DB2 for z/OS Version 9
Automatic Buffer Pool Size Management

IBM documentation for DB2 V9 has a paragraph about *Automatically Managing Buffer Pool Size*. So let's look at the statement, and determine if the average installation will derive a benefit from this approach.

Automatic buffer pool size management: You can reduce the amount of time that you spend monitoring and adjusting DB2 buffer pools by enabling DB2's automatic buffer pool size management feature. The AUTOSIZE(YES) option of the ALTER BUFFERPOOL command allows you to activate dynamic buffer pool size adjustments that are based on real time workload monitoring. For example, a noncritical DB2 subsystem can automatically reduce its buffer pool size. By doing so, it frees the storage so that it can be used by a mission-critical subsystem on the same LPAR, if important transactions fail to meet performance goals. When you enable automatic buffer pool management, DB2 reports the *buffer pool size and hit ratio for random reads* to the z/OS Workload Manager (WLM) component. DB2 also automatically **increases or decreases** buffer pool size, as appropriate, by **up to 25% of the originally allocated size**. **Automatic buffer pool management does not completely replace existing tools to configure, monitor, and tune buffer pool size.** However, when you have initially sized your buffer pools, DB2 and WLM can "fine tune" the buffer pool size, based on long term trends and steady state growth. The DISPLAY BUFFERPOOL output includes an AUTOSIZE attribute. You can enable or disable automatic buffer pool management at the individual buffer pool level. Automatic buffer pool management is off by default.

Generally, we're not concerned with reducing the size of a noncritical system, we've already done that if memory is constrained, so this is a non-issue. The important reason for pool tuning is getting better performance by reducing IOs, and IO elapsed times. Let's pick apart the above statement and place the pieces into an understandable performance tuning perspective. On the surface - the announcement, if you don't both understand pool tuning, and read it thoroughly, implies that you won't need to tune your pool sizes, that DB2 will do it for you.

There are several parts of the statement to analyze:
a. Initial sizing of your pools
b. Hit ratio for random reads
c. Fine tuning the pools based on long term trends
d. A 25% maximum increase in pool size

Initial sizing and Tuning of your pools

The assumption that the pools at any installation are properly configured, sized, and tuned is a pre-requisite for this approach. The proven technique for tuning pools is grouping objects into multiple pools, based upon both access type and working set size (no relevance to catalog statistics). Random pools should have little or no sequential scan activity. The access and usage of random objects is a crucial piece of the performance perspective. There are three disparate classes of random objects we may need to address, and this applies to indexes as well as data.

The first group is ***small to medium working set*** objects (there is no rule of thumb for small, medium, large, etc, every system and application is different), the next group is ***large working set objects***, and the last and most difficult is ***very large, and very random objects***. This last group is impossible to tune, and almost always cause an IO no matter how many buffers you provide. The very large and very random objects must be isolated into their own pool, and rarely benefit from very large numbers of buffers.
The key to effective tuning is giving memory to the pools and objects that will have a high re-reference rate of pages in the pool, and avoid causing IOs.

Hit ratio for random reads
While the hit ratio has been used for decades to track pool performance, it is not a measurable metric. See other published articles from Responsive Systems (The DB2 Buffer Pool Hit Ratio is Dead) for a detailed explanation.

Fine tuning the pools based on long term trends
Pool performance is something that should be monitored and tracked over time, as part of the performance analyst's job. The statement in the manual does not provide any indication of the duration of measurement for a long term trend - is this minutes, hours, days or weeks. If performance is degrading, the performance analyst should know this, and it will not be detectable from *small* variations of a hit ratio. Depending upon pool activity, a variation of 1% for a hit ratio can be many or few IOs per second.

25% increase in pool size
When a pool has 20,000 buffers, the increase can be up to 5,000 buffers. Since there are pools in use at some companies today that use 500,000 buffers, the increase may be substantial. However, the important thing to understand, is that a memory increase does not always provide a performance increase. It may simply waste memory and not provide any benefit. This is not a predictive tuning approach, it's a ***throw memory at the pool and hope*** approach. The current documentation does not provide any indication of the increase increments - only a maximum amount. Using the two performance graphs below as examples, Figure 1 illustrates the change in hit ratio, and Figure 2 illustrates the change in the IO rate as pool size is increased.

First range increase
Let's look at the change between points three and four in the graphs. Point three represents 20,000 buffers, and point four represents 27,000 buffers. An increase of 35%. **Figure 1** shows that the Hit Ratio increases .8%, while **Figure 2** shows a decrease of 11.2 IOs per second. This is a good payback for the 28 megabytes of memory, if it is available.

Second range increase
Using larger pool sizes, increasing the buffers from 75,000 to 95,000, an increase of 26%, improves the Hit Ratio by .6%, while saving only 2 IOs per second. Compared to the first range increase, this is 80 megabytes compared to 28 megabytes - 52 megabytes more. The hit ratio increased 25% less, and the IO rate decreased by *80% less than the first range increase example.*

Improvements in pool performance are rarely linear, and usually show decreased benefit as the pool size increases. These illustrations clearly show a *memory to performance benefit is more accurately represented by the IO rate than a hit ratio.*

Summary
While there may be some performance potential for this approach, it's unlikely it will benefit any well tuned environment. Without the ability to predict a benefit from increasing a pool size, the greatest probability for this facility is wasting memory. Especially since a hit ratio is not a viable performance metric with an adequate level of granularity, it is unlikely that this facility will provide any benefit. If you have lots of memory to waste, it won't hurt you, and WLM will monitor overall memory usage on the machine before making increases.

This analysis was created in September 2006, and **modified Nov 2009** after IBM's presentations at the IOD Conference in Las Vegas. Over these years IBM has not provided additional technical information about this proposed facility. There are no implementation examples, there are no measurements of improved performance - either at client sites, or the DB2 development labs. If IBM can't dummy up some performance numbers to justify this approach - this lack should speak for itself. Please focus upon the red highlighted text on the first page - that whole section is directly from the IBM DB2 manuals.

Are you willing to be IBM's Guinea pig with your production systems?

Therefore, it seems that the primary intent of IBM's presentation of this subject is to prevent corporations from considering other software products in the marketplace, and wait for IBM's wave of a magic wand.

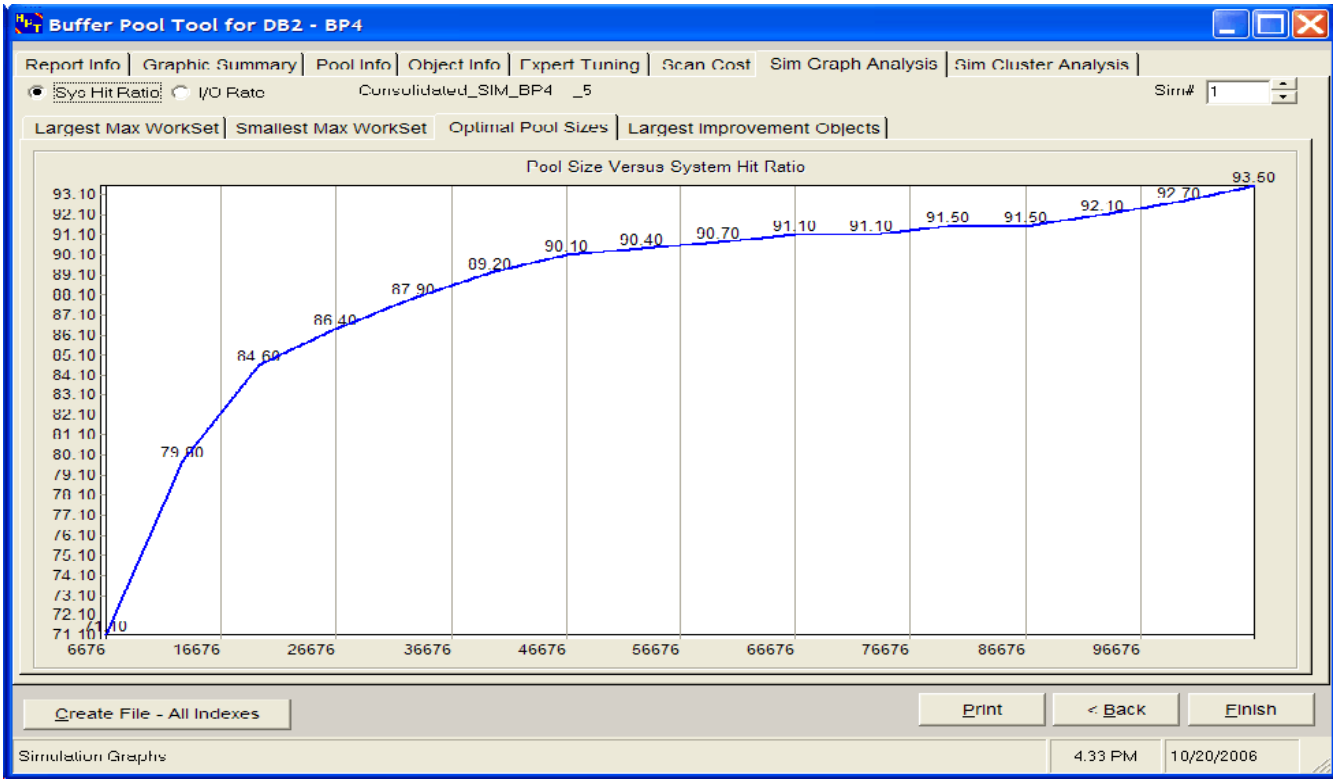
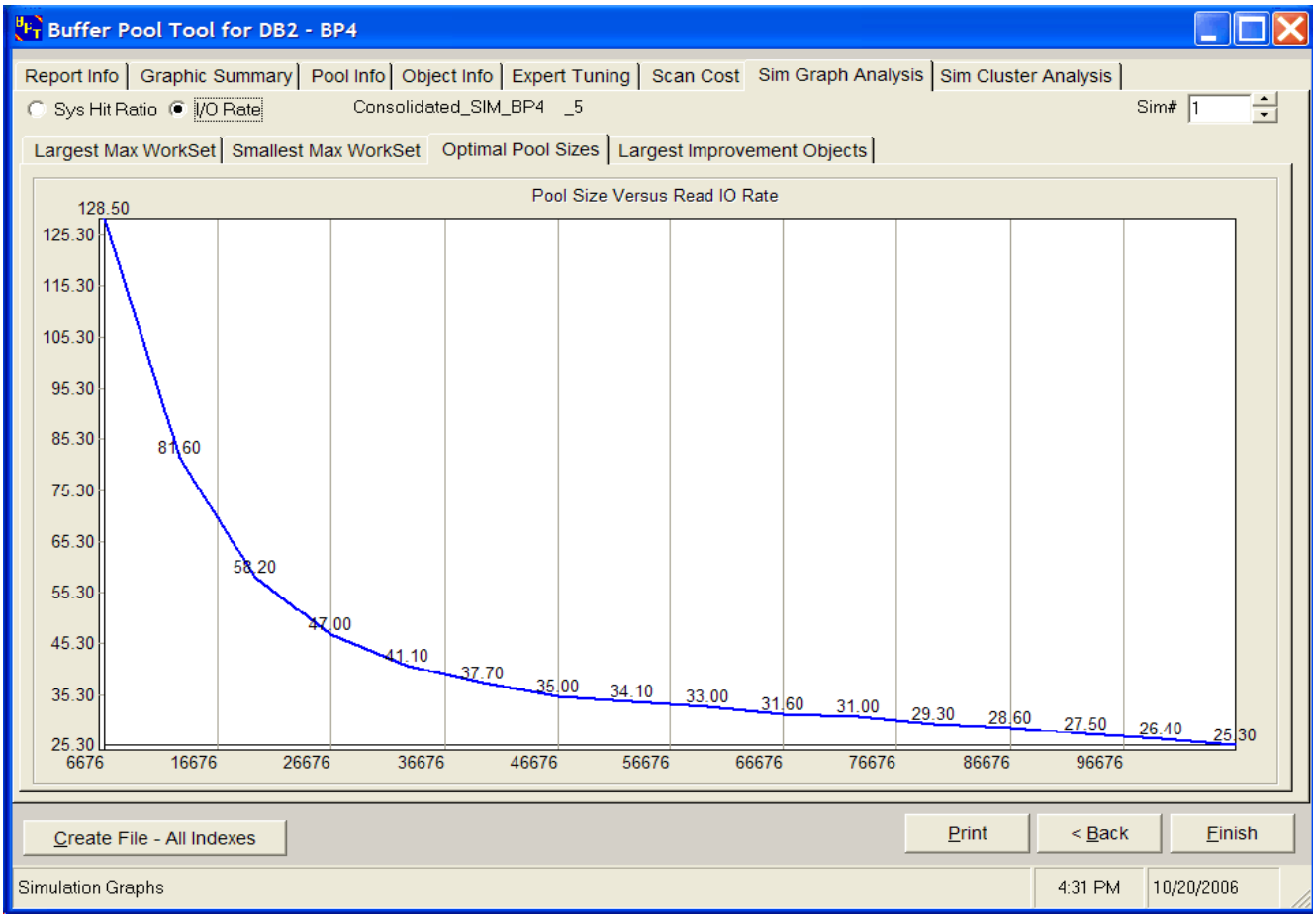


Figure 1 Shows the relatively small increases of Hit Ratio at different pool sizes

Figure 2 Shows the difference in the IO rate at different pool sizes, that can be converted to CPU cost



savings, and measured application elapsed time reductions.