●원작: 김 규복 · 카와무카이이치로(EXEM Japan)
●번역: 고 혜원(EXEM QA 팀)

SPECIAL FEATURE 2

PART 1

Oracle 내부를 분석하여 성능개선에 연결시키는 OWI의 기초

통계정보로부터 AP처리의 병목을 읽어내다.

실제 환경에서 DB시스템의 성능 문제와 만나게 될 때, 현장에서는 CPU와 메모리의 사용상태, I/O상태, 네트워크 부하를 확인하는 것이 일반적이다. 이것들을 파악함으로써 문제발생 시점에서의 전체의 대략적인 상태가 알 수 있고 어느 정도의 추측이 가능하게 된다.

성능문제에 의한 병목현상은 50% 이상이 DB 또는 Application에서 발생하고 있다고 하며 특히, 대용량에 다수의 사용자를 기반으로 한 요즘의 시스템에서는 그 경향이 보다 강해지고 있다. 그렇기 때문에 성능장애가 발생한 때에는 DB시스템 환경의 개요정보는 물론, DB내부의 성능통계정보를 참조해야 할 필요가 있다. 수집된 Data가 없을 경우에는 재실행이나 재현 Test를 통해 STATSPACK이나 SQL Trace 파일을 취득하는 것이 현장의 실정이지 않을까 싶다.

본 기고에서는 현장에서 자주 발생하는 패턴의 성능문제를 간단히 한 테스트 시나리오(환경: Windows 2003서버, Oracle 10.2.0.3)에서 성능분석에 자주 사용되는 STATSPACK 보고서와 Trace 요약(보기 쉽게 일부를 편집 함)에서 자주 접하게 되는 Oracle Wait Interface(이하 OWI, 상세한 내용은 추후 서술)의 시간통계정보를 어떻게 이해해야 하는지를 기술하고자 한다.

먼저 지금부터 예로 드는 리포트를 보고 Oracle 내부의 처리가 늦어졌는지 어떤지 생각해 보자. 성능 저하가 발생한 경우에는 장애의 발생지점 및 원인, 그리고 개선포인트를 같이 검토하길 바란다. 또한 지금부터는 사고방식의 힌트를 표시하는 것으로 하며 자세한 해설과 개선책은 본 Part의 후반에서 다루기로 한다.

케이스 1 Library Cache(LIST 1)

CPU 사용시간의 1.5배에 이르는 Waits이 발생하고 있다. 특히, 대기가 가장 긴 Wait으로써 'latch: library cache' 이벤트가 발견되고 있는 점을 통해, Library Cache에서의 SQL 파싱 또는 Object 참조에서 병목이 발생하고 있다고 생각할 수 있다.

케이스 2 Data File 읽기작업(LIST 2)

시간 관점에서 보면 대부분의 처리시간이 이벤트에 의한 지연으로 시간이 소요되고 있다. 특히 'db file sequential read' 이벤트에 의한 지연이 대부분을 차지하고 있는 상황을 통해 Disk에 대한 Single Block 읽기 처리시의 경합에 의해 병목이 발생하고 있다고 생각할 수 있다. Data처리에 있어서 Disk I/O는 피할 수 없기 때문에 I/O관련 이벤트의 발생 자체를 성능저하 현상이라고 간주할 수는 없지만, 이번 결과처럼 대기시간이 다른 처리 시간에 비해서 현저히 긴 경우 그 발생 원인에 대해서 진단/분석할 필요가 있다.

케이스 3 Buffer Cache의 검색작업 (LIST 3)

CPU 사용시간의 5배 이상이 대기상태가 되고 있다. 특히 'latch: cache buffers chains' 이벤트에 의한 지연이 대부분을 차지 하고 있는 점에서 Buffer Cache에 대한 검색작업에서 병목이 발생하고 있다고 생각할 수 있다.

케이스 4 Buffer Block에 대한 경합 (LIST 4)

상위 대기시간의 합계가 CPU 사용시간을 18배 이상 상회하고 있다.

특히 'buffer busy waits'와 'log file sync' 이벤트에 따른 대기가 대부분을 차지하고 있는 점에서, 동일 block에 대한 경합과 REDO Log File에의 쓰기작업에서 병목현상이 발생하고 있다고 생각할 수 있다. 먼저 진단 해야 할 포인트를

Top 5 Timed Events (Library cache로 인한 병목현상!)

Event	Waits	Time (s)	Avg wait (ms)	%Total Call Time
CPU time		1,055		34.7
latch: library cache	7,278	750	103	24.7
latch: library cache lock	4,194	465	111	15.3
job scheduler coordinator slave wait	23	371	16124	12.2
cursor: pin S wait on X	2,019	206	102	6.8

LIST 1 케이스 1의 현상

Top 5 Timed Events (Data File의 읽기작업이 최대의 병목현상!)

Event	Waits	Time (s)	Avg wait (ms)	%Total Call Time
db file sequential read	15,077	182	12	90.8
CPU time		13		6.4
control file sequential read	366	2	5	1.0
log file parallel write	71	1	14	.5
control file parallel write	99	1	9	.4

LIST2 케이스 2의 현상

OWI에 의한 Oracle 성능개선과 장애대책

LIST UP하여 개선 효과를 예측해 보자.

케이스 5 Row Lock Wait(LIST 5)

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                               Waits   Time (s)  (ms)   Time
--------------------------------------------- ------------ ---------- ------ ------
latch: cache buffers chains                         10,587      2,091    197   67.2
CPU time                                                           460          14.8
latch free                                           1,994        322    161   10.4
job scheduler coordinator slave wait                    11        179  16307    5.8
read by other session                               13,009         35      3    1.1
```

Buffer Cache의 검색작업에서 병목현상!

LIST3 케이스 3의 현상

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                               Waits   Time (s)  (ms)   Time
--------------------------------------------- ------------ ---------- ------ ------
buffer busy waits                                   43,921      1,727     39   44.8
log file sync                                       43,096      1,107     26   28.7
log file switch (checkpoint incomplete)                355        283    797    7.3
CPU time                                                           175          4.5
job scheduler coordinator slave wait                     8        128  16010    3.3
```

Buffer Log에 의한 경합!

LIST4 케이스 4의 현상

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                               Waits   Time (s)  (ms)   Time
--------------------------------------------- ------------ ---------- ------ ------
enq: TX - row lock contention                        1,784      4,699   2634  93.0
PL/SQL lock timer                                    4,658        319     68   6.3
CPU time                                                            22          .4
log file switch completion                               6          5    814   .1
db file sequential read                                375          3      8   .1
```

Row Lock Wait 이 TOP!

LIST5 케이스 5의 현상

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                               Waits   Time (s)  (ms)   Time
--------------------------------------------- ------------ ---------- ------ ------
log buffer space                                    12,143      2,773   228   52.7
buffer busy waits                                    4,101        737   180   14.0
enq: HW - contention                                 2,405        429   178    8.2
log file switch completion                           1,147        422   368    8.0
log file parallel write                              1,143        160   140    3.0
```

REDO Log Buffer에서의 영역경합!

LIST6 케이스 6의 현상

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse     1149      0.28      38.39          0          0          0           0
Execute 3641058   729.93   12967.91         82  124576169   11813911     1822467
Fetch      387      0.01       0.00          0        228          0         330
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total   3642594   730.23   13006.31         82  124576397   11813911     1822797

Elapsed times inclu                                ents:
  Event waited on                                   Times   Max. Wait  Total Waited
  ---------------------------------------------   Waited  ----------  ------------
  log file sync                                       4163        5.35       486.24
  latch: enqueue hash chains                          2737        0.57       292.84
  latch free                                          1539        0.99       283.74
  latch: cache buffers lru chain                      3046        0.57       279.13
  log file switch completion                           397        1.05       219.50
  latch: cache buffers chains                         1974        0.84       210.64
  buffer busy waits                                   2679        1.32       165.21
```

REDO Log File의 동기화 처리로 인한 병목현상!

LIST7 케이스 7의 현상

CPU를 거의 사용하지 않고 대부분의 세션이 DB 내부에서 대기상태에 빠져 있다. Lock 중에서도 'enq: TX row lock contention' 이벤트에 의한 대기현상인 점에서 Data의 갱신/추가 작업에서 발생한 경합에 따른 병목현상으로 생각할 수 있다.

케이스 6 REDO Log Buffer에서의 영역경합 (LIST 6)

CPU 사용시간이 상위에 들지 못할 만큼 대기 이벤트에 대부분의 처리 시간이 사용되고 있다. 특히 'log buffer space'와 'buffer busy waits' 이벤트에 따른 대기가 대부분을 차지한다는 점에서 REDO Log Buffer에의 쓰기작업과 동일 Block에 대한 경합으로 병목이 발생하고 있다고 생각할 수 있다. 먼저, 진단 포인트를 LIST UP하여 개선 효과를 예측해 보자.

케이스 7 REDO Log File의 동기화 처리 (LIST 7)

CPU 사용시간보다 2배 이상의 대기가 발생하고 있다. 특히 TOP Wait으로서 'log file sync' 이벤트가 기록되고 있는 점에서 REDO Log File과의 동기화 작업에서 병목이 발생하고 있다고 생각할 수 있다.
먼저 해당 지연현상을 해결하고, 상태의 변화에 따라 그 다음 방법을 생각하면 좋을 것이다.

케이스 8 Sequence에 대한 Lock Wait (LIST 8)

상위 대기 시간의 합계가 CPU 사용시간의 3배를 상회하고 있다. 특히 'enq: SQ-contention' 이벤트에 따른 지연이 두드러지고 있는 점에서 Sequence의 취득 작업 상의 Lock에 의한 지연현상이라고 생각할 수 있다. 그 이외의 대기현상은 보이지 않지만 TOP 이벤트에 의해 감춰져 있을 가능성도 있기 때문에 TOP 지연현상이 해소되었을 때 표출될 가능성도 있다.

케이스 9 DB Link를 경유로 한 Data의 전송 (LIST 9)

시간을 기준으로, 대부분의 처리 시간이

'SQL*Net message from dblink' 이벤트에 의해 소요되고 있는 점에서 DB Link를 경유하는 Data 전송 작업이 병목이 되고 있다고 생각할 수 있다. 그 이외의 대기 현상은 보이지 않지만, TOP 이벤트에 의해 감춰져 있을 가능성도 있기 때문에 TOP 지연현상이 해결되면 밖으로 표출될 수도 있다.

Oracle진단/분석 방법론으로써의 OWI

소요시간 및 응답시간은 DB뿐만 아니라 Application이나 네트워크 계층서도 성능의 측정/검증/진단/분석에 있어서 가장 중요한 지표가 된다. 앞서 몇 개의 사례를 STATSPACK 리포트와 TRACE 파일의 요약을 통해 확인 해 보았지만 공통적으로 진단/분석의 열쇠가 되는 것은 시간 기반(Time based)의 정보였다.

지금까지는 별로 중요시 되지 않았을지도 모르지만, Oracle은 항시 Time을 기반으로 한 통계정보를 OWI라는 방법으로 제공해 왔고 성능 진단/분석을 위한 실마리 역할이 되어 왔다. 여기서부터는 OWI을 보다 이해하고, 보다 효율적으로 활용하는 것을 목표로 새로이 OWI의 개념을 소개하고자 한다.

OWI란?

Oracle은, User로부터의 요청을 처리하는 과정에서 필요한 리소스를 획득하지 못한 경우에 그 리소스가 사용가능 해 질 때까지 대기 상태에 들어간다. (그림1)의 예를 들어 앞서 살펴 본 케이스 5에서처럼 특정 Data를 갱신하기 위해서는 해당 레코드에 대한 Row Lock 'enq: TX row lock contention'을 획득하기까지 대기하지 않으면 안 된다. 이 사례에서는 '4699'초 대기해서 '20'초가 실질적으로 갱신작업을 수행하는데 소요되었다.

이처럼 어떤 프로세스가 어떤 리소스에 대하여 어느 정도의 대기 상태였는지, 실질적인 처리에 어느 정도 소요 되었는지 등의 통계정보를 기록하여 추후에라도 확인 가능한 수법을 제공하는 일련의 기능과 인터페이스, 그리고 진단/분석 방법을 통틀어 OWI(Oracle Wait Interface)라 부른다. 또 리소스를 획득하기까지 대기하는 상태를 'Wait Event(이벤트를 대기한다)'라고 한다.

OWI는 시간을 기준으로 한 진단/분석을 제공 해, CPU 처리시간 또는 대기시간을 최소한으로 튜닝하기 위한 힌트를 제시한다. 그 튜닝 결과는 아래와 같이 User가 느끼는 처리 시간(응답시간)의 개선으로 그대로 연결된다.

처리시간(응답시간)
=서비스시간(CPU사용시간)+대기시간.

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse   151721      2.70       5.77          0          8          0           0
Execute 3151954   397.14    1870.02          1     284020     613423      153627
Fetch   3000795   163.75    2062.03         43      10227     168477     3001986
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total   6304470                               44     294255     781900     3155613

  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  enq: SQ - contention                       348423        1.63       1848.59
  log file switch completion                      8        0.99          2.48
  latch: library cache                          896        0.06          1.61
  enq: TX - row lock contention                  32        0.26          1.54
  latch free                                   1112        0.02          1.54
```

(Sequence에 의한 Lock 대기!)

LIST8 케이스 8의 현상

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse      131      0.64       9.46          0         40         10           0
Execute  30132     58.43    1464.62          0       7020        468         262
Fetch    30062     63.81     637.83          0         67          0       30050
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    60325    122.89    2111.92          0       7127        478       30312

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message from dblink                 60082        1.00       1004.46
  single-task message                             9        1.17          6.14
  SQL*Net message to dblink                   60082        0.00          0.24
```

(DB 링크 경유에 의한 데이터 전송 부화)

LIST9 케이스 9의 현상

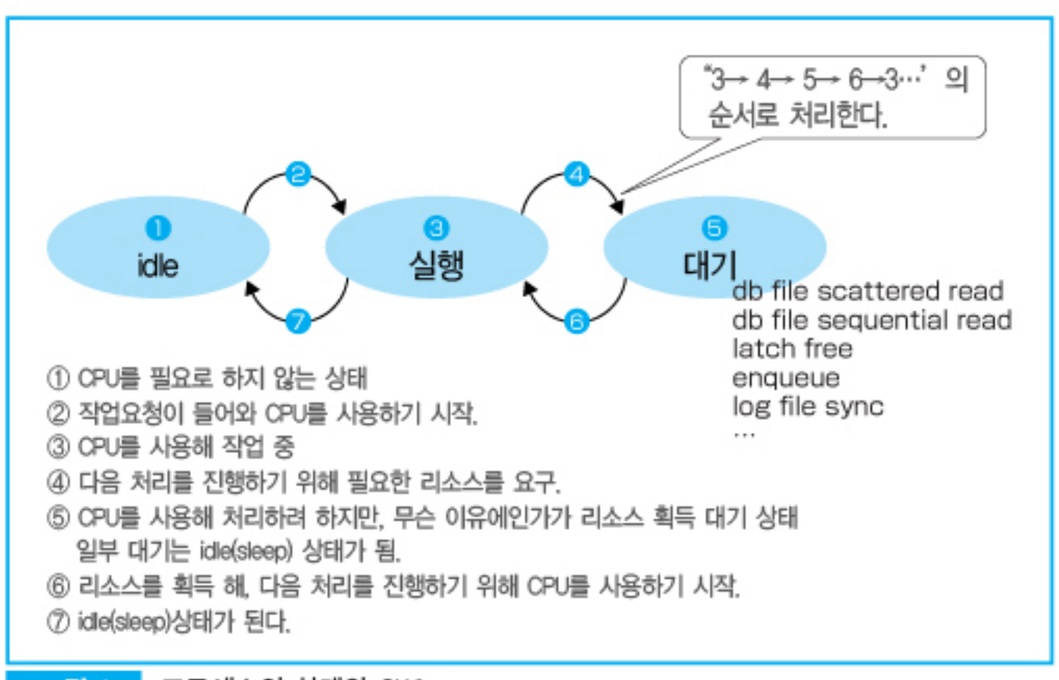


그림 1 프로세스의 상태와 OWI

특집 2

OWI에 의한 Oracle 성능개선과 장애대책

OWI 활용의 메리트

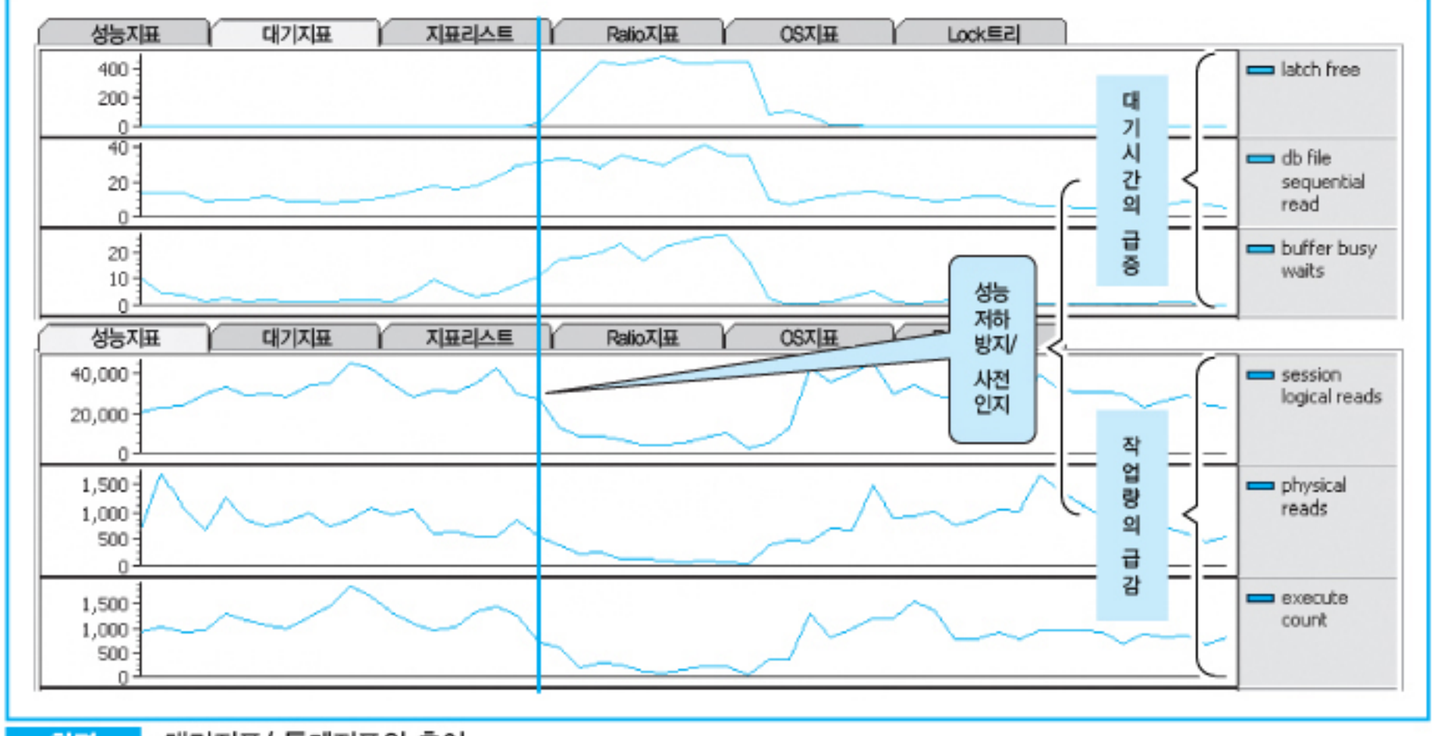
여러분들의 시스템에 있어서 성능문제는 어떤 식으로 발생하고 있는가? 유저로부터 '클레임이 있었다.' '어느 화면에서 1분이 넘도록 결과가 나오지 않는다.' '패치작업이 제한 시간을 넘고 있다……' 라는 기준은 성능문제를 인식하는데 있어서 하나의 중요한 바로미터가 된다. 성능문제는 최적의 타이밍에서 정확히 인식하는 것으로부터 바른 개선안이 나온다.

LIST 10 을 봐주길 바란다. 평균 Buffer Cache Hit Ration가 99.77%이라던가 Memory Sort Ratio(In memory Sort)가 100%라고 하는 정보에서 그 구간에서는 성능문제가 없었다고 판단할 수 있을 것인가? 실제로 이 데이터는 케이스 6에서 사용된 데이터의 일부분이다. Ratio정보는 참고 정도의 지표는 되지만 정확한 성능 진단에 있어서는 별로 도움이 되지 않는다. OWI는 케이스 6의 해설에서처럼 성능문제를 시간을 기준으로 인식할 뿐만 아니라 이벤트의 명칭에서 어떤 부분이 문제인가를 직감적으로 알 수 있도록 한다. 이 때문에 각 대기현상을 해결하는 것으로 어느 정도의 개선 효과가 예상되는지 조차도 그 자리에서 알 수 있는 것이다.

또한 이벤트가 발생하는 구조를 이해(Part Ⅱ에서 추후 설명)함으로써 튜닝포인트의 아이디어도 제공한다. 즉, OWI는 성능문제에 있어서 '인식→진단/분석→튜닝' 의 수단을 정확하고 직감적으로 제공하는 것이다.

```
Instance Efficiency Percentages
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            Buffer Nowait %:   99.45       Redo NoWait %:   99.64
            Buffer  Hit   %:   99.77   In-memory Sort %:  100.00
            Library Hit   %:   98.81      Soft Parse %:   91.64
         Execute to Parse %:   97.33       Latch Hit %:   99.90
Parse CPU to Parse Elapsd %:   38.68      Non-Parse CPU:   96.56
```

LIST 10 STATSPACK 레포트에 의한 'INSTANCE 효율' 예



화면 대기지표/ 통계지표의 추이

사전징후 감시

OWI는 사전징후 감시를 위한 운용을 서포트한다. (그림 2) 는 시스템레벨의 성능지표와 이벤트를 1분 간격으로 취득해서 그 추이를 그래프화 한 화면이다. 어떠한 경향을 읽어낼 수 있는가?

추이를 통해 대기지표(이벤트: latch free, db file sequential read, buffer busy waits)의 대기시간이 붉은 세로선부터 급증하기 시작한 반면, 성능지표(logical reads, physical reads, execution counts)가 급감하기 시작한 것을 알 수 있다. 해당 그래프의 추이는 30분 동안의 변화로, 대기/ 성능지표의 증감을 사전에 감지할 수 있다면 급격한 성능저하는 사전에 예방할 수 있다.

위의 추이에서도 알 수 있듯이 성능문제라는 것은 갑자기 발생하는 현상이 아니다. 1~2분 전에 그 징조가 나타나는 경우도 있고, 몇 달에 걸쳐 서서히

COLUMN

대기 이벤트의 배경

대기 이벤트(Wait Event)는 Oracle7부터 제공되었다. 이 시기는 정확히 Oracle이 엔터프라이즈 환경에서 보급되기 시작한 것과 겹친다. 대용량 데이터 환경이 유발한 동시 접속 수 또는 트랜잭션수의 급증은 그때까지 그다지 문제가 되지 않았던 리소스에 대한 경합을 DB내부에서 급증하도록 하였다. 이로 인해 대기 시간을 측정하는 타이머로써 대기 이벤트가 내부 코드로서 추가되기 시작했다고 생각된다. 이 타이머는 Oracle DB의 버전업 및 기능 확장과 함께 100개 전후로 시작하여 Oracle 10g 에서는 800개(10.2.0.3를 기준으로 876개)를 넘고 있다(그림A).

그 경향에는 enque나 latch의 세분화 등 이벤트의 세밀도를 높이기 위한 방법도 있어 Oracle DB가 보다 자세히 성능 문제를 리포트하고 있는 것을 의미한다.

그러면 차기 버전인 Oracle 11g에서는 어느 정도 증가할 수 있을까?

일본에서는 아직 대기 이벤트에 대한 자료가 보급되지 않은 점도 있고 하여 많은 엔지니어가 아직 유용하게 활용하지 못하고 있는 실정이지만, Oracle의 진화와 함께 그 중요성이 점차 높아져 가고 있는 것은 틀림없다.

그림 A Oracle DB 버전에 따른 OWI 수의 증가

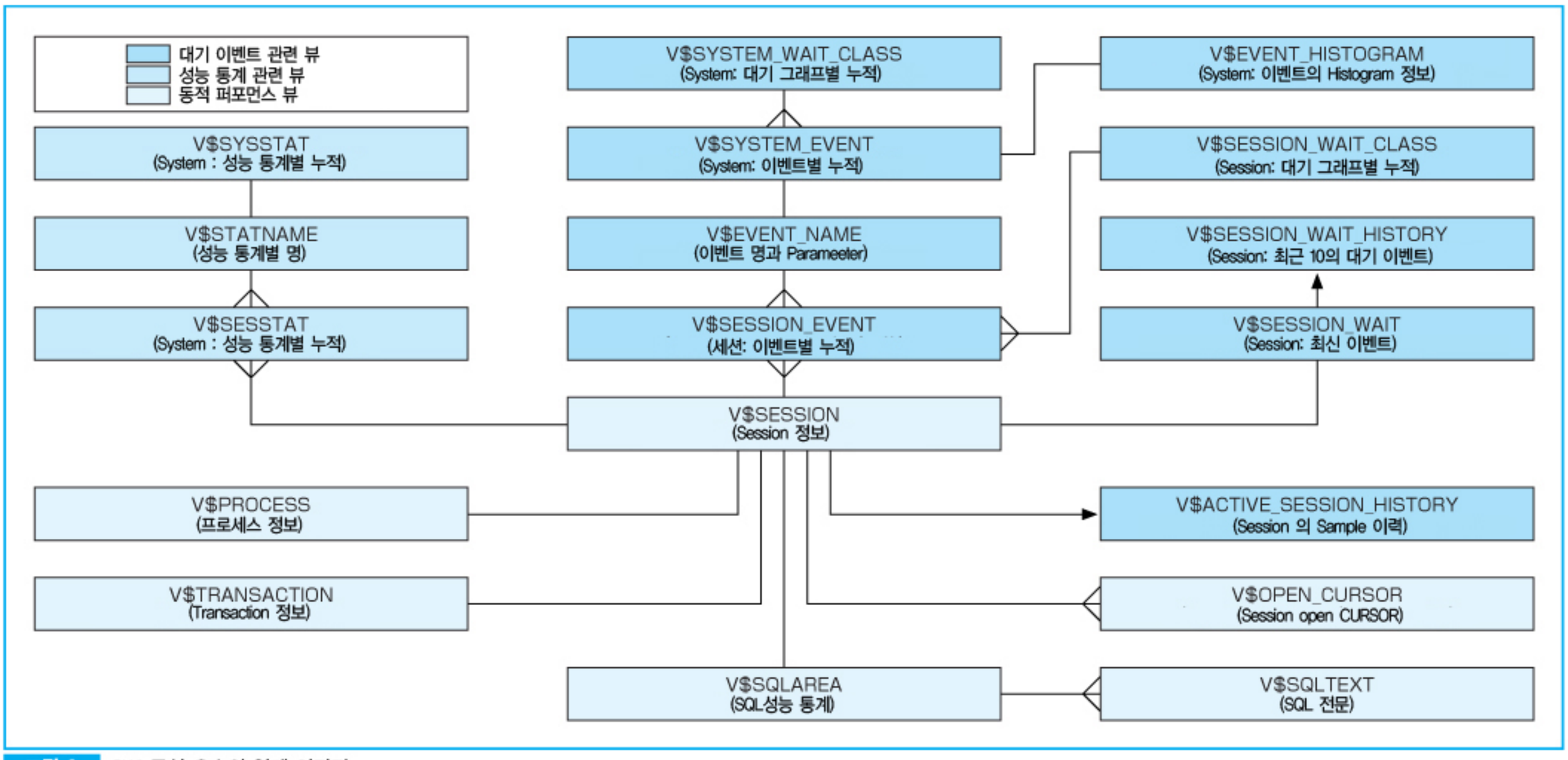


그림 2 OWI 구성 요소의 연계 이미지

징후가 나타나는 현상도 있다. 그 징후는 확실하게 OWI의 지표로써 표현되기 때문에 사전에 그 특징을 파악해 두면 성능개선은 물론 문제의 회피 및 잠재적 리스크의 해소에도 활용할 수 있게 된다. 이 같은 운영방법을 '사전징후감시 운용'이라고 한다.

OWI를 제공하는 툴

Oracle은 다양한 인터페이스를 통해 OWI를 제공하고 있다. 앞서 소개한 테스트 결과에서 볼 수 있듯이 STATSPACK 리포트와 SQL Trace 파일을 통해 대기 이벤트의 통계정보를 참조할 수 있다. 전자는 일정 기간의 운영 상태를 요약해서 간단히 살펴볼 수 있기 때문에 시스템 전체의 상태를 파악하는데 적합하다. 후자는 SQL이나 커서 단위로 이벤트 통계를 수집하기 때문에 상세한 분석이 필요 할 때 최적이다.

또한, 가장 중요한 인터페이스로서 동적 파라메터뷰, 대기 이벤트 관련 뷰, 성능통계 관련 뷰가 제공되고 있다(그림 2). 각종 뷰를 적절히 연결해서 참조하는 것으로써 시스템 레벨에서 세션레벨, SQL레벨까지의 요약정보에서 상세 정보까지, 폭넓고 정확한 통계정보를 간단히 확인 할 수 있다. 특히, Oracle 10g에서는 이력정보를 보관하는 뷰가 추가되어 보다 손쉽게 분석할 수 있게 되었다.

무엇보다 이상적인 OWI 활용방법은, 각종 뷰를 정기적으로 참조하여 그 정보를 보존함으로써 운용 상황의 추이분석이나 특정 시점에서의 운용 상황을 분석하는 것이다. 그러나 SQL로 빈번하게 정보를 취득하는 것이 거꾸로 DB에 부하를 주거나 수집한 데이터의 가공이 너무 어렵거나 수고스럽기 때문에 대부분 이용되고 있지 못하는 것이 실정이다.

Oracle DB의 아키텍처와 OWI

OWI는 Oracle DB의 아키텍처와 밀접한 관계가 있어 OWI 그 자체가 Oralce DB의 내부 구조를 그대로 나타내고 있다 해도 과언이 아니다.

(그림 3)는 통계정보가 DB 내의 어떤 부분에서 발생하고 있는지를 나타내고 있다. (그림 4)는 'SELECT' 처리시에 (그림 5)는 'UPDATE' 처리시에 DB 내부에서 발생하는 이벤트의 흐름을 간단히 이미지화 한 것이다. 각 영역과 OWI의 상세에 대해서는 Part II서 자세히 설명하겠다.

각 지표는 Oracle DB의 각 리소스와 연계하여 프로세스에 따른 리소스의 사용상황을 수치화하고

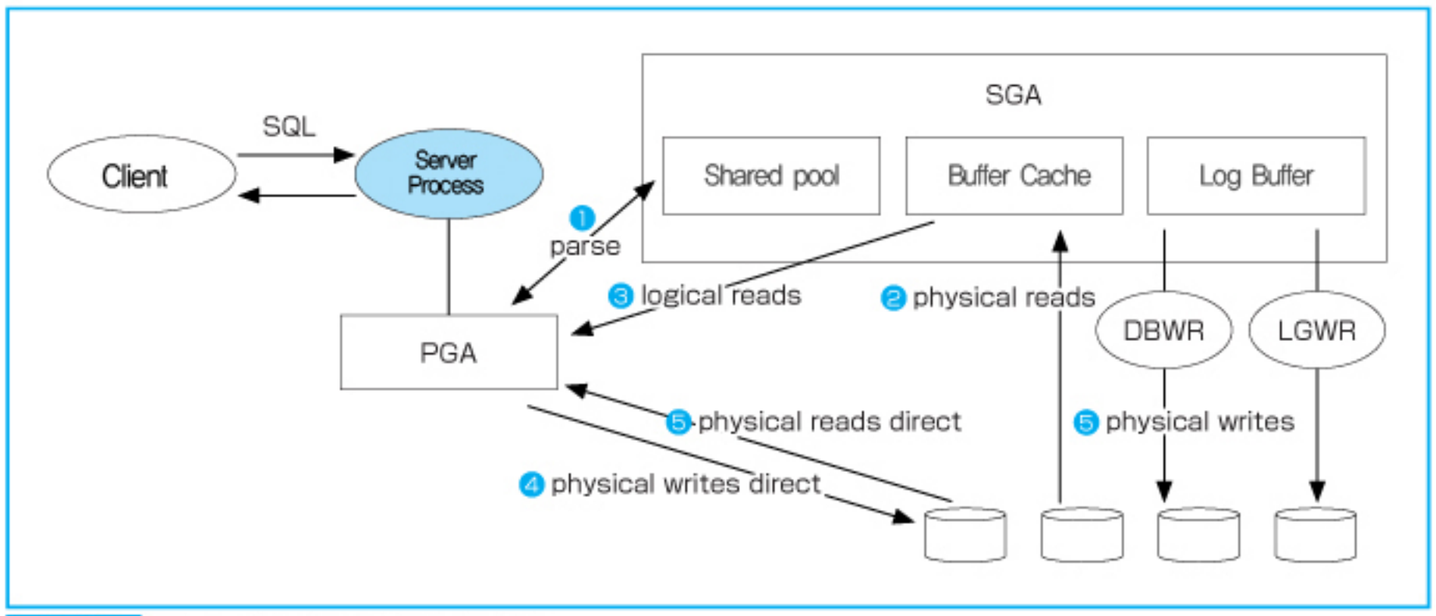


그림 3 Oracle DB의 아키텍처와 통계정보

특집 2

OWI에 의한 Oracle 성능개선과 장애대책

있으므로, 각 성능 및 대기 지표가 발생하는 구조와 영역을 이해하는 것으로 OWI를 보다 활용할 수 있게 된다. 다름아닌 Oracle DB의 내부 매커니즘을 이해하는 툴로써 OWI를 활용할 수 있는 것이다.

OWI를 활용한 성능개선

여기서부터는 앞서 소개한 시나리오별 병목현상에 대하여 구체적인 개선책과 각각의 테스트 결과를 전후의 통계정보를 비교해 봄으로써 OWI의 효과를 이해 해 주었으면 한다.

케이스 1의 개선 포인트

먼저, 대량의 SQL 파싱이 DB 내부에서 어느 정도의 병목으로 나타나는지를 확인한다. 통상적으로 바인드 변수를 사용한 SQL의 파싱은 시스템에 부하를 주지 않는다고 여겨지고 있지만 이는 절반의 사실일 뿐이다. 물론 신규 SQL의 파싱 정도의 부하는 아니지만 Library Cache 내부에 격납되어 있는 기존의 Object를 검색하는 작업은 건너 뛸 수 없기 때문에 SQL 파싱은 회수 그 자체가 데이터베이스 전체의 성능에 영향을 줄 수 있다.

특히 동시 작업이 많아지게 되면, 파싱에 사용되는 리소스(상세한 내용은 Part II의 칼럼 'Oracle의 동기화 매커니즘, Latch와 Lock'을 참조)는 한정되어 있기 때문에 일부 프로세스는 'latch: library cache' 이벤트에 의한 대기 상태가 된다. 이번에는 소프트 파싱(상세한 설명은 Part II 칼럼 'SQL해석 처리의 플로우'를 참조)도 하지 않도록 'SESSION_CACHED_CURSORS'를 설정하여 '2847→1734'초(39.1% 단축)의 응답시간의 성능개선을 이루었다.(LIST 11)

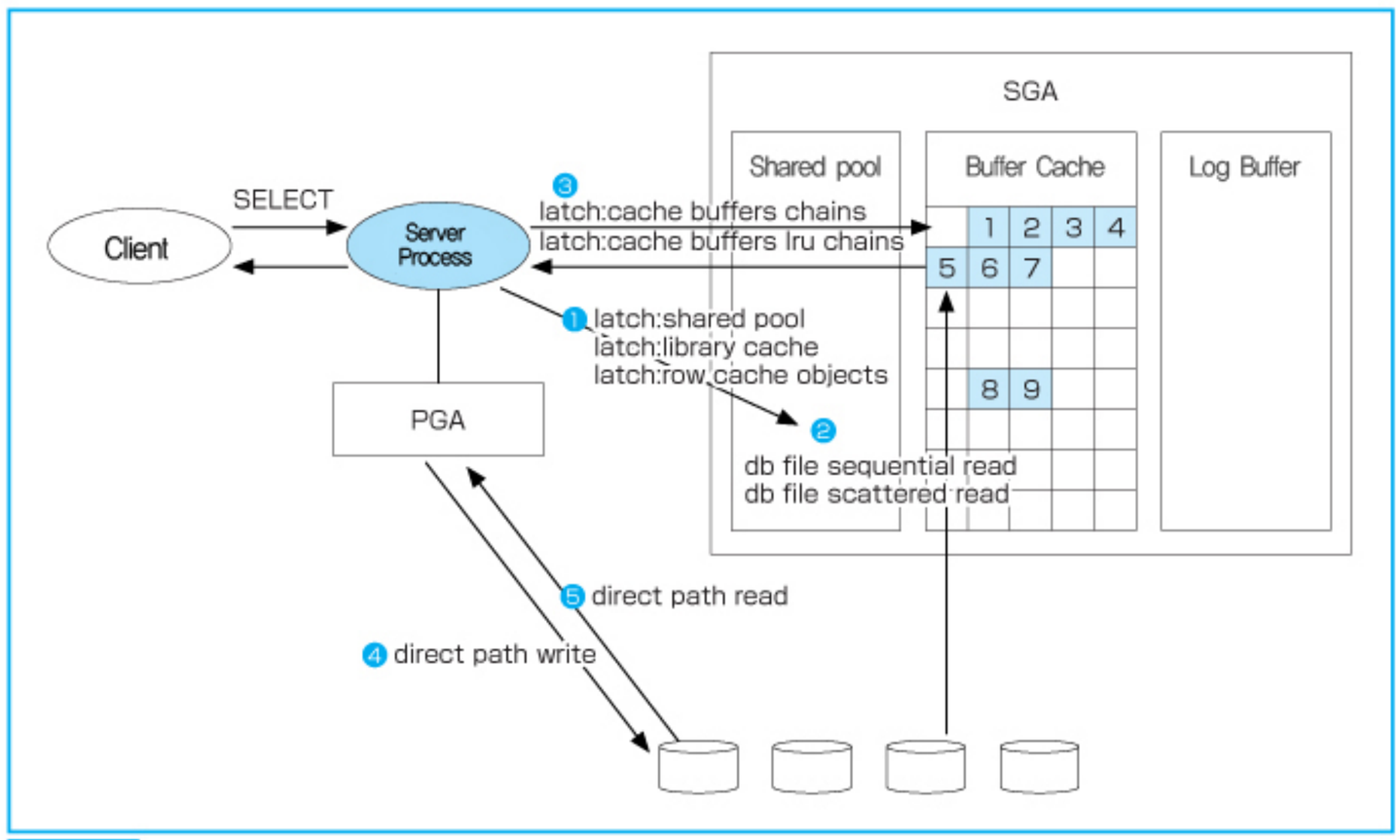


그림 4 SELECT 작업시의 OWI

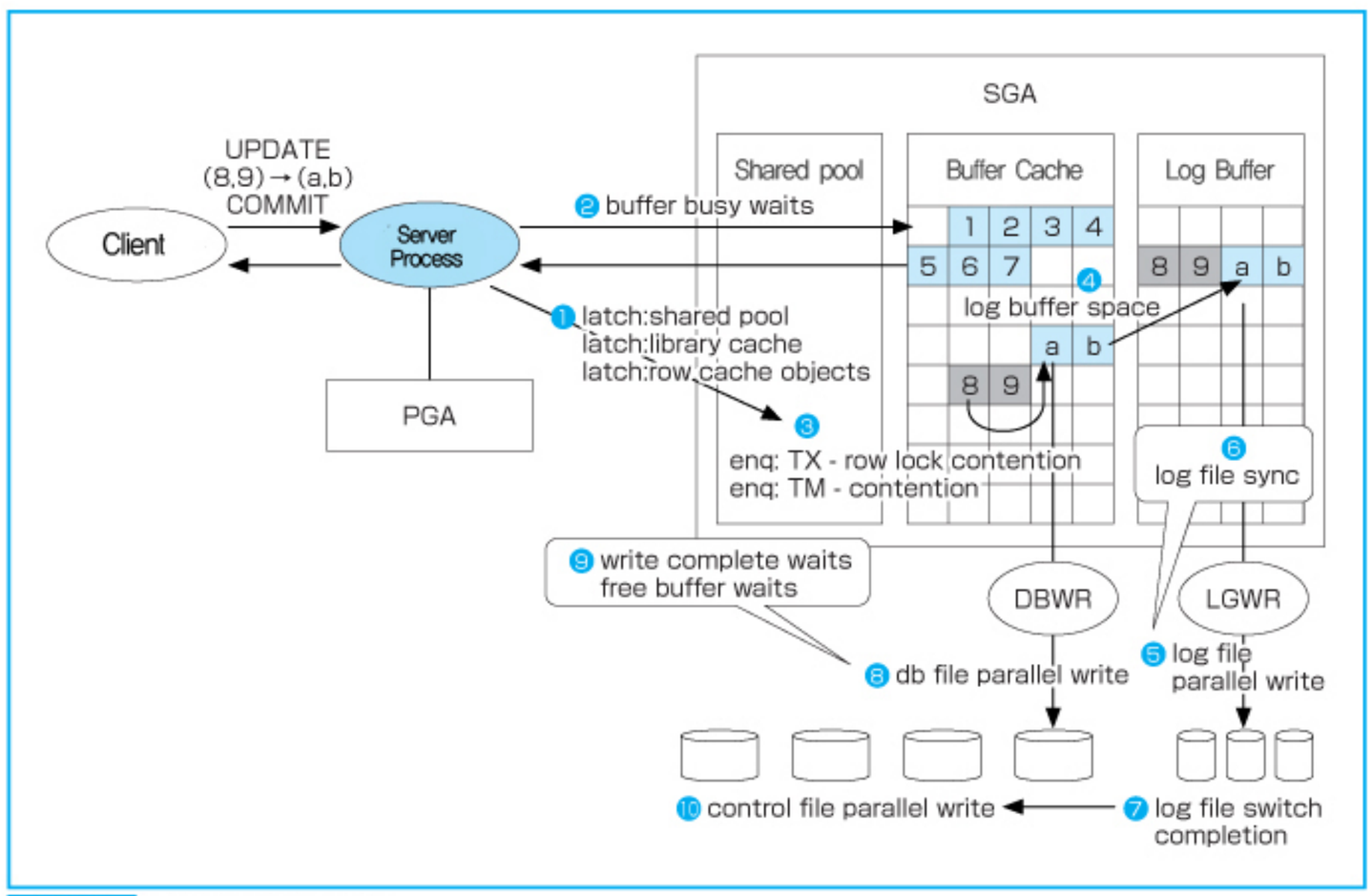


그림 5 UPDATE 작업시의 OWI

케이스 2의 개선 포인트

프로세스에서 디스크I/O의 요구가 있을 경우, 대상 Data를 포함한 Block이 SGA에 로드 될 때까지 해당 프로세스는 대기상태가 되지만 그때 로드 된 Block의 단위가 1block일 때에는 'db file sequential read', 복수 block의 경우에는 'db file scattered read' 이벤트가 발생한다.

이 테스트에서는 인덱스를 경유하여 access하는 Data의 분산상황에 따라 디스크 I/O 처리 시간이 급격하게 변화함을 대기 이벤트를 기준으로 하여 검증한 시뮬레이션이 된다. 당연한 일이지만 access하는 Data Block의 수가 많을수록 성능이 떨어지고 적을수록 성능이 좋아져 처리시간이 짧아진다.

이번에는 인덱스 키의 순서에 맞게 테이블 Data를 다시 입력한 결과 '199→13'초(93.5% 단축)로 처리 시간이 급감했다(LIST 12) 실제로 access하는 Data Block이 줄어듦으로써 낭비되는 CPU 처리 시간이 없어졌기 때문이다.

케이스 3의 개선 포인트

이 개선책에서는 먼저 광범위 한 검색으로 발생하는 병목현상을 재현했다. 이러한 작업이 인식되지 못한 채 실행될 경우 어떠한 병목현상이 발생하는지를 확인할 수 있다. 서버 프로세스로부터 Data의 요구(SELECT)가 있으면

Oracle DB는 해당 Block이 메모리에 로드 되어 있는지를 확인하지만 Data의 정합성과 그 검색 작업을 효율적으로 수행하기 위해 'cache buffers chains' latch라는 구조를 사용한다.
검색 범위가 넓으면 그 검색에 걸리는 시간도 증가하여 같은 검색작업을 요구하는 다른 프로세스는 대기 상태가 된다. 이 때의 대기 상태를 'latch: cache buffers chains' 이벤트라고 한다.
이번에는 Data 참조작업의 SQL에 맞추어 인덱스를 최적화(항목 추가)하는 것으로 '3087→13' 초(99.6% 단축)이라는 극적인 성능개선을 실현했다. (LIST 13)이것은 access하는 Block의 범위를 최적화 함으로써 CPU 처리와 메모리 Block에 대응하는 경합이 폭넓게 감소했기 때문이다.

케이스 4의 개선 포인트

Oracle DB는 Data Block 단위로 I/O를 실행하기 때문에 복수의 세션이 동시에 다른 Data를 변경해도 그 데이터가 물리적으로 같은 Block에 위치하는 경우에는 해당하는 메모리 Block을 보호하기 위해 배타적 작업을 수행한다. 그때 Buffer Lock에 대한 경합이 발생하지만 Buffer Lock을 획득하지 못한 프로세스는 Lock이 해제될 때까지 'buffer busy waits' 이벤트로 대기한다.
이 이벤트는 어떤 시스템에서도 어느 정도는 발생하기 때문에 발생 그 자체를 문제시화 할 필요는 없지만 TOP 5에 들어갈 때에는 주의가 필요하다. 보통 'UPDATE'와 'UPDATE', 'INSERT'와 'UPDATE'에 의해 발생하지만 'SELECT'와 'UPDATE'에 의해서도 발생한다.
이번 테스트에서는 동시에 복수의 세션이 같은 Block에 있는 다른 데이터를 갱신하고 있는 상태이다.
이러한 지연을 회피하기 위해서는 Application의 특성을 고려하여 설계 시에 같은 Block의 Data에 대해서는 추가/변경 작업을 최소화 하도록 하는 것이 기본이 된다. 일반적으로 파티션화, FREELISTS의 조정 또는 ASSM 적용 등으로 동시성이 많은 Block을 분산하는 것을 권장하지만 운용중인 실제 환경에서는 현실적이지 못한 부분도 있어 해결책의 모색에 고민했던 쓰라린 경험이 있는 사람도 있을 것이다.
이 현상이 상위에 들어가는 경우는 해당 하는 세그먼트를 특정하는 것부터 시작할 필요가 있다.
이번 테스트에서는 해쉬 파티션을 적용하여 '3420→1530' 초(55.3% 단축)의 응답시간 성능 개선을 실현하였지만(LIST 14), REDO LOG FILE에서의 경합 해소가 아직 남아 있다고 생각되어진다.

케이스 5의 개선 포인트

특히 동시 유저가 막대한 시스템에서 볼 수 있는 현상으로, Application의 Transaction 제어가 제대로 실행되지 않은 경우에 발생하는 Row Lock 대기 상태를 재현하였다. 커밋하지 않고 Data에 대한 Lock을 장시간 대기토록 하는 작업에서 자주 발생하지만 어플리케이션마다 Data 갱신 대상의 순서가 다른 경우도 많아 운용면 또는 트랜잭션 제어의 수정을 통해 대처해야 한다.
이번 테스트에서는 Data 갱신 후 커밋을 수행토록 수정함으로써 비록 Row Lock 대기 상태는 발생하지만 '5048→401' 초(92.1%단축)의 응답시간의 성능 개선을 이루었다.(LIST 15)

케이스 6의 개선 포인트

이번 개선책에서는 복수의 세션이 동시에 DML을 실행했을 경우의 REDO Log Buffer에 걸리는 부하를 시뮬레이션 해 보았다.
통상적으로 커밋시의 3초마다, REDO Log Buffer의 1/3 이상 혹은 1MB 이상의 Data가 쌓였을 때 LGWR에 의해 REDO Log Buffer의 내용이 REDO Log에 기록된다.
그러나 단시간에 커밋이 없는 대량의 REDO Entry가 발생하는 경우에는 LGWR이 상태를 감지하고 쓰기 전에 REDO Log Buffer는 가득 찬다. 이는 디스크 I/O보다 메모리의 처리 속도가 훨씬 빠르기 때문이며 REDO Log Buffer가 가득 차면 그 다음의 REDO Entry는 더 이상 들어갈 수 없게

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                             Waits   Time (s)    (ms)   Time
----------------------------------------- ------------ ---------- ------ ------
CPU time                                                      1,006           52.3
cursor: pin S wait on X                          2,815        389    138   20.2
cursor: pin S                                    3,944        177     45    9.2
job scheduler coordinator slave wait                 6         98  16274    5.1
os thread startup                                   76         64    847    3.3
```

(SQL 파싱의 횟수를 최소화 해서 해결)

LIST 11 케이스 1의 개선효과

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                             Waits   Time (s)    (ms)   Time
----------------------------------------- ------------ ---------- ------ ------
CPU time                                                          7           43.7
db file sequential read                            665          6      9   41.2
control file sequential read                       344          0      1    3.0
db file scattered read                              71          0      6    2.6
control file parallel write                         59          0      7    2.6
```

(Data를 다시 넣어 디스크I/O 대기를 최소화 함)

LIST 12 케이스 2의 개선효과

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                             Waits   Time (s)    (ms)   Time
----------------------------------------- ------------ ---------- ------ ------
CPU time                                                          8           63.9
enq: TX - row lock contention                       95          2     20   14.7
db file sequential read                            138          1      6    6.1
os thread startup                                   30          1     21    4.8
control file sequential read                       321          1      2    4.3
```

(인덱스 구성의 갱신으로 access 범위를 최적화)

LIST 13 케이스 3의 개선효과

OWI에 의한 Oracle 성능개선과 장애대책

되어 프로세스는 대기상태가 된다. 이때 발생하는 대기상태를 'log buffer space' 이벤트라고 한다.

이 테스트에서는 REDO Log Buffer의 사이즈를 크게 하여 '4521→3536' 초(21.8% 단축)의 응답시간 성능개선을 실현했다.(LIST 16) 실제로 TOP을 점했던 'log buffer space' 이벤트가 거의(95% 이상) 사라졌음에도 불구하고 전체의 소요시간이 기대한 만큼 줄어들지 않은 것은 최초의 병목현상에 의해 표출되지 않았던 잠재적인 병목이 표면으로 표출되었기 때문이다. 'enq: HW-contention' 이 HWM(하이워터 마크)의 이동 시 발생하는 이벤트라는 것을 힌트로 계속 진단/분석 해야 할 것이다.

전후의 통계정보를 비교해 봄으로써 OWI의 효과를 이해 해 주었으면 한다.

케이스 7의 개선 포인트

이 테스트는 빈번한 커밋 수행이 처리 전체의 응답시간에 어떠한 영향을 주는가를 확인 하기 위한 시뮬레이션이 된다.

