facebook

LinkBench

A database benchmark based on the Facebook social graph

Tim Armstrong*, Vamsi Ponnekanti †, Dhruba Borthakur †, Mark Callaghan†
*University of Chicago, †Database Engineering @ Facebook
June 25, 2013

Agenda

1	Why another benchmark?
2	Generating a synthetic social graph
3	Generating a realistic query workload
4	Using LinkBench

Why another benchmark?

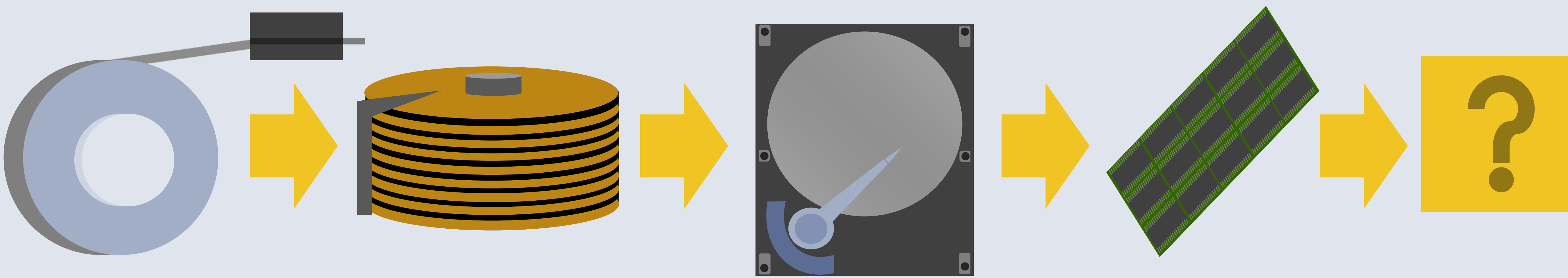
Database Engineering @ Facebook

- Core open-source technologies:
  - MySQL
  - RocksDB for embedded use (fork of LevelDB + ideas from HBase)
- Team goals:
  - Improve, extend and maintain database systems
  - Develop tooling around database systems
  - *Explore and evaluate alternative database systems*

A changing landscape

Hardware and software

- Hardware innovations: challenges and opportunities
  - Solid state disks: IOPS less of a bottleneck, capacity more so
  - Many core: high concurrency unavoidable
- Many new competing database systems and paradigms
  - NoSQL, NewSQL, graph databases

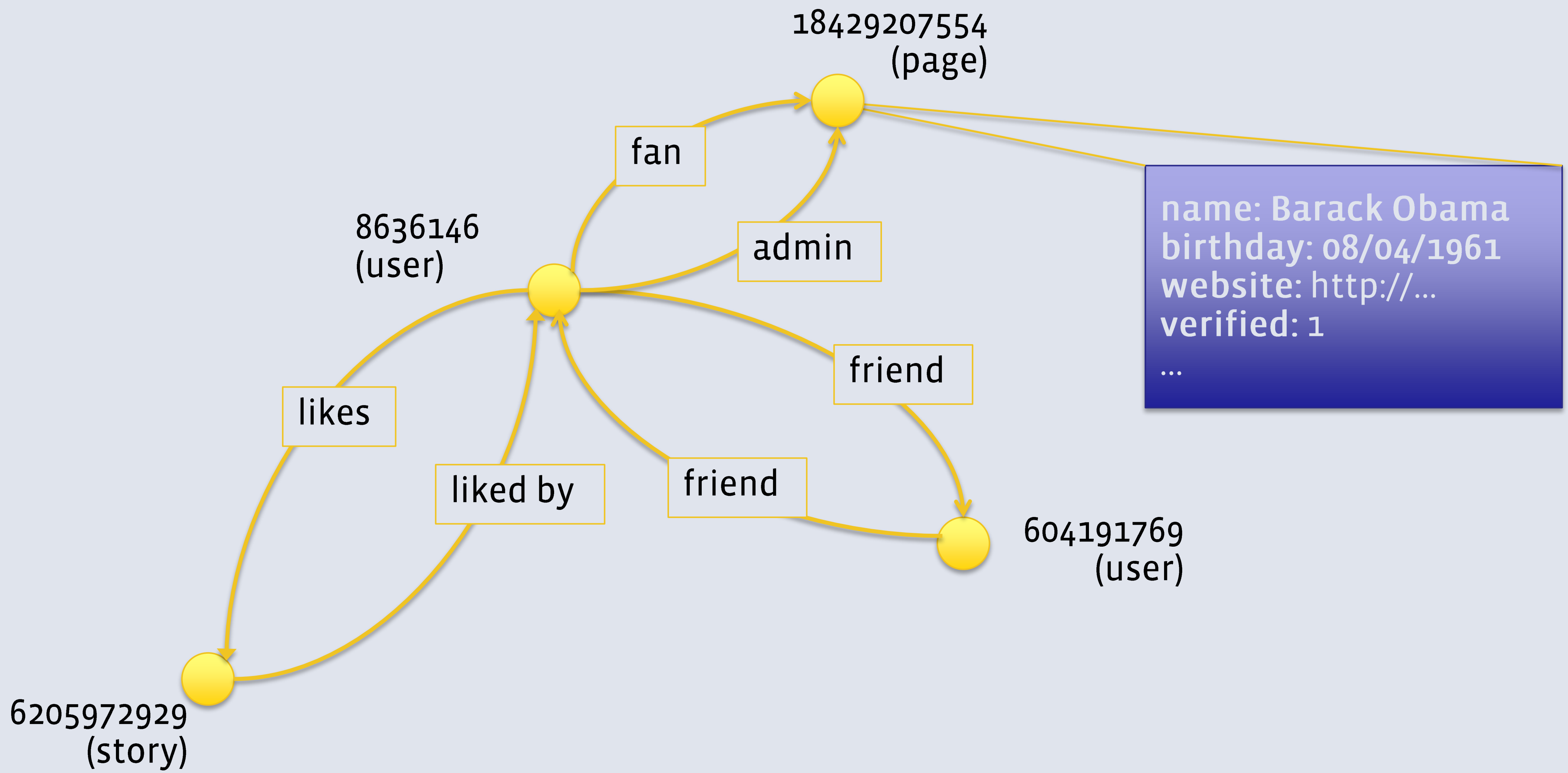


Large-scale social networks

A major application class

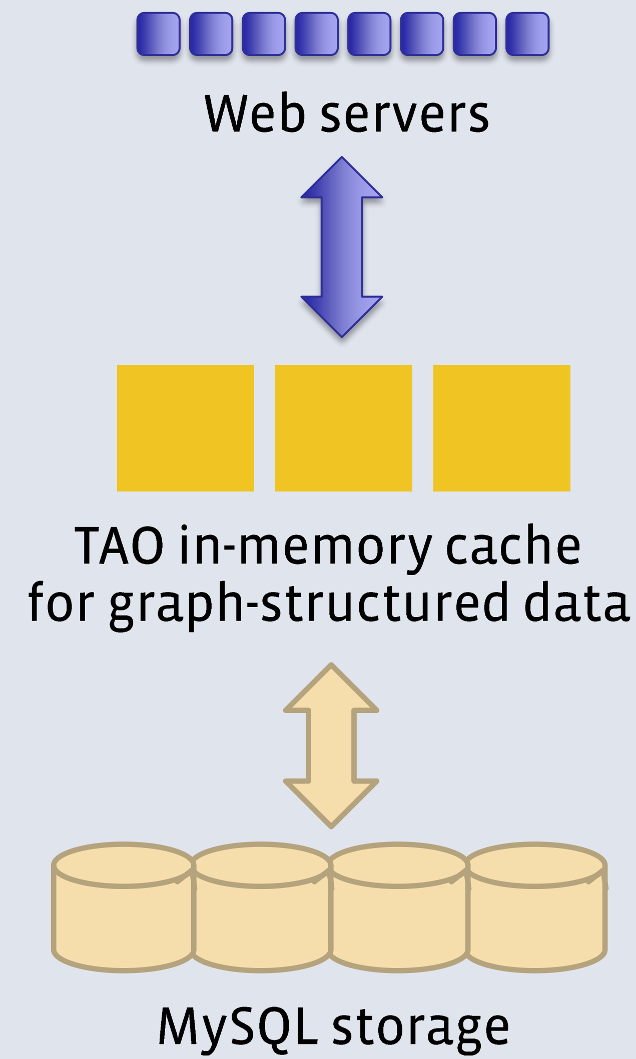
Social graph data model

Graph structured, highly interconnected data



Social graph serving architecture

- Core component of Facebook infrastructure
- Suitable for low-latency serving of large data sets
- MySQL for persistent storage
- Efficient, in-memory cache clusters for hot data
- We focus on persistent storage



Example of cache + database architecture for Social Graph

Motivation for LinkBench

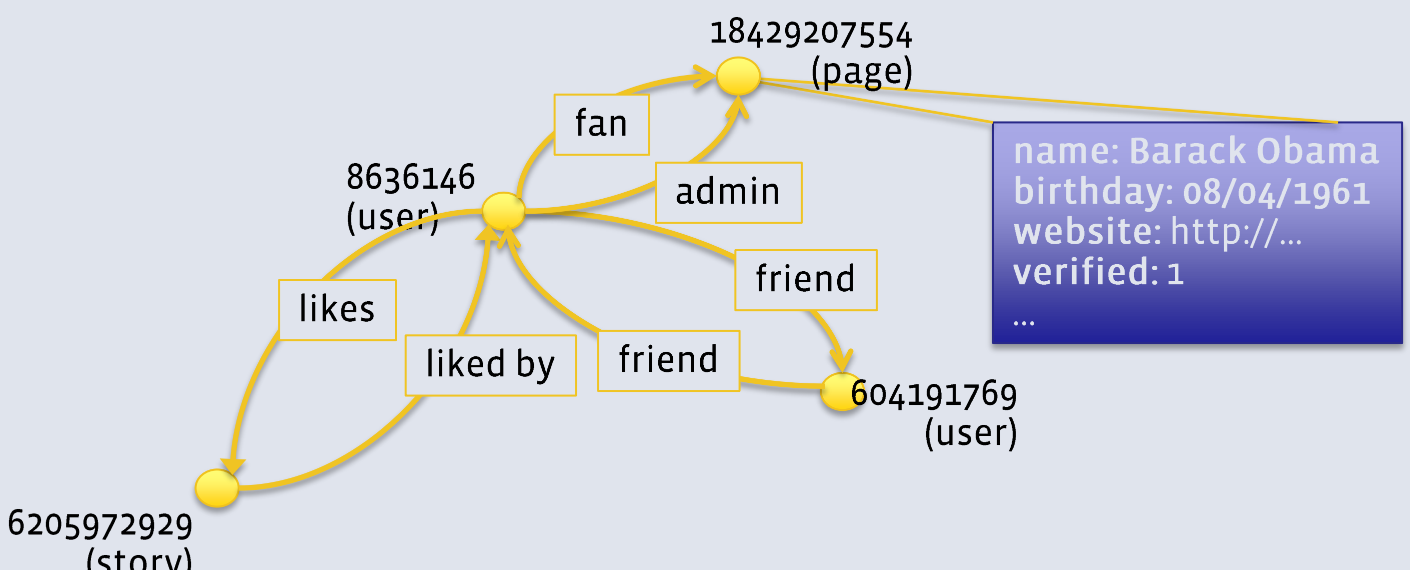
- Inside Facebook: running realistic benchmarks made simpler
  - Simple micro-benchmarks insufficient
  - Mirroring full production workload extremely labor intensive
- Outside of Facebook:
  - Compare systems for social application serving
  - No privacy issues (unlike workload traces)

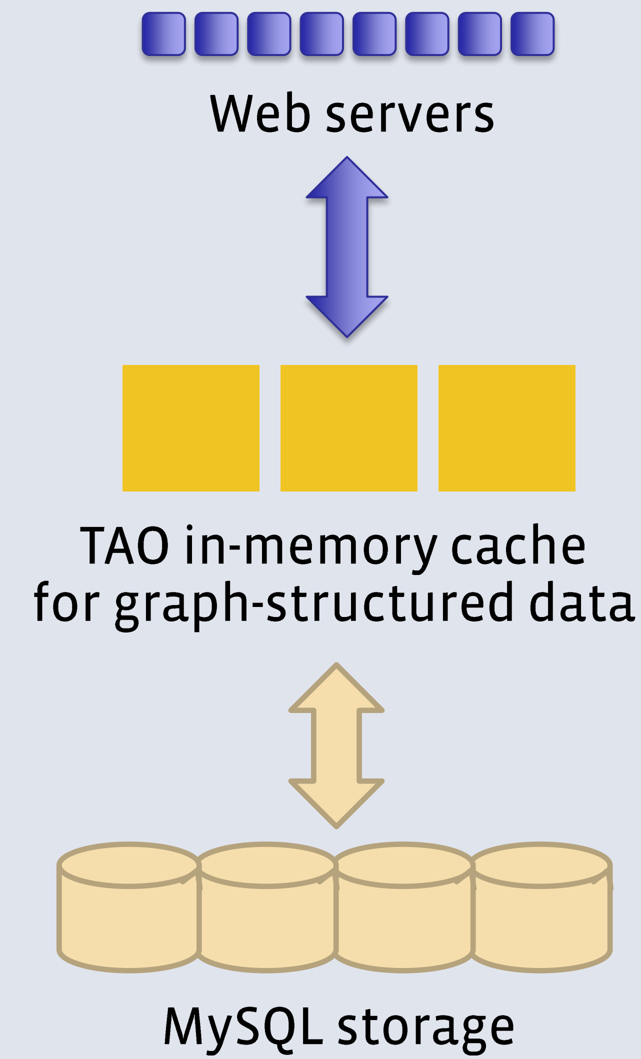
Existing benchmarking tools

- Transaction-processing, e.g. TPC-C:
  - Business-oriented schemas and workloads
  - Exercises transaction handling heavily
- Key-value web serving: (e.g. YCSB):
  - Related application space
  - Simple data models

LinkBench social graph workload

- Richer data model than key-value
- Simple, short-running queries
- Limited ACID properties required
- Based closely on analysis of production workload





Example of cache + database architecture at Facebook

Generating a synthetic social graph

Graph generation goals

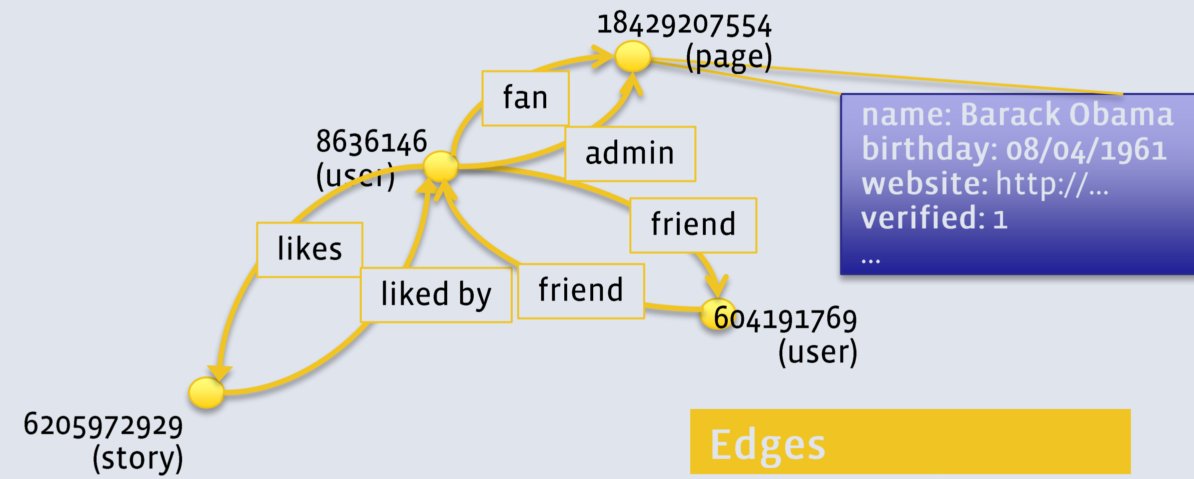
When is a synthetic social graph “realistic enough”?

- Synthetic social graph must be realistic in key dimensions that affect performance:
  - Data model and schema
  - Result-set size
  - Storage/compression efficiency

Mapping Social Graph to Relational Model

Implementation in MySQL

- Node and edge tables
- Edge count table for efficient count queries
- Partitioned:
  - between servers by node id (source id for edges)
  - between tables by type
- LinkBench uses identical data model



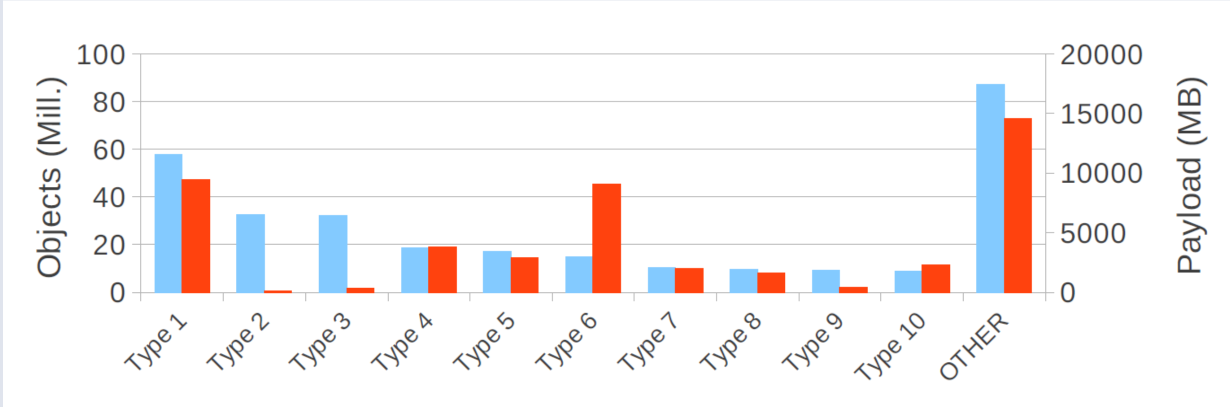
Nodes	
id	int64
type	int32
version	int64
update_time	int32
data	text

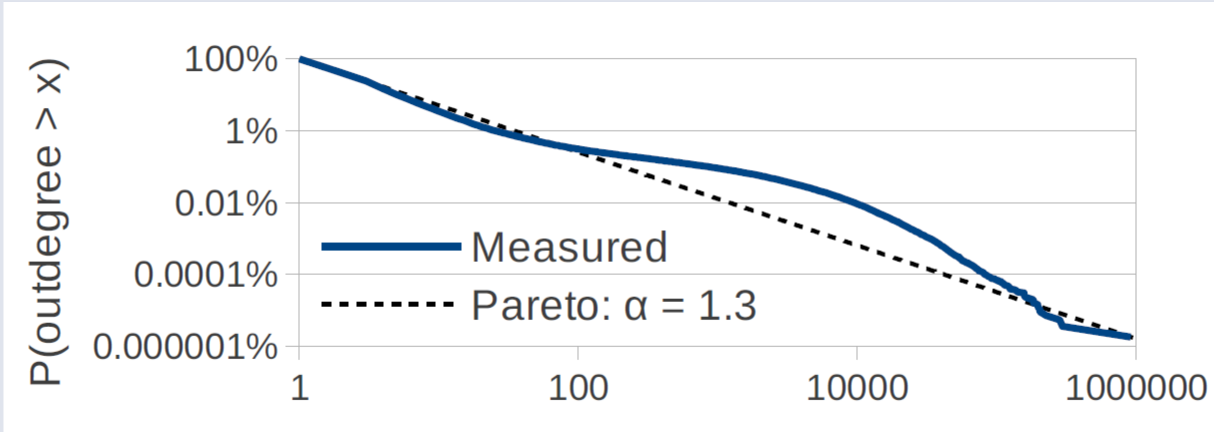
Edges	
id1, id2	int64
type	int64
visibility	int8
timestamp	int64
version	int32
data	varchar

Edge Counts	
id	int64
type	int32
count	int32
timestamp	int64
version	int32

Analysis of social graph structure

- Many, many edge and node types
- Power-law distribution of node outdegree
  - Previously observed in friendship networks
  - Also occurs in general social graph with other node types
- Empirical outdegree distribution used directly in LinkBench





Node and edge payload data

Compressibility matters

- Solid state drives: capacity is scarce
- Huge variability in compression ratio between database systems
- LinkBench data generators:
  - Motif data generator: random payload data with repeated motifs
  - Default parameters tuned to match real-world compression ratio

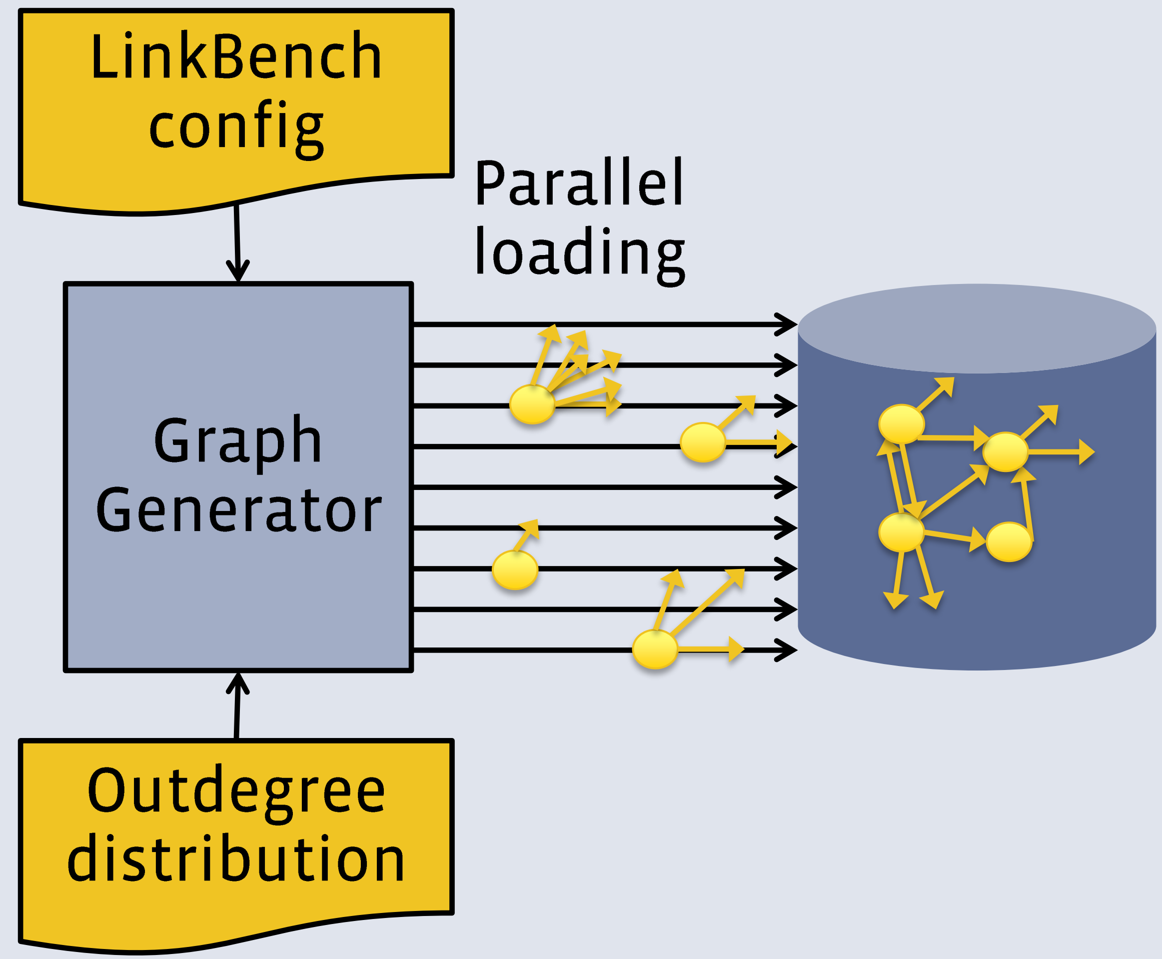
Size after bzip2 compression	
Nodes in database	61%
Edges in database	31%

Compressibility of social graph payload data

Graph generation in LinkBench

Configurable, extensible social graph generator

- Graph can be scaled up and down (typical benchmark: 1B nodes)
- Default degree distribution matches real social graph
- Community structure not emulated: little effect on single-hop query performance



Generating a realistic query workload

Query generation goals

When is a synthetic social graph “realistic enough”?

- Query workload must exercise database system in similar way
  - Query mix
  - Patterns of node/edge access
  - Result set sizes

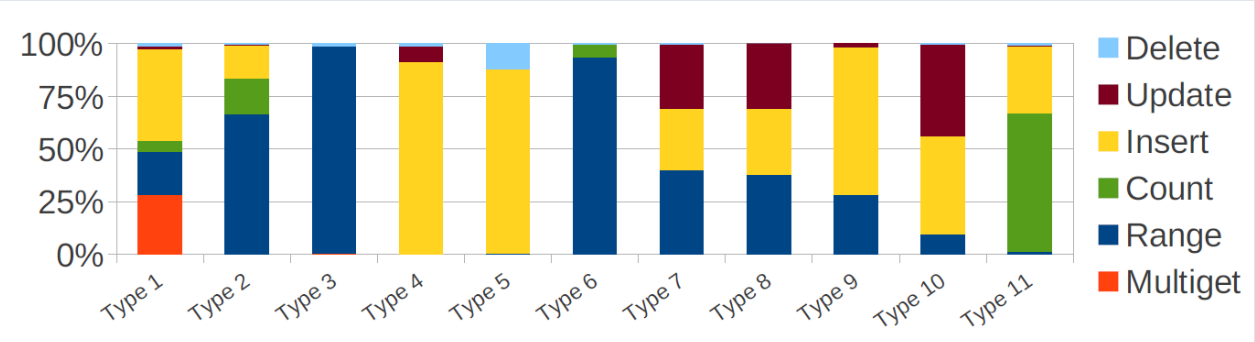
Production query trace

- Collected trace of queries issued from TAO to MySQL over six days
- Post-cache workload: all writes and cache-miss reads
- Observations:
  - Mostly edge operations
  - Quite read-heavy, even after cache
  - Edge range queries dominate
    - e.g. “most recent comments for post 12345”

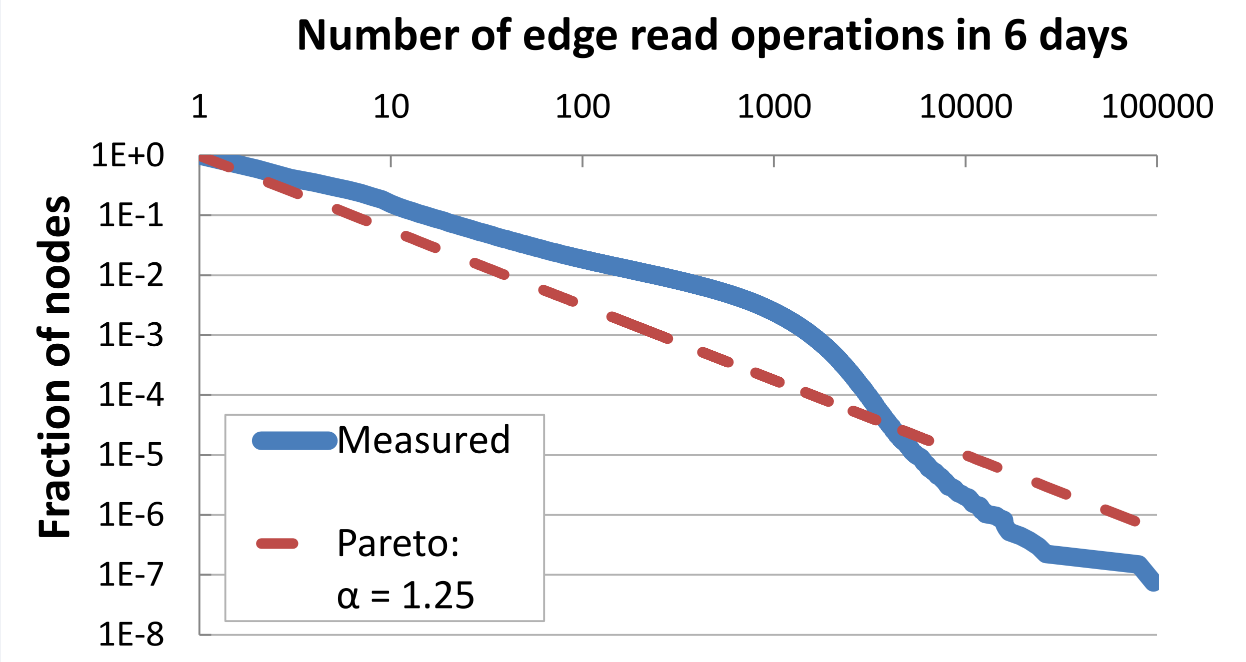
Data Type	Operation	% Queries
Object (graph node)	Get	12.9%
	Insert	2.6%
	Delete	1.0%
	Update	7.4%
Association (graph edge)	Get Count	4.9%
	Get Range	50.7%
	Multiget by Key	0.5%
	Insert	9.0%
	Delete	3.0%
	Update	8.0%

Access patterns

- Node/edge types exhibit markedly different use patterns
- Power-law distribution for reads & writes on node/node out-edges
- Most data is "cold": not accessed
- Graph structure has small influence: read/write frequency correlated with outdegree



Heterogeneity in workload for top 10 edge types.

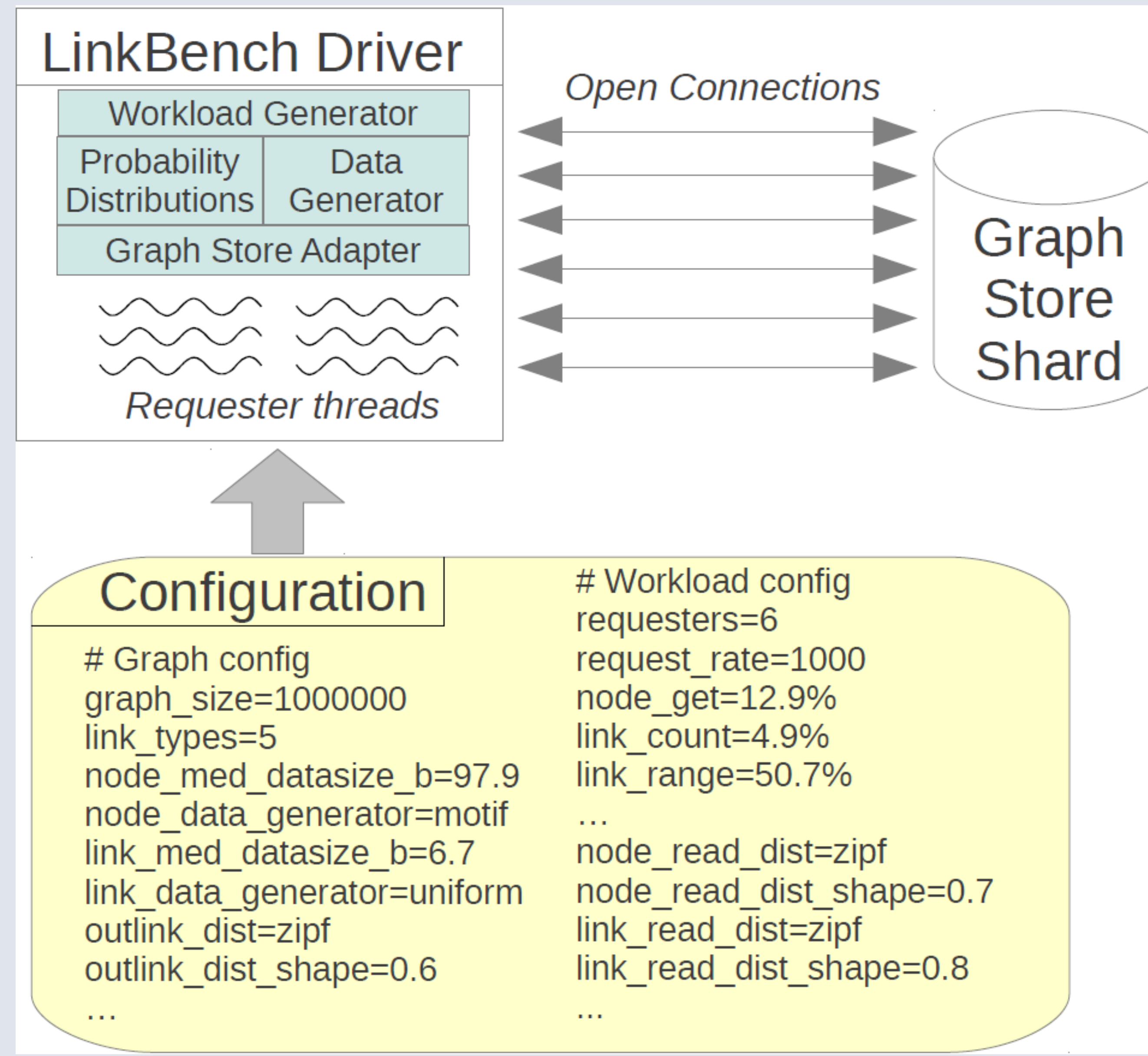


Power-law access frequency for edge read queries. Other query categories show similar distributions.

Synthetic workload in LinkBench

Emulating database clients

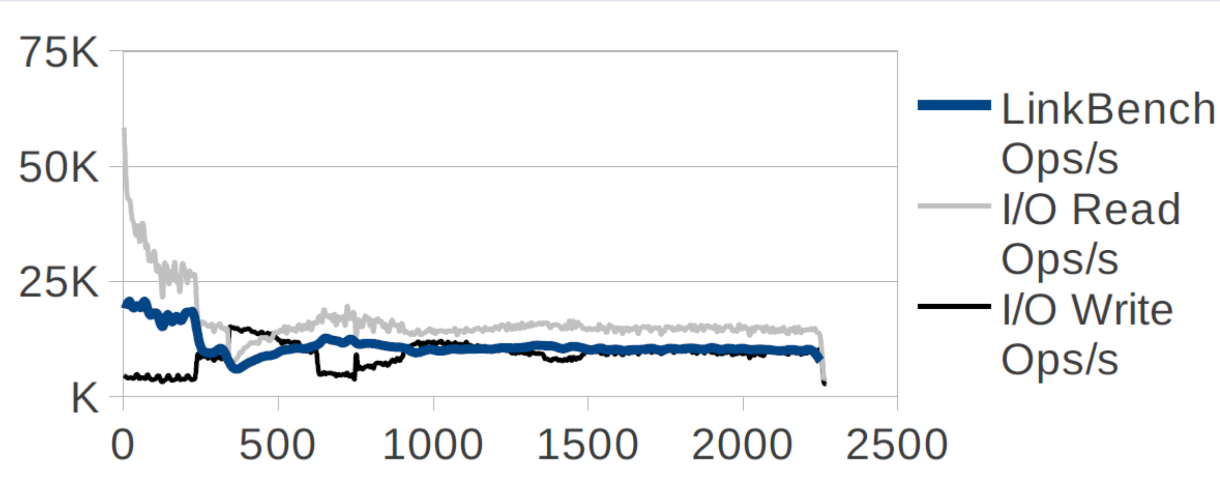
- Independent threads generate query streams
- Statistical properties of query streams fitted to real workload
- Workload is (mostly) stateless: reasonably accurate for post-cache workload



Using LinkBench

Using LinkBench for MySQL

- MySQL 5.1.53 with Facebook patch using InnoDB tables
- 1.2 billion node/5 billion edge graph: 1.4TB on disk
- All data on Solid State Disk
- 16 cores, 144GB RAM
- 11,029 operations/sec average



LinkBench throughput and I/O over time

	mean	p50	p75	p99
object_get	1.6	0.6	1	13
object_insert	4.2	3	5	20
object_delete	5.2	3	6	21
object_update	5.3	3	6	21
assoc_count	1.3	0.5	0.9	12
assoc_range	2.4	1	1	15
assoc_multiget	1.7	0.8	1	14
assoc_insert	10.4	7	14	38
assoc_delete	5.1	1	7	31
assoc_update	10.3	7	14	38

LinkBench operation latencies in milliseconds

LinkBench in use

- Facebook internal testing and development:
  - Comparing MySQL and RocksDB
  - Internal debugging and perf. work
- Publically posted benchmarks:
  - Percona benchmarked stock MySQL vs. Percona MySQL
  - Mark Callaghan benchmarked MySQL's default InnoDB storage engine vs. TokuDB storage engine

Using LinkBench in your work

github.com/facebook/linkbench

- Would love to see more adopters
- Potential uses:
  - Evaluate database systems for a realistic social network workload
  - Addition to general benchmark suites
- We welcome code contributions
  - Adapters for new databases
  - Extensions and improvements

facebook

on

GitHub

Questions?

1 Why another benchmark?

2 Generating a synthetic social graph

3 Generating a realistic query workload

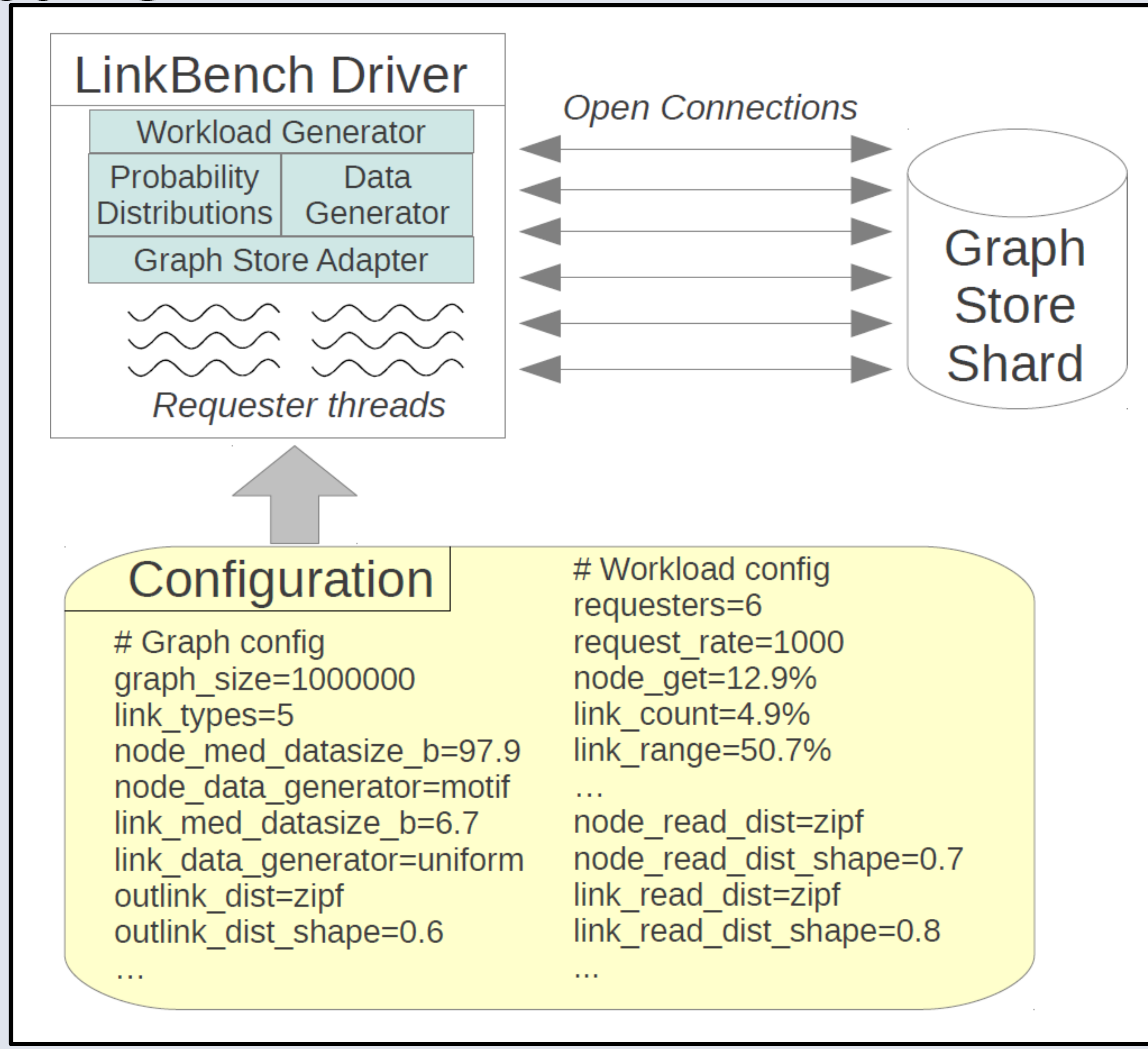
4 Using LinkBench

facebook

Backup slides

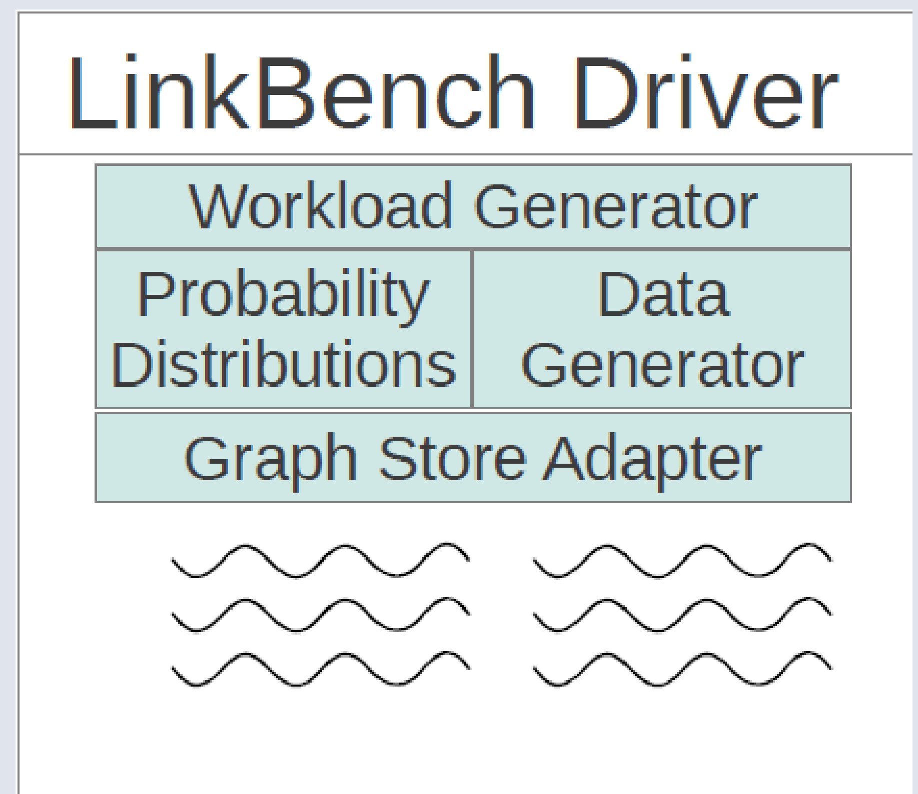
LinkBench architecture

- LinkBench driver simulates client of a graph store
- Configurable/extensible:
  - New databases
  - Different social graph sizes and structure
  - Different query workloads



Workload customization in LinkBench

- Query mix:
  - Read-heavy vs. write-heavy
  - Edges vs. nodes
  - Point vs. range queries
- Client composition:
  - # of concurrent clients
  - request rate
- Access distributions:
  - Alternative probability distributions
  - Changes of distribution or parameters affect working-set size



Additional graph customization

- Number of different edge types
- Degree distribution: empirical, Zipf, uniform, etc.
- Payload data size distribution
- Payload data (e.g. vary compressibility)