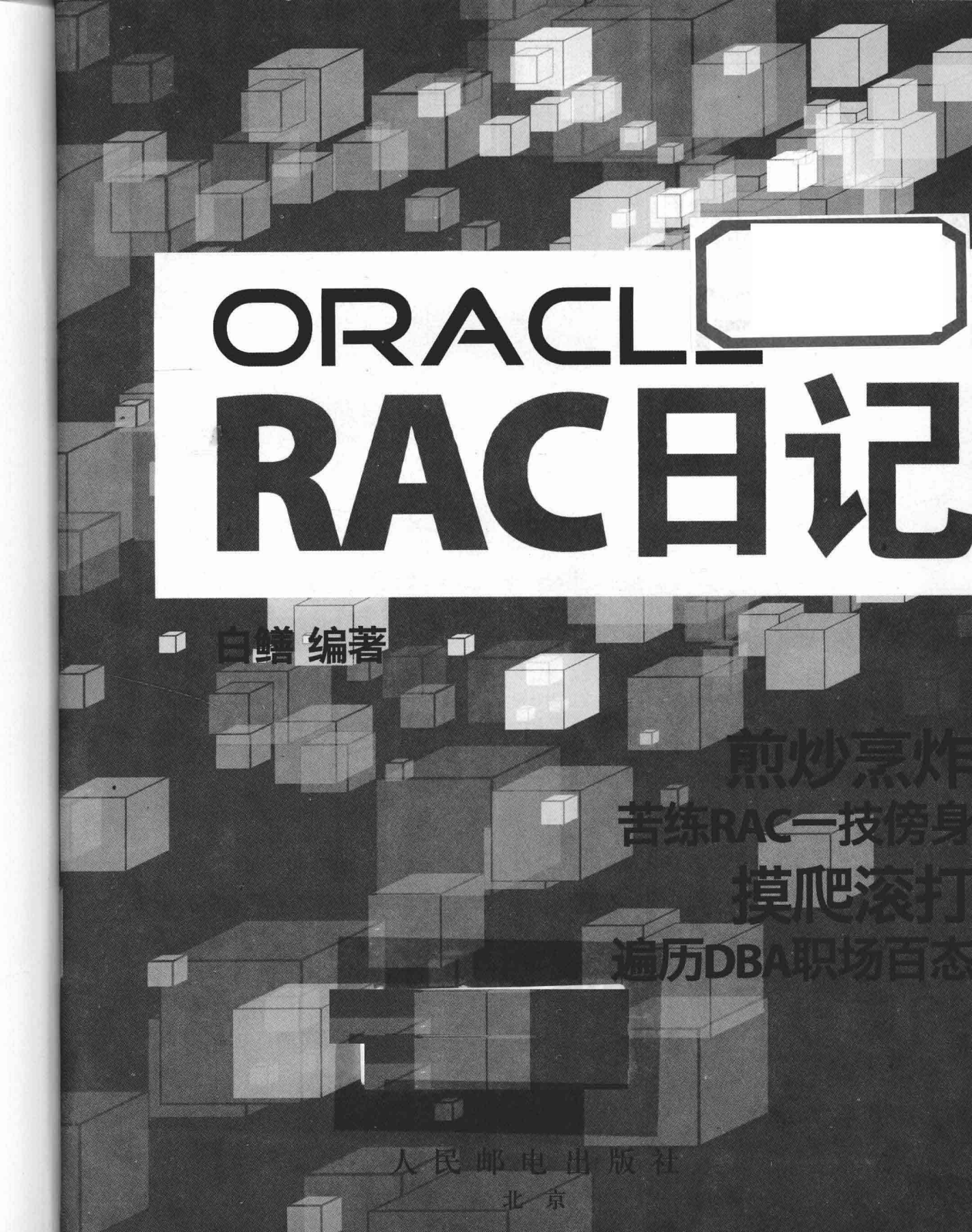
恩墨科技创始人
Oracle ACE Director
盖国强倾力推荐

ORACLE RAC日记

白鳝 编著

煎炒烹炸
苦练RAC一技傍身
摸爬滚打
遍历DBA职场百态

人民邮电出版社
POSTS & TELECOM PRESS


ORACLE
RAC日记
白鳝 编著
煎炒烹炸
苦练RAC一技傍身
摸爬滚打
遍历DBA职场百态
人民邮电出版社
北京

图书在版编目（CIP）数据

ORACLE RAC日记 / 白鳝编著. -- 北京 : 人民邮电出版社, 2010.5
ISBN 978-7-115-22422-4

Ⅰ. ①0… Ⅱ. ①白… Ⅲ. ①关系数据库－数据库管理系统, Oracle 10g Ⅳ. ①TP311.138

中国版本图书馆CIP数据核字(2010)第035963号

内 容 提 要

本书以 Oracle 10g 为基础，从基础知识、安装升级、故障处理、性能优化 4 个角度，由浅入深地介绍了 Oracle RAC 项目实施的一些方法和思路。

本书一共包括了 9 个综合案例，每一个案例构成独立的一章，按照“遇到问题→解决问题→案例总结”的思路进行展现，首先对现实问题进行描述和分析，然后提供合适的解决方案，最后自然地引出 Oracle 中的理论知识点，这种讲解方法能够有效地降低阅读难度，帮助读者更好地掌握相关技能。此外，在每个案例中，都再现了大量真实的工作情景，包括客户交流、人员沟通、寻求资源等，可以帮助读者更好地融入职场，掌握很多高效工作的技巧。

