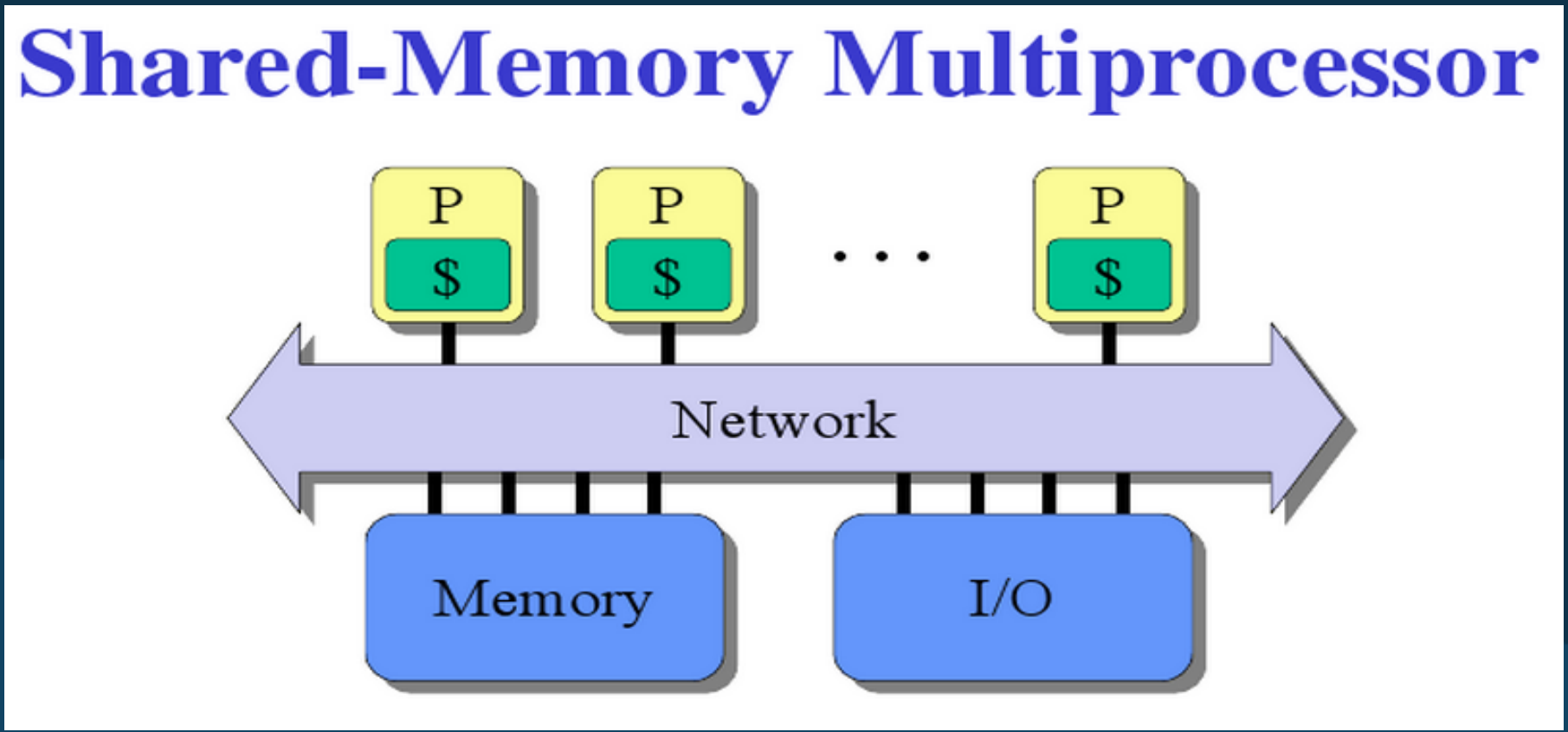
Multithreaded Programming in

Cilk (MIT Prof. Charles E. Leiserson)
Cilk++ (Cilk Arts)
Cilk Plus (Intel)

Emrah SARI
200411209

Cilk

A C language for programming dynamic multithreaded applications on shared-memory multiprocessors.



