Microsoft

Open-Source Technologies for Streaming & State Management

https://aka.ms/Trill
https://aka.ms/FASTER

Speaker: Badrish Chandramouli

Database group, MSR Redmond

OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, …

- Recently, we have open-sourced several research projects

OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, …

- Recently, we have open-sourced several research projects
  - **Trill**: proven **streaming engine** for real-time and offline analytics
    - https://github.com/Microsoft/Trill

OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, …

- Recently, we have open-sourced several research projects
  - **Trill**: proven **streaming engine** for real-time and offline analytics
    - https://github.com/Microsoft/Trill
  - **FASTER**: fast key-value store for resilient **state management**
    - https://github.com/Microsoft/FASTER

OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, …
- Recently, we have open-sourced several research projects
  - **Trill**: proven **streaming engine** for real-time and offline analytics
    - https://github.com/Microsoft/Trill
  - **FASTER**: fast key-value store for resilient **state management**
    - https://github.com/Microsoft/FASTER
  - **CRA**: powerful **distributed runtime** for dataflow graphs
    - https://github.com/Microsoft/CRA

Today's Focus

OSS in the MSR Database Group

- Work on diverse areas such as streaming, big data, key-value stores, storage, security, scale-out, ML for systems, …

- Recently, we have open-sourced several research projects
  - **Trill**: proven **streaming engine** for real-time and offline analytics
    - https://github.com/Microsoft/Trill
  - **FASTER**: fast key-value store for resilient **state management**
    - https://github.com/Microsoft/FASTER
  - **CRA**: powerful **distributed runtime** for dataflow graphs
    - https://github.com/Microsoft/CRA
  - **Ambrosia**: author highly **robust applications** & microservices easily
    - https://github.com/Microsoft/Ambrosia

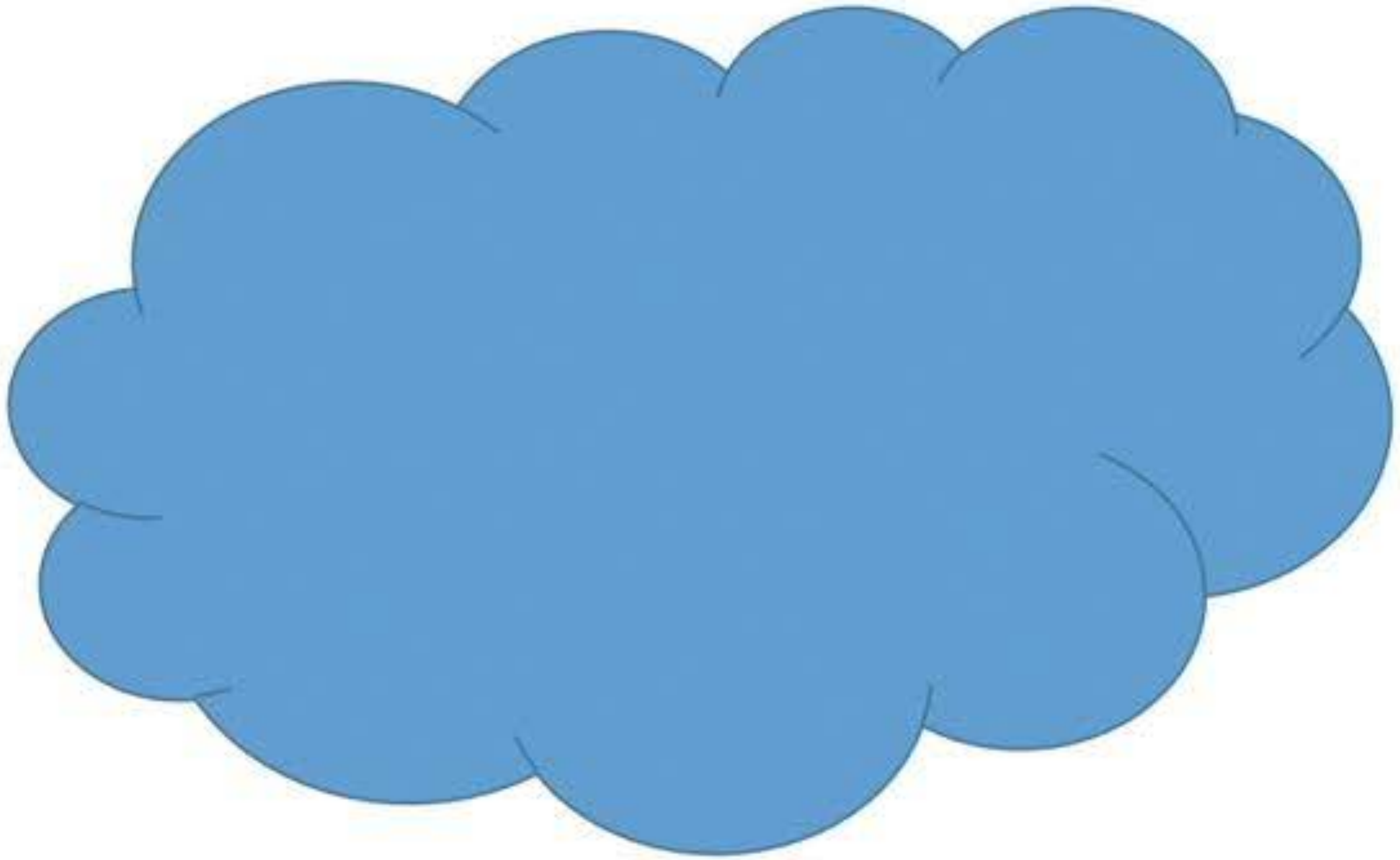
Today's Focus

Trill

Streaming engine for the cloud & edge

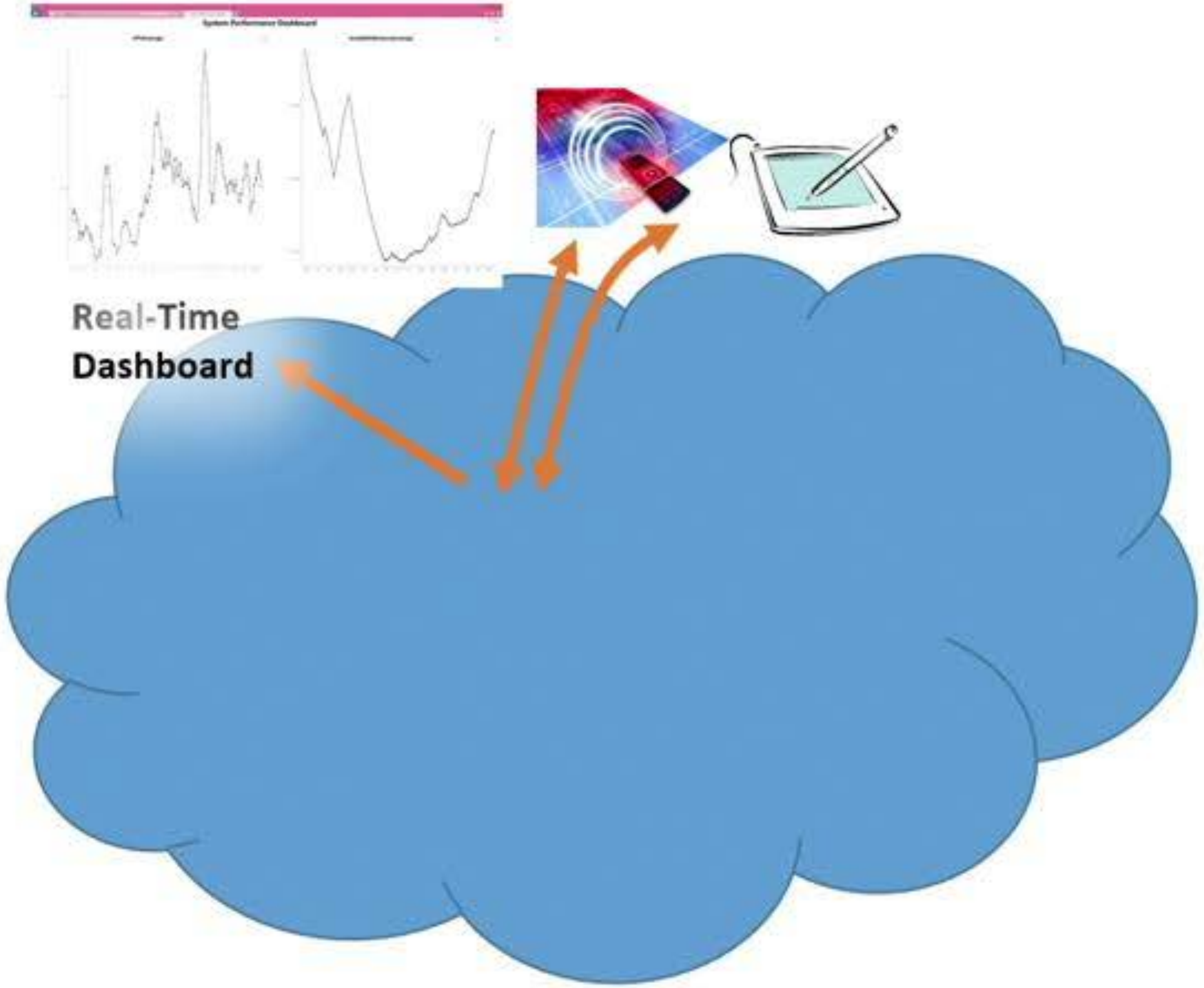
Badrish Chandramouli, Jonathan Goldstein, James Terwilliger, Mike Barnett, Yinan Li,
Peter Freiling, Zhong Chen, and others

Scenarios for Big Data Analytics



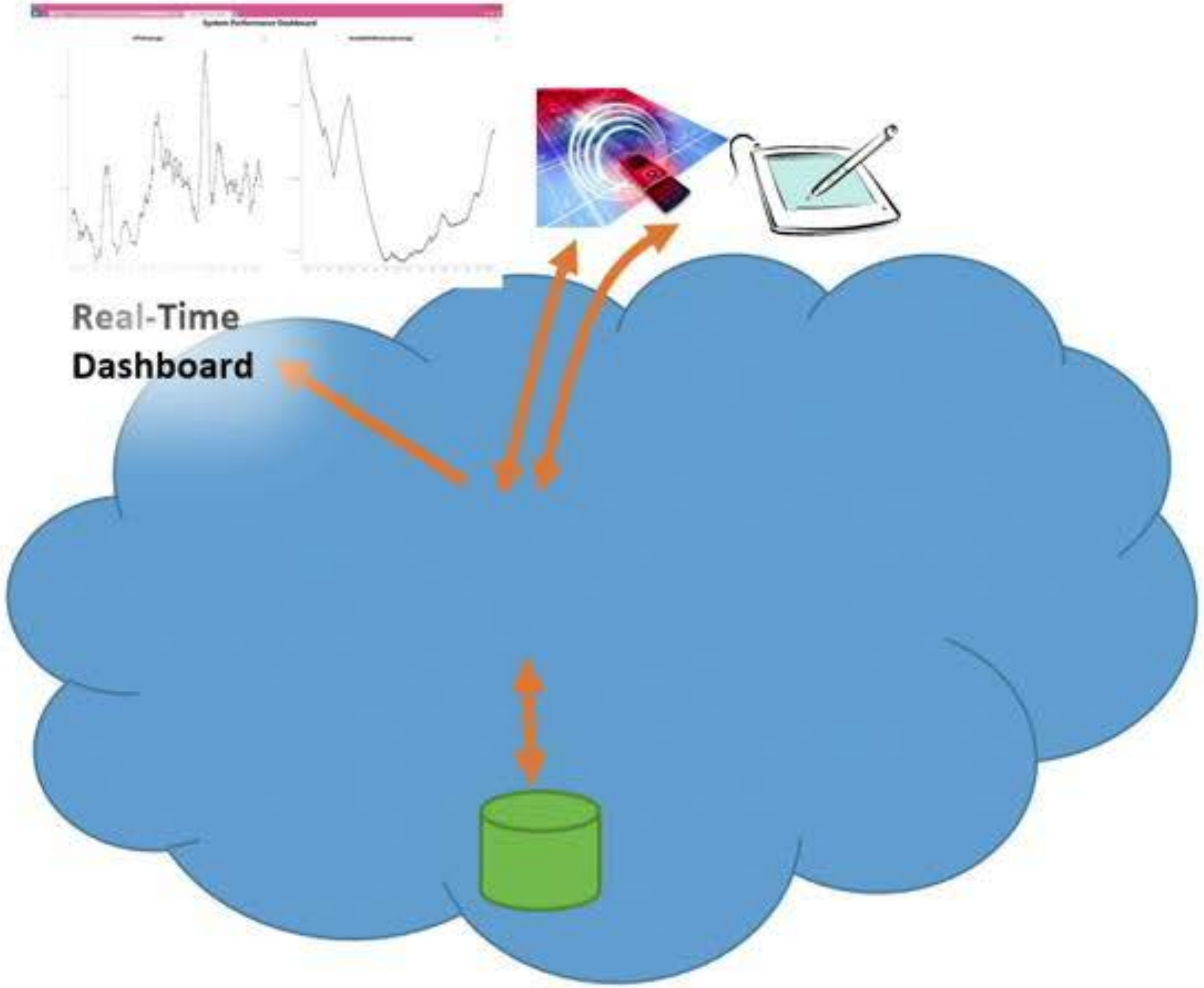
Scenarios for Big Data Analytics

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected



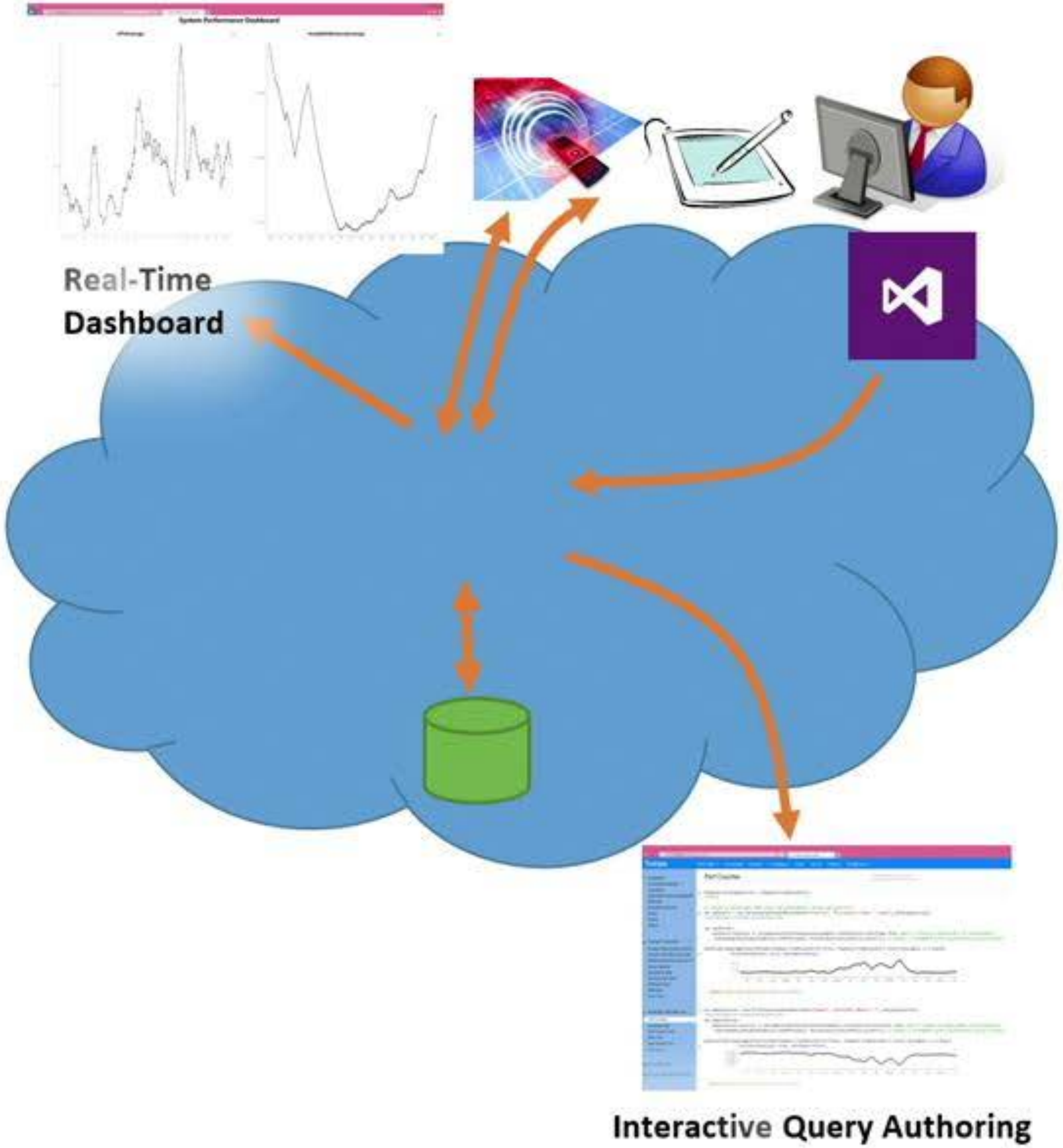
Scenarios for Big Data Analytics

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected
- Real-time with historical
  - Correlate live data stream with historical activity (e.g., from 1 week back)



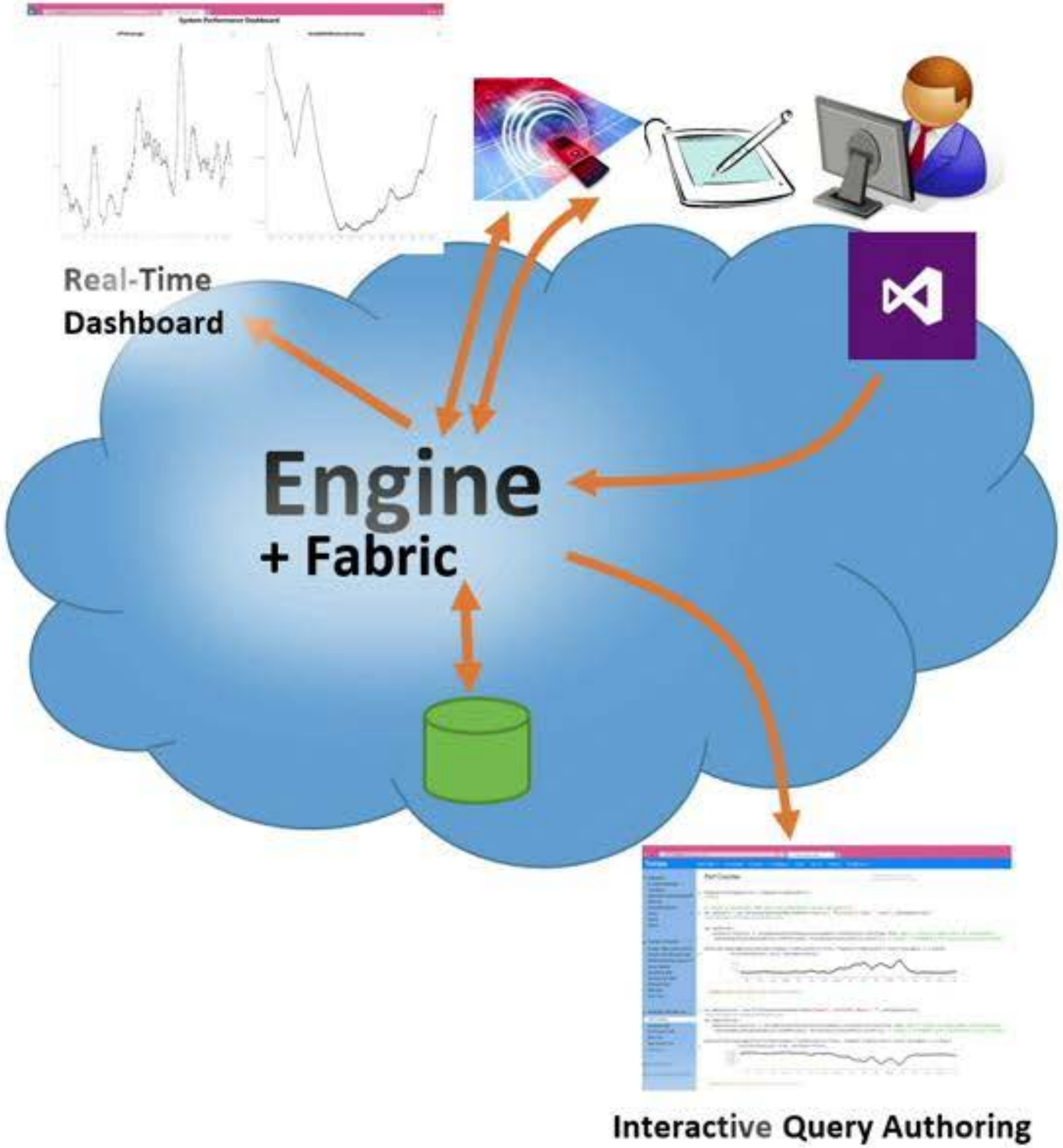
Scenarios for Big Data Analytics

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected
- Real-time with historical
  - Correlate live data stream with historical activity (e.g., from 1 week back)
- Offline
  - Develop initial monitoring query using logs
  - Back-test monitoring query over historical logs
  - Run interactive queries on data



Scenarios for Big Data Analytics

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected
- Real-time with historical
  - Correlate live data stream with historical activity (e.g., from 1 week back)
- Offline
  - Develop initial monitoring query using logs
  - Back-test monitoring query over historical logs
  - Run interactive queries on data



Requirements for “one engine”

Scenarios

- monitor telemetry & raise alerts
- correlate real-time with logs
- develop initial monitoring query
- back-test over historical logs
- offline analysis (BI) with early results

Requirements for "one engine"

- Performance
  - High throughput: critical for large offline datasets
  - Low latency & overhead: Important for real time monitoring

Scenarios

- monitor telemetry & raise alerts
- correlate real-time with logs
- develop initial monitoring query
- back-test over historical logs
- offline analysis (BI) with early results

Requirements for "one engine"

- Performance
  - High throughput: critical for large offline datasets
  - Low latency & overhead: Important for real time monitoring

- Fabric & language integration
  - Cloud app/service acts as driver, *uses* engine as library
  - Need rich data-types, integrate custom logic seamlessly

Scenarios

- monitor telemetry & raise alerts
- correlate real-time with logs
- develop initial monitoring query
- back-test over historical logs
- offline analysis (BI) with early results

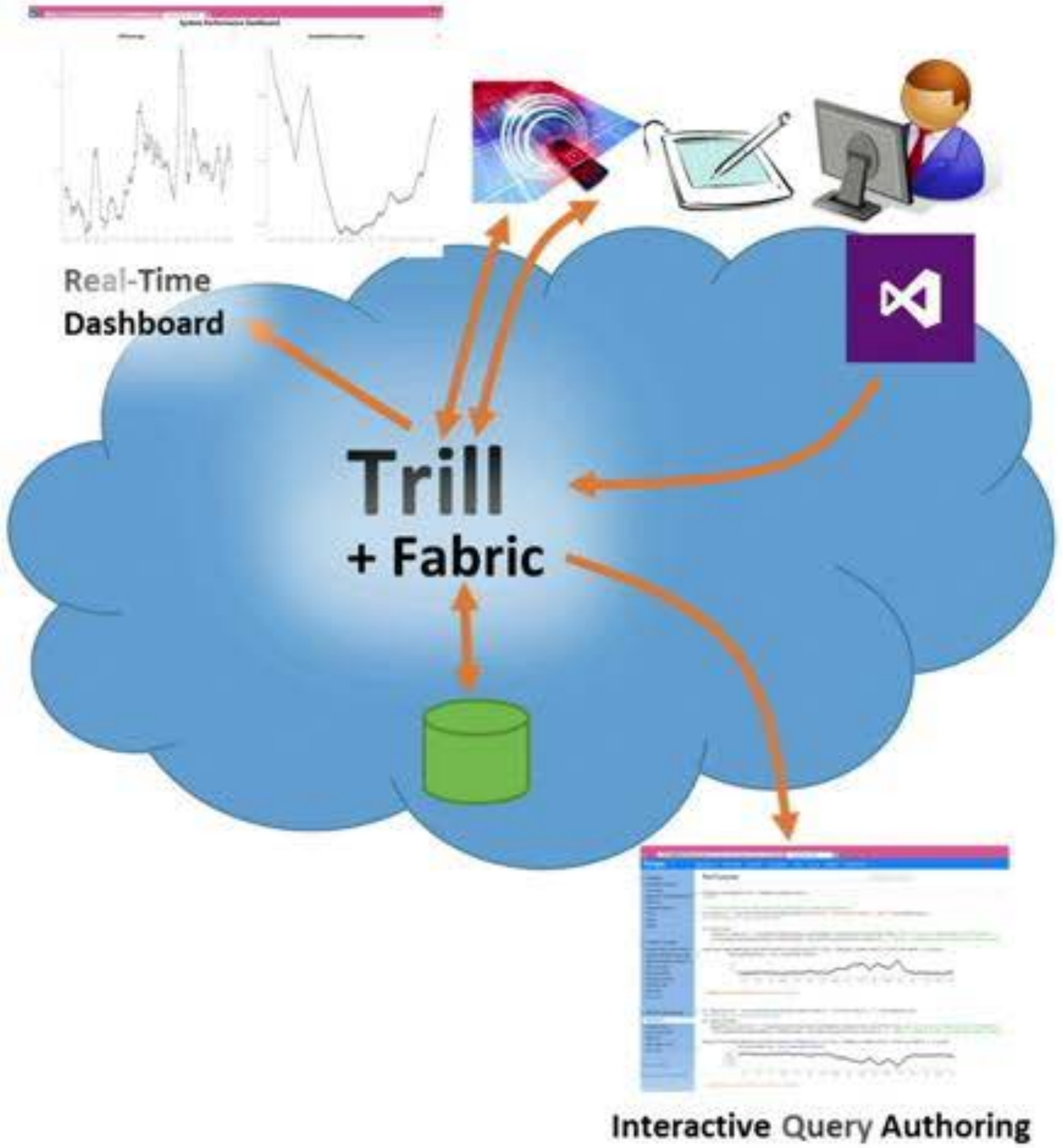
Requirements for "one engine"

- Performance
  - High throughput: critical for large offline datasets
  - Low latency & overhead: Important for real time monitoring
- Fabric & language integration
  - Cloud app/service acts as driver, *uses* engine as library
  - Need rich data-types, integrate custom logic seamlessly
- Query model
  - Need to support real-time and offline data, temporal and relational queries, interactive queries

Scenarios

- monitor telemetry & raise alerts
- correlate real-time with logs
- develop initial monitoring query
- back-test over historical logs
- offline analysis (BI) with early results

Trill: Fast Streaming Analytics Library

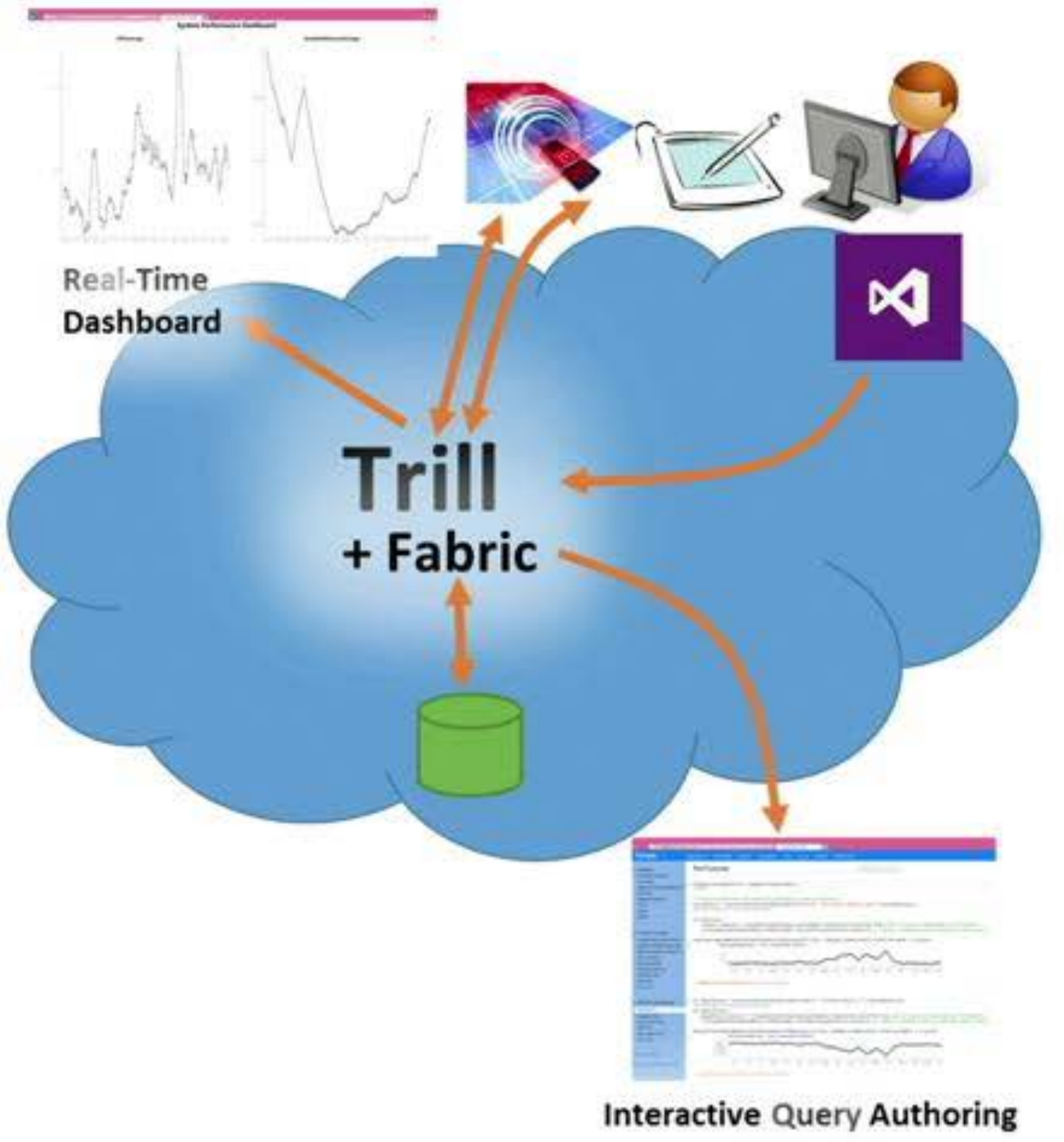


Trill: Fast Streaming Analytics Library

- Performance
  - 2-4 **orders of magnitude** faster than traditional SPEs
  - For relational, comparable to best columnar DBMS
  - User-controlled latency specification
    - explicit latency vs. throughput tradeoff

Trill: Fast Streaming Analytics Library

- Performance
  - 2-4 **orders of magnitude** faster than traditional SPEs
  - For relational, comparable to best columnar DBMS
  - User-controlled latency specification
    - explicit latency vs. throughput tradeoff
- Fabric & language integration
  - Built as high-level language (HLL) library component
  - Works with arbitrary HLL data-types & libraries



Trill: Fast Streaming Analytics Library

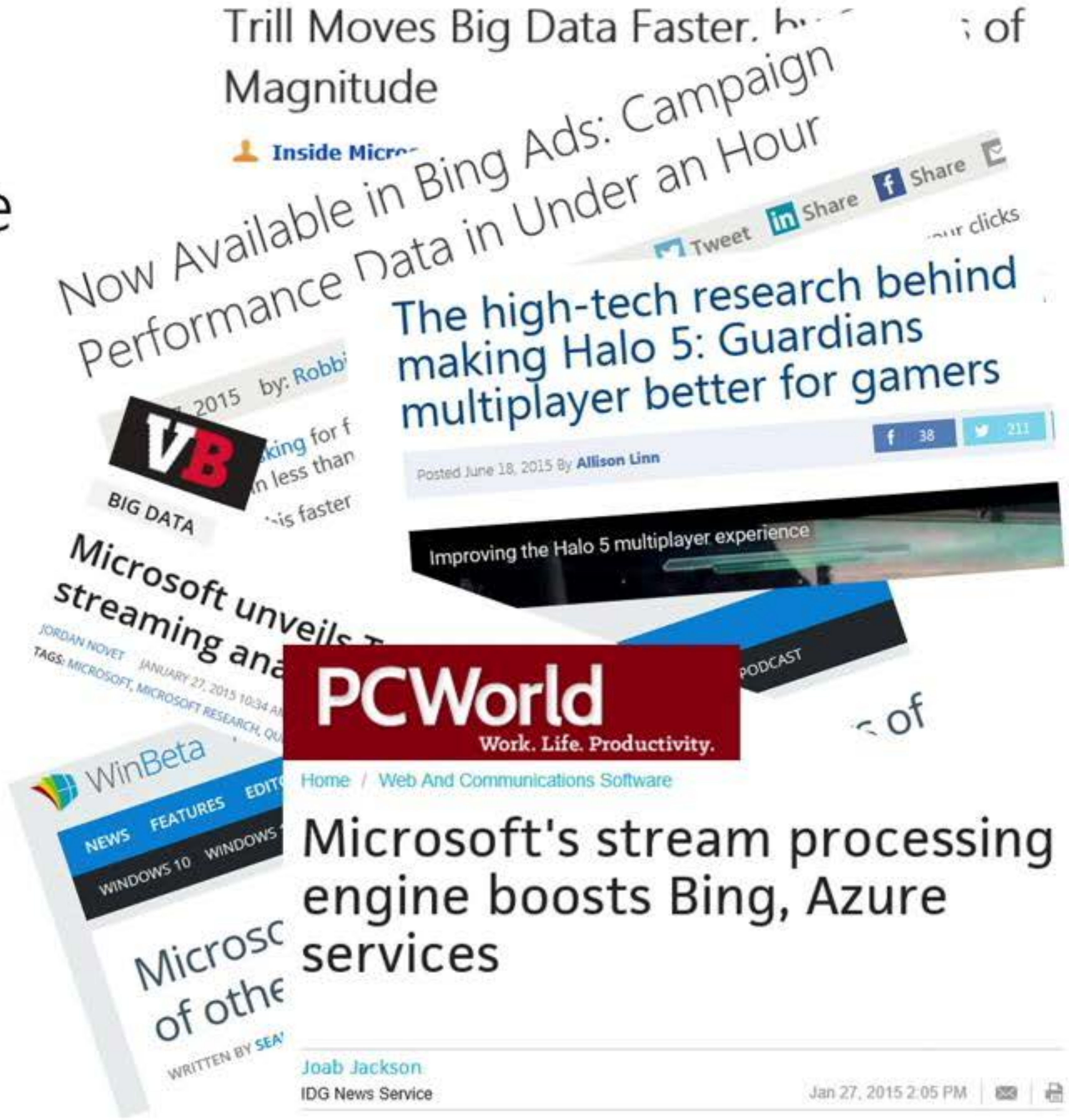
- Performance
  - 2-4 **orders of magnitude** faster than traditional SPEs
  - For relational, comparable to best columnar DBMS
  - User-controlled latency specification
    - explicit latency vs. throughput tradeoff
- Fabric & language integration
  - Built as high-level language (HLL) library component
  - Works with arbitrary HLL data-types & libraries
- Query model
  - Extended LINQ syntax based on temporal + patterns

Used Across Microsoft

- Azure Stream Analytics service
- Bing Ads
- Office, Exchange, Windows
- Halo game monitoring & debugging
- …

Used Across Microsoft

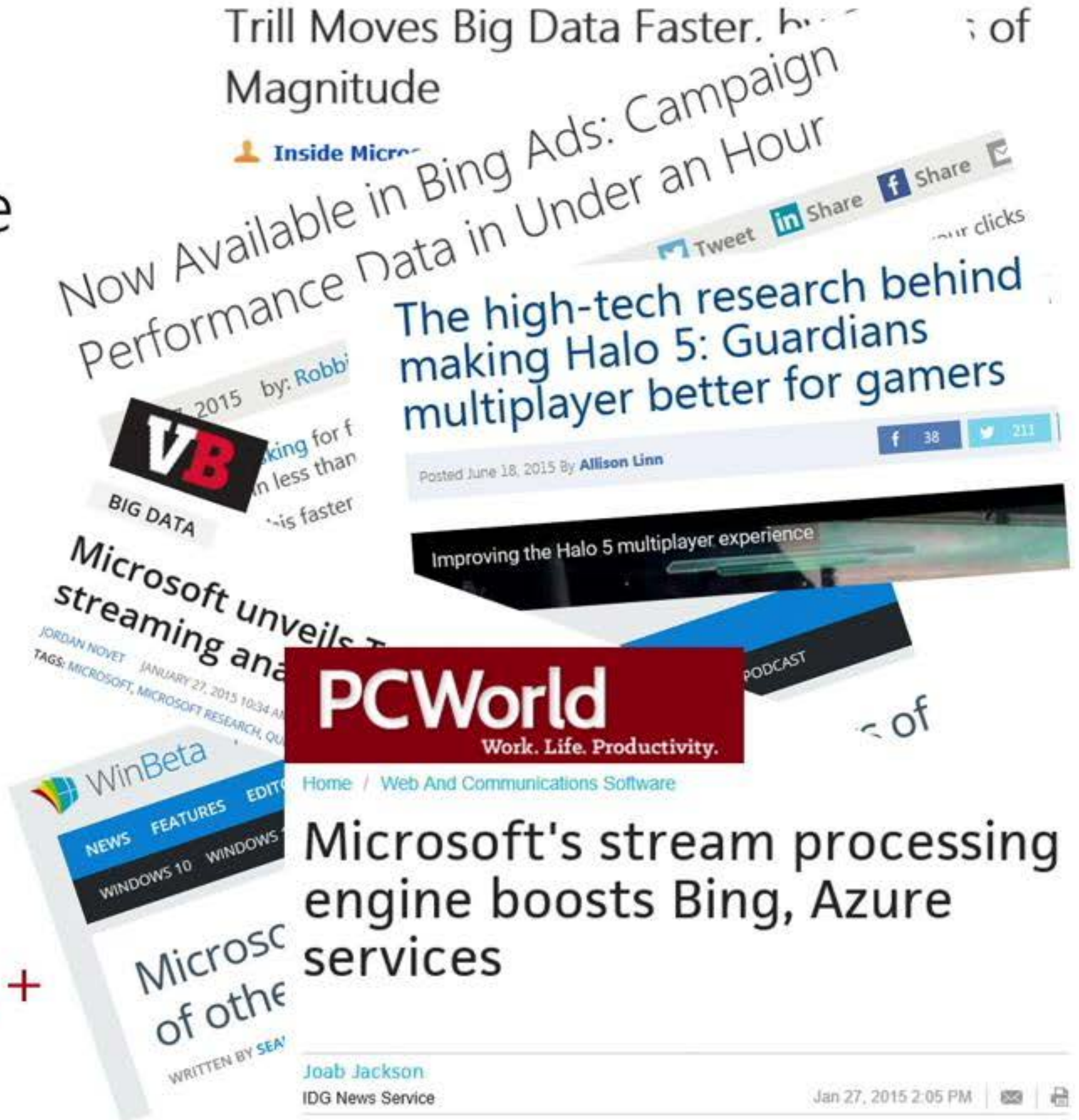
- Azure Stream Analytics service
- Bing Ads
- Office, Exchange, Windows
- Halo game monitoring & debugging
- …



Used Across Microsoft

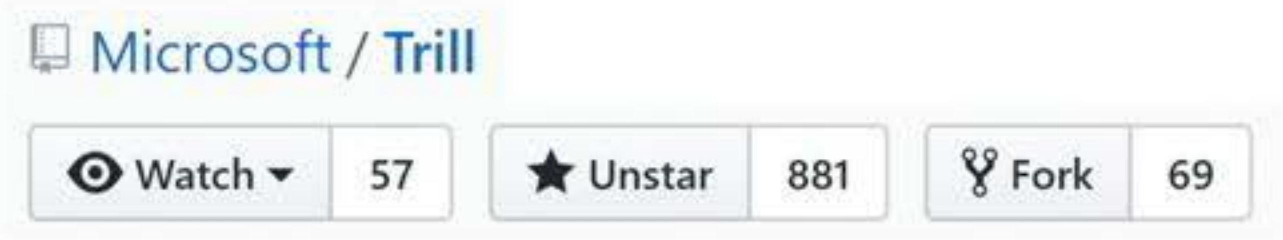
- Azure Stream Analytics service
- Bing Ads
- Office, Exchange, Windows
- Halo game monitoring & debugging
- …

- Key enabler: performance + fabric & language integration + query model



Current Status – https://aka.ms/Trill

- Use and contribute
  - Open source at https://github.com/Microsoft/Trill
  - Library binaries available on NuGet.org



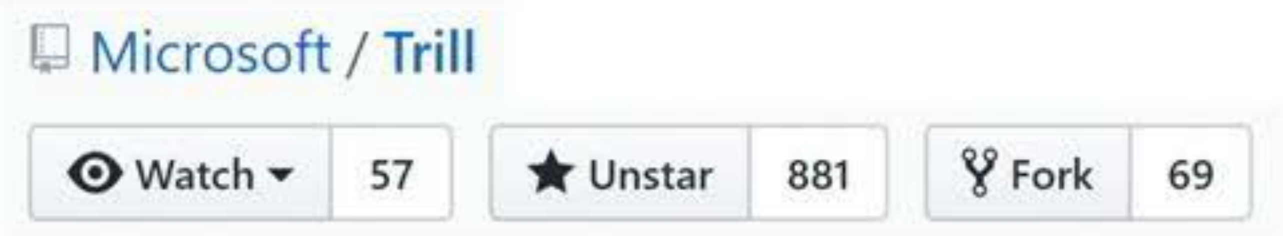
Current Status – https://aka.ms/Trill

- Use and contribute
  - Open source at https://github.com/Microsoft/Trill
  - Library binaries available on NuGet.org

- Features
  - **.NET core** → works on edge, cloud, Windows, Linux, ...
  - Pattern detection, signal processing, extensibility endpoints
  - Trill + CRA → **Quill** for multi-node scan-based analytics
  - Trill + Ambrosia → real-time query pipelines
  - Trill + FASTER → externalize operator state, in progress (covered next)

Current Status – https://aka.ms/Trill

- Use and contribute
  - Open source at https://github.com/Microsoft/Trill
  - Library binaries available on NuGet.org



- Features
  - **.NET core** → works on edge, cloud, Windows, Linux, ...
  - Pattern detection, signal processing, extensibility endpoints
  - Trill + CRA → **Quill** for multi-node scan-based analytics
  - Trill + Ambrosia → real-time query pipelines
  - Trill + FASTER → externalize operator state, in progress (covered next)
- Research Papers
  - Trill paper (VLDB 2015), Trill article (IEEE Data Engg. Bulletin 2016), Quill (VLDB 2016), Signal Processing (SIGMOD 2017), Stream Sorting (ICDE 2018), ...

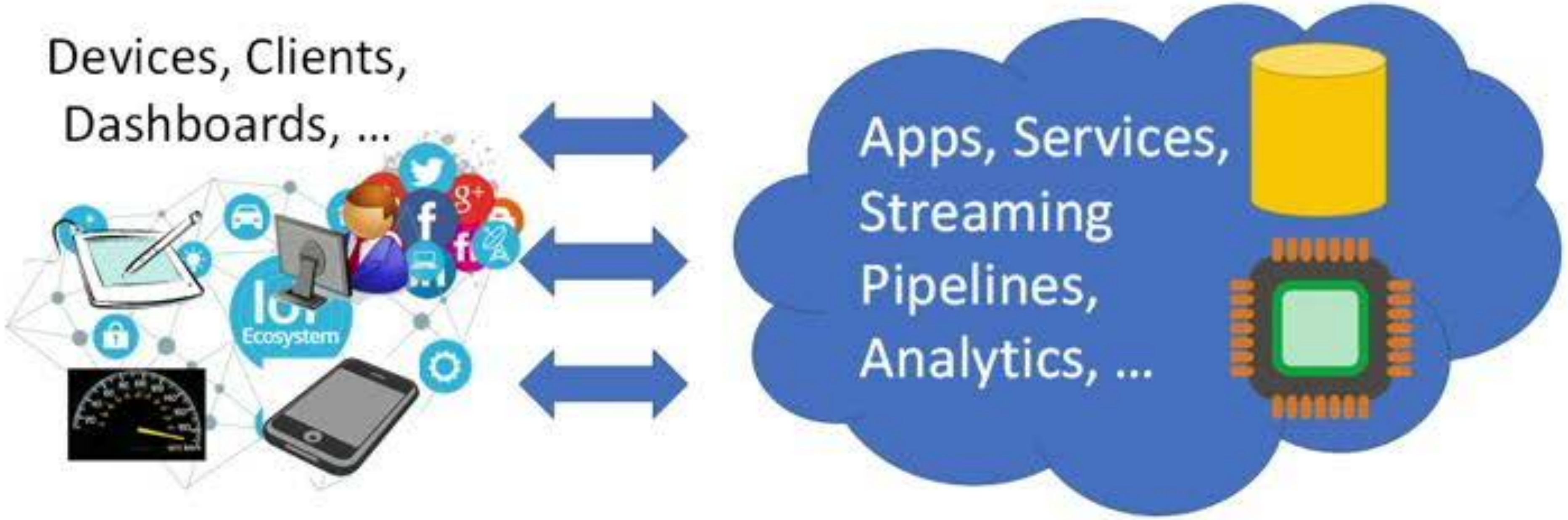
FASTER

Embedded key-value store for state management

Badrish Chandramouli, Donald Kossmann, Guna Prasaad, James Hunter, Justin Levandoski, Mike Barnett, Peter Freiling, James Terwilliger, and others

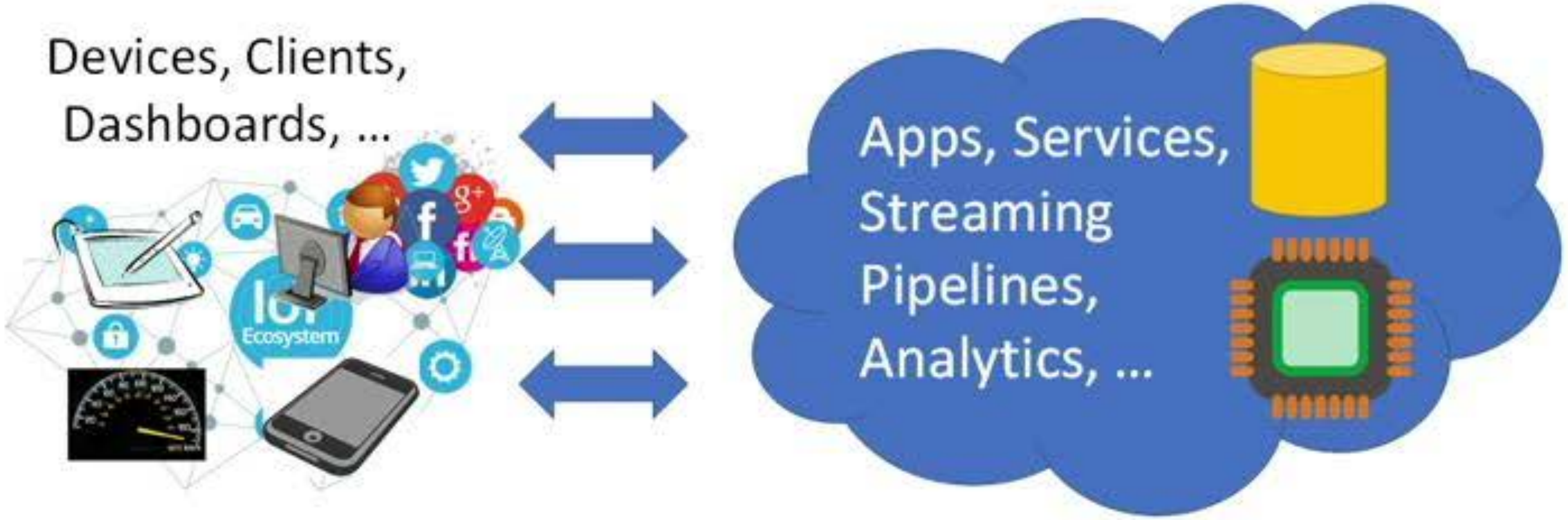
The State Management Problem

- Tremendous growth in data-intensive applications and services
  - Tracking IoT devices, data center monitoring, streaming, online services, …



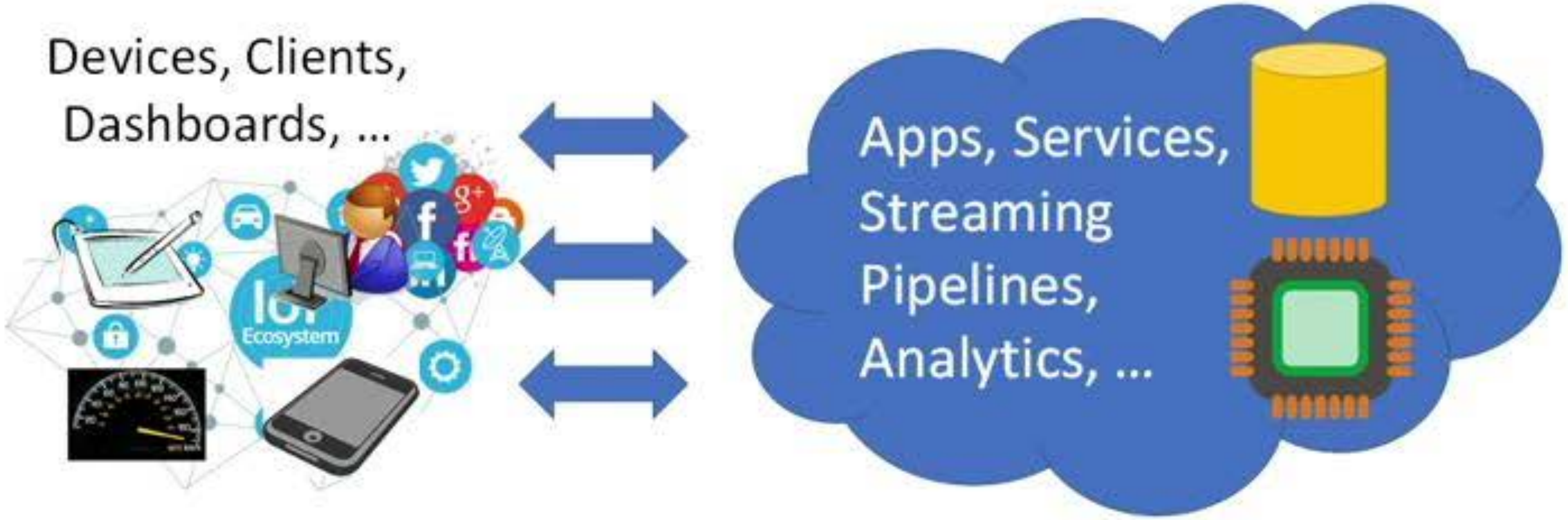
The State Management Problem

- Tremendous growth in data-intensive applications and services
  - Tracking IoT devices, data center monitoring, streaming, online services, …
- State management is a hard problem
  - State consists of independent objects – *devices, users, ads*
  - State does not fit in memory → **problem for edge & multi-tenant as well**
  - Point ops with lots of updates – *e.g., update per-device average CPU reading*
  - State needs to be recoverable



The State Management Problem

- Tremendous growth in data-intensive applications and services
  - Tracking IoT devices, data center monitoring, streaming, online services, …
- State management is a hard problem
  - State consists of independent objects – *devices, users, ads*
  - State does not fit in memory → **problem for edge & multi-tenant as well**
  - Point ops with lots of updates – *e.g., update per-device average CPU reading*
  - State needs to be recoverable



