WEB System Tuning Guide

Prepared by	Computing 사업부
Authors	DB 지원팀
Reviewers	이진철 차장
Creation Date	2005/09/01
Last Update	2005/09/01

목 차

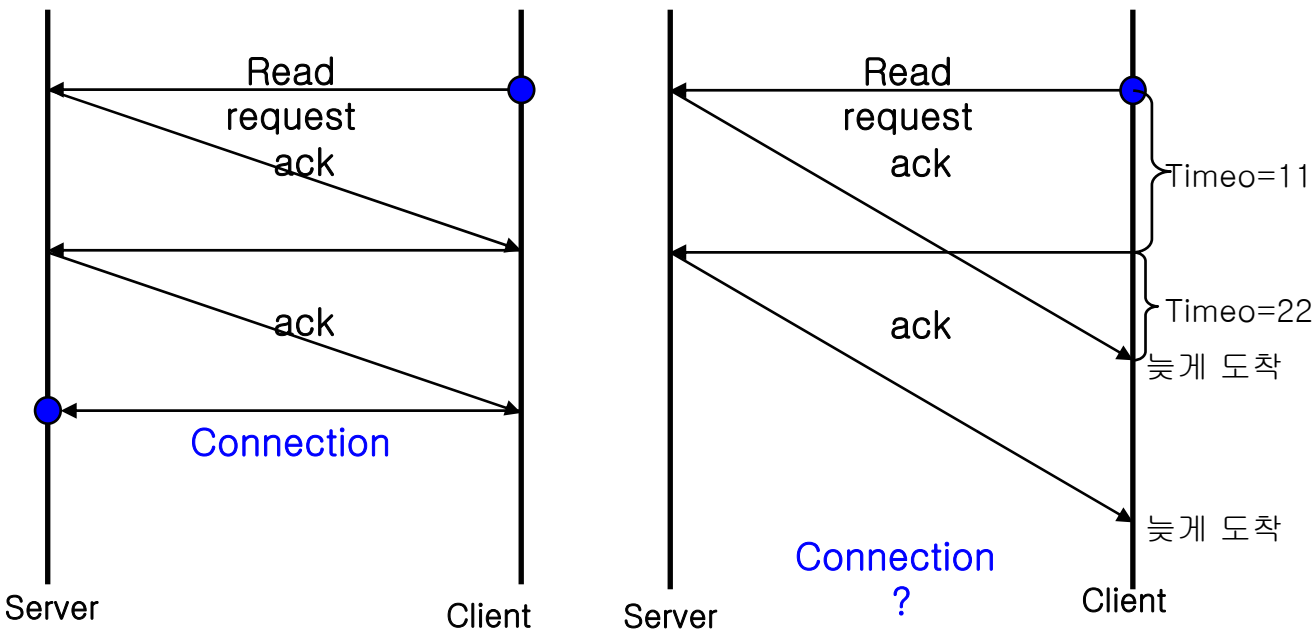
2. 웹 서버의 튜닝

2-1. TCP/IP Kernel Parameter

일반적으로 최초 설치(Install)된 운영체제(Operating System)는 범용의 목적 하에 Kernel Parameter 설정이 이루어진다. 따라서 Web Service만을 위한 Kernel Parameter값은 반드시 최적화시켜야 한다. 특히 웹서버의 성능에 영향을 많이 주는 TCP/IP parameter setting은 20~30% 정도의 성능 향상을 가져 올 수 있을 정도로 중요하다. 본 장은 웹서버의 형태로 운용되는 유닉스 서버의 TCP/IP 일반과 Tuning에 관한 내용을 기술한다

2-1-1. TCP connection initiation background

TCP는 reliable connection oriented protocol 이다. 즉 2개의 machine사이에 데이타를 주고 받기 전에 connection이 이뤄지는 것이다.
Adrian Cockcroft(Sun Performance and Tuning, Java and the Internet, 2nd Edition의 저자)는 TCP connection의 설명을 Phone Call에 비유를 하였다. (dialing, talking, hanging up). Incoming Call은 수동적으로(passive) open되고 (Server Part에서 아무런 동작 없이 일어난다.) 그러나 Server는 Call을 Accept하기 위해서 특별한 프로그램이 동작하고 있어야 한다. Outgoing Call은 Server에서 시작 되는데, 이것은 능동적인(active) open으로 분류된다. 이러한 경우의 예로서 다른 host에 rlogin하는 경우에 해당이 된다.
Connection의 시작은 3-way handshake sequence로 구성이 된다. 이러한 handshake process는 round-trip 시간과 모든 delay에 영향을 받는다. 최초의 handshake는 **SYN(synchronize sequence numbers)** 로 구성된 incoming packet 이다. Server는 자신의 listen queue(**tcp_conn_req_max_q0** on solaris, **tcp_syn_rcvd_max** on HP-UX) 에 entry를 만들고 connection은 **SYN_RCVD** 상태로 남게 된다. Server는 고유한 **SYN** 을 포함하는 **acknowledges(ACK)**를 가지고 SYN packet에 응답을 한다.
Client에 서 오는 다음 packet은 second SYN에 대한 acknowledge이다. 두번째 SYN/ACK를 client로 부터 받고 난 다음에 connection은 accepted listen queue (**tcp_conn_req_max_q** on Solaris, **tcp_conn_request_max** on HP-UX)로 이동한다.



만약 client가 SYN을 보내기만 하고 어떤 ACK도 받지 않으면 time out 때까지 SYN_RCVD 상태로 남게 된다. 이러한 TCP의 작동 원리는 웹 서버 에서 권장하는 tcp 파라메타를 이해하는데 충분한 설명일 것이다. 만약 더 자세한 설명을 원한다면 아래 참고자료를 읽어보기 바란다.

1. Solaris - Tuning Your TCP/IP Stack
   http://www.rvs.uni-hannover.de/people/voeckler/tune/EN/tune.html
2. "Sun Performance and Tuning, Java and the Internet, 2nd Edition",
   Adrian Cockcroft, Richard Pettit ISBN 0-13-095249-4, 1998,pp.55-59
3. Tips For TCP/IP Monitoring And Tuning To Make Your Network Sing
   http://www.sunworld.com/swol-12-1996/swol-12-perf.html

2-1-2. TCP Kernel Parameter의 튜닝

UNIX는 사용자가 tune을 할 수 있도록 하고 있으며, 시스템 운영 중에도 TCP/IP stack 관련한 다양한 파라메타는 set과 reset이 가능하다. 이러한 작업은 'ndd' 명령어 (network device driver)로 가능하다. Ndd 명령은 시스템 재부팅 없이 변경이 가능하지만, 변경한 내용은 시스템 재 부팅 시 default 값이 되기 때문에 시스템 start script에 넣어 주어야 한다.

For Solaris Startup see : /etc/rc2.d/S69inet
For HP/UX 11.x Startup see : /etc/rc.config.d/nddconf
본 문서에서 언급하는 파라메타는 time interval이다 . 모든 interval은 milisecond 단위이다 (e.g. 1000=1 second).

가. 현재 설정된 TCP 커널 파라미터의 값 확인

```
# ndd /dev/tcp \?                    → 모든 파라미터의 현재 설정 값을 Display한다.
# ndd /dev/tcp tcp_close_interval    → 특정 파라미터의 현재 설정 값을 Display한다.
# ndd -set /dev/tcp tcp_conn_req_max_req 10240
                                     → 특정 파라미터의 현재 설정 값을 변경한다
```

2-1-3. Parameter Description

tcp_conn_request_max
Maximum number of outstanding inbound connection requests

tcp_conn_req_max_q0 (Solaris) / **tcp_syn_rcvd_max** (HP)
Handshake incomplete 상태의 최대 connection 갯수. 많은 수의 SYN이 몰려들 때 이 파라메타에 영향을 줄 수 있고, 특별한 알고리듬에 의해서 유효한 connection이 지속될 수 있도록 한다. 즉, 현재 큐에 있는 connection은 방금 초기화 된 것들다. SYN는 client 에서 도착되었고, TCP connection은 SYN_RCVD 상태로 된다. Connection은 handshake가 완료 되어야만 accept 된다.

tcp_conn_req_max_q
CPU time을 얻고 accept를 return받은 completed connection의 최대 갯수

2-1-3. Parameter Description (계속)

tcp_close_wait_interval(pre Solaris 2.7)/**tcp_time_wait_interval**(Solaris7 이후, HP-UX) default 240000 (according to RFC 1122, 2MSL), recommended 60000 possibly lower. 이 파라메타는 TIME_WAIT interval을 나타낸다. 대다수의 사용자가 local에서 사용한다면, 60000 ms 정도면 충분하다. 그러나 만일 사용자 분포가 많다면, RFC 1122 에서 조언하는 대로 240000 정도가 되야 할 것이다.

