

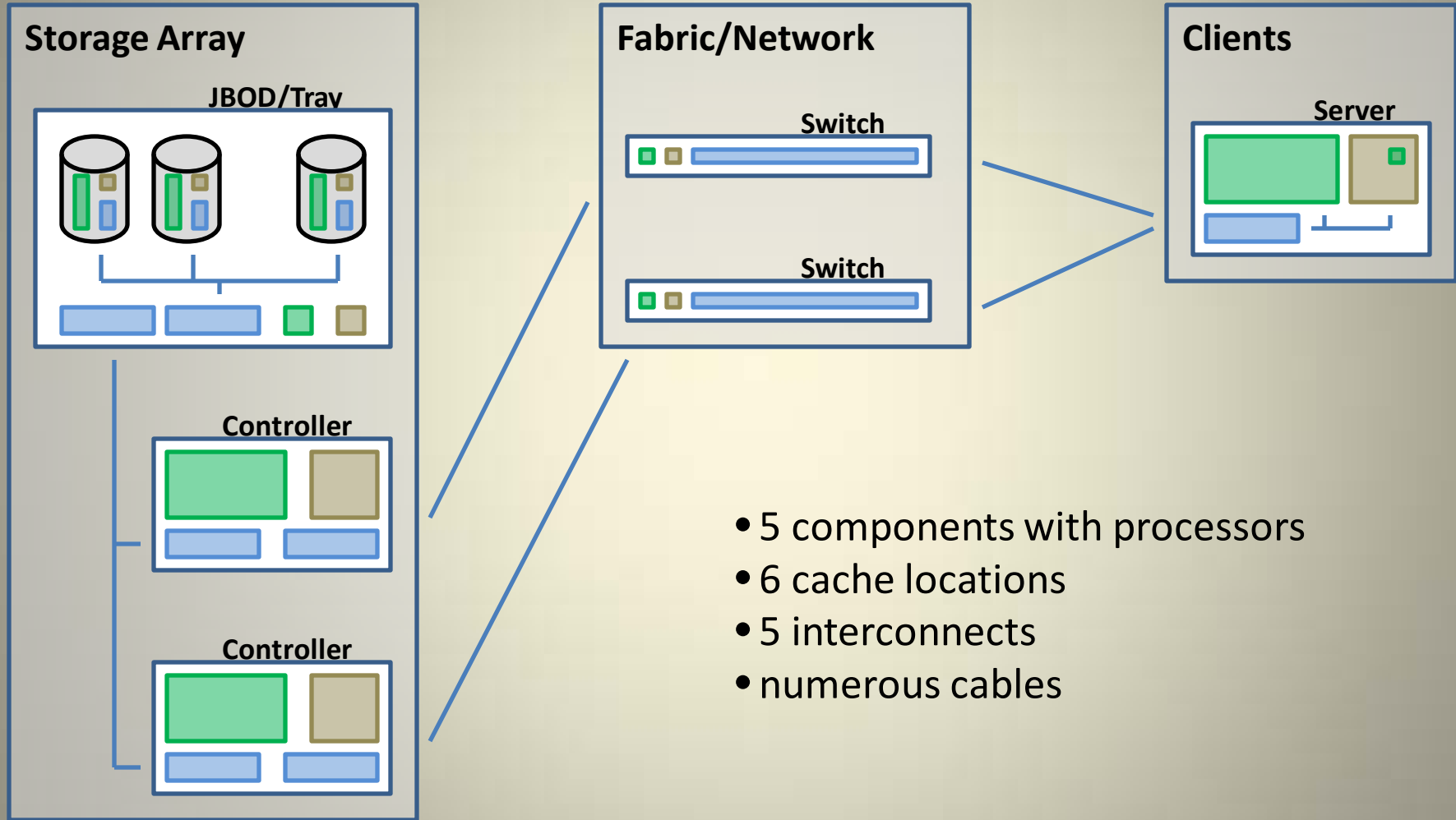
Benchmarking SAN Storage

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



- 5 components with processors
- 6 cache locations
- 5 interconnects
- numerous cables

