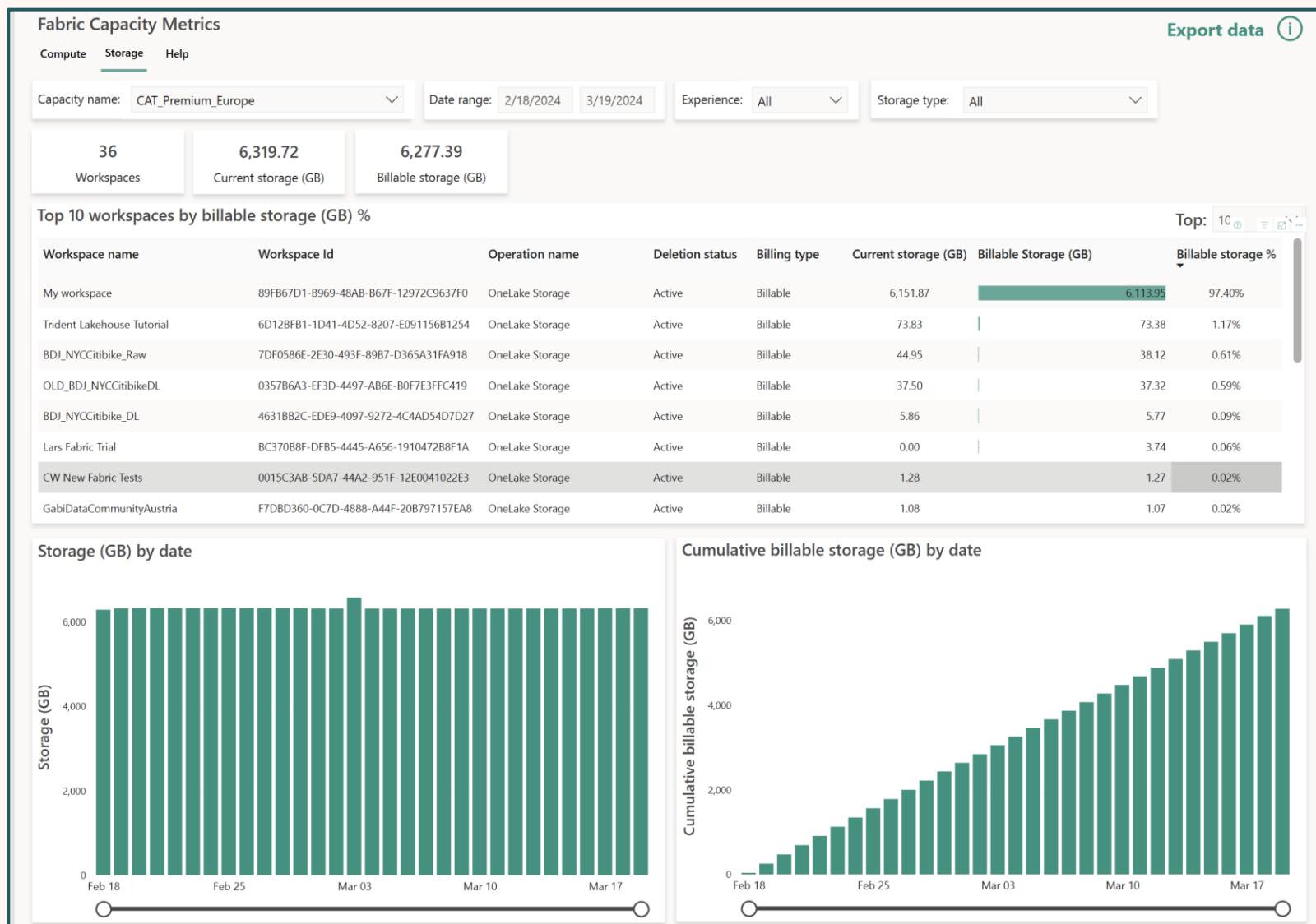


Capacity Metrics

Monitor OneLake consumption

Measure the trends of workspace storage consumption against capacity limits, by day or hour

Reconcile costs with internal chargeback processes



Fabric Capacity Metrics

Pages < File Export Share Chat in Teams Get insights Subscribe to report Set alert ... Copilot Filters

Fabric Capacity Metrics

Compute Storage Help

Capacity name: CAT_Premium_US

Pick a capacity from the Capacity name slicer to see data. All visuals on the page will refresh each time a capacity is picked. Learn how to use this page by clicking the "info" button.

CU Duration Operations Users

Utilization Throttling Overages System events

Multi metric ribbon chart

Dataflow Dataset Lakehouse Pipeline Report SemanticModel SynapseNotebook Warehouse

CU % over time

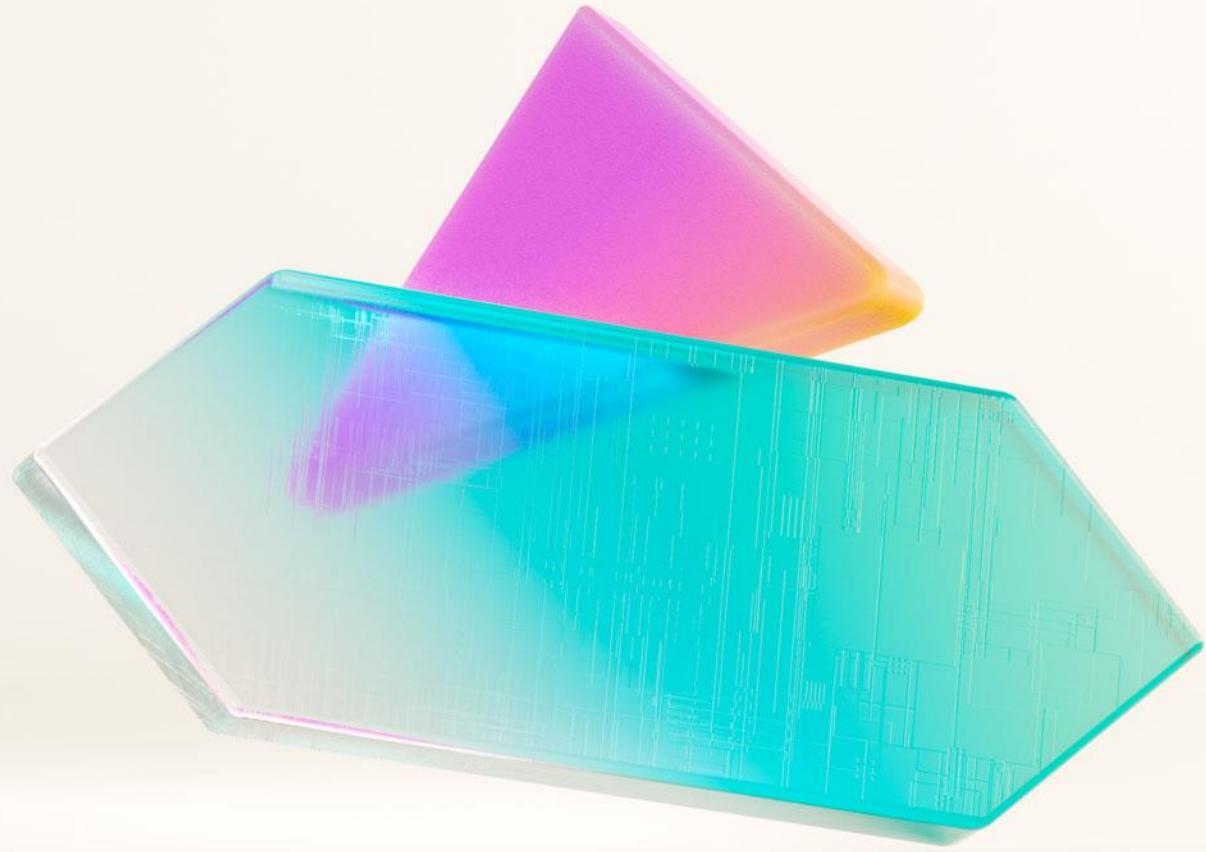
Background % Interactive % Background non-billable % Interactive non-billable % Autoscale % CU % Limit

Select a field to obtain more details Explore

Select item kind(s): All Select optional column(s): Multiple selections

Items (14 days)

Workspace	Item kind	Item name	CU (s)	Duration (s)	Users	Rejected count	Overloaded minutes	Successful count	Performance delta	Billing type
Fabric for Power BI users (alpowe...)	Dataflow	Dataflow 1Nashville	902,712.8000	14,585.3890	1	0	0.0000	97		Billable
Data Factory in an Hour (old)	Pipeline	dfPipeline	475,200.0000	19,858.7140	1	0	0.0000	1247		Billable
Analytics at the speed of Direct L...	Dataset	Metrics	359,352.5440	11,100.0940	3	0	1.3333	1095		Billable
Day After DIAD (alpowers)	Dataflow	OnlineSalesDataflow	355,036.3520	14,154.7430	1	0	0.0000	12		Billable
DADIAD Sample	Dataflow	DataflowTest	343,605.4880	15,609.4410	1	0	0.0000	12		Billable
Survey_test	SynapseNotebook	Survey_Notebook	214,899.6685	24,328.8260	2	0	0.0000	42		Billable
Mahoney_MetricsSnapshots	SynapseNotebook	MetricsAppSnapshots	184,471.7595	20,047.6250	1	0	0.0000	0		Billable
Analytics at the speed of Direct L...	SynapseNotebook	Notebook 1	165,965.6310	19,895.5380	2	0	0.0000	0		Billable
BDJ_NYCCitibike_Base	Warehouse	NYCCitibike_BASE	87,065.7860	50,125.5260	3	0	0.0000	3164		Billable
Lars - ready for demo	SynapseNotebook	Framing Demo	86,681.3000	8,158.5510	1	0	0.0000	0		Billable
CW Direct Lake Paging	Dataflow	CollectDMVData	83,097.0720	7,427.1870	1	0	0.0000	52		Billable
Fabric_Demo	Dataflow	Ih_Sample	82,095.6800	4,454.1890	2	0	0.0000	73		Billable
Fabric for Power BI users (alpowe...)	Dataset	WeirdDemo	77,008.5280	1,806.6590	2	0	0.0000	540		Billable
Mahoney_MetricsSnapshots	SynapseNotebook	Notebook 1	63,871.2960	7,408.3180	1	0	0.0000	0		Billable
Total			4,907,150.8628	461,929.2860	4	0	2.0000	1662399		

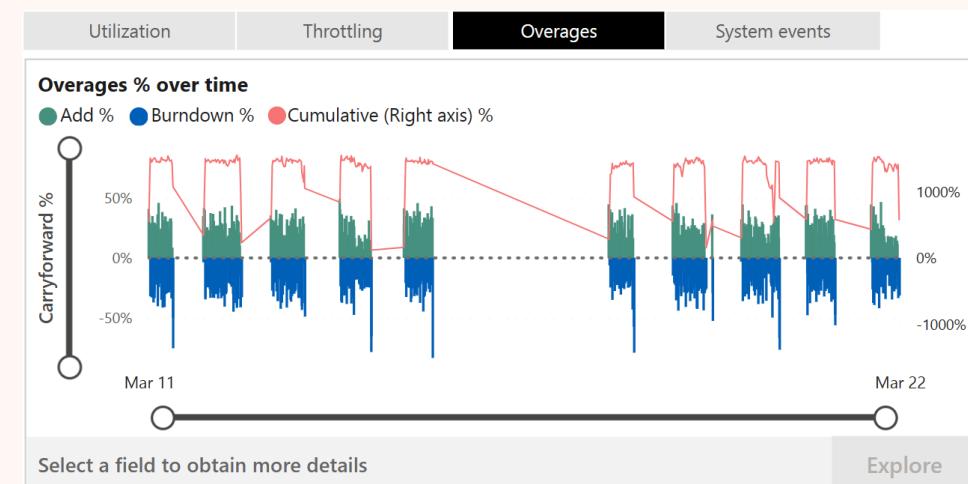
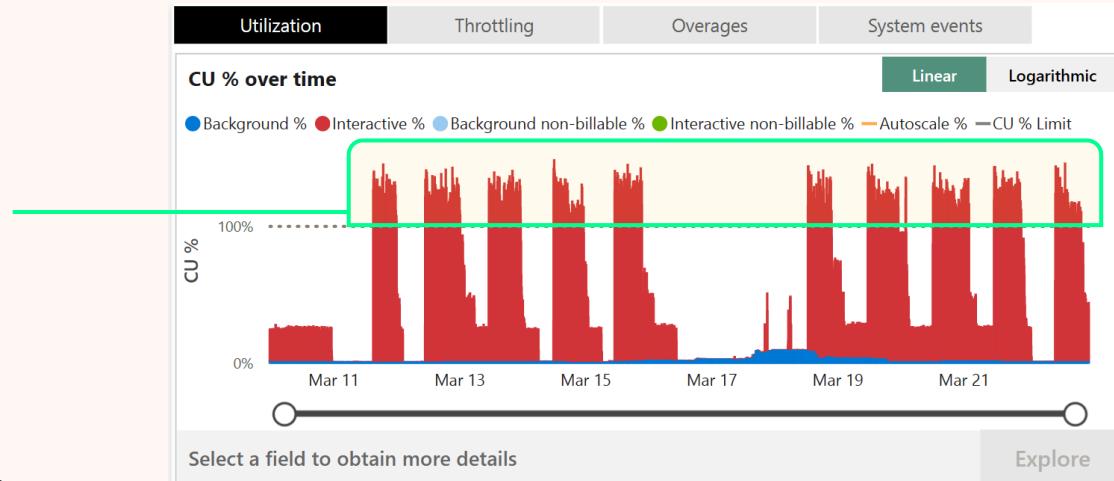


Capacity Throttling Policies

Throttling intro

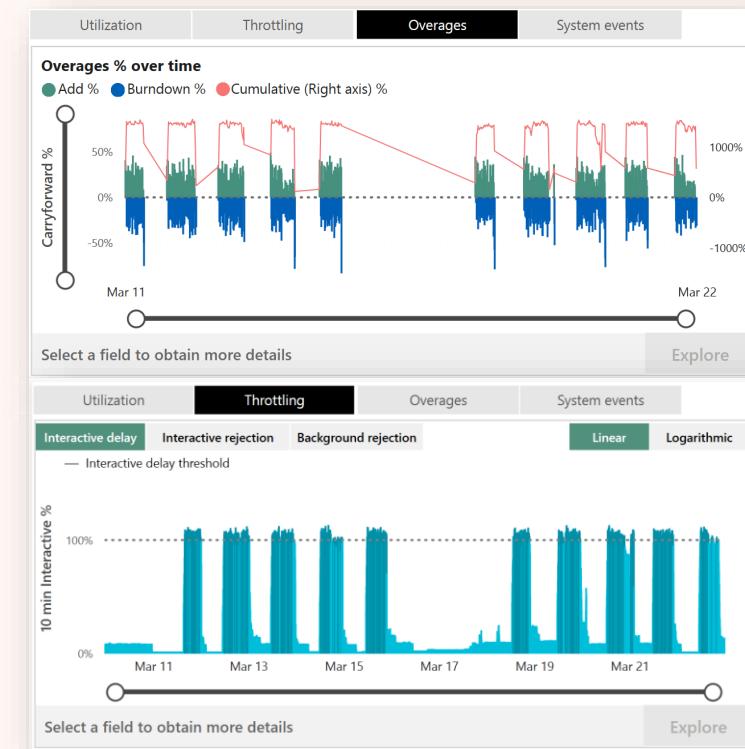
- Throttling is the platform policy for managing consumption that exceeds throughput is provided by SKU choice
- When workloads exceed the throughput of a capacity a cumulative debt is tracked to be burned down
- Cumulative debt is used to determine throttling policies and is burned down when resources are free

Overage Operation	Description
Overages - Added	<ul style="list-style-type: none">• Timepoint when job requests exceed the throughput of a capacity, overages are added to the cumulative buffer to burn down.• This graph simplifies identification of the optimal timepoint to load timepoint drill to analyze the user operations that contributed to an overage.
Overages - Burndown	<ul style="list-style-type: none">• Overages being reconciled when future capacity is free to burn down
Overages - Cumulative	<ul style="list-style-type: none">• The total amount of queued work on the capacity to be burned down in the future when the capacity is not fully utilized



Capacity throttling evolution for Fabric

- For Fabric, throttling policies were refined to deliver multiple benefits
 - Reduced throttling** for capacities that only experience occasional spikes
 - Added overage protection** – rejection policies prevent overloaded capacities from irrecoverable overload
 - Optimizations for long-running jobs:** We're optimizing the platform for long-running jobs, so if a job exceeds capacity limits, it will run to completion and the overage will be burned down against future capacity



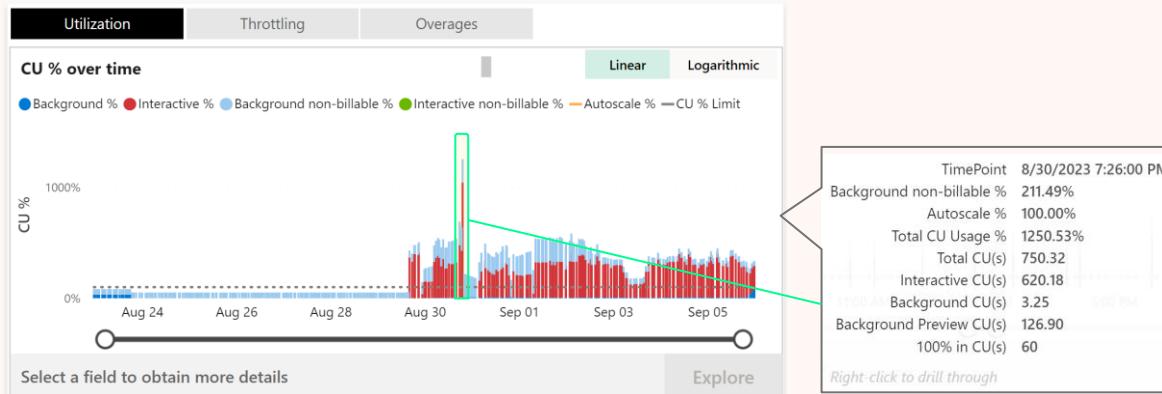
Smoothed Capacity - Future Use	Platform Policy	Customer Impact
<= 10m	Overage Protection	Jobs can consume 10 minutes of future capacity use without throttling
> 10m → <= 60m	Interactive Delay	User requested interactive type jobs will be throttled
> 60m → <= 24h	Interactive Rejection	User requested interactive type jobs will be rejected
> 24h	Background Rejection	User Scheduled background jobs will be rejected from execution



Capacity Planning with Capacity Metrics

Capacity planning case study - measurement

Start with a test or trial capacity to evaluate the load of specific Fabric Experiences i.e., Power BI Datasets, Spark Notebooks or a Datawarehouse



If usage is above the current capacity limits , choose the desired utilization rate to accommodate via capacity scale up

The interface shows a timeline for 8/30/2023 7:26:00 PM. A callout box highlights the CU % limit at 100.00% for that timepoint, with a total CU consumed of 1,249.04% and a limit of 60 CU(s).

Interactive operations

Item	Operation	Start	End	Duration (s)	Total CU (s)	Timepoint CU (s)	Throttling (s)	% of Base Capacity	Billing type
...	Query	8/30/2023 7:11:25...	8/30/2023 7:11:26...	10	900	30.00	0	50.00%	Billable
...	Query	8/30/2023 7:12:34...	8/30/2023 7:12:35...	14	1,110	30.00	0	50.00%	Billable
...	Query	8/30/2023 7:19:30...	8/30/2023 7:20:03...	32	509	29.98	0	49.97%	Billable
...	Query	8/30/2023 7:18:05...	8/30/2023 7:18:16...	10	958	29.95	0	49.91%	Billable
...	Query	8/30/2023 7:12:46...	8/30/2023 7:12:58...	12	986	29.90	0	49.84%	Billable
...	Query	8/30/2023 7:20:06...	8/30/2023 7:20:16...	10	980	29.71	0	49.52%	Billable
...	Query	8/30/2023 7:17:28...	8/30/2023 7:18:02...	33	531	29.54	0	49.24%	Billable
...	Query	8/30/2023 7:10:32...	8/30/2023 7:10:50...	17	944	29.50	0	49.17%	Billable
...	Query	8/30/2023 7:16:41...	8/30/2023 7:17:14...	33	530	29.49	0	49.14%	Billable
...	Query	8/30/2023 7:17:15...	8/30/2023 7:17:26...	11	1,000	29.43	0	49.04%	Billable
Tot..				13,812	13,894	619.94	0	1033.23%	

Fabric Capacity Metrics

Overview Help

8/30/2023 7:26:00 PM Timepoint

CU % CU % Limit

8/30/2023 7:26:00 PM

- CU % 1,249.04%
- CU % Limit 100.00%
- Autoscale % 100.00%

287 Interactive operations 100K Background operations F2 SKU 60 CU(s)

Load Capacity Metrics timepoint drill to analyze :

- Total CU's consumed : **749 CU(s)**
- Capacity Size : (F2)
- CU(s) available on your capacity : 60 CU(s)

Capacity planning case study – SKU selection

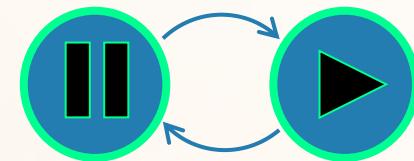
Universal Compute Capacities SKU Sizing

To accommodate a **749 CU(s)** load the admin can purchase an F32 capacity providing 960 CU(s) of throughput

SKU	Capacity Units (CU)	CU's (per 30s)	Power BI SKU	Power BI V-cores
F2	2	60	-	0.25
F4	4	120	-	0.5
F8	8	240	A1	1
F16	16	480	A2	2
F32	32	960	A3	4
F64	64	1920	P1	8
F128	128	3840	P2	16
F256	256	7680	P3	32
F512	512	15360	P4	64
F1024	1024	30720	P5	128
F2048	2048	61440	-	256



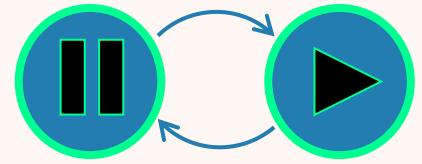
Pausing and Resuming Capacities



Pausing and Resuming Capacities

Why pause capacities?

- 1) It helps manage compute costs.
- 2) It clears any debt that has accumulated. Use it to quickly resolve throttling.



What does it do?

Workloads stop execution within 10 minutes of Pause action

New requests are not allowed to Start

Smoothed usage will be reconciled

Note: OneLake storage costs continue to be billed while a capacity is paused



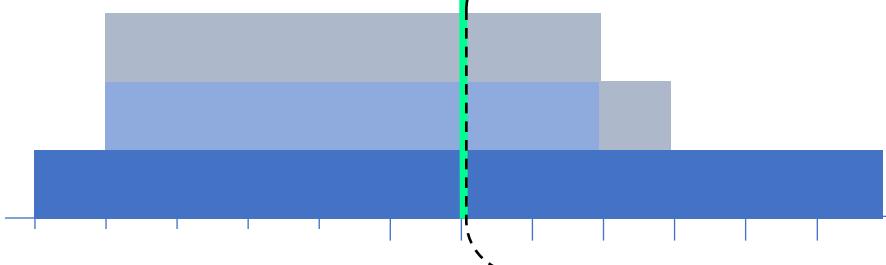
How Capacity Pause & Resume works

When a capacity is **paused**...

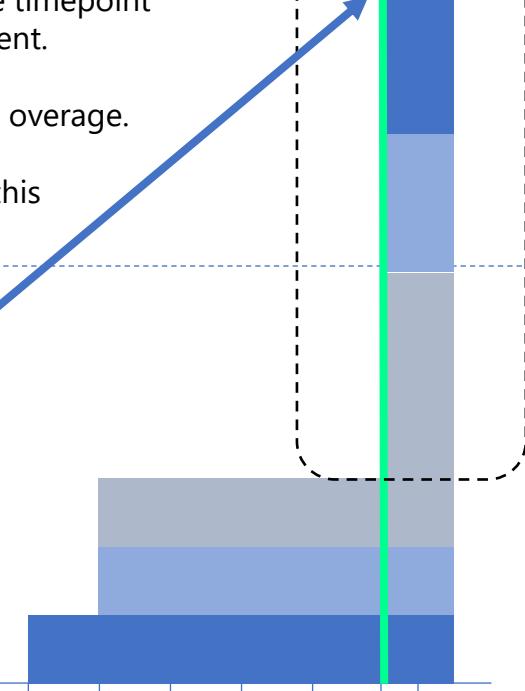
Pause event on
Capacity



64 CUs



Smoothed usage is **reconciled**.

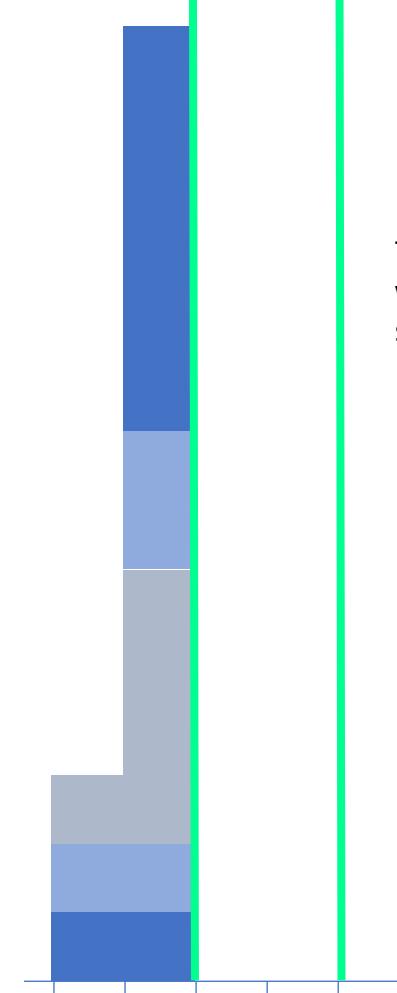


Total smoothed usage is shown as
compute utilization on the timepoint
directly after the Pause event.

PayGo Price applies to the overage.

A billing event is sent for this
consumed compute.

Later, it can be **resumed**.



The capacity starts
with zero utilization or
smoothed usage.

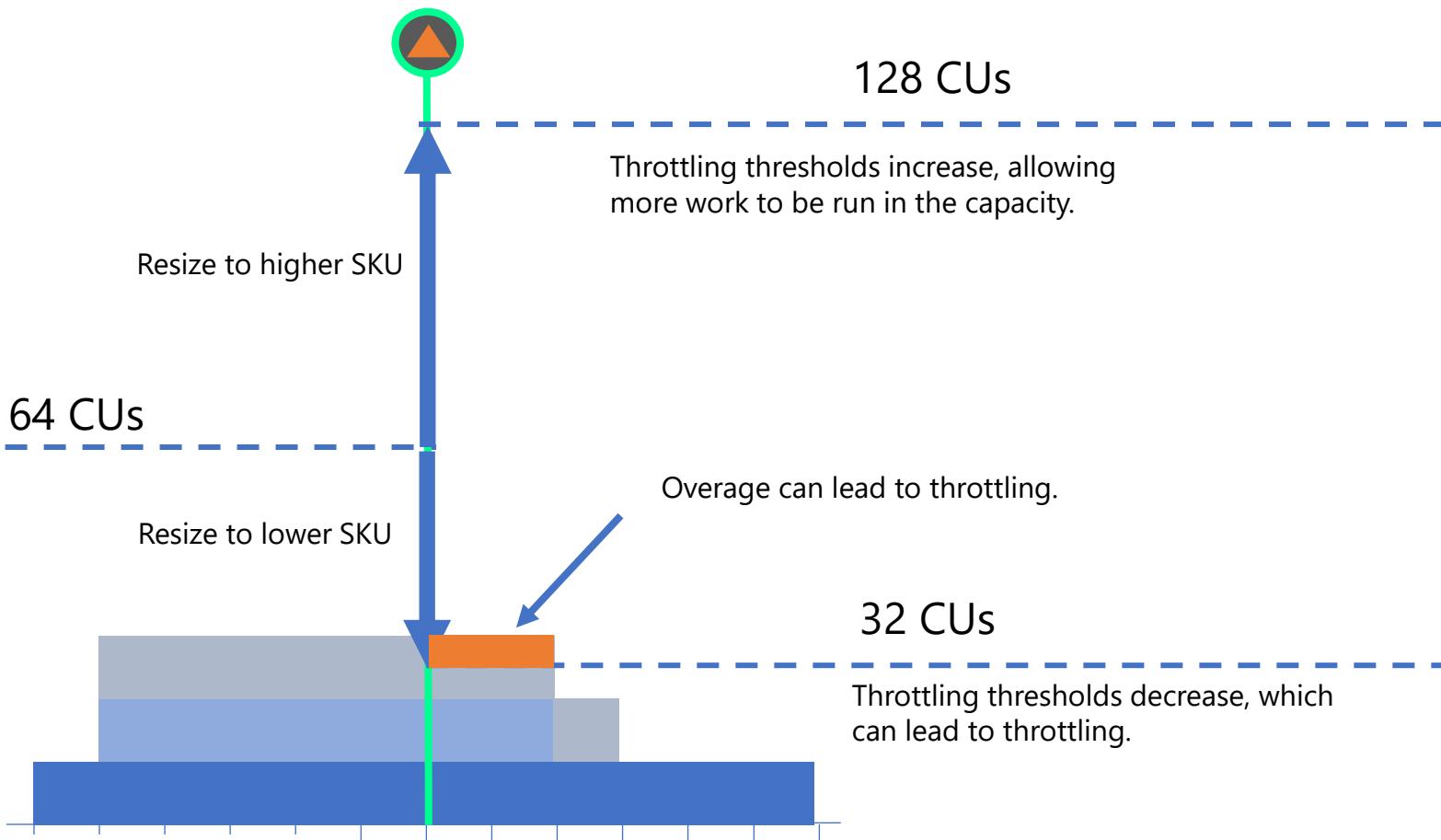
How Capacity Resize works

When a capacity is **resized**...

The allowed CUs per timepoint increase or decrease.

This changes the throttling allowed limits based on the new SKU's CUs and the throttling windows.

SKU Change



Key Insights

- Sizing up will incur the cost of the new SKU
- Sizing down could lead to more throttling
- Review your Throttling Thresholds before sizing down your SKU.

Bursting and Smoothing

Jobs Executed

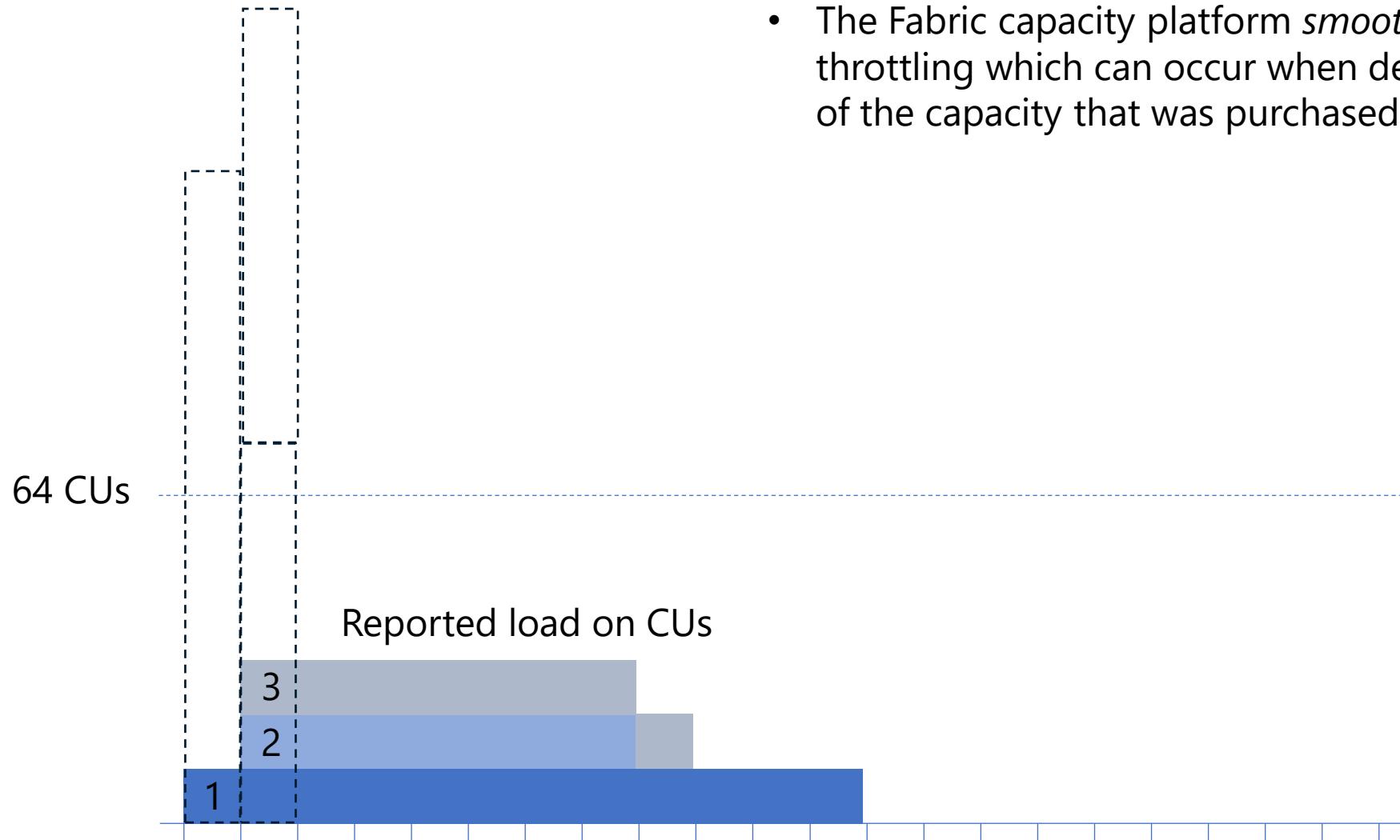


- Job execution in Fabric workloads happens on-demand via capacity powered compute engines
- Fabric *bursting* will automatically allocate resources as needed to execute at maximum performance

Bursting and Smoothing

Actual execution

- The Fabric capacity platform *smooths* usage out to reduce throttling which can occur when demand exceeds the throughput of the capacity that was purchased



Smoothing and Paused Capacities

Pause event on
Capacity

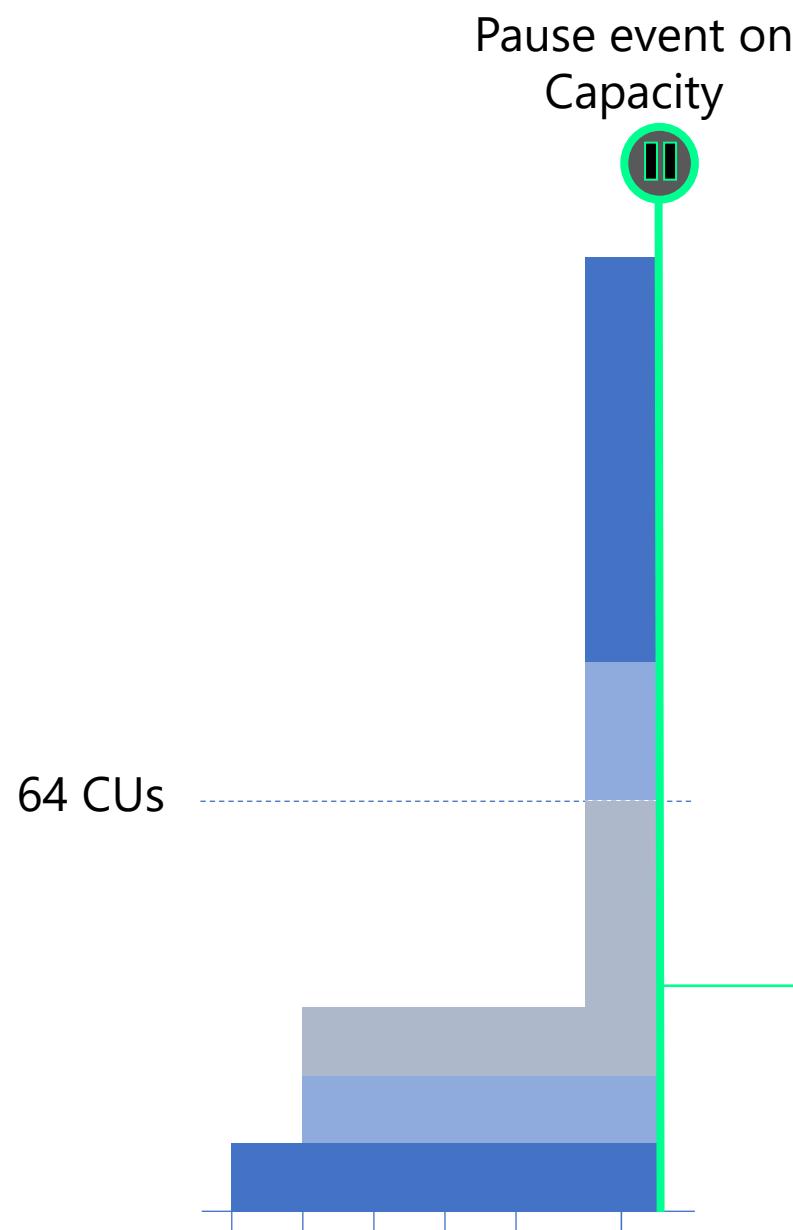


64 CUs

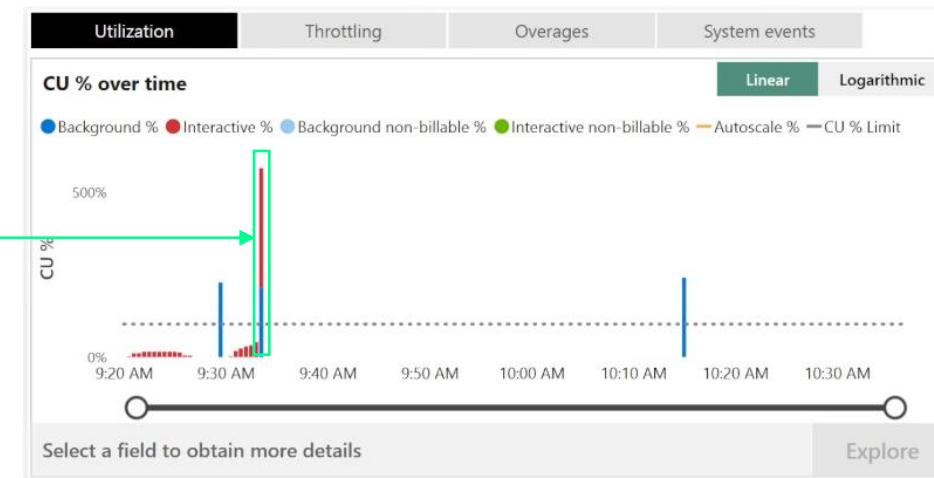


- When a capacity is paused...

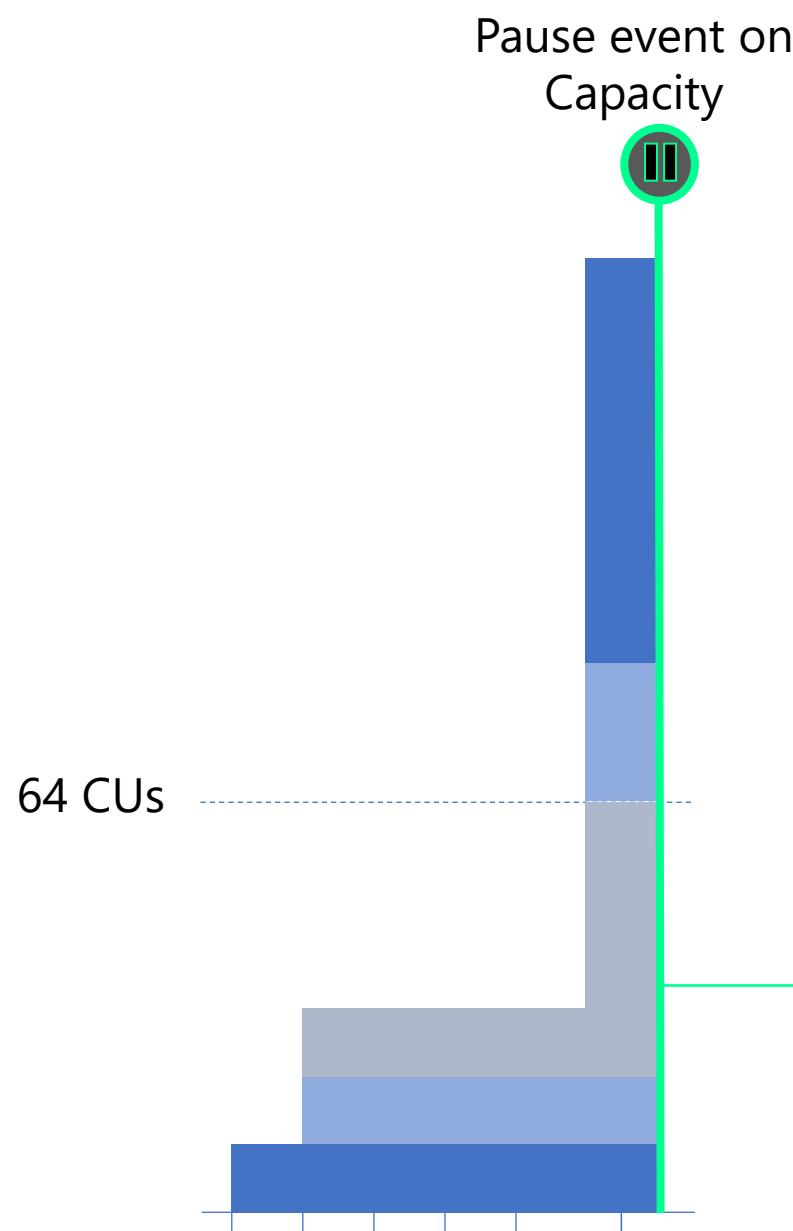
Smoothing and Paused Capacities



- When a capacity is paused...
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Reconciled usage will show up as a spike in capacity metrics



Smoothing and Paused Capacities



- When a capacity is paused...
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Pause events can be viewed in the new System events tab

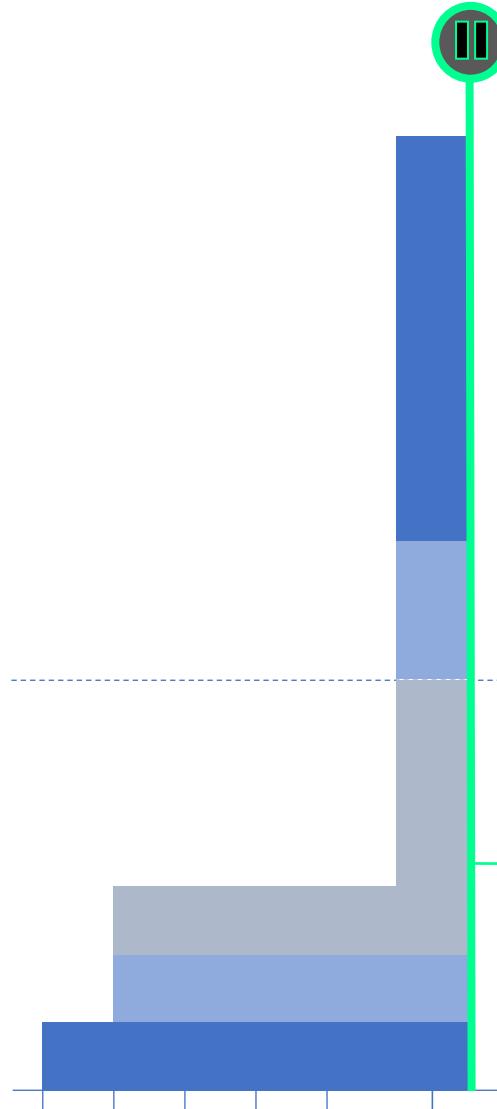
Utilization	Throttling	Oversages	System events
System events			
State transition time	Capacity state	Capacity state change reason	
12/13/2023 9:12:14 AM	Active	Created	
12/13/2023 9:29:12 AM	Suspended	ManuallyPaused	
12/13/2023 9:30:15 AM	Active	ManuallyResumed	
12/13/2023 9:33:29 AM	Suspended	ManuallyPaused	
12/13/2023 9:34:58 AM	Active	ManuallyResumed	
Select a field to obtain more details			Explore

Smoothing and Paused Capacities

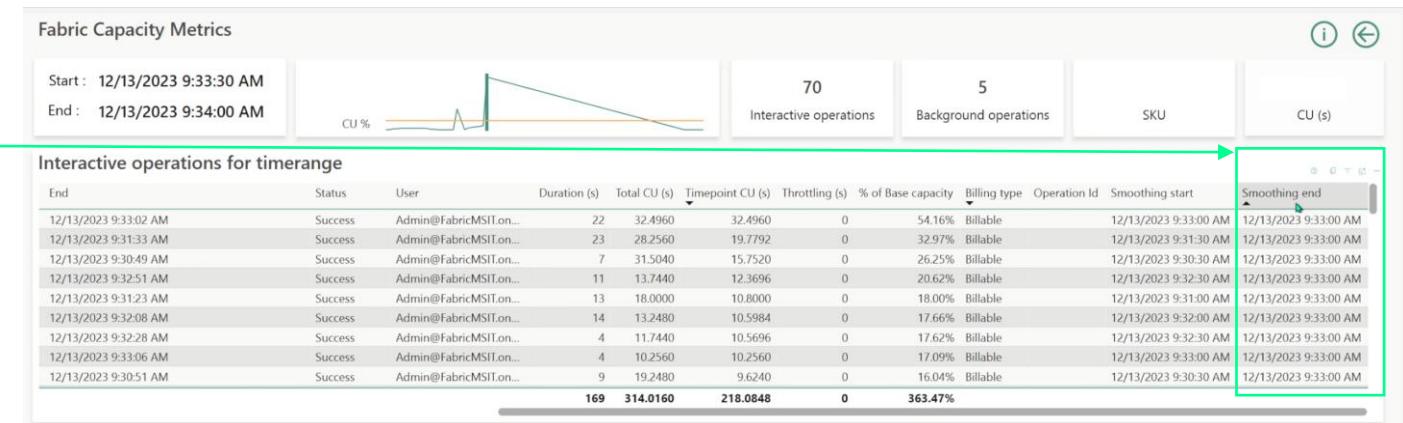
Pause event on Capacity



64 CUs



- When a capacity is paused...
- Usage that was smoothed into the future will be “reconciled” and charged against the capacity at the timestamp the capacity was paused
- Pause events timestamp is shown in the smoothing end field in timepoint drill views





Capacities Platform Roadmap

Capacity Management

Improved monitoring for large scale deployments with many capacities

Cross Capacity

Last 24 hour Last 1 hour Help

Capacity Name SKU Region

Capacities 15 Avg Utilization % 2.21 WoW: 189.56% ▲ # Throttled Capacities -- # Interactive Rejection Capacities -- # Background Rejection Capacities --

Choose a capacity from the visuals to explore cross-report drill-through options and examine the specific behavior of selected capacities at a more detailed level.

Capacity breakdown

Capacity Name	SKU	User	Avg Utilization %	Cumulative Debt	Throttling (s)	Throttling	P95 Interactive Delay	P95 Interactive Rejection	P95 Background Rejection	Usage Variance	Rejected	Failure	Cancelled
UVV8j6W4mUe0IJGeWkYbma	FT1	25	0.28	0.00	0.29	0.28	0.15	0.00	0	0	0	0	0
Trial-20240407T120129Z-xOm07SwFV kem&kDvehDwIA	FT1	32	10.35	0.00	12.17	11.99	7.07	0.43	0	0	0	0	0
Trial-20240408T093736Z-rrv4bmWgVUqk72T-yUQKQ	FT1	25	0.28	0.00	0.29	0.28	0.15	0.00	0	0	0	0	0
Trial-20240408T181028Z-FRKpQamMqkCHw36V2TBrg	FT1	32	10.35	0.00	12.17	11.99	7.07	0.43	0	0	0	0	0
Trial-20240409T183132Z-	FT1	25	0.28	0.00	0.29	0.28	0.15	0.00	0	0	0	0	0

1h 24h 7d 30d

Utilization % Throttling Duration (s) Invalid Failure Cancelled InProgress Rejected Successful Throttling Hits Background Hits Interactive Hits

Metrics 30 Day

0.2M
0.0M
Apr 09 Apr 11 Apr 13 Apr 15

Capacity Management

Chargeback

Fabric Chargeback Reporting

Capacity name Date range Top N Item kind Operation name Billing type Virtualized Item

15 Capacities 36 Workspaces 220 Items

Reset filters Export Data Help & feedback

Filters

Utilization (CU) by date

Utilization (CU)

0.02bn 0.67bn 1.57bn

Apr 07, 12AM Apr 07, 6AM Apr 07, 12PM Apr 07, 6PM Apr 08, 12AM Apr 08, 6AM Apr 08, 12PM Apr 08, 6PM Apr 09, 12AM

Date

Select a workspace or item to be able to explore more details. Explore

Utilization (CU) details

Month Year	Apr 2024				Total			
Capacity Workspace Item Item kind Billing type	Utilization (CU)	Utilization (CU) %	Users	Users %	Utilization (CU)	Utilization (CU) %	Users	Users %
Trial-20240408T093736Z-rrv4bmWgVUqk72T--yUQKQ	1,051,845,360.40	46.36%	3	42.8571%	5,921,080,820.89	66.56%	4	36.3636%
Trial-20240407T050816Z-1fk8F0-ffES_u3cA1k1lxg	1,166,800,926.77	51.42%	5	71.4286%	2,798,298,288.07	31.45%	6	54.5455%
hagai					59,314,081.60	0.67%	2	18.1818%
Trial-20240407T113749Z-8wOryPGt6Eul-SYKfrEXyQ	26,347,500.00	1.16%	1	14.2857%	44,835,000.00	0.50%	1	9.0909%
Trial-20240407T114250Z-UvV8j6W4mUe0IJGeWkYbmA	20,527,688.20	0.90%	2	28.5714%	39,582,106.00	0.44%	2	18.1818%
Total	2,268,982,439.97	100.00%	7	100.0000%	8,896,472,054.29	100.00%	11	100.0000%

Select a date dimension to change view: Month Week Day

Microsoft Confidential Shared Under NDA.

Capacity Platform Observability improvements

Admin Monitoring Integration with deprecation of template app shipping in App store

Federated Platform Telemetry Data Access

Real-Time Intelligence

Subscribe to high fidelity capacity metrics detailed usage data (summary, operation details and capacity state changes)

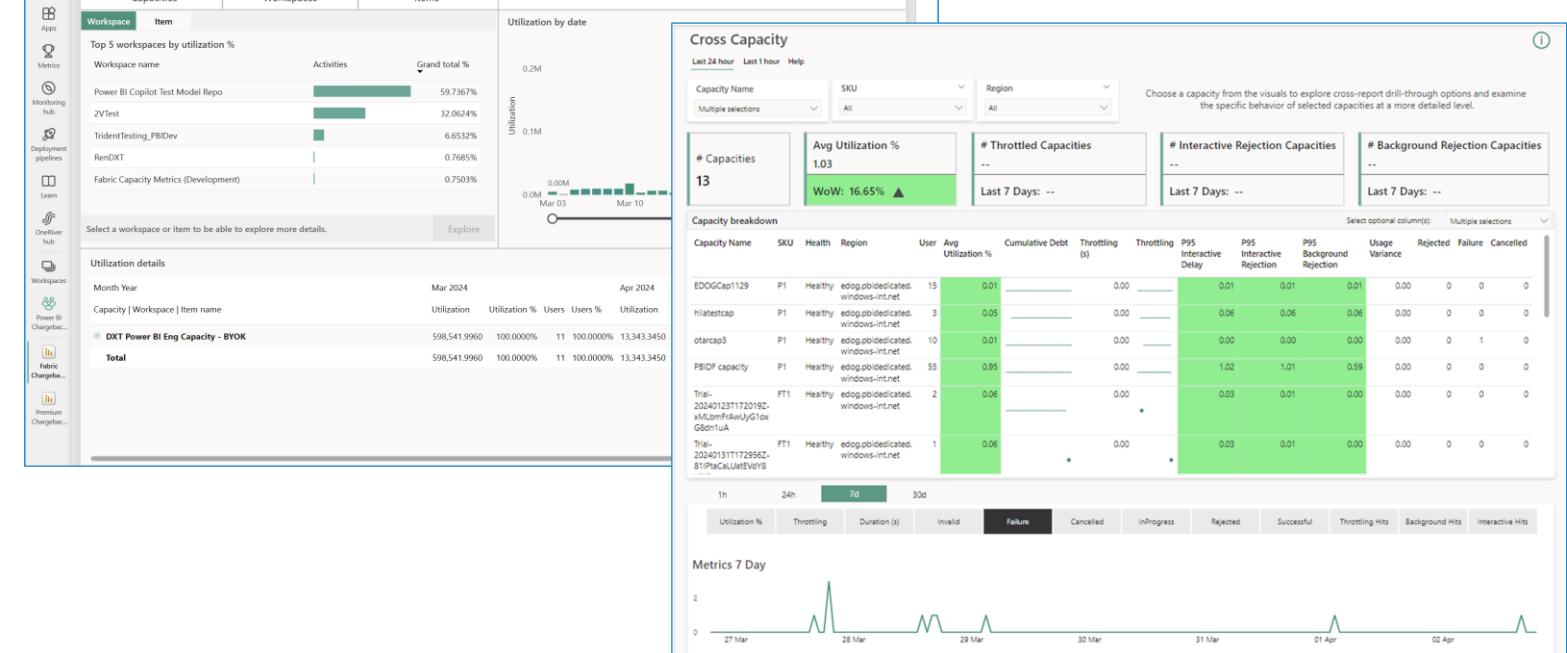


Operational and Historical Analysis

Aggregated data in Admin Monitoring:

Query aggregated historical data from a Lakehouse or OneLake for chargeback / capacity planning or forecasting with up to 2 years data retention out of box. North start for OneLake integration.

The screenshot shows the Microsoft DXT Fabric Chargeback Reporting interface. It displays a summary table with 1 Capacity, 7 Workspaces, and 120 Items. Below this, there's a chart titled 'Utilization by date' showing utilization levels over time. Another section shows 'Top 5 workspaces by utilization %' with Power BI Copilot Test Model Repo at 59.7367% and others like 2VTest, TridentTesting_PBIDev, RemDXT, and Fabric Capacity Metrics (Development) following. A sidebar on the left lists various DXT services like Home, Create, Browser, Apps, Metrics, Monitoring Hub, Deployment pipelines, Learn, OneRiver Hub, Workspaces, Power BI Chargeback, Fabric Chargeback, and Premium Chargeback.



Capacity Metrics Improvements

Tenant Capacity Health dashboard gives you a single pane of glass to monitor your capacities.

Optimized for large customers / ISVs with domain integration.

Plan for scaleup based on preview usage.

Historical Analysis

Provides lineage and historical analyses like trends, regression, success rates, and scheduling abuse.

Chargeback GA

Analytics to help Admins and ISV's distribute a Fabric bill based on resource consumption via workspace / domain or workload

Announcing



Fabric AI Capacities

Dedicate capacity for
Fabric AI workloads

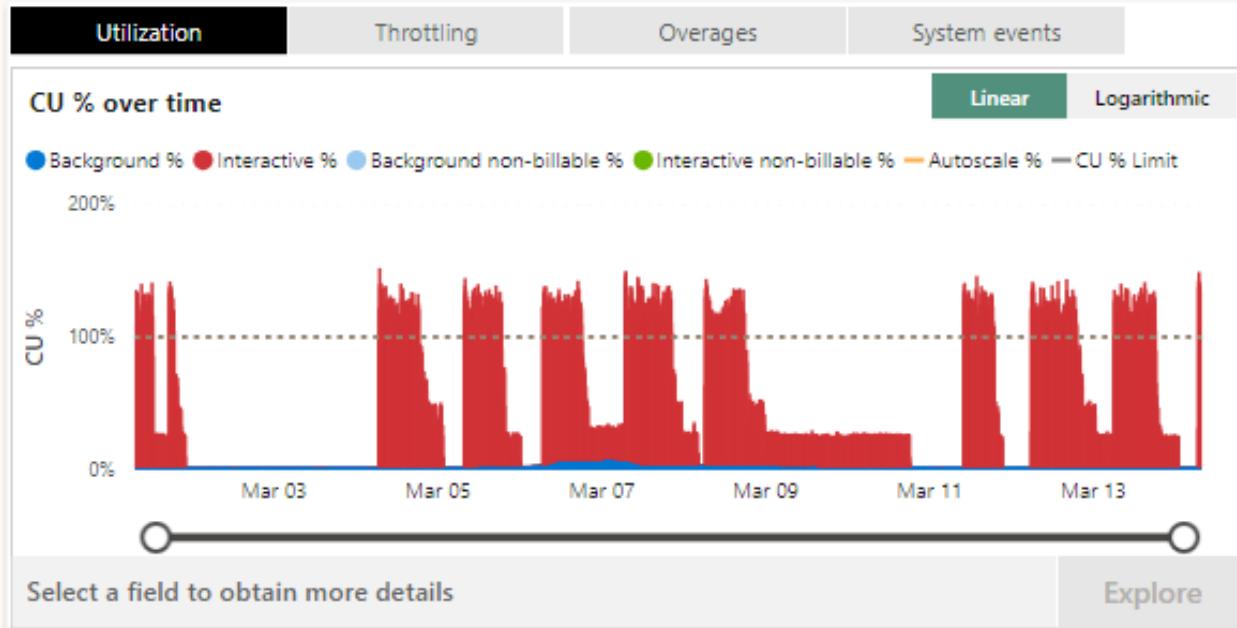
Enable access to
Copilot across Fabric
experiences

Improve capacity
management &
monitoring for AI

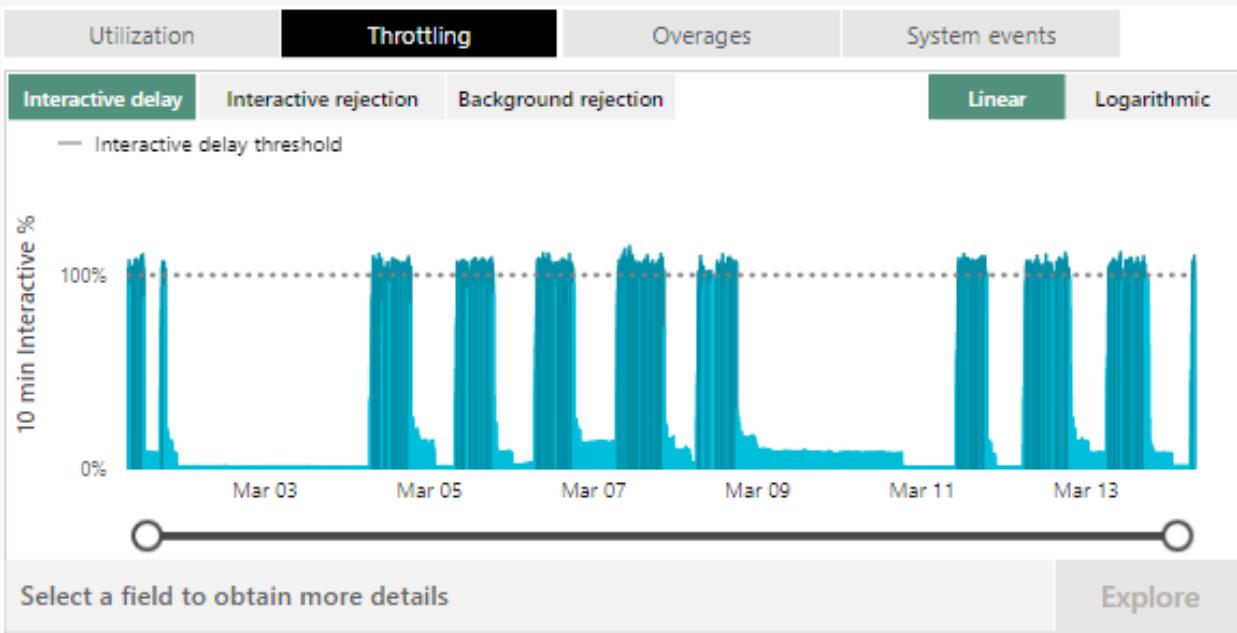


Bonus: Tips and Tricks
for capacity
management and
monitoring

My capacity is being throttled! What can I do?



Over 100% utilization doesn't always result in throttling



No penalty until you hit 100% on one of the throttling tabs

Note: For F SKU, if throttled, you can pause/resume to pay now and clear the carry forward, but that is not a long-term solution

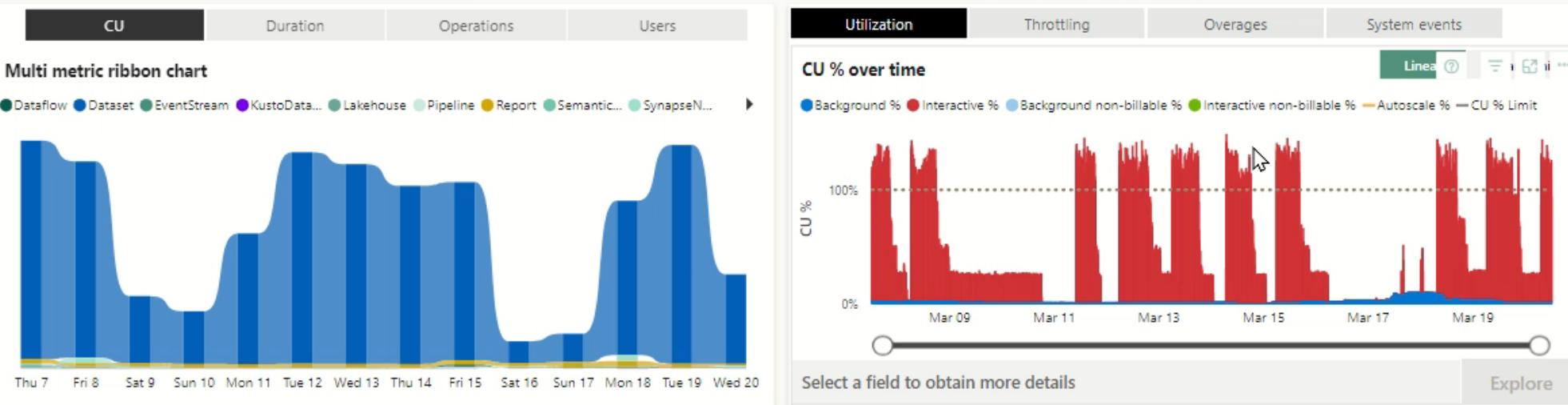
Metrics App Tips

Fabric Capacity Metrics

Compute Storage Help

Capacity name: CAT_Premium_Europe

Pick a capacity from the Capacity name slicer to see data. All visuals on the page will refresh each time a capacity is picked. Learn how to use this page by clicking the "info" button.



Select item kind(s): All Select optional column(s): Rejected count

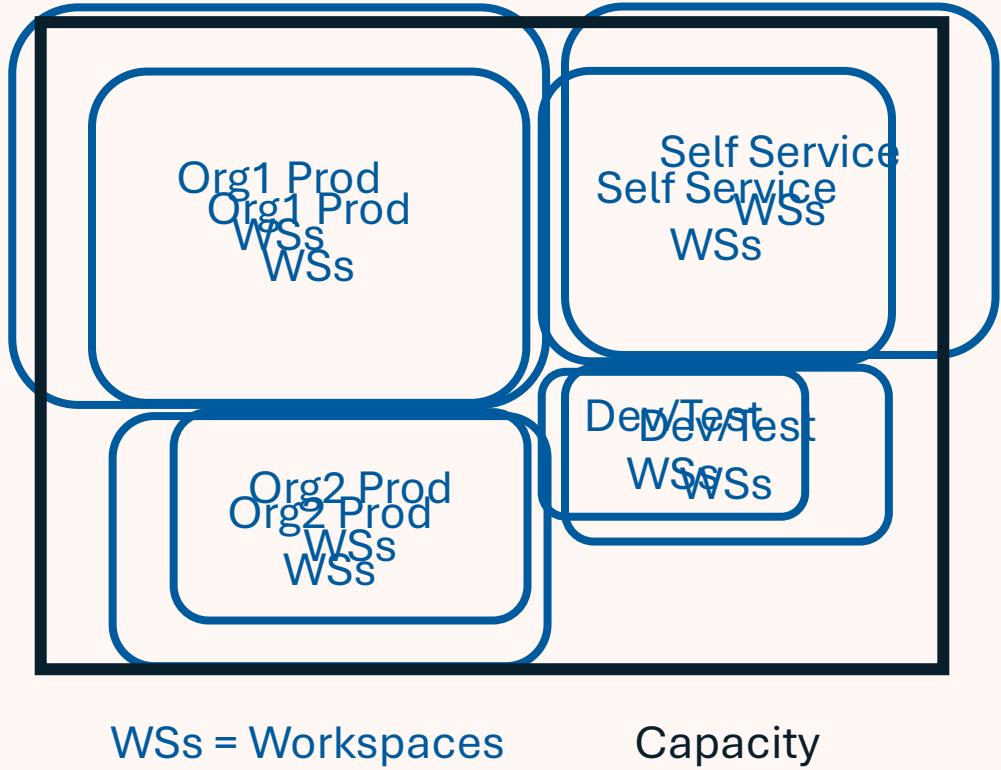
Items (14 days)

Workspace	Item kind	Item name	CU (s)	Duration (s)	Users	Rejected count	Billing type
LoadTest	Dataset	Phils test dataset_YR_Max	32,451,086.7200	2,302,896.3680	2	0	Billable
CSA PT	Report	Life expectancy	789,600.0000	14,755.2200	1	0	Billable
BDJ_NYCCitibike_StarSchemaAllT...	Dataset	00_NYCCitibike_FLAT	322,122.5920	7,810.0520	2	0	Billable
GabiDataCommunityAustria	Dataset	wwlakehouse	253,104.3680	12,047.0100	2	0	Billable
BDJ_NYCCitibike_StarSchemaAllT...	Dataset	00_NYCCitibike_STAR	133,060.0640	26,935.4120	2	0	Billable
# ankit copilot demo	SynapseNotebook	Notebook 1	127,017.4050	12,927.6500	2	0	Billable
# ankit copilot demo	Report	Power BI Session Service	93,539.2000	5,580.0000	1	0	Billable
GabiDataCommunityAustria	Lakehouse	wwlakehouse	73,768.2224	2,4700	1	0	Billable
BDJ_NYCCitibike_StarSchemaAllT...	Dataset	00_NYCCitibike_STAR (Full)	72,103.2960	9,025.5060	2	0	Billable
DbrownneFabricTest	SynapseNotebook	Notebook 1	41,743.9430	4,891.2610	1	0	Billable
BDJ_NYCCitibike_DL	Lakehouse	NYCCitibike	37,969.7683	2,660.9720	2	0	Billable
# ankit copilot demo	Dataflow	CopilotDemoDataflowGen2	35,391.4400	1,929.9490	2	0	Billable
PBI Monitor - PBICAT	Dataflow	PBI - Activity Monitor - Dataflow	32,553.7440	1,815.4310	1	0	Billable
BDJ_NYCCitibike_Raw	Pipeline	pino_BASE_NYCCitibike	28,800.0000	2,233.1790	1	0	Billable
Total			34,896,258.5263	6,028,289.7330	6	0	

Note: Didn't even go to Timepoint Detail page. Useful for usernames and individual operation time/CU

When Capacity Units Run Out

Option 1 – Optimize



Approach

- Work with content creators to follow best practices and reduce CU consumption

Pros

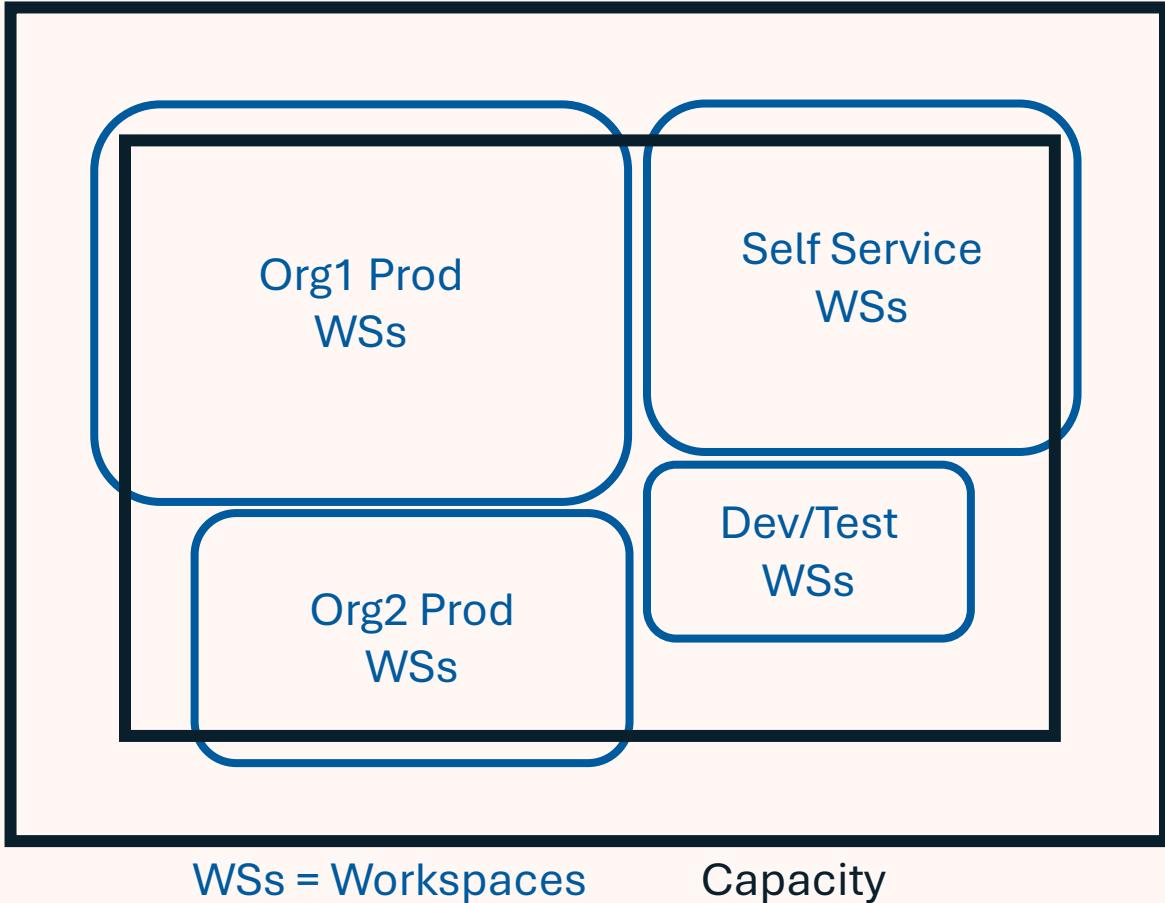
- Avoids increased cost
- Learning carries over to future content

Cons

- Can be difficult/time consuming

When Capacity Units Run Out

Option 2 – Scale Up



Options to add compute

- Move to a bigger P SKU or RI F SKU
- Turn on autoscale (P SKU)
- Manual/Dynamic change size (F SKU)

Pros

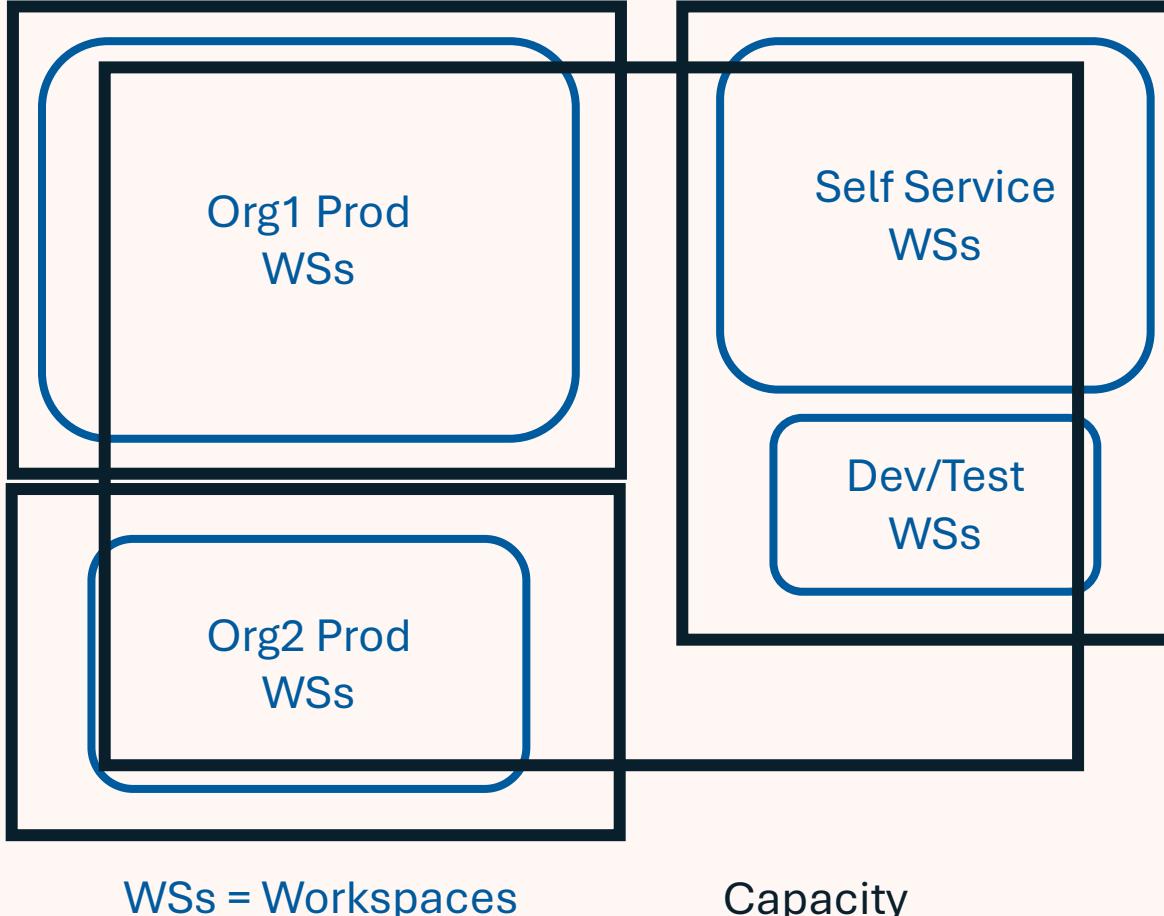
- Add CUs for all items
- Easy

Cons

- Cost
- Bad actors (items with unintentionally high CU burn) can still be a problem

When Capacity Units Run Out

Option 3 – Scale Out



Options

- Create multiple smaller P or F SKUs based on organization, type of work, etc.

Pros

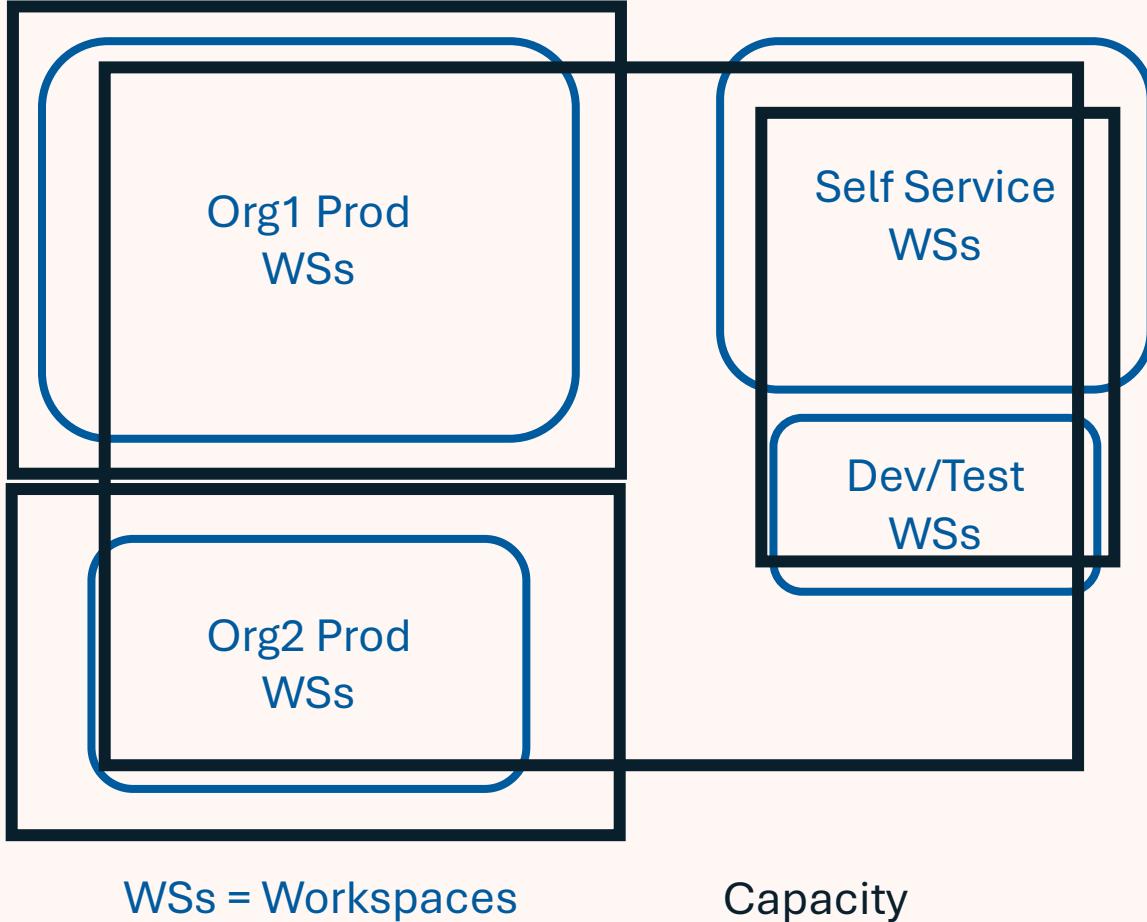
- Easy
- Provides some isolation from bad actors (items with unintentionally high CU burn)
- Flexibility in capacity settings/governance

Cons

- Cost
- High CU items have increased chance of throttling

When Capacity Units Run Out

Option 4 – Isolate



Approach

- Provide isolated capacity for key items built by experienced developers

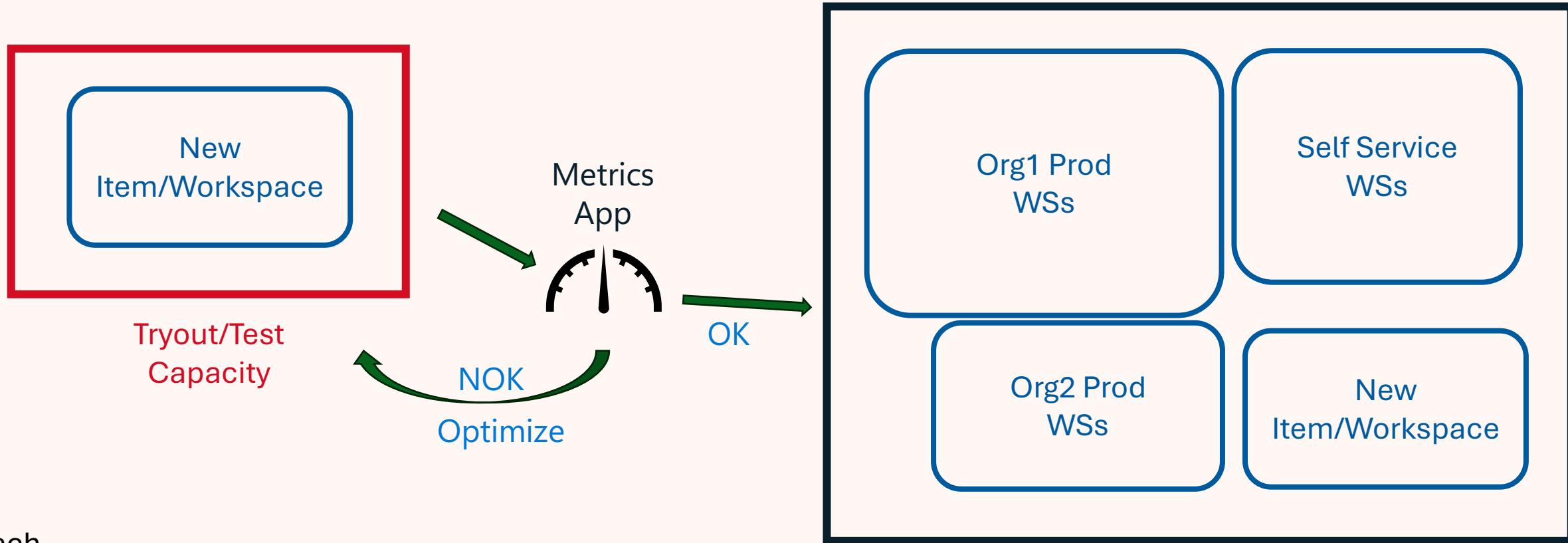
Pros

- Easy
- Provides isolation from items built by inexperienced developers and/or rapid unplanned usage growth
- Flexibility in capacity settings/governance

Cons

- Cost
- May lead to frustration of lower priority content developers/consumers

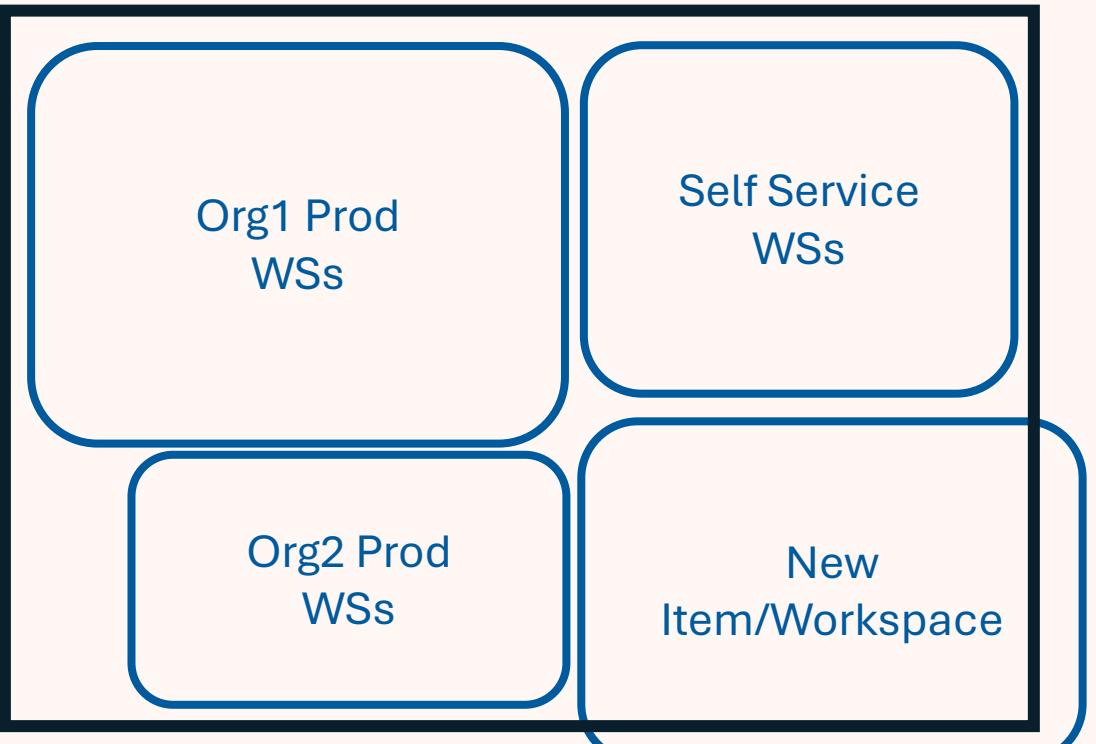
Isolation Strategy #4a – Tryout Capacity



Approach

- Create a small F SKU capacity to “tryout” new workspaces/items
- Assess CU consumption using metrics app
- If acceptable, move to prod capacity
- If not, optimize
- Pause tryout capacity when not in use, if possible
- Note size limits for semantic model size

Isolation Strategy #4b – Timeout Capacity

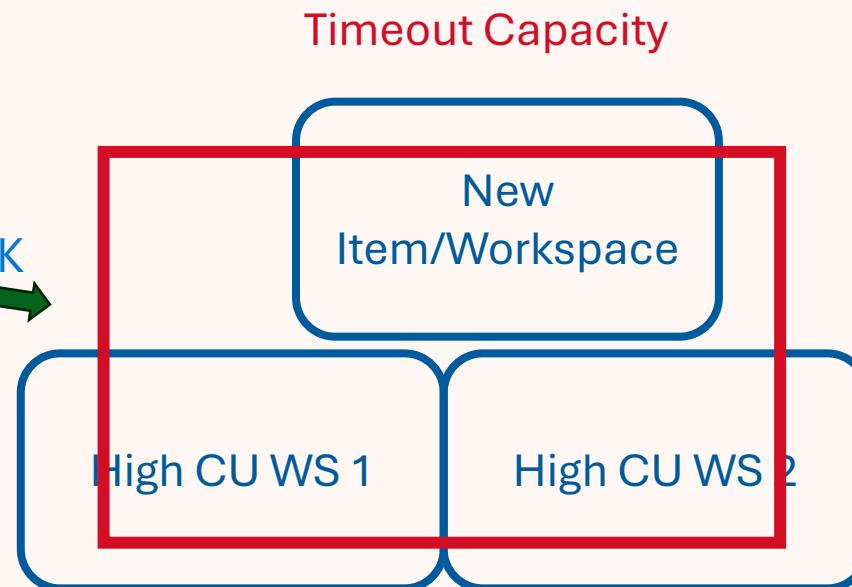


WSs = Workspaces

Prod Capacity

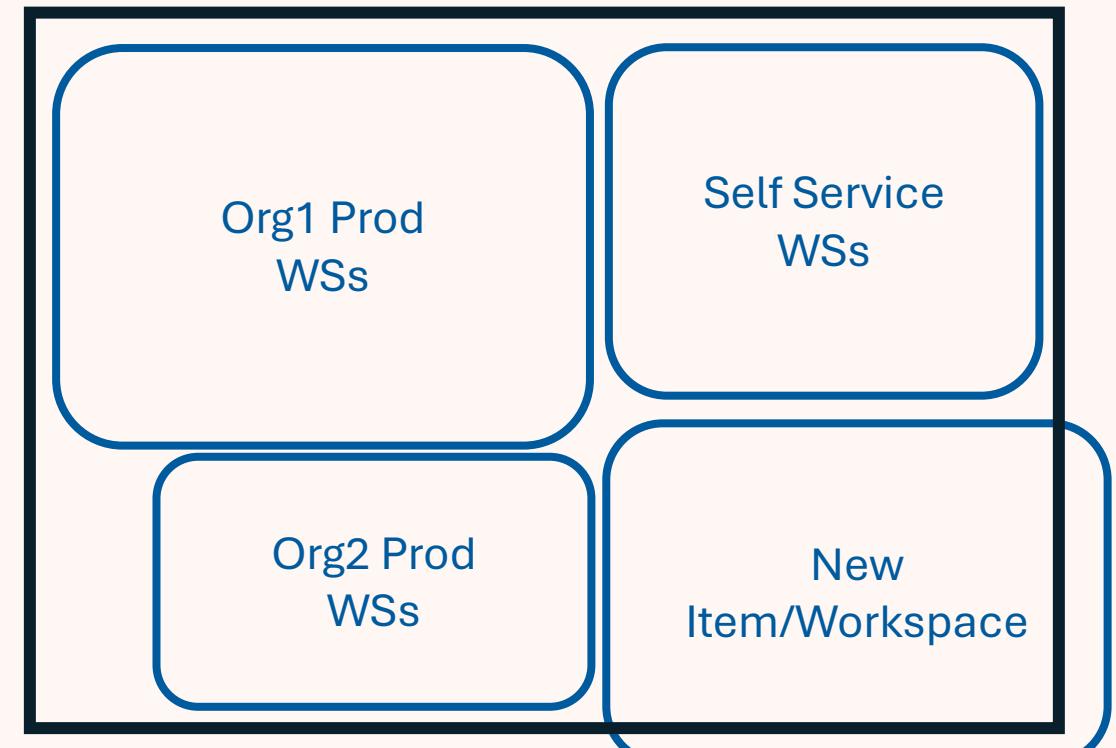
Approach

- Create a small F SKU capacity
- Assess CU consumption using metrics app
- If CU for new items/workspaces affects existing workloads (throttling), move WS to timeout capacity (Admin Portal/Capacity Settings)
- High CU items/WSs share smaller capacity (or you can pause it post move)
- Note size limits for semantic model size



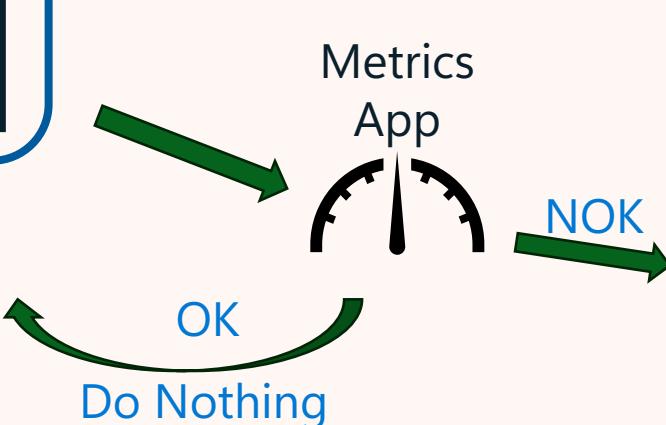
Timeout Capacity

Isolation Strategy #4c – Rescue Capacity



Approach

- Create an F SKU capacity, keep it paused
- Assess CU consumption using metrics app
- If CU for new items/workspaces affects priority workloads (throttling), resume the new capacity and move priority WS to it (Admin Portal/Capacity Settings)
- Address issues with new content, then bring it back to original capacity, and pause the new one
- Note size limits for semantic model size



Recommendations for Cost/CU Savings

- Invest in education, knowledge/best practice sharing, COE, etc. for creators and consumers (proactive optimization)
- Avoid data/report sprawl (leverage certified/promoted models, OneLake shortcuts, etc.)
- Leverage a multi-capacity strategy (isolate, tryout, timeout, etc.)
- Right size your capacities and leverage F SKUs for pause/resume/resize, or reserved instances for discounts
 - Consider a combo of RI and PAYGO (for predictable surge activity)
- Choose the right tool for the job and stay up to date on Fabric feature releases
 - High concurrency mode for notebooks

Leverage the capacity settings in the UI

- Notifications on CU overuse
- Power BI workloads settings (e.g., query limits, page refresh)

The screenshot shows the 'Capacity settings' page. On the left, a sidebar lists options: Refresh summary, Embed Codes, Organizational visuals, Azure connections, Workspaces, Custom branding, Protection metrics, Featured content, and Help + support. The main area displays a message: 'Your P1 SKU gives you access to 64 capacity units.' Below this is a large green button labeled 'Change size'. To the right of the button is a list of links:

- Disaster Recovery
- Capacity usage report
- Notifications
- Contributor permissions
Enabled for a subset of the organization
- Admin permissions
- Power BI workloads
- Preferred capacity for My workspace
- Data Engineering/Science Settings
- Workspaces assigned to this capacity

The screenshot shows two configuration panels. The top panel is titled 'Notifications' and includes the following sections:

- Send notifications when:**
 - You're using % of your available capacity
 - You've exceeded your available capacity and might experience slowdowns
 - An Autoscale v-core has been added
 - You've reached your Autoscale maximum
- Send notifications to:**
 - Capacity admins
 - These contacts:

The bottom panel is titled 'Power BI workloads' and includes the following semantic models settings:

Setting	Value
Query Memory Limit (%)	0
Query Timeout (seconds)	3600
Max Intermediate Row Count	10000
Max Result Row Count	21474
Max Offline Dataset Size (GB)	0
Automatic page refresh	On
Minimum refresh interval	5 Seconds
Change detection measure	On
Minimum execution interval	30 Seconds
XMLA Endpoint	ReadWrite

Custom Solutions

- Modify the Metrics App to meet your needs
- Build a custom report off the semantic model
- Send DAX queries to the metrics app semantic model in your own solution
 - Power Automate, Notebook (SemPy), PowerShell, etc.
 - Get throttling % values (Interactive Delay, Interactive Rejection, and/or Background Rejection)
 - Latest values and/or trends over time
 - Best for summarized data only (e.g., hour, day)

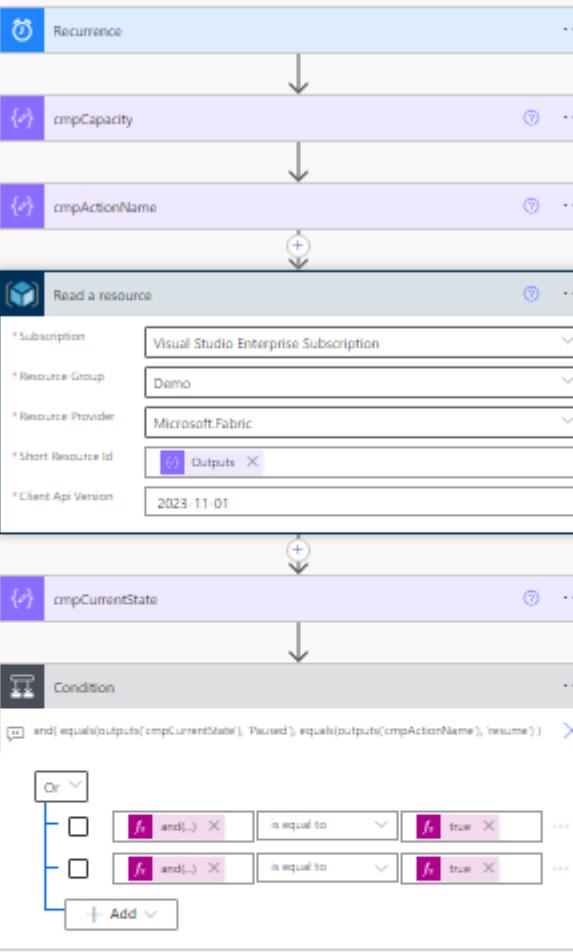
Incorporate Metrics App queries into custom solutions

The screenshot shows two cards in the Metrics App:

- DAX Query**: A card with a purple header containing a DAX query editor. It includes fields for workspace, dataset, and query text, along with an Outputs section and advanced options.
- Query**: A card with a yellow header containing a query editor. It also includes fields for workspace, dataset, and query text, along with an Outputs section and advanced options.

Collect data from multiple capacities and store it long term

```
8 # Get max date from current delta table (to avoid loading duplicate days)
9 try:
10     df_max = spark.sql(f"""
11         SELECT MAX(Date) as MaxDate
12         FROM throttling;
13     """
14     )
15     maxdate = df_max.first()['MaxDate']
16 except:
17     maxdate = datetime.today() + timedelta(days=-6)
18 maxdateforDAX = maxdate.strftime('%Y,%m,%d')
19
20 if maxdate.date() < (datetime.today() + timedelta(days=-1)).date():
21
22     # Get data for each capacity, write daily csv and append delta
23     for capacity in lst_capacities:
24         querytext = """
25             DEFINE
26             MPARAMETER 'CapacityID' = "{capID}"
27             VAR yesterday =
28                 FILTER(ALL('Dates'[Date]), 'Dates'[Date] < TODAY() && 'Dates'[Date] > DATE({MD}) )
29
30             EVALUATE
31             SUMMARIZECOLUMNS(
32                 'Dates'[Date],
33                 'TimePoints'[Start of Hour],
34                 yesterday,
35                 "IntDelay", ROUND( 'All Measures'[Dynamic InteractiveDelay %] * 100, 2 ),
36                 "IntReject", ROUND( 'All Measures'[Dynamic InteractiveRejection %] * 100, 2 ),
37                 "BackReject", ROUND( 'All Measures'[Dynamic BackgroundRejection %] * 100, 2 )
38             )
39             """.format(capID=capacity, MD=maxdateforDAX)
40     df_throttling = fabric.evaluate_dax(workspace=MetricsWS, dataset=MetricsModel, dax_string=querytext)
41     if len(df_throttling) >= 1:
42         df_throttling.columns = df_throttling.columns.str.replace(r'(\.*\[|\]\.)*', '', regex=True)
43         df_throttling.columns = df_throttling.columns.str.replace(' ', '_')
44         df_throttling['capacityId'] = capacity
45         filename = capacity + '_throttling_' + (datetime.today()).strftime('%Y%m%d') + '.csv'
46         df_throttling.to_csv("/lakehouse/default/Files/ThrottlingData/" + filename)
47         spk_throttle = spark.createDataFrame(df_throttling)
48         spk_throttle.write.mode("append").format("delta").option("overwriteSchema", "true").saveAsTable('Throttling')
```



Pause/Resume on
a Schedule

Automate With F SKUs

- Pause/resume on a schedule
 - Automate with Power Automate, Logic Apps, or a Notebook
- Resize at peak/slow times
 - Mix with Reserved Instance (PAYGO when at increased size)
 - Query the metrics app and respond to actual demand (DIY autoscale)

DIY Autoscale – Fabric Notebook (Bret Myers)

Query metrics app model

Set SKU Ranges and Values

```
1 # Parameters to be passed in from pipeline.
2 minSku = 'F2' # min sku size we can scale down to
3 maxSku = 'F128' # max sku size we can scale up to
4 utilizationTolerance = 90 # Percentage of CU used to st
5 capacityName = 'fabricbamdemo' #capacity name to be mon
6 subscriptionId = 'REDACTED'
7 metricsAppWorkspaceName = 'WS_FabricCapacityMetrics' #
8 metricsAppModelName = 'Fabric Capacity Metrics' # name o
9 alertEmail = '' # email address to send alert that we s
```

Get credentials

```
1 tenantId = mssparkutils.credentials.getSecret('keyVaultEndpoint', 'secretName_tenantId')
2 clientId = mssparkutils.credentials.getSecret('keyVaultEndpoint', 'secretName_clientId')
3 secret = mssparkutils.credentials.getSecret('keyVaultEndpoint', 'secretName_clientSecret')
4
5 api_pbi = 'https://analysis.windows.net/powerbi/api/.default'
6 api_azuremgmt = 'https://management.core.windows.net/.default'
```

Not all code shown

FabricTools/CapacityAutoScale at main · bretamyers/FabricTools · GitHub

```
1 from azure.identity import ClientSecretCredential
2 import requests, json, math
3 from pyspark.sql.functions import explode
4
5 auth = ClientSecretCredential(tenant_id=tenantId, client_id=clientId, client_secret=secret)
6 access_token = auth.get_token(api_pbi)
7
8 header = {'Authorization': f'Bearer {access_token.token}', 'Content-type': 'application/json'}
9
10 body = {
11     "queries": [
12         {
13             "query": f"""
14             DEFINE
15                 MPARAMETER 'CapacityID' = "{capacityId}"
16
17                 VAR __DS0FilterTable =
18                     FILTER(
19                         KEEPFILTERS(VALUES('TimePoints'[TimePoint])),
20                         'TimePoints'[TimePoint] >= NOW() - 1
21                     )
22
23                 VAR __DS0FilterTable2 = TREATAS({{"{capacityId}"}, 'Capacities'[capacityId])
24
25                 VAR __DS0Core =
26                     SELECTCOLUMNS(
27                         KEEPFILTERS(
28                             FILTER(
29                                 KEEPFILTERS(
30                                     SUMMARIZECOLUMNS(
31                                         'Capacities'[capacityId],
32                                         'Time'[BillableTime]
33                                     )
34                                 )
35                             )
36                         )
37                     )
38
39             */
40             SELECTCOLUMNS(
41                 __DS0Core,
42                 'Capacities'[capacityId],
43                 'Time'[BillableTime]
44             )
45         }
46     ]
47 }
```

Change SKU Size

```
19 url = f'https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{res
20
21 body = {
22     "sku": {
23         "name": f'{scaleSku}',
24         "tier": "Fabric"
25     }
26 }
27
28 response = requests.patch(url, headers=header, data=json.dumps(body))
```

Most Common Capacity Issues (Power BI)

Bad Practice	Recommendations/Typical Resolution
Model issues (M:M, bi-di, snowflake, etc.) and/or inefficient DAX	Follow best practices (e.g., BPA), star schema
Too many visuals	Multi card, small multiples, Deneb, PowerPoint background, etc.
Big single visual (i.e., matrix with lots of rows, columns, and/or measures)	Improve report design (e.g., drillthrough, apply all Slicers, report page tooltip), field parameters, calc group guardrails, etc.
Complex RLS	Remodel to enable simple filter like Table[Email] = USERPRINCIPALNAME()
Very high concurrency	Optimize reports, DAX, etc. (big multiplier) Consider QSO, data subsets
Direct Query	Switch to import or Direct Lake, if possible. Aggregations, hybrid tables, etc.
Analyze in Excel	Automate downstream analytics with a Power BI report instead, subscriptions, DAX connected table, slicers/measures first, etc.
Excessive refresh	Don't "break the fold", incremental refresh, reduce frequency, optimize M code

Save Those CUs – Getting Data Into Excel

Analyze in Excel

The screenshot shows the Power BI Datasets interface. At the top, there's a search bar labeled "Search for a dataset". Below it, a list of datasets is shown, with "DemoSales" selected. The details for "DemoSales" are displayed: Workspace: PaginatedDemos, Owner: Patrick Mahoney, Refreshed: 2/22/2024, 11:45:46 AM. Under "Tables in this dataset (10)", there are two items: "Tables in this dataset (10)" and "Reports using this dataset (2)". At the bottom, there are two buttons: "+ Insert PivotTable" and "+ Insert Table". A large green arrow points from the "Connected Table" text below to the "+ Insert Table" button.

Connected Table

Key Takeaways

- How you build it matters
 - Filters & measures first!
- This shows durations but it's CU that matters (test your use cases/models)
- Opt for DAX Connected Tables
 - Create pivot table from that, if needed

✗ Rows, Measures, Filter

StartTime	Type	Duration	User	Database	Query
11:49:30	MDX	2,328ms	Power Bl...	DemoSales	SELECT {[Measures].[To]
11:49:26	MDX	0ms	Power Bl...	DemoSales	SELECT [AddCalculated
11:49:23	MDX	0ms	Power Bl...	DemoSales	SELECT [AddCalculated
11:49:17	MDX	1,875ms	Power Bl...	DemoSales	SELECT {[Measures].[To]
11:49:03	MDX	4,469ms	Power Bl...	DemoSales	SELECT {[Measures].[To]
11:48:54	MDX	3,938ms	Power Bl...	DemoSales	SELECT {[Measures].[To]

✓ Filter, measures, rows

StartTime	Type	Duration	User	Database	Query
10:06:13	MDX	1,625ms	Power Bl...	DemoSales	SELECT {[Measu
10:06:03	MDX	781ms	Power Bl...	DemoSales	SELECT {[Measu
10:05:49	MDX	109ms	Power Bl...	DemoSales	SELECT {[Measu
10:05:46	MDX	312ms	Power Bl...	DemoSales	SELECT {[Measu
10:05:43	MDX	234ms	Power Bl...	DemoSales	SELECT FROM [N
10:05:14	MDX	0ms	Power Bl...	DemoSales	SELECT [AddCal

Refresh (same for both)

StartTime	Type	Duration	User	Database	Query
11:50:30	MDX	2,234ms	Power Bl...	DemoSales	SELECT {[Measures].[To]

✗ Rows, Measure, Filter

StartTime	Type	Duration	User	Database	Query
01:28:50	DAX	31ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:41	DAX	1,516ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:40	DAX	16ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:34	DAX	156ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:33	DAX	16ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:31	DAX	0ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:30	DAX	141ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:15	DAX	2,047ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:11	DAX	1,797ms	Power Bl...	DemoSales	DEFINE VAR _C
01:28:08	DAX	594ms	Power Bl...	DemoSales	DEFINE VAR _C
01:27:56	DAX	281ms	Power Bl...	DemoSales	DEFINE VAR _C
01:27:50	DAX	16ms	Power Bl...	DemoSales	DEFINE VAR _C

✓ Filter, measures, rows

StartTime	Type	Duration	User	Database	Query
09:14:20	DAX	16ms	Power Bl...	DemoSales	DEFINE VAR _DS0filte
09:14:07	DAX	1,000ms	Power Bl...	DemoSales	DEFINE VAR _DS0filte
09:14:02	DAX	1,188ms	Power Bl...	DemoSales	DEFINE VAR _DS0filte
09:13:59	DAX	594ms	Power Bl...	DemoSales	DEFINE VAR _DS0filte
09:13:51	DAX	531ms	Power Bl...	DemoSales	DEFINE VAR _DS0filte
09:13:50	DAX	0ms	Power Bl...	DemoSales	DEFINE VAR _DS0Cor

Refresh (same for both)

StartTime	Type	Duration	User	Database	Query
11:54:49	DAX	1,969ms	Power Bl...	DemoSales	DEFINE VAR _DS0filterTable = TREATA



Session Feedback



Slides

