HIGH-PERFORMANCE GENOME STUDIES

Lucas Beyer
Diego Fabregat-Traver and Prof. Paolo Bientinesi

RWTH Aachen University
19 June 2012, SIAM Conference on Applied Linear Algebra, Valencia, Spain

Thanks to the AICES HPAC group and DFG grant GSC111

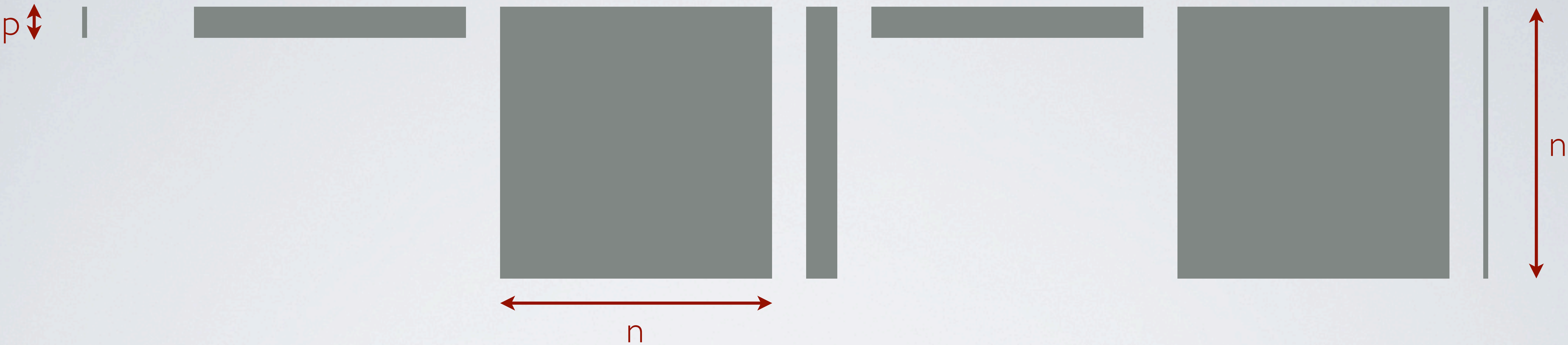
OUTLINE

- Problem description
- CPU-only algorithm
- Leveraging the GPU
- Results and conclusion

OUTLINE

- Problem description
- CPU-only algorithm
- Leveraging the GPU
- Results and conclusion

$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$

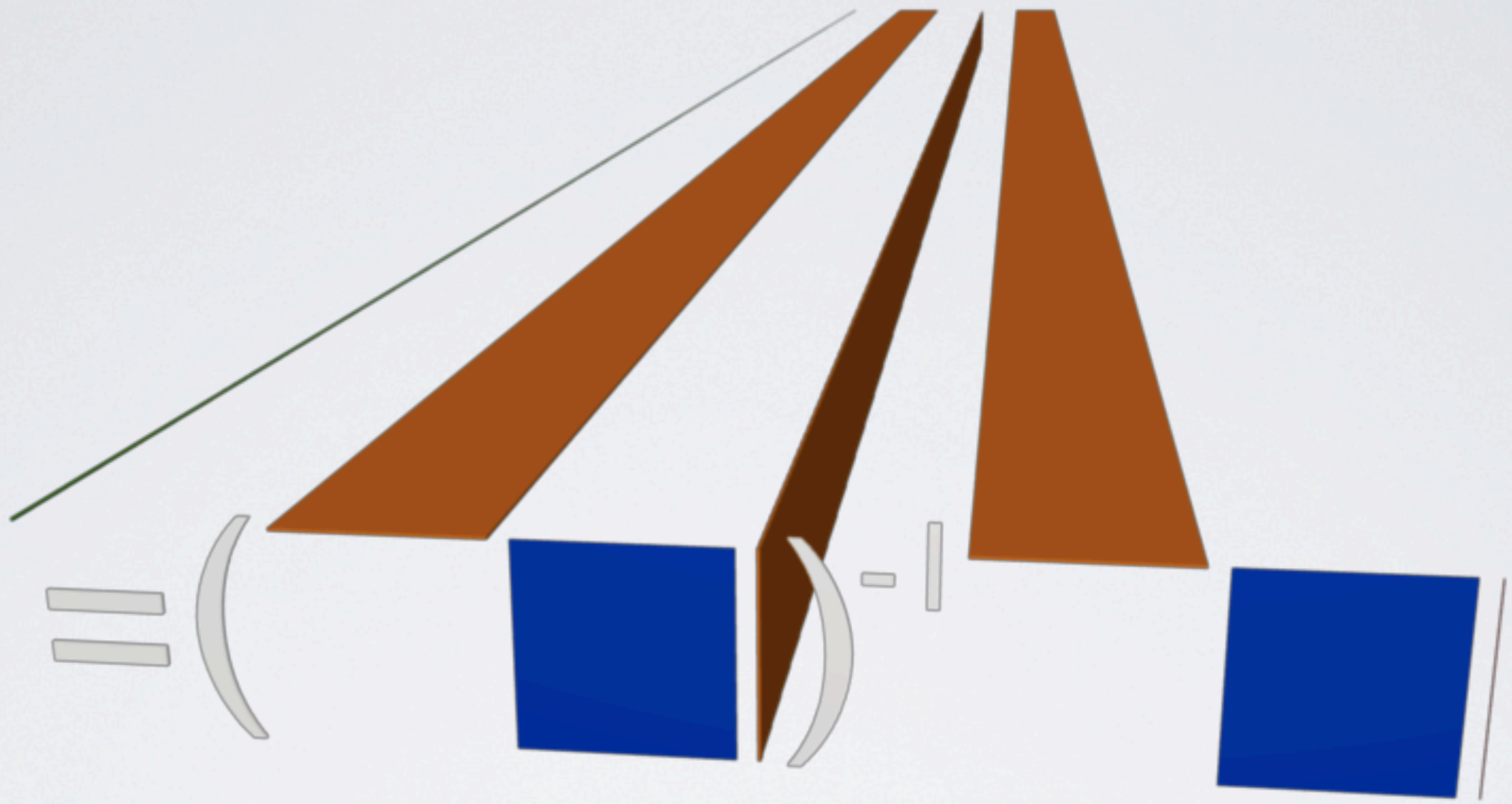


GENOME-WIDE ASSOCIATION STUDIES

lots of GLS because i = 0..millions

input	$y \in \mathbb{R}^n$	observations (phenotype)
	$X_i \in \mathbb{R}^{n \times p}$	genome measurements/covariates
	$M \in \mathbb{R}^{n \times n}$	observation dependencies
output	$r_i \in \mathbb{R}^p$	relations between phenotype and genome variations

$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$



THE NUMBERS

# DNA fragments (nucleotides)	m ~ 48 – 250 000 000
# samples	n ~ 10 000
# covariates	p = 20

$\Rightarrow$

$y \in \mathbb{R}^{n}$	80 MB
$M \in \mathbb{R}^{n \times n}$	800 MB
$r \in \mathbb{R}^{p \times m}$	7-40 GB
$X \in \mathbb{R}^{n \times p \times m}$	72 TB – 373 PB

OUTLINE

- Problem description
- CPU-only algorithm
- Leveraging the GPU
- Results and conclusion

