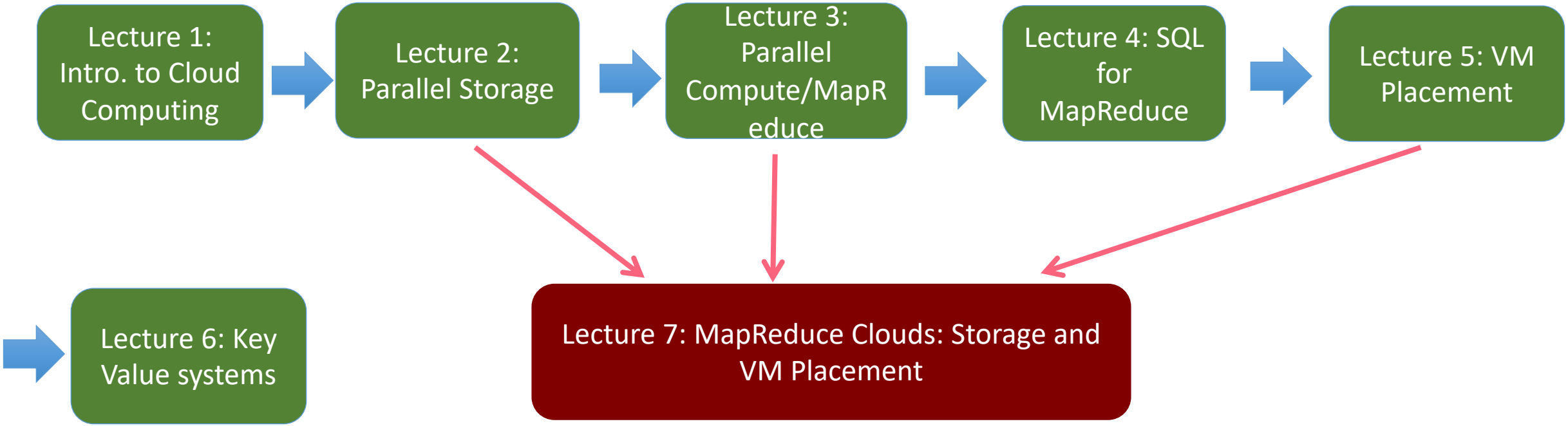
MapReduce Cloud services

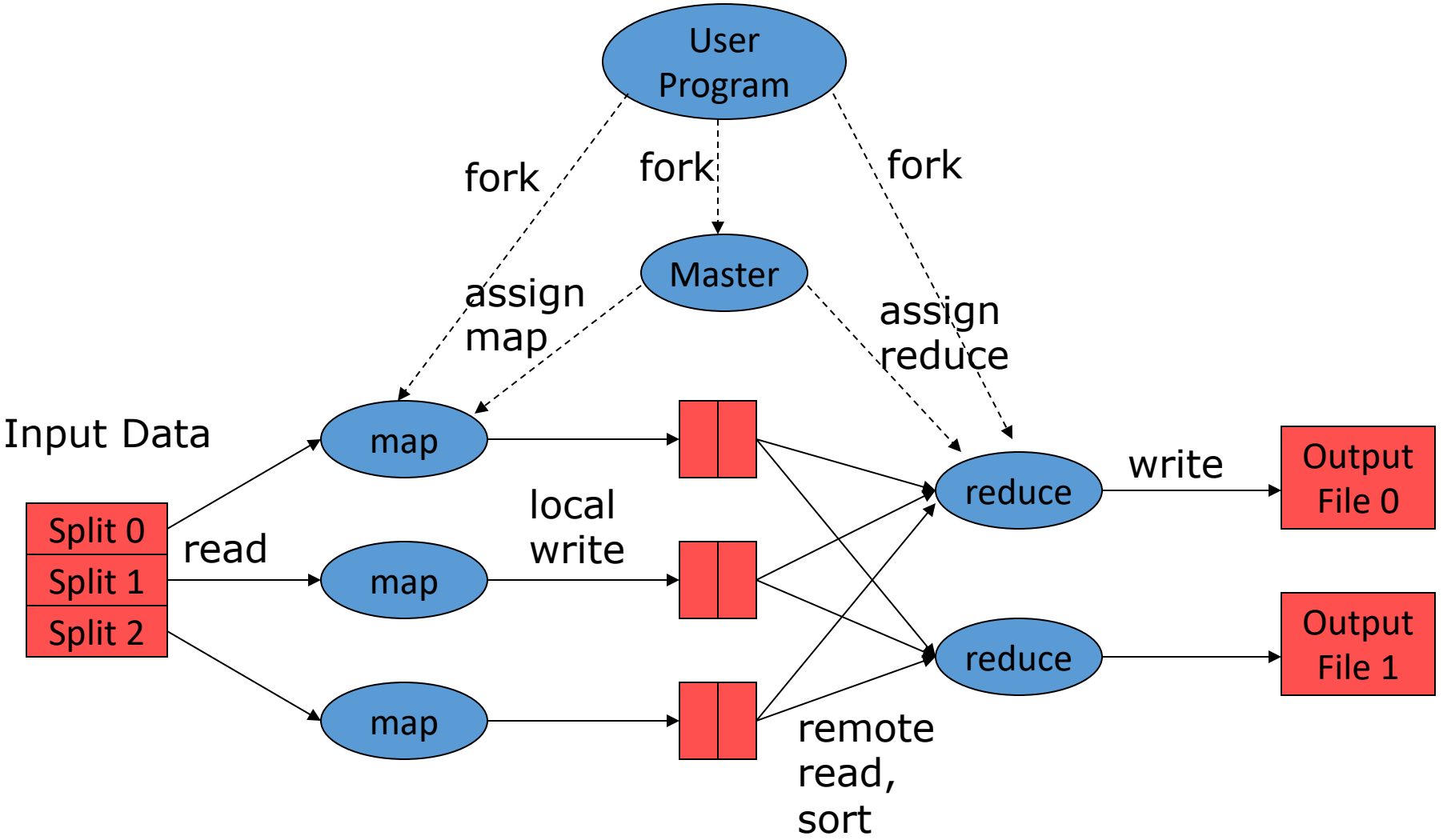
Balaji Palanisamy

Associate Professor
School of Computing and Information
University of Pittsburgh
bpalan@pitt.edu

Recap


Lecture 1: Intro. to Cloud Computing
Lecture 2: Parallel Storage
Lecture 3: Parallel Compute/MapReduce
Lecture 4: SQL for MapReduce
Lecture 5: VM Placement
Lecture 6: Key Value systems
Lecture 7: MapReduce Clouds: Storage and VM Placement

