IBM Software Group

2009 IBM CDL Informix L2 蓝色脉动系列之四：

Informix 常见问题及处理方法总结

高旸 Advisory I/T Specialist 2009.12

ON DEMAND BUSINESS

议程

- 系统表（Catalog Table） 篇
- 空间（Dbspaces, Chunk）篇
- 表、锁及索引（Table, Lock and Index）篇
- 逻辑日志（Logical Log）篇
- 长事务（Long Transaction）篇

系统表(Catalog Table)篇

DBA应当对一些系统表比较熟悉，如：systables、sysindexes、sysindices

1、通过系统表查找“Partnum”

1.1、 查找标准表的partnum

（partnum在Informix数据库内部是表的唯一标识，默认partnum是十进制的如果需要可以转为十六进制）

例如：

Dbaccess->your database

select hex(partnum) from systables where tabname='test1';

(expression)

0x001000DC

1.2、查找分片表的partnum

（分片表的partnum比较特殊）

注：Partnum格式 前3位: Dbspace number, 后5位: Logical page number within tblspace tblspace

首先建立分片表：

```
create table "informix".test2
(
id integer,
name char(10)
) fragment by expression
(id < 10 ) in datadbs ,
((id > 20 ) AND (id < 30 ) ) in datadbs1
extent size 16 next size 16 lock mode row;
```

常规方法查询：

dbaccess->your database

select hex(partnum) from **systables** where tabname='test2';

结果：

(expression)
0x00000000

select hex(partn),tabid from **sysfragments** where tabid=(select tabid from systables where tabname='test2')

查询结果：

(expression)	tabid
0x00200002	101
0x00300002	101

2、查找rowid

（rowid是确定表内行信息的数据库标识）

2.1、普通表查询rowid

select rowid from tabname

2.1、分片表查询

select rowid,id from test2;

857: Rowids do not exist on table

Rowid格式：
例如0x12345678
123456 前6位logical offset into the tblspace
78 后2位slot number on the page

3、查找数据库的**Locale**

select * from **sysdbslocale** where dbs_dbsname='testdb';
结果：
dbs_dbsname testdb
dbs_collate en_US.819

4、查找相关的**constraints**

alter table test1 add constraint primary key (col1);
select * from **sysconstraints** where tabid=(select tabid from systables where tabname='test1');

结果：
constrid 3
constrname u100_3
owner informix
tabid 100
constrtype P
idxname 100_3
collation en_US.819

5、查找锁owner

5.1、我们使用session 82显示的锁住表test2

onstat -u

125daa488 Y--P--- 82 informix 15 126dfad50 0 2 3 0

语法

```
begin work;
lock table test2 in exclusive mode;
```

5.2、另起一个终端界面模仿另一个session去查询到底谁锁了表

dbaccess-> sysmaster

select owner from **syslocks** where tabname='test2';

查询结果：

dbsname testdb
tabname test2
rowidlk 0
keynum 0
type X
owner 82

6、如何查找相关用户的权限

6.1、数据库级权限

select usertype from **sysusers** where username='informix';

结果：
usertype
D
D代表DBA、C代表connect、R代表Resource

6.2、表级权限

dbaccess-> your database
select * from **systabauth** where tabid=(select tabid from systables where tabname='test1');

查询
grantor informix
grantee public
tabid   100
tabauth su-idx---

其中tabquth字段代表grantee所拥有的权限，如果是小写意味着不能grant给其他用户，如果大写则可以
s=select、u=update 、i=insert、d=delete、x=index

空间（Chunk/ Dbspace）篇

1、如何监控Informix数据库空间的异常离线?

1.1、在相关的chunk进行I/O操作时如果相应的Chunk有问题，数据库会报相应的I/O错误，并将CHUNK置为“PD”，另外当数据库启动或是进行数据库备份时，Informix数据库会对所有空间进行例行的健康检查“sanity check”， 如果相应的chunk有错误，Informix数据Informix online.log中输出如下的错误信息：

```
08:39:58  IBM Informix Dynamic Server Version 9.40.FC4
08:39:58   Who: Session(1, informix@HBDB84_1, 0, c000000000b63028)
   Thread(15, main_loop(), c000000000b21028, 3)
   File: rspartn.c Line: 7747
08:39:58   Results: Chunk 117 is being taken OFFLINE.
08:39:58   Action: Restore chunk from archive.
08:39:58  stack trace for pid 9747 written to /tmp/af.3f7f15e
08:39:58   See Also: /tmp/af.3f7f15e
08:39:59  chunk failed sanity check
08:39:59  I/O error, Primary Chunk '/opt/informix/chunks/npmhis2008
_data/npmdb_npmchk_07' -- Offline (sanity)
```

1.2、“Oncheck –pr ” 输出
当数据库停止后，数据库的所有状态信息都会写到相应的Informix保留页中，此时离线状态下运行oncheck -pr可以准确的看到相关已经被置为"offline"的空间信息“oncheck -pr |grep ffline” 的输出：

Chunk number 73
Flags 0x10020 **Chunk is offline**
Chunk path /opt/informix/chknew/npm/hpmchk1_b
Chunk offset 5 (p)
Chunk size 10000000 (p)
Number of free pages 6723197
DBspace number 38

1.3、“Onstat –d ” 输出
当Chunk 离线后，“onstat -d”输出的"free"栏中的值通常是0，有时会被误认为是空间满不可用了，其实是chunk异常PD后导致的。

“onstat -d |grep PD” 的输出：
Chunks
address chunk/dbs offset size free bpages flags pathname
c00000020a8434f0 73 38 5 10000000 0 PD-B
/opt/informix/chknew/npm/hpmchk1_b

1.4、相应的堆栈函数中我们可以看到“sane_chopen”，“afwarn_interface ”函数，表示对chunk进行sanity check,发现了错误

相应的堆栈输出：
(0) legacy_hp_afstack
(1) afstack
(2) afhandler
(3) afwarn_interface
(4) sane_chopen
(5) chopen_util
(6) chopen
(7) rscon
(8) aud_iscon
(9) chgstat
(10) onspace
(11) startup
(12) resume

2、如何检查确认空间离线的原因?

Chunk/Dbspace离线的原因多种多样，在数据库DBA寻找Informix 技术支持工程之前，应该排除诸如：物理磁盘设备损坏、
硬件设备状态异常、操作系统异常导致的Informix 空间异常的情况

2.1、检查相应的硬件日志、操作系统日志看是否有磁盘或硬件报错的信息；
2.2、对相应的磁盘物理设备进行硬件级检查，排除磁盘物理损坏的可能；
2.3、使用操作系统命令对相应的磁盘设备进行I/O读取检查

例如:dd if=/opt/informix/chknew/npm/hpmchk1_b of=/dev/null count=20000

2.4、检查相应物理磁盘设备的属组是否为"informix:informix 660“

例如:ls -lt /opt/informix/chknew/npm/hpmchk1_b

通过上述步骤的逐一排查，将会定位空间离线的原因，根据不同的原因我们将采取不同的方法进行修复

3、如何解决、修复空间离线问题

3.1、如果是磁盘物理损坏导致了数据丢失或损坏，最终导致的Informix数据库空间异常离线，需通过磁盘修复和数据库恢复进行修复或是强行将坏掉的空间从数据库删除修复；

3.2、如果是LV切换或是LV状态异常导致的Informix数据库空间离线，而磁盘硬件未有任何的损坏，则DBA可以通过　如下命令自行修复离线的空间;

例如：onspaces -s <spacename> -p <path> -o <offset> -O -y

3.3、如果是相关的属组权限的问题，修改为正确的属组和权限（客户在进行相应的磁盘维护和划分时，有时会改变 相应磁盘空间的属组和权限导致Informix数据库的离线）；

例如：
chown informix:informix /opt/informix/chknew/npm/hpmchk1_b
chmod 660 /opt/informix/chknew/npm/hpmchk1_b

3.4、如果磁盘修复后，数据并未损坏，DD操作也没有报错，备份恢复时间过长，此时可以考虑需求Informix技术支持使用内部工具尝试修复，但前提是内部工具存在风险，客户一定要有备份，以备不时之需；

3.5、**按照原有的onstat -d情况，重建相关空间，备份恢复；**

备注：当部分空间离线后，建议立刻停掉所有的业务操作，因为虽然有些空间依然在线可用，但离线的空间此时不可操作，数据的一致性和业务的一致性在这种情况下很难保障。

4、删除作废空间的方法

Informix 空间(Chunk/Dbspace)里边存放着客户重要的生产数据和索引，非到万不得已，不建议进行空间删除操作，关键的数据库空间例如“rootdbspace”、“physical log dbspace” 和“logical log dbspace” 不能删除。进行删除操作前最好做好相关的备份，以便需要时进行恢复。

4.1、Informix不允许删除非空的数据库空间，所谓的非空并不是指什么信息都没有，而是指除了系统所需的53页外不允许有其他的页被使用，如果数据库空间（rootdbspace除外）仅有53页被使用，则认为是“空”可以通过“onspaces -d”删除；

例如：

```
/home/informix/940:oncheck -pe testdbs
DBspace Usage Report: testdbs          Owner: informix  Created: 10/13/2009
 Chunk Pathname                         Size      Used      Free
    2 ./dsk/gydbs                       1000        53       947

 Description                                    Offset     Size
 ---------------------------------------------------------- -------- --------
 RESERVED PAGES                                      0        2
 CHUNK FREELIST PAGE                                 2        1
 testdbs:'informix'.TBLSpace                         3       50
 FREE                                               53      947
 Total Used:       53
 Total Free:      947
```

4.2、临时表空间可以直接通过“onspaces -d”删除；

4.3、 如果有异常表或数据库存在，通过drop table或drop database无法正常删除相关使用的空间，使用Informix技术支持内部工具强行删除，但此操作存在一定风险

表、索引(Table、Index)篇

1、删除、修改主键的

有时在磁盘损坏导致数据或索引不一致后，客户发现主键损坏，如何删除和重建主键？

ALTER TABLE manufact DROP CONSTRAINT con_name;

ALTER TABLE tabname ADD PRIMARY KEY (coluname);

2、PDQ建索引

PDQ功能可以是单个会话使用到更多的数据库的资源以提高速度，一般在建索引时效果比较明显

2.1、在onconfig中可以设置如下的参数：

```
MAX_PDQPRIORITY 100          # Maximum allowed pdqpriority
DS_TOTAL_MEMORY 2000         # Decision support memory (Kbytes)
```

如果不能重启实例，可以通过如下命令开启当前session的PDQ

```
onmode -D   <max PDQ priority allowed>
onmode -M   <decision support memory in kbytes>
```

2.2、开启PDQ的语法：

```
set pdqpriority 80;
```

2.3、监控PDQ使用情况的方法
onstat -g mgm
PDQ应酌情使用，适用于单个会话需要使用大量资源的场景，如果多个会话同时需要使用PDQ，应注意优先级的合理分布，以防影响数据库的正常运行

3、Extent和页限制

3.1、 由于数据库结构的限制，相应的数据库都有Extent和页的限制
一般2K页操作系统的Extent限制在200左右，4K页的再400左右，建议定期进行数据库健康检查，防止由于Extent超限导致的数据无法插入的情况；页的限制是hex (FFFFFF) = dec (16777215)，如果页数超限页会报数据无法插入的错误。

1、如果应用报数据无法插入，此时发现dbspace/chunk的空间都未满，此时应当用oncheck -pT 检查相关表、索引的“Number of extents”、 “Number of pages used ”使用情况；
2、相关表相关的触发器所涉及的表也需要检查；

4、锁表的处理

通过onstat -k,onstat -u以及onstat -g ses处理锁表问题的一般方法

4.1、找出被锁表的partnum

```
select hex(partnum)from systables where tabname='gy'
 partnum = 10005E
```

4.2、onstat -k 查找相关的锁的owner

```
/home/informix/940:onstat -k |grep 10005e
10a2ca328        0             10e9442a8       10a2c9ef0       HDR+X    10005e   0         0
```

4.3、onstat -u中查找相应的user信息

```
/home/informix/940:onstat -u |grep 10e9442a8
10e9442a8        Y--P---  2073     informix 16       10f861738       0    2    0       0
```

4.4、onstat -g ses sid查出具体的session信息

4.5、onmode -z 2073 如果有必要杀死相关会话解锁

5、裸表、分片表

5.1、Raw table由于不记日志，没有相关的约束和检查，性能比较高。进行批量业务时最好不要一次进行大数据量的操作，大批量数据的装载，修改，删除最好分批执行。或将先此表暂时修改为raw table，防止数据库写入大量的日志。操作完成后，再将表改回正常的模式。

```
alter table <tabname> type(raw);
alter table <tabname> type(standard);
```

5.2、分片表的建立

对大表进行分片，有利于I/O的负载均衡、空间上限的增加等，分片有两种方法，第一种round robin法,第二种expression法，round robin是均分所有数据到各个dbspace中去，expression是根据客户提供的表达式分配到不同的dbspace中去。

例如：

1、round robin法，数据将均分到datadbs,datadbs1中：

```
create table "informix".test1
( id integer,name char(10))
fragment by round robin in datadbs , datadbs1
extent size 16 next size 16 lock mode row;
```

2、表达式法，id < 10的将分配到datadbs, 30>id >20将分配到datadbs1中：

```
create table "informix".test2
( id integer,name char(10))
fragment by expression
(id < 10 ) in datadbs ,
((id > 20 ) AND (id < 30 ) ) in datadbs1
extent size 16 next size 16 lock mode row;
```

6、如果监控系统中消耗CPU较高的

6.1、现象：数据库突然变慢，系统CPU占用较高，重启后问题依旧。

分析：通常出现在下面几种情况：系统运行一段时间，数据量增大修改或增加了新的应用。应用并发量上升，系统负荷增加大多数情况下是由大量的顺序扫描造成的。顺序扫描是数据库在访问表时，不通过索引，而直接扫描数据页，访问表的一种方式。这种方式不但读取效率很低，而且会消耗大量的包括IO，CPU在内的系统资源，导致整个系统性能严重下降。

6.2、检查顺序扫描的方法

```
select first 50 a.tabname, b.nrows, sum(a.seqscans) tot_scans
from sysmaster:sysptprof  a, systables b
where seqscans > 0
and dbsname = 'crmdb'
and a.tabname[1,3]<>'sys'
and a.tabname = b.tabname
group by 1,2 order by 2 desc;
```

6.3、检查cost值大的SQL语句

在sysmaster库中执行

```
select sqx_estcost,
sqx_sqlstatement
from syssqexplain
order by sqx_estcost desc
```

7、通过onstat -k信息发现锁的级别

有时知道上了锁，但并不知道锁的级别即是锁了整个表？还是锁了相关的页，还是某一行？下面的例子将解答这个问题。

注释：tblsnum 0x100002 是数据库database tablespace，是informix数据库的内部结构，在onstat -k输出中可以忽略相关内容

$ onstat -k

IBM Informix Dynamic Server Version 9.40.UC7 -- On-Line -- Up 2 days 03:59:27 -- 39736 Kbytes

Locks

address	wtlist	owner	lklist	type	tblsnum	rowid	key#/bsiz
4409ad50	0	44d5bac0	0	S	00002	303	0
4409ada8	0	44d5bac0	4409ad50	HDR+IX	1003ba	0	0
4409ae00	0	44d5bac0	4409afb8	**HDR+X**	**1003bb**	**101**	**K- 1**
4409ae58	0	44d5bac0	4409ae00	**HDR+X**	**1003bb**	**201**	**K- 1**
4409aeb0	0	44d5bac0	4409ae58	HDR+X	1003bc	101	K- 1
4409af08	0	44d5bac0	4409aeb0	HDR+X	1003bc	201	K- 1
4409af60	0	44d5bac0	4409ada8	HDR+X	1003ba	101	0
4409afb8	0	44d5bac0	4409af60	**HDR+X**	**1003ba**	**201**	**0**
440b0510	0	44d5a878	0	HDR+S	100002	303	0
440b0568	0	44d5a878	440b0510	HDR+IX	1003b8	0	0
440b05c0	0	44d5a878	440b0568	**HDR+X**	**1003b8**	**300**	0
440b0720	0	44d5c0d8	0	S	100002	303	0
440b0778	0	44d5c0d8	440b0720	**HDR+X**	**1003c2**	**0**	0

13 active, 2000 total, 2048 hash buckets, 0 **lock table** overflows

1、绿色的字显示的是在表0x1003c2上的表级别锁

2、蓝色字显示的是在表0x1003b8第三个页(0x3)的页级锁

3、红色的字显示在表0x1003ba，第二页(0x2)第一个槽的行级锁

4、紫色的字显示的是在表0x1003bb，第一页第一个槽，第二页第一个槽索引(K- 1)上的锁

如果想要知道是哪一个索引引起了锁，可以通过oncheck -pT <0xtblsnum> 查找

```
$oncheck -pT 0x1003bb | grep "Index Usage Report"
Index Usage Report for index  101_3 on stores_demo:informix.orders
Index Usage Report for index  101_70 on stores_demo:informix.orders
```

8、数据分布和update statistics 的使用

8.1、统计更新对于更新优化器查询路径信息，索引路径信息和数据分布都有重要作用，在进行数据库升级、迁移、索引的重建、表的重建后都建议进行统计更新，另外建议定期执行统计更新提高数据库的性能。

如果数据删除和插入频繁，需要执行或在数据迁移后

删除数据分布的方法：

update statistics low for table tabname drop distributions;

对非索引字段执行中级优化:

update statistics medium for table tabname (col...);

对索引字段，复合索引的首字段:

update statistics high for table tabname (idex1...);

8.2、查看数据分布的方法：

Dbschema –hd tabname –d dbname

8.3、通过系统表查看数据分布的方法：

Select * from **sysdistrib** where tabid=(select tabid from systables where tabname='test1');

逻辑日志（Logical Log)篇

有时在没有数据库审计的情况下，也需要对数据库的某些操作进行监控或审计，此时一般只能通过对逻辑日志的分析来实现，下面我们将举两个通过逻辑日志监控alter table和update table的例子

例子1：假设怀疑数据库“stores7”中的表“state”再 2003年9月29日 16:00 到18:00并恶意的修改了，我们需要查处谁在什么时间发起了这个alter操作。

1、通过在online.log 中通过CKPT完成的时间来确认相关的“logical logs”

Mon Sep 29 17:20:06 2003

17:20:38 Checkpoint loguniq 287, logpos 0xd2018
17:23:34 Checkpoint loguniq 287, logpos 0xd5018
17:26:22 Logical Log 287 Complete.
17:26:26 Checkpoint loguniq 288, logpos 0x18
17:47:05 Checkpoint loguniq 288, logpos 0x1018
17:52:06 Checkpoint loguniq 288, logpos 0x2018

通过上边的日志信息，我们可以猜测修改操作大概发生在逻辑日志 287 或 288。

2、确认这些被选中的逻辑日志的状态和位置

（有没有被备份，在磁带中还是在磁盘上），通过onstat -l我们发现逻辑日志287和288还没有被备份，依然在磁盘上

flags   uniqid
U------ 287
U---C-L 288

3、找到state表的partnum

```
$ dbaccess stores7 -
Database selected.
> select hex(partnum) from systables where tabname='state';
(expression)
0x001001FC
1 row(s) retrieved.
```

4、查找相关操作记录

4.1、使用onlog命令找出关于表state的在相应时间段的所有记录

```
$ onlog -n 287 -l -t 0x001001FC > onlog-287.out
$ onlog -n 288 -l -t 0x001001FC > onlog-288.out
```

4.2、在相应的记录输出中查找相应的ALTER关键字

```
$ grep ALT onlog288.out

308c 52 PTALTER 04 0 3040 1001fc 0 1130 1131 1 1 -20
```

4.3、分析整个记录确认所需事物信息

```
addr     len  type          xid      id link
308c     52   PTALTER  104      0  3040     1001fc   0       1130     1131     1    1    -20
         34000000 00005d00 10000000 68000000 4.....]. ....h...
         40300000 e3858800 00000000 fc011000 @0..... ........
         01000100 00000000 6a040000 6b040000 ........ j...k...
         ecff0000                            ....

Transaction id: xid = 104
```

5、使用onlog和BEGIN关键字得到关于事物104的详细信息

```
onlog -n 288 -l -x 104 > onlog288-104.out

log number: 288.

addr     len  type     xid      id link
3018     40   BEGIN    104      288 0       09/29/2003 17:55:26 13       Administrator
         28000000 20010100 00000000 68000000 (... ... ....h...
         00000000 e1858800 00000000 4eaa783f ........ ....N.x?
         06000000 0d000000                   ........
```

逻辑日志的记录我们可以看出"Administrator"用户，在09/29/2003 17:55:26发起了相应的alter table操作

例子2：审计相关数据的update情况

1、 建相关测试表并确认相关表的partnum

```
begin work;
create table lc (id int);
commit;
```

2、 插入测试数据并确认相关行的**rowid**

```
begin work;
  insert into lc values(1);
commit;
```

3、 简单模仿客户的业务

```
begin work;
 select * from lc where id=1;
 update lc set id=2 where id=1;
commit;
```

4、 onlog -l -n 18

```
1018    48   BEGIN    8       18 0       11/25/2008 20:25:52 27      informix
        00000030 00120001 00000000 00007b78 ...0.... ......{x
        00000008 00000000 003529f6 00000000 ........ .5).....
        00000000 492beed0 0000006a 0000001b ....I+.. ...j....
1048    52   HINSERT  8       0  1018    200448   101     4
        00000034 00000028 0001c112 00008b1f ...4...( ........
        00000008 00001018 003529f6 00200448 ........ .5).. .H
        00200448[I1] 00000101[I2] 00040000 00000000 . .H.... ........
        00000001                          ....
```

通过分析我们可以看出informix用户在11/25/2008 20:25:52 27 修改了表（0x 00200448）的行（0x00000101)数据

长事务（Long Transaction）篇

1、如何区分不同种类的长事务

有些Informix客户经常会遇到长事务的情况，通常在online.log会发现有不同的关于长事务的警告信息，如何区分这些不同的长事务类型，从而采取不同的方法去处理

通常在online.log会有两种不同的关于“long transaction”的信息提示：

第一种：
在online.log中有如下信息提示：“Continuing Long Transaction (for COMMIT): tx 0xc0000000b28f5338 username: fisp uid: 107” 在这种情况下，事务使用逻辑日志的量已经超过了长事务高水位值（LTXHWM），但此时的事务自己已经进入了“commit”或“roll back”阶段，此时数据库引擎将允许事务继续使用逻辑日志，而不会强行回滚该事物，所以上述类型的长事务不会阻塞数据库，并且系统会自动等待事务自行处理完毕。

第二种：
在online.log中有如下信息提示：“Aborting Long Transaction: tx 0xc0000000b8d20c18 username: fisp uid: 107” 这种情况的长事务已经超过了长事务高水位值（LTXHWM）并且没有自动进入到“commit”或“roll back”阶段，此时数据库会开始主动回滚该长事务在这种情况下，客户需要监控事务的回滚状态，必要时需要采取一定的措施防止该长事务阻塞数据库，影响生产。

2、如何监控长事务回滚进度

介绍：IBM Informix 产品中，由于错误不得当或是业务逻辑设计的不得当，经常会出现长事务的情况，这种情况下，数据库引擎一般都会阻塞在 “Blocked:LOGTX “ 状态，此时业务将被迫中断，数据库被阻塞，知道长事务回滚结束，数据库方可恢复正常。在此过程中，往往需要耗费大量的时间，如何监控长事务回滚的状态也就成了一般数据库管理员的当务之急。

步骤： 从online.log或是通过“onstat -x”输出中，识别并找到相应的长事务，然后执行“onstat -g dmp 0x<transaction address> rtx_t”得到更多的关于这个事务的信息，例如：

1、执行“onstat -x”得到如下信息：

IBM Informix Dynamic Server Version 7.31.UD8 -- Fast Recovery -- Up 01:17:23 -- 44520 Kbytes

Transactions
address flags userthread locks log begin isolation retrys coordinator

a1fb4028 A---- a1f7c028 0 0 COMMIT 0
a1fb4220 A---- a1f7c718 0 0 COMMIT 0
a1fb4418 A---- a1f7ce08 0 0 COMMIT 0
a1fb4610 A---- a1f7d4f8 0 0 COMMIT 0
a1fb4808 A---- a1f7dbe8 0 0 COMMIT 0
a1fb4a00 A---- a1f7e2d8 0 0 COMMIT 0
a1fb4bf8 A-R-- a1f7e2d8 0 2755 COMMIT 0

2、执行onstat -g dmp 0xa1fb4bf8 rtx_t | \
egrep "tx_logbeg|tx_loguniq|tx_logpos|tx_longtx" 得到如下信息：

```
tx_logbeg    = 2755
tx_loguniq   = 2758
tx_logpos    = 579480
tx_longtx    = 0
```

从上边的信息我们可以看到，这个例子中的事物开始于逻辑日志2755，现在正在逻辑日志2758进行回滚。

3、如何计算事务回滚时间

首先logpos的结构是 “0x008f61c8” （不够位的前边可以用零补足） 前五位是“page offset”本例中是"0x008f6"，后三位是“byte offset into page”本例中是“1c8”,计算式我们只粗略考虑page的情况 如下例（平台是AIX,默认页大小是4KB)

```
onstat –x –r 1
address  flags userthread locks  beginlg curlog  logposit   isol    retrys coord
d745b58  A-R-- d715e7c    4904   51      52      0x8f61c8   COMMIT  0
address  flags userthread locks  beginlg curlog  logposit   isol    retrys coord
 d745b58  A-R-- d715e7c    4904   51      52      0x8a1acc   COMMIT  0
```

在一秒内逻辑日志位置从“ 0x8f61c8 ”变化到“ 0x8a1acc”这其中页的变化时“0x8f6”到“0x8a1”,用0x8f6-0x8a1=0x55=85个页 85*4KB=340KB,也就是说一秒钟内回滚了340KB，我们得到了回滚速度是340KB/s 相应的当前的回滚时间即等于0x8a1*4/340=26秒

4、如何处理和解决长事务

作为DBA我们知道了长事务的类型、如何监控长事务回滚、如何计算回滚速度后，我们就需要知道如何处理长事务

1、应该是从应用角度考虑，尽量避免长事务的出现，尽量将批量操作分成小的事务实现；
2、在“onconfig”中辩证的设置合理长事务参数 “ LTXHWM”和“LTXEHWM”；
3、根据业务情况，设置合理的逻辑日志的大小和数量，设置合理的逻辑日志空间；
4、必要时，可以合理设置“OFF_RECVRY_THREADS”通过“fast recovery”来加快长事务回滚；
5、“备份”恢复，备份对与DBA来说是“重中之重”；
6、Informix 800技术支持的内部工具，进行日志截取

备注：这是一种非常规的手段，在IDS9.40使用fuzzy ckpt后，无法估计损失的数据情况， 有时对数据库一致性存在负作用，存在相当高的风险，不到万不得已不建议采用；

5、分布式事务导致数据库阻塞的解决方法

有时候Informix数据库发生长事务阻塞数据库的情况是由于分布式事务导致的，此时由于事务不是由本机发起，使用常规的方法很难将导致长事务的会话解开，数据库不能及时的进行回滚，有时会造成数据库和生产的长时间的阻塞。

问题描述：

分布式事务引发Informix数据库阻塞的表现如下：

1、“onstat -” 会有如下的示例提示信息

```
IBM Informix Dynamic Server Version 9.40.FC2W3   -- On-Line (LONGTX) -- Up
16:58:50 -- 2748536 Kbyts  Blocked:LONGTX
```

2、"onstat -m"中，会有如下示例错误信息出现：

```
19:49:23  Aborting Long Transaction: tx 0xc000000089d56ce8 no user info due to
XA or distributed (2n
19:49:23  Aborting Long Transaction: tx 0xc000000089d56f48 no user info due to
XA or distributed (2n
19:49:23  Session completed abnormally. Rolling back tx id 79, flags 0x108463b
19:49:23  Session completed abnormally. Rolling back tx id 80, flags 0x108463b
```

检查方法：

1、执行"onstat -x" 检查标志位信息“--H-G”，看是否与上述online.log 中相应的全局事务的地址信息一致，例如：

```
Transactions
address       flags userthread       locks  beginlg curlog  logposit   isol    retrys coord
c000000089d56ce8 --H-G 0              0      10150   10154   0x4038
COMMIT  0
c000000089d56f48 --H-G 0              0      10150   10154   0x3080
COMMIT  0
```

2、执行“onstat -G”,显示当前系统中所有的分布式事务信息：

Global Transaction Identifiers
address flags fID gtl bql data
c000000089d56ce8 0x10806ba 48801 10 13
3346845D147D57B372BD43415053636F72657358415257
c000000089d56f48 0x10806ba 48801 10 13
3338845D147D57B372BD43415053636F72657358415257
2 active, 384 total

解决方法：

1、可以尝试使用"onmode -H" 强行解除全局事务，然后反复执行onstat -, onstat -G 和 onstat -x命令，确认相应的
全局事务已经解除，例如：

onmode -H 0xc000000089d56ce8
onmode -H 0xc000000089d56f48

2、使用"onmode -Z"强行解除全局事务，例如
使用onstat -x命令发现有全局事务

IBM Informix Dynamic Server Version 11.50.FC4 -- On-Line (LONGTX) -- Up 09:31:54 -- 100000 Kbytes
Blocked:LONGTX

Transactions
address flags userthread locks beginlg curlog logposit isol retrys coord
...
70769468 --H-S 0 0 179 212 0x334015c COMMIT 8 prod3

使用“onmode -Z 70769468”尝试解除相应的事物


THANK YOU