Optimization: Loop Optimization

CPSC 501: Advanced Programming Techniques
Fall 2022

Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

Wednesday, November 23, 2022

UNIVERSITY OF CALGARY

Code Tuning Loops

Code Tuning

- Loop techniques
  - Looping is often very important target for compiler and CPU optimizations (we can help it out)

- Guidelines:
  - Save each version of your code using version control
  - Use the profiler to find a bottleneck
  - Tune the bottleneck, using just one technique
  - Measure the improvement
    - If none, revert to the prior version
  - Repeat until desired performance is achieved

Unswitching

Loop Techniques - Unswitching

- Un-switching
  - Switching is where a decision is made inside a loop on every iteration
  - If the decision doesn't change while looping, ***unswitch it***
    - i.e. turn the loop inside out

Loop Techniques – Unswitching (cont'd)

- E.g.

```
for (int i = 0; i < count; i++) {
    if (type == NET) {
        netSum += amount[i];
    } else {
        grossSum += amount[i];
    }
}
```

Loop Techniques – Unswitching (cont'd)

```
if (type == NET) {
    for (int i = 0; i < count; i++) {
        netSum += amount[i];
    }
} else {
    for (int i = 0; i < count; i++) {
        grossSum += amount[i];
    }
}
```

- Note: two loops must now be maintained in parallel (design danger!!!)

Jamming

Loop Techniques – Jamming

- Jamming (fusion)
  - Combines two or more loops into one
    - Their loop counters should be similar
  - Reduces loop overhead
  - E.g.

```
for (int i = 0; i < employeeSalary.length; i++) {
    employeeSalary[i] = 0.0;
}
for (int  i = 0; i < employeeCode.length; i++) {
    employeeCode[i] = C;
}
```

Loop Techniques – Jamming (cont'd)

- Jammed version:

```
for (int i = 0; i < employeeSalary.length; i++) {
    employeeSalary[i] = 0.0;
    employeeCode[i] = C;
}
```

Unrolling

Loop Techniques - Unrolling

- Unrolling
  - A complete unrolling replaces a loop with straight-line code
    - Practical only for short loops
  - E.g.

```
for (int i = 0; i < 10; i++) {
    a[i] = i;
}
```

Loop Techniques – Unrolling (cont'd)

- Is replaced with:

```
a[0] = 0;
a[1] = 1;
a[2] = 2;
. . .
a[9] = 9;
```

- Maintenance!!!!

Loop Techniques – Partial Unrolling

- With partial unrolling, two or more cases are handled inside the loop instead of just one
- E.g.

```
for (int i = 0; i < count; i++) {
    a[i] = i;
}
```

- Unrolled once, becomes:

```
int i;
for (i = 0; i < count - 1; i += 2) {
    a[i] = i;
    a[i + 1] = i + 1;
}
if (i == count - 1) {
    a[count - 1] = count - 1;
}
```

Loop Techniques – Partial Unrolling (cont'd)

- Unrolled twice, becomes:

```
for (i = 0; i < count - 2; i += 3) {
    a[i] = i;
    a[i + 1] = i + 1;
    a[i + 2] = i + 2;
}
if (i == count - 2) {
    a[count - 2] = count - 2;
    a[count - 1] = count - 1;
} else if (i == count - 1) {
    a[count - 1] = count - 1;
}
```

Minimize work

Loop Techniques – Minimize Work

- Minimizing work inside loops
  - Put calculations that result in a constant before the loop
  - E.g.

```
for (int i = 0; i < rateCount; i++) {
    netRate[i] = baseRate[i] * rates.discount() / 0.93;
}
```

Loop Techniques – Minimize Work (cont'd)

- Is better as:

```
quantityDiscount = rates.discount() / 0.93;
for (int i = 0; i < rateCount; i++) {
    netRate[i] = baseRate[i] * quantityDiscount;
}
```

Sentinels

Loop Techniques - Sentinels

- **Sentinel Values**
  - Are used to simplify loop control
    - Replaces expensive compound tests
  - A sentinel is a special value that marks the end of an array
    - Is guaranteed to terminate a search through the loop
    - Declare the array one element bigger so it can hold the sentinel

Loop Techniques – Sentinels (cont'd)

- E.g.

```
found = false;
int i = 0;
while (!found && (i < count)) {
    if (item[i] == searchKey) {
        found = true;
    } else {
        i++;
    }
}
if (found) {
    // ...
}
```

Loop Techniques – Sentinels (cont'd)

- With a sentinel, becomes:

```
item[count] = searchKey;
i = 0;
while (item[i] != searchKey) {
    i++;
}
if (i < count) {
    // ...
}
```

Put sentinel at end of the array
(now bigger by one element)

Busy Loop Inside

Loop Techniques – Busy Loop Inside

- Putting the busiest loop on the inside
  - E.g.

```
for (int column = 0; column < 100; column++) {
    for (int row = 0; row < 5; row++) {
        sum += table[row][column];
    }
}
```

  - loop operations
    - Outer loop: 100
    - Inner loop: 100 * 5 = 500
    - Total: 600

Code Tuning

- Switching inner and outer loops, gives:

```
for (int row = 0; row < 5; row++) {
    for (int column = 0; column < 100; column++) {
        sum += table[row][column];
    }
}
```

- loop operations
  - Outer loop: 5
  - Inner loop: 5 * 100 = 500
  - Total: 505

Onward to …
more optimization.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY