

# **Performance Study of a Whole Genome Comparison Tool on a Hyper-Threading Multiprocessor**

**Juan del Cuvillo<sup>1</sup> , Xinmin Tian<sup>2</sup>, Guang R. Gao<sup>1</sup> and  
Milind Girkar<sup>2</sup>**

(1) Department of Electrical and Computer Engineering, University of Delaware,  
DE 19716 Newark, USA

(2) Intel Compiler Laboratory, SSG/EPG, Intel Corporation, 3600 Juliette Lane,  
CA 95052 Santa Clara, USA

Presented By: Ratna Singh  
Department of Computer Science, NCSU

# Agenda

- Understanding Genomics.
- Bioinformatics Key Applications.
- Motivation for ATGC.
- ATGC (Another Tool for Genome Comparison).
- Parallel Computation of Sequence Alignment.
- Performance Analysis.
- Future Direction.

# Genetics Glossary

- **Genomics**

- study of genes and their function.

- **Genomic sequence / DNA** (deoxyribonucleic acid)

- molecule that encodes genetic information.

- DNA is a double-stranded molecule held together by weak bonds between base pairs of nucleotides.

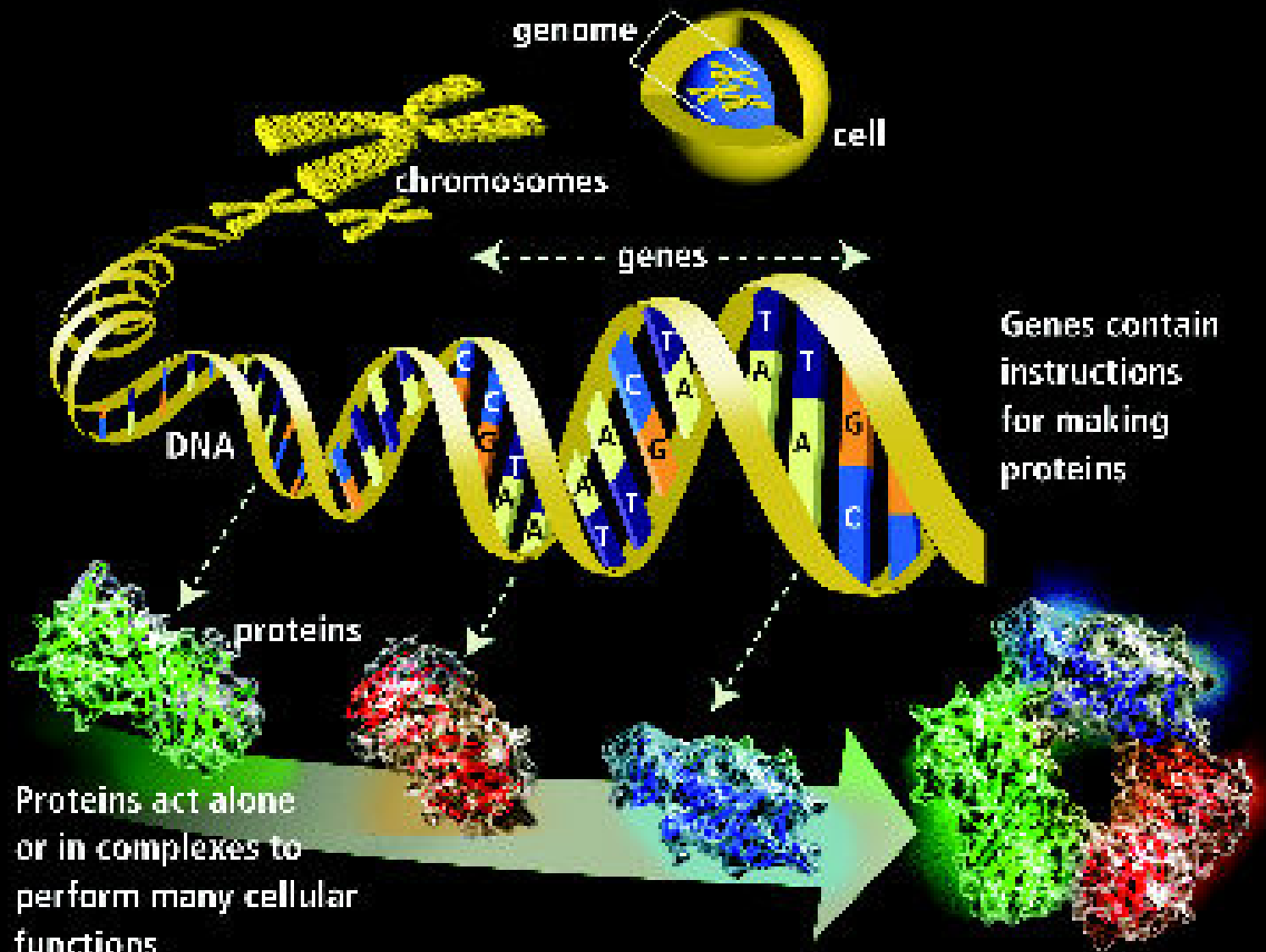
- four nucleotides in DNA contain the bases adenine (A), guanine (G), cytosine (C), and thymine (T).

- **Gene expression**

- process by which a gene's coded information is converted into the structures present and operating in the cell.

- **Bioinformatics**

- science of managing & analyzing genomic research data using advanced computing techniques.



# Bioinformatics Key Applications

- database search
  - Exponential growth of biological sequence data.
- multiple sequence comparison
  - Two constraints: execution time & memory bandwidth.

# Motivation for ATGC

- Need for
  - faster sequence comparison algorithms.
  - tools for comparative genomics.
- Parallelization of sequence alignment algorithms.
  - reliable output and reasonable cost.

# ATGC

## Another Tool for Genome Comparison

- genome comparison
- sequence analysis
- parallel computation
- multithreading

# Smith-Waterman Algorithm

- Instead of looking at each sequence in its entirety this compares segments of all possible lengths (LOCAL alignments) and chooses whichever maximize the similarity measure.
- For every cell the algorithm calculates ALL possible paths leading to it. These paths can be of any length and can contain insertions and deletions.

# Smith-Waterman Algorithm

- Only works effectively when gap penalties are used
- In example shown
  - match = +1
  - mismatch = -1/3
  - gap = -1+1/3k (k=extent of gap)
- Start with all cell values = 0
- Looks in subcolumn and subrow shown and in direct diagonal for a score that is the highest when you take alignment score or gap penalty into account

[illegible]

# Smith-Waterman Algorithm (cont.)

## Construct Alignment

- The score in each cell is the maximum possible score for an alignment of ANY LENGTH ending at those coordinates
- Trace pathway back from highest scoring cell
- This cell can be anywhere in the array
- Align highest scoring segment

|   | C   | A   | G   | C   | C   | U   | C   | G   | C   | U   | U   | A   | G   |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| A | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| A | 0.0 | 1.0 | 0.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.7 |
| U | 0.0 | 0.0 | 0.8 | 0.3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.7 |
| G | 0.0 | 0.0 | 1.0 | 0.3 | 0.0 | 0.0 | 0.7 | 1.0 | 0.0 | 0.0 | 0.7 | 0.7 | 1.0 |
| C | 1.0 | 0.0 | 0.0 | 2.0 | 1.3 | 0.3 | 1.0 | 0.3 | 2.0 | 0.7 | 0.3 | 0.3 | 0.3 |
| C | 1.0 | 0.7 | 0.0 | 1.0 | 3.0 | 1.7 | 1.3 | 1.0 | 1.3 | 1.7 | 0.3 | 0.0 | 0.0 |
| A | 0.0 | 2.0 | 0.7 | 0.3 | 1.7 | 2.7 | 1.3 | 1.0 | 0.7 | 1.0 | 1.3 | 1.3 | 0.0 |
| U | 0.0 | 0.7 | 1.7 | 0.3 | 1.3 | 2.7 | 2.3 | 1.0 | 0.7 | 1.7 | 2.0 | 1.0 | 1.0 |
| U | 0.0 | 0.3 | 0.3 | 1.3 | 1.0 | 2.3 | 2.3 | 2.0 | 0.7 | 1.7 | 2.7 | 1.7 | 1.0 |
| G | 0.0 | 0.0 | 1.3 | 0.0 | 1.0 | 1.0 | 2.0 | 3.3 | 2.0 | 1.7 | 1.3 | 2.3 | 2.7 |
| A | 0.0 | 1.0 | 0.0 | 1.0 | 0.3 | 0.7 | 0.7 | 2.0 | 3.0 | 1.7 | 1.3 | 2.3 | 2.0 |
| C | 1.0 | 0.0 | 0.7 | 1.0 | 2.0 | 0.7 | 1.7 | 1.7 | 3.0 | 2.7 | 1.3 | 1.0 | 2.0 |
| G | 0.0 | 0.7 | 1.0 | 0.3 | 0.7 | 1.7 | 0.3 | 2.7 | 1.7 | 2.7 | 2.3 | 1.0 | 2.0 |
| G | 0.0 | 0.0 | 1.7 | 0.7 | 0.3 | 0.3 | 1.3 | 1.3 | 2.3 | 1.3 | 2.3 | 2.0 | 2.0 |

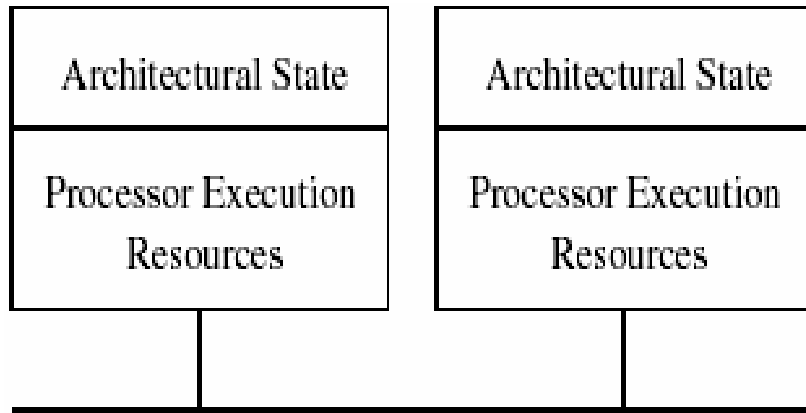
GCC-UCG

GCCAUUG

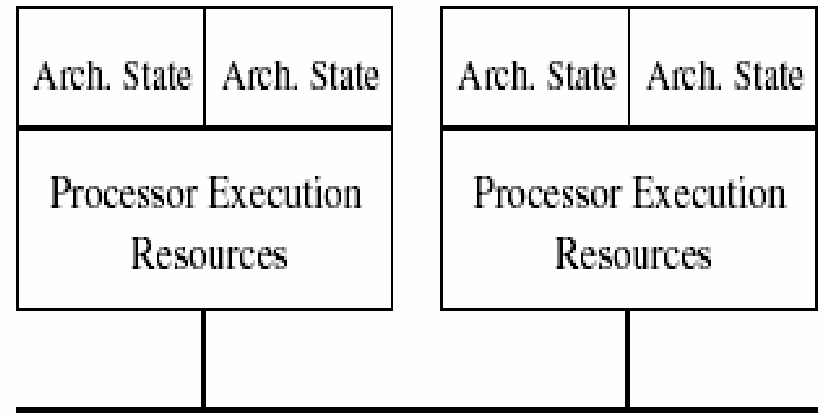
# Parallel Computation of Sequence Alignment

- Parallelization of Smith-Waterman Algorithm decreases execution time almost linearly.
- Parallelization is achieved by
  - OpenMP pragmas.
  - rotating buffers for thread communication.
  - Hyper threading multiprocessors.
- Benefits
  - Locks/critical sections not needed.
  - Minimum interthread communication.

# Hyper-Threading Technology



(a) Hyper-threading non-capable processors



(b) Hyper-threading-enabled processors

Traditional and HT-capable multiprocessor systems

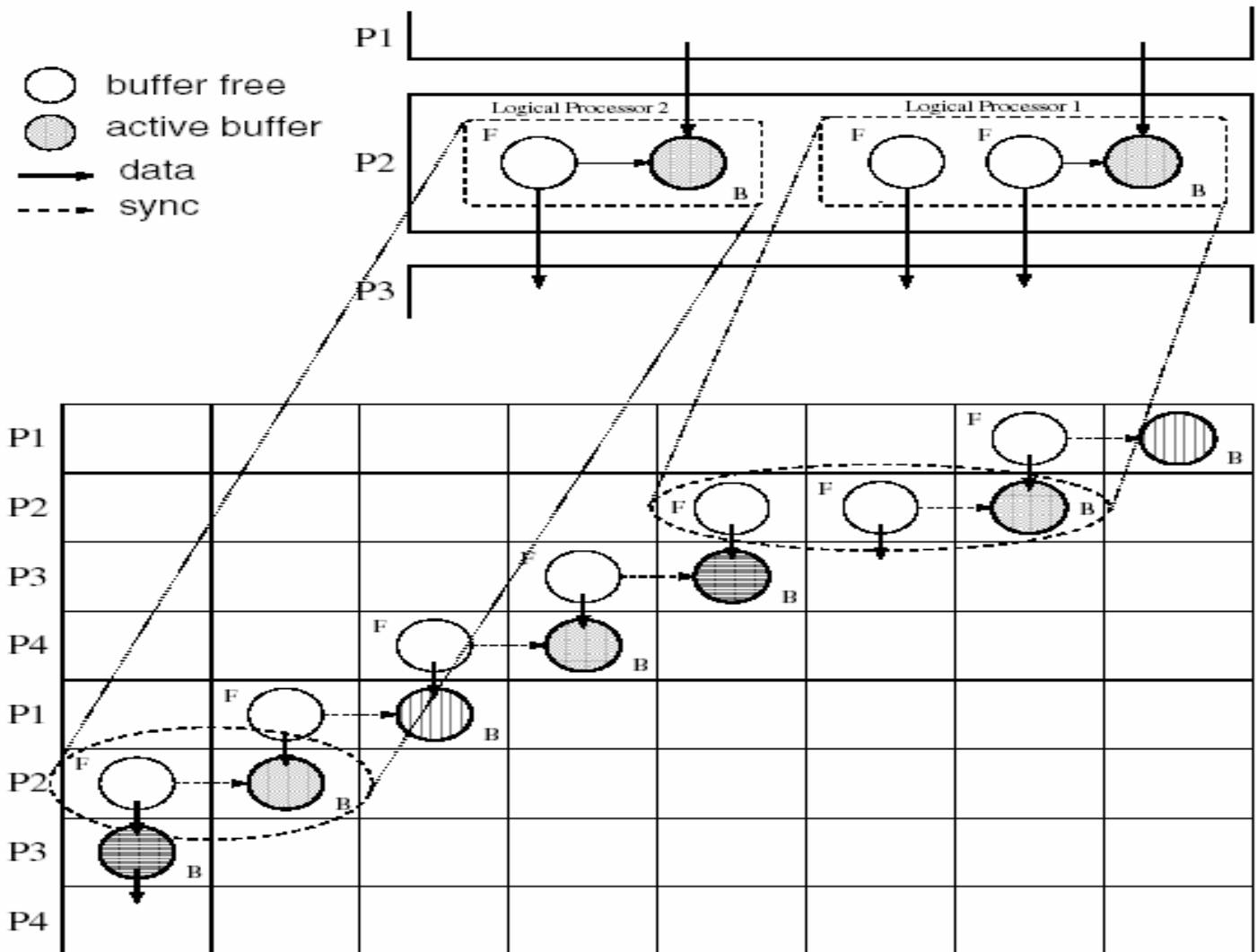
# Computation of the similarity matrix on a SMP system

\* 2 threads per physical processor

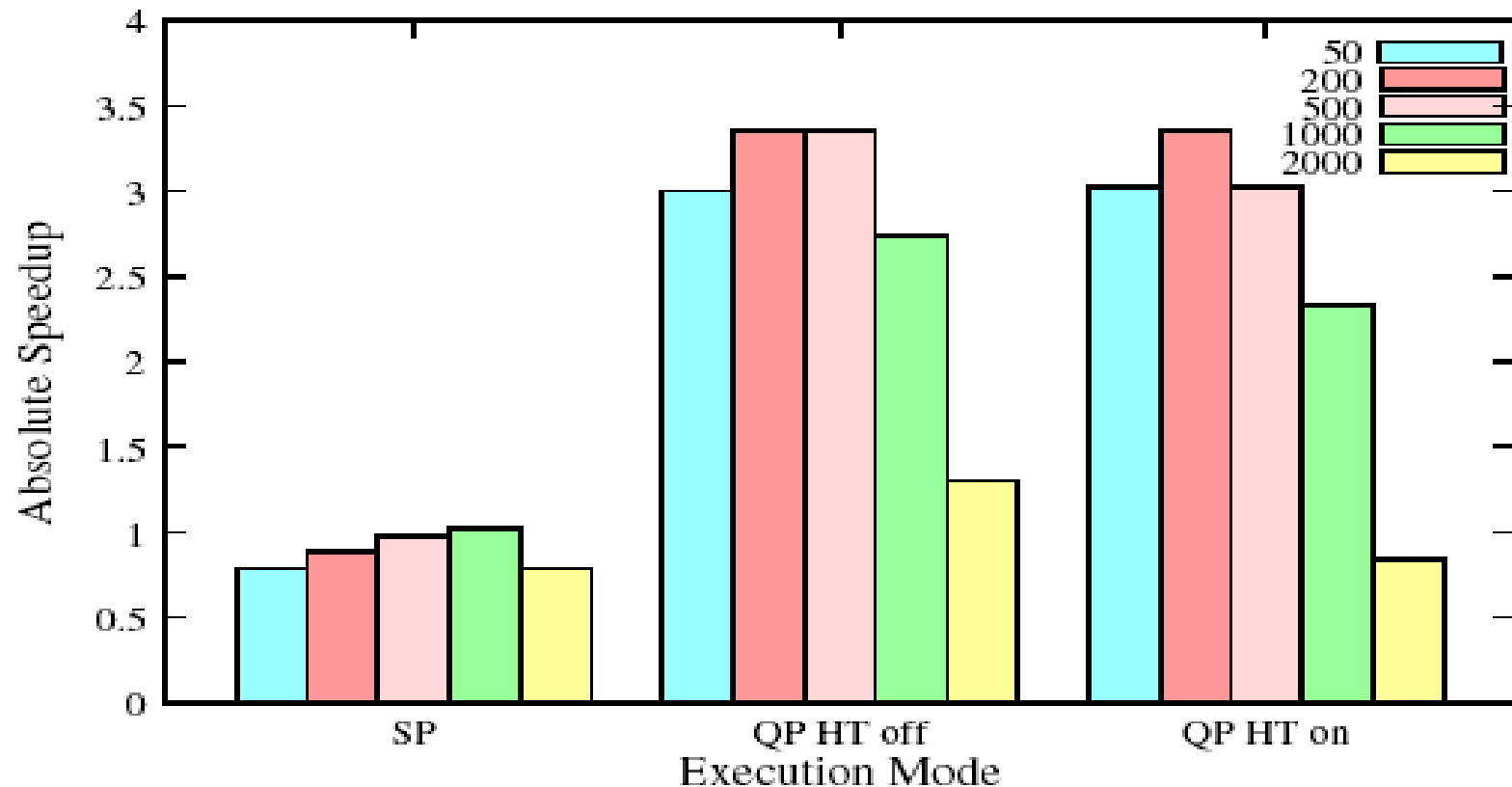
\* Event-driven

\* Asynchronous

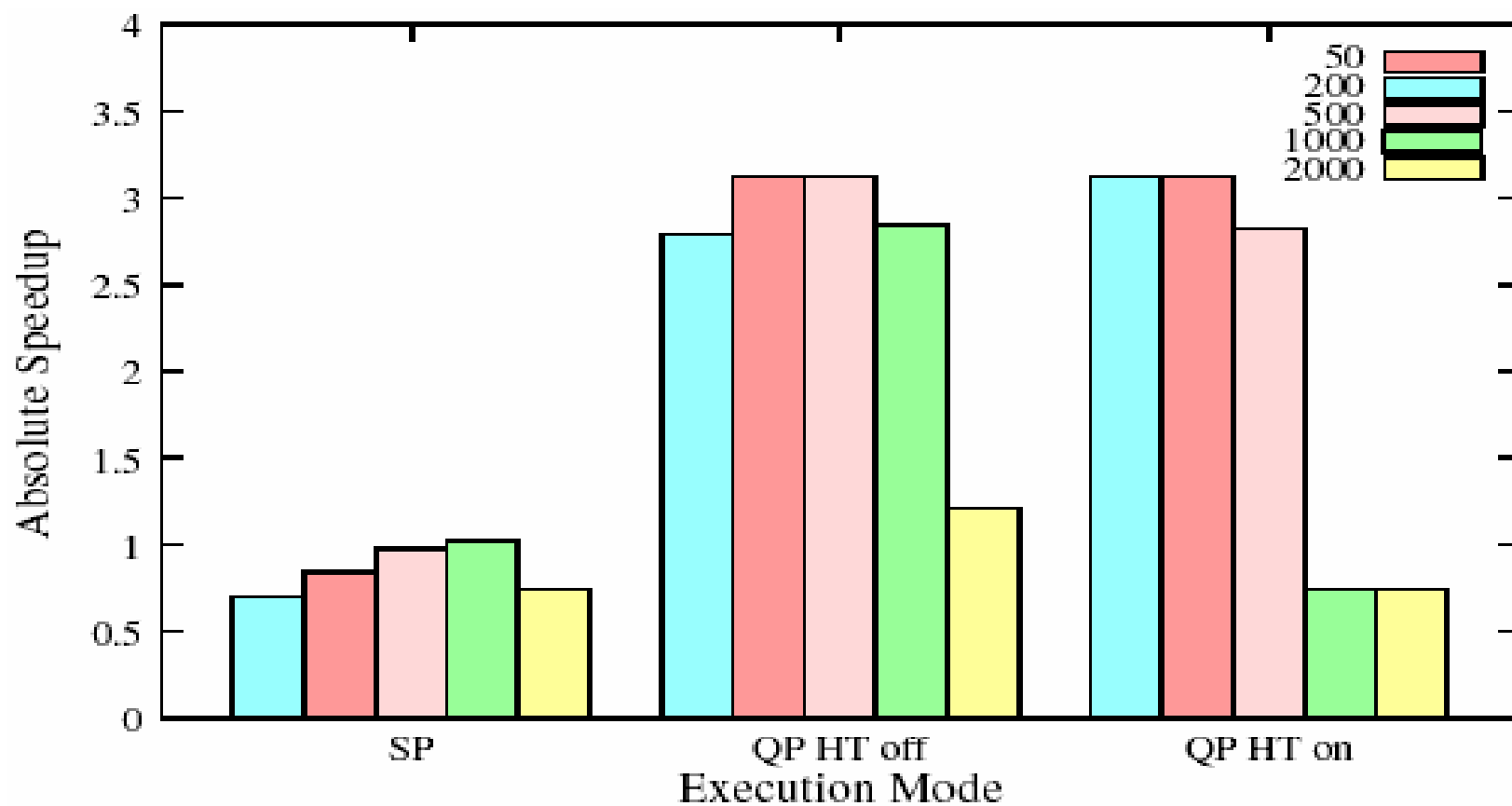
\* Implicit communication/synchronization  
within threads



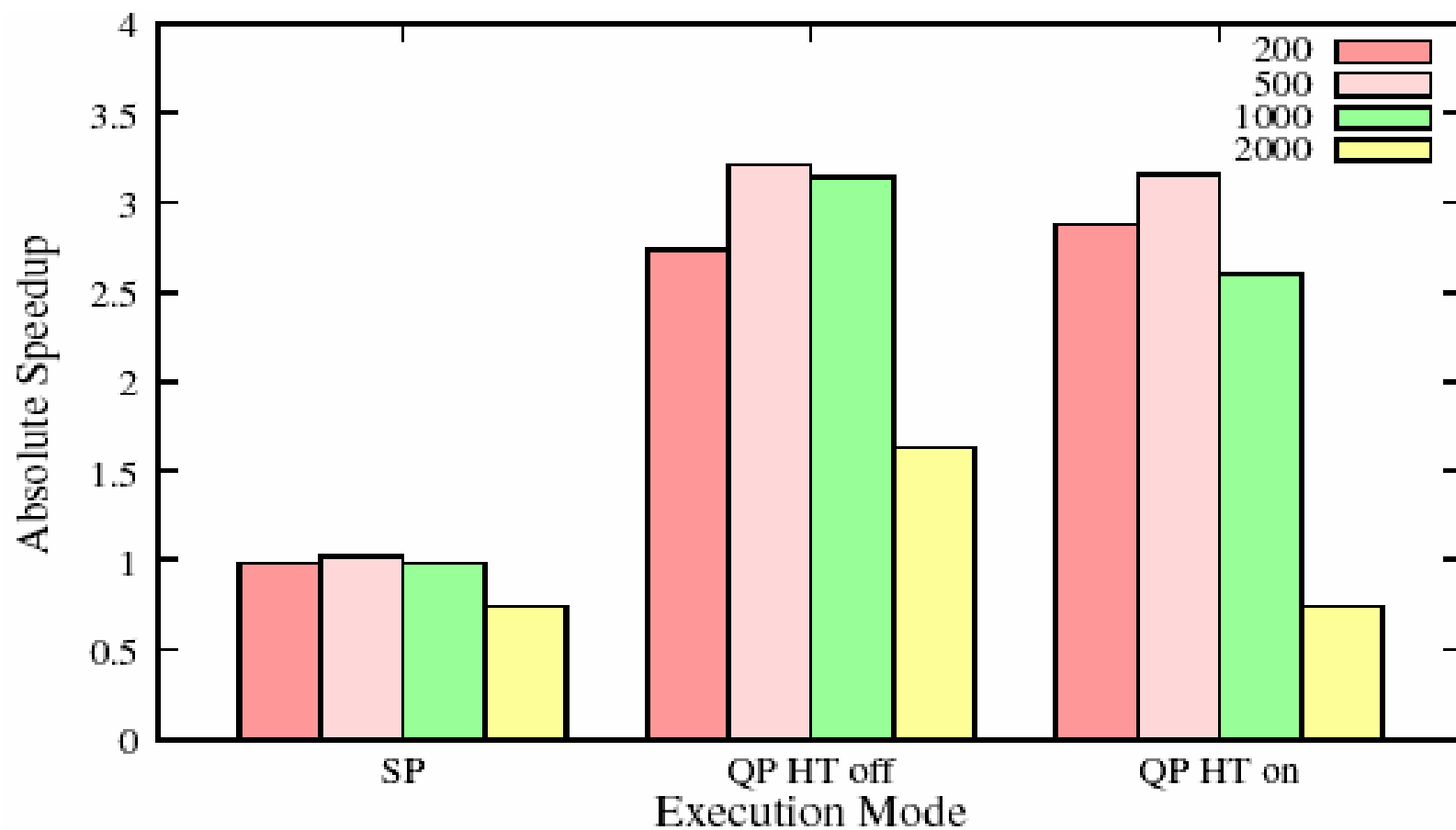
# Absolute speedup for mitochondrial genome comparisons



(a) Human vs. mice



(b) Human vs. *Drosophila*



(c) HHV1 vs. HHV2

**Table 1.** Workload characteristics for the human vs. mouse and human vs. *Drosophila* genome comparisons with block width 1,000

|                                  | SP     | HT off  | HT on  | SP     | HT off | HT on   |
|----------------------------------|--------|---------|--------|--------|--------|---------|
| Execution Time (seconds)         | 30     | 9       | 9      | 28     | 9      | 9       |
| Instructions retired (millions)  | 24,547 | 42,606  | 44,819 | 24,429 | 42,967 | 42,347  |
| $\mu$ ops retired (millions)     | 23,175 | 52,376  | 54,418 | 20,551 | 49,942 | 53,134  |
| Trace cache delivery rate        | 74.40% | 102.45% | 87.96% | 68.08% | 98.11% | 100.00% |
| Load accesses (millions)         | 9,990  | 17,035  | 17,873 | 9,816  | 16,658 | 16,662  |
| L1 cache load misses (millions)  | 712    | 1,365   | 1,236  | 758    | 1,349  | 1,222   |
| L2 cache load misses (millions)  | 32     | 290     | 27     | 30     | 338    | 29      |
| L1 cache miss rate               | 7.13%  | 8.01%   | 7.62%  | 7.72%  | 8.09%  | 7.33%   |
| L2 cache miss rate               | 0.32%  | 1.70%   | 0.15%  | 0.30%  | 2.03%  | 1.74%   |
| L3 cache miss rate               | 0.02%  | 0.02%   | 0.04%  | 0.02%  | 0.02%  | 0.05%   |
| Branches retired (millions)      | 3,111  | 5,255   | 5,528  | 3,318  | 5,823  | 5,473   |
| Mispredicted branches (millions) | 112    | 220     | 217    | 138    | 335    | 231     |
| Branch misprediction rate        | 3.60%  | 4.19%   | 3.90%  | 4.16%  | 5.75%  | 4.22%   |

# Conclusions

- Meets the requirements for small and medium size genomes.
- Intel compiler yields to a 3.3 absolute speedup on a quadprocessor machine.
- high 1<sup>st</sup> level cache miss rate of 7-8%.
- Logical processors with HT technology enabled cannot achieve further improvement.
  - high 1<sup>st</sup> level cache miss rate of 7-8%.
  - lack of memory bandwidth.

# Future Work

## Investigate

- why hyper-threading does not bring additional performance gain.
- overhead caused by
  - loop with the PAUSE instruction, which accounts for 5% of the execution time.
  - four load-store operations that account for more than 90% of the L2 cache misses.

# References

- A. N. Arslan et al. **A new approach to sequence comparison: Normalized sequence alignment.** Bioinformatics, 17(4):327–337, 2001.
- J. del Cuvillo. **Whole genome comparison using a multithreaded parallel implementation.** Master's thesis, U. of Delaware, Newark, Del., Jul. 2001.
- A. L. Delcher et al. **Alignment of whole genomes.** Nucleic Acids Res., 27(11):2369–2376, 1999.
- D. T. Mar et al. **Hyper-threading technology architecture and microarchitecture.** Intel Tech. J., 6(1):4–15, Feb. 2002.
- W. S. Martins et al. **Whole genome alignment using a multithreaded parallel implementation.** In Proc. of the 13th Symp. on Computer Architecture and High Performance Computing, Pirenópolis, Brazil, Sep.10–12, 2001.
- W. S. Martins et al. **A multithreaded parallel implementation of a dynamic programming algorithm for sequence comparison.** In Proc. of the Pacific Symp. On Biocomputing, pages 311–322, Mauna Lani, Haw., Jan. 3–7, 2001.
- D. M. Tullsen et al. **Simultaneous multithreading: Maximizing on-chip parallelism.** In Proc. of the 22nd Ann. Intl. Symp. on Computer Architecture, pages 392–403, Santa Margherita Ligure, Italy, Jun. 1995.
- M. S. Waterman. **Introduction to Computational Biology: Maps, Sequences, and Genomes.** Chapman and Hall, 1995.