IBM Power10 performance optimization for IBM i

Best practices guide to maximize throughput of IBM i applications on IBM Power

Table of contents

Introduction

This paper is intended to provide guidance, best practice recommendations, and a checklist for maximizing throughput, minimizing response time, and minimizing CPU utilization of IBM® i applications running on IBM Power® servers. This paper by no means covers all best practices and should be used in conjunction with other IBM PowerVM®, IBM AIX®, or IBM i documents.

Using IBM Power features with IBM i

The operating system and most applications for IBM i are built on a technology-independent machine interface (TIMI) that isolates programs from differences in processor architectures and allows the system to automatically capitalize on new IBM Power Architecture® features without changes to existing programs.

Multipage size support on IBM i

Most of the IBM i operating systems use 4 KB pages but selected system functions automatically use 64 KB pages. Running applications can create shared memory objects that use 64 KB pages (typically by using shmctl with the SHM_PAGESIZE command). IBM technology for Java™ programs can use 64 KB pages for Java heap. IBM PASE for i programs automatically use 64 KB pages for shared library text and data, and can request 64 KB pages for program text, stack, and data.

Vector Scalar eXtension

IBM i automatically uses vector instructions to improve the performance of some cryptographic operations. PASE for i applications can use Vector Scalar eXtension (VSX).

Decimal floating point

IBM i supports decimal floating point (DFP) in selected programming languages (such as, ILE C/C++ and ILE COBOL) and in IBM Db2® for i. DFP operations (outside of PASE for i) automatically use DFP hardware instructions on all current processor families. IBM i improves the performance of many DFP operations (compared to prior releases) by the increased use of DFP hardware instructions.

ZLIB support for SAVLIB

IBM i 7.5 adds support for compressing data using the DEFLATE compression format (which is the underlying format for .zip and .gzip files, and the C zlib library). This support is available as a new compression format option on the SAVLIB command's DTACPR parameter (*ZLIB) and a new format option for the CPRDATA MI instruction. In addition, an IBM i partition running on a Power10 processor-based system can use the Power processor's Nest Accelerator (NX) to accelerate handling of DEFLATE data in those interfaces.

The performance of any compression algorithm varies drastically based on the nature of the data that is being compressed. All data compression works by reducing the redundancy in the input; data that is completely unpredictable cannot be compressed, and data with predictable patterns can be compressed very well. DEFLATE compressors will also vary in their throughput depending on the input data.

On the data that IBM i has tested, IBM i's DEFLATE implementation typically compresses better than LZ1 (which is the compression offered by SAVLIB DTACPR(*HIGH)); that is, DEFLATE typically eliminates more redundancy, leading to smaller output than LZ1. If NX support is not available, DEFLATE typically gives similar CPU and wall-clock time throughput to LZ1. When NX is used, DEFLATE compression operations typically offer at least five times faster throughput than LZ1.

Given that, there are significant performance gains to be had by ensuring that a partition has access to NX resources. There is a limited amount of NX capacity available system wide for DEFLATE operations, which is shared by all the partitions in the system. Partitions can reserve NX resources ("Quality of Service credits") for DEFLATE/gzip from the Hardware Management Console (HMC); each credit is the capability to have one request outstanding to an NX. Any credits reserved this way reduce the number of outstanding requests that other partitions can have. The credits that are not reserved by any partition are shared by any partitions that attempt to send more requests at a time than they have reserved.

To ensure that NX resources are available to your partition, you must have your partition is in the IBM Power10 compatibility mode, and reserve credits at the HMC partition using

the **General Properties**, **Advanced** section, **Supported Hardware Accelerator Types** option.

IBM i operating system-specific optimizations

This section describes optimization methods that are specific to IBM i.

IBM i advanced optimization techniques

Optimization methods specific to the creation of IBM i programs and service programs include the following methods:

- 8-byte pointers in C and C++ code
  The performance of C and C++ code that uses pointers can be improved when the code is compiled to use 8-byte pointers, rather than 16-byte pointers (default). To take full advantage of 8-byte pointers, specify the STGMDL(*TERASPACE) and DTAMDL(*LLP64) parameters when you compile code.
- Program profiling
  Program profiling is an advanced optimization technique to reorder procedures, or code within procedures, and to direct code generation decisions in ILE programs and service programs based on statistical data that is gathered while running the program. The reordering can improve instruction cache utilization and reduce the paging that is required by the program, improving performance.
- Argument optimization
  The argument optimization parameter, with ARGOPT(*YES), is available with the CRTPGM and CRTSRVPGM commands to support advanced argument optimization, where an analysis across modules that are bound to the program is performed. In general, this improves the performance of most procedure calls within the program. Argument optimization is a technique for passing arguments (parameters) to ILE procedures to improve performance of call-intensive applications.

- Interprocedural analysis

  Interprocedural analysis is a mechanism for performing optimization across all modules that are bound into a program and were compiled with the MODCRTOPT(*KEEPILDTA) option. Interprocedural analysis is enabled with IPA(*YES) on the CRTPGM or CRTSRVPGM command. In contrast, intraprocedural analysis is a mechanism for performing optimization for each function within a compilation unit, by using only the information that is available for that function and compilation unit.

- Licensed Internal Code Options (LICOPTs)

  LICOPTs are compiler options that are passed to the Licensed Internal Code to affect how code is generated or packaged. You can use some of the options to fine-tune the optimization of your code. The TargetProcessorModel LICOPT instructs the translator to perform optimizations that are tuned for the specified processor model. Programs that are created with this option run on all supported hardware models, but run faster on the specified processor model.

  A TargetProcessorModel value can be specified so that the code is tuned to run optimally on a specific processor family. The CodeGenTarget LICOPT specifies the creation target model for a program or module object. The creation target model indicates the hardware features that the code that is generated for that object can use.

  The CodeGenTarget option specifies the creation target model for a program or module object. The creation target model indicates the hardware features that the code generated for that object can use.
  The TargetProcessorModel and CodeGenTarget LICOPTs are two of several factors, including adaptive code generation (ACG), which determine the processor model to which code should be tuned and targeted when creating a module, changing a module or program, or re-creating a module or program. The default behavior is to use all the features available on the current machine.

- Adaptive code generation (ACG)

  ACG is a technology that allows you to take advantage of all the processor features on your systems, regardless of whether those features are present on other system models that are supported by the same release. Furthermore,

programs can be moved from one system model to another and continue to run correctly, even if the new system does not have all the processor features that were available on the original system.

ACG can work without user intervention in most scenarios. However, if you build and distribute software to run on various system models, you might want to exercise some control over which processor features are used by ACG. The first time a program object is activated on a system to which it is moved, the system performs a compatibility check to ensure that your program does not use any features that are unavailable on your system. If the program requires any processor feature that is not supported by the system to which it was moved, then the system automatically calls the optimizing translator to convert the program to be compatible.

After a restore operation, incompatible module and program objects are converted on their first activation by default, but options also exist to immediately convert them after the restore.

For more information about these optimizations in an IBM i environment, see ILE Concepts, SC41-5606; Chapter 13, Advanced Optimization Techniques.

Activation group considerations

This section explains what an activation group is and the considerations to work with an activation group.

What is an activation group

All ILE programs and service programs are activated within a substructure of a job called an activation group. This substructure contains the resources necessary to run the programs. These resources fall into the following general categories:

- Static program variables
- Dynamic storage
- Temporary data management resources
- Certain types of exception handlers and ending procedures

Activation groups use either single-level storage or teraspace for supplying storage for static program variables. For more information, see Teraspace and Single-Level Storage. When single-level storage is used, the static program variables and dynamic storage are assigned separate address spaces for each activation group, which provides some degree of program isolation and protection from accidental access. When teraspace is used, the static program variables and dynamic storage may be assigned separate address ranges within teraspace, which provides less program isolation and protection from accidental access.

