A Parallel Implementation of Quicksort and its Performance Evaluation

Philippas Tsigas Yi Zhang
Department of Computing Science
Chalmers University of Technology

The aim of our work

- Sorting is an important kernel
- Parallel implementations of sorting
  - Based on message-passing machines,
  - Sample sort
- New developments in computer architecture bring us new research opportunities
  - Cache-Coherent shared memory
  - Tightly-coupled multiprocessor

Quicksort

- Advantages
  - General purpose
  - In-place
  - Good cache-behavior
  - Simple
- Disadvantages
  - Parallel implementations do not scale up.

Our Approach

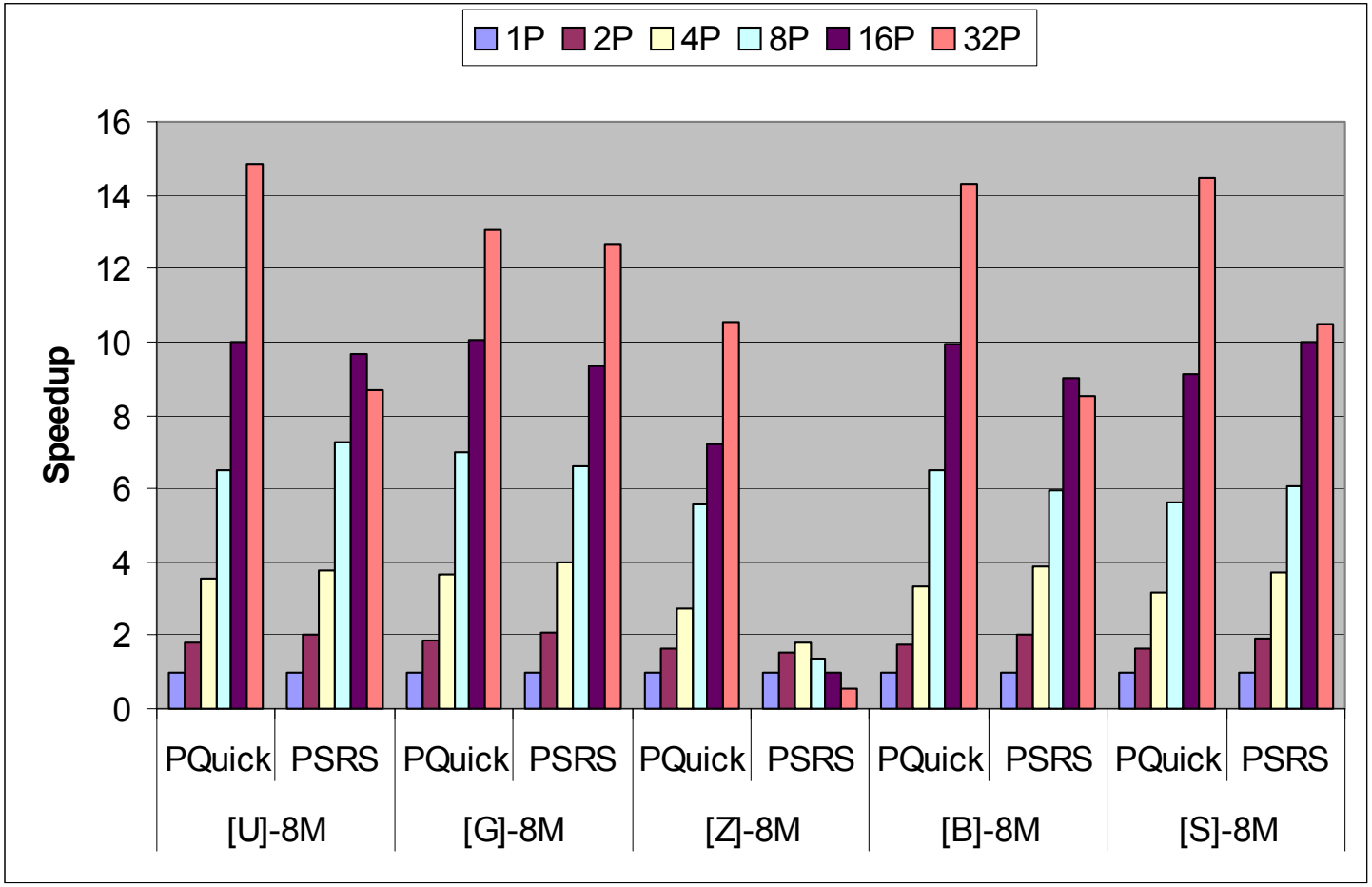
3+1 Phases

- Parallel Partition of the Data
  - Block based partition
  - Cache efficient
- Sequential Partition of the Data
  - At most P+1 blocks (P: Number of processors)
- Process Partition
- Sequential Sorting with Helping
  - Load-balancing
  - Non-blocking synchronization

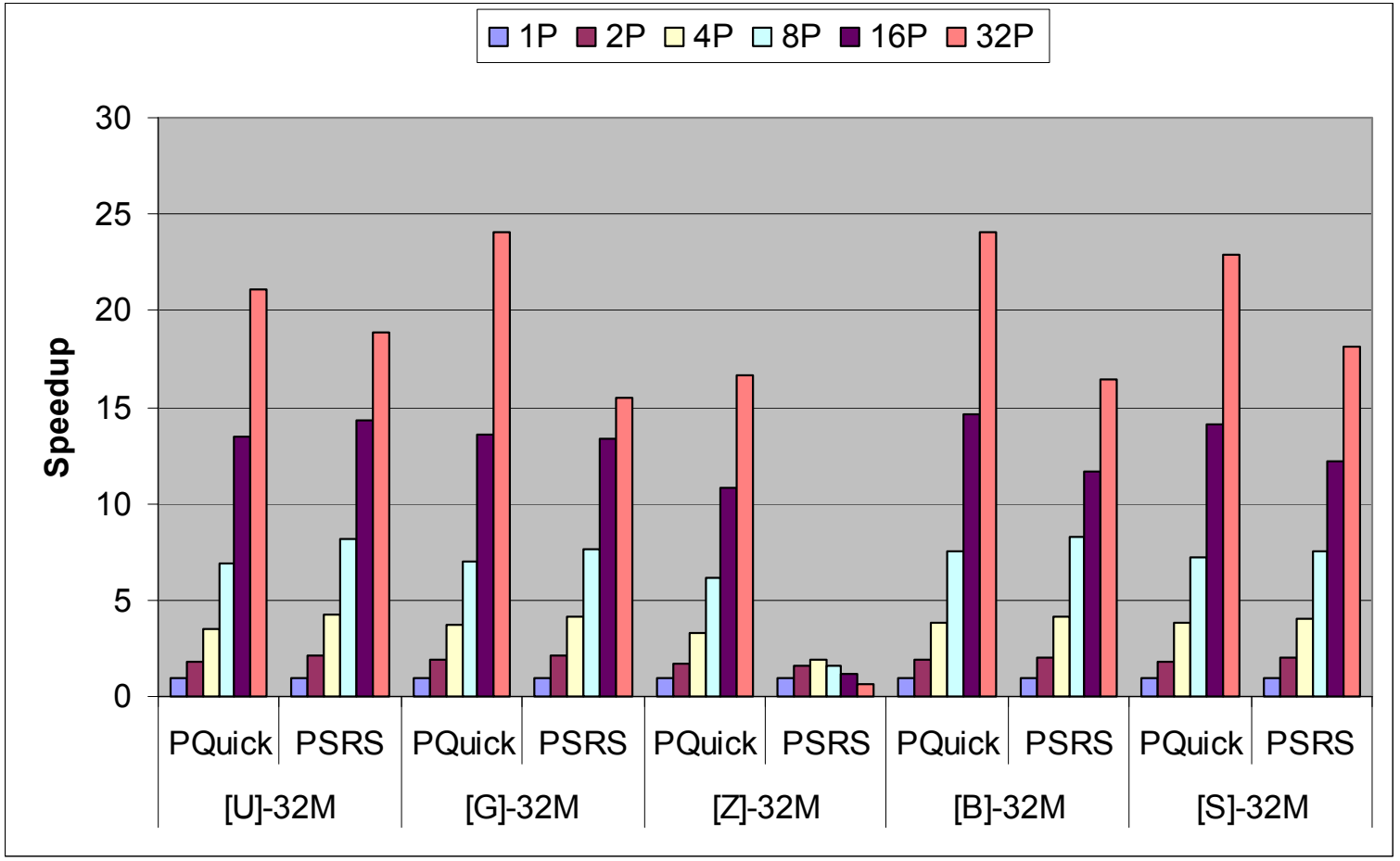
The advantages of our approach

- General purpose
- In-place
- Good cache-behavior
- Fine grain parallelism
- Good speedup in theory

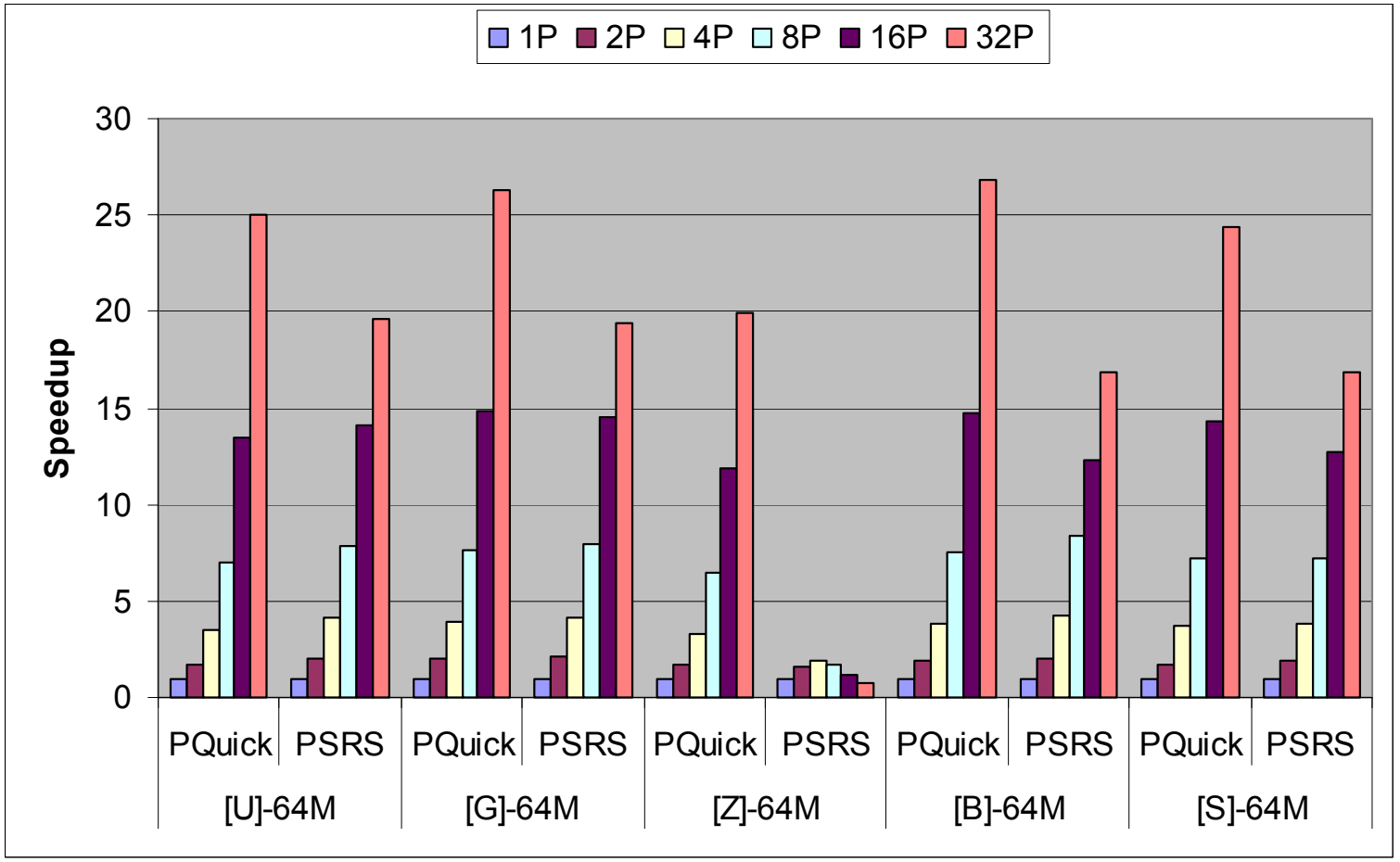
Experimental Results (8M Integers)



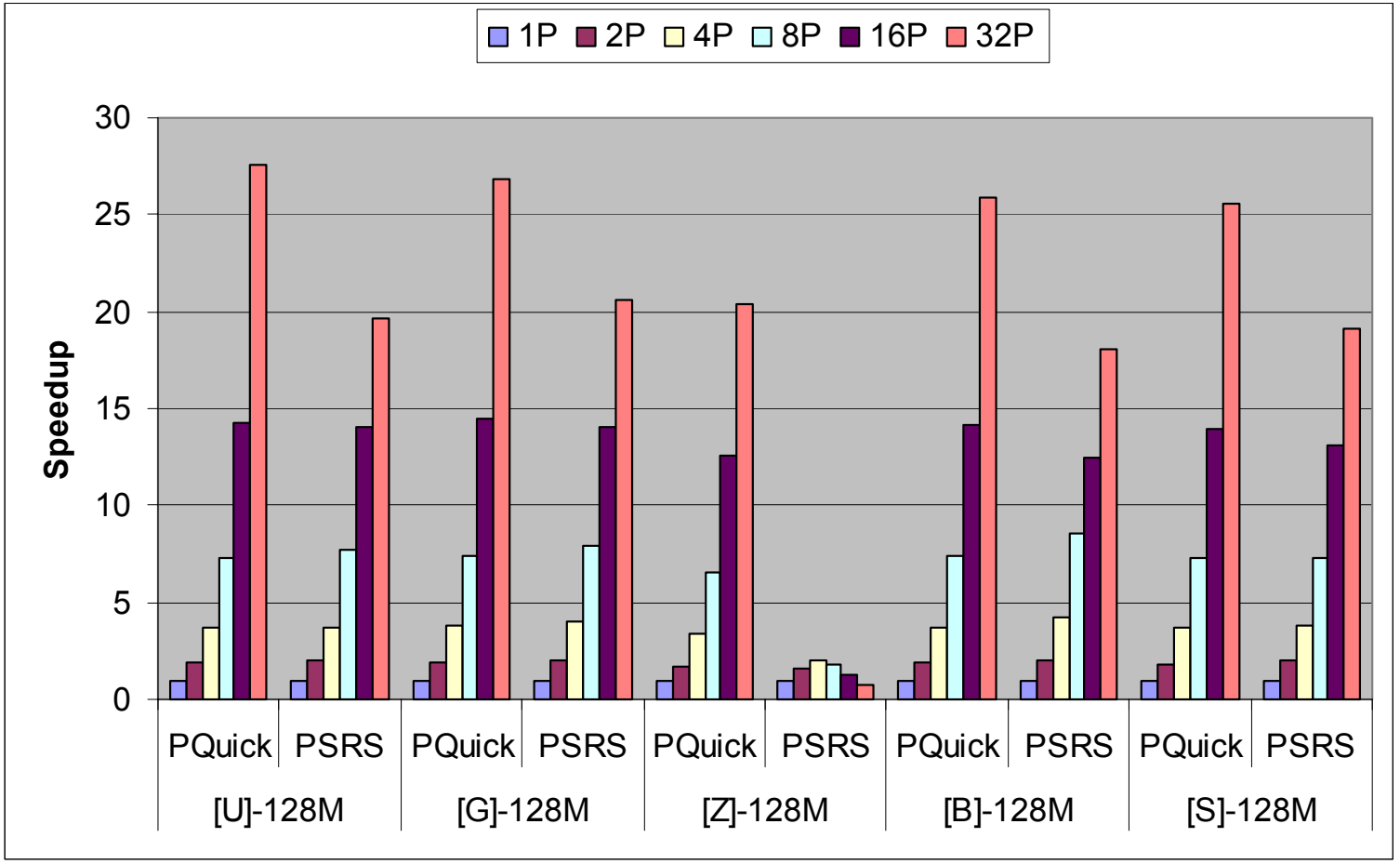
Experimental Results (32M Integers)



Experimental Results (64M Integers)



Experimental Results (128M Integers)



Conclusions

- Quicksort can beat Sample Sort on cache-coherent shared memory multiprocessors.
- Fine grain parallelism that incorporates non-blocking synchronization can be efficient.
- Cache-coherent shared memory multiprocessors offer many new research opportunities.