Performance Profiling for Parallel Cilk Applications

James Thomas, T.B. Schardl, Charles Leiserson
Keel Foundation Undergraduate Research and Innovation Scholar

Cilk Background

```c
main() {
    compute();
    cilk_spawn compute();
    compute();
    cilk_sync;
    compute();
}
```

- Extensions to C/C++ to support fork (spawn)-join (sync) parallelism
- Programmer simply needs to specify code paths that can execute in parallel
- Runtime handles thread creation and assignment of work to threads

Callback API

Compiler inserts calls to callback functions in important points in Cilk program’s execution, and profiling tools can implement these callbacks to get data on program execution

```c
main()
{
    cilk_enter_spawning_fn();
    cilk_spawn compute();
    compute();
    cilk_sync;
    cilk_sync_end();
    cilk_leave_spawning_fn();
}
```

A sampling of inserted callbacks (smaller program than above)

DAG Modeling of Cilk Programs

- Each blue node represents a call to compute()
- The programmer has specified that the second and third calls can execute in parallel, and the fourth call must execute only after the second and third have completed
- The critical path length (span) is 3 calls and the total work is 4 calls

Span and Work Collection

Implementations of some of the callbacks for a tool that measures program work and span:

```c
// Points in Cilk program's execution
// Each blue node is a call to compute()
// What are work and span assuming these calls are program roots?
```

Additional Statistics

- Profiling information on parallel loops:
  ```c
cilk_for (int i = 0; i < n; i++)
    compute();
  ```

- Work and span information for all possible program roots while maintaining algorithmic efficiency of profiler:
  ```c
  main()
  {
    cilk_spawn compute();
    compute();
    cilk_sync;
  }
  ```

- Better data for recursive code

User Interface

- Report statistics in spreadsheet form (numbers are in units of time)

```plaintext
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Call Site</td>
<td>Work</td>
<td>% Program Work</td>
<td>% Program Work</td>
</tr>
<tr>
<td>2</td>
<td>fib.o.c</td>
<td>2000</td>
<td>0.666666667</td>
<td>0.66666667</td>
</tr>
<tr>
<td>3</td>
<td>fib.o.c</td>
<td>1000</td>
<td>0.333333333</td>
<td>0.333333333</td>
</tr>
</tbody>
</table>
```

- May add UI where these statistics are displayed on a function call graph

What are work and span assuming these calls are program roots?