tcp_slow_start_initial
solaris와는 다르게 NT 4.0에서는 초기 접속 시작에 대한 응답을 곧바로 하지 않기 때문에 시작에 대한 지연이 발생한다. 그러나 NT 4.0에서는 2개의 패킷이 보내졌을 때 곧바로 응답을 하게 된다. NT와 Solaris의 구현상의 차이점 때문에 생기는 성능차이와 높은 응답 시간 때문에 NT client가 Solaris Server에 접속할 경우에는 고속이나 LAN-bases 네트웍을 이용하여 접속하여야 한다. Solaris에서 Congestion Window를 tcp_slow_start_initial = 2로 셋업하도록 한다.

tcp_recv_hiwat
default 8192, recommended 32768
이 파라메타는 초기의 TCP reception buffer의 최대 크기이다. 명시된 값은 MSS의 2배까지 될 수 있다. Buffer의 free space에서 지정된 window size를 검사할 수 있다. Reception window의 크기는 remote peer에 영향을 준다.

tcp_xmit_hiwat
default 8192, recommended 32768, maximum 65535
LFN bulk data transfer 131071 or above (아래 설명 참조)
이 파라메타는 initial send window 크기를 휴리스틱하게 결정하는데 영향을 준다. 실제 값은 MSS의 두배까지 될 수가 있다. (예, 8760 = 6*1460). Solaris 에서는 65535 이상의 값이 가능하다. 만약 peer host가 RFC 1323 (http://www.merit.edu/internet/documents/rfc/frc1323.txt) 에 의해 구현이 되었다면 buffer size를 65535 이상으로 설정하여 이득을 얻을 수 있다. 그러나 만약 host가 window scale option으로 구현되지 않았다면, window의 한계는 아직 64K 이다.

tcp_conn_hash_size (Solaris Only)
Solaris에서 tcp_conn_hash_size 파라메타는 connection backlog를 지정하는데 도움을 준다. High connection rate에서 TCP data structure kernel의 lookup은 매우 비싼 비용을 치루고 있고, server를 느리게 할 수 있는 요소이다. 따라서 이 파라메타를 늘려줌으써 hash table의 크기를 늘려서 lookup 효율을 높여줄 수 있다. 이 값은 active TCP connection을 관리하는데 사용되는 kernel hash table의 크기이다. 큰 값을 가질수록 많은 접속을 하고 있는 서버에서 높은 효율을 갖는다. Solaris의 경우 2의 배수의 값을 갖을 수 있고, 256(default)이 가장 작은 값이고, 밴치마킹에서 일반적으로 사용되는 값으로 가장 큰 값은 262144이다. 더 큰 tcp_conn_hash_size는 많은 메모리를 필요로 하지만, 많은 접속을 해야하는 경우에는 메모리에 대한 투자를 고려해 볼 가치가 있다. 이 값은 반드시 2의 배수이어야 하고 /etc/system 커널 configuration 화일에서 조정할 수 있다. 따라서 현재 size는 ndd로 조회했을 때 tcp_conn_hash 값을 읽기만 할 수 있도록 되어 있다.

2-1-4. Recommended Parameter의 설정

본 문서에서 추천하는 파라메타 값은 각 운영체제(Solaris, Aix, Hp-ux) 의 Administration / Performance and Tuning Section을 참조하고 있다. 그러나 시스템을 위한 최적의 파라메타 값은 정해져 있지 않으므로 반드시 백업을 수행하고 나서 테스트와 tuning을 하여야 한다. 아래에서 기술하는 사항은 최적의 tcp 성능을 위하여 어떻게 파라메타 값을 설정하고 이해하는데 도움이 될 것이다.

[약어 설명]:

MSL - Maximum segment lifetime,TCP segment가 net상에서 존재할수 있는 최대 기간
MSS - Maximum segment size
LFN - Long Fat Network ([RFC 1323] 에 따르면 elephan(t)로 발음 된다.)
RFC - Request for comments, 문서 시리즈로 관련된 실험을 통하여 최적의 값들을 추천한 것이다.

가. Solaris Recommended Parameter

Parameter	Default Value	Tuning Value	File
rlim_fd_max	1024	8192	/etc/system
rlim_fd_cur	64	8192	
sq_max_size	2	0	
tcp_time_wait_interval	240000	30000	/etc/rc2.d/S69inet
tcp_conn_req_max_q	128	1024	
tcp_conn_req_max_q0	1024	4096	
tcp_ip_abort_interval	480000	60000	
tcp_keepalive_interval	7200000	900000	
tcp_rexmit_interval_initial	3000	3000	
tcp_rexmit_interval_max	240000	10000	
tcp_rexmit_interval_min	200	3000	
tcp_smallest_anon_port	32768	1024	
tcp_slow_start_initial	1	2	
tcp_xmit_hiwat	8129	32768	
tcp_recv_hiwat	8129	32768	

다음 명령을 통하여 설정하며, 시스템 리부팅시에도 적용될 수 있도록 관리한다.

```
#set rlim_fd_max=8192 : /etc/system
#ndd -set /dev/tcp tcp_close_wait_interval 60000 : /etc/rc2.d/S69inet
```

2-1-4. Recommended Parameter의 설정(계속)

나. HP-UX Recommended Parameter

Parameter	Default Value	Tuning Value	File
maxfiles	2048	4096	/stand/system
maxfiles_lim	2048	4096	
tcp_time_wait_interval	60000	60000	/etc/rc.confing.d/nddconf
tcp_conn_request_max	20	1024	
tcp_ip_abort_interval	600000	60000	
tcp_keepalive_interval	72000000	900000	
tcp_rexmit_interval_initial	1500	1500	
tcp_rexmit_interval_max	60000	60000	
tcp_rexmit_interval_min	500	500	
tcp_xmit_hiwater_def	32768	32768	
tcp_recv_hiwater_def	32768	32768	

/etc/rc.config.d/nddconf 파일에 기록하여 영구적으로 적용된다.

나. AIX Recommended Parameter

Parameter	Default Value	Tuning Value	File
rfc1323	0	0	/etc/rc.net
sb_max	1048576	1048576	
tcp_sendspace	16384	65536	
tcp_recvspace	16384	65536	
udp_sendspace	9216	65536	
udp_recvspace	41920	327680	
lpqmaxlen	100	512	

다음 명령으로 현재 파라미터값 확인 및 설정(수정)이 가능하다.

```
#/usr/sbin/no -a : 디스플레이
#/usr/sbin/no -o rfc1323=0 : 수정
```

이러한 TCP 권장 파라미터는 Oracle의 Apache(Httpd) 데몬 운영 시에도 적용되고 있다. 오라클이 설치된 서버의 $ORACLE_HOME/Apache/Apache/bin/tcpset.sh 파일을 열어보면 아래와 같은 내용이 설정 값으로 되어 있는 것을 볼 수 있다.

[ora920:$ORACLE_HOME/Apache/Apache/bin]% **cat tcpset.sh**

```
#!/bin/sh
# Set tcp parameters to values for a web server on Solaris
# get entries out of hash table ASAP
case `uname -r` in
"5.8")
        /usr/sbin/ndd -set /dev/tcp tcp_time_wait_interval 60000
        echo  "tcp_time_wait_interval =" ₩
        `/usr/sbin/ndd -get /dev/tcp tcp_time_wait_interval` ;;
"5.7")
        /usr/sbin/ndd -set /dev/tcp tcp_time_wait_interval 60000
        echo  "tcp_time_wait_interval =" ₩
        `/usr/sbin/ndd -get /dev/tcp tcp_time_wait_interval` ;;
"5.6")
        /usr/sbin/ndd -set /dev/tcp tcp_close_wait_interval 60000
        echo "tcp_close_wait_interval =" ₩
        `/usr/sbin/ndd -get /dev/tcp tcp_close_wait_interval` ;;
esac
# set queue lengths to maximum to prevent tcpListenDrops
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q 1024
echo "tcp_conn_req_max_q =" ₩
`/usr/sbin/ndd -get /dev/tcp tcp_conn_req_max_q`
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q0 1024
echo "tcp_conn_req_max_q0 =" ₩
`/usr/sbin/ndd -get /dev/tcp tcp_conn_req_max_q0`
# change slow start algorithm to bypass congestion problem
/usr/sbin/ndd -set /dev/tcp tcp_slow_start_initial 2
echo "tcp_slow_start_initial=" ₩
`/usr/sbin/ndd -get /dev/tcp tcp_slow_start_initial`
# set high transmission window for efficient TCP transfers
/usr/sbin/ndd -set /dev/tcp tcp_xmit_hiwat 32768
echo "tcp_xmit_hiwat =" ₩
`/usr/sbin/ndd -get /dev/tcp tcp_xmit_hiwat`
# set high receiving window for efficient TCP transfers
/usr/sbin/ndd -set /dev/tcp tcp_recv_hiwat 32768
echo "tcp_recv_hiwat =" ₩
`/usr/sbin/ndd -get /dev/tcp tcp_recv_hiwat`
# remind the user to set the tcp_conn_hash_size parameter, etc.
echo ""
echo "Set tcp_conn_hash_size in /etc/system to 32768"
echo ""
echo "Reboot and rerun this program if it had to be changed “
```

