A Cloud Computing Approach to On-Demand and Scalable CyberGIS Analytics

Pierre Riteau (UChicago), Myunghwa Hwang (UIUC), Anand Padmanabhan (UIUC), Yizhao Gao (UIUC), Yan Liu (UIUC), Kate Keahey (Argonne), Shaowen Wang (UIUC)

Presented by Kate Keahey
Geographic Information Systems (GIS)

"Geographic Information Systems (GIS) are simultaneously the telescope, the microscope, the computer, and the Xerox machine of regional analysis and synthesis of spatial data." (Abler 1988)

www.nimbusproject.org
www.cybergis.org
The Problem

• Consistent response time in peak demand
  – Example: online education
  – Demand from many users varies over time and across tasks
  – Response time has critical impact on user experience
• Adaptation to varying sizes of analytical problems
  – Example: Problem Solving Environments
  – Real-time interaction, requests with potentially large spatial data
Using Cloud Resources

Potential

• On-demand provisioning of resources
• Pay-as-you-go cost model

Challenges

• Deploying spatial analytics modules on cloud resources
• Integrating cloud resources with existing CyberGIS infrastructure and middleware
• Balancing computational workload across resources
• Scaling resources dynamically so that acceptable quality of service can be maintained
PySAL on CyberGIS Gateway

www.nimbusproject.org
www.cybergis.org
CyberGIS: Current Architecture

User

CyberGIS Gateway

GISolve Middleware

Data Store

Download Inputs

Input Data Management

Run Regression

Download Outputs

VM

Static cluster

VM
CyberGIS: Original Architecture (cntd)

- Users submit jobs through the Gateway
- Input data uploaded to the Data Store
- GISolve middleware distributes requests in round robin to a static cluster of VMs with PySAL installed
- No queuing: extra requests rejected
- Output downloaded directly from VM
  - Assumes static deployment
Moving CyberGIS to a Cloud Platform

- Need to add/remove instances on the fly
- Our solution:
  - Add queuing load balancer behind GISolve
  - No need to modify GISolve middleware code
  - Use Nimbus Phantom and the load balancer information to implement auto-scaling
Nimbus Phantom

Application-specific qualities: e.g., workload queALiEn, PBS, AMQP, and others

Generic/system qualities: deployment status, load, bank account, etc.

create, manage, destroy

Policy

Lifecycle states

Paper: “Infrastructure Outsourcing in Multi-Cloud Environment”
CyberGIS: Modified Architecture

- User
- CyberGIS Gateway
- GLSolve Middleware
- Queuing Load Balancer
- Data Store
- VM
  - VM
  - VM
  - VM
  - Dynamically-scaled virtual cluster
- Nimbus Phantom
- Phantom Decision Engine
- Query metrics
- Scale domain
- Run regression
- Download Inputs
- Upload outputs
- Download Outputs
- Input Data Management
Implementation

- HAPerxy as load balancer
  - Metrics extracted using haproxyctl
- Custom Phantom decision engine
  - Tracks the number of connections to HAPerxy
  - Requests changes in number of instances
- Policy
  - Requests new instances when VMs fill to capacity
  - “Lazy termination” based on history to avoid thrashing
- Instances are integrated in HAPerxy when booted and removed when terminated
- Output files stored on data store
  - Instances can be terminated any time
Experimental Platform

- Used OpenStack Alamo on FutureGrid
- Dedicated instances for:
  - HAProxy (m1.tiny)
  - Data Store (m1.small)
  - Regression service (m1.small)
- Comparison of:
  - Static cluster (original architecture)
  - Static cluster + dynamically added instances
Experiments

• Two use cases scenarios
  • Scenario 1
    – Small number of users
    – Large data files
    – Example: scientists conducting a study
  • Scenario 2
    – Large number of users
    – Smaller data files
    – Example: labs conducted as part of a class
• Generated load with Apache JMeter

6/23/14
www.nimbusproject.org
www.cyberGIS.org
Scenario 1 (Large Requests)

- Number of users varies from 4 to 16
- 5 requests per user
- 10 second pause between requests
- Static cluster of 5 VMs
- Maximum of 10 dynamic cloud instances
- 2 minutes auto-scaling history buffer
- Single request per VM (no concurrency)
Scenario 1 (cntd)

**Response time**

![Graph showing response time versus concurrent users and number of cloud instances.](image)

- **Response time (static)**
- **Response time (static + dynamic)**
- **Number of cloud instances**

- **Concurrent users**:
  - 4
  - 8
  - 12
  - 16

- **Average response time (s)**

6/23/14

www.nimbusproject.org
www.cybergis.org
Auto-scaling with 16 users

Impact of concurrent requests

Number of connections
Dynamic cloud instances
Total instances

Number of connections
Time (s)

Number of instances
Auto-scaling with 16 users (cntd)

Impact of dynamic cloud instances over response time

Response time
Dynamic cloud instances
Total instances

0 100 200 300 400 500 600 0 5 10 15 20

Response time (s)
Number of instances

Time (s)

0 100 200 300 400 500 600

6/23/14
Scenario 2 (Small Requests)

- Number of users from 32 to 64
- 5 requests per user
- 10 second pause between requests
- Single static VM
- Maximum of 10 dynamic cloud instances
- 2 minutes auto-scaling history buffer
- 8 concurrent requests per VM
Scenario 2 (cntd)
Summary

• Response time is critical for CyberGIS users
• Requirement for a system that can react to changes in demand
• Integrated Nimbus Phantom auto-scaling
• Maintains low response time
• Future work:
  – Better request management
  – Scaling policy improvements
  – Data storage scalability