MapReduce Execution Overview



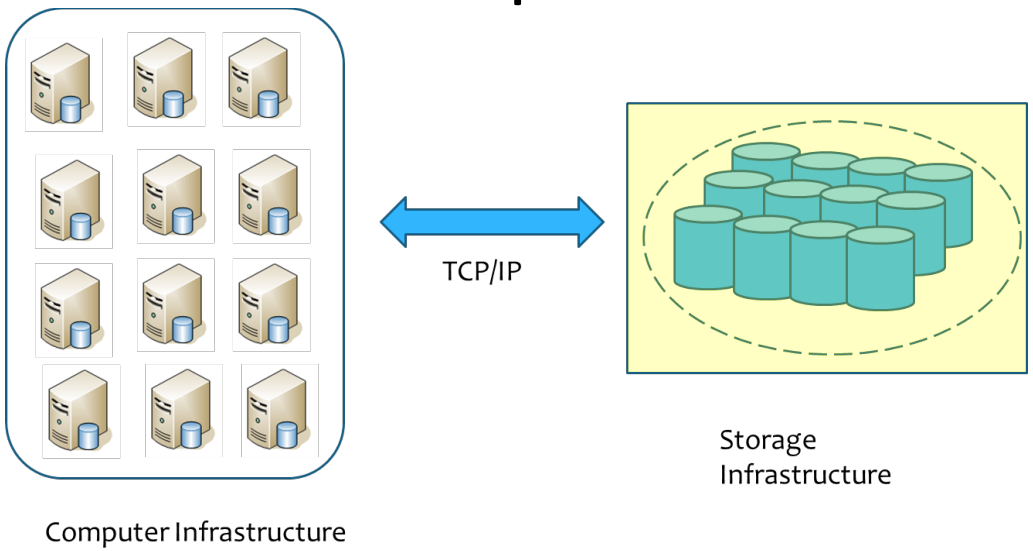
MapReduce cloud provider - Challenges

- 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



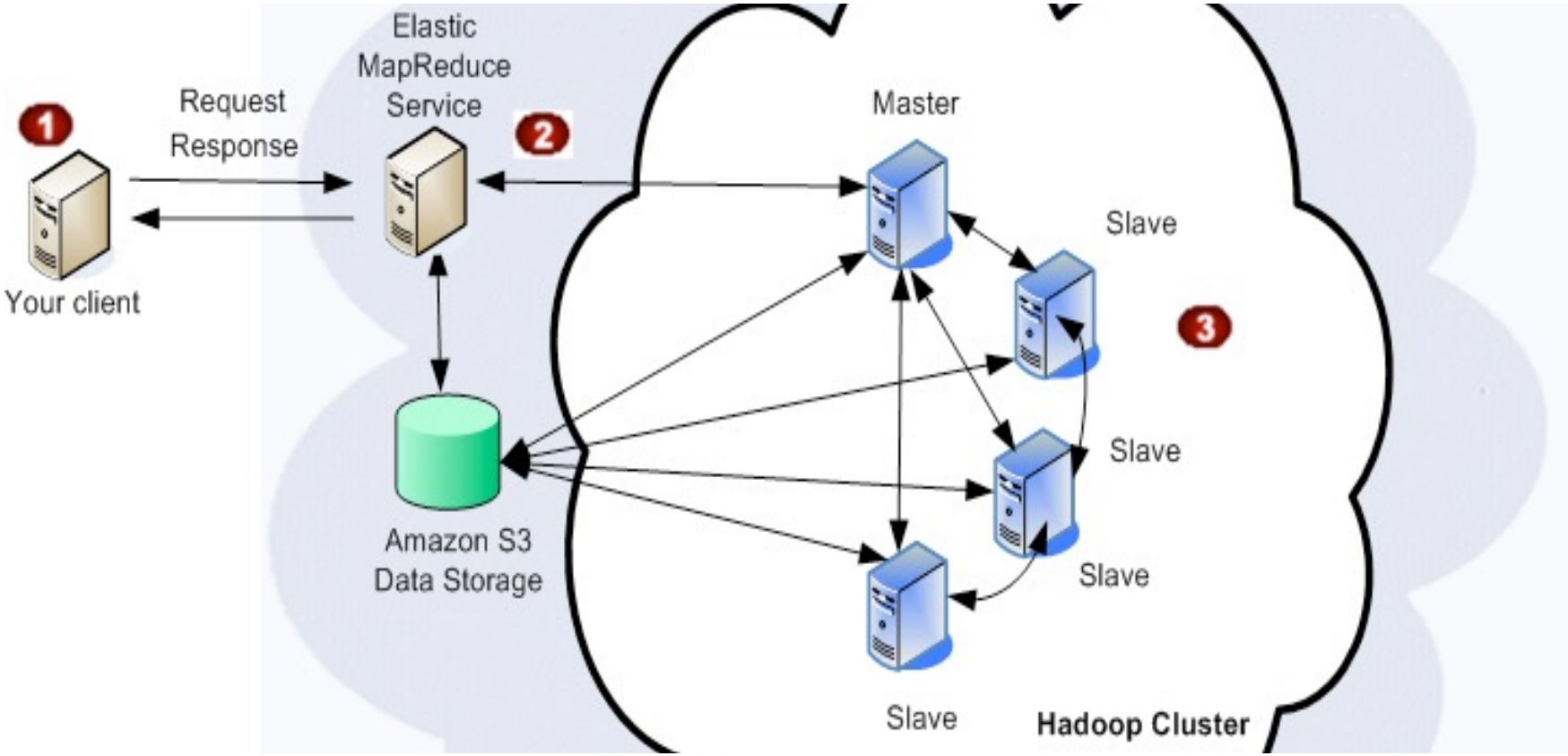
Amazon Elastic MapReduce Model

- 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*

Amazon Elastic MapReduce Model



Problem

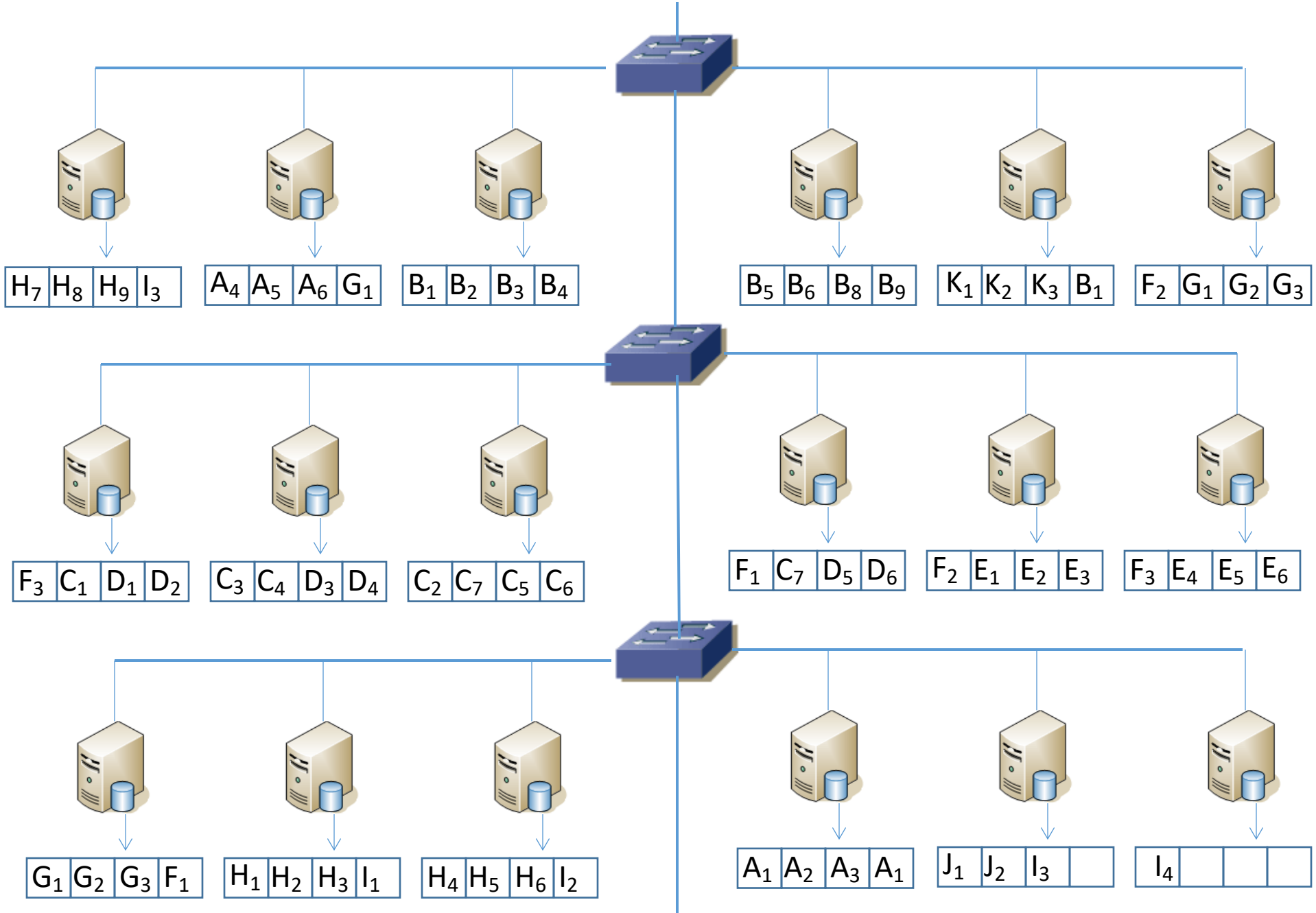
- How long does it for 200 VMs to download a 300 TB dataset in parallel from a Storage Cloud to Compute Cloud at the rate of 100 Megabytes per second per VM?

  - 200 VMs download 300 TB in parallel. Therefore, each VM downloads 300/200 = 1.5 TB
  - Time taken for a VM to download 1.5 TB
    - = 1.5 TB/100MB = 1.5 * 1000000/ 100 seconds
    - = 15000 seconds = 250 minutes

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

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

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

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

- 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

Problem

- Categorize the following jobs into (i) Map and Reduce input Heavy, (ii) only Map input heavy and (iii) only Reduce input heavy

Job	Input data	Output data	Job Category
Job1	20 GB	10 KB	
Job 2	20 GB	20 GB	
Job 3	1 GB	40 GB	

Problem

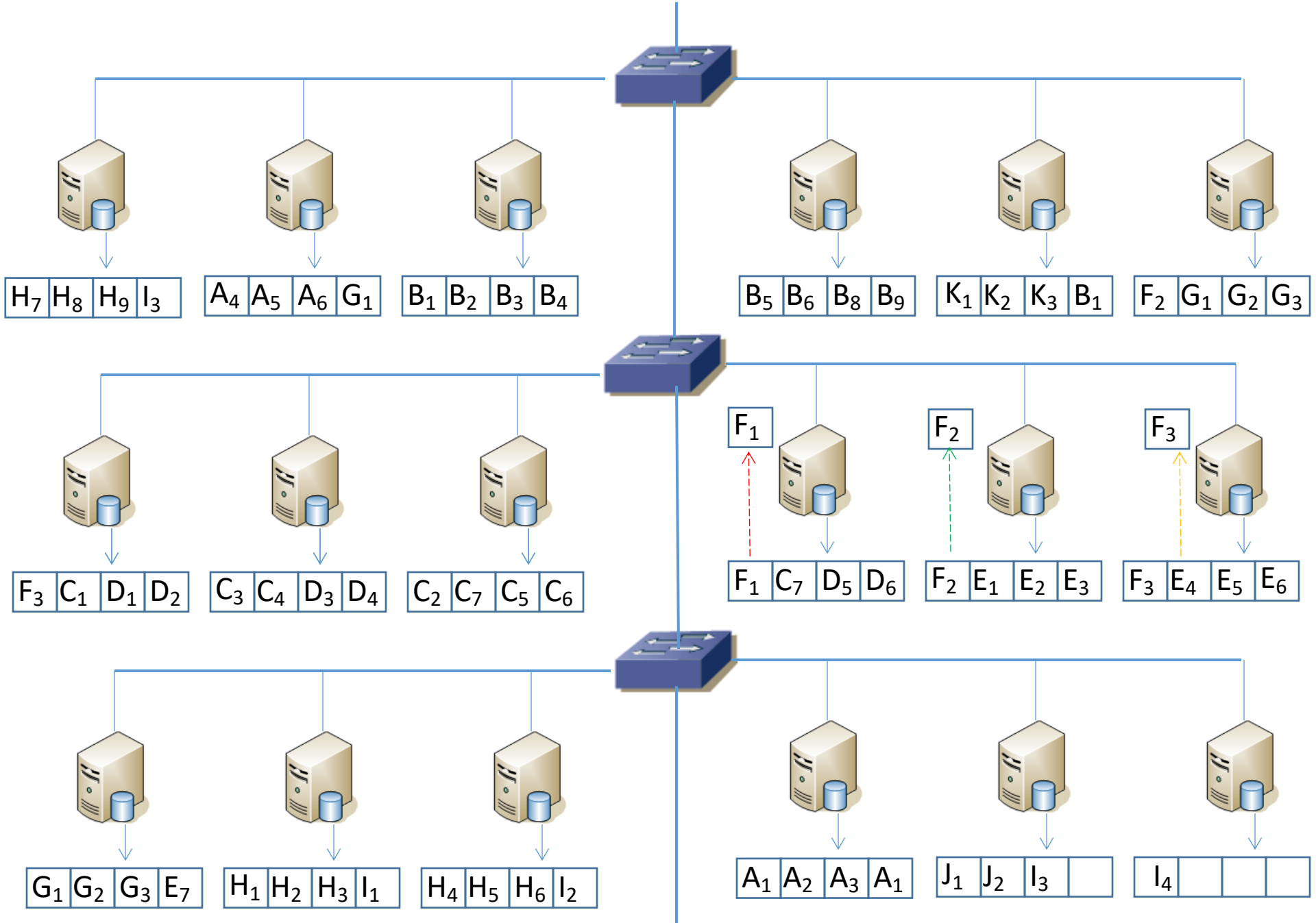
- Categorize the following jobs into (i) Map and Reduce input Heavy, (ii) only Map input heavy and (iii) only Reduce input heavy

Job	Input data	Output data	Job Category
Job 1	20 GB	10 KB	Map input heavy
Job 2	20 GB	20 GB	Map and Reduce input heavy
Job 3	1 GB	40 GB	Reduce input heavy

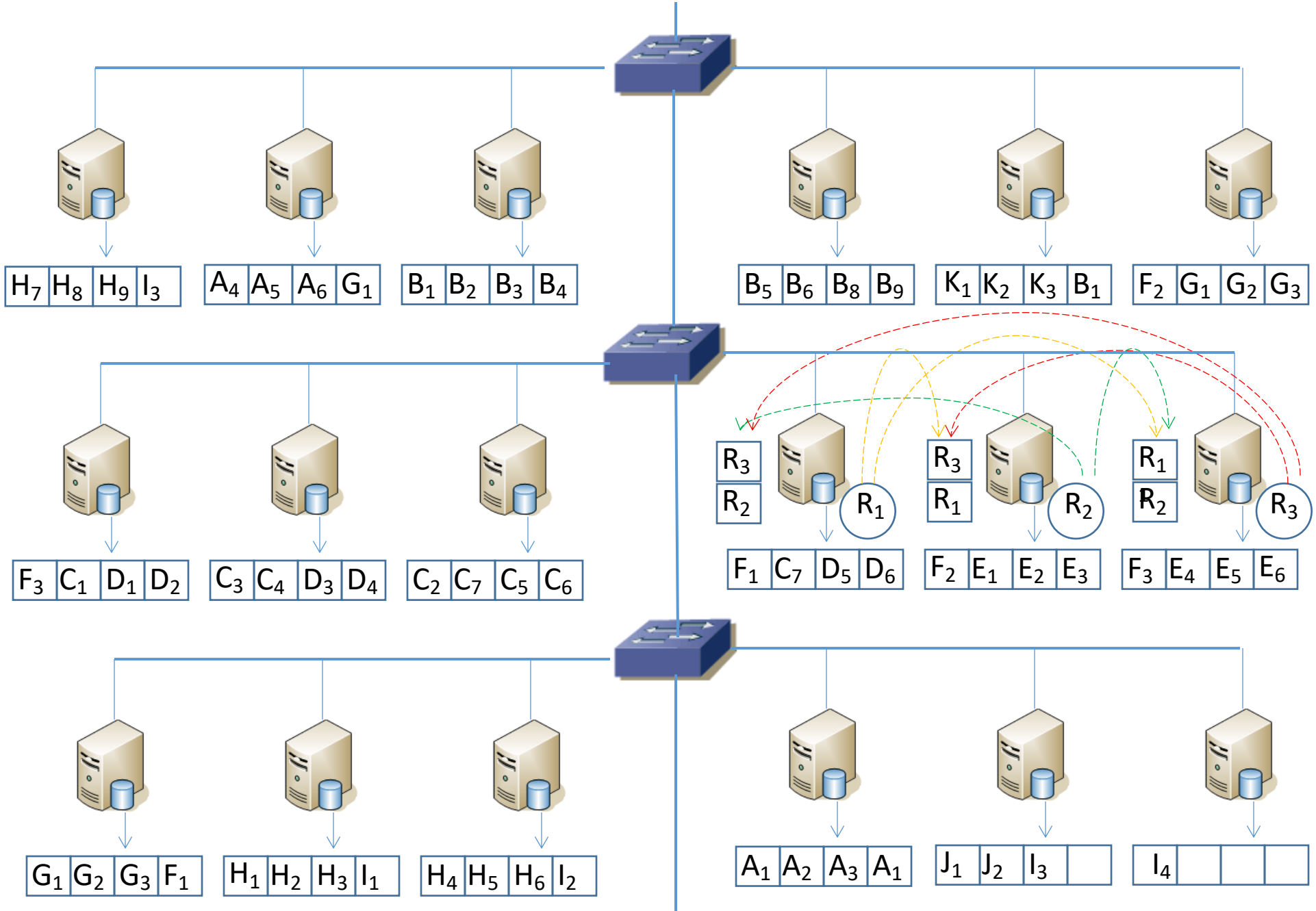
Map and Reduce input-heavy jobs

- 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

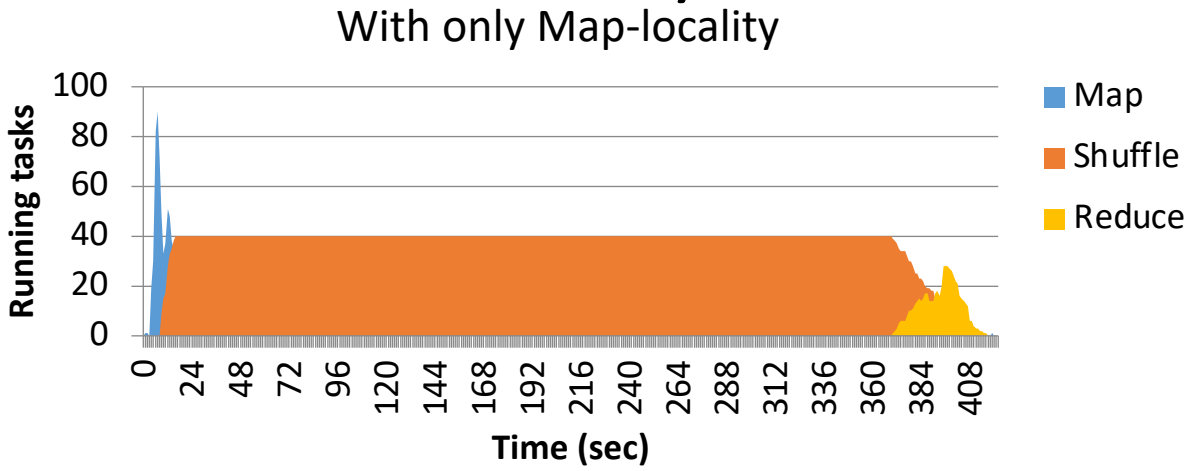
Scheduling Map and Reduce-input heavy tasks (map-phase)

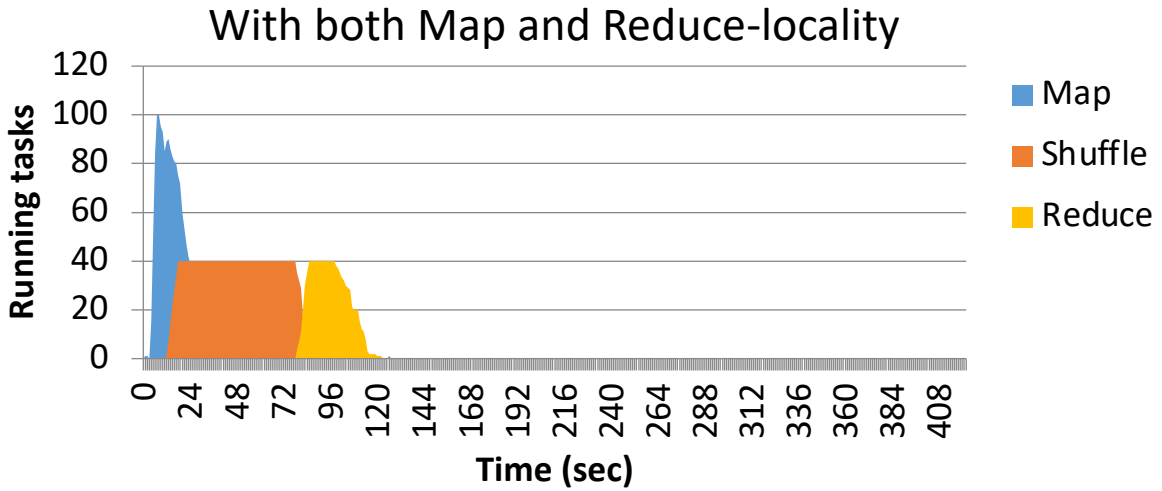


Scheduling Map and Reduce-input heavy tasks (reduce-phase)



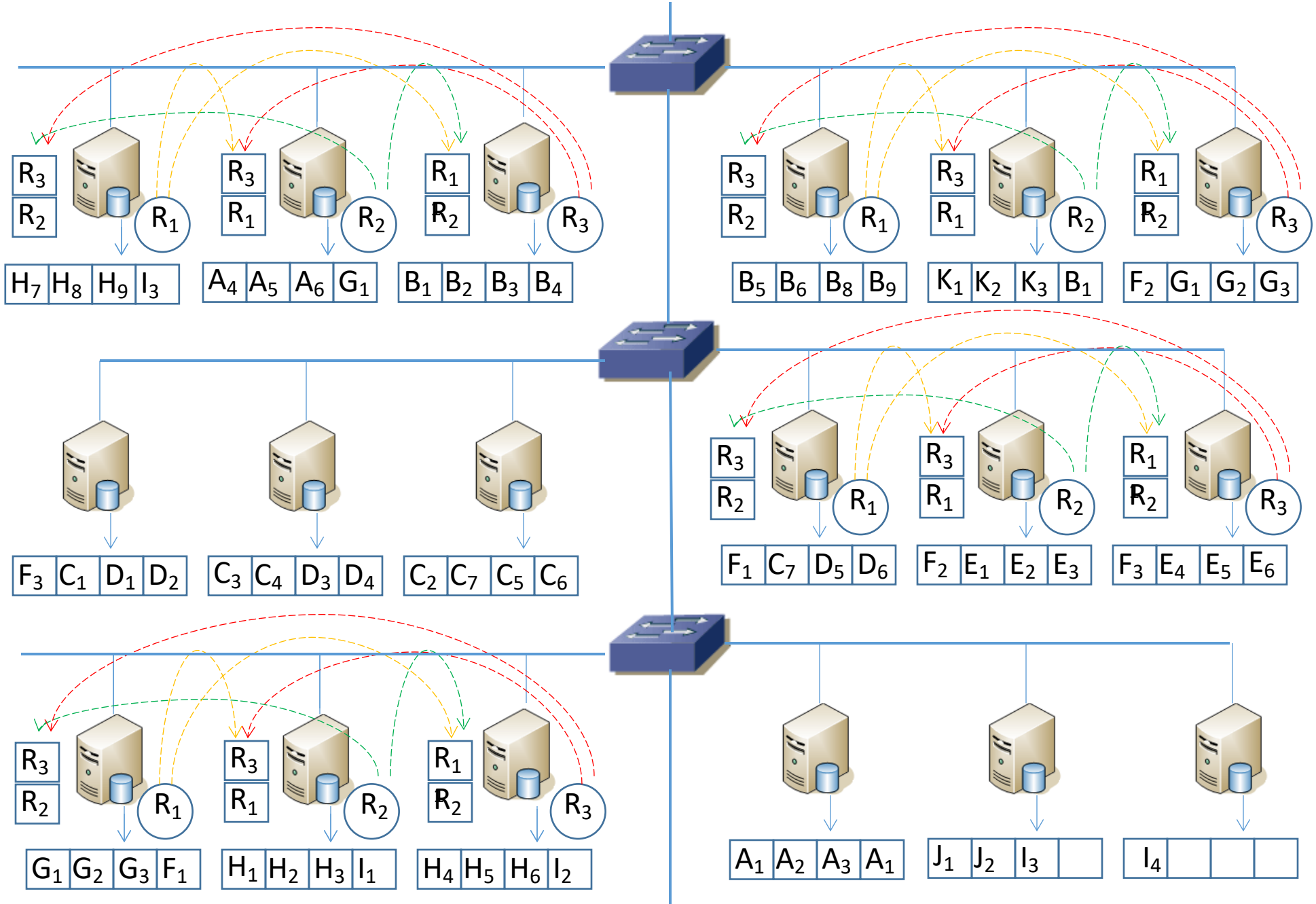
Impact of Reduce-locality





Timeline plotted using Hadoop *job_history_summary*

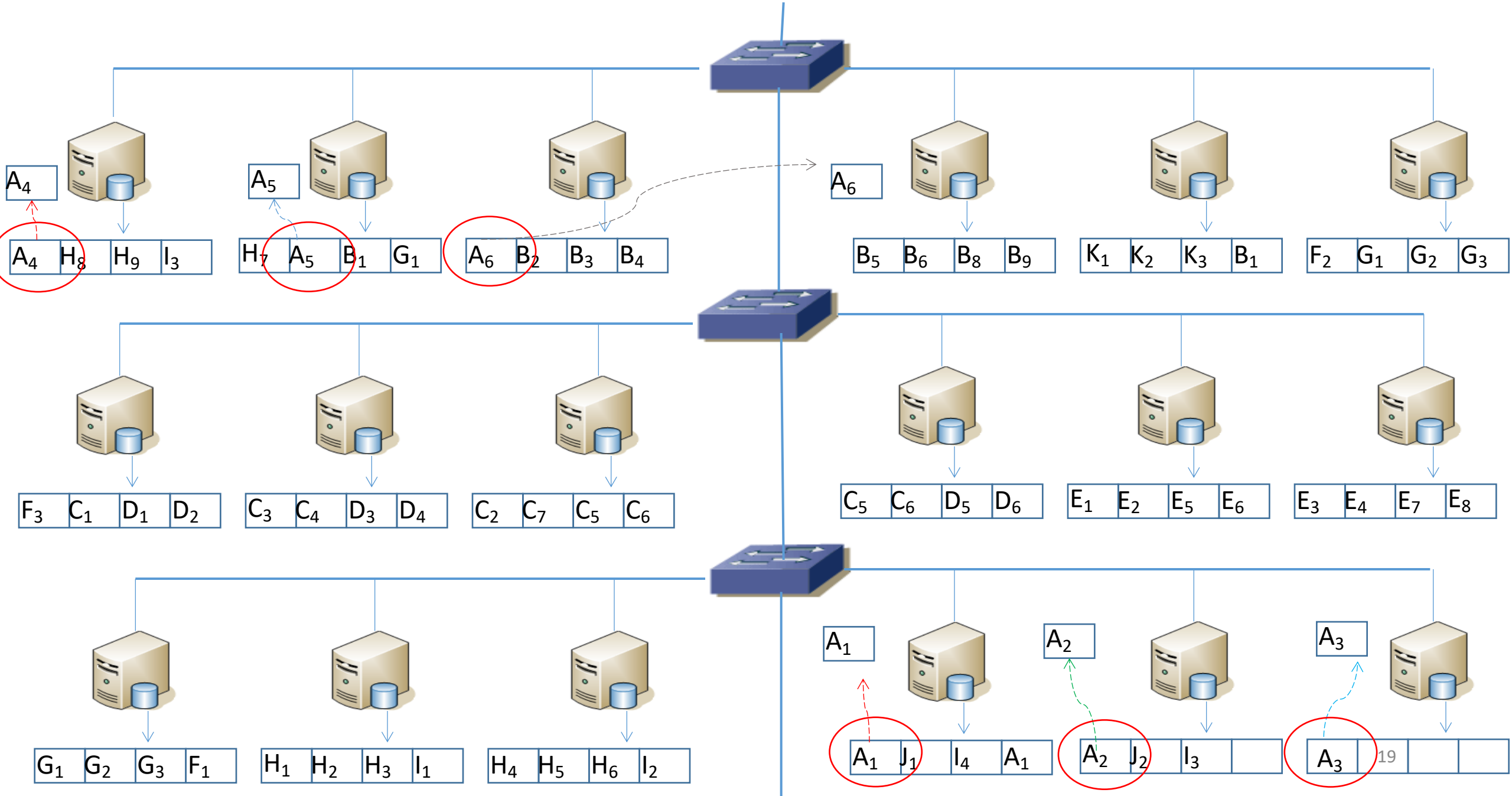
Data center network stress



Map input-heavy jobs

- Example: counting the number of occurrences of a given word in a huge file.
- Only Map locality of the jobs is important
- Map phase
  - Incorporates map locality by pushing compute towards data
- Reduce phase:
  - No hard constraints on reduce locality

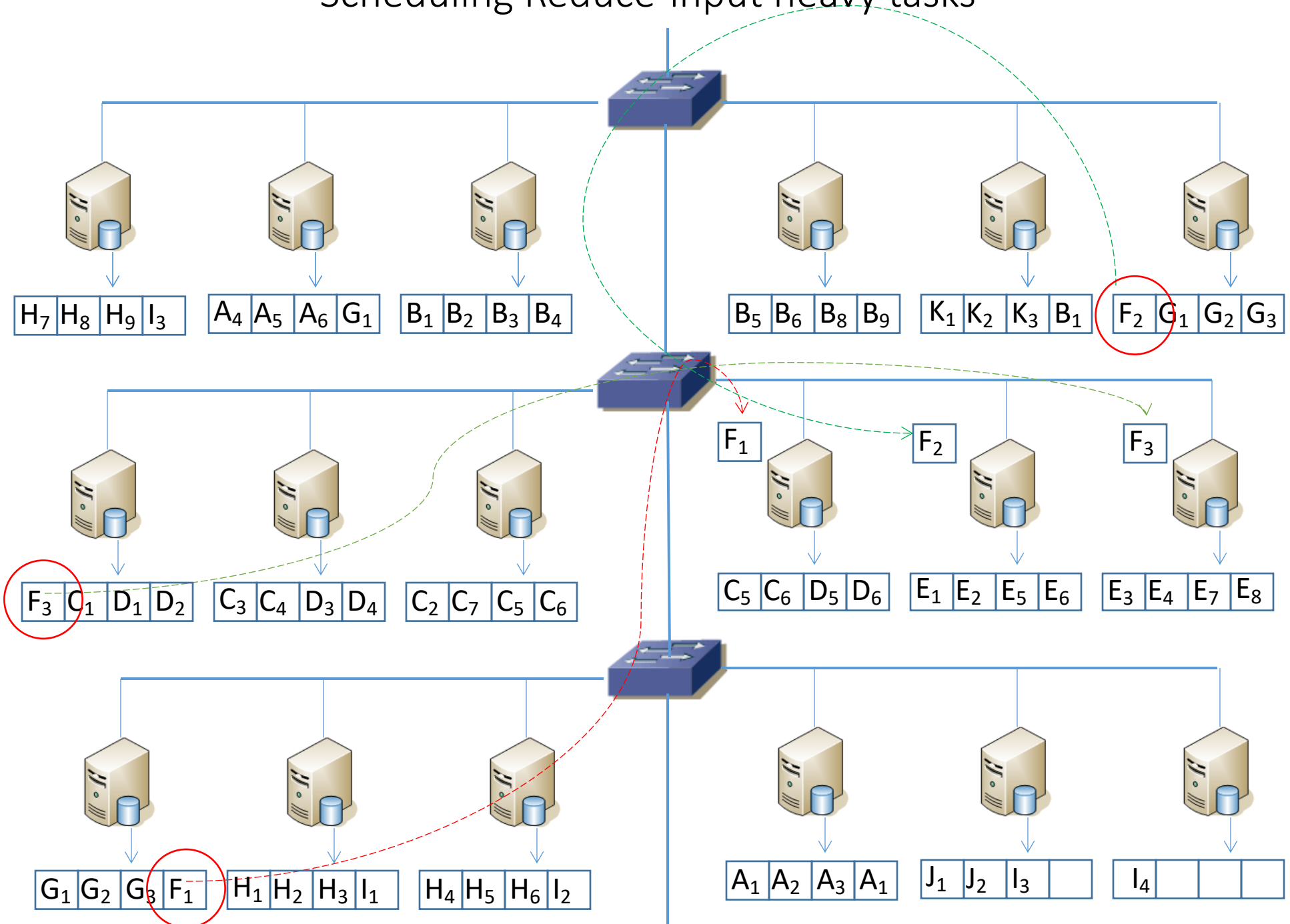
Scheduling Map-input heavy jobs



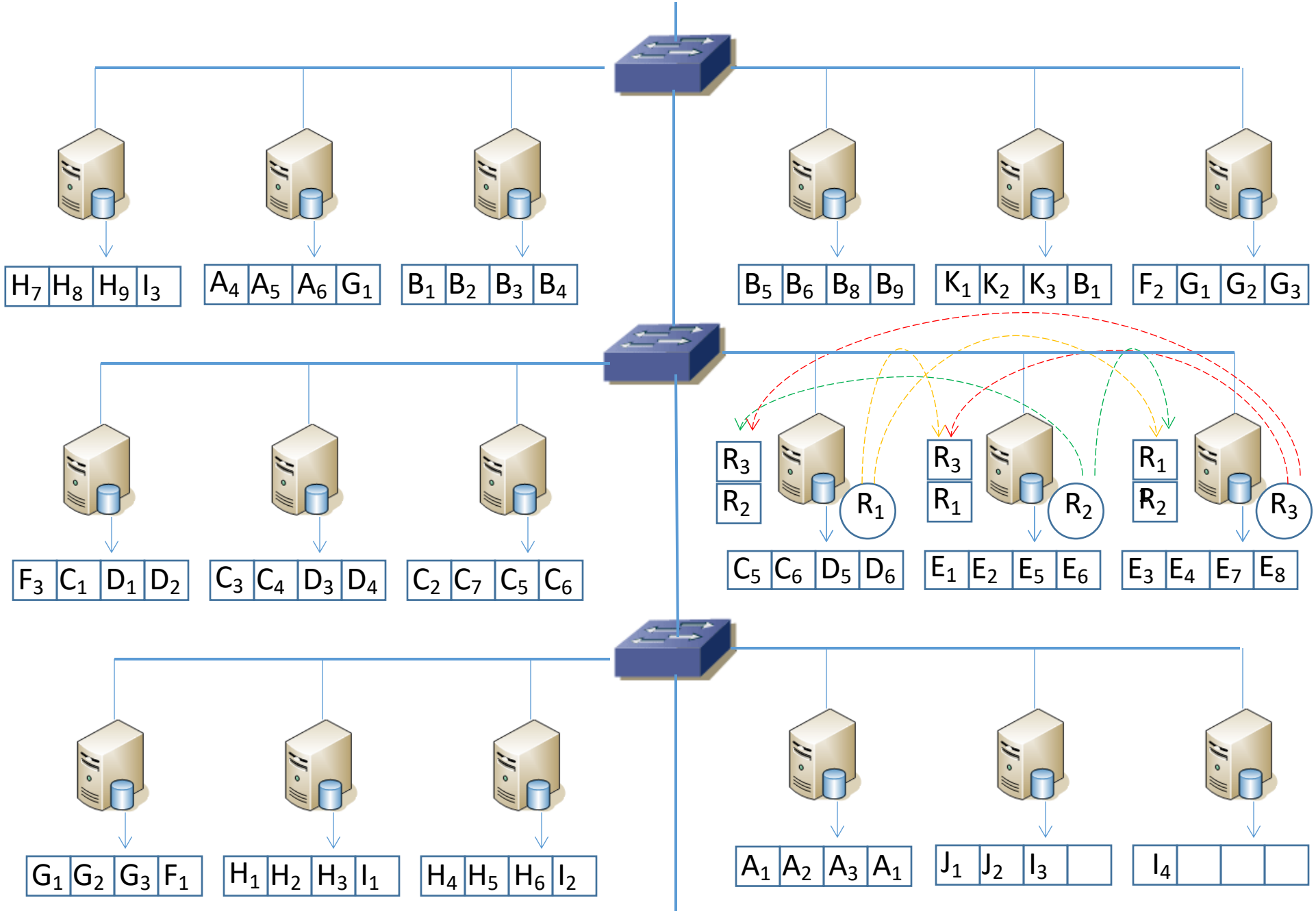
Reduce input-heavy jobs

- Example
  - Generating a synthetic data set
- Do not require map-locality, however reduce-locality is crucial
- Map phase
  - No hard constraints on map locality
  - Need not push compute towards data
- Reduce phase
  - Minimizes cross-rack traffic

Scheduling Reduce-input heavy tasks



Scheduling Reduce-input heavy tasks



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.

Map and Reduce-input heavy workload

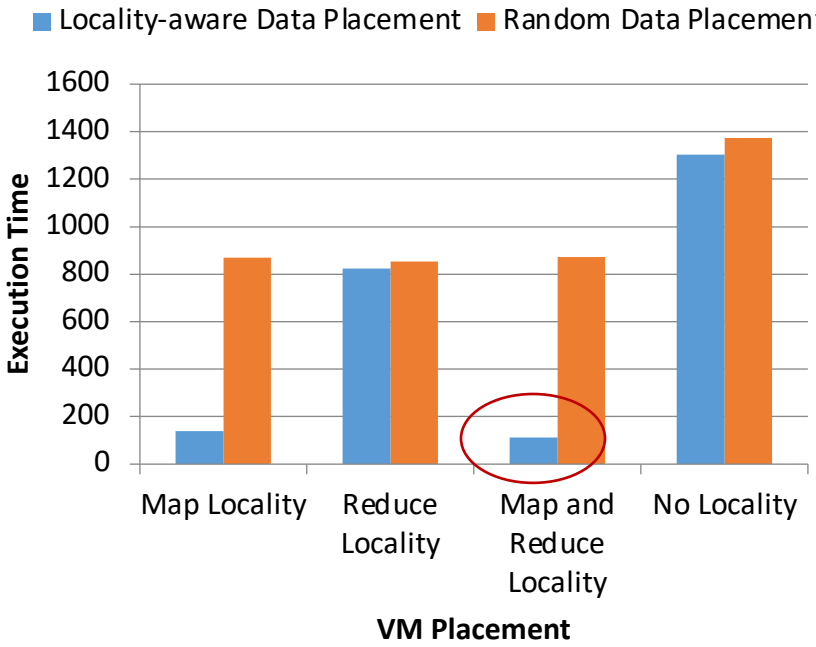


Fig 3. Execution time

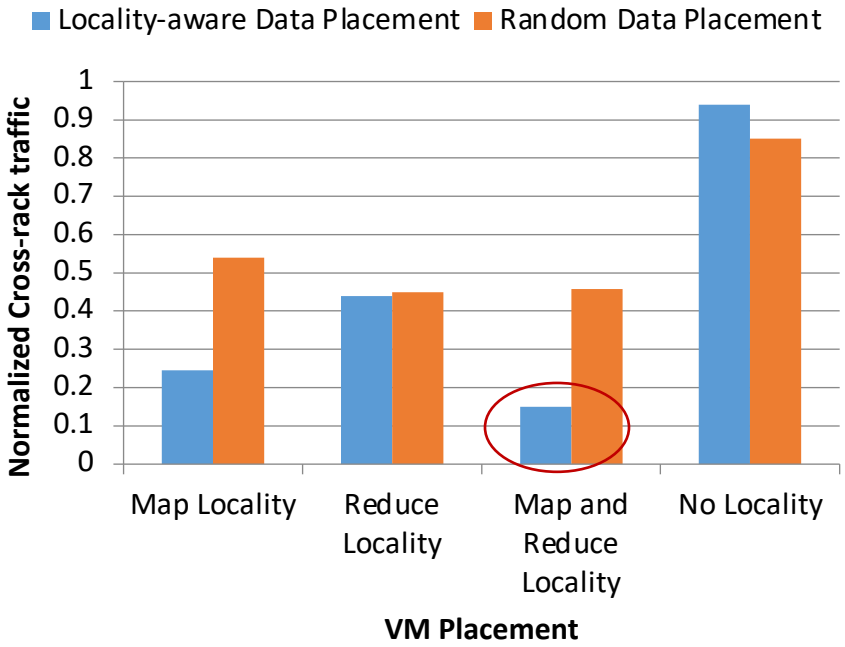


Fig 4. Cross-rack traffic

Performance depends on both Map and Reduce locality !

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

Reduce-input heavy workload

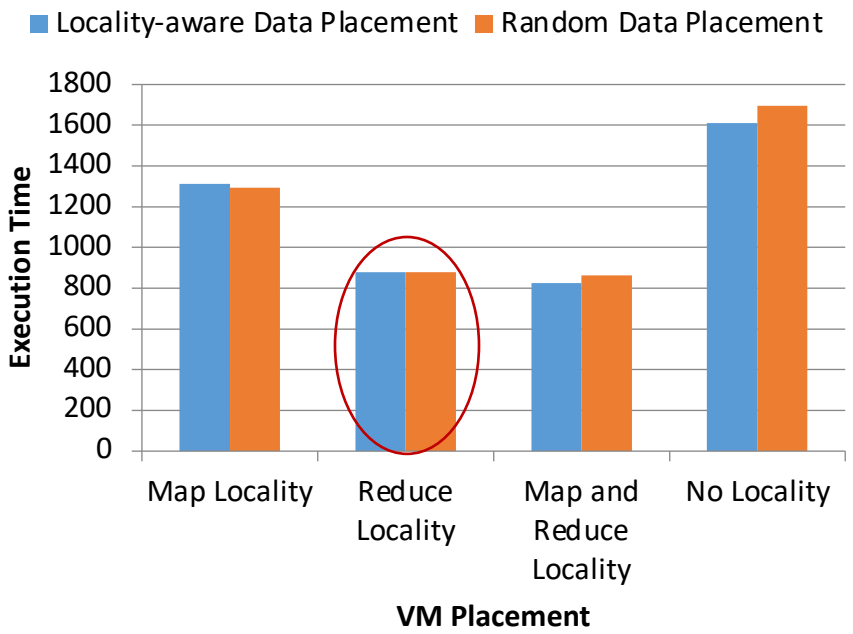


Fig 3. Execution time

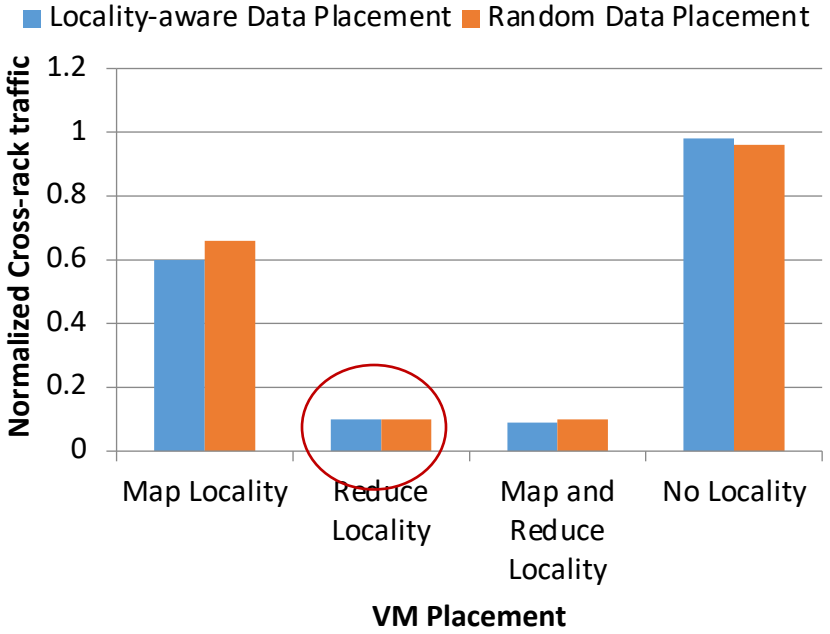


Fig 4. Cross-rack traffic

Techniques that optimize for reduce-locality perform well !

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

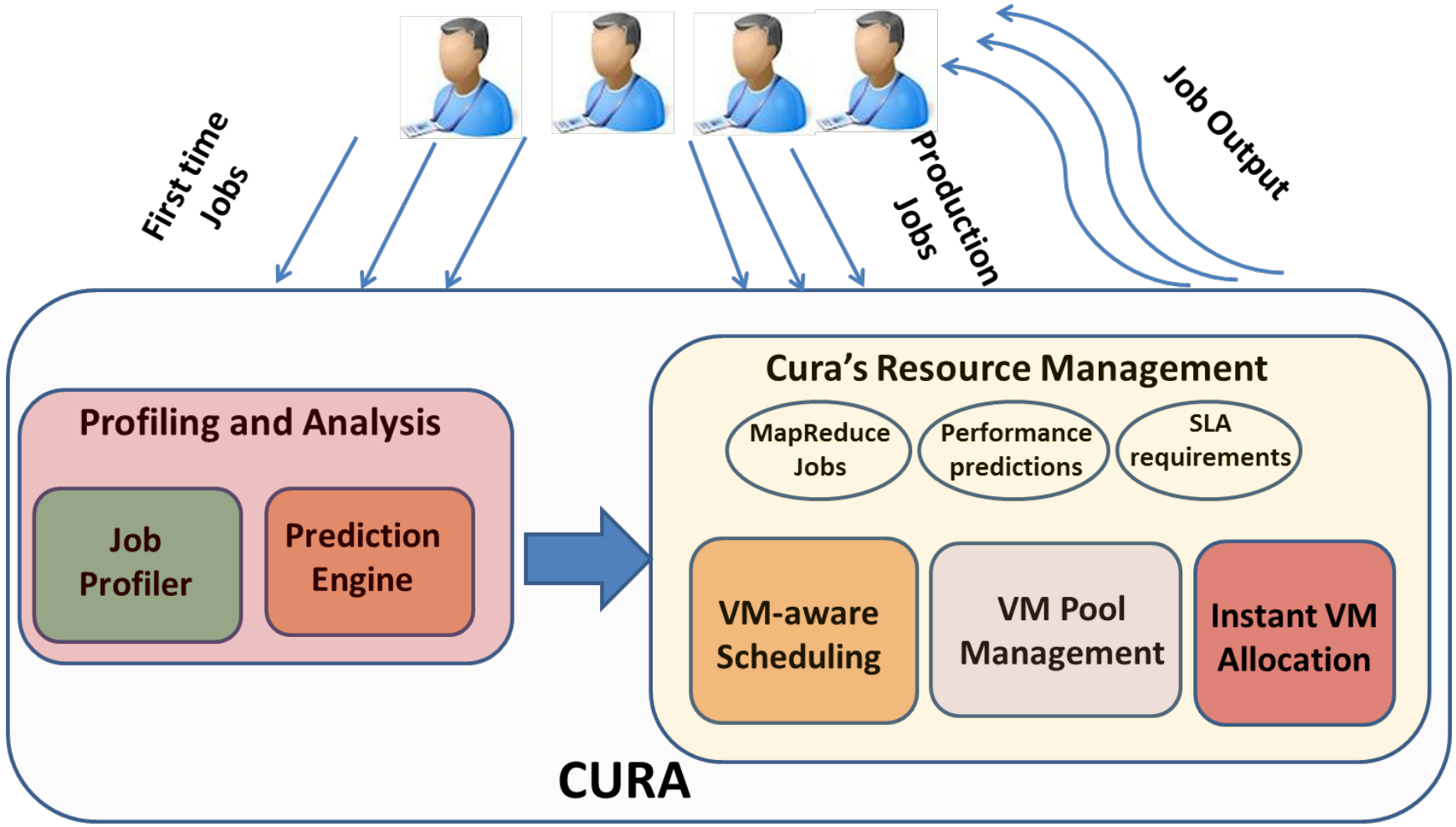
80 concurrent VMs

Cloud Scheduling

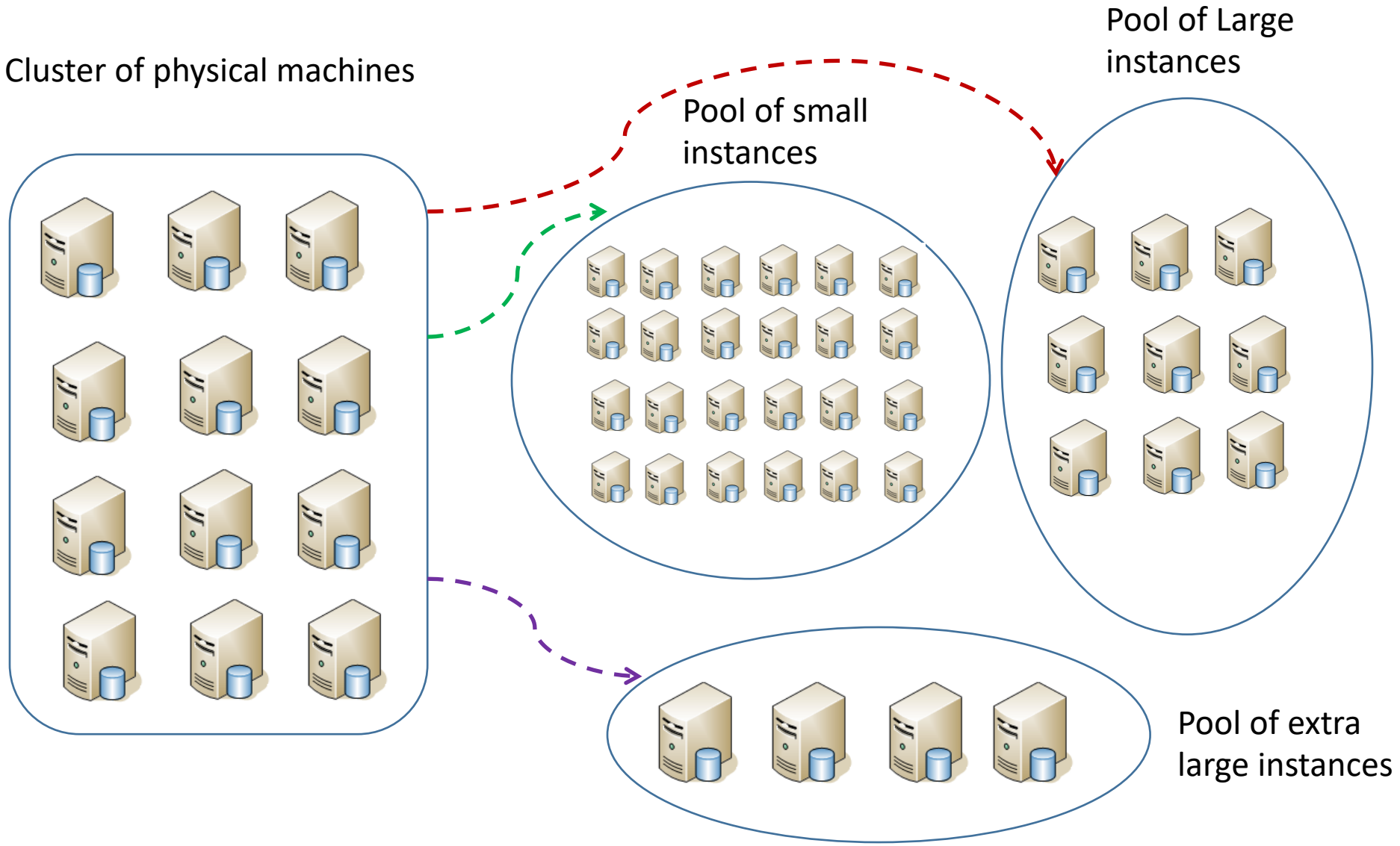
Job #	1	2	3	4
Start time	20	25	40	45

40 concurrent VMs

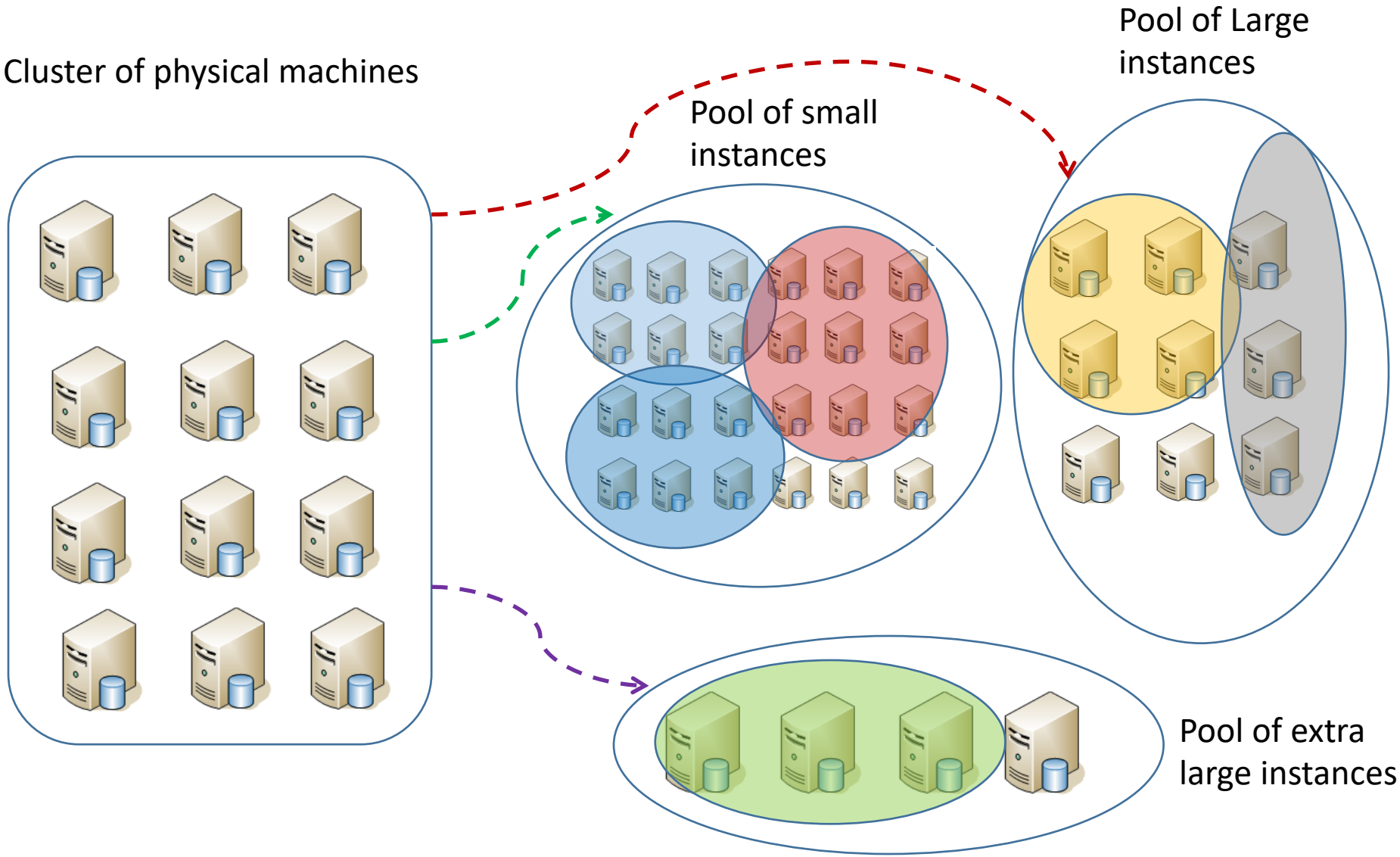
Cura System Architecture



Static Virtual Machine sets



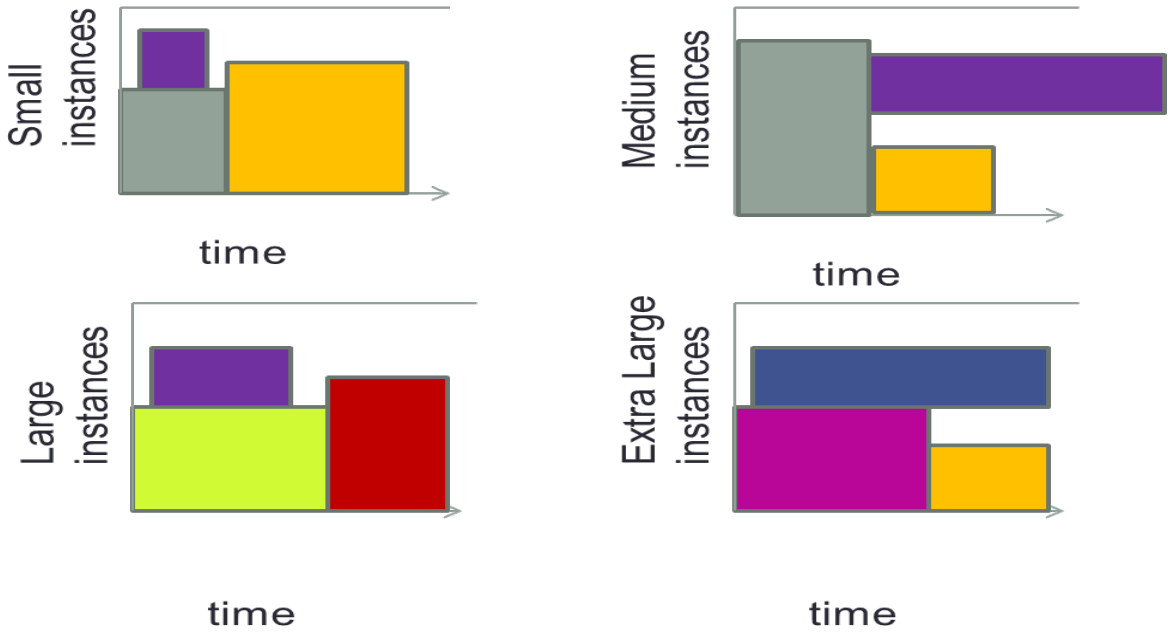
Static Partitioning of Virtual Machine sets



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

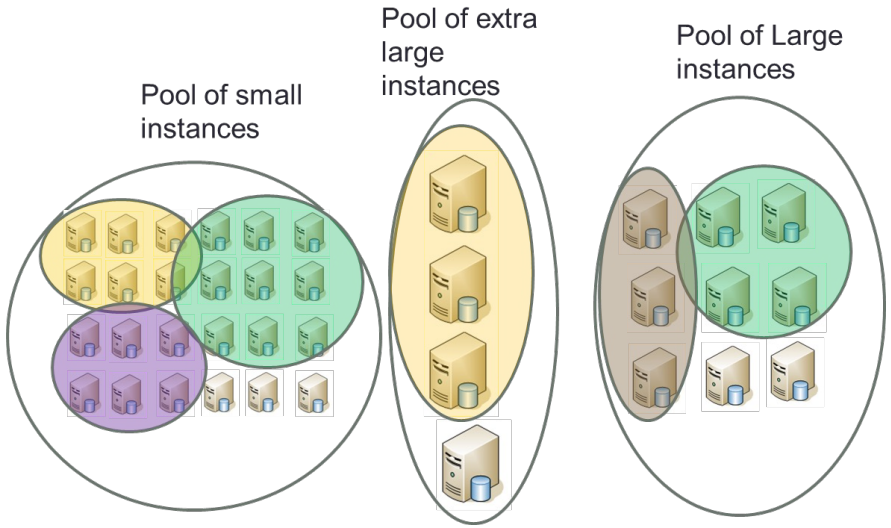
- Goal:
  - VM-aware scheduler decides (a) when to schedule each job in the job queue, (b) which VM instance pool to use and (c) how many VMs to use for the jobs.
- Minimum reservations without under-utilizing any resources.
- Job J_i has higher priority over Job J_j if the cost of scheduling J_i is 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

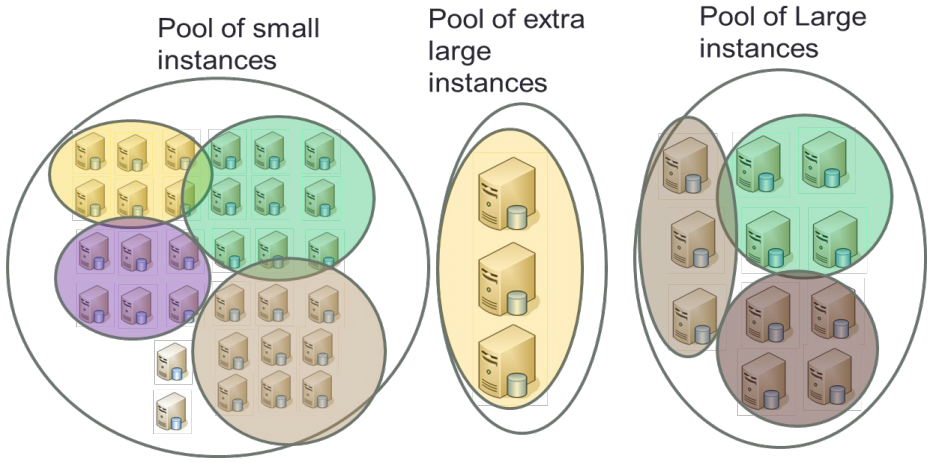
Assume two new jobs need a cluster of 9 small instances and 4 large instances respectively.

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

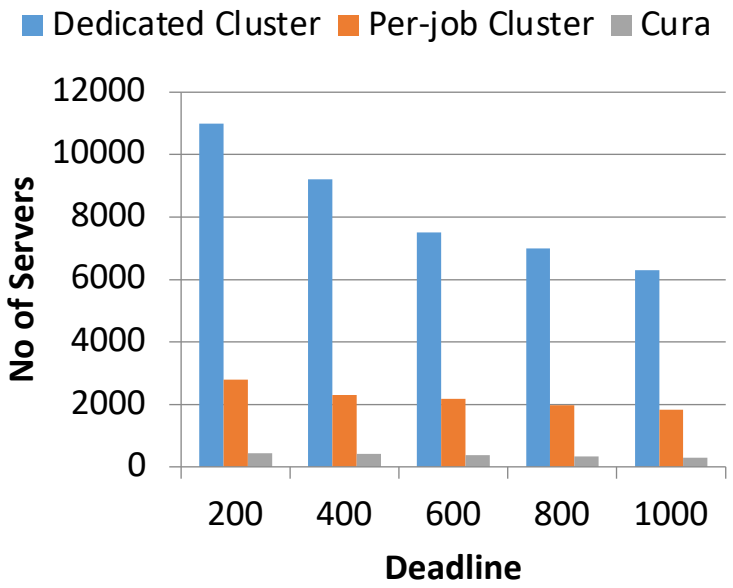


Fig 1. No. of Servers

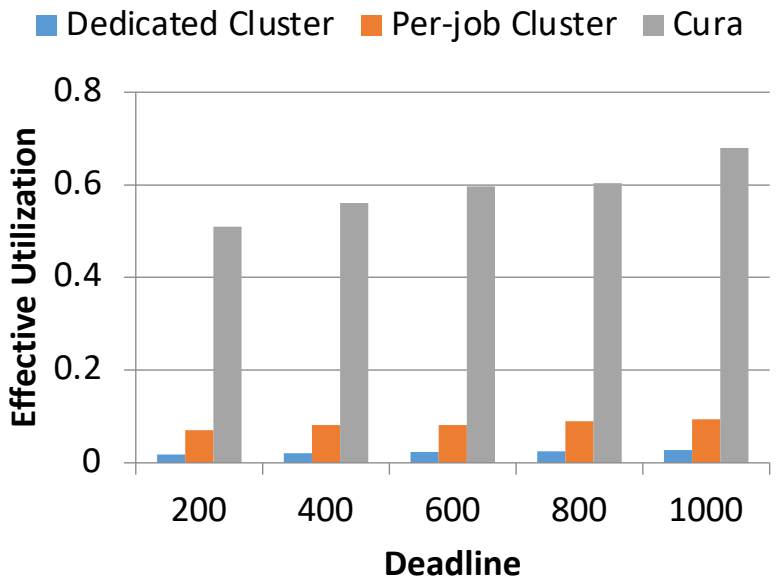


Fig 2. Effective Utilization

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

Response time and Cost

Fig 7. Response time

Fig 8. Cost

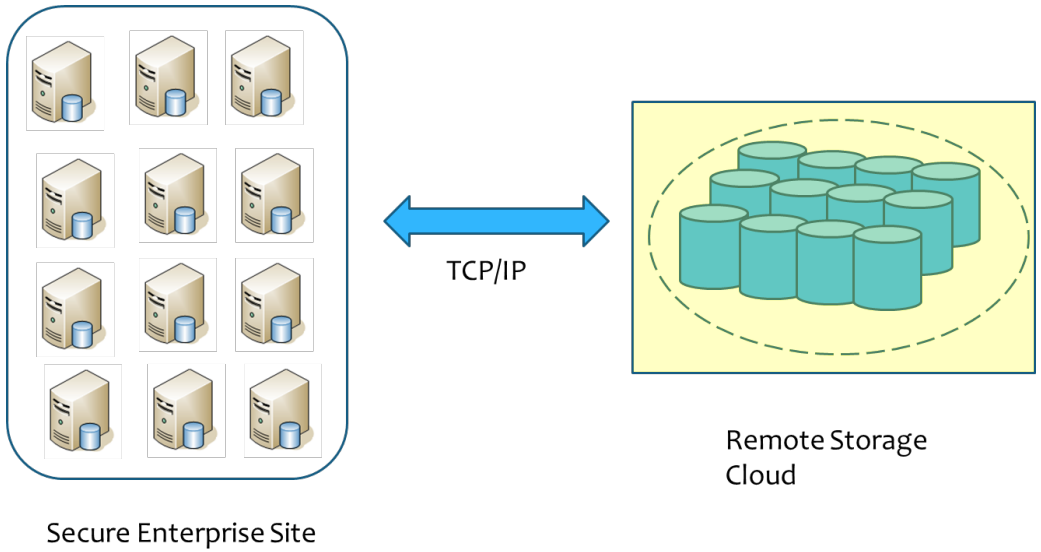
- With lower number of servers, Cura provides short response times
- Cura incurs much lower infrastructure cost

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.

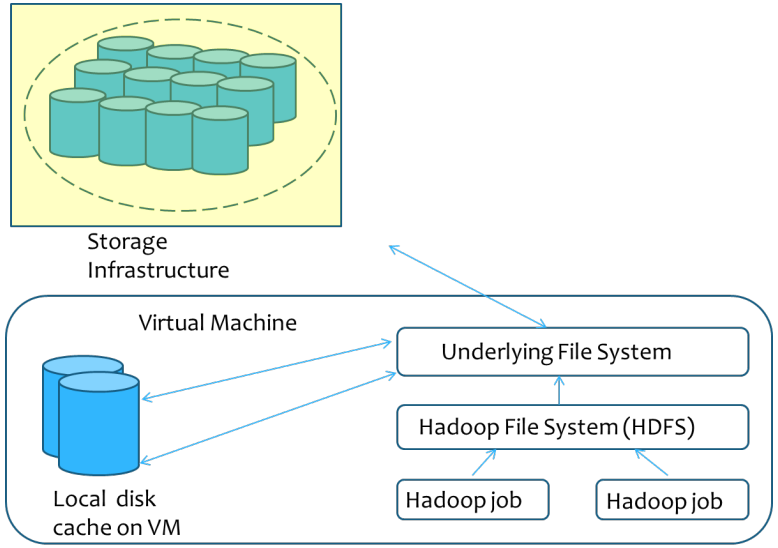
Current analytics solutions for Cloud-archived Data

- 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.

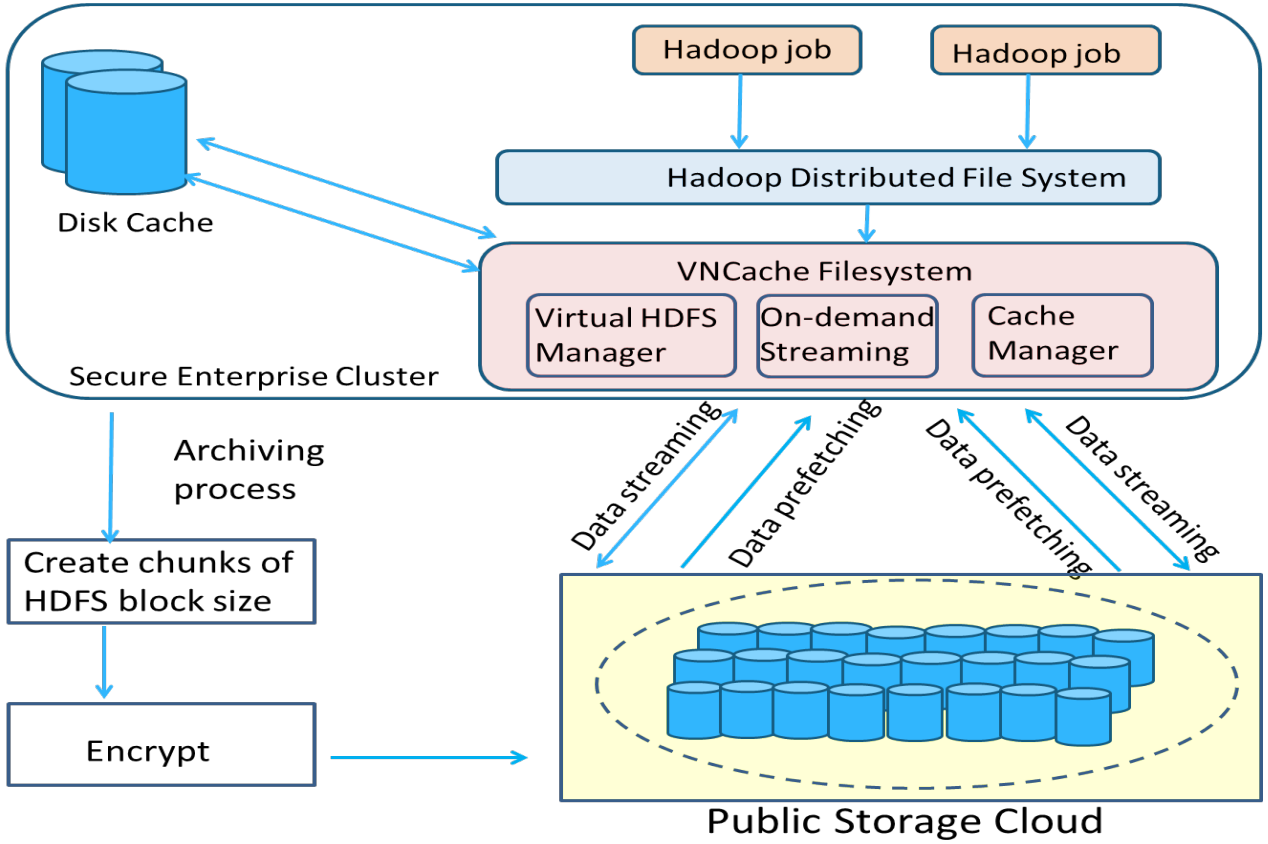
VNCache: Seamless Data Caching



- 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

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 block 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

Impact of Cache size – Grep Workload

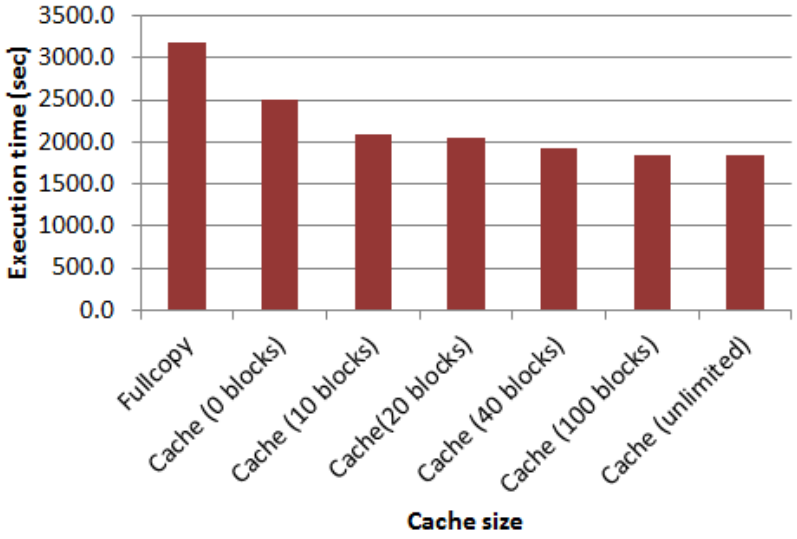


Fig 15. Execution time

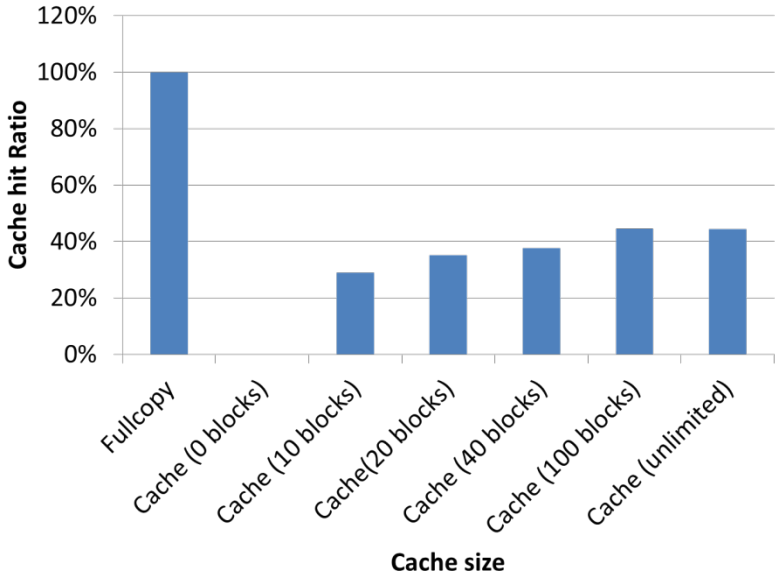


Fig 16. Cache Hit Ratio

Even with a reasonable cache size, VNCache achieves good performance