Predictable Data Centers

Hitesh Ballani, Paolo Costa, Fahad Dogar, Keon Jang,
Thomas Karagiannis, and Ant Rowstron

Systems & Networking
Microsoft Research, Cambridge

http://research.microsoft.com/datacenters/

Predictable Data Centers

Goal: Enable predictable application performance in multi-tenant data centers

Multi-tenant data center is a data center with multiple (possibly competing) tenants

Private data centers

- Run by organizations like Facebook, Microsoft, Google, etc
- Tenants: Product groups and applications

Cloud data centers

- Amazon EC2, Microsoft Azure, Rackspace, etc.
- Tenants: Users renting virtual machines

Unpredictability

Often cited as a key hindrance to cloud adoption

Root cause: Shared resources

In multi-tenant data centers, resources like the network and storage are shared amongst users

Variable resource performance → Unpredictable performance for applications and services

Dimensions of unpredictability

Performance

- No throughput or latency guarantees
- Private data centers: SLA violations, starvation
- Public data centers: Impossible to provide SLAs

Costs

- Absence of performance guarantees implies unpredictable costs
- Location-dependent

Fairness

- Same payment may not always translate to same performance

Outline

Public cloud

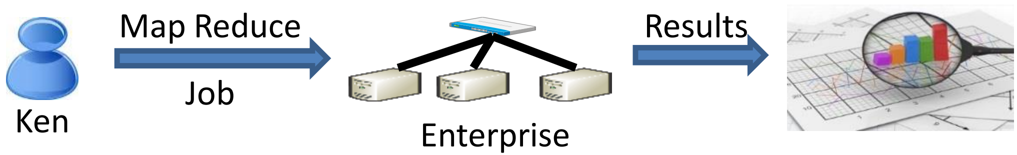
- Dealing with performance & cost unpredictability

Private data centers

- Meeting SLAs
- Reducing completion time

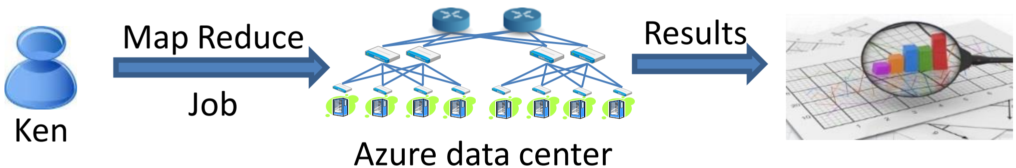
Performance Unpredictability

Data analytics on an isolated cluster

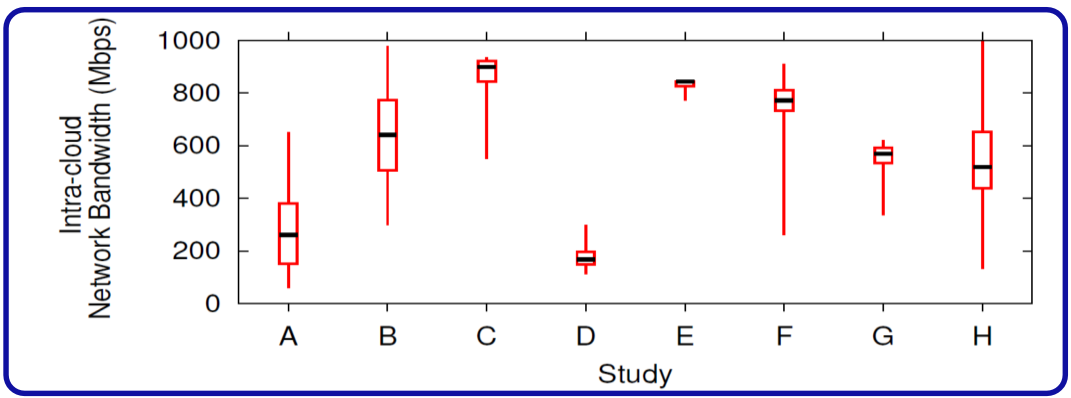


Completion Time 4 hours

Data analytics in the public cloud

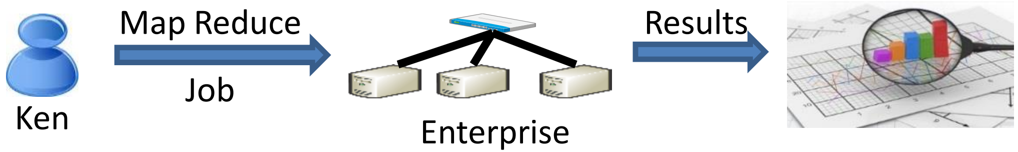


Completion Time 4-8 hours



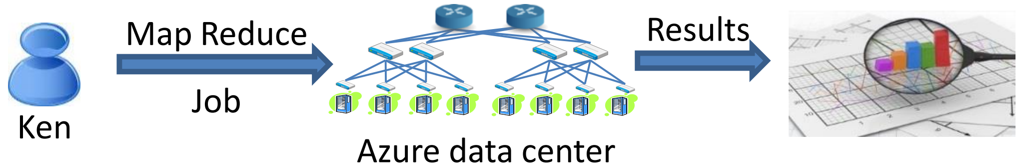
Performance Unpredictability

Data analytics on an isolated cluster



Completion Time 4 hours

Data analytics in the public cloud



Completion Time 4-8 hours

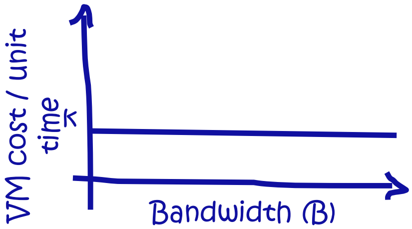
Variable tenant costs

Expected cost (based on 4 hour completion time) = $100

Actual cost = $100-200

Cost unpredictability

Today's Price



CPU-bound jobs

Job Cost = \$ $k \cdot N \cdot T$
(e.g.., k= \$0.085 /hour)

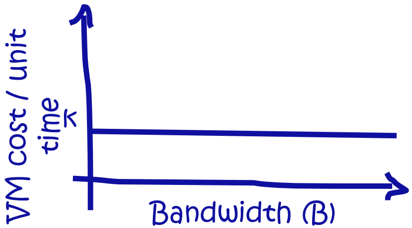
Network-bound jobs

Job Cost = \$ $k \cdot N \cdot T$
but … $T = \frac{L}{B}$, hence..
Job Cost = \$ $k \cdot N \cdot \frac{L}{B}$

✓ Simple and intuitive

Cost unpredictability

Today's Price



CPU-bound jobs

Network-bound jobs

Location-dependent pricing!

Job Cost
(e.g.., k= $0.085 /hour)

$N \cdot T$

but … $T = \frac{\;}{B}$, hence..

Job Cost = $ $k \cdot N \cdot \frac{L}{B}$

✓ Simple and intuitive

Towards a predictable cloud

Performance

- Guarantee network throughput
- Virtual Network Abstractions
  [Oktopus, SIGCOMM 11]

Extend the tenant-provider interface to account for the network

Towards a predictable cloud

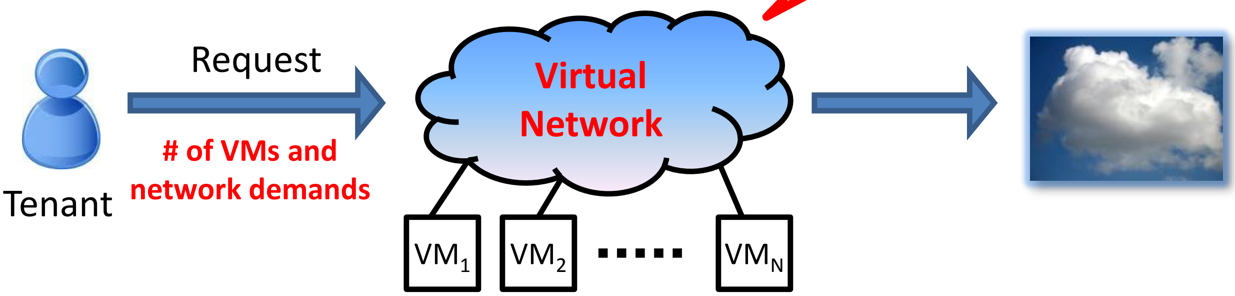
Performance

- Guarantee network throughput
- Virtual Network Abstractions
  [Oktopus, SIGCOMM 11]

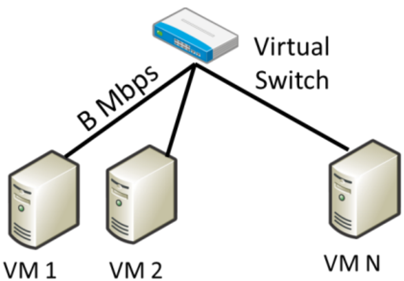
Key Idea: Tenants are offered a virtual network with bandwidth guarantees

This decouples tenant performance from provider infrastructure

Extend the tenant-provider interface to account for the network



Oktopus



Request <N, B>

N VMs. Each VM can send and receive at B Mbps

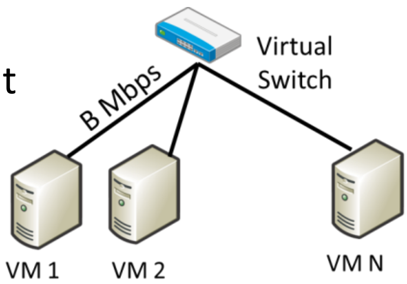
Two main components

- Management plane: ***Allocation of tenant requests***
  - Allocates tenant requests to physical infrastructure
  - Accounts for tenant network bandwidth requirements
- Data plane: ***Enforcement of virtual networks***
  - Enforces tenant bandwidth requirements
  - Achieved through rate limiting at end hosts

Towards a predictable cloud

Performance

- Guarantee network throughput
- Virtual Network Abstractions
  [Oktopus, SIGCOMM 11]

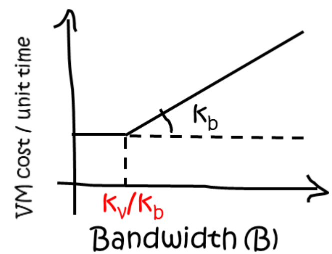


Request <N, B>

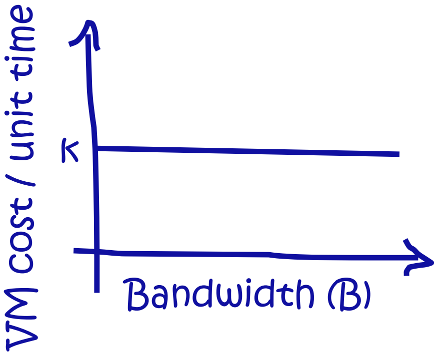
N VMs. Each VM can send and receive at B Mbps

Pricing

- Dominant resource pricing
  [HotNets 11]



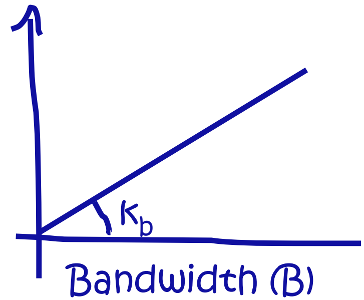
How to combine the two pricing models?



Occupancy

✓ CPU-bound jobs

✗ Network-bound jobs



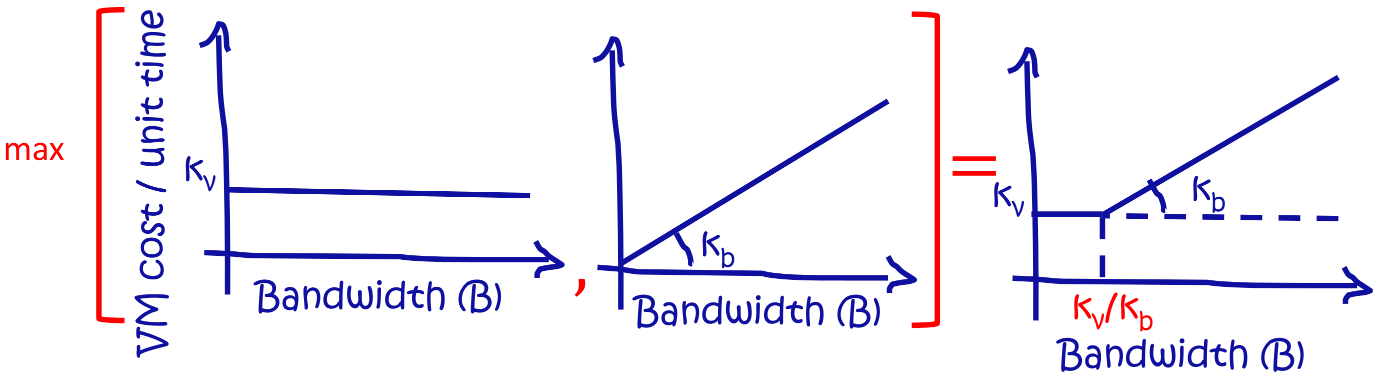
Usage

✗ CPU-bound jobs

✓ Network-bound jobs

How to combine the two pricing models?

Dominant Resource Pricing (DRP)



VM Cost / unit time

= k_v if $B < \frac{k_v}{k_b}$

= $k_b \cdot B$ if $B \geq \frac{k_v}{k_b}$

Job Cost

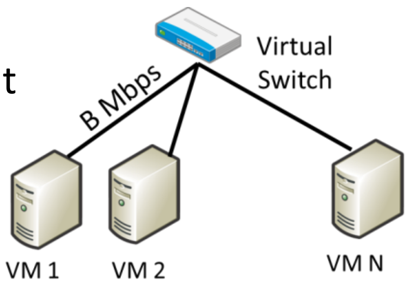
= $\$\, k_v \cdot N \cdot T$ if $B < \frac{k_v}{k_b}$ (occupancy)

= $\$\, k_b \cdot N \cdot L$ if $B \geq \frac{k_v}{k_b}$ (usage)

Towards a predictable cloud

Performance

- Guarantee network throughput
- Virtual Network Abstractions
  [Oktopus, SIGCOMM 11]

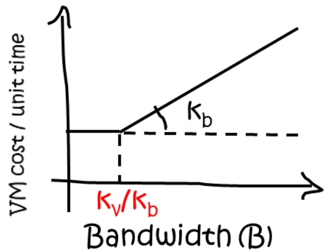


Request <N, B>

N VMs. Each VM can send and receive at B Mbps

Pricing

- Dominant resource pricing
  [HotNets 11]

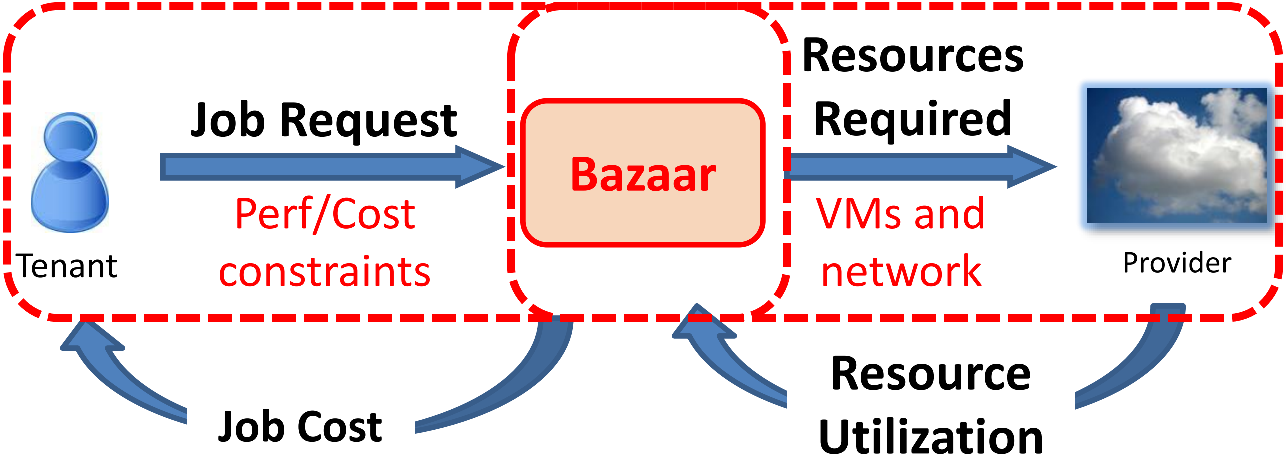


Change the cloud model!

- Job-based pricing
  [Bazaar, SOCC 12]

Bazaar

Enables predictable performance and cost



Today's pricing: Resource-based

Bazaar enables job-based pricing!

Outline

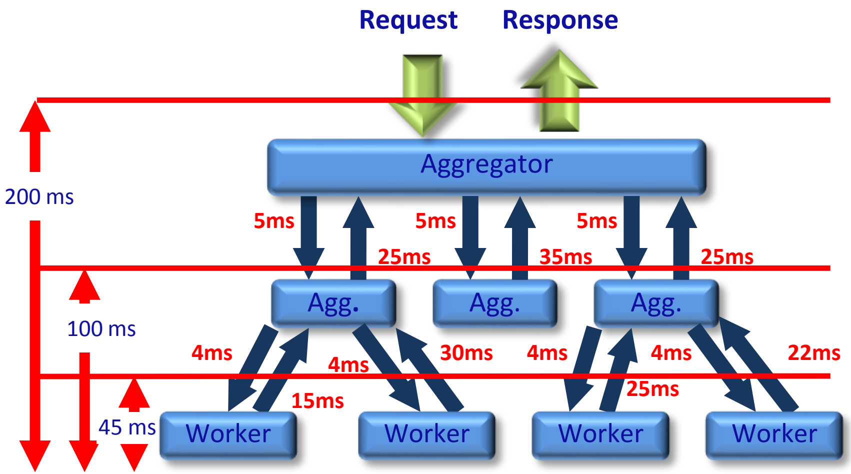
Public cloud

- Dealing with performance & cost unpredictability

Private data centers

- Meeting SLAs
- Reducing completion time

SLA violations: User-facing online services



Application SLAs

Component SLAs

SLAs for components at each level of the hierarchy

Network SLAs

Deadlines on communications between components

Flow Deadlines

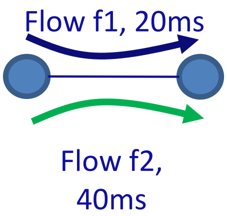
A flow is useful **if and only if** it satisfies its deadline

Today's transport protocols:

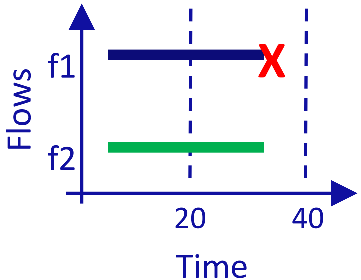
Deadline agnostic and strive for fairness

Limitations of Fair Sharing

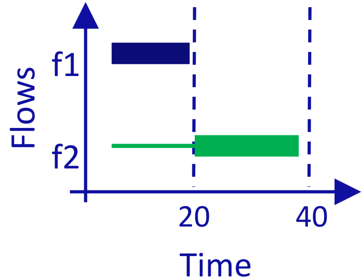
Case for unfair sharing:



Status Quo



Deadline aware

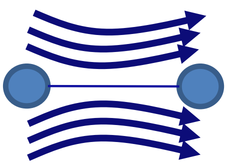


Flows f1 and f2 get a fair share of bandwidth

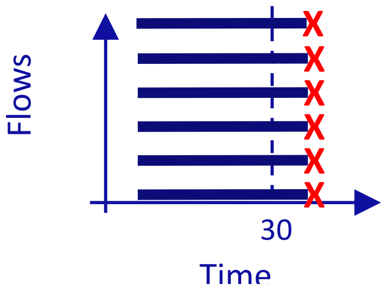
Flow f1 misses its deadline (incomplete response to user)

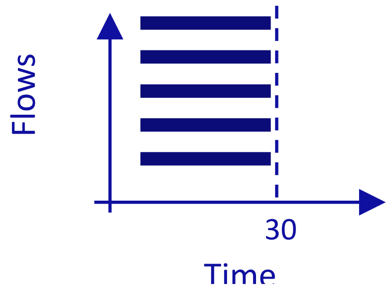
Limitations of Fair Sharing

Case for flow quenching:



6 flows with
30ms deadline





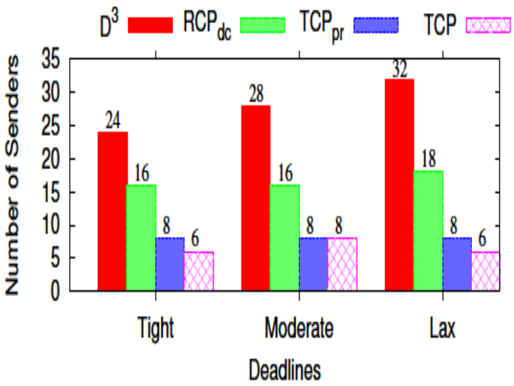
With deadline awareness, one flow can be quenched
All other flows make their deadline (partial response)

Predictability in private data centers

Deadline-driven flow scheduling

D^3 [SIGCOMM 11]

- Prioritize flows based on deadlines
- Expose flow deadlines to the network
- Explicit rate control



Task aware data centers

- Reducing task completion times
- Amazon: extra 100ms costs 1% in sales

Task Oriented Applications

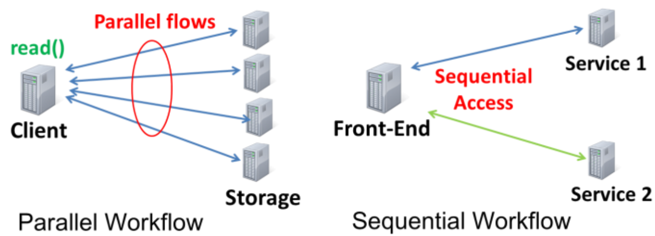
Typical DC apps perform “tasks”

- Unit of work that can be linked to a waiting user

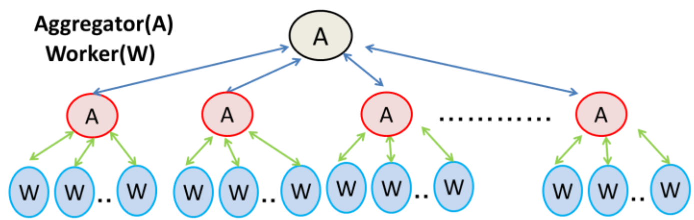
Examples

Answering a user’s search query

Generating a user’s wall



From the network’s perspective, tasks generate rich workflows



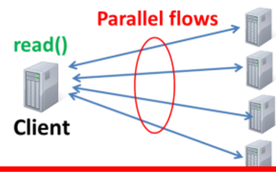
Partition-Aggregate Workflow

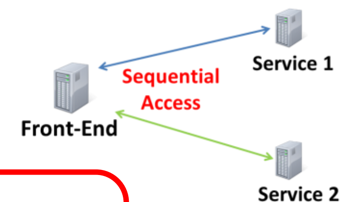
Task Oriented Applications

Typical DC apps perform "tasks"

- Unit of work that can be linked to a waiting user

Flows of tasks traverse different parts of the network at different times

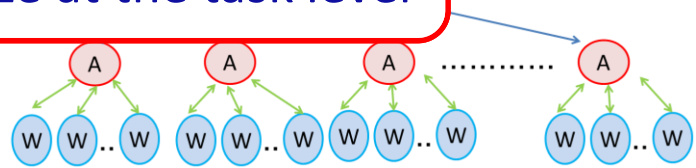




ial Workflow

To reduce task completion times it is important to optimize at the task level

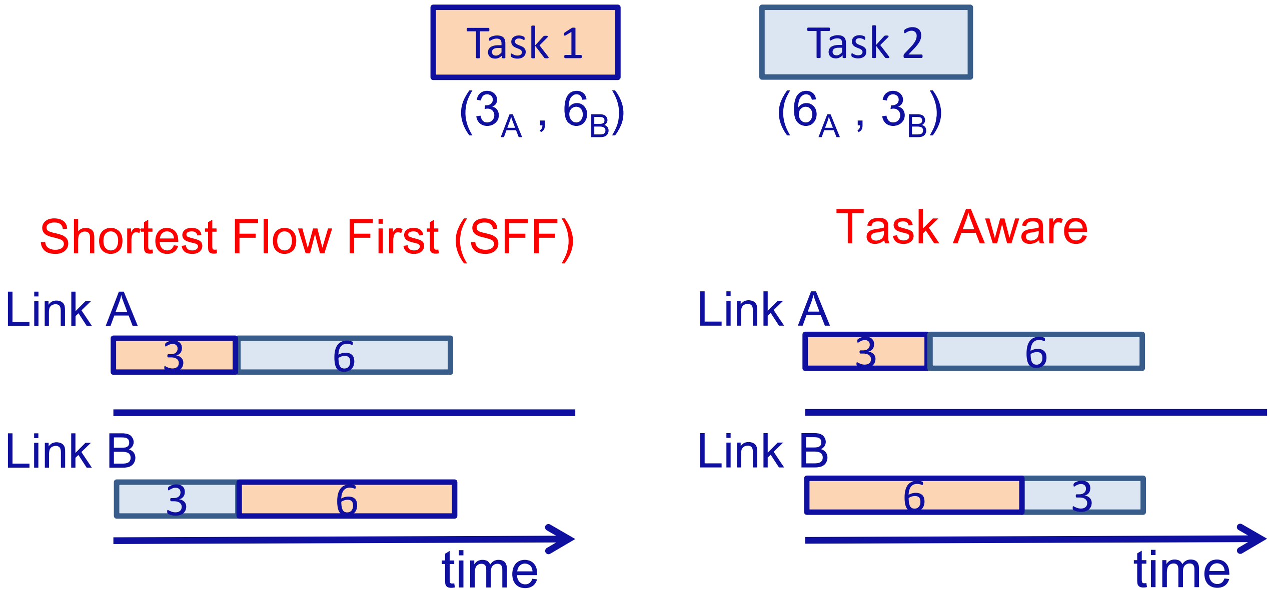
From the r
tasks gen



Partition-Aggregate Workflow

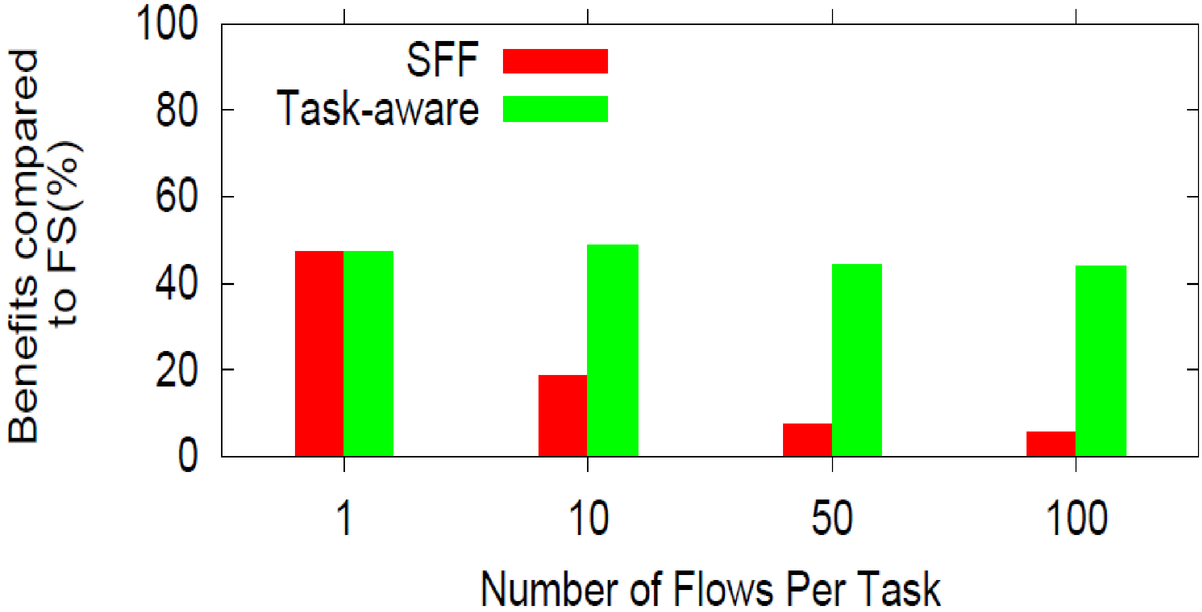
Flow vs. Task aware optimizations

Goal: Minimize Task Completion Times



Flow vs. Task aware optimizations

Goal: Minimize Task Completion Times



Task aware data centers

Designing a practical task-aware scheduling system

- Policy – order in which tasks should be processed
- Decentralized mechanisms to prioritize tasks

Benefits

- 65% reduction in task completion time

Summary

Unpredictability

- Key hindrance to cloud adoption
- Root cause: Shared resources
- Several challenges: performance, cost, fairness

http://research.microsoft.com/datacenters/