CMSC 691
High Performance Distributed Systems

Big Data and MapReduce

Dr. Alberto Cano
Assistant Professor
Department of Computer Science
acano@vcu.edu

Server upgrade!!

- New address: **ace-tesla.egr.vcu.edu**
- Username: VCUeID Password: VCUpassword (not V#)
- Run the following commands to finish the setup:

```
$ ssh ace-tesla.egr.vcu.edu
$ echo "export PATH=/usr/local/cuda/bin:$PATH" >> ~/.bashrc
$ echo "export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH" >> ~/.bashrc
$ source ~/.bashrc
```

- Server hardware:
  - 24 CPU cores and 96 GB RAM
  - 2 GPUs Titan X (3072 cores each) CC 5.2
  - 6 GPUs Tesla C2050/70 (448 cores each) CC 2.0
- **Backup any files you have currently in 172.18.199.200**

What is Big Data?

- No single definition, nobody knows but everyone talks about it
- Gartner definition: “Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation”
- Reality: big data is the term for a collection of data sets so large and complex that it becomes difficult to process using traditional database tools, data processing applications, and data mining algorithms in reasonable time
- Challenges: capture, clean, storage, search, sharing, transfer, analysis, processing and visualization of data

Characteristics of big data (7 V's)

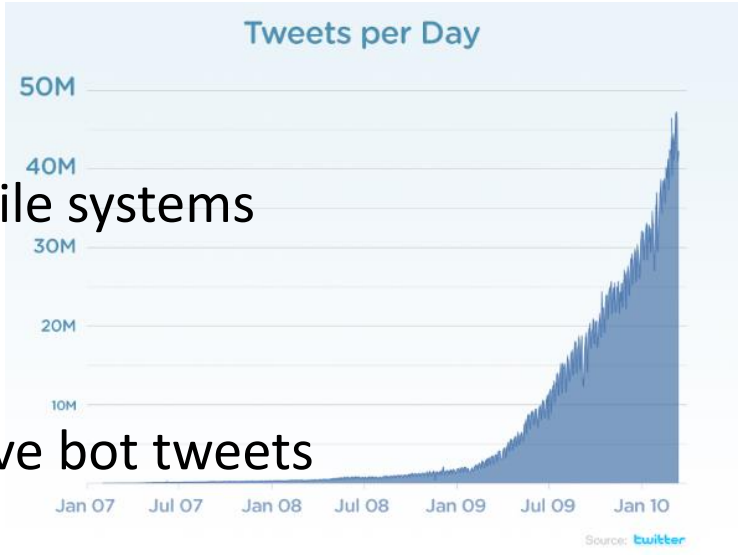
- Volume: amount of data generated and stored
- Velocity: speed at which data is generated
- Variety: type and nature of data
- Veracity: quality of data
- Variability: inconsistency of data
- Value: importance and usefulness of data
- Visibility: visualization of data

Big Data: volume

- The amount of data generated and stored in a system
- When is a problem considered big? terabytes, petabytes, exabytes?
- Reality: many real-world information systems show exponential increase in generated data
- Problems with large volumes of data:
  - How to store the data?

    Fault-tolerant redundant distributed file systems
  - How to process the data?

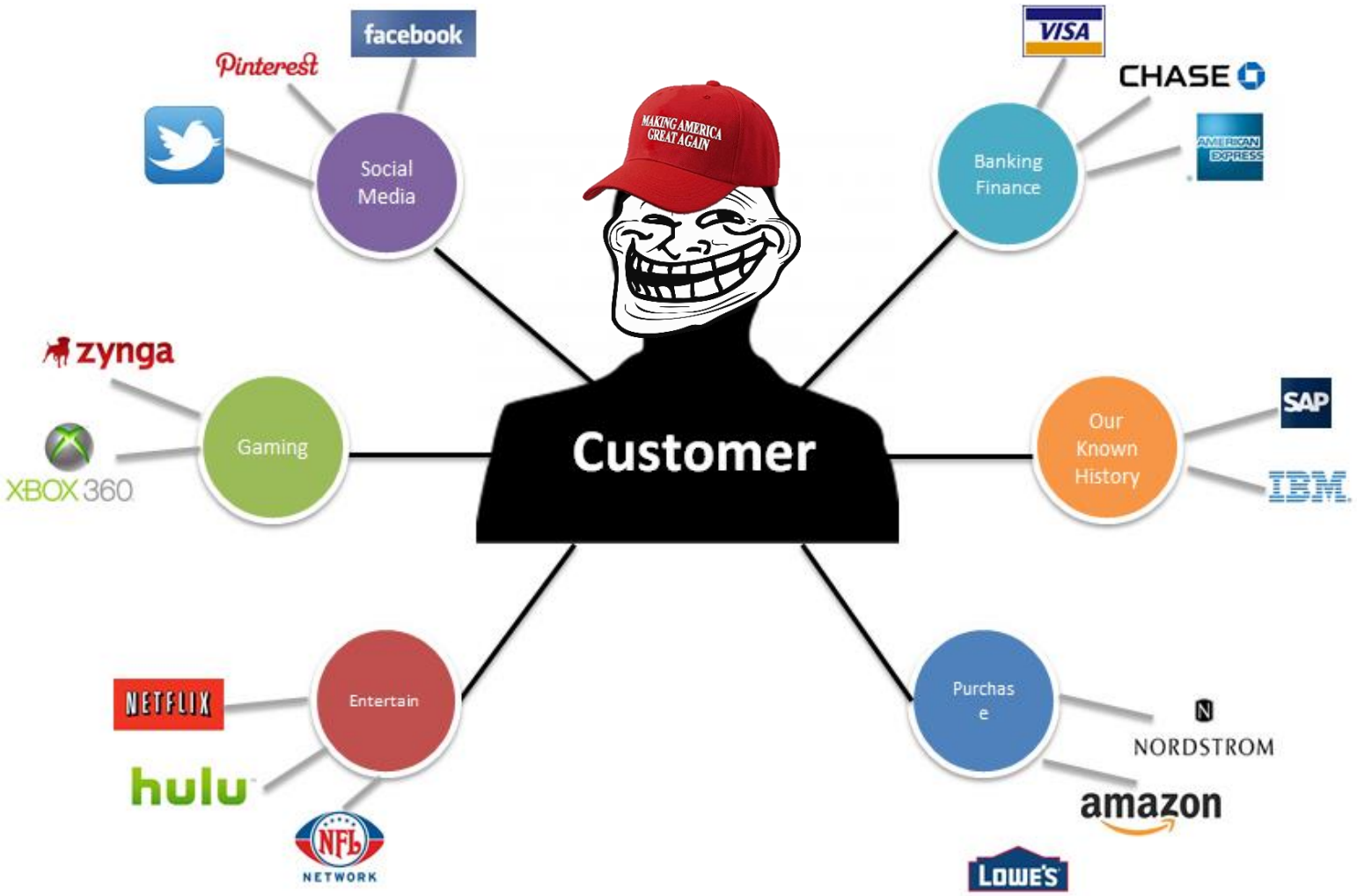
    Machine learning algorithms to remove bot tweets



Big Data: variety

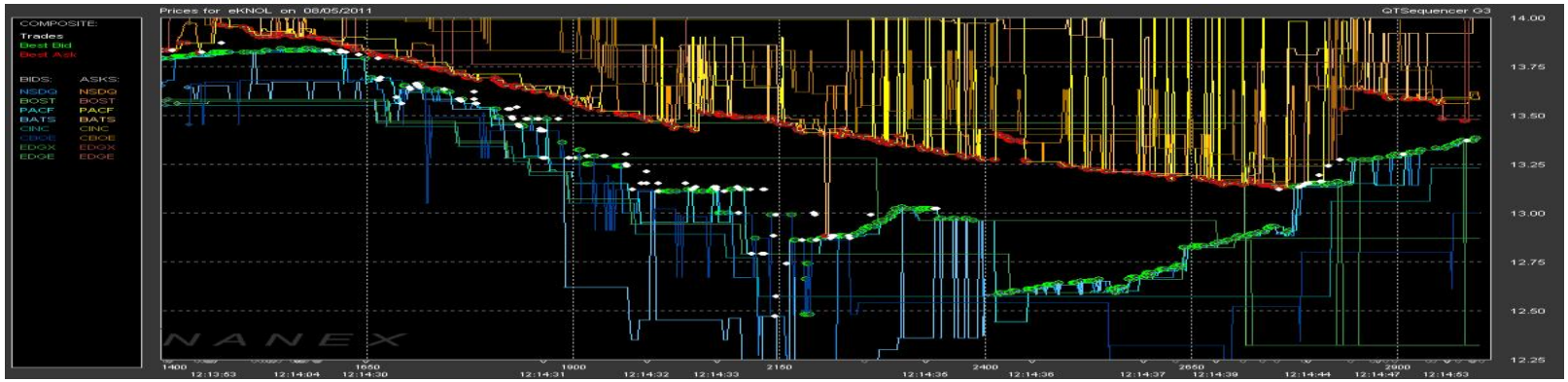
- Data coming from multiple sources with different structured and unstructured formats
- Relational data (SQL), Text data (web), semi-structured data (XML), graph data (social networks), streaming data, image data, online public data
- Building a knowledge discovery system handling all input formats and sources is extremely difficult. Example: medical decision support system with data feed from images, clinical records, geographic data, events data, etc altogether

Data -> Information -> Knowledge is **money** and they make money off **you**



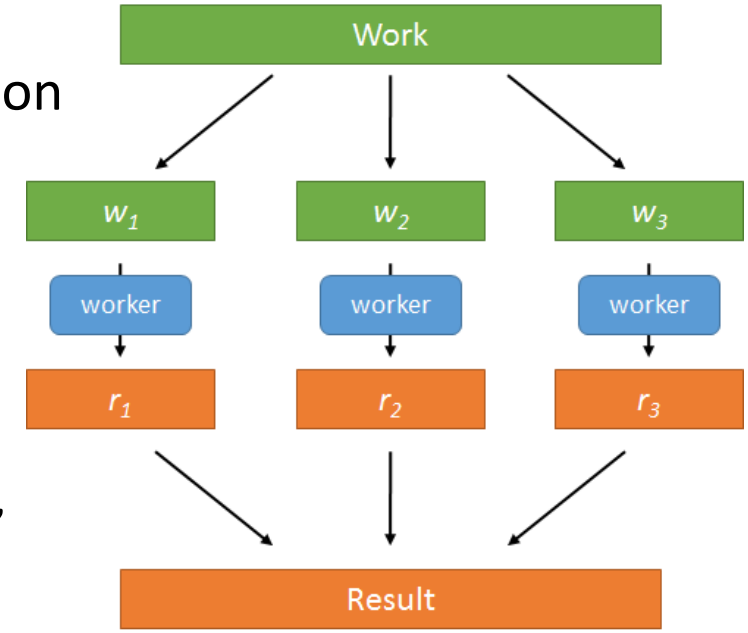
Big Data: velocity

- Data is generated fast and need to be processed faster
- Not simply adding a bigger cluster will make a given data to be processed faster
- Online data analytics, data streams mining
- Late decisions = missing opportunities = no money
- Example: high frequency trading in the stock market



Divide and conquer

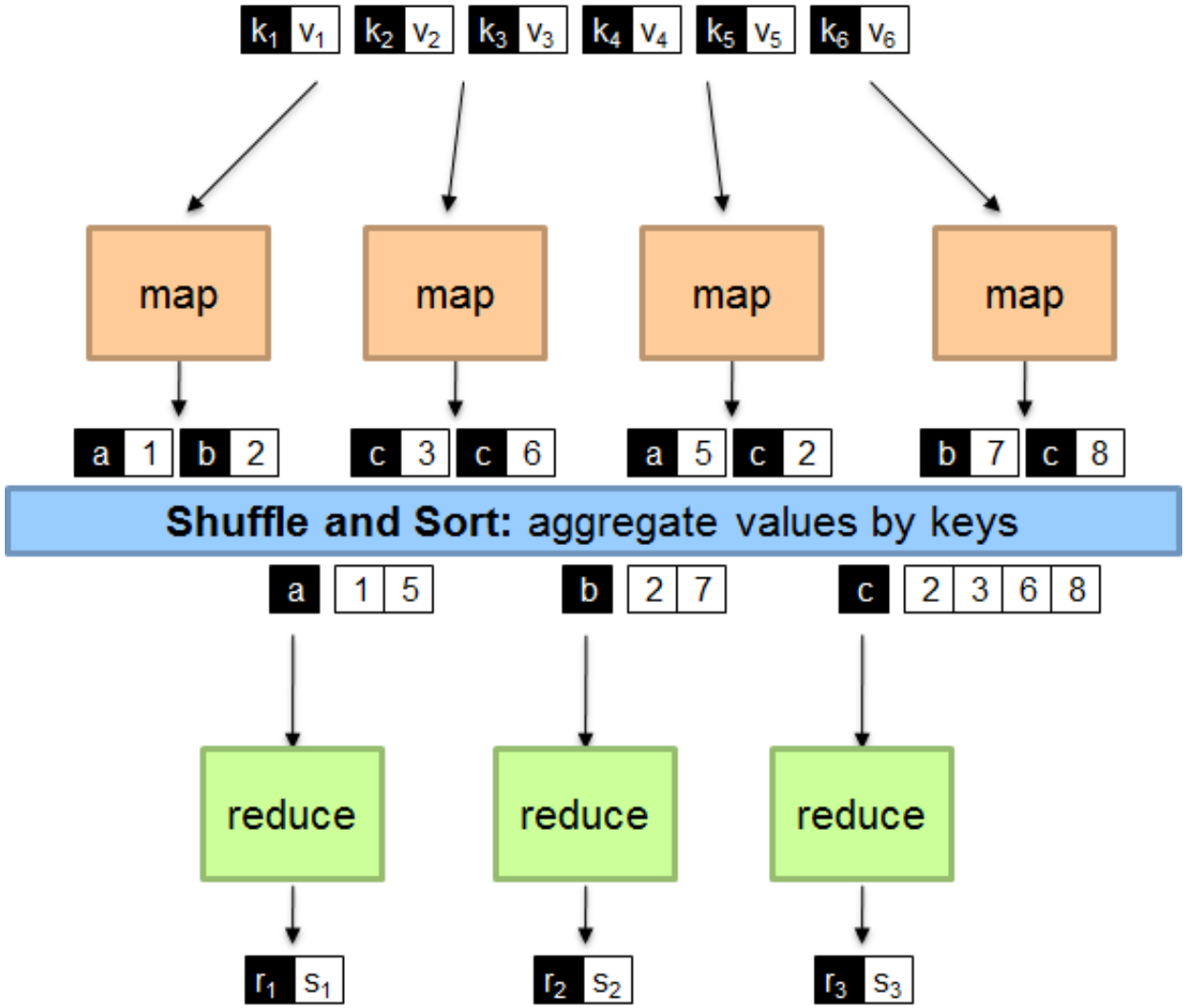
- Workload divided into multiple parallel workers, each worker resolves a portion of the problem
- We've already discussed parallelization in the GPU having multiple blocks of threads with shared memory
- Parallelization challenges: efficient distribution of the workload, synchronization and critical sections, aggregation of results, fault-tolerant workers, communication overheads



MapReduce

- Programming model for processing large data sets (k,v) with a distributed algorithm on a cluster
- MapReduce is a Google proprietary technology while Apache Hadoop is a open-source implementation in Java
- Advantages: provides automatic parallelization and distribution, fault tolerance, I/O scheduling, monitoring and status updates
- Programmers specify to functions:
  - map (k, v) → [(k', v')]
  - reduce (k', [v']) → [(k', v')]
  - All values with the same key are sent to the same reducer
  - The execution framework handles everything else

MapReduce


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 3 6 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3

MapReduce runtime automatically

- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles data distribution
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts tasks
- Everything happens on top of a distributed file system

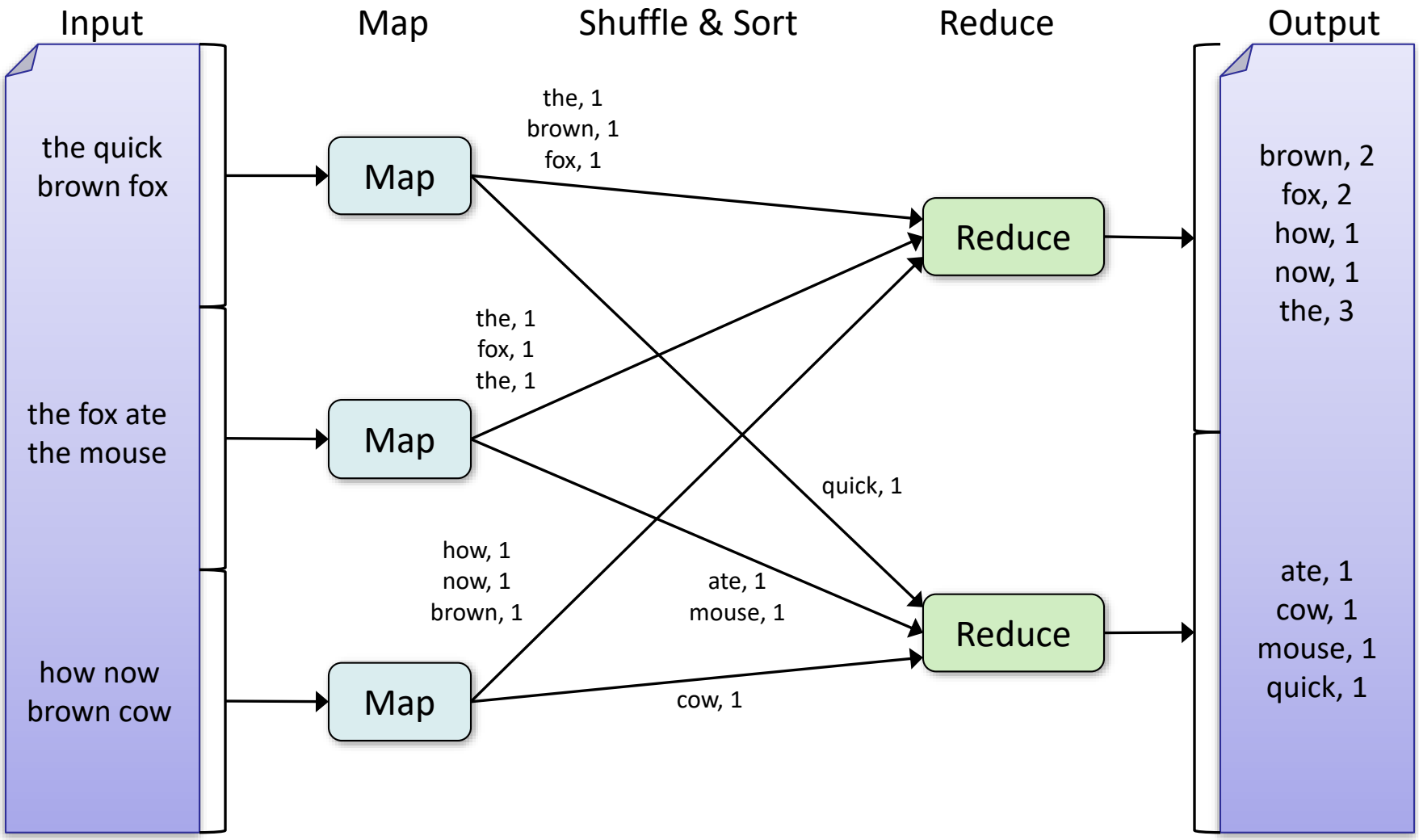
HelloWorld: Word Count

- Compute the frequency of every word in a very large document
- Naïve sequential approach: loop every word in the document and keep a HashTable<String, Integer>
- MapReduce approach (in parallel):

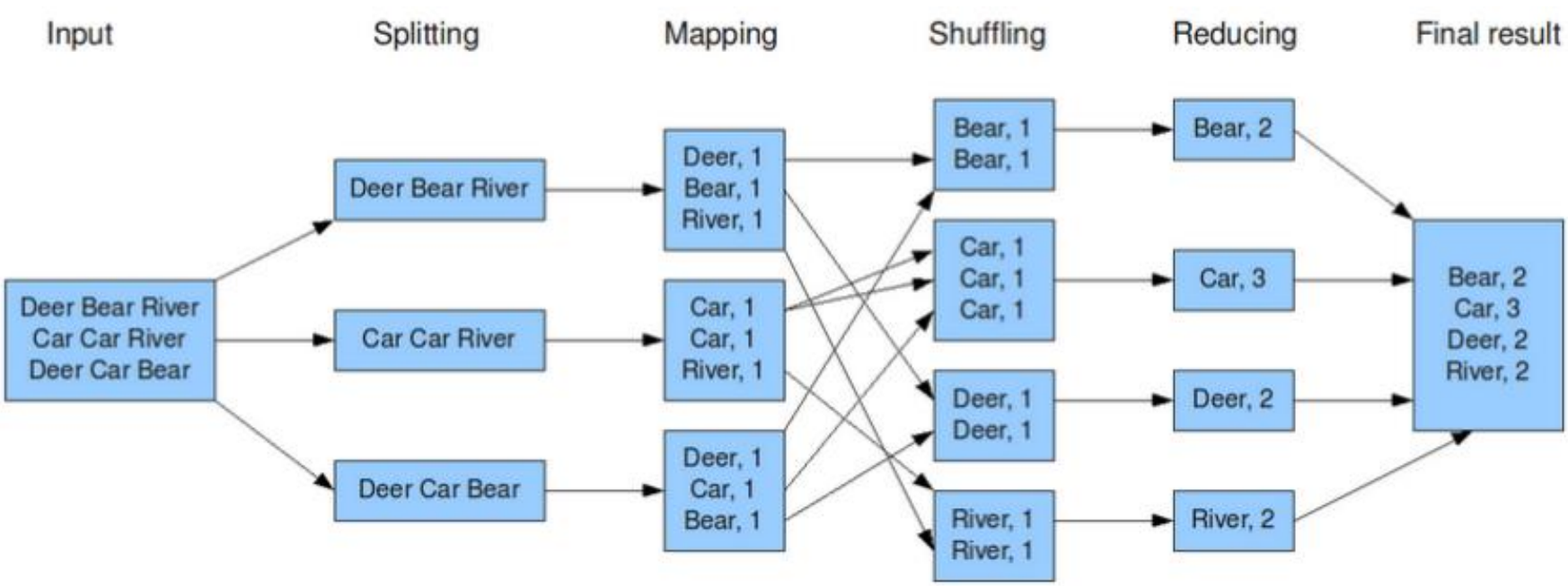
```
Map(String text):
    for each word w in text:
        Emit(w, 1);

Reduce(String word, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum += v;
    Emit(term, sum);
```

HelloWorld: Word Count



HelloWorld: Word Count



MapReduce

- What would be your approach for the midterm exercise?

f_3: Shifted Ackley's Function

$$f_3(\mathbf{z}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e$$

- $\mathbf{z} = \Lambda^{10} T_{\text{asy}}^{0.2}(T_{\text{osz}}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$
- $\mathbf{x} \in [-32, 32]^D$
- Global optimum: $f_3(\mathbf{x}^{\text{opt}}) = 0$

- What would be your approach for a MapReduce KNN?

Input: $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)\}$

$\mathbf{x} = (x_1, \ldots, x_n)$ new instance to be classified

FOR each labelled instance (x_i, c_i) calculate $d(\mathbf{x}_i, \mathbf{x})$

Order $d(\mathbf{x}_i, \mathbf{x})$ from lowest to highest, $(i = 1, \ldots, N)$

Select the K nearest instances to $\mathbf{x}$: $D_{\mathbf{x}}^K$

Assign to $\mathbf{x}$ the most frequent class in $D_{\mathbf{x}}^K$

CMSC 691
High Performance Distributed Systems

Big Data and MapReduce

Dr. Alberto Cano
Assistant Professor
Department of Computer Science
acano@vcu.edu