Oracle Datenbank Tuning richtig gemacht und viel Geld gespart

Ing Christian Pfundtner
OCA, OCP, OCE, OCM, ACE

ORACLE CERTIFIED MASTER

- Oracle Datenbank Spezialist seit 1993
  - Über 21 Jahre Erfahrung im Oracle Datenbank High End Consulting und Troubleshooting
  - Über 17 Jahre Erfahrung als Oracle Datenbank Trainer im High End Bereich

DB Masters GmbH

Datenbanken sind unsere Welt

- Gegründet / operativ tätig: Dez 2010 / Jan 2011
- Eigentümer: Christian Pfundtner
- Umsatz 2013 (gerundet): 863k Euro
- Umsatz 2014 (erwartet): > 1.2 Mio Euro
- Mitarbeiter aktuell: 7, davon 5 Techniker
- Externe Mitarbeiter (Projekte): 3, davon 3 Techniker

Wer kennt noch KIWH (kill it with hardware)? (1)

Im letzten Jahrtausend konnte man sicher sein, dass man nach einem Serveraustausch (alle 12-18 Monate hatten sich die MHz verdoppelt) die Applikationsperformance steigern konnte → fehlende Optimierung, ineffizientes Design,… konnten damit überspielt werden.

Wer kennt noch KIWH (kill it with hardware)? (2)

Doch in diesem Jahrtausend stimmt das nicht mehr!

Statt mehr MHz wurden es mehr Cores, nur das hat die Performance für eine Verarbeitung nicht gesteigert!

Wer kennt noch KIWH (kill it with hardware)? (3)

Dann kauft Oracle SUN und sagt: KIWH, jetzt erst recht!

- Exadata
- Exalogic
- Sparc T mit vielen Cores
- Spart M mit vielen TB

Wer kennt noch KIWH (kill it with hardware)? (4)

Aber

- Jede Applikation wird mit Exadata schneller, leider oft nur um den Faktor 2-3
- Sparc T mit den mehrerem 100 Cores ist super für bestimmte Anwendungen, aber nicht für alle
- Sparc M mit vielen TB an Memory, super für alle die DWH, DSS und In-Memory Cache Option brauchen

Nur ist auch der Preis super…

Wer kennt noch KIWH (kill it with hardware)? (5)

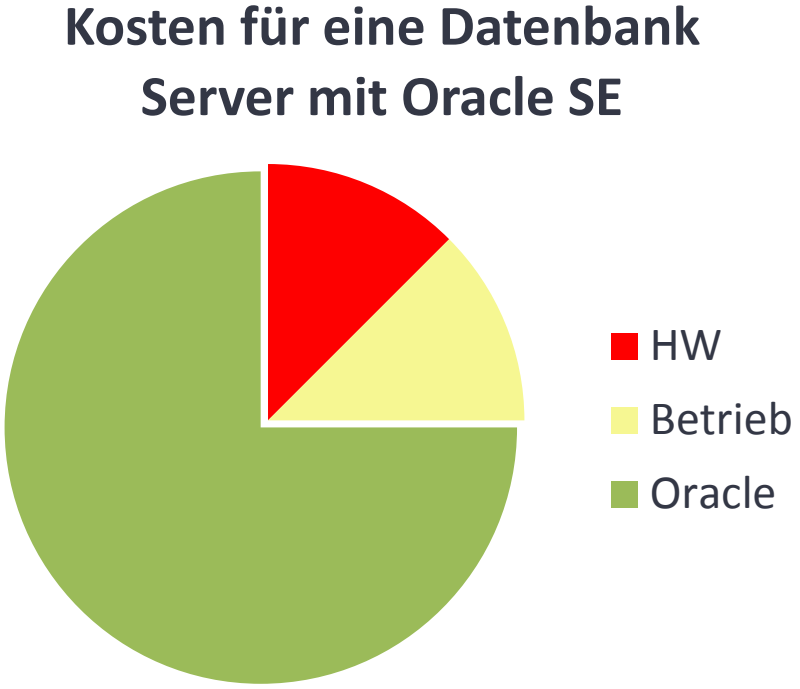
Ich sage aber auch: KIWH – soweit es (sinnvoll) geht

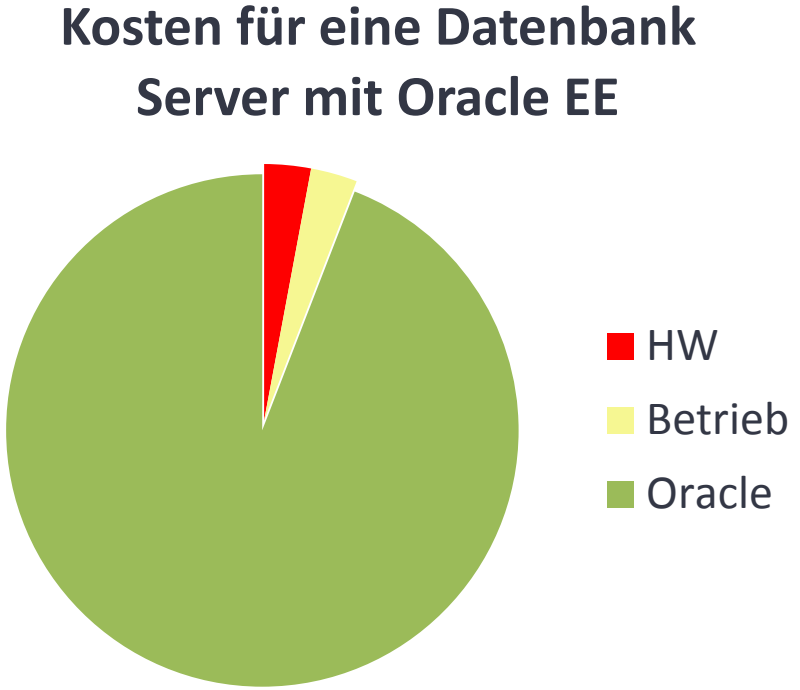
- Elimieren von Read I/O durch Cache
- CPUs mit weniger Cores aber mehr GHz für Single Thread Performance
- Optimieren von I/O durch Infrastrukturanpassungen
- Aber auch Reduktion der Kosten bei gleichzeitiger Steigerung der Performance

Wo liegen die Kosten einer Oracle Lösung
Beispiel: Oracle Datenbank – auf 5 Jahre gerechnet

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

Wo liegen die Kosten einer Oracle Lösung
Beispiel: Oracle Datenbank – auf 5 Jahre gerechnet





Was lernt man daraus?

- Die Hardware und Betriebskosten sind (fast) vernachlässigbar
  - Ersetzen Sie Hardware, die Ihren Anforderungen nicht entspricht!
  - Versuchen Sie die vorhandene Lizenzen optimal auszunutzen
  - Analysieren Sie Engpässe und beseitigen Sie diese
  - Sinnvolle Konsolidierung
- Lizenztype
  - Sofern Sie Oracle EE nicht wirklich nutzen, sollten Sie auf Oracle SE zurück steigen!
  - Oracle EE (Core Lizensierung) birgt auch noch das Problem/Risiko, dass neue Systeme immer mehr Cores haben als alte!

Pitfalls

- Verlieren Sie Verfügbarkeitsanforderungen nicht aus den Augen
  - Aber überdenken Sie wie Sie diese konzipieren
    - Oracle RAC, Oracle Data Guard, Oracle FailSafe, Oracle VM, VMware,…
- Belegen Sie, welche Einsparungen es gibt, wenn man in Upgrades / neue Systeme investiert – sonst bekommen Sie das Geld dafür nicht
  - Einsparungen durch weniger Lizenzbedarf
  - Einsparungen durch weniger Hardware und Betriebskosten

Tuning
Schritt 1 - Bestandsaufnahme

- Wie geht es meinen Datenbanken WIRKLICH?
- Wait Event Analyse
- I/O Performance Analyse
- Engpasserkennung und Lösungsmöglichkeiten

Wie geht es meinen Datenbanken WIRKLICH?

- Sie haben schon 99,87% Cache Hit Ratio?
- Der Memory Advisor sagt: kein Bedarf an mehr Memory…
- Aber wie erklären Sie sich das?
  - 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…

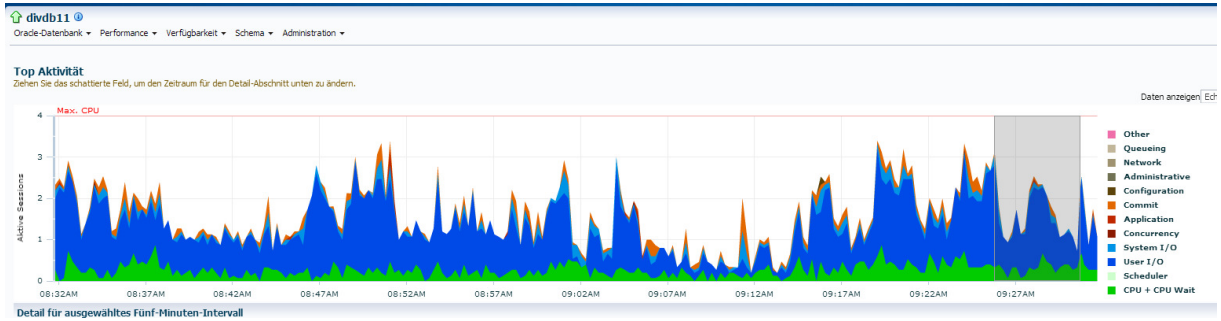
SQL_ID	EXECUTIONS	CPU_TIME_SEC	ELA_TIME_SEC	USER_IO_WAIT_SEC	PCT_IO_WAIT	POT_TIMES_FASTER	BUFFER_GETS	DISK_READS	BC_HIT_RATIO	SQL_TEXT
0dukg1v1ycz5r	1234	.008	1.745233	1.625439	93.13	218	1421	22	98.47	SELECT * FROM "/PBS/RSDANLOBJ" WHERE "VI
21yjw21bhz5tt	1	.008	1.246586	1.235624	99.12	155	821	6	99.27	SELECT COUNT(*) FROM "/BIC/B0001671000"
2swr2k1xrpwwu	50698	.16001	21.885152	20.843505	95.24	136	101877	259	99.74	SELECT "MANDT", "PROFN", "AKTPS", "SUBPR
4329xkqswpv8z	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 "RSPCP
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"
cp0qg59kb7xj7	2644	.096005	5.899696	5.741311	97.31	61	39691	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_WIEGEP" 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
gm27wkxr9hyzd	16822	.220019	12.680465	11.976031	94.44	57	33644	655	98.09	SELECT "RFCDES", "LIMBO", "RFCTYPE", "RF
c24kpbckz3qxd	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

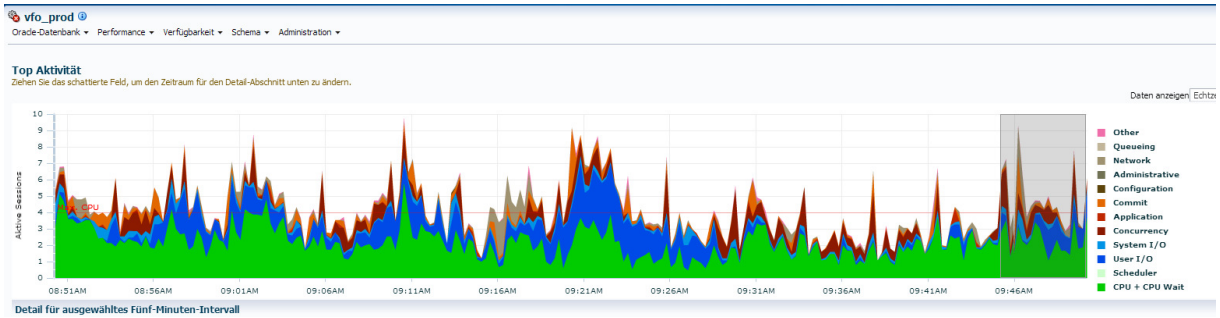
Wait Event Analyse
AWR Report oder PERFSTAT

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

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

Wait Event Analyse
Beispiel im Cloud Control 12c





I/O Performance Analyse
Probleme dabei

- Infrastruktur – vor allem Storage – wird von vielen Systemen genutzt
- Analyse erfolgt oft mit zu grober Auflösung
  - Oracle Snapshots 1 Stunde
  - Storage Performancedaten oft 5 Min oder gar 15 Min
  - OS Monitoring Sekunden oder Minuten
- Wer mit „Mist misst, misst Mist“
  - Irgendwelche Tools, Benchmarks, Utilities,… – alles liefert Werte, aber diese passen nicht zu Ihren Anforderungen!

Engpasserkennung und Lösungsmöglichkeiten Überblick

- I/O
  - Wenn der I/O Calibrate nur noch wenige IOPS, MB/s liefert
  - Wenn die I/O Distribution Ihren Peak jenseits der 16ms hat
- Memory
  - Wenn der Memory Advisor sagt: alles in Ordnung
- CPU
  - Die CPU nur noch im WAIT steht
- Konfigurationsfehler
  - Nur ein HBA genutzt wird
  - Falsche CPU Technologie / Hardwaretechnologie gesetzt wird

Engpasserkennung und Lösungsmöglichkeiten I/O und Memory (1)

- permanenter Read I/O = schlechtes Applikationsdesign und Implementierung oder zu wenig Memory
  - Cache Hit Ratio > 99% bedeutet nicht, dass alles in Ordnung ist
  - Gegen schlechtes Applikationsdesign können wir kurzfristig nichts machen
  - Aber zu wenig Memory muss nicht sein!

Früher	Jetzt
Ziel „Cache Hit Ratio“ > 98%	Reduktion des Read I/Os auf ein Minimum
Ziel „Read : Write“ = 20 : 1	Im Betrieb: weniger Read als Write

Engpasserkennung und Lösungsmöglichkeiten I/O und Memory (2) – was bringt das?

- Oracle dNFS ist effektiver und schneller als SAN Anbindung!
  - Sofern das Infrastrukturlayout passt
- Jeder nicht gemachte I/O reduziert den CPU Verbrauch!
  - Weniger Waits
  - Weniger Context Switches
  - Weniger OS Overhead (unbedingt Direct I/O + ASYNC!)

Was bringt das?

  - Alle SQL Statements, die aktuell noch I/O machen sind dann deutlich schneller.

Engpasserkennung und Lösungsmöglichkeiten CPU

- Nutzung der passenden CPU Technologie für Ihre Anwendung
  - Intel XEON, Power = DWH, DSS, Mixed und Single Thread Performance
  - Sparc T4/T5 = WebServer, viele Sessions mit vielen Statements die wenige Datensätze liefern
- Wählen Sie CPUs mit möglichst vielen GHz und nicht vielen Cores!
  - Aufpreis für 2 Xeon E5-2643v2 (6 Core á 3.5Ghz = 42Ghz) statt E5-2450v2 (8 Core á 1.9GHz = 30GHz) sind 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)!

Engpasserkennung und Lösungsmöglichkeiten Konfigurationsfehler = Troubleshooting

Beispiele aus unserer Erfahrung

- Virtualisierung, Oracle schafft nur wenige 100 IOPS → es war nur ein HBA aktiv, 3 HBAs waren deaktiviert
- Die Applikationsperformance schwankt ohne Grund stark → Hyperthreading war aktiv, je nachdem auf welcher „Core“ die Verarbeitung gelaufen ist war der Laufzeitunterschied Faktor 4-5
- Backup / Restore dauert ewig → weil das Backup über ein Gbit LAN gemacht wurde, statt über eine sinnvoll schnelle Anbindung
- …

Tuning
Schritt 2 – Proof of Concept

- Auswahl von „Kandidaten“
- Umsetzung
- Ergebniskontrolle
- Pitfalls

Auswahl von Kandidaten

- Grundsätzlich wird Oracle mit mehr Memory sicher nicht langsamer (sofern man alles korrekt konfiguriert hat), aber sinnvollerweise nimmt man Datenbanken, die aktuell einen höheren I/O Wait Anteil haben.
- Sofern man keine Möglichkeit hat, mit der Applikation sinnvoll zu testen, sucht man sich einfach im AWR Report oder Statspack die TOP Statements heraus und nimmt diese als Basis.

Umsetzung

- Besorgen Sie sich einen Server mit möglichst schnellen CPUs und viel Memory (Leihstellung ist auch eine Möglichkeit!)
- Immer 3 Durchführungen, alle Messwerte sind interessant!
  Ermitteln Sie die Laufzeit der Statements, während Sie gleichzeitig den I/O, den Memory Verbrauch und die CPU Auslastung monitoren:
  - auf Ihrem aktuellen System
  - auf dem neuen System bei gleichen Instanzparametern
  - auf dem neuen System mit möglichst großem Buffer Cache (maximal DB Size / 2)

Ergebniskontrolle

- Durch die unterschiedlichen Laufzeiten auf einem System sehen Sie die Auswirkungen von Caching (Infrastruktur und Oracle)
- Auf einem unter Last stehenden System sind die Laufzeiten immer schlechter, weil
  - I/Os durch Queueing länger brauchen
  - Man öfter ins UNDO schauen muss (read consistency)
  - Applikationsconcurrency dazu kommt
  - Die Clients nicht unendlich schnell die Daten abholen können (oder im schlimmsten Fall single row fetches machen)

Pitfalls

- Die Ausgabe des Ergebnisses verfälscht die Messung genau so wie der Unterschied von Client/Server zu direkt am Server.
  - Empfehlung: SQLPLUS mit SET AUTOTRACE TRACEONLY STATISTICS von einem Client aus
- Messungenauigkeiten – bei Laufzeiten von unter 0.1 Sekunde sind oft die Messungenauigkeiten von Oracle schuld, wenn man sich nicht wirklich verbessert
- Rückschluss: das was für die Teststatements gilt, gilt auch für den Rest der Applikation – die Test Statementauswahl waren die TOP Statements, für andere Statements ist der Vorteil geringer!

Tuning
Schritt 3- die Umsetzung

- Evaluieren Sie, welche Oracle Datenbank Edition wirklich benötigt wird
- Versuchen Sie aus dem Proof of Concept und durch weitere Analysen mögliche Engpässe zu identifizieren, die Sie mit der neuen Infrastruktur und Server Hardware ausmerzen können.
- Bei der Server Hardware
  - Lieber mehr GHz als mehr Cores (was zählt ist Cores * GHz/Core)
  - So viel Memory wie sinnvoll leistbar
  - Aktuelle Anbindungen nutzen (16GB SAN, 10GB NAS)
  - NAS für Oracle überlegen (dNFS ist schneller bei weniger Overhead)
- Nutzen Sie Konsolidierungsmöglichkeiten und überlegen Sie Virtualisierung (auch als HA Lösung, aber Vorsicht – Lizensierungsfallen!)

Beispiele aus der Praxis (1)

- Bei einem Kunden mit SAP (ca. 40-50 Instanzen), dessen Storage auf Grund der I/O Last „am Ende“ war, haben wir
  - Die I/O Last um über 30 % reduziert
  - Die CPU Last je nach Datenbank um 10-30% reduziert
  - Die Antwortzeiten bei einigen Verarbeitungen um den Faktor >10 gesteigert
- Investitionskosten
  - 4 neue Server und Memory Upgrades für einige alte Server
  - Meist musste nur die SGA angepasst werden
- Kostenersparnis
  - Viele 100k Euro, weil keine größere Storage gekauft werden musste.

Beispiele aus der Praxis (2)

- Kunde massiv unterlizensiert, weil für jede Datenbank ein eigener, neuer Server verwendet wurde ohne auf die Oracle Lizenzen Rücksicht zu nehmen.
  - Umsetzung mit Konsolidierung auf Oracle VM
  - Einige Applikationen sind jetzt 2-3 mal schneller, einige nur unwesentlich
- Investitionskosten
  - Zwei neue Server, Oracle VM
  - Nachlizensierung der noch fehlenden Lizenzen
- Kostenersparnis
  - Mehrere Millionen Euro, hätte ein Lizenzaudit stattgefunden

Zusammenfassung

Kann man sich klassisches Statement/Applikationstuning ersparen?

- Nein, definitiv nicht!

Wie spart man sich jetzt Geld?

- Durch Reduktion der Oracle Lizenzkosten und der Anforderungen an die Infrastruktur (IOPS)

Und was war jetzt das Tuning?

- Die heutige Hardware kann viel, viel mehr Hauptspeicher nutzen als noch vor einigen Jahren → dadurch erspart man sich I/O und wird gleichzeitig schneller

Q & A