IDUG Leading the DB2 User Community since 1988 | International DB2 Users Group | #IDUG

C10 – DB2 DPF Successes: Monitoring and Tuning a hybrid IBM InfoSphere Warehouse

Pavan Kristipati
Huntington Bank

Scott Hayes
DBI Software

Session Code: C10
Thu, May 15, 2014 (9:15 AM – 10:15 AM)| Platform: DB2 for LUW

Welcome to the presentation.
Thank you for taking your time for being here.

Few success stories that are shared in this presentation could be familiar to some of you.
I would still hope that most of you would get something useful out of these lessons that we learnt.
We will have time at the end for questions.

Objectives

- Understand key performance metrics in a DB2 DPF database
- High level overview of configuring KPIs and monitoring Data Warehouse using DBI tools
- Success stories -- Tuning a Data Warehouse using DBI tools and home grown UNIX scripts
- Physical design recommendations in a DB2 DPF environment

Objectives of the presentation that were included when abstract was submitted to IDUG for consideration.

IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

Agenda

- Introductions and Background
- Database Environment and Tool set - Pavan Kristipati
- Performance Methodology in a Data Warehouse - Scott Hayes
- Challenges and Tuning Successes - Pavan Kristipati
  - High CPU Utilization
  - SAS Analytics Workload
  - Longer Backup duration
  - Uncertainty around REORGCK/ REORG
- Summary and Physical design recommendations in a DB2 DPF environment – Pavan Kristipati
- Questions

High level Agenda for this presentation.
Agenda could be divided into 2 parts.

In Part 1, we will cover database performance methodology (what to pay attention for) particularly in Data Warehouses and how to monitor using DBI tools.

In Part 2, we will cover challenges we faced and tuning success stories.

International DB2 Users Group

#IDUG

Introductions and Background

- Pavan Kristipati

IS Technical Specialist / Sr. Database Administrator

Huntington Bank

Huntington Welcome:

- Scott Hayes

President & Founder

DBI Software

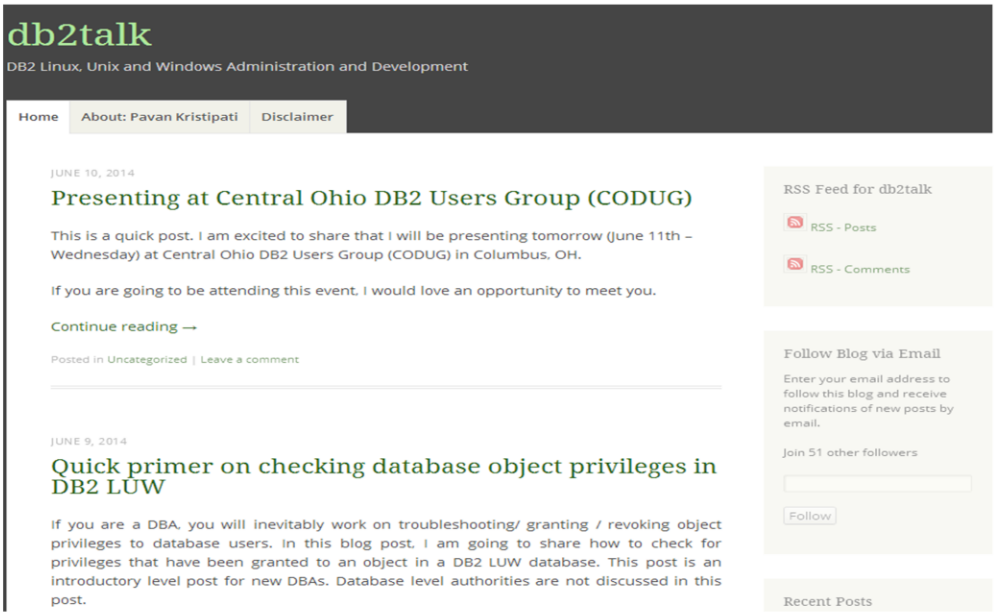
Smarter Tools for Smarter People

www.DBISoftware.com

Pavan Kristipati, an IBM certified Database Administrator is currently working as Information Systems Technical Specialist at Huntington Bank in Columbus, Ohio. Pavan could be reached at https://www.linkedin.com/in/pavankristipati or by email at pavan.kristipati@gmail.com

Scott Hayes is President and CEO of DBI Software, a frequent speaker at IDUG conferences, published author, blogger on DB2 LUW performance topics, and the host of The DB2Night Show Edutainment Webinar Series.

db2talk


db2talk
DB2 Linux, Unix and Windows Administration and Development

Home | About: Pavan Kristipati | Disclaimer

JUNE 10, 2014

Presenting at Central Ohio DB2 Users Group (CODUG)

This is a quick post. I am excited to share that I will be presenting tomorrow (June 11th – Wednesday) at Central Ohio DB2 Users Group (CODUG) in Columbus, OH.

If you are going to be attending this event, I would love an opportunity to meet you.

Continue reading →

Posted in Uncategorized | Leave a comment

JUNE 9, 2014

Quick primer on checking database object privileges in DB2 LUW

If you are a DBA, you will inevitably work on troubleshooting/ granting / revoking object privileges to database users. In this blog post, I am going to share how to check for privileges that have been granted to an object in a DB2 LUW database. This post is an introductory level post for new DBAs. Database level authorities are not discussed in this post.

RSS Feed for db2talk

RSS - Posts

RSS - Comments

Follow Blog via Email

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 51 other followers

Follow

Recent Posts

Pavan Kristipati, an IBM certified Database Administrator is currently working as Information Systems Technical Specialist at Huntington Bank in Columbus, Ohio. Pavan could be reached at https://www.linkedin.com/in/pavankristipati or by email at pavan.kristipati@gmail.com

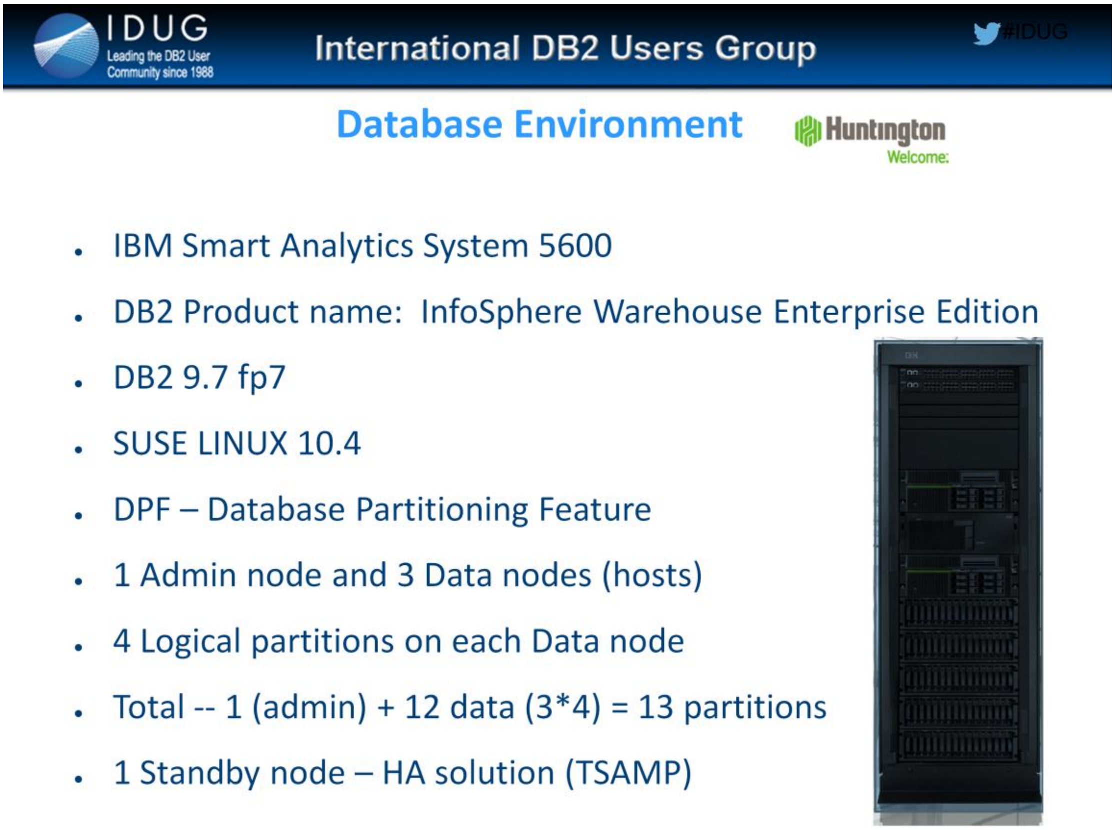
Scott Hayes is President and CEO of DBI Software, a frequent speaker at IDUG conferences, published author, blogger on DB2 LUW performance topics, and the host of The DB2Night Show Edutainment Webinar Series.

About Huntington Bank

Huntington
Welcome:

- Founded in 1866 – Close to its 150th year anniversary.
- Six States – Ohio, Michigan, Pennsylvania, Indiana, West Virginia and Kentucky.
- $59 Billion Regional Bank – Headquarters in Columbus, OH.
- Services
  - Commercial, Small Business and Consumer Banking Services
  - Mortgage Banking, Treasury Management and Foreign Exchange Services
  - Wealth and Investment, Trust and Brokerage Services
  - Other Financial Services
- 700+ branches, 1500+ ATMs, Internet and Mobile Banking
- 11000+ employees

Brief overview of Huntington Bank and its operations and footprint..



Data Warehouse Database environment at Huntington Bank on which the monitoring/tuning efforts shared in this presentation.

Details about Enterprise Data Warehouse (EDW) at Huntington Bank.

As EDW provides 360 degree view of the customer, its database performance and availability is of paramount importance to the Bank.

Huntington Bank purchased an enterprise license for DBI tools to manage performance of DB2 LUW databases. These tools are widely used in performance management of both DW and OLTP databases.

At Huntington, DBAs use a combination of DBI tools and home grown UNIX scripts to monitor and manage DB2 LUW databases.

IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Scott Hayes

dbi Smarter Tools for Smarter People www.DBISoftware.com

- About DBI
- Data Warehouse Performance Best Practices

DBI Software is Your **trusted** partner for **Breakthrough** DB2 Performance Solutions that **DELIVER INVALUABLE RESULTS** for Organizations having the most **Demanding Requirements** and **Discriminating Preferences**.

Scott Hayes began working with DB2 on distributed platforms at V1 DB2/6000. 22 years later, after having worked with 100’s of customers and clients around the world, he has learned a couple of things about what makes Data Warehouse Databases run fast.

IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

Data Warehouse Performance Best Practices 1

- I/O, I/O, It's off to spin we go…
  - The "art" and importance of parallel I/O optimization
  - Physical Design – Tablespaces & TEMPSPACES
- DPF
  - Only as fast as the worst performing partition
  - SKEW
    - DATA SKEW – how many GBs on each part?
    - PROCESSING SKEW – how many Logical Reads on each part?

While transactional databases focus on indexes and optimized synchronous I/O, for data warehouses you need to optimize parallel I/O and prefetching.

While many people pay attention to having data evenly distributed across the partitions, it is also important, maybe more important, to make sure that PROCESSING is evenly distributed across your partitions. If Parts 1, 2, 3, and 4 are working really hard, and 9, 10, 11, and 12 are sitting on their hands, then you've got an opportunity for improvement. Logical Read COST (LREADs/Transactions) is a good measurement to evaluate.

Data Warehouse Performance Best Practices 2

- Beware of Overflows
  - Overflows = Double the I/O !!!
  - Catalog Tables! SYSCAT.INDEXES, TABLES, COLUMNS…
- Monitor & Trend Read & Write I/O Latency Times
  - Prefetch/Async Read I/O is normally VERY fast, <3ms (Sync I/O <10ms)
    - Async Read % (ARP) normally 60-90%, Sync Read % (SRP) 10-20%
      - For Tablespaces with ARP>60%, assign to ASYNC BP's w/ NUMBLOCKPAGES
    - High Async Write % desirable
      - TEMPSPACE is often a bottleneck

DW DBs do plenty of I/O without the unnecessary burden of doing double the I/O due to overflows. If your ETL processes do UPDATES that increase lengths of VARCHAR columns, you can end up with a lot of overflow rows – need to REORG, or cleanup the Overflows in V10.5. DB2 catalog tables are frequent victims of high Overflows. REORG & RUNSTATS – love your catalog tables!

db2set DB2_PARALLEL_IO=*

Randomly read tablespaces can be assigned to a Random bufferpool, including small hot lookup tables.

Most tablespaces will have scans – put into an ASYNC bufferpool and don't forget NUMBLOCKPAGES set to ~3% of Pool size

Place your TEMPSPACE on storage paths separate from tablespaces & indexspaces, if you can

We have seen some customers be able to identify storage system problems by detecting abnormally high Read/Write latency times in DB2.

Data Warehouse Performance Best Practices 3

- INDEXES
  - Low Cardinality Indexes are DAMAGING to ETL & LOAD Performance
    - Use MDC Tables, Range Partition Tables, and a Few Strategic High Quality/High Cardinality Indexes
  - BIG BIG BIG BIG BIG & **COMMON** MISTAKE:
    - FAILURE TO INDEX SMALL TABLES
    - Billions of Rows Read, Burn CPU

You still need indexes in a DW! Good ones!

It's mind boggling, but we've seen some DW databases run queries twice as fast, or faster, just by putting a needed index on a 10MB table! Yes, small tables will likely remain resident in a bufferpool, but you will suck the life out of your CPUs by scanning zillions of rows in memory!

Low cardinality indexes can be very adverse to LOAD & ETL performance because of high cost of updating long rid list chains.

Up next

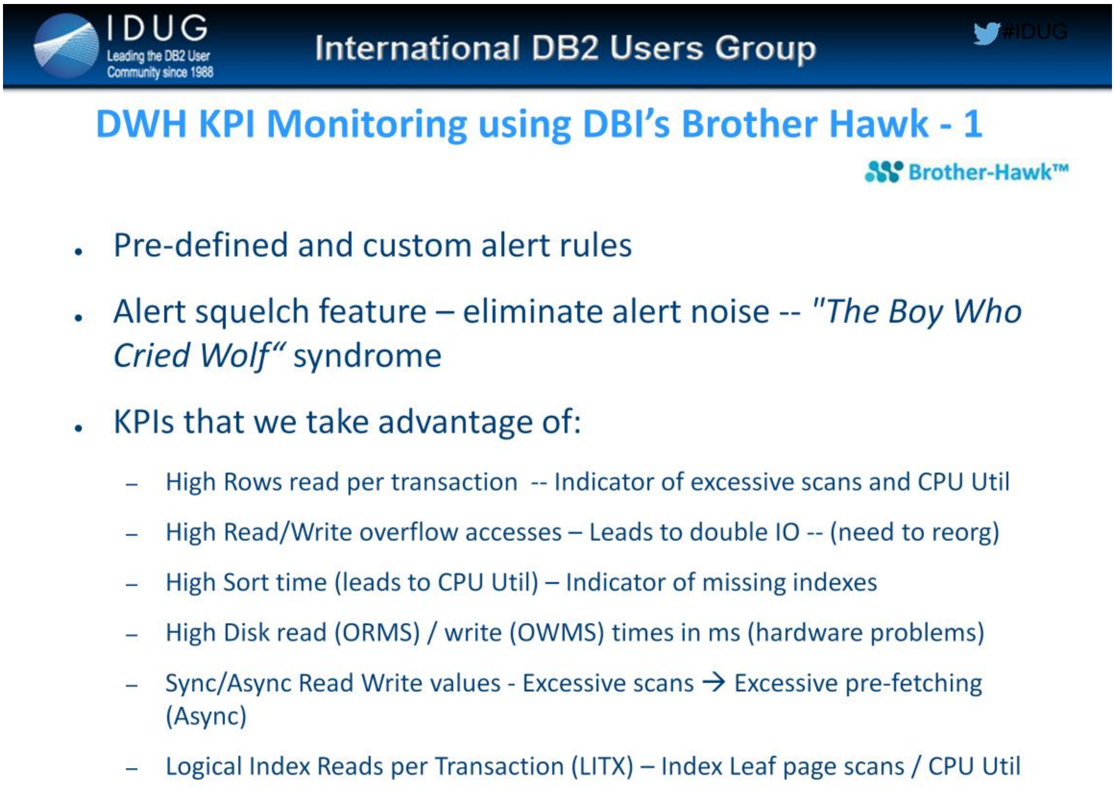
Pavan Kristipati

IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

- Data Warehouse KPI monitoring using DBI tools
- Tuning Success Stories
- Physical Design Recommendations

High level agenda for the remainder of the presentation. We will have few minutes for questions at the end.

Data Warehouse KPI monitoring using DBI tools



Out of the box, DBI's Brother Hawk has 100+ pre-defined alerts. This helps DBAs to start to keep tabs on database performance and get alerted when thresholds are crossed as soon as the toolset is installed.

Squelch feature gives option to snooze an alert until a custom defined interval (for each alert type). Repeated alerts for the same alert type will indicate on-going problem.

ORMS = Overall Disk Read Time in milli seconds

OWMS = Overall Disk Write Time in milli seconds

International DB2 Users Group

#IDUG

DWH KPI Monitoring using DBI's Brother Hawk 2

This table contains all alert rules that have been defined for Brother-Hawk. To enable or disable an alert rule, simply click the Enabled check-box.

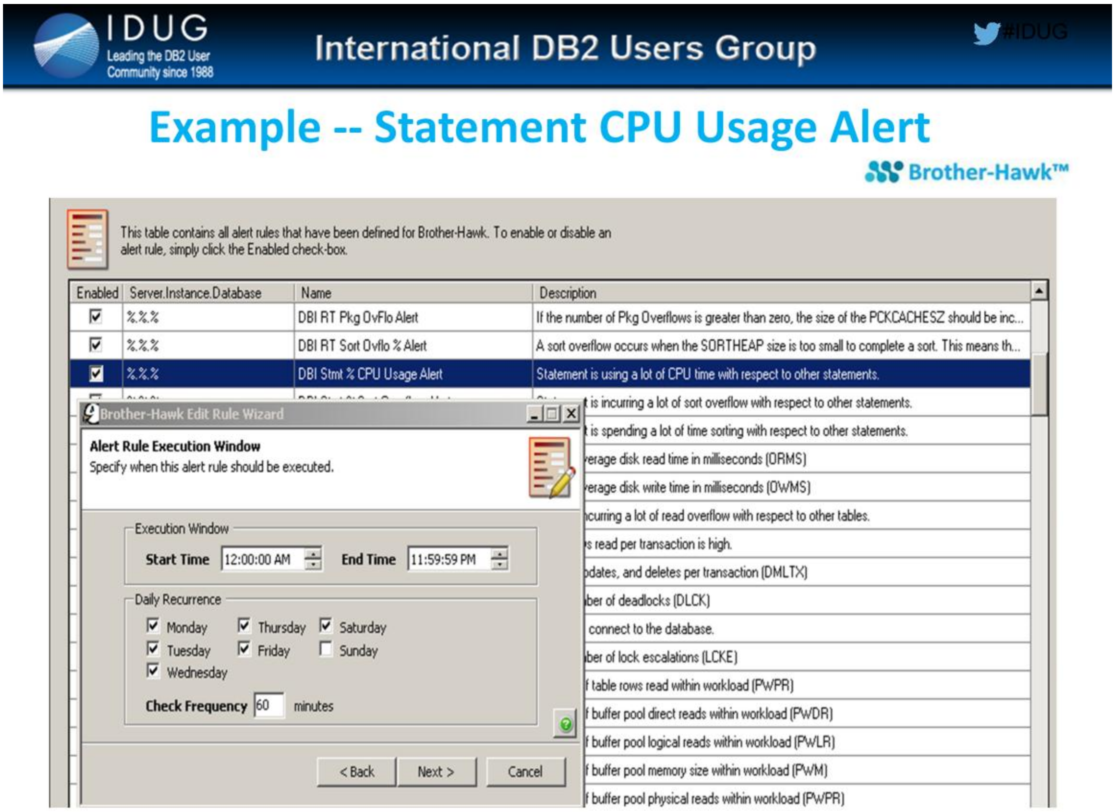
Enabled	Server.Instance.Database	Name	Description
☑	%.%.%	DBI DB Catlg OvFlo Alert	Catalog cache overflows (CATOV)
☑	%.%.%	DBI DB Locks Waiting Alert	Total number of lock waits (LCKW)
☑	%.%.%	DBI DB Log Used % Alert	Log used percentage (LUPCT)
☑	%.%.%	DBI DB Pkg OvFlo Alert	Package cache overflows (PKCOV)
☑	%.%.%	DBI Overall DB Performance Alert	Databases overall performance health is poor.
☑	%.%.%	DBI Overall Partition Performance Alert	Databases overall performance health is poor.
☑	%.%.%	DBI Part Catlg OvFlo Alert	Catalog cache overflows (CATOV)
☑	%.%.%	DBI Part Locks Waiting Alert	Total number of lock waits (LCKW)
☑	%.%.%	DBI Part Log Used % Alert	Log used percentage (LUPCT)
☑	%.%.%	DBI Part Pkg OvFlo Alert	Package cache overflows (PKCOV)
☑	%.%.%	DBI RT BP Hit % Alert	If the Bufferpool Hit Percentage is lower than 80%, performance improvements may possibly be o...
☑	%.%.%	DBI RT Catlg Hit % Alert	If the Catalog Cache Hit Ratio is less than 91%, consider increasing the CATALOGCACHE_SZ b...
☑	%.%.%	DBI RT Catlg OvFlo Alert	Catalog Cache Overflows can occur when the demand for concurrent Catalog Cache Memory e...
☑	%.%.%	DBI RT Files Closed Alert	Closing files needlessly consumes CPU time and slows down SQL performance. If you observe a...
☑	%.%.%	DBI RT Locks Waiting Alert	Connections that are in a wait state will continue to wait up to the number of LOCKTIMEOUT se...
☑	%.%.%	DBI RT Pkg OvFlo Alert	If the number of Pkg Overflows is greater than zero, the size of the PCKCACHESZ should be inc...
☑	%.%.%	DBI RT Sort Ovflo % Alert	A sort overflow occurs when the SORTHEAP size is too small to complete a sort. This means th...
☑	%.%.%	DBI Stmt % CPU Usage Alert	Statement is using a lot of CPU time with respect to other statements.
☑	%.%.%	DBI Stmt % Sort Overflow Alert	Statement is incurring a lot of sort overflow with respect to other statements.

Add | Edit | Delete | Test | Copy

Snapshot of some of the KPIs that Huntington DBAs use to manage EDW.

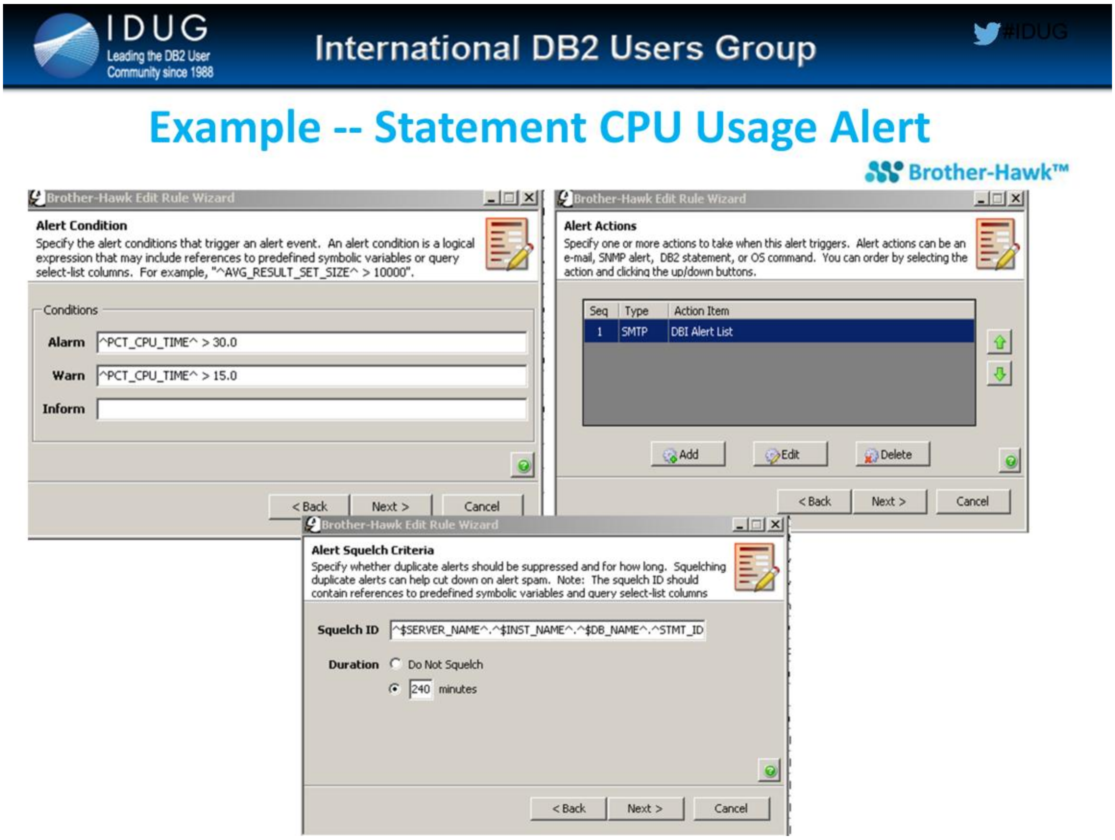
Lock waits, Log Used % would be more useful in OLTP environments.

Sort overflow, Statement CPU %, IREF, % Table overflows are few KPIs that we keep tabs on to monitor for trends.



Example of how to configure an alert in DBI's Brother Hawk.

Double click on alert of interest and schedule (start, end times and days of the week) a window to monitor.

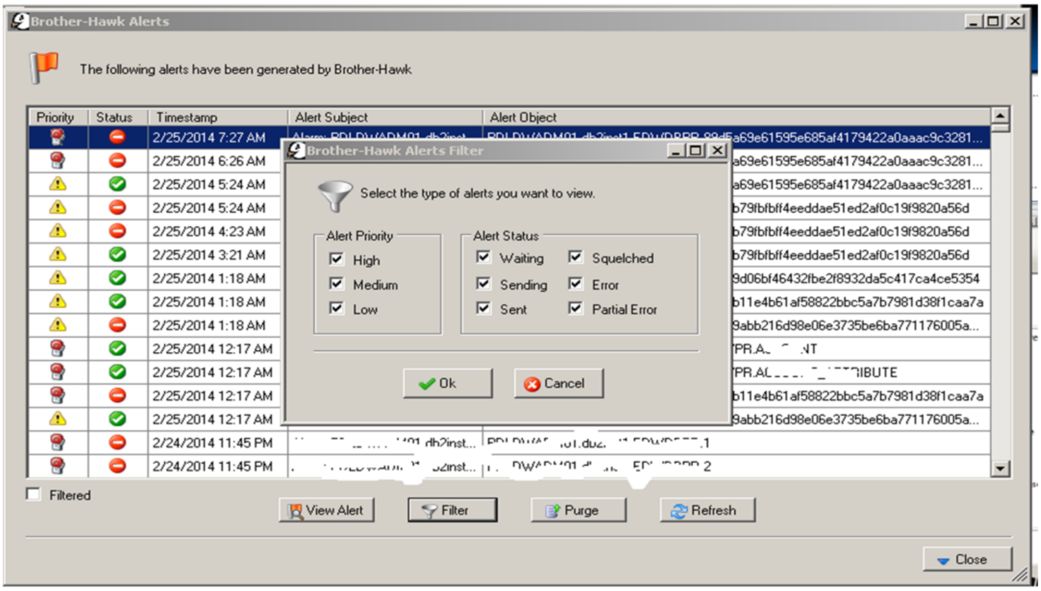


Thresholds could be picked for alarm / Warning / Information.

SMTP alerts could be sent and / or OS / Database commands could be run when a threshold is breached.

In one of the slides, there is an example of how we take an action when table overflows are more than 3%.

Hawk's Alerts Console



Brother Hawk's Alerts Console to identify High /Medium/Low priority alerts and identify any action that is needed to be taken.

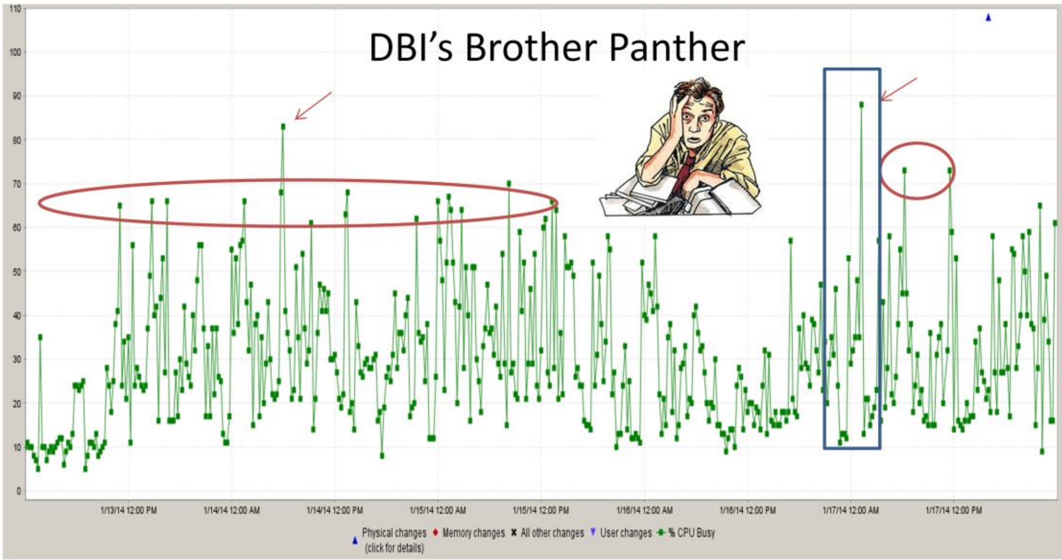
IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Tuning Success Stories

We are now at 2nd part of the presentation in which I am going to share 4 tuning success stories on Huntington’s DPF database.

IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Challenge 1 – High CPU Utilization



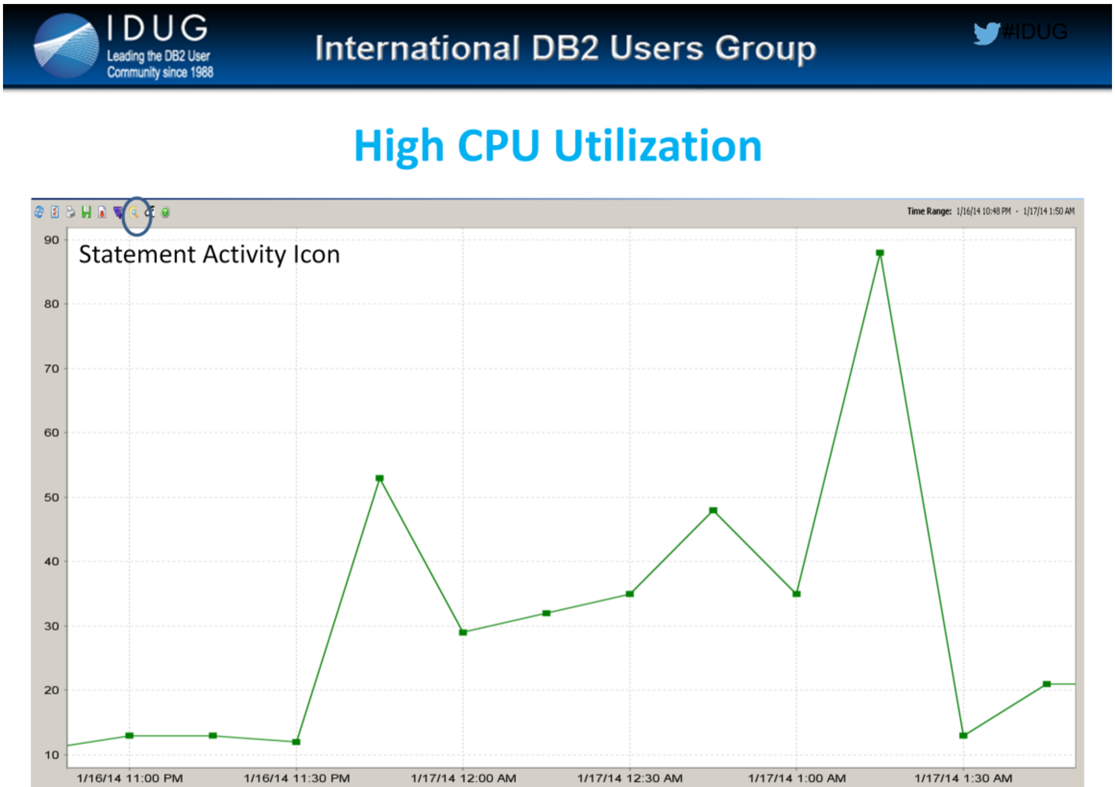
CPU Utilization over one business week

We started noticing (getting alerted for) CPU spikes during ETL cycle duration when new code was migrated to production.

Besides new code in production, there were power users who contributed to CPU spikes in production. This ad hoc activity was unpredictable unlike ETL activity.

CPU utilization was frequently ~70% and sometimes reaching 90+%.

DBI's Panther lets you focus on a particular time interval for more focused analysis.



DBI’s Brother Panther allows to zoom on an area of interest to get focused look at CPU utilization.

There is also an option to view the ‘statement activity’ during the timeframe selected.

Instead of doing this, what we did was we wanted to take a holistic look and find out what are our problematic tables over 24 hour timeframe.

IDUG
Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Tables contributing to high CPU utilization over 24 hours timeframe

Schema	Table	Size (MB)	% Space	Rows Read	% Rows Read	Rows Read/Tx	Rows Read/Sec	Rows Written	% Rows Written
EDMPR	BP THS	98,312.813	5.750%	3,866,145,028	34.180%	1,896.85	4,971.88	1,014,491	0.900%
EDMPR	I M PAR Y	27,407.984	1.600%	3,281,922,669	29.020%	3,091.52	4,748.17	21,013	0.010%
INFODELPR	E A DEM ND DEPO IT	160,615.109	9.390%	676,083,728	5.970%	331.71	869.45	1,621,453	1.440%
EDWPR	I AR _A CO NT_R LSH	23,159.125	1.350%	461,543,570	4.080%	434.77	667.74	18,098	0.010%
EDWPR	. CO NT SO RCE_ YST M_K Y	15,820.172	0.920%	338,013,904	2.980%	318.40	489.03	45,247	0.040%
AUDITPR	J B_ AT H_ RROF LOG	6,186.109	0.360%	233,281,497	2.060%	114.46	300.00	5,669,999	5.030%
EDWPR	A CO NT	5,207.375	0.300%	190,159,934	1.680%	179.13	275.12	31,210	0.020%
EDMPR	D M_ C UN	18,885.406	1.100%	170,601,844	1.500%	160.70	246.82	16,108	0.010%
EDMPR	D M_ AF Y_ CC UNT_R LS P	16,717.234	0.970%	170,742,818	1.500%	160.84	247.02	17,305	0.010%
EDMPR	F CT D _T AN DETA L	0.516	0.000%	121,213,943	1.070%	1.87	2.87	2,984,954	2.650%
EDWPR	ART	1,674.734	0.090%	106,163,903	0.930%	100.00	153.59	15,781	0.010%
EDWSTG1PR	LEMA D_EP SI DAIL A. .	32,876.438	1.920%	101,941,232	0.900%	7.39	11.34	3,402,819	3.020%

76%

80-20 Rule – 80% of CPU activity is driven by 20% of the tables

This slide shows % Rows Read and Absolute value of Rows Read at table level for top 10+ tables. Data is sorted by % Rows Read.

As you notice some values for Rows Read were in Billions of rows and that was over 24 hour period.

High # for rows read indicates presence of table scans which are usually costly especially when tablescans are done multiple times.

Our original goal was to free up CPU cycles to avoid potential costly hardware upgrades. Top contributors to CPU cycles are tables with too many tablescans.

As noticed in this slide, top 5 tables use 76% of the total rows read in a 24 hour time period.

Relation between Rows Read and CPU Cycles

- Rows Read is the number of rows that DB2 picked up from the data pages and evaluated for potential inclusion in the result set ➔ Tablescans !! → Burn CPU cycles
- Rows Read is not incremented when Index Only Access is used.

Follow Up	Stmt ID	CPU Time (sec)	% CPU Time	Rows Read	% Exec Time	IX Read Efficiency	Avg IX L Reads
	B55BB11E...	184.885648	43.216%	0	95.786%	0.000	10.604
	FC3AA463...	91.144886	21.305%	186,497,185	1.507%	9,324,859.250	0.000
	6F219C1E...	64.830702	15.154%	160,126,801	1.086%	77.116	0.000
	9D9A62C5...	52.294778	12.224%	160,158,298	1.099%	27,008.145	0.000
	5AD293D3...	22.720013	5.311%	53,306,909	0.344%	6,663,363.625	0.000
	3660A7B9...	11.938742	2.791%	26,656,973	0.178%	8,885,657.667	0.000

Avg. IX Logical Reads = 0 indicates absence of index scans (and hence presence of table scans).

High values for rows read lead to high CPU consumption mostly due to costly table scans.

International DB2 Users Group

Statements Contributing to 76% CPU over 24 hours aka. Bad SQLs

Follow Up	Stmt ID	Verb	Type	# Execs	CPU Time (sec)	% CPU Time	Rows Read	% Exec Time	IX Read Efficiency	Avg IX L Reads
	B55BB11E...	INSERT	DYNAMIC	590,803	184.885648	43.216%	0	95.786%	0.000	10.604
	FC3AA463...	SELECT	DYNAMIC	20	91.144886	21.305%	186,497,185	1.507%	9,324,859.250	0.000
	6F219C1E...	SELECT	DYNAMIC	10	64.830702	15.154%	160,126,801	1.086%	77.116	0.000
	9D9A62C5...	SELECT	DYNAMIC	11	52.294778	12.224%	160,158,298	1.099%	27,008.145	0.000
	5AD293D3...	SELECT	DYNAMIC	8	22.720013	5.311%	53,306,909	0.344%	6,663,363.625	0.000
	3660A7B9...	SELECT	DYNAMIC	3	11.938742	2.791%	26,656,973	0.178%	8,885,657.667	0.000

Follow Up	Stmt ID	Verb	Type	# Execs	CPU Time (sec)	% CPU Time	Rows Read	% Exec Time	IX Read Efficiency	Avg IX L Reads
	B4A27F58...	SELECT	DYNAMIC	6	9.468306	18.489%	7,437	3.853%	1,239.500	20,798.333
	1D734058...	SELECT	DYNAMIC	16	4.331035	8.457%	26,133,614	15.526%	1,633,350.875	0.000
	216CD564...	SELECT	DYNAMIC	9	3.540647	6.914%	13,330,919	4.547%	1,481,213.222	1,736.889
	DBE90AFD...	SELECT	DYNAMIC	9	3.539309	6.911%	13,330,920	4.898%	1,481,213.333	1,740.778
	49A9FFAF...	SELECT	DYNAMIC	11	3.308988	6.461%	16,331,425	8.579%	1,484,675.000	0.000

Follow Up	Stmt ID	Verb	Type	# Execs	CPU Time (sec)	% CPU Time	Rows Read	% Exec Time	IX Read Efficiency	Avg IX L Reads
	90F116E7...	SELECT	DYNAMIC	2	13.528253	56.690%	6,450,906	4.946%	3,225,453.000	3,469.000
	2FD9E5D2...	SELECT	DYNAMIC	7	5.054934	21.182%	1,392,354	13.284%	198,907.714	17,585.571
	E7996DBE...	SELECT	DYNAMIC	11	3.207219	13.440%	125,450	41.311%	11,404.545	71,394.091

Follow Up	Stmt ID	Verb	Type	# Execs	CPU Time (sec)	% CPU Time	Rows Read	% Exec Time	IX Read Efficiency	Avg IX L Reads
	85A0F1F4...	SELECT	DYNAMIC	5	8.342354	95.221%	26,859,055	93.237%	5,3[illegible]1,811.000	10.200

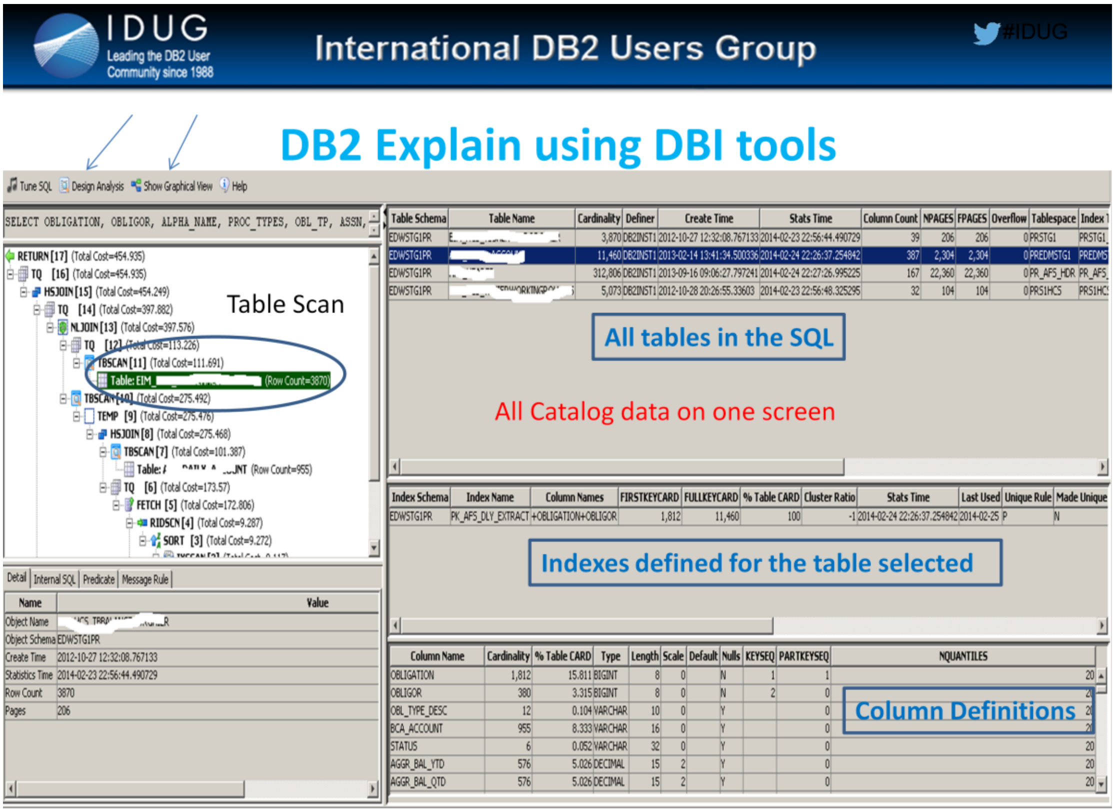
Once we identified tables that contributed to high CPU usage, next task was to identify statements that were actually running against them.

One of the most important KPIs for a statement is Index Read Efficiency (IREF) for a statement defined as the number of rows read by DB2 for potential inclusion for each one row that was selected..

IREF = (No. of rows read) / (No. of rows selected)

High IREF indicates table scans due to missing indexes (and hence the term) or possibility of leaf page scans due to bad indexes indicating a need for better quality indexes.

After running explain plans (DBI tools allow to do this), we noticed there were few missing indexes for these statements that were contributing to 76% of CPU usage.



For each of the costly statements, right click on the statement and click on “Generate Explain” to take a look at DB2’s Explain Plan.

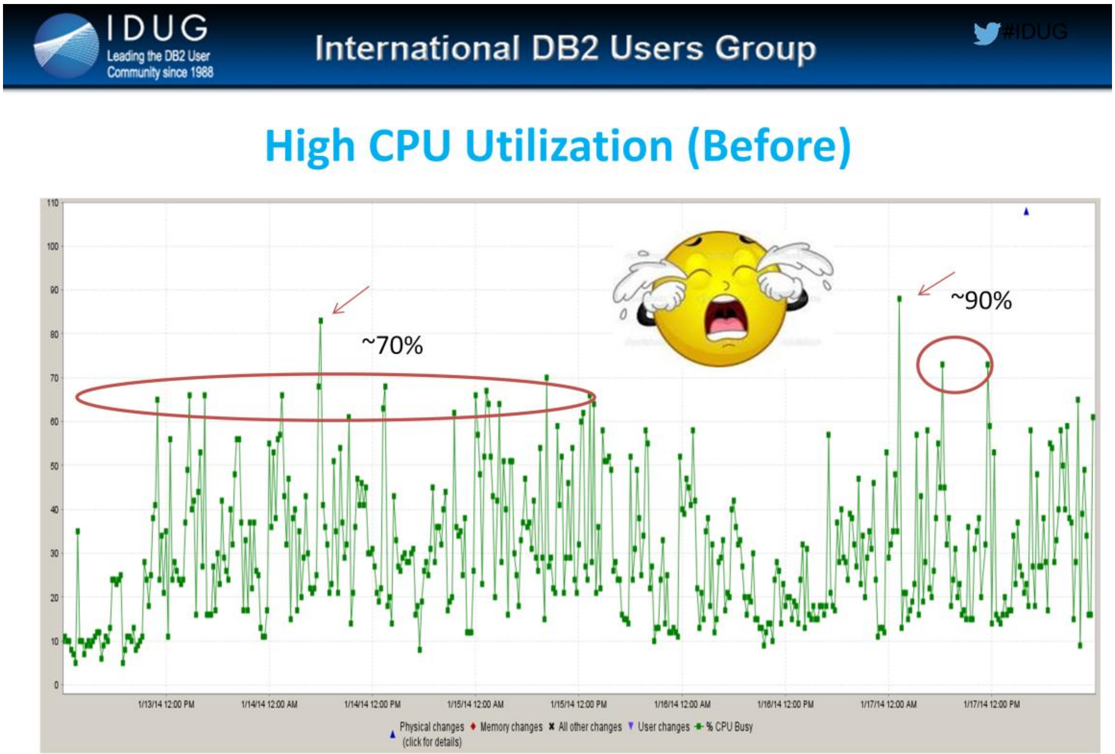
We noticed Table Scans (as guessed in previous slides) for most of the statements.

Clicking on “Design Analysis” on the top left of the screen gave access to DB2’s Advisor and its analysis as shown in next slide.

This slides shows the output of DB2 Advisor and its analysis.

As evident from the output, for one of the statements, we were able to reduce the cost of the query by ~64% by creating 4 indexes.

There is an option to either ‘save’ the output or ‘Run’ (create indexes right then and there). In production environment, we saved the report and scheduled index creation activity after obtaining approvals.



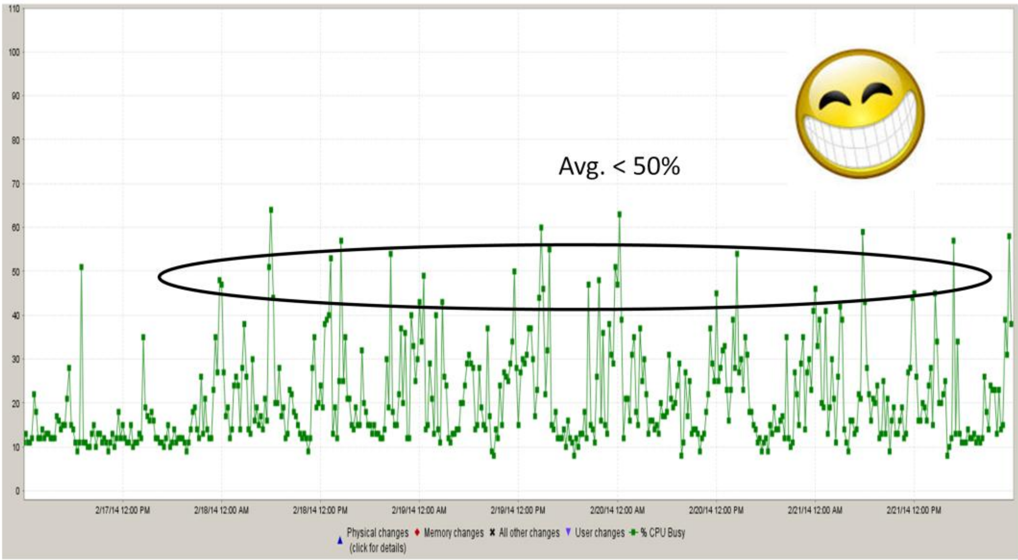
Just to recollect…This slide shows the CPU usage before tuning activity was performed.

IDUG
Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

CPU Utilization After Workload Tuning



After 4 simple steps, we noticed the Avg. CPU consumption is ~50% a big drop from earlier ~75-80% !!

Steps taken to reduce CPU Utilization

Step 1: Identify top tables driving CPU utilization

- High % Rows Read, High Rows Read / Tx

Step 2: Identify Statements driving workload to tables from step1

- High IREF (Index Read Efficiency)
- High% CPU Time

Step 3: Look at Explain Plans and Design Advisor recommendations

Step 4: Create High Quality (Good) Indexes

Result: Reduced CPU Utilization and faster Query Run time

4 steps taken to reduce CPU utilization.

While it might appear to be like finding few needles in haystack, with step 1, we tend to reduce the hay stick size to manageable level.

Once Tables with high % of rows read are identified, next step is to identify statements and analyzing cost, creating good quality indexes.

Challenge 2: SAS Analytics Workload

- Workload for SAS Retail Risk Analytics dept. – Credit / Debit Card Fraud Analysis – **Very important to finish on-time**
- Original Workload on DB2 Z/OS (Online Banking !)
  - 26+ hours runtime (SLAs not met)
  - Timeouts and Threshold breaches
- Request to move the workload to DB2 LUW (DPF) – June 2013
- 2 Tables -- 400 MM and 10 MM rows (grow fast every hour)
- SQL and workload were unknown (SAS generates SQL)
- Used DBI tools
- **Same workload now finishes in < 4 hours**

This is challenge #2 in which we had to tune a SAS workload.

IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

SAS Analytics Workload Tuning Success

User ID SAS workload	# Execs	CPU Time (sec)	CPU Cost ($)	% CPU Time	IX Read Efficiency	IX L Reads	Avg IX L Reads	Sort Time (ms)
SPEDWD1.dsadm. sen.	11,697,310	1,785.957562	$178.5958	86.766%	11,624.146	28,984,204	2.478	4,276
SASRRPD.zenadv.	404,901	137.465412	$13.7465	6.678%	0.674	7,671,999	18.948	703
HB09518.HB09518....	54	58.453666	$5.8454	2.840%	534,086.222	408,109	7,557.574	15,358
DB2INST1.db2inst1. adm01.	6,152	29.476224	$2.9476	1.432%	11.281	203,146	33.021	18,778
HB98894.HB98894.CNU252BNY8.	40	15.904307	$1.5904	0.773%	8,403.704	370	9.250	75
HB09846.HB09846.CNU2509WVL.	29	14.869992	$1.4870	0.722%	1,649,261.000	348,769	12,026.517	5,361
DB2INST1.db2inst1. data01.	7,140	5.241251	$0.5241	0.255%	0.000	0	0.000	0
DB2INST1.db2inst1. data02.	6,195	3.082949	$0.3083	0.150%	0.000	0	0.000	0
DB2INST1.db2inst1. ata03.	8,916	3.061884	$0.3062	0.149%	0.000	0	0.000	0
SASRRPD.hb06910.	86	2.922350	$0.2922	0.142%	794.738	159	1.849	0
HB34939.HB34939.2UA0401809.	1	1.409962	$0.1410	0.068%	12,085.608	0	0.000	0
DB2INST1.DB2INST1.CNU235B8VJ.	25	0.494639	$0.0495	0.024%	1.307	2	0.080	5
MSDB2PR.MSDB2PR.ODBC4DB2Linux.	5	0.001413	$0.0001	0.000%	15.000	18	3.600	0
DB2INST1.DB2INST1.DBI_BHD_...	60	0.008202	$0.0008	0.000%	0.000	0	0.000	0

DBI's workload

This slide shows Brother Panther's Work Load analysis by user over 24 hours.

SASRRPD (2nd from top) is the USER that runs SAS work load.

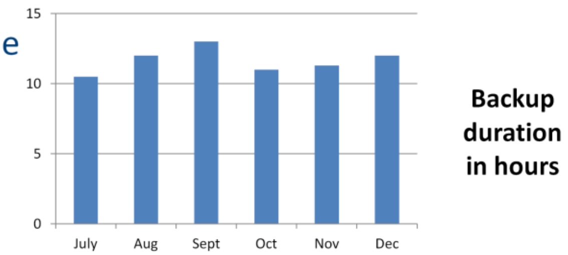
As noticed, after tuning the workload, the % CPU time taken by SASRRPD user is < 7% with Index Read Efficiency < 1 !

This is a result of tuning effort that could be easily shared with management !

While we could potentially capture workload using native db2 tools like event monitors at the command line, it is very difficult to gauge the relative cost of the workload at user ID or application ID level.

Challenge 3: Longer Database Backup Runtimes

- Full nightly backups of DPF database (~3.5 TB) to TSM (Incremental backups being tested)
- Earlier – Backup duration was 10 to 12 hours (unacceptable)
- Restore time ~ Backup time



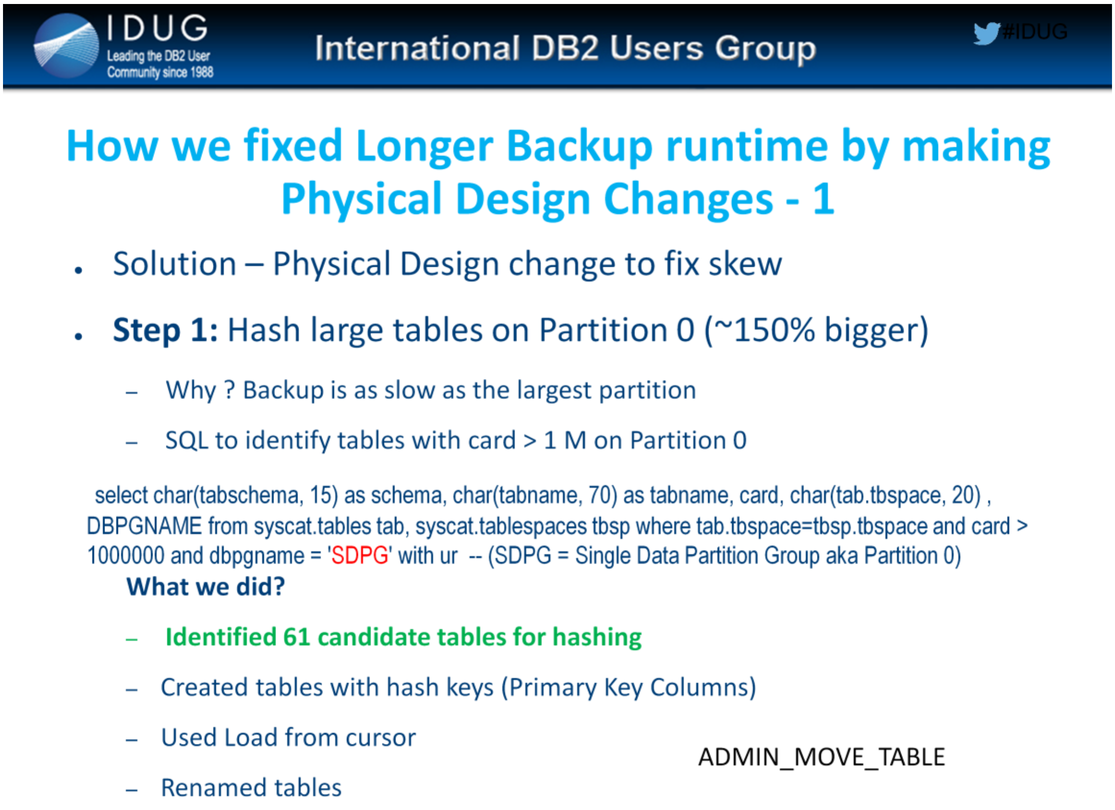
- Backup was slowing down ETL workload – SLAs were being missed

Before incremental backups were implemented, backup policy was to take full online backups nightly.
Before switching to combination of full + incremental backups, we wanted to identify opportunities to tune full online backup.
Original backup runtimes were 10-12 hours and were unacceptable.

Root Cause for Long Backup runtimes

- Observation – Backup was slower on few partitions compared to others
- Root Cause(s) in database layer
  - Partitions were heavily skewed in size
    - Faster growth of some of the tables than anticipated -> data skew
    - Partition 0 size -- ~150% bigger than other partitions
    - Data skew at table level -- Hash keys were picked to collocate tables
  - Few tablespaces were very large (~1000x) compared to others

Backup progress for each partition could be monitored using DB2's "list utilities show detail" command.



An alternative to using load from cursor is to use ADMIN_MOVE_TABLE stored procedure

IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group #IDUG

How we fixed Longer Backup runtime by making Physical Design Changes - 2

- **Step 2:** Move large tables into their own tablespaces
  - Why? Very large tablespaces (compared to others) are bottleneck for backup)
  - SQL to identify candidate tablespaces in notes
    - Prints tablespace, size, table_count in desc

What we did?

- Created new tablespaces and tables
- Used load from cursor or ADMIN_MOVE_TABLE to move tables into new tablespaces

```
with
  temp1 as
   (select TBSP_NAME as tbsp_name, sum(TBSP_USED_SIZE_KB/1024) as
    tbsp_size
      from sysibmadm.tbsp_utilization
--         WHERE DBPGNAME in ('PDPG', 'ALLPG')
group by TBSP_NAME) ,
  temp2 as
   (select t1.tbspace as tbsp_name, count(*) as tbl_cnt
         from syscat.tables t1, syscat.tablespaces t2
               where t1.type='T‘ and t1.tbspace=t2.tbspace
--                and t2.ngname in ('ALLPG', 'PDPG')
                     and t1.tbspace <> 'DWEDEFAULTCONTROL'
               group by t1.tbspace
                having count(*) > 1)

   select char(temp1.TBSP_NAME,20),  temp1.tbsp_size, temp2.tbl_cnt
     from temp1, temp2 where temp1.tbsp_name=temp2.tbsp_name
order by temp1.tbsp_size desc fetch first 100 rows only with ur
;
```

IDUG
Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

How we fixed Longer Backup runtime by making Physical Design Changes - 3

- **Step 3:** Reduce High Water Mark (HWM) wherever possible
  - Why? Backup runs until High Water Mark (HWM)
  - Identify top candidates to lower HWM
    - SQL 1 in notes – Identifies tablespaces with > 1 GB HWM reduction opportunity
    - SQL2 in notes (if curious) – How much HWM (MB) would be reduced in each container (DB2 automatically calculates and reduces)
  - How to reduce HWM?
    - ALTER TABLESPACE $tbsp LOWER HIGH WATER MARK
  - **200 GB reduction in HWM**

SQL1 -- Identify tablespaces with > 1 GB HWM reduction opportunity:

```
SELECT CHAR(TBSP_NAME,18) AS TBSP_NAME,
  SUM((TBSP_PAGE_TOP-TBSP_USED_PAGES)*TBSP_PAGE_SIZE/1024/1024/1024) as
    TO_BE_REDUCED_SPACE_GB
  FROM SYSIBMADM.TBSP_UTILIZATION
  GROUP BY TBSP_NAME
  HAVING SUM((TBSP_PAGE_TOP-TBSP_USED_PAGES)*TBSP_PAGE_SIZE/1024/1024/1024) > 1
   ORDER BY 2 DESC
   FETCH FIRST 100 ROWS ONLY WITH UR;
```

SQL 2 – Identify how much space (MB) would be reduced from each container:
(if you are curious to find out how much HWM in MB would be reduced by DB2)

```
SELECT
    char(TBSP_UTIL.TBSP_NAME,20) AS TABLESPACE,
    char(CONTAINER_NAME,50) as CONTAINER,
    TBSP_UTIL.DBPARTITIONNUM as PARTITION,
    TBSP_PAGE_TOP                as HWM,
    (TBSP_FREE_SIZE_KB/1024)         as FREE_MB,
    (TBSP_PAGE_TOP-TBSP_USED_PAGES)*TBSP_PAGE_SIZE/1024/1024 as
     HWM_REDUCTION_OPPORTUNITY_MB

 from SYSIBMADM.TBSP_UTILIZATION TBSP_UTIL,
     SYSIBMADM.CONTAINER_UTILIZATION CONT_UTIL
WHERE TBSP_UTIL.TBSP_NAME=CONT_UTIL.TBSP_NAME AND
     TBSP_UTIL.DBPARTITIONNUM=CONT_UTIL.DBPARTITIONNUM
AND
TBSP_UTIL.TBSP_NAME='$TBSP' order by TBSP_UTIL.DBPARTITIONNUM with ur;
```

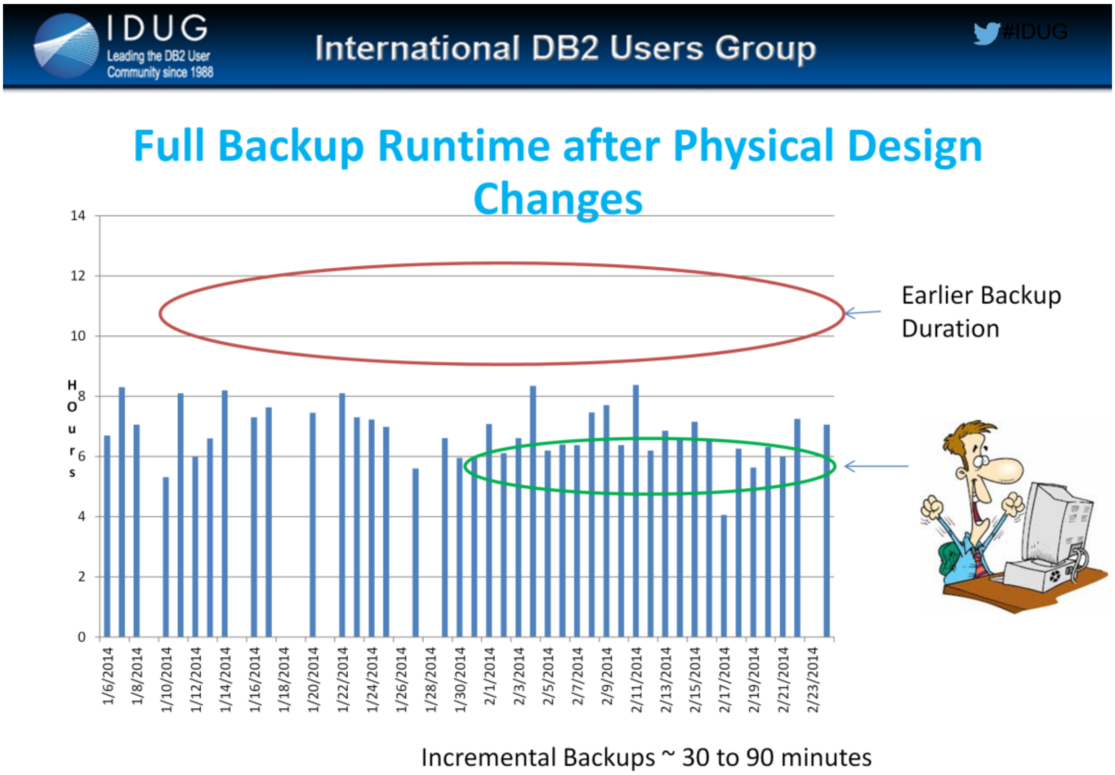
How we fixed Longer Backup runtime by making Physical Design Changes - 4

- **Step 4:** Fix skew at table level
  - Custom script (in crontab) records data skew info for all tables
  - Stored procedure “estimate_existing_data_skew”
  - Stored procedure “estimate_new_data_skew”
  - New Hash keys (PK) for 2 large tables
  - In notes
    - Info Center URL to download Stored Procedures
    - Usage with an example

URL: http://www.ibm.com/developerworks/data/library/techarticle/dm-1005partitioningkeys/

Example from IBM Info Center:

```
$db2 "set serveroutput on"
$ db2 "CALL estimate_existing_data_skew('TPCD', 'SUPPLIER', 25)"
CALL estimate_existing_data_skew('TPCD', 'SUPPLIER', 25)
Return Status = 0
DATA SKEW ESTIMATION REPORT FOR TABLE: TPCD.SUPPLIER
Accuracy is based on 25% sample of data
------------------------------------------------------------------------
TPCD.SUPPLIER
Estimated total number of records in the table: : 19,994,960
Estimated average number of records per partition : 2,499,368
Row count at partition 1 : 1,599,376 (Skew: -36.00%)
Row count at partition 2 : 2,402,472 (Skew: 3.87%)
Row count at partition 3 : 4,001,716 (Skew: 60.10%)
Row count at partition 4 : 2,394,468 (Skew: -4.19%)
Row count at partition 5 : 1,600,028 (Skew: -35.98%)
Row count at partition 6 : 1,599,296 (Skew: -36.01%)
Row count at partition 7 : 2,397,116 (Skew: -4.09%)
Row count at partition 8 : 4,000,488 (Skew: 60.05%)
Number of partitions: 8 (1, 2, 3, 4, 5, 6, 7, 8)
------------------------------------------------------------------------
Total execution time: 20 seconds
```



Earlier backup duration was in the range of 9 to 13 hours.

After physical design changes, backup runtime is averaging between 5 to 6 hours.

To save disk space on TSM and to reduce impact of backups on workload, combination of Full+Incremental backups were implemented.

Because of the physical design changes that were made to help full backups, Incremental backups mostly run in less than 1 hour and average between 30 to 90 minutes on any given day depending on the amount of data that needs to be backed up.

IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

But.. did these physical design changes negatively impact the SQL Workload?

A major question that is left over is if the physical design changes that were done impacted SQL workload in a negative way?

DBI's Panther answers that question.

Workload Comparison
(Before and After Physical Design Changes)

Before: 12/16/13 12:00 AM to 12/20/13 11:59 PM
After: 1/13/14 12:00 AM to 1/17/14 11:59 PM

Last Refresh: 2/25/14 2:10 PM
Rows: 9112

Follow Up	Stmt ID	Avg CPU Time (s) Improved	Avg CPU Time (s) Before	Avg CPU Time (s) After	Avg CPU Time (s) Change	% CPU Time After	Avg Exec Time (s) Improved	Avg Exec Time (s) Before	Avg Exec Time (s) After	Avg Exec Time (s) Change	% Exec Time After	IX Read Efficiency Improved	IX Read Efficiency Before	IX Read Efficiency After	IX Read Efficiency Change	Avg IX L Reads Improve
	65368757...	Y	5.330373	0.000073	-5.330300	0.000%	Y	6.392949	0.000078	-6.392871	0.000%	Y	45,453.063	0.000	-45,453.063	Y
	A2409341...	Y	5.497087	3.963057	-1.534030	0.084%	Y	9.009444	6.329062	-2.680382	0.009%	N	4,194.825	4,893,524.714	4,889,329.889	U
	4EB642AA...	Y	4.314527	2.858869	-1.455658	0.043%	N	6.848947	13.763687	6.914740	0.014%	Y	2,163,901.846	1,420,371.600	-743,530.246	Y
	A76C8AF5...	Y	1.176815	0.009955	-1.166860	0.000%	N	8.685561	12.668864	3.983303	0.015%	N	41.775	1,024.833	983.058	Y
	2FE77159...	Y	0.917433	0.000238	-0.917195	0.000%	Y	1.549721	0.015445	-1.534276	0.000%	Y	1,029,116.000	0.000	-1,029,116...	N
	FCE1B97C...	Y	0.862766	0.000032	-0.862734	0.000%	Y	14.181679	0.000036	-14.181643	0.000%	Y	0.083	0.000	-0.083	U
	065312EE...	Y	0.825831	0.000040	-0.825791	0.000%	Y	2.840681	0.000096	-2.840585	0.000%	Y	1,227,588.500	0.000	-1,227,588...	Y
	1BB05023...	Y	2.439344	1.636408	-0.802936	0.055%	N	4.395555	4.526009	0.130454	0.010%	Y	2,968.705	1,304.298	-1,664.407	Y
	35CD88EA...	Y	0.767251	0.000053	-0.767198	0.000%	Y	9.118302	0.000074	-9.118228	0.000%	Y	142.545	0.000	-142.545	Y
	A03E54E7...	Y	0.846003	0.086681	-0.759322	0.002%	Y	18.644749	7.612915	-11.031834	0.011%	N	0.153	5,085.143	5,084.990	U
	FEEA3A35...	Y	0.729046	0.000188	-0.728858	0.000%	Y	0.885512	0.004355	-0.881157	0.000%	Y	1,141,368.000	0.000	-1,141,368...	Y
	6FB6EC27...	Y	0.763215	0.051303	-0.711912	0.001%	Y	23.095594	7.209786	-15.885808	0.013%	N	0.154	28,612.333	28,612.179	U
	E6BA707E...	Y	0.707421	0.000045	-0.707376	0.000%	Y	8.637303	0.000050	-8.637253	0.000%	Y	237.667	0.000	-237.667	Y
	6A8AA127...	Y	0.703603	0.000038	-0.703565	0.000%	Y	8.292555	0.000076	-8.292479	0.000%	Y	9.000	0.000	-9.000	Y
	BC440C7B...	Y	0.965968	0.270833	-0.695135	0.010%	Y	7.134208	2.366273	-4.767935	0.006%	Y	6,463,438.773	2,357,884.583	-4,105,554...	U
	C964589B...	Y	0.661686	0.000033	-0.661653	0.000%	Y	8.832503	0.000038	-8.832465	0.000%	Y	541.615	0.000	-541.615	Y
	50FDF3D2...	Y	0.657285	0.000033	-0.657252	0.000%	Y	8.599815	0.000038	-8.599777	0.000%	Y	541.615	0.000	-541.615	Y
	C94D780F...	Y	0.650217	0.000040	-0.650177	0.000%	Y	8.002826	0.000049	-8.002777	0.000%	Y	216.615	0.000	-216.615	Y
	21B4CD9A...	Y	0.650157	0.000032	-0.650125	0.000%	Y	4.642008	0.000037	-4.641971	0.000%	Y	15,120.636	0.000	-15,120.636	Y
	EA94E261...	Y	0.721421	0.115442	-0.605979	0.004%	N	1.875428	2.149862	0.274434	0.004%	Y	1,930,660.750	215,161.000	-1,715,499...	Y
	A00B0804...	Y	2.483330	1.885399	-0.597931	0.046%	N	4.796761	5.281201	0.484440	0.008%	N	1,835.435	2,225,381.125	2,223,545.690	N
	4C73106B...	Y	3.218855	2.690391	-0.528464	0.122%	N	7.083161	13.276764	6.193603	0.040%	Y	1,882,753.000	387,467.915	-1,495,285...	N
	FD2ECDFF...	Y	0.637097	0.115451	-0.521646	0.003%	Y	10.341673	10.129154	-0.212519	0.016%	N	0.077	19,350.500	19,350.423	U
	B8085D91...	Y	0.700146	0.197952	-0.502194	0.005%	Y	5.573665	2.222859	-3.350806	0.004%	Y	780,407.235	587,077.625	-193,329.610	Y
	C4116F6D...	Y	0.500660	0.000027	-0.500633	0.000%	Y	20.082552	0.000031	-20.082521	0.000%	Y	0.383	0.000	-0.383	U
	04B5E759...	Y	0.489023	0.000147	-0.488876	0.000%	Y	0.674693	0.001099	-0.673594	0.000%	Y	907,162.824	0.000	-907,162.824	Y
	FD45E984...	Y	0.492404	0.008796	-0.483608	0.000%	Y	3.909104	0.520449	-3.388655	0.001%	N	0.076	91.889	91.813	U
	9719064A...	Y	1.602582	1.151037	-0.451545	0.052%	Y	7.265112	4.812770	-2.452342	0.014%	N	329.102	567.657	238.555	N
	67ED12C6...	Y	0.423344	0.015816	-0.407528	0.001%	Y	1.201140	0.215977	-0.985163	0.001%	N	28,521.316	29,283.500	762.184	Y

One exciting feature in DBI's Panther is that it could do workload comparison between 2 time intervals.

