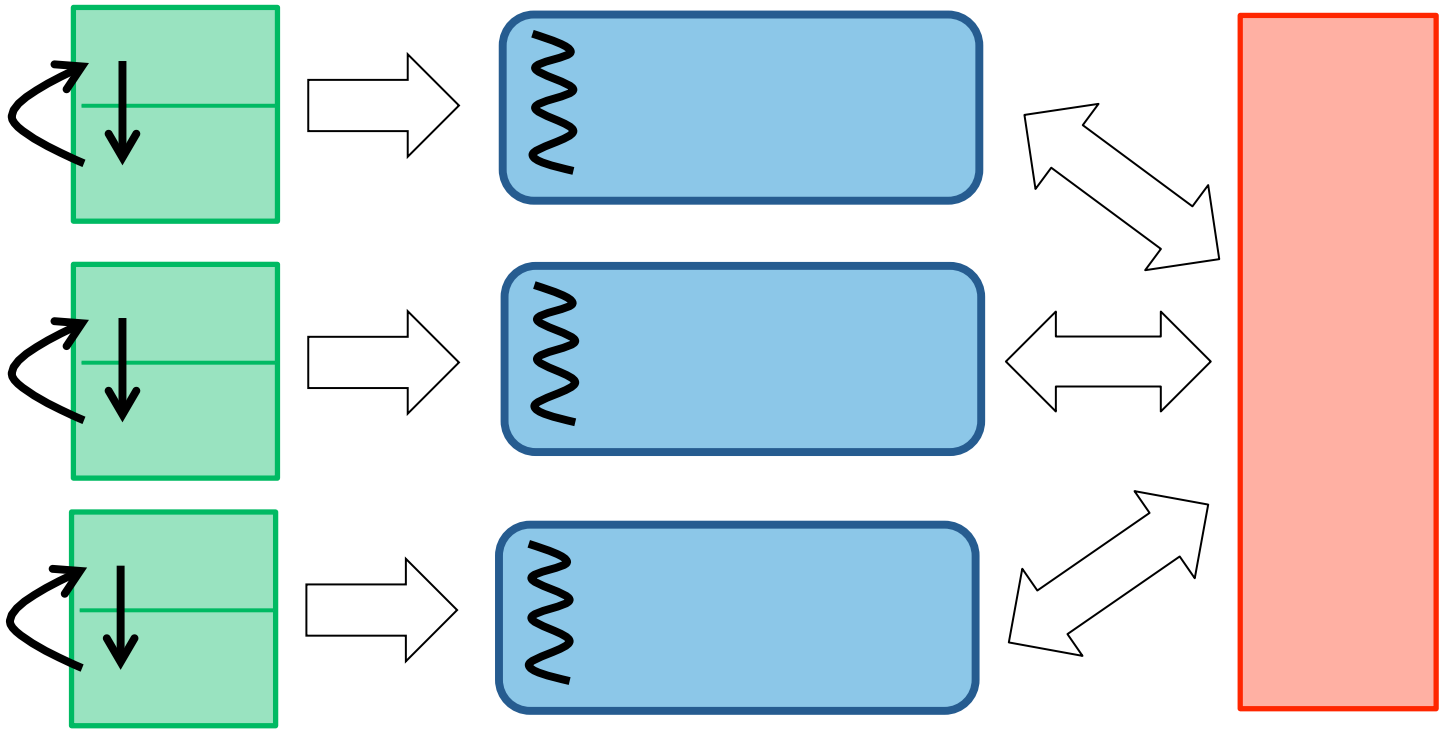
Iterativeness-Aware Optimization for Big Data Analytics

Henggang Cui, Lianghong Xu

15799b Course Project Final Presentation

Big Data Analytics

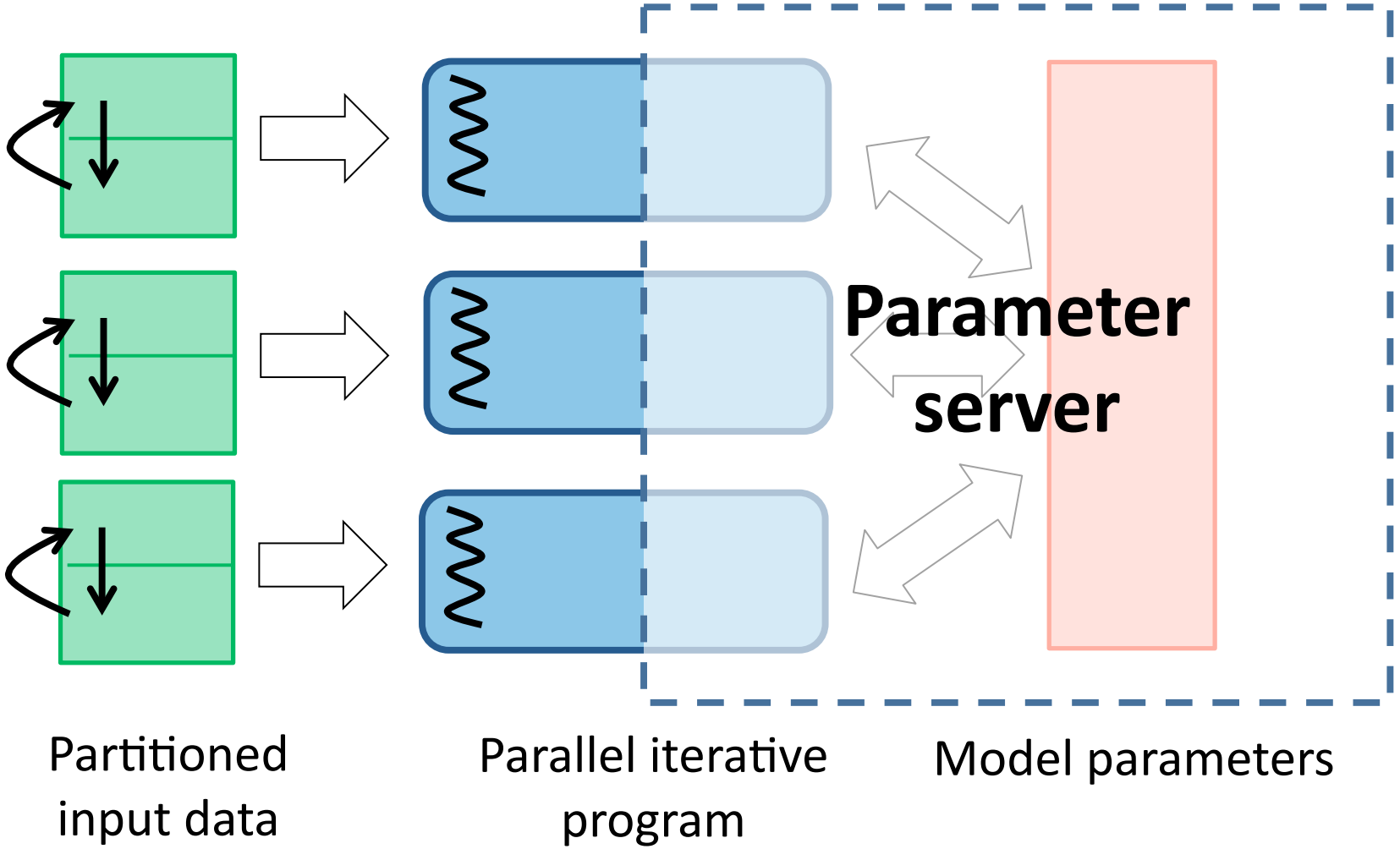


Partitioned input data

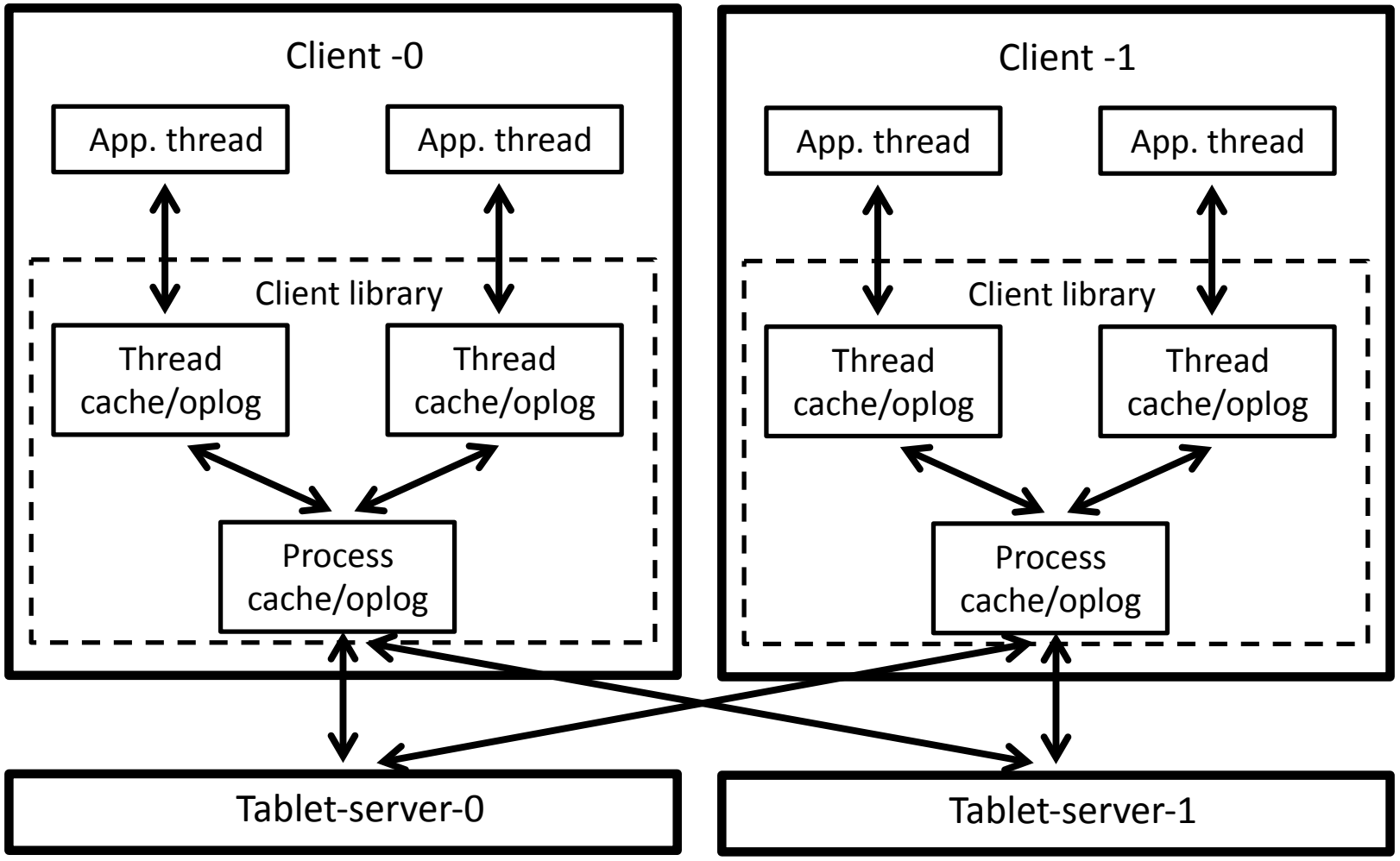
Parallel iterative program

Model parameters

Big Data Analytics



A Typical Parameter Server



Application is Iterative

- The application is often iterative
  - iterates over the input data
  - applies the same sequence of operations
  - reads/updates the same set of parameter data

