WHITEPAPER

INGRES

# DISK CONFIGURATION TIPS FOR INGRES

BY CHIP NICKOLETT, INGRES CORPORATION

## TABLE OF CONTENTS:



## ABOUT THE AUTHOR


Chip Nickolett is the Director of Consulting Services for the Americas at Ingres. He has over 23 years of business and IT experience, and has been using Ingres RDBMS technology for over 20 years. He is a recognized expert in performance tuning and management of mission-critical Ingres installations.

# DISK CONFIGURATION TIPS FOR INGRES

## PREFACE

So, you have the fastest multi-socket quad-processor configuration with several gigabytes of RAM, yet your Ingres performance is not exactly what you expected. You have ruled out schema problems (incorrect storage structures and/or missing indexes) and statistics as a potential cause. You have validated your Ingres configuration (especially the DMF cache and locking parameters) and environment variables (seen with the "ingprenv" command). What could possibly be the root cause?

For the majority of production installations (probably 90% or more based on my experience) the need for absolute high-performance is not essential. I can just imagine your thoughts right now, "What, performance is not essential?" Let me be clear. Performance is always important, but an "absolute high-performance" scenario where a 10 millisecond query is "marginally acceptable" is the focus of this white paper. These are environments where a 98%+ cache hit ratio is important to minimize disk I/O, and disk performance is optimized to get 3 ms - 6 ms average response times.

The principles and recommendations provided hold true in any environment, but are not always needed for typical Ingres installations using managed disk devices (such as a SAN or NAS). In those environments it is often difficult to separate physical volumes from logical volumes, and it may not be possible to specify the highest-performance disk drives or controllers. Ingres will operate in a perfectly acceptable manner in those environments, and will deliver great performance for all but the most demanding environments. In those cases this becomes a business decision (based on availability, response time, cost, capacity, etc.) as much as a technical decision. So, please read on and decide for yourself: How much is enough?

## OVERVIEW

Purchasing and configuring new disks for Ingres is an important but often confusing task. How many physical drives do we need? What size and type of drives should we purchase? Should we use RAID? This article should provide useful information to help you answer these questions. The topics discussed are generic in nature and apply to any hardware platform.

This article also discusses the use of multi-location tables and indexes within Ingres, as well as more generic Ingres disk configuration suggestions. The Database Administrators Guide is a good reference manual for this information and should certainly be studied prior to installation. The configuration suggestions described in this document are relevant for the larger, higher-performance installations and should be viewed as an extension to the recommendations made in the DBA Guide.

## HOW MANY DISKS DO I NEED?

Let's start with a few basics. Ingres allows tables and indexes to span multiple locations. Ingres writes data to these locations in a round-robin manner, writing 16 pages to a location before moving to the next location. It should be noted that if a location fills, Ingres will not simply skip to the next location, it will generate an error and abort the transaction.

The Ingres tuple identifier (TID) is bitmapped to track row number within a page number within a table. By knowing the page number and the number of locations that the table/index spans, Ingres knows exactly which location the data page resides on. A row of data needs to reside on a single data page (i.e., it cannot span pages), so this becomes a limiting factor for data utilization within a data page.

For example, if a row has a size of 1,005 bytes there will only be one row of data on every 2K data page. By estimating the number and size of rows for each table and the planned growth and archival strategy, it is relatively easy to determine the baseline size of the database. This size does not take into account index overhead or fillfactors (which often increase that size by 20%-25%).

Next, you will want to take into account disk space required for sort / work areas (typically 3 or more times the size of the largest table, often much larger depending on the anticipated amount of sorting and modification activity). For example, sorting multiple tables at once or taking advantage of concurrent index creation causes more sort space to be used.

It is not unreasonable to have 2 GB or more dedicated for sort / work. It is also recommended to leave at least as much free space on a location as the largest piece of a table for that location. That is very important with respect to table modification since by default both the original file (for that piece of the table) and the new file (for them modified table) will reside on the same location until the transaction ends.

So, with this knowledge it is easy to order or allocate disks. Simply divide the total database size by the size of the largest drives you can find, right? Not quite. In addition to page overhead there is storage structure overhead, and there are also system catalogs. You would also want to factor in additional space for growth over a reasonable period of time (e.g., the next year or two). But, there is more... Performance of the system will be affected by the decisions made at this stage of planning.

It is possible to take a 400+ GB physical disk drive and place everything on it. That drive could even be partitioned to appear as several separate filesystems (which is often worse). There are two problems with this approach. The first is that the very large disk drives are typically slow (10,000 rpm). The second is that, due to their size, performance typically suffers because the disk heads are thrashing around to get data, increasing the I/O request queue and associated I/O wait times for that device.

A better approach is to select several smaller (70 GB to 150 GB), faster (15,000 rpm) disk drives. This can be a difficult "sell" in an environment using a SAN where space is at a premium, but benchmarks that I've performed time and time again (even on large SANs with robust caching controllers) demonstrate that this configuration leads to better, more consistent performance under a heavy production load. Just as important is the type of disk controller used (Fibre Channel, Serial Attached SCSI [sas], and SATA preferred), and the amount of on-board cache on the controller. Since the disk subsystem is the slowest component in any computer system it pays to design performance into your disk configuration.

*Note: Versions of Ingres prior to 2.5 do not provide large file support for files > 2 GB in size. Since a table in Ingres references one or more underlying operating system files, this means that a table cannot exceed 2 GB in size without residing on multiple locations.*

## SHOULD I USE RAID?

Now, what about RAID (redundant array of inexpensive disks)? RAID 5 (with its hot swapping capabilities) is typically the worst choice for a write intensive environment (a category that most databases fall into). Performance degradations of 50%-100% are not uncommon. This has nothing to do with Ingres and everything to do with hardware.

More and more high-end (read "very expensive") storage arrays have very large caching controllers which can help minimize the performance impact of RAID 5. It is important to find out where those controllers become saturated (thus degrading performance) and see if the threshold is lower than what would be expected in a true production environment.

It should also be noted that RAID 5 is not foolproof. We have seen several instances of failures that were due to either a controller failure, or to failure of more than one disk at a time (sometimes caused by someone hot-swapping the wrong disk drive).

RAID 1 (mirroring) is absolutely recommended (as is journaling, as an aside). The loss of a single data drive will leave the database inconsistent. The hardware cost of having a complete set of mirror disks is generally less than the cost of unscheduled downtime to many businesses.

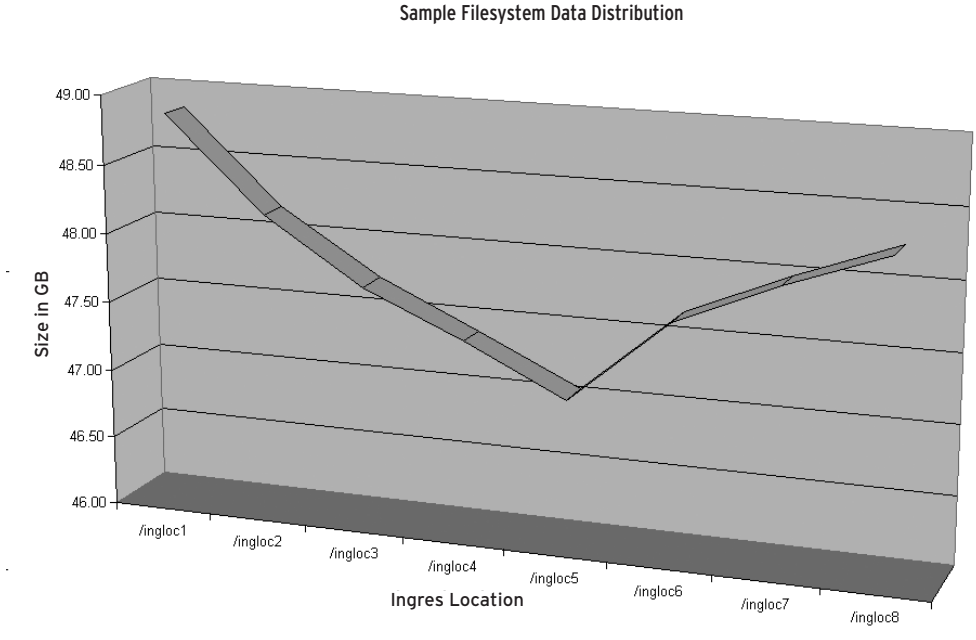
What about striping? Our recommendation for RAID 0 (striping, not true RAID) is not to use it. While RAID 0 performs better than a single device in streamed write operations, it has been my experience (supported by benchmarks) that "Ingres striping" (i.e., multi-location tables) performs much better than tables on a single device, and performs marginally better (typically 1%-3%) than single location or multi-location tables on striped disks. Our recommendation is to distribute all tables and all indexes over all available data locations (this works well in most environments and is described further in the following section).

## INGRES CONFIGURATION RECOMMENDATIONS

As mentioned earlier, with multi-location tables (and indexes) the data is distributed among all locations in a round-robin manner, providing an approximately even data distribution among those locations for a single table. But, when there are hundreds of tables the slight variations in size are amplified and the initial locations will have a higher concentration of data than the locations at the end of the location list. This increases the I/O load on the heavier populated locations and causes some degree of performance degradation.

There are two different approaches to address the data distribution concern. The first is to alternate location usage (e.g., different starting locations but the same sequence, or random location specification). I personally prefer to have things standardized to improve the maintainability of the system and simplify automation of routine tasks. Therefore, I would not use this approach.

The second approach is to consistently apply the location names for all tables in one direction (for example: location1, location2, location3, location4) and consistently apply the same locations for indexes, but in reverse order (for example: location4, location3, location2, location1). This helps provide a level of data distribution that is more even while maintaining consistency and ease of management. This approach has been provide to yield slight benefits (typically 1% to 2%) over other configurations in disk intensive benchmarks on large databases (.5 TB+) in high concurrency (1,000+ simulated user) scenarios. Below is a graph that demonstrates the effect of this configuration.



