Cloud and DataCenter Systems: `Fast Data' -> Online Management

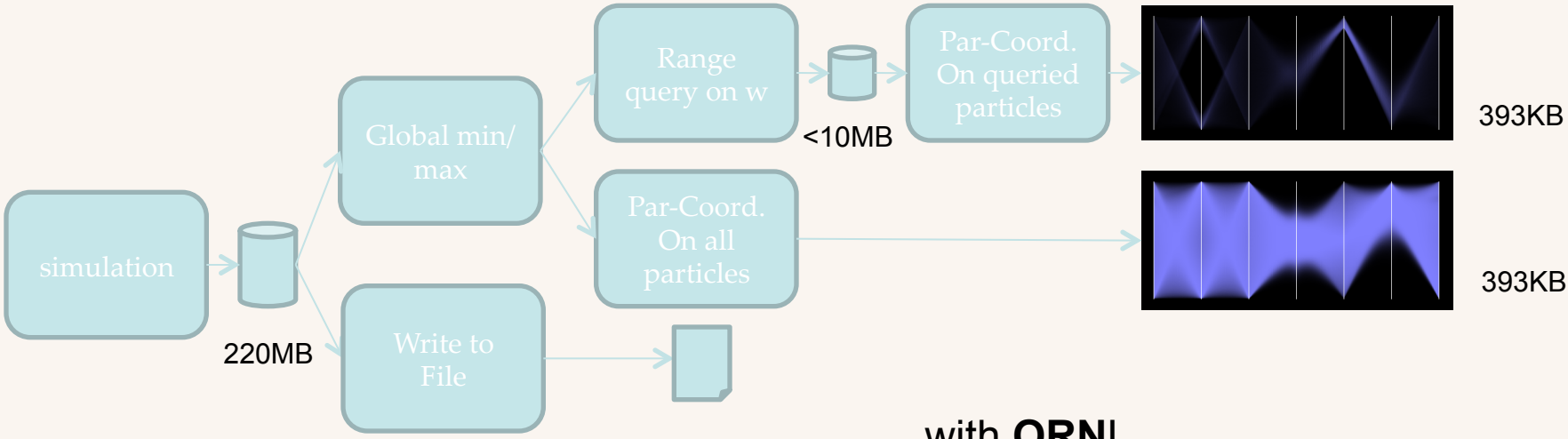
Karsten Schwan, Greg Eisenhauer, Ada Gavrilovska, Hrishikesh Amur, Liting Hu, Chengwei Wang, Junwei Li, …

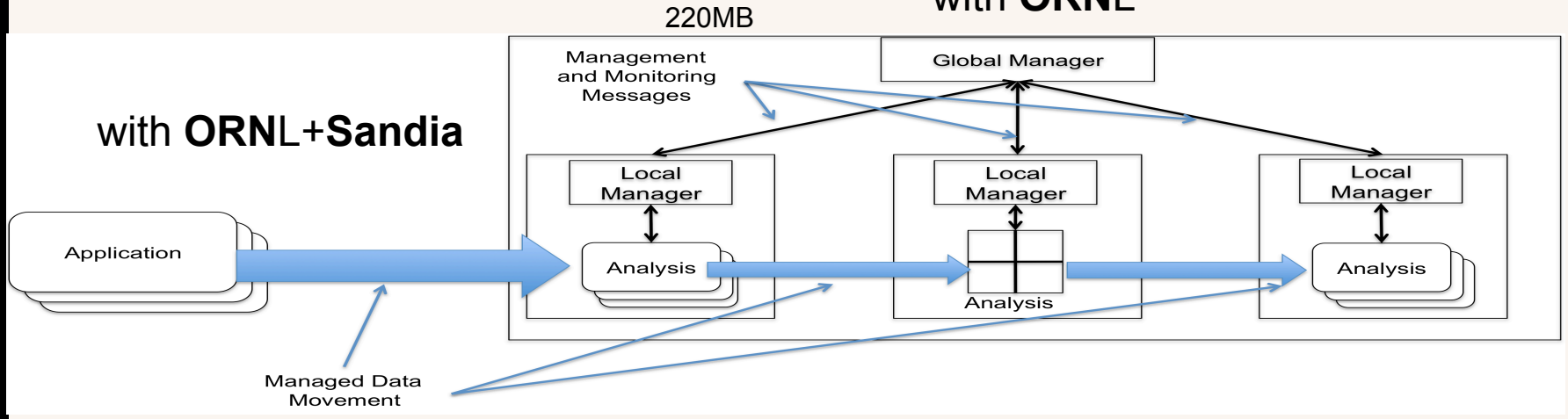
Motivation

- **Motivation**
  - Microsales, Product Bundling, Realtime Recommendations, …
  - Leveraging our stream processing research in the HPC (with **DOE**) domain to construct new methods for online web data processing
    - From Over-Provisioning to Active Stream Management
    - Avoiding the Storage Tier
    - Decentralized Operation for multiple stream processing jobs, with ‘cross-stream’ interaction

