Scaling State Machine Replication

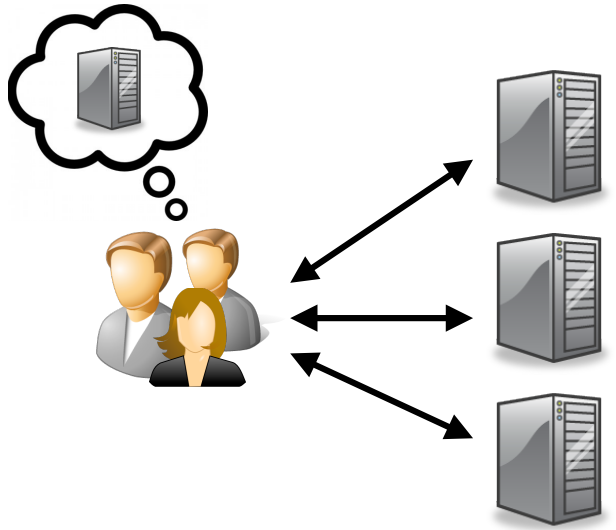
Fernando Pedone
University of Lugano (USI)
Switzerland

State machine replication

- Fundamental approach to fault tolerance
  - ✦ Google Spanner
  - ✦ Apache Zookeeper
  - ✦ Windows Azure Storage
  - ✦ MySQL Group Replication
  - ✦ Galera Cluster, …

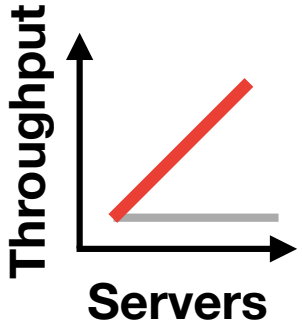
State machine replication is intuitive & simple

- Replication transparency
  - ✦ For clients
  - ✦ For application developers
- Simple execution model
  - ✦ Replicas **order** all commands
  - ✦ Replicas **execute** commands deterministically and in the same order



Configurable fault tolerance but bounded performance

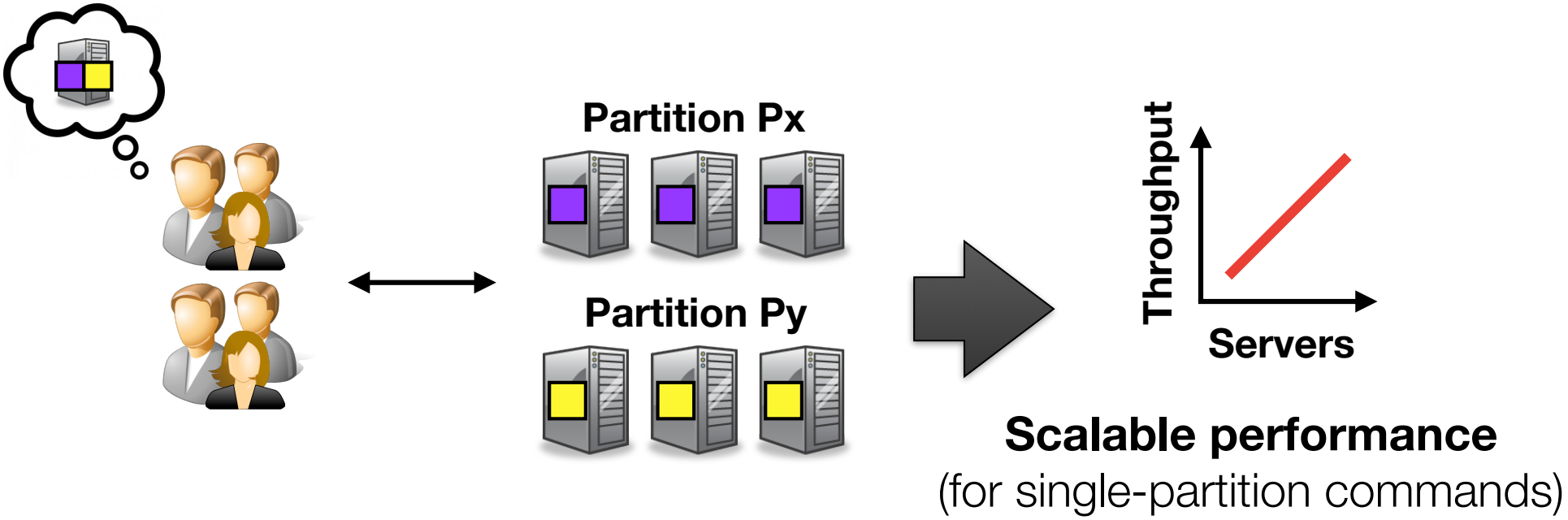
- Performance is bounded by what one replica can do
  - ✦ Every replica needs to execute every command
  - ✦ More replicas: same (if not worse) performance



How to scale state machine replication?

Scaling performance with partitioning

- Partitioning (aka sharding) application state

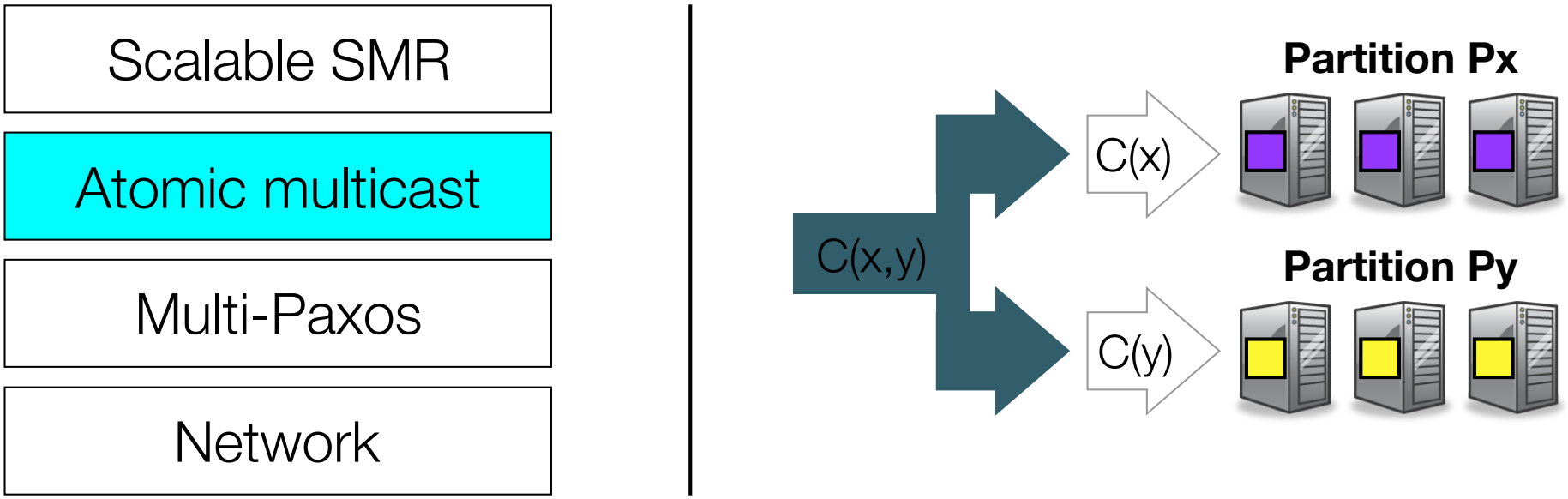


Problem #1: How to order commands in a partitioned system?

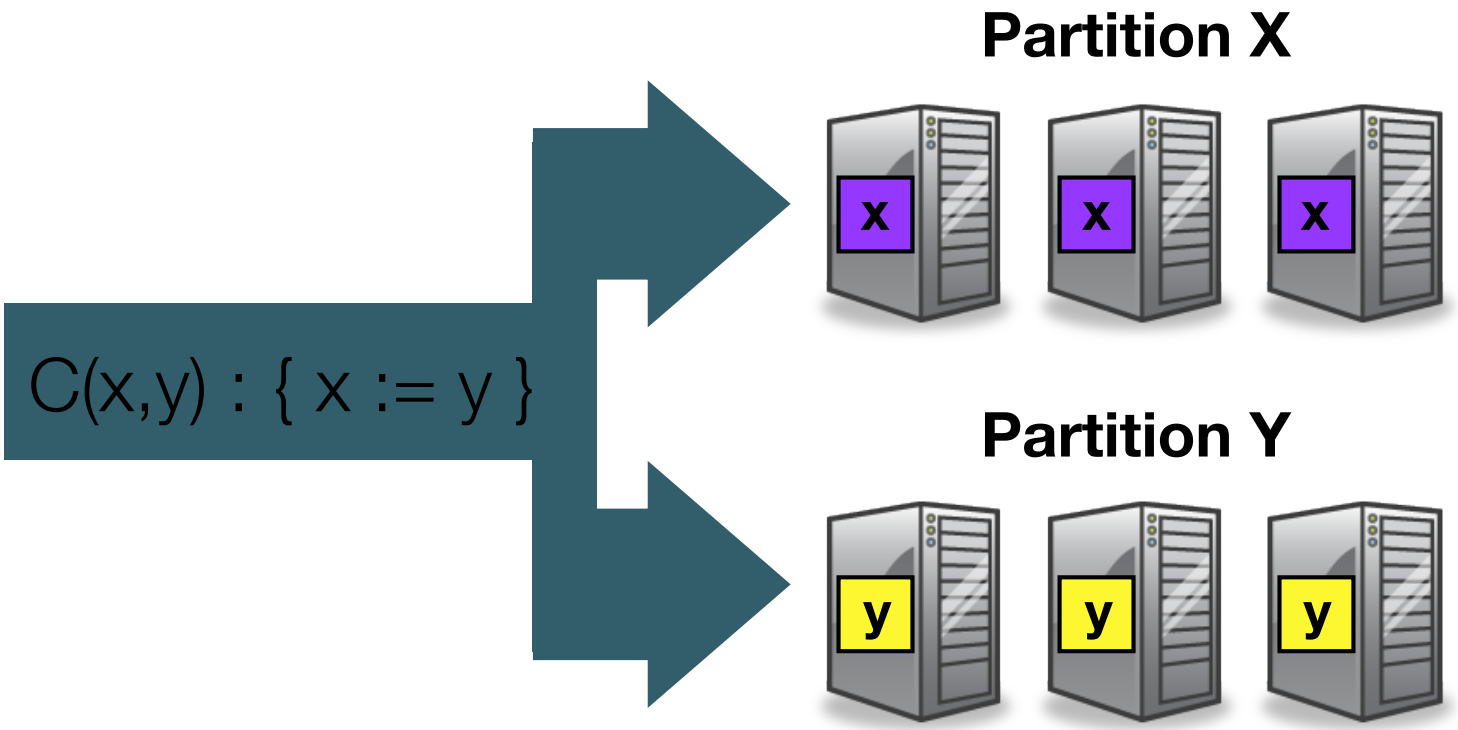
Problem #2: How to execute commands in a partitioned system?

Ordering commands in a partitioned system

- Atomic multicast
  - ✦ Commands addressed (multicast) to one or more partitions
  - ✦ Commands ordered within and across partitions
    - If S delivers C before C', then no S' delivers C' before C



Executing multi-partition commands



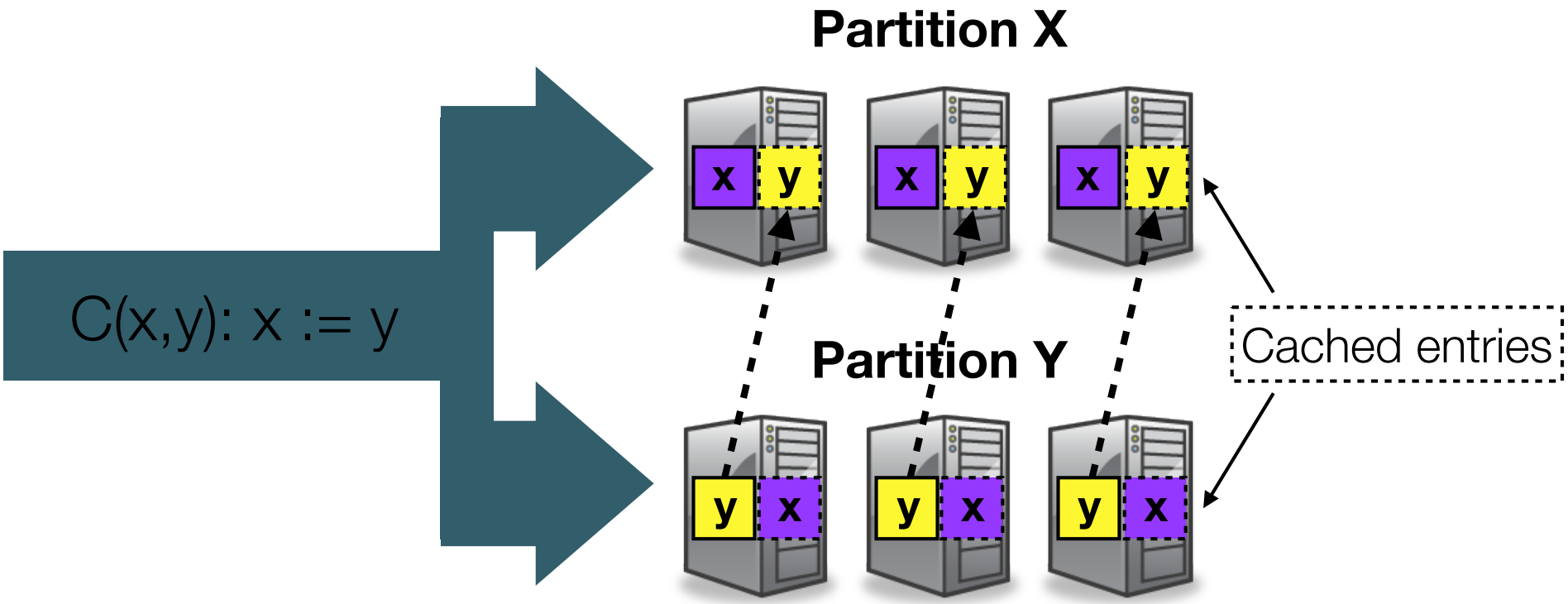
Solution #1: Static partitioning of data

Solution #2: Dynamic partitioning of data

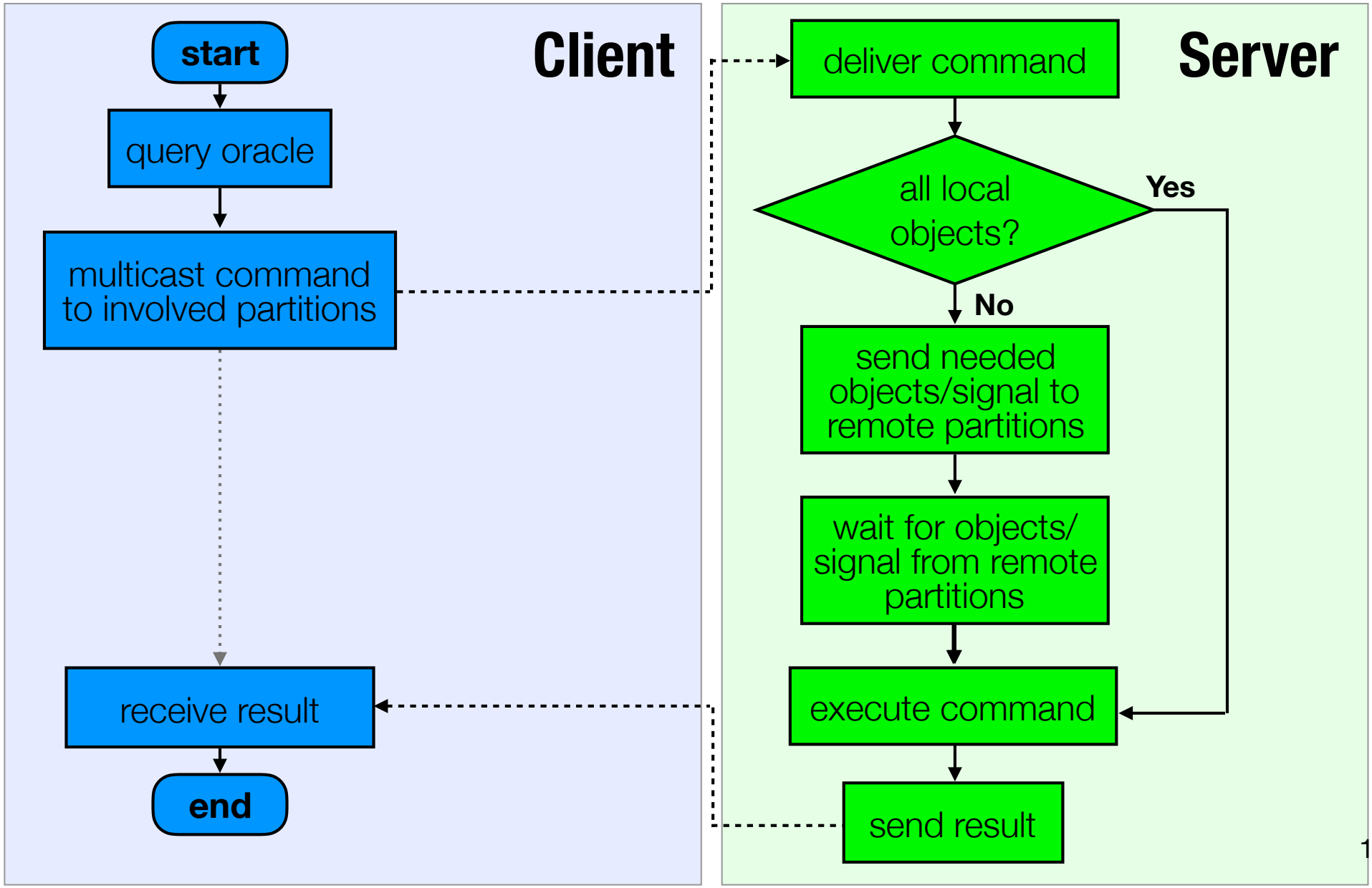
Solution 1: Static partitioning of data

- Execution model
  - ✦ Client queries location oracle to determine partitions
  - ✦ Client multicasts command to involved partitions
  - ✦ Partitions exchange and temporary store objects needed to execute multi-partition commands
  - ✦ Commands executed by all involved partitions
- Location oracle
  - ✦ Simple implementation thanks to static scheme

How to execute multi-partition commands?



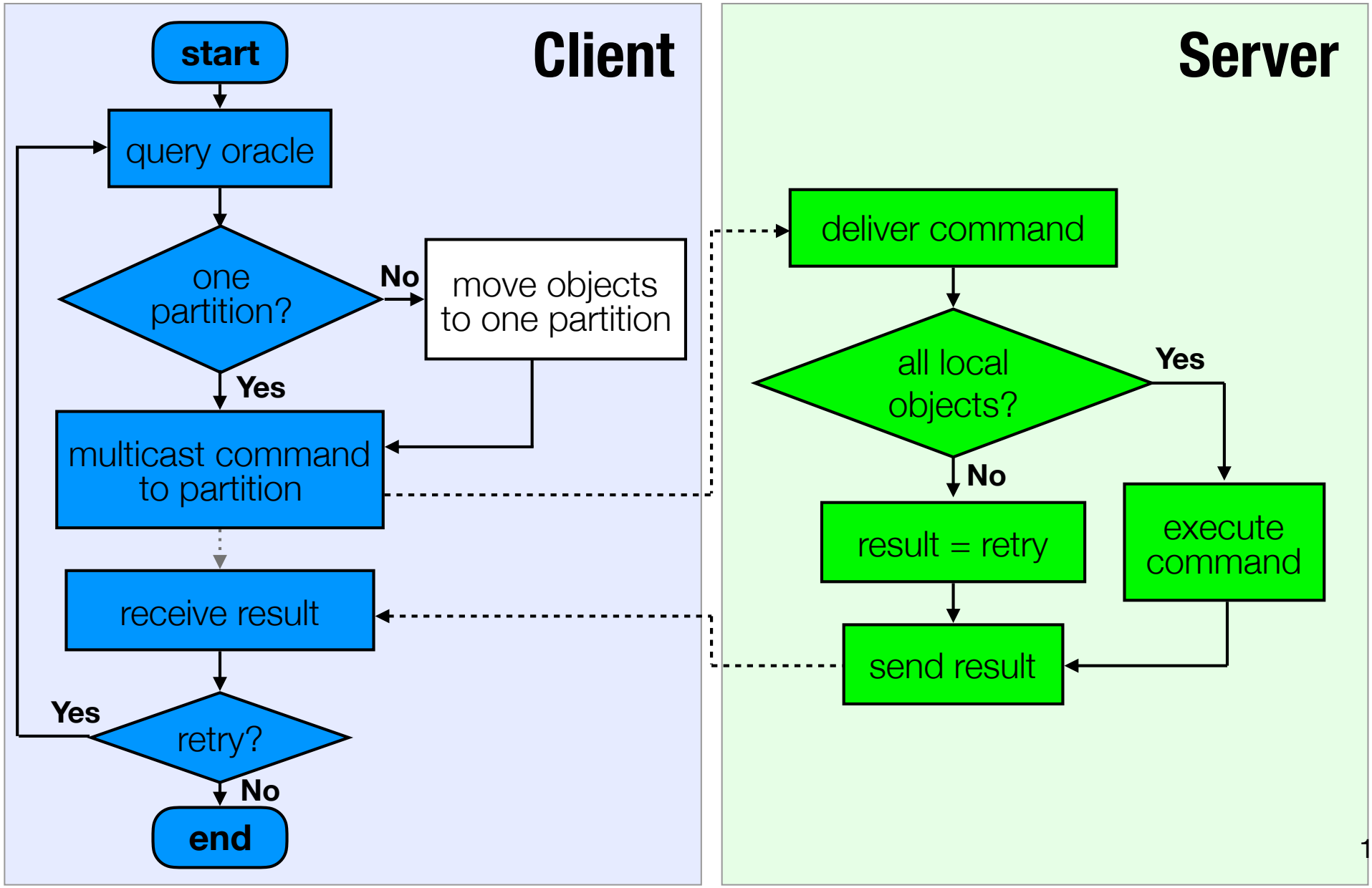
Static scheme, step-by-step



Solution 2: Dynamic partitioning of data

- Execution model (key idea)
  - ✦ Turn every command single-partition
  - ✦ If command involves multiple partitions, move objects to a single partition before executing command
- Location oracle
  - ✦ Oracle implemented as a “special partition”
  - ✦ Move operations involve oracle, source and destination partitions

Dynamic scheme, step-by-step



Termination and load balance

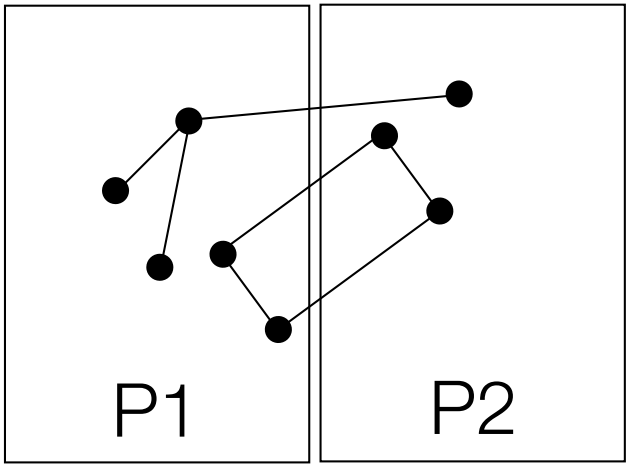
- Ensuring termination of commands
  - ✦ After retrying n times, command is multicast to all partitions
  - ✦ Executed as a multi-partition command
- Ensure load balancing among partitions
  - ✦ Target partition in multi-partition command chosen randomly

Oracle: high availability and performance

- Oracle implemented as a partition
  - ✦ For fault tolerance
- Clients cache oracle entries
  - ✦ For performance
  - ✦ Real oracle needed at first access and when objects change location
  - ✦ Client retries command if cached location is stale

Dynamically (re-)partitioning the state

- Decentralized strategy
  - ✦ Client chooses one partition among involved partitions
  - ✦ Each move involves oracle and concerned partitions
  - 👍 ✦ No single entity has complete system knowledge
  - 👍 ✦ Good performance with strong locality, but…
  - 👎 ✦ …slow convergence
  - 👎 ✦ Poor performance with weak locality



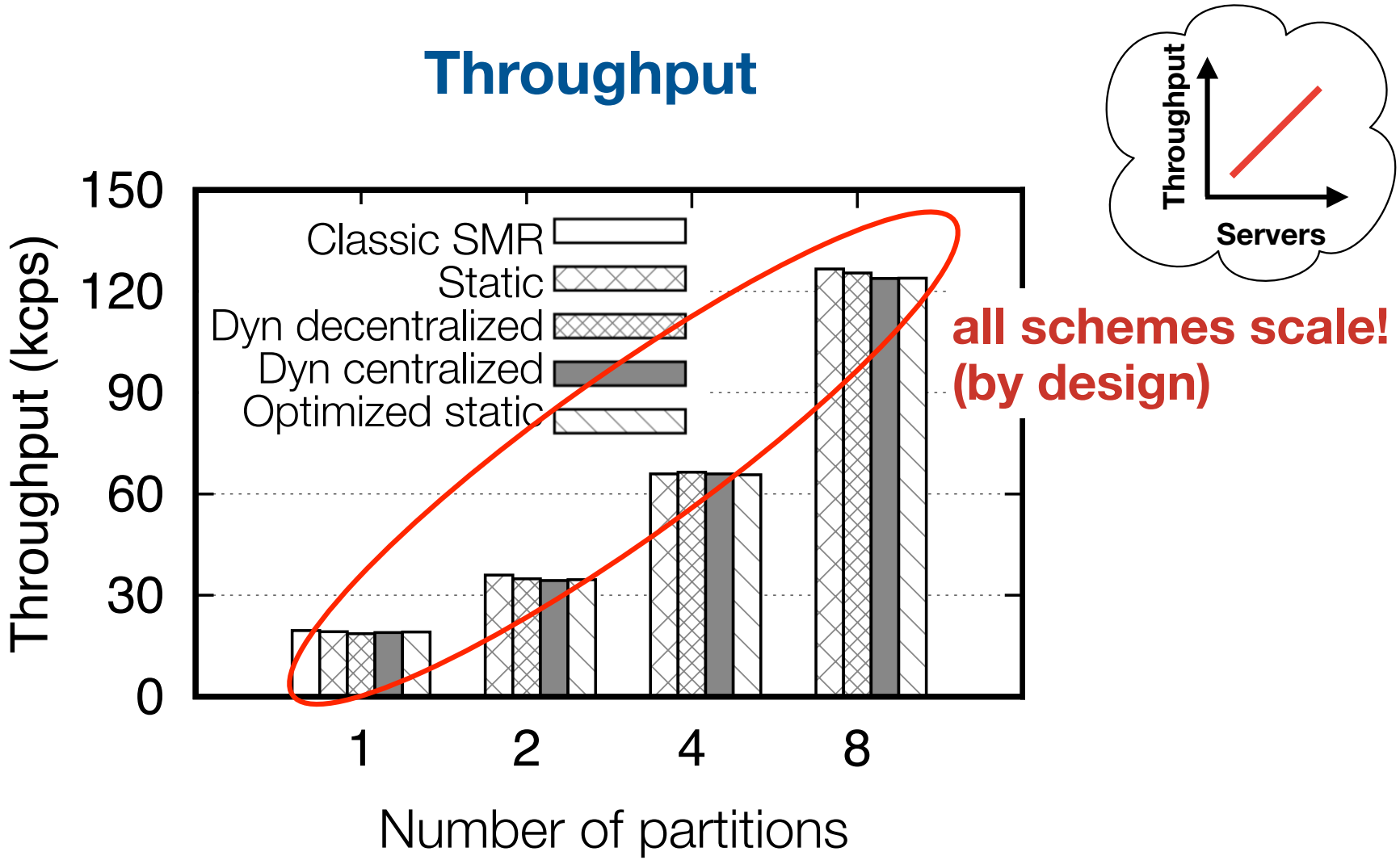
Dynamically (re-)partitioning the state

- Centralized strategy
  - ✦ Oracle builds graph of objects and relations (commands)
  - ✦ Oracle partitions O-R graph (METIS) and requests move operations to place all objects in one partition
- 👍 ✦ Near-optimum partitioning (both strong and weak locality)
- 👍 ✦ Fast convergence
- 👎 ✦ Oracle knows location of and relations among objects
- 👎 ✦ Oracle solves a hard problem

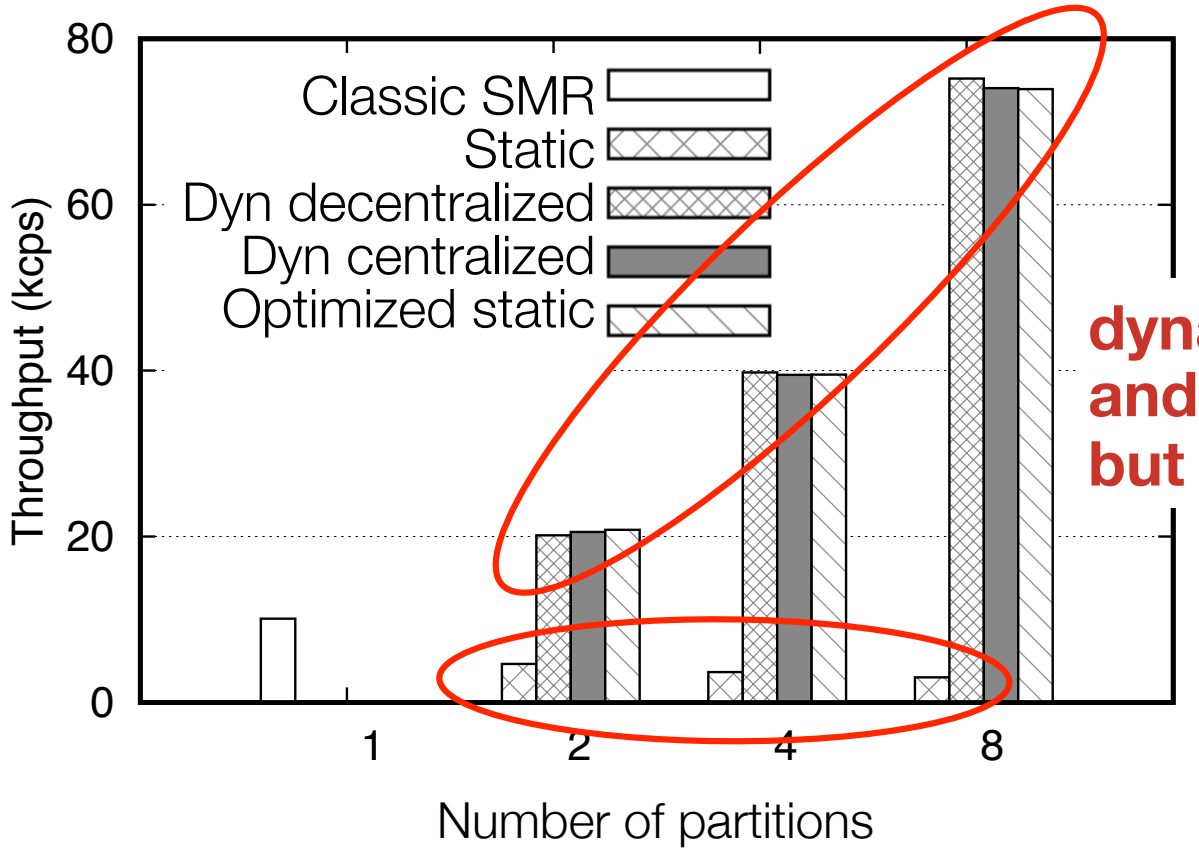
Social network application (similar to Twitter)

- GetTimeline
  - ✦ Single-object command => always involves one partition
- Post
  - ✦ Multi-object command => may involve multiple partitions
  - ✦ Strong locality
    - 0% edge cut, social graph can be perfectly partitioned
  - ✦ Weak locality
    - 1% and 5% of edge cuts, after partitioning social graph

GetTimelines only (single-partition commands)

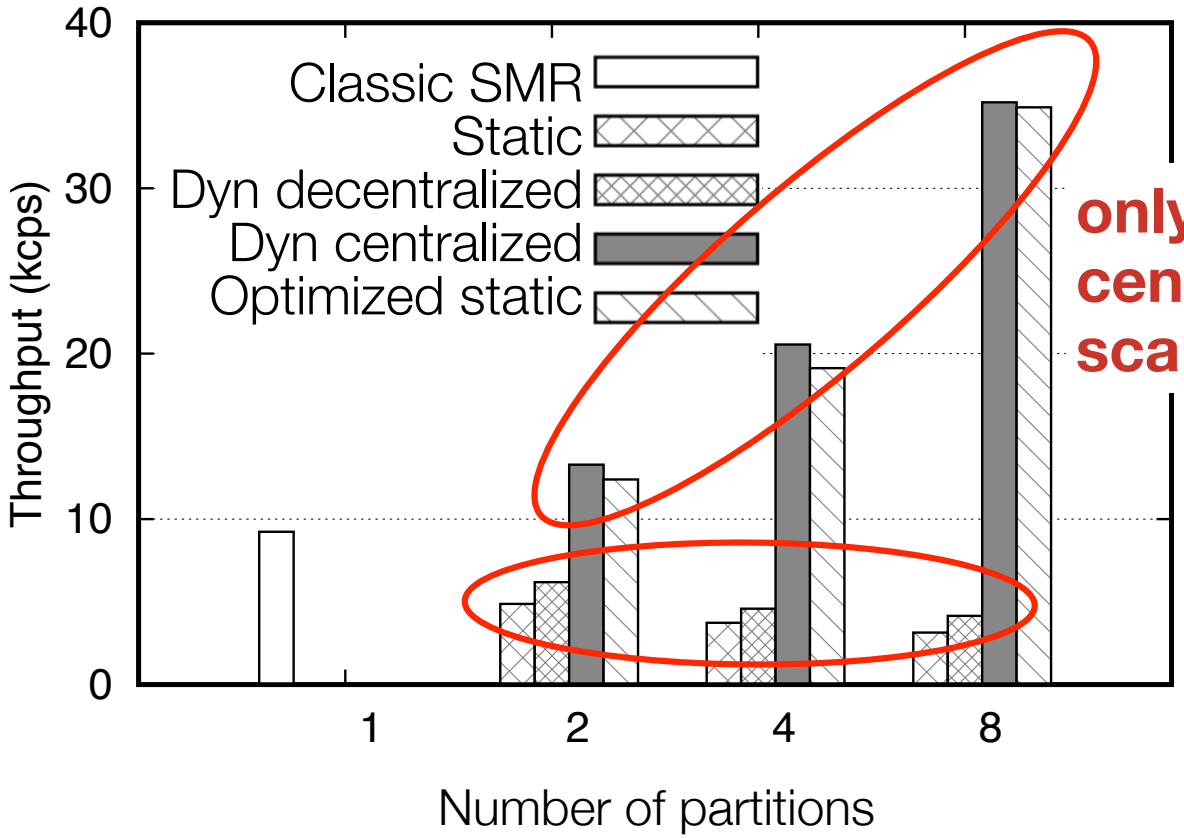


Posts only, strong locality (0% edge cut)



dynamic schemes and optimized scale, but not static

Posts only, weak locality (1% edge cut)



only optimized and centralized dynamic scale

Conclusions

- Scaling State Machine Replication
  - ✦ Possible but locality is fundamental
    - OSs and DBs have known this for years
  - ✦ Replication and partitioning transparency
- The future ahead
  - ✦ Decentralized schemes with quality of centralized schemes
  - ✦ Expand scope of applications (e.g., data structures)
  - ✦ “The inherent limits of scalable state machine replication”

More details:
http://www.inf.usi.ch/faculty/pedone/scalesmr.html

THANK YOU!!!

Joint work with…
Long Hoang Le
Enrique Fynn
Eduardo Bezerra
Robbert van Renesse