Twine: A Unified Cluster Management System for Shared Infrastructure

Chunqiang Tang, **Kenny Yu***, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long Cheng, Ben Christensen, Alex Gartrell, Maxim Khutornenko, Sachin Kulkarni, Marcin Pawlowski, Tuomas Pelkonen, Andre Rodrigues, Rounak Tibrewal, Vaishnavi Venkatesan, and Peter Zhang

FACEBOOK Infrastructure



Data center geographic regions







3

📍 Luleå, Sweden

Singapore



Ľ 1 **News Feed**





Search



Cache

What design decisions did Twine make differently?

Decision 1

Dynamic machine partitioning

over

static clusters

Decision 2

Customization in shared infrastructure

over

private pools

Decision 3

Small machines

over

big machines

What design decisions did Twine make differently?

Decision 1

Dynamic machine partitioning

over

static clusters

Decision 2

Customization in shared infrastructure

over

private pools

Decision 3

Small machines

over

big machines

Kubernetes Documentation / Getting started / Best practices / Building large clusters

Building large clusters

Support

At v1.19, Kubernetes supports clusters with up to 5000 nodes. More specifically, we support configurations that meet *all* of the following criteria:



Stranded capacity: M6, M7, M8 are available, but jobs in Cluster 1 cannot use them

M1	M2	
M3	M4	M
Cluster 1		



Stranded capacity: M6, M7, M8 are available, but jobs in Cluster 1 cannot use them

	uition: No signme	
M3	M4	
Cluste	er 1	

achine amic.











Use M5 in newly-constructed Data Center 3 to improve spread for fault tolerance

How does Twine perform fleet-wide optimization for an entire geographic region?



How does Twine perform fleet-wide optimization for an entire geographic region?



Shard Twine Scheduler by entitlements

How well does the Twine scheduler scale?





How do we mitigate risks with 1M machines per deployment?



gradual and frequent releases

run dependencies

run itself

network redundancy

Network partitions within a region

recurring large-scale failure test

What design decisions did Twine make differently?

Decision 1

Dynamic machine partitioning

over

static clusters

Decision 2

Customization in shared infrastructure

over

private pools

Decision 3

Small machines

over

big machines

Private pools or shared infrastructure?





Private pools or shared infrastructure?



M6 and M7 are underutilized and cannot be shared across private pools



Private pools or shared infrastructure?



M6 and M7 are underutilized and cannot be shared across private pools



M6 and M7 can be reused to run other workloads





What are the challenges with supporting ubiquitous shared infrastructure?

Challenge 1

Performance wins through host customization

What is host customization?

NIC settings

CPU Turbo

sysctls (e.g., huge pages, kernel scheduler settings)

kernel version

file system settings (e.g., btrfs, XFS)

What is host customization?

Intuition: Dynamically reconfigure machines as needed by workloads.

Challenge: Performance wins through host customization



Challenge: Performance wins through host customization

Automatically reconfigure M3 host settings



What is the overhead for host profile switches?

On average, a machine changes host profiles once every ~2 days





What drives host profile changes?



What drives host profile changes?



What drives host profile changes?

Change machines to run

Shared infrastructure makes machines fungible and improves efficiency!

Hour of Day

Change machines to run non-ALM-tracking services



What are the challenges with supporting ubiquitous shared infrastructure?

Challenge 1

Performance wins through host customization



Tasks are not homogenous

Approach

Host profiles

Challenge 2

Challenge: Tasks are not homogenous



2 3

Need at least 1 replica of each shard up at all times





Challenge: Tasks are not homogenous



2

Task 4 becomes unavailable.









3

Challenge: Tasks are not homogenous



Task 1 restarted for software release. Shard C unavailable!




Challenge: Tasks are not homogenous



software release. Shard C unavailable!





How does Twine collaborate with applications?



How does Twine collaborate with applications? TaskController State С В Α Α С В 2 4 3 4 unavailable Scheduler **TaskController S0** Update Job Time request=[**1,2,3,4**] completed=[] **S1** ack=[2] **S2 S**3



2 and 3 have no overlapping shards with 4, but overlap in B. Can update only one of these.



How does Twine collaborate with applications? TaskController State С В A Α С В 2 3 4 4 unavailable Scheduler **TaskController S0** Update Job Time request=[**1,2,3,4**] completed=[] **S1** ack=[2] completed=[2] request=[1,3,4] Can only update 3 to **S2** maintain C's availability ack=[3] **S**3 40





How does Twine collaborate with applications? TaskController State С В A Α С В 2 4 3 4 recovered! Scheduler **TaskController S0** Update Job Time request=[**1,2,3,4**] completed=[] **S1** ack=[2] request=[1,3,4] completed=[2] **S2** ack=[3] **S**3



How does Twine collaborate with applications? TaskController State С В Α Α С В 2 4 3 Scheduler **TaskController S0 Update Job** Time request=[1,2,3,4] completed=[] **S1** ack=[2] completed=[2] request=[1,3,4] **S2** ack=[3] request=[1,4] completed=[3] **S**3 ack=[1]



Can update 1 or 4, but they overlap in C. Can update only one of these

What are the challenges with supporting ubiquitous shared infrastructure?

Challenge 1

Performance wins through host customization



Tasks are not homogenous

Approach

Host profiles



TaskControl API

Challenge 2

Approach



How easy is it to migrate onto shared infrastructure?

Case study with PGx: O(100s) services, O(100Ks) machines

800 Entitlements, dense stacking

3 TaskControllers 3 Host profile settings

How easy is it to migrate onto shared infrastructure?

Case study with PGx: O(100s) services, O(100Ks) machines

0% to 70% capacity in twshared in 9 months. Er No new customizations needed. der

Power is our most constrained resource







47

📍 Luleå, Sweden

Singapore

What design decisions did Twine make differently?

Dynamic machine partitioning

over

static clusters

Decision 1 Decision 2

Customization in shared infrastructure

over

private pools

Decision 3

Small machines

over

big machines

Big machines or small machines?

Big machines

- 2 CPUs, 20 cores each
- 256GB RAM
- Dedicated NIC
- 30 machines per rack



Small machines

- 1 CPU, 18 cores
- 64GB RAM
- 4 machines share 1 multi-host NIC
- 92 machines in a rack



A small-machine rack vs. a big-machine rack

- 38% more cores
- 34% less memory

Why use small machines?

Existing large services are optimized to fully utilize whole machines

Resource control historically not mature

Not a public cloud: no external users

How much do we save by using small machines?

Relative total cost of ownership



How much do we save by using small machines?

Relative total cost of ownership



18% power savings fleet-wide. 17% savings in total cost of ownership.



What lessons did we learn using small machines?

Stacking tasks is less effective

More work required when working with hardware vendors Large effort to rearchitect services to fit into less memory

Conclusion

Evolving Twine over the past 10 years

Dynamic machine partitioning

Avoids stranded capacity in isolated clusters and enables fleet-wide optimizations

Customization in shared infrastructure

Support ubiquitous shared infrastructure to improve efficiency without sacrificing workload performance or capability

Small machines

Achieve higher power efficiency globally

Thank you!

Chungiang Tang, **Kenny Yu***, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long Cheng, Ben Christensen, Alex Gartrell, Maxim Khutornenko, Sachin Kulkarni, Marcin Pawlowski, Tuomas Pelkonen, Andre Rodrigues, Rounak Tibrewal, Vaishnavi Venkatesan, Peter Zhang, and everyone who contributed to Twine's development over the past decade.

Contact: kennyyu@fb.com



FACEBOOK Infrastructure