A Comparative Study about Task Parallelism in OpenMP and Cilk

Guilherme Rezende Alles, Lucas Mello Schnorr
{gralles, schnorr}@inf.ufrgs.br

WSPPD
Porto Alegre, September 2nd, 2016

Task Parallelism

- Fundamental parallel abstraction
- Tasks defined as functions
- Function/task independency
- Task pool abstraction
  - Worker threads get tasks from the pool
- Tradeoffs: granularity *vs* overhead
  - Fine grain: better workload distribution
  - Coarse grain: less task management overhead

OpenMP and Cilk

Cilk

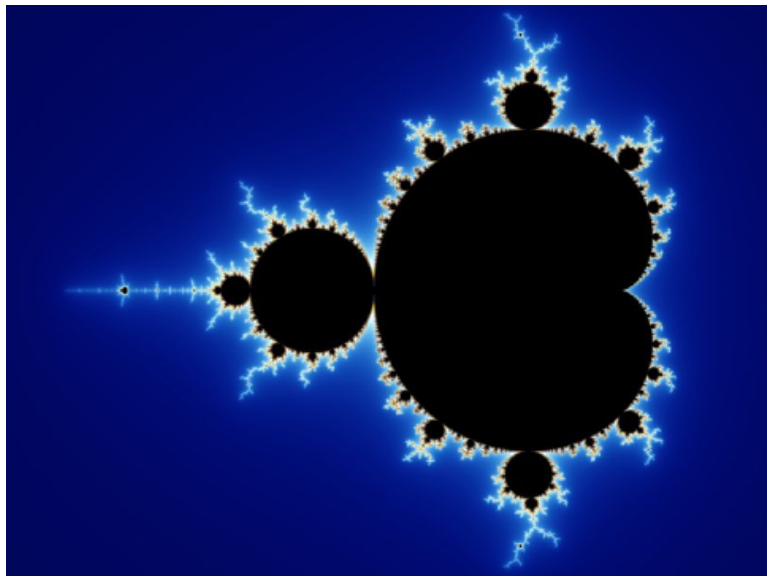
- Runtime library
- Simpler constructs
  - *cilk_spawn*, *cilk_for*, *cilk_sync*
- More independent runtime
- Parallelism is expresssed, but not commanded
  - Work stealing routine handles the distribution of tasks

OpenMP

- Compiler directives + runtime library
- Supports data and task parallelism
- Large control over runtime execution settings
  - Threads scheduler, synchronization, shared memory
  - Lower overhead (probably)

Mandelbrot

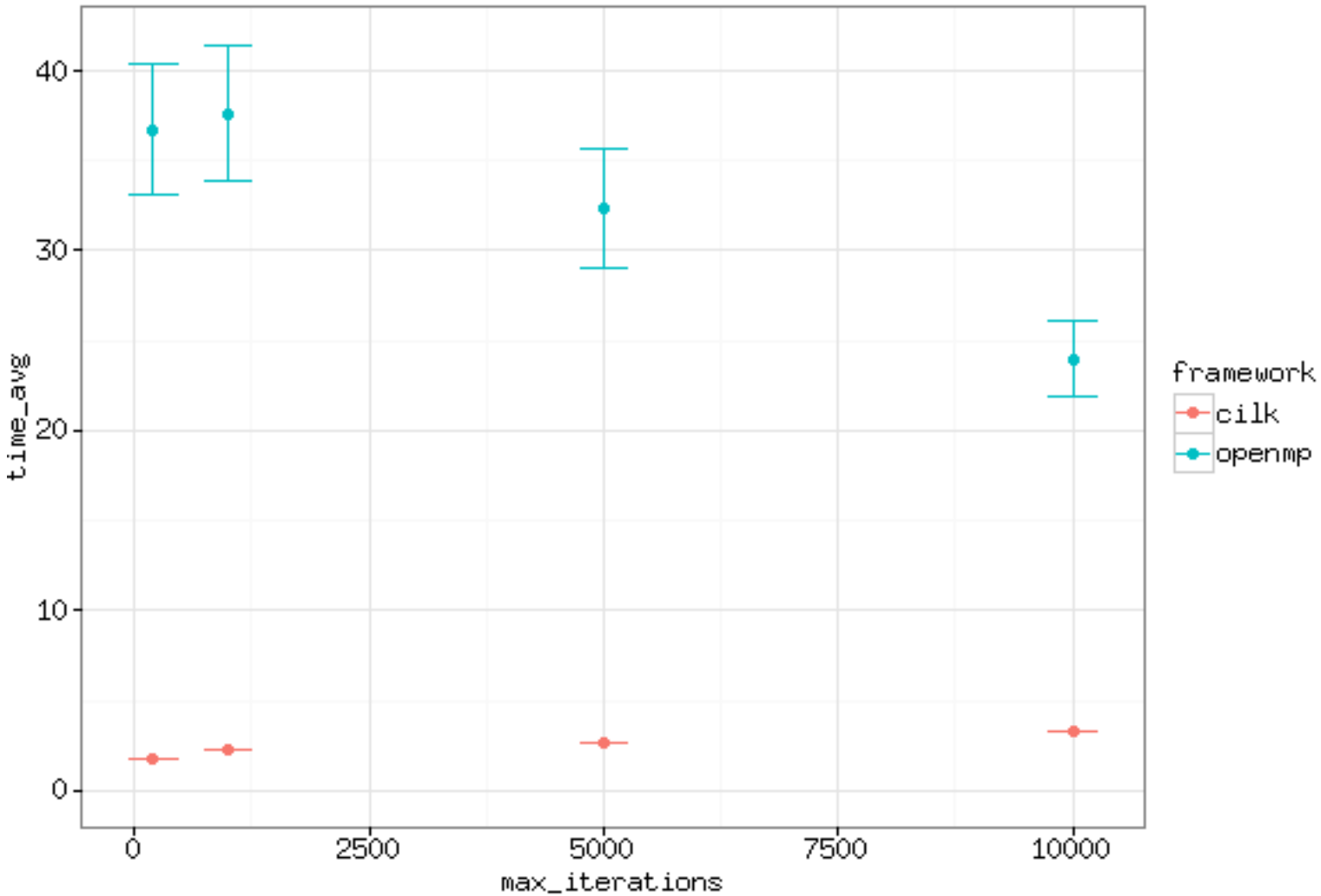
- Set of complex numbers
  - Don't diverge when applied to a mathematical function
- Simple algorithm
- Classic irregular workload problem
- Used to asses how well each framework adapts to irregularity



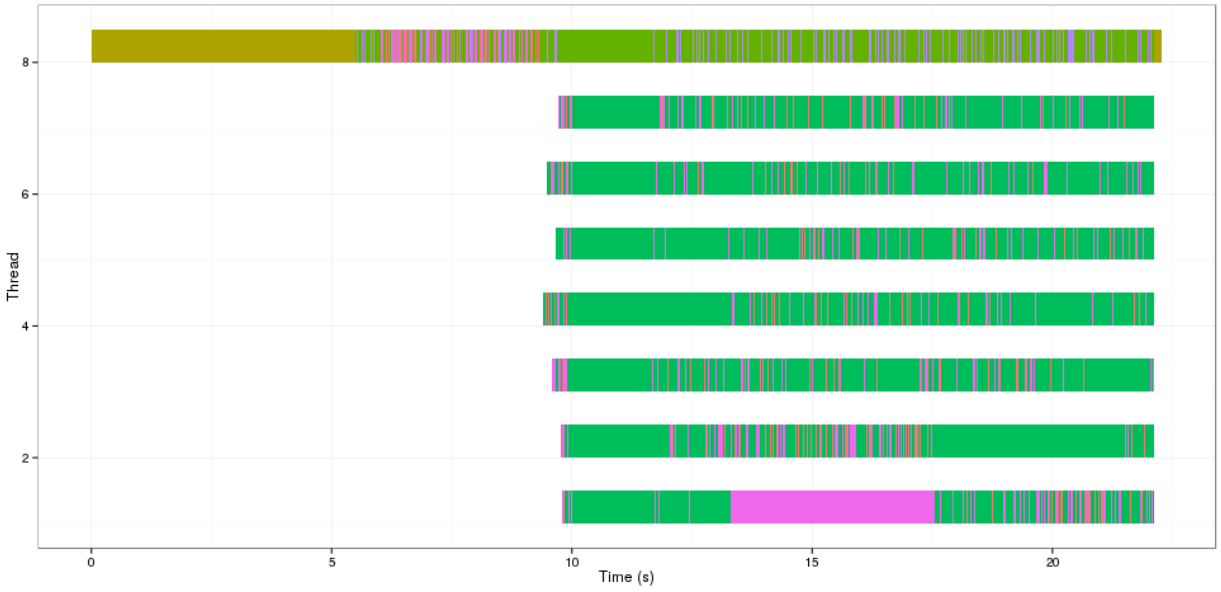
Design of Experiment

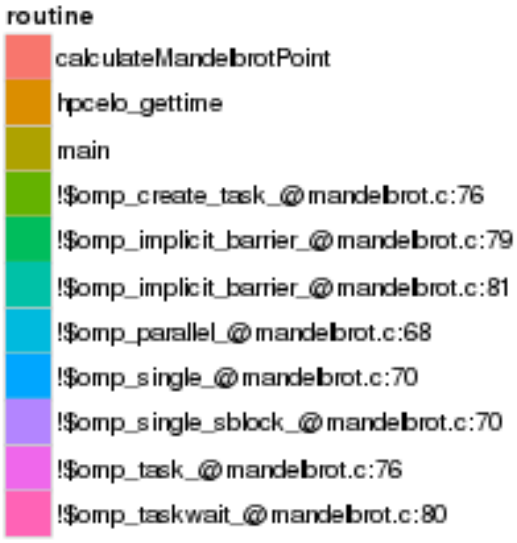
- Full factorial design
  - Factors: maximum iterations and grain size
- 20 replications, each generating a 2000x2000 image
- Reproducible!
  - Execution configs in a CSV file (generated in R)
  - Bash script to run the experiment and save results
- Hardware: Bali2
  - Dual Xeon E5-2650 machine
  - 32 Hardware threads total
  - 32 GB RAM

Grain size 1

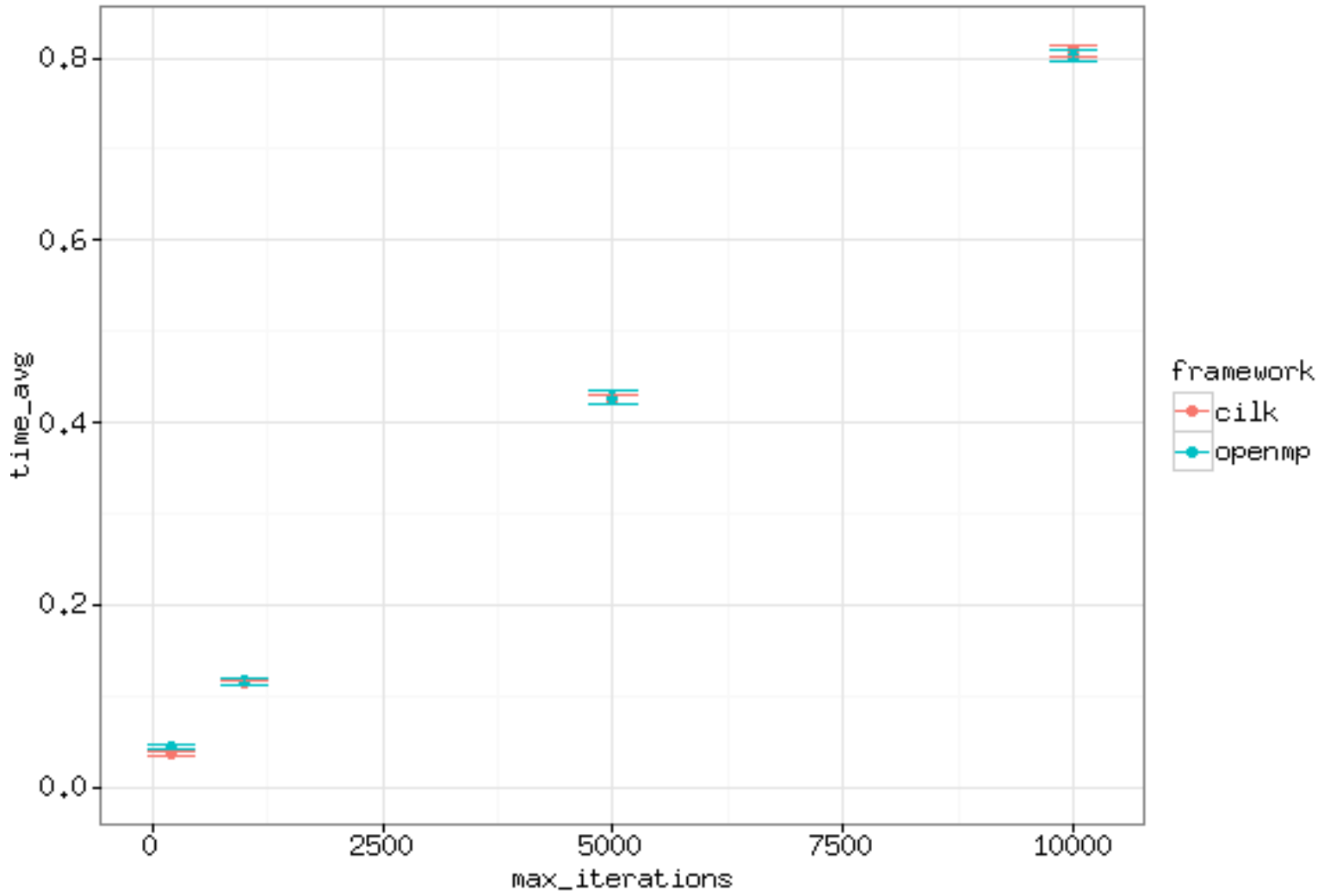

time_avg
40
30
20
10
0
0
2500
5000
7500
10000
max_iterations
framework
cilk
openmp

OpenMP Trace





Grain size 2000



Conclusion

- OpenMP is somewhat sensitive to the creation of tasks
  - Cause: implicit barriers
- Cilk's runtime and work-stealing routine is much more efficient
  - Explosion of tasks is not an issue
- If the grain is sufficiently big, their performance is comparable

Future work

- Where do these barriers come from?
- Trace the coarse-grained application
  - Ensure there is no unnecessary barriers there
  - OpenMP might be faster
- Study ways to improve the OpenMP scheduler
  - Load balancing issue
- Incorporate StarPU in the analysis

Thanks for your attention!

Questions?