The temporary data management resources include the following functions:

- Open files [open data path (ODP)]
- Commitment definitions
- Local SQL cursors
- Remote SQL cursors
- Hierarchical file system (HFS)
- User interface manager
- Query management instances
- Open communications links
- Common Programming Interface (CPI) communications

The separation of these resources among activation groups supports a fundamental concept. That is, the concept that all programs activated within one activation group are developed as one cooperative application.

Activation group create/delete performance effects

ACTGRP(*NEW) should be used carefully. When *NEW is used, each time a program is executed, it creates a new activation group for all the resources it needs to run that program, and then system memory is allocated, the program is copied into the activation group, variables are initialized, and so on. When the program completes, the activation group is then cleaned up, static and dynamic storage that has not be deallocated is also returned to the system, and all the other temporary resources are reclaimed by the system.

If *NEW is not absolutely required, change ACTGRP(*NEW) to *CALLER or use a named

activation group. This allows the program to reuse the resources that are already allocated instead of rebuilding the environment.

When your ILE program calls an original program model (OPM), use DFTACTGRP(*YES) for your ILE program. Better yet, change your OPM program to ILE. For more information on how to convert to ILE see the following document: ILE Concepts. Only run RCLACTGRP to explicitly reclaim activation groups when needed.

PASE application optimization

PASE for i provides an integrated runtime environment that allows you to run selected applications without the complexity of managing operating systems, such as AIX or Linux®. PASE for i also provides industry-standard and defacto-standard shells and utilities that provide you with a powerful scripting environment. For more details, refer to: https://www.ibm.com/docs/en/i/7.5?topic=programming-pase-i.

Most applications that run in PASE utilize a high-level language runtime such as Java, Node.js, PHP, or Python. To compile source code for PASE, there are two possible approaches. One approach is to use AIX compilers as documented at: https://www.ibm.com/docs/en/i/7.5?topic=i-compiling-your-aix-source.

The other approach is to use the open source GCC compilers freely available in the open source ecosystem. For information on getting started, see: http://ibm.biz/ibmi-rpms

Matrix-Multiply Assist

The Matrix-Multiply Assist (MMA) in an IBM Power10 processor-based system provides the computational strength and data bandwidth to handle the demanding AI inferencing and machine learning (ML) workloads. This facility is a natural match for implementing numerical linear algebra computations. OpenBLAS is a widely used open source BLAS (Basic Linear Algebra Subprograms) library to speed up linear algebra computations with low-level routines that operate on vectors and matrices with platform-specific optimizations. For more details, refer to: https://developer.ibm.com/tutorials/mma-exploitation-in-openblas-on-aix/

Java performance

There are some actions you can take to achieve better Java performance:

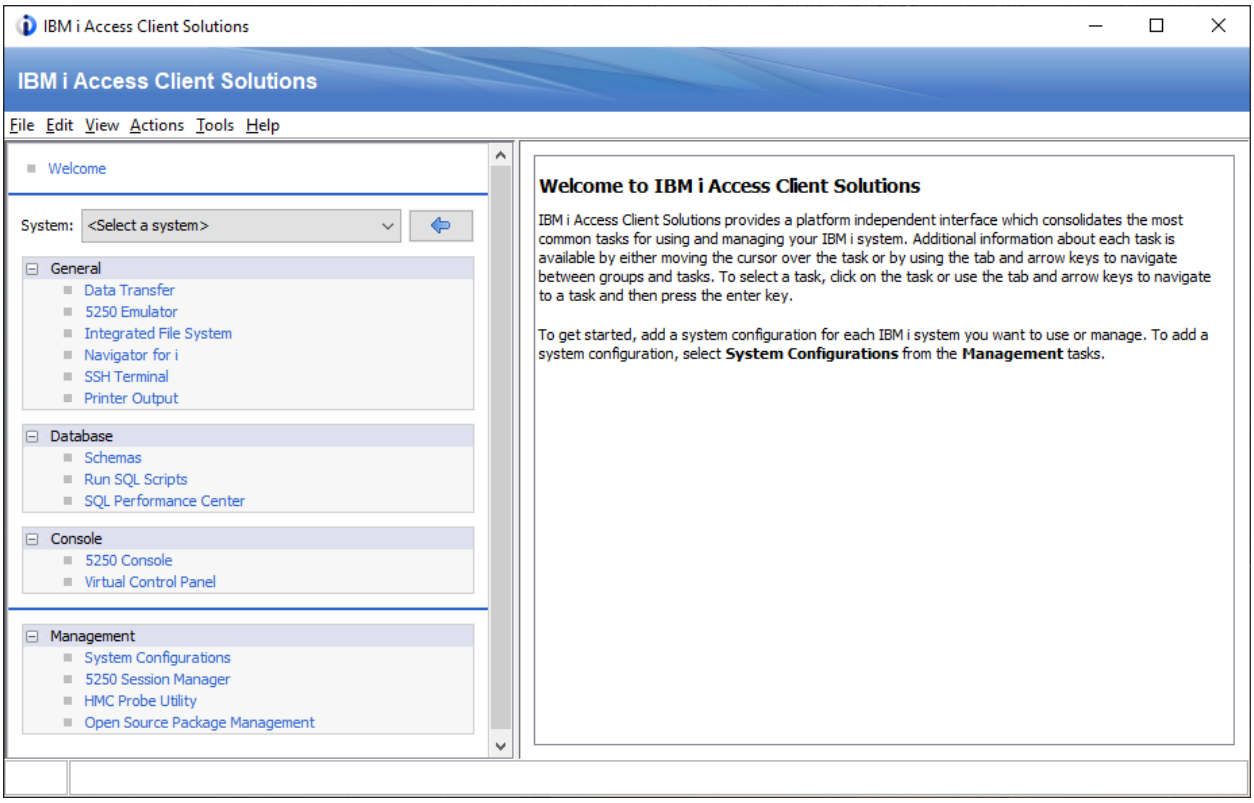
- Improve performance of your Java code by using the just-in-time compiler or using a shared class cache.
- Carefully set your values for optimal garbage collection performance.
- Only use native methods to start system functions that are relatively long running and are not available directly in Java.
- Use Java exceptions in cases that are not the normal flow through your application.
- Load the latest Java program temporary fix (PTF) group as there are often performance improvements delivered during the life of the Java virtual machine (JVM).

For more information on Java tuning, refer to: Tuning Java program performance.

Db2 for i optimization

The goal of database performance tuning is to minimize the response time of your queries by making the best use of your system resources. The best use of these resources involves minimizing network traffic, disk I/O, and CPU time. This goal can only be achieved by understanding the logical and physical structure of your data, the applications used on your system, and how the conflicting uses of your database might affect performance.

The best way to avoid performance problems is to ensure that performance analysis is a part of your ongoing development activities. Many of the most significant performance improvements are realized through careful design at the beginning of the database development cycle. To optimize performance most effectively, you must identify the areas that yield maximum performance increases over the widest variety of situations. Focus your analysis on these areas.



For more information, refer to: IBM i Access - Client Solutions

Db2 for i tools and tuning

Query optimization is an iterative process. You can gather performance information about your queries and control the processing of your queries.

Db2 for IBM i – Health Center

Use the Db2 for IBM i Health Center to capture information about your database. You can view the total number of objects, the size limits of selected objects, the design limits of selected objects, environmental limits, and activity level.

Monitoring your queries using the Database Monitor

The Start Database Monitor (STRDBMON) command gathers information about a query in real time and stores this information in an output table. This information can help you determine whether your system and your queries are performing well, or whether they need fine-tuning. Database monitors can generate significant CPU load and disk storage increase when in use.

Using the SQL Performance Center with detailed monitors

You can work with detailed monitors from the SQL Performance Center. The detailed SQL performance monitor is the SQL Performance Center version of the STRDBMON database monitor, available through the native interface.

Index advisor

The query optimizer analyzes the row selection in the query and determines, based on default values, if creation of a permanent index improves performance. If the optimizer determines that a permanent index might be beneficial, it returns the key columns necessary to create the suggested index.

Viewing your queries with Visual Explain

You can use the Visual Explain tool with IBM i Access Client Solutions (ACS) to create a query graph that graphically displays the implementation of an SQL statement. You can use this tool to see information about both static and dynamic SQL statements. Visual Explain supports the following types of SQL statements: SELECT, INSERT, UPDATE, and DELETE.

Optimizing performance using the Plan Cache

The SQL Plan Cache contains a wealth of information about the SQE queries being run through the database. Its contents are viewable through the IBM i Access Client Solutions interface. Certain portions of the plan cache can also be modified.

Query optimization tools: Comparison

Use the table in Query optimization tools: Comparison to find the information each tool can provide, when it analyzes your queries, and the tasks it can perform to improve your queries.

Changing the attributes of your queries

You can modify different types of query attributes for a job with the Change Query Attributes (CHGQRYA) CL command. When using Run SQL Scripts, you can modify the attributes by selecting the Connection menu, then Connected - [system name], followed by Change Query Attributes.

Query Supervisor

The Db2 for i SQL Query Engine (SQE) provides a Query Supervisor which enables real-time monitoring of resource consumption by SQL and native queries, for example the Open Query File (OPNQRYF) command. Query Supervisor threshold levels can be established for general or specific scenarios. When a threshold is met or exceeded, the query execution is interrupted, and user-supplied exit programs are called.

Setting resource limits with the Predictive Query Governor

The Db2 for i Predictive Query Governor can stop the initiation of a query if the estimated run time (elapsed execution time) or estimated temporary storage for the query is excessive. The governor acts before a query is run instead of while a query is run. The governor can be used in any interactive or batch job on the system. It can be used with all Db2 for i query interfaces and is not limited to use with SQL queries.

Controlling parallel processing for queries

There are two types of parallel processing available. The first is a parallel I/O that is available at no charge. The second is Db2 Symmetric Multiprocessing (SMP), packaged as option 26 of the operating system and available at no charge. You can turn parallel processing on and off. IBM i 7.5 provides improved support for SMP processing: SQE Improved SMP processing and New QAQQINI control PARALLEL_MIN_TIME.

Limiting temporary storage and CPU used by queries

Follow the recommendations in Limiting temporary storage and CPU used by queries to limit temporary storage and CPU usage.

SMP parallel database processing

Consider selectively enabling Db2 Symmetric Multiprocessing feature of IBM i (5770SS1 Product option 26) to allow for parallel processing on longer-running queries. When enabling the use of parallel processing, it is recommended to use a parallel degree setting of *OPTIMIZE. IBM i 7.5 provides additional function and tuning capabilities when using parallel processing.

Application design tips for Db2 for i applications

There are some design tips that you can apply when designing SQL applications to

maximize your database performance.

Minimizing the use of live data

The term *live data* refers to the type of access that the database manager uses when it retrieves data without making a copy of the data. Using this type of access, the data, which is returned to the program, always reflects the current values of the data in the database. The programmer can control whether the database manager uses a copy of the data or retrieves the data directly. This control is done by specifying the allow copy data (ALWCPYDTA) parameter on the precompiler commands or the Start SQL (STRSQL) command.

Reducing the number of open operations

The SQL data manipulation language statements must perform database open operations to create an open data path (ODP) to the data. An open data path is the path through which all input/output operations for the table are performed. In a sense, it connects the SQL application to a table. **The number of open operations in a program can significantly affect performance.**

For more information on reducing full opens, see the following reference: File OPEN and CLOSE Operations Causing a High Volume of Disk Activity.

Retaining cursor positions

You can improve performance by retaining cursor positions.

Programming techniques for database performance

By changing the coding of your queries, you can improve their performance.

Use the OPTIMIZE clause

If an application is not going to retrieve the entire result table for a cursor, using the OPTIMIZE clause can improve performance. The query optimizer modifies the cost estimates to retrieve the subset of rows using the value specified in the OPTIMIZE clause.

Use FETCH FOR n ROWS

Applications that perform many FETCH statements in succession could be improved by using FETCH FOR n ROWS. With this clause, you can retrieve multiple rows of table data with a single FETCH statement, putting them into a host structure array or row storage area.

Use INSERT n ROWS

Applications that perform many INSERT statements in succession could be improved by using INSERT n ROWS. With this clause, you can insert one or more rows of data from a host structure array into a target table. This array must be an array of structures where the elements of the structure correspond to the columns in the target table.

Control database manager blocking

To improve performance, the SQL runtime attempts to retrieve and insert rows from the database manager, a block at a time, whenever possible.

Optimize the number of columns that are selected with SELECT statements

For each column in the SELECT statement, the database manager retrieves the data from the underlying table and maps it to a host variable in the application program. By minimizing the number of columns that are specified, processing unit resource usage can be conserved.

Eliminate redundant validation with SQL PREPARE statements

The processing which occurs when an SQL PREPARE statement is run is like the processing which occurs during precompile processing.

Improve concurrency by avoiding lock waits

The concurrent access resolution option directs the database manager on how to handle cases of record lock conflicts under certain isolation levels.

Coding tips for Db2 for i applications

As you code your applications, there are some general tips that can help you optimize performance.

Effects on database performance when using long object names

Long object names are converted internally to system object names when used in SQL statements. This conversion can have some performance impacts. Names of tables, views, indexes, and aliases that are 30 characters or less will generally perform much better than names longer than 30 characters.

Effects of precompile options on database performance

Several precompile options are available for creating SQL programs with improved performance. They are only options because using them could impact the function of the application. For this reason, the default value for these parameters is the value that ensures successful migration of applications from prior releases. However, you can improve performance by specifying other options.

Effects of the ALWCPYDTA parameter on database performance

Some complex queries can perform better by using a sort or hashing method to evaluate the query instead of using or creating an index.

Tips for using VARCHAR and VARGRAPHIC data types in databases

Variable-length column (VARCHAR or VARGRAPHIC) support allows you to define any number of columns in a table as variable length. If you use VARCHAR or VARGRAPHIC support, the size of a table can typically be reduced.

In addition, the following tools are now available free of charge:

- Performance Tools (5770-PT1)
- Db2 Query Manager and SQL Development Kit (5770-ST1)
- IBM i Access Family (5770-XW1)
- Db2 Symmetric Multiprocessing – Option 26
- Db2 Multisystem – Option 27

For more details on the free of charge tools, refer to the IBM i License topics at:
https://www.ibm.com/support/pages/ibm-i-license-topics

For more details on IBM i database optimization, refer to:
https://www.ibm.com/docs/en/ssw_ibm_i_75/pdf/rzajqpdf.pdf

To get more information about the latest improvements in Db2 for i performance, refer to:
https://www.ibm.com/support/pages/db2-i-performance-enhancements

Performance tooling and IBM i wait accounting

Monitoring and managing your system's performance is critical to ensure you are keeping pace with the changing demands of your business.

To respond to business changes effectively, your system must change too. Managing your system, at first glance, might seem like just another time-consuming job. But the investment pays off soon because the system runs more efficiently, and this is reflected in your business. It is efficient because changes are planned and managed.

Managing performance of any system can be a complex task that requires a thorough understanding of that system's hardware and software. IBM i is an industry leader in performance management and has many qualities that are not found in other systems, including unparalleled performance metrics, always-on collection services, and graphical viewing of performance data. While understanding all the different processes that affect system performance can be challenging and resolving performance problems requires the effective use of a large suite of tools, the functions offered by IBM i are intended to make this job easier for users.

IBM i performance tools

Managing performance requires the use of a variety of specialized applications. Each of these applications offers a specific insight into system performance. Some applications collect the data, while others are used to display, analyze, monitor or manage the data collected.

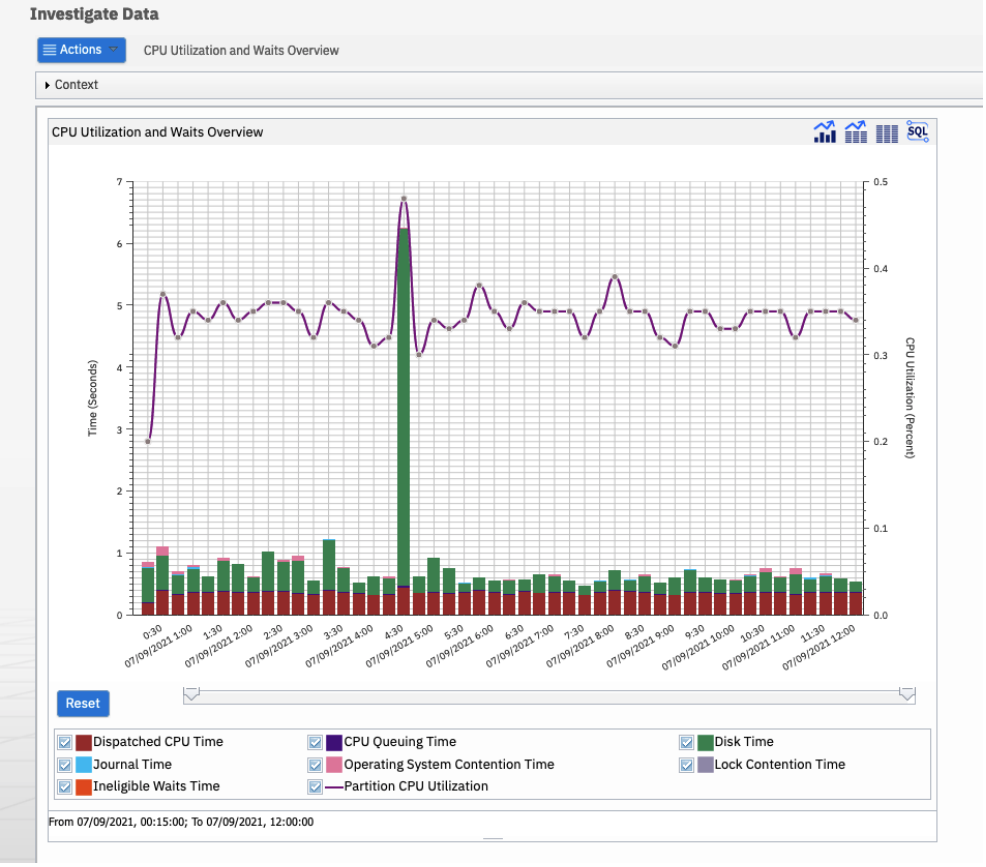
For more details, see the following reference: IBM i Performance

Performance Data Investigator

Performance Data Investigator (PDI), a function included in IBM Navigator for i, provides a web-based GUI for performance data with interactive charts and tables. With PDI you can view and analyze performance data for each of the collectors (Collection Services, IBM i Job Watcher, IBM i Disk Watcher, and Performance Explorer). PDI also contains

