HUGECTR – 端到端点击率预估训练解决方案介绍（二）

15 Nov 2019

NVIDIA

AGENDA

HugeCTR Introduction

Embedding on GPU

Pipeline on GPU

HUGECTR INTRODUCTION

WHAT IS HUGECTR

HugeCTR is a high efficiency GPU framework designed for Click-Through-Rate (CTR) estimating training.

Key Features in 2.0:

- GPU Hashtable and dynamic insertion
- Multi-node training and enabling very large embedding
- Mixed precision training

HOW HUGECTR HELP

1. Prototype: Showing performance and possibility on GPUs. (v1.0)
2. Reference Design: Developers and NV can work together to modify HugeCTR according to the specific requirements (v2.0 current stage)
3. Framework: Developers can train their model easily on HugeCTR (v3.0)

NETWORK SUPPORTED

Embedding + MLP

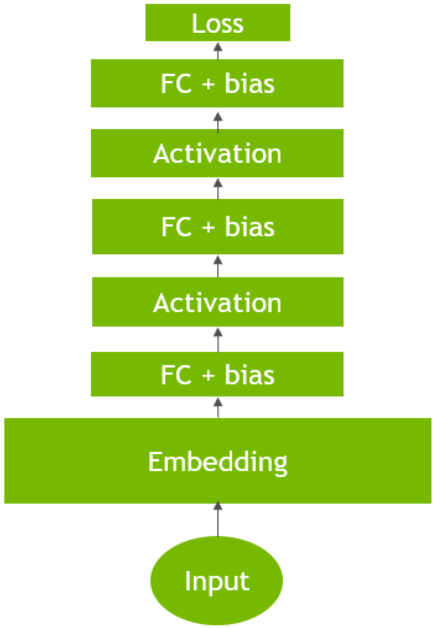
Multi slot embedding: Sum / Mean

Layers: Concat / Fully Connected / Relu / BatchNorm / elu

Optimizer: Adam/ Momentum SGD/ Nesterov

Loss: CrossEngtropy/ BinaryCrossEntropy

* Supporting multiple labels and each label will have a unique weight



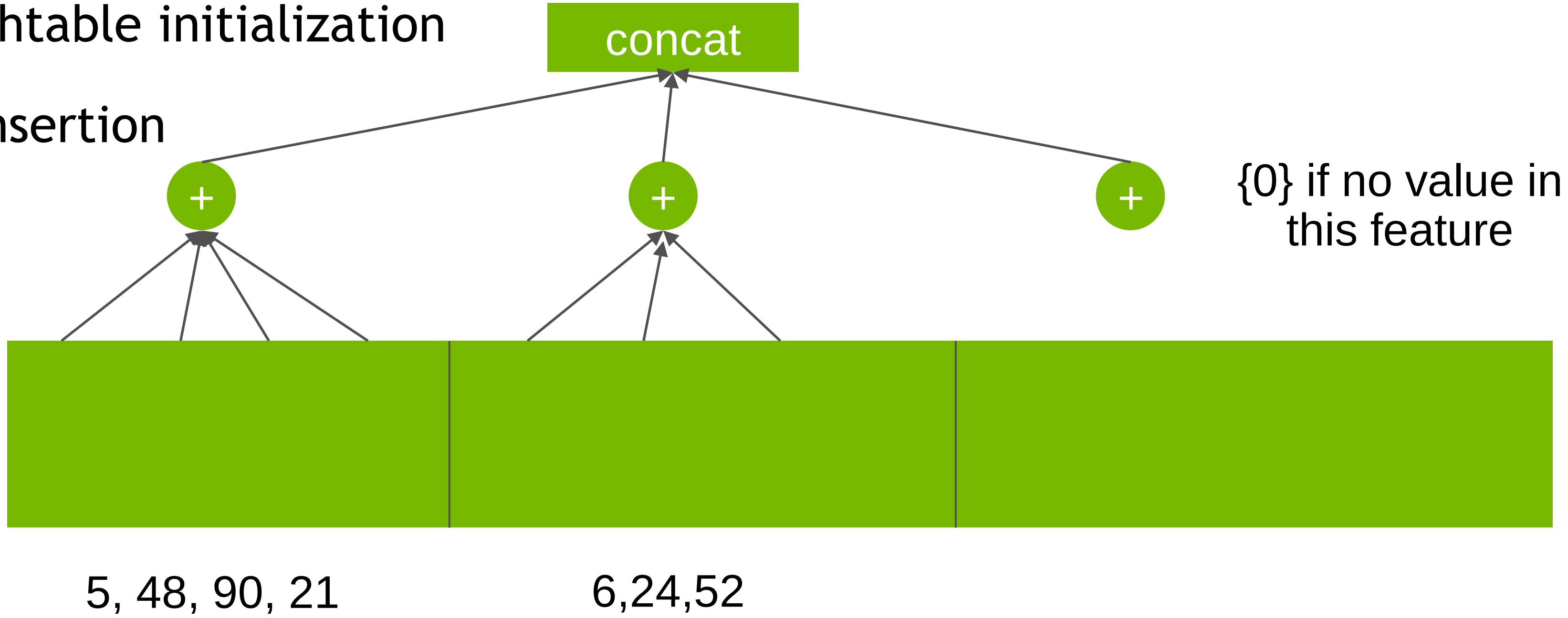
NETWORK SUPPORTED

Sparse Model

Supported reduce ‘+’: sum / mean

Empty Hashtable initialization

Dynamic insertion



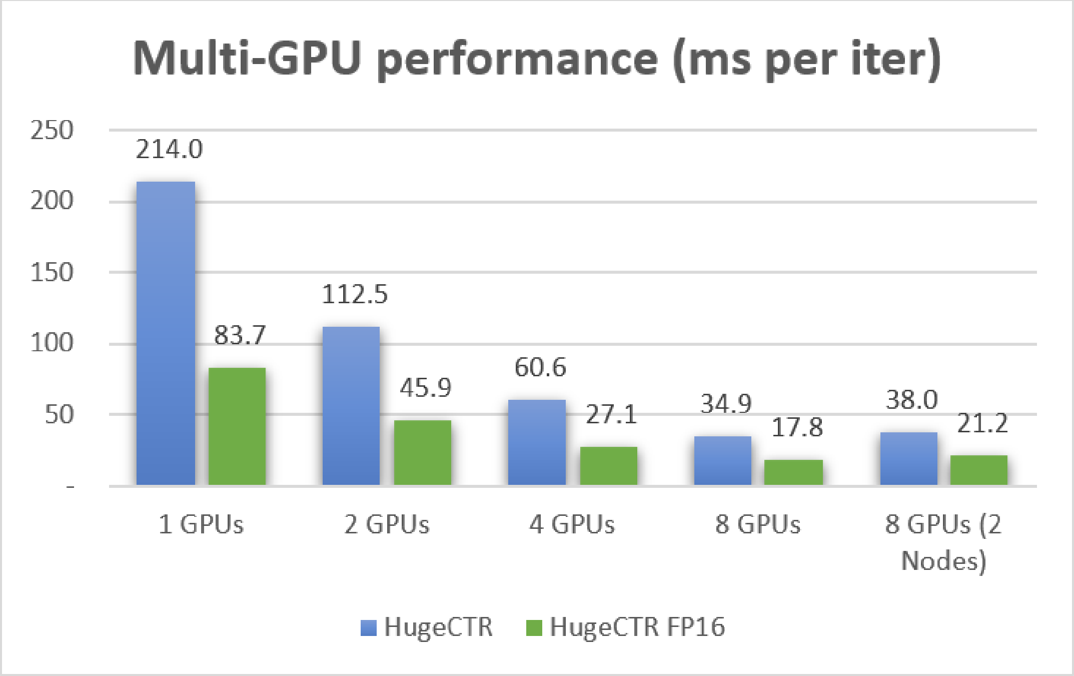
PERFORMANCE

Good Scalability

NCCL 2.0

Three stages pipeline:

- reading from file
- host to device data transaction (inter / intra nodes)
- GPU training

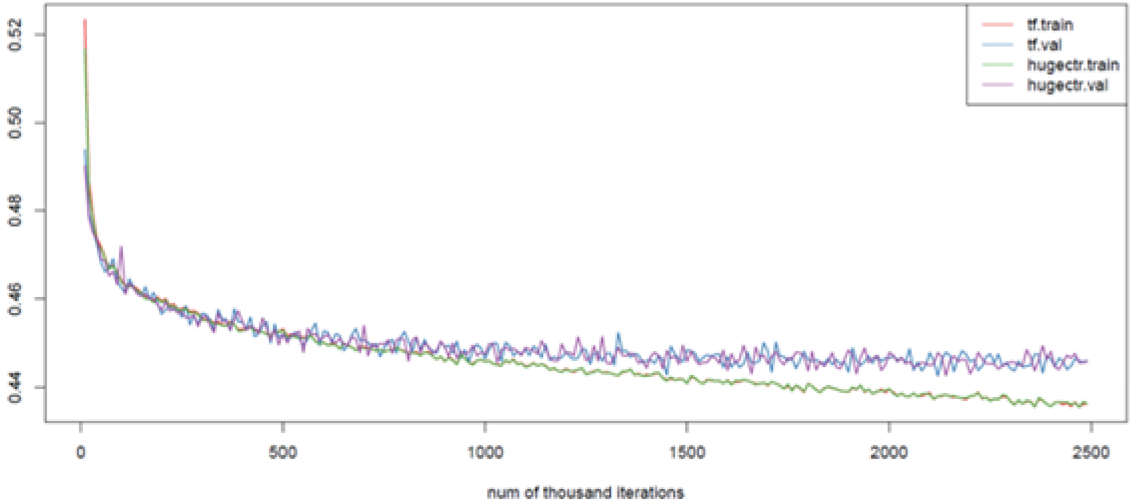


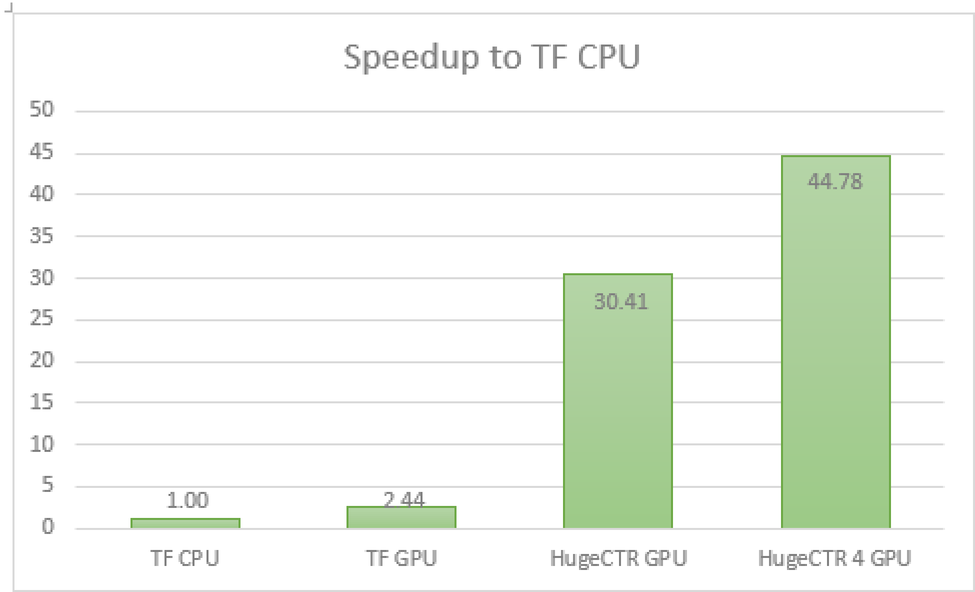
*MLP Layers: 12 / MLP Output: 1024 / Embedding Vector: 64 / Table Number: 1

PERFORMANCE

TensorFlow

44x Speedup to CPU TF and same loss curve





Embedding Vector: 64/ Layers: 4 / MLP Output: 200 / Table Number: 1

PERFORMANCE

Pytorch DLRM

HugeCTR	slot_num	embedding_vec	num_layers	output of MLP
	64	64	4	512

GPUs	Batchsize	HugeCTR Time (s per 200iters)	DLRM (200iters)	Speedup
1	40960	13.5	17.7	131%
2	40960	10.3	19.4	188%
4	40960	6.3	17.3	275%
8	40960	4.3	33.8	786%
1	4096	1.6	4.5	281%
2	4096	1.34	6.5	485%
4	4096	0.9	8.4	933%
8	4096	0.75	13.7	1827%

Embedding Vector: 64 / Layers: 4 / MLP Output: 512 / Table number: 64

RESOURCES

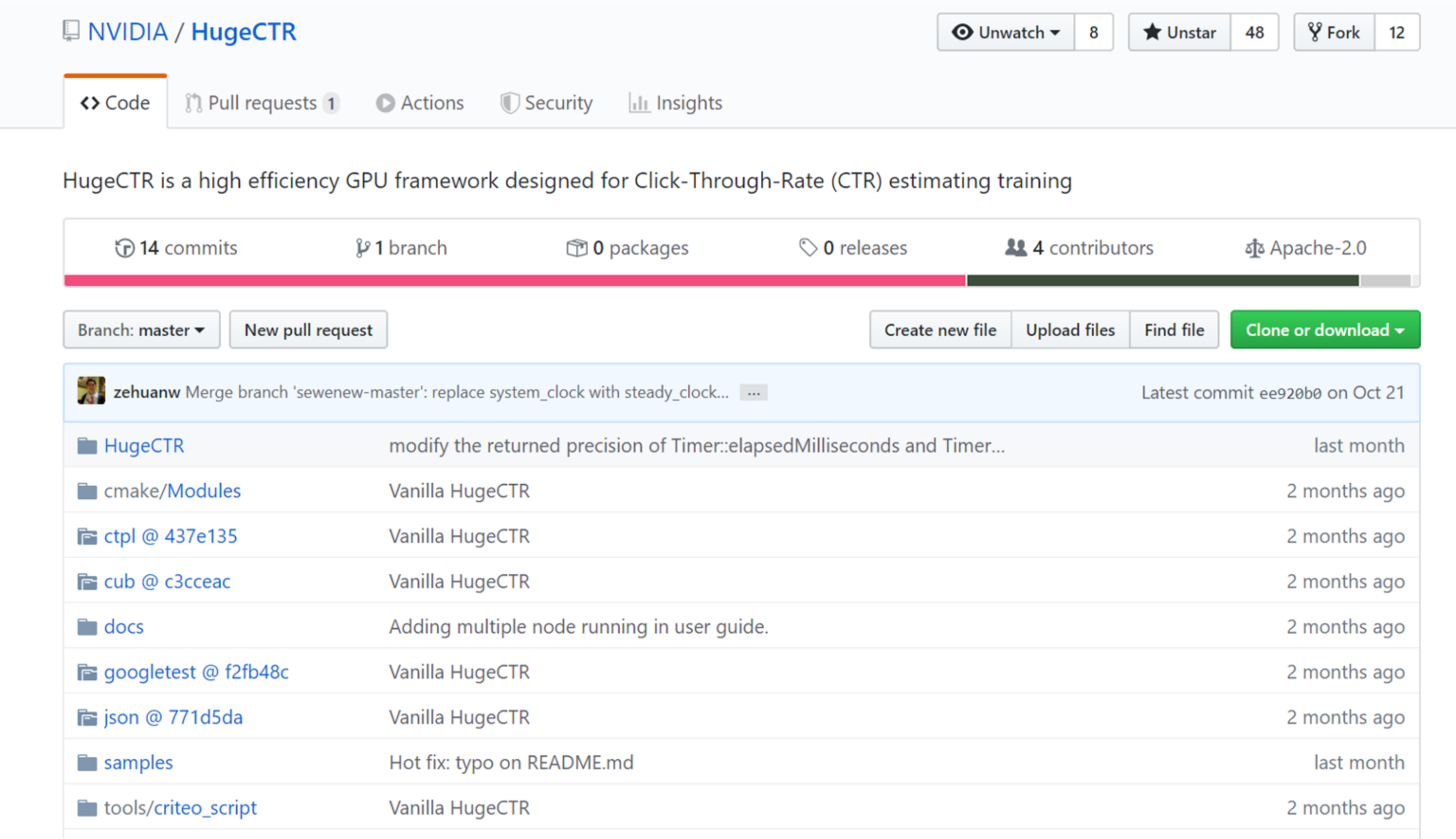
源码：

https://github.com/NVIDIA/HugeCTR

公众号文章：

https://mp.weixin.qq.com/s/Oieuhvt2vzFEfKklTHiuOg

REPOSITORY



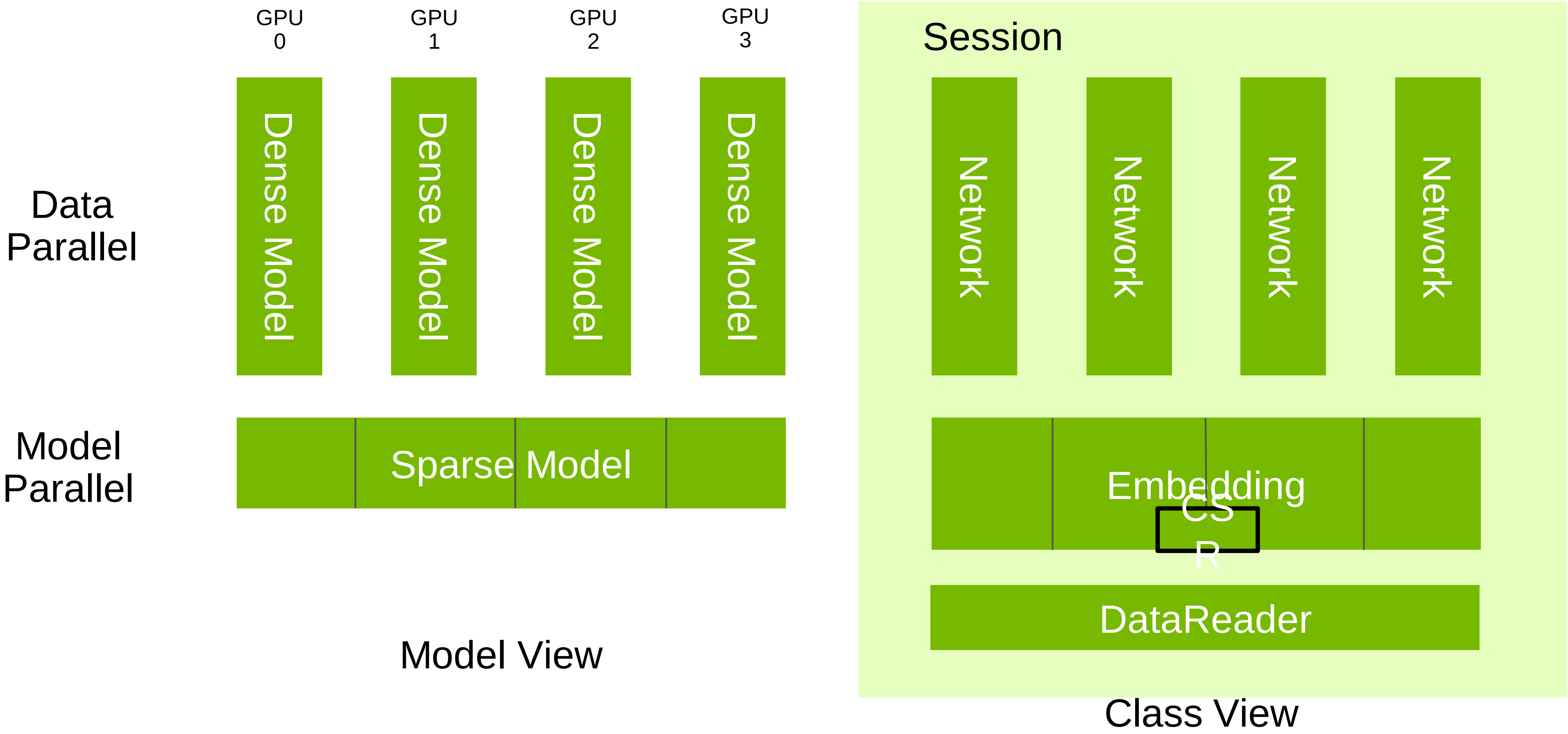
NVIDIA / HugeCTR
Unwatch 8
Unstar 48
Fork 12
Code
Pull requests 1
Actions
Security
Insights
HugeCTR is a high efficiency GPU framework designed for Click-Through-Rate (CTR) estimating training
14 commits
1 branch
0 packages
0 releases
4 contributors
Apache-2.0
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
zehuanw Merge branch 'sewenew-master': replace system_clock with steady_clock...
Latest commit ee920b0 on Oct 21
HugeCTR
modify the returned precision of Timer::elapsedMilliseconds and Timer...
last month
cmake/Modules
Vanilla HugeCTR
2 months ago
ctpl @ 437e135
Vanilla HugeCTR
2 months ago
cub @ c3cceac
Vanilla HugeCTR
2 months ago
docs
Adding multiple node running in user guide.
2 months ago
googletest @ f2fb48c
Vanilla HugeCTR
2 months ago
json @ 771d5da
Vanilla HugeCTR
2 months ago
samples
Hot fix: typo on README.md
last month
tools/criteo_script
Vanilla HugeCTR
2 months ago

FILE SYSTEM

```
/HugeCTR-------------------------source
|      |include---------------------headers
|      |src---------------------------source
|cmake/Modules-----------------cmake modules
|docs-------------------------------documents like user guide and doxygen files
|samples---------------------------Good samples for starters
|tools/criteo_script-------------tools, only data set format convertor now
|utest-------------------------------unit tests for developers
```

EMBEDDING ON GPU

SYSTEM



GPU 0
GPU 1
GPU 2
GPU 3
Data Parallel
Dense Model
Dense Model
Dense Model
Dense Model
Model Parallel
Sparse Model
Model View
Session
Network
Network
Network
Network
Embedding
CSR
DataReader
Class View

APPROXIMATE CSR

Input data (keys):

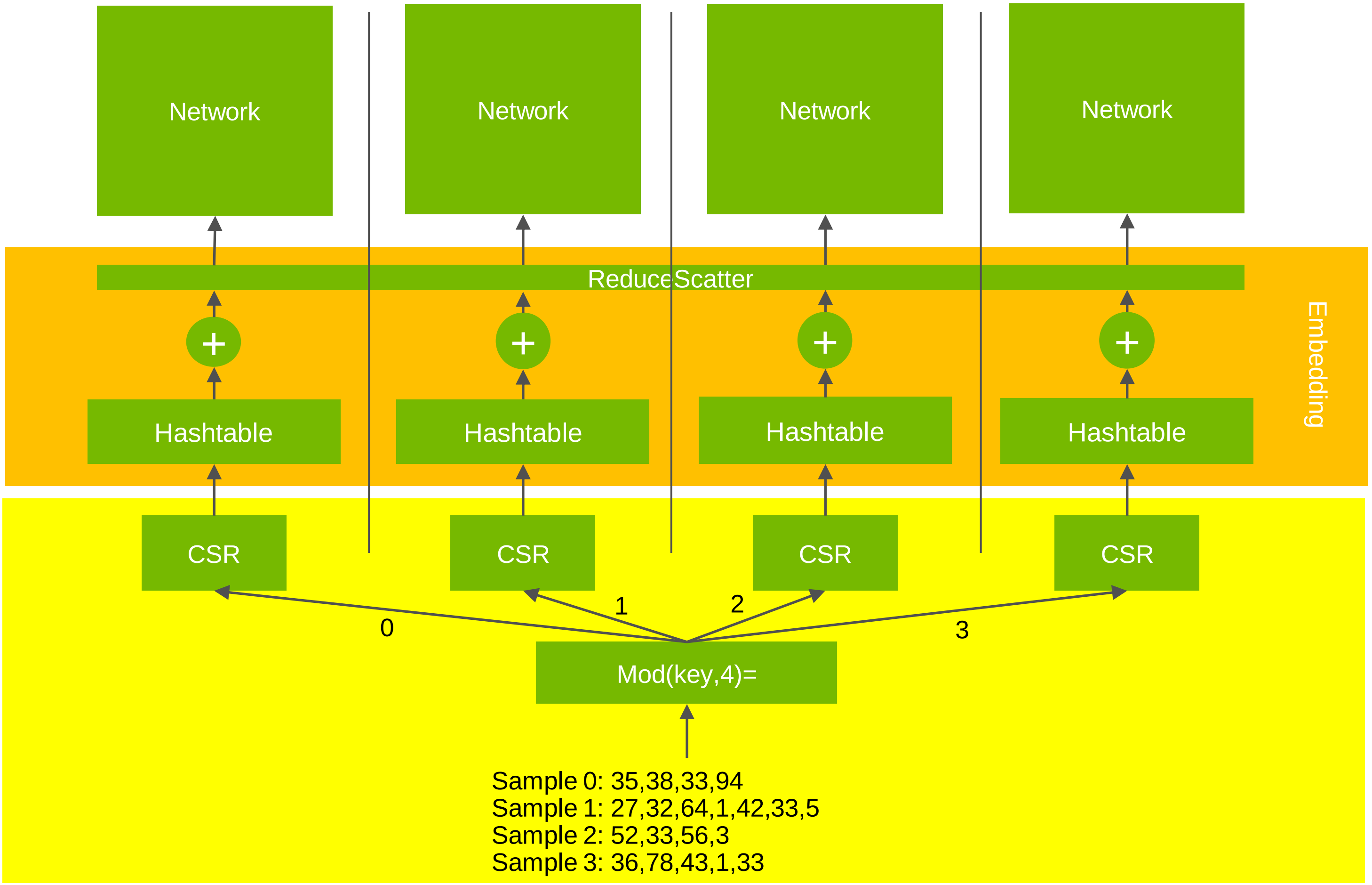
Sample 0: 35,38,33,94
Sample 1: 27,32,64,1,42,33,5
Sample 2: 52,33,56,3
Sample 3: 36,78,43,1,33

CSR:

Row: 0,4,11,15,20

Value:
35,38,33,94,27,32,64,1,42,33,5,5
2,33,56,3,36,78,43,1,33

EMBEDDING



Network
Network
Network
Network
ReduceScatter
+
+
+
+
Hashtable
Hashtable
Hashtable
Hashtable
Embedding
CSR
CSR
CSR
CSR
0
1
2
3
Mod(key,4)=
Sample 0: 35,38,33,94
Sample 1: 27,32,64,1,42,33,5
Sample 2: 52,33,56,3
Sample 3: 36,78,43,1,33

HASHTABLE

GPU Hashtable + Value Matrix (RAW buffer)

1) A hash table of <Key, value-index> pair:

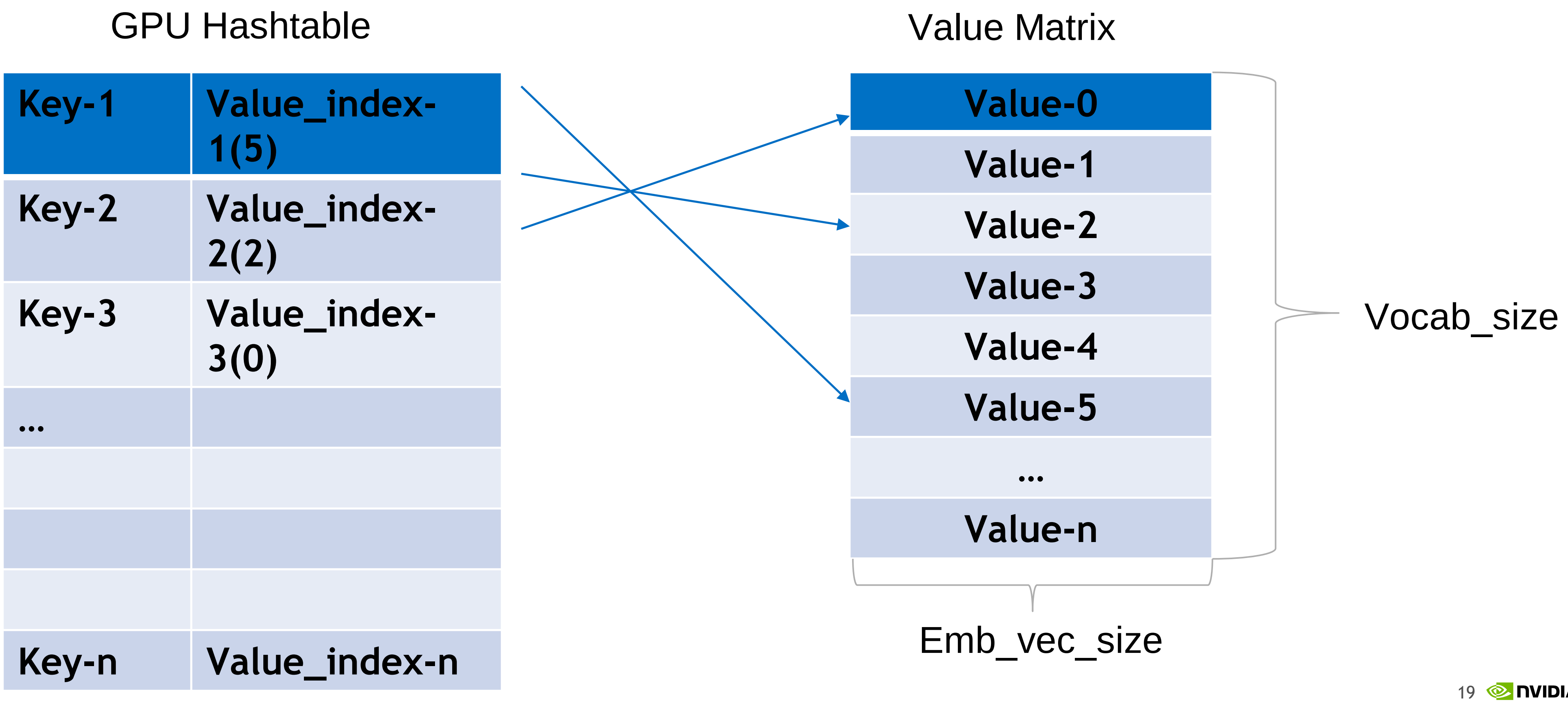
Developed based on cudf in rapids： https://github.com/rapidsai/cudf

The value-index corresponding to the row number of value matrix

Key and value_index have the same type(TypeHashKey): unsigned int or long long

2) Hash table value matrix(just like the embedding table in HugeCTR 1.0): shape = [vocabulary_size, embedding_vec_size]

HASHTABLE


GPU Hashtable
Key-1
Value_index-1(5)
Key-2
Value_index-2(2)
Key-3
Value_index-3(0)
…
Key-n
Value_index-n
Value Matrix
Value-0
Value-1
Value-2
Value-3
Value-4
Value-5
…
Value-n
Vocab_size
Emb_vec_size

HASHTABLE

get_insert()

Fusing get() and insert():

1. Get in hashtable
2. If not found, insert this key and initialized value

How to do:

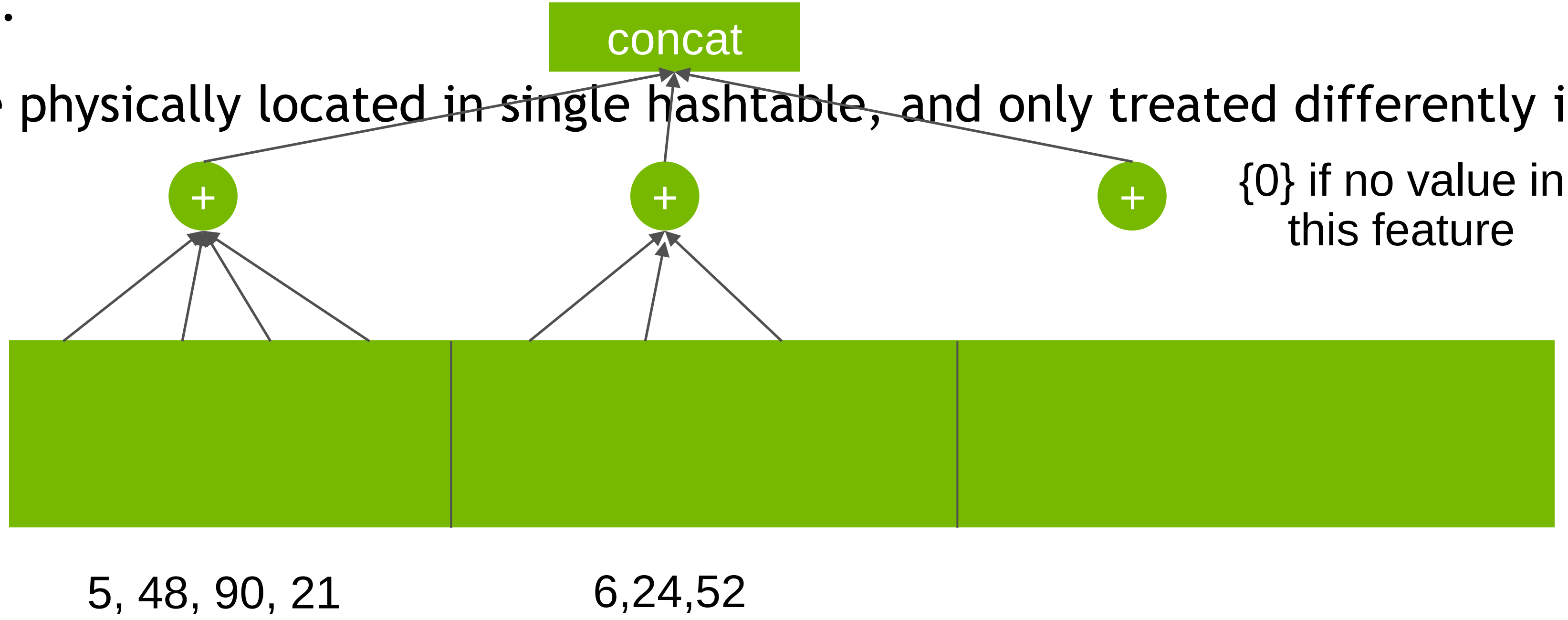
1. Value matrix is initialized with random small numbers.
2. A counter to count the #used rows in this value matrix
3. Once a key is missing, insert <key, counter> into hashtable and counter++

MULTI-TABLE

HugeCTR supports multi-table (feature field/slot) embedding.

Each of the tables will be distributed across all the GPUs (nodes) to avoid load imbalance.

Tables are physically located in single hashtable, and only treated differently in reduction.



BACKWARD

The most important issue in backward:

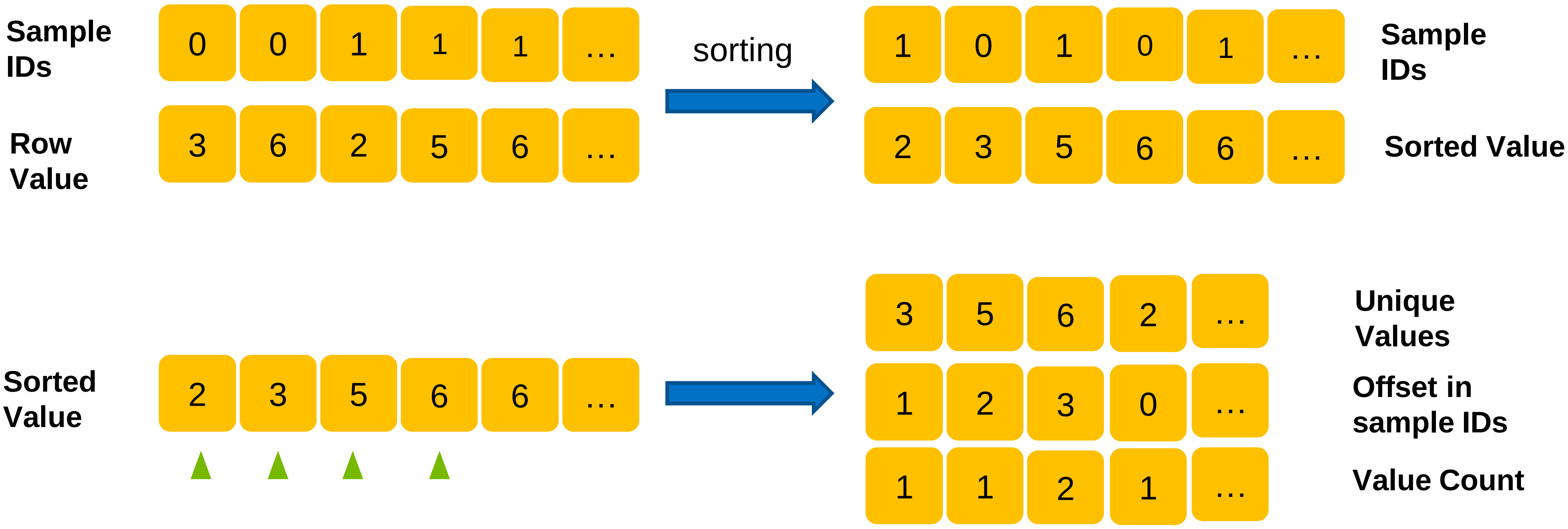
How to find the inversed index of the keys in sample

Sample 0: 35,38,33,94
Sample 1: 27,32,64,1,42,33,5
Sample 2: 52,33,56,3
Sample 3: 36,78,43,1,33

→

1: sample 1 / sample 3
3: sample 2
33: sample 0 / sample 1 / sample 3

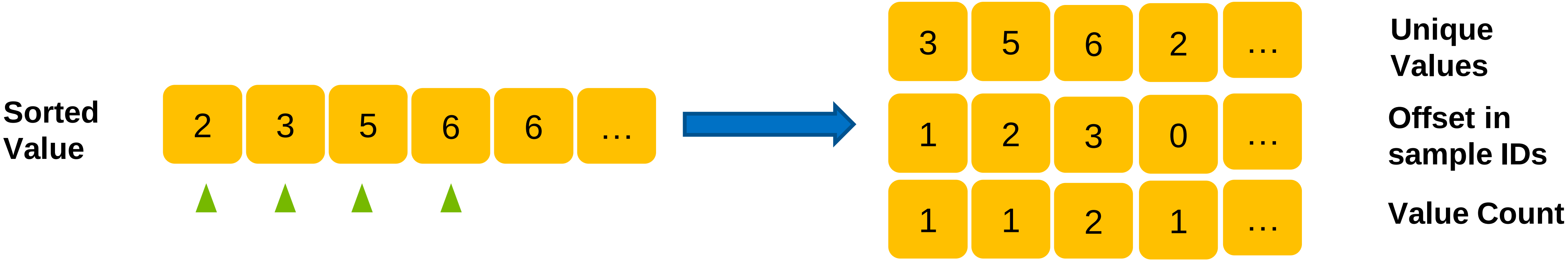
BACKWARD


Sample IDs
0 0 1 1 1 …
Row Value
3 6 2 5 6 …
sorting
1 0 1 0 1 …
Sample IDs
2 3 5 6 6 …
Sorted Value
Sorted Value
2 3 5 6 6 …
3 5 6 2 …
Unique Values
1 2 3 0 …
Offset in sample IDs
1 1 2 1 …
Value Count

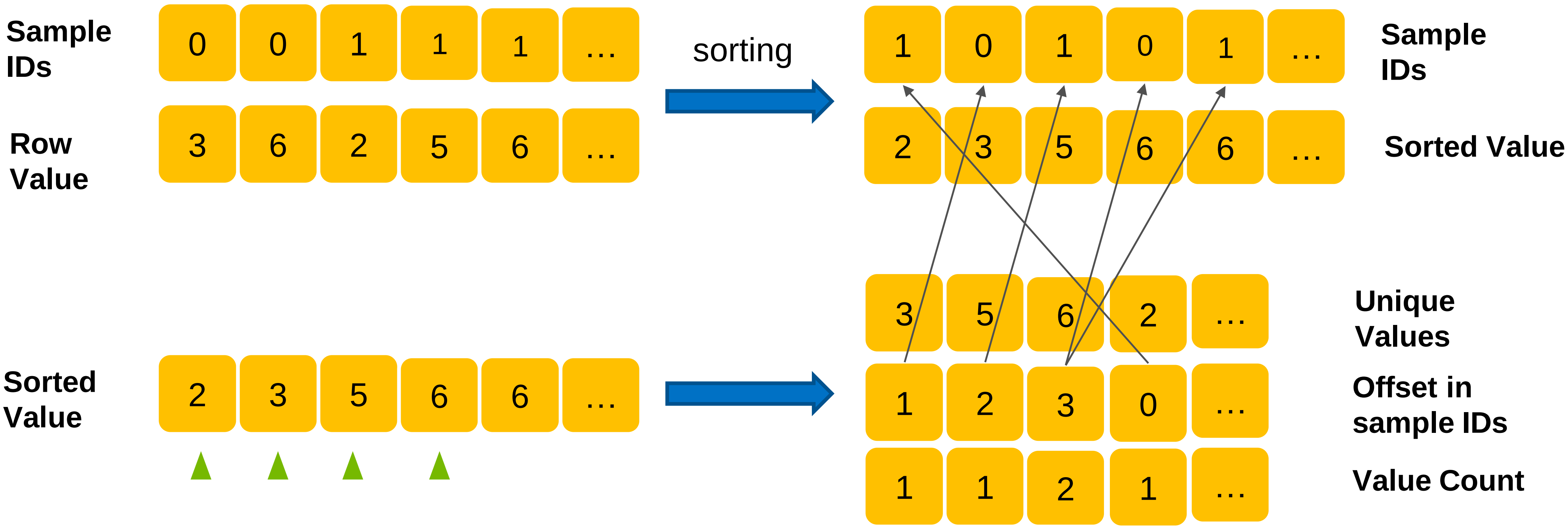
BACKWARD

Each CUDA thread maps to one element (value[tid]) in sorted value.
Each CUDA looks one element in front of it, if different{

```
  offset=atomicAdd(counter,1);
  uniq_values[offset]= value[tid]; offset_in_sample_IDs[offset]=tid;
  v_c = 0;
  do{v_c++}while(value[tid]==value[tid+v_c]);
  value_count[offset]=v_c;
}
```



BACKWARD



Sample IDs
0 0 1 1 1 …
Row Value
3 6 2 5 6 …
sorting
1 0 1 0 1 …
Sample IDs
2 3 5 6 6 …
Sorted Value
Sorted Value
2 3 5 6 6 …
3 5 6 2 …
Unique Values
1 2 3 0 …
Offset in sample IDs
1 1 2 1 …
Value Count

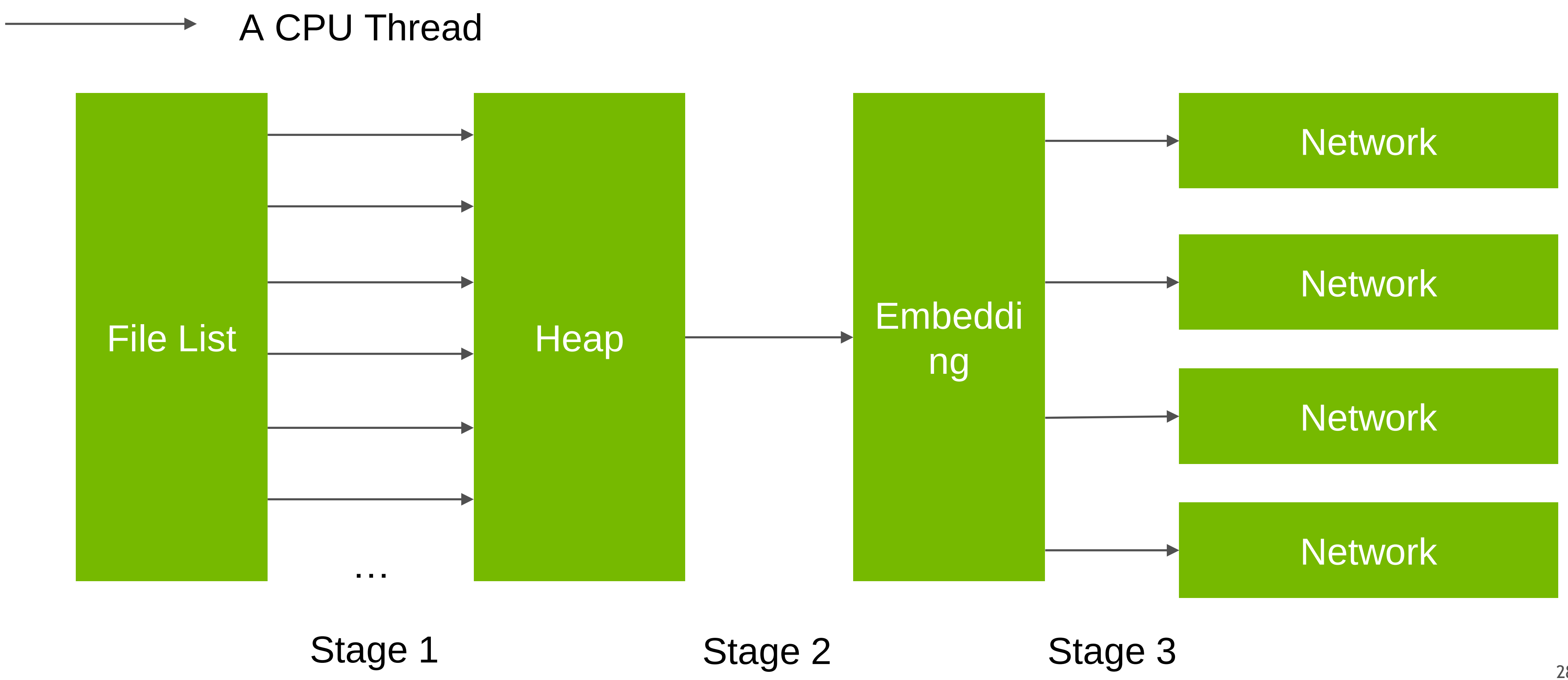
PIPELINE ON GPU

THREE STAGE PIPELINE

CPU / GPU / H2D can overlap each other on any GPU platform.

CPU	Reading File	Reading File	Reading File	
PCI-E		Copy to GPU	Copy to GPU	Copy to GPU
GPU			Train	Train

THREE STAGE PIPELINE



HEAP

Two groups, four routines

Write CSR + label into the Heap, then checkin:

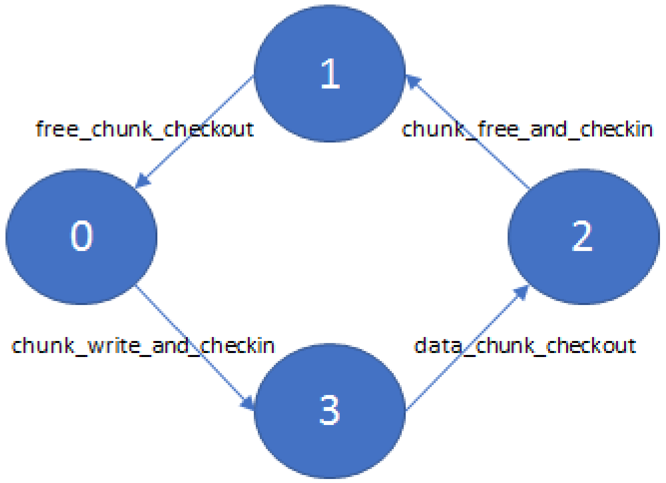
-free_chunk_checkout(ptr: T*) //Blocked function

-chunk_write_and_checkin(ref: const T&)

Read CSR + label from the queue, then free:

-data_chunk_checkout(ptr: T*) //Blocked function

-chunk_free_and_checkin(ref: const T&)



HEAP

uint lower_bits

uint higher_bits

higher_bits	lower_bits	available for read y1	available for write y2
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

y1 = lower_bit&higher_bit

y2 = lower_bit&!higher_bit

Note: R/W to lower_bit and higher_bit should be atomic

KEY CONTRIBUTORS

Fan Yu
Hashtable

Yong Wang
Algorithm Advisor

Ryan Jeng
Competitive Study

Joey Wang
Project Management

Xiaoying Jia
Mixed Precision

Minseok Lee
Multi-Node

David Wu
Embedding

NVIDIA GTC

2019 年 12 月 16 日 – 19 日
苏州, 中国

沟通

与来自 NVIDIA 和其他业界领先组织的技术专家互动。

学习

通过百余场讲座、动手实验和研究海报获取宝贵见解和实践培训。

发现

了解 GPU 技术如何为深度学习等重要领域带来重大突破，描绘最新 AI 世界观。

创新

共同探索改变世界的颠覆性创新，定义未来。

立即注册， 扫码立享 75 折邀请优惠购票

或使用我的优惠邀请码：NVZEHUANW

前往 www.nvidia.cn/gtc/ 完成报名

RYAN JENG
NVIDIA 高級工程師

扫码注册，经典课程+全新主题，
AI实践经验升级

CUDA PYTHON

探讨如何使用 Numba（即时，专用类型的 Python 函数编译器）在 NVIDIA 大规模并行运算的 GPU 上加速 Python 应用程序。

您将学习如何：

- **使用 Numba 从NumPy ufuncs 编译 CUDA 内核**
- **使用 Numba 创建和启动自定义 CUDA 内核**
- **应用关键的GPU内存管理技术**

完成本课程后，您将能够使用Numba编译并启动 CUDA 内核，以加速 NVIDIA GPU上的 Python 应用程序。

NVIDIA