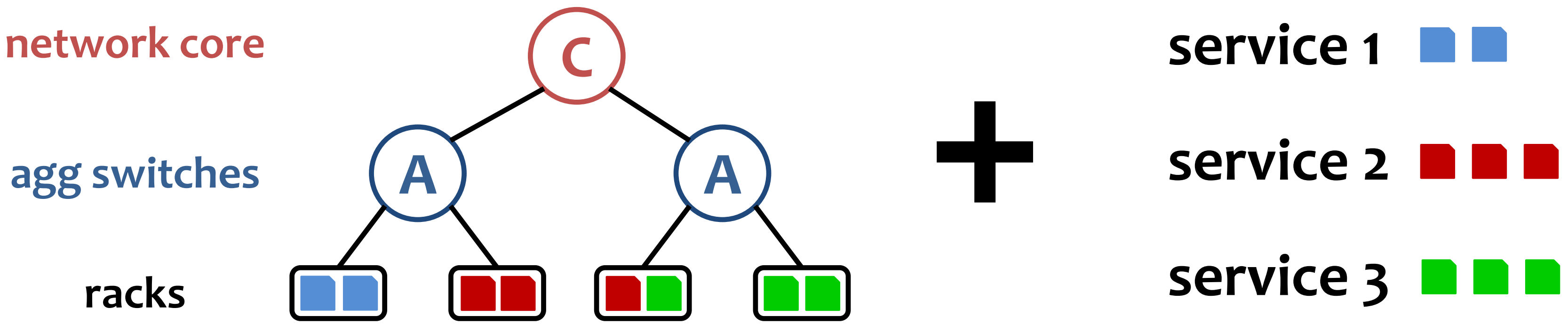
Surviving Failures in Bandwidth-Constrained Datacenters

Peter Bodík[2], Ishai Menache[2], Mosharaf Chowdhury[3], Pradeepkumar Mani[1], Dave Maltz[1], Ion Stoica[3]

Microsoft[1] Research[2], UC Berkeley[3]

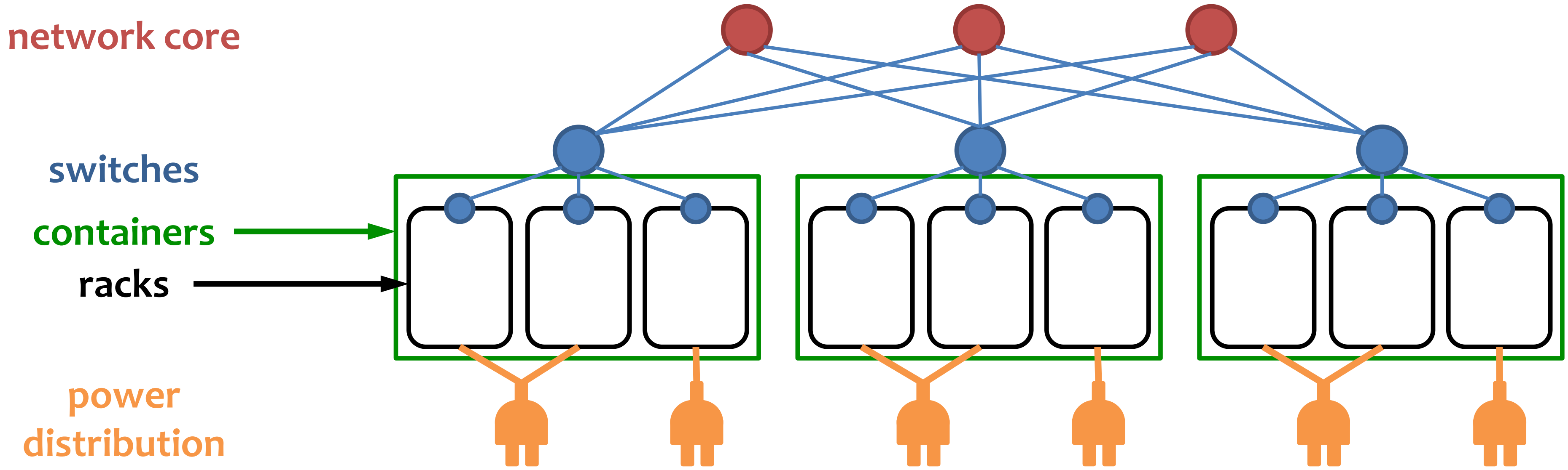
How to allocate services to physical machines?



Three important metrics considered together

- FT: service fault tolerance
- BW: bandwidth usage
- #M: # machine moves to reach target allocation

FT: Improving fault tolerance of software services

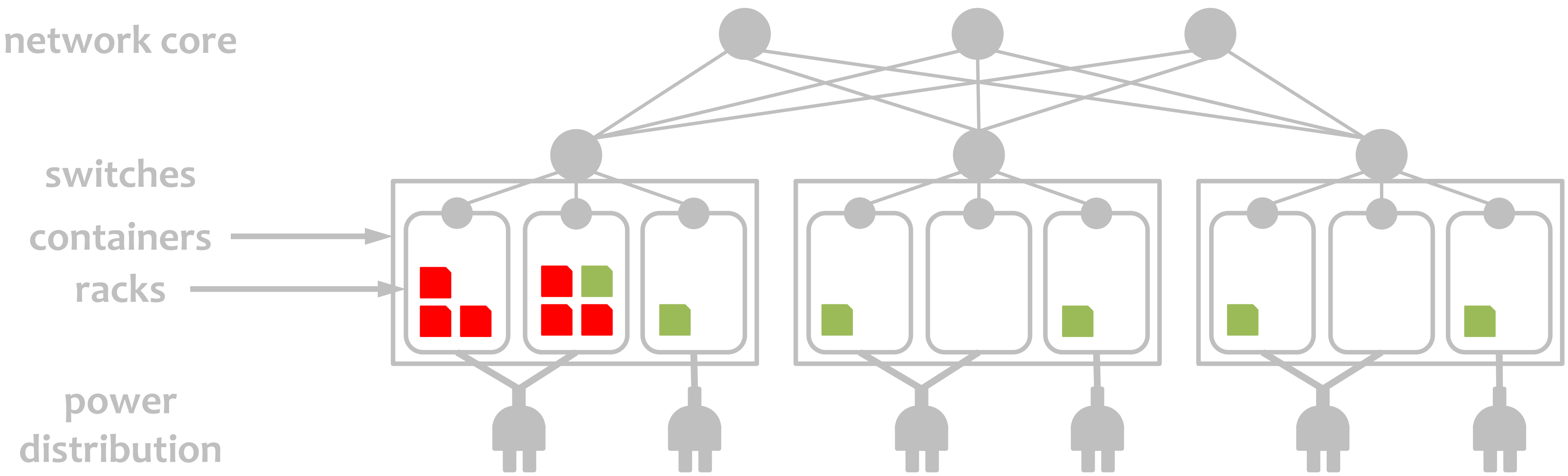


Complex fault domains: networking, power, cooling

Worst-case survival = fraction of service available during single worst-case failure

– corresponds to service throughput during failure

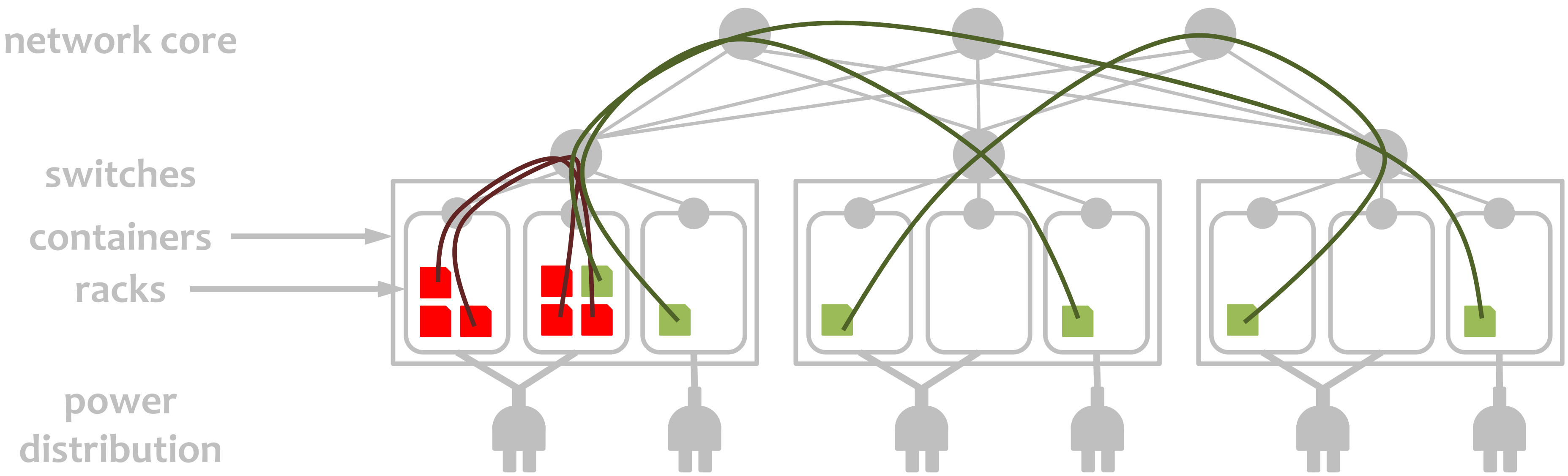
FT: Service allocation impacts worst-case survival



Worst-case survival:

- red service: 0% -- same container, power
- green service: 67% -- different containers, power

BW: Reduce bandwidth usage on constrained links



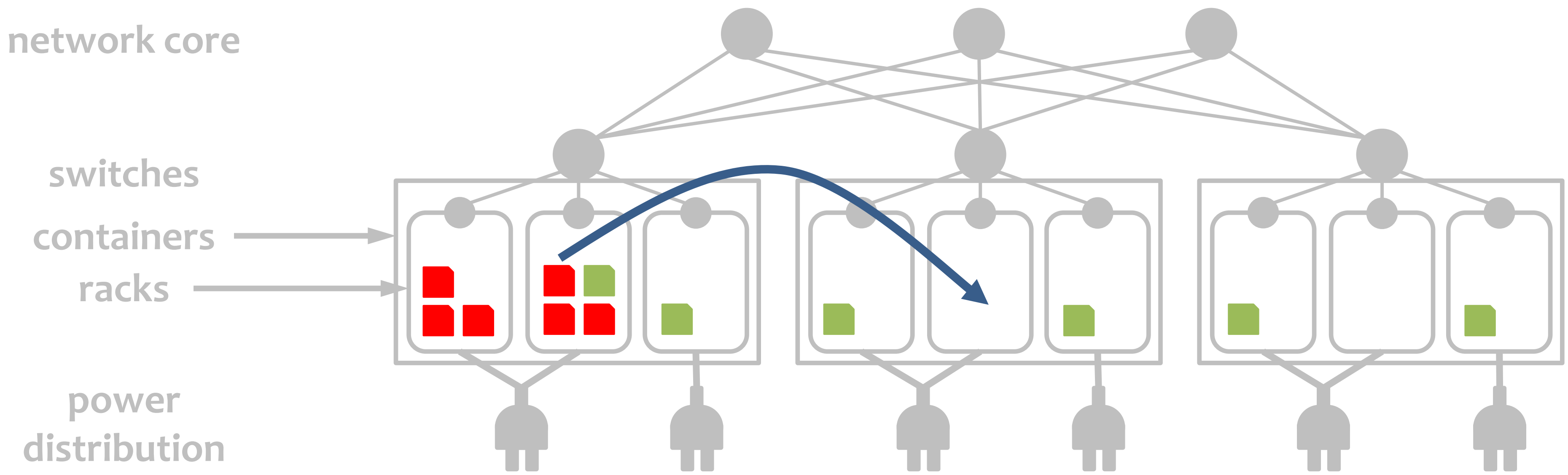
BW = bandwidth usage in the core

Goal

– reduce cost of infrastructure

– consider other service location constraints

#M: Need incremental allocation algorithms



High cost of machine move

- need to deploy potentially TB of data
- warm up caches
- could take tens of minutes, impact network

Outline

Why is it difficult?

