SAP IQ 16.x Sizing Guide

A comprehensive sizing guidline for SAP IQ

PUBLIC

Mark Mumy, Chief Architect, SAP

THE BEST RUN SAP®

TABLE OF CONTENTS

INTRODUCTION

Purpose

This document will attempt to highlight the main areas of concern for sizing a IQ environment. Sizing of SAP IQ is generally confined to CPU, memory, and storage.

It should also be noted that this document is a work in progress and most points contained in it come from real world experience. Should this document be followed it is at the sole responsibility and discretion of the reader and implementation team.

Terminology

It should be noted that the term CPU is used throughout this document. In the SAP IQ world, a CPU, processor, and core all relate to the same physical hardware: the processor core that performs the work for the computer. Some systems use single core processors while others have multi-core (2, 4, 8, and beyond) processors. For the purposes of this document all references to CPUs and processors refer to the actual processor core.

For terminology purposes, the term hyperthreading is used to describe any type of processor core threading model whether it is IBM's SMT, Oracle CoolThreads, or even Intel's Hyperthreading. It is meant as a generic term that represents ways to give the OS, and IQ, the appearance of more physical cores than there really are.

Hyperthreaded systems haven't proven themselves to provide a significant boost to IQ performance and as such, hyperthreading cannot be taken into account for the guidelines and algorithms in this document. In fact, for most systems, it is strongly recommended to disable any type of hyperthreading capability.

As such, a core that is hyperthreaded should be treated as a single core/CPU. To effect this change in IQ, use the –iqnumbercpus parameter to tell the optimizer the correct number of cores on the machine. A 4 core machine in which each core is hyperthreaded 2 ways would show that there are 8 cores/CPUs to the operating system. The –iqnumbercpus parameter would be set to 4 so that the optimizer does not overly tax the system with work.

The caveat to hyperthreading and setting of –iqnumbercpus is on the IBM Power platforms. Most implementations today on IBM hardware follow the pattern of enabling SMT (SMT2, SMT4, and SMT8), and then letting IQ default to seeing all threads as available cores for processing.

A key point should be made when setting –iqnumbercpus to something other than the number of cores. While it can improve performance, it can also oversaturate hardware and cause performance degradation. Every platform and every application behaves differently, so structured testing is certainly called for. It is even possible that two nodes in the same IQ multiplex have different settings because they have different workloads.

Many times, it is not enough to simply reduce the core count that IQ has access to via –iqnumbercpus. If the system has more than 128 threads, it is recommended that the hyperthreading be disabled at the host or OS level so as to avoid oversaturating the OS. This is especially true of Oracle Solaris Cool Threads. The nature of their threading model is known to cause performance issues on larger systems.

Once all threading has been disabled, performance metrics should be gathered and the system fully tested so that the impact of enabling the threading model can be fully understand and the performance degradation or gain measured properly. Starting with a fully threaded system is not advised and should certainly be avoided if SAP IQ does not appear to be functioning as expected or seen on smaller systems.

WHAT'S NEW TO IQ 16?

For those migrating from older versions of SAP IQ (15.x or earlier), there was a change to how memory is used as well as a new default storage (index) mechanism.

IQ Large Memory

A new, fixed size memory section ("cache") was added to SAP IQ 16. This new area is called IQ Large Memory (Large Memory Allocator or LMA). This is also referred to as the Large Memory Allocator or LMA. This new memory area was the net result of an overall memory management enhancement in SAP IQ 16.

The large memory section (-iqlm option) offers several advantages over IQ 15 memory use. It introduces the concept of flexible and inflexible memory and provides a central location to manage both as a shared resource. The LMA allows users of flexible memory for hash and hash-range table loading to request a fair share of available LMA memory and to dynamically adjust their share based upon system workload. The LMA also provides a central location for all large n-bit metadata memory allocations. This allows LMA to function as a cache for the n-bit metadata, such that memory objects may later be retrieved from memory for reuse if the memory has not been reclaimed for use by another object

N-bit Indexes

The n-bit encoding that was introduced (and is default) in IQ 16 is superior to the tokenized FP encoding in IQ 15. The primary improvements are:

1. Values on n-it pages are encoded as 1 to 31 bits, based upon the maximum token on the page. In IQ 15 the optimized FP used 8 bit, 16 bit or 24 bit encodings. Therefore, IQ 16 n-bit columns will have better in-memory and on-disk compression.
2. N-bit pages use a per-page encoding. There is no column-level "rollover" from one encoding to another. E.g., if a column consists of 9 bit pages and new values are added such that a 10 bit encoding is needed, only the new pages are written in the 10 bit format.
3. N-bit columns use less memory for metadata at query time and load time. The n-bit dictionaries and count tables are allocated from the large memory section (LMA) and are of variable size as needed. The IQ 15 metadata objects use fixed size pages in the IQ main or temp cache.

There are a few primary guidelines to follow in IQ 16 regarding n-bit columns and the Large Memory Allocator sizing. Though there may be exceptions to the rules, the guidelines are the following:

1. Do NOT use the IQ UNIQUE() clause
2. Do NOT try to calculate how much LMA memory will be used for loads or queries.
3. A setting of IQ UNIQUE(0) will disable the optimized FP for that column

SIZING OVERVIEW

Below is a quick glance at the areas covered in more detail in this document. This should provide a simplified, quick reference to most of the formulas discussed in detail later. The sizing recommendations in this, and future, sections, are general guidelines for a properly configured system. The values may need to be raised or lowered depending on cost, business needs, and required performance levels.

General Hardware Setup

What type of server hardware	Windows (Intel/AMD), Linux (Intel/AMD/IBM), AIX, HP-UX (Itanium), Solaris (SPARC/x64)
What type of storage hardware	SAP IQ supports a SAN, direct attached storage, and NFS/GPFS (Linux only)
RAID level	RAID 1 or RAID 1+0 offer the best performance while RAID 5 offers the best cost for IQ storage
Volume management	IQ has no requirement for this, though it can be used

Cluster and HA software	None needed, but can be used to extend high availability and disaster recovery

IQ Storage Sizes

DW size (data and index)	60-80% smaller than the input data
Raw storage size	Will vary depending on the RAID level used
Storage block size	As large as possible. General rule of thumb is to match or double the IQ page size.

General Database Configuration

Database page size	64K-512K (default 128K)
Database block size	Default of 1/16th the page size (4K-32K). This need not be changed
Database case sensitivity	Recommend using “Case Respect” for optimal performance
Swap space	Minimum of 1GB. The amount of swap should be sufficient to keep the system running should memory be over allocated.
General memory guideline	12-16 GB per CPU core

CPUs

CPUs for queries	0.1-2 CPUs per query
CPUs for loads	A core can process and load 20 MB of raw, filesystem data per second. Size cores based on business load requirements.

Memory

Total RAM per host	12-16 GB per core
Total IQ memory for all operations	Total RAM less 10-20% for the OS
IQ Large Memory, IQ main cache, IQ temporary cache	Each parameter (-iqlm, -iqmc, -iqtc) should be set to 1/3rd of available RAM (roughly 30% of the host RAM)
RLV Store (new)	Will vary depending on the number of tables flagged to use the RLV. If used, the amount of available RAM for main, temp, and large memory caches needs to be reduced.
Main cache during loads	5-10 pages per index and column
Temporary cache during loads	Each HG (max): (8 + sizeof(datatype)) * numberRowsBeingLoaded Each WD (max): (8 + sizeof(datatype)) * numberRowsBeingLoaded * numberTokens (rounded to next highest page boundary)
Bitmap memory during loads	8,192 bytes per distinct value loaded into each LF, HNG, DTTM, DATE, TIME, and CMP index
Catalog Cache (-c or -cl/-ch)	4-16x the size of the catalog (.db) file

Storage and I/O

Required I/O bandwidth per CPU core	50-500 MB/sec

Total bandwidth per server	#cores * 50-100 MB/s (increase for stricter service level agreements)
Minimum number of physical disk drives per core	Not critical so long as the storage can support the necessary bandwidth
IQ shared temporary store disk drives	Not critical so long as the storage can support the necessary bandwidth. SSD is preferred for performance reasons.
RLV store disk drives	Enough disks to sustain the burst writes from memory to disk during commit. Needs to be at least as large as the RLV memory store

SAP IQ MEMORY MANAGEMENT

It is important to understand exactly how IQ uses memory that it was given during startup as well as how IQ allocates memory (heap memory) during its normal operations. Improper configuration of memory can lead to poor performance as it generally induces swapping.

Operating System and Non-SAP IQ Memory

It is important to remember that not all memory on a machine is available to IQ. First and foremost, the operating system will consume part of the memory. How much depends on the OS, but typically 10-20% is used as a rough guideline. Beyond that, it is a good idea to inventory all other applications, processes, etc. that will run on the host. Once that inventory has been completed a truer picture of how much RAM IQ can consume will be drawn.

If the analysis cannot be completed, it is safe to assume that 25% of RAM will be allocated to tasks outside IQ. From this point, OS and IQ monitoring should be done to reclaim any portion of that 25% that is not used.

SAP IQ Memory

Memory allocation inside IQ can be broken down into five main areas: server configuration, versioning, bitmap memory, and backup memory.

SAP IQ memory will consist of the following items:

- Catalog cache (-c/-ch/-cl options in the configuration file)
- Thread memory (stack size * number of IQ threads)
- Main cache (-iqmc option in the configuration file)
- Temporary cache (-iqtc option in the configuration file)
- IQ large memory (-iqlm option in the configuration file)
- IQ RLV memory (-iqrlv option in the configuration file)
- Version memory

Server Configuration

When the IQ server is started, a series of parameters affect how much memory it will use during its normal operations. These parameters are generally set in the configuration file and consist of:

- **Catalog cache** – The –c, -cl, and –ch parameters control how much RAM the server will dedicate to the catalog. –ca will control whether or not the RAM size is static or dynamic.
- **Thread memory** – For each thread allocated at startup, there is a chunk of memory allocated for the stack space of the thread. The total thread memory is computed via this algorithm: stack size * number of IQ threads. Stack size is set via the –iqtss parameter and the number of IQ threads is set via the –iqmt parameter.
- **Main cache** – This is set via the –iqmc parameter to size the main, persistent data cache

- **Temporary cache** – This is set via the –iqtc parameter to size the temporary, transient data cache
- **Large memory** – This memory segment is new to IQ 16 and is designed to be used for loading operations
- **RLV memory** – This memory segment is new to IQ 16 and is designed as the in-memory write optimized store for IQ. One designates which tables need this level of performance and size the memory segment large enough to handle the incoming data for a period of time. As this is a fixed amount of memory, once it fills up write operations to these tables will encounter and out of space condition.

Main and Temporary Cache

Main cache is used to house static, persistent user data and the system structures controlling access to the user data in the form of the IQ indexes. Temporary cache is that transient, volatile area of memory. Temporary cache is used to house temporary tables, internal work tables used during query and load operations, and any other data structures that are temporary in nature.

Once the amount of RAM that can be allocated to the caches has been determined, one-third of that available RAM should be allocated to main cache and one-third allocated to temporary cache.

With the advent of the optimized FP indexes in IQ 15 (FP1/2/3), main cache played a role in the use of that index type. With IQ 16, the new n-bit FP indexes (now the default) leverage the new large memory cache (LMA) so that the reliance on main cache has decreased.

In systems that perform more loads than queries, more memory is typically allocated to the temporary cache to help with the HG or WD index processing. Keep in mind, though, that lowering the main cache size will force fewer pages to be in cache at any point in time, thus increasing IO operations to move data from disk to memory and back.

In systems that tend to process more queries than loads, more memory is typically allocated to the main cache to help reduce the amount of trips that IQ must make to the main store to retrieve user data. The caveat to this is those environments where a considerable amount of temporary tables, GROUP BY, or ORDER BY statements are being used. In those situations, revert to the 40/60 memory split.

In all situations, the caches should be monitored and adjusted according to the use patterns.

IQ Large Memory Cache (LMA)

The –iqlm (Large Memory Allocator or LMA) startup parameter specifies the maximum amount of large memory that SAP IQ can dynamically request from the operating system. This new cache offers several advantages over prior versions memory management. It introduces the concept of flexible and inflexible memory that provides a central location to manage both areas as a shared resource. The LMA allows users of flexible memory for hash and hash-range table loading to request a fair share of available LMA memory and to dynamically adjust their share based upon system workload. The LMA provides a central location for all n-bit metadata memory allocations and functions as a cache, such that memory objects may later be retrieved from memory for reuse if the memory has not been reclaimed for use by other objects.

This memory is a dedicated memory area to be used during load operations as well as for the new n-bit index data structures.

IQ 16 has the ability to auto size columns such that columns with low or medium cardinality are encoded using n-bit compression and high cardinality columns are stored in a flat, compressed on disk but uncompressed in memory, structure. The two database options **FP_NBIT_Autosize_Limit** and **FP_NBIT_Lookup_MB** determine which columns will be n-bit and which will be flat.

Unless forced to be flat via an IQ UNIQUE(0) clause, all IQ columns other than LOB or BIT begin as n-bit columns when created. As data is added to the column, the optimized FP index on the column will roll over from an n-bit structure to flat storage. This will happen if the number of dictionary entries exceeds the value FP_NBIT_Autosize_Limit or if the size of the dictionary exceeds FP_NBIT_Lookup_MB.

N-bit columns use LMA memory for the dictionary, a count table and, during load/insert time, a hash table. The dictionary is the list of values stored in the column. It is an array for fixed width datatypes and a simple variable width structure for variable width datatypes. The count table is an 8 byte array of the number of occurrences of each token in the dictionary. The hash table used at insert time uses 32 bytes per token, rounded up to the nearest power of 2. For datatypes wider than 8 bytes, additional memory, the string table, is required to store the token values externally to the hash table.

The default values of 1,048,576 for FP_NBIT_Autosize_Limit and 64 for FP_NBIT_Lookup_MB are very conservative. The default values will limit a column with a datatype with a width of 8 bytes or less to use a maximum of 8 MB for the dictionary, 8 MB for the count table and 32 MB at insert time for the hash table. Most columns will use far less memory.

Typically, in a wide table, i.e. a table of 50 or so columns up to thousands or more of columns, the vast majority of columns will have cardinalities of single digits up to a few hundred or, possibly, a few thousand values. The majority of the remaining columns will typically have cardinalities of anywhere from tens of thousands up to 10 or 20 million values, heavily tilted toward the lower cardinalities. The remaining small number of columns, perhaps 10s of columns or fewer, will have large cardinalities. Therefore, with the default option settings, even very wide tables will typically use only a few gigabytes of memory for the n-bit column indexes.

During an initial load(s) into an empty table, the majority of memory used will likely not be used for the n-bit columns but rather for the columns that rollover to a flat structure. With the default values, the hash table for columns with datatypes of 8 bytes of less in width will grow to 32 MB before the column rolls over to flat, with up to an additional 64MB if the datatype is wider than 8 bytes. This memory will not be released until the end of the load. Even if 100 columns roll over to flat the memory used will only be 3 GB and it will only be used on initial loads until the column rolls over.

Because columns with a low cardinality automatically size themselves to the appropriate n-bit level and because columns with a high cardinality automatically roll over to flat, **there is very little reason to ever use an IQ UNIQUE clause** in SAP IQ 16.

If a database has a small number of tables or applications you may want to set the n-bit options higher; perhaps 10 or 20 million for FP_NBIT_Autosize_Limit and 250-500 for FP_NBIT_Lookup_MB. These options may be set as temporary options. The option settings are used both at create time and at runtime during load, insert and update. The dictionary sizes, number of tokens, datatype, etc. of a table, database, or user may be examined with sp_iqcolumnmetadata.

In special cases, IQ UNIQUE(0) can be used to force a column to be flat, and IQ UNIQUE(N), where N is greater than the auto size limit, can allow a column to be n-bit that would otherwise be flat.

RLV Cache

The row-level versioning (RLV) store is an in-memory store for high-performance row-level updates. If a table is registered for RLV storage, then all LOAD TABLE, INSERT, UPDATE, and DELETE commands write directly to the RLV store.

Multiple connections can make simultaneous updates to different rows of an RLV-enabled table. In the IQ main store, only one connection can write to a table at one time.

The RLV store periodically, and automatically, merges its in-memory contents with the IQ main store; although you can change merge preferences. You can trigger a manual merge when desired.

The RLV store combines with the existing on-disk IQ main store to provide a hybrid storage mechanism that combines the extreme performance and low-latency of the in-memory store with the robust high-performance and scalability of on-disk storage. Immediate data modifications (load table / insert / update / delete) occur within the write-optimized RLV store. The RLV store is periodically merged into the read-optimized IQ main store through asynchronous data transfer. Thus, most data in an IQ table can be accessed via indexes, and provides expected IQ query performance.

Query Tuning via Max_Hash_Rows

Currently, IQ has an option called Max_Hash_Rows that controls how much of the caches can be used for performing the various hash join algorithms at runtime. The default has been set based on systems with just 4 GB of RAM. Most systems today are larger than this and accordingly this option should be increased to help with throughput.

A good starting point for setting Max_Hash_Rows is to account for the 4 GB RAM factor that went in to it:

New Value ➔ 2.5 Million * (IQ Memory / 4GB Memory)
Where IQ Memory = IQ Main Cache in GB + IQ Temp Cache in GB

For systems with a relatively low user count, the memory component can be raised to include all RAM on the host:

New Value ➔ 2.5 Million * (HostRAM / 4GB Memory)
Where HostRAM is the total amount of RAM on the IQ host machine

This will allow for each user to use slightly more RAM.

When increasing the maximum number of hash rows to keep in RAM it is also important to increase the amount of RAM in which those hash rows can be pinned. If you just increased the number of rows that can fit into memory, you may not get the full benefit since the hash keys must fit in pinnable cache. Typically, increasing the setting for Hash_Pinnable_Cache_Percent to 30% for a typical system (35% for a lightly loaded system) will help to fit the hash keys into the caches.

The –gm startup option (number of connections) also plays a role in this discussion as it directly determines how much RAM each user connection can consume. As such, it is important to set –gm to a realistic, and fairly accurate, value to cover all situations.

Catalog Cache

The catalog cache is a portion of memory that is reserved for work that must take place outside of the IQ environment. The catalog is a portion of the IQ engine that is managed and maintained by the SQL Anywhere RDBMS (ASA or SA). The catalog handles any operations that do not reference SAP IQ objects and structures. Also, when results are returned to the client application they are handled by the catalog as well.

While, typically, the catalog is not used heavily, it is prudent to understand how to size it when heavy use of the catalog is needed.

In most environments, the catalog cache should be sized such that the amount of RAM is two to eight times the size of the catalog file. If the catalog file (.db file) is 25 MB, then the catalog cache should be 50 to 200 MB in size. For environments that need to support more than 50-100 concurrent users, the ratio should increase from 2-8:1 to 4-16:1.

The catalog cache will also control, to a degree, how many connections and users can be actively running queries in IQ or ASA. An undersized catalog cache can lead to poor performance or even client errors due to resource constraints.

In the SAP IQ server, the amount of catalog cache memory available to each active request is calculated by taking into account the server cache (-c) and the number of active requests (-gn). When a connection issues a request and the cache usage exceeds that limit, the server will reject the request with the error "Statement size or complexity exceeds server limits". To compensate for this, the amount of catalog cache can be increased and/or the number of active requests can be decreased at server startup.

Typically -gn is implicitly set by the IQ start script as -gm * 1.5. With -gm set to 100 the value of -gn would default to 150.

It should be noted that certain types of SQL can increase the amount of catalog cache in use.

Given a stored procedure that returns a result set:

```
create procedure sp_test()
begin
        select * from my_table;
end;
```

IQ provides two mechanisms to return the data. You can directly execute the procedure via call or execute:

```
call sp_test();
```

You can also select from the stored procedure:

```
select * from sp_test();
```

The second option forces the data set to be handled and stored by the catalog engine. As such, there is a dramatic increase in the amount of catalog cache that a user can consume. Additionally, there can be an increase in the amount of catalog temporary space (/tmp by default on UNIX) that must be used.

It is recommended that for typical application use, the second form of return data from a stored procedure (via a *SELECT*) not be used as it adds undue overhead to the system when the first example of simply executing the procedure accomplishes the same task with none of the overhead.

The best method for determining the amount of catalog cache used by requests is to measure it from a typical production day workload. There are certain connection and server properties that can be monitored to provide the appropriate statistics.

- **UnschReq** - Number of requests currently queued up waiting for an available server thread
- **ActiveReq** - Number of server threads currently handling a request
- **LockedHeapPages** - The number of heap pages locked in the cache
- **CurrentCacheSize** - The current cache size in kilobytes
- **PageSize** - The size of the database server cache pages in kilobytes (Note: This is the catalog page size, not the IQ page size)

The Average Cache usage for requests can be calculated from the cache usage and number of active requests as follows:

PagesPerRequest = LockedHeapPages / ActiveReq

These properties can be queried using the SQL Anywhere property() function. Since these properties represent instantaneous values, they need be queried frequently while evaluating the workload to capture the worst case values.

The following SQL will take 100 samples at 5 second intervals and calculate the maximum cache usage:

```
begin
        declare c1 int;
        declare local temporary table #tmp1(ar bigint, lp bigint) in SYSTEM;
        set c1 = 0;
        lp: loop
        set c1 = c1 + 1;
        if c1 > 100 then
                leave lp
        end if;
        insert #tmp1 select property('ActiveReq'), property('LockedHeapPages');
        waitfor delay ’00:00:05’;
        end loop;
        select max(lp) as MaxLockedHeapPages,
                max(ar) as MaxActiveReq ,
                max(lp)/max(ar) as AvgPagesPerRequest
        from #tmp1;
end;
```

The maximum number of requests that can be executed concurrently can now be calculated using the cache size and request size as follows:

MaxActiveReq = CurrentCacheSizeKB / PageSizeKB / AvgPagesPerRequest

Assuming that the current cache is set to 256 MB, the catalog page size is 4 KB, and the average pages per request is 500, the total number of concurrent active requests would be:

MaxActiveReq = (256 * 1024) / 4 / 500
MaxActiveReq = 131.072 rounded down to 131

Versioning

Version memory is allocated during runtime of the SAP IQ engine. Generally, this is a very small amount of RAM (1K-3K per version being tracked). Typically, the total version memory used is measured in KB or single digit MB sizes. On systems with hundreds of thousands or millions of versions currently active, this can become significant.

Cache Memory Used During Loads

It should be noted that there is memory allocated from within the main cache during loads. Depending on the system memory requirements and the number of columns in the table(s) being loaded, this can affect the minimum amount of RAM needed for the main cache.

IQ will allocate one page for each default (FP or n-bit) index and one page for each distinct value for LF index. For optimal performance, this memory is taken directly from the main cache. For systems where there are a lot of concurrent loads, or where the tables are very wide, this memory requirement can add up quickly. Should there not be enough cache to keep these pages pinned in RAM, performance will degrade as IQ will have to go to disk to re-read the pages necessary for the load.

HG and WD indexes use temporary cache during the load process. For a specific HG or WD index, the amount of temp required (in bytes) is roughly:

(8 + sizeof(datatype)) * numberRowsBeingLoaded

Loading 100 rows of integer (4 byte datatype) is roughly

(8 + 4) * 100 ➔ 1,200 bytes

Loading 100 rows into a varchar(20) is roughly:

(8 + 20) * 100 ➔ 2,800 bytes

WD uses substantially more because each word (token) in the data value requires some temporary cache. Each token would require the same memory as the HG index. In short, the memory requirement would be:

numberTokens * (8 + sizeof(datatype)) * numberRows

To illustrate with a simple case, assume that there is a char(20) field and within each row there are 4 tokens. The temporary cache usage is roughly:

4 * (8 + 20) * 100 ➔ 11,200 bytes

This makes the WD memory use much harder to predict because the usage amount is truly data dependent on the number of tokens being loaded.

Bitmap Memory

There is an additional amount of virtual, or heap, memory that is allocated during the loads for storing bitmap information. This memory is allocated outside of all other memory allocated for IQ. There are entities for which the load cannot and does not know anything about prior to loading the data. These entities come in the form of bitmaps to store the distinct values of data being loaded.

The need for bitmap memory during loads applies to the following index types: LF, HNG, DTTM, DATE, TIME, and CMP.

For example, with an LF index there will be a bitmap for each distinct value. For each distinct value and row grouping, the load will set a bit in the appropriate bitmap - the bitmap associated with the distinct value.

Setting individual bits in a bitmap one at a time is relatively slow. For performance reasons, bits are set in a bitmap during a load in groups of bits at a time. Storage for the groups is presently maintained in virtual memory. The size of the storage for a group is 8,192 bytes.

How does this impact memory? Suppose a given LF index encounters N distinct values during a load. The amount of virtual memory consumed by this individual LF index will be:

Total Bytes = 8,192 * N

Using an example of 500 LF indexes and assuming N distinct values per column, the virtual memory consumed is:

8,192 * 500 * N = 4,096,000 * N

As you can see, even a relatively small number of distinct values will consume a noticeable amount of virtual memory. 100 distinct values (N) in each column will use a cumulative 400 MB (4,096,000 * 100 = 409,600,000 bytes or roughly 400 MB).

Since the load engine, prior to the load, cannot know the number of distinct values that will be encountered during the load, SAP IQ cannot possibly predict how much virtual memory the LF index will use.

For LF indexes, the bitmap cache can be adjusted via the LF_BITMAP_CACHE_KB parameter. The default is 4 KB with a maximum of 8 KB per grouping.

There are other relatively small caches that are used but the number and/or size of these is independent of distinct count. For example, the HG/WD indexes use some caches to help in the updating of B-tree pages during a load. These additional heap memory caches are relatively small in size and shouldn't drastically effect memory allocations

Should bitmap memory not be fully taken into account when loading data, it is possible to induce swapping, or paging, at the OS level. This is evident by using OS tools to monitor memory usage (vmstat can alert one to swapping and paging) or I/O usage (iostat can be used to watch the swap devices for any activity). The stored procedure sp_iqstatus can also be used to point to a memory contention issue. The line for "IQ Dynamic Memory" will list the current and maximum amount of RAM that IQ has allocated for all memory structures (caches, backup memory, and bitmap memory). Watching this value will alert to an issue should this number increase the amount of RAM on the system, or the amount of RAM that was thought to be in use.

Should the utilities show that the OS is heavily swapping/paging out to swap space, or the swap space device is being used at all, there is memory contention and something must give in order to increase performance. Total system RAM can be increased or the caches can be lowered (main, temp, RLV, large memory)

How Much RAM for IQ?

There are two approaches to sizing RAM for IQ: compute all components in use and accurately size memory or use a rough starting point and adjust parameters as necessary. Computing memory sizes has been covered in detail. Should that take too much time or not be feasible, the alternative is to make an educated guess and adjust according to usage monitoring.

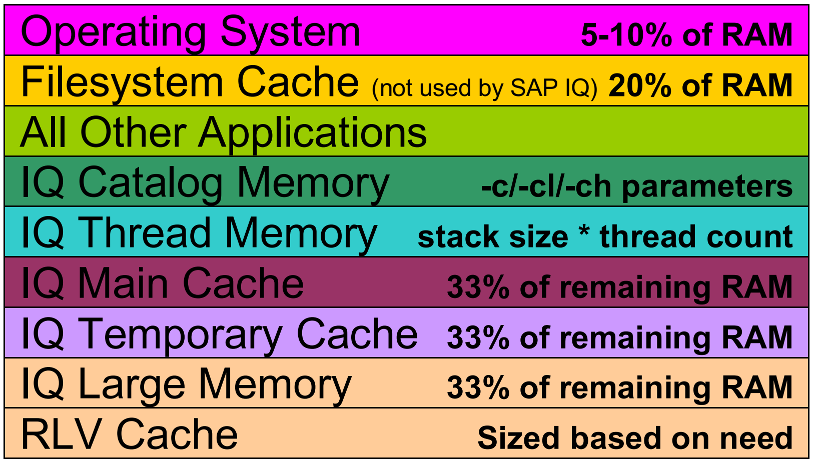
A rule of thumb for this educated guess is to allocate no more than two-thirds of the total RAM to the IQ main and temporary caches. On a system with 64 GB RAM, this would mean that the combined total of the main (-iqmc) and temporary (-iqtc) caches would consume no more than 48 GB RAM.

This will allow IQ to use a considerable amount of RAM for its operations without overburdening the system. Since the IQ caches tend to consume 80% or more of all IQ memory, over sizing these caches can cause severe performance issues should the OS need to swap out the memory to disk.

Swap Space Allocation

It is generally recommended that the system swap space be set to twice the amount of RAM on the machine. However, systems with a large amount of RAM (64 GB or more) should be able to allocate 1x physical RAM for swap space. This should provide enough of a buffer, should swapping begin, to allow administrators to detect and address the problem. As was pointed out in the previous sections, IQ can allocate a considerable amount of RAM outside of the main and temp caches (bitmap memory and backup memory). In order to prevent a situation where all virtual memory has been exhausted on the system, swap space should be set to roughly twice the amount of RAM. This gives the system enough space to compensate for unforeseen situations where the heap memory must grow beyond the physical RAM. Certainly, this will come at the cost of performance, but the operations won't fail for lack of memory (physical or virtual).

The Complete Memory Picture



Operating System	**5-10% of RAM**
Filesystem Cache (not used by SAP IQ)	**20% of RAM**
All Other Applications	
IQ Catalog Memory	**-c/-cl/-ch parameters**
IQ Thread Memory	**stack size * thread count**
IQ Main Cache	**33% of remaining RAM**
IQ Temporary Cache	**33% of remaining RAM**
IQ Large Memory	**33% of remaining RAM**
RLV Cache	**Sized based on need**

SAP IQ DISK FUNDAMENTALS

There are two primary types of storage in SAP IQ: main store and temporary store.

Main store (user defined and system defined) devices store the user data and indexes that are part of the user created tables. The main store is a permanent structure in which the data persists through a system reboot.

Temporary store devices are used to house data and indexes that are created as part of temporary or global temporary tables. During loads, the temporary store is used to cache the intermediate data to be loaded into the HG and WD indexes. During queries, the optimizer uses the temporary store for sorting and temporary repositories whose lifetime is just for a particular operation. Such repositories occur for things like updatable cursor, intermediate result sets, and join index management (synchronize join index command).

Depending on the configuration of the server, there are additional storage types to be aware of: IQ_SYSTEM_MAIN, IQ_SHARED_TEMP, and the RLV store. See the following sections for details.

IQ_SYSTEM_MAIN

The IQ_SYSTEM_MAIN dbspace manages important database structures including the free list, which lists blocks in use, the TLV, node-to-node system communication for object changes, and other internal structures for proper operation of SAP IQ.

It is highly recommended that IQ_SYSTEM_MAIN NOT be used for user data. SAP IQ allows for the creation of user defined dbspaces for this purpose.

It is best to think of IQ_SYSTEM_MAIN, in its entirety, as a system area much like master (and master.dat) in SAP ASE: something that should never be used to store user data. In SAP IQ, it is time to think of IQ_SYSTEM_MAIN as a system area like master and our catalog. In most RDBMS engines, it is not typical to mix user data and structures with system data and structures. SAP IQ is no different.

In IQ_SYSTEM_MAIN, the dbspace and file free list is maintained. Additionally, versioning space, some node to node communication, the TLV replay, and more is done on this dbspace. By default, 20% of IQ_SYSTEM_MAIN is reserved for this. If more space is needed in IQ_SYSTEM_MAIN to store user data, more space will be reserved. Also, when adding space to IQ_SYSTEM_MAIN the entire multiplex must be shut down and the nodes synchronized.

For these reasons, it is best to leave IQ_SYSTEM_MAIN as a system area with little or no user data.

IQ Shared Temporary Storage

IQ_SHARED_TEMP is a dbspace that is only available in a multiplex. Initially, the dbspace was used as a method to shared intermediate query results between nodes when a query was run in DQP mode. With IQ 16 this has been extended so that it can also be a shared system temporary store instead of requiring a separate local temporary store for each secondary server. The shared system temporary store simplifies multiplex configuration, improves performance, and supports distributed query processing.

On multiplex systems:

- When you set the logical server policy option TEMP_DATA_IN_SHARED_TEMP ON, SAP IQ creates all temporary objects on the IQ_SHARED_TEMP dbspace. You must restart secondary nodes after setting this option or after adding a read-write file to the shared temporary store. (If the shared temporary store contains no read-write file, or if you do not restart secondary nodes, data instead writes to IQ_SYSTEM_TEMP.)
- Temporary user objects (such as tables or table indexes) that you create using the IN IQ_SYSTEM_TEMP clause go in either IQ_SYSTEM_TEMP or IQ_SHARED_TEMP, depending on the value of the logical server option TEMP_DATA_IN_SHARED_TEMP:
  - If TEMP_DATA_IN_SHARED_TEMP is 'OFF', objects go in IQ_SYSTEM_TEMP.
  - If TEMP_DATA_IN_SHARED_TEMP is set 'ON', objects go in IQ_SHARED_TEMP.

RLV Store Persistence Log

The asynchronous write-ahead persistence log for the RLV store is separate from the database persistence log. It tracks and makes durable all new and modified data stored the RLV store.

Each RLV enabled table has its own logical persistence log. The space for these logs comes exclusively from the RLV dbspace. Log space is consumed during transactions and is freed by merge. At the end of a merge, the log for a table is truncated back to the oldest open transaction at the beginning of the merge.

The RLV persistence log contains a disk-based copy of the contents of the RLV store. It is stored in a compressed format to balance disk utilization and runtime recovery performance. The log is organized per-table and is stored exclusively in the RLV dbspace. It uses efficient, asynchronous I/O to minimize table modification overhead, and efficient, parallel processing for fast recovery on restart. The log is used to restore the in-memory RLV store on server restart after clean or abnormal shutdown.