Background – Scalable Stream Processing for HPC Codes

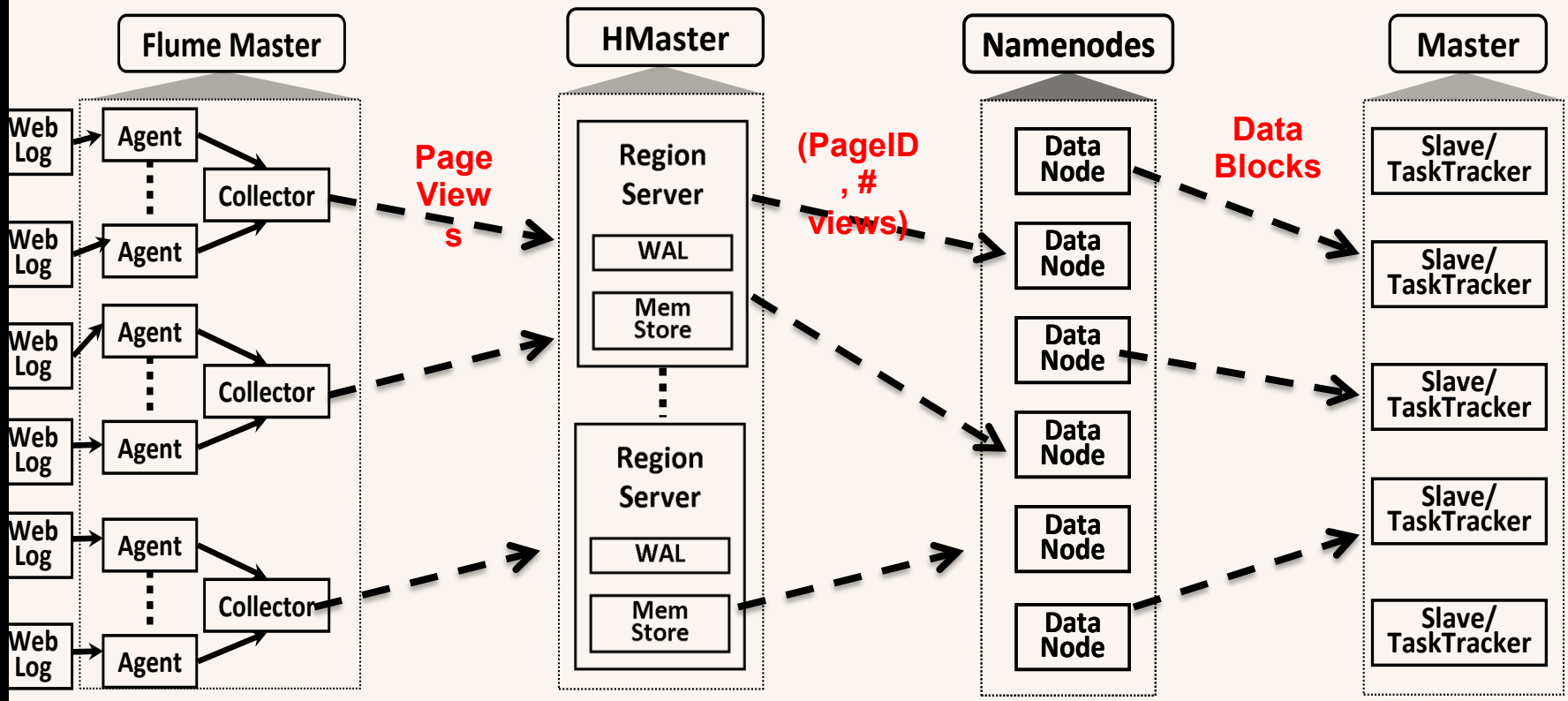
From Over-provisioned 'Staging' to actively 'Managed Runtimes'





Web Data Processing

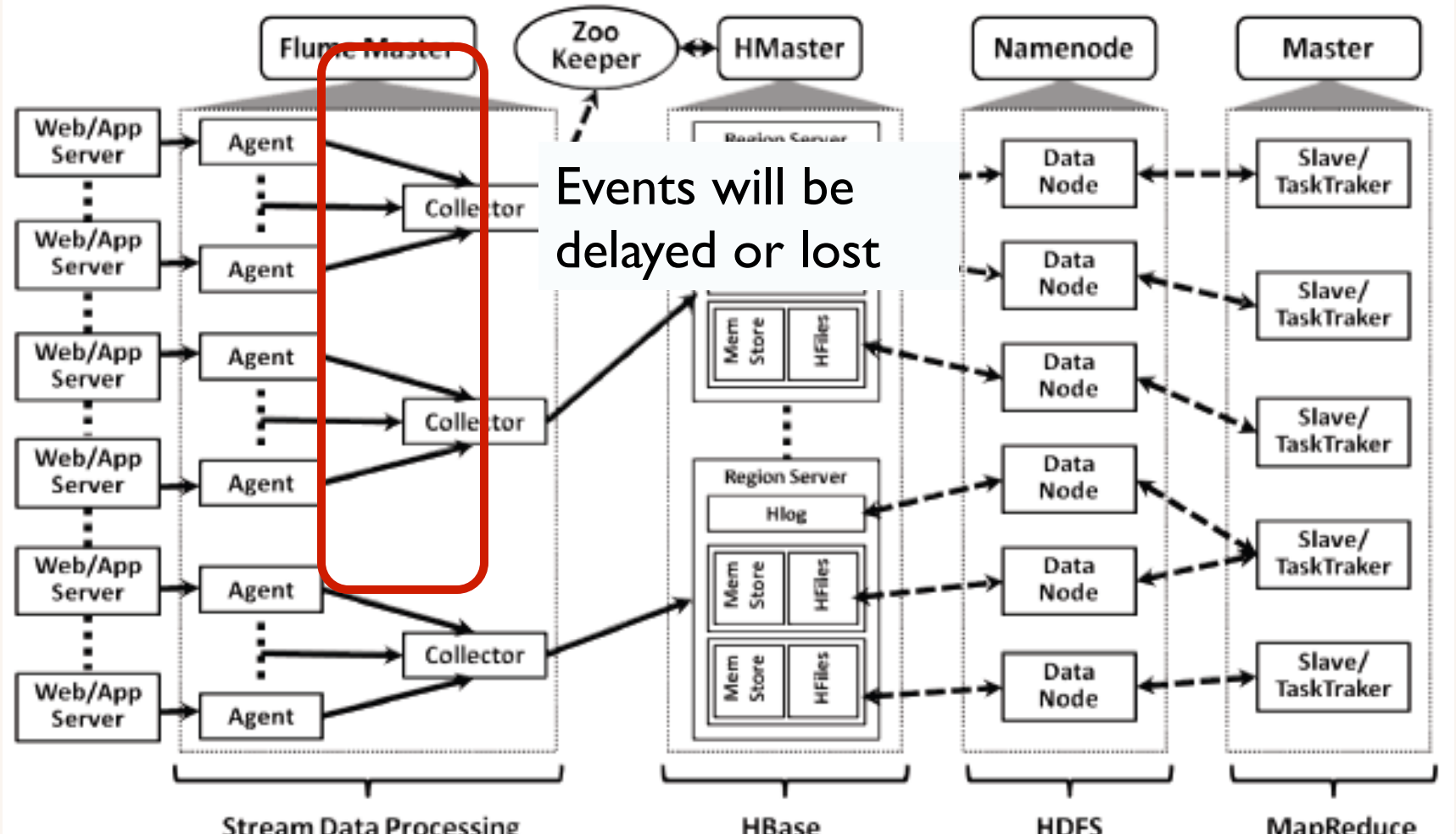
From over-provisioned log processing using the storage tier: (with help from **Amazon**)



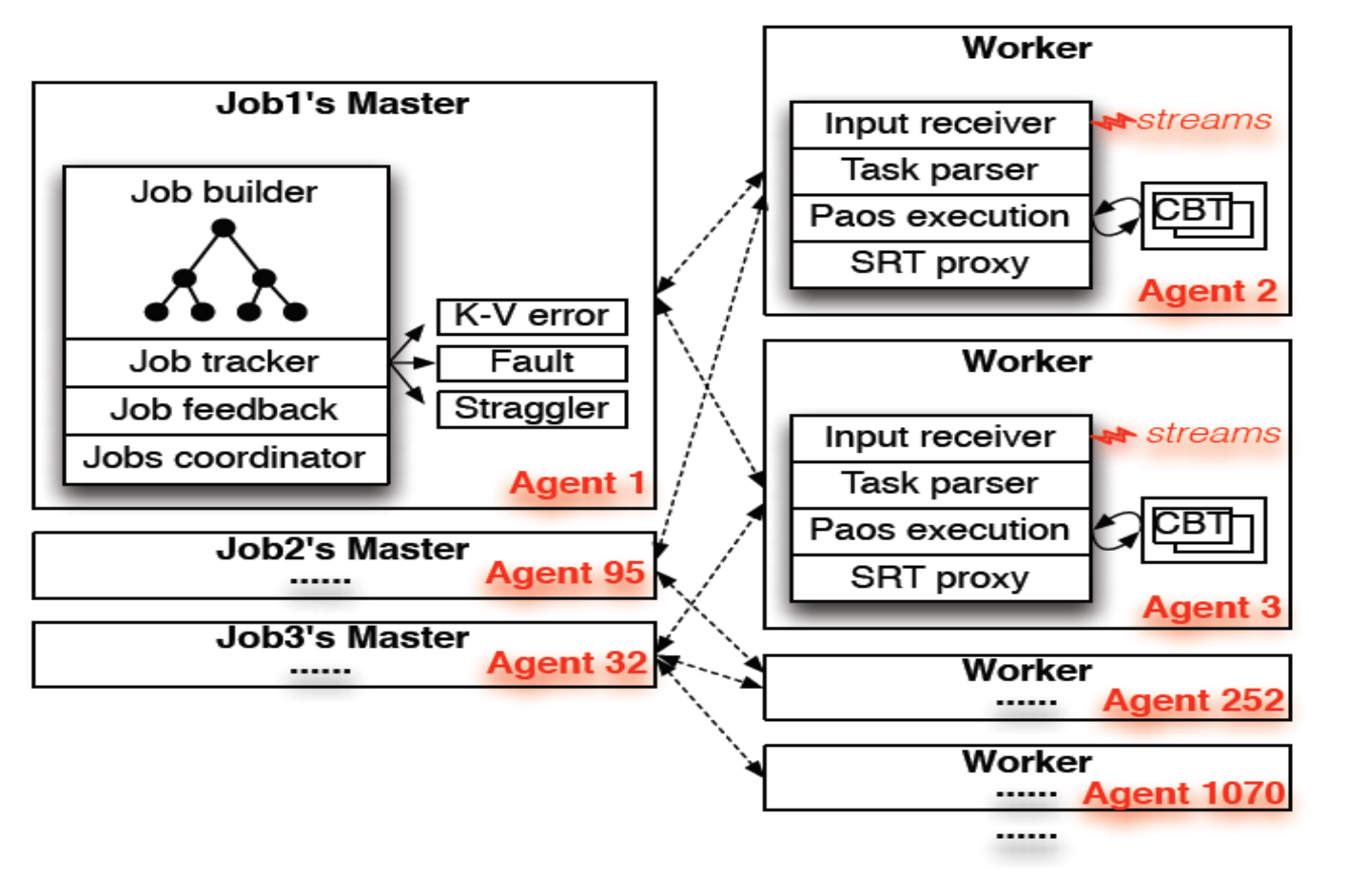
Web Data Processing

To managed streams:

If collectors can't keep pace with log events

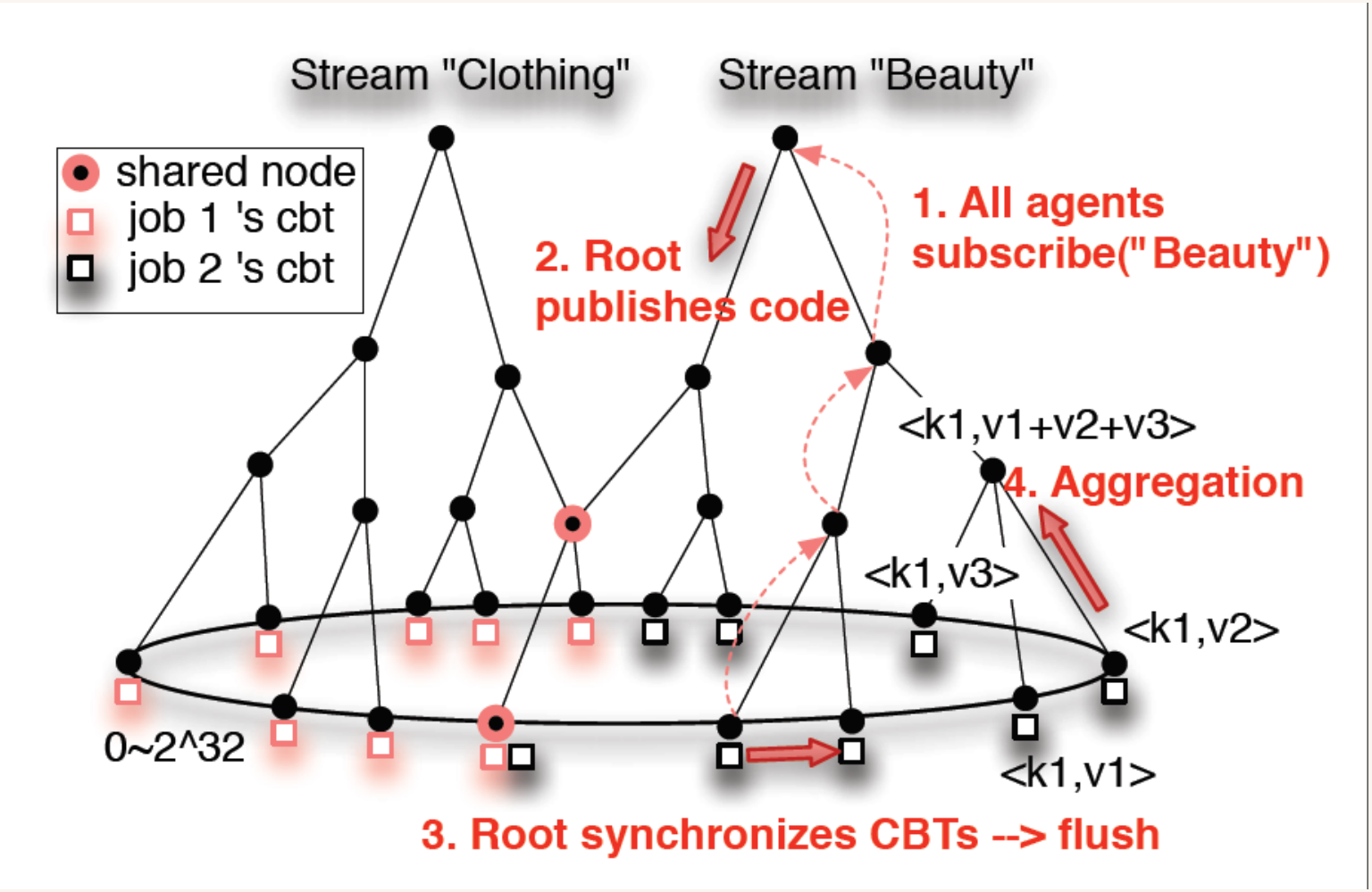


To Multi-Stream Execution
avoiding the Storage Tier

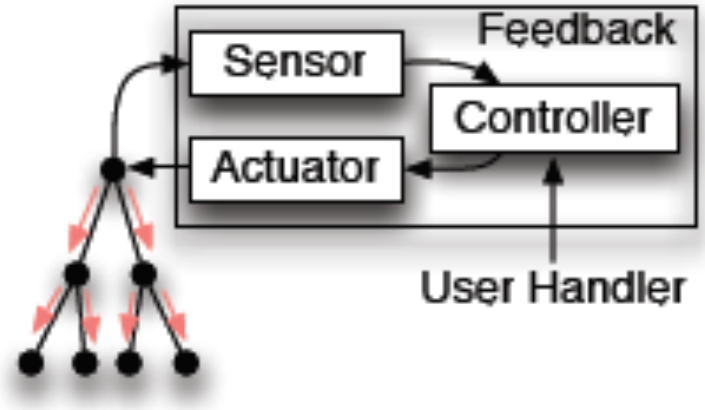


SRT-Shared Reducer Tree

per job logical SRT mapped onto shared tree nodes



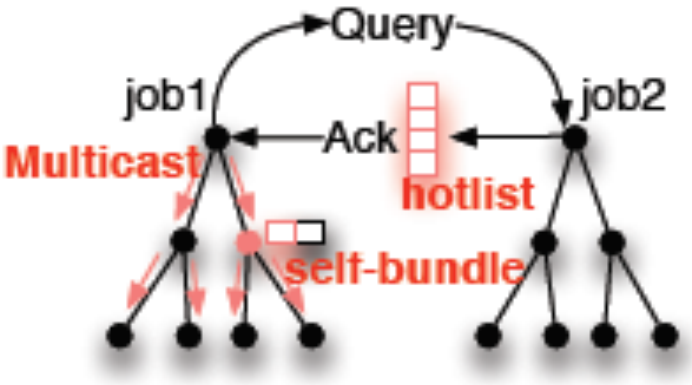
Feedback and Coordination



CouponExecutor:

sensor(jobId, key, value)
//fetch K-V pairs from root of SRT
controller(jobId, handler)
//register user-specified handler
actuator(jobId)
//generate coupons to nodes

Figure 5: Feedback abstraction for **CouponExecutor**.



ProductBundling:

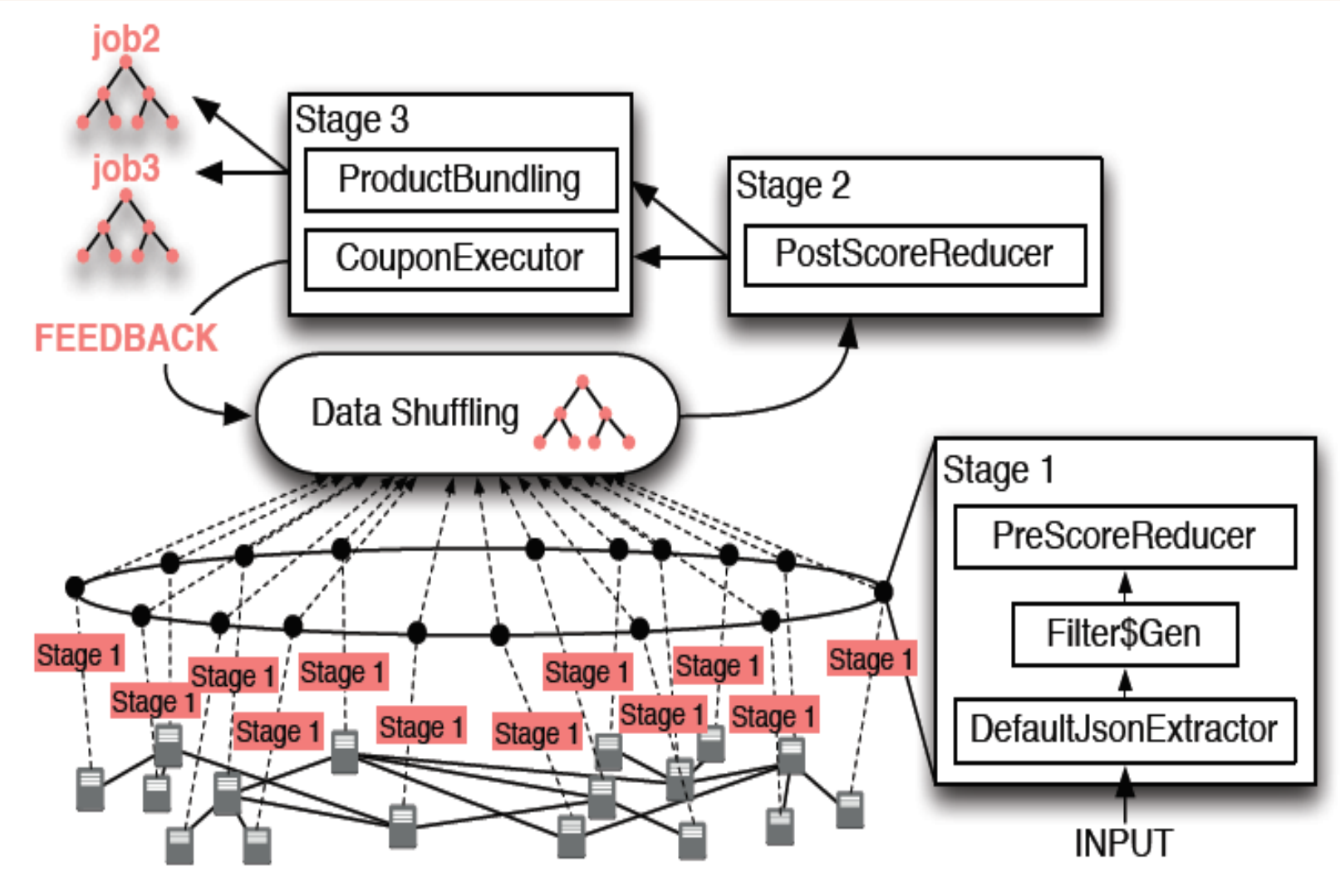
job1.anycast(job2, hotlist)
//job1 query job2's hot items
job1.multicast(job1, hotlist)
//multicast job2's hot items to members
agent.selfbundling(hotitem)
//agent decides by itself which to bundle

Figure 6: Coordination abstraction for **ProductBundling**.

Summary: Online Stream Processing

- Upstream buffering to avoid the storage tier: in CBTs (with **CMU**)
- Logical SRTs for each stream job, sharing a single set of 'tree nodes'
- Cross-SRT interactions, including feedback-coordination and on-the-fly function changes

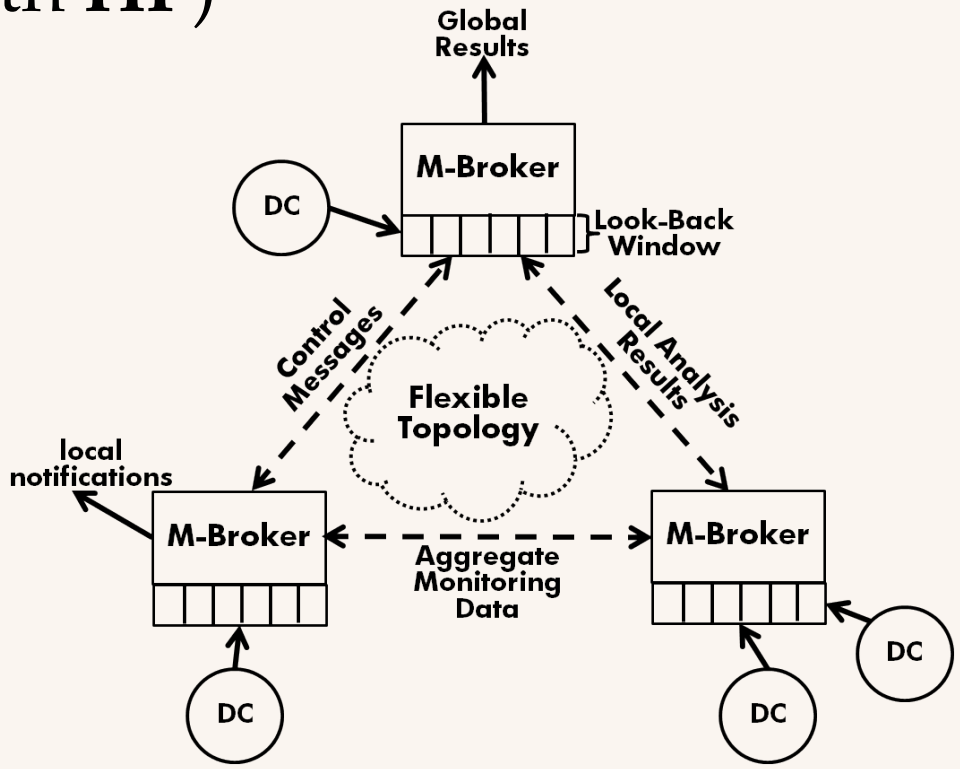
Software Architecture - Summary



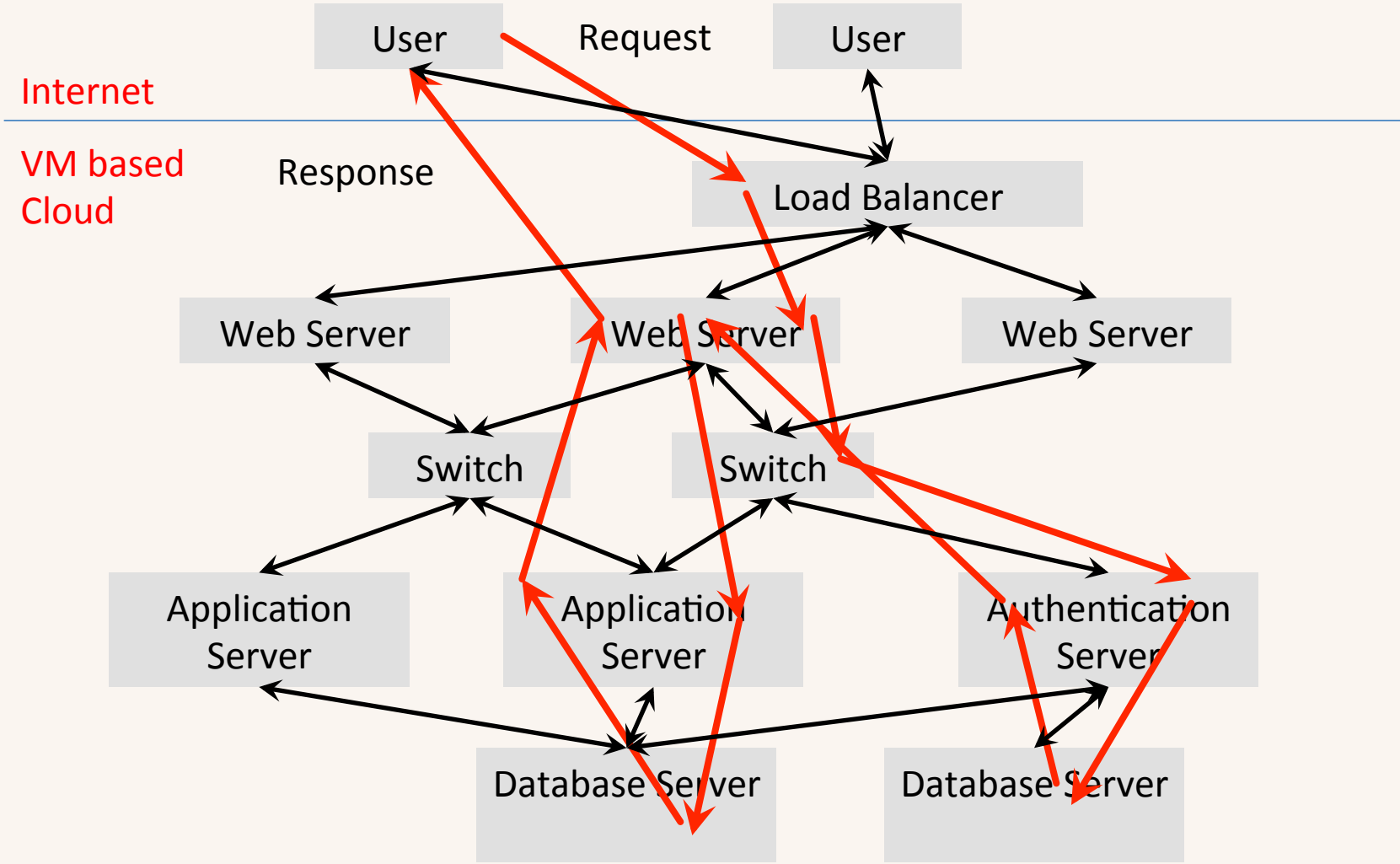
Online Management: Monitoring as a Prerequisite

Building Blocks: Stream Processing with:

- Distributed Processing Graphs (DPGs) (with **HP**)



Building Blocks:
To Determine Dynamic Interaction Graphs



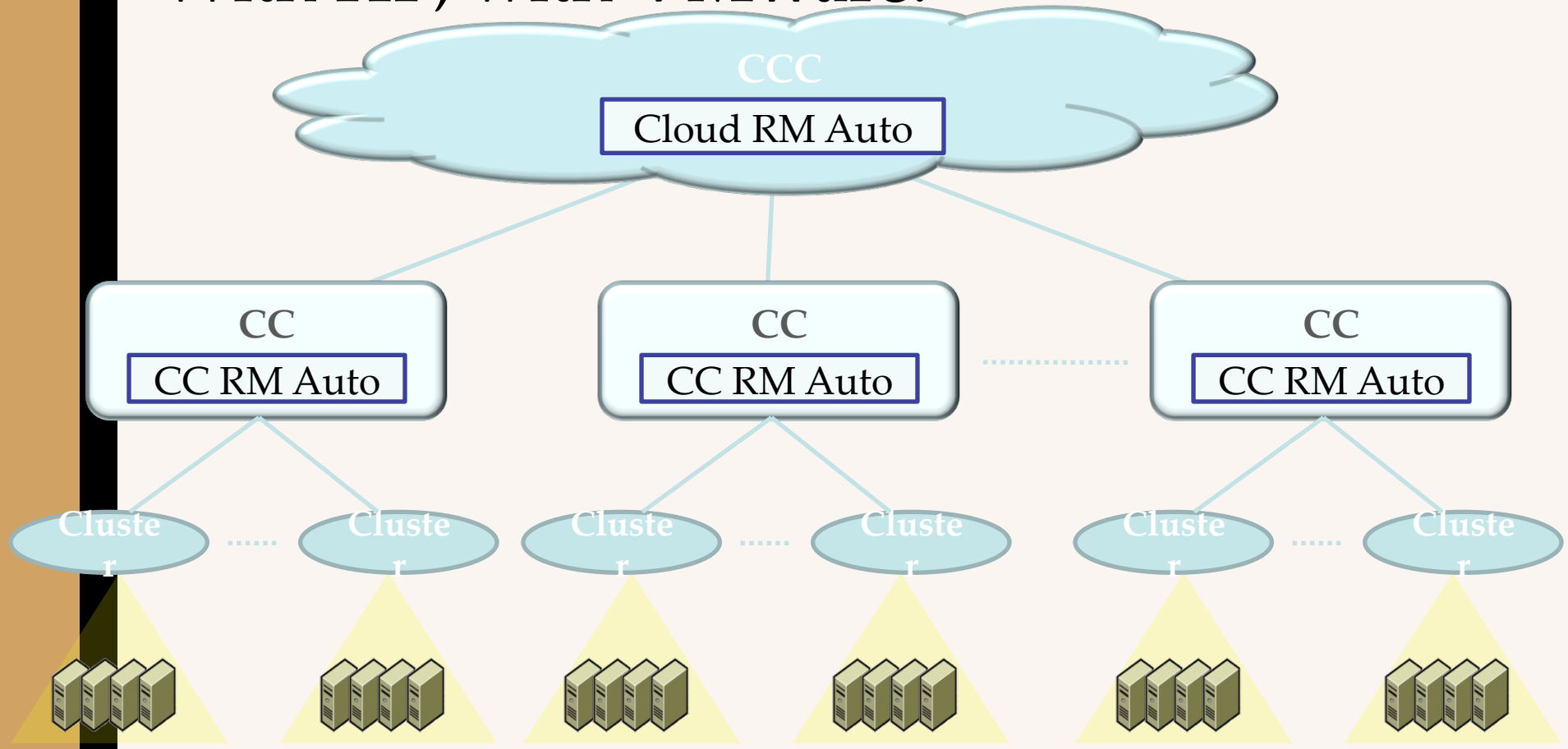
Issue: How to find DIGs? Can be easy (e.g., hypervisor monitoring for VM location)
Or hard (with **ATT**): dynamic call tracing using blackbox methods
Or approximate (with **HP**): statistical techniques

Building Blocks: Analytics

- Rich set of methods (many from scientific domain, others from data mining and machine learning)
- We typically ‘use’ rather than innovate, driven by realistic use cases (e.g., troubleshooting) (with **Amazon**)

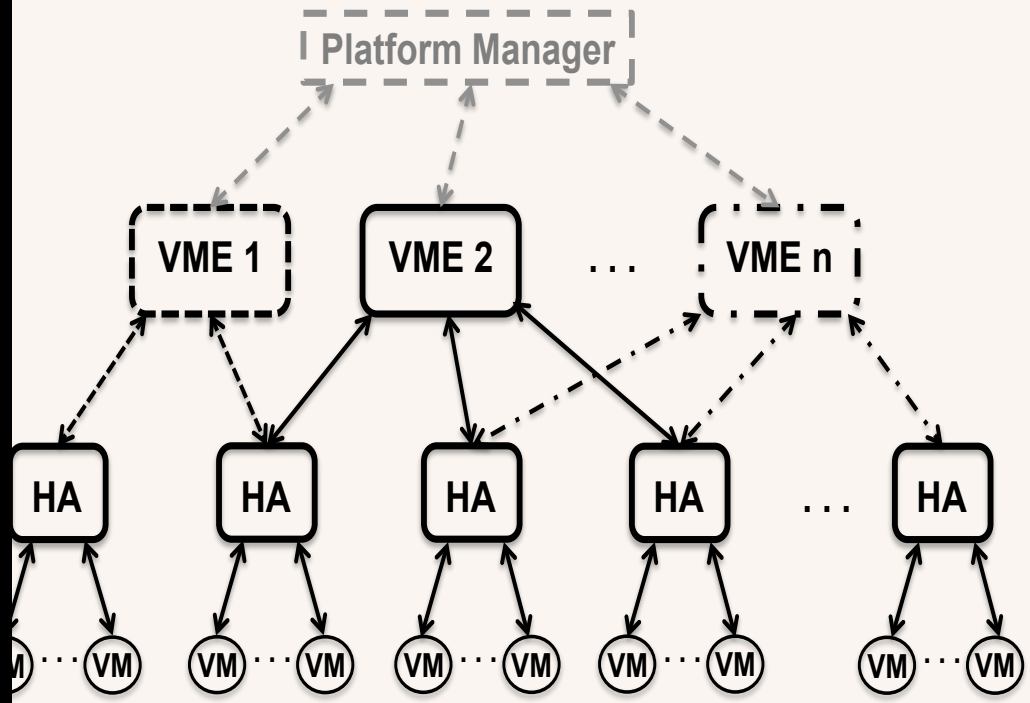
Managing at Scale

With **HP**, with **VMware**:



RM scalability achieved by two additional levels of automated load balancing

Retain High Performance
(inspired by **ICE**)



Enable Distributed Coordinated Resource Management using Resource Pricing per VME

- **Distributed Resource Exchange (DRX) Framew**
- **Platform Manager (PM)**
  - Cluster-wide policy enforc
  - Allocate Resos
  - Calculate Resource Price/V
  - Distributes updated price using ZeroMQ Library
- **Ensemble Manager (EM)**
  - Distribute Resos for VMs
  - Monitor VME performance
  - Report to PM
- **Host Agent (HA)**
  - Contains **IBMon**
  - Manages VMs Resources
  - Deducts Resos for VMs
  - Perform CPU Capping

Conclusions and Future Work

Fast Data:

- Should avoid the storage tier
  - But what about NVM? Ongoing Work (with **Intel**, **HP, DOE**)
- Operate in a decentralized fashion
- Efficiently use shared resources (staging, SRTs, …)
- Requires online management
  - Monitoring and analytics
  - Feedback and coordination
  - Runtime change