Market-Driven Resource Allocation in Rack-Scale Systems

Muli Ben-Yehuda Lior Segev Ariel Maislos
Etay Bogner Ron Asher

Stratoscale

The Stratoscale Rack-Scale Hyper-Converged System

- Hyper-converged x86 system: compute + storage + networking
- Scales from 4–1000 nodes
- Resources: CPU cycles, memory, network and storage bandwidth
- Cloud-like: many rational and selfish users
- Distributed storage and distributed memory
- Fast live migration of workloads between nodes

The Problem

How do you allocate resources efficiently?

Efficient Resource Allocation

Give the right resource to the right workload at the right time

Difficulty Increases As Load Rises

Metrics

- System utilization
- Fairness
- Responsiveness
- Useful work


MAXIMUM
SATISFACTION

Potential Solutions

1. Fixed static allocation
   - \+ Simplest possible approach
   - \- Workloads come and go, demands change
2. Let the users decide
   - \+ Users know best
   - \- Rational users will game the system to get more
3. Monitor behavior and give suffering workloads more resources
   - \+ Fair
   - \- Breaks when load increases: someone will have to suffer
   - \- Reactive not proactive: alleviates suffering but does not prevent it
   - \- Rational users can still game the system

So what should we do?

Let The Market Work It Out

RackCoins: The Virtual Rack Currency

Workloads use RackCoins to pay for resources

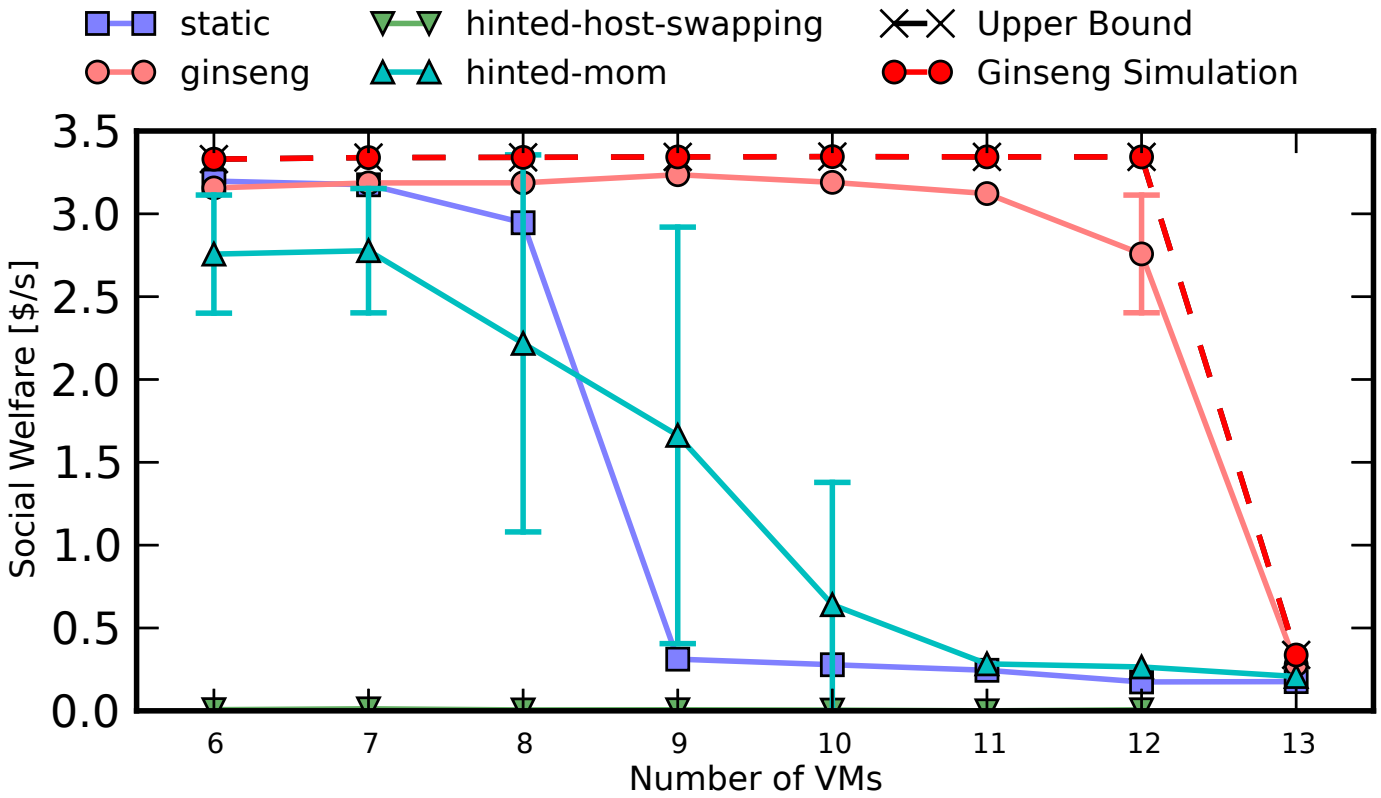
RackCoins: Where Do They Come From?

- Each workload has a RackCoins budget
- A workload may buy RackCoins using real currency (e.g., €)
- A workload may be granted RackCoins by the system's admin
- When the budget runs out, the workload or the admin need to replenish it or the workload will stop running

RackCoins: How Do You Spend Them?

- Workloads are guaranteed some minimal amount of resources
- Workloads bid for additional resources
- If they win, they use them
- If they lose, they can bid again next round
  - presumably with a higher bid
- The system collects bids, calculates the going market rate for each resource, decides who won, and allocates resources to the winning workloads
- The system migrates workloads between nodes to maintain a rough price equilibrium between nodes
- Goal: maximize satisfaction, not (necessarily) profit

Does Bidding Make Sense?



Ginseng [VEE'14] shows a 6.2x–15.8x improvement in social welfare (83%–100% of the optimum) when workloads bid for memory

Why Does It Make Sense?

- A properly designed bidding mechanism induces rational clients to tell the truth about their needs
- Truth telling enables the system to give the right resource to the right workload at the right time
- Utilization is maximized because no resource is wasted
- Fairness is maximized because the workload that needs a resource most gets it
- Useful work is maximized because the workload that does the most important work (i.e., needs a resource the most) gets it
- The users' satisfaction is maximized due to all of the above

A Bidding Workload's Woes

- Bidding requires workload awareness: How much are resources worth to me and how much should I bid?
- Bidding requires workload elasticity: How do I make do with less or more resources?
- Bidding requires profit-maximizing workloads: How do I maximize my profit given the resources I have been allocated?

Conclusions

- Rack-scale systems should allocate resources according to supply and demand by having workloads bid for them
- This is the only approach we are aware of that (1) cannot be gamed; and (2) maximizes users' satisfaction
- We are experimenting with market-driven resource allocation in the Stratoscale system