Oracle Datenbank Tuning richtig gemacht und dabei viel Geld gespart

Christian Pfundtner
DB Masters GmbH
Gerasdorf bei Wien, Österreich

Schlüsselworte
Oracle Datenbank, Tuning, Memory, CPU, Performanz, Infrastruktur, Lizenzkosten, Kostengünstig

Einleitung
In meinen über 30 Jahren in der IT habe ich Eines gelernt – die Hardware verändert sich laufend, nur reagieren wir oft nicht (passend) darauf. In diesem Vortrag werden einerseits die Erfahrungen aus Kundenprojekten der letzten Jahre herangezogen als auch die „üblichen Verdächtigen" im Bereich Datenbank Tuning behandelt.

KIWH – wer kennt diesen Spruch noch?
Im letzten Jahrtausend war KIWH – „kill it with hardware" ein sehr oft verwendeter Ausspruch bei IT Verantwortlichen, wenn es um das Thema Applikationsperformanz gegangen ist. Das hat auch sehr gut funktioniert, da sich im Schnitt alle 12-18 Monate die Taktfrequenz der CPUs verdoppelt haben – damit, dass die Systeme alle 3 Jahre somit zumindest vier Mal so schnell waren wie die vorhandenen, konnte man mangelnde Applikationsperformanz ausgleichen. Doch mit dem Erreichen von ca. 4GHz war auf einmal Schluss. Die CPU-Hersteller konnten die Taktfrequenz nicht mehr sinnvoll steigern, da man die CPUs nicht mehr kühlen konnte, also haben sie sich umorientiert und nicht mehr die GHz gesteigert, sondern die CORE Anzahl. Solange man deutlich mehr gleichzeitige Anforderungen wie Cores hat, kann man damit ebenfalls skalieren – nicht mehr über die Single Thread Performance, aber über die gleichzeitige Verarbeitung mehrere Anfragen.

Leider ändern sich aber die Applikationen nicht dementsprechend – es gibt mehr als genug Anwendungen, wo es auf die Single Thread Performance ankommt – also darauf, wie schnell EINE Verarbeitung (ohne Parallelisierung) abgearbeitet werden kann.

Natürlich gibt es Applikationen, die von vielen Cores profitieren – erwähnt seien hierdie Bereiche Web und DWH, wo man die Abfragen in den meisten Fällen sehr gut parallelisieren kann. Das ist aber definitiv nicht die Mehrheit aller Anwendungen.

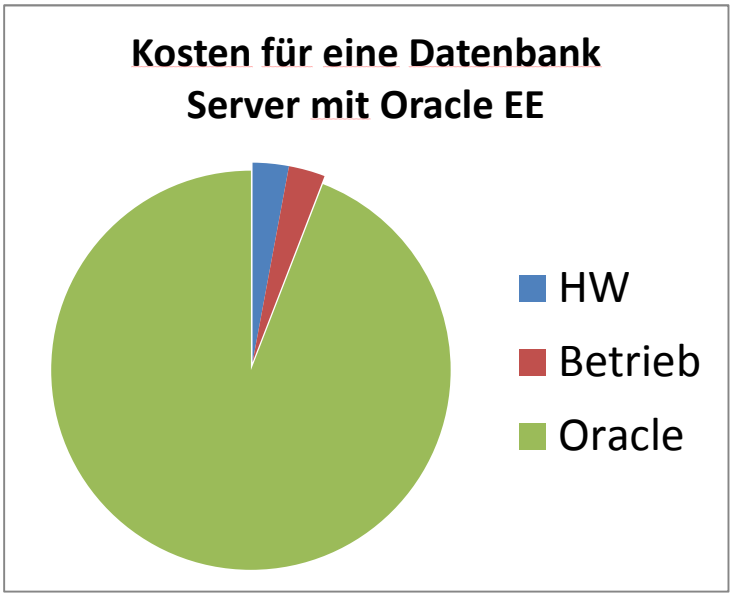
Die Oracle Hardware (Exadata, Exalogic, Sparc T5, Sparc M5,…) gibt, richtig eingesetzt, einen einmaligen Performanceschub – man befindet sich dann aber in einer Falle, da man aus heutiger Sicht (und sofern man die Hardware aus Performancegründen auch wirklich benutzt) auf keine andere Hardware wechseln kann ohne eine Performancedegradierung zu erfahren.

Wo liegen die Kosten einer Oracle Datenbank Lösung auf 5 Jahre gesehen?

In den Preisangaben für 5 Jahre sind auch die entsprechenden Wartungskosten inkludiert, wobei die Wartungskosten und Betriebskosten sowohl bei der Hardware als auch bei den Lizenzkosten in etwa gleich viel ausmachen wie die Anschaffungskosten.

Bezeichnung	**Preis**
Server, 2 Sockel, 192GB Memory, 2 HBA oder 2 10GBit NICs	5-10k Euro
Betrieb (Strom und Kühlung)	5-10k Euro
Oracle Standard Edition 2 Sockel	60k Euro
Oracle EE für 2 Stück 4 Core CPUs = 8 Cores (Intel = 4 Lizenzen)	320k Euro
Oracle EE für 2 Stück 8 Core CPUs = 16 Cores	640k Euro
Oracle EE für 2 Stück 12 Core CPUs = 24 Cores	960k Euro

Bei den Preisen für die Oracle Datenbank wurde der Listenpreis verwendet und selbstverständlich bekommt man darauf einen mehr oder weniger großen Rabatt, letztendlich ist trotzdem die Oracle Lizenz mit ABSTAND der größte Brocken.



Was lernt man daraus?

Es kann durchaus günstiger sein, wenn man vorhandene Hardware durch "besser passende" Hardware ersetzt, wenn man gleichzeitig weniger für Oracle Lizenzen bezahlen muss. Wenn man beispielsweise in einem Projekt neue Server und Lizenzen kaufen müsste, ist es defacto immer kostengünstiger zu KONSOLIDIEREN. Ja, ich weiß, das ist nicht neu – neu ist aber der Ansatz, dass man gleichzeitig auch Performancesteigerungen erzielen kann.

In vielen Fällen wird auch die Oracle Enterprise Edition eingesetzt, obwohl man keine Funktionalitäten davon ausnutzt – der Umstieg auf die Oracle Standard Edition rechnet sich immer innerhalb von wenigen Jahren und man hat auch nicht mehr das Problem, dass neue Systeme immer mehr Cores haben (und man entsprechend nachlizensieren müsste). In den letzten Jahren haben wir bei mehreren Kundenprojekten evaluiert ob die Applikation wirklich EE benötigt und in deutlich mehr als

der Hälfte der Fälle hat auch eine Oracle Standard Edition ausgereicht. Bitte nicht falsch verstehen – ich bin PRO Enterprise Edition, wenn man die Features und Optionen auch wirklich einsetzt! Query Rewrite, Verschlüsselung, Online Operations, Parallelisierung, Partitioning, usw. sind ihr Geld wert, wenn man diese auch wirklich nutzt.

Immer öfter wird die Storage (und die Storage Infrastruktur) zum Engpass, weil immer mehr Server an die schon vorhandene Storage angeschlossen werden und sich somit immer mehr Systeme die verfügbare I/O Performance teilen müssen. Natürlich kann man in immer größere, bessere, schnellere,… Storages und den Infrastrukturausbau investieren, oft reicht es aber auch schon aus, wenn man „einfach" die durch Datenbanken verursachten lesenden I/Os reduziert.

Ein weiteres Thema ist die Verfügbarkeitsanforderung an die Systeme – diese steigen immer weiter, werden aber gleichzeitig im Unternehmen nicht klar definiert. Dadurch kommt es zu Fehleinschätzungen, die letztendlich Geld kosten – entweder gleich, weil man zu viel investiert, oder später, weil die Downtimekosten höher sind als gedacht.

Tuning, Schritt 1: Bestandsaufnahme

Wie geht es Ihren Datenbanken WIRKLICH?

Das möchte ich anhand eines Real Live Kundenbeispiels – einer SAP Datenbank – aufzeigen.

Der Oracle Memory Advisor sagt: man könnte das Memory für die Instanz sogar leicht reduzieren und die Cache Hit Ratio liegt bei 99,87%. Wenn man aber genauer nachsieht, findet man folgendes:

- V$DB_CACHE_ADVICE: 55% ESTD_PCT_OF_DB_TIME_FOR_READS
- Über 99% der Verarbeitung ist I/O Wait
- Bei einem Statement Cache Hit Ratio von über 99%
- Wo aber häufig laufende Statements um den Faktor 100 schneller werden könnten…

SQL_ID	EXECUTIONS	CPU_TIME_SEC	ELA_TIME_SEC	USER_IO_WAIT_SEC	PCT_IO_WAIT	POT_TIMES_FASTER	BUFFER_GETS	DISK_READS	BC_HIT_RATIO	SQL_TEXT
0dukg1v1ycz5r	1234	.008	1.745233	1.625439	93.13	218	1421	22	98.47	SELECT * FROM "/PBS/RSDANLOBJ" WHERE "VI
2iyjw21bhs5tt	1	.008	1.246586	1.235624	99.12	155	821	6	99.27	SELECT COUNT(*) FROM "/BIC/B0001671000"
2swr2klxrpwwu	50698	.160001	21.885152	20.843505	95.24	136	101877	259	99.74	SELECT "MANDT", "PROPN", "AKTPS", "SUBPR
4329xkgxwpv8z	506537	4.012249	420.50097	404.160866	96.11	104	1013315	11462	98.88	SELECT "RNR", "IDOCNUM", "FIELD", "LNR",
3ythjygwnzkua	34491	.508027	37.299274	35.00915	93.86	73	138550	847	99.39	SELECT MAX("STARTTIMESTAMP") FROM "BSPCP
0nyqjj920f0pg	9844	.124006	8.2767	7.917053	95.65	66	10509	184	98.27	SELECT "LOCK_ID", "UNIT_KIND" FROM "BGRF
awf3ppv9cmasw	4568	.064003	4.197239	3.967189	94.51	65	63721	454	99.29	SELECT * FROM "/BIC/SSD_KUNNR" WHERE "SI
189qwyhsdu25w	16106	.412028	25.628337	24.859538	97	62	294491	2359	99.2	SELECT "/BIC/SD_DOCITM", "SID", "CHCKFL"
cp0qg59kb7kj7	2644	.096005	5.899696	5.741311	97.31	61	99691	569	98.58	SELECT "/BIC/TIS_EXTI2", "SID", "CHCKFL"
6rp5ndhxkjvfz	5039	.092004	5.538074	5.220329	94.26	60	75336	537	99.29	SELECT * FROM "/BIC/SHO_WIKGEP" WHERE "/
dmyvzgc3zfkck	3063	.108008	6.397326	6.215976	97.16	59	47726	614	98.72	SELECT "/BIC/TIS_TRAN1", "SID", "CHCKFL"
ayjc75bjz795m	3664	.160013	9.42156	9.191342	97.55	58	52074	1015	98.08	SELECT "/BIC/TIS_FRANU", "SID", "CHCKFL"
fsudh5h77sg08	3456	.124011	7.307523	7.093473	97.07	58	47503	882	98.17	SELECT * FROM "/BIO/SCOORDER" WHERE "SID
gkdcd3byg60t5	3974	.036001	2.122232	1.997487	94.12	58	15846	238	98.52	SELECT * FROM "/BIO/TCOORDER" WHERE "COO
fyyx3tngcyqv7	37800	2.388137	136.942033	132.775216	96.95	57	1022667	11236	98.91	SELECT "/BIC/SD_DOCITM", "OBJVERS", "CHA
gmz7wkxr9hyrd	16822	.220019	12.680465	11.976031	94.44	57	33644	655	98.09	SELECT "RFCDES", "LIMBO", "RFCTYPE", "RF
e24kpbckz3qxd	585773	6.072383	341.551488	324.200566	94.91	56	2343107	41920	98.24	SELECT "/BIC/TIS_TRAN1", "SID", "CHCKFL"
5qudzd8mbr5zp	20257	.504037	26.150203	24.168024	92.42	51	77033	581	99.25	SELECT "JOBNAME", "JOBCOUNT", "ROOT_CONT
du7zfyrndj3bx	2226	.044003	2.267992	2.149828	94.78	51	31169	291	99.07	SELECT * FROM "/BIO/SBPARTNER" WHERE "SI

Abb. 1: Output einer Query gegen V$SQLAREA sortiert nach „POT_TIMES_FASTER

In dem Bild sieht man einen Output einer Query gegen V$SQLAREA in der die CPU_TIME im Verhältnis zur USER_IO_WAIT_TIME gesetzt wird. Die CPU_TIME gibt an, wie lange die CPU mit der Verarbeitung des Statements beschäftigt war. Die USER_IO_WAIT_TIME zeigt die Zeit, die Oracle auf I/O Requests warten musste. Wenn ein Statement nur einmal ausgeführt wird – also EXECUTIONS =1 – dann ist es wahrscheinlich, dass nicht alle Daten im Cache von Oracle stehen werden, wie man aber an diesem Output sieht, werden fast alle Statements mehrere bis viele 1000 mal ausgeführt → warum also soll man die dafür benötigten Daten nicht im Memory behalten, sofern dafür eine sinnvolle Cache Größe reicht?

Bei dieser Datenbank hat es gereicht, den Buffer Cache von 40G auf ca. 180GB zu vergrößern – das war möglich, weil der Server sowieso über 256GB verfügt hat und man diese jetzt einfach der Oracle Instanz zur Verfügung gestellt hat. Das Ergebnis: die CPU Belastung ist deutlich gesunken (weil

weniger I/O Requests bearbeitet werden mussten) und die Performance ist over all gestiegen, einige Abfragen sogar um Faktoren von 10 bis 100! Gleichzeitig ist das I/O-Aufkommen um über 50% zurückgegangen. Durch die Umsetzung solcher Buffer Cache Anpassungen über (fast) alle Datenbanken des Kunden konnte die I/O Last am Storage (wo nicht nur Oracle Datenbanken liegen) um über 30% reduziert werden → der Kunde hat sich ein sehr teures Storageupgrade erspart.

AWR Reports oder Statspack Report

Zur Analyse nimmt man nicht nur die Empfehlungen der Oracle Memory Advisorn, sondern einen AWR Report oder Statspack Report. In diesen findet man die folgenden beiden Abschnitte (eine andere Kundendatenbank):

TOP 5 WAIT EVENTS

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	3,383,657	26,378	**8**	**55.6**	User I/O
DB CPU		11,130		23.4	
db file parallel read	114,083	1,868	16	3.9	User I/O
db file scattered read	129,357	1,138	9	2.4	User I/O
gc current block 2-way	774,175	511	1	1.1	Cluster

TIME MODEL

Wait Class	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	%DB time
User I/O	3,870,898	0	29,800	8	62.8
DB CPU			11,130		23.4
Cluster	1,727,897	0	1,107	1	2.3
Other	517,232	60	463	1	1.0
Commit	127,033	0	378	3	0.8
Network	73,203,283	0	208	0	0.4
Concurrency	65,524	1	120	2	0.3
System I/O	107,523	0	56	1	0.1
Application	6,914	0	55	8	0.1
Configuration	215	22	17	77	0.0
Administrative	3	67	0	83	0.0

Hier sieht man ganz eindeutig, dass die Datenbank ein I/O Thema hat – die I/O Zeiten von durchschnittlich 8ms entsprechen eine Entry Level oder Midrange Storage und sind nicht zu beanstanden – allerdings der Fakt, dass die Datenbank READ I/O BOUND ist, schon! Auch bei dieser Datenbank behauptet die Oracle Memory Advisor, dass alles in Ordnung ist – siehe auch Instance Efficency.

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.97	Redo NoWait %:	100.00
Buffer Hit %:	**97.67**	In-memory Sort %:	100.00
Library Hit %:	99.95	Soft Parse %:	99.23
Execute to Parse %:	95.92	Latch Hit %:	99.88
Parse CPU to Parse Elapsd %:	68.56	% Non-Parse CPU:	94.37

Die Cache Hit Ratio ist zwar „nur“ 97.67% aber alle Oracle Advisoren sind damit zufrieden.

Man kann sich das Ganze auch im Cloud Control 12c / Enterprise Manager ansehen (sofern man das Diagnostic Pack lizensiert hat) – hier ein Beispiel eines TOP Aktivity Outputs einer Datenbank:

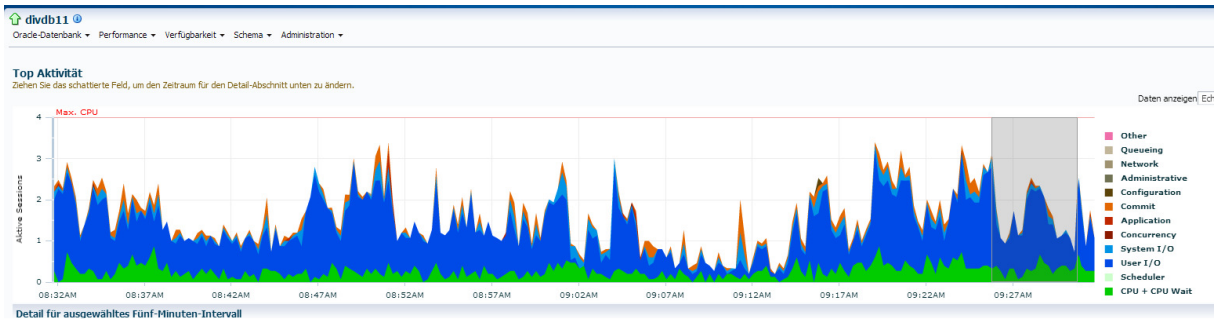


Abb. 2: TOP Activity einer Datenbank mit viel User I/O Wait

Hier ist es ganz eindeutig, dass die Datenbank praktisch kaum etwas macht, außer darauf zu warten, dass die I/Os fertig werden.

Probleme bei der I/O Performanceanalyse

Dadurch, dass Storages von vielen Systemen genutzt werden und die Anzahl der I/O Requests entsprechend schwankt und auch das Cache der Storage nicht immer gleich gut befüllt ist, schwankt auch die Antwortzeit und somit die Performance der Verarbeitung. Da bei Oracle meist die Lese-Performance (betrifft die Verarbeitung direkt) wichtiger ist als die Schreibperformance (Ausnahme LGWR), kann man genau hier durch entsprechend große Buffer Caches zwei Fliegen mit einer Klappe schlagen:

1) Die Performance der SQL Statements steigt, je mehr sich im Buffer Cache befindet und je weniger I/O gemacht werden muss.
2) Man entlastet das Storage System deutlich, wenn man einen großen Teil der lesenden Anforderungen der Datenbanken weg bekommt – das kommt auch anderen Systemen zugute und verbessert sogar die Performance der schreibenden I/Os für die Datenbanken selbst.

Memory für Server ist in den letzten Jahren immer billiger geworden und gleichzeitig kann man auch immer mehr Memory in „günstige“ Server einbauen. Mit aktuellen CPUs und Memories sind bis zu 1.5TB Hauptspeicher bei einem 2 Sockel System Intel Xeon E7 technisch möglich (aber noch zu teuer). Aktuell preislich sinnvoll sind Systeme mit 2 Sockel und 384GB (Intel Xeon E5 oder E7) oder bei Bedarf auch 768GB (Intel Xeon E7).

Ein weiteres Problem bei der Analyse stellt der „Durchschnitt“ dar. Oracle macht die AWR Snapshots jede Stunde, Storages liefern meist 5 oder 15 Minuten Werte und auch beim OS Monitoring werden vielfach alle paar Minuten Werte geliefert. Das ist ja grundsätzlich nicht schlecht – minimaler Einfluss durch das Monitoring bei gleichzeitig sinnvollem Überblick. Wenn man aber einem I/O Performance Thema oder einem Engpass auf der Spur ist, reicht diese zeitliche Auflösung nicht – hier sollte man wirklich versuchen, die benötigten Messwerte im Sekundentakt zu erhalten (nicht immer möglich). Ganz schlimm ist es aber, wenn jemand „sein Performancetool“ auspackt – hier wird dann immer etwas ganz Tolles gemessen, das leider mit den Anforderungen nichts zu tun hat. Für Analyse ist es wichtig, dass man bei Tests und Benchmarks möglichst nahe an den Anforderungen der Applikation bleibt.

Engpasserkennung und Lösungsmöglichkeiten

Gleich vorweg – Applikationstuning kann durch Nichts ersetzt werden! Was man mit Datenbank und Hardwareoptimierung machen kann, liegt im Bereich von % (meist zweistellig) – durch Applikationstuning sind Faktoren an Performancesteigerung möglich!

Permanente Read I/Os bedeuten – abgesehen vom schlechten Applikationsdesign und Implementierung – immer zu wenig Memory, und genau hier kann man ansetzen. Die Empfehlungen von früher, dass die Cache Hit Ratio > 98% sein soll, oder das READ zu WRITE Verhältnis nicht über 20:1 sind definitiv nicht mehr zeitgemäß. Diese Empfehlungen kommen aus Zeiten, wo der Hauptspeicher auf einige wenige GB limitiert war und man glücklich sein konnte, wenn man 8 oder gar 16GB Hauptspeicher nutzen konnte. Der aktuelle Ansatz sollte sein: Reduzieren Sie den lesenden I/O auf ein Minimum, idealerweise sollte die Datenbank weniger lesen als schreiben!

Dies bringt nicht nur den Vorteil, dass man die Storage entlastet – weniger I/O bedeutet auch weniger WAITS, Context Switches, OS Overhead, usw. Es fällt somit nicht nur WAIT Anteil der Verarbeitung, auch der CPU Anteil nimmt ab → das System benötigt weniger von den teuer zu lizensierenden CPU Cores. Mit der Einführung von Oracle dNFS mit Oracle 11g ist eine weitere Möglichkeit zum Reduzieren der CPU Last gekommen – NAS statt SAN. Wir haben in verschiedenen Benchmarks auf Kundensystemen nachgewiesen, dass durch das „auslagern“ des Filesystem Overheads in die Storage bei korrekter Konfiguration die CPU Last bei 10GBit NFS trotz ca. 15% höheren Durchsatz niedriger ist als bei einer Umsetzung mit 8GBit SAN und lokalem Filesystem oder ASM.

Unabhängig von der Entscheidung SAN oder NAS, alle SQL Statements, die auf Grund zu kleiner Buffer Caches noch lesende I/Os benötigt haben, profitieren entsprechend von größeren Buffer Caches.

Entscheidend ist aber nicht nur, wie viel Memory man in den Servern zur Verfügung stellt, sondern auch die genutzte CPU Technologie. Oracle Sparc T CPUs unterscheiden sich durch ihre vielen Cores/Threads deutlich von klassischen CPUs wie Intel XEON, IBM Power oder auch Oracle Sparc M. Dieser Unterschied hat Vor- und Nachteile.

Klassische CPUs sind ideal für DWH, DSS, Mixed Workload und Single Thread Performance.

SPARC T4/T5 gewinnen dort, wo eine hohe Anzahl von gleichzeitigen Requests verarbeitet werden müssen (WebServer, Datenbanken mit sehr vielen Sessions wo die SQL Statements nur wenige ms lang laufen).

Für die meisten Anwendungen gilt aber somit:
Wählen Sie CPUs mit möglichst vielen GHz und nicht vielen Cores!

Ein Beispiel dazu?
Der Aufpreis für 2 Xeon E5-2643v2 (6 Core á 3.5Ghz = 42Ghz) statt E5-2450v2 (8 Core á 1.9GHz = 30GHz) liegt bei unter 1.000 Euro, aber

- SQL Statements sind fast doppelt so schnell (3.5GHz versus 1.9GHz)
- Sie haben ca. 30% mehr Gesamt-CPU-Performance (42GHz statt 30GHz)
- Bei deutlich geringeren Oracle EE Lizenzkosten (8 – 6 = 2 Core ca. 140k auf 5 Jahre)!

Wichtig ist somit, wie viele GHz bekomme ich, wenn ich die Anzahl der Cores mit den nominellen GHz multipliziere – das sagt am besten aus, welche CPU Leistung man erwarten darf.

Ein lediges Thema sind Konfigurationsfehler – hier gibt es unendlich viele Möglichkeiten etwas falsch zu machen, das sprengt definitiv den Rahmen dieses Vortrags (deswegen wird es nur der vollständigkeithalber erwähnt).

Tuning, Schritt 2: Proof of Concept

Zuerst sucht man sich 1-2 Kandidaten (= Datenbanken), die einen hohen I/O Wait Anteil aufweisen. Sofern man die Applikation nicht einfach gegen eine Kopie der Datenbank testen kann, empfiehlt es sich aus dem AWR oder Statspack Report die TOP SQL Statements als Basis für eine Benchmark heran zu ziehen.

Dann nehmen Sie ein System mit möglichst hoch getakteten CPUs und viel Memory (Leihstellungen sind auch eine Möglichkeit Systeme zu testen!). Führen Sie Ihre Benchmarks bei gleichzeitigem Monitoring von OS, Memory Usage, CPU Usage und Storage Auslastung immer 3 mal durch, und zwar:

- Auf ihrem aktuellen System – um eine IST Basis zu haben
- Auf dem neuen System bei gleichen Instanzparametern – um die Auswirkung der „schnelleren“ Hardware zu messen. Wenn die Messwerte schlechter als auf Ihrem aktuellen System sind, habe Sie irgendwo einen Konfigurationsfehler.
- Auf dem neuen System mit möglichst großen Buffer Cache – hier sollten einige Statements deutlich von dem mehr an Daten im Cache profitieren, andere weniger. Keines der Statements darf schlechter werden!

Analysieren Sie die Ergebnisse und staunen Sie, welche Unterschiede bei den Laufzeiten einiger Statements auftreten (vor allem bei 2. und 3. Durchlauf).

Hört sich eigentlich ganz einfach an, bietet aber genügend Raum um Fehler zu machen. Meist werden die verschiedenen Auswirkungen und Abhängigkeiten nicht berücksichtigt – beispielsweise um wie viel entlaste ich die CPU, wenn ich weniger I/O mache,…

Um den größten Fehlerquellen aus dem Weg zu gehen, ist es am besten, SQL*PLUS Client/Server von einem anderen System aus zu nutzen und die Statements mit „SET AUTOTRACE TRACEONLY STATISTICS“ auszuführen. Dadurch gibt es keine verfälschenden Bildschirmausgaben oder interne Verarbeitungen.

Tuning, Schritt 3: die Umsetzung

- Evaluieren Sie, welche Oracle Datenbank Edition wirklich benötigt wird
- Versuchen Sie aus dem Proof of Concept durch weitere Analysen mögliche Engpässe schon im Vorfeld auszuschließen.
- Bei der Server Hardware: Lieber mit GHz als mehr Cores, so viel Memory wie sinnvoll leistbar, überlegen ob Oracle dNFS eine Option ist.
- Nutzen Sie die Konsolidierungsmöglichkeiten und überlegen Sie Virtualisierung (auch als HA Lösung, Vorsicht: Lizensierungsfallen)

Zusammenfassung

In den letzten Jahren ist der in „günstigen“ Servern nutzbare Hauptspeicher praktisch explodiert – Systeme mit 384GB Memory kosten schon unter 5.000 Euro! Leider ist dies noch nicht in den Köpfen der Administratoren, noch in den Oracle Memory Advisoren angekommen. Statt Systeme mit immer mehr Cores zu kaufen, setzen Sie auf CPUs mit möglichst vielen GHz – damit erhöhen Sie definitiv die Gesamtperformance.

Kontaktadresse:
Christian Pfundtner
DB Masters GmbH
Stammersdorfer Str. 463
A-2201 Gerasdorf

Telefon: +43 (699) 150 37 884
E-Mail vorname.nachname@dbmasters.at
Internet: www.dbmasters.at