You Probably Don't Need RAC

If you've been holidaying in Siberia or similar places for about a year, you have probably not talked to an Oracle Sales rep yet about RAC. But you will no doubt find that there's a voice mail waiting for you when you turn your mobile phone on again after returning home from the vacation.

RAC is being pushed very hard by Oracle. You will get high availability, incredible scalability, a much improved personal life, the ability to partition workloads, buy cheap Linux servers and what have you.
It sounds pretty good. How can anyone say no to that kind of offer?

RAC is not OPS

No, RAC is not OPS, but it looks a lot like it. Oracle Marketing tries really hard to distance RAC from OPS, and I don't understand why. I mean: If the basic code has been around for many years it means it's stable, debugged and tried. If it's all new, who dares install it in a critical system? Fortunately, it's not true that RAC is not OPS. The basic parts of the code – GES and GCS – are pretty much the same as they've always been.

GES stands for Global Enqueue Service and GCS stands for Global Cache Service. More about that later.

A little history: OPS was created for version 6 of Oracle. The only clusters around then were VAX/VMS clusters, but unfortunately the VAX/VMS Distributed Lock Manager (DLM) was created originally to handle the coordination of relatively few resources, such as files and devices, not 1000s of buffers in a buffer cache (Oracle or others). It proved way too slow for OPS.

So Oracle had to create their own DLM for VAX/VMS, which they did. It took a while, though, so it wasn't until 6.0.35 (which was called "6.2" to celebrate the OPS feature) that it finally came out.

I remember taking one of the first OPS classes (in Chicago) shortly after joining Oracle and thinking that Oracle Development had gone mad – creating their own DLM instead of letting the Digital guys do it (they had, after all, created the clusters and the whole concept).

I was wrong. Oracle's own DLM worked very well, and Digital adopted Oracle's technology and ideas in their own DLM, so when version 7 of Oracle came out, it was again Digitals native DLM that was used.

The UNIX vendors then started doing Clusters (well, NCR had done it for a while). And they mostly got the DLM technology from Oracle.

Microsoft certainly didn't get their DLM technology from Oracle when they started making Windows clusters. Oh no. They got it from Digital .

Fun Fact: The GES/GCS code was already in Oracle version 5. Bjørn Engsig, who has worked with Oracle source code since 1983, found out about this and implemented his own, very crude, lock manager on a Danish unix system running version 5. He got it to work, but only for demonstration purposes – his home-written lock manager basically used database-level locking which is not really useful .

PING

Oracle had to make sure that a buffer wasn't modified by two different processes at the same time – which one should then be written to disk later? So instead of "just" serialising the access to one copy of the block in one buffer (which can be achieved with the combination of hash buckets, chains and latches that we know so well), Oracle had to coordinate several copies in several buffer caches across nodes.

This was achieved using a new kind of locking (called Parallel Cache Management or PCM locks) which was coordinated across nodes/instances using the DLM and various background processes.

When there was a "conflict", ie the same block/buffer was requested by more than one instance, the "exclusive" lock held by the first "holder" had to be down-graded to a "shared" lock held by all "holders". This down-grade/sharing could only be done by first making sure that all holders were seeing the same image of the block/buffer.

So the copy of the block that was in the buffer cache of the first holder was written to disk and then that copy of the block was read into the other buffer caches. The term "ping" was introduced to describe other instances requesting a buffer held exclusively by one instance.

Pinging via disk is slow. If you had an index on a column that kept growing on the right-hand side the right-most leaf block could get pinged back and forth non-stop between instances. Pinging via disk could kill your system's performance.

The workarounds included data partitioning, temporary tablespaces (introduced in 7.3) where each instance had their own latch instead of a shared Dictionary lock (ST-lock – remember the ora-1575?), reverse indexes (7.3) which meant that it was random which leaf block you would hit even if you had monotonically increasing indexing) and other tricks.

Oracle 8i: Cache Fusion introduced

Oracle 8i (that's 8.1 where the dot is moved on top of the 1) introduced a new way of pinging via the HSI (High-Speed Interconnect) or similar mechanism, ie a kind of memory-to-memory transport instead of memory-to-disk-to-memory. It's not easy to do, and it was initially only done for CR blocks/buffers.

It worked for some and didn't work for others. On several OPS installations here in Denmark they had to deliberately turn it off in 8.1.6 and 8.1.7.

By the way: Oracle had introduced their own, generic Lock Manager (LM) mechanism in Oracle 8.0, signalling that they would soon be pretty independent of the DLM code from the various vendors.

You could say that the LM was the equivalent of the Oracle source code being OS independent and then having a small layer in the code known as the OSD (Operating System Dependent). With the introduction of the integrated LM Oracle only had to manage a small OS-dependent layer for each port – the rest was generic code. Respect again to the engineers at Oracle Development.

Oracle9i: Cache Fusion all over – and a new name

With Oracle9i (called 9.0 and 9.2 just to confuse the enemy) all pinging is done via the memory channel or high-speed interconnect. That's it.

But just as it was time to call the version 6.2 instead of 6.0.35 back then, it was time to call it RAC instead of OPS.

Oracle sales people actually started dissing OPS which they had been promoting for a decade. At least they did here in Denmark

A lot of the way RAC works is of course just like OPS worked (and works in many installations still).

Of course RAC is smarter. Way smarter. Much improved technology, etc.
But don't forget that the engineers at Oracle build on solid, tried and tested code which they then improved. For instance the GES and GCS layers in the code.

So why is RAC better than OPS?

For two main reasons:

First, pinging via disk limits scalability, so pinging via memory channels will improve scalability because it's faster.

How much faster? That's a very, very good question. Oracle needs to do a lot of checking, latching, etc. in order to ensure coherency in many ways. For the best answer available at the moment, please see Cary Millsap's article on www.hotsos.com on why one should focus on logical IO's instead of physical IO's. It would appear that logical IO's are in the vicinity of 100 times faster than physical ones.

This is in stark contrast to pure memory IO done by operating systems – they are perhaps up to 10000 times faster than a disk IO. But Oracle has to do lots of things (for which we love it), and that has an impact. This necessary overhead is of course higher with RAC since more checking still has to be performed.

Second, clever tricks have been put into the code in order to make all sorts of coordination tasks between instances faster, easier – and sometimes even avoidable. The best tuning is as always not to do it at all. If Oracle doesn't have to send a copy of a buffer across to another instance it will try to not do.

Does that mean that RAC will give you a better life? Yes and No. Or as any good consultant will say: "It depends".

Here are the things to consider before you go RAC'ing all over the world: Price, availability, scalability, manageability, skills required and troubleshooting.

Price

This section talks about Oracle list prices. Discounts may vary .

Oracle Enterprise Edition costs US$40.000,- per cpu or US$800,- per named user plus (NUP), as it's called now. RAC costs 50% on top of that, which means US$60.000,- and US$1200,- per cpu or per NUP.

As I write this, I'm aware that RAC has been offered at a 50% discount, ie US$10,000, on the American market since around January or February. But it's not something officially reflected in the global price list.

(By the way: The Partitioning option costs 25% on top of the cpu/NUP price. OLAP and Data Mining are 50% each. Spatial, Advanced Security and Label Security are 25% each. Diagnostics Pack, Tuning Pack, Change Management Pack and Management Pack for SAP R/3 are US$3.000,- and US$60,- per cpu/NUP.)

So let's play around with Larry's vision of cheap Intel-based Linux clusters. Let's buy those two cheap, 4-cpu Intel boxes and put them together in a cluster with Oracle9i and RAC on top:

Price for the hardware: About US$15.000,- or so.
Price for the OS (Linux): About US$0.50,- or thereabout (it depends!)
Price for Oracle w/ RAC: US$480.000,-

So that's half a million to Oracle. Put another way: It's 1 dollar to the box movers for every 32 dollars Oracle gets.

Psychologically it's hard for the customers to understand that they have to buy something that expensive to run on such cheap hardware. The gap is too big, and Oracle will need to address it soon.

There's nothing like RAC on the market, but that doesn't mean you have to buy RAC. I usually joke that it's like buying a car for US$10.000,- that has all the facilities you need from a good and stable car. Airbags and ABS brakes are US$500.000,- extra, by the way. Well, airbags and ABS are wonderful to have and they increase your security. But it's a lot of money compared to the basic car price.

There are other indirect costs associated with going RAC: You'll need more skills in your organisation, both with respect to RAC and clusters. If your organisation is not familiar with clusters you'll need to learn a lot, for instance.

You'll also have to consider to have a development environment (and maybe a test environment) that consist of both a cluster and RAC. Sometimes Oracle will let you run Oracle for free on those systems, sometimes not (it depends).

RAC is very cool technology. But it's expensive.

Availability, Part 1: 99.x% -> 98%

I think there are two ways of looking at availability (but one day I will as usual be proved wrong, of course).

One way of looking at availability is this: If you have a standalone Unix box it will usually give you 99.9% availability over a year (some say 99.5, some say 99.9). It just runs. And so does Oracle usually. If you have a two-node Unix cluster the availability over a year drops to 98%.

Yep, quite surprising, but the two main reasons are that the increased complexity (extra layers of code, extra hardware, etc.) introduced with clusters and RAC will cause some additional downtime – and that it just takes longer to boot a cluster, startup RAC, and perform some other management tasks.

And if you believe what Gartner Group, Oracle and others are saying, namely that 70% or more of downtime is caused by human errors and lack of knowledge…well, what will happen when you introduce more complex hardware and more complex software?

Another way of looking at availability is this: Your standalone box is available 99.9% of the time. That 0.1% is what the other nodes in a cluster are for.

I have worked with clusters (VMS, Unix and now Windows) since around 1988 or 1989 or whenever 6.0.35 came out, and clusters are just harder to setup, manage and run. If you need to run clusters in your business (for whatever valid business reasons) you also will need extra personnel, extra consultants and extra skills in your organisation.

As a technical director in a small company that sells rather specialized consultants, I'm of course delighted when the complexity of a customer's setup increases. It means they'll call us (or Oracle) sooner or later. Since we live off catastrophes, problems and troubles, I'm looking into a bright future, I'm sure .

What are the alternatives? Various standby database options. That can be either the standard standby database introduced in 7.3, or Data Guard, or some 3rd party tool like Shareplex from Quest (I have absolutely no knowledge of this specific product – I just know it exists, so don't buy it without listening to people more clever than me).

Note, that Oracle's pricing policy changed recently regarding standby options (as pointed out by one alert contributor on the Oracle-L list) so suddenly you now have to

pay full license for the standby nodes if you use them more than 10 days a year. And always full price for the Data Guard nodes.

You could of course also create something fancy and creative yourself. We used to do standby databases back in version 6 by applying archive logs manually on another database in constant recovery mode. Lots of issues, of course. But it was done. Or you could use log miner to extract DML from the archive logs and apply them on a standby database. Or you could have system triggers that caught all DDL and DML on a system and put them in load files that were then loaded in real time, near real time or much later on another system.

Those kinds of alternatives will need a little work, but they have one thing in common: You could even do it with Oracle Standard Edition, which means that the price drops from US$40.000,- per cpu to US$15.000,-.

Availability, Part 2: 25/8/370?

Recently, a client of mine were down for close to five hours because they needed to upgrade their RAC from 9.2.0.1.0 to 9.2.0.3.0. They were well prepared and everything went as planned. It just took close to five hours. Here are the steps they went through:

0. Shutdown of Weblogic
1. Shutdown of both instances
2. Put SW on
3. Patch one node, which automatically patches the other node
4. Start one instance in non-clustered mode (parameter cluster_database = false)
5. startup migrate (ie. underscore parameters are set)
6. run catpatch.sql
7. shutdown
8. startup

I'm sure it could have been done faster, but all this time the server/disk system was busy doing stuff it's supposed to do. There was no "waste" or "errors" during the upgrade. And it was done according to the Oracle documentation (which worked completely as described). It just took five hours.

Many people I talk to are surprised by this. But when you think about it, you don't patch nodes or instances. You patch databases. Oracle has one database in a RAC installation.

So there really is no such thing as 24/7/365 (or 25/8/370 which is about as realistic in the real world). There are HA options to the database *plus* scheduled downtimes for maintenance.

Oracle doesn't have rolling upgrades. Oracle doesn't have "online patching". So your database needs to be down while you upgrade or patch it.

Could you duplicate your database, eg with Oracle DataGuard, so your users could be running on one database while the other is being patched? I'm sure it's possible, but I can't see how since DataGuard requires you to be on the exact same patch level on both Oracle and the OS. So it would appear that you need to shutdown and patch both databases at the same time.

If it's supported to let DataGuard run while upgrading the primary database to a higher version or patch level, I'd be interested in the details.

Did you notice what was missing from the list of actions above? The client didn't take a backup before upgrading. For very good reasons Oracle recommends that you perform a full and valid backup before applying patches or upgrading your system. This client didn't, but you should. Oracle might even recommend (again, for very good and valid reasons) that you take a new backup when you're done with your upgrade/patch actions.

So there's RAC plus scheduled downtimes. But there are also emergency patches that need to be applied fast. This could be due to an error encounted in the environment or it could be a security patch. The time needed to apply emergency patches is hard to plan .

With RAC you get duplicate nodes, duplicate instances and one database. That database can be hit by dictionary corruptions (I'm sure we've all seen one) or it can be hit by the need for patching and upgrading. That's downtime for your whole RAC system.

Scalability

Scalability is of course much better with RAC than with OPS, and you don't need as many fancy tricks in order to make it scale well.

But. If you remove a bottleneck in any system (IT or other) a new bottleneck will now be present. It might be smaller than the old bottleneck (hopefully and usually, but not always), but it's still a bottleneck.

With RAC pinging is done using CPU resources. Yes, that's much faster than disk resources. But what if you are strapped for CPU in your system and RAC therefore cannot get enough CPU for the pinging?

Mario Broodbakker from Digital/Compaq/HP in Holland has done some interesting benchmarks on RAC that prove two things:

It's important to have enough CPU left for the RAC pinging activity.

And you can still get into situations where traditional OPS workarounds are needed (data partitioning, etc.) in order to achieve maximum performance. Even the wonderfully complex and mythic GC_FILES_TO_LOCKS parameter can be useful at times. It has deliberately been removed from the documentation because it was seen by Oracle Marketing to send the wrong message.

So you say: Of course you should have the necessary CPU available for the RAC pinging activities – hey, you should always size your system professionally. Yes. But what if you suddenly have a bunch of batch jobs or batch-like processes fighting over the available CPU resources (PX, DBMS_JOB, backup, file copying, whatever)?

Yes, you can plan for many situations, but sooner or later your system will be in a situation where the system is running at 100% CPU, and that's when you'll see some really bad performance with RAC.

If you're interested in Mario's whitepaper about his RAC testing let me know, and I'll be happy to send it to you. RAC Development are planning to address several issues he has pointed out or has already addressed them. Yet the lack of required CPU resources is not something Oracle or RAC currently can do anything about.

Manageability

OPS has actually never been that bad to manage. Assign an instance number to each instance, startup and shutdown the instances in the same order every time, create simple scripts for these things, and you're pretty much rolling.

It's now possible to do very clever things with groups of instances, and OEM has been greatly enhanced to handle RAC – but most customers will still run a two-node cluster with two Oracle instances on it and can do fine with the good, old features from the OPS days.

But it still requires more skills and more time to manage RAC than not. Added complexity means additional skills and additional time. You can of course define your way out of it by calling it "planned downtime" or "maintenance" or "service time" instead of plain downtime.

The end goal, though, is often to have the database (or rather the applications that depend on the database) available most of the time.

Skills Required

I have already touched on this in several places in the paper, so let's just repeat here that it's not only RAC skills that are needed, but also (and probably most) cluster skills if your organisation is going RAC.

There's one external Oracle RAC class (three days) and one so-called DSI (Data Server Internals) class for internal Oracle consumption available out there.

There are also RAC classes available from various external companies.

And there are lots of other people out there that know about OPS and RAC. Listen carefully to the bitter, twisted old men who've worked with OPS. When RAC is pushed to the limit, you could still need to do the same things that were required with OPS.

Troubleshooting

Ah yes, troubleshooting. I've seen many clusters that just froze for no apparent reason in my time. It's always possible to make the OS or Cluster software dump a trace/log file when it happens.

The resulting trace/log file from the cluster will normally be the size of Texas, and only one or two people in the entire vendor organisation can truly understand them, you will be told.

Then the files (often with sizes measured in GB) are shipped to the vendor and some months later they will report back that it wasn't possible to pinpoint the exact reason for the complete cluster freeze or crash, but that this parameter was probably a bit low and this parameter was probably a bit high.

That's what always happens. I have never – really: never – seen a vendor who could correctly diagnose and explain a hanging cluster or a cluster that kept crashing.

As to Oracle trouble shooting I'm not so worried. Oracle will either have a performance problem, which is easy to diagnose using the Wait Interface or you'll get ora-600 errors that are fairly easy to diagnose, although you'll need to spend the required 42 hours logging and maintaining an iTAR or SR or whatever the name is these days.

In other words: Finding out what's wrong (if anything) in Oracle is much easier than finding out what's wrong with a cluster.

Conclusion

If you have a system that needs to be up and running a few seconds after a crash, you probably need RAC.

If you cannot buy a big enough system to deliver the CPU power and or memory you crave, you probably need RAC.

If you need to cover your behind politically in your organisation, you can choose to buy clusters, Oracle, RAC and what have you, and then you can safely say: "We've bought the most expensive equipment known to man. It cannot possibly be our fault if something goes wrong or the system goes down".

Otherwise, you probably don't need RAC. Alternatives will usually be cheaper , easier to manage and quite sufficient.

Now please prove me wrong.

Mogens Nørgaard (mno@miracleas.dk) was with Oracle Support in Denmark for 10 years (three as an RDBMS analyst, four as head of RDBMS Support and three as head of Premium Services). He is co-founder and technical director of Miracle A/S (www.miracleas.dk), which provides consulting, support and training on Oracle and SQL Server, in Maaloev, Denmark.

First claim to fame: First manager within Oracle to demand that his team (about 40 people in Premium) used the YAPP performance diagnostics method created by Anjo Kolk. .

Second claim to fame: The OakTable Network (www.oaktable.net) was named after his dining table where some of the better Oracle scientists will gather a couple of times each year.

Mogens and his co-director Lasse (lch@miracleas.dk) will use the profits from Miracle A/S to start up a micro brewery that can stop Carlsberg from taking over the world. He believes Carlsberg is the Danish equivalent to the American Budweiser. If nothing else is available, though, he'll drink both.