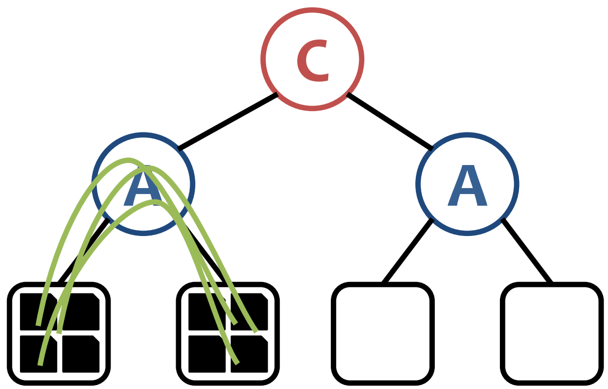
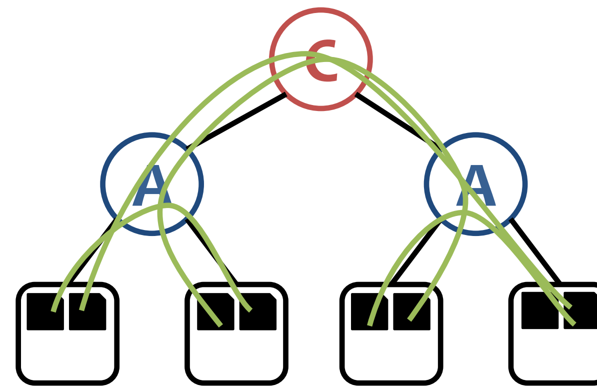
Traffic analysis

Optimization framework

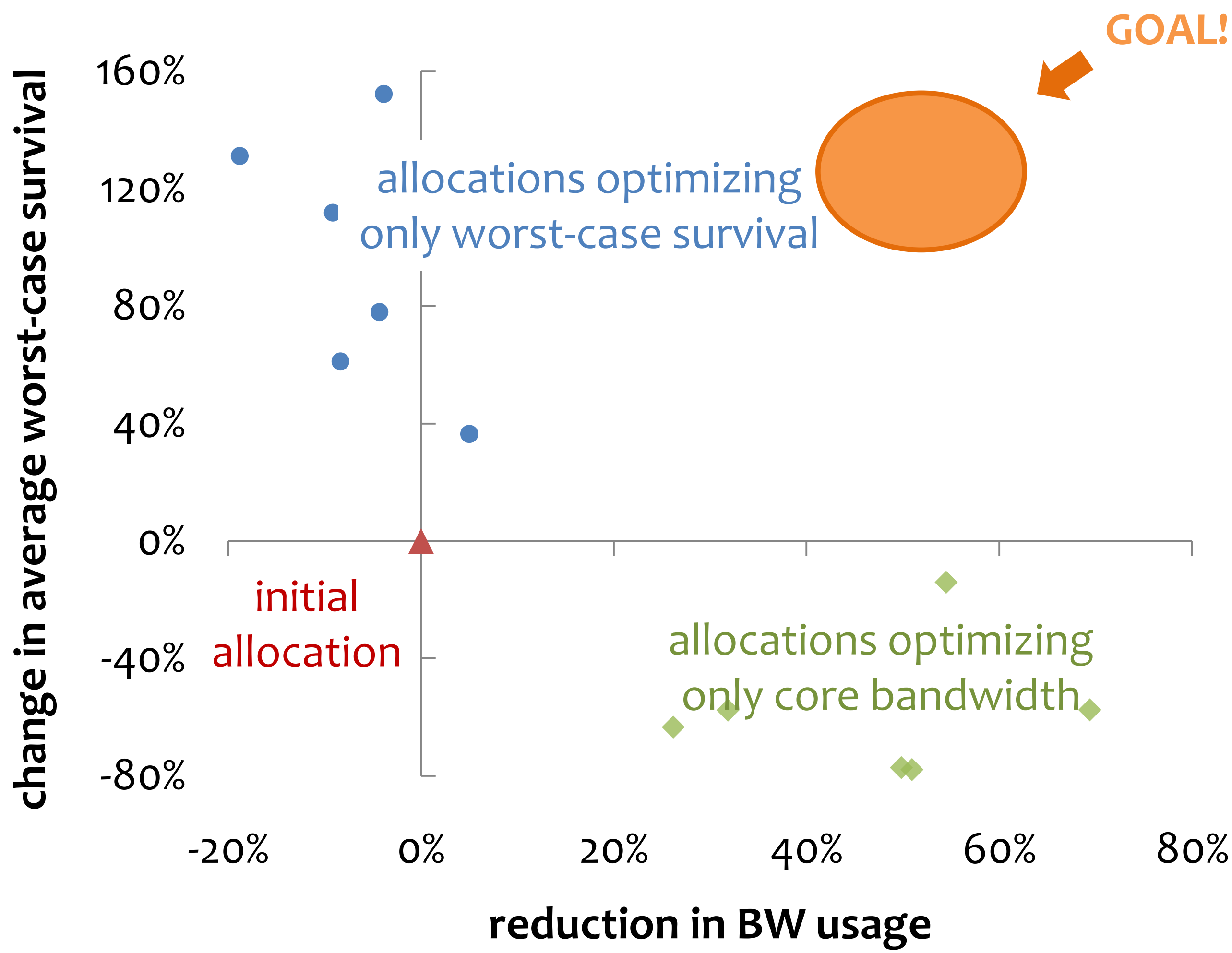
- FT + #M
- FT + BW + #M

Evaluation

Trade-off between *bandwidth usage* and *fault-tolerance*

	optimize for bandwidth	optimize for fault tolerance
network core agg switches racks		
BW: utilization in core	LOW ✓	HIGH ✗
FT: fault tolerance (for agg switches)	LOW ✗	HIGH ✓
worst-case survival	0 ✗	0.5 ✓

Optimizing for one metric degrades the other



Results from 6 Microsoft datacenters

FT-only and BW-only are both NP-hard, hard to approximate

FT reduces to max independent set

BW reduces to min-cut in a graph

– considered previously in [Meng et al., INFOCOM'10]

Most algorithms not incremental, ignore #M

Key insights

Improve FT using convex optimization

– local optimization leads to good solutions

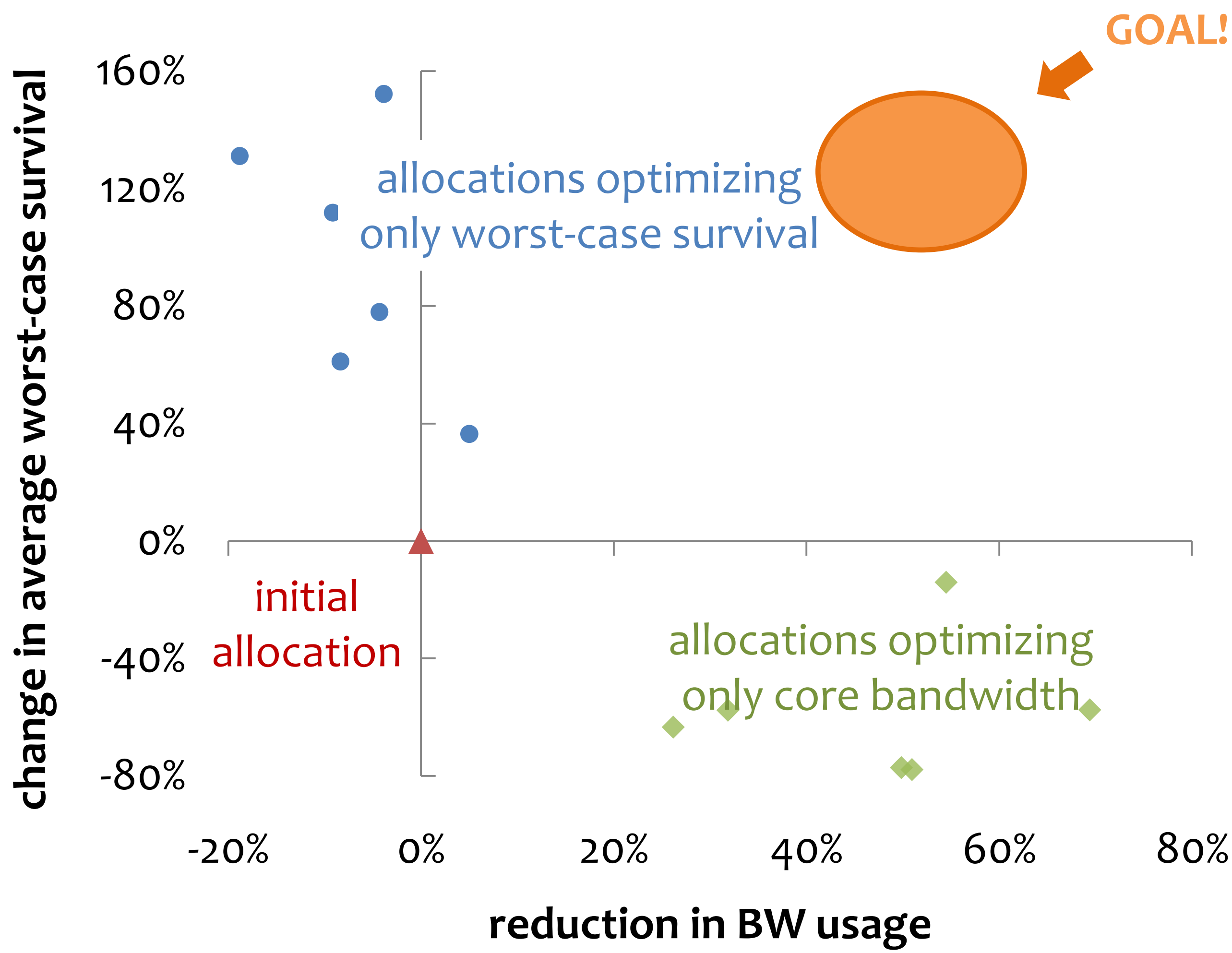
Symmetry in the optimization space

– machines, racks, containers are interchangeable

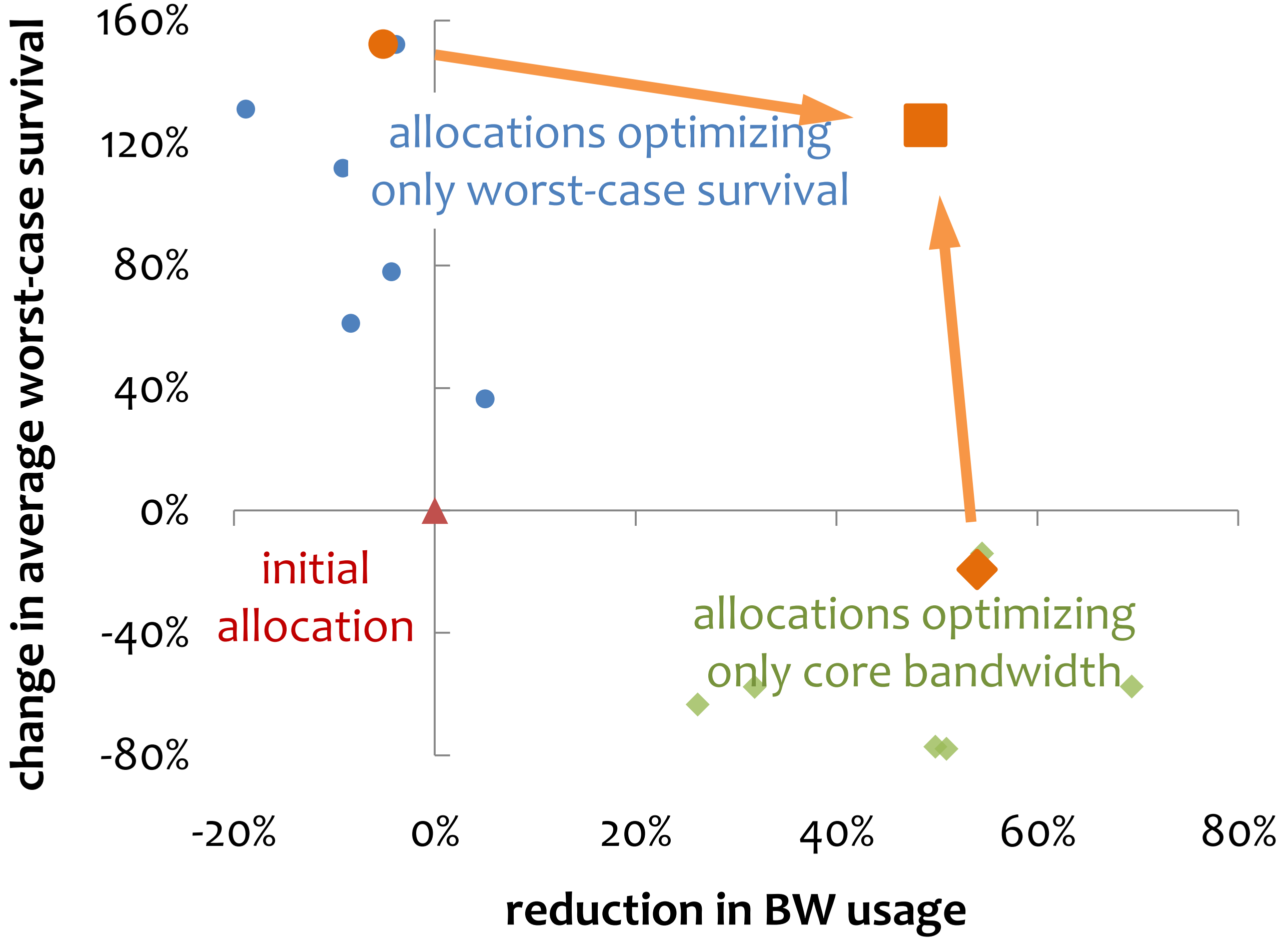
Communication pattern is very skewed

– can spread low-talkers without affecting BW

Results preview



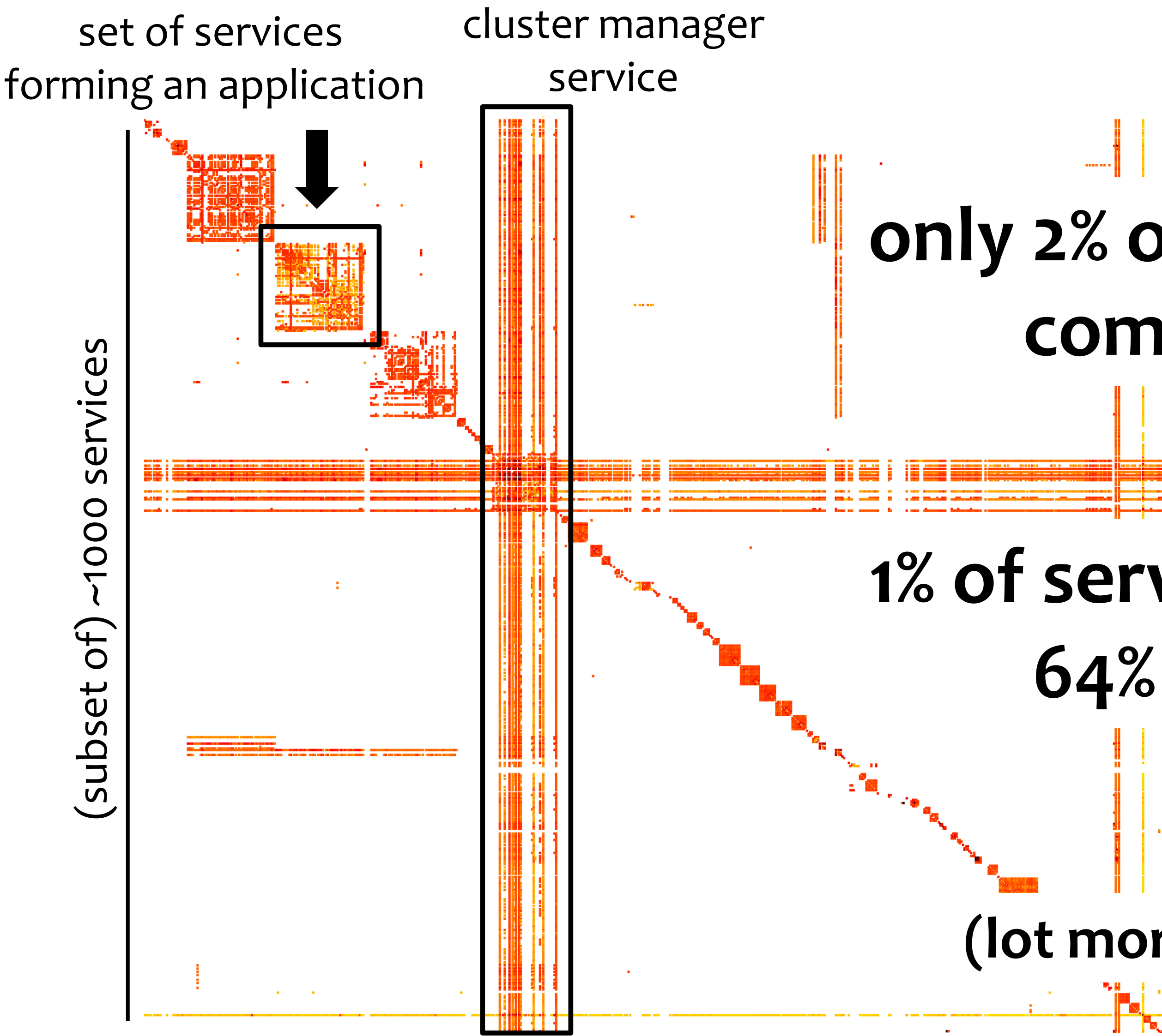
Results preview



Outline

- Why is it difficult?
- Traffic analysis
- Optimization framework
  - FT + #M
  - FT + BW + #M
- Evaluation

Service communication matrix is very sparse and skewed



only 2% of service pairs communicate

1% of services generate 64% of traffic

(lot more in the paper)

Outline

- Why is it difficult?
- Traffic analysis
- Optimization framework
  - FT + #M
  - FT + BW + #M
- Evaluation

FT optimizing FT and #M

Spread machines across all fault domains

– FTC negatively correlated to worst-case survival

Convex optimization

$$\text{FTC} = \sum_{s} c_s \sum_{f} w_f \cdot z_{s,f}^2$$

c_s: service weight

w_f: fault domain weight

$z_{s,f}$: number of machines of service *s* in domain *f*

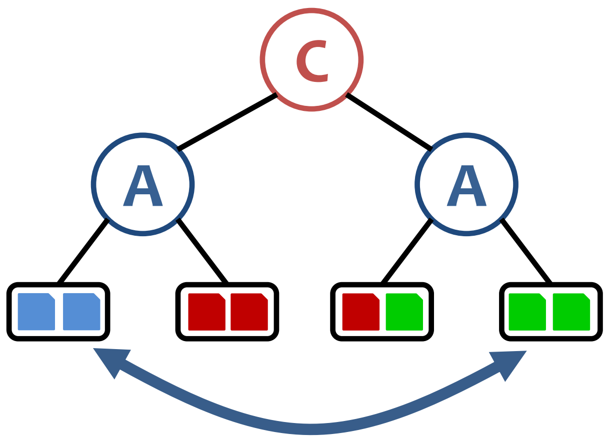
Advantages of convex cost function

– local actions lead to improvement of global metric

– directly considers #M

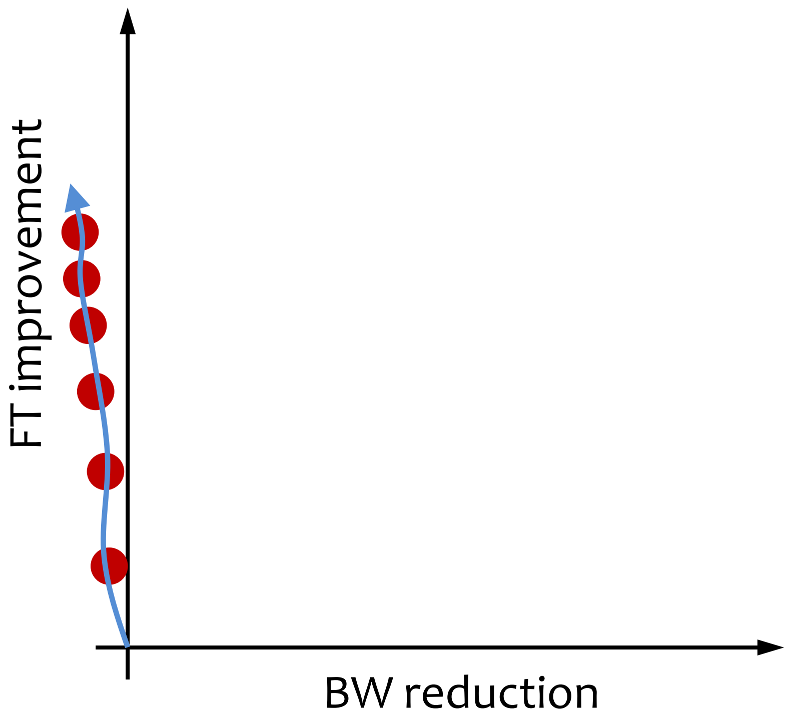
FT machine swap as a basic move

- Keeps the current allocation feasible
  - doesn't change number of machines per service



- Steepest descent swap = largest reduction in cost
- Only evaluate a small, random set of swaps
  - symmetry => many "good" swaps exist

FT path of steepest descent



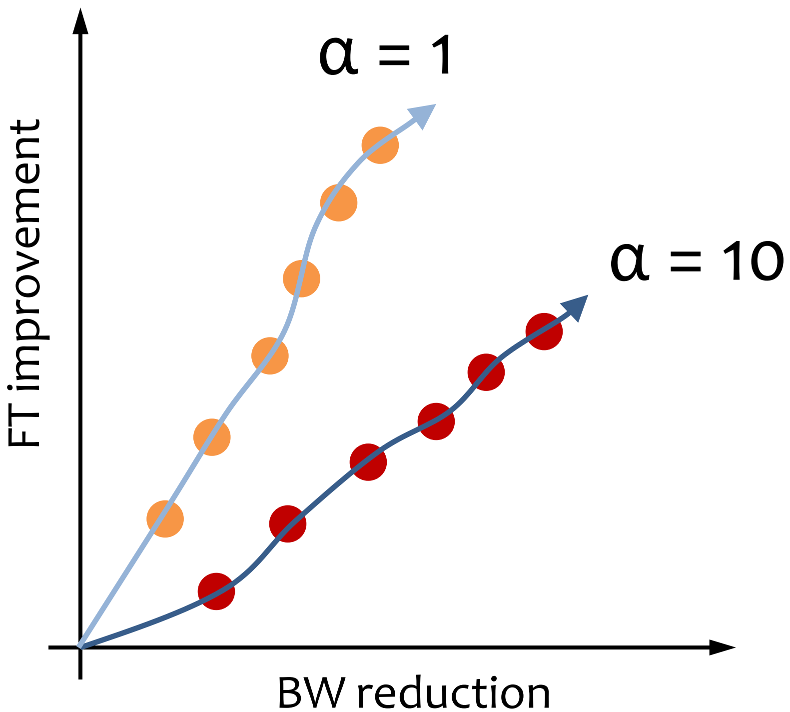
FT+BW Optimizing FT, BW, and #M

Steepest descent on FTC + α BW

- non-convex
- no guarantees on reaching optimum

α determines the FT-BW trade-off

FT+BW path of steepest descent



Benchmark algorithm

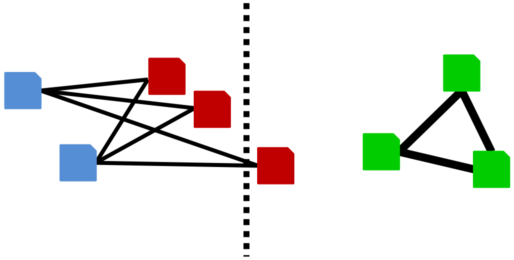
cut **FT+BW**

k-way minimum graph cut

– optimizes BW only

– ignores #M

machine communication graph



k-way min cut

followed by steepest descent on FT+BW

Outline

Why is it difficult?

Traffic analysis

Optimization framework

- FT + #M
- FT + BW + #M

Evaluation

Evaluation setup

Simulations based on 4 production clusters

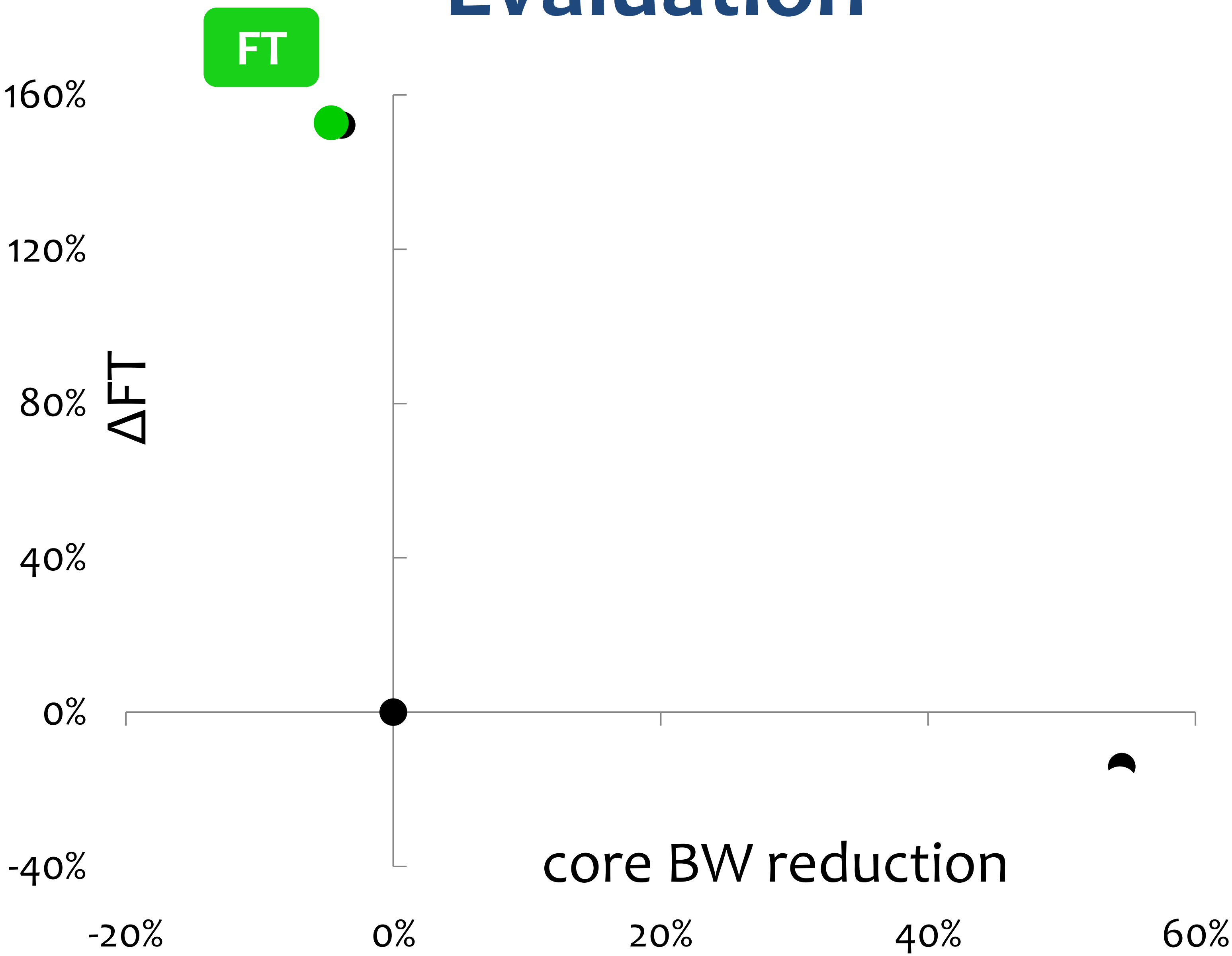
- services + machine counts
- network topology
- fault domains
- network trace from pre-production cluster

Metrics relative to initial allocation

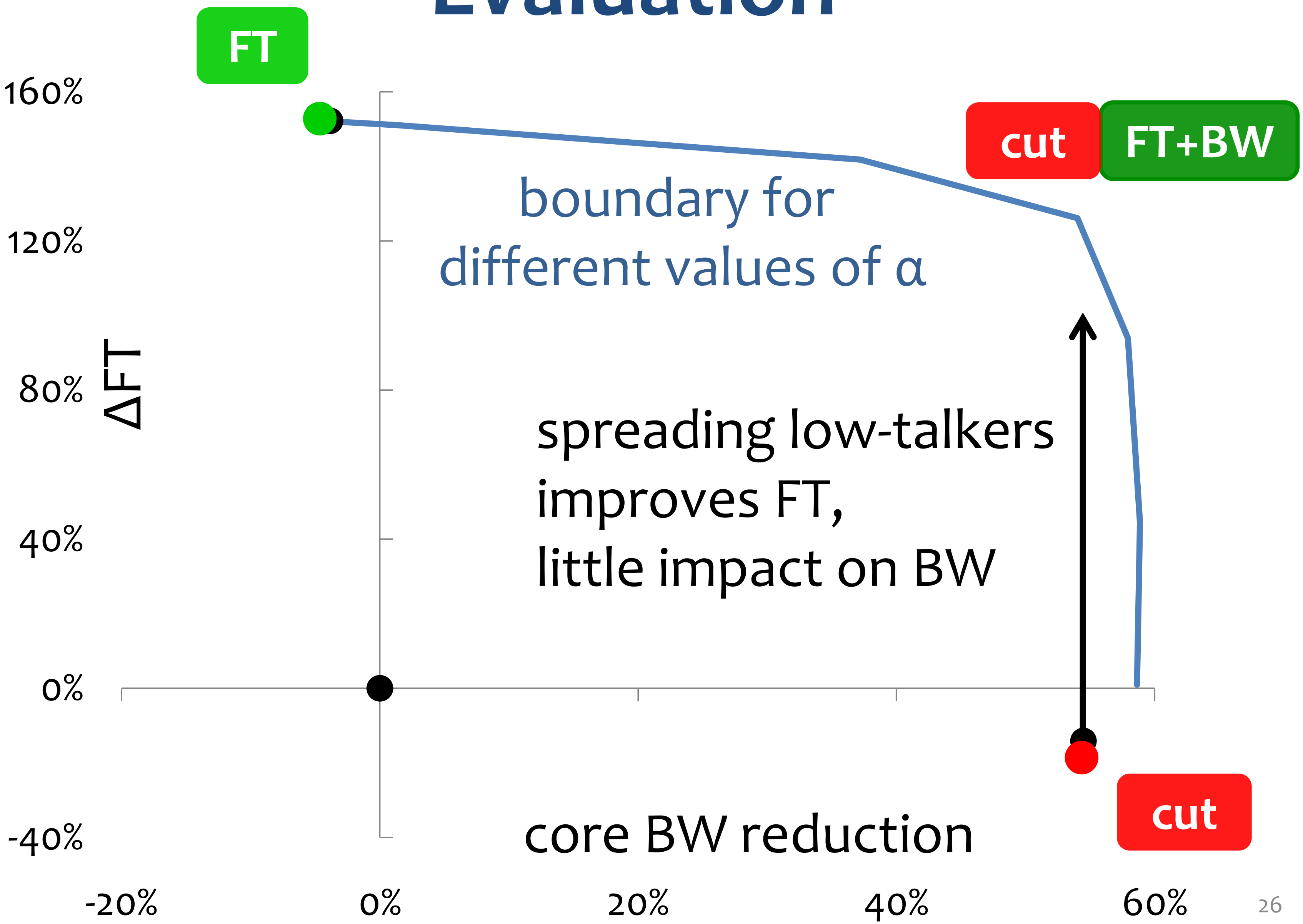
- don’t know actual optimum

Choosing next swap takes seconds to a minute

Evaluation

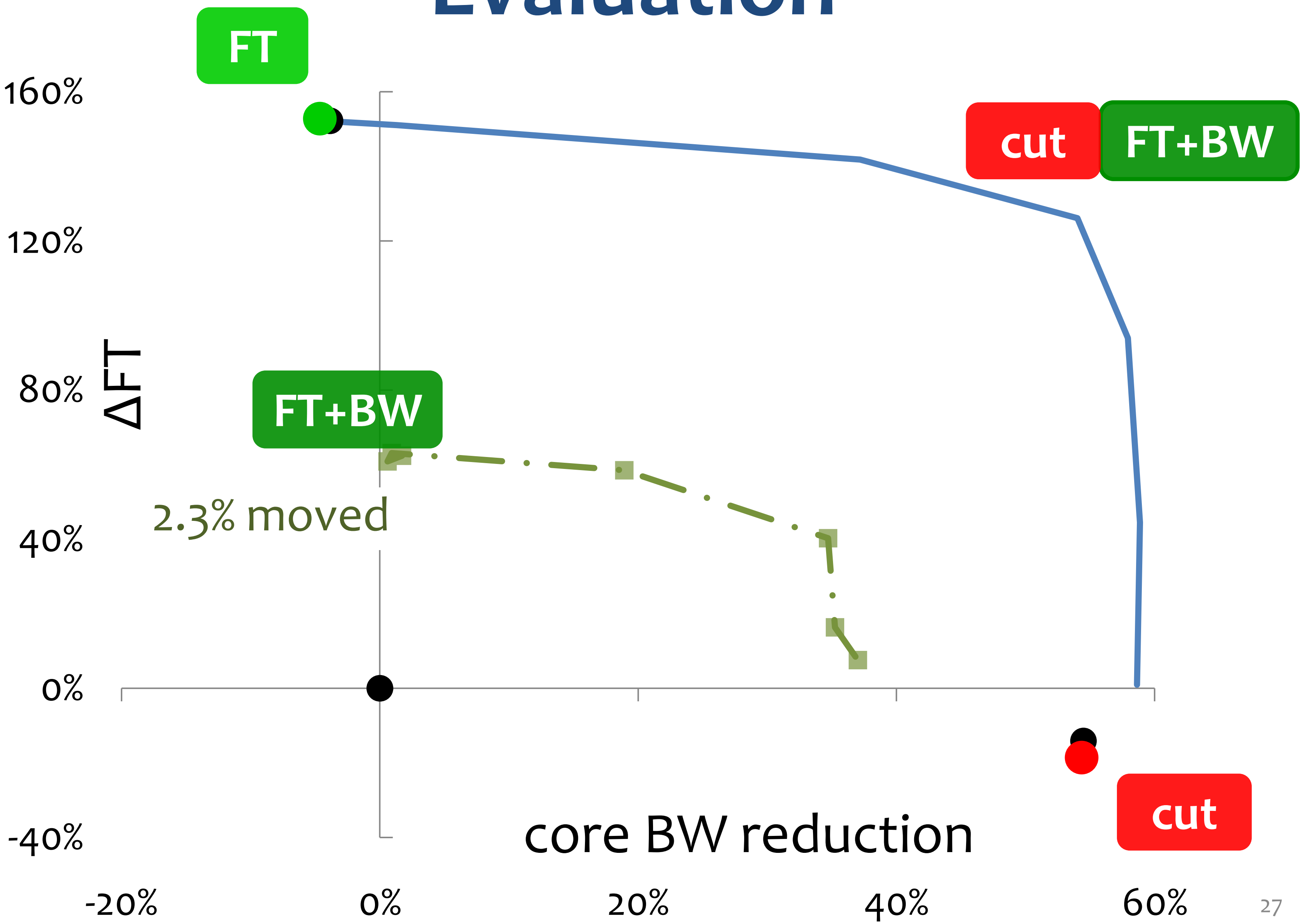


Evaluation

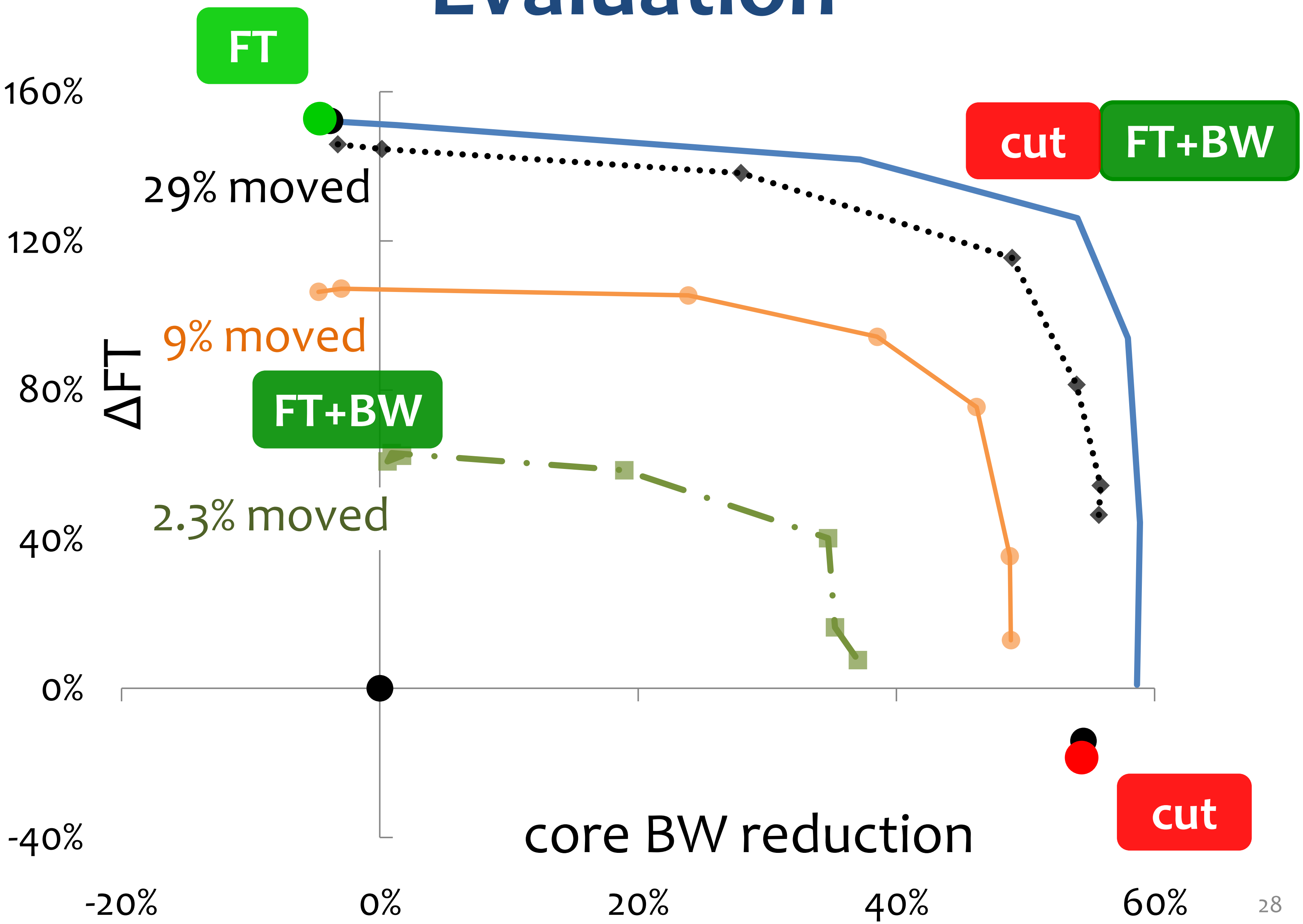


FT
160%
120%
80%
40%
0%
-40%
ΔFT
cut
FT+BW
boundary for
different values of α
spreading low-talkers
improves FT,
little impact on BW
cut
core BW reduction
-20%
0%
20%
40%
60%

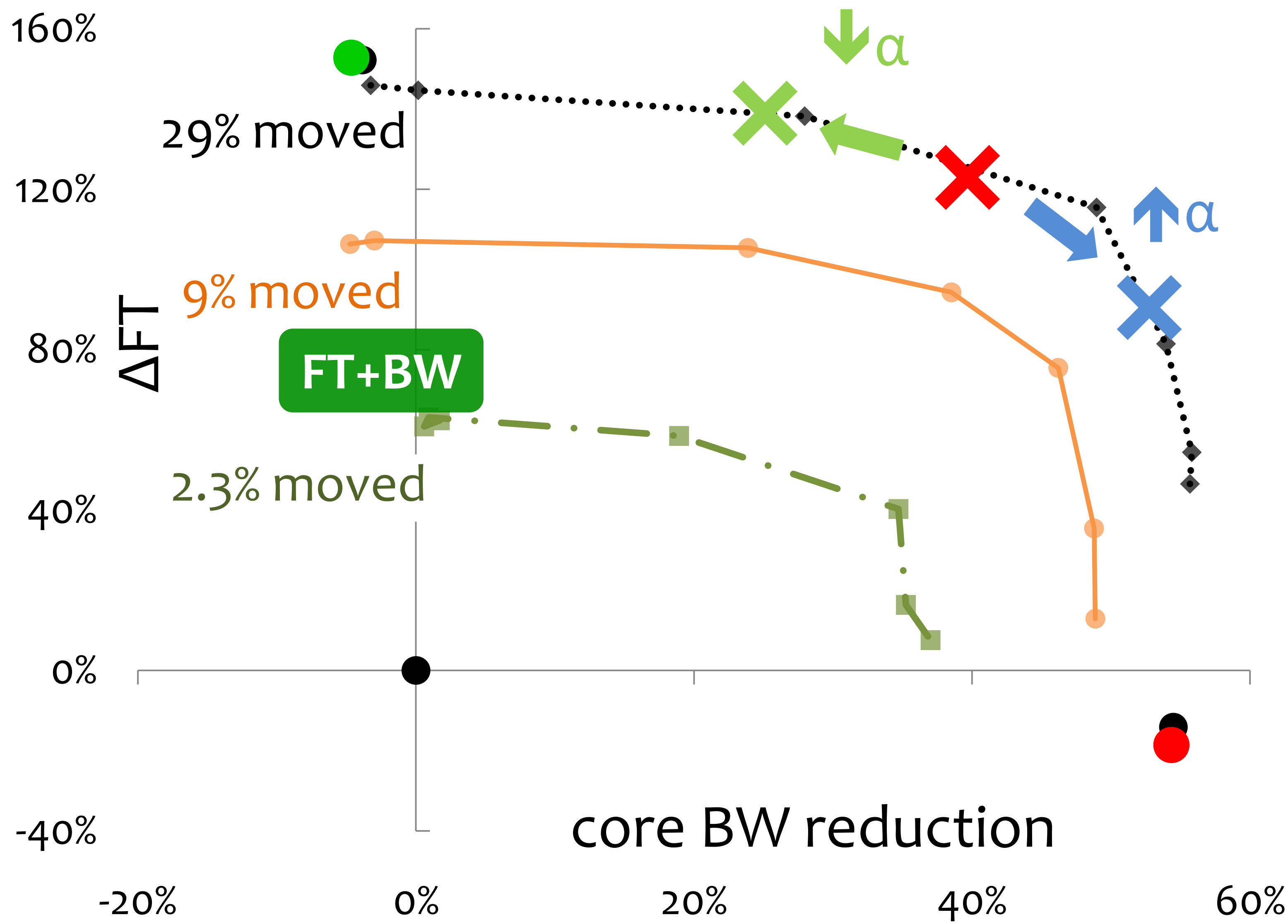
Evaluation


FT
160%
120%
80%
40%
0%
-40%
ΔFT
cut
FT+BW
FT+BW
2.3% moved
cut
core BW reduction
-20%
0%
20%
40%
60%

Evaluation


FT
160%
120%
80%
40%
0%
-40%
ΔFT
29% moved
9% moved
FT+BW
2.3% moved
cut
FT+BW
cut
core BW reduction
-20%
0%
20%
40%
60%

α changes the FT-BW tradeoff



Summary

Trade-off between fault tolerance and bandwidth

- algorithm that achieves improvement in both

Improvements (across 4 production datacenters)

- FT: 40% – 120%
- BW: 20% – 50%
- partially deployed in Bing

Key insights

- approximate NP-hard problem using convex optimization
- lot of symmetry in search space
- sparse and skewed communication matrix

Extensions

Hard constraints on FT, BW, #M

– e.g., pick a few services with FT>80%

Hierarchical BW optimization on agg switches

Applies to fat-tree networks

Main observations

Most traffic generated by few services (pairs)

➡ spread low-talkers to improve fault-tolerance

Complex, overlapping fault domains

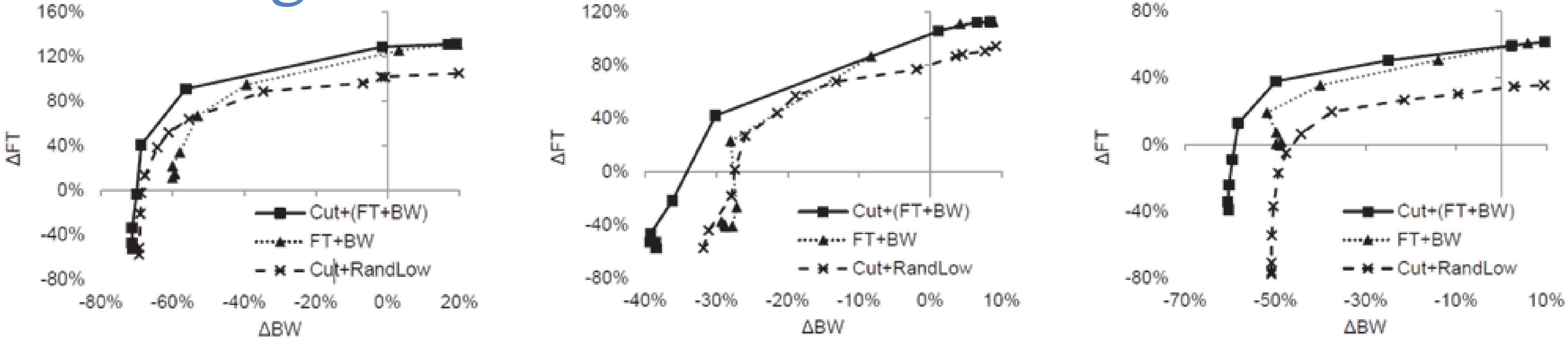
– hierarchical network fault-domains

– power fault domains not aligned with network

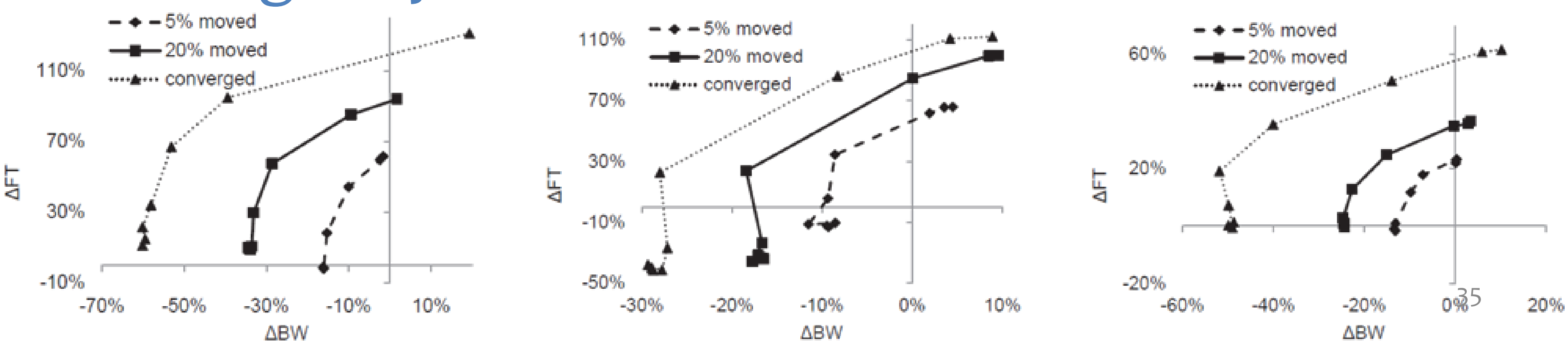
➡ **cell:** set of machines with identical fault domains

Evaluation

Moving most of machines



Moving only fraction of machines



Our optimization framework

Cost function considers FT and BW

- both problems NP-hard and hard to approximate
- non-convex

Cut + FT + BW:

1. minimum k-way cut of communication graph
   - reshuffles all machines
2. gradient descent moves using machine swaps

FT + BW:

1. only machine swaps
   - only moves small fraction of machines

Conclusion

Study of communication patterns of Bing.com

- sparse communication matrix
- very skewed communication pattern

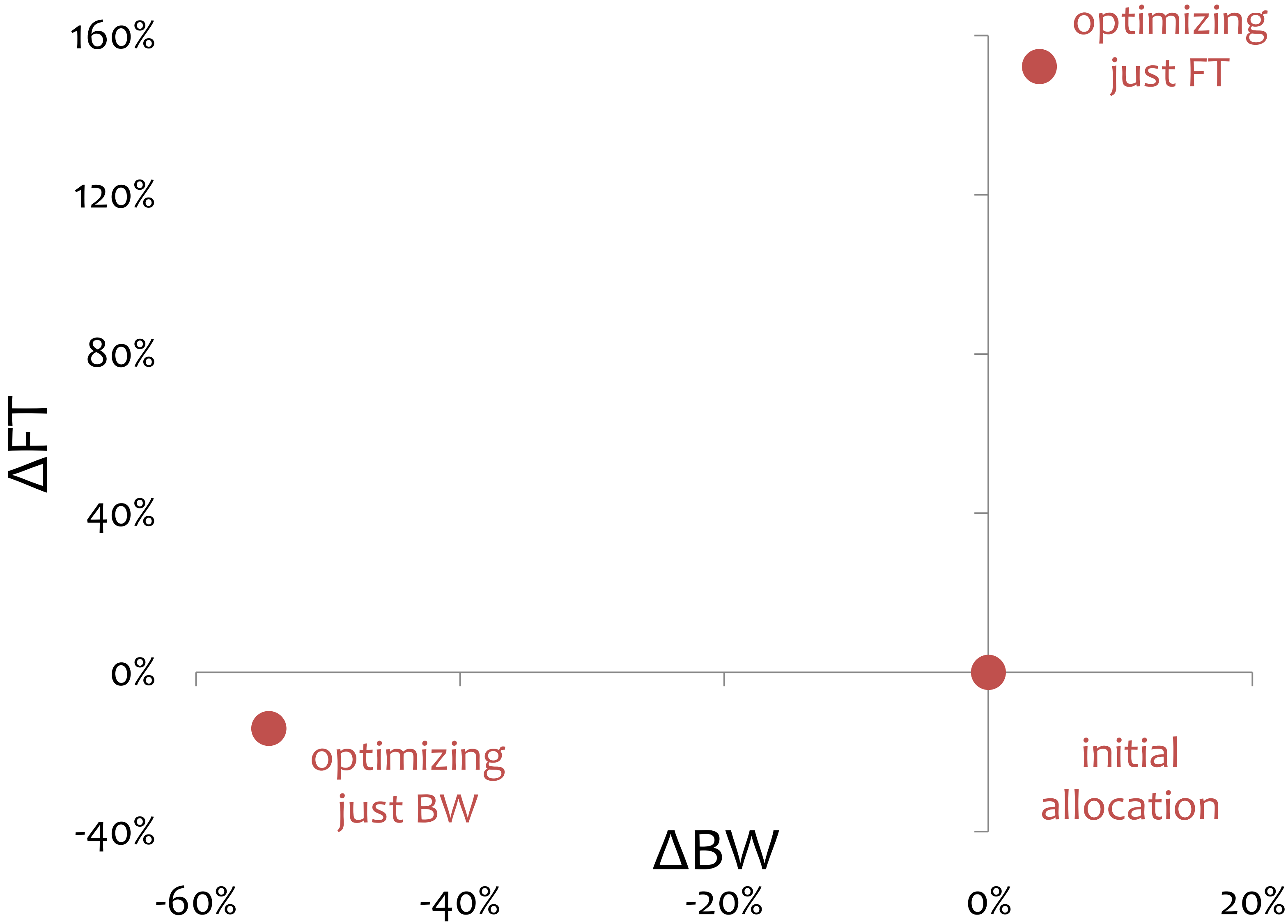
Principled optimization of both BW and FT

- exploits communication patterns
- can handle arbitrary fault domains

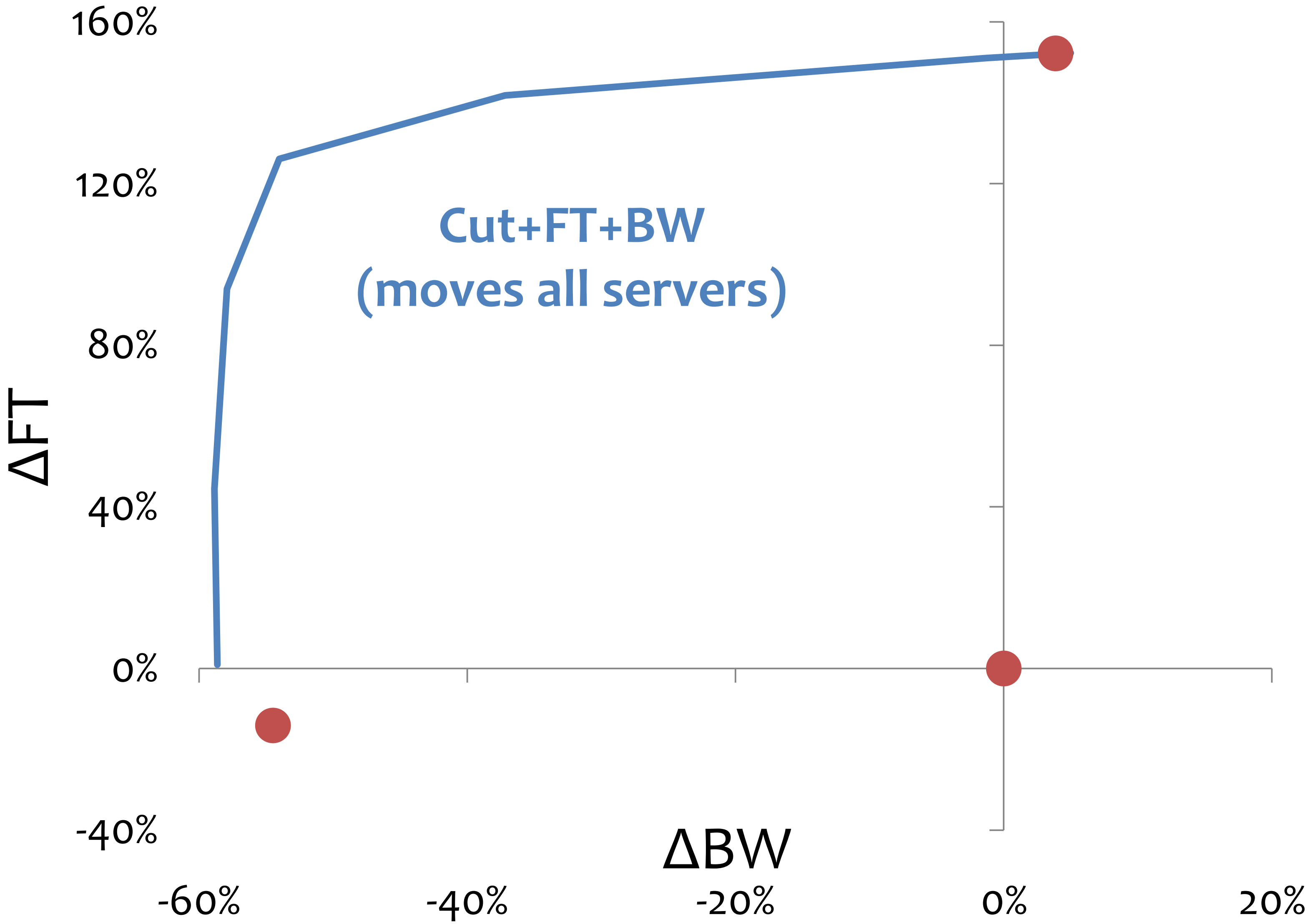
Reduction in BW: 20 – 50%

Improvement in FT: 40 – 120%

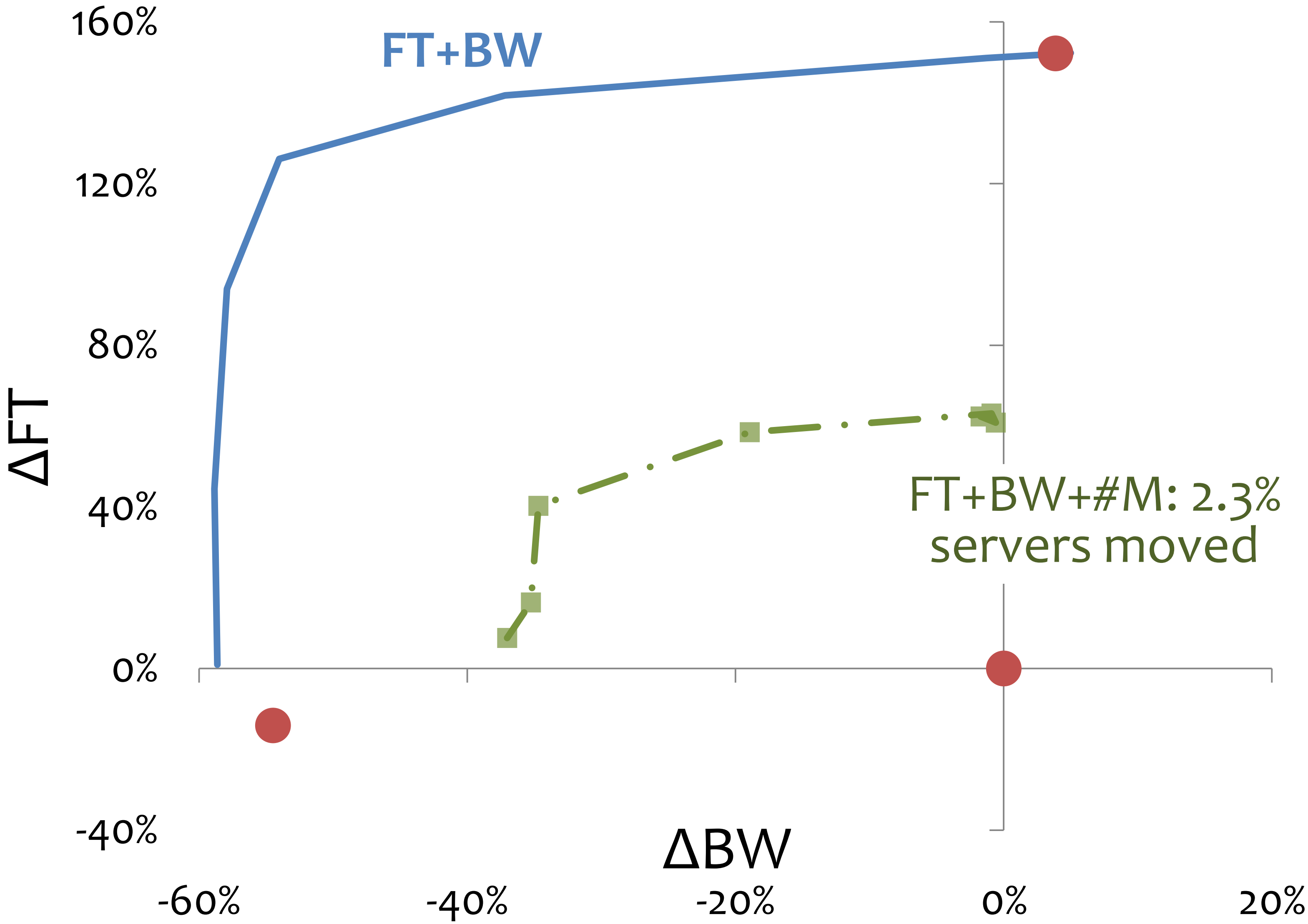
Evaluation (1 datacenter)



Evaluation

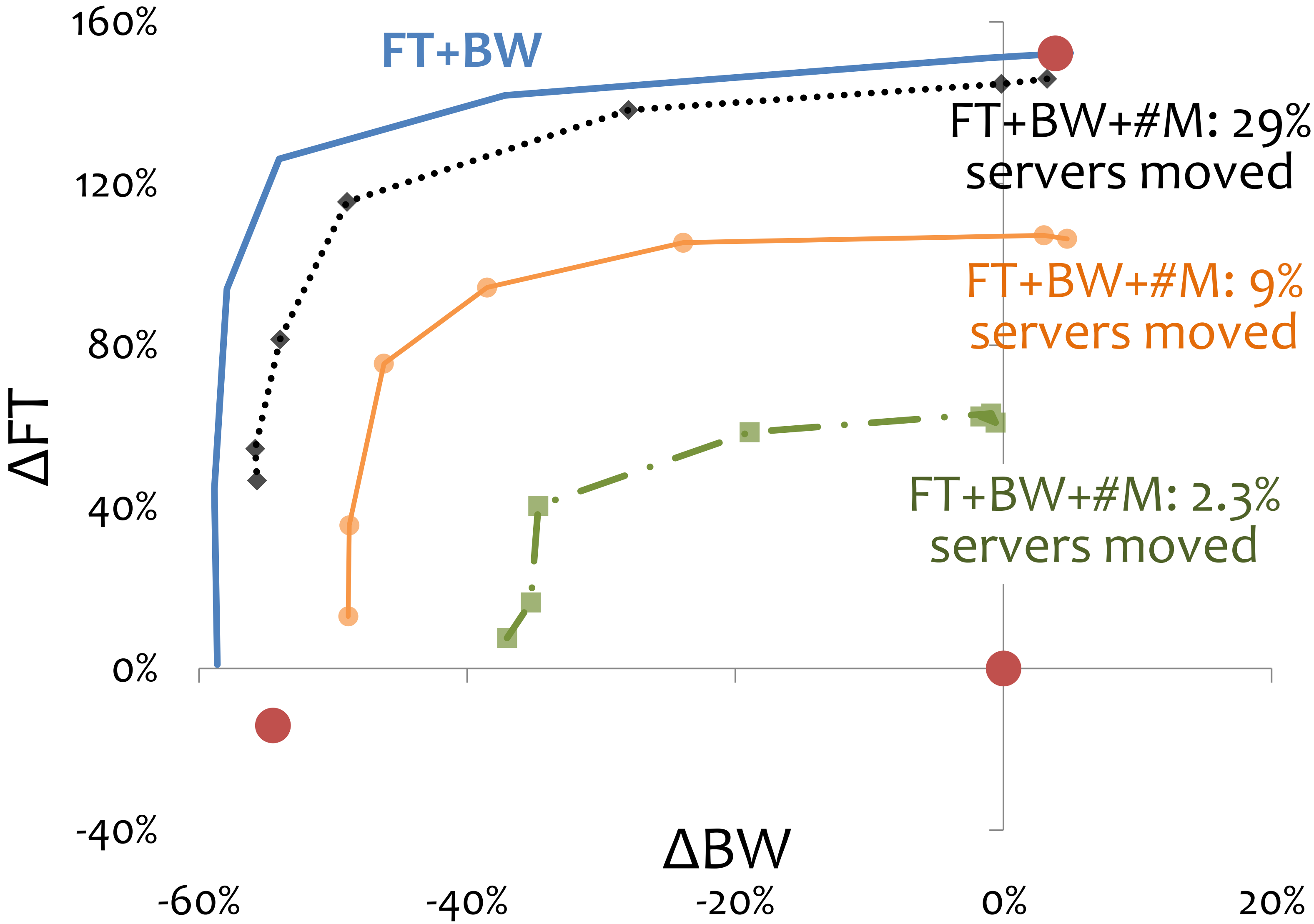

160%
120%
80%
40%
0%
-40%
ΔFT
Cut+FT+BW
(moves all servers)
ΔBW
-60%
-40%
-20%
0%
20%

Evaluation



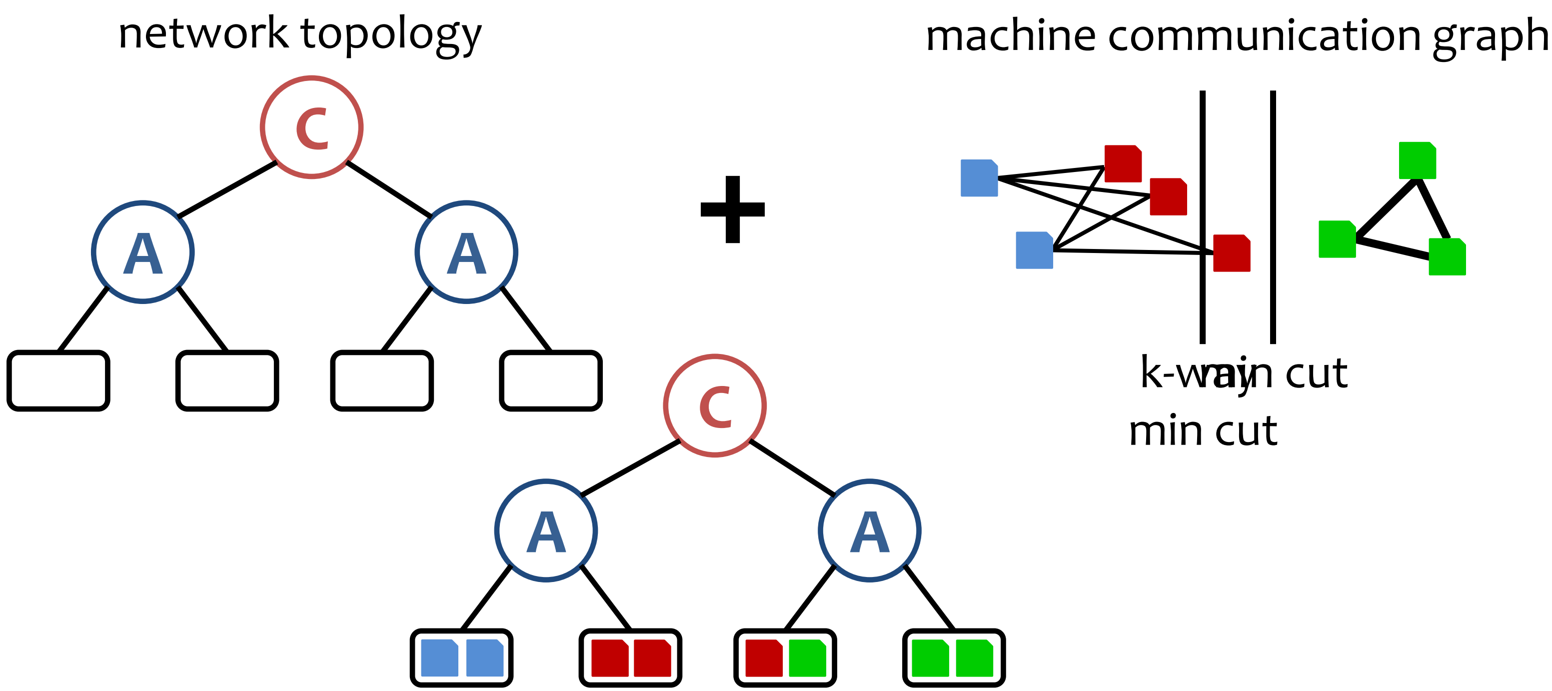
160%
120%
80%
40%
0%
-40%
FT+BW
ΔFT
FT+BW+#M: 2.3% servers moved
ΔBW
-60%
-40%
-20%
0%
20%

Evaluation



160%
120%
80%
40%
0%
-40%
-60%
-40%
-20%
0%
20%
ΔFT
ΔBW
FT+BW
FT+BW+#M: 29% servers moved
FT+BW+#M: 9% servers moved
FT+BW+#M: 2.3% servers moved

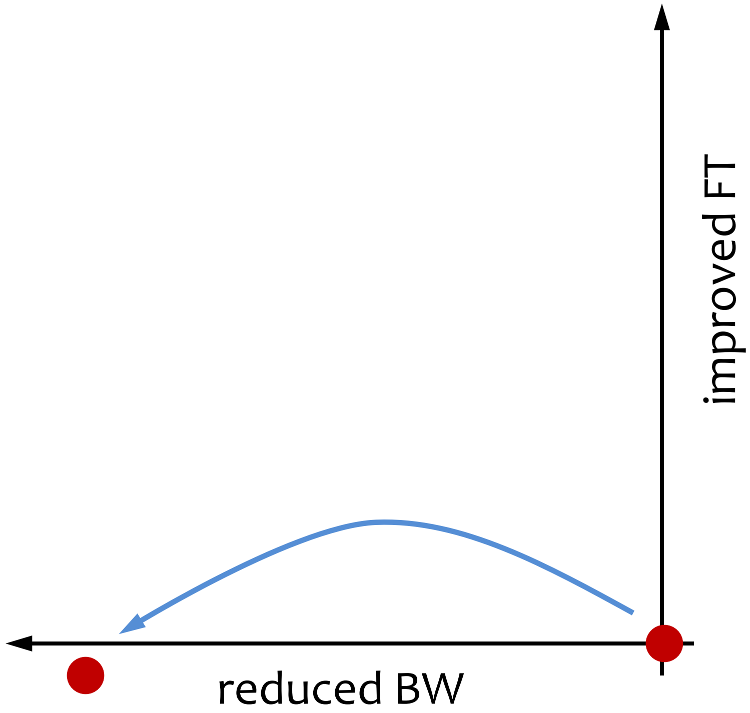
BW k-way graph cut



k-way min graph cut

– ignores #M: reshuffles almost all machines
– ignores FT: can't be easily extended

BW k-way graph cut



Scaling algorithms to large datacenters

Only evaluate a small, random set of swaps

- symmetry => many "good" swaps exist

Cell = set of machines with same fault domains

Reduce size of communication graph for cut

FT BW cut + steepest descent

Step 1: min-cut

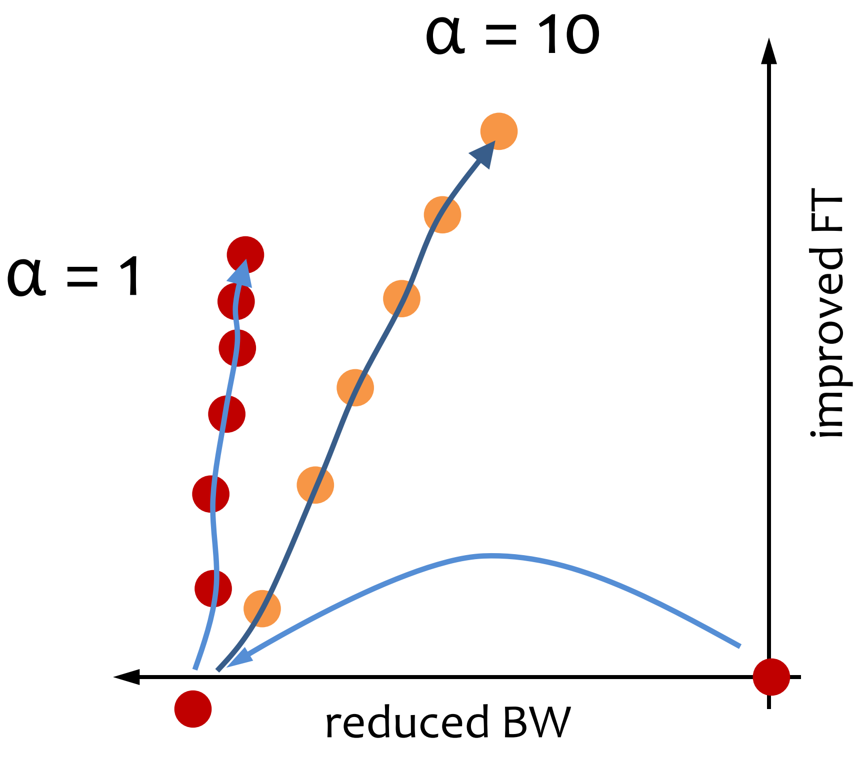
– optimizes BW

Step 2: steepest descent on FTC + α BW

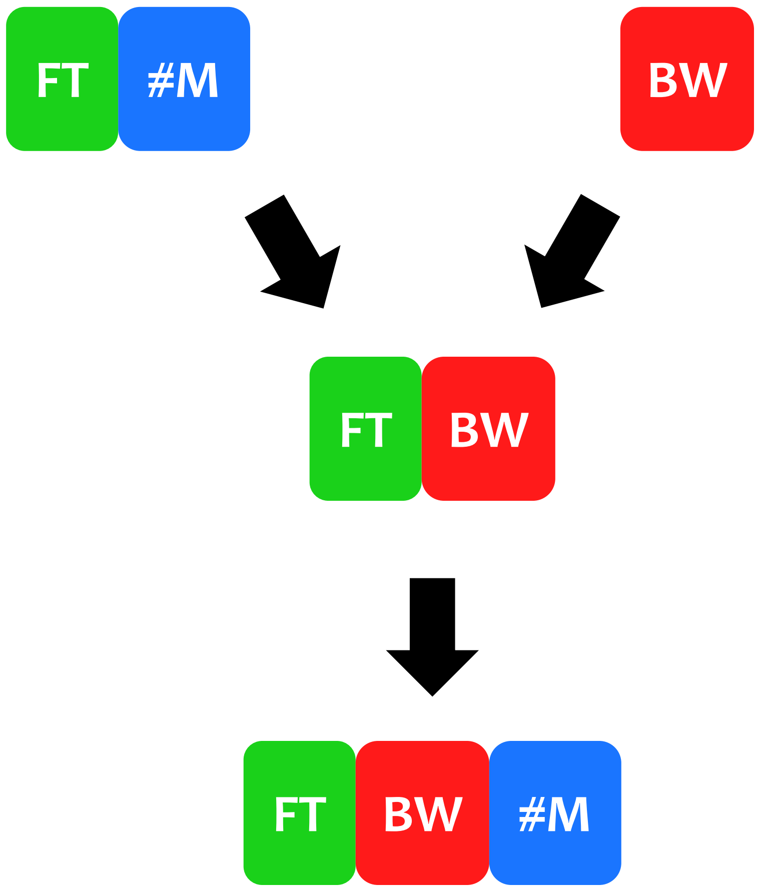
– non-convex
– no guarantees on reaching optimum
– α determines the trade-off

Reshuffles all machines

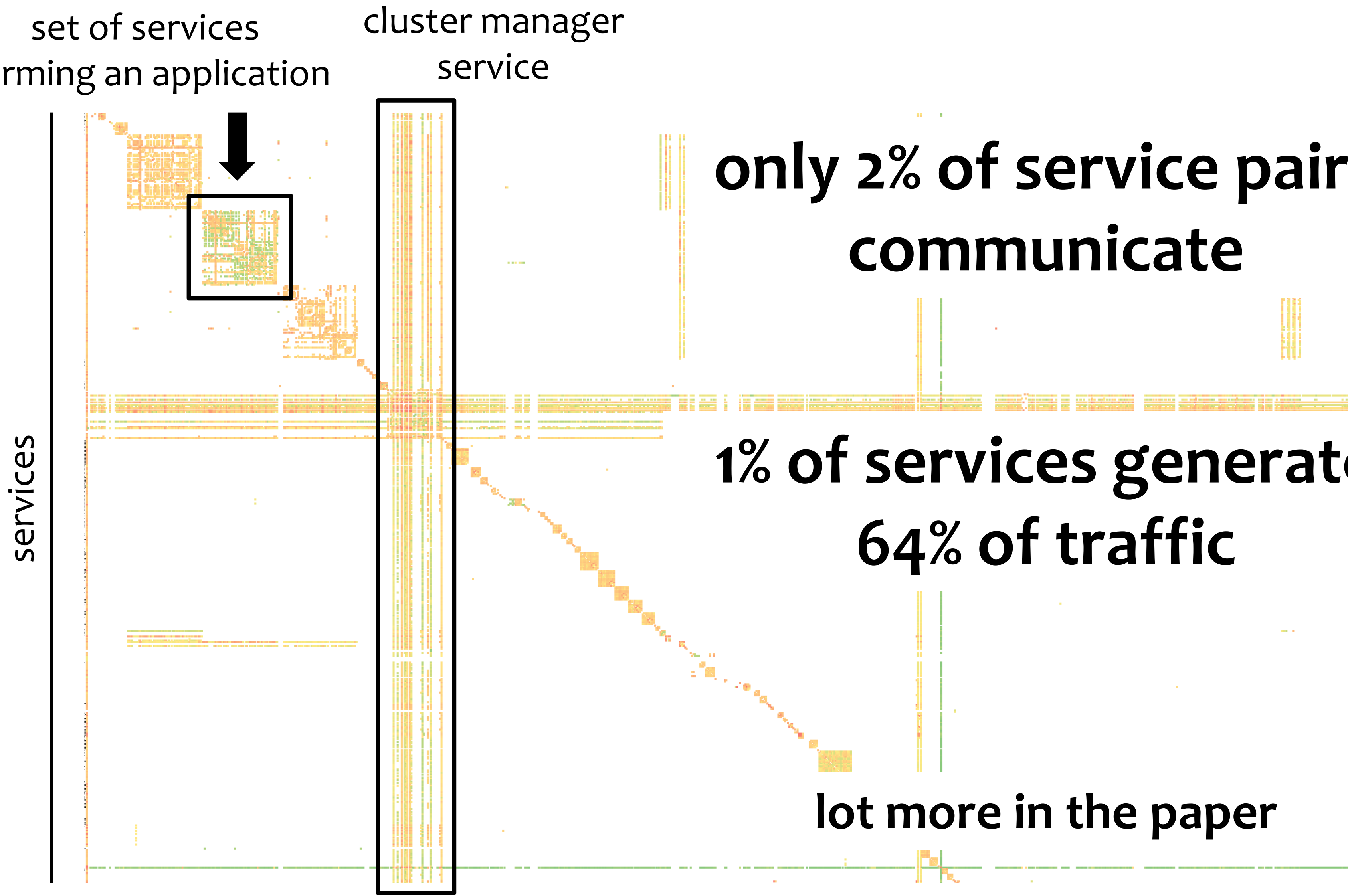
FT BW cut + steepest descent



Properties of allocation algorithms



Service communication matrix is very sparse and skewed



only 2% of service pairs communicate

1% of services generate 64% of traffic

lot more in the paper

Which metrics matter?

FT: fault tolerance

- service should survive infrastructure failures
- failures despite redundancy

BW: bandwidth usage

- reduce usage on constrained links
- lower cost of infrastructure

#M: number of moves

- moving some servers is expensive
- want incremental allocation