Scaling CloudStack

to 100K Hosts and Millions of Instances

Image from freepik.com

Abhishek Kumar

About ME

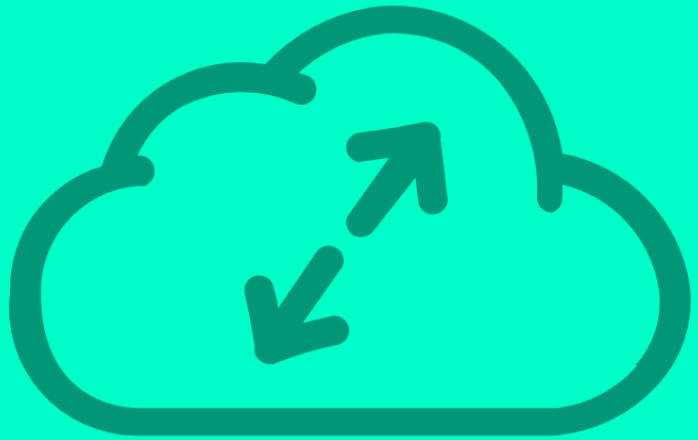
- Long-time CloudStack Committer and PMC
- Software engineer @ ShapeBlue
- Father to a delightful 8-month-old baby girl
- When I’m not working, you’ll find me tending to my ever-growing jungle of houseplants

HOw MANY HYPERVISOR HOSTS CLOUDSTACK CAN SUPPORT???

How many virtual machines, volumes and other resource?

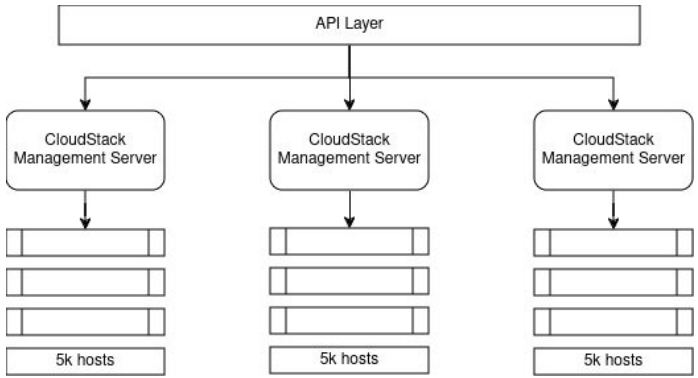
CLOUDSTACK & SCALABILITY

- CloudStack is scalable
- Many users already running production environment with over 5K hypervisor hosts
- Alternative to hyperscalers
- Identify limits



SCALING THROUGH CELL MODEL

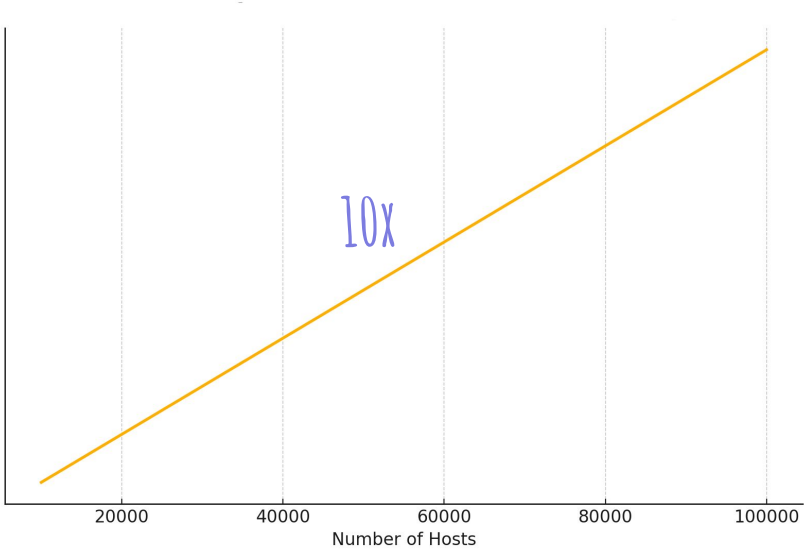
- Multiple CloudStack installations
- Each installation with 5-10k hosts will act as cell



CHALLENGE

Inter-cell communication

SCALE SINGLE INSTALLATION



- Deploy a test environment
  - Simulator based environment
  - Connected agents such as KVM - How to access thousands of KVM hosts?
- How to identify issues, bottlenecks?

- Database
- Concurrency

  Management
- Fault Tolerance
- Agent Communication
- Other issues

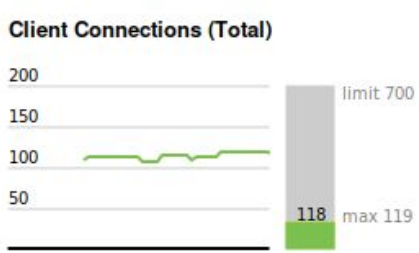
Database

- As the number of resources (e.g., VMs, networks) grow, the database becomes a primary bottleneck
- Centralized database

Use distributed database? A locking service?

DATABASE - CONTD...

MySQL server itself isn't the problem!
It can handle upwards of 50k queries per second



- Old structure and usage, old code
- Inefficient usage - missing indexes, slow queries, repetitive queries

API HANDLING and Concurrency Management

- High API request volumes from many users
- Automation scripts and background tasks

Self-Healing and Fault Tolerance

- Area where CloudStack has not fared very well in the past
- Intermittent failures shouldn’t affect whole environment

Agent Communication Overhead

- Managing many thousands of agents communicating with the CloudStack management server
- Bottlenecks during management server rolling restarts
- Access thousands of hypervisor hosts to create such a large environment

Other ISSUES & CONSIDERATIONS

- Resource Allocation and Scheduling
- Periodic maintenance
- Tuning - underlying infrastructure, CloudStack features and functionalities
- Security at scale


IMPROVEMENTS

IMPROVEMENTS & CHANGES

- Exhaustive changes but focus on the **persistence layer** and infrastructure resources.
- Primary emphasis on refining **KVM hypervisor** integration.
- Targeted improvements in API and server layers for immediate gains.
- Benchmarking and profiling to identify bottlenecks and achieve peak resource utilization.

Management Server Host

- Heap size set to 10GB but memory isn’t scaling well
- Many threads in blocking state

BEFORE

Management Server Host

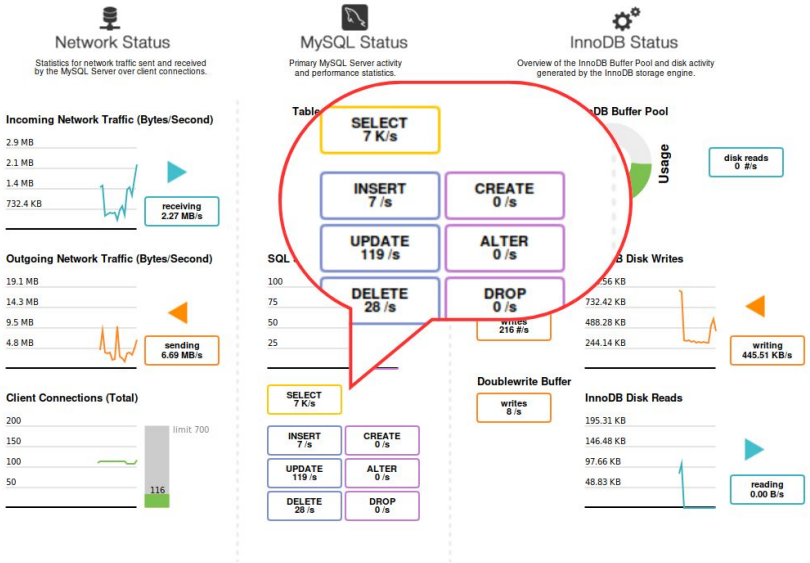
After

- Memory is scaled better
- Blocking threads are minimal

HikariCP & DATABASE CHANGES

- Default connection pooling library changed from DBCP2 to more performant HikariCP, https://github.com/brettwooldridge/HikariCP
- Option to configure library using db.properties
- Added proper indexes
- Improved JAVA code at different places

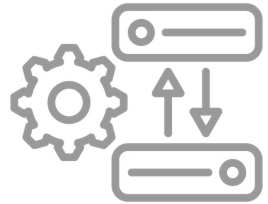
Before



After

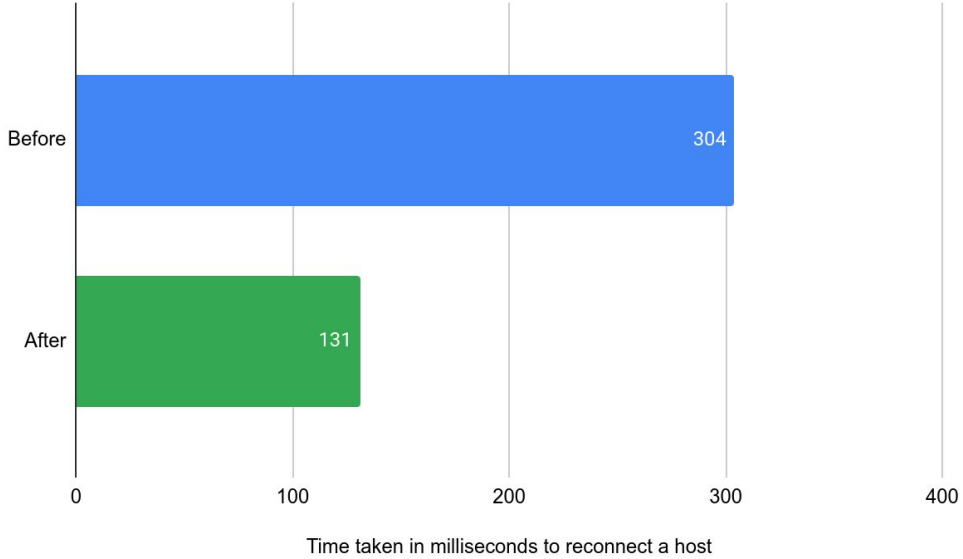
Significant reduction in DB reads

AGENT-SERVER COMMUNICATION



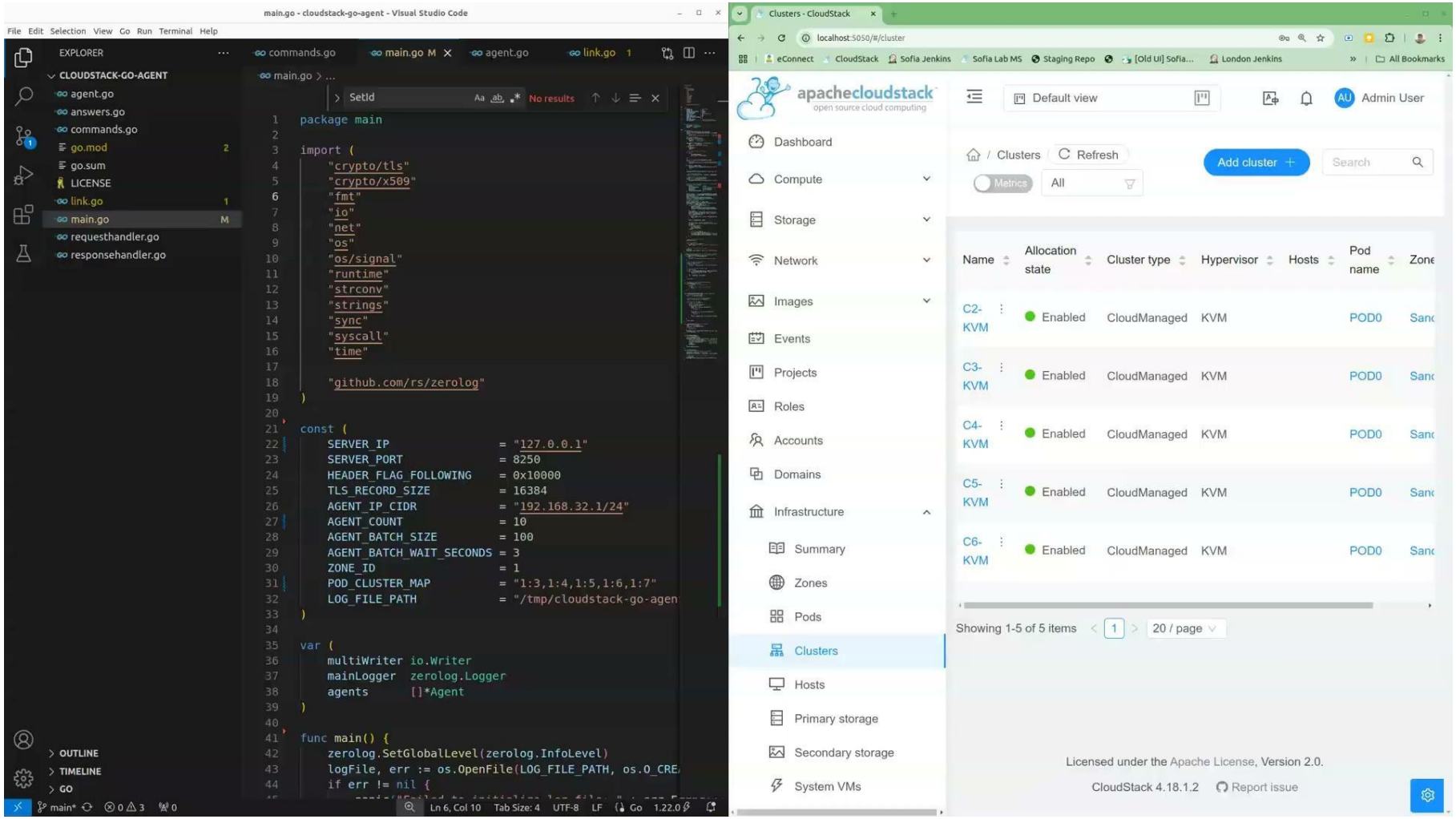
- Better handling connections
- Concurrency in TLS/SSL handshakes
- Configuration flexibility

