UNRAVELING THE E-BUSINESS SUITE OUTPUT POST PROCESSOR (OPP)

ALFREDO ABATE - BRAKE PARTS, INC.

Have you ever received a support call from a business user reporting that their concurrent request completed with a warning and no PDF to download, or that they never received the output of their request to the printer?

As you start looking into the issue, you find that the E- Business Suite (EBS) Output Post Processor (OPP) is either not responding or is down all together!

The OPP runs as a service that is often treated as a black box—neglected and only appreciated when it is not available.

What Exactly is the Function of the EBS OPP?

The OPP performs post-processing operations of a concurrent request's output file. The most common one performed in EBS is converting an XML output into a PDF file by applying a BI Publisher template, which provides an easy-to-read format for a business user.

If you are on versions of EBS 11i – R12.2.x, all of the information that follows applies since the OPP architecture, tuning and troubleshooting guidelines are consistent across those releases.

Architecture

The OPP is not like your other concurrent managers. If you have ever looked at the Administer Concurrent Managers Form, OPP appears to be doing nothing since the running and pending columns will always be blank. Concurrent managers, like the standard manager, have direct connections to the database for every process defined. For example, if a standard manager has 10 processes, you will have that many database connections. If that same manager currently has five requests running, you will have that many active database sessions while the other five will be inactive. OPP, on the other hand, will always have an active database connection for each process defined, but because it is a service that is multi-threaded, it has the ability to process multiple requests simultaneously (up to the maximum number defined).

Behind the scenes, OPP uses the Oracle Database Advanced Queueing (AQ) mechanism for requests requiring post-processing actions. The producers of these queuing messages are initiated by concurrent managers such as the standard manager. The message is sent to the queuing table, and OPP then consumes that message (**Figure 1**). The number of messages OPP can consume simultaneously is determined by a combination of the settings processes, service threads and request threads.

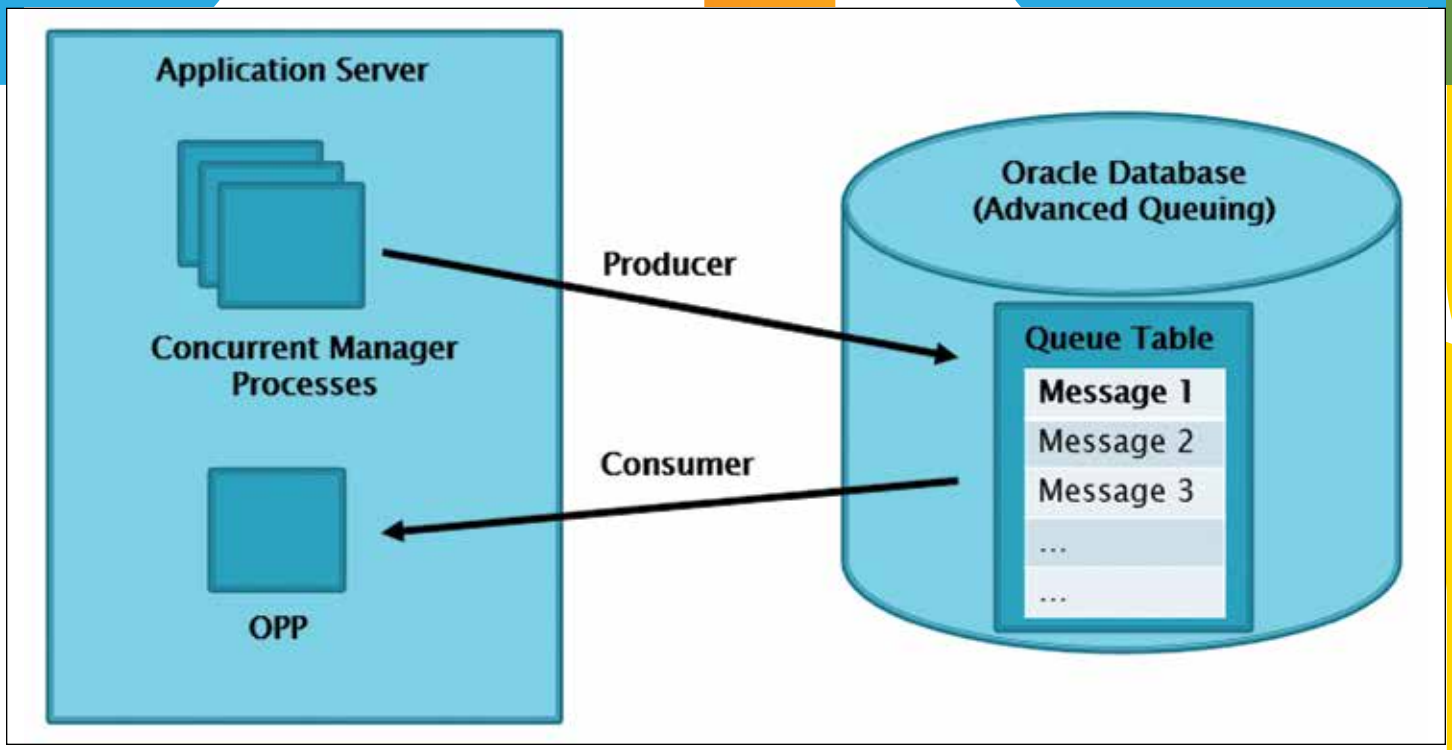

Figure 1:

By default, EBS is defined with one active OPP process with 512 MB of heap memory, two service threads and five request threads for each service thread (**Figure 2**). These default settings will allow you to process **10 requests at a time** (two service threads x five request threads).

Next, there are three System Profile Values related to OPP that every applications DBA should be familiar with. These values are all specified in seconds.

- Concurrent: OPP Initialization Delay (default 30)
  - This profile option specifies the amount of time the manager waits for the OPP service to initialize.
- Concurrent: OPP Process Timeout (default 300)
  - This profile option specifies the amount of time the manager waits for the OPP to actually process the request.
- Concurrent: OPP Response Timeout (default 120)
  - This profile option specifies the amount of time a manager waits for the OPP to respond to its request for post processing.

Additionally, it is important to know that the OPP log file is located in the log directory $APPLCSF/log and will always have a naming convention of FNDOPP<process id>.log for each OPP process that is defined. An example would look like this:

$APPLCSF/log/FNDOPP86282.log

Looking at the contents of the log file shows the process id that was started as well as the initialization parameters for service and process threads.

> [3/6/17 2:08:30 PM] [main] Starting GSF service with concurrent process id = 86282.
>
> [3/6/17 2:08:30 PM] [main] Initialization Parameters: oracle.apps.fnd.cp.opp.OPPServiceThread:2:0:max_threads=5
>
> [3/6/17 2:08:30 PM] [Thread-22] Service thread starting up.
>
> [3/6/17 2:08:30 PM] [Thread-23] Service thread starting up.

Settings Guideline

Now that we have a better understanding of OPP's architecture and its default parameters, let us take a look at what should be adjusted and how. Here are a few things to consider when modifying these settings:

- Increase the request threads first before adding another processor.
- Do not increase the request threads higher than a value of 20.
- It is not recommended to modify the service thread (leave it set to 2).
- All threads under one process share the total Java Virtual Machine (JVM) heap memory to process requests.
- The maximum size of the JVM heap memory is 2048 MB.

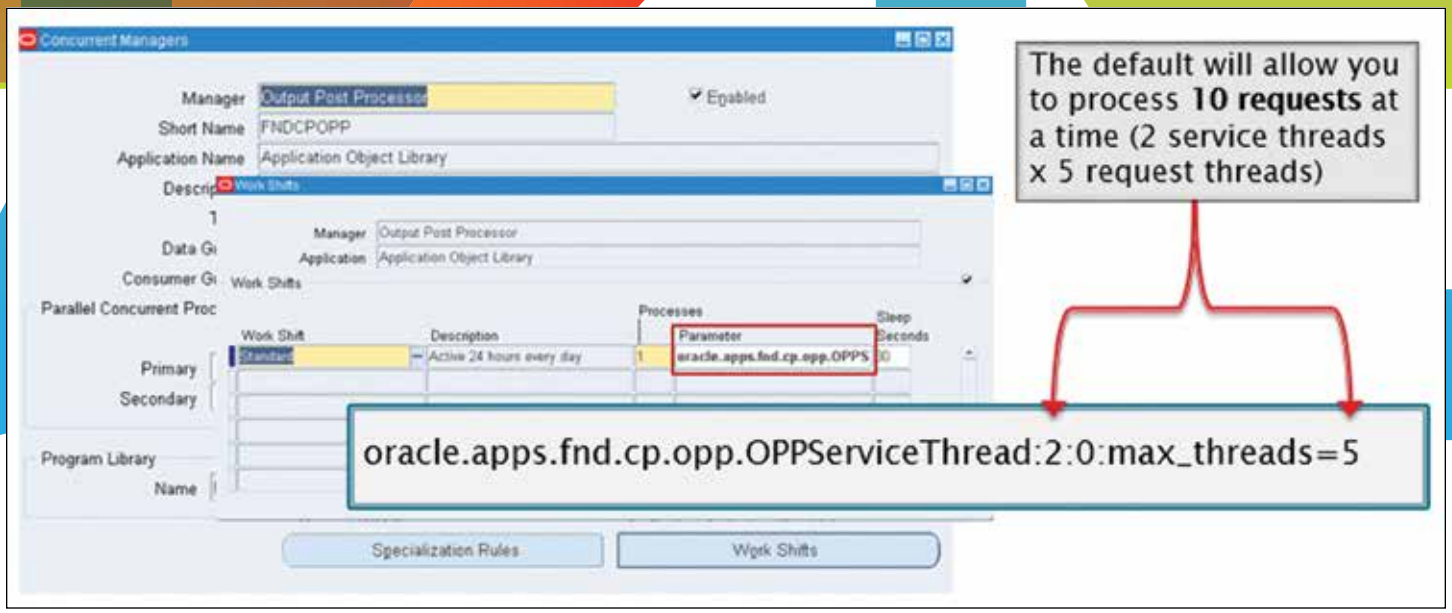

Figure 2:

There is no graphical user interface (GUI) available to adjust OPP's JVM heap memory; use SQL to update it manually.

First, we can execute a SELECT statement to determine the current amount of memory being allocated. (In this case it is 512 MB.)

```
SQL> SELECT service_id, service_handle,
developer_parameters
 FROM fnd_cp_services
 WHERE service_id = (SELECT manager_type
                     FROM fnd_concurrent_queues
                     WHERE concurrent_queue_name =
'FNDCPOPP');
SERVICE_ID SERVICE_ DEVELOPER_
PARAMETERS
----------------------------------------------------------------
    1080 FNDOPP   J:oracle.apps.fnd.cp.gsf.
    GSMServiceController:-mx512m
```

As an example, let us say we want to increase the memory to 1024 MB, so we would execute this UPDATE statement:

```
SQL> UPDATE fnd_cp_services
  SET developer_parameters =
  'J:oracle.apps.fnd.cp.gsf.GSMService
  Controller:-mx1024m'
 WHERE service_id =
(SELECT manager_type
     FROM fnd_concurrent_queues
     WHERE concurrent_queue_name =
     'FNDCPOPP');
```

Note: OPP will need to be restarted after the update is made to reflect the increase in memory.

We can use a SQL script to gather data to determine what post processing requests are taking the most amount time. The FND_CONCURRENT_REQUESTS table has two columns called PP_START_TIME and PP_END_TIME that will be populated with post-processing start and end time.

```
SELECT fcr.request_id,
     fcp.user_concurrent_program_name,
     fcr.phase_code,
     fcr.status_code,
     fcr.output_file_type,
     fcr.pp_start_date,
     fcr.pp_end_date,
     ROUND (1440 * 60 * (fcr.pp_end_date - fcr
     pp_start_date), 2) "Elapsed Time - Sec"
  FROM fnd_concurrent_requests fcr, fnd_
  concurrent_programs_tl fcp
 WHERE  fcr.output_file_type = 'XML'
     AND fcr.PP_START_DATE IS NOT NULL
     AND fcr.concurrent_program_id =
     fcp.concurrent_program_id
     AND fcp.language = 'US'
ORDER BY 8 DESC;
```

REQUEST_ID	USER_CONCURRENT_PROGRAM_NAME	PHASE_C...	STATU...	OUT...	PP_START_DATE	PP_END_DATE	Elapsed Time - Sec
14672397		C	C	XML	3/2/2017 12:05:38 PM	3/2/2017 12:06:38 PM	59
14672252		C	C	XML	3/2/2017 12:05:39 PM	3/2/2017 12:06:38 PM	58
14688391		C	C	XML	3/3/2017 9:29:21 AM	3/3/2017 9:29:57 AM	36
14684610		C	C	XML	3/3/2017 4:33:25 AM	3/3/2017 4:33:55 AM	29
14679213		C	C	XML	3/2/2017 8:33:24 PM	3/2/2017 8:33:53 PM	28
14664358		C	C	XML	3/2/2017 2:34:54 AM	3/2/2017 2:35:23 AM	28
14668027		C	C	XML	3/2/2017 7:48:13 AM	3/2/2017 7:48:39 AM	26
14669705		C	C	XML	3/2/2017 8:52:45 AM	3/2/2017 8:53:10 AM	25
14644352		C	C	XML	3/2/2017 6:00:07 AM	3/2/2017 6:00:32 AM	25

Figure 3:

Once you have retrieved the post-processing elapsed time data (Figure 3), you can use it to make a couple of determinations:

- Use your highest elapsed time to estimate the setting for the system profile value of Concurrent: OPP Process Timeout. If your longest running report is, say, 26 minutes, then you may want to set this to 1800 (30 minutes).
- Are any of these using poorly written XML Publisher templates that can be resolved by a patch if it is Oracle seeded or re-written by a developer if it is custom?
- Are proper parameters being entered for these concurrent requests? Missing a parameter or using an incorrect parameter could cause the output file to be very large, thus increasing the length of time OPP takes to process. I have seen this happen where output files are over 1 GB in size and eventually cause OPP to exhaust its JVM heap memory.

Finally, if you have set OPP's memory to the maximum of 2048 MB and increased the request threads to 20 but are finding that OPP still cannot keep up with your requests, then it is time to add another OPP process. Keep in mind that every time you increase the OPP processes parameter, you'll need to make sure that you have enough free memory on your server to accommodate for it.

Troubleshooting

If an OPP process is consuming a large amount of CPU on your server, I recommend reviewing My Oracle Support Document ID 2028617.1, which outlines the steps to determine the offending report template that is causing the issue. Once the report template is determined, you will have two options:

- **Seeded Oracle Template** – You can search My Oracle Support to see if a document already exists with a solution, which in most cases involves applying a patch. Otherwise, open up a service request (SR) with Oracle Support.
- **Custom Template** – You will need to determine if the report can be written more efficiently. Tips to do this are available on My Oracle Support Document ID 1410160.1.

Additionally, there are three common OPP errors that are typically found in concurrent request logs that complete with a warning. The solutions to these errors revolve mostly around modifying the System Profile Values we reviewed earlier. My Oracle Support document 352518.1 should be reviewed to learn more about the suggested solutions.

Known Issues

There are two well-known issues with OPP in EBS 11i – 12.1. There are patches available that should be applied to resolve these issues. The fixes have already been included in EBS 12.2.x. From My Oracle Support document 1399454.1, these patches are described as:

- When the OPP service would experience a serious exception (out of memory, for example), it would enter a state where it would no longer process requests but would not shut down either.

  This patch will enable the service to shut down gracefully in this situation.

  - For 12.1 - Patch 16000609 - OPP IS UNSTABLE / UNRELIABLE
  - For 12.0 - Patch 16634935 - OPP IS UNSTABLE / UNRELIABLE
  - For 11i - Patch 9724344 - OPP IS UNSTABLE / UNRELIABLE
- The number of OPP configured and started don't match number of running OPP. The following patch will resolve target/active mismatch:
  - For 12.1 - Patch 15981176
  - For 12.0.6 - Patch 15981173:R12.FND.A
  - For 11i - Patch 15900099

Monitoring

Having the ability to monitor OPP allows the DBA to be notified of problems before the user is. Simple shell scripts can be written to monitor areas such as:

- OPP service up/down.
- OPP log scraping for common errors such as JVM heap memory errors.
- Concurrent request log scraping for OPP errors such as response or process timeout.

If you own the Oracle Application Management (AMP) pack for Oracle Enterprise Manager (OEM), you can take advantage of the already built-in metrics that will monitor the following:

- Output Post Processor Hang Error.
- No. of No Response and Timeout Error Requests.
- No Response Error Requests.
- Response.
- Timeout Error Requests.

Summary

With this basic understanding of the OPP architecture, including tuning, troubleshooting and monitoring, you can have a well running system.

No longer should the OPP be treated as a black box; it can be properly cared for to ensure ultimate performance in your system and to provide the highest reliability to your business users! ◆

References

12.2 Setup Guide - E22953-10

12.2 Concepts Guide - E22949-09

Enterprise Manager Oracle Application Management Pack for Oracle E-Business Suite Metric Reference Manual - E24628_01

Tuning Output Post Processor (OPP) to Improve Performance (Doc ID 1399454.1)

Concurrent Processing - Using the Concurrent Request Output Post Processor (OPP) in Oracle Applications (Doc ID 291792.1)

Concurrent Requests Fail Due To Output Post Processing (OPP) Timeout (Doc ID 352518.1)

Alfredo Abate is the Senior Oracle Systems Architect at Brake Parts, Inc. where he enjoys working with a full suite of Oracle technologies. He is an Oracle Ace Associate, a regular presenter at various conferences and an active participant with the North Central OAUG and the OAUG E-Business Suite Applications Technology Stack Special Interest Group (SIG).