



System Resource Planning and Management

Version 2025.1
2025-06-03

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™, HealthShare® Health Connect Cloud™, InterSystems® Data Fabric Studio™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Congress Street, Boston, MA 02114, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

Table of Contents

System Resource Planning and Management	1
1 Memory Planning	1
1.1 Shared Memory Allocations	2
1.2 Configuring Huge and Large Pages	3
1.3 System Requirements for Parallel Dejournaling	5
2 Storage Planning	6
2.1 Storage Configuration	7
2.2 Storage Connectivity	8
2.3 File System Separation	9
2.4 Buffered I/O vs. Direct I/O	9
2.5 noatime Mount Option	10
3 CPU Sizing and Scaling for InterSystems IRIS	10
3.1 Basic CPU Sizing	11
3.2 Balancing Core Count and Speed	11
3.3 Virtualization Considerations for CPU	11
3.4 Leveraging Core Count with Parallel Query Execution	12
4 Configuring InterSystems IRIS Shared Memory	12
5 Reviewing and Monitoring Memory Usage	13
6 Data Compression and Reducing Storage Costs	14
 List of Figures	
Figure 1: Parallel Query Execution	12

System Resource Planning and Management

When developing and running an application on InterSystems IRIS, you should consider the system resources that you have at your disposal and manage them to optimize the performance of your system. This page can help you to estimate the resources needed for a given workload or to optimize your available resources for your workload.

Before deploying your instance you can use the following sections to estimate the resources required for optimal performance:

- [Memory Planning](#)
- [Storage Planning](#)
- [CPU Sizing and Scaling for InterSystems IRIS](#)

The following sections describe the process of configuring shared memory, either during or after deployment:

- [Configuring InterSystems IRIS Shared Memory](#)

You can optimize performance by regularly monitoring the usage of your resources as described in the following sections:

- [Reviewing and Monitoring Memory Usage](#)
- [Data Compression and Reducing Storage Costs](#)

1 Memory Planning

This section provides guidelines on planning system memory resources. It also discusses specific workload scenarios that may require extra attention to memory resources. You can use these guidelines to assist in correctly allocating resources.

The goal of memory planning is to provide enough memory on each host for all of the entities running on the host under all normal operating circumstances. This is a critical factor in both performance and availability. Once you've [determined shared memory allocations](#) you should [configure shared memory](#) accordingly either during or after deployment. After deployment, you should regularly [review and monitor actual memory usage](#) in operation and make adjustments as needed.

Important: InterSystems products do not support memory ballooning nor over-provisioning. InterSystems products require 100% memory reservation for all virtualization and containerization platforms.

If you have not configured sufficient physical memory on a Linux system and thus regularly come close to capacity, you run the risk that the out of memory killer may misidentify long-running InterSystems IRIS processes that touch a lot of memory in normal operation, such as the write daemon and CSP server processes, as the source of the problem and terminate them. *This will result in an outage of the InterSystems IRIS instance and require crash recovery at the subsequent startup.*

Disabling the out of memory killer is *not recommended*, however, as this safety mechanism keeps your operating system from crashing when memory runs short, giving you a chance to intervene and restore InterSystems IRIS to normal operation. The recommended way to avoid this problem is to configure enough physical memory and swap space to avoid any chance of the out of memory killer coming into play. (For a detailed discussion see [Process Memory in InterSystems Products](#).)

1.1 Shared Memory Allocations

Review the following guidelines for information on how to appropriately allocate shared memory to address your system's needs:

Database Cache ([globals](#))

Memory allocated to the database cache (also known as the global buffer pool) is used for buffering data.

The default for this is a conservative estimate; you should adjust this parameter based on your system's size and your application's needs. In general, you should allocate a significant portion of your total system memory to the database cache. The database cache will make up the most significant part of your shared memory allocation. If the database cache is reasonably sized, the other shared memory parameters do not need to be adjusted from their defaults.

Routine Cache ([routines](#))

Memory allocated to the routine cache is used for buffering routines and classes that server-side processes will execute.

The default setting for this parameter allocates a reasonable amount determined by the size of your database cache.

Shared Memory Heap ([gmheap](#))

Memory allocated to the shared memory heap is used as needed for a variety of purposes including global mapping, database name and directory information, and the security system.

With a few exceptions, this should be left as the default which allocates a reasonable amount determined by the size of your database cache.

