DB2锁问题处理最佳实践

徐明伟

北京普远天成科技有限公司 技术总监

议题

1. DB2锁概述
2. DB2锁问题监控和定位
3. DB2锁问题调优
4. DB2 9.7锁机制深入分析
5. DB2锁案例分享

为什么需要锁

- 一致性机制
  - 事务
  - 日志
  - 锁
  - 隔离级别

- 锁
  - 维护数据一致性
  - 控制并发性

- 锁分类
  - 锁的对象（表、行、表空间、索引）
  - 锁的模式（S,X等）

锁导致问题/锁现象

- 锁的几种现象
  - 锁等待
  - 锁超时
  - 死锁
  - 锁升级
  - 锁转换

- 锁产生的问题
  - 系统运行慢
  - 应用回滚

议题

1. DB2锁概述
2. **DB2锁问题监控和定位**
3. DB2锁问题调优
4. DB2 9.7锁机制深入分析
5. DB2锁案例分享

锁问题监控和定位

- 锁问题监控定位工具
  - ✓Snapshot快照
  - ✓deadlock event monitor with details history
  - ✓db2pd (8.2后)
  - ✓db2pdcfg (9.1)
  - ✓db2_capture_locktimeout (9.5)
  - ✓New Locking event monitor (9.7)

- 锁是症状，不是根源

锁快照监控

- 通过get snapshot for database on <db> 或sysibmadm.snapdb
  - 死锁、锁等、锁超时、锁升级等统计信息

Database Snapshot

Database name	= CRMDB
First database connect timestamp	= 10/10/2011 20:50:28.421301
Snapshot timestamp	= 03/07/2012 11:10:08.120847
Locks held currently	= 1
Lock waits	= 471967
Time database waited on locks (ms)	= 2039693347
Lock list memory in use (Bytes)	= 1740480
Deadlocks detected	= 470
Lock escalations	= 0
Exclusive lock escalations	= 0
Agents currently waiting on locks	= 0
Lock Timeouts	= 24782

数据库连接时间

快照时间

锁等数量

发生锁等的时间

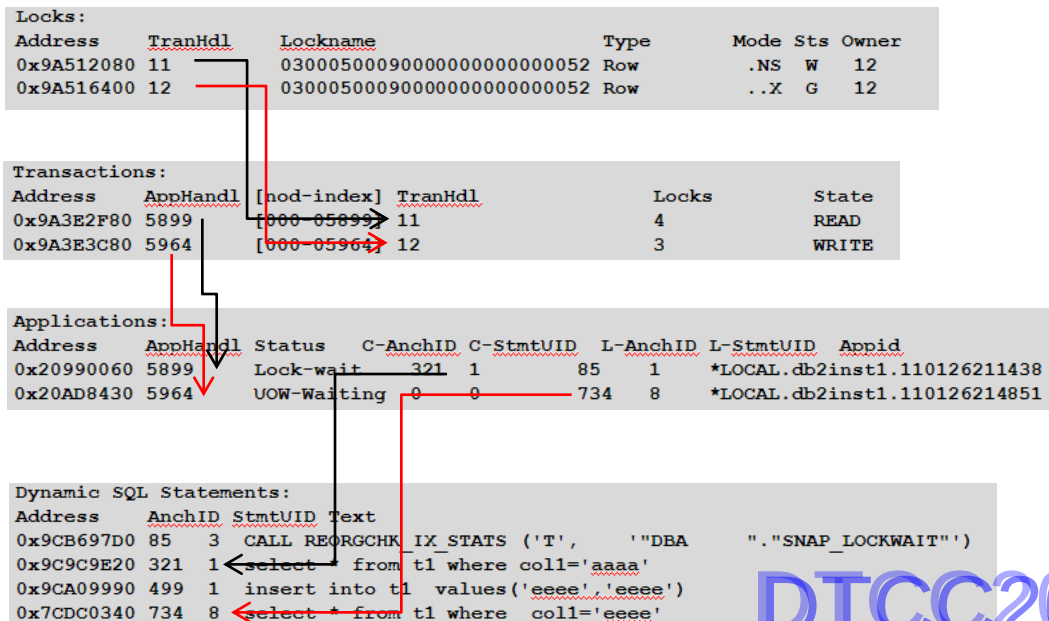
死锁数量

锁升级数量

锁超时数量

db2pd监控锁等

- db2pd –d <db_name> -locks wait –tra –app –dyn
  - 定位引起锁等的事务、应用和动态语句



db2pdcfg捕获锁超时 (9.1版本)

