Cyberinfrastructure Foundations
https://www.cs.rit.edu/~wjz/cisc830/

Weijie Zhao

01/16/2024

Cyberinfrastructure

United States federal research funders use the term cyberinfrastructure to describe research environments that support advanced data acquisition, data storage, data management, data integration, data mining, data visualization and other computing and information processing services distributed over the Internet beyond the scope of a single institution. In scientific usage, cyberinfrastructure is a technological and sociological solution to the problem of efficiently connecting laboratories, data, computers, and people with the goal of enabling derivation of novel scientific theories and knowledge.

--- https://en.wikipedia.org/wiki/Cyberinfrastructure

Course Goals

- Be capable to write your own high-performance computing code for your research on CPUs, GPUs, and multi-node computing clusters
  - Common parallel computing algorithms
  - Communication and synchronization overhead
  - Storage I/O
  - Underlying mechanisms of the framework you use

Course Schedule (Tentative)

- Week 1. Intro to CI and parallel computing frameworks
- Week 2-3. Multicore computing
- Weeks 4-5. GPU, CUDA computing
- Week 6-7. Tensor computing in deep learning
- Weeks 8-10. Cluster computing (no class on week 9: Spring break)
- Weeks 11-12. Parallel computing frameworks, e.g., Spark, and parallel deep learning frameworks
- Week 13-14. Scientific computing, Quantum computing

Grading

- Homework 40%
- Reading 10%
- Project 50%

- 90%<=A<=100%
- 80%<=B<90%
- 70%<=C<80%
- 60%<=D<70%
- 0%<=F<60%

Grading

- Homework 40%
- Reading 10%
- Project 50%

- 90%<=A<=100%
- 80%<=B<90%
- 70%<=C<80%
- 60%<=D<70%
- 0%<=F<60%

- 5 pieces of homework.
- No late submissions.
- No 3rd party code
- Automatically tested: Please strictly follow the output format. An incorrect format is considered as a wrong answer.
- The best 4 scores among the 5 are counted in your final grade.
- The fastest correct solution in each homework gets 10% bonus score in the final grade.
- Other correct solutions that are no slower than 2X of the fastest one gets 5% bonus score in the final grade.

Grading

- Homework 40%
- Reading 10%
- Project 50%

- 90%<=A<=100%
- 80%<=B<90%
- 70%<=C<80%
- 60%<=D<70%
- 0%<=F<60%

- You may discuss general issues with other students, or use other reference materials. Here, the general discussion precludes the detailed techniques or algorithms for solutions. In this case, any help you receive from someone or the references must be acknowledged.
- Note that this does not mean that someone else can do your work. Unless otherwise explicitly specified, all programming projects and written assignments you submit must be done independently. You are not allowed to look at another student's code/solutions or use the code that others have submitted in the past.
- Failure to acknowledge the source of a significant idea or approach will be considered cheating or plagiarism. It results a direct F.

Grading

- Homework 40%
- Reading 10%
- Project 50%

- 90%<=A<=100%
- 80%<=B<90%
- 70%<=C<80%
- 60%<=D<70%
- 0%<=F<60%

- One presentation: a related paper (from a paper pool or a pre-approved paper from a prestigious conference) in 20-30 minutes.

Policy on Large Language Model Usage

Our policy on the usage of Large Language Models (LLMs) is centered around fostering innovation, creativity, and learning. We enthusiastically endorse and promote the incorporation of LLMs in various educational endeavors, including homework, presentations, and projects. We firmly believe that harnessing the capabilities of LLMs aligns with the forefront of technological advancement and offers an invaluable tool for students and professionals alike. By embracing this trend, we aim to empower individuals to explore and utilize the immense potential of LLMs to enhance their academic and professional pursuits. Through responsible and ethical usage, we envision a future where LLMs contribute to the enrichment of knowledge and the cultivation of groundbreaking ideas.

(This paragraph is generated by ChatGPT 3.5.)

Policy on Large Language Model Usage

Our policy on the usage of Large Language Models (LLMs) is centered around fostering innovation, creativity, and learning. We enthusiastically endorse and promote the incorporation of LLMs in various educational endeavors, including homework, presentations, and projects. We firmly believe that harnessing the capabilities of LLMs aligns with the forefront of technological advancement and offers an invaluable tool for students and professionals alike. By embracing this trend, we aim to empower individuals to explore and utilize the immense potential of LLMs to enhance their academic and professional pursuits. Through responsible and ethical usage, we envision a future where LLMs contribute to the enrichment of knowledge and the cultivation of groundbreaking ideas.