Below is a configuration that I have standardized over the years, with size recommendations that are reasonable for medium to large installations. These are only rules of thumb, but provide an example of a standard that has been proven in many large production environments. Please note that the physical separation of filesystems for the Ingres locations among the available hard disk drives should still be consistent with the recommendations provided in the DBA Guide.

- **/ingsys** - This is your II_SYSTEM area that contains Ingres binaries, utilities, configuration files, and log files. Typically this is sized between 1 GB and 4 GB.

- **/ingckp** - Checkpoint directory (I like to size for a minimum of 3-5 checkpoints, which can be compressed as part of the checkpointing process).
*Note: From an Installation perspective, I prefer to have as many checkpoint locations as there are data locations. This allows for simple mapping for parallel checkpoints.*

- **/ingjnl** - This is for journal files and needs to support the minimum number of checkpoints being saved (these can be compressed daily depending on the volume of journal files and the retention policy). This is usually 8 GB or larger.

- **/ingdmp** - This is the dump location for online checkpoints. Usually 1-2 GB is sufficient.

- **/inglog** - Transaction log file (cooked, not raw). Usually 2 GB is sufficient.

- **/ingdata** - This is your data location. If only one location is configured then I define ii_database to point to this location (otherwise I will point it to /ingsys and not use it - an approach that makes it easy to find non-production tables). If multiple data locations are used then all tables and indexes should be configured for all locations.
*Note: When configuring more than one data location I will use the naming convention: /ingdata1, /ingdata2, ingdata3, etc.*

- **/ingwork** - This is the sort / work area. Sizing is dependent on the anticipated volume and largest anticipated table size, but typically sized between 2-8 GB per location, with the possibility of multiple locations being configured.
*Note: When configuring more than one sort/work location I will use the naming convention: /ingwork1, /ingwork2, etc.*

## FREE TOOL DOWNLOAD

The source code is intentionally very simple. The goal was to demonstrate that 99%+ of the work being performed (which can be proven using a profiling tool) is actual OS disk write activity. The write size is 2K since many RDBMS products perform 2K writes. This tool accurately demonstrates differences in write performance between various devices.

Attached are two archives (Zip and "gzipped" (GNU Zip) tar) that contain the source code for our tool (WriteBench), working binaries (executables) for several platforms, a readme file (Unix ASCII format - will look strange using "notepad" on Windows, but looks fine when opened with Word). You have the choice of using a pre-compiled version, or compiling this for yourself.

The tool is very simple to use (usage information below and in the archives). It takes 3 command line parameters (first 2 required, 3rd is optional).

1. The first is for the name of the test file to be generated. **PLEASE MAKE SURE THAT THIS IS NOT THE NAME OF A FILE THAT YOU WANT TO SAVE, SINCE THE PROGRAM WILL OVERWRITE THAT FILE!**

2. The second is the number of 2K lines to be created. For example, to create a 100 MB file you would enter "50000". We recommend test file sizes of between 100 MB and 1 GB. Smaller files generally run too fast to provide useful data for comparison. The maximum file size on most platforms is 2 GB, so the test file needs to be smaller than that.

3. The third parameter is the number of tests (iterations) to perform. By default it runs 5. We recommend using a value of 10-25. The output of the program is the results for each test, and the average for all tests. In general we will look at the min, max, and average values. The larger the sample of tests performed (within reason), the better the quality of the data.

*Please note that the source code for this simple disk benchmark is GPL and included in the files linked below. The program can easily be modified to test larger page sizes that are supported by Ingres.*

**UNIX tar format: http://community.ingres.com/forums/home.php**

**Windows zip format: http://community.ingres.com/forums/home.php**

## SUMMARY

Hopefully this article has provided enough information to help you determine the ideal disk configuration for your environment. The proper disk configuration and table layout can significantly improve performance by increasing the overall I/O bandwidth of the system. Keep in mind that the I/O subsystem is the slowest link in the hardware chain. Anything that can be done to improve I/O performance, no matter how small, is important (especially with fast processors).

## NOTES

## NOTES

## About Ingres Corporation

Ingres Corporation is a leading provider of open source database management software. Built on over 25 years of technology investment, Ingres is a leader in software and service innovation, providing the enterprise with proven reliability combined with the value and flexibility of open source. The company's partnerships with leading open source providers further enhance the Ingres value proposition. Ingres has major development, sales and support centers throughout the world, supporting thousands of customers in the United States and internationally.

**INGRES CORPORATION** : 500 ARGUELLO STREET : SUITE 200 : REDWOOD CITY, CALIFORNIA 94063
PHONE 650.587.5500 : FAX 650.587.5550 : www.ingres.com : For more information, contact info@ingres.com

INGRES



WP-301A