A **new** mock KVM agent plugin developed



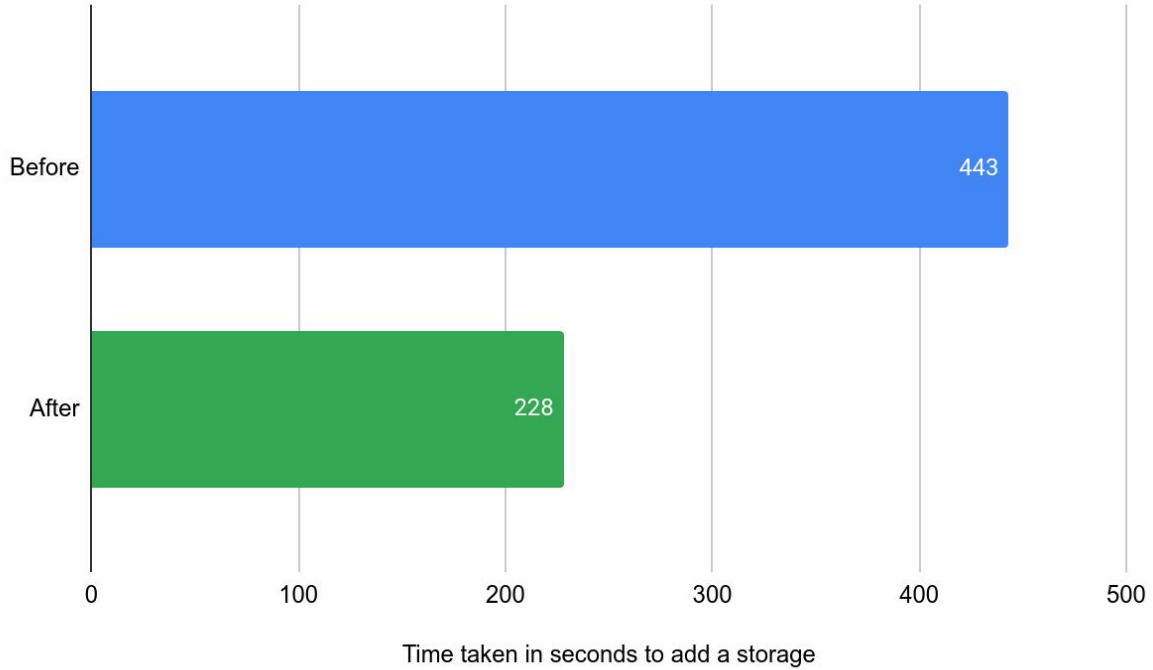
...2 MOCK KVM AGENT PLUGINS

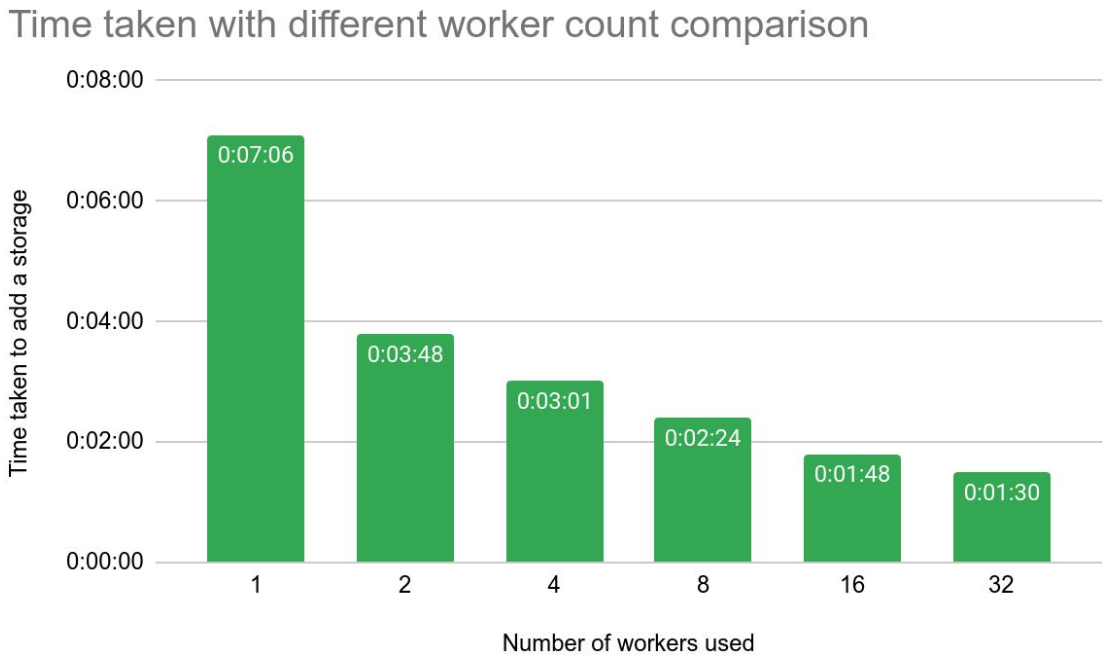
- JAVA based using existing agent code
- Go-based agent written from scratch


main.go - cloudstack-go-agent - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
CLOUDSTACK-GO-AGENT
agent.go
answers.go
commands.go
go.mod
go.sum
LICENSE
link.go
main.go
requesthandler.go
responsehandler.go
commands.go
main.go M
agent.go
link.go 1
main.go > ...
SetId
No results
package main
import (
"crypto/tls"
"crypto/x509"
"fmt"
"io"
"net"
"os"
"os/signal"
"runtime"
"strconv"
"strings"
"sync"
"syscall"
"time"
"github.com/rs/zerolog"
)
const (
SERVER_IP = "127.0.0.1"
SERVER_PORT = 8250
HEADER_FLAG_FOLLOWING = 0x10000
TLS_RECORD_SIZE = 16384
AGENT_IP_CIDR = "192.168.32.1/24"
AGENT_COUNT = 10
AGENT_BATCH_SIZE = 100
AGENT_BATCH_WAIT_SECONDS = 3
ZONE_ID = 1
POD_CLUSTER_MAP = "1:3,1:4,1:5,1:6,1:7"
LOG_FILE_PATH = "/tmp/cloudstack-go-agen
)
var (
multiWriter io.Writer
mainLogger zerolog.Logger
agents []*Agent
)
func main() {
zerolog.SetGlobalLevel(zerolog.InfoLevel)
logFile, err := os.OpenFile(LOG_FILE_PATH, os.O_CRE
if err != nil {
OUTLINE
TIMELINE
GO
main*
Ln 6, Col 10
Tab Size: 4
UTF-8
LF
Go
1.22.0
Clusters - CloudStack
localhost:5050/#/cluster
eConnect
CloudStack
Sofia Jenkins
Sofia Lab MS
Staging Repo
[Old UI] Sofia...
London Jenkins
All Bookmarks
apachecloudstack
open source cloud computing
Default view
Admin User
Dashboard
Compute
Storage
Network
Images
Events
Projects
Roles
Accounts
Domains
Infrastructure
Summary
Zones
Pods
Clusters
Hosts
Primary storage
Secondary storage
System VMs
Clusters
Refresh
Metrics
All
Add cluster
Search
Name
Allocation state
Cluster type
Hypervisor
Hosts
Pod name
Zone
C2-KVM
Enabled
CloudManaged
KVM
POD0
C3-KVM
Enabled
CloudManaged
KVM
POD0
C4-KVM
Enabled
CloudManaged
KVM
POD0
C5-KVM
Enabled
CloudManaged
KVM
POD0
C6-KVM
Enabled
CloudManaged
KVM
POD0
Showing 1-5 of 5 items
1
20 / page
Licensed under the Apache License, Version 2.0.
CloudStack 4.18.1.2
Report issue

STORAGE CONNECTIONS

- Added concurrency and allow setting worker counts using new global config - **storage.pool.host.connect.workers**
- **50k** hypervisor hosts present in the zone and set worker count to 2

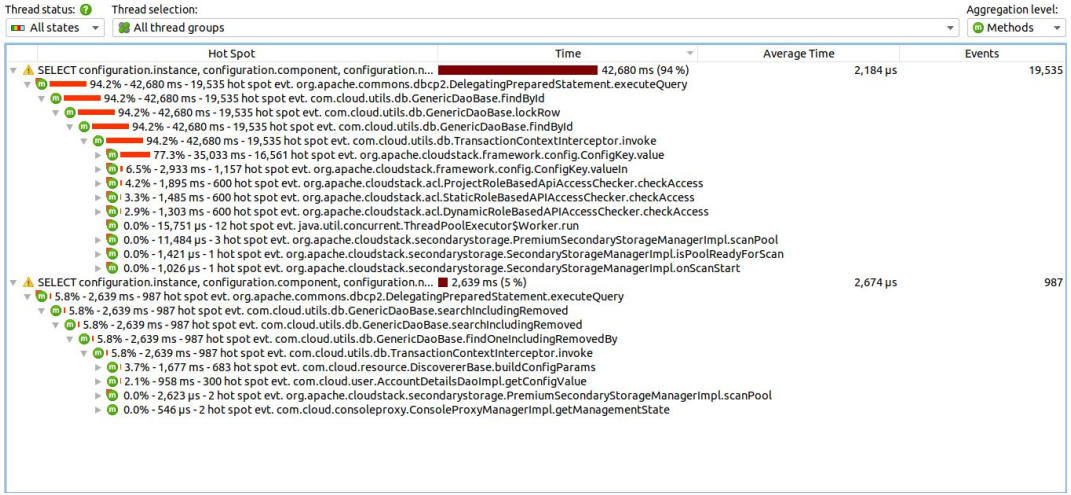




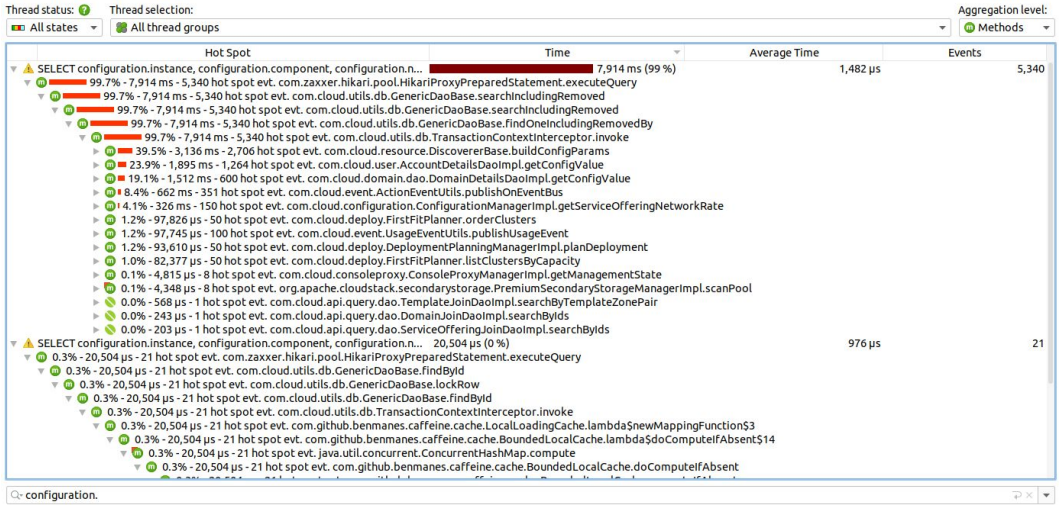
More workers will not always result in more performance

Caching framework

- Not everything can be made faster on its own—sometimes we need caching to bridge the gap
- Caffeine in-memory caching library, https://github.com/ben-manes/caffeine
- Caching added for repeated retrievals:
  - Dynamic Config Keys
  - Account/User role API access



BEFORE

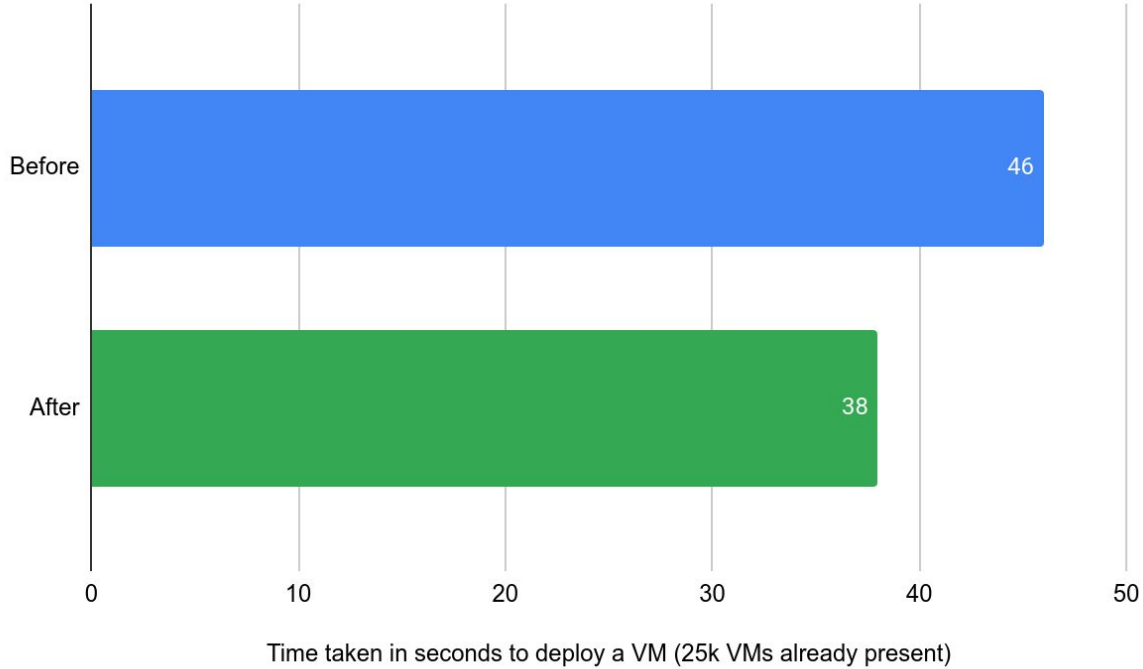


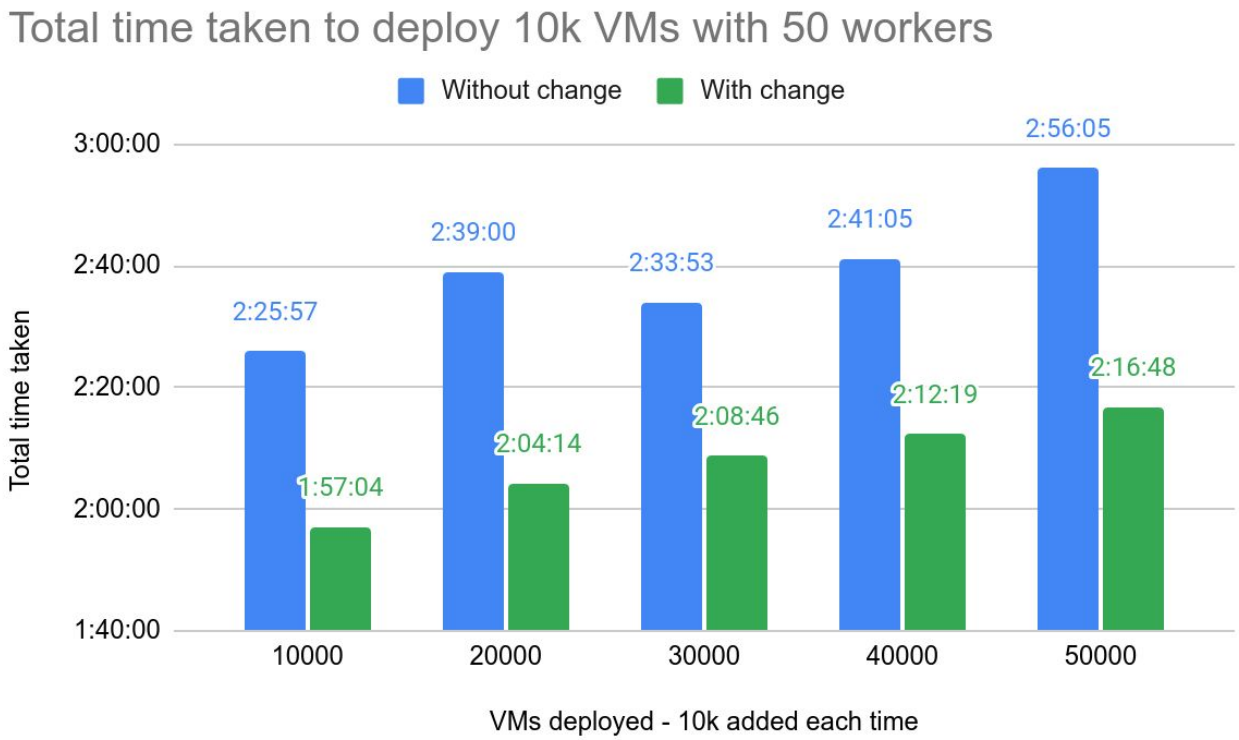
AFTER

- Config value retrievals down to 1/4th
- 30% lesser average retrieval time

VM AND DEPLOYMENTS

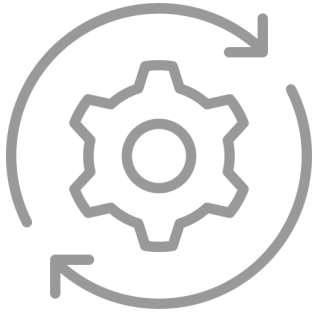
- While deployment was not the focus but overall changes resulted in gains
- **50k** hypervisor hosts present in the zone and VMs deployed with 50 workers





- Good part - lesser time after changes
- Bad part - linear increase

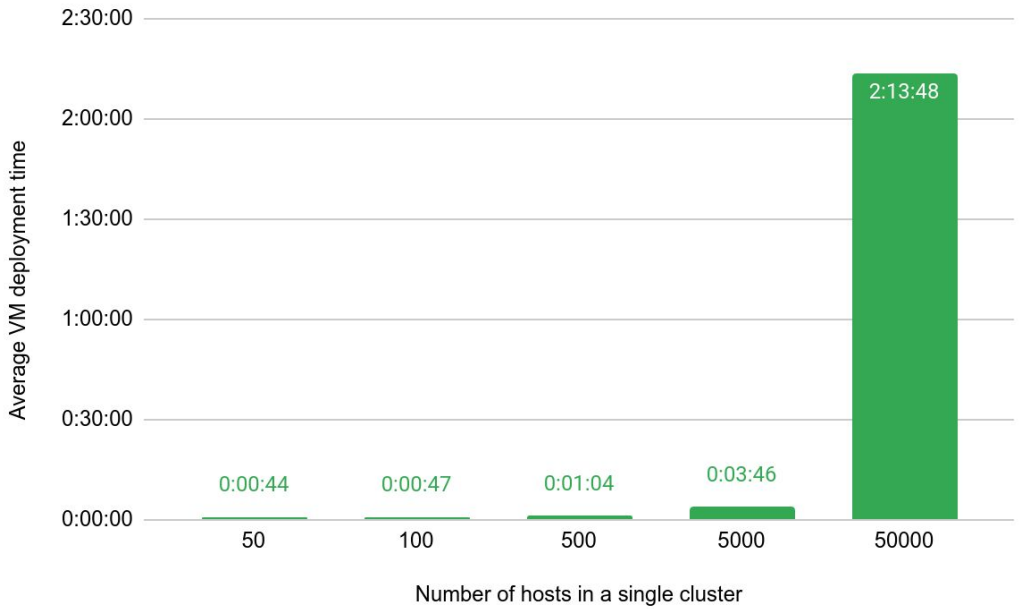
TUNING



- Identified configurations which can be tuned for optimal performance. These include:
  - Database configurations
  - Global settings
  - JVM tuning - moved to using G1GC instead of ParallelGC
  - OS-level changes#
- Environment recommendations which would allow better performance

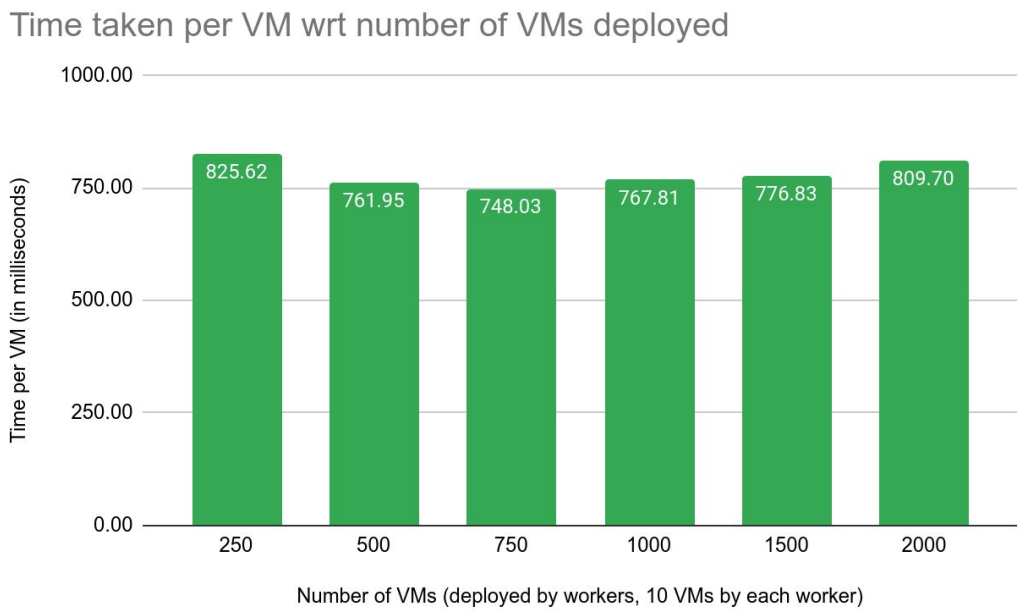
CLUSTER-HOST BEHAVIOUR

- Performance becomes inversely proportional to number of hosts in a cluster
- # On putting 50k hosts in a single cluster only one VM got deployed



WORKER BEHAVIOR

- Test server was comfortable handling 50-100 workers



Work DOne - contD...

- 4.20.0
  - Move to more performant HikariCP database connection pooling library, https://github.com/apache/cloudstack/pull/9518
  - Introduced caching framework and dynamic config key caching using Caffeine library, https://github.com/apache/cloudstack/pull/9628
- 4.20.1#
  - Larger scaling work around caching usage, agent-server connection improvements, concurrency, optimisations. https://github.com/apache/cloudstack/pull/9840
  - list*Metrics API related UI improvement, https://github.com/apache/cloudstack/pull/9825
  - Management server maintenance, https://github.com/apache/cloudstack/pull/9854
  - And more…

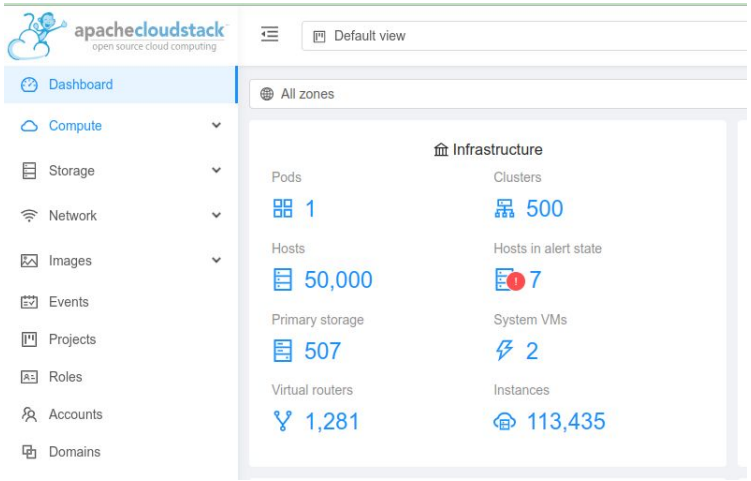
FUTURE WORK

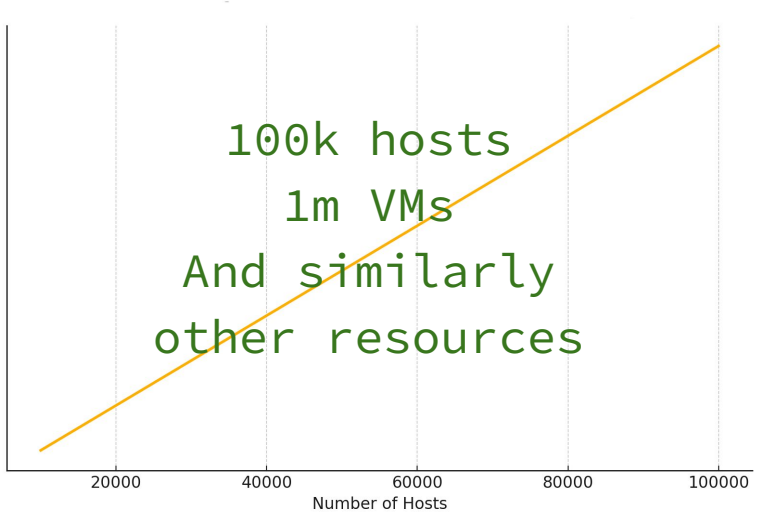
FUTURE WORK

- Extend optimization efforts to domains, accounts, VMs, volumes and other resources
- Optimize the allocation and deployment logic
- Develop comprehensive guidelines for architecting and tuning CloudStack environments
- Enhance the efficiency of background tasks
- Incorporating modern design patterns and re-engineering legacy modules

HOw MANY HYPERVISOR HOSTS CLOUDSTACK CAN SUPPORT???

How many virtual machines, volumes and other resource?





Q & A

Thank you!