Scalability Bugs: When 100-Node Testing is Not Enough

Riza O. Suminto, Cesar A. Stuardo and Haryadi S. Gunawi
UCARE Lab

U of C systems research on Availability, Reliability & Efficiency
Bugs in Large Distributed Systems?

- "Classic" critical bugs
  - Concurrency
  - Configuration
  - Performance
  - Security

- New classes of bugs?

---

**What Bugs Live in the Cloud?**
A Study of 3000+ Issues in Cloud Systems

**Abstract**

We conduct a comprehensive study of development and deployment issues of six popular and important cloud systems (Hadoop, MapReduce, HDFS, Thrust, Twitter, and Flavor). From the bug repository (21,990 submitted issues within a 1 year period), we find that bugs are no less prevalent in cloud systems than in single-server systems. However, bugs have different characteristics in cloud systems. The most prevalent "classic" critical bugs in cloud systems are: concurrency bugs, configuration bugs, performance bugs, and security bugs. We also identify new classes of bugs that are unique to cloud systems. These new classes of bugs include bugs related to large distributed systems and bugs related to the cloud infrastructure.

---

**Why Does the Cloud Stop Computing?**
Lessons from Hundreds of Service Outages

**Abstract**

We conducted a cloud outage study (COS) of 32 popular cloud services to understand the reasons for service outages. From our analysis of 1247 outages, we find that the most common cause of cloud outages is human error, followed by infrastructure issues and software issues. We also find that outages are more likely to occur on weekends and holidays. We conclude that cloud outages are a major concern for cloud service providers and users.
Scale dependency?

- **Cassandra**
  - Bug #6409: "With >500 nodes, ... gossip protocol slow"

- **Hadoop/HDFS**
  - Bug #3990: "multi-thousand node NameNode get unresponsive”

Latent bugs that are **scale dependent**, whose symptoms surface in large-scale deployments, but not necessarily in small/medium scale deployments.
Root causes

Over thousands of bug repositories analyzed, our insights reveal that scalability bugs are related to:

- CPU-intensive nested loops on scale-dependent data structures (12 bugs)
- Disk IO loops (20 bugs)
  - Heavy IO inside scale-dependent loop
- Locking-related loops (8 bugs)
  - Lock inside scale-dependent loop
  - scale-dependent loop inside lock
State of the art: Real scale testing

“For Apache Hadoop, testing at thousand-node scale has been one of the most effective ways of finding bugs, but it’s both difficult and expensive. It takes considerable expertise to deploy and operate a large-scale cluster, much less debug the thousands of dollars an hour, making scale testing impossible for the solo contributor. As it stands, we are heavily reliant on test clusters operated by large companies to do scale testing. A way of finding scalability bugs without requiring running a large-scale cluster would be extremely useful.”

— Andrew Wang (Cloudera and Apache Hadoop PMC Member and Committer).
Challenges

- **How to find scalability bugs?**
  - We propose a **static analysis based tool** to find potential scalability issues

- **How to test distributed systems in a single machine?**
  - We created a methodology to
    - Address **memory** bottlenecks
    - Address **CPU** bottlenecks
    - Address **network** bottlenecks

---

**Is Scale Your Enemy, Or Is Scale Your Friend?**

By John Ousterhout

**Although the nominal topic of the following paper is managing crash reports from an installed software base, the paper's greatest contributions are its insights about managing large scale systems.** Einchumona et al. as the scale of Windows deployment increased. As the number of Windows installation skyrocketed, so did the rate of error reports. In addition, the size and complexity of the Windows system increased, making it more difficult...
Scalability bugs: When 100 node testing is not enough [HotOS 17]

Find
Problematic code paths using static analysis

Record
Remove non-deterministic behaviour & reduce CPU usage

Test
Debug system scalability in a single machine
Preliminary results

- **c3831: Decommission**
  - Real
  - Colo
  - SC+PIL
  - High accuracy

- **c3881: Scale-Out**
  - Real
  - Colo
  - SC+PIL

- **c5456: Scale-Out**
  - Real
  - Colo
  - SC+PIL

- **c6127: Bootstrap**
  - Real
  - Colo
  - SC+PIL
Experiment Challenges

- Still need to compare our results against real deployment result?
  - Where to get the resources?

- How to deploy modified system over hundreds of nodes easily?
OpenStack

Description
This appliance deploys OpenStack Mitaka with DevStack over one controller node and a configurable number of compute nodes.

Launch Complex Appliance at CHI@UC
Launch Complex Appliance at CHI@TACC
Experiment Strategy

- Shared Filesystem
  - Install modified test system in one shared directory

- Internal DNS resolution
  - Ease of system configuration and experiment script

*Not enabled by default*
OpenStack: Shared FS

Create our own NFS Server!
OpenStack: Shared FS

NFS Server custom script:

```bash
#!/bin/bash
apt-get update
apt-get install -y nfs-kernel-server
echo "/home 10.1.1.0/24(rw,sync,no_root_squash,no_subtree_check)" >> /etc/exports
service nfs-kernel-server restart
```

NFS Client custom script:

```bash
#!/bin/bash
# apt install nfs-common  # installed by default in ubuntu
mount -t nfs 10.1.1.10:/home /home
```
OpenStack: Internal DNS

- VMs, by default, only accessible through IP
- Virtual network IP is assigned randomly from pool
- Easier for test system and experiment scripts to access/ssh node by hostname
OpenStack: Internal DNS

1. Edit `/etc/neutron/neutron.conf`
   ```
   dns_domain = ucare.edu.
   ```

2. Add `dns` to `extension_drivers` in the `[ml2]` section of `/etc/neutron/plugins/ml2/ml2_conf.ini`
   ```
   [ml2]
   extension_drivers = port_security,dns
   ```

3. Restart `neutron-server` & `neutron-dhcp-agent`
   ```
   ssh node-1.ucare.edu
   ssh node-1
   ```

https://docs.openstack.org/newton/networking-guide/config-dns-int.html
Result

- Spawn up to 256 small VM nodes over 20 machines of CHI@UC
- NFS for installing test system, configs, scripts, and collecting experiments data
- Access between nodes by hostname
Things to try in future

- Orchestrate using pre-built OpenStack image
- Save experiment setup in Volumes
- Easy traffic shaping via Neutron QoS
Thank you!
Questions?

http://ucare.cs.uchicago.edu