

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?



HOW IS RAC PERFORMANCE TUNING DIFFERENT?

After working on numerous performance problems on both RAC and single-node systems, I thought it would be helpful to reflect on what I see as differences between the two. I wanted to specifically address the question, “*What exactly do I do differently in performance tuning on a RAC cluster compared to single-node?*”

Note that the perspective of this paper is above all, *practical*. For just a moment, forget about theories on cache fusion, or marketing fluff about “self-tuning” databases, magical tuning-tools, and the like. I am not interested in what RAC “could do,” “should do,” or “might do.” I am only interested in what I do differently in *practice*—that is, resolving my customers’ performance problems.¹

I think my answer as to “what is different with RAC” will surprise you. First, however, let’s remind ourselves about the scenarios where RAC can really make a difference.

WHERE RAC HELPS

RAC offers a good option for improved *availability*, as well as easier *scalability*. So RAC, with it’s “scale out” approach gets around maximum cpu limitations.

That is not at all the same thing as better *performance*. Occasionally, marketing folks, in their zeal, suggest that RAC will actually make things go *faster*. Of course, that’s really not true. Using RAC doesn’t somehow make things work faster. What the marketeers really mean is that the overall throughput of your application will be larger if your application is strictly cpu-limited, and simply needs more processing power. Thus, RAC provides the opportunity to increase the number of threads run concurrently.

¹ For purposes of this paper, I assume that the RAC environment is properly configured.

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?

We must always remember, however, there are numerous other bottlenecks that must be considered as well. For example, you might actually be experiencing waits due to available log buffers. In this scenario, adding more CPUs would be counterproductive.

RAC cannot make an application scale, if it otherwise would not. As Oracle's Admin Guide states, "If an application does not scale on an SMP system, then moving the application to an Oracle RAC database cannot improve performance."²

So, it's worth remembering that performance issues on a single-node systems won't magically go away once RAC is installed. If anything, performance issues are intensified unless you actually resolve the underlying issues.

For further information, see my earlier paper, [Performance Tuning: Is RAC Like "Bolt-on Power?"](#)

THE ESSENCE OF PERFORMANCE TUNING IS THE SAME

In practice, I have found the steps I use to accomplish performance tuning to be nearly identical, whether on a RAC cluster, or on a single-node instance. The differences are mostly pretty mundane details, such as changing V\$ to GV\$, or trivial (but necessary) details, such as making sure you're querying the correct node.

I think some DBAs will be surprised by my perspective. After all, a RAC system is a lot different than a single-node system. So how can performance tuning be so similar?



HOW CAN THIS BE?

² Oracle Real Application Clusters Administration and Deployment Guide 11g Release 1 (11.1) This limitation noted in the official Oracle documentation has actually been used by RAC competitors as an argument against RAC. Actually, however, it simply points out the logical limitations of *any* clustering technology.

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?

The reason is really a tribute to RAC technology—especially the efficiency of cache fusion. Of course, no one would argue that cache fusion is “free,” but in practice, I’ve found that the overhead due to RAC is not very high—in fact, in most cases, I don’t even care that I’m on RAC.

RAC OVERHEAD

Let’s take a look at the overhead due to cache fusion. These statistics come from a very large database, on an 8-node cluster. In particular, let’s look at one very busy node, which handles OLTP queries.

For a recent 24-hour period, let’s see what the top wait events were. We’ll use the familiar AWR report to glean this information:

Top-5 wait events for a 24-hour period:

db file sequential read	277k secs
CPU time	140k secs
gc current block 3-way	24k secs
gc cr grant 2-way	16k secs
log file sync	8k secs

The above chart confirms that RAC-specific events, while certainly noticeable, aren’t anywhere close to being the top-wait events. Roughly speaking, they amount to about 15% of the total wait time. Let’s examine this issue more closely.

“What! Does RAC really cause degradation of 15%?”

