Cost-effective and Privacy-conscious MapReduce Cloud services

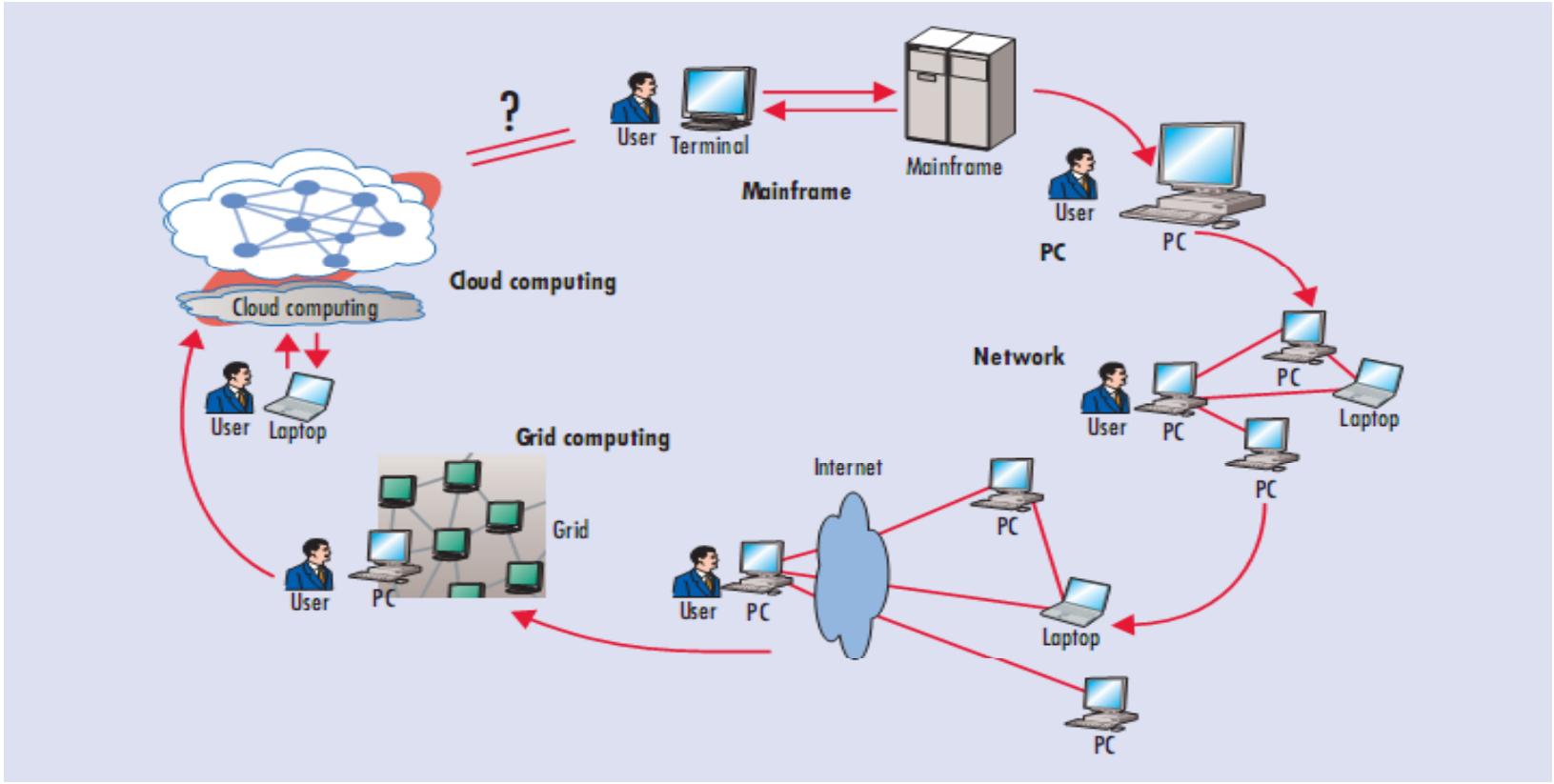
Guest lecture by

Dr. Balaji Palanisamy

Computing Paradigm Shift

Cloud Challenges

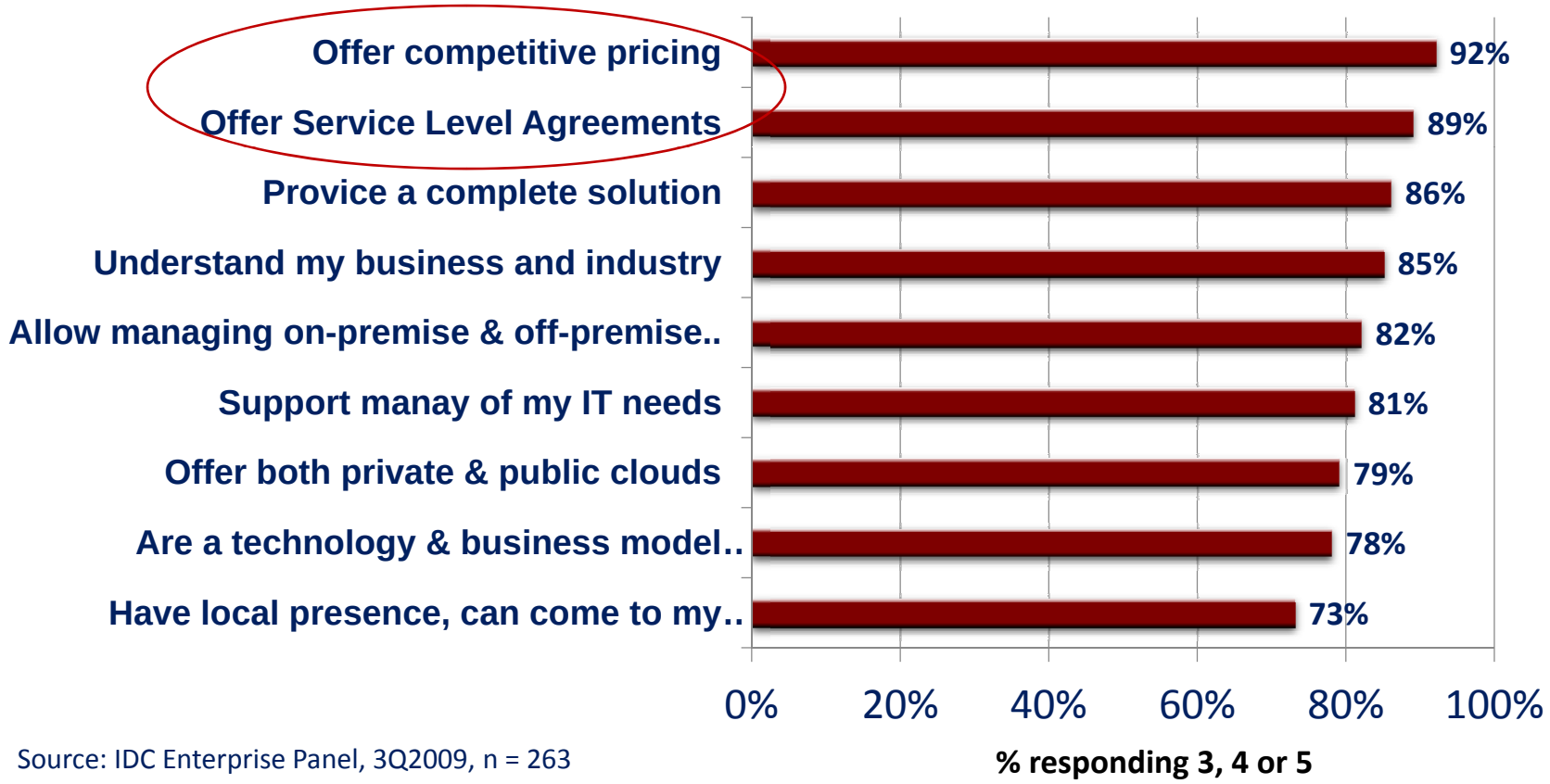
- Cost-effectiveness, Performance and Security



Cloud Computing Challenges

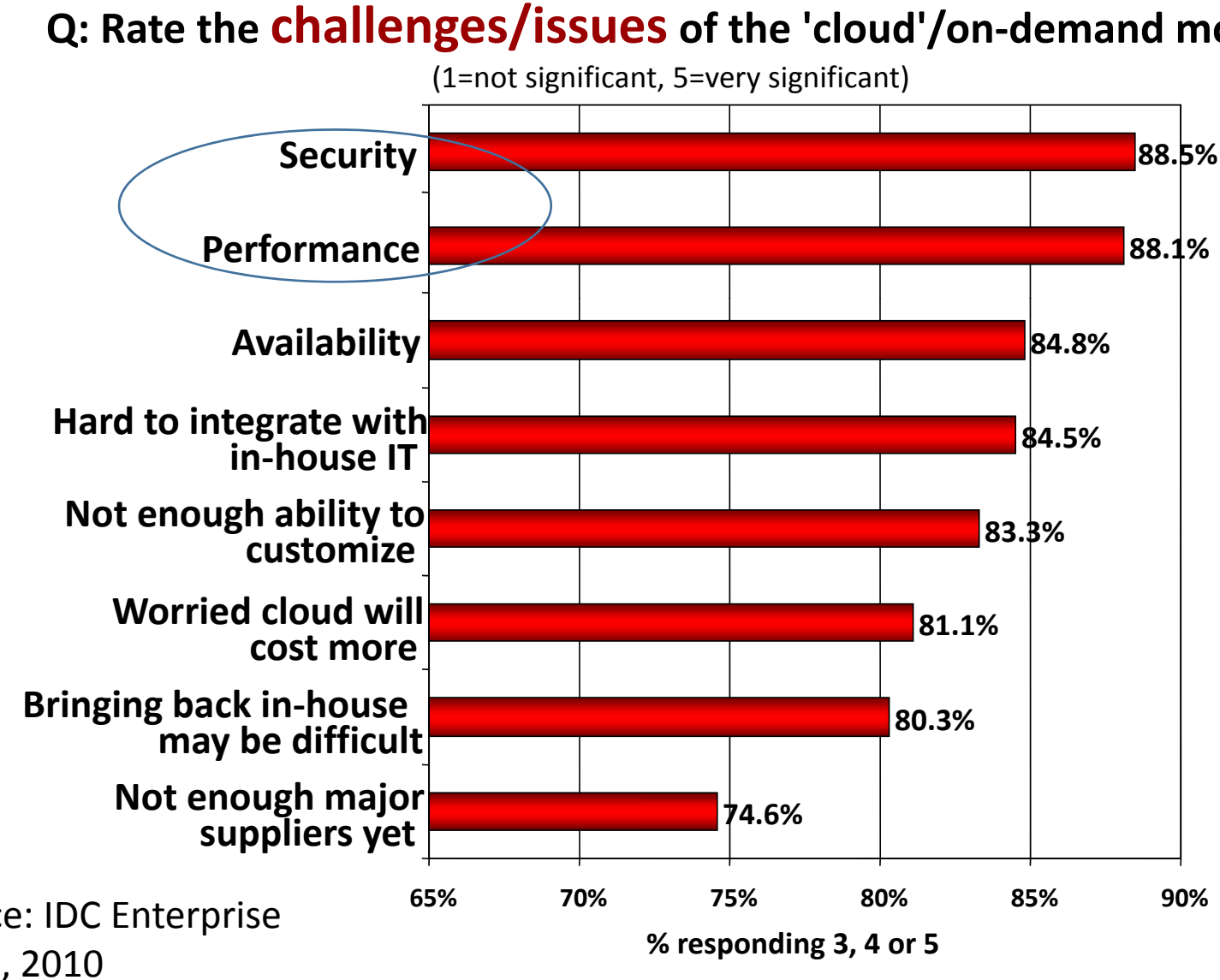
Q: How important is it that cloud service providers…

(scale: 1-5; 1=not at all important, 5=very important)



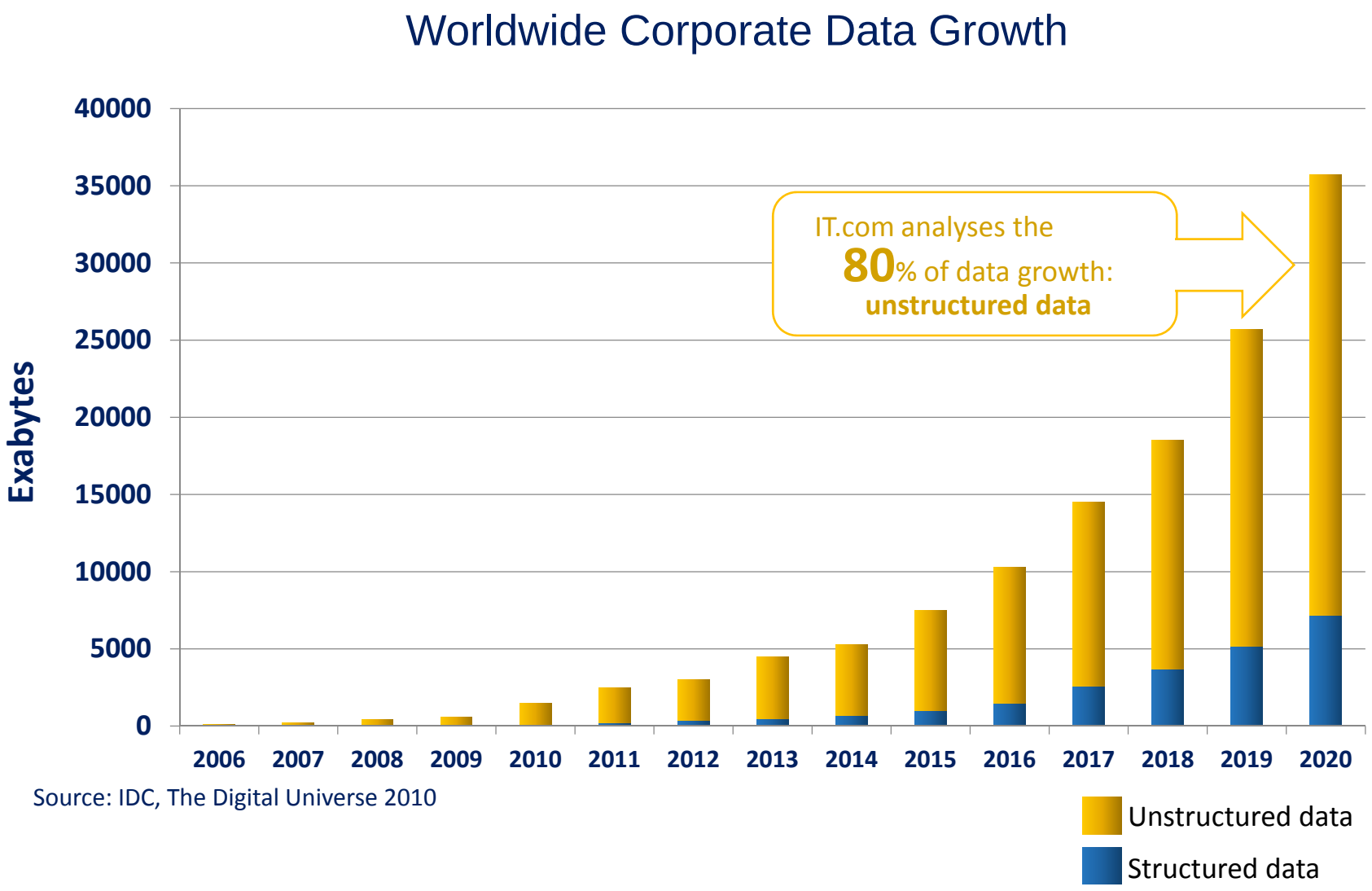
Source: IDC Enterprise Panel, 3Q2009, n = 263

Cloud Computing Challenges



Source: IDC Enterprise Panel, 2010

Data Growth



Source: IDC, The Digital Universe 2010

Big Data processing in a Cloud

- MapReduce and Big Data Processing
  - Programming model for data-intensive computing on commodity clusters
  - Pioneered by Google
    - Processes 20 PB of data per day
  - **Scalability** to large data volumes
    - Scan 100 TB on 1 node @ 50 MB/s = 24 days
    - Scan on 1000-node cluster = 35 minutes
  - It is estimated that, by 2015, more than half the world's data will be processed by Hadoop – Hortonworks
- Existing Dedicated MapReduce Clouds
  - Natural extensions of rental store models
  - Performance and cost-inefficiency

Outline

- Goals:
  - Cost-effective and scalable Big Data Processing using MapReduce
  - Privacy conscious Data access and Data processing in a cloud

- Outline

  - Cost-aware Resource Management
    - Goal: to devise resource management techniques that yield the required performance at the lowest cost.
    - Cura – Cost-optimized Model for MapReduce in a Cloud

  - Performance-driven Resource Optimization:
    - Goal: Optimizations based on physical resource limits in a data center such as networking and storage bottlenecks
    - Purlieus – Locality aware Resource Allocation for MapReduce

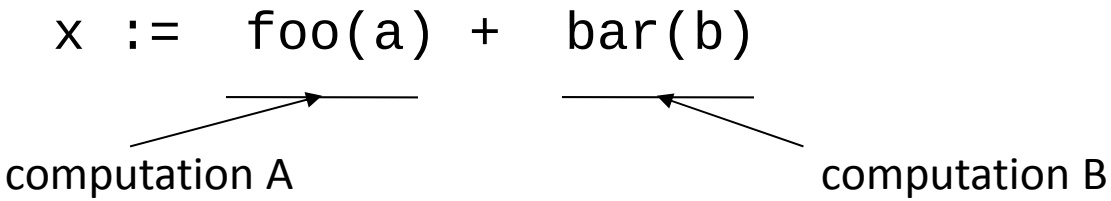
  - Privacy-conscious MapReduce Systems
    - VNCache – MapReduce analysis for Cloud-archived data
    - Airavat: Security and Privacy for MapReduce

MapReduce - Data Parallelism

```
x := (a * b) + (y * z);
```

- At the micro level, independent algebraic operations can *commute – be processed in any order.*
- *If commutative operations are applied to different memory addresses, then they can also occur at the same time*
- *Compilers, CPUs often do so automatically*

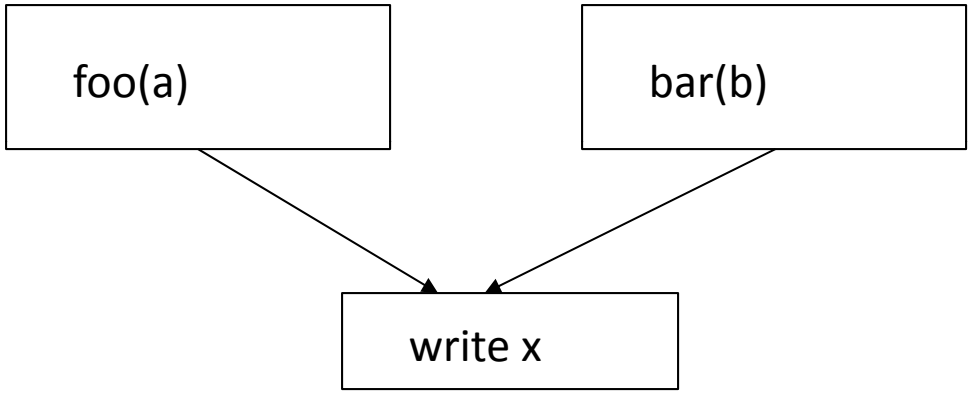
Higher-level Parallelism



- *Commutativity can apply to larger operations. If foo() and bar() do not manipulate the same memory, then there is no reason why these cannot occur at the same time*

Parallelism: Dependency Graphs

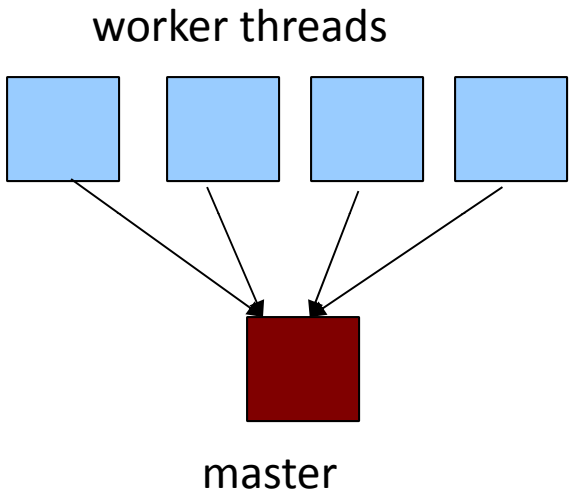
```
x :=  foo(a) +  bar(b)
```



- Arrows indicate dependent operations
- If *foo* and *bar* do not access the same memory, there is not a dependency between them
- These operations can occur in parallel in different threads
- *write x* operation waits for predecessors to complete

Master/Workers

- One object called the *master initially owns all data.*
- *Creates several workers to process individual elements*
- *Waits for workers to report results back*

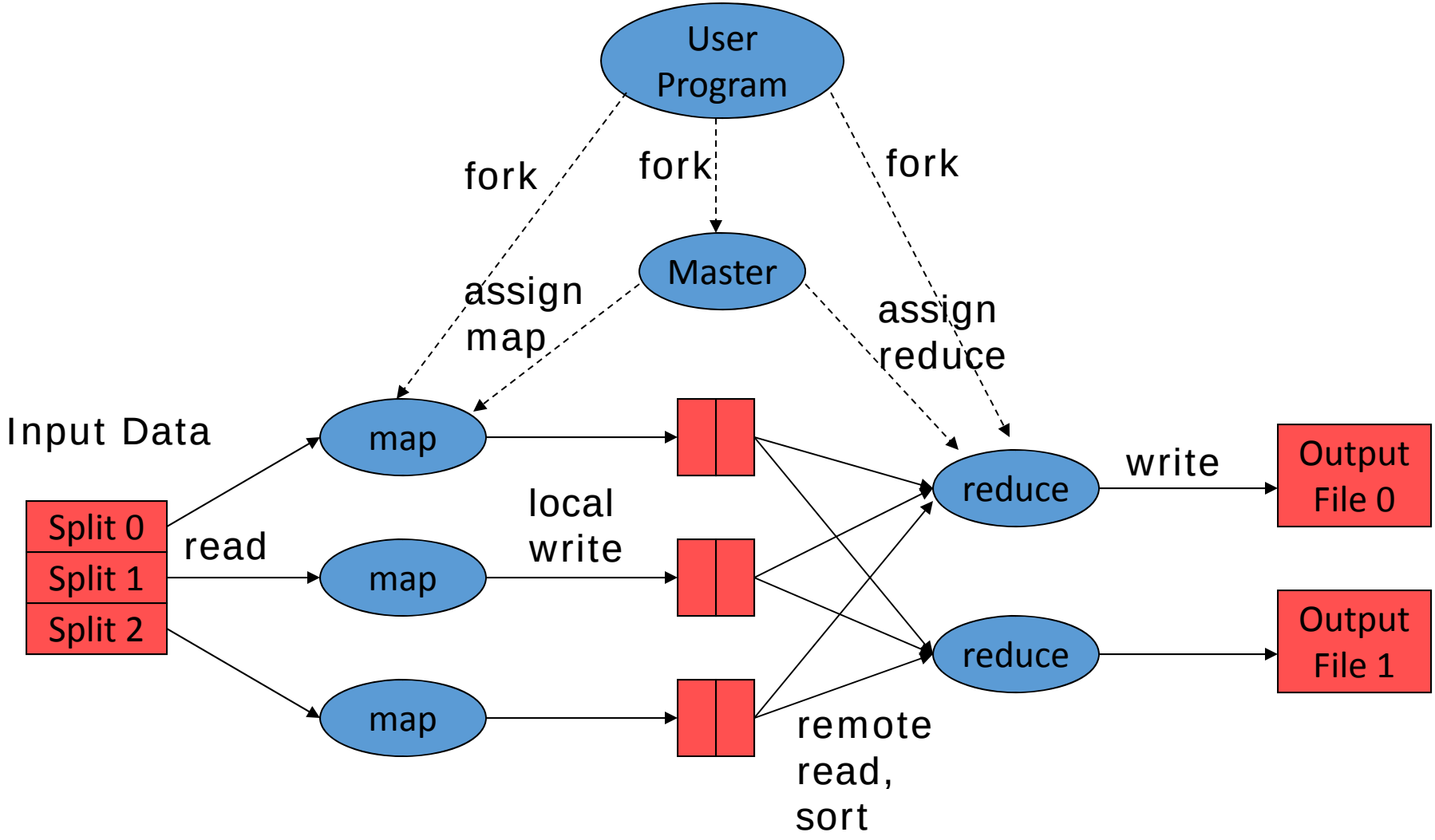


Example: Count word occurrences of each word in a large collection of documents

```
map(String input_key, String input_value):
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    EmitIntermediate(w, "1");


reduce(String output_key, Iterator
  intermediate_values):
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += ParseInt(v);
 Emit(AsString(result));
```

MapReduce Execution Overview



Need for Cost-optimized Cloud Usage Model

Existing Per-job Optimized Models:

- *Per-job customer-side greedy optimization may not be globally optimal*
- *Higher cost for customers*

Cura Usage Model:

User submits job and specifies the required service quality in terms of job response time and is charged only for that service quality

Cloud provider manages the resources to ensure each job's service requirements

Other Cloud Managed Resource models:

Database as a Service Model

Eg: Relational Cloud (CIDR 2011)
Cloud managed model for Resource management

Cloud managed SQL like query service
Delayed query model in Google Big Query execution results in 40% lower cost

User Scheduling Vs Cloud Scheduling

Job#	Arrival time	Deadline	Running time	Optimal no. Of VMs
1	20	40	20	20
2	25	50	20	20
3	30	75	20	20
4	35	85	20	20

User Scheduling

Job #	1	2	3	4
Start time	20	25	30	35

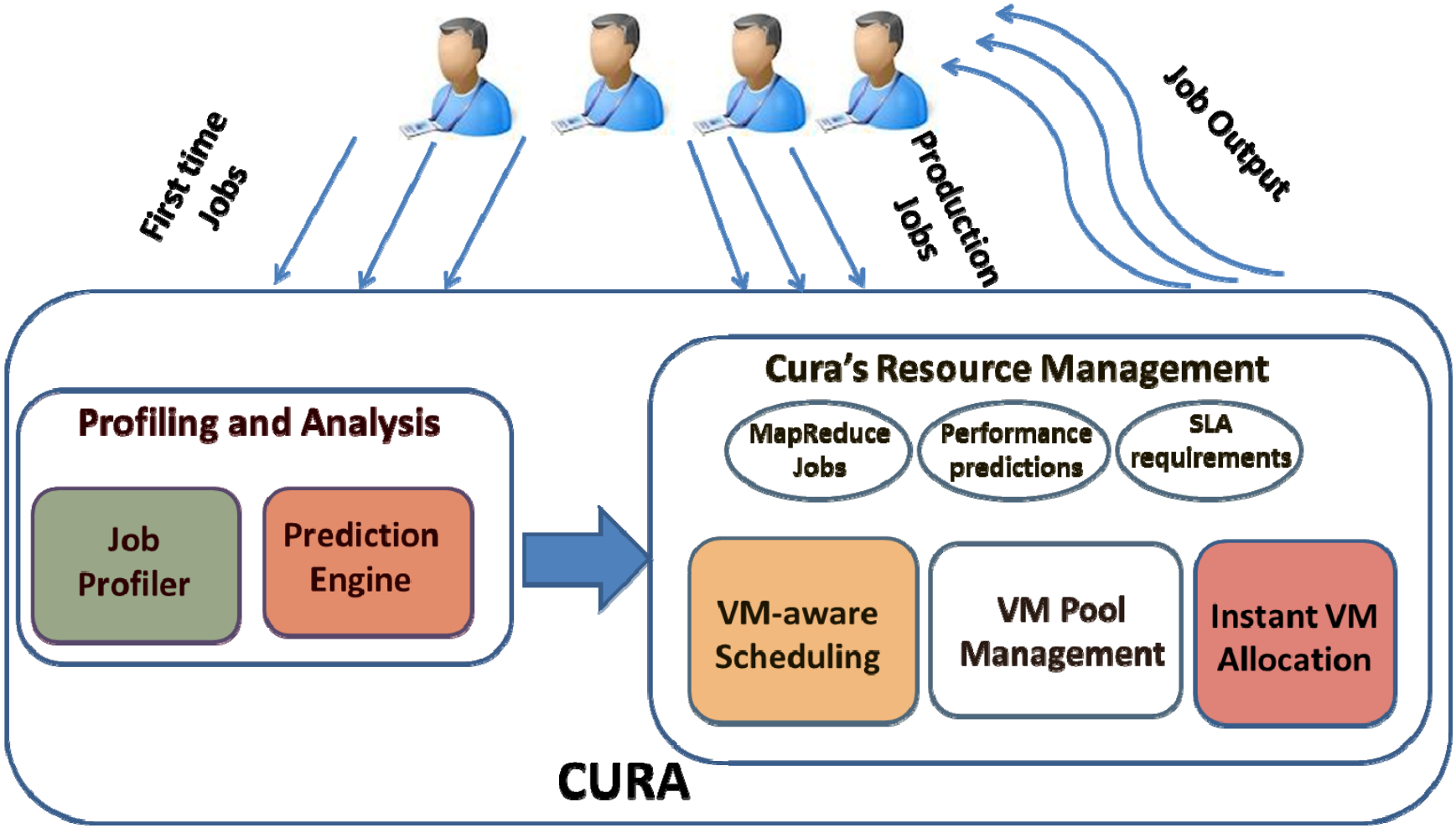
80 concurrent VMs

Cloud Scheduling

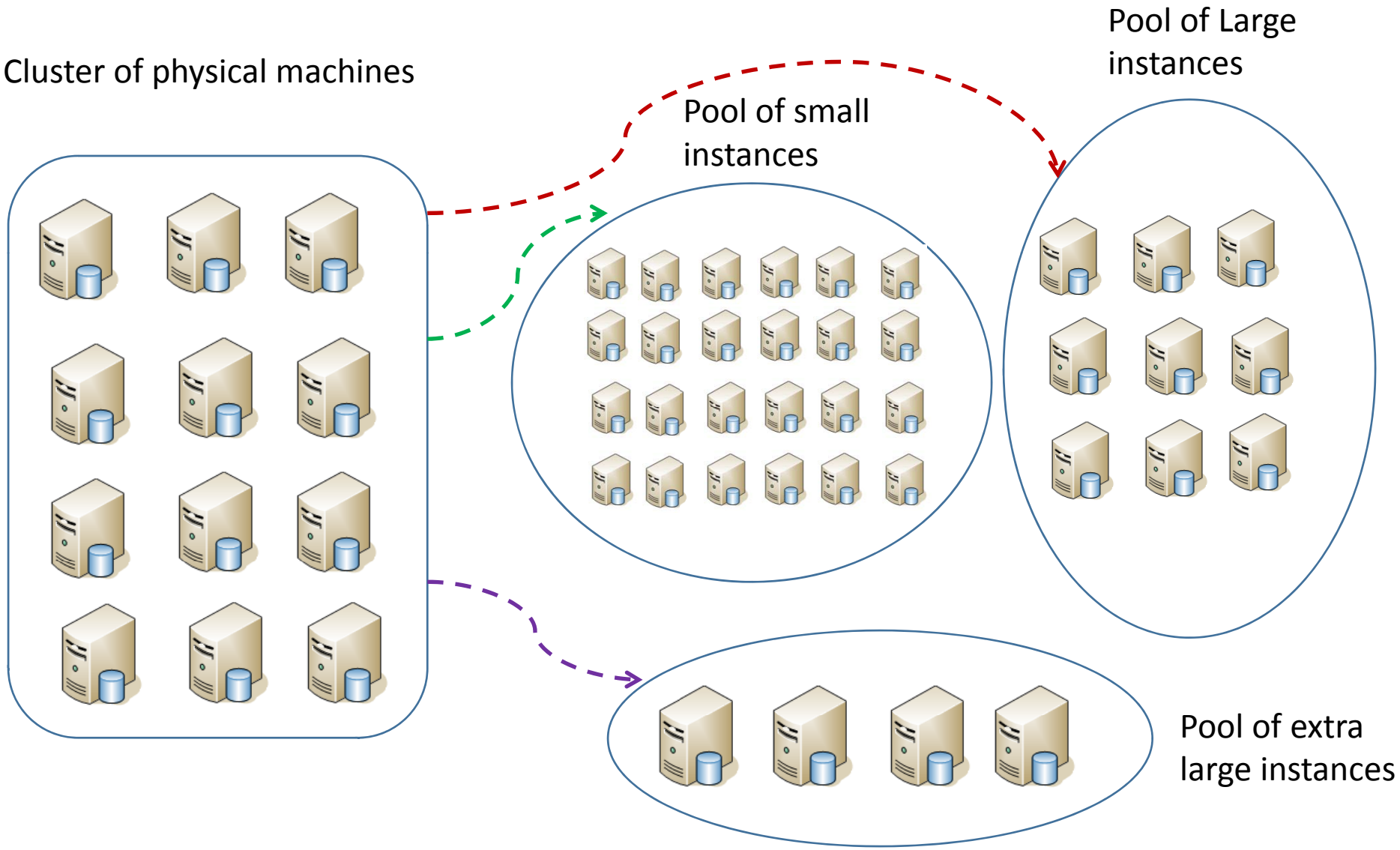
Job #	1	2	3	4
Start time	20	25	40	45

40 concurrent VMs

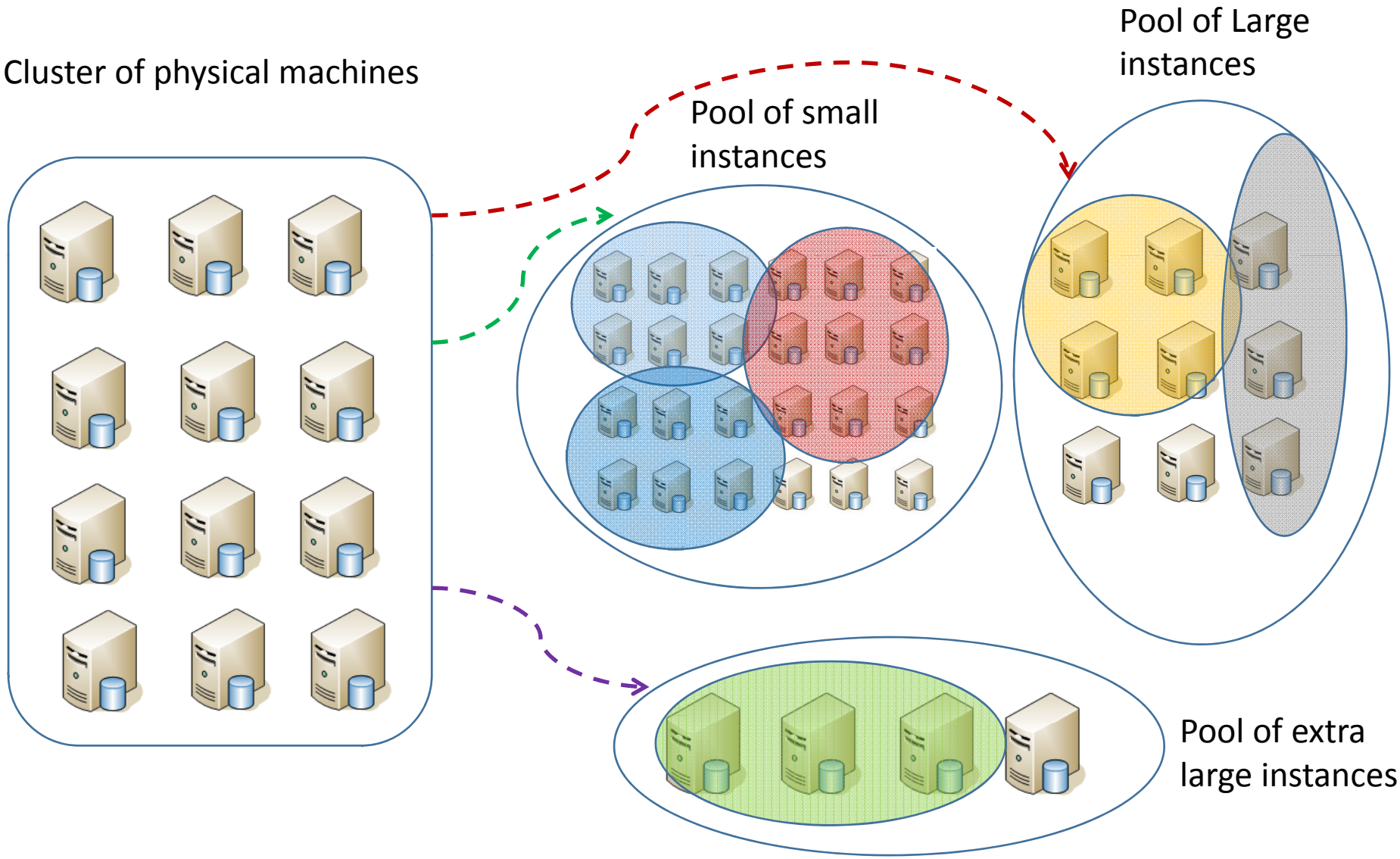
Cura System Architecture



Static Virtual Machine sets



Static Partitioning of Virtual Machine sets



Key Challenges in Cura Design

Resource Provisioning and Scheduling

- Optimal scheduling
- Optimal Cluster Configuration
- Optimal Hadoop Configuration

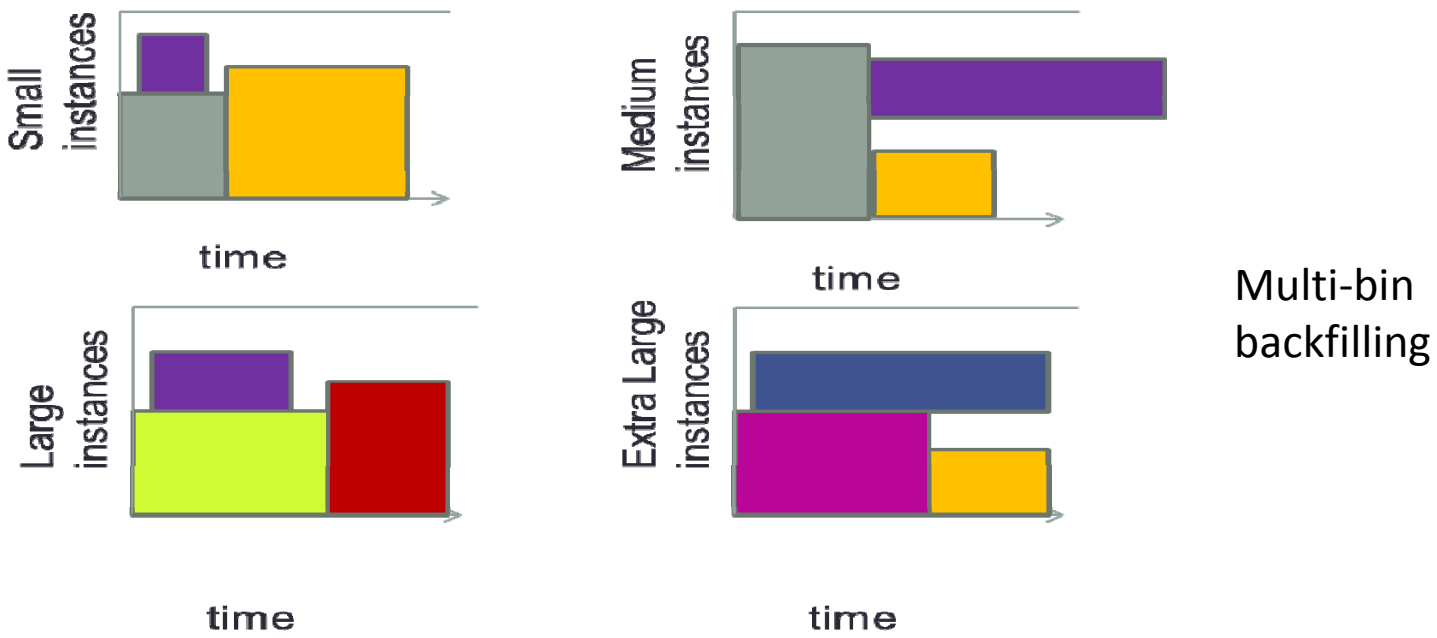
Virtual Machine Management

- Optimal capacity planning
- Right set of VMs(VM types) for current workload?
- Minimize Capital expenditure and Operational expenses

Resource Pricing

- What is the price of each job based on its service quality and job characteristics?

VM-aware Job Scheduling



Multi-bin backfilling

- Scheduler needs to decide on which instance type to use for the jobs

- The job scheduler has two major goals:
  - (i) complete all job execution within the deadlines
  - (ii) minimize operating expense by minimizing resource usage

VM-aware Scheduling Algorithm

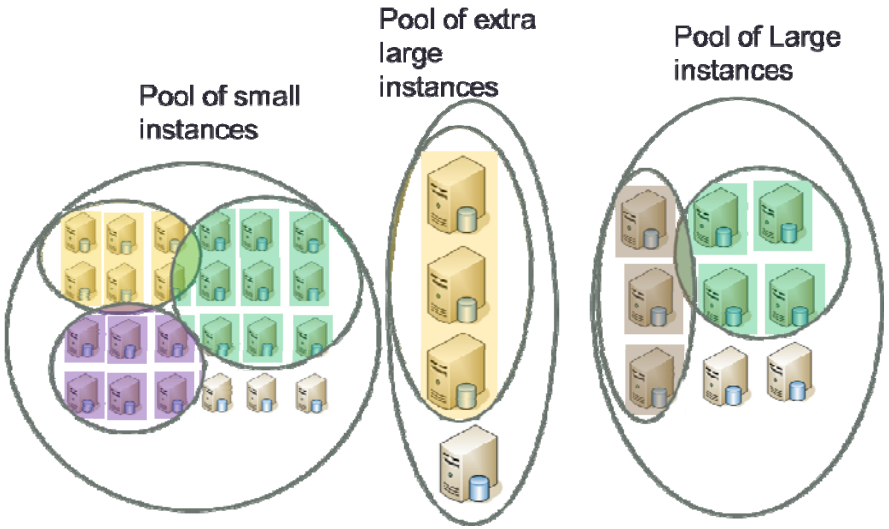
- Goal:
  - VM-aware scheduler decides (a) when to schedule each job in the job queue, (b) which VM instance pool to use and (c) how many VMs to use for the jobs.
- Minimum reservations without under-utilizing any resources.
- Job J_i has higher priority over Job J_j if the cost of scheduling J_i is higher.
- For each VM pool picks the highest priority job, J_{prior} in the job queue and makes a reservation.
- Subsequently, the scheduler picks the next highest priority jobs in the job queue by considering priority only with respect to the reservations that are possible within the current reservation time windows of the VM pools.
- Runs in $O(n^2)$ time
- Straight forward to obtain a distributed implementation to scale further

Reconfiguration-aware Scheduler

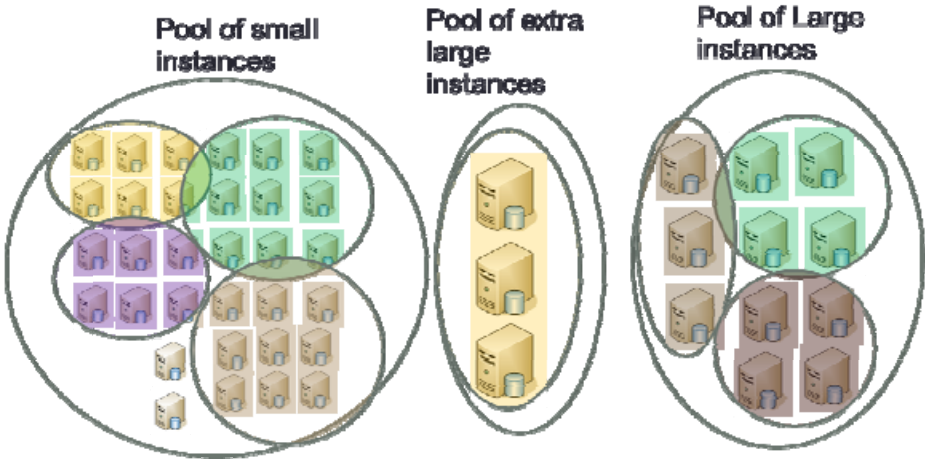
Assume two new jobs need a cluster of 9 small instances and 4 large instances respectively.

The scheduler has the following options:

1) Wait for some other clusters of small instances to complete execution
2) Run the job in a cluster available of extra large instances
3) Convert some large or extra large instances into multiple small instances



Reconfiguration-unaware Scheduler



Reconfiguration-aware Scheduler

Reconfiguration Algorithm

- Reconfiguration time window- Observation period for collecting history of job executions.
- For each job, J_i, the optimal cost $C_{opt}(J_i)$ that incurs the lowest resource usage is found.
- The reconfiguration plan reflects the proportion of demands for the optimal instance types.
- Reconfiguration Benefit:

$$Overallcost_{observed} = \sum_{i,k,n} Cost(J_i, C^{k,n}) \times Z_i{}^{k,n}$$

$$Overallcost_{estimate} = \sum_{i} Cost(J_i, C_{opt}(J_i))$$

$$Reconf_{benefit} = Overallcost_{estimate} - Overallcost_{actual}$$

Number of servers and Effective Utilization

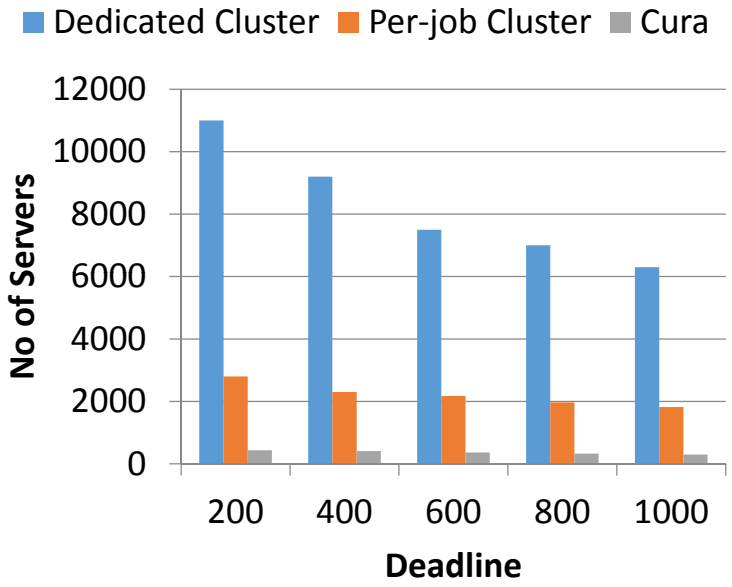


Fig 1. No. of Servers

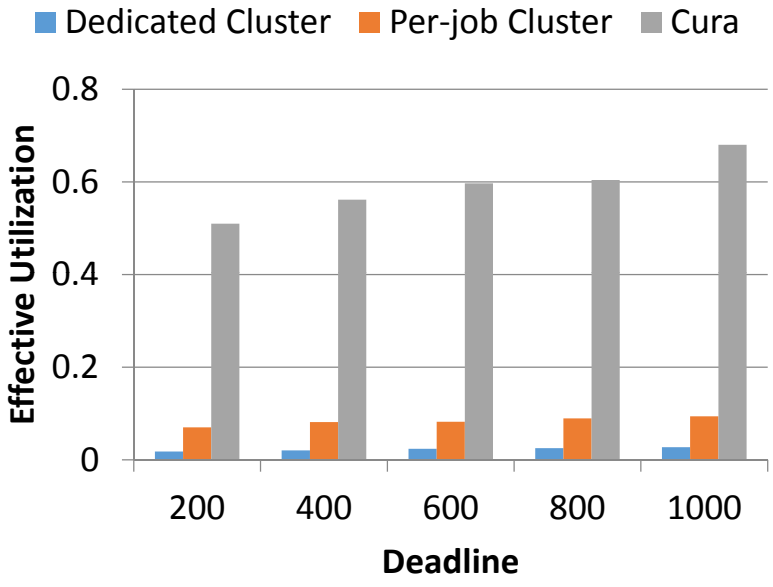


Fig 2. Effective Utilization

- Cura requires 80% lower resources than conventional cloud models
- Cura achieves significantly higher resource utilization

Talk Outline

- Cost-optimized Cloud Resource Allocation
  - Cura – Cost-optimized Model for MapReduce in a Cloud
- **Performance-driven Resource Optimization**
  - **Purlieus – Locality aware Resource Allocation for MapReduce**
- Privacy-conscious Processing of Cloud Data
  - VNCache: Efficient MapReduce Analysis for Cloud-archived Data
  - Airavat: Security and Privacy for MapReduce

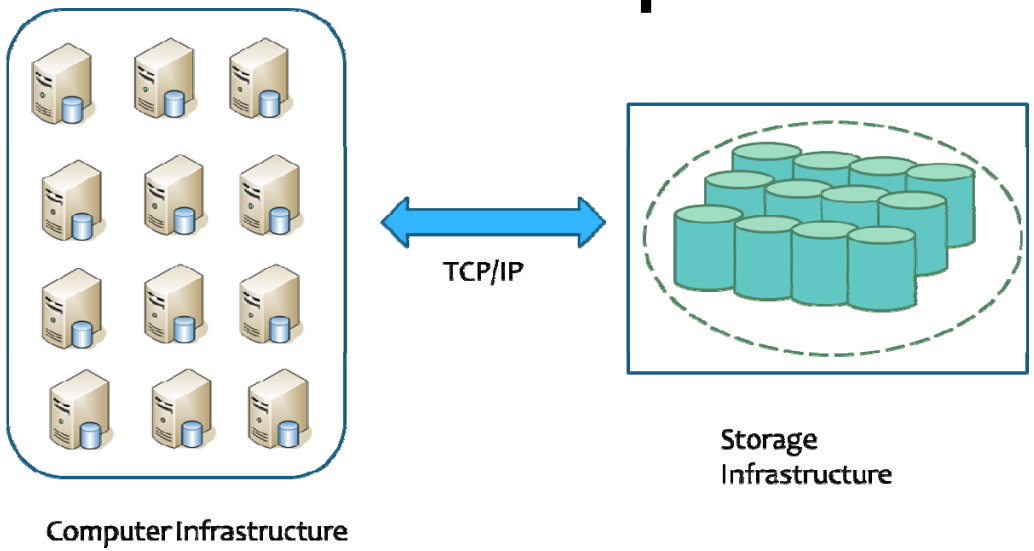
MapReduce cloud provider - Challenges

- Different loads on the shared physical infrastructure
  - Computation load: number and size of each VM (CPU, memory)
  - Storage load: amount of input, output and intermediate data
  - Network load: traffic generated during the map, shuffle and reduce phases

- The network load is of special concern with MapReduce
  - Large amounts of traffic can be generated in the shuffle phase

  - As each reduce task needs to read the output of *all* map tasks, a sudden explosion of network traffic can significantly deteriorate cloud performance

Existing Dedicated MapReduce Clouds



Amazon Elastic MapReduce Model

- Utilize a hosted Hadoop framework running on the Compute Cloud (e.g.: Amazon EC2) and use a separate storage service (e.g.: Amazon S3)

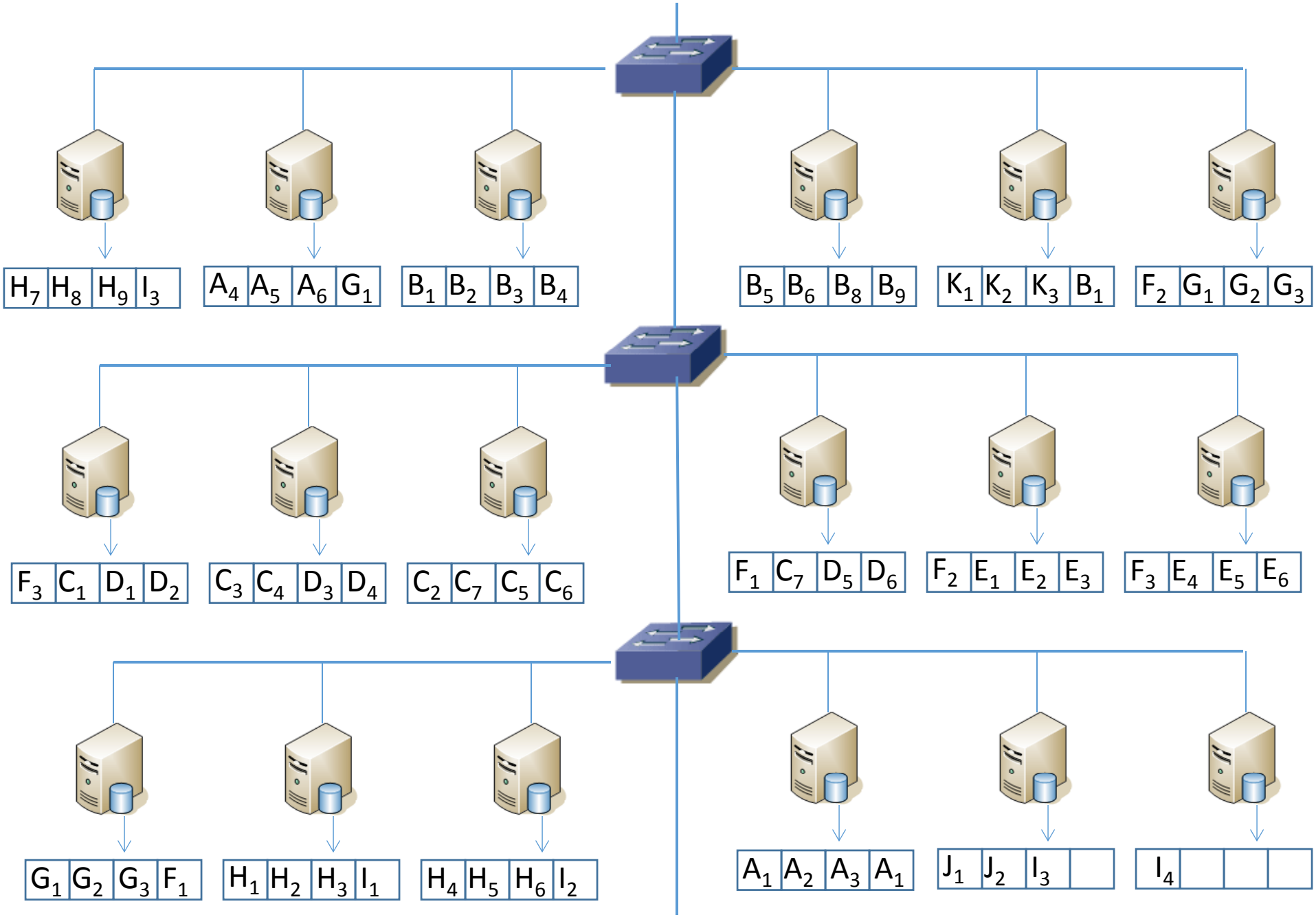
- Drawbacks
  - Adversely impacts performance as it violates data locality
  - Unnecessary replication leading to higher costs

  - *Processing 100 TB of data using 100 VMs takes 3 hours just to load the data*

Purlieus: Storage Architecture

- Semi-persistent storage architecture
- Data is broken up into chunks corresponding to MapReduce blocks
  - stored on a distributed file system of the *physical* machines
- Ability to transition data stored on physical machines in a seamless manner
  - i.e. without requiring an explicit data-loading step
- The data on physical machines is seamlessly made available to VMs
  - Uses *loopback mounts and VM disk-attach*

Purlieus Architecture and Locality-optimization Goals



Job Specific Locality-awareness

- Job-specific locality cost

$$Cost(A, Di) = MCost(A, Di) + RCost(A, Di)$$

$$Mcost(A, Di) = \sum_{i<j<Qi} size(Bi_{,j}) \times dist(Snode(B_{i,j}), Cnode(B_{i,j}))$$

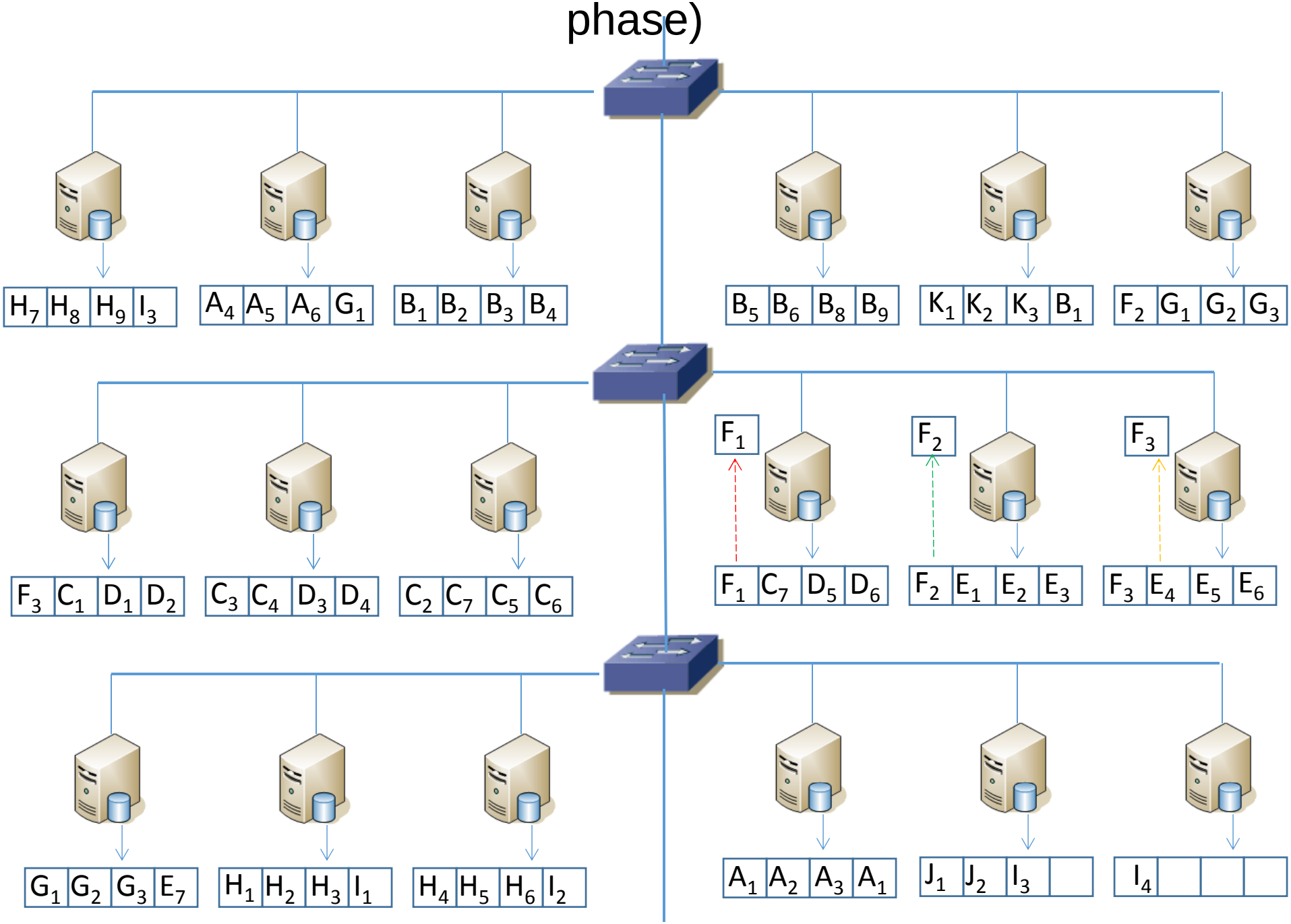
$$Rcost(A, Di) = \sum_{i<j<Qi, 1<l<L(A)} dist(Snode(B_{i,j}), Cnode(rtask_l)) \times m_{out}(A, (B_{i,j}), (rtask_l))$$

- Job categories
  - Map and Reduce input-heavy
    Sort workload : map-input size = map-output size
  - Map input-heavy
    Grep workload: large map-input and small map-output
  - Reduce input-heavy
    Synthetic Dataset Generators: small map-input and large map-output

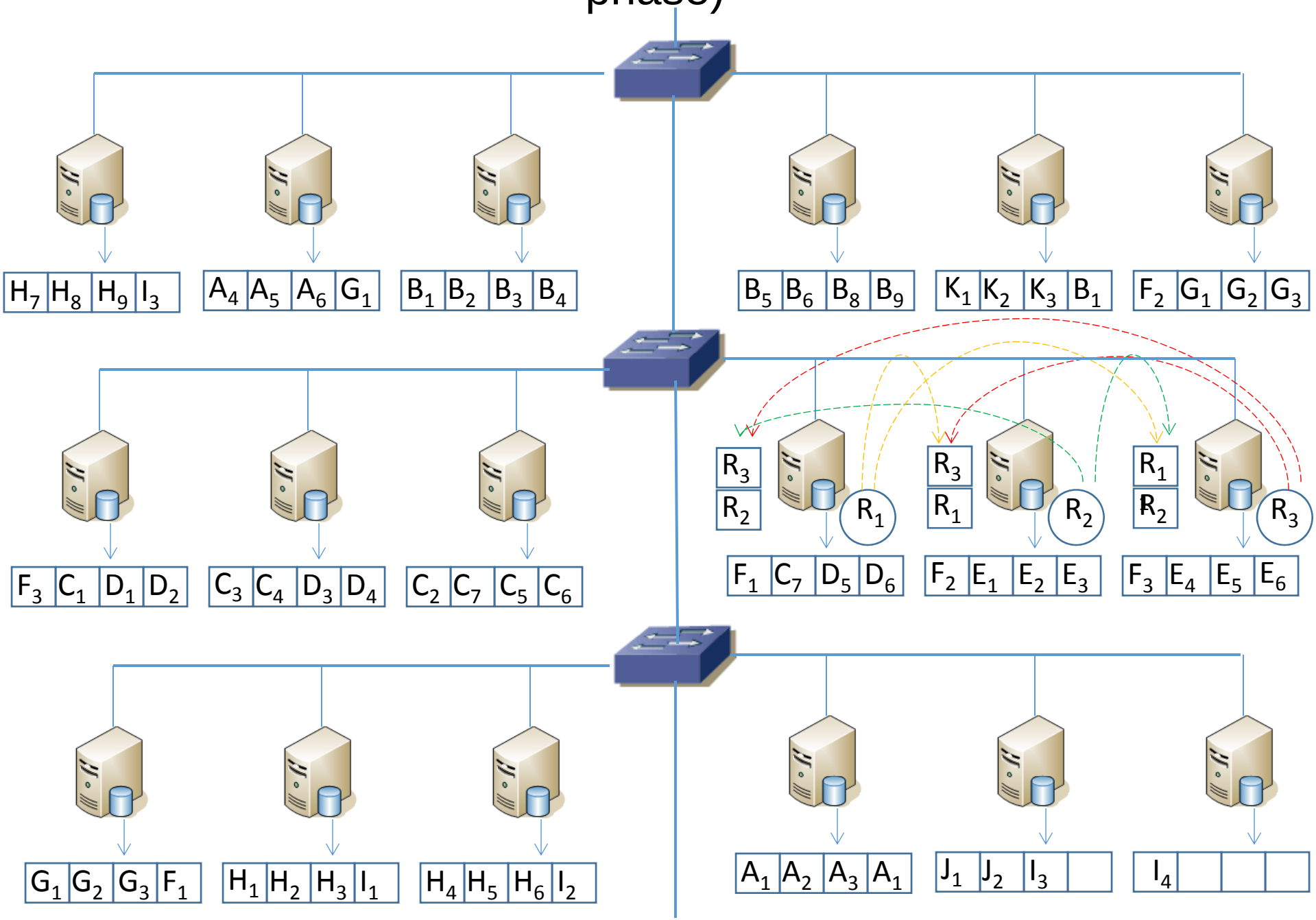
Map and Reduce input-heavy jobs

- Example: sorting a huge data set (output size = input size)
- Both map and reduce localities are important.
- Map phase:
  - Incorporates map locality by pushing compute towards data
- Reduce phase:
  - Make communications happen within a set of closely connected nodes

Scheduling Map and Reduce-input heavy tasks (map-phase)



Scheduling Map and Reduce-input heavy tasks (reduce-phase)



Experimental Setup

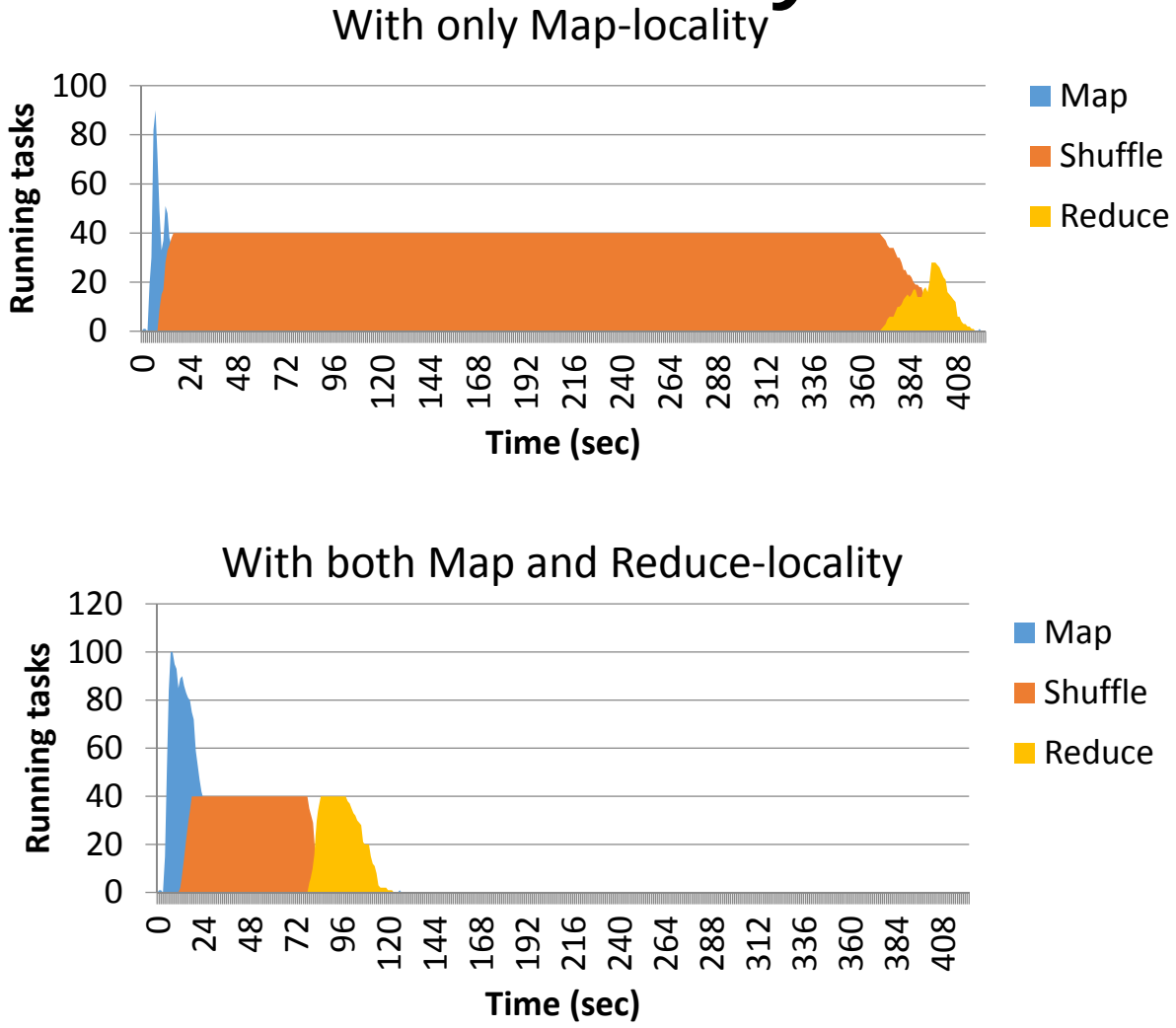
- Cluster Setup
  - Testbed of 20 CentOS 5.5 physical machines (KVM as the hypervisor)
  - Each physical machine has 16 core 2.53 GHz Intel Processors.
  - The network is 1 Gbps.
  - Nodes are organized in 2 racks, each containing 10 physical machines

Job types:

Workload Type	Job	Input data	Output data
Map-input heavy	Grep: word Search	20 GB	2.43 MB
Reduce-input heavy	Permutation Generator	2 GB	20 GB
Map and Reduce-input heavy	Sort	10 GB	10 GB

By default, each job uses 20 VMs with each VM configured with 4 GB memory and 4 2GHz vCPUs.

mpact of Reduce-locality



Timeline plotted using Hadoop *job_history_summary*

Map and Reduce-input heavy workload

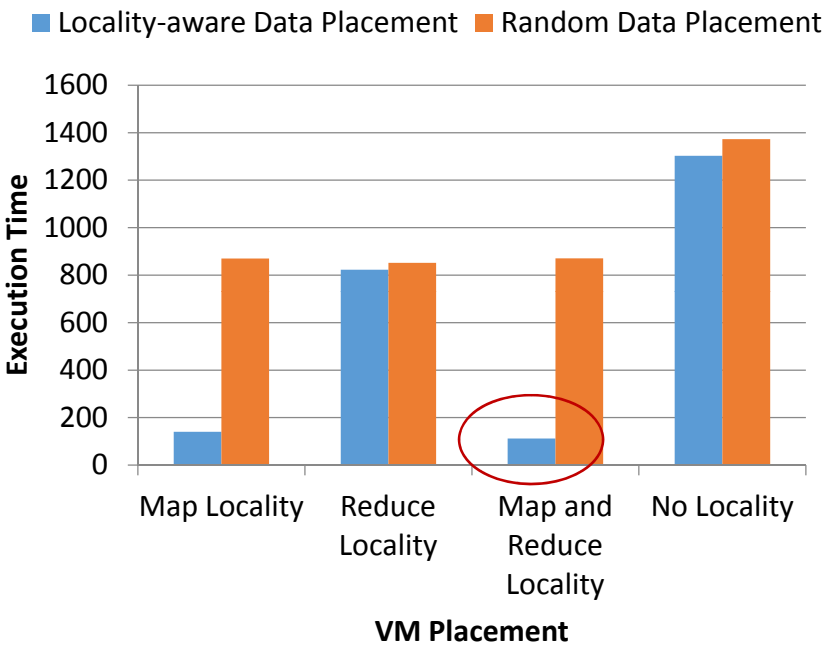


Fig 3. Execution time

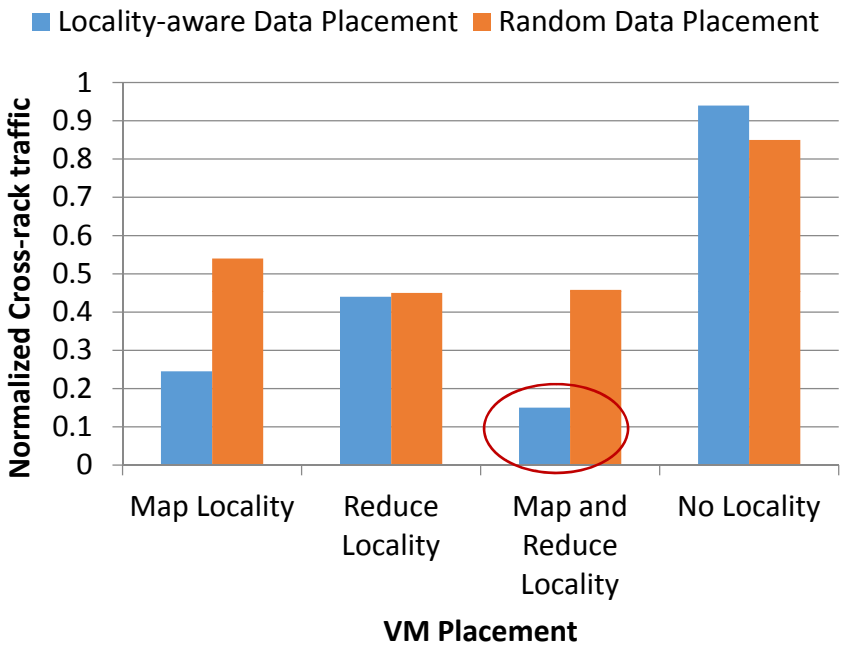


Fig 4. Cross-rack traffic

Performance depends on both Map and Reduce locality !

Locality depends not only on compute placement but also on data placement

Talk Outline

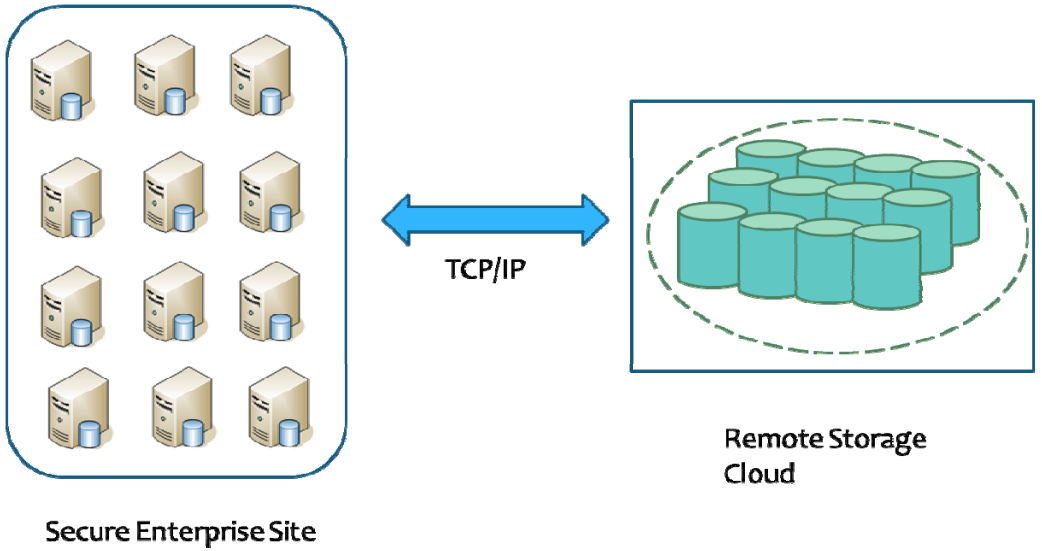
- Cost-optimized Cloud Resource Allocation
  - Cura – Cost-optimized Model for MapReduce in a Cloud
- Performance-driven Resource Optimization
  - Purlieus – Locality aware Resource Allocation for MapReduce
- **Privacy-conscious Processing of Cloud Data**
  - **VNCache: Efficient MapReduce Analysis for Cloud-archived Data**
  - Airavat: Security and Privacy for MapReduce

MapReduce Analysis for Cloud-archived Data

- Compute Cloud and Storage Cloud can not be collocated always
  - E.g. When there is a privacy/security concern
- Example use case: private log data archived in Clouds
  - Logs can contain sensitive information
  - require data to be encrypted before moved to the cloud.
- Current solutions
  - require the data to be transferred and processed in a secure enterprise cluster.

Current analytics solutions for Cloud-archived Data

- MapReduce analysis for Cloud-archived data
  - Current analytics platforms at secure enterprise site(e.g. Hadoop/MapReduce)
    - first download data from these public clouds
    - decrypt it and then process it at the secure enterprise site

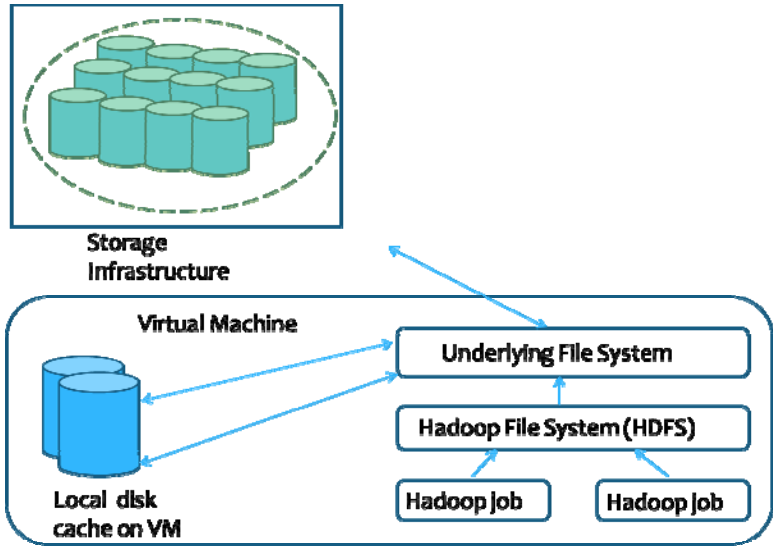


- Data needs to be loaded before the jobs can even start processing
- Duplicates data at both enterprise site and Storage clouds.

Current analytics solutions for Cloud-archived Data

- Wan copy time, decrypt time and HDFS load time are predominant factors of job execution time
- Actual job execution is significantly slowed down because of this overhead

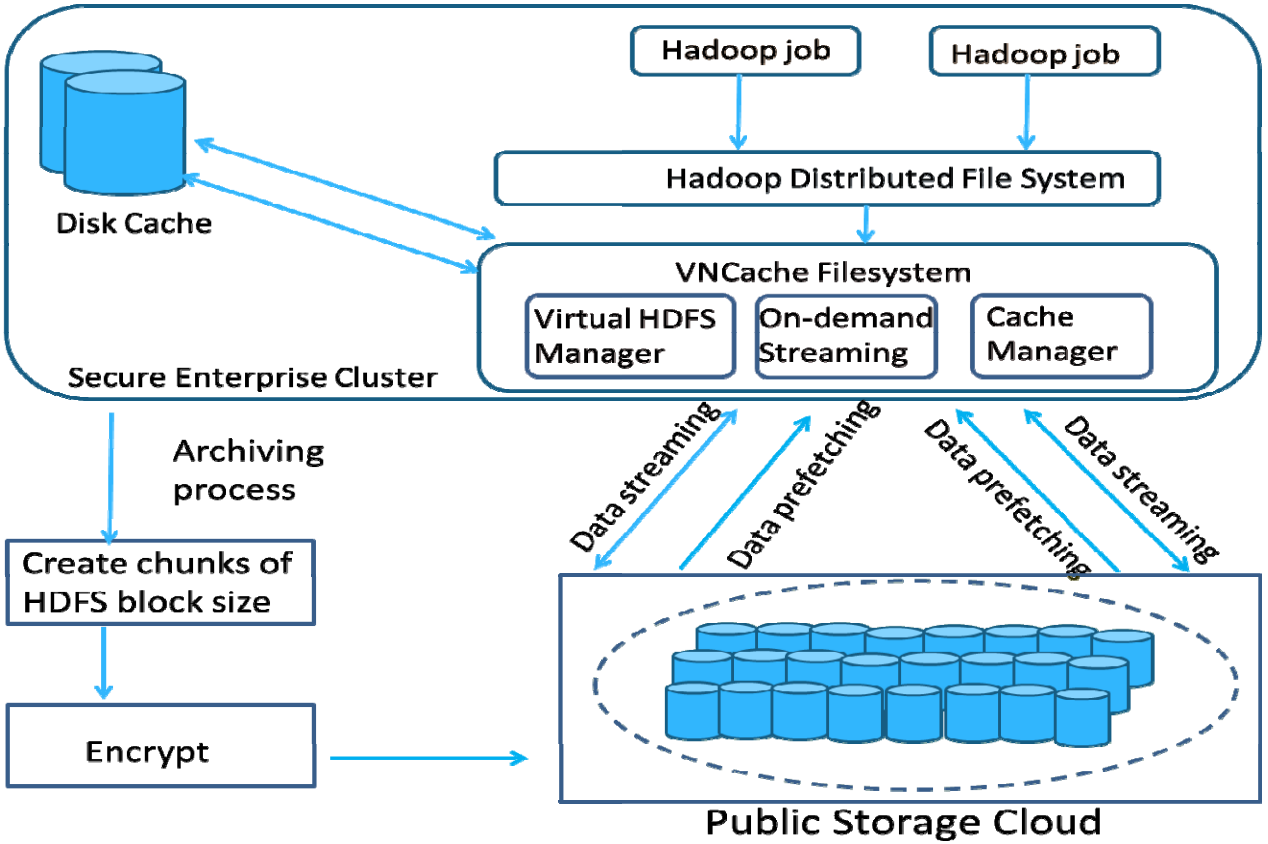
VNCache: Seamless Data Caching



- VNCache – Key idea
  - to use a virtual HDFS Filesystem image that provides virtual input data files on the target cluster.
  - to provide a virtual existence of the Hadoop datablocks to the HDFS
  - enables jobs to begin execution immediately

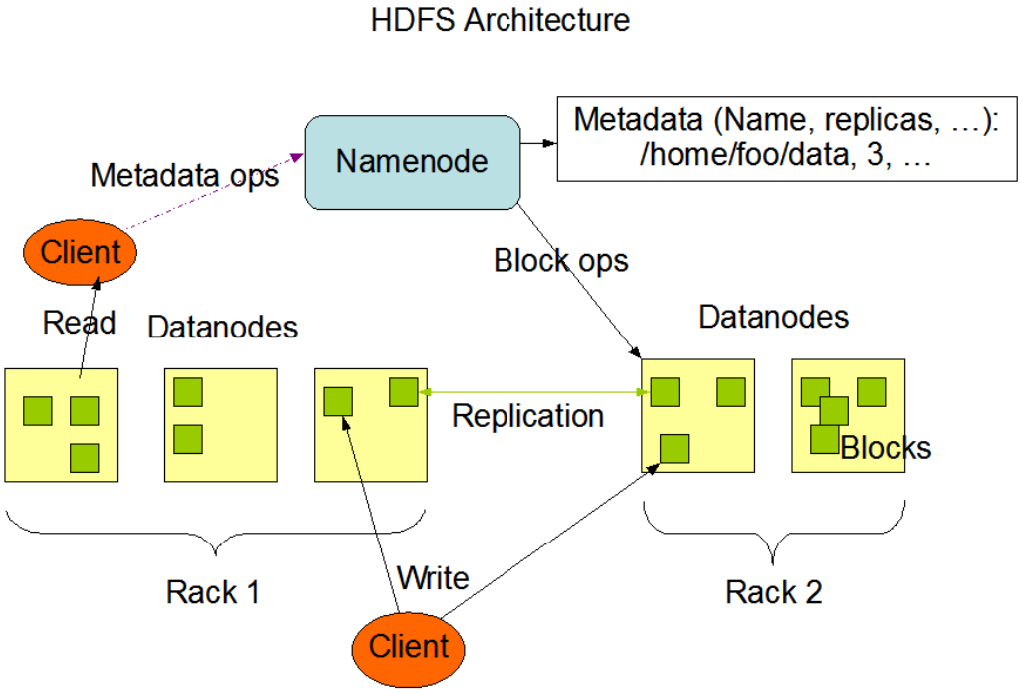
- For HDFS filesystem running on top of the VNCache
  - All data appears to be local though initially the data is at the storage cloud

VNCache: Data flow



- Data is archived at the public storage cloud as chunks of HDFS block size
- For job analysis, VNCache seamlessly streams and prefetches data from the storage cloud

Overview of HDFS



- HDFS - primary storage by Hadoop applications
- The HDFS Namenode
  - manages the file system namespace and regulates access to files by clients
- The individual Datanodes
  - Manage the storage attached to the nodes that they run on.

nteraction of HDFS with its underlying Filesystem

Namenode at the master node

- stores the HDFS filesystem, *fsimage*

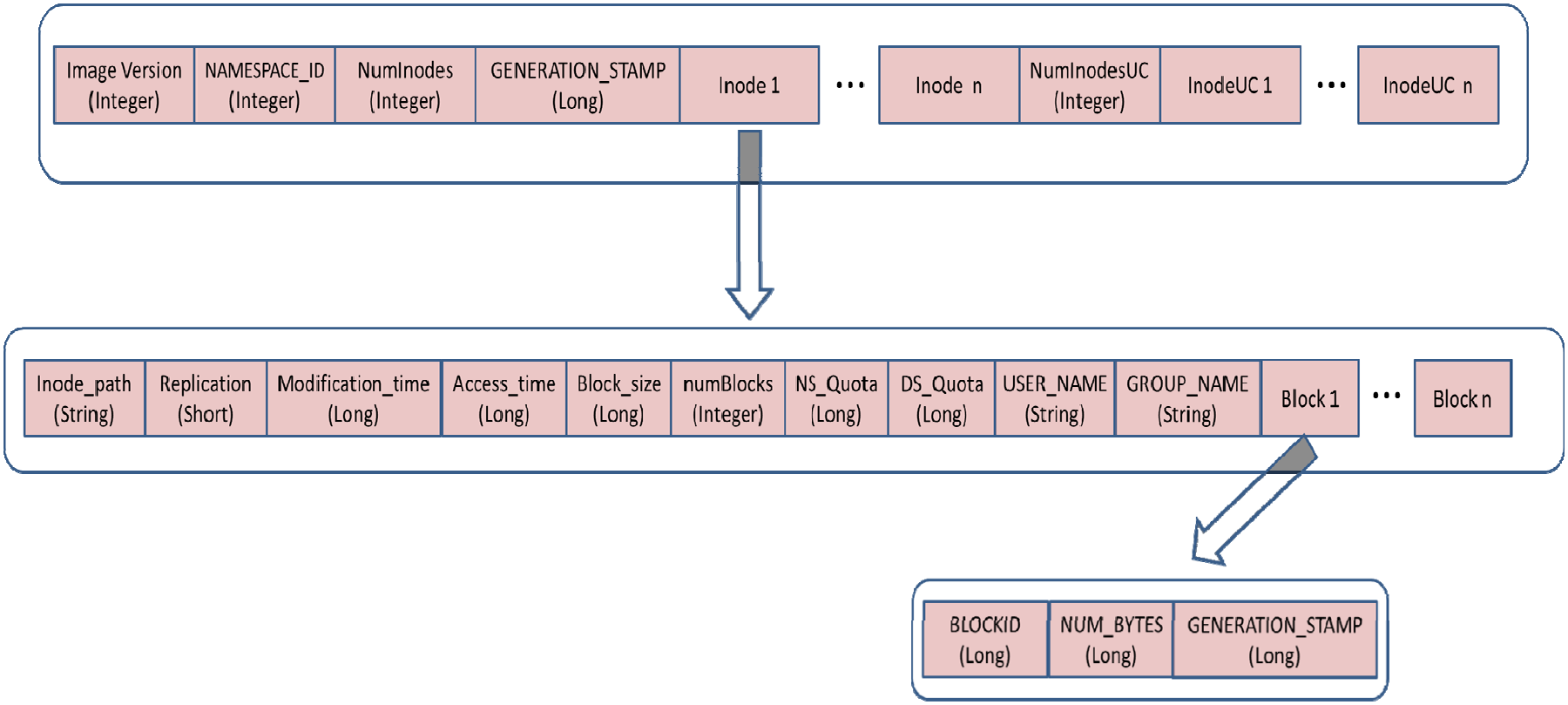
HDFS filesystem namespace

- includes the mapping of HDFS files to their constituent blocks
- Block identifiers
- Modification and Access times

Datanode

- stores a set of HDFS blocks
- uses separate files in the underlying filesystem
- both metadata and actual blocks

HDFS Filesystem Image layout



- Consists of an Inode structure for each HDFS file
- Each Inode has the information for the individual data blocks corresponding to the file

HDFS Virtualization

VNCache Filesystem

- FUSE based filesystem used as the *underlying* filesystem on the Namenode and Datanode.
- It is a virtual filesystem (similar to */proc* on Linux)
- simulates various files and directories to the HDFS layer placed on it

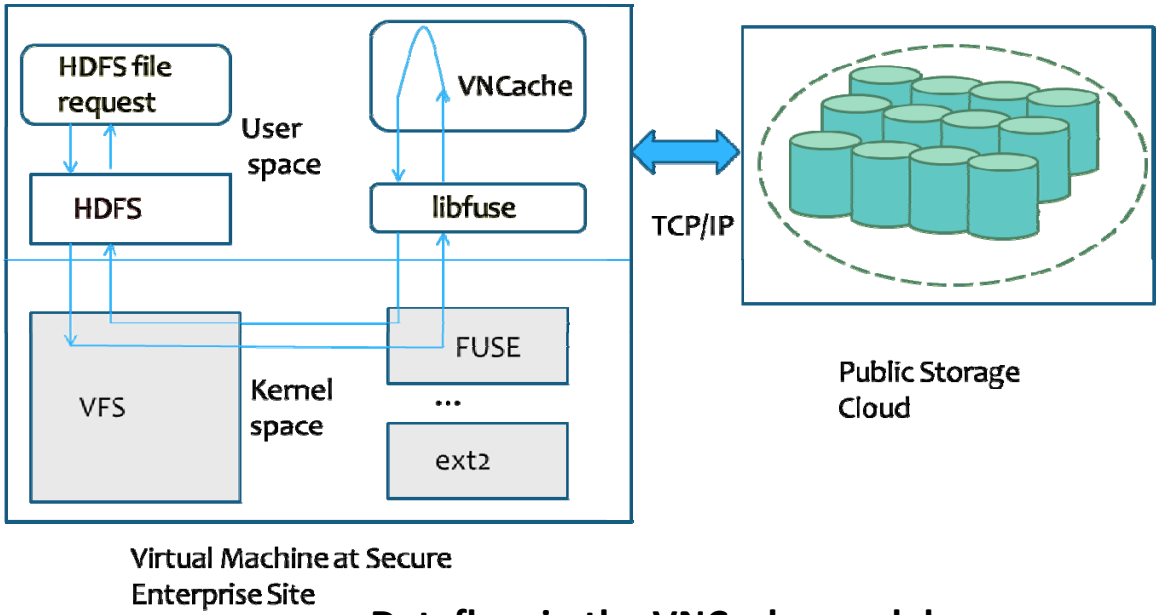
Namenode Filesystem (*fsimage)* virtualization

- a virtual HDFS namespace is created with a list of relevant HDFS blocks

Datanode virtualization

- exposes a list of data files corresponding to the HDFS blocks placed on that datanode

VNCache's Dataflow



Dataflow in the VNCache model

Two models of operation: Streaming and Prefetching

- Seamless Streaming:
  - For data block accesses on demand, the VNCache Filesystem streams the data seamlessly from the storage cloud.

- Cache Prefetching:
  - Cache-prefetching obtains the block ahead of processing and significantly reduces job execution time.

Cache Prefetching

- Cache Prefetching Logic
  - Analyzes job input data and prefetches data blocks
  - Carefully considers the order of task processing in the input data
- Prefetching order
  - task ordering based on decreasing order of file sizes in the job input
  - based on the order of blocks in the HDFS file system image.
- Distributed Cache Prefetching algorithm
  - Master Prefetcher
    - Makes prefetching decisions and delegates prefetch tasks to slave prefetchers
  - Slave Prefetchers
    - transfer the individual data blocks from the remote cloud

Caching Algorithm

Rate-adaptiveness

- monitors the progress of the Hadoop jobs in terms of task processing rate and the HDFS bloc request rate
- adaptively decides the set of data blocks to prefetch

Cache Eviction Policy

- enforces minimal caching principle.
- Keeps only the most immediately required blocks in the cache
- minimizes the total storage footprint

Workflow-awareness

- Workflows may have multiple jobs processing the same input dataset
- VNCache recognizes it and prefetches those blocks only once
- uses a persistent workflow-aware caching

Experimental Setup

- Cluster Setup
  - Testbed of 20 CentOS 5.5 physical machines (KVM as the hypervisor)
  - Each physical machine has 16 core 2.53 GHz Intel Processors.
  - 5 nodes used as remote Storage cloud servers
- Individual job Workloads:
  - Grep, Sort, Facebook jobs using Swim workload generator
- Workflow-based Workloads:
  - Facebook workflows, Tfidf
- Dataset sizes:
  - 5 GB, 10 GB and 15 GB
- Network Latencies:
  - 45 ms, 90 ms, and 150 ms
- By default, each job uses 5 VMs with each VM configured with 4 GB memory and 3.5GHz vCPUs.

Performance of Grep workload

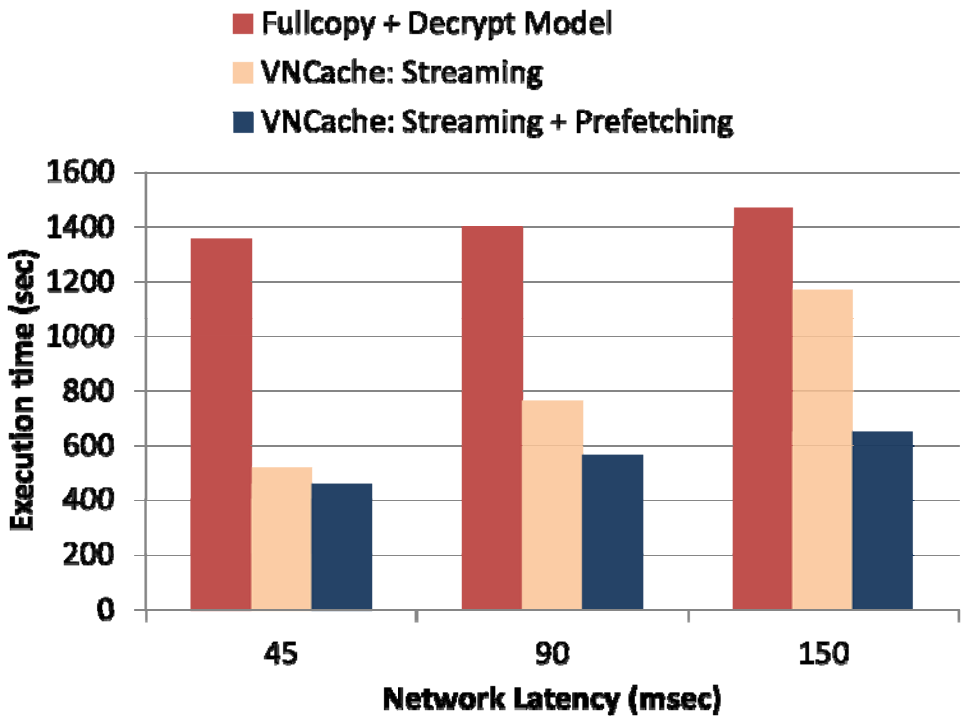


Fig 5. Execution time

VNCache:Streaming model achieves 42% reduction in execution time

VNCache Prefetch optimization further improves it by 30%

Performance of Facebook workload

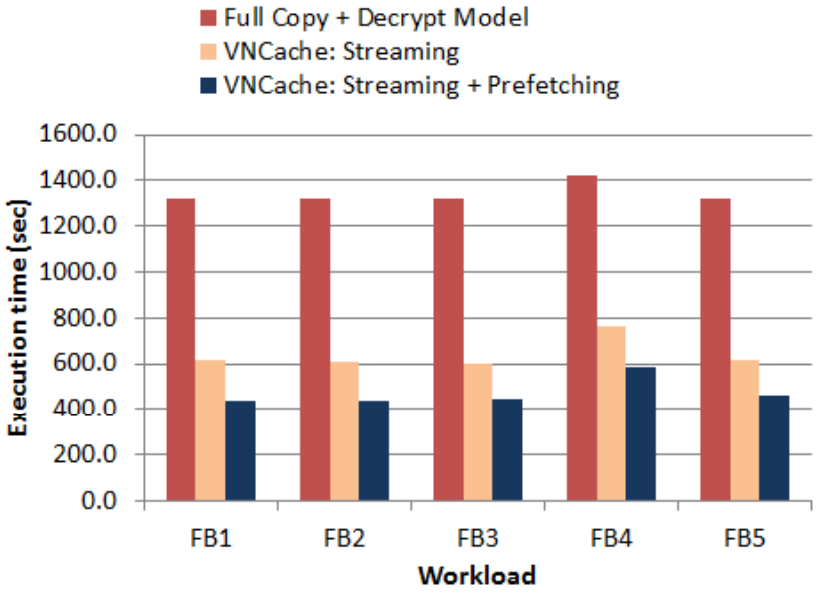


Fig 6. Execution time

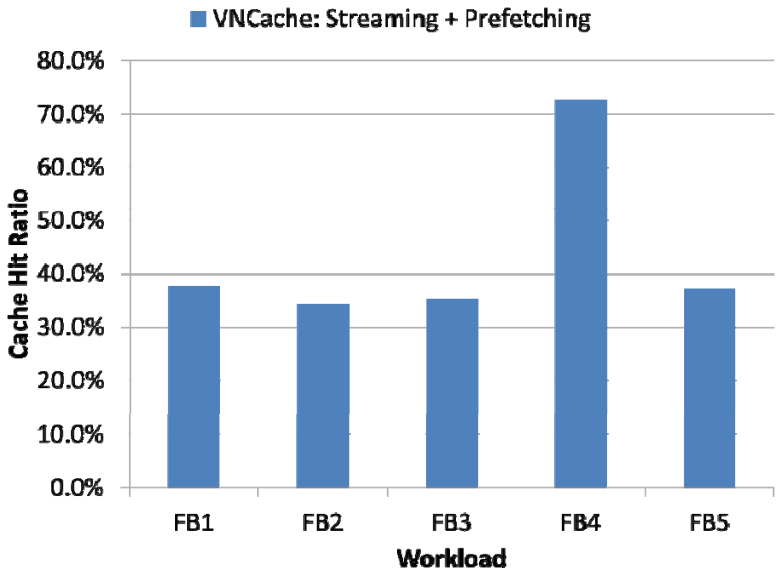


Fig 7. Cache Hit Ratio

VNCache gets more than 55% reduction in job execution time

Achieves an average cache hit ratio of 43.5%

Tfidf workflow

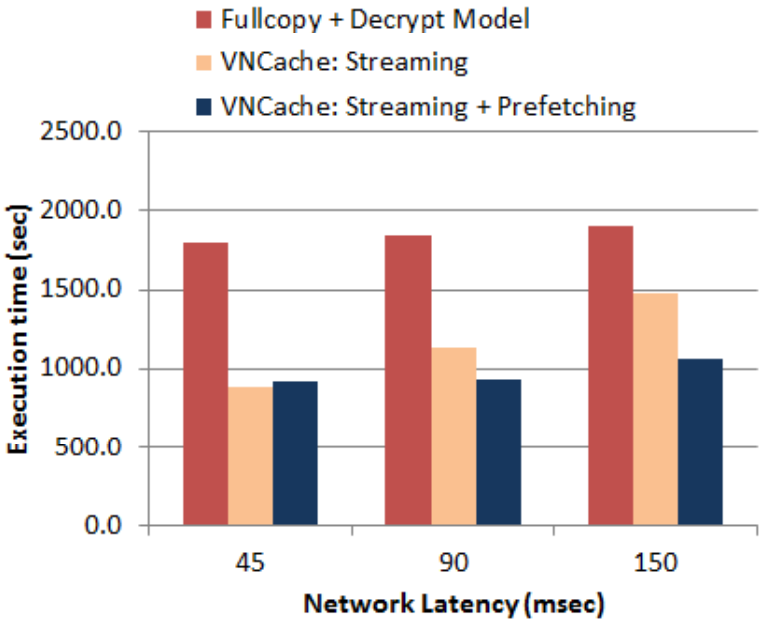


Fig 9. Execution time

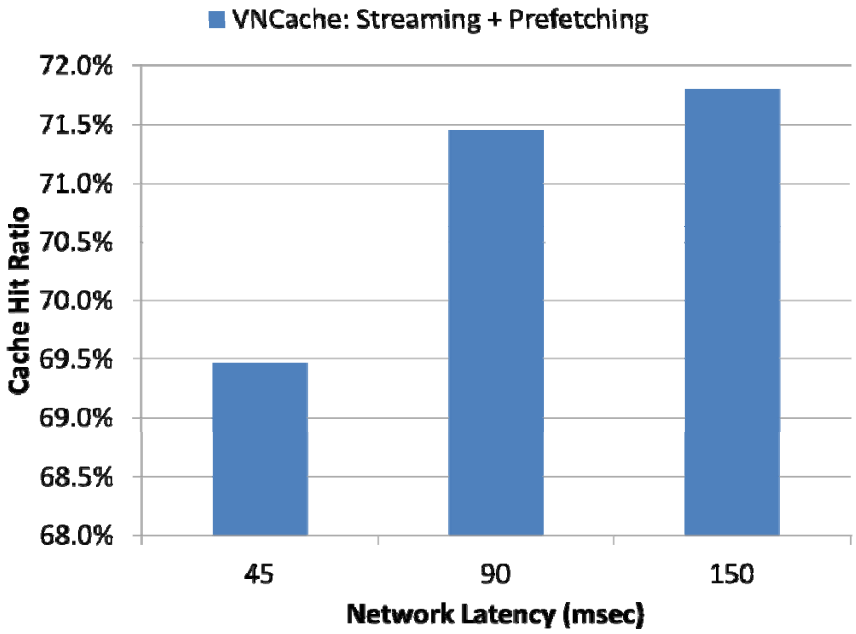


Fig 10. Cache Hit Ratio

VNCache reduces the execution time by 47% and results in 70% cache hits on average

mpact of Cache size – Grep Workload

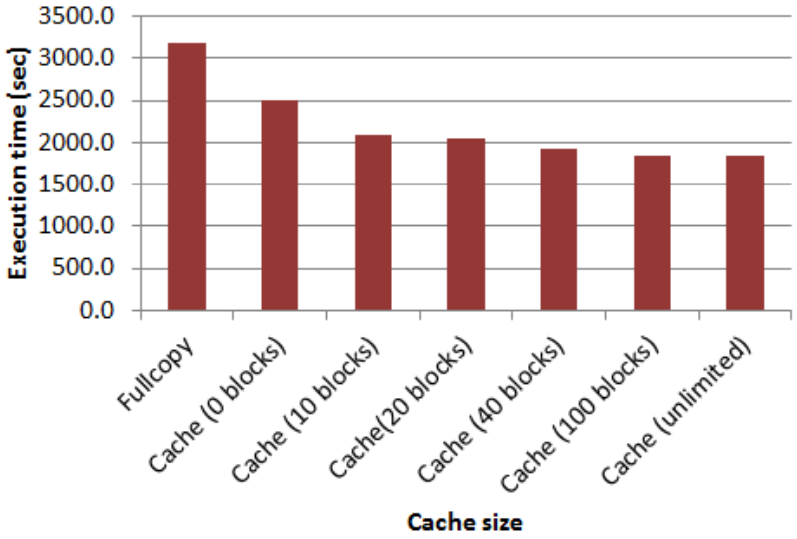


Fig 15. Execution time

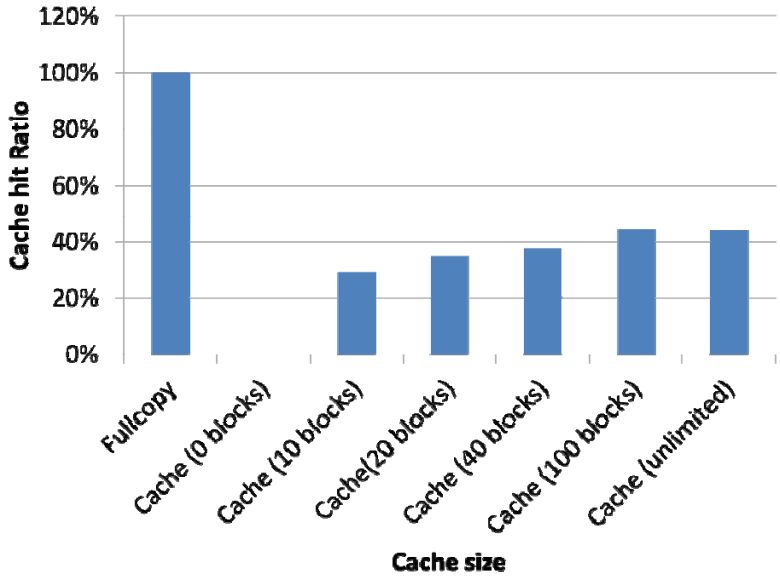


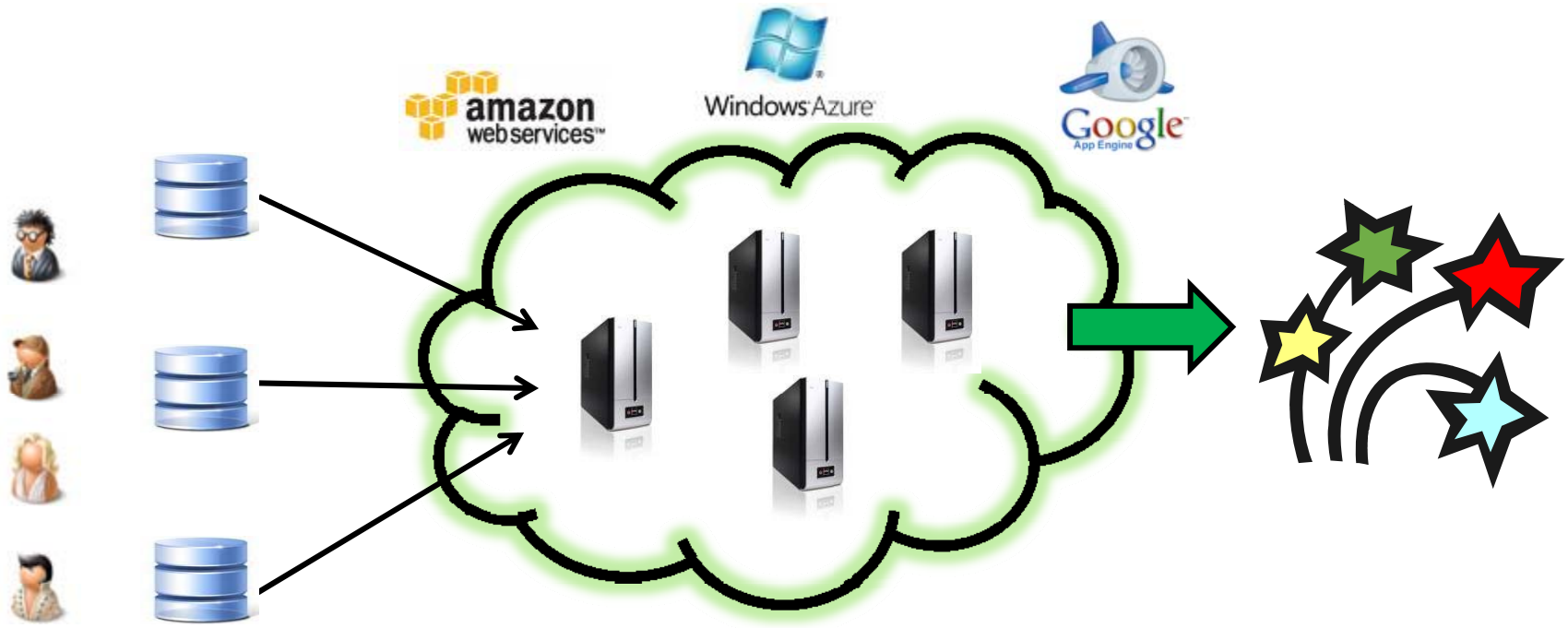
Fig 16. Cache Hit Ratio

Even with a reasonable cache size, VNCache achieves good performance

Talk Outline

- Cost-optimized Cloud Resource Allocation
  - Cura – Cost-optimized Model for MapReduce in a Cloud
- Performance-driven Resource Optimization
  - Purlieus – Locality aware Resource Allocation for MapReduce
- Privacy-conscious Processing of Cloud Data
  - VNCache: Efficient MapReduce Analysis for Cloud-archived Data
  - **Airavat: Security and Privacy for MapReduce**
    - **(slides adapted from Indrajit Roy et. al, NSDI 2010 paper)**

Computing in the year 201X



Data

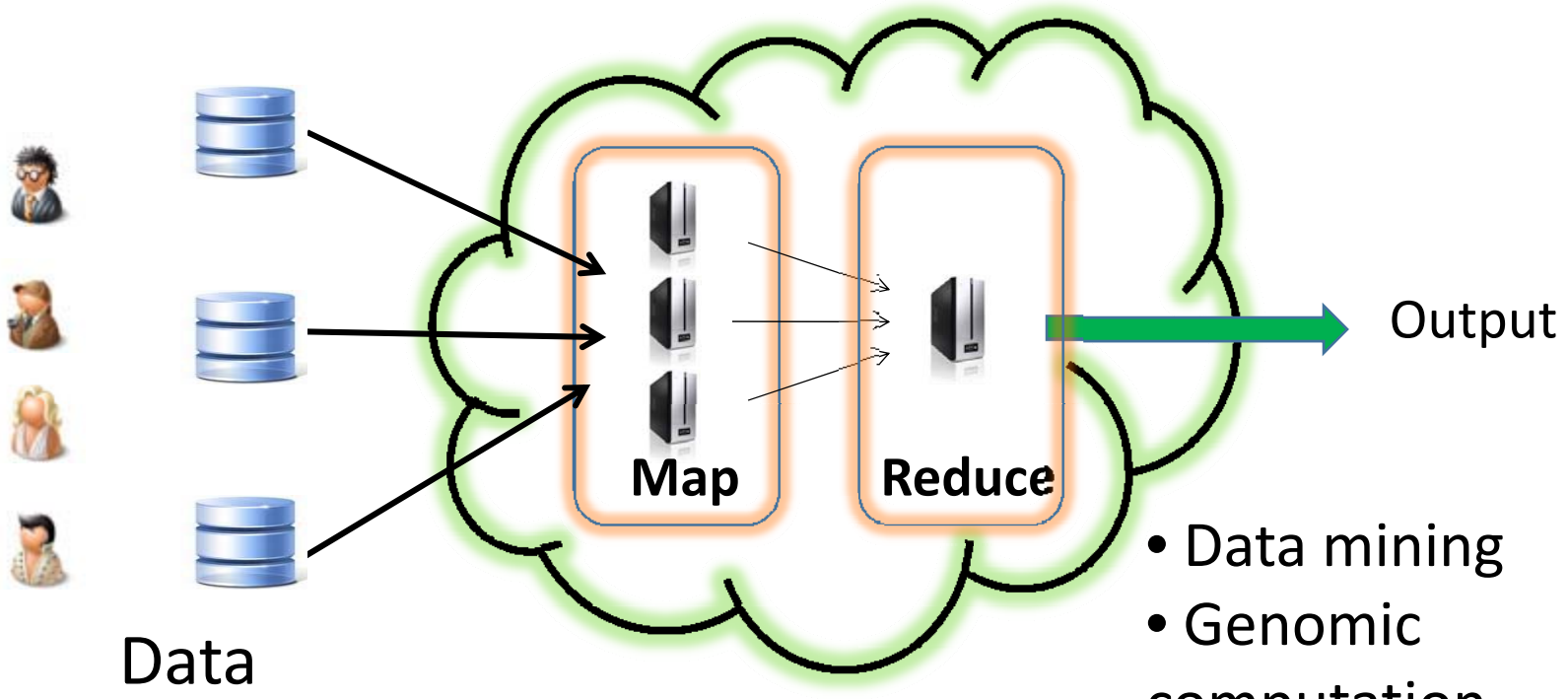
- Illusion of infinite resources
- Pay only for resources used
- Quickly scale up or scale down …

Indrajit Roy et. al, NSDI 2010

Programming model in year 201X

Frameworks available to ease cloud programming

MapReduce: Parallel processing on clusters of machines



Indrajit Roy et. al, NSDI 2010

- Data mining
- Genomic computation
- Social networks

Programming model in year 201X

Thousands of users upload their data

- Healthcare, shopping transactions, census, click stream

Multiple third parties mine the data for better service

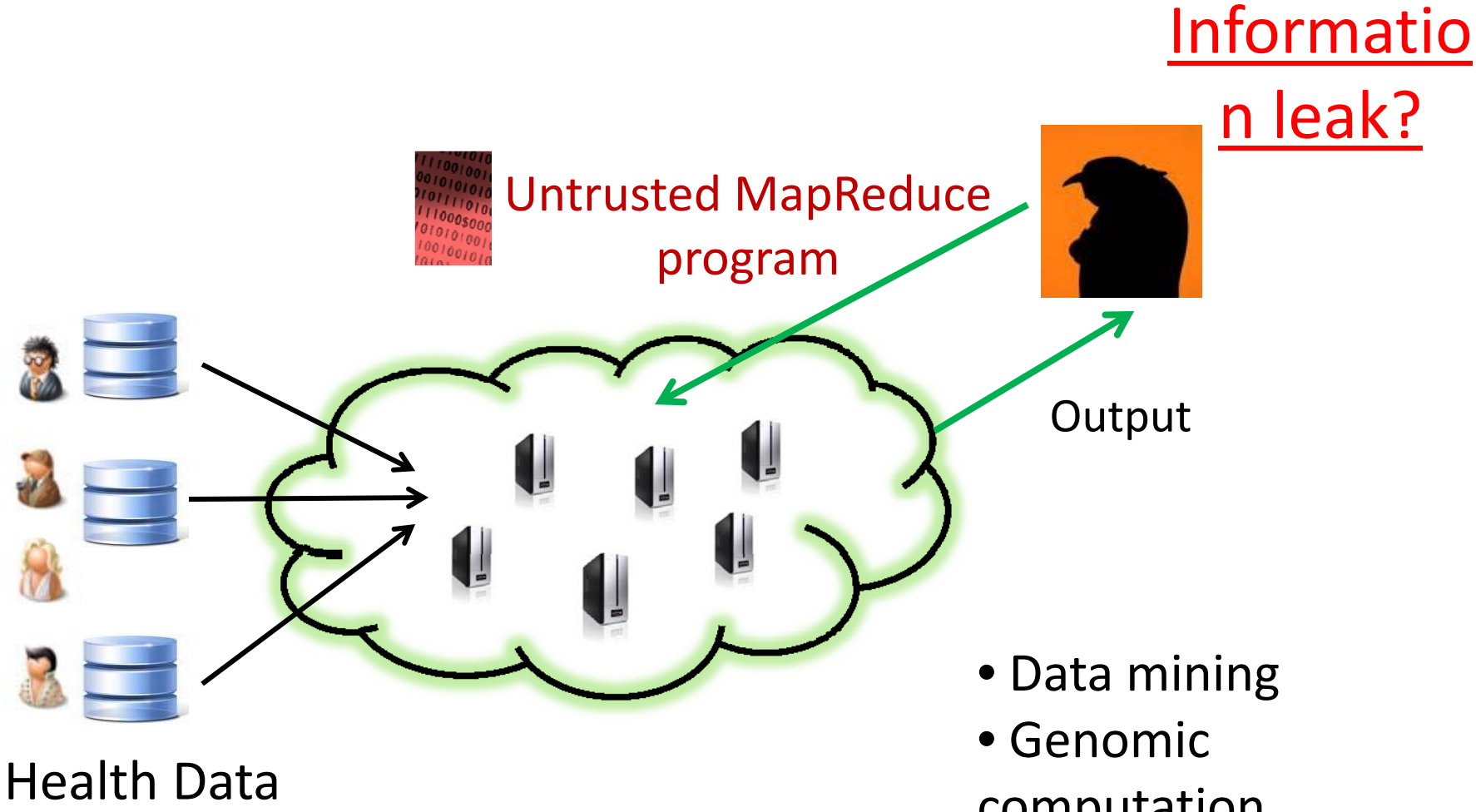
Example: Healthcare data

Incentive to contribute: Cheaper insurance policies, new drug research, inventory control in drugstores…

Fear: What if someone targets my personal data?

- Insurance company can find my illness and increase premium

Privacy in the year 201X ?



- Data mining
- Genomic computation
- Social networks

Indrajit Roy et. al, NSDI 2010

Use de-identification?

Achieves ‘privacy’ by syntactic transformations

- Scrubbing , k-anonymity …

Insecure against attackers with external information

- Privacy fiascoes: AOL search logs, Netflix dataset

Run untrusted code on the original data?

How do we ensure privacy of the users?

This talk: Airavat

Framework for privacy-preserving MapReduce computations with untrusted code.

Airavat is the elephant of the clouds (Indian mythology).

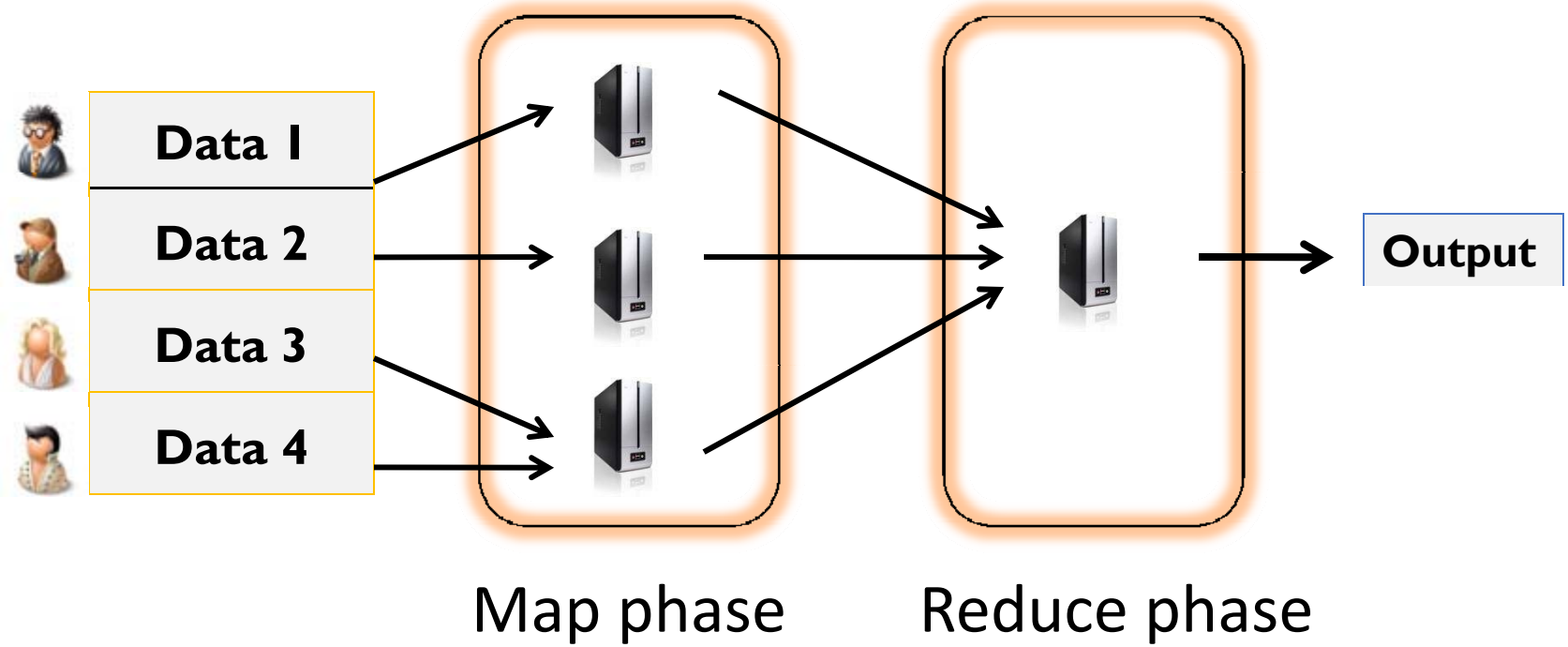
Airavat guarantee

Bounded information leak* about any individual data after performing a MapReduce computation.

*Differential privacy

Background: MapReduce

$map(k_1,v_1) \rightarrow list(k_2,v_2)$
$reduce(k_2, list(v_2)) \rightarrow list(v_2)$

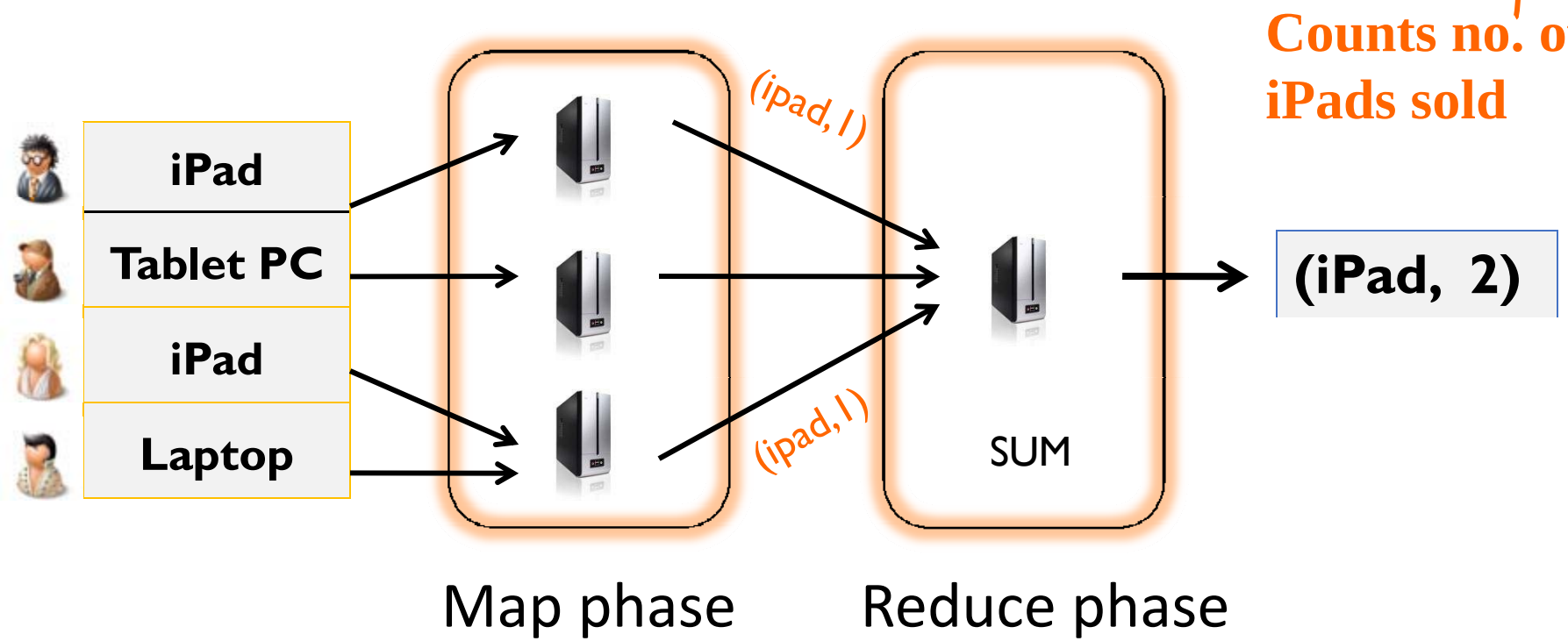


Indrajit Roy et. al, NSDI 2010

MapReduce example

Map(input)→{ if (input has iPad) print (iPad, 1) }

Reduce(key, list(v))→{ print (key + “,”+ SUM(v)) }

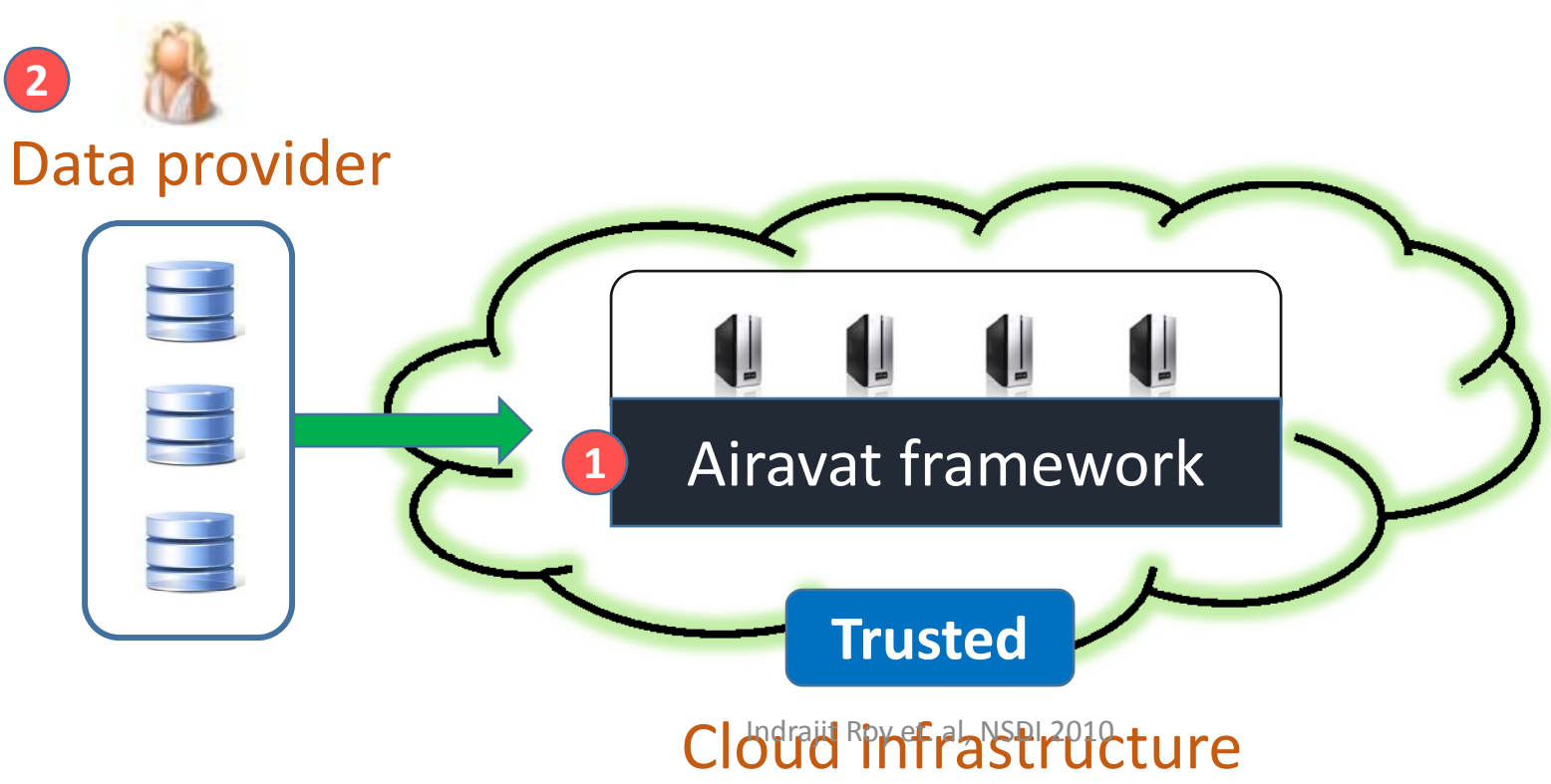


Airavat model

- Airavat framework runs on the cloud infrastructure
  - Cloud infrastructure: Hardware + VM
  - Airavat: Modified MapReduce + DFS + JVM + SELinux

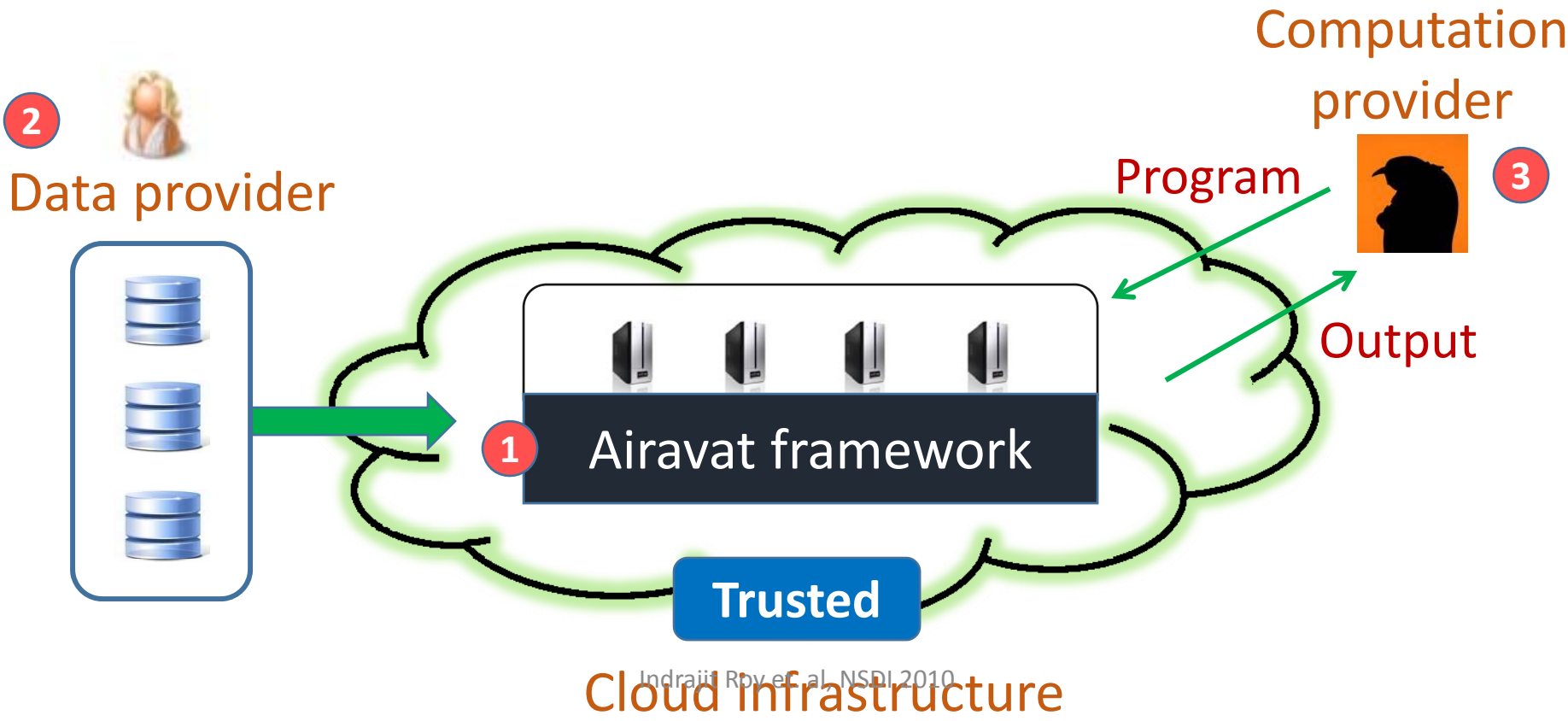
Airavat model

- Data provider uploads her data on Airavat
  - Sets up certain privacy parameters



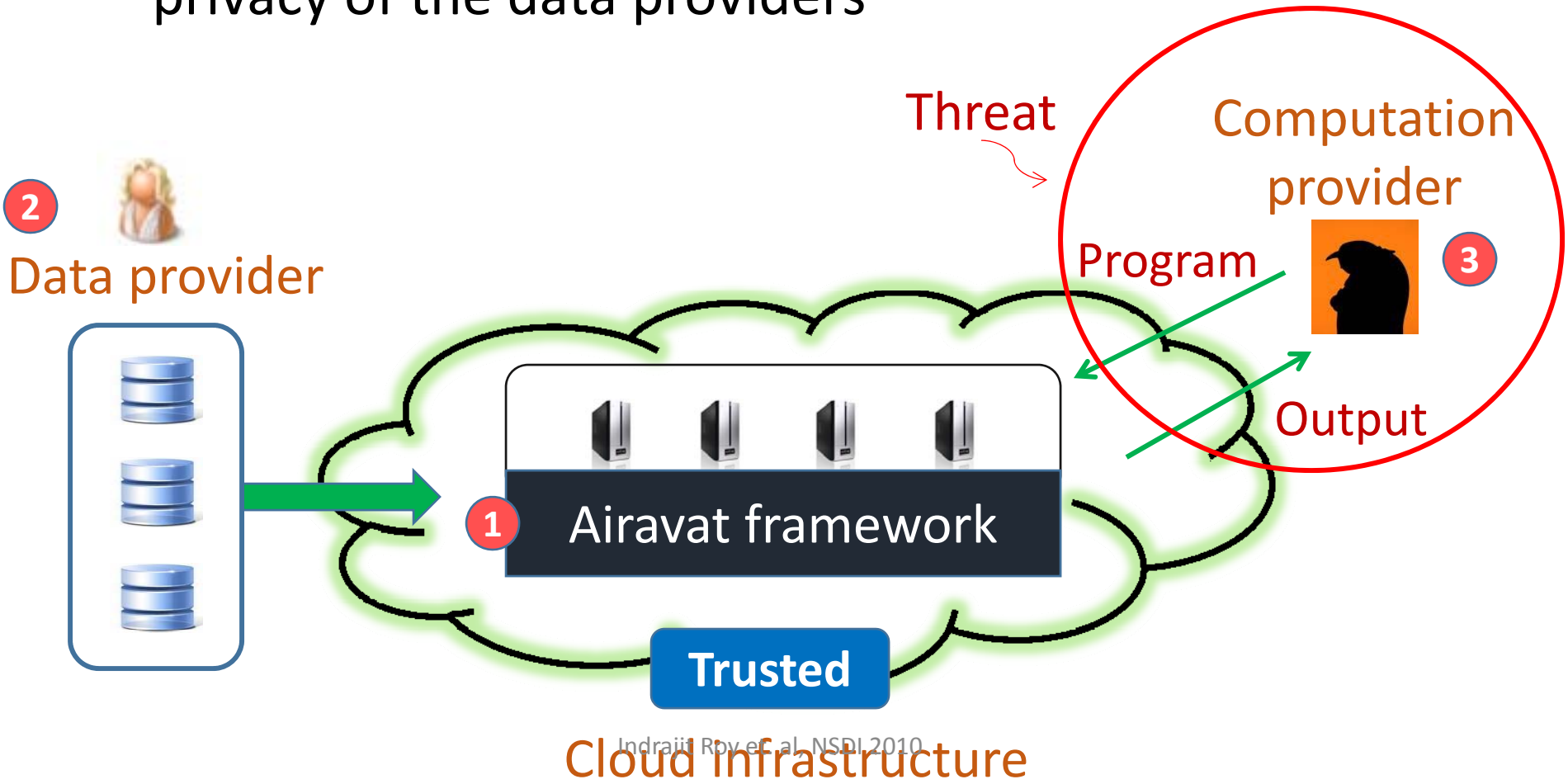
Airavat model

- Computation provider writes data mining algorithm
  - Untrusted, possibly malicious



Threat model

- Airavat runs the computation, and still protects the privacy of the data providers



Roadmap

What is the programming model?

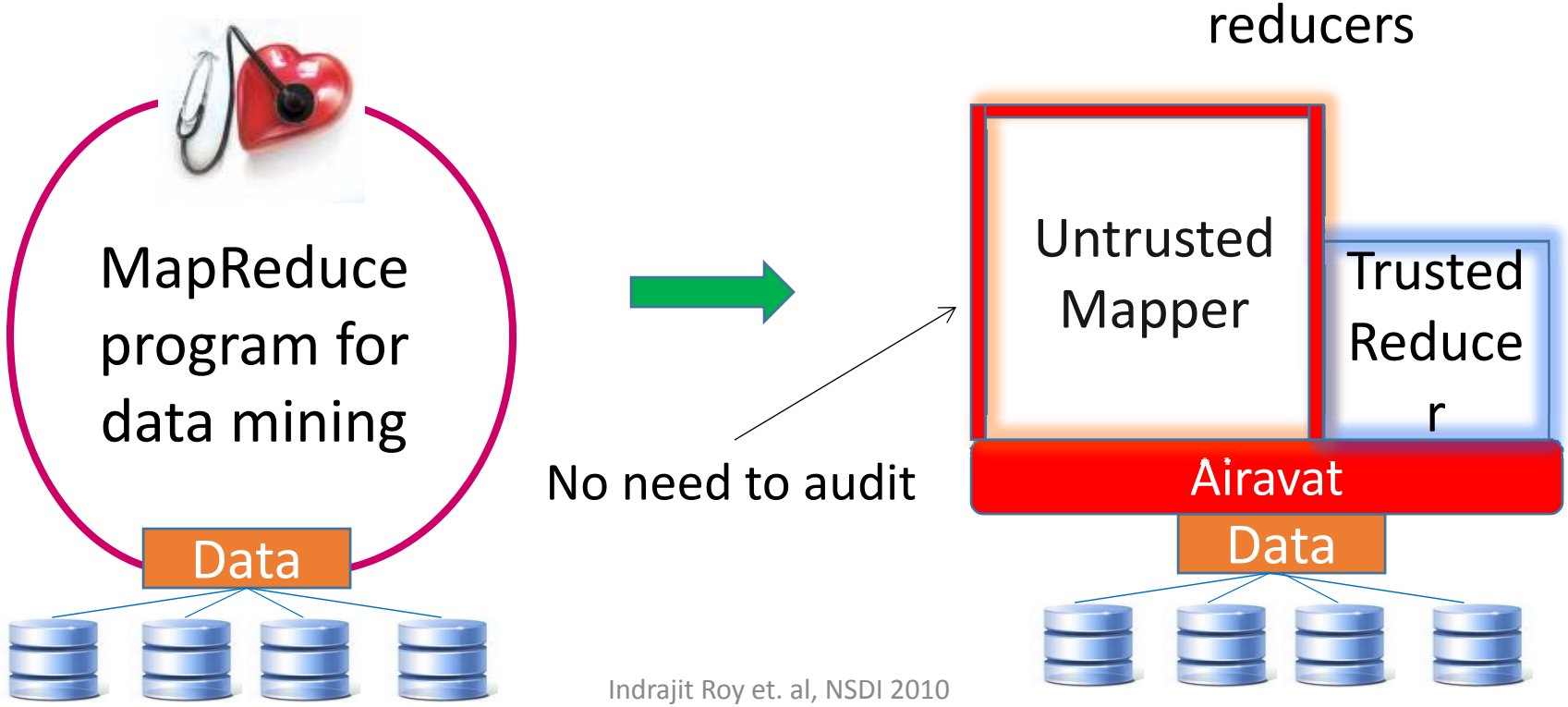
How do we enforce privacy?

What computations can be supported in Airavat?

Programming model

Split MapReduce into untrusted mapper + trusted reducer

Limited set of stock reducers

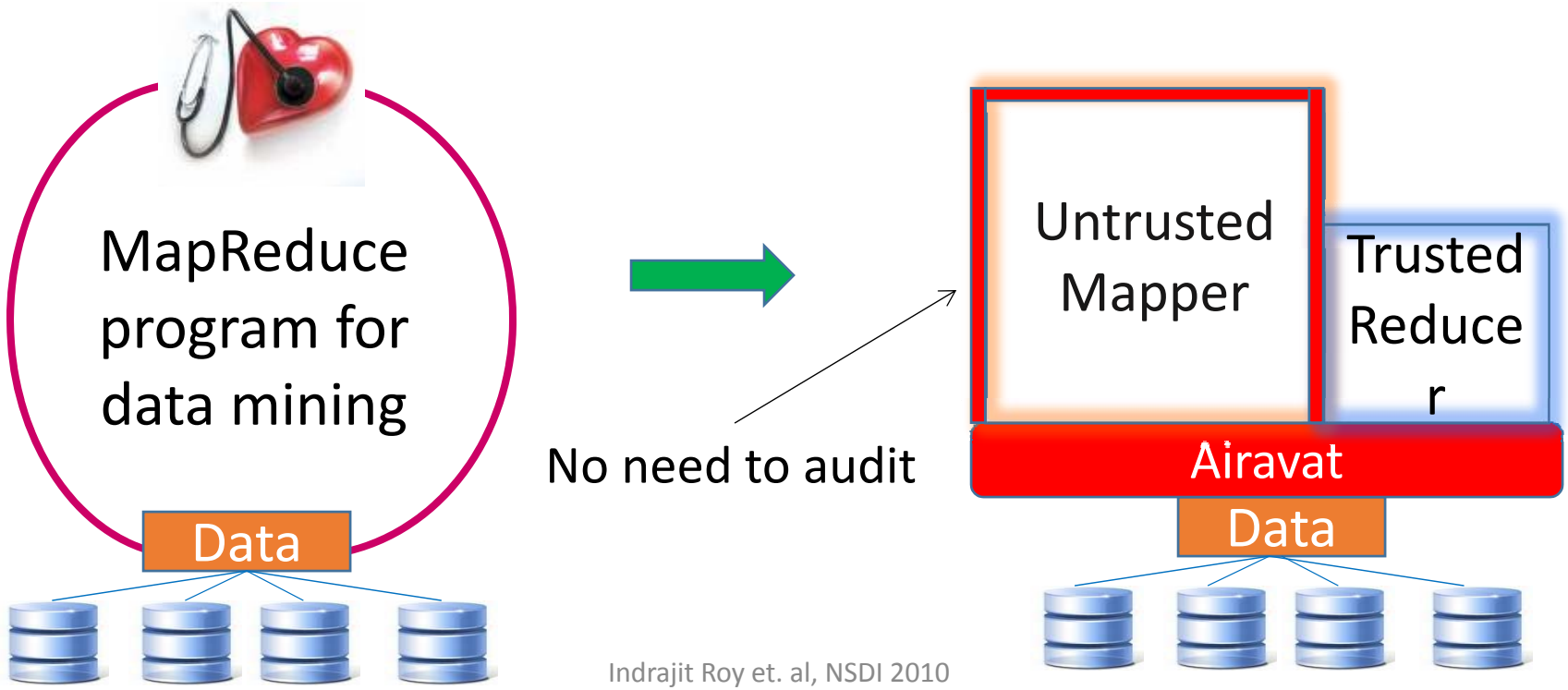


Indrajit Roy et. al, NSDI 2010

Programming model

Need to confine the mappers !

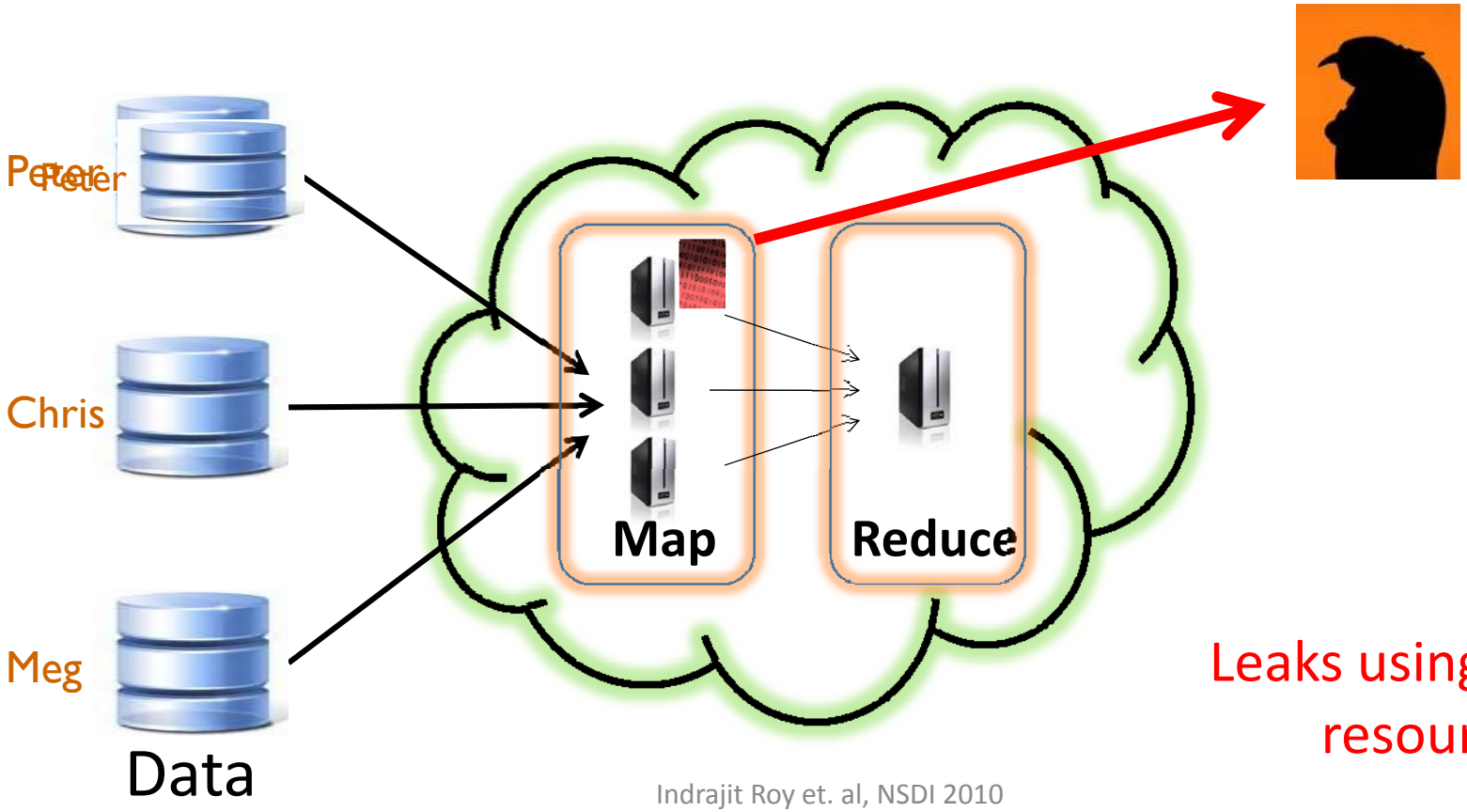
Guarantee: Protect the privacy of data providers



Indrajit Roy et. al, NSDI 2010

Challenge 1: Untrusted mapper

Untrusted mapper code copies data, sends it over the network

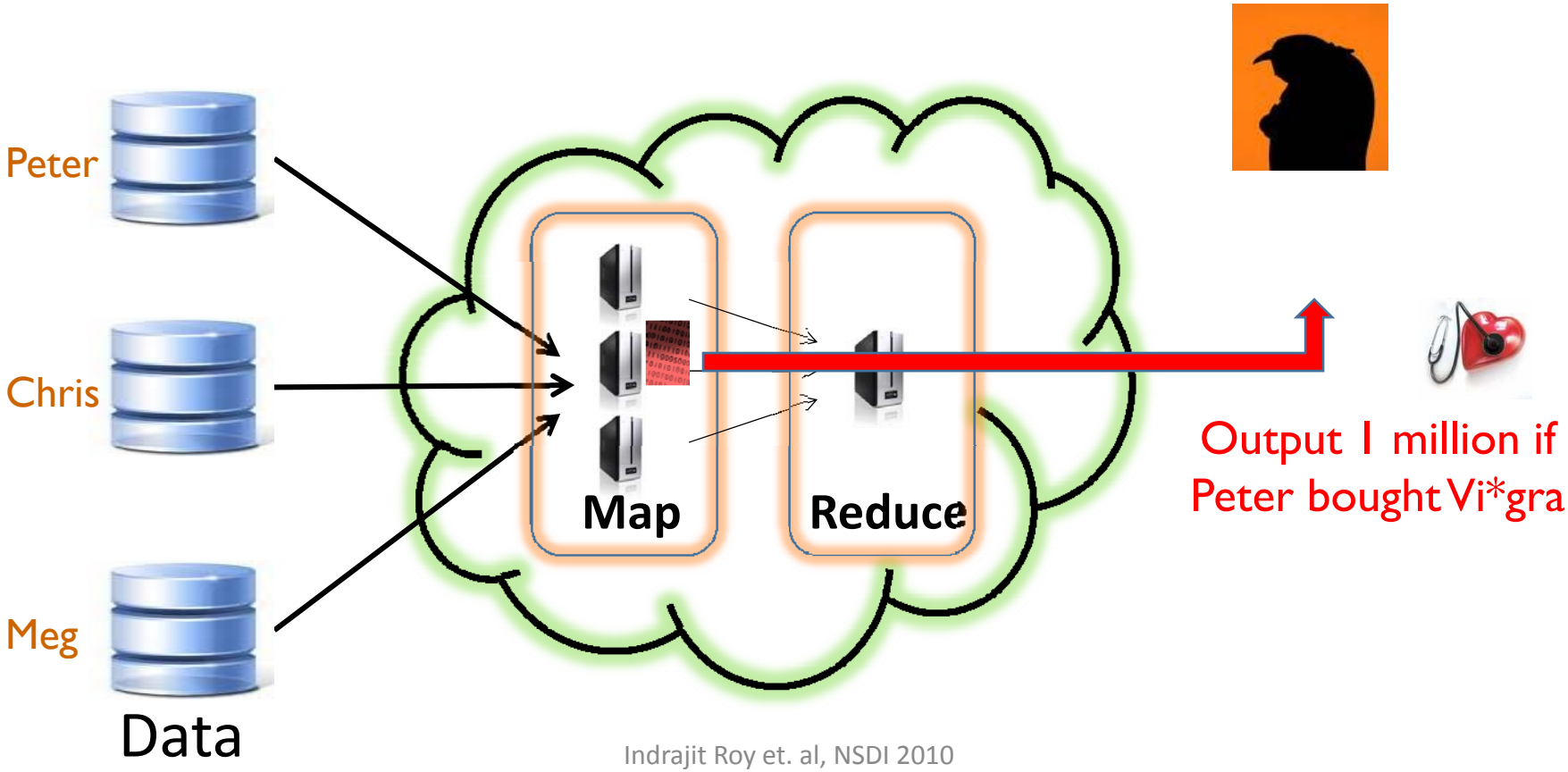


Leaks using system resources

Indrajit Roy et. al, NSDI 2010

Challenge 2: Untrusted mapper

Output of the computation is also an information channel



Indrajit Roy et. al, NSDI 2010

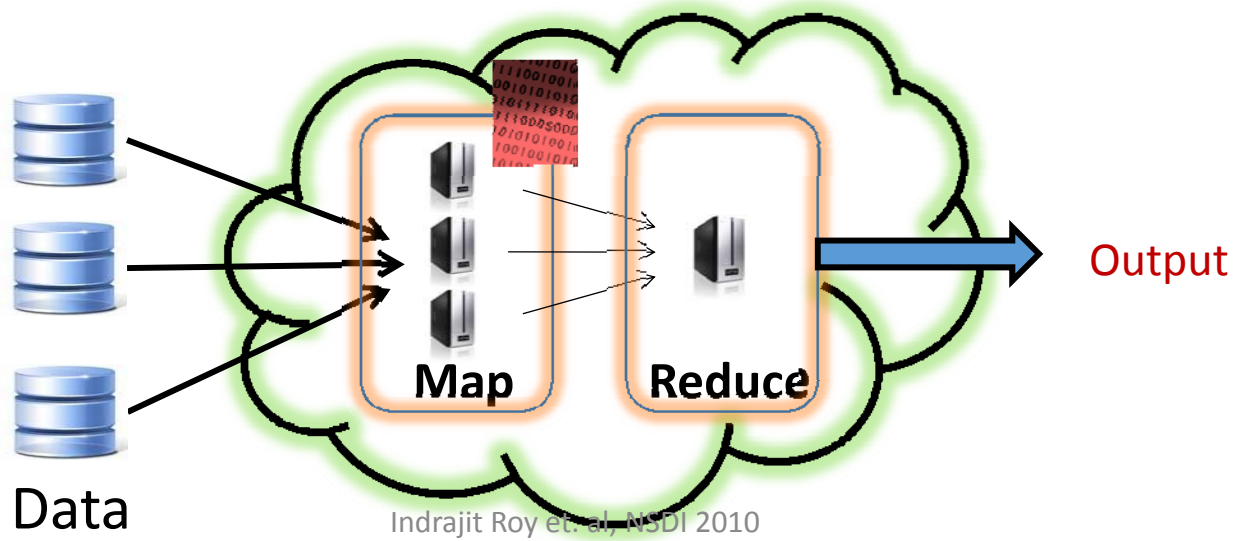
Airavat mechanisms

Mandatory access control

Prevent leaks through storage channels like network connections, files…

Differential privacy

Prevent leaks through the output of the computation



Indrajit Roy et. al, NSDI 2010

Back to the roadmap

What is the programming model?

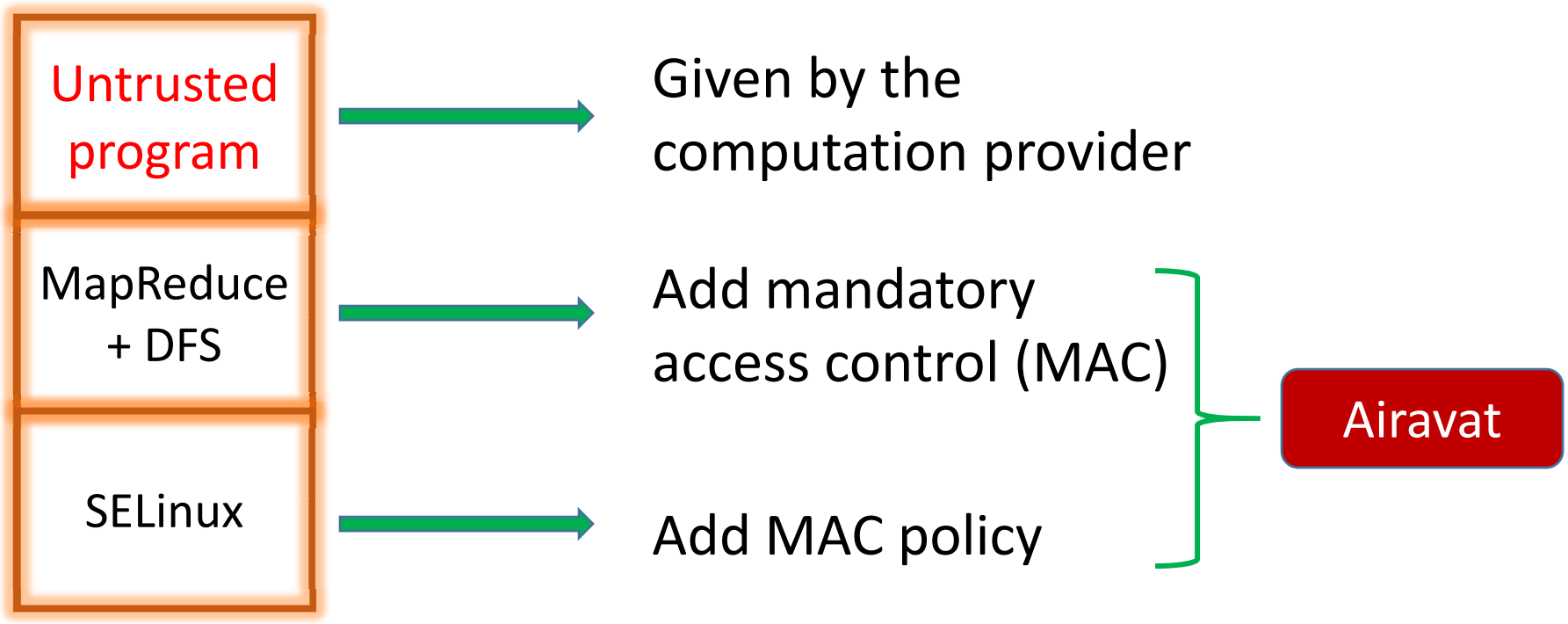
Untrusted mapper + Trusted reducer

How do we enforce privacy?

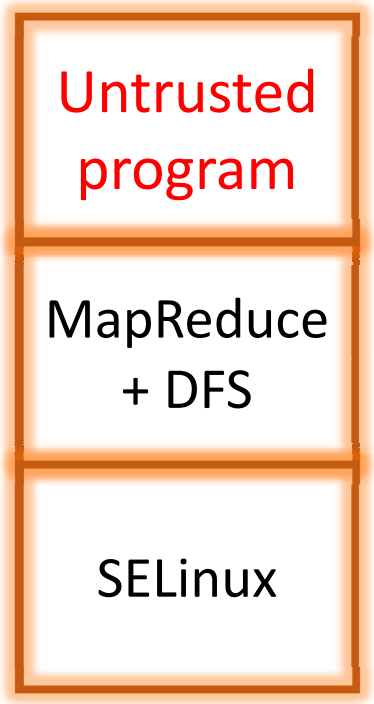
- Leaks through system resources
- Leaks through the output

What computations can be supported in Airavat?

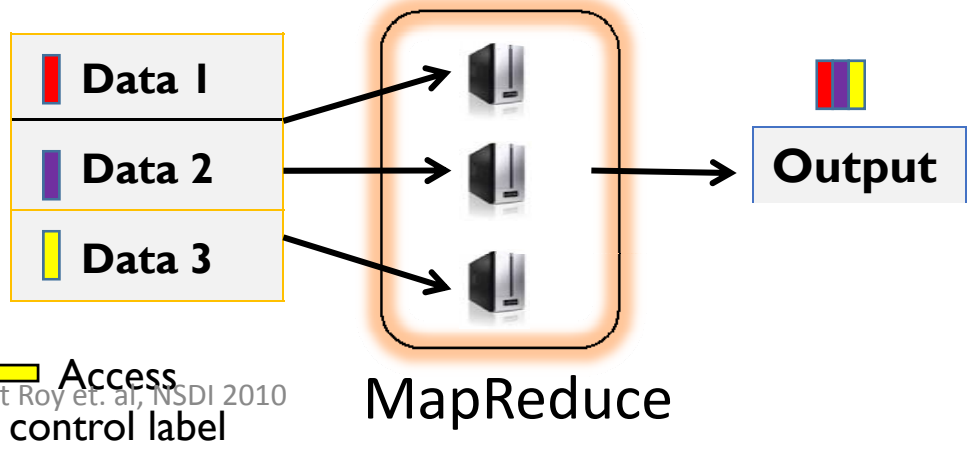
Airavat confines the untrusted code



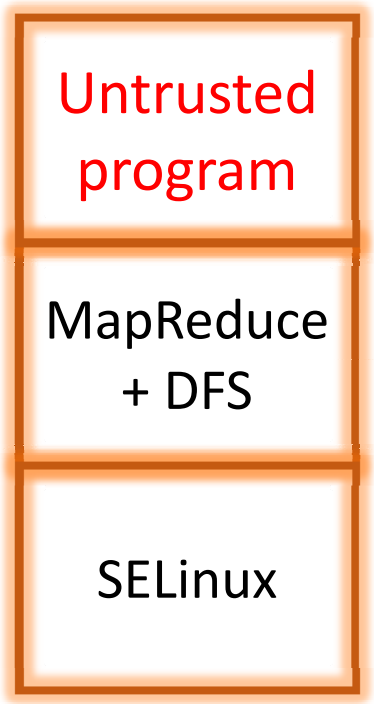
Airavat confines the untrusted code



- We add mandatory access control to the MapReduce framework
- Label input, intermediate values, output
- Malicious code cannot leak labeled data



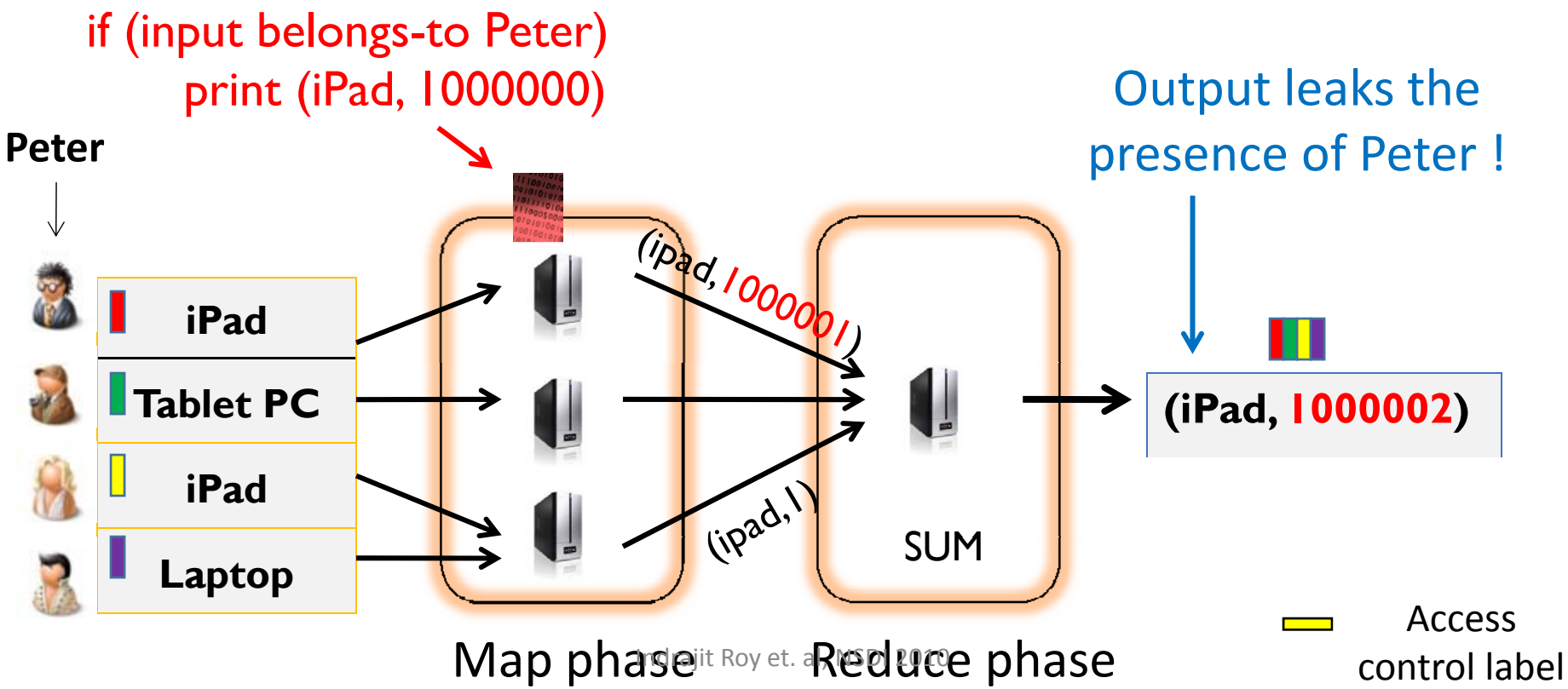
Airavat confines the untrusted code



- SELinux policy to enforce MAC
- Creates trusted and untrusted domains
- Processes and files are labeled to restrict interaction
- Mappers reside in untrusted domain
  - Denied network access, limited file system interaction

But access control is not enough

- Labels can prevent the output from been read
- When can we remove the labels?



But access control is not enough

Need mechanisms to enforce that the output does not violate an individual's privacy.

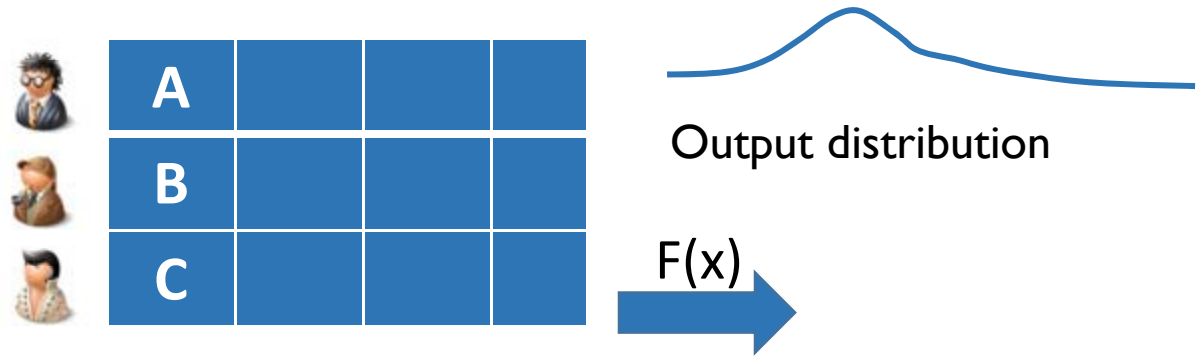
Background: Differential privacy

A mechanism is differentially private if every output is produced with similar probability whether any given input is included or not

Cynthia Dwork. *Differential Privacy*. ICALP 2006

Differential privacy (intuition)

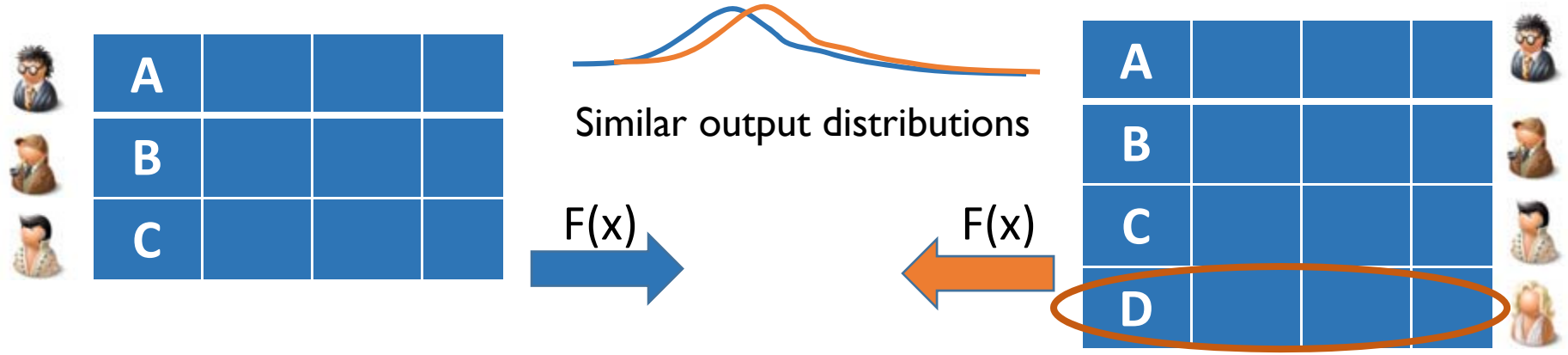
A mechanism is differentially private if every output is produced with similar probability whether any given input is included or not



Cynthia Dwork. *Differential Privacy*. ICALP 2006

Differential privacy (intuition)

A mechanism is differentially private if every output is produced with similar probability whether any given input is included or not

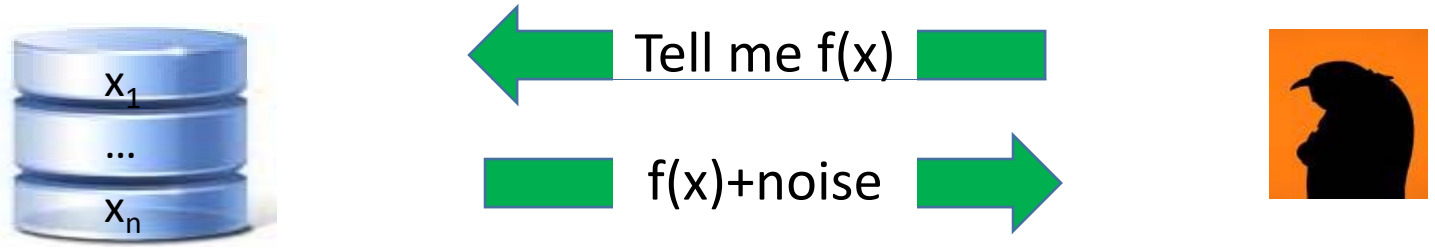


Bounded risk for D if she includes her data!

Cynthia Dwork. *Differential Privacy*. ICALP 2006

Achieving differential privacy

A simple differentially private mechanism



How much noise should one add?

Achieving differential privacy

Function sensitivity (intuition): Maximum effect of any single input on the output

- Aim: Need to conceal this effect to preserve privacy

Example: Computing the average height of the people in this room has low sensitivity

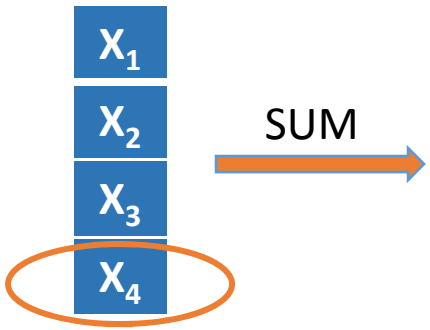
- Any single person's height does not affect the final average by too much
- Calculating the maximum height has high sensitivity

Achieving differential privacy

Function sensitivity (intuition): Maximum effect of any single input on the output

- Aim: Need to conceal this effect to preserve privacy

Example: SUM over input elements drawn from [0, M]

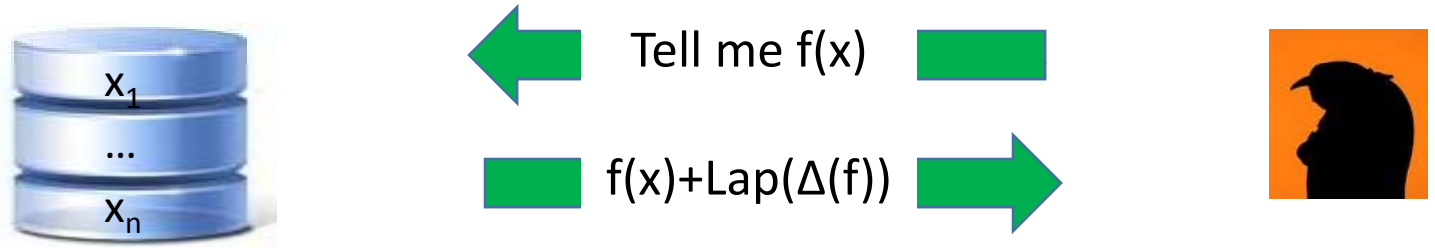


Sensitivity = M

Max. effect of any input element is M

Achieving differential privacy

A simple differentially private mechanism



Intuition: Noise needed to mask the effect of a single input

Δ(f) = sensitivity

Indrajit Roy et. al, NSDI 2010

Lap = Laplace distribution

Back to the roadmap

What is the programming model?

Untrusted mapper + Trusted reducer

How do we enforce privacy?

- Leaks through system resources
- Leaks through the output

MAC

What computations can be supported in Airavat?

Enforcing differential privacy

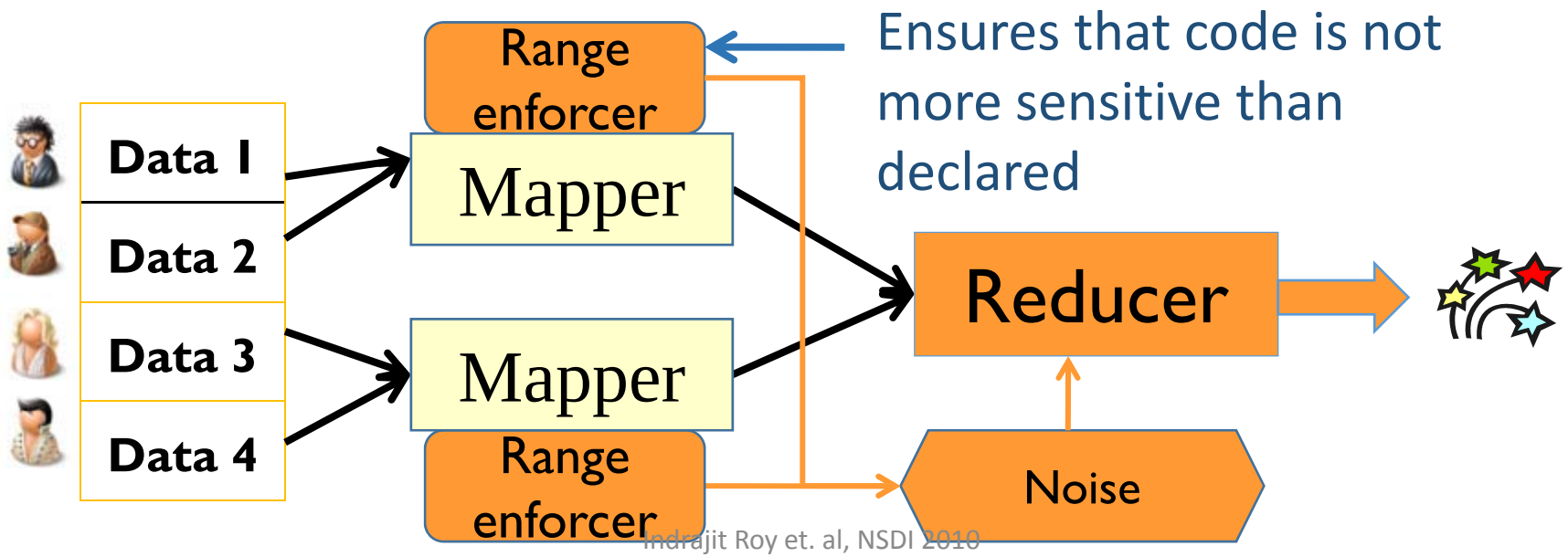
- Mapper can be any piece of Java code ("black box") but…
- Range of mapper outputs must be declared in advance
  - Used to estimate "sensitivity" (how much does a single input influence the output?)
  - Determines how much noise is added to outputs to ensure differential privacy
- Example: Consider mapper range [0, M]
  - SUM has the estimated sensitivity of M

Enforcing differential privacy

Malicious mappers may output values outside the range

If a mapper produces a value outside the range, it is replaced by a value inside the range

- User not notified… otherwise possible information leak



What can we compute?

- Reducers are responsible for enforcing privacy
  - Add an appropriate amount of random noise to the outputs
- Reducers must be trusted
  - Sample reducers: SUM, COUNT, THRESHOLD
  - Sufficient to perform data mining algorithms, search log processing, recommender system etc.
- With trusted mappers, more general computations are possible
  - Use exact sensitivity instead of range based estimates

Sample computations

- Many queries can be done with untrusted mappers
  - How many iPads were sold today? ← Sum
  - What is the average score of male students at UT? ← Mean
  - Output the frequency of security books that sold more than 25 copies today. ← Threshold
- … others require trusted mapper code
  - List all items and their quantity sold

Malicious mapper can encode information in item names

Revisiting Airavat guarantees

Allows differentially private MapReduce computations

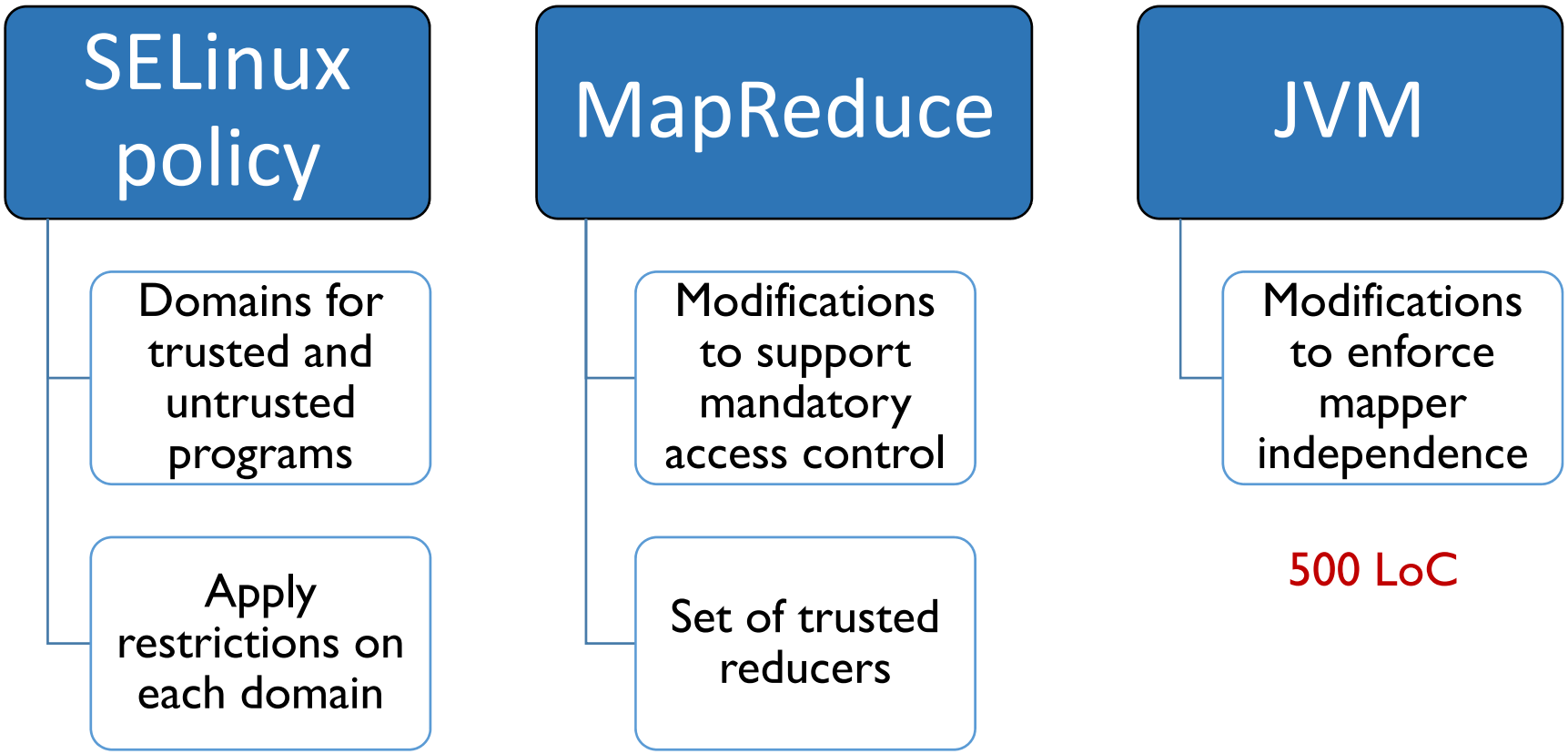
- Even when the code is untrusted

Differential privacy => mathematical bound on information leak

What is a safe bound on information leak ?

- Depends on the context, dataset
- Not our problem

mplementation details



Indrajit Roy et. al, NSDI 2010

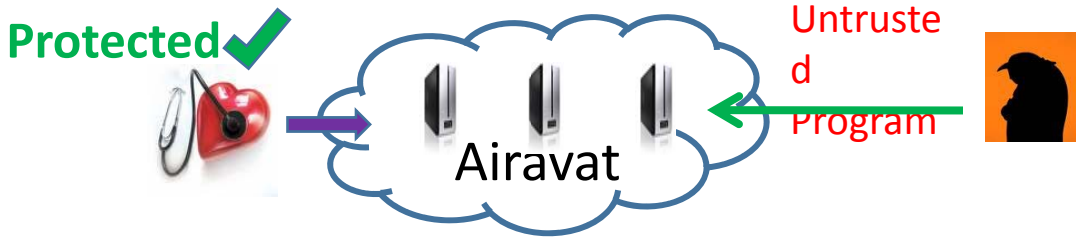
LoC = Lines of Code

Airavat in brief

Airavat is a framework for privacy preserving MapReduce computations

Confines untrusted code

First to integrate mandatory access control with differential privacy for end-to-end enforcement



References

- Balaji Palanisamy, Aameek Singh, Ling Liu and Bryan Langston, "Cura: A Cost-Optimized Model for MapReduce in *IPDPS'13*
- Balaji Palanisamy, Aameek Singh, Ling Liu and Bhushan Jain, "Purlieus: Locality-aware Resource Allocation for MapReduce in a Cloud", Proc. of *IEEE/ACM Supercomputing (SC' 11).*
- Balaji Palanisamy, Aameek Singh, Nagapramod Mandagere, Gabriel Alatorre and Ling Liu, "VNCache: Bridging Compute and Storage for Big Data in a Cloud" in CCGrid 2014.
- Indrajit Roy, Srinath T.V. Setty, Ann Kilzer, Vitaly Shmatikov, Emmett Witchel, "Airavat: Security and Privacy for MapReduce", in NSDI 2010.
- Balaji Palanisamy, Aameek Singh, Ling Liu and Bryan Langston, "Cura: A Cost-Optimized Model for MapReduce in *IEEE Transactions on Parallel and Distributed Systems (TPDS 2014)*