Read and Write Operations

SAP IQ makes all read and write operations directly to cache. In short, nothing ever directly goes to disk. All main and temp store operations go directly to the IQ buffer cache. From there, data only flushes to disk from the cache if either:

- The cache 'dirty' pages exceeds some threshold
- At 'commit' time if the transaction is 'committing' a modification to a table

Given enough memory for main or temporary cache the store will rarely be used. Since most systems won’t have enough memory to cache all operations careful consideration must be taken when designing the disk layout and placement of the stores for optimal performance.

Once the data is written to cache, it will then be written to disk. When IQ writes to disk it takes a memory page and the data it contains (user data, indexes, bitmaps, etc.) and applies a compression algorithm to that memory page. The end result of the compression algorithm is a chunk of data that is, by default, a fraction of the page size that is evenly divisible by 16 (the default configuration allows for 16 blocks per page). For a 256K page, each block would equate to 16 KB. A read or write will always be in multiples of the block size to disk. It could be as few as 1 block (max compression) or as many as 16 blocks (no compression). This read or write operation will be a single I/O request. The data on the page will drive the compression ratio and thus the disk write size.

IQ will very rarely issue a read or write that is the size of 1 block unless we compressed the page down to 1 block. All reads and writes to disk are always done in multiples of the block size. It is very rare to ever see a page compress down to a single block on disk. Basically, if you are monitoring the storage subsystem what you would observe is variable length reads and writes. The "variable-ness" will depend on our compression.

File Placement and Location

The placement of various IQ files (catalog, transaction log, IQ message file, server log, and server error) can impact performance. Typically, though, these files are written too infrequently so that placement does not play much of a role.

Recently, we have seen more customers placing these files on a network filesystem (NFS). Internal testing and customer testimonials has shown that the NFS based log devices can be 5-20x slower than placing those files on a local filesystem. This must be considered when deciding where to place the log files as it can directly impact performance of IQ operations. Those files should be placed on local filesystems so that all network overhead associated with NFS based reads and writes can be eliminated.

SIZING CPUS, CORES, AND PROCESSORS

Data Loading/Manipulation

Single row data changes in SAP IQ are invoked via the *insert, update,* and *delete* statements. In short, CPU count and memory size won’t matter much with these operations. Having one to two CPUs is sufficient to handle these types of changes for a user. As a side note, these types of operations are not optimal for changing data in SAP IQ and should only be used if performance is not a concern.

The highest performance load mechanism in SAP IQ is the bulk loader. Bulk load sizing encompasses any SAP IQ loads that are done via the *load table*, *insert from location*, *select into* or *insert select* syntax. These mechanisms invoke the internal IQ bulk loader for making mass changes to the database. Multi-row updates and deletes would fall into this category as well due to the parallel nature of the operation. This is a logical categorization of where the operation falls, not an indication that the *update* and *delete* operators invoke the IQ bulk loader.

Sizing an IQ system for loads is straight forward. The new bulk loader design in SAP IQ 16 allows for a fully parallel load for nearly all bulk load operations. This includes *load table*, *insert from location*, *select into* and *insert select* operations. The bulk loader has been redesigned such that it will scale linearly as cores are added to the system. In short, a load on 16 cores will take twice as long as the same load on 32 cores.

The current lab and customer load rates observed are 20+ MB per second per core.

- 16 cores: 16 * 20 MB * 3600 seconds = 1125 GB/hour
- 32 cores: 32 * 20 MB * 3600 seconds = 2250 GB/hour
- 64 cores: 64 * 20 MB * 3600 seconds = 4500 GB/hour

The overall performance will vary based on processor speed as well as available IO bandwidth. It is important to note that IQ cannot achieve these load rates without a source filesystem that can read data at the same, or higher, rate.

Queries

It is important to classify the queries so that we can better understand the CPU utilization to be expected from each query. It is hard to determine the criteria for queries without looking at runtime. It is impossible to say that all queries with 17 table joins will be long/complex queries. They may run in 30 seconds, they may run in 30 minutes.

As a general rule, expect that any given query will need 1 to 2 CPUs for the duration of its execution. The duration of query execution may be milliseconds, seconds, or even hours depending on how much data the query is churning through. To further refine the query performance and execution profile of queries, we will attempt to classify queries based on execution time.

For the purposes of this section, we will define the following four query classes:

- **Super fast** – Queries that generally take less than five seconds to run
- **Fast** – Queries that generally take less than three minutes to run
- **Medium** – Queries that generally take between three and 10 minutes to run
- **Long** – Queries that generally take more than 10 minutes to run
- **Super long** – Queries that generally take more than 30 minutes to run

Now that the types of queries have been defined, we need to apply the types of queries to CPUs on the host.

- **Super fast** – Each CPU should be able to handle 10 or more concurrent queries
- **Fast** – Each CPU should be able to handle between five and 10 concurrent queries
- **Medium** – Each CPU should be able to handle between two and five concurrent queries
- **Long** – Each CPU should be able to handle between one and two concurrent queries
- **Super Long** – Each CPU should be able to handle at most one query, generally a query will need more than one CPU

There is one large exception to the above sizing: parallel processing. Over the past few years quite a lot of effort has been put into expanding the queries that can run in parallel on a single node. Even more effort has been put behind expanding those single node parallel queries to run fully parallel across all nodes in a multiplex.

A query will perform quite differently on a system with 4 CPUs, 8 CPUs and 12 CPUs. Just as that same query running on 4 nodes with 8 cores each will perform very differently than on a single node with 8 cores.

This makes the notion of sizing user workload more difficult in that the business, user expectation, and service level agreements must be taken into account now. But it does also now allow for adding more cores to a system (inside an existing node or adding more nodes) to affect performance improvements for even a single user.

Overall system performance will also change as more users are added to a single instance. As more users are added and the queries can be executed in parallel, the landscape of resources available to IQ to perform the queries will be altered.

The parallel processing enhancements keep a core tenet to strike a balance between single user performance and overall system concurrency. This is achieved through a dynamic reallocation of resources at runtime. Simply put, the first user in will get all cores needed for the query. The second user in will not wait for the first user to complete, but rather the optimizer will pull resources away from the first user so that both users can equally consume the CPU resources. The third user in would experience the same behavior; with all three queries would be consuming roughly one-third of the system each. As a query comes to completion the resources are freed up and given back to any currently executing queries.

SIZING MEMORY

Cache and Large Memory Sizing

This new IQ Large Memory (LMA) cache offers several advantages over prior versions memory management. It introduces the concept of flexible and inflexible memory that provides a central location to manage both areas as a shared resource. The LMA allows users of flexible memory for hash and hash-range table loading to request a fair share of available LMA memory and to dynamically adjust their share based upon system workload. The LMA provides a central location for all n-bit metadata memory allocations and functions as a cache, such that memory objects may later be retrieved from memory for reuse if the memory has not been reclaimed for use by other objects.

The key is determining how much memory to set aside for the caches. Fortunately, this process has been made fairly simple due to the enhancements to the memory subsystem. In testing done in the lab as well as with pre-release systems at customers, the recommended sizing for the caches is 33% each: 33% for main cache, 33% for temporary cache, and 33% for the large memory cache. The amount of RAM to be divided into thirds is typically 90% of available RAM on the system. A proper memory inventory is advised to properly determine the amount of RAM to be used.

RLV Store

The RLV store need only be sized large enough to keep the data in memory that is being loaded and changed prior to being merged with the SAP IQ main store data. This amount will vary depending on the number of tables, load frequency, and merge frequency.

A good starting point for most systems is 250-500 MB per table marked for RLV operation. This should allow for enough of a safety margin in case load frequencies spike on all objects simultaneously.
M as performance of the main and temp caches will now be drastically impacted.

Data Loads and Modifications

In general, single row operations in IQ don't require a significant amount of memory. The amount of memory necessary for any single row *insert, update, or delete* operation would be calculated using the same algorithm as for bulk operations. The difference is that during single row operations there is just one row being affected.

Generally, having more RAM for main and temp cache is better. For loads, though, this isn't always true. Temp cache needs to be sized large enough to store the HG and WD index data used for building the index structures. Main cache needs to be large enough to store the header pages for all indexes so that they don't have to be retrieved from disk due to being swapped out. Besides the main and temporary caches, though, you must allow enough RAM to exist outside of IQ for IQ large memory that is used for each load.

During loads, HG indexes will use temporary cache to store the intermediate data necessary for the HG indexes. During pass one of the load process, the data and information necessary to build the HG index from the inbound data is stored in the temporary cache. During pass 2 of the load process this data is then integrated with the current HG index structures and written back out as the final step during pass 2 of the load process. Should there not be enough temporary cache to store the data in pass 1, the server will flush pages to disk in the temporary store. To better optimize loads, it is recommended that temporary cache be sized so that paging from memory to disk is reduced or eliminated. For large loads, this can be costly so weigh performance versus price accordingly.

Temporary cache sizing is quite simple. This cache, like main cache, should be sized at roughly 30% of the available RAM on the host.

NOTE: A system with 512GB RAM should allocate roughly 150 GB to main cache, 150 GB to temporary cache, and 150 GB to large memory. Changes to these ratios should only be made after monitoring the system and fully understanding the impact of deviating from these ratios.

Queries

When sizing memory for queries, the general rule of thumb is that the system should be configured for a minimum of 8-16 GB of RAM per core. For smaller systems (less than 8 CPUs) it is recommended that the system be sized closer to 16 GB RAM while larger systems be sized in the range of 8-16 GB per core.

Multi-host Configuration

SAP IQ has the ability to be run on multiple hosts that share the same main disk space. This is often referred to as SAP IQ Multiplex, multiplex, or a multi-node configuration.

Given the above CPU sizing guide for queries, it is possible to put the CPUs on different machines, rather than having to create a single system with a tremendous number of CPUs. This method also allows the load to be segregated between hosts. It is possible, via an application layer, to assign certain hosts the ability to handle certain types of queries.

For instance, let's assume from the above guide that 16 CPUs are needed for the warehouse. It will matter little to most queries whether those 16 CPUs are on a single host, or multiple hosts. It is possible to implement this solution with 4 systems that each has 4 CPUs and 16-32 GB RAM. A solution using smaller hardware may be significantly less expensive to purchase when compared to a single system with 16 CPUs and 64-128 GB RAM.

SIZING STORAGE

IQ_SYSTEM_MAIN Sizing

Sizing the default main store (IQ_SYSTEM_MAIN) is relatively straightforward and based on the size of the user defined dbspaces, in total, as well as the number of nodes in the multiplex.

For databases that are less than 1 TB, it is recommended that IQ_SYSTEM_MAIN be 15-20% of the user defined main space size, with a minimum of 32 GB. If this instance is migrated to a multiplex, an additional 10 GB of space should be added per node in the multiplex. For a 2 node system with a 100 GB database, the size would then be 52 GB (32GB base + 10 GB per node).

For databases that exceed 1 TB, it is recommended that IQ_SYSTEM_MAIN be at least 150 GB plus 10 GB for every additional TB. A 10 TB database would require 250 GB of IQ_SYSTEM_MAIN (150 + 10*10)

For a multiplex, an additionally 10 GB of space should be allocated for each node of the multiplex. For a 4 node system with a 10 TB database, you would need 290 GB of system main =(150 GB + 10TB*10GB per TB + 4 nodes*10GB each).

General Guidelines

Disk sizing requirements for SAP IQ change as the CPU count and speeds increase. The numbers in this section provide a general guideline across all systems. If the system in question is one of the faster on the market, consider increasing some of these values. This is done to compensate for the amount of work that the CPUs and memory are doing. SAP IQ is generally not disk or I/O bound, but rather CPU bound. As CPUs increase in speed and throughput, it drives the bottleneck closer to the disk subsystem. With the enhancements made to IQ 16 more of a burden is being placed on storage due to the massive parallelism that can now be achieved.

Make sure that multiple FC HBAs do not share the same bus as that may become a bottleneck as well.

During data loads into permanent or temporary tables, each CPU can process 20+ MB/sec of data. In addition to loading, queries can consume 50-100 MB/sec per core. As a rule, design the disk farm to be able to deliver

20+ MB/sec to each core in the multiplex that will be loading data. Add an additional 50-100 MB/sec per core that will be processing queries. Addition bandwidth capacity needs to be included for the IQ temporary store. That number is typically equal to the bandwidth needed for the IQ main store due to the high speed, low latency requirements that the temporary store (shared and system) have.

Using an LVM is not a requirement for SAP IQ to operate and generally provides no value add for SAP IQ (this is not to say that an LVM does not have operational benefits outside of SAP IQ, though). In multi-host configurations, using an LVM can actually increase the cost of deploying a system as the special “clustered” versions of the LVM must be used. These versions add cost to the total system.

Raw devices enable higher I/O speed compared to file systems, and elegantly enable multiple server nodes to access the same storage devices without additional complexity of shared file systems. With this said, though, SAP IQ has been enhanced and now allows direct I/O for those sites that cannot use raw devices. Proper configuration of the filesystems must be done to ensure that SAP IQ continues to perform at its peak levels. Typically, this means that the filesystem should be configured to allow for very large disk operations that are not broken down into much smaller units of works on disk.

In years past, and many revisions of the Sizing Guidelines, we paid careful attention to the type and number of disks that IQ has access to. In recent years, however, sizing methodologies have moved away from this and towards a simpler sizing metric that does not dictate what type of storage, RAID levels, LVMs, etc are used.

The general sizing metric is to size storage for roughly 50-100 MB of IO per second per core (across all nodes where applicable). So long as this throughput can be achieved by the storage architecture, any type of disks and RAID level can be used.

Storage IO Block Sizes

With most modern storage systems it is recommended that the storage block/stripe size match the IQ page size as closely as possible without exceeding it. This offers the best performance as a single, very large chunk of data can be laid to a single disk in a single contiguous chunk.

Physical Drives and Device Controllers

The general sizing metric is to size storage for roughly 50-100 MB of IO per second per core (across all nodes where applicable). This can be done with any type and number of physical disks from SAS to SATA to fiber channel to SSDs and even flash storage. They key is to maintain the throughput across all storage at peak system load.

When sizing the controllers for IQ, main store and temporary store controllers follow the same algorithm of 50-100 MB/sec per core. However, main and temporary store controllers and disk paths should be sized separately from one another for optimal performance and workload segregation.

Device Placement and Mapping

It is recommended, but not mandatory, that the IQ main and temporary store devices be physically separated and not reside on the same LUNs and physical drives in the storage system. Primarily, we want to increase overall throughput and placing these devices on the same hard drives will cause thrashing. Secondarily, though, most storage systems have caching mechanisms that can be tuned by device. This allows for tuning the drive array caches differently for the different IQ device types and the varied use each has. It also allows faster drives like SSDs and flash based disks to be used for the temporary storage where latency and throughput are both critical.

It is equally important for best performance to segregate the workload between filesystems, main store, and temporary store devices. The segregation should include separate HBAs or I/O controllers as well as different paths through the storage infrastructure.

SIZING NETWORK

Performance

There is very little node to node communication in an IQ multiplex when DQP is disabled and data loading is done on the coordinator.

If write nodes are used to load data, there is a slight increase in network traffic as each write node must communicate with the coordinator for locks, storage allocation and deallocation, and commit/rollback operations.

IQ 16 has the ability to use the network for data transmission in DQP operation (DQP_Enabled_Over_Network option being set to ON). This will have a significant impact on overall network requirements.

The tendency in the user community is to use IQ with DQP. With the new feature of network based data sharing coming into this release, it is strongly recommended that 10 gb Ethernet be used.

SAP IQ also supports both public and private networks. It is strongly recommended that private networks be used for all node to node communication and DQP data transmissions. This will isolate server to server communication as well as give the full and isolated bandwidth to both the private network communication as well as all client-server communication over the public networks.

In order to improve performance and reduce the potential contention the following should be considered:

- Use faster network cards
- Use a faster network topology (10 gigabit vs. 100 megabit)
- Add multiple network cards to the IQ host and allow the applications to use the different IP interfaces
- Increase the packet size of the client application (the method varies depending on ODBC, JDBC, or Open Client connectivity)
- Increase the packet size of the remote data load by using the ‘packetsize’ parameter to the insert…location syntax. It should be noted that this is a documented feature of IQ, but as of yet this is currently limited to 512 byte packets and is being addressed in a future release.

Content Switches and IQ Multiplex

A new feature was added to version 16: login redirection. This provides load balancing when a user tries to log into an overloaded node by redirecting the attempted login to a node that is less loaded in the same logical server. It also allows for connections to be redirected to appropriate logical servers based on the login policies set forth.

Use the LOGIN_REDIRECTION logical server policy to enable redirection. You can then define your own logical server policies and assign them to one or more logical servers to specify server behavior. Connection parameters, logical server policies, login policies, and user privileges determine the nodes available to a particular user for processing.

You can specify the current coordinator without knowing the host name by using the built-in logical server, COORDINATOR.

If login redirection is enabled and a connection is allowed, SAP IQ redirects connections when:

- The initial connection node is not a member of the target logical server.
- The initial connection node is a member of the target logical server but has a role other than that requested.
- The initial node is a member of the target logical server and has the requested role, but the user has reached the limit of maximum connections on the current logical server member node.

