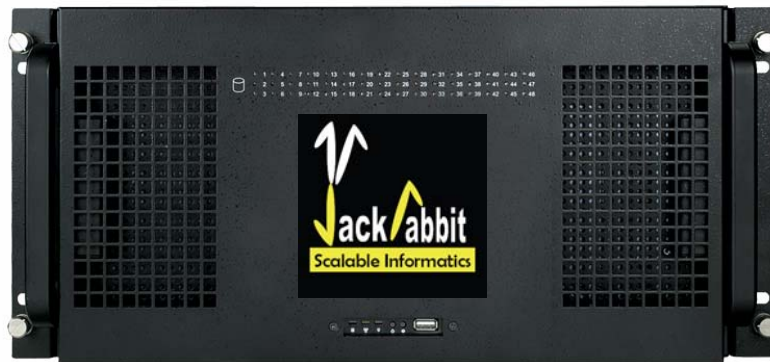


Scalable Informatics JackRabbit Benchmark Results

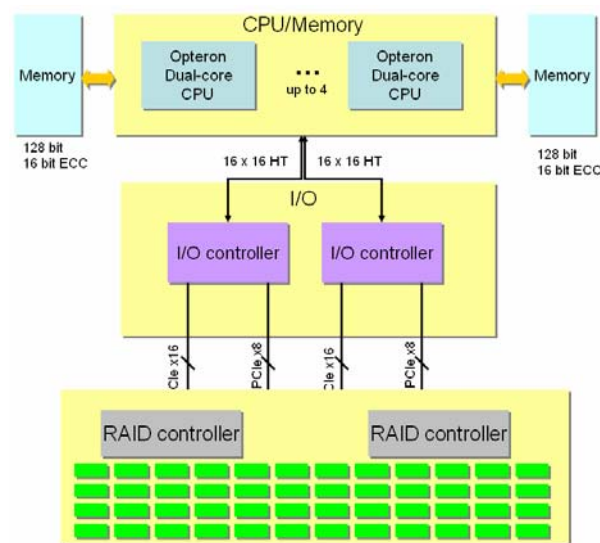


Introduction:

Scalable Informatics JackRabbit systems are multiprocessor storage servers providing up to 36 TB of fast, but low cost storage in support of solving today's most complex computationally and data intensive problems. Scalable Informatics JackRabbit supports a full range of storage subsystems including RAID 0, 1, 5, 6, 10, 50 and 60 with hot-swappable SATA 300/NCQ disks

Architected to scale: I/O bandwidth

Scalable Informatics JackRabbit is designed to scale, with significant available I/O bandwidth. Each JackRabbit unit includes multi PCIe lanes providing in excess of 44 lanes (22 GB/s) of internal IO bandwidth. These PCIe lanes are available for connecting RAID and network interfaces. The 5U Scalable Informatics JackRabbit product has two x16 and two x8 PCIe interfaces. An additional 2 PCIx busses are available, providing 2 x 850 MB/s of bandwidth



Multiple RAID controllers may be used for redundancy or performance. A minimum of two RAID controllers are required for the 5U system. Each RAID controller is connected to the system via a multiple lane PCIe bus. Network I/O may be attached to a PCIx or PCIe bus if required by network bandwidth considerations, such as DDR Infiniband or 10 Gb Ethernet.

Each unit has a quad Gigabit Ethernet adapter, which may be channel bonded. This provides a default of 400MB/s in network bandwidth. An additional 2 quad Gigabit Ethernet adapters options are available, for a maximum of 1.2 GB/s bandwidth to a Gigabit-class network per chassis. Additional options, namely Infiniband or 10Gb Ethernet may also be selected. If these options are chosen, the adapters would occupy a PCIe slot, unless a PCIx slot version was available.

With a dual port, DDR Infiniband connection, Scalable Informatics JackRabbit would have approximately 4 GB/s of network bandwidth. With a single port 10 Gb Ethernet card, Scalable Informatics JackRabbit would have approximately 1 GB/s of network bandwidth.

Benchmark

System configuration

A JackRabbit unit with the following configuration was used for the tests indicated below.

- 2x 2.6 GHz Opteron™ dual core processors
- 16 GB DDR2-533 Registered ECC memory
- 46x 750GB SATA disks (data)
- 2x 250GB SATA disks (OS)
- 1x Quad port gigabit ethernet
- 2x RAID controllers with 1 GB battery backed cache each

The JackRabbit unit was run with Ubuntu 6.10. An updated 2.6.20 kernel was compiled from www.kernel.org in order to implement file system fixes. Updated RAID drivers were built against this kernel. The xfs file system was used. Various kernel parameters were adjusted to provide more consistent server performance. Similar results to the reported were obtained using OpenSuSE 10.2.

The disks were organized as a RAID60, with a RAID0 stripe across two RAID6 volumes composed of 18 drives each, with 3 hot spares.

Detailed configuration descriptions may be provided upon request, by contacting Scalable Informatics or our partners at sales@scalableinformatics.com.

Critique

Often times, published results rarely report outside of disk controller cache, or system buffer cache. More often than not, conclusions are drawn about disk performance from looking at data sets which are clearly sitting in a cache or memory bound region, and have little to do with disk system performance. For units with 1GB RAID cache, results for tests below 1 GB in total size will yield a test of the cache performance. System and buffer cache figure into this as well. Buffer cache can often fill 70-80% of physical ram, hence it is important to measure this region and beyond. It is instructive to note that many benchmark tests which test performance in cached regions often rarely spill buffer and other cache to disk, or do so asynchronously. You may see this effect by watching the disk activity LEDs for in-cache versus out-of-cache measurements.

Unfortunately, most of the published reviews will tend to ignore this, and cut off their reporting or plots before they take a significant performance hit for operating out of cache region. This report will show what happens when the system operates in cached as well as uncached regions. The cached region is important for small file IO and file

operations. The uncached region is important for huge file IO. Cache only helps small file IO, it could get in the way of huge file IO.

Tests

IOzone (www.iozone.org) was used in addition to bonnie++ (www.coker.com.au/bonnie++/), and dbench (samba.org/ftp/tridge/dbench/). Neither IOzone nor bonnie++ reflects “real-world” use cases, and thus have limited utility in systems comparisons, though they have become standard system metrics. The dbench test deals with specific code testing the systems abilities as a windows file server, and thus is less specifically stressful upon the disk system as compared to the rest of the computing system.

Dbench

The dbench benchmark used v3.04, running for 60 seconds on this system. The specified number of clients was set on the command line. Specifically, dbench was run as

```
dbench -t 60 N
```

where N is the number of clients.

The JackRabbit dbench results appear as follows:

Number of clients	Throughput
20	958 MB/s
50	865 MB/s
100	682 MB/s

The dbench benchmark exercises systems in a nearly identical manner to how SAMBA would under the indicated load. This is relatively a more “realistic” case than the subsequent tests, though there is less physical disk IO during these tests. This test does test the system as a whole under load, and gives a somewhat better perspective on how it will perform as a SAMBA/CIFS file server than the subsequent tests.

Most of this IO is in cached regions, so this test also demonstrates the impact of resource contention (multiple processes vying for a limited number of resources) upon throughput.

Bonnie++

The bonnie++ benchmark is basically streaming reads and writes. These tests are usually recommended to be run on at least 2x the system RAM size in order to eliminate caching affects. This benchmark will show how fast the system can read and write in a non-cached scenario.

The bonnie++ 1.03 code was run with 1 thread to generate a performance baseline measurement.

```
jackrabbit1,32104M,,,585906,88,288982,51,,,906829,80,384.4,0,,,,,,,,,,,,,
```

This indicates that for this test, 32GB files were used, the system sustained 586 MB/s of streaming writes (uncached), 289 MB/s rewriting (uncached read, modify, uncached write), and 907 MB/s of streaming reads (uncached). With 32 GB of test file, a system memory of 16 GB, and a total RAID cache of 2 GB, these files will not fit into cache, and hence caching was rendered ineffective. This is why the word “uncached” follows the above.

As this is a single thread performing these operations, it would be instructive to see what multiple threads would be able to do. A bonnie++ run using 4 threads was performed. Assuming that the bandwidth is well partitioned between threads, one might expect 586/4 MB/s of 146.5 MB/s writing per thread. Similarly, for rewriting 289/4 MB/s or 72.3 MB/s, and for reading 907/4 MB/s or 227 MB/s.

The particular bonnie++ run was performed as follows:

```
bonnie++ -r 16384 -s 32g:32k -u root -y -f -n0 \  
-d /storage/1 & bonnie++ -r 16384 \  
-s 32g:32k -u root -y -f -n0 \  
-d /storage/1 & bonnie++ -r 16384 \  
-s 32g:32k -u root -y -f -n0 \  
-d /storage/1 & bonnie++ -r 16384 \  
-s 32g:32k -u root -y -f -n0 -d /storage/1
```

The resulting measurement is as follows:

```
jackrabbit1,32G:32k,,,131022,21,76012,14,,,180677,17,116.4,0,,,,,,,,,,,,,  
jackrabbit1,32G:32k,,,130855,21,76127,14,,,180814,17,113.4,0,,,,,,,,,,,,,  
jackrabbit1,32G:32k,,,144861,23,73766,13,,,180639,17,112.9,0,,,,,,,,,,,,,  
jackrabbit1,32G:32k,,,130949,21,73615,13,,,181276,18,110.5,0,,,,,,,,,,,,,
```

Or 537 MB/s aggregate streaming writes, 300 MB/s aggregate rewriting, and 724 MB/s aggregate streaming reads, all on a set of files 32 GB in size, using a 32k chunk size. As this test exceeds the largest size cache system (main memory) by a factor of 2, caching effects would not contribute to overall performance. These results are in line with expectations. While few use cases have similar profiles, one can discern good overall performance is maintained even outside cached regions.

IOzone

Several IOzone versions were run on this system. IOzone was patched to allow for greater than 16 GB file size, and greater than 16 MB records. This patch will be available on jackrabbit.scalableinformatics.com website. This patch allows testing in regions not originally contemplated by the IOzone authors, and also allows testers to test far outside of the cached regions.

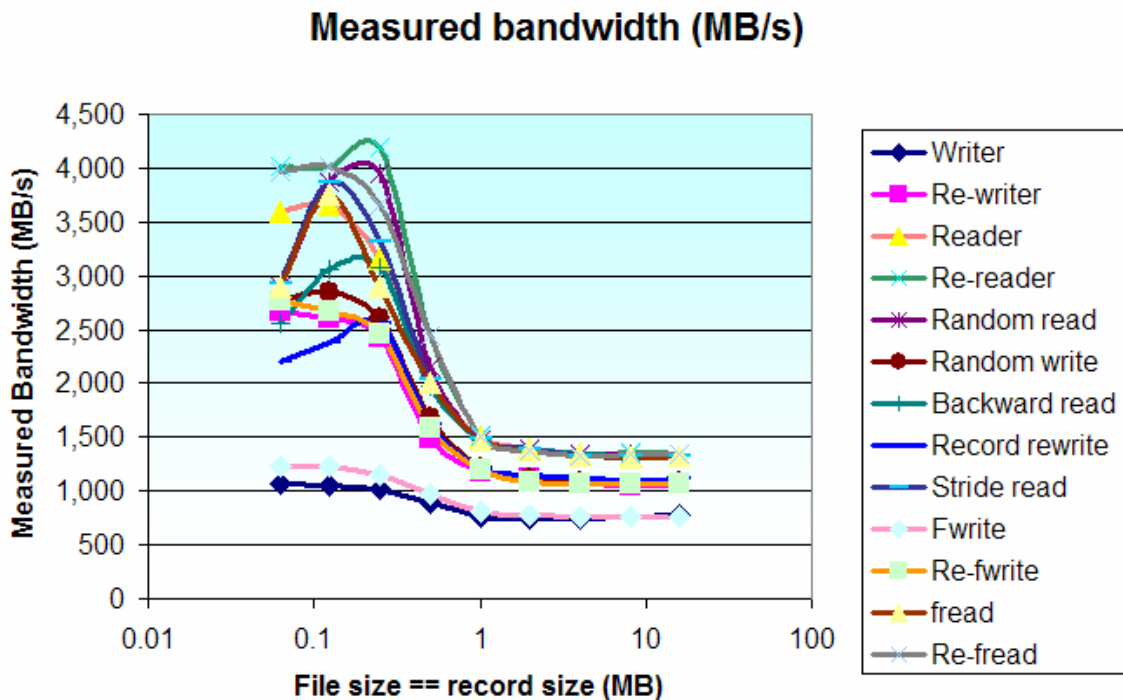
Tests were run from 16 GB files through 64 GB file size, with records up to 64MB. The data below was gathered via the following:

```
/home/oem/iozone -Ra -g 16g \  
                -b /root/JR-ubuntu-external-log.xls
```

```
/opt/iozone/bin/iozone -Ra -g 32g -y 1m -n 1m \  
                    -b /root/updated-iozone-bm-100.xls
```

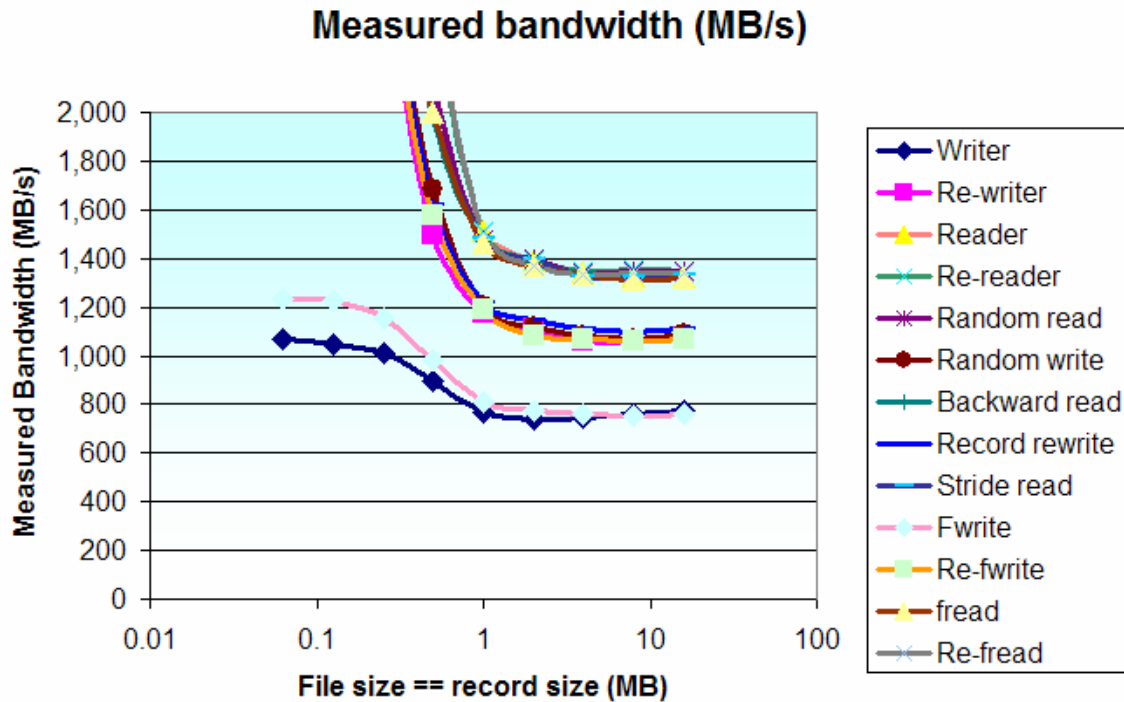
```
/opt/iozone/bin/iozone -Ra -g 64g -q 64m -n 1024 \  
                    -y 1024 -m -b /root/updated-iozone-bm-2-large.xls
```

In the first measurement, data for file size equivalent to record size is reported. This report has been requested in the past. For the smaller run (up to 16 GB) the data appears as follows.



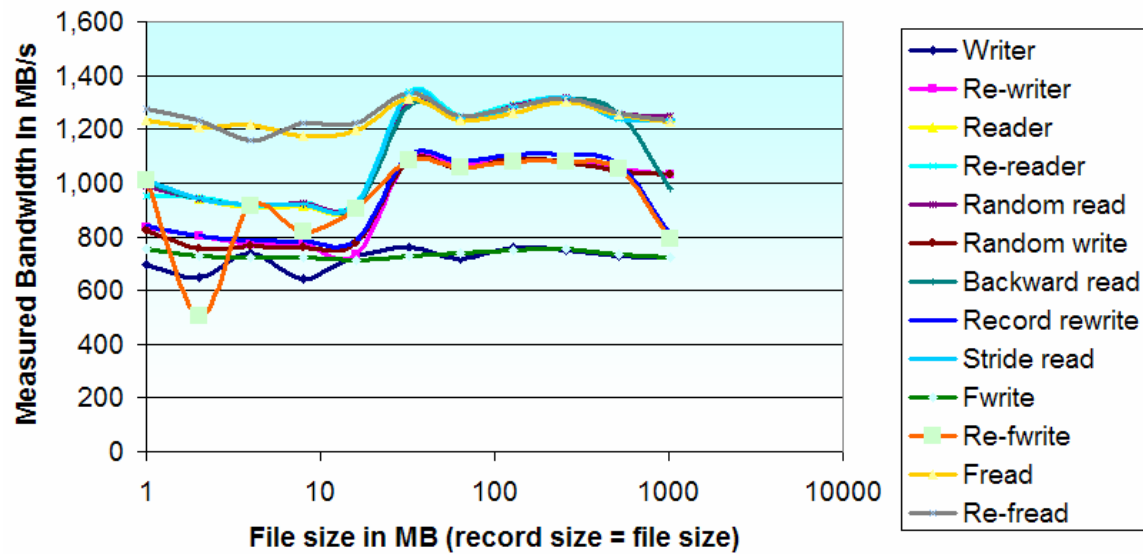
As processor cache is 1MB, and 4 processors will have a combined 4 MB of cache, it is important to note that this would be cached at the lowest levels of the memory hierarchy, and hence data below 1MB, and more reasonably, 4MB, would not be meaningful as a measure of disk performance, but could be meaningful in terms of very small IO performance.

Expanding the vertical axis, and noting that the un-patched version of IOzone stopped at a 16 MB record size, the same data appears as follows:



Again, as the RAID cache is 1 GB per controller, with 2 RAID controllers, this data represents entirely RAID cache bound IO. In order to see the impact of larger IO operations, such as dumping a huge bolus of data, or reading the same, at once, the patched IOzone is used. Using a 1 MB starting record size, and working up to a 1 GB file size, a similar analysis and plot is performed.

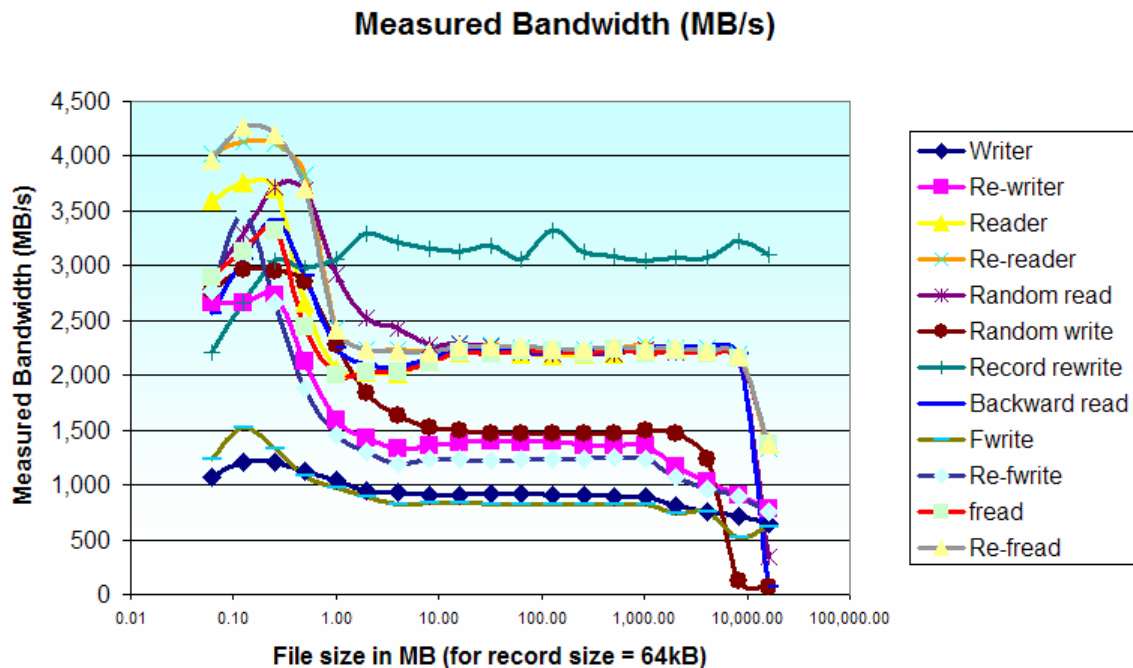
Measured Bandwidth



As can be seen in the plot, as the size of the IO increases, the random write and re-writing performance increases from slightly below 800 MB/s to above 1100 MB/s. The reading, re-reading, and related operations increase to about 1300 MB/s. Again, these measurements are all bound by the size of the RAID cache, which is larger than the IO measurements themselves. As this is cache bound, these measurements are more meaningful for small files and IO operations which may themselves be cache bound.

64 kB record size

Running up to 16 GB in file size, well outside of RAID cache, but the same size as physical RAM, IOzone provided the following results.



Notice several specific items:

First, the re-fwrite and the writer are the only tests which dip below 1 GB/s over most of the range of the test.

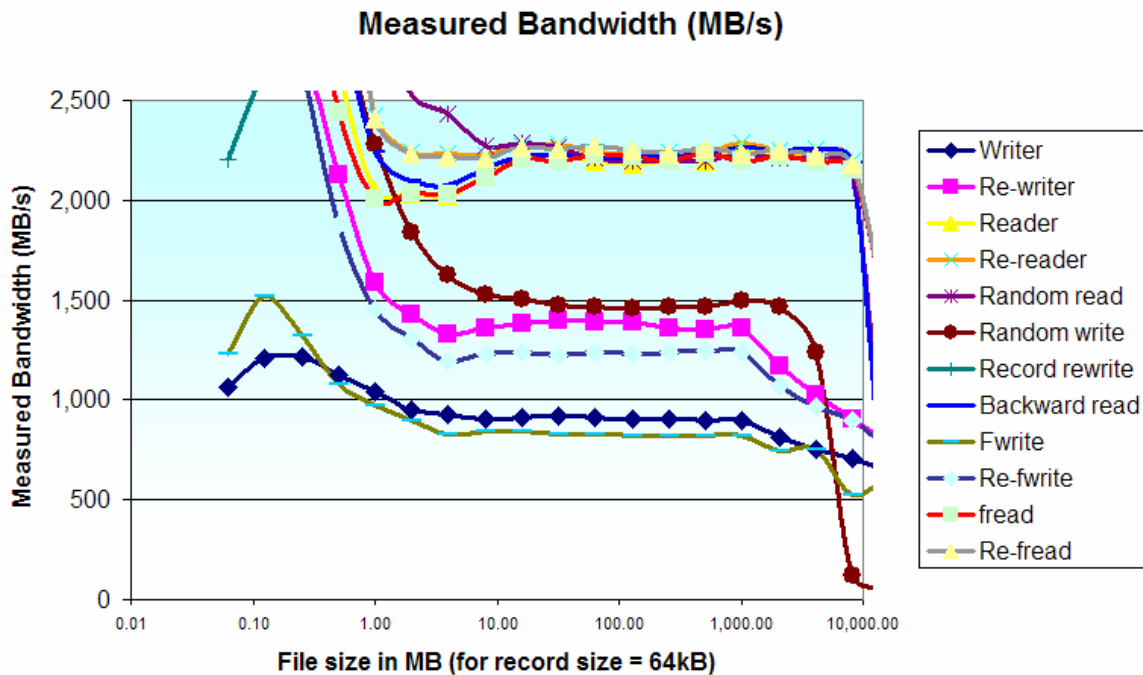
Second, there is a significant fall off in performance after 10 GB (10,000.00 MB) in file size, as buffer cache is rendered ineffective.

Third, there is a fall off in performance after about 1 MB files, as they are now outside processor cache size.

Fourth, there is a slight fall off in performance after 1 GB file size, when RAID cache is rendered ineffective.

Its this last element that is the most important for large files. When transitioning from cached to uncached regions, the native system throughput (as indicated in the bonnie++ tests previously) would be represented in the file sizes above 16 GB.

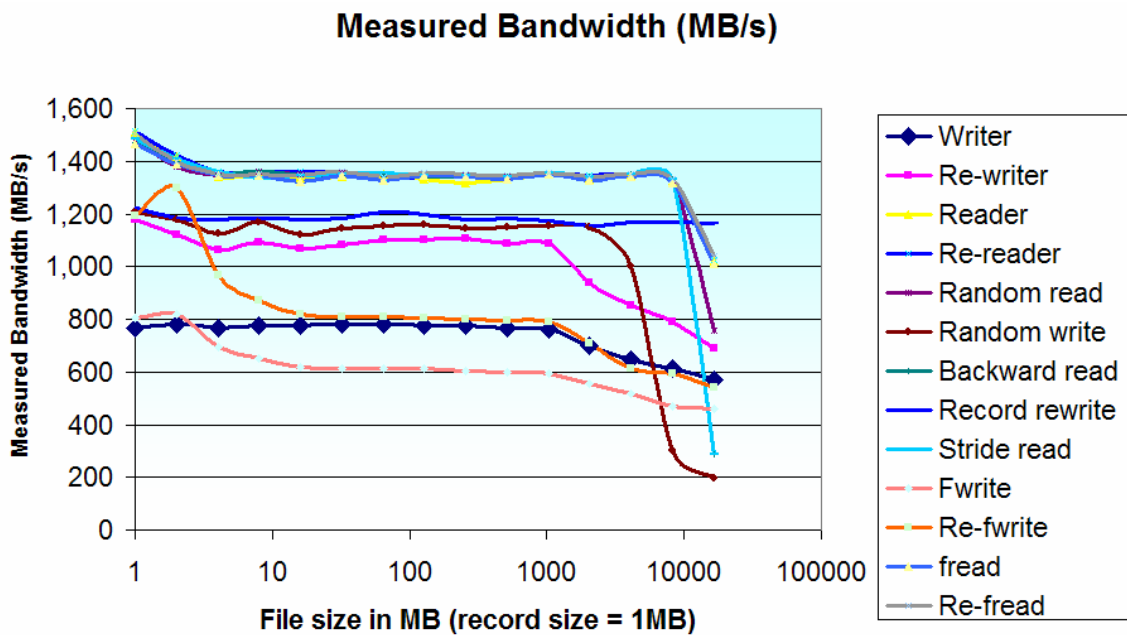
Expanding this plot to provide more detail in the cached region, we see the data as follows.



Again, these are 64 kB records. Larger records were also plotted.

1 MB record size

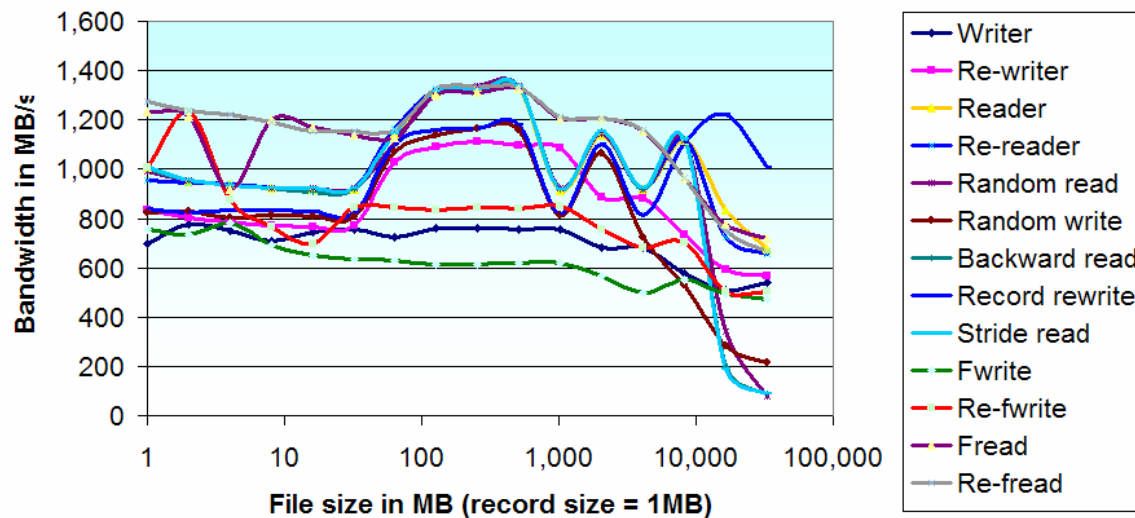
Using a 1 MB record size rather than a 64 kB record size effectively washes out the processor cache effects. As can be seen in the plot below, the larger record size means fewer IOs, but more data per IO.



Similar to the preceding points, the RAID cache effect is seen through 1 GB file size (1000 MB). The system buffer cache effect is seen through about 10 GB, and beyond that the system is in transition to the large file size (uncached) performance. More specifically, the observations show most operations performing in excess of 1 GB/s, with specific writing operations performing in the 800-600 MB/s range over most of the region.

Since the previous data stopped at file sizes of 16 GB, and it appeared that the performance drops dramatically after file sizes of 10 GB, measurements were performed up to 32 GB (64 GB data may be reported in the future) to understand if the “dramatic drop” in performance was real, or an artifact of the plotting on a logarithmic horizontal axis.

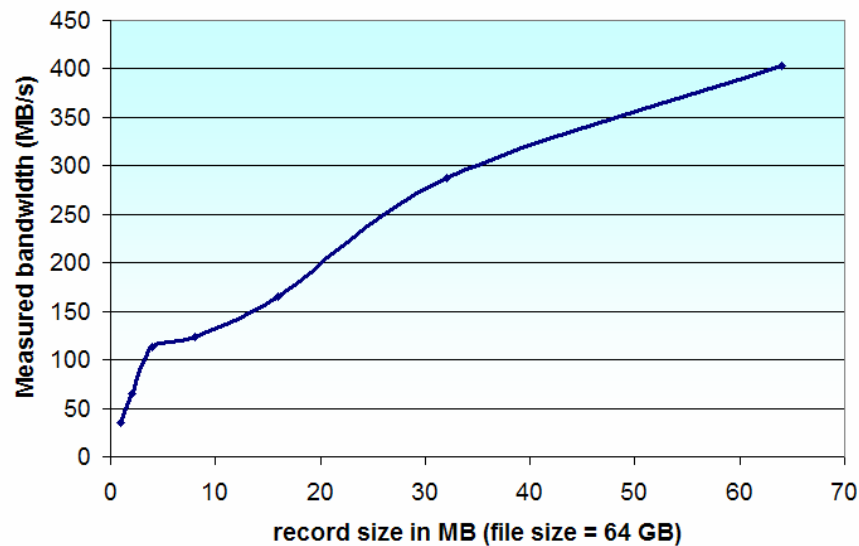
Measured Bandwidth



As can be seen, several of the tests did indeed experience a significant performance dropoff. In large part, these tests are dominated by seeks (stride read and random read). This is the domain of OLTP systems, where huge battery backed caches, solid state memory, and 15k RPM drives are used. As these are 7.2k RPM drives, each seek costs at least 8.5 ms, or a maximum of 117 IOPs per drive. Even with 20 drives in each RAID6 array, each seek results in three reads (two parity and one data) which is highly inefficient for small reads. Each write is a read-modify-write operation in the system, with at least 3 reads and 3 writes. Hence small record seeks for huge files are not well handled at this time.

Indeed, looking at the huge file performance of random read, we see the following

Random read performance vs record size



This suggests that there is a seek dominated region below about 5 MB in record size, and a “streaming” transition region above that. Each disk has a 16 MB cache on board, and can read at 70 MB/s in the outer rim, and about 40 MB/s on the inner rim. The time to complete an IO operation for this would be (highly simplified)

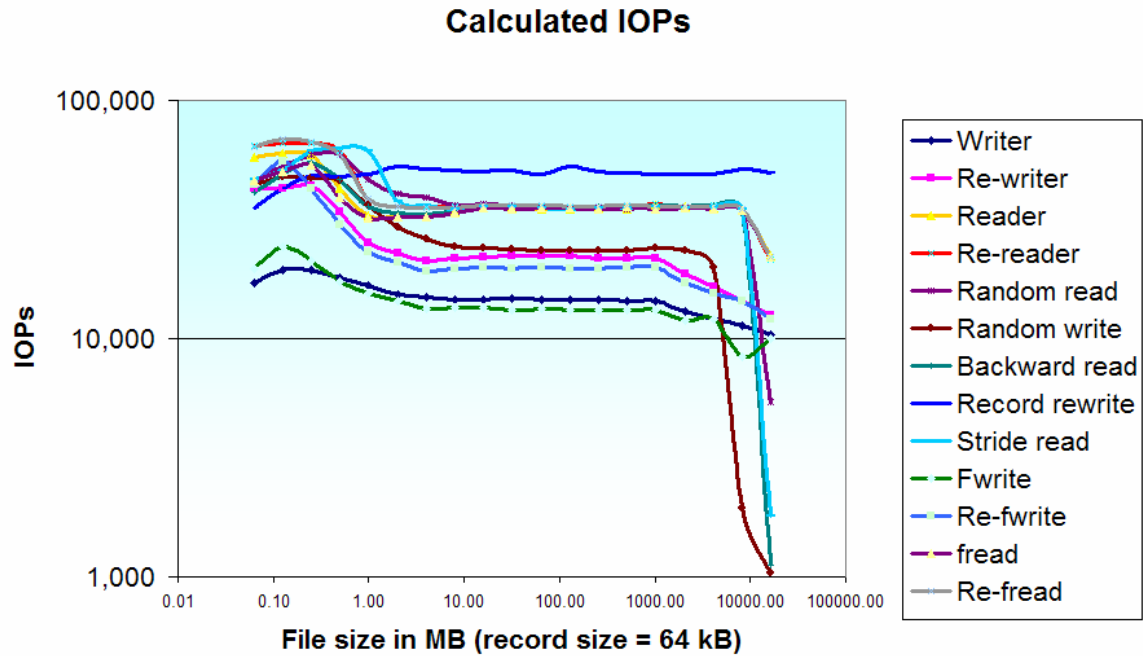
$$T_{io} = L_{seek} + S/B_{disk}$$

Where T_{io} is the time to complete an IO, the L_{seek} is the seek latency plus any additional rotationally induced latency, the S is the size of the transfer, and B_{disk} is the bandwidth of the disk operation. Moreover, the RAID6 chunk size is on the order of 8-128kB, so a 1 MB IO operation, would require between 128 down to 8 seek operations to retrieve data. Each seek operation would require reading 2 parity, and one data block. Unless the S/B_{disk} was of sufficient size to dominate the time, the total transfer time would be bound by the individual disk latencies, and their 24 to 384 seeks. The sharp transition at about 4 MB in record size occurs as the stripe across RAID6 units allows a parallelization of the seeks across multiple controllers. The performance increase after that may be in part due to fewer seeks in order to read more data, and IO elevator scheduling of operations for each disk.

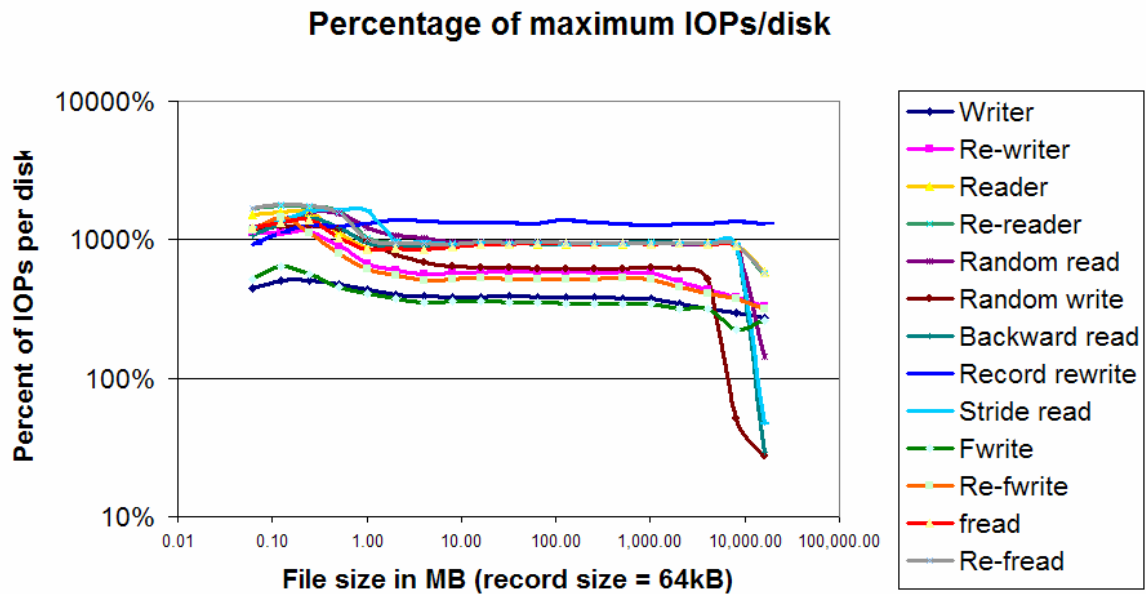
Hence in regions which are not seek dominated, JackRabbit performs exceptionally well, even in large file size and non-cached regions.

IOPs

IOPS were not directly measured, but calculated from bandwidth and IO size. Again, cache play critical roles in IOPs performance.

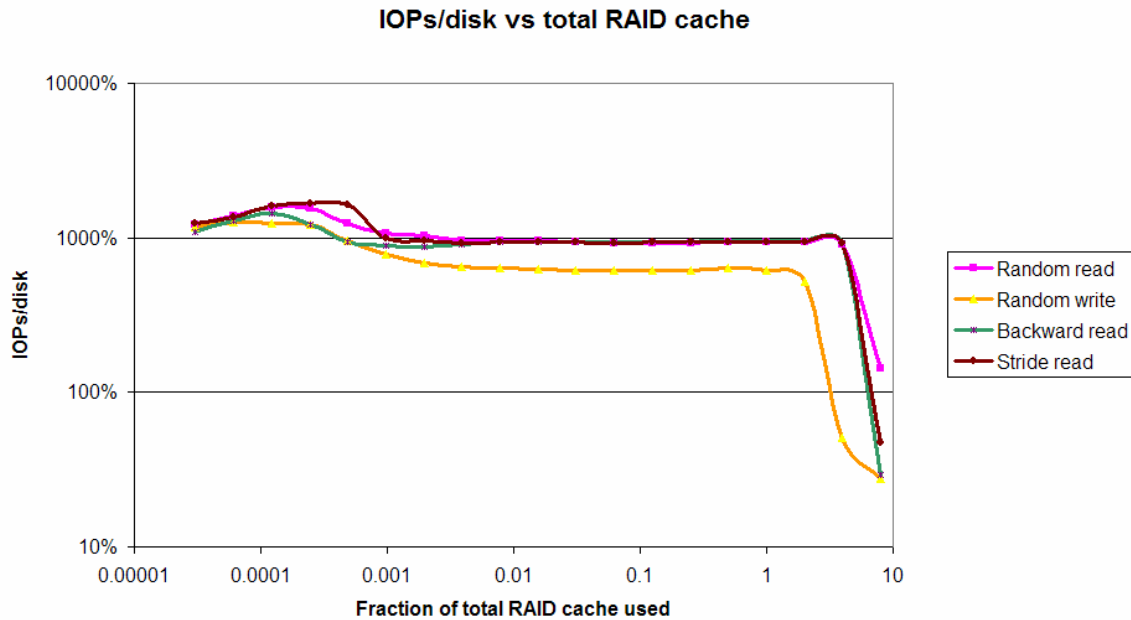


Since most of this is within the cached region, it is instructive to look at the percentage of IOPs calculated above on a per disk basis.



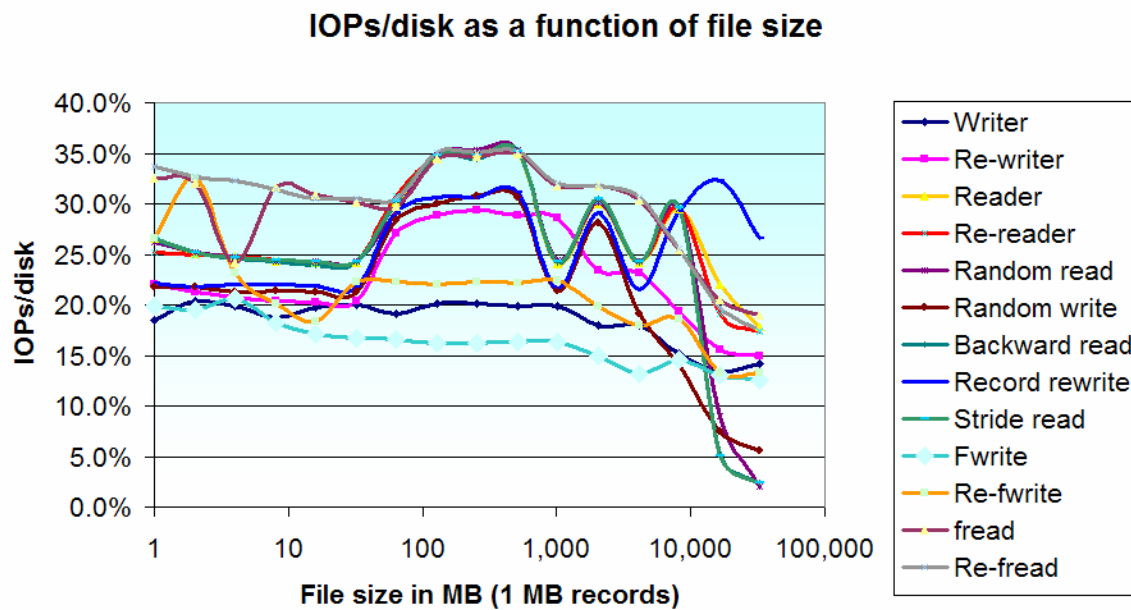
As can be seen, the impact of cache for IOP bound problems can be quite dramatic. For values over 100%, most of these operations must be IOPs to cache. For values under 100%, these are actual disk operations.

More to this point, by plotting this not in terms of file size, but in terms of RAID cache size, the cache effects become obvious.



As these operations benefit most from the cache, the impact upon performance as RAID cache size (total of 2 GB) is exceeded is striking. These benefit most from cache in part due to being able to hide most of the latency of the seek operation.

As noted before, performance “appears” to fall off above 10 GB. Extending the test through 32GB files, and using 1MB records shows the following.



For larger IO operations, more fewer seeks are required, which allows the system to operate in an overall higher performance region.

Summary:

Scalable Informatics JackRabbit systems provide excellent benchmark results over wide ranges of file and record sizes, including file sizes significantly larger than any possible cache on the system.

Sustained performance in excess of 1GB/s is commonly observed for operations tested. For non-cached operations, performance from 500 MB/s to 1 GB/s is observed for most operations. Moreover, this performance was obtained with a simple stripe across 2 RAID6 block devices. As RAID6 allows for higher redundancy, these results indicate that it is possible to have data safety simultaneous with high performance.

While there is a performance decrease for larger file operations, it is not an order of magnitude as with other systems when moving outside of their cache. The performance delta for larger records and larger files is on the order of 20-30%. Seek bound small record systems would be better served by tuning JackRabbit to these requirements, and utilizing higher RPM drives to reduce the seek latency. This specific system was not tuned for small seek bound operations, though JackRabbit units are available that can be tuned for this.

For more information, please contact us at

sales@scalableinformatics.com

<http://www.scalableinformatics.com>

voice: +1 734 786 8423

fax : +1 866 888 3112