IBM i Performance Tools for Application Developers

Dawn May – dmmay@us.ibm.com
@DawnMayiCan

Performance Disclaimer

• “it depends ...”
• Performance information and recommendations in this presentation are based on measurements, analysis, and projections in a controlled environment for specific performance workloads.
• Your results may vary significantly and are dependent on the application and configuration.
• This information is provided along with general recommendations for you to better understand system performance.
• Information is provided *AS IS* without warranty of any kind.
Agenda

- Brief Overview
- Introduction to IBM i Wait Accounting

- Performance Data Collectors
  - Collection Services
  - Job Watcher
  - Performance Explorer

- Performance Data Visualization and Diagnostics
  - Performance Data Investigator
  - iDoctor

- Examples

Keep Current on PTFs

🌟 It’s always good practice to keep current on the latest fixes from IBM

- PTFs address defects
- PTFs introduce new capabilities
  - IBM i Technology Refresh Updates
  - IBM i Group PTFs
    - Database
    - Performance tools
    - Java
    - HTTP Server
      - HTTP Server Group PTF for latest Navigator for i functionality
  - PTFs for performance data collectors
    - Collection Services, Job Watcher, Disk Watcher, Performance Explorer
IBM i Performance Tools

- Green Screen
- Performance Tool Reports
- System commands
- Performance Explorer
- PM for Power Systems
- WCE
- Navigator Performance
  - Investigate Data
  - Batch Model
  - System Monitors
  - Graph History
- System Navigator
  - Perf Tool GUI
  - DB/SQL Monitor
  - Active Jobs
  - System Status
  - Graph History
- iDoctor Suite
  - Job Watcher
  - PEX Analyzer
  - Heap Analyzer
  - VIOS Investigator

IBM i Performance Tools

Collection Services
Job Watcher
Disk Watcher
Performance Explorer

Performance Instrumentation and Data Collection

The Advantage

- IBM develops the software stack, top to bottom
  - Instruments the software with component-specific performance metrics

- IBM develops the performance data collectors that harvest those performance metrics

- IBM i has an integrated database – DB2
  - This is a BIG DEAL
  - Performance data is stored in the database automatically
    - No “add on” application is necessary – it’s all in the Operating System

- IBM provides the graphical analysis tools
  - Analysis of the performance data in the DB2 files using SQL

IBM i has the best performance instrumentation and data collection capabilities in the industry!
Performance Instrumentation and Data Collection

- **Analysis**: Analysis applications mine this data
- **Integrated DB2**: Metric data is stored in DB2 files
- **Collectors**: Performance Collectors harvest metric data
- **Instrumentation**: Operating System Components identify and save metric data

Introduction to Wait Accounting
Performance Fact

“All computers wait at the same speed”

What is Wait Accounting?

- Wait Accounting is the ability to determine what a job is doing when it is not running

i Exclusive! Patented technology built into IBM i.
Wait Accounting Overview

- When a job is not running (i.e., not using CPU), it is waiting
  - But what is it waiting for?

- Waits may be normal, some waits are not normal
  - Wait Accounting helps to determine what the wait is and if it is a problem

- IBM i has instrumented most of the key wait conditions
  - Wait information is automatically collected by Collection Services and Job Watcher

Wait States

- Wait information is tracked for each job, thread and task on system
- A job/thread/task is in one of three states:

  **Using CPU**
  - “Dispatched CPU”
    Assigned to a virtual processor so it can begin execution of instructions

  **Waiting for CPU**
  - “CPU Queuing”
    Ready to use processor, but waiting for it to become available

  **Waiting for something else...**
  - Idle waits
  - Blocked waits

  These waits are typically the most interesting waits to focus on
Types of Waits

- **Idle waits**
  - Typically a normal wait condition; for example:
    - Waiting for the “Enter” key to be pressed on a 5250 display session
    - Waiting for data from the network

- **Blocked waits**
  - Waits that occur while performing a work request
  - Blocked waits are what we want to look at more closely

Wait Accounting - Buckets

- **Wait Buckets** = “Wait condition groups” instrumented in the operating system.
  - Buckets can then be analyzed to determine where a job is spending its time (running or waiting)
  - Categorized into 32 buckets
  - Buckets found in both Collection Services and Job Watcher data
  - Waits can be viewed at a system-level or at an individual job/thread/task level
    - Can also be grouped by generic job name, subsystem, user profile, pool ID, etc.
## 32 Wait Buckets

1. Time dispatched on a CPU
2. CPU queuing
3. Reserved
4. Other waits
5. Disk page faults
6. Disk non-fault reads
7. Disk space usage contention
8. Disk operation start contention
9. Disk writes
10. Disk other
11. Journaling
12. Semaphore contention
13. Mutex contention
14. Machine level gate serialization
15. Seize contention
16. Database record lock contention
17. Object lock contention
18. Ineligible waits
19. Main storage pool contention
20. Classic Java™ user including locks (to 6.1)
   → (7.2) journal save while active
21. Classic Java JVM (up to 6.1), now reserved
22. Classic Java other (up to 6.1), now reserved
23. Reserved
24. Socket transmits
25. Socket receives
26. Socket other
27. IFS
28. PASE
29. Data queue receives
30. Idle/waiting for work
31. Synchronization Token contention
32. Abnormal contention

---

### Wait Accounting – “Run-wait” signature

Applying the concepts of wait accounting, we are now able to identify the amount of time the thread/task was running and the time the thread/task was waiting.

Consider the following:

**Batch job with total run time of 6 hours**

**Run-wait signature**

```
CPU 140 min  | CPU queuing 50 min  | Wait 170 min
```

Elapsed time 6 hours (360 mins)
Wait Accounting – “Run-wait” signature

Wait breakdown
170 minutes

Potential for job to complete sooner if waits can be eliminated

Disk Page Fault Time
170 minutes

- Disk Writes: 30 mins
- Journal: 15 mins
- DB Record Locking: 15 mins
- Object Lock: 10 mins

Now you can start asking questions such as:
- Are my pool sizes appropriate? What objects is the faulting occurring on?
- Is the write cache being overrun? Is the application forcing writes out synchronously?
- Are all the journals optimally configured? Are unnecessary objects being journaled?
- Am I locking records or objects unnecessarily?

Metrics related to components of wait time

<table>
<thead>
<tr>
<th>Component</th>
<th>Count</th>
<th>Time</th>
<th>Avg. Per Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk reads</td>
<td>3,523</td>
<td>42 sec</td>
<td>0.012 sec</td>
</tr>
<tr>
<td>Disk writes</td>
<td>17,772</td>
<td>73 sec</td>
<td>0.004 sec</td>
</tr>
<tr>
<td>Record Locks</td>
<td>355</td>
<td>45 sec</td>
<td>0.126 sec</td>
</tr>
<tr>
<td>Journal</td>
<td>5,741</td>
<td>44 sec</td>
<td>0.007 sec</td>
</tr>
</tbody>
</table>

- Tools capture this level detail
- Useful to know both counts and time
**Why developers should leverage wait accounting!!**

- Helps you understand application characteristics
  - Is it CPU bound? I/O bound?

- Helps you to understand where to focus your effort and investment
  - Is there a bottleneck on CPU, Memory, I/O, Contention time?
  - Invest resources where greatest benefit will be

- Can offer insight into potential performance issues before end-users are affected
  - Can leverage aspects of wait accounting in test environment
    - Eliminate surprises
    - Identify bottlenecks that prevent scaling

- Provides valuable clues to help analyze performance issues as they arise

- Instrumentation part of base IBM i operating system, IBM tools available to help you analyze

**Common Waits that Applications use**

- Disk Waits
- Semaphores, Mutexes, Synchronization Tokens
- Journaling
- Database record locks
- Object locks
- Sockets
A few other things to know about waits…

- Some waits are “expected” and others “unexpected”
  - A record lock may be expected, but one that lasts for a very long duration is unexpected

- If waits can be reduced or eliminated, CPU can be used more efficiently

- One wait may be reduced/eliminated, only to have another wait surface

- Likely won’t be able to remove all wait times

- When is a wait “bad”?
  - Is there a business impact? Are users complaining?
  - “It depends” but waits more than 25% of run time may need additional analysis

Tools for analyzing Wait Accounting information
Wait Accounting - IBM i Performance Data Collectors

- **Collection Services**
  - Collects data automatically 24 X 7 at specified intervals (typically 5 or 15 minutes)
  - System and job level data
  - Starting point!

- **Job Watcher**
  - Needs to be started/stopped (typically 5 or 10 second intervals)
  - Additional detailed data such as call stacks, object waited on, holder
  - Frequently needed to solve performance issues

Performance Data

*Collection vs Visualization*

- The performance data *collection* capability is built into the operating system
  - **Everyone** can collect all types of IBM i performance data

- The *visualization* of the data may require additional products
  - Visualization is the display of charts or tables of performance data
  - The Performance Data Investigator is part of the operating system
    - **Everyone** can visualize Collection Services data
Two Graphical Analysis Tools

- **Performance Data Investigator – Job Watcher**
  - Requires 57xx-PT1 – Job Watcher feature
  - Geared to *average* user

- **iDoctor – Job Watcher**
  - IBM i Service offering, yearly license by serial number
  - Geared to *advanced* user

Wait Accounting IBM i Graphical Analysis Tools

- Two powerful graphical tools to help make your analysis more efficient and productive:

  **Performance Data Investigator (PDI)**
  - Component in IBM Navigator for i (browser-based)
  - Nothing to install, can view Collection Services for free

  **IBM iDoctor for IBM i**
  - Microsoft Windows based client
  - Requires Job Watcher yearly license to see Collection Services data (IBM Service offering)
  - [https://www-912.ibm.com/i_dir/idocor1nsf](https://www-912.ibm.com/i_dir/idocor1nsf)
Wait Accounting IBM i Graphical Analysis Tools

- Both graphical tools sit on top of same rich IBM i instrumentation, but not equivalent in presentation and function

Wait Accounting Analysis Strategy

- Understand the “big picture” first
  - Understand overall partition characteristics first and where system bottlenecks may be that affecting your application
  - Typically done using Collection Services data
  - Drill down to job level
  - Waits can be analyzed in various useful ways:

- Continue detailed analysis at a Job Level using Job Watcher
  - Narrow focus to interesting timeframes / jobs
  - Many more job level details available
Using Performance Data Investigator (PDI)

- IBM Navigator for i is the Web console for managing IBM i
  - Has much of the function as System i Navigator
    - but with a browser user interface

---

PDI Wait Accounting Perspectives - Where to start

Performance -> Investigate Data -> Collection Services:

Option 1: **CPU Utilization and Waits Overview**
- Combines related waits into higher level buckets

Option 2: **Waits Overview**
- All individual “blocked” wait buckets shown
System “run-wait” signature

CPU Utilization and Waits Overview

- A chart that combines CPU utilization as well as the wait buckets can be very beneficial in assessing the health of your partition.
- In this chart, we can see that the majority of the time, the jobs were spending time in CPU as well as in Disk. Minor amounts of Journal wait time and operating system contention time are also present.

CPU Utilization and Waits Overview

Drilldown analysis

Because Disk wait time was fairly significant, drilldown to Disk Waits Overview to further examine the detailed waits contributing to this time:

We can see that Disk Page Fault time is the biggest contributor to Disk Time. (A job needed something in memory, it wasn’t there, had to do an I/O to bring it into memory before job could continue running).
Waits by Job or Task

The next question likely would be which job(s) are incurring this wait time. Drilling down further, we can see the list of jobs incurring this wait time:

This type of chart can also be used to understand a job(s) “run-wait” signature.

Efficient System with Little Waiting

Primarily Dispatched CPU Time
Processor Bound System

I/O Bound System
I/O – Further Investigation

Disk Page Faults Wait Time

Lock Contention Time Bottleneck

Job Watcher data is typically needed to solve lock related issues.
Lock – Further Investigation

Object Lock Contention time

Job Watcher data will show object waited on, the holder, and call stacks for both the waiter and the holder (example shown later on...)

Wait Accounting at a Job Level

Would this job benefit from additional memory?

<table>
<thead>
<tr>
<th>PRICER/JSMITH/010628</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatched CPU Time</td>
</tr>
<tr>
<td>Disk Non-fault Reads Time</td>
</tr>
<tr>
<td>Disk Writes Time</td>
</tr>
<tr>
<td>Seize Contention Time</td>
</tr>
<tr>
<td>Socket Receives Time</td>
</tr>
<tr>
<td>CPU Queuing Time</td>
</tr>
<tr>
<td>Disk Space Usage Contention Time</td>
</tr>
<tr>
<td>Journal Time</td>
</tr>
<tr>
<td>Database Record Lock Contention Time</td>
</tr>
<tr>
<td>Main Storage Pool Overcommitment Time</td>
</tr>
<tr>
<td>Object Lock Contention Time</td>
</tr>
<tr>
<td>Abnormal Contention Time</td>
</tr>
</tbody>
</table>
Wait Accounting at a Job Level

Would this job benefit from additional memory? CPU? Disk?

---

Wait Accounting at a Job Level

Would this job benefit from an improved I/O subsystem?
Job Watcher - Additional Benefits

- Collects more detailed performance data than Collection Services
  - Call Stacks
  - SQL Statements
  - Additional wait accounting information:
    - Objects being waited on
    - Holder of object
- More frequent intervals (seconds)
- Need to start/stop Job Watcher
  - Navigator for i, iDoctor, green screen commands
- To see charts in PDI, need Performance Tools LPP Job Watcher option (chargeable) or iDoctor Job Watcher license for viewing in iDoctor

Job Watcher - Holders versus Waiters

- IBM i keeps track of who is holding a resource, and if applicable, who is waiting to access that resource
  - A Holder is the job/thread/task that is holding the serialized resource
  - A Waiter is the job/thread/task that wants to access the serialized resource
- IBM i also maintains call stacks for every job/thread/task
- The combination of
  - Who - holders and waiters ... who has it? who wants it?
  - What – object being waited on
  - How - call stacks
  provides a very powerful solution for analyzing wait conditions
Job Watcher – Where to Start

Performance -> Investigate Data -> Job Watcher:

Option 1: CPU Utilization and Waits Overview
– Combines related waits into higher level buckets

Option 2: Waits Overview
– All individual “blocked” wait buckets shown

Notice similar perspectives available as Collection Services

---

Job Watcher – Waits Overview

Notice same wait buckets, but more granular intervals
Job Watcher – Additional Interval Details

Thread or Task Details
Job information: QZDIAGSET/QGRS/120992 - 300000000000000015
Current user profile: LSEAW
Object waited on: INVENTORY INVENTORY
Wait duration: 981 milliseconds
Current or last wait: DB record lock: update
Holding job or task: QZDIAGSET/QGRS/120990

Call Stack

SQL Statement:
- Include host variables
SELECT QUANTITY FROM WAREHOUSE.Inventory WHERE ID='DATA FORMAT ERROR' FROM WAREHOUSE.Inventory WHERE ID='UPDATE'

Wait Accounting: Be proactive!

- Use the rich IBM i wait accounting instrumentation found in
  - Collection Services & Job Watcher
  - Use PDI or iDoctor to view/analyze

- Understand your partition’s “run-wait” signature and normal patterns

- Identify bottlenecks
**Recommendations: Be proactive!**

- **Keep a baseline**
  - Collection Services (Job Watcher data is also nice to have)
    - Weekly, end-of-month, end-of-year
    - Prior to any hardware, software, configuration related change

- **A baseline provides a reference point**
  - It is the expected performance characteristics over a defined period of time
  - Having one makes it easier to recognize changes and its effect

- Wait bucket information can make it easier to determine what has changed! Both at a partition level as well as an individual job level

---

**IBM i Performance Data Collectors**
Performance Data Collection Architecture

- Collection Services
- Job Watcher
- Disk Watcher
- Performance Explorer

Collect System-wide Performance Data

Collect Performance Data 24/7

If something goes wrong, you have data that will help analyze the problem, fix it, and prevent it from happening in the future.

- If you can’t solve the problem, you have information that makes it easier for IBM Support to solve the problem faster.
- To provide a reliable baseline so you can understand the impact that a software, network, or environmental change had on the performance of your system.
- To provide historical information that enables you to plan for future growth based on real trends, not guesses.
Patterns in Performance Data

- Performance data typically has patterns
  - Daily, weekly, monthly, yearly

**Understand your typical patterns**

- Recognize change
Job Watcher

• Job Watcher returns real-time information about a selected set of jobs, threads, or LIC tasks

• Job Watcher collects additional types of data that Collection Services does not, as well as more frequent intervals
  – Job Watcher has more overhead than Collection Services

• Data collected by Job Watcher includes
  – Wait times
  – CPU
  – I/O activity
  – Call Stacks
  – SQL statements
  – Communications statistics
  – Activation Group statistics

Run Job Watcher when you need detailed performance data for diagnostic purposes.

There are clients that run Job Watcher 24x7 to always have diagnostic data available.

Need to manage the data carefully.

Job Watcher

• Job Watcher collects more detailed performance data than Collection Services and at more frequent intervals
  – CPU and I/O (like Collection Services)
  – Call Stacks
  – SQL Statements
  – Detailed Wait information:
    • Objects being waited on, even records number of files
    • Holder of object

• Job Watcher does not collect everything that Collection Services collects.
  – It does not always collect information about every thread
    • Thread must use CPU during interval
    • Thread must exist for entire interval
  – It does not collect memory pool or detailed I/O statistics

• Data is written to DB2 files
Job Watcher Usage Tips

- Use Job Watcher when you need detailed performance data to resolve a problem
  - Typically problem has been scoped first by Collection Services
  - Consider using a definition like the IBM-supplied Q10SECSQL (10 second intervals, gathers Call Stacks and SQL if it can) as a starting point

- For problem determination Job Watcher can be run on specific jobs
  - Caution: When using Job Watcher on specific job(s), you may not get detailed Holder information

- Multiple collections can be run at the same time

- Need to manage the amount of data collected
  - Specify Maximum Disk Space to consume on Start
  - Will stop collecting if specified ASP threshold is reached
  - iDoctor Monitors will clean up after themselves

Basic Job Watcher Data Collection Steps

- Select an IBM-supplied definition to use
  - Or create your own custom definition

- Start the Job Watcher collection

- Let it run until the problem has occurred

- Stop the Job Watcher collection

- Analyze the data

- There may be times when you want to run Job Watcher for long periods of time
  - Consider 1 hour collection size
  - Manage size of data
Job Watcher in Navigator for i

- Navigator for i Web console provides a GUI for Job Watcher
  - Requires the Performance Tools LPP, Job Watcher option to be installed

IBM-Supplied Job Watcher Definitions

- Many pre-defined Job Watcher definitions are available
  - The main difference is the sample interval
- Recommendations:
  - Collect with Call Stacks and SQL
  - Use 10 second intervals for general analysis (Q10secsql)
  - Use 5 second intervals for complex or intermittent issues, or for contention related problems (Q5secsql)
## Job Watcher Definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1sec</td>
<td>IBM-supplied</td>
<td>1 second intervals, Call stacks</td>
</tr>
<tr>
<td>Q1secj</td>
<td>IBM-supplied</td>
<td>1 second intervals, Call stacks, J9</td>
</tr>
<tr>
<td>Q1secsqj</td>
<td>IBM-supplied</td>
<td>1 second intervals, Call stacks, Sql</td>
</tr>
<tr>
<td>Q1secsqj</td>
<td>IBM-supplied</td>
<td>1 second intervals, Call stacks, Sql, J9</td>
</tr>
<tr>
<td>Q10sec</td>
<td>IBM-supplied</td>
<td>10 second intervals, Call stacks</td>
</tr>
<tr>
<td>Q10secj</td>
<td>IBM-supplied</td>
<td>10 second intervals, Call stacks, J9</td>
</tr>
<tr>
<td>Q10secsqj</td>
<td>IBM-supplied</td>
<td>10 second intervals, Call stacks, Sql</td>
</tr>
<tr>
<td>Q10secsqj</td>
<td>IBM-supplied</td>
<td>10 second intervals, Call stacks, Sql, J9</td>
</tr>
<tr>
<td>Q3minqzdas</td>
<td>IBM-supplied</td>
<td>QZDASQINIT jobs, 3 min intervals</td>
</tr>
<tr>
<td>Q5minqzdas</td>
<td>IBM-supplied</td>
<td>QZDASQINIT jobs, triggers PEX stats</td>
</tr>
<tr>
<td>Q5sec</td>
<td>IBM-supplied</td>
<td>5 second intervals, Call stacks</td>
</tr>
<tr>
<td>Q5secj</td>
<td>IBM-supplied</td>
<td>5 second intervals, Call stacks, J9</td>
</tr>
<tr>
<td>Q5secsqj</td>
<td>IBM-supplied</td>
<td>5 second intervals, Call stacks, Sql</td>
</tr>
<tr>
<td>Q5secsqj</td>
<td>IBM-supplied</td>
<td>5 second intervals, Call stacks, Sql, J9</td>
</tr>
</tbody>
</table>

### Job Watcher Definition

**Add Job Watcher Definition**

- **Basic Options**
  - **Definition name:**
  - **Description:**

- **Data collection options**
  - **Include call stacks**
  - **Include SQL statements**

- **Collection interval**
  - **User-defined (0.1 - 3600.0) 10 seconds**
  - **No delay** Data is collected as fast as possible **This is resource intensive**

© 2017 IBM Corporation
Start Job Watcher

- Some of the parameters on the definition can be overridden when starting the Job Watcher collection
  - Allows one definition to be use for multiple types of collections
Give some thought to these!
Active Job Watcher Collections

Stop the Collection
How Do I Run Job Watcher with the Commands?

- Add Job Watcher Definition (ADDJWDFN) to define the collection
  - Identifies the performance data that is to be collected

- Remove Job Watcher Definition (RMVJWDFN) to remove a definition
  (Note: Job Watcher Definitions can only be displayed through the GUI)

- Start Job Watcher (STRJW) to start the collection
- End Job Watcher (ENDJW) to end the collection (optionally)

Running Job Watcher Continuously

- If you have a problem but Job Watcher is not running, you will need to recreate the problem to capture Job Watcher data

- Many clients prefer to run Job Watcher 24x7
  - Ensures you have diagnostic data the first time the problem occurs

- Options:
  - QMGTOOLS has the ability to run Job Watcher
  - iDoctor monitors
  - Your own CL program
Running Job Watcher Continuously - example

- Create the objects and the job watcher data in a your own library – e.g., CRTLIB JOBWATCH

- Create a dataarea to control how many collections you want to have saved. In the STRJW command, we use the RPLDTA parameter to automatically replace the older collections with the same name. This way we don't have to worry about deleting older collections - they are simply overwritten when the new one with the same name is started.

- CRTDTAARA DTAARA(JOBWATCH/JOBWATCH) TYPE(*DEC) LEN(15 5) VALUE(S)

- When the following program is called with the data area at 5 it will save 5 collections that have the names JWCOL1 .. to JWCOL5.

- "Note: The sample program has a hardcoded "5" in it for the reset number when the counter goes to 1 - if you want to have more than 5 collections saved, you need to have that 5 changed in the little CLP. A simple improvement to this program would be to save the value retrieved from the data area and reset to that value rather than a hard-coded number.

- In this example, we used the system supplied definition Q5SECSQL - you'll may want to set up your own definition and use it when you do the STRJW.

- Every hour a new JW collection will be started. The collection will run for 65 minutes (to have a 5 minute overlap for starting the next one and the previous one ending). This program will loop forever, keeping 5 job watcher collections, until the program is canceled.

- Submit the job to batch and let it run; end the batch job when you want to end your data collections. The last job watcher collection will remain active until it completes.

Sample CL Program – Provided AS IS

```
PGM DCL        VAR(&DECNUMBER) TYPE(*DEC) LEN(15 5)
DCL        VAR(&NUMBER) TYPE(*UINT) LEN(2)
DCL        VAR(&COLNAME) TYPE(*CHAR) STG(*AUTO) LEN(10) + VALUE(JWCOL)
ADDLIB JOBWATCH
MCHMSG: CPF9999
BEGIN:
RTVDTAARA DTAARA(JOBWATCH/JOBWATCH *ALL) + RTNVAR(&DECNUMBER)
CHGVAR     VAR(&NUMBER) VALUE(&DECNUMBER)
IF         COND(&DECNUMBER *EQ 1) THEN(CHGVAR + VAR(&DECNUMBER) VALUE(5))
ELSE       CMD(CHGVAR VAR(&DECNUMBER) VALUE(&DECNUMBER + - 1))
CHGDTAARA DTAARA(JOBWATCH/JOBWATCH *ALL) + VALUE(&DECNUMBER)
CHGVAR     VAR(&COLNAME 6 2) VALUE(&NUMBER)
STRJW      DFN(Q5SECSQL) COL(&COLNAME) LIB(JOBWATCH) + RPLDTA(*YES) ENDCOL(*NBRSEC 3900)
DLYJOB 3600
GOTO BEGIN
ENDPGM
```
Job Watcher Authority Requirements

- **Commands:**
  - You must have service (*SERVICE*) special authority
    - Change User Profile to add *SERVICE* authority to create Job Watcher Definitions or to Start Job Watcher
  - *-OR-* Be authorized to the Job Watcher function of the operating system
    - Change Function Usage (CHGFCNUSG) command, with a function ID of QIBM_SERVICE_JOB_WATCHER can be used to change the list of users that are allowed to use this command.
      - `CHGFCNUSG FCNID(QIBM_SERVICE_JOB_WATCHER) USER(<usrprofile>) USAGE(*ALLOWED)`

- **Definitions:**
  - Additional authority is needed to see the definitions for each as they are shipped with public authority *EXCLUDE*. To see the definitions shipped in Job Watcher, users will need authority to the QAPYJWDFN file in QUSRYSYS


---

Job Watcher – PDI interface

- Job Watcher
  - CPU Utilization and Waits Overview
  - CPU Utilization by Thread or Task
  - Resource Utilization Overview
  - Job Statistics Overviews
  - Waits
  - CPU
  - Physical Disk I/O
  - Synchronous Disk I/O
  - Storage Allocation
  - Page Faults
  - Logical Database I/O
  - Sys250 Display Transactions
  - Job Watcher Database File

[Graph showing CPU Utilization and Waits Overview]
Performance Explorer

• Performance Explorer helps identify the causes of performance problems that cannot be resolved using one of the other performance data collectors
  – Collects more detailed information about a specific application, program, or resource

• Performance Explorer is typically used for two main reasons:
  – Detailed performance trace data is needed to identify the performance problem
  – Analyzing the performance of applications

• Performance Data Investigator supports profile collections only
• iDoctor is required for advanced PEX Analysis

Performance Explorer

• Performance Explorer is the most sophisticated IBM i performance tool
  – Can collect the details of every I/O operation, every task switch
  – Hundreds of events collected
  – Thus, most complex to use
  – More overhead

• Typically, problem has been scoped by other tools first

• Generally used by IBM performance analysis experts
  • Except.....
Performance Explorer – “TPROF” usage

- “Trace-profile” is a fairly easy, and fairly “light-weight” PEX collection that can be useful to application developers, especially when trying to diagnose high CPU issues
  - Provides CPU usage at a program/module/procedure level
  - Make sure you have latest PEX PTFs applied
    [Link](http://www-01.ibm.com/support/docview.wss?uid=nas8N1012020)
  - Can be run over subset of jobs

- Steps:
  1. Add a PEX definition:
     ```
     ADDPXD FN DFN(TPROF)
     TYPE(*TRACE)
     JOB(*ALL *ALL)
     TASK(*ALL)
     MAXSTG(4000000)
     INTERVAL(1)
     TRCTYPE(*SLTEVT)
     SLTEVT(*YES)
     MCHINST(*NONE)
     BASEVT(*PMCO *NONE *FORMAT2)
     ```

2. Collect data

   - Start Performance Explorer (STRPEX)
   - Run the command, program, or workload that you want to have for data analysis
   - End Performance Explorer (ENDPEX)

- Repeat as necessary
### Performance Explorer TPROF reports – PDI

#### Program Name | Module Name | Procedure Name | Component | Hit Count
---|---|---|---|---
GETENV | db_custom_parmflag | SQL Common Funds | | 332(46.61%) |
GETENV | db_custom_parmflag | SQL String Functions | | 853(16.47%) |
GETENV | db_custom_parmflag | SQL Database | | 85(12.42%) |
GETENV | db_custom_parmflag | SQL Other | | 492(50.55%) |
READER | READER | READER | Other | 272(30.95%) |
SMSCD | dsmc_adsr-sunos Growing disk list | SQL Common Funds | | 6(0.07%) |
SMSCD | dsmc_adsr-sunos Growing disk list | SQL String Functions | | 6(0.07%) |
SMSCD | dsmc_adsr-sunos Growing disk list | SQL Database | | 5(0.06%) |
SMSCD | dsmc_adsr-sunos Growing disk list | SQL Other | | 2(0.02%) |
SMSCD | dsmc_adsr-sunos Growing disk list | Other | | 7(0.08%) |

#### Task/Thread List

<table>
<thead>
<tr>
<th>Select</th>
<th>Task/Thread Name</th>
<th>Cumulative CPU Time in Seconds</th>
<th>Active Time in Seconds</th>
<th>Hit Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>145.59193</td>
<td>221551.17</td>
<td>50904550.93(99.99%)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>OPTCSPD</td>
<td>60.429</td>
<td>123151.72</td>
</tr>
</tbody>
</table>

#### Hierarchical Trace Profile

<table>
<thead>
<tr>
<th>Select</th>
<th>Name</th>
<th>Hit Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>683(100%)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>LIB</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>LIB Code Reuse</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>CUSTOMER CUSTOMER</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CUSTOMER CUSTOMER</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Amount of elements that did not match the filter</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>PEE 8.3</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>PEE 8.3</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

---

Examples

© 2017 IBM Corporation
What is causing disk wait time?
Reviewing wait times with Collection Services

Let’s Look at the Disk Waits

That mountain of green is Disk Time. What caused it?
We see it’s faulting…. let’s find out who did it

Drill down into “Waits by Job or Task”.

We can see it is the QRWTSRVR DDM/DRDA server jobs.

Let’s find out who the user is

• We now have several clues:
  – We know the jobs - QRWTSRVR - DDM/DRDA server jobs
  – We know the time - early afternoon
  – We know the user profile - QUSER
  • But QUSER isn’t helpful. We need the job’s current user profile
  – Waits by Job Current User Profile shows us VCPANYLT is the guilty party
Let’s look at the entire collection

- CPU Utilization and waits gives an excellent overview of the entire partition
- There are a lot of interesting things to investigate.....
Zoom into the time where we see a large drop in CPU Utilization

We can see operating system contention occurred during the time when the CPU Utilization dropped.

Drill-down based upon what you see

Select the beginning and ending intervals to investigate and then drill into Contention Waits Overview. Be sure to select the metric you are investigating.
Machine level gate serialization is a major reason for the contention waits. The pink is semaphore contention (see the flyover). This is usually an uninteresting wait.

We want to see if we can figure out who might be causing the contention. Drill into All Waits by Thread or Task Sorted by Machine Level Gate Serialization so we can see the jobs/threads/tasks that are all waiting. We selected Machine Level Gate Serialization because that's the predominate “bad” wait our graph shows us.

Note: Drilling into waits by thread or task can take some time... be patient.
Zoom into see more detail

We can’t see the machine level gate serialization details at first; Zoom in and we can see it appear in many threads. This tells us many threads were waiting.

But why?

Select a thread and look at the waits for that one thread

It may be necessary to drill down into interval details for several threads to find the one with the information we need…
Select an interval
View Interval details for one thread or task

All Waits for One Thread or Task

And here is where we discover Job Watcher’s power....
We can review the call stack to see how we got to this wait point.

This thread is waiting for the QAUDJRN journal at 8:51:05.

In the call stack you will see an entry that shows the job is creating an audit journal entry.

Note that access to the audit journal is serialized by a “gate”. So why is this job blocked and waiting to create the audit record?

**Call Stack**

```
<table>
<thead>
<tr>
<th>Call Level</th>
<th>Program</th>
<th>Module</th>
<th>Procedure</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>quide_block_trace</td>
<td></td>
<td>00000000E4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c4</td>
<td></td>
<td>0000003F0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>fjournal</td>
<td></td>
<td>0000066C</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>fjournal</td>
<td></td>
<td>00000444</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>fjournal</td>
<td></td>
<td>00000000E4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>fjournal</td>
<td></td>
<td>000003F0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>fjournal</td>
<td></td>
<td>0000066C</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>fjournal</td>
<td></td>
<td>00000444</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>fjournal</td>
<td></td>
<td>00000000E4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>fjournal</td>
<td></td>
<td>000003F0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>fjournal</td>
<td></td>
<td>0000066C</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>fjournal</td>
<td></td>
<td>00000444</td>
<td></td>
</tr>
<tr>
<td>Total: 28</td>
<td></td>
<td></td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>
```

We can easily go look at the thread that is holding the resource

The call stack shows a “Change Journal” program (QJOCHJN)
Since we know the object that was causing the contention was QAUDJRN, let’s have a look....

Note the audit journal entries from the matching time period. Observe the job name...

NR is Next Receiver
PR is Previous Receiver

What next?
- We observed significant system-wide machine gate serialization contention
- With Job Watcher, we determined that at that point in time, the QAUDJRN was changing journal receivers
- Perhaps we should review our security audit settings
- In this example, it was a stress-test System Test partition with a heavy workload and extreme settings
Viewing Waits with Job Watcher

Example of Object Lock Contention

Once again, we start with CPU Utilization and Waits Overview
Observe the gray... “Lock Contention Time”
Drill into Seizes and Locks Waits Overview

Tip: Drill-downs are named to drill into what you see

- This confirms we have some job holding a lock that another job is waiting on

Seizes and Lock Waits Overview ➔ All Waits by Thread or Task...

Drill down into All Waits by Thread or Task Sorted by Seizes and Lock Waits
We can clearly see the thread that is waiting for the lock.

Select that thread and drill down to All Waits for One Thread or Task

We are now to the magic drill-down...
Select the interval for that job and drill down to
Interval Details for One Thread or Task
Interval Details

Interval Details for One Thread or Task (Interval = '47', Initial Thread Task Count = '2549')

Thread or Task Details
- Job information: QSCLICEV/QUSER/014614 - 0000000000000001
- Current user profile: QSYS
- Object waited on: WAITEVENTSPACE
- Wait duration: 4,482 milliseconds
- Current or last wait:
- Holding job or task: QZERCSRV5/QUSER/014697

Priority: 25
Pool: 2
Type description: TEMPORARY - SPACE
Segment type description: BASE M SYSTEM OBJECT
SQL client job: None detected this interval
Interval timestamp: Dec 13, 2007 2:48:00 PM
Interval (1 to 91): < 47 >

Call Stack

More PDI Examples
Java Perspectives
Drilldown for one job -
Look at the heap and memory usage over time for one selected job.

Database Full Opens
Full Opens are expensive resource-wise

General recommendation is to keep Native Full Opens per second < 1000

Next, find jobs doing full opens...
Database Full Opens

In an RPG program, full opens are caused by the use of SETON *LR instead of RETRN. Avoid if possible. Not setting on LR (in OPM program) will keep the program in memory, keeps file open and pointer set, retains variable values, etc. Also, avoid *NEW for the ACTGRP.

Shared file opens are far less expensive than full file opens. They consume less CPU, less storage and are faster than full opens. There are some implications of changing full file opens to shared file opens, but implementation of shared opens is typically easier to manage and implement than changing SETON LR to RETRN in RPG programs. The following links provide explanation, usage and considerations of using shared ODPs:

- Sharing database files in the same job or activation group
  http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/topic/dbp/rbafosfile.htm
- Open considerations for files shared in a job or an activation group
  http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/topic/dbp/rbafopenc.htm
- Input/output considerations for files shared in a job or an activation group
  http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/topic/dbp/rbafiocon.htm
- Close considerations for files shared in a job or an activation group
  http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/topic/dbp/rbafoclose.htm

Java Perspectives in Collection Services

Find that job using a lot of heap...
Temporary Storage Allocation/Deallocation Perspectives

Storage Allocation Perspectives

Where is my temporary storage going?

Expand Collection Services

Temporary Storage Allocation / Deallocation Overview

Generally, allocations and deallocations following a similar pattern

© 2017 IBM Corporation
From an overview perspective, drill down to more detail

What does the faulting look like when I was testing?
IBM iDoctor for i

- Product developed by the IBM Rochester Support Center for deep, detailed performance analysis

- Major components
  - Job Watcher
    - Job Watcher
    - Collection Services Investigator
    - Disk Watcher
    - Plan Cache Analyzer - graphical analysis of the system’s SQL Plan Cache
  - PEX Analyzer
  - VIOS Investigator

- [https://www-03.ibm.com/systems/services/labservices/iDoctor/](https://www-03.ibm.com/systems/services/labservices/iDoctor/)
IBM iDoctor for i

iDoctor Monitors

© 2017 IBM Corporation
iDoctor versus Performance Data Investigator

- You have two graphical interfaces for performance data analysis...
  Which should you use? It depends....

<table>
<thead>
<tr>
<th>Feature</th>
<th>iDoctor</th>
<th>PDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Windows Client</td>
<td>Browser</td>
</tr>
<tr>
<td>Wait Analysis</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Collection Services</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Job Watcher</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Disk Watcher</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance Explorer</td>
<td>Yes</td>
<td>Profile collections only</td>
</tr>
<tr>
<td>Database</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Job Watcher Monitors</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Customizable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User Defined graphs and queries</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Update Frequency</td>
<td>Monthly</td>
<td>Twice Yearly</td>
</tr>
<tr>
<td>Support</td>
<td>Defect only</td>
<td>Standard SWMA</td>
</tr>
<tr>
<td>Chargeable</td>
<td>Nearly License</td>
<td>Property Services at no additional charge with iDoctor</td>
</tr>
<tr>
<td>Experimental Features</td>
<td>Yes(e.g., VIOS Investigator)</td>
<td>No</td>
</tr>
<tr>
<td>Multinational language support</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
www.ibm.com/power/i

References
IBM i Performance FAQ - a MUST read!


IBM i on Power - Performance FAQ
April 3, 2017

© 2017 IBM Corporation

IBM i Web Sites with Performance Information

IBM i Knowledge Center
http://www.ibm.com/support/knowledgecenter/ssw.ibm_i/welcome
  - 7.3 - https://www.ibm.com/support/knowledgecenter/ssw.ibm_i.73/rzahg/welcome.htm

IBM i Performance Management
http://www-03.ibm.com/systems/power/software/i/management/Web3

• Performance Management for Power Systems
  http://www-03.ibm.com/systems/power/support/prv/index.html

• IBM Workload Estimator
  http://www.ibm.com/systems/support/tools/estimator

• iDoctor

• Job Waits Documentation

© 2017 IBM Corporation
Understanding “Time Dispatched on a CPU”

- **Time dispatched on a CPU (Bucket 1)**
  - Thread or task has been assigned to a processor and is NOT waiting
  - Complicated by certain features
    - Hardware Multi Threading (HMT)
      - Allows multiple threads/tasks to be assigned to a single physical processor
      - Causes bucket 1 time to be greater than actual CPU time
    - Background assisting tasks
      - Promote their CPU usage back into the client job/thread
      - Causes client thread's bucket 1 time to be smaller than measured CPU time
    - LPAR shared/partial processors
      - Bucket 1 records time dispatched to the virtual processor
      - Bucket 1 time may be greater than CPU time because it may include time the thread/task is waiting for the physical processor behind the virtual processor

**Bucket 1 does NOT equal CPU Time**

Understanding “CPU Queuing”

- **CPU Queuing (Bucket 2)**
  - Thread or task has been assigned to a processor and is waiting for the CPU to become available
    - Too much work on the partition causing threads to need to wait for the processors
    - Spiky workloads
    - Workload Groups
    - Shared processors
      - Latency due to hypervisor sharing the physical processors among multiple partitions
CPU Queueing ... technical details

- A spikey workload. I/O completing in batches can cause this but so can software design. For instance, there are some seize-lock scenarios that unblock herds of tasks. A task switch trace can be used to instrument how the workload is behaving. This is a common reason.

- Shared processors. When there are idle processors, tasks are enqueued directly to a special per-processor queue until they overflow to a shared queue. On a dedicated processor system, the time it takes for the wait-state task to dequeque the task from this special queue and dispatch it is very low, but non-zero. The latency can be slightly higher when donation is used. In a SLPAR, the latency can vary depending on outside factors, namely shared pool utilization, because the hypervisor is sharing the physical processors among multiple partitions. The separate components of VCPU latency is instrumented in Collection Services as of 7.1.

Nodal dispatching.

In a multi-node system, load imbalance could cause tasks to be queued in some nodes while other nodes are idle. This is generally not a problem with the default nodal affinity scheduling policy. However, a QTHDRSCAFN value set to *HIGH instead of the default *NORMAL can cause this problem. Node imbalance can be instrumented in 6.1 and later.

- Workload capping. The workload group can be over-committed even though the system is under-committed. There is separate instrumentation of workload group dispatch latency in Collection Services.
IBM i Performance on developerWorks

- developerWorks
- Performance Tools
- IBM i Performance Data Investigator
- IBM i Performance Data Investigator – Edit Perspectives
- IBM i Wait Accounting
- How to use the Batch Model performance tool
- A new way to analyze historical performance data on IBM i

For a simple list of all blogs on one page:

“i Can” Blog of Blogs
[https://www.ibm.com/developerworks/community/blogs/i-can](https://www.ibm.com/developerworks/community/blogs/i-can)

© 2017 IBM Corporation
IBM i Performance Analysis Workshop

Managing and analyzing the data can be quite complex. During this workshop, the IBM Systems Lab Services IBM i team will share useful techniques for analyzing performance data on key IBM i resources, and will cover strategies for solving performance problems. It will aid in building a future foundation of performance methodology you can apply in your environment.

Overview:

- **Topics covered include:**
  - Key performance analysis concepts
  - Performance tools
  - Performance data collectors (Collection Services, Job Watcher, Disk Watcher, and Performance Explorer)
  - Wait accounting
  - Core methodology and analysis of:
    - Locks
    - Memory
    - I/O subsystem
    - CPU
  - Concept reinforcement through case studies and lab exercises
- Discussions on theory, problem solving, prevention and best practices

Workshop details:

- Intermediate IBM i skill level
- 3-4 day workshop, public or private (on-site)
- For public workshop availability and enrollment: IBM i Performance Analysis Workshop
- For additional information, including private workshops, please contact Eric Barsness at ericbar@us.ibm.com or Stacy Benfield at stacylvus.ibm.com, members of Systems Lab Services Performance Optimization team.

IBM i Performance and Optimization Services

The IBM i Performance and Optimization team specializes in resolving a wide variety of performance problems. Our team of experts can help you tune your partition and applications, including:

- Reducing batch processing times
- Resolving SQL query and native I/O performance problems
- Tuning RPG, COBOL, C, and Java (including WebSphere Application Server) programs
- Removing bottlenecks, resolving intermittent issues
- Resolving memory leaks, temporary storage growth problems, etc.
- Tuning memory pools, disk subsystems, system values, and LPAR settings for best performance
- Optimizing Solid State Drive (SSD) performance
- Tuning client interfaces such as ODBC, JDBC, .Net and more

Skills transfer and training for performance tools and analysis also available!

Contact Eric Barsness at ericbar@us.ibm.com for more details.
Performance and Scalability Services

- The IBM i Performance and Scalability Services Center can provide facilities and hardware IN ROCHESTER to assist you in testing hardware or software changes
  - "Traditional" benchmarks
  - Release-to-release upgrades
  - Assess and tune application and database performance
  - Stress test your system
  - Determine impact of application changes
  - Proofs of Concept (e.g. HA alternatives; SSD analysis, external storage, etc.)
  - Evaluate application scalability
  - Capacity planning

- ... all with the availability of Lab Services IBM i experts and development personnel

To learn more about Power offerings:
https://www.ibm.com/it-infrastructure/services/lab-services/power

Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of these products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquiries, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied. IBM is not responsible for printing or information inaccuracies.

All prices shown are IBM’s United States suggested list prices and are subject to change without notice. Reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software configuration. Some measurements quoted in this document may have been estimated through extrapolation. Uses of this document should verify the applicable data for their specific environment.

Revised September 26, 2006