CS5412 Recitation 1

Yifan Wang, Tancheng Yuan
Contents

• Logistics.
• Computation resources.
• Projects.
• Week 1 Material Revisit.
Logistics
Logistics: Who we are?

• Yifan Wang (yw2399)
  • 3rd-year PhD student, Computer System.
  • Research interest: low-code programming, machine learning engineering, full-stack development.
  • Worked at VMware.
Logistics: Who we are?

• TC Yuan (ty373)
  • 2nd-year MS student
  • Research interests: Federated Learning, MLsys, High Precision Training on Hyperbolic space, Attention Mechanisms, De-centralized Training
Logistics: Who we are?

• Rahul Sharnappa (rb772)
  • MEng student in Computer science.
  • Worked in software industry majorly in Software Defined Networks, Cloud Networking.

• Zack Ashen (za44)
Logistics, cont’d

• What will be covered in the recitations?
  • Course material revisit/discussion.
  • Related technology/research introduction.
  • Projects, hands-on labs, etc.

• We’ll record the sessions but will only address “on-premises” questions.

• Not all the contents are examination-related.
Computation Resources
Resources

Azure Account:

• Once you form groups, you will receive Azure credit for your project

• We will have examples using Azure technologies latter in the course

• https://portal.azure.com

- Other Cloud vendors:

• Aws and other cloud resources may be used. We will not provide funding for them
Resources: How to learn to program your own Cloud Computing programs?

- Azure documentations
Resources: How to learn to program your own Cloud Computing programs?

- Azure documentations
- IoT project, cloud application tutorials
Resources: How to learn to program your own Cloud Computing programs?

- Azure documentations
- IoT project, cloud application tutorials
- Example programs
Projects

• Group Project.
  • 10% -> Intermediate report
  • 10% -> Peer Reviews
  • 30% -> Final Report
  • 30% -> Presentation
  • 20% -> Demo
Projects: Group Formation & Project Idea

• Finding team members:
  • 2-3 students per group. Larger group size means higher expectation.
  • Posting ad on edstem: https://edstem.org/us/courses/26354/discussion/
  • Team formation event, 4PM, Sep. 8, Gates Hall. Details to be disclosed later.
The report should consist of the following sections:

1. Goal / idea (what is the question this project is solving?)
2. Background (what is the current state of research/industry application of it?)
3. Architecture & implementation
4. Evaluation (System performance, accuracy if using ML models)
5. Conclusions

The intermediate and final report could evolve from previous report
Projects: Group Formation & Project Idea

• Group with 2~3 people

• List of group members. (*skills for each member in this project)
  • Name, Net ID

• Two paragraphs about the project idea
  • What is the question the project solves? Which components in Cloud Computing system does this project involve?
  • Briefly plan how you are going to achieve this (input data, analysis, platform, etc.)

• [Tentative] If this is an M.Eng. Project mention who is taking the M.Eng. Credits (4999 if not in M.Eng for the extra work) explain extra piece of work for the M.Eng. Project.
Project idea

• Some Questions
  • Real-world questions: Body Condition Score Prediction; Trajectory Estimation ...
  • What kind of tasks: classification, generation ...
  • Supervised? Semi-supervised? Un-supervised?

• Promising Models
  • Simple and Effective
  • ViT takes days to train on CIFAR-100

• Dataset
Project Idea – IoT application

- IoT: the Internet of Things is the concept of connecting any device to the Internet and to other connected devices.
Project Idea – IoT application

- **Sensing and power**: sensors, cameras, real-time systems on the devices
Project Idea – IoT application
Project Idea – IoT application

• Sensing and power: sensors, cameras, real-time systems on the devices

• Edge computing: distribute computing from cloud data centers to the sources of data
Project Idea – IoT application

Architecture
**Project Idea – IoT application**

- **Sensing and power**: sensors, cameras, real-time systems on the devices
- **Edge computing**: distribute computing from cloud data centers to the sources of data
- **Data communication**: WPANs (Wireless Personal Area Network), LANs (Local area network);
  - how will the data transfer between the edge and the internet?
Project Idea – IoT application

[2]. IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security, 2nd Edition
Project Idea – IoT application
Project Idea – IoT application

- **Sensing and power**: sensors, cameras, real-time systems on the devices
- **Edge computing**: distribute computing from cloud data centers to the sources of data
- **Data communication**: WPANs, LANs; (how will the data get off the edge and on the internet?)
- **Cloud**: infrastructure as a service provider, manage database, streaming and batch processing, support data analytics packages, machine learning services
- **Security**: end-to-end components from physical sensors to the CPU and digital hardware to the communication system needs to ensure security, authenticity, and integrity
Project Idea – IoT application

Cloud Layer

Ingestion Services
Security Services
Storage Services
Compute and Analytics
UI and webservice

Network Layer

Edge Layer

Edge server

[2]. IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security, 2nd Edition, Perry Lea
IoT application  - Challenges and Opportunities

• Local Hardware:
  • Local devices and sensors installation, and connectivity to the cloud.

• Data Collection:
  • Integrate data from multiple difference resources with different formats and granularity

• Data Analysis:
  • Edge and Cloud Coordination: How to distribute the workloads to improve latency issues, response time
  • Apply machine learning tools to analyze the real-time data

• Scalability:
  • Scale up to million/thousands devices

• Energy-efficient with high resource utilization Solutions
For this course, an excellent IoT project need to have edge-cloud interactions:

- It may pull the data from outside source to the cloud in a streaming pattern, and ideally, in a streaming pattern with continuous updates flowing all the time.
Project Idea - Traditional cloud applications

• Real-time data processing and storage systems
• Digital twin applications, such as items tracking
• Cloud based video processing
Project direction 3 - Traditional cloud researchs

- Examples – key/value store, storage system, banking application etc.
- Distributed system guarantee availability and consistency across failures
- Manage application membership (sharding/replication/leaves/joins)
- Performance optimization - high throughput with low latency
- Scalability - Scale with the number of users, computing machines, request rates
- Recommended for students interested in pursuing MS/PhD or a career in Systems
Project Examples and Resources
Image-based authentication for banking

Yu Gu, Gloria Xiao, George Li
An IoT System for Automatic Fall Detection in Smart Homes

Benjamin Posnick, Arjun Bhalla

Based on sensor data, perform feature engineering

- define trajectory
- train machine learning model
- end-to-end pipeline

$$\text{trajectory}(a, b, c) = \begin{cases} 
1 & \text{if } a < b \text{ and } b < c \\
0 & \text{if } a < b \text{ and } b > c, \text{ or } a > b \text{ and } b < c \\
-1 & \text{if } a > b \text{ and } b > c 
\end{cases}$$
Nearby Plane in Flight Information Service
- Web app to allow those on a flight so see nearby planes’ relative locations

Brynn Szczesniak, Freya Ryd, Saleh Hassen

Data input is taken from the OpenSky Network (https://opensky-network.org)
Week 1 Material Revisit
CS5412 Week 1: A Glimpse on Database

- Relational database
  - Data are organized in predefined relations or schemas.
  - Data are stored in tables.
  - MySQL, Postgres, etc.

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>ContactName</th>
<th>Address</th>
<th>City</th>
<th>PostalCode</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alfreds Futterkiste</td>
<td>Maria Anders</td>
<td>Obere Str. 57</td>
<td>Berlin</td>
<td>12209</td>
<td>Germany</td>
</tr>
<tr>
<td>2</td>
<td>Ana Trujillo Emparedados y helados</td>
<td>Ana Trujillo</td>
<td>Avda. de la Constitución 2222</td>
<td>México D.F.</td>
<td>05021</td>
<td>Mexico</td>
</tr>
<tr>
<td>3</td>
<td>Antonio Moreno Taquería</td>
<td>Antonio Moreno</td>
<td>Mataderos 2312</td>
<td>México D.F.</td>
<td>05023</td>
<td>Mexico</td>
</tr>
<tr>
<td>4</td>
<td>Around the Horn</td>
<td>Thomas Hardy</td>
<td>120 Hanover Sq.</td>
<td>London</td>
<td>WA1 1DP</td>
<td>UK</td>
</tr>
<tr>
<td>5</td>
<td>Berglunds snabbköp</td>
<td>Christina Berglund</td>
<td>Berguvsvägen 8</td>
<td>Luleå</td>
<td>S-958 22</td>
<td>Sweden</td>
</tr>
</tbody>
</table>

https://www.w3schools.com/sql/default.asp
CS5412 Week 1: A Glimpse on Database, cont’d

- SQL
  - Structured Query Language.
  - A standard language used to query database(s).
  - Even KVS can support SQL queries.
  - Lots of optimizations.
  - E.g., `SELECT CustomerName, City FROM Customers;`
CS5412 Week 1: A Glimpse on Database, cont’d

• KVS
  • Value can represent the design of schemas.
  • There can be multiple levels of keys.
  • NoSQL = not only SQL
  • E.g., DynamoDB …
CS5412 Week 1: A Glimpse on Database, cont’d

• Some operations are really expensive, e.g., join

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```
• When your operation impacts multiple tables, the join operation is even more expensive. Schema design is very important.

• For SQL, using efficient queries, like using a proper order of join operation, can greatly enhance performance.

• For KVS, traversing multiple tables is also expensive.
CS5412 Week 1: Memcached

• Architecture

https://realpython.com/python-memcache-efficient-caching/
CS5412 Week 1: Memcached, cont’d

• Very easy to use.
(pip install python-memcached)

from pymemcache.client import base
client = base.Client(('localhost', 11211))
client.set('some_key', 'some value')
client.get('some_key')
CS5412 Week 1: Memcached, cont’d

• It can be much more powerful.
  • Distributed cache across multiple hosts.
  • In-memory cache.
There are small “caveats”.

- Size limit per cache line is 1MB. It can be configured to a larger size (memcached -l 5m) or you can: compress and cut your data, build multiple levels of cache, etc.…
- Programmers need to handle data synchronization on their own.
- Fancy features like versioning, time stamping and etc. do not come out of the box.
CS5412 Week 1: Memcached, cont’d

• You might want try something different.
  • Store the data structure, not a string.
  • Being tired of manually checking and synchronizing data.
  • What if I lost a server? Any data backup?
  • Let’s tune our clock from 2003 (Memcached) to 2009
CS5412 Week 1: Redis

- Redis=Remote Dictionary Service
- Features:
  - In-memory data structure storage
  - Address lots of programmers’ concerns as mentioned above.
- Developed by
  - Salvatore Sanfilippo
  - VMware/Pivotal Engineer
  - Originate from real problem in 2009
    - Analyzing web log.
    - Traditional database is too slow.
CS5412 Week 1: Redis, cont’d

• Sometimes I
  • Regard Memcached as a light-weight cache layer.
  • But treat Redis as a fully-fledged database.

• It is an effective tool to address Gray’s concerns on database scaling, as Ken mentioned in class.

• Nearly all major software companies use Redis at some level of their software stack. Who doesn’t?
## CS5412 Week 1: Redis, cont’d

<table>
<thead>
<tr>
<th>Feature</th>
<th>Memcached</th>
<th>Redis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-millisecond latency</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Developer ease of use</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data partitioning</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support for a broad set of programming languages</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Advanced data structures</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Multithreaded architecture</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Snapshots</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Replication</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Transactions</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Pub/Sub</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Lua scripting</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Geospatial support</td>
<td>-</td>
<td>Yes</td>
</tr>
</tbody>
</table>

CS5412 Week 1: Redis, cont’d

• After 13 years
  • Redis is still popular and like an industry standard.
  • We get new challengers.
CS5412 Week 1: Dragonfly

Dragonfly VS Redis

Table showing comparison between Dragonfly and Redis:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Dragonfly</th>
<th>Redis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis API compatible</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Snapshot persistency</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lua</td>
<td>5.4.4</td>
<td>5.1</td>
</tr>
<tr>
<td>QPS per instance</td>
<td>3M</td>
<td>200K</td>
</tr>
<tr>
<td>Async core</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>LRU eviction</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Memcached API compatible</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Native Open Telemetry</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

https://dragonflydb.io/

https://redis.com/blog/redis-architecture-13-years-later/#:~:text=Redis%20vs%20Dragonfly%20%E2%80%93%20Redis%20throughput,we%20were%20able%20to%20reproduce.
CS5412 Week 1: Dragonfly, cont’d

• Open question
  • What is the most popular use case?
  • What is the performance need?