

Concurrent Manager Queue Overlap Analysis

Andy Rivenes
AppsDBA Consulting

1. Abstract

Managing the Oracle E-Business Suite's batch job workload can be a complicated process. The E-Business Suite uses batch jobs, known as "concurrent programs", and a submission system called the "Concurrent Manager". The concurrent manager supports multiple batch queues, workshifts, prioritization and specialization rules. The E-Business Suite uses concurrent programs for more than just running traditional batch jobs. In fact other critical parts of the E-Business Suite also run as concurrent programs such as workflow and transaction managers. All of these "batch" type processes can easily overwhelm a system's capacity or can severely impact interactive response time if they are left to run unconstrained.

This paper will explore how to map Oracle Applications concurrent manager queue utilization for a given time period. With this information it is possible to understand how to properly manage concurrent manager workload to meet service level requirements. Traditional techniques have simply used requested and actual start dates and actual completion dates to determine what was running in a queue during an interval period. The problem with this type of analysis is that it is prone to error (e.g. long running processes that span interval periods are missed) and there's no way to tell how many processes were running at any given point in time. A better method is to perform a technique that I am calling "queue overlap analysis" to determine exactly what was running and when. This enables the Oracle Applications professional to make a much more accurate assessment of concurrent manager queue utilization.

2. Introduction

Oracle Applications concurrent manager queue utilization analysis can be a complex process. Each concurrent manager queue is defined with a specific number of processes, can be assigned to work shifts with specific process allocations, and can have specialization rules that apply to each of those queues. Jobs can have priorities assigned to them and can be part of request sets.

As part of managing their concurrent processing workload most sites will create reports that detail how many jobs ran in each queue, how long they ran and even the delays that might have occurred before the jobs ran. The following shows a summary of the type of information that is often produced:

Concurrent Program Profile for 18-JAN-06

Queue Name	Total Jobs	Total Time (Min)	Min Time (Min)	Avg Time (Min)	Max Time (Min)	Avg Delay (Min)	Max Delay (Min)
INVMGR	1,367	105.37	0.02	0.08	2.52	.696	2.867
MRPMGR	1	0.12	0.12	0.12	0.12	1.033	1.033
COSTROLLUP	141	274.29	1.37	1.95	4.53	.529	2.117
FASTMGR	5,648	1355.47	0.00	0.24	16.77	1.059	15.817
CUSTMGR	259	300.58	0.00	1.16	9.30	1.366	13.817
EXTERNALAPPS	31	54.49	0.10	1.76	5.05	.091	.417
PLANMGR	2,082	44.24	0.00	0.02	0.15	1.427	8.150
PLANSCHD	833	356.43	0.00	0.43	14.97	1.418	14.017
COLLECT_IMPORT	128	781.09	0.03	6.10	186.37	120.191	348.317
STANDARD	4,240	7327.75	0.00	1.73	465.32	4.305	123.415
STDCST	141	102.03	0.05	0.72	5.57	.956	8.083

Many times this type of information is broken down into a much shorter interval period to give better insights into what jobs ran during peak processing intervals:

Concurrent Program Profile for 17-JAN-06

Date	Hr	Queue Name	Total Jobs	Total Time (Min)	Min Time (Min)	Avg Time (Min)	Max Time (Min)	Avg Delay (Min)	Max Delay (Min)
01/17/06	10	INVMGR	86	8.43	0.02	0.10	2.17	.82	3.65
		COSTROLLUP	20	54.66	1.58	2.73	4.72	.35	.78
		FASTMGR	564	235.55	0.00	0.42	9.63	4.16	15.45
		CUSTMGR	23	75.25	0.00	3.27	8.10	4.18	20.25
		EXTERNALAPPS	4	5.42	0.30	1.36	2.45	.05	.11
		PLANSCHD	43	26.39	0.00	0.61	4.17	.32	.81
		STANDARD	253	360.99	0.00	1.43	43.50	2.51	269.48
		STDCST	16	18.95	0.15	1.18	4.88	3.37	8.28

Many other types of reports can be produced as well. Most of these reports are based on the view FND_CONCURRENT_REQUESTS and make use of the columns ACTUAL_START_DATE , ACTUAL_COMPLETION_DATE and REQUESTED_START_DATE.

When performing queue utilization analysis however, most of these reports suffer from rather large inaccuracies for other than broad interval periods, and none provide the detail necessary to actually observe individual queue utilization. This paper will explore how to more accurately measure Oracle Applications concurrent manager queue utilization for a given time period, and will introduce a technique called “queue overlap analysis” to determine exactly what was running in a given queue and when. This technique enables the Oracle Applications professional to make a much more accurate assessment of concurrent manager queue utilization.

3. Concurrent Manager Queue Analysis

As stated in the introduction concurrent manager queue analysis traditionally deals with reporting on the number of jobs that have run, the total run time, and average, min and max run times. Very often delays in start times are also reported. More often than not though, these reports will over state run times, especially when reporting on shorter intervals. This occurs because start and ending interval times are not considered in the

calculation of job end times and total run times. This means that when a job overruns an interval its run time is only attributed to the interval that it started in. This poses a serious problem when trying to control peak workloads by restricting concurrent manager usage.

Accurate interval attribution is also key to being able to perform queue overlap analysis. Included in Appendix A is the SQL code that will accurately calculate the adjusted start and end times of all jobs during an interval and will calculate the specific minutes each job runs during the interval.

4. Queue Overlap Analysis

Queue overlap analysis is the ability to show actual request run time for a concurrent manager queue within a given hourly period. This allows the Oracle Applications professional to review not only job run time by minute but also which minute(s) and how many jobs ran simultaneously. This allows for much better prediction of maximum process numbers for each queue. This report also allows a better understanding of jobs that span interval periods by showing us only the time the job ran during each interval period.

Below are two examples of queue overlaps. A summary is shown first and then the overlap sections list the job name. One or more “+” signs under the time scale shows when the job was active. The time scale is in minutes from 0 to 59 representing an hourly interval.

4.1 Concurrent Manager Queue Overlap - Example 1

4.1.1 Concurrent Manager Summary – Date: 01/17/06, Hour: 07

Date	Hr	Queue Name	Total Jobs	Total Time (Min)	Min Time (Min)	Avg Time (Min)	Max Time (Min)	Avg Delay (Min)	Max Delay (Min)
01/17/06	07	INVMGR	10	0.37	0.02	0.04	0.10	.62	1.05
		FASTMGR	197	73.75	0.00	0.37	13.57	.41	2.90
		CUSTMGR	6	0.76	0.00	0.13	0.55	1.14	1.88
		EXTERNALAPPS	2	3.22	1.47	1.61	1.75	.05	.08
		PLANSCHD	1	0.02	0.02	0.02	0.02	.95	.95
		COLLECT_IMPORT	3	166.74	31.30	55.58	67.72	110.31	120.75
		STANDARD	167	206.58	0.00	1.24	22.47	5.83	275.66

Concurrent Manager Queue Overlap – FASTMGR Queue

ENTER the interval date (format: MM/DD/YYYY) > 01/17/2006
 ENTER the interval hour (format: HH24) > 20
 ENTER the CM queue (format: INVMGR, leave blank for all) > FASTMGR

Processing requests for concurrent queue: FASTMGR

Time		1	2	3	4	5
Min:		01234567890123456789012345678901234567890123456789				
Material cost transaction work	+					
Pick Selection List Generation	+					
Pick Slip Report	+					
Pick Selection List Generation		+++++				
Pick Selection List Generation	+					
Pick Selection List Generation	+					
Pick Slip Report	+					
Pick Slip Report	+					
Pick Selection List Generation	+					
Pick Slip Report	+					
Pick Selection List Generation	+					
Pick Slip Report	+					
Material cost transaction work	+					
Pick Selection List Generation	+					
Pick Slip Report	+					
Interface Trip Stop			+			
Packing Slip Report			+			
Material cost transaction work			+			
Order Import				+		
Order Import				+		
Pick Slip Report				+++++		
Material cost transaction work				+		
Material cost transaction work					+	
Material cost transaction work						+
Material cost transaction work						
Re-process Plan Schedule						+
Material cost transaction work						
Interface Trip Stop						+
Packing Slip Report						+
Material cost transaction work						+
Receiving Transaction Processo						+

Note that in this example there is a “Pick Slip Report” that runs all the way up to the last minute. In this case this job extended into the next interval. Again traditional reporting would have attributed the total run time of this job to this interval only, and would not have attributed any time to the next interval at all. It’s also interesting to note that during the 9th minute there were an interval maximum of 5 processes running. When analyzing concurrent manager queues in environments with critical job run requirements, and where start delays can be detrimental to overall processing, this technique is the only way to understand how to minimize start delays.

The traditional summary shows that there were a total of 154.41 minutes of run time. The overlap analysis tells us that there were really only a maximum of 81 minutes of runtime. This is quite a large margin for error and for many sites leads to disastrous decisions when trying to accommodate runaway batch processing.

5. Conclusion

Hopefully this paper has shown a new and useful technique in analyzing concurrent manager queue utilization. The ability to see what was running and when, and how many processes were running at any given time, can be very helpful when trying to determine how many processes to allocate to individual concurrent manager queues. This is also very useful when trying to insure that critical processes will always be able to run or when trying to restrict jobs so as to not overwhelm server resources.

6. About The Author

Andy Rivenes is an Oracle DBA who has been working with Oracle products since 1992. Mr. Rivenes has specialized in Oracle Applications systems as well as custom Oracle environments. He has worked for Oracle Corporation and currently works for Lawrence Livermore National Laboratory as an Oracle DBA. Mr. Rivenes presents at various Oracle related conferences and chaired the OAUG Database SIG. Mr. Rivenes maintains the AppsDBA.com web site and currently focuses on topics related to Oracle system management. Mr. Rivenes can be reached at andy@appsdba.com.

7. Appendix A – fnd_cp_ovlp_bymgr.sql

```
-- FILE:    fnd_cp_ovlp_bymgr.sql
--
-- AUTHOR:  Andy Rivenes, arivenes@appsdba.com, www.appsdba.com
--         Copyright (C) 2006 AppsDBA Consulting
--
-- DATE:    01/17/2006
--
-- DESCRIPTION:
--         Query to show concurrent program overlap by concurrent manager
--         for a given hour.
--
-- MODIFICATIONS:
--         09/18/2006, A. Rivenes, General cleanup and added more comments.
--         10/16/2006, A. Rivenes, Updated date formats to catch all overlaps
--         and simplified interval calculations with Jerry Ireland's suggestions.
--         10/18/2006, A. Rivenes, Further simplified with more of Jerry's
--         suggestions.
--
SET LINESIZE 200;
SET TRIMSPOOL ON;
SET PAGES 9999;
SET HEAD ON;
SET SERVEROUTPUT ON;
--
SET HEAD OFF;
ACCEPT var_date PROMPT 'ENTER the interval date (format: MM/DD/YYYY) > ' ;
ACCEPT var_hr PROMPT 'ENTER the interval hour (format: HH24) > ' ;
ACCEPT var_mgr DEFAULT '%' PROMPT 'ENTER the CM queue (format: INVMGR, leave blank for all) > ' ;
PROMPT ;
SET HEAD ON;
--
SPOOL fnd_ovlp_bymgr.txt;
--
DECLARE
-- Cursor 1:
```

```

--
-- Determine queues with jobs that were active queues during any part
-- of the interval period
--
CURSOR cm_queue_cur(var_intvl VARCHAR, var_queue VARCHAR) IS
SELECT
  DISTINCT q.concurrent_queue_name CMQUEUE
FROM
  applsys.fnd_concurrent_requests r,
  applsys.fnd_concurrent_processes p,
  applsys.fnd_concurrent_queues q
WHERE
  ( r.ACTUAL_START_DATE <= TO_DATE(var_intvl||':59:59','MM/DD/YYYY HH24:MI:SS')
    AND r.ACTUAL_COMPLETION_DATE >= TO_DATE(var_intvl||':00:00','MM/DD/YYYY HH24:MI:SS') )
  AND q.concurrent_queue_name LIKE var_queue
  AND R.controlling_manager = P.concurrent_process_id
  AND p.concurrent_queue_id = q.concurrent_queue_id
  AND p.queue_application_id = q.application_id
ORDER BY
  q.concurrent_queue_name;
--
--
-- Cursor 2:
--
-- Get all requests by queue_name, program and run time
-- Use input parameters for date and queue
--
--   STIM - Start time
--   SMIN - Start minute
--   RMIN - Run minutes (interval duration)
--
CURSOR cm_runtime_cur(var_intvl VARCHAR, var_queue VARCHAR) IS
SELECT
  q.concurrent_queue_name CMQUEUE,
  DECODE(cptl.user_concurrent_program_name,
    'Report Set', r.description,
    cptl.user_concurrent_program_name) CMPROG,
  r.ACTUAL_START_DATE STIM,
  TO_CHAR(GREATEST(r.ACTUAL_START_DATE,TO_DATE(var_intvl||':00:00','MM/DD/YYYY HH24:MI:SS')), 'MI')
SMIN,
  CEIL((LEAST(r.ACTUAL_COMPLETION_DATE,TO_DATE(var_intvl||':59:59','MM/DD/YYYY HH24:MI:SS'))
    - GREATEST(r.ACTUAL_START_DATE,TO_DATE(var_intvl||':00:00','MM/DD/YYYY HH24:MI:SS')))*60*24) RMIN
FROM
  applsys.fnd_concurrent_requests r,
  applsys.fnd_concurrent_processes p,
  applsys.fnd_concurrent_queues q,
  applsys.fnd_concurrent_programs_tl cptl
WHERE
  ( r.ACTUAL_START_DATE <= TO_DATE(var_intvl||':59:59','MM/DD/YYYY HH24:MI:SS')
    AND r.ACTUAL_COMPLETION_DATE >= TO_DATE(var_intvl||':00:00','MM/DD/YYYY HH24:MI:SS') )
  AND q.concurrent_queue_name = var_queue
  AND R.controlling_manager = P.concurrent_process_id
  AND p.concurrent_queue_id = q.concurrent_queue_id
  AND p.queue_application_id = q.application_id
  AND r.program_application_id = cptl.application_id
  AND r.concurrent_program_id = cptl.concurrent_program_id
ORDER BY
  STIM,
  RMIN;
--
var_runtime VARCHAR2(61);
var_filler CHAR(1) := '+';
ctr INTEGER;
--
BEGIN
--
  DBMS_OUTPUT.ENABLE(1000000);
--
  -- Loop for each active queue during requested interval (e.g. hour)
  --
  FOR cm_queue_rec IN cm_queue_cur('&var_date' || ' ' || '&var_hr', '&var_mgr') LOOP

```

```

--
DBMS_OUTPUT.PUT_LINE('Processing requests for concurrent queue: '||cm_queue_rec.CMQQUEUE);
DBMS_OUTPUT.PUT_LINE(CHR(13));
DBMS_OUTPUT.PUT_LINE('Time                               1           2           3           4
5');
DBMS_OUTPUT.PUT_LINE('Min:
012345678901234567890123456789012345678901234567890123456789');
--
-- Create overlap view for each queue
--
FOR cm_runtime_rec IN cm_runtime_cur('&&var_date' || ' ' || '&&var_hr', cm_queue_rec.CMQQUEUE) LOOP
  var_runtime := '';
  --
  -- Blank pad until start minute
  --
  FOR ctr IN 0..TO_NUMBER(cm_runtime_rec.SMIN) LOOP
    var_runtime := var_runtime || ' ';
  END LOOP;
  --
  -- Mark run time
  --
  FOR ctr IN 1..TO_NUMBER(cm_runtime_rec.RMIN) LOOP
    var_runtime := var_runtime || var_filler;
  END LOOP;
  --
  -- Output record
  --
  DBMS_OUTPUT.PUT_LINE(RPAD(cm_runtime_rec.CMPROG,30,' ') || ' ' || var_runtime);
END LOOP;
--
DBMS_OUTPUT.PUT_LINE(CHR(13));
END LOOP;
END;
/
--
SPOOL off;

```

8. References

- [1] A. S. Rivenes, "SYSMON, Oracle Workload Management Utility", AppsDBA Consulting, www.appsdba.com.
- [2] Oracle Applications System Administrator's Guide, Volume 1, Release 11i, Oracle Corporation, Part No. A96155-03
- [3] eTRM Technical Reference, FND – Application Object Library (table descriptions), Oracle Corporation (Metalink)