Storing and Accessing Lifelog Data with MySQL & NoSQL (MongoDB) Databases

LUZ LOPEZ

DIVISION OF SCIENCE AND MATHEMATICS

UNIVERSITY OF MINNESOTA, MORRIS

NOVEMBER 18TH, 2017

Outline

1. INTRODUCTION
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
4. CONCLUSION

Outline

1. INTRODUCTION
   i. What is *lifelogging*?
   ii. Lifelog Mashup
   iii. Lifelog Common Data Model (LLCDM)
   iv. Lifelog API (LLAPI)
   v. Previous Work Limitations
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL IMPLEMENTATION WITH MongoDB
4. CONCLUSION

What is Lifelogging? /Importance /Lifelog Mashup

Lifelogging:

Also known as "life catching"

A social act to record and share human life events in an open and public form [1,2]

Lifelog Mashup:

Integrating scattered lifelogs would implement more sophisticated and value-added services, than using them separately [1]

- Personal
  - Personal health achievements
  - Productivity
  - Self-enhancement
- Public
  - Memories
  - Photos
  - Connections

Lifelog Common Data Model

LLCDM:

Lifelog common data model prescribes a generic data schema for lifelog records, which does not rely on any specific lifelog service.

Designed with standard attributes of

what, who(m), why, where, how. [2]

Importance of LLCDM

Data record of Twitter

{
“created_on”: “Friday Jul 05 2013” ”time”: "03:45:35+000”,
“id”: 33531553508768450O2,
“text”: “Working outside today”,
“source”:“<ahref=http://twitter.com/,
…
“geo”:{
“type”: “Point”
“coordinates”: [32.8753586, 135.874874]
}
”coordinates”: {…}
}

Data record of SensorLoggingService

{
Time:”12:46:57”,
…
User: “koupe”,
Weather: “Sunny”,
TempF: 76.73,
Brightness: 310,
Temperature: 26.6,
…
Id: 23654,
Date: “2013-07-05”
}

Importance of LLCDM

Data record of Twitter

{
“created_on”: “Friday Jul 05 2013” ”time”: "03:45:35+000”,
“id”: 33531553508768450O2,
“text”: “Working outside today”,
“source”:“<ahref=http://twitter.com/,
…
“geo”:{
“type”: “Point”
“coordinates”: [32.8753586, 135.874874]
}
”coordinates”: {…}
}

Data record of SensorLoggingService

{
Time:”12:46:57”,
…
User: “koupe”,
Weather: “Sunny”,
TempF: 76.73,
Brightness: 310,
Temperature: 26.6,
…
Id: 23654,
Date: “2013-07-05”
}

Lifelog Mashup API

LLAPI:

- *Lifelog mashup API* is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.

Lifelog Mashup API

LLAPI:

- *Lifelog mashup API* is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.
- Using **getLifeLog()** heterogeneous lifelogs can be accessed uniformly without proprietary knowledge of lifelog services.

Lifelog Mashup API

LLAPI:

- *Lifelog mashup API* is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.
- Using **getLifeLog()** heterogeneous lifelogs can be accessed uniformly without proprietary knowledge of lifelog services.

Using getLifeLog example wrapping an **SQL** statement [2]:

getLifeLog(s_date, e_date, s_time, e_time, user, party, object, location, application, device, select)

Lifelog Mashup API

LLAPI:

- *Lifelog mashup API* is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.
- Using **getLifeLog()** heterogeneous lifelogs can be accessed uniformly without proprietary knowledge of lifelog services.

Using getLifeLog example wrapping an **SQL** statement [2]:

getLifeLog(s_date, e_date, s_time, e_time, user, party, object, location, application, device, select)

Parameters:

s_date, e_date : Query of <date> (start, end)

s_time, e_time : Query of <time> (start, end)

user, party, object: Query of <user, party, object >

location : Query of <location>

application : Query of <application>

service : Query of <device>

select : Query of <select>

Lifelog Mashup Platform

fitbit

tumblr.

