LEHIGH UNIVERSITY

Multithreaded Architectures and The Sort Benchmark

Phil Garcia

Hank Korth

Dept. of Computer Science and Engineering

Lehigh University

About our Sort Benchmark

- Based on the benchmark proposed in *A measure of transaction processing power* (Anonymous et al).
- Sorts 100 byte records containing 10 byte keys.
- Modified to run in main-memory.
- Modified to sort 250MB of records (instead of 100MB).

Results

- 2-way SMT can result in speedups of over 60%.
- SMT can tolerate cache misses.
- Gains increase as the processor/memory gap widens.
- The order of threads’ actions significantly affects speed.
- Merge sort can be more efficient than selection trees.

Test Platform

- Xeon dual 3.0GHz.
  - 2-way SMT
  - 512KB L2 cache
  - 1MB L3 cache.
  - 2GB of RAM
  - 533MHz Bus
- Pentium 4 2.8GHz
  - 2-way SMT
  - 2GB of RAM
  - 1MB L2 cache
  - 800 MHz Bus

Debian GNU/Linux

Kernel 2.6.6

gcc v3.3

Optimized for test architecture.

Algorithm Design

Based on Alphasort (Nyberg et al.)

For Each Set

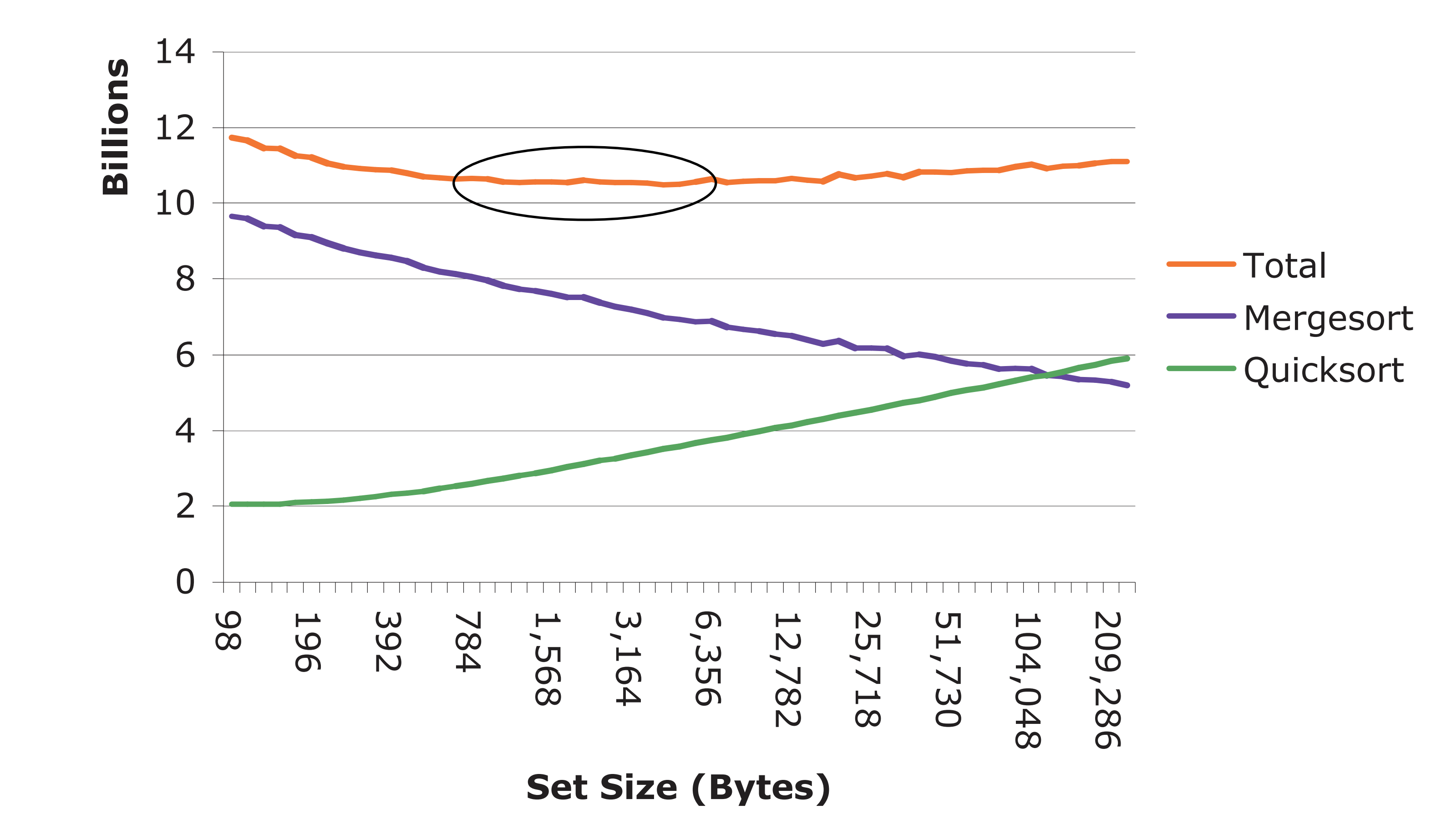
    Extract (key, pointer) pairs

    Quicksort on keys

Mergesort 2 sets at a time until done

    Final merge materializes output.

Single Threaded Breakdown



Xeon single processor

Mergesort vs. Selection Tree

- Selection tree requires large memory footprint.
  - Results in many cache misses per traversal.
- Mergesort has a smaller overall runtime (for larger sorts)
- Mergesort is limited by memory bandwidth because hardware prefetching hides memory latency.

The Final Merge

Set 1

aaa	6
cat	2
dog	5
egg	7

Set 2

bat	3
car	1
dim	0
fog	8

Unsorted Input

	key	data
0	dim	data0
1	car	data1
2	cat	data2
3	bat	data3
4	for	data4
5	dog	data5
6	aaa	data6
7	egg	data7
8	fog	data8
9	hog	data9

The Final Merge

Set 1

aaa	6
cat	2
dog	5
egg	7

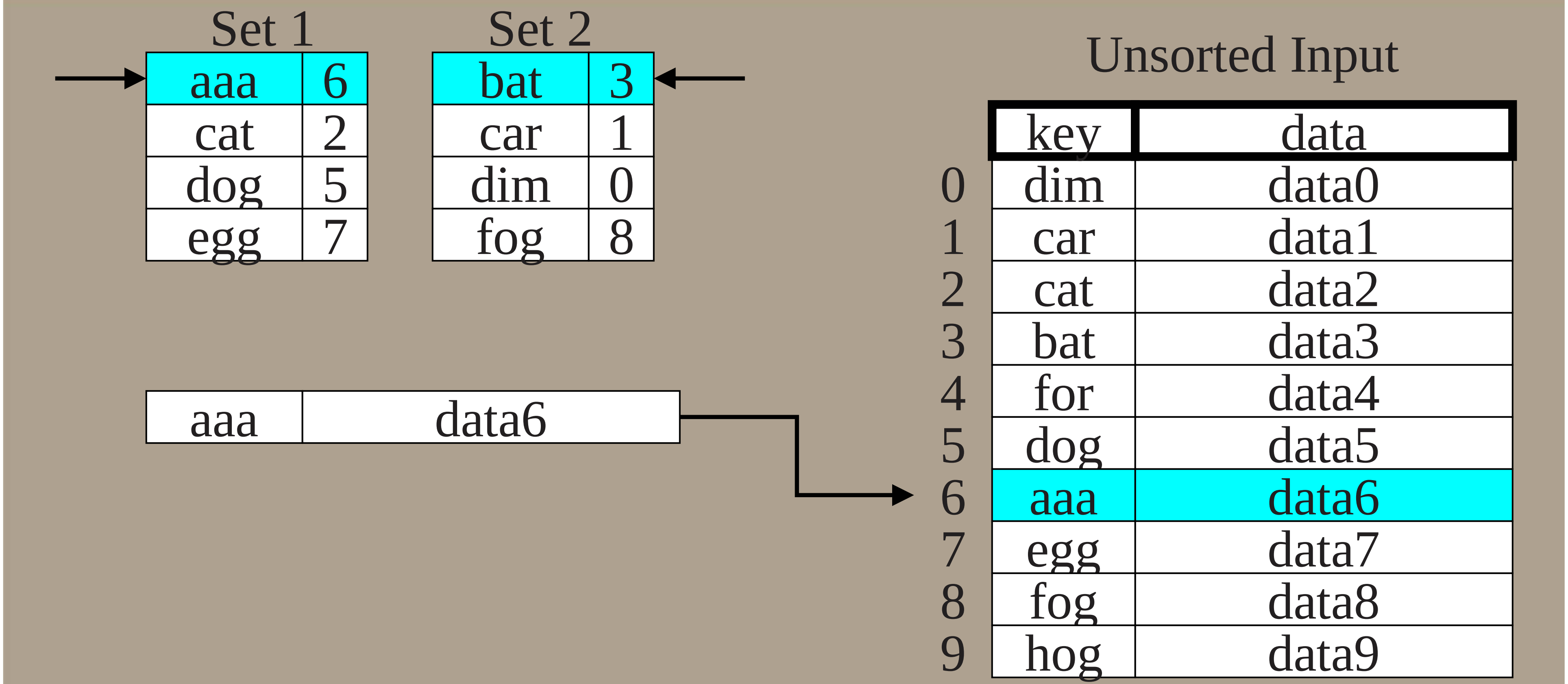
Set 2

bat	3
car	1
dim	0
fog	8

Unsorted Input

	key	data
0	dim	data0
1	car	data1
2	cat	data2
3	bat	data3
4	for	data4
5	dog	data5
6	aaa	data6
7	egg	data7
8	fog	data8
9	hog	data9

The Final Merge



Set 1

aaa	6
cat	2
dog	5
egg	7

Set 2

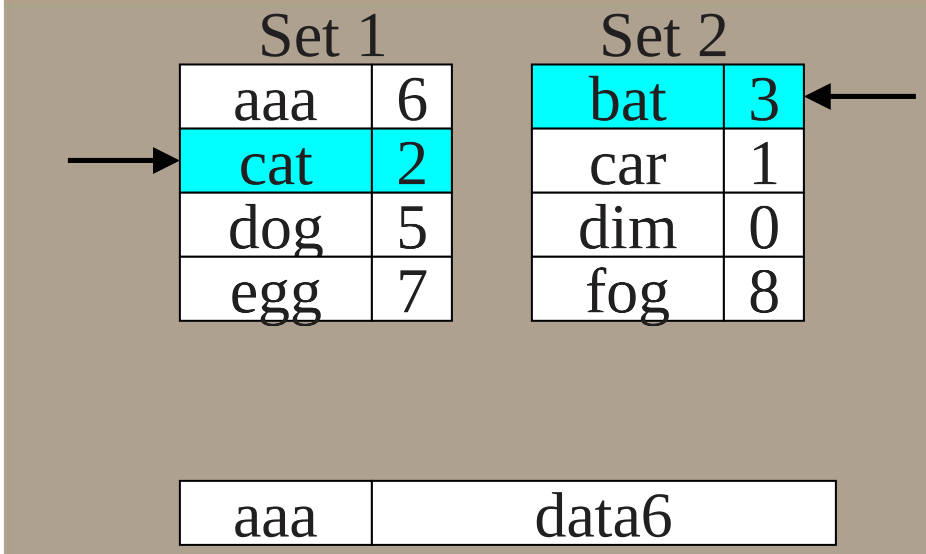
bat	3
car	1
dim	0
fog	8

aaa	data6

Unsorted Input

	key	data
0	dim	data0
1	car	data1
2	cat	data2
3	bat	data3
4	for	data4
5	dog	data5
6	aaa	data6
7	egg	data7
8	fog	data8
9	hog	data9

The Final Merge



Set 1

aaa	6
cat	2
dog	5
egg	7

Set 2

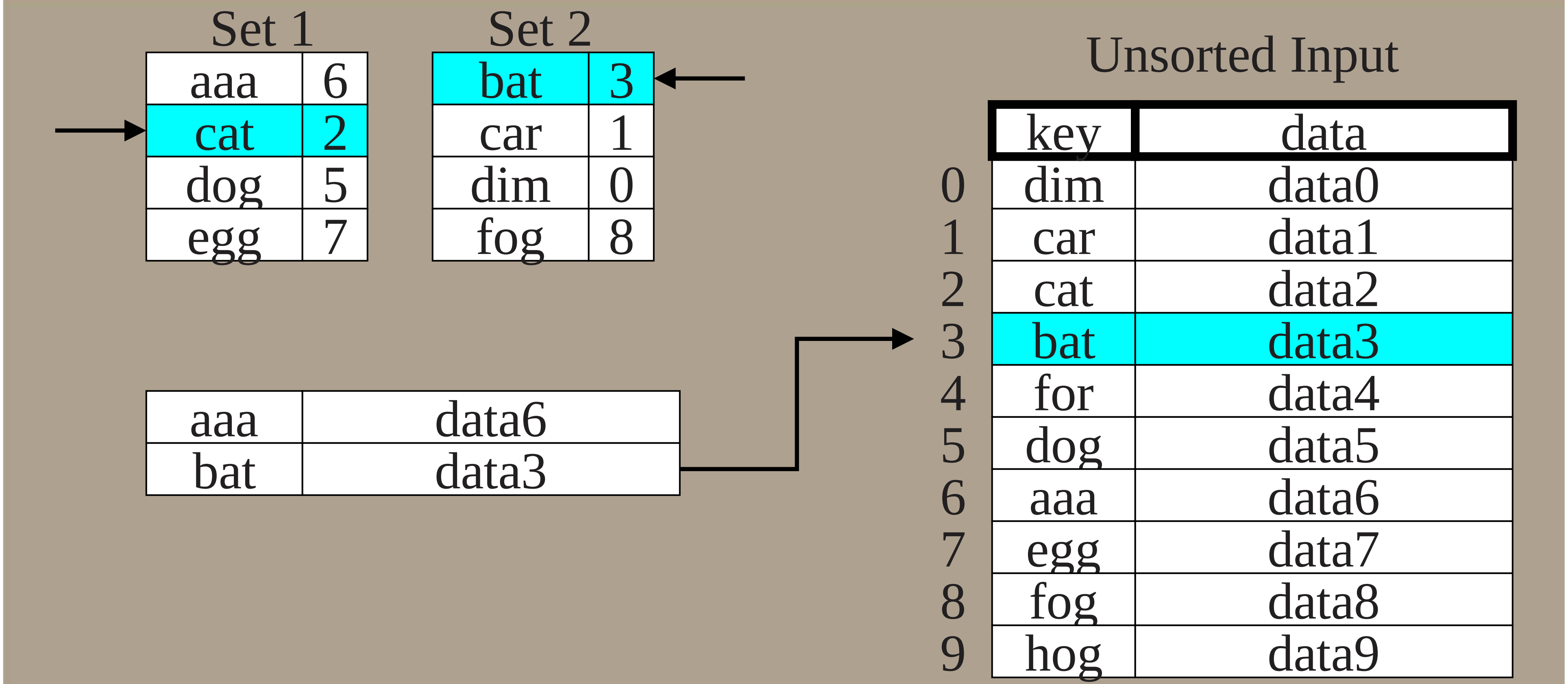
bat	3
car	1
dim	0
fog	8

aaa	data6

Unsorted Input

	key	data
0	dim	data0
1	car	data1
2	cat	data2
3	bat	data3
4	for	data4
5	dog	data5
6	aaa	data6
7	egg	data7
8	fog	data8
9	hog	data9

The Final Merge



Set 1

aaa	6
cat	2
dog	5
egg	7

Set 2

bat	3
car	1
dim	0
fog	8

aaa	data6
bat	data3

Unsorted Input

	key	data
0	dim	data0
1	car	data1
2	cat	data2
3	bat	data3
4	for	data4
5	dog	data5
6	aaa	data6
7	egg	data7
8	fog	data8
9	hog	data9

The Final Merge

Set 1

aaa	6
cat	2
dog	5
egg	7

Set 2

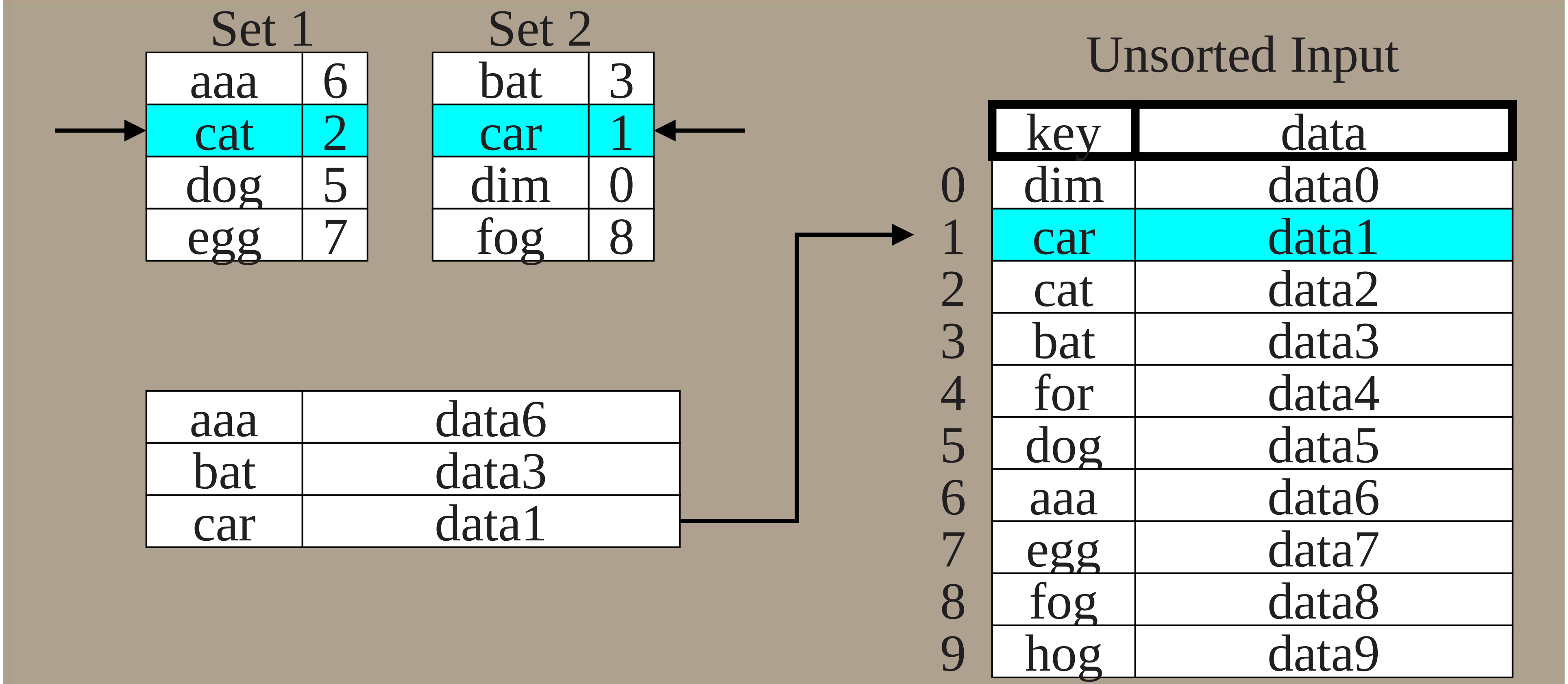
bat	3
car	1
dim	0
fog	8

aaa	data6
bat	data3

Unsorted Input

	key	data
0	dim	data0
1	car	data1
2	cat	data2
3	bat	data3
4	for	data4
5	dog	data5
6	aaa	data6
7	egg	data7
8	fog	data8
9	hog	data9

The Final Merge



Set 1

aaa	6
cat	2
dog	5
egg	7

Set 2

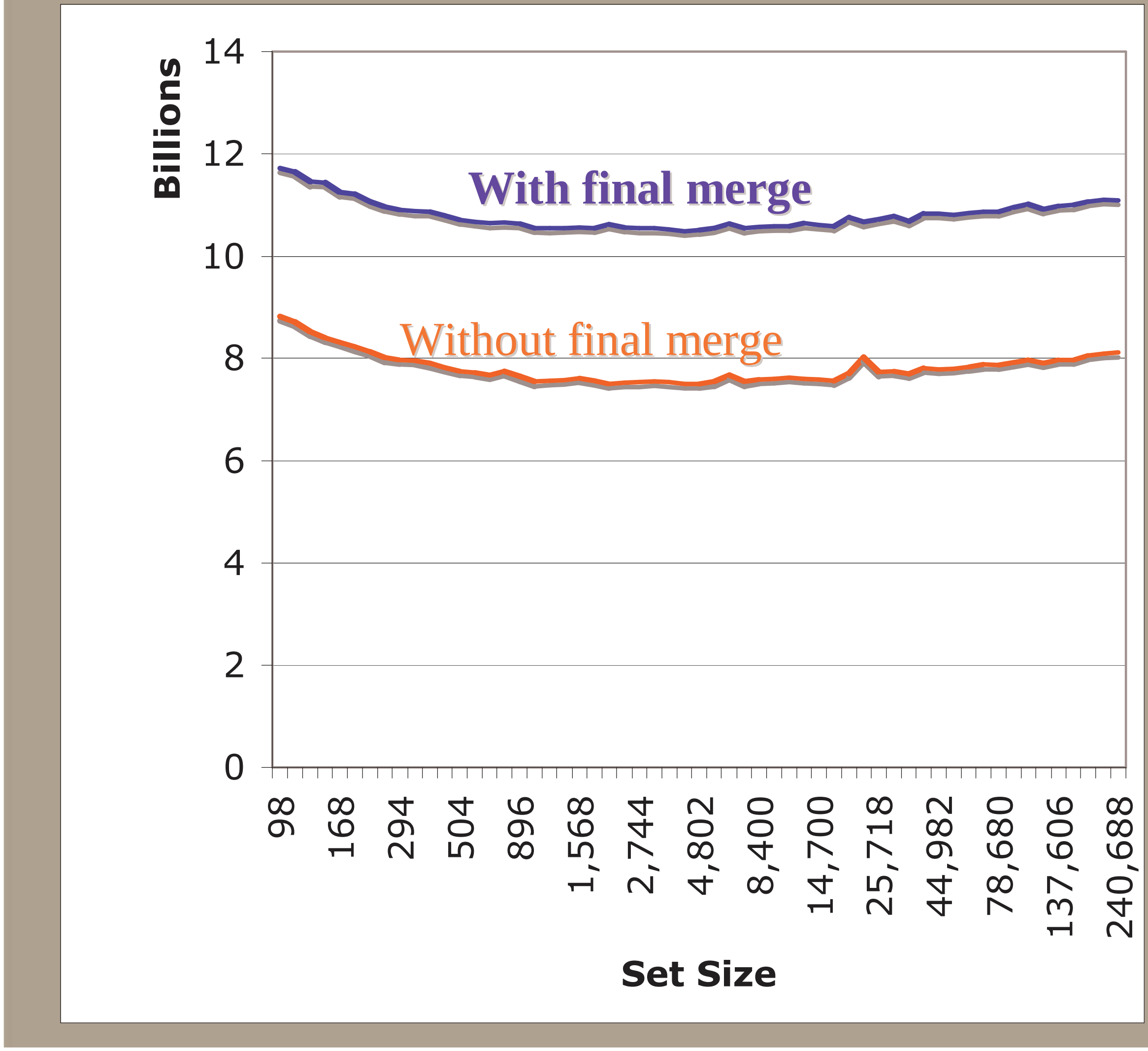
bat	3
car	1
dim	0
fog	8

aaa	data6
bat	data3
car	data1

Unsorted Input

	key	data
0	dim	data0
1	car	data1
2	cat	data2
3	bat	data3
4	for	data4
5	dog	data5
6	aaa	data6
7	egg	data7
8	fog	data8
9	hog	data9

Final Merge Comparison



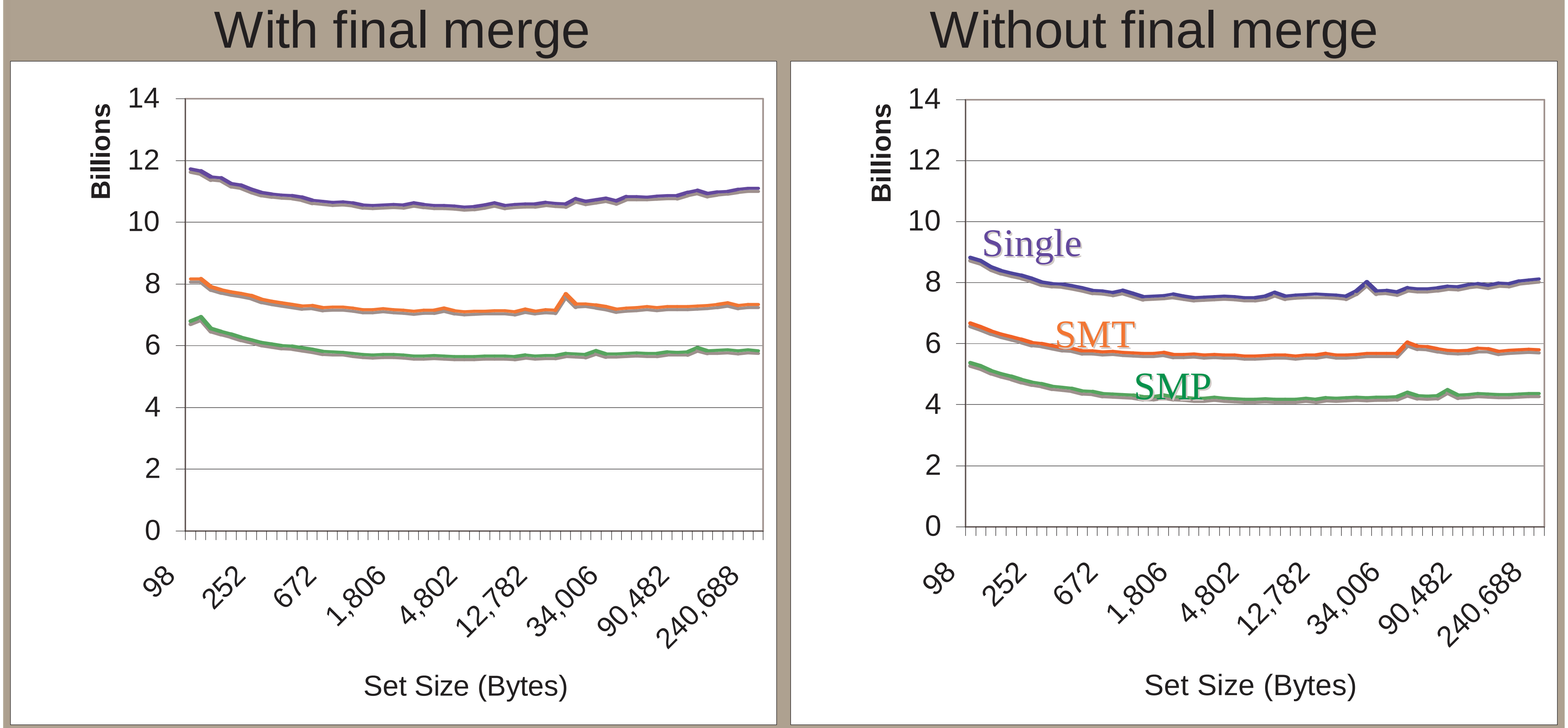
- Takes a significant portion of runtime.
  - Cache thrashing
- Propose not dereferencing pointers.
- Could be useful if the sort was just one operation within a query pipeline.

Multithreading

- Partitioned data among threads based on an estimated median value (Lyer et al.)
- Multiple threads sort simultaneously.
- Ran for both SMT and SMP for two threads.

LEHIGH UNIVERSITY

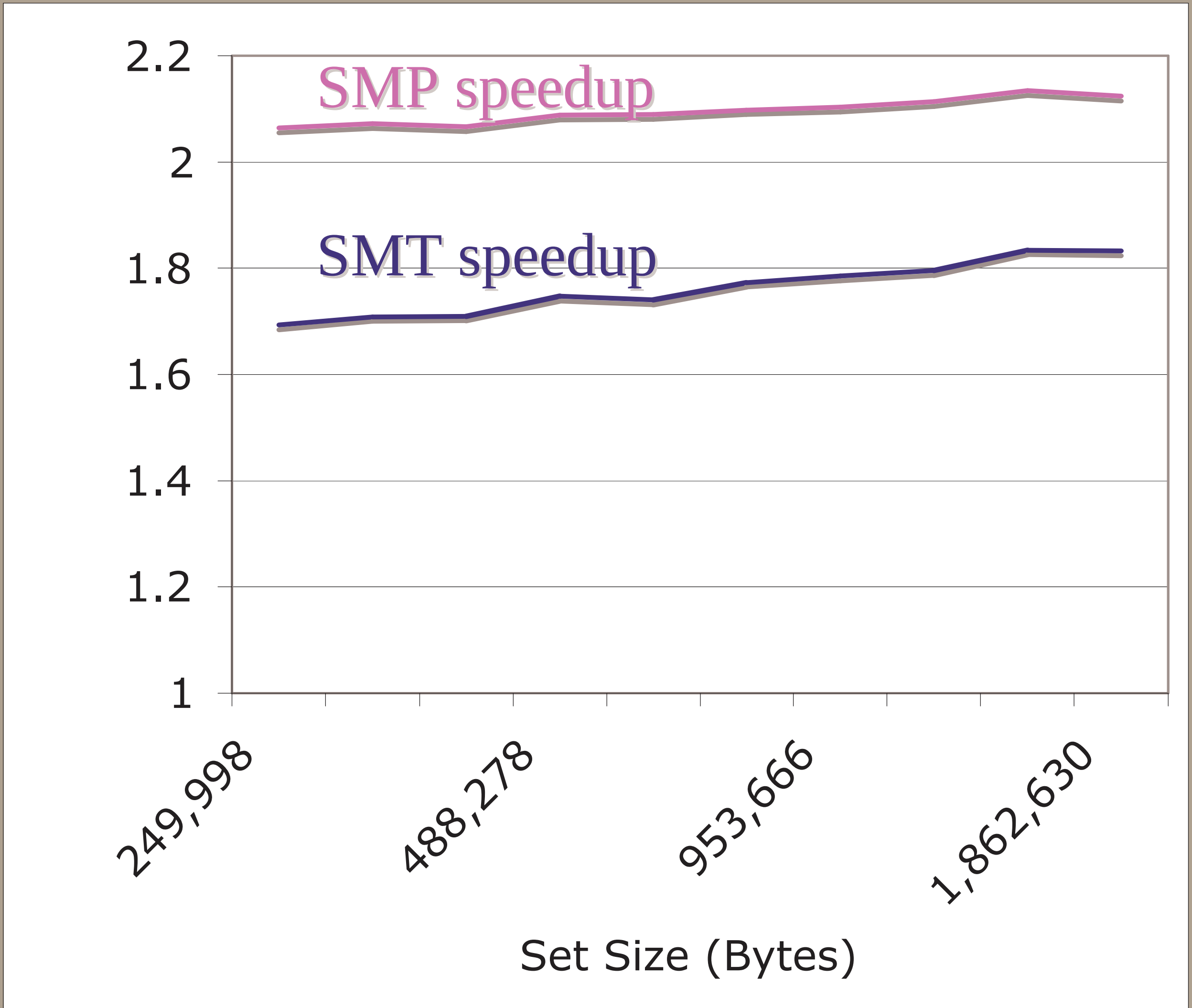
Multithreading (continued)


With final merge

Without final merge

Total runtimes on Xeon Processor

Final Merge (detailed)

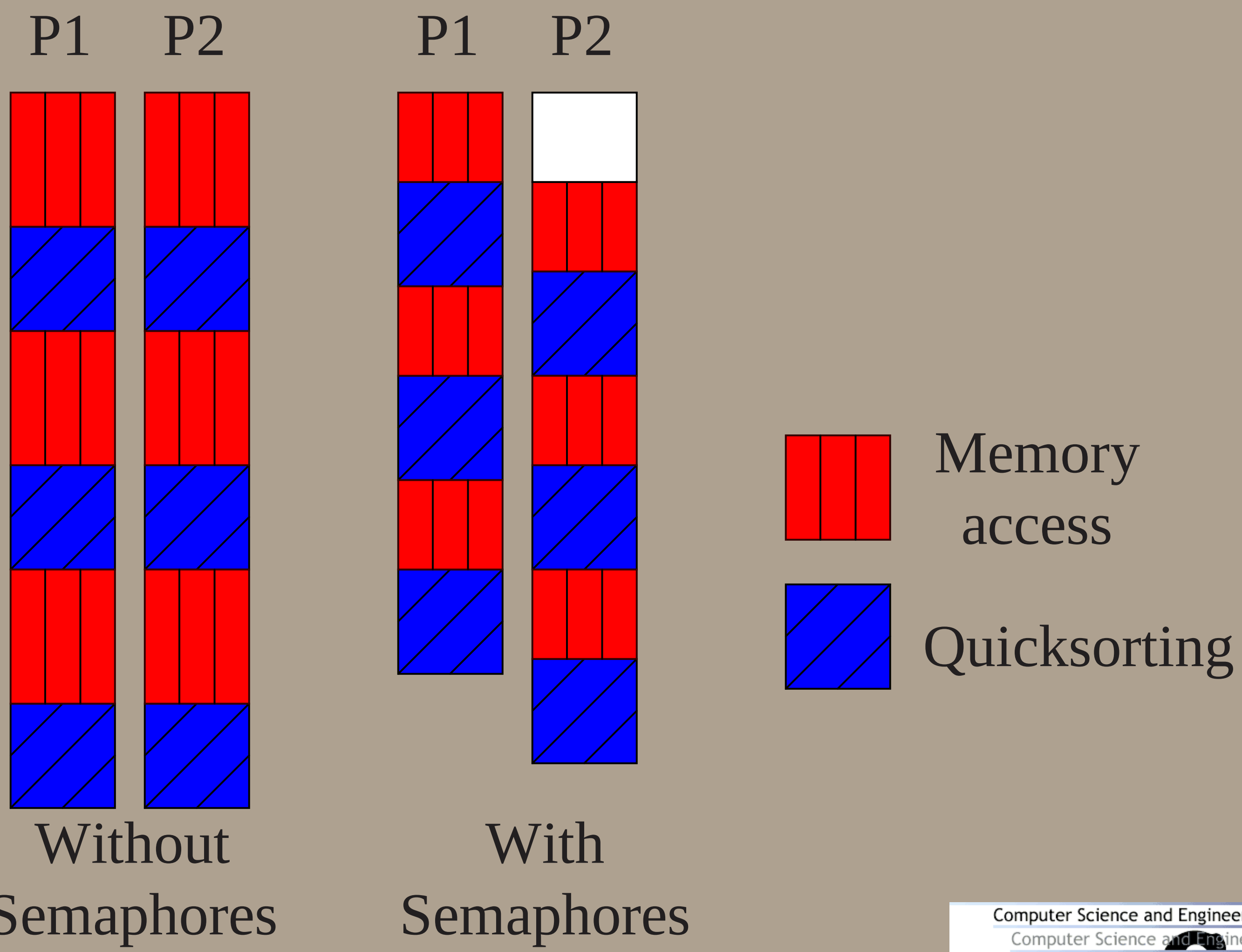


- For the final merge itself we see extremely large speedup.
- SMT speedup similar to that achieved by SMP.

Memory/Processor Gap

- As the memory/processor gap widens so does the speedups obtainable through SMT.
- Ran on both Xeon and P4
  - Xeon showed overall speedup of 47%
  - P4 showed overall speedup of 33%
- Mostly due to Pentium 4’s faster memory and slower clock
  - Enabled a single thread to better utilize processor resources.

Semaphores For Speed, Not Correctness



Semaphores (continued)

- Memory bandwidth does not scale with the number of processors using it.
- Therefore whenever possible:
  - Coordinate threads to share resources.
  - Simple synchronization methods (such as semaphores) work well.
- Large performance gains possible on multiprocessor.

Further Improving Sort

- Sort key-prefixes rather than the full key.
- Enable more threads to speedup the sort
  - 2 processors each running 2 threads.
- Optimize *memcpy*.
- Using multithreaded sort within a query pipeline.

Future Work

- Impact of future processors:
  - Chip Multiprocessors (CMP)
  - Massively Parallel (Sun Niagara/Rock)
- Database pipelines:
  - How best to utilize processor resources.
- Impact on vertically partitioned databases (Manegold, Boncz et al.)

Contact Information

Philip Garcia

philipgar@lehigh.edu

Henry F. Korth

hfk@lehigh.edu

Dept. of Computer Science and Engineering
Packard Lab
19 Memorial Dr. West
Bethlehem, PA 18015