- db2pd结合db2cos回调脚本捕获锁超时或死锁
- 改写db2cos回调脚本

```
"LOCKTIMEOUT")
    echo "Lock Timeout Caught"                    >> $logfile
    if [ ! -n "$database" ]
    then
      db2pd -inst                                 >> $logfile
    else
      db2pd -db $database -locks -tra -app -dyn   >> $logfile
    fi
 ;;
```

- db2pdcfg –catch locktimeout count=1
- 当发生锁超时时，会调用db2cos脚本

db2_capture_locktimeout捕获锁超时(9.1 fp4-9.5)

- 设置db2_capture_locktimeout注册变量
  - db2set DB2_CAPTURE_LOCKTIMEOUT=ON
- 创建死锁事件监控器
  - db2 “create event monitor dlockevm for deadlocks with details history write to file ‘/home/db2inst1/locks’ ”

```
Lock Requestor:                 --锁请求者相关信息
  System Auth ID:               DB2INST1
  Application Handle:           [0-54]
  Application ID:               *LOCAL.db2inst1.110128054807
  Application Name:             db2bp
  Requesting Agent ID:          72
  Coordinator Agent ID:         72
  Coordinator Partition:        0
  Lock timeout Value:           15000 milliseconds
  Lock mode requested:          .NS
  Application Status:           (SQLM_UOWEXEC)
  Current Operation:            (SQLM_FETCH)
  Lock Escalation:              No

  Context of Lock Request:
  Identification:               UOW ID (1); Activity ID (2)
  Activity Information:
     Package Schema:            (NULLID  )
     Package Name:              (SQLC2H20NULLID  )
     Package Version:           ()
     Section Entry Number:      201
     SQL Type:                  Dynamic
     Statement Type:            DML, Select (blockable)
     Effective Isolation:       Cursor Stability
     Statement Unicode Flag:    No
     Statement:                 select * from t1 where col1='xxxx'
```

锁请求信息

请求锁模式

请求的SQL语句

```
Lock Owner (Representative):   --锁拥有者相关信息
  System Auth ID:           DB2INST1
  Application Handle:       [0-46]
  Application ID:           *LOCAL.db2inst1.110128054759
  Application Name:         db2bp
  Requesting Agent ID:      19
  Coordinator Agent ID:     19
  Coordinator Partition:    0
  Lock mode held:           ..X                          --锁持有模式

  List of Active SQL Statements:  Not available

  List of Inactive SQL Statements from current UOW:

    Entry:                #1
    Identification:             UOW ID (3); Activity ID (2)
    Package Schema:             (NULLID )
    Package Name:               (SQLC2H20)
    Package Version:            ()
    Section Entry Number:       201
    SQL Type:                   Dynamic
    Statement Type:             DML, Select (blockable)
    Effective Isolation:        Cursor Stability
    Statement Unicode Flag: No
    Statement:                  select * from t1 where col1='ffff'  --SQL语句

   Entry:                       #2
   Identification:              UOW ID (3); Activity ID (1)
   Package Schema:              (NULLID )
   Package Name:                (SQLC2H20)
   Package Version:             ()
   Section Entry Number:        203
   SQL Type:                    Dynamic
   Statement Type:              DML, Insert/Update/Delete
   Effective Isolation:         Cursor Stability
   Statement Unicode Flag:      No
   Statement:                   insert into t1 values('ffff','ffff')  --SQL语句
```

锁拥有者信息

当前锁模式

占有锁的SQL语句

Deadlock event monitor监控死锁

- 创建deadlock事件监控器
  - db2 "create event monitor dlockevm for deadlocks with details history write to file '/home/db2inst1/deadlock' “

```
5) Deadlocked Connection ...
  Deadlock ID:   1
  Participant no.: 2
  Participant no. holding the lock: 1
  Appl Id: *LOCAL.db2inst1.110128063201
  Appl Id of connection holding the lock: *LOCAL.db2inst1.110128054759
  Seq. no. of connection holding the lock: 00001
  Lock wait start time: 01/27/2011 22:37:04.601677
  Lock Name        : 0x030005000400000000000000052
  Current Mode     : none
  Deadlock detection time: 01/27/2011 22:37:09.788844
  Table of lock waited on      : T1
  Schema of lock waited on     : DB2INST1
  Type of lock: Row
  Mode of lock: X    - Exclusive
  Mode application requested on lock: NS  - Share (CS/RS)
  Deadlocked Statement:
    Type      : Dynamic
    Operation: Fetch
    Text      : select * from t1

6) Deadlock statement history ...
  Deadlock ID              : 1
  Participant No           : 2
  Stmt history ID          : 2
  Type                     : Dynamic
  Statement text           : select * from t1

7) Deadlock statement history ...
  Deadlock ID              : 1
  Participant No           : 2
  Stmt history ID          : 1
  Type                     : Dynamic
  Statement text           : insert into t2 values('bbb')
```

Deadlock id

死锁的表

表上已有的锁模式

当前请求的锁模式

死锁语句

Deadlock event monitor监控死锁(2)

```
9) Deadlocked Connection ...
  Deadlock ID:   1
  Participant no.: 1
  Participant no. holding the lock: 2
  Appl Id: *LOCAL.db2inst1.110128054759
  Appl Seq number: 00018
  Appl Id of connection holding the lock: *LOCAL.db2inst1.110128063201
  Seq. no. of connection holding the lock: 00001
  Lock wait start time: 01/27/2011 22:37:03.515204
  Lock Name        : 0x0300060005000000000000052
  Current Mode     : none
  Deadlock detection time: 01/27/2011 22:37:09.795276
  Table of lock waited on        : T2
  Schema of lock waited on       : DB2INST1
  Type of lock: Row
  Mode of lock: X    - Exclusive
  Mode application requested on lock: NS  - Share (CS/RS)
  Deadlocked Statement:
    Type      : Dynamic
    Operation: Fetch
    Text      : select * from t2

10) Deadlock statement history ...
  Deadlock ID                    : 1
  Participant No                 : 1
  Stmt history ID                : 2
  Type                           : Dynamic
  Statement text                 : select * from t2

11) Deadlock statement history ...
  Deadlock ID                    : 1
  Participant No                 : 1
  Stmt history ID                : 1
  Type                           : Dynamic
  Statement text                 : insert into t1 values('aaa')
```

Deadlock id

锁的表

请求的锁类型

死锁语句

锁事件监控器(9.7版本)

- 9.7引入了新的锁事件监控器
- 用同一个锁事件监控器就可捕获死锁、锁超时和锁等待
- 采用UE表存取锁事件结果，使得分析更简单

- 锁事件监控器的使用包含三个步骤：
  - 1. 创建锁事件监控器
  - 2. 设置数据收集的类型和级别
  - 3. 格式化并分析数据

创建锁事件监控器

```
create event monitor lockevmon for locking
  write to unformatted event table (table
  locks)
```


```
set event monitor lockevmon state=1
```

- Unformatted Event(UE)表
  - 解释说明

设置锁事件参数

- 捕获死锁事件
  - db2 update db cfg using mon_deadlock hist_and_values
- 捕获锁超时事件
  - db2 update db cfg using mon_locktimeout hist_and_values
- 捕获锁等待事件
  - db2 update db cfg using mon_lw_thresh 5000000
  - db2 update db cfg using mon_lockwait hist_and_values
- 要监控的锁事件信息详细程度:
  - Without_hist ：收集基本事件信息
  - With_hist：一个事务里最多收集250个活动
  - Hist_and_values： 收集活动和值
  - None：不收集事件

格式化锁事件输出结果

- 一旦捕获了锁事件，下一步就要对锁数据进行分析
- 对UE表数据的格式化有三种方法：
  - Db2evmonfmttool
    - ✓产生文本格式输出报告
  - EVMON_FORMAT_UE_TO_XML_UDF函数
    - ✓产生XML格式输出报告
  - EVMON_FORMAT_UE_TO_TABLE存储过程
    - ✓将UE数据格式化成关系表数据
    - ✓分析处理变得简单

死锁举例

Application #1

```
update t2 set name='t2...' where id=200
select * from employee
```

Application #2

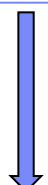
```
update t1 set name='t11...' where id=100
update t2 set name='t2new...' where id=200
```

```
update t1 set name='t222' where id=100 (911 error)
```

模拟一个死锁场景

使用EVMON_FORMAT_UE_TO_TABLE格式化

```
call EVMON_FORMAT_UE_TO_TABLES(
'LOCKING',NULL,NULL,NULL,NULL,NULL,'RECREATE_FO
  RCE',-1, 'select * from locks order by event_timestamp‘
  )
```



LOCK_ACTIVITY_VALUES	INST97	T	2012-04-03-22.22.42.368824
LOCK_EVENT	INST97	T	2012-04-03-22.22.41.600176
LOCK_PARTICIPANTS	INST97	T	2012-04-03-22.22.41.921119
LOCK_PARTICIPANT_ACTIVITIES	INST97	T	2012-04-03-22.22.42.112358

查看死锁和参与者信息

```
Select xmlid, event_id, event_type,
  event_timestamp,member,dl_conns,
  rolled_back_participant_no from lock_event
```

查看锁信息

```
XMLID   EVENT_TYPE EVENT_TIMESTAMP    MEMBER DL_CONNS  ROLLED_BACK_PARTICIPANT_NO
---------- --------------- ---------- -------------------------- ------ ----------- --------------------------
db2LockEvent_4…  DEADLOCK   2012-04-03-21.27.19.660632      0          2                    2
```

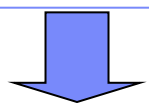
```
Select xmlid, participant_no, participant_type,
  participant_no_holding_lk, application_handle from
  lock_participants where xmlid like db2LockEvent_1%'
```

查看死锁参与者信息

```
XMLID      PARTICIPANT_NO PARTICIPANT_TYPE PARTICIPANT_NO_HOLDING_LK
   APPLICATION_HANDLE
---------- -------------- ---------------- ------------------------- --------------------
db2LockEvent_4…  1 Requester                    2                    7
db2LockEvent_4…  2 Requester                    1                    60
```

查看死锁参与者SQL活动信息

```
select activity_id, activity_type, uow_id, stmt_type,
   substr(stmt_text,1,100)as stmt_text
from lock_participant_activities
where xmlid like 'db2LockEvent_4%'
```



ACTIVITY_ID	ACTIVITY_TYPE	UOW_ID	STMT_TYPE	STMT_TEXT
1	past	4	2	update t1 set name='t11...' where id=100
2	current	4	2	update t2 set name='t2new...' where id=200
1	past	1	2	update t2 set name='t2...' where id=200
2	past	1	2	select * from employee
3	current	1	2	update t1 set name='t222' where id=100

议题

1. DB2锁概述
2. DB2锁问题监控和定位
3. **DB2锁问题调优**
4. DB2 9.7锁机制深入分析
5. DB2锁案例分享

性能调优关键

- 锁是症状，SQL是问题的根源
- I/O 最关键
  - 减少I/O
  - 最大化I/O效率
  - 存储规划
  - 物理设计
- CPU 2大杀手
  - 表扫描
  - 排序
- Memory命中率可能会骗人

锁的调优(1) — 应用层

- 写优秀的SQL语句
- 创建合适的索引避免表扫描
- 选择合适的隔离级别: UR,CS(CC),RS,RR
- 事务尽可能频繁的提交
- 在事务结尾执行insert/update/delete

锁的调优(2) —数据库层

- 调优locklist 和maxlocks数据库参数
  - 9.1后设为automatic
- Locktimeout 锁超时参数
  - OLTP系统, 建议设置为15-30秒
  - DW系统,建议60-120秒
- 锁延迟设计考虑
  - db2set DB2_EVALUNCOMMITTED=ON
  - db2set DB2_SKIPDELETED=ON
  - db2set DB2_SKIPINSERTED=ON
- 9.7 当前已提交读特性(currently committed)
- 数据表维护
  - Runstats更新统计信息
  - Reorg重组表和索引碎片
  - Rebind重新绑定包，更新执行计划

