Impact of cache on the performance of the HP StorageWorks XP12000 Disk Array white paper

Executive summary

This white paper is intended for use by Storage Architects and advanced knowledge customers to determine if increasing cache will be a benefit for their XP Array. This applies to all XP Arrays from the HP StorageWorks XP512 Disk Array to the HP StorageWorks XP12000 Disk Array.

Should I increase the cache in my XP Array?

The answer to that question is "it depends." Performance improvement due to a larger cache will be determined by the nature of the workload. This paper presents a procedure based on measured data for predicting the performance increase that can be obtained by increasing the cache.

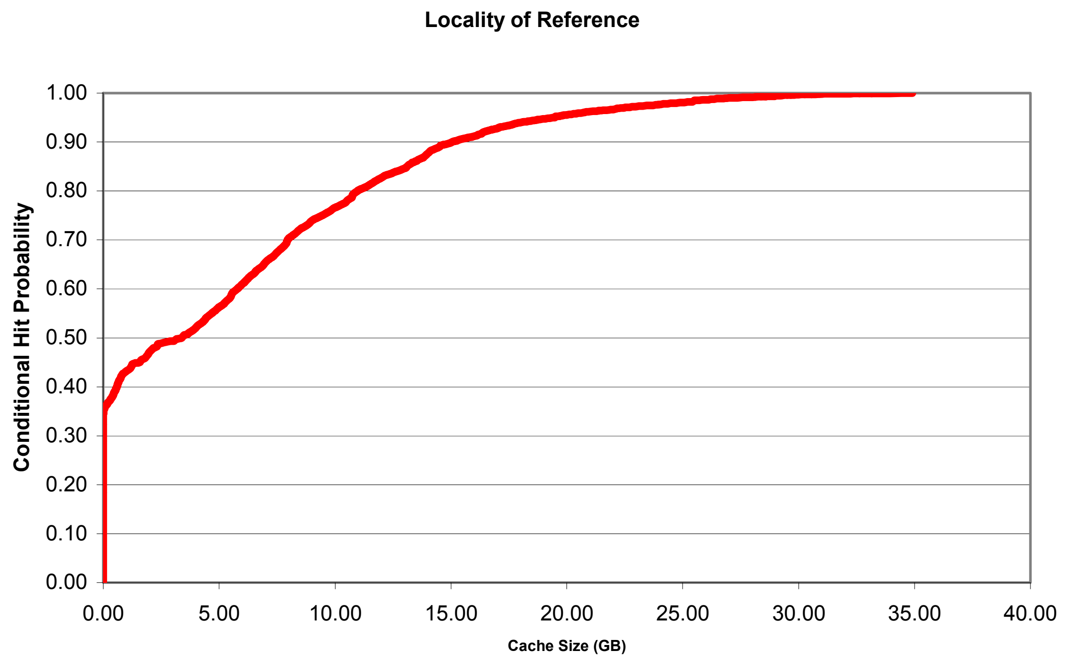
Gathering the data

The first step to predicting the impact of additional cache capacity is to collect a trace of the workload driving the array. In this test case a tool called Measurement Interface I/O Trace (MIIOT) was used on an HP-UX server, which ties into the midaemon to collect details of each I/O request issued by the server. Other required data includes the current configuration of the array, the output of xpinfo –d, and the host configuration as captured by the miiot_sys_info utility. Using this tool on a production system is not recommended.

Analyzing the data

The data is processed to determine the locality of reference in the workload. If there is no locality of reference, then cache effects are limited to pre-fetch and write-buffering operations. The locality of reference is measured using the stack distance curve.

Figure 1.



The stack distance curve shows the probability of finding a re-referenced block in an LRU cache of a given size.

The XP Array that this data came from has 16 GB of cache. After the mirrored write buffer is taken into account, approximately 30% or 4.8 GB of cache is available for use as a traditional read cache. From the stack distance curve, read that 4.8 GB of cache will capture approximately 50% of the re-references. Analysis of the locality of reference in the trace has indicated that 36% of the references in the trace are re-references, that is, they reference blocks that were already referenced earlier in the trace. Therefore, the cache hit ratio is approximately 0.5 * 0.36 = 0.18 or about 18%.

To compute the effect of cache hits on the average response time, use the following formula:

$$R(\rho) = \rho\tau_{hit} + (1-\rho)\tau_{miss},$$

Where ρ is the cache hit ratio, τ_{hit} is the average response time for a cache hit, and τ_{miss} is the response time for a cache miss.

From the trace, compute the average response time for read requests satisfied by the cache to be 0.9 ms. The average response time achieved by all other requests is 11 ms. Therefore, you have $\tau_{hit} = 0.9ms$, $\tau_{miss} = 11ms$, $\rho = 0.18$, and the overall average response time is:

$$R(0.18) = 0.18 \cdot 0.9 + 0.82 \cdot 11 = 9.2ms$$

This is the equation used to explore the effect of different cache hit ratios on application performance. See the calculations at the conclusion of this paper.

If the cache is doubled to 32 GB, you would then have 9.6 GB of cache. From the stack distance curve, read that for that cache size you would capture approximately 75% of the re-references, yielding a hit ratio of approximately 0.75 * 0.36 = 0.27 or about 27%.

To compute the speedup, compute the new average response time as:

$$R(0.27) = 0.27 \cdot 0.9 + 0.73 \cdot 11 = 8.3ms$$

Comparing the two results, you get:

$$\frac{8.3 - 9.2}{9.2} = -0.098$$

This is about a 10% reduction in overall average response time due to additional read cache hits.

Additional considerations

In addition to the performance improvement for the random reads, additional cache will boost performance two other ways. First, there is more room for pre-fetching on sequential streams. Second, there is a larger write buffer so the cache can accommodate larger bursts of write requests.

The average response time for the I/O requests in the trace was 6.4 ms, indicating that pre-fetching and write buffering were active contributors to reducing the overall average response time. Analysis of the trace revealed that 25% of the read requests were part of sequential read streams. Assuming pre-fetching was effective, most of these read requests would be cache hits.

Therefore, the average response time computation becomes:

$$R(0.43) = 0.43 \cdot 0.9 + 0.57 \cdot 11 = 6.7ms$$

Carrying this result through to include the effects of increasing the cache, you would expect the average response time to decrease to:

$$R(0.52) = 0.52 \cdot 0.9 + 0.48 \cdot 11 = 5.8ms$$

The performance improvement is:

$$\frac{5.8 - 6.7}{6.7} = -0.13$$

This is a 13% reduction in average response time.

Calculation of application speedup based on cache hit ratio

Consider the following equation previously explained, using cache hit ratios of 50–90% in 10% intervals. In addition, there are several assumptions to enumerate:

- Cache assistance affects reads only. Writes on XP Arrays are always cached. Therefore, the application speedup is dependent on the read/write mix of the application.
- In this example, assume a 60%/40% read/write mix of application I/Os. Therefore, 40% of the workload is not affected by cache size.
- The cache hit ratio is used as the input to this discussion. To relate that to cache size, use the stack distance curve previously illustrated.

First, calculate the baseline performance. Speedup will then be relative to this.

- For a 50% cache hit ratio (6-GB capacity of cache):
  R(0.5) = .5x0.9 + 0.5x11 = 5.95ms (this is the baseline read access time)
  This is the baseline, so the performance equals 100%.

- For a larger cache size—12 GB—10 GB of cache capacity correlates to a 75% hit ratio:
  R(0.75) = .75x0.9 + .25x11 = 3.425ms (this is the new read access time)
  However, only 60% of the accesses are reads. So, (5.95-3.425)/5.95x(.6)=**25% speedup.**

- For a 16-GB cache, 15 GB of cache capacity correlates to a 90% cache hit ratio:
  R(0.9) = 0.9x0.9 + 0.1x11 = 1.91 ms. So, (5.95-1.91)/5.95x(.6) = **41% speedup.**

- For a 20-GB cache, 20 GB of cache capacity correlates to a 95% cache hit ratio:
  R(0.95) = .95x0.9 + .05x11 = 1.4 ms. So, (5.95-1.4)/5.95x(.6) = **46% speedup.**

Note that these figures represent the cache-related application performance improvements **only** for the case of the preceding workload used to derive the stack distance curve. Any other working sets or associated access patterns will likely result in differing speedup ratios.

For more information

- For additional information, refer to the HP StorageWorks RAID Manager XP user guide at: http://h20000.www2.hp.com/bizsupport/TechSupport/Home.jsp
- For more information on the HP StorageWorks XP Array family, visit: www.hp.com/go/arrays

4AA0-7924ENW, September 2006