Trends in High-Performance Distributed Computing

Nimbus Platform: Infrastructure Outsourcing for Science

Kate Keahey
keahey@mcs.anl.gov
Argonne National Laboratory
Computation Institute, University of Chicago
What is Nimbus?

High-quality, extensible, customizable, open source implementation

**Nimbus Platform**
- Context Broker
- Cloudinit.d
- Elastic Scaling Tools

*Enable users to use IaaS clouds*

**Nimbus Infrastructure**
- Workspace Service
- Cumulus

*Enable providers to build IaaS clouds*

*Enable developers to extend, experiment and customize*
Applications and Patterns
• Why run in the cloud?
  – Reduce “time to science”
  – Near real-time processing

Challenge: what makes a proton spin?

Images courtesy of J. Balewski
• Towards Observatory Science
• Sensor-driven processing
• Scalable and Highly Available (HA) services
• Nimbus team leading the development of Common Execution Infrastructure
• From regional Nimbus clouds to commercial clouds
• Building platform services for integrated, repeatable support for on-demand science
Scaling Scenarios

- 2008: The ALICE proof-of-concept
- 2009: ElasticSite prototype
- 2009: OOI pilot

Paper: “Elastic Site”, CCGrid 2010
Trends and Patterns

• On-demand, **elastic** processing…
  – Observatories, experiments, conference deadlines, fluctuating workload, growth management…
• … over multiple providers….
  – **Risk-mitigation**: not enough cycles, failure, market factors
• … with elementary ease of use…
  – **Automated** provisioning of infrastructure resources
• … facilitates going from one-offs to **production** runs…
  – Steadily increasing in both size and buy-in
• … given the right model
  – **Failure-prone** environment
  – Achieving “uninterrupted power supply"
Outsourcing for Science: Building an Infrastructure Platform
Goals for Nimbus Scaling Tools

• Any Scale
  – Scale in response to a diverse set of sensors/triggers
  – Both system and application sensors

• High Availability
  – “Any VM can die”: system or user VMs
  – Minimizing time to recovery (TTR)

• Multi-cloud
  – Work across private, community and commercial clouds

• Your Polices, Our Enactment
  – User-defined sensors/triggers and policies

• Engineered from the ground up to work with infrastructure clouds

• Easy on the user
Escalation Pattern

Applications: absorb resources and tolerate failure

User Domain
(configuration and security)

Operational Units

Domain Management:
Monitor and regulate domain properties based on system-specific and application-specific metrics
Scaling Mechanics

- Monitor scaling factors:
  - Generic/system qualities: deployment status, load, bank account, etc.
  - Application-specific qualities, e.g., a workload queue for ALiEn, PBS, AMQP,…
- Evaluate against policies
- Scale and/or recover
  - For user components
  - For system components
  - Across different cloud providers
- Release as a Service
- Beta release this year
  - Customizable to input, policy, decision engine, provider, etc.
  - Initially available on FutureGrid resources
Adventures in Availability

Mean time between failures

\[ A = \frac{MTBF}{MTBF + MTTR} \]

Maximum time to repair

- Time to repair (TTR)
  - Diagnosis
  - Time to scale (TTS)
    - PENDING (request)
    - STARTED (deployment)
    - RUNNING (contextualization)

TTS preliminary results (mean + min/max)

Node Information (relative to new_node event)
Applications

- Elastic MapReduce
- Portals and Gateways
- Custom Applications
- Schedulers
- Data Transfer Systems
- Workflow Systems (Swift)

Application adaptation:
- Library of generic sensors
- Application-specific sensors
- Policies
- Decision Engine

Nimbus Platform
Contextualization, multi-cloud bridge, repeatable launches, scaling and High Availability
Parting Thoughts

• Cloud computing is a solution!
• Computation outsourcing for science
  – Ease of use and automation: “computing power on tap”
  – Science has unique outsourcing needs
• Escalation Pattern: domains
  – From desktop to cloud
  – Enforcing domain properties
• Escalation Pattern: applications
  – The ability to absorb resources is critical
  – Fault-tolerance is increasingly a requirement
  – Emphasis on programming models that support them
The Nimbus Team

• Project lead: Kate Keahey, ANL&UC
• Exceptional programmers:
  – David LaBissoniere - University of Chicago
  – John Bresnahan - Argonne National Laboratory
  – Patrick Armstrong - University of Chicago
  – Pierre Riteau - University of Chicago
• 12 github contributors
• And many others
  – See http://www.nimbusproject.org/about/people/