

Scalability Bugs: When 100-Node Testing is Not Enough

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



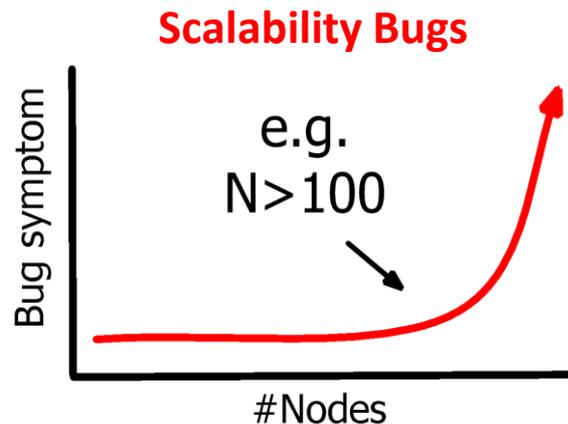


U of **C** systems research on
Availability, **R**eliability & **E**fficiency



Scale dependency?

- ❑ Cassandra
 - Bug #6409: "With **>500 nodes**, ... gossip protocol slow"
- ❑ Hadoop/HDFS
 - Bug #3990: "**multi-thousand node** NameNode get unresponsive"

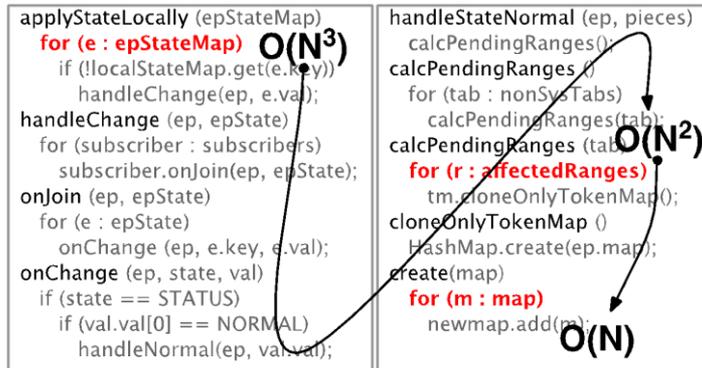


Latent bugs that are **scale dependent**, whose symptoms surface in large-scale deployments, but not necessarily in small/medium scale deployments



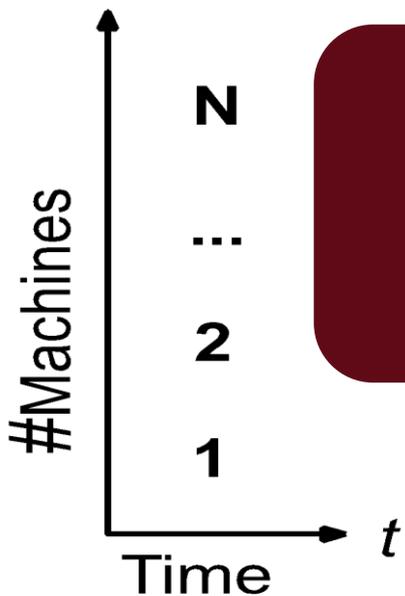
Root causes

- 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

Real-scale Testing



Is scale-testing on just one machine possible?

“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 **run a large cluster**, much less debug the **cluster**. **Debugging a large cluster costs thousands of dollars an hour**, for the solo contributor. As it **becomes more difficult to debug large test clusters operated by large companies**, **debugging bugs without requiring running a large-scale cluster would be extremely useful.**”

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

Challenges

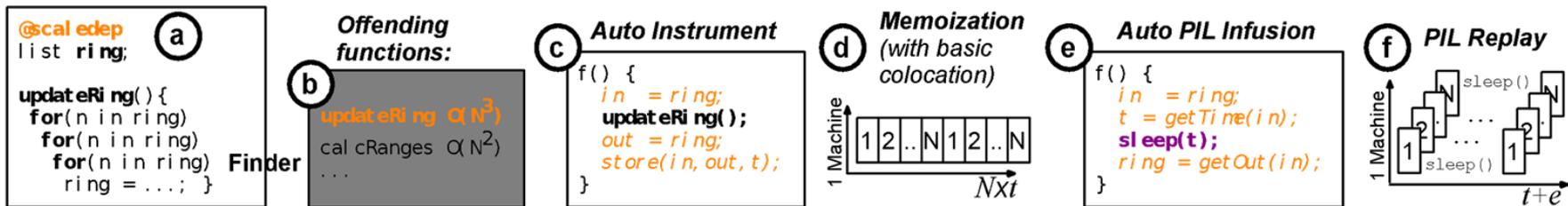
- ❑ How to **find** scalability bugs?
 - We propose a **static analysis based tool** to find potential scalability issues
- ❑ How to **test** distributed systems in a single machine?
 - We created a methodology to
 - Address **memory** bottlenecks
 - Address **CPU** bottlenecks
 - Address **network** 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. Kinchumana 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 dif-

Scalability bugs: When 100 node testing is not enough [HotOS 17]



Find



Record



Test

Problematic code paths using **static analysis**

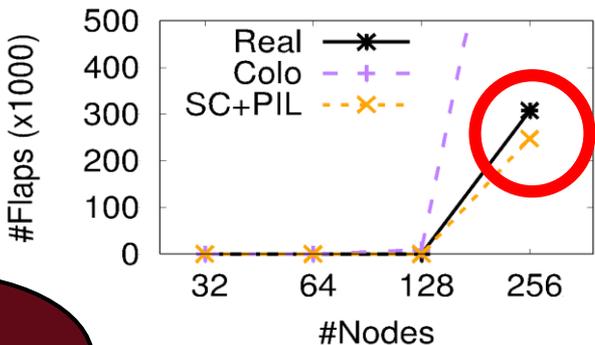
Remove **non-deterministic behaviour** & **reduce CPU usage**

Debug system scalability in a **single machine**

Preliminary results

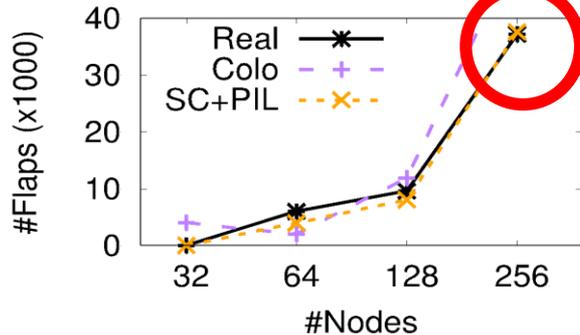
The bug symptom

c3831: Decommission

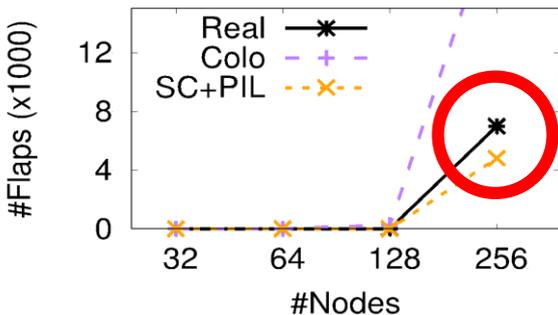


High accuracy

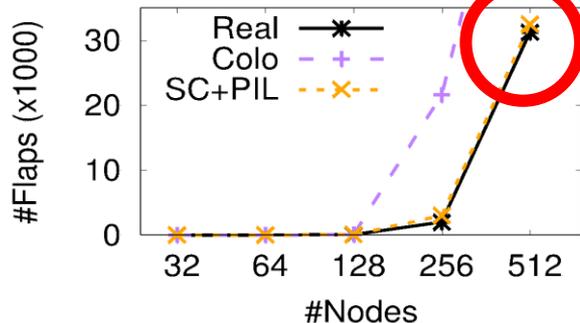
c3881: Scale-Out



c5456: Scale-Out



c6127: Bootstrap



Experiment Challenges

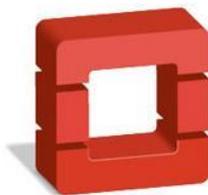
- ❑ Still need to **compare** our results against **real deployment** result?
 - Where to get the resources?
- ❑ How to **deploy** modified system **over hundreds of nodes** easily?





OpenStack

The screenshot shows a web browser window with the URL <https://www.chameleoncloud.org/appliances/30/>. The page features the Chameleon logo and a navigation menu. The main content area displays the breadcrumb "Appliances / OpenStack Mitaka (DevStack)" and the title "OpenStack Mitaka (DevStack)". Below the title, there are two green buttons: "Launch Complex Appliance at CHI@UC" and "Launch Complex Appliance at CHI@TACC". A "Description" section follows, stating: "This appliance deploys OpenStack Mitaka with DevStack over one controller node and a configurable number of compute nodes."



openstack™
CLOUD SOFTWARE

CHI@UC

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!

| <input type="checkbox"/> | Instance Name ▲ | Image Name | IP Address |
|--------------------------|-------------------------|-------------|-------------|
| <input type="checkbox"/> | nfshost | trusty-java | • 10.1.1.10 |
| <input type="checkbox"/> | node-1 | trusty-java | • 10.1.1.7 |
| <input type="checkbox"/> | node-2 | trusty-java | • 10.1.1.11 |
| <input type="checkbox"/> | node-3 | trusty-java | • 10.1.1.6 |
| <input type="checkbox"/> | node-4 | trusty-java | • 10.1.1.16 |
| <input type="checkbox"/> | node-5 | trusty-java | • 10.1.1.18 |
| <input type="checkbox"/> | node-6 | trusty-java | • 10.1.1.9 |
| <input type="checkbox"/> | node-7 | trusty-java | • 10.1.1.23 |
| <input type="checkbox"/> | node-8 | trusty-java | • 10.1.1.8 |
| <input type="checkbox"/> | node-9 | trusty-java | • 10.1.1.17 |
| <input type="checkbox"/> | node-10 | trusty-java | • 10.1.1.21 |



OpenStack: Shared FS

NFS Server custom script:

```
#!/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:

```
#!/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 [m12] section of /etc/neutron/plugins/ml2/ml2_conf.ini

```
[m12]  
extension_drivers = port_security,dns
```

3. Restart neutron-server & neutron-dhcp-agent

```
ssh node-1.ucare.edu  
ssh node-1
```

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>