Key:



RAM/cache



Processor



HBA/NIC/Interconnect



Cabling/bus

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
```

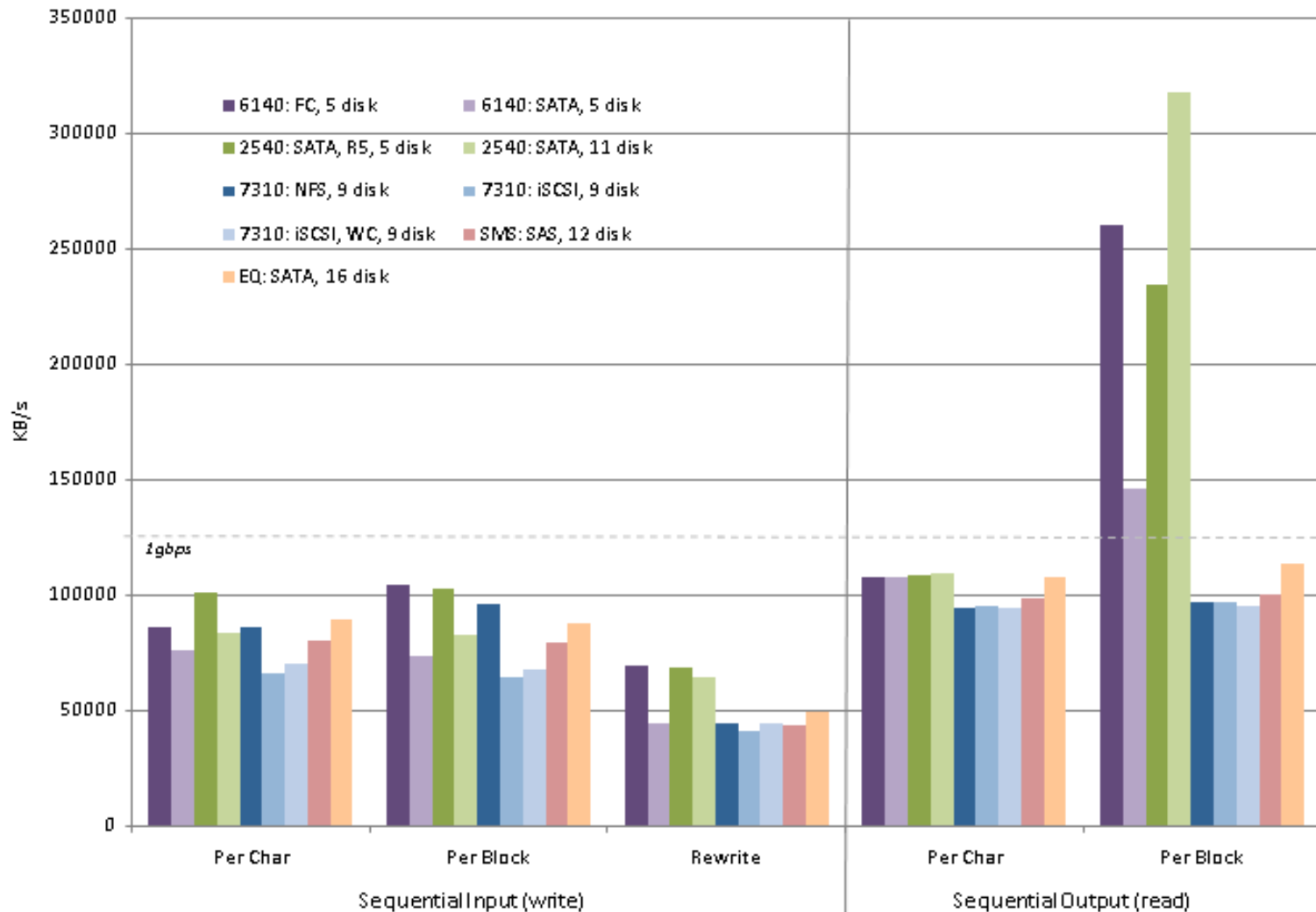
```
bonnie++ -d /testfs -s 20g -n 200 -r10000 (x4)
```

```
Using uid:0, gid:0.
Writing with putc()...done
Writing intelligently...done
Rewriting...done
Reading with getc()...done
Reading intelligently...done
start 'em...done...done...done...
Create files in sequential order...done.
Stat files in sequential order...done.
Delete files in sequential order...done.
Create files in random order...done.
Stat files in random order...done.
Delete files in random order...done.
Version 1.03d
-----Sequential Output----- --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine      Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP /sec %CP
server       20G 21084 78 20856 14 16146 17 27286 99 77723 31 468.1 3
-----Sequential Create----- -----Random Create-----
-Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
files /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP
200 1569 70 7084 98 2855 70 1520 69 8706 85 2516 76
server,20G,21084,78,20856,14,16146,17,27286,99,77723,31,468.1,3,200,1569,70,7084
,98,2855,70,1520,69,8706,85,2516,76
```

Benchmark Tools: bonnie++

Storage Benchmarks on Sun T5220 - General I/O

(bonnie++ results, taller is better)



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:

```
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 /testfs/iozone -a -b 2540.csv -g 8g -q 512
```

```
Output is in Kbytes/sec
```

```
Time Resolution = 0.000001 seconds.
```

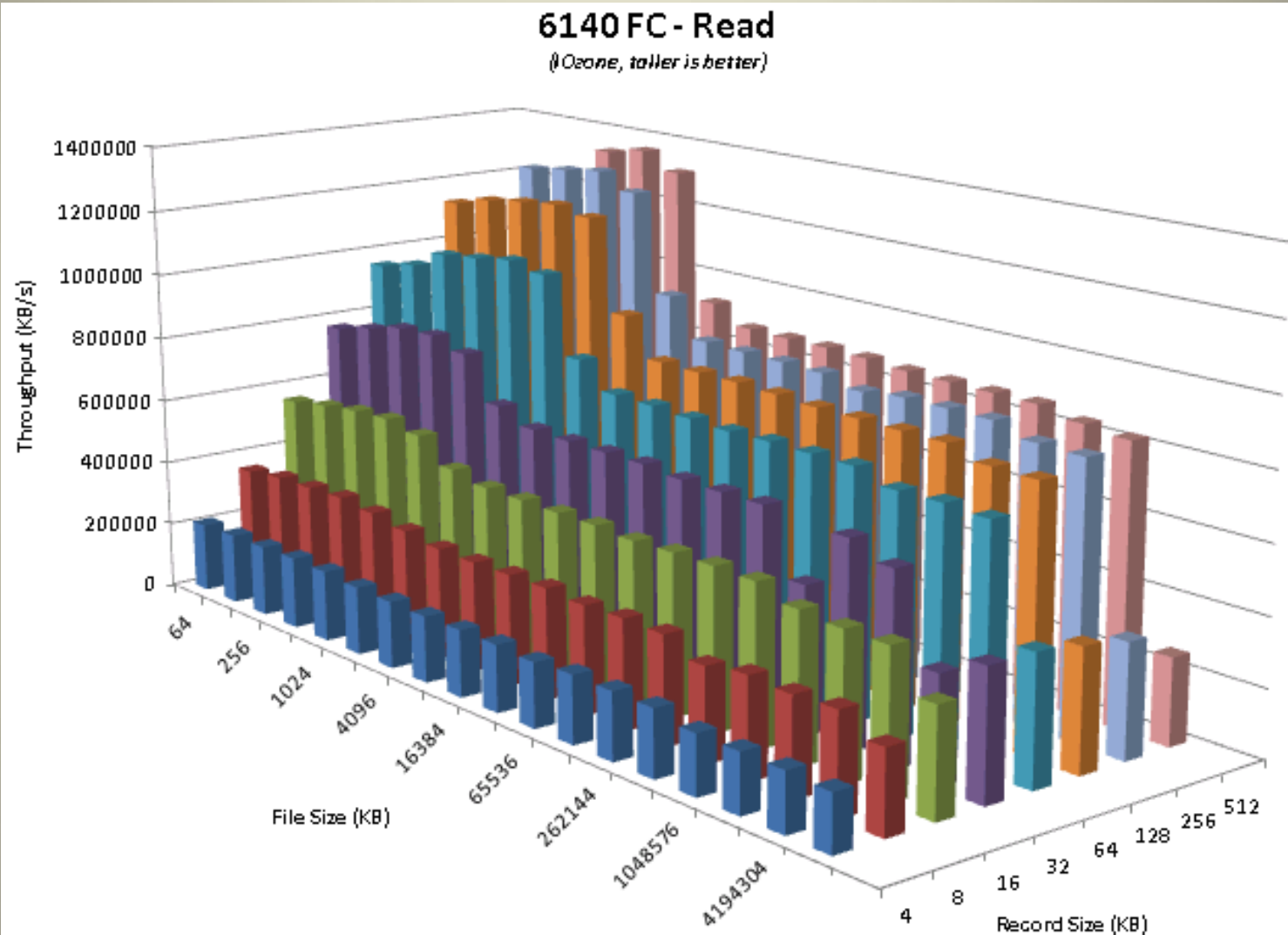
```
Processor cache size set to 1024 Kbytes.
```

```
Processor cache line size set to 32 bytes.
```

```
File stride size set to 17 * record size.
```

	KB	reclen	write	rewrite	read	reread	random	random	bkwd	record	stride	fwrite	frewrite	fread	freread
							read	write	read	rewrite	read				
64	4	51564	102074	209747	214781	175355	93570	167061	95264	173879	77008	87930	146381	155722	
64	8	99060	189867	359444	371894	306501	173429	269837	176276	304761	128511	152797	232854	239716	
64	16	177324	321554	557144	586352	488236	300328	389711	304761	484710	187611	241440	303383	329852	
64	32	329852	508587	763021	821391	695778	481234	500060	484710	686877	245190	342477	385237	397796	
64	64	552557	668072	955947	1033216	901377	627470	592827	645579	889431	290890	409946	432398	450541	
128	4	39836	103149	215887	218435	179807	94472	179028	96597	171809	81174	88333	156668	158424	
128	8	82636	191002	366763	374176	313811	176788	296318	178730	310724	133622	153094	242400	246632	
128	16	154237	332467	574312	592699	508012	307696	452366	313079	505620	199076	244721	335166	343310	
128	32	277050	535368	800337	836500	731625	501839	600657	503723	727658	261785	346859	413039	425141	
128	64	467329	723735	1000121	1057236	940549	692017	747933	687585	940549	317711	422464	473930	481150	
128	128	814915	914904	1165070	1217930	1111981	876086	836500	883293	1102844	354415	486822	508012	520321	
256	4	58078	94922	214413	215835	179141	86894	183238	88027	172966	79821	81474	154211	158210	
256	8	106672	174144	371561	378103	318352	165476	312515	163784	292907	131944	137697	211042	248792	
256	16	190217	315638	573946	600928	519266	295324	490336	295243	514045	203521	226776	340891	348752	
256	32	332036	515279	799625	842278	748353	483927	664935	486559	750970	271709	321209	427989	437582	
256	64	509895	709299	996229	1062263	972763	679237	865358	699594	976301	332447	397563	492360	499925	
256	128	815414	861885	1052888	1230218	1142514	842278	876662	891956	1128110	365242	443730	475779	537729	
256	256	889002	947860	1274008	1319408	1248819	924200	1048775	927393	1248819	459687	498069	551823	559005	

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