AWR analysis

(o di come utilizzare l'AWR per condurre un'analisi di un database)

ITOUG Tech Day – Database Stream

Milano – 8 giugno 2017

Chi sono

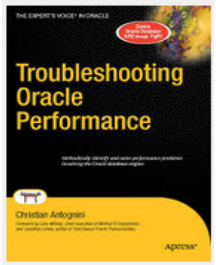
- CEO di ICTeam (dal 2006)
- Cofondatore di ICTeam (1999)
- Banca Popolare di Bergamo (1994-1999)
- Olivetti (1986-1994)
- Laurea in Scienze dell'Informazione a Milano (1986)

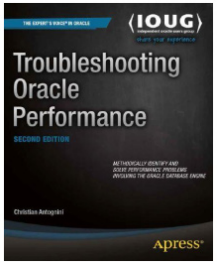
Performance geek

- la passione nasce fin dalla tesi sull'ottimizzazione delle performance di uno UNIX BSD (Olivetti 1985 - 1986)
- l'analisi delle performance Oracle inizia nel 1988 nella relazione Oracle-Unix
- sviluppo le funzionalità asynchronous I/O, post/wait, list I/O, priority ed affinity cpu, pipe in memory nel kernel per supportare Sybase, Informix ed Oracle nel benchmark TPC-A (1990 Olivetti è terza al mondo con Oracle nei sistemi x86, prima è Compaq)
- inizio a lavorare sulle performance più dal punto di vista Oracle (modellazione, SQL, parametri, …) dal 1994
- ancora oggi faccio il Performance Consultant (da 9.2 a 12.2)

Da sempre problem buster and solver (in campo informatico)

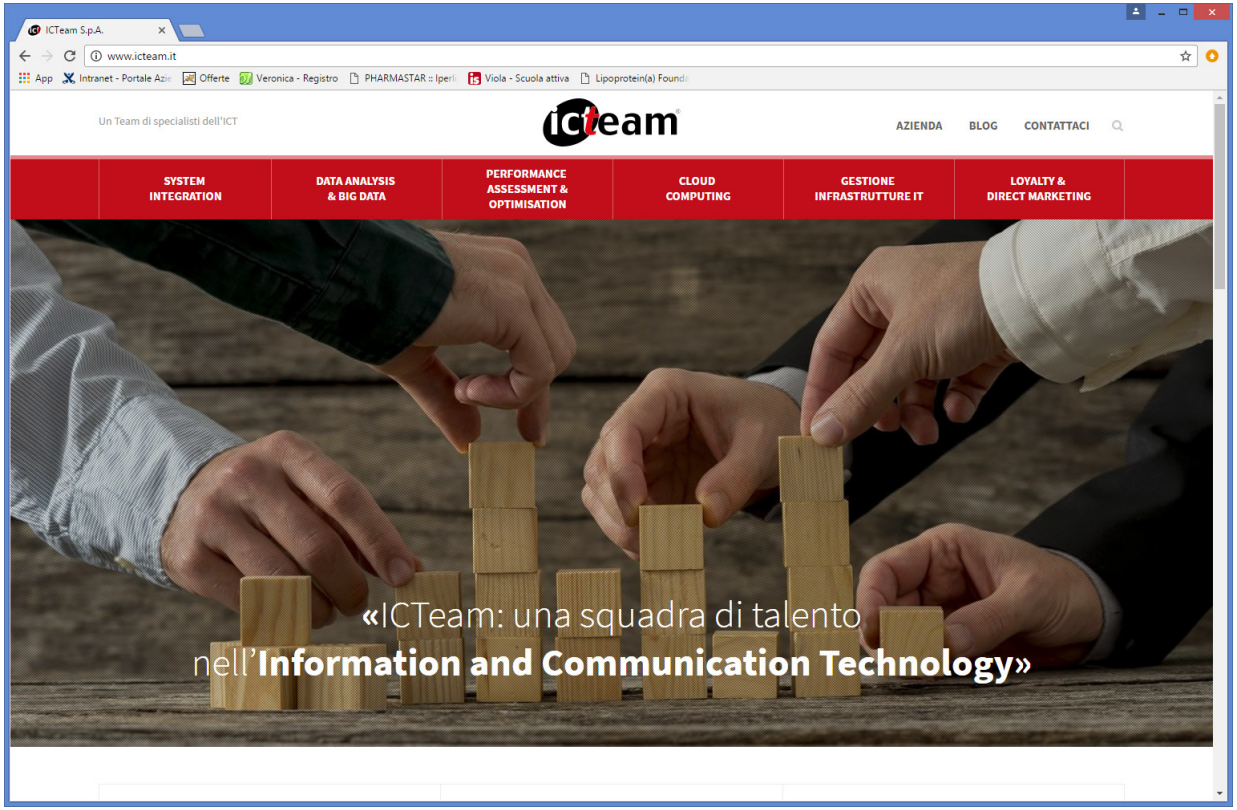
Ho avuto l'onore di essere uno dei technical reviewer del libro dell'amico Chistian Antognini





Chi è ICTeam

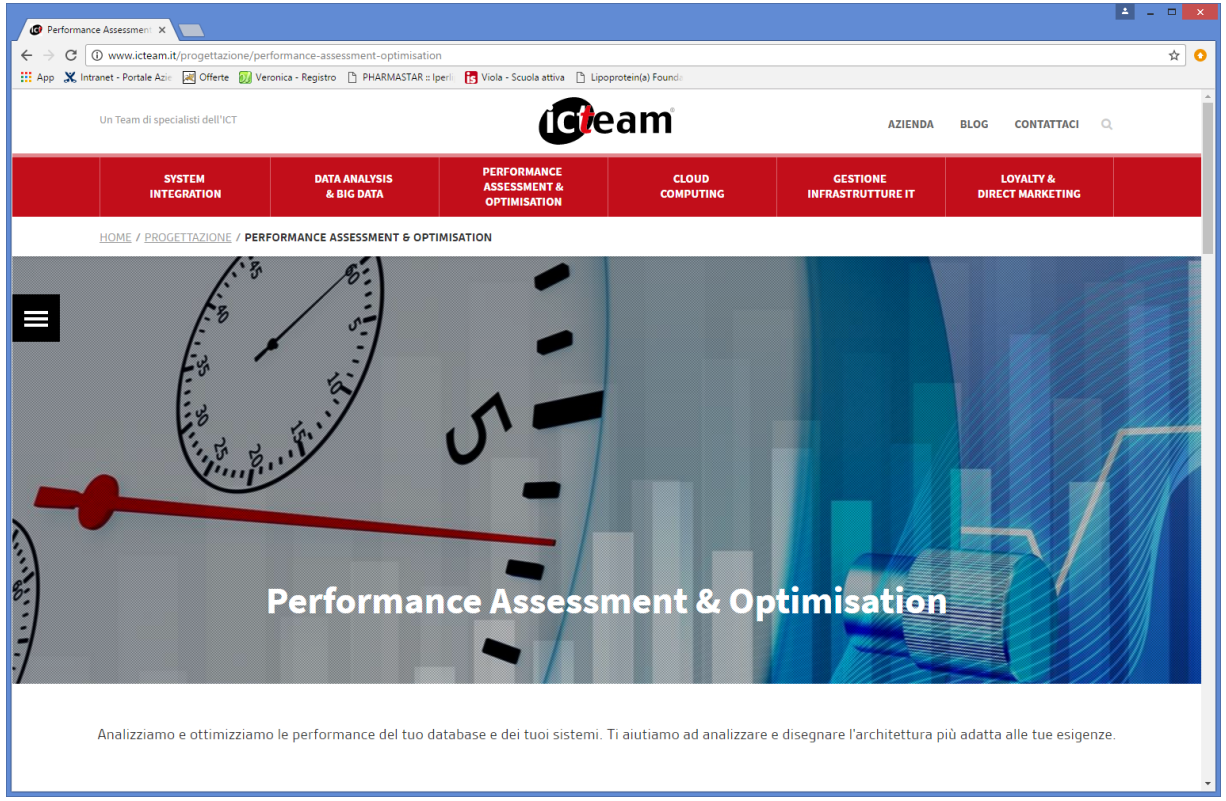
- ICTeam nasce nel 1999
- 6 soci fondatori tutti provenienti dalla Banca Popolare Bergamo che implementò un progetto più unico che raro: la completa rifondazione del sistema informativo senza mainframe, tutto su Unix ed Oracle



Sul mercato abbiamo messo la nostra competenza tecnica ed il nostro entusiasmo nel fare progetti e nel risolvere problemi.

Siamo 120 professionisti, di cui 114 tecnici e tecnici vogliamo restare!

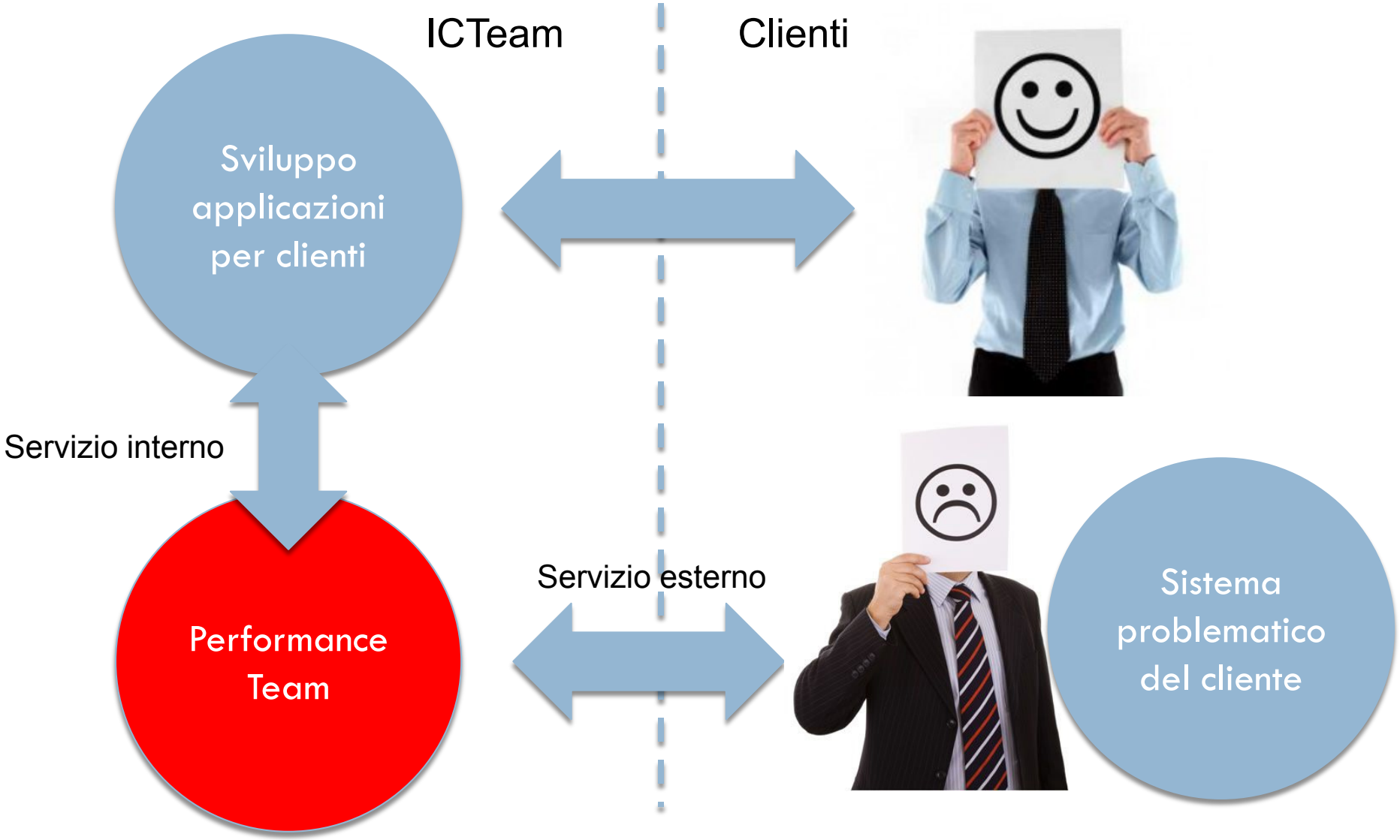
Performance team (penultima slide … di marketing)



performance_team@icteam.it

- dal 2013 abbiamo creato uno specifico gruppo di persone che si occupa esclusivamente di performance e prevalentemente di performance su Oracle
- siamo in grado di analizzare architetture complesse soffermandoci su tutte le componenti in gioco (hardware e software, di sistema ed applicativo)
- abbiamo esaminato alcuni dei più grandi ed importanti database Oracle d'Italia
- NEW: abbiamo iniziato ad occuparci anche di performance in ambito Big Data

Performance team - ruolo in ICTeam



Due parole sull'approccio *strategico* - 1

Non ho mai avuto l'ambizione di sapere tutto di Oracle (o in generale delle cose del mondo), ma di saperne abbastanza per risolvere problemi e di imparare sempre qualcosa dai problemi che mi sono stati sottoposti negli anni.

Solo in questi giorni, però, ho trovato casualmente (*ma davvero per caso?*) una frase che *probabilmente* descrive il mio approccio intuitivo ed implicito su cui non ho mai di fatto *riflettuto analiticamente*:

Al problem solver strategico non interessa conoscere le verità profonde e il perché delle cose, ma solo «come» funzionano e «come» farle funzionare nel miglior modo possibile.
La sua prima preoccupazione è quella di adattare le proprie conoscenze alle «realtà» parziali che si trova di volta in volta ad affrontare, mettendo a punto strategie fondate sugli obiettivi da raggiungere e in grado di adattarsi, passo dopo passo, all'evolversi della «realtà».

tratta da https://www.nardonegroup.org/problem-solving-strategico/ ed attribuita a Ernst von Glasersfeld (filosofo e cibernetico)

Due parole sull'approccio *strategico* - 2

Ancora, dal libro *Problem Solving strategico da tasca* di Giorgio Nardone a pagina 12 e 13 alcune indicazioni interessanti:

Il rapporto fra tecnologia e scienza è il medesimo di quello che intercorre tra la filosofia e la logica. La prima si interessa del «sapere», la seconda del «saper fare».

La conoscenza logica e tecnologica si differenzia da quella filosofica e scientifica in quanto saperi operativi e non speculativi. Difatti per saper fare non è necessario sapere tutto, ma solo ciò che è indispensabile al raggiungimento dello scopo.

[…]

Troppo spesso, di fronte ad un problema, si ha la tendenza a cercare la spiegazione piuttosto che la soluzione. La trappola è che la soluzione non necessita prima della spiegazione del problema, anzi sarà ciò che porterà al suo effettivo svelamento, mentre le spiegazioni sprovviste di prova empirica sono fuorvianti e basate sulla nostra conoscenza a priori.

Su cosa verterà questa presentazione

Di cosa vorrei parlare durante questo intervento?

Fondamentalmente di come si possa utilizzare l'AWR (Automatic Workload Repository) come strumento di indagine sulle performance di un database Oracle e dell'ecosistema in cui il database si trova.

Il tutto mostrando un esempio reale e recente, anche se basato su release 11g, ed utilizzando solo script del mondo *awr*sql* e *ash*sql*.

Non parlerò invece:

- di *teoria* dell'AWR (se non qualche cenno sulle basi del campionamento dei dati);
- dei singoli comandi con cui configurare l'AWR (esiste il manuale);
- di ADDM (Automatic Database Diagnostic Monitor) che sarebbe argomento a sé stante;
- di Enterprise Manager e quindi di modalità grafiche con cui interrogare i dati presenti nel repository;
- di Statspack, che tanta storia ha avuto nel mondo Oracle e che è ancora disponibile e da utilizzarsi quando non si dispone della licenza del Diagnostic Pack.

Lo strumento AWR - 1

Nella seconda edizione del libro Troubleshooting Oracle Performance, l'amico Christian Antognini ha inserito un capitolo su AWR e Statspack intitolandolo *Postmortem Analysis or Irreproducible Problems*.

Sembrerebbe che utilizzare l'AWR sia quasi come effettuare una autopsia … Si parla ovviamente di una autopsia virtuale (virtopsy) che è già diventata realtà.

Lo strumento AWR - 2

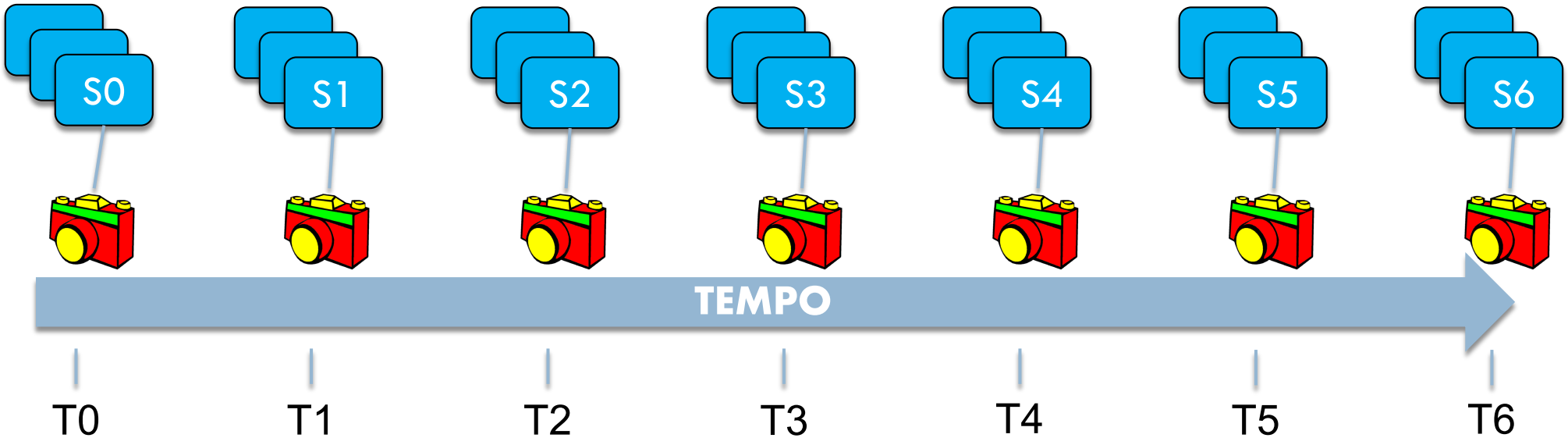
Personalmente sono un po' meno categorico sull'utilizzo dell'AWR:

- lo considero uno strumento empirico in grado di fornire un buon numero di informazioni di contesto e talvolta anche puntuali;
- utile per esaminare un paziente vivo che respira e che potrebbe ripetere, anche a breve, la prestazione da osservare;
- in grado di mantenere informazioni di trend per attività di confronto e capacity planning;
- fondamentale nel cercare il cambio piano di statement critici campionati;
- capace di fornire informazioni ulteriori a quelle del comportamento del database;
- ovviamente con dei limiti: non tutto può essere tracciato ed il campionamento può portare alla perdita di dati.

