IBM System z Advanced Workload Analysis Reporter (IBM zAware) implementation in zPET

Introduction

The IBM System z Advanced Workload Analysis Reporter (IBM zAware) provides a smart solution for detecting and diagnosing anomalies in z/OS systems. IBM zAware creates a model of normal system behavior based on existing previous system data, and uses pattern recognition techniques to identify unexpected messages in current data from the z/OS systems that it is monitoring. This analysis of events provides nearly real-time detection of anomalies that you can easily view through a graphical user interface (GUI). You also use the GUI to diagnose the cause of past or current anomalies.

In order to use IBM zAware, you set up a specialized logical partition (LPAR) that is dedicated to running the IBM zAware server. This LPAR runs on an IBM zEnterprise EC12 (zEC12) central processor complex (CPC).

In this article we talk about our experiences with configuring IBM zAware in zPET. This article is not meant to be an introduction to IBM zAware or step by step instructions on how to implement IBM zAware. For complete and detailed instructions please see System z Advanced Workload Analysis Reporter (IBM zAware) Guide, SC27-2623-00.

Hardware Resources

The following are the hardware resources we used.

Processor storage resources

The *IBM zAware Guide* says that you must allocate a minimum of 4 GB of processor storage to activate the IBM zAware partition and that 6 GB is sufficient to support 6 or fewer clients, so we decided to go with 1 GB per monitored LPAR.

When we initially configured IBM zAware we only had 9 LPARs within our sysplex so we assigned 9 GB to our IBM zAware partition. Currently we are up to 24 LPARs in our sysplex. All 24 are connected to IBM zAware and we have not had any issues related to processor storage.

Unfortunately, there is no way to tell processor storage utilization from within the IBM zAware user interface or externally from traditional monitors like RMF or Tivoli Omegamon. We have opened a marketing requirement to offer this functionality in a future IBM zAware release.

DASD storage resources

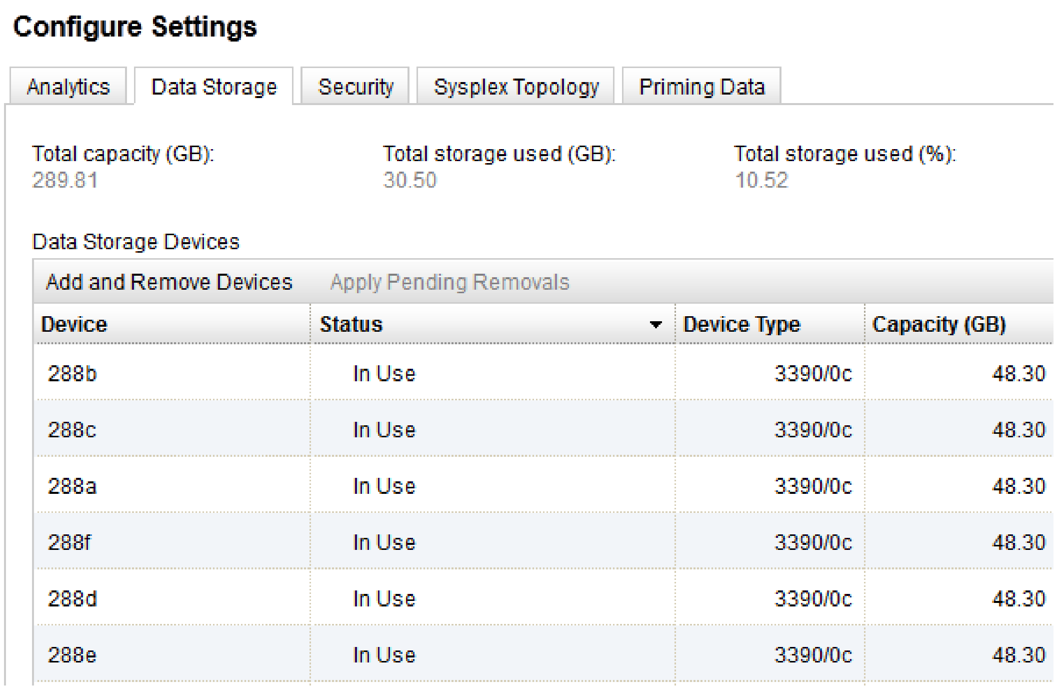
IBM zAware uses Extended Count Key Data (ECKD) direct-access storage devices (DASD) for persistent storage of analytical data for each monitored client. These volumes can not be SMS-managed and because of the way that the IBM zAware server uses storage, you need to configure these devices so that no other partitions can use them.

Storage requirements vary depending on the retention times for each type of analytical data – adjustable through the IBM zAware graphical user interface (GUI) – and on the number of monitored systems that you plan to connect to IBM zAware.

The *IBM zAware Guide* suggests that you start with 500 GB of DASD storage. During our testing this recommendation was not yet available and we chose to go with 300 GB across 6 volumes.

Currently we have 24 systems connected to IBM zAware. Nineteen of these systems have approximately 60 days worth of SYSLOG data within the database while the remaining 5 have approximately 7 days worth. We are using only about 11% of our 300 GB DASD storage.

However, please note that 15 out of 24 of our systems are relatively new to our sysplex and the amount of SYSLOG data they generate is relatively small. We also used only 2 weeks worth of SYSLOG data to train IBM zAware for each client, rather than the default of 90 days. Therefore, we recommend that you follow the *IBM zAware Guide* recommendation and begin with 500 GB, then monitor your DASD storage utilization and make adjustments as necessary. DASD storage utilization can be seen in the Configure Settings panel within the IBM zAware (GUI) as shown in Figure 1 below.



Total capacity (GB): 289.81

Total storage used (GB): 30.50

Total storage used (%): 10.52

Data Storage Devices

Device	Status	Device Type	Capacity (GB)
288b	In Use	3390/0c	48.30
288c	In Use	3390/0c	48.30
288a	In Use	3390/0c	48.30
288f	In Use	3390/0c	48.30
288d	In Use	3390/0c	48.30
288e	In Use	3390/0c	48.30

Figure 1: DASD devices dedicated to IBM zAware

Processor resources

IBM zAware Guide recommends allocating two shared IFLs or CPs for the IBM zAware LPAR. We chose to allocate two shared CPs.

In our environment, with 24 LPARs connected to IBM zAware, we observed that the average CPU utilization of the partition ranged anywhere between 0.7 to 2.8 %.

z/OS monitored client configuration

Remember that IBM zAware supports z/OS systems that run in z/OS partitions or as z/VM guests. The number of clients is limited only by the resources assigned to the IBM zAware partition.

One of the requirements for z/OS monitored clients is that they use the operations log (OPERLOG) as the hardcopy medium. We have been using OPERLOG since it became available a number of years ago.

OPERLOG in zPET

OPERLOG is a base MVS function which uses system logger services to create a merged SYSLOG of every system in the sysplex.

Step by step instructions and details on setting up OPERLOG are documented in z/OS Problem Management, G325-2564-09.

At a high level the setup for OPERLOG requires the following:

- A logstream and a structure definition in the system logger policy. In zPET:

```
LOGSTREAM NAME(SYSPLEX.OPERLOG)
STRUCTNAME(LOGGER_OPERLOG)
LS_DATACLAS() LS_MGMTCLAS() LS_STORCLAS()
HLQ(OPERLOG) MODEL(NO) LS_SIZE(4000)
STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
LOWOFFLOAD(20) HIGHOFFLOAD(80) STG_DUPLEX(NO) DUPLEXMODE()
RMNAME() DESCRIPTION() RETPD(0) AUTODELETE(NO) OFFLOADRECALL(YES)
ZAI(YES) ZAIDATA('OPERLOG') DASDONLY(NO) DIAG(NO)
LOGGERDUPLEX(UNCOND) EHLQ(NO_EHLQ) GROUP(PRODUCTION)
```

- A structure definition in the CFRM policy. In zPET:

```
STRUCTURE NAME(LOGGER_OPERLOG) SIZE(110592)
INITSIZE(105472)
PREFLIST(CF1,CF2,CF3)
REBUILDPERCENT(1)
FULLTHRESHOLD(95)
DUPLEX(ENABLED)
```

- A parameter setting in CONSOLxx parmlib members to enable the function:

```
HARDCOPY DEVNUM(SYSLOG,OPERLOG)
```

The OPERLOG can be turned off/on with the commands:

```
V OPERLOG,HARDCPY,OFF
V OPERLOG,HARDCPY
```

Note that `HARDCPY` is not a typo.

Viewing OPERLOG data

The data in the OPERLOG logstream is viewable from TSO through the 'LOG' function of SDSF or (E)JES.

Off-loading OPERLOG data

In zPET the OPERLOG data in the logstream is off-loaded daily to a Generation Data Group (GDG) data set:

```
LOGWRTR.LOGPET.GxxxxV00
```

The off-load job is submitted through Tivoli Workload Scheduler every day at 2:00 AM.

The job uses the **IEAMDBLG** utility (available in **SYS1.SAMPLIB**) and is set up to copy and then delete all the data in the log stream that is greater than 0 days old (that is, 24 hours of data from the day before).

Here are the parameters zPET uses to execute this utility:

```
//STEP1 EXEC PGM=IEAMDBLG,
// PARM='COPY(>0),DELETE(>0)'
//SYSLOG DD DSN=LOGWRTR.LOGPET(+1),
// DISP=(NEW,CATLG),
// UNIT=LOGS,
// DSNTYPE=LARGE,
// SPACE=(TRK,(35000,5000),RLSE),
// DCB=(LOGWRTR.MODEL.DSCB)
```

z/OS system logger

z/OS system logger on the monitored clients must be configured to send data to the IBM zAware server. Details are listed in:

- *z/OS MVS Setting Up a Sysplex*, SA22-7625-22, including "Preparing for z/OS IBM zAware log stream client usage."
- Parmlib member descriptions in *z/OS MVS Initialization and Tuning Reference*, *SA22-7592-25*.

In zPET we edited the IXGCNF00 parmlib member to define IBM zAware server communication details:

```
ZAI  SERVER(xx.xx.xx.xxx) PORT(2001)
```

where `xx.xx.xx.xxx` is the IBM zAware server IP address. Note that you can use the `SET IXGCNF=xx` command to dynamically apply the changes to the IXGCNF parmlib member.

We also had to update our OPERLOG log stream IBM zAware related parameters. We used the following JCL:

```
//OZIXC JOB MSGCLASS=H,CLASS=A,
//  MSGLEVEL=(1,1),REGION=4M
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DATA TYPE(LOGR) REPORT(YES)
UPDATE LOGSTREAM NAME(SYSPLEX.OPERLOG)
ZAI(YES)
ZAIDATA('OPERLOG')
```

where:

- ZAI(YES) specifies that this log stream data is to be sent to the IBM zAware server.
- ZAIDATA('OPERLOG') is an optional value, indicating that the data being sent is OPERLOG data.

z/OS bulk load client for IBM zAware

To provide analytical data for a monitored client, the IBM zAware server requires a model of normal system behavior to use for comparison. IBM zAware builds these models using SYSLOG data from the clients. You have two options for building a model:

1. Waiting for the server to build a model from data collected over a specific time period
2. Priming the server with prior data (data that existed before?)

We exploited both options. We connected a subset of our systems to IBM zAware and let it build models for them automatically over a period of time. For the majority of our systems however we used the z/OS bulk load client for IBM zAware to prime the server with SYSLOG data that already existed.

The z/OS bulk load client for IBM zAware load modules AIZBLKR and AIZBLKM reside in SYS1.MIGLIB and sample files AIZBLK and AIZBLKE reside in SYS1.SAMPLIB. Here are the customizations we had to make to these files to execute z/OS bulk load client in zPET. (Note that the list does not cover all necessary steps that are required to execute these modules.)

1. We copied the AIZBLKR and AIZBLKM modules to an authorized library that is in LINKLIST concatenation.
2. We copied AIZBLK to a JCL data set and AIZBLKE to a data set that holds various other REXX execs. Note that AIZBLKE contains the sample REXX exec to run the AIZBLKR load module.
3. We modified the AIZBLK JCL as follows:
   - Replaced TSOUSER with our local user ID.
   - Deleted STG_DATACLAS(LOGR4K) from the log stream definition section. In our environment it defaults to a data class with a CI size of 4K.
   - Replaced the DSN in the SYSEXEC DD of the bulk load step with the name of the data set that contains the AIZBLKE REXX exec.
4. We reviewed the AIZBLKE REXX exec and found that we didn't have to make any changes to it.
5. Finally, we updated the AIZBLK JCL BULKLOAD step to list the data sets with the SYSLOG data in them as follows:

```
AIZBLKE OZ2.ZAI.CONTROL ADDSYSLOGDSN +
                LOGWRTR.LOGJX3.G0163V00
        AIZBLKE OZ2.ZAI.CONTROL ADDSYSLOGDSN +
                LOGWRTR.LOGJX3.G0164V00
          …
```

where each LOGWRTR.LOGJX3.* data set contains one day’s worth of SYSLOG data for system JX3.

At this point we were ready to execute z/OS bulk load client for IBM zAware. Because we wanted to test our configuration before sending a whole lot of data to the server, we chose to connect a single system and upload SYSLOG data only for that system.

First we verified the connection from our z/OS LPAR to the IBM zAware server through the D LOGGER,STATUS,ZAI,VERIFY command, and then we submitted the AIZBLK JCL to transfer the SYSLOG data sample? to the server.
When the data was transferred we followed the instructions in the *IBM zAware Guide* to assign the data to the correct sysplex and manually train IBM zAware to generate the system model.

We have also opened another marketing requirement to enhance the way z/OS bulk load client for IBM zAware accepts SYSLOG or OPERLOG data set names as input. Today you have to manually enter all data set names into the AIZBLK JCL. We recommend that you use 90 days worth of messaging data to generate a normal system behavior model. In our environment that is 90 data sets since we offload our SYSLOG data to GDGs on a daily basis.

Some useful commands

Here are some commands that we have been using almost daily since we started exploiting IBM zAware:

1. SETLOGR FORCE,ZAQUIESCE command to quiesce a client’s connection to the IBM zAware server:

```
SETLOGR FORCE,ZAIQUIESCE,LSNAME=SYSPLEX.OPERLOG
   IXG382I ZAI LOGSTREAM CLIENT QUIESCED
   FOR LOGSTREAM SYSPLEX.OPERLOG
   REASON:  SETLOGR COMMAND REQUEST.
   IXG651I SETLOGR FORCE ZAIQUIESCE COMMAND ACCEPTED
   FOR LOGSTREAM=SYSPLEX.OPERLOG
   IXG387I ZAI LOGSTREAM CLIENT CONNECTION ENDED SUMMARY 033
   FOR LOGSTREAM SYSPLEX.OPERLOG
   CONNECTION WAS ESTABLISHED AT: 10/17/2012 23:00:41 LOCAL
   LOG BLOCKS SENT TO SERVER OK: 934861, FAILED: 0
```

2. SETLOGR FORCE,ZAICONNECT to connect a client to the IBM zAware server:

```
SETLOGR FORCE,ZAICONNECT,LSNAME=SYSPLEX.OPERLOG
```

```
   IXG651I SETLOGR FORCE ZAICONNECT COMMAND ACCEPTED
   FOR LOGSTREAM=SYSPLEX.OPERLOG
   IXG386I ZAI LOGSTREAM CLIENT CONNECT ATTEMPT IN PROGRESS
   FOR LOGSTREAM SYSPLEX.OPERLOG
   STATUS:  ATTEMPTING SOCKET CREATE
   IXG386I ZAI LOGSTREAM CLIENT CONNECT ATTEMPT IN PROGRESS
   FOR LOGSTREAM SYSPLEX.OPERLOG
   STATUS:  SOCKET CREATE SUCCESSFUL
   IXG386I ZAI LOGSTREAM CLIENT CONNECT ATTEMPT IN PROGRESS
   FOR LOGSTREAM SYSPLEX.OPERLOG
   STATUS:  ATTEMPTING SOCKET CONNECT
   IXG386I ZAI LOGSTREAM CLIENT CONNECT ATTEMPT IN PROGRESS
   FOR LOGSTREAM SYSPLEX.OPERLOG
   STATUS:  SOCKET CONNECT SUCCESSFUL
   IXG386I ZAI LOGSTREAM CLIENT CONNECT ATTEMPT IN PROGRESS
   FOR LOGSTREAM SYSPLEX.OPERLOG
   STATUS:  INITIATING SOCKET VALIDATION
   IXG386I ZAI LOGSTREAM CLIENT CONNECT ATTEMPT IN PROGRESS
   FOR LOGSTREAM SYSPLEX.OPERLOG
   STATUS:  SOCKET VALIDATION SUCCESSFUL
   IXG380I ZAI LOGSTREAM CLIENT ESTABLISHED
   FOR LOGSTREAM SYSPLEX.OPERLOG
```

3. DISPLAY LOGGER command to dDisplay if the client is currently connected to the IBM zAware server:

```
D LOGGER,C,LSN=SYSPLEX.OPERLOG,D
   IXG601I   17.58.15  LOGGER DISPLAY 102
   CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM TPN
   LOGSTREAM                  STRUCTURE        #CONN  STATUS
   ---------                  ---------        ------ ------
   SYSPLEX.OPERLOG            LOGGER_OPERLOG   000006 IN USE
     DUPLEXING: STRUCTURE, LOCAL BUFFERS
     GROUP: PRODUCTION   ZAI CLIENT: YES - CONNECTED
     ZAIDATA: OPERLOG
   LOG BLOCKS SENT TO SERVER OK: 0002569018, FAILED: 0000000000
   …
   NUMBER OF LOGSTREAMS:  000001
```

Interesting scenarios

Scenario 1: Training IBM zAware following the addition of a new workload

We wanted to see when we added a new workload to our environment if IBM zAware would “flag” the new messages and after running the workload for a few days and "retraining" IBM zAware, whether it would stop “flagging” the messages as anomalies.

We used a program to generate unique message IDs and ran it for 3 days. The program does the following:

- Issues 250 WTOs with message IDs of AIZ9000I – AIZ9249I. Repeats every hour.
- Starting 15 minutes later, issues another 250 WTOs with message ids of AIZ9250I to AIZ9499I. Repeats every hour.
- Starting 15 minutes later, issues another 250 WTOs with message ids of AIZ9500I to AIZ9749I. Repeats every 2 hours.
- Starting 15 minutes later issues another 250 WTOs with message ids of AIZ9750I to AIZ9999I. Repeats every 2 hours.

We started the program on system JL0 on 9/18 at 11:39 am (UTC); it corresponds to the first yellow bar in Figure 2. Remember that the length of the bar indicates a large number of unique messages within the interval and the color yellow indicates that IBM zAware noticed significantly different messages than expected during these intervals.

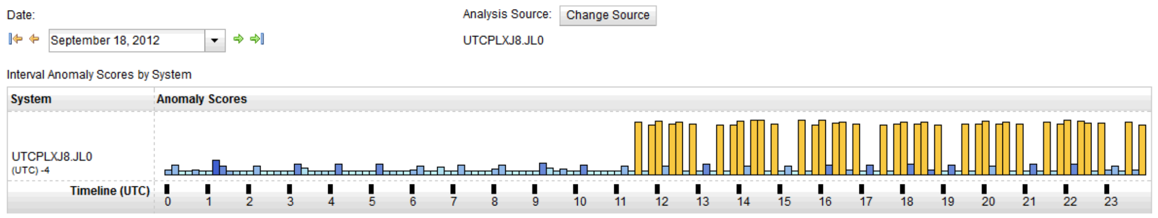


Figure 2: Large number of unique and new messages

On 9/20 10:58 am (UTC), we manually retrained IBM zAware for system JL0. We included 9/18 and 9/19 syslog data. Right after the training, the bars turned blue as seen on Figure 3. This indicated that the messages IBM zAware is seeing are not significantly different from those that it expected. Note that the bar heights are still the same and very high. This makes sense since IBM zAware is still seeing a large number of unique messages within these intervals.

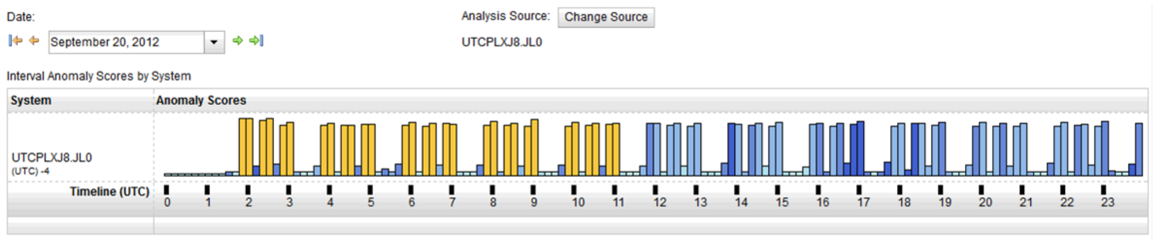


Figure 3: Large number of unique and new messages followed by the same number of unique but now known messages

Scenario 2: Using IBM zAware for problem analysis

On system JB0 we were moving along nicely and seeing “blue” when we started noticing some yellow intervals.

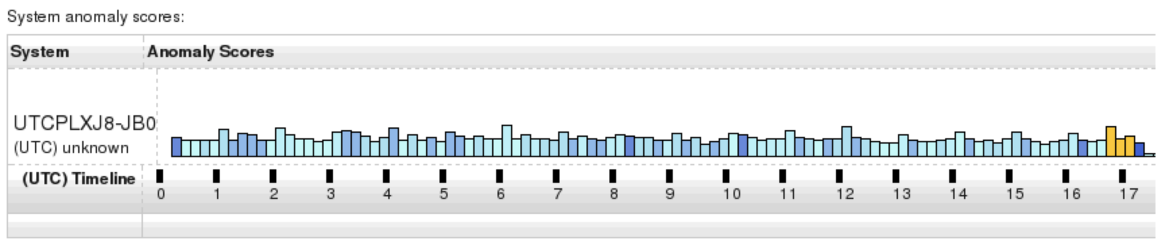
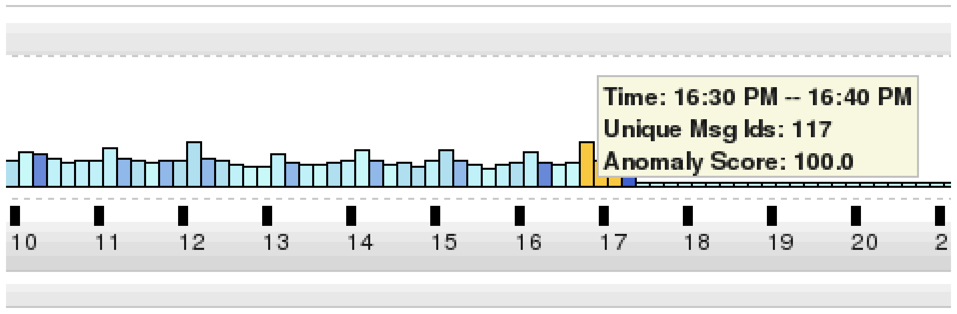


Figure 4: System JB0

Placing the mouse over the first yellow bar, we saw that the anomaly score was 100 and there were 117 unique messages within that 10 minute interval as seen in Figure 5.

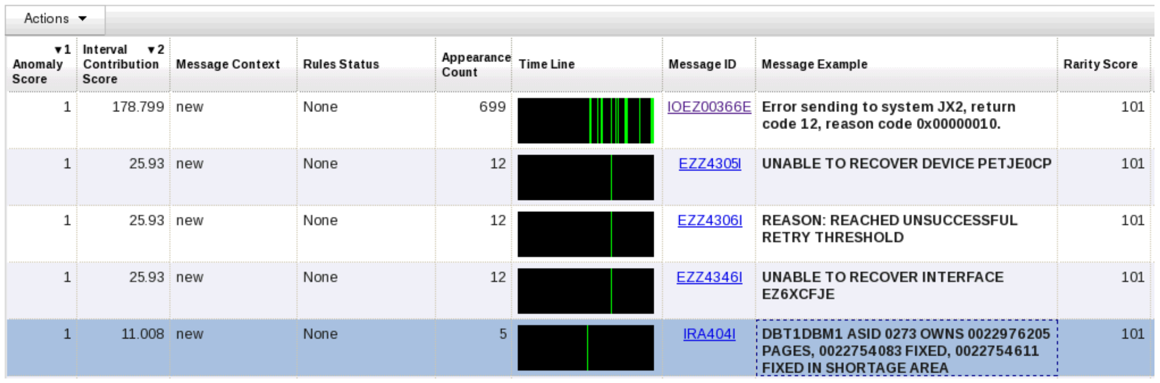


Time: 16:30 PM -- 16:40 PM
Unique Msg Ids: 117
Anomaly Score: 100.0

Figure 5: Interval details pop-up for system JB0

We went into the interval view for the 16:30 to 16:40 interval and noticed that the highest interval contributor was the IOEZ message followed by the EZZ messages displayed in Figure 6 below. However past experiences have taught us that these messages are usually not related to the root cause, and that there are no doubt other underlying problems.

Next we noticed the IRA04I message that indicates that there is a pageable storage shortage and the DB2 address space (DBT1DBM1 in Figure 6) is the largest user of pageable central storage frames in the shortage area.



Actions

▾1 Anomaly Score	Interval ▾2 Contribution Score	Message Context	Rules Status	Appearance Count	Time Line	Message ID	Message Example	Rarity Score
1	178.799	new	None	699		IOEZ00366E	Error sending to system JX2, return code 12, reason code 0x00000010.	101
1	25.93	new	None	12		EZZ4305I	UNABLE TO RECOVER DEVICE PETJE0CP	101
1	25.93	new	None	12		EZZ4306I	REASON: REACHED UNSUCCESSFUL RETRY THRESHOLD	101
1	25.93	new	None	12		EZZ4346I	UNABLE TO RECOVER INTERFACE EZ6XCFJE	101
1	11.008	new	None	5		IRA404I	DBT1DBM1 ASID 0273 OWNS 0022976205 PAGES, 0022754083 FIXED, 0022754611 FIXED IN SHORTAGE AREA	101

Figure 6: Interval view for system JB0 – 16:30 to 16:40 interval

Upon further analysis and discussions with folks that are responsible for DB2 we found that they were running some tests with very large group buffer pools that were page fixed.