서버 프로세스에서 커밋 또는 롤백이 수행되면 그 데이터를 보증하기 위해 LGWR이 그 시점까지의 REDO Entry를 REDO Log File에 기록한다.

이 때, 서버 프로세스는 LGWR의 처리가 완료하기까지 'log file sync' 에 의한 대기 상태가 된다. 이 경우는 그룹 커밋(예를들어 1,000건) 등을 통해 그 횟수를 줄임으로써 해당 대기를 해소할 수 있다. 이번에는 '2667.5→1347.0' 초(49.5% 단축)의 개선을 꾀할 수 있었다.(LIST 17)

또한 그 정도 심각하지는 않지만 다른 대기 이벤트에 대해서도 계속해서 진단/개선책을 검토하는 것이 좋을 것이다.

케이스 8의 개선 포인트

SQ Lock은 낮은 값의 'CACHE' 계층을 가진 시퀀스를 동시에 대량으로 발행했을 때 발생한다. 시퀀스는 적은 부하로 유닉크한 값을 취득할 수 있는 것을 목적으로 하고 있기 때문에 일정 구간의 값을 메모리에 캐쉬 해 놓는다. 다만 딕셔너리의 갱신이 이루어질 때 복수의 세션 중 하나의 세션만이 시퀀스 풀을 재차 캐쉬 하는 것이 보증되기 때문에 다른 세션은 'enq: SQ-contention' 이벤트를 대기하게 된다.

예를 들어, Default로 작성된 시퀀스는 '20' 의 CACHE 속성을 가지기 때문에 '20' 회 발행하면 재차 딕셔너리를 갱신하여 다음 20개의 시퀀스를 메모리에 CACHE하지 않으면 안 된다. 그 때문에 트랜잭션이 동시에 대량으로 발생하는 테이블의 키로써 시퀀스를 사용하는 경우에는 CACHE 속성 값이 너무 작으면 병목의 원인이 되기 때문에 충분히 크게 설정할 필요가 있다.

이번 테스트에서는 시퀀스의 'CACHE' 속성을 '20→10000' 으로 변경함으로써, '2419→531' 초(78.1% 단축)의 응답시간 성능개선을 실현하였다.(LIST 18) 또한 시퀀스에서의 병목 현상이 해소되어 'latch: library cache' 등의 대기가 증가하여 상위에 점하게 되었지만 CPU 사용시간에 비교해 신경 쓸 정도의 현상은 아니라고 생각한다.

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                            Waits   Time (s)     (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
log file sync                                    [illegible]   790     26   42.1
log file switch (checkpoint incomplete)              309       282    913   15.0
buffer busy waits                                  2,571       197     76   10.5
CPU time                                                       135           7.2
enq: TX - row lock contention                      2,654       126     47   6.7
```

파티션화에 따른 Data분산으로 병목 현상 해소

LIST 14 케이스 4의 개선효과

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                            Waits   Time (s)     (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
PL/SQL lock timer                                [illegible]   318     69   77.8
enq: TX - row lock contention                      2,418        48     20   11.7
job scheduler coordinator slave wait                   1        16  16000    3.9
CPU time                                                        12           2.9
latch: cache buffers chains                          165         7     41   1.7
```

Transaction 제어를 다시 하여 해결

LIST 15 케이스 5의 개선효과

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                    wait   Call
Event                                            Waits   Time (s)     (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
enq: HW - contention                               3,973     2,184    550   52.5
log file switch (checkpoint incomplete)              580       505    871   12.1
buffer busy waits                                [illegible]   464     98   11.2
log file switch completion                           470       248    527   6.0
log buffer space                                     210       135    643   3.2
```

REDO Log Buffer의 크기 변경으로 해결

LIST 16 케이스 6의 개선효과

케이스 9의 개선 포인트

이 테스트는 리모트 서버와의 연계 작업이 많은 상태에서 네트워크의 부하가 높아졌을 때 나타나는 현상을 대기 이벤트를 중심으로 검증한 시뮬레이션이 된다. 이번처럼 DB Link를 경유해서 대량의 data를 참조/추가/갱신하면, 로컬 서버에서는 'SQL*Net message from dblink' 이벤트를 대기하는 등, 네트워크의 병목은 'SQL*Net …' 이벤트로서 나타난다. 이번에는 리모트 테이블을 로컬에 정기적으로 Refresh하는 것으로써 '1133.7→92.6' 초(91.8% 단축)의 처리 시간의 성능개선이 이루어져(LIST 19), CPU 사용시간을 포함한 통계정보가 안정적인 수치를 나타내게 되었다.

*　　*　　*

지금까지 Part I에서는 몇 가지 사례를 통해 OWI를 활용한 진단/분석의 패턴을 살펴 봤는데 어떠하였는가? 역시 익숙하지 않은 부분도 있었을 거라고 생각한다. 새로운 진단/분석 메소드를 익히기 위해서는 그 구조에 대한 이해와 약간의 시행 착오로부터 얻게 되는 지식 및 경험이 필요하게 된다. OWI의 보다 상세한 구조에 대해서는 다음에 연재되는 Part II에서 각 영역으로 나누어 개요와 흐름을 설명하고 있기 때문에 계속 참고해 주었으면 한다. 끝까지 다 읽고 나서 다시 Part I의 시나리오별 사례를 살펴보면 통계 Data가 보다 선명하게 보일 것이다.

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse     1185      0.03       0.24          0          0          0           0
Execute 1819095    554.15    8243.49       1858  141324571    3392489     1802240
Fetch       466      0.00       0.01          0        391          0         367
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total   1820746    554.18    8243.74       1858  141324962    3392489     1802607

Elapsed times include waiting on following ev
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  latch: cache buffers chains                  1483          67       189.80
  latch: cache buffers lru chain               1822        0.07       188.13
  latch free                                   1333        0.74       144.34
  log file switch completion                    161        1.07        97.06
  buffer busy waits                            3024        1.16        89.87
  latch: undo global data                       346        0.45        39.29
  log file sync                                 206        0.63        44.28
```

그룹 커밋으로 해결

LIST 17 케이스 7의 개선효과

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse     1741      0.26       2.41          0          2          0           0
Execute 2681741    320.85    1745.00          0      42022       5273        3520
Fetch   2680403    107.28     465.12          0          7        274     2680403
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total   5363885    428.40    2212.54          0      42031       5547     2683923

Elapsed times include waiting on following e
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  latch: library cache                         1347        1.31       43.32
  latch free                                    960        0.13       27.14
  enq: SQ - contention                          895        0.15       16.11
  cursor: pin S                                 712        0.15        8.13
  latch: library cache pin                      222        0.12        5.58
  enq: TX - row lock contention                  23        0.43        1.13
  buffer busy waits                               4        0.52        0.98
```

Sequence 메모리 Cache량 조정으로 개선

LIST 18 케이스 8의 개선효과

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       99      1.09      13.85          0          0          0           0
Execute  27099     22.96     207.14        137    2159177        281         232
Fetch    27027     48.23     234.80        137    3024000          0       27027
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    54225     72.29     455.80        274    5183177        281       27259

Elapsed times include waiti                                        s:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  latch: cache buffers chains                   388        0.13       12.33
  cursor: pin S wait on X                       530        0.02        5.72
  latch free                                     40        0.10        1.17
  read by other session                         619        0.08        1.02
  db file sequential read                        28        0.00        0.03
  db file scattered read                         27        0.01        0.04
```

리모트 테이블을 로컬에 주기적으로 복사함으로써 대기시간을 개선

LIST 19 케이스 9의 개선효과

DBM