Performance Profiling for Parallel Cilk Applications

James Thomas, T.B. Schardl, Charles Leiserson

Keel Foundation Undergraduate Research and Innovation Scholar

MIT EECS ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

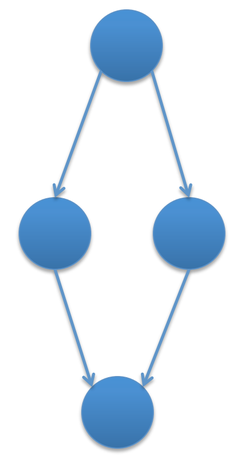
SuperUROP

Cilk Background

```
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

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

Callback API

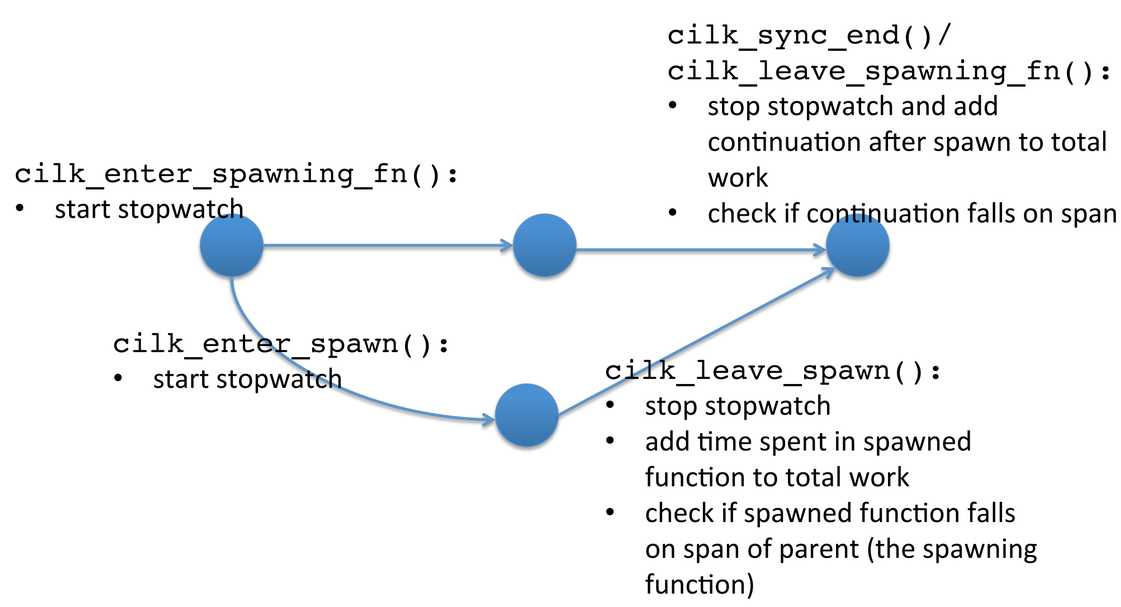
Compiler inserts calls to callback functions in important points in Cilk program's execution, and profiling tools can implement these callbacks to get data on program execution

```
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)

Span and Work Collection

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



Data on each function's contribution to work and span is also maintained in tables by the tool

Additional Statistics

- Profiling information on parallel loops:

```
cilk_for (int i = 0; i < n; i++)
   compute();
```

- Work and span information for all possible program roots while maintaining algorithmic efficiency of profiler:

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

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

- Better data for recursive code

User Interface

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

	A	B	C	D	E
1	Call Site	Work	Contribution to Program Span	% Program Work	% Program Span
2	fib.c:3	2000	95	0.666666667	0.95
3	fib.c:4	1000	5	0.333333333	0.05

fib.c

```
1: int fib(int n) {
2:    if (n <= 1) return n;
3:    int a = cilk_spawn fib(n – 1);
4:    int b = fib(n – 2);
5:    cilk_sync;
6:    return a + b;
7: }
```

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