

Concurrent Manager Queue Overlap Analysis

Andy Rivenes (andy@appsdba.com)
AppsDBA Consulting

Oracle Applications Users Group (OAUG)

- THE users group for all Oracle E-Business Suite, PeopleSoft, Siebel, Oracle Retail and Portal Software customers
- Networking opportunities with over 100,000 members worldwide
- Over 20,000 available white papers in the online OAUG Conference Paper Database
- FREE online training every Tuesday for OAUG members
- Introductory FREE Web membership – sign up during Oracle OpenWorld in the Oracle Users Group Pavilion
- Learn more about the OAUG in the Oracle Users Group Pavilion, Moscone West, Lobby Level 2



Global Users. Global Solutions.

My Background

- Started as an IBM systems programmer in 1984.
- Became an Oracle DBA in 1992.
- Worked for Oracle Corporation for a couple of years.
- Founded AppsDBA Consulting in 1998.
- Currently employed at Lawrence Livermore National Laboratory.

Introduction

- Concurrent manager queue utilization is probably more complicated than you think.
- We will explore how to accurately measure queue utilization for a given time period.
- We will also add a new technique to help pinpoint what was running in a queue and how long it really took.

Traditional Queue Utilization

- Most sites and scripts that I have seen use the table FND_CONCURRENT_REQUESTS and the columns ACTUAL_START_DATE and ACTUAL_COMPLETION_DATE.
- Some sites also report start delays. This is usually calculated based on the difference between REQUESTED_START_DATE and ACTUAL_START_DATE.

Traditional Queue Utilization (continued)

- Most of the results will be broken down by queue name or job name.
- Time intervals typically span an hour, a day or a work shift.
- Many times total job count and total run time will also be reported.

Let's Not Forget

- Queues
 - Workshifts
 - Processes per queue
 - Specialization Rules
- Jobs
 - Priorities

Traditional Report Daily Queue Summary

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
MRFMGR	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
CUSTOMGR	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

Traditional Report Hourly Queue Summary

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

Traditional Report Daily Job Summary

program short name	program	job_count	Total Time Minutes
QLTTRAWB	Collection Import Worker	110	726
FNDWFBG	Workflow background process for deferred and timeout ac	696	553
PROC_SCHED_BY_ORDER	Process Schedule Records By Order	1393	499
CSTRBNR	Cost Rollup - No Report	263	489
CUST_OMSC	CUSTOM: Ship Orders	30	416
ALECTC	Check Event Alert	6380	393
CMCICU	Update Standard Cost from SRS	269	385
SALES_ORDER_INTF	Sales Order Interface	187	316
RMACLOSELINE	RMA Close Line	48	305
SSOCP	Show Sales Order	288	228
WICMLP	WIP Mass Load	937	227
JOBRESPR	Job Response	885	195
MRCSDW	Snapshot Delete Worker	24	189

Traditional Report Job Details

Queue Name	Program Name	Total Jobs	Total Time (Min)	Min Time (Min)	Avg Time (Min)	Max Time (Min)	Avg Delay (Min)	Max Delay (Min)
INVWGR	Overhead Cost Worker	113	5.48	0.02	0.05	0.33	.826	2.217
	Resource Cost Worker	9	1.16	0.08	0.13	0.20	1.163	1.767
Queue Totals		122						
MRPMGR	Planning Manager	1	0.15	0.15	0.15	0.15	1.050	1.050
Queue Totals		1						
COSTROLLUP	Cost Rollup - No Report	223	409.54	1.43	1.84	5.70	.650	3.867
Queue Totals		223						
FASTGR	Interface Trip Stop	405	70.47	0.03	0.17	1.48	.610	5.917
	Journal Import	27	3.18	0.03	0.12	0.57	.567	1.233
	Material cost transaction worker	319	33.44	0.02	0.10	0.47	.661	9.317
	Packing Slip Report	116	45.03	0.23	0.39	0.65	2.090	10.693
	Re-process Scheduling	10	6.83	0.02	0.68	5.17	.385	.883
	Update Order Schedule	18	20.86	0.00	1.16	5.32	1.897	4.250
	Concurrent Program	19	2.98	0.05	0.16	0.33	.473	.950
	Response	139	20.50	0.03	0.15	1.42	.569	2.967
	RMA Receipt Label Report	468	267.96	0.08	0.57	8.70	2.874	10.833
	Order Import	286	58.65	0.00	0.21	20.50	.955	22.500
	Pick Selection List Generation	423	18.54	0.00	0.04	0.37	.555	7.833
	Pick Slip Report	141	57.50	0.07	0.41	1.43	.209	10.767
	Receiving Transaction Processor	170	221.51	0.02	1.30	71.53	.378	3.033
	WIP Mass Load	984	287.63	0.07	0.29	8.72	1.758	10.833
Queue Totals		3,525						

10/25/2006

Copyright © 2006, AppsDBA Consulting, All Rights Reserved.

Slide 11

Traditional Report Job Chronology

Queue Name	Program Name	Total Jobs	Total Time (Min)	Min Time (Min)	Avg Time (Min)	Max Time (Min)	Avg Delay (Min)	Max Delay (Min)
01/19/06 00	INVWGR	Inventory transaction worker	3	1.78	0.20	0.59	1.23	.083
01/19/06 00		WIP Move Transaction Manager	30	1.12	0.02	0.04	0.08	1.333
01/19/06 00		Process transaction interface	11	0.68	0.03	0.06	0.13	.517
01/19/06 01		Inventory transaction worker	1	1.58	1.58	1.58	1.58	.100
01/19/06 01		WIP Move Transaction Manager	29	0.86	0.02	0.03	0.05	3.867
01/19/06 01		Process transaction interface	12	0.59	0.03	0.05	0.08	.800
01/19/06 02		Inventory transaction worker	3	1.35	0.05	0.45	1.20	.067
01/19/06 02		WIP Move Transaction Manager	29	0.97	0.02	0.03	0.05	2.717
01/19/06 02		Process transaction interface	11	0.63	0.03	0.06	0.15	.833
01/19/06 03		WIP Move Transaction Manager	29	0.89	0.02	0.03	0.05	2.350
01/19/06 03		Inventory transaction worker	1	0.73	0.73	0.73	0.73	.083
01/19/06 03		Process transaction interface	11	0.58	0.03	0.05	0.08	1.417

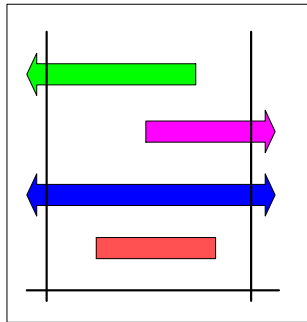
10/25/2006

Copyright © 2006, AppsDBA Consulting, All Rights Reserved.

Slide 12

The Problem With Intervals

- Traditional reporting lumps all run time into the interval the job started in.



Measurement Error

Concurrent Program Profile for 20-JAN-06

Queue	Name	Total Jobs	Total Time (Min)	Min Time (Min)	Avg Time (Min)	Max Time (Min)	Avg Delay (Min)	Max Delay (Min)
01/20/06 09	INVMGR	81	7.05	0.03	0.09	0.50	.741	1.817
	COSTROLLUP	16	35.59	1.48	2.22	3.53	.328	.850
	FASTMGR	428	87.48	0.00	0.20	5.92	1.584	6.500
	CUSTOMGR	25	104.82	0.00	4.19	8.73	1.008	5.867
	POWER	9	12.48	0.35	1.39	2.02	.096	.150
	WEBPLAN	68	1.69	0.00	0.02	0.13	.114	.317
	WEBPLANSCHD	19	22.35	0.02	1.18	7.18	.339	.867
	COLLECT_IMPORT	2	462.58	183.93	231.29	278.65	236.067	236.067
	STANDARD	318	433.10	0.00	1.36	54.63	.639	13.850
	STDCST	22	36.60	0.13	1.66	3.73	3.013	12.683

So, Is This The Information We Really Need?

- What happens when a job runs longer than the interval being measured?
- How do you pinpoint which jobs or queues are contributing to system resource utilization?
- How do you reduce start delay times?

Is There A Better Way?

- There are two basic reasons for wanting to track concurrent manager usage:
 - How long did jobs run and could they run in less time.
 - How many jobs ran and could their workload be spread out or reduced to better manage system resources.

Running Jobs In Less Time

- To run a job in less time it must either be made more efficient or it must be given more resources (i.e. either reduce the work or do the work faster).

