Towards Transactional Cloud Resource Orchestration

Changbin Liu†, Yun Mao‡, Xu Chen‡, Mary Fernandez‡, Boon Thau Loo†, Jacobus Van der Merwe‡

†University of Pennsylvania ‡AT&T Labs – Research

Motivation

- **Infrastructure-as-a-Service (IaaS) cloud computing is increasingly attractive**
- Supply: service providers and device vendors
- Demand: cloud users

- **Cloud resource orchestration**
  - Management and manipulation of resources
  - Basis of IaaS cloud computing

- **Challenges**
  - Large-scale datacenters
  - Wide diversity of resources: compute, storage, and network
  - Highly dynamic environment
  - Frequent failures and unhandled errors
  - Enforcing system-wide policy constraints
  - Large number of concurrent operations

- **Our approach: DMF**
  - Data-centric Management Framework
  - A resource orchestration platform to build cloud services

Design

- **Core concept:**
  - Orchestration procedures → Transactions

- **Atomicity:**
  - Either all state transitions in a transaction are performed or none are

- **Consistency:**
  - All the policy constraints are satisfied, which reflect service and engineering rules

- **Isolation:**
  - Concurrent transactions are maximally parallelized with no race conditions

- **Durability:**
  - Committed transactions survive system crashes

System Architecture

- **Transaction execution**
  - Phase 1: logical layer simulation to check constraints and errors
  - Phase 2: physical layer operations
  - If failures: abort and rollback

- **Concurrency control**
  - 2^n states management ↔ n transactions either commit or abort

- **Cross-layer consistency maintenance**
  - Logical and physical layer out-of-sync
  - Reload or repair

- **Opportunistic transactions**
  - Certain operations succeed with high probability
  - Only keep the successful case to reduce overhead

- **Complexity**
  - Exponential logical layer overhead vs. physical layer parallelism
  - Opportunistic transactions achieve linear speedup

Implementation

- **Language choice: Python**
  - Rich libraries, highly readable syntax
  - Easy to learn for cloud administrators and users
  - Embed a domain specific language inside DMF

- **Data structure optimization**
  - Copy-on-write to avoid expensive memory copy

- **Cloud resources modeling**
  - Compute: Xen VMs, libvirt APIs
  - Storage: DRBD, GNBD
  - Network: Juniper routers

- **Cloud services: SimpleCloud**
  - Deployment on 16 hosts in 3 datacenters (California, Illinois, Texas) on ShadowNet
  - Manage 1000+ VMs

Future work

- **Decentralized architecture**
  - Higher scalability and availability
  - Distributed transactions

- **Open-source code release**
  - Contribute to open-source community
  - Gain a larger user base to further explore and improve the usability of DMF

Acknowledgements: This work is supported by NSF grants CCF-0820208 and CNS-1040672, and the AT&T Labs Research summer student internship program.

Performance Microbenchmarks

- **Cloud Workload**
  - Transaction completion time (cloud workload)

Figure 1: Scalability trends of DMF
Figure 2: CDF of transaction completion time (synthetic workload)
Figure 3: # of cloud operations derived from the hosting trace
Figure 4: CDF of transaction completion time (cloud workload)