Cilk Is Simple

- Cilk extends the C language with just a handful of keywords.
- Every Cilk program has a serial semantics.
- Not only is Cilk fast, it provides performance guarantees based on performance abstractions.
- Cilk is processor-oblivious.
- Cilk's provably good runtime system automatically manages low-level aspects of parallel execution, including protocols, load balancing, and scheduling.
- Cilk supports speculative parallelism.

Comparing Cilk++ and OpenMP

The *omp task* and the Cilk++ *spawn/sync* are very similar.

" If your code looks like a sequence of parallelizable Fortran-style loops, OpenMP will likely give good speedups. If your control structures are more involved, in particular, involving nested parallelism, you may find that OpenMP isn't quite up to the job: "

OpenMP Fibonacci

```
#pragma omp parallel
/*Parallelregion,ateamofthreadsiscreated*/
#pragma omp single
{
/*Executed bythefirstthread*/
fib_result=fib(n);
}
}/*Endofparallelregion*/

int fib ( int n) {
int x , y ;
if (n < 2)
return n ;
else {
#pragma omp taskshared ( x )
x = fib ( n - 1);
/* A new task */
#pragma omp taskshared ( y )
y = fib ( n - 2);
/* A new task */
#pragma omp taskwait
/* Wait for the two tasks above to complete */
return x + y ;
}
}
```

Cilk++ Fibonacci

```
int fib ( int n) {
int x , y ;
i f (n < 2)
return n ;
else {
x = cilk_spawn f i b ( n – 1);
y = cilk_spawn f i b ( n – 2);
cilk_sync ;
return x + y ;
}
}
```

Parallel QuickSort using OpenMP and Cilk++

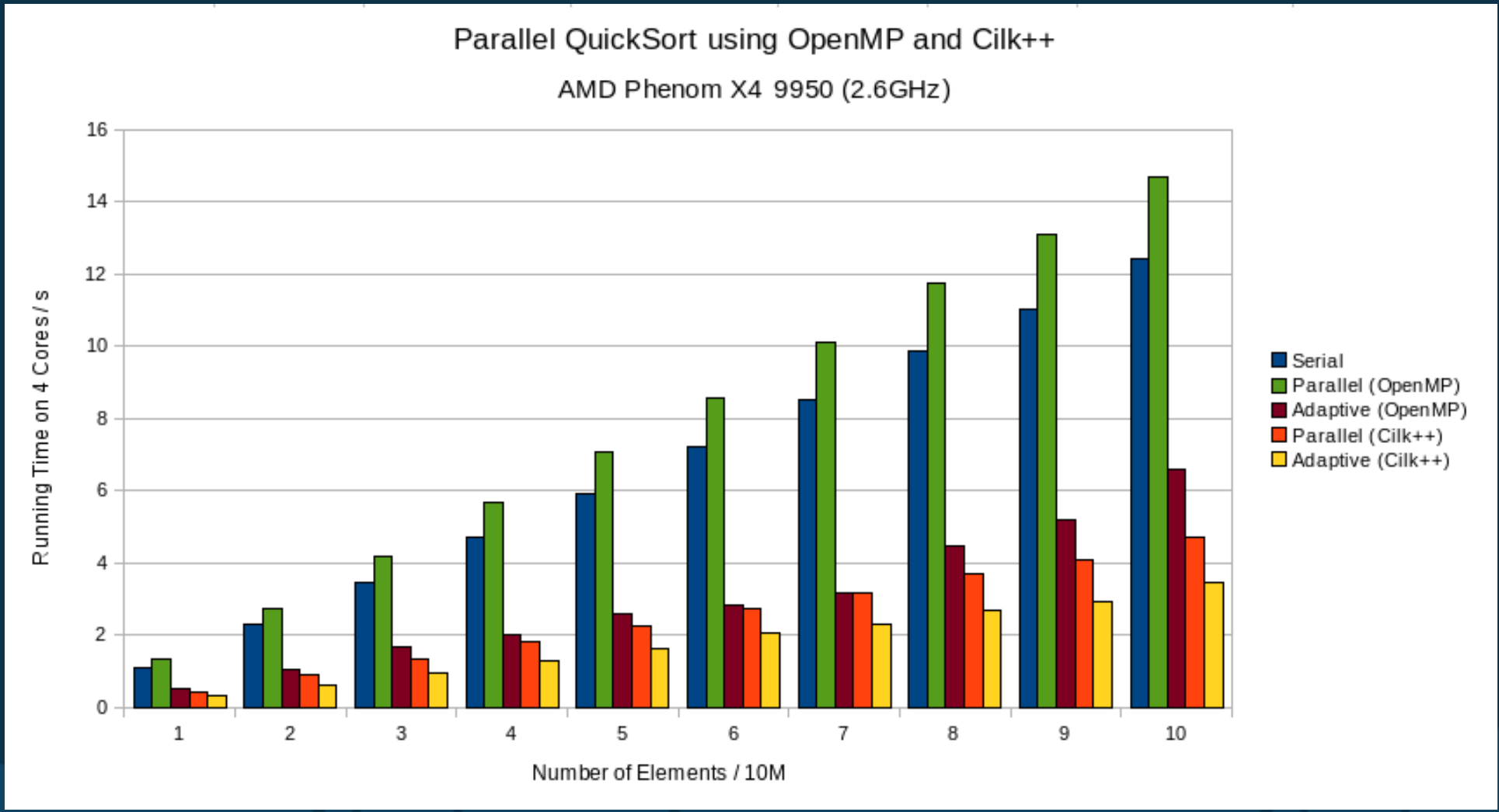
```
$ OMP_NUM_THREADS=1 ./a.out 35

Serial fib(35) = 9227465 Time: 0.0800
Parallel fib(35) = 9227465 Time: 3.0700
Parallel fib(35) cutoff(30) = 9227465 Time: 0.0700
$ OMP_NUM_THREADS=2 ./a.out 35
Serial fib(35) = 9227465 Time: 0.1100
Parallel fib(35) = 9227465 Time: 11.0000
Parallel fib(35) cutoff(30) = 9227465 Time:
0.060000
$ OMP_NUM_THREADS=4 ./a.out 35
Serial fib(35) = 9227465 Time: 0.0800
Parallel fib(35) = 9227465 Time: 16.8300
Parallel fib(35) cutoff(30) = 9227465 Time: 0.0400
```

```
$ CILK_NPROC=1 ./a.out 35

Serial fib(35) = 9227465 Time: 0.0900
Parallel fib(35) = 9227465 Time: 0.7300
Parallel fib(35) cutoff(30) = 9227465 Time: 0.0700
$ CILK_NPROC=2 ./a.out 35
Serial fib(35) = 9227465 Time: 0.0800
Parallel fib(35) = 9227465 Time: 0.4900
Parallel fib(35) cutoff(30) = 9227465 Time: 0.0400
$ CILK_NPROC=4 ./a.out 35
Serial fib(35) = 9227465 Time: 0.1300
Parallel fib(35) = 9227465 Time: 0.2600
Parallel fib(35) cutoff(30) = 9227465 Time: 0.0200
```

Parallel QuickSort using OpenMP and Cilk++



Cilk++

- Simple keywords

  Simple, powerful expression of task parallelism:
  cilk_for – Parallelize for loops
  cilk_spawn – Specify the start of parallel execution
  cilk_sync – Specify the end of parallel execution

When to use Intel Cilk Plus over other Parallel Methods?

- simple expression of opportunities for parallelism, rather than control of execution to perform operations on arrays

- higher performance obtainable with inherent data parallelism semantics – array notation

- to use native programming, as opposed to managed deployment: no managed runtime libraries – you express the intent

- to mix parallel and serial operations on the same data