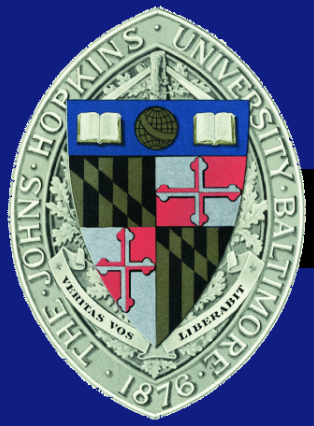
Lecture 8.1
Loop Dependencies

EN 600.320/420

Instructor: Randal Burns

22 February 2017



Department of Computer Science, *Johns Hopkins University*

The Loop Recipe

- Find the bottlenecks (profile)
- Eliminate loop carried dependencies
- Parallelize the loops
  - **Semantically neutral directives** are very helpful. This is perhaps the main factor behind OpenMP's success.
- Optimize the loop schedule
  - Load balance, avoid task skew, amortize startup

Loop Carried Dependencies

- When one iteration of a loop depends upon the computations of other iterations

- Can be addressed via loop rewriting
  - *Can't my compiler do this?*
- Removable dependencies
  - Code transformations
- Separable dependencies
  - Accumulation operations (mean, sum, count)
  - Extrema (max, min)
  - Connections to the reduce in map/reduce

Dependent Loop

```
int offset1 = c;
int offset2 = 0;

for ( int i=0; i<N; i++ )
{
  offset1 = offset1 + 1;
  d[offset1] = big_time_work ( offset1 );
  offset2 = offset2 + i;
  a[offset2] = other_big_calc ( offset2 );
}
```

Independent Loop

- Semantically equivalent loop with no dependencies

```
for ( int i=0; i<N; i++ )
{
  d[c+i] = big_time_work ( c+i );
  a[(i*i+i)/2] = other_big_calc ( (i*i+i)/2 );
}
```

Independent Loop

- Better example

Serial version containing flow dependency

```
for (i = 1; i < n; i++) {
    b[i] = b[i] + a[i - 1];
    a[i] = a[i] + c[i];
}
```

Parallel version with dependencies removed by **loop skewing**

```
b[1] = b[1] - a[0];
#pragma omp parallel for shared(a,b,c)
for (i = 1; i < n; i++) {
    a[i] = a[i] + c[i];
    b[i + 1] = b[i + 1] + a[i];
}
a[n - 1] = a[n - 1] + c[n - 1];
```

http://www.akira.ruc.dk/~keld/teaching/IPDC_f10/Slides/pdf4x/4_Performance.4x.pdf

Anti-dependency

Serial version containing anti dependency

```
for (i = 0; i < n; i++) {
    x = (b[i] + c[i]) / 2;
    a[i] = a[i + 1] + x;
}
```

Parallel version with dependencies removed

```
#pragma omp parallel for shared(a,a_copy)
for (i = 0; i < n; i++)
    a_copy[i] = a[i + 1];
#pragma omp parallel for shared(a,a_copy) private(x)
for (i = 0; i < n; i++) {
    x = (b[i] + c[i]) / 2;
    a[i] = a_copy[i] + x;
}
```

http://www.akira.ruc.dk/~keld/teaching/IPDC_f10/Slides/pdf4x/4_Performance.4x.pdf