How to derive ring all-reduce's mathematical property step by step

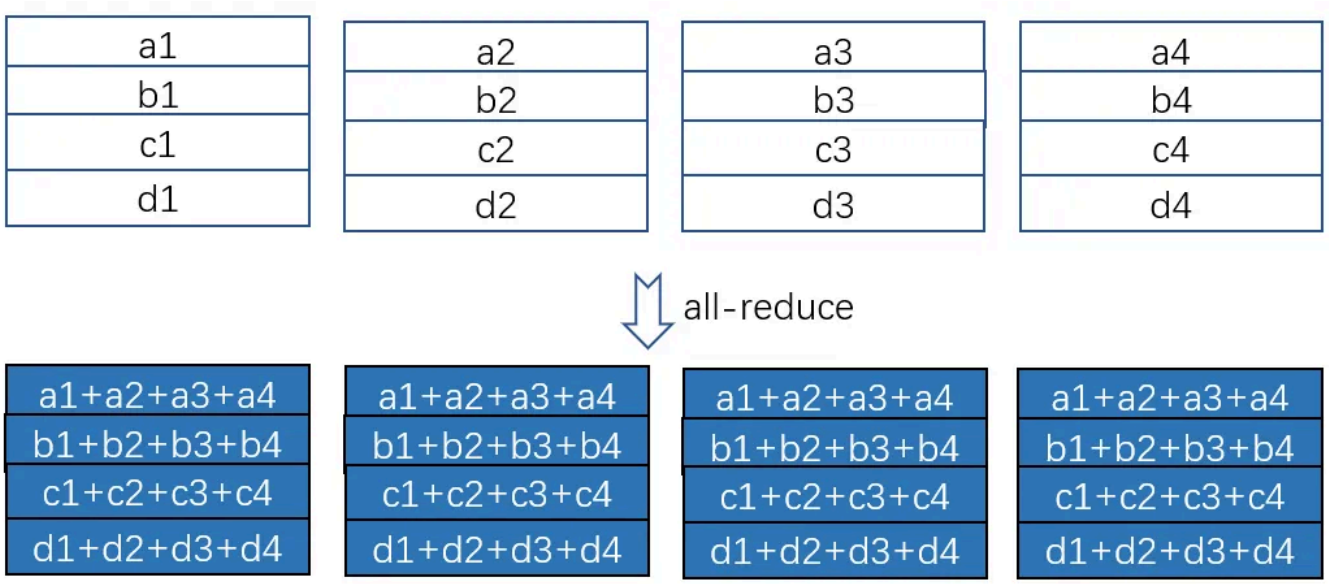
OneFlow · Follow
7 min read · Jun 16, 2022

Written by Jinhui Yuan; Translated by Jiali Shen, Yushan Zhang

> *In our previous blog: Combating Software System Complexity: Appropriate Abstraction Layer, we mentioned that the communication in a distributed deep learning framework is highly dependent on regular collective communication operations like all-reduce, reduce-scatter, all-gather, and so on. Therefore, it's crucial to implement a highly optimized collective communication and select an ideal algorithm based on task requirements and communication typology.*

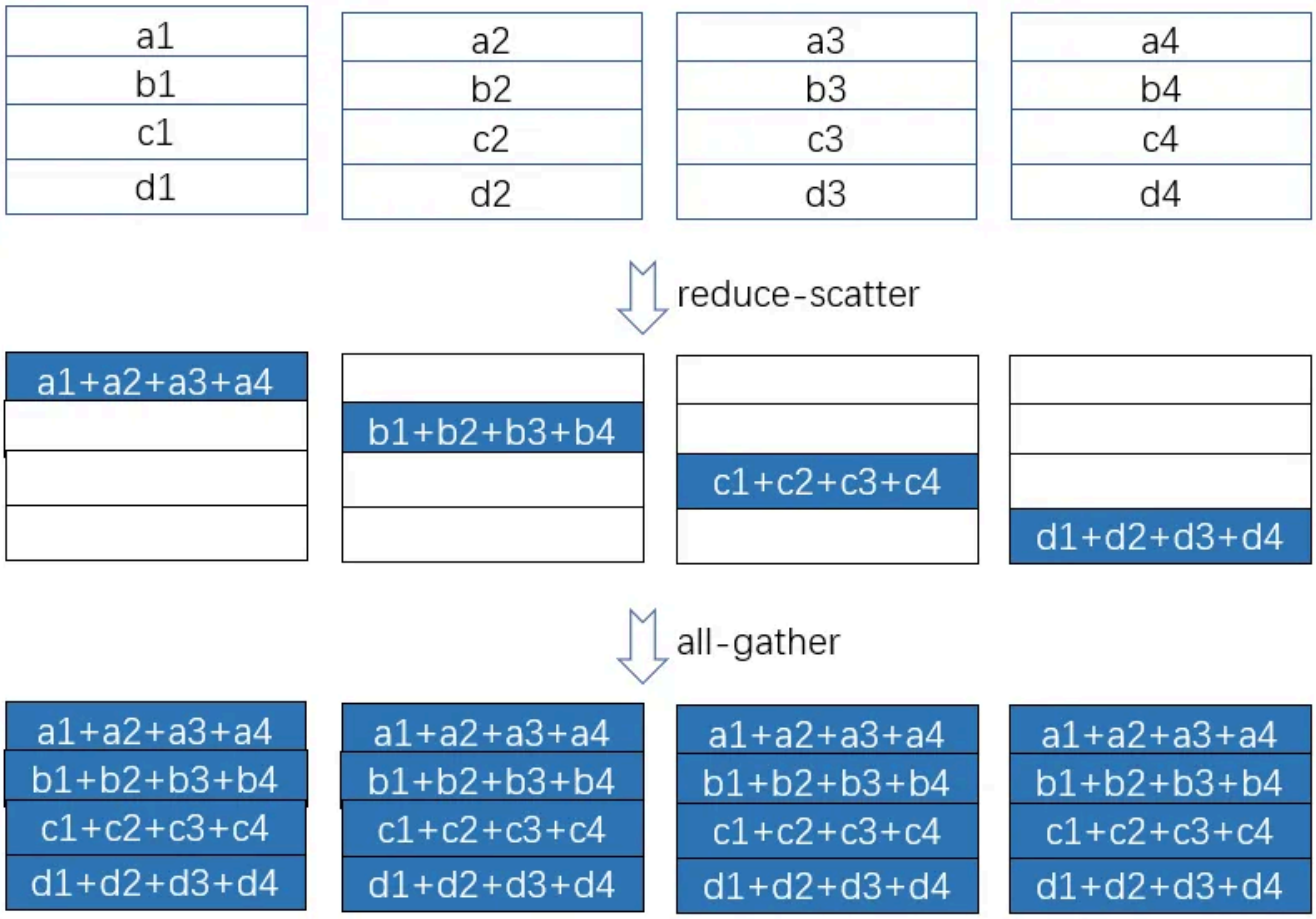
This article will unveil the mathematical property of collective communication operations by analyzing the case of all-reduce, which is common in data parallelism.

1 What is all-reduce



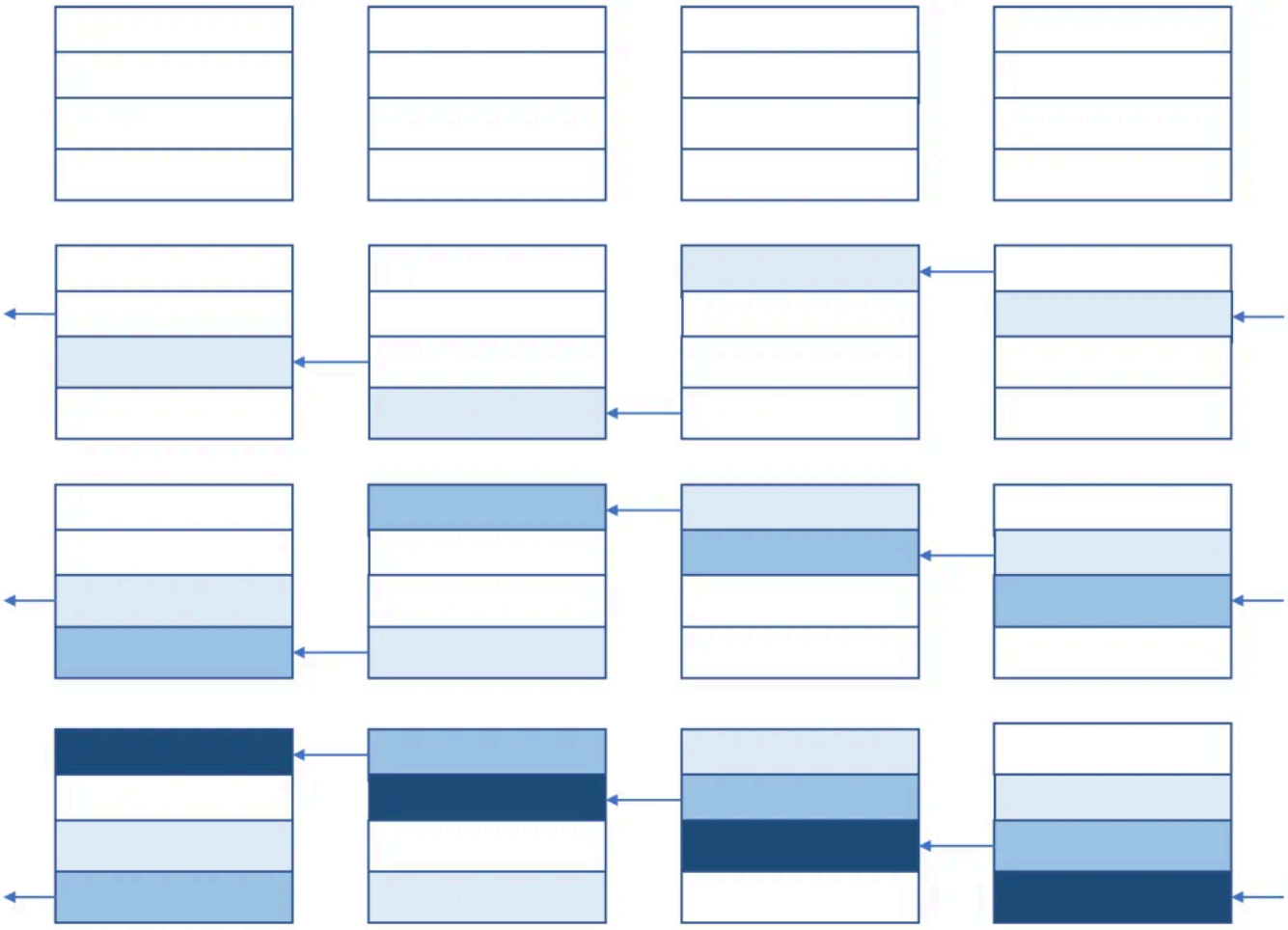
As illustrated in Figure 1, there are four devices, each with one matrix (to keep things simple, each row in these matrices has only one element). And

all-reduce is an operation that sums up the same row's input value across devices and returns the resultant value to the corresponding row.



As shown in Figure 2, the all-reduce operation can also be performed through two more basic collective communication operations, reduce-scatter and all-gather. Besides, a ring communication can efficiently implement reduce-scatter and all-gather operations, as seen in the diagram below.

2 The implementation of reduce-scatter operation and its property



As seen in Figure 2, a reduce-scatter is an operation that reduces input values among devices, with each device only receiving a subpart of the result. Before a further discussion, it is necessary for us to define a few symbols.

We assume that there are p devices (p=4 in the example above), and the size of the matrix is V. Then after a reduce-scatter operation, each device will receive a data chunk with the size of V/p.

If the communication among devices is duplex, and its bandwidth is β, then each device's input/output bandwidth can both reach β, and the sum of input/output bandwidth of all devices will also equal p×β.

The key to high-performance collective communication is fully utilizing bandwidth across devices, which is echoed by collective communication algorithms based on ring communication. Then we'll see how this algorithm works with the case of a reduce-scatter operation.

There are a total of p devices, and the data on each device is divided into p parts, so a ring reduce-scatter operation must perform p-1 steps.

Step 1: Each device takes charge of one data chunk of size V/p and sends this data chunk to the device on its left. Just as Figure 3 shows, device 1 takes charge of data chunk 2 and sends it to device 0 (i.e. device 4), device 2 takes charge of data chunk 3 and sends it to device 1, and the rest of devices perform the same.

Each device receives data from the device on its right and integrates the newly-received data into its originally corresponding data chunk (the color of the data chunk being darker means more data has been cumulated). Under these conditions, **all devices' input/output bandwidth will be efficiently utilized, and there is no need to scramble for bandwidth**. (Apart from ring collective communication, can you suggest another more powerful collective communication operation?)

Step 2: Device 1 sends the cumulated data chunk 3 to device 0 (i.e. device 4), device 2 sends the cumulated data chunk 4 to device 1, and the rest of the devices operate the same.

Each device receives data from the device on its right and integrates the data into its originally corresponding data chunk (then the color of the data chunk becomes darker than in step 1).

Step 3: device 1 sends the cumulated data chunk 4 to device 0 (i.e. device 4), device 2 sends the cumulated data chunk 1 to device 1, and the remaining devices do the same.

Each device receives data from the device on its right and integrates newly-received data into its previously corresponding data chunk (and the color of the data chunk becomes darker than in step 2).

After p-1 steps, each device owns a piece of data being reduced at the corresponding position of all devices. During the whole process, the amount of the data being sent out and received by each device is (p-1)V/p , and the output or input bandwidth is β, so the time needed for the process is (p-1)V/pβ. If p is big enough, the completion time will be close to V/β. What is amazing is that the completion time is irrelevant to the number of devices p. Of course, the amount of data transmitted among all the devices is (p-1)V, which is proportional to p, the number of devices.

It should be emphasized that **the implementation time of collective communication algorithm based on ring communication is almost irrelevant to the number of devices, but the total communication traffic is proportional to the number of devices.**

3 The implementation of all-gather and its property

After the execution of reduce-scatter, all-reduce can be realized through the all-gather process, and all-gather can also be realized through the ring communication algorithm.

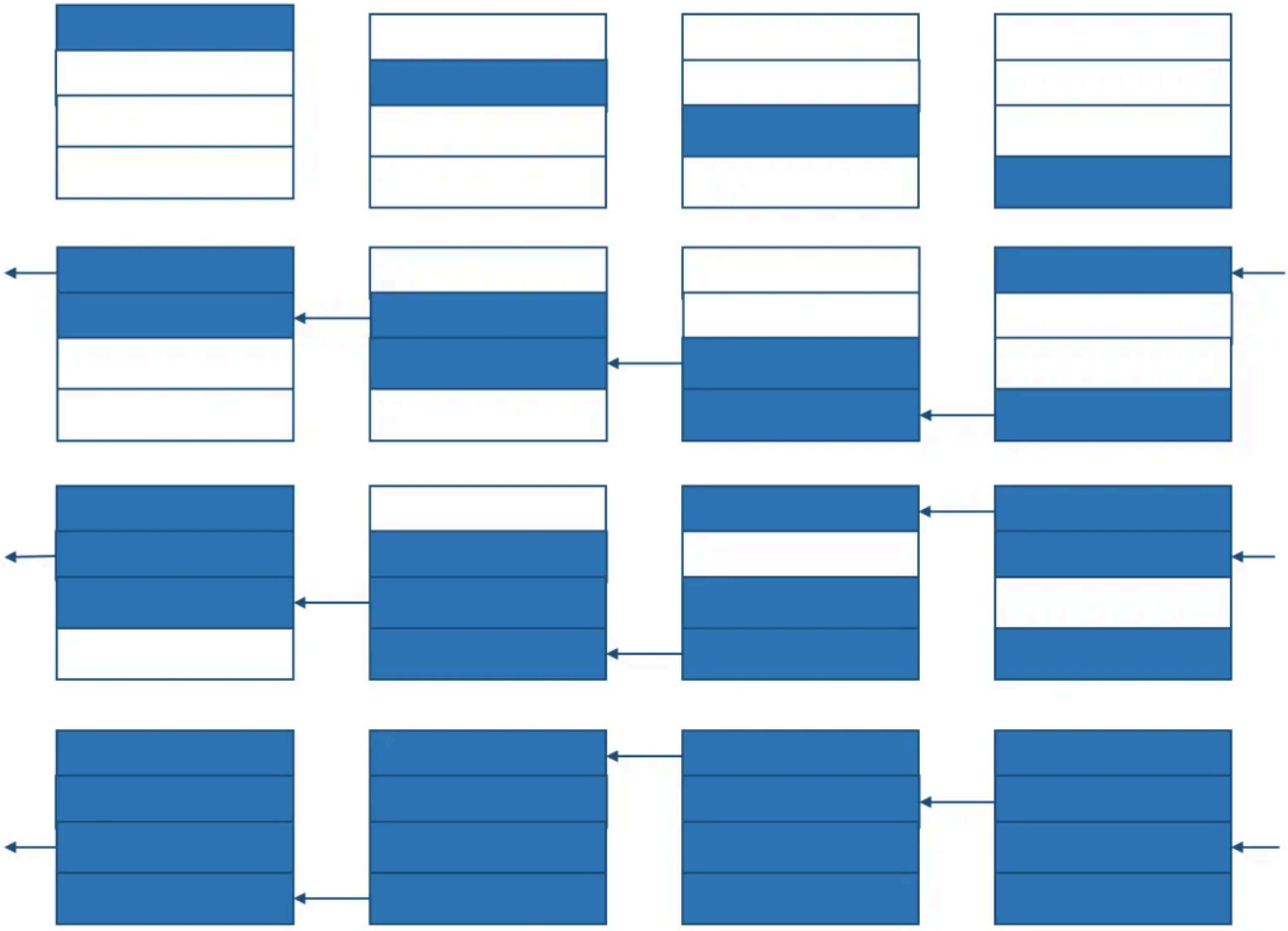


Figure 4 shows the process of implementing a ring all-gather. It is worth noting that its communication time and traffic are just the same as that in reduce-scatter: the time needed for the process is $(p-1)V/p\beta$. If p is big enough, the completion time is close to V/β, which is irrelevant to p, the number of devices. Of course, the amount of data transmitted among all the devices is $(p-1)V$, which is proportional to p, the number of devices p.

However, in a reduce-scatter algorithm, V refers to the data size of the whole matrix, i.e., the data size of the input matrix in reduce-scatter and the data amount of the output matrix in all-gather.

4 The relationship between communication traffic and redundant memory

The above only analyzes the communication traffic, but not the consumption of device memory. Take Figure 3 as an example, the size of the input matrix in each device is V, but after a reduce-scatter operation, each device only needs V/p space of memory, which means (p-1)V/p of space is redundant. There are a total of p devices, so in each cluster, (p-1)V of memory can be saved. Note that the redundant memory in each device is just the same as the communication traffic in each device, and the redundant memory in all devices is the same as the overall communication traffic in all devices.

Take Figure 4 as an example. The size of the input matrix on each device is V/p, but after an all-gather operation, the memory needed for each device is V, and the size and value of matrix on each device are identical. In other words, after an all-gather operation, different devices stores some identical data, which causes memory redundancy. Similarly, **the volume of redundant memory on each device equals that of communication traffic on each device, so the redundant memory on all devices also equals the overall communication traffic.**

Of course, the equivalence of redundancy and communication traffic is not accidental. It is the communication that causes the data redundancy between devices.

So, **when V is kept unchanged, increase p, the number of devices (let's call p as the parallel width of collective communication), and the communication traffic between all devices will increase proportionally, and the redundant memory in all devices will also increase proportionally. Of course, the time needed to complete a certain collective communication is almost irrelevant to p, the parallel width.**

So, to increase the parallel width p is a double-edge sword. On the one hand, it makes each device to process less data, i.e., V/p, thus making the computing time shorter. But on the other hand, it requires more communication bandwidth (p-1)V, and more memory space (p-1)V.

5 The optimality of ring algorithm

We raised a question above: can you think of an implementation of collective algorithm better than ring algorithm? The answer is, theoretically, there isn't any better algorithm.

We have analyzed that to finish reduce-scatter and all-gather, each device should at least send out (and receive at the same time) a data amount of (p-1)V/p. No matter what algorithm is used, the data amount can't be less.

With this data amount, what's the shortest time needed? The output bandwidth is β, so the shortest time needed for a device to send out data is (p-1)V/pβ, which is also the time needed for ring algorithm.

Of course, the communication time here only includes the transmission of bandwidth, but not the latency in each transmission. When the data amount V is relatively big, the latency can be ignored, and the analysis above is true.

But when V is extremely small, or the number of devices p is extremely big, the bandwidth β becomes less important, and latency is more important. In this situation, a tree algorithm will be our first choice, not the ring algorithm. This is why NVIDIA NCCL implements both ring all-reduce and double-tree all-reduce algorithms.

Related articles：

1. ***How to Increase Computational Efficiency for PReLU in CUDA — OneFlow Performance Optimization***
2. ***OneFlow v0.7.0 came out!***

Welcome to visit OneFlow on ***GitHub*** *and follow us on* ***Twitter*** *and* ***LinkedIn*****.**

Also, welcome to join our ***Discord group*** to discuss and ask OneFlow related questions, and connect with OneFlow contributors and users all around the world.

AI | Deep Learning | Pytorch | Oneflow | Machine Learning

Written by OneFlow

167 Followers

OneFlow is a deep learning framework designed to be user-friendly, scalable and efficient.
https://github.com/Oneflow-Inc/oneflow

More from OneFlow

OneFlow

How to Choose the Grid Size and Block Size for a CUDA Kernel?

Written by Juncheng Liu; Translated by Xiaozhen Liu, Chenyang Xu

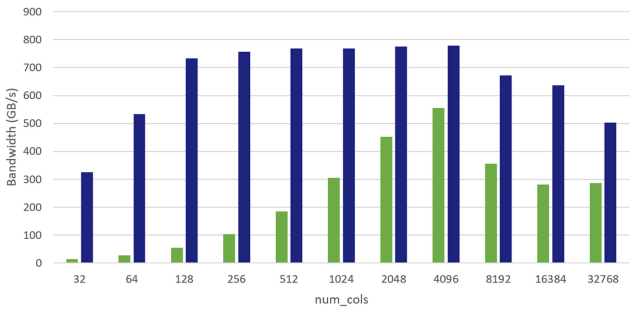
Jan 14, 2022 66 2

OneFlow

How to Implement an Efficient LayerNorm CUDA Kernel—...

Written by Ran Guo, Chi Yao, Zekang Zheng, Juncheng Liu; Translated by Xiaozhen Liu,...

Dec 23, 2021 17 2



OneFlow

How to Implement an Efficient Softmax CUDA kernel?—OneFlo...

Written by Ran Guo; Translated by Kaiyan Wang

Nov 26, 2021 65

OneFlow

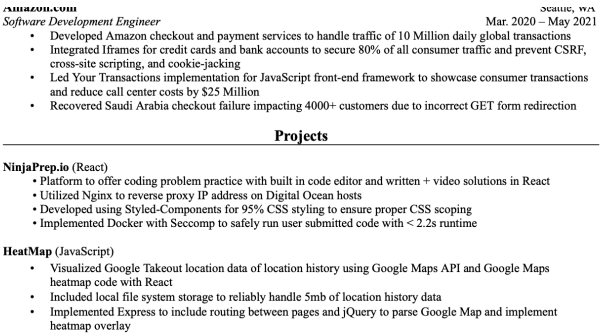
The Development of Credit-based Flow Control (Part 2)

Written by Jing Qiao, Chi, Yao; Translated by Xiaozhen Liu, Chenyang Xu, Yakun Zhou

Feb 10, 2022 13

See all from OneFlow

Recommended from Medium


Amazon.com Seattle, WA
Software Development Engineer Mar. 2020 – May 2021
- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

May 31 14.1K 211

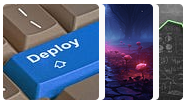
Francis Benistant

Deep Dive into Deep Learning: Layers, RMSNorm, and Batch...

Introduction:

Mar 14 95

Lists



Predictive Modeling w/ Python

20 stories · 1394 saves

Practical Guides to Machine Learning

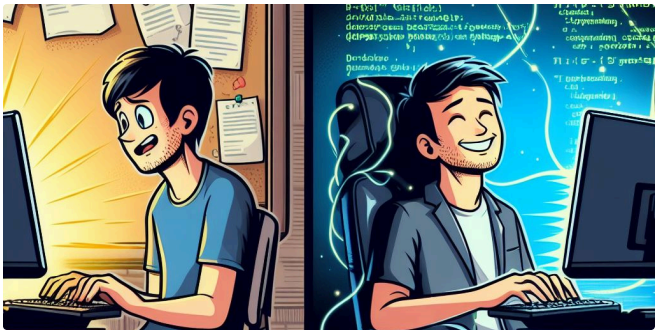
10 stories · 1687 saves

Natural Language Processing

1592 stories · 1143 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 423 saves

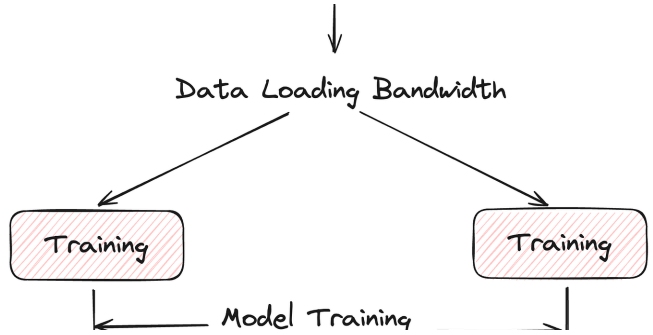


Abhay Parashar in The Pythoneers

17 Mindblowing Python Automation Scripts I Use Everyday

Scripts That Increased My Productivity and Performance

Jul 10 3.7K 33

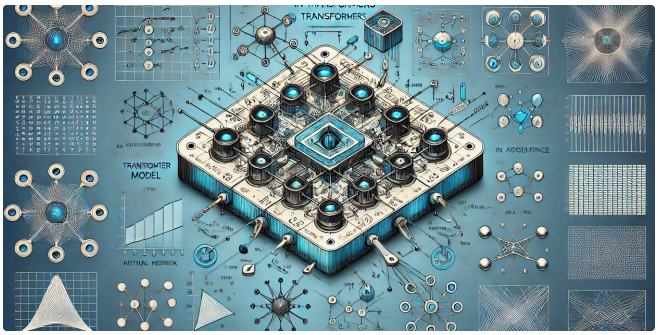


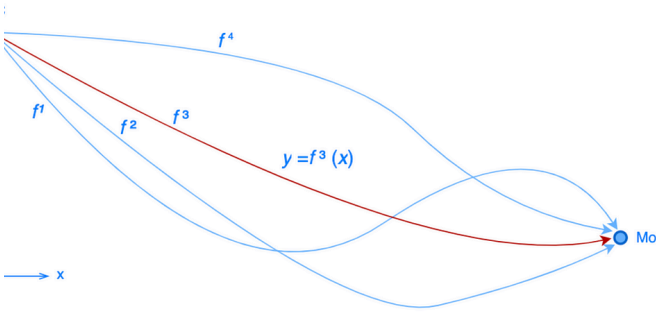
Jonathan Nguyen

Distributed Machine Learning Training (Part 1 — Data Parallelism)

With the ever-increasing size and complexity of datasets, the need for efficient and scalabl...

Mar 25





Cristian Leo in Towards Data Science

The Math Behind Multi-Head Attention in Transformers

Deep Dive into Multi-Head Attention, the secret element in Transformers and LLMs....

6d ago 265 2

Jonathan Hui

Optimization: Calculus of Variations

To find the optimal values of a function f, we solve for the points where its derivative f'...

Jul 3 35

See more recommendations