(This paragraph is generated by ChatGPT 3.5.)

TL;DR: No limit

Reading List

- https://dblp.org/db/conf/hpdc/hpdc2023.html
- https://dblp.org/db/conf/ics/ics2023.html
- https://dblp.org/db/conf/sc/sc2023.html
- https://proceedings.mlsys.org/paper_files/paper/2023
- 1) High performance computing: super computing, grid computing
- 2) Cyberinfrastructure and applications: biology, physics, astronomy, space weather, engineering
- 3) Data storage, management, visualization, mining
- 4) Cloud computing: virtualization, architecture, e.g., Hadoop/MapReduce, K8s, Docker, Mesos
- 5) Data analytics tools, e.g., Spark, Cassandra, Kafka, MongoDB

Things to consider for the presentation

- What is the objective?
- What problem do we run into?
- What is the idea to address the problem?
- Details of the solution idea?
- Any data available to show the solution works?
- Any drawbacks?
- References

Reading Grading

- Was it easy to follow? (3 pts)
- Did it include in-depth analysis? (3 pts)
- Did it effectively use all given time? (2pts)
- Was the Q&A session useful? (2 pts)

Grading

- Homework 40%
- Reading 10%
- Project 50%

- 90%<=A<=100%
- 80%<=B<90%
- 70%<=C<80%
- 60%<=D<70%
- 0%<=F<60%

- 1-2 students form a group.
- Complete a related project (from a project pool or a pre-approved project related to the class or your own research) and present the project in 20-30 minutes.
- Everyone in the group should be familiar to the entire project. The one who presents the project and answers questions is randomly chosen at the present time.

Cyberinfrastructure

United States federal research funders use the term cyberinfrastructure to describe research environments that support advanced data acquisition, data storage, data management, data integration, data mining, data visualization and other computing and information processing services distributed over the Internet beyond the scope of a single institution. In scientific usage, cyberinfrastructure is a technological and sociological solution to the problem of efficiently connecting laboratories, data, computers, and people with the goal of enabling derivation of novel scientific theories and knowledge.

--- https://en.wikipedia.org/wiki/Cyberinfrastructure

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed

A Toy Problem: Particle Sorting

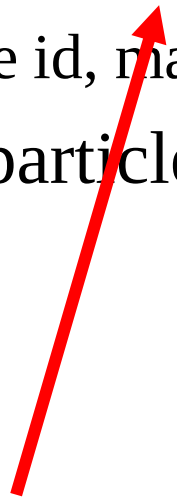
- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed

It’s a naïve problem!

Just sort it!

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



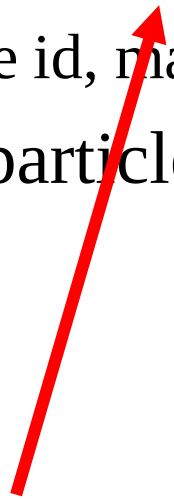
It's a ~~naïve~~ problem!

Just sort it!

What if it does not fit in the memory?
Say, we have TB-scale observations.

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



It's a ~~naïve~~ problem!

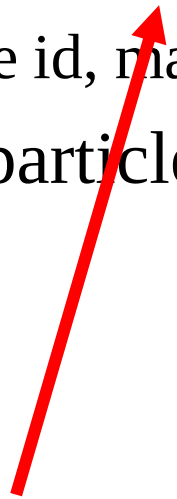
Just sort it!

What if it does not fit in the memory?
Say, we have TB-scale observations.

How should we store them?

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



It's a ~~naïve~~ problem!

Just sort it!

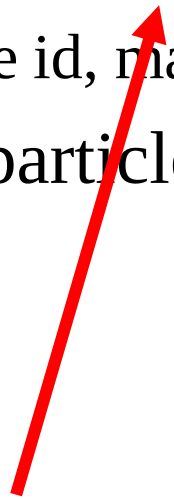
What if it does not fit in the memory?
Say, we have TB-scale observations.

How should we store them?

- Network file system, e.g., HDFS
- Column-wise store
- Auxiliary column (materialized view)

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



It's a ~~naïve~~ problem!

Just sort it!

Do NOT touch my data!

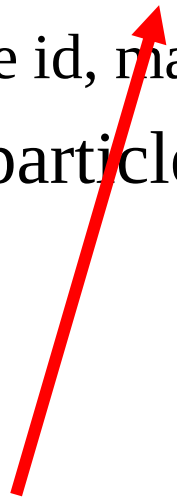
What if it does not fit in the memory?
Say, we have TB-scale observations.

How should we store them?

- ~~Network file system, e.g., HDFS~~
- ~~Column-wise store~~
- ~~Auxiliary column (materialized view)~~

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



It's a ~~naïve~~ problem!

Just sort it!

Do NOT touch my data!

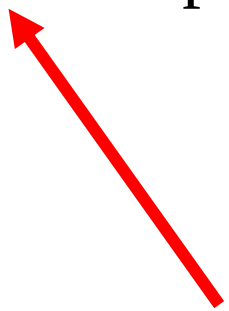
What if it does not fit in the memory?
Say, we have TB-scale observations.

How should we store them?

- In-situ data processing
- Scientific data processing

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



It's a ~~naïve~~ problem!

~~Just~~ sort it!

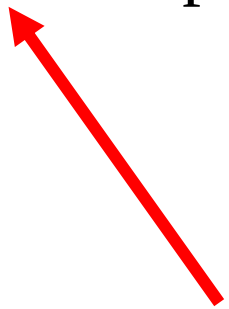
We have so many things to sort.

How to sort data without sufficient memory?

How to make this faster?

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



It's a ~~naïve~~ problem!

~~Just~~ sort it!

We have so many things to sort.

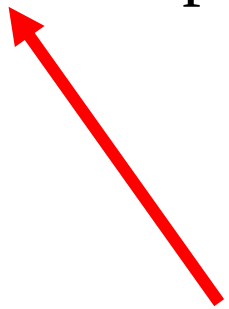
How to sort data without sufficient memory?

How to make this faster?

- Parallel computing
- GPUs
- Distributed computing

A Toy Problem: Particle Sorting

- Given a collection of particle observation:
  - Particle id, mass, velocity (3d), position (3d)
- Sort the particle observations according to their speed



It's a ~~naïve~~ problem!

~~Just~~ sort it!

We have so many things to sort.

How to sort data ~~without sufficient memory~~?

How to make this faster?

- Parallel computing
- GPUs
- Distributed computing

Parallel Sort

- Rank sort
- Odd-even sort
- Odd-even merge sort
- Bitonic sort

Rank Sort

- For each element
  - Compute its rank
  - Put it to its rank position

Odd-Even Sort

- For each odd position, compare its next element
- For each even position, compare its next element
- Repeat until sorted

Odd-Even Merge Sort

- Odd-even merge
- 0/1 principle

Odd-Even Merge Sort

```
# note: the input sequence is indexed from 0 to (n-1)
for p = 1, 2, 4, 8, ... # as long as p < n
  for k = p, p/2, p/4, p/8, ... # as long as k >= 1
    for j = mod(k,p) to (n-1-k) with a step size of 2k
      for i = 0 to min(k-1, n-j-k-1) with a step size of 1
        if floor((i+j) / (p*2)) == floor((i+j+k) / (p*2))
          compare and sort elements (i+j) and (i+j+k)
```

Bitonic Sort

- Bitonic sequence: monotonically non-decreasing then monotonically non-increasing, or a circular shift
- Bitonic split

Bitonic Sort

```
// given an array arr of length n, this code sorts it in place
// all indices run from 0 to n-1
for (k = 2; k <= n; k *= 2) // k is doubled every iteration
    for (j = k/2; j > 0; j /= 2) // j is halved at every iteration, with truncation of fractional parts
        for (i = 0; i < n; i++)
            l = bitwiseXOR (i, j); // in C-like languages this is "i ^ j"
            if (l > i)
                if (  (bitwiseAND (i, k) == 0) AND (arr[i] > arr[l])
                   OR (bitwiseAND (i, k) != 0) AND (arr[i] < arr[l]) )
                      swap the elements arr[i] and arr[l]
```

https://en.wikipedia.org/wiki/Bitonic_sorter