A Course-Based Usability Analysis of Cilk Plus and OpenMP

Michael Coblenz, Robert Seacord, Brad Myers, Joshua Sunshine, and Jonathan Aldrich

PROGRAMMING PARALLEL SYSTEMS

- Parallel programming is notoriously hard
  - Must coordinate work of many different processing units

C LANGUAGE PARALLEL PROGRAMMING

- C has only low-level parallel programming features
- The CPLEX study group wants to fix this!
- Can they decide on a human-centered basis?

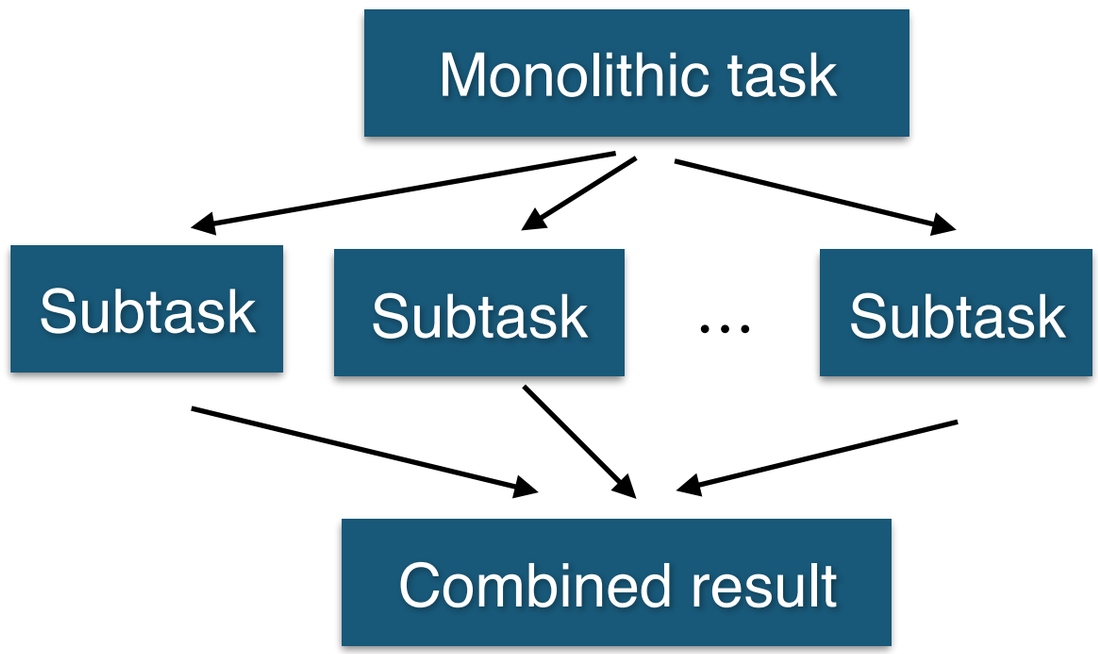

B5112

THE DESIGN SPACE

- Cilk Plus and OpenMP: both existing, popular approaches
- Both use shared-memory, fork-join parallelism

FORK-JOIN PARALLELISM

- Overall approach in Cilk Plus and OpenMP: split task into subtasks; assign subtasks to different threads

- 

Tasks, e.g. loop iterations, are split across threads

Reducers combine partial results

OpenMP vs. Cilk Plus

- OpenMP: very popular approach using compiler directives
  - Introduced in 1997
  - Managed by OpenMP ARB
  - Came from industry. Supports FORTRAN too.
- Cilk Plus: now owned by Intel, originated at MIT
  - Introduced in 1995
  - Keyword-based approach

EMPIRICAL COMPARISON

- Does one approach lead to fewer bugs?
  - Faster performance?
  - Faster task completion times?
- Preliminary study; didn't plan for statistical significance

METHOD

- Master's level Secure Coding class assignment
- 9 students; 8 submitted the assignment; 8 participated in experiment
- Experienced programmers but no apparent experience with parallel programming
- Gave lectures on OpenMP, Cilk Plus, parallel programming (including race conditions)

TASK

- Parallelize provided serial anagram-finding code *twice*
- Told to use reducers and get speedup ≥ 1.5
- All students used both extensions
  - controlled for ordering effects
- Students given VM with Eclipse + Fluorite

CILK PLUS CRASH COURSE (1)

```
cilk_for (int i = 0; i < 10; i++) {
  printf("Hello, world!");
}
```

CILK PLUS CRASH COURSE (2)

```
CILK_C_DECLARE_REDUCER(results_t) results =
 CILK_C_INIT_REDUCER(
  results_t, reduce, identity, destroy
 );
```

```
cilk_for (int i = 0; i < word_len - pos; i++) {
 find_anagrams(
   dict, permutations[i], results, word_len, pos+1
 );
}
…
results_append(&REDUCER_VIEW(*results), word);
```

OPENMP CRASH COURSE (1)

```
#pragma omp parallel for {
    for (int i = 0; i < 10; i++) {
     printf("Hello, world!");
    }
}
```

OPENMP CRASH COURSE (2)

```
#pragma omp declare reduction
 (results_reduction :
   results_t :
   results_reduce(&omp_out, &omp_in)
  )
 initializer(results_init(&omp_priv))

results_t results; …

#pragma omp parallel for reduction(results_reduction:
results)
  for (int i = 0; i < word_len; i++) {
   find_anagrams(
    dict, permutations[i], &results, word_len, 1
   );
  }
```

WHICH DO YOU PREFER?

```
CILK_C_DECLARE_REDUCER(results_t) results =
 CILK_C_INIT_REDUCER(
  results_t, reduce, identity, destroy
 );
```

```
#pragma omp declare reduction
 (results_reduction :
   results_t :
   results_reduce(&omp_out, &omp_in)
  )
 initializer(results_init(&omp_priv))
```

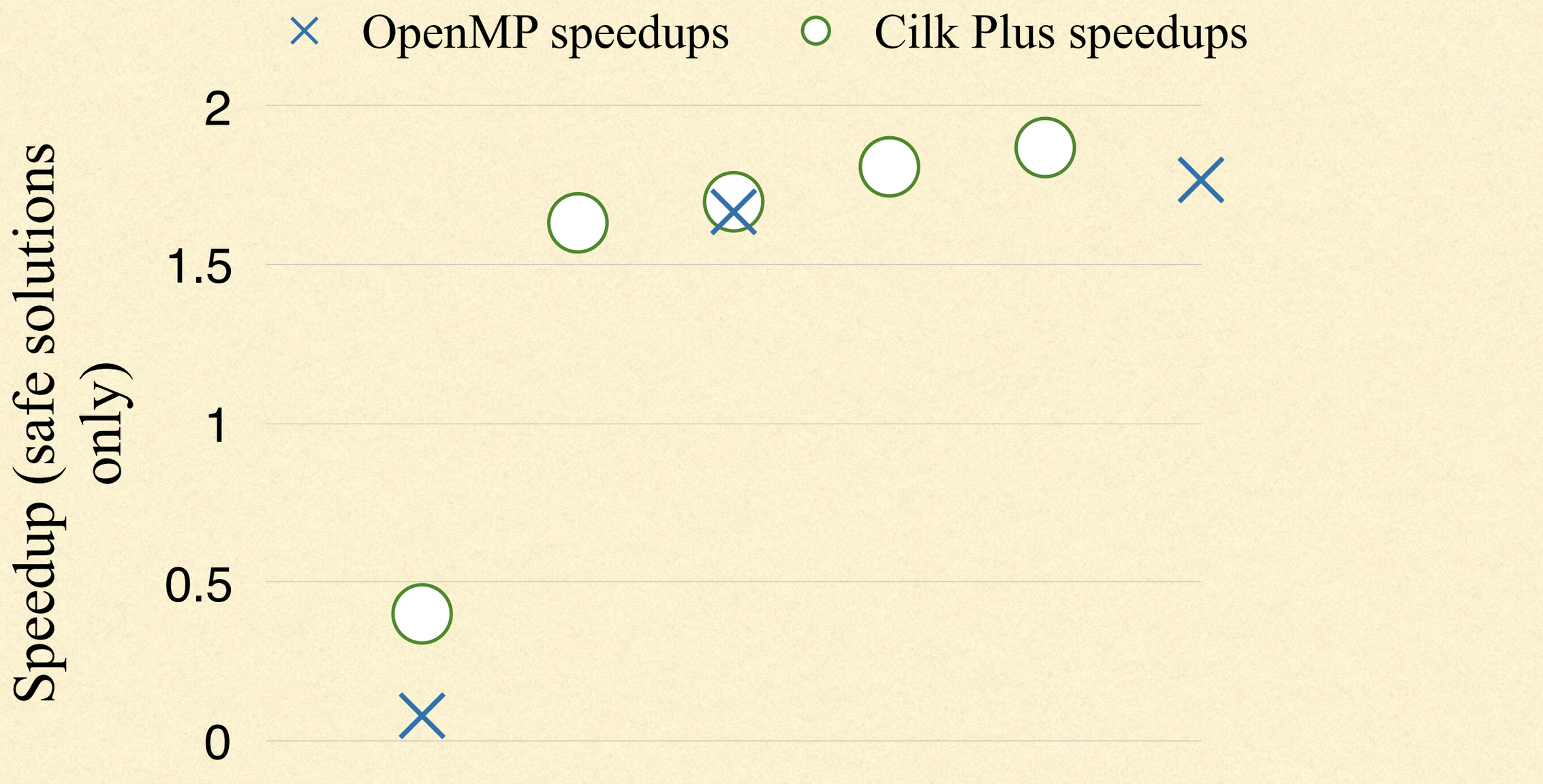
SUMMARY OF RESULTS

	Cilk Plus	OpenMP
Number of correct programs	5	3
Average speedup	1.5	1.2
Number of correct programs with speedup at least 1.5	4	2

CORRECTNESS

- 4/8 OpenMP solutions attempted to use reducer
  - One tried but failed; three were successful
- 4/8 OpenMP solutions didn't use reducers at all (but two tried)
  - One tried to use #pragma omp critical but neglected {}
- 8/8 Cilk Plus solutions attempted to use reducer
  - Two tried but failed (one declared reducer but didn't use it; one called REDUCER_VIEW outside parallel region)

PERFORMANCE



ESTIMATED TIME ON TASK

Estimates (hours) based on Fluorite logs

	OpenMP average task time	Cilk Plus average task time	Total task time
OpenMP first, Cilk Plus second	11	4	15
OpenMP second, Cilk Plus first	3	9	12

Second task First task

HYPOTHESES

- Parallel programming languages cannot be used safely (yet) by naïve programmers 😱
- Distinct reducer, value types may reduce error rates
- Reducer syntax in OpenMP impedes programmers

LIMITATIONS

- Small sample size
- Results affected by instruction and provided materials
- One small task with learning effects
  - Short time frame
  - Small code size
- Novices in these extensions specifically; students in general

CONCLUSIONS

- Fork/join parallel programming cannot be used safely (yet) by naïve programmers 😱
  - Maybe we can design better languages, tools, or training
- OpenMP seems to be harder for novices to use than Cilk Plus
- Differences seem to affect productivity and correctness — study is worthwhile

THANKS!

- Thanks to the DoD, NSA and NSF for support, and to our anonymous reviewers

- Contact: Michael Coblenz (mcoblenz@cs.cmu.edu)