facebook

last.fm

twitter

Devices and Web services

Lifelog Mashup Platform

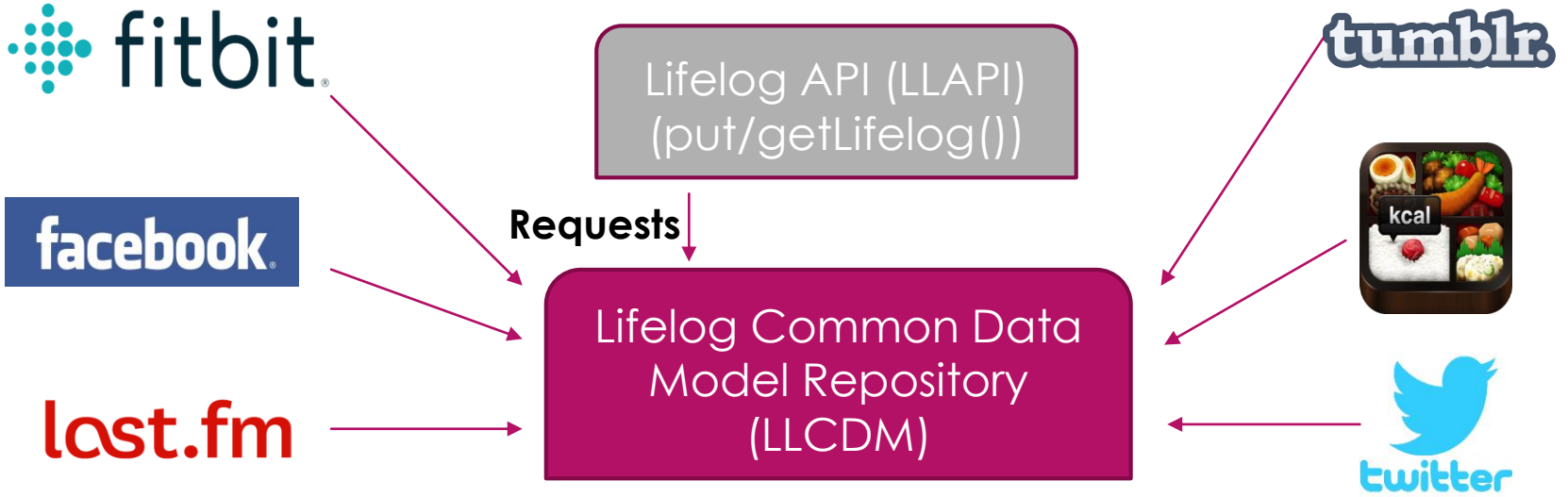
Lifelog Common Data Model Repository (LLCDM)

Introduce the LLCDM

Lifelog Mashup Platform

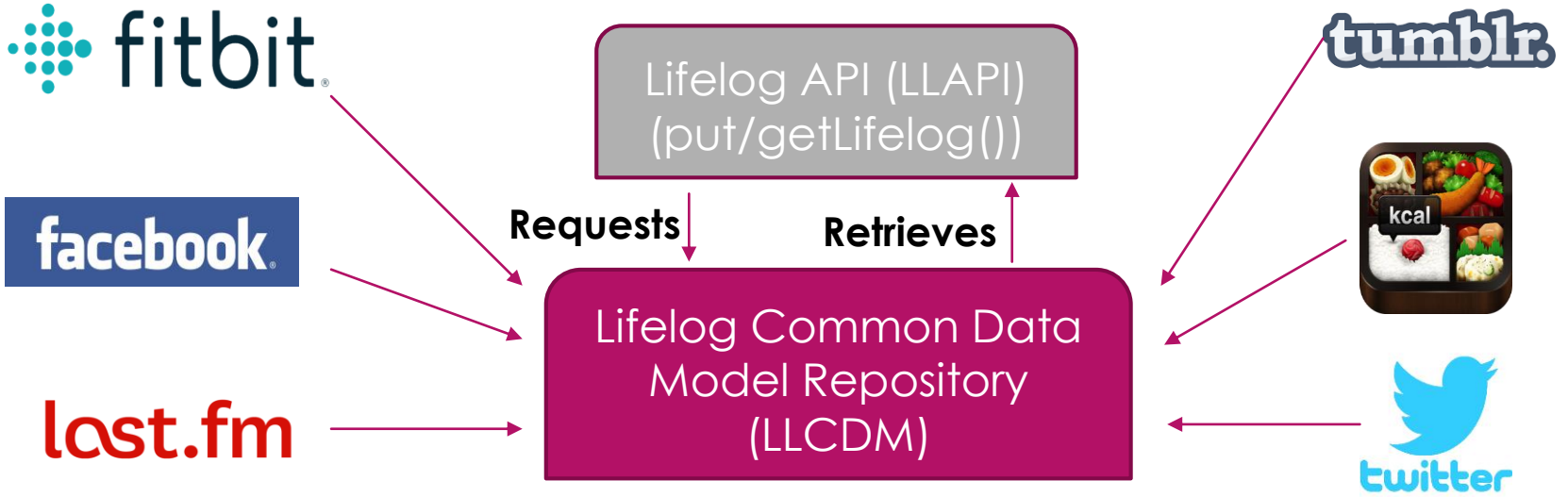
Retrieve data from services API

Lifelog Mashup Platform



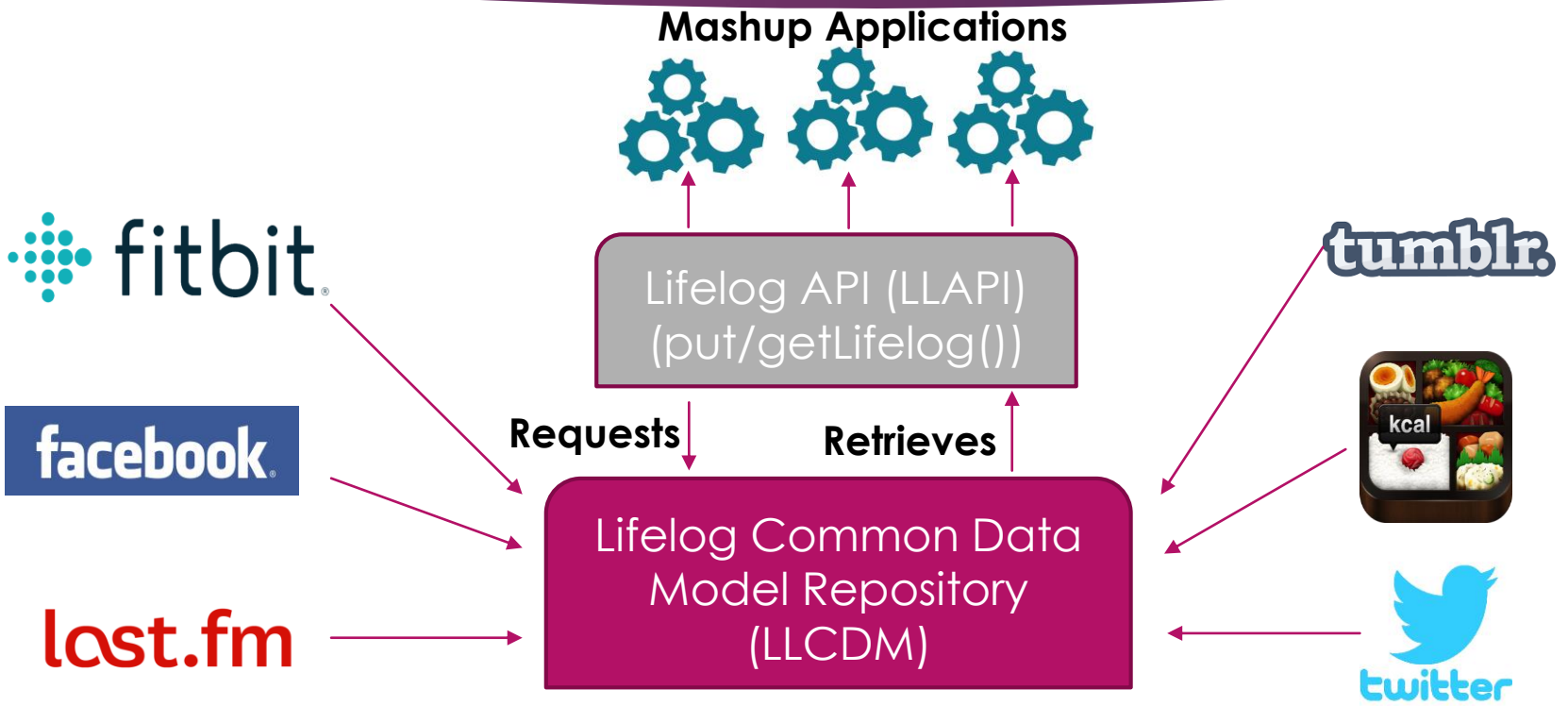
LLAPI requests lifelog data from LLCDM using method

Lifelog Mashup Platform



LLAPI retrieves lifelog data from LLCDM using method

Lifelog Mashup Platform



Mashup applications return user-friendly visuals

Lifelog Mashup Experiments

Experiment 1: First they proposed a lifelog mashup LLCDM and LLAPI to access standardized data

- Poor portability
- Low performance

Lifelog Mashup Experiments

Experiment 1: First they proposed a lifelog mashup LLCDM and LLAPI to access standardized data

- Poor portability
- Low performance

Experiment 2: Then re-engineered it with relational MySQL and Web services.

- Evaluated

Lifelog Mashup Experiments

Experiment 1: First they proposed a lifelog mashup LLCDM and LLAPI to access standardized data

- Poor portability
- Low performance

Experiment 2: Then re-engineered it with relational MySQL and Web services.

- Evaluated

Experiment 3: Once again re-engineered, this time with NoSQL

- Evaluated

Limitations with XML Prototype

- Low performance
  - Had to convert data into raw XML files then store it
- Poor Portability
  - Prototype was written in ***Perl*** language, no choice for developers to use other languages to build mashup applications

How to improve limitations

- Low performance
  - Had to convert data into raw XML files then store it

Put data in *relational database (RDB)* instead of having data as raw XML files.

Faster data search and access.

- Poor Portability
  - Prototype was written in ***Perl*** language, no choice for developers to use other languages to build mashup applications

Programmers create and implement two versions of the mashup application to evaluate the feasibility of new implementation

Outline

1. INTRODUCTION
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
    i. Process
    ii. Evaluation
    iii. Limitations
3. LIFELOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
4. CONCLUSION

Process

1. Importing lifelog data to LLCDM repository
2. Re-engineering LLAPI
3. Evaluate Performance
   - SOAP and REST Web-service Protocols
   - Mashup Example TabetaLog

Goal: To show the practical feasibility of the *proposed* LLAPI.

1. Importing lifelog data to LLCDM Repository

Steps to import data from heterogonous lifelog services to the LLCDM repository:

1. Importing lifelog data to LLCDM Repository

Steps to import data from heterogonous lifelog services to the LLCDM repository:

1. Obtain original data
    1. Obtain the original data from service and store data in XML

1. Importing lifelog data to LLCDM Repository

Steps to import data from heterogonous lifelog services to the LLCDM repository:

1. Obtain original data
    1. Obtain the original data from service and store data in XML
2. Transform data to LLCDM
    1. Raw data → to the LLCDM format

1. Importing lifelog data to LLCDM Repository

Steps to import data from heterogonous lifelog services to the LLCDM repository:

1. Obtain original data
    1. Obtain the original data from service and store data in XML
2. Transform data to LLCDM
    1. Raw data → to the LLCDM format
3. Insert data into database
    1. Insert the XML into the database
    2. Parses the converted XML data
    3. Extracts the attributes and inserts the values to appropriate tables.

Comparison of execution times

	Query 1	Query 2	**Query 3**	Query 4
SOAP (sec)				
REST (sec)				
OLD (sec)				
# OF ITEMS				
DATA SIZE (kB)				

Comparison of execution times

	November 15-November 16	Query 2	**Query 3**	Query 4
SOAP (sec)	0.131			
REST (sec)	0.015			
OLD (sec)	4.238			
# OF ITEMS	36			
DATA SIZE (kB)	118			

Comparison of execution times

	November 15-November 16	September 1-September 30	**Query 3**	Query 4
SOAP (sec)	0.131	1.006		
REST (sec)	0.015	0.100		
OLD (sec)	4.238	4.028		
# OF ITEMS	36	119		
DATA SIZE (kB)	118	381		

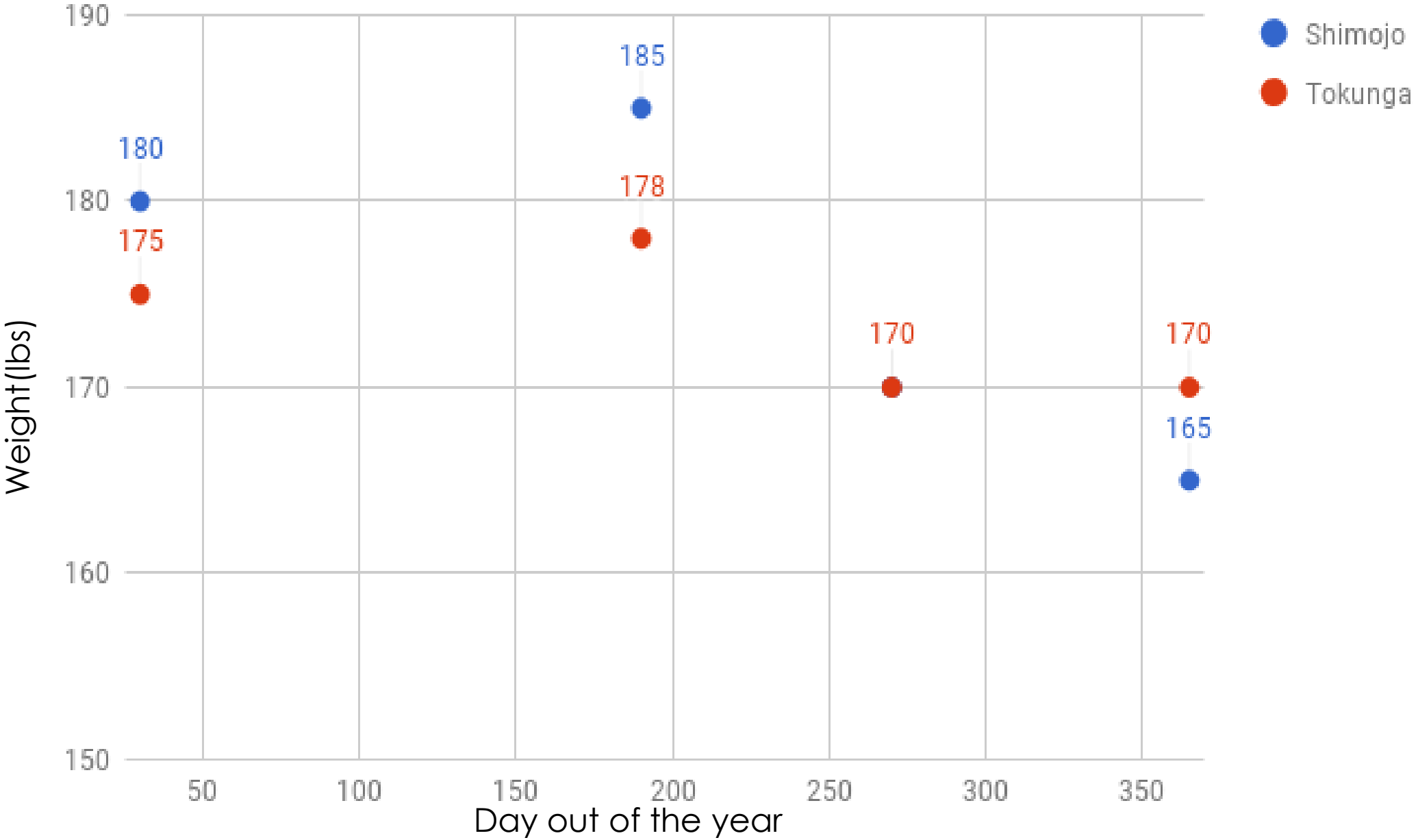
Comparison of execution times

	November 15-November 16	September 1-September 30	**9:00:00-10:15:00 On any date**	**Query 4**
SOAP (sec)	0.131	1.006	0.281	
REST (sec)	0.015	0.100	0.019	
OLD (sec)	4.238	4.028	4.254	
# OF ITEMS	36	119	195	
DATA SIZE (kB)	118	381	1,450	

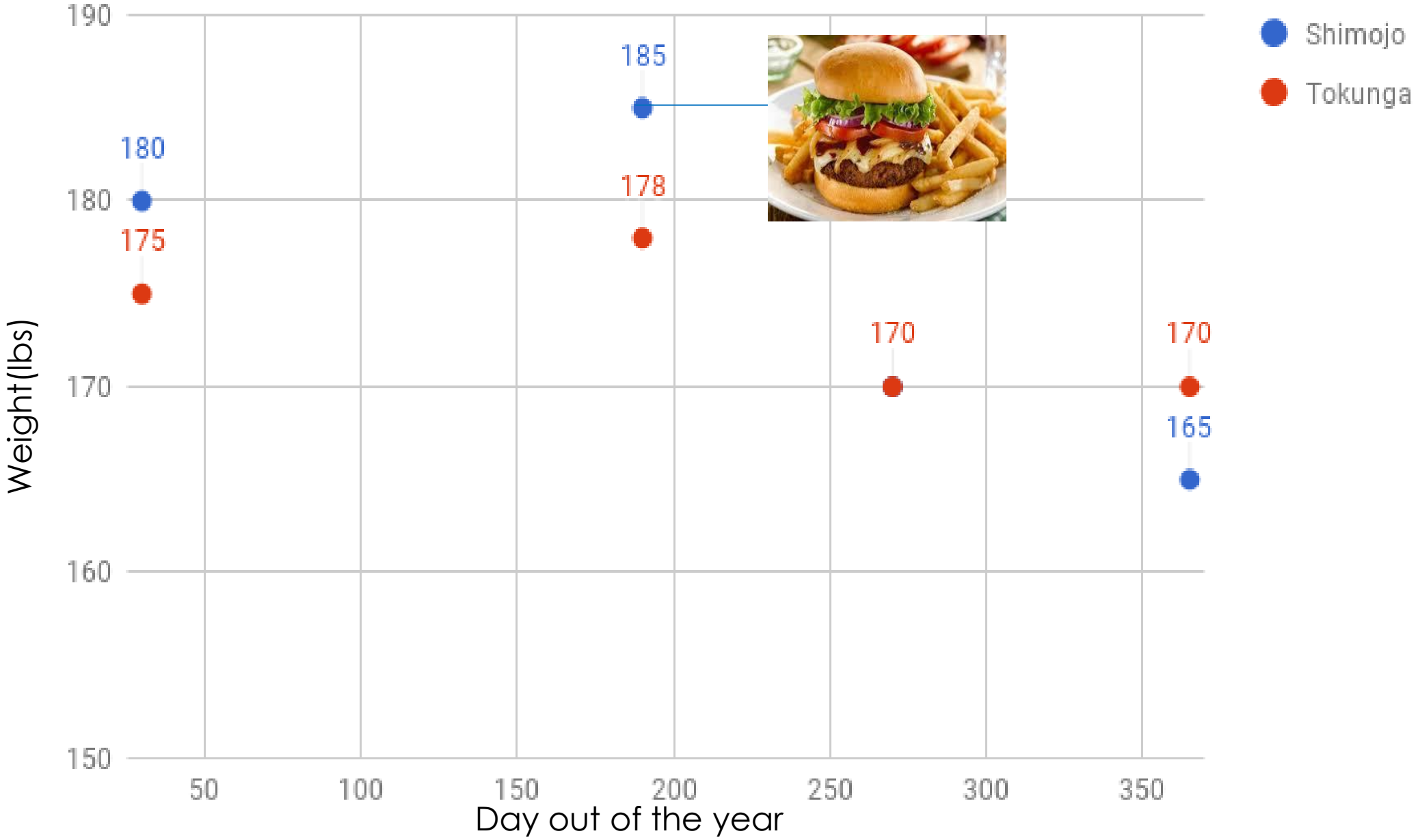
Comparison of execution times

	November 15-November 16	September 1-September 30	**9:00:00-10:15:00 On any date**	**User – "Shimojo"**
SOAP (sec)	0.131	1.006	0.281	0.422
REST (sec)	0.015	0.100	0.019	0.025
OLD (sec)	4.238	4.028	4.254	0.581
# OF ITEMS	36	119	195	449
DATA SIZE (kB)	118	381	1,450	630

TabetaLog – FoodLogService + Flickr



TabetaLog – FoodLogService + Flickr



Process for TabetaLog

TabetaLog was an experimental evaluation lifelog mashup application.

Steps for creating the TabetaLog:

1. Obtain original lifelog records
   - *Web-service API*
2. Extract data items
   - Parsing records
3. Join data items
   - Joined records are stored in JSON format file
4. Create TabetaLog
   - Using ActionScript, visualize the JSON data

Evaluation/Results

Programmer	1	2	3	4	5	Correct
Order of Development	P_{llapi} P_{conv}	P_{conv} P_{llapi}	P_{conv} P_{llapi}	P_{conv} P_{llapi}	P_{llapi} P_{conv}	-
	P_{llapi} P_{conv}	P_{llapi} P_{conv}	P_{llapi} P_{conv}	P_{llapi} P_{conv}	P_{llapi} P_{conv}	P_{llapi} P_{conv}
Programming Language	Perl Perl	Perl Perl	Java Java	Java Java	Java Java	
Source lines of code	115 365	227 379	480 612	423 397	150 181	
SLOC (w.out blank and comments)	71 223	103 188	351 426	286 263	106 125	
# of source-code classes			7 7	5 5	2 2	
# of source-code files	1 4	1 3				
Man-hour (man minute)	114 196	54 205	96 252	147 514	132 397	
# of weight records <Shimojo>	53 54	53 54	32 53	53 54	52 52	53 54
# of weight records <Togunaga>	102 101	102 101	52 103	103 104	103 115	102 101
# of picture records <Shimojo>	8 9	8 9	8 9	8 9	8 9	8 9
# of picture records <Tokunaga>	85 86	85 86	60 87	85 44	65 85	85 86

Evaluation/Results

Programmer	1	2	3	4	5	Correct
Order of Development	P_{llapi} P_{conv} P_{llapi} P_{conv}	P_{conv} P_{llapi} P_{llapi} P_{conv}	P_{conv} P_{llapi} P_{llapi} P_{conv}	P_{conv} P_{llapi} P_{llapi} P_{conv}	P_{llapi} P_{conv} P_{llapi} P_{conv}	P_{llapi} P_{conv}
Source lines of code	115 365	227 379	480 612	423 397	150 181	
Programming Language	Perl Perl	Perl Perl	Java Java	Java Java	Java Java	
Man-hour (man minute)	114 196	54 205	96 252	147 514	132 397	

3. Evaluating Performance

- Compared to previous prototype.
- 1,591 records of data were stored in MySQL database

Five subjects implement a program generating the TabetaLog JSON file. Subjects implement two versions of the program: one with the *proposed* LLAPI and one with the *conventional* LLAPI. [1]

The subjects were instructed to mashup the weight records and the picture records of user "Shimojo" and "Tokunaga" for one year (May 18th, 2010-May 17th, 2011) and to output the resulting JSON file.

Outline

1. INTRODUCTION
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
    i. Limitations
    ii. Process
    iii. Evaluation
4. CONCLUSION

Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]
2. Scalability

Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]

Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]
   - Data was stored in an unstructured plain text file, which SQL **cannot** interpreted.
   - Queries with application-specific attributes had to be managed by individual mashup applications. Causing large application overhead and expensive development cost [2].

Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]
   - Data was stored in an unstructured plain text file, which SQL **cannot** interpreted.
   - Queries with application-specific attributes had to be managed by individual mashup applications. Causing large application overhead and expensive development cost [2].
2. Scalability

Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]
   - Data was stored in an unstructured plain text file, which SQL **cannot** interpreted.
   - Queries with application-specific attributes had to be managed by individual mashup applications. Causing large application overhead and expensive development cost [2].
2. Scalability
   - As more lifelog services appear, the platform should be scalable enough to keep up with larger data.

Benefits of MongoDB

Resolves limitation 1

- Document-orientated storage
  - MongoDB BSON object represents dynamically-typed data in the **<content>** column
- Full index support
  - Useful for queries over the **<content>** column

Resolves limitation 2

- Supports MapReduce
  - Programming model and an associate implementation for processing and generating large datasets of a variety of real-world tasks [2]

Process

- Design LLAPI with MongoDB
  - Implementation
  - Evaluation with SensorLoggingService

Designing LLAPI with MongoDB

Once the lifelog data is stored in the LLCDM, the data is retrieved using a greater queries language MongoDB offers.

Expanding the capability of the previous LLAPI implemented with SQL.

Improved **getLifelog** method is as follows:

getLifeLog([s_date, e_date, s_time, e_time, s_term, e_term, user, party, object, s_alt, e_alt, s_lat, e_lat, s_long, e_long, loc_name, address, ~~location~~, application, device, content, select, limit, order, offset])

Evaluation

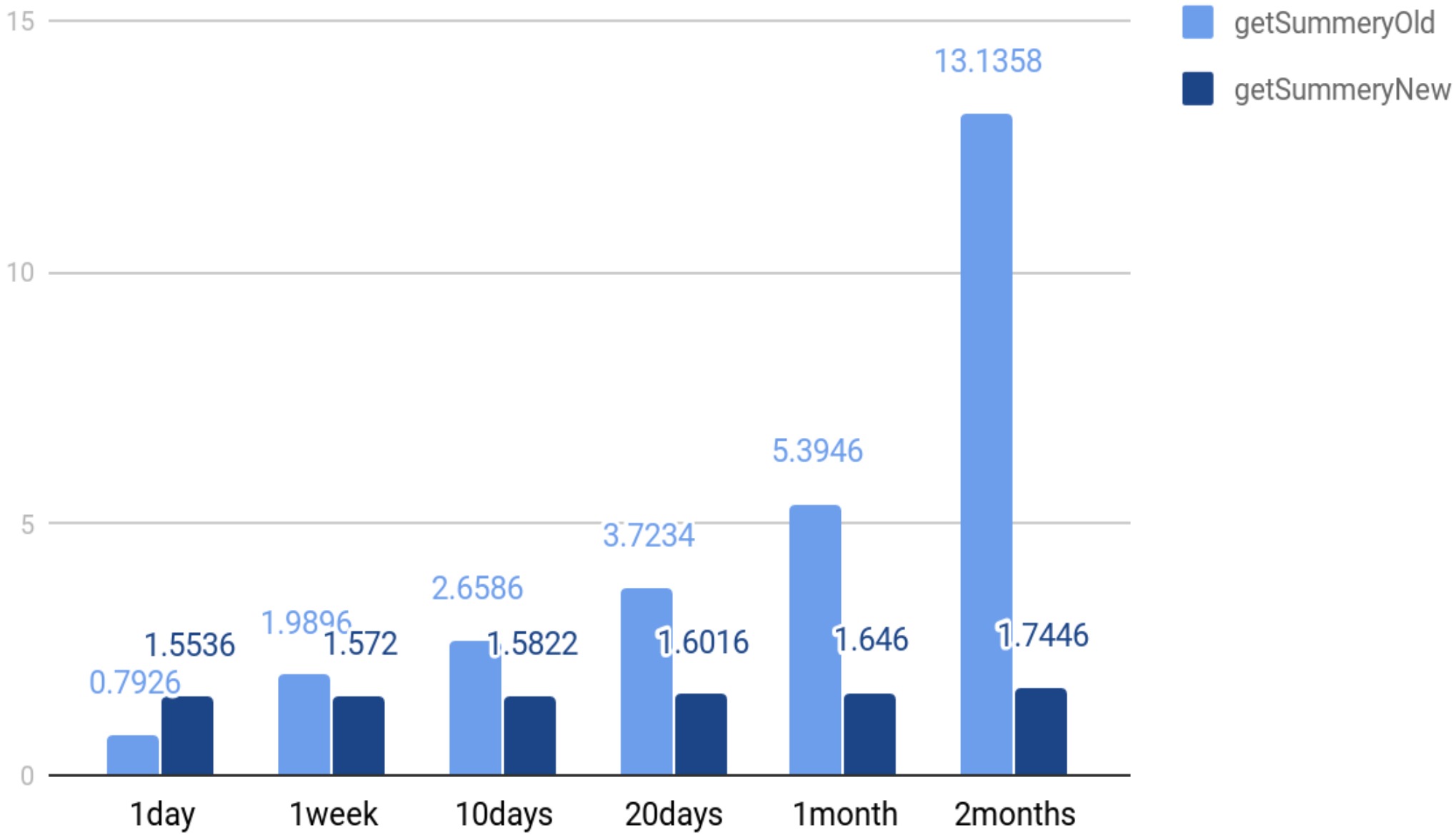
Experiment using environmental sensor log from **SensorLoggingService,** deployed in their *smart home*

This service measures environment inside/outside of their laboratory using various sensors including temperature, humidity, brightness, pressure, motion, and the number of people. The sensor has recorded every minute for three years, a total of **1,664,937.**

Records are then imported to new(MongoDB, NoSQL) and old(RDB, MySQL) platform.

