Author: Sanjay J. Sontakke **Date : 10th Apr, 2011**

Performance optimization measures are seen at various levels. In this article we will look at optimization at middle layer where application developed using following technologies:

Layer	Technology	Usage
User Interface/Front End	Java Swing/ Web	Presentation Layer
Application Layer	JBoss	Application Server
Middle Layer	Pro*C executables	Business Layer
Data layer	Oracle Database	Business logic + Data

Platform – Red Hat Linux
*Here Pro*C executables are build using GCC compiler (Version 4.1.2).*

Summary: GCC optimizer setting "**O3**" can double the performance of your Pro*C code without making any code change.

Code sample with problem and its solution narrated with detailed information

Code Sample:

```
EXEC SQL DECLARE cur_emp CURSOR FOR
     SELECT emp_code, designation, sal
     FROM emp
     WHERE dept_code = :h_code
     ORDER BY dept_code,emp_code;

       EXEC SQL OPEN cur_emp;
       while(1)
       {
        Vempty(h_emp_code);
        Vempty(h_designation);
        h_sal=0;

        EXEC SQL FETCH cur_emp INTO :h_emp_code,:h_designation,:h_sal;

        h_designation.arr[h_designation.len] = '\0';
        h_emp_code.arr[h_emp_code.len] = '\0';
       }
```

Above code defines cursor for "emp" table and depending upon dept_code passed it pulls data from DB. Such type of code is common across application as its standard practice.

When we compile this code with required libraries and debug option (on/off). Execution behavior is

1. Open cursor
2. Initialize variables
3. Fetch record from DB session

Step 2 and 3 re-occurs till complete set of data retrieved.

This means as many number of records that cursor holds. In execution phase this process has to go back and forth to Data layer for record wise information. This is an over killing of IO and bandwidth. Following is runtime trace output of the execution process.

```
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\7\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\10\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\t\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\n\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\v\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\331\0\0\6\0\0\0\0\0\6\0\2\34\2\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 217
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\f\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\r\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\16\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\17\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\20\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\331\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 217
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\21\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\22\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\331\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 217
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\23\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\24\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\331\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 217
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\25\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\26\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\331\0\0\6\0\0\0\0\0\6\0\2\34\2\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 217
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\27\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\30\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\31\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\32\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\331\0\0\6\0\0\0\0\0\6\0\2\34\2\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 217
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\33\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\331\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 217
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\34\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\325\0\0\6\0\0\0\0\0\6\0\2\34\1\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 213
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5v\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\327\0\0\6\0\0\0\0\0\6\0\2\34\2\0\0\0\0\0\2\0\0\0\0\0\0\0\240\255|Z"..., 2064) = 215
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5w\2\0\0\0\2\0\0\0", 21) = 21
read(5, "\0\254\0\0\6\0\0\0\0\0\4\1\0\0\0v\0\1\343\0\0\0{\5\0\0\0\0\2\0\0\0"..., 2064) = 172
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 3), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x2b9ab3a9d000
write(1, "\n", 1
)                       = 1
write(1, " Executed 1 times ", 18 Executed 1 times )      = 18
exit_group(0)                           = ?
```

Actually speaking these round trips could be minimized using local memory of an execution server (Machine on this executable is residing). While executing it could go to Data layer and pull data in bulk and store it in local available memory. This will minimize server call and IOs.

This can be managed without making any code change. But, your compiler settings needs some changes –

How?

While compiling code along with other parameters give an optimizer parameter “**O3**” or “**Os**”. This enables “**fprefetch-loop-arrays**”. This feature force caching of data to the local memory of execution server and minimizes round trips. Following is an output post using this feature.

```
write(5, "\2*\0\0\6\0\0\0\0\0\21k\4\377\3\0\0\261\232\0\0\1\0\0\0\3^\5i\200\0\0"..., 554) = 554
read(5, "\2\30\0\0\6\0\0\0\0\0\20\27\204\230\27C\304b\n\2\202\277\307I>\224n\327xo\4\4"..., 2064) = 536
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\6\2\0\0\0e\0\0\0", 21) = 21
read(5, "\4\377\0\0\6\0\0\0\0\0\6\1\2\374\3\0\0\0\0\0e\0\0\0\0\0\0\0\0\0\0"..., 2064) = 1279
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\7\2\0\0\0e\0\0\0", 21) = 21
read(5, "\4\366\0\0\6\0\0\0\0\0\6\0\2\374\1\0\0\0\0\0e\0\0\0\0\0\0\0\240\255\233\270"..., 2064) = 1270
write(5, "\0\25\0\0\6\0\0\0\0\0\3\5\10\2\0\0\0e\0\0\0", 21) = 21
read(5, "\1\341\0\0\6\0\0\0\0\0\6\0\2\374\2\0\0\0\0\0e\0\0\0\0\0\0\0\240\255\233\270"..., 2064) = 481
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x2b3fddee2000
write(1, "\n", 1
)                       = 1
write(1, " Executed 1 times ", 18 Executed 1 times )      = 18
exit_group(0)                           = ?
```

What are Limitations?

This can use as much as available physical memory on local machine.
Record wise debugging of the code becomes little tricky.

Sanjay J. Sontakke – Database Architect
Designed and developed many critical applications requires high uptime.
Lead many performance tuning projects.