

# CS 561: Data Systems Architectures

Tarikul Islam Papon, Zichen Zhu

[papon@bu.edu](mailto:papon@bu.edu), [zczhu@bu.edu](mailto:zczhu@bu.edu)

<https://bu-disc.github.io/CS561/>

# Tarikul Islam Papon



**PhD candidate @ Boston University**  
**Lecturer @ BUET (Bangladesh University of Engineering & Technology)**

<https://cs-people.bu.edu/papon/>  
Office: **CCDS 925 (TBD)**  
Office hours: **Tu 9-10am**

# Zichen Zhu



**PhD candidate @ Boston University**  
**MPhil @ HKU (the University of Hong Kong)**

<https://cs-people.bu.edu/zczhu/>  
Office: **CCDS 925 (TBD)**  
Office hours: **Th 10-11am**

Today

big data

data-driven world

data systems

*which are the driving trends?*

*why do we need new designs?*

**CS 561 goals & logistics**



I want you to speak up!  
[and you can always interrupt me]

# CS 561 philosophy

cutting-edge research

question everything (to understand it better!)

*There are no stupid questions!*

interactive & collaborative

projects, presentations, labs, OH



# Understanding a design/system/algorithm ...

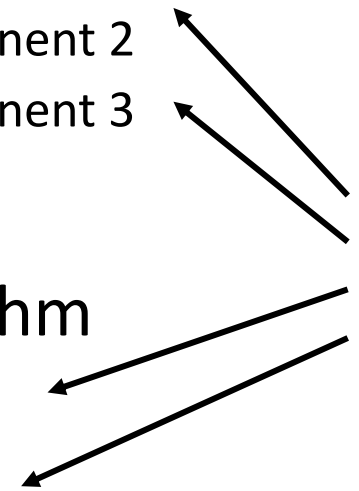
## system

- component 1
- component 2
- component 3

## algorithm

- step 1
- step 2
- step 3

why?  
why not?

A diagram consisting of four arrows pointing from a central point on the right towards the components and steps of the system and algorithm. Two arrows point to 'component 2' and 'component 3' of the system. Two arrows point to 'step 1' and 'step 2' of the algorithm.

understanding all steps and all decisions  
helps us see the ***big picture***  
and do **good research!**

(otherwise, we make ad hoc choices!)



# Ask Questions!

... and answer my questions!

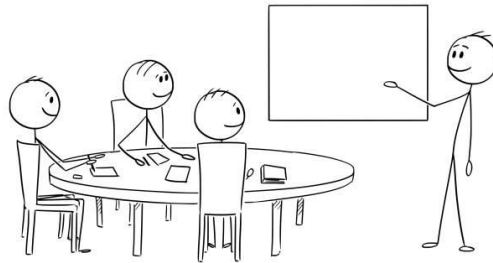
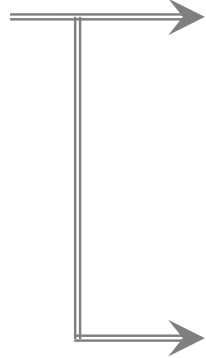
our **main goal** is to have **interesting discussions** that will help to gradually understand what the material discusses

**(it's ok if not everything is clear, as long as you have questions!)**

# What do we do in this class?



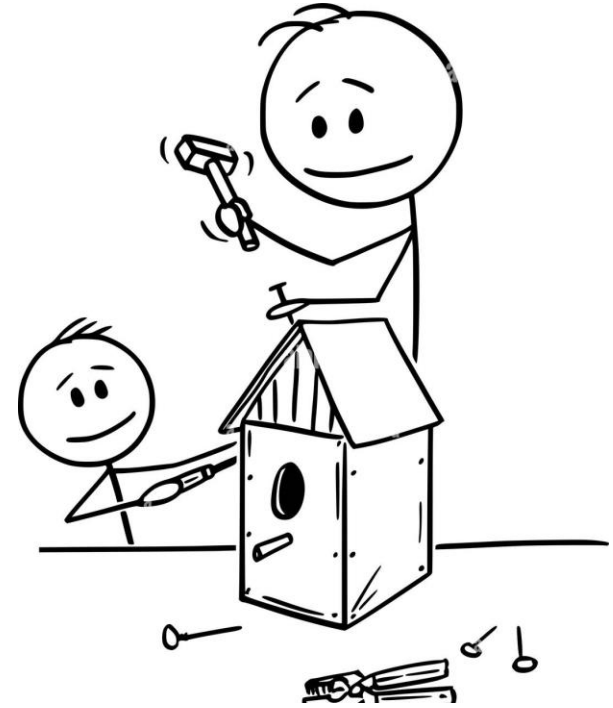
reading papers



presentations



reviews



projects

# Reading Papers



every class **1-2 papers to discuss** in detail

***in some classes the discussion will be led by a group of students***

***so that, each student will present one paper during the semester***

(background papers also available to provide more details)

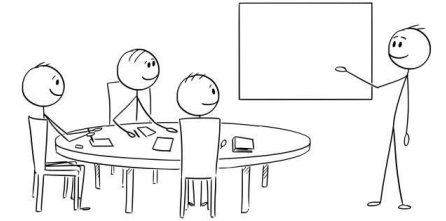
read all of them!

write 4 reviews

answer one technical question per week (for a subset of the papers)



# Presenting Papers



**3-4 students will be responsible for presenting** the paper  
(discussing all main points of a review – see next slide)

during the presentation **anyone can ask questions** (including me!)  
and each question is **addressed to all** (including me!)

prepare slides at least a **week before your presentation**



# Writing Reviews

## 4 reviews and the 5 technical questions

(some weeks will be “free”!)

### **review (up to one page)**

what is the problem & why it is important?

why is it hard & why older approaches are not enough?

what is the key idea and why it works?

what is missing and how can we improve this idea?

does the paper support its claims?

possible next steps of the work presented in the paper?

### **single technical question**

to make sure the heart of the paper is clearly understood

learn

critic

remember, this will help us do **good research!**

# Projects

## project 0

A small implementation project  
to sharpen dev skills

independent project



Due on Feb 2, 2024

**AND**

## project 1

A medium project to give you a flavor of  
large-scale production system

groups of 3



Due on Feb 16, 2024

# Projects

***AND***

## **project 0**

A small implementation project  
to sharpen dev skills

independent project



Due on Feb 2, 2024

## **project 1**

A medium project to give you a flavor of  
large-scale production system

groups of 3



Due on Feb 16, 2024

# Projects

## systems project

groups of 2/3

implementation-heavy C/C++ project

```
01:0 cout<<endl<<endl<<"Iterations #<<"<<" X(1)<<"X(2)<<" X(3):";
02:0 cout<<endl<<" 0<<setw(7)<<"<<setw(15)<<setw(14)<<j3; float temp[3];
03:1 cout<<endl; 110010 10 010001 100111 0 0011011 110101 100 110100 long float j1,j2,j3;
04:1100110101 0101 1000101 100 0001 0 0001 100110000101 10011011 cout<<endl<<"Formatting:"<<endl;
05:000016for(int s=1;s<=20;s++) 10 01 01100111 1100011000 011 1010011010 cout<<endl<<"X(1) =";
06:00110 11 01 01 011 1011 00101111010 01000 01010111001 001 001 cin>>j1;
07:1 0 001 1 temp[0]=1;temp[1]=2;temp[2]=3;01 0 00010 100000101 010111 cout<<endl<<"X(2) =";
08:011011100 j1=(a[3]-a[1])*temp[1]-a[2]*temp[2];a[0]; 110111001100110010101 cin>>j2;
09:1110 0 0 j2=(b[3]-b[0])*temp[0]+b[2]*temp[2];a[1]; 001 01101101011 0001011 cout<<endl<<"X(3) =";
10:11101110 j3=(c[3]-c[0])*temp[0]-c[1]*temp[1];c[2]; 1011 101100 0101 100 cin>>j3;
11:000110100 cout<<"<<setw(7)<<j1<<setw(15)<<setw(14)<<j3<<endl;
12:11001001 if(j1==temp[0]&&j2==temp[1]&&j3==temp[2]) 1000 1000 01 01000001 100000 0
13:11010110 break; 0 110 0101000000 1100 011000 10111 011100 0011 010001 100101010110001
14:100101010 0 0010111 0100 101110 01 10 01110100 0 00101101000100011001 011110110
15:0 0 001011100 101 010 for(int i=0;i<3;i++)
16:0//////////Function Of 000111010001 01001010 110 0000 0001 00101 01011 0000
17:1 transcoding////////// 101111001 0011100 <<endl<<"(X(1)+X(2))^2"; 111010001 0001 001
18:0 void swap(float a[],float b[]) /* function definition */ 101100 cout<<endl;
19:1 float temp[4]; 00001011 0010010 00001 01100 10 0 10111 1101 010001 1001001 11010011110111
20:00 100111010 01 01001110110000 0 0000 cout<<"b(1) ="; 0 101101011001 1001 0000 0111 00
21:0 1101011 0000110111 10111 0100111 10001000101010 cin>>[3]; 01 011110110100 011011100 1001 00
22:0 cout<<"-PROCESSING- "<<endl; 000100100000 10110 cout<<endl;
23:11 cout<<"Preparing Encode X.Y case."<<endl; 01 11 011100 for(int i=0;i<3;i++) 001 001 0001 11 011101 0110
24:1001 cout<<"Procedure starting"<<endl; 10 000 100000101 011 001 1001 01101011
25:1 for(int i=0;i<4;i++) 00010 1010 1100000 cout<<"(2*<j+1<*>); 100001000 0011101
26:1 1 0100101101110 1100111 0100 000110101011 cout<<endl; 1 000 1001 00000101 0000010
27:10 0 j temp[i]=a[i]; 00110 00010 101 100 110 111111000 011011 } 0011101001 101001 1101 100110011
28:10010 a[i]=b[i]; 11001001101110 101100111 100 01 cout<<"(2) ="; 1000001010101011100 1010 10101
29:10000 b[i]=temp[i]; 000001 1001 011 1010101011 0011010100 cin>>[3]; 1 001 001 011101 11000001000000100
30:1 0 01 10 11 1110 10 00 00000111 10001 0101 01 cout<<endl; 01001000 0111 000 01010 01 110000
31:1 101110100 110100 0001000011101011 01 for(int i=0;i<3;i++) 000 00 11 00 01010 10111000
32:0100 cout<<"-Parsing-"<<endl<<endl; 1 10 00110 010 1100 { 0001100 00011001010111 00011000 01001
33:000 cout<<"X.Y transcoded"<<[0]<<"X(1) + "<<[1]<<"X(2) + 01110 cout<<"(3*<k+1<*>"; 0 001110100010 0
34:010<<"(2)<<"X(3) = "<<[3]<<endl 1100 1100100 1011 10101 100 cin>>[k]; 0 0001101 01010 0 01101010
35:1 1<<"UV transcoded"<<[0]<<"X(1) + "<<[1]<<"X(2) 00110101011 cout<<endl; 0100100011 001001101110111
36:<<"(3)<<endl 0 10 11 10101 11110 011010001 00 001110 1 1100 11010101 0 10 1001000 00 11000
37:0<<"Summary"<<[0]<<"X(1) + "<<[1]<<"X(2) 01 01 0 10 cout<<"(3) ="; 0111 0111 01 1010 01101 001 0011100
38:1 1110 00011 001 0 00 00010 100 cin>>[3]; 1 111 001110010 110000101 001 1001110
39:1 cout<<endl<<endl; 001011101001010 0 10 1101; cout<<endl; 0 00011011100100 101 110 111 0
40:11} 000100001101101 0 0 010 1 101 011 0011 11 1010 000010110 1000100 00010011001 0100 011
41:0//////////Function Of 000111010001 01001010 110 0000 0001 00101 01011 0000
```

# OR

## research project

groups of 3

pick a subject (list will be available)

design & analysis

experimentation



# Projects

## systems project

groups of 2/3

implementation-heavy C/C++ project

```
01:0 cout<<endl<<endl<<"Iterations #<< " <<" X(1)<<X(2)<< " X(3);
0:0 cout<<endl<< " 0<<setw(7)<<1<<setw(15)<<2<<setw(14)<<3; float temp[3];
1:00110101 0101 1001001 100111 0 0011011 1101100 1101100 long float j1,j2,j3;
1:00110101 0101 1001001 1001001 0 0011 10011000101 10011011 cout<<endl<<"Formatting"<<endl;
0000for(int s=1;s<=20;s++) 10 01 01100111 1100011000 01110100011010 cout<<endl<<"X(1) =";
00 0110 11 01 0 011 101 0010111010 011000 0 0110111001 001 001 cin>>j1;
1 0 001 1 temp[0]=1;temp[1]=2;temp[2]=3;01 0 00010 1 00000101 010111 cout<<endl<<"X(2) =";
011011100 j1=(a[3]-a[1])*temp[1]-a[2]*temp[2];a[0]; 11011100110011010101 cin>>j2;
01110 0 0 j2=(a[3]-b[0])*temp[0]-b[2]*temp[2];b[1]; 001 01101101011 0001011 cout<<endl<<"X(3) =";
1001110 j3=(c[3]-c[0])*temp[0]-c[1]*temp[1];c[2]; 1011 1011 00 0101 101 cin>>j3;
000110100 cout<< " <<setw(7)<<1<<setw(15)<<2<<setw(14)<<3<<endl;
110 0 001 101 101=temp[0]&8;2==temp[1]&8;3==temp[2]; 100 100 100 100 1000 1000 01 0100000 01 10000 0
11010110 break; 1 1 0 01 0101000000 11000 011000 10111 011100 0011101000 1 100101101100 100011
10010101j1=0;01111 0100 101110 01 11 01110100 0 000101010001001100110110101
0 } 0 001011100 101 010 101 110 00000000 00000 100
110//////////Function Of 000111010001 01001010 110 1000 100 0010101011 0000
011 transcoding////////// 10111100 0011100 0011100 0011100 110100011000 001
02 void swap(float a[],float b[]) /* function definition */ 1011100 cout<<endl;
1:1 float temp[4]; 00001011 00100010 00000 101100 10 0 10111 1101 010000 1001000 1101001101110111
00 100111010 01 011001110110000 0 0000 cout<<"b(1) ="; 0 101101011001 1001 0000 0111 100
00 1110111000 01 10111 10111 0100111 10001000101010 cin>> b[3]; 01 011110110100 011011100 10011 00
1:0 cout<<"PROCESSING"<<endl; 000100100000 1011100 cout<<endl;
011 cout<<"Preparing Encode X.Y cos"<<endl; 01 1 01 100 for(int i=0;i<3;i++) 001 001 1001 11 011101 0110
0 11 cout<<"Procedure starting"<<endl; 10 00 0100001011 001 01 01 01101 01100111
1001 cout<<"2*<sj+1<<*>"; 100001000 0011101
01 for(int i=0;i<4;i++) 000010 1010 1300020 cin>> b[i]; 000 001 1 000 0100101011011 01
1:03 1010011101110 1100111 0100 000110010 0111 cout<<endl; 1 000 1001 000000101 0000010
110 10 temp[i]=a[i]; 00110 0010 101 100 110 01 10111000 01 1011 00111010011101 1100110011
1010 i=i+1; 11 001001101 110 101100111 100 01 cout<<"(2) ="; 1000001010101011100 1010 10101
1000 b[i]=temp[i]; 000001 1001 011 1010101011 00110110100 cin>> b[3]; 1 001 1001 011101 110000001000 001100
1100 i++; 010 10 11 0110 00 000000111101001 0101 01 001 001 01 011 0001010 01 11000 0101 0 110000
0 1 101110100 1101 00 0001000111010101 101 for(int i=0;i<3;i++)000 00 1 100 0 010101 10111000
0100 cout<<"Parsing"<<endl<<endl; 1 10 00110 0010 1 100 { 0001100 00011001010111 000011000 01001
000 cout<<"X.Y transcode"<<[0]<<"X(1) + "<<[1]<<"X(2) + 01110 cout<<"(3*<k+1<<*>"; 0100110100010 0
010<<"[2]<<"X(3) = "<<[3]<<endl 1100 1100100 1011 10101 100 cin>> k; 0001101 01010 0 011010110
1 1<<"UV transcode"<<[0]<<"X(1) + "<<[1]<<"X(2) 00110101011 cout<<endl; 0100100011001001101110111
00 "<<[3]<<endl 0 10 11 10101 11 110 011010001 00 01110 1 1 100 110101010 0 10 1001000 00 11100
20<<"Summary"<<[0]<<"X(1) + "<<[1]<<"X(2) 01 01 0 10 00000000 01 01 0100000000 01 1100 001010 0 110000
10 "<<[3]; 1110 00011 001 0 00 00010 100 cin>> b[3]; 11 11 001 110010 1100010101 001 1001110
01 cout<<endl<<endl; 0010011101001010 0 10 1101 101 0001 1001100100 101 110 111 10
011 000100001101101 00 010 1 101 011 00 11 11 1010 00001101 1000100 000110011001 0100 011
```

# OR

## research project

groups of 3

pick a subject (list will be available)

design & analysis

experimentation



# Research Project: open questions

*skew-aware* join optimization

*context-aware* spatial indexes

exploit *near-sorted data* with concurrency control

quantify *Write Amplification* in modern SSDs

come up with your **own topic!**

*more on the website (soon)*



# A good project

- (1) has a clear plan by project proposal by **end-February** (5%)
- (2) has significant preliminary work done by **end-March** (5%)

evaluation at the **end of the semester** (30%)

- (i) present the key ideas of the implementation/new approach
- (ii) present a set of experiments supporting your claims

come to OH!

(more details for the projects in Class 4)





# Class Goal

understand the internals of  
data systems for data science

tune data systems through **adaptation** and **automation**

get acquainted with research in the area

# Can I take this class?



## background

C++ programming  
data structures  
algorithms  
comp. architecture

## pre-req

CS460/660 & CS210  
contact Papon/Zichen if not sure

## how to be sure?

if familiar with most, then maybe!  
if familiar with **none**, then no!

# Next classes

## **Class 1-2**

logistics, big data, data systems, trends and outlook

## **Class 3**

more basics on data systems, systems classification, graph, cloud

## **Class 4**

intro to class project

## **Class 5 and beyond**

present and **discuss** research papers from Papon/Zichen + students + guest lectures



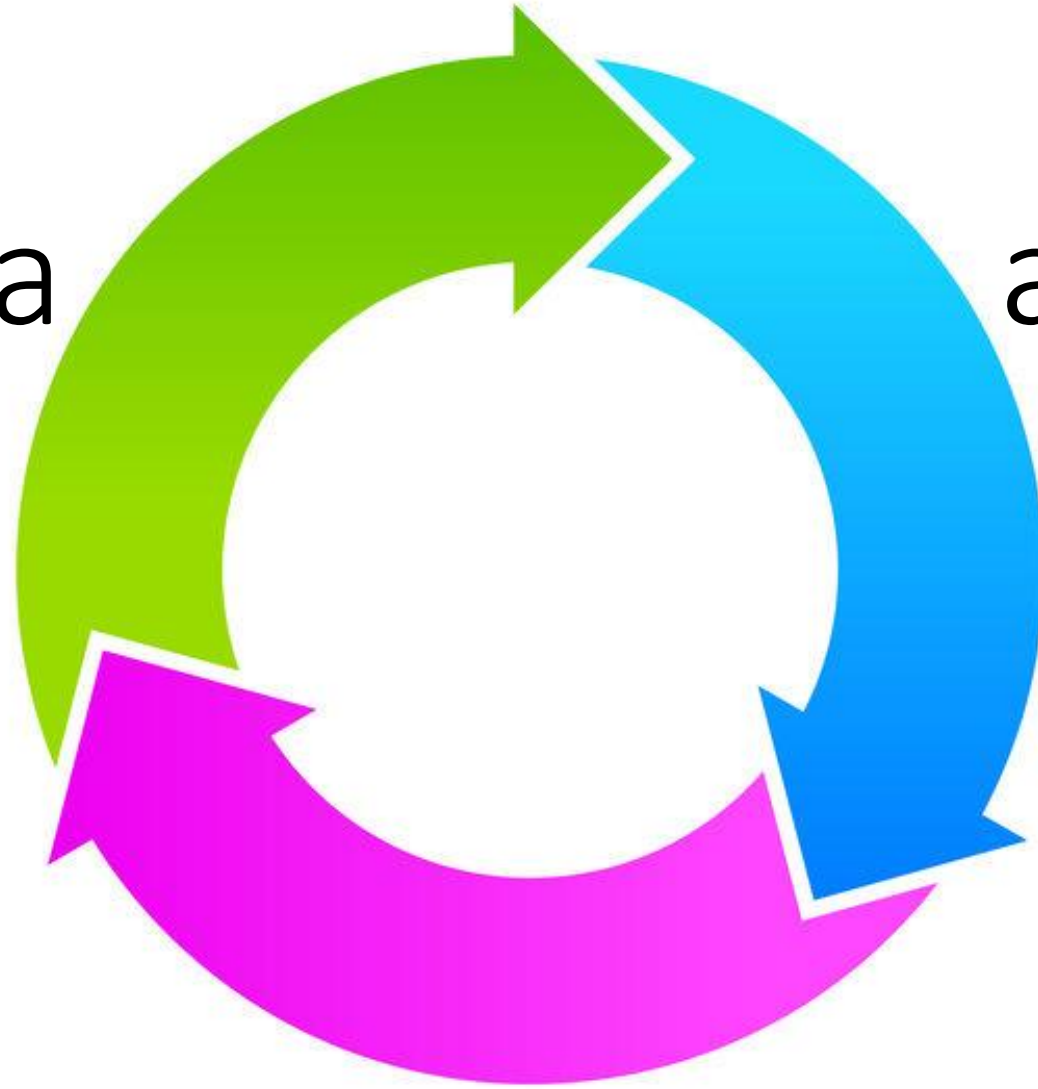
big data?

*who doesn't have a lot of data?*

So what do we do with this data?

data

analysis



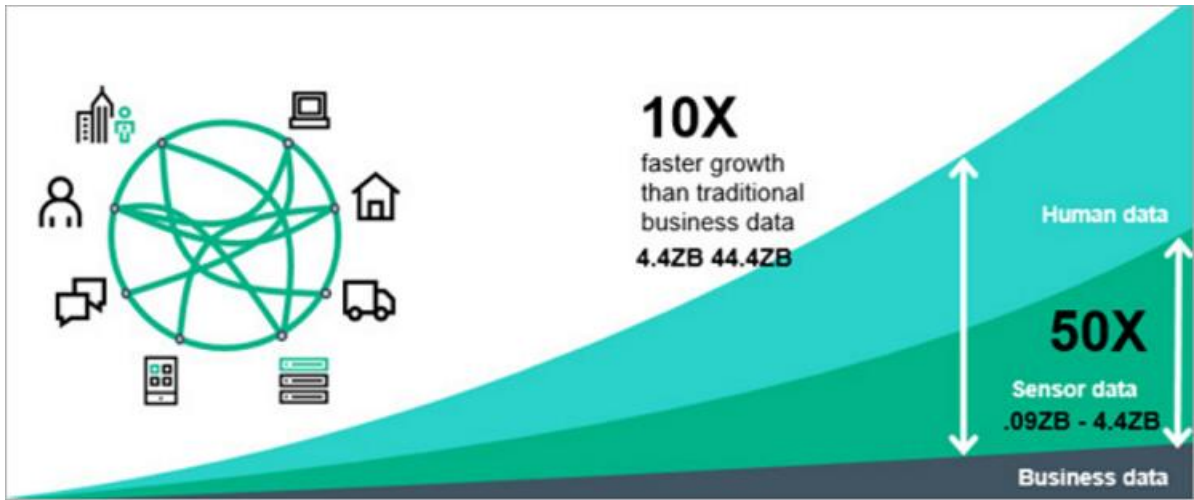
knowledge

is data  
analysis new?



what is  
*really* new?





Every day, we create 2.5 exabytes\* of data — 90% of the data in the world today has been created in the last two years alone.

[Understanding Big Data, IBM]

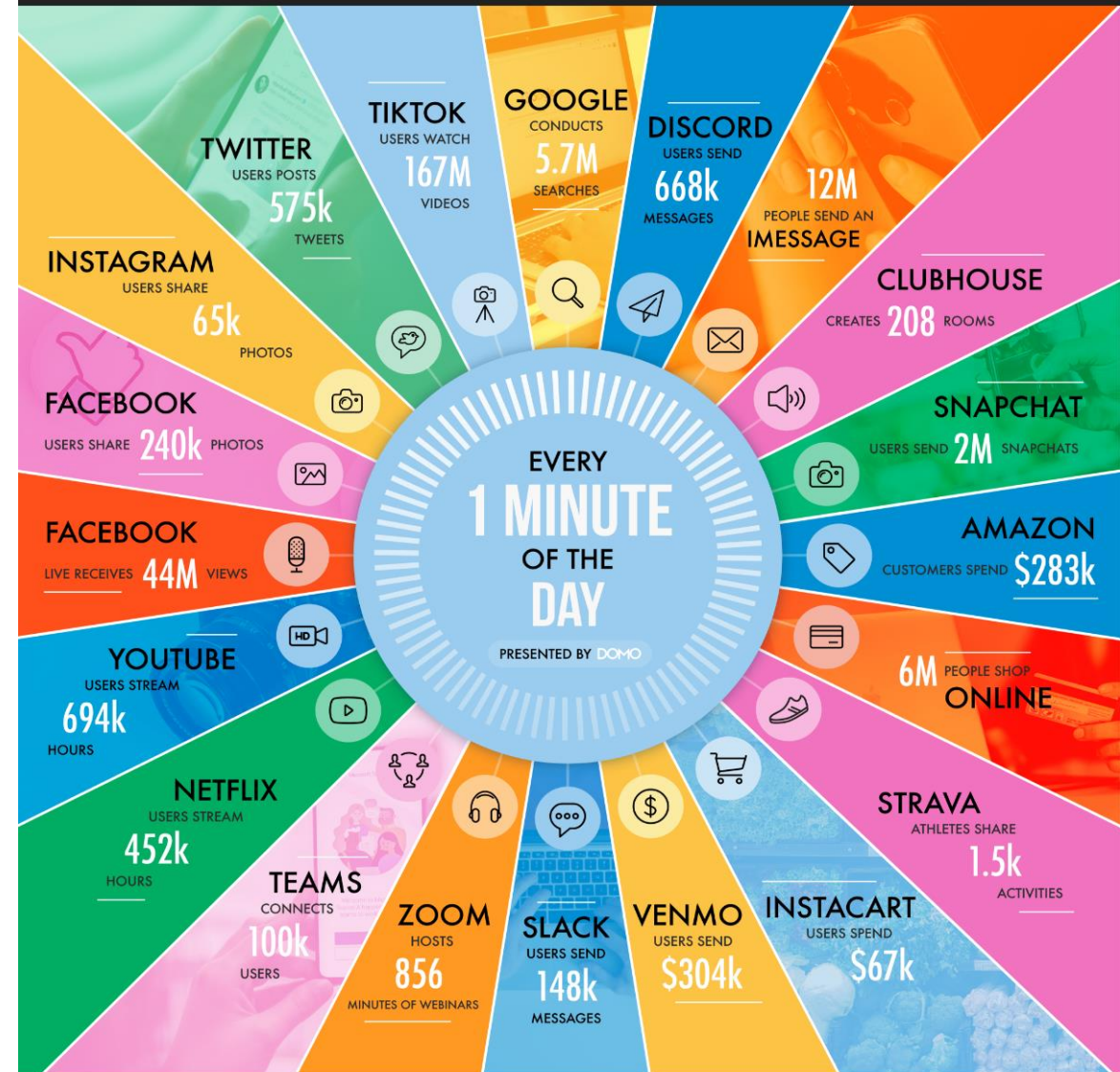
\*exabyte =  $10^9$  GB

DOMO

# Data Never Sleeps 9.0

How much data is generated every minute?

The 2020 pandemic upended everything, from how we engage with each other to how we engage with brands and the digital world. At the same time, it transformed how we eat, how we work and how we entertain ourselves. Data never sleeps and it shows no signs of slowing down. In our 9th edition of the "Data Never Sleeps" infographic, we bring you a glimpse of how much data is created every digital minute in our increasingly data-driven world.



# data management skills needed



100s of entries

**pen & paper**

$10^3$ - $10^6$  of entries

**UNIX tools and excel**

$10^9$  of entries

**custom solutions, programming**

$10^{12+}$  of entries

**data systems**



size (volume)

rate (velocity)

sources (variety)




big data

(it's not only about size)

*all of the above plus ...*


our ability to collect *machine-generated* data

 scientific experiments

 sensors

social 

monitoring 

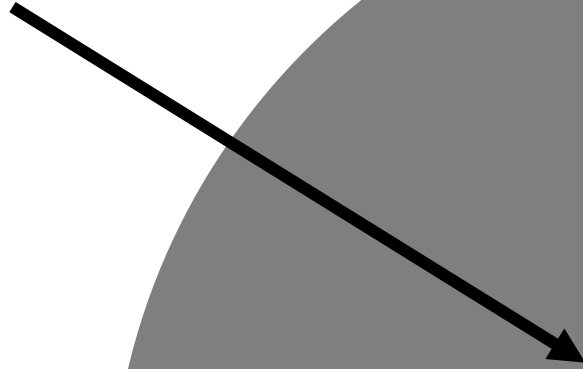
 micro-payments

Internet-of-Things 

cloud 

**big data**

data systems are  
in the middle of this!



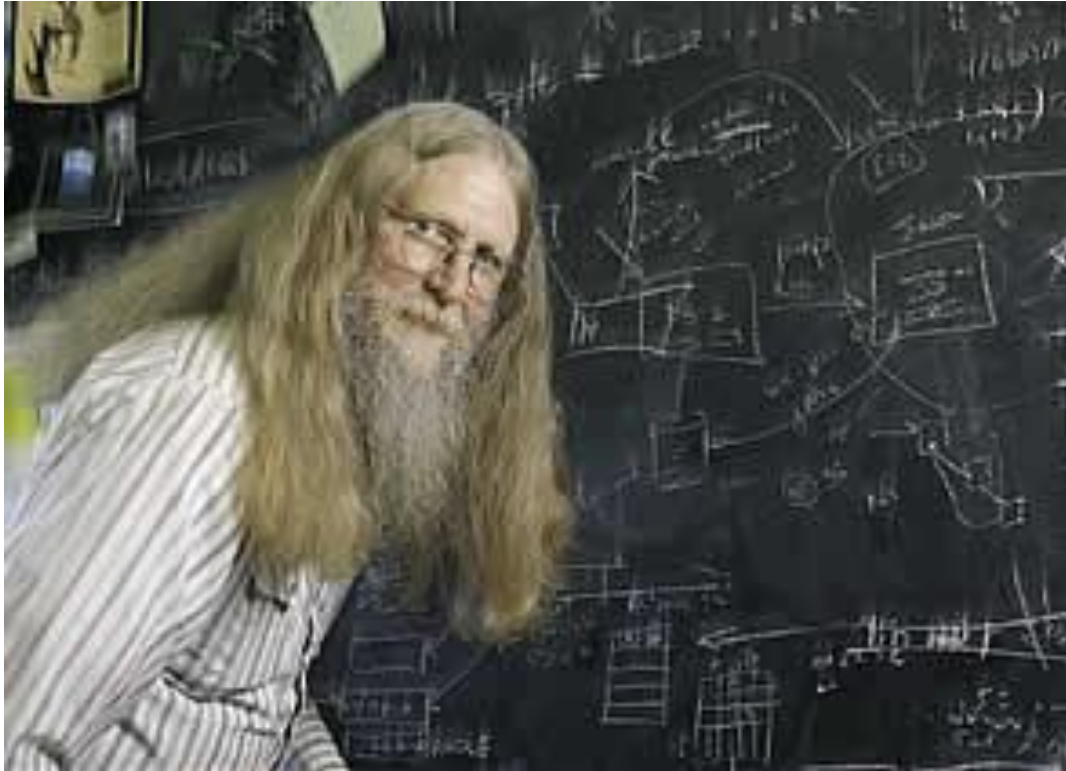
**big data**

**data  
systems**

what is a **data system**?

a **data system** is a large software system  
(a collection of algorithms and data structures)  
that **stores data**, and provides the **interface** to  
**update** and **access** them **efficiently**

the end goal is to make **data analysis** easy



*“relational databases  
are the foundation of  
western civilization”*

Bruce Lindsay, IBM Research

ACM SIGMOD Edgar F. Codd Innovations award 2012

Big  
Tech Era

2019



***+ growing need for tailored systems***



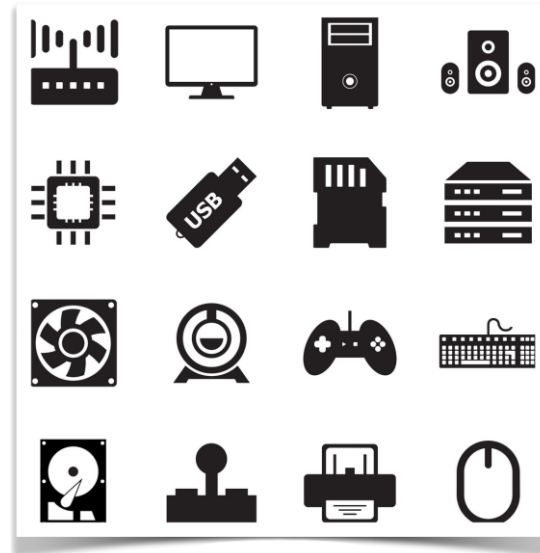
# Why?



more data



new hardware



new applications



new performance goals

ORACLE®

facebook



IBM

Google

# The big success of 5 decades of research

a declarative interface!

“ask and thou shall receive”

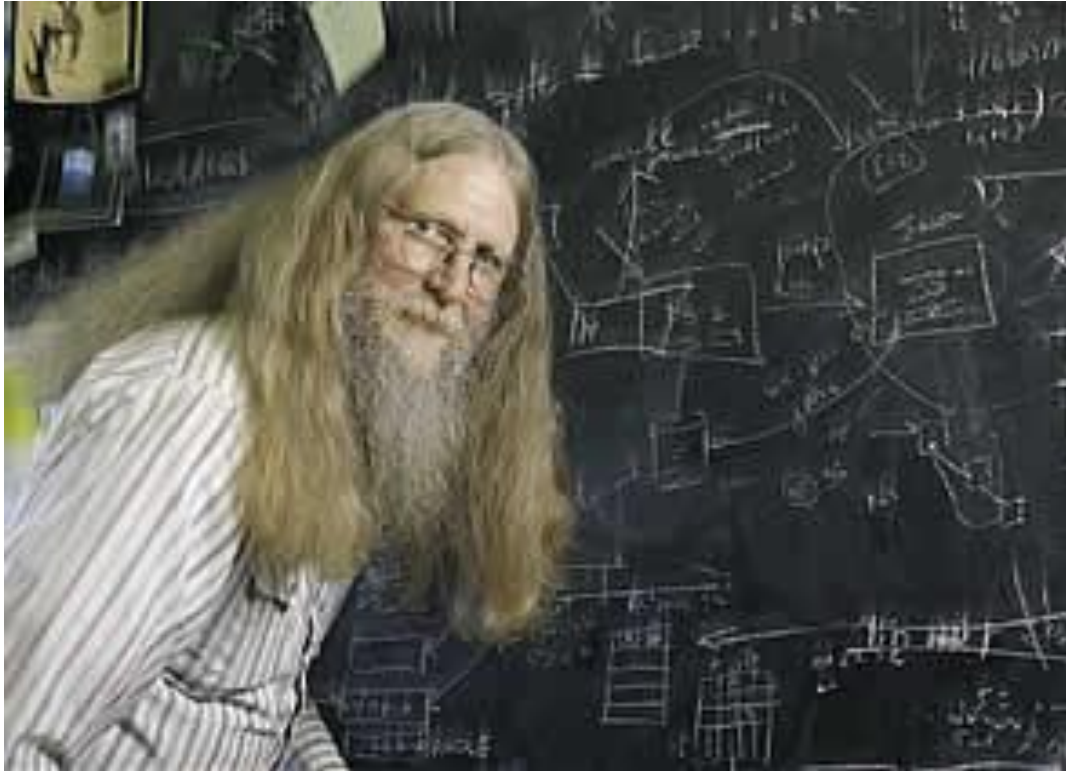
ask *what* you want

data system

system decides *how*  
*to store & access*



is this good?

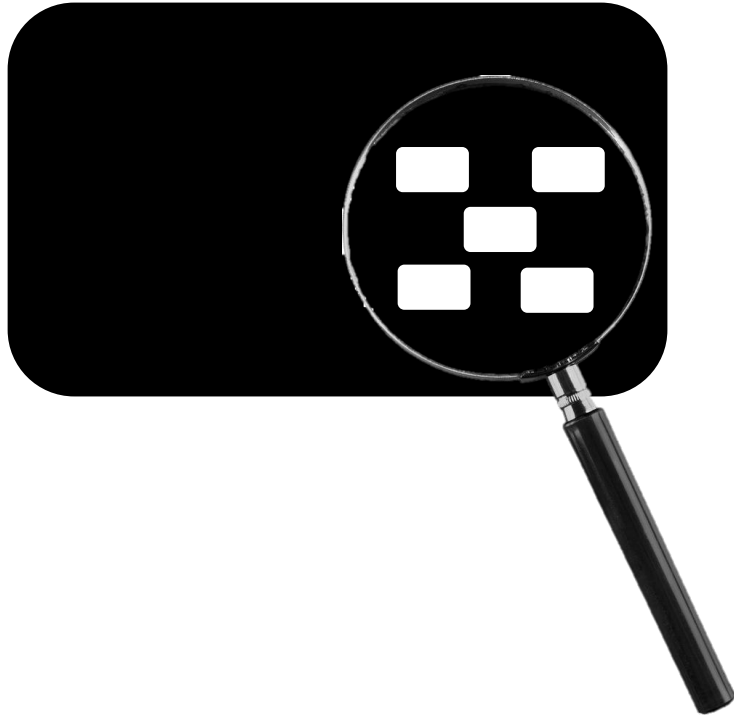


*“three things are important  
in the database world:  
**performance, performance,  
and performance”***

Bruce Lindsay, IBM Research

ACM SIGMOD Edgar F. Codd Innovations award 2012

# CS561: data systems **kernel** under the looking glass



this is where we will spend our time!

system architecture (row/column/hybrid)

indexing

relational/graph/key-value

scale-up/scale-out

***goal: learn to design and implement a DB kernel***

# how to design a data system kernel?

what are its basic components?

algorithms/data structures/caching policies

what decisions should we make?

how to combine? how to optimize for hardware?

*designing a DB kernel is **complex***

# data system design complexity



application



performance



budget

thousands of options  
millions of decisions  
billions of combinations

# let's think together: a simple DB kernel

a key-value system, each entry is a {key,value} pair

**main operations:** *put, get, scan, range scan, count*

workload has both reads (*get, scan, range scan*) *and writes (put)*

data

how to store and how to access data?

how to efficiently delete?



# designing a simple key-value system

what is the key/value?

are they stored together?

can read/write ratio change over time?

what to use? b-tree, hash-table, scans, skip-lists, zonemaps?

how to handle concurrent queries? million concurrent queries?

what happens if data does not fit in memory?

how to compress data?

what about privacy and security?

how to offer robustness guarantees?



# what happens when we move to the cloud?

hardware at massive scale

performance tradeoffs different

10GB app: 1% less memory in your machine

10GB app: 1% less memory in 1M instances



so what?

$1M * 10GB * 1\% = 100TB!$

~800k\$ in today's price

class key goal

understand **system design tradeoffs**

**design** and **prototype** a system

with other **side-effects**:

**sharpening your systems skills**

**(C/C++, profiling, debugging, linux tools)**

data system designer & researcher  
any business, any startup, any scientific domain

# CS 561: more logistics

# topics

storage layouts, HTAP systems, adaptive indexing, solid-state storage, data integration, data skipping, data systems and ML, learned index

# past but still relevant topics

relational systems, row-stores, query optimization, concurrency control, SQL

**no textbook – only research papers**

# grading



class participation: 5%

project 0: 10%

project 1: 15%

reviews: 7%

technical questions: 8%

paper presentation: 15%

project proposal: 5%

mid-semester project report: 5%

project: 30%

# Survival Guide

class website: <https://bu-disc.github.io/CS561/>

## Project 0 [10%]

- Individual
- Due on **Feb 2**

## Project 1 [15%]

- 3 persons per group
- Due on **Feb 16**

## Paper Presentation [15%]

- 3 persons per presentation
- Signup soon here: <http://tinyurl.com/S24-CS561-presentations>

## Reviews [7%]

- Individual
- Each student will do 4 reviews

## Technical Questions [8%]

- Individual
- Each student will do 5 TQs

## Class Participation [5%]

## Class Project [40%]

- 2/3 persons per group
- Project proposal (5%), due on **Feb 23**
- Mid-way Report (5%), due on **Mar 22**
- Final Report + Presentation + Contribution (30%), due on **Apr 26**

# Piazza



all discussions & announcements

<https://piazza.com/bu/spring2024/cs561>

also available on class website

# How can I prepare?

## 1) Read background research material

- **Architecture of a Database System.** By J. Hellerstein, M. Stonebraker and J. Hamilton. Foundations and Trends in Databases, 2007
- **The Design and Implementation of Modern Column-store Database Systems.** By D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden. Foundations and Trends in Databases, 2013

## 2) Start going over the papers



# class summary

2 classes + 2 OH + 1 Lab (5 days) per week

Review + Technical Questions

Paper Presentation

project 0 + project 1 + ***systems or research project***



proposal + mid-semester report + final report + project presentation

# what to do now?

- A) read the syllabus and the website**
- B) register to Piazza + Gradescope**
- C) start working on project 0**
- D) register for the presentation (week 2)
- E) start submitting paper reviews/answering tech. questions (week 3)
- F) go over the class project (end of next week will be available)
- G) start working on the proposal (week 3)

# Resources

**class website:** <https://bu-disc.github.io/CS561/>

**piazza website:** <https://piazza.com/bu/spring2024/cs561>

**presentation registration:** <http://tinyurl.com/S24-CS561-presentations>

**gradescope:** <https://www.gradescope.com/courses/693490> (**code in Piazza**)

**office hours:** Papon (Tu 9-10am)

Zichen (Th 1:30-2:30pm)

**material:** papers available from the BU network

# Welcome to CS 561: Data Systems Architectures!

Tarikul Islam Papon, Zichen Zhu

[papon@bu.edu](mailto:papon@bu.edu), [zczhu@bu.edu](mailto:zczhu@bu.edu)

next time: more detailed logistics and start with data systems design