# Heterogeneous Parallel Programming COMP4901D

MapReduce on the GPU

# Overview

- The original MapReduce
- Follow-up MapReduce work prior to Mars
- Mars: MapReduce on the GPU

# Overview of MapReduce

- Map and reduce are data-parallel programming primitives.
- The MapReduce programming framework was proposed by Google for processing large datasets on thousands of computers.
  - Input: a set of key-value pairs
  - Output: another set of key-value pairs
  - Advantages:
    - simple interface; portability; reliability.

# MapReduce in More Detail

- Programmers specify two functions
  - map (in_key, in_value)
  - reduce (out_key, list(intermediate_value))

- The MapReduce runtime takes care of
  1. data distribution
  2. parallelization
  3. load balancing
  4. fault tolerance

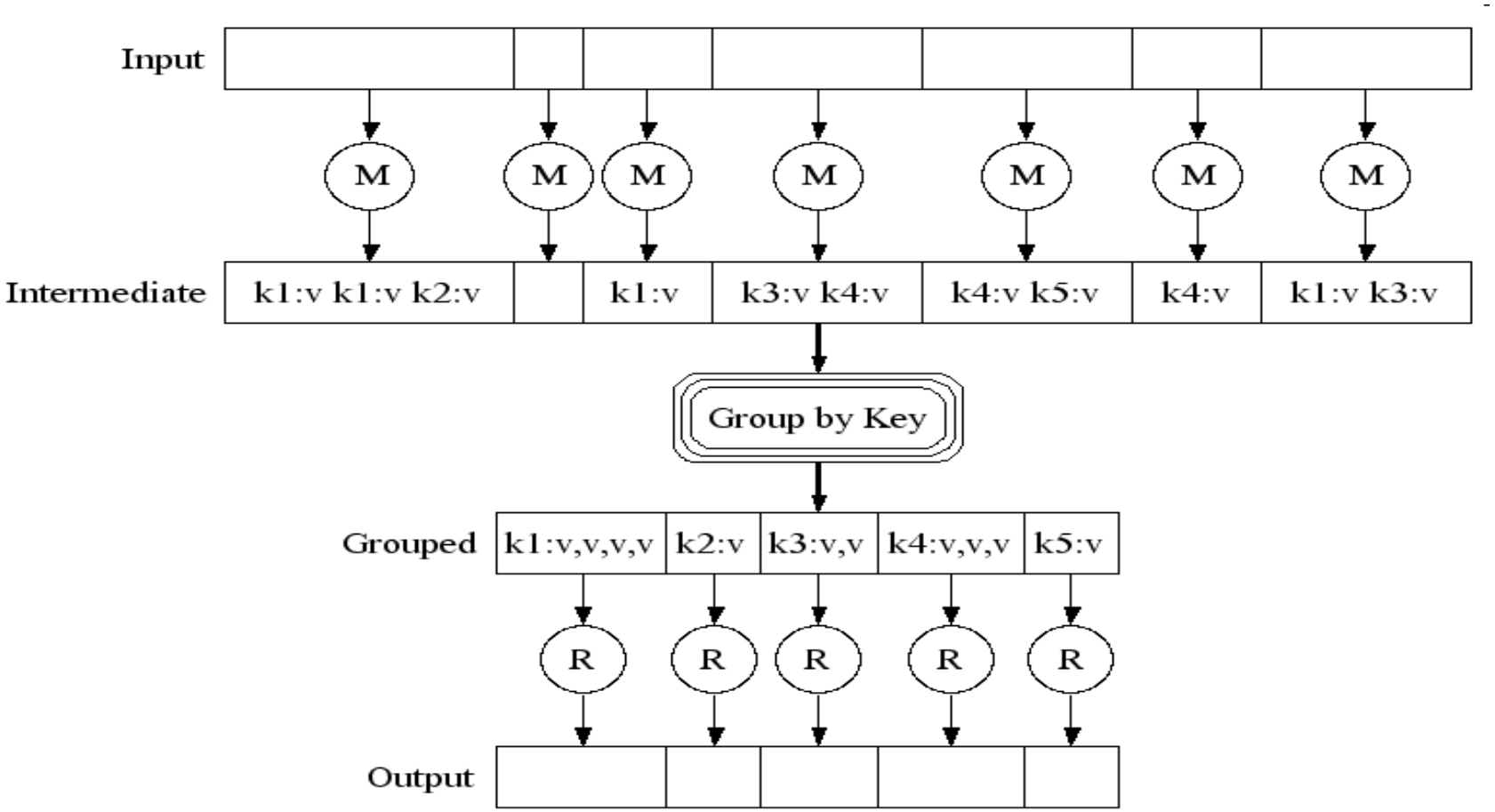
# MapReduce Functions

Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters. OSDI'04. [2]

```
Map(void *doc) {
 for each word w in doc
 EmitIntermediate(w, 1); // count each word once
}

Reduce(void *word, Iterator values) {
 int result = 0;
 for each v in values
   result += v;
   Emit(word, result); // output word and its count
}
```

# MapReduce Workflow



# Follow-up Work on MapReduce

- Hadoop [Apache project]
- MapReduce on multicore CPUs -- Phoenix [HPCA'07, Ranger et al.]
- MapReduce on Cell [07, Kruijf et al.]

- Merge [ASPLOS '08, Linderman et al.]
- MapReduce on GPU [stmcs'08, Catanzaro et al.]

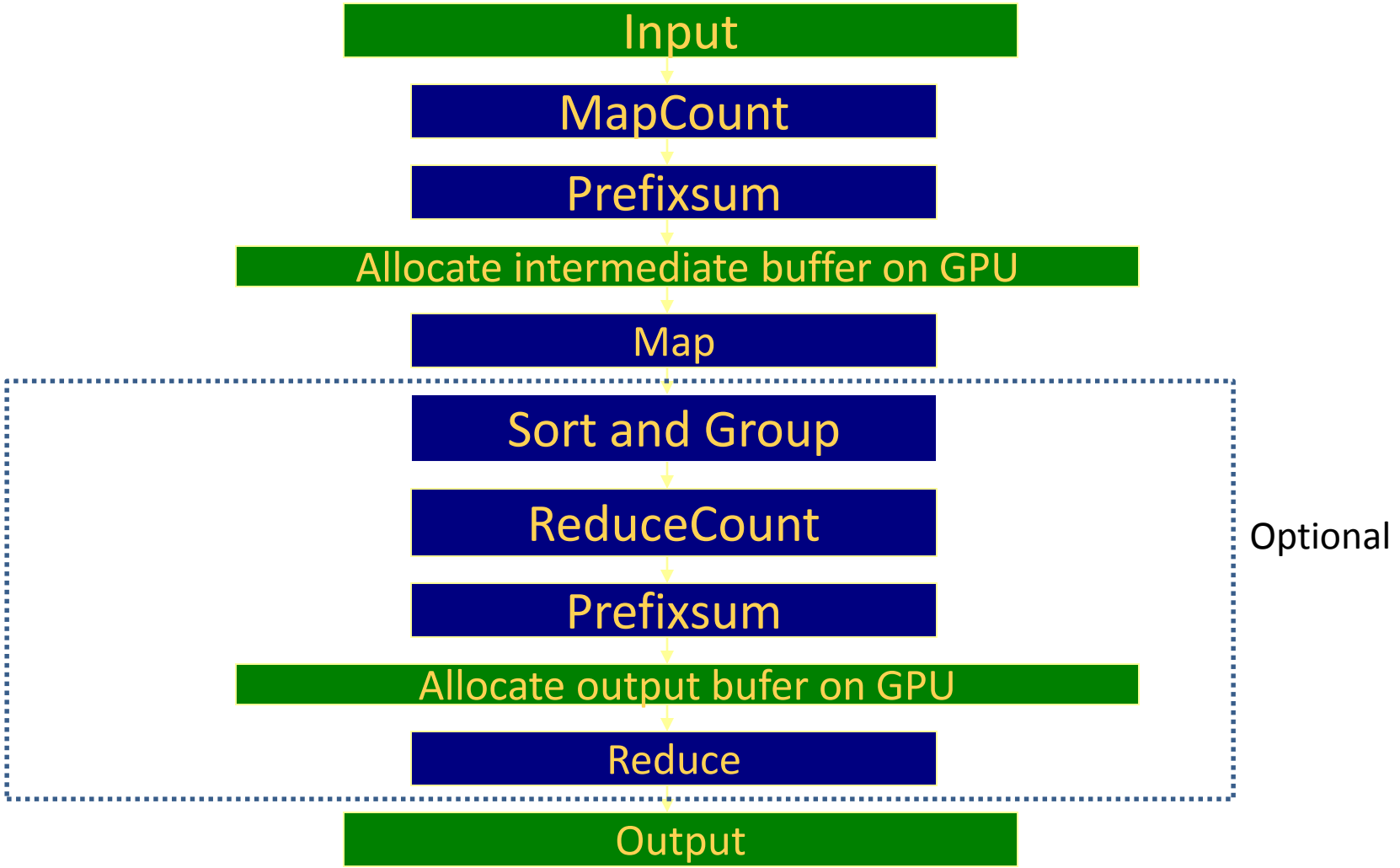
# Mars: MapReduce on the GPU

- A MapReduce system accelerated by the GPU
  - Mars Modules running on:
    - An NVIDIA GPU: MarsCUDA (MarsGPU)
    - An AMD GPU: MarsBrook
    - A Multi-core CPU: MarsCPU
    - Multi-core CPUs + GPUs: Co-processing
    - Distributed System: MarsHadoop
  - Design goals
    - Simple interface; high performance

# Challenges on the GPU

- Lack of support for global synchronization
  - Go for lock-free algorithms
- Dynamic memory allocation
  - Count memory size before allocation
  - Require user to provide mapcount and reducecount functions

# MarsGPU Workflow



# Map Result Output on the GPU

- Call user-defined MapCount function
  - Each function emits output key size and value size
- Prefix sum on output key sizes and value sizes
  - The total size of output buffer
  - The deterministic write position for each Map
- Allocate output buffer
- Call user-defined Map function
- Output records according to the write position

# Example of Map Result Output

Map1 → ”123456789”, Map2 → ”abcd”, Map3 → ”ABCDED”

MapCount

- MapCount1 → 9
- MapCount2 → 4
- MapCount3 → 6

Prefix Sum, Allocate buffer, and Map

- 9, 4, 6 – size array
- 0, 9, 13 – offset array
- Allocate a buffer of size 19
- Output: ”123456789abcdABCDED”

# Optimizations in MarsGPU

- Coalesced access
- Shared memory
- Built-in vector types (int4, float4)
- Page-locked host memory to improve PCI-E bus transfer bandwidth

# Experimental Setup

- Software
  - CPU: Phoenix, MarsCPU (#cpu thread = 4)
  - GPU: **MarsGPU**

| | CPU (P4 Quad) | GPU (NV GTX8800) |
|---|---|---|
| Processors (HZ) | 2.66G*4 | **1.35G*128** |
| Cache size | **8MB** | 256KB |
| Bandwidth (GB/sec) | 10.4 | **86.4** |
| OS | Fedora Core 7.0 (Linux) | |

# Workloads

- String Match (SM): Find the position of a string in a file.

  [S: 32MB, M: 64MB, L: 128MB]

- Inverted Index (II): Build inverted index for links in HTML files.

  [S: 16MB, M: 32MB, L: 64MB]

- Similarity Score (SS): Compute the pair-wise similarity score for a set of documents.

  [S: 512x128, M: 1024x128, L: 2048x128]

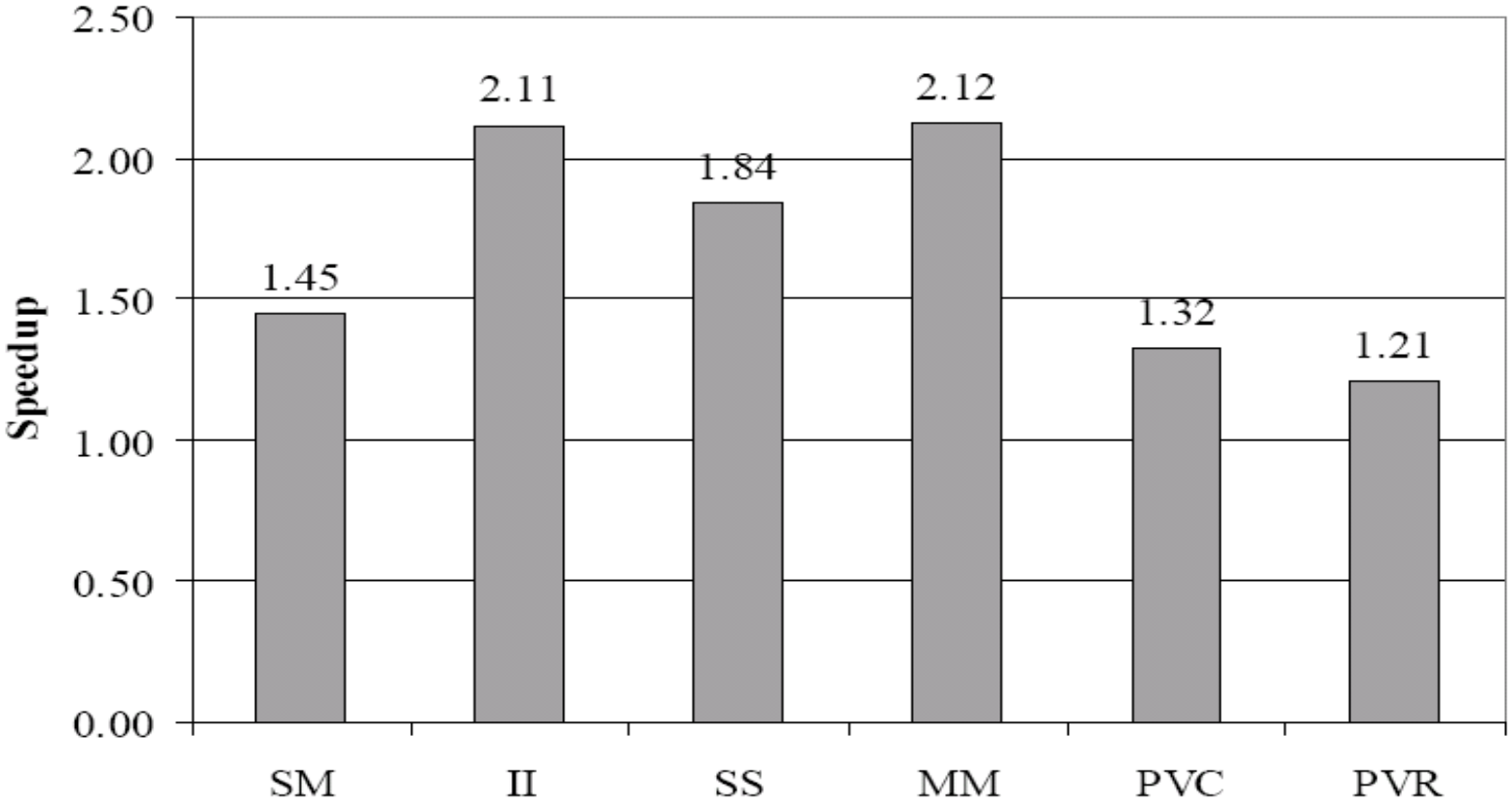
# Workloads (Cont.)

- Matrix Multiplication (MM): Multiply two matrices.
  [S: 512x512, M: 1024x10242, L: 2048x2048]

- Page View Rank (PVR): Count the number of distinct page views from web logs.
  [S: 32MB, M: 64MB, L: 96MB]

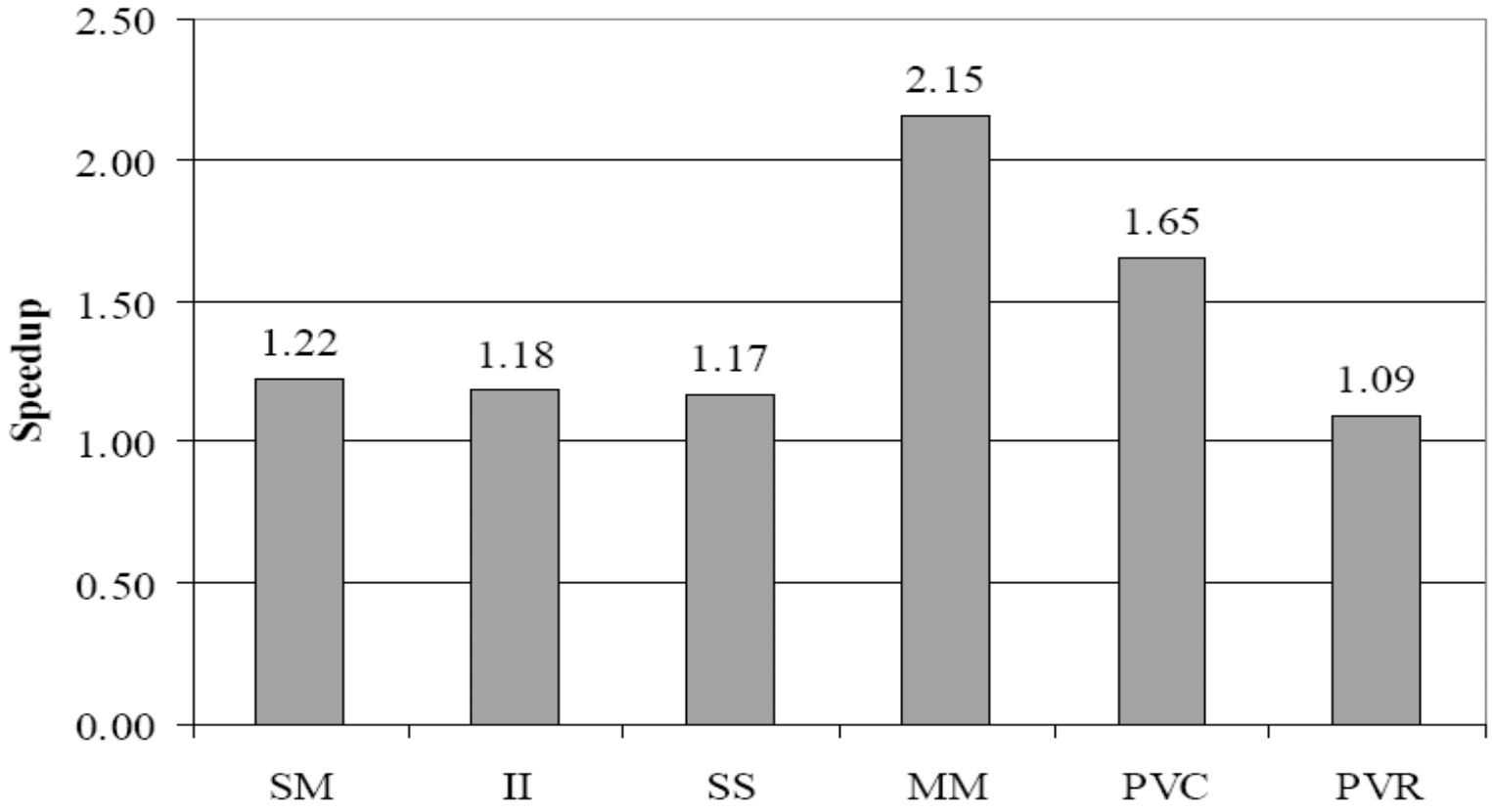
- Page View Count (PVC): Find the top-10 hot pages in the web log.
  [S: 32MB, M: 64MB, L: 96MB]

# Effect of Coalessed Access



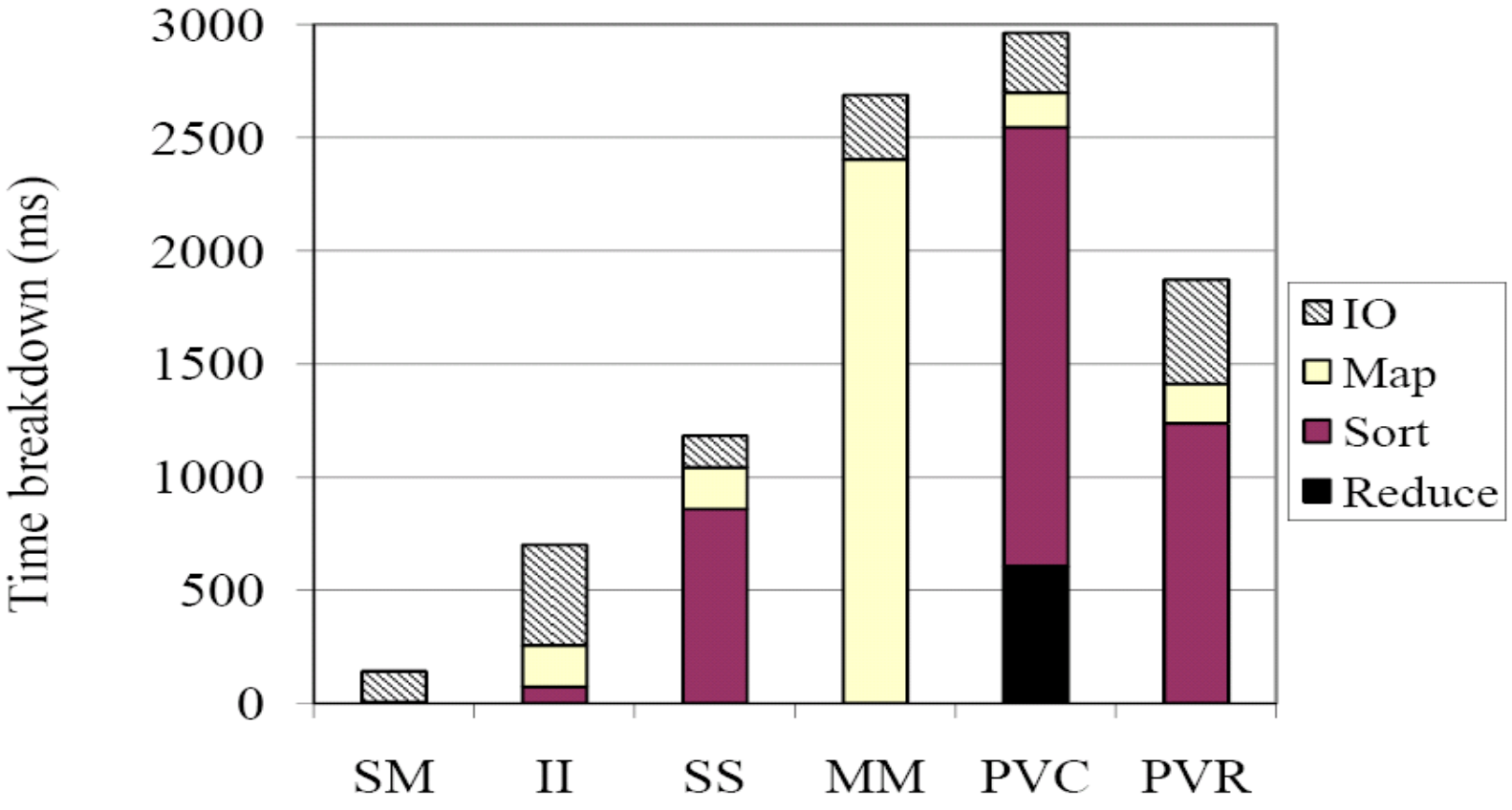

Coalessed access achieves a speedup of 1.2-2X

# Effect of Built-In Data Types



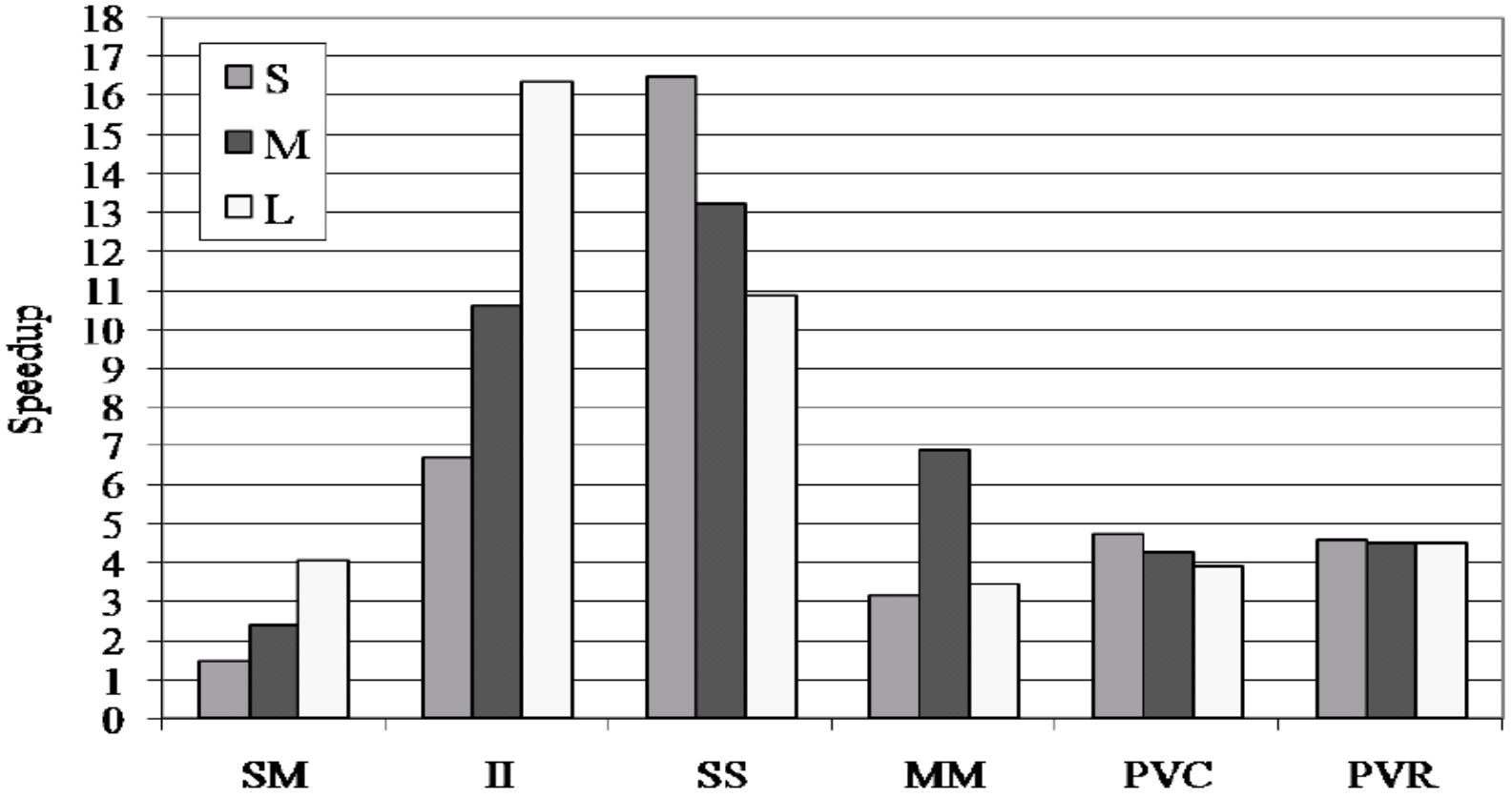

Built-in data types achieve a speedup up to 2 times

# GPU accelerates computation in MapReduce



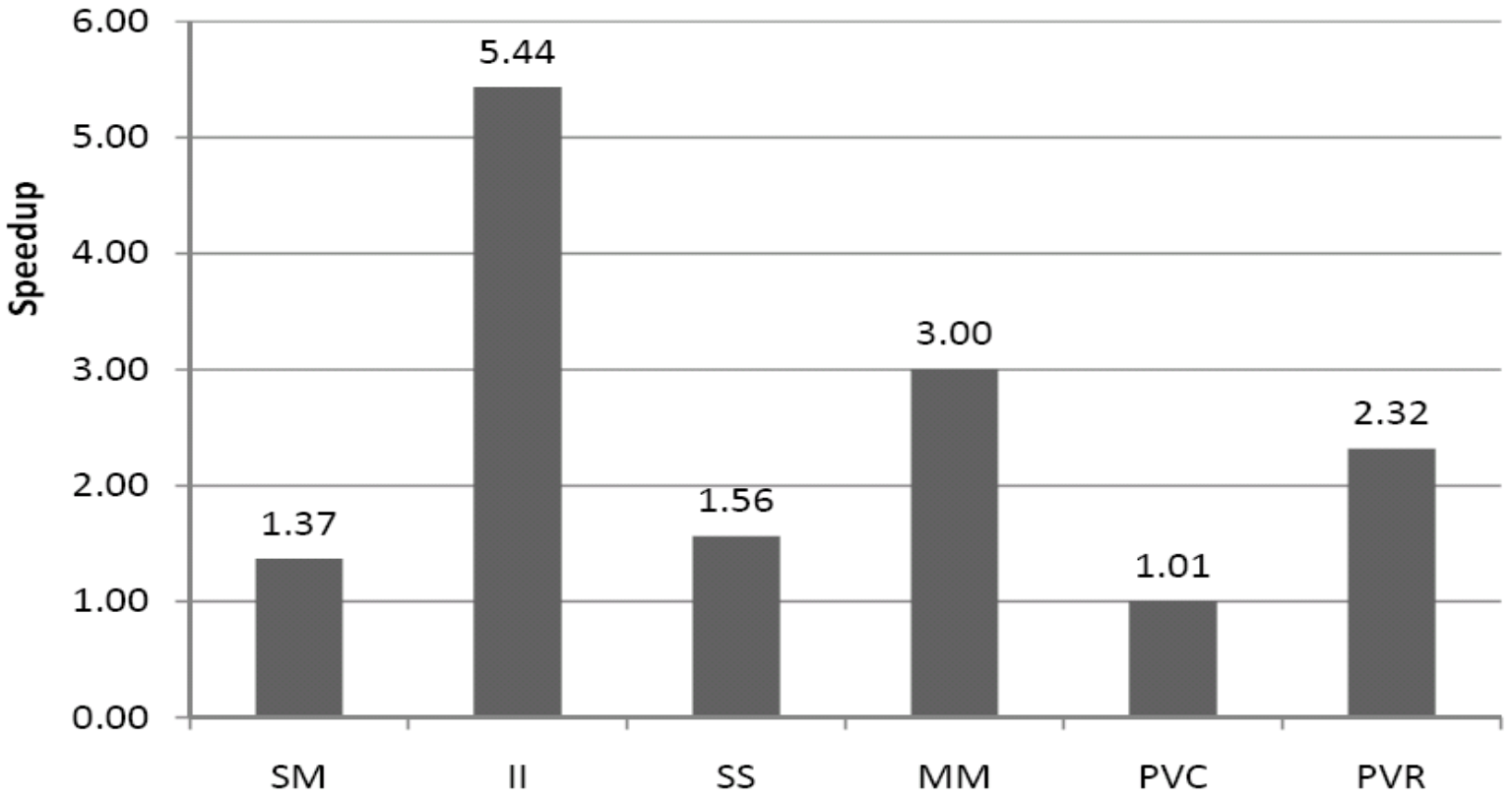

With large data set

# MarsGPU vs. Phoenix




The speedup is 1.5-16 times with various data sizes

# MarsCPU vs. Phoenix




MarsCPU is 1-5 times as fast as Phoenix

# Summary

- MapReduce is a familiar data-parallel programming framework on distributed computers.
- When some MapReduce nodes have GPUs installed, Mars becomes useful.
- MarsGPU speeds up a CPU-based MapReduce by up to an order of magnitude.

http://www.cse.ust.hk/gpuqp/Mars.html