DOAG 2011 Konferenz + Ausstellung

AS-SYSTEME

Optimierung von Solaris für Oracle DB

Christian Wolbert

Optimierungsziele

- Reduktion der Antwortzeiten / Latenzzeiten
- Effiziente Ressourcenauslastung
- Vermeidung von negativen Interaktionen zwischen Betriebssystem und Datenbank
- Eliminierung von Fehlern

Voraussetzungen

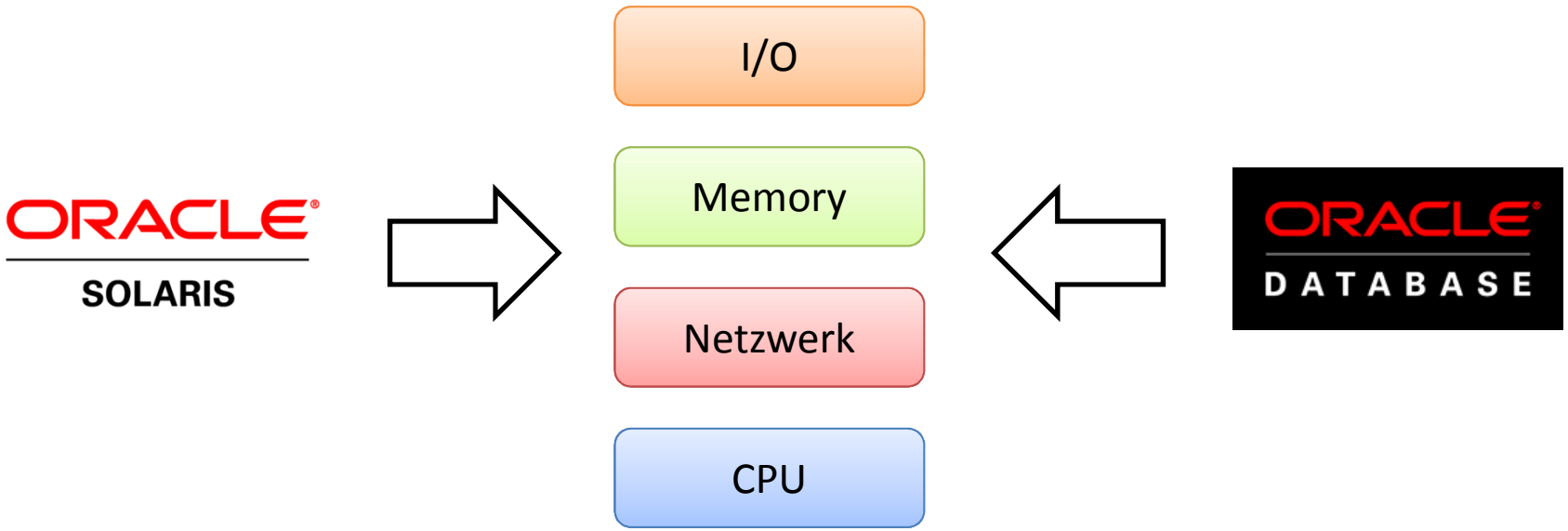
- Die Datenbank ist bezüglich ihres spezifischen Anwendungsaspektes entsprechend konfiguriert.

  db_block_size. Im OLTP mit vielen Transaktion, ist eine kleine db_block_size (8KB) optimal, im Datawarehouse mit großen Tabellen kann von größerer Blocksize profitieren.

- Fehlerfreies Arbeiten

  "checkpoint has not completed" ; fehlende Indexe

Optimierungsbereiche



Teil 1: I/O

I/O

I/O

Der I/O-Bereich häufig der kritischste Bereich.
Verschiedenste Aspekte beeinflussen die Performance:

- Servicetime
- Sättigung
- Anzahl der I/O - Operationen
- Subsystemlayout
- Redundanz

Diskutilities

Neben den klassischen Werkzeugen liefert vor allem Dtrace detailliertere, Prozessspezifische Informationen.

- # iostat
- # iosnoop (Dtrace Tool Kit)
- # iopattern (Dtrace Tool Kit)
- # bitsize.d (Dtrace Tool Kit)
- # sar

Allgemeine Empfehlungen

- SSD verwenden für: Index, Redologs, temporäre Tablespaces, Rollback data
- Logs separat halten
- Kontrolldateien spiegeln
- Caches verwenden

SSD Performance

- 4K – Alignment

Neuere SSD's verwenden 4K physikalische Cluster-größen und emulieren 512 Byte Sektoren. EFI-Label (34 Sektoren) sorgt für Versatz zwischen logischen und physikalischen Blöcken: -> Performance-Einbußen.

Optimierung: Slice 0 mit n*4K Offset anlegen und unter ZFS einbinden.

SSD Schreibalgorithmus

- Speicherzellen nur begrenzt wiederbeschreibar
- Read-Modify-Write Algorithmus daher häufig komplex -> geringe Performance

Solaris 10 Update 10

- Firmware-Algorithmus ausschalten
- sd.conf -> Parameter: emulation-rmw

Layout-Empfehlungen

Übersicht der *Oracle Managed Files* und das empfohlene Subsystemlayout

OMF	Empfohlenes Layout
Database	Raid 1+0, HW-Raid5
Redolog	Raid 1
Archivelog	Raid 1
Index	Raid 1+0
Temp data	Raid 1+0
Control file	Raid 1, Raid 1+0

ZFS-Pool

ZFS besitzt integrierte Volumemanagerfunktionalitäten. Das Layout wird auf Poolebene definiert. Aufgrund der allgemeinen und Layout-Empfehlungen sollten folglich zwei, respektive drei Pools erzeugt werden.

- Datenpool
- Redopool
- Archivepool

ZFS – Poolerweiterungen

Zusätzlich zu den Festplatten zur Datenspeicherung, kann der Pool um 2 Komponenten erweitert werden:

- Intent Log-Devices (ZIL)
- Cache Devices

Für diese Komponenten empfiehlt es sich Solid State Disks (SSD) zu verwenden.

ZFS – Intent Log

ZIL: # zpool add pool log mirror <SSD1> <SSD2>

- ZFS schreibt asynchron die Daten heraus
- Ausnahme: Applikation verlangt eine synchrone Transaktion
- Problem: Bei vielen random synchronen Transaktionen sinkt die Performance
- Intent Log: Synchrone Transaktionen werden sequentiell auf den Log geschrieben
- Log-Devices spiegeln!

ZIL - Empfehlungen

- Nachteil: Die Anzahl der I/O's erhöht sich
- Pro Dateisystem kann man erwägen, ob es sich lohnt
- Einstellung über den Parameter: logbias
- Wert throughput: ZIL-devices nicht nutzen.

Transaktionen direkt ins Dateisystem

- Wert latency: nutzt Logdevices

Data Files	Redo Logs	Index Files	Undo Data	Temp Data	Archive Data
throughput	latency	throughput	throughput	throughput	throughput

Zpool – Cachedevices

zpool add pool cache <SSD1>

(Secondary Cache)

- Performancegewinn bei Leseintensiven Arbeitslasten
- Speichert keine Daten über einen Neustart hinweg
- Kein Vorteil bei:
  - Redopool
  - Archivepool

ZFS – Cache

Cache-Nutzung von Primary Cache (RAM) und Secondary Cache (Devices) sind pro Dateisystem einstellbar:

- all (metadata und data)
- metadata
- none

Empfohlene Primary Cache Settings:

Data Files	Redo Logs	Index Files	Undo Data	Temp Data	Archive Data
all	all	all	metadata	all	metadata

ZFS Pool %Usage

- Änderung der Schreibstrategie wenn 80% vom Pool belegt sind. Negativ bezüglich der Performance.
- Änderung des Prozentwertes über Kernelparameter metaslab_df_free_pct:

  /etc/system:

  set zfs:metaslab_df_free_pct=10

- Alternativ die Verwendung von Quotas in Betracht ziehen.

ZFS - recordsize

- Default Recordsize 128 KB
- Änderung wirkt sich nur auf neue Objekte aus!
- Datenbank: db_block_size = ZFS recordsize

Beispieltabelle für recordsize-Werte, wenn db_block_size = 8 KB

Data Files	Redo Logs	Index Files	Undo Data	Temp Data	Archive Data
8 KB	128 KB	8 KB	128 KB/8 KB	128 KB	128 KB

Für Archive Data: compression=on

Hinweis: Legacy Storage

- Problem: I/O-Begrenzung desTreibers bezüglich der maximalen Anzahel der Verbindungen
- Storage-Administrator verlangt Throttle-Mechanismus
- Unnötiges Queing vom ZFS-Layer zum darunterliegenden Layer vermeiden.

Beispiel:

/etc/system

set ssd:ssd_max_throttle= 20 wäre gesetzt

set zfs:zfs_vdev_max_pending = 20 hinzufügen

Teil 2: Memory

Memory

Speicher

Oracle verwendet Speicher unterschiedlich.

- SGA (System global area): data buffer, redo buffer, buffer cache, java pool, …
- PGA (Program global area): Speicher, den jeder einzelne Oracle Server für sich nutzt
- Anforderungen unterschiedlich: OLTP benötigt mehr SGA, Datawarehousebereich mehr PGA

Speicherverwendung

Analyse und Überwachung der Speicherressourcen:

- # ipcs -im
- # pmap –xs <pid>
- # prctl
- # prstat -J | -Z (Projekte|Zonen)
- # rctladm

Bei paging / swapping, swap auf SSD legen.

Ressource Management

Ein Projekt mit Ressourenkontrollen wird für eine Datenbank benötigt:

- project.max-shm-memory
- project.max-constracts
- project-max-device-locaked-memory
- project.max-locked-memory
- project.max-ports-ids
- rcap.max-rss

Multiple Page Sizes

- Support für große Speicherseiten (# pagesize –a)
- Defaultgröße: 8K (Sparc), 4K (x86)
- # ppgsz | mpss.so Library vorschalten
  - Schlankere Speicherverwaltung
  - Effizientere Ausnutzung der Cachestrukturen
  - Reduktion der Übersetzungen von physikalischen zu virtuellen Adressen (Translation Lookaside Buffer: *# trapstat -t*)

Teil 3: Netzwerk

Netzwerk

Netzwerk

Kritische Aspekte:

Paketgröße, Durchsatz, Retransmissions/Duplikate

Oracle DB interne Parameter:

- SDU (Session Data Unit): Größe der gesendeten Pakete (default 2048)
- TDU (Transport Data Unit): Größe der Pakete in Oracle*Net
- Schnelle Netzwerke: MTU=SDU=TDU

Jumbo-Frames

- Jumbo-Frames MTU > 1500
- Konfiguration:

```
# vi /kernel/drv/ixgbe.conf
default_mtu=9000

# dladm set-linkprop –p mtu=9000 ixgbe0
```

Drops

Wenn die Anwendung nicht schnell genug die Anfragen bearbeiten kann, erfolgen Paket-Drops.

- # netstat –sP tcp |grep tcpListenDrop
- # kstat –ps tcp_listendrop

Tuningparameter:

tcp_conn_req_max_q (Maximale Anzahl der ausstehenden Verbindungen)

Parameter in /etc/system erhöhen (kostet Memory)

Aggregate

Netzwerkkarten aggregieren für erhöhten Durchsatz.

dladm create-aggr …

- höhere Brandbreite
- Redundanz
- Loadbalancing

Alternative: IPMP

IP Parameter: ip_squeue_fanout

- Parameter unter Solaris 10 verfügbar
- squeue: Assoziation einer TCP/IP-Verbindung mit einer CPU
- Default: Die CPU, die den Interrupt der Netzwerk-karte verarbeitet, bindet die squeue an sich: Es kann eine Ungleichverteilung der squeues geschehen.
- Optimierung: Anzahl CPU's > Anzahl NIC's => Wert auf 1 setzen.

Solaris 11: Crossbow

Crossbow-Projekt: Redesign des Netzwerkstacks

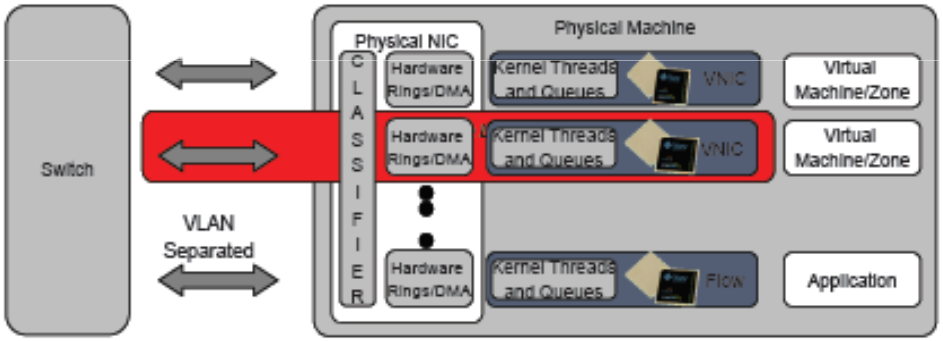
- Komplette Abbildung eines Netzwerkstacks im Host möglich
- Virtuelle Netzwerkkarten
- Bandbreitenregulierung
- Flowmanagement

Hardwarelanes

Hardwarelane:

- Dedizierte Verbindung NIC <-> CPU
- Anbindung CPU-Sets, CPU-Pools ebenso möglich

```
# dladm set-linkprop –p
pool=ora_pool vnic1
```



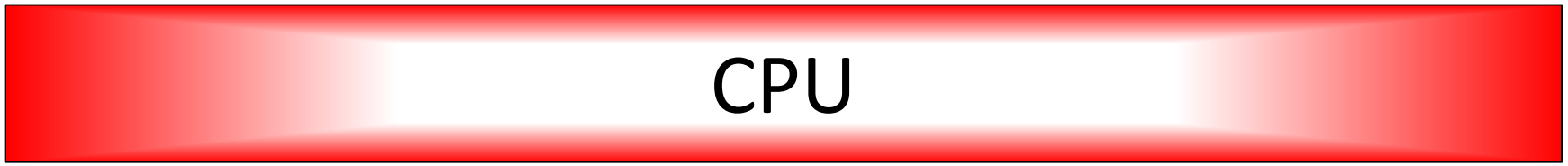
Bildquelle: Oracle

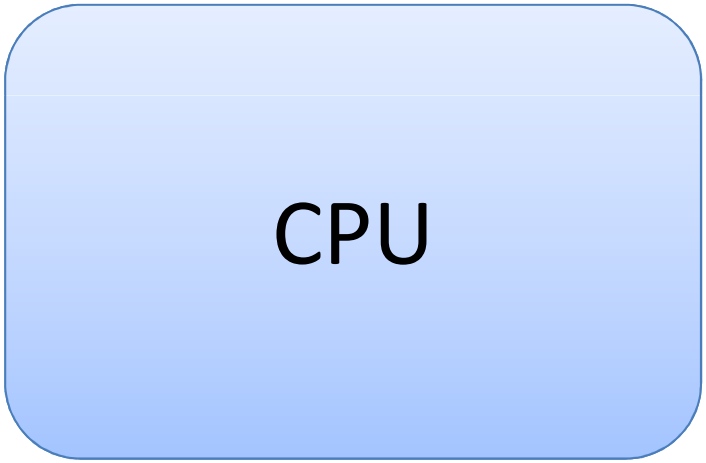
Crossbow-Performance

- Über Hardwarelanes direkte Verbindung zwischen Anwendung zu NIC assoziierbar: Isolation des Netzwerkverkehrs!
- Falls NIC noch Dynamic Polling (Pollingrate an erwarteten Netzwerkverkehr anpassen) unterstützt:

Reduktion bis zu 30% bezüglich

- Interrupts
- context switches
- lock contentions


CPU


CPU

Resourcemanagement

Im Bereich CPU sind zwei wichtige Mechanismen des Resourcemanagements von Bedeutung:

- Partitionierung: Verwendung von Prozessorsets und CPU-Pools
- Gewichtung: Verwendung des Fair Share Schedulers zur Bestimmung der CPU-Anteile

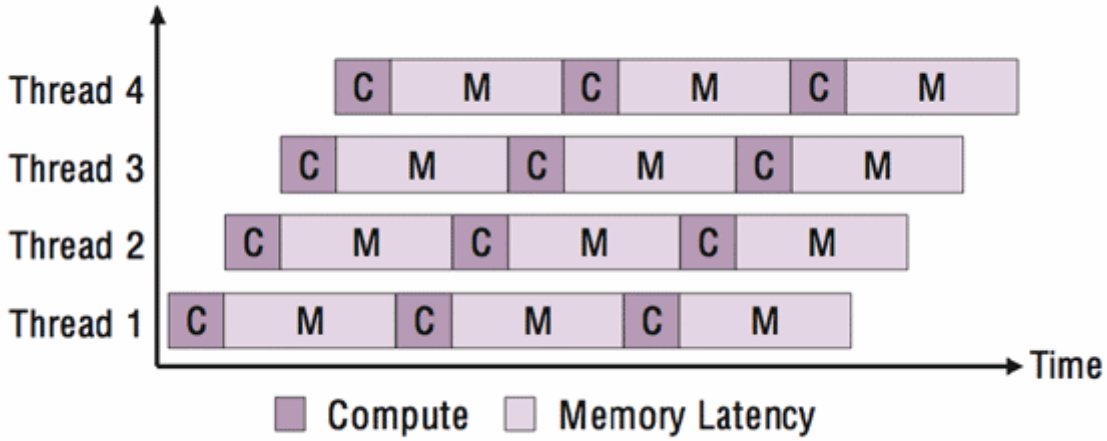
Scheduling

- Standardscheduler : TS (Time Sharing)
  - behandelt alle Prozesse gleich, gemäß der Last
  - kann keine Garantien abgeben
- Schedulingklasse: Fair Schare Scheduler (FSS)
- FSS-Shares definieren die CPU-Anteile
- Erlauben Mindestgarantien
- Nicht mit anderen Schedulingklassen mischen

CMT

Prozessoren mit Chip-Multi-Threading:

- Bessere Skalierung
- effektiv "kürzere" Latenzzeiten



Virtualisierung

Verschiedene Ebenen der Virtualisierung möglich:

- Betriebssystem: Zonen
- Hypervisor: Oracle VM Container for Sparc (Ldom)

Best Practice Tipp:

Zonen nutzen: -> Nativer I/O. (Beim Hypervisor kann die Service Domain den Engpaß darstellen.)

Zonenprivilegien: default,proc_lock_memory

Links

- Oracle Whitepaper - Configuring Oracle Solaris für Oracle Databases

  http://developers.sun.com/solaris/docs/wp-oraclezfsconfig-0510_ds_ac2.pdf

- Solarisinternals – ZFS für Datenbanken

  http://www.solarisinternals.com/wiki/index.php/ZFS_for_Databases

- Crossbow Projekt

  http://opensolaris.org/os/project/crossbow

Links

- Solaris Container Resourcemanagement Guide

  http://download.oracle.com/docs/cd/E19683-01/817-1592/index.html

- Oracle Database Performance Tuning Guide

  http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/toc.htm

- Oracle FAQ's

  http://www.orafaq.com/wiki/Oracle_database_Performance_Tuning_FAQ

Ende

"Experience shows that approximately 80% of all Oracle system performance problems are resolved by coding optimal SQL."