

Concurrent Managers 101 and 202

By
Gary Piper
www.piper-rx.com

1 Introduction

One basic performance issue many sites have in common across the world relates to the configuration of concurrent managers.

This paper will review a couple of real world concurrent manager configurations, it will discuss in detail each of the configurations, identify the advantages and disadvantages of each configuration and the issues caused by inappropriate configurations. The paper will specifically focus on the details of each configuration, including:

- ❖ Appropriateness of the number of concurrent managers
- ❖ Configuration and usage of Custom Managers including FSGs
- ❖ Appropriate work shifts
- ❖ High transaction rate managers
- ❖ The impact on the application
- ❖ The impact on the business
- ❖ Alternative configurations e.g. corrective action

The areas we will focus on in this paper are:

- How many concurrent managers should you have?
- Sleep and cache settings
- Unleashing the power of Concurrent Manager Request Types

2 How many concurrent managers should I have?

It is a common misconception that the more managers you have the better throughput you will get. This situation is further compounded once the business finds out you can add managers. They too generally believe there are spare resources you are not utilising, and therefore that you should add more managers.

The problem is, once you have added managers it is almost impossible to convince the business that you need to reduce the number of managers in order to guarantee throughput and prevent performance bottlenecks.

I have seen many such cases over the years...E.g. A site with a 12 CPU machine had configured over 50 standard and custom managers. It took over 3 months of negotiation and testing to convince the business users that reducing the total number to 24 was the optimal for the site. Overall performance did increase when this change was implemented..

When configuring managers it is best to start low, and work up; it is easier to get permission to add managers than to take them away.

A general rule of thumb is you should have no more managers than 2 (standard and custom) times the number of CPUs (optimal is 1.8 as this leaves head room for user connections). It always amazes me how many managers I find configured at sites. For example:

- ❖ Thirty-five (35) on a 2 CPU box (17.5 per CPU)
- ❖ Seventy (70) on an 8 CPU box (8.8 per CPU)
- ❖ Forty eight (48) on a 12 CPU box (4 per CPU)

The one thing these all sites had in common was CPU bottleneck and I/O issues, particularly during high processing times like month end.

Tip: A reasonable indicator as to historical bottlenecks is a wide variation in concurrent request run times

Another issue often encountered is the doubling or addition of manager processes for overnight processing. Sites do this on the assumption that there are no active users overnight and hence no one to experience performance issues. This action can also cause bottlenecking.

Always resist the urge to add managers

So how many managers should I have?

- ❖ Generally no more than 2 per CPU
- ❖ Custom managers for FSGs (Assume 1 FSG will consume one CPU)
- ❖ Custom managers for selected slow jobs
- ❖ Custom managers for selected fast jobs

Tip: When you creating custom managers do not forget to reduce the number of standard managers.

2.1 Case Study 1: Too many managers

The following example is from a site running on an eight (8) CPU machine:

Manager Name	Max Proc	Run Proc	Sleep	Cache
FNDICM	1	1	30	0
STANDARD	70	70	30	100
FNDCRM	1	1	30	0
FNDSCH	1	1	30	0
INVMGR	1	1	60	5
PODAMGR	3	3	60	0
RCVOLTM	3	3	60	0

Note: The site runs Financial Statement Generator reports.

This particular site had a number of additional issues:

- ❖ Heavily customised
- ❖ Several custom programs that have an average run time of over 10 hours.
- ❖ Had not purged concurrent requests for over 6 months
- ❖ Account Hierarchy reports take over 14 hours to run
- ❖ This site has a seven segment chart of accounts with over 200,000 code combinations and no indexes.

- ❖ Other sessions:
 - 103 Discoverer sessions
 - 30 Forms users
 - 1 ADI session
 - 5 Toad session ← On Production
 - 1 SQL PLUS ← On Production

It will take a little more than code tuning is required to fix this site!

So beware, in cases such as this, just fixing the concurrent manager configuration is not a silver bullet.

3 Sleep and cache settings

When a queue wakes up (sleep time) it queries the *FND_CONCURRENT_REQUESTS* table for any pending requests. The value of the cache size will determine the number of rows that will be returned. If a request completes within the managers sleep time, additional requests held in cache can be run prior to the next manager wake up time.

3.1 Case study 2 – Sleep and cache settings need review as the business changes

The following example is from an equipment hire company. Inventory is either dispatched or received when an item is either hired out or returned. The item is scanned and a concurrent request is submitted to record and set the new inventory level. The company was quite successful and has grown remarkably, increasing the number of hire outlets from 5 to 40 over a 2 year period.

Environment:

- ❖ 4 CPU box
- ❖ 8 standard managers
- ❖ No custom managers
- ❖ The concurrent program (inventory scanning) runs in less than 6 seconds.

Issue

When the company was small the scan rate was 1 scan per 1 – 2 hours. Items are now scanned at a rate of up to 10 per minute which means at any given time all eight (8) standard managers may be running the inventory program, with a back log of two requests. At certain times of the day this caused a severe backlog of concurrent requests. It also caused internal issues as the backlog affected the normal running of the business.

Let's look at and evaluate a range of possible remedies that might be considered:

Option 1 – One Manager

- ❖ Single Manager Sleep 60 with cache 2
- ❖ 2 runs @ 6 seconds = 12 seconds
- ❖ Wasting 48 seconds of processing per 60 seconds, 80 % waste
- ❖ Input rate 600 per hour
- ❖ Run rate 120 per hour
- ❖ The deficit of 480 runs per hour

Option 2 - Three Managers

- ❖ Three Managers, Sleep 60 with cache 2
- ❖ 6 runs @ 6 seconds = 36 seconds
- ❖ Wasting 144 seconds of processing per 180 seconds 80 % waste
- ❖ Input rate 600 per hour
- ❖ Run rate 360 per hour
- ❖ The deficit of 240 runs per hour

...plus additional processing and memory overhead for the additional managers.

Option 3 - Single manager with amended sleep and cache settings

- ❖ One Manager. Sleep 60 with cache 10
- ❖ 10 runs (current load) @ 6 seconds = 60 seconds
- ❖ Wasting 0 seconds of processing per 60 seconds, 100 pct
- ❖ Input rate 600 per hour
- ❖ Run rate 600 per hour

Option 4 – 2 managers with amended sleep and cache settings

- ❖ Two Manages, Sleep 60 with cache 10
- ❖ 10 runs (current load) @ 6 seconds = 60 Seconds
- ❖ Input rate 600 per hour
- ❖ Possible run rate 1200 per hour, allows up to 100 % for growth

Option 5 - A more ideal solution

- ❖ Create a custom manager fast with two (2) processes
- ❖ Set sleep 60 and cache to 10
- ❖ Decrease the number of standard managers by 1
- ❖ Assign the concurrent program to that manager
- ❖ Exclude the concurrent program from the standard queue

4 Case Study 3: What bad can look like...

The following is the configuration from a site running 8 CPUs:

Manager Name	Max Proc	Run Proc	Sleep	Cache	Pending
Internal Manager	1	1	15		
Conflict_Resolution_Manager	1	1	30		48
Financial Statement Generator	12	12	60		
General Accounting	10	10	8		128
INV Remote Procedure Manager	10	10	30		4
CRP Inquiry Manager	2	0	60		
Inventory Manager	1	1	60	1	
MRP Manager	1	0	60	1	
Material Transactions Manager	3	3	300		3
PA Streamline Manager	1	0	1	1	
Shipping Transaction Manager	1	0	60		
Standard (Disabled)	0	0	60	1	128

Observations:

- ❖ Internal manager sleep time too low
- ❖ General Accounting manager sleep time way too low
- ❖ 22 standard and custom managers on an eight (8) processor box is too many
- ❖ Financial Statement Generator with 12 processes will allow 12 concurrent FSG's this will the eight (8) processor box
- ❖ Mismatch between the General Accounting and CRM manager sleep time
- ❖ Disabled Standard Manager

I would like to have seen the specialisation rules they used for this one! But even seeing them I can tell they have not been configured correctly as there are 128 requests pending in the disabled standard manager. Looks like the programs have been allowed in the General Accounting manager and have not been disallowed in the Standard manager.

5 Specialisation rules

Specialisation rules are used to assign specific concurrent manager programs to specific queues.

Specialisation rules can be one of the more complex Oracle Applications features to set up and once set up, one of the more complex to find using the current E-Business Suite screens. This makes it hard for an administrator taking over a site to understand and troubleshoot. Remember, with specialisation rules simplicity is the key.

A common problem I have witnessed is where a concurrent program has been “allowed” in the custom queue but has not been disallowed from its default manager (usually standard). This will cause the concurrent program to be run in either queue depending on availability, which I believe defeats the purpose of assigning the job to a specific queue.

To add to the complexity, there are a number of specialisation rule combinations:

- ❖ By Program
- ❖ By Application user
- ❖ By Oracle User
- ❖ By Request Type
- ❖ By Complex Rule, which is a combination of Program, application user and / or Oracle user

Always try and avoid complex rules; once set up they are **very** hard to interpret

5.1 Case study 4: Creating a custom manager and setting specialisation rules

Let's work through the following example: Create a custom concurrent manager called “Slow” and assign a program to it.

The basic steps are as follows:

- ❖ Create a custom manager “Slow”
- ❖ Activate the manager
- ❖ Reduce the number of standard managers
- ❖ Ensure the new manager is active
- ❖ Set the specialisation rules:
 - “Include” the selected program in the Slow manager
 - “Exclude” the selected program from the Standard manager

Once the changes to specialization rules have been committed, the [FND_CONCURRENT_WORKER_REQUESTS](#) view will be rebuilt reflecting the new configuration and the managers will be restarted.

From this point on the selected program will run in the “Slow” queue and not in the standard queue

Tip: It is not a good idea to change specialisation rules during high processing times. As bouncing the managers during this time would be frowned upon

Tip: I have encountered on a number occasions that after the creation of specialisation rules the [FND_CONCURRENT_WORKER_REQUESTS](#) view does not rebuild. It is possible to rebuild the view concurrent worker requests manually.

FNDLIBR FND FNDCPBWV apps/apps SYSADMIN 'System Administrator' SYSADMIN

Tip: New custom developed programs should be initially assigned to the “Slow” queue and earn the right to be moved to the “normal” queues.

6 How to configure Concurrent Manager Request Types

A request type is a logical folder which is assigned to a specific queue by specialisation rules e.g. request type “Fast” is assigned to the concurrent manager “fast” queue and disallowed from the standard manager. It follows that when a program is assigned the request type “Fast” that program will run in the “fast” queue, and when the request type is removed from the program the program will run in its default queue.

So in affect what you do is assign a program to a request type and that request type will determine the manager that will run the program.

An example where this could be useful might be where the standard manager is bottlenecked and the slow manager is relatively free; you could assign particular programs to the slow manager for a period until the back log is cleared then assign them back to the standard manager.

Another example is to assign particular programs to an overnight queue. Or you could set up a “special” queue for when you want to help users.

The limitation is that you cannot change a request’s queue once the request has been submitted. For scheduled jobs that have a resubmit, the current scheduled job will run in the originally assigned manager where as the new submission will run in the manager assigned by the request type you set.

Tip: Don’t let the users know you can do this as you will get large numbers of “special” requests.

One of the powerful features of request types is that it allows you to change the queues on the fly without rebuilding the [FND_CONCURRENT_WORKER_REQUESTS](#) view and restarting the concurrent managers.

6.1 Creating a request type

Step 1: Using the Oracle E-Business application, first create a request type:

System Administrator > Concurrent > Program > Types

Name	Application	Description
SLOW	Application Object Library	Request Type for slow requests

Step 2: Assign a request type to a manager using specialisation rules:

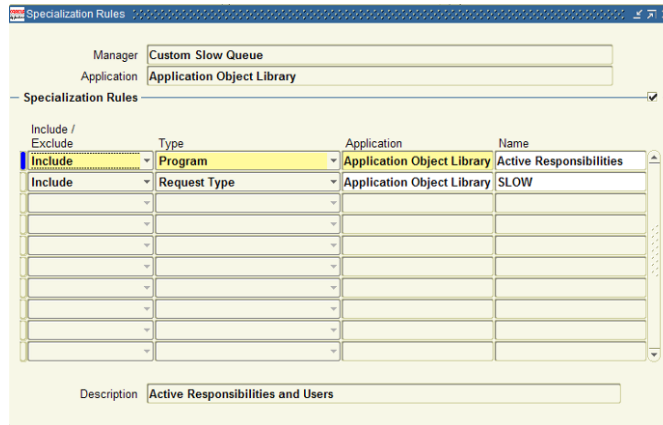
System Administrator > Concurrent > Manager > Define

Select the standard manager and then the Specialisation Rules button. **Exclude** the request type from the Standard Manager.

Include / Exclude	Type	Application	Name
Exclude	Program	Application Object Library	Active Responsibilities
Exclude	Request Type	Application Object Library	SLOW

Description: Request Type for slow requests

Step 3: Now select the Custom Manager and then the Specialisation Rules button. **Include** the request type for the Custom manager:



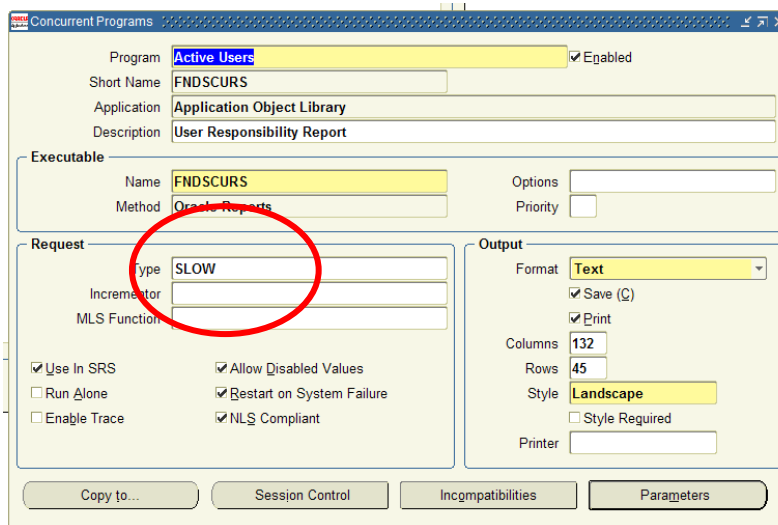
Once the specialisation rules have been created, the [FND_CONCURRENT_WORKER_REQUESTS](#) view will be rebuilt with new configuration and the managers will be restarted.

Note: There will be no change in the running of the programs at this time.

Step 4: Assign a program to a request type:

System Administrator > Concurrent > Program > Define

Add the request type to the selected program:



All new requests for this program will now be run in the “Slow” queue. If you remove the request type all subsequent requests for this program will be run in the standard queue.

Examples of use:

#1 Overnight:

Only allow selected programs to run overnight.

- ❖ Create a request type Overnight
- ❖ Create a manager that runs only overnight (work shifts)
- ❖ Disallow the request type from standard / Allow in the overnight manager
- ❖ Assign the program you wish to run overnight to the Overnight request type

If a person tries to run the program it will have a status of Pending error; until the overnight manager starts.

#2 Preventing specific programs from being run:

- ❖ Create a Request Type Don't Run
- ❖ Create a manager Don't Run with no processes
- ❖ Disallow the request type from standard / Allow in the Don't Run manager
- ❖ Assign the program you do not wish to run to the Don't Run request type

If a person tries to run the program it will have a status of Pending error; the user can be talked to about this and the job cancelled.

Tip: It is far simpler to trace request type rules than to trace individual specialisation rules.
--

7 Disclaimer

All material contained in this document is provided by the author "as is" and any express or implied warranties, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of any content or information, even if advised of the possibility of such damage. It is always recommended that you seek independent, professional advice before implementing any ideas or changes to ensure that they are appropriate.

Oracle®, Oracle Applications® & Oracle E-Business Suite® are registered trademarks of Oracle Corporation