- Knows everything about the access patterns

Example: PageRank

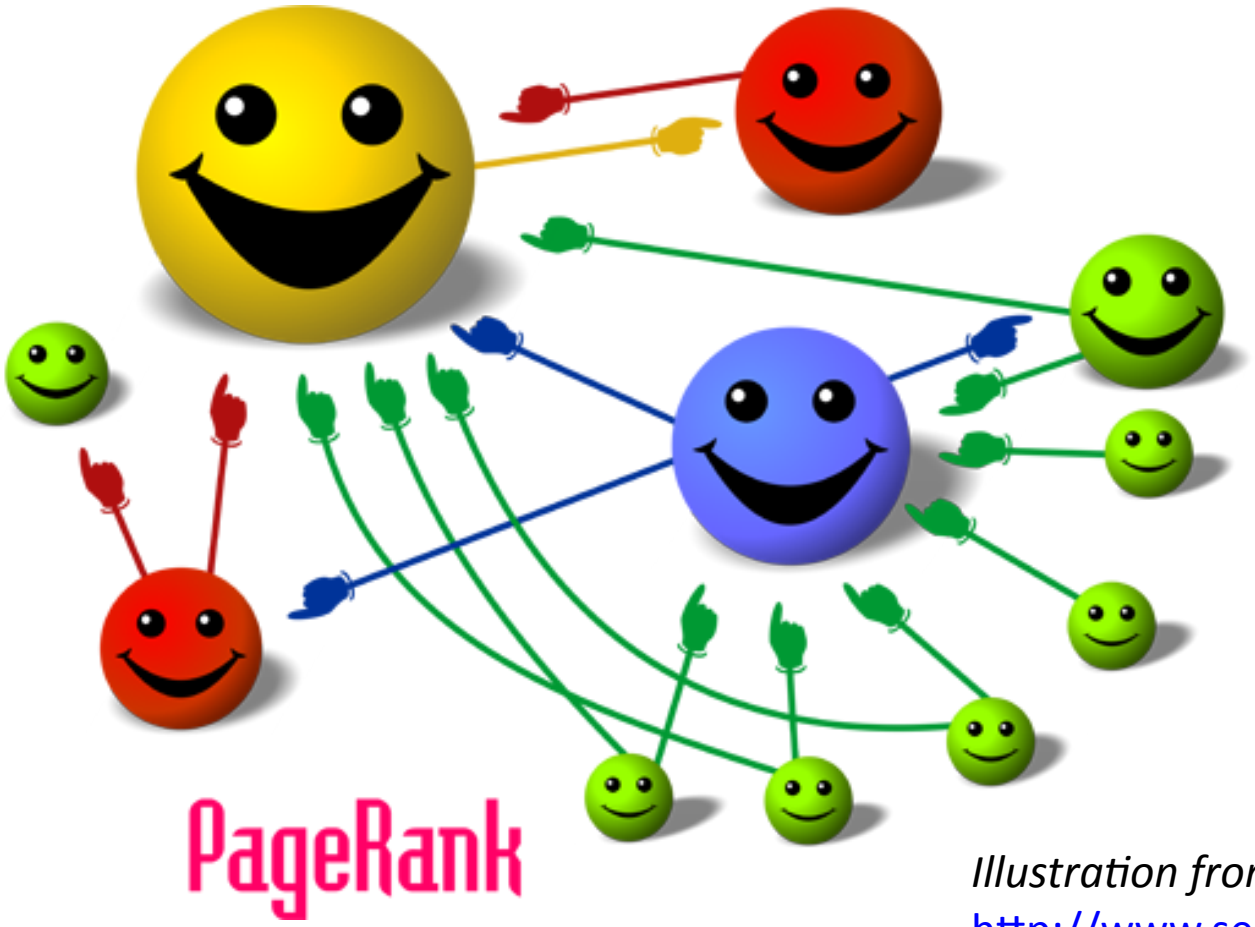


Illustration from
http://www.seoxp.net/?p=40

Example: Topic Modeling

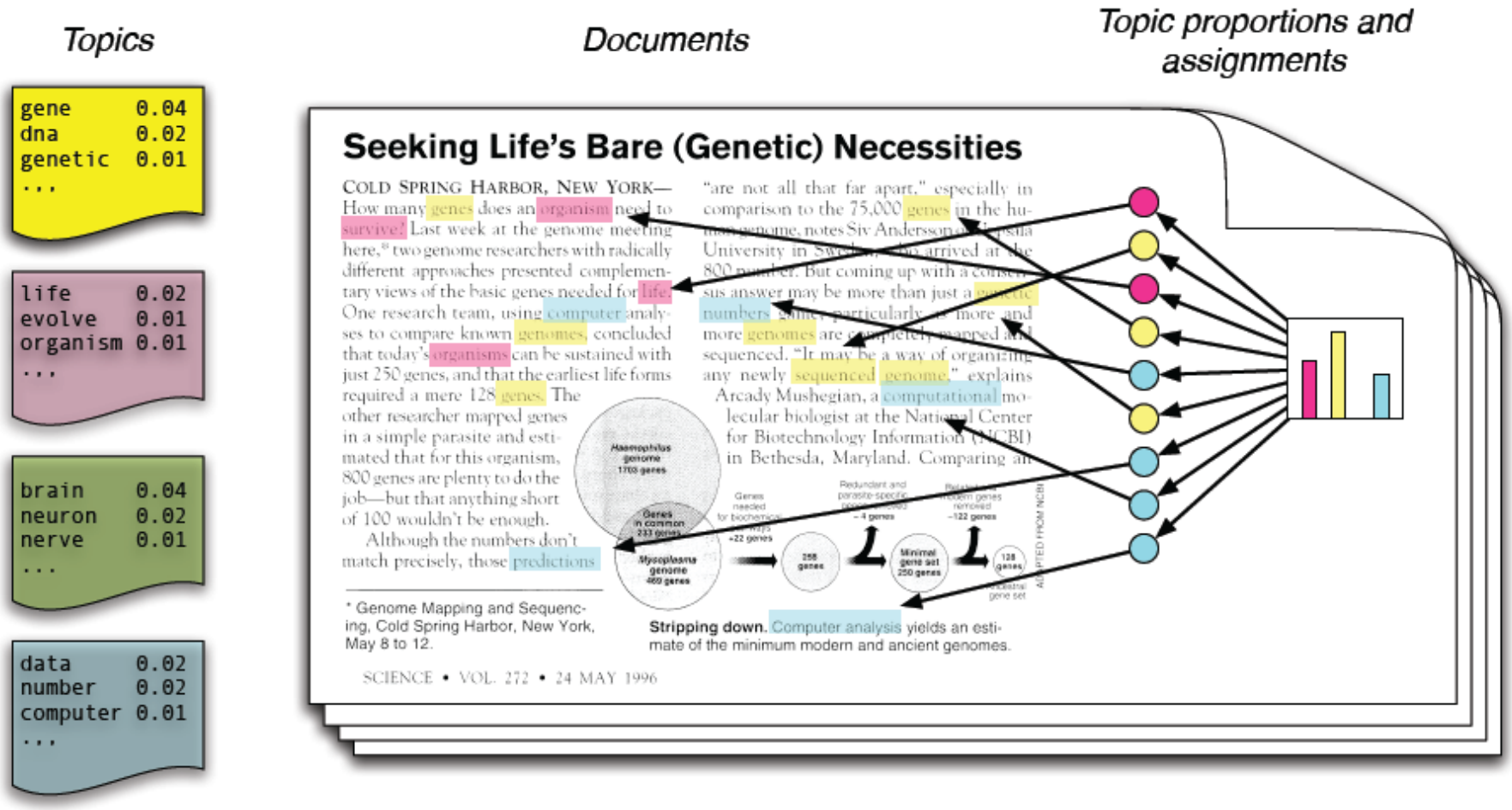
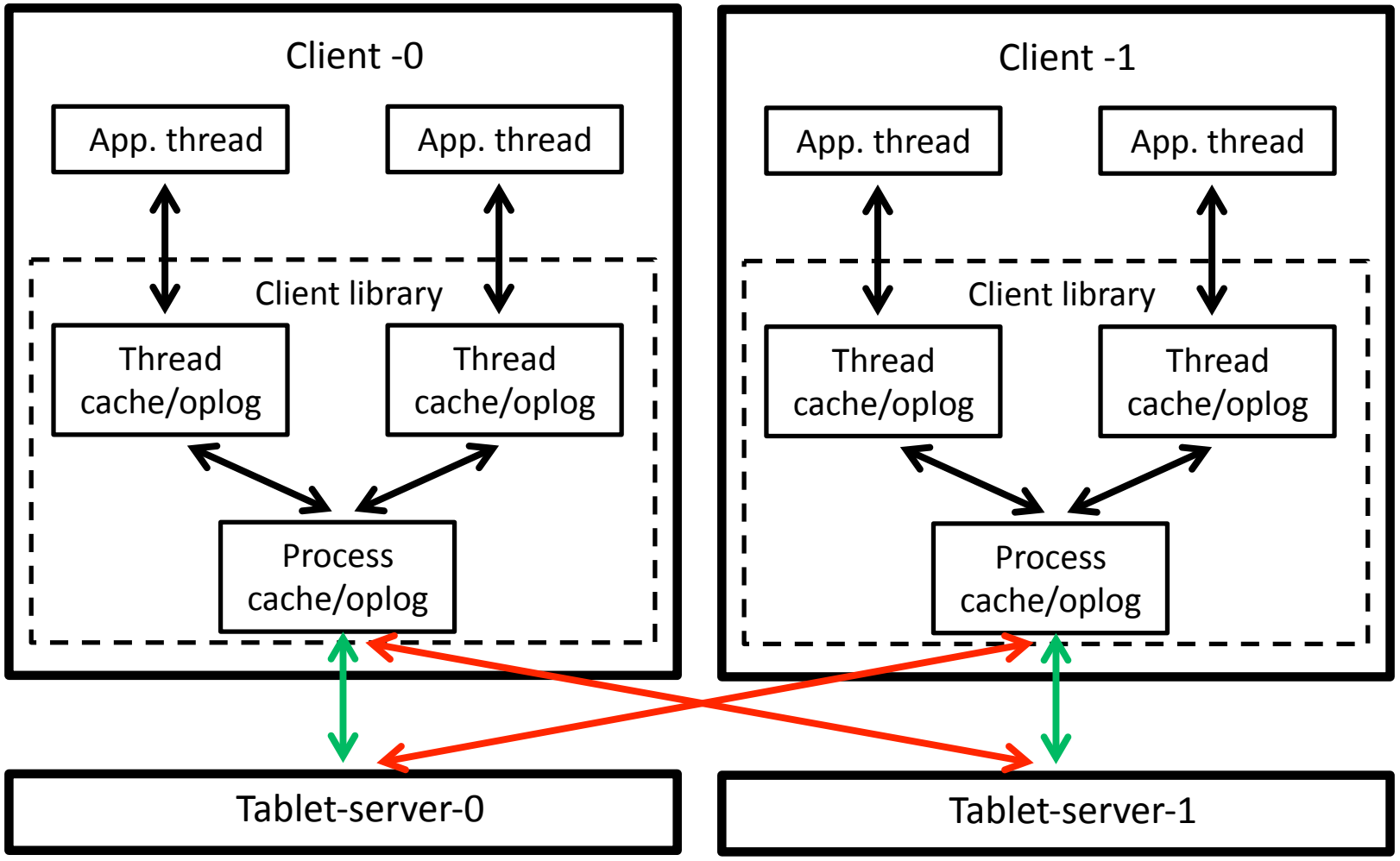


Illustration from Blei, D. 2012

Project Idea

- Exploit the iterative nature
  - much optimization opportunity w/ known access patterns
  - this project exploits one of them:
    parameter data partitioning

Parameter Data Partitioning



Parameter Data Partitioning

- Intuition: minimize remote access
  - keeps the parameter data in the machine that accesses it with the highest frequency
- More considerations: load balancing
  - no server should have too many requests to service
  - no client should be delayed too much

Gathering Access Patterns

- A “Virtual Iteration”
  - has the application report the operation sequence
    - read, write, clock (sync point)

- Virtual iteration has almost no overhead
  - not any computation or data access involved!
  - worthwhile to do it before the real iterations

Example Info from a Virtual Iteration

Thread-0 on Machine-0	Thread-1 on Machine-1
READ ROW 0	READ ROW 1
WRITE ROW 0	WRITE ROW 1
CLOCK	CLOCK
READ ROW 2	READ ROW 3
WRITE ROW 2	WRITE ROW 3
CLOCK	CLOCK

Access Pattern to Tablet Servers

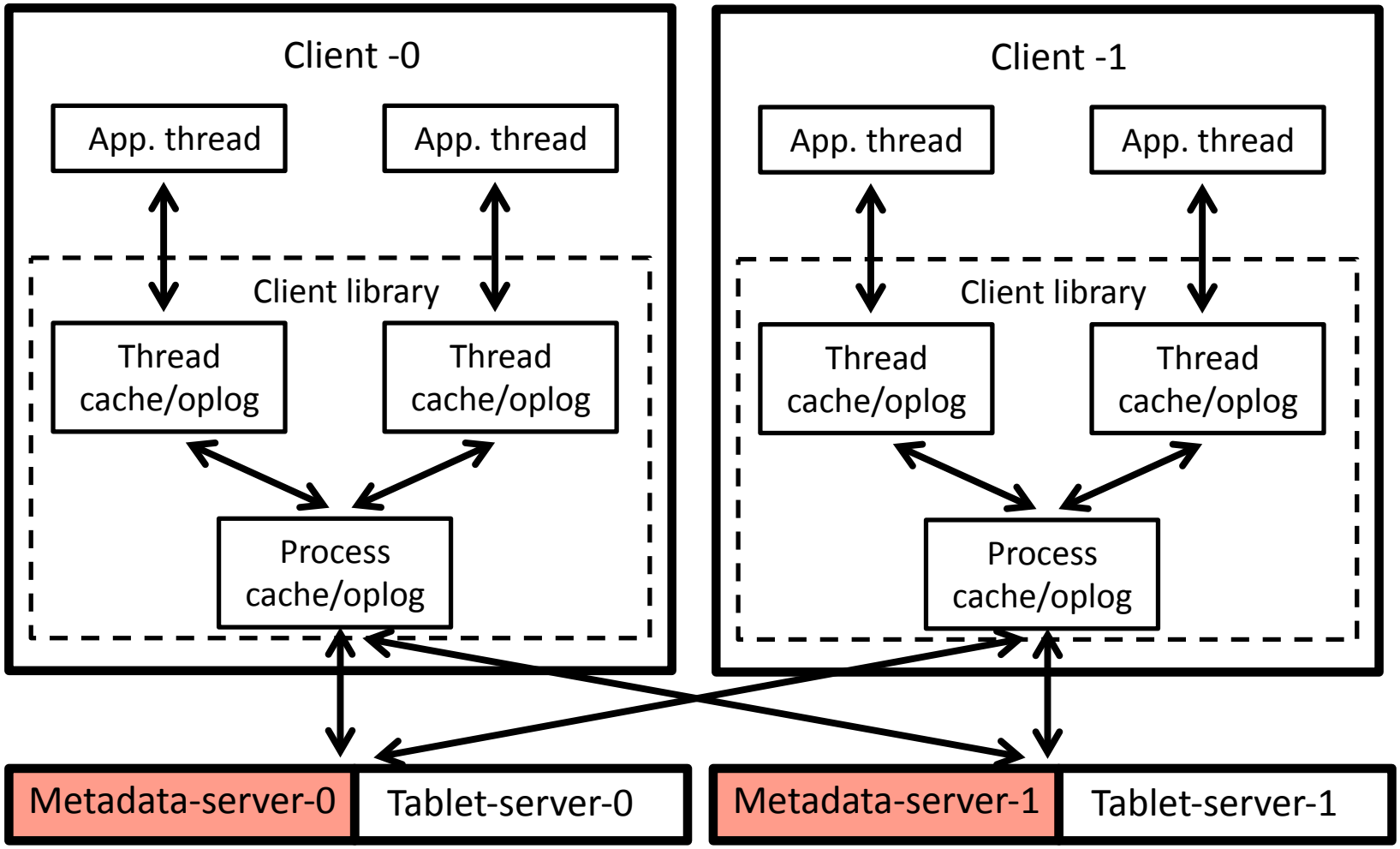
- The access pattern to servers is a little different
  - because of caching/batching
  - each client has only one read/write to a row per clock

Managing Parameter Data Location

- Manage location using a cluster of metadata servers
  - each manages the location of a subset of rows
  - client finds the right metadata server by hashing row ID

- Metadata servers also decides the mapping
  - clients send access info to them in the virtual iteration

Metadata Servers



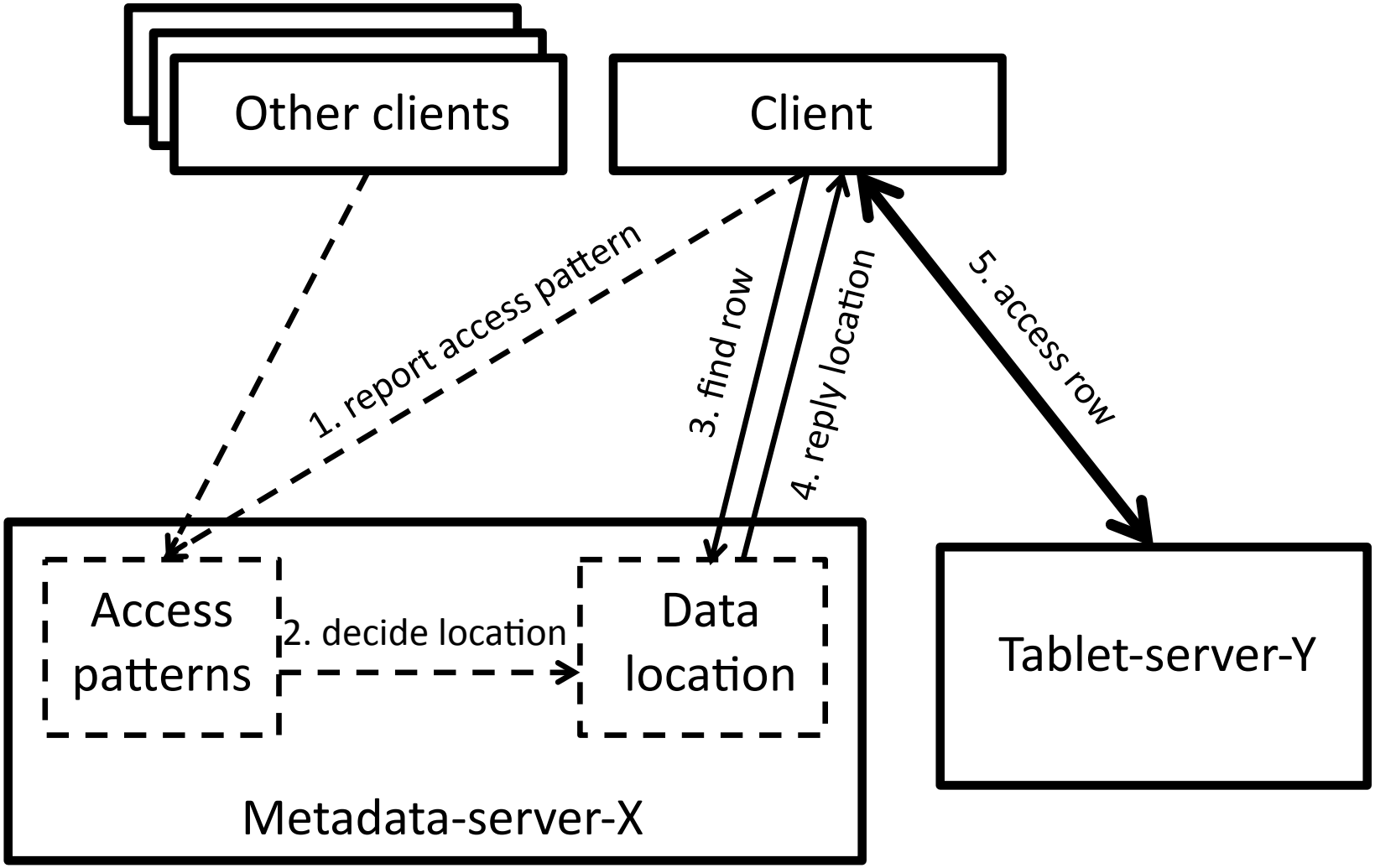
Reasons of Design Choices

- Why distributed instead of centralized?
  - not much more complexity
  - don't want one machine has more load than others
    - load balance issue
    - clients and servers are in the same set of machines
  - don't have any centralized components before
    - had better not to break this

Reasons of Design Choices

- Why metadata server also decides the mapping?
  - the servers never talk to each other
  - communication only happens between clients and servers

Detailed Workflow



Partition Policies

- Mapping: row_id -> server_id
- Policy 1: Random partitioning
- Policy 2: First accessing client
- Policy 3: Minimize remote access + load balancing

Random partitioning

- server_id = row_id % nr_tablets
- Pros: Load balancing
- Cons: Many non-local accesses

First Accessing Client

- server_id = first_accessing_client(row_id)
- Pros: At least 1 local access for any row
- Cons:
  - No explicit load balancing
  - Depending on arrival time of client requests

Min(non-local) + load balancing

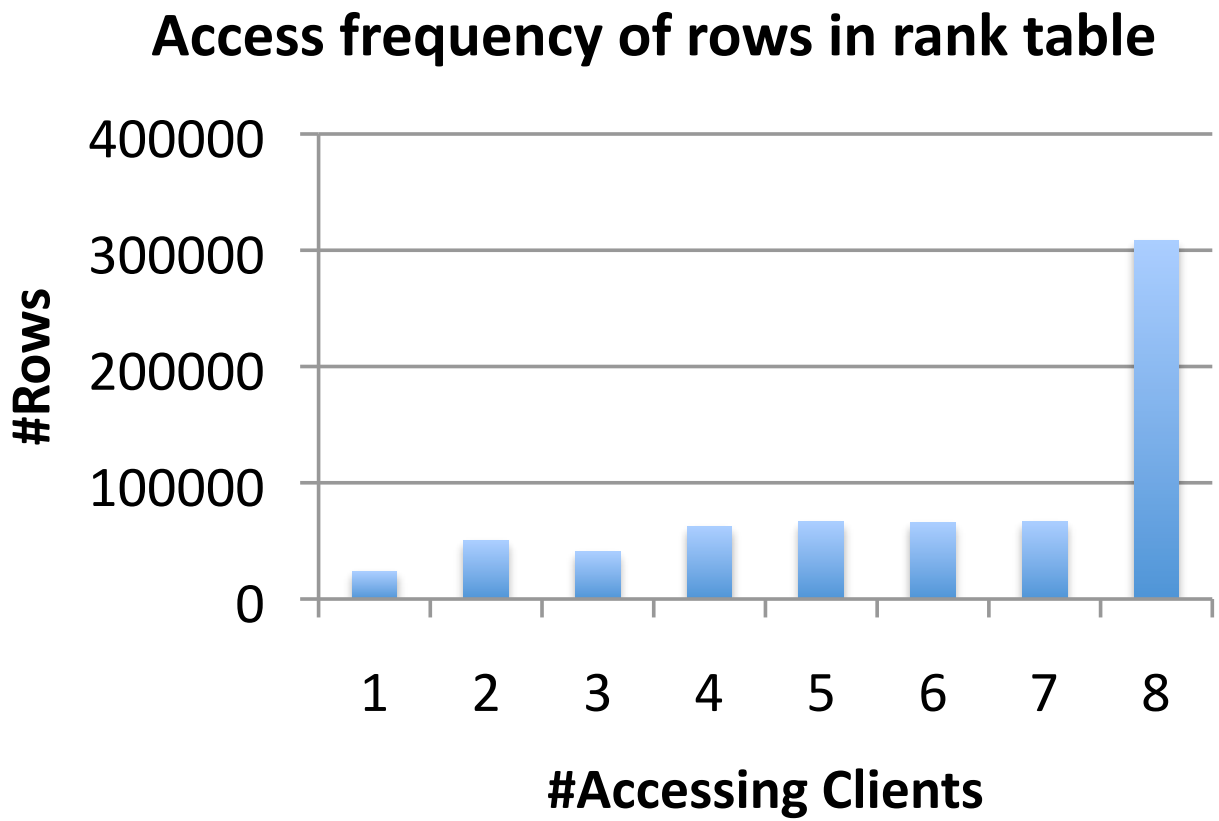
- An row is placed on the server co-located with the most-frequently accessing client
- When access frequency is equal, choose the one with the least load
- Load: #rows
  - Future plan: use #(non-local) accesses

Experiment Setup

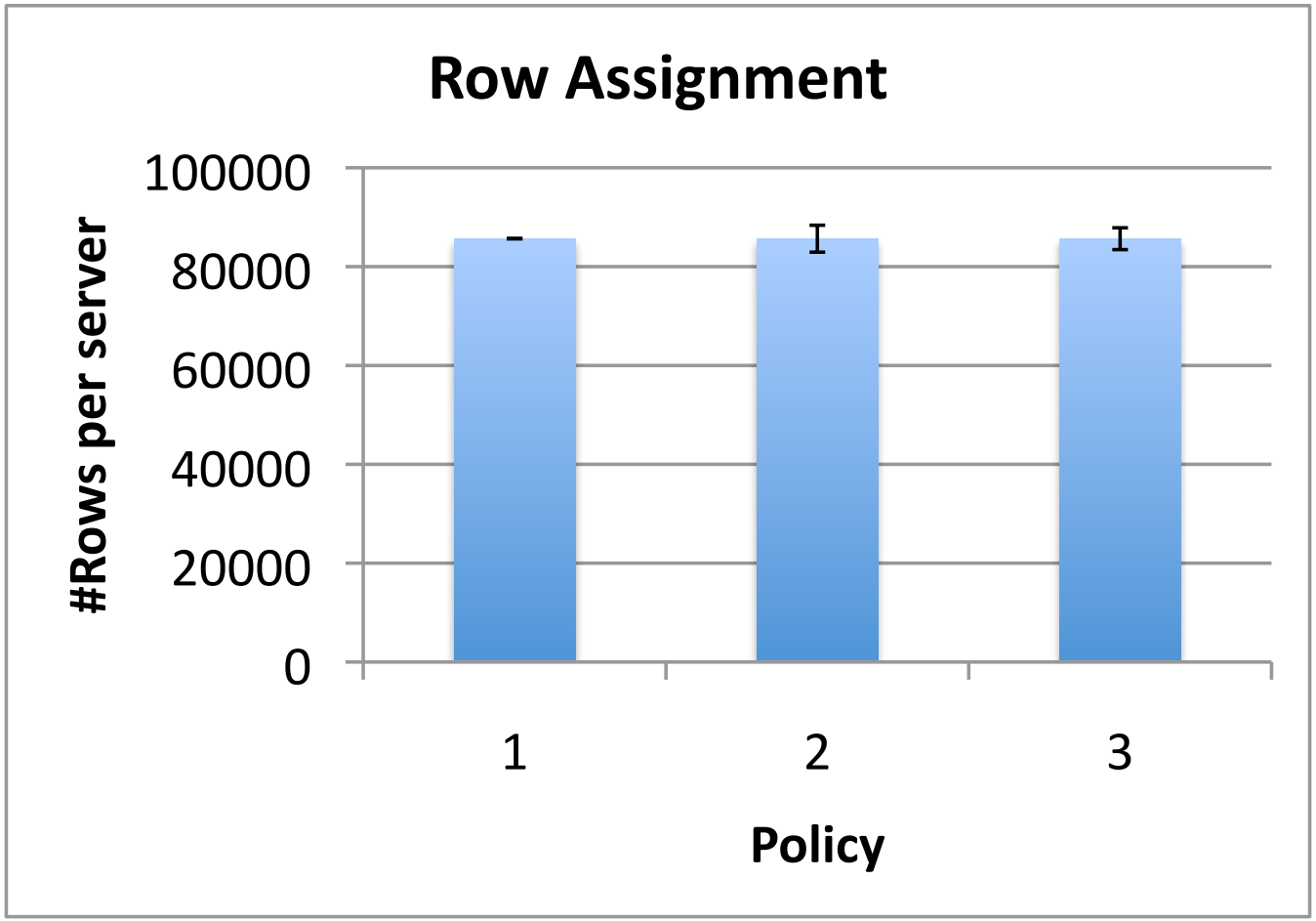
- 8 OpenCirrus machines
  - Each has 8 cores, 16GB memory
- Each machine runs a client, metadata server and tablet server
- Benchmarks:
  - PageRank (Berkstan dataset: 685k nodes and 7.6m edges)
  - Topic Modeling (ACL Anthology dataset: 15,032 docs)
  - Each benchmark is run for 3 times

PageRank Dataset

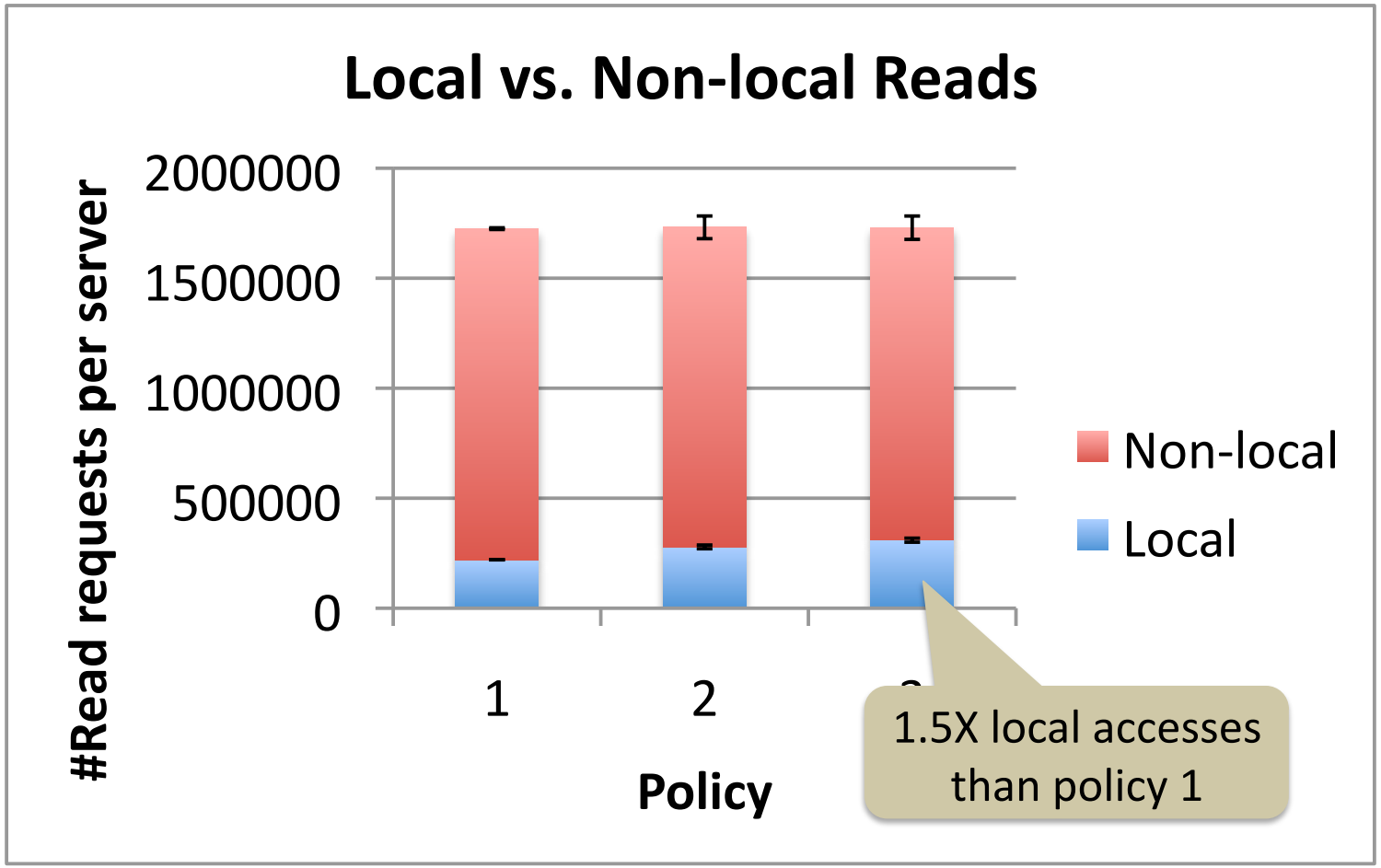
- 1 rank table (685229 rows)



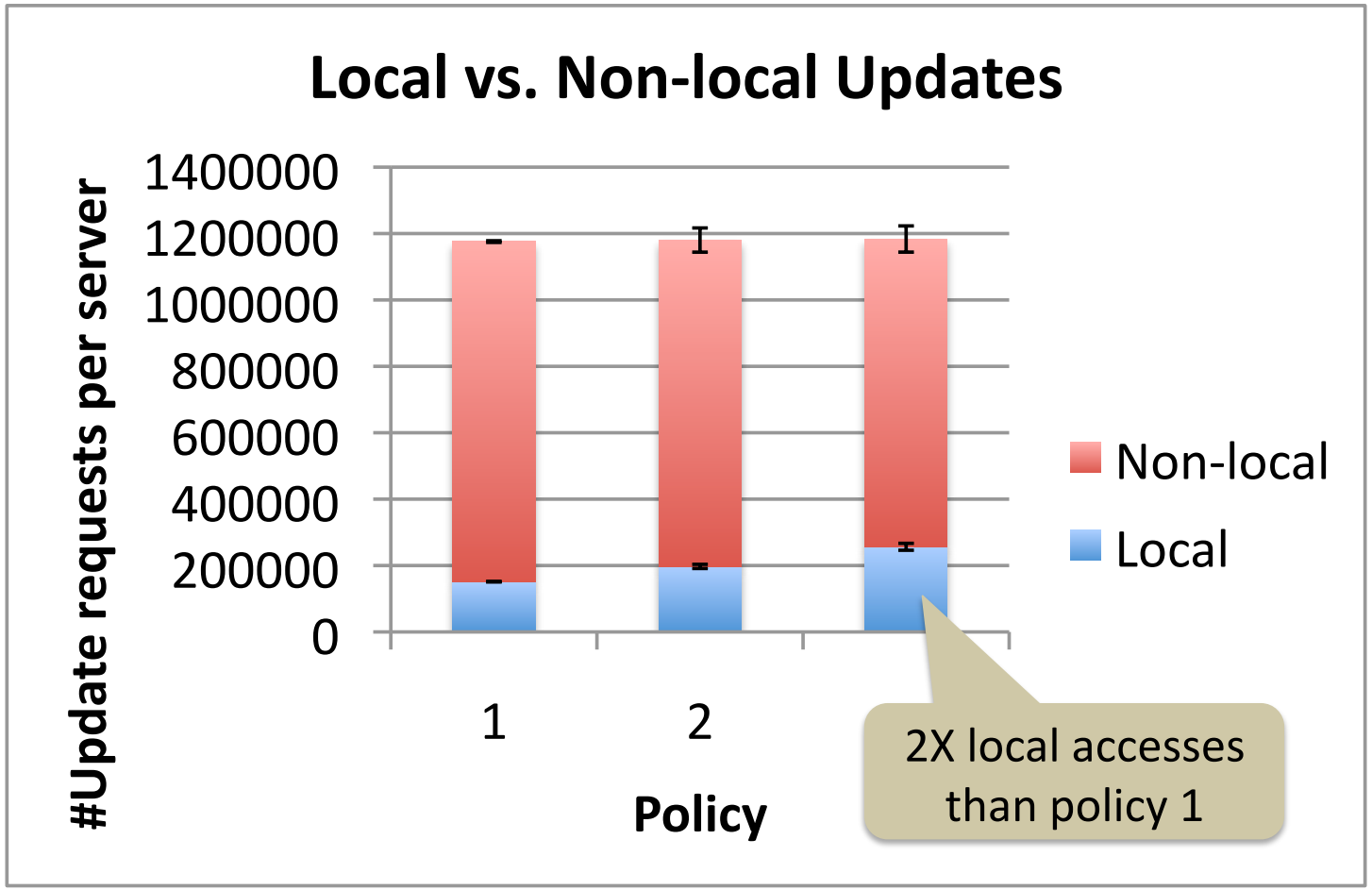
Experimental Results: PageRank



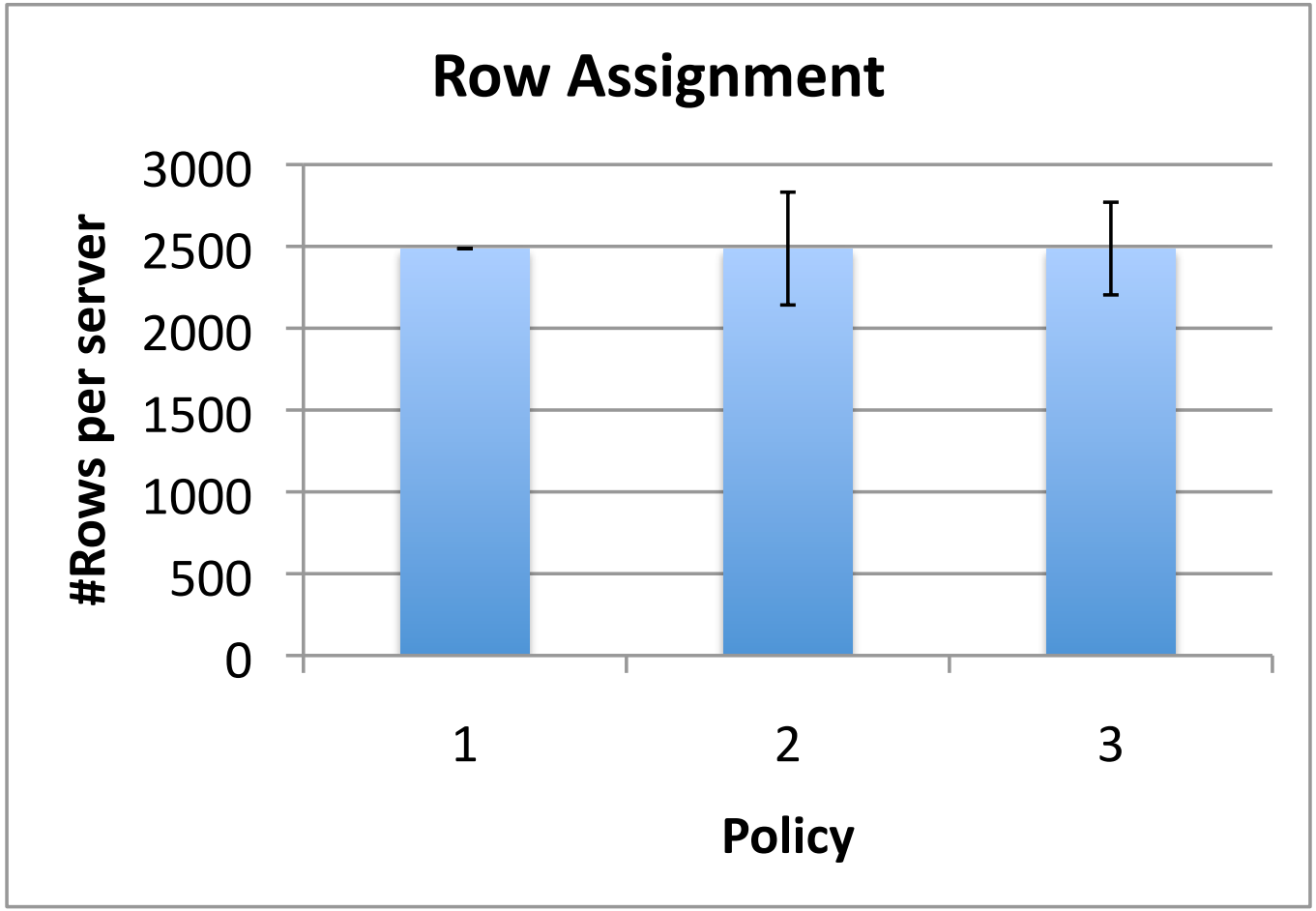
Experimental Results: PageRank



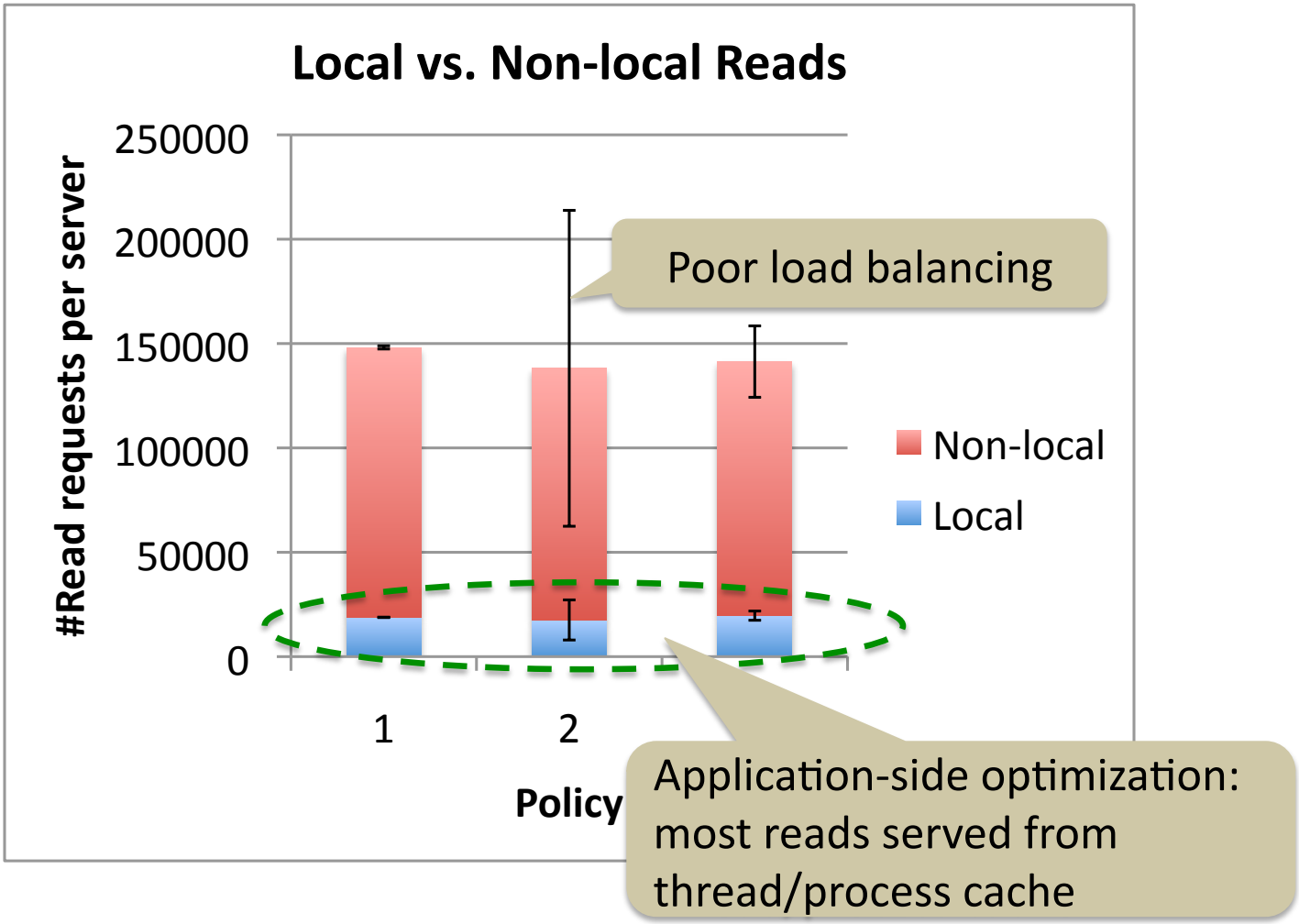
Experimental Results: PageRank



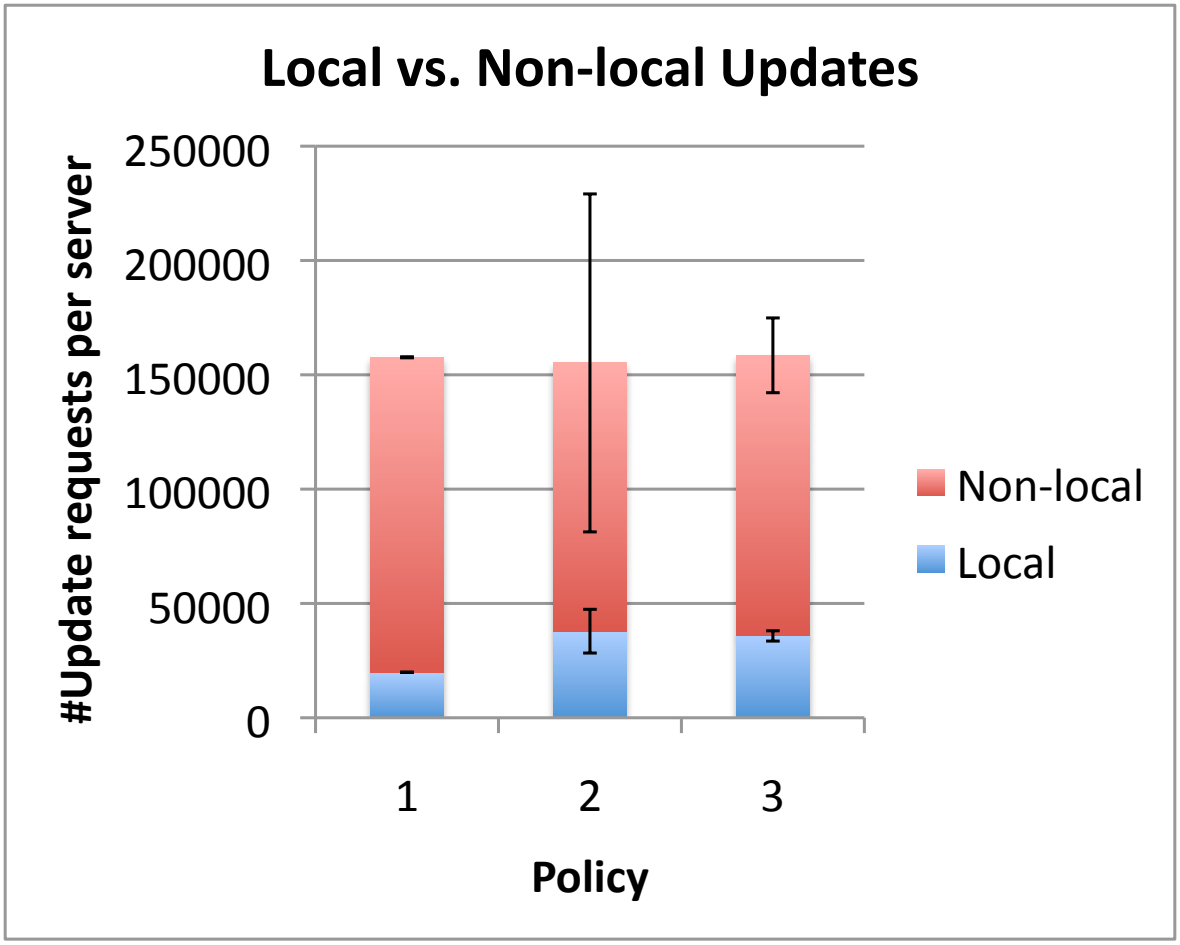
Experimental Results: Topic Modeling



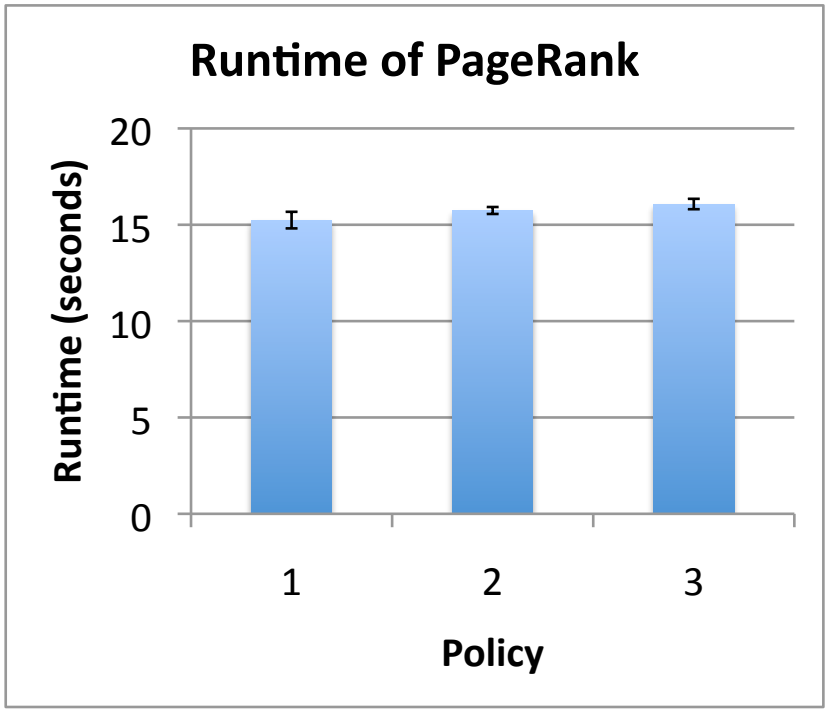
Experimental Results: Topic Modeling

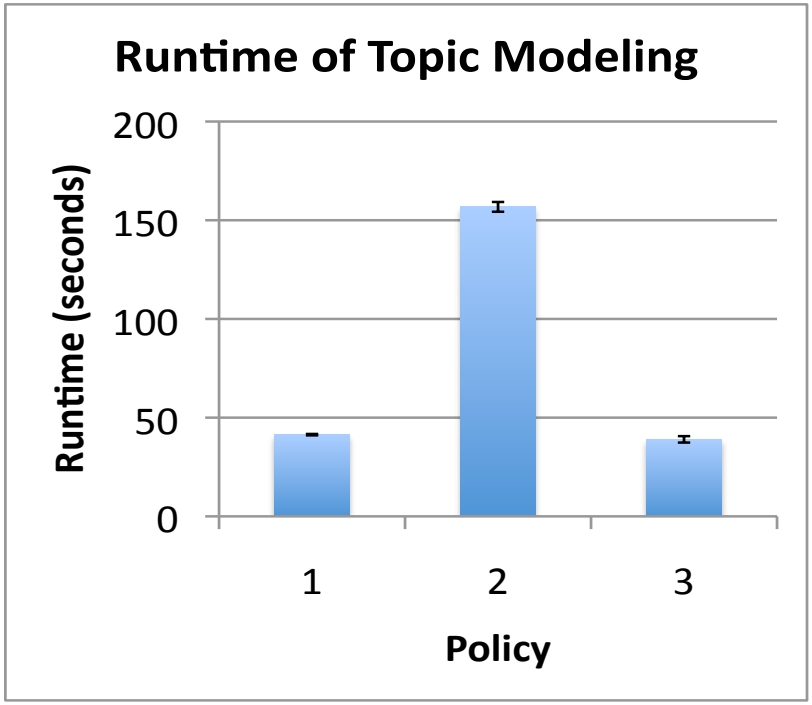


Experimental Results: Topic Modeling



Experimental Results: Runtime





Current implementation: Local access is as slow as remote access

Future Plans

- Look for more applications
- Optimize performance of local access
- Optimize implementation of policy 3
- Implement policy 4: using non-local requests as load metric

Topic Modeling Dataset

- 1 word table (101636 rows)
- 1 document table (299752 rows)