Flash
Solutions Engineering

Evidence-Based Decision Making: Using AWR and Statspack to Decide If Flash Is Right for You

Michael Ault IBM
Oracle FlashSystem Consulting Manager

NCOUG Nov 2015

Michael R. Ault

Oracle Guru

- Nuclear Navy 6 years
- Nuclear Chemist/Programmer 10 years
- Kennedy Western University Graduate
- Bachelors Degree Computer Science
- Certified in all Oracle Versions since 6
- Oracle DBA, author, since 1990
- Worked with Flash since 2007

ORACLE ACE

ORACLE CERTIFIED PROFESSIONAL

Storage Infrastructure Matters

Data Economics for Today's Workloads

Storage Systems

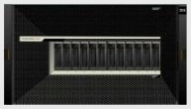
Tier 0 Acceleration

FlashSystem Family

Enterprise Storage Systems

DS8000

XIV



FlashSystem V9000

Midrange and Entry Storage Systems

Storwize V3700

Storwize V5000

Storwize V7000 & Unified

Tape and Virtual Tape Systems

TS7700

ProtecTIER Deduplication

Tape Library and Automation

Tape drives for Enterprise and LTO6

Software Defined Storage

- IBM Spectrum Control
- IBM Spectrum Protect
- IBM Spectrum Archive
- IBM Spectrum Virtualize
- IBM Spectrum Accelerate
- IBM Spectrum Scale

Integrated Solutions

PureFlex

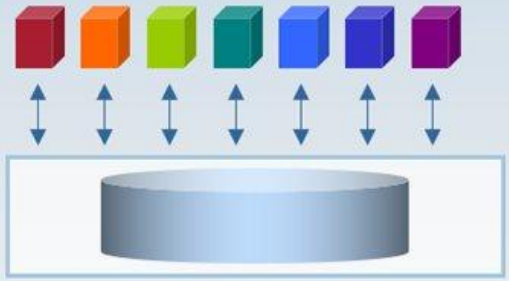
Elastic Storage Server

cisco IBM

VersaStack

Database workload environments

E-commerce



Scalable Transactional Database

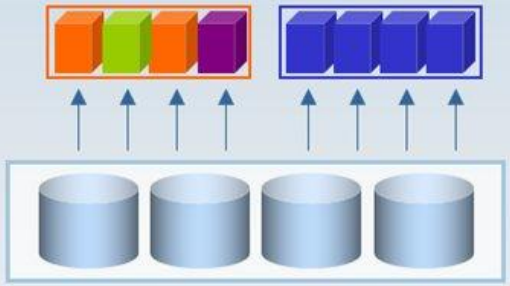
Transaction Processing

Random reads & updates

Many transactions with narrow data scope accessing the same database

Requires low latency and high IOPS

Customer Analysis



Analytics Data Warehouse

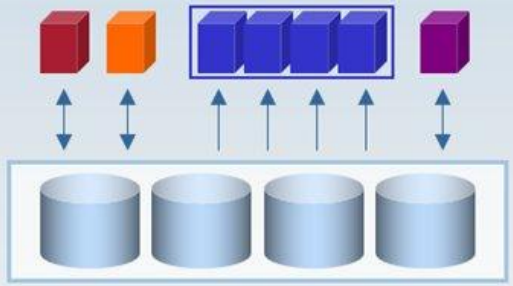
Reporting and Analytics

Random reads & sequential data loads

Analytics with broad data scope, parallelized across data partitions

Requires low latency and high bandwidth

Real Time Fraud Detection



Operational Data Warehouse

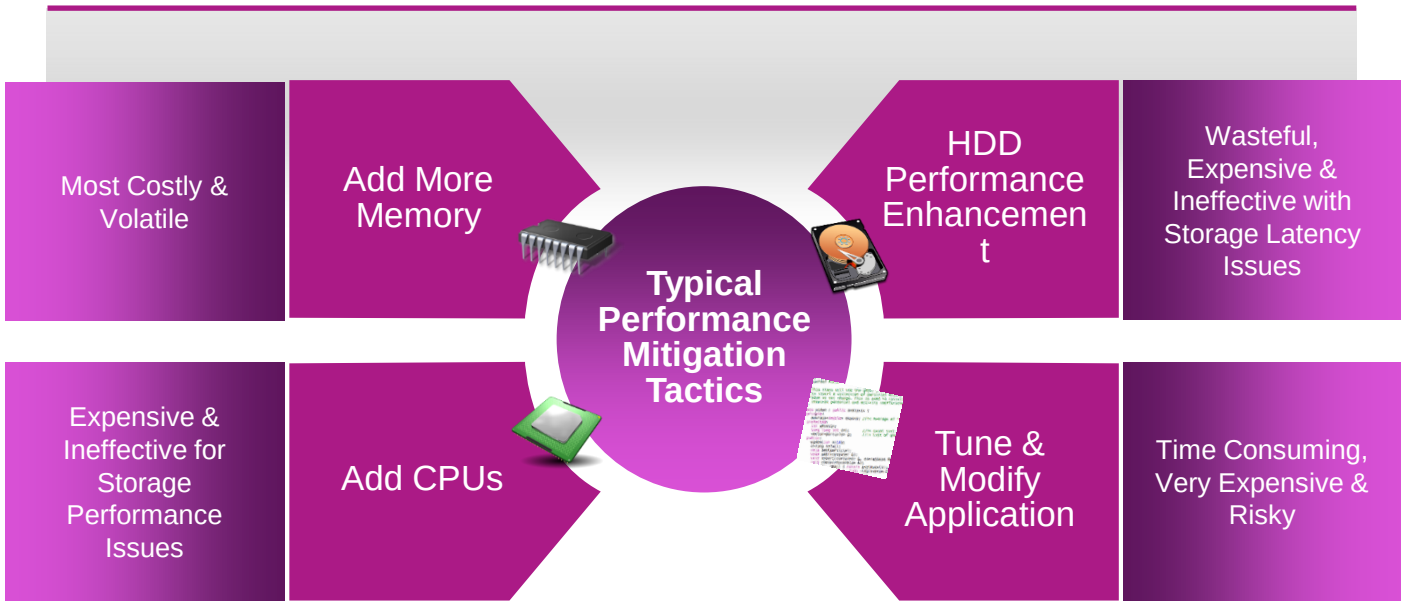
Operational Analytics

Random and sequential reads & data loads + continuous ingest

Analytics split into many parts and narrow scope operations, all running in parallel

Requires low latency and high IOPS

Datacenter's Response to Bridge Disk Performance Gap



What if we only reduced latency? Little's Law – Queuing Theory

Considering Little's Law as it applies to application performance..

Q / T = Rate

Q = the number of parallel IO requests
T = the I/O request service time
R = the rate, measured in IOPS or bandwidth

Assigning disk values to this equation:

$$\frac{20}{0.005} = 4{,}000 \text{ IOPS}$$

(5 milliseconds)

Substituting Exadata performance*:

$$\frac{80}{0.001} = 80{,}000 \text{ IOPS}$$

(1 millisecond)

Substituting FlashSystem performance:

$$\frac{20}{0.0001} = 200{,}000 \text{ IOPS}$$

(0.1 millisecond)

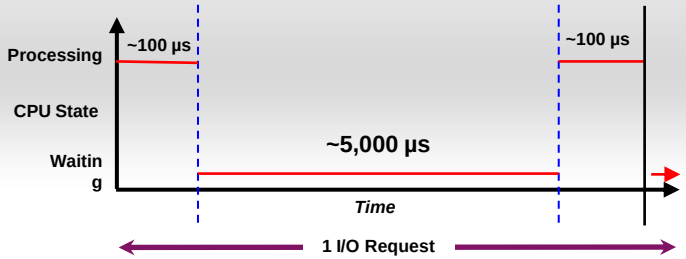
This is a 50X improvement in response time and the amount of work completed!

Microsecond latency maximizes CPU utilization

I/O Serviced by Disk

1. Issue I/O request	~ 100 µs
2. Wait for I/O to be serviced	~ 5,000 µs
3. Process I/O	~ 100 µs

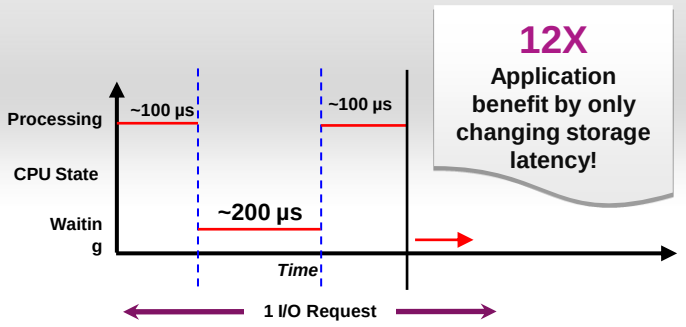
- **Time to process 1 I/O request = 200 µs + 5,000 µs = 5,200 µs**
- **CPU Utilization = Wait time / Processing time = 200 / 5,200 = ~4%**



I/O Serviced by IBM FlashSystem

1. Issue I/O request	~ 100 µs
2. Wait for I/O to be serviced	~ 200 µs
3. Process I/O	~ 100 µs

- **Time to process 1 I/O request = 200 µs + 200 µs = 400 µs**
- **CPU Utilization = Wait time / Processing time = 200 / 400 = 50%**



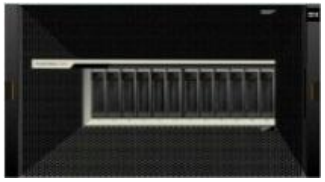
12X
Application benefit by only changing storage latency!

What is IBM FlashSystem ?

- Fully RAS compliant
- Highly reliable and redundant
- Up to 57 TiB per enclosure
- Latency from 95-195 us at interface
- IOPS to 1.2 million
- FC, IB, iSCSI, etc
- Full SAN features in V9000
- Latency 95-190us at enclosure, less then 0.4 ms (400 us) at Application

FS900

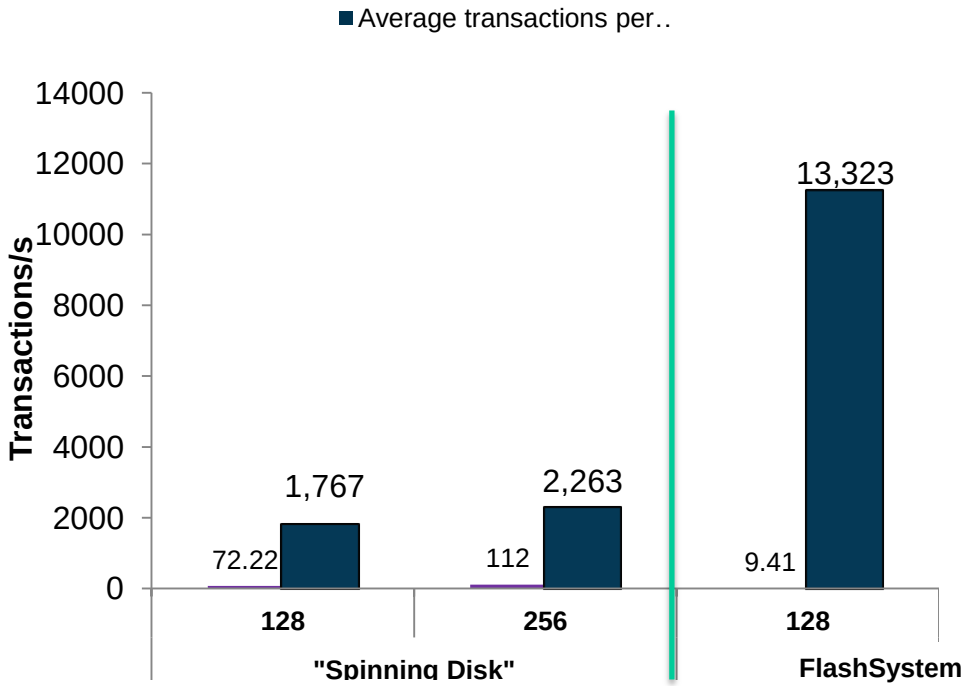
V9000



FlashSystem increases performance for storage

Comparison of transaction rates and response times for 6,000 clients.

Even at double the thread count for disk, FlashSystem out performed spinning disk by almost 6X while maintaining an application response time under 10ms.



Please see IBM POWER8 and IBM FlashSystem accelerate Oracle database whitepaper for more information

Where does latency matter with Databases?

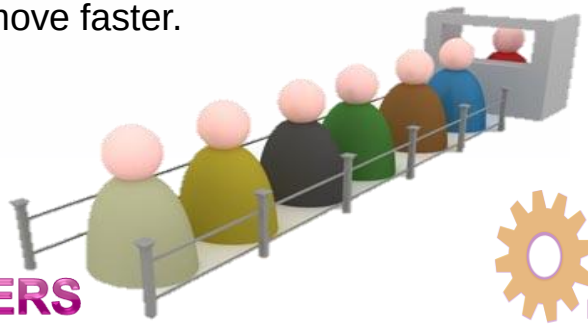
Reads

- In a read intensive environments, the ability of user processes to read from tables and indices can be a performance bottleneck.
- Think of the speed that data can pass through the funnel as determining the latency. The width of the funnel is equivalent to the bandwidth.
- Decreasing latency allows the queue to move faster while higher bandwidth allows more data to move in parallel.
- The faster reads can be completed, the faster analytic lookups are completed as well.

Usually 80/20 reads/writes!

Writes

- In write intensive environments, for example: the Oracle redo log can be a performance bottleneck.
- Much like a single queue, a redo log write is extremely sensitive to latency.
  - Write… wait… write…. Wait
- Temporary writes are another source
  - Sorts, bitmaps, global tables
- Decreasing latency allows the queue to move faster.



LATENCY MATTERS EVERYWHERE

HOW DO YOU DECIDE IF FLASH WILL HELP?

You use the Evidence

- AWR - Allows diff based reporting of database statistics
- ASH – Allows for review of current session history
- ADDM – Provides Oracle internal analysis based recommendations

- Each can be a full presentation by itself
- We will cover AWR

AWR – A Brief History

- Forerunner was BSTAT-ESTAT
  - Two scripts
  - First created tables and populated with key statistics from V$ views
  - Second took new set of data from V$, did diff, generated report and dropped tables
- Father was Statspack
  - A set of scripts no license required and is still available
  - Use setup to create a tablespace and user and a set of tables
  - Tables were permanent
  - Used JOBS to run a collection script to put V$ stats and timestamps into tables
  - Used reports to do diff, global, comparison reports
- AWR
  - Internalized Statspack
  - Requires Diagnostic and Tuning packs

AWR Contents

- Thousands of statistics
- Report can run to dozens of pages
- Header
- Summary
  - Basic system details, CPUs, Memory, Configuration, summarized statistics
- Time Model
- OS Statistics
- Waits
  - Foreground and Background
  - Aggregates by type
  - Aggregates by service
- SQL
  - Slices and dices top SQL statements by different criteria
- Instance Statistics
- IO Stats
- Buffer Pool Stats
- Advisory
- Wait stats
- Undo Stats
- Latch Stats
- Segment Statistics
- Dictionary Cache
- Library Cache
- Memory
- Streams
- Queues
- Resource Limits
- Shared Server
- Init.ora parameters
- RAC stuff

We Aren’t Looking at All of it

- For this presentation we are looking at IO related
  - Some summary stuff from header
  - Some time model statistics
  - Top 5/Top 10 report
  - Foreground/Background wait events
  - Some service statistics
  - SQL Physical Reads and Versioning
  - Some instance activity statistics
  - Thread stats
  - IO Stats
  - Buffer pool advisory
  - PGA Histogram
  - UNDO stats
  - Segment Read Statistics
  - Memory Thrashing (If using AMM)
  - Is shared server running?
  - Initialization parameters

What Collection Interval

- Get report from peak period
- The longer the period the more averaging and less likely to get peaks
- Needs to be when DB is most active
- A report from an idle DB is useless
- Defaults to 1 hour which is generally sufficient

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
TEST1	757959171	TEST1	1	16-Mar-13 19:03	11.2.0.2.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
TEST1	AIX-Based Systems (64-bit)	28	14		145.00

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	135860	29-Mar-13 00:00:20	572	236.8
End Snap:	136525	11-Apr-13 15:00:46	585	235.0
Elapsed:		19,620.44 (mins)		
DB Time:		284,745.51 (mins)		

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
EXAMP1	1580239890	EXAMP1	1	07-Oct-14 20:10	11.2.0.3.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
EXAMP1	AIX-Based Systems (64-bit)	208	52		256.00

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	63128	10-Oct-14 18:00:53	909	2.8
End Snap:	63139	10-Oct-14 20:45:57	993	2.7
Elapsed:		165.07 (mins)		
DB Time:		42,618.24 (mins)		

What Makes a Good AWR for Analysis?

- Proper time interval
  - 7 days is not good, 1-2 hours at peak load ideal
- DB time to Elapsed time ratio
  - Should be multiple not fractional
- IO wait to DB time or Busy time ratio
  - Should be 30% or greater unless we are looking at batch time or specific SQL time reductions
  - IO waits should dominate top 5 events list otherwise not an IO issue!

What About RAC?

- RAC – Real Application Clusters
- You can get one report for each node
- You can get a single report for all nodes
- Usually the one report for each node is easier to work with otherwise you lose some stats!
- If it is a RAC system you must look at all active nodes to get complete picture
- May change “Top 5 IO events” so beware!

IBM

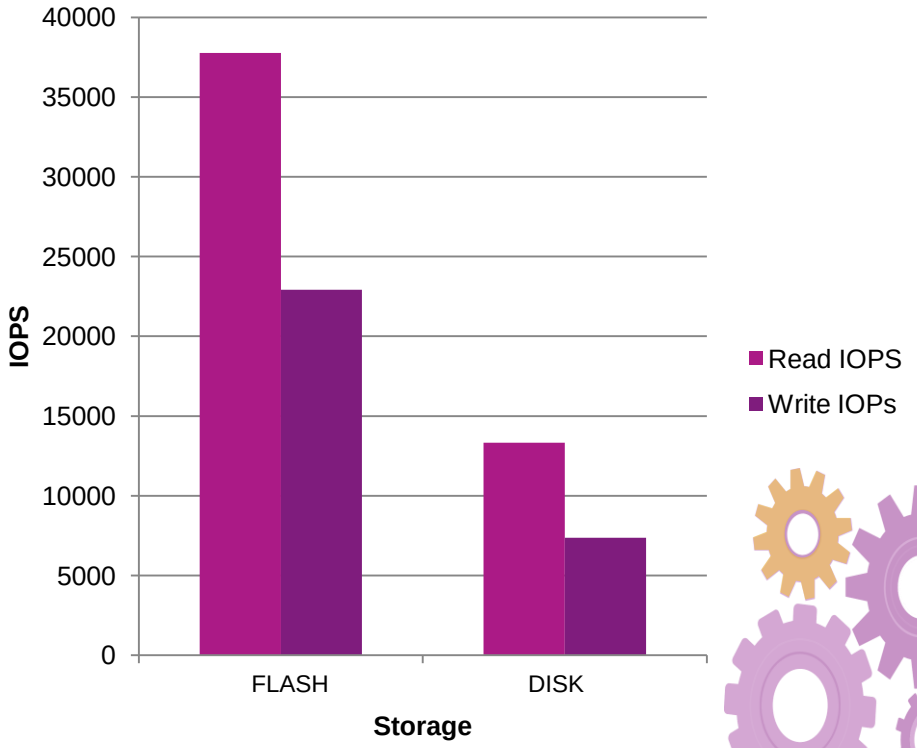
What Are We Interested In?

- IO related information
  - IOPS (I/Os per second)
  - Total time waited
  - IO related wait events
  - Tuning relating to IO
    - Co-location of logs/data
    - Memory issues
    - Tables/Indexes that do most IO
  - Average size of reads and writes
- General Tuning Information
  - CPU usage
  - Parameter settings
  - SQL Review
  - Latches, Enqueues and Mutexs

Where Do We Get It?

IOPS

- Four general locations
  - Load Profile
    - This is actually blocks IOPS not physical IOPS
  - Instance Activity Statistics
    - Gives true physical IO call data
  - IOStat breakouts (11g)
  - Tablespace IO Listing



Load Profile

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
Physical reads:	**5,366.3**	**178.3**		
Physical writes:	**71.5**	**2.4**		

- Sum the per second values here for block IOPS
- Generally block IOPS will be greater than actual IOPS
- Block IOPS will not include non-data IO

Instance Activity Statistics

Instance Activity Stats

•Ordered by statistic name

Statistic	Total	Per Second	Per Transaction
physical read total IO requests	16,623,876	4,613.35	82,705.85
physical read total bytes	951,898,777,600	264,164,710.22	4,735,814,813.93
physical write total IO requests	3,072,181	852.57	15,284.48
physical write total bytes	292,110,947,840	81,064,715.81	1,453,288,297.71

- Sum the per second values here for total IOPS
- Use total bytes per second with IOPs to generate KB/Op for reads and writes

IOStat Breakouts

- IOStat by Function summary
- 'Data' columns suffixed with M,G,T,P are in multiples of 1024 other columns suffixed with K,M,G,T,P are in multiples of 1000
- ordered by (Data Read + Write) desc

Function Name	Reads: Data	Reqs per sec	Data per sec	Writes: Data	Reqs per sec	Data per sec	Waits: Count	Avg Tm(ms)
Others	51G	12.14	7.24189	47.4G	7.48	6.74345	44.1K	1.55
LGWR	200M	1.81	.027760	95.7G	433.63	13.6025	855.7K	2.57
RMAN	39.6G	10.42	5.63289	199M	0.11	027621	6501	1.05
DBWR	0M	0.00	0M	30.1G	138.96	4.27526	0	
Buffer Cache Reads	12G	11.50	1.70074	0M	0.00	0M	80.5K	8.60
Direct Reads	10.4G	2.78	1.47200	0M	0.01	0M	0	
Direct Writes	0M	0.00	0M	1.3G	0.98	.190714	0	
TOTAL:	113.1G	38.65	16.0752	174.8G	581.16	24.8395	986.8K	3.01

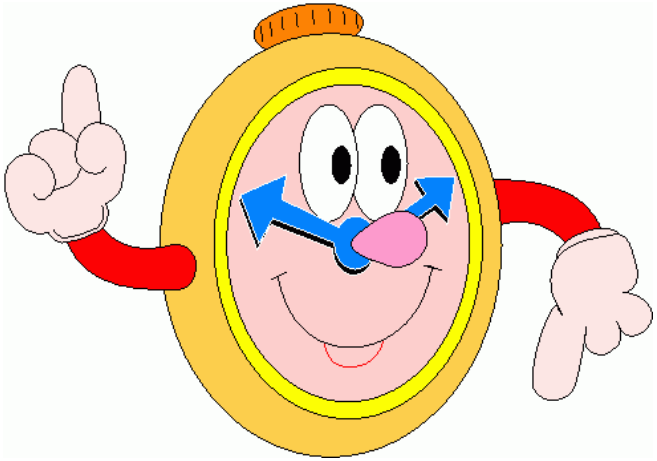
Tablespace IO Statistics

- Tablespace IO Stats
- ordered by IOs (Reads + Writes) desc

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
DATA	4,763,047	663	0.85	1.65	314,786	44	399,033	0.34
IDX	1,856,960	259	0.98	1.00	347,741	48	8,735	0.91
INTER	2,044,717	285	0.47	1.04	16,989	2	351	0.37
XDATA	664,821	93	0.89	3.21	287,366	40	3,010	11.91
XPURGE	377,822	53	1.01	1.67	440,894	61	2,971	0.57
UNDTS2	161,347	22	0.81	1.00	314,238	44	4,032	0.04
QUEUES	324,049	45	0.62	5.79	18,673	3	54	0.37
QVD	161,084	22	0.47	1.03	87	0	0	0.00
UNDTS4	156,039	22	0.96	1.00	0	0	49	0.00

Sum Av reads/s and Av writes/s. Older versions may not have total stats.

Where Do We get It?



Total IO call time

- 4 General locations
  - Operating System Statistics
    - Least accurate
  - Foreground Wait Class
  - Service Wait Class Statistics
  - Instance Activity Statistics
- Use the highest number (after conversion to same units)

Operating System Statistics

Operating System Statistics

- *TIME statistic values are diffed. All others display actual values. End Value is displayed if different

Statistic	Value	End Value
BUSY_TIME	1,234,204	
IDLE_TIME	6,104,112	
IOWAIT_TIME	567,177	

- May be in milli or centa seconds depending on OS
- IOwait time here is usually wrong

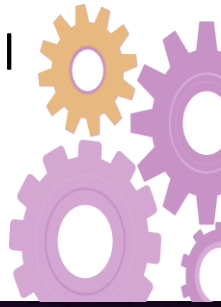
Foreground Wait Class

- s - second, ms - millisecond - 1000th of a second
- ordered by wait time desc, waits desc
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0
- Captured Time accounts for 93.8% of Total DB time 25,979.68 (s)
- Total FG Wait Time: 17,377.04 (s) DB CPU time: 7,001.59 (s)

Wait Class	Waits	%Time - outs	Total Wait Time (s)	Avg wait (ms)	%DB time
User I/O	**6,078,880**	**0**	**16,818**	**3**	**64.74**
System I/O	**7,093**	**0**	**1**	**0**	**0.00**

- Sum System and User IO, may not include all background IO time

If *Log File Sync* is in Top 5 then you may want to add Commit totals as well



Service Wait Class Statistics

Service Wait Class Stats

- Wait Class info for services in the Service Statistics section.
- Total Waits and Time Waited displayed for the following wait classes: User I/O, Concurrency, Administrative, Network
- Time Waited (Wt Time) in seconds

Service Name	User I/O Total Wts	User I/O Wt Time	Concurcy Total Wts	Concurcy Wt Time	Admin Total Wts	Admin Wt Time	Network Total Wts	Network Wt Time
SYS$USERS	6071136	16794	41834	61	54	5	6123220	175
Other	7496	24	21	0	0	0	36148	4
SYS$BACKGROUND	20502	95	174	12	0	0	280838	27

- Sum down the User I/O Wait time column

OTHER – The sum of all services not USER or BACKGROUND

Instance Activity Statistics

Instance Activity Stats

•Ordered by statistic name

Statistic	Total	per Second	per Trans
user I/O wait time	1,691,405	468.66	15.57

- This is in milli or centa-seconds
- Not in Oracle11.2.04
- Back in Oracle12c

Top 5 IO waits - Where Do We Get It?

- IO related Wait events
  - Anything starting with “db file”
  - Some starting with “Log”
  - Some starting with “direct path”
  - In Exadata start with “cell”
  - Start with TOP Five listing
  - Foreground Wait events listing
  - Background Wait Events listing
- Generally once you get the top five you have enough

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	208,743,998	992,504	5	38.81	User I/O
DB CPU		324,737		12.70	
db file parallel read	12,405,740	199,652	16	7.81	User I/O
SQL*Net more data to client	145,669,425	164,734	1	6.44	Network
read by other session	17,428,039	137,564	8	5.38	User I/O

General Waits

- DB File
- Log
- Undo
- Parallel Query (PX)
- RAC (GC)
- Exadata

DB File Type Waits

DB File Sequential Reads – memory starvation, non-selective indexes
DB File Scattered Reads – full table scans, insufficient indexing
Direct Path Writes – Appends, data loads
Direct Path Reads – Parallel slaves used to retrieve data
DB File Parallel Writes – Backup and partition use
DB File Parallel Reads – Partition use
DB File Single Write – File header writes, excessive data files
Direct path read temp – Temp file activity (sorts, hashes, temp tables, bitmaps)
Direct path write temp – Temp file activity (sorts, hashes, temp tables, bitmaps)

Log Type Waits

log file sync – Could indicate excessive commits
log file parallel write – Look for log file contention

log buffer space – Look at increasing log buffer size
log file switch (checkpoint incomplete) – May indicate excessive db files or slow IO subsystem
log file switch (archiving needed) – Indicates archive files are written too slowly

log file switch completion – May need more log files per threadc

Exadata

- Cell single block physical read – Same as db file sequential read
- Cell multi block physical read – Same as db file scattered read
- Cell list of blocks physical read – Similar to index scan
- Cell smart table scan – Offloaded full table scan

GC Events

gc cr multi block request – Full table or index scans
gc current multi block request – Full table or index scans

Top Five Wait Events

Top 5 Timed *Foreground* Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	4,359,251	10,553	2	40.62	User I/O
DB CPU		7,002		26.95	
db file scattered read	829,528	2,216	3	8.53	User I/O
read by other session	587,095	2,131	4	8.20	User I/O
direct path read	263,108	1,585	6	6.10	User I/O

Starting in 11g, background waits aren't shown as top five events, you must manually review and include them. In 12c they have been put back in.

Foreground Wait Events

Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn	% DB time
db file sequential read	4,359,251	0	10,553	2	40.13	40.62
db file scattered read	829,528	0	2,216	3	7.64	8.53
read by other session	587,095	0	2,131	4	5.41	8.20
direct path read	263,108	0	1,585	6	2.42	6.10
db file parallel read	13,085	0	322	25	0.12	1.24
SQL*Net message from dblink	24,587	0	161	7	0.23	0.62

Background Wait Events

Event	Waits	%Time - outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn	% bg time
log file parallel write	144,466	0	335	2.32	1.33	57.53
db file sequential read	15,497	0	71	5	0.14	14.29
db file parallel write	4,130	0	45	11	0.04	8.91
LNS wait on SENDREQ	280,888	0	27	0	2.59	5.30
db file scattered read	1,175	0	18	16	0.01	3.70

Always validate that the physical waits in the foreground are greater than in the background, note that *log file parallel read* should be in top 5.

Script to Get Log List

```
column group#  format 999999
column  member format a32
column meg format 9,999
set lines 80 pages 60 feedback off verify off
ttitle 'Redo Log Physical Files'
break on group#
spool rdo_file
select distinct b.thread#,a.group#,a.member,b.bytes/(1024*1024) meg, b.status
from sys.v_$logfile a,
sys.v_$log b
where a.group#=b.group#
order by thread#/
spool off
clear columns
clear breaks
ttitle off
```

Redo Log List

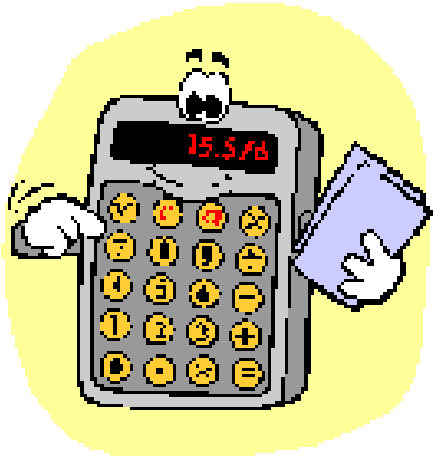
Date: 09/04/07 Page: 1
Time: 02:20 PM Redo Log Physical Files SYSTEM
ault11g2 database

THREAD#	GROUP#	MEMBER	MEG	STATUS
1	1	+DATA/ault11g/onlinelog/group_1.	50	INACTIVE
1		+RECOVERY/ault11g/onlinelog/grou	50	INACTIVE
1	2	+DATA/ault11g/onlinelog/group_2.	50	CURRENT
1		+RECOVERY/ault11g/onlinelog/grou	50	CURRENT
2	3	+DATA/ault11g/onlinelog/group_3.	50	CURRENT
2		+RECOVERY/ault11g/onlinelog/grou	50	CURRENT
2	4	+DATA/ault11g/onlinelog/group_4.	50	INACTIVE
2		+RECOVERY/ault11g/onlinelog/grou	50	INACTIVE

So What do We do With this Information?

- I fill in my spread sheet
- I use an example report to draft findings report
- I also still review AWR for “low hanging fruit”
- You do this manually…

How to Get Numbers?



- I calculate IO balance based on:
  - Small block reads
  - Large/average IOs
  - Writes
- Using IO balance establish current latencies
- Then calculate projected latency for replacement system
  - Based on linear graph of projected latency per blocksize
  - Use single block for sequential wait events
    - Db file sequential reads
    - Control file sequential reads
    - Maybe read by other session
  - Use calculated average IO size to project latency for large block IO
    - IOPS and total bytes for reads and writes

Example

Instance Activity Stats

Statistic	Total	per Second	per Trans
physical read total IO requests		1,313.00	
physical write total IO requests		73.00	
user I/O wait time	476,000		

Version 11.2

	Bytes/second	KB per Op	Proj Latency
Read	14,631,466.18	10.88	512.38
Write	1,051,415.68	14.07	250.46

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file seq read	886,534	4,456	5.03	60%	
db file parallel read	46,381	278	6.00	4%	
log file sync	19,710	39	1.95	1%	
db file paralle write	1,027	21	20.45	0%	
read by other session seq	2,237	20	8.81	0%	
Total	955,889	4,814	5.04	65%	

DB time	124.03	
Expected RL	512.38	
Expected WL	250.46	
CPU Time	722.00	
CPUS	16.00	Accounted for:
Total Time	1,440,960.00	744,171.00
Elapsed	15.01	
Default Lat	500.36	

Measures we can improve	%Wait time accounted fo	Current response time (µseconds)	Projected response time (µseconds)	%New wait time	Proj Time (s)
db file seq read	59.88%	5026	500.36	5.96%	443.59
db file parallel read	3.74%	6000	512.38	0.32%	23.76
log file sync	0.52%	1953	512.38	0.14%	10.10
db file paralle write	0.28%	20448	250.46	0.00%	0.26
read by other session seq	0.26%	8806	500.36	0.02%	1.12
	64.68%	42234	500.92	6.43%	478.83

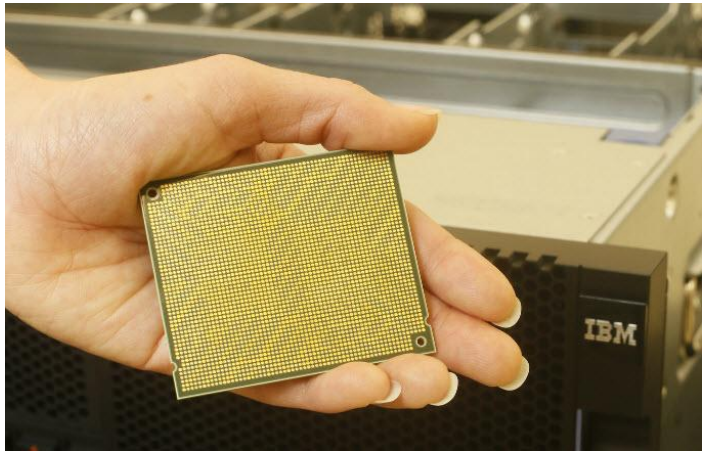
Wait Time

Current	Projected	Total Improvement	Total Batch Reduction
64.68%	6.43%	905.27%	88.95%

What do You do With the Numbers

- Determines how much time can be returned to CPU
- Determines how much shorter batch cycles can be
- May help improve the users experience
  - Many factors including network speed and application

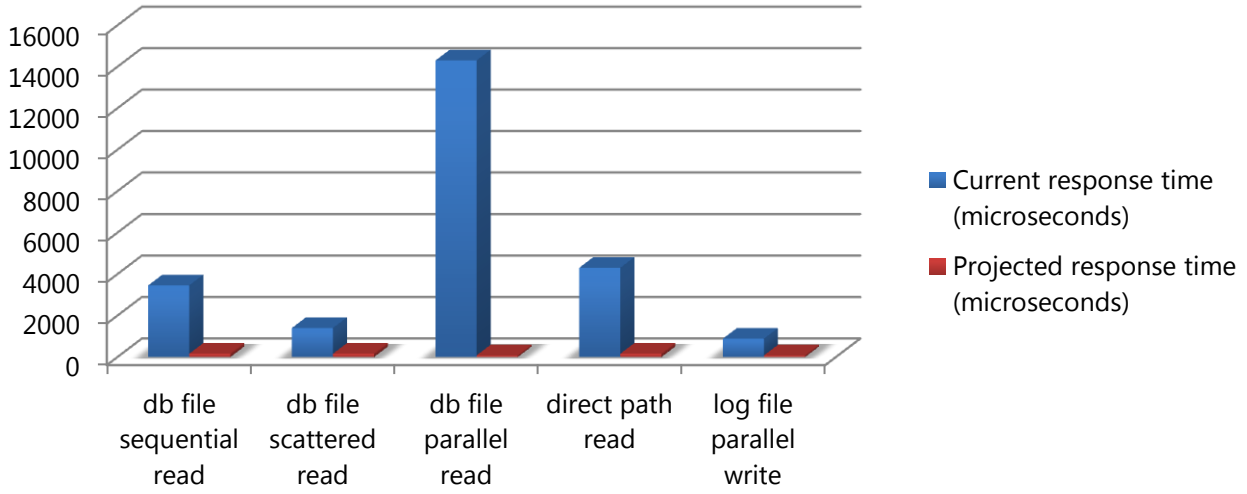
	CentSeconds	
DB Time	588,960.00	
	Current	Projected
Busy	259,571.00	695,965.07
IOWait	484,600.00	48,205.93
Percentage of Total Time		
	Current	Projected
Busy	18%	48%
IOWait	34%	3%



For Example

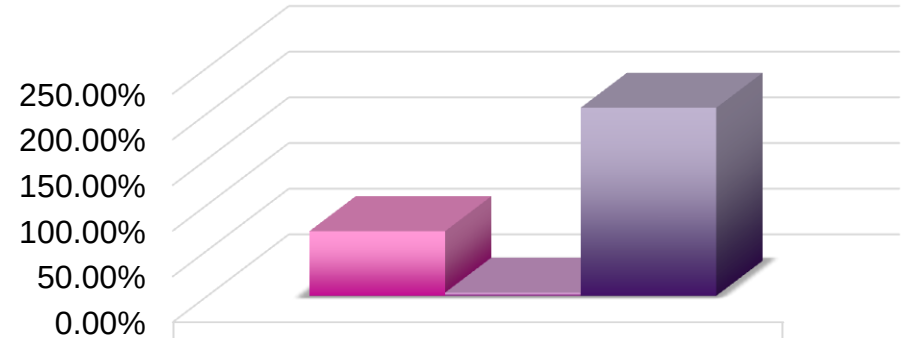
- SLA says report must complete within 10 minutes
- Currently report takes 25 minutes
- To be valuable your reduction must be greater than 60%
- If your current IO contribution to this process is greater than 60% then you can fix it with low latency IO
- Use the evidence from AWR to prove your point!

Wait Improvement Graph



On my spreadsheet auto-populated from waits, use top 5 IO waits from fore- and Background listings.

Overall Improvement Graph

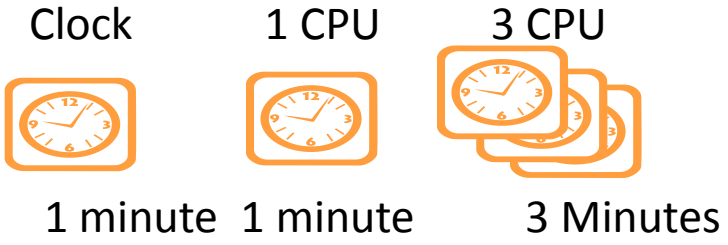


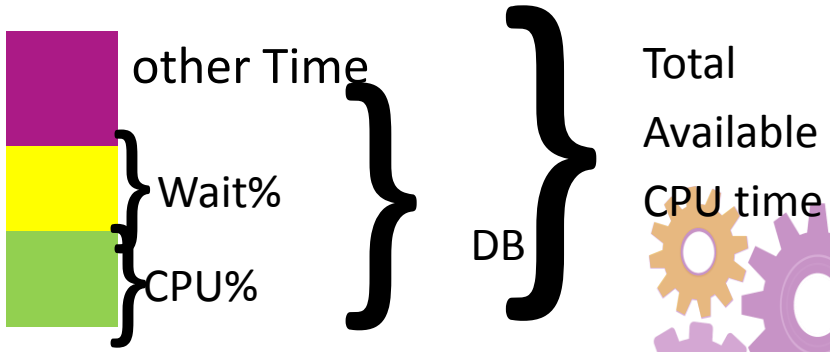
	1
Current wait time total	70.68%
New project wait time	3.39%
Total Improvement	205.79%

How much we can improve the physical IO wait times, not overall performance!

A Few Words on CPU%

- CPU time and Clock time are different
  - CPU=Clock*effective CPU count
    - CPU count is effected by number of CPUs/core and hyperthreading
- CPU Time and DB Time are different
  - DB time is the amount of time out of CPU time that the DB used the processors
  - The CPU% reported in AWR is the percent of DB Time spent in the CPU, the rest was spent waiting for resources
  - CPU% + Wait%=100% of DB Time





Time and AWRs

- Sometimes the report tells you time units
- Sometimes it doesn’t
- It may use centa-seconds, nano-seconds, milli-seconds or seconds depending on the statistic
- You may have to adjust the IO wait time calculations to be the same units. Usually it will be a factor of 1, 10 or 100 depending on the OS and version of AWR/statspack
- Usually User IO wait time, if there is an IO problem, will be at the same level of magnitude as BUSY time so if it is showing up a factor of 10 or 100 low, you can usually figure it out.

Segment Statistics

- Tells by type of action what segments are seeing most activity
- Tune objects by adding indexes, changing rows per block (RAC), tuning object related SQL

Segment Statistics

Segments by Physical Reads DB/Inst: TEST/test Snaps: 24080-24107
-> Total Physical Reads: 27,349,451
-> Captured Segments account for 53.8% of Total

Tablespace Owner	Name	Object Name	Subobject Name	Obj. Type	Physical Reads	%Total
DNE4	MEP_D01	MODE_EXECUTION_POINT		TABLE	3,350,928	12.25
FMS	FMS_DATA	ACCTNG_EVENT		TABLE	2,720,574	9.95
DNE4	SHIPMENT_D	SHIPMENT_PLAN		TABLE	1,046,173	3.83
ISS	ISS_DATA	IB_SHIPMENT_STATUS_H		TABLE	852,046	3.12
DNE4	TMS_AUDIT	TEMP_SQL		TABLE	552,676	2.02

Segment Statistics

Segments by Direct Physical Reads DB/Inst: TEST/test Snaps: 24080-24107
-> Total Direct Physical Reads: 6,048,316
-> Captured Segments account for 77.9% of Total

Tablespace Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Direct Reads	%Total
FMS	FMS_DATA	ACCTNG_EVENT		TABLE	2,720,470	44.98
DNE4	MEP_D01	MODE_EXECUTION_POINT		TABLE	1,468,961	24.29
FMS	FMS_DATA	FIN_EVENT		TABLE	521,600	8.62
FAM	FAM_D	SYS_LOB0000148435C00		LOB	360	.01
SYS	SYSAUX	SYS_LOB0000008958C00		LOB	19	.00

Segment Statistics

Segments by Table Scans DB/Inst: TMSP/tmsp Snaps: 24080-24107
-> Total Table Scans: 6,618
-> Captured Segments account for 2.5% of Total

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Table Scans	%Total
DNE4	INDEX01	GP_CITY_I4		INDEX	41	.62
DNE4	RATE_INFO_	RATE_I3		INDEX	40	.60
DNE4	MEP_D01	MODE_EXECUTION_POINT		TABLE	25	.38
FMS	FMS_DATA	ACCTNG_EVENT		TABLE	10	.15
DNE4	DNE4_D02	ME_SERVICE_GROUP		TABLE	6	.09

Low Hanging Fruit

- Number of sorts and workarea executions single and multi pass
  - Look at sort histogram – PGA Aggr Target Histogram if most are below 512 mb then OOB sorts look at DISPATCHER and SHARED SERVER settings
- Number of SQLNet roundtrips to client per transaction (Instance Activity Stats)
  - >100 impacts clients perception of performance
- Number of redo log switches per hour (Instance Activity Stats – Thread Activity)
  - Less the better, shoot for 4 per hour per Oracle ROT
  - If excessive impacts logfile sync wait and performance
- Be sure filesystemio_options is set to setall if not using ASM
  - If not in initialization settings defaults to **none** and 30% performance hit
- Default and multi-cache size – Buffer Pool Advisor
  - If at double cache size physical IO reduced more than 20% indicates cache too small

Low Hanging Fruit

- SQL Versioning – SQL Version report
  - Excessive versions of SQL statements cause shared pool, latch and enqueue bloat and poor performance
- DB cache, shared pool, large pool trashing – Memory Resize Ops
  - Excessive thrashing between db cache and shared pool causes shared pool misses and reloads resulting in CPU and IO hit
  - Excessive thrashing between large pool and others can result in sorting issues
  - Excessive thrashing between streams pool and others can cause expdb and impdb failures

Temporary Activity to Disk

- Temporary activity includes:
  - Sorts
  - Hashes
  - Global Temporary Table Overflow
  - Bitmap Operation overflow
    - Create
    - Merge
- In 10g PGA_AGGREGATE_TARGET automated temporary segment processing
- Use of shared server negates automated temporary segment processing
  - Oracle uses old parameters
    - SORT_AREA_SIZE – default 64 mb
    - HASH_AREA_SIZE – default 2X SORT_AREA_SIZE
    - CREATE_BITMAP_AREA_SIZE – default 8 mb
    - BITMAP_MERGE_AREA_SIZE– default 1 mb

Temporary Activity to Disk

- How is shared servers turned on?
  - By default
    - Oracle sets DISPATCHERS – defaults to a derived name ending in XDB
    - Oracle sets SHARED_SERVERS – defaults to 1 (not shown in parameter listing)
  - Oracle development tools use the DISPATCHER that is created to connect
  - After development is over turn them off
  - If you don’t use Oracle tools turn them off

Signs Shared Server is turned on

Shared Servers Rates

Common Queue Per Sec	Disp Queue Per Sec	Server Msgs/Sec	Server KB/Sec	Common Queue Total	Disp Queue Total	Server Total Msgs	Server Total(KB)
0	0	0	0.00	0	0	0	0

Shared Servers Utilization

- Statistics are combined for all servers
- Incoming and Outgoing Net % are included in %Busy

Total Server Time (s)	%Busy	%Idle	Incoming Net %	Outgoing Net %
3,689	0.00	100.00	0.00	0.00

Shared Servers Dispatchers

- Ordered by %Busy, descending
- Total Queued, Total Queue Wait and Avg Queue Wait are for dispatcher queue
- Name suffixes: "(N)" - dispatcher started between begin and end snapshots "(R)" - dispatcher re-started between begin and end snapshots

Name	Avg Conns	Total Disp Time (s)	%Busy	%Idle	Total Queued	Total Queue Wait (s)	Avg Queue Wait (ms)
D000	0.00	3,689	0.00	100.00	0	0	

Symptoms in AWR of Temporary Segments to Disk

- Temporary tablespace one of top sources of IO

Tablespace IO Stats

•ordered by IOs (Reads + Writes) desc

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
TST_AUTO_TBL03	2,610,787	723	0.96	12.13	1,272,803	353	27,228	1.05
TST_TEMP	262,510	73	0.00	3.02	1,616,935	448	0	0.00
TST_AUTO_TBL04	814,444	226	0.05	31.12	3,678	1	55	5.64
LRX_LRG_T01	447,172	124	0.01	31.93	88	0	0	0.00
TST_RULE_3_PARTIX07	77,400	21	7.17	1.00	70,278	19	0	0.00
TST_RULE_3_PARTIX08	77,578	21	7.10	1.00	70,076	19	0	0.00

Symptoms in AWR of Temporary Segments to Disk

- Statistics in Instance Statistics section

Instance Activity Stats

•Ordered by statistic name

Statistic			
sorts (disk)	3	0.00	0.01
workarea executions - onepass	6,542	1.81	20.83
workarea executions - multipass	100	0	0

Symptoms in AWR of Temporary Segments to Disk

- PGA Aggr Target istogram shows segments <512 mb

PGA Aggr Target Histogram

•Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	9,315	9,315	0	0
64K	128K	89	89	0	0
128K	256K	1,081	1,081	0	0
256K	512K	1,723	1,723	0	0
512K	1024K	926	926	0	0
1M	2M	3,520	3,520	0	0
2M	4M	5,987	5,733	254	0
4M	8M	2,983	2,983	0	0
8M	16M	5,490	368	5,122	0
16M	32M	1,221	90	1,131	0
32M	64M	41	39	2	0
64M	128M	22	20	2	0
128M	256M	2	0	2	0
256M	512M	2	2	0	0

Why is 512 MB Important?

- PGA_AGGREGATE_TARGET is used to control overall memory assigned for temporary actions
- Each process gets 5% up to maximum set by _PGA_MAX_SIZE
- _PGA_MAX_SIZE defaults to 512 mb
- The automated process should handle temporary activity below 512 MB
- If you see temporary activity less than 512 mb going to storage this is out-of-band (OOB)

How do you know PGAT Is Right?

- PGA Memory Advisor

PGA Memory Advisory

•When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count	Estd Time
5,000	0.13	133,611,373.09	66,289,825.32	67.00	426	125,932,781,781
10,000	0.25	133,611,373.09	51,376,357.58	72.00	20	116,537,668,132
20,000	0.50	133,611,373.09	43,447,055.91	75.00	0	111,542,405,347
30,000	0.75	133,611,373.09	43,186,367.60	76.00	0	111,378,178,196
40,000	1.00	133,611,373.09	37,192,946.09	78.00	0	107,602,471,753
48,000	1.20	133,611,373.09	10,565,649.96	93.00	0	90,827,937,632

Things you Can Surmise

- If log file sync is excessive after moving to flash, look at lun setup, large IOs such as redo log should be on separate luns.
- Control file related waits probably indicate control files are co-located with other files
- Excessive full table scans (db file scattered reads) or direct IO indicates indexing issues in many cases
- In the last two columns of the Undo reports, non-zero entries may indicate undo tuning is needed
- Look at ratio of hard parse time to parse time, if greater than 50% may have SQL issues (ad-hock SQL, not sharing cursors, lack of bind variables, etc)
- At very beginning of report look at cache size compared to shared pool size, cache size should be several times larger than shared pool, if not indicates SQL issues

SO…is Flash right for your system?

- Are you IO challenged?
- Are IO related SLAs not being met?
- Is batch window stretching into Operational time?
- Are users complaining (ok, so they always complain, maybe more than usual?)