BASIC ALGORITHM

$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$

BASIC ALGORITHM

$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$

Cholesky once during initialization

$$LL^T := M$$

BASIC ALGORITHM

$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$

Cholesky once during initialization

$$LL^T := M$$

$$r_i \leftarrow (X_i^T L^{-T} L^{-1} X_i)^{-1} X_i^T L^{-T} L^{-1} y$$

BASIC ALGORITHM

$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$

Cholesky once during initialization

$$LL^T := M$$

$$r_i \leftarrow (X_i^T L^{-T} L^{-1} X_i)^{-1} X_i^T L^{-T} L^{-1} y$$

$$r_i \leftarrow ((L^{-1} X_i)^T L^{-1} X_i)^{-1} L^{-1} X_i L^{-1} y$$

BASIC ALGORITHM

$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$

Cholesky once during initialization

$$LL^T := M$$

$$r_i \leftarrow (X_i^T L^{-T} L^{-1} X_i)^{-1} X_i^T L^{-T} L^{-1} y$$

$$r_i \leftarrow ((L^{-1} X_i)^T L^{-1} X_i)^{-1} L^{-1} X_i L^{-1} y$$

One trsm per iteration step i

$$\hat{X}_i := L^{-1} X_i$$

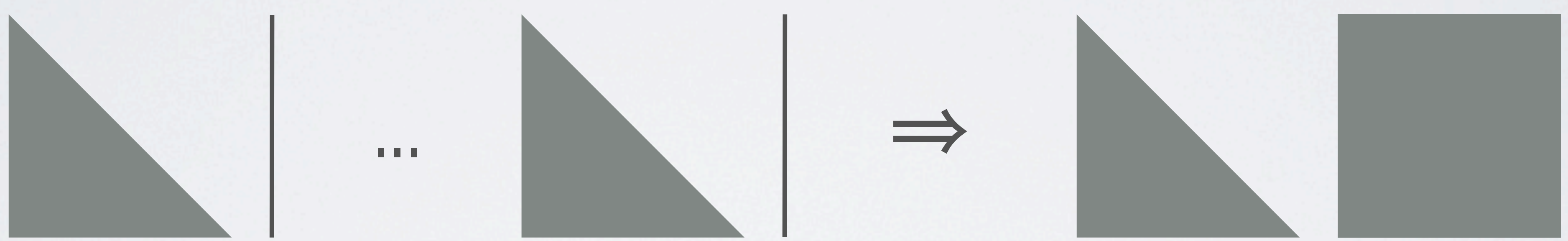
$$r_i \leftarrow (\hat{X}_i^T \hat{X}_i)^{-1} \hat{X}_i L^{-1} y$$

OPTIMIZATIONS

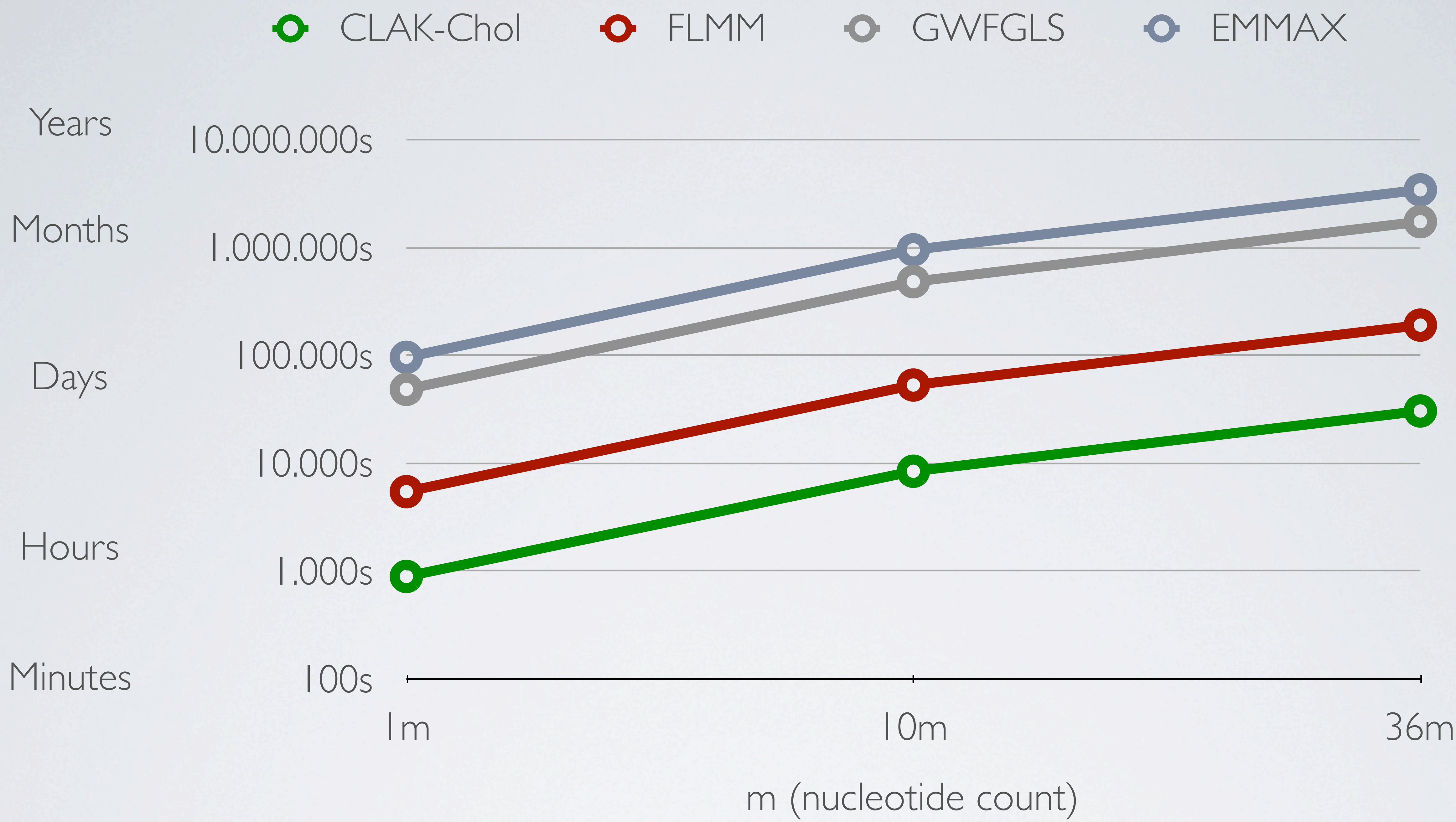
- Blocking in i
  - many small trsms vs. one big trsm

OPTIMIZATIONS

- Blocking in i
  - many small trsms vs. one big trsm



- Out-of-core algorithm
  - read block b_{+1} while computing block b
  - double-buffering technique necessary



