cloud STACK COLLABORATION CONFERENCE

CloudStack Scalability

Our observations in the field scaling from 20 to 20,000 Hosts

About me

QA Manager at Shapeblue

With Cloudstack since 2016

Dad of 2 girls

Background

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…

- Last year at CCC, Abhishek talked about scalability and performance improvements*.
- Customer stories
  - Telco 3000+ hosts
  - Telco ~200 hosts
  - Tech company – 15-20k hosts

* https://www.cloudstackcollab.org/wp-content/uploads/2024/12/CCC-2024-Scaling-CloudStack-Abhishek-Kumar.pdf

Before We Start

These are patterns we've seen repeat in real environments and simulation

Hosts is not the only aspect of scaling

Scaling Journey: Tree Key Phases

Foundation – Up to 200 hosts

- Hardware
- Basic System Setup – Man/DB/Network and Storage
- Resource Monitoring

Optimization – Up to 2k hosts

- Architectural Changes
- DB Optimization
- Network & Storage
- Data Purging

Full Scale – Up to 20k hosts

- DB and General Pooling
- API Concurrency
- Agent Communication
- Resource Allocation & Placement

Phase 1: Scaling to 200 Hosts

cloud STACK COLLABORATION CONFERENCE

Basic Infrastructure Setup

Storage Solution

Hardware – Hosts/Network Devices

Resource Distribution and Monitoring

Basic System Tuning

Management Server and Database

- Niente

Network

Network Configuration
Advanced network with traffic segmentation:

- Guest
- Management
- Public
- Storage

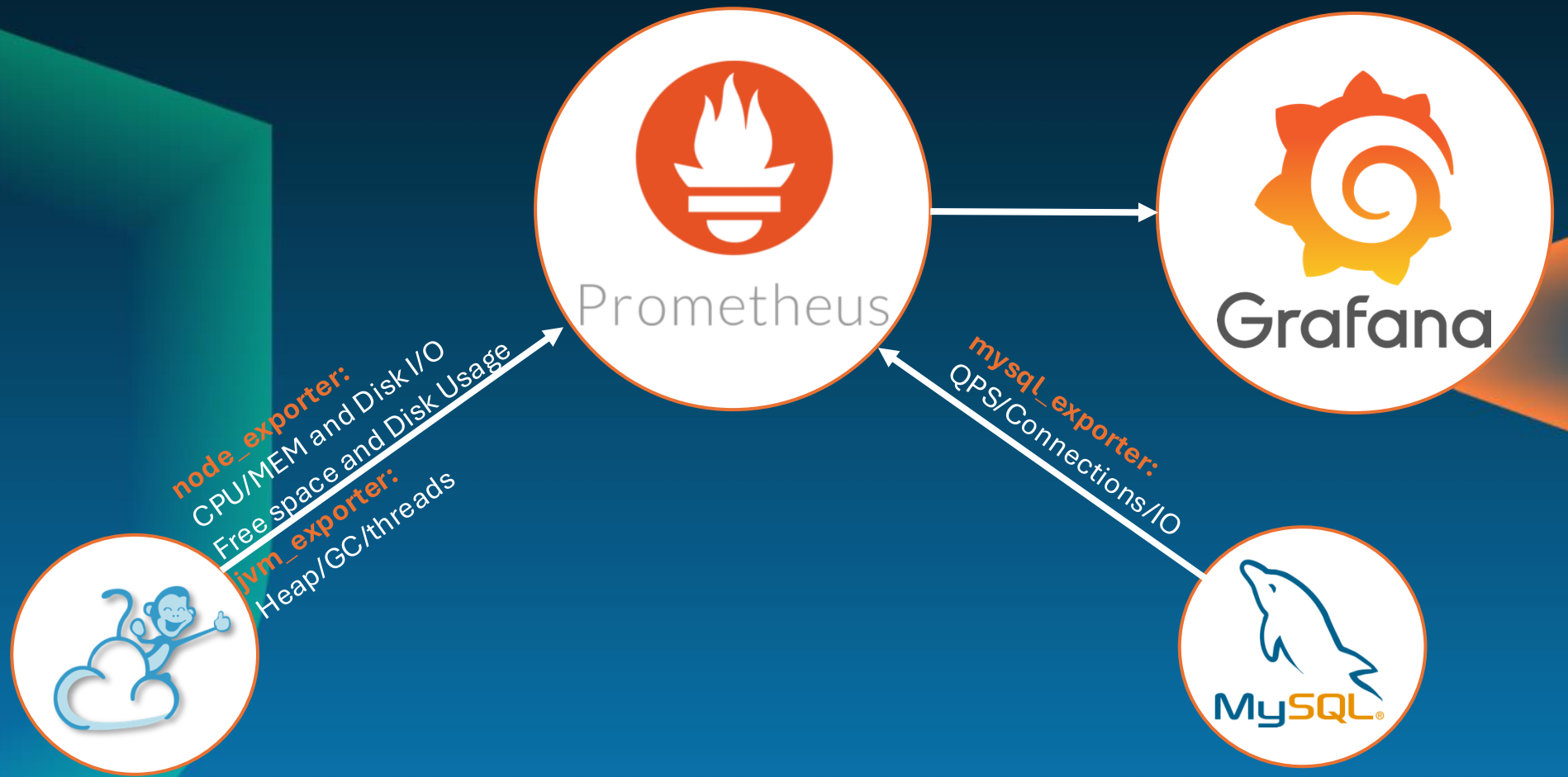
Storage

Storage – consider NFS

- mount options: mount -t nfs -o vers=3,tcp,rsize=65536,wsize=65536,noatime,nodiratime,nfsvers=3,timeo=600
- Server side: in nfs.conf:
  [nfsd]
  threads=128

Resource Monitoring

Management Server and Database Monitoring



Resource Monitoring

Hypervisor and User Instance Monitoring

CloudStack Native Metrics View:

- **Hosts:**
  - CPU Usage
  - Memory Usage
- **User Instances:**
  - CPU usage
  - Memory usage
  - Network traffic usage

CloudStack Alerts:

- Agent disconnects
- Capacity threshold
- Primary Storage usage
- Secondary Storage usage

Phase 1: Summary

Go with stock installation

No actual scalability changes required

Phase 2: Optimizations

- Architectural changes on our deployment
- Database optimizations
- Network and Storage optimizations
- Purging/Managing large data

Architecture Changes

- Management server and DB separation

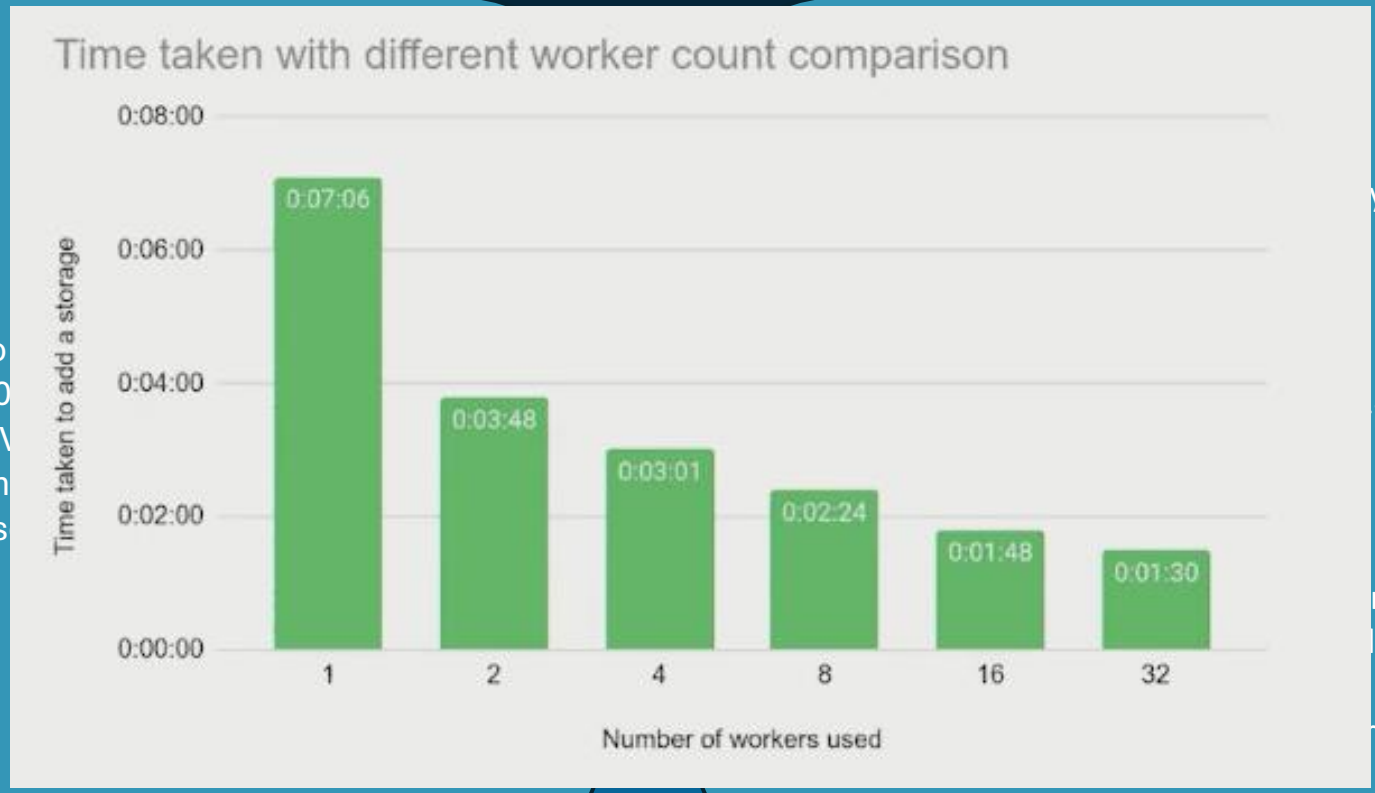
- Key benefits
  - Distributed arch for improved scalability
  - Dedicated HW resource/ avoid noisy neighbor

Network and Storage

Network

Advance Netwo
- Enables 100
- Implement V
- Dedicated n

Consider SDN s



Storage

ypes

ing level

nect=8
ld.perhost=

kers = 4

Database Optimization

- MySQL itself can handle up to 50k QPS
  - Slow Queries and Fragmented tables
- innodb_buffer_pool_size = 60–70% of total RAM.
- innodb_log_file_size = 1G
- Add indexes on large tables (e.g., vm_instance, async_job, event, host)
- Run ANALYZE TABLE and OPTIMIZE TABLE periodically
- Implement purging strategy

Purge Strategy

Scheduled Purges:

- expunged.resource.purge.job.delay = 180
- expunged.resources.purge.batch.size = 50
- expunged.resources.purge.delay = 300
- expunged.resources.purge.enabled = true
- expunged.resources.purge.interval = 86400
- expunged.resources.purge.keep.past.days = 30
- expunged.resources.purge.resources = [leave empty]
- expunged.resources.purge.start.time = yyyy-MM-dd HH:mm:ss

On demand Purging:

```
cmk purge expungedresources startdate=2024-04-15 enddate=2024-04-20
resourcetype=VirtualMachine

cmk remove rawusagerecords interval=60
```

Summary

Systems are large enough that small issues start to matter

Introduce changes in architecture and optimise storage and networks

Apply changes and take advantage of monitoring

Phase 3: Full Scale

Database and General Pooling

API Concurrency and Performance

Agent Communication Issues

GC Hiccups and Stalling

Database – Pooling and concurrency

Connection pooling - in db.properties:

- db.<DATABASE>.connectionPoolLib = DBCP2 -> HikariCP

Repeated queries and thread concurrency

In my.cnf

- query_cache_size = 64M and query_cache_type = 1 for repetitive queries like listVirtualMachines;
- thread_concurrency = 2 * CPU cores (e.g., 64 for 32-core); boosts parallel query execution

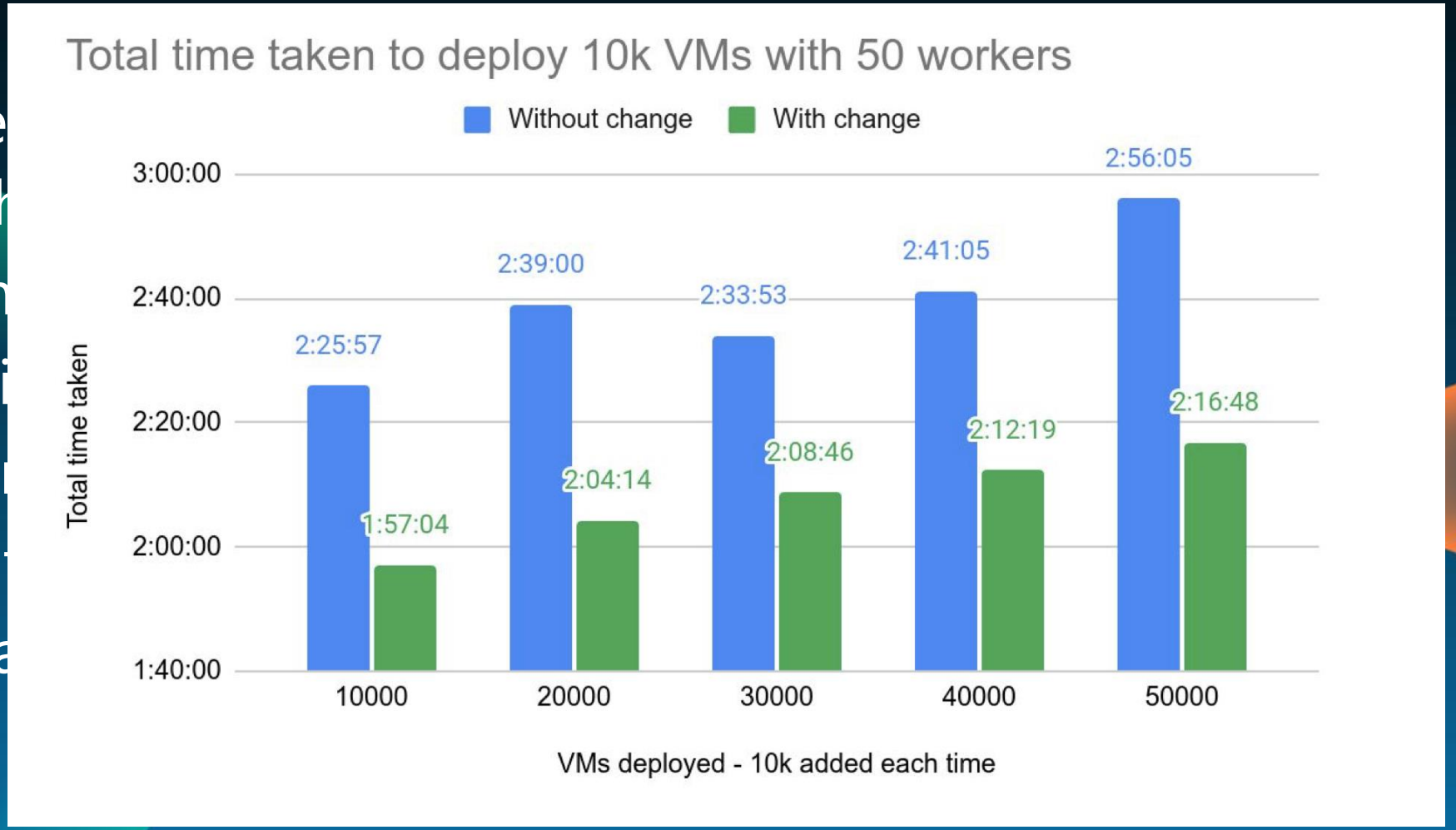
General Pooling

Resource and Stats Pooling

- ping.interval and ping.timeout – tune host health check frequency
- capacity.calculate.workers – parallelize capacity checks
- storage.pool.host.connect.workers – speed up storage attach operations
- vm.stats.interval and vm.sync.interval – reduce collection overhead
- vm.sync.power.state.transitioning=false – skip noisy transitions
- storage.stats.interval – control polling frequency
- network.gc.interval, ipaddress.gc.interval – adjust cleanup cycles

Caching Framework

Some
- Cach
- In m
- Avoi
- Impl
- Con
- Insta



* https://github.com/apache/cloudstack/pull/9628

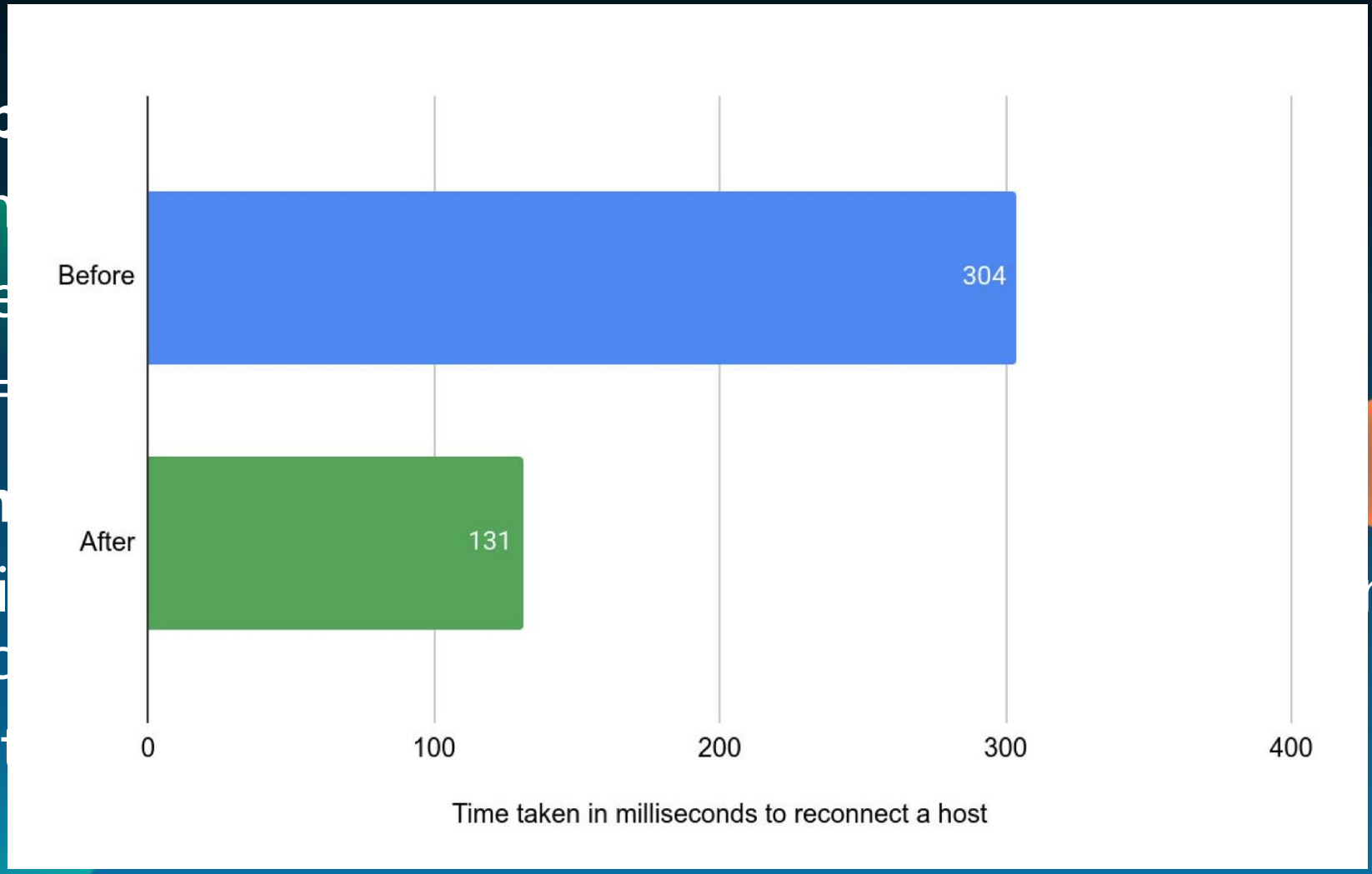
Agent Communication

Load b

- Use n
- indire
- host=

Comm

- 4.20 i ncurrency and c
- agent



JVM GC algo

At large scale ParallelGC starts to stall

- Switching from ParallelGC to G1GC

In /etc/default/cloudstack-management

```
JAVA_OPTS="$JAVA_OPTS -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -
XX:+ParallelRefProcEnabled”
```

```
systemctl restart cloudstack-management
```

- Advantages:
  - More predictable API latency
  - Coping better under burst load

Final thoughts

- Phase 1: Keep it simple and build solid foundation
- Phase 2: Start shaping the performance for predictable behavior
- Phase 3: Fine tuning of pools, threads, and GC to avoid stalls

- These are not rules or full guide
- Take this as a starting point, not a final solution

cloud STACK COLLABORATION CONFERENCE

Q&A

Tell us your story!