Automated Control for Elastic Storage

Harold Lim, Shivnath Babu, Jeff Chase

Duke University
Motivation

- We address challenges for controlling elastic applications, specifically storage.

- Context: Cloud providers that offer a unified hosting substrate.
Motivation

- Let us consider an *infrastructure as a service* cloud provider (e.g., Amazon EC2).
  - Cloud API allows customers to request, control, release virtual server instances on demand.
  - Customers are charged on a per instance-hour usage.
- Cloud computing allows customers to request only the number of instances they need.
- Opportunity for elasticity - where the customer acquires and releases resources in response to dynamic workloads.
Motivation

- Mechanisms for elastic scaling are already present in a wide range of applications.

- We need to have a good automated control policy for elastic scaling.

- We build on the foundations of previous works (e.g., Parekh2002, Wang2005, Padala2007).
System Overview

- Figure shows our target environment.
  - Controlled Elasticity.
  - Dynamic workload.
  - Meet response time SLO.
- We designed a control policy for multi-tier web services.
  - We use Cloudstone application, a Web 2.0 events calendar, with HDFS as the storage tier.
  - Our approach views the controller as combining multiple elements with coordination.
  - Controlling the storage tier is a missing element of an integrated cluster-based control solution.
System Overview

- **Controller**
  - Runs outside of the cloud and distinct from the application itself.
  - Application control left to the guest.
  - Can combine multiple control elements.
  - Allows application-specific control policies.

- **Control Goals**
  - Handle unanticipated changes in the workload.
  - Resource efficiency (guest pays the minimum necessary to meet its SLO).
System Overview

- **Cloudstone Application**
  - Application has mechanism for elastic scaling.
  - There is a mechanism to balances Cloudstone requests across servers.

- **HDFS Storage System**
  - Data is distributed evenly across servers.
  - Storage and I/O capacity scales roughly linearly with cluster size.
System Overview

- Controller Issue – Discrete Actuator
  - Cloud providers allocate resources in discrete units.
  - No access to hypervisor-level continuous actuators.

- New Issues with Controlling Storage
  - Data Rebalancing
    - Need to move/copy data before getting performance benefits.
  - Interference to Guest Services
    - Data rebalancing uses the same resources to serve clients.
    - The amount of resources to use affects completion time and the degree of interference to client performance.
  - Actuator Delays
    - There is delay before improvements.
Outline

- Motivation
- System Overview
- *Controller Design*
- Implementation
- Evaluation
- Related Work
Controller Design

- The elastic storage system has three components.
  - **Horizontal Scale Controller (HSC)**
    - Responsible for growing and shrinking the number of nodes.
  - **Data Rebalance Controller (DRC)**
    - Responsible for controlling data transfers to rebalance the cluster.
  - **State machine**
    - Responsible for coordinating the actions of HSC and DRC.
Horizontal Scale Controller

Control Policy

- Applied proportional thresholding (Lim2009) to control storage cluster size, with average CPU utilization as sensor.
  - Modifies classical integral control to have a dynamic target range (dependent on the size of the cluster).
  - Prevents oscillations due to discrete/coarse actuators.
  - Ensures efficient use of resources.

\[
u_{k+1} = \begin{cases} 
  u_k + K_i \times (y_h - y_k) & \text{if } y_h < y_k \\
  u_k + K_i \times (y_l - y_k) & \text{if } y_l > y_k \\
  u_k & \text{otherwise}
\end{cases}
\]
Data Rebalance Controller

- Uses the rebalance utility that comes with HDFS.
- Actuator – The bandwidth $b$ allocated to the rebalancer.
  
  - The maximum amount of outgoing and incoming bandwidth each node can devote to rebalancing.
  
  - The choice of $b$ affects the tradeoff between lag (time to completion of rebalancing) and interference (performance impact on foreground application).
  
  - We also discovered that HDFS rebalancer utility has a narrow actuator range.
Data Rebalance Controller

Sensor and Control Policy

- From the data gathered through a planned set of experiments, we modeled the following:
  - Time to completion of rebalancing as a function of bandwidth and size of data (Time = $f_t(b,s)$).
  - Impact of rebalancing as a function of bandwidth and per-node workload (Impact = $f_i(b,l)$).
- The choice of $b$ is posed as a cost-based optimization problem. Cost = $A \times \text{time} + B \times \text{Impact}$.
  - The ratio of $A/B$ can be specified by the guest based on the relative preference towards Time over Impact.
State Machine

- Manages the mutual dependencies between HSC and DRC.
  - Ensures the controller handles DRC's actuator lag.
  - Ensures interference and sensor noise introduced by rebalancing does not affect the HSC.
Implementation

- **Cloud Provider**
  - We use a local ORCA cluster as our cloud infrastructure.
    - A resource control framework developed at Duke University.
    - Provides resource leasing service.
  - The test cluster exports an interface to instantiate Xen virtual machine instances.
Implementation

- **Target Guest Service**
  - Cloudstone - Mimics a Web 2.0 events calendar application that allows users to browse, create, join events.
    - Modified to run with HDFS for unstructured data.
    - HDFS does not ensure requests are balanced but the Cloudstone workload generator accesses data in a uniform distribution.
    - Modified HDFS to allow dynamically setting $b$
Implementation

- Controller
  - Written in Java.
  - Uses ORCA's API to request/release resources.
  - Storage node comes with Hyperic SIGAR library that allows the controller to periodically query for sensor measurements.
  - HSC and DRC runs on separate threads and are coordinated through the controller's state machine.
Evaluation

- Experimental Testbed
  - Database server (PostgreSQL) runs on a powerful server (8GB RAM, 3.16 GHz dual core CPU).
  - Forward Tier (GlassFish Web Server) runs in a fixed six-node cluster (1GB RAM, 2.8GHz CPU).
  - Storage nodes are dynamically allocated virtual machine instances, with 30GB disk space, 512MB RAM, 2.8GHZ CPU.
  - HDFS is preloaded with at least 36GB of data.
Evaluation

- 10-fold increase in Cloudstone workload volume
- Static vs Dynamic provisioning
Evaluation

- Small increase (35%) in Cloudstone workload volume
- Static vs Dynamic provisioning
Evaluation

- Decrease (30%) in Cloudstone workload volume
- Static vs Dynamic provisioning
Evaluation

- **Comparison of rebalance policies**

- An aggressive policy fixes SLO problems faster but incurs greater interference.

- A conservative policy has minimal interference but prolongs the SLO problems.
Related Work

- Control of Computing Systems

- Data Rebalancing

- Actuator Delays
  - Soundararajan2006

- Performance Differentiation
Thank You

- Controller runs outside of the cloud.
- Controller fixes SLO violations.
- Proportional thresholding to determine cluster size.
- For elastic storage, data rebalancing should be part of the control loop.
- State machine to coordinate between control elements.
System Overview

- **Controller**
  - Reflects the separation of concerns in the functionalities among provider and guests.
    - Guests are insulated from details of underlying physical resources.
    - Provider is insulated from details of application.
  - Application control is factored out of the cloud platform and left to the guest.
Horizontal Scale Controller

- **Actuator** - Uses cloud APIs to change the number of active server instances.

- **Sensor** – A good choice must satisfy the following properties.
  - Easy to measure without intrusive code instrumentation.
  - Should measure tier-level performance.
  - Should not have high variations and correlates to the measure of level of service as specified in the client's SLO.

- We use average CPU utilization of the nodes as our sensor.
  - Note that for other target applications, one has to find a suitable sensor and may differ from our choice of using CPU utilization.