本书可以作为数据库开发人员、数据库管理员、数据库初学者及其他数据库从业人员的工作参考手册，也可以作为各大中专院校相关专业师生的参考用书和相关培训机构的培训教材。

ORACLE RAC 日记

◆ 编　　著　白　鳝
责任编辑　杜　洁
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：20.75
字数：506 千字　　2010 年 5 月第 1 版
印数：1－4 000 册　　2010 年 5 月北京第 1 次印刷

ISBN 978-7-115-22422-4

定价：48.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

最近这段时间我一直在考虑要写点什么。《DBA 日记》以博客的形式在 Oracle 粉丝网（http://www.oraclefans.cn/forum/index.jsp）上连载快一年了，在完成了沈阳那个优化项目后，我一直在考虑下面该写一个什么样的故事。最近在 Oracle 粉丝网和白鳝的洞穴群里讨论的比较多的是 RAC，很多网友在问我能不能写一些关于 RAC 方面的案例，因为 RAC 目前在国内越来越流行了，但是由于 RAC 在国内的使用水平并不高，因此在 RAC 方面的故障处理和性能优化一直是一个难题。另一方面，目前市面上关于 RAC 的技术书籍也相对较少，读者的选择余地不大。在这些年里我实施过的 RAC 项目也非常多了，从规划、安装、升级、故障分析到优化，在实施的过程中，我也得到了很多启示，我觉得很有必要把这些经验写出来，与大家分享。

我一直认为 RAC 的历史沿革是来自于 DEC 公司的 VAX CLUSTER，实际上 Oracle 并不是第一个支持 CLUSTER 技术的数据库，Oracle 公司的另外一个数据库产品 RDB 在 20 世纪 80 年代就已经支持 CLUSTER 技术了。在我的印象里 VAX/VMS 平台也是第一个真正支持 OPS 的平台（这个感觉也许并不准确），OPS 是并行服务器的简称，也就是我们现在耳熟能详的 RAC 前身。OPS 技术是在 Oracle 7.3 版本首次推出的，在那个年代，除了 OpenVMS，在 UNIX 平台上，除了 DEC 公司的 OSF/1 中支持的 DEC CLUSTER 外，IBM、SUN 和 HP 还没有推出自己真正的 CLUSTER，因此 7.3 的 OPS 在国内应用得十分稀少。

OPS 真正被大规模使用是 Oracle 8.0，Oracle 8.0 的 OPS 技术也基本上成型了，这个技术框架一直延续到互联网时代著名的版本 Oracle 8i。在 Oracle 销售的嘴中，OPS 被夸张成一种无限可扩展的架构，应用软件几乎不用做修改就可以在 OPS 架构中获得无限的可靠性和可扩展性。不过国内的很多用户使用 OPS 的经历是十分痛苦的。很多客户最终选择放弃 OPS，将原来的服务器拆分为单机。10 年前，我的一个客户在使用 Oracle 8.0.4 OPS 出现严重故障后，由于无法得到相关的技术支持，做出了放弃 Oracle 数据库的痛苦选择，虽然数年后这个客户又痛苦地放弃了 Sybase 重新投奔 Oracle 的怀抱，不过这个早期的吃螃蟹者的教训，至今还让人心有余悸。

RAC 的出现是一种革命，RAC 改进了 OPS 节点间数据变更代价太大的问题，当一个节点需要访问一个被另外节点修改过的数据，再也不需要让持有节点先写入硬盘，然后再从硬盘读取，而是可以从 INTERCONNECT 网络直接传输这个数据。RAC 的这种改进，也取决于网络技术的发展，如果没有百兆级、千兆级网络的出现，这种传输的性能是不可想象的。

虽然从历史传承上来看，RAC 是 OPS 的升级版本，不过从技术上来看，我们甚至可以认为 RAC 和 OPS 是两种不同的产品，Oracle 公司甚至将这个产品的名字也做了修改。这种名称上的修改，实际上在其他数据库厂商也宣布推出 CLUSTER 产品后，Oracle 为了表示和他们的区别而特意进行的商业炒作，REAL APPLICATION CLUSTER 中的 REAL 已经说明了一切，这和当年 DEC 公司将自己的 CLUSTER 产品命名为 TRU64 CLUSTER 有异曲同工之妙。在商业敏感度上，拉里永远是赢家。

虽然 RAC 很好地解决了 INSTANCE 之间共享 CACHE 的问题，但是这种跨实例的 CACHE 协同操作的成本还是十分高的，事实上 RAC 并不像 Oracle 销售所说的那样可以提供与应用几乎无关的无限可扩展性。基于 RAC 架构的应用，应该在底层架构上就进行针对性的设计，才能最大限度地发挥 RAC 可扩展性的特点。单实例环境的应用在升级到 RAC 环境时，的确是应该做一些有针对性的优化，否则升级到 RAC 后会由于 INTERCONNECT 带来很多负面的问题。本书将会通过几个案例来向大家介绍这方面的一些技术和方法，希望帮助大家掌握 RAC 在维护和优化方面的一些基本概念和原则。

老白的很多客户都使用了 RAC，而且也感觉到最近这几年采用 RAC 的系统也越来越多。其中有一些客户的应用底层架构做了针对 RAC 的设计，这样的系统今后随着 RAC 节点数的增加可以很平稳的扩展；有些客户为了减少 RAC INTERCONNECT 的开销，对 RAC 中的多个实例做了分工，使多个节点之间需要共同修改的数据减少到较少的水平，比如某些移动公司规定了某些地市的客户端连接到某个实例上，这样就可以通过分区表技术使不同分区被不同实例交叉修改的可能性降低到最小；还有一些客户为了避免 INTERCONNECT 带来的问题，甚至在两个实例上跑不同的应用，这些应用之间共享的数据基本上是只读的。这些都是为了解决 RAC 存在的问题采取的一些应用优化的措施，不过，这些措施大多都存在一定的问题，是一种权宜之计。

实际上如果从应用软件底层架构开始做好设计，RAC 系统确实是可以做到十分强大的可扩展性。我见过国外有超过 20 个节点的 Linux 系统，最初是从 2 个节点开始发展起来的，随着业务的增长，通过添加节点就可以实现扩展了。

由于 RAC 对于底层应用设计要求很高，所以很多用户至今不敢使用 RAC，随着 IT 技术的发展，目前的 IT 系统越来越庞大，使用 RAC 技术是一个趋势，因为越来越多的系统的容量增长都是呈几何级数的。在 10 年前，几十 GB 的数据库已经属于十分大型的数据库了，而现在几个 TB 的数据库比比皆是。目

前各行业的客户都在想找到一种不需要总是更换更大的服务器的解决方案，而RAC的可扩展性正是这些客户所需要的。

前阵子和一个工行的朋友谈到大型机的问题，IBM的大型机对于中国的金融企业来说，既是一个天使又是一个魔鬼。大型机造就了今天金融企业IT系统的成功，但是大型机也大量吞噬着金钱，其昂贵的维护费用让大型银行都感到有点心痛。不过除了大型机，还有什么解决方案能够作为替代呢？我当时提出RAC完全可以解决工行面临的问题，不过那些朋友对RAC都不以为然，他们认为RAC只是一个梦，其技术上的不成熟，导致银行不可能选择这个方案。事实是，在我见过的银行核心系统中，哪怕是某些中小型商业银行的核心交易系统，也没有见过RAC的系统，顶多是Oracle的HA系统。金融企业不使用RAC主要还是国内的金融解决方案供应商对RAC的认识，由于缺乏基于RAC的应用设计思路，从而导致RAC在金融行业的口碑不佳。

事实上，RAC的特点决定了只要解决多节点之间的CACHE FUSION问题，RAC对性能的负面影响也就不成为问题了，不过这个看来不是问题的问题，目前在国内还是一个大问题。国内的应用开发商往往注重于功能性的实现以及开发的难易程度，而往往忽略了系统底层架构的合理性，正是系统设计人员在底层架构设计上的缺陷，导致了RAC应用并不能从应用的角度去解决CACHE，而只能依赖于DBA的优化。

在这本书中，除了最后一个案例不是以RAC为主外，其他几个案例都是与RAC相关的。这些案例从3个不同的角度阐述了RAC系统设计、优化与故障处理的方方面面。由于RAC项目的实施周期一般都比较短，一般情况下几天时间一个项目，因此这本书是由一个一个不相干的小故事组成，老白会尽可能把一些相近技术内容的故事组织在一起，让读者能够在相对集中的时间里，对某个技术有更深入的认识。

本来我给这本书起的名字叫《与RAC相关》，不过总觉得这个名字有点过于直白，今天早上我坐地铁从北京的天坛到东直门，有一种很强烈的感觉：我在路上。在深圳的时候这种感觉不是很明显，从家里开车到公司也不过10分钟左右的车程，因此对于上班没有什么感觉，而在北京这种感觉特别的明显。实际上整个2009年我一直是在路上行走，偶尔停下来给客户做一些分析，解决一些问题。作为DBA，我们一直是在路上，而且这条路我希望还能多走几年。今天突然有个感觉，在路上是一个很好的名字，于是我想把这本书的名字暂时叫做《在路上》。不过真正的书名我想交给出版社来决定，无论这本书叫

做阿猫阿狗，总之这是一本关于 RAC 的书。另外就是，《DBA 日记》不是老白一个人的故事，欢迎大家对 DBA 日记提出好的建议，如果有好的案例，也可以发给老白（可以发送到老白的邮箱 xuji755@139.com，或者在论坛留言 http://www.oraclefans.cn/forum/listtopic.jsp?boardcode = DBA_D&CPages=1&threadType=0），这样就可以经过整理后放到 DBA 日记里。

本书的写作方式也有所不同，是由一个一个的小故事组成，故事之间的关联性并不强，所以可能没有《DBA 日记》那么多的悬念和有趣的故事情节，因此本书中取消了“每日点评”和“每日一技”。通过这样调整后，本书的每个小故事将更加连贯。为了保证读者在阅读的时候不会由于某些技术问题而感到困惑，本书在每个案例结束后将会有一个小记，在小记中讲解了一些技术要点，将正文中的一些没有交代清楚的技术细节深入展开，此外，还会对这个项目进行一定的总结，在这里大家可以看到对案例的点评和思考。

总体来说本书是一本介绍关于 RAC 优化与故障处理的书，和老白的其他书类似，在书中老白主要介绍的是方法，而不是技术。之前也有不少人希望我写一本关于 RAC 技术的书，我觉得 RAC 技术浩如烟海，以老白目前的写作能力，还没能写得十分完美。所以老白只能通过日记的形式，将 RAC 的一些相关技术和处理问题的方法融入在日记里提供给大家分享。

白　鳝

2010 年 4 月

老白学 RAC

老白接触 RAC 的时候 RAC 还叫 OPS，OPS 翻译成中文就是 Oracle 并行服务器，不过那个时候 DBA 还是喜欢把 OPS 叫做 Oracle CLUSTER。老白对 CLUSTER 的认识来自于 DEC 公司的 VAX CLUSTER，VAX CLUSTER 是一种十分优秀的集群系统，在集群里的任何一个资源（文件、磁盘、端口、设备等）只要被设置为 CLUSTER 资源，那么这个资源就可以在整个 CLUSTER 中共享。VAX CLUSTER 这种古老但是十分优秀的技术现在已经基本上快消亡了，不过 VAX CLUSTER 技术给了老白很多误导，认为 CLUSTER 就应该是这样的，不过随着 DEC 的灰飞烟灭，我们看到 20 年后的今天，UNIX 上的 CLUSTER 还只能做到 VAX CLUSTER 的一小部分功能。

15 年前，老白开始使用 Oracle 数据库的时候，绝大多数客户对 CLUSTER 的认识还停留在 HA 层面上，当时最常用的词汇是双机热备和双机冷备这个概念，这两种技术都被宣传为 CLUSTER 技术，所不同的是双机热备就是我们现在所认知的 RAC，而双机冷备就是我们现在常说的 HA 系统。早期老白接触 CLUSTER 系统时，实际上客户对 Oracle CLUSTER 的要求只是基于热备和冷备的需求，而不是来自于性能和可扩展性方面的需求。那时候的数据库往往较小，一般在几百 MB 到 10 多个 GB，而且应用系统也比较简单，因此系统在性能上的问题较少，DBA 的主要工作就是帮助客户安装 Oracle 系统，当然 OPS 的安装是这中间技术含量最高的了。

早期的 OPS 系统安装十分复杂，需要在操作系统上打大量的补丁，进行复杂的配置，因此 OPS 系统的安装一般会选择购买原厂的服务，Oracle 的销售一般会把 OPS 的安装说成是一个和阿波罗登月一样复杂的工作，因此敢于不购买原厂安装服务的客户少之又少。说实在的，不要说是 OPS 系统，早期的 Oracle 系统安装也十分复杂，老白曾经帮客户安装过 Oracle 5.1 FOR OPENVMS，介质存储在 266MB 的磁带上，从磁带上将安装介质导出来，到编译链接，再到创建数据文件和表空间，整个过程的复杂程度确实和火箭发射没多大区别了。

不过从 Oracle 6 开始，Oracle 的安装变得简单起来了。有一次老白给一个客户安装一套 SCO UNIX 5.0 上的 Oracle 7，当客户看到老白拿出厚厚的两盒子安装盘的时候，不禁生出敬畏之心，于是他搬了张椅子坐在老白的身边，想好好学习一下传说中的复杂工艺，开始的一些眼花缭乱的字符界面和配置确实让他感到有些神秘，他不停地问老白每个配置选项的含义，几分钟的配置很快就过去了，剩下的就是好像永无止尽的换盘。对于一个几十 MB 的“大型”软

件来说，1.44MB 软盘的容量太小了。等老白换到第十几张盘的时候，那哥们儿终于打着哈欠离开了，临走的时候还说了一句："老白，原来你们 DBA 干的就是这种没有什么技术含量的活啊，下回有客户安装 Oracle 我去帮你干吧。"

虽然说 Oracle 的安装变得简单了，不过 OPS 的安装还是一个在业界认为是技术含量最高的活，甚至有些 Oracle 的工程师都十分惧怕安装 OPS，我就曾经见过一个客户到了客户那里拼命地说 OPS 的坏话，并且建议客户把系统做成 HA 模式。最后在客户的一再坚持下，他硬着头皮开始安装 OPS，在折腾了两天后，他不得不承认这是自己第一次安装 OPS，估计是搞不定了，等过两天公司派个高手再过来安装。

看到传说中的 OPS 居然是如此得难缠，于是老白也在工作中尽可能地避开 OPS，好在那时候的客户对系统的高可用性和高可扩展性方面的需求并不明显，因此在最初的几年里，老白都能很轻松地说服客户不要使用 OPS。不过该来的事情总是要来的，1998 年经历了那次 8.0.4 OPS 大型故障后，虽然老白对 OPS 更加排斥，但是在一个项目中，老白终于无法躲开 OPS 了。

当时我们承接一个比较大型的软件项目，这个项目是从国家火炬计划的经费下拨的重点项目，因此该项目在前期方案认证的时候请了一大批清华大学的专家，于是高可用性、高可扩展性之类的需求就放在了十分重要的位置。再加上数据库选择了使用 Oracle，那么上 OPS 就是必然的事情了。更让老白头疼的是因为在经费预算上的不足，最终这个项目确定的时候连集成费和购买 Oracle 原厂安装服务的钱都没有了，这就意味着从小型机的安装、存储的安装、数据库的安装到应用软件开发这一条龙的工作都必须由我们这家专门从事软件开发的公司来承担了。当甲方的科技处处长一脸坏笑的对老白说"要么你们全接下来，要么我们就另请高明"时，老白也只能苦笑着拍胸脯了——这百把万的单子够老白全公司吃一年的了，就是砸锅卖铁也必须接下来。

胸脯好拍，事情难做。为了做好这个项目，合同还没签订，老白就已经开始了 OPS 技术的学习。从 oracle.com 上老白下载了一套 Oracle 8.1.6 的技术资料，其中一个《Oracle 8i Parallel Server Documentation Online Roadmap》文档介绍了一个学习 OPS 的路线图。首先需要学习第三方厂商的 CLUSTERWARE 的相关技术，这方面还算好，老白好歹还是在 DEC 混过一段时间的，早就从原厂找了一个 ALPHA 产品的高手来帮助安装系统，这部分内容就先省了吧。第二步就是学习《Oracle 8i Parallel Server Concepts》，这是一本介绍 OPS 概念的书，从概念入门，循序渐进的学习理念和老白一贯的学习方法是一致的，老白

决定先用个把星期时间认真研读一下这本书。第三步是阅读《Oracle 8i Parallel Server Setup and Configuration Guide》，这是一本介绍安装配置的书，真正实战安装就要靠这本书来指导了，这本书不一定需要面面俱到的阅读，不过主要的脉络是一定要认真看看的。第四步是阅读《Oracle 8i Parallel Server Administration，Deployment，and Performance》，这本书和我们今后的应用开发、部署和性能优化有关，是十分重要的。不过在目前阶段我们还不会碰到这方面的问题，可以稍微往后放放，不过关于性能优化的内容还是要尽早阅读，最好能在我们数据字典设计完成之前把 OPS 性能方面要考虑的问题搞清楚。

阅读《Oracle 8i Parallel Server Concepts》这本书对老白的帮助十分大，在阅读这本书之前，老白根本不知道 OPS 是怎么工作的，只是知道 OPS 可以支持多个实例同时访问一个数据库。而这本书开场的一段文字给了老白十分深刻的印象，就是这句话为老白认识 OPS 打开了一扇门：

> ***You can also use Oracle Parallel Server to deliver high performance, throughput, and high availability. Whatever your goal, your challenge is to successfully deploy these technologies to take full advantage of their multiprocessing powers. To do this, you must understand how Oracle Parallel Server works, what resources it requires, and how to effectively use it.***

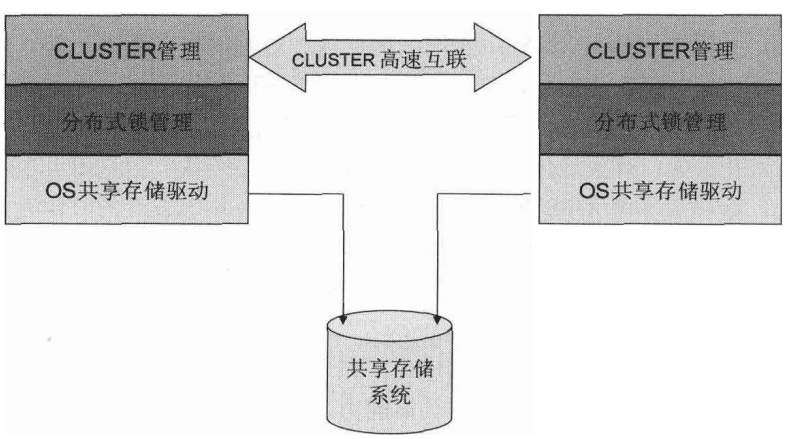
在看到这段文字之前，老白认为 OPS 可以为开发人员解决一切问题，开发人员只需要将应用部署在 OPS 上，就可以自然而然的获得高可用性、高可扩展性，并且可以随着 OPS 中实例的增加获得更强的处理能力。而这段文字中明确指出了，无论你的目的是怎么样的，你必须面临如何成功地部署应用的挑战，要做到这一点，你必须了解 OPS 是如何工作的，OPS 需要获得什么样的资源，并且如何有效地使用这些资源。看样子我必须认真把这本近 300 页的书看完，起码要了解 OPS 的基本工作原理，并且在系统设计的时候充分利用 OPS 的优势，避开 OPS 的弱点，才能充分发挥 OPS 的优势。

Oracle OPS 概念的第一章里就介绍了 OPS 的几个重要的特点，包括可扩展性、高可用性、透明性（Transparency）。透明可扩展性是 Oracle 销售经常对客户宣传的，就是一个单机环境的应用，只要部署到 OPS 上，就可以实现透明的可扩展性，获得几乎无限的可扩展能力，但是我看到 Transparency 的概念并不是这样的，根据 Oracle OPS 概念的描述：

Transparency is the functional equivalent of single-instance exclusive Oracle and shared configurations that use Oracle Parallel Server. Applications that run on single instance Oracle execute with the same results using Oracle Parallel Server.

这里对 Transparency 的描述只是说在单实例环境下和并行服务器模式下的执行结果是相同的，而并没有说更多的内容。带着这个疑问，我继续阅读后面的内容，不过在后面 High Performance Features of Oracle Parallel Server 中仅仅介绍了在 OPS 下 BUFFER CACHE 的管理和 FAST COMMIT 等在 RAC 环境下的基本原理外，并没有再提到关于透明的可扩展性这个问题。看样子传说中的 Oracle 高可扩展性只是一个镜花水月的东西，还是有一定的原因的，也许 OPS 真的不像 Oracle 销售所宣传得那么美好。

第 2 章介绍了 OPS 硬件结构，这一章虽然很短，不到 10 页的篇幅，不过通过对这一章的阅读，老白这个对 OPS 只知道一些皮毛的人，了解了大量的概念。首先是节点的概念，知道了 OPS 概念下的节点的 4 大要素：CPU、内存、存储和互联网络。从而了解到一个 CLUSTER 的性能取决于 CLUSTER 内的节点和节点互联的性能，比如内存、CPU、存储的能力，以及内存到内存通信的性能和节点间通信的性能。通过这一章，老白明白了一个 OPS 系统的基本硬件结构，就是多个独立的服务器节点，通过 INTERCONNECT 网络互联，并且通过某种方式共享存储。了解了 OPS 的硬件架构，很容易就可以明白后面关于 OPS 软件架构的描绘，如下图所示。



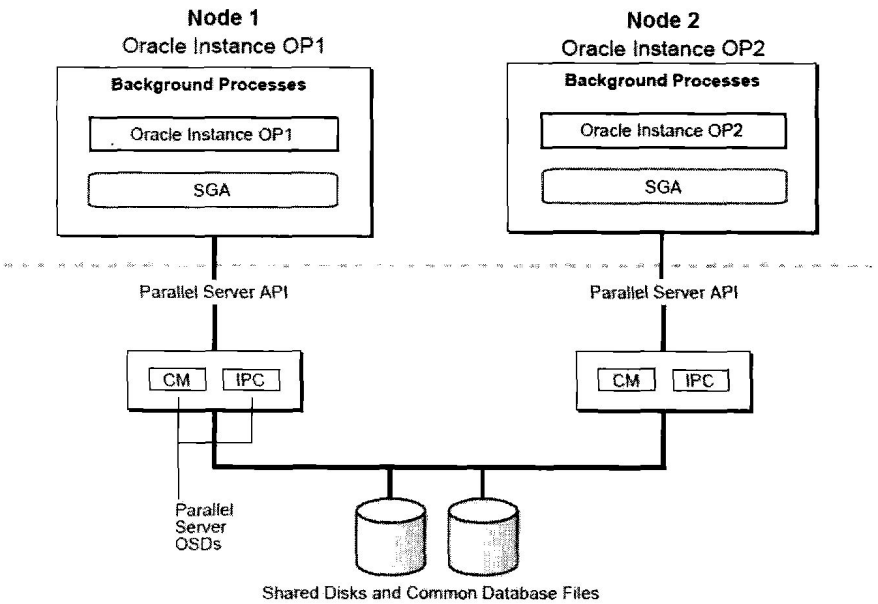
CLUSTER 管理实现多个节点之间的协同操作，包括集群的成员管理等一系列功能。这部分功能一般来说是操作系统厂家提供的（从 10g 开始 CRS 取代了大部分第三方集群软件的功能，甚至能实现完全替代，在 10g 上，可以不使用任何第三方 CLUSTERWARE 运行 RAC 数据库）。CLUSTER 管理包括了故障侦测和节点监控的功能。

分布式锁管理机制实现多个节点之间资源的协同访问，实现数据的一致性访问。而最底层的共享存储系统是实现 OPS 多个实例访问同一个数据库的底层基础。

另外一个不容忽视的组件就是 CLUSTER 节点间的高速互联，CLUSTER 之间的 IPC 通信机制是实现 OPS 的消息机制的基础，该组件实现一个高速的、异步的、基于消息队列的消息传递机制，目的是通过硬件的支持，最快速地实现消息的可靠传输，尽可能快地将一个消息在节点之间传递。

这本书的第一部分虽然只有不到 40 页的内容，不过通过对这部分的阅读，老白对 OPS 的基本原理有了一个概括性的认识，以老白近 10 年的软件开发与应用经验，大体上也对 OPS 的主要特点、优势及弱点有了初步的了解。老白感觉到节点间的高速通信和分布式锁管理将会成为决定 OPS 上应用系统性能的关键。

阅读了《Oracle 8i Parallel Server Concepts》，老白对 OPS 的基本概念有了一个初步的认识，OPS 在老白眼里也就没有那么神秘了。虽然有了很多 OPS 方面的理论知识，老白对于如何安装 OPS 数据库还是一无所知。在网上用 Google 搜索了一下也没有发现什么有价值的资料，于是老白只能静下心来，认真阅读《Oracle8i Parallel Server Setup and Configuration Guide》，这又是一本近 300 页的天书。这本安装指南的开头部分也介绍了 ORACLE OPS 的一些基本概念，重点重申了节点（NODE）、CLUSTER 和 DATABASE 3 个概念。NODE 是容纳 Oracle 实例的载体；CLUSTER 是一组共享同一个存储的互相连接的节点；而数据库是 CLUSTER 内所有实例共享数据文件的集合。实际上这 3 个基本概念构成了 Oracle CLUSTER 的核心。在一个 OPS 集群里，所有的实例共享相同的数据文件和控制文件，并且拥有自己独立的 REDO LOG 组。如下图所示可以很好地诠释这个概念。



安装 Oracle OPS 必须依赖第三方厂家提供的 CLUSTERWARE 和专门的组件（对于 RAC 来说，9i RAC 和 OPS 类似，需要第三方厂家提供 CLUSTERWARE 和专门的 RAC 组件，Oracle 10g 可以不依赖第三方的 CLUSTERWARE，并且不需要第三方提供专门的组件，只要使用 CRS 就可以实现上述功能），Oracle 的 CLUSTER 是建立在第三方的 CLUSTERWARE 的基础上的，通过叫做“Operating System Dependent Layer”的组件来实现 CLUSTER 的管理。“Operating System Dependent Layer”（OSD）包括 Cluster Manager（CM）和 IPC 两个模块，CM 是负责管理集群的成员的，可以监控每个成员的状态，并且在成员状态发生变化的时候通知其他组件进行相应的处理。IPC 是实现 CLUSTER 的节点之间消息通信的载体。

由于 Oracle CLUSTER 的这种体系结构，因此在安装 Oracle 时必须首先安装、配置操作系统和 CLUSTER 软件，然后安装 OSD 软件，最后再安装 Oracle 数据库。虽然这本手册对如何安装 OPS 描述得十分详细，不过对于老白这样一个刚刚接触 Oracle CLUSTER 的新手来说，这本书的有些地方描述得过于简单了，特别是关于 OSD 方面的知识，几乎是一带而过。不过这本书中关于 OPS 配置方面的描述还是相当精彩，让老白受益匪浅。

虽然老白对 OSD 方面的了解还是比较粗浅，不过好在老白在 DEC 公司工

作过，很快就找到一个原来的同事愿意协助安装 CLUSTER 和 OSD 组件。于是老白的第一次数据库安装变得十分的轻松，当 DEC 的朋友搞定了 CLUSTER 和 OSD 后，Oracle 8i OPS 的安装及数据库建库就和普通数据库没有多大区别了。Oracle 在 CLUSTER 的安装方面确实做到了像他们宣传得那么轻松。

虽然在 DEC 朋友的帮助下，老白在安装 Oracle OPS 时并没有遇到太多的麻烦，不过在随后的应用中，老白发现确实像 Oracle OPS Concepts 中所说的那样，要想发挥 OPS 系统的优势，必须针对 OPS 进行相应的设计和优化，而当时的应用软件并没有针对 OPS 进行设计，因此软件上线后，一直存在性能不稳定的现象。我建议客户的 OPS 系统中，平时所有的连接都连到 1 号节点上，当 1 号节点出现故障的时候再切换到 2 号节点上。客户觉得自己很亏，坚决不同意我的建议。确实也是，花了不少钱买了 OPS，只能享受到 HA 的待遇，这对谁来说都是不好接受的。

情急之下，老白想起了 OPS 提供的一个功能就是设置主从节点的模式，通过设置 ACTIVE_INSTANCE_COUNT=1，限制两节点的 CLUSTER 中只有一个节点平时接受客户端的连接，另外一个节点只接受 FAILOVER 的连接。这种模式和 HA 还是有着本质的区别的，HA 是双机冷备的模式，只有当主节点宕机后才会将数据库切到备用节点，并重新启动数据库。HA 的这种切换起码需要 10 分钟左右，而目前这种方式我们可以称为是双机热备，因为一旦主节点出现故障，会话马上就会自动切换到第二个实例，而且切换工作是透明的。

双机冷备和双机热备的区别还是比较容易理解的，当客户听了我的解释后，也只能哭笑不得的接受了这个结果，以目前系统的负载，单个节点处理起来是绰绰有余的，采用了我的方案后，系统也变得稳定起来。于是老白的第一个 Oracle CLUSTER 的项目就这样完成了。实际上这是一次不算成功的尝试，从这次尝试中，老白充分地认识到了 CLUSTER 环境下应用开发和优化的关键是如何避免过多的节点间的数据争用。

随着时间的推移，CLUSTER 越来越流行，Oracle 公司也推出了革命性的产品 Oracle 9i。在 Oracle 9i 中，OPS 被更名为 RAC，RAC 从名字上看就足够威猛，真正的应用集群，其主要目的是要区分于其他数据库提供的也号称 CLUSTER 的产品。

在这些年里，老白接触了大量的 RAC 应用案例，包括安装、设计、规划、故障处理、性能优化等方方面面。每一次接触 RAC 的应用案例，都有很强的

新鲜感，每次都会遇到新的问题，在解决这些问题的过程中，都会有很大的收获，最终都能够加深老白对 RAC 的认识。老白也更加真切地认识到，学习 RAC 必须在工作中学习才能认识得更为深刻，光凭看几本书是远远不够的。几本好书只能让你掌握 RAC 的一些基本概念，但是 RAC 的精髓，确实只有在工作中不断摸索，才能够真正掌握。

学习 Oracle RAC 的几点建议

学习 Oracle 需要理论结合实践，而 DBA 在学习成长的过程中往往会走两个极端，一个极端是拼命看书，而忽视了实践；另外一个极端是不看书，只实践。这两种 DBA 都会成长，也可能可以成才，但是都很难成为真正的高手。

RAC 也是实践性很强的，需要 DBA 在实际的生产环境中不断的磨炼，才能体会到 RAC 系统管理和优化的精髓。而事实上，DBA 能够接触 RAC 环境的机会十分少，这也导致了大部分 DBA 对 RAC 的认知仅仅在于安装这个层面上，甚至很多 DBA 认为学 RAC 就是学习如何安装 RAC。其实学习安装 RAC 只是学习 RAC 的第一步，也是最为简单的步骤，虽然 RAC 在不同的服务器平台上的安装会有所不同，但是其精髓部分是完全一样的，如果你能够理解 RAC 的概念和基本原理，这些不同点完全不会成为问题。学习安装 RAC 最简单的办法是通过 VMWARE，大多数 DBA 很难拥有两台小型机加一个共享存储的实验平台，VMWARE 解决了这个问题，通过 VMWARE 我们可以很方便地在一台笔记本电脑上完成一套 RAC 环境的搭建工作。

不过在做这些之前，老白还是建议大家好好去阅读一下那本领着老白入门的书籍《Oracle 8i Parallel Server Concepts》（大家可以到粉丝网上去下载这本书，http://www.oraclefans.cn/forum/showtopic.jsp?rootid=15202）。通过这本书理解 Oracle RAC 的一些基本概念，可以为今后深入学习 RAC 打下很坚实的理论基础。目前新版本的数据库文档中，这份文档已经不再单独发行了，RAC 的概念融入了《Oracle Concepts》中，通过对相关章节的阅读来更新知识。

实际上大家只要理解 RAC 就是多个实例同时访问一个数据库，其最底层的基础就是一组服务器共享一个存储系统。CLUSTERWARE 的主要作用是提供和管理共享的存储资源，以及负责节点的监控、管理和资源的重组。无论是第三方的 CLUSTERWARE 还是 10g 以后 Oracle 自己提供的 CRS，都只是提供一个 RAC 数据库运行的平台和容器。RAC 数据库是运行在这个平台和容器上的。实际上很多 DBA 在折腾的 RAC 安装，主要是在折腾这部分内容。而 RAC 技术方面最核心、最有价值的部分是 RAC 数据库的管理和应用优化。

CLUSTERWARE 提供了一个 RAC 底层的平台和运行容器，Oracle RAC 数据库就是运行在这个平台上，RAC 的数据库和单实例的数据库在本质上并没有多大的区别，数据文件、控制文件都是共享的，不过每个实例有自己独立的 REDO LOG 组，对于自动 UNDO 管理模式，每个实例都拥有独立的 UNDO 表空间。在数据文件和控制文件的格式上，RAC 系统和单实例系统并没有本质的区别，因此单实例系统可以很方便地升级为 RAC 系统。基于这种一致性，

在RAC环境下创建数据库实际上和在单实例环境下并无本质的区别。10g RAC在RDBMS安装和建库方面比9i麻烦，主要还是在于DBCA不仅仅要创建数据库，而且要在CRS中注册实例和各种APPLICATION，这些工作都是可以通过srvctl命令手工完成的，即使是这些注册工作没有成功，实际上数据库的正常运行是不受影响的。只是有些应用没有正常注册，可能会导致部分与RAC相关的新功能的使用受到影响。比如ONS注册不成功，或者ONS无法正常启动，那么就可能导致FAN等高可用性的功能受到影响，不过RAC数据库的常规功能是不受影响的。

刚才老白也说过，安装RAC只是DBA学习RAC走出的第一步，实际上更多的挑战来自于今后的维护和优化工作。每个DBA可能会碰到不同的问题，面临不同的挑战。在国内目前的开发水平下，很少公司能够在底层架构开始针对RAC进行设计，因此大多数DBA维护的RAC系统，都是通过应用分区来避免CACHE FUSION带来的负面影响。在一般情况下应用之间访问不同的数据，可能出现冲突的情况较少。作为DBA，必须了解应用分区的一些细节，特别是系统中的一些访问量比较大的核心业务表，他们在哪个实例上操作必须要了解清楚。在DBA做日常维护的时候，也尽可能不要打破这种关系，比如A表的访问都是在INSTANCE 1上访问的，那么当DBA需要对A表进行维护的时候（比如做一个大的删除清理工作或者UPDATE一批数据），也尽可能在INSTANCE 1上做，而不要为了减轻INSTANCE 1的负担而把操作放到INSTANCE 2上。

实际上为什么要这么做的原因很简单，主要是避免CACHE FUSION带来的负面影响。实际上RAC的维护和优化也并不是很复杂的事情，只要了解了RAC的CACHE FUSION的基本原理，在维护工作中围绕这个核心问题，就不会有太大的问题。本书中，老白会通过几个案例来从多个方面介绍维护、优化的要点，希望可以帮助大家打消对RAC的恐惧。