写好的SQL语句

- 只返回需要的行，避免用select * from t1
- 加过滤条件限制返回的行数
- 避免笛卡尔乘积，select * from a,b

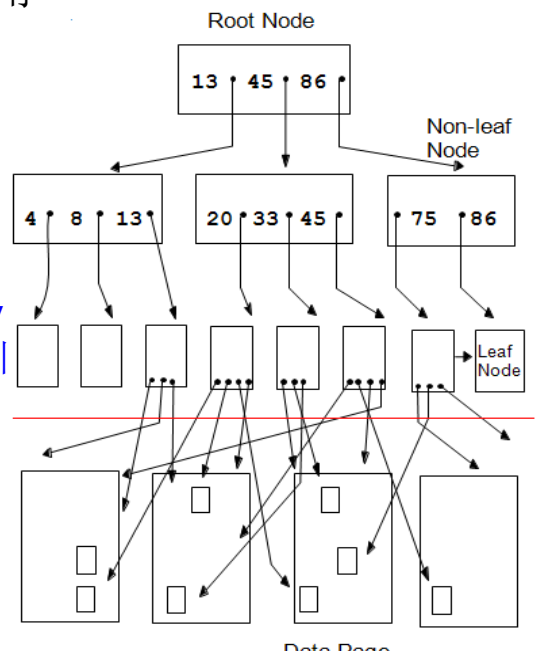
- 使用参数化查询，where col1=?，减少编译时间
  - 案例分析
- 避免对查询条件计算，where salary*2>xx 改为salary > xx/2
  - 无法利用索引

- 使用for read only或for fetch only
- 使用for update of
- 避免数字类型转换
- 避免数据类型不匹配
- 如果可能，尽量避免使用order by和distinct
- 尽量使用exists而不是用in
- 函数的效率很高，充分利用

创建索引

- 索引对于提高SQL读有无可替代作用
  - 提高查询性能
  - 避免不必要的表扫描
  - 减少排序、减少锁
  - 减少CPU和I/O使用

- 索引创建最佳实践
  - 为where查询条件、Sort排序(order by 、max()、min ()等）、join谓词创建索引
  - 分析组合索引键的顺序
    - (a,b), (b,a) 完全不同
  - 不要创建冗余索引
    - (a), (a,b)　(a)为冗余索引
  - 确保索引被用到
  - 创建cluster 索引
  - 通过include语句创建index-only索引
  - 对于不稳定数据，可考虑用volatile强制走索引
  - 推荐使用db2advis建议索引



物理设计

- 海量数据库物理设计
  - 索引
  - 多维索引
  - 物化视图
  - 分区表
  - 压缩
  - 数据库分区

- 数据归档
  - 减少数据，减少I/O等访问

议题

1. DB2锁概述
2. DB2锁问题监控和定位
3. DB2锁问题调优
4. **DB2 9.7锁机制深入分析**
5. DB2锁案例分享

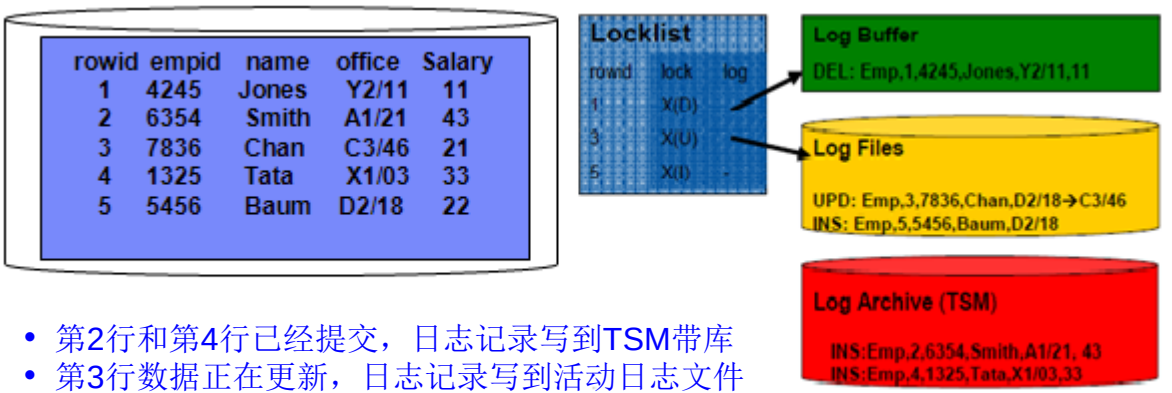
隔离级别(isolation level)

- 隔离级别控制锁的范围和粒度，隔离级别只适用于读

DB2 Isolation	ANSI Isolation	Dirty Write	Dirty Read	Fuzzy Read	Phantom Read
Uncommitted Read (UR)	Read Uncommitted (Level 0)	✗	✓	✓	✓
Cursor Stability (CS)	Read Committed (Level 1)	✗	✗	✓	✓
Read Stability (RS)	Repeatable Read (Level 2)	✗	✗	✗	✓
Repeatable Read (RR)	Serializable (Level 3)	✗	✗	✗	✗

- DB2 9.7版本引入了currently committed机制
  - ✓ 如果发现未提交的行，则使用当前已经提交的数据(before image)
  - ✓ 写不阻碍读，减少锁等和死锁，超时等
  - ✓ 在locklist锁信息中加入标识，当另外事务读取数据时基于标识判断
  - ✓ 通过cur_commit数据库参数控制

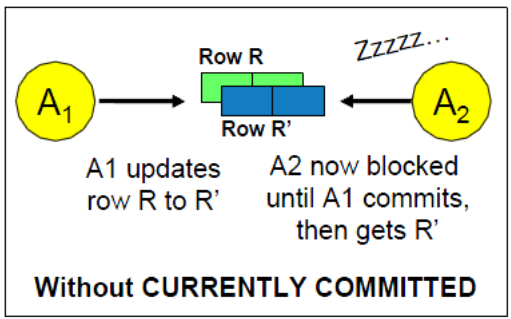
Currently Commited机制

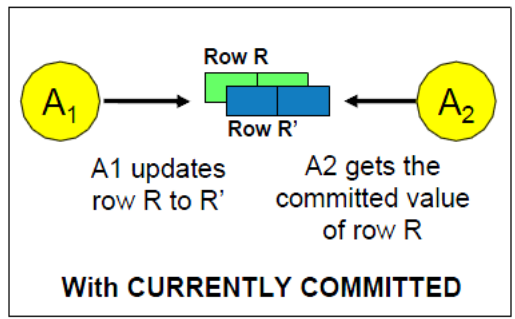


- 第2行和第4行已经提交，日志记录写到TSM带库
- 第3行数据正在更新，日志记录写到活动日志文件
- 第5行数据正在插入，日志记录写到活动日志文件
- 第1行数据正在删除，日志记录在Log buffer内存

- 当启用了CC机制，DB2会在每个行锁上增加一个标识：
  - No information：表示记录已经加锁。
  - Uncommitted insert identifier：表示这行是新插入未提交行。
  - Log information：表示这行没有提交，包含了当前已落实数据的日志LSN。

Currently Commited性能影响





- 使用currently committed的tips
  - 如果观察到大量读写锁等时间，使用cc更好
  - 严重读写竞争的应用会有5-10%以上的性能提升
  - 建议调整logbufsz为2048，9.7缺省是256页

议题

1. DB2锁概述
2. DB2锁问题监控和定位
3. DB2锁问题调优
4. DB2 9.7锁机制深入分析
5. DB2锁案例分享

案例1：某通信行业电子运维系统

- 问题概述：
  - 全省统一的电子运维管理系统
  - 1000多个并发，高峰期业务无法处理
  - 大量锁等，响应慢，CPU使用率80%以上

- 数据库监控：

First database connect timestamp	= 10/10/2011 20:50:28.421301
Snapshot timestamp	= 11/09/2011 17:19:31.409497
Number of Threshold Violations	= 0
Locks held currently	= 4
Lock waits	= 86248
Time database waited on locks (ms)	= 1306986473
Lock list memory in use (Bytes)	= 2186752
Deadlocks detected	= 222
Lock escalations	= 0
Exclusive lock escalations	= 0
Agents currently waiting on locks	= 0
Lock Timeouts	= 15560

First database connect timestamp	= 11/10/2011 20:09:23.982
Snapshot timestamp	= 12/11/2011 18:32:05.902
Number of Threshold Violations	= 0
Locks held currently	= 4
Lock waits	= 9429
Time database waited on locks (ms)	= 183176718
Lock list memory in use (Bytes)	= 198742
Deadlocks detected	= 42
Lock escalations	= 0
Exclusive lock escalations	= 0
Agents currently waiting on locks	= 0
Lock Timeouts	=2305

- 解决办法：
  - 通过db2pd找到占有锁的SQL语句，优化
  - 调优后，死锁和锁超时数据降低了80%，性能大大提升

案例2：北京某部委优化项目

- 问题概述：
  - 北京市人口统计信息统一管理系统
  - 平常800个并发，高峰1000多个，高峰期查询需要10分钟
  - 大量锁等，响应慢，CPU利用率低

- 数据库监控：

First database connect timestamp	= 01/16/2012 19:23:39.036989
Snapshot timestamp	= 02/17/2012 13:45:47.147392
Locks held currently	= 263
Lock waits	= 2772575
Time database waited on locks (ms)	= 9986324507
Lock list memory in use (Bytes)	= 49920
Deadlocks detected	= 4
Lock escalations	= 0
Exclusive lock escalations	= 0
Agents currently waiting on locks	= 1
Lock Timeouts	= 0

说明：

平均每天锁等达到 90万个
平均每个锁等等待 3.6秒
累计锁等时间达到 115天

锁超时次数为0？

- 解决办法：
  - 调优bufferpool，从1.5G到25G
  - 调优了5条SQL语句
  - 调优后，性能大幅度提升，最慢的响应时间从10分钟降为秒级

案例3：某银行核心交易死锁问题处理(1)

- 问题概述：
  - 核心银行交易系统
  - 12月25日调优后，性能下降严重，死锁数量急剧增加

- 数据库监控：

First database connect timestamp	= 12/29/2011 00:13:12.846036
Snapshot timestamp	= 12/30/2011 11:50:17.975290
Locks held currently	= 263
Lock waits	= 62245
Time database waited on locks (ms)	= 10514064
Lock list memory in use (Bytes)	= 254720
Deadlocks detected	= 531
Lock escalations	= 0
Exclusive lock escalations	= 0
Agents currently waiting on locks	= 0
Lock Timeouts	=10

说明：

1天半时间发生了**531**个死锁

- 分析：
  - 应用是通过sqlc开发，嵌入式静态语句
  - 创建deadlock event monitor，发现有一条语句存在重大嫌疑
  - 该语句访问的表数据量大概20万行，单条语句执行速度很快，大概2秒

案例3：某银行核心交易死锁问题处理(2)

- 通过db2expln查看执行计划，发现该语句cost高达9645

```
Statement:
  DECLARE BIGAMT_CUR CURSOR
  FOR
    SELECT *
    FROM dpsbigamttrans
    WHERE trandate =:H01057 AND acctno =:H01069 AND subacct
=:H01071
    WITH UR

Statement Isolation Level = Uncommitted Read

Section Code Page = 1386

Estimated Cost = 9645.297852
Estimated Cardinality = 0.000002
```

```
Access Table Name = VBSRUN.DPSBIGAMTTRANS  ID
= 3,3337
| #Columns = 26
| Relation Scan
| | Prefetch: Eligible
| Lock Intents
| | Table: Intent Share
| | Row  : Next Key Share
| Sargable Predicate(s)
| | #Predicates = 3
| | Return Data to Application
| | | #Columns = 26
Return Data Completion

End of section


Optimizer Plan:

    RETURN
    (   1)
      |
    TBSCAN
    (   2)
      |
Table:
VBSRUN
DPSBIGAMTTRANS
```

- 解决办法：
  - 创建索引
  - Runstats
  - Rebind package
  - 锁数量大大降低

联系方式

- 徐明伟
  - 资深DB2顾问（咨询、培训、服务支持）
  - 手机：13701141650
  - Email：xumwdb2@163.com
  - qq: 907342263
  - 新浪微博：徐明伟_db2咨询

- 取得成绩：
  - 2012年“IBM DB2迁移之星大赛”冠军团队
  - 2011年，出版《DB2数据库管理最佳实践》书籍
  - 2011年“IBM DB2十大人物”
  - 2010年“IBM软件技术精英年度成就会员”

DTCC2012