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

Moustafa AbdelBaky, Hyunjoo Kim, Ivan Rodero and Manish Parashar

moustafa.a, irodero, parashar@rutgers.edu, hyunjoo.kim@xerox.com

The NSF Cloud and Autonomic Computing Center
Electrical & Computer Engineering Department
Rutgers, The State University of New Jersey, USA
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.
**Objective**

- 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 (www.cometcloud.org)

- **Application/Programming layer autonomics**: Dynamics 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

Redbox denotes open source.
CometCloud space management

- A virtual semantically-specialized shared space
  - Constructed from a application define multi-dimensional information space, which is deterministically mapped onto the dynamic system of peer nodes

- The space is associatively accessible by all system nodes.
  - Access is independent of the physical locations of data tuples or hosts

- Dynamically constructed transient spaces enable application to exploit context locality
CometCloud space: basic idea

- Constructed from a semantic multi-dimensional information space
  - Numbers, English letters, wild card ‘*’

- Application specific semantics
  - Dimensions, coordinate, keywords

2D keyword space for a P2P file sharing system

3D keyword space for resource sharing, using the attributes: storage space, base bandwidth and cost
CometCloud space: tuple management

- XML tuples and templates

```
<contact>
  <name> Smith </name>
  <phone> 7324451000 </phone>
  <email> smith@gmail.com </email>
  <dep> ece </dep>
</contact>
```

```
<contact>
  <name> Smith </name>
  <phone> 7324451000 </phone>
  <email>* </email>
  <dep>* </dep>
</contact>
```

```
<contact>
  <na*> Smith </na*>
  <**>
  <dep> ece </dep>
</contact>
```

(a) (b) (c)

- Basic coordination primitives

- **out** \((ts, t)\): a non-blocking operation that inserts tuple \(t\) into space \(ts\)

- **in** \((ts, t')\): a blocking operation that removes a tuple \(t\) matching template \(t'\) from the space \(ts\) and returns it

- **rd** \((ts, t')\): a blocking operation that returns a tuple \(t\) matching template \(t'\) from the space \(ts\). The tuple is not removed from the space
Managing autonometrics

- **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.
Managing autonomics procedure

- Sampling and estimation
  - Estimate runtime of all tasks on all resource classes

- Scheduling and provision
  - Schedule each task to the most appropriate resource class based on policy, constraints or the objective and the number of nodes per resource class is decided

- Monitor and adaptation
  - The actual runtime of each task is monitored and scheduling decision is adapted if the runtime is different from the estimated runtime enough to affect objective.
Components for autonomics

- **Workflow Manager**
  - Coordinates the execution of the overall application workflow, based on user-defined policies, using Comet spaces.
  - Consists of workflow planner and task monitors/manager

- **Runtime Estimators**
  - Translate hints about computational complexity provided by the application into runtime and/or cost estimates on a specific resource

- **Autonomic Scheduler**
  - Performs key autonomic management tasks
    - Estimate the relative complexities of the tasks to be scheduled and clusters these to identify potential scheduling blocks.
    - Compute anticipated runtimes for these blocks on available resource classes
    - Determine the initial hybrid mix HPC Grids/Clouds/Clusters resources based on user/system-defined objectives, policies and constraints
  - Communicates resource-class-specific scheduling policies to the agents
Components for autonomic

- **Grids/Clouds/Clusters Agents**
  - Provisioning resources on their specific platforms, configuring workers as execution agents on these resources, and appropriately assigning tasks to these workers.
  - Pull-based task assignment, where workers directly pull tasks from the Comet space
    - Direct pull model on a Cloud
    - Combined push-pull model on HPC Grid
      - Push smart pilot-jobs containing workers into the batch queues of the HPC Grids, then pull tasks from the Comet space when they are scheduled to run

- **Monitor**
  - The monitor observes the tasks runtimes which workers report to the master in the results.
  - The monitor makes a function call to the analyzer to check if the changing resource status still meets the user objective when the time difference becomes larger than a certain threshold.
Components for autonomics

- Analyzer
  - The analyzer is responsible for re-estimating the runtime of remaining tasks based on the historical data gathered from the results and evaluating the expected TTC of the application and total cost.

- Adaptor
  - The adaptor retrieves a new schedule for the remaining tasks from the autonomic scheduler.
  - The adaptor launches more workers or terminating existing workers based on the new schedule.
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.
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 allocatable 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 ENVIRONMENT
- 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.
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.
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.
Workflow management

- Federated cloud management
- Application management
Workflow management
Cloudburst in Master/Worker

Added autonomies

**Autonomic Manager**
- Estimation
- Scheduling
- Monitoring
- Adaptation

**User Policy**
Cloudburst in MapReduce

Added autonomies
Cloudburst in Workflow

Added autonomies

Workflow Application

CometCloud

Grid Agent

Cloud Agent

HPC Grid

Cloud

Workflow Manager

Runtime estimator

Autonomic Manager

User Policy

TTC Estimator

Cost Estimator

Task Management/monitoring

Resource provision/adaptation

Scheduled Task-Resource Map

Application adaptivity

Infrastructure adaptivity

Adaptivity Manager

Monitor

Analysis

Adaptation
CometCloud – Applications

- Deployed real-world applications
  - VaR analytics engine (master/worker)
  - Medical informatics (master/worker, workflow)
  - Molecular dynamics & drug design (mapreduce)
  - PDEs solvers using synchronous and asynchronous iterations (workflow)
- Federated cloud
  - Rutgers cloud, Amazon EC2, ACS cloud, TeraGrid, FutureGrid, IBM BlueGene/P
Summary

- **CometCloud**
  - Federated cloud with cluster/grid/cloud
    - Rutgers cluster, Amazon EC2, TeraGrid, FutureGrid, IBM BlueGene, ACS private cloud
  - Programming abstractions
    - Master/worker, MapReduce/Hadoop, Workflow
  - User objective-driven scheduling and resource provisioning
  - Deployed various real-world applications
    - Value@Risk, Protein data bank, Medical image registration, Ensemble Kalman Filter for oil reservoir simulation, Jacobi

- [http://cometcloud.org](http://cometcloud.org)
Questions?

For more info please visit

http://cometcloud.org