White Paper

—

Performance Improvement and Cost Control for IBM Tailored Fit Pricing Customers

IBM®

IBM platform pricing

Based on customer requirements, IBM Z software pricing models for Z have evolved over the years. During 1970 through to 1999, a Full Capacity model was offered to provide customers the capability to leverage computing power of the entire infrastructure. As the Z hardware evolved, it offered better performance capabilities with each newer model, enabling customers to do more with less MSU consumption. This led to the Sub-Capacity Pricing model which allowed customers to manage their software cost based on the Rolling 4 Hour Average (R4HA) peak utilization.

This sub-capacity pricing metric was modelled in the late 1990's and assumed as a minimum an 80% utilization of Z hardware and helped align software licensing to less than full capacity. Both of these models served their intended purpose based on customer needs during their respective time frames. However, the world of IT is dramatically different now as compared to late 1990s. For example, average customer utilization is now typically much lower than 80%; a function of today's workloads being spikier with higher peaks and therefore lower utilization.

Throughout all of these pricing model changes; customers have always purchased the hardware for peak usage.

During mid-2010s, IBM noticed that customers were starting to experience increased challenges with sub-capacity planning irrespective of where they were in their digital transformation journey. Customers often fell into three broad categories:

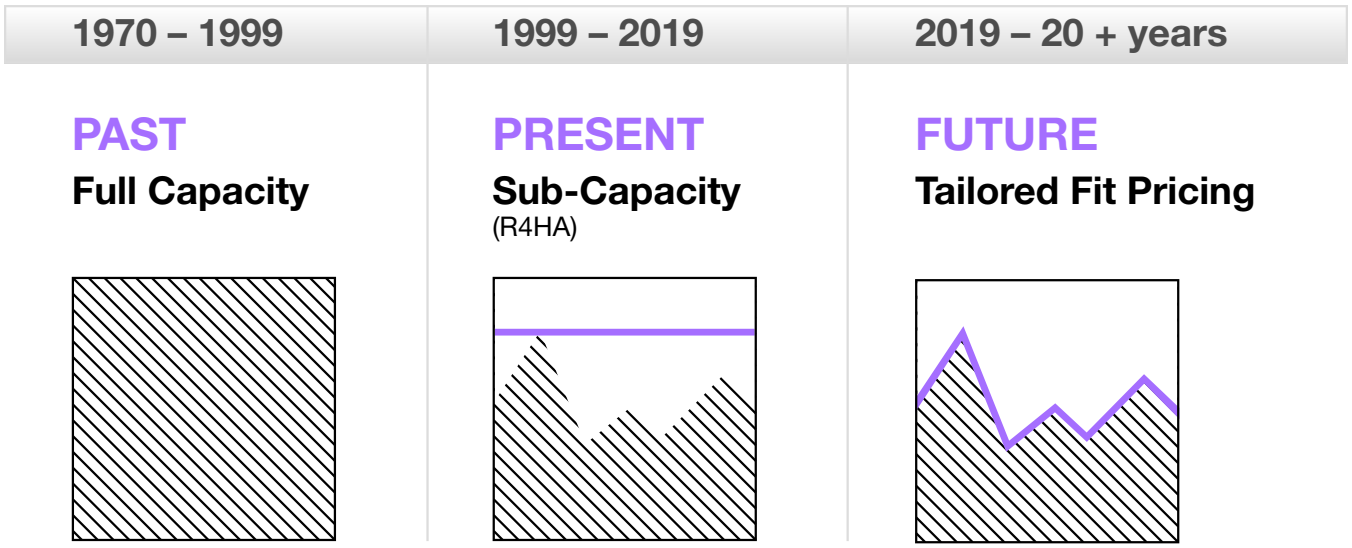
- Those investing in the Z platform to solve new business requirements caused by an evolution of the API Economy, DevOps practices and more as part of their transformation.

  These customers believed in the Z platform; however, they did run into challenges of meeting their SLAs while maintaining sub-capacity limitations.
- Those that recognized the Z platform to be the right choice for building a hybrid cloud architecture around but were not able to move forward due to concerns over the impact on billing that new and unpredictable resource demands would bring.
- Those that are not growing or growing just organically, but with an increasingly spikey workload profile (see below) producing a disproportionate impact on software billing through the R4HA.

Additionally, these categories of customers were often affected by seasonal and unexpected business/economic events, which drove even more unpredictability in the usage patterns. As

such, IBM felt a whole new cloud like pricing approach was required and it was clear that any new software pricing model would not include the sub-capacity (R4HA) metric and simplify quantification of all workload value in terms of system resource consumption.

And that was the driving for behind IBM's latest pricing option—IBM's Tailor Fit Pricing (TFP).



You've Made the Right Call with Tailored Fit Pricing

As a TFP user, you probably know by now that it is the best pricing model for mainframe environments experiencing growing workloads. TFP is the killer pricing solution for your class of enterprise—you are no longer impacted by those occasional sharp MSU usage spikes driving up the charges on your monthly bill.

With TFP's Enterprise Capacity Pricing, you get a very simple pricing solution that doesn't save money, but is predictable, with no overage charges.

With TFP's Enterprise Consumption Pricing, total consumption is reconciled at the end of the year—in effect, those months with the crazy spikes are normalized into a year-end cost calculation. The monthly cost is the year's total divided by 12, so those spikes no longer have such a detrimental effect on the monthly bill. Essentially, you pay for what you use at a per-MSU consumption rate. This one is the killer pricing solution.

So is TFP the Final Piece to the Puzzle?

TFP is a great first step. However, if your workloads are increasing, that usually means that your transaction processing loads are increasing, too – consuming your available capacity, and potentially driving up costs for the next year. Will TFP save you money? The answer to that is yes, it most likely will – any increased workloads will cost you much less than the going rate for increased MSU usage. That by itself has made TFP doubly worthwhile for any mainframe shop with growing workloads.

Is there anything you can do to augment the benefits of TFP?

The answer is a resounding YES:

- Db2 buffer management
- Db2v12 in-memory features
- IZTA – IBM's Mainframe High-Performance In-Memory Technology

But first, let's take a quick peek under the hood.

Moving Forward

First, most organizations running TFP have opted for the more flexible Enterprise Consumption Pricing option. Part of the calculation of your monthly bill under TFP is based on your current R4HA baseline. Before you signed on for TFP, you demonstrated a 12-month pre-TFP adoption baseline. That is what your current 12 monthly bills are based on, bringing you cost certainty.

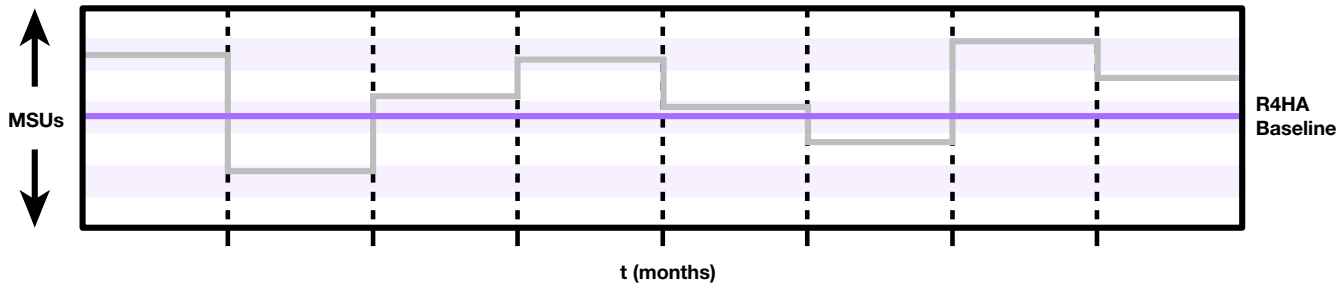
At the end of every year, there is a consumption reconciliation—usage above your annual entitlement will be charged at a "growth pricing" rate, typically at 50% of your going MSU rate.

That is where there is another opportunity for cost savings under the umbrella of TFP – if you can knock down that MSU-usage baseline this year, you may not require as much increase in MSU usage for next year, even in a growing workload environment. In some cases, you may be able to avoid growth costs completely.

The key to this gameplan is to reduce MSU usage by augmenting TFP; specifically, leveraging your mainframe memory in a few highly effective ways.

Figure 1

The TFP R4HA baseline



Augment TFP

Leverage your memory

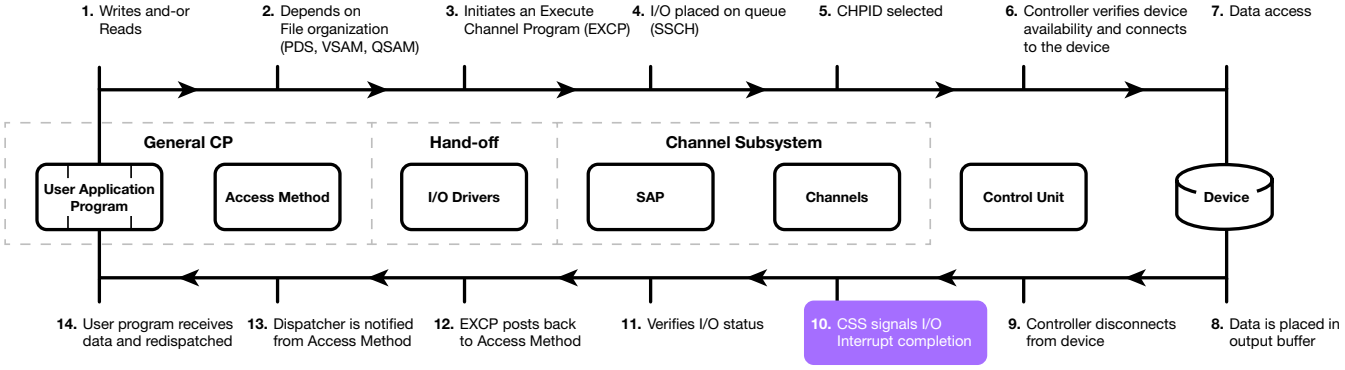
It's obvious that disk access is much slower than memory access, and that accessing data from memory is orders-of-magnitude faster than it is to read it from disk. Not only does disk access take much longer, it is also far more resource intensive (I/O, CPU, MSU). The performance benefit is self-explanatory—memory access is usually measured in microseconds, whereas disk access is measured in milliseconds. The truth is that avoiding I/O improves performance because there is a LOT going on "behind the scenes" when you request an I/O.

This diagram depicts 14 steps that occur when you request an I/O operation on a system Z machine. An I/O operation is expensive and several things have to happen in order to successfully process an I/O. With this, it is easy to see how processing I/O requests also drives up CPU usage to accomplish and track all of these steps.

For more information on this image, and on z/OS I/O operation, see the IBM publication, An I/O White Paper.

Figure 2

Disk access I/O steps



Benefits of memory usage.

CPU efficiency is improved with large memory when paging is avoided. As well, batch workload processing time can be reduced, as more memory allows increased parallelism for sorts and improves single threaded performance of complex queries. For OLTP workloads, large memory provides substantial latency reduction, which leads to significant response time reductions and increased transaction rates.

More information can be found in the IBM RedBook, Benefits of Configuring More Memory in the IBM z/OS Software Stack.

1. Db2 buffer management

After running mainframe computing for as many years as your organization has, you probably have a handle on Db2 buffer management by now. If not, there are many experts at IBM who can help you in this area, not to mention several high-profile consultants and third-party products available. You're probably better off going the consulting route vs using third-party products for obvious reasons, but getting the most out of buffer management will help to noticeably drive down your MSU usage. More information can be found in the IBM Redbook, Buffer Pool Monitoring and Tuning.

This is your first step in controlling your TFP baseline; there are two more highly-effective techniques that will complement buffer management.

2. Db2 v12

As you are probably aware, Db2 version 12 has introduced several performance-enhancing in-memory features that promise to make a big difference in your datacenter. Let's take a closer look at the four important in-memory features that were made available with Db2 12:

- Index Fast Traversal Blocks
- New Fast Insert Algorithm
- Contiguous Buffer Pools
- In-Memory Sort Processing

Fast Traverse Blocks

As Db2 table sizes grow larger and larger, so do the indexes grow in size, resulting in a greater CPU/time cost to randomly access index data. With Db2 12, IBM introduces Fast Traverse Blocks (FTBs), which are designed to make index lookups faster and cheaper, in terms of CPU/time. An FTB is an in-memory area that stores index pages, facilitating very fast index lookups for random index access for unique indexes.

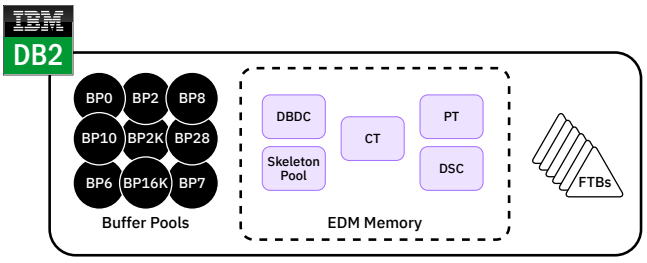
An FTB is a memory-optimized structure, containing the non-leaf pages of the index; for example, it can store the top two levels of a three-level index, the top three levels of a four-level index, etc.

As shown in Figure 3, Db2 requires memory for many structures—including buffer pools, which can be configured in varying sizes. There are also Environmental Descriptor Management (EDM) structures that cache operational details in memory as Db2 programs run. This includes DBDs for data structures, PTs, CTs, and skeleton structures for program structures, and the Dynamic Statement Cache. The FTB area resides outside of these structures, in a memory area above the bar managed by Db2. This means you will need additional real memory for Db2 to use FTBs.

The best candidates for using FTB are indexes that support heavy read access and indexes on tables with a random insert or delete pattern. They will not help cases where indexes suffer from frequent leaf page splits due to many inserts and updates. Straight from IBM's Db2 for z/OS Performance Topics Redbook, CPU usage can be improved by upwards of 20%.

Figure 3

FTB storage area in memory



Fast Insert Algorithm

Inserting data into a table where the data does not need to be clustered in any particular method is a special case for which Db2 offered little in terms of optimization. That changes with Db2 12—Insert Algorithm 2 (also known as Fast Insert Algorithm) improves performance for unclustered data. It applies only to universal table spaces that use MEMBER CLUSTER, and is now the default algorithm for this table space type. It is not applicable to other table space types.

An in-memory structure called an Insert Pipe is used to control INSERTs across data sharing members. Insert Algorithm 2 uses an asynchronous background system task—the Insert Pipe is filled asynchronously.

The best candidates for using Insert Algorithm 2 are workloads that are constrained by lock/latch contentions on the space map pages and data pages. Depending upon the number of indexes, the insert rate can be improved by upwards of 20%, with some improvements in elapsed time and CPU time as well.

Contiguous Buffer Pools

In Db2v12, IBM has dropped the term in-memory buffer pool in favor of a new term that encompasses a new set of in-memory features: contiguous buffer pool. In-memory buffers are now treated as a single block of storage; they do not require chain maintenance, and are ideal for code tables and frequently used smaller tables (stable data).

In IBM's tests, an 8% CPU time reduction and a 7% reduction in elapsed time was observed using contiguous buffer pools when compared to standard in-memory buffer pools.

In-Memory Sort Processing

Db2 12 continues to improve in-memory RDS sort processing over previous versions of Db2. It increases the number of cases when Db2 can avoid writing intermediate sort results to work files so the results can be fetched directly from memory instead of from work files. It also expands the maximum number of nodes in a sort tree, from 32,000 to 512,000 for non-parallel sorts or 128,000 for parallel sorts under child tasks. These enhancements might require more memory to be allocated to the thread for sort activities, but can result in a significant CPU reduction.

In IBM's testing, in-memory sorts that previously required work files for sort and merge processing showed a 75% reduction in CPU time. A 50% reduction in CPU time was seen for queries that benefitted from increasing the SRTPOOL size ZPARM in Db2 12. The largest improvements were for queries that can now complete their group by or distinct processing in one sort pass because of the larger sort tree size.

3. Mainframe high-performance in-memory technology

IZTA is enhanced high-performance in-memory technology that is used to augment your mainframe Db2 databases, and to further improve your system performance and throughput capacity.

What is high-performance in-memory technology? It is an in-memory accelerator for mainframe applications, providing dramatic decreases in elapsed time and CPU requirements. The idea is to reduce the code-path for read-only data accesses. That is accomplished by first identifying your applications' most-often accessed data, and copying it into high-performance in-memory tables, where it can be accessed 100 times faster. This amounts to about 5% of your total data, and in most cases, much less than that. The applications will access this data from the in-memory tables, and access all other data from Db2, as before. The access method used is a simple, tight API.

How high-performance in-memory technology works

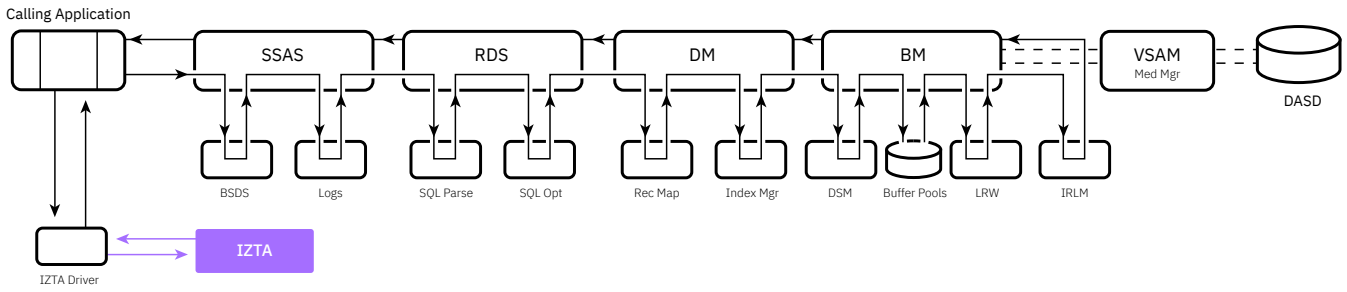
Figure 4 (top) shows what is involved in an I/O operation, or any fetch of data from a database: the code path that the data request and fetch takes—from the user application program to the database and back. The bottom shows the code path from the user application program to the in-memory table and back.

A typical DBMS code path uses between 10,000 and 100,000 machine cycles, while a typical access using the shorter code path of IZTA in-memory technology uses between 400 and 500 machine cycles.

Candidates for this technology are typically read-only tables, or highly accessed tables experiencing a comparatively very small number of updates.

Figure 4

The shorter code path



Results using high-performance in-memory technology

Depending upon what the current state of memory usage is within your workshop, adopting high-performance in-memory technology can result in truly astounding outcomes. CPU usage can be reduced dramatically, while elapsed batch time can be cut equally so.

Figure 5

Reduced CPU usage

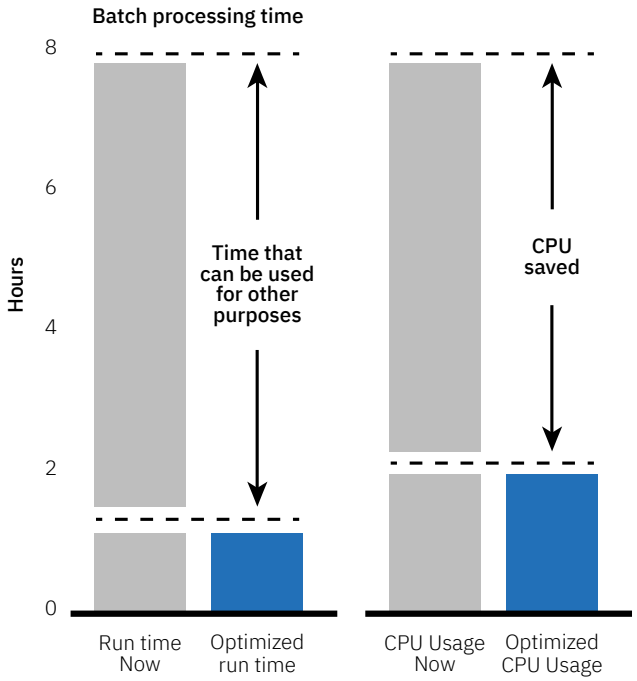


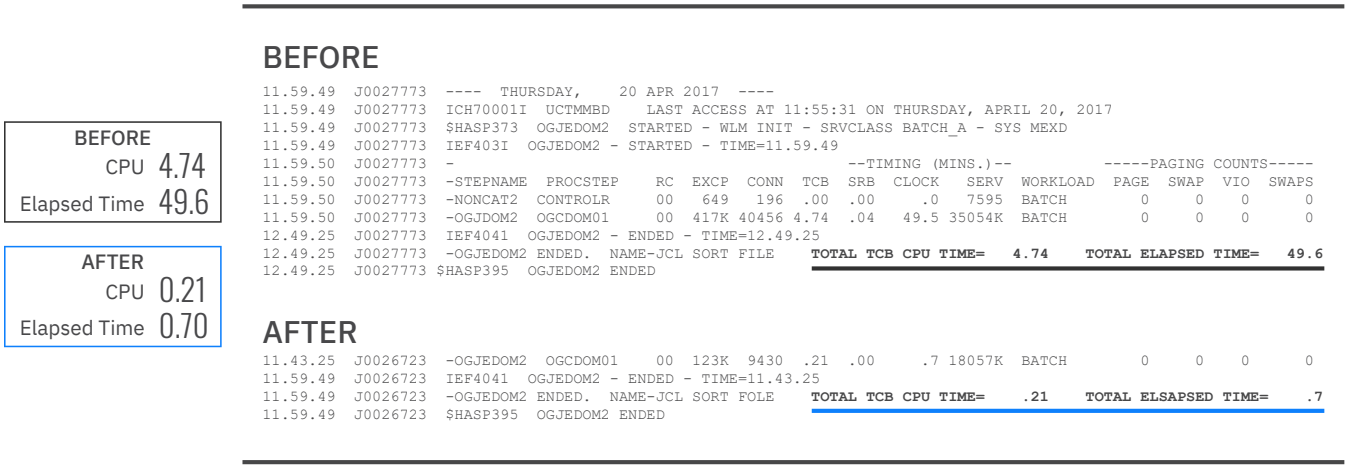
Figure 5 (top) shows that the time needed to run batch jobs can be markedly reduced— this can solve serious problems within the batch windows of mainframe shops struggling to deal with growing workloads while being pressured to reduce batch window footprints. Figure 5 (bottom) also shows that CPU resource usage can be dramatically cut for mainframe online transaction processing systems—this can help to solve the growing need to increase the amount of security measures required to take place in real time during the transaction, and to minimize the amount of security measures that must wait until reconciliation during the batch process.

Figure 6 shows an example of how one customer optimized a single table in their mainframe transaction processing system—this is from their SDSF output showing their job status. Their system used a Db2 table to house a small amount of data that was being accessed millions of times per hour, and it consumed 4.74 CPU seconds every time it ran, with an elapsed time of just under 50.

After copying the table into a high-performance in-memory table, the CPU consumption fell to 0.21 CPU seconds, and elapsed time dropped through the floor to 0.7.

Figure 6

One optimized table



Conclusions

TFP is the “promised land” for IBM mainframe customers who have lived through years of mainframe pricing techniques that, while workable, often required much work to maximize benefits, and to balance performance versus cost. With TFP, there is cost certainty, as well as cost savings, especially for organizations experiencing growing mainframe workloads.

TFP customers can realize further cost-savings along with significant performance benefits by augmenting TFP with three proven IBM mainframe in-memory solutions.

While TFP does make a difference, augmenting it with buffer management best practices, implementing powerful Db2 in-memory features, as well as mainframe in-memory table acceleration, gives customers compounded performance and cost-saving benefits.

If you run a mainframe datacenter with growing workloads, and you care about improving performance and throughput, these augmentation solutions should be at the top of your to-do list—it is the responsible thing to do for the business.

Solution	Performance improvement	Cost Certainty	Cost Savings
TFP	n/a	YES	YES for next year’s new workload needs
Buffer management	YES up to 90% less I/O under optimum conditions	n/a	YES mostly I/O savings
Db2v12 in-memory features	YES 50% or more under optimum conditions	n/a	YES I/O related plus efficiency improvements
IZTA	YES 99% less I/O, 80x faster run times, under optimum conditions	n/a	YES I/O, efficiency and code-path related

The Next Step

Please contact us for more information—we can send you more information, or alternatively, you can speak to one of IBM’s Professional Services staff to discuss your specific challenges. Use the following contact information to get started:

Phone: 1-818-378-9030
Email: lpdekans@us.ibm.com
Mailing address: Luke P De Kansky, Los Angeles, IBM