Oracle VM3: Threads, Cores, Hard-Partitioning; und die Performance?

Martin Bracher
Trivadis AG
Zürich

Schlüsselworte
OracleVM, XEN, CPU-Performance

Einleitung

OracleVM for x86, im Folgenden oft OVM genannt, ist die Virtualisierungslösung von Oracle, mit welcher Oracle in Konkurrenz zu etwa VMWare ESX tritt. OracleVM basiert auf einem Enterprise Linux mit der XEN Technologie zur Virtualisierung. OracleVM bietet zwei grosse Vorteile:

- Zertifizierung und Support für die Oracle Produkte (ID 464754.1)
- Hard-Partitioning: Die Möglichkeit, nur bestimmte CPU-Cores den VM's zuzuordnen, und dann auch nur diese lizensieren zu müssen (ID 466538.1)

Prozessor, Sockel, CPU, Core, Thread

In diesem Vortrag sind folgende Begriffe von zentraler Bedeutung:

Ein Mainboard besitzt sogenannte Sockel zum Einstecken von Prozessoren, in Servern oft 2 oder 4 Stück. Für die Lizensierung von Oracle Standard Edition ist die Anzahl Sockel massgebend, wobei nicht mehr wie 2 Sockel lizensierbar sind.

Ein solcher Prozessor besteht aus 1-N Prozessorkernen, genannt Cores, üblicherweise zwischen 4 - 24 Cores. Für die Lizensierung von Oracle Enterprise Edition ist die Anzahl Cores massgebend, die Anzahl ist nicht limitiert. Bei x86 (Intel/AMD) Prozessoren ist für 2 Cores eine Lizenz erforderlich.

Socket 0		Socket 1	
Core 0	Core 1	Core 0	Core 1
CPU0	CPU1	CPU2	CPU3

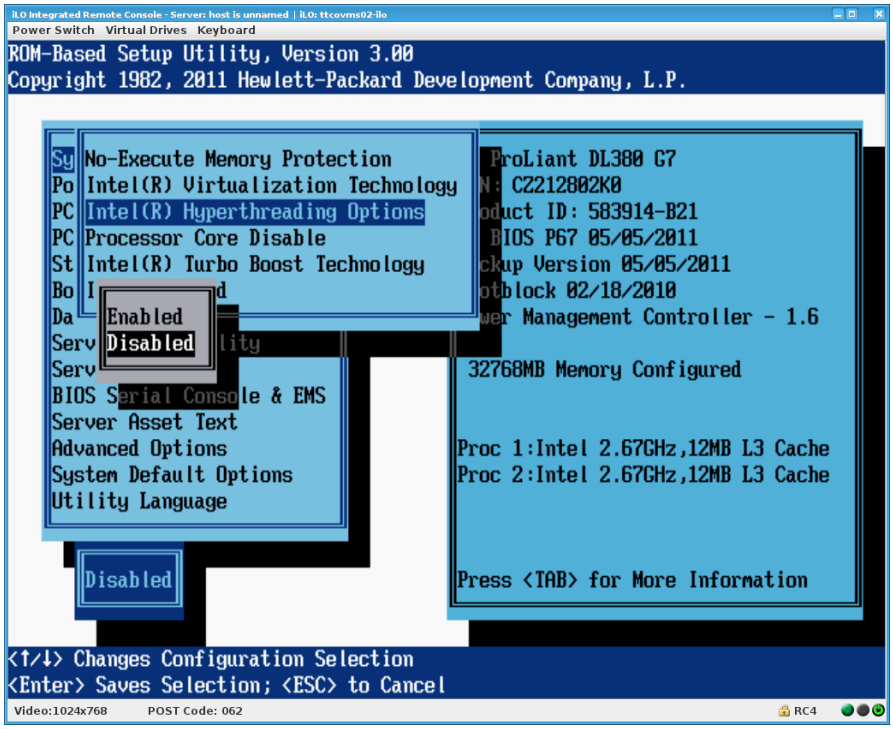
Im Betriebssystem sind bei einem 8 Core Prozessor 8 „CPU" sichtbar.

```
[root@ttcovms02 ~]# xenpm get-cpu-topology
CPU      core     socket   node
CPU0      0        0        0
CPU1      1        0        0
CPU2      9        0        0
CPU3      10       0        0
CPU4      0        1        1
CPU5      1        1        1
CPU6      9        1        1
CPU7      10       1        1
```

Intel-Prozessoren kann man auch mit Hyperthreading betreiben (kann im Server-Setup ein- oder ausgeschaltet werden). Dabei wird ein Core gegenüber dem Server als 2 unabhängige CPU präsentiert. Man spricht bei einer solchen CPU vonn einem Thread.

Socket 0				Socket 1			
Core 0		Core 1		Core 0		Core 1	
HT 0	HT 1	HT 0	HT 1	HT 0	HT 1	HT 0	HT 1
CPU0	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6	CPU7

Auf dem Server mit einem 8 Core Prozessor sind 16 „CPU“ sichtbar.




```
[root@ttcovms01 ~]# xenpm get-cpu-topology
CPU      core     socket   node
CPU0      0        0        0
CPU1      0        0        0
CPU2      1        0        0
CPU3      1        0        0
CPU4      9        0        0
CPU5      9        0        0
CPU6      10       0        0
CPU7      10       0        0
CPU8      0        1        1
CPU9      0        1        1
CPU10     1        1        1
CPU11     1        1        1
CPU12     9        1        1
CPU13     9        1        1
CPU14     10       1        1
CPU15     10       1        1
```

Was ist nun besser? Hyperthreading oder nicht? Dies werden wir im Vortrag genauer anschauen. Kurz zusammengefasst: es kommt drauf an. Es ist stark von der Hardware- und Software-Implementierung abhängig. Bestimmte Szenarien profitieren davon, andere nicht.

Im Bereich der Virtualisierung werde ich bei den „CPU“ (CPU0-CPU7 bei Cores oder CPU0-CPU15 bei Threads) von physischer CPU sprechen, auch wenn ein Thread eigentlich nur eine virtuelle CPU ist. Das für die reduzierte Lizensierung erforderliche Hard-Partitioning bezieht sich auf diese CPUs.

Innerhalb der VM sind virtuelle CPU zugewiesen. Eine virutelle CPU wird wechselnd auf eine möglichst freie physische CPU geschoben

Auf physischer Hardware ist klar. Der Kernel weiss, wie stark welche CPU ausgelastet ist und kann die Last entsprechend verteilen.

In einer virtuellen Maschine sieht der Kernel eine Anzahl von virtuellen CPU, die meist auf irgendeinem physischen Core bzw. Thread laufen können, oft neu zugewiesen werden und in der Regel mit anderen VM’s geteilt werden.

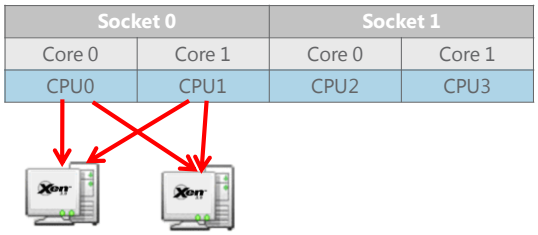
Wenn eine virtuelle CPU keine Last, die andere aber 60% Last hat, wird er diese unbelastete CPU verwenden. Wenn er aber Pech hat, wird der physische Core aber schon von einer anderen VM stark belastet.

Um das zu verhindern, versucht nun der Hypervisor, diese virtuelle CPU auf eine weniger ausgelastete physische CPU zu verschieben. Dieses Verschieben ist jedoch nicht gratis zu haben, es kann einen durchaus spürbaren Overhead darstellen. Man kann sich deshalb Strategien überlegen, um diese Wechsel möglichst gering zu halten.

Hard-Partitioning

Die Oracle-Datenbank wird nach „CPU“ lizensiert (dieser Lizenz „CPU“ Begriff ist nich zu verwechseln mit der oben beschriebenen Definition). Je nach Plattform und Edition bedeutet „CPU“ etwas anderes. Bei Intel/AMD Architektur benötigt man mit Enterprise Edition (EE) eine „CPU“-Lizenz pro 2 Cores, mit der Standard-Edition eine Lizenz pro Socket (wobei maximal 2 Sockets lizensiert werden können).

Massgebend sind nicht die benötigten, sondern physisch vorhandenen CPUs. Wenn ein Server nun mehr CPUs besitzt als für Oracle gebraucht werden, müssen trotzdem alle lizensiert werden. Hierbei gibt es jedoch eine Ausnahme: Bei OracleVM kann man den VMs explizit CPUs zuweisen. Dies wird als Hard-Partitioning bezeichnet. In diesem Fall sind nur noch die zugewiesenen CPUs zu lizensieren (Metalink ID 466538.1)



Für diese 2 VMs wird genau eine EE Lizenz benötigt.

Das Hard-Partitioning kann nicht direkt über das Web-Frontend, und auch nicht über das OVM Commandline-Interface gemacht werden. Es müssen zusätzlich die Oracle VM3 Utilities (Patch 13602094) installiert werden. Danach kann es mit dem ovm_vmcontrol Utility aktiviert werden.

```
ovm_vmcontrol –u admin –p pwd –h localhost –vm vmname –c setvcpu –s 0-3
```

Die Verteilung der vcpu der VM auf die physischen cpu wird dann auf die cpu 0-3 limitiert. Das Prinzip, eine möglichst wenig belastete CPU zu verwenden, bleibt bestehen, einfach limitiert auf die CPUs 0-3. Wie die aktuelle Vereilung aussieht, kann man sich mit diesem Befehl anschauen:

```
# xm vcpu-list 0004fb0000060000a980f351496991cc
Name                                ID  VCPU   CPU State   Time(s) CPU
Affinity
0004fb0000060000a980f351496991cc     4     0     1   -b-       1.0 1,3
0004fb0000060000a980f351496991cc     4     1     1   -b-       0.7 1,3
0004fb0000060000a980f351496991cc     4     2     3   r--      20.5 1,3
0004fb0000060000a980f351496991cc     4     3     1   ---       0.9 1,3
```

Wenn wir gleich viele vcpu spezifiziert, wie physische cpu zugewiesen haben, kann man sich überlegen, eine 1:1 Mapping zu definieren, um den Overhead des Suchens einer weniger ausgelasteten cpu zu vermeiden.

Die Variante mit dem 1:1 Pinning lässt sich nicht mit OVM konfigurieren. Dies muss man nach dem Start der VM auf dem OVM Server zur Laufzeit erledigen.

```
xm vcpu-pin <uuid> 0 0
```

Ist diese Variante besser? Auch hier gibt es keine eindeutige Antwort. Es hängt von der Hard- und Software ab... In einem Fall kann es kontraproduktiv sein, in einem anderen Fall jedoch eine spürbare Verbesserung bringen. Auf einem Kundensystem mit viel Last konnte durch das 1:1 Pinning der Produktiv-VM eine verbesserte Leistung von fast 1/3 festgestellt werden.

Viel Erfolg beim Einsatz von Trivadis-Know-how wünscht Ihnen

Kontaktadresse:
Martin Bracher
Trivadis AG
Sägereistrasse 29
CH-8152 Glattbrugg

Telefon: +41 58 459 56 56
Fax: +41 58 459 56 66
E-Mail martin. bracher@trivadis.com
Internet: www.trivadis.com