In this slide, the workload comparison in done between before and after Physical design changes.

24 hour time interval was taken as comparison and as you notice good physical design changes mostly help SQL queries to run faster and DBI's Panther provides the proof for this !!

IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group #IDUG

Challenge 4: Uncertainty around REORGCHK / REORG

- Quick Background
  - Data Warehouse -- Updates on ~20% of data

    Example: COLUMNN – VARCHAR(100)

    -- Value “PAVAN KRISTIPATI” updated to “PAVAN KUMAR KRISTIPATI”

    -- Row (wider) doesn’t fit into the same page

    -- DB2 relocates row to new page;

    -- Pointer in the original location to the new location
  - Result – Double IO !!

    How to fix this?
    - DB2 attempts to read the row from its original location
    - Finds a pointer instead
    - DB2 now reads the row from its new location

    REORG

On a given day, about 20% of our Data Warehouse was being “changed” (updates mostly).

While REORG operations could avoid costly double IO, fundamental question that remains is which tables to REORG?

IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

Uncertainty around REORGCHK / REORG

- REORGCHK
  - Utility to indicate when / what to REORG
  - Formulas based on Table Cardinality, Overflows etc.
- Limitations with REORGCHK's analysis
- Observation 1:
  - Analysis based on data sampling of the **entire table**
  - Criteria used does not work well with large tables
  - Example: In Large tables
    - Older data – Well Organized (No updates after last Reorg)
    - Newer data – Fragmented (due to Updates)
    - Interest in newer data – Accessed more frequently (Could be a small %)
  - Could miss marking large tables to be reorg'd.

REORGCHK is a native IBM DB2's utility to help to identify which tables to REORG.

IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Uncertainty around REORGCHK / REORG

- Observation 2:
  - Uses one of the two options – Current stats or Update stats
  - Current stats: 24 hours old (AUTO MAINT off) – stats probably no longer valid
  - Update stats: New stats alter SQL access plans for Dynamic SQL

- Observation 3:
  - Formula F1
    - based on how big (card) the table is and not how active the table is
    - # Overflows do not get reset after reorg (needs database deactivation !)
  - F1: 100 * overflows / card < 5 (If F1 > 5, mark for reorg)

DBI's tools give a huge advantage in knowing which tables need to be reorg'd in that the tools help in identifying those tables that were read/accessed in the timeframe of interest.

Taking the traditional approach (reorgchk) does not give this option. Formulae are based on the row size of the table and not how "active" (rows read) the table really is.

SQL to find overflows for tables. Please note that ROWS_READ, ROWS_WRITTEN, OVERFLOW_ACCESSESS values are since database activation time.

```
db2 "select char(tabschema, 20), char(tabname, 40),
  sum(overflow_accesses) as
  total_overflow_accesses, sum(rows_read) as
  total_rows_read, sum(rows_written) as
  total_rows_written from sysibmadm.snaptab group
```

by tabschema, tabname order by total_overflow_accesses desc , total_rows_read desc fetch first 20 rows only with ur“

Uncertainty around REORGCHK / REORG

Alternative: DBI's Approach:

- Based on # **overflows and rows read in a** specific timeframe
- % Overflow = 100*overflows/(Rows Read + 1)

P.S.: SQL in the notes to find out overflows from "database activation time"

DBI's tools give a huge advantage in knowing which tables need to be reorg'd in that the tools help in identifying those tables that were read/accessed in the timeframe of interest.

Taking the traditional approach (reorgchk) does not give this option. Formulae are based on the row size of the table and not how "active" (rows read) the table really is.

SQL to find overflows for tables. Please note that ROWS_READ, ROWS_WRITTEN, OVERFLOW_ACCESSESS values are since database activation time.

db2 "select char(tabschema, 20), char(tabname, 40), sum(overflow_accesses) as total_overflow_accesses, sum(rows_read) as total_rows_read, sum(rows_written) as total_rows_written from sysibmadm.snaptab group by tabschema, tabname order by total_overflow_accesses desc , total_rows_read desc fetch first 20 rows only with ur“

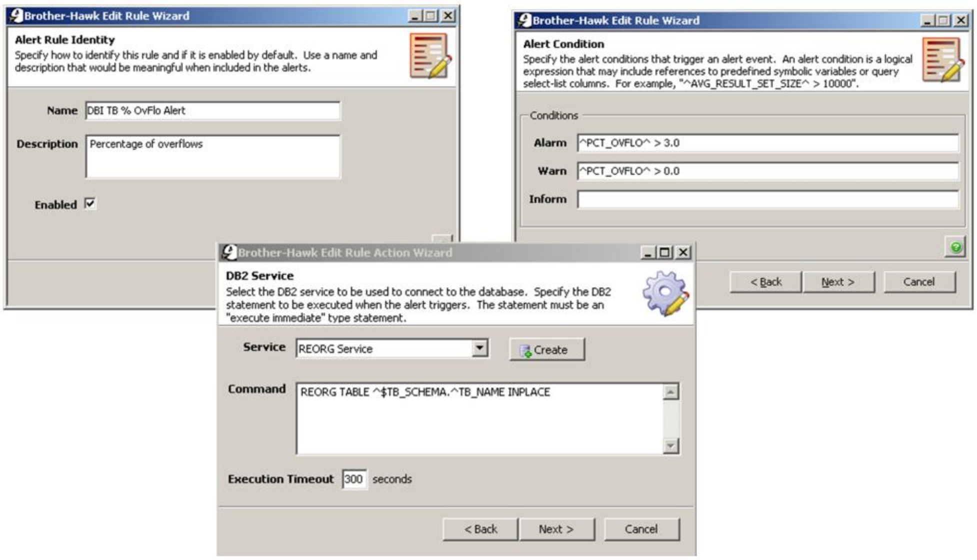
Candidate tables for REORG

- Tables with % overflows > 3%
- An effective way to avoid double IO
- Pay attention to overflows in “Catalog” tables

Schema	Table	% OvFlo	% Read OvFlo
EDMPR	[illegible]	12.168%	12.168%
SYSIBM	SYSCOLUMNS	6.960%	6.960%
EDMPR	[illegible] ON	5.390%	5.390%
SYSIBM	SYSINDEXES	4.108%	4.142%
EDMPR	D[illegible]_RE[illegible]	1.016%	1.016%
EDMPR	DI[illegible]IONS	0.847%	0.854%
EDMPR	D[illegible]_PROPE[illegible]Y	0.584%	0.584%
EDMPR	FA[illegible]AIL	0.521%	0.521%
EDWSTG1PR	AF[illegible]	0.323%	0.323%
EDMPR	I[illegible]_CONSUMER_[illegible]G	0.230%	0.230%
EDMPR	F[illegible]L	0.149%	0.298%
AUDITPR	[illegible]	0.062%	0.062%
AUDITPR	[illegible]	0.059%	0.059%
SYSIBM	SYSTABLES	0.052%	0.052%
SYSIBM	SYSKEYCOLUSE	0.025%	0.025%
CODEPR	[illegible]	0.005%	0.005%
SYSIBM	SYSDATATYPES	0.005%	0.005%

Often system catalog tables show up in the list of top tables with overflows.

REORG Automation using DBI's Brother-Hawk



Using DBI's Brother Hawk, we reorg tables based on % table overflow values.

If % table overflow > 3, table is reorg'd.

DB2 DPF – Top Recommendations at Database level

- Partition groups – SDPG (0), PDPG (1-12) and ALLPG (0-12)

Table Card < 1 M	SDPG
Table Card > 1 M	PDPG
MQTs	ALLPG

- Lookup tables -- heavily used code values are integers (rather than character) -- Faster
- Goal -- Equal workload on all data partitions
  - High Cardinality columns (Primary Key) as hash key
  - Collocate if absolutely necessary for SQL performance
- Goal – Faster backup and recovery times
  - Avoid very big tablespaces – large table in its own tablespace
  - Avoid db partition level skew (best practice for all over-all database performance)

DB2 DPF – Top Recommendations at Database level

- Database Performance
  - Create Good (High Cardinality) Indexes
  - Avoid Bad (Low Cardinality) Indexes – Drop them if they exist. (Ember Crooks in DB2Nightshow on April 18th 2014)
  - Take advantage of MQTs and MDCs
  - Drop duplicate indexes – Index on col1 and Index on col1, col2
  - Optimize Async / Sync IO (For DW, aim for SRP > 25%)
- Use Range Partitioned tables for large tables. Helps in implementing data retention/archival.
- Use Compression on large tables
- Review DB2 Advisor's output for quality of indexes

IDUG
Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Another talk at IDUG NA 2014
C08 - DB2 DPF Tips and Tricks: UNIX, SQL scripts and more for Lazy DBAs

Pavan Kristipati
Huntington Bank

Rao Balaga
Huntington Bank

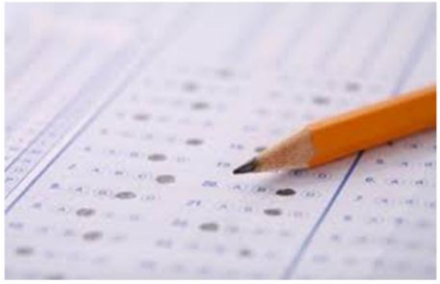
Session Code: C08
Wed, May 14, 2014 (02:15 PM - 03:15 PM) | Platform: DB2 for LUW

Thank you for attending.

Any Questions?

Pavan Kristipati
Huntington Bank
pavan.kristipati@gmail.com
www.db2talk.com

Scott Hayes
Database Brothers Inc.
shayes@dbisoftware.com



Please fill out your session evaluation before leaving!

C10 - DB2 DPF Successes: Monitoring and tuning a hybrid IBM InfoSphere Warehouse