SYBASE®
An SAP Company

TECHNICAL WHITE PAPER

A Practical Hardware Sizing Guide for Sybase® IQ 15

Sybase IQ and RAP Store a Component of RAP – The Trading Edition®

Abhay Kale, Application Engineer, Sybase Inc.

TABLE OF CONTENTS

INTRODUCTION

Purpose

This document will attempt to highlight the main areas of concern for sizing a Sybase IQ environment. This sizing guide also applies to the RAP Store component of RAP – The Trading Edition. The RAP Store component is based on Sybase IQ and, as such, would be sized the same as a stand-alone Sybase IQ instance.

Sizing of Sybase 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 Sybase 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.

This document does not include hyperthreading and its capabilities into the CPU terminology. Hyperthreaded systems haven't proven themselves to provide a significant boost to Sybase IQ performance and as such, hyperthreading cannot be taken into account for the guidelines and algorithms in this document.

As such, a core that is hyperthreaded should be treated as a single core/CPU. To effect this change in Sybase 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 AIX Power 6 and Power 7 platforms. Most implementations today on IBM hardware follow the pattern of enabling SMT (SMT2 for Power 6 and SMT2 or SMT4 for Power 7), and then letting Sybase IQ default to seeing all threads as available cores for processing.

Copyright

CONCEPTS OF NOTE SYBASE IQ 15.0 TO 15.2

Parallelism

Version 15 of Sybase IQ brought about a fundamental change in both load and query operations. The shift was to an engine that is able to perform significantly more operations in parallel than ever before. As Sybase IQ has matured from Sybase IQ 15.0 to 15.3, many more operations have been made to execute in parallel.

The parallelism that came into being with Sybase IQ 15 attempts 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 load or query operation. The second user in will not wait for the first user to complete, but rather the engine and 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 operations consuming roughly one-third of the system each. As an operation comes to completion the resources are freed up and given back to any other currently executing tasks that can benefit from more CPU resources.

Multiplex Architecture

Many new items came into being when Sybase IQ 15 was released. From a sizing perspective, two main features need to be taken into account: the coordinator and multiple writers.

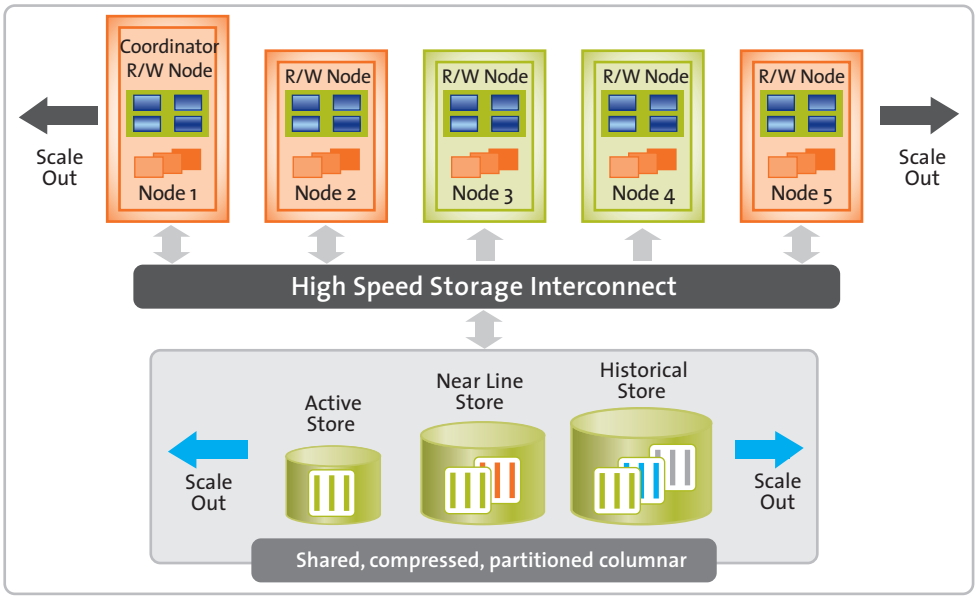
To paraphrase the *Sybase IQ Multiplex Guide*, Sybase IQ supports read-write transactions from multiple servers in the multiplex. The primary server, or coordinator, manages all global read-write transactions and maintains the global catalog and metadata. The table version log (TLV log) is also maintained by the coordinator and stores information about DDL operations and communicates information about new table versions to the secondary servers (readers or writers).

The configuration can be described as an "shared everything asymmetrical cluster," because node capabilities may differ, unlike other database cluster architectures, which usually follow either a "shared nothing" architecture.

One thing to note is that all DDL operations are run on the coordinator regardless of which Sybase IQ instance the SQL was actually issued from. Typically, this is not an issue. However, when creating indexes on fully populated tables, the actual work is performed by the coordinator, not the write node that the command was run on.

This is an important point to keep in mind when sizing the host for the Sybase IQ coordinator. Two methodologies spring from this. First, place the coordinator on an isolated host that is sized for all DBA maintenance and any real-time DDL operations such as create index that may take place. Second, have the coordinator also play the role of writer.

As previously mentioned, Sybase IQ 15 also allows for multiple nodes to write or change data simultaneously. As with Sybase IQ 12.x, though, any object can have at most a single user changing the data or structure; this architecture has not yet been changed. In systems with a lot of tables that are undergoing constant data refreshes, however, it is quite possible to architect a system where there are two or more write nodes whose sole responsibility is to handle data changes while N number of other nodes are defined as query, or read, nodes to handle all read operations in the database.



Other Features of Note

Partitioning was also introduced in Sybase IQ 15. From a sizing perspective, however, this does not come into the equation other than how it pertains to multiple dbspaces and partition placement.

In order to handle tiered storage, partitioning, and object placement on disk, Sybase IQ 15 introduced multiple dbspaces. The DBA can create any number of dbspaces to hold data objects. These objects include tables, columns, indexes, and partitions. The storage sizing guidelines contained in this document cover sizing of an individual dbspace, not all dbspaces for the database.

NEW FEATURES IN SYBASE IQ 15.3

Prior to Sybase IQ 15.3, Sybase IQ did not have a query scale out feature. The architecture of Sybase IQ was such that an application would connect to a single node in the multiplex and run the users workload against that one node. This allowed the application and database administrators to have tight controls over the Sybase IQ Multiplex and direct certain users or applications at specific hosts for true workload segregation. Should that node become unavailable, though, it was up to the application and/or user to connect to another functioning Sybase IQ node.

There are many situations where such a manual process is unacceptable and workload management application coding is a difficult, if not impossible, task.

Sybase IQ 15.3 mitigates the problem by introducing the PlexQ™ Distributed Query Platform, a Massively Parallel Processing (MPP) shared everything architecture that accelerates highly complex queries by distributing work to many computers in a Multiplex configuration. Unlike shared-nothing MPP architectures, PlexQ utilizes a shared-everything approach that dynamically manages and balances query workloads across all the compute nodes in a Logical Server within a Sybase IQ MPP configuration. PlexQ's automatic workload re-balancer aggressively works to avoid contention among users for system resources, thereby providing predictable high performance and resource efficiency for a broad spectrum of concurrent workloads.

As a result of these changes, the IO sizing guidelines (20-40 mb/s per CPU) now relate to Simplex and Multiplexes without DQP. When DQP is enabled, administrators can configure 25-200 mb/s/cpu for IO bandwidth. Since DQP behavior varies widely from instance to instance, the effect of the additional bandwidth will vary from application to application.

In order to achieve distributed processing, Sybase IQ 15.3 introduces 2 new concepts:

- Shared Temp DBSPACE
- Logical Server

Both these entities need careful planning from a sizing perspective. While the Shared Temp affects planning for disk storage, the Logical Server affects the CPU provisioning. These concepts and their impact on the hardware sizing are described in detail below.

Shared Temp DBSPACE

Shared Temp DBSPACE is a shared DBSPACE managed by the co-ordinator. Creation of the dbspace automatically enables DQP. All nodes in the multiplex have access to this dbspace.

The query engine distributes parallelized query fragments to available worker nodes. The worker nodes post the results of work assigned to them via the Shared Temp.

In Sybase IQ15.3, the Shared Temp is used only for DQP. As a result, optimum size for this dbspace will depend on the parallelism of the queries in the workload. However, the Shared Temp dbspace should be configured at the same level as the main store for the multiplex. In particular:

- Shared disks must be presented as raw devices with uniform file paths on all multiplex nodes
- Shared Disks should be provided by one or more high-performance SAN devices
- Each host should have dedicated Host Bus Adapters and adequate bandwidth
- A worst case high DQP workload may require shared storage approaching the sum of the local temporary storage usage for all nodes. Typically much less is required.
- For existing multiplex configurations, take data points for peak local temporary storage use (sp_iqstatus during a typical query workload)
- A good starting point is to provide shared temp storage equaling 1/2 the sum of the peak local temporary storage for each node

Logical Servers

A logical server allows one or more servers of a multiplex to be grouped together and represented as a logical entity. Users are granted access to logical servers via the login policy associated with the user.

All member nodes of a logical server are available for a Distributed Query. The Sybase IQ query engine determines if a query can be distributed. Once a query is deemed distributable, all spare CPU capacity available across the member nodes of the logical server is harnessed by the query engine.

Out of the box, Sybase IQ 15.3 comes with built-in OPEN logical server that includes all servers that are not members of any user-defined logical server. If there are no user-defined servers, all nodes in the multiplex may participate in DQP, because they are part of the OPEN server.

This concept allows planning of the entire logical server cluster as an entity instead of planning for peak load in each node. Significant savings in CPU costs can be achieved if it is possible to combine nodes with varying peak loads in a logical server. Since Sybase IQ 15.3 allows a login policy level of node/logical server assignment, more nuanced planning for CPU configurations can be achieved.

Multiplex Interprocess Communication (MIPC) connects all multiplex nodes to support distributed query processing and high availability. MIPC is a fully meshed communication framework. It runs on both public and private interconnection configurations. Public interconnection configuration is mandatory while private configuration is optional.

Private high-speed interconnection configurations are for distributed query processing. Currently, private interconnection configurations are restricted to physical networks supporting the TCP/IP protocol. Using a private interconnect helps maintain SLA as inter node DQP communication will not be interfered with by public network traffic, and also helps with HA as private interconnect failure will fail over to public network.

In order to achieve optimum performance, it is recommended that a minimum of Gigabit Ethernet bandwidth network be used. Higher speed networks (10g or more) will reduce communication overhead and may provide a performance boost for DQP queries with large intermediate result sets.

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.

General Hardware Setup

CPUs	At least 25% fewer than the existing DW with no CPU-per-TB ratio required
What type of server hardware	Windows (Intel/AMD), Linux (Intel/AMD/IBM), AIX, HP-UX (PA-RISC/Itanium), Solaris (Sparc/AMD)
What type of storage hardware	Any current hardware that supports fiber channel or SATA drives with built in RAID and caching
RAID level	RAID 1 or RAID 1+0 offer the best performance while RAID 5 offers the best cost for Sybase IQ storage
Volume management	None needed for IQ main and temporary devices
Cluster and HA software	None needed, but can be used to extend high availability and disaster recovery

Sybase IQ Storage Sizes

DW size (data and index)	20-70% smaller than the input data
Raw storage size	Sybase IQ DW size + 15% overhead for RAID 5 or 100% overhead for RAID mirroring
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)
Database case sensitivity	Recommend using "Case Respect"
Swap space	1-2x physical RAM
General memory guideline	4-8 GB per CPU core

CPUs

CPUs for queries	0.1-1.5 CPUs per query
CPUs for loads	0.05-0.5 CPUs per index and column, though the HG, WD, and TEXT indexes are fully parallel and can consume all CPUs on a system

Memory

Total Sybase IQ memory for all operations	Total RAM less 10-20% for the OS
Main and temporary cache during queries	40% main cache, 60% temporary cache of remaining RAM
Main cache during loads	5-10 pages per index and column
Temporary cache during loads	Each HG: (8 + sizeof(datatype)) * numberRowsBeingLoaded Each WD: (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
Backup memory	TmpVal = max(2 * numberOfCpus, 8 * numberOfMainLocalSpaces) Memory = (TmpVal * 20) * (block factor * block_size)
Catalog Cache (-c or -cl/-ch)	2-16x the size of the catalog (.db) file

Storage and I/O

Required I/O bandwidth per CPU core	20-40 MB/sec
Total bandwidth per server	#cores * 20-30 MB/s
Minimum number of physical fiber channel disk drives per CPU core	0.3-1 drives per core Calculate separately for main and for temp
Minimum number of physical SAS disk drives per CPU core	0.5-3 drives per core Calculate separately for main and for temp
Minimum number of physical SATA disk drives per CPU core	2-5 drives per core Calculate separately for main and for temp
Maximum number of CPU cores per fiber controller for main	4-8 cores per fiber controller Calculate separately for main and for temp
Number of Sybase IQ main store LUNs	3 + (#cores / 10) (rounded up)
Number of Sybase IQ temporary store LUNS	3 + (#cores / 10) (rounded up)

OVERVIEW OF SYBASE IQ MEMORY MANAGEMENT

It is important to understand exactly how Sybase IQ uses memory that it was given during startup as well as how Sybase 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-Sybase IQ Memory

First, you must assess what is running on the machine and how much memory it uses (exclusive of Sybase IQ). Typically these include:

- Operating System
- OLAP Servers
- Middleware
- Other applications
- Monitoring applications
- Shells, scripts, etc

Once this assessment has been done, deduct that amount of memory from the total RAM on the machine. This will leave you with the amount of free RAM remaining. At this point, we must carve out memory for the operating system. Filesystem cache is generally set to 20% on most platforms. This amount should be deducted from the available RAM. Also, the OS will use RAM for the kernel and its basic operations. A rough rule of thumb is 500 MB to 1 GB of RAM for the OS. The remaining amount of RAM is what is left over for Sybase IQ.

Sybase IQ Memory

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

Sybase 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)
- Version memory
- Memory used during backups

Server Configuration

When the Sybase 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
- **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

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 Sybase IQ indexes. Temporary cache is that transient, volatile area of memory. Temporary cache is used to house temporary tables, internal work tables, 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, the next step is to compute the ratio between main and temporary cache. For typical systems a starting point is to split the available RAM for caches and allocate 40% of that for main cache and 60% for temporary cache.

With the advent of the 3-byte FP index, main cache plays a role in the use of that index type. As the main cache size is increased (see below), Sybase IQ will allow more distinct values to be represented in the 3-byte FP structures. As such, it is often advised to increase the main cache use from 40% to 50% and decrease the temp cache use to 50% while the initial data is being loaded into the tables.

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 keeping in mind, though, that lowering the main cache size can restrict the use of the 3-byte FP index.

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 Sybase IQ must make to the main store to retrieve user data. The exception to this is those environments where a considerable amount of temporary tables, group bys, or order bys 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.

Optimized Fast Projection Index Main Cache Memory Use

In Sybase IQ 12 when an optimized Fast Projection (FP) index was used, the index would follow the basic rule that a 1-byte FP (FP1) would be converted to a 2-byte FP (FP2) and finally to a flat style FP should the cardinality increase and cross the 1-byte and 2-byte FP index boundaries of 255 and 64k, respectively.

Sybase IQ 15 changes this behavior due to the addition of an optimized 3-byte FP index.

Two new options were added to control the amount of main cache to be used for storing the lookup pages for the optimized FP indexes: FP_LOOKUP_SIZE and FP_LOOKUP_SIZE_PPM. FP_LOOKUP_SIZE controls the absolute amount of RAM (16 MB by default) that the lookup pages can consume while FP_LOOKUP_SIZE_PPM (2500 parts per million by default) sets the same limit based up a percentage of the main cache.

The maximum number of lookup pages used in Sybase IQ is controlled by the FP_LOOKUP_SIZE option and the FP_LOOKUP_SIZE_PPM option, whichever value is smaller. When the threshold is crossed for any of the optimized FP indexes, that index will automatically be converted to a flat FP structure.

To determine the amount of RAM needed to store an optimized FP index we can use the following algorithm:

RAM_MB –> ((Num_Distinct_Values * (Column_Data_Size + Cardinality_Size))
/ 1024 / 1024)

- Column_Data_Size is the binary width of the data type (4 bytes for integer, 10 bytes for char(10), etc)
- Cardinality_Size is either 4 or 8 bytes and stores statistics on the number of distinct values per column. The safe assumption is that 8 bytes will always be used per column.

To illustrate with a few examples:

- Assume that the datatype is an integer with 250,000 distinct values:
  – (250,000 * (4 + 8)) / 1024 / 1024 == 2.86 MB
- Assume that the datatype is an integer with 2,000,000 distinct values:
  – (2,000,000 * (4 + 8)) / 1024 / 1024 == 22.88 MB
- Assume that the datatype is a varchar(50) with 250,000 distinct values:
  – (250,000 * (50 + 8)) / 1024 / 1024 == 13.82 MB
- Assume that the datatype is a varchar(50) with 2,000,000 distinct values:
  – (2,000,000 * (50 + 8)) / 1024 / 1024 == 100.62 MB

As we can see, the wider datatypes will soon exceed the default values for FP_LOOKUP_SIZE and FP_LOOKUP_SIZE_PPM. With a default, at most, of 16 MB for the FP memory structure, a table with 2 million distinct integers will need 22 MB and thus convert from an optimized 3-byte FP to a flat FP.

Per the Sybase IQ Administration Manual, Volume 2, the table below illustrates the maximum number of distinct values per column width with respect to various size settings for FP_LOOKUP_SIZE. The table assumes that an 8 byte field will be required for the Cardinality_Size. The table also follows the following algorithm:

Cardinality –> FP_LOOKUP_SIZE / (Column_Data_Size + Cardinality_Size)

- FP_LOOKUP_SIZE is converted to bytes

Table 1 - Maximum unique values in and FP(3) index

	COLUMN DATA TYPE WIDTH (BYTES)					
FP_LOOKUP_SIZE	**4**	**8**	**32**	**64**	**128**	**255**
1 MB	87381	65536	26214	14563	7710	3986
4 MB	349525	262144	104857	58254	30840	15947
8 MB	699050	524288	209715	116508	61680	31895
16 MB	1398101	1048576	419430	233016	123361	63791
32 MB	2796202	2097152	838860	466033	246723	127583
64 MB	5592405	4194304	1677721	932067	493447	255166
128 MB	11184810	8388608	3355443	1864135	986895	510333
256 MB	16777216	16777216	6710886	3728270	1973790	1020667

Query Tuning via Max_Hash_Rows

Currently, Sybase 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 into it:

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

This recommendation may not always yield performance gain. It is advised that setting for this value be tested for performance varying the weight of the 2.5 million factor.

Catalog Cache

The catalog cache is a portion of memory that is reserved for work that must take place outside of the Sybase IQ environment. The catalog is a portion of the Sybase IQ engine that is managed and maintained by the Sybase SQL Anywhere® technology (ASA or SA). The catalog handles any operations that do not reference Sybase 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 Sybase IQ or Sybase SQL Anywhere. An undersized catalog cache can lead to poor performance or even client errors due to resource constraints.

In the Sybase 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 Sybase IQ start script as -gm + 5. With -gm set to 100, the value of gn would default to 105.

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 Sybase 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 Sybase 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 Sybase 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.

Sybase IQ will allocate one page for each FP 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 Sybase 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 Sybase 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.

An LF index is an example of a bitmapped index. For an LF, 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 load 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, Sybase 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 Btree 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 Sybase IQ has allocated for all memory structures (caches, load memory, 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: increase the RAM on the machine, lower the load memory limit, or decrease the main and temporary cache sizes.

Backup Memory

In the ideal situation, the amount of virtual memory used by a backup process is a function of the following items:

- number of CPUs
- number of main dbfiles to be backed up
- block factor
- IQ block size (as seen in column 'block_size' in sys.sysiqinfo)

Here's how you would roughly calculate the amount of virtual memory required.

y = max(2 * number_of_cpus, 8 * number_of_main_dbfiles)
z = (y * 20) * (block factor * block_size)

'z' represents a rough estimate on the amount of virtual memory used during the backup operation. For example, if the system has:

- dbfiles = 50
- block factor = 100
- number of CPUs = 4
- block_size = 8,192

Using the above assumptions and the previous algorithm, the amount of RAM needed during the backup operation would be:

'y' is max(8, 400) **–>** y=400
'z' is (400 * 20) * (100 * 8,192) **–>** 6.5GB

This memory comes entirely from the OS: it is heap memory and is released only when the operation completes.

The only mechanism to control this is 'block factor' which is sufficient. In the previous example, if 'block factor' were changed to 10, then the memory usage drops to:

(400 * 20) * (10 * 8,192) **–>** 655MB

Lowering the memory usage can make the backup run slower as it won't have nearly as much RAM to operate in; assuming that disk I/O is not a bottleneck. The backup runs at the rate at which the disk I/O subsystem can read and write blocks. To reduce I/O overhead, the backup attempts to read contiguous blocks in one I/O, and correspondingly writes them out a 'block factor' units.

The choice at hand when calculating the amount of RAM needed for backups is to use the block factor to reduce the memory or increase the memory available. Both have their trade-offs (cost vs. performance) and the correct path can only be made with those trade-offs in mind.

The Complete Memory Picture

For each concurrent load that will be run, compute the amount of load memory necessary and add the values up. This should be subtracted from the total RAM that Sybase IQ can use. The remaining memory is what can be allocated to the IQ caches.

Operating System	*.5 to 1 GB RAM*
Filesystem Cache	*20% of RAM*
All Other Applications	
IQ Catalog Memory	*-c/-cl/-ch parameters*
IQ Thread Memory	*stack size * thread count*
Bitmap Memory	*per concurrent load*
IQ Main Cache	*40-50% of remaining RAM*
IQ Temporary Cache	*50-60% of remaining RAM*
Backup Memory	*per backup instance*

How Much RAM for Sybase IQ?

There are two approaches to sizing RAM for Sybase 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, Sybase 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).

Example

For example, consider two different systems: one with 16 GB RAM and another with 32 GB RAM. For the sake of this exercise, it is assumed that there will be five concurrent load processes and that load memory has been set to 200 MB. It is also assumed that the loads won't need a considerable amount of bitmap memory for their processing.

	16 GB SYSTEM	32 GB SYSTEM
Operating System	1 GB	1 GB
Filesystem Cache	1 GB	2 GB
All Other Applications	500 MB	500 MB
IQ Catalog Memory	128 MB	256 MB
IQ Thread Memory	250 MB	500 MB
Bitmap Memory	256 MB	256 MB
IQ Main Cache	4.9 GB	10.9 GB
IQ Temporary Cache	7.1 GB	16.1 GB
Backup Memory	N/A	N/A
Total	**15.125 GB**	**31.5 GB**

OVERVIEW OF SYBASE IQ DISK FUNDAMENTALS

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

Main store 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).

Read and Write Operations

Sybase 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 Sybase 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 Sybase 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.

Sybase 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.

IQ_SYSTEM_MAIN

In Sybase IQ 12.7 we talked in terms of dbspaces. A dbspace was a flat file or raw device. There was a 1 to 1 mapping.

In Sybase IQ 15, that has changed. A dbspace is a logical storage container for data. This new dbspace will be made up of one or more flat files or raw devices. In Sybase IQ 15 terms, a dbspace is made up of files. Files (sometimes referred to as dbfiles) are the physical storage, while a dbspace is the logical store. In Sybase IQ, a file is a filesystem based file or a raw device.

Sybase IQ 15 only allows 1 temp dbspace. This dbspace can be made up of 1 or more files (flat file or raw device). To add a file to a dbspace in Sybase IQ, you would use the ALTER DBPSACE command to add more storage to the IQ temporary store.

The meaning of the term ("dbspace") varies according to the product version you are using. Sybase IQ 12.7 implemented one-to-one mapping between a dbspace and a database file. With DEFAULT_DISK_STRIPING option 'ON', Sybase IQ automatically distributed data across all available dbfiles for optimal performance and ease of administration. Users could not specify in which dbspace a table or index should be created.

The term file, with a corresponding logical filename and physical file path, refers to each operating system file to be used to store data.

Each dbspace name, file name, and physical file path must be unique. The file name can be the same as the dbspace name.

A store is one or more dbspaces that store persistent or temporary data for a special purpose. Sybase IQ has three types of data stores:

- The catalog store contains the SYSTEM dbspace and up to twelve additional catalog dbspaces.
- The IQ main store contains the IQ_SYSTEM_MAIN dbspace and other user dbspaces.
- The IQ temporary store contains the IQ_SYSTEM_TEMP dbspace.

It is highly recommended that IQ_SYSTEM_MAIN NOT be used for user data. Sybase 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 Sybase ASE: something that should never be used to store user data. In Sybase 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. Sybase IQ should be no different beginning with version 15.

In the new 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.

SIZING CPUS, CORES, AND PROCESSORS

Data Loads and Changes

Single Row Data Changes

Single row data changes in Sybase 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 Sybase IQ and should only be used if performance is not a concern.

Bulk Load Operations

Bulk load sizing encompasses any Sybase IQ loads that are done via the *load table, insert from location*, or *insert select* syntax. These mechanisms invoke the Sybase 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 Sybase IQ bulk loader.

Sizing a Sybase IQ system for loads is straightforward. The CPU resources necessary to perform a single load, without impacting load performance, can be directly measured by the number of indexes on a table.

A rough guideline for CPU sizing, assuming that the load operation should occur without contention, is as follows:

- 1 CPU for every 5-10 columns in the table being loaded (default FP index)
- 1 CPU for every 5-10 indexes (HNG, LF, CMP, DATE, TIME, DTTM) on the table that have not been mentioned
- The HG, WD, and TEXT indexes are all massively parallel and can consume all CPU resources on the host to improve performance. A rough starting point would be to size the system such that it has 1 core for every 1-2 HG, WD, or TEXT indexes. Once a load pattern is established, the CPU resources can be adjusted up or down to meet business needs.

It is important to note that the above guideline is for sizing a system in which the loads will utilize nearly 100% of the source system and run without contention. This should be balanced against the timeframe in which loads have to run.

Another algorithm used to size a system is based on the amount of raw data to be loaded into Sybase IQ. This approach focuses more on loading the data in a known time, rather than sizing the system for all out performance.

- For systems with 4 or fewer CPUs, expect to load roughly 10-15 GB of data per hour per CPU. A 4 CPU system should be able to load about 40 GB of raw data per hour.
- For systems with 8 or more CPUs, expect a load rate of 15-40 GB per hour per CPU. An 8 CPU system should be able to load between 120 and 200 GB of raw data per hour.
- Load times with this approach will vary greatly based on CPU count and the number and types of indexes on the table being loaded.

Sybase IQ also supports unstructured (binary and character large objects, BLOB and CLOB) data. When loading BLOB and CLOB data into Sybase IQ, load performance will be directly attributed to the number of CPUs as well as the performance of the disk subsystem (storage array, number of spindles, host bus adapters, etc) on the system.

For each BLOB or CLOB being loaded into Sybase IQ a single CPU will be necessary. A system with 8 CPUs will be roughly twice as fast as a system with 4 CPUs when loading the same BLOB and CLOB data. Sizing BLOB and CLOB data will depend squarely on the performance expectations of the customer for these datatypes.

With the advent of Sybase IQ 15 all load operations are run in parallel; this include pass 2 of the HG and WD index loading process. Above and beyond the sizing guidelines, the HG index load performance can be significantly improved by adding more cores to the write node. This will allow Sybase IQ to greatly increase the parallelism of the HG index work and thus reduce load times.

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 generality, 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

A caveat to this method is parallel execution. If there are CPUs available and the query can be broken into smaller execution pieces the optimizer has the ability to break the serial execution up into many parallel processes. A query will perform quite differently on a system with 4 CPUs, 8 CPUs and 12 CPUs.

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 Sybase IQ to perform the queries will be altered.

As Sybase IQ 15 continues to be enhanced more and more query operations are run in parallel. While the above guidelines are a good starting point for system sizing, it is important to keep in mind that a single user can benefit from having more cores available.

The parallelism that came into being with Sybase IQ 15 attempts 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.

Please note that the Distributed Query Processing feature introduced in the v15.3 release does not require any additional resources to be configured.

SIZING MEMORY

Data Loads and Changes

Single Row Operations

In general, single row operations in Sybase 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.

Bulk Operations

Generally, having more RAM 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.

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. The following algorithm would apply to all columns that contain an HG or WD index that are being loaded:

total_pages = 1 + ((number_of_rows_in_load_files * width_of_columns_in_hg)
/ page_size_in_bytes)

number_of_rows_in_load_files – The total number of rows that are being loaded via the LOAD TABLE or INSERT command

width_of_columns_in_hg – This is the total binary width of all columns in the HG index. For HG indexes (primary key, unique constraint, foreign key, etc) that support multiple columns, the binary width of all columns must be taken into account.

page_size_in_bytes – This is the page size, in bytes, that the database was created with

As an example, let's assume that a load of 10 million rows is taking place on a table that contains a single HG index on an integer column and that the database was created with a 256K page. The total temporary cache necessary for this load would be:

total_pages = 1 + ((10,000,000 * 4) / 262,144)
total_pages = 1 + (40,000,000 / 262,144)
total_pages = 153 (256KB) pages or 38 MB

If that same table and load had a primary key with an integer and char(10) column, the memory necessary for that index would be:

total_pages = 1 + ((10,000,000 * (4 + 10)) / 262,144)
total_pages = 1 + (140,000,000 / 262,144)
total_pages = 535 (256KB) pages or 134 MB

To size main cache, one must consider the index types in detail that exist on the table being loaded.

- FP –> 1 page for each FP index (plus 3 in temp cache for each optimized FP)
- LF –> 1 page for each distinct value currently being loaded into the LF index
- HNG, DATE, TIME, and DTTM –> 1 page for each bit in the bitmap
- CMP –> 3 per index
- HG and WD –> Reliant on temporary cache during the first pass (see above) and the main cache during the second pass to build the final page structures. There is minimal main cache needed for HG and WD indexes due to the reliance on temporary cache.

A rough rule of thumb is 5-10 pages of main cache need to be allocated for each index on the table being loaded (this includes all index types, including the default FP index). This would have to be tuned if there are a significant number of distinct values being loaded, as this is just a rough estimate to be used as a starting point.

For a table with 50 columns and 25 additional indexes, this would equate to:

(50 FP indexes + 25 additional indexes) * (5,10) –> 375-750 pages
375 pages * 128K page size –> 46.875 MB
750 pages * 128K page size –> 93.75 MB

Queries

When sizing memory for queries, the general rule of thumb is that the system should be configured for a minimum of 4-8 GB of RAM per CPU. For smaller systems (less than 8 CPUs) it is recommended that the system be sized closer to 8 GB RAM while larger systems be sized in the range of 4-8 GB per core.

Multi-host Configuration

Sybase IQ has the ability to be run on multiple hosts that share the same main disk space. This is often referred to as Sybase 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 is relatively straightforward and based on the size of the user defined dbspace files, in total, as well as the number of nodes in the multiplex.

For databases that are less than 100 GB, it is recommended that IQ_SYSTEM_MAIN be at least 4 GB in size, and typically sized at 5-10% of the user defined main space size (5-10 GB for a 100 GB database). If this instance is migrated to a multiplex, an additional 1 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 7-12 GB in size.

For databases that exceed 100 GB, it is recommended that IQ_SYSTEM_MAIN be at least 8 GB for a simplex and 16 GB for a multiplex. IQ_SYSTEM_MAIN would typically be sized at 1-2% of the user defined main space size (10-20 GB for a 1 TB database). If this instance is migrated to a multiplex, an additional 0.1-0.3% (1-3 GB per 1 TB) of space should be added per node in the multiplex. For a 4 node system with a 1 TB database, the size would be the 16 GB minimum plus 1-3 GB per node (4 nodes) in the multiplex; or 20-28 GB.

General Guidelines

Disk sizing requirements for Sybase IQ change as the CPU 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. Sybase 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.

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

Each CPU can process (on average) 20-30 MB/sec of data from Sybase IQ. As a rule, design the disk farm to be able to deliver 20-30 MB/sec to each CPU in multiplex. The disk bandwidth, in MB/second, should equal 25 * the number of CPUs in the multiplex.

Using an LVM is not a requirement for Sybase IQ to operate and generally provides no value add for Sybase IQ (this is not to say that an LVM does not have operational benefits outside of Sybase, 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, Sybase IQ 15.1 ESD 3 and later, direct I/O has been implemented for those sites that cannot use raw devices for IQ devices. Proper configuration of the filesystems must be done to ensure that Sybase 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.

When choosing disks for Sybase IQ, it is recommended that larger (146GB, 300GB, 500GB, 750GB and larger) disks be used without concern for the speed of the disks. The lower RPM (i.e. 7200) ratings of the newer, large capacity disks has been tested with Sybase and shown a marginal impact on performance. The result is that Sybase IQ is not dependent on high RPM (i.e. 10K and 15K rpm) disk drives. The lower latency, larger disks can be used and will result in lower cost per TB for the Sybase IQ implementation. Of course, should smaller disks already be in place, it is OK to reuse and redeploy those drives to the Sybase IQ environment.

The following sections outline an algorithm to be used to compute the number of devices that should be allocated for the IQ main and temporary stores. These algorithms should serve as a starting point. Systems that can expect heavy load and/or query use should use higher numbers than those computed below. Also, systems that plan on load data into the Large Object structures of Sybase IQ (BLOB and CLOB datatypes) should increase these numbers as well. The increases come as a result of the increase in I/O bandwidth necessary for handling these tasks optimally.

Storage Stripe Size, Stripe Width, and Block Sizes

When Sybase IQ writes to disk, the goal is to optimize the write process so that Sybase IQ writes to as many disks, simultaneously, without forcing contention on those disks. To achieve this, the stripe size, stripe width, and block sizes at the storage subsystem need to be configured properly.

For the purposes of sizing the storage and speaking in terms that the storage administrators should be familiar with, we will define the following terms. These terms, while similar to those in Sybase IQ, differ in definition.

- **Stripe width** – The number of physical disks that comprise the RAID group for an IQ dbfiles. A RAID 5 in a 4+1 configuration would have a width of 4. 8 disks in a RAID 0+1 configuration would have a width of 4 (4 primary and 4 mirrored).
- **Block size** – The amount of data that is written to the entire RAID group as a single operation. The goal is to have this match the IQ page size as closely as possible.
- **Stripe size** – The amount of data written to each disk in the RAID group

For the purposes of Sybase IQ implementations we don't typically care what the stripe width is for the RAID group so long as we meet the minimum requirements for the total number of disks/spindles that Sybase IQ requires for optimal performance.

With most modern storage systems it is recommended that the 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.

If a larger stripe size can be used, Sybase IQ does offer the ability to write to disk in sizes that are larger than the page size. By default, IQ writes pages in a single I/O. The option DEFAULT_KB_PER_STRIPE (with a default value of 1k) guarantees that whatever the page size chosen, Sybase IQ will write the next I/O to the next IQ file in that dbspace. A single page is not broken into blocks or smaller sizes.

Contrary to the value, the default setting is not too low. When Sybase IQ writes, 1 page is compressed and that compressed page is written to disk as a single I/O. Sybase IQ will not break the page or I/O across files.

The DEFAULT_KB_PER_STRIPE option only means that the subsequent writes, up to this total, are made to the same file.

The real sizing question is how large do we want to set this so that we guarantee that Sybase IQ will write to a single file before moving to the next.

We could set this to 256K. Sybase IQ will then write pages to a dbfile up to a maximum of 256k (compressed to blocks) before moving on to the next file. For highly volatile systems with disks optimized for very large writes, DEFAULT_KB_PER_STRIPE can be increased in increments of the IQ page size to determine the best possible value. The option is dynamic and takes effect for all writes after the option has been set.

For RAID 5 storage, the stripe width should be one less than the total drives in the RAID group. This will help compensate for the RAID 5 parity disk and the writing that takes place there.

The block size should be equal to the Sybase IQ page size that the database was created with. Some systems may not support block sizes that large. In those situations, choose a block size of 64K or larger.

To illustrate this writing process, consider the following diagram. Sybase IQ will perform a write to disk. During this operation, the SAN or OS may or may not break the single Sybase IQ I/O operation into smaller chunks. Regardless, the block is then broken up into striped chunks to be written to the individual disks.

For the two most common RAID types that Sybase IQ runs on, we will illustrate RAID 5 and RAID 0+1.

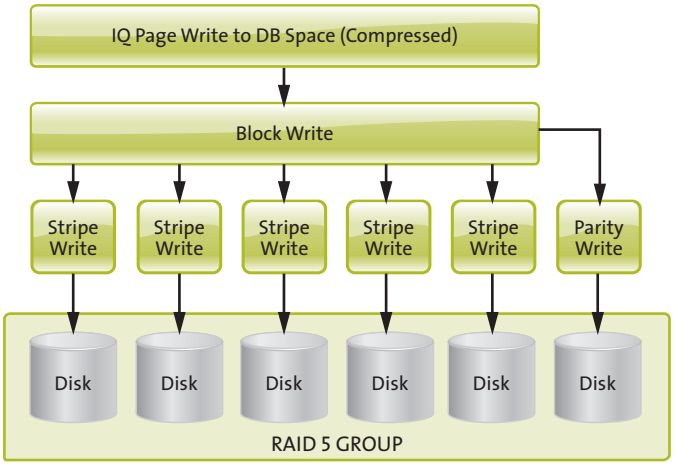


Figure 1 – RAID 5 (4+1) Example

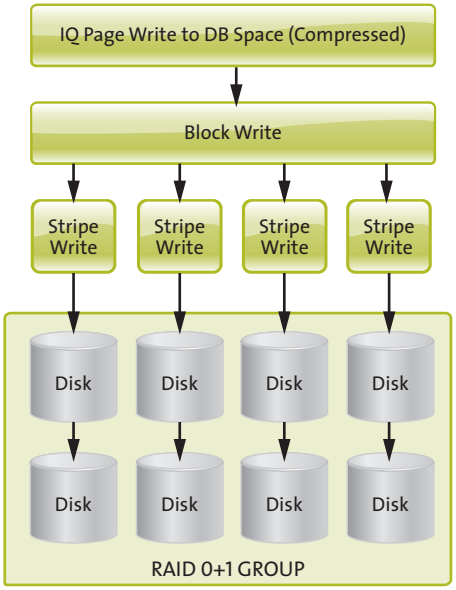


Figure 2 – RAID 0+1 (4 disk) Example

Physical Drives and Device Controllers

When using fiber channel drives, there should be a minimum 0.3-1 hard drives for every CPU core in the single or multi-host configuration. Using SAS drives would increase this ratio to 0.5-3 SAS drives per core in the single or multi-host configuration. Using SATA drives would increase this ratio to 2-5 SATA drives for every CPU core in the single or multi-host configuration. This number should be rounded up to the next highest whole number.

For systems that expect heavy use the above ratios should be increased by 50-100% to handle the increased disk activity by additional users or atypically disk intensive operations.

As an example, 20 cores would require 7-20 fiber disks, 10-50 SAS disks, or 40-100 SATA disks. In an I/O intensive system, the number of physical drives would increase to 10-40 fiber disks, 15-100 SAS disks, or 60-200 SATA disks.

When sizing the physical drives for IQ, main store and temporary store devices follow the same algorithms. However, main and temporary store disks should be sized separately from one another for optimal performance and load segregation.

In order to control the disk drives, disk controllers (fiber controllers, host bus adapters, HBA) will be needed. There should be one disk controller for every five to 10 CPUs. On heavier used systems more controllers should be added to accommodate the load.

The storage system that will house the drives and the connectivity from the host(s) to the storage system will also affect Sybase IQ performance. It is not enough to have a lot of drives and a lot of disk controllers if the path from the machine to the storage or the storage itself does not have enough bandwidth to support Sybase IQ.

As a rule of thumb, the total bandwidth that the storage system needs to support should follow this algorithm:

number_of_total_cores * 20 MB/sec

Number_of_total_cores is the total number of cores for the entire Sybase IQ system. For a single host system, this is the number of CPUs on that host. For a multi-host system, this is the sum of all CPUs on all Sybase IQ hosts. For systems that expect heavy use, the 20-40 MB/sec number should be increased accordingly.

When sizing the controllers for IQ main store and temporary store controllers follow the same algorithms. However, main and temporary store controllers and disk paths should be sized separately from one another for optimal performance and load segregation.

With implementation of DQP in Sybase IQ 15.3, Sybase IQ can now handle throughput in excess of 100-200 mb/sec per CPU. Experiment with higher IO bandwidth in case your application needs extreme IO at peak loads.

Devices for a Single Node Implementation

The current rule of thumb for determining the number of main and temporary store IQ devices (dbspaces) to allocate for a single host Sybase IQ configuration should follow this algorithm:

IQ_dbfiles = 3 + (number_of_singlehost_cores / 10)

This number should be rounded up to the next highest whole number and calculated separately for main store and temporary store devices.

For example, a 4 core/CPU machine would need 3 + (4/10) or 3.4 devices. Rounded up, this system should have a minimum of 4 devices for the main store and 4 devices for the temporary store.

Devices for a Multi Node Implementation

For a multi-host IQ configuration, all CPUs/cores in the multiplex should be taken into account for the number of main store devices and this algorithm should be used:

- Add up the total CPUs on all machines
- Feed this value into the above algorithm like this:

IQ_dbfiles = 3 + (number_of_multihost_cores / 10)

For example, if there are 6 machines with 4 CPUs on each machine there would be 24 total CPUs for the multiplex. That would equate to 6 main store devices (3 + (24/10)).

Temporary store devices would follow the same algorithm as in a single node implementation. Each node would have its complement of temporary store devices that is based on the number of CPUs on the individual node.

IQ_dbfiles = 3 + (number_of_singlehost_cores / 10)

This number should be rounded up to the next highest whole number.

If there were 4 machines, each with 4 CPUs, then each machine would require 4 temporary store devices, or a grand total of 16 temporary store devices for all nodes.

IQ Device Placement

It is recommended, but not mandatory, that the Sybase 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 Sybase IQ device types.

For additional information on configuring storage devices, LUNs and, a SAN, search for "IQ Reference Architecture Sizing Guide", "Reference Architecture Implementation Guide", and "NonStopIQ" on the www.sybase.com website search box.

IQ Device Mapping

When developing the Sybase IQ storage subsystem, it is important to size the entire storage system appropriately. It is also equally important for 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 SAN infrastructure. At the SAN, the physical disks or spindles should also not be shared with any other application.

As an example, the following is a high level conceptual diagram of how Sybase IQ storage could be laid out. The Sybase IQ main store is comprised for 4 different RAID 5 groupings. Each RAID 5 grouping is a 5 disk RAID 5 (4+1 parity drive) configuration. The entire RAID group is presented as a single device to any machine connected to the SAN so that they see the entire storage content of those 4 disks (plus 1 for parity).

The RAID 5 "disk" is presented to the OS as a LUN. That LUN is, in turn, used by IQ as a single dbfile. A logical volume manager (LVM) need not be present. If one is, it is the job of the LVM to present the original LUN to Sybase IQ as a single disk that hasn't been striped together with other devices.

It is also important to keep in mind that in a multi-node Sybase IQ configuration multiple hosts will need to see the same IQ main store devices simultaneously. Adding ports and bandwidth to the SAN and SAN switches will be critical to performance as more nodes are added.

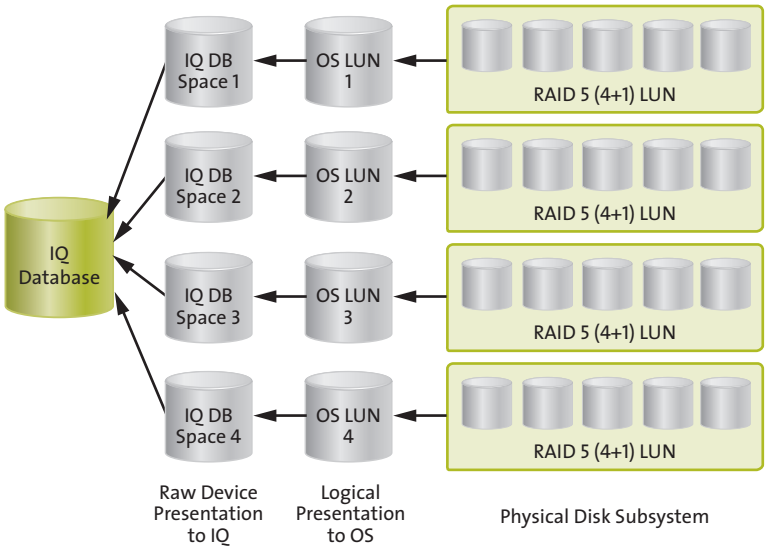


Figure 3 – IQ Main Store Drive Mapping per Multiplex

The disk layout and presentation is not much different for an IQ temporary store. The above example of 4 RAID 5 groupings of 5 disks (4 + 1 parity) will be carried through for the IQ temporary store device example.

The main difference, though, is that the IQ temporary store devices are dedicated to a single host while the main store devices are shared amongst all nodes. It is just as important, though, to make sure that the temporary store devices for each Sybase IQ server are not sharing disk/spindle resources with any other Sybase IQ instance or non-Sybase IQ application.

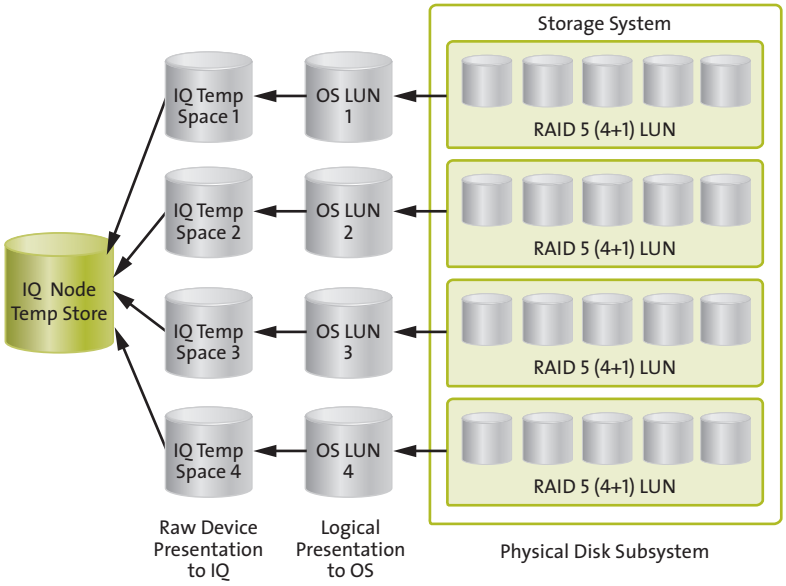


Figure 4 – IQ Temporary Store Drive Mapping per Node

SIZING NETWORK

Performance

There is very little node to node communication in an Sybase IQ multiplex: so little as to not be a major consideration when configuring networking on the Sybase IQ hosts. There are two areas, though, where network topology and performance do matter:

- Queries
- Remote data loads (insert...location syntax)

During queries and remote data loads, the data retrieval from the client will be as fast as the network. For example, 100 Mbytes of data will take:

- 80 seconds on a 10 Megabit LAN
- 8 seconds on a 100 Megabit LAN
- 0.8 seconds on a Gigabit LAN

LAN speed may be a performance bottleneck for queries that return large amounts of data or loads that bring in large amounts of data from remote servers. The faster the network cards and LAN the better off concurrency will be: more available bandwidth per operation.

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

- Use faster network cards
- Use a faster network topology (1 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.

SYBASE IQ PAGE SIZES

Sybase 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), Sybase 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 rowcount 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 *Sybase 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 Sybase 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 Sybase 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 rowcount 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

Startup Thread Allocation

The default number of threads allocated in Sybase 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:

60*(min(numCores,4)) + 50*(numCores - 4) + 2*(numConnections + 2) + 1

On a 4 core system with -gm (numConnections) set to 20, that would be:

iqmt = 60 * (4) + 50 * (4-4) + 2 * (20+2) + 1
iqmt = 285

On a 12 core system with -gm (numConnections) set to 50, that would be:

iqmt = 60 * (4) + 50 * (12-4) + 2 * (50+2) + 1
iqmt = 745

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: 2*(numConnections + 2). 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" all that are left are those kept back for connection use with 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, Sybase 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 a theoretical upper limit of 4096 threads for a 64-bit system. Some platforms will not boot with the max limit of 4096. 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 teams of threads that perform disk I/O: sweeper and prefetchers. The sweepers write out dirty buffers and prefetchers read in buffers.

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 Sybase 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.

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. Sybase 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.

Sybase, Inc.
Worldwide Headquarters
One Sybase Drive
Dublin, CA 94568-7902
U.S.A
1 800 8 sybase

www.sybase.com

SYBASE® An SAP® Company