2-2. Source Code의 튜닝

본 노트에서는 Web Server의 Application 중, 가장 성능 문제를 많이 야기시키는 게시판 리스팅(Listing) 부분에 관해서 예시를 들고자 한다. JSP Source Code가 Database 단으로 던지는 SQL 문장의 로직이 전체적인 성능과 어떤 연관관계를 가지고 있는지 설명하고자 한다.

2-2-1. 게시판 리스팅 SQL 문장의 성능 분석

게시판 리스팅을 위한 SQL 문장의 원문과 Plan 분석은 아래와 같다.

```
SELECT RROWNUM, BDID, BDCD, BDNM, EGID....
FROM (
        SELECT ROWNUM AS RROWNUM, BDID, BDCD, BDNM, EGID.....
        FROM (
              SELECT BDID, BDCD, BDNM, EGID....
                FROM TBD9002M
               WHERE BDCD= '1607'        ➔ 게시판 분류코드
               ORDER BY GROU DESC, STEP ASC ) )
WHERE RROWNUM >0
  AND RROWNUM <= 10;
```

DBCD 컬럼에 인덱스를 생성하여 Full Table Scan을 회피한 경우의 Plan 분석

Execution Plan

```
--------------------------------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=70 Card=233 Bytes=1M)
   1    0   VIEW (Cost=70 Card=233 Bytes=1M)
   2    1     COUNT
   3    2       VIEW (Cost=70 Card=233 Bytes=1M)
   4    3         SORT (ORDER BY) (Cost=70 Card=233 Bytes=261K)
   5    4           TABLE ACCESS (BY INDEX ROWID) OF 'JAY.TBD9002M' (Cost=27
                    Card=233 Bytes=261K)
   6    5             INDEX (RANGE SCAN) OF 'JAY.IDX_TBD9002M_BDCD' (NON-UNIQUE)
                      (Cost=1 Card=233)
```

플랜에서 보여지는 것처럼 제일 안쪽의 인덱스에서부터 많은 량의 데이터를 가져와서 끝까지 이 것을 가공/처리하고 있다. 실제로 tkprof 결과를 분석해 보면 제일 안쪽부터 Index → Table → Sort → View 를 거치는 내내 1601 건의 데이터를 퍼올려서 마지막에 rownum 에 의하여 10건을 가져와 게시판 리스트로 뿌려주고 있다. 이러한 SQL 문장의 튜닝 시에는 항상 최초 시작하는 집합의 범위를 최소화 시킬 것을 권장하고 있기 때문에 이 문장은 다음과 같이 수정될 수 있다.

[수정된 문장과 플랜]

이 SQL 문장은 아래와 같이 수정될 수 있다. 게시판 테이블의 분류코드와 시퀀스 번호로 작성된 인덱스를 INDEX_DESC 하게 타게 되면 가장 최신의 게시물 순서로 자동 소트되어 있으므로 ORDER BY 절이 필요 없게 되며, 가장 안쪽의 퀄리 블록에서 ROWNUM < 10 을 사용함으로써 STOPKEY 역할을 하게된다.

```
SELECT RROWNUM, BDID, BDCD, BDNM, EGID....
FROM (
        SELECT ROWNUM AS RROWNUM, BDID, BDCD, BDNM, EGID.....
        FROM (
              SELECT /*+ INDEX_DESC(TBD9002M TBD9002M_IDX2) */
                     BDID, BDCD, BDNM, EGID....
                FROM TBD9002M
               WHERE BDCD= '1607'        ➔ 게시판 분류코드
                 AND ROWNUM <= 10) );
- - WHERE RROWNUM >0                     ➔ 주석처리
- -  AND RROWNUM <= 10;                  ➔ 주석처리
```

이 결과 아래와 같은 플랜이 생성된다.

Execution Plan

```
----------------------------------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=27 Card=233 Bytes=1M)
   1    0   VIEW (Cost=27 Card=233 Bytes=1M)
   2    1     COUNT
   3    2       VIEW (Cost=27 Card=233 Bytes=1M)
   4    3         COUNT (STOPKEY)
   5    4           TABLE ACCESS (BY INDEX ROWID) OF 'TBD9002M' (Cost=27 Card=233
                    Bytes=261K)
   6    5             INDEX_DESC (RANGE SCAN) OF 'TBD9002M_IDX2' (NON-UNIQUE)
                      (Cost=1 Card=233)
```

이 플랜에 의하여 동작되는 SQL문장은 최초 인덱스에서 테이블 Access까지는 1601건의 데이터를 핸들링 하지만, 이 후부터는 Stopkey에 의하여 오로지 10건의 데이터를 가지고 최종 수행단계까지 처리/도달한다.
수행 결과 평균 18초 걸리던 이 문장은 1.2초 이내에 결과를 리턴하게 된다. 웹 사용자들이 게시판 리스트가 18초 걸려서 나타나게 된다면 조금 심각한 지경이라고 생각할 것이다. 그런데, 문제는 이 SQL 문장을 튜닝하려면 Source Code를 수정해야 한다는 것이다. 이 JSP Source Code가 어떻게 되어 있는 지 살펴보자

[JSP 소스 코드에서 SQL 문장 생성 부분]

```
  if(field01.equals(""))     { tail02 = "' ) WHERE BDCD = '" + bdcd01 + "' AND
RROWNUM >= " + page_start + " AND RROWNUM <= " + page_end + " ORDER BY GROU DESC,
STEP ASC"; }
  if(field01.equals("suje")) { tail02 = "'  AND ( SUJE LIKE '%" + search01 + "%'))
WHERE BDCD = '" + bdcd01 + "' AND  RROWNUM >= " + page_start + " AND RROWNUM <= " +
page_end + " ORDER BY GROU DESC, STEP ASC"; }
  if(field01.equals("writ")) { tail02 = "'  AND ( WRIT LIKE '%" + search01 + "%'))
WHERE BDCD = '" + bdcd01 + "' AND  RROWNUM >= " + page_start + " AND RROWNUM <= " +
page_end + " ORDER BY GROU DESC, STEP ASC"; }
  if(field01.equals("yymm")) { tail02 = "'  AND ( WRDA LIKE '%" + search01 + "%'))
WHERE BDCD = '" + bdcd01 + "' AND  RROWNUM >= " + page_start + " AND RROWNUM <= " +
page_end + " ORDER BY GROU DESC, STEP ASC"; }

  sql01 =          " SELECT ";
  sql01 = sql01 +  " RROWNUM,BDID,BDCD,BDNM,EGID,WRIT,BUCD, ";
  sql01 = sql01 +  " BUNM,EMAI,REGI,TELON,TELTW,PAWO,PDSON, ";
  sql01 = sql01 +  " PDSTW,PDSTH,TYCD,TYNM,DECD,DENM,VICD,VINM, ";
  sql01 = sql01 +  " STCD,STNM,SGCD,SGNM,SGID,IGCD,IGNM,";
  sql01 = sql01 +  " HIT,SUJE,CONTON,CONTTW,CONTTH,WRDA,MODA,
                   GROU,STEP,LEVE  ";
  sql01 = sql01 +  " FROM (SELECT ROWNUM AS RROWNUM,BDID,BDCD,BDNM, ";
  sql01 = sql01 +  " EGID,WRIT,BUCD,BUNM,EMAI,REGI,TELON,TELTW, ";
  sql01 = sql01 +  " PAWO,PDSON,PDSTW,PDSTH,TYCD,TYNM,DECD,DENM, ";
  sql01 = sql01 +  " VICD,VINM,STCD,STNM,SGCD,SGNM,SGID,IGCD,IGNM,";
  sql01 = sql01 +  " HIT,SUJE,CONTON,CONTTW,CONTTH,WRDA,MODA,
                   GROU,STEP,LEVE ";
  sql01 = sql01 +  " FROM " + TBD9002M + " WHERE BDCD = '" + bdcd01 + tail02;
```

이 소스코드 부분이 게시판 리스팅 알고리즘의 열쇠를 갖고 있다. 이 소스코드의 의도는 게시판의 최신 데이터 순으로 첫 번째부터 열 번째까지의 결과를 리스팅하고, 만약 사용자가 게시판 리스트 화면의 하단 번호를 눌러서 그 다음 열 개의 문서 제목을 보고 싶어 한다면 열 한 번째부터 스무 번째까지의 결과를 리스팅 하고자 하는 것이다. 앞 페이지의 SQL 문장만을 분석했을 경우와는 조금 다른 Source Code의 논리적인 흐름을 읽을 수 있을 것이다. 위 소스 코드에서는 다행히 DB Table의 이름이 명시(Fix)되어 있고, where 절의 조건이 명시되어 있으므로 가장 안쪽의 SQL Block에 힌트 처리(/*+ INDEX_DESC ….*/)와 튜닝된 where 절 조건(rownum <= 10)이 처리되도록 넣으면 성능 튜닝 SQL절이 DBMS로 날아올 것이다.

항상 느끼는 것이지만 개발자의 성능(Performance)에 대한 관심이 문제가 된다. 조금만 성능을 고려하고 JSP Programming을 했더라면 이러한 코드는 나오지 않았을 것이다. 개발자들은 모든 것을 Procedural한 관점에서 이해하고자 하지만 SQL은 SET(집합)의 관점에서 받아들여 수행되어 진다는 것을 명심해야 한다.

2-3. SQL*Net 환경의 튜닝

Web Server와 DB Server간의 인터페이스에서 튜닝되어야 할 대상 중의 하나가 바로 SQL*Net Configuration이다. 너무 정형화된 listener.ora 파일과 tnsnames.ora 파일의 셋팅에 의하여 흔히 간과되는 것이 바로 SQL*Net 구성인데, 이 네트워크 환경을 잘 튜닝함으로써 40% 이상의 성능 개선을 이룬 사례도 있다. 어떤 SQL*Net 환경의 튜닝 요소가 있는 지 살펴 보자.

2-3-1. SQL*NET의 PACKET SIZE 조정 (SDU와 TDU)

SQL*NET에서는 tnsnames.ora와 listener.ora에 SDU와 TDU의 parameter를 추가함으로써 Session Data Unit과 Transport Data Unit의 size를 변경할 수 있다. 이는 SQL*NET을 경유하여 Network 상에서 보내지는 packet의 sizes를 변경할 수 있다는 것을 의미한다. 최대값은 32K이며 SQL*NET V2.3 이전 버젼에서 기본값은 2K이며 그 이상은 허용하지 않는다. 즉, 이러한 parameter들을 이용하여 services와 transport layer를 경유하여 전달되는 Oracle 의 데이터에 대한 각각의 buffer sizes를 조정할 수 있다. Oracle V7.3 이전에서는 SDU와 TDU를 2K로 제한하였으나 Oracle V7.3 이상에서는 이 parameter를 2K 이상으로 조정할 수 있게 되었다. SDU size의 경우는 Oracle 8 Release2 이상부터는 32K 이상까지도 가능하다.

또한 사용자는 SDU와 TDU의 값을 다르게 지정할 수도 있으나 이렇게 한다 하더라도 장점은 별로 없다. 예로 TDU는 1024로 SDU는 1536으로 설정을 하였다고 하더라도 사용자는 처음에는 512 bytes를, 그런 다음 1024 bytes를 보내는 것을 볼 수 있을 것이다.

예)

tnsnames.ora: 이 parameters는 반드시 DESCRIPTION 절 안에 있어야 합니다.

```
KRRCSUN =
     (DESCRIPTION =
         (SDU=8192)            <**** Calls to this alias will
         (TDU=8192)            <**** try to use 8K packets.
         (ADDRESS = (COMMUNITY = TCP.kr.oracle.com)
                        (PROTOCOL = TCP)
                        (HOST = krrcsun.kr.oracle.com)
                        (PORT = 1521)
                    )
                    (CONNECT_DATA =
                              (SID = ORA733)
                    )
     )
```

Web Server단의 tnsnames.ora 파일의 설정에 추가하여 준다.

예)

listener.ora : 이 Parameter는 반드시 SID_DESC 절 안에 있어야 합니다.

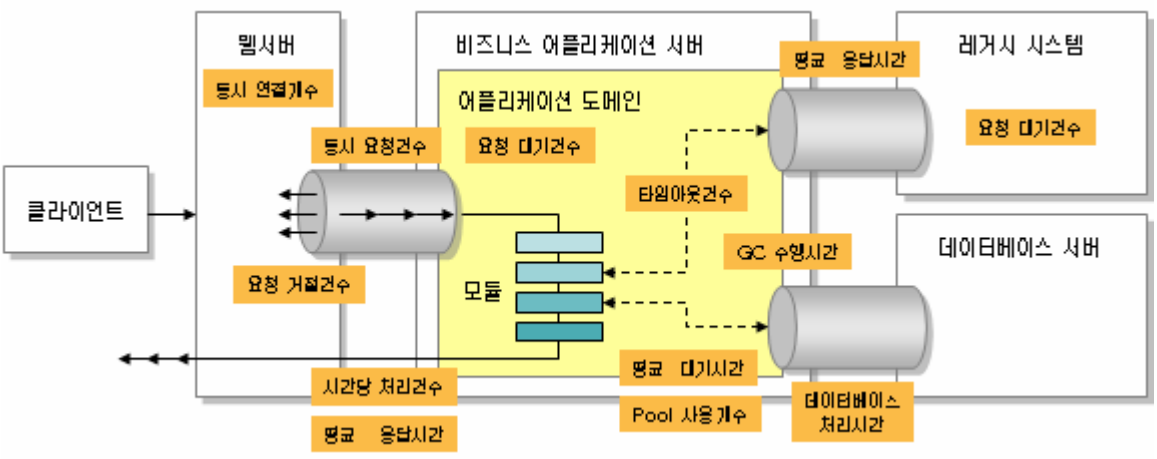
```
SID_LIST_LISTENER =
   (SID_LIST =
       (SID_DESC =
          (SDU = 8192)   <**** Connects to this SID
          (TDU = 8192)   <**** will try to use 8K.
          (SID_NAME = ORA733)
          (ORACLE_HOME = /oracle2/app/oracle/product/oracle8)
       )
      (SID_DESC = <*** This one will default generally to 2K
           (SID_NAME = V723)
           (ORACLE_HOME = /oracle/product/7.2.3)
      )
   )
```

DB Server 단의 listener.ora 파일에 설정하여 준다.

이러한 SQL*Net 레벨의 운반단위를 튜닝하는 것과 Application 레벨의 Array Fetch 크기를 같이 튜닝함으로써 SQL*Net의 Packet 운반 효율을 극대화 시킴으로써 좋은 튜닝 효과를 볼 수 있다.

3. 결 언

웹 서버가 정의된 성능 목표를 만족하는 지, 성능을 저하시키는 주요 병목 현상은 무엇인지, 성능 저하에 대한 최적의 개선방안은 무엇인지 등을 판단하기 위해서는 웹 서버를 포함한 전체 시스템 구성요소(서버, 네트워크, 데이터베이스 등)에 대한 성능을 측정하고 적절한 지표 값들을 수집하여 이를 종합적으로 분석하는 것이 필요하다. 이때, 사용되는 주요 성능 지표로 응답 시간, 시간당 처리량 및 자원 사용량(어플리케이션 수행 중 사용되는 CPU, 메모리, 디스크 I/O, 네트워크 대역폭 등) 등에 대하여 알아 보았다.



< 웹 서버의 주요 트랜잭션 구성도 >

웹 시스템 튜닝 가이드라는 제목하에 1,2 부에 걸쳐 데이터베이스(인스탄스 및 SQL), 운영체제, TCP/IP, SQL*Net 환경을 파헤쳐 보았다. 이 모든 부분을 한 명의 웹 서버 관리자가 전문적인 지식을 가지고 성능 조정(튜닝)한다는 것은 무리가 있다. 그럼에도 불구하고 웹 서버는 모든 조직의 상징적인 얼굴이 되어 조금 느리거나 문제가 발생하면 조직의 장에서부터 일반 사용자(End-User)에 이르기까지 그 원성의 대상이 되는 것이 상례가 되었다.

이 기술노트의 몇 줄 지식으로 모든 웹 서버 관리자가 모든 트랜잭션을 3초 이내의 초 고성능 웹 서버로 향상시켜 관리하게끔 하는 것은 꿈일 것이다. 그러나, 웹 서버의 향상을 위하여 관리 대상이 되는 요소들이 어떤 것들이며, 그 개선 방향이나 가능한 접근 방법이 이런 것들이 있다는 것을 안내하는 선에서 웹 서버 개선(튜닝) 가이드의 의도를 변(辯)하고자 한다.