Multicore challenges

Development time:
- Will you get your product out in time?
- Where will you find enough parallel-programming talent?
- Will you be forced to redesign your application?

Application performance
- How will you achieve superior processor utilization?
- Will your solution scale as the number of processor cores increases?

Software reliability
- How will you test your multicore applications?
Cilk Arts’ solution

**Development time:**
- No application redesign required.
- Can be learned in days by average programmers.
- Seamless path forward (and backward).

**Application performance**
- Best-in-class performance.
- Linear scaling as cores are added.
- Minimal overhead on a single core.

**Software reliability**
- Multithreaded version as reliable as the original.
About Cilk Arts

Mission:
To provide the easiest, quickest, and most reliable way to optimize application performance on multicore processors.

About:
- Incorporated in 2006 to commercialize 15 years of research.
- Launched in March 2007.
- Headquartered in Lexington, MA.

Team:
- Duncan McCallum, CEO.
- Prof. Charles E. Leiserson, CTO.
- Matteo Frigo, Chief Scientist.
- Steve Lewin-Berlin, VP Engineering and Operations.
What is Cilk?
A C language for programming dynamic multithreaded applications on shared-memory multiprocessors.

Applications:
- Virus shell assembly.
- Graphics rendering.
- \( n \)-body simulation.
- Chess programs.
- Dense and sparse linear algebra.
- Lattice-Boltzmann methods.
- Chip power grid analysis.

Provably good runtime system:
Automatically manages low-level aspects of parallel execution, including protocols, load balancing, and scheduling.
Cilk is a **faithful** extension of C/C++. The **serial elision** of a Cilk program is a valid implementation. Cilk provides no new data types.
Basic Cilk keywords

Fibonacci

cilk int fib(int n)
{
    if (n < 2) return n;
    else {
        int x, y;
        x = spawn fib(n - 1);
        y = spawn fib(n - 2);
        sync;
        return x + y;
    }
}

cilk:
Identifies a function as a Cilk procedure, capable of being spawned in parallel.
Fibonacci

cilk int fib(int n)
{
    if (n < 2) return n;
    else {
        int x, y;
        x = spawn fib(n - 1);
        y = spawn fib(n - 2);
        sync;
        return x + y;
    }
}

cilk:
Identifies a function as a Cilk procedure, capable of being spawned in parallel.

spawn:
The child procedure can be executed in parallel with the parent.
Basic Cilk keywords

Fibonacci

cilk int fib(int n) {
    if (n < 2) return n;
    else {
        int x, y;
        x = spawn fib(n - 1);
        y = spawn fib(n - 2);
        sync;
        return x + y;
    }
}

cilk: Identifies a function as a Cilk procedure, capable of being spawned in parallel.

spawn: The child procedure can be executed in parallel with the parent.

sync: Cannot be passed until all spawned children have returned.
Dynamic multithreading

Example: fib(4)

cilk int fib(int n) {
    if (n < 2) return n;
    else {
        int x, y;
        x = spawn fib(n - 1);
        y = spawn fib(n - 2);
        sync;
        return x + y;
    }
}

The computation dag unfolds dynamically.
Cactus stack

Cilk supports C’s rule for pointers:
A pointer to stack space can be passed from parent to child, but not from child to parent. (Cilk also supports malloc/new.)

Cilk’s cactus stack supports several stack views in parallel.

cilk void A(void)
{
    spawn B();
    spawn C();
}

cilk void C(void)
{
    spawn D();
    spawn E();
}
Cilk’s thread scheduler

Randomized work stealing
Load-balances the computation at run-time.

Scales down
Slowdown of Cilk program on 1 processor vs. the serial elision is negligible.

Scales up
A mathematical proof guarantees near-perfect linear speed-up on applications with sufficient parallelism, as long as the architecture has sufficient memory bandwidth.

Spawn is cheap
A spawn/return in Cilk is over 450 times faster than a pthread create/exit and about 3 times slower than an ordinary C function call on contemporary x86 processors.
The *Nondeterminator* debugging tool:

Provably guarantees to detect and localize data-race bugs.

**Data race:**

Occurs whenever a thread modifies a location and another thread, holding no locks in common, accesses the location simultaneously.
Advanced features

Support for nondeterministic programs:

- The **inlet** keyword specifies an internal function that can be called to incorporate a returned result into the parent frame in a nonstandard way when a spawned child returns.
- The **abort** keyword forces all spawned children to terminate abruptly.
- The **SYNCHED** pseudovariable tests whether a sync would succeed.
- A Cilk library provides **mutex locks** for atomicity.
2006 Class 2 Award for Best Overall Productivity

<table>
<thead>
<tr>
<th>P</th>
<th>HPL Gflop/s</th>
<th>η</th>
<th>DGEMM Gflop/s</th>
<th>η</th>
<th>STREAM GB/s</th>
<th>η</th>
<th>PTRANS GB/s</th>
<th>η</th>
<th>FFTE Gflop/s</th>
<th>η</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.2</td>
<td></td>
<td>5.1</td>
<td></td>
<td>0.8</td>
<td></td>
<td>0.7</td>
<td></td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9.4 89</td>
<td></td>
<td>9.7 96</td>
<td></td>
<td>0.9 56</td>
<td></td>
<td>0.5 36</td>
<td></td>
<td>0.9 67</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>17.3 85</td>
<td></td>
<td>19.7 97</td>
<td></td>
<td>1.8 57</td>
<td></td>
<td>0.9 33</td>
<td></td>
<td>1.8 68</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>30.8 73</td>
<td></td>
<td>35.7 88</td>
<td></td>
<td>2.9 46</td>
<td></td>
<td>1.7 30</td>
<td></td>
<td>2.9 55</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>52.5 63</td>
<td></td>
<td>64.9 80</td>
<td></td>
<td>4.0 32</td>
<td></td>
<td>3.3 29</td>
<td></td>
<td>4.0 38</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>88.6 52</td>
<td></td>
<td>118.9 73</td>
<td></td>
<td>6.8 27</td>
<td></td>
<td>6.1 27</td>
<td></td>
<td>6.8 32</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>101.6 30</td>
<td></td>
<td>248.0 76</td>
<td></td>
<td>14.0 28</td>
<td></td>
<td>11.6 26</td>
<td></td>
<td>14.0 33</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>463.1 71</td>
<td></td>
<td>25.0 25</td>
<td></td>
<td>18.3 20</td>
<td></td>
<td>25.9 31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>943.0 73</td>
<td></td>
<td>44.2 22</td>
<td></td>
<td>27.2 15</td>
<td></td>
<td>49.5 29</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Kuszmaul 2006], SGI Altix 3700.
Productivity

Only submission that implemented all six benchmarks

One programmer, about one week of work.

Cilk-ifying all six benchmarks required 137 keywords

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>MPI SLoC</th>
<th>Cilk SLoC</th>
<th>Dist. to Multicore</th>
</tr>
</thead>
<tbody>
<tr>
<td>STREAM</td>
<td>658</td>
<td>58</td>
<td>11</td>
</tr>
<tr>
<td>PTRANS</td>
<td>2261</td>
<td>81</td>
<td>13</td>
</tr>
<tr>
<td>RandomAccess</td>
<td>1883</td>
<td>123</td>
<td>18</td>
</tr>
<tr>
<td>HPL</td>
<td>15608</td>
<td>348</td>
<td>41</td>
</tr>
<tr>
<td>DGEMM</td>
<td>184†</td>
<td>97</td>
<td>19</td>
</tr>
<tr>
<td>FFTE</td>
<td>1747</td>
<td>230</td>
<td>35</td>
</tr>
</tbody>
</table>

†MPI DGEMM uses the HPL parallel matrix multiplication.
Conclusion

Cilk offers:

- Compatibility with existing C/C++ serial code.
- Lightweight primitives for expressing parallelism.
- Robust load-balancing scheduler.
- Tools to ensure parallel correctness.