ETRI Electronics and Telecommunications Research Institute

Self-Driving Database Management Systems

Database meets deep learning

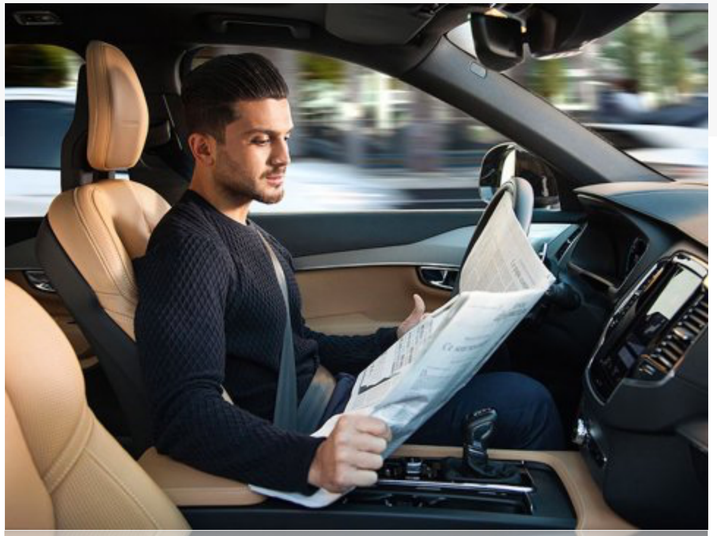
16 September 2017

Sung-Soo Kim

sungsoo@etri.re.kr

Smart Data Research Group

ETRI



References & Slide Credits

- Andrew Pavlo and *et al*, *Self-Driving Database Management Systems*, CIDR '17.
- Joy Arulraj and *et al*, *Bridging the archipelago between row-stores and column-stores for hybrid workloads*, SIGMOD'16.
- Joy Arulraj and *et al*, *Write-Behind Logging*, VLDB'16.
- Sung-Soo Kim and *et al*, *Sweet KIWI: Statistics-Driven OLAP Acceleration using Query Column Sets*, EDBT'16.
- Sung-Soo Kim and *et al*, *Flying KIWI: Design of Approximate Query Processing Engine for Interactive Data Analytics at Scale*, BigDAS'15.

Outline

- Introduction
- Problem Overview
- Self-Driving Architecture
    - Workload Classification
    - Workload Forecasting
    - Action Planning & Execution
- Experimental Results
- Related Work
- Conclusion

Self-Driving Database Management Systems

Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon
Todd C. Mowry, Matthew Perron, Ian Quah, Siddharth Santurkar, Anthony Tomasic
Skye Toor, Dana Van Aken, Ziqi Wang, Yingjun Wu*, Ran Xian, Tieying Zhang
Carnegie Mellon University, *National University of Singapore

ABSTRACT

1. INTRODUCTION

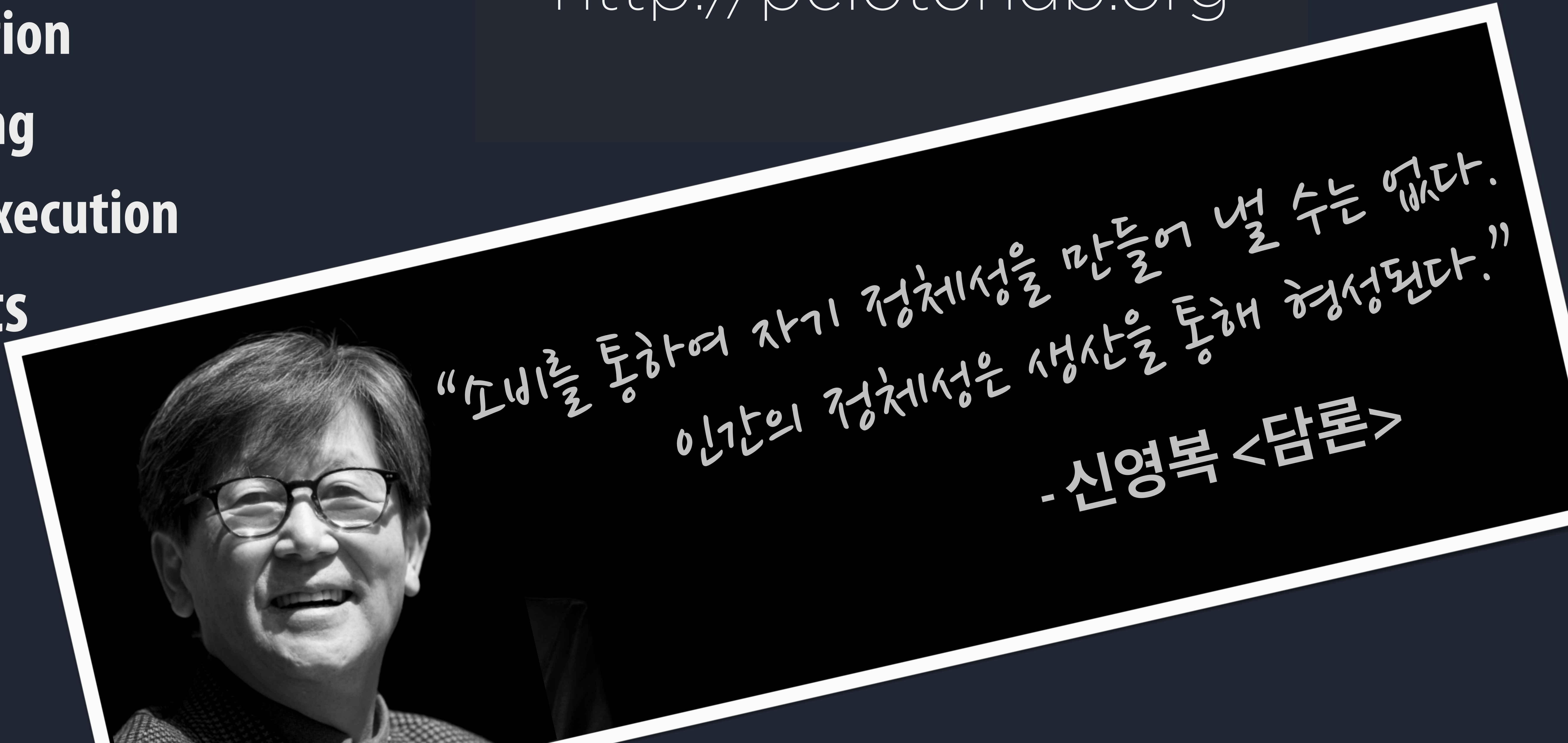
2. PROBLEM OVERVIEW

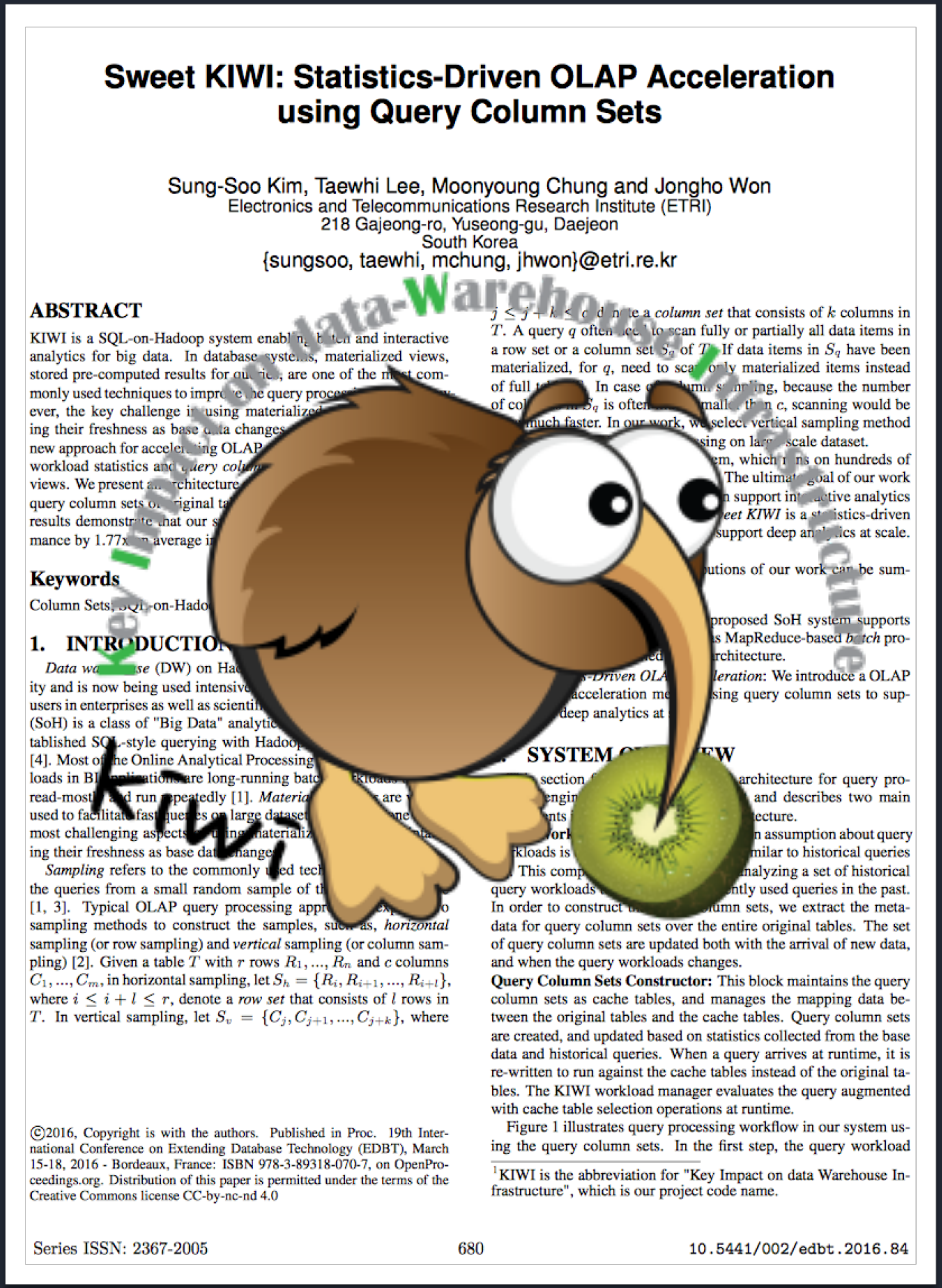
Self-Driving Database Management Systems

CIDR'17

Peloton

http://pelotondb.org





Sweet KIWI: Statistics-Driven OLAP Acceleration using Query Column Sets

Sung-Soo Kim, Taewhi Lee, Moonyoung Chung and Jongho Won
Electronics and Telecommunications Research Institute (ETRI)
218 Gajeong-ro, Yuseong-gu, Daejeon
South Korea
{sungsoo, taewhi, mchung, jhwon}@etri.re.kr

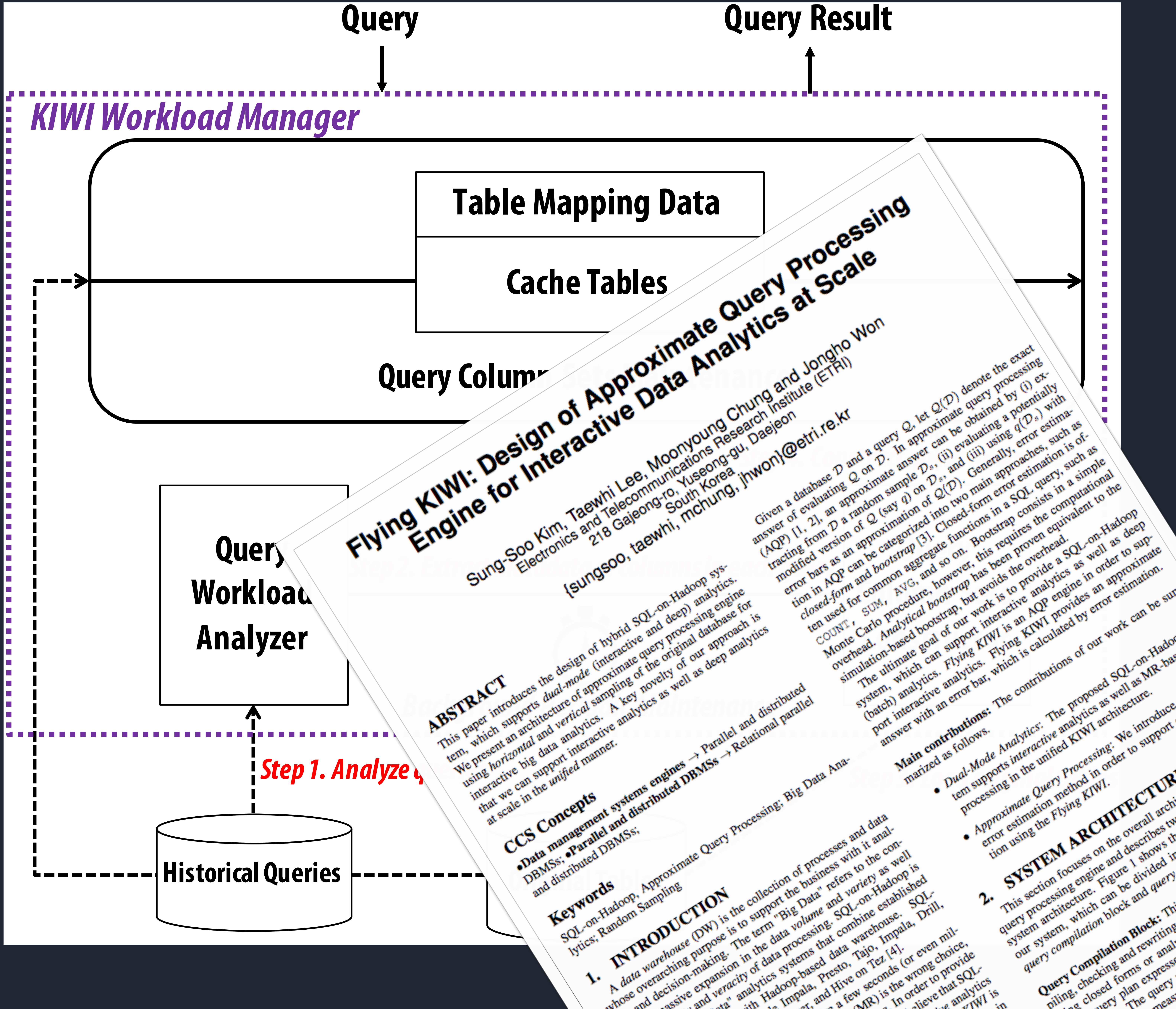
Query Column Sets Constructor: This block maintains the query column sets as cache tables, and manages the mapping data between the original tables and the cache tables. Query column sets are created, and updated based on statistics collected from the base data and historical queries. When a query arrives at runtime, it is re-written to run against the cache tables instead of the original tables. The KIWI workload manager evaluates the query augmented with cache table selection operations at runtime.

Figure 1 illustrates query processing workflow in our system using the query column sets. In the first step, the query workload

[1] KIWI is the abbreviation for "Key Impact on data Warehouse Infrastructure", which is our project code name.

©2016, Copyright is with the authors. Published in Proc. 19th International Conference on Extending Database Technology (EDBT), March 15-18, 2016 - Bordeaux, France: ISBN 978-3-89318-070-7, on OpenProceedings.org. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0

Series ISSN: 2367-2005 680 10.5441/002/edbt.2016.84



Flying KIWI: Design of Approximate Query Processing Engine for Interactive Data Analytics at Scale

Sung-Soo Kim, Taewhi Lee, Moonyoung Chung and Jongho Won
Electronics and Telecommunications Research Institute (ETRI)
218 Gajeong-ro, Yuseong-gu, Daejeon
South Korea
{sungsoo, taewhi, mchung, jhwon}@etri.re.kr

Sweet KIWI: Statistics-Driven OLAP Acceleration using Query Column Sets

EDBT 2016

Why?

- DBMSs are now the critical part of every *data-intensive* application.
- Exsisting *automated tuning tools* require too much human intervention.
- Average *salary* for database administrators (DBAs) in 2015 was **$81,710**.

BUREAU OF LABOR STATISTICS

Follow Us | What's New | Release Calendar | Blog

Search BLS.gov

Home | Subjects | Data Tools | Publications | Economic Releases | Students | Beta

OOH HOME | OCCUPATION FINDER | OOH FAQ | OOH GLOSSARY | A-Z INDEX | OOH SITE MAP | EN ESPAÑOL

OCCUPATIONAL OUTLOOK HANDBOOK

Search Handbook Go

Computer and Information Technology >

Database Administrators

EN ESPAÑOL PRINTER-FRIENDLY

Summary | What They Do | Work Environment | How to Become One | Pay | Job Outlook | State & Area Data | Similar Occupations | More Info

Summary

Quick Facts: Database Administrators	
2015 Median Pay	$81,710 per year $39.29 per hour
Typical Entry-Level Education	Bachelor's degree
Work Experience in a Related Occupation	Less than 5 years
On-the-job Training	None
Number of Jobs, 2014	120,000
Job Outlook, 2014-24	11% (Faster than average)
Employment Change, 2014-24	13,400

Database administrators ensure that data are available to many different users.

What Database Administrators Do

Database administrators (DBAs) use specialized software to store and organize data, such as financial information and customer shipping records. They make sure that data are available to users and are secure from unauthorized access.

Self-Driving

A DBMS that can configure, tune, and optimize itself without any human intervention.

Can We Automate This?

YES

DATABASE DESIGN

DATA PLACEMENT

QUERY OPTIMIZATION

KNOB CONFIGURATION

BACK-UP & RECOVERY

PROVISIONING

NO

SECURITY & ACLs

DATA INTEGRATION

UNPLANNED HALTS

VERSION CONTROL

What's New?

Previous efforts are reactive & human-driven.

A self-driving DBMS has to be predictive.

Why Now?

Recent advancements in hardware and deep neural networks make autonomous operation now possible.

Peloton

Non-volatile

- IN-MEMORY
- OLTP+OLAP
- LLVM EXEC
- AUTONOMOUS

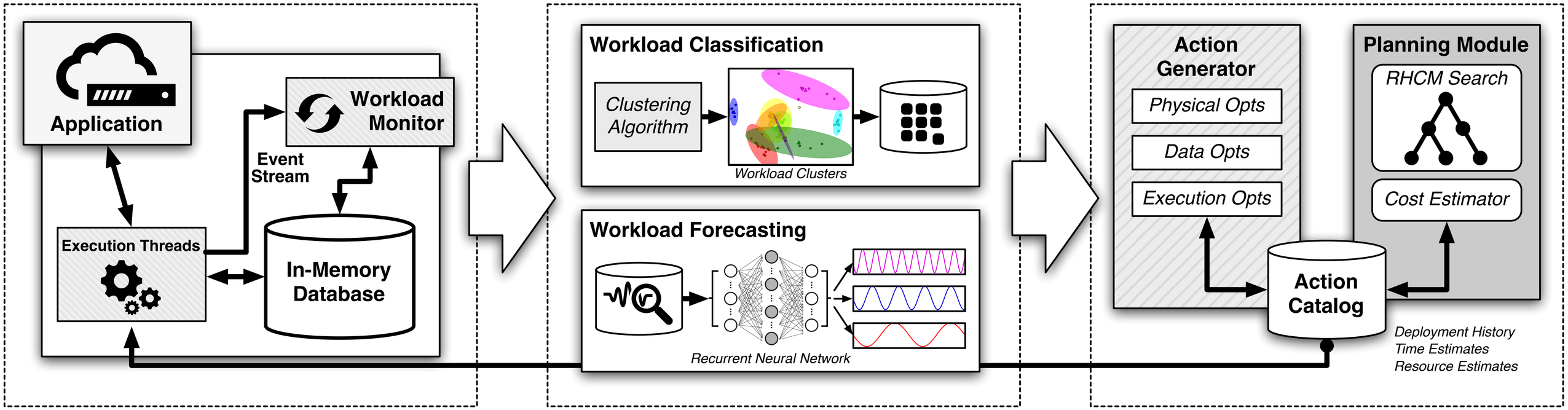
The Brain

INTEGRATED DEEP LEARNING FRAMEWORK TO MODEL, PREDICT, AND OPTIMIZE HTAP DATABASE WORKLOADS.

Self-Driving Database Management Systems
CIDR 2017

Features
(1) *a query's runtime metrics*
(2) *a query's logical semantics*

DBSCAN algorithm

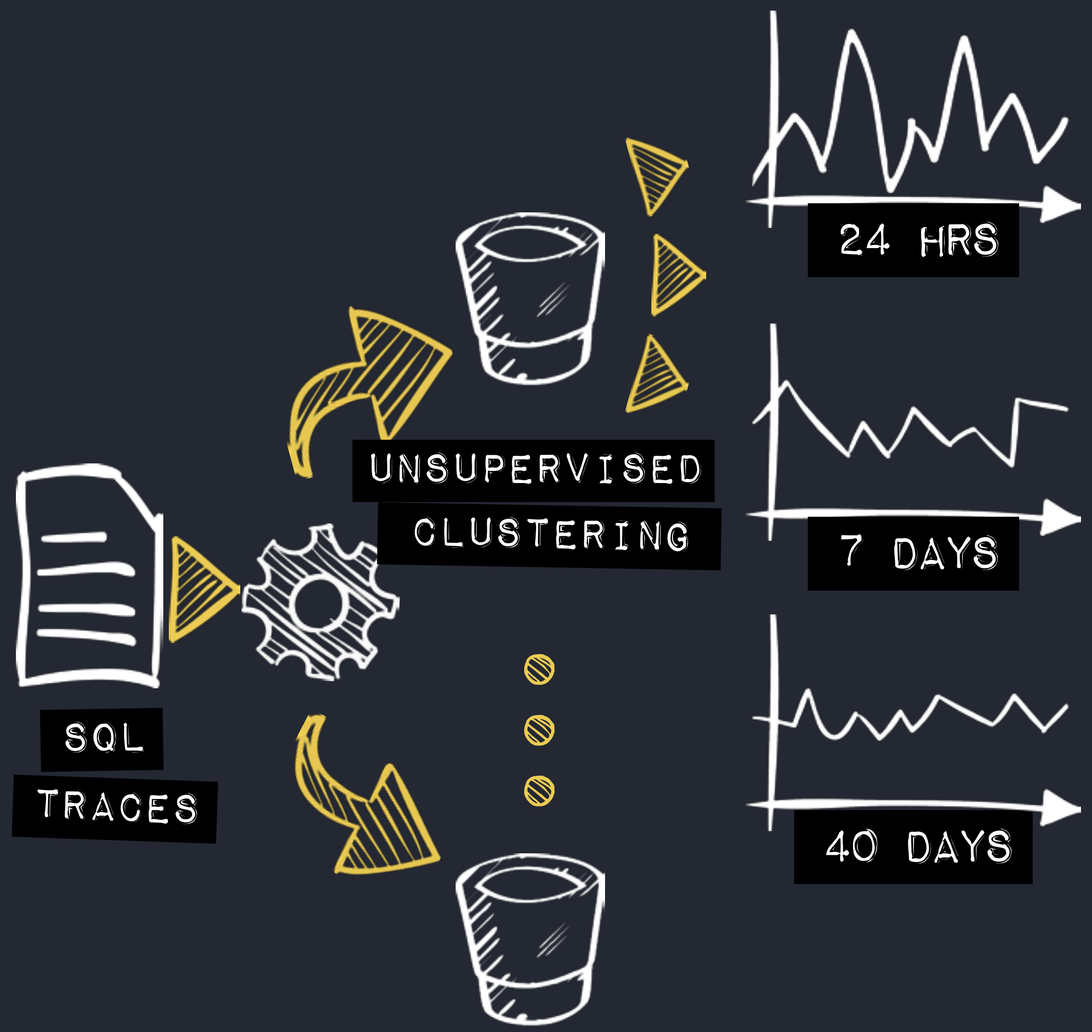


Recurrent neural networks (RNNs)

Previous attempts at autonomous systems:
Auto-regressive-moving average model (ARMA)

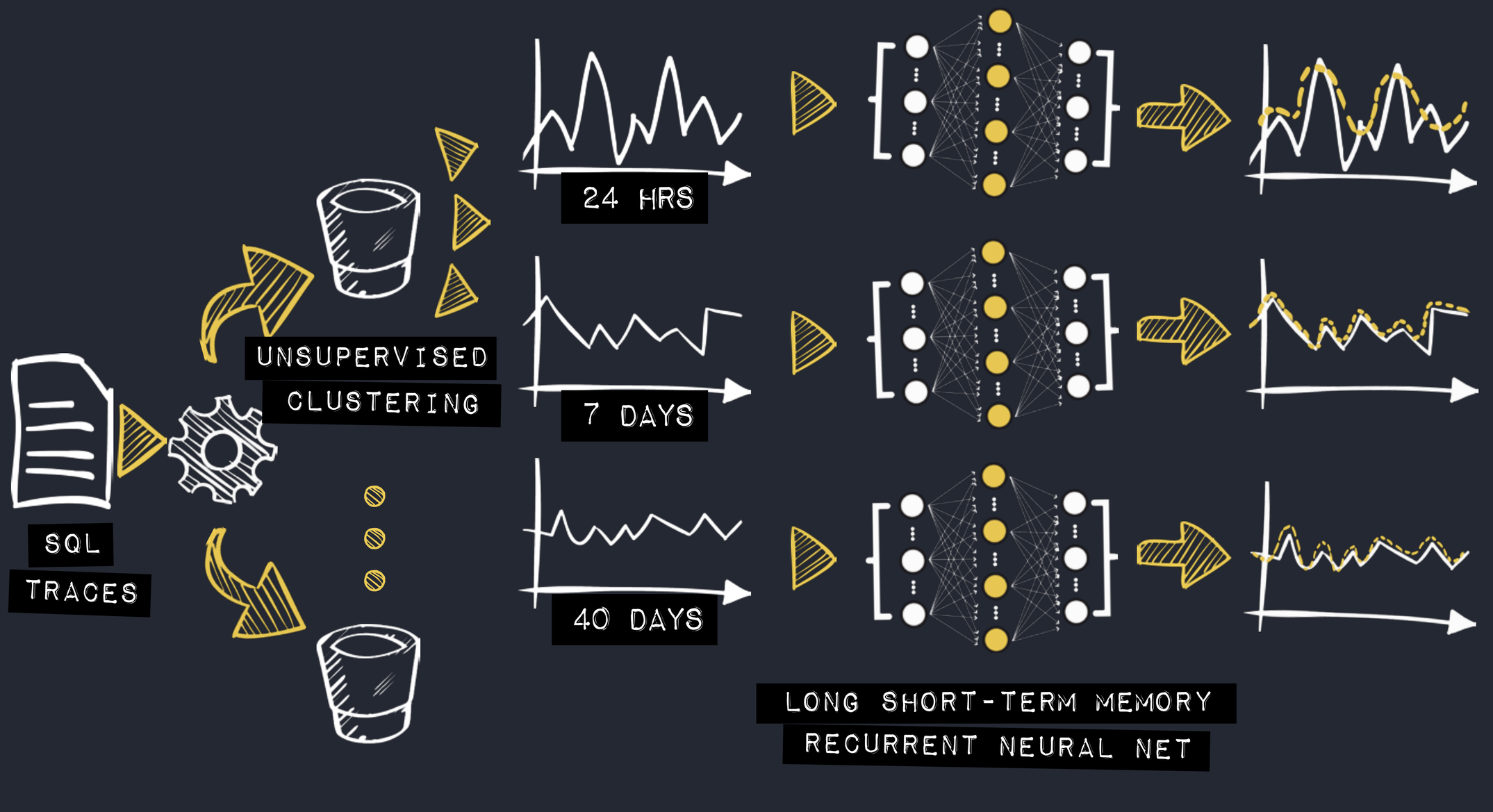
Long short-term memory (LSTM)

Workload Categorization



Workload Categorization

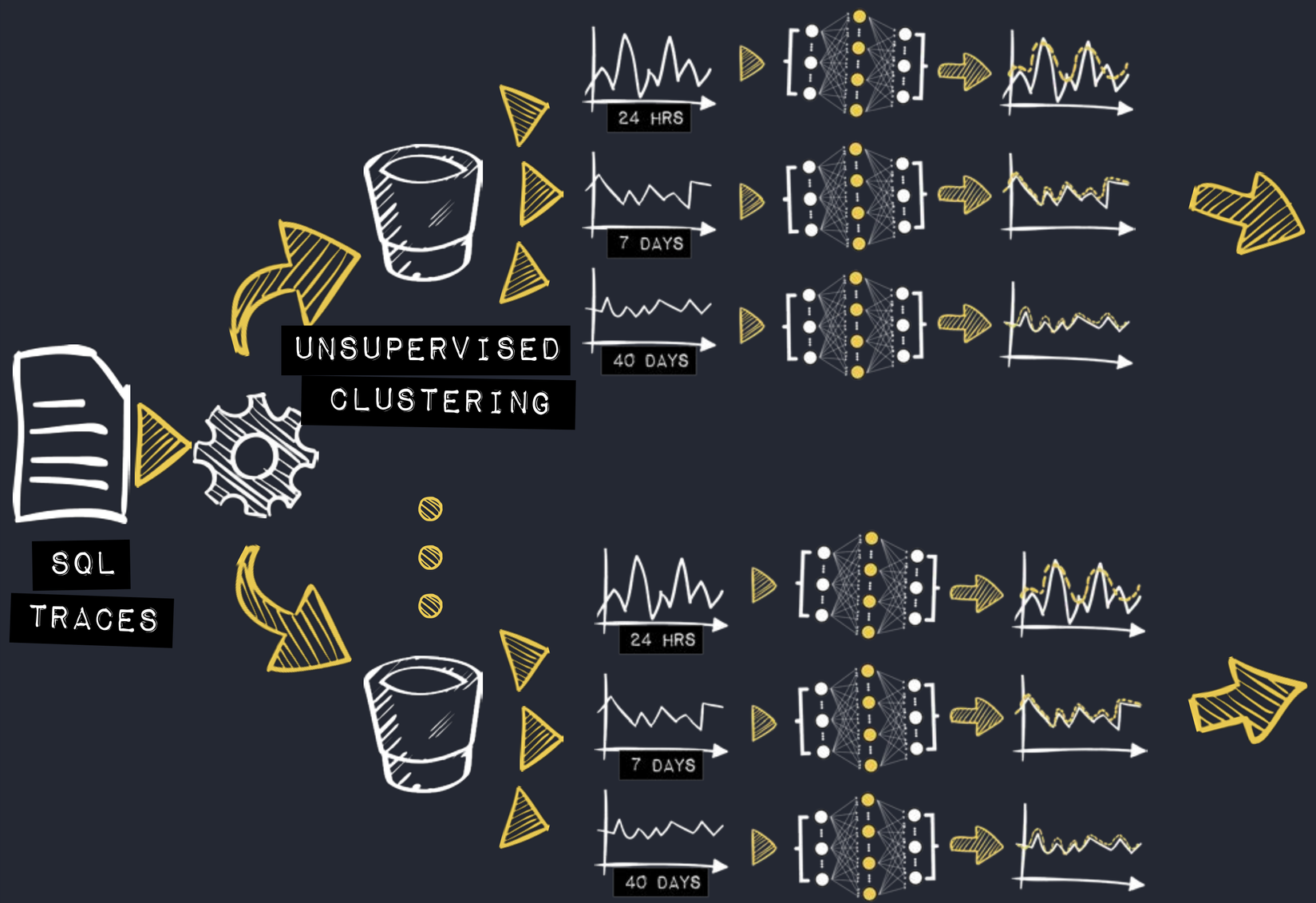
Workload Forecasting



Workload Categorization

Workload Forecasting

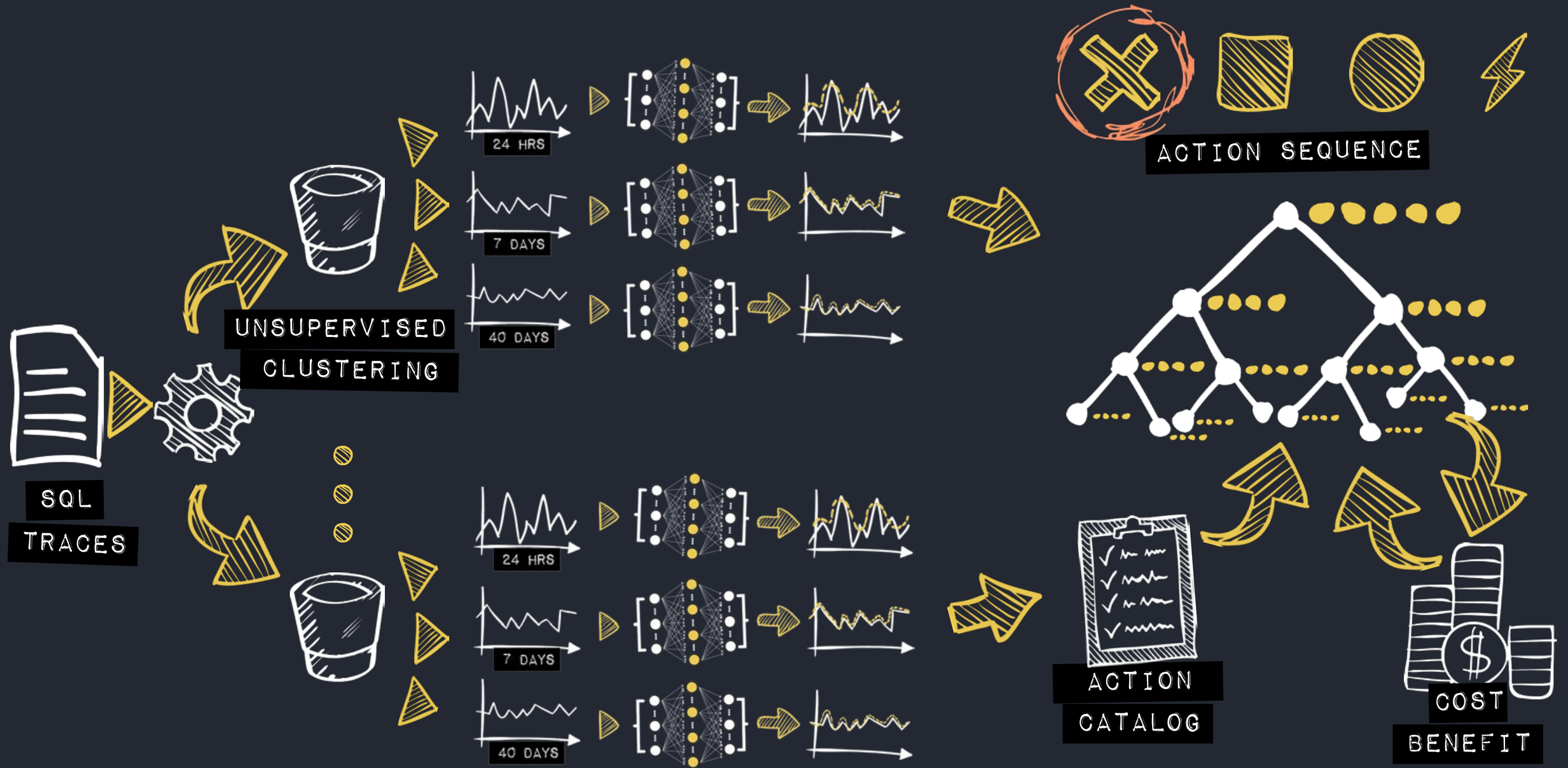
Optimization Planning



Workload Categorization

Workload Forecasting

Optimization Planning

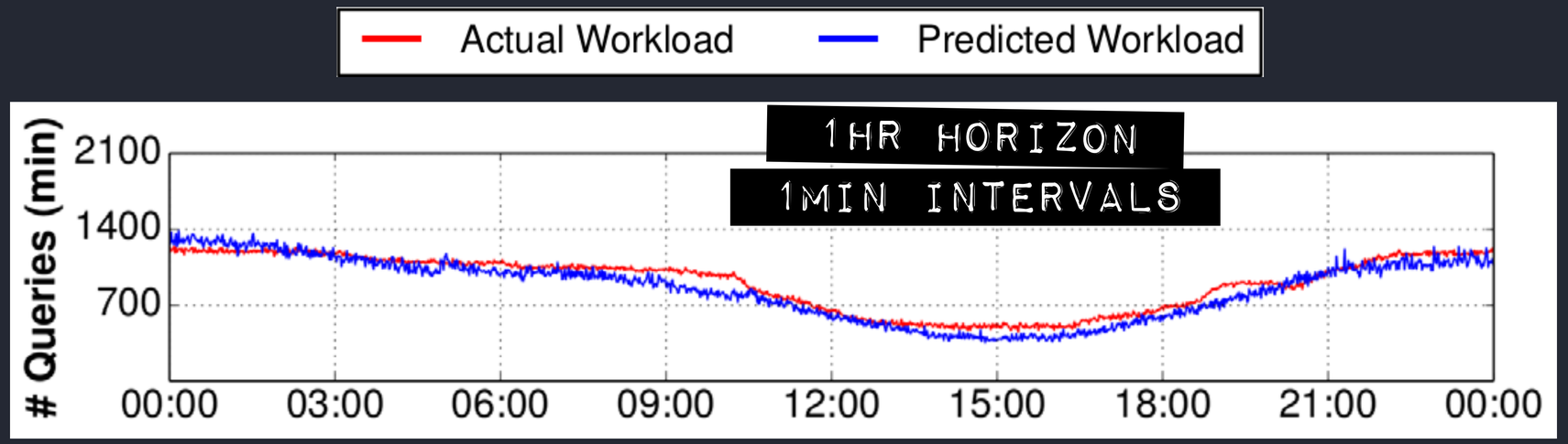


Evaluation

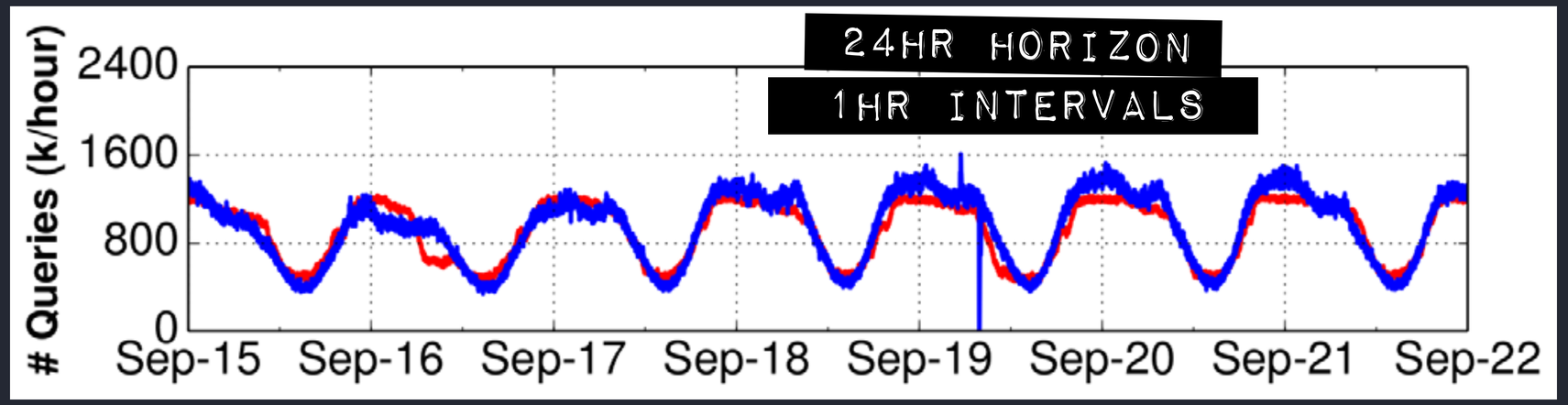
Synthetic workload based on Reddit traffic data.

Forecast with TensorFlow.

Adaptive storage.



ERROR RATE: 14.7%
CPU TRAINING: 25MIN
PROBE: 2MS
UPDATE: 5MS
SIZE: 2MB



ERROR RATE: 17.9%
CPU TRAINING: 18MIN
PROBE: 2MS
UPDATE: 5MS
SIZE: 2MB

TensorFlow

Adaptive Storage

Change the layout of data over time based on how it is accessed.

Bridging the Archipelago Between Row-Stores and Column-Stores for Hybrid Workloads
SIGMOD 2016

OLTP

```
UPDATE myTable
   SET A = 123,
       B = 456,
       C = 789
 WHERE D = “xxx”
```

OLAP

```
SELECT AVG(B)
  FROM myTable
 WHERE C < “yyy”
```

ORIGINAL TABLE

A	B	C	D

Hot

Cold

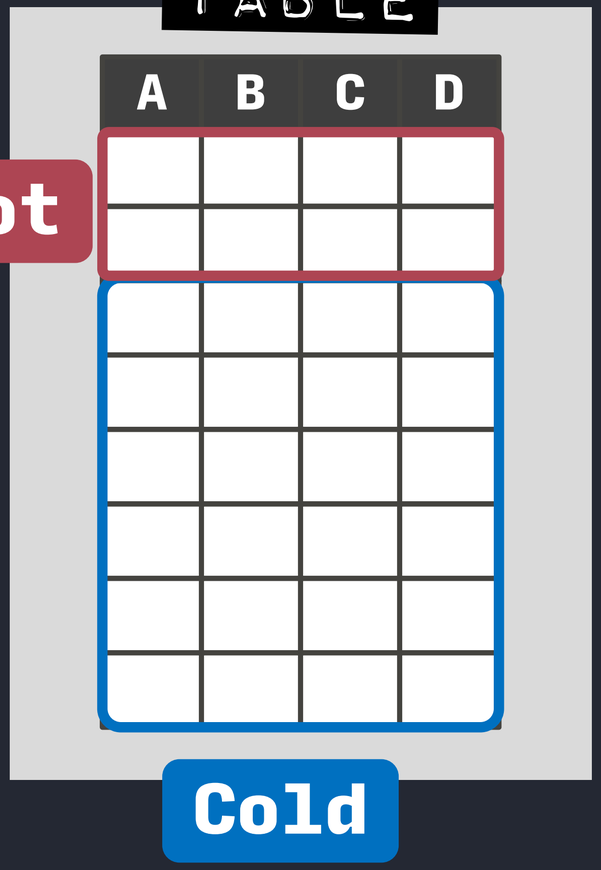
OLTP

```
UPDATE myTable
   SET A = 123,
       B = 456,
       C = 789
 WHERE D = “xxx”
```

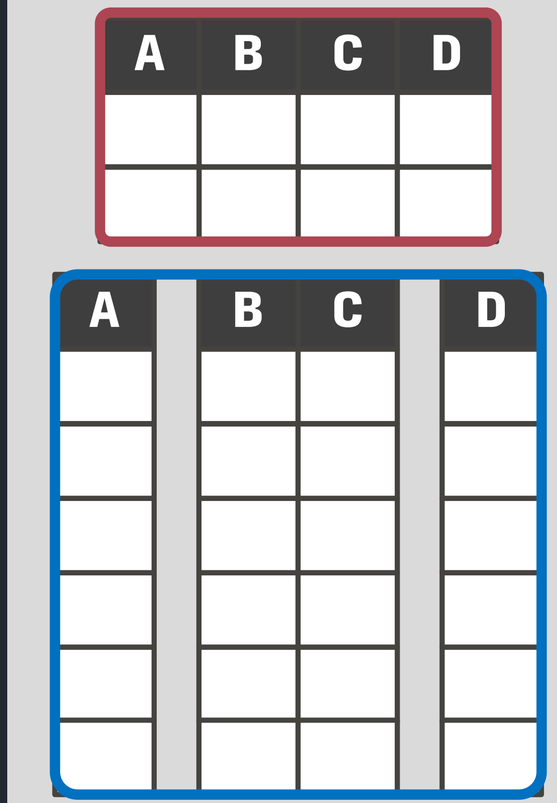
OLAP

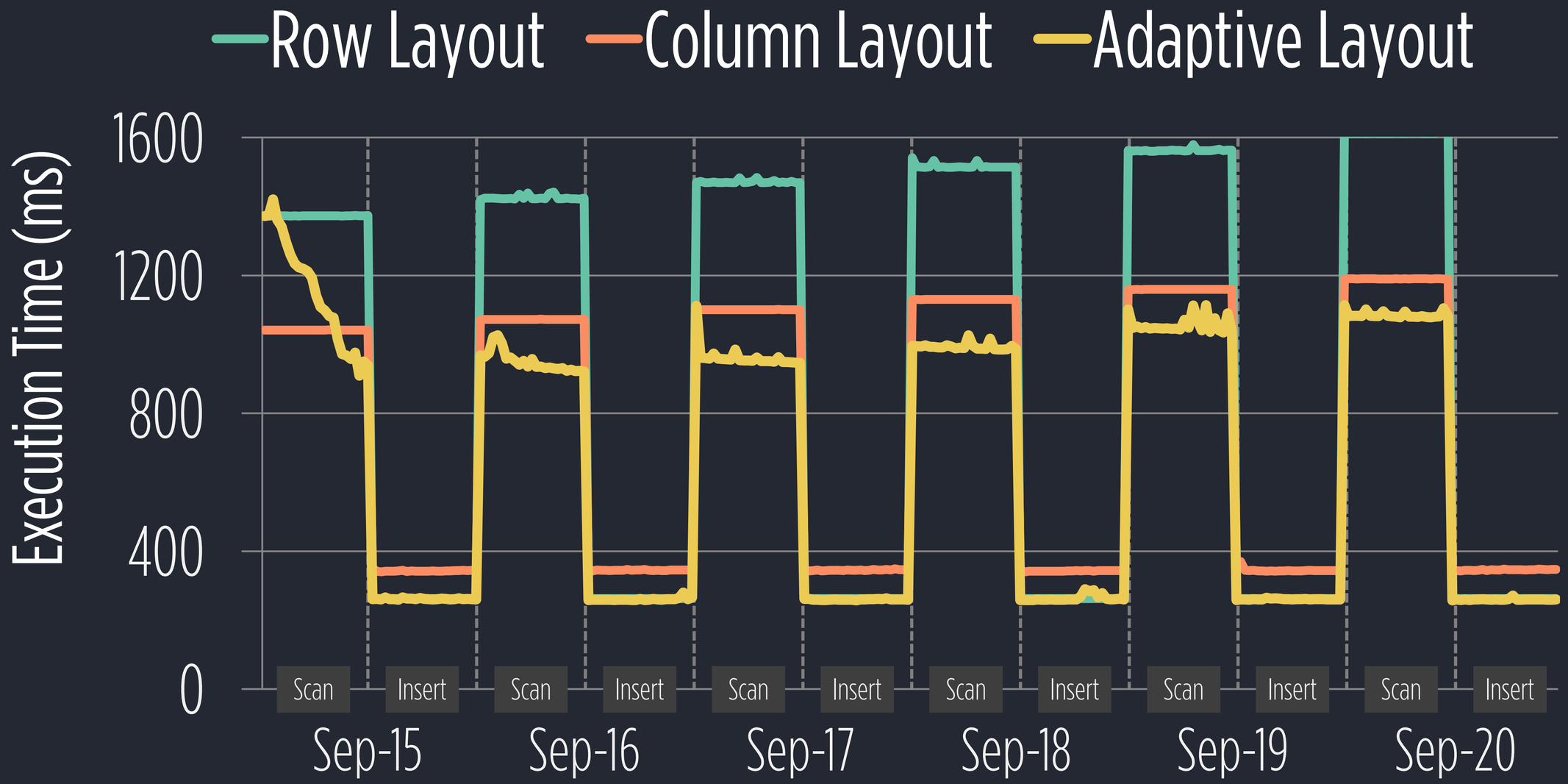
```
SELECT AVG(B)
  FROM myTable
 WHERE C < “yyy”
```

ORIGINAL TABLE



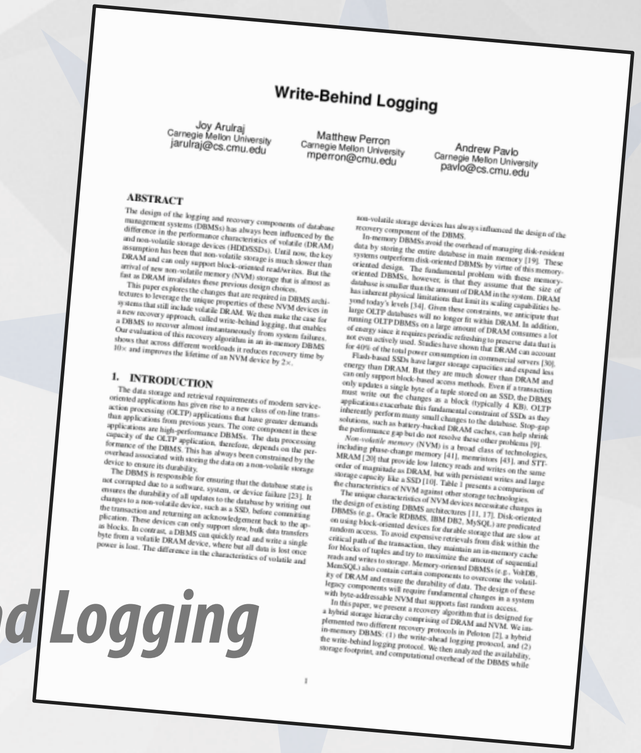
ADAPTED TABLE




Row Layout
Column Layout
Adaptive Layout
Execution Time (ms)
1600
1200
800
400
0
Scan
Insert
Scan
Insert
Scan
Insert
Scan
Insert
Scan
Insert
Scan
Insert
Sep-15
Sep-16
Sep-17
Sep-18
Sep-19
Sep-20

NVM Optimizations

» Avoid the overhead of DBMS recovery
» Larger-than-Memory databases
» Write-Behind Logging
» Data Tiering



Write-Behind Logging

Joy Arulraj
Carnegie Mellon University
jarulraj@cs.cmu.edu

Matthew Perron
Carnegie Mellon University
mperron@cmu.edu

Andrew Pavlo
Carnegie Mellon University
pavlo@cs.cmu.edu

ABSTRACT

The design of the logging and recovery components of database management systems (DBMSs) has always been influenced by the difference in the performance characteristics of volatile (DRAM) and non-volatile storage devices (HDD/SSDs). Until now, the key assumption has been that non-volatile storage is much slower than DRAM and can only support block-oriented read/write. But the arrival of new non-volatile memory (NVM) storage that is almost as fast as DRAM invalidates these previous design choices.

This paper explores the changes that are required in DBMS architectures to leverage the unique properties of these NVM devices in systems that still include volatile DRAM. We then make the case for a new recovery approach, called write-behind logging, that enables a DBMS to recover almost instantaneously from system failures. Our evaluation of this recovery algorithm in an in-memory DBMS shows that across different workloads it reduces recovery time by 100× and improves the lifetime of an NVM device by 2×.

1. INTRODUCTION

The data storage and retrieval requirements of modern service-oriented applications has given rise to a new class of on-line transaction processing (OLTP) applications that have greater demands than applications from previous years. The core component in these applications are high-performance DBMSs. The data processing capacity of the OLTP application, therefore, depends on the performance of the DBMS. This has always been constrained by the overhead associated with storing the data on a non-volatile storage device to ensure its durability.

The DBMS is responsible for ensuring that the database state is not corrupted due to a software, system, or device failure [23]. It ensures the durability of all updates to the database by writing out changes to a non-volatile device, such as a SSD, before committing the transaction and returning an acknowledgment back to the application. These devices can only support slow, bulk data transfers as blocks. In contrast, a DBMS can quickly read and write a single byte from a volatile DRAM device, where bit all data is lost once power is lost. The difference in the characteristics of volatile and non-volatile storage devices has always influenced the design of the recovery component of the DBMS.

In-memory DBMSs avoid the overhead of managing disk-resident data by storing the entire database in main memory [19]. These systems outperform disk-oriented DBMSs by virtue of this memory-oriented design. The fundamental problem with these memory-oriented DBMSs, however, is that they assume that the size of database is smaller than the amount of DRAM in the system. DRAM has inherent physical limitations that limit its scaling capabilities beyond today's levels [34]. Given these constraints, we anticipate that large OLTP databases will no longer fit within DRAM. In addition, running OLTP DBMSs on a large amount of DRAM consumes a lot of energy since it requires periodic refreshing to preserve data that is not even actively used. Studies have shown that DRAM can account for 40% of the total power consumption in commercial servers [30].

Flash-based SSDs have larger storage capacities and expend less energy than DRAM. But they are much slower than DRAM and can only support block-based access methods. Even if a transaction only updates a single byte of a tuple stored on an SSD, the DBMS must write out the changes as a block (typically 4 KB). OLTP applications exacerbate this fundamental constraint of SSDs as they inherently perform many small changes to the database. Stop-gap solutions, such as battery-backed DRAM caches, can help shrink the performance gap but do not resolve these other problems [9].

Non-volatile memory (NVM) is a broad class of technologies, including phase-change memory [41], memristors [45], and STT-MRAM [20] that provide low latency reads and writes on the same order of magnitude as DRAM, but with persistent writes and large storage capacity like a SSD [10]. Table 1 presents a comparison of the characteristics of NVM against other storage technologies.

The unique characteristics of NVM require us to rethink the design of existing DBMS architectures [11, 17]. Disk-oriented DBMSs (e.g., Oracle RDBMS, IBM DB2, MySQL) are predicated on using block-oriented devices for durable storage that are slow at random access. To avoid expensive retrievals from disk within the critical path of the transaction, they maintain an in-memory cache for blocks of tuples and try to maximize the amount of sequential reads and writes to storage. Memory-oriented DBMSs (e.g., VoltDB, MemSQL) also contain certain components to overcome the volatility of DRAM and ensure the durability of data. The design of these legacy components will require fundamental changes in a system with byte-addressable NVM that supports fast random access.

In this paper, we present a recovery algorithm that is designed for a hybrid storage hierarchy comprising of DRAM and NVM. We implemented two different recovery protocols in Peloton [2], a hybrid in-memory DBMS: (1) the write-ahead logging protocol, and (2) the write-behind logging protocol. We then analyzed the availability, storage footprint, and computational overhead of the DBMS while

Write-Behind Logging

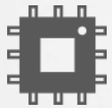
VLDB2016

Write-Behind Logging

```
UPDATE myTable
   SET A = 123,
WHERE C = “xxx”
```

NVM Heap

A	B	C	D
x1			
x2			
x3			



PCOMMIT

In-Memory Heap

ver	A	B	C
x1			
x2			
x3			

NVM Log

```
001:Txn1-0x0001
```

Conclusion

- The demand for an *autonomous DBMS*
- The *self-driving architecture* of the Peloton DBMS

- They argued that the ambitious system that they proposed is *now possible*
  - due to *advancements in deep neural networks*,
  - *improved hardware*,
  - and *high-performance database architectures*.


рахмат
danke
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
faafetai lava
mersi
kia ora
barka
welalin
tack
spas
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
kiitos
dankie
nanni
nandri
dhanyavad
bedankt
hvala
mauruuru
köszönöm
thank you
gracias
хвала
asante
manana
obrigada
tenki
bayarlalaa
gràçie
enkosi
dankon
aciú
akun
djiere dieuf
tau
mochchakkeram
murakoze
дякую
mamnun
chokrane
dziękuję
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
go raibh maith agat
sobodi
dēkuji
obrigado
sagolun
sukriya
kop khun krap
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
shukriya
mercé
мерси
kam sah hamnida
rahmat
terima kasih
rahmet
ευχαριστώ
diolch
dhanyavadagalu
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci