Redpaper

Jeff Berger
Paolo Bruni

DB2 for z/OS and List Prefetch Optimizer

Overview

The tight integration that IBM® DB2® for z/OS® has with the IBM System z® architecture and the z/OS environment creates a synergy that allows DB2 to use advanced z/OS platform functions. This synergy includes the enterprise storage systems.

DB2 10 for z/OS, available from October 2010, has improved its sequential detection algorithm by introducing the row level sequential detection function. This function helps reduce synchronous I/Os when doing a range scan, and the cluster ratio is less than 100%. DB2 10 for z/OS also introduced the use of list prefetch for disorganized index scans.

The IBM System Storage® DS8000® Licensed Machine Code (LMC) level R6.2, made available in November 2011, delivered several performance enhancements. When using R6.2, all DB2 I/Os, including format writes and list prefetches, are eligible for IBM High Performance FICON® (zHPF). In addition, DB2 can benefit from the new caching algorithm at the DS8000 level called List Prefetch Optimizer that the R6.2 introduced.

In 2012 IBM also introduced the DS8870, which, among other performance advantages, improves the performance of solid-state drives. (SSDs).

In June 2012, the DS8000 R6.3 increased the capacity of SSD storage to 400 GB (33 percent more capacity), introduced SSD full disk encryption, and further improvements to List Prefetch Optimizer.

In this IBM Redpaper™, we show DB2 10 for z/OS measurements using DS8000 R6.3 with the DS8800 control unit and FICON Express 8S channels and zHPF on a zEC12 processor with z/OS V1R13. Spinning HDD disks of 10K RPM and 15K RPM are evaluated, along with SSDs. The measurements highlight both the reduction gained in synchronous I/O and the reduced dependency on data reorganization. This combination offers the opportunity for better performance to critical SQL applications that access large fragments of disorganized data.

It has been recognized that solid-state drives help to reduce the need for reorganizations (REORGs), which are costly to perform periodically. However, solid-state drives are not sufficient to achieve this goal. List Prefetch Optimizer is another requirement to achieve this

goal. And yet, hardware changes alone are still *not sufficient*. DB2 for z/OS began an effort in DB2 10 to solve some of the performance problems associated with disorganized data and indexes, but DB2 10 was just a beginning, not the end of this journey toward a reduced need for REORGs. A question remains for many people whether the accumulated benefits of DB2 10 and List Prefetch Optimizer are sufficient to justify the purchase of expensive solid-state drives. To those people who remain in doubt, you should await further developments from DB2.

The type of query that benefits the most from REORG is index-to-data access when doing a range scan. In this case, if the cluster ratio is 100%, DB2 can perform sequential I/O and performance is great. When the cluster ratio starts to deteriorate, DB2 9 often used synchronous I/Os for both the clustered pages and the non-clustered pages. DB2 10 helped by ensuring that DB2 did sequential I/O for the clustered pages, but DB2 10 still does synchronous I/O for the unclustered pages. REORGs avoid these synchronous I/Os altogether. Solid-state drives only makes them faster. To more effectively reduce the need for REORGs for a range scan, further DB2 development is needed to avoid synchronous I/Os.

Introduction

The trends in DB2 storage over the past several years have seen great strides, especially with the introduction of solid-state drives (SSDs) for enterprise systems. In 2010, IBM introduced the DS8800 storage control. This storage control unit is the first from IBM to support 2.5-inch form factor disks, which enables better performance and storage capacity in a given footprint. The DS8800 also introduced 8 Gbps host adapters and device adapters.

In the channel arena, IBM delivered High Performance FICON, known as zHPF, which is a more efficient channel protocol. With FICON Express 8 channels, zHPF triples the maximum IO per second for random operations. In 2011, IBM shipped FICON Express 8S channels for its z196 processors, which are now standard on it zEC12 processors. FICON Express 8S is specifically optimized for zHPF.

zHPF requires that the old FICON channel programs be rewritten using a new System z I/O architecture and device commands. Consequently it requires support from z/OS. zHPF also requires support in the storage control unit. Prior to 2011 only a subset of the I/Os done by DB2 were eligible for zHPF.

When using z196 and R6.2, all DB2 I/Os are eligible for zHPF, including format writes and list prefetch. These two types of I/Os are also the I/Os with the most to gain from zHPF. R6.2 runs on a DS8700 or DS8800; not a DS8100 or DS8300. These enhancements also require the z196 processor or later, because on the IBM z10™ processor, I/Os that transfer more than 64 KB are not eligible for zHPF. Furthermore, PTFs to z/OS R11 or later release are required.

In addition to zHPF list prefetch, R6.2 introduced a new caching algorithm called List Prefetch Optimizer. List Prefetch Optimizer requires zHPF. Unlike zHPF list prefetch, the List Prefetch Optimizer is internal to the DS8000 code. Whereas the objective of zHPF list prefetch is to reduce the I/O connect time, the objective of List Prefetch Optimizer is to reduce disconnect time. R6.3 went further by fine tuning List Prefetch Optimizer for some types of list prefetch I/O.

DB2 10 for z/OS improved its I/O capabilities. One of the strategic goals for DB2 is to eliminate synchronous I/Os. DB2 10 improved its sequential detection algorithm with a change called Row Level Sequential Detection that helps eliminate synchronous I/Os when doing a range scan and the cluster ratio is less than 100%. DB2 10 also makes use of list

prefetch for disorganized index scans. Together these changes improve the performance when REORG is not used.

In this IBM Redpaper, after providing preliminary information about “DB2 list prefetch” on page 3 and “The changes in I/O connectivity and storage systems” on page 4, we show “The DB2 performance measurements” on page 6.

All of the measurements were done on a DS8800 storage control unit with R6.3 at Silicon Valley Laboratory. The measurements described in Figure 2 on page 7 were done on a z196 processor configured with both FICON Express 8 and FICON Express 8S channels. All other measurements were done on a zEC12 processor with FICON Express 8S channels.

DB2 list prefetch

List prefetch is not new, nor is it unique to DB2, although other products call it by a different name. List prefetch has been an integral part of DB2 for z/OS since DB2 was first developed. In other social circles a list prefetch I/O is called a scattered read, that is the pages are scattered on disk. For example, when z/OS needs to read a fragmented file from a PDSE or HFS data set, it uses scattered reads, meaning that the data is not contiguous on disk. DB2 calls these types of I/Os list prefetch I/O. VSAM does not support the notion of scattered reads and neither does IBM IMS™, nor do any other operating systems such as Linux, UNIX or Windows.

DB2 uses list prefetch I/O in a variety of situations. The following list roughly shows them in terms of relative importance:

1. Disorganized index scans (introduced in DB2 10)
2. List prefetch access path
3. Fast log apply
4. Incremental copy
5. Fragmented LOBs
6. RUNSTATS table sampling (introduced in DB2 10)

We examine the first two items in this list because they are associated with DB2 queries, and then we provide a typical query example.

- Disorganized index scans

  DB2 does lots of index scans. Frequent inserts cause page splits, which cause an index to become disorganized.

  When DB2 9 scans a disorganized index, it does synchronous I/Os, one page at a time.

  DB2 10 uses list prefetch instead. List prefetch runs asynchronously under a prefetch engine. Also starting in DB2 10, the prefetch engines are eligible to run on a System z Integrated Information Processor (zIIP). Therefore, for disorganized index scans, DB2 10 saves the CPU cost of the synchronous I/Os, and helps make possible to overlap the I/O with the other class-2 CPU time.

- List prefetch access path

  This access path refers to the way that DB2 can sort the row identifiers (RID) so that DB2 can read the data in skip-sequential[1] order. Sorting the RIDs ensures that DB2 does not have to reread the same page and minimizes seek distances. List prefetch access path is also a characteristic of DB2 for Linux, UNIX, and Windows (LUW). DB2 for z/OS and LUW both have the capability to construct a sorted RID list but DB2 for LUW then has to do a lot

[1] *Skip sequential* means that the pages numbers are in ascending (or descending) order but the pages are not contiguous.

more I/Os (which causes extra CPU time) because the operating system does not support the notion of scattered reads. On z/OS, reading 32 discontiguous pages costs in the range of 7 - 9 times less CPU time than 32 individual I/Os. Until now, that CPU savings has been the singular advantage of list prefetch. zHPF and List Prefetch Optimizer now extend that advantage.

The DB2 REORG utility reorganizes the data according to a cluster index. The cluster ratio is then said to be 100%. If a query subsequently uses the cluster index to do a range scan of some of the rows, the row access is sequential. However, if the query filters the rows according to a filter predicate, then the row access is skip sequential. If the non-filtered rows are dense, DB2 can use dynamic prefetch to read all of the skipped pages. If the non-filtered rows are sparse, DB2 uses synchronous I/O. DB2 does not use list prefetch in any of these cases.

When more rows are inserted into the table, DB2 tries to keep the rows in cluster sequence. Optionally REORG can distribute free space throughout the table, and the inserts use that free space; eventually, however, the free space becomes exhausted and DB2 can no longer maintain perfect clustering. Gradually the cluster ratio degrades. If the RUNSTATS utility is executed to collect statistics, the DB2 optimizer recognizes that the cluster ratio is low, in which case it may select the list prefetch access path.

Using a US phone book as an example, suppose this phone book is organized by Lastname, Firstname, and State. Now, suppose that the phone book is reorganized (REORG). If a query needs to read the rows for all persons named Smith, the rows are sequential. If a query needs to read only the Smith names in a particular state, the rows are skip-sequential. If that state is Alaska, the rows are sparse, in which case the query does synchronous I/O. If that state is California, the rows are dense, in which case DB2 does sequential I/O and it reads all of the intermediate pages. For example, if one out of five Smith pages contain at least one person from California, then DB2 reads five times as many pages as the number of Getpages, but the I/O is fast and it is asynchronous.

What happens now as the cluster ratio degrades? New Smiths added to the phone book are stored out of sequence, which means that they are stored on unclustered pages. These queries have to do synchronous I/O for all unclustered pages. The only way to avoid these synchronous I/Os is either by using REORG again, or by running RUNSTATS and getting the DB2 optimizer to choose the list prefetch access path.

The changes in I/O connectivity and storage systems

FICON Express 8S channels are optimized for High Performance FICON for System z (zHPF), providing more I/O throughput and lower response time than FICON Express 8. FICON Express 8S is available for the IBM zEnterprise® 196 (z196) and zEnterprise 114 (z114) servers.

IBM System Storage DS8000 series offers higher scalability with additional tiering capabilities and new drive options. Of particular interest for the z platform and DB2 for z/OS are the following options:

- DS8000 I/O Priority Manager

  I/O Priority Manager together with z/OS Workload Manager (zWLM) enables more effective storage consolidation and performance management, integrating with zWLM. It is intended to improve disk I/O performance for important workloads and drive I/O prioritization to the disk system[2].

- DS8000 enhancements to support new zHPF format writes

zHPF has also been enhanced to support format writes. The performance value of these enhancements are highest for small records, which are typically used for databases.

- DB2 list prefetch optimizer

  zHPF has been enhanced to support DB2 list prefetch. These enhancements include a new cache optimization algorithm that can greatly improve performance and hardware efficiency. When combined with DB2 10 for z/OS, it can demonstrate large sequential processing performance improvements.

We provide a brief description of the enhancements benefitting list prefetch I/O.

List prefetch channel program

We now look more closely at several performance problems that ail list prefetch with FICON, starting with the channel itself. Here, we consider list prefetch using 4 KB pages.

Figure 1 shows what a FICON list prefetch channel program looks like when DB2 reads 32 discontiguous pages. Each page (or record, in terms of storage lexicon) requires two channel command words (CCWs.) The first CCW, known as Locate Record, contains the seek address on disk. Subsequent CCWs are considered to be within that locate record domain. The second CCW then points at the DB2 buffer. Next, there is another Locate Record CCW, which is referred to as an imbedded locate record. This begins a new domain. Thus, each page requires a pair of CCWs, and a list prefetch I/O for 32 pages requires 64 CCWs. That is many more CCWs than is required to read 32 sequential pages using Read Track Data. Using Read Track Data, z/OS can read every page on a track using a single CCW. Thus, DB2 can read 32 contiguous pages using just three or four CCWs (actually, it is four or five if we count the initial Prefix CCW).

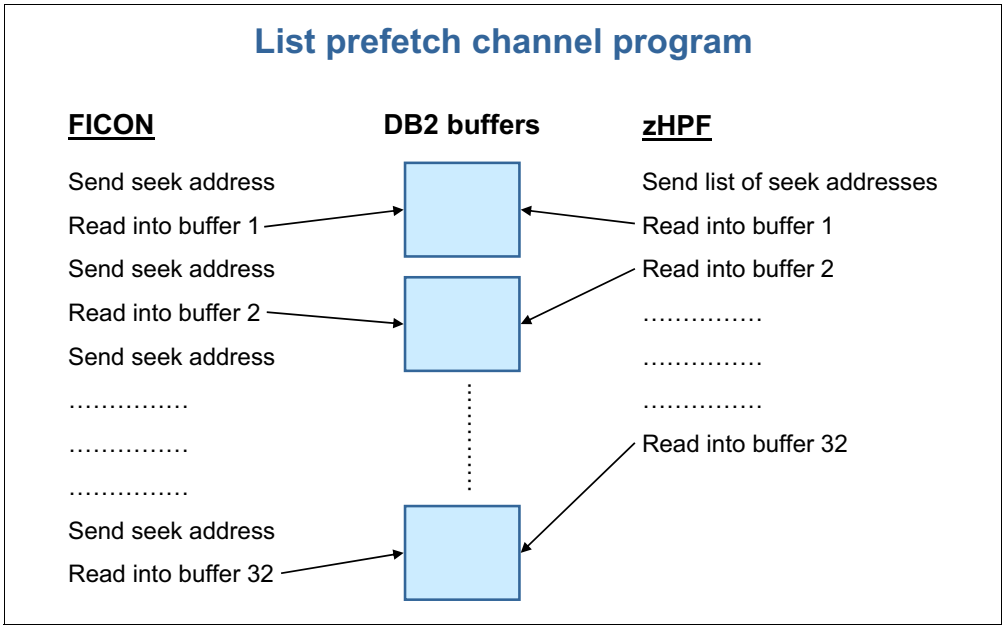

Figure 1 List prefetch channel program

In contrast, zHPF enables z/OS to send a list of seek addresses in a single transfer known as a transport control word (TCW). This TCW is followed by a series of read commands that look the same whether or not the data is contiguous. Thus, from a channel perspective, whether

[2] This function is not included in these measurements. For its full description, see *DS8000 I/O Priority Manager*, REDP-4760.

the pages are contiguous or not is nearly irrelevant. One problem, however, is that the original zHPF architecture was limited to a list of 22 seek addresses in one I/O. To deal with this limitation, media manager must divide 32 pages into two distinct I/Os. So, transitioning from FICON to zHPF causes the number of I/Os to increase. This architecture limitation was removed with FICON Express 8S. Thus, with FICON Express 8S, transitioning from FICON to z/OS does not cause an increase in the number of I/Os.

List prefetch from disk

Another problem with FICON list prefetch pertains to the physical disks. Because FICON protocols process the pages serially, FICON list prefetch performs one seek at a time. Compared to synchronous I/O, there is a little bit of savings in terms of channel time, but there is no savings when it comes to disk access time. Thus, the fact that the pages might be spread across various disks in a RAID rank is immaterial with FICON. For example, if it takes 3.5 ms to read one page, FICON takes 32x3.5 ms, or 112 ms to process 32 pages. Most people perceive that 3.5 ms is a fast I/O response time for a cache miss, and those same people would perceive 112 ms to be slow, and yet these are typical I/O response time characteristics for FICON.

That is not so with List Prefetch Optimizer. With List Prefetch Optimizer, list prefetch has considerably more advantages over synchronous I/O because the pages are not processed serially.

The DB2 performance measurements

All measurements used in this zHPF study were done using a DS8800 control unit. The measurements described in Figure 2 on page 7 were done on a z196 processor configured with both FICON Express 8 and FICON Express 8S channels. All other measurements were done on a zEC12 processor with FICON Express 8S channels. Spinning disks of 10K and 15K RPM, as well as SSD, were evaluated. The DS8870 has not been evaluated with DB2 list prefetch, but it is expected that SSD would perform better on a DS8870 than on a DS8800

zHPF to FICON

We start with measurements of channel performance comparing contiguous data and non-contiguous data. An example would be an index scan. If the index is organized, the pages are contiguous; if the index is disorganized, the pages are discontiguous. In the case of indexes, the pages are randomly distributed across the data set. Another example of discontiguous data is data access using a sorted RID list scan, in which case the pages are skip-sequential rather than random. But, in our comparison of contiguous to non-contiguous data where the cache-hit ratio is 100%, whether the pages are random or skip-sequential does not matter. This particular comparison is designed to illustrate how zHPF list prefetch reduces the performance gap between contiguous pages and non-contiguous pages. List Prefetch Optimizer and the type of disks are not relevant to this discussion because the data is assumed to be cache-resident.

Figure 2 compares the I/O response time for contiguous pages to non-contiguous pages, and compares zHPF to FICON, assuming that the data is cache-resident. These measurements were done using FICON Express 8S. The FICON response time for contiguous pages was 0.64 ms and zHPF had little effect. The FICON response time for discontiguous pages was 2.4 ms and zHPF reduced this to 0.983 ms. So, the effect of data being disorganized with

FICON is to increase the response time by 3.7x, and the effect of data being disorganized with zHPF is to add only 37% to the response time.

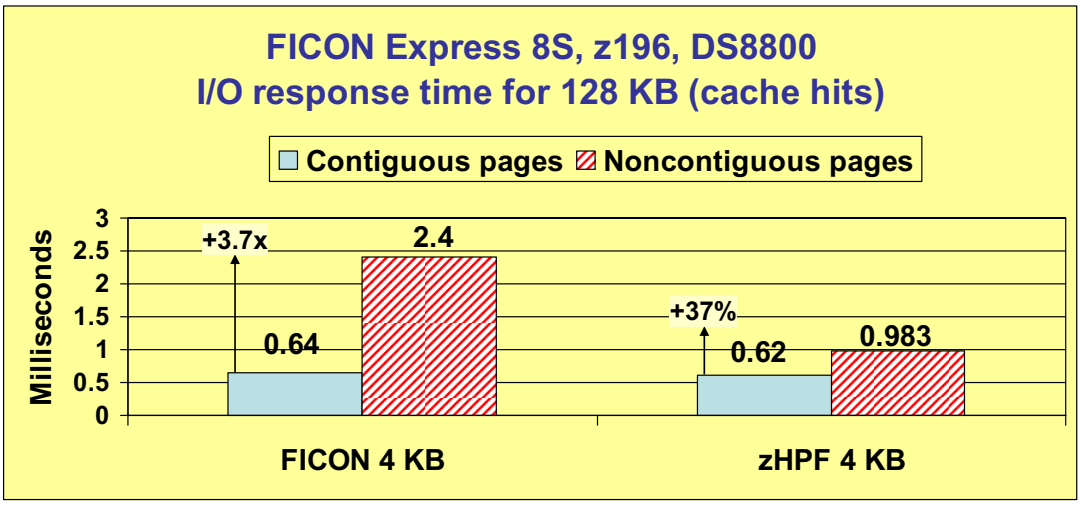

Figure 2 Cache-resident response time: Contiguous pages versus non-contiguous pages, HPF versus FICON

List prefetch with sparse skip sequential

The performance of list prefetch is sensitive to many factors including the type of disk and the page access patterns. To start, see Figure 3 on page 8.

The test case here is skip-sequential using 4 KB pages and spinning 10K RPM disks, and where the skip distances are above 17 pages. Spinning disks of 10K RPM and 15K RPM were both measured. The performance is measured in terms of MBps throughput. For example, when the skip distance is 65 pages, FICON throughput is 2.7 MBps and zHPF throughput is 7.5 MBps. The reason that zHPF performs so much better than FICON is the List Prefetch Optimizer.

Figure 3 also shows how much faster the 15K disks are than the 10K disks, with or without zHPF. As the pages become denser, the advantage of the 15K disks increases because when the pages become dense, the rotation speed becomes a more dominant component of response time than seek time.

When using SSD, the same test case with a skip distance of 17 pages achieves 12 MB/sec with FICON and 98 MB/sec with zHPF, again due to List Prefetch Optimizer. So, the combination of zHPF and SSD raises the throughput from 15.6 MB/sec all the way to 90 MB/sec. That is a profoundly large difference. Furthermore, the difference is even greater if we compare SSDs to the 10K disks.

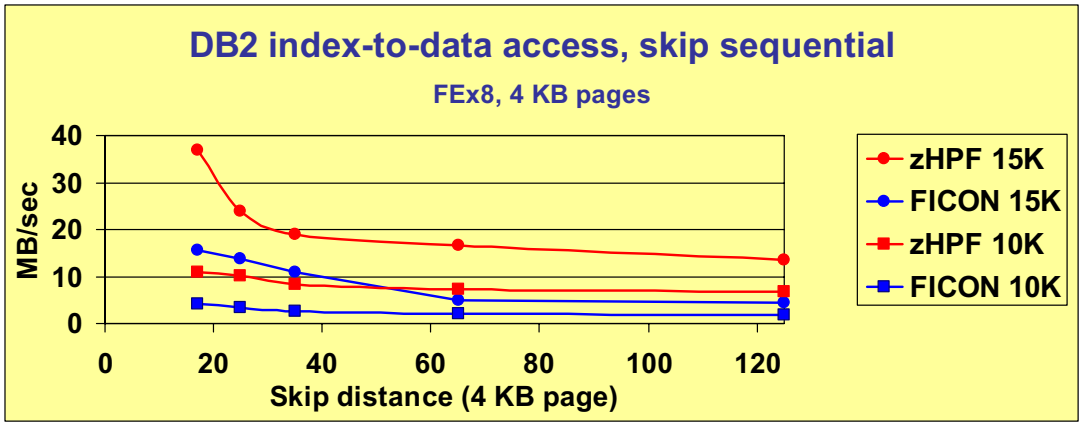

Figure 3 Sparse skip sequential for disks

Figure 4 shows the same set of test cases, but it compares SSDs to the 15K disks.

When using SSD, the same test case with a skip distance of 17 pages achieves 12 MB/sec with FICON and 98 MB/sec with zHPF, again due to List Prefetch Optimizer. So, the combination of zHPF and SSD raises the throughput from 15.6 MB/sec all the way to 90 MB/sec. That is a profoundly difference. Furthermore, the difference is even greater if we compare SSDs to the 10K disks

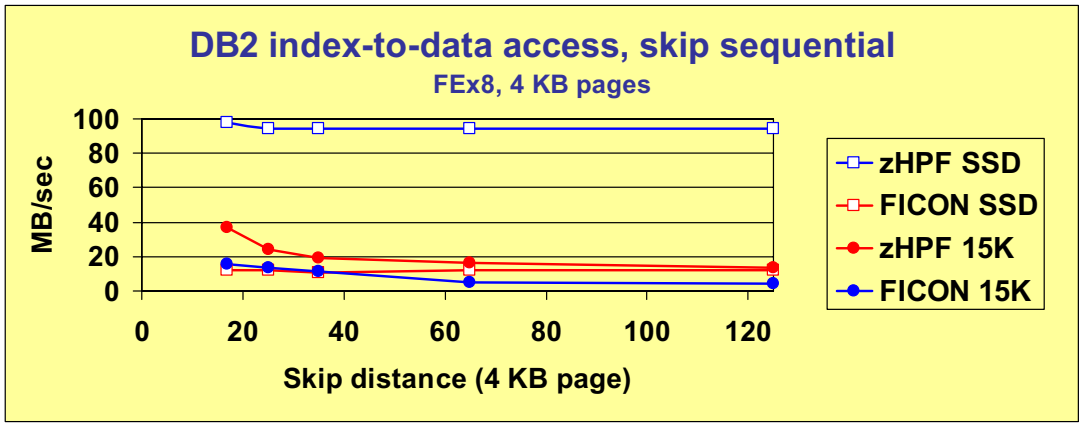

Figure 4 Sparse skip sequential for solid-state drives

Figure 5 helps to illustrate how List Prefetch Optimizer takes advantage of a RAID 5 configuration. RAID 5 is the most common type of RAID configuration today. In the DS8000 the disks are grouped in sets of eight, called *ranks*. For example, one DS8800 frame may contain up to 30 ranks of eight disks, for a total of 240 disks. Some RAID 5 ranks contain one spare disk; such a RAID 5 rank is called *6+p*. Those ranks that do not contain a spare are called *7+p*. The “p” indicates parity, which is distributed evenly across the disks (other than the spare disk).

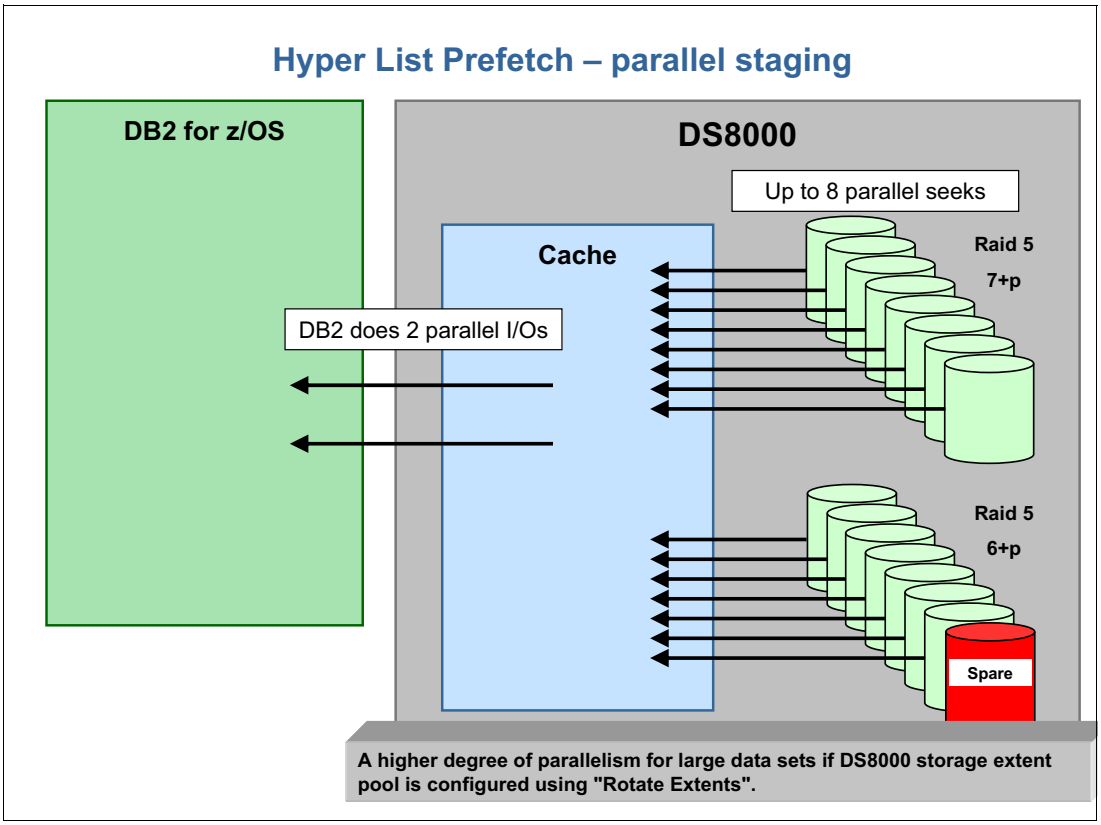

Figure 5 Hyper List Prefetch: Parallel staging

The user data is spread across all of the non-spare disks. So, by scheduling all of the pages in the list in the disk subsystem at the same time, the List Prefetch Optimizer increases the amount of seek parallelism. Because DB2 typically does two parallel list prefetch I/O operations, FICON also achieves some parallelism, but not nearly as much as List Prefetch Optimizer.

Figure 6 and Figure 7 on page 10 illustrate how list prefetch appears in an IBM RMF™ Cache Activity Report with one of these sparse skip-sequential test cases. The skip distance shown is 17 pages. In both cases, the cache started out empty.

Figure 6 is the report with FICON.

Sparse skip sequential (1 out of 17 pages) – FICON

```
CACHE I/O    -------------READ I/O REQUESTS-------------
REQUESTS       COUNT      RATE       HITS      RATE      H/R
NORMAL        457329      1066        597       1.4    0.001
SEQUENTIAL         0       0.0          0       0.0      N/A
-----------------------CACHE MISSES-----------------------
REQUESTS      READ     RATE   WRITE     RATE  TRACKS     RATE
NORMAL      456732     1065       0      0.0  457854     1067
SEQUENTIAL       0      0.0       0      0.0       0      0.0

----HOST ADAPTER ACTIVITY---   --------DISK ACTIVITY-------
                BYTES  BYTES             RESP   BYTES  BYTES
                 /REQ   /SEC             TIME    /REQ   /SEC
READ             4.1K   4.4M   READ    1.463   55.2K  59.0M
```

Figure 6 List prefetch in an RMF Cache Activity Report: FICON

Figure 7 shows that zHPF improved the cache hit ratio to 0.876, reducing the number of cache misses from 456732 (in Figure 6) to 58250. And yet, zHPF did not significantly change the number of tracks touched (it actually increased by 4.6%). The cache-hit statistics do not show any sequential activity, which indicates that the pages were not dense enough to merit any sequential prestaging.

Sparse skip sequential (1 out of 17 pages) – zHPF (LPO)

```
CACHE I/O    -------------READ I/O REQUESTS-------------
REQUESTS       COUNT      RATE       HITS      RATE      H/R
NORMAL        470598      2285     412348      2002    0.876
SEQUENTIAL         0       0.0          0       0.0      N/A
-----------------------CACHE MISSES-----------------------
REQUESTS      READ     RATE   WRITE     RATE  TRACKS     RATE
NORMAL       58250    282.8       0      0.0  479002     2325
SEQUENTIAL       0      0.0       0      0.0       0      0.0

----HOST ADAPTER ACTIVITY---   --------DISK ACTIVITY-------
                BYTES  BYTES             RESP   BYTES  BYTES
                 /REQ   /SEC             TIME    /REQ   /SEC
READ             4.1K   9.4M   READ    3.551    4.7K  10.9M
```

Figure 7 List prefetch in an RMF Cache Activity Report: zHPF

The Host Adapter Activity shows that each Locate Record domain reads 4 KB. In the FICON report (Figure 6), the Device Activity shows that for every 4 KB page read, the device read 55.2K. That is because the DS8800 read a full track, including some metadata. However, the zHPF report shows that each time that a disk was accessed, the DS8000 read 4.7 KB, which is, the one 4 KB page plus some metadata. So, List Prefetch Optimizer read considerably less data from disk compared to FICON.

List prefetch with dense skip sequential

Figure 8 shows some skip-sequential patterns with dense pages. The RUNSTATS table sampling feature of DB2 10 was used as a proxy for DB2 queries. Whereas RUNSTATS ordinarily uses sequential prefetch, the table sampling option of RUNSTATS uses list pre-fetch to sample a set of pages. The higher the sampling rate is, the denser the pages are. Ideally we want to see that the fewer pages that we read, the faster the utility is, but this ideal situation is not generally what happens, because list prefetch incurs a performance penalty for skipping a small number of pages. So, for these scenarios, we start by looking at the elapsed time instead of throughput. The table in this measurement contained 8 million 4 KB pages.

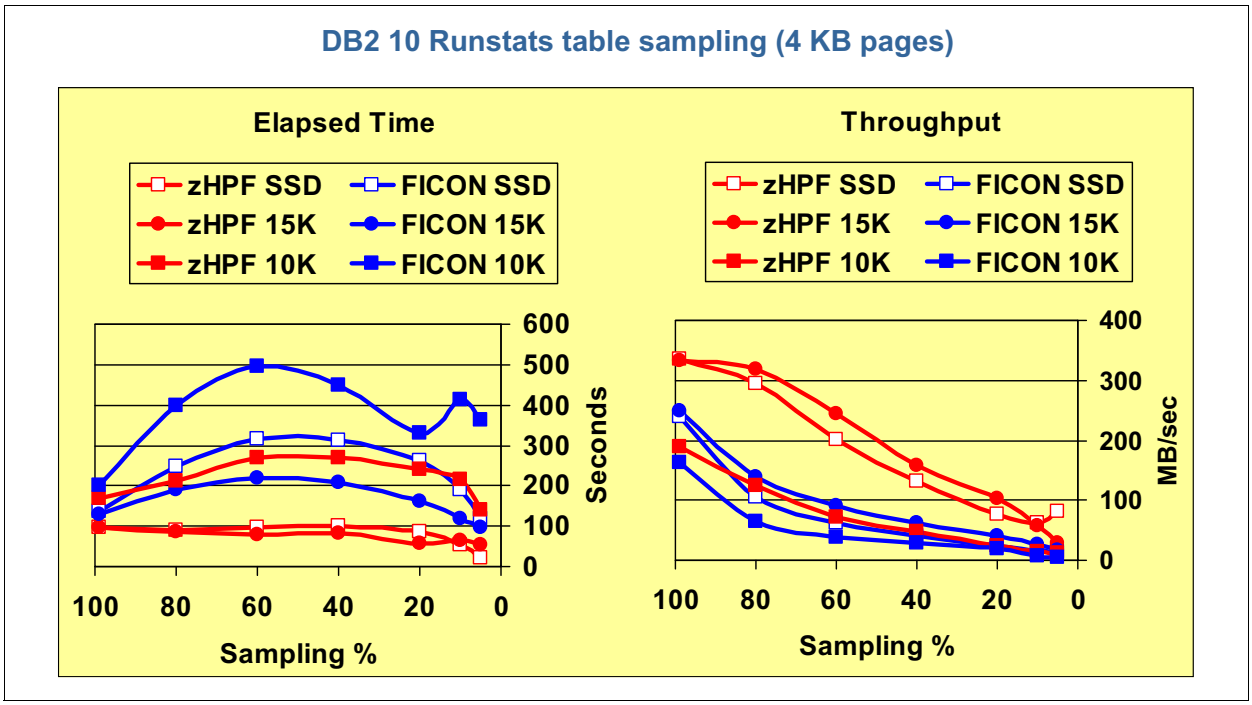

Figure 8 Dense skip sequential

To illustrate, look at the FICON performance. With 99% sampling, the elapsed time was 200 seconds. With 80% sampling, the elapsed time doubled to 399 seconds, and with 60% sampling the elapsed time peaked at nearly 500 seconds. Subsequently as the sampling rate was lowered, the elapsed time started to drop, but even with sampling in the range of 5 - 20%, the elapsed time remained higher than it was for 99% sampling.

Roughly the same behavior occurs with solid-state drives. Readers may observe that with the 10K spinning drives, the elapsed time briefly spiked down, going from 20% sampling to 10% sampling. That was because of the “two steps forward, one step back” nature of list prefetch when DB2 uses list prefetch with FICON. That phenomenon does not occur with SSD disks nor does it occur with List Prefetch Optimizer for the most part. (There is actually a little bit of “two steps forward, one step back” with FICON Express 8, but not with FICON Express 8S, because FICON Express 8S can read all 32 pages using zHPF in a single I/O.)

Figure 8 shows that the 15K drives outperformed the SSD drives. This is a surprising result, but it should be noted that the DS8870 is likely to make SSD look more favorable relative to 15K spinning drives. Nevertheless, since List Prefetch Optimizer streams the data into the cache when the pages are dense, thereby eliminating most of the seek latency and rotational

delays, spinning 15K drives do very well for this scenario. List Prefetch Optimizer is what makes this possible.

The combination of List Prefetch Optimizer and SSDs or 15K drives achieves our objective that reading fewer pages should not increase the elapsed time. With List Prefetch Optimizer and with either SSD or 15K drives, the elapsed time with 99% sampling was 100 seconds. With SSD the elapsed time stays at roughly 100 seconds until the sampling rate drops to 20%. Thereafter the elapsed time starts to drop. With 15K drives the elapsed time drops more than the SSD drives, but when DB2 reads 10% or fewer pages, SSD performance begins to overtake the 15K drives.

The combination of List Prefetch Optimizer and SSD is the only combination that actually achieves the objective that reading fewer pages should not increase the elapsed time. With List Prefetch Optimizer and SSD, the elapsed time with 99% sampling was 100 seconds, and the elapsed time stays at roughly 100 seconds until the sampling rate drops to 20%. Thereafter the elapsed time starts to drop.

Figure 8 on page 11 also shows the throughput for these dense skip-sequential cases.

Figure 9 and Figure 10 illustrate the RMF Cache Activity reports for dense skip sequential using 50% sampling respectively for FICON and zHPF. Here the Host Adapter Activity indicates that on average each Locate Record domain read two pages (8 KB). Using FICON, only 11% of the tracks were read into the cache sequentially. Using List Prefetch Optimizer, 87% of the tracks were read into the cache sequentially. That difference is because List Prefetch Optimizer is better at detecting that the pages are dense skip sequential.

```
CACHE I/O    -------------READ I/O REQUESTS-------------
REQUESTS       COUNT     RATE      HITS     RATE     H/R
NORMAL          1832K     3629     1366K     2705   0.746
SEQUENTIAL    229902    455.3    227853    451.2   0.991
----------------------CACHE MISSES-----------------------
REQUESTS      READ    RATE   WRITE    RATE  TRACKS    RATE
NORMAL      466212   923.2       0     0.0  595320    1179
SEQUENTIAL    2049     4.1       0     0.0   71861   142.3

----HOST ADAPTER ACTIVITY---   --------DISK ACTIVITY-------
                BYTES  BYTES             RESP   BYTES  BYTES
                 /REQ   /SEC             TIME    /REQ   /SEC
READ             8.0K  32.7M  READ     0.952   55.3K  73.1M
```

Figure 9 RMF Cache Activity reports for dense skip sequential using 50% sampling: FICON

Figure 10 shows zHPF.

```
CACHE I/O     -------------READ I/O REQUESTS-------------
REQUESTS        COUNT      RATE       HITS      RATE      H/R
NORMAL         122374     413.4     118240     399.5    0.966
SEQUENTIAL      1940K      6553      1930K      6522    0.995
-----------------------CACHE MISSES-----------------------
REQUESTS       READ     RATE    WRITE     RATE  TRACKS     RATE
NORMAL         4134     14.0        0      0.0   88811    300.0
SEQUENTIAL     9284     31.4        0      0.0  608290     2055

----HOST ADAPTER ACTIVITY---    --------DISK ACTIVITY-------
                  BYTES  BYTES             RESP   BYTES  BYTES
                   /REQ   /SEC             TIME    /REQ   /SEC
READ              8.0K  55.7M  READ      1.620   53.2K 125.4M
```

Figure 10 RMF Cache Activity reports for dense skip sequential using 50% sampling: zHPF

List prefetch with disorganized index scans

DB2 10 is the first version of DB2 for z/OS to use list prefetch for a disorganized index scan. The measurements shown in Figure 11 on page 13 describe the throughput for a disorganized index scan in terms of MBps as a function of the percentage of the amount of index that is read. Figure 11 shows FICON-only numbers, comparing DB2 10 to DB2 9, with both 10K spinning disks and solid-state drives. Throughput increases as a higher percentage of the index is read, because the adaptive caching algorithm in the DS8000 increases the cache-hit ratio as DB2 reads more pages in the index.

These measurements show that SSD is up to six times faster than the 10K spinning disks. For the short scans, with SSD DB2 10 is up to 10 times faster than DB2 9. With spinning disks, the advantage of DB2 10 is higher for long scans than for short scans, with increased throughput by as much as 150%.

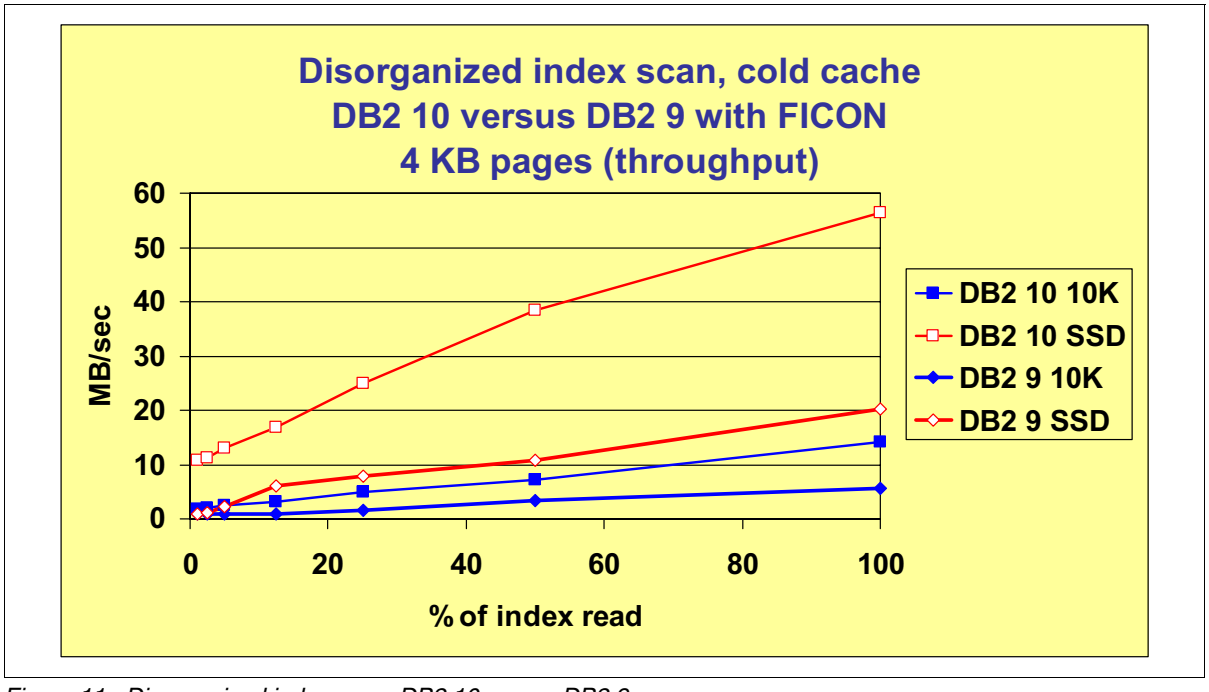

Figure 11 Disorganized index scan: DB2 10 versus DB2 9

Figure 12 shows the elapsed time of DB2 10 comparing three device types with and without zHPF. Previously we showed only FICON Express 8 measurements with 10K spinning drives and solid-state drives. Here, we also have FICON Express 8S measurements using 15K spinning drives.

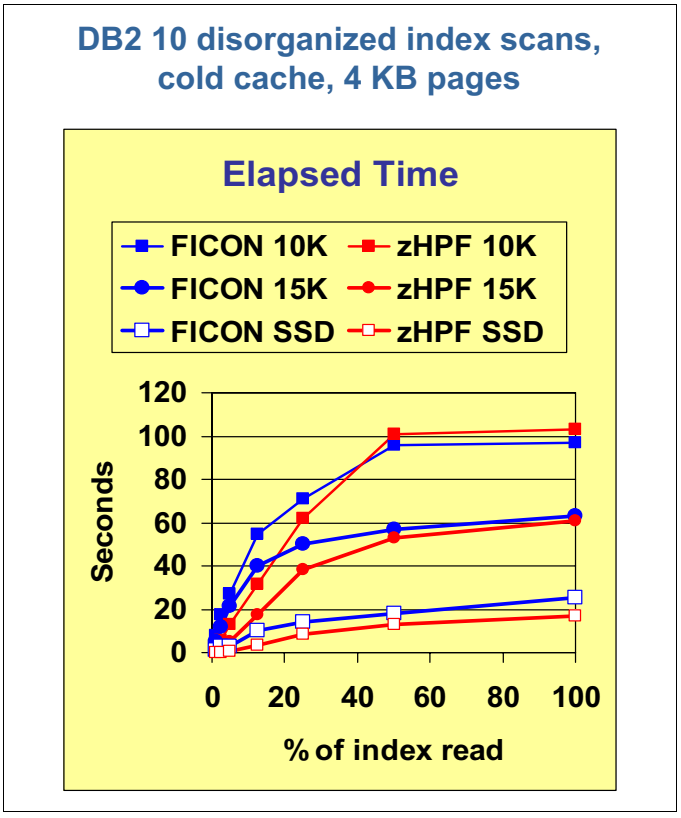

Figure 12 Disorganized index scan: Elapsed time by device type

Figure 13 shows the I/O response time for the same measurements.

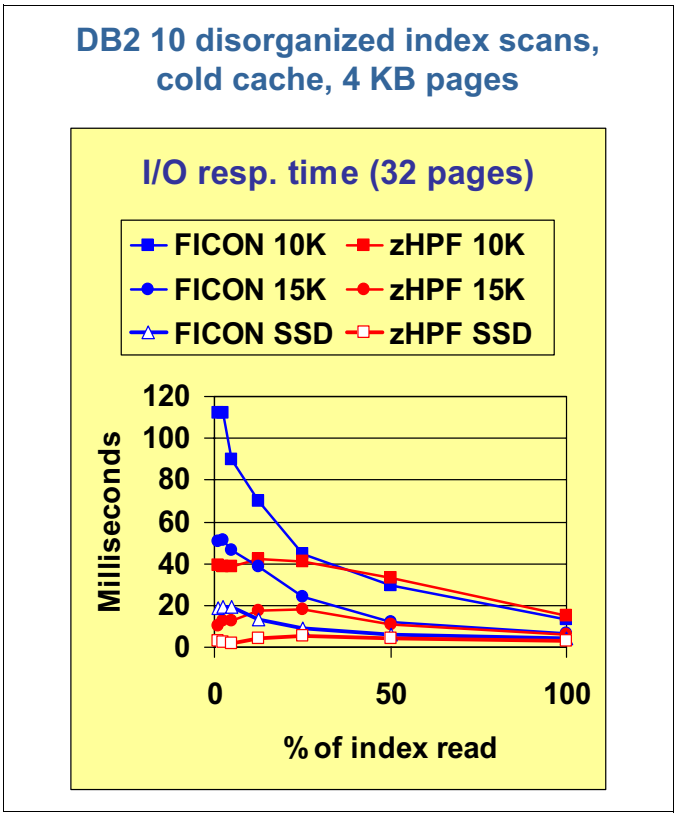

Figure 13 Disorganized index scan: zHPF by device type

Notice that for the 10K spinning drives, when DB2 reads more than 35% of the index, List Prefetch Optimizer is a little bit slower than FICON. That happens because FICON uses full track caching, which brings the entire index into the cache quickly. List Prefetch Optimizer, however, starts off doing record caching, which is optimal for a small index scan, but not so efficient for a large index scan.

The cache statistics in Figure 14 and Figure 15 on page 16 illustrate what happens, respectively with FICON and zHPF with List Prefetch Optimizer. When DB2 read 12.5% of the index using FICON, the cache-hit ratio was 0.469 and the DS8800 read 12 index pages into the cache every time there was a cache-miss. For that same test case using List Prefetch Optimizer, the cache-hit ratio was 0.788, but the DS8800 read 18.8 KB into the cache each time that a track was touched. So, the amount of time is longer for List Prefetch Optimizer to get the entire index into the cache.

```
CACHE I/O    -------------READ I/O REQUESTS-------------
REQUESTS       COUNT      RATE       HITS      RATE      H/R
NORMAL         42333     717.5      19847     336.4    0.469
SEQUENTIAL         0       0.0          0       0.0      N/A
-----------------------CACHE MISSES-----------------------
REQUESTS       READ     RATE   WRITE     RATE  TRACKS     RATE
NORMAL        22486    381.1       0      0.0   22533    381.9
SEQUENTIAL        0      0.0       0      0.0       0      0.0

----HOST ADAPTER ACTIVITY---   --------DISK ACTIVITY-------
                 BYTES  BYTES             RESP   BYTES  BYTES
                  /REQ   /SEC             TIME    /REQ   /SEC
READ             4.1K   3.0M   READ     4.706   55.3K  21.1M
```

Figure 14 Disorganized index scan: FICON for 12.5% index-read

Figure 15 shows zHPF.

```
CACHE I/O    -------------READ I/O REQUESTS-------------
REQUESTS       COUNT      RATE       HITS      RATE      H/R
NORMAL         42134      1317      33208      1038    0.788
SEQUENTIAL         0       0.0          0       0.0      N/A
-----------------------CACHE MISSES-----------------------
REQUESTS       READ     RATE   WRITE     RATE  TRACKS     RATE
NORMAL         8926    278.9       0      0.0   37273     1165
SEQUENTIAL        0      0.0       0      0.0       0      0.0

----HOST ADAPTER ACTIVITY---   --------DISK ACTIVITY-------
                 BYTES  BYTES             RESP   BYTES  BYTES
                  /REQ   /SEC             TIME    /REQ   /SEC
READ             4.2K   5.5M   READ     6.063   18.9K  22.0M
```

- On average HLP read only 18.9K per DA request
- Record caching and partial track staging causes the "number of tracks" staged/touched to exceed that of FICON by as much as 100%

Figure 15 Disorganized index scan: zHPF for 12.5% index-read

Figure 16 graphically shows these cache statistics. The index consists of about 28,000 tracks. The number of FICON cache misses never exceeds the number of tracks in the data set. List Prefetch Optimizer has a lot fewer cache misses than FICON, and yet the number of times that a partial track is staged eventually becomes more than double the number of tracks in the data set. Sometimes, List Prefetch Optimizer might even reread pages into the cache that were already there.

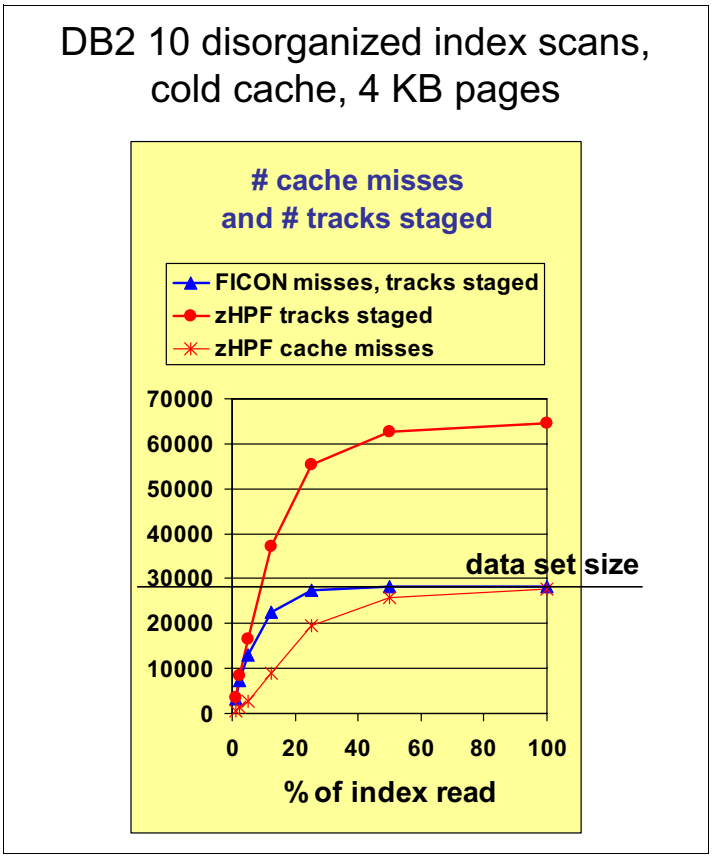

Figure 16 Disorganized index scan: Cache statistics

With that though in mind, let's examine Figure 17. This graph focuses on I/O response time for solid-state drives, comparing List Prefetch Optimizer to FICON. With List Prefetch Optimizer, the I/O response time is as low as 1.66 ms (when DB2 read 5% of the index). As DB2 reads more of the index, the DS8800 transitions to partial track-staging and then full track-staging. As the cache misses result in more data being read, the SSD I/O response time increases up to a maximum of 5 ms. Finally as a result of the entire index being in cache, the SSD I/O response time starts to drop, and it finishes at an average of 2.72 ms after DB2 has read 100% of the index. Thus, it appears that the best strategy to use with SSD is record caching to avoid rereading the same pages into cache.

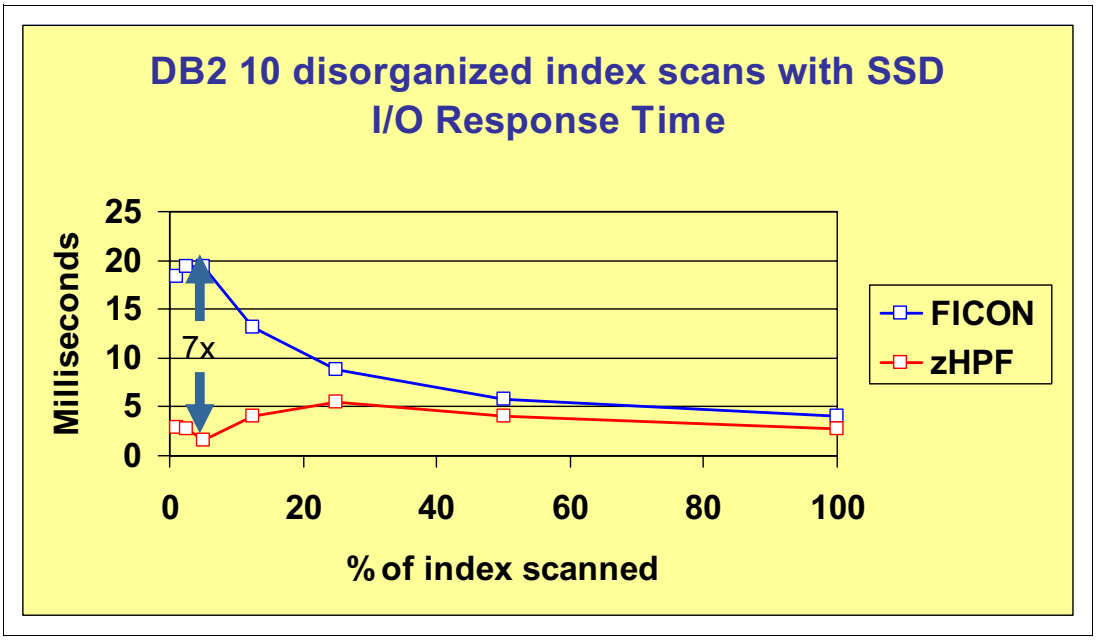

Figure 17 Disorganized index scan: SSD I/O response time

Figure 18 is another graph showing throughput, similar to Figure 11 on page 13, which compared DB2 10 to DB2 9 without zHPF.

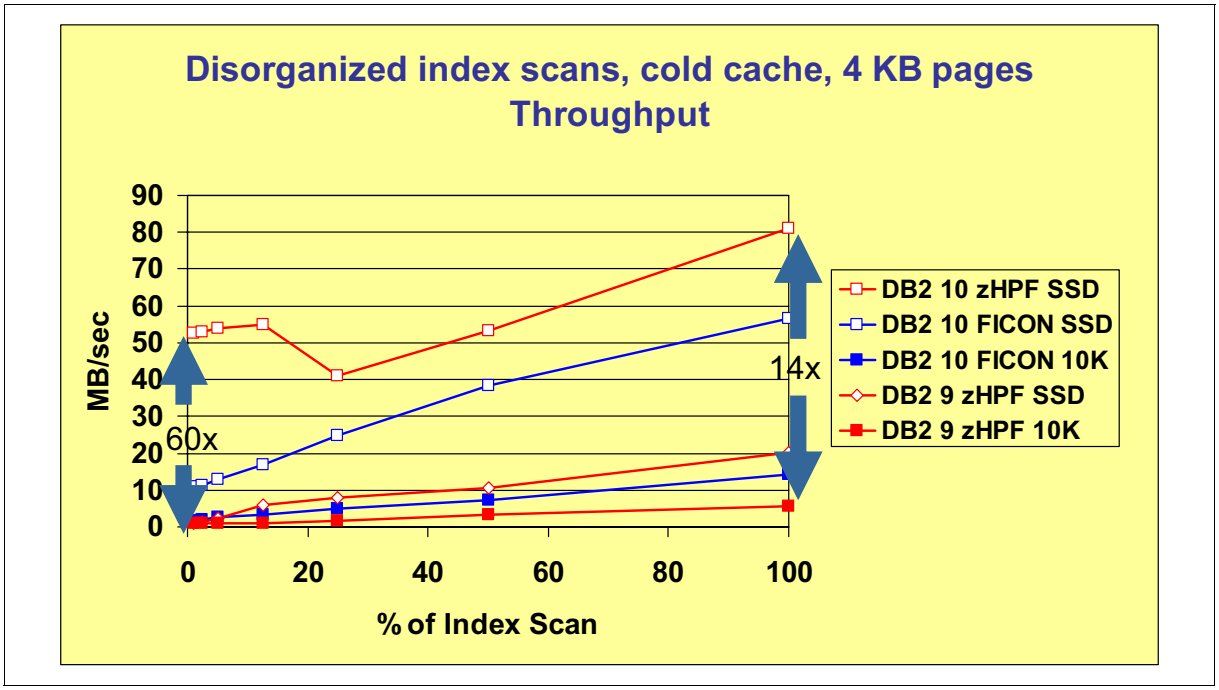

Figure 18 Disorganized index scan: Cold cache, DB2 10 versus DB2 9

Figure 18 has zHPF added to the graph. DB2 9 with FICON is removed from the graph, because the effect of zHPF on DB2 performance is insignificant compared to other factors. Figure 11 on page 13 showed how SSD and DB2 10 each contributed significantly to the performance improvements. zHPF compounds the benefits of DB2 10 and SSD, resulting in a combined throughput increase of 60 times for short scans and 14 times for a full index scan.

Conclusions

zHPF is the strategic direction for channel architecture in z/OS. At the present time, the IBM DS8700 and DS8800 are the only storage products that support zHPF for list prefetch I/O operations. List Prefetch Optimizer is a new IBM cache management algorithm used for zHPF list prefetch I/Os. List Prefetch Optimizer lowers I/O response time by exploiting RAID architecture to dramatically increase the cache hit ratio.

Prior to DB2 10 for z/OS, list prefetch I/O was commonly associated with sorted RID list scans, which are used for index-to-data access when the index cluster ratio is low, or when the data is disorganized. DB2 10 for z/OS introduced the use of list prefetch I/O when indexes are disorganized, thereby eliminating synchronous index I/Os. Because DB2 10 is more dependent on the performance of list prefetch, hardware functions such as List Prefetch Optimizer are important for getting the most out of DB2 10.

Solid-state drives are able to take greater advantage of List Prefetch Optimizer than spinning disks, because much of the performance improvement achieved by List Prefetch Optimizer comes from the high degree of I/O parallelism that spinning disks have difficulty achieving. Solid-state drives are still expensive for the average z/OS customer, but the cost is expected to decrease as the technology matures. Furthermore, some of that cost may be mitigated by reducing the frequency of REORGs, which is made possible by the fact that disorganized data and indexes perform better.

References

For additional information, see the following publications:

- *DB2 10 for z/OS Performance Topics,* SG24-7942
- *Ready to Access DB2 for z/OS Data on Solid-State Drives*, REDP-4537
- *DS8000 I/O Priority Manager*, REDP-4760-01
- *IBM System Storage DS8000: Architecture and Implementation,* SG24-8886-02
- *DS8000 Performance Monitoring and Tuning*, SG24-8013
- *IBM System Storage DS8700 and DS8800 Introduction and Planning Guide*, GC27-2297-08
- Does REORG Matter?

  http://www.ibm.com/developerworks/data/library/dmmag/DMMag_2011_Issue4/ReOrg/?ca=drs

The team who wrote this IBM Redpaper

This paper was produced by a team of specialists working at the Silicon Valley Laboratory, San Jose.

Jeff Berger is a member of the DB2 for z/OS performance department with IBM Silicon Valley Laboratory. For most of his 27 years at IBM, Jeff has worked on system performance of IBM mainframes, specializing in DASD storage and database systems, both hardware and software. Jeff has contributed several patents, IBM Redpaper publications, and several papers, which were published by Computer Measurement Group and the DB2 IDUG Solutions Journal.

Paolo Bruni is an ITSO Project Leader based in the IBM Silicon Valley Laboratory. Paolo has authored several IBM Redbooks® and Redpaper publications about DB2 for z/OS.

Thanks to the following people for their contributions to the previous version of this paper:

Jeff Josten
Distinguished Engineer
DB2 for z/OS Development

Harry Yudenfriend
IBM Fellow, Mainframe I/O Technology
IBM Systems &Technology Group, Power & z Systems

Now you can become a published author, too!

Here is an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4862-00 was created or updated on February 19, 2013.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. 1WLB Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks