An Autonomic Computing Engine for Computational Science and Engineering Applications in Cloud Computing and Grid Environments

Moustafa AbdelBaky, Ivan Rodero, Hyunjoo Kim and Manish Parashar

moustafa, irodero, hyunjoo, parashar@cac.rutgers.edu

The NSF Cloud and Autonomic Computing Center

Electrical & Computer Engineering Department

Rutgers, The State University of New Jersey, USA

cac

NSF

RUTGERS

THE STATE UNIVERSITY OF NEW JERSEY

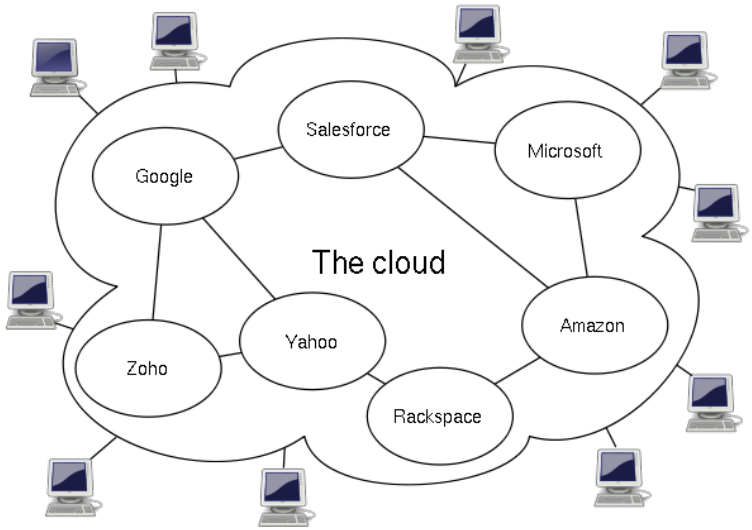
Outline

- Cloud Computing for CDS&E Applications
- Introduction to CometCloud (http://cometcloud.org)
  - Concept
  - Architecture & Autonomics
  - Programming models
  - Cloud Agents
  - Cloud bursting
- Break
- Hands on session
  - Matrix Multiplication using Master/Worker
  - Overview of Map/Reduce

Introduction

Defining Cloud Computing

- Wikipedia – Cloud computing is Internet-based computing, whereby shared resources, software and information are provided to computers and other devices on-demand like a public utility.
- NIST – A cloud is a computing capability that provides an abstraction between the computing resource and its underlying technical architecture (e.g., servers, storage, networks), enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction



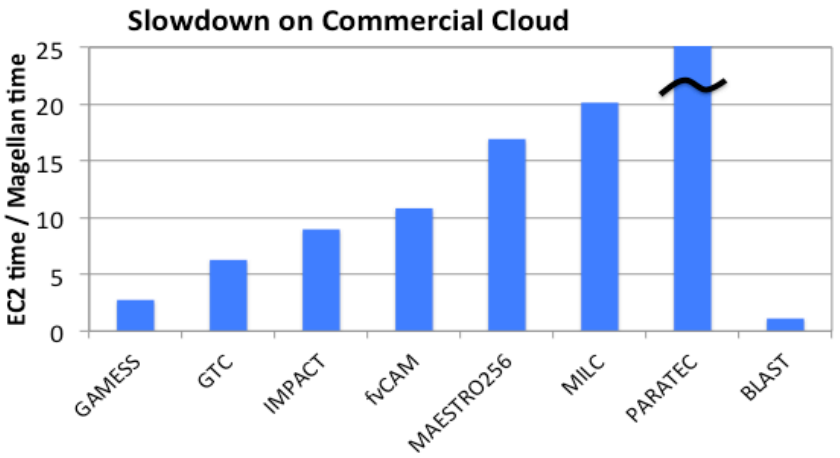
- SLAs
- Web Services
- Virtualization

Clouds and Science

- Clouds are rapidly joining traditional CI as viable platforms for scientific exploration and discovery
  - What application formulations and usage modes that are meaningful in such a hybrid infrastructure?
  - How can application workflows effectively utilize it?
- Possible usage modes:
  - Clouds can simplify the deployment of applications and the management of their execution, improve their efficiency, effectiveness and/or productivity, and provide more attractive cost/performance ratios
  - Cloud abstractions can support new classes of algorithms and enable new applications formulations
  - Democratization
  - Application driven by the science, not available resources!
    - Cloud abstractions for science?
- Many challenges
  - Application types and capabilities that can be supported by clouds?
  - Can the addition of clouds enable scientific applications and usage modes that are not possible otherwise?
  - What abstractions and systems are essential to support these advanced applications on different hybrid platforms?

HPC in the Cloud

- Run applications on commodity clouds (e.g.,EC2)
  - Quick start-up, easy to use
  - Good performance for specific application classes
- However not suited for many applications
- And costs can add up quickly……
- For example, at NERSC*
  - Baseline Cloud is 8x more expensive
  - Effective performance of the cloud is a minimum of 16x less cost effective (and worst case 320x less effective) for DOE midrange HPC workloads



* HPC in the Clouds, John Shalf Feb 3, 2011

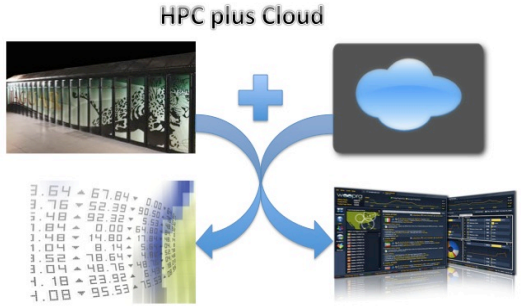
Beyond the Obvious Candidates

- New application formulations
  - Asynchronous, resilient
  - E.g., Asynchronous Replica Exchange Molecular Dynamics, Asynchronous Iterations
- New usage modes
  - Client + Cloud accelerators
  - E.g., Excel + EC2
- New hybrid usage modes
  - Cloud + HPC + Grid

Integrating Cloud & CI

- ***HPC in the Cloud***, where researchers outsource entire applications to current public and/or private Cloud platforms.

- ***HPC plus Cloud***, focused on exploring scenarios where Clouds can complement HPC/Grid resources with Cloud services to support science and engineering application workflows, for example, to support heterogeneous requirements, unexpected spikes in demand, etc.



- ***HPC as a Cloud***, focused on exposing HPC/Grid resources using elastic on-demand Cloud abstractions, aiming to combine the flexibility of Cloud models with the performance of HPC systems.

Summary

- The future is *Cloudy*…
  - Cloud becoming a part of production computational environments
  - Clouds will play a role in Science and Engineering
  - Many Cloud Computing Benefits:
    - Shift CapEx to OpEx , Scale OpEx to demand
    - Startups and prototyping, One-off tasks (Wash. Post)
    - Cost associativity
    - Research at scale
- Clouds transforming science
  - New applications, new delivery models, new usage modes, democratization
- A challenging research agenda
  - Not just the Science of Clouds but also ***Science on Clouds***

An Introduction to

COMETCLOUD

Motivation

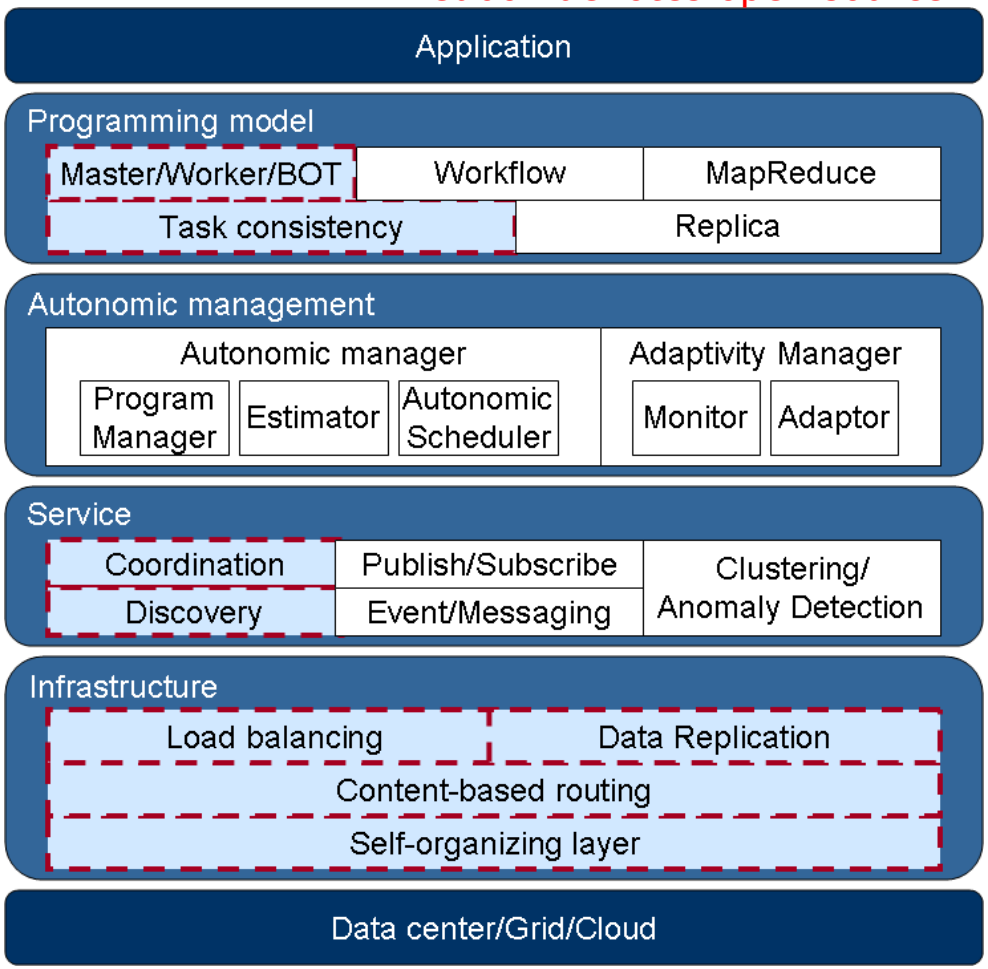
- Highly dynamic demands for resources with heterogeneous and dynamic workloads
  - Number of tasks, computational requirements
- Various and dynamic QoS requirement
  - Throughput, budget, time, etc.
- "Rent" services of clouds
  - Rent required resources from clouds on-demand and pay for what you use
  - Amazon EC2, Microsoft Azure, GoGrid, etc.
- So provisioning an appropriate mix of resources for applications is non-trivial.

CometCloud – Key Features

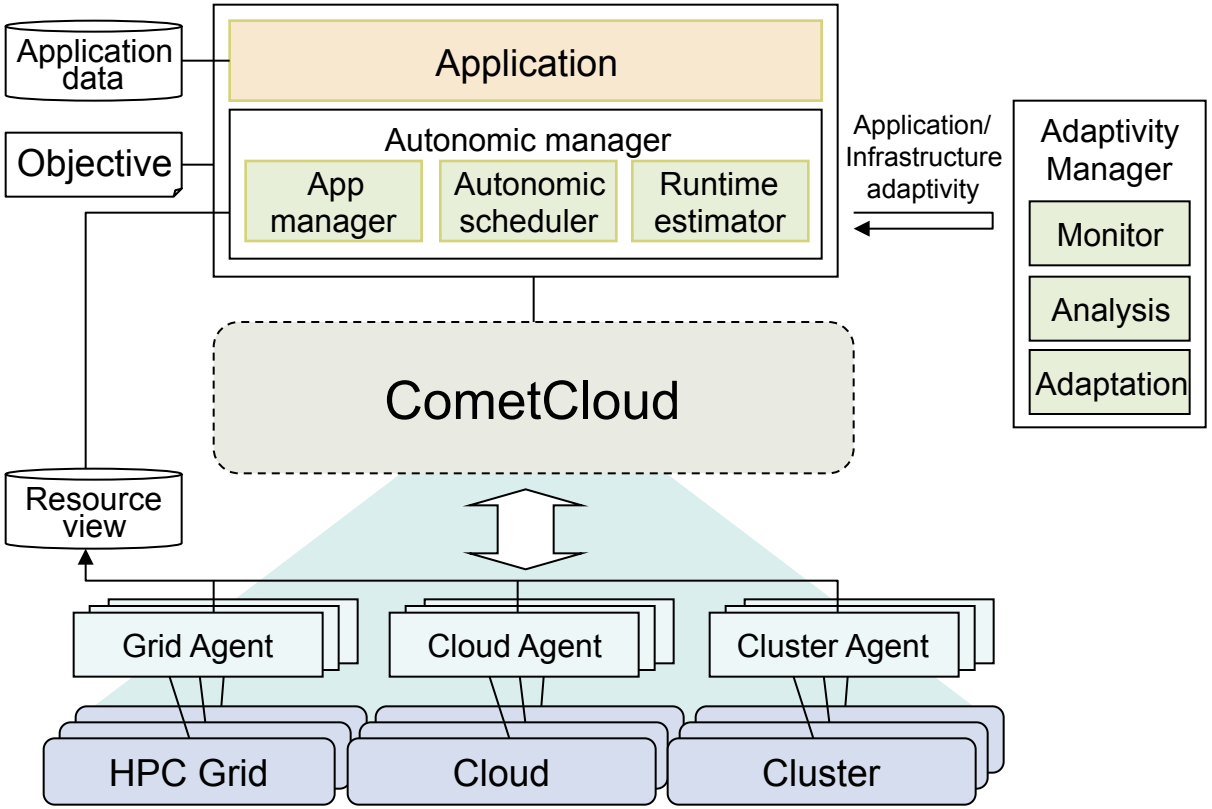
- Builds a federated cloud consolidating clouds and grids.
  - *Cloud-bursting*: dynamic application scale-out/up to address dynamic workloads, spikes in demand, and other extreme requirements
  - *Cloud-bridging*: on-the-fly integration of different resource classes
- Provides policy-driven autonomic resource provisioning from the federated clouds.
  - User's policy (deadline, budget, etc), resource constraints (failure, network, availability, etc)
- Provides programming abstractions to develop applications and deploy them on the federated clouds
  - Master/worker, MapReduce/Hadoop, Workflow

