Benchmarking SAN Storage

francisco roque
Benchmarking SAN Storage

• SAN/NAS hardware chain very complex!
• One change = big difference (e.g. RAID blocksize)
• Common uses = big differences (e.g. Oracle difference than Apache)
SAN/NAS Storage Components

- **Storage Array**
  - JBOD/Tray
  - Controller
  - Controller

- **Fabric/Network**
  - Switch

- **Clients**
  - Server

- Key:
  - RAM/cache
  - Processor
  - HBA/NIC/Interconnect
  - Cabling/bus

- 5 components with processors
- 6 cache locations
- 5 interconnects
- Numerous cables
Problem components

- Processors run code that may not be optimized
- Cache misses make this slower
- Interconnects impose bottlenecks
- Cables can be flaky
SAN Tunables

- Write cache – on/off, with/without battery
- Read ahead – on/off
- Blocksize – 4k, 8k, 16k ... 512k
- Raid level – 0, 1, 5, 6, 10, 50, 60, 7.n
- Spindles – e.g. quantity of disks
- Disk type – SATA/SAS/FC, enterprise/home
Switch Tunables

- Arp cache – size, age, poisoning
- Vlan – where are your packets going?
- Speed – 10/100/1000, full/half duplex
Server Tunables

- FS cache (read/write) – sizes, uses
- Direct IO – on/off
- Read ahead – on/off
- Blocksize – 4k, 8k, 16k ... 512k
- Checksum, compression, etc – on/off, type...
Effect of RAID Levels: RAID0

• Data striped across N+1 disks
• Read/write can be done to/from all disks
• Effective throughput is sum of all disks
• Minimal processing
• Disk failure requires complete rebuild/restore
Effects of RAID Levels: RAID1

• Data mirrored across N+1 disks
• Read can be from all disks
• Write effectively same as single disk
• Write throughput slow, read throughput same as RAID0 (controller dependant)
• Minimal processing
• Disk failure requires copy entire dataset to new disk
Effects of RAID levels: RAID5

- Data striped across N+1 disks, plus single data parity kept on all disks
- Read done from all disks
- Write done to all disks, but requires 1/N+1 extra data for parity
- Heavy processing, must calculate parity for each block
- Disk failure requires accessing each disk and parity calculation
Effects of RAID levels: RAID6

• Data striped across N+2 disks, plus double data parity kept on all disks
• Read done from all disks
• Write done to all disks, but requires 2/N+2 extra data for parity
• Heavy processing, must calculate double parity for each block
• Disk failure requires accessing each disk and parity calculation
Other Bottlenecks

- Hard Drive Zone bit Recording – outer tracks faster than inner (short stroking)
- SSD degradation over time – as disk gets full, writes get slower (under provision helps)
- CPU usage – processing done on card or on cpu?
Measurement Units

• IOps – seek time * access time
• Throughput (MBps)

• Transactions per second
• Files per second
• Web hits per second
• Etc, etc
Benchmark Tools: bonnie++

- Sequential/random read/write
- Sequential/random file create/delete
- Also measures cpu used during tests
- Can be run as multiple processes
Benchmark Tools: bonnie++

- Sample usage:
  
  ```
  bonnie++ -p4
  bonnie++ -d /testfs -s 20g -n 200 -r10000
  ```
Benchmark Tools: bonnie++

Storage Benchmarks on Sun T5220 - General I/O
(bonnie++ results, taller is better)

- 6140: FC, 5 disk
- 6140: SATA, 5 disk
- 2540: SATA, R5, 5 disk
- 2540: SATA, 11 disk
- 7310: NFS, 9 disk
- 7310: iSCSI, 9 disk
- 7310: iSCSI, WC, 9 disk
- SMS: SAS, 12 disk
- EQ: SATA, 16 disk

- Sequential Input (write)
  - Per Char
  - Per Block
  - Rewrite

- Sequential Output (read)
  - Per Char
  - Per Block

1gbps
Benchmark Tools: IOzone

- 13 different measurements
- Each measurement done for range of recordsize (aka blocksize) and filesize
- Takes very long time to run! (but thorough)
Benchmark Tools: IOzone

**Sample usage:**

```plaintext
iozone -f /testfs/file -a -b output.xls -g 16g -q 512
```

Run began: Sun Jan 24 22:01:01 2010

Auto Mode

Using maximum file size of 8388608 kilobytes.
Using Maximum Record Size 512 KB

Command line used: /export/home/frisco/bin/iozone -f /testsms/iozone -a -b 2540.csv -g 8g -q 512

Output is in Kbytes/sec

Time Resolution = 0.000001 seconds.

Processor cache size set to 1024 Kbytes.
Processor cache line size set to 32 bytes.

File stride size set to 17 * record size.

<table>
<thead>
<tr>
<th>KB</th>
<th>reflen</th>
<th>write</th>
<th>rewrite</th>
<th>read</th>
<th>reread</th>
<th>random</th>
<th>random</th>
<th>blkwd</th>
<th>record</th>
<th>rewrite</th>
<th>stride</th>
<th>fread</th>
<th>frewrite</th>
<th>fread</th>
<th>frewrite</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>4</td>
<td>51564</td>
<td>102074</td>
<td>209747</td>
<td>214781</td>
<td>315255</td>
<td>93570</td>
<td>167061</td>
<td>95264</td>
<td>173879</td>
<td>77008</td>
<td>87330</td>
<td>146381</td>
<td>157722</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>99060</td>
<td>189867</td>
<td>392444</td>
<td>371484</td>
<td>306901</td>
<td>173429</td>
<td>269837</td>
<td>176276</td>
<td>304761</td>
<td>128611</td>
<td>152797</td>
<td>232684</td>
<td>239716</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>16</td>
<td>173724</td>
<td>321554</td>
<td>557144</td>
<td>566352</td>
<td>462236</td>
<td>300238</td>
<td>389711</td>
<td>304761</td>
<td>458410</td>
<td>187611</td>
<td>241440</td>
<td>303333</td>
<td>329682</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>32</td>
<td>329882</td>
<td>508587</td>
<td>735021</td>
<td>821391</td>
<td>695778</td>
<td>481234</td>
<td>500050</td>
<td>484710</td>
<td>666877</td>
<td>245190</td>
<td>342477</td>
<td>385237</td>
<td>397796</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>532567</td>
<td>668072</td>
<td>929547</td>
<td>1039216</td>
<td>901577</td>
<td>627470</td>
<td>956597</td>
<td>640574</td>
<td>889431</td>
<td>290890</td>
<td>409346</td>
<td>432938</td>
<td>450841</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>4</td>
<td>39826</td>
<td>103149</td>
<td>215837</td>
<td>238435</td>
<td>173007</td>
<td>94472</td>
<td>179028</td>
<td>96527</td>
<td>171809</td>
<td>81174</td>
<td>88333</td>
<td>156668</td>
<td>159424</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>8</td>
<td>802636</td>
<td>191002</td>
<td>366763</td>
<td>374176</td>
<td>313811</td>
<td>176788</td>
<td>296318</td>
<td>176780</td>
<td>310724</td>
<td>133622</td>
<td>153094</td>
<td>242400</td>
<td>246322</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>16</td>
<td>165427</td>
<td>332467</td>
<td>574312</td>
<td>552689</td>
<td>500012</td>
<td>307696</td>
<td>452366</td>
<td>310709</td>
<td>505620</td>
<td>199076</td>
<td>244721</td>
<td>335166</td>
<td>343310</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>32</td>
<td>277050</td>
<td>533668</td>
<td>800337</td>
<td>836500</td>
<td>731625</td>
<td>501839</td>
<td>600657</td>
<td>503723</td>
<td>726759</td>
<td>261785</td>
<td>346859</td>
<td>413039</td>
<td>425414</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>64</td>
<td>467329</td>
<td>723735</td>
<td>1000121</td>
<td>1057236</td>
<td>940549</td>
<td>692017</td>
<td>747933</td>
<td>687585</td>
<td>940549</td>
<td>317711</td>
<td>422464</td>
<td>473930</td>
<td>481150</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>128</td>
<td>814915</td>
<td>914904</td>
<td>1165070</td>
<td>1217930</td>
<td>111981</td>
<td>876806</td>
<td>836500</td>
<td>883233</td>
<td>1102844</td>
<td>354415</td>
<td>486822</td>
<td>508012</td>
<td>520321</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>55078</td>
<td>94922</td>
<td>214413</td>
<td>2156395</td>
<td>177141</td>
<td>860984</td>
<td>832385</td>
<td>860579</td>
<td>172593</td>
<td>79821</td>
<td>81474</td>
<td>154211</td>
<td>158210</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>106572</td>
<td>174144</td>
<td>371561</td>
<td>378103</td>
<td>313552</td>
<td>165976</td>
<td>312515</td>
<td>165974</td>
<td>252307</td>
<td>131944</td>
<td>137597</td>
<td>211042</td>
<td>248792</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>16</td>
<td>330812</td>
<td>515838</td>
<td>692446</td>
<td>600588</td>
<td>518366</td>
<td>305341</td>
<td>490336</td>
<td>295024</td>
<td>514045</td>
<td>203821</td>
<td>262776</td>
<td>346591</td>
<td>349572</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>32</td>
<td>332036</td>
<td>512527</td>
<td>759926</td>
<td>842278</td>
<td>749133</td>
<td>483927</td>
<td>664935</td>
<td>496529</td>
<td>750970</td>
<td>271709</td>
<td>321208</td>
<td>427299</td>
<td>439528</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>64</td>
<td>509895</td>
<td>702999</td>
<td>996229</td>
<td>1062263</td>
<td>972763</td>
<td>679237</td>
<td>865358</td>
<td>699594</td>
<td>976301</td>
<td>332447</td>
<td>397563</td>
<td>492350</td>
<td>499925</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>815414</td>
<td>861885</td>
<td>1052388</td>
<td>1230218</td>
<td>1142514</td>
<td>842278</td>
<td>876662</td>
<td>891956</td>
<td>1128110</td>
<td>365242</td>
<td>443730</td>
<td>475779</td>
<td>537729</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>884902</td>
<td>947860</td>
<td>1274008</td>
<td>1319408</td>
<td>1248819</td>
<td>924200</td>
<td>1048775</td>
<td>927333</td>
<td>1248819</td>
<td>459367</td>
<td>498069</td>
<td>551823</td>
<td>559005</td>
<td></td>
</tr>
</tbody>
</table>
Benchmark Tools: IOzone
Benchmark Tools: Other

- FileBench
- IOMeter
- SQLIO
- TPC – various db
- SPEC – CPU, java, mail, others
Relating it back:
what is your current usage?

- Measure IOps/throughput of current volumes
- Avg, max, 95% (to remove cache hit)
- % read vs % write