SpringFS: Bridging Agility and Performance in Elastic Distributed Storage

Lianghong Xu

Jim Cipar, Elie Krevat, Alexey Tumanov, Nitin Gupta,

Mike Kozuch (Intel), Greg Ganger

Carnegie Mellon University

Carnegie Mellon
Parallel Data Laboratory

Elasticity in Distributed Storage

- "Elasticity" in distributed storage:
  - ability to resize dynamically as workload varies
  - More difficult than elastic computing
- Benefits
  - Re-use for other purposes or reduce energy usage
  - Save machine hours (operating cost)
- Most distributed storage is not elastic
  - Designed for load balancing, not elasticity
  - E.g., GFS and HDFS
  - Deactivating servers may make data unavailable

SpringFS Contributions

1. Fill the gap

Some elasticity
Max peak write performance

Max elasticity
Sub-max peak write performance

2. Propose and address agility

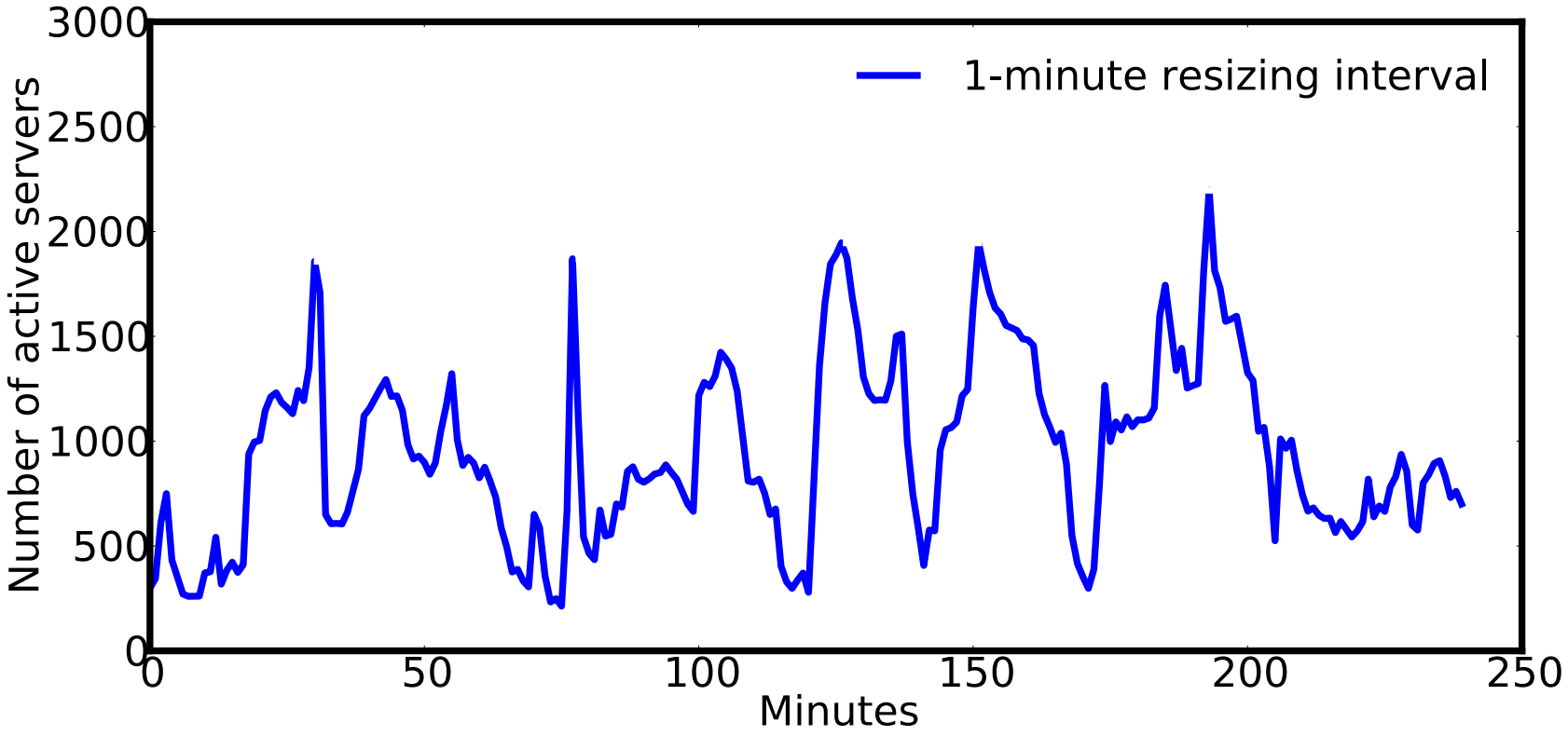
"**Agility**": speed of elastic resizing

minimize data migration ⇒ minimize machine hours

Key metrics

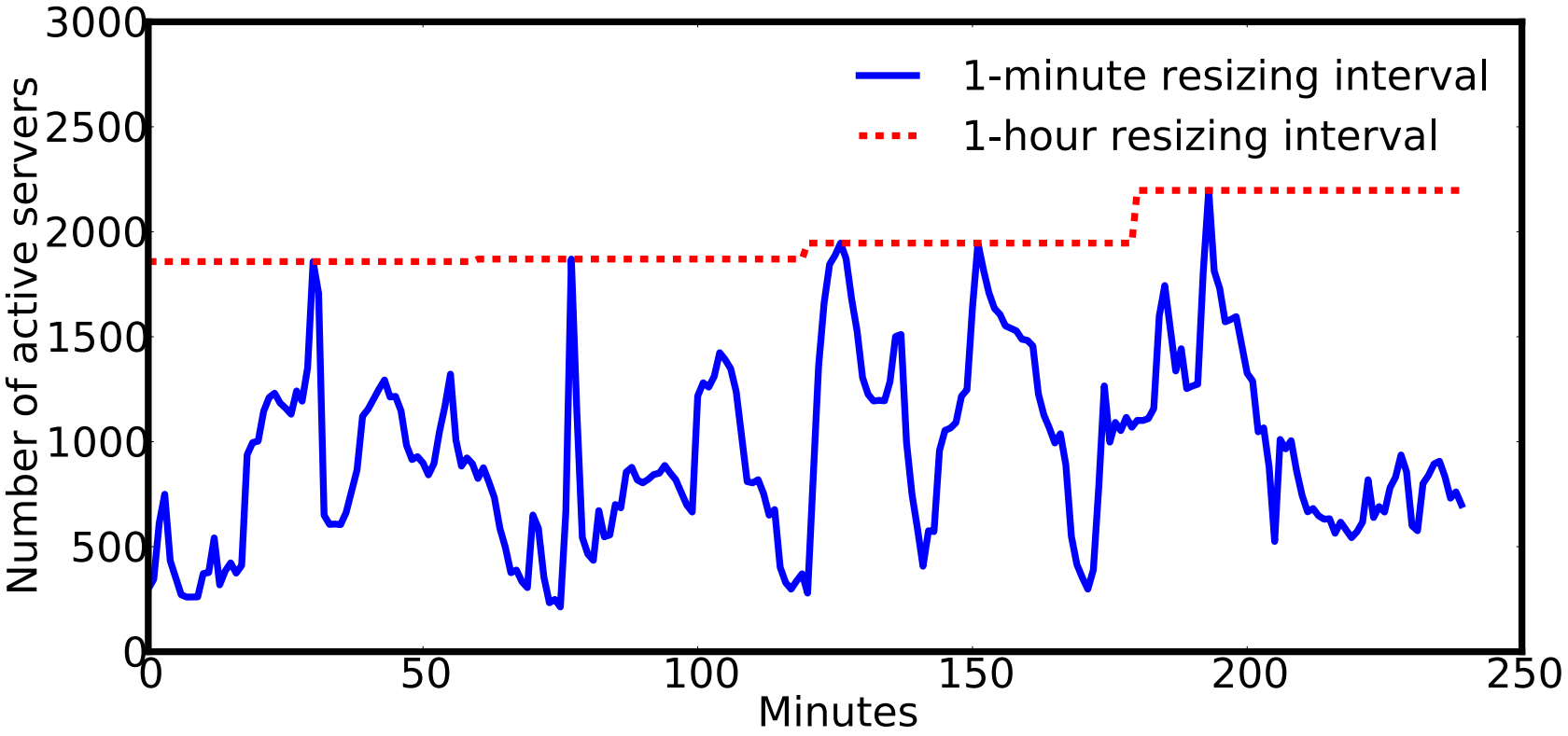
Agility is Important

"Burstiness" in the Facebook HDFS trace



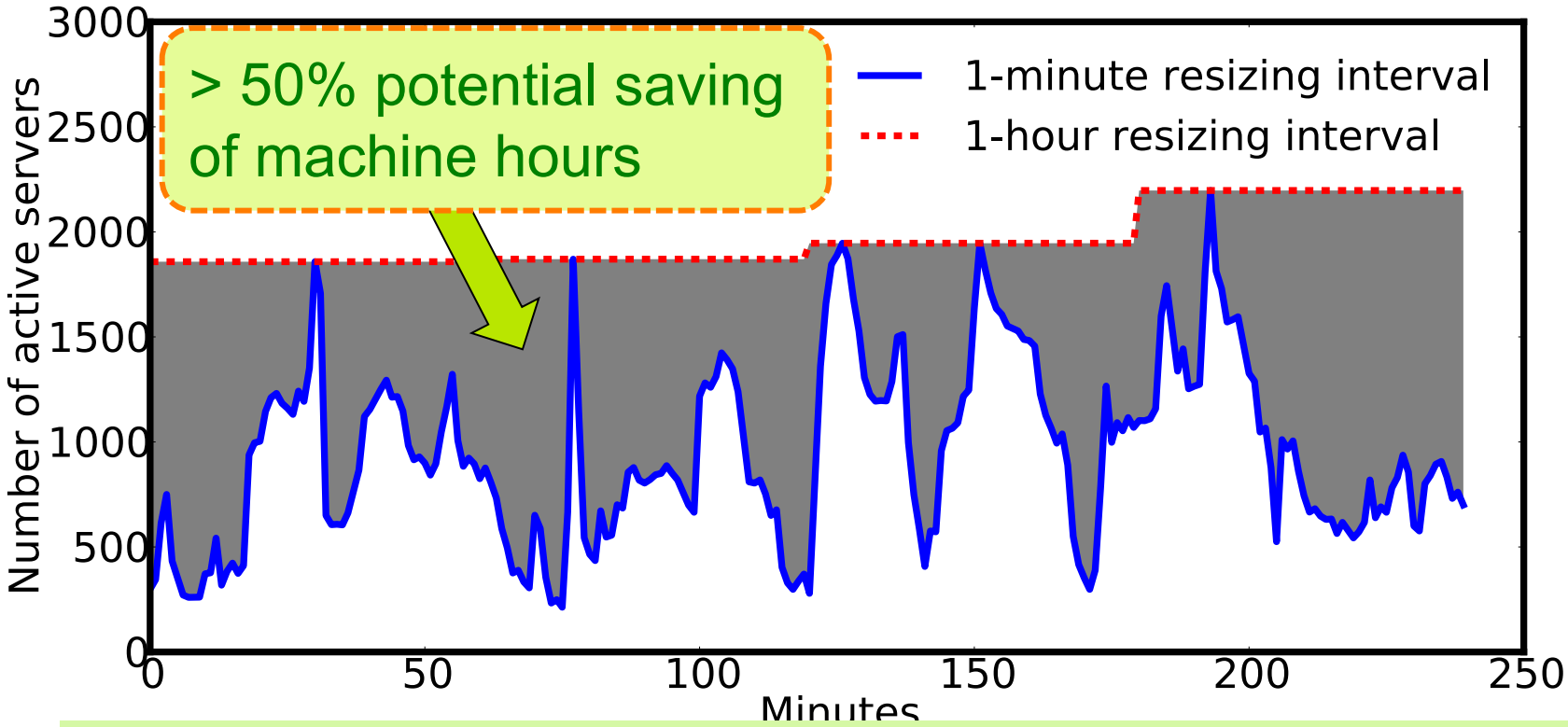
Agility is Important

"Burstiness" in the Facebook HDFS trace



Agility is Important

"Burstiness" in the Facebook HDFS trace

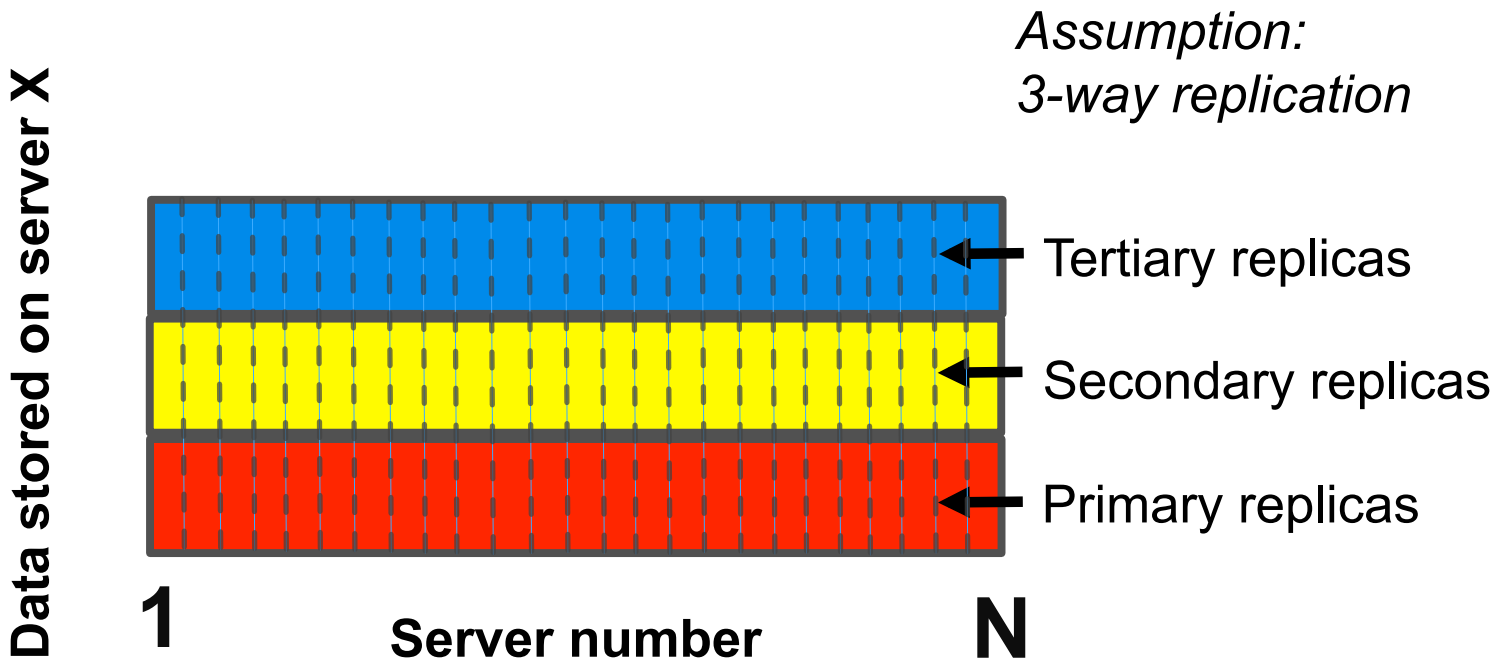


Agility allows close tracking of workload variation

Outline

- Introduction
- **Background and motivation**
- SpringFS design
- Evaluation
- Conclusion

Non-elastic Example: HDFS



Pseudo-random placement, even data layout

Almost all servers must be “on” to ensure 100% availability

- Little potential for elastic resizing

Data Layout in Elastic Storage

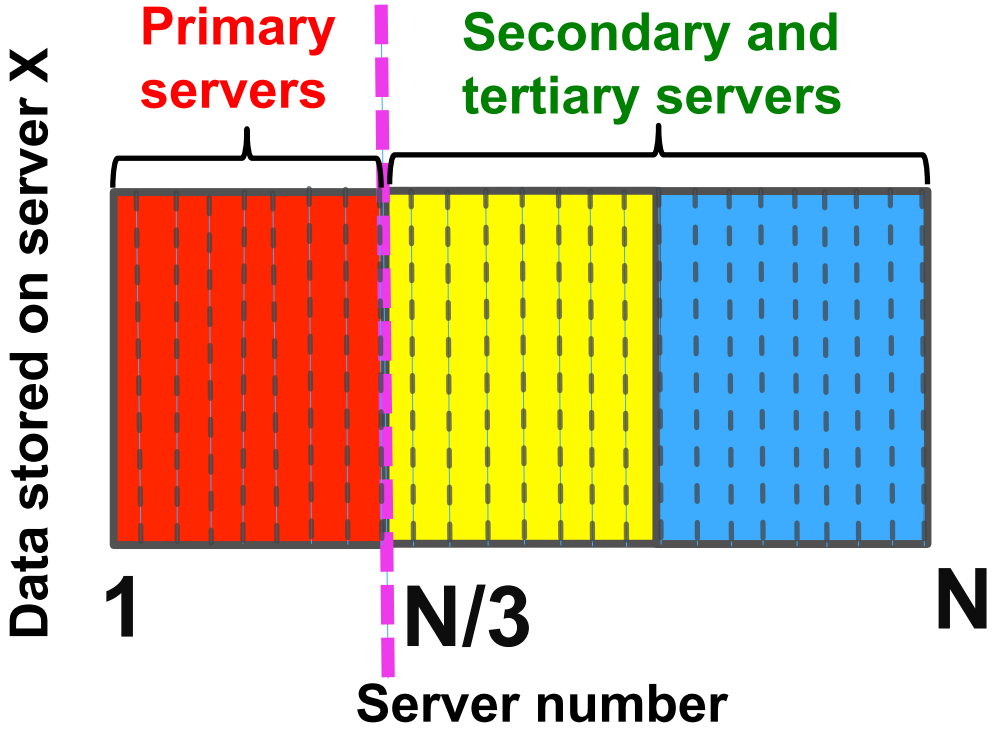
- General rule:
  - Take advantage of replication
  - Always keep the first (primary) replicas “on”
  - The other replicas can be activated on demand

- Notable examples: Sierra [1] and Rabbit [2]

[1] E. Thereska et al. Sierra: Practical Power-proportionality for Data Center Storage. Eurosys 2011.

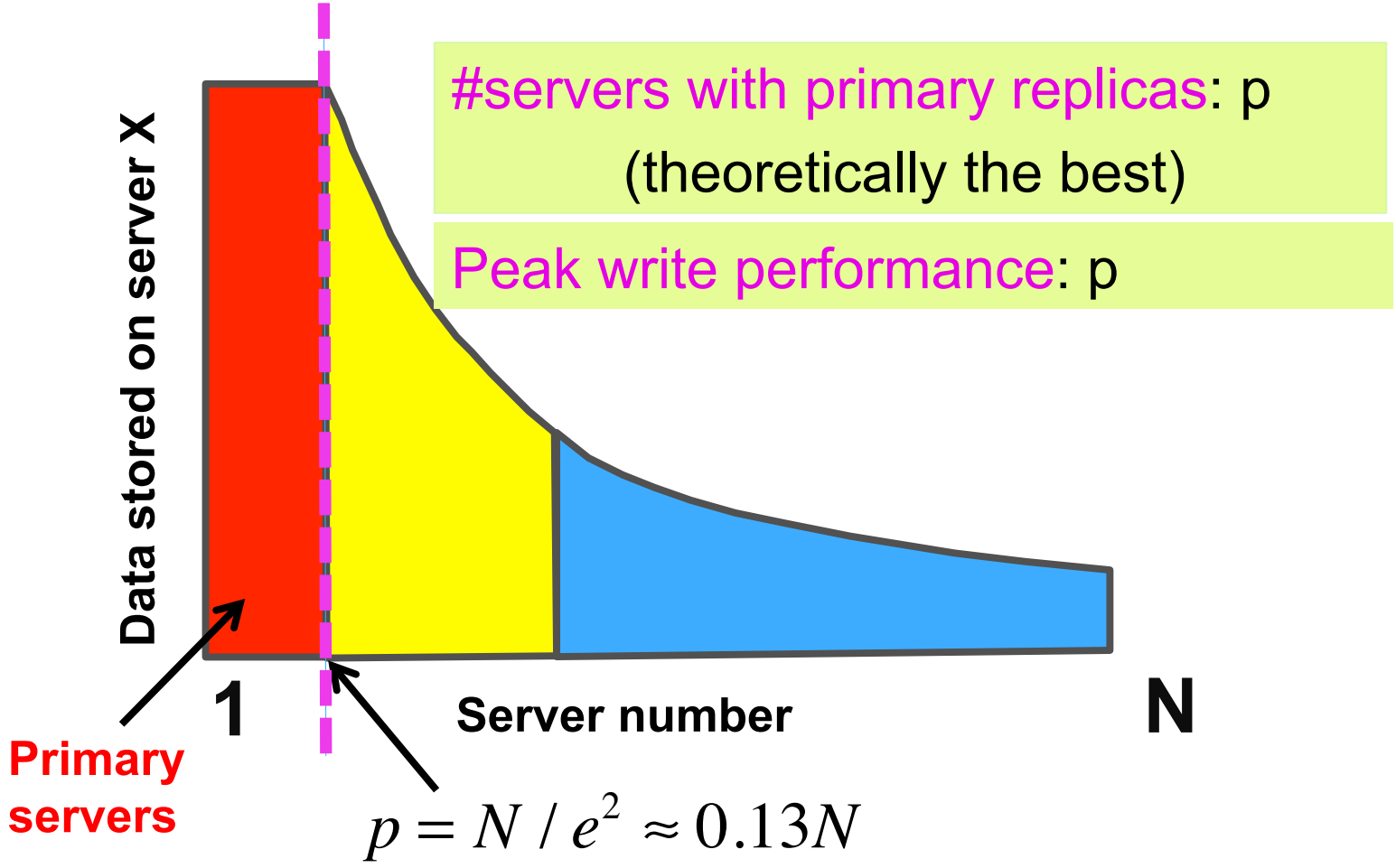
[2] J. Cipar et al. Robust and flexible power-proportional storage. SoCC 2010.

Sierra Data Layout

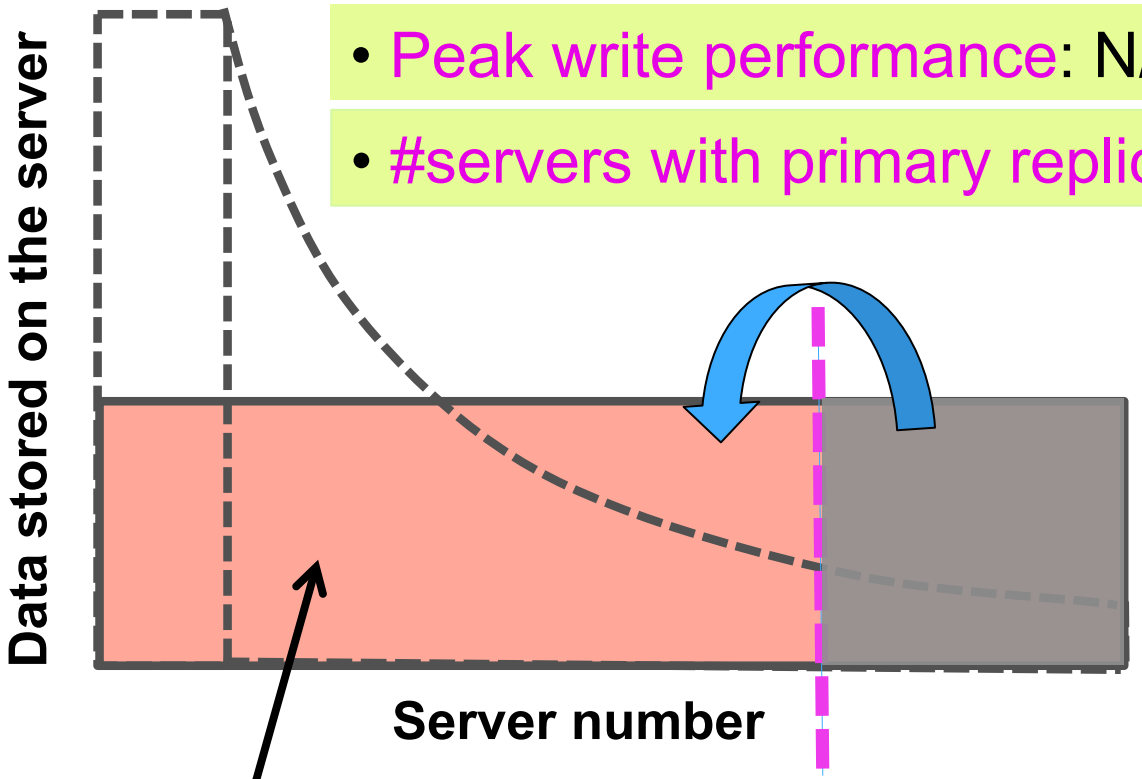


- Peak write performance: N/3 (as good as HDFS)
- #servers with primary replicas: N/3

Rabbit Equal-work Data Layout

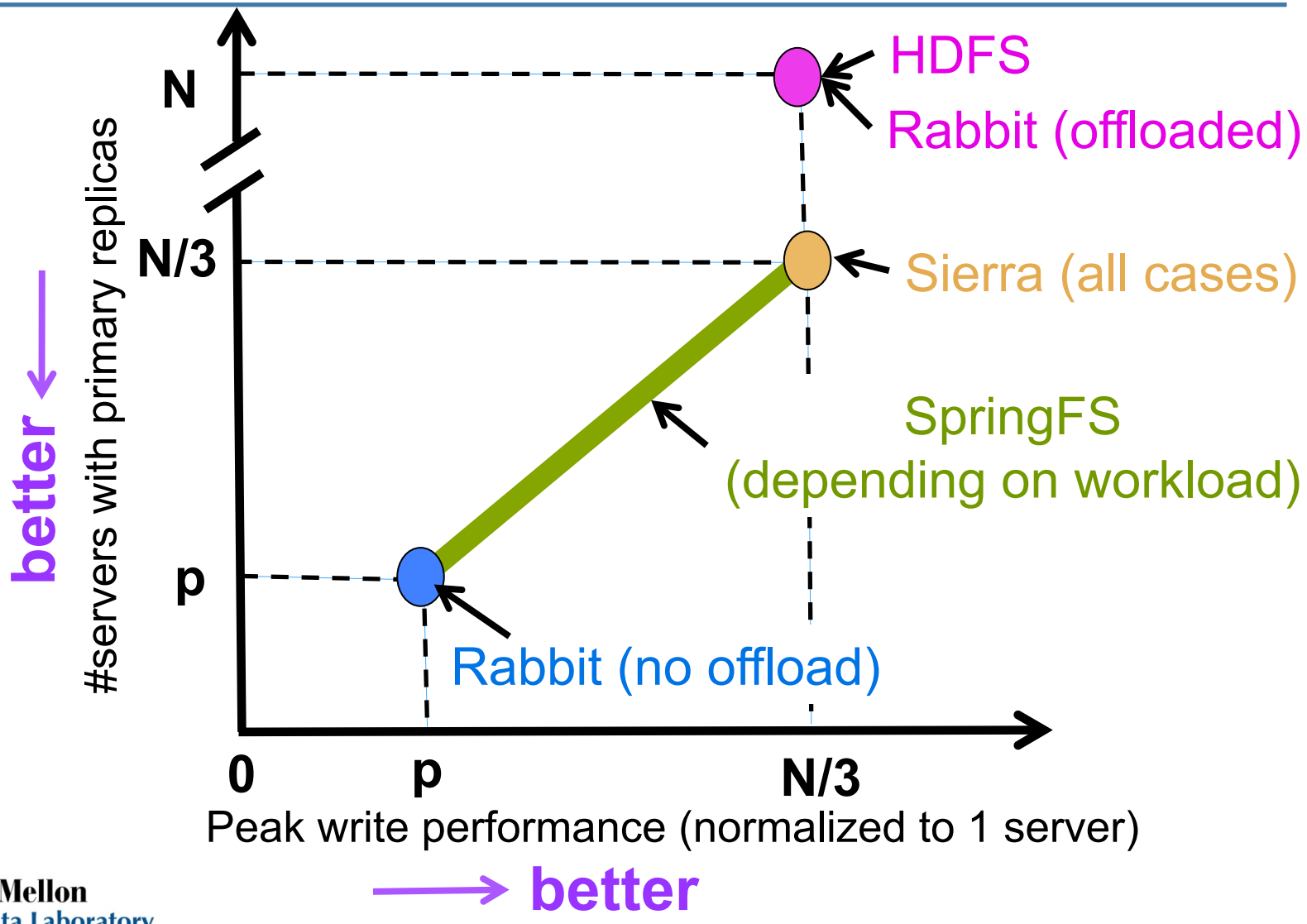


Rabbit When Using Offloading



Primary replicas spread across all active servers

Tradeoff Space



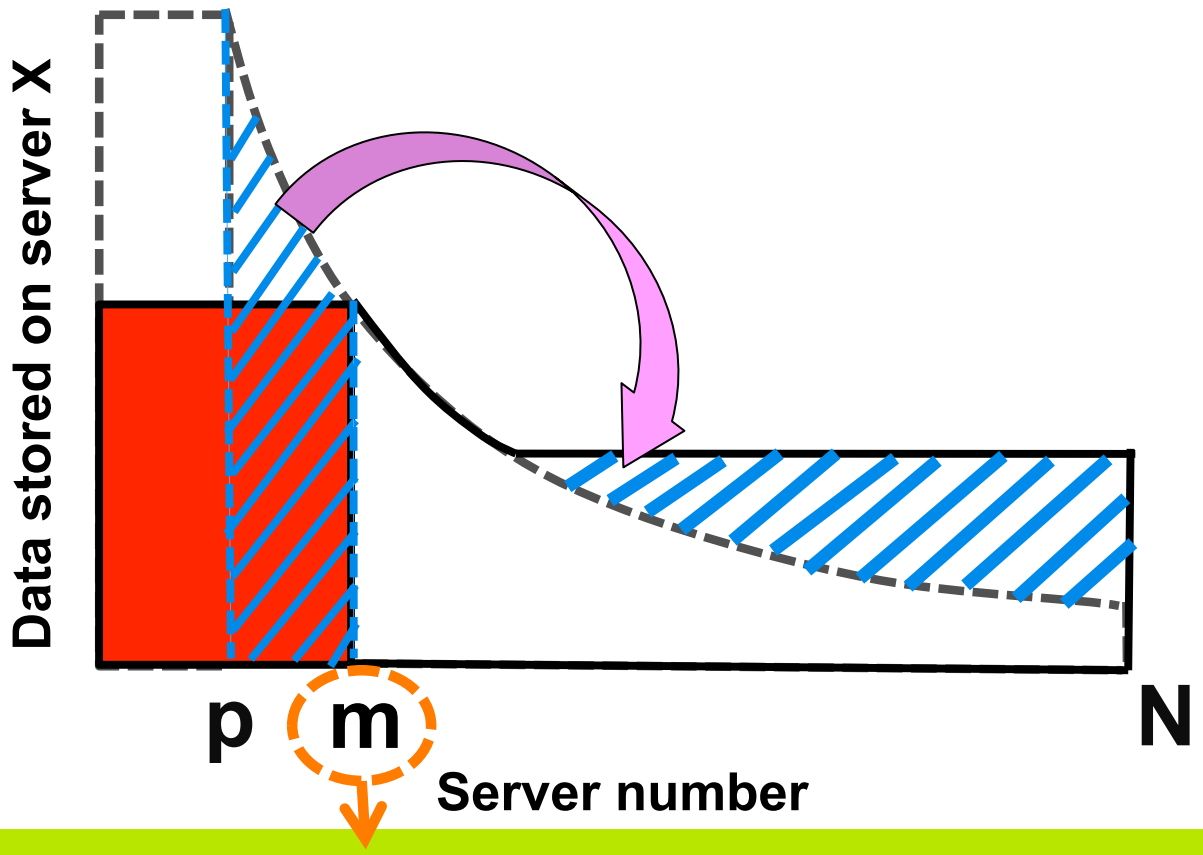
Outline

- Introduction
- Background and motivation
- **SpringFS design**
- Evaluation
- Conclusion

SpringFS Design

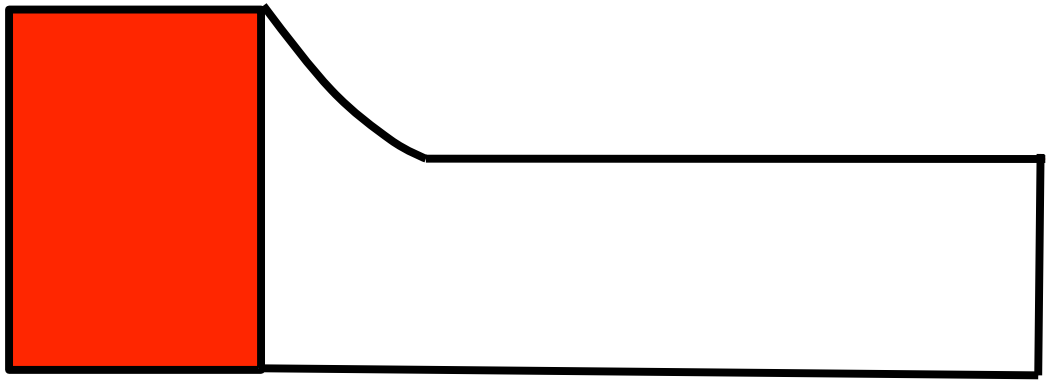
- ***Bounded write offloading***
  - Dynamically constrain distribution of primary replicas
- ***Read offloading***
  - Preferentially offload reads from write-heavy servers
- ***Passive migration***
  - Delay migration on server re-integration

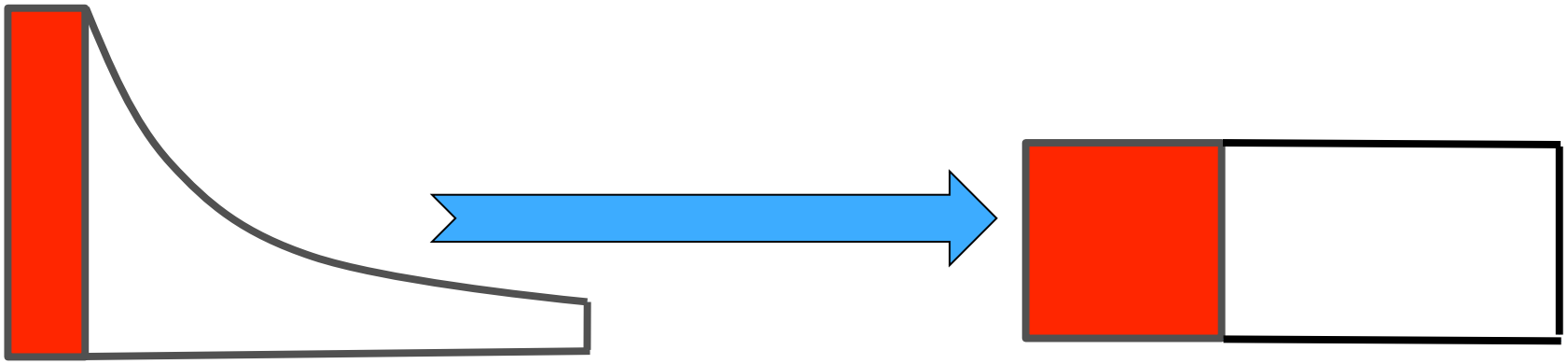
Bounded Write Offloading



Offload set: automatically adapts to workload

Bounded Write Offloading





SpringFS Implementation

- Modified instance of HDFS
  - Written in Java and Python
- Built and used a Scriptable HDFS interface
  - Data placement
  - Load balancing
  - Data Migration
  - Implement SpringFS and Rabbit in the same system
- Resizing agent
  - Activate/deactivate servers according to workload

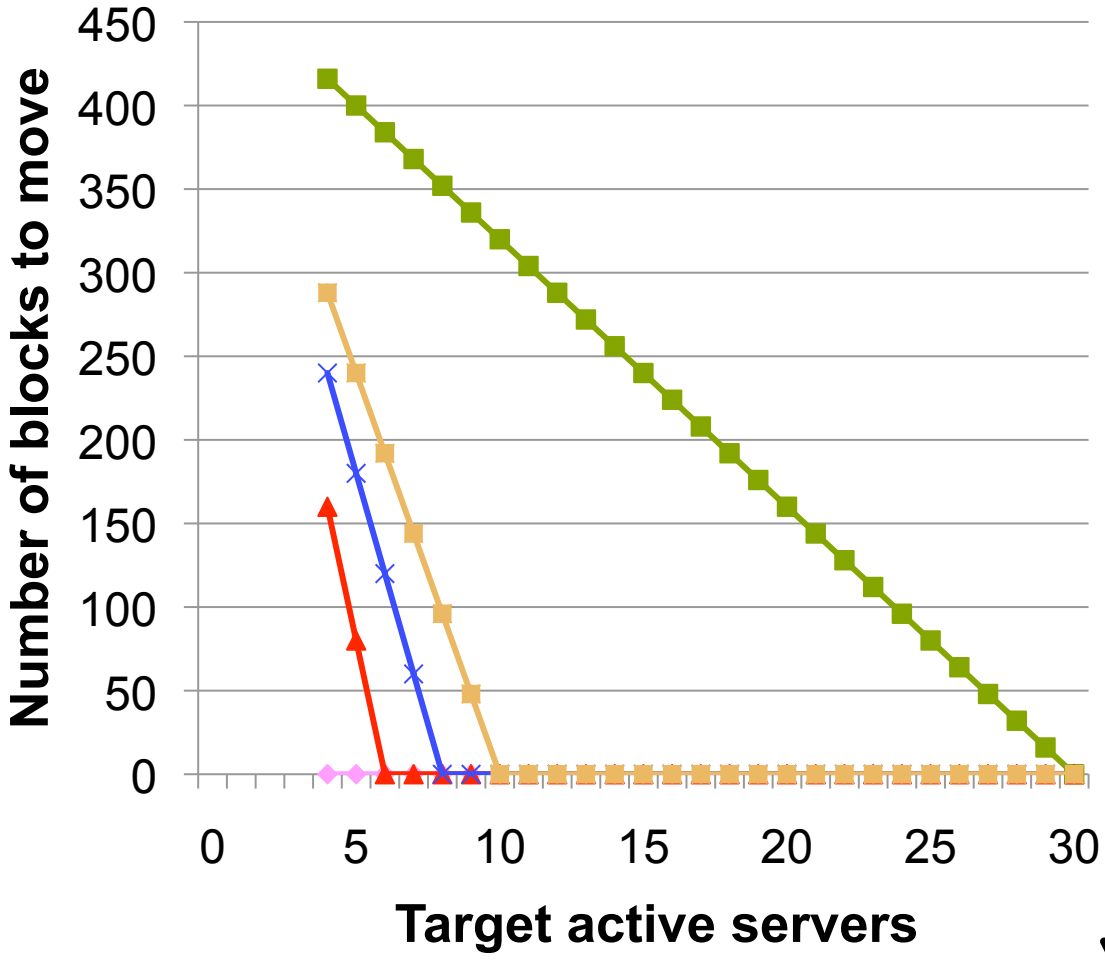
Outline

- Introduction and motivation
- Background
- SpringFS design
- **Evaluation**
- Conclusion

Evaluation Overview

- Experiments with SpringFS prototype
- Analysis with real-world traces
  - Hadoop/HDFS deployments at Facebook (FB) and Cloudera Customers (CC)
- Summary of results
  - SpringFS improves over state-of-the-art designs
    - Reduces data migration
    - Reduces machine hour usage (often near-ideal)
    - Provides range of options between Rabbit and Sierra

Data Migration in SpringFS Prototype



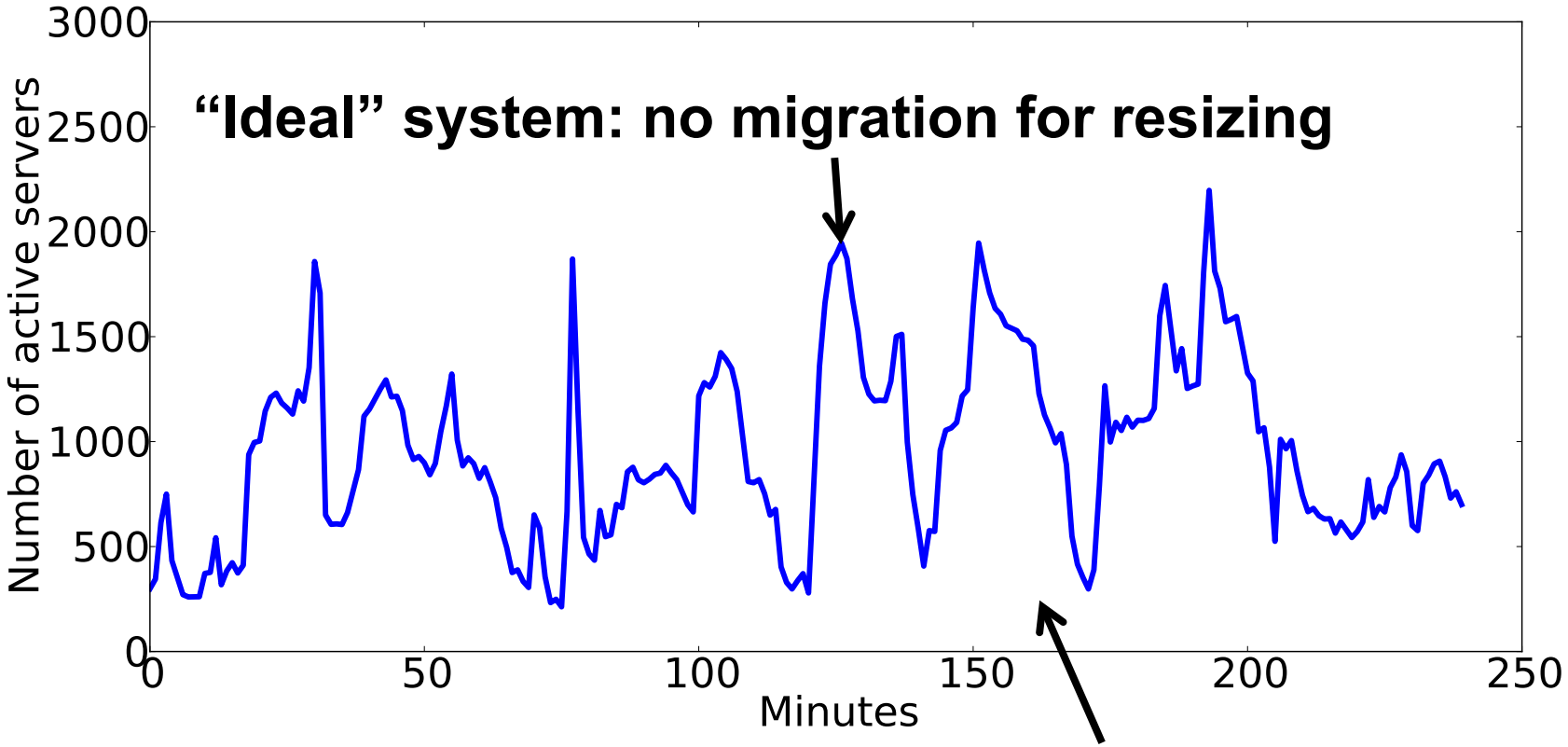
Experimental setup:

- 30 DataNodes
- 4 primary servers
- 2GB file per server
- 128MB block size

Rabbit (no offload)
Rabbit (offload=30)
SpringFS(offload=6)
SpringFS(offload=8)
SpringFS(offload=10)

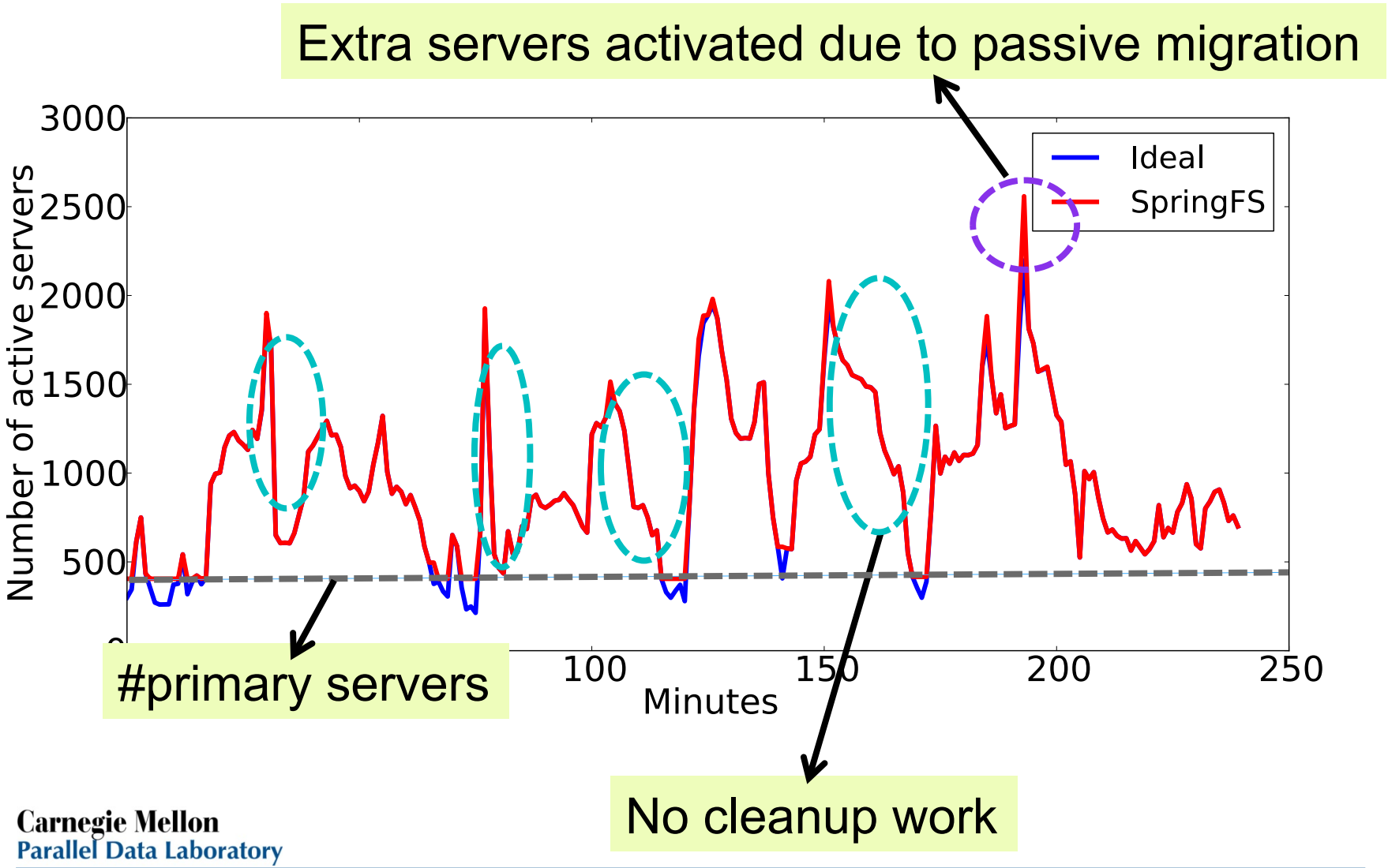
Same performance

Policy Analysis with the Facebook Trace

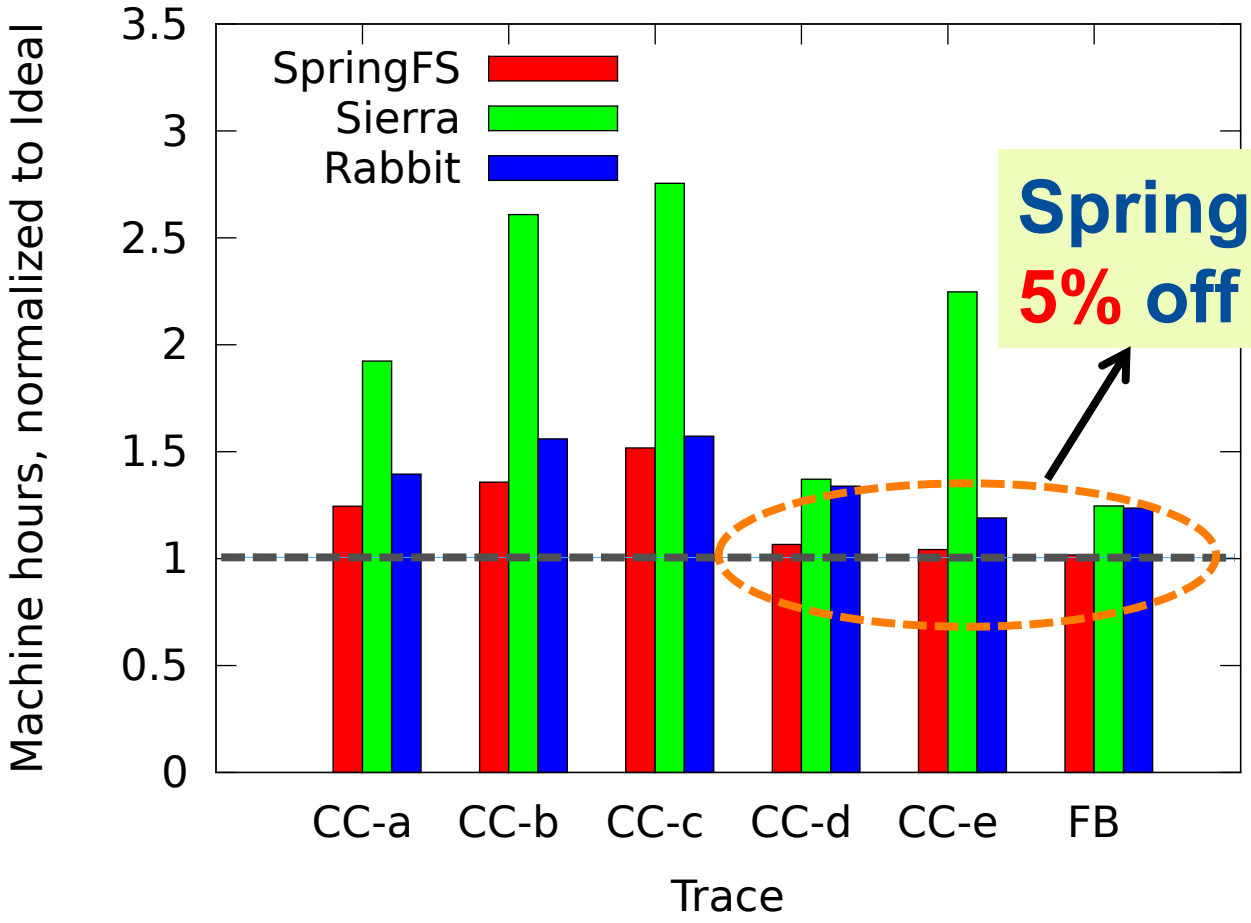


Area under curve: aggregate machine hour usage

Machine Hours (SpringFS vs. Ideal)



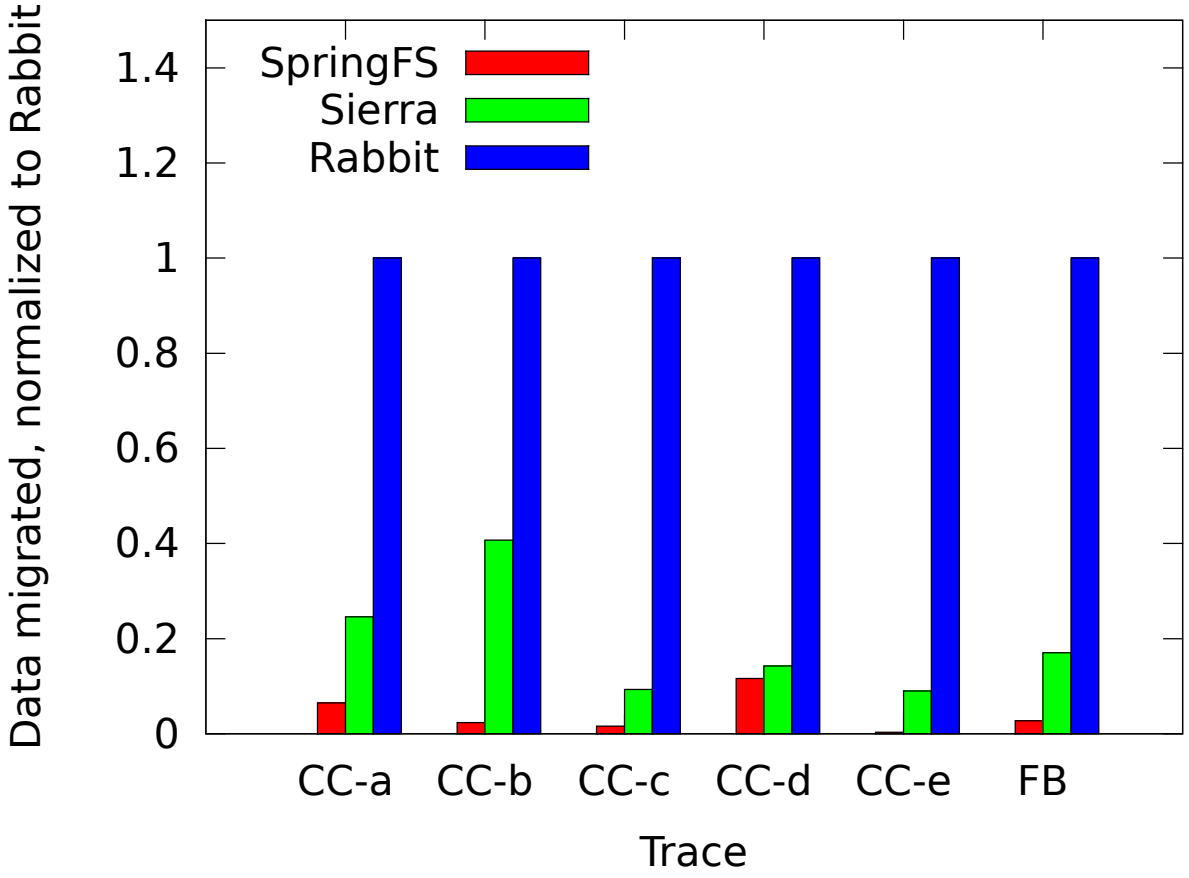
Machine Hour Usage



SpringFS within 5% off "Ideal"

SpringFS: 6-120% improvement

Data Migration



SpringFS: 9-208X improvement

Conclusion

- SpringFS: a new elastic distributed storage
  - Fills the gap in state-of-the-art designs
- Agility is important
  - Ability to track workload burstiness
- Address agility by minimizing data migration
  - Bounded write offloading
  - Read offloading + passive migration
- Much lower machine hour usage