IO Performance in virtualisierten Umgebungen

Bruno Harsch | El. Ing. HTL/FH | Managing Partner
Tel +41 52 366 39 01 |bruno.harsch@idh.ch | www.idh.ch
IDH GmbH | Lauchefeld 31 | CH-9548 Matzingen

idh
The open system company

Die Firma IDH wurde 1996 gegründet

Schwerpunkt Systems Engineering Unix und Storage und sind damit vorwiegend im Enterprise Segment tätig

Ein Team von Engineer's mit langjähriger Erfahrung im Fachgebiet Unix und Storage

IDH – Wer sind wir?

idh
The open system company

Immer wieder gibt es Diskussionen ob ein Linux System nicht besser auf physischer Hardware betrieben werden soll um eine vernünftige Performance zu erreichen

Speziell im IO Bereich gibt es zusätzliche Latenzen, daher haben wir uns auf Vergleiche und Optimierungen im IO Bereich konzentriert

Wir haben in verschieden physischen und virtualisierten Umgebungen Vergleichsmessungen gemacht

Die Systeme waren über SAN an Enterprise Storage verbunden

Grenzen gesucht: IOPS, Bandbreite, Parallelisierung usw.

Zusammenhänge: IOPS, Bandbreite, Blockgrösse und Latenz

Ausgangslage

idh
The open system company

Gewünscht: max IOPS und max Bandbreite zusammen

Gegeben: Blockgrössen und Latenz

Gefordert: Tuning der Effekte des Gegebenen damit möglichst das Gewünschte erreicht wird

Anforderungen

Enterprise Storage über SAN vs. lokalem Diskarray vs. Flash Array (nicht SSD!)

Single LUN vs. Stripe (md,LVM)

Wenig parallelisierter IO vs. Hoch parallelisiert (e.g. Full Table Scan vs. Row Operationen)

Raw gemapptes Device vs. VMDK

Was haben wir verglichen

idh
The open system company

80% Random Read / 20% Random Write

Ohne FS direkt auf Linux Raw Devices, da wir nicht OS caching Effekte messen wollten. Diese sind bei kleinen Datenmengen dominierend (BTW: Bei Überforderung des FS Cache, haben wir einen overhead im kleinen einstelligen Bereich gemessen)

Beim einem konkreten Fall hatten wir viele Java threads welche IO generierten (Jboss)

VDBench ist ein Java Tool, welches die gleichen Eigenschaften hat und sich daher gut für diese Vergleiche eignete

Nur VMware als Virtualisierer (mangels anderen Umgebungen)

Wie haben wir verglichen

idh
The open system company

Bandbreite
Typisch zwischen wenigen MB/s bis 150MB/s

IOPS
150 für SAS bis 40'000 für Enterprise Storage mit viel Cache

Latenz
Typisch zwischen 2ms und 8ms für Enterprise Storage und 0.2ms bis 0.5ms für Flash

Blockgrössen
Extrem Workload abhängig
In DB Umfeld häufig wenige KB bis paar hundert KB

Begriffe & Zusammenhänge

IOPS x Blockgrösse = Bandbreite

z.B. SAS Disk 150 IOPS,16KB Blöcke
150 x 16KB = 2400 KB/s

z.B. Enterprise Storage, viel Cache 40'000 IOPS, 16KB Blöcke
40'000 x 16KB = 640 MB/s (theoretisch, nur möglich mit entsprechendem FC Attachment und viel cache)

Latenz bestimmt Anzahl IOPS bei einzelnem Paket
IOPS = 1/Latenz
z.B. 2ms -> 500 IOPS

Für mehr IOPS braucht es entsprechende Queue's, Parallelisierung und Sequenzen

Begriffe & Zusammenhänge

CFQ vs. NOOP IO Scheduler

CFQ

```
Oct 25, 2013  interval        i/o   MB/sec   bytes   read     resp     resp
resp    cpu%  cpu%
                             rate  1024**2     i/o    pct     time      max
15:38:35.106  avg_2-12   42456.71   165.85    4096  80.01    1.130 1015.295
```

NOOP

```
Oct 25, 2013  interval        i/o   MB/sec   bytes   read     resp     resp
resp    cpu%  cpu%
                             rate  1024**2     i/o    pct     time      max
15:43:57.059  avg_2-12   34169.24   133.47    4096  79.98    1.405  732.328
```

Bei hohen IOPS (40K) war NOOP mal schneller / mal langsamer

Bei kleinen IOPS (2k bis 4k) bringt NOOP ca. 30 % bessere Resultate

Ein paar Ergebnisse...

idh
The open system company

md Stripes vs. LV/FS Stripes

Wir bekamen bei beiden Varianten in etwa die gleichen Ergebnisse

Die Messschwankungen waren grösser...

Ein paar Ergebnisse...

idh
The open system company

VMDK vs. Raw mapped Device

Die VMDK Disk waren meist schneller (!), da der Cache im VMware Layer den stärkeren Einfuss hat, als die leicht erhöhte Latenz

Nur 1 VMDK / LUN

Ein paar Ergebnisse...

Im Enterprise Umfeld schwankt die Performance des SAN/Storage auch

Wir sehen Unterschiede zwischen Bürozeiten und Nachtzeiten

Tagsüber sind Antwortzeiten etwas länger, aber immer noch < 1ms Unterschied

Höhere CPU Last ausserhalb der Bürozeiten, da die IO Peformance besser ist und die CPU(s) mehr pro Sekunde schaufeln können

Phänomene

Bei wenigen IO Threads und höherer Latenz bleibt die Runqueue < 1
und es findet damit auf CPU Ebene keine Parallelisierung statt.
Da die IO's (gemütlich) sequentiell an VMware und Storage weiter gereicht wird, kommen keine bis wenige Caches zu Hilfe

Bei vielen IO Threads und trotz höherer Latenz steigt die Runqueue > 1
und es findet damit auf CPU Ebene eine Parallelisierung statt.
Da die IO's jetzt parallel kommen steigen die IOPS zuerst um x Anz. CPU (mind. 8 CPU's nehmen)
Im VMware Layer und Storage werden durch den stark erhöhten IO Caches aktiv und dadurch sinkt die Latenz (z.B. Von 2ms auf 1ms)
Dies bewirkt wiederum eine Verdoppelung der IOPS

Jetzt kann die Anzahlt IO Threads wieder reduziert werden. Die CPU's arbeiten weiter parallel durch die reduzierte Latenz.

Phänomene (Folding Problem?)

```
7:21:33.007 Starting RD=run1; I/O rate: Uncontrolled MAX; elapsed=60; For loops: threads=48.0

Oct 25, 2013  interval        i/o   MB/sec   bytes   read     resp     resp     resp    cpu%  cpu%
                            rate  1024**2     i/o    pct     time      max   stddev sys+usr   sys
17:22:33.066  avg_2-12   18912.73    73.88    4096  79.99    2.538  593.429   10.041     4.6   3.9

17:22:36.003 Starting RD=run2; I/O rate: Uncontrolled MAX; elapsed=60; For loops: threads=480.0

Oct 25, 2013  interval        i/o   MB/sec   bytes   read     resp     resp     resp    cpu%  cpu%
                            rate  1024**2     i/o    pct     time      max   stddev sys+usr   sys
17:23:36.058  avg_2-12   39055.47   152.56    4096  80.02   12.290 1923.890   20.471    17.6  15.6

17:23:38.004 Starting RD=run3; I/O rate: Uncontrolled MAX; elapsed=60; For loops: threads=48.0

Oct 25, 2013  interval        i/o   MB/sec   bytes   read     resp     resp     resp    cpu%  cpu%
                            rate  1024**2     i/o    pct     time      max   stddev sys+usr   sys
17:24:38.108  avg_2-12   40382.42   157.74    4096  79.97    1.188 1299.861    7.700    13.7
12.4.
```

Phänomene ... die Zahlen dazu

idh
The open system company

Experiment mit 2ms „künstlicher“ Latenz (dm_delay)

```
48  threads -> approx. 4 CPU in use (out of 8) IO Rate  9400
480 threads -> approx. 8 CPU in use (out of 8) IO Rate 50100
48  threads -> approx. 4 CPU in use (out of 8) IO Rate  8100
```

Ohne Latenz

```
48  threads -> approx. 8 CPU in use (out of 8) IO Rate 43700
480 threads -> approx. 8 CPU in use (out of 8) IO Rate 46100
48  threads -> approx. 8 CPU in use (out of 8) IO Rate 42800
```

Phänomene ...

idh
The open system company

CPU Folding ausschalten, d.h. Parallelisieren stärker als Affinität gewichten

Bei RHEL 5 gibt nichts

Bei RHEL 6 gibst 2 Paramter, welche aber bei uns (bis jetzt) keinen Effekt gezeigt haben.

kernel.sched_migration_cost
kernel.sched_nr_migrate

Tunning

Stripping

Wir haben mit verschiedenen stripe sizes (16k/64k/256K) und unterschiedlicher Anzahl (2..10) gemessen

Das Optimum hängt vor allem von den LUN Eigenschaften (Attachment, Cache, Latenz usw.) ab und ist daher für jede Umgebung unterschiedlich.

Blöcke im Verhältnis zur transfer size zu gross, kommt Stripping nicht in Effekt
Blöcke im Verhältnis zur transfer size zu klein, generiert mehr IO's als nötig

Analoges gilt für die Ermittlung der Anzahl stripes

Wir haben gute Ergebnisse mit 8 Stripes und 64K Blöcken erreicht

Wir haben viel gemessen und viele Zahlen ausgewertet
Die Essenz sollte aber erzählt sein

FRAGEN?

und zum Schluss...

idh
The open system company