You should only need to adjust this parameter from the default if your application will utilize a large number of SQL queries, if you plan on enabling parallel dejournaling, or if you plan on creating a large number of interoperability-enabled namespaces. In these cases, you will need to allocate additional memory to gmheap. For more information, see [Shared Memory Considerations](#) in Parallel Query Processing, [System Requirements for Parallel Dejournaling](#), and [Create an Interoperability-Enabled Namespace](#).

Journal Buffers ([jrnbufs](#))

Memory allocated to journal buffers determines the amount of journal data that can be held in memory.

The default value for this parameter is 64 MB which is generally sufficient.

1.1.1 Estimating Shared Memory Total

Once you have determined an initial setting for each of these parameters, you can estimate the instance's total shared memory requirement using the following formula. ([MaxServers](#) and [MaxServerConn](#) specify the maximum number of ECP connections allowed *from* the instance and *to* the instance respectively; the default is 2 for the former and 1 for the latter.)

$$\text{globals} * 1.08 + \text{routines} * 1.02 + \text{gmheap (in MB)} + \text{jrnbufs} + (\text{MaxServers} + \text{MaxServerConn}) * 2 + 300$$

Actual process and shared memory use may differ from the above estimates. In particular:

- The multiplier of 1.08 for the database cache ([globals](#)) applies to the default database block size of 8 KB. This factor is lower for larger block sizes; if you [allow and allocate database caches to multiple block sizes](#), the smaller the proportion of the total allocated to the 8 KB block size, the smaller the effective multiplier will be.
- The size of the shared memory heap may be automatically increased to reflect the host's CPU count.

[Reviewing and Monitoring Memory Usage](#) describes how to review the instance's actual use of process and shared memory.

1.2 Configuring Huge and Large Pages

Where supported, the use of huge/large memory pages can be of significant performance benefit and is highly recommended for systems where performance is a priority.

By default, when huge/large pages are configured, InterSystems IRIS attempts to utilize them on startup. If there is not enough space, InterSystems IRIS reverts to standard pages. However, you can use the [memlock](#) parameter to control this behavior and fail at startup if huge/large page allocation fails.

On Windows, the amount of large pages scales automatically. On IBM AIX® and Linux distributions, you must configure the amount of huge/large pages. The huge/large pages allocation should be slightly greater than the shared memory allocation. If you allocate exactly as many huge/large pages as your shared memory, you may have to adjust when upgrading to account for small changes in shared memory between versions. To prevent having to recalculate each upgrade, allocate several additional MB to account for this potential variance. There are two options for determining the amount to allocate:

- [Estimate your shared memory](#) and determine the amount of large/huge pages to allocate accordingly.
- Deploy InterSystems IRIS without huge/large pages enabled, record the shared memory allocation that is displayed in the messages.log file, then allocate huge/large pages accordingly and reboot the system.

To configure huge/large pages, review the relevant documentation for your system:

- [IBM AIX®](#)
- [Linux \(all distributions\)](#)
- [Windows](#)

1.2.1 Configuring Large Pages on IBM AIX®

AIX supports multiple page sizes: 4 KB, 64 KB, 16 MB, and 16 GB. Use of 4 KB and 64 KB pages is transparent to InterSystems IRIS. In order for InterSystems IRIS to use 16 MB large pages, you must configure them within AIX. AIX does not automatically change the number of configured large or huge pages based on demand. Currently, InterSystems IRIS does not use 16 GB huge pages.

Large pages should be configured only in high-performance environments because memory allocated to large pages can be used only for large pages.

Configure large pages on AIX using the **vmo** command as follows:

```
vmo -r -o lgpg_regions=<LargePages> -o lgpg_size=<LargePageSize>
```

where *<LargePages>* specifies the number of large pages to reserve, and *<LargePageSize>* specifies the size, in bytes, of the hardware-supported large pages.

Note: On systems that support dynamic Logical PARtitioning (LPAR), you can omit the **-r** option to dynamically configure large pages without a system reboot.

For example, the following command configures 1 GB of large pages:

```
# vmo -r -o lgpg_regions=64 -o lgpg_size=16777216
```

Once you have configured large pages, run the **bosboot** command to save the configuration in the boot image. After the system comes up, enable it for pinned memory using the following **vmo** command:

```
vmo -o v_pinshm=1
```

However, if *memlock=64*, `vmo -o v_pinshm=1` is not required.

1.2.2 Configuring Huge Pages on Linux

On Linux platforms with sufficient huge pages available, the InterSystems IRIS shared memory segment is allocated from the huge page pool. Using huge pages saves memory by saving space in page tables, and allows InterSystems IRIS to lock in its shared memory segment so its pages are not paged out. InterSystems recommends the use of huge pages on systems hosting InterSystems IRIS under most circumstances.

The default memory page size on Linux systems is 4 KB. Most current Linux distributions include an option for huge pages, a memory page size of 2 MB or 1 GB depending on system configuration. When huge pages are configured, InterSystems IRIS automatically uses them in memory allocation.

Important: With the 2.6.38 kernel, some Linux distributions have introduced transparent huge pages (THP) to automate the creation, management, and use of huge pages. However, THP does not handle the shared memory segments that make up the majority of the InterSystems IRIS memory allocated, and can cause memory allocation delays at runtime that may affect performance, especially for applications that have a high rate of job or process creation. For these reasons, InterSystems recommends that THP be *disabled* on all systems hosting InterSystems IRIS. For more detailed information on this topic, see [Linux Transparent huge pages and the impact to InterSystems IRIS](#) on InterSystems Developer Community.

To configure huge pages on Linux:

1. Check the status.

/proc/meminfo contains huge pages information. By default, no huge pages are allocated. Default Huge Page size is 2 MB. For example:

```
HugePages_Total:      0
HugePages_Free:       0
HugePages_Rsvd:       0
Hugepagesize:        2048 KB
```

2. Change the number of huge pages.

You can change the system parameter directly: For example, to allocate 2056 huge pages, execute:

```
# echo 2056 > /proc/sys/vm/nr_hugepages
```

Note: Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w vm.nr_hugepages=2056
```

Huge pages must be allocated contiguously, which may require a reboot. Therefore, to guarantee the allocation, as well as to make the change permanent, do the following:

- a. Enter a line in **/etc/sysctl.conf** file:

```
echo "vm.nr_hugepages=2056" >> /etc/sysctl.conf
```

- b. Reboot the system.
- c. Verify **meminfo** after reboot; for example:

```
[root woodcrest grub]# tail -4 /proc/meminfo
HugePages_Total:    2056
HugePages_Free:     2056
HugePages_Rsvd:      0
Hugepagesize:       2048 KB
```

3. Verify the use of huge pages by InterSystems IRIS.

When InterSystems IRIS is started, it reports how much shared memory was allocated; for example, a message similar to the following is displayed (and included in the `messages.log` file):

```
Allocated 3580MB shared memory: 3000MB global buffers, 226MB routine buffers
```

The amount of memory available in huge pages should be greater than the total amount of shared memory to be allocated; if it is not greater, huge pages are not used. For example, with 3580 MB of shared memory and a huge page size of 2 MB, at least 1791 huge pages should be configured.

4. For instances using Linux in Docker, complete the enablement of huge pages support with the following steps:

When running the IRIS container, add the `--cap-add=IPC_LOCK` flag to ensure that the container can lock memory pages and prevent them from being swapped out. Example:

```
docker run ..... --cap-add=IPC_LOCK
```

Alternatively, for containers managed with `docker-compose.yml`, include the `cap_add` configuration under the service definition for IRIS to allow locking memory pages:

```
services:
  ...
  iris:
    ...
    cap_add:
      - IPC_LOCK
```

Note: It is advisable to set **HugePages_Total** close to the shared memory usage, but to allocate several additional MB of huge pages. These can serve as a cushion to address any small changes in shared memory usage that may occur when upgrading your system.

1.2.3 Configuring Large Pages on Windows

InterSystems recommends enabling InterSystems IRIS to allocate its memory as large pages on Windows systems, which provides more efficient use of memory and paging space. To do so, grant the Windows “Lock Pages in Memory” (SELockMemory) privilege to the [Windows user account](#) used to run InterSystems IRIS, which allows InterSystems IRIS to request its memory as large pages. For finer grain control over memory allocation at startup, see [memlock](#).

Note: If InterSystems IRIS is running under the default SYSTEM account, it has the “Lock Pages in Memory” privilege by default.

When you restart InterSystems IRIS while using large pages, you typically also need to restart Windows to guarantee that all configured memory is allocated. If startup is unable to allocate the full amount of configured memory, it attempts to start up with less memory and may or may not use large pages. You can check the actual memory allocated by reviewing the most recent InterSystems IRIS startup in the messages log.

1.3 System Requirements for Parallel Dejournaling

Parallel dejournaling can be configured to allow multiple updater jobs to apply records in parallel. When parallel dejournaling is enabled, a number of processes will run simultaneously including updater, reader, and prefetcher jobs. The default number of updater jobs will be determined based on your system resources in order to optimize performance. This feature increases throughput and is used in mirroring (see [Configuring Parallel Dejournaling](#)) and journal restores (see [Restore Globals From Journal Files Using ^JRNRESTO](#)).

Note: Since parallel dejournaling allows for multiple updaters to record separate globals within the same database, it can increase your throughput even when working with a small number of databases.

You can use the following table to estimate the necessary resources for any number of updaters up to a maximum of 16.

Minimum number of CPUs	Minimum database cache size	Minimum free <code>gmheap</code>	Number of updaters
8	0 GB	0.4 GB	2
9	0 GB	0.6 GB	3
12	0 GB	0.8 GB	4
15	160 GB	1.0 GB	5
18	192 GB	1.2 GB	6
21	224 GB	1.4 GB	7
24	256 GB	1.6 GB	8
27	288 GB	1.8 GB	9
30	320 GB	2.0 GB	10
33	352 GB	2.2 GB	11
36	384 GB	2.4 GB	12
39	416 GB	2.6 GB	13
42	448 GB	2.8 GB	14
45	480 GB	3.0 GB	15
48	512 GB	3.2 GB	16

Note: For information on determining the number of CPUs your system has, see [CPUs \(Processors\)](#).

Enabling additional updaters will most likely require allocating more memory to [gmheap](#). By default, `gmheap` is configured to 3% of the memory allocated to global buffers. For each additional updater, you will need to allocate an additional 200 MB.

2 Storage Planning

Many storage technologies are available today, from traditional magnetic spinning HDD devices to SSD and PCIe Flash based devices. In addition, multiple storage access technologies include NAS, SAN, FCoE, direct-attached, PCIe, and virtual storage with hyper-converged infrastructure.

The storage technology that is best for your application depends on application access patterns. For example, for applications that predominantly involve random reads, SSD or Flash based storage would be an ideal solution, and for applications that are mostly write intensive, traditional HDD devices might be the best approach.

This section provides guidelines for determining the best storage solutions for your application and best practices including:

- [Storage Configuration](#)
- [Storage Connectivity](#)
- [File System Separation](#)
- [Buffered I/O vs. Direct I/O](#)
- [noatime Mount Option](#)

2.1 Storage Configuration

The storage array landscape is ever-changing in technology features, functionality, and performance options, and multiple options provide optimal performance and resiliency for InterSystems IRIS. This section provides general best practice guidelines for optimal InterSystems IRIS performance and data resiliency. Your application's I/O patterns should help you decide with your storage vendor which storage RAID levels and configuration provide the best solution. Different file systems are supported and optimal depending on your configuration.

Where possible, it is best to use block sizes similar to that of the file type. While most storage arrays have a lower limit on the block size that can be used for a given volume, you can approach the file type block size as closely as possible; for example, a 32KB or 64KB block size on the storage array is usually a viable option to effectively support IRIS.DAT files with 8KB block format. The goal here is to avoid excessive/wasted I/O on the storage array based on your application's needs.

Important: Your journal files should be located on [separate](#) for optimal performance and recoverability.

The following table is provided as a general overview of storage I/O within an InterSystems IRIS installation.

I/O Type	When	How	Notes
Database reads, mostly random	Continuous by interface and user processes	Interface query or user process initiates disk I/O to read data	Database reads are performed by daemons serving web pages, SQL queries, or direct user processes
Database writes, ordered but non-contiguous	Approx. every 80 seconds or when pending updates reach threshold percentage of database cache, whichever comes first	Database write daemons (8 processes)	Database writes are performed by a set of database system processes known as write daemons. User processes update the database cache and the trigger (time or database cache percent full) commits the updates to disk using the write daemons. Typically expect anywhere from a few MBs to several GBs that must be written during the write cycle depending on update rates.
WIJ writes, sequential	Approx. every 80 seconds or when pending updates reach threshold percentage of database cache, whichever comes first	Database master write daemon (1 process)	The Write Image Journal (WIJ) is used to protect physical database file integrity from system failure during a database write cycle. Writes are approximately 256KB each in size.
Journal writes, sequential	Every 64KB of journal data or 2 seconds, or sync requested by ECP or application	Database journal daemon (1 process)	Journal writes are sequential and variable in size from 4KB to 4MB. There can be as low as a few dozen writes per second to several thousand per second for very large deployments using ECP and separate application servers.

Bottlenecks in storage are one of the most common problems affecting database system performance. A common error is sizing storage for data capacity only, rather than allocating a high enough number of discrete disks to support expected Input/Output Operations Per Second (IOPS).

I/O Type	Average Response Time	Maximum Response Time	Notes
Database Random Reads (non-cached)	<=1 ms	<=2 ms	Database blocks are a fixed 8KB, 16KB, 32KB, or 64KB—most reads to disk are not cached because of large database cache on the host.
Database Random Writes (cached)	<=1 ms	<=2 ms	All database file writes are expected to be cached by the storage controller cache memory.
Journal Writes	<=1 ms	<=2 ms	Journal writes are sequential and variable in size from 4KB to 4MB.

Please note that these figures are provided as guidelines, and that any given application may have higher or lower tolerances and thresholds for ideal performance. These figures and I/O profiles are to be used as a starting point for your discussions with your storage vendor.

2.2 Storage Connectivity

This section contains considerations about storage area network (SAN) and network-attached storage (NAS).

2.2.1 SAN Fibre Channel

Use multiple paths from each host to the SAN switches or storage controllers. The level of protection increases with multiple HBAs to protect from a single card failure, however a minimum recommendation is to use at least a dual-port HBA.

To provide resiliency at the storage array layer, an array with dual controllers in either an active-active or active-passive configuration is recommended to protect from a storage controller failure, and to provide continued access even during maintenance periods for activities such as firmware updates.

If using multiple SAN switches for redundancy, a good general practice is to make each switch a separate SAN fabric to keep errant configuration changes on a single switch from impacting both switches and impeding all storage access.

2.2.2 Network Attached Storage (NAS)

With 10Gb Ethernet commonly available, for best performance 10Gb switches and host network interface cards (NICs) are recommended.

Having dedicated infrastructure is also advised to isolate traffic from normal network traffic on the LAN. This helps ensure predictable NAS performance between the hosts and the storage.

Jumbo frame support should be included to provide efficient communication between the hosts and storage.

Many network interface cards (NICs) provide TCP Offload Engine (TOE) support. TOE support is not universally considered advantageous. The overhead and gains greatly depend on the server's CPU for available cycles (or lack thereof). Additionally, TOE support has a limited useful lifetime because system processing power rapidly catches up to the TOE performance level of a given NIC, or in many cases exceeds it.

2.3 File System Separation

In the interests of performance and recoverability, InterSystems recommends a minimum of four separate file systems for InterSystems IRIS to host:

1. Installation files, executables, and system databases (including, by default, the write image journal, or WIJ, file)
2. Database files (and optionally the WIJ)
3. Primary journal directory
4. Alternate journal directory

In addition, you can add another separate file system to the configuration for the WIJ file which, by default, is created in the `install-dir\mgr` directory. Ensure that such a file system has enough space to allow the WIJ to grow to its [maximum size](#). For more information on the WIJ, see the “[Write Image Journaling and Recovery](#)” chapter in *Data Integrity Guide*.

Note: On UNIX®, Linux, and macOS platforms, `/usr/local/etc/irissys` is the InterSystems IRIS registry directory and therefore must be on a local filesystem.

In the event of a catastrophic disk failure that damages database files, the journal files are a key element in recovering from backup. Therefore, you should place the primary and alternate journal directories on storage devices that are separate from the devices used by database files and the WIJ. (Journals should be separated from the WIJ because damage to the WIJ could compromise database integrity.) Since the alternate journal device allows journaling to continue after an error on the primary journal device, the primary and alternate journal directories should also be on devices separate from each other. For practical reasons, these different devices may be different logical units (LUNs) on the same storage array; the general rule is the more separation the better, with separate sets of physical drives highly recommended. See [Journaling Best Practices](#) in the “Journaling” chapter of the *InterSystems IRIS Data Integrity Guide* for more information about separate journal storage.

The journal directories and the WIJ directory are not configured during installation. For information on changing them after you install InterSystems IRIS, see [Configuring Journal Settings](#) in the *InterSystems IRIS Data Integrity Guide*.

InterSystems does not support the use of symbolic links for database directories.

Note: Current storage arrays, especially SSD/Flash-based arrays, do not always allow for the type of segregation recommended in the preceding. When using such a technology, consult and follow the storage vendor’s recommendations for performance and resiliency.

2.4 Buffered I/O vs. Direct I/O

In general, most operating systems offer two distinct I/O strategies:

- *Buffered I/O* — The operating system caches reads and writes. This is the normal mode of operation, unless a program (such as InterSystems IRIS) or user specifies otherwise.
- *Direct I/O, Concurrent I/O, or Unbuffered I/O* — Reads and writes bypass the operating system cache, providing better performance and scalability characteristics for some InterSystems IRIS files.

InterSystems IRIS takes advantage of Direct, Concurrent or Unbuffered I/O if available. The choice of file system and mount options may require additional attention depending on the type of files, as follows:

Journal Files

Some platforms have specific recommendations to use direct or concurrent I/O mount options for optimal performance, as documented in Supported File Systems in the *InterSystems Supported Platforms* document for this release. On other platforms, InterSystems IRIS uses direct I/O automatically for journal files as appropriate and no special consideration is required.

Databases (IRIS.DAT files)

InterSystems IRIS uses its own database cache, so buffering at the operating system level is not advantageous for database files. Be sure to [allocate](#) sufficient space for the database cache, and also review the details of the optimal use of direct I/O for your platform.

- Linux — InterSystems IRIS uses direct I/O for database files.
If using the VxFS file system, this can be overridden by mounting the file system for concurrent I/O with the **cio** mount option.
- macOS — InterSystems IRIS uses buffered I/O for database files.
- Windows — InterSystems IRIS uses unbuffered I/O for database files.
- IBM AIX — InterSystems IRIS uses concurrent I/O for database files, regardless of whether the **cio** file system mount option is used.

Note: On AIX, in unusual configurations in which an external command is used to read a database file while InterSystems IRIS is running, the external command may fail because the file is opened for concurrent I/O by InterSystems IRIS. An example is performing an external backup using the **cp** command instead of a more sophisticated backup or snapshot utility. Mounting the file system with the **cio** option resolves this by forcing all programs to open files with concurrent I/O.

Installation files and executables

InterSystems IRIS uses buffered I/O for these files. On platforms that support the direct I/O mount option, InterSystems still recommends using buffered I/O for these files.

External application files and streams

Applications that use external files typically benefit from those files being located on a buffered file system.

2.5 noatime Mount Option

Generally, it is advisable to disable updates to the file access time when this option is available. This can typically be done using the **noatime** mount option on various file systems.

3 CPU Sizing and Scaling for InterSystems IRIS

InterSystems IRIS is designed to make the most of a system's total CPU capacity. Keep in mind that not all processors or processor cores are alike. There are variations at the surface such as clock speed, number of threads per core, and processor architectures, and also the varying impact of virtualization.

- [Basic CPU Sizing](#)
- [Balancing Core Count and Speed](#)
- [Virtualization Considerations for CPU](#)

- [Leveraging Core Count with Parallel Query Execution](#)

3.1 Basic CPU Sizing

Applications vary significantly from one to another, and there is no better measurement of CPU resource requirements than benchmarking and load testing your application and performance statistics collected from existing sites. If neither benchmarking or existing customer performance data is available, start with one of the following calculations:

- 1-2 processor cores per 100 users.
- 1 processor core for every 200,000 global references per second.

Important: These recommendations are only starting points when application-specific data is not available, and may not be appropriate for your application. It is very important to benchmark and load test your application to verify its exact CPU requirements.

3.2 Balancing Core Count and Speed

Given a choice between faster CPU cores and more CPU cores, consider the following:

- The more processes your application uses, the greater the benefit of raising the core count to increase concurrency and overall throughput.
- The fewer processes your application uses, the greater the benefit of the fastest possible cores.

For example, an application with a great many users concurrently running simple queries will benefit from a higher core count, while one with relatively fewer users executing compute-intensive queries would benefit from faster but fewer cores. In theory, both applications would benefit from many fast cores, assuming there is no resource contention when multiple processes are running in all those cores simultaneously. The number of processor cores is a factor in estimating the memory to provision for a server, so increasing the core count may require additional memory.

3.3 Virtualization Considerations for CPU

Production systems are sized based on benchmarks and measurements at live customer sites. Virtualization using shared storage adds very little CPU overhead compared to bare metal, so it is valid to size virtual CPU requirements from bare metal monitoring.

Note: For hyper-converged infrastructure (HCI) deployments, add 10% to your estimated host-level CPU requirements to cover the overhead of HCI storage agents or appliances.

In determining the best core count for individual VMs, strike a balance between the number of hosts required for availability and minimizing costs and host management overhead; by increasing core counts, you may be able to satisfy the former requirement without violating the latter.

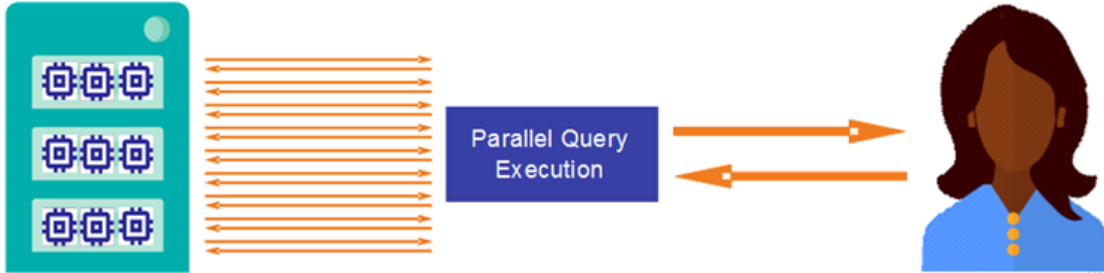
The following best practices should be applied to virtual CPU allocation:

- Production systems, especially database servers, are assumed to be highly utilized and should therefore be initially sized based on assumed equivalence between a physical CPU and its virtual counterpart. If you need six physical CPUs, assume you need six virtual CPUs.
- Do not allocate more vCPUs than required to optimize performance. Although large numbers of vCPUs can be allocated to a virtual machine, there can be a (usually small) performance overhead for managing unused vCPUs. The key here is to monitor your systems regularly to ensure that vCPUs are correctly allocated.

3.4 Leveraging Core Count with Parallel Query Execution

When you upgrade by adding CPU cores, an InterSystems IRIS feature called parallel query execution helps you take the most effective advantage of the increased capacity.

Figure 1: Parallel Query Execution



[Parallel query execution](#) is built on a flexible infrastructure for maximizing CPU usage that spawns one process per CPU core, and is most effective with large data volumes, such as analytical workloads that make large aggregations.

4 Configuring InterSystems IRIS Shared Memory

The shared memory allocations discussed in the previous section can be made during deployment using the [configuration merge feature](#) by including the parameters cited above in a configuration merge file, as shown in the following example:

```
[config]
globals=0,0,5000,0,0,0
gmheap=165888
jrnbufs=64
routines=151
```

Note: The [gmheap](#) value is specified in KB, the others in MB. The multiple fields in the [globals](#) value specify the size of the database cache for different database block sizes; typically only the cache for the 8 KB block size is specified, as shown here.

The settings in the file are merged into the default configuration parameter file (CPF) during deployment, so when the instance starts up for the first time it will do so with the specified parameter values in place and its shared memory allocated accordingly. (Configuration merge is very useful in automated deployment, allowing you to deploy differently-configured instances from the same source by applying different merge files.)

If these allocations are not made using configuration merge at deployment, they can be specified immediately after deployment, or at any time, in the following ways:

- Use configuration merge by executing the [iris merge command](#).
- Use the Management Portal; procedures are documented in the following locations:
 - Database cache and routine cache — [Allocating Memory to the Database and Routine Caches](#) in the *System Administration Guide*
 - Shared memory heap — [Configuring the Shared Memory Heap](#) in the *System Administration Guide*
 - Journal buffers — [Configuring Journal Settings](#) in the *Data Integrity Guide*

- Use the appropriate ObjectScript class for the purpose, as described in the class reference and indicated in the parameter's entry in the [Configuration Parameter File Reference](#); for the memory parameters discussed here, this would be the `Config.config` class.
- Edit the instance's `iris.cpf` file (which is located in the [installation directory](#)) and change the values of the parameters described in the previous section.

When you have made all the desired changes by any of these methods, restart the instance so they can take effect.

5 Reviewing and Monitoring Memory Usage

Performance problems in production systems are often due to insufficient memory for application needs. [Adding memory](#) to the server hosting one or more InterSystems IRIS instances lets you allocate more to the database cache, the routine cache, shared memory heap, or some combination. A database cache that is too small to hold the workload's working set forces queries to fall back to disk, greatly increasing the number of disk reads required and creating a major performance problem, so this is often a primary reason to add memory. Increases in shared memory heap and the routine cache may also be helpful under certain circumstances such as when using [parallel dejournaling](#) or [parallel query processing](#).

At the end of instance startup, messages summarizing the instance's shared memory allocations are written to the [messages log](#), similar to this example, which incorporates the values from the example in [Configuring InterSystems IRIS Shared Memory](#):

```
11/06/21-10:59:37:513 (91515) 0 [Generic.Event] Allocated 5682MB shared memory using Huge Pages
11/06/21-10:59:37:514 (91515) 0 [Generic.Event] 5000MB global buffers, 151MB routine buffers,
64MB journal buffers, 289MB buffer descriptors, 162MB heap, 6MB ECP, 9MB miscellaneous
```

Note: The term *buffer descriptors* refers to the control structures associated with the global, routine, and journal buffers. The *heap* figure may be more than you specified for `gmheap` due to an automatic adjustment.

If you use configuration merge to deploy with your desired shared memory allocations, you can confirm that you have the allocation you intended by reviewing these messages. If you allocate shared memory following deployment, you can review them following the restart you initiate after making changes.

Once the system is operating, in testing or production, you can review actual memory usage within InterSystems IRIS as follows:

- To view the instance's shared memory usage, go to Management Portal's System Usage Monitor page (**System Operation > System Usage**), then click the **Shared Memory Heap Usage** button to display the Shared Memory Heap Usage page; for a description of the information displayed on that page, see [Shared Memory Heap Usage](#) in the *Monitoring Guide*.
- To roughly estimate the maximum memory usage by InterSystems IRIS processes, multiply the peak number of running processes by the default Maximum Per-Process Memory (`bbsiz`) setting of 262.144 MB. However, if this setting has been changed to -1 for "unlimited" (see [Setting the Maximum Per-Process Memory](#)), which is recommended by InterSystems for most production systems, a more detailed analysis is required to estimate the maximum memory usage by these processes is required. To learn more about memory use by InterSystems IRIS processes, see [Process Memory in InterSystems Products](#).

Important: If [System Monitor](#) generates the alert **Updates may become suspended due to low available buffers** or the warning **Available buffers are getting low (25% above the threshold for suspending updates)** while the system is under normal production workload, the database cache (global buffer pool) is not large enough and should be increased to optimize performance.

6 Data Compression and Reducing Storage Costs

Storage costs often make up a large part of an application's overhead, particularly when using cloud infrastructure. Careful monitoring and regular data compression can help to reduce these costs. InterSystems IRIS offers several different strategies for data compression that have different benefits and risks. You can use the table below as a guide for determining which strategies will be best for your individual application.

Strategy	Benefits	Risks
----------	----------	-------

Strategy	Benefits	Risks
Journal compression	<ul style="list-style-type: none"> Offers significant and immediate storage reduction. Compatible with journal encryption. Daily journal volumes are often larger than the sum of the databases, so compression ratios of over 5:1 are possible. 	<ul style="list-style-type: none"> Can increase the overhead necessary for journal restores. Compression ratio may be lower depending on the contents of the journal file.
In-Database stream compression	<ul style="list-style-type: none"> Suitable for compression of large text or binary streams. Works with encrypted databases. You may be able to reduce the size of the .DAT file by using compact freespace and truncate. 	<ul style="list-style-type: none"> Stream compression only works for Stream data types. Only compresses <i>new</i> streams. May require updates or changes to your code.
Purging data	<ul style="list-style-type: none"> Data volume is reduced directly by the amount purged. You may be able to reduce the size of the .DAT file by using compact freespace and truncate. 	<ul style="list-style-type: none"> The data is permanently removed. Purging data other than at the .DAT level is complicated.
Archiving data	<ul style="list-style-type: none"> The data remains accessible. The data can be moved to less expensive storage options. You may be able to reduce the size of the .DAT file by using compact freespace and truncate. 	<ul style="list-style-type: none"> You may have higher latency and worse performance when trying to access archived data. Archiving data other than at the .DAT level is complicated.
Compacting globals	<ul style="list-style-type: none"> Consolidates global data into fewer blocks which increases the amount of free space in a database. You retain the data while saving space. You may be able to reduce the size of the .DAT file by using compact freespace and truncate. 	<ul style="list-style-type: none"> High global block density can significantly reduce performance for random-inserts. Globals with random-inserts or random-deletes may need to be compacted regularly. Compacting globals is an I/O heavy operation that you need to run on both members when mirroring.

Strategy	Benefits	Risks
Storage level compression	<ul style="list-style-type: none">• Happens automatically, requiring the least effort of any strategy.• When using the same storage array for mirrored members, de-duplication features can have significant benefits.• Compatible with storage level encryption.	<ul style="list-style-type: none">• Storage level compression is supplied by your vendor and can come with additional costs.• You should consider the break-even point where any additional costs outweigh savings on storage.• Encrypted databases may negate the benefits of storage level compression.• The size of the .DAT file will not change.

Database management is an important tool in reducing storage overhead. See [Maintaining Local Databases](#) for more information on the topic.