There are many situations where the above login redirection capability needs to be extended. Many customers employ content switches for this purpose. Network manufacturers such as Cisco, F5, and ServerIron provide content switches that are typically used in Web or FTP farms. These switches provide a single IP address and port interface to applications rather than allow multiple TCP/IP network connections. The switch then maps this IP/port combination to a series of backend servers (physical). The number, configuration, and makeup of these servers are immaterial to the application and completely shielded via the content switch.

This design allows any application to connect to an IQ Multiplex farm without having to be aware of the individual nodes. It also provides a mechanism for seamlessly handling outages, both planned and unplanned. When a node goes offline the content switch simply redirects the connection to an available system. The old system is marked as being offline and no future connections will be sent there until it is brought online and marked as available to the content switch.

Multiple ports can also be used to further segregate workload. For instance, a second port could be added for high profile users. Since all redirection in the content switch is done at the port level, that port could be reserved for use by certain client machines. In short, workload segregation is not lost with this architecture, but further enhanced. Applications and users can be segregated as before, however, the added benefit of seamless failover is brought to the infrastructure.

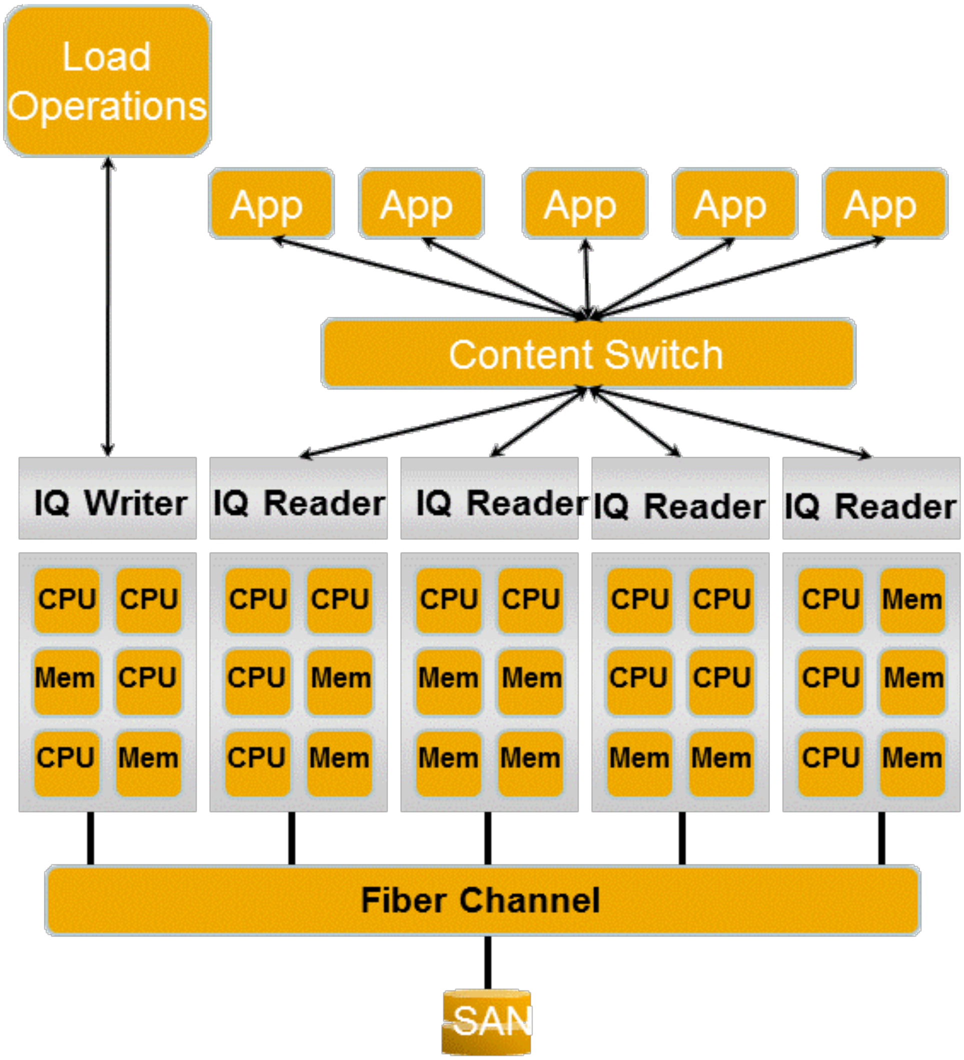


Figure 1 – IQ Multiplex with Content Switch

IQ PAGE SIZE

SAP IQ allows the database creator to set the page size that is used for the IQ main store and temporary store disk devices. The current default is a page size of 128 KB. You can, however, define a page size of 64 KB, 128 KB, 256 KB, and 512 KB.

There are very few situations, today, where a page size as small as 64KB is warranted. This size is generally viable on systems with very little RAM (under 4-8 GB RAM), 32-bit systems, high concurrent user count environments with limited storage and memory, and very small databases.

When moving data from memory (IQ page) to disk (IQ block), IQ compresses the page into IQ blocks on disk. When IQ writes to disk it takes a memory page and the data it contains (user data, indexes, bitmaps, etc.) and applies a compression algorithm to that memory page. The end result of the compression algorithm is a chunk of data that is, by default, a fraction of the page size that is evenly divisible by 16 (the default configuration allows for 16 blocks per page).

The output of the compression step will always be a contiguous chunk of data to be written to disk. That chunk will be variable depending on how well, or not, that the data on that page compresses. The chunk will always be in multiples of the IQ block size (default is 1/16th of the page size).

Though the IQ block size can be specified during database creation, it is generally not needed. The default block size (1/16th of the IQ page size setting) is acceptable in most cases.

The IQ page size applies to both main store devices as well as temporary store devices. This is an important aspect to keep in mind. A system with a high concurrent user count may drive quite a bit of temporary table and work table usage through temporary store. Usually, temporary tables contain a fraction of the data that the source tables do. In quite a few cases this can be just tens or even thousands of rows. This low row count can incur more overhead due to the minimum requirements to store data on a page or pages.

A page will be allocated for each column and index on those temporary objects. If an object has just 100-1000 rows, a vast majority of the page may be empty, but still allocated. This can dramatically increase the amount of space needed to the IQ temporary store and the temporary cache for a given data set when compared to the data being stored.

For example, consider a table with 10 columns and no indexes. This will require a minimum of 10 pages (one per column). Depending on datatypes and index optimizations, a page may be able to store tens of thousands of rows. Creating a table to store a few hundred rows of data will require the same storage as the same table with thousands of rows of data.

When deciding which page size to use, several factors need to be considered. Some factors will drive the page size; others will be consequences of the page size setting.

- Concurrent users
- Table row count
- Operational impact (memory, temporary storage space, versioning space)

Concurrent Users

There is no guideline which says that a system with fewer than X users should use a certain page size. Concurrency shouldn't drive the IQ page size that a database is created with. However, the total system impact that concurrency and page size has can be quite drastic. It is important to balance the page size, concurrent users, and virtual storage (caches and dbspaces) for IQ temporary store.

As the page size increases so must the amount of RAM and disk space necessary to support the larger page size. As the concurrency increases, the impact of a larger page size will drive RAM and disk space to be increased to handle the increased users.

Table Row Count

Though the manuals reference ultimate database size as a factor for the page size, the current practice is to look at this as a last resort when no other information is available to determine an optimal page size. The best data related factor for determining page size is the total number of rows expected in the largest table.

To paraphrase chapter 5 "Choosing an IQ Page Size" of the *SAP IQ System Administration Guide* these are the recommendations for setting the IQ page size:

- 64KB -- For databases whose largest table contains up to 1 billion rows. This is the absolute minimum for a new database. On 32-bit platforms, a 64KB IQ page size gives the best performance.
- 128KB -- For databases on a 64-bit platform whose largest table contains more than 1 billion rows and fewer than 4 billion rows. 128KB is the default IQ page size. This is generally acceptable for most environments.

- 256KB -- For databases on a 64-bit platform whose largest table contains more than 4 billion rows.
- 512KB -- For databases on a 64-bit platform whose largest table contains more than 10 billion rows.

Experience in recent implementations of SAP IQ reflects the following:

- 64KB -- Hasn't been used for systems other than 32-bit Windows
- 128KB (IQ default) -- A majority of implementations have used this default page size. These systems generally have tables with fewer than 2-3 billion rows.
- 256KB -- Systems that have used this page size have had tables in the 4-8 billion row range
- 512KB -- The larger systems with tens of billions of rows in tables have used this page size

Using row counts alone should not be the only indicator of what IQ page size is appropriate. The general consensus and use pattern is that as the amount of RAM on the system is increased, the page size can also be increased.

Operational Impact

There are many downstream effects of increasing the IQ page size that must be considered, including space consumption, memory use, and versioning.

Space consumption for objects may not be an issue for those in main store as they are generally larger and will fill pages regardless of size. Small lookup and reference tables (less than a few thousand rows, generally), however, may see an increase in size with an increased page size. A smaller page may be 50% full with data while a larger page may be only 33% or even 25% full with the same data; the data didn't change, just the storage "bucket" that the data is held in.

While IQ applies compression algorithms prior to storing the data on disk, this may still be a concern. The final data at rest on disk will be done in blocks. A block, by default, is 1/16th the size of a page. As the page size increases, so does the block size. A data page of 128KB may compress to a single 8KB block. A 256KB page with the same data on it will consume 16KB on disk.

Memory use won't increase or decrease as the page size is changed, per se, as the main and temporary caches are still the same size. If the memory that IQ can reference in its caches isn't increased, though, performance may suffer.

A system with just a handful of users will, more than likely, not experience as many issues related to undersized memory as those environments with a lot of concurrent users.

As concurrency increases, the number of pages that are in use will increase in both main and temporary cache. When moving to a larger page size (for instance, doubling from 128 KB page to 256 KB page) the number of pages that can be stored in the same amount of RAM is cut in half.

If a user is referencing a fraction of the data on a page more pages will need to be brought in to handle the query. Should the amount of RAM dedicated to the caches not be increased, a burden will be added to the system that can force pages that are the least recently used to be flushed out to disk before the new data can be brought into RAM. To properly handle the same number of users with a doubled page size, it is possible that the IQ cache sizes need to be doubled in size.

Should the page size be changed solely on the number of rows in the table, memory contention and a possible increase in disk activity can follow. Typically, systems with larger objects have more memory available and thus don’t suffer from the impacts. If the RAM sizes don’t change over time and the tables are expected to grow to the point where the row count sizing would force a larger page size, it is recommended that the page size increase be thought through as possibly disregarded.

Versioning is one aspect that is often overlooked as it relates to page sizes. The number of pages needed to track versions won't change; it is the amount of cache or disk that would increase. If one were to double the page size, the amount of cache or storage needed for versions would also double.

THREADS

Catalog Startup Thread Allocation (-gn) and Concurrent Users (-gm)

By default, the catalog starts with 1 thread per user connection (-gm) plus five. This ratio is too low.

When setting the -gm option to set the total number of concurrent logins, it is critical to also set the -gn option. Given that -gm is 200, this means that IQ will only allocate 205 threads for the catalog, of those 205, 200 will be reserved for network connections. This leaves just 5 threads for all other catalog activity, including the critical coordinator workload.

As a general rule, follow the guidelines below for the -gm and -gn settings:

- If -gm is less than 100, set -gn = -gm + 50
- If -gm is >= 100, set -gn = -gm * 1.5
- On a coordinator node, set -gn to (-gm + 50 + number_of_iq_nodes)
- On a coordinator, high user concurrency is not needed for external connections so -gm should be relatively low

It is of vital importance to understand what happens when setting -gm too high. This option reserves resources at startup so that the connections will always suceed. It is important to set -gm to a value that will meet the peak workload of the system and that the peak be reasonable. On average, imagine that a system has no more than 20 connections from users/applications. But yet, -gm is set to 200. This will leave 180 threads (200 defined – 20 in use) allocated but not being used.

IQ Startup Thread Allocation

The default number of threads allocated in IQ during startup depends on two things only: the number of CPU cores and the number of user connections (-gm startup parameter). The total number of threads IQ allocates at startup can also be overridden by using the -iqmt startup option.

By default, -iqmt is set to:

numCatalogThreads = If set, uses -gn. If -gn is not set, it is computed by -gm + 5

60*(min(numCores,4)) + 50*(numCores - 4) + (numCatalogThreads + 6-10)

The number of extra threads (6-10) is version dependent

On a 4 core system with -gm set to 20 and no value for -gn, that would be:

iqmt = 60 * (4) + 50 * (4-4) + (20+6)
iqmt = 266

On a 12 core system with -gm set to 30 and -gn set to 50, that would be:

iqmt = 60 * (4) + 50 * (12-4) + (50+6)
iqmt = 696

There are two distinct types of threads: connection threads and server threads. They do not form one big pool for all uses.

Connection threads come from the part of the default calculation based on number of connections the server is configured for: (numConnections + 6). These are reserved for connections and will not be used as workers for query or load processing.

This behavior can be observed by using the IQ Buffer Cache Monitor with the -threads or -debug option. When "Free Threads" equals "Reserved Threads" the only threads that are left are those kept back for connection use. Consequently, there will be no threads left over for parallel operations.

Server threads come from the part of the default calculation based on the number of CPU cores: 60*(min(numCores,4)) + 50*(numCores - 4). These threads are used in support of parallel loads or parallel query operations as well as those used for disk I/O. You must have free server threads to perform operations in parallel.

When you specify a value for -iqmt, it is imperative that -iqmt be larger than the default number of threads. If a value lower than the default is specified, IQ will ignore it and start the server with “minimum threads” which provides ONLY the connection threads. Set -iqmt no lower than the default calculation as previously described. This allows IQ to keep a pool of threads for query and load work as well as for connections.

There is an upper limit of 4096 threads for a 64-bit system. The total number of threads represented by -iqmt plus startup parameter -gn must not exceed this number. The value of –gn defaults to –gm + 5 where -gn is the number of threads used by the SQL Anywhere engine.

Disk I/O Threads

There are two types of threads that perform disk I/O: sweeper and prefetch. The sweeper threads write out dirty buffers and prefetch threads read in buffers. Within SAP IQ, each cache (main and temporary) has a set of these threads.

The number of threads for each team is controlled by the following options:

SWEEPER_THREADS_PERCENT
PREFETCH_THREADS_PERCENT

Each of these is a percentage of the total number of threads allocated to IQ, not including the SQLAnywhere threads. The default for the above two options is 10. This means that 10 percent of the total IQ threads will be allocated to the sweeper team and 10 percent will be allocated to the prefetch team per cache. Using the default setting of 10 for the above options means that 40% of all threads are reserved for I/O work.

Normally the sweepers will write data out and the prefetch will read data in. If the sweepers fall behind in the wash area, then the buffer manager will end up handing back out dirty buffers in which case the write gets done by the receiving thread "inline" before it reuses the buffer. Similarly if prefetch cannot keep up with prefetching, then the requesting thread will not find the block in memory and will have to do the read itself. In an ideal world, sweeper and prefetch threads would be the only threads doing disk I/O.

The sweepers go through the least recently used (LRU) chain buffer by buffer; all the while looking for buffers to write. There is no disk and thread queuing, per se, as the thread team just watches the wash area and writes whatever is found.

The prefetch team is given buffers to fetch by the optimizer. If the optimizer has no need for data, the prefetch team will wait to be tasked. It is possible for the prefetch threads to do double duty for other reads besides prefetches. This is controlled internally and cannot be changed.

Are Enough Threads Available?

The -debug output of the IQ Buffer Cache Monitor has an interesting section on threads in which detailed thread information can be gathered. This same information is also available via the *sp_iqsysmon* procedure.

From a sample report:

```
ThreadLimit= 1499 ( 100.0 %)
ThrNumThreads= 1499 ( 100.0 %)
ThrReserved= 566 ( 37.8 %)
ThrNumFree= 787 ( 52.5 %)
NumThrUsed 712 ( 47.5 %)
```

We notice a few things from this example.

First, ThreadLimit is always equal to ThrNumThreads, which is always the value of -iqmt minus 1, or the default number of calculated total threads minus one. IQ holds back one thread for emergency connection use. The 1499 above was from a system with -iqmt set to 1500.

Second, (ThrNumFree + NumThrUsed) always equals ThreadLimit. At any point in time the number of free threads plus the number used is equal to the total number of threads available. There is no counter for NumThrUsed; it is calculated as (ThreadLimit - ThrNumFree).

IQ counts reserved threads via ThrReserved. This, generally, represents the number of threads in the connection thread pool. This counter does not relate directly to the others -- it can fluctuate, slightly, without seeing the other counters go up or down by the same amount. The slight fluctuation comes from threads that are reserved, shortly, for I/O operations.

Free threads are tracked via ThrNumFree. This represents how many threads are free for all server activity: connection threads and server threads. When ThrNumFree shows the same number as ThrReserved it means that all the available worker threads are being used and only those held back for connections are available. When this situation arises, or when they are found to be close in value, using -iqmt to provide more worker threads is advisable.

www.sap.com/contactsap

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

THE BEST RUN SAP®