# Optimizing Data Partitioning for Data-Parallel Computing

*Qifa Ke*, Vijayan Prabhakaran, Yinglian Xie, Yuan Yu

Microsoft Research Silicon Valley

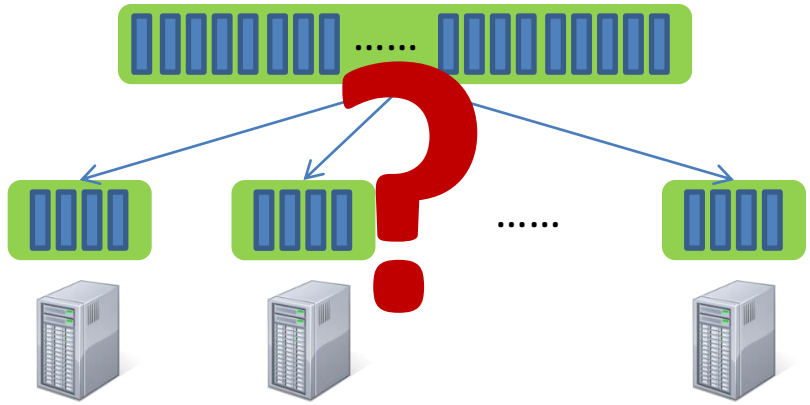
Jingyue Wu, Junfeng Yang

Columbia University

# Partition Data for Data-Parallel Computing

```
// 270 GB input data

var output =
      input.GroupBy(x => x.UserId)
            .Select(g => GetStats(g))
```



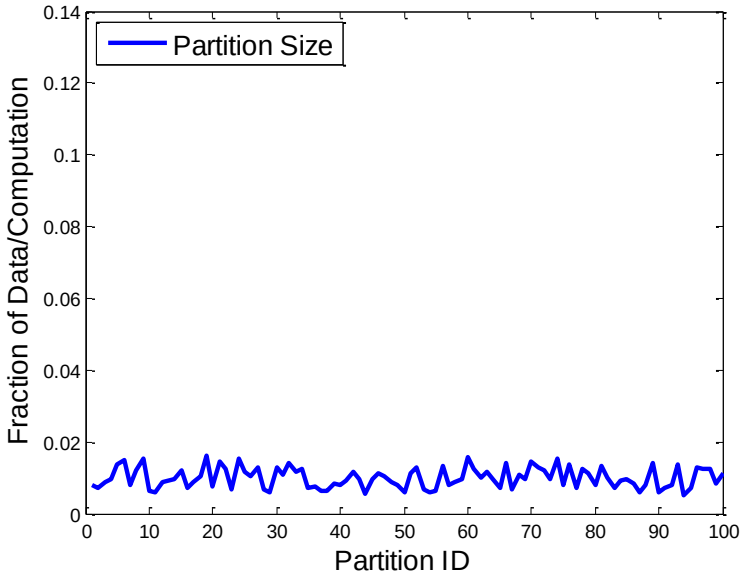

- Data partitioning controls the degree of parallelism
- What partition function to choose?
  - Hash partition, range partition, …?
- How many partitions to generate?
  - 100, 1000, 10000, ….?

**Data partitioning ⇨ performance and costs**

# Problem 1: Do We have a Skew?

- Data skew and **computation skew**

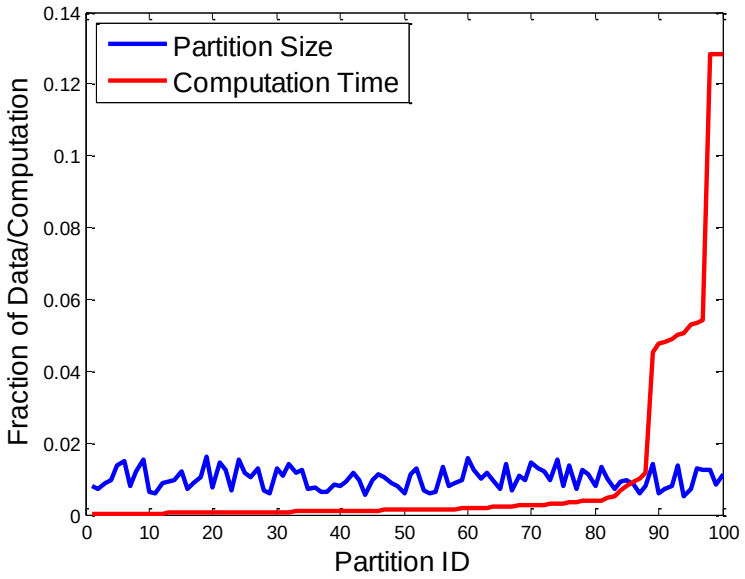
```
// process 20 GB images in 100 partitions
var output =  Imgs.Select( x => ProcessImages (x))
```



# Problem 1: Do We have a Skew?

- Data skew and **computation skew**

```
// process 20 GB images in 100 partitions
var output =  Imgs.Select(x=>ProcessImages (x))
```



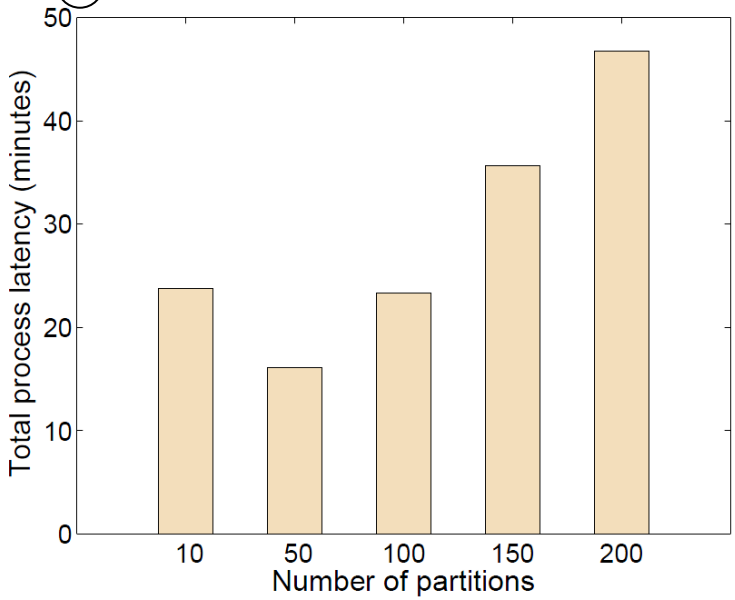

- Image processing time depends on both image and `ProcessImage()`:
  - Number of images
  - Image features

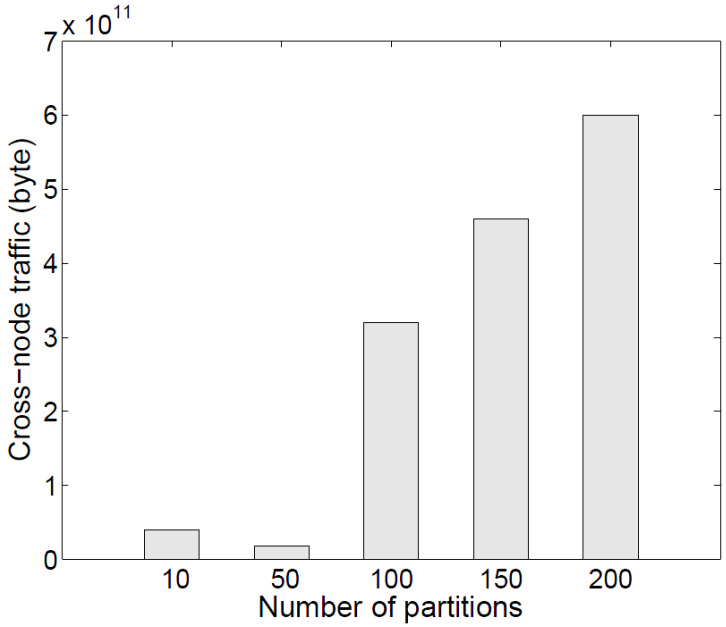
  `ProcessImage()` is targeting to compute

# Problem 2: What's Optimal?

- Balanced workload ≠ optimal performance
  - Tradeoff: workload vs. cross-node traffic

```
// construct a user-user graph for botnet deteciton
var records = input1.Apply(x => SelectRecords(x)).HashPartition(x=>x.label, nump);
var output  = input1.Apply(records, (x,y) => ConstructGraph(x,y));
```



# Optimal Data Partitioning

Given code and data, can we generate a data partitioning scheme to optimize performance, without running code on whole data set?

- Performance and cost metrics
  - Job latency
  - Number of processes
  - Memory consumption
  - Disk and network I/O

# Why not DB Solutions

- Need to understand both **code** and **data**
- Programming model
  - Predefined operators (e.g., select, join) vs. arbitrary user-defined functions (UDF)
- Data model
  - Structured tables vs. unstructured data
  - Static, indexed data vs. dynamic dataset
  - Minimize intermediate disk writes vs. using disk as communication channel

# Code Analysis


- Data processing flow
- Computational & I/O complexity
- Relevant data features

- Challenges: user defined functions (UDF)
  - How data is accessed, processed, and transformed

```
IEnumerable<stats> ProcessRecord( Ienumerable<record> users) {
    foreach (var u in users) {
        if (NumRecipients(u)  > 10) {
            yield return GetStat(u);
        } else {
            yield return GetSimpleStat(u);
        }
    }
}
```

- **Number of recipients is a relevant feature**
- **Different records take different code paths to process**

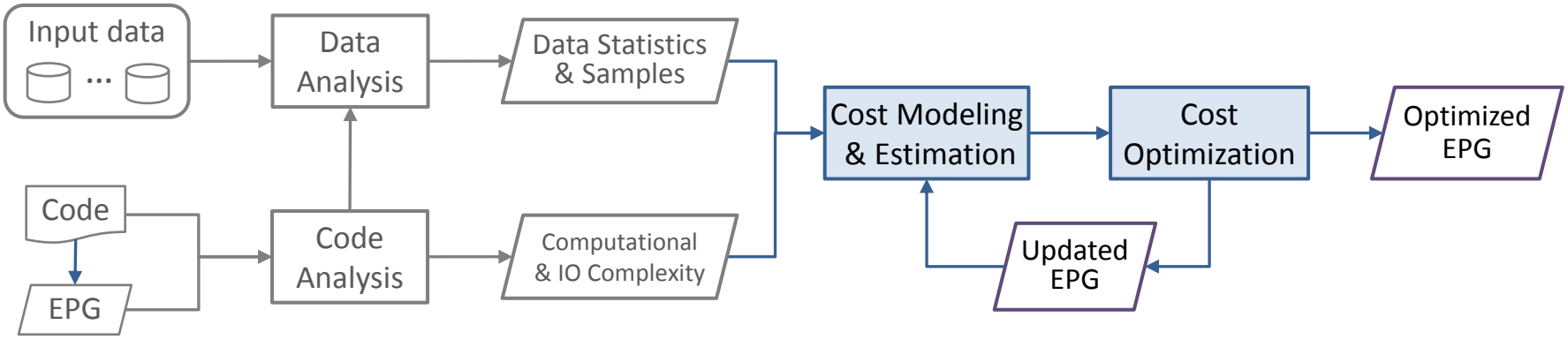
# Data Analysis


Statistics of relevant data features

- Challenge: compact data representation
  - Representative samples of input data
  - Data summarizations
  - Approximate histogram
  - Approximate number of distinct keys
- Streaming algorithms in a distributed setting

# Cost Modeling and Optimization

- **Modeling:** compare different partitioning schemes
- **Estimation:** predict the potential cost
  - White-box approach
    - Analytically based on code/data analysis
  - Black-box approach
    - Sampling + regression analysis
- **Optimization:** search for best partitioning scheme



# Conclusion

- Preparing your input before you start
  - Data partitioning is critical to performance
- New research opportunities in different fields
  - Programming language analysis
  - Data analysis
  - Optimization
  - Distributed systems