Optimizing Storage Overhead of User Behavior Log for ML-embedded Mobile Apps

Chen Gong, Yan Zhuang, Zhenzhe Zheng, Yiliu Chen, Sheng Wang, Fan Wu, Guihai Chen

Shanghai Jiao Tong University & ByteDance Client AI

2026-06-10

ACM SIGMETRICS 2026

Outline

1. **Background**
2. Motivation
3. System Design
4. Evaluation

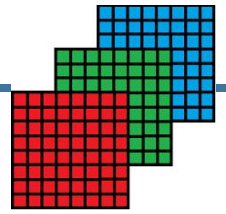
The Rise of On-Device ML

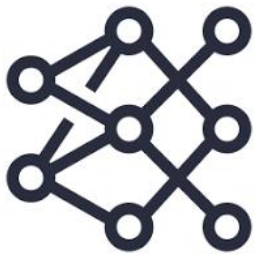
ML is widely integrated into mobile apps for personalized services

Static vs. Dynamic Features

Traditional CV/NLP models: static input features

Image Pixels





Token Embeddings

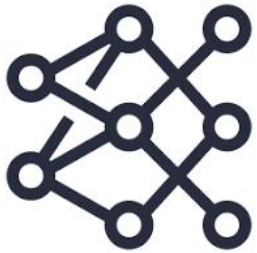
Token String	Token ID	Embedded Vector
"The"	101	[0.5, 0.8, ..., 0.65]
"cat"	2598	[0.1, 0.3, ..., -0.82]
"sat"	4450	[0.41, -0.53, ..., 0.8]
"."	119	[0.17, 0.75, ..., 0.2]

On-device ML models: dynamic user features

T-2: Search

T-1: Watch

T: Favorite

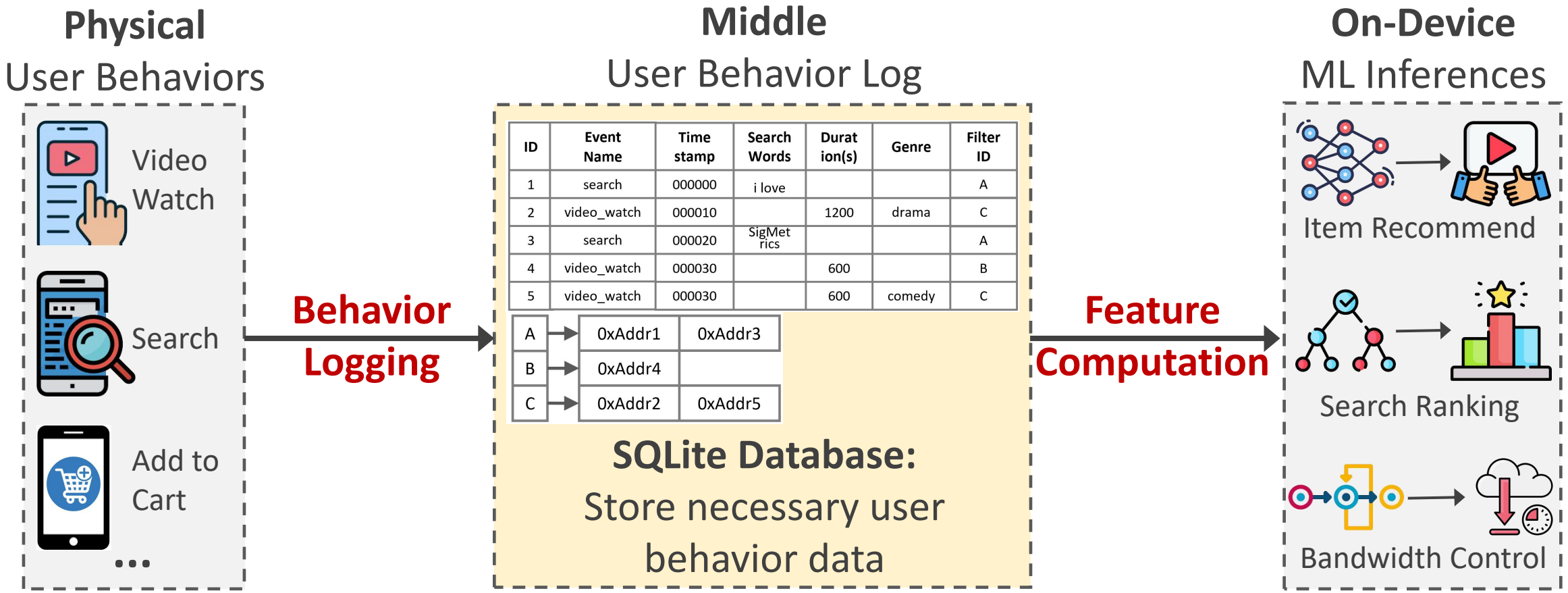


T+1: ?

Evolving user behaviors

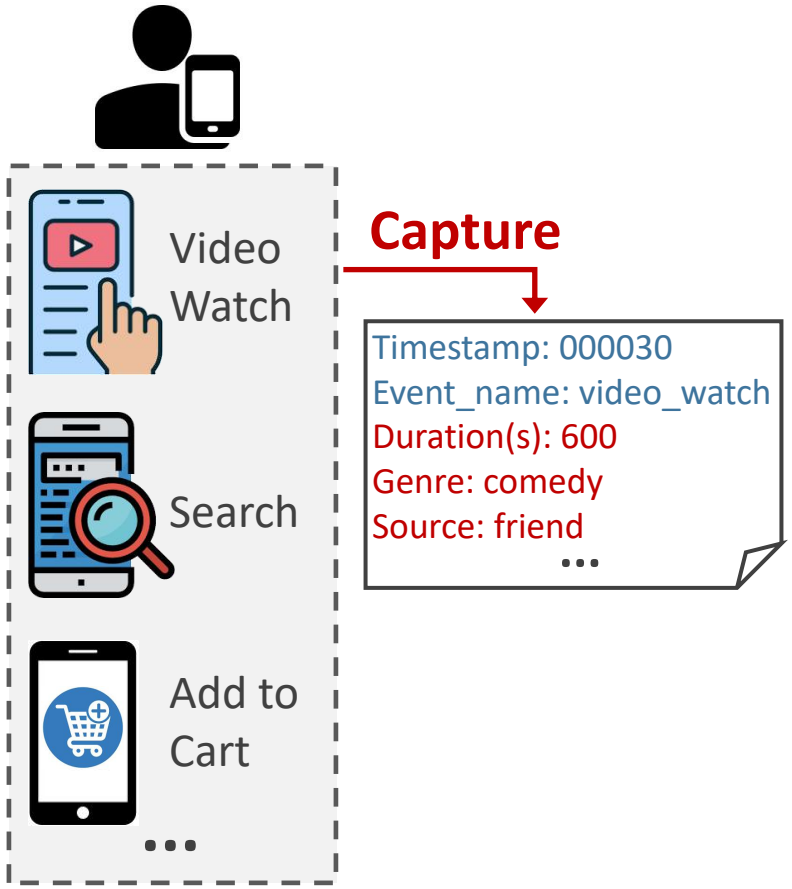
The Role of User Behavior Logs

A database acts as a middle layer connecting physical behaviors and on-device ML models



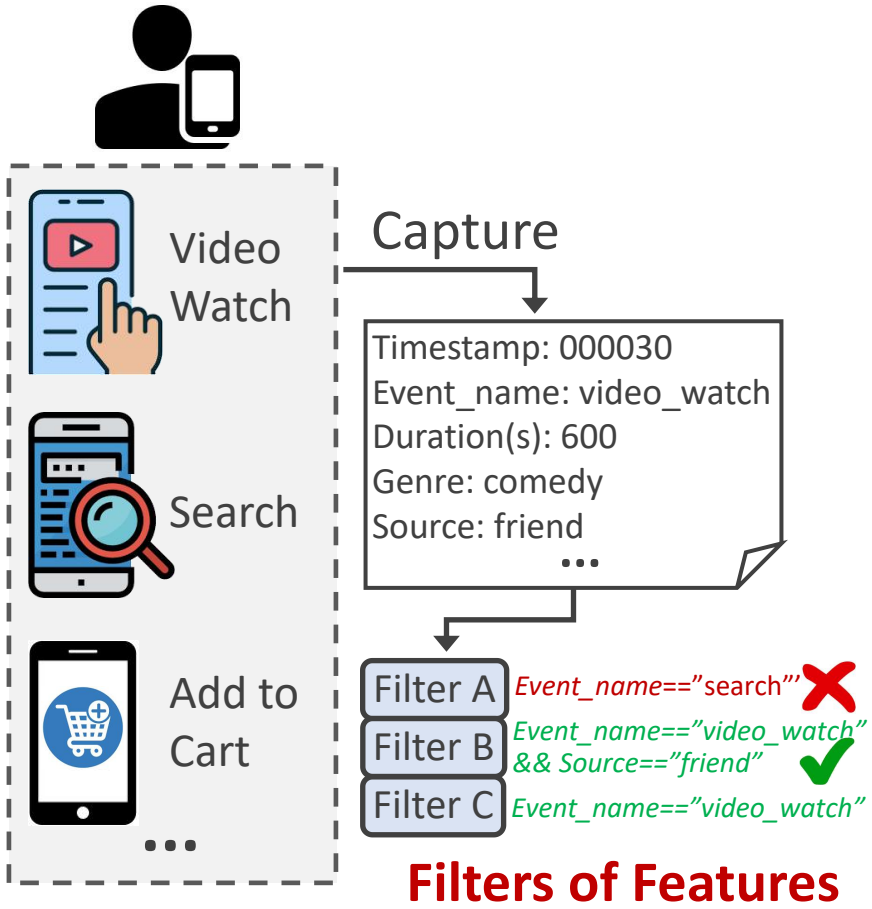
#1 Capturing Behavior Events

Each behavior is captured as an event with massive attributes



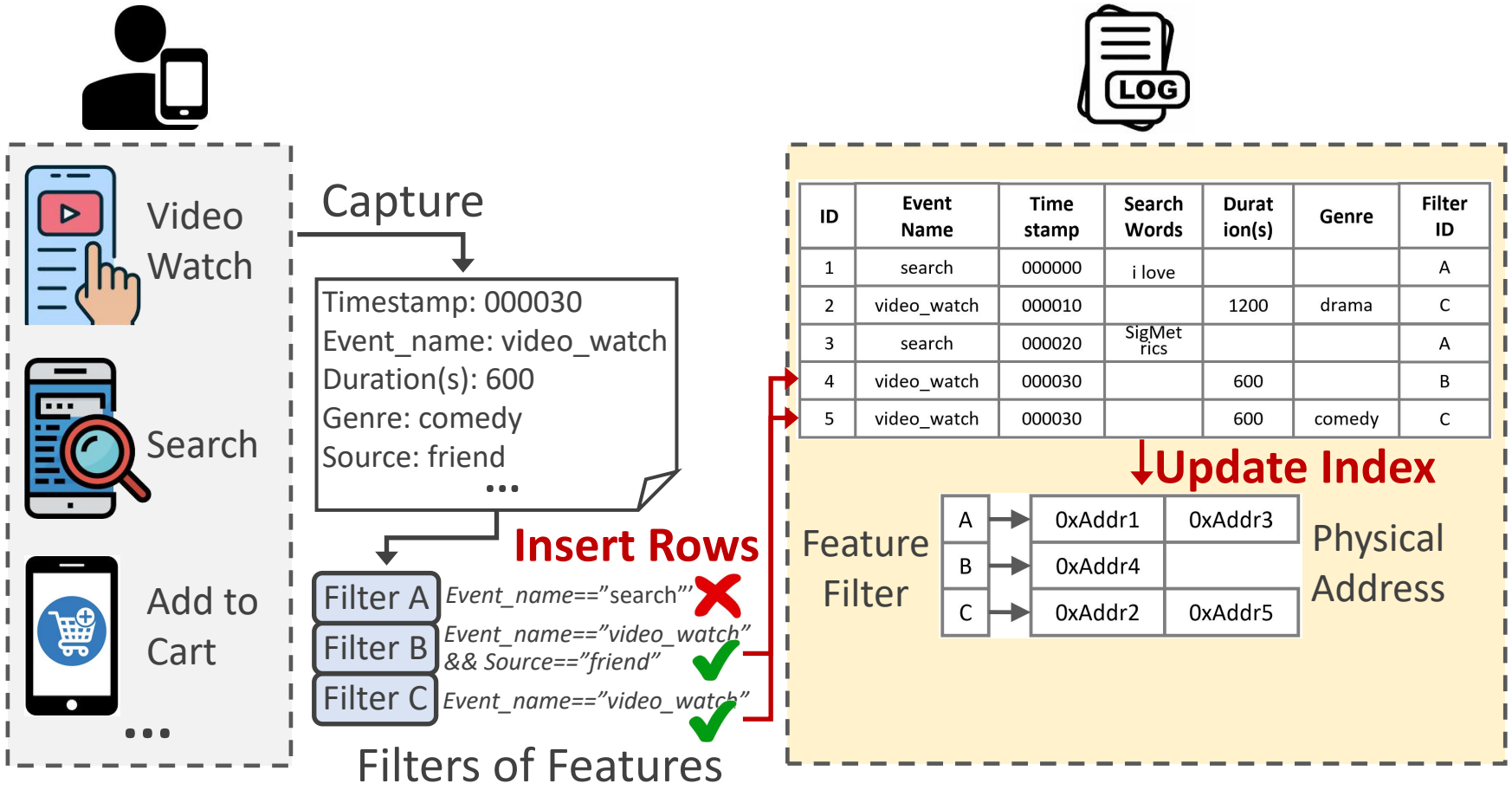
#2 Feature-Specific Filtering

Each event is filtered by different model features in real time



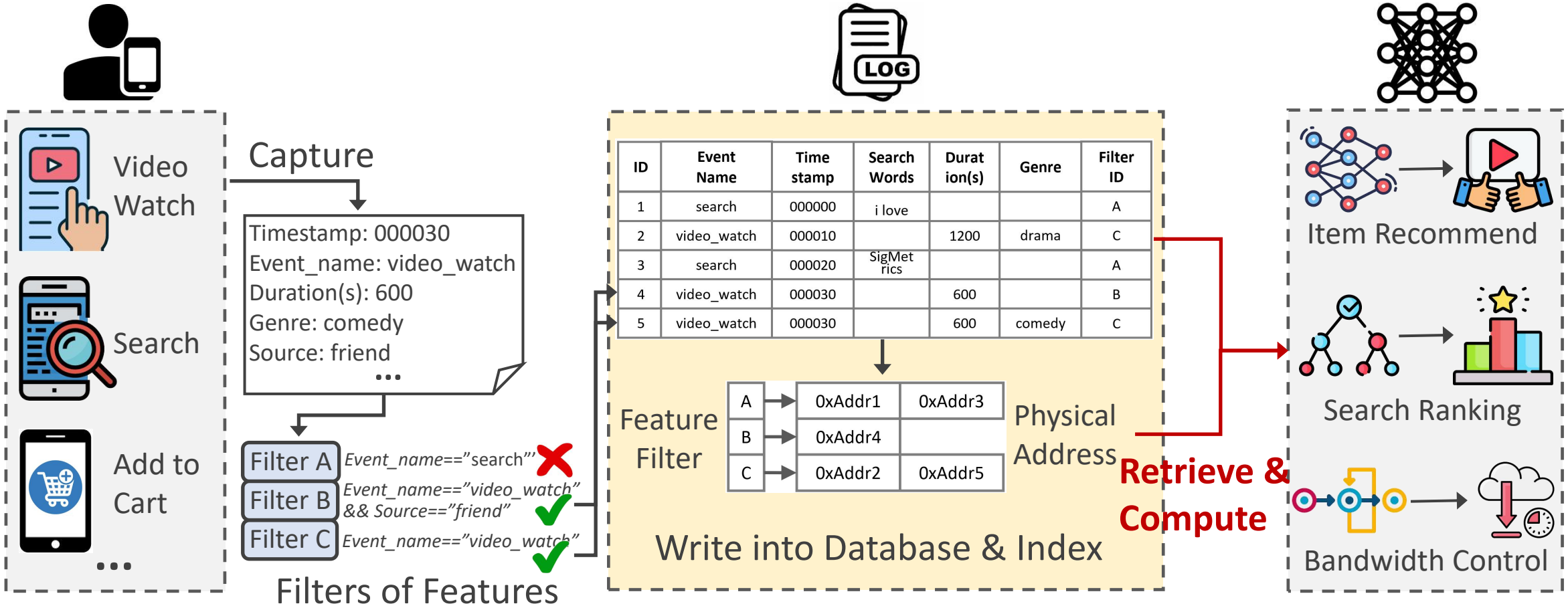
#3 Data Storage and Indexing

Each feature inserts an entry into database and index



#4 Feature Computation

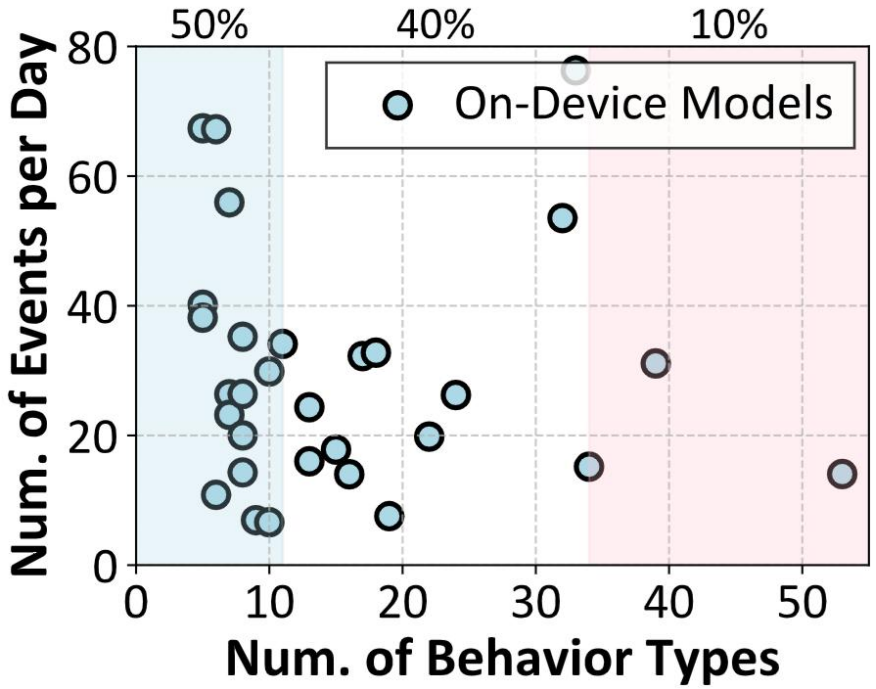
Retrieve necessary attributes from relevant event rows and compute feature values.



Emerging Storage Bottleneck

Behavior log size becomes an emerging storage bottleneck:

- Each single model accumulates massive user data over time

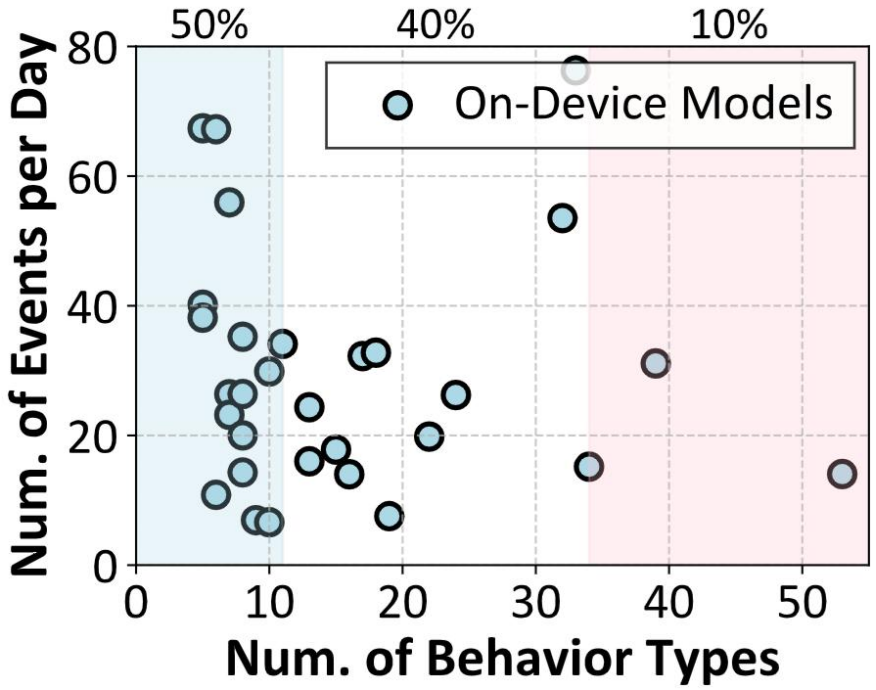


30 MB per model over 6 months

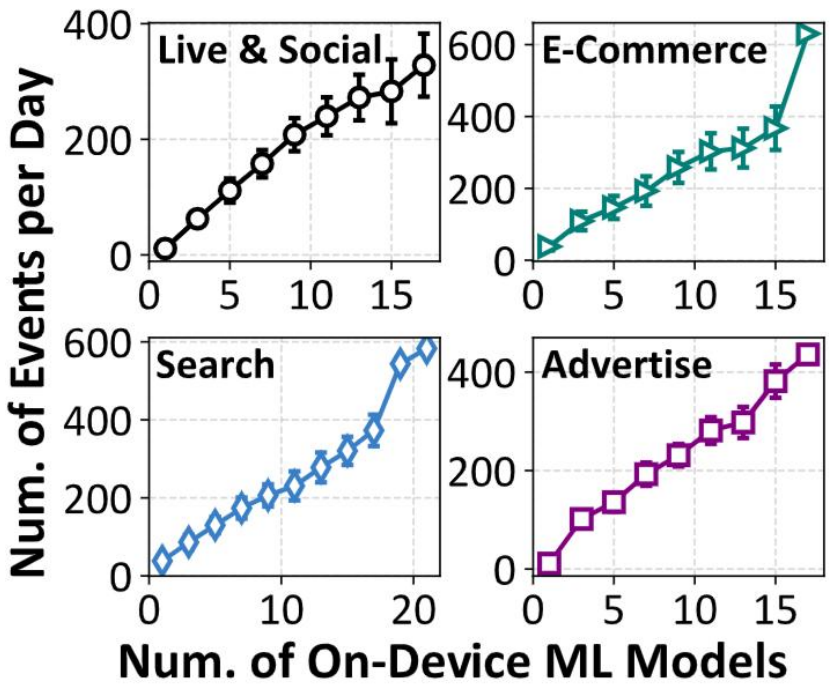
Emerging Storage Bottleneck (Cont')

Behavior log size becomes an emerging storage bottleneck:

- Each single model accumulates massive user data over time
- Storage cost scales linearly with the number of models



30 MB per model over 6 months



Log size increases linearly with models

Emerging Storage Bottleneck (Cont')

Behavior logs account for over 50% of the total app size

Primary driver of
app uninstallation [1]

Every 10 MB size leads to
- 61,000 fewer daily active users
- $7,000 financial loss per day

Dilemma: personalized experiences vs. app size

Dilemma: ML-powered services improve personalization but imposes growing storage burden.

Outline

1 Background

2 **Motivation**

3 System Design

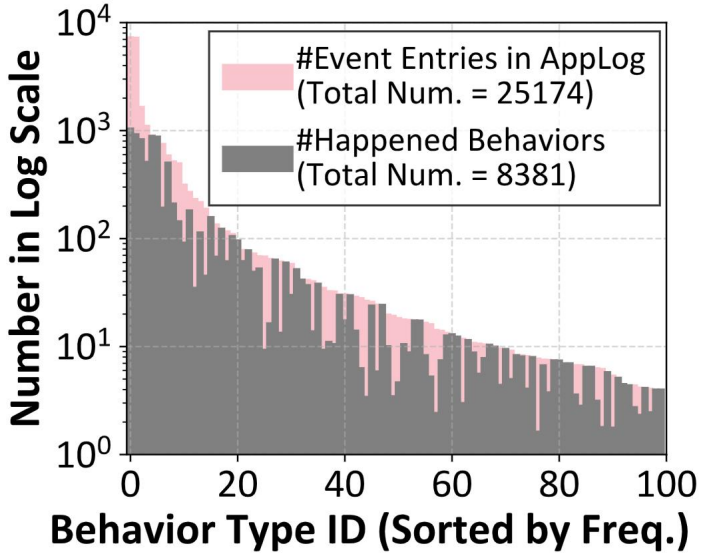
4 Evaluation

Observation 1: Storage Redundancy

Behavior log database exhibits up to 67% redundant event rows due to overlapping filters of different features.

ID	Event Name	Time stamp	Search Words	Duration(s)	Genre	Filter ID
1	search	000000	i love			A
2	video_watch	000010		1200	drama	C
3	search	000020	SigMetrics			A
4	video_watch	000030		600		B
5	video_watch	000030		600	comedy	C

Video watch at time 000030 is needed by both features B and C but saved separately for fast retrieval.



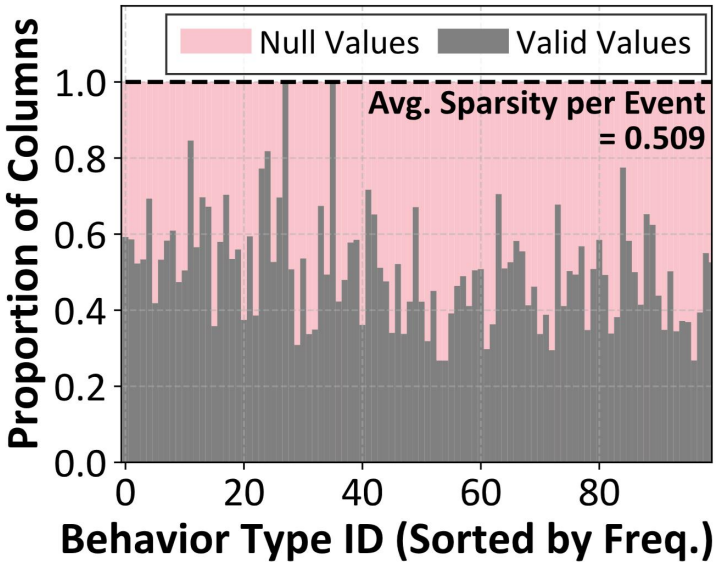
14 days of TikTok Data:
8,381 behaviors events
vs. 25,174 event rows

Observation 2: Storage Sparsity

Storing heterogeneous behavior data in a unified log file creates massive column sparsity (50.9% null values)

ID	Event Name	Time stamp	Search Words	Duration(s)	Genre	Filter ID
1	search	000000	i love			A
2	video_watch	000010		1200	drama	C
3	search	000020	SigMetrics			A
4	video_watch	000030		600		B
5	video_watch	000030		600	comedy	C

Video watch and search have different attributes but are stored centrally.



14 days of TikTok Data:

50.9% of all attribute values are null

(11% wasted storage for null values)

Introducing AdaLog

AdaLog: the first system to reduce redundancy and sparsity of behavior logs without compromising inference quality

ID	Event Name	Time stamp	Search Words	Durat ion(s)	Genre	Filter ID
1	search	000000	i love			A
2	video_watch	000010		1200	drama	C
3	search	000020	SigMet rics			A
4	video_watch	000030		600		B
4	video_watch	000030		600	comedy	C

Insight 1: Reduce Redundancy

Merge rows from the same event

ID	Event Name	Time stamp	Search Words	Durat ion(s)	Genre	Filter ID
1	search	000000	i love			A
2	video_watch	000010		1200	drama	C
3	search	000020	SigMet rics			A
4	video_watch	000030		600	comedy	B & C

Insight 2: Eliminate Sparsity

Distribute distinct behaviors in different logs

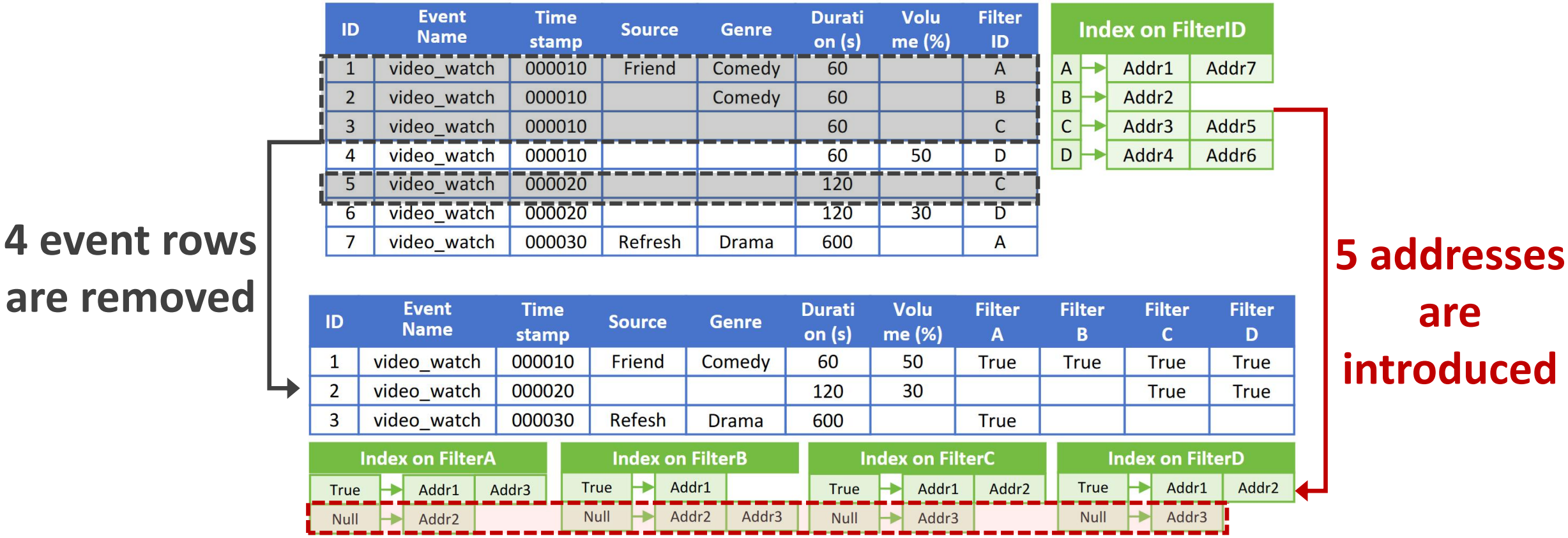
ID	Event Name	Time stamp	Durat ion(s)	Genre	Filter ID
2	video_watch	000010	1200	drama	C
4	video_watch	000030	600	comedy	B & C

ID	Event Name	Time stamp	Search Words	Filter ID
1	search	000000	i love	A
3	search	000020	SigMet rics	A

C1: Feature-Level Data Merging

Identifying which features to merge is an NP-hard problem

- Merge redundant rows saves data space but increase index cost

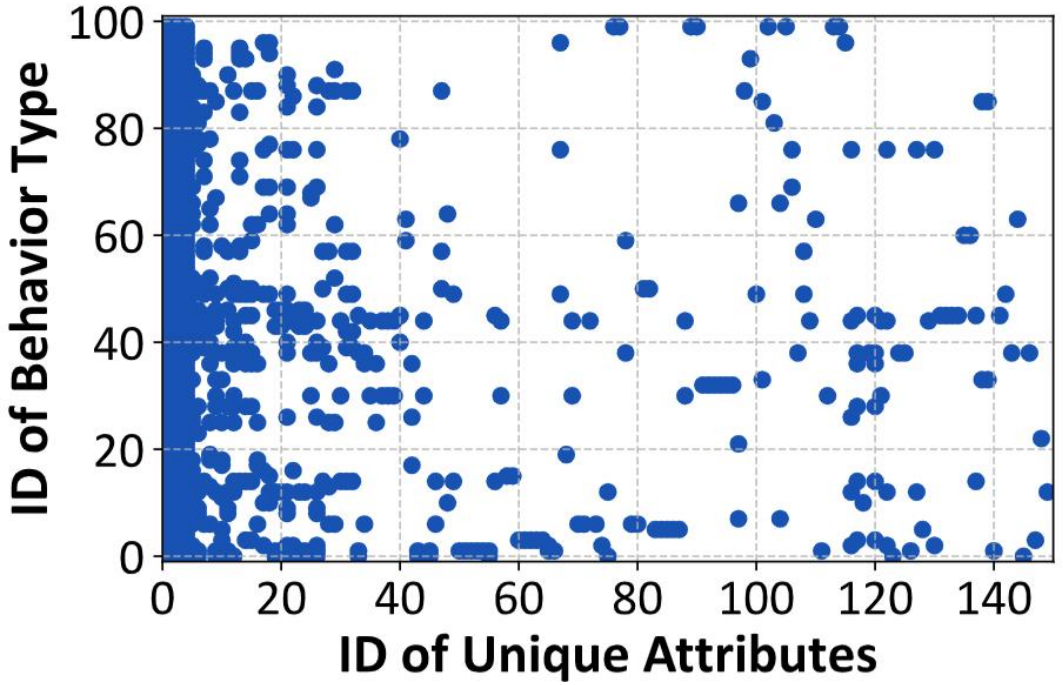


ID	Event Name	Time stamp	Source	Genre	Duration (s)	Volume (%)	Filter ID
1	video_watch	000010	Friend	Comedy	60		A
2	video_watch	000010		Comedy	60		B
3	video_watch	000010			60		C
4	video_watch	000010			60	50	D
5	video_watch	000020			120		C
6	video_watch	000020			120	30	D
7	video_watch	000030	Refresh	Drama	600		A

ID	Event Name	Time stamp	Source	Genre	Duration (s)	Volume (%)	Filter A	Filter B	Filter C	Filter D
1	video_watch	000010	Friend	Comedy	60	50	True	True	True	True
2	video_watch	000020			120	30			True	True
3	video_watch	000030	Refesh	Drama	600		True			

C2: Behavior-Level Log Splitting

Deciding where to store each behavior data to eliminate storage sparsity is not straightforward either.



Ideal Solution:

Store behaviors with different attributes in separate log files

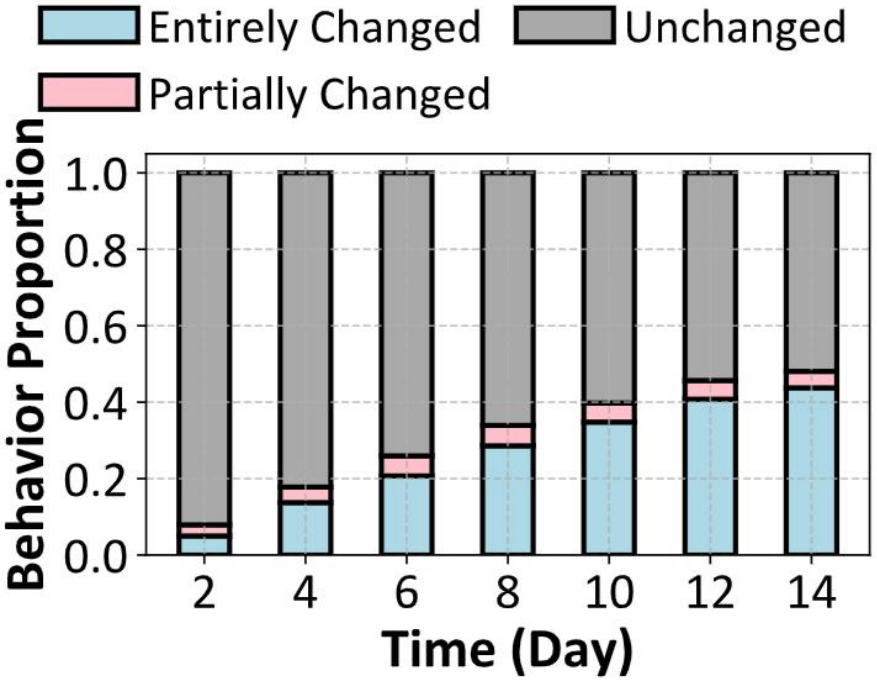
Infeasibility:

250 heterogeneous behaviors → ≥100 fragmented files

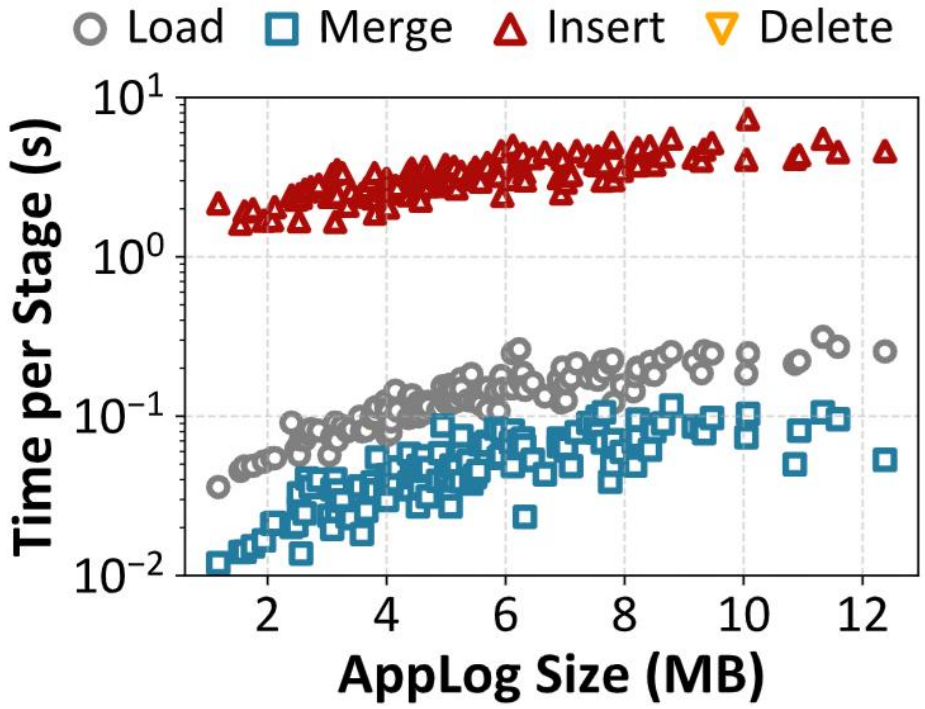
C3: Scalability to Dynamic Patterns

Maintaining an up-to-date behavior log is essential but costly

User behaviors and optimal merging strategy are dynamic



Full log reconstruction requires imposes severe I/O costs.



Outline

1 Background

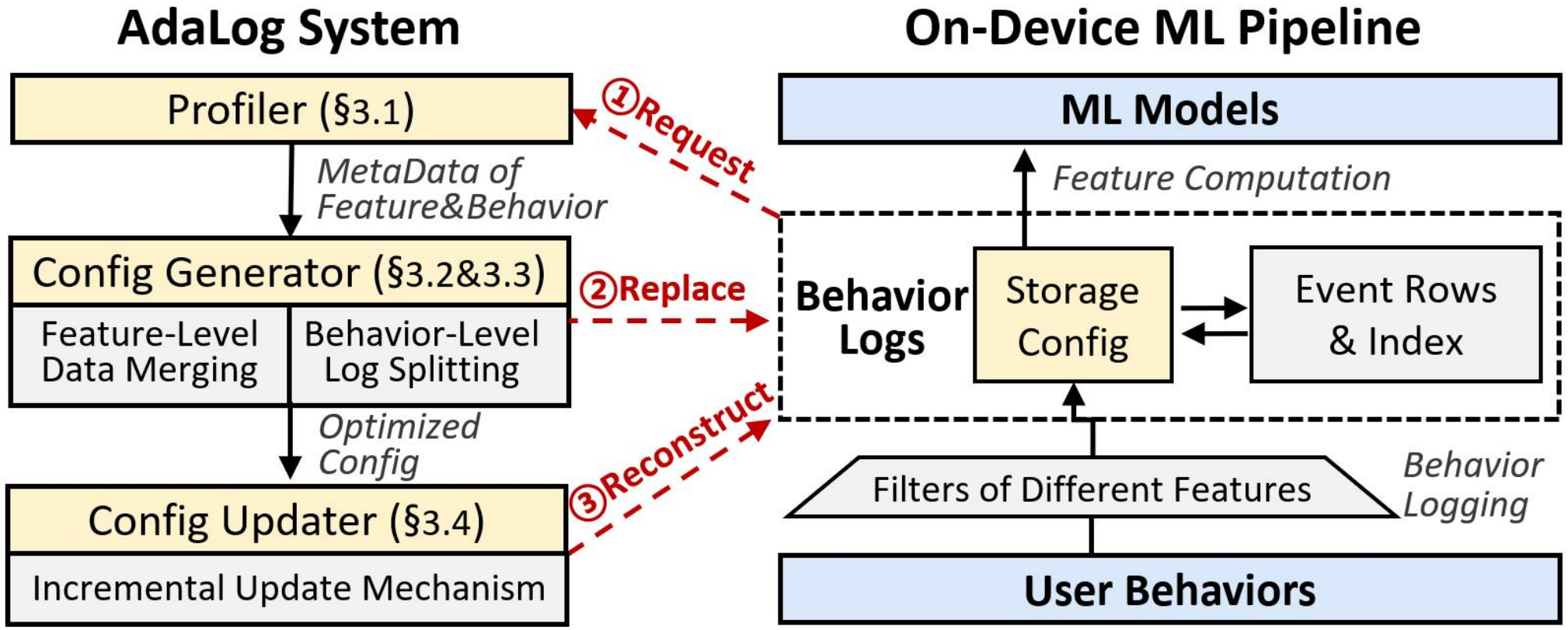
2 Motivation

3 **System Design**

4 Evaluation

Overview of AdaLog

Core Components: Profiler, Config Generator, Config Updater

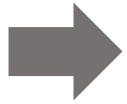


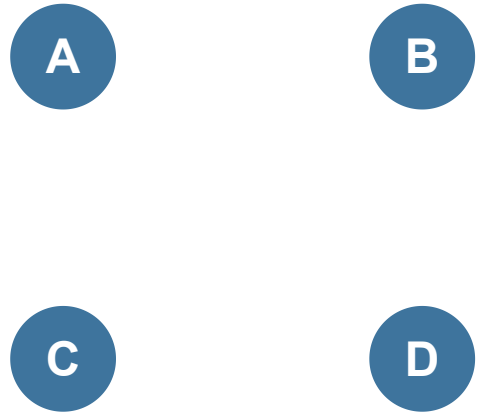
C1: NP-Hard Merging Problem

Formulate feature-level data merging decision as a maximum weighted matching problem in a hypergraph

- Each feature → Node

ID	Event Name	Time stamp	Source	Genre	Duration (s)	Volume (%)	Filter ID
1	video_watch	000010	Friend	Comedy	60		A
2	video_watch	000010		Comedy	60		B
3	video_watch	000010			60		C
4	video_watch	000010			60	50	D
5	video_watch	000020			120		C
6	video_watch	000020			120	30	D
7	video_watch	000030	Refresh	Drama	600		A



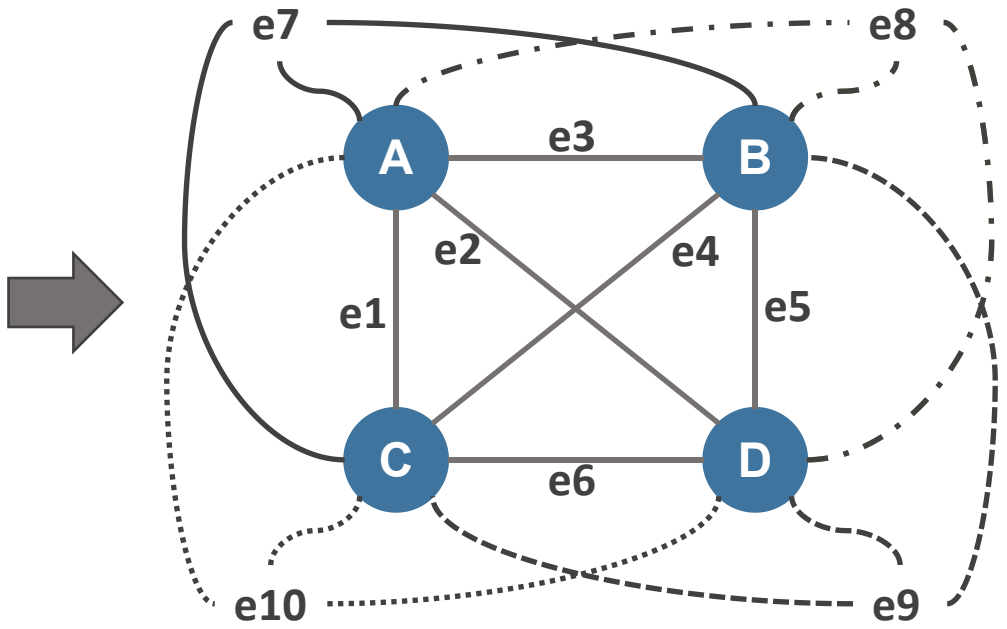


C1: NP-Hard Merging Problem (Cont')

Formulate feature-level data merging decision as a maximum weighted matching problem in a hypergraph

- Each feature → Node
- Any potential feature group → Edge

ID	Event Name	Time stamp	Source	Genre	Duration (s)	Volume (%)	Filter ID
1	video_watch	000010	Friend	Comedy	60		A
2	video_watch	000010		Comedy	60		B
3	video_watch	000010			60		C
4	video_watch	000010			60	50	D
5	video_watch	000020			120		C
6	video_watch	000020			120	30	D
7	video_watch	000030	Refresh	Drama	600		A

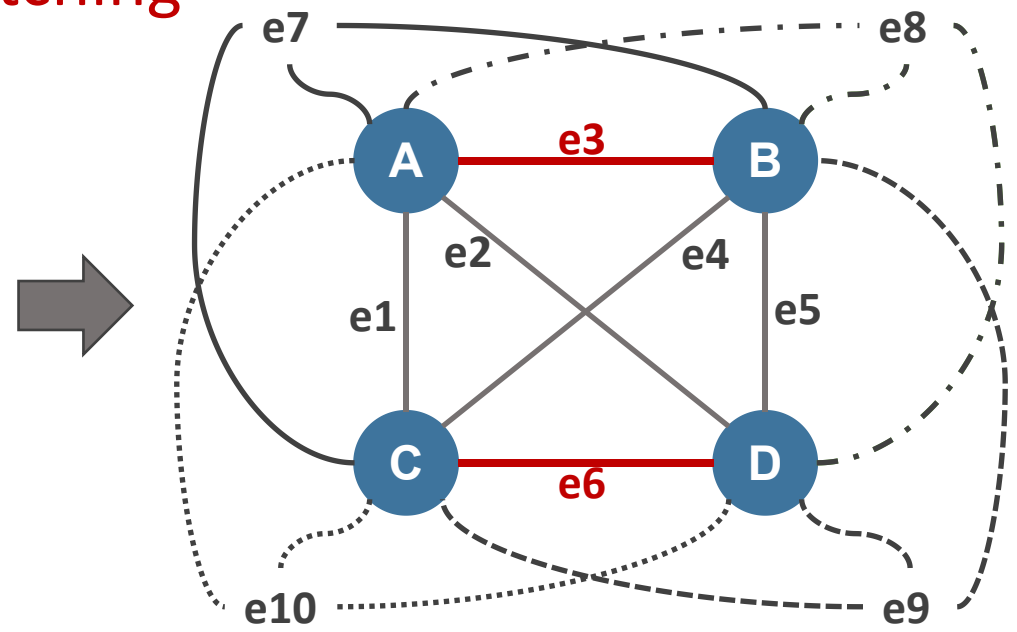


C1: NP-Hard Merging Problem (Cont')

Formulate feature-level data merging decision as a maximum weighted matching problem in a hypergraph

- Each feature → Node
- Any potential feature group → Edge
- Each feature merging decision → Matching

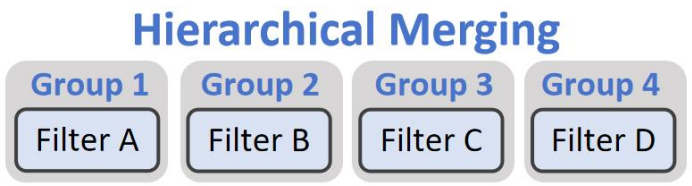
ID	Event Name	Time stamp	Source	Genre	Duration (s)	Volume (%)	Filter ID
1	video_watch	000010	Friend	Comedy	60		A
2	video_watch	000010		Comedy	60		B
3	video_watch	000010			60		C
4	video_watch	000010			60	50	D
5	video_watch	000020			120		C
6	video_watch	000020			120	30	D
7	video_watch	000030	Refresh	Drama	600		A



Tech1: Hierarchical Merging

Decompose into a series of tractable 2D matchings

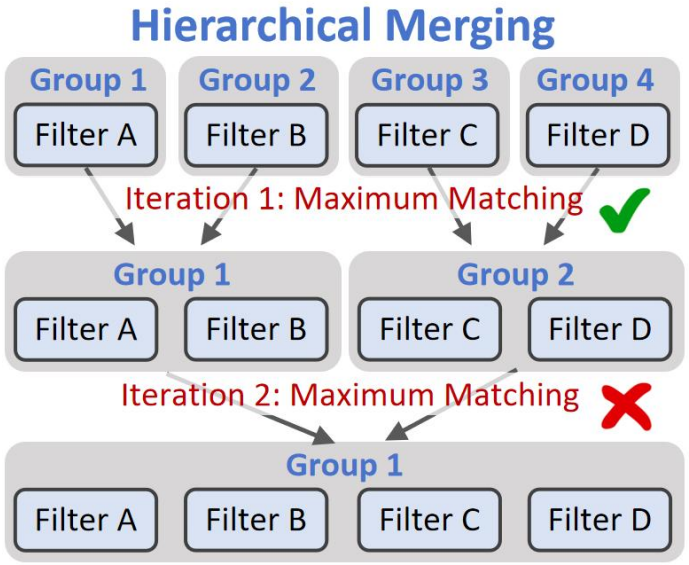
- Start with each feature as a standalone group



Tech1: Hierarchical Merging

Decompose into a series of tractable 2D matchings

- Start with each feature as a standalone group
- Iteratively merge pairs of feature groups with highest storage reduction.



Polynomial Complexity:

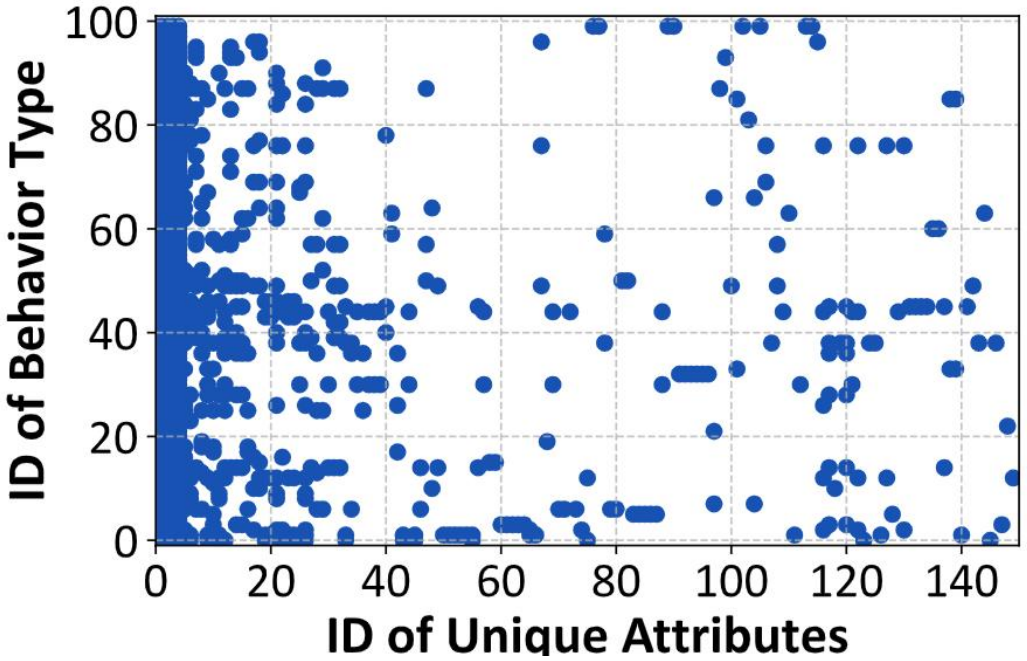
$$\sum_{t=1}^{\log_2 |\mathcal{F}|} O\left(\left(\frac{|\mathcal{F}|}{2^{t-1}}\right)^3\right) = O(|\mathcal{F}|^3) \cdot \sum_{t=1}^{\log_2 |\mathcal{F}|} \left(\frac{1}{8}\right)^{t-1} = O(|\mathcal{F}|^3).$$

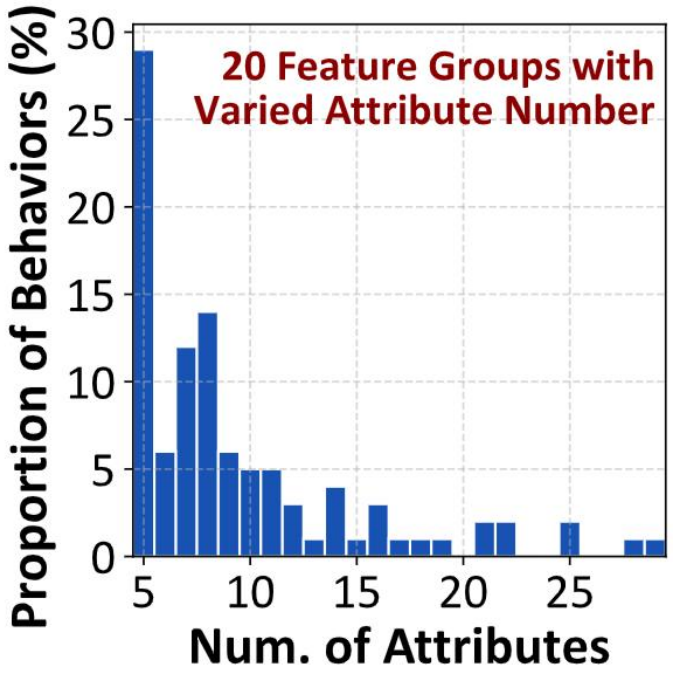
Run for at most $log|F|$ iterations

Each iter's compexity is polynomial

C2: Massive Heterogeneous Behaviors

Observation: 250 behavior types have ≥100 distinct attribute sets but only 20 possible attribute counts.





Tech2: Virtually Hashed Attribute

Dense Storage: store different attributes in shared physical columns using virtual IDs.

ID	Event Name	Time stamp	Durat ion (s)	Genre	Query Word	Price ($)	Filter ID
1	video_watch	000010	600	Comedy			A
2	search	000020		News	Network		B
3	add_to_cart	000030		Toy		10	C

Sparse Attribute Columns

Virtually Hashed Attribute Name

ID	Event Name	Time stamp	Attri bute 1	Attri bute 2	Filter ID
1	video_watch	000010	600	Comedy	A
2	search	000020	Network	News	B
3	add_to_cart	000030	10	Toy	C

Behavior Log File

video_watch:
{Duration: 1, Genre: 2},
search:
{Query Word: 1, Genre: 2},
dislike:
{Price: 1, Genre: 2}

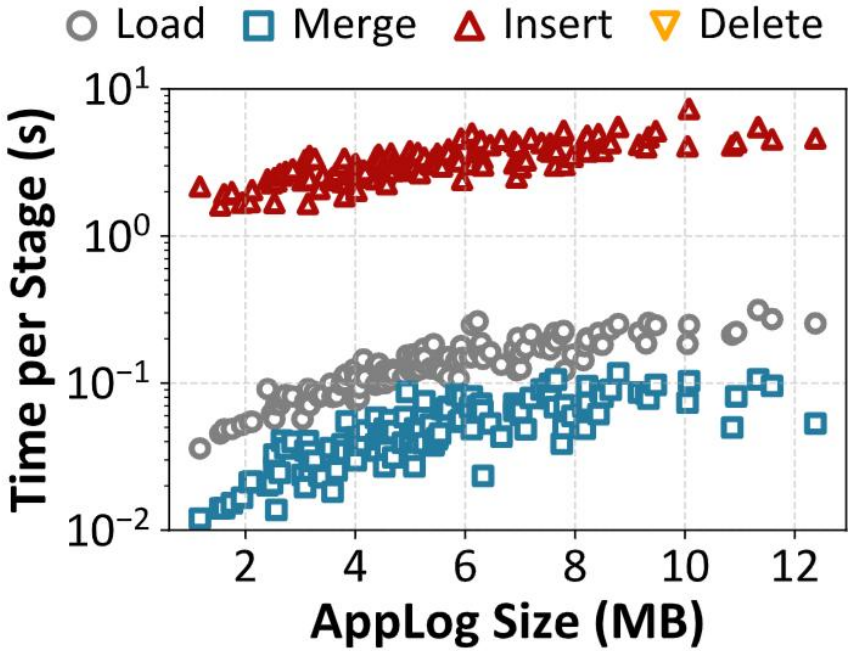
Attribute Mapping

Dense Storage

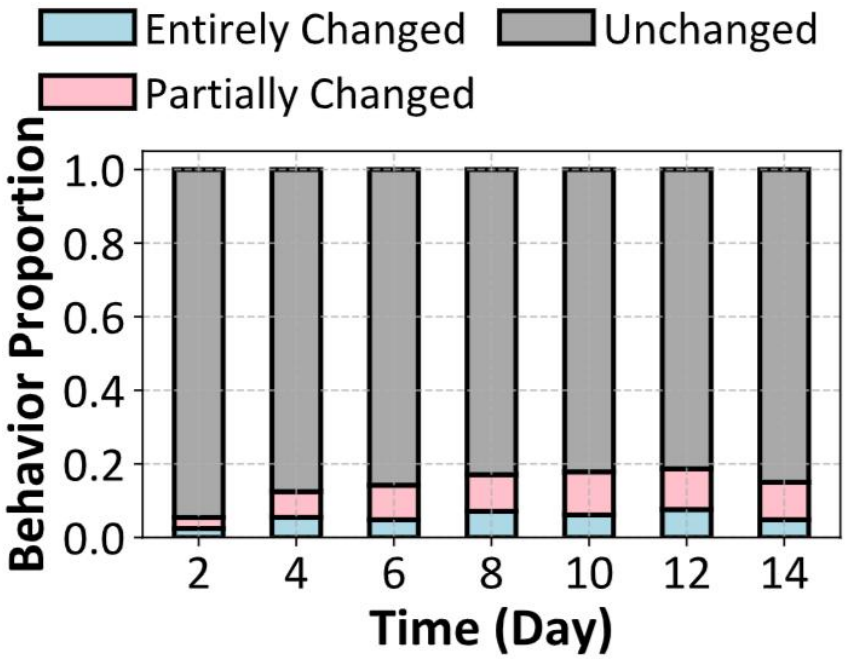
C3: Scalability to Dynamic Patterns

Frequent full log reconstruction leads to heavy I/O costs

- Solution: full reconstruction → incremental update



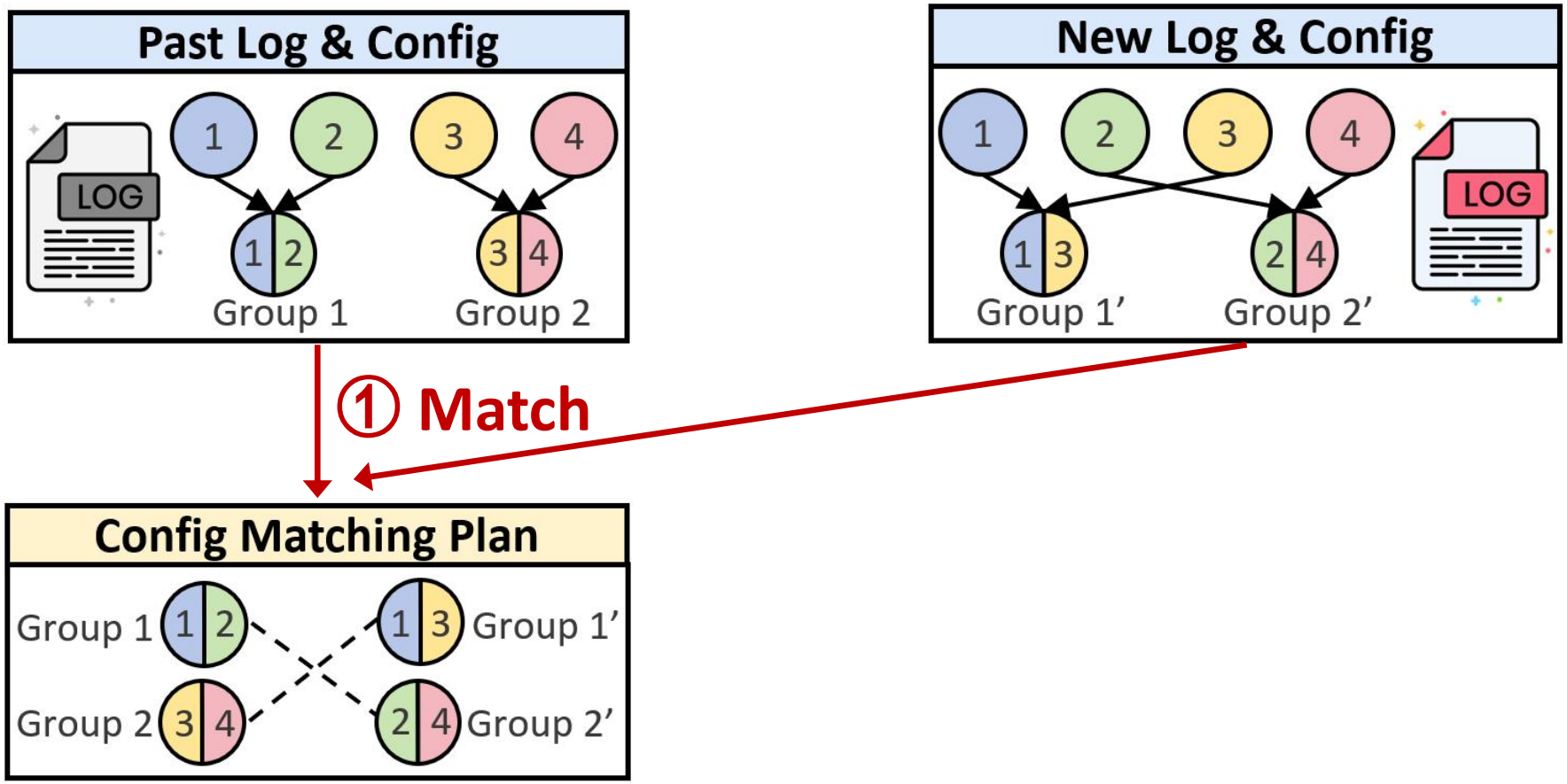
- 95% of reconstruction time is I/O operations



- 86% of behaviors have unchanged strategies

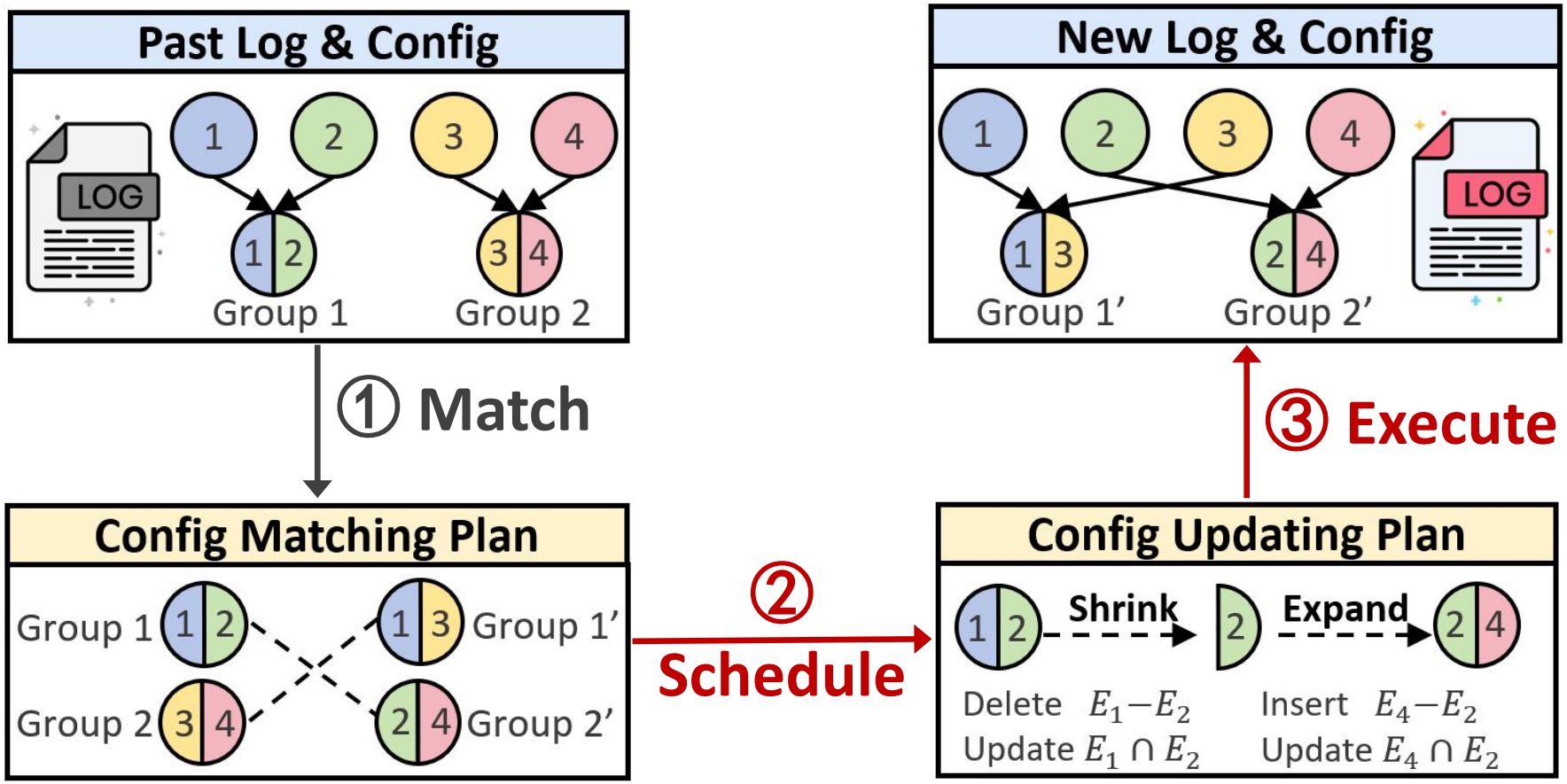
Tech3: Incremental Update

Match: reuse as much historical data as possible through **bipartite matching**



Tech3: Incremental Update (Cont')

Schedule & Execute: A shrink-and-expand strategy efficiently updates log files with minimal I/O overhead



Outline

1 Background

2 Motivation

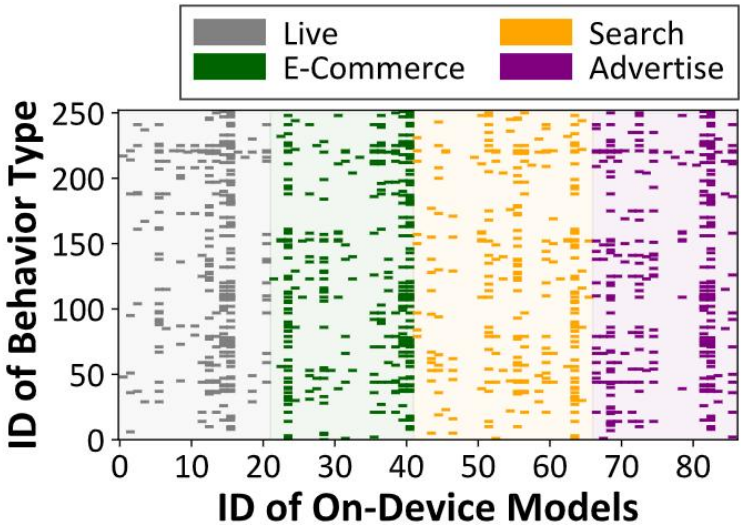
3 System Design

4 **Evaluation**

Evaluation Setup

Diverse App Domains:

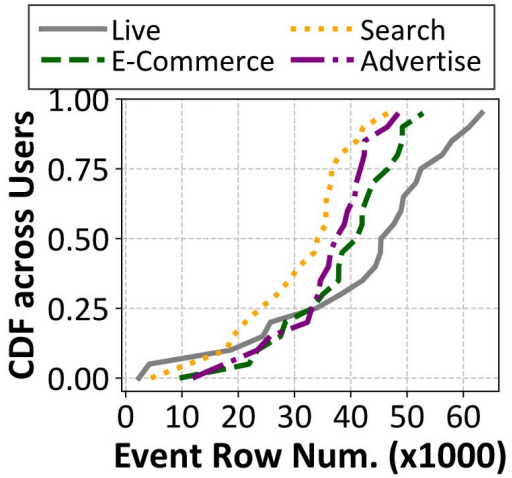
- Live streaming (21 models)
- E-Commerce (20 models)
- Search (25 models)
- Advertisement (20 models)

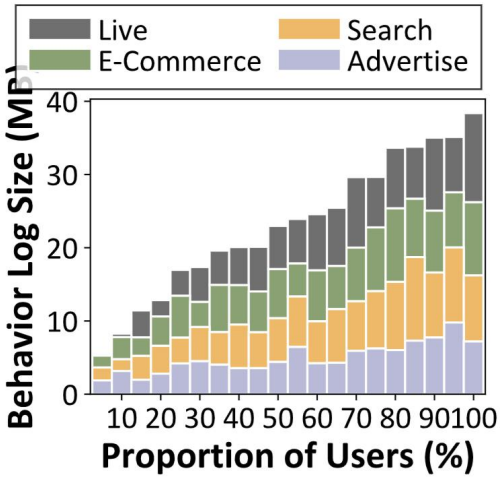


Varied behaviors required by different models and apps

Real-World Users & Data:

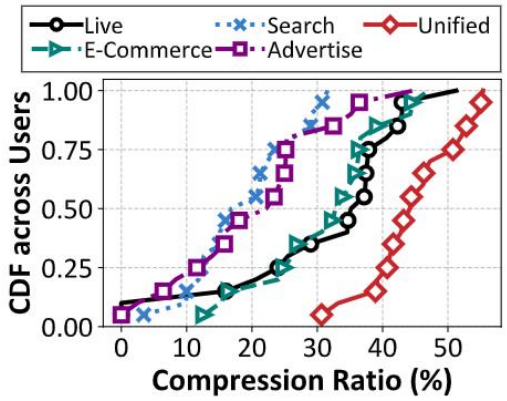
- Actual behavior logs over 14-day time periods
- Massive variance in usage patterns (1000 - 50,000 events /14 days)

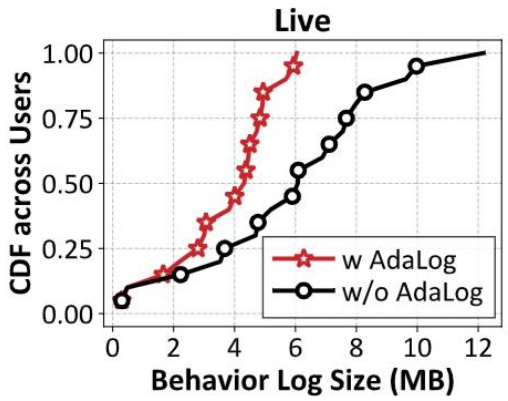


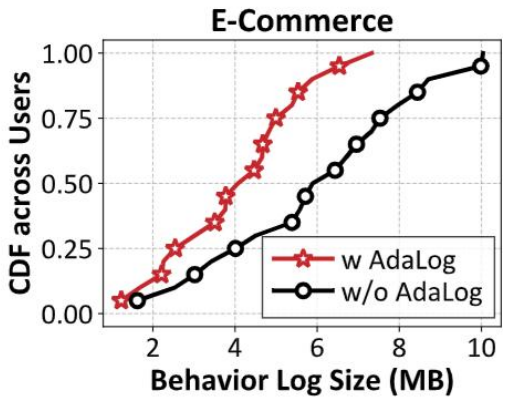


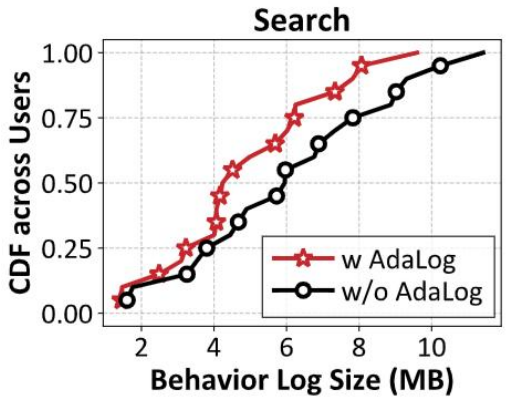
Storage Reduction

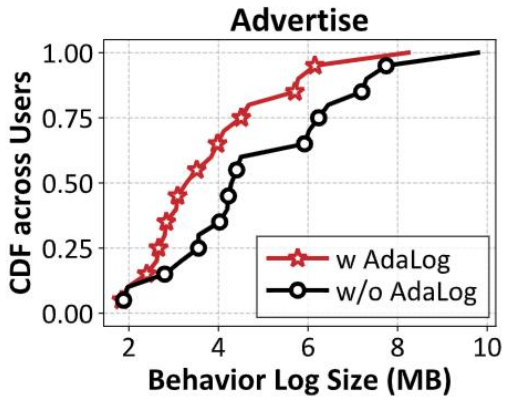
AdaLog achieves up to 44% log size reduction, allowing 1.82x more ML models to be supported

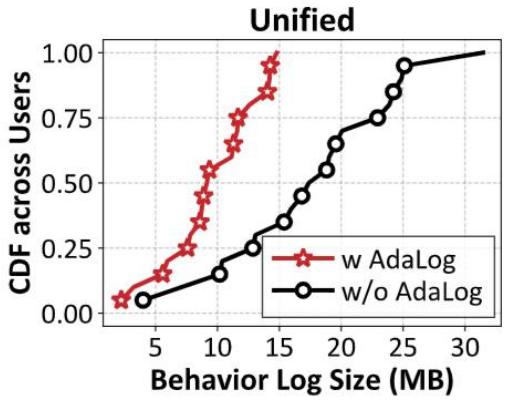








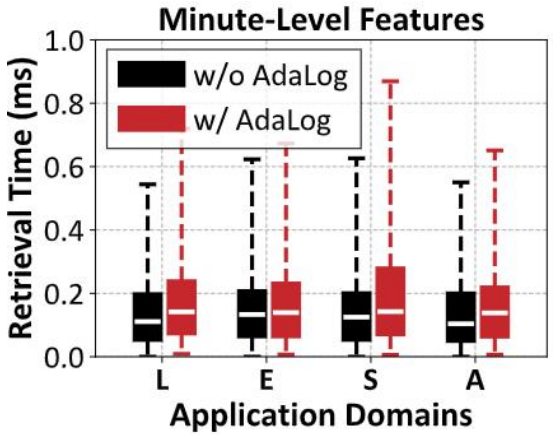


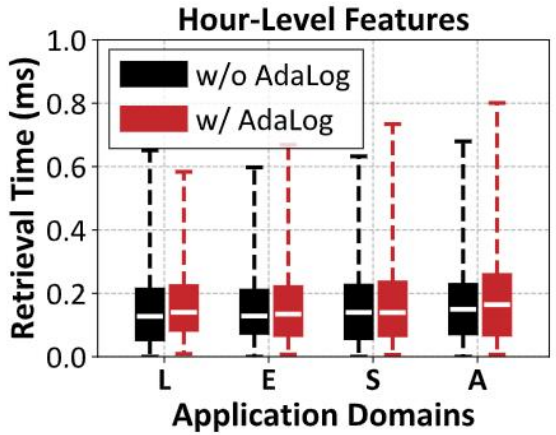


- Live Streaming: 35.1%
- E-Commerce: 32.8%
- Search: 18,9%
- Advertisements: 23.4%
- Unified: 44%

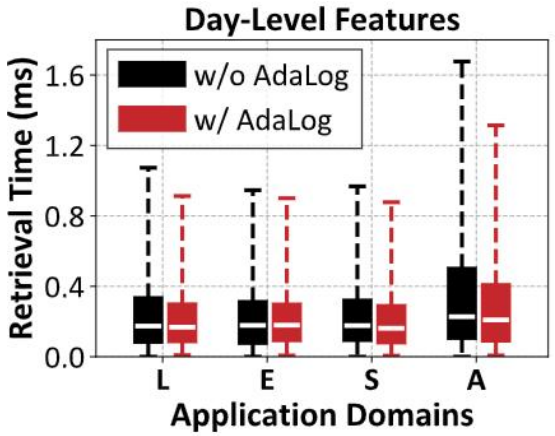
Feature Computation Speed

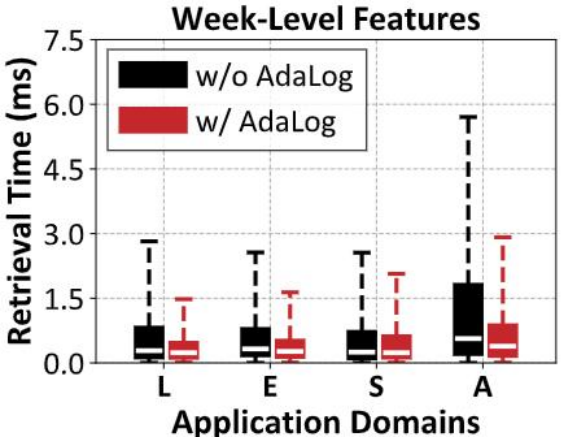
For real-time model inferences, AdaLog introduces zero latency penalties.

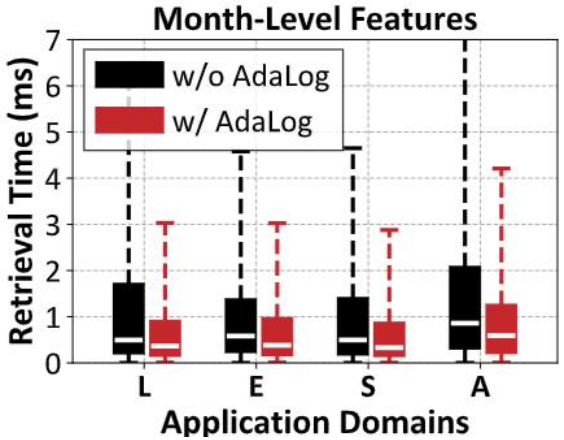




Minute and Hour-Level Features:
Near identical computation speed

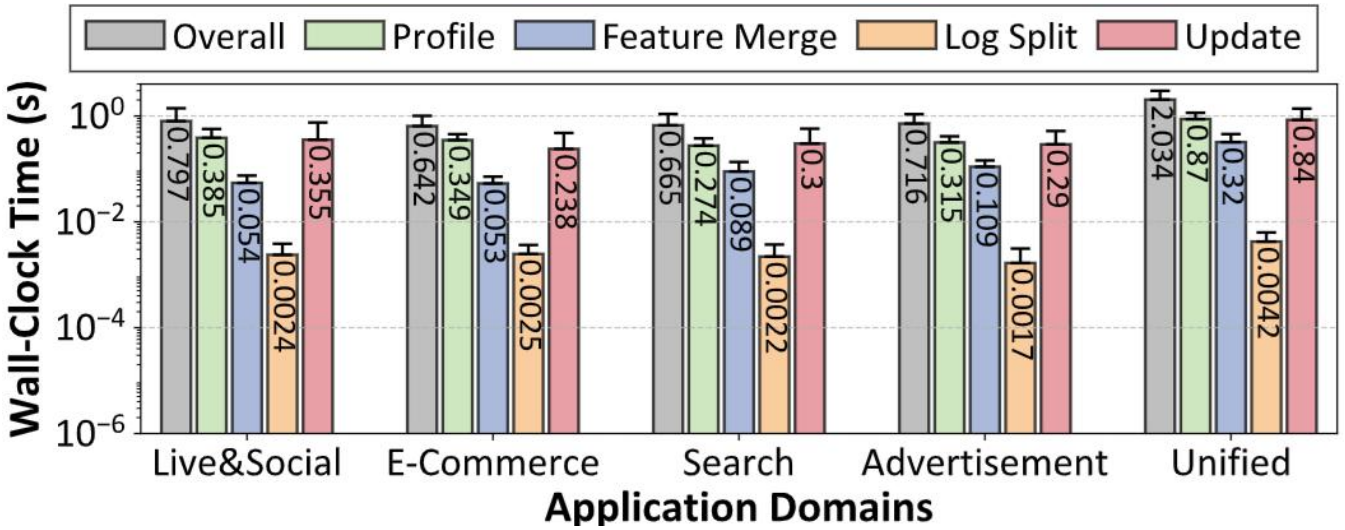


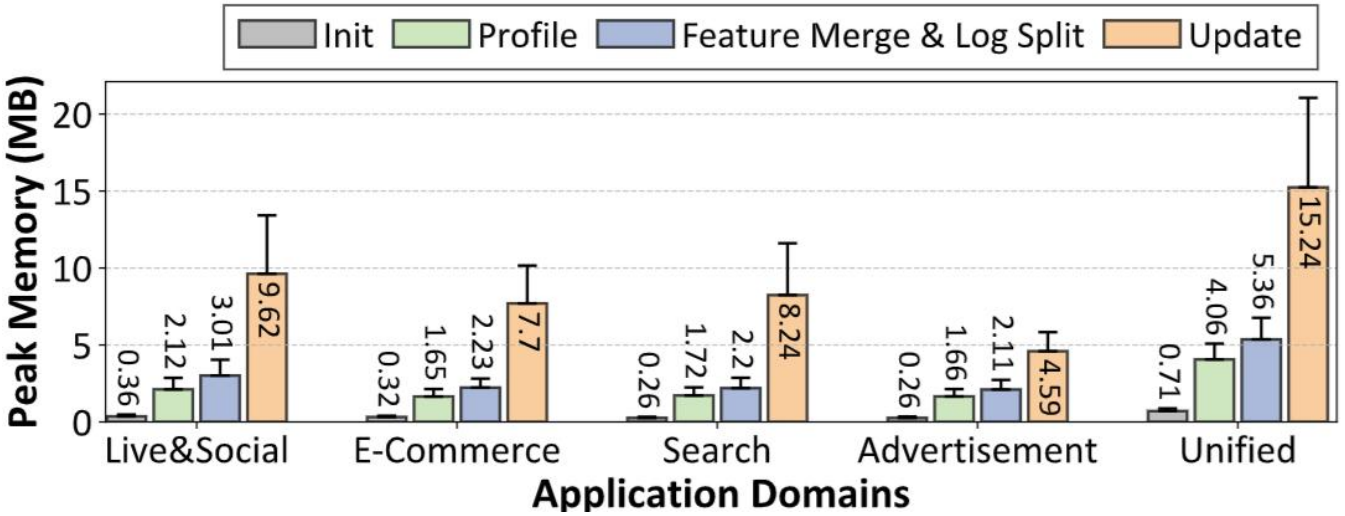




Long-Period Features:
Even faster
computation speed

System Overhead for Devices





Latency:

0.64 – 2.03 seconds across different users and apps

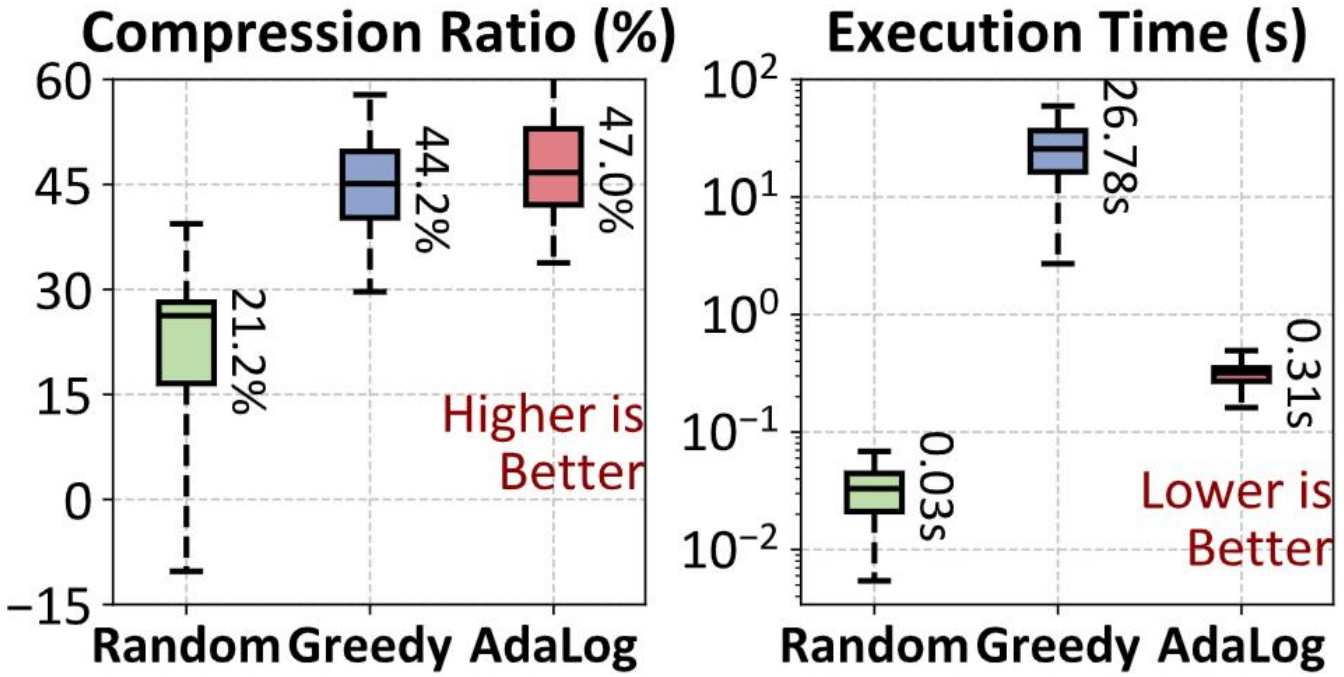
Memory Footprint:

Consistently below 20 MB

Component-wise Analysis

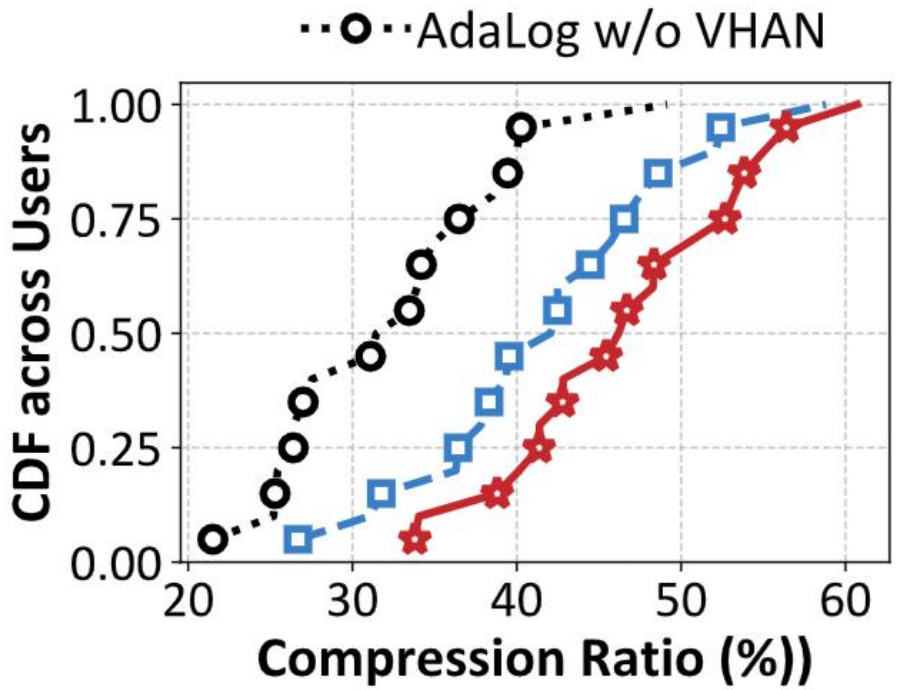
Hierarchical Merging:

- Greedy solution achieves similar compression but 86x latency
- Random approach performed 26% worse in compression



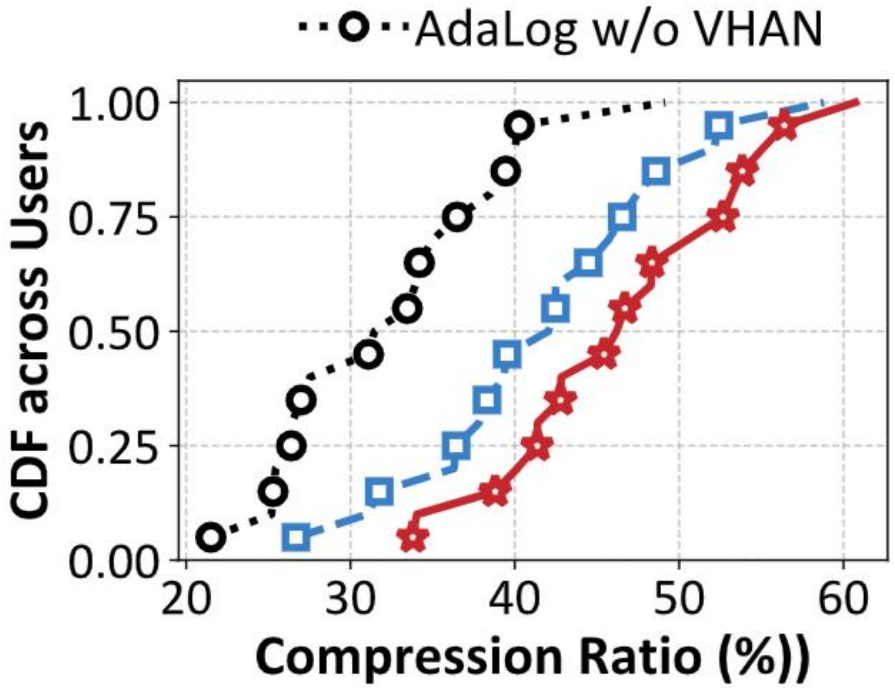
Component-wise Analysis (Cont')

Virtually Hashed Attribute:
contributes to 14% compression
and 35% sparsity reduction.



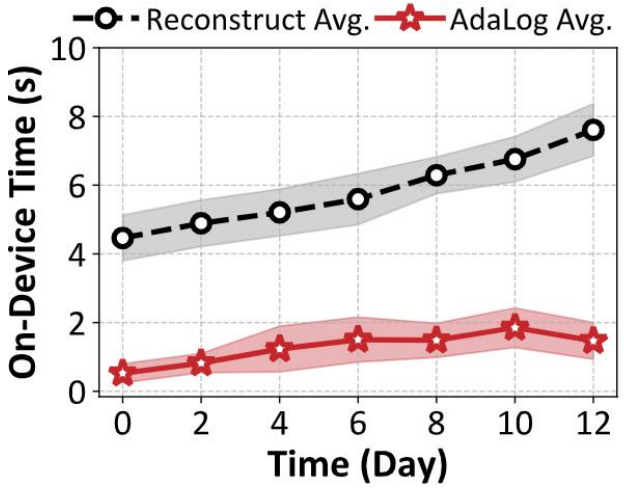
Component-wise Analysis (Cont')

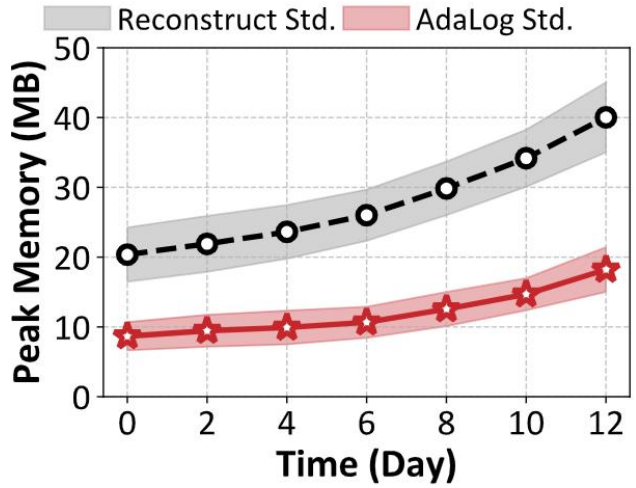
Virtually Hashed Attribute:
contributes to 14% compression and 35% sparsity reduction.



Incremental Update Mechanism:
Compared to full reconstruction, it has

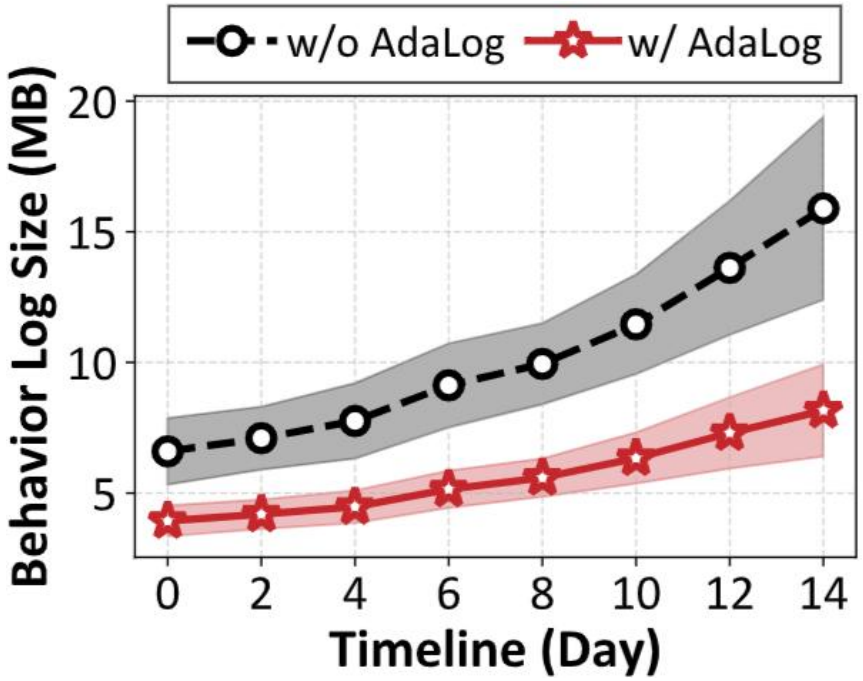
- 3.4x – 9.1 x speedup in latency
- 2.2x – 2.5x reduction in memory

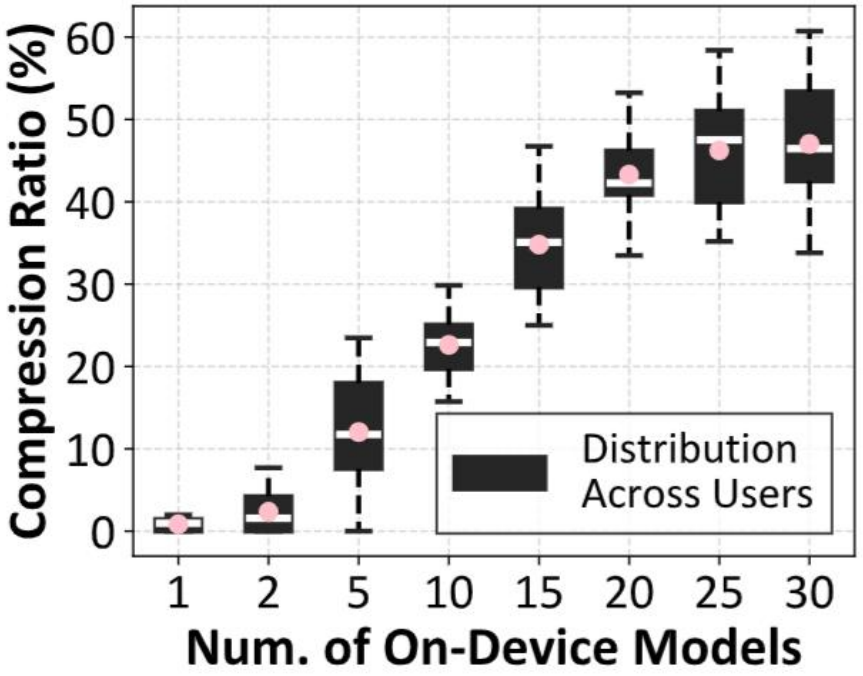




Sensitivity Analysis

AdaLog **scales linearly over time** and effectively **stabilizes compression** as the number of models grows.





Coclusion

Problem:

Unveil behavior log storage bottleneck in on-device ML development

Solution:

AdaLog optimizes behavior log storage efficiency without affecting model inference latency for ML-embedded mobile apps

Result:

Up to 44% log size reduction in online evaluation of industrial services.

Thank You for Your Attention !

Chen Gong
gongchen@sjtu.edu.cn