A Comparison of Sorting Methods for Sorting Data on Lower End Hardware and Software

Dan Cojocaru
dan.cojocaru00@e-uvt.ro

Introduction

- **Sorting algorithms are widely discussed**
- **The methods to put them in practice aren't always as discussed**
- **USA:**
  - FCC Report: 10% of Americans lack access to 25 Mbps / 3 Mbps internet service[1]
- **Global:**
  - Akamai's "State of the Internet": 7.2 Mbps average global internet speed[2]
- **How to optimise sorting to take advantage of the potential delay in communication imposed by internet speed?**

Online Sorting Algorithm?

A promising idea, a disappointing result.

Online Algorithm

An online sorting algorithm is one that can process its input piece by piece.

Insertion Sort!

- Every iteration, insertion sort takes one element from the input, finds the place it belongs to in the output and inserts it there.
- That’s an online algorithm!
- If insertion sort can start right away, it should perform faster, right?

Wrong.

- Test results show that the advantage is nearly non existant for small inputs and performance didn’t improve at all for large inputs.
- What now?

Chunking

What if we chunk the input into bits that we process individually?

Chunking

- Several ways to do it with different results
- Probably needs to be tweaked according to the situation in which it is used
- Allows the usage of arbitrary algorithms
- Provides significant improvements compared to waiting for all the data to arrive before starting to process it

Examples of chunking

- Constant chunking
  - Once a constant number of unsorted elements are available, treat them as one chunk and sort them
- Exponential chunking
  - Choosing a number n, the target for the next unsorted elements chunk size is n * the current target
- Equal chunking
  - When the number of unsorted items equals the number of sorted items, take the unsorted items as one chunk and sort them

Performance? TBD.

- Certain:
  - Faster than waiting for data to arrive fully before starting
- Uncertain:
  - Which of the three (or more) approaches are faster
  - Which constants give better performances in which situations
  - Can unoptimal chunking methods lower the performance compared to the regular approach?
  - Is performance severely affected when the transfer speeds are high?

Conclusions

Also known as: “Pfew, this presentation is finally over...”

Conclusions

- **Think about constraints that aren't obvious at first**
  - *"Not everybody has fast internet"*
- **Dividing the input into smaller chunks and sorting those smaller chunks can lead to significant performance improvements when all data is not available instantly**
  - Dividing a problem into smaller bits is generally a good idea in computer science in general
  - *"Divide et impera"*
- **If anybody is interested, further research into this chunking method for sorting is gladly welcome!**

Questions?

The slides, the notes and (soon) the source code will be available at:

https://go.dcdevelop.ml/mpi-presentation-1

Dan Cojocaru

dan.cojocaru00@e-uvt.ro