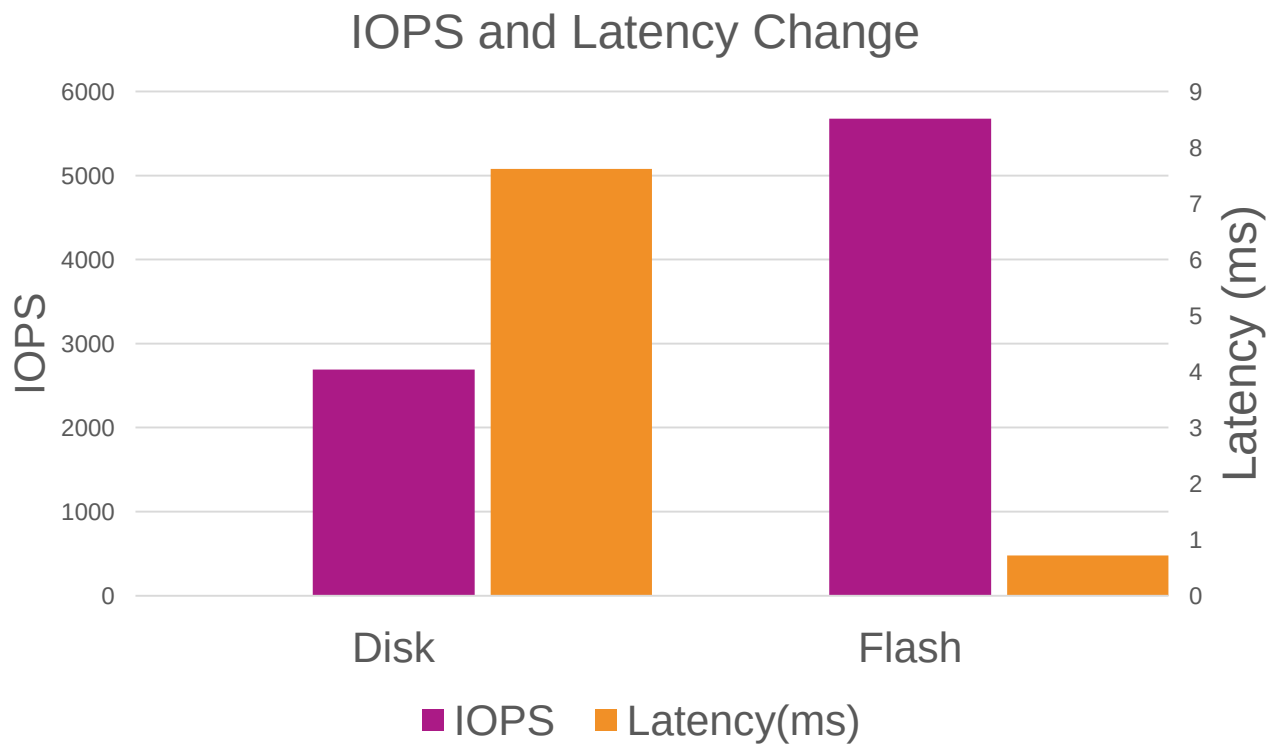
- Look at your AWR, ADDM, ASH, Statspack
- Review the evidence, use facts not feelings
- IBM will do this for you…for free!
- Contact your CTS

ACTUAL CLIENT POC

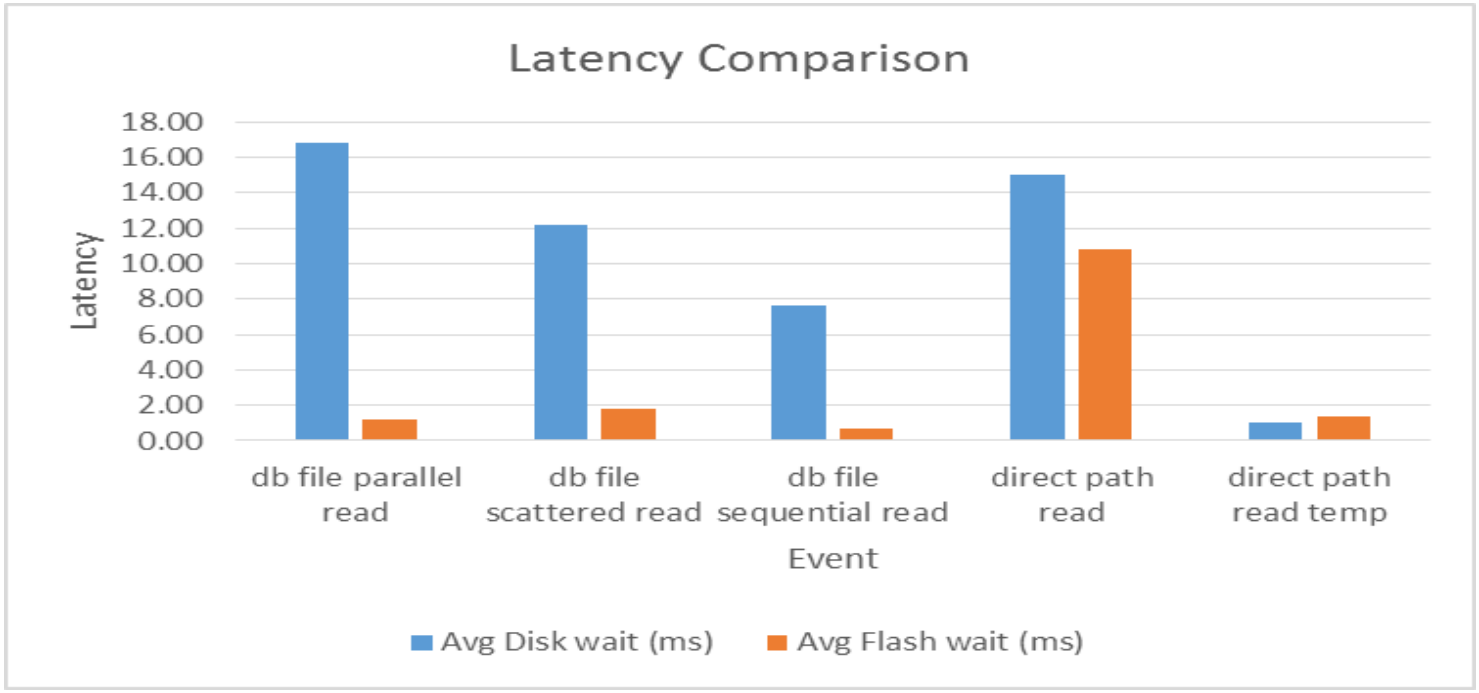
Actual Client POC Results

- 126 GB memory
- Using SVC
- Using FS900
- Average 74% read activity

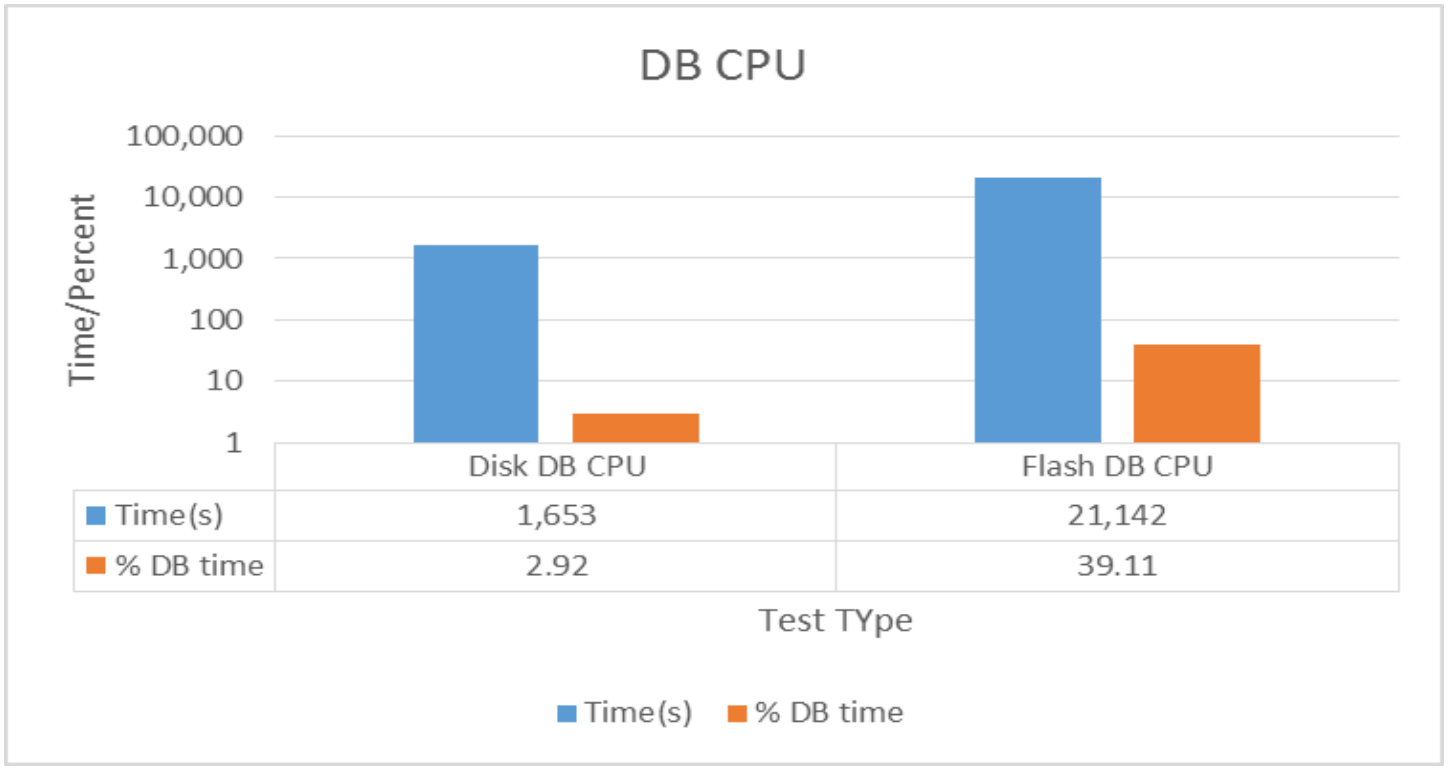
Actual Client POC – Average IOPS and Latency



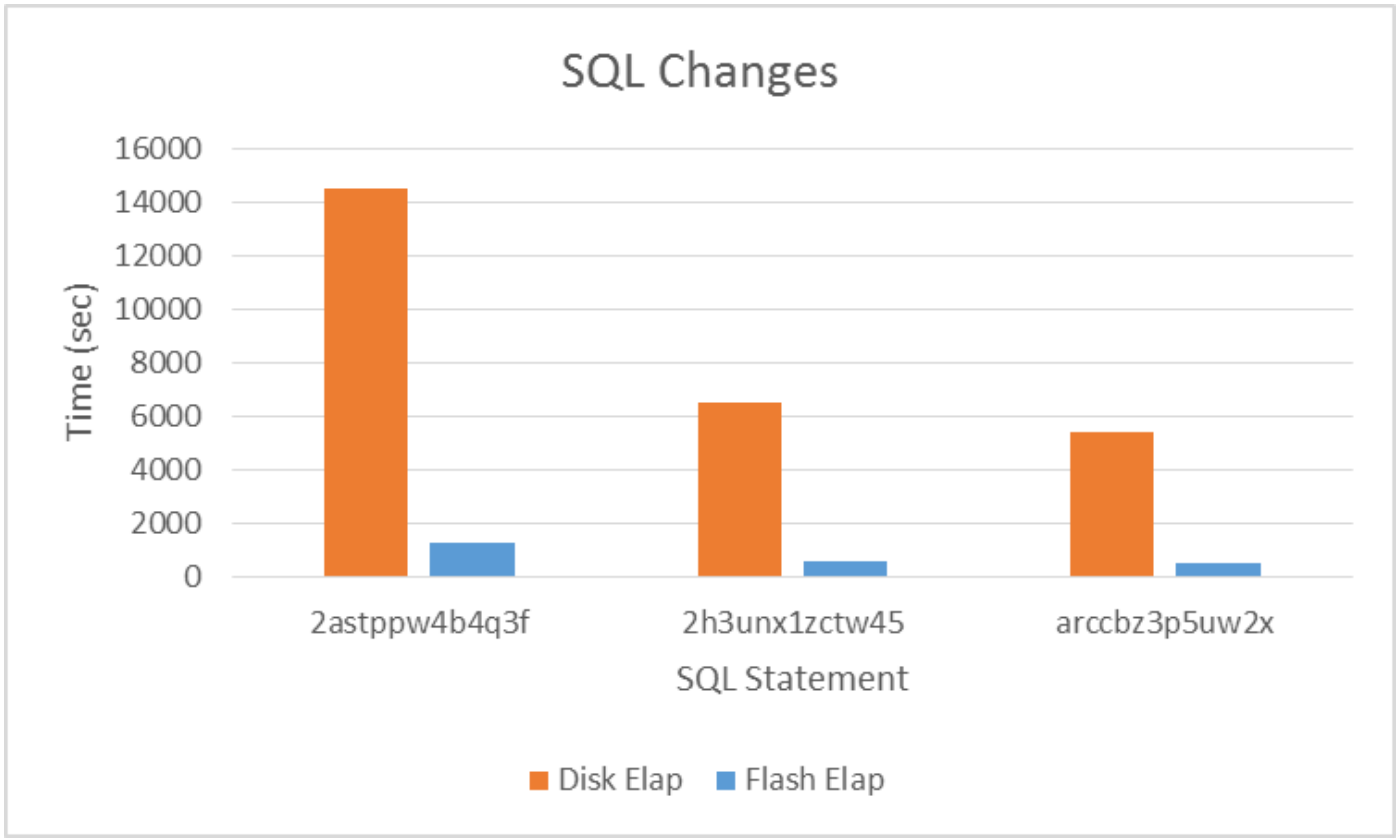
Actual Client POC Results – Top Five Waits



Actual Client POC - CPU



Actual Client POC – SQL Execute Times



ACTUAL CLIENT RESULTS

IBM FlashSystem® provides performance boost

Helps improve patient healthcare experience

Healthcare consumers often accept care without knowing the financial implications of their decision. TriZetto strives to change that through their proven claims administration systems and require high-performing infrastructure to capture and reliably deliver healthcare information seamlessly, with no unplanned downtime.

12% improvement
in application performance

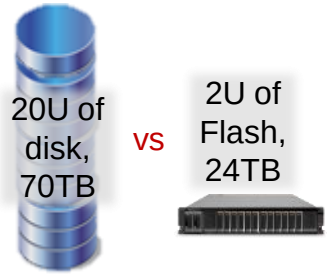
90% OPEX reduction
compared to disk

90% reduction
in storage latency

up to 10X improvement
In database maintenance operations

Hardware
- IBM FlashSystem storage
- Flex System™ server with IBM POWER7+™, IBM AIX®

Software
- TriZetto QNXT application w/Microsoft SQL Server
- TriZetto Facets application w/Sybase ASE 15.7



Replacing the disk storage solution with IBM FlashSystem provided TriZetto tangible performance improvement in its payer software. The reduction in batch cycles and improvement in online response in TriZetto payer applications seen with FlashSystem are vital.

With IBM FlashSystem storage, TriZetto can process claims faster, which can improve the patient experience, help customers make more informed healthcare decisions and reduce healthcare costs for insurers and members.

You can read more this solution in our whitepaper.

FIS™ accelerates global banking with IBM FlashSystem®

Profile is FIS' premier real-time ultra-scalable core banking application. It is based on FIS GT.M—a transaction processing database engine.

Why FlashSystem?

Customer growth leads to data growth. With over **17 million** credit card accounts, **42 million** loyalty accounts, **151 million** prepaid cards and **8 million** debit cards. FlashSystem produced the performance needed to service current and future growth.

A Smarter, Faster Solution

IBM FlashSystem, FIS Profile, FIS GT.M Database Engine, IBM Power Systems, IBM AIX

100X Improvement
in system latency

38% Improvement
in application response time

75% Improvement
in online backup time

40% Reduction
in batch processing

The IBM FlashSystem solution delivered improvements in front-end and back-end application response times and reduced overnight scheduled processing timelines and overall operational latency, enabling FIS to continue to meet their customer's SLA requirements, now and in the future.

You can read more this solution in our whitepaper.

IBM FlashSystem® storage solution enables massive scaling

Amid rising consumer demand, technologies like **smart meters** generate orders of magnitude more data and analysis than traditional meters, driving the need for faster, more efficient storage solutions in the eclectic utility industry.

5 Million Smart meters
Support with 16 TB of Flash

30% faster processing
Over traditional SSD storage

4.7X number of data streams
over traditional SSD storage

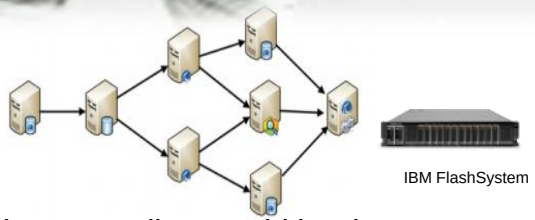
70 Million transactions
in 40 minutes

"IBM FlashSystem provided a platform that enabled PI Server 2014 to scale to levels never before met.."

—Alton Loe, Director, OSIsoft

Software
- OSIsoft PI Server 2014

Hardware
- IBM FlashSystem
- PI application servers
- PI analytics servers



OSIsoft determined the key to scaling would be data storage and retrieval performance. The response times of traditional storage could no longer keep pace with the accelerating volumes and velocity of data and, in fact, adding SSDs to these conventional arrays proved inadequate. This is why they turned to IBM FlashSystem.

You can read more this solution in our whitepaper.

Evidence-Based Decision Making: Using AWR and Statspack to Decide If Flash Is Right for You

Questions?

Mike Ault
mrault@us.ibm.com

76