PERFORMANCE

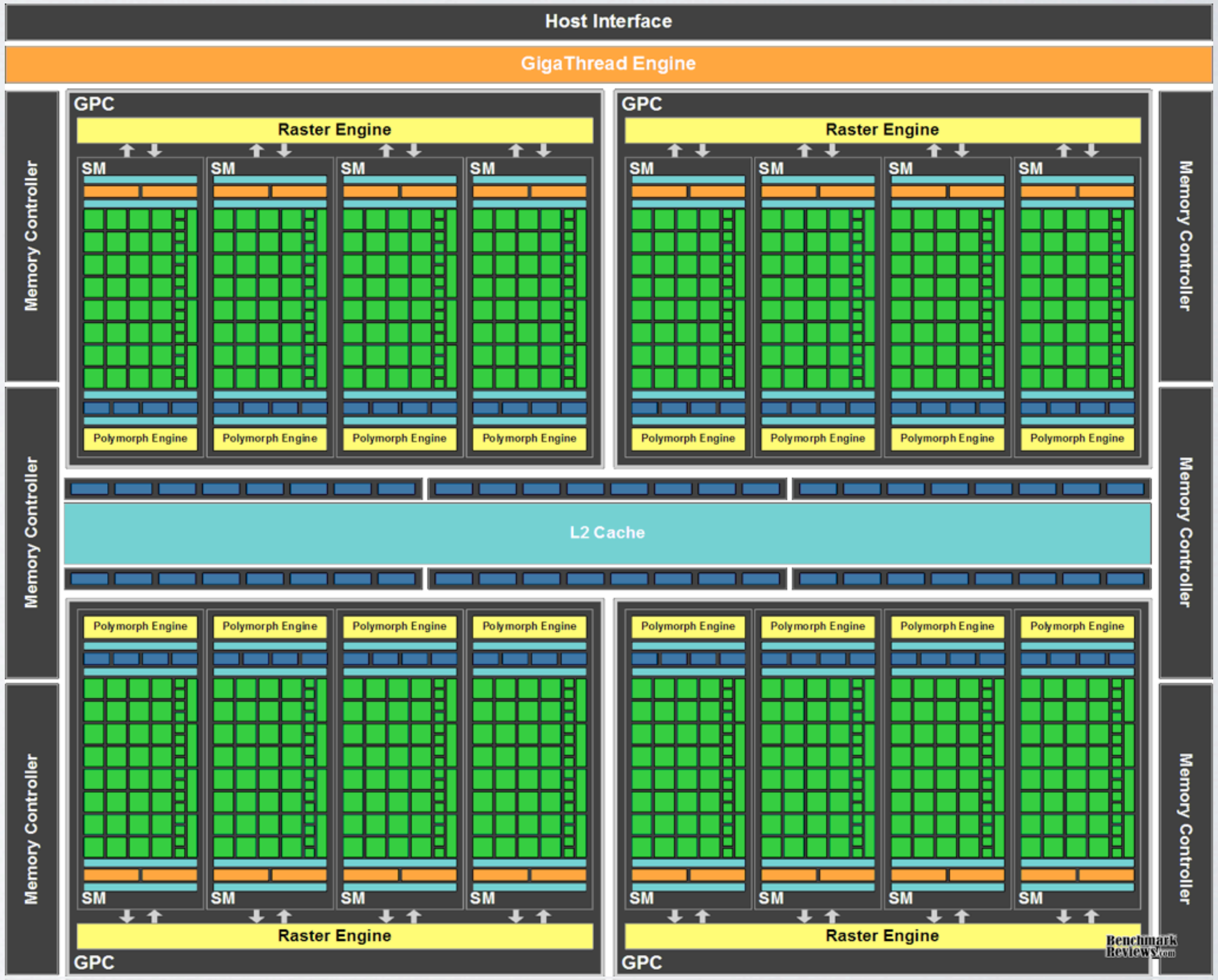
From years/months down to weeks/days

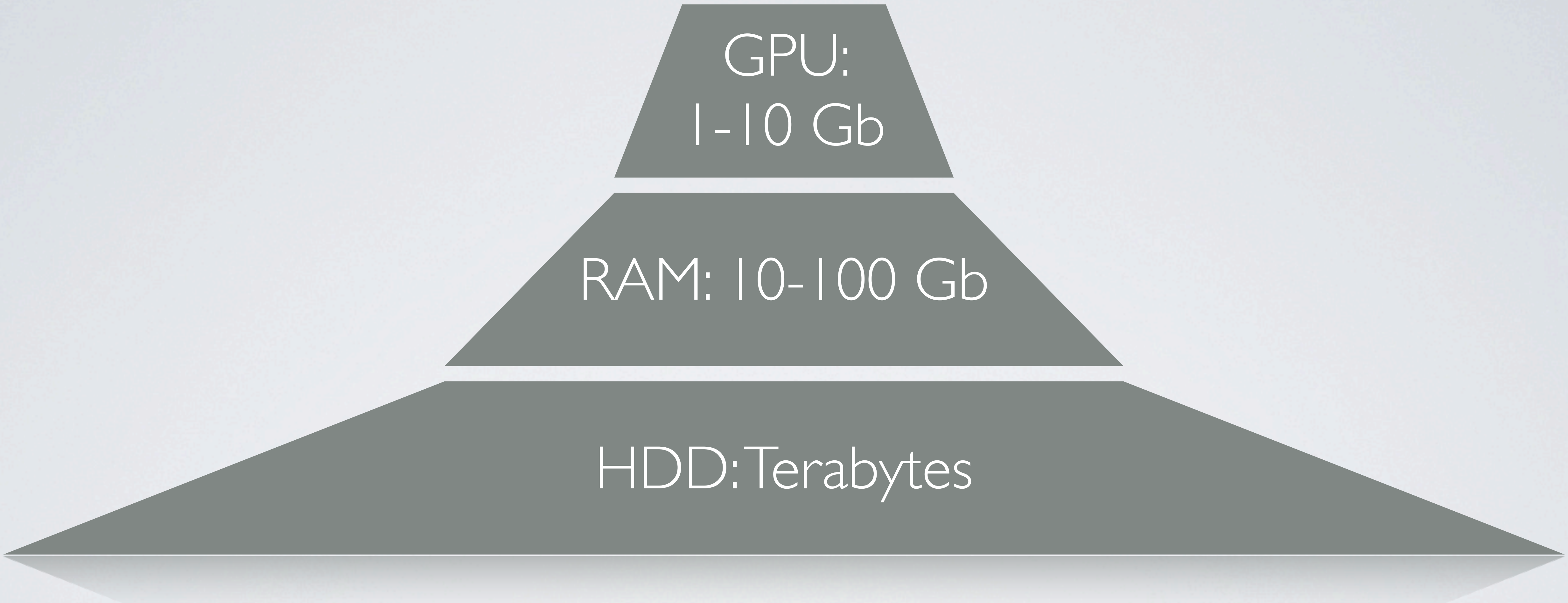
OUTLINE

- Problem description
- CPU-only algorithm
- Leveraging the GPU
- Results and conclusion

CAN GPUs HELP GO FURTHER?

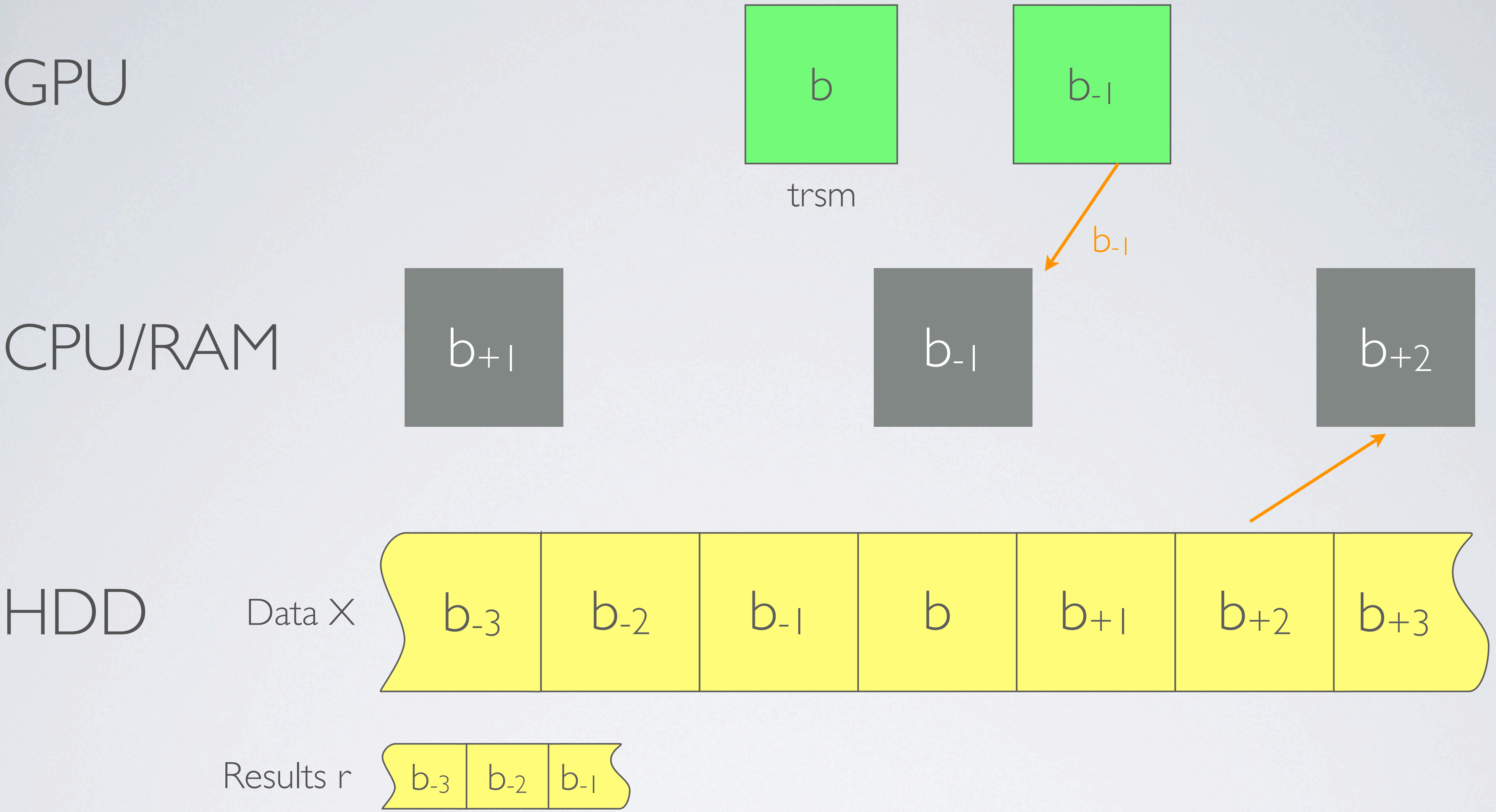
- trsm takes 90-95% of time
  - compute on the GPU
- while GPU computes:
  - CPU computations
  - CPU ⇄ GPU transfers
- our cluster: nVidia Fermi ☞





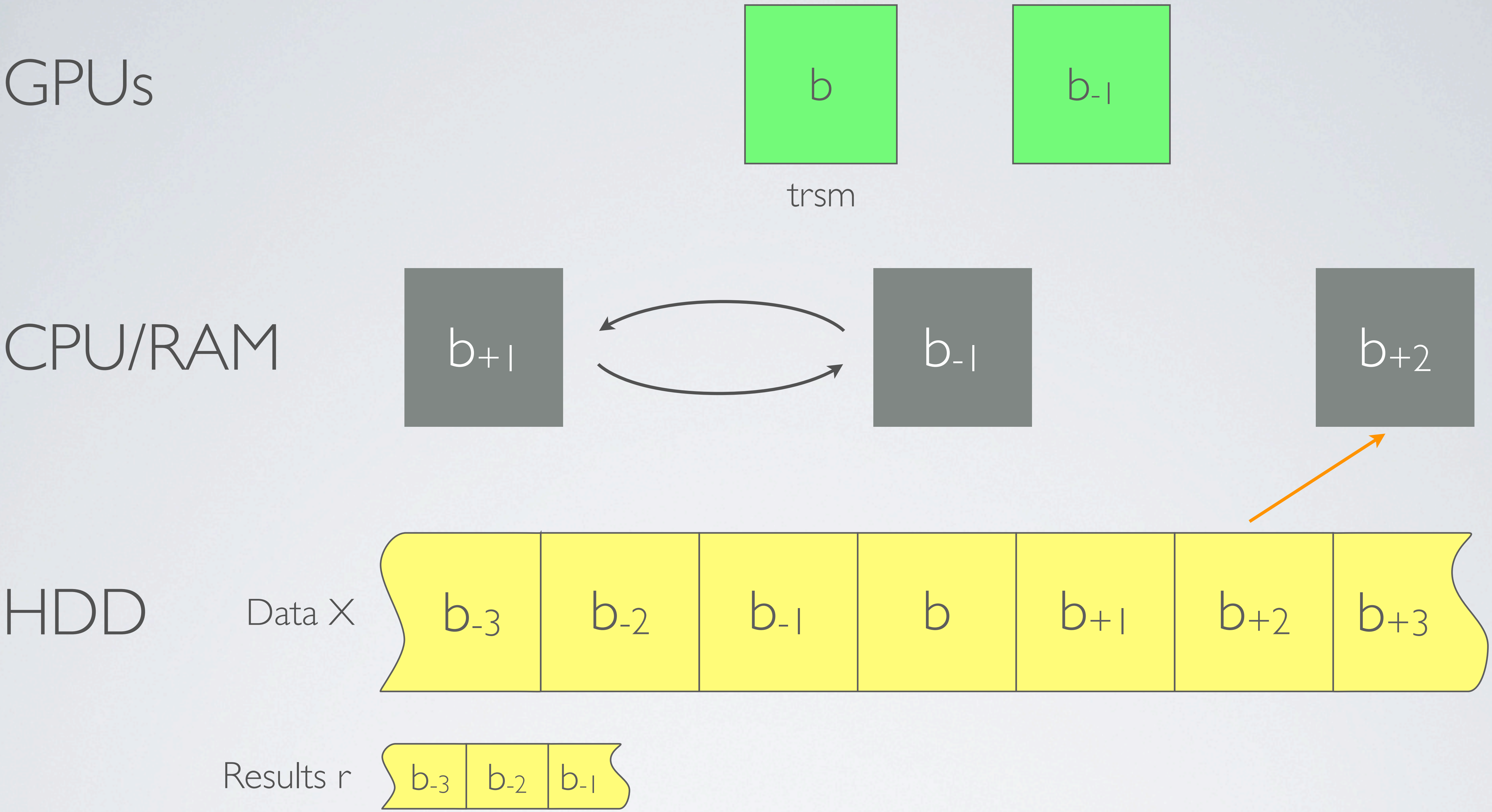
MEMORY PYRAMID

Need for streaming computation
Need for two levels of double-buffering



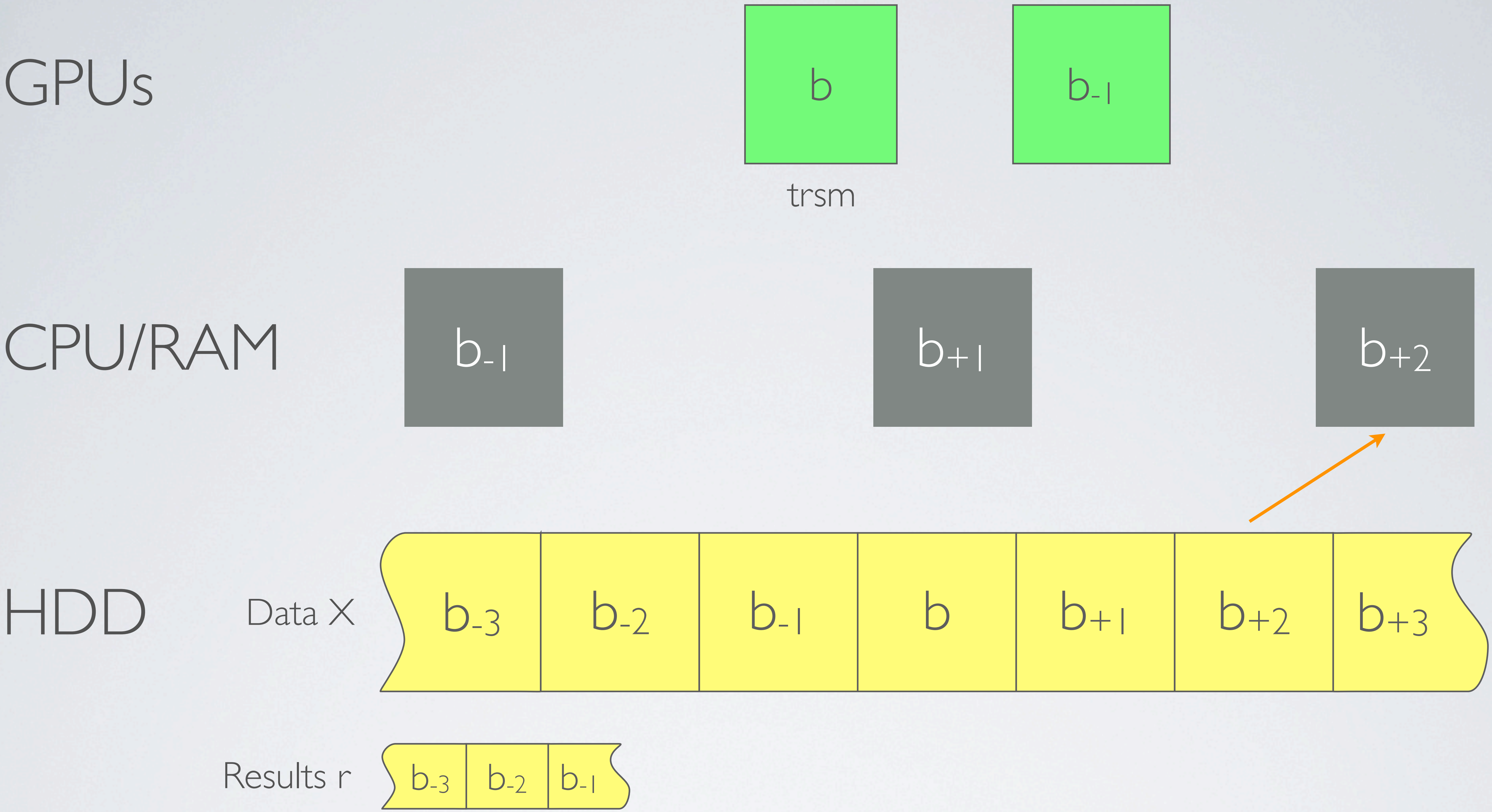
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(1) Retrieve previous results from GPU,
start reading second-next block from HDD



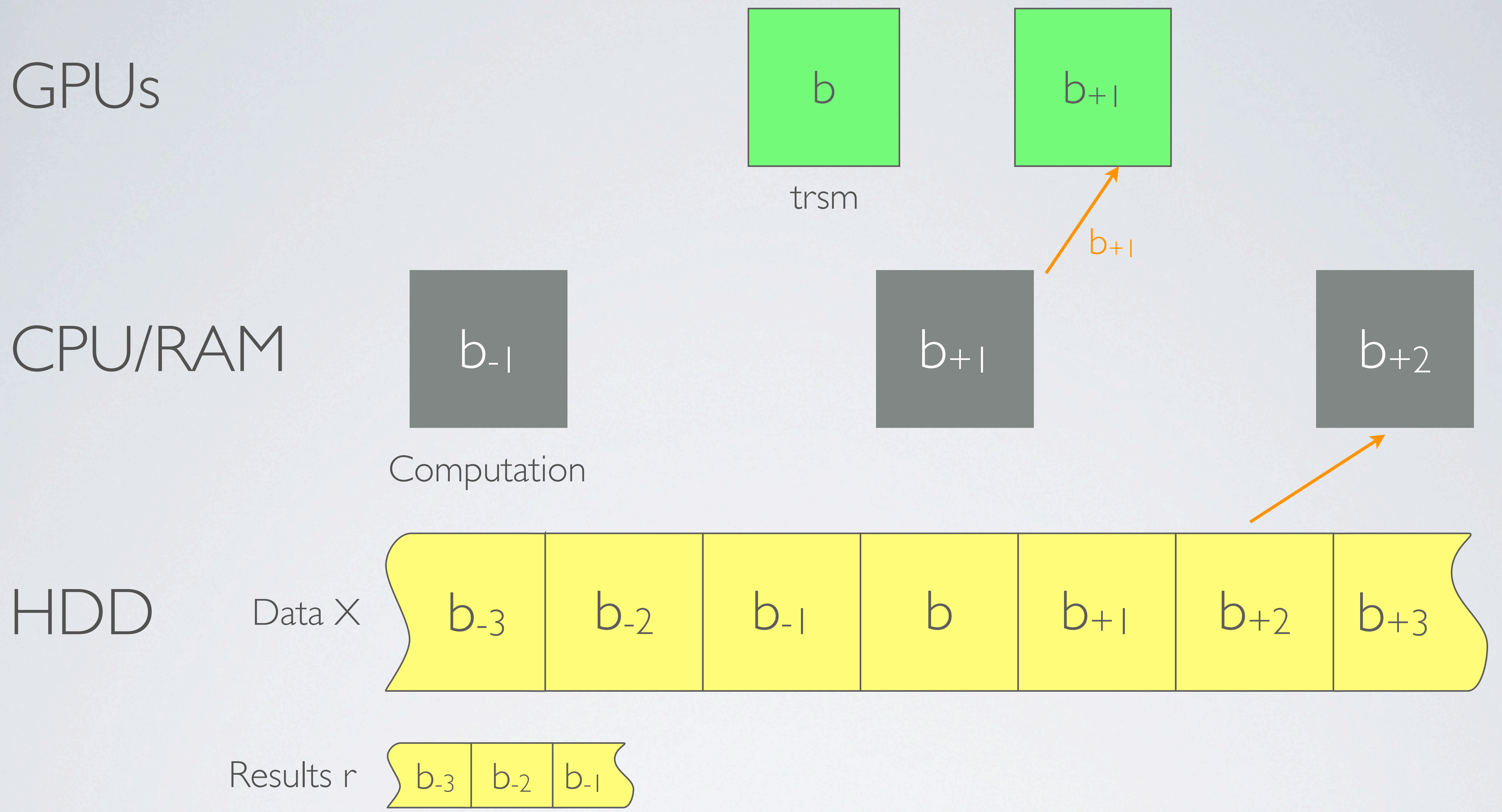
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(2) Buffer switch (no copying)



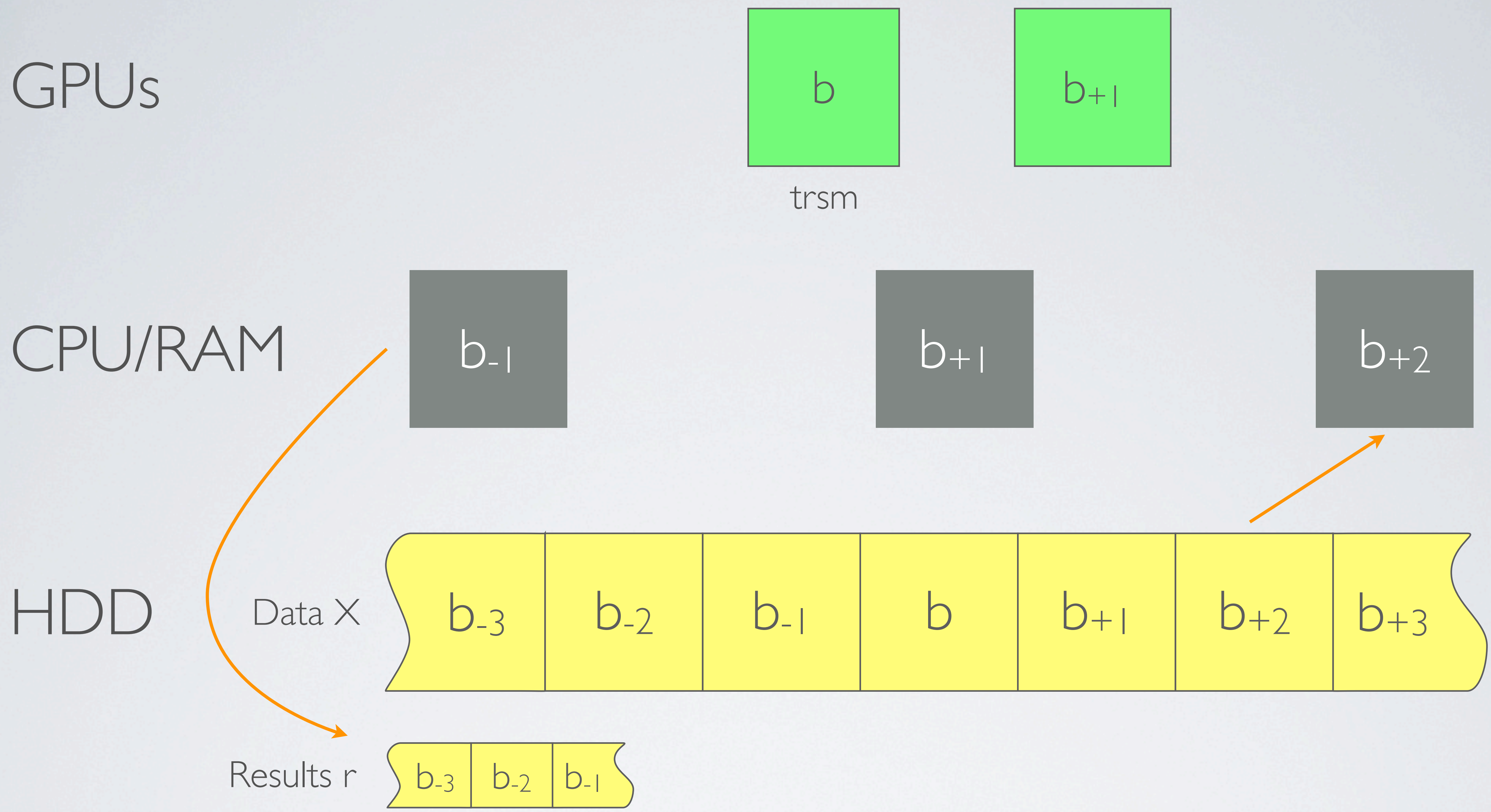
2-LEVEL TRIPLE-DOUBLE-BUFFERING

Buffers switched



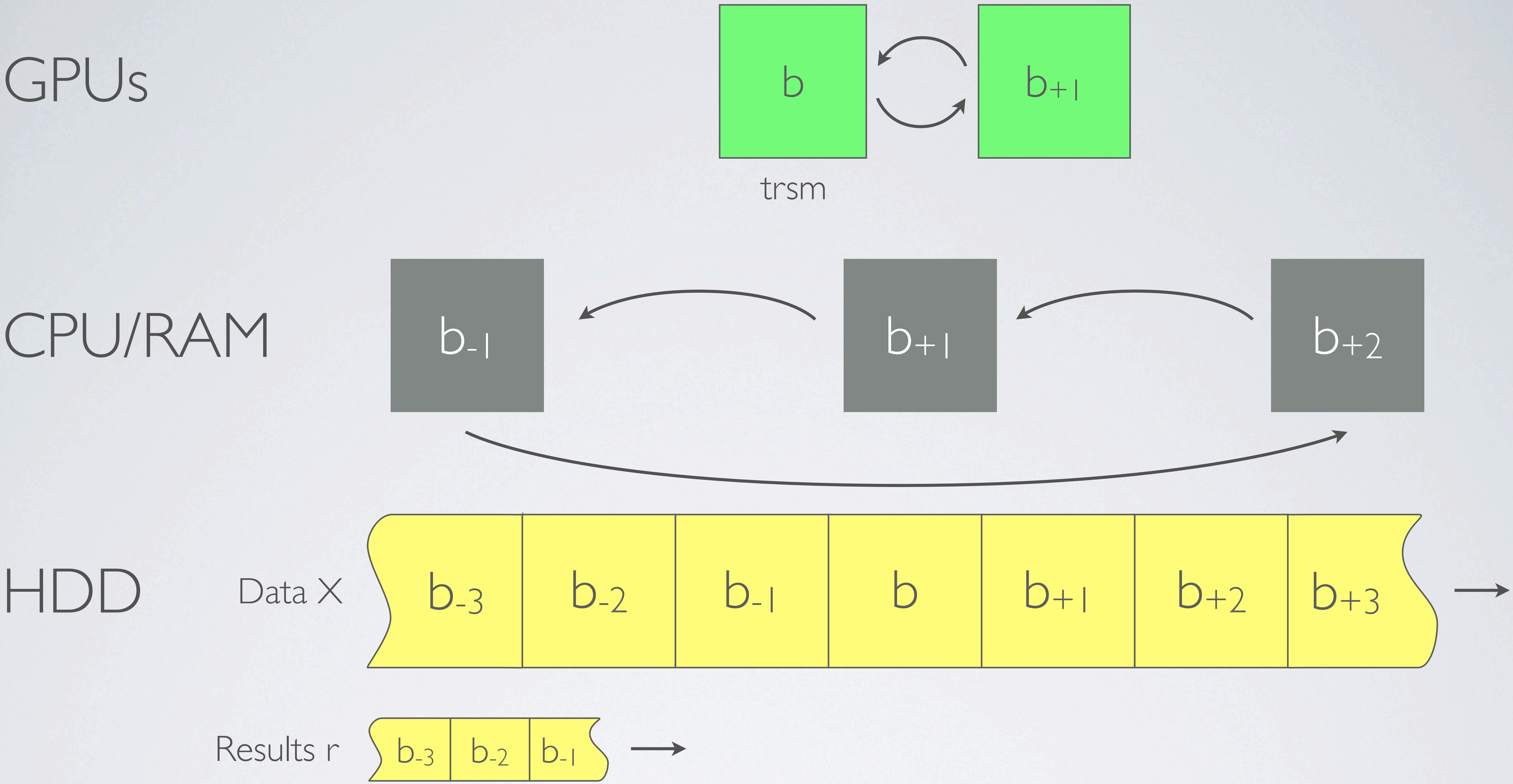
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(3) Send next block to GPU, start CPU computation



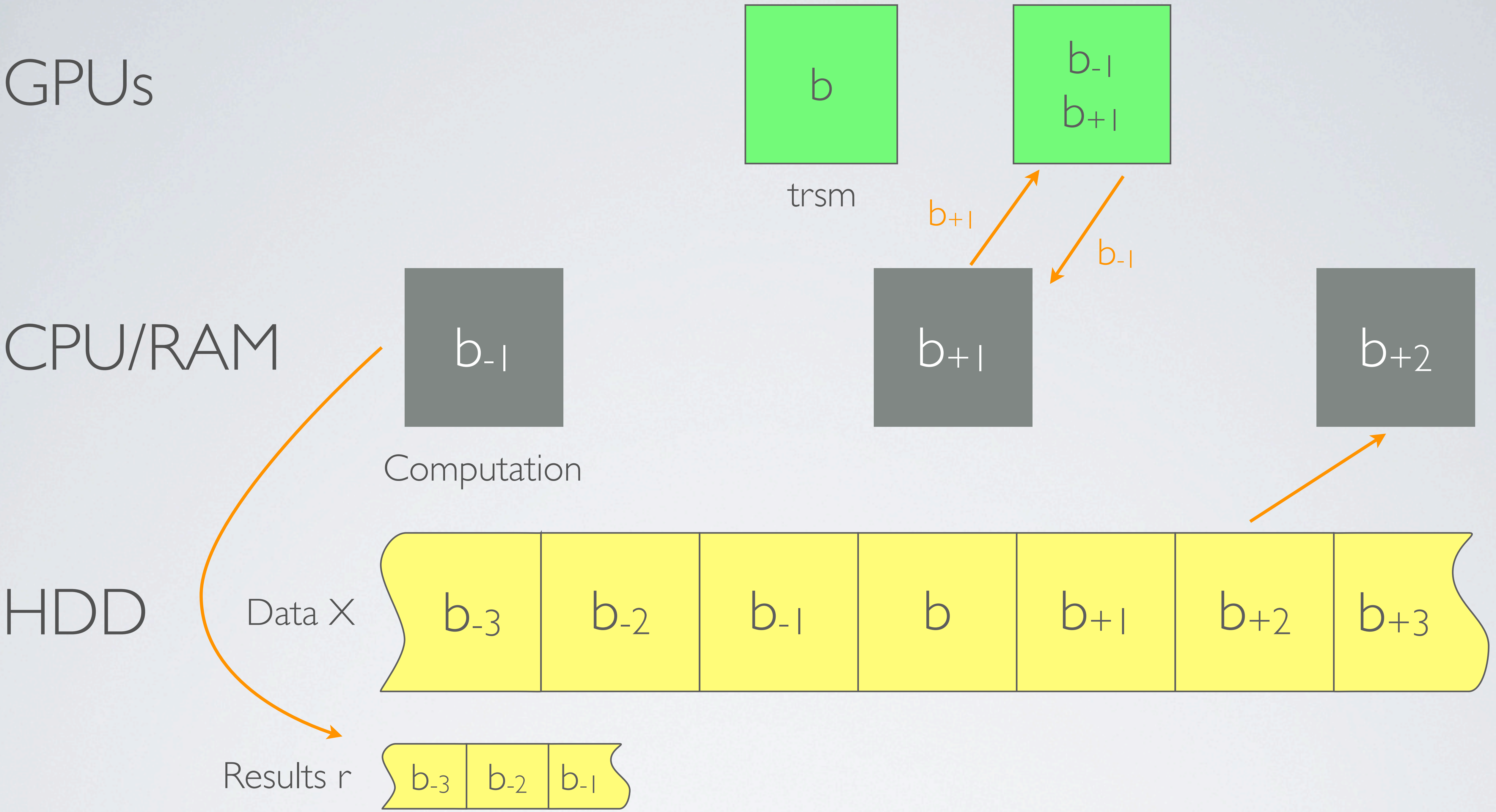
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(4) Write results to disk (fast because small)



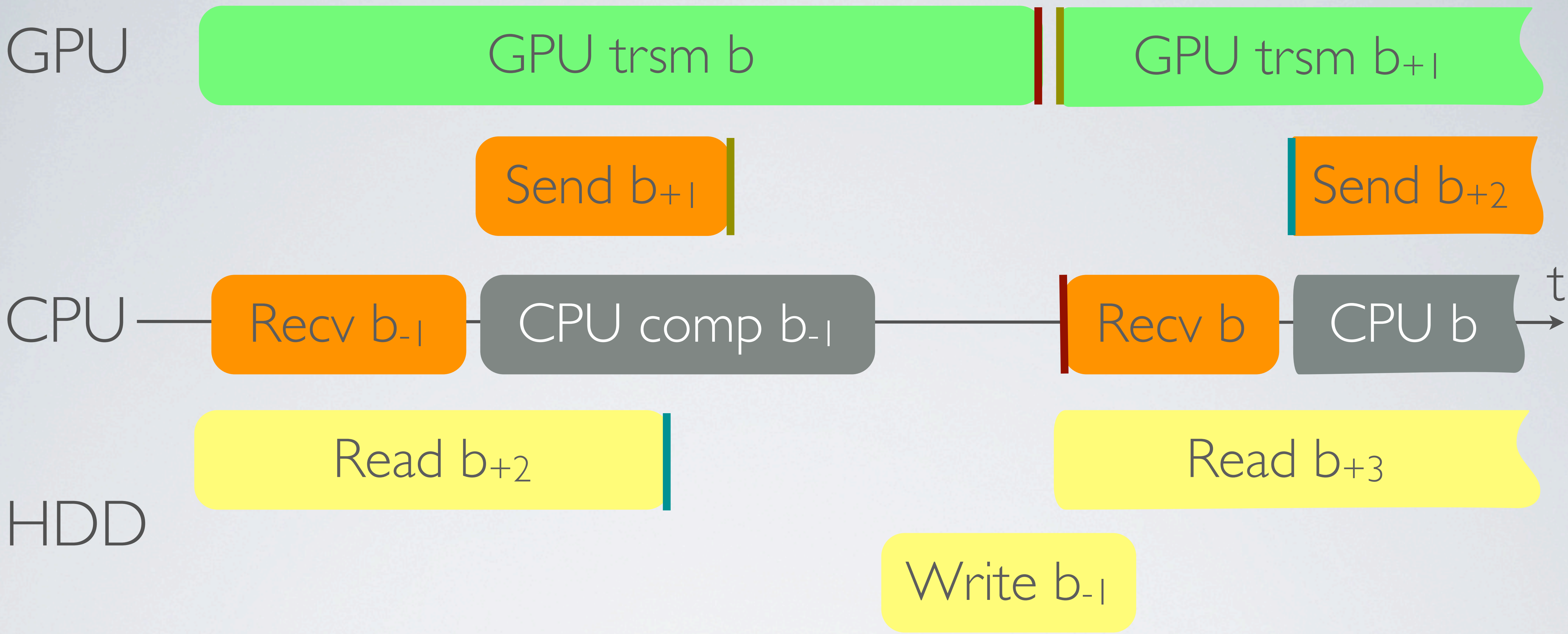
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(5) Buffer switch (no copying)



2-LEVEL TRIPLE-DOUBLE-BUFFERING

One full iteration



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

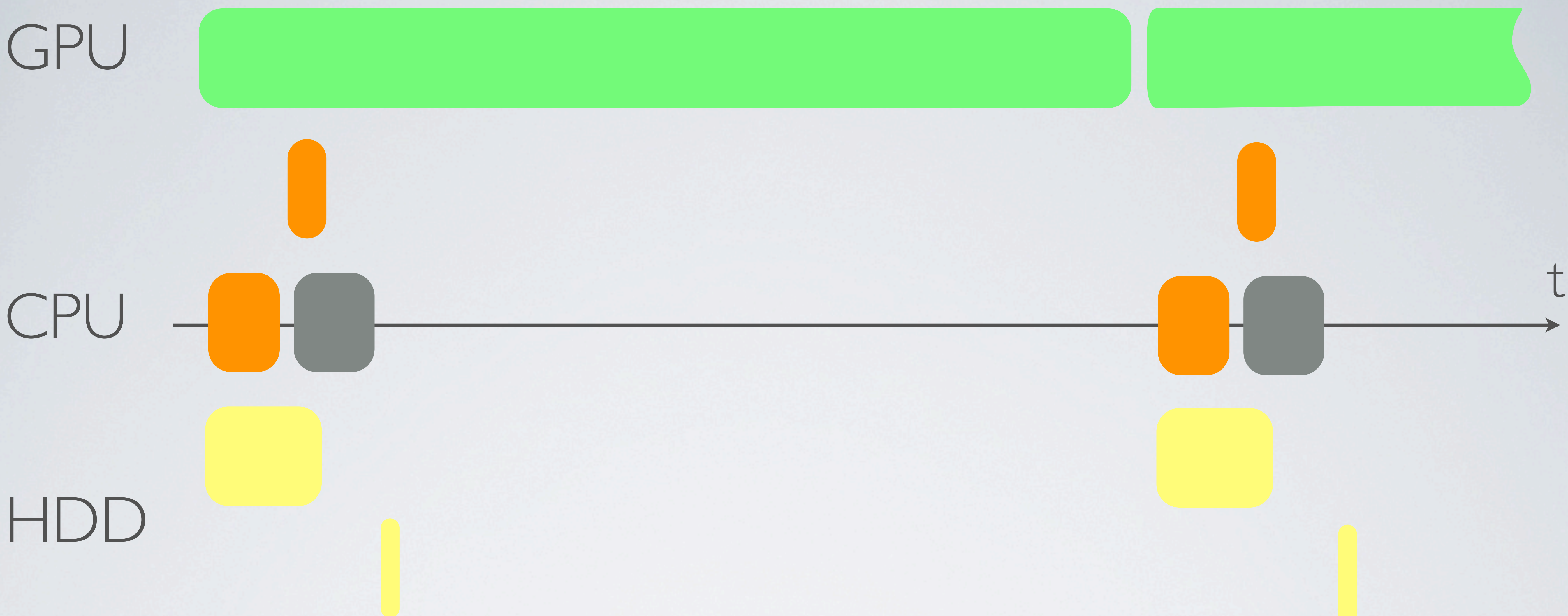
CPU ⇄ GPU transfer

HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies



TIMELINE, TO SCALE

problem sizes: n=10k, m=100k, block=10k

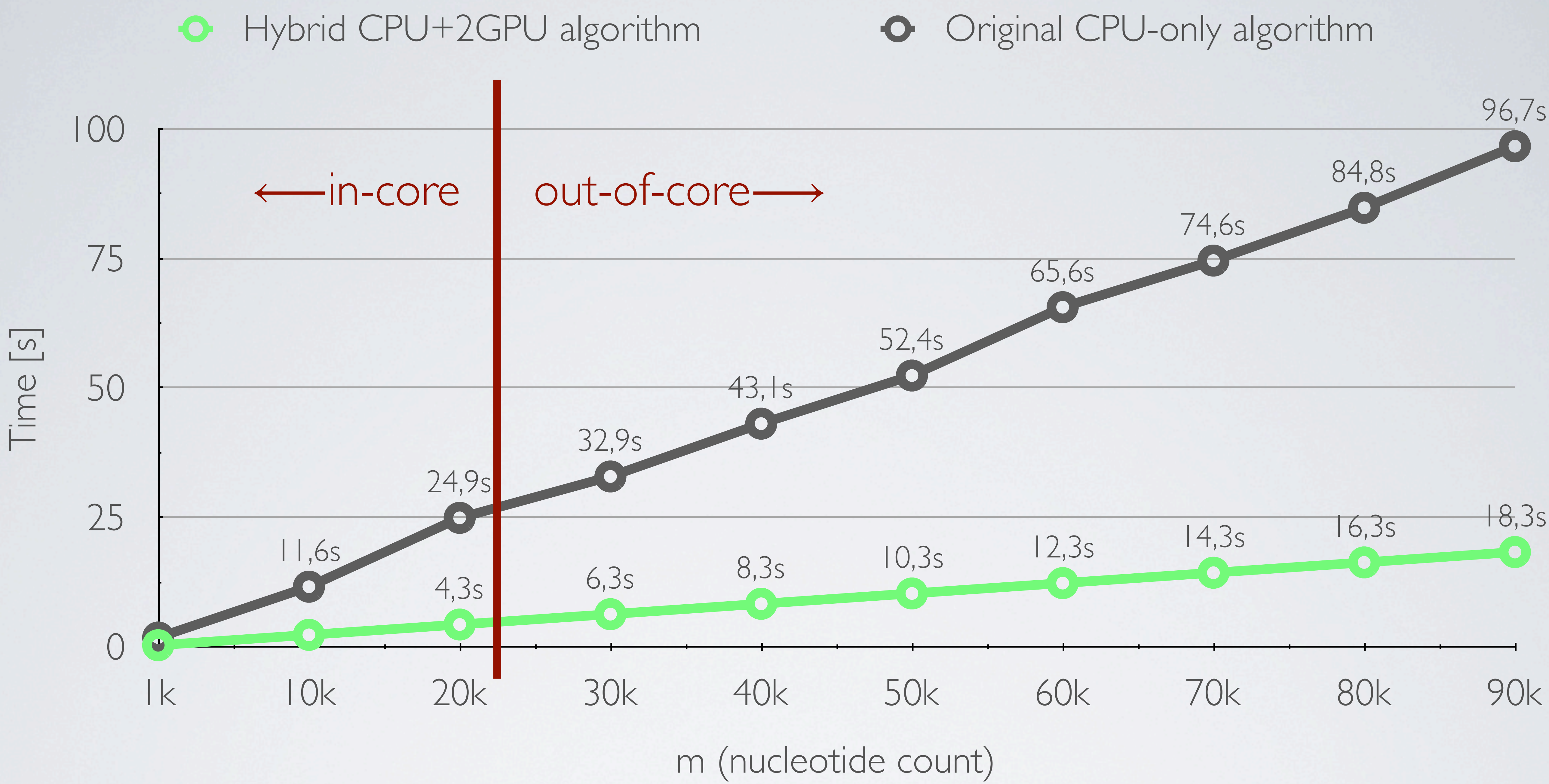
GPU: **2x** nVidia Quadro 6000 (Fermi, 515 GFlops **each**, 6GB memory) = 10.000$
CPU: 2x Intel Xeon X5650 (6cores, 128 GFlops, 24GB memory) = 2000$

- CPU ⇄ GPU transfer
- HDD ⇄ CPU transfer
- GPU computation
- CPU computation

Blas: Intel MKL 10.2
Compiler: icc 12.1

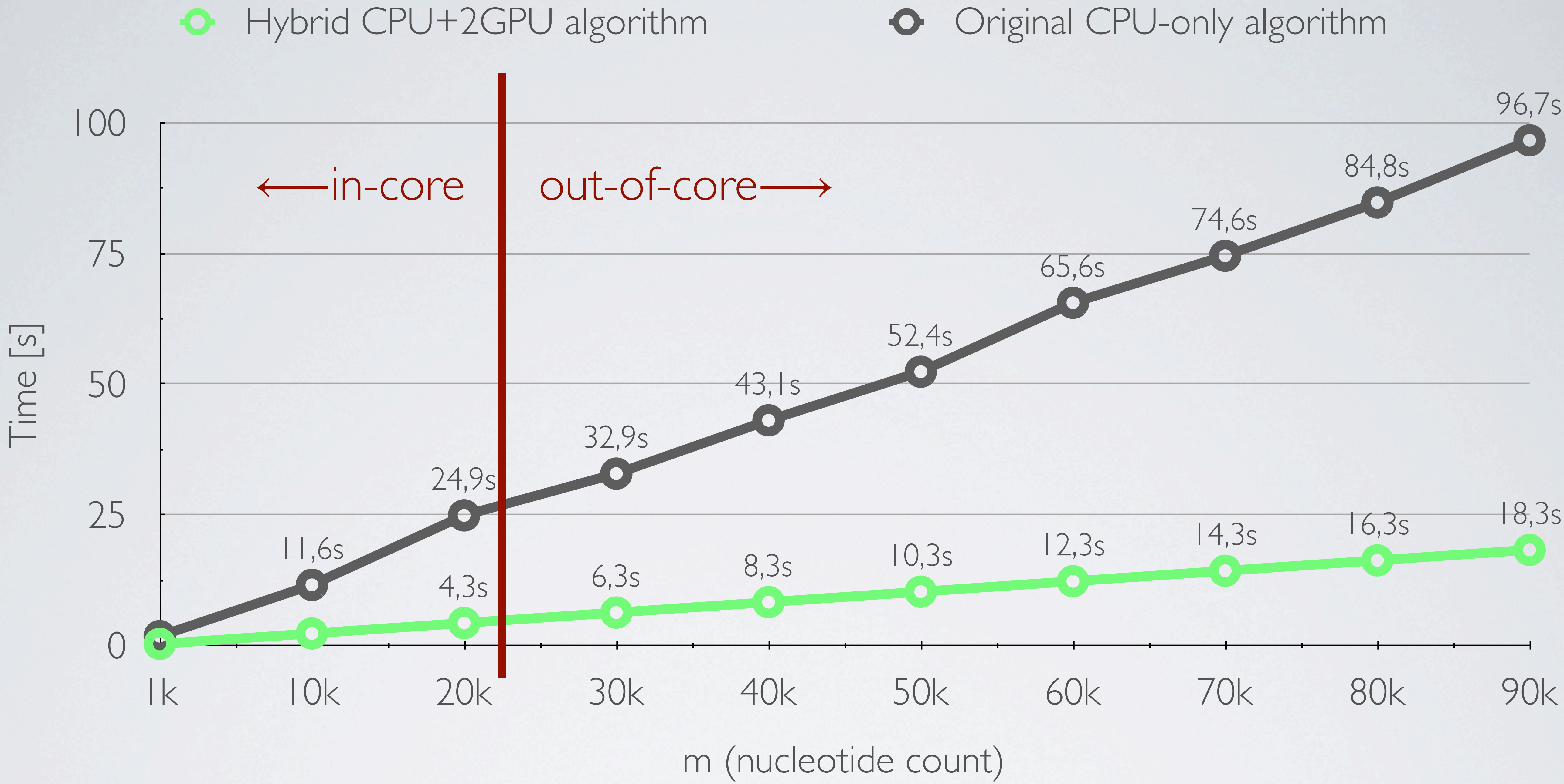
OUTLINE

- Problem description
- CPU-only algorithm
- Leveraging the GPU
- Results and conclusion



PERFORMANCE

sustained in-core performance when out-of-core
extrapolated: 13h/70h vs 2.5h/13h

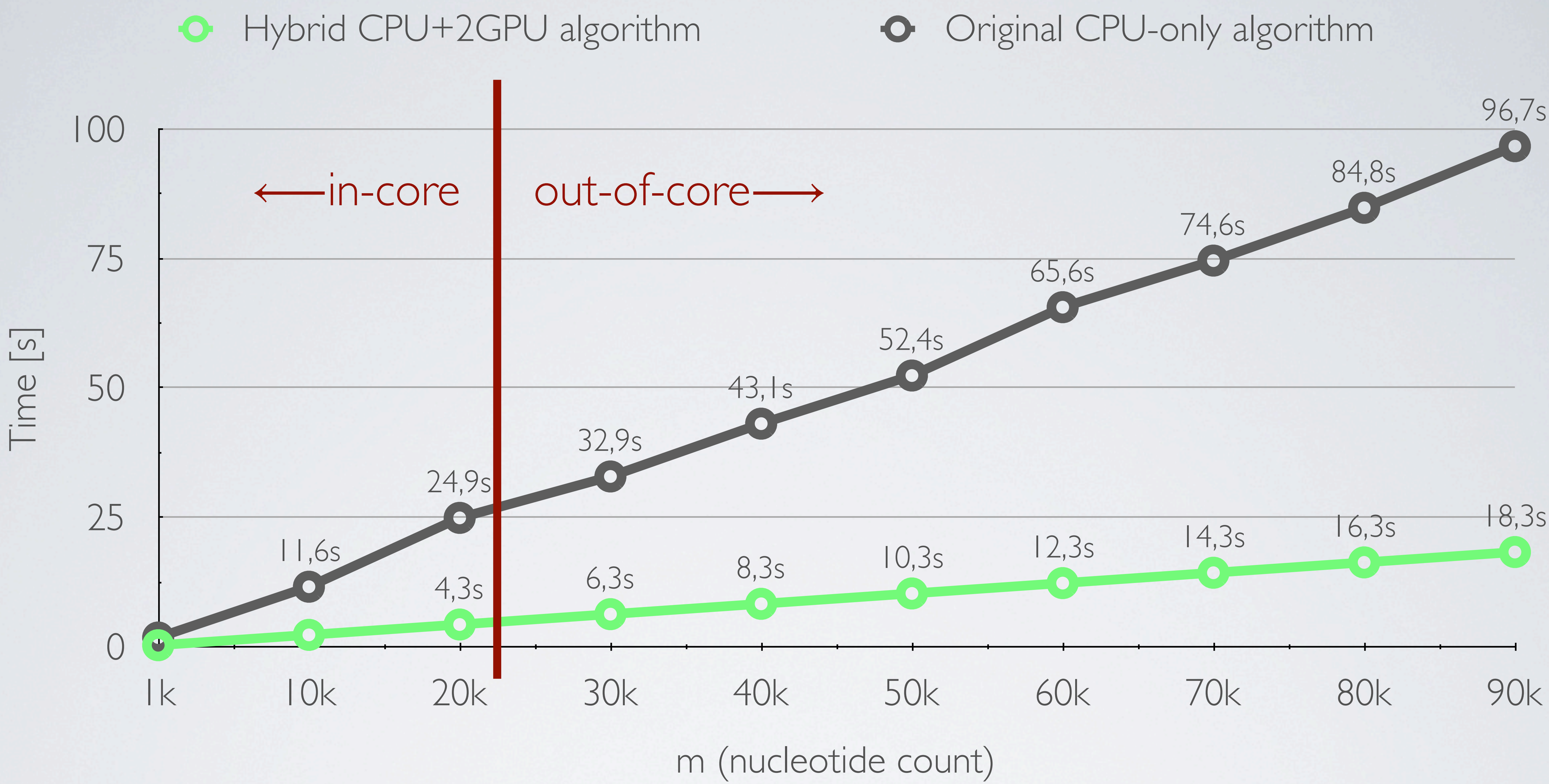


PERFORMANCE

5.2x speedup using 2 GPU, 10x using 4 GPUs
from days to hours (when m is millions)

CONCLUSION

- Don't replace the CPU by GPU
  - Combine them
- Hide data transfer latency by overlapping with computation
  - Double/triple-buffering
  - GPU never stops computing
- GPUs order of magnitude faster?
  - V. Volkov (http://www.cs.berkeley.edu/~volkov/)
  - Victor W. Lee et al. («Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU», 2010)



QUESTIONS?

beyer@aices.rwth-aachen.de

FUTURE WORK

- Solution for L too big for GPU memory
- Apply similar technique to similar problems
- Extension to multiple phenotypes (y)