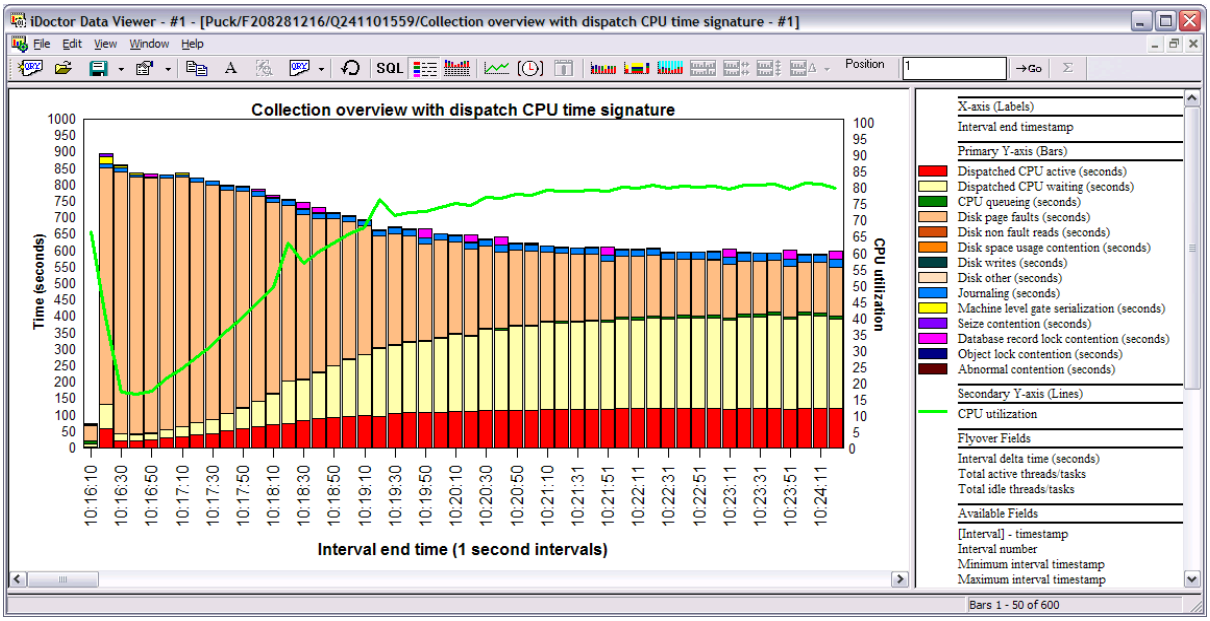
charts for Health Indicators, Database, System Monitors, and Graph History (available in IBM i 7.3 and later releases).

Other useful performance tasks include configuration commands for Collection Services, Disk Watcher and Job Watcher, and Managing Collections.



iDoctor

IBM iDoctor for IBM i is a Microsoft® Windows®-based suite of performance tools that can be used for analysis of Collection Services, Job Watcher, PEX, Disk Watcher, and SQL Plan Cache Snapshots. iDoctor also has a Power Connections component capable of collecting and analyzing data from the HMC and VIOS, AIX, and Linux partitions.



For more information, refer IBM iDoctor for IBM i.

IBM i Wait Accounting

IBM i Wait Accounting is a technology built into the IBM i operating system that provides the ability to identify what any thread or task is doing when it is not using the CPU. Because threads and tasks wait for a wide variety of reasons, wait accounting can be a very powerful capability to aid in understanding the wait conditions and possibly eliminating or reducing wait time, which can have a significant effect on performance.

For more information about IBM i Wait Accounting, refer to: IBM i Wait Accounting.

Performance management on IBM i

For links to general performance resources, performance education resources, performance papers, and articles for IBM i, see the following websites:

- IBM i Performance Docs
- IBM i performance and capacity on IBM Support

For a basic understanding of IBM i on Power performance concepts, workloads and benchmarks on Power, capacity planning, performance monitoring and analysis, frequently asked questions, and guidelines addressing common performance issues, see

the IBM i on Power - Performance FAQ.

Summary

IBM i and Power10 provide powerful functionality, tools, and application performance for developers. Using the operating system, application and database functions, tools, and tips in this document allows developers to make good use of this technology and produce high performance applications.

Get more information

To learn more, refer to:

- IBM i on Power - Performance FAQ
- PASE Programming
- Database Performance and Query Optimization
- ILE Concepts, SC41-5606

About the authors

This document was created by the IBM i Development teams, IBM Performance teams, and IBM Technology Services.

If you'd like to explore consulting services related to the concepts included in this document, contact IBM Technology Services at technologyservices@ibm.com

IBM Corporation New Orchard Road Armonk, NY 10504

Produced in the
United States of America November 2022