Glücksspiel Systemstatistiken –
das Märchen vom typischen Workload

Thorsten Grebe, November 2012

Schlüsselworte

Systemstatistiken 11.2.0.3.

Einleitung

Damit der kostenbasierte Optimizer optimale Ausführungspläne berechnen kann, benötigt er diverse Statistiken. Am wichtigsten sind von diesen die Objektstatistiken, mit denen die aktuelle Befüllung von Tabellen und Indizes charakterisiert wird. Im günstigen Normalfall muss sich der Administrator um diese nicht kümmern, sie werden täglich während des Standardwartungsfensters automatisch aktualisiert. Darüber hinaus gibt es die *Dictionary*-Statistiken und die *Fixed Table* Statistiken. Diese müssen in der Regel nur selten aktualisiert werden. Auch diese Statistiken sind Objektstatistiken, nur dass sie nicht Benutzertabellen, sondern Oracle datenbankeigene Systemtabellen charakterisieren. Was dem Optimizer darüber hinaus jedoch noch fehlt, sind Kenndaten für die Hardware, auf der die Datenbankinstanz läuft. Wie schnell ist die CPU, um Dekomprimierungen, Sortieroperationen oder Hashumwandlungen durchzuführen? Wie viele I/O Operationen können die Festplatten pro Sekunde liefern, falls viele Datensätze über Indexzugriffe identifiziert werden sollen? Wie hoch ist der Durchsatz bei einem Segmentscan, falls ein Index oder eine Tabelle in Gänze eingelesen werden soll? Ohne Messung hat der Optimizer von diesen Kenndaten keine Kenntnis und kann nur provisorische Vorgabewerte verwenden. Und genau hier kommen die Systemstatistiken ins Spiel, um die es in der Präsentation gehen soll. Die aktuell gesetzten Systemstatistiken lassen sich über die Tabelle *sys.aux_stats$* auslesen:

```
SQL> select rownum as rn, a.* from aux_stats$ a;

RN SNAME          PNAME      PVAL1  PVAL2            [Kommentar]
-- -------------- ---------- ------ ---------------- -------------------------------------
 1 SYSSTATS_INFO  STATUS            COMPLETED        [MANUAL-, AUTOGATHERING]
 2 SYSSTATS_INFO  DSTART            08-29-2012 18:47 [Messung-Startzeit]
 3 SYSSTATS_INFO  DSTOP             08-29-2012 18:47 [Messung-Ende, bei NoWorkload=DSTART]
 4 SYSSTATS_INFO  FLAGS           1                  [0=gelöscht, 1=gesetzt]
 5 SYSSTATS_MAIN  CPUSPEEDNW   1386                  [NoWorkload: Mio Ops/Sek pro CPU]
 6 SYSSTATS_MAIN  IOSEEKTIM      10                  [NoWorkload: Zugriffszeit in ms]
 7 SYSSTATS_MAIN  IOTFRSPEED   4096                  [NoWorkload: Durchsatz in KB/sec]
 8 SYSSTATS_MAIN  SREADTIM                           [Zugriffszeit in ms für Einzelblock]
 9 SYSSTATS_MAIN  MREADTIM                           [Zugriffszeit in ms für Multiblock]
10 SYSSTATS_MAIN  CPUSPEED                           [Workload: Mio Ops/Sek pro CPU]
11 SYSSTATS_MAIN  MBRC                               [Durschnittl. Zahl Blöcke bei FTS, FFS]
12 SYSSTATS_MAIN  MAXTHR                             [Max. Dursatz in Byte/sec]
13 SYSSTATS_MAIN  SLAVETHR                           [Durschn. Dursatz in B/s für PX-Prozesse]
```

13 Zeilen hat die Tabelle, die ersten vier (`SNAME=SYSSTATS_INFO`) geben Auskunft über den Status der Statistikerhebung (Zeile 1, `STATUS=COMPLETED`), den Start (Zeile 2, `DSTART=29.8.2012 18:47`) und das Ende (Zeile 3, `DSTOP=29.8.2012 18:47`) der Erhebung mit einer minutengenauen Auflösung (bei *No-Workload*-Statistiken ist `DSTOP` immer gleich `DSTART`). Bei den meisten Systemen dürfte dieses Datum dem Geburtstermin der Datenbank entsprechen. Der Wert `FLAGS` (Zeile 4) kann auf "0" oder "1" stehen: "0" bedeutet die Werte wurden über `dbms_stats.delete_system_stats` auf Default-Einstellungen zurückgesetzt, "1" bedeutet, dass eine Messung mit `dbms_stats.gather_system_stats` ausgeführt worden ist oder mit `set_system_stats` mindestens ein Wert gesetzt wurde. Die nächsten drei Werte (Zeile 5-7) sind die sogenannten *No-Workload* Statistiken, die Oracle automatisch setzt. *No-Workload* soll dabei bedeuten, dass der Datenbankserver in dem Moment der Erhebung nicht unter Last steht, der Messvorgang erzeugt selbst die notwendige Last, die zum Messen erforderlich ist. `CPUSPEED`**`NW`** ist die gemessene CPU-Leistung in Millionen Operationen pro Sekunde. Der Wert kann in der Nähe der tatsächlichen CPU-Taktung liegen, die bei Linux über „`cat /proc/cpuinfo`" oder unter Windows über „`wmic cpu get CurrentClockSpeed`" ausgelesen werden kann. Es ist aber auch möglich, Werte zu messen, die leicht über dieser oder deutlich unterhalb dieser liegen. Welche Operationen Oracle hier tatsächlich zählt, ist nicht dokumentiert. Das …**`NW`** bedeutet *No-Workload*, es gibt daneben auch einen Eintrag für `CPUSPEED` (Zeile 10), der unter Last ermittelt wird, also der tatsächlich während des „typischen" Workloads erreichten CPU-Geschwindigkeit entsprechen soll. `IOSEEKTIM` (Zeile 6) ist die gemessene Zugriffszeit auf den Festplattenspeicher in Millisekunden (ms). 10ms ist der Defaultwert, der bei einer neu erstellten Datenbank oder beim Löschen der Systemstatistiken als Standard-Latenz eingetragen wird. `IOTFRSPEED` (Zeile 7) ist der Wert für den Durchsatz der Storage in KB/Sekunde. 4096 ist der Defaultwert (= 4MB/s), der bei einer Neuinstallation oder beim Zurücksetzen der

Statistiktabelle über `dbms_stats.delete_system_stats` eingetragen wird. Oracle geht hier davon aus, dass die Storage bei einem *Full Table Scan* 4MB/s lesen kann, was der Leistung eines alten USB-Sticks entspricht und meist um das 5 – 10fache unterhalb der Realität liegen dürfte. Für die anderen Werte (Zeile 8-13) gibt es gar keine Einträge. Es handelt sich dabei um die sogenannten *Workload*-Statistiken:

`SREADTIM` = Zugriffsgeschwindigkeit in Millisekunden für den Einzelblockzugriff, z.B. bei Indexzugriffen.

`MREADTIM` = Zugriffsgeschwindigkeit in Millisekunden für den Multiblockzugriff, z.B. bei *Full Table Scans* (FTS) oder *Fast Full Index Scans* (FFS).

`CPUSPEED` = Millionen (nicht dokumentierte) Oracle-Operationen pro Sekunde unter „typischem Workload", die eine einzelne CPU verrichtet.

`MBRC` = Wie viele Blöcke werden bei einem Multiblock-Zugriff tatsächlich gelesen; finden während des Messzeitraums keine Segmentscans statt, bleibt der Wert leer.

`MAXTHR` = Wie hoch ist der maximal gemessene Durchsatz der Storage in Bytes/Sekunde bei parallelen Abfragen (andere Einheit als `IOTFRSPEED`), der Wert wird nicht immer gesetzt.

`SLAVETHR` = Wie hoch ist der gemessene Datendurchsatz von einzelnen Slave-Prozessen bei parallelen Abfragen in Bytes/Sekunde – werden während des Messzeitraums keine parallelen Abfragen ausgeführt, bleibt der Wert leer.

All dies wären natürlich sehr nützliche Informationen, von denen man sich wünscht, dass der Optimizer sie berücksichtigen würde. Diese Werte werden jedoch nicht automatisch ermittelt. Der Administrator muss sich selbst darum kümmern, dass sinnvolle Werte für die *Workload-Statistiken* eingetragen werden.

Für die Berücksichtigung der Tabelle `aux_stats$` durch den Optimizer gelten folgende Daumenregeln:

1. Wenn niemals eine Messung von *No-Workload-Statistiken* oder *Workload-Statistiken* durchgeführt wurde, oder die Statistiken gelöscht wurden, dann verwendet der Optimizer die sogenannten *Default-No-Workload Statistiken*.
   Für diese gilt:
   `CPUSPEEDNW` wird spontan gemessen, wobei zwischen einzelnen Messungen erhebliche Schwankungen auftreten können.

IOSEEKTIM wird auf den Standard 10ms gesetzt.

IOTFRSPEED wird auf den Standard 4096 KB/Sek gesetzt.

2. Wurden über `dbms_stats.gather_system_stats('NOWORKLOAD')` *No-Workload-Statistiken* ermittelt, dann werden diese in die Zeilen 5-7 eingetragen und vom Optimizer sofort verwendet. Nur Statements, die bereits geparst und optimiert im *Shared Pool* liegen, bleiben von der Neuberechnung unberührt.
   Wird die Messung als ungültig gewertet, weil z.B. gleichzeitig Kopieraktionen auf den Festplatten ausgeführt werden, dann geht es zurück zu Punkt 1 und es werden für `IOSEEKTIM und IOTFRSPEED` die Standardwerte 10 bzw 4096 gesetzt. Die gemessene CPU-Leistung wird dennoch eingetragen, möglicherweise mit einem viel zu niedrigen Wert, da die CPU während der Messung nur eingeschränkt zur Verfügung stand. In den ca. 20 Sekunden bis 5 Minuten, in denen die Methode `dbms_stats.gather_system_stats('NOWORKLOAD')` ausgeführt wird, sollte das System nach Möglichkeit nicht anders beschäftigt werden.

3. Wurden über `dbms_stats.gather_system_stats({START/STOP | INTERVAL})` *Workload-Statistiken* ermittelt, so werden diese vom Optimizer verwendet und eingetragene *No-Workload-Statistiken* werden ignoriert.
   Ob alle Werte der Zeilen 8 – 13 tatsächlich belegt werden, liegt nicht in der Hand des Administrators. Finden während des Messintervalls keine Segment-Scans statt, bleiben `MBRC` und `MREADTIM` leer. Ebenso bleiben `SLAVETHR` und `MAXTHR` leer, wenn keine parallelen Zugriffe stattfanden.

4. Fehlt einer der der Werte für sreadtim, mreadtim, mbrc oder cpuspeed, werden *Workload-Statistiken* als ungültig betrachtet und der Optimizer verwendet stattdessen *No-Workload-Statistiken* .

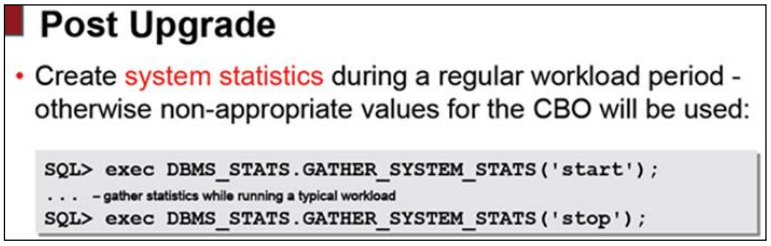
5. Können `MBRC` und `MREADTIM` nicht ermittelt werden, verwendet der Optimizer zum Berechnen der Kosten eines Tablescans den Init-Parameter `db_file_multiblock_read_count`, sofern dieser vom Administrator gesetzt wurde. Wurde er auf der Datenbankvorgabe belassen (meist 128) oder auf „0“ gesetzt, so wird bei der Kostenberechnung `MBRC=8` gesetzt (abgleitet von `_db_file_optimizer_read_count`).

6. Es gibt Wertekombinationen für ermittelte Workload-Statistiken, die Oracle als invalide identifiziert, z.B. wenn `mreadtim` (Zeile 9) nicht größer er ist als `sreadtim` (Zeile 8).

Wie sollte man Workload-Statistiken ermitteln?

Oracle gibt im *Performance Tuning Guide* und im *Upgrade Best Practices Guide* den Rat, *Workload-Statistiken* zu sammeln, damit der Optimizer für seine Kostenberechnung realistische Werte verwendet. Wenn es um die konkrete Anleitung geht, wie man diese Werte ermitteln soll, dann spielt immer wieder der „typische" oder „repräsentative" Workload eines Systems die entscheidende Rolle. Man aktiviere das Sammeln der *Workload-Statistiken* vor dem Beginn dieses typischen Workloads mit `dbms_stats.gather_system_stats('START')` und beende die Datensammlung am Ende des typischen Workloads mit `dbms_stats.gather_system_stats('STOP')`.

Zitat aus dem *Upgrade Best Practices Guide:*



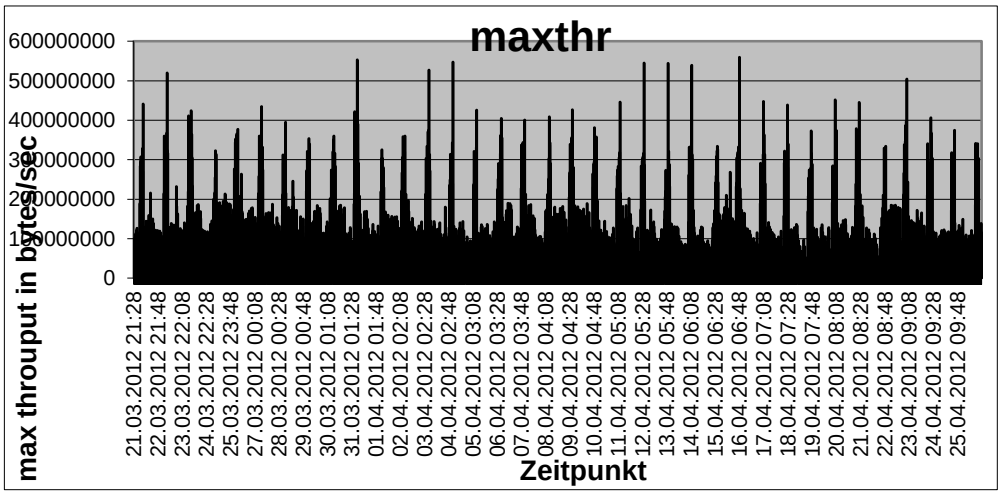
Befolgt man diese Anleitung, sieht man sich mit folgenden Schwierigkeiten konfrontiert:

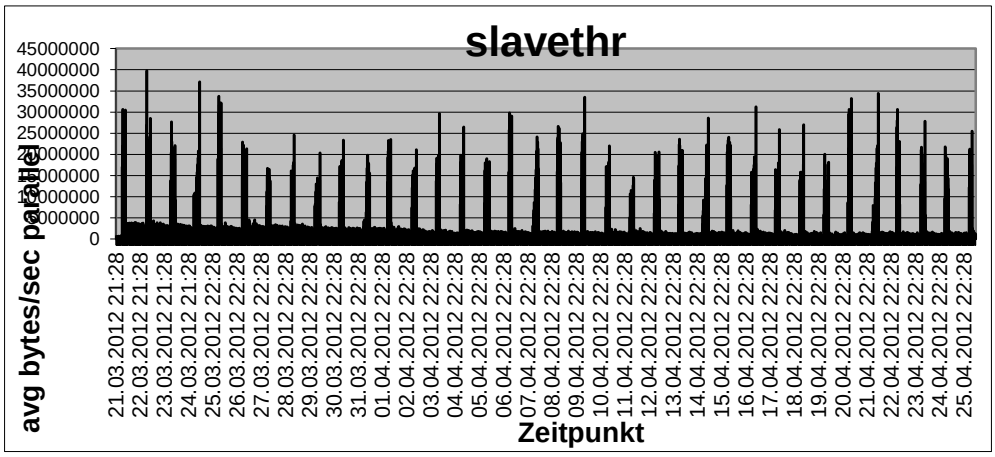
- Häufig ist unklar, wie der typische Workload eines Systems aussieht und somit auch, wann ein günstiger Startpunkt für eine Messung wäre.
- Ebenso unklar ist die optimale Länge des Messintervalls. Soll die Sammlung über wenige repräsentative Minuten oder eher über mehrere Stunden erfolgen.
- Mit der in beiden Ratgebern angegebenen Methode werden die gemessenen Werte sofort in die Tabelle `aux_stats$` eingetragen und sind sofort für den Optimizer relevant, ohne die Werte vorher auf Plausibilität prüfen zu können.
- Unklar ist bei dem empfohlenen Vorgehen, wie die erhaltenen Zahlenwerte auf Qualität untersucht werden können. Es bleibt stets die Unsicherheit, ob die in `aux_stats$` eingetragenen Werte realistisch oder unsinnig sind.

Möchte man diese Schwierigkeiten umgehen, bleibt dem Administrator nichts anderes übrig als eigene Messreihen zu entwickeln, auszuwerten und schließlich manuell über die Methode `dbms_stats.set_system_stats` editierte Werte in die Statistiktabelle einzutragen. Zu Beginn der Ermittlung geeigneter Workload-Statistiken ist dabei in der Regel erst einmal unklar, wie der typische Workload eines Datenbankservers aussehen mag – besonders dann, wenn man als externer Berater hinzugezogen wird. Wann sind am Tag oder nachts Lastspitzen zu erwarten? Wiederholt sich dieser

Workload täglich oder wöchentlich? Wann finden parallele Zugriffe, wann Tablescans statt, damit Werte für `MBRC`, `MREATIM` und `SLAVTHR` erhalten werden? Ebenso unklar ist, wie groß das Messintervall gewählt werden soll. Genügt es über ein paar Tage 4h-Intervalle zu messen, oder sollten die Intervalle auf 10 Minuten begrenzt werden?

In der Präsentation wird beispielhaft gezeigt, wie mit Hilfe des Job Schedulers über eine Woche 10minütige Messintervalle aufgezeichnet und in einer Zwischentabelle gespeichert werden (erstellt über `dbms_stats.create_stat_table`). Es findet praktisch keine Systembelastung statt, da pro Messung lediglich 2 Zeilen mit ein paar Messpunkten aus InMemory-Performanceviews in die Protokolltabelle eingetragen werden. Es wird noch nichts in `aux_stats$` eingetragen, so dass während der Messung keine Beeinflussung des Optimizers stattfindet. Die Statistiktabelle enthält am Ende der Messung mehrere Hundert Datensätze, die weniger als 1MB Platz verbrauchen. Anschließend werden die Messreihen aus der Zwischentabelle analysiert, um sinnvolle Systemstatistiken zu extrahieren, die manuell über `dbms_stats.set_system_stats` in die Tabelle `aux_stats$` eingetragen werden. Als Graphen dargestellt zeigen die Messreihen ein präzises Abbild des Workloads – eben genau den viel zitierten typischen Workload, hier als Beispiel für `maxthr` und `slavethr`:





Das Erstellen solcher Graphen ist mit einem gewissen Aufwand verbunden. Doch wie wertet man solche Messreihen am elegantesten aus? Maximalwerte nehmen, Durchschnittswerte berechnen? Woran erkennt man ungültige Messungen oder Ausreißer? Und was ist von dem Vorschlag zu halten, mehrere Sätze von Systemstatistiken vorzubereiten, um diese nach Bedarf zu rotieren? Wie schwerwiegend beeinflusst man den Optimizer, wenn man versehentlich unsinnige Werte einträgt? Und lohnt dieser ganze Aufwand überhaupt oder sollte man von vornherein die Finger von den Systemstatistiken und alles beim Default lassen? Was empfehlen die Experten? Mit diesen Fragen wollen wir uns beschäftigen.

Kontaktadresse:

Dr. Thorsten W. Grebe
twg-it Berlin
Geisenheimer Straße 6
D-14197 Berlin

Telefon: +49 (0) 30 2160 2272
E-Mail thorsten.grebe@twg-it.de
Internet: www.twg-it.de