



Analysis of Algorithms

Unit 2 - Code Tuning Techniques



Code Tuning Techniques

- Code Tuning refers to modifying the implementation of a specific design rather than modifying the design itself
- Related to how to code better
- How to review a given code with performance as a concern



Copyright © 2004, Infosys
Technologies Ltd

2

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

Code Tuning Techniques (CTT):

- Used to write better code
- Needs better understanding of the programming language and its compiler
- Removes un-necessary portion of code
- Is equivalent of code optimization at Higher Language Level

In this chapter we will study some of the CTT that can be applied to looping constructs, logic expressions, type conversions and other expressions in a program.

CTT - Loops

- Looping constructs in a program are executed many times. There are quite a few of code tuning techniques which can be applied to looping constructs in order to improve the performance of the code.

Jamming of loops:

Jamming of loops is an outcome of combining loops which operate over the same range of values.

```
for (k = 1 to n) { initialize T[k] };  
for (m = 1 to n) { Max[m] = m * Max[m] };
```

**Before Jamming,
2n Loop Checks**

```
for (k = 1 to n) {  
    initialize T[k];  
    Max[k] = k * Max[k] };
```

**After Jamming,
n Loop Checks**



Copyright © 2004, Infosys
Technologies Ltd

3

ER/CORP/CRS/SE15/003

Version No: 2.0

Infosys®

Jamming of loops can be applied to other similar examples like finding the maximum and minimum, from a given set of elements, in one loop. In the above example, we have highlighted the fact that boundary checks in a loop are expensive. Loops which operate over different ranges cannot be jammed. We could face problems with jamming of loops if at a later stage the range for two jammed loops change!

CTT – Loops (Contd...)

Unswitching of Loops:

Unswitch loops that contain if tests,
if the results of these tests do not
change inside the loop

This is most suited for 1997-98
since the earlier budget had highest
slab of 40% and the new proposal
was with slab = 30% but the budget
was not passed because of
instability in the govt.

```
for (l=0; i<noofemployees; l++)  
{if budgetpassed = YES  
.....  
else  
.....  
}
```

Move the if condition outside the loop



Copyright © 2004, Infosys
Technologies Ltd

4

ER/CORP/CRS/SE15/003

Version No: 2.0

Infosys®

Switching basically refers to making a decision (using a selectional construct) inside a loop every time the loop is executed. If the result of the selectional statement does not change inside the loop then it makes more sense to unswitch the loop by making the decision outside the loop. In such cases the loop is turned inside out by putting the loop inside the selectional construct. The basic idea in unswitching is to minimize / remove unnecessary computation inside a looping construct.

CTT – Loops (Contd...)

Unrolling of Loops:

```
i = 1;  
while(i < num) {  
    a(i) = i;  
    i = i + 1; }
```

**Before unrolling,
n loop checks**

```
i = 1;  
while(i < num) {  
    a(i) = i;  
    a(i + 1) = i + 1;  
    i = i + 2; }
```

**After unrolling,
n/2 loop checks**



Copyright © 2004, Infosys
Technologies Ltd

5

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

Unrolling of loops reduces the amount of housekeeping done in loops. As shown in the example above the loop is rewritten (unrolled) so that the number of loop checks are reduced. If the number, *num*, is odd then the assignment statement for *a(num)* has to be done outside the loop.

A word of caution. Unrolling of loops helps in reducing the time taken by the code to execute but it affects the quality and readability of the code.

CTT – Loops (Contd...)

Minimize work performed inside loops:

```
For (i = 1 to n/2)
{
...
}
```

```
n_2 = n/2;
For (i = 1 to n_2)
{
...
}
```

Need to compute $n/2$ in every iteration is removed



Copyright © 2004, Infosys
Technologies Ltd

6

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

To minimize the work performed inside a loop, we need to move the constant expression evaluation and function calls outside the loop.

CTT – Loops (Contd...)

Use of Sentinel Values:

```
While (i < n) and (x < > a[i])  
{  
    i = i + 1;  
}
```

```
a[n+1] = x;  
While (x < > a[i])  
{  
    i = i + 1;  
}
```

One boundary check is reduced, as it is achieved by the sentinel



Copyright © 2004, Infosys
Technologies Ltd

7

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

Sentinel values can be used in a search loop which looks for an element in an array. If the required size of the array is 10, declare it as 11. In all search operation, use the 11th position to store the element to be searched.

CTT – Loops (Contd...)

Reduce the strength of operations performed inside loops:

All operations which do not depend on loop variant may be moved outside loops

```
for (i = 1 to Num ) {  
  commission ( i ) = i * Revenue * BaseCommission * Discount  
}
```

```
Commission = Revenue * BaseCommission * Discount  
for ( i = 1 to Num ) {  
  commission ( i ) = i * Commission  
}
```



Copyright © 2004, Infosys
Technologies Ltd

8

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

Strength reduction in a loop refers to replacing the operations such as multiplication by cheaper operations like addition.

CTT – Logic

Order tests in case statements by frequency:

```
read (empNo)
case Grade(empNo) of
1: {...}
2: {...}
3: {...}
....
7: {...}
endcase
```

```
read (empNo)
case Grade(empNo) of
7: {...}
5: {...}
6: {...}
....
1: {...}
endcase
```

**Every time, the more frequent, Grade 7 is encountered,
the code executes faster**



Copyright © 2004, Infosys
Technologies Ltd

9

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

This code tuning technique can be applied to *if-then-else* statement also!

CTT – Logic (Contd...)

Stop testing when you know the result:

```
if(a < 10 ) and (b < 20) then  
{  
  ....  
}
```

```
if(a < 10) then  
{  
  if(b < 20) then  
  {  
    ....  
  }  
}
```



Copyright © 2004, Infosys
Technologies Ltd

10

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

In the example above it makes more sense to stop testing for $b < 20$ when ever a is *not less than 10*. Some of the programming languages support this kind of evaluation which is known as **short circuit evaluation**. Else we need to re-write the code as shown above.

CTT – Data Transformations

Minimize array references:

If the same array element is repeatedly referred to inside a loop, then move it outside the loop

```
for (a=0; a < 5; a++)
{
    for (b =0; b < 10; b++)
    {
        total[b] = total[b] * sum[a];
    }
}
```

```
for (a=0; a < 5; a++)
{
    sum_now = sum[a];
    for (b =0; b < 10; b++)
    {
        total[b] = total[b] * sum_now;
    }
}
```



Copyright © 2004, Infosys
Technologies Ltd

11

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

Data Transformations:

- This code tuning technique aims at making small changes to the data structures in a code so that the performance of the code increases.
- In the above mentioned example, we see that the `sum[a]` does not change in the second loop and it is constantly referred to in the inner loop. This array reference can be minimized by moving it outside the inner loop as shown in the example.

CTT – Data Transformations (Contd...)

Augment data structures with indexes:

For example we can add an index to the linear linked list data structure. This index helps in speeding up the search operation in a linked list which is otherwise strictly linear.



Copyright © 2004, Infosys
Technologies Ltd

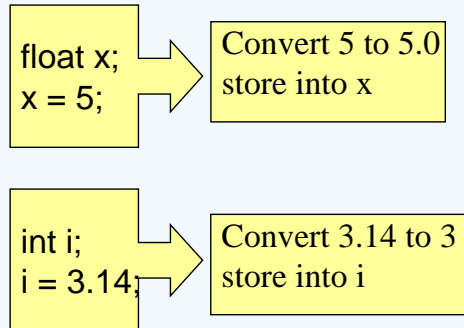
12

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

CTT – Expressions

Use constants of the correct type:



"Converting to a desired type" is an overhead



Copyright © 2004, Infosys
Technologies Ltd

13

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

Expressions:

- This section deals with the code tuning techniques that could be applied to the mathematical and logical expressions in a program so as to improve the efficiency of the code.
- Type conversions though done by the compiler are a costly overhead which can be avoided where ever possible, by the programmers.

CTT – Expressions (Contd...)

Precompute results:

```
y = log(x) / log(2)
b = log(a) / log(2)
```

Four Function Calls

instead have

```
LOG2 = log(2)
y = log(x) / LOG2
b = log(a) / LOG2
```

Two Function Calls



Copyright © 2004, Infosys
Technologies Ltd

14

ER/CORP/CRS/SE15/003

Version No: 2.0

Infosys®

Log(x) is function call for a given value of x. Log(2) is log of a constant. Hence it will be advisable to calculate the Log of a constant value and store it in a variable. This will reduce unnecessary computation of Log of a constant at run time.

Also when a particular result is used many times, then it advisable to precompute this result and store it so that when ever required a look up can be done. For example the result of a tax-saving calculation formula can be put in a lookup table.

If the time taken for a look up is more than the actual computation then the purpose of precomputation is defeated.

CTT – Expressions (Contd...)

Exploit Algebraic Identities:

- Algebraic identities can be used to replace costlier operations by cheaper ones
 - Whenever we need to find whether $\sqrt{x} < \sqrt{y}$, we can use the algebraic identity which says $\sqrt{x} < \sqrt{y}$ only when $x < y$. So it is enough to check if $x < y$ in this case.
 - ***not (A or B)*** is cheaper than ***not A and not B***



Copyright © 2004, Infosys
Technologies Ltd

15

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

CTT – Expressions (Contd...)

Strength Reduction in Expressions:

- Strength reduction refers to replacing costlier operations by cheaper ones. This can be achieved by:
 - Replacing multiplication with addition
 - Replacing exponentiation with multiplication...
 - $Ax^3 + Bx^2 + Cx + D$ is better computed as $((Ax + B)x + C)x + D$



Copyright © 2004, Infosys
Technologies Ltd

16

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

CTT – Expressions (Contd...)

Be Wary of System Routines:

- System Routines, like the math routines, provided accuracy which is more often wasted. If we do not need the level of accuracy as provided by the math routines then it makes sense for us to write the piece of code for the same. For example the math routine which computes ***log(x)*** provides the result in a floating point number whereas most of the times we might be interested only in the integral part.



Copyright © 2004, Infosys
Technologies Ltd

17

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

Summary

- Code Tuning Techniques for Loops
- Code Tuning Techniques for logic
- Code Tuning Techniques for Data Transformations
- Code Tuning Techniques for Expressions



Copyright © 2004, Infosys
Technologies Ltd

18

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®

In this chapter we had studied the Code Tuning Techniques:

- Which can be applied to looping constructs.
- Which covers the logic of the program
- Which aims at the modifying the implementation of data structures in a code
- Which deals with the mathematical and logical expressions in a code



Thank You!



Copyright © 2004, Infosys
Technologies Ltd

19

ER/CORP/CRS/SE15/003
Version No: 2.0

Infosys®