GWAS ON GPUS:

Streaming Data from HDD for Sustained Performance

Lucas Beyer

Diego Fabregat-Traver and Prof. Paolo Bientinesi

RWTH Aachen University

29 August 2013, EuroPar 2013, Aachen, Germany

Thanks to the AICES HPAC group and DFG grant GSC111

OUTLINE

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

OUTLINE

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

SINGLE NUCLEOTIDE POLYMORPHISM

nucleotide differs between two individuals of a species

link to traits, diseases?

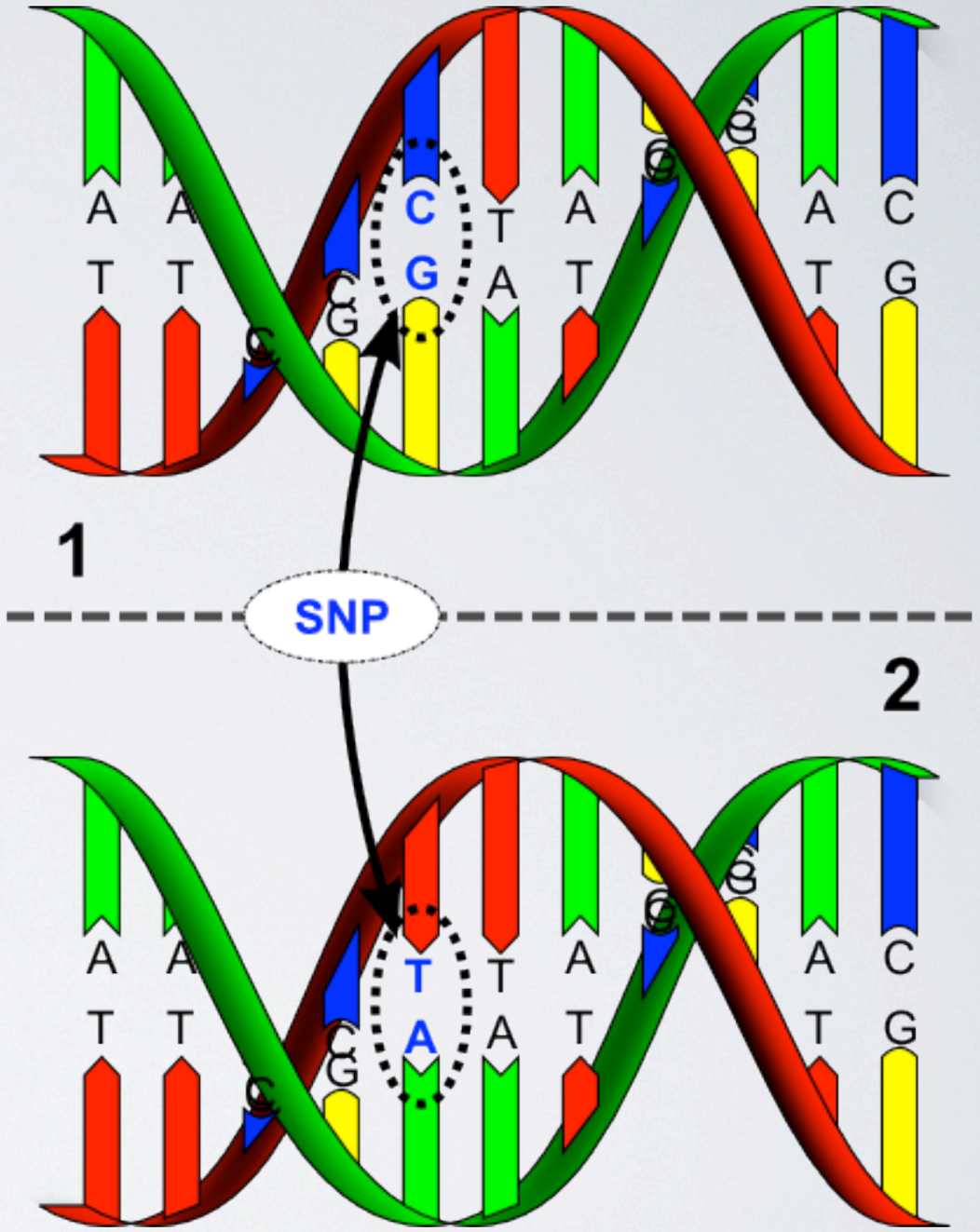


Image credit: David Hall

GWAS

- Human Genome project HUGO (2004)
- Genome-wide association studies (GWAS)
  - Find correlations between SNPs and traits (diseases)

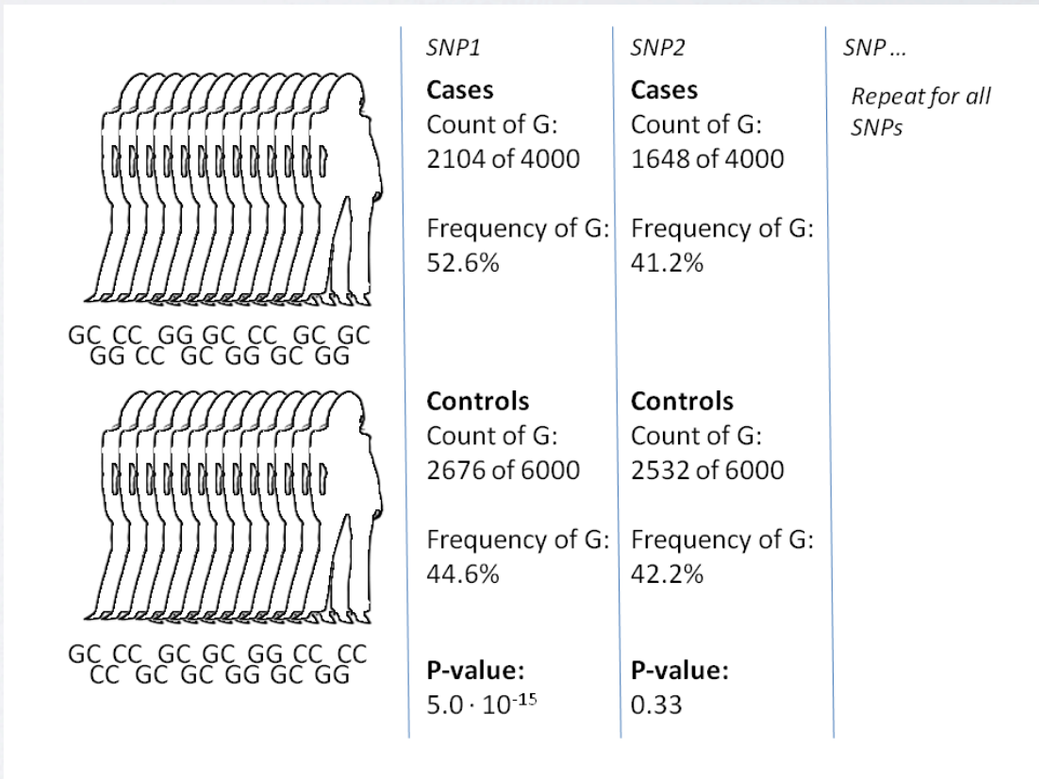
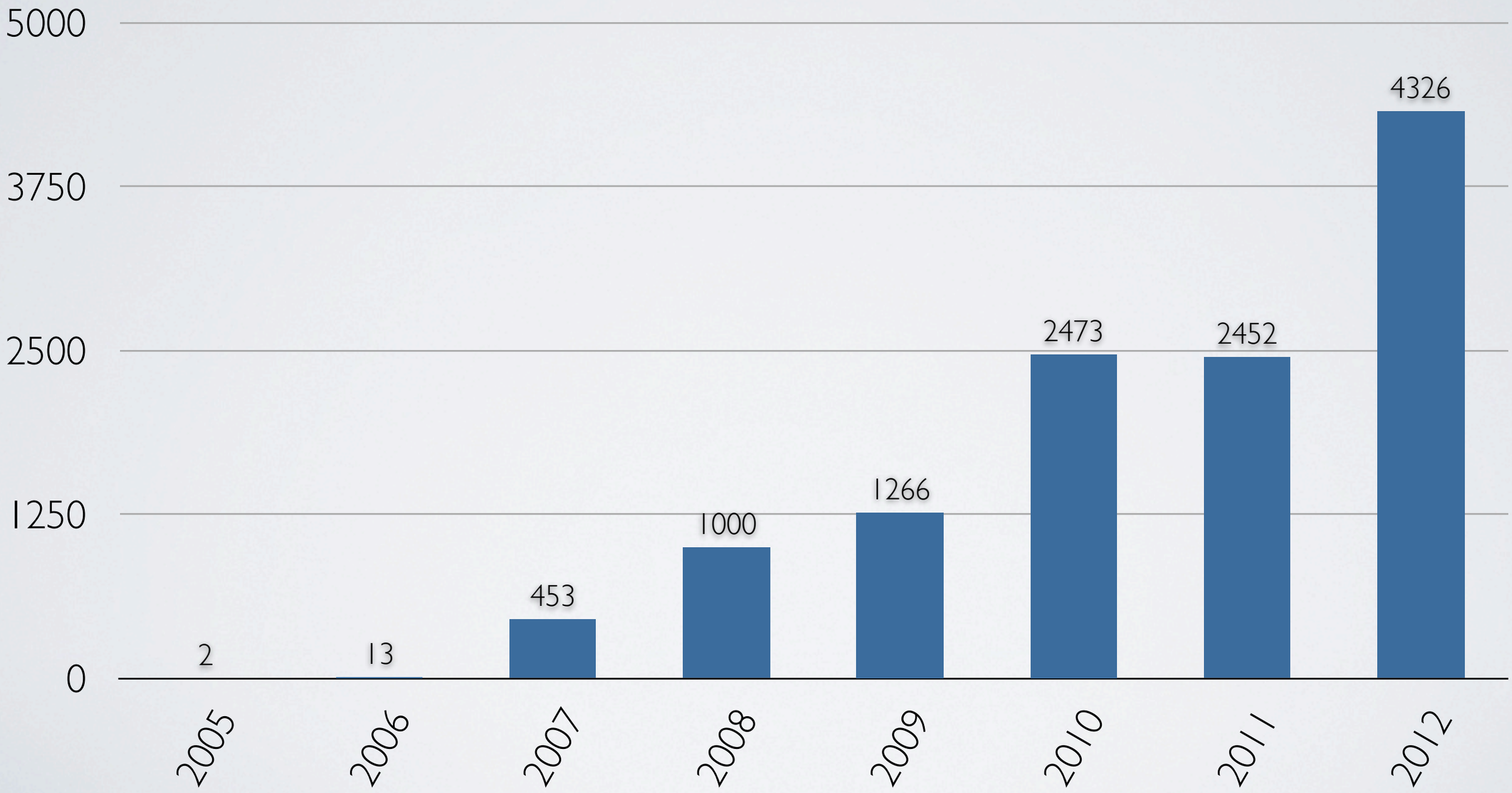


Image credit: Lasse Folkersen

GWAS STATS

of GWAS carried out each year



OUTLINE

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

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

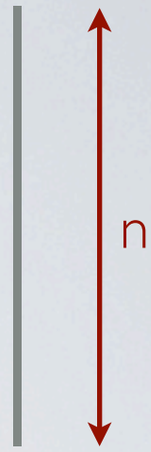
GENOME-WIDE ASSOCIATION STUDIES

lots of GLS because i = 0 ... millions

input

output

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



GENOME-WIDE ASSOCIATION STUDIES

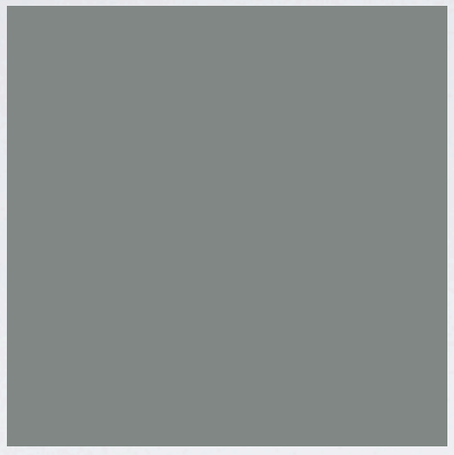
lots of GLS because i = 0 ... millions

$y \in \mathbb{R}^n$ observations (phenotype)

input

output

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

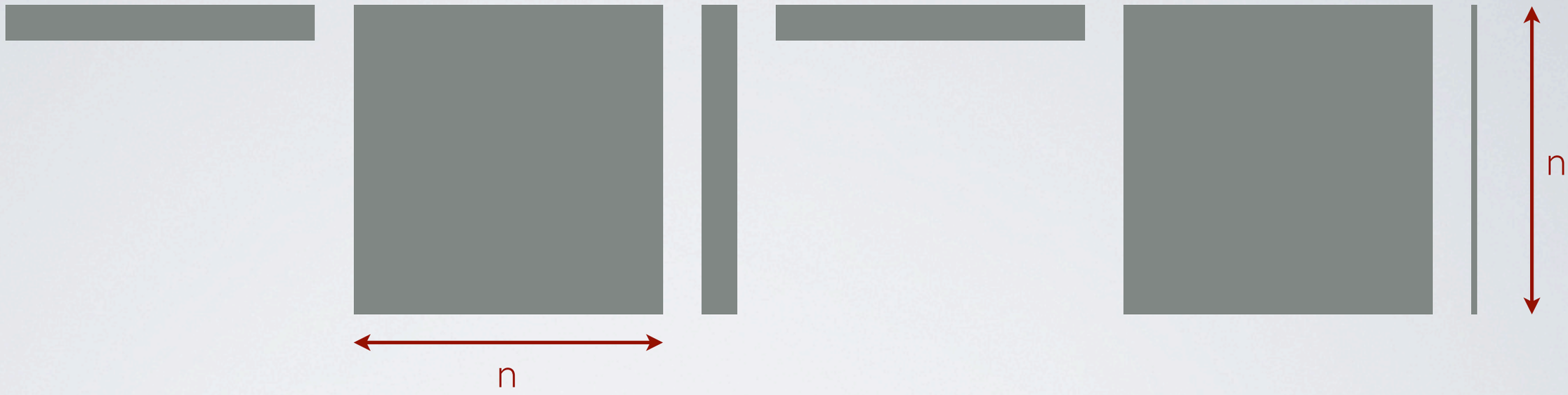


GENOME-WIDE ASSOCIATION STUDIES

lots of GLS because i = 0 ... millions

input	$y \in \mathbb{R}^n$	observations (phenotype)
	$M \in \mathbb{R}^{n \times n}$	observation dependencies
output		

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

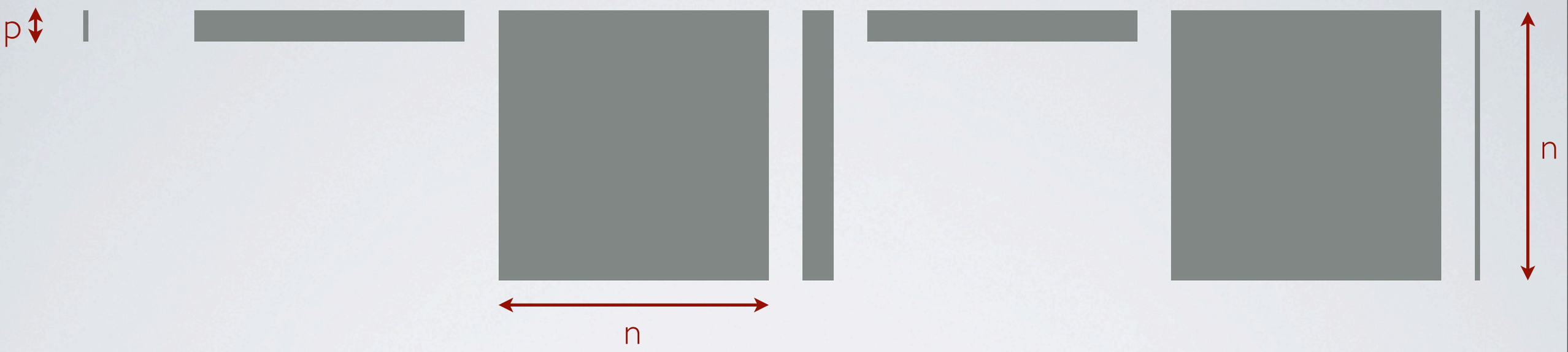


GENOME-WIDE ASSOCIATION STUDIES

lots of GLS because i = 0 ... millions

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

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

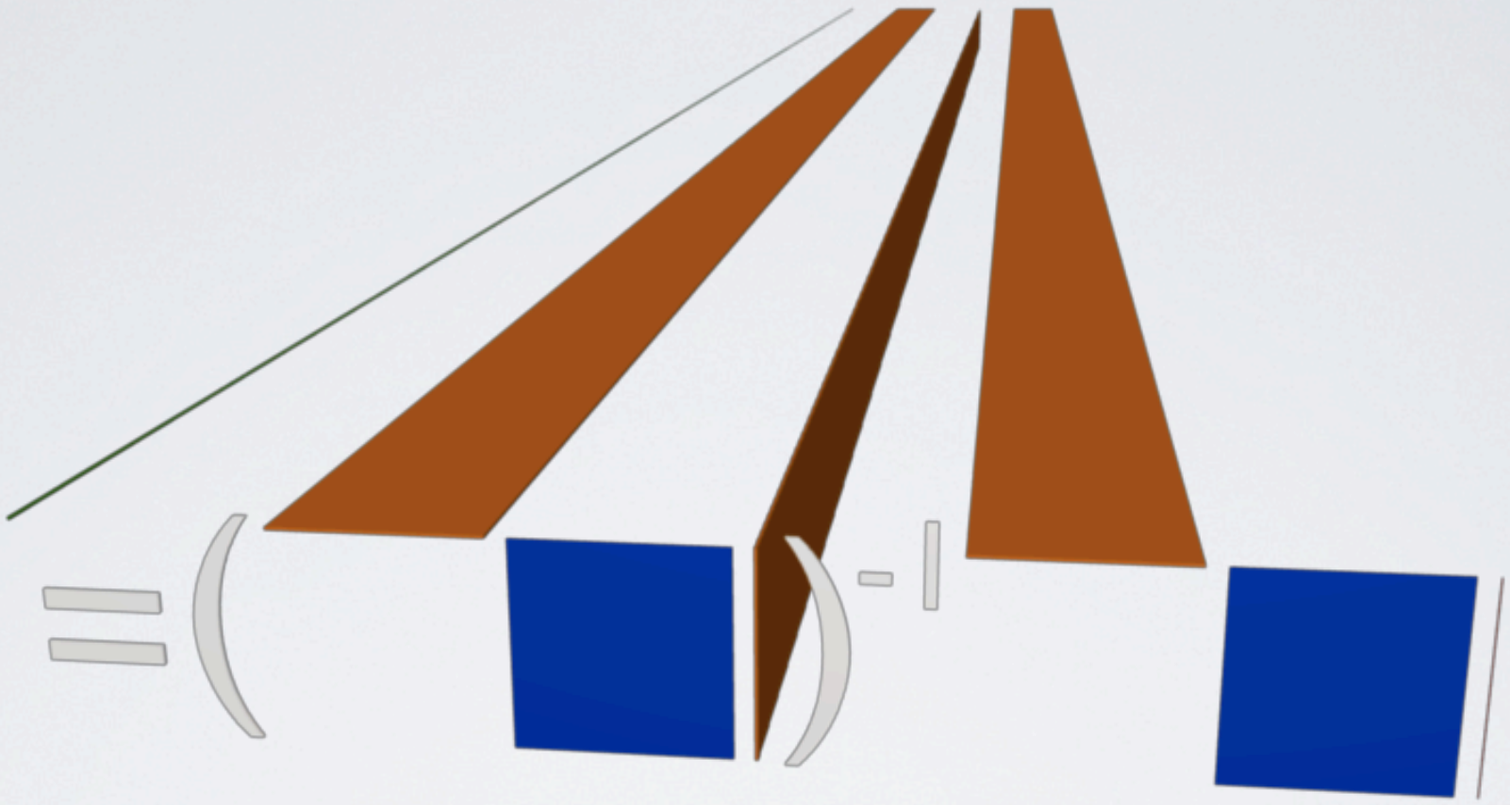


GENOME-WIDE ASSOCIATION STUDIES

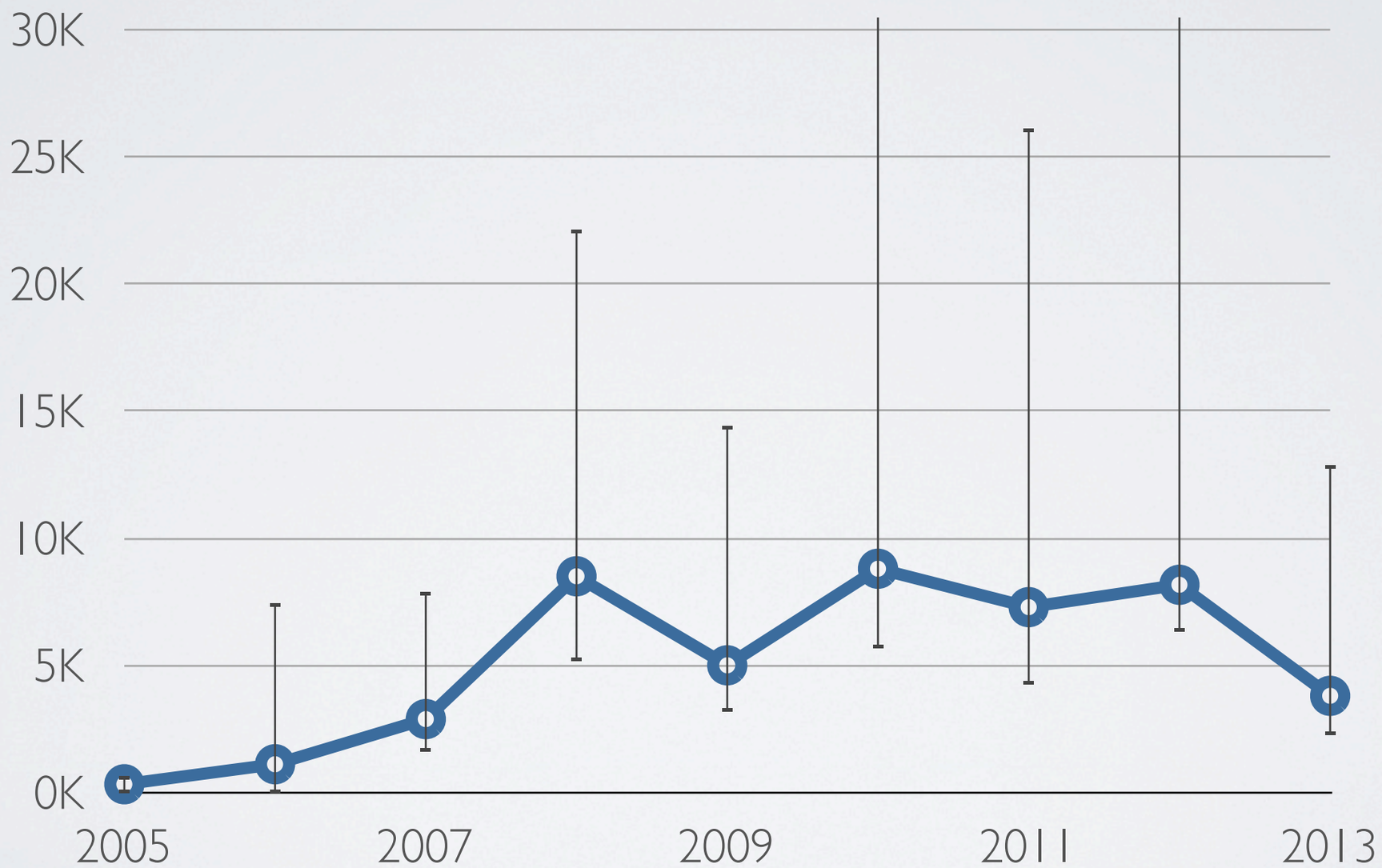
lots of GLS because i = 0 ... millions

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

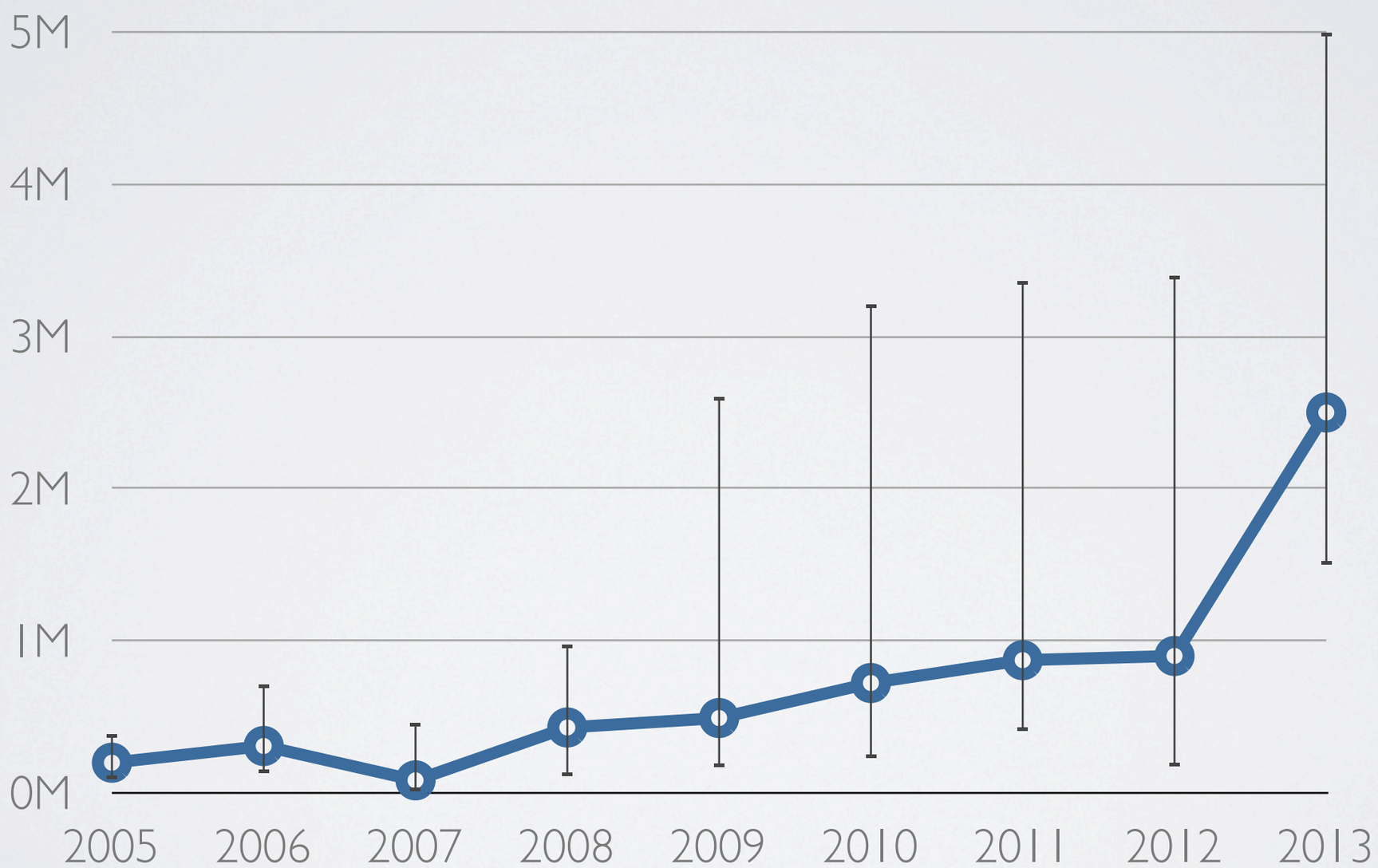
$$r_i \leftarrow (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y$$



GWAS STATS


Sample size n (# of people)
30K
25K
20K
15K
10K
5K
0K
2005
2007
2009
2011
2013

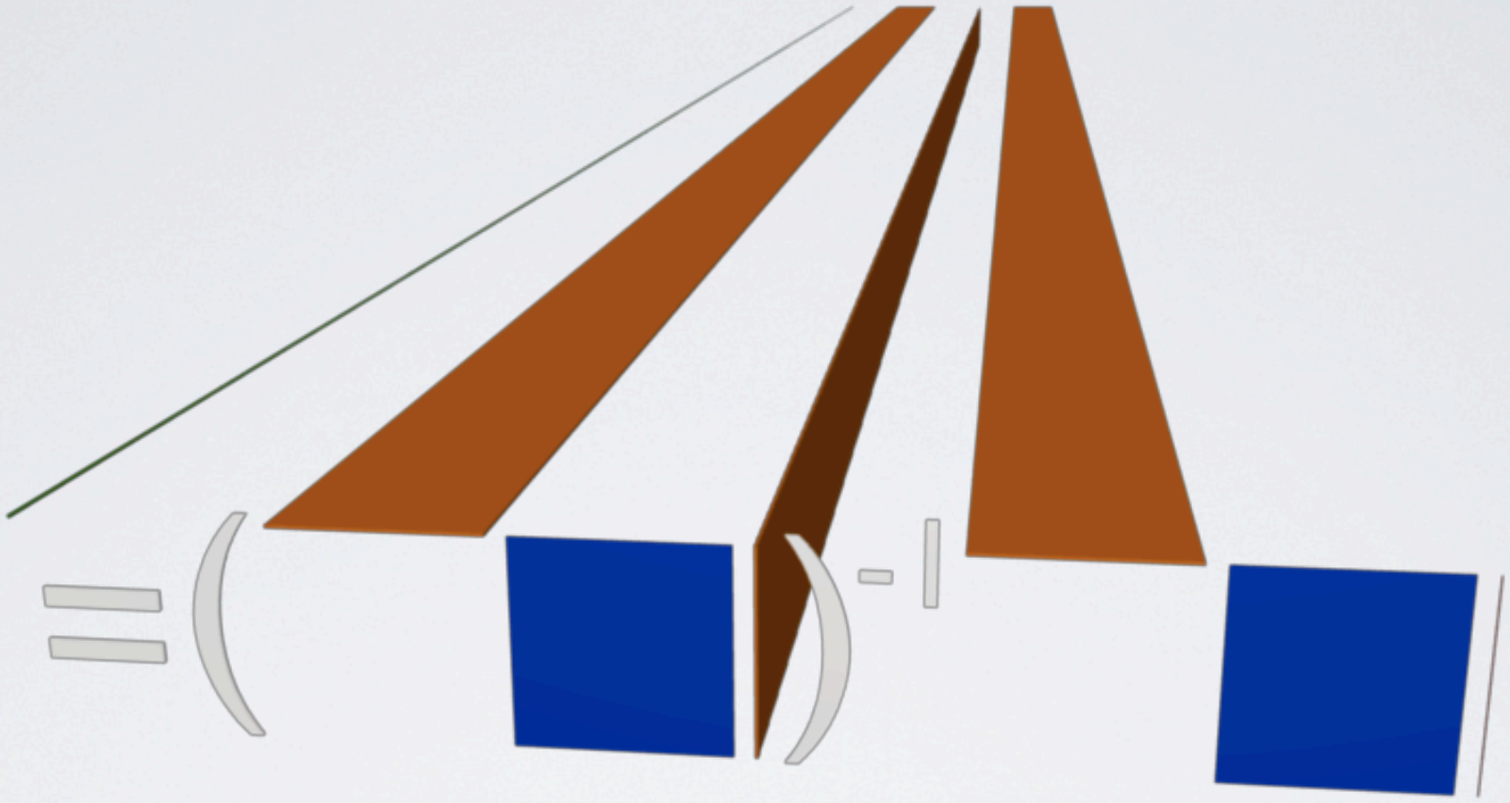
GWAS STATS


#SNPs m (passing QC)
5M
4M
3M
2M
1M
0M
2005
2006
2007
2008
2009
2010
2011
2012
2013

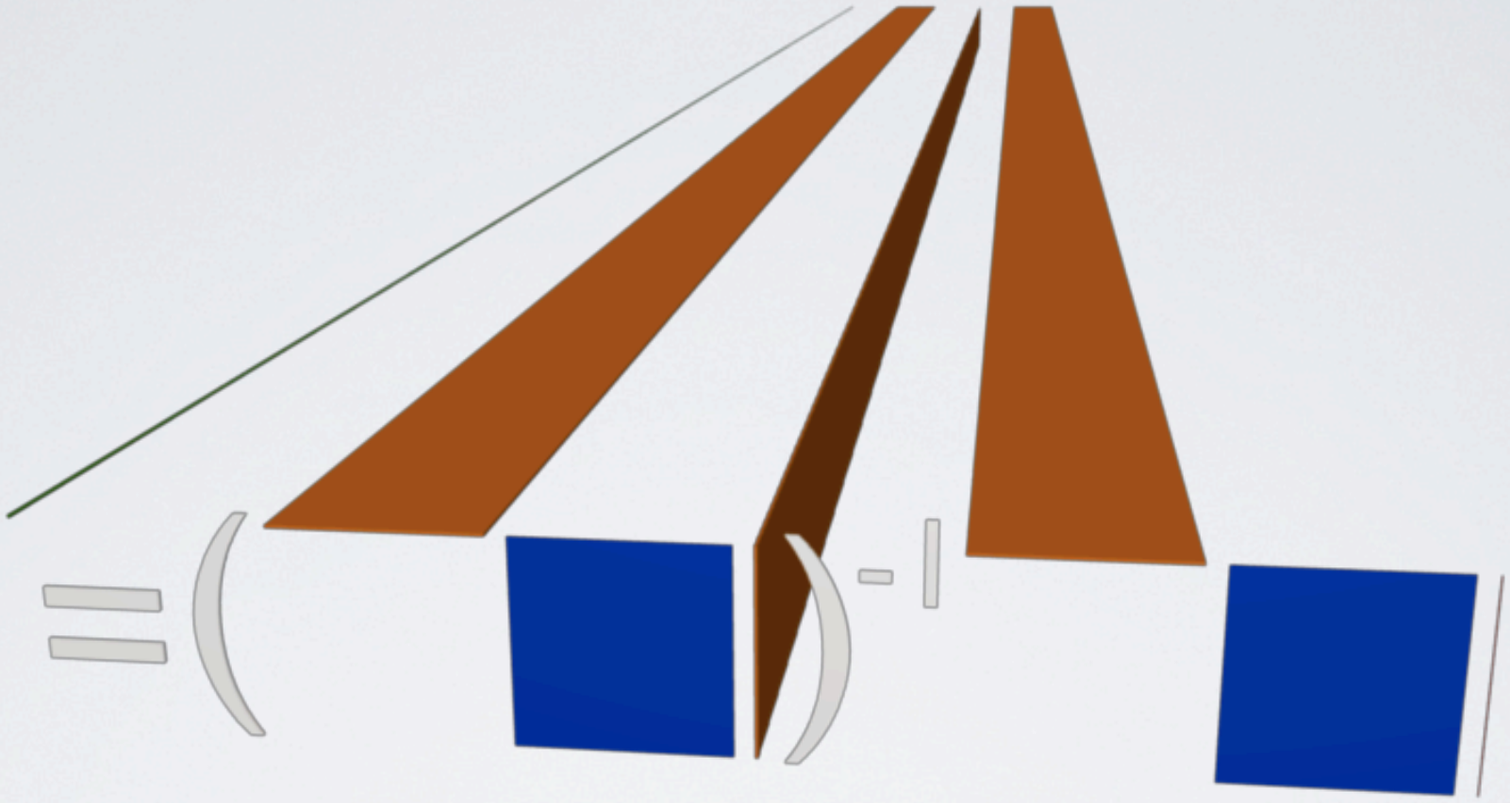
GWAS STATS

Largest #SNPs m (passing QC)

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



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



THE NUMBERS

# DNA fragments (nucleotides)	$m \sim 48 - 250$ million
# samples	$n \sim 10\,000$
# 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 TB

OUTLINE

- Intoduction and motivation
- 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: $LL^T := M$

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

BASIC ALGORITHM

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

Cholesky once during initialization: $LL^T := M$

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

One trsv during initialization: $\hat{y} := L^{-1} y$

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

BASIC ALGORITHM

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

Cholesky once during initialization: $LL^T := M$

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

One trsv during initialization: $\hat{y} := L^{-1} y$

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

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

BASIC ALGORITHM

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

Cholesky once during initialization: $LL^T := M$

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

One trsv during initialization: $\hat{y} := L^{-1} y$

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

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

One trsm per iteration step i: $\hat{X}_i := L^{-1} X_i$

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

BASIC ALGORITHM

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

Cholesky once during initialization: $LL^T := M$

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

One trsv during initialization: $\hat{y} := L^{-1} y$

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

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

One trsm per iteration step i: $\hat{X}_i := L^{-1} X_i$

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

BASIC ALGORITHM

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

Cholesky once during initialization: $LL^T := M$

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

One trsv during initialization: $\hat{y} := L^{-1} y$

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

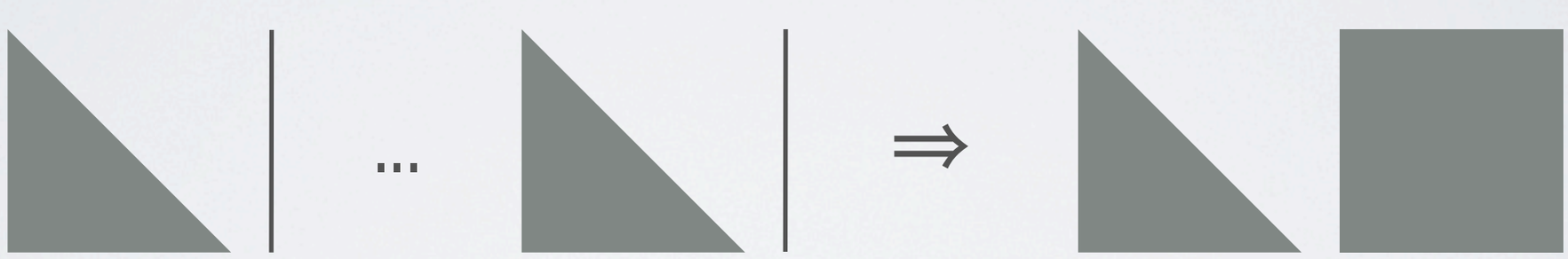
$$r_i \leftarrow ((L^{-1} X_i)^T L^{-1} X_i)^{-1} (L^{-1} X_i)^T \hat{y}$$

One trsm per iteration step i: $\hat{X}_i := L^{-1} X_i$

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

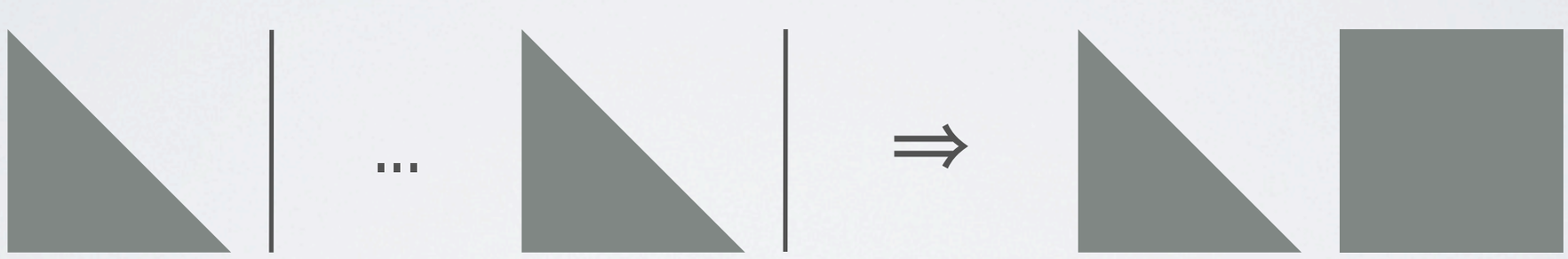
OPTIMIZATIONS

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

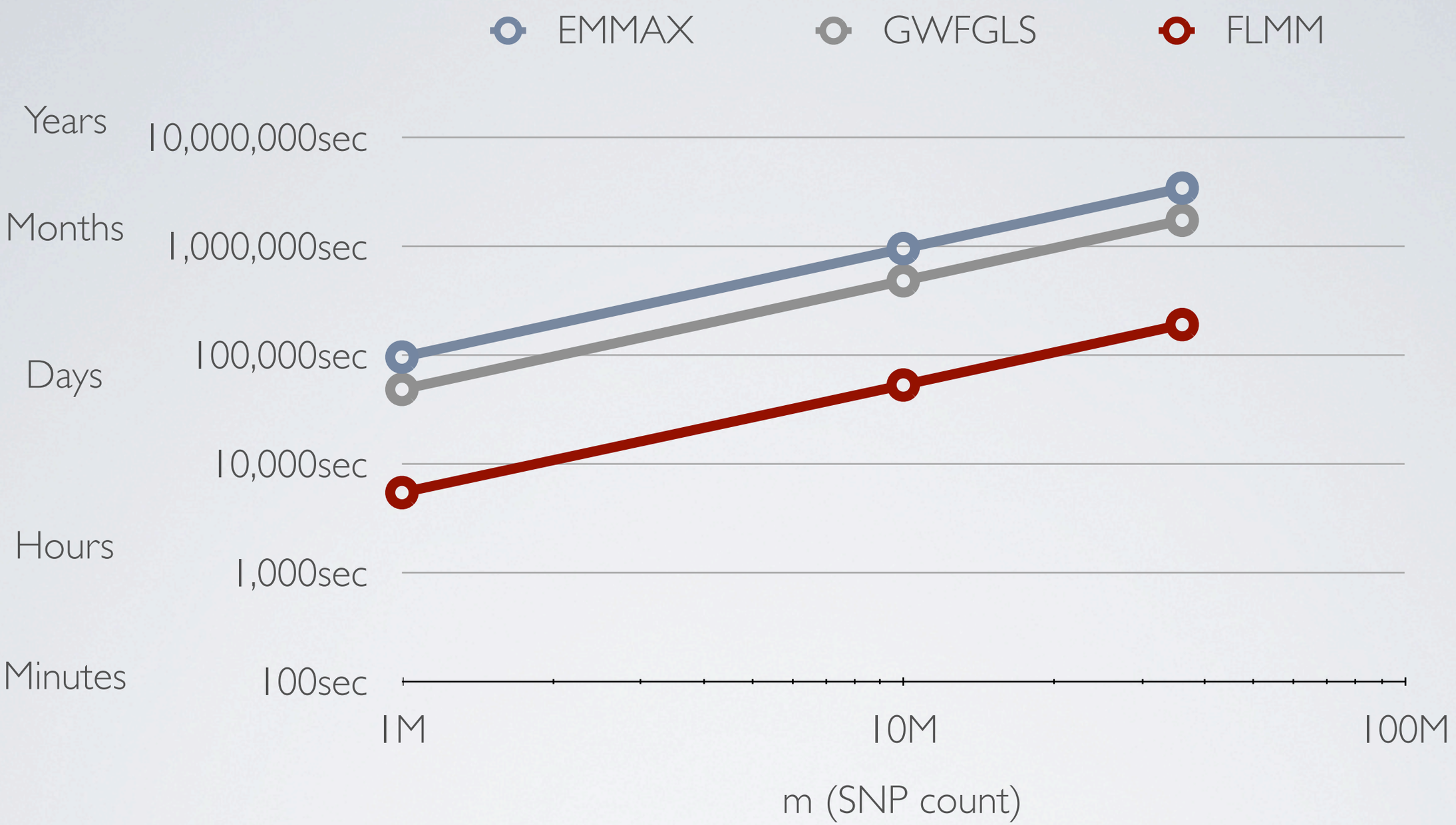


OPTIMIZATIONS

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

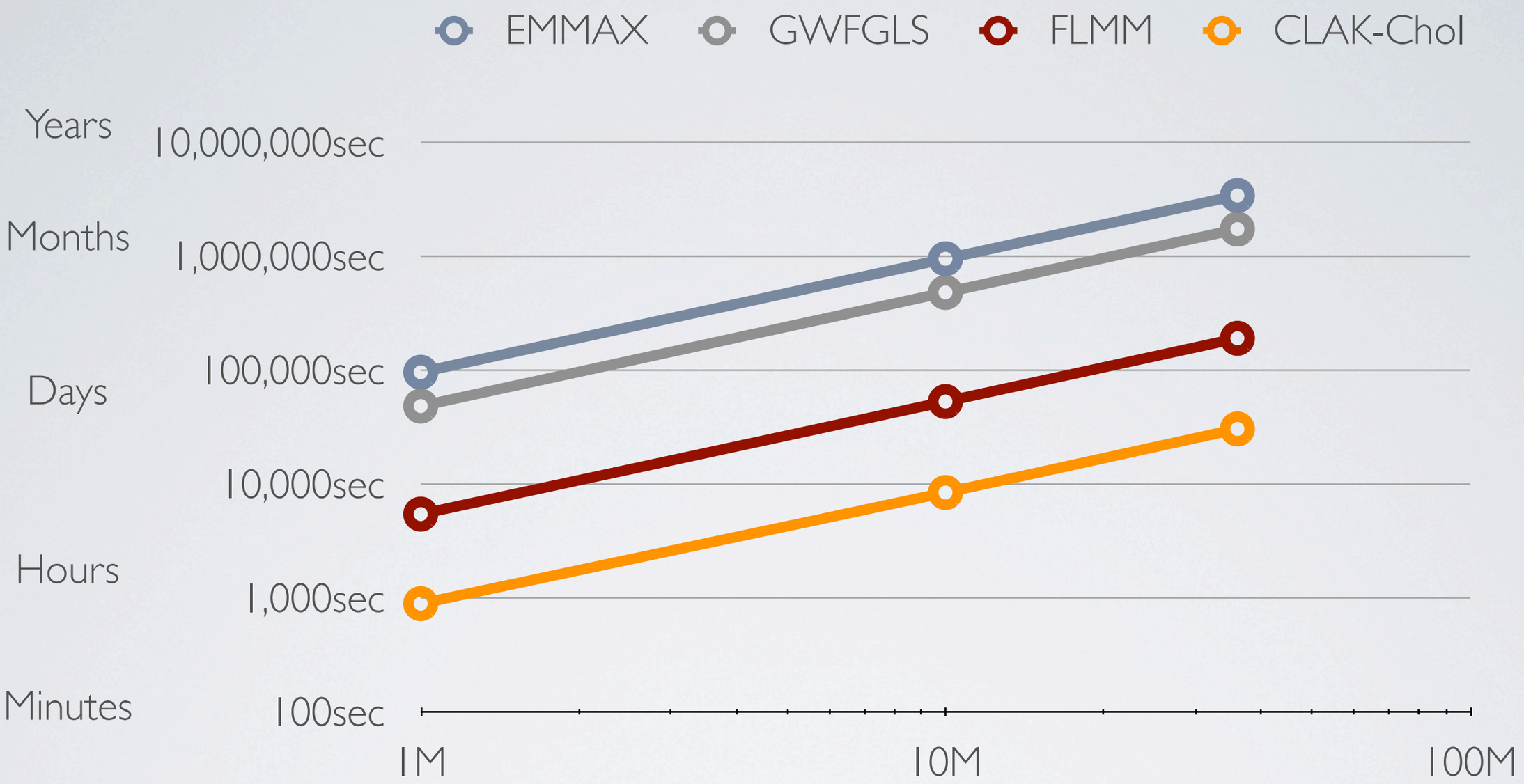


- Out-of-core algorithm
  - read block b_{+1} while computing block b
  - double-buffering technique necessary