Repository AWR - 1

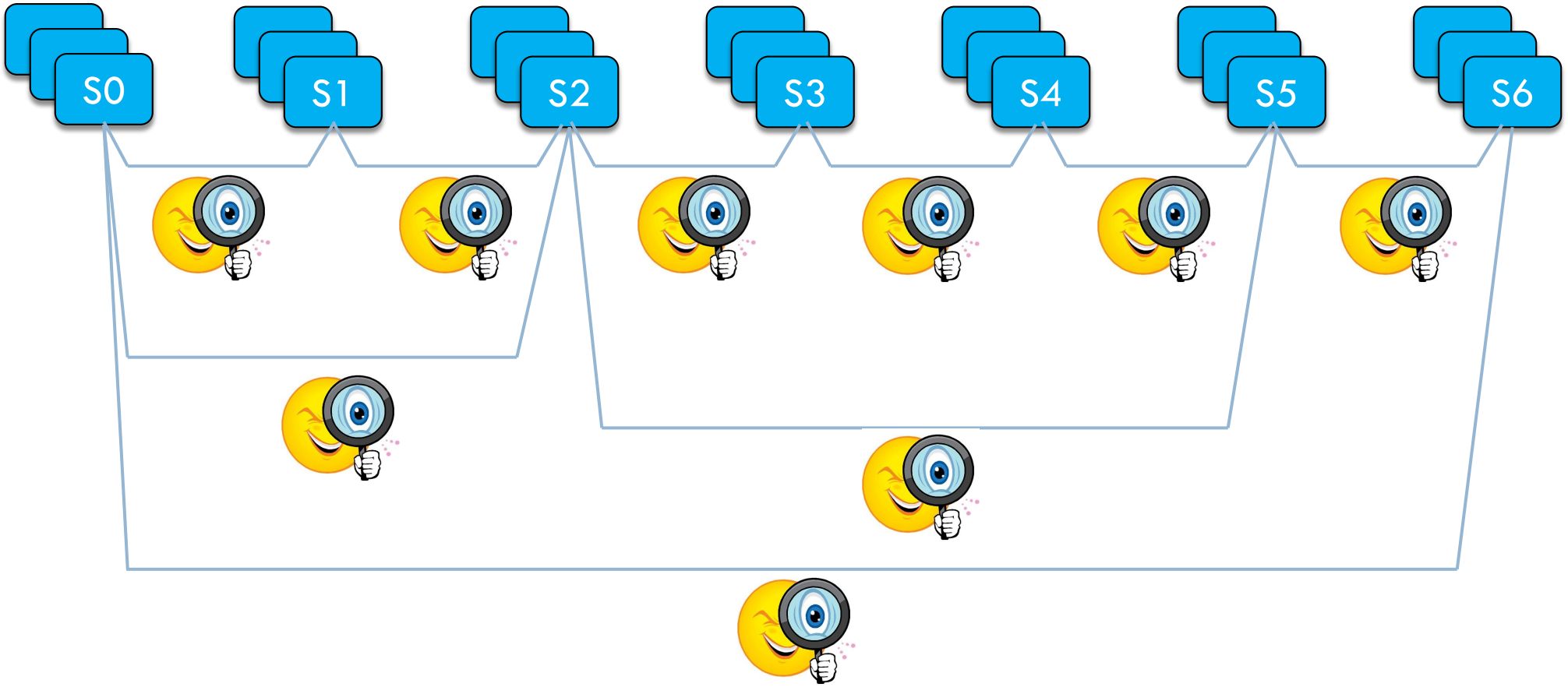
L'AWR (Automatic Workload Repository) si fonda sullo *scatto di fotografie* con frequenza definita (default 1h, minimo 10m).

Queste fotografie, definite snapshot, salvano la situazione delle view dinamiche (le v$, alcune vengono salvate totalmente, altre solo parzialmente anche in funzione di parametri di configurazione come i top SQL), per rendere possibile poi una analisi a posteriori di come si siano mossi i valori contenuti appunto nelle view dinamiche.



Repository AWR - 2

Le fotografie, attenzione non il film, possono essere confrontate per cercare di interpretare il workload avvenuto fra le due fotografie prescelte:



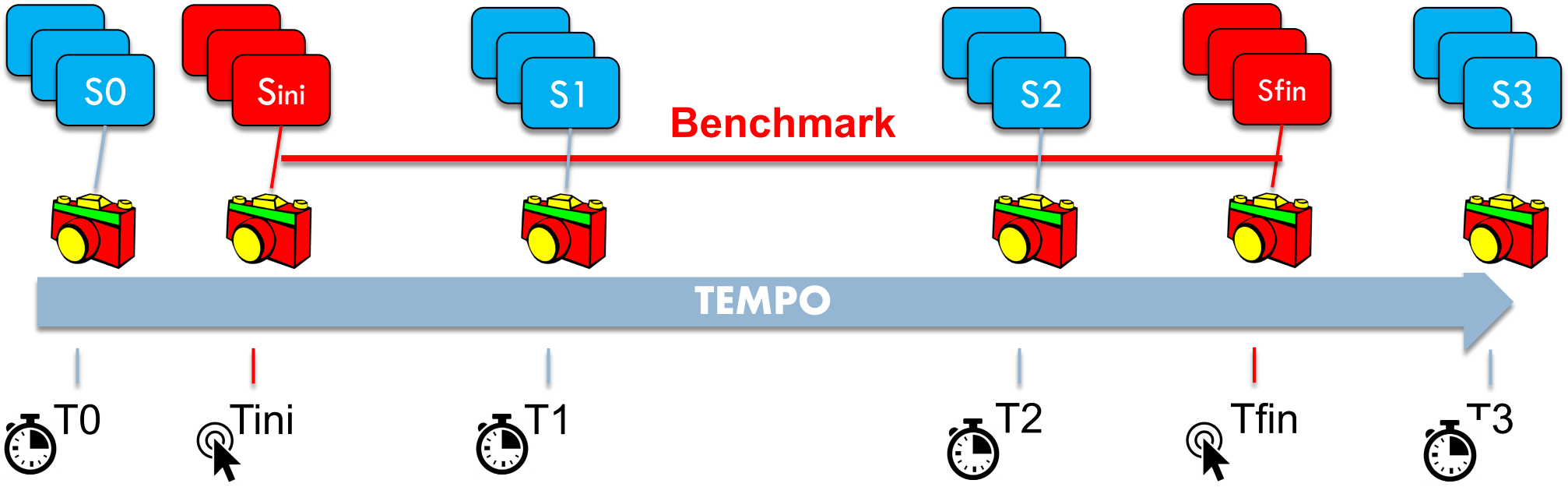
Repository AWR - 3

A fronte di una configurazione RAC, l'AWR salva congiuntamente le fotografie di tutti i nodi (instance) coinvolti con la frequenza definita:

Repository AWR - 4

Oltre che affidarsi allo scatto di fotografie automatiche a frequenza definita, è possibile scattare una foto, alias salvare uno snapshot, con specifica richiesta e relativa apposita nomenclatura come commento.

Ciò è utile, ad esempio, a fronte dell'esecuzione di benchmark o di batch particolari che si vuole monitorare macroscopicamente (o per cui in prima battuta non è possibile applicare strumenti di analisi tipicamente più invasivi e più precisi come il tracing delle sessioni).



Pro e contro dell'AWR

Alcuni pro nell'utilizzo dell'AWR:

- facilità di utilizzo
- bassa invasività
- permette analisi a posteriori (virtopsy)
- integrato con Enterprise Manager
- presenza di script (awr*.sql) per l'estrazione di dati testuali o in HTML
- mantenimento dei piani di esecuzione dei top statement con possibilità di intercettare variazioni di piano
- campionamento dello stato delle sessioni (active session history) e relativi script ash.sql)
- continuo improvement ed aumento di informazioni nelle varie release dalla 10g alla 12c
- ….

Alcuni contro:

- la quasi totalità dei valori ottenibili sono delle medie relative all'intervallo di campionamento, di fatto si perdono le informazioni di picco
- non tutti gli statement vengono campionati
- attenzione al licensing, l'AWR di default è attivo ma per poterlo utilizzare (package e repository) è necessaria la licenza del Diagnostic Pack!

Esempio di analisi con AWR

Il caso che verrà presentato ha le seguenti caratteristiche:

- è reale;
- deriva da una richiesta ufficiale molto recente da parte di un cliente;
- la consulenza è sostanzialmente in corso;
- l'obiettivo è quello di capire se vi siano interventi possibili (applicativi, sistemistici e/o infrastrutturali) per permettere di sopportare un carico maggiore dovuto da un incremento importante della customer base (almeno il 20% in più);
- l'infrastruttura attuale è data da un cluster in RAC di 2 server x86 a 40 core con Oracle 11.2.0.4;
- l'applicazione è la fatturazione con diversi step (durata complessiva circa una settimana). Lo step qui preso in esame è quello temporalmente più corposo: 7h di durata (intervallo 20 maggio 11-18);
- a detta del cliente gli statement SQL eseguiti dovrebbero essere già stati *tirati a lucido*.

WORKLOAD REPOSITORY REPORT (RAC)

Database Summary

Database				Snapshot Ids		Number of Instances		Number of Hosts		Report Total (minutes)	
Id	Name	RAC	Block Size	Begin	End	In Report	Total	In Report	Total	DB time	Elapsed time
1892790460	[redacted]	YES	8192	47776	47783	2	2	2	2	1,164.26	419.72

3x 6h

Database Instances Included In Report

- Listed in order of instance number, I#

I#	Instance	Host	Startup	Begin Snap Time	End Snap Time	Release	Elapsed Time(min)	DB time(min)	Up Time(hrs)	Avg Active Sessions	Platform
1	[redacted]		13-May-17 19:23	20-May-17 11:00	20-May-17 18:00	11.2.0.4.0	419.72	86.56	166.61	0.21	Linux x86 64-bit
2			13-May-17 19:23	20-May-17 11:00	20-May-17 18:00	11.2.0.4.0	419.72	1,077.70	166.60	2.57	Linux x86 64-bit

attività sul secondo nodo

Report Summary

Cache Sizes

- All values are in Megabytes
- Listed in order of instance number, I#
- End values displayed only if different from Begin values

	Memory Target		Sga Target		DB Cache		Shared Pool		Large Pool		Java Pool		Streams Pool		PGA Target		
I#	Begin	End	Begin	End	Begin	End	Begin	End	Begin	End	Begin	End	Begin	End	Begin	End	Log Buffer
1	61,440		39,936		26,880		8,960		1,024		1,792		256		21,504		139.65
2	61,440		39,936		27,136		8,960		1,280		1,792		256		21,504		139.65
Avg	61,440		39,936		27,008		8,960		1,152		1,792		256		21,504		139.65
Min	61,440		39,936		26,880		8,960		1,024		1,792		256		21,504		139.65
Max	61,440		39,936		27,136		8,960		1,280		1,792		256		21,504		139.65

Workload Repository Report RAC - 2 (awrgrpt)

OS Statistics By Instance

- Listed in order of instance number, I#
- End values are diplayed only if different from begin values

I#	Num CPUs	CPU Cores	CPU Sckts	Load Begin	Load End	% Busy	% Usr	% Sys	% WIO	% Idl	Busy Time (s)	Idle Time (s)	Total Time (s)	Memory (M)
1	80	40	4	4.09	2.49	1.89	1.70	0.15	0.27	98.11	37,972.38	1,976,216.20	2,014,188.58	516,124.13
2	80	40	4	1.36	1.31	1.70	1.49	0.18	1.33	98.30	34,146.81	1,979,978.68	2,014,125.49	516,124.13
Sum											72,119.19	3,956,194.88	4,028,314.07	

praticamente nulla! tutto idle

Back to Top

Time Model Statistics

- Time Model
- Time Model - % of DB time

Back to Top

Time Model

tempo speso in esecuzione statement

I#	DB time (s)	DB CPU (s)	SQL Exec Ela (s)	Parse Ela (s)	Hard Parse Ela (s)	PL/SQL Ela (s)	Java Ela (s)	bg time (s)	bg CPU (s)
1	5,193.76	1,198.66	3,566.72	562.79	540.89	80.11	0.00	2,018.27	654.89
2	64,662.05	5,355.28	61,248.58	413.12	306.34	79.89	0.00	5,264.09	713.22
Sum	69,855.81	6,553.94	64,815.30	975.90	847.24	160.00	0.00	7,282.36	1,368.12
Avg	34,927.91	3,276.97	32,407.65	487.95	423.62	80.00	0.00	3,641.18	684.06
Std	42,050.43	2,939.17	40,787.24	105.83	165.85	0.15	0.00	2,295.14	41.24

Workload Repository Report RAC - 3 (awrgrpt)

Foreground Wait Classes

I#	User I/O(s)	Sys I/O(s)	Other(s)	Applic (s)	Commit (s)	Network (s)	Concurcy (s)	Config (s)	Cluster (s)	DB CPU (s)	DB time
1	1,626.15	175.15	1,005.20	0.17	283.50	214.18	0.32	5.35	221.20	1,198.66	5,193.76
2	54,368.42	144.83	977.33	1.07	911.20	290.03	1.99	7.29	1,911.80	5,355.28	64,662.05
Sum	55,994.58	319.97	1,982.52	1.24	1,194.69	504.20	2.31	12.64	2,132.99	6,553.94	69,855.81
Avg	27,997.29	159.99	991.26	0.62	597.35	252.10	1.15	6.32	1,066.50	3,276.97	34,927.91
Std	37,294.42	21.44	19.71	0.63	443.85	53.63	1.18	1.38	1,195.44	2,939.17	42,050.43

Quasi 1h e 30m di cpu su 7h di elapsed non è molto

Foreground Wait Classes - % of DB time

- % of Total DB time - instance DB time as a percentage of the cluster-wide total DB time

I#	User I/O	Sys I/O	Other	Applic	Commit	Network	Concurcy	Config	Cluster	DB CPU	% Total DB time
1	31.31	3.37	19.35	0.00	5.46	4.12	0.01	0.10	4.26	23.08	7.43
2	84.08	0.22	1.51	0.00	1.41	0.45	0.00	0.01	2.96	8.28	92.57
Avg	57.70	1.80	10.43	0.00	3.43	2.29	0.00	0.06	3.61	15.68	

Il tempo è speso quasi tutto in User I/O !!!

Complessivamente il tempo speso nel database sul nodo 2 è di 15h e qualche minuto. Non poco, ma neanche molto a fronte di 7h di elapsed.

Workload Repository Report RAC - 4 (awrgrpt)

	Wait		Event		Wait Time			Summary Avg Wait Time (ms)				
I#	Class	Event	Waits	%Timeouts	Total(s)	Avg(ms)	%DB time	Avg	Min	Max	Std Dev	Cnt
*	User I/O	direct path read	2,556,427	0.00	43,079.50	16.85	61.67	10.20	3.56	16.85	9.40	2
	User I/O	db file sequential read	6,177,274	0.00	10,944.44	1.77	15.67	1.36	0.86	1.86	0.71	2
		DB CPU			6,553.94		9.38					2
	System I/O	db file parallel write	1,320,835	0.00	2,104.17	1.59	3.01	2.41	1.53	3.29	1.25	2
	Other	enq: IV - contention	326,570	0.48	1,678.42	5.14	2.40	5.12	4.96	5.29	0.23	2
	Commit	log file sync	228,262	0.00	1,194.88	5.23	1.71	4.89	3.94	5.83	1.33	2
	User I/O	db file scattered read	293,061	0.00	1,126.56	3.84	1.61	3.44	2.73	4.14	1.00	2
	Administrative	Backup: MML write backup piece	71,597	0.00	1,094.70	15.29	1.57	14.04	7.01	21.07	9.94	2
	Cluster	gc current grant busy	211,095	0.00	1,036.94	4.91	1.48	4.16	3.39	4.93	1.09	2
	System I/O	log file parallel write	360,846	0.00	833.06	2.31	1.19	2.26	1.85	2.67	0.58	2
1		DB CPU			1,198.66		23.08					
	User I/O	db file scattered read	230,860	0.00	956.75	4.14	18.42					
	Other	enq: IV - contention	180,239	0.42	952.99	5.29	18.35					
	User I/O	db file sequential read	551,047	0.00	471.31	0.86	9.07					
	System I/O	log file parallel write	159,424	0.00	295.16	1.85	5.68					
	Commit	log file sync	71,914	0.00	283.55	3.94	5.46					
	System I/O	control file sequential read	563,693	0.00	237.86	0.42	4.58					
	Administrative	Backup: MML write backup piece	29,428	0.00	206.29	7.01	3.97					
	System I/O	db file parallel write	44,930	0.00	148.03	3.29	2.85					
	User I/O	direct path read temp	60,243	0.00	137.15	2.28	2.64					
2	User I/O	direct path read	2,556,204	0.00	43,078.70	16.85	66.62					
	User I/O	db file sequential read	5,626,227	0.00	10,473.13	1.86	16.20					
		DB CPU			5,355.28		8.28					
	System I/O	db file parallel write	1,275,905	0.00	1,956.13	1.53	3.03					
	Cluster	gc current grant busy	208,303	0.00	1,027.49	4.93	1.59					
	Commit	log file sync	156,348	0.00	911.33	5.83	1.41					
	Administrative	Backup: MML write backup piece	42,169	0.00	888.41	21.07	1.37					
	Other	enq: IV - contention	146,331	0.55	725.43	4.96	1.12					
	System I/O	log file parallel write	201,422	0.00	537.90	2.67	0.83					
	Other	DFS lock handle	72,386	52.03	433.42	5.99	0.67					

User I/O complessivo che cuba per il 77%

Il problema sembrerebbe essere stato individuato. Alto DB Time e tempi di risposta non ottimali per le letture in direct path.

Workload Repository Report RAC - 5 (awrgrpt)

SQL ordered by Elapsed Time (Global)

- Captured SQL account for 89.7% of Total DB Time (s): 69,856
- Captured PL/SQL account for 19.6% of Total DB Time (s): 69,856

sufficientemente alto rispetto al DB Time

	Total								Percentage of Total							
SQL Id	Elapsed (s)	CPU (s)	IOWait (s)	Gets	Reads	Rows	Cluster (s)	Execs	DB time	DB CPU	IO Wait	Gets	Reads	Cluster	Execs	SQL Text
830y0r6h7qg68	44,609.86	1,755.39	42,819.51	86,086,883	85,490,845	7,215,707	4.29	1	63.86	26.78	0.76	23.82	83.48	0.19	0.00	INSERT INTO ZTMPIDRETTIFICHE (...
bxhr6s8516643	8,153.35	809.32	5,993.46	54,967,566	3,671,181	347	1,463.49	347	11.67	12.35	0.11	15.21	3.58	66.26	0.00	DECLARE inaccountid_TARGET VA...
cwx2j4bsf36nx	5,052.98	403.53	4,044.22	35,798,867	2,326,097	3,378,569	685.96	155	7.23	6.16	0.07	9.90	2.27	31.06	0.00	INSERT INTO INVOICEDETAIL ID (...
gkk3ru9t4czsm	2,295.59	120.64	2,221.51	1,654,071	1,231,551	1	3.05	1	3.29	1.84	0.04	0.46	1.20	0.14	0.00	DECLARE inCycle_TARGET VARCHA...
d2rp6c9wtdy3d	2,295.52	120.64	2,221.48	1,654,029	1,231,546	7,564	3.03	1	3.29	1.84	0.04	0.46	1.20	0.14	0.00	DELETE FROM INVOICEPROCESS IP ...
ajfgv22avyzh6	1,844.71	99.03	1,781.27	1,233,146	944,697	0	0.12	320	2.64	1.51	0.03	0.34	0.92	0.01	0.00	SELECT t0.UIDTRANSACTION, t0.A...
0bhagqcj4txfj	1,080.13	344.17	634.28	17,098,691	1,796,612	1	126.95	1	1.55	5.25	0.01	4.73	1.75	5.75	0.00	DECLARE job BINARY_INTEGER := ...
86v7rpcmafs1f	895.32	61.73	811.50	3,493,941	2,952,829	1	30.39	1	1.28	0.94	0.01	0.97	2.88	1.38	0.00	BEGIN MONITORAGGIO_PIANO_SENDM...
cp637yfy0z5q6	883.60	222.75	587.33	3,294,076	1,593,443	12,022,172	98.40	1	1.26	3.40	0.01	0.91	1.56	4.45	0.00	INSERT /*+ BYPASS_RECURSIVE_CH...
7r5uf5nnhj8nc	878.46	59.43	802.26	3,232,984	2,925,813	1	25.40	1	1.26	0.91	0.01	0.89	2.86	1.15	0.00	DELETE FROM ZTMPFAUNEGFUORIFAT...

%

1

Il problema!
Un solo statement, una sola esecuzione per il 64% del DB time, il 76% dell'I/O wait, l'83% delle physical read !!!

Workload Repository SQL Report - 1 (awrsqrpi)

WORKLOAD REPOSITORY SQL Report

Snapshot Period Summary

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
[redacted]	1892790460	[redacted]	2	13-May-17 19:05	11.2.0.4.0	YES

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	47776	20-May-17 11:00:18	71	1.6
End Snap:	47783	20-May-17 18:00:01	66	1.5
Elapsed:		419.71 (mins)		
DB Time:		1,077.70 (mins)		

SQL Summary

SQL Id	Elapsed Time (ms)	Module	Action	SQL Text
830y0r6h7qg68	44,609,860	JDBC Thin Client		INSERT INTO ZTMPIDRETTIFICHE (UIDFACILITY, TARIFFYEAR, STARTREADTIME, ...

Back to Top

SQL ID: 830y0r6h7qg68

- 1st Capture and Last Capture Snap IDs refer to Snapshot IDs witin the snapshot range
- INSERT INTO ZTMPIDRETTIFICHE (UIDFACILITY,TARIFFYEAR,STARTREADTIME,STO...

#	Plan Hash Value	Total Elapsed Time(ms)	Executions	1st Capture Snap ID	Last Capture Snap ID
1	3933163802	44,609,860	1	47777	47777

Ma …. lo statement sembra essere stato campionato solo in uno snapshot !!!

Workload Repository SQL Report – 2 (awrsqrpi)

Plan Statistics

- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100

Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	44,609,860	44,609,859.75	68.99
CPU Time (ms)	1,755,388	1,755,388.11	32.78
Executions	1		
Buffer Gets	86,086,883	86,086,883.00	27.65
Disk Reads	85,490,845	85,490,845.00	87.67
Parse Calls	321	321.00	0.01
Rows	7,215,707	7,215,707.00	
User I/O Wait Time (ms)	42,819,507		
Cluster Wait Time (ms)	4,293		
Application Wait Time (ms)	9		
Concurrency Wait Time (ms)	71,950		
Invalidations	0		
Version Count	2		
Sharable Mem(KB)	124		

321 parse per 1 sola esecuzione è decisamente anomalo.
La spiegazione più probabile è che partano troppi processi parallel ed in effetti nel testo dello statement l'hint di parallel non indica il degree … quindi vale il default di 160 processi per passo, con l'hash unique 320 !!!

SQL Id	SQL Text
830y0r6h7qg68	INSERT INTO ZTMPIDRETTIFICHE (UIDFACILITY, TARIFFYEAR, STARTREADTIME, STOPREADTIME, UIDBA, UIDINVOICE) SELECT /*+ parallel(id) full(idr) full(id) use_hash(id idr) */ ID.UIDFACILITY, ID.TARIFFYEAR, ID.STARTREADTIME, ID.STOPREADTIME, ID.UIDBA, ID.UIDINVOICE FROM INVOICEDETAIL ID INNER JOIN (SELECT DISTINCT UIDFACILITY, TARIFFYEAR FROM ZTMPIDSRETTIFICHE) IDR ON ID.UIDFACILITY = IDR.UIDFACILITY WHERE ID.TARIFFYEAR = IDR.TARIFFYEAR AND ID.UIDBA NOT IN (:B6 , :B5 , :B4 , :B3 , :B2) AND ID.UIDINVIDTYPE = :B1

Workload Repository SQL Report - 3 (awrsqrpi)

Execution Plan

Id	Operation	Name	Rows	Bytes	Temp Spc	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	INSERT STATEMENT					123K(100)						
1	LOAD TABLE CONVENTIONAL											
2	PX COORDINATOR											
3	PX SEND QC (RANDOM)	:TQ10002	25M	1726M		123K (2)	00:24:39			Q1,02	P->S	QC (RAND)
4	VIEW	VM_NWVW_1	25M	1726M		123K (2)	00:24:39			Q1,02	PCWP	
5	HASH UNIQUE		25M	1504M	2083M	123K (2)	00:24:39			Q1,02	PCWP	
6	PX RECEIVE		25M	1504M		123K (2)	00:24:38			Q1,02	PCWP	
7	PX SEND HASH	:TQ10001	25M	1504M		123K (2)	00:24:38			Q1,01	P->P	HASH
8	HASH JOIN		25M	1504M		123K (2)	00:24:38			Q1,01	PCWP	
9	BUFFER SORT									Q1,01	PCWC	
10	PX RECEIVE		2236K	21M		5183 (2)	00:01:03			Q1,01	PCWP	
11	PX SEND BROADCAST LOCAL	:TQ10000	2236K	21M		5183 (2)	00:01:03				S->P	BCST LOCAL
12	TABLE ACCESS FULL	ZTMPIDSRETTIFICHE	2236K	21M		5183 (2)	00:01:03					
13	PX BLOCK ITERATOR		1254M	59G		117K (2)	00:23:35	1	17	Q1,01	PCWC	
14	TABLE ACCESS FULL	INVOICEDETAIL	1254M	59G		117K (2)	00:23:35	1	17	Q1,01	PCWP	

Il piano di esecuzione è comunque adeguato, certo l'impegno di così tanti processi parallel (320) appare un po' esagerato, soprattutto perché vi è il broadcast della tavola ZTMPIDSRETTIFICHE, stimata in poco più di 2 milioni di record, verso 160 processi.
Ma se l'esecuzione è confinata all'interno di un solo snapshot (quindi una sola ora), non può essere responsabile della durata di 7h della fatturazione!

Active Session History SQLID Report (ashrpti)

	Sample Time	Data Source
Analysis Begin Time:	20-May-17 11:00:00	DBA_HIST_ACTIVE_SESS_HISTORY in AWR snapshot 47776
Analysis End Time:	20-May-17 18:00:00	DBA_HIST_ACTIVE_SESS_HISTORY in AWR snapshot 47783
Elapsed Time:	420.0 (mins)	
Sample Count:	4,387	
Average Active Sessions:	1.74	
Avg. Active Session per CPU:	0.01	
Report Target:	SQL_ID like '830y0r6h7qg68'	58.3% of total database activity

L'intervallo analizzato è completo, dalle 11 alle 18.

Activity Over Time

- Analysis period is divided into smaller time slots
- Top 3 events are reported in each of those slots
- 'Slot Count' shows the number of ASH samples in that slot
- 'Event Count' shows the number of ASH samples waiting for that event in that slot
- '% Event' is 'Event Count' over all ASH samples in the analysis period

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
11:00:00 (5.0 min)	183	direct path read	162	3.69
		CPU + Wait for CPU	18	0.41
		os thread startup	3	0.07
11:05:00 (5.0 min)	3,328	direct path read	3,229	73.60
		CPU + Wait for CPU	99	2.26
11:10:00 (5.0 min)	876	direct path read	827	18.85
		CPU + Wait for CPU	48	1.09
		gc current multi block request	1	0.02

L'attività dello statement è però confinata solo fra le 11:00 e le 11:15.
Anche i dati di ASH confermano quindi che l'attività più *pesante* è limitata nel tempo e non può essere causa della durata di 7h della fatturazione. **Tutto da rifare!**

Workload Repository Report RAC - 1 (awrgrpt)

Primo step ripartire dal dato macroscopico escludendo la prima ora campionata (11-12). Obiettivo: controllare se la distribuzione del carico fra i due nodi cambi o resti più pesante sul secondo nodo come i dati in effetti continuano a dimostrare:

Time Model

I#	DB time (s)	DB CPU (s)	SQL Exec Ela (s)	Parse Ela (s)	Hard Parse Ela (s)	PL/SQL Ela (s)	Java Ela (s)	bg time (s)	bg CPU (s)
1	3,778.14	918.46	2,265.60	558.65	540.17	62.50	0.00	1,766.69	574.99
2	16,489.51	3,089.65	13,398.70	359.06	288.16	63.51	0.00	4,714.20	639.20
Sum	20,267.64	4,008.11	15,664.29	917.71	828.33	126.01	0.00	6,480.89	1,214.20
Avg	10,133.82	2,004.06	7,832.15	458.85	414.16	63.01	0.00	3,240.44	607.10
Std	8,988.30	1,535.27	7,872.29	141.13	178.20	0.71	0.00	2,084.20	45.40

Ma ora la domanda più complessa è la seguente: il carico sul nodo 2 come si distribuisce nel tempo? Ed è tale da risultare critico per l'esecuzione del billing?
Il DB time è di 16.489 secondi, vale a dire 5h a fronte di 6h di osservazione.
E' importante comprendere la distribuzione del DB time sia temporalmente che come impegno nel database.

Workload Repository Report RAC - 2 (awrgrpt)

Lo scenario di distribuzione degli eventi si presenta diversamente fra le 12 e le 18:

	Wait		Event		Wait Time			Summary Avg Wait Time (ms)				
I#	Class	Event	Waits	%Timeouts	Total(s)	Avg(ms)	%DB time	Avg	Min	Max	Std Dev	Cnt
*	User I/O	db file sequential read	4,877,031	0.00	8,419.31	1.73	41.54	1.28	0.72	1.85	0.80	2
		DB CPU			4,008.11		19.78					2
	System I/O	db file parallel write	1,300,062	0.00	2,020.95	1.55	9.97	2.34	1.50	3.18	1.18	2
	Other	enq: IV - contention	325,307	0.43	1,672.51	5.14	8.25	5.12	4.96	5.29	0.24	2
	Commit	log file sync	201,038	0.00	1,063.18	5.29	5.25	4.94	4.09	5.79	1.21	2
	Cluster	gc current grant busy	193,923	0.00	957.40	4.94	4.72	4.12	3.27	4.96	1.19	2
	Administrative	Backup: MML write backup piece	60,066	0.00	873.69	14.55	4.31	14.40	7.01	21.78	10.45	2
	System I/O	log file parallel write	311,708	0.00	720.98	2.31	3.56	2.24	1.86	2.62	0.54	2
	Other	DFS lock handle	79,915	52.82	458.40	5.74	2.26	5.33	4.66	5.99	0.94	2
	Cluster	gc current block 2-way	128,922	0.00	398.34	3.09	1.97	3.10	3.08	3.11	0.02	2
1	Other	enq: IV - contention	179,695	0.37	950.76	5.29	25.16					
		DB CPU			918.46		24.31					
	User I/O	db file sequential read	529,932	0.00	381.39	0.72	10.09					
	Commit	log file sync	59,575	0.00	243.58	4.09	6.45					
	System I/O	log file parallel write	126,987	0.00	236.21	1.86	6.25					
	Administrative	Backup: MML write backup piece	29,428	0.00	206.29	7.01	5.46					
	System I/O	control file sequential read	482,128	0.00	193.07	0.40	5.11					
	User I/O	direct path read temp	60,243	0.00	137.15	2.28	3.63					
	System I/O	db file parallel write	41,001	0.00	130.24	3.18	3.45					
	User I/O	db file scattered read	37,016	0.00	113.47	3.07	[illegible]					
2	User I/O	db file sequential read	4,347,099	0.00	8,037.92	1.85	48.75					
		DB CPU			3,089.65		18.74					
	System I/O	db file parallel write	1,259,061	0.00	1,890.71	1.50	11.47					
	Cluster	gc current grant busy	191,603	0.00	949.81	4.96	5.76					
	Commit	log file sync	141,463	0.00	819.60	5.79	4.97					
	Other	enq: IV - contention	145,612	0.49	721.75	4.96	4.38					
	Administrative	Backup: MML write backup piece	30,638	0.00	667.39	21.78	4.05					
	System I/O	log file parallel write	184,721	0.00	484.76	2.62	2.94					
	Other	DFS lock handle	64,582	51.90	386.93	5.99	2.35					
	User I/O	db file parallel read	57,193	0.00	375.51	6.57	2.28					

Il 48,75% del DB time è ancora User I/O ma lettura di singolo blocco.
Il tempo di risposta medio è buono 1,85ms e lascia pensare ad accessi alla cache dello storage

ASH Report- 1 (ashrpt)

Dal punto di vista della concentrazione delle attività nell'arco temporale, la situazione è abbastanza piatta.
Nel repository AWR vengono mantenuti i campionamenti ASH (Active Session History) di ogni 10 secondi. Trattasi di dati statisticamente validi.
Generando il report ASH con raggruppamento dell'informazione ogni 5 minuti, potenzialmente si hanno 30 campionamenti salvati. Il numero di slot indica il numero di sessioni attive presenti al momento dei campionamenti.
Gli slot registrati non superano mai il valore di 89, quindi in teoria mai più di 3 sessioni attive contemporaneamente per 5 minuti sui due nodi …

Activity Over Time

- Analysis period is divided into smaller time slots
- Top 3 events are reported in each of those slots
- 'Slot Count' shows the number of ASH samples in that slot
- 'Event Count' shows the number of ASH samples waiting for that event in that slot
- '% Event' is 'Event Count' over all ASH samples in the analysis period

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
11:15:00 (5.0 min)	89	db file sequential read	26	0.88
		Backup: MML write backup piece	22	0.75
		CPU + Wait for CPU	17	0.58
11:20:00 (5.0 min)	44	db file sequential read	28	0.95
		CPU + Wait for CPU	11	0.37
		DFS lock handle	2	0.07

ASH Report – 2 (ashrpti)

Verificando le top session ne spicca una: la 1715 che è presente come attiva per il 44,73% dei campionamenti, considerando entrambi i nodi:

Sid, Serial#	% Activity	Event	% Event	User	Program	# Samples Active	XIDs
1715,17041	44.73	db file sequential read	32.77	SCHEITG	JDBC Thin Client	967/2,430 [40%]	101
		CPU + Wait for CPU	4.47			132/2,430 [5%]	34
		gc current grant busy	3.69			109/2,430 [4%]	8
1902,33071	3.66	CPU + Wait for CPU	1.08	USER_MNGR_FATITG_01	oracle@igsv1851 (J000)	32/2,430 [1%]	1
1753,36699	2.58	db file sequential read	0.95	USER_MNGR_FATITG_01	oracle@igsv1850 (J000)	28/2,430 [1%]	1
1141, 1	2.03	log file parallel write	1.93	SYS	oracle@igsv1851 (LGWR)	57/2,430 [2%]	0
2892,21121	1.76	log file sync	0.61	APPITG	JDBC Thin Client	18/2,430 [1%]	0

Considerando solo il secondo nodo la percentuale diviene il 55%. La sessione compare in 967 slot, approssimandone ognuno con 10 secondi, ciò significa 9.670 secondi di attività: 2h e 41m su 6h e 45m … il grosso è fuori dal database

Sid, Serial#	% Activity	Event	% Event	User	Program	# Samples Active	XIDs
1715,17041	55.32	db file sequential read	40.53	SCHEITG	JDBC Thin Client	967/2,430 [40%]	101
		CPU + Wait for CPU	5.53			132/2,430 [5%]	34
		gc current grant busy	4.57			109/2,430 [4%]	8
1902,33071	4.53	CPU + Wait for CPU	1.34	USER_MNGR_FATITG_01	oracle@igsv1851 (J000)	32/2,430 [1%]	1
		db file sequential read	1.22			29/2,430 [1%]	1
1141, 1	2.51	log file parallel write	2.39	SYS	oracle@igsv1851 (LGWR)	57/2,430 [2%]	0
2892,21121	2.18	log file sync	0.75	APPITG	JDBC Thin Client	18/2,430 [1%]	0
2858, 653	1.76	Backup: MML write backup piece	1.68	SYS	rman@igsv1850 (TNS V1-V3)	40/2,430 [2%]	0

ASH Report Session ID - 1 (ashrpti)

Generando il report ASH fissando la sessione specifica si possono esaminare alcuni comportamenti macro della sessione (sempre statisticamente parlando e considerando l'intervallo 12 – 18)

Top User Events

Event	Event Class	% Event	Avg Active Sessions
db file sequential read	User I/O	69.47	0.34
CPU + Wait for CPU	CPU	11.15	0.05
gc current grant busy	Cluster	9.07	0.04
db file parallel read	User I/O	3.88	0.02
gc current grant 2-way	Cluster	1.32	0.01

70% in attesa di letture di singolo blocco

Top SQL Command Types

- 'Distinct SQLIDs' is the count of the distinct number of SQLIDs with the given SQL Command Type found over all the ASH samples in the analysis period

SQL Command Type	Distinct SQLIDs	% Activity	Avg Active Sessions
INSERT	5	60.96	0.30
SELECT	9	30.91	0.15
UPDATE	3	7.09	0.03

tipologia statement campionati

Top PL/SQL Procedures

- 'PL/SQL entry subprogram' represents the application's top-level entry-point(procedure, function, trigger, package initialization or RPC call) into PL/SQL.
- 'PL/SQL current subprogram' is the pl/sql subprogram being executed at the point of sampling . If the value is 'SQL', it represents the percentage of time spent executing SQL for the particular plsql entry subprogram

PLSQL Entry Subprogram	% Activity	PLSQL Current Subprogram	% Current
USER_MNGR_FATITG_01.LSP_PROCESS_INVOICE_BY_ACCOUNT	74.86	SQL	74.76

lanciati anche via PL/SQL

ASH Report Session ID - 2 (ashrpti)

Sempre dallo stesso report è possibile individuare gli statement a maggior impatto

Top SQL with Top Events

SQL ID	Planhash	Sampled # of Executions	% Activity	Event	% Event	Top Row Source	% RwSrc	SQL Text
cwx2j4bsf36nx	3672077813	62	55.48	db file sequential read	42.06	LOAD TABLE CONVENTIONAL	32.89	INSERT INTO INVOICEDETAIL ID (...
				gc current grant busy	6.33	LOAD TABLE CONVENTIONAL	6.33	
				CPU + Wait for CPU	4.35	LOAD TABLE CONVENTIONAL	3.40	
ajfgv22avyzh6	3346443437	75	17.96	db file sequential read	17.39	TABLE ACCESS - BY INDEX ROWID	13.14	SELECT t0.UIDTRANSACTION, t0.A...
bmwfnbxmfrmdp	3372529928	32	11.72	db file sequential read	4.54	TABLE ACCESS - BY INDEX ROWID	4.06	(SELECT /*+index(invs)*/ INVS...
				db file parallel read	3.88	TABLE ACCESS - BY INDEX ROWID	3.88	
				CPU + Wait for CPU	1.70	HASH JOIN - OUTER	0.85	
a3vc2rpga0rhy	4053969308	57	5.39	gc current grant busy	2.74	UPDATE	2.74	UPDATE BE_TB_CR_EVENT SET IS_B...
				db file sequential read	1.51	TABLE ACCESS - BY INDEX ROWID	1.42	
d2yd7m5yrjyu4		49	4.82	db file sequential read	3.59	** Row Source Not Available **	3.59	INSERT INTO LSTRANSACTION (UID...

Back to Top SQL
Back to Top

Top SQL with Top Row Sources

SQL ID	PlanHash	Sampled # of Executions	% Activity	Row Source	% RwSrc	Top Event	% Event	SQL Text
cwx2j4bsf36nx	3672077813	62	55.48	LOAD TABLE CONVENTIONAL	43.48	db file sequential read	32.89	INSERT INTO INVOICEDETAIL ID (...
				TABLE ACCESS - BY INDEX ROWID	11.53	db file sequential read	9.17	
ajfgv22avyzh6	3346443437	75	17.96	TABLE ACCESS - BY INDEX ROWID	13.61	db file sequential read	13.14	SELECT t0.UIDTRANSACTION, t0.A...
				INDEX - RANGE SCAN	4.35	db file sequential read	4.25	
bmwfnbxmfrmdp	3372529928	32	11.72	TABLE ACCESS - BY INDEX ROWID	9.64	db file sequential read	4.06	(SELECT /*+index(invs)*/ INVS...
a3vc2rpga0rhy	4053969308	57	5.39	UPDATE	3.21	gc current grant busy	2.74	UPDATE BE_TB_CR_EVENT SET IS_B...
				TABLE ACCESS - BY INDEX ROWID	1.61	db file sequential read	1.42	
d2yd7m5yrjyu4		49	4.82	** Row Source Not Available **	4.82	db file sequential read	3.59	INSERT INTO LSTRANSACTION (UID...

ASH Report Session ID - 3 (ashrpti)

E gli oggetti maggiormente coinvolti

Top DB Objects

- With respect to Application, Cluster, User I/O and buffer busy waits only.

Object ID	% Activity	Event	% Event	Object Name (Type)	Tablespace
483706	33.18	db file sequential read	31.85	USER_MNGR_FATITG_01.IX_INVDET_FAC (INDEX)	F1_TBS_INDEX_08
		gc current grant busy	1.23		
85385	20.60	db file sequential read	13.23	USER_MNGR_FATITG_01.INVOICEDETAILSTAGE (TABLE)	F1_TBS_DATA_12
		db file parallel read	3.88		
		gc cr block 2-way	1.04		
84621	13.14	db file sequential read	13.14	USER_MNGR_FATITG_01.LSTRANSACTION (TABLE)	F1_TBS_DATA_01
310378	4.63	gc current grant busy	2.74	USER_MNGR_FATITG_01.BE_TB_CR_EVENT (TABLE)	F1_TBS_DATA_01
		db file sequential read	1.42		
87995	4.25	db file sequential read	4.25	USER_MNGR_FATITG_01.IX_TRANS_CANCELTIME (INDEX)	F1_TBS_INDEX_01

L'indice IX_INVDET_FAC è indice della tavola INVOICEDETAIL, su cui avvengono le INSERT SELECT. Non è il solo indice, vi è anche quello di Primary Key, ma evidentemente il peso del mantenere questo indice è superiore in termini di I/O indotto. La tavola per inciso contiene più di 1 miliardo e 700 milioni di righe …

Workload Repository SQL Report - 1 (awrsqrpi)

Lo statement di insert è il seguente:

SQL Id	SQL Text
cwx2j4bsf36nx	INSERT INTO INVOICEDETAIL ID (UIDINVOICE, UIDFACILITY, TARIFFYEAR, REGNUMBER, STARTREADTIME, STARTREADUSAGE, CORRSTARTREADUSAGE, STOPREADTIME, STOPREADUSAGE, CORRSTOPREADUSAGE, KCOEFF, CORRECTORFLAG, TOTALUSAGE, CURYEARUSAGE, ARTICLECHARGE, JURISCODE, PCS, MCOEFF, PRIORYEARUSAGE, ISADJ, FXDCHRGTYPENAME, TOTALFIXEDCHARGE, TOTALVARCHARGE, TOTALDISTCHARGE, BILLPERIOD, NUMREADINGS, READINGCHARGE, MAINTENANCECHARGE, TIER1USAGE, TIER1CHARGE, TIER2USAGE, TIER2CHARGE, TIER3USAGE, TIER3CHARGE, TIER4USAGE, TIER4CHARGE, TIER5USAGE, TIER5CHARGE, TIER6USAGE, TIER6CHARGE, TIER7USAGE, TIER7CHARGE, UIDFACTOR, ALPHA, BETA, UIDTYPEOFUSE, ESTIMATEDUSAGE, UIDNEWFACTOR, UIDOLDFACTOR, UIDBA, UIDTARIFF, AQVARCHARGE, COLCHARGE, TIER8USAGE, TIER8CHARGE, TOTALUSAGEMC , UG2FIXEDCHARGE, UG2VARCHARGE, UG2TIER1CHARGE, UG2TIER2CHARGE, UG2TIER3CHARGE, UG2TIER4CHARGE, UG2TIER5CHARGE, UG2TIER6CHARGE, UG2TIER7CHARGE, UG2TIER8CHARGE, UG2TIER1USAGE, UG2TI ER2USAGE, UG2TIER3USAGE, UG2TIER4USAGE, UG2TIER5USAGE, UG2TIER6USAGE, UG2TIER7USAGE, UG2TIER8USAGE, RECHARGE, RSCHARGE, UG1CHARGE, GSCHARGE, T1DISCHARGE, T1MISCHARGE, T2MISCHARGE, T3MISCHARGE, TCOTCHARGE, DELTATCOTCHARGE, HAS_GS, UIDINVIDTYPE , CURYEARESTUSAGE, LASTREADDATE, UIDREAD, READUSAGE, CURYEARREADUSAGE, UG3INTCHARGE, UG3UICHARGE, UG3FTCHARGE, AQVARTIER1USAGE, AQVARTIER2USAGE, RETIER1CHARGE, RETIER2CHARGE, RSTIER1CHARGE, RSTIER2CHARGE, UG1TIER1CHARGE, UG1TIER2CHARGE, GSTIER1CHARGE, GSTIER2CHARGE, STCHARGE, VRCHARGE, CODE, GCVAL, LSTIME , STOPTIME , STOPREADING , CORRSTOPREAD , UIDMETERSTATUSRSN , SERVIZIO_TECNICO) SELECT /*+index(IDS)*/ :B3 , IDS.UIDFACILITY, IDS.TARIFFYEAR, IDS.REGNUMBER, IDS.STARTREADTIME, IDS.STARTREADUSAGE, IDS.CORRSTARTREADUSAGE, IDS.STOPREADTIME, IDS.STOPREADUSAGE, IDS.CORRSTOPREADUSAGE, IDS.KCOEFF, IDS.CORRECTORFLAG, NVL (IDS.TOTALUSAGE, 0), IDS.CURYEARUSAGE, IDS.ARTICLECHARGE, IDS.JURISCODE, IDS .PCS, IDS.MCOEFF, IDS.PRIORYEARUSAGE, IDS.ISADJ, IDS.FXDCHRGTYPENAME, IDS.TOTALFIXEDCHARGE, IDS.TOTALVARCHARGE, IDS.TOTALDISTCHARGE, IDS.BILLPERIOD, IDS.NUMREADINGS, IDS.READINGCHARGE, IDS.MAINTENANCECHARGE, IDS.TIER1USAGE, IDS.TIER1CHARGE, IDS.TIER2USAGE, IDS.TIER2CHARGE, IDS.TIER3USAGE, IDS.TIER3CHARGE, IDS.TIER4USAGE, IDS.TIER4CHARGE, IDS.TIER5USAGE, IDS.TIER5CHARGE, IDS.TIER6USAGE, IDS.TIER6CHARGE, IDS.TIER7USAGE, IDS.TIER7CHARGE, IDS.UIDFACTOR, IDS.ALPHA, IDS.BETA, IDS.UIDTYPEOFUSE, IDS.ESTIMATEDUSAGE, IDS.UIDNEWFACTOR, IDS.UIDOLDFACTOR, IDS.UIDBA, IDS.UIDTARIFF, IDS.AQVARCHARGE, IDS.COLCHARGE, IDS.TIER8USAGE, IDS.TIER8CHARGE, IDS.TOTALUSAGEMC , IDS.UG2FIXEDCHARGE, IDS.UG2VARCHARGE, IDS.UG2TIER1CHARGE, IDS.UG2TIER2CHARGE, IDS.UG2TIER3CHARGE, IDS.UG2TIER4CHARGE, IDS.UG2TIER5CHARGE, IDS.UG2TIER6CHARGE, IDS.UG2TIER7CHARGE, IDS.UG2TIER8CHARGE, IDS.UG2TIER1USAGE, IDS.UG2TIER2USAGE, IDS.UG2TIER3USAGE, IDS.UG2TIER4USAGE, IDS.UG2TIER5U SAGE, IDS.UG2TIER6USAGE, IDS.UG2TIER7USAGE, IDS.UG2TIER8USAGE, IDS.RECHARGE, IDS.RSCHARGE, IDS.UG1CHARGE, IDS.GSCHARGE, IDS.T1DISCHARGE, IDS.T1MISCHARGE, IDS.T2MISCHARGE, IDS.T3MISCHARGE, IDS.TCOTCHARGE, IDS.DELTATCOTCHARGE, IDS.HAS_GS, :B2 , IDS.CURYEARESTUSAGE, IDS.LASTREADDATE, IDS.UIDREAD, IDS.READUSAGE, IDS.CURYEARREADUSAGE, UG3INTCHARGE, UG3UICHARGE, UG3FTCHARGE, IDS.AQVARTIER1USAGE, IDS.AQVARTIER2USAGE, IDS.RETIER1CHARGE, IDS.RETIER2CHARGE, IDS.RSTIER1CHARGE, IDS.RSTIER2CHARGE, IDS.UG1TIER1CHARGE, IDS.UG1TIER2CHARGE, IDS.GSTIER1CHARGE, IDS.GSTIER2CHARGE, IDS.STCHARGE, IDS.VRCHARGE, IDS.CODE, IDS.GCVAL, IDS.LSTIME , IDS.STOPTIME , IDS.STOPREADING , IDS.CORRSTOPREAD , IDS.UIDMETERSTATUSRSN , IDS.SERVIZIO_TECNICO FROM INVOICEDETAILSTAGE IDS WHERE IDS.PROCESSID = :B1 AND (IDS.ESTCOMPLETE = 'Y' OR IDS.ESTCOMPLETE IS NULL) ORDER BY IDS.UIDFACILITY

Trattasi di uno statement di INSERT SELECT ORDER BY con piano di esecuzione corretto

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	INSERT STATEMENT					3809 (100)	
1	LOAD TABLE CONVENTIONAL						
2	SORT ORDER BY		13077	5670K	8728K	3809 (1)	00:00:46
3	TABLE ACCESS BY INDEX ROWID	INVOICEDETAILSTAGE	13077	5670K		2573 (1)	00:00:31
4	INDEX RANGE SCAN	IX_INVDETST_PROCESSID	13081			112 (0)	00:00:02

Workload Repository SQL Report - 2 (awrsqrpi)

Le statistiche di esecuzione sono le seguenti:

Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	4,896,680	32,004.44	29.70
CPU Time (ms)	399,475	2,610.95	12.93
Executions	153		
Buffer Gets	35,524,421	232,185.76	17.62
Disk Reads	2,307,792	15,083.61	35.67
Parse Calls	152	0.99	0.00
Rows	3,351,410	21,904.64	
User I/O Wait Time (ms)	3,965,534		
Cluster Wait Time (ms)	611,414		
Application Wait Time (ms)	0		
Concurrency Wait Time (ms)	115		
Invalidations	53		
Version Count	20		
Sharable Mem(KB)	3,081		

Tempo medio di esecuzione: 32 secondi.
Quasi tutto tempo di I/O

Le possibilità di migliorare questo statement sono decisamente scarse.
Vorrebbe dire evitare di fare I/O, ma l'indice coinvolto è quasi di 70GB, andrebbe mantenuto in memoria.

Considerazioni sull'analisi

L'analisi condotta con i soli script del mondo AWR (awr*sql e ash*sql) ha permesso di capire:

- che il database non è particolarmente stressato;
- che l'attività osservata si svolge sul secondo nodo con un processo applicativo monothread;
- che in una fase iniziale vi è uno statement in parallel con hint di parallel a default che porta alla partenza di 320 processi, un numero esagerato. Questa fase però occupa solo i primi 15 minuti;
- che successivamente l'attività è prevalentemente basata su statement SQL quasi sicuramente tutti, o buona parte, lanciati all'interno di una stored procedure;
- che l'evento principale è l'I/O a singolo blocco che però mostra un tempo medio di risposta ottimale (< 2ms);
- che alla fine non è il database il problema (nel senso sistemistico o di comportamento degli statement), ma l'architettura applicativa di questa fase che è monothread;
- che avendo la bacchetta magica un miglioramento si potrebbe avere eliminando l'I/O (indice in cache avendo memoria a disposizione?).

Approfondimento sull'I/O - 1

Lo statement che occupa la maggior parte del tempo nel database per la sessione di billing è, come si è visto, quello di INSERT nella INVOICEDETAIL che viene campionata per il 55% del tempo.

L'I/O in questo statement cuba l'81%. L'abbattimento dell'I/O porterebbe al risparmio dell'81% del 55%, vale a dire il 44% del tempo, sufficiente, almeno lato database, ad assorbire una crescita del 20% dei clienti da trattare.

Le operazioni di I/O battono su due oggetti:

- l'indice IX_INVDET_FAC sulla tavola target INVOICEDETAIL
- la tavola driver INVOICEDETAILSTAGE

Diverse sono le possibilità di miglioramento:

- a risorse infinite i due oggetti potrebbero essere mantenuti in memoria (in buffer cache KEEP). L'indice IX_INVDET_FAC è di 70GB in crescita, la tavola INVOICEDETAILSTAGE è invece di 31GB anch'essa in crescita. Con 128GB da dedicare alla KEEP di buffer cache si potrebbero eliminare le operazioni di I/O su questi due oggetti
- una revisione della struttura fisica degli oggetti …

Approfondimento sull'I/O - 2

Per revisione della struttura fisica degli oggetti si intende l'intervento sull'indice IX_INVDET_FAC.

La tavola INVOICEDETAIL è partizionata sulla colonna TARIFFYEAR che contiene l'anno di riferimento della fattura. La tavola è costituita da 17 partizioni, ognuna con almeno un centinaio di milioni di record (204 milioni nel 2016)

Sulla tavola insistono due indici:

- la primary key PK_INVDET composta da ben 6 colonne con la prima che è il numero di fattura UIDINVOICE;
- l'indice secondario IX_INVDET_FAC sulla sola colonna UIDFACILITY.

Nessuno dei due indici è partizionato, ma la fase di INSERT va a provocare I/O sui blocchi del secondo indice perché il valore UIDFACILITY si distribuisce. Mediamente nella tavola INVOICEDETAILSTAGE a parità di PROCESSID per 100 record si hanno 81 valori differenti sulla colonna UIDFACILITY. Ciò porta ad accedere a 81 diversi blocchi dell'indice ogni 100 record trattati.

Nel caso della primary key, la sequenza del valore UIDINVOICE porta invece ad una località di accessi ai blocchi dell'indice stesso.

Approfondimento sull'I/O - 3

Un'idea per ridurre l'I/O è quella di partizionare l'indice secondario, rendendolo di tipo LOCAL rispetto alla colonna TARIFFYEAR.

Questo non evita l'alta distribuzione dei valori UIDFACILITY, e quindi di conseguenza l'alto numero di blocchi da aggiornare, ma rende più semplice il caching degli stessi perché sarebbe possibile porre in KEEP la sola partizione dell'indice relativa all'anno in corso. Invece di dover cachare 70 o più GB, la necessità potrebbe essere di soli 10GB.

Anche la tavola INVOICEDETAILSTAGE potrebbe essere partizionata per TARIFFYEAR in modo da mantenere in cache, sempre in apposita area KEEP della buffer cache, i record dell'anno che si sta trattando. Questo potrebbe abbattere le attività di I/O sulla tavola driver senza doverla porre tutta in cache, anche se forse il risparmio di spazio potrebbe essere esiguo.

Grazie dell'attenzione!

francesco.renne@icteam.it

Domande?