I don’t think so—here’s why: The above RAC wait events—while indeed true delays—are occurring so that *even greater delays will be avoided*. The inter-node block transfers are occurring in order to avoid disk reads. With RAC, we have a multi-node, monstrous-sized cache, and another node has the desired block cached. So, we do a little work (block transfer) to *avoid* bigger work (disk reads).

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?



LET'S CHECK SOME NUMBERS

In practice, on very large RAC systems, I have found that Oracle spends roughly 1 ms getting a block in order to avoid spending 5 ms for a single block read (sequential read.)

Let's take another view of the impact due to RAC wait events. Again from a recent AWR, here is a list of the top sql which are impacted by RAC:

SQL ordered by Cluster Wait Time

Cluster Wait Time (s)	CWT % of Elapsed Time	Elapsed Time(s)	CPU Time(s)	Execs
4,820.79	15.29	31,525.18	2,429.77	300,054
4,649.80	19.62	23,695.91	1,867.30	189,178
4,040.22	59.84	6,751.62	1,181.02	617,979
3,397.00	19.72	17,226.23	1,010.73	319,561
2,443.81	13.98	17,476.77	1,001.37	318,937
2,111.91	14.81	14,259.06	790.73	24,010
2,013.39	10.53	19,127.11	1,003.17	212,313
1,509.99	18.13	8,327.73	502.36	209,420

Observing the statistics above, we see that indeed the overhead of RAC is about 15% in most cases. It might at first seem like there is a real problem with one of the sql shown above, because the CWT (cluster wait time) is about 60% of the total run time. This is misleading, however. The sql in question runs so fast (about 1 ms) that the small delay from cluster overhead becomes a large chunk of the runtime. In practice, however, a total runtime of 1 ms is so extraordinarily fast, that I think it's fair to ignore this datum.

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?

I think it's reasonable to say that the 15% figure represents an *upper-range* for the RAC overhead. Statistically, one could say that RAC causes 15% degradation—but only when compared to a single-node instance with a single, similarly-monstrous cache. And that is probably not a realistic comparison.

RAC DIFFERENCES

In practice, the #1 difference I experience will likely seem absurd at first: simply figuring out on which node the problem sql is running. With eight nodes, however, this is not always as trivial as it seems. Oftentimes, a user will complain of some issue with a report, but they honestly don't know the node.



WHAT NODE ARE YOU ON?

LONG_SESSIONS.SQL

Here's an easy way to home-in on the problem session. The script below identifies sessions, across all nodes, that have been running for at least 8 hours³:

```
Col Sid Format 99999
Col Serial# Format 999999
Col Machine Format A15 Truncate
Col Event Format A30 Truncate
Col Inst Format 9999

Select Inst_Id Inst,Sid,Serial#,Username,Machine,Event,
Logon_Time,Sql_Id,Prev_Sql_Id
From Gv$Session
where type != 'BACKGROUND' and event not like 'SQL*Net%'
and event not like 'Streams AQ: waiting%'
And Nvl(24 * (Sysdate - Logon_Time),0) > 8
Order By Username;
```

ALL_ACTIVE.SQL

³ This script courtesy of Ken Jordan of PG&E, San Ramon, California.

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?

This script is not so selective; however, it shows all active sessions on all nodes—not just the long-running sessions:

```
SELECT DISTINCT osuser, gv$session.inst_id, Sid, username,  
Substr(program,1,19) PROG , sql_text  
From gv$Session, gv$sql  
Where status = 'ACTIVE'  
and gv$session.inst_id = gv$sql.inst_id  
And username is not null  
And gv$session.sql_hash_value = hash_value  
and gv$session.sql_address = gv$sql.address  
and sql_text not like '%DISTINCT osuser%'  
order by 2;
```

ASH_BY_TIME.SQL

Very often, I hear of a problem report such as, “My report ran too long last night.” The user will know the approximate start/end time, but will almost never know the node. (Plus, sometimes, various threads run on multiple nodes.)

One of the first scripts I run is a simple ASH script that categorizes the long-running sql by node, over a particular time period.

```
With P1 As (Select /*+Parallel(A 6) */  
Distinct A.*  
From Dbahist_active_sess_history A  
Where Sample_Time Like '22-APR-10%4.%AM'  
) Select Instance_Number, Sql_Id, Count(*)  
From P1  
Group By Instance_Number, Sql_Id Having Count(*) > 20  
Order By Count(*)
```

In the script above, I look for the activity, for all nodes, at 4 to 5 A.M. on April 22. I employ Parallel 6 to reduce the runtime due just a minute or so. I find it convenient to use query subfactoring (the “with” syntax), but of course that is not mandatory.

HIGH RESOURCE.SQL

Many performance drilldown queries are based on V\$ views. I’ve found it easiest to substitute the GV\$ view in most cases. Keep in mind, however, that behind the scenes Oracle will perform a union of the underlying V\$ views, possibly making the runtime quite a bit longer.

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?

One of my most often-run scripts queries *V\$\$sql* to find high-resource sql. With RAC, you need to slightly alter this to query *GV\$\$sql*. Additionally, the result set will return the sql from each instance meeting the criteria.

```
Select Inst_id, Sql_Id, First_Load_Time, Round(Elapsed_Time/1000000)
Secs, Rows_Processed, Executions, Buffer_Gets, Disk_Reads, Sql_Text
From Gv$$Sql
Where Upper(Sql_Text) Like '%SELECT TBD%'
And Executions > 0
Order By 1
```

In the above script, Oracle references *Inst_Id*, but the exact field name changes from view to view! Just when you think you've got it figured out, Oracle switches the column name.



WHAT WAS THAT NAME, AGAIN?

Note that Oracle uses slightly different field names to identify the instance, depending on the data dictionary view. I count eight different ways:

```
INST#
INSTANCE
INSTANCE#
INSTANCE_ID
INSTANCE_NUM
INSTANCE_NUMBER
INST_ID
INST_NUMBER
```

After seeing all the ways that Oracle can identify an instance, I feel much better about getting the name mixed up all the time.

SOME OTHER DIFFERENCES IN RAC

AWR REPORTS

Here's another nit-picky difference. With RAC, you need to pick a node for which you want an AWR report. (At least, if you're running the official AWR

PERFORMANCE TUNING ON ORACLE RAC VERSUS SINGLE-NODE: WHAT'S DIFFERENT?

report script.) On a two-node cluster, not a big deal. Not quite so trivial on an eight-node system. Of course, you can always guess, and run multiple reports.

RAC-SPECIFIC BUGS

In my experience, this has proven to be a major issue. There are some bugs peculiar only to RAC. For example, Oracle 10.2.0.3 has a RAC bug related to parallelism. With RAC, the child parallel processes will often jump to a node different than where the parent query originated. There is nothing wrong with this, but when the query is particularly complicated, with multiple-layers of parallelism, Oracle gets confused, and the parallel processes spin forever. Of course, we applied this patch, but in the meantime, there was much confusion as to what was happening.



FINALE

Over the course of resolving thousands of performance issues—both on RAC and on single-node systems, I have slowly realized that there are few major differences in the performance tuning process. In practice, what I do to resolve a performance issue is amazingly similar. I just remember to ask that question, “What node are you on again?” Of course, depending on your particular business or configuration, you may encounter very specific differences in your RAC setup.

In general, the same clear, logical thinking process works in both environments. If you are adept at solving performance problems in single-node, you will likely be similarly adept in the RAC environment. Likewise, if you are rather clumsy at solving performance problems in single-node . . .

* * *

Chris Lawson is an Oracle *Ace* and performance specialist in the San Francisco Bay Area. He is the author of *The Art & Science of Oracle Performance Tuning*, as well as *Snappy Interviews: 100 Questions to Ask Oracle DBAs*. Chris would be happy to hear from you. He can be reached at, RAC@OracleMagician.com