Temporal Locality

- **Search engine maintains per-user stats over last week**
- **Billions of users "alive"**
- **Only millions actively surfing at given instant of time**

What is FASTER

- Latch-free concurrent multi-core hash key-value store
  - Designed for high performance and scalability across threads (shared memory)
  - Supports data larger than main memory + recovery
  - Shapes the (changing) hot working set in memory → integrated cache

What is FASTER

- Latch-free concurrent multi-core hash key-value store
  - Designed for high performance and scalability across threads (shared memory)
  - Supports data larger than main memory + recovery
  - Shapes the (changing) hot working set in memory → integrated cache

- Performance: up to 200 million ops/sec for YCSB variants
  - One Intel Xeon machine, two sockets, 72 threads
  - Exceeds throughput of pure in-memory systems when working set fits in memory

What is FASTER

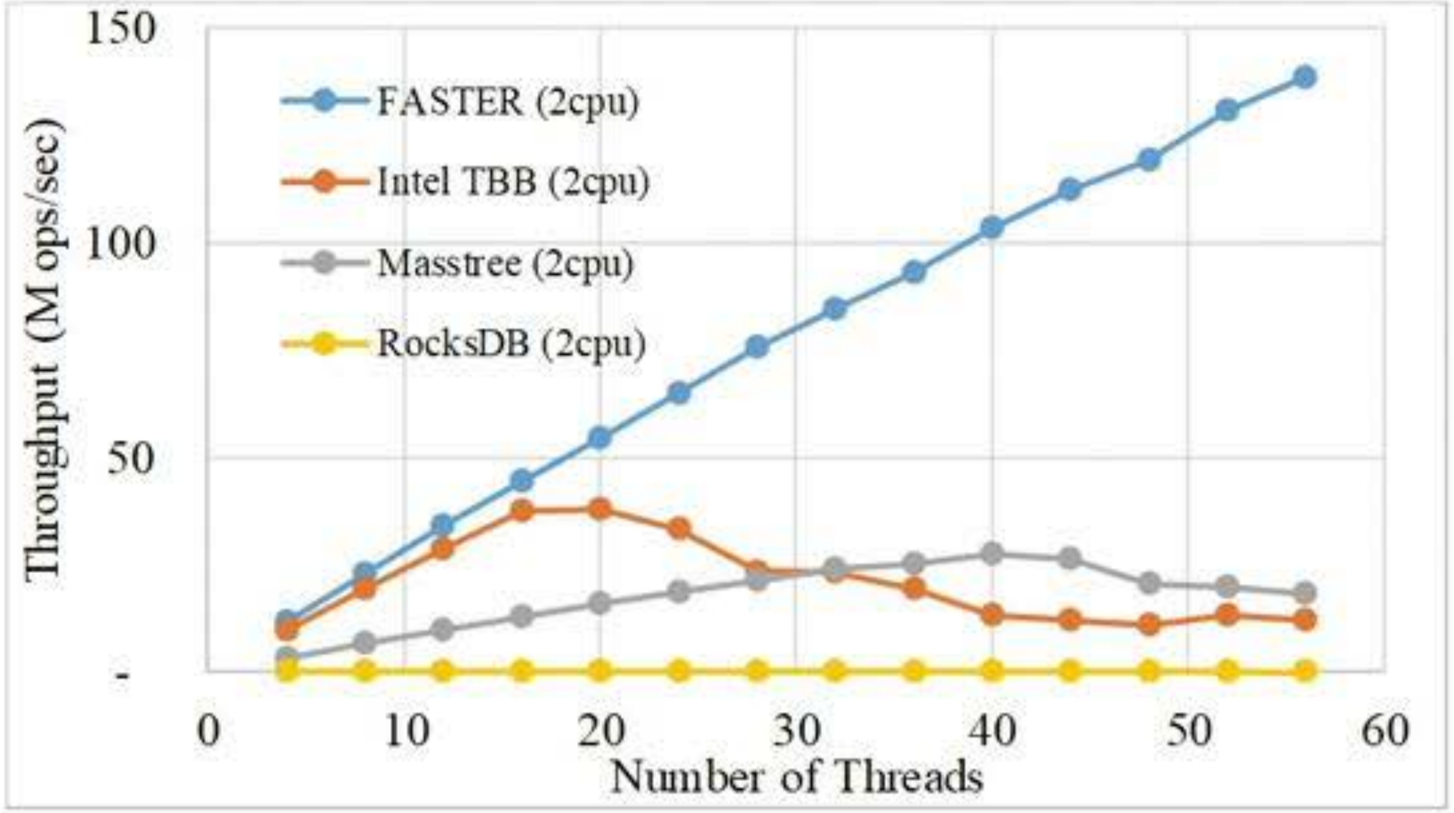
- Latch-free concurrent multi-core hash key-value store
  - Designed for high performance and scalability across threads (shared memory)
  - Supports data larger than main memory + recovery
  - Shapes the (changing) hot working set in memory → integrated cache

- Performance: up to 200 million ops/sec for YCSB variants
  - One Intel Xeon machine, two sockets, 72 threads
  - Exceeds throughput of pure in-memory systems when working set fits in memory

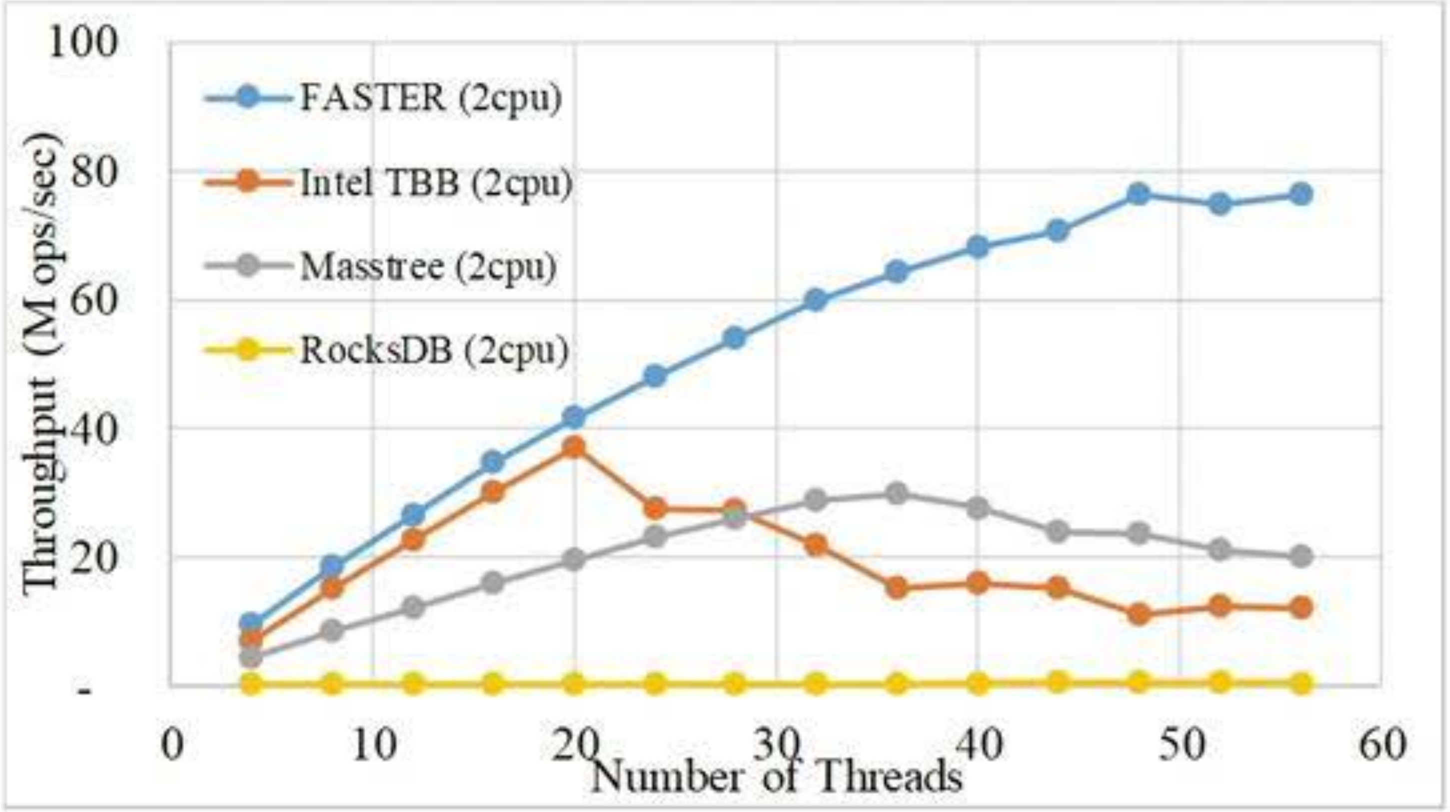
- FASTER Interface
  - Read, Blind Update
  - Atomic read-modify-write (RMW) - for running aggs (like sum), partial field updates

Scalability with # Threads

- When current working set "happens to fit" in memory

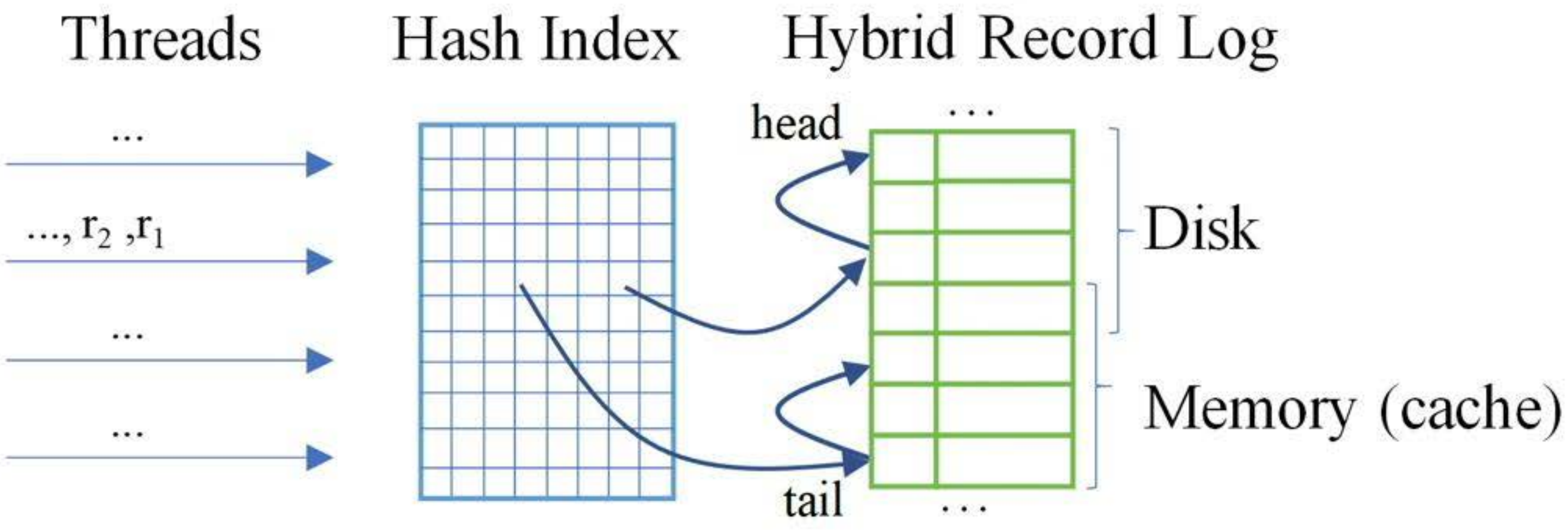


100% RMW; 8 byte payloads

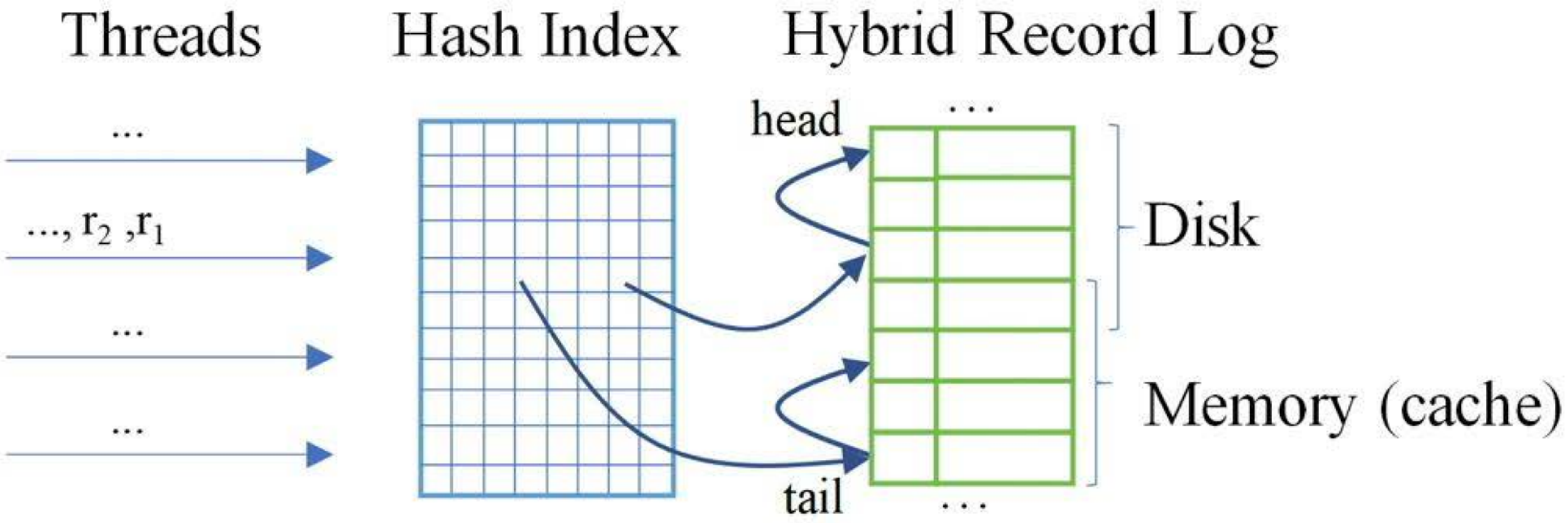


100% blind updates; 100 byte payloads

System Architecture



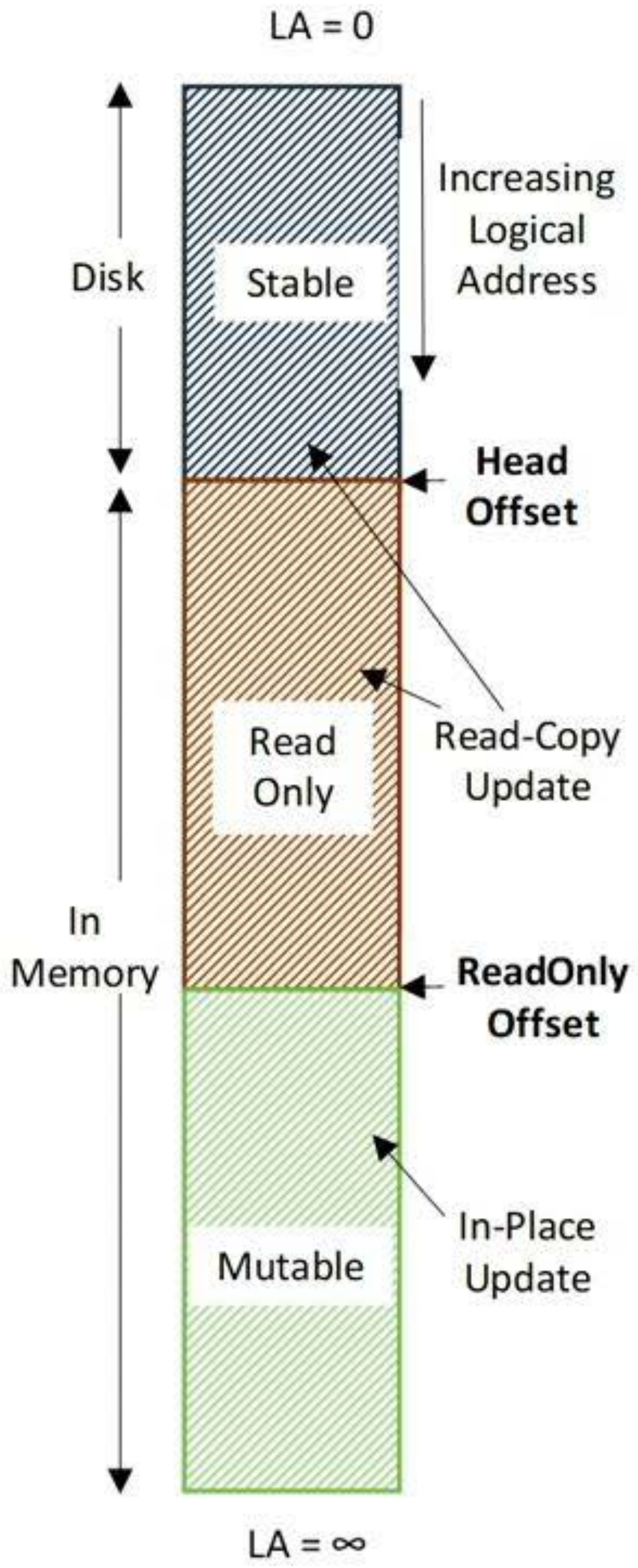
System Architecture



- Technical Innovations
  - **Indexing**: Concurrent hash index (see paper)
  - **Record Storage**: "Hybrid Log" record allocator
  - **Threading**: Epoch Protection Framework with Trigger Actions (see paper)

Hybrid Log Allocator

- Divide memory into three regions
  - Stable (on disk) → Read-Copy-Update (RCU)
  - Mutable (in memory) → In-Place Update (IPU)
  - Read-only (in memory) → Read-Copy-Update (RCU)

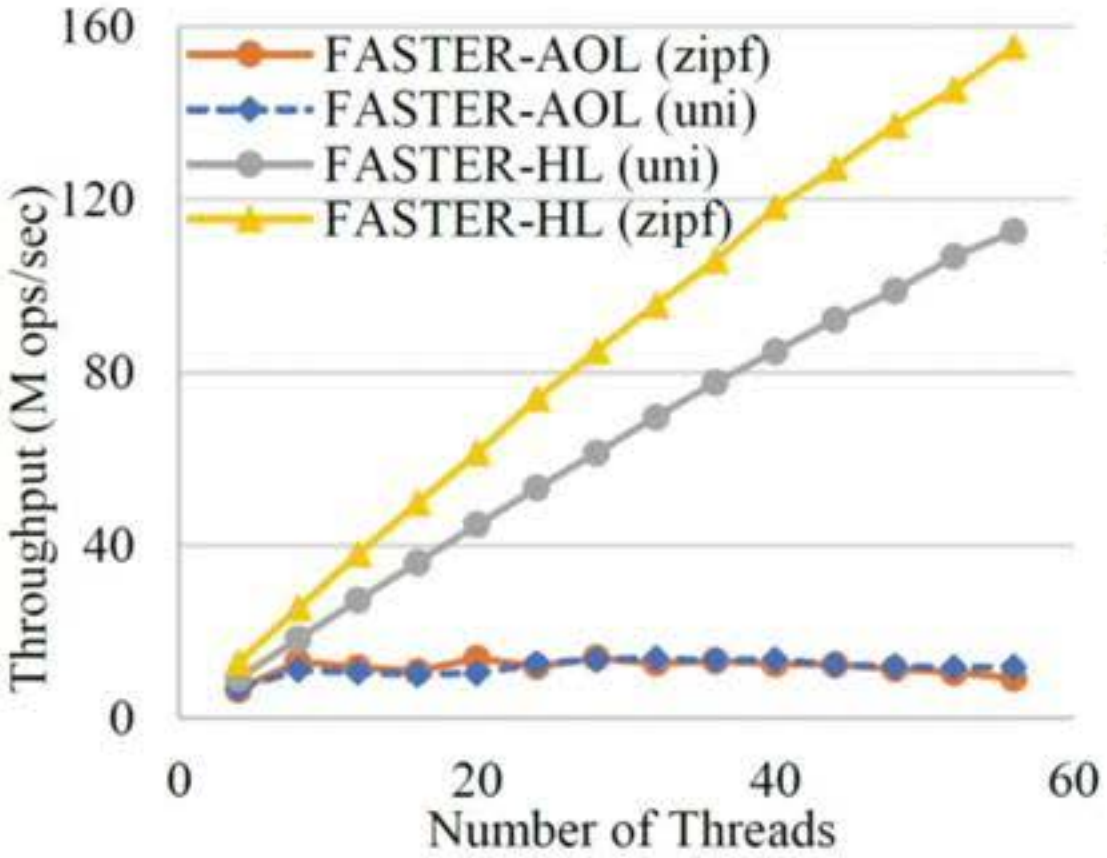


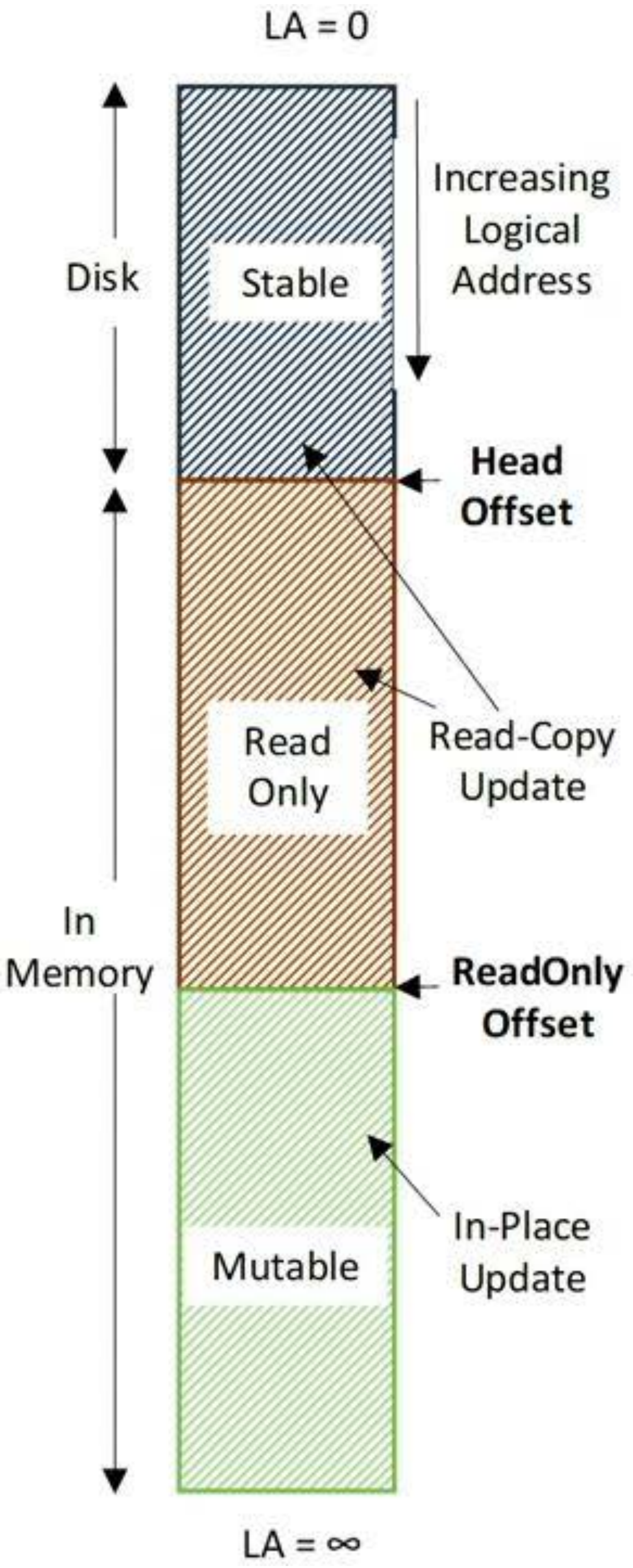
Hybrid Log Allocator

- Basic RMW Algorithm

Logical Address	Operation
< Head Offset	Issue async IO request
< ReadOnly Offset	Copy to tail, CAS-update hash index
< Infinity	Update in-place
New Record	Add to tail, update hash table

- Removes append-only log bottleneck
- Elegant design, but hard to maintain multi-threaded correctness
  - See SIGMOD 2018 paper





Status – https://aka.ms/FASTER

- Open sourced August 2018 (github.com/Microsoft/FASTER)
  - NuGet package available as well, **C# and C++** versions of code

Status – https://aka.ms/FASTER

- Open sourced August 2018 (github.com/Microsoft/FASTER)
  - NuGet package available as well, **C# and C++** versions of code

- Reached front page of Hackernews twice
- Papers: SIGMOD 2018 (core system), VLDB 2018 (demo), SIGMOD 2019 (recovery)
- Integrating FASTER as state store of Trill

Talk Summary

- We have recently open sourced several research projects
  - **Trill**: proven **streaming engine** for real-time and offline analytics
    - https://github.com/Microsoft/Trill
  - **FASTER**: fast key-value store for resilient **state management**
    - https://github.com/Microsoft/FASTER
  - **CRA**: powerful **distributed runtime** for dataflow graphs
    - https://github.com/Microsoft/CRA
  - **Ambrosia**: author highly **robust applications** & microservices easily
    - https://github.com/Microsoft/Ambrosia

Covered Today

- Invite everyone to use, contribute, and perform follow-up research
- Talk to us for more details, go to GitHub for docs & guides

Democratizing Data Preparation for AI

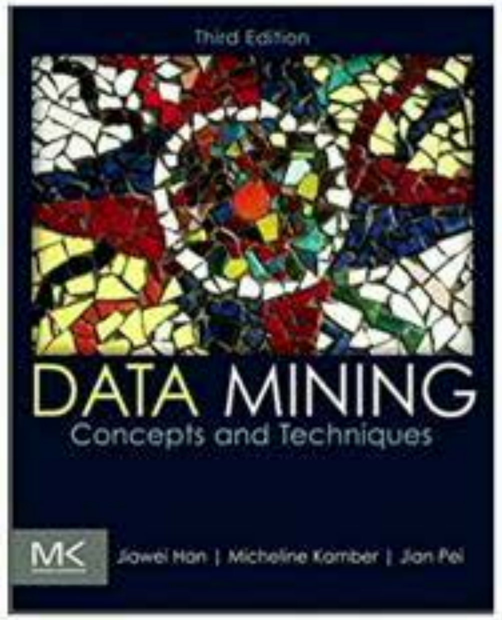
Jiannan Wang

Simon Fraser University

SFU DB/DM Group

History

- Over 30 years of research experience in database and data mining
- Wrote a Data Mining Textbook widely used in the world
- Invented many famous data mining algorithms (e.g., FP-Growth, DBScan, Prefixspan)



Mining frequent patterns without candidate generation J Han, J Pei, Y Yin ACM sigmod record 29 (2), 1-12	7978	2000
Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth J Pei, J Han, B Mortazavi-Asl, H Pinto, Q Chen, U Dayal, MC Hsu icccn, 0215	2662	2001
A density-based algorithm for discovering clusters in large spatial databases with noise. M Ester, HP Kriegel, J Sander, X Xu Kdd 96 (34), 226-231	12578	1996

SFU DB/DM Group

- **Research Areas:** Machine Learning, Data Science, and Big Data Systems
- **Research Strengths:** Cloud Databases, Crowdsourced Data Management, Data Cleaning and Integration, Data Security and Privacy, Fraud Detection, Interpretable Machine Learning, Precision Medicine, Recommender Systems
- **Ranked 13th** in databases and data mining in North America (source: csrankings.org)

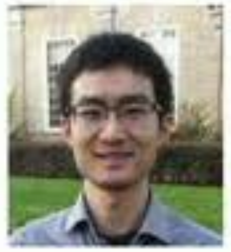
#	Institution	Count	Faculty
1	Carnegie Mellon University	17.7	33
2	Univ. of Illinois at Urbana-Champaign	14.9	11
3	Stanford University	13.0	15
4	Georgia Institute of Technology	11.5	23
4	University of Michigan	11.5	14
6	Massachusetts Institute of Technology	10.3	18
7	Cornell University	10.2	24
8	Purdue University	8.8	13
9	Pennsylvania State University	8.7	8
10	University of California - Los Angeles	8.6	10
10	University of Massachusetts Amherst	8.6	16
12	University of Illinois at Chicago	8.2	7
13	Simon Fraser University	8.1	7
13	University of Maryland - College Park	8.1	11
15	University of Waterloo	7.9	20
16	Duke University	7.6	8
16	University of California - Santa Barbara	7.6	8
18	University of California - Santa Cruz	7.5	12
19	University of Wisconsin - Madison	7.4	13
20	Ohio State University	7.2	11
21	University of California - Riverside	7.0	10
22	University of California - San Diego	6.7	14
23	University at Buffalo	6.4	12

Ke Wang
(Joined in 2000)

Martin Ester
(Joined in 2001)

Jian Pei
(Joined in 2004)

Jiannan Wang
(Joined in 2016)



Tianzheng Wang
(Joined Fall 2018)

Democratizing AI

Democratizing AI

Computing

Algorithms

Training Data

Democratizing AI

Computing

Azure

amazon web services

Google Cloud Platform

Algorithms

Training Data

Democratizing AI

Computing

Azure

amazon web services

Google Cloud Platform

Algorithms

Microsoft CNTK

PYTORCH

TensorFlow

Training Data

Democratizing AI

Computing

Azure

amazon web services

Google Cloud Platform

Algorithms

Microsoft CNTK

PYTORCH

TensorFlow

Training Data

BigGorilla

TRIFACTA

snorkel

Democratizing AI

Computing

Azure

amazon web services

Google Cloud Platform

Algorithms

Microsoft CNTK

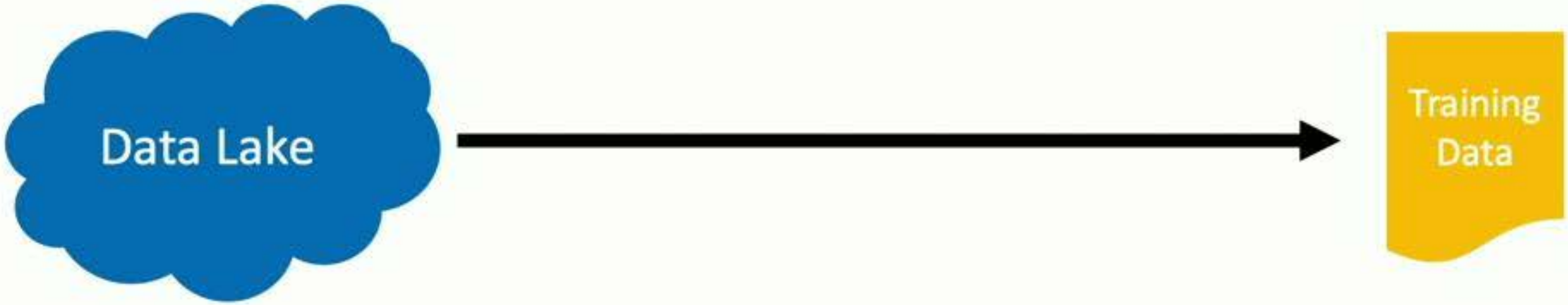
PYTORCH

TensorFlow

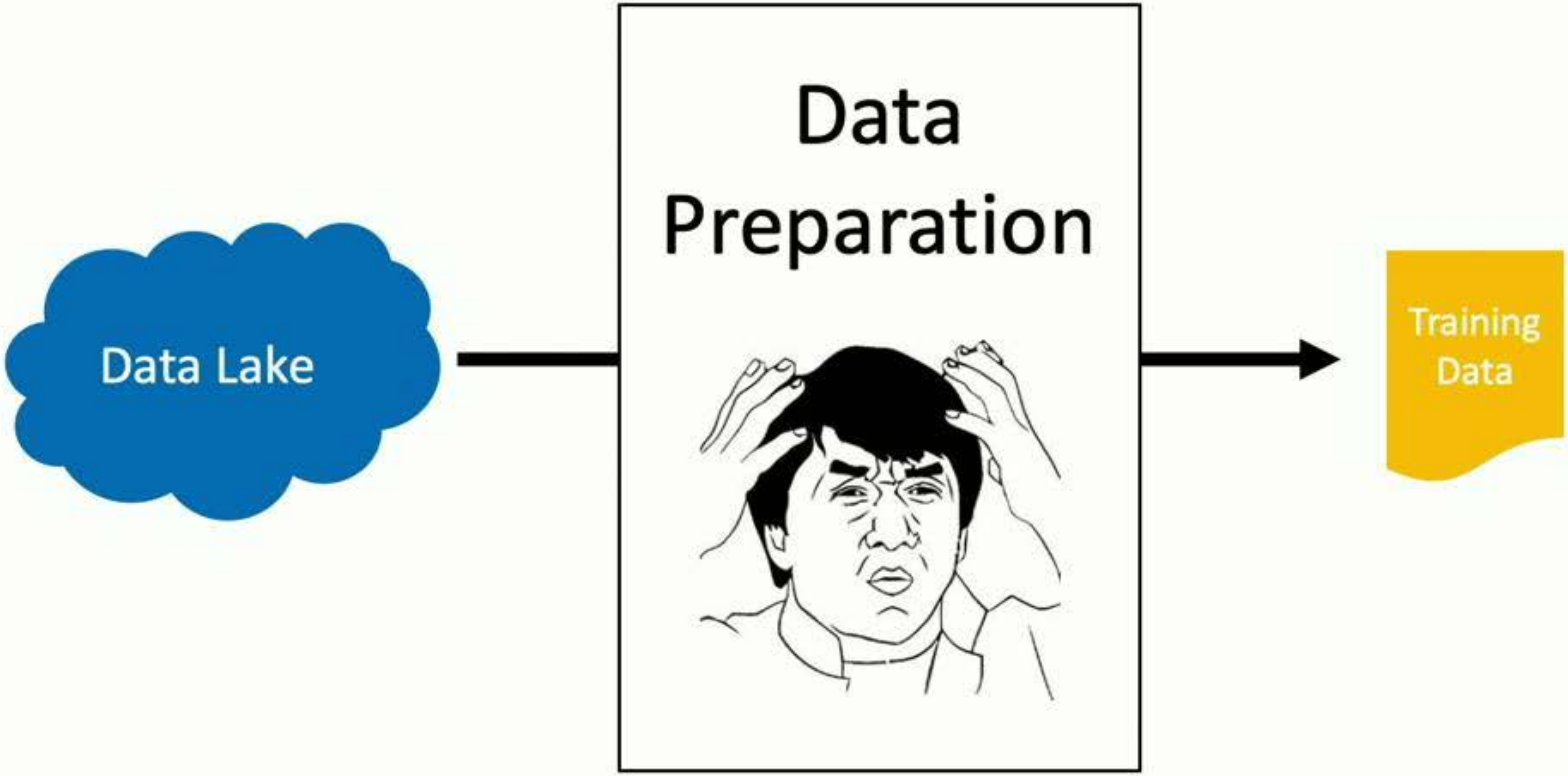
Training Data

The Bottleneck

What is Data Prep?



What is Data Prep?



Why is Data Prep so hard?

- Data Discovery
- Data Profiling
- Data Extraction
- Data Normalization
- Data Enrichment
- Data Transformation
- Data Filtering
- Data Provenance
- Data Labeling
- Error Detection
- Schema Matching
- Deduplication
- Outlier Detection
- Imputation
- …

Inspired by the conversation with Dr. Phil Bernstein at CIDR 2017

New Opportunities for DB Community

Focus on reducing **data scientists' time**

- Ease of Use
- Extensibility
- Composability

New Opportunities for DB Community

Focus on reducing **data scientists' time**

- Ease of Use
- Extensibility
- Composability

Focus on using **advanced ML technologies**

- Automated Machine Learning
- Meta Learning (a.k.a. Learning to Learn)

Recent Progress

Deeper [SIGMOD 2018 (Demo), SIGMOD 2019]

- Reduce data enrichment time

AQP++ [SIGMOD 2018]

- Reduce exploratory data analysis time

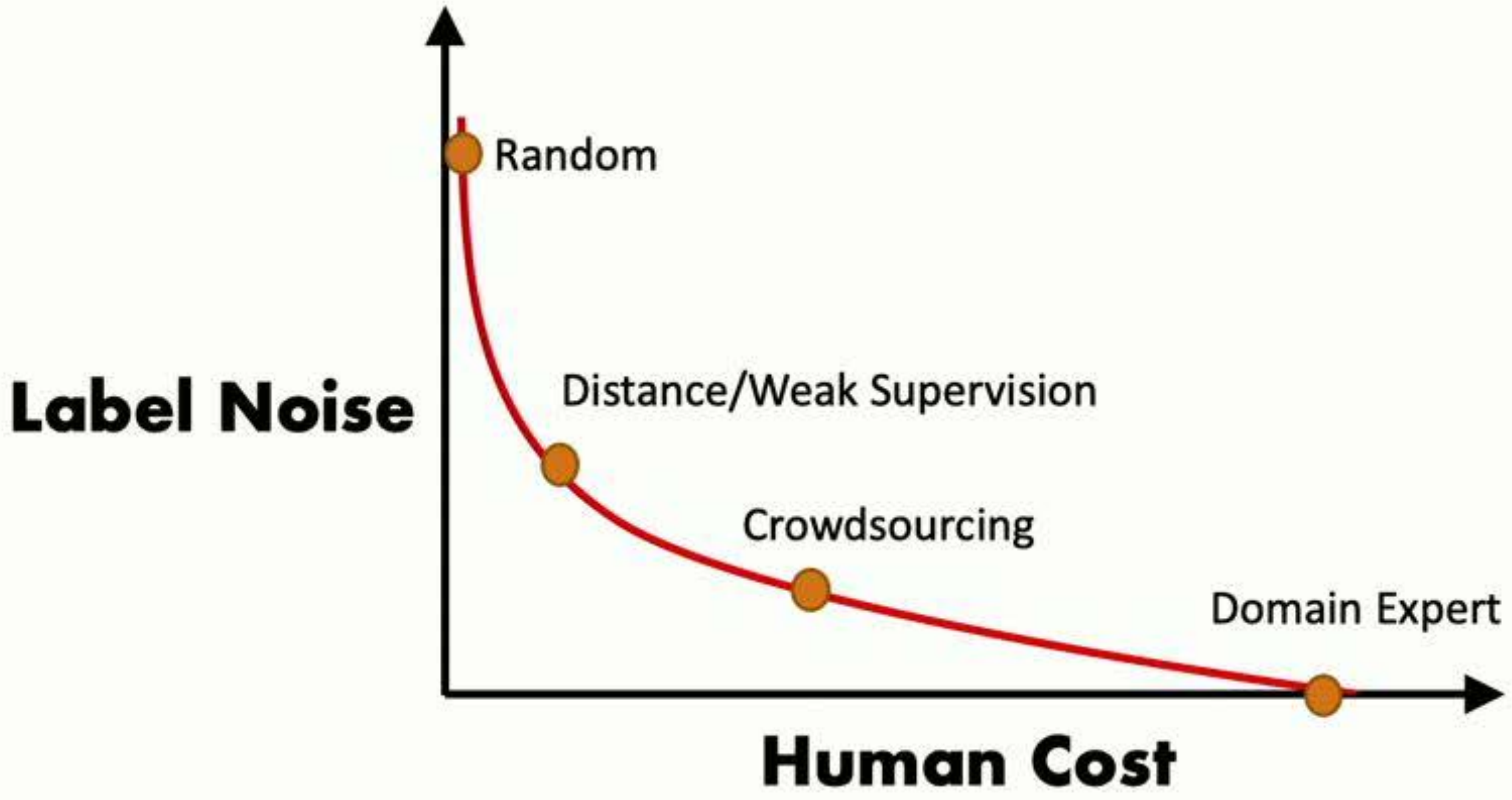
Recent Progress

Deeper [SIGMOD 2018 (Demo), SIGMOD 2019]
- Reduce data enrichment time

AQP++ [SIGMOD 2018]
- Reduce exploratory data analysis time

TARS [VLDB 2019]
- Reduce data labeling time

A Promising Solution



Label Noise vs. Human Cost Trade-off

Cleaning Noisy Label

Existing Work*

- No Cleaning
- Machine-based Cleaning

* Frénay and Verleysen: Classification in the Presence of Label Noise: A Survey. IEEE Trans. Neural Netw. Learning Syst. 2014

Cleaning Noisy Label

Existing Work*

- No Cleaning
- Machine-based Cleaning

Our Solution

- Oracle-based Cleaning

* Frénay and Verleysen: Classification in the Presence of Label Noise: A Survey. IEEE Trans. Neural Netw. Learning Syst. 2014

TARS [named after an intelligent robot in the movie *interstellar*]

Label Cleaning Advisor for Crowdsourced Noisy Labels

Mohamad Dolatshah

Mathew Teoh

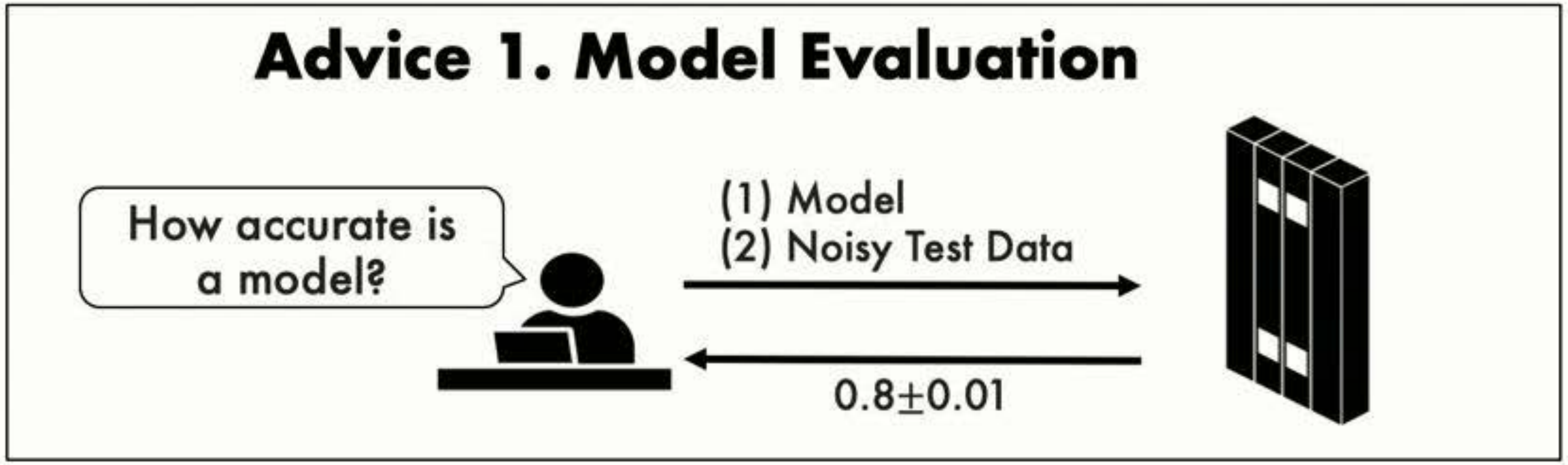
Jiannan Wang

Jian Pei

Dolatshah et al. Cleaning Crowdsourced Labels Using Oracles For Statistical Classification. **PVLDB 2019**

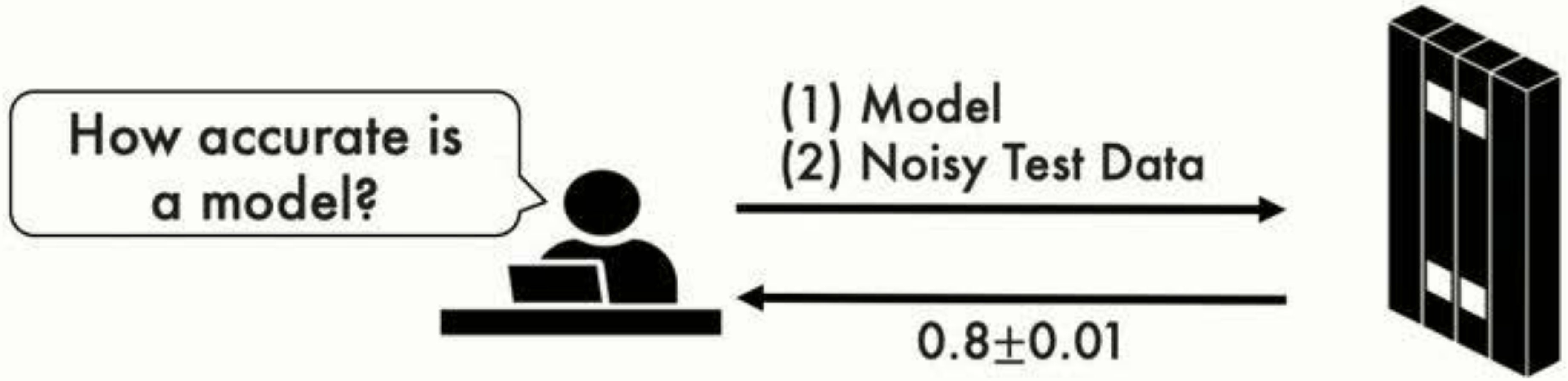
Two Pieces of Advice

Two Pieces of Advice

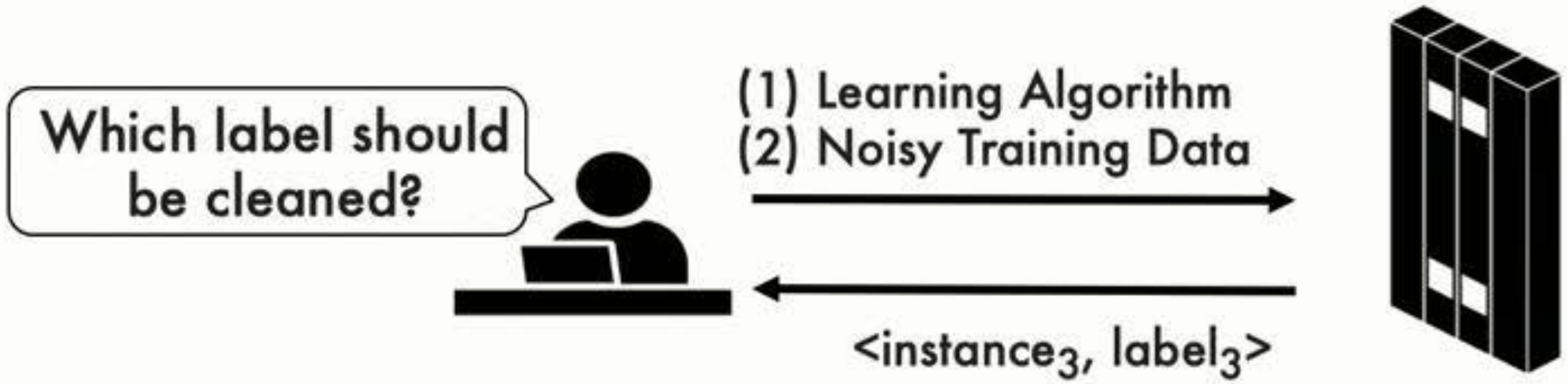


Two Pieces of Advice

Advice 1. Model Evaluation



Advice 2. Cleaning Strategy



Take-away Messages

Take-away Messages

DB community should play an important role in democratizing data preparation for AI

Take-away Messages

DB community should play an important role in democratizing data preparation for AI

We build TARS, a label cleaning advisor to reduce data labeling time for AI

Take-away Messages

DB community should play an important role in democratizing data preparation for AI

We build TARS, a label cleaning advisor to reduce data labeling time for AI

Poster 1: Extracting Highlights from Recorded Live Videos (Changbo)

Poster 2: Explaining ML-embedded SQL Queries (Weiyuan)

INFLUENCE MAXIMIZATION IN MASSIVE GRAPHS

NWDS'2019

Diana Popova
Department of Computer Science
University of Victoria, Canada

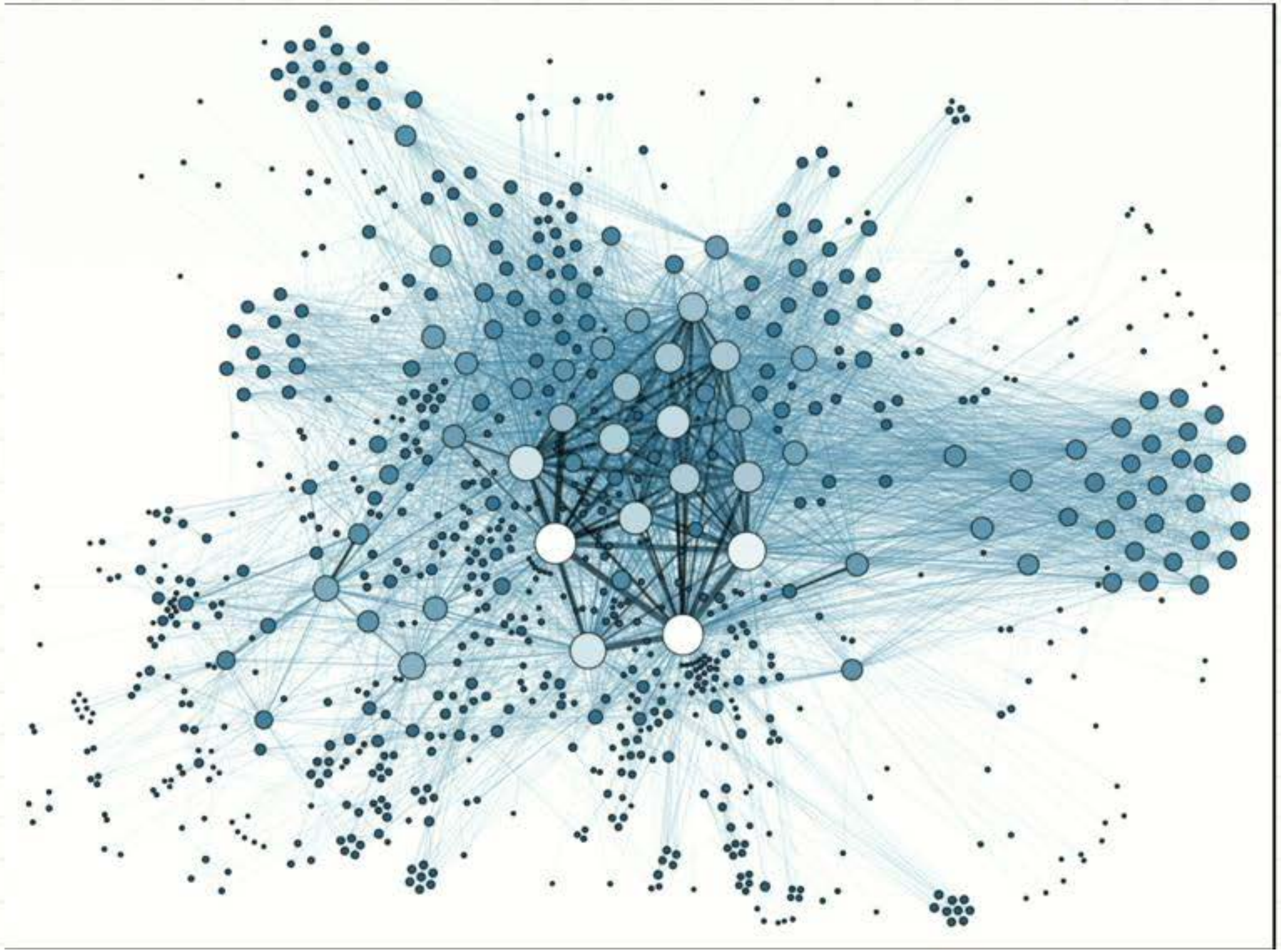
February 2019

Overview

- Influence Discovery in Graphs
- Algorithms Scalability
- Influence Maximization

Influence Discovery

Influence Discovery in Graph



Graph's Incidence List

0	1
0	2
0	3
0	4
0	5
0	6
0	7
0	8
0	9
0	10
0	11
0	12
0	13
0	14
0	15
0	16
0	17
0	18
0	19
0	20
0	21
0	22
0	23
0	24
0	25
0	26
0	27
0	28
0	29
0	30
1	0
1	48
1	53
1	54
1	73
1	88
1	92
1	119
1	126
1	133
1	194
1	236
1	280
1	299
1	315
1	322
1	346
2	0
2	20
2	115
2	116
2	149
2	226
2	312
2	326
2	333
2	343
3	0
3	9
3	25
3	26
3	67
3	72
3	85
3	122
3	142
3	170
3	188
3	200
3	228
3	274
3	280
3	283
3	323
4	0
4	78
4	152
4	181
4	195
4	218
4	273
4	275
4	306
4	328
5	0
5	87
5	122
5	156
5	158
5	169
5	180
5	187
5	204
5	213
5	235
5	315
5	316
6	0
6	89
6	95
6	147
6	219
6	319
7	0
7	22
7	31
7	38
7	65
7	87
7	103
7	129
7	38
7	65
7	87
7	103
7	129
7	136
7	168
7	213
7	246
7	291
7	304
7	308
7	315
7	322
7	339
7	340
7	347
8	0
8	91
8	110
8	193
8	201
8	245
8	259
8	264
9	0
9	3
9	21
9	25
9	26
9	30
9	56
9	66
9	67
9	69
9	72
9	75
9	79
9	85
9	105
9	113
9	119
9	122
9	128
9	133
9	134
9	141
9	142
9	148
9	156
9	161
9	169
9	170
9	176
9	185
9	186
9	188
9	199
9	200
9	203
9	224
9	231
9	232
9	252
9	258

0	1
0	2
0	3
0	4
0	5
0	6
0	7
0	8
0	9
0	10
0	11
0	12
0	13
0	14
0	15
0	16
0	17
0	18
0	19
0	20
0	21
0	22
0	23
0	24
0	25
0	26
0	27
0	28
0	29
0	30

1	0
1	48
1	53
1	54
1	73
1	88
1	92
1	119
1	126
1	133
1	194
1	236
1	280
1	299
1	315
1	322
1	346
2	0
2	20
2	115
2	116
2	149
2	226
2	312
2	326
2	333
2	343

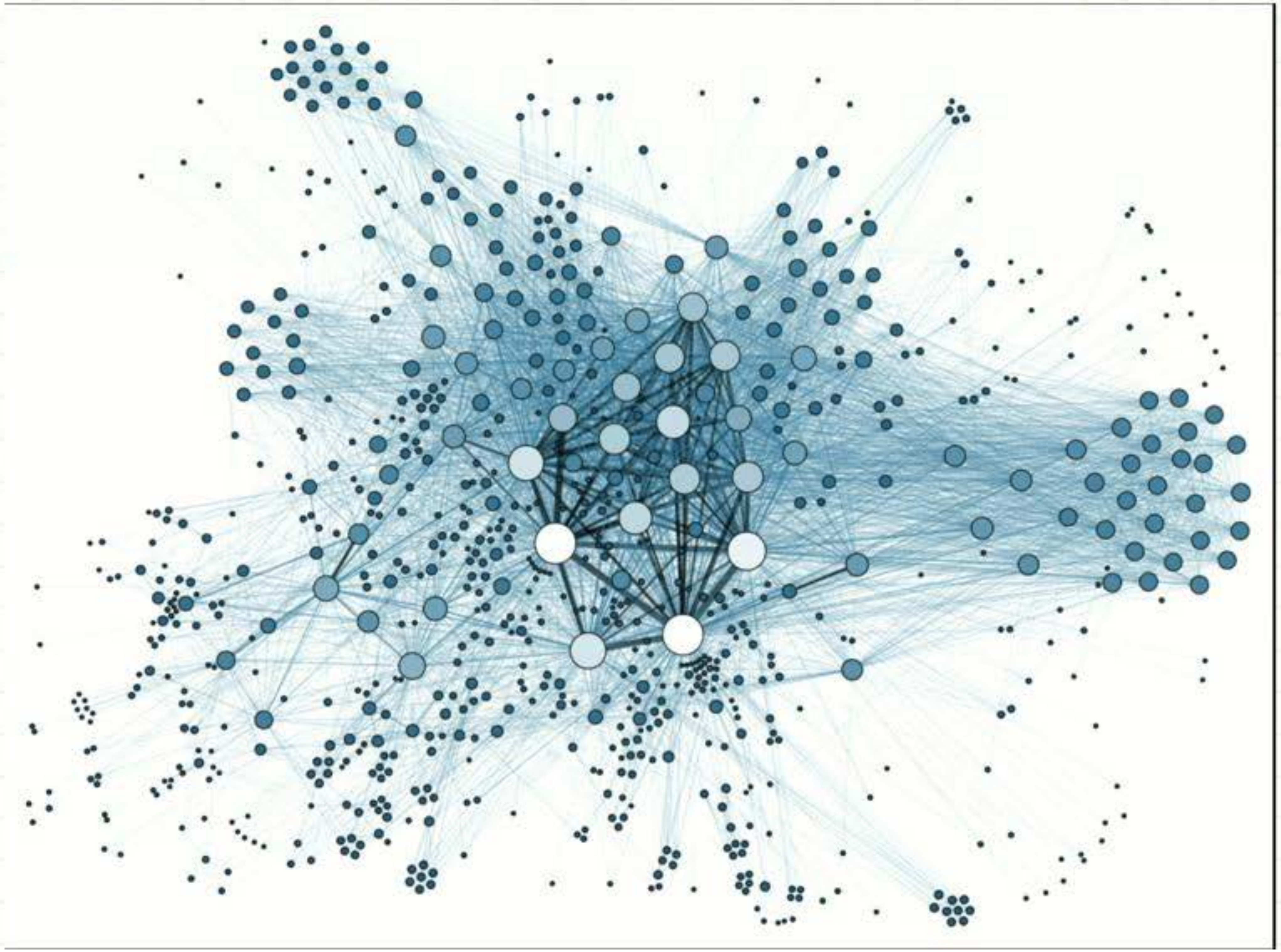
3	0
3	9
3	25
3	26
3	67
3	72
3	85
3	122
3	142
3	170
3	188
3	200
3	228
3	274
3	280
3	283
3	323
4	0
4	78
4	152
4	181
4	195
4	218
4	273
4	275
4	306
4	328

5	0
5	87
5	122
5	156
5	158
5	169
5	180
5	187
5	204
5	213
5	235
5	315
5	316
6	0
6	89
6	95
6	147
6	219
6	319
7	0
7	22
7	31
7	38
7	65
7	87
7	103
7	129

7	38
7	65
7	87
7	103
7	129
7	136
7	168
7	213
7	246
7	291
7	304
7	308
7	315
7	322
7	339
7	340
7	347
8	0
8	91
8	110
8	193
8	201
8	245
8	259
8	264

9	0
9	3
9	21
9	25
9	26
9	30
9	56
9	66
9	67
9	69
9	72
9	75
9	79
9	85
9	105
9	113
9	119
9	122
9	128
9	133
9	134
9	141
9	142
9	148
9	156
9	161
9	169
9	170
9	176
9	185
9	186
9	188
9	199
9	200
9	203
9	224
9	231
9	232
9	252
9	258

Influence Discovery in Graph



Graph's Incidence List

0	1
0	2
0	3
0	4
0	5
0	6
0	7
0	8
0	9
0	10
0	11
0	12
0	13
0	14
0	15
0	16
0	17
0	18
0	19
0	20
0	21
0	22
0	23
0	24
0	25
0	26
0	27
0	28
0	29
0	30

1	0
1	48
1	53
1	54
1	73
1	88
1	92
1	119
1	126
1	133
1	194
1	236
1	280
1	299
1	315
1	322
1	346
2	0
2	20
2	115
2	116
2	149
2	226
2	312
2	326
2	333
2	343

3	0
3	9
3	25
3	26
3	67
3	72
3	85
3	122
3	142
3	170
3	188
3	200
3	228
3	274
3	280
3	283
3	323
4	0
4	78
4	152
4	181
4	195
4	218
4	273
4	275
4	306
4	328

5	0
5	87
5	122
5	156
5	158
5	169
5	180
5	187
5	204
5	213
5	235
5	315
5	316
6	0
6	89
6	95
6	147
6	219
6	319
7	0
7	22
7	31
7	38
7	65
7	87
7	103
7	129

7	38
7	65
7	87
7	103
7	129
7	136
7	168
7	213
7	246
7	291
7	304
7	308
7	315
7	322
7	339
7	340
7	347
8	0
8	91
8	110
8	193
8	201
8	245
8	259
8	264

9	0
9	3
9	21
9	25
9	26
9	30
9	56
9	66
9	67
9	69
9	72
9	75
9	79
9	85
9	105
9	113
9	119
9	122
9	128
9	133
9	134
9	141
9	142
9	148
9	156
9	161
9	169
9	170
9	176
9	185
9	186
9	188
9	199
9	200
9	203
9	224
9	231
9	232
9	252
9	258

Algorithms Scalability

Scalability

Fair comparison:

- Same graph

Scalability

Fair comparison:

- Same graph
- Max graph size on the same machine

Scalability

Fair comparison:

- Same graph
- Max graph size on the same machine

Tests of eleven different IM algorithms by Arora et al.

Scalability

Fair comparison:

- Same graph
- Max graph size on the same machine

Tests of eleven different IM algorithms by Arora et al.

A. Arora, S. Galhotra, and S. Ranu. Debunking the myths of influence maximization: An in-depth benchmarking study. In *Proceedings of the 43rd ACM SIGMOD International Conference on Management of Data*, pages 651–666, 2017.

Scalability

Evaluation:

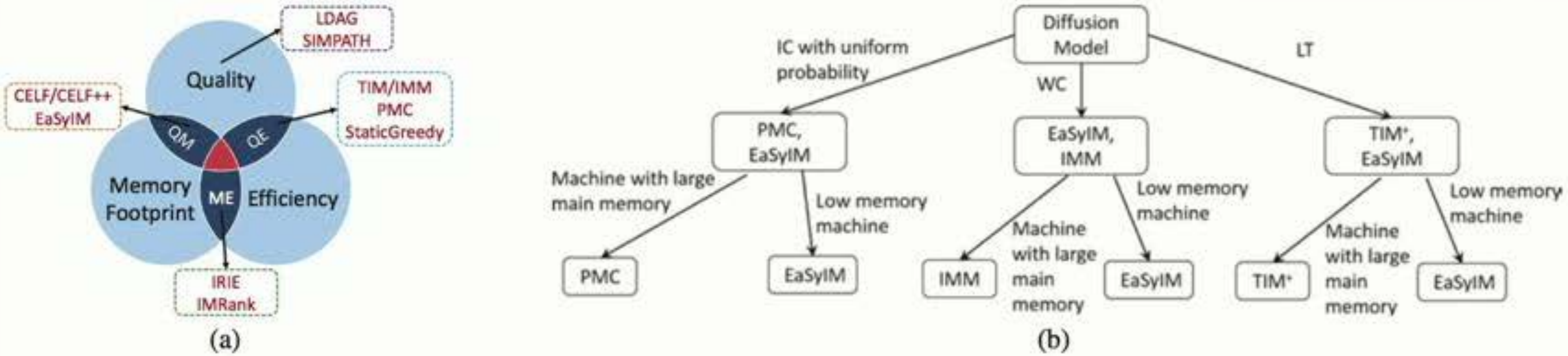


Figure 11: **(a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.**

Scalability

Evaluation:

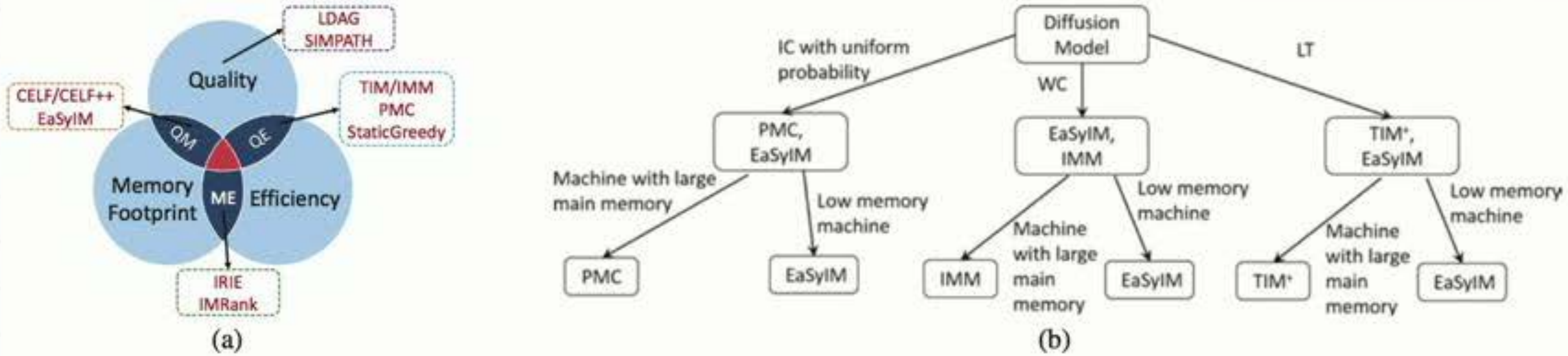


Figure 11: **(a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.**

Scalability

Evaluation:

- Quality

- Time and Space

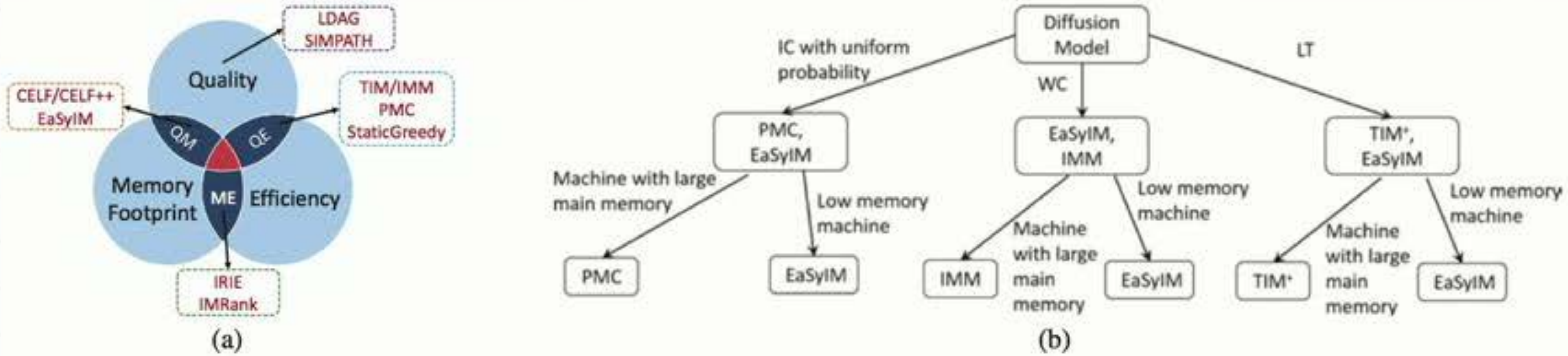


Figure 11: **(a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.**

Scalability

Evaluation:

- Quality
- Time and Space

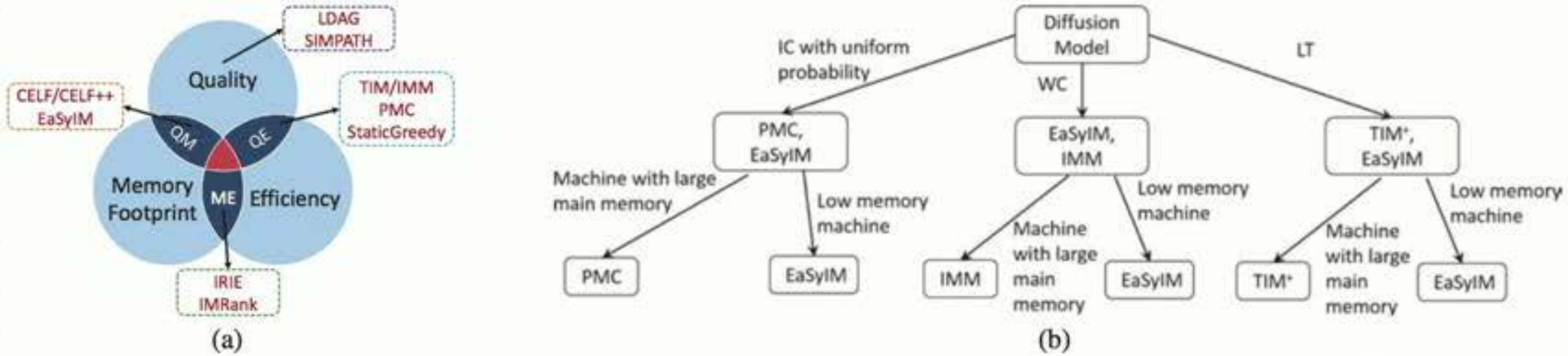


Figure 11: **(a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.**

Previous Work

Influence Maximization

Previous Work

- Kempe, Kleinberg, and Tardos, 2003:
  - *Independent Cascade (IC)* model of influence propagation.
  - *Greedy* algorithm for finding the best seed set for a given *k* (number of seeds).
  - Monte Carlo simulations, randomized selection of edges, and averaging over coverage.

- Borgs, Brautbar, Chayes, and Lucier, 2014:
  - Reverse Influence Sampling: randomized *sketching* of the transposed graph.
  - Theoretical guarantees: approximation factor of $(1-1/e-\varepsilon)$, for any $\varepsilon > 0$, with 60% confidence.

Influence Maximization (IM)

- Node *Influence* – the number of graph nodes reachable from a given node under a certain model.
- *Information propagation* is a process of spreading information from node to node using edges.
- **IM Problem**: find a given number of *seed* nodes, such that the information would spread far and wide. Class NP.

The graph is probabilistic, and the result of influence maximization is an approximation to optimal. Class P.

Our approach:

- Data Structures for small memory footprint

Data Structures for Efficient Computation of Influence Maximization

Reverse Influence Sampling (**RIS**) idea:

- find the nodes that *would* influence a randomly selected node;
- do it multiple times;
- if a node appears often as *influencer*, it is a good candidate for a seed.

Our implementation:

- Webgraph format for the input graph.
- Instead of list of lists, we use flat arrays and boolean arrays (bitset).
- Java 8 parallel streams and lambda expressions.
- Lazy Greedy technique.

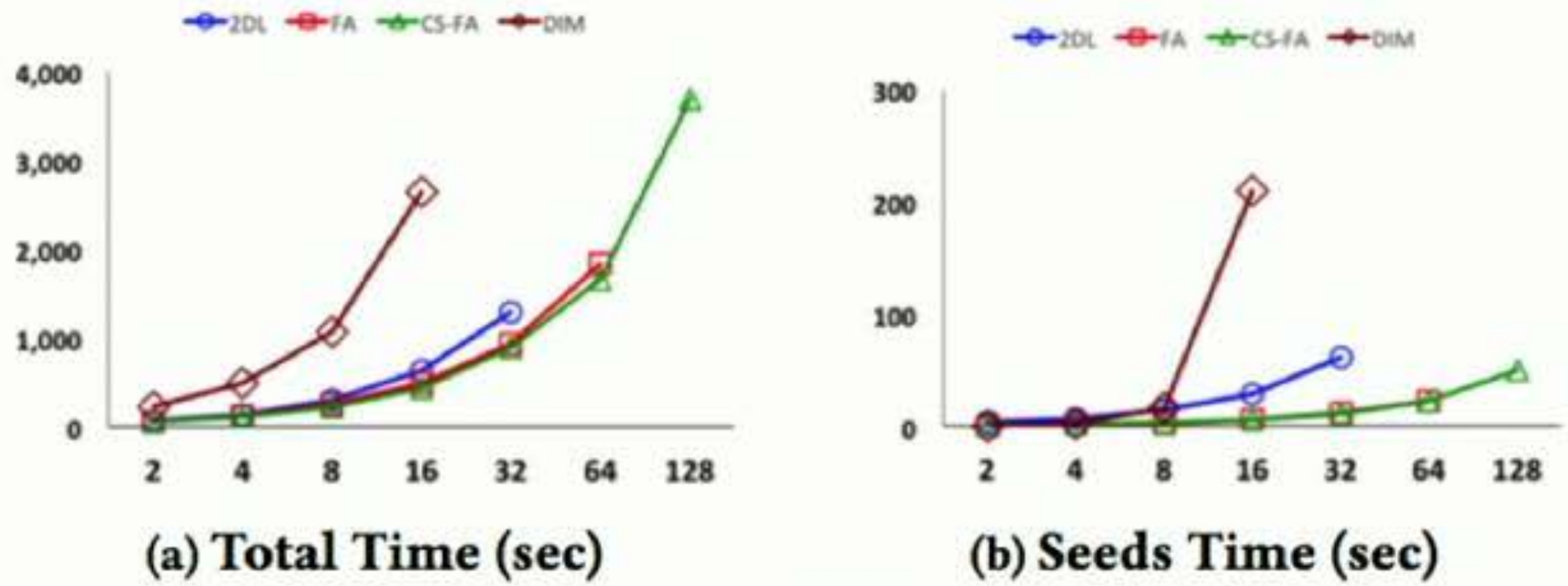


Figure 1: Processing time for cnr-2000; k=10, varying β.

Comparison to DIM

Webgraph format for storing intermediate results

node ID	sketch numbers
0	0, 43, 240, 329, 432, 1000
	• • •
i	240, 329, 1000
	• • •
l	0, 240, 1784, 2567, 2568
	• • •
n-1	1, 12, 13, 248, 329, 765, 1087, 1589

sketch numbers	node ID
0	0, … l, …
	• • •
43	0, …
	• • •
240	0, …, i, …, l, …
	• • •
sk-1	1, i, 248, 329, l, 1589

Left: hypergraph as Borgs *et al.* described in RIS.
Right: hypergraph as built by NoSingles.

NoSingles: a Space-Efficient Algorithm for Influence Maximization

Idea: Do not store sketches containing only one node.

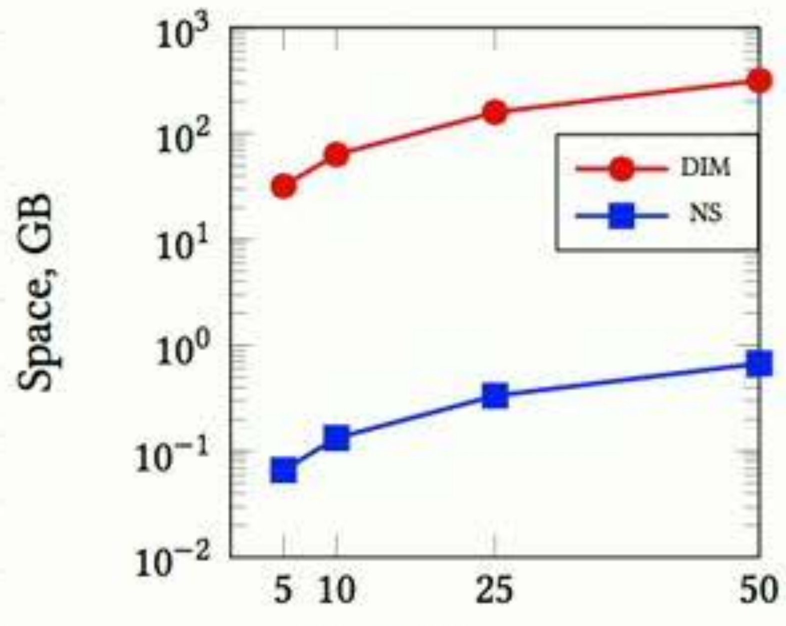
NS hypergraph and *node_count* array are stored on disk.

Dataset	min	max	**median**	**1node sketches**
uk100K	1	2925	**1**	**91%**
cnr2000	1	794	**1**	**96%**
eu2005	1	858	**1**	**90%**
ljounal2008	1	78018	**1**	**90%**
arabic2005	1	20708	**1**	**93%**

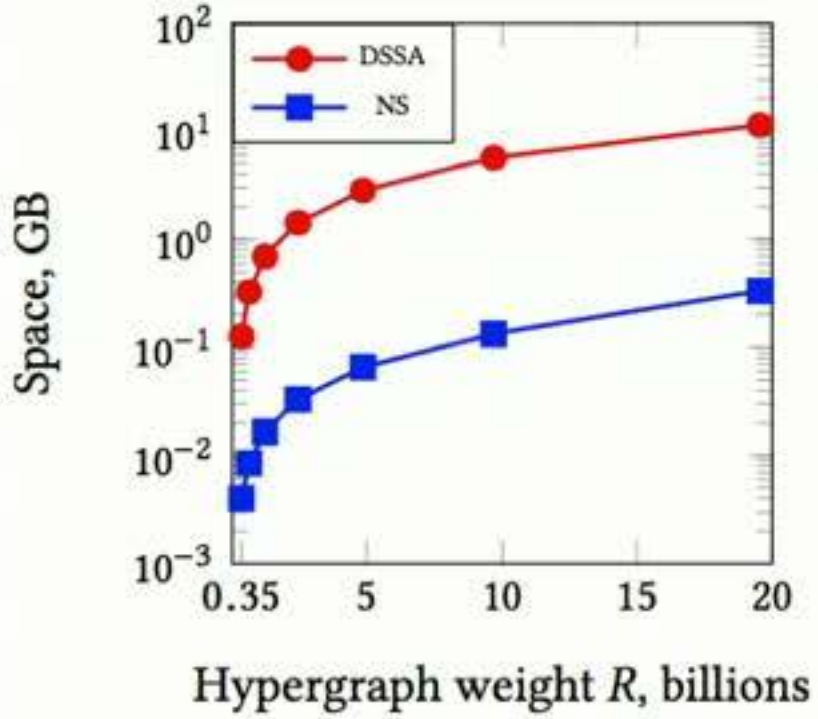
Table 6.4: Sketch Cardinality Statistics ($p = 0.01$).

NoSingles: a Space-Efficient Algorithm for Influence Maximization

CPU=Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, running OS CentOS, with RAM=1TB; 48 logical cores.



(1) $p = 0.1$



(1) $p = 0.1$

Comparison to two leading IM algorithms, DIM and D-SSA, shows three orders of magnitude savings in required main memory.

NoSingles: a Space-Efficient Algorithm for Influence Maximization

NoSingles can successfully run on a consumer-grade laptop for large graphs.

Borgs' et al. formula from Theorem 3.1

Dataset	n	m	ϵ	p	k
arabic2005	22.7 M	0.63 B	0.2	0.001	5

Table 6: Parameters.

R	sk, total	sk, saved	H size, edges
6.4 T	2.5 B	36.3 M	2.7 B

Table 7: Intermediate results.

H **space**	H time	Seeds time	accuracy	confidence
1 GB	90.5 hrs	136.5 sec	0.43	0.6

Table 8: Results.

CutTheTail: a Space-Efficient Heuristic Algorithm for Influence Maximization

Idea

CutTheTail1: Do not store sketches containing only nodes with low out-degree.

CutTheTail2: Do not store short sketches.

Dataset	n	m	type
WordAsn	10.6K	72K	association, directed
Caida	65.5K	106.7K	social, directed
FB	4K	176K	social, undirected
EnronD	69K	275K	e-mails, directed
Enron	36.7K	368K	e-mails, undirected
Deezer	54.6K	996K	social, undirected
DBLP2010	326 K	1.6 M	collaboration, undirected
UK100K	100 K	3 M	web, directed
CNR2000	326 K	3.2 M	web, directed
DBLP2011	986K	6.7M	collaboration, undirected
Arabic2005	23M	640M	web, directed

Table 1: Test datasets ordered by m.

Confidence test: log(n) runs, for (1 – 1/n) confidence.

Statistics on saved sketches: CTT2 can save only 0.01% sketches.

Monte Carlo simulation of seeds quality: TopDegree varies from 33% of NS spread to 100% of NS spread, but never better than NS.

Conclusion

- Choice of Data Structure proved to be instrumental in raising the scalability of graph analytics.

- Focus on space complexity allowed to design and implement smart algorithms processing large graphs on a consumer-grade laptop.

Integrity Constraints Revisited: From Exact to Approximate Implication

Batya Kenig Dan Suciu
University of Washington

Problem Statement (Informal)

- Fix a *single* relation instance R.
- Integrity Constraints: FDs and MVDs only
  - Hard: either $R \models \tau$ or $R \not\models \tau$.
  - Soft: R satisfies τ to some degree.
- Relaxing exact implications:
  - Suppose $\Sigma \models \tau$ holds for hard constraints.
  - If the constraints in Σ hold to a large extent, to what extent does τ?
- Lots of applications.
  - Mining of approximate integrity constraints in a DB instance (Chu et al. 2014, Giannela and Robsertson 2004, Kruse and Naumann 2018)
  - Data cleaning (Ilyas and Chu 2015)
  - Learning structure of Probabilistic Graphical Models

Outline

- Key Concepts & Ideas
- Main Results

Outline

- Key Concepts & Ideas
- Main Results

Conditional Independence Statements

- We consider discrete probability distributions.
- X is a set of random variables.
- A,B,C,… are subsets of X.
- $A \perp B | C \Leftrightarrow P(A,B|C)=P(A|C)P(B|C)$.
- $A \perp B | C$ is *saturated* if $X=A \cup B \cup C$.
- $A \perp B | C$ is *marginal* if $C=\emptyset$.
- Σ is a set of CI statements, τ is a single CI statement.
- An important concept in probabilistic modeling and reasoning.

Definition: Probabilistic CI Implication Problem

Let Σ be a set of CI statements and let τ be a CI statement. We say that Σ *implies* τ, denoted $\Sigma \models \tau$, if every probability distribution that satisfies the CI statements in Σ also satisfies the CI statement τ.

Definition: Probabilistic CI Implication Problem

Let Σ be a set of CI statements and let τ be a CI statement. We say that Σ *implies* τ, denoted $\Sigma \models \tau$, if every probability distribution that satisfies the CI statements in Σ also satisfies the CI statement τ.

The semi-graphoid axioms, Pearl 1988

$A \perp \emptyset \| C$	Triviality
$A \perp B \| C \rightarrow B \perp A \| C$	Symmetry
$A \perp BD \| C \rightarrow A \perp D \| C$	Decomposition
$A \perp B \| CD \wedge A \perp D \| C \rightarrow A \perp BD \| C$	Contraction
$A \perp BD \| C \rightarrow A \perp B \| CD$	Weak Union

Definition: Probabilistic CI Implication Problem

Let Σ be a set of CI statements and let τ be a CI statement. We say that Σ *implies* τ, denoted $\Sigma \vDash \tau$, if every probability distribution that satisfies the CI statements in Σ also satisfies the CI statement τ.

The semi-graphoid axioms, Pearl 1988

$A \perp \emptyset \mid C$	Triviality
$A \perp B \mid C \rightarrow B \perp A \mid C$	Symmetry
$A \perp BD \mid C \rightarrow A \perp D \mid C$	Decomposition
$A \perp B \mid CD \wedge A \perp D \mid C \rightarrow A \perp BD \mid C$	Contraction
$A \perp BD \mid C \rightarrow A \perp B \mid CD$	Weak Union

Theorem (Geiger+Pearl 1993)

Axioms are (1) Sound, and (2) Complete for Saturated and Marginal CIs.

Review: FD and MVD

Functional Dependency (FD)

- R satisfies the FD A→B if $\forall t_1,t_2 \in R$, $t_1.A=t_2.A \Rightarrow t_1.B=t_2.B$

(Embedded) Multivalued Dependency:

- R satisfies the EMVD A→(B|C) if $\Pi_{ABC}(R)=\Pi_{AB}(R)\bowtie\Pi_{AC}(R)$
- MVD: A ↠ B is an EMVD A → (B|C) where ABC=all attrs

A	B	C
1	1	1
1	1	2
1	2	1
1	2	2
2	2	2

→(B|C)
A→(B|C)

Implication:

- Armstrong's axioms, Beeri's algorithm

Review: FD and MVD

Functional Dependency (FD)

- R satisfies the FD $A \rightarrow B$ if $\forall t_1, t_2 \in R$, $t_1.A = t_2.A \Rightarrow t_1.B = t_2.B$

(Embedded) Multivalued Dependency:

- R satisfies the EMVD $A \rightarrow (B|C)$ if $\Pi_{ABC}(R) = \Pi_{AB}(R) \bowtie \Pi_{AC}(R)$
- MVD: $A \twoheadrightarrow B$ is an EMVD $A \rightarrow (B|C)$ where ABC=all attrs

Implication:

- Armstrong's axioms, Beeri's algorithm

A	B	C
1	1	1
1	1	2
1	2	1
1	2	2
2	2	2

$A \rightarrow (B|C)$

Review: FD and MVD

Functional Dependency (FD)

- R satisfies the FD $A \rightarrow B$ if $\forall t_1, t_2 \in R$, $t_1.A = t_2.A \Rightarrow t_1.B = t_2.B$

(Embedded) Multivalued Dependency:

- R satisfies the EMVD $A \rightarrow (B|C)$ if $\Pi_{ABC}(R) = \Pi_{AB}(R) \bowtie \Pi_{AC}(R)$
- MVD: $A \twoheadrightarrow B$ is an EMVD $A \rightarrow (B|C)$ where ABC=all attrs

Implication:

- Armstrong's axioms, Beeri's algorithm

A	B	C
1	1	1
1	1	2
1	2	1
1	2	2
2	2	2

$A \rightarrow (B|C)$

Between Integrity Constraints and CIs

Between Integrity Constraints and CIs

The Empirical Distribution of relation R

The probability space of the support of R, where each tuple t∈R is sampled with probability $^1/_N$.

Between Integrity Constraints and CIs

The Empirical Distribution of relation R

The probability space of the support of R, where each tuple $t \in R$ is sampled with probability $^1/_N$.

Fix R, and its empirical distribution.

- $A \twoheadrightarrow B$ iff $B \perp C | A$ where ABC=all vars.
- Fails for EMVD
  - $\emptyset \twoheadrightarrow B | C$, but $\neg(B \perp C)$
  - $p(C = 1) = {}^2/_5$
  - $p(C = 1 | B = 1) = {}^1/_2$

A	B	C	
1	1	1	1/5
1	1	2	1/5
1	2	1	1/5
1	2	2	1/5
2	2	2	1/5

Review: Information Theory

- X = r.v. with n outcomes; its entropy is:

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

- The conditional entropy is:

$$H(Y|X) = H(XY) - H(X)$$

- The conditional mutual information is:

$$I(X;Y|Z) = H(XZ) + H(YZ) - H(XYZ) - H(Z)$$

Soft Constraints

- For CIs: $X \perp Y | Z \Leftrightarrow I(X;Y|Z)=0$.
- We will use $I(X;Y|Z)$ to quantify the *degree of independence* of X, Y given Z.

Soft Constraints

- For CIs: $X \perp Y | Z \Leftrightarrow I(X;Y|Z)=0$.
- We will use $I(X;Y|Z)$ to quantify the *degree of independence* of X, Y given Z.

Theorem (Lee 1987)	
FDs	$X \rightarrow Y$ iff $H(Y|X)=0$
MVDs	$X \twoheadrightarrow Y|Z$ iff $I(Z;Y|X)=0$

Known Impossibility Results

- Implication problem for EMVDs is undecidable (Herrmann 2006)
- Implication problem for conditional independence is not finitely axiomatizable (Studeny 1990)

Outline

- Key Concepts & Ideas
- Main Results

The Relaxation Problem

Fix a set of CIs $\Sigma=\{\sigma_1,\ldots,\sigma_m\}$, and a CI $\tau\notin\Sigma$.

Assume*: $\Sigma\models\tau$

Problem: find a bound on τ in terms of Σ.

Relaxation: $\tau\leq\sum_i \lambda_i\sigma_i$ where $\lambda_i\geq 0$

Unit relaxation: $\tau\leq\sum_i \sigma_i$

* e.g. using Armstrong's axioms, Beeri's algorithm, or semi-graphoid axioms

FDs Admit Unit Relaxation

Theorem

The following are equivalent:

- $X_1 \rightarrow Y_1, \ldots, Xm \rightarrow Ym \models X \rightarrow Y$
- $H(Y|X) \leq H(Y_1|X_1) + \ldots + H(Ym|Xm)$

Example: $AB \rightarrow C, AD \rightarrow E, CE \rightarrow F \models ABD \rightarrow F$

Therefore this is a valid information-theoretic inequality:

$$H(F|ABD) \leq H(C|AB) + H(E|AD) + H(F|CE)$$

CI's Do Not Admit Relaxation!

Theorem (Kaced&Romashchenko 2013)

$(C \perp D|A)$, $(C \perp D|B)$, $(A \perp B)$, $(B \perp C|D)$ $\vDash$ $C \perp D$
However, for any $\lambda_1, \ldots, \lambda_4 \geq 0$ there exists a distribution s.t.
$I(C;D) > \lambda_1 I(C;D|A) + \lambda_2 I(C;D|B) + \lambda_3 I(A;B) + \lambda_4 I(B;C|D)$.

However, we can relax "in the limit"

Theorem

If the exact implication $\Sigma \vDash \tau$ holds, then for any $\varepsilon > 0$ there exist $\lambda_i \geq 0$ such that:

$$\tau \leq \sum_i \lambda_i \sigma_i + \varepsilon H(\text{all-variables})$$

CI's Do Not Admit Relaxation!

Theorem (Kaced&Romashchenko 2013)

$(C \perp D|A)$, $(C \perp D|B)$, $(A \perp B)$, $(B \perp C|D)$ $\vDash$ $C \perp D$

However, for any $\lambda_1, \ldots, \lambda_4 \geq 0$ there exists a distribution s.t.

$I(C;D) > \lambda_1 I(C;D|A) + \lambda_2 I(C;D|B) + \lambda_3 I(A;B) + \lambda_4 I(B;C|D)$.

However, we can relax "in the limit"

Theorem

If the exact implication $\Sigma \vDash \tau$ holds, then for any $\varepsilon > 0$ there exist $\lambda_i \geq 0$ such that:

$$\tau \leq \sum_i \lambda_i \sigma_i + \varepsilon H(\text{all-variables})$$

Saturated CIs

Disjoint CIs

Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are *disjoint* if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$.

Saturated CIs

Disjoint CIs

Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are *disjoint* if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$.

Note: All semi-graphoid axioms are disjoint.

Saturated CIs

Disjoint CIs

Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are *disjoint* if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$.

Note: All semi-graphoid axioms are disjoint.

Theorem

If Σ is a set of disjoint CIs, and τ is saturated, then the implication $\Sigma \models \tau$ (by the Shannon inequalites) admits unit relaxation: $\tau \leq \sum_i \sigma_i$.

Saturated CIs

Disjoint CIs

Two CIs $X \perp Y \mid Z$ and $A \perp B \mid C$ are *disjoint* if at least one of the following is non-empty: (1) $X \cap C$ (2) $Y \cap C$ (3) $Z \cap A$ (4) $Z \cap B$.

Note: All semi-graphoid axioms are disjoint.

Theorem

If Σ is a set of disjoint CIs, and τ is saturated, then the implication $\Sigma \models \tau$ (by the Shannon inequalites) admits unit relaxation: $\tau \leq \sum_i \sigma_i$.

Example: *Contraction Axiom* in semi-graphoids:

$$X \perp Y|Z \quad \& \quad X \perp W \mid YZ \quad \models \quad X \perp YW \mid Z$$

Relaxes to:

$$I(X;YW \mid Z) \leq I(X;Y|Z) + I(X;W|YZ) \quad \text{// in fact, equality}$$

Conclusions

- The connection between constraints and information theory has been known for a long time.
- The *relaxation problem* appears to be new.
- Great practical importance: real data satisfies constraints only approximatively, need to relax.
- Open problems: bound on the coefficients λ_i in various settings.

Thank You!

Automating Machine Learning Model Building with Clinical Big Data

Gang Luo

Department of Biomedical Informatics and Medical Education

University of Washington

luogang@uw.edu

Challenges of Using Machine Learning for Clinical Predictive Modeling

- Requires many labor-intensive manual iterations and special computing expertise to select among complex algorithms and hyper-parameter values
- Most machine learning models give no explanation of prediction results
  - Explanation is essential for a learning healthcare system

Challenge 1 – Efficient and Automatic Model Selection

- Automatic selection methods for algorithms and hyper-parameter values have been developed
  - to help individuals with little computing expertise perform machine learning
  - but existing methods cannot efficiently handle clinical big data
  - Search can take several days on a data set with a moderate number of rows and attributes
  - Search time is daunting on large data sets

Challenge 1 – Cont.

- To leverage clinical big data, automated approaches appealing to healthcare researchers are needed for selecting algorithms and hyper-parameter values
  - Completely automatic
  - Efficient

Challenge 2: Explaining Prediction Results

- Explanation is essential for clinicians to
  - Trust prediction results
  - Determine appropriate, tailored interventions
    - E.g., provide transportation for patients who live far from their physicians and have difficulty accessing care
  - Defend their decisions in court if sued for medical negligence
  - Formulate new theories or hypotheses for biomedical research

Challenge 2 – Cont.

- Most machine learning models give no explanation of prediction results
  - Most models are complex
- Prediction accuracy and giving explanation of prediction results are frequently two conflicting goals
- Need to achieve both goals simultaneously
  - Explain prediction results without sacrificing prediction accuracy

Outline

- Our approach to address the challenges
  [HISS'15, HISS'16, HISS'17, JMIR-RP'15, JMIR-RP'17]
  - Efficient and automatic model selection

Current Bayesian Optimization Approach

Test multiple combinations of algorithms and hyper-parameter values;

Build a regression model *R* to predict a combination's performance;

While time permits {

    Use *R* to find a promising combination;

    Evaluate the combination's performance;

    Update *R*;

}

Return the combination with the best performance;

Integrity Constraints Revisited: From Exact to Approximate Implication

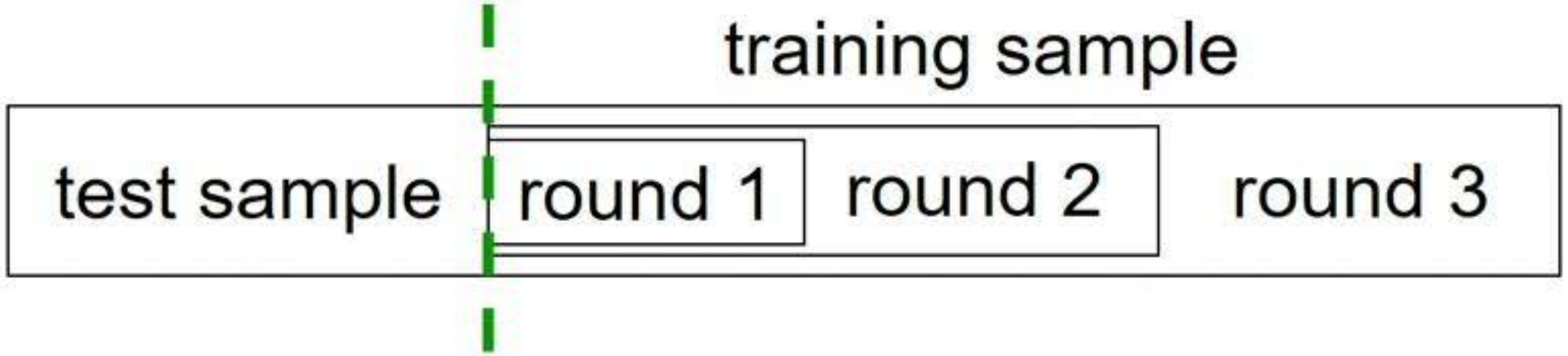
Batya Kenig Dan Suciu
University of Washington

Main Ideas

- Major obstacle: A long time is needed to examine a combination of an algorithm and hyper-parameter values on the entire data set
  - E.g., it takes two days on a modern computer to train a champion ensemble model once on 10K patients with 133 independent variables
  - The entire space of algorithms and hyper-parameter values is extremely large
- Solution: Perform progressive sampling, filtering, and fine-tuning to quickly narrow the search space

Main Ideas – Cont.

- Use progressive sampling to generate a sequence of random samples of the data set, one nested within another

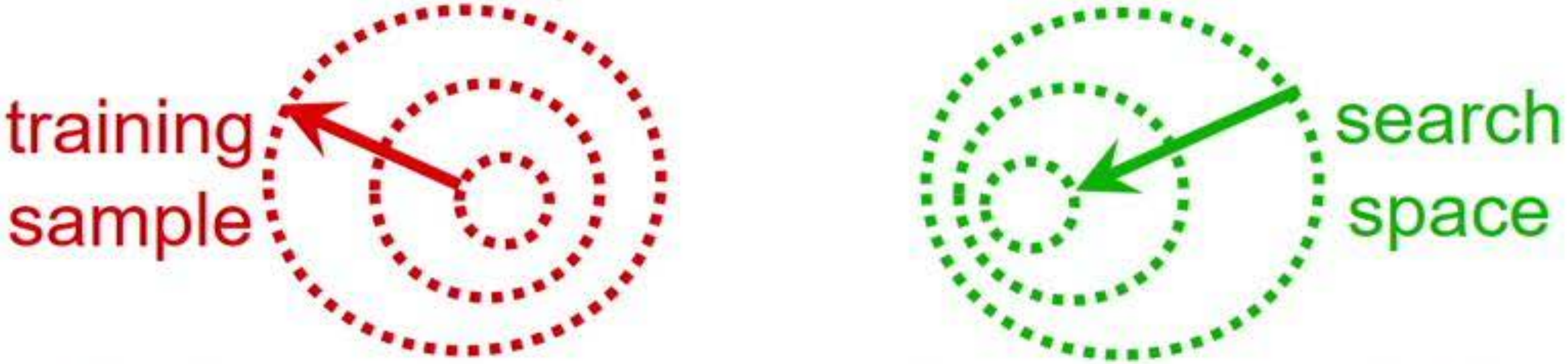


Main Ideas – Cont.

- Conduct inexpensive tests on small samples of the data set to eliminate unpromising algorithms and identify unpromising combinations of hyper-parameter values as early and as much as possible
- Devote more computational resources to fine-tuning promising algorithms and combinations of hyper-parameter values on larger samples of the data set

Main Ideas – Cont.

- The search process is repeated for one or more rounds
- As the sample of the data set expands, the search space shrinks



- In the last round, use (a large part of) the entire data set to find an effective combination of an algorithm and hyper-parameter values

Preliminary Results

- Compared to the state of the art Auto-WEKA automatic selection method on
  - 27 prominent machine learning benchmark data sets
  - A single computer
- On 27 data sets, on average our method
  - Reduces search time by 28 fold
  - Reduces the classification/prediction error rate by 11%

Outline

- Our approach to address the challenges
  - Automatically explain prediction results and suggest tailored interventions

Main Ideas

- A model achieving high accuracy is usually complex and gives no explanation of prediction results
- Challenge: Need to achieve high prediction accuracy as well as explain prediction results
- Key idea: Separate prediction and explanation by using two models concurrently
  - The first model makes predictions and targets maximizing accuracy
  - The second model is rule-based
    - Used to explain the first model's results rather than make predictions

Main Ideas – Cont.

- The rules used in the second model are mined directly from historical data
- Use one or more rules to explain the prediction result for a patient
- Suggest tailored interventions based on the reasons listed in the rules

Some Results

- Test case: Predicting type 2 diabetes diagnosis within the next year
- Electronic medical record data of 10K patients
- Can explain prediction results for 87% of patients who were correctly predicted by a champion machine learning model to have type 2 diabetes diagnosis within the next year

An Example Rule

- The patient had prescriptions of angiotensin-converting-enzyme (ACE) inhibitor in the past three years AND the patient's maximum body mass index recorded in the past three years is ≥35 → the patient will have type 2 diabetes diagnosis within the next year
  - ACE inhibitor is used mainly for treating hypertension and congestive heart failure
  - Obesity, hypertension, and congestive heart failure are known to correlate with type 2 diabetes
- Example intervention: Enroll the patient in a weight loss program

Thank you 

Generating Application-specific In-memory Databases

Cong Yan Alvin Cheung

University of Washington

Database Application With Object-oriented Programming Interfaces

- Developed using object-oriented languages
  - Java, Python, Ruby, …
- Object-relational Mapping (ORM) framework
  - Hibernate, Django, Rails
- Example: web applications

Performance Issues

Performance Issues

Application	# github stars
Discourse (forum)	22k
Lobsters (forum)	1.9k
Gitlab (collaboration)	49k
Redmine (collaboration)	3k
Spree (E-commerce)	17k
ROR Ecommerce	1.7k
Fulcrum (task mgmt)	697
Tracks (task mgmt)	3.5k
Diaspora (social network)	18k
Onebody (social network)	1.2k
Openstreetmap (map)	8k
Fallingfruit (map)	1.1k

Profiling result from 12 open-source web apps:

Performance Issues

Application	# github stars
Discourse (forum)	22k
Lobsters (forum)	1.9k
Gitlab (collaboration)	49k
Redmine (collaboration)	3k
Spree (E-commerce)	17k
ROR Ecommerce	1.7k
Fulcrum (task mgmt)	697
Tracks (task mgmt)	3.5k
Diaspora (social network)	18k
Onebody (social network)	1.2k
Openstreetmap (map)	8k
Fallingfruit (map)	1.1k

Profiling result from 12 open-source web apps:

- 0.1-0.9G of data, 3.3 pages >2sec
- Most slow pages spend >80% on querying data

Why?

- Nested data model
- Predicate involving associated objects
- Program-generated predicate

Chestnut

- Generate app-specific in-memory DB
  - Customize data layout given a workload and a memory budget, minimizing the overall query time
- Specific for database apps using object-oriented programming interface, solves the issues by:
  - Using non-relational storage model
  - Extending index syntax
  - Synthesis-based plan enumeration

1. Nested Data Model

- A mismatch between how the app access data and how data is stored.
  - slow data conversion
  - Example: a chatting app, showing top channels and activities, as well as users for each activity

```
Class Channel:
  has_many: activities => Activity
  …

Class Activity:
  has_one: user => User
  string type
  …

Class User:
  …
```

1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

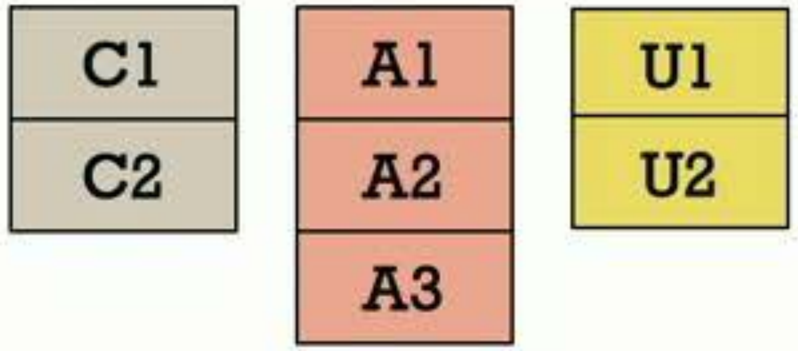
```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)

        SELECT * FROM channel ORDER BY id LIMIT 50;
        SELECT * FROM activity WHERE channel_id IN (…);
        SELECT * FROM user WHERE id IN (…);
```

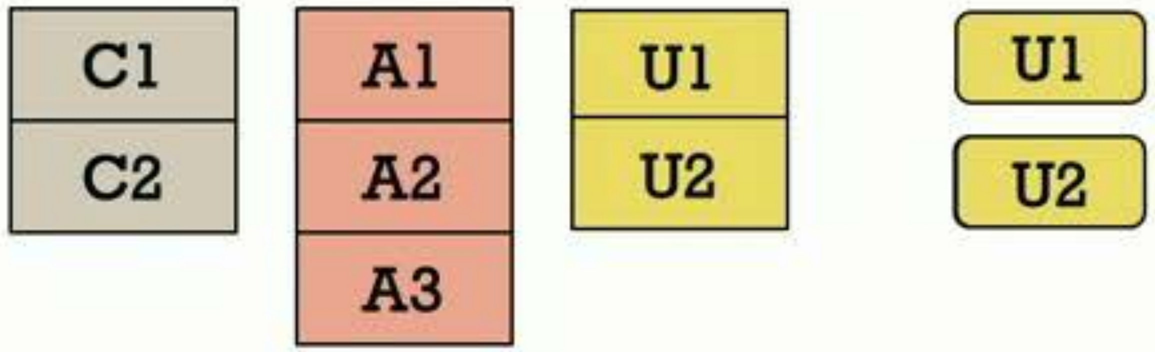


1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)

SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```

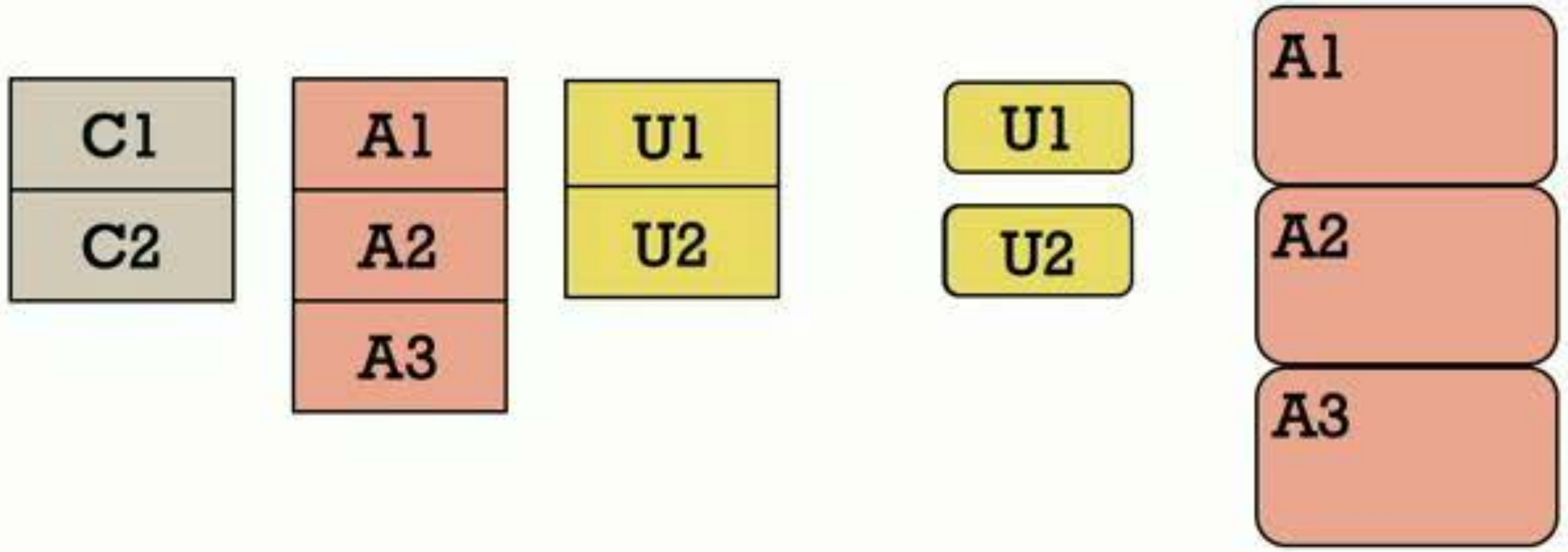


1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```

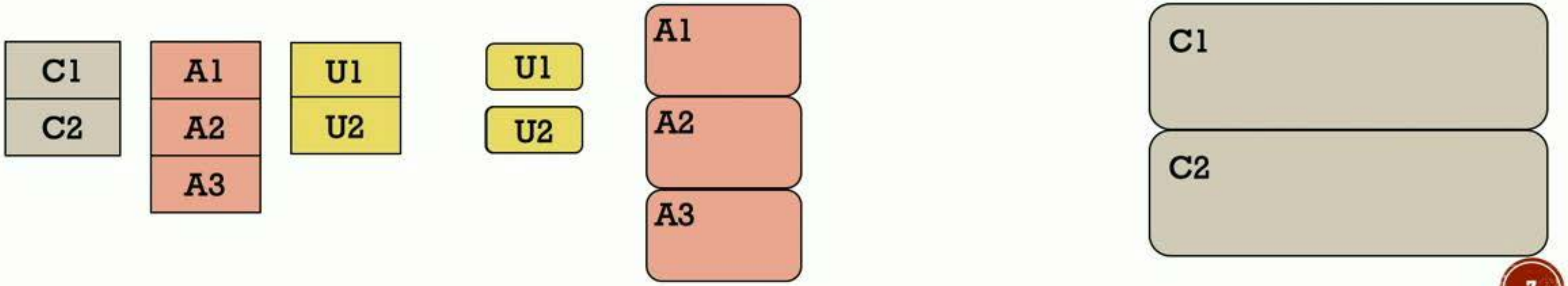


1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```

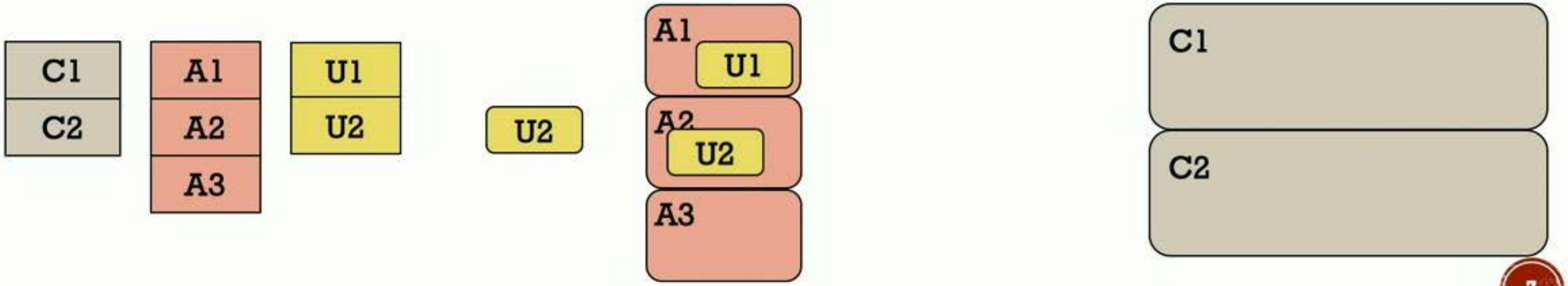


1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```

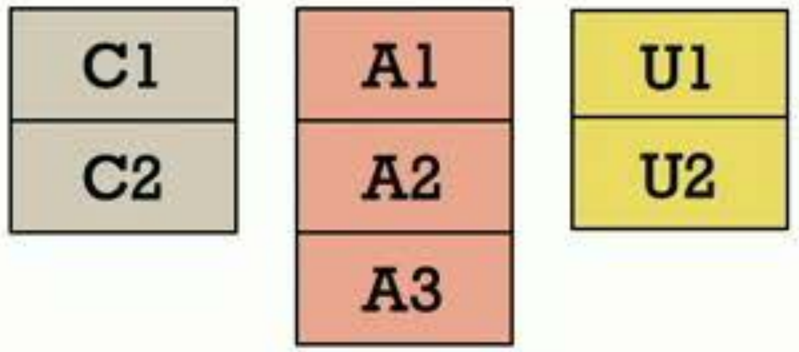


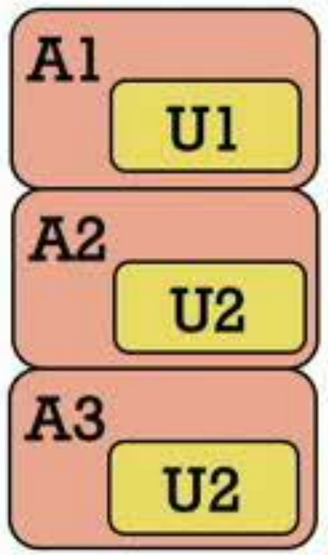
1. Nested Data Model

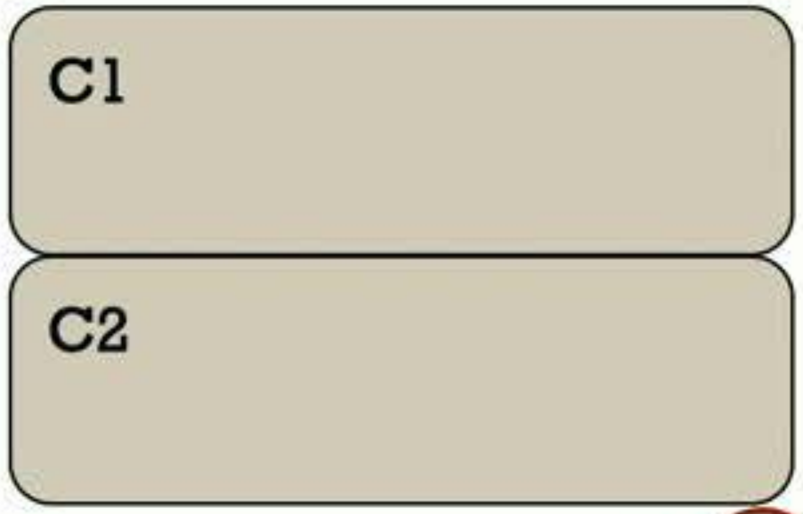
- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```





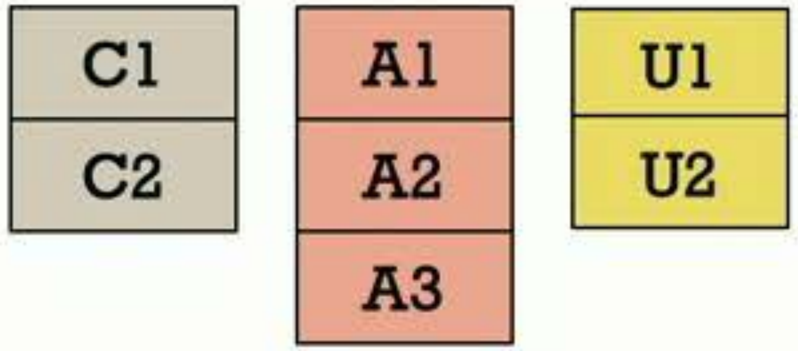


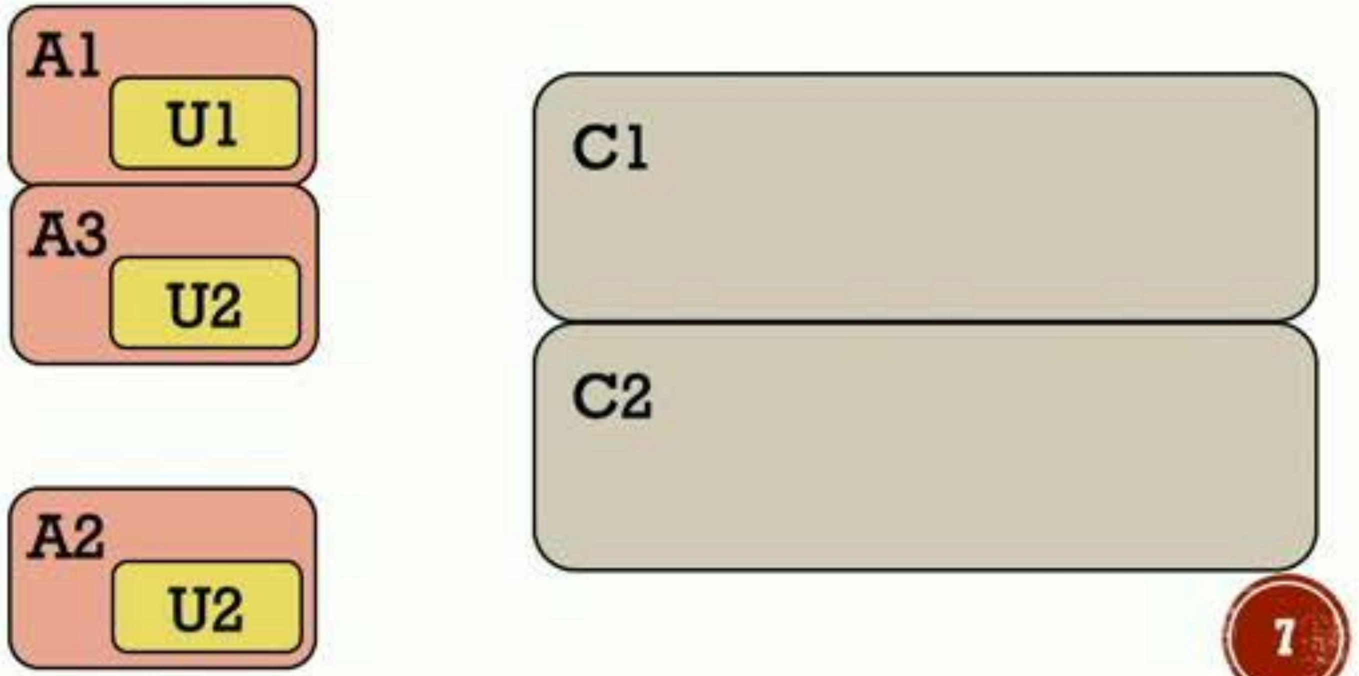
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```



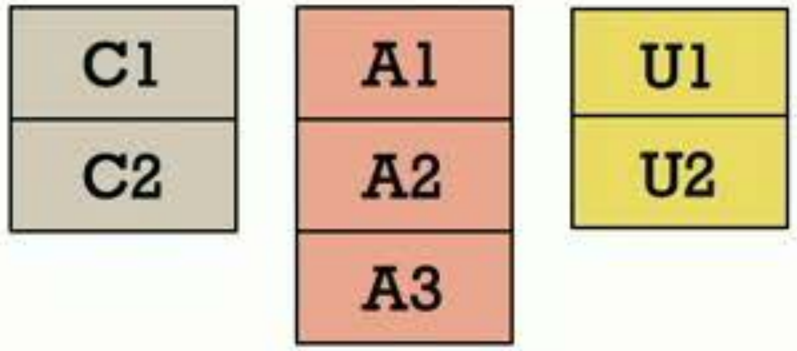


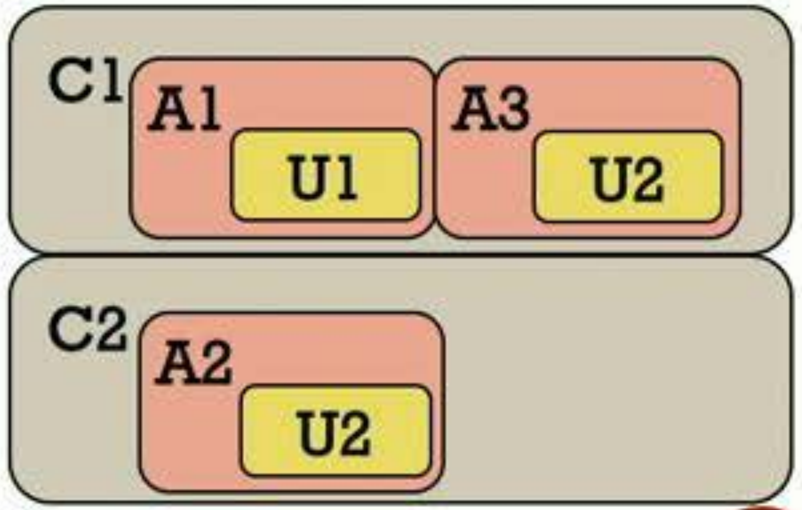
1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```



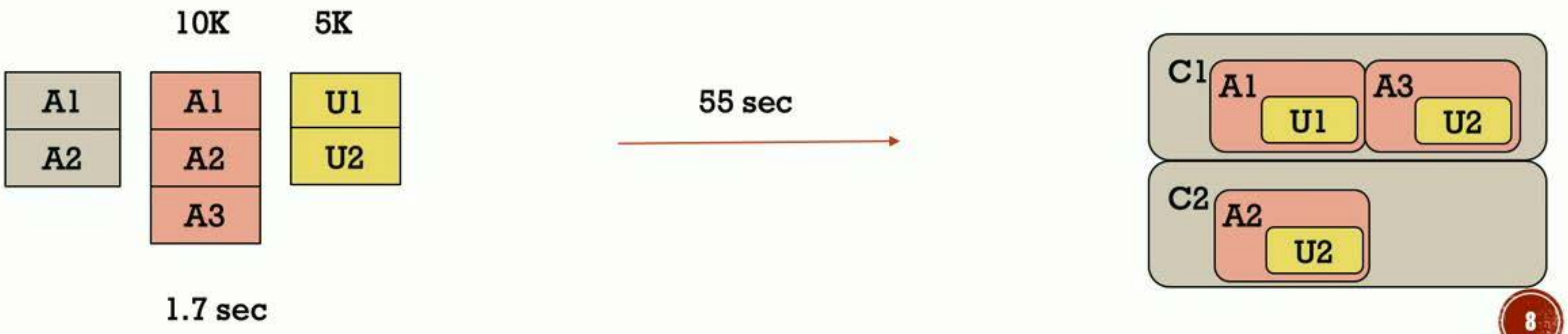


1. Nested Data Model

- A mismatch between how the app access data and how data is stored
  - slow data conversion

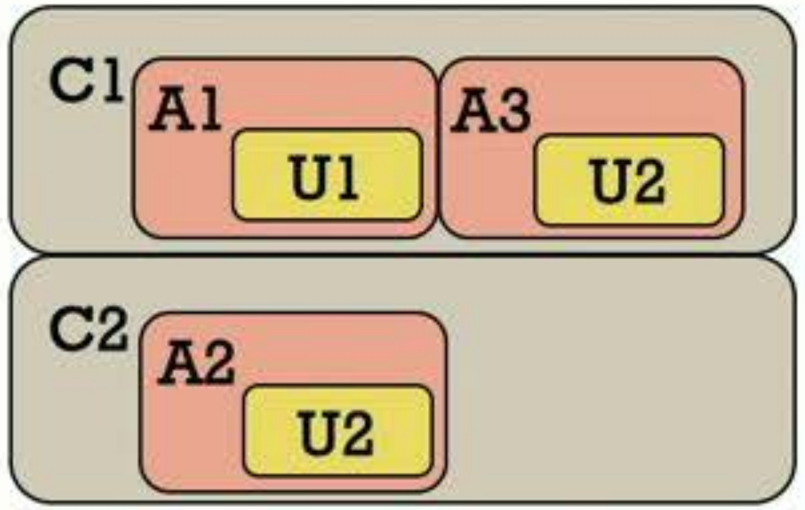
```
Channel.includes(activities, includes(user)).order(id).limit(50)
```

```
SELECT * FROM channel ORDER BY id LIMIT 50;
SELECT * FROM activity WHERE channel_id IN (…);
SELECT * FROM user WHERE id IN (…);
```



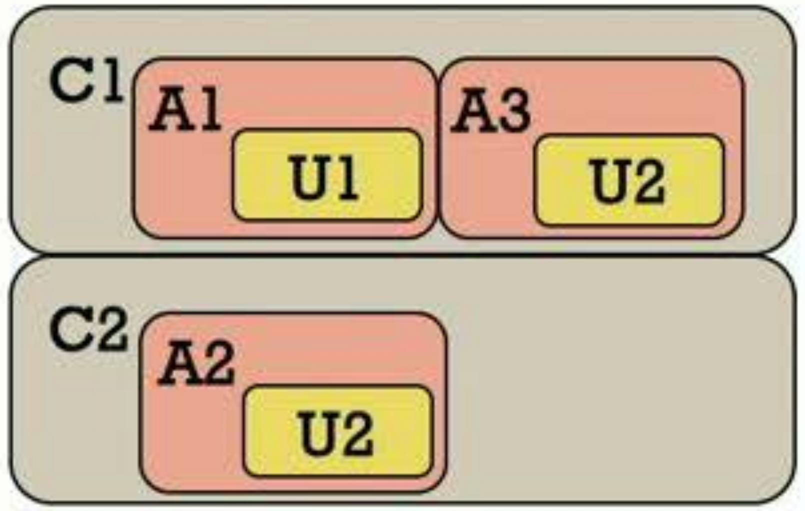
Chestnut: Using Non-relational Storage Model

- Storing data as array of objects and nested objects, and return objects



Chestnut: Using Non-relational Storage Model

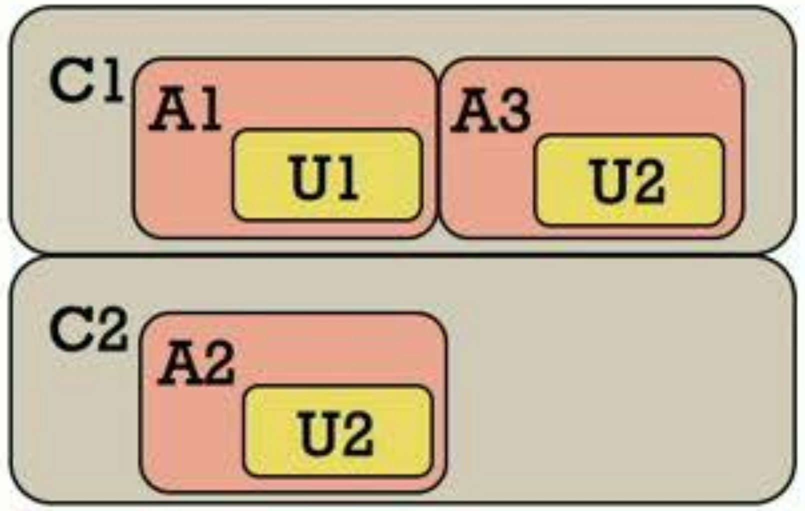
- Storing data as array of objects and nested objects, and return objects



Data conversion: C++ object -> Ruby object

Chestnut: Using Non-relational Storage Model

- Storing data as array of objects and nested objects, and return objects



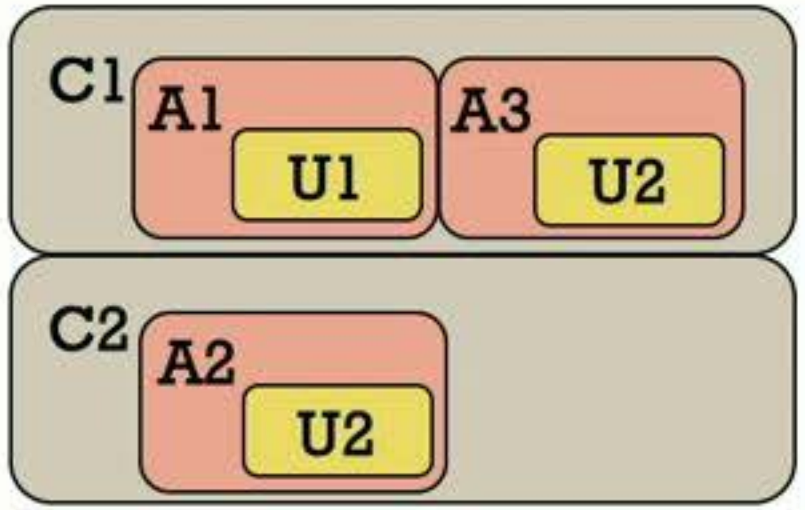
Data conversion: C++ object -> Ruby object

1.5 sec

2.3 sec

Chestnut: Using Non-relational Storage Model

- Storing data as array of objects and nested objects, and return objects



Data conversion: C++ object -> Ruby object

1.5 sec

2.3 sec

15x speedup!

2. Query Predicate Involving Associated Objects

- **Partial index supported by relational databases**

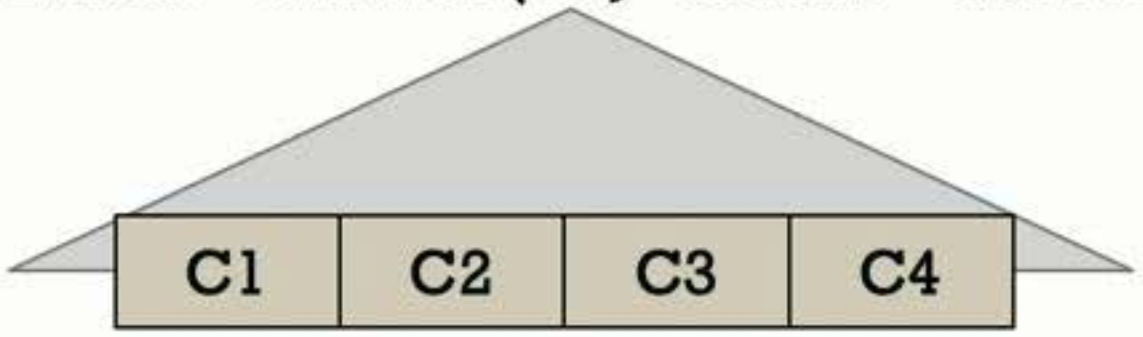
```
Class Channel:
  has_many: activities => Activity
  string status
  …

Class Activity:
  has_one: user => User
  string type
  …

Class User:
  …
```

```
Channel.where(status='active').order(id)
```

```
index: channel(id, status='active')
```



2. Query Predicate Involving Associated Objects

- Partial index **not** supported by relational databases

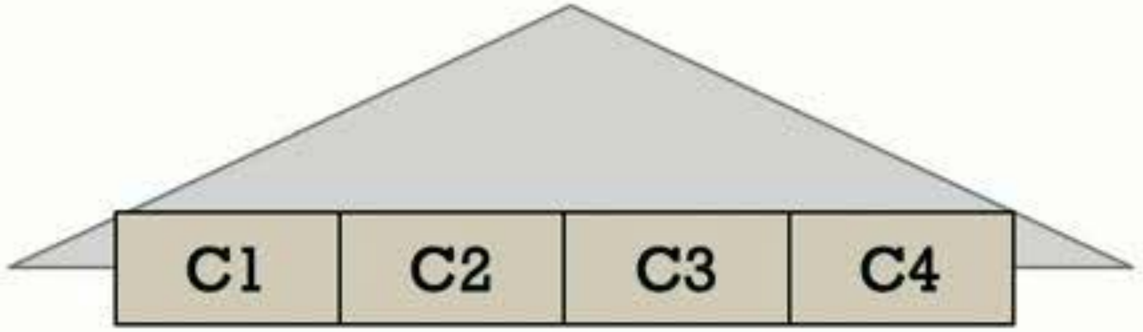
```
Class Channel:
  has_many: activities => Activity
  string status
  …

Class Activity:
  has_one: user => User
  string type
  …

Class User:
  …
```

```
Channel.where(
  exists(activities, type='msg'))
.order(id)
```

index: ??



Chestnut: Extending Index Syntax

- Such partial index is considered by Chestnut

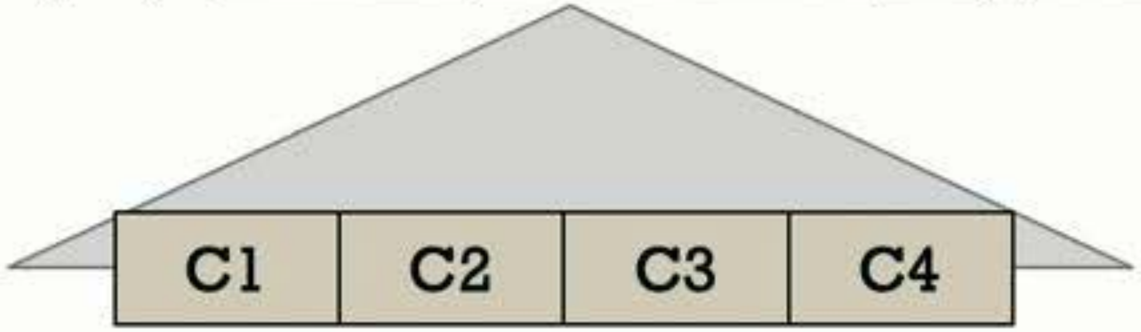
```
Class Channel:
  has_many: activities => Activity
  string status
  …

Class Activity:
  has_one: user => User
  string type
  …

Class User:
  …
```

```
Channel.where(
  exists(activities, type='msg'))
.order(id)
```

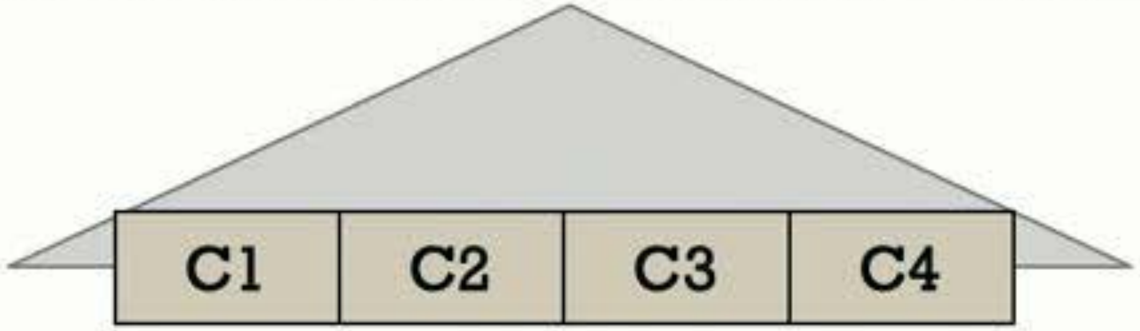
```
index:
channel(id, exists(activities, type='msg'))
```



Chestnut: Extending Index Syntax

- Allow associated object's field to appear in keys and predicates

```
index:
channel(id, exists(activities, type='msg'))
```




```
sorted_array: channel(activities.id)
```

3. Program-generated Query Predicate

- Predicates are generated by chained function calls, often containing overlapping or redundant predicates.
  - E.g., a webpage showing 'join' or 'leave' (and non-'msg') activities created or updated recently

3. Program-generated Query Predicate

- Predicates are generated by chained function calls, often containing overlapping or redundant predicates.
  - E.g., a webpage showing 'join' or 'leave' (and non-'msg') activities created or updated recently

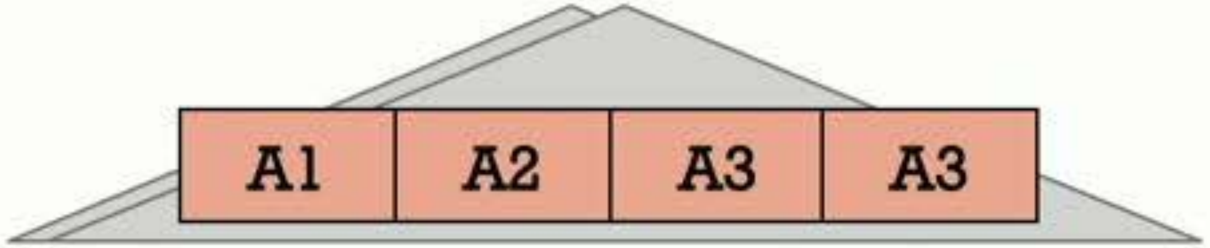
```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or
type='leave') AND (created>? or updated>?)
```

3. Program-generated Query Predicate

- Predicates are generated by chained function calls, often containing overlapping or redundant predicates.
  - E.g., a webpage showing ‘join’ or ‘leave’ (and non-’msg’) activities created or updated recently

```
SELECT * FROM activity WHERE type!=‘msg’ AND (type=‘join’ or
type=‘leave’) AND (created>? or updated>?)
```

```
index1: activity(type, created)
index2: activity(type, updated)
```



3. Program-generated Query Predicate

index1: activity(type, created)
index2: activity(type, updated)

SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)

3. Program-generated Query Predicate

```
index1: activity(type, created)
index2: activity(type, updated)
```


```
SELECT * FROM activity WHERE type!=‘msg’ AND (type=‘join’ or
type=‘leave’) AND (created>? or updated>?)
```

```
  Workers Planned: 2
  ->  Parallel Seq Scan on activities  (cost=0.00..479177.81 rows=81168 width=368)
        Filter: ((type <> 9) AND ((type = 2) OR (type = 3)) AND ((created > '2018-12-12') OR (updated > '2018-12-12')))
```

3. Program-generated Query Predicate

index1: activity(type, created)
index2: activity(type, updated)

SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)

Seq scan: 2.6 sec

3. Program-generated Query Predicate

index1: activity(type, created)
index2: activity(type, updated)


```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or
type='leave') AND (created>? or updated>?)
```

Seq scan: 2.6 sec

```
SELECT * FROM activity WHERE type in ('join', 'leave') AND
(created>? or updated>?)
```

3. Program-generated Query Predicate

```
index1: activity(type, created)
index2: activity(type, updated)
```

SELECT * FROM activity WHERE type!='msg' AND (type='join' or type='leave') AND (created>? or updated>?)

Seq scan: 2.6 sec

SELECT * FROM activity WHERE type in ('join', 'leave') AND (created>? or updated>?)

```
Workers Planned: 2
->  Parallel Bitmap Heap Scan on activity  (cost=5855.50..410948.41 rows=94795 width=368)
      Recheck Cond: (((type = ANY ('{2,3}'::bigint[])) AND (created>'2018-12-12')) OR ((type = ANY ('{2,3}'::bigint[])) AND (updated>'2018-12-12')))
      Filter: (type = ANY ('{2,3}'::bigint[]))
      ->  BitmapOr  (cost=5855.50..5855.50 rows=227604 width=0)
            ->  Bitmap Index Scan on idx_type_created  (cost=0.00..23.54 rows=669 width=0)
                  Index Cond: ((type = ANY ('{2,3}'::bigint[])) AND (created > '2018-12-12'))
            ->  Bitmap Index Scan on idx_type_updated  (cost=0.00..5718.20 rows=226935 width=0)
                  Index Cond: ((type = ANY ('{2,3}'::bigint[])) AND (updated > '2018-12-12'))
```

3. Program-generated Query Predicate

```
index1: activity(type, created)
index2: activity(type, updated)
```


```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or
type='leave') AND (created>? or updated>?)
```

Seq scan: 2.6 sec

```
SELECT * FROM activity WHERE type in ('join', 'leave') AND
(created>? or updated>?)
```

Use index: 0.5 sec

Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

- Enumerate plans
  - From small-size plans to larger-size

```
r=index1.scan((‘join’,2018-01-01), (‘msg’, ∞))
```

…

```
r1=index1.scan((‘join’,2018-01-01), (‘join’, ∞))
r2=index2.scan((‘leave’,2018-01-01), (‘leave’, ∞))
…
r=distinct(union(r1, r2, r3, r4))
```

Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!
- Enumerate plans
  - From small-size plans to larger-size
- Verify each plan against query
  - Symbolic execution

```
r=index1.scan((‘join’,2018-01-01), (‘msg’, ∞))
```

…

```
r1=index1.scan((‘join’,2018-01-01), (‘join’, ∞))
r2=index2.scan((‘leave’,2018-01-01), (‘leave’, ∞))
…
r=distinct(union(r1, r2, r3, r4))
```

Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

- Enumerate plans
  - From small-size plans to larger-size
- Verify each plan against query
  - Symbolic execution

```
r=index1.scan((‘join’,2018-01-01), (‘msg’, ∞))
```

…

```
r1=index1.scan((‘join’,2018-01-01), (‘join’, ∞))
r2=index2.scan((‘leave’,2018-01-01), (‘leave’, ∞))
…
r=distinct(union(r1, r2, r3, r4))
```

- Slower than existing query optimizer, but sometimes can find better plans

Chestnut: Synthesis-based Plan Enumeration

- Rules are not enough to handle all cases!

- Enumerate plans
  - From small-size plans to larger-size
- Verify each plan against query
  - Symbolic execution

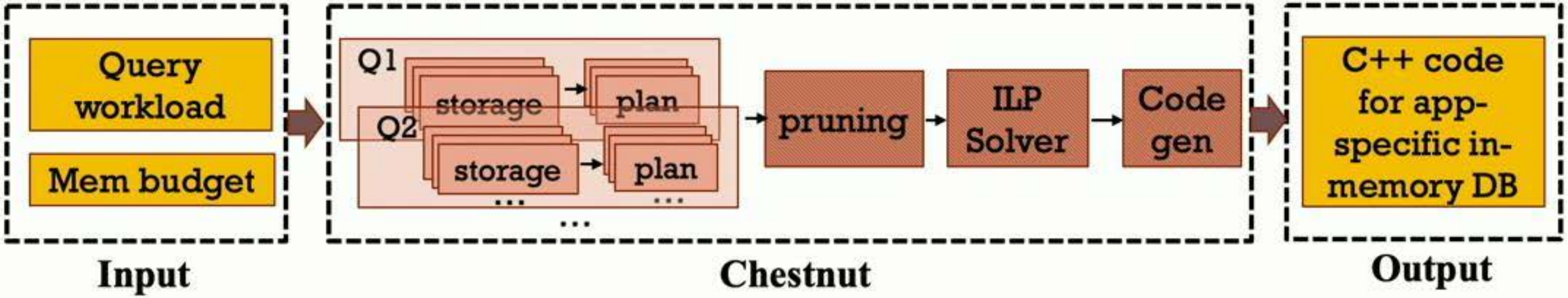
```
r=index1.scan((‘join’,2018-01-01), (‘msg’, ∝
```

…

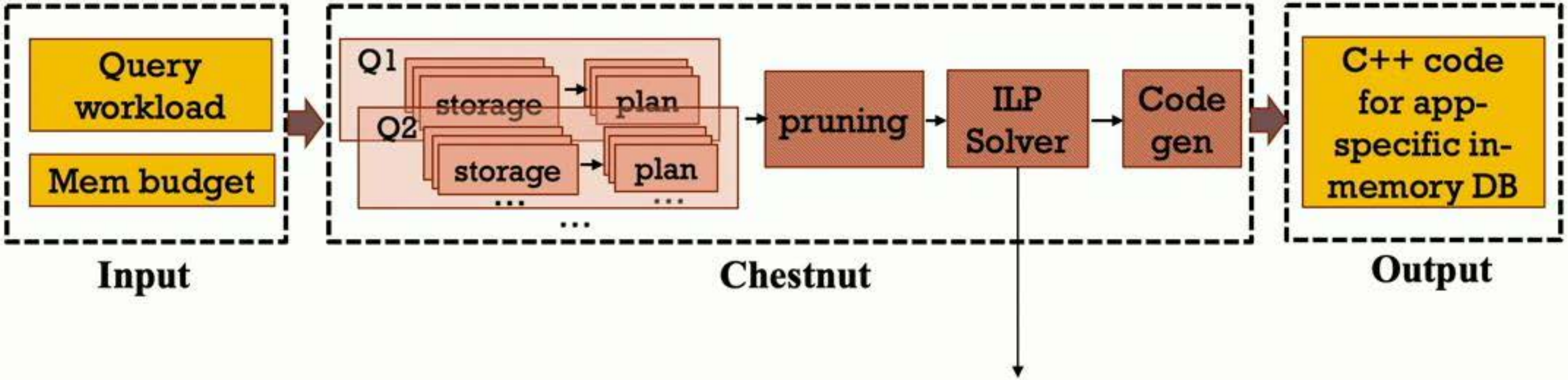
```
r1=index1.scan((‘join’,2018-01-01), (‘join’, ∞))
r2=index2.scan((‘leave’,2018-01-01), (‘leave’, ∞))
…
r=distinct(union(r1, r2, r3, r4))
```

- Slower than existing query optimizer, but sometimes can find better plans

Workflow


Query workload
Mem budget
Input
Q1
storage
plan
Q2
storage
plan
...
...
...
pruning
ILP Solver
Code gen
Chestnut
C++ code for app-specific in-memory DB
Output

Workflow



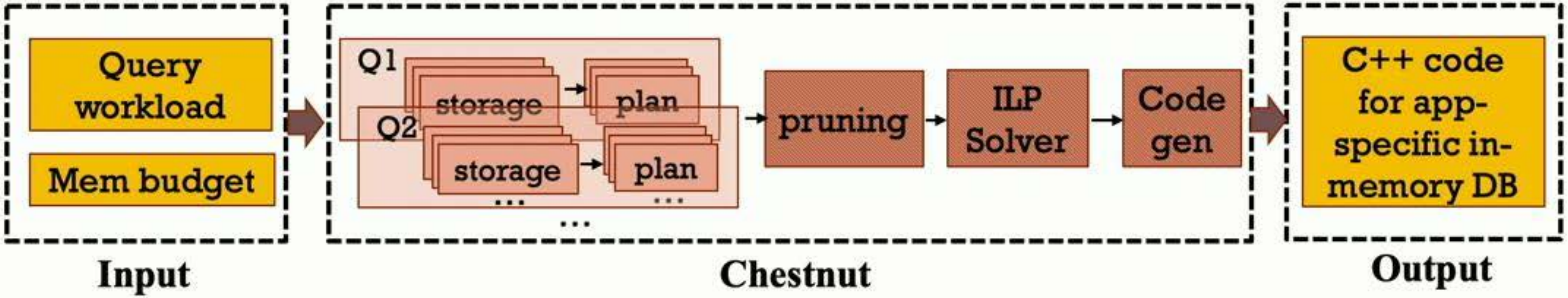
Constraint:

- Each query plan uses some data structures
- The used data structures is within mem budget

Goal:

minimize $\sum query\ time$

Workflow



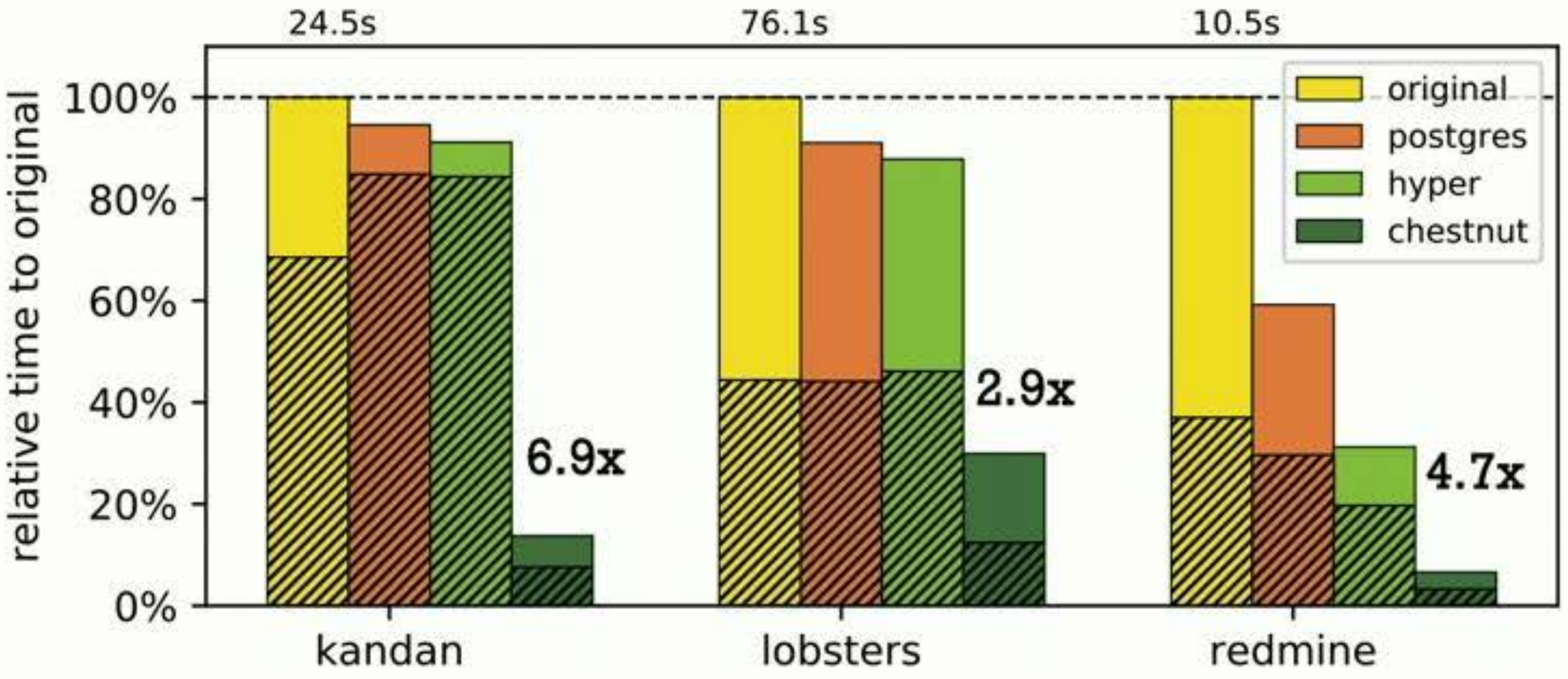
Query workload
Mem budget
Input
Q1
storage
plan
Q2
storage
plan
...
...
...
pruning
ILP Solver
Code gen
Chestnut
C++ code for app-specific in-memory DB
Output

Evaluation

- 3 open-source popular web applications built with Rails
  - kandan: Hipchat-like chatting app
  - redmine: GitHub-like project management
  - lobsters: Hackernews-like forum app
- Compare against:
  - Original setting with MySQL (in-memory)
  - PostgreSQL + automatic indexer (in-memory)
  - Hyper + automatic indexer
  - Chestnut

Evaluation

- 3 open-source popular web applications built with Rails



(average query time with the same memory)
shaded area: convert relational data into objects

Chestnut running time:

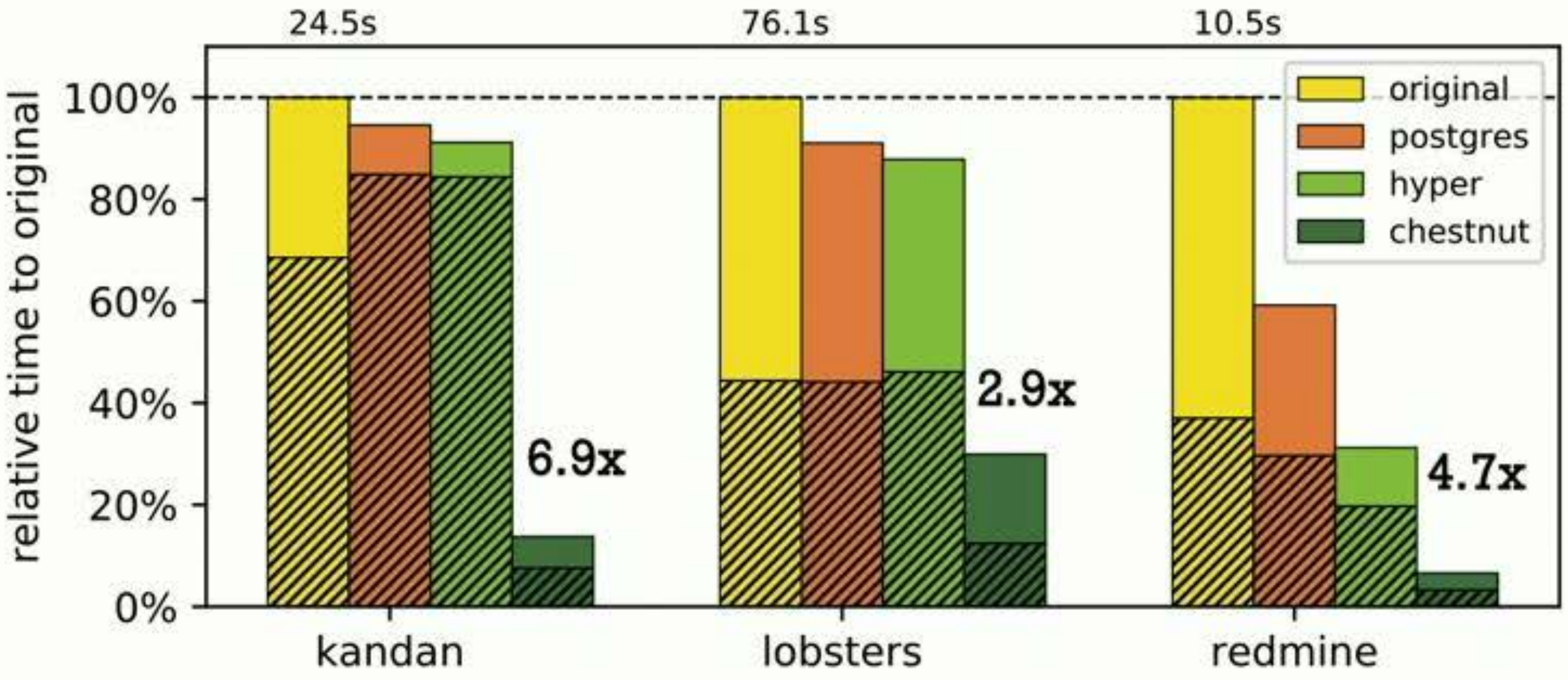
- kandan: 1min
- redmine: 10min
- lobsters: 54min

Conclusion

- Chestnut generates in-memory app-specific database
  - Customize data layout given a workload and a memory budget, optimizing the overall query performance
- Uses non-relational storage model, storing data as objects and nested objects
- Extends index syntax, allowing associated object's field in keys and predicates
- Synthesis-based plan enumeration, enumerate plans and verify each plan
- Achieve significant speedup in real-world web apps
  - >4.8x avg speedup compared to using state-of-the-art in-memory databases

Evaluation

- 3 open-source popular web applications built with Rails

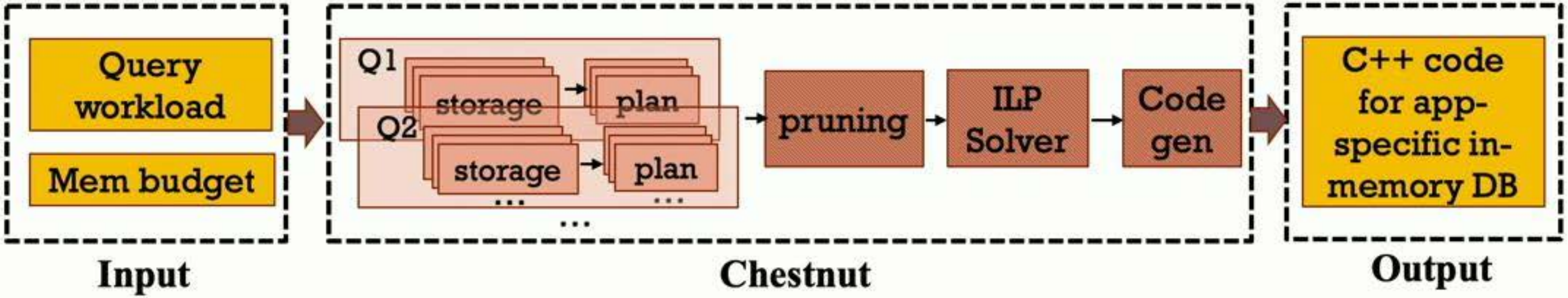


(average query time with the same memory)
shaded area: convert relational data into objects

Chestnut running time:

- kandan: 1min
- redmine: 10min
- lobsters: 54min

Workflow



Query workload
Mem budget
Input
Q1
storage
plan
Q2
storage
plan
...
pruning
ILP Solver
Code gen
Chestnut
C++ code for app-specific in-memory DB
Output

3. Program-generated Query Predicate

```
index1: activity(type, created)
index2: activity(type, updated)
```


```
SELECT * FROM activity WHERE type!='msg' AND (type='join' or
type='leave') AND (created>? or updated>?)
```

Seq scan: 2.6 sec

```
SELECT * FROM activity WHERE type in ('join', 'leave') AND
(created>? or updated>?)
```

Use index: 0.5 sec

>5x speedup!