Running Jobs In Less Time (continued)

- Making a job more efficient will typically involve response time optimization and Method R (see hotsos.com).
- Giving a job more resources will typically require either reducing competition for existing resources or making those resources faster (e.g. an upgrade).

Better Workload Management

- This was the motivation behind the creation of Queue Overlap Analysis.
- In order to better manage workload you have to understand what the workload is.
- As another extension to Hotsos Method R: optimize the activity that will have the greatest net payoff to the business.

Queue Overlap Analysis

- Managing Concurrent Processing
 - To provide a process to systematically throttle unconstrained concurrent manager workload while still allowing business critical processes to run and protecting interactive response time.
- Queue Overlap Analysis
 - A tool to help manage concurrent processing.

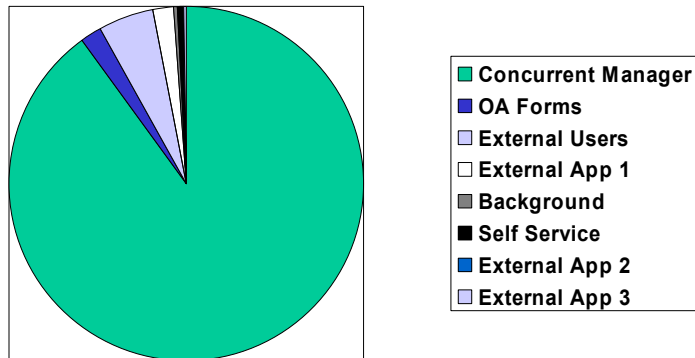
A Real Life Example

- A client site was having trouble meeting their workload requirements.
- Batch jobs appeared to be overwhelming the system.
- During peak processing interactive users experienced unacceptable response time.

More Background

- The client had added too many workers.
- During peak times the system was swamped with too many batch jobs.
- Since processes were taking too long to run, they added even more workers.
- This just made their problems worse.

Original Situation



10/25/2006

Copyright © 2006, AppsDBA Consulting, All Rights Reserved.

Slide 23

Our Goals

- We were having trouble convincing the client that they needed to restructure their batch workload.
- We set two goals:
 - Reduce critical batch job start delays.
 - Protect interactive response time during heavy batch demand.

10/25/2006

Copyright © 2006, AppsDBA Consulting, All Rights Reserved.

Slide 24

Solutions

- We defined workshifts that matched business processing
- We re-defined concurrent manager queues and classified jobs based on priority to the business
- But - We didn't have enough information to properly size high priority queues in order to insure adequate throughput.

Solutions (continued)

- We needed to know how many and what type of jobs were running at any given time.
- We created three classes of jobs:
 - Business critical
 - Important
 - Run when resources are available
- We also had to protect interactive response time.

Identifying Jobs

- These were mostly workflow jobs and transaction manager jobs.
- Most of these jobs were of very short duration and part of a sequence of processes that were dependent on the previous step.
- There were also plenty of long running jobs.

Queue Overlaps

- So, we used queue overlap analysis to show us how many jobs were running within a given interval and how many were running at the same time.
- This allowed us to understand how many processes a queue needed during a given interval and still meet the resource demand with an acceptable delay.

Results

- Reconfigured the concurrent manager queues based on the information provided by Queue Overlap Analysis:
 - During peak interactive usage overall concurrent manager queue processes were lowered to insure that they didn't dominate system utilization.
 - To insure that critical batch jobs could always run we further restricted low priority batch jobs by segregating them to designated queues and lowering their process numbers even further.

The Code

- A SQL script with an anonymous PL/SQL block to list each job that ran during the interval.
- The job uses a one hour interval with a one minute granularity.
- Markers are output for each specific minute that the job was active.

Determine Active Queues

```
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;
```

10/25/2006

Copyright © 2006, ApsDBA Consulting, All Rights Reserved.

Slide 31

Determine Requests

```
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;
```

10/25/2006

Copyright © 2006, ApsDBA Consulting, All Rights Reserved.

Slide 32

Key Information

- Get all requests by queue name, program and run time
- Use input parameters for date and queue
- Output:
 - STIM - Start time
 - SMIN - Start minute
 - RMIN - Run minutes (interval duration)

Print Markers

```

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.CMQUEUE);
  DBMS_OUTPUT.PUT_LINE(CHR(13));
  DBMS_OUTPUT.PUT_LINE('Time                               1      2      3      4      5');
  DBMS_OUTPUT.PUT_LINE('MIN:                               01234567890123456789012345678901234567890123456789');
--
  -- Create overlap view for each queue
  --
  FOR cm_runtime_rec IN cm_runtime_cur('&var_date' || ' ' || '&var_hr', cm_queue_rec.CMQUEUE) 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;

```

Create Overlap

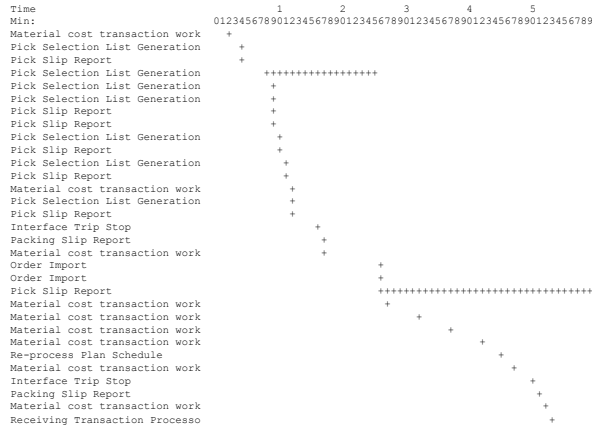
- Using PL/SQL output a marker for each minute the job was active during the interval from 0 to 59 minutes for a 1 hour interval.

Queue Overlap Output

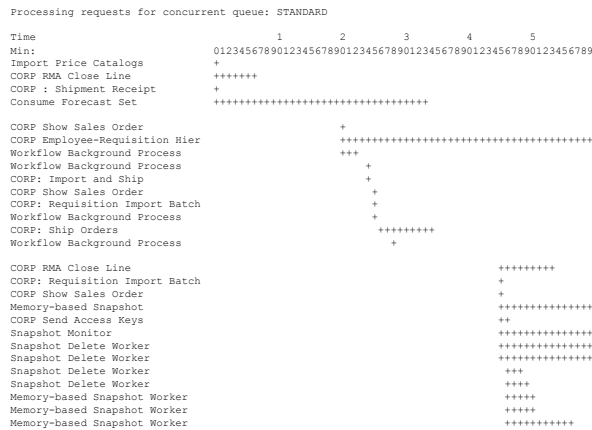
```

Time
Min: 012345678901234567890123456789012345678901234567890123456789
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 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 +
Re-process Plan Schedule +
Material cost transaction work +
Interface Trip Stop +
Packing Slip Report +
Material cost transaction work +
Receiving Transaction Processo +
    
```


Queue Overlap Analysis



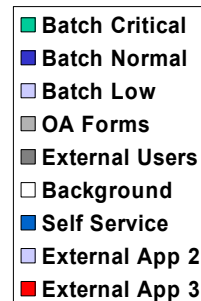
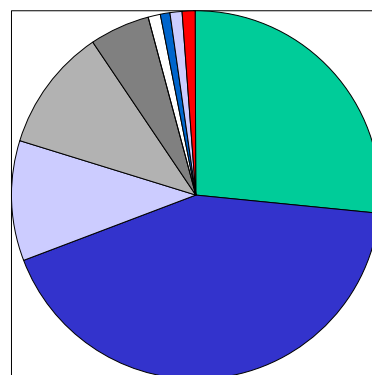
Another Example



Results

- Critical batch job start delays minimized, preferably non-existent.
- Interactive response time maintained.
- Non-critical batch jobs delayed until either a queue process becomes available or a workshift change and more processes are made available.
- We expect to see start delay times lengthen for low priority batch jobs!

New Environment (est.)



Success!

- Now we have the information we need.
- We can determine exactly how many processes are required to meet the demand for an interval period.
- With this information we also know how much we may have to throttle the other queues to insure that critical processes run.
- We now have a tool to use to help see how to further adjust these numbers to protect interactive users.

Conclusion

- Queue overlap analysis helps provide a more accurate picture of how many jobs run and when in a concurrent manager queue for a given interval.
- This allows more precise sizing of queues to meet demand.
- This also provides the ability to make other workload decisions.

Questions?

Thank You

This presentation and more
information is available at
www.appsdba.com