PERFORMANCE

From years/months down to weeks/days



PERFORMANCE

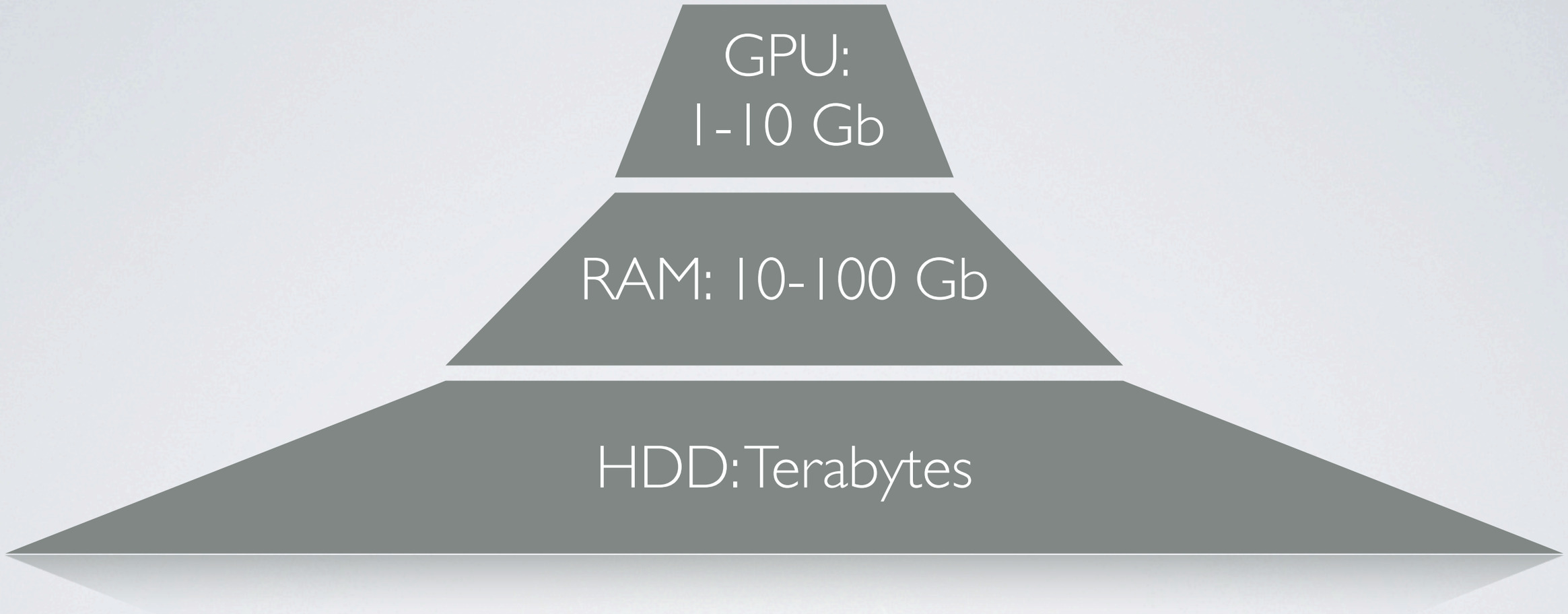
From years/months down to weeks/days

OUTLINE

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

CAN GPUs HELP GO FURTHER?

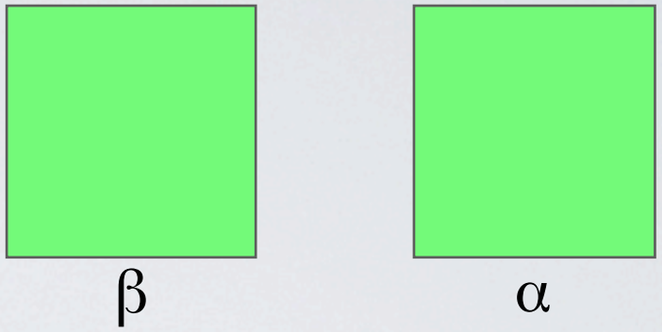
- trsm takes 90-95% of time
  - compute on the GPU
- while GPU computes:
  - CPU computations
  - CPU ⇄ GPU transfers
- our cluster: nVidia Fermi ☞



MEMORY PYRAMID

Need for two levels of streaming