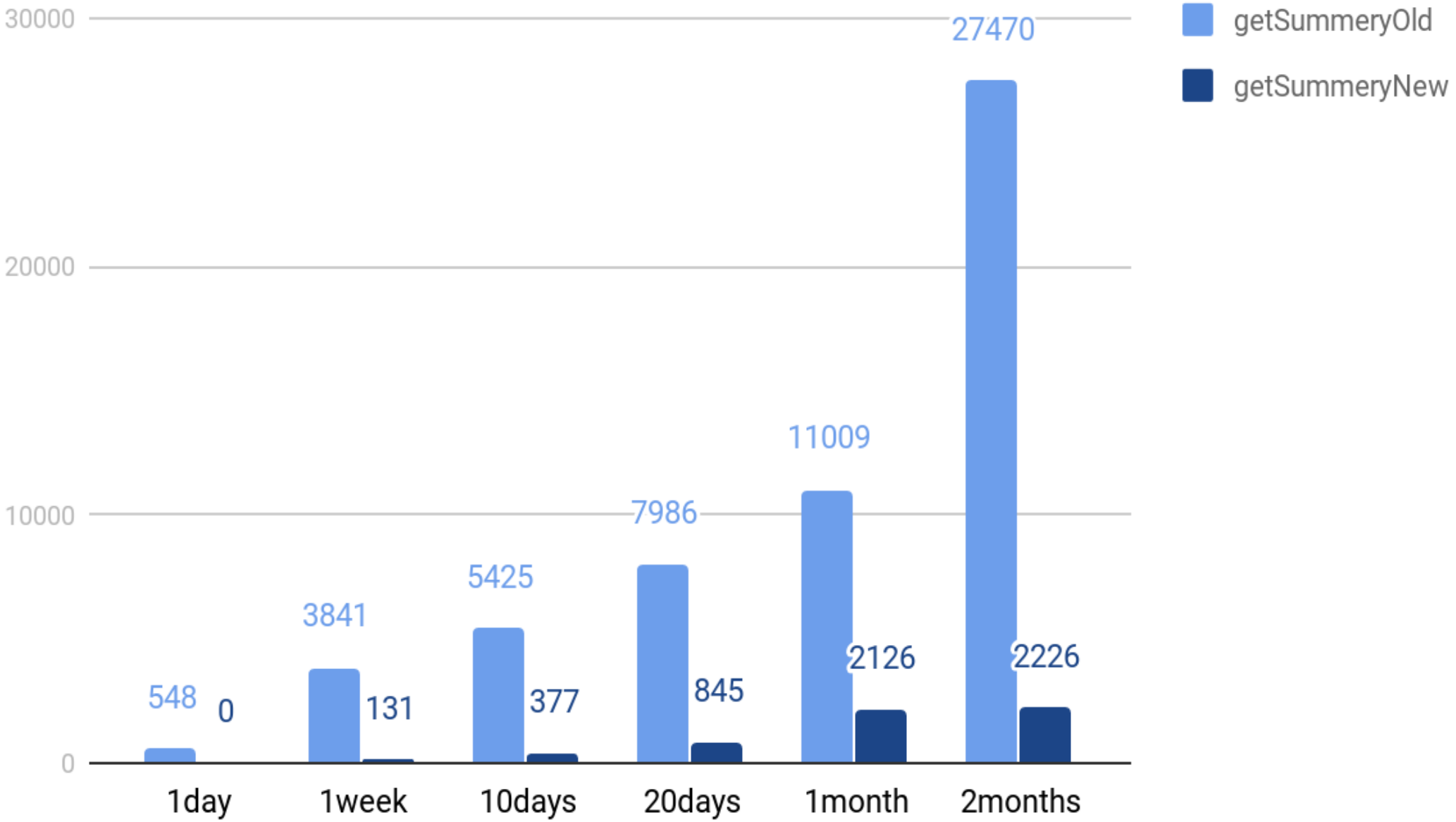
A client application was developed where it picks out *summery days*, which means a day that between 9 AM and 6 PM, the maximum temperature exceeds 25 degrees Celsius.

Graph of time (sec) to retrieve data within given time period



mongoDB

Number of items retrieved within the given time period



mongoDB

Outline

1. INTRODUCTION
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
4. CONCLUSION
    i. Comparisons between SQL and NoSQL

SQL vs. NoSQL

The experimental results showed that the application with the new LLAPI with MongoBD achieves a higher performance and scalability with lower application complexity, compared to the the XML and MySQL implementation.

Thank you for your time and attention

QUESTIONS?

Contact:

E-mail:

Lopez477@morris.umn.edu

LinkedIn URL:

www.linkedin.com/in/luz-m-lopez-her

References

1. Akira Shimojo, Shinsuke Matsumoto, and Masahide Nakamura. 2011. Implementing and evaluating life-log mashup platform using RDB and web services. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*(iiWAS '11). ACM, New York, NY, USA, 503-506. DOI=http://dx.doi.org/10.1145/2095536.2095640
2. Kohei Takahashi, Shinsuke Matsumoto, Sachio Saiki, and Masahide Nakamura. 2013. Design and Evaluation of Lifelog Mashup Platform with NoSQL Database. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services* (IIWAS '13). ACM, New York, NY, USA, , Pages 133 , 7 pages. DOI: http://dx.doi.org/10.1145/2539150.2539229
3. Akira Shimojo, Saori Kamada, Shinsuke Matsumoto, and Masahide Nakamura. 2010. On integrating heterogeneous lifelog services. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services* (iiWAS '10). ACM, New York, NY, USA, 263-272. DOI: https://doi.org/10.1145/1967486.1967529