IBM

Information Management software

IBM DB2 pureScale

实现透明的应用扩展 技术手册


DB2

目录

简介

在经济复苏的过程中，对核心业务数据的即时访问始终是企业赖以生存，乃至获得成功的关键因素。随着越来越多的美元流入国内市场，企业需要借助具备高可用性和灵活的架构来提高业务的敏捷性，以便能够抓住新的发展机会。

大多数分布式软件公司在营销时都将可用性水平与“类大型机”或“5-9”可用性这样的术语联系在一起。这些短语都在试图传达业界公认的高可用性“黄金”标准(即 DB2® for z/OS®)所设定的持续可用性目标。

可用性	每年宕机时间
99.999%	5 minutes
99.99%	50 minutes
99.9%	8 hours, 20 minutes
99%	3 day, 11 hours, 18 minutes
95%	18 days, 6 hours
90%	34 days, 17 hours, 17 minutes
85%	54 days, 18 hours

如今，可用性并不仅仅意味着能够从容应对组件故障并恢复正常的事务处理。如果您的服务水平协议(SLA)指定预期的查询响应时间应在数秒之内，而服务器却花费了1分钟才返回查询，那么这就是可用性方面的问题。要确保可用性，您的系统不仅需要提供事务服务，还需要在SLA指定的期限内提供服务。

举例来说，如果业务周期中的季节性波动造成了扩展方面的可用性问题，则真正具备可用性的架构必须能透明地增加资源，同时避免更改应用，以满足不断变化的性能需求。透明性是一个关键因素：在提高产能时，不应该让应用具备集群感知性(应用知道哪些数据在哪个节点上，以避免节点之间的争用)。企业无法投入足够的资金来开发这些复杂的应用，因此无法实现

合理的扩展。这是为什么呢？首先，显而易见的是，集群感知的应用需要适应数据量和分布状态的不断变化。集群感知的应用并不仅仅要求代码随着集群的发展而改变：这些代码还需要经历测试、质量保证(Q/A)、部署和认证等过程。这可能造成企业花费数周时间来进行协调，并且不可避免地会耗尽基础设施中本应该有更好用途的资源。

用于在分布式平台(非大型机)上扩展数据库事务的其他产品通常采用过时的架构，因此会为扩展带来不必要的困难(比如说增加开销)，从而无法确保符合SLA协议。

IBM DB2 pureScale技术(以下称为DB2 pureScale)可以将高可用性与真正的透明应用扩展结合在一个系统中，以便满足您当前和未来对持续可用性的需求。IBM® Power™ Systems服务器和IBM存储解决方案的整合是DB2 pureScale架构交付这种高价值解决方案的内在基础。

到目前为止，“类大型机”仍然是一个引人注目的市场营销词汇。DB2 pureScale标志着真正透明的扩展架构首次应用于分布式平台。本文将介绍DB2 pureScale技术的基本概念、背景信息，以及它在高可用性和透明应用扩展方面具备独特优势的奥秘。

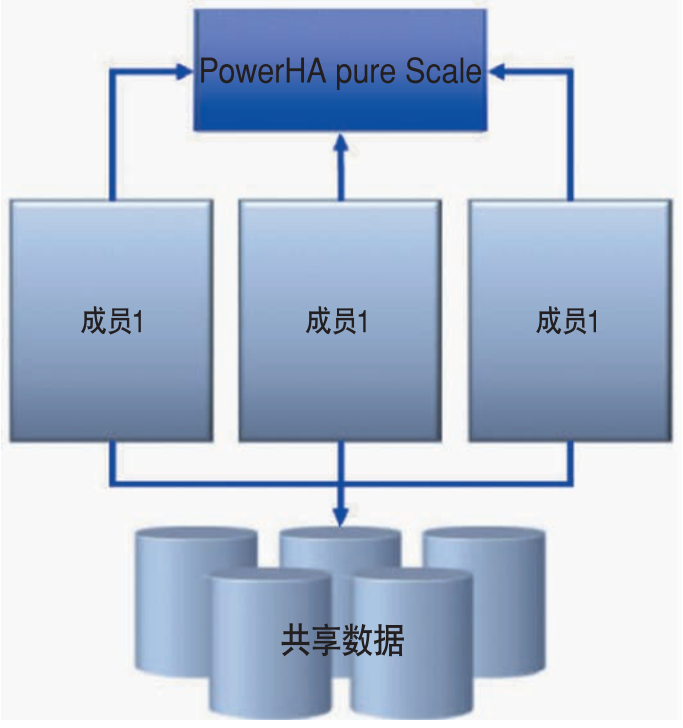
DB2 pureScale基本信息

DB2 pureScale是一种新的DB2 V9.8可选特性，它允许您通过“双机(active-active)”配置将数据库扩展到一组服务器上，以便交付高水平的可用性和可伸缩性。在这种配置中，运行于各主机(或服务器)上的DB2副本可以同时读取和写入相同的数据。

共享DB2数据的一台或多台DB2服务器被称作数据共享组。数据共享组中的DB2服务器是该组的成员。目前，数据共享组支持的最大成员数量是128。

除了DB2成员外，PowerHA pureScale™组件还提供了整合的锁管理以及针对数据页面的全局缓存(称作分组缓冲池)。

数据共享组中的各成员可以通过一个非常有效的InfiniBand™网络直接与PowerHA pureScale组件交互，如下图所示。这意味着各成员与集中化的锁和缓存设备之间建立了点到点(P2P)连接。



DB2 pureScale的起源

您所听到或看到的任何关于大型机可用性的描述均指的是DB2 for z/OS设定的高可用性黄金标准。事实上，世界上还没有任何一款数据库解决方案能在可用性方面与运行DB2 for z/OS的System z®服务器相提并论。

DB2 for z/OS数据共享所采用的底层技术确保服务器持续满足SLA需求，因为Coupling Facility提供了集中化的锁管理和全局缓存，这为快速从故障中恢复提供了保障。事实上，DB2 for z/OS从严格意义上讲已经实现了“5-9s”级的可用性，同时在无缝线性扩展工作负载方面享有很高的声望。

说起DB2 for z/OS，很多人都会想到广泛的可伸缩性和极高的可用性。这种市场声誉并非空穴来风，而是源于这些系统在数据库工作负载可用性方面的市场领先地位始终无人撼动。或许，最能佐证DB2 for z/OS强大功能的莫过于Oracle创始人兼CEO Larry Ellison的评论[1]：

我取笑过其他许多数据库，但唯独对大型机版本的DB2抱有尊重之心。它是当之无愧的一流技术。

DB2 for z/OS究竟有何独特之处，让Ellison对它如此赞赏有加？DB2 for z/OS在数据共享领域中的“独门秘笈”对其用户来说再熟悉不过了，那就是众所周知的Coupling Facility。Coupling Facility不仅为DB2 for z/OS赋予了线性扩展的能力，还提供了一个集中化设备来管理锁。除此之外，它还充当脏页(dirty page)的全局共享缓冲池(有助于可伸缩性和可恢复性操作)。

Coupling Facility技术为DB2 Z/OS贴上了可用性和可伸缩性方面的“黄金”标准的标签，DB2 PureScale技术秉承了DB2 for z/OS Coupling Facility的血脉。这是如何做到的呢？DB2 pureScale提供了一个IBM powerHA pureScale组件，该组件提供了同样集中化的锁管理和真正的全局共享缓冲池架构。

其他供应商实现了采用共享磁盘架构的数据库，其中最有影响力的是Oracle Real Application Clusters (Oracle RAC)。但是，当时在开发和设计Oracle RAC时，分布式平台技术还不允许有效地访问**集中**共享缓存。结果，Oracle RAC的设计最终成为了一次模拟DB2 for z/OS的一次尝试；这也是Oracle RAC的**分布式**锁管理技术和**分布式**缓存架构的起源。Oracle RAC在引入横向扩展的共享磁盘架构之后失去了DB2 for z/OS解决方案的简洁性优势。另一方面，DB2 for z/OS和DB2 pureScale提供了相同的**集中化**资源管理，因此解决了这些复杂的可伸缩性和可用性问题。本文将在稍后讨论这方面的内容。

起初市场上只有一种架构交付了真正透明的应用可伸缩性和高可用性。随着现代硬件在分布式平台上实现了互连，以及基于InfiniBand的无中断Remote Direct Memory Access (RDMA)的深入发展，DB2 for z/OS所采用的集中锁和缓冲缓存算法已经不再是它所独享的专利。DB2 pureScale将这项久经行业考验的技术引入到了分布式平台中，而这也代表了整个IBM家族的进步。

DB2 pureScale
实现透明的应用可伸缩性

在横向扩展的数据库环境中节省成本的关键是实现真正透明的应用扩展机制。透明的扩展意味着数据库引擎可以为OLTP应用提供更大的吞吐量和更快的响应速度，而对数据本地性没有要求。

数据的本地性表示应用所需的数据位于它所连接的服务器上，并且节点之间很少会争用相同的数据页面。在横向扩展架构中，如果采用基于网络的消息架构共享集群中的数据，数据的本地性就显得格外重要。

依靠数据本地性实现有效扩展的横向扩展架构要求开发人员创建复杂的事务应用来实现**集群感知性**。集群感知的应用在开发和部署方面不仅更加复杂，而且成本更加高昂，同时当集群发生更改时还要求重新设计应用。一些供应商可能声称它们的架构能运行任何应用，而不需要修改；但是，如果在设计时没有实现某种形式的集群感知性，它们将不能**扩展**任何应用。

透明的应用扩展意味着应用不需要具备集群感知性便可利用横向扩展架构。DB2 pureScale是分布式平台上所特有的，其高效性源于对现代网络和硬件架构，以及pureScale的集中化锁和缓存机制的利用。

为了减少集群中各节点之间的通信，以便实现锁管理和全局缓存服务，DB2 pureScale使用powerHA pureScale集群加速设备(以下简称为CF)和RDMA技术来提供透明的应用可伸缩性。

RDMA允许集群中的各个成员直接访问CF中的内存，而CF也可以直接访问各成员的内存。举例来说，假定集群中的某成员(成员1)希望读取未存储在本地缓冲池中的数据页面。DB2会分配一个代理(或线程)来执行此事务；然后，代理使用RDMA直接向CF的内存写入数据，声称自己需要读取某个特定页面。如果成员1希望读取的页面位于CF的全局集中缓冲池中，则CF会将该页面直接推送到成员1的内存中，而不是让该成员的代理执行I/O操作从磁盘读取它。通过使用RDMA，成员1的代理只需向远程服务器发起一个 memcopy (内存复制)调用，从而避免了成本较高的进程间通信、处

理器中断、IP栈调用等。简单来说，pureScale允许成员的代理通过执行看似本地的内存复制操作来执行远程内存复制操作。

这些轻量级的远程内存调用，连同集中缓冲池和锁管理设备，意味着应用不需要连接到已经包含数据的成员。集群中的任何成员都可以有效地从全局缓冲池接收数据页面，无论集群有多大。大多数RDMA调用都非常迅速，这使得发起调用的DB2进程在等待CF的响应时不需要让出已分配的CPU时间，并且不需要重新调度便可完成任务。举例来说，为了向CF通知某行即将更新(因此需要一个X锁)，某个成员的代理需要执行Set Lock State (SLS)请求，也就是将锁信息直接写入到CF上的内存中。CF会确认集群中的其他成员没有锁定这个行，并直接修改请求成员的内存以批准锁请求。

这个SLS只需15微秒就可以完成整个过程，因此代理不需要让出已分配的CPU时间。代理可以持续高效运行，而不需要像其他横向扩展架构那样等待IP中断(避免不必要的上下文切换)。对于长时间运行的批量事务等特定操作来说，DB2代理有必要让出CPU时间，而DB2会自动决定是否动态让出CPU时间。

DB2 pureScale内置的针对集群成员的负载均衡机制是另一个重要的DB2可伸缩性特性。应用不需要具备集群感知性便可利用负载均衡机制。DB2 for z/OS数据共享客户如今所使用的客户端驱动程序可以为DB2 pureScale提供集群负载均衡特性。

DB2 pureScale实现可用性

横向扩展架构的作用并不仅为了处理能力的增加。采用这种架构设计的系统在遇到组件故障时可以继续处理事务，从而能够交付更高的可用性。

与分布式平台上的其他产品相比，DB2 pureScale将可用性提升到了一个新的高度。DB2 pureScale允许访问所有不需要恢复的数据页面，并且随时可以洞察哪些页面需要恢复，而不需要执行任何I/O操作。这是通过集中化CF的独特功能实现的另一项重要创新。

每当成员将一个页读取到它的缓冲池中时，CF都会感知到这一事件并持续对其进行跟踪。任何时候当成员希望更新一页中的行，CF同样能够知晓相关事件。当一个应用执行事务时，成员会将脏页直接写入到CF的内存中。此流程允许集群中希望读取这些经过更改的页的任何其他成员直接从CF获取更新。更加重要的是，从恢复的角度来说，如果任何成员出现故障，CF中会保留该失败成员正在处理更新的页列表，同时还有一些页已经完成更新和提交，但尚未写入磁盘。

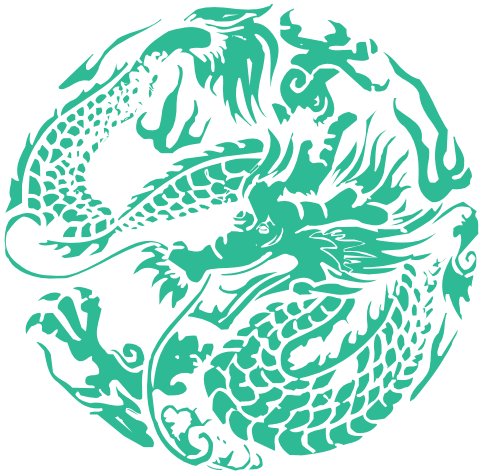
任何关系数据库管理系统(RDBMS)的恢复流程首先都需要重新执行任何已提交的事务，以确保磁盘上这些事务的页是最新的(此流程称作**redo恢复**)。此外，任何数据库服务器都需要撤销任何未完成的事务，即在故障之前对磁盘数据执行了更改但尚未提交(此流程称作**undo恢复**)。

在共享磁盘集群中，非常关键的一点是要确保集群中的其他节点没有读取或更新尚未恢复的磁盘中的任何页面(恢复这些页面之后才可以对这些行执行新的事务)。这正是CF的闪光之处：由于CF知道哪些页正处于故障节点的更新过程之中，并且CF已经将该节点提交的脏页保存在它的集中缓冲池中，因此DB2 pureScale在确定哪些页面需要恢复时不必阻塞其他成员持续处理事务。其他架构则需要了解哪些节点占用的处理时间较多，以便根据锁信息的分布来确定哪些节点必须恢复。

从较高的层面来看，可以很容易地解释DB2 pureScale环境中的这种恢复进程。每个成员都有空闲的进程，但它们都随时准备着处理故障事件。如果某个成员出现故障时，其中一个恢复进程便会激活；既然这些进程已经存在，因此操作系统不必浪费宝贵的系统时间来创建进程，为它分配内存等。此恢复进程会立即将CF中的脏页预取到它自己的本地缓冲池中。大部分恢复过程都不需要I/O操作，因为需要恢复的页已经在CF的集中缓冲池中了。此外，页预取机制使用轻量级的RDMA在CF与恢复成员之间实现迅速有效的传输。在这段时间内，所有其他成员上的所有其他应用将继续处理请求。如果它们需要从不需要恢复的任何页获取任何数据，那么它们可以继续执行自己事务。因此，它们可以继续从磁盘读取页，因为CF已经知道磁盘上的哪些页是干净的，以及哪些页需要恢复。然后，恢复进程读取故障成员的日志文件，以便于重放必要的事务来重做或撤销故障成员所做的更新。

对于典型的事务工作负载来说，从成员出现故障到故障节点未更新完的页面可供其他事务使用的时间间隔通常在20秒以内。注意，这同时还包括故障检测时间，而某些供应商在提到恢复时间时都排除了这一时间。数据库中的所有其他页面无时不刻(**甚至在成员出现故障之后**)都是完全可用的。

此外，系统中像PowerHA pureScale集群加速器这样的组件是冗余的。DB2 pureScale支持双重CF功能，这样锁和共享缓存信息就可以存储在两个相互独立的位置，以应对主CF出现故障的情况。



探究本质

与Oracle Real Application Clusters比较

如前所述，集中锁和缓冲池管理的应用为DB2 pureScale提供了其他同类市场产品所无法比拟的可伸缩和可用性优势。为了更加详细地探究这种差异化，我们将使用目前市面上常见的Oracle Real Application Clusters (RAC)与之进行比较。

DB2 pureScale与Oracle RAC的可用性比较

为了演示DB2 pureScale在系统可用性方面的竞争优势，我们将使用DB2 pureScale实现三个成员实例，并让其中一个成员出现故障。然后，我们将Oracle RAC三节点集群所需的恢复处理与之进行比较。两款产品都整合了用于节点故障检测的集群管理器，并且两者都可以迅速检测出软件和服务器故障，因此我们将重点关注恢复过程，而不是节点故障检测时间。需要说明的是，DB2 pureScale将使用一些创新的方法来迅速检测软件和硬件故障，同时能避免错误的故障转移。

首先，我们看一下Oracle RAC: 在Oracle RAC中，每个数据页(在Oracle术语中称作数据块)都由集群中的一个实例来管理和控制。Oracle采用分布式的锁机制，因此集群中的每个实例都需要负责管理和批准对它所管理的页的锁请求。当某个节点出现故障时，故障节点所管理的数据页会立即变为孤立的，同时RAC会通过重新分配流程将这些孤立的页分配给集群中健康的节点。在Global Resource Directory (GRD)重新配置的过程中，对该页的任何读取和锁请求都会立即被冻结。应用可以继续在健康的节点上处理，但这时它们不能执行任何I/O操作或请求任何新锁。这会造成许多应用出现冻结，如图1所示。

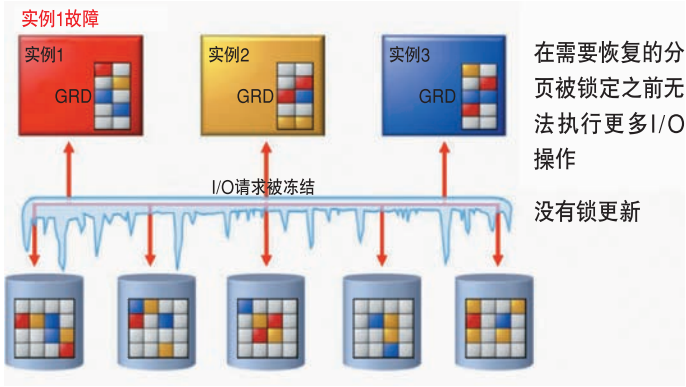


图1 Oracle RAC I/O在恢复流程中被冻结。

RAC节点恢复流程的第二步是锁定所有需要恢复的数据页面。这项任务必须在GRD冻结被释放之间完成。如果某个实例被允许在获得适当的页级锁之前读取磁盘中的页面，则故障实例的更新操作可能会丢失。恢复实例将一次读取故障实例的重做日志文件，并锁定任何需要恢复的页面，如图2所示。这可能需要大量随机的I/O操作，因为日志文件乃至需要恢复的页都可能不在任何健康节点的内存中。

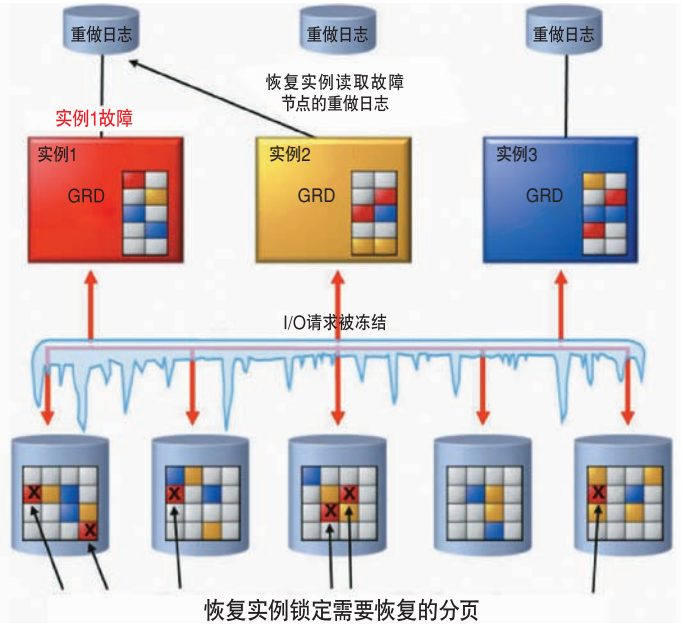


图2 Oracle RAC一次性读取日志

仅当恢复实例执行了所有这些I/O操作并锁定适当的页之后，GRD冻结才会被释放，停滞的应用才能继续运行。根据故障节点在出现故障时已经完成的工作量，这一过程可能会花费几十秒到几分钟不等的时间。有关这种GRD冻结以及在这段时间内不能执行I/O操作或批准新锁请求的详细信息，请参考本文档结束部分所提出的一些关于Oracle RAC的书籍[2]。

比较而言，DB2 pureScale环境则不需要在集群中进行全局冻结。CF在任何成员出现故障时始终知道哪些页面需要恢复。如果某个成员出现故障，集群中的所有其他成员可以继续处理事务和执行I/O操作。只有对需

要恢复的页面的访问请求会在故障成员的恢复流程完成之前被阻塞，如图3所示。

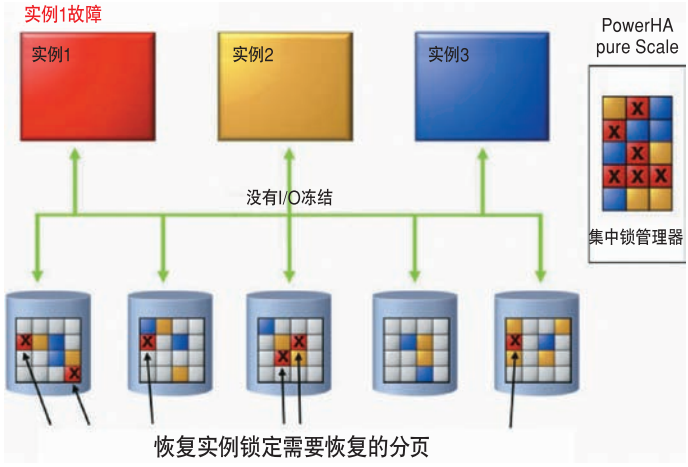


图3 DB2 pureScale恢复流程

DB2 pureScale与Oracle RAC的可伸缩性比较

为了清楚地展示DB2 pureScale和Oracle RAC之间的可伸缩性差异，我们将通过一系列示例来演示各成员之间的交互方式。然后，我们将描述DB2 pureScale和Oracle RAC如何在以下场景中工作:

1. 某成员希望读取其他成员都未缓存的页。
2. 某成员希望更新另一个成员正在更新页面。
3. 两个成员希望更新相同数据页面的不同行(这通常称作热页访问)。

注意: 在接下来的示例中，我们将Oracle和DB2中代表应用运行的进程分别称作服务器(SERVER)进程和代理(AGENT)进程。

对于Oracle RAC，它使用一些进程来管理集群中各节点之间的通信。这些进程处理全局锁请求(LMD进程)和全局缓存请求(LMSn进程)。为了简单起见，在这些示例中，我们将它们统一称作全局缓存服务(GCS)进程。但是，您应该知道进程间通信造成的开销要比下面的示例高很多，因为它需要使用单独的锁和缓存管理进程。[13]

场景1: 某成员希望从磁盘读取一个页面

Oracle RAC

如果某个服务器进程希望访问图4中的页面501，那么它首先会在自己的本地缓冲池中查找页面(图4中的步骤①)。如果未找到页面，则服务器进程会向GCS进程发送一个进程间通信(IPC)请求，以便向主节点请求该数据页面(②)。这会使服务器进程让出CPU时间，并且CPU将执行上下文切换，以便在CPU上重新建立GCS进程来处理中断。高级上下文切换有时会造成很高的开销。然后，GCS进程向主节点发送一个IP请求，以便获取所请求的数据块(③)。由于IP调用将在操作系统内核中处理，因此GCS进程需要将所请求的信息复制到内核内存中，然后再执行开销很高的IP栈调用，以将请求推送到远程节点中。即便使用了InfiniBand网络，Oracle仍然采用IP over InfiniBand机制，或者有时采用Reliable Datagram Sockets (RDS) over InfiniBand机制。在InfiniBand上使用套接字协议会因处理器中断、IP栈移动而造成很高的开销。

接下来，远程主GCS进程将接收到一个中断并将被调度来在CPU上处理请求。它会检查其他成员的缓冲缓存中是否有页面。在本例中，成员都没有页面，因此GCS进程会返回一条IP消息给请求者，通知它从磁盘读取页面(④)。请求节点上的GCS进程会再次中断来处理传入的IP请求，然后返回一个IPC中断(⑤)给服务器进程，通知它集群中的其他节点都没有页面。然后，服务器进程将磁盘中的页面读取到它自己的缓冲缓存中(⑥)。

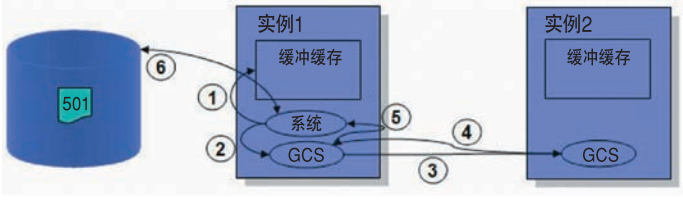


图4 Oracle RAC读取数据页面

DB2 pureScale

使用DB2 pureScale功能来执行相同示例之后，DB2 pureScale更具

伸缩性并且不需要数据本地性的原因将会更加明朗。其第一步与Oracle RAC相同: 代理进程首先检查自己的缓冲池, 确定所需的页面是否在内存中(图5中的步骤①)。如果未找到页面, 则代理进程本身会通过RDMA发送一个 Read and Register (RaR)请求给CF (②)。这与图4中的Oracle RAC的流程极为不同。在DB2 pureScale中, 代理本身会直接写入CF的内存(没有对其他进程的执行IPC操作, 没有中断, 没有通过内核进行IP调用, 没有上下文切换等)。如果代理希望读取页面501, 则写入CF内存的消息将类似于“我想要页面501, 并且希望将它存储在缓冲池的第42槽中”。当CF接收到此请求之后, 它会在自己的全局缓冲池中查找该页面。如果找到该页面, 则CF会根据请求通过RDMA将该页面直接写入到成员的内存中。如果未找到该页面, 则会直接将一个响应写入到成员的内存中以指出页面未在CF中, 这样代理便会从磁盘检索页面(③)。然后, 代理进程将执行所需的I/O (④)。

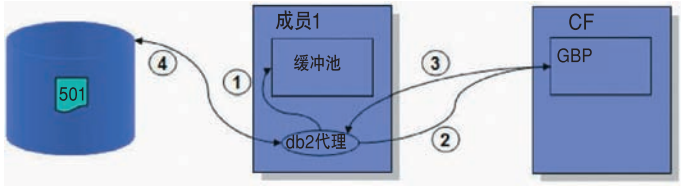


图5 DB2 pureScale读取数据页面

使用RDMA的效率要比IP通信高很多, 因为代理在许多情况下都不需要让出CPU。代理在请求数据时会有效执行一个memcopy命令, 然后从它自己的本地内存中读取响应, 如图6所示。

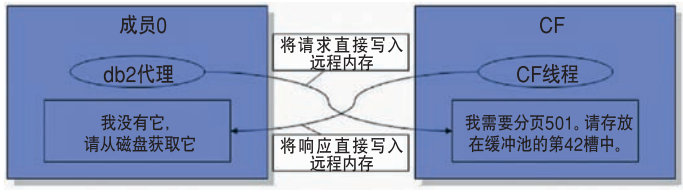


图6 DB2 pureScale环境中的RDMA请求

场景2: 某成员希望更新另一个成员/节点正在读取的页面

Oracle RAC

在本例中, 我们假设在一个由三个节点组成的RAC集群中, 实例A希望更新由实例B管理但当前却位于实例C的缓冲缓存中的某个页面(页面501)。在本例中, 初始步骤仍然是相同的。特别需要说明的是, 代理会发送一个IPC请求给本地GCS进程, 以指明它希望更新某个页面(图7中的步骤①) GCS进程发送一个IP请求给主节点(实例B), 以请求更新页面(②)。在本例中, 主节点知道实例C已经读取了页面, 因此它会向实例C再发送一个IP请求, 指示它放弃对页面的读取锁并将页面传递给实例A (③)。实例C上的GCS进程在接收到IP消息之后会针对中断做出反应, 释放对页面的共享锁, 将页面复制到IP栈, 并通过网络将页面发送给实例A (④)。在实例A的IP栈中检索到页面之后, 该节点上的GCS进程随后会返回一个IP消息给主节点, 以指示它正采用独占模式执行页面更新(⑤)。如果集群中的节点数量多于3个, 则主节点必须通知所有在本地缓冲池中有页面的节点放弃全局锁, 因为集群中的某个节点正独占该页面以执行更新操作。

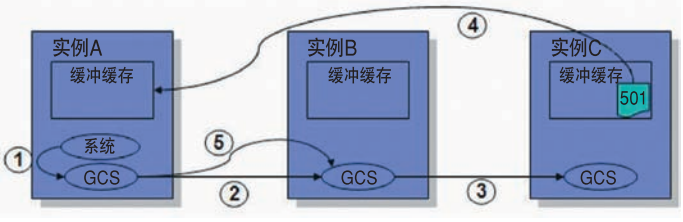


图7 某实例希望更新另一个实例缓存中的页面

DB2 pureScale

当集群中的两个或多个成员希望操作同一页面时, DB2 pureScale的执行方式有所不同。假设成员1希望更新其他成员已向CF请求的页面或集中缓冲池中的已有页面。初始步骤仍然是相同的: 代理进程首先检查本地缓冲池, 如果未找到页面, 则直接在CF内存中写入页面请求以及希望存储页面的缓冲池槽(图8中的步骤①)。在本例中, 由于CF中有页面, 因此它会将数据页面直接写入到成员1的内存中(②)。然后, 代理可以对页面中的行执

行处理。这时，CF不需要通知任何其他成员执行任何操作，因为在DB2 pureScale环境中，只有当成员对页面执行字节更改时才会在页面上采用独占锁。

如果此成员上的代理随后找到了需要修改的行，则会通知CF使用Set Lock State (SLS)调用。SLS调用也是一个RDMA操作，成员可以直接将需要锁定的行以及锁类型写入到CF内存中(③)。在本例中，如果它希望更新某行，则会通知CF它希望对该行使用X锁，并对页面使用P-lock (这表示它会对页面执行字节修改)。假设集群中没有其他成员正在更新这个页面，并且其他成员没有对此行使用X锁，则CF会直接将批准锁的响应写入到成员的内存中(④)。对于未被争用的锁请求，SLS调用需要花费大约15微秒(或更少)的时间，因此这种高效性再一次确保了代理不需要让出CPU。

当页面上的字节发生更改之后，如果集群中的其他成员希望更新同一页面，该成员可以立即释放P-lock；但是，如果没有其他成员对此页面感兴趣，该成员会保留P-lock并在事务提交后释放该锁(称作**迟缓锁释放**)。这些锁不会在整个事务执行过程中被持续持有，除非没有其他成员请求锁。当其他成员请求这个锁时，占用锁的成员便会立即释放它，即使事务还没有完成。

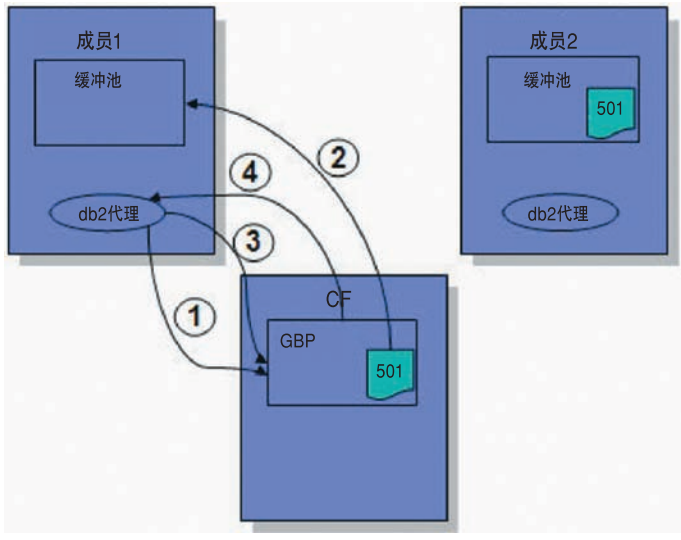


图8 成员1希望更新全局缓冲池中的页面

场景3: 两个成员希望更新相同页面中的不同行

Oracle RAC

当两个成员希望更新相同页面中的不同行时，数据缺乏本地性有时会发展成Oracle RAC中的一个严重问题。已出版的一些关于Oracle RAC的书籍描述了如何识别集群中的热页面，以及如何尝试缓解这种问题，比如减少页面的行数量、对应用进行分区等。图9中的示例演示了Oracle RAC集群中的两个节点希望更新相同数据页面中的不同行的场景。

注意: 在本例中，我们避免了反复提及的GCS和IP中断的开销。各节点之间的每一个请求都需要经过这个开销高昂的栈。

假设实例A希望更新页面中由实例B管理的某一行。现在，再假设实例C也希望更新该页面中的某一行(与实例A希望更新的行不同)，并且实例C已经拥有该页面上的全局X锁。实例A会询问实例B它是否可以对页面进行更新(图9中的步骤①)。实例B通知实例C释放其页面锁并将数据块传递给实例A (②)。实例C中断其处理并将数据块发送给实例A (③)。实例A随后通知主节点它已经拥有独占的全局页面锁(④)。然后，实例A开始在页面中查找需要更新的行。但是，实例C上的服务器进程并未完成对该页面的处理，因为它还没有检查各行，因此它向实例B上的主节点发送消息，请求再次独占页面(⑤)。主节点中断实例A并通知它释放对页面的独占锁，并将它传回给实例C (⑥)。实例 A 服从指示并通过网络将数据块传递给实例C (⑦)，然后实例C 通知主节点它占用了该页面(⑧)。

这时，两个实例会通过网络来回传递页面，直到各实例检查了页面中的每一行并执行了各处的更新(处于简单起见，图9并未显示此过程；但是，消息和页面在实例之间的循环传递将继续进行)。

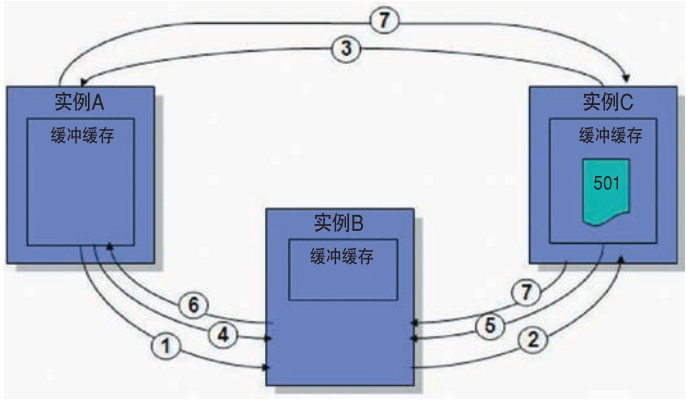


图9 Oracle RAC热页面访问

DB2 pureScale

这个场景在DB2 pureScale环境中截然不同。假定成员1希望更新CF中的某一行，并且成员2也希望更新相同页面中的不同行。

在此场景中，成员1向CF请求页面(图10中的步骤①)。与之前的示例相似，该页面会直接从CF推送到成员1的内存中(②)。每个成员都可以同时读取页面中的行，即使它们都希望更新某一行。当各成员找到需要更新的行之后，它们请求对要更新的行使用X锁，并请求对CF中的页面使用P-lock (③)。第一个执行此操作的成员将获得批准，它将被允许通过更新来修改页面(④)。其他成员(在本例中为成员2)将被允许继续读取页面中的行，直到发生以下两个事件之一:

- 如果成员1提交事务，它会将新版本的页面推送到CF的内存中，并且CF随后会通知成员2它当前所拥有的页面是无效的。CF会将新的页面副本推送给成员2 (均通过RDMA实现)。

- 如果成员2找到了希望更新的行，则它会向CF请求锁(⑤)。这时，CF会通知成员1释放它的P-lock。成员1将使用内存复制RDMA操作并将更新后的页面存入全局缓冲池中(⑥)。然后，CF将新版本的页面复制到成员2的内存中(⑦)，并批准锁请求，允许它对页面执行更改。

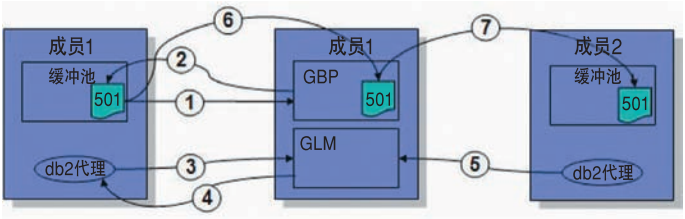


图10 DB2 pureScale更新热页面

两者之间主要区别在于: 在Oracle RAC中, 即使有意更新页面中的某个行也会获得页面锁, 而在DB2 pureScale中, 当且仅当需要实际更改行时才会请求页面锁(因此页面会在服务器之间移动)。这种页面级的并发性增强意味着可以不需要担心数据的本地性, 甚至对于多个更新者来说也是如此, 从而能够实现较高水平的可伸缩性而无需更改应用。

Oracle专家对构建可伸缩的应用的建议

在Oracle RAC中, 实现可伸缩性的最佳方法是确保应用希望访问的数据页面的节点由它所连接到的节点管理。注意, 随着集群的不断壮大, 主节点同时也是所连接到的节点的几率将不断降低。举例来说, 在由两个节点组成的集群中, 特定数据页面由当前节点管理的几率是50%。然而, 在10个节点的集群中, 此几率就降到了10%。因此, 为了实现更好的可伸缩性, 您应该确保将更新某组页面的事务与其他节点上访问不同页面的事务分开(以避免热页面问题)。这也就是所谓的**通过应用和数据库分区来实现应用的数据本地性。**

Oracle Japan Database Group的高级工程师Tsutsui Tomonari在一篇演示文稿中提到通过应用分区来实现数据本地性的技巧。Tomonari先生在Oracle Unbreakable Summit (C-4会议)上提交了一份标题为“Oracle Real Application Clusters 10g Best Practices”的演示文稿。在这个演示文稿中, Tomonari称新应用实现Oracle可伸缩性的最佳

实践是通过应用分区来减少争用和避免从多个节点访问相同的表。此外，他表示为了减少节点之间的数据块争用，您应该减少每个数据块和分区表的行数，以避免热页面问题。甚至在2008年2月发表在Oracle-l论坛上的一篇帖子还说到:

“我建议

- **通过应用分区来避免数据块级并发性**
- **对最常用的对象使用较小的数据块”** (3)

同样，José Valerio 在2009年7月8日发表的博客文章中讨论了节点间通信以及相关的等待事件问题。在这篇文章中，Valerio建议“较少地提交需要修改应用的操作”，以及“通过应用分区来避免数据块级并发性”(4)。还应注意，Valerio并非只是一名Oracle专家，他曾经任职于Oracle的Oracle US RAC Development团队并在Oracle公司担任“Oracle技术和RAC专家”(5)。

开发这种集群感知的应用的成本要比不需要数据本地性的扩展方案高很多。此外，如果数据库将增长为4个节点，则包含3个节点的应用在经过分区之后可能需要修改(因此需要重新测试和重新部署)。

在DB2 pureScale中，无论集群的大小如何，成员与集中化CF之间始终维持着通信。通过使用一种更加有效的“内存到内存”复制架构，并将热页面集中存储在真正全局化的缓存中，DB2 pureScale交付了透明的应用伸缩性并降低了在页面式平台上开发和重新设计应用的成本。

结束语

通过利用现代化的硬件架构，DB2 pureScale可以将之前仅在DB2 for z/OS上可用的集中锁和缓存功能引入到分布式平台中。对硬件和网络的利用提高了并发性水平并显著降低了开销，从而提供了更高水平的可伸缩性。此外，集中锁和页面缓存允许DB2 pureScale持续感知在成员遇到故障时需要恢复哪些页面。因此，在遇到故障时，所有不需要恢复的数据仍然能供其他应用使用，而故障节点正在更新的页面将更加迅速地被恢复。

对于需要高可用性以及通过横向发展实现成本收益的应用来说，DB2 pureScale提供了一个可以满足这些需求的解决方案，并且已经过了市场的考验。

参考资料

eWeek文章http://www.eweek.com/c/a/Database/In-Larrys-Own-Words/2/

Global Resource Directory Freeze的出版物:

- Oracle Real Application Clusters – ISBN 978-1555582883 – 493页
- Oracle 1 0g Real Application Clusters Handbook – ISBN 978-007 1465090 – 182页
- Oracle 10g RAC: grid, services & clustering – ISBN 978-1555583217 – 265页

Oracle-L论坛帖子http://www.freelists.org/post/oracle-l/gc-current-block-busy-wait-in-ConcurrentBATCH-Processes-Run,2

博客文章http://jose-valerio.com.ar/blog/?p=196

José Valerio简介http://jose-valerio.com.ar/blog/?page_id=2

使用DB2 pureScale实现透明的应用扩展——技术比较，2009年10月

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
加拿大

在中国印刷

2009年10月