# MassiveClicks

# A Massively-parallel Framework for Efficient Click Models Training

August 28, 2023

Skip Thijssen, Ana-Lucia Varbanescu, Pooya Khandel, Andrew Yates

**University of Amsterdam**

## What is a Click Model?

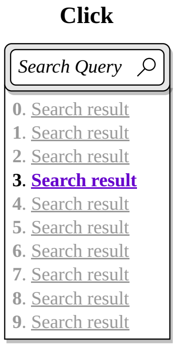
1. Users interact with **search engines**.
2. **Clicks** on search results show **relevance**.
3. Relevance assessments help **improve search engines**[1].

Search
Search Query
0. Search result
1. Search result
2. Search result
3. Search result
4. Search result
5. Search result
6. Search result
7. Search result
8. Search result
9. Search result

[1]Filip Radlinski et al. "How Does Clickthrough Data Reflect Retrieval Quality?"

## What is a Click Model?

1. Users interact with **search engines**.
2. **Clicks** on search results show **relevance**.
3. Relevance assessments help **improve search engines**[1].

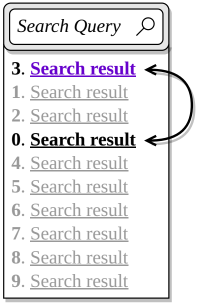


[1] Filip Radlinski et al. "How Does Clickthrough Data Reflect Retrieval Quality?"

## What is a Click Model?

1. Users interact with **search engines**.
2. **Clicks** on search results show **relevance**.
3. Relevance assessments help **improve search engines**[1].

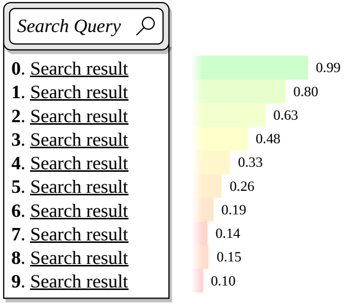
**Improve**



[1] Filip Radlinski et al. "How Does Clickthrough Data Reflect Retrieval Quality?"

# What is a Click Model?

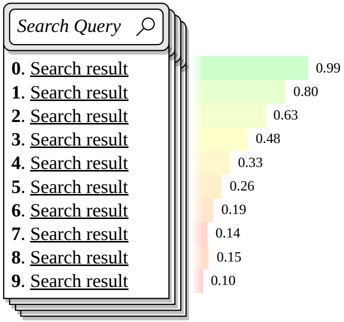
A click model assigns a **click probability** to a search engine result page (SERP).



# How is a Click Model built?

1. A search engine logs a user's clicks on a SERP inside a *click* log.
2. Click models use parameters to quantify relevance
3. EM-based models (for example) are used to train these parameters using data from click logs.

Click logs from real-world search engines can be very large[2] making training expensive.



[2] https://www.internetlivestats.com/google-search-statistics/

# Limitations of Existing Solutions

— **Training** EM-based models is **challenging due to the size** of click logs.

— **Existing tools** like *PyClick* are sequential and slow.

— ***ParClick***, though parallel, is limited to single-node multi-core systems.

## MassiveClicks

*A scalable multi-GPU solution to parallelize generic EM-based training algorithms for click models.*

**Why MassiveClicks?**

— First multi-GPU, distributed click model training framework.

— Efficient GPU kernels and data-partitioning.

— Outperforms ParClick on GPUs/multi-node.

**Requirements**

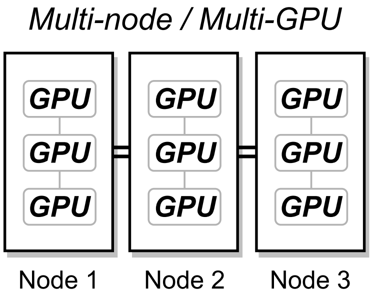
— Scalable EM-based training.

— Efficient, multi-GPU, multi-node distributed processing.

— General, adaptable framework.

## MassiveClicks

*A scalable multi-GPU solution to parallelize generic EM-based training algorithms for click models.*



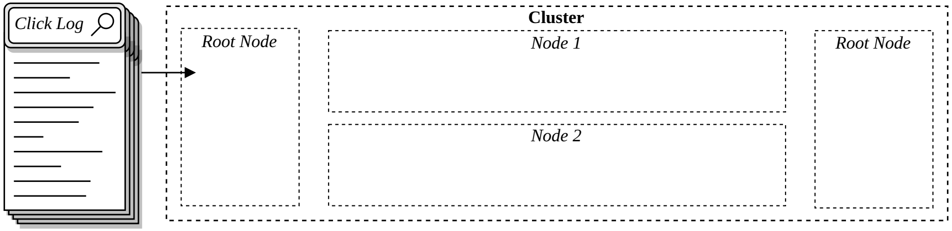
# EM-based Click Models

| Click models supported by MassiveClicks: | *Computational difficulty:* |
|---|---|
| • Position-based Model (PBM) | 8 % |
| • Click Chain Model (CCM) | 88 % |
| • Dynamic Bayesian Network model (DBN) | 100 % |
| • User Browsing Model (UBM) | 10 % |

## EM-based Click Models

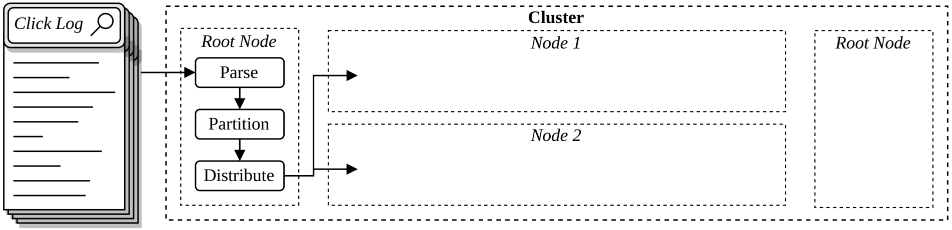
| Click models supported by MassiveClicks: | *Computational difficulty:* |
|---|---|
| • **Position-based Model (PBM)** | **8 %** |
| • **Click Chain Model (CCM)** | **88 %** |
| • Dynamic Bayesian Network model (DBN) | 100 % |
| • User Browsing Model (UBM) | 10 % |

# High-level Design



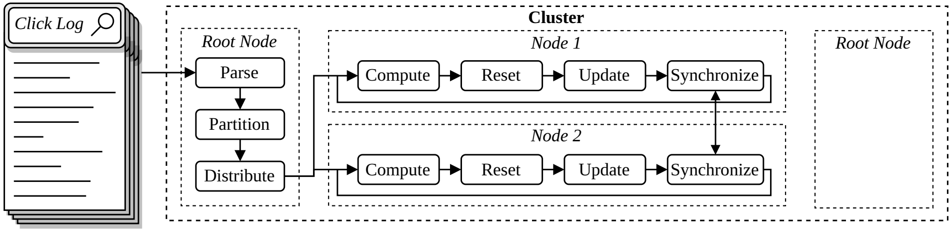
1. Read click log.

# High-level Design



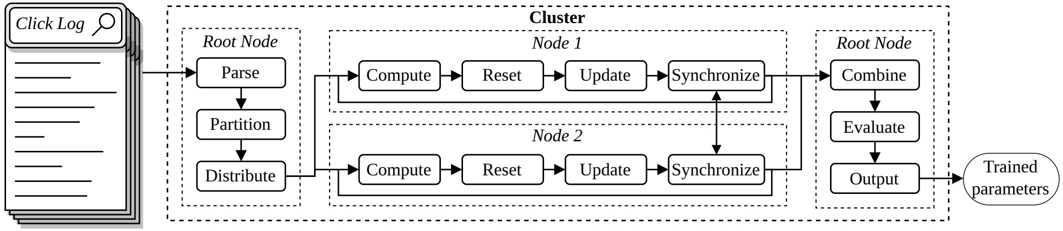
2. Preprocess input.

## High-level Design



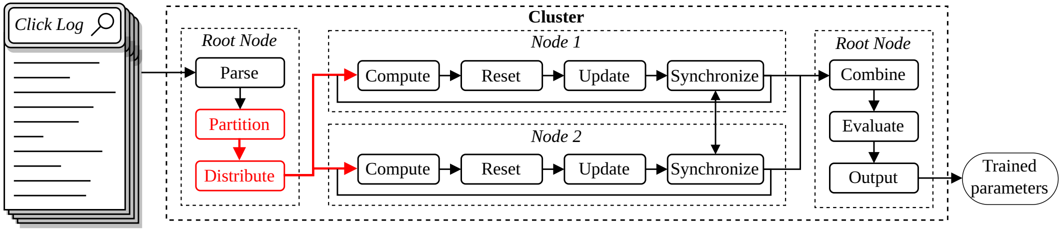
3. Estimate parameters.

## High-level Design



4. Evaluate and output results.

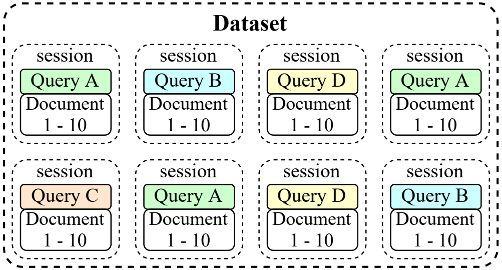
## Data Distribution



Partitioning and distribution

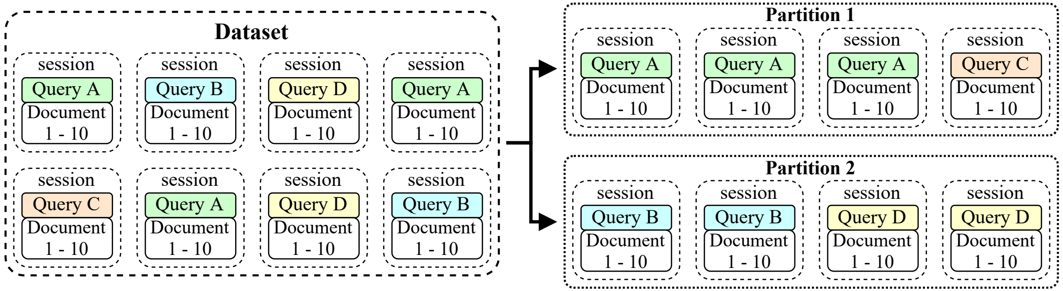
## Data Distribution

Constrain data distribution to **reduce communication** during parameter estimation. Sessions are *grouped by query*.



## Data Distribution

Constrain data distribution to **reduce communication** during parameter estimation. Sessions are *grouped by query*.



## Data Distribution

Nodes *parse sessions independently* from root, ensuring dataset size isn't limited by the root node's memory.

**Difficulties:**

- Similar queries grouped on same node.
- Query IDs on other nodes unknown.
- IDs are inconsistent due to gaps.
- Variable query count per node.
- Differing memory available per node.

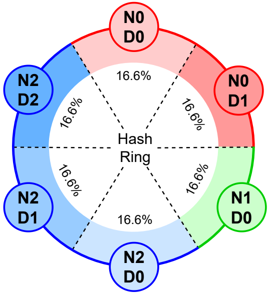
**Desirable Characteristics:**

- Minimize inter-node communication.

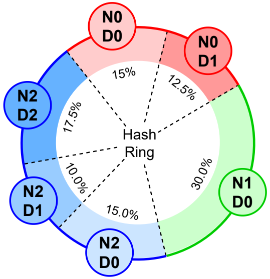
## Data Distribution

Decide session distribution across nodes by adjusting *node ranges* on a *hash ring* based on some property, i.e., number of CUDA cores.

**Uniform Distribution**



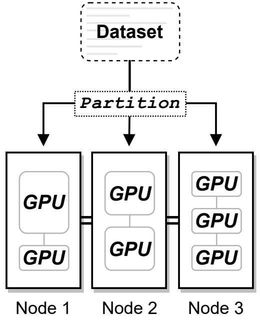
**Property-Based Distribution**



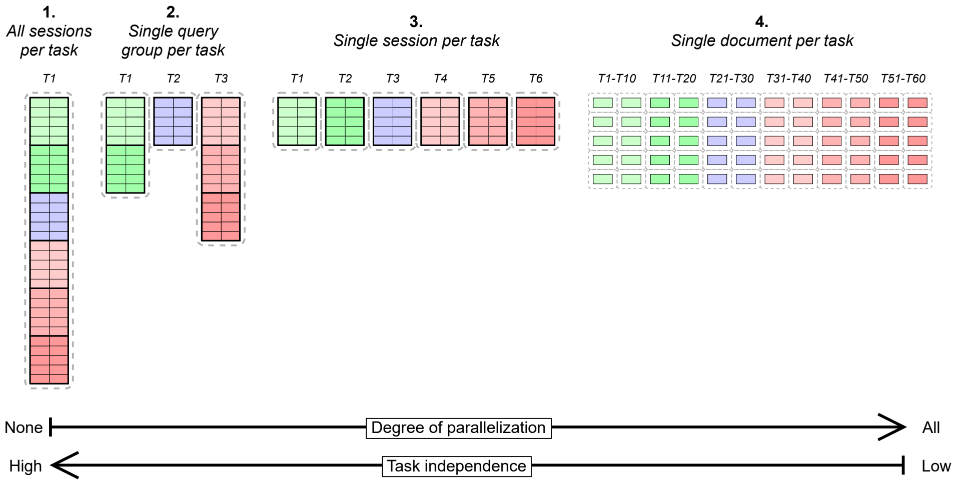
## Data Distribution

Five partitioning policies:

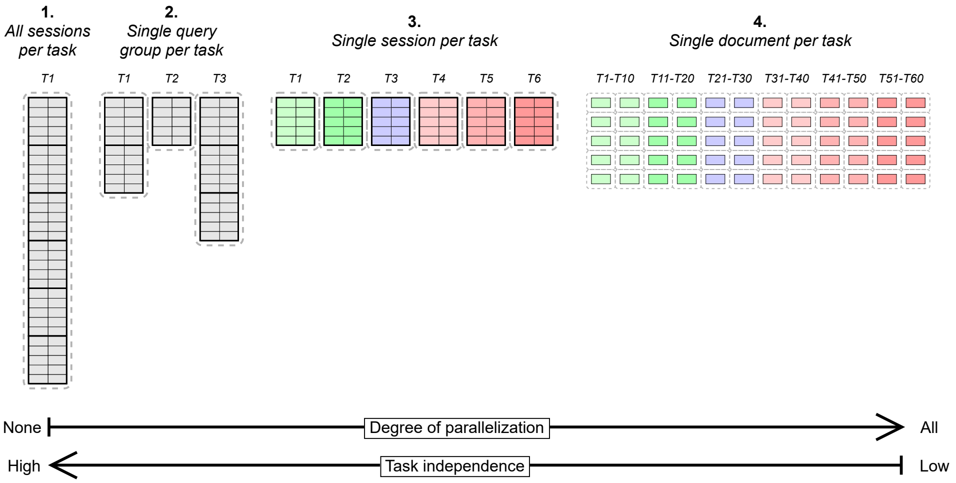
- Round-robin
- Maximum-Utilization
- Proportional to:
    - Available Memory
    - CUDA-core Count
    - Theoretical Peak Performance



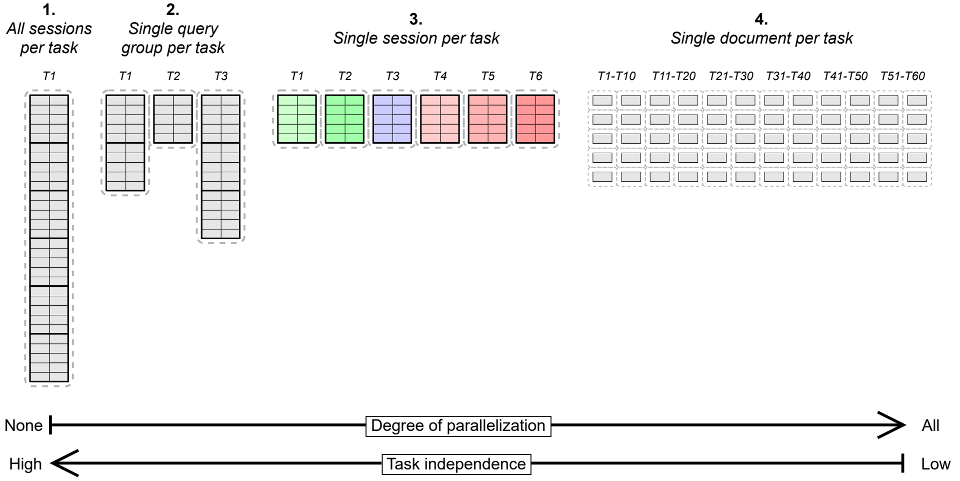
# Session Distribution



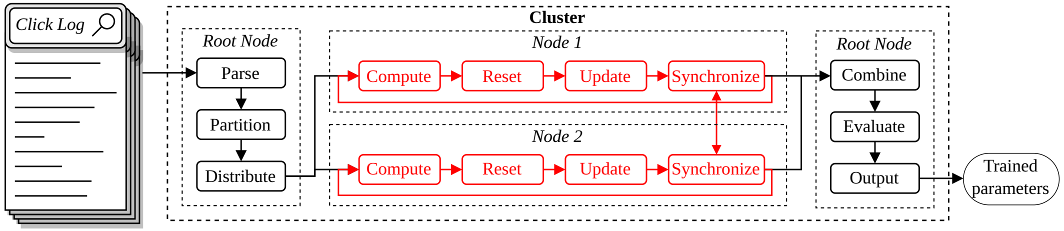
## Session Distribution



## Session Distribution



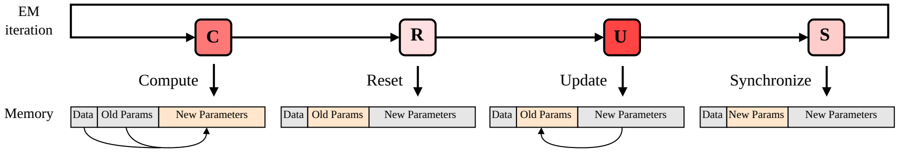
## Parameter Estimation



Iterative parameter estimation

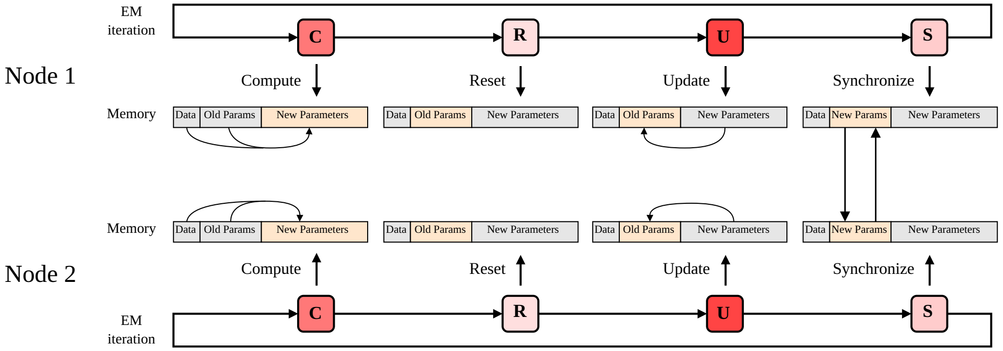
## Parameter Estimation

EM-based parameter estimation is divided into four phases.



# Parameter Estimation

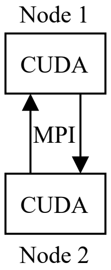
EM-based parameter estimation is divided into four phases.



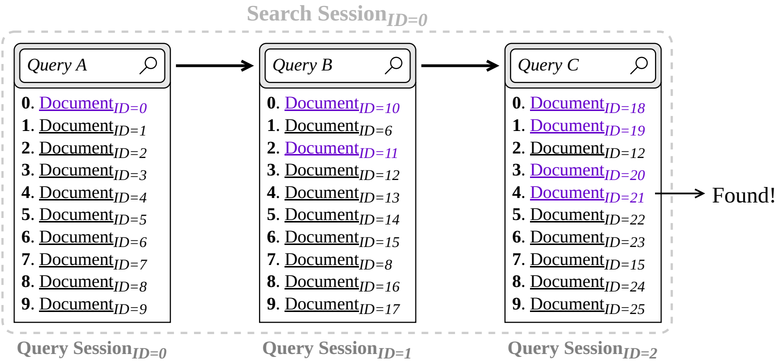
## Multi-GPU/Multi-node Architectures

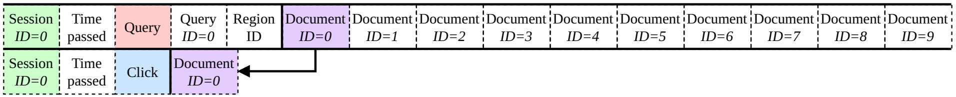
Programming models:

- **C++** framework.
- **CUDA** for (multi) GPU execution.
- **MPI** for multi-node communication.



## Click Log Layout

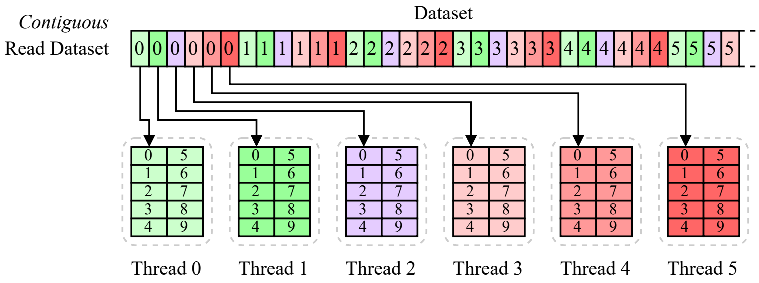




| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Session *ID=0* | Time passed | Query | Query *ID=0* | Region ID | Document *ID=0* | Document *ID=1* | Document *ID=2* | Document *ID=3* | Document *ID=4* | Document *ID=5* | Document *ID=6* | Document *ID=7* | Document *ID=8* | Document *ID=9* |
| Session *ID=0* | Time passed | Click | Document *ID=0* | | | | | | | | | | | |

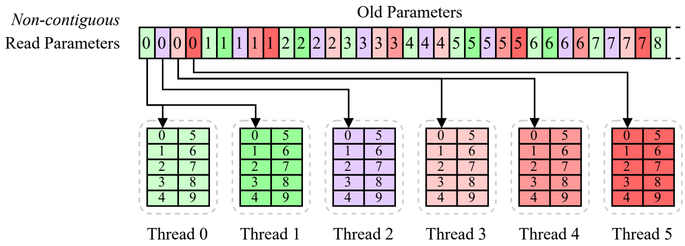
# Allocation in Memory

Dataset placement in global GPU memory.



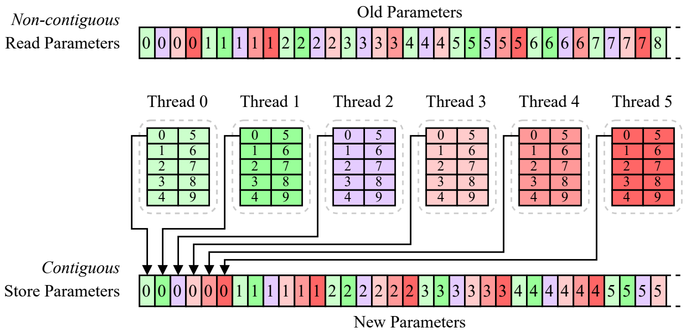
## Memory Access Pattern

**Compute** — Threads *read* parameters from *previous* iteration.



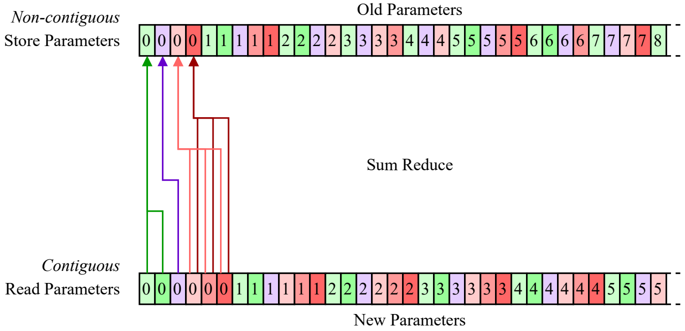
# Memory Access Pattern

**Compute** — Threads *write* parameters from *current* iteration.



# Memory Access Pattern

**Update** — Threads *write* parameters to *previous* iteration.



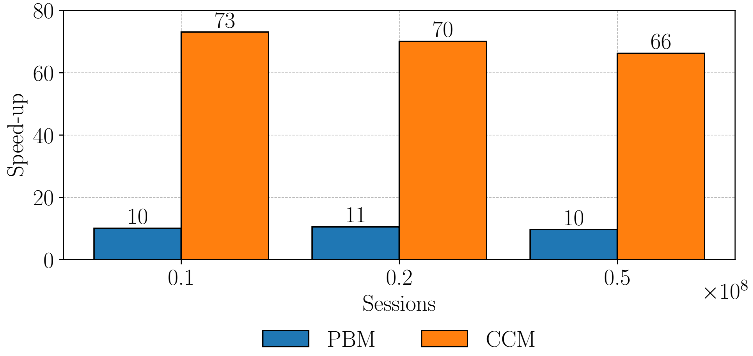
## Experimental Setup

Measure performance for (up to 14) multi-GPU/multi-node configurations using **NVIDIA RTX A4000 GPUs** and 24-core **AMD EPYC 7402P CPUs** and subsets of the Yandex dataset of varying sizes (D$n$ = dataset of $n$ million sessions).

Report:

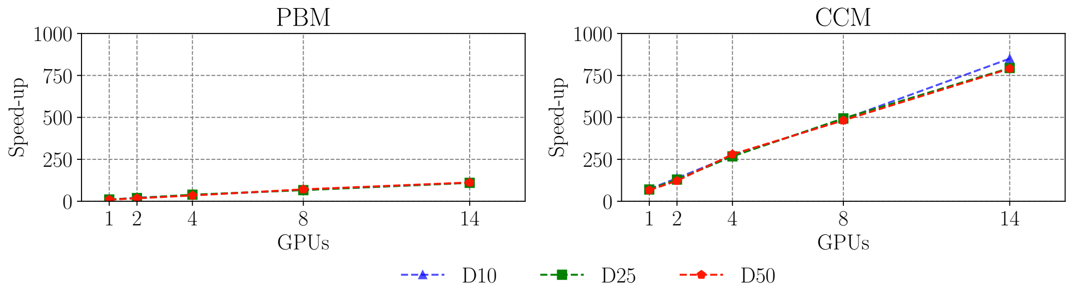
- **Speed-up** vs. ParClick, the **only alternative** for parallel click model training.
- **Scalability** for multi-GPU/multi-node setups and click logs.
- **Usability** for real-world problems.
- **Kernel performance** using a **roofline model**.

## Single GPU Speed-up



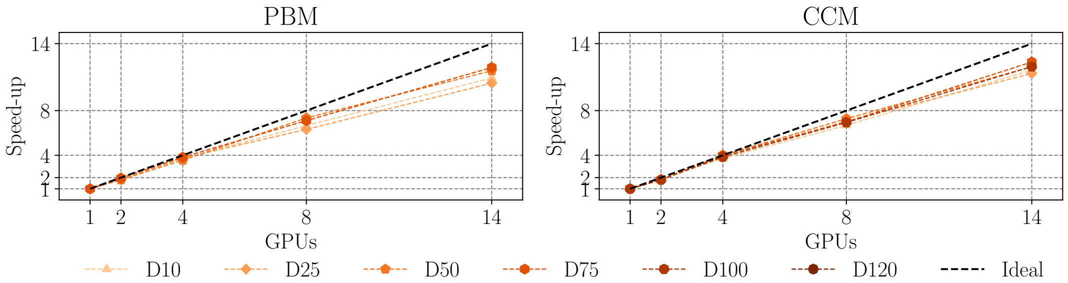
**Speed-up** of MassiveClicks for **PBM** and **CCM** on a single **NVIDIA RTX A4000** GPU.

## Multi-GPU Speed-up



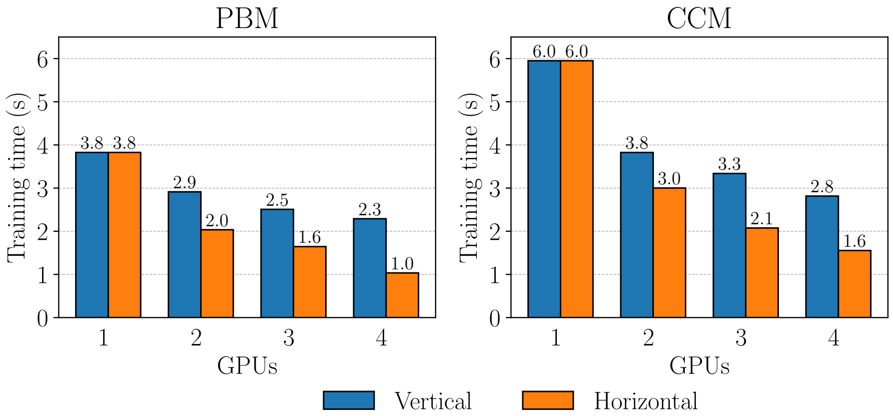
**Speed-up** for **PBM** (left) and **CCM** (right) for different datasets and **NVIDIA RTX A4000 GPUs** compared to ParClick on an **AMD EPYC 7402P CPU** with **48 threads**.

## Scalability



**Scalability** for **PBM** (left) and **CCM** (right) for different datasets and **NVIDIA RTX A4000 GPUs** compared to ParClick on an **AMD EPYC 7402P CPU** with **48 threads**.

## Scalability



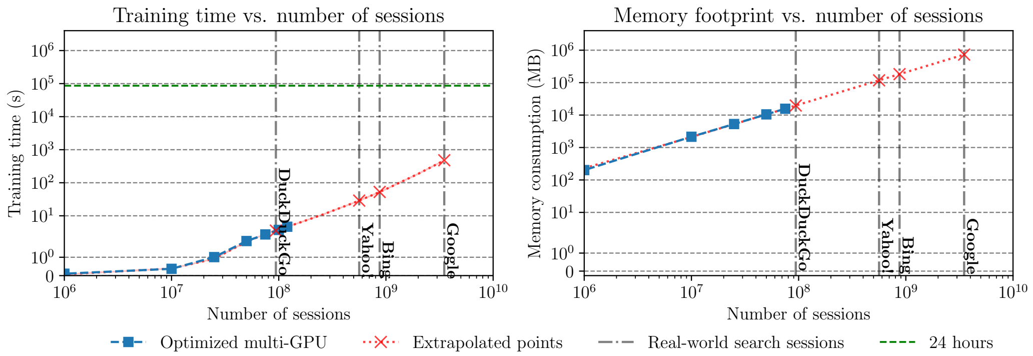
**Training time** of MassiveClicks for **PBM** (left) and **CCM** (right) for **D10** with different number of **NVIDIA RTX A4000** GPUs per node.

## Usability

Real-world applications:

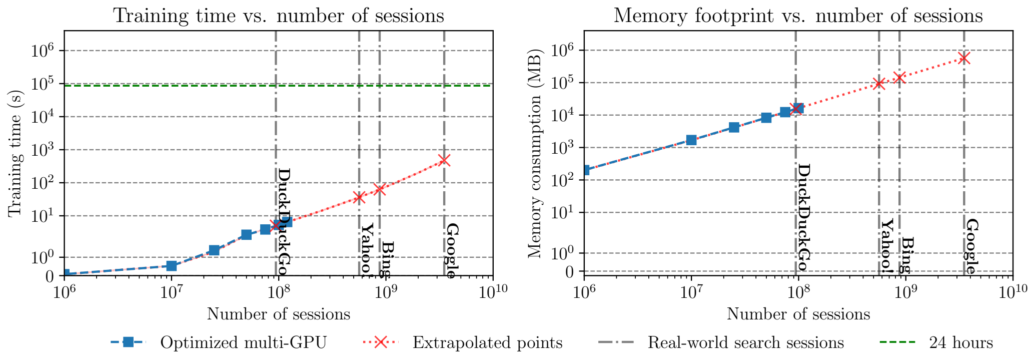
- 'Search engine'-size click logs.
- Energy consumption.

## Projection



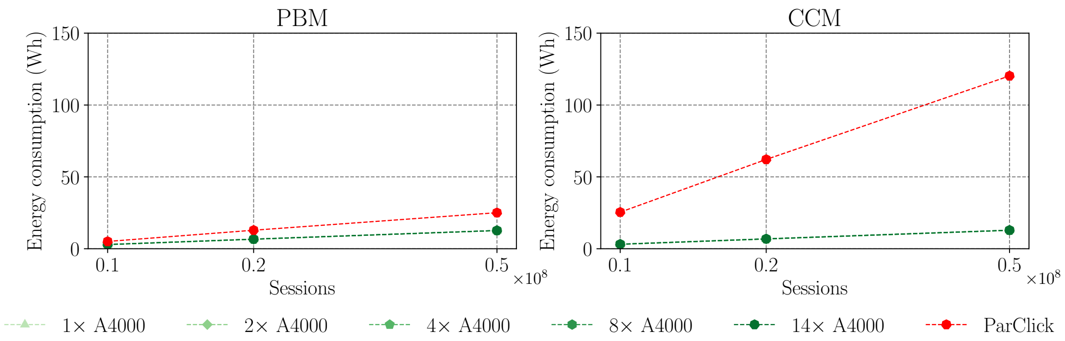
**Training time** (left) and **memory footprint** (right) of MassiveClicks of **PBM** for real-world datasets using 14 **NVIDIA RTX A4000** GPUs.

# Projection



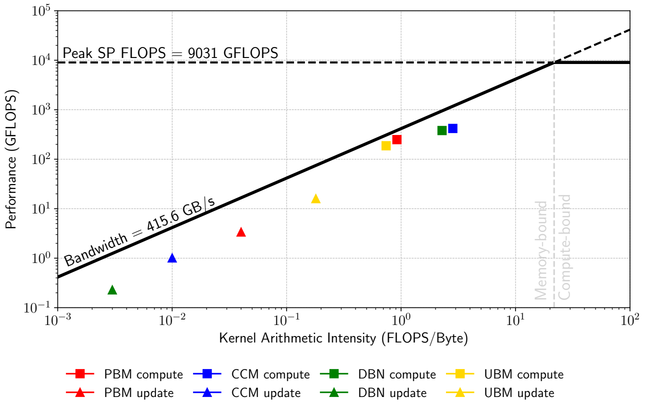
**Training time** (left) and **memory footprint** (right) of MassiveClicks of **CCM** for real-world datasets using 14 **NVIDIA RTX A4000** GPUs.

# Energy Consumption



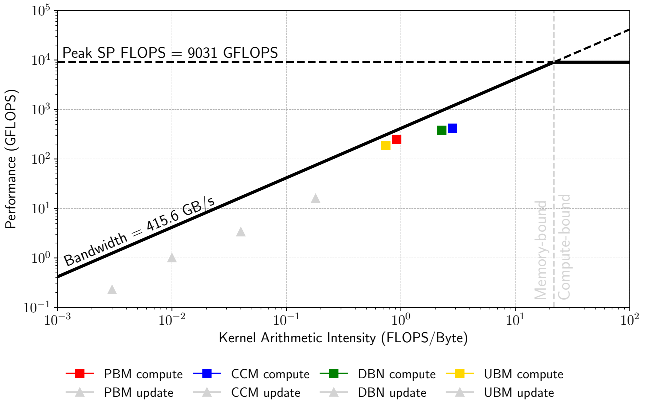
Expected **energy consumption** of MassiveClicks for **PBM** and **CCM** compared to ParClick.

# Kernel Performance



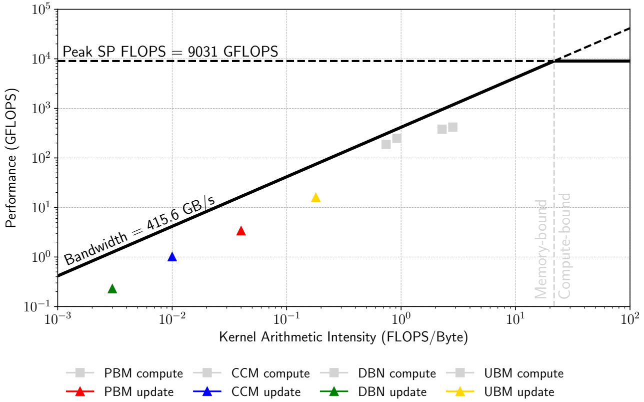
**Roofline model** of MassiveClicks for **PBM**, **CCM**, **DBN**, and **UBM** on a single **NVIDIA RTX A4000** GPU.

# Kernel Performance



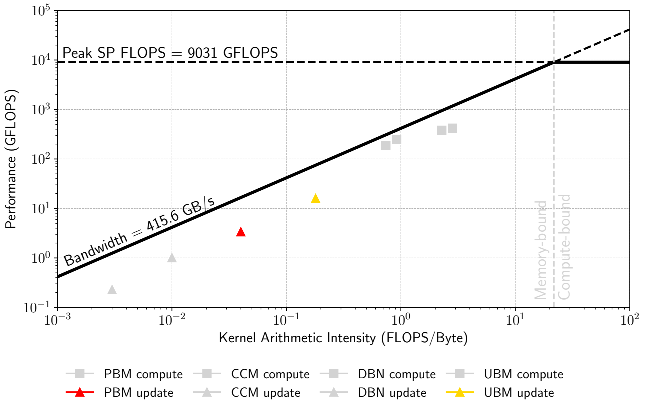
**Roofline model** of MassiveClicks for **PBM**, **CCM**, **DBN**, and **UBM** on a single **NVIDIA RTX A4000** GPU.

# Kernel Performance



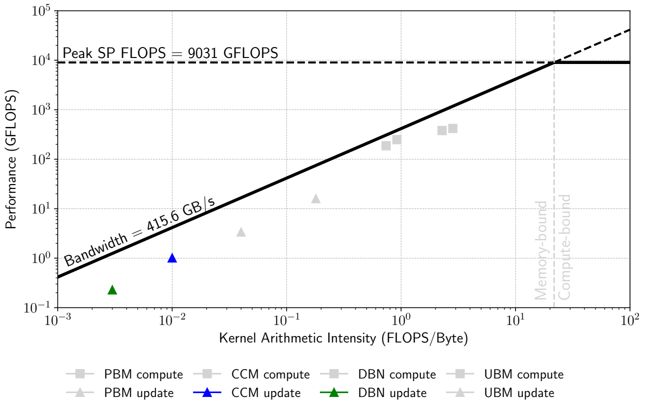
**Roofline model** of MassiveClicks for **PBM**, **CCM**, **DBN**, and **UBM** on a single **NVIDIA RTX A4000** GPU.

# Kernel Performance



**Roofline model** of MassiveClicks for **PBM**, **CCM**, **DBN**, and **UBM** on a single **NVIDIA RTX A4000** GPU.

## Kernel Performance



**Roofline model** of MassiveClicks for **PBM**, **CCM**, **DBN**, and **UBM** on a single **NVIDIA RTX A4000** GPU.

## Conclusion and Future Directions

- **In Summary:** We introduced MassiveClicks, a tool for training EM-based click models on heterogeneous multi-GPU/multi-node setups, offering a significant performance improvement over existing solutions.
- **Future Work:**
  - Exploring hybrid training methods.
  - Transition to HIP for broader compatibility.
- **Repository:** Find the code and documentation for MassiveClicks at *github.com/skip-th/MassiveClicks* or the QR-code.

## Hybrid training

Automatic

- Compute optimal data distribution.

Manual

- User chooses desired distribution.

## Compatibility

The heterogeneity only extends to CUDA GPUs.

- AMD GPUs are not supported.
- Currently converting codebase to HIP using HIPify[1].

[1] `https://github.com/ROCm-Developer-Tools/HIPIFY`

## Position-based Model (PBM)

$$P(C_d = 1) = P(E_d = 1) \cdot P(A_d = 1)$$

A user **clicks** a search result if, and only if, they **examined** the result and were **attracted** to it.

| | |
|---|---|
| $\gamma_r$ | – rank-dependent *examination* parameters. |
| $\alpha_{qd}$ | – query-dependent *attractiveness* parameter. |

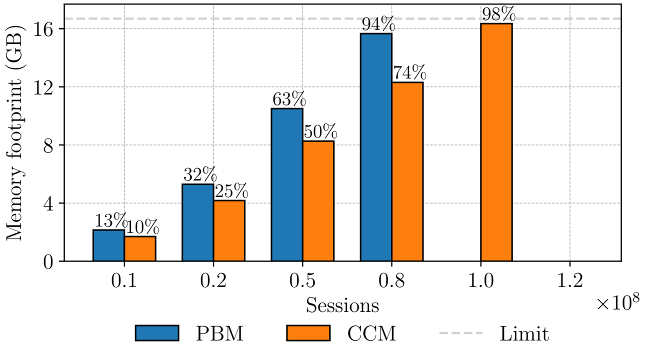
## Click Chain Model (CCM)

$$P(C_d = 1) = P(E_d = 1) \cdot P(A_d = 1)$$

A user **clicks** a search result if, and only if, they **examined** the result and were **attracted** to it.

| | |
|---|---|
| $\tau_1, \tau_2, \tau_3$ | – document-dependent *continuation* parameters. |
| $\alpha_{q_r d}$ | – query-dependent *relevance* parameter. |

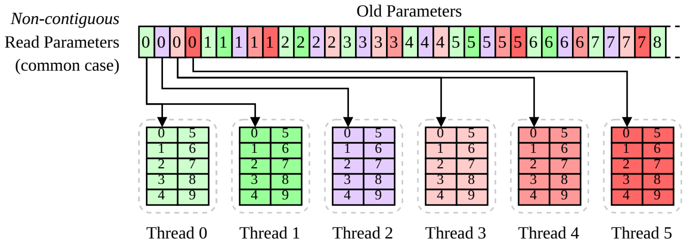
## Memory Footprint



**Memory footprint** of MassiveClicks for **PBM** and **CCM** on a single **NVIDIA RTX A4000** GPU.

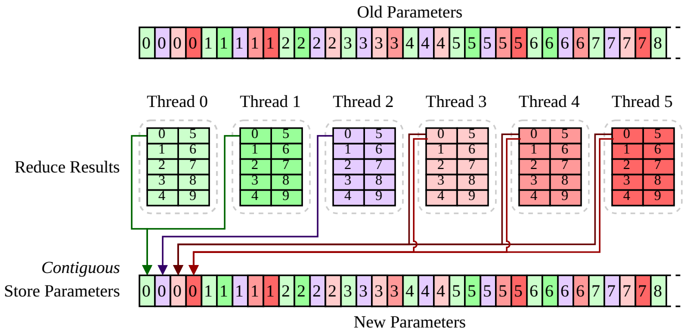
# New Memory Access Pattern

**Compute** — Threads *read* parameters from *previous* iteration.



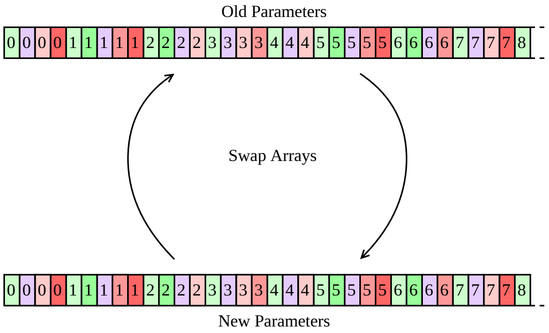
# New Memory Access Pattern

**Update** — Threads *reduce* parameters to *current* iteration.



## New Memory Access Pattern

**Update** — Threads *swap* old and new parameters.


Old Parameters
0 0 0 0 1 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 7 7 7 7 8
Swap Arrays
0 0 0 0 1 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 7 7 7 7 8
New Parameters