NAV TECH DAYS 2018

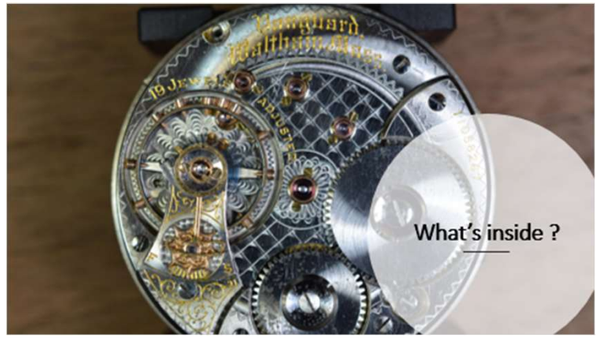
mibuso.com

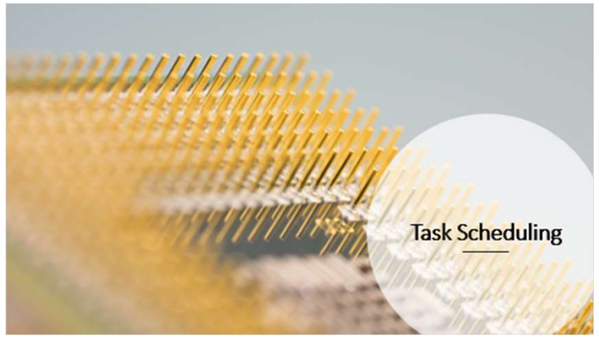
Performance: Business Central reloaded for the Cloud

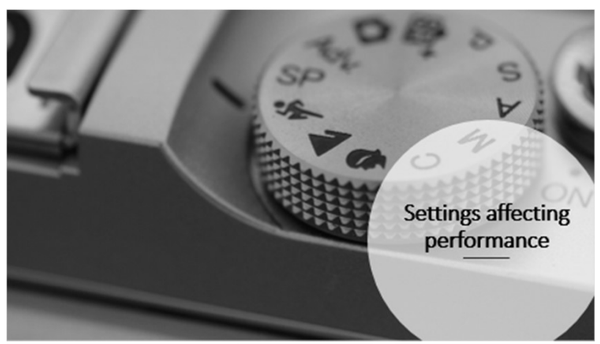
TORBEN WIND MEYHOFF, BARDUR KNUDSEN, JENS MØLLER-PEDERSEN
MICROSOFT DEVELOPMENT CENTER COPENHAGEN

Agenda

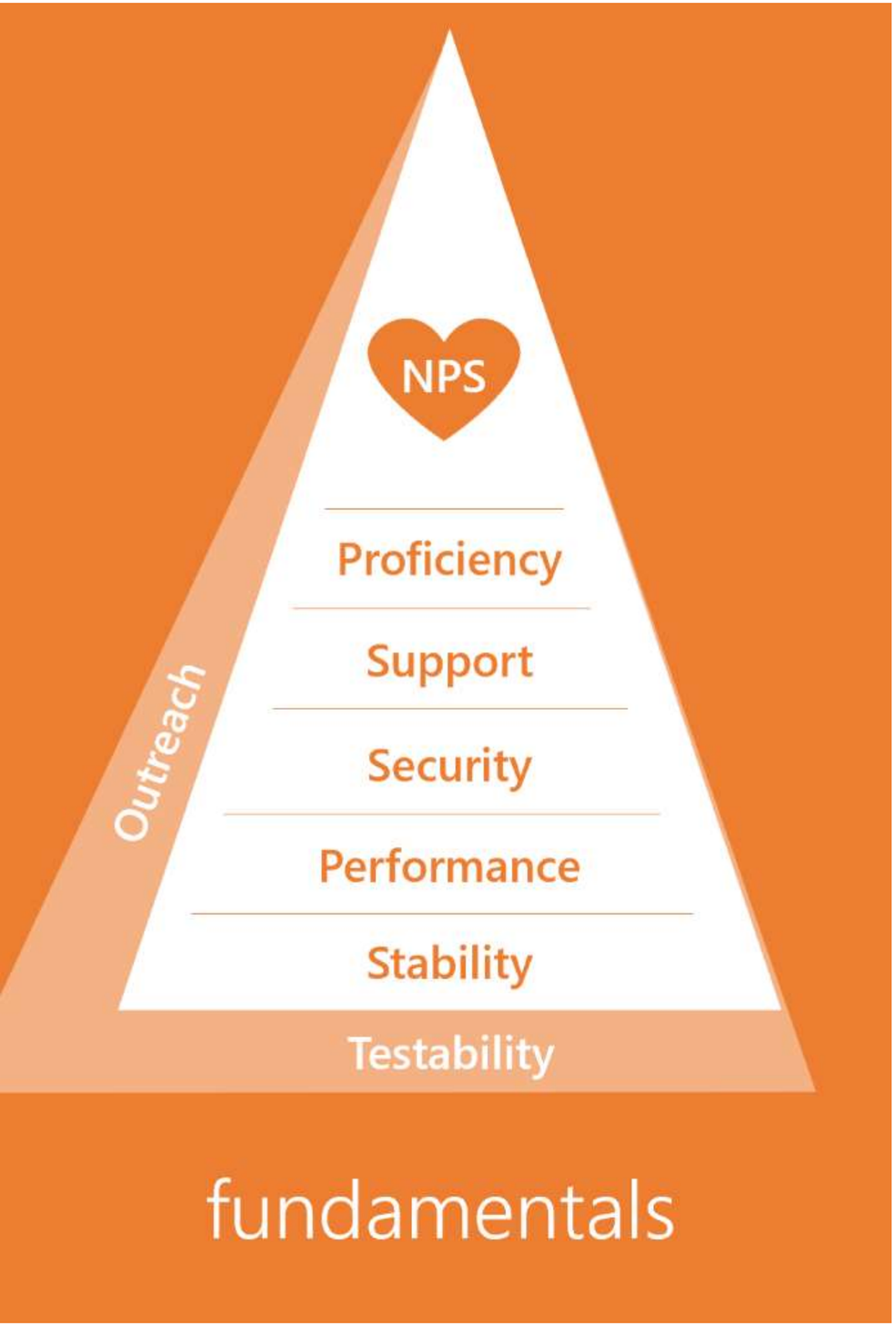
NAV TECH DAYS 2018
mibuso.com







BC Fundamentals



Focus for spring release

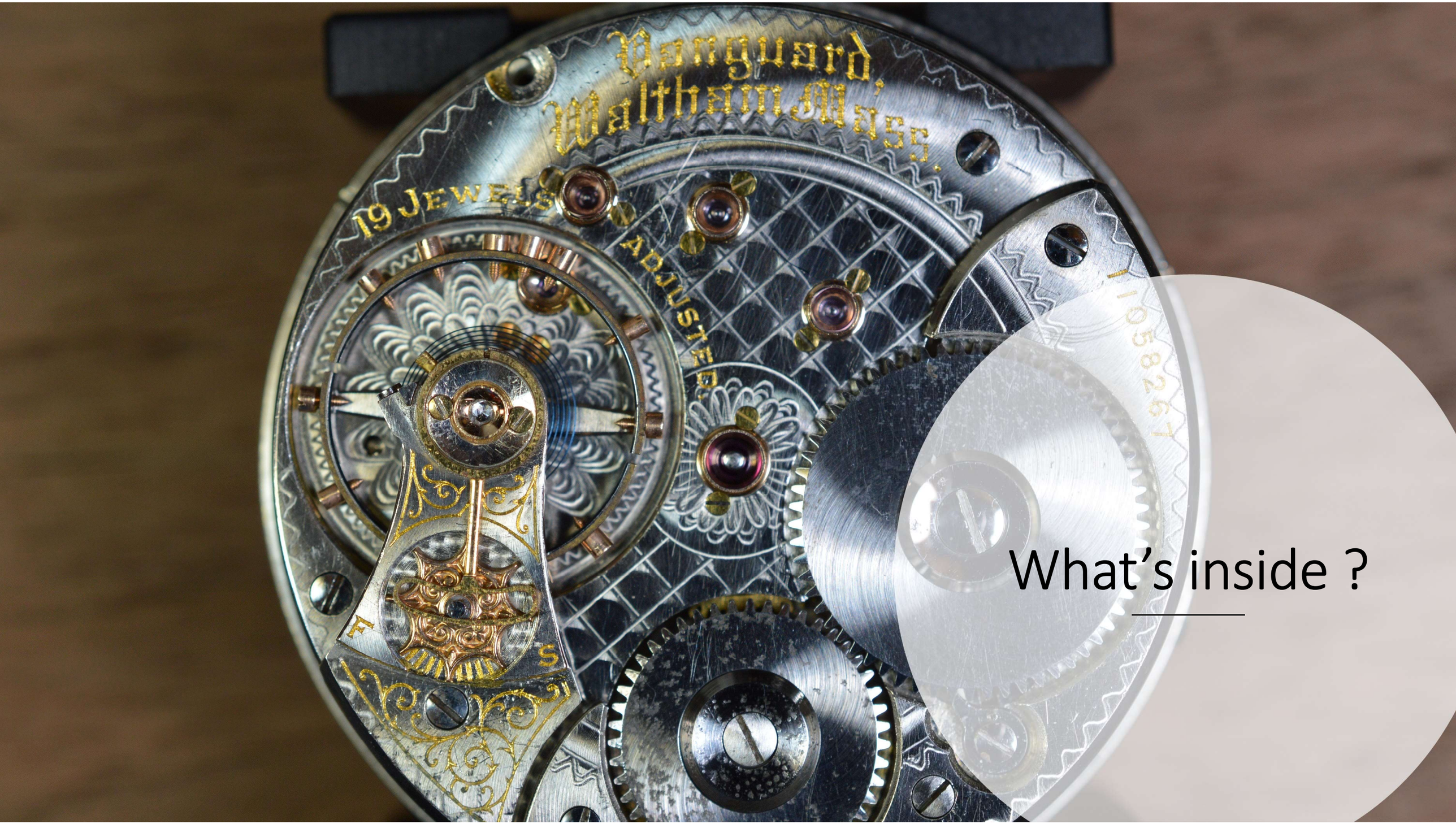
- Build on the momentum
- Drive satisfaction for
  - New customers
  - New partners
- Incremental value

Focus for Spring 19: Performance

- **Focus on the end user**

- **UX Goals**
  - Opening pages: 2 - 5 seconds
  - Entering values: 0,5 – 1 second
  - Key press: < 100 ms

- **Data sizes**
  - Think millions



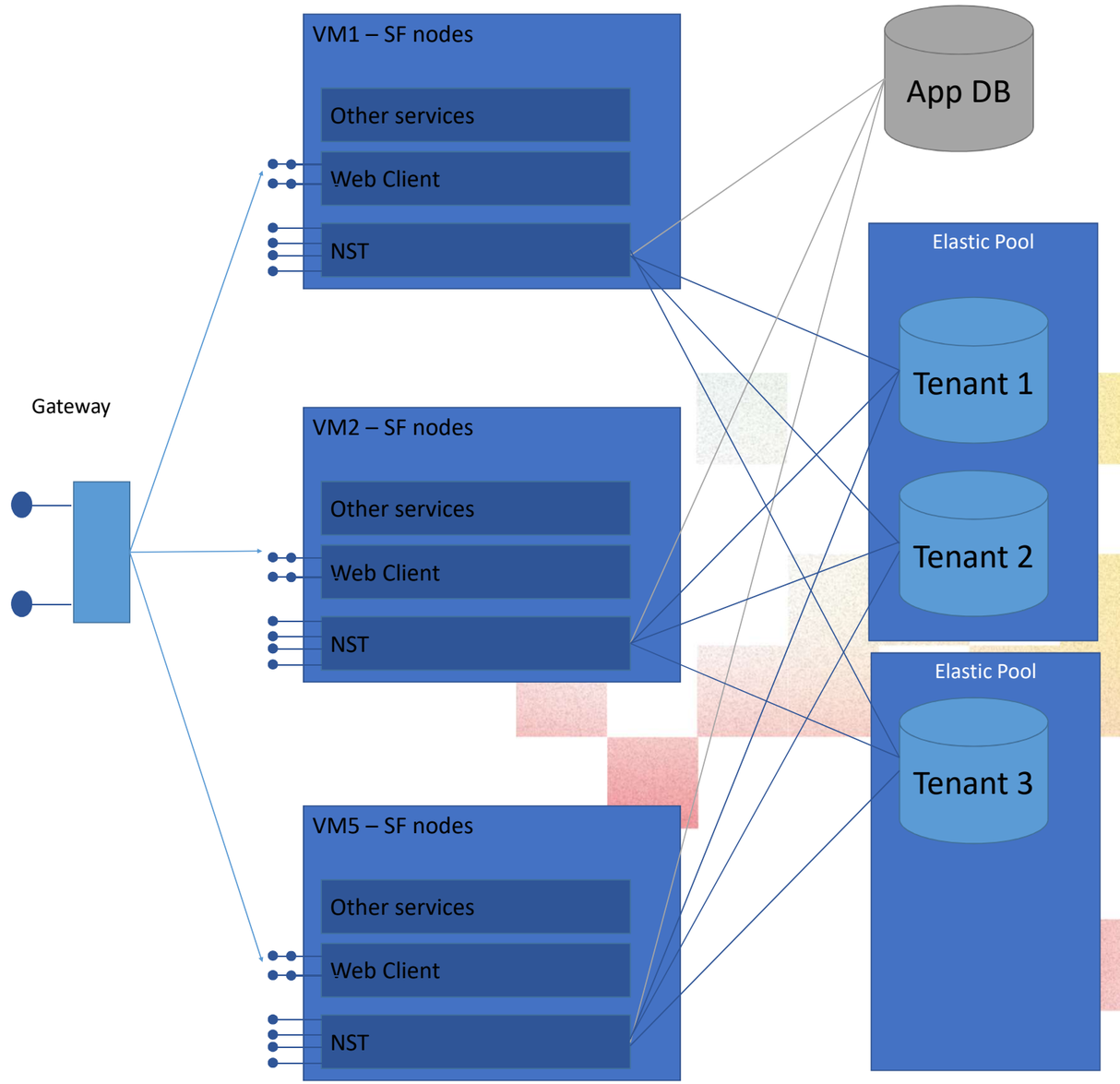
What’s inside ?

Current BC service topology

- 1+ clusters per country version
  - 5 VMs
  - 1 application database
  - 1-2000 tenants

- Service fabric orchestration

- Azure SQL databases
  - Mix of Standard and Premium
  - Elastic pool for resource sharing



Azure SQL replication

- High Availability features always enabled
- Replication to 2 secondary databases
  - Stateless compute layer
  - Stateful data layer with database files
- Performance implications
  - Every commit of data validated in 3 storage accounts
  - Latency goes from ~150µs to ~1 ms

Considerations for Azure SQL

- Writing (committing) data is an order of magnitude slower
  - Avoid frequent commits

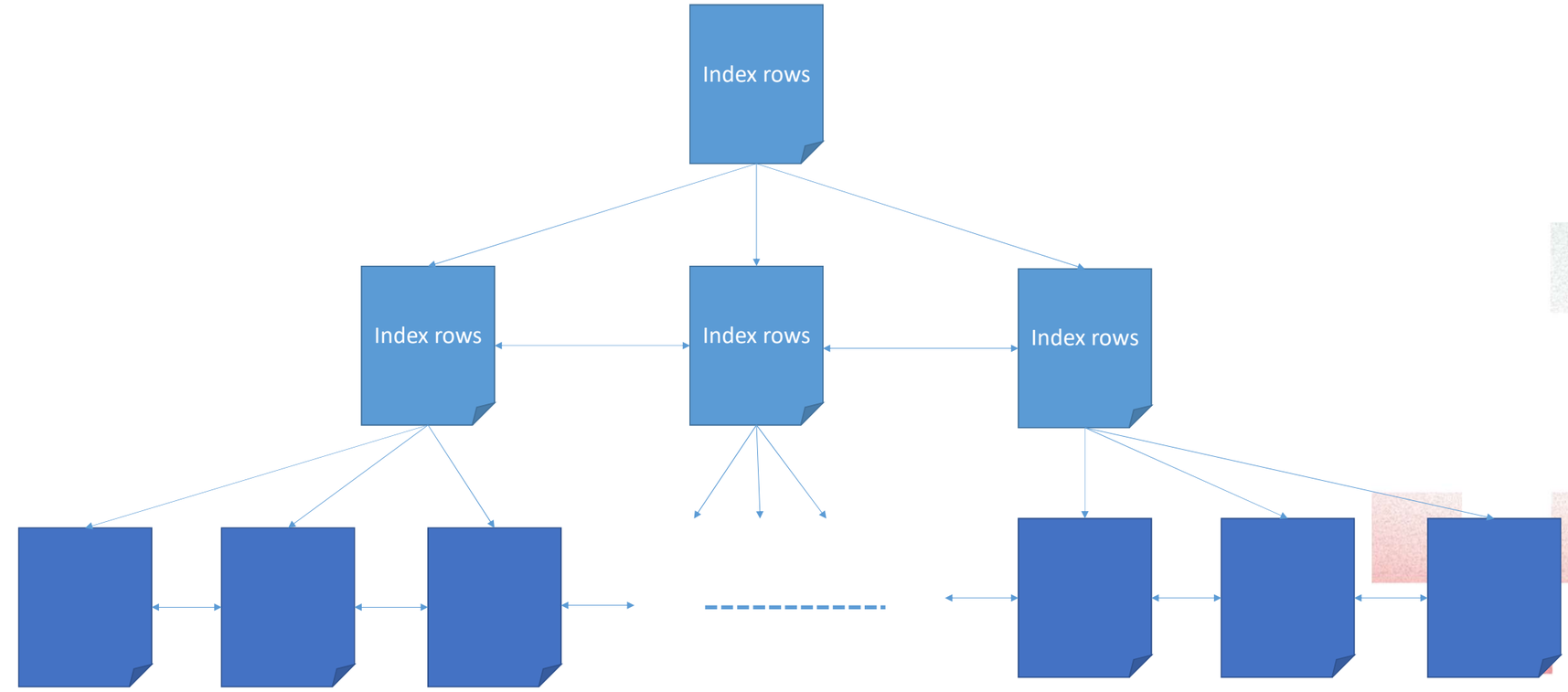
- Reading from data pages can be more expensive
  - Hitting a warm cache becomes more important
  - Limit what you need to put into the cache

- Business Central service
  - Always using the latest compatibility level
  - Directing one tenant's traffic to specific node
  - Running with index tuning advisor

Business Central Implementation details

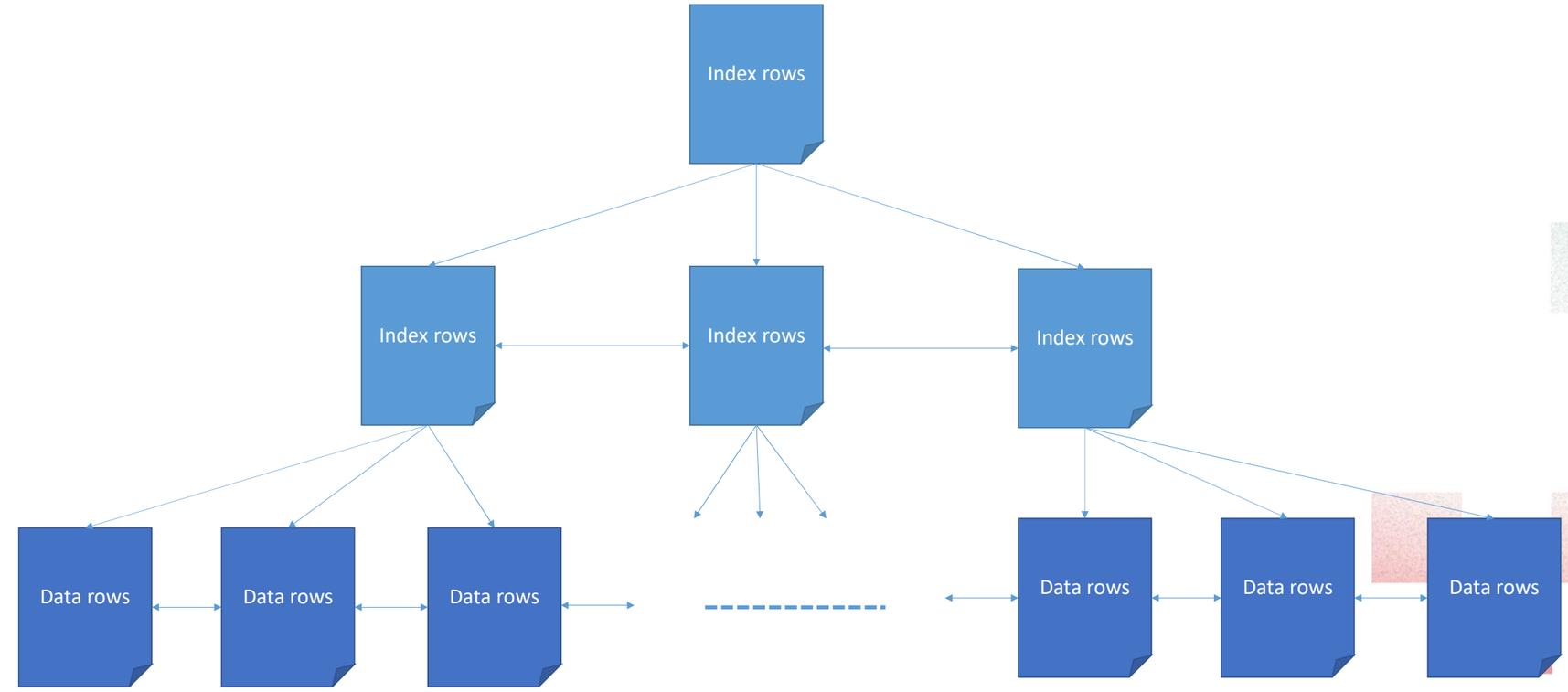
- **All indices are made unique by adding the primary key**
  - Wider index rows
- **Indexes cannot span to companion tables**
  - Consider related tables instead
- **BC records always retrieves *all columns***
  - Except BLOB, Media and Flowfields
  - Lots of data pages being fetched
- **BC runs with 'Optimize for Unknown'**
  - Favoring stable performance over peak performance
- **BC is case-sensitive currently**
  - Search box is slow – but now has a timeout

Basic index structure



Leaf nodes contains a row locator – for BC tables this is the clustered index key
Extra data fields can be added to leaf nodes – but not in Business Central

Clustered index structure

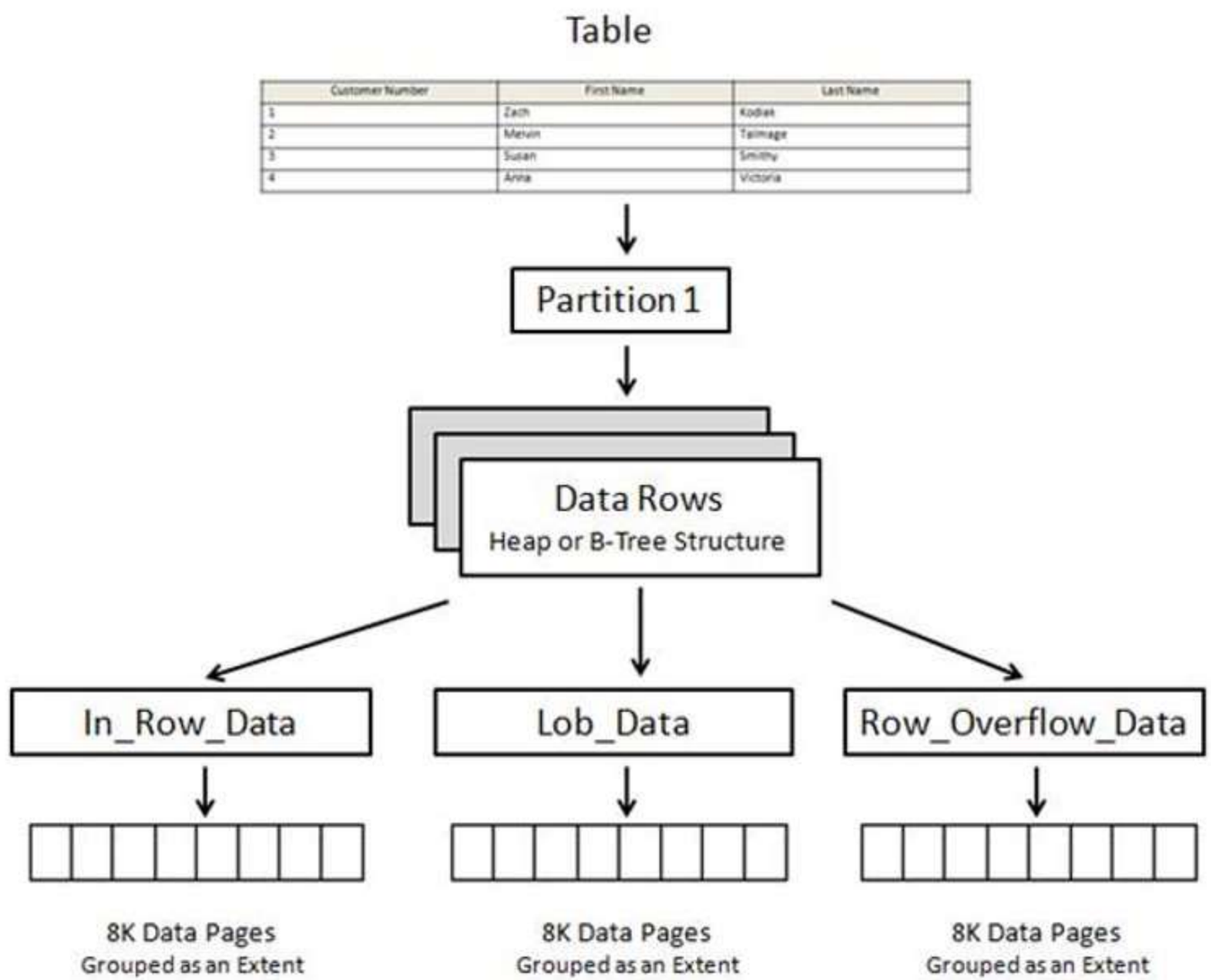


Data is ordered by clustering key into data pages
Inserting in the middle causes page splits (fragmentation)

Selecting good keys

- **Picking a good clustered index improves performance**
  - Compact
  - Naturally grouping items together
  - Usable for requests
- **Every extra index causes a performance hit on modify and insert**
  - Only add indexes for important read scenarios

Data row contents



Key takeaways

- Select highest possible database compatibility level
- Limit the width of indexes
- Select a good clustering index
- Consider using Query object for heavy calculations
- Limit amount of COMMIT statements

Task Scheduling

Parallel Tasks

Many good reasons for Parallel Tasks

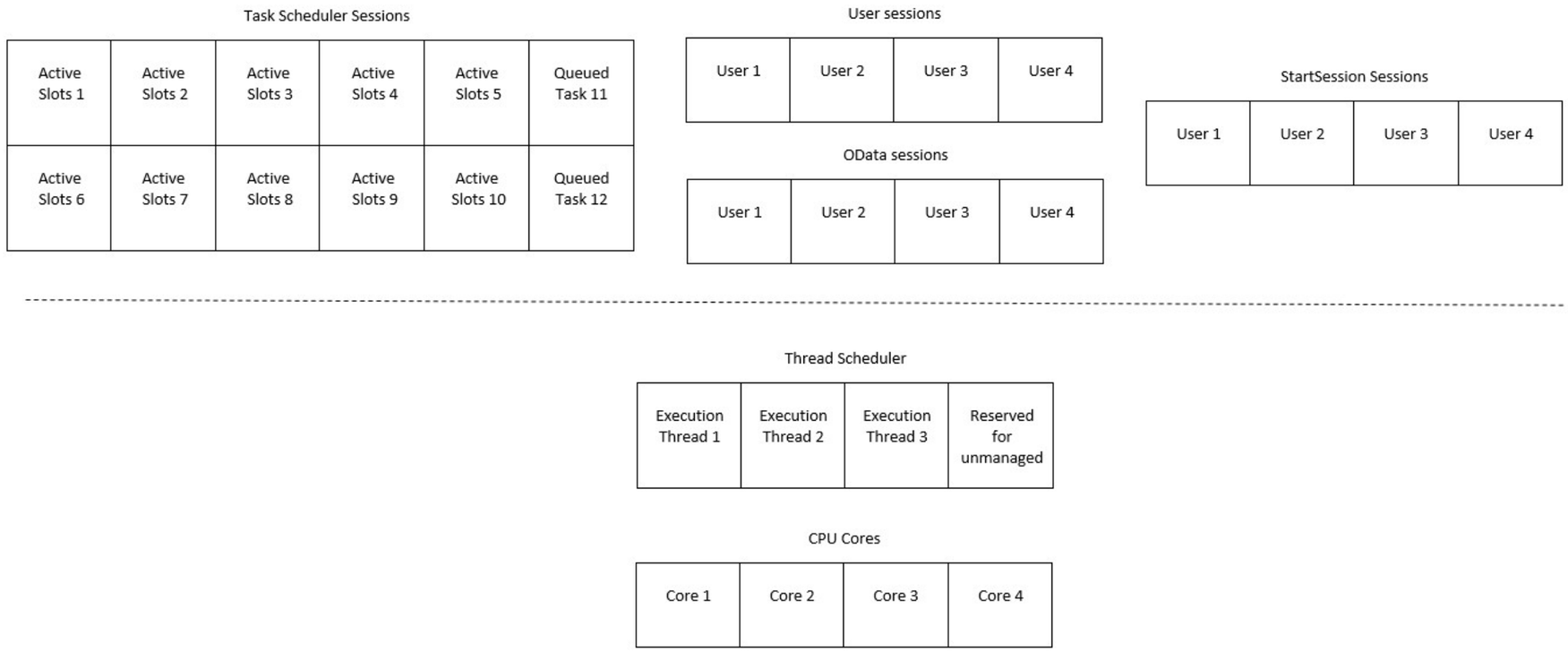
- Offload the UI thread
- Don’t let the user wait for batches
- Use the CPU cores on your box
- Splitting tasks into smaller tasks

Spinning up new Tasks is easy

- Users opening sessions
- Incoming Integration Request: OData, Soap etc.
- Job Queue
- TaskScheduler.CreateTask
- StartSession

What's the urgency of background task

Task Resource Governance



Thread Scheduler Sample

Without thread scheduling

Session	50ms	100ms	150ms	200ms	250ms	……	1050ms	1100ms	1150ms	1200ms	1250m	1300m	1350m
Client session 1 - request	Done (15)												
Heavy long running	Run (5)	Run (50)	Run (50)	Run (50)	Run (50)	……	Done (45)						
Client session 1 - request 2		Waiting	Waiting	Waiting	Waiting	……	Run (5)	Done (10)					
Odata call		Waiting	Waiting	Waiting	Waiting	……	Waiting	Run (40)	Run (50)	Run			
Client session 2			Waiting	Waiting	Waiting	……	Waiting	Waiting	Waiting	Waiting	Run	Run / Done	

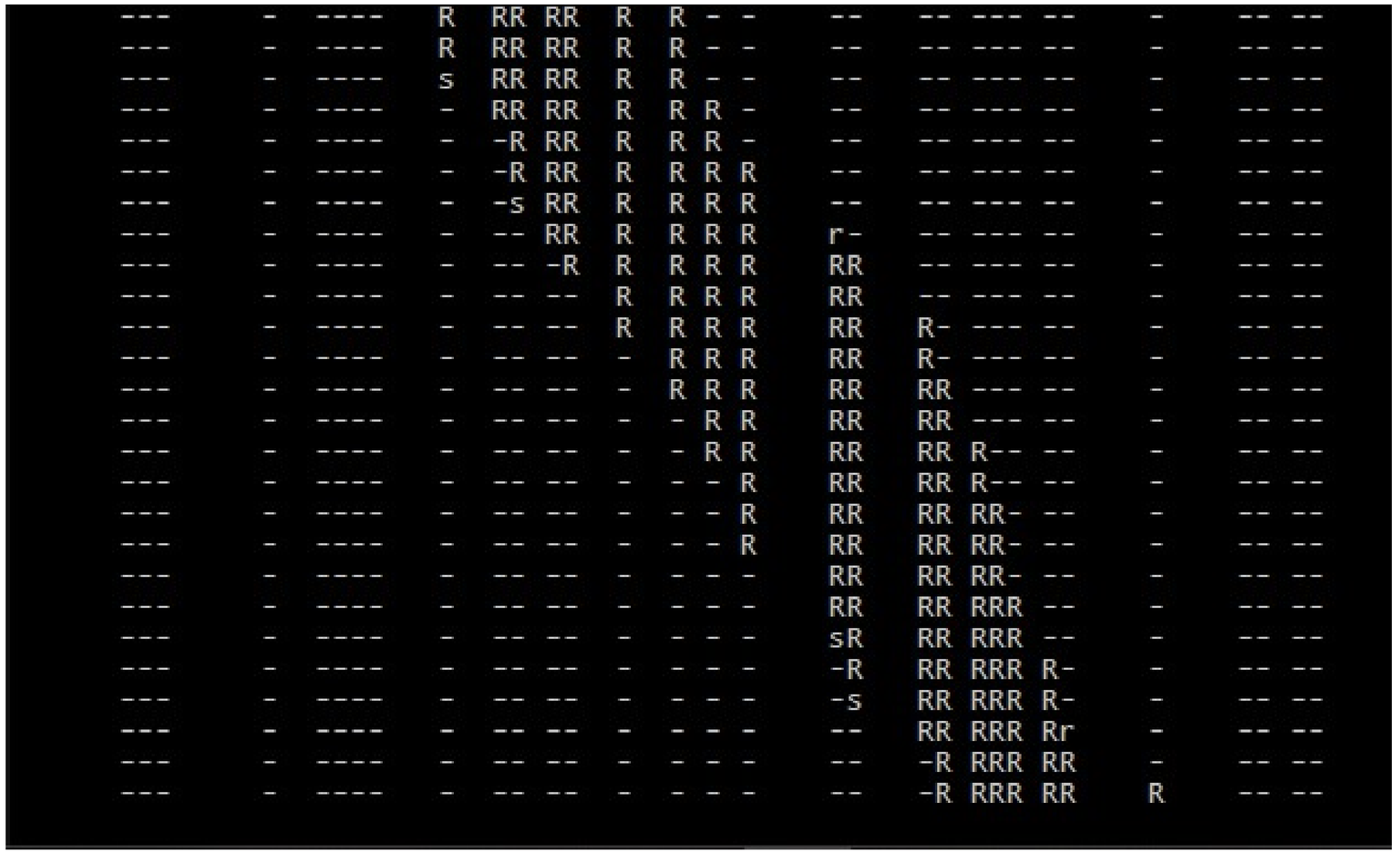
With thread scheduling - 1 task

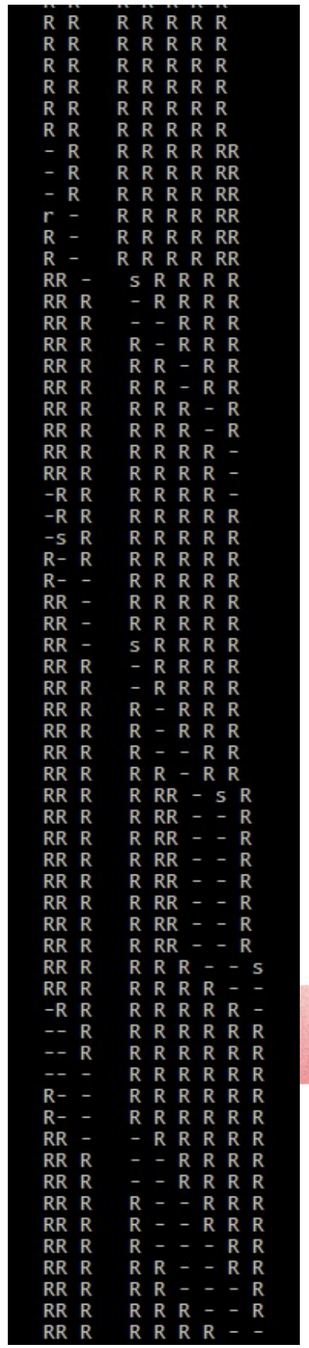
Session	50ms	100ms	150ms	200ms	250ms	300ms	350ms	400ms	450ms	500ms	550ms	600ms	…..
Client session 1 - request	Done (15)												
Heavy long running	Run (5)	Run/StopRe	Stopped	Waiting	Waiting	Run/StopRequ	Stopped	Run (25)	Run/StopRe	Stopped / R	Run	Run	Run
Client session 1 - request 2		Waiting	Run/Done										
Odata call		Waiting	Run (35)	Run / StopR	Stopped	Waiting	Run/StopRe	Stopped	Waiting	Run/Done (5)			
Client session 2			Waiting	Waiting	Run/StopRe	Stopped	Waiting	Run/Done (25)					

Thread Scheduler Sample

8 cores

Growing amount of busy users





Summary Tasks

All background tasks involve session start and CompanyOpen

Recurrence – Better batch rather than kicking off every second

Only start with the urgency you need

Sorry, we just updated this page. Reopen it, and try again.

Summary
Tasks

All background tasks involve session start and CompanyOpen

Recurrence – Better batch rather than kicking off every second

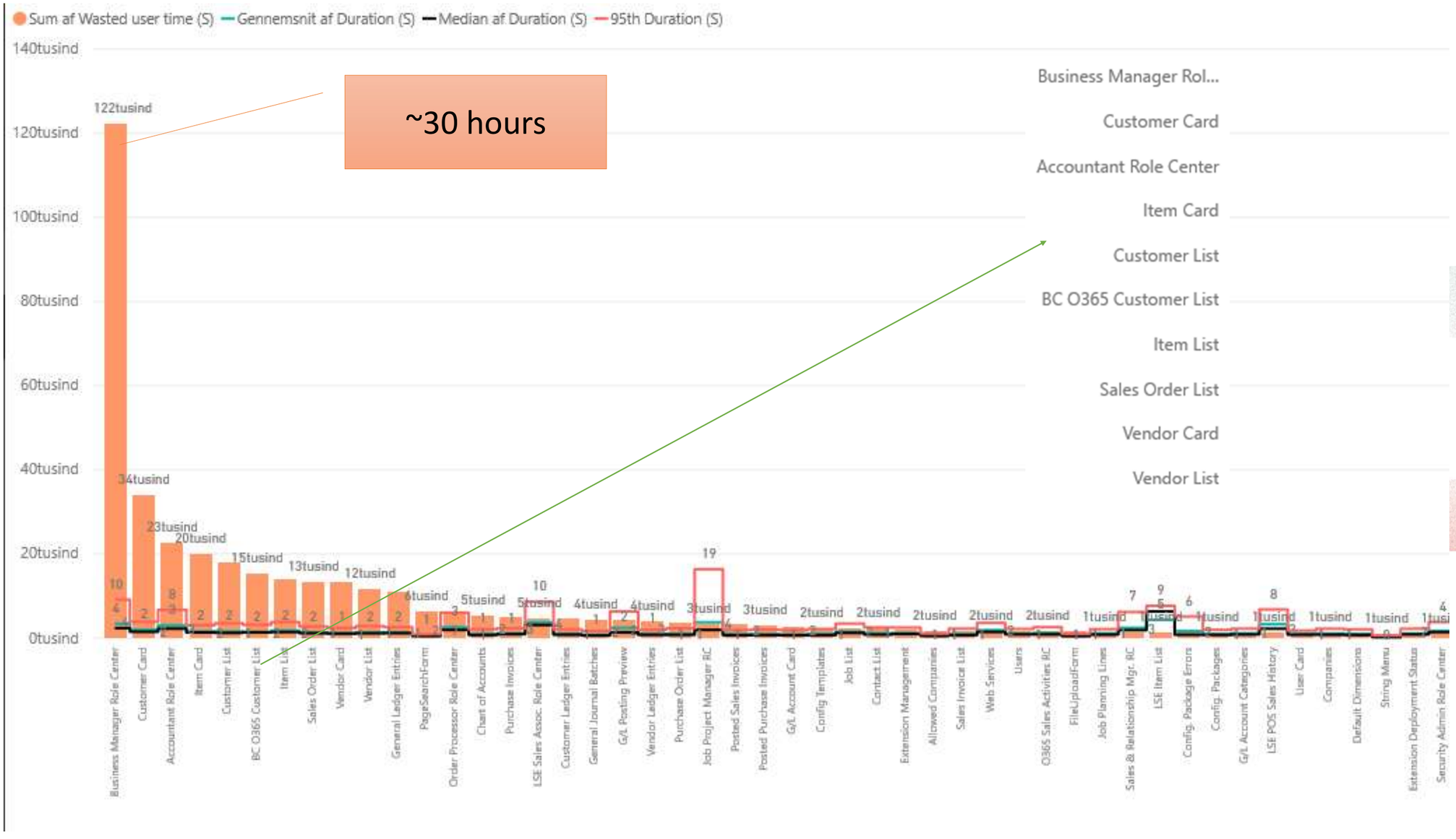
Only start with the urgency you need

Careful with record modifications with another session

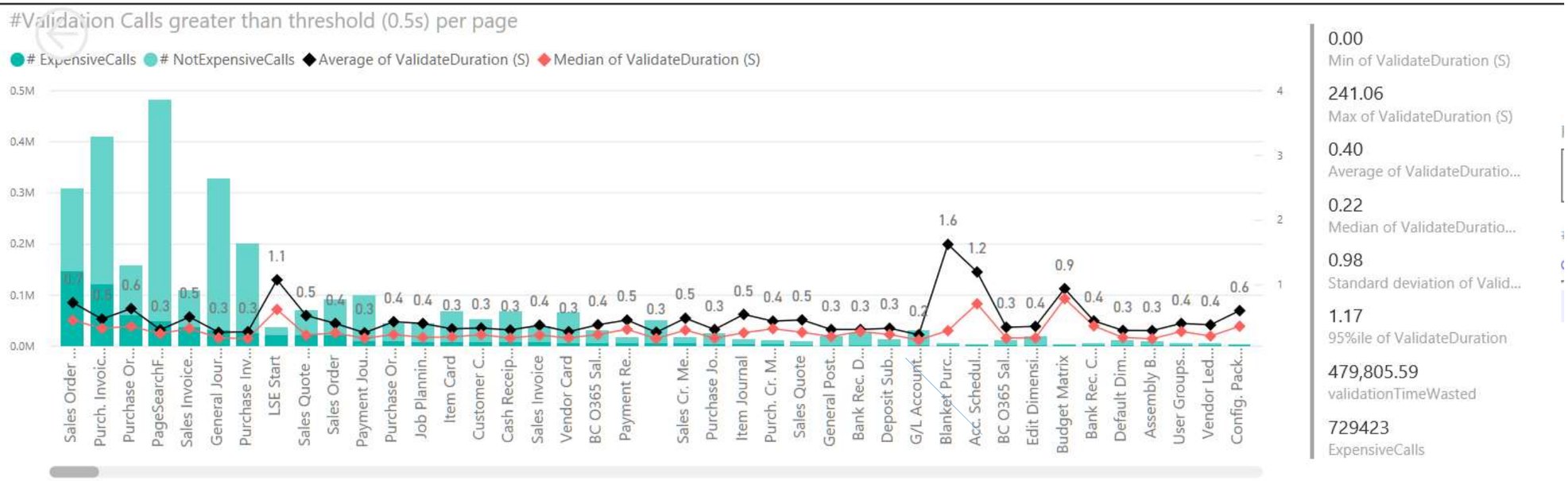

ILS APP
LDZ10 226°
10.2NM
TERR
VOR1
109.85
3.8NM
VOR2
113.70
NM
TERR ON ND
GND SPLRS ARMED
SEAT BELTS
NO SMOKING
PARK BRK
PRED W/S OFF
TCAS STBY
APU AVAIL
FOB: 6860 KG
LDG GEAR
BRK FAN
AUTO BRK
LO
MED
MAX
A/SKID & N/W S
OFF
TERR ON ND
F/CTL
SPD BRK
L AIL
R AIL
ELAC 1
SEC 1
PITCH TRIM
RUD
L ELEV
R ELEV
TAT +24 °C
SAT +24 °C
19 H 33
GW 59500 KG
UP
DOWN
ACCU
PRESS
BRAKES
SWITCHING
TAKE OFF
ACTIVE
STBY/CRS
118.300
ENG
BLEED
PRESS
ELEC
HYD
FUEL

Telemetry

Page Load Times for one week (25/9-1/10)

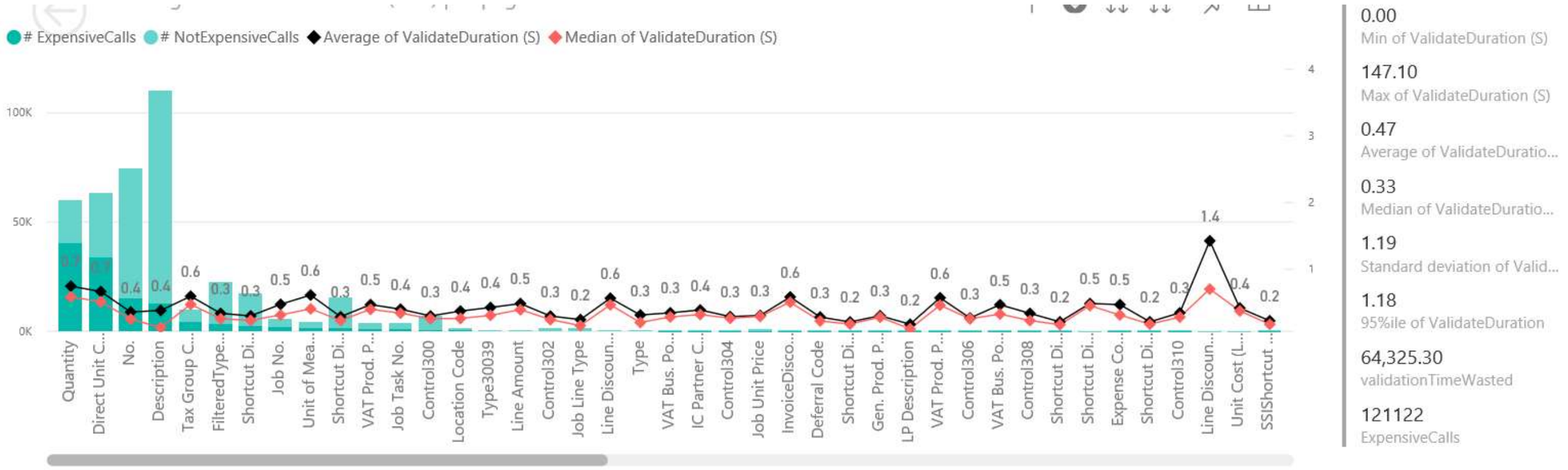


Applying values to controls



- We have work to do on Sales Order and Purchase Invoice
- Blanket Purchase Order and Acc. Schedule Overview are slow
- Hard to push median below 200 ms

Drill-down into Purchase Invoice



0.00
Min of ValidateDuration (S)

147.10
Max of ValidateDuration (S)

0.47
Average of ValidateDuratio...

0.33
Median of ValidateDuratio...

1.19
Standard deviation of Valid...

1.18
95%ile of ValidateDuration

64,325.30
validationTimeWasted

121122
ExpensiveCalls

- Quantity, Direct Unit Cost, No., Description
  - Recalculate totals
- What's up with Line Discount?

Long running queries for last 72 hours. Note that a number of system calls are not registered per alternateTenantId

navTenantId	ExecutionTime	Count	AvgExecutionTime	alternateTenantIds
	46308668	4607	10,051.81	
	22024899	2230	9,876.64	
	5804720	765	7,587.87	
	3606921	598	6,031.64	
	3137510	299	10,493.34	
	2854611	212	13,465.15	
	1634987	118	13,855.82	
	733480	112	6,548.93	
	570846	100	5,708.46	
	1010037	88	11,477.69	
	1366413	83	16,462.81	
	688498	65	10,592.28	
	768523	61	12,598.74	
	512601	53	9,671.72	
	347806	48	7,245.96	
	407476	40	10,186.90	
	254215	36	7,061.53	
Total	**105160247**	**10526**	**9,990.52**	

9/23/2018 9:30:25 PM

AppObjectType	AppObjectId	Count
		2100
Page	31	157
CodeUnit	1281	132
CodeUnit	9010	115
Report	357	105
CodeUnit	1439	102
Table	472	81
CodeUnit	1280	71
CodeUnit	1437	68
CodeUnit	2160	65
CodeUnit	5340	65
CodeUnit	396	34
Table	5338	32
CodeUnit	1326	29
CodeUnit	8611	26
Table	1502	21
Total		**10526**

platformVersion	Count
12.3.23590.24511	10202
12.3.23590.23828	225
12.3.23590.23730	19
12.1.22475.23032	14
13.0.24623.0	14
12.2.22932.23595	11
13.0.24279.0	11
13.0.23366.0	10
13.0.24569.0	10
12.3.23452.0	4
13.0.24425.0	4
12.0.21405.22470	2
Total	**10526**

env_cloud_name	Count of TaskId
D365FF	10300
INV	226
Total	**10526**

ALCallStack	Count	Avg	Median
"CRM Customer-Contact Link"(CodeUnit 5351).SyncPrimaryContactLinkFromCRMAccountPrimaryContactId line 2 "CRM Customer-Contact Link"(CodeUnit 5351).OnRun(Trigger) line 9 "Job Queue Start Codeunit"(CodeUnit 449).OnRun(Trigger) line 6 "Job Queue Dispatcher"(CodeUnit 448).HandleRequest line 18 "Job Queue Dispatcher"(CodeUnit 448).OnRun(Trigger) line 12	6930	9,936.98	10,005.00
"Item List"(Page 31).OnNextRecord(Trigger) line 8	144	10,460.53	10,425.00
"Azure AD User Management"(CodeUnit 9010).UpdateUserFromAzureGraph line 2	115	6,602.19	6,569.00

QueryString	Count	Avg	Median
SELECT "18"."timestamp","18"."No_","18"."Name","18"."Search Name","18"."Name 2","18"."Address","18"."Address 2","18"."City","18"."Contact","18"."Phone No_","18"."Telex No_","18"."Document Sending Profile","18"."Our Account No_","18"."Territory Code","18"."Global Dimension 1 Code","18"."Global Dimension 2 Code","18"."Chain Name","18"."Budgeted Amount","18"."Credit Limit (LCY)","18"."Customer Posting Group","18"."Currency Code","18"."Customer Price Group","18"."Language Code","18"."Statistics Group","18"."Payment Terms Code","18"."Fin_ Charge Terms Code","18"."Salesperson Code","18"."Shipment Method Code","18"."Shipping Agent Code","18"."Place of Export","18"."Invoice Disc_ Code","18"."Customer Disc_ Group","18"."Country_Region Code","18"."Collection Method","18"."Amount","18"."Blocked","18"."Invoice Copies","18"."Last Statement No_","18"."Print	3547	10,003.99	10,004.00

Page loading	• Business Manager Role Center • Customer card
Data entry	• Sales Order subform • Purchase Invoice • Purchase order
Reports/Batch jobs	• Match bank entries • Suggest Worksheet Lines
SQL queries	• CRM contact sync • Item list - OnFindRecord

Telemetry insights – top time consumers

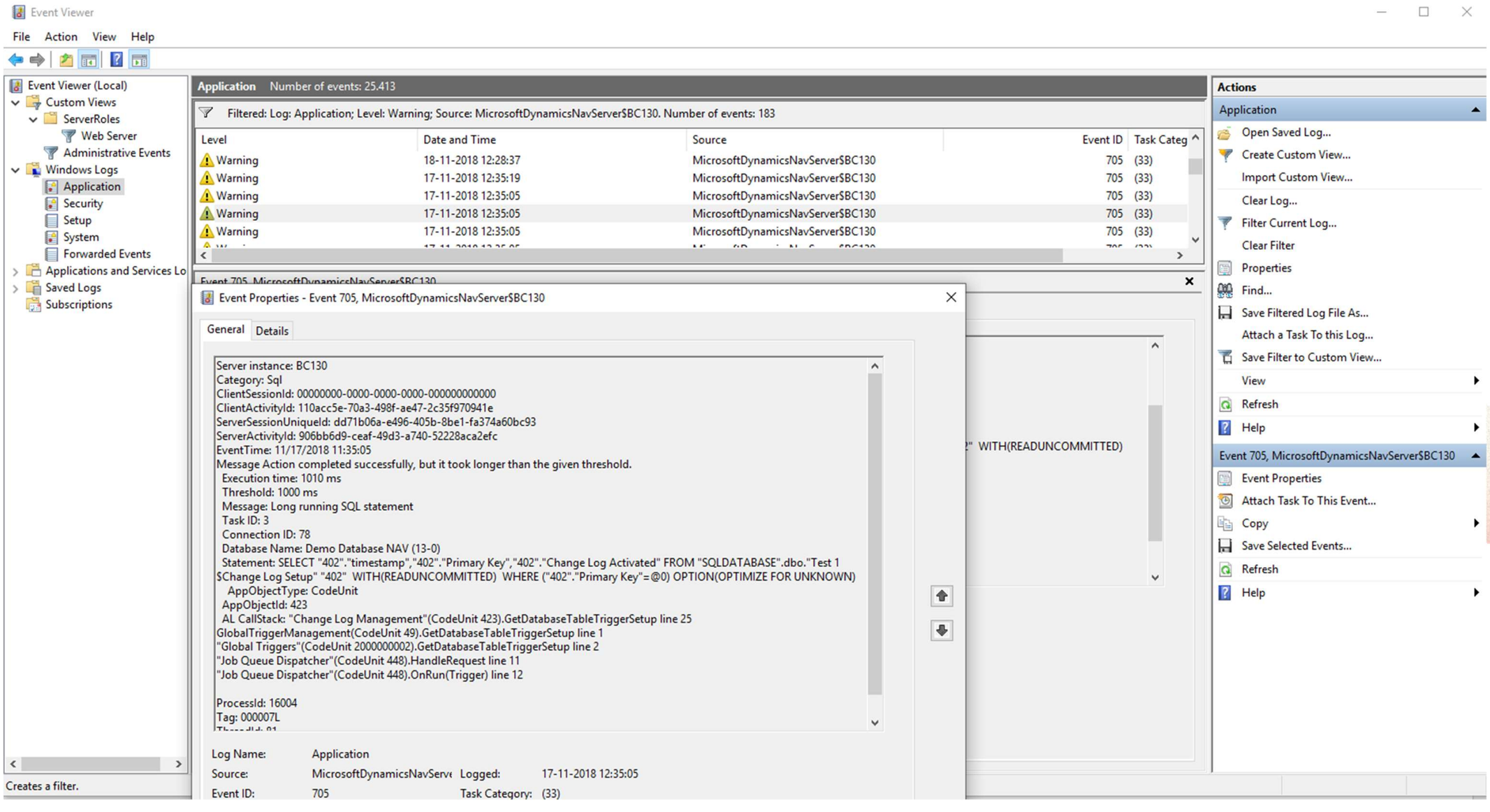
Getting telemetry locally

- **All** telemetry is logged to ETW
- **Event Viewer**
  - OK for viewing Warnings & Errors
  - Impractical for general telemetry
- **Build a custom event listener**
  - Use Nuget package from Microsoft
  - Send data to Log Analytics
- We reserve the right to change the telemetry format

```
Microsoft-DynamicsNAV-Common            {E8077DE8-19F5-54B6-ABCA-708D7816893F}
Microsoft-DynamicsNAV-Server            {85423FD1-C021-5A63-F214-C4819F8809F3}
```

```
<ItemGroup>
  <PackageReference Include="Microsoft.Diagnostics.Tracing.TraceEvent" Version="2.0.30" />
</ItemGroup>
```

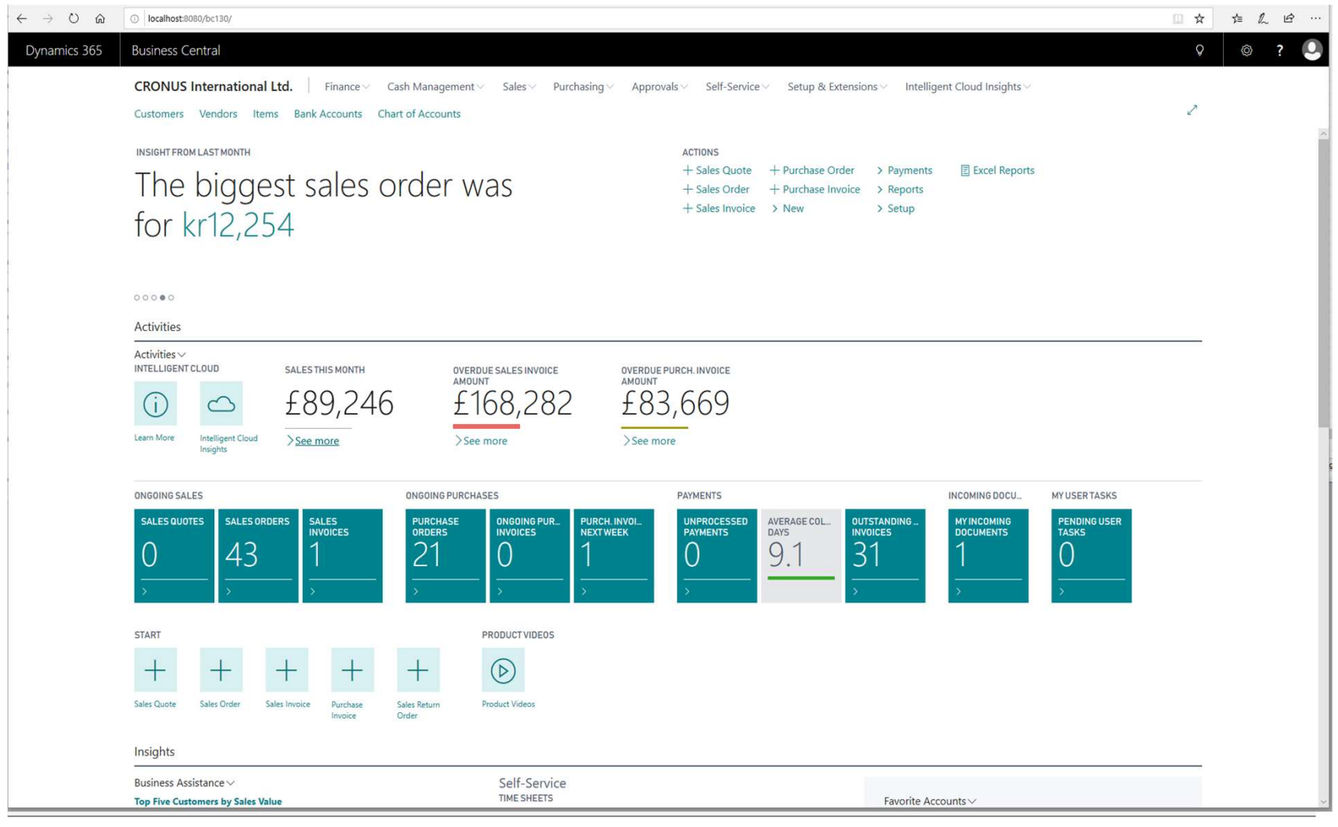
Long running queries in event viewer



C/AL Examples

What is the first thing our users see?

- Nothing!
  - They are waiting for ‘CompanyOpen’ to finish
  - …including all the subscribers to OnCompanyOpen
  - Then they wait for the role center parts to update



Opening of pages

- FlowFields
  - We see some flowfields based on
- FactBoxes
  - We want to impress our customers with a lot if in
  - A lot.
- OnAfterGetRecord
  - Often we have heavy code here.
  - We sometimes call functions for every single record in a repeater, instead of just the one that has focus (OnAfterGet**Current**Record)
- OnOpenPage/OnInitPage
  - Sometimes we some initialization before we display the page

We have a Spring19 feature for static code analysis for detecting FlowFields on tables that reference un-indexed data.

You are the captain of the ship! If your customer requests expensive factboxes, it's your responsibility to resist.

Once the page is open: Field validation…

- CurrPage.UPDATE
- Commits
- Expensive calculations
- Totals

Why do we only focus on SQL statements?

A. GLEntry.Amount := ROUND(34 * i / 7,0.01);
B. IF GLEntry.GET(i) THEN;

A takes 200 times longer than B
on a developer machine... - on 3-tier it’s x800!! (Azure, P1 DB)

A few oops's from ourselves…

Ex. 1: Job Queue failing (Bug #261328)

NAV TECH DAYS 2018
mibuso.com

Symptom: Randomly, jobs would fail with one of these errors:

1. *The operation could not complete because a record in the **Job Queue Entry** table was locked by another user. Please retry the activity.*
2. *The activity was deadlocked with another user who was modifying the **Job Queue Entry** table. Please retry the activity.*
3. *The operation could not complete because a record in the **Scheduled Task** table was locked by another user. Please retry the activity.*
4. *The activity was deadlocked with another user who was modifying the **Scheduled Task** table. Please retry the activity.*

Ex. 1 Problem #1

We want to make sure that only one job with the same Category runs at the same time.

We had a JobQueueEntry.LOCKTABLE before this function, to ensure we have the latest, freshest version.

```
LOCAL WaitForOthersWithSameCategory(VAR CurrJobQueueEntry : Record "Job Queue Entry") : Boolean
OnBeforeWaitForOthersWithSameCategory(CurrJobQueueEntry);

IF CurrJobQueueEntry."Job Queue Category Code" = '' THEN
  EXIT(FALSE);

WITH JobQueueEntry DO BEGIN
  SETFILTER(ID,'<>%1',CurrJobQueueEntry.ID);
  SETRANGE("Job Queue Category Code",CurrJobQueueEntry."Job Queue Category Code");
  SETRANGE(Status,Status::"In Process");
  EXIT(NOT ISEMPTY);
END;
```

Solution: Substitute LOCKTABLE with SELECTLATESTVERSION

Ex. 1 Problem #2

When re-scheduling a recurring job, we wanted to make sure that ‘this’ Job Queue Entry wasn’t already scheduled in another Scheduled Task

```
LOCAL HasSheduledTask() : Boolean
ScheduledTask.SETRANGE(Record,RECORDID);
EXIT(NOT ScheduledTask.ISEMPTY);
```

1. Scheduled Task was in transaction, i.e. locked
2. Field ‘Record’ was not indexed => sql table scan
3. Table scan + lock => entire table gets locked

Solution:

1. There is no (longer) need to check for already scheduled, so function was removed.
2. Index on ‘Record’ was added to ”Scheduled Task” – just to be sure

Ex. 2: Slow reports

Customer incident:

Bug 262040: PowerBI factbox and spinner part do web requests as part of page load, potentially blocking the user for a long time

Bug 262048: Report 104 Customer - Detail Trial Bal. is very slow for customer [redacted]

Bug 262046: Report 10040 Aged Accounts Receivable is very slow for customer [redacted]

Bug 262045: Report 10103 Vendor Account Detail is very slow for customer [redacted]

Ex. 2 One of the issues in REP10103

```
WITH VendLedgerEntry2 DO BEGIN
  RESET;
  IF NOT SETCURRENTKEY("Vendor No.","Currency Code","Posting Date") THEN
    SETCURRENTKEY("Vendor No.","Posting Date","Currency Code");
  SETRANGE("Vendor No.",Vendor."No.");
  SETFILTER("Currency Code",'=%1','');
  IF FINDFIRST THEN BEGIN
    TempCurrency.INIT;
    TempCurrency.Code := '';
    TempCurrency.Description := GLSetup."LCY Code";
    TempCurrency.INSERT;
  END;
END;
WITH Currency DO
  IF FIND('-') THEN
    REPEAT
      VendLedgerEntry2.SETRANGE("Currency Code",Code);
      IF VendLedgerEntry2.FINDFIRST THEN BEGIN
        TempCurrency.INIT;
        TempCurrency.Code := Code;
        TempCurrency.Description := Description;
        TempCurrency.INSERT;
      END;
    UNTIL NEXT = 0;
```

Scenario:
We want to figure our which currencies have been posted on a specific vendor. Saved in *TempCurrency*

1. The blank currency (=local currency)

2. Check the defined currencies

Standard setup has 49 currencies!

So, for each vendor we made 50 lookups for currencies!

Ex. 2 Faster way of finding currencies

NAV TECH DAYS 2018
mibuso.com

```
WITH VendLedgerEntry2 DO BEGIN
  RESET;
  SETCURRENTKEY("Vendor No.","Currency Code");
  SETRANGE("Vendor No.",Vendor."No.");
  SETFILTER("Posting Date",'%1..%2',FromDate,ToDate);
  WHILE FINDFIRST DO BEGIN
    TempCurrency.INIT;
    TempCurrency.Code := "Currency Code";
    TempCurrency.INSERT;
    SETFILTER("Currency Code",'>%1',"Currency Code");
  END;
END;
```

Ex. 3 Factboxes

https://icm.ad.msft.net/imp/v3/incidents/details/72020788/home

Customer: *"Jobs module very slow"*

Microsoft: *"Weird. Let's investigate"*

Database inspection showed 159 jobs, 134398 job ledger entries and 104288 job planning lines, i.e. on average ~1000 of each per job.

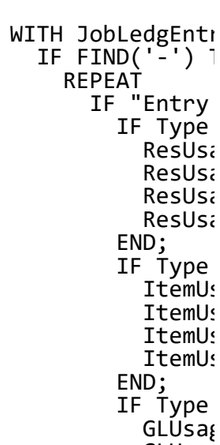
No problem – we have SIFT indexes, right?

This factbox was both on the card and the list, so navigating the list was virtually impossible.

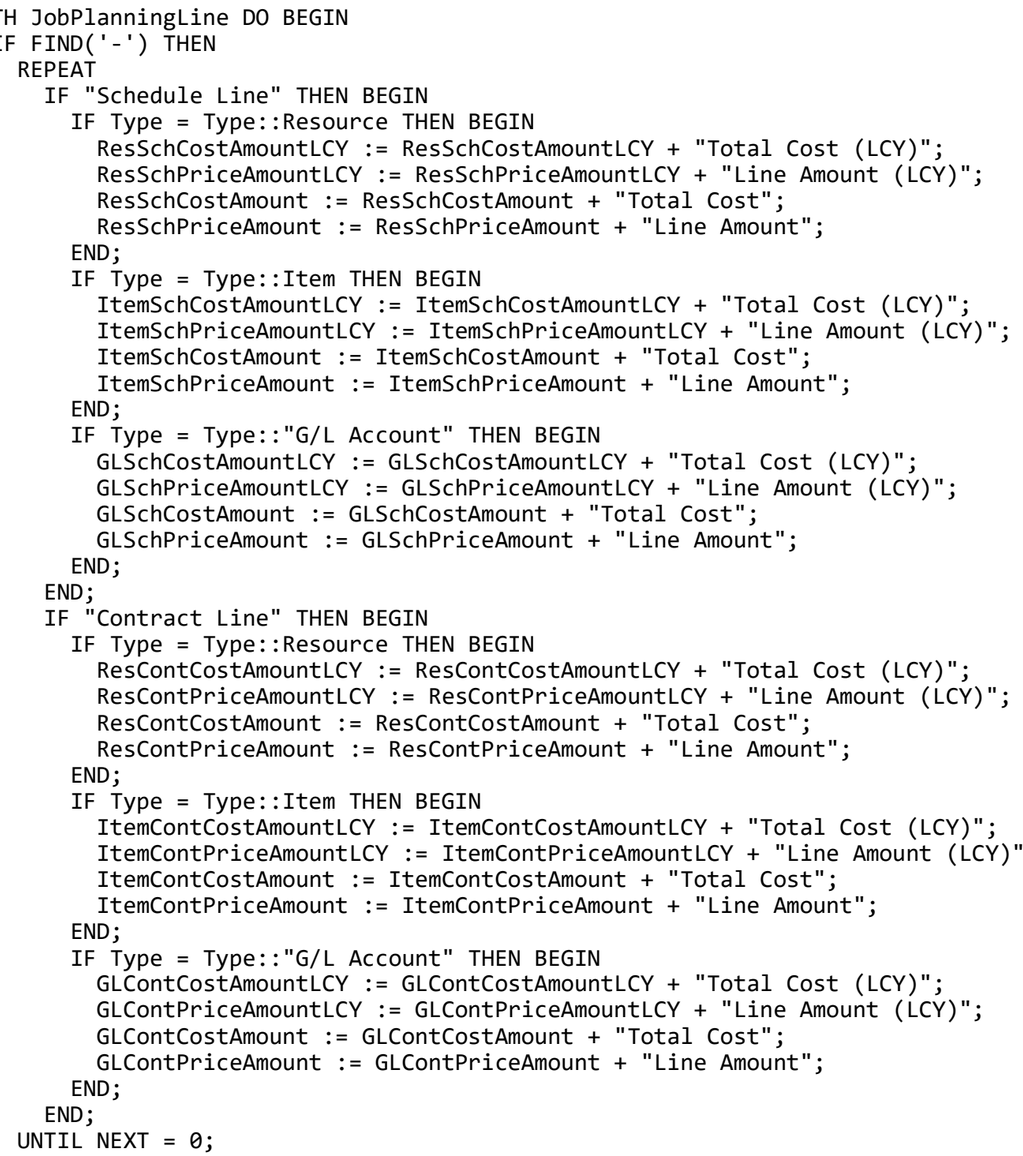
Job Details

Job No.	DEERFIELD, 8 WP
BUDGET COST	
Resource	1.069,20
Item	10.362,10
G/L Account	0,00
Total	**11.431,30**
ACTUAL COST	
Resource	336,88
Item	0,00
G/L Account	0,00
Total	**336,88**
BILLABLE PRICE	
Resource	4.272,00
Item	13.921,00
G/L Account	0,00
Total	**18.193,00**
INVOICED PRICE	
Resource	0,00
Item	0,00
G/L Account	0,00
Total	**0,00**

Ex. 3 Careful with your assumptions….



```
WITH JobLedgEntr
  IF FIND('-') T
    REPEAT
      IF "Entry
        IF Type
          ResUsa
          ResUsa
          ResUsa
          ResUsa
        END;
        IF Type
          ItemUs
          ItemUs
          ItemUs
          ItemUs
        END;
        IF Type
          GLUsag
          GLUsag
          GLUsag
          GLUsag
        END;
      END;
      IF "Entry
        IF Type
          ResSa
          ResSa
          ResSa
          ResSa
        END;
        IF Type
          ItemSa
          ItemSa
          ItemSa
          ItemSa
        END;
        IF Type
          GLSale
          GLSale
          GLSale
          GLSale
        END;
      END;
    UNTIL NEXT =
```



```
WITH JobPlanningLine DO BEGIN
  IF FIND('-') THEN
    REPEAT
      IF "Schedule Line" THEN BEGIN
        IF Type = Type::Resource THEN BEGIN
          ResSchCostAmountLCY := ResSchCostAmountLCY + "Total Cost (LCY)";
          ResSchPriceAmountLCY := ResSchPriceAmountLCY + "Line Amount (LCY)";
          ResSchCostAmount := ResSchCostAmount + "Total Cost";
          ResSchPriceAmount := ResSchPriceAmount + "Line Amount";
        END;
        IF Type = Type::Item THEN BEGIN
          ItemSchCostAmountLCY := ItemSchCostAmountLCY + "Total Cost (LCY)";
          ItemSchPriceAmountLCY := ItemSchPriceAmountLCY + "Line Amount (LCY)";
          ItemSchCostAmount := ItemSchCostAmount + "Total Cost";
          ItemSchPriceAmount := ItemSchPriceAmount + "Line Amount";
        END;
        IF Type = Type::"G/L Account" THEN BEGIN
          GLSchCostAmountLCY := GLSchCostAmountLCY + "Total Cost (LCY)";
          GLSchPriceAmountLCY := GLSchPriceAmountLCY + "Line Amount (LCY)";
          GLSchCostAmount := GLSchCostAmount + "Total Cost";
          GLSchPriceAmount := GLSchPriceAmount + "Line Amount";
        END;
      END;
      IF "Contract Line" THEN BEGIN
        IF Type = Type::Resource THEN BEGIN
          ResContCostAmountLCY := ResContCostAmountLCY + "Total Cost (LCY)";
          ResContPriceAmountLCY := ResContPriceAmountLCY + "Line Amount (LCY)";
          ResContCostAmount := ResContCostAmount + "Total Cost";
          ResContPriceAmount := ResContPriceAmount + "Line Amount";
        END;
        IF Type = Type::Item THEN BEGIN
          ItemContCostAmountLCY := ItemContCostAmountLCY + "Total Cost (LCY)";
          ItemContPriceAmountLCY := ItemContPriceAmountLCY + "Line Amount (LCY)";
          ItemContCostAmount := ItemContCostAmount + "Total Cost";
          ItemContPriceAmount := ItemContPriceAmount + "Line Amount";
        END;
        IF Type = Type::"G/L Account" THEN BEGIN
          GLContCostAmountLCY := GLContCostAmountLCY + "Total Cost (LCY)";
          GLContPriceAmountLCY := GLContPriceAmountLCY + "Line Amount (LCY)";
          GLContCostAmount := GLContCostAmount + "Total Cost";
          GLContPriceAmount := GLContPriceAmount + "Line Amount";
        END;
      END;
    UNTIL NEXT = 0;
```

Good idea with few records...

But this customer had ~1000 records in each of the datasets

Ex. 3 Consider CALCSUMS instead


```
WITH JobPlanningLine DO BEGIN
  SETRANGE("Schedule Line");
  SETRANGE("Contract Line");
  IF PlanLineTypeParm = PlanLineType::Schedule THEN
    SETRANGE("Schedule Line",TRUE)
  ELSE
    SETRANGE("Contract Line",TRUE);
  SETRANGE(Type,TypeParm);

  CALCSUMS("Total Cost (LCY)","Line Amount (LCY)","Total Cost","Line Amount");

  JobPlanAmounts[1 + PlanLineTypeParm,1 + TypeParm,1 + AmountType::TotalCostLCY] := "Total Cost (LCY)";
  JobPlanAmounts[1 + PlanLineTypeParm,1 + TypeParm,1 + AmountType::LineAmountLCY] := "Line Amount (LCY)";
  JobPlanAmounts[1 + PlanLineTypeParm,1 + TypeParm,1 + AmountType::TotalCost] := "Total Cost";
  JobPlanAmounts[1 + PlanLineTypeParm,1 + TypeParm,1 + AmountType::LineAmount] := "Line Amount";
END;
```

Translates roughly to
SELECT sum(...),sum(..),..
FROM <some table>
WHERE <conditions>

Ex. 4 OnAfterGetRecord on lists

Bug description (customer case)

"We have a customer case where they run a scenario:

- *Opened Windows Client* ***-> 36 seconds*** *– very slow, but, yeah, the service has been restarted*
- *Opened Posted Sales Invoices* ***-> 9 seconds*** *– very slow*
- *In quick search entered a value for No.* ***-> 12 seconds*** *– very slow*
- *Opened document page for the resulting posted sales invoice* ***-> 11 seconds*** *– very slow*
- *Closed document page* ***-> 8 seconds*** *– very slow"*

Ex. 4 Posted Sales Invoices

CRONUS International Ltd. | Finance | Cash Management | **Sales** | Purchasing | Approvals | Self-Service | Setup & Extensions

Posted Sales Invoices: All | Search | Delete | Invoice | Show Attached | Open in Excel | More options

NO.	CUSTOMER NO.	CUSTOMER	CURRENCY CODE	DUE DATE	AMOUNT	AMOUNT INCLUDING VAT	REMAINING AMOUNT	LOCATION CODE
103033	10000	The Cannon Group PLC		28-02-2020	30,70	38,38	38,38	BLUE
103032	10000	The Cannon Group PLC		23-02-2020	30,70	38,38	38,38	BLUE
103027	35451236	Gagn & Gaman	ISK	29-01-2020	88.164,00	88.164,00	88.164,00	YELLOW
103026	35963852	Heimilisprydi	ISK	31-01-2020	203.417,25	203.417,25	203.417,25	YELLOW
103025	47563218	Klubben	NOK	31-01-2020	114.728,73	114.728,73	114.728,73	YELLOW
103024	20000	Selangorian Ltd.		27-01-2020	916,26	1.145,33	1.145,33	
103023	01445544	Progressive Home Furnishings	USD	01-02-2020	2.310,38	2.310,38	2.310,38	YELLOW
103003	30000	John Haddock Insurance Co.		31-01-2020	5.454,00	5.999,40	5.999,40	
103002	20000	Selangorian Ltd.		03-02-2020	6.337,98	6.971,78	6.971,78	
103001	10000	The Cannon Group PLC		20-02-2020	7.438,50	8.182,35	8.182,35	BLUE
103022	46897889	Englunds Kontorsmöbler AB	SEK	31-01-2020	6.807,56	6.807,56	6.807,56	YELLOW
103021	49633663	Autohaus Mielberg KG	EUR	25-01-2020	1.441,31	1.441,31	0,00	GREEN
103028	10000	The Cannon Group PLC		15-02-2020	3.281,50	4.101,88	4.101,88	BLUE
103020	32656565	Antarcticopy	EUR	20-01-2020	3.999,38	3.999,38	3.999,38	YELLOW
103019	20000	Selangorian Ltd.		26-01-2020	172,66	215,83	215,83	
103031	30000	John Haddock Insurance Co.		31-01-2020	688,90	861,13	861,13	BLUE
103018	20000	Selangorian Ltd.		22-01-2020	629,92	787,40	787,40	
103017	43687129	Designstudio Gmunden	EUR	10-02-2020	3.868,23	3.868,23	3.868,23	RED
103016	42147258	BYT-KOMPLET s.r.o.	CZK	10-02-2020	60.218,65	60.218,65	60.218,65	RED
103015	10000	The Cannon Group PLC		02-02-2020	6.615,23	8.269,04	8.269,04	BLUE
103014	49858585	Hotel Pferdesee	EUR	02-02-2020	1.232,24	1.232,24	1.232,24	GREEN
103013	43687129	Designstudio Gmunden	EUR	02-02-2020	7.248,48	7.248,48	7.248,48	RED
103012	43687129	Designstudio Gmunden	EUR	02-02-2020	5.798,78	5.798,78	5.798,78	RED
103011	43687129	Designstudio Gmunden	EUR	02-02-2020	4.349,00	4.349,00	4.349,00	RED
103010	49633663	Autohaus Mielberg KG	EUR	16-01-2020	6.271,29	6.271,29	6.271,29	GREEN

Ex. 4 Page 143 Posted Sales Invoices

Hint...

```
OnAfterGetRecord=VAR
                   SalesInvoiceHeader@1000 : Record 112;
                 BEGIN
                   DocExchStatusStyle := GetDocExchStatusStyle;
                   SalesInvoiceHeader.COPYFILTERS(Rec);
                   IsPostedSalesInvoicesEmpty := SalesInvoiceHeader.ISEMPTY;
                   SalesInvoiceHeader.SETFILTER(
                     "Document Exchange Status",'<>%1',
                     "Document Exchange Status"::"Not Sent");
                   DocExchStatusVisible := NOT SalesInvoiceHeader.ISEMPTY;
```

Why not just:
IsPostedSalesInvoicesEmpty := FALSE; ?

Why calculate this for every row in the list?
Why not just OnAfterGetCurrentRecord?
Or even in OnOpenPage?

Other Patterns

Finding a Setup Record

E.g. PrinterSelection in codeunit 44

Potentially 4 sql calls

```
IF NOT PrinterSelection.GET(USERID,ReportID) THEN
  IF NOT PrinterSelection.GET('',ReportID) THEN
    IF NOT PrinterSelection.GET(USERID,0) THEN
      IF PrinterSelection.GET('',0) THEN;
```

1 sql call

```
PrinterSelection.SETFILTER("User ID",'%1|%2',USERID,'');
PrinterSelection.SETFILTER("Report ID",'%1|%2',ReportID,0);
PrinterSelection.SETCURRENTKEY("Report ID","User ID");
IF PrinterSelection.FINDLAST THEN;
```

Tests are inconclusive.

Seem to slightly favor the existing code due to primary key caching.

Calculate Sums – even when no SIFT

```
CalcSum1()
PerfTestTable.SETRANGE(Weekday,PerfTestTable.Weekday::Thursday);
IF PerfTestTable.FIND('-') THEN
  REPEAT
    Total := Total + PerfTestTable.Amount;
  UNTIL PerfTestTable.NEXT = 0;
```

~350ms

```
CalcSum2()
PerfTestTable.SETCURRENTKEY(Weekday);
PerfTestTable.SETRANGE(Weekday,PerfTestTable.Weekday::Thursday);
PerfTestTable.CALCSUMS(Amount);
```

~12ms

Old example from upgrade

```
Loop1()
WITH Item DO BEGIN
  SETFILTER("Roll-up Kit Pricing",'<>%1',0);
  MODIFYALL("Roll-up Kit Pricing",0);

  SETFILTER("Components on Sales Orders",'<>%1',0);
  MODIFYALL("Components on Sales Orders",0);

  SETFILTER("Components on Invoices",'<>%1',0);
  MODIFYALL("Components on Invoices",0);
END;
ERROR('hi');
```

~1 sec.

Original

```
Loop2()
WITH Item DO
  IF FIND('-') THEN
    REPEAT
      IF ("Roll-up Kit Pricing" <> "Roll-up Kit Pricing"::Never) OR
         ("Components on Sales Orders" <> 0) OR
         ("Components on Invoices" <> 0)
      THEN BEGIN
        "Roll-up Kit Pricing" := 0;
        "Components on Sales Orders" := 0;
        "Components on Invoices" := 0;
        MODIFY;
      END;
    UNTIL NEXT = 0;
ERROR('hi');
```

~12 sec.

```
Loop3()
WITH Item DO BEGIN
  MODIFYALL("Roll-up Kit Pricing",0);
  MODIFYALL("Components on Sales Orders",0);
  MODIFYALL("Components on Invoices",0);
END;
ERROR('hi');
```

~5 sec.

MODIFY ~15k Item Ledger Entries

```
LOCAL SimpleLoop() : Integer
NewString := FORMAT(RANDOM(9999999));
T0 := TIME;
WITH ItemLedgerEntry DO BEGIN
  LOCKTABLE;
  IF FINDSET THEN
    REPEAT
      IF (Quantity > 1) OR (Description > 'Berlin') OR ("Dimension Set ID" > 0) THEN BEGIN
        IF Quantity > 1 THEN
          "Cross-Reference No." := NewString;
        IF Description > 'Berlin' THEN
          "Originally Ordered No." := NewString;
        IF "Dimension Set ID" > 0 THEN
          "Item Category Code" := NewString;
        MODIFY;
      END;
    UNTIL NEXT = 0;
  END;
COMMIT;
EXIT(TIME - T0);
```

27s

```
LOCAL CrudeModifyAll() : Integer
NewString := FORMAT(RANDOM(9999999));
T0 := TIME;
WITH ItemLedgerEntry DO BEGIN
  LOCKTABLE;
  MODIFYALL("Cross-Reference No.",NewString);
  MODIFYALL("Originally Ordered No.",NewString);
  MODIFYALL("Item Category Code",NewString);
END;
COMMIT;
EXIT(TIME - T0);
```

15s

```
LOCAL FilteredModifyAll() : Integer
NewString := FORMAT(RANDOM(9999999));
T0 := TIME;
WITH ItemLedgerEntry DO BEGIN
  LOCKTABLE;
  SETFILTER(Quantity,'>%1',1);
  MODIFYALL("Cross-Reference No.",NewString);
  RESET;
  SETFILTER(Description,'>%1','Berlin');
  MODIFYALL("Originally Ordered No.",NewString);
  RESET;
  SETFILTER("Dimension Set ID",'>%1',0);
  MODIFYALL("Item Category Code",NewString);
END;
COMMIT;
EXIT(TIME - T0);
```

13s

MODIFY 100k customers


```
8  LOCAL SimpleLoop() : Integer
9  NewString := FORMAT(RANDOM(9999999));
10 T0 := TIME;
11 WITH Customer DO BEGIN
12   LOCKTABLE;
13   IF FINDSET THEN
14     REPEAT
15       IF ("Currency Code" = 'USD') OR (City > 'Berlin') OR (Name > 'King') THEN BEGIN
16         IF "Currency Code" = 'USD' THEN
17           "Name 2" := NewString;
18         IF City > 'Berlin' THEN
19           "Telex Answer Back" := NewString;
20         IF Name > 'King' THEN
21           "Telex No." := NewString;
22         MODIFY;
23       END;
24     UNTIL NEXT = 0;
25   END;
26 COMMIT;
```

268s

```
30 LOCAL CrudeModifyAll() : Integer
31 NewString := FORMAT(RANDOM(9999999));
32 T0 := TIME;
33 WITH Customer DO BEGIN
34   LOCKTABLE;
35   MODIFYALL("Name 2",NewString);
36   MODIFYALL("Telex Answer Back",NewString);
37   MODIFYALL("Telex No.",NewString);
38 END;
39 COMMIT;
```

937s

```
43 LOCAL FilteredModifyAll() : Integer
44 NewString := FORMAT(RANDOM(9999999));
45 T0 := TIME;
46 WITH Customer DO BEGIN
47   LOCKTABLE;
48   SETFILTER("Currency Code",'%1','USD');
49   MODIFYALL("Name 2",NewString);
50   RESET;
51   SETFILTER(City,'>%1','Berlin');
52   MODIFYALL("Telex Answer Back",NewString);
53   RESET;
54   SETFILTER(Name,'>%1','King');
55   MODIFYALL("Telex No.",NewString);
56 END;
57 COMMIT;
```

319s

Gotcha!

Subscriber Codeunit...	Subscriber Codeunit Name	Subscriber Function	Event Type	Publisher Object ...	Publisher Object ID	Publisher Object Name	Published Function	Active	Number of Calls	Subscriber Instance
1173	Document Attachment Mgmt	DeleteAttachedDocumentsOnAfterDeleteCus...	Trigger	Table	18	Customer	OnAfterDeleteEvent	☑	0	Static-Automatic
1535	Approvals Mgmt.	DeleteApprovalEntriesAfterDeleteCustomer	Trigger	Table	18	Customer	OnAfterDeleteEvent	☑	0	Static-Automatic
1520	Workflow Event Handling	RunWorkflowOnCustomerChanged	Trigger	Table	18	Customer	OnAfterModifyEvent	☑	2944	Static-Automatic
9150	My Records Update Mgt.	UpdateMyCustomerAfterCustomerModify	Trigger	Table	18	Customer	OnAfterModifyEvent	☑	2944	Static-Automatic
1550	Record Restriction Mgt.	RemoveCustomerRestrictionsBeforeDelete	Trigger	Table	18	Customer	OnBeforeDeleteEvent	☑	0	Static-Automatic

Event Subscribers

- May or may not be expensive in themselves
- But they prevent bulk updates
  - MODIFYALL
  - DELETEALL

Caching… Or not

With update 19 (~Fall release) we made PowerBI default active for all users.

1. We upgraded some customer clusters to Update19
2. When users opened their role center, the PowerBI factbox automatically uploaded PBI packages to the PowerBI service
3. After a short while, the PowerBI called back to the BC servers to get data.
4. All at the same time…
5. It turns out that several of our web services (pages and queries) are expensive to query
6. Result: Our servers max'ed out on CPU and our users couldn't access the service.
7. One of our web services, page 197 returns finance data that requires some dataprocessing (Account Schedule) to deliver its data.

Page 197 web service (~PowerBIFinance)

Each cell is calculated using Account Schedule -> 1 sql call

Same data for all users

Minimal change over a day

Caching to the rescue!

Incremental Update

New entries are usually for a very recent period – no need to recalculate old periods

Account Schedule

No filters applied

Date	Clos... Peri...	Schedul...			Net Change Actual	Balance at Date Actual	Net Change Budget	Balance at Date Budget	Net Change	Balance at	Net Change	Balance at	Net Change Forecast	Balance at Date Forecast
01-02-2019	☑	I_MINTRIAL	30	Gross Margin	298.815,14	519.701,45	0,00	0,00					298.815,14	519.701,45
01-02-2019	☑	I_MINTRIAL	40	Gross Margin %	45,07	42,84	0,00	0,00					45,07	42,84
01-02-2019	☑	I_MINTRIAL	50	Operating Expenses	224.030,04	438.951,96	0,00						224.030,04	438.951,96
01-02-2019	☑	I_MINTRIAL	60	Operating Margin	74.785,10	80.749,49		0,00	0,00	0,00	0,00	0,00	74.785,10	80.749,49
01-02-2019	☑	I_MINTRIAL	70	Op ting Margin %	11,28		0,00	0,00	0,00	0,00	0,00	0,00	11,28	6,66
01-02-2019	☑	I_MINTRIAL	80	Ot r Expen s		197,94	0,00	0,00					197,94	197,94
01-02-2019	☑	I_MINTRIAL	90	Income f terest and Tax	74.587,16	80.551,55	0,00	0,00					74.587,16	80.551,55
01-03-2019	☑	I_CACYCLE	10	Total Revenue	-1.685.536,35	-1.68 536,35	00						-1.685.536,35	-1.685.536,35
01-03-2019	☑	I_CACYCLE	20	Total Receivables		407 28,59	00						407.828,59	407.828,59
01-03-2019	☑	I_CACYCLE	30	Total Payables	4								42.797,21	-286.455,26
01-03-2019	☑	I_CACYCLE	40	Total In y			0,00	0,00					0,00	915.168,76
01-03-2019	☑	I_CACYCLE	100	s Outstanding	87,10	87,10	0,00	0,00	0,00	0,00	0,00	0,00	87,10	87,10
01-03-2019	☑	I_CACYCLE	110	Days of Payment Outstanding	-9,	61,18	0,00	0,00	0,00	0,00	0,00	0,00	-9,14	61,18
01-03-2019	☑	I_CACYCLE	120	Days Sales of Inventory	,00	195,46	0,00	0,00	0,00	0,00	0,00	0,00	0,00	195,46
01-03-2019	☑	I_CACYCLE	200	Cash Cycle (Days)	77,96	-47,18	0,00	0,00	0,00	0,00	0,00	0,00	77,96	-47,18
01-03-2019	☑	I_INCEXP	10	Total Revenue (Credit)	-472.502,17	-1.685.536,35	0,00	0,00	0,00	0,00	0,00	0,00	-472.502,17	-1.685.536,35
01-03-2019	☑	I_INCEXP	11	Total Revenue	472.502,17	1.685.536,35	0,00	0,00	0,00	0,00	0,00	0,00	472.502,17	1.685.536,35
01-03-2019	☑	I_INCEXP	20	Total Goods Sold	323.254,08	1.016.586,81	0,00	0,00	0,00	0,00	0,00	0,00	323.254,08	1.016.586,81
01-03-2019	☑	I_INCEXP	30	Total External Costs	48.969,23	191.202,88	0,00	0,00	0,00	0,00	0,00	0,00	48.969,23	191.202,88
01-03-2019	☑	I_INCEXP	40	Total Personnel Costs	156.439,71	451.916,02	0,00	0,00	0,00	0,00	0,00	0,00	156.439,71	451.916,02
01-03-2019	☑	I_INCEXP	50	Total Depr. on Fixed Assets	750,00	1.992,00	0,00	0,00	0,00	0,00	0,00	0,00	750,00	1.992,00
01-03-2019	☑	I_INCEXP	60	Other Expenses	512.40	710.34	0.00	0.00	0.00	0.00	0.00	0.00	512.40	710.34

RoleCenters

- We want to provide insightful and actionable information on the front page
- The user starts in the role center
- The user returns to the role center after almost every action
- …so the role center needs to be snappy!
- The Business Manager Role Center shown on next slide takes ~5s to open (on a private deployment with a large Azure VM and P1 DB!)
- 95 Percentile shows ~10s in the telemetry, which includes demo companies.

HEADLINE

Want to learn more about Business Central?

ACTIONS

+ Sales Quote | + Sales Order | + Sales Invoice | + Purchase Order | + Purchase Invoice | > New | > Payments | > Reports | > Setup | Excel Reports

Caching?

Activities

Activities

INTELLIGENT CLOUD

Learn More | Intelligent Cloud Insights

SALES THIS MONTH	OVERDUE SALES INVOICE AMOUNT	OVERDUE PURCH. INVOICE AMOUNT
0£	165.229.932£	454.564£
> See more	> See more	> See more

ONGOING SALES			ONGOING PURCHASES			PAYMENTS			INCOMING DOC...	MY USER TASKS
SALES QUOTES	SALES ORDERS	SALES INVOICES	PURCHASE ORDERS	ONGOING PU... INVOICES	PURCH. INVOI... NEXT WEEK	UNPROCESSED PAYMENTS	AVERAGE COL... DAYS	OUTSTANDIN... INVOICES	MY INCOMING DOCUMENTS	PENDING USER TASKS
0	42	1	21	1	1	0	12,0	31	1	0

START

Sales Quote | Sales Order | Sales Invoice | Purch. Invoice | Sales Return Order

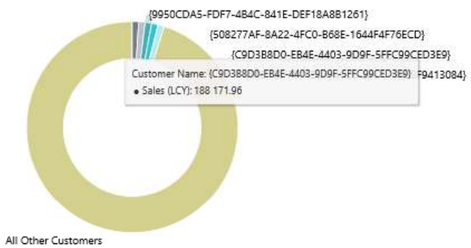
PRODUCT VIDEOS

Product Videos

Insights

Business Assistance

Top Five Customers by Sales Value



Self-Service

TIME SHEETS	PENDING TIME SHEETS			APPROVALS
OPEN TIME SHEETS	SUBMITTED TI... SHEETS	REJECTED TIME SHEETS	APPROVED TI... SHEETS	REQUESTS TO APPROVE
0	0	0	0	0

Favorite Accounts

ACCOUNT NO.	NAME	BALANCE
2910	Cash	7.164,36
2920	Bank, LCY	102.475,44
2930	Bank Currencies	142.489,08
2940	Giro Account	8.972.686,44
5310	Revolving Credit	-49.714.357,68

Trial Balance

Description	01-01-20..31-01-20	01-02-20..29-02-20
Total Revenue	3.528.302,81	4.656,17
Total Cost	-7.722.462,31	-31.840,31
Gross Margin	-4.194.159,50	-27.184,14
Gross Margin %	**-118,87**	**-583,83**
Operating Expenses	10.227.893,94	595,69
Operating Margin	-14.422.053,44	-27.779,83
Operating Margin %	**-408,75**	**-596,62**

Report Inbox

CREATED DATE-TIME	DESCRIPTION	OUTPUT TYPE
18-10-2018 21:30	**Balance Sheet**	**PDF**

Power BI Reports

Get started with Power BI

Where is the slow code?

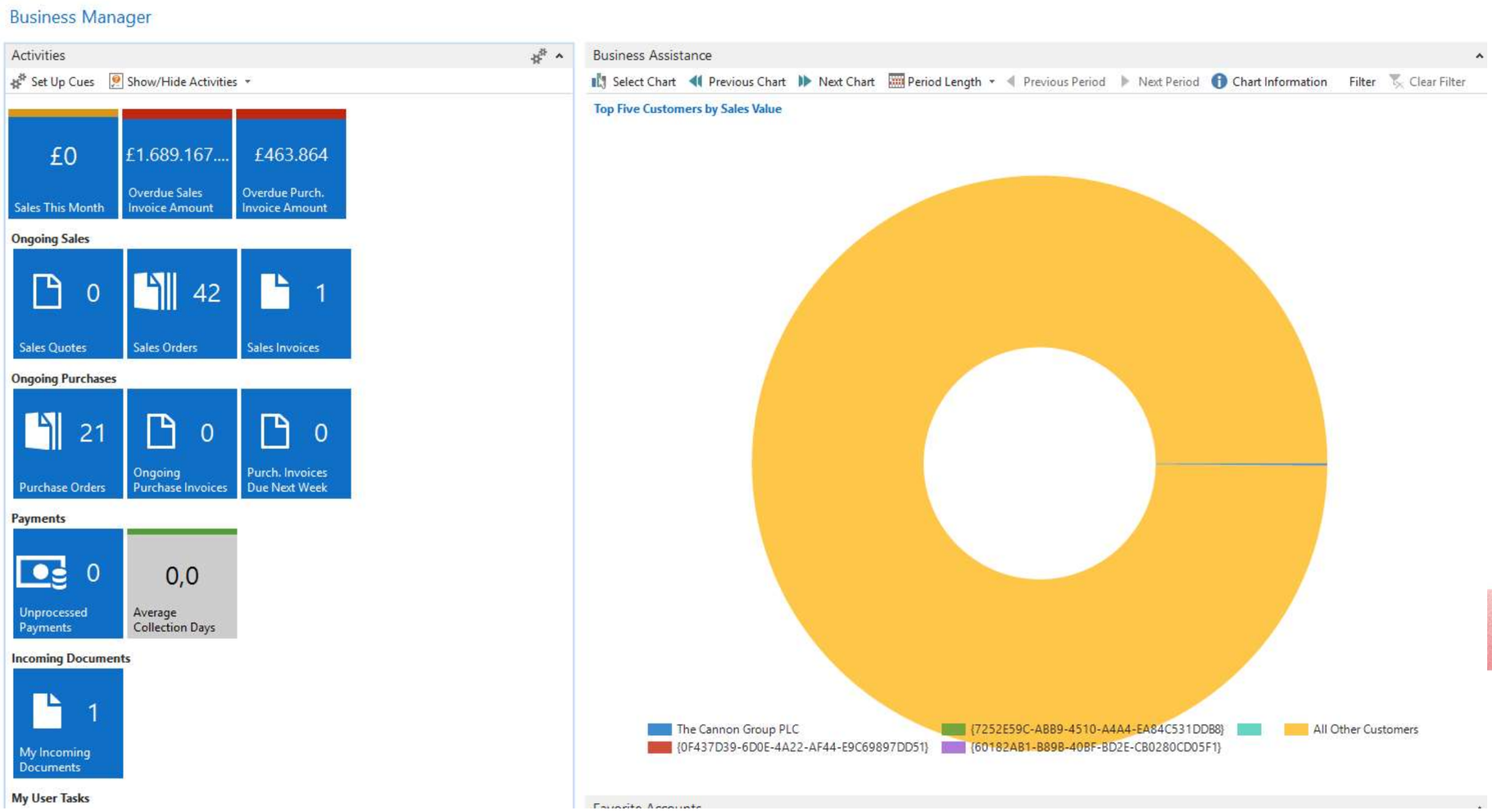
Let’s try the NAV App Performance Profiler

https://blogs.msdn.microsoft.com/nav/2014/07/04/introducing-the-microsoft-dynamics-nav-application-profiler/

Updating the RoleCenter

- Set RoleCenter 9022 as default
- Make sure you have some customers (more than CRONUS)
- … And let's fire up our Performance profiler…
- Performance profiler combines a traditional profiler with insights on which SQL statements have been executed.

Top 5 Customers by Sales



Performance Profiler – Open Role Center

NAV TECH DAYS 2018
mibuso.com

System	0	0	Session: 88; User: <pi>EUROPE\bknudsen</pi>;	37.320,00
TableData	0	0	*SQL COMMIT*	2,00
TableData	0	0	*SELECT TOP 1 transaction_id FROM sys.dm_tran_session_tra...*	1,00
TableData	0	0	*SELECT TOP (@0) ISNULL("Performance Profiler Events"."tim...*	13,00
TableData	0	0	*SELECT TOP (@0) ISNULL("Performance Profiler Events"."tim...*	11,00
Page	1310	0	OnInit	11,00
TableData	0	0	*SELECT TOP (1) "2000000177"."timestamp","2000000177"."Ap...*	3,00
Page	1393	0	OnInit	9,00
Page	2196	0	OnInit	9,00
Page	1310	0	OnOpenPage	47,00
TableData	1313	0	GetAmountFormat	25,00
Codeunit	1	0	GetCueStyle	16,00
Page	1392	0	OnOpenPage	12.332,00
Page	9043	0	OnOpenPage	8,00
Page	9153	0	OnOpenPage	8,00
Page	1393	0	OnOpenPage	73,00
Page	6303	0	OnOpenPage	2,00
Page	1392	0	BusinessChart::AddInReady	24.670,00
Page	681	0	PingPong::AddInReady	2,00

Total Duration (ms):	**37.320,00**	Total SQL Queries:	39
Total SQL Query Duration (ms):	5.316,00	Total SQL Query Hit Count:	300329
Max SQL Query Duration (ms):	1.397,00	Max SQL Query Hit Count:	100021

What's wrong with this picture?

100.000 customers

```
LOCAL CalcTopTenSalesCustomers(VAR CustomerName : ARRAY [6] OF Text[50];VAR SalesLCY : ARRAY [6] OF Decimal)
ColumnIndex := 1;
Customer.CALCFIELDS("Sales (LCY)");
Customer.SETCURRENTKEY("Sales (LCY)");
Customer.ASCENDING(FALSE);
WITH Customer DO BEGIN
  IF FIND('-') THEN
    REPEAT
      // Return Sales (LCY) for top 10 customer, and as 11th measure - the sum of Sales (LCY) for all other cu
      IF ColumnIndex <= 5 THEN BEGIN
        CustomerName[ColumnIndex] := Name;
        SalesLCY[ColumnIndex] := "Sales (LCY)";
      END ELSE
        SalesLCY[6] += "Sales (LCY)";
      ColumnIndex := ColumnIndex + 1;
    UNTIL NEXT = 0;
  CustomerName[6] := AllOtherCustomersTxt;
END;
```

One possible improvement


```
LOCAL CalcTopTenSalesCustomers(VAR CustomerName : ARRAY [6] OF Text[50];VAR SalesLCY : ARRAY [6] OF Decimal)
ColumnIndex := 1;
CustLedgerEntry.CALCSUMS("Sales (LCY)");
SalesLCY[6] := CustLedgerEntry."Sales (LCY)";
Customer.SETCURRENTKEY("Sales (LCY)");
Customer.ASCENDING(FALSE);
WITH Customer DO BEGIN
  IF FIND('-') THEN
    REPEAT
      // Return Sales (LCY) for top 10 customer, and as 11th measure - the sum of Sales (LCY) for all other cu
      IF ColumnIndex <= 5 THEN BEGIN
        CustomerName[ColumnIndex] := Name;
        SalesLCY[ColumnIndex] := "Sales (LCY)";
        SalesLCY[6] -= "Sales (LCY)";
      END;
      ColumnIndex := ColumnIndex + 1;
    UNTIL (NEXT = 0) OR (ColumnIndex = 5);
  CustomerName[6] := AllOtherCustomersTxt;
END;
```

Maybe add caching?

```
LOCAL CalcTopTenSalesCustomers(VAR CustomerName : ARRAY [6] OF Text[50];VAR SalesLCY : ARRAY [6] OF Decimal)
IF LastUpdateDateTime > CURRENTDATETIME - 1000 * 60 * 10 THEN BEGIN // refresh after 10 min
  FOR ColumnIndex := 1 TO 6 DO BEGIN
    CustomerName[ColumnIndex] := SavedCustomerName[ColumnIndex];
    SalesLCY[ColumnIndex] := SavedSalesLCY[ColumnIndex];
  END;
  EXIT;
END;
ColumnIndex := 1;
//Customer.SETRANGE("Date Filter",CALCDATE('<-1Y>',WORKDATE),WORKDATE); // maybe only last year
CustLedgerEntry.CALCSUMS("Sales (LCY)");
SalesLCY[6] := CustLedgerEntry."Sales (LCY)";
Customer.SETCURRENTKEY("Sales (LCY)");
Customer.ASCENDING(FALSE);
WITH Customer DO BEGIN
  IF FIND('-') THEN
    REPEAT
      // Return Sales (LCY) for top 10 customer, and as 11th measure - the sum of Sales (LCY) for all other c
      IF ColumnIndex <= 5 THEN BEGIN
        CustomerName[ColumnIndex] := Name;
        SalesLCY[ColumnIndex] := "Sales (LCY)";
        SalesLCY[6] -= "Sales (LCY)";
      END;
      ColumnIndex := ColumnIndex + 1;
    UNTIL (NEXT = 0) OR (ColumnIndex = 5); // *** only top 5
  CustomerName[6] := AllOtherCustomersTxt;
END;
LastUpdateDateTime :=  CURRENTDATETIME;
FOR ColumnIndex := 1 TO 6 DO BEGIN
  SavedCustomerName[ColumnIndex] := CustomerName[ColumnIndex];
  SavedSalesLCY[ColumnIndex] := SalesLCY[ColumnIndex];
END;
```

Performance of Events

Performance of Events

- Subscription models
  - Static-Automatic binding
    - Pros:
      - Always subscribed to the event and no one can turn it off
      - Best for single invocations
      - State can be held in single instance codeunits
    - Cons:
      - Expensive with FOR loops, OnAfterGetRecord etc.
        Each subscriber call might be a complete object creation
      - Even small subscriber classes, are twice as slow as manual binding

Performance of Events

- **Subscription models**
  - **Manual binding**
    - **Pros:**
      - Cheap if no subscription
      - Very good for iterative loops
      - State can be held in normal instance or single instance codeunits
      - If you are running on temporary records, you can skip the binding
    - **Cons:**
      - You are in charge of binding the subscriber

```
52
53
54
55
56
57
58
    - reference
59  codeunit 50104 ManualBoundSubscriber
60  {
61      EventSubscriberInstance = Manual;
62
        - reference
63      [EventSubscriber(ObjectType::Codeunit, Codeunit::LogInManagement, 'OnAfterCompanyOper
64      procedure OnLogin()
65      begin
66      end;
67  }
68
    - reference
69  codeunit 50105 AutomaticBoundSubscriber
70  {
71      EventSubscriberInstance = StaticAutomatic;
72      SingleInstance = true;
        - reference
73      [EventSubscriber(ObjectType::Codeunit, Codeunit::LogInManagement, 'OnAfterCompanyOper
74      procedure OnLogin()
75      begin
76      end;
77  }
78
79
80
81
82
83
```

NAV TECH DAYS 2018
mibuso.com

Subscription Models

Performance of Events

- **What happens during an event**
- IF subscriber attached
  - Get list of subscribers and check
  - IF any static-automatic or manual found
    - CASE of Manual binding
      - Invoke if subscribed
    - **CASE of Static-Automatic**
      - **Check permissions/License**
      - **Find metadata**
      - **Create instance**
  - ELSE
    - RETURN
- ELSE
- RETURN

Rather heavy operation compared to just invoking on an existing instance

Performance of Events

- Table of execution methods and time

Type	Time in ms
Local Function	943
Function call in another Codeunit	909
Event Publisher no subscriber attached	538
Event Publisher with manual bound subscriber	4440
Event Publisher with static automatic subscription	9092
Event Publisher with static automatic in codeunit 80	26991
Single Instance Event Publisher with static automatic subscription	4763
Single Instance Event Publisher with static automatic in codeunit 80	4810

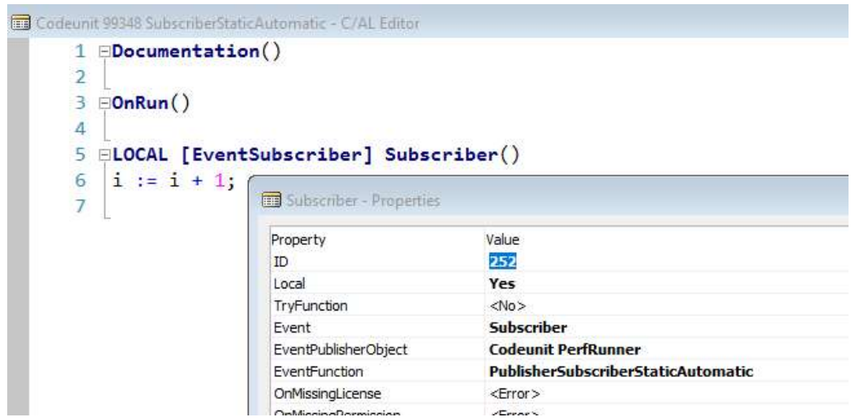
```
5  FOR i := 0 TO 1000000 DO BEGIN
6  LocalFunction;
7  END;
8
9  FOR i := 0 TO 1000000 DO BEGIN
10 OtherCodeunit.OtherCodeunitFunction;
11 END;
12
13 FOR i := 0 TO 1000000 DO BEGIN
14 PublisherNoSubscriber;
15 END;
16
17 BINDSUBSCRIPTION(SubscriberManual);
18 FOR i := 0 TO 1000000 DO BEGIN
19 PublisherSubscriberManual;
20 END;
21
22 FOR i := 0 TO 1000000 DO BEGIN
23 PublisherSubscriberStaticAutomatic;
24 END;
```

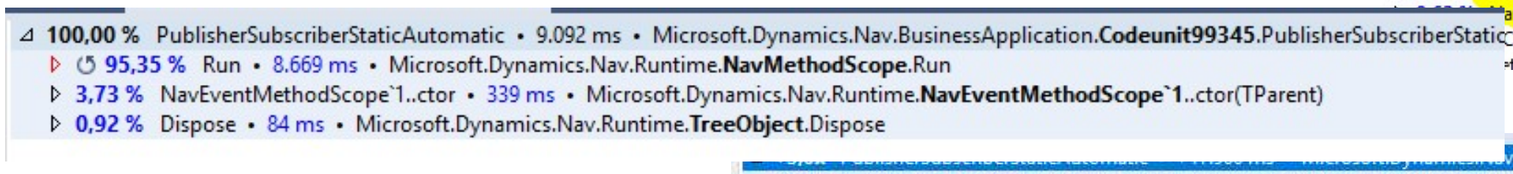
```
28 LOCAL LocalFunction()
29 i := i + 1;
30
```

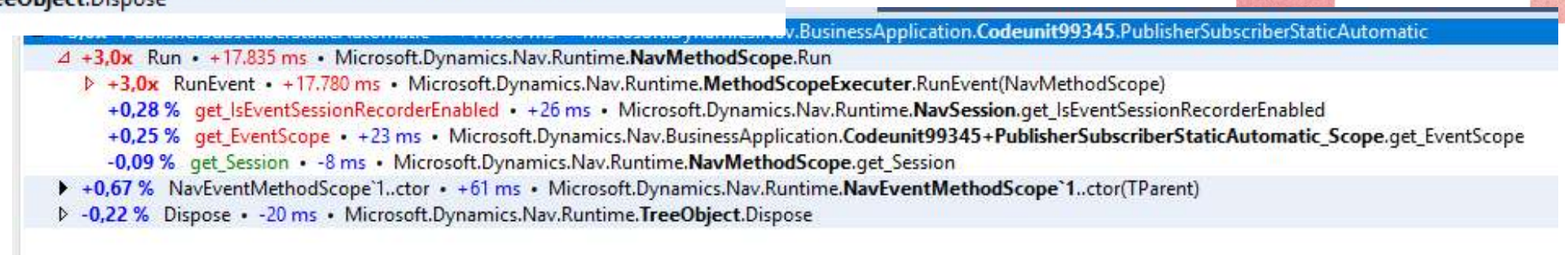
```
100,00 %  OnRun • 16.845 ms • Microsoft.Dynamics.Nav.Busines
  53,97 %  PublisherSubscriberStaticAutomatic • 9.092 ms • Mic
  26,36 %  PublisherSubscriberManual • 4.440 ms • Microsoft.D
   5,60 %  LocalFunction • 943 ms • Microsoft.Dynamics.Nav.Bus
   5,39 %  Invoke • 909 ms • Microsoft.Dynamics.Nav.Runtime.N
     5,39 %  OnInvoke • 909 ms • Microsoft.Dynamics.Nav.Busi
       5,39 %  OtherCodeunitFunction • 909 ms • Microsoft.D
         3,21 %  Run • 541 ms • Microsoft.Dynamics.Nav.Runt
         1,83 %  NavMethodScope`1..ctor • 309 ms • Microsof
         0,30 %  Dispose • 50 ms • Microsoft.Dynamics.Nav.R
   3,68 %  StmtHit • 620 ms • Microsoft.Dynamics.Nav.Runtime.
   3,20 %  PublisherNoSubscriber • 538 ms • Microsoft.Dynamics
   1,71 %  get_Target • 289 ms • Microsoft.Dynamics.Nav.Runtim
```

Performance of Events

- Size matters of subscriber codeunits







Performance of Events

Behavioral changes based on event subscriptions

- **Table Events changes behavior of SQL optimizations**
  - **MODIFYALL/DELETEALL/GlobalTriggers Database Triggers**
    - SQL update/delete statement per row in a for loop rather than one SQL statement
    - Impacts MODIFYALL/DELETEALL from bulk to single row

- **OnAfterXXX triggers**
  - Beware of uncommited changes on records

Summary Events

Be conscious about invocation model

- Static-Automatic and manual subscription

Only subscribe when you are ready to react

- Don’t waste users time if you don’t about an event right now

Size of subscriber codeunits

- If you choose static-automatic consider the amount of code you need in the subscriber codeunit

Performance types in AL

Dictionary - New AL types in extensions

NAV TECH DAYS 2018
mibuso.com

- **Key value lookups**
  - In C/AL we would use in-memory Temporary tables
    - IF KeyCacheRec.GET('Some Value') THEN
      Complete data stack execution for a relatively simple operation
  - AL Has a new Dictionary Type
  - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/methods-auto/dictionary/dictionary-data-type

- **Dictionary: Cannot be used with record instances**

Dictionary Data Type

10/17/2018 • 2 minutes to read • Contributors

Represents a collection of keys and values.

The following methods are available on instances of the Dictionary data type.

Method name	Description
Count()	Gets the number of key/value pairs contained in the Dictionary.
Keys()	Gets a collection containing the keys in the Dictionary.
Values()	Gets a collection containing the values in the Dictionary.
Get(TKey, var TValue)	Gets the value associated with the specified key.
Get(TKey)	Gets the value associated with the specified key.
Set(TKey, TValue)	Sets the value associated with the specified key.
Set(TKey, TValue, var TValue)	Sets the value associated with the specified key.
Add(TKey, TValue)	Adds the specified key and value to the dictionary.
ContainsKey(TKey)	Determines whether the Dictionary contains the specified key.
Remove(TKey)	Removes the value with the specified key from the Dictionary.

```
30
31
32
33
34
35
    - reference
36  codeunit 50102 DictionarySample
37  {
38      trigger OnRun()
39      var
40          dictionary: Dictionary of [Text, Text];
41          output: Text;
42      begin
43          dictionary.Add('Ford', 'some value Ford');
44          dictionary.Add('BMW', 'some value BMW');
45          dictionary.Add('Volvo', 'some value Volvo');
46          dictionary.Add('Mercedes', 'some value Mercedes');
47
48          if dictionary.ContainsKey('BMW') then
49              output := dictionary.Get('BMW');
50      end;
51  }
52
53
54
55
56
57
```


Dictionary

List - New AL types in extensions

- **Unbound Arrays**
  - In C/AL we would use in-memory Temporary tables
    - listRec.Value := 'Some Value';
    - listRec.INSERT();

      Complete data stack execution for a relatively simple operation
  - AL Has a new List Type

https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/methods-auto/list/list-data-type

- **Lists: Cannot be used with record instances**

List Data Type

10/17/2018 • 2 minutes to read • Contributors

Represents a strongly typed list of objects that can be accessed by index.

The following methods are available on instances of the List data type.

Method name	Description
Count()	Gets the number of elements contained in the List.
Add(T)	Adds a value to the end of the List.
AddRange(T, [T,...])	Adds the elements of the specified collection to the end of the list.
AddRange(List of [T])	Adds the elements of the specified collection to the end of the list.
Get(Integer, var T)	Gets the element at the specified index.
Get(Integer)	Gets the element at the specified index. This method will raise an error if the index is outside the valid range.
Set(Integer, T)	Sets the element at the specified index.
Set(Integer, T, var T)	Sets the element at the specified index.
Contains(T)	Determines whether an element is in the List.
IndexOf(T)	Searches for the specified value and returns the one-based index of the first occurrence within the entire List.
Insert(Integer, T)	Inserts an element into the List at the specified index.
LastIndexOf(T)	Searches for the specified value and returns the one-based index of the last occurrence within the entire List.
Remove(T)	Removes the first occurrence of a specified value from the List.

```
    - reference
48  codeunit 50103 ListSample
49  {
50      trigger OnRun()
51      var
52          list : List of [Text];
53          output: Text;
54      begin
55          list.Add('Ford');
56          list.Add('BMW');
57          list.Add('Volvo');
58          list.Add('Mercedes');
59
60          foreach output in list do
61              message(output);
62      end;
63  }
```

List

TextBuilder - New AL types in extensions

- **Concatenation of long strings**
  - In C/AL we would TEXT type and concatenate them together
    - myText := myText + 'Some Value';
    - myText += 'Some Value';

    Creating many strings adds stress to the garbage collector
  - AL Has a new TextBuilder Type

https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/methods-auto/textbuilder/textbuilder-data-type

TextBuilder Data Type

10/17/2018 • 2 minutes to read • Contributors

Represents a lighweight wrapper for the .Net implementation of StringBuilder.

The following methods are available on instances of the TextBuilder data type.

Method name	Description
Append(Text)	Appends a copy of the specified string to this TextBuilder instance.
AppendLine([Text])	Appends a copy of the specified string followed by the default line terminator to the end of the current TextBuilder object. If this parameter is omitted, only the line terminator will be appended.
Capacity([Integer])	Gets or sets the maximum number of characters that can be contained in the memory allocated by the current instance.
Clear()	Removes all characters from the current TextBuilder instance.
EnsureCapacity(Integer)	Ensures that the capacity of this TextBuilder instance is at least the specified value.
Insert(Integer, Text)	Inserts a string into this TextBuilder instance at the specified character position.
Length([Integer])	Gets or sets the length of this TextBuilder instance.
MaxCapacity()	Gets the maximum capacity of this TextBuilder instance.
Remove(Integer, Integer)	Removes the specified range of characters from this TextBuilder instance.
Replace(Text, Text)	Replaces all occurrences of a specified string in this TextBuilder instance with another specified string.
Replace(Text, Text, Integer, Integer)	Replaces, within a substring of this instance, all occurrences of a specified string in this TextBuilder instance with another specified string.
ToText()	Converts the value of this TextBuilder instance to a Text.
ToText(Integer, Integer)	Converts the value of a substring of this TextBuilder instance to a Text.

```
codeunit 50101 MyCodeunit
{
    trigger OnRun()
    var
        textBuilder: TextBuilder;
        output: Text;
    begin
        output += 'abcde';
        // output := 'abcde'
        output += 'abcde';
        // output := 'abccdabcde'
        output += 'abcde';
        // output := 'abcdeabcdeabcde'
        output += 'abcde';
        // output := 'abcdeabcdeabcdeabcde'

        textBuilder.Append('abc');
        textBuilder.Append('abc');
        textBuilder.Append('abc');
        textBuilder.Append('abc');
        output := textBuilder.ToText();

        // Only two allocations, because we had a buffer
        // to write into, that only got expanded once

        // output := 'abcabcabcabc'
    end;
}
```

The bigger an append, the more unnecesary strings are created

Wasted CPU

NAV TECH DAYS 2018
mibuso.com

TextBuilder

Settings affecting performance

JIT compilation

EnableDebugging	• Application objects are compiled with debug information. • C# files persisted between server restarts • 'Compile and Load Business Application' automatically enabled.
Debugging Allowed	• Specifies whether C/AL debugging is allowed for this Dynamics 365 Business Central Server instance. • No runtime performance impact – only raises an error if debugger is started
Compile and LoadBusinessApplication	• Compile and load the full business app on startup

Telemetry trace levels

- Diagnostics Trace Level
  - Consider setting to Warning if you do not have dedicated listener
  - Some performance impact when listening
- Enable Full C/AL function tracing
  - Captures every AL statement execution
  - Significant performance degradation

Crowd control

- **Non-Interactive Sessions Log Retain Time Interval**
- **Session Event Table Retain Time Interval**
  - Reduces the size of the Session Event table
  - OData and SOAP can add millions of records
  - Avoid using directly

- **ODataMaxPageSize**
  - Determines how many rows an OData call can return
  - Many calls required to fetch e.g. 1 million records

- **ODataMaxConcurrentSessions**
- **Maximum Concurrent Running Tasks**
  - Consider having a dedicated NST for running tasks
  - Reserve capacity for UI by setting this to a low value

Reporting

- **NetFx40_LegacySecurityPolicy**
  - True -> reports are allowed to run inside AppDomain
- **EnableApplicationDomainIsolation**
  - RDLC can run .Net code
  - Isolate in app domain for security
  - Significant performance impact

- **ReportPDFFontEmbedding**
  - Ensures that all fonts render correctly
  - Creates larger documents

Search Timeout

Specifies the time (in seconds) that a search operation on lists in the client will continue until it is terminated.

When the limit is reached, the following message displays in the client:

Searching for rows is taking too long. Try to search or filter using different criteria.

Default: 10
Dynamically Updatable: Yes

Key takeaways

- Consider dedicated NSTs
  - Tune settings to match the workload
- Reserve capacity for UI operations
- Run reports in the background
- Consider the impact of AppDomain isolation

Be a good citizen

- Many tenants on clusters
- Many tenants in parallel
- Many users in parallel

- Shared CPU Cores
- Shared Memory
- Shared SQL Capacity

NAV TECH DAYS 2018
mibuso.com

Any Questions?

NAV TECH DAYS 2018
mibuso.com

Thank you

THANK YOU