CometCloud

Redbox denotes open source.



Autonomics in CometCloud



- **Autonomic manager** manages workflows, benchmarks application and provision resources.
- **Adaptivity manager** monitors application performance and adjusts resource provisioning.
- **Grid/Cloud/Cluster agent** manages local cloud resources, accesses task tuples from CometCloud and gathers results from local workers so as to send them to the workflow (or application) manager.

Autonomics in CometCloud

- **Application/Programming layer autonomics:** Dynamic workflows; Policy based component/ service adaptations and compositions
- **Autonomics layer:** Resource provisioning based on user objectives; estimation of resource requirement initially, monitor application performance, and adjust resource provisioning.
- **Service layer autonomics:** Robust monitoring and proactive self-management; dynamic application/ system/context-sensitive adaptations
- **Infrastructure layer autonomics:** On-demand scale-out; resilient to failure and data loss; handle dynamic joins/departures; support “trust” boundaries

E.g., User Objectives

- Acceleration
  - This use case explores how Clouds can be used as accelerators to improve the application time to completion by, for example, using Cloud resources to alleviate the impact of queue wait times or exploit an additionally level of parallelism by offloading appropriate tasks to Cloud resources, given appropriate budget constraints.
- Conservation
  - This use case investigates how Clouds can be used to conserve HPC Grid allocations, given appropriate runtime and budget constraints.
- Resilience
  - This use case will investigate how Clouds can be used to handle unexpected situations such as an unanticipated HPC Grid downtime, inadequate allocations or unanticipated queue delays.

E.g., Constraints

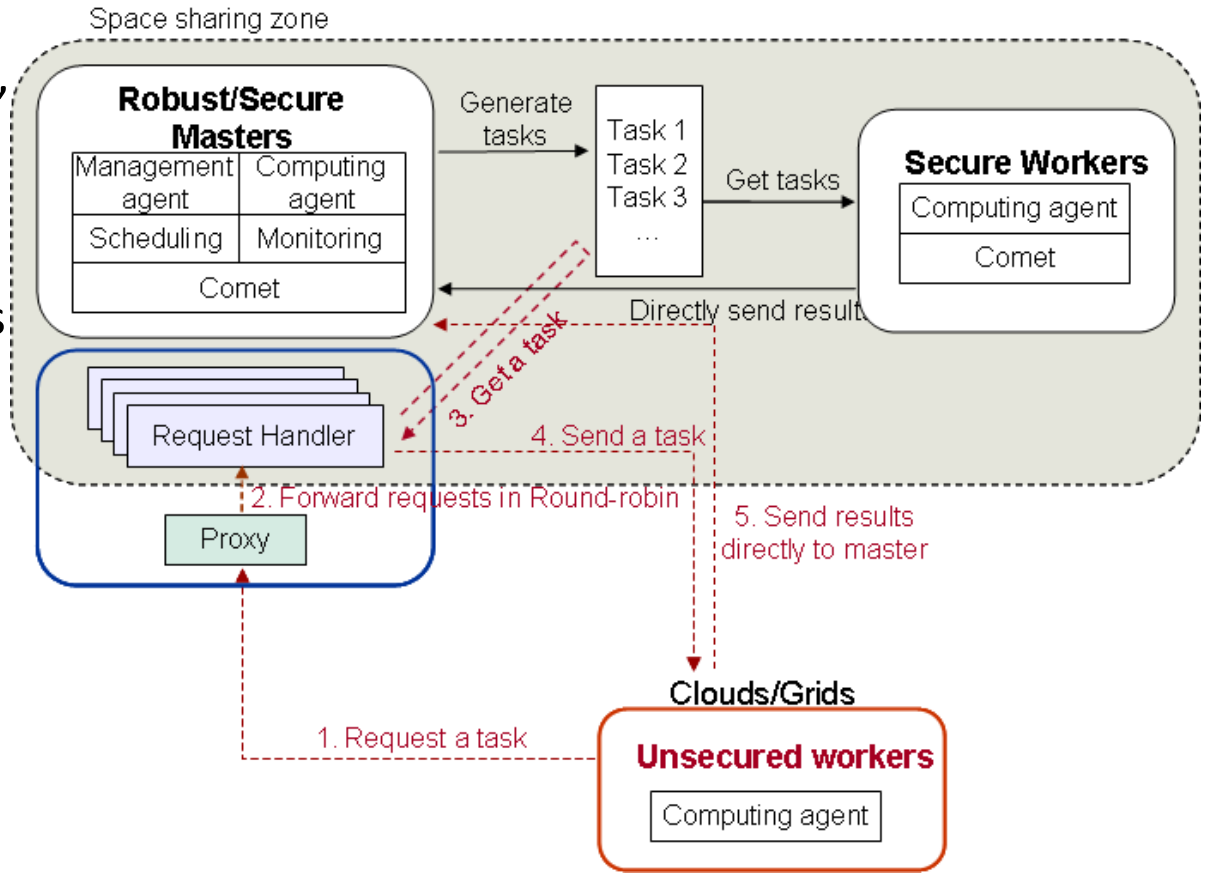
- Deadline
  - Time constraint to complete an application
  - To select the fastest resource class for each task and to decide the number of nodes per resource class based on the deadline.
- Budget
  - Budget constraint to complete an application
  - When a budget is enforced on the application, the number of allocable nodes is restricted by the budget.
- Economical deadline
  - Resource class can be defined as the cheaper but slower resource class that can be allocated to save cost unless the deadline is violated.

CometCloud

PROGRAMMING MODELS

Programming models – Master/Worker

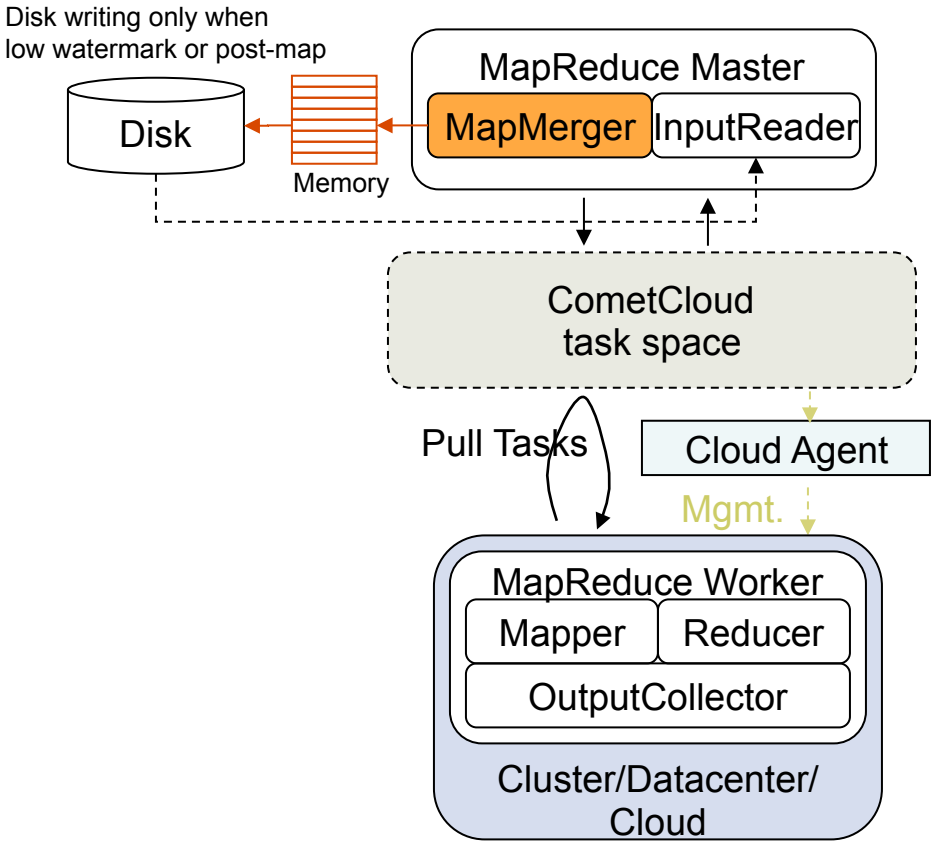
- A **master** generates tasks, submits them into the coordination space, and collects results.
- A **secure worker** provides its local space as the part of the coordination space as well as providing computing capability.
- An **unsecured worker** provides only computing capability and gets a task through the proxy and a request handler.



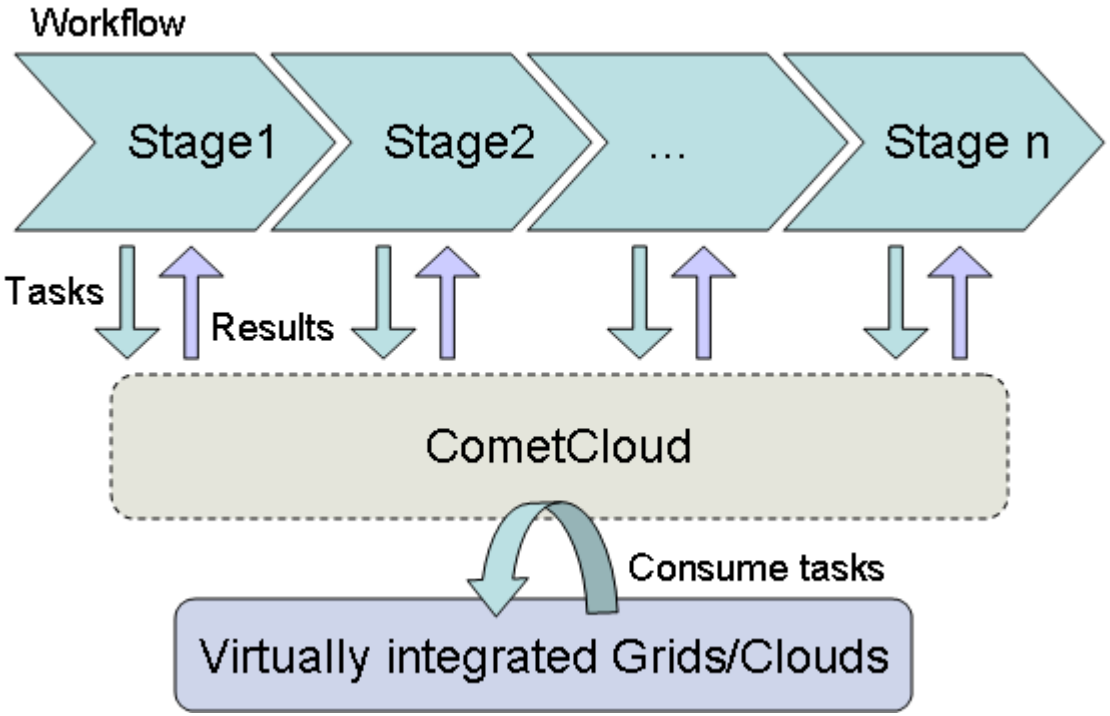
- **Proxy** receives task requests from unsecured workers and forward the requests to one of request handler. **Request handler** is part of the overlay so as to host Comet space and picks up a task for unsecured workers.

Programming models – Map/Reduce

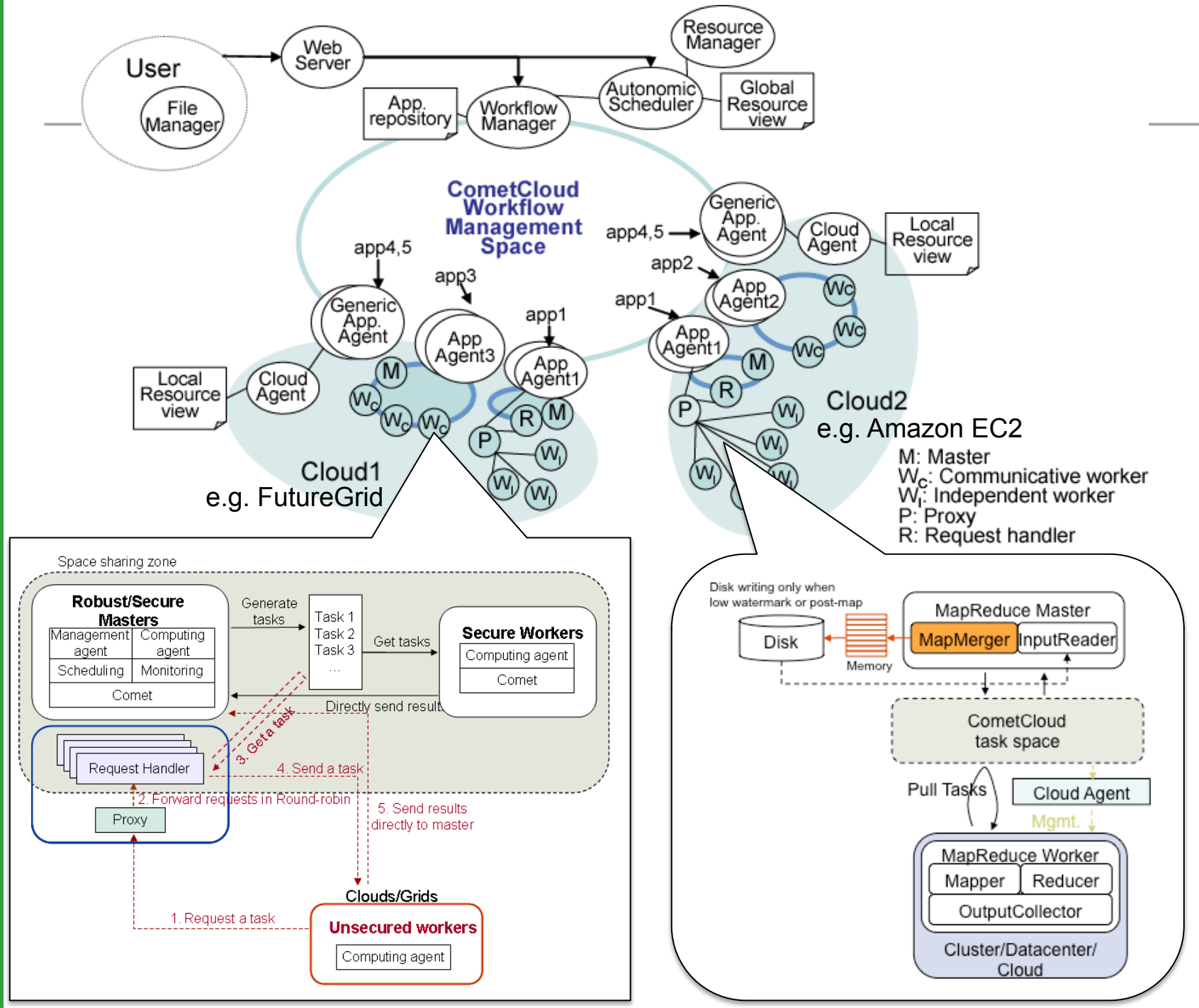
- Accelerate MapReduce for the application running with a large number of small or medium files.
- MapMerger to keep results in memory as much as possible.



Programming models – Workflow



- Each stage is heterogeneous in terms of behavior, the length of computation, the amount of required resources, etc.
- Stages should be completed in an sequence since the output of a stage becomes the input of the next stage.


User
File Manager
Web Server
Resource Manager
App. repository
Workflow Manager
Autonomic Scheduler
Global Resource view
CometCloud Workflow Management Space
app4,5
app3
app1
Generic App. Agent
App Agent3
App Agent1
Local Resource view
Cloud Agent
M
Wc
R
P
Wi
Cloud1
e.g. FutureGrid
app4,5
app2
app1
Generic App. Agent
Cloud Agent
Local Resource view
App Agent2
App Agent1
Cloud2
e.g. Amazon EC2
M: Master
Wc: Communicative worker
Wi: Independent worker
P: Proxy
R: Request handler
Space sharing zone
Robust/Secure Masters
Management agent
Computing agent
Scheduling
Monitoring
Comet
Generate tasks
Task 1
Task 2
Task 3
Get tasks
Secure Workers
Computing agent
Comet
Directly send result
Request Handler
3. Get a task
4. Send a task
2. Forward requests in Round-robin
Proxy
5. Send results directly to master
Clouds/Grids
1. Request a task
Unsecured workers
Computing agent
Disk writing only when low watermark or post-map
Disk
Memory
MapReduce Master
MapMerger
InputReader
CometCloud task space
Pull Tasks
Cloud Agent
Mgmt.
MapReduce Worker
Mapper
Reducer
OutputCollector
Cluster/Datacenter/ Cloud

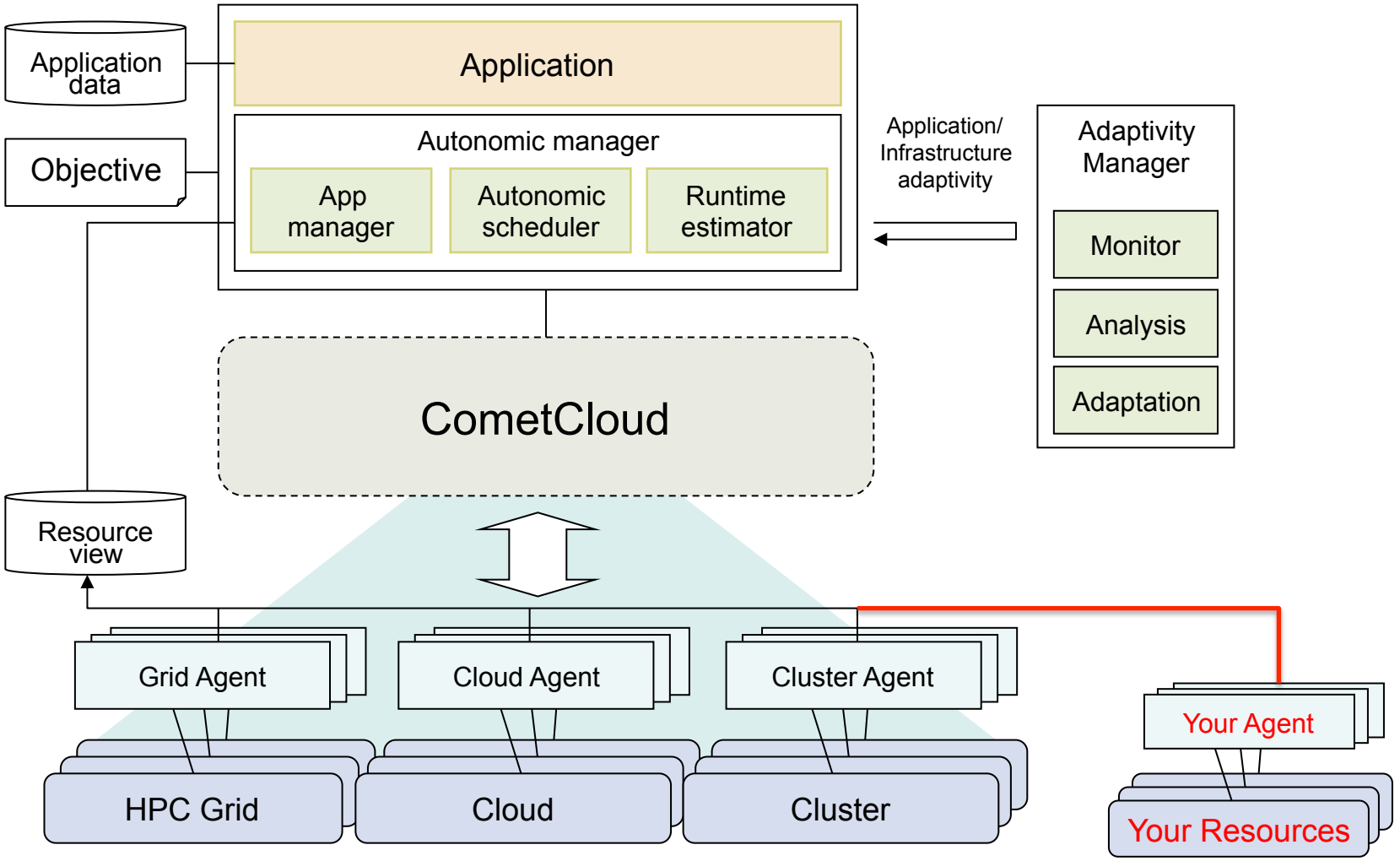
Resource Agents

cac

NSF

RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Add your own resources to CometCloud



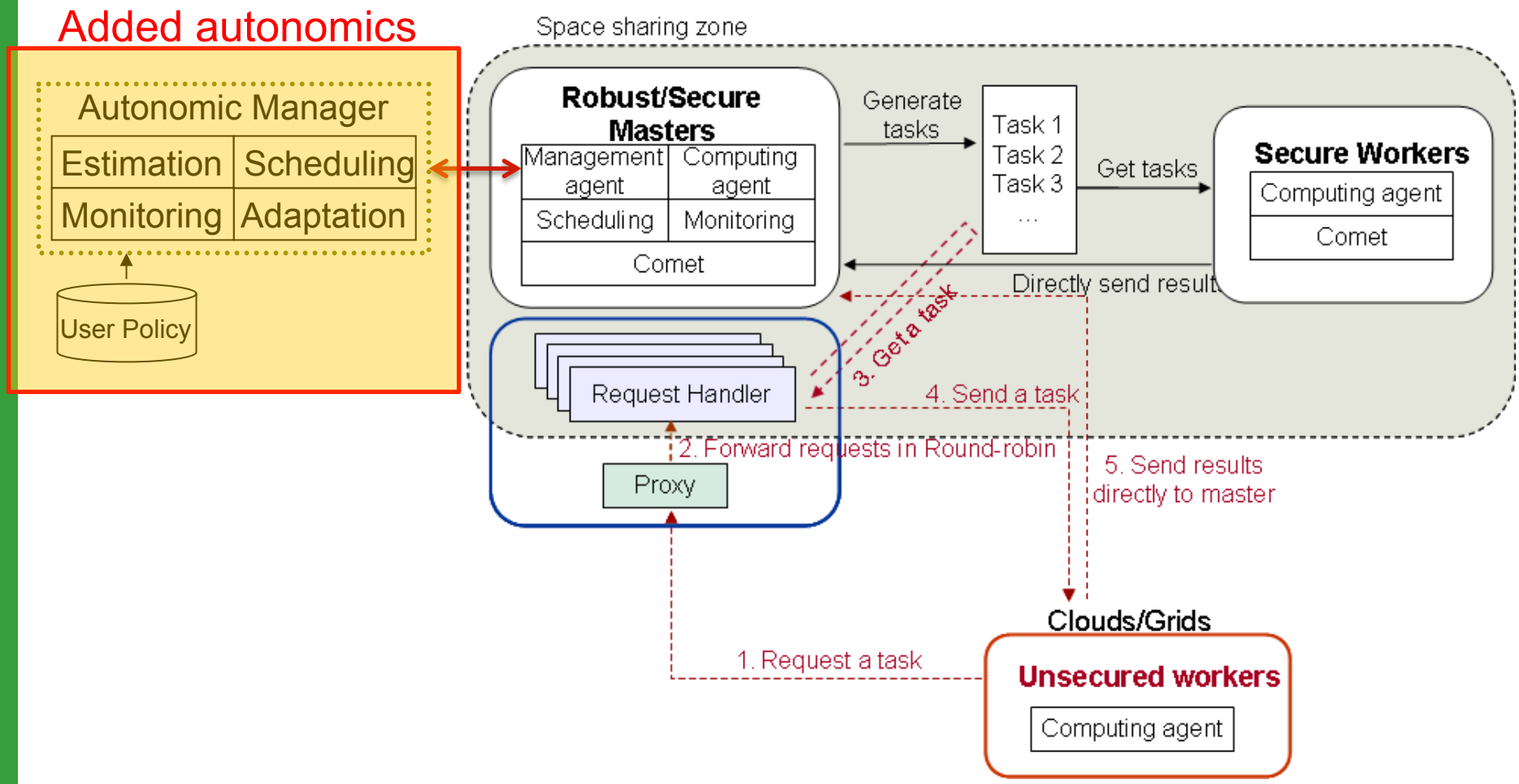
Agent Interface

- Extend the Java Agent Interface
- tassl.automate.agent.CloudAgentFramework
  - allocateResource
  - allocateCometWorker
  - startResource
  - startCometWorker
  - shutdownResource
  - getAvailableResources
- Implement these functions for your resource

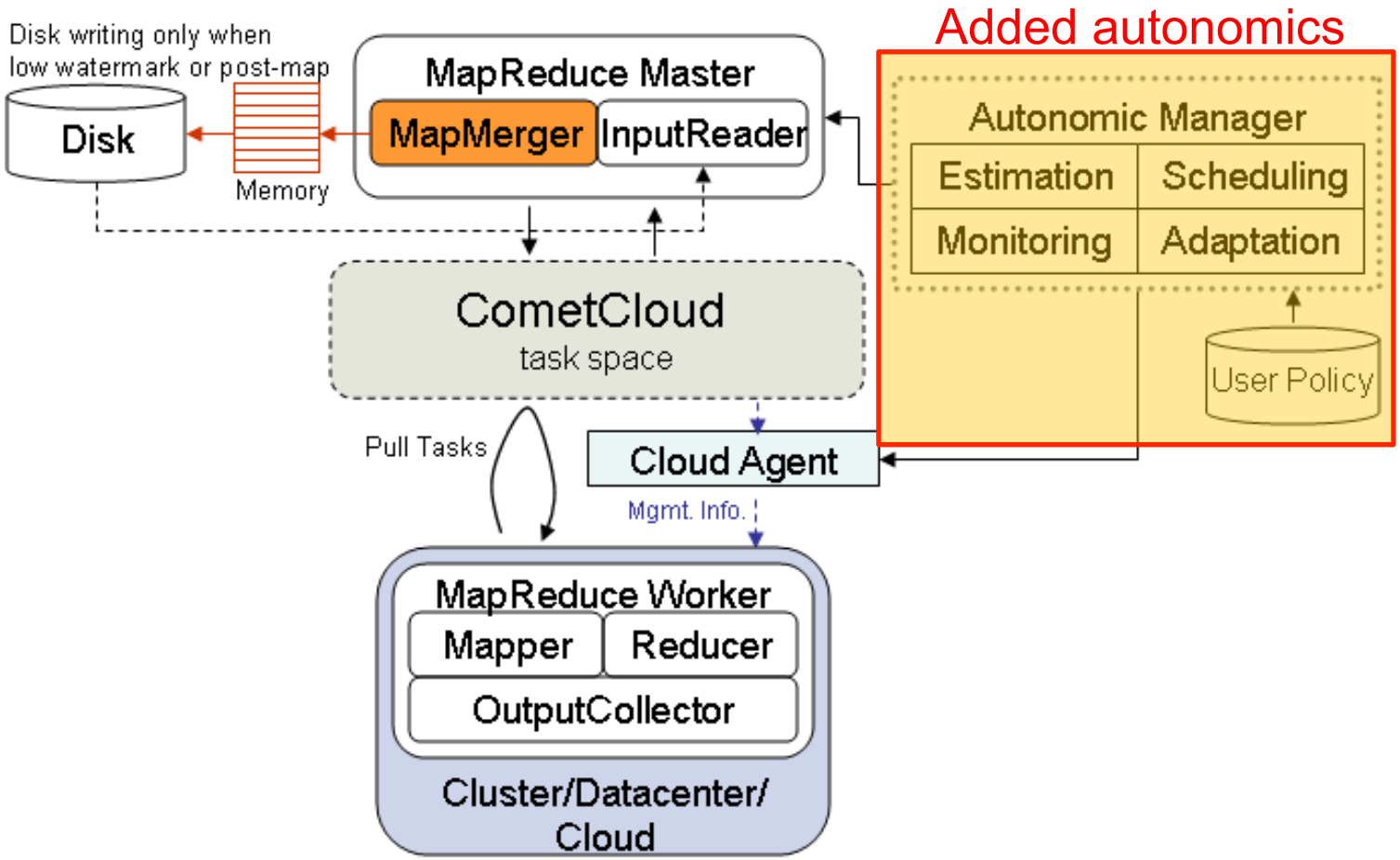
CometCloud

CLOUD BURSTING

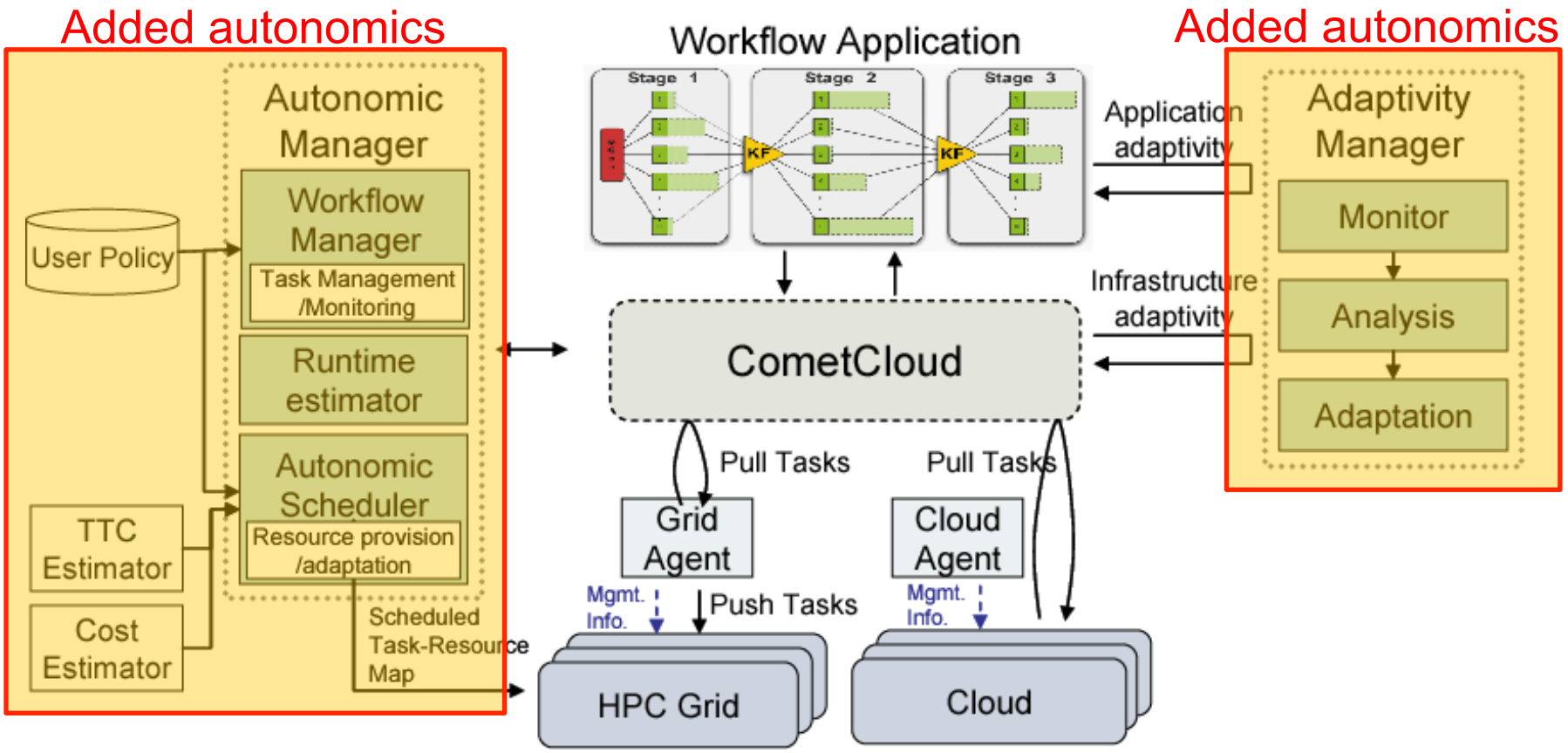
Cloudburst in Master/Worker



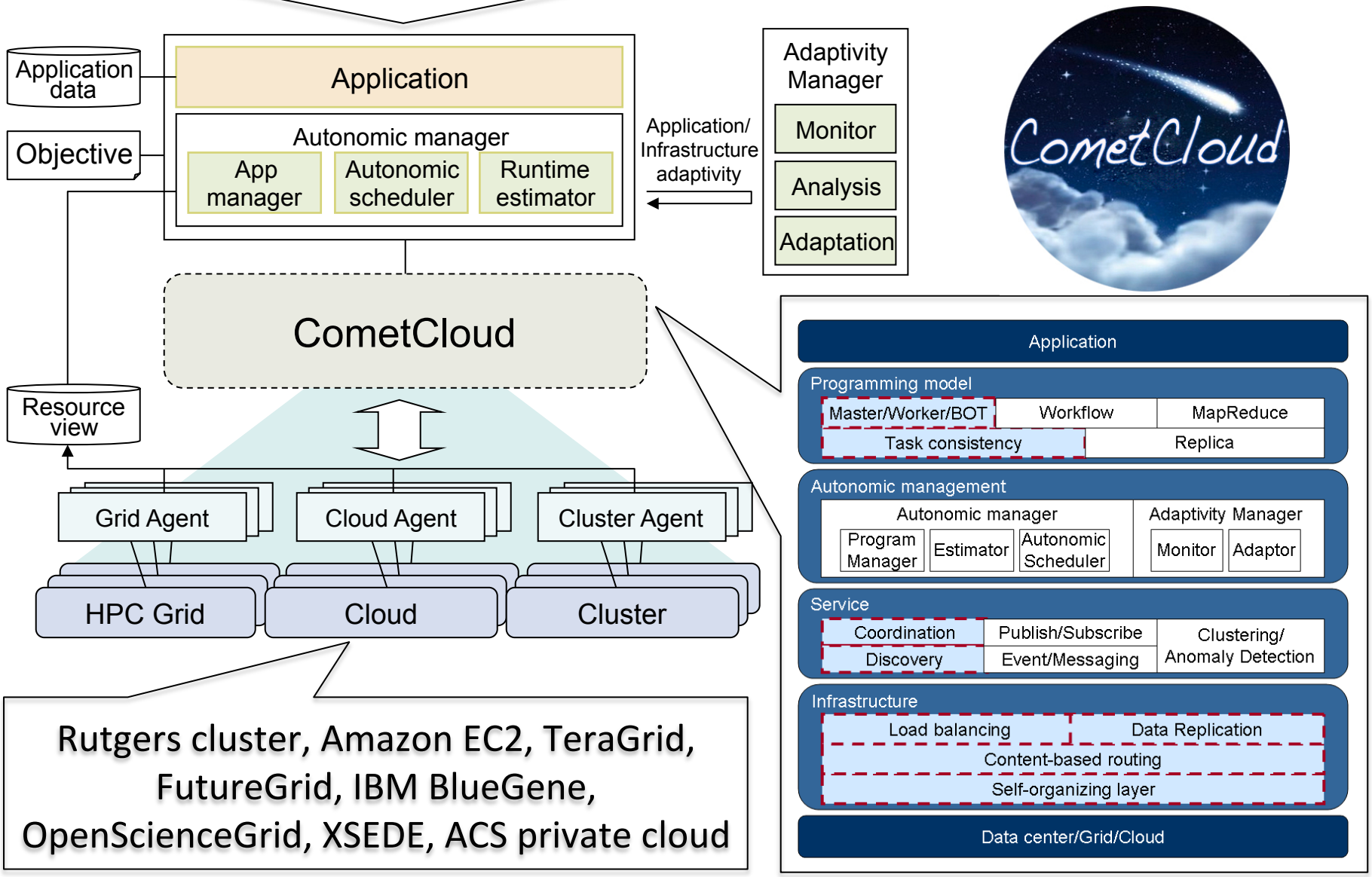
Cloudburst in MapReduce



Cloudburst in Workflow



Value@Risk, Protein data bank, Medical informatics, Ensemble Kalman Filter for oil reservoir simulation, Molecular dynamics & drug design, Jacobi, PDEs solvers using synchronous and asynchronous iterations



Rutgers cluster, Amazon EC2, TeraGrid, FutureGrid, IBM BlueGene, OpenScienceGrid, XSEDE, ACS private cloud

Questions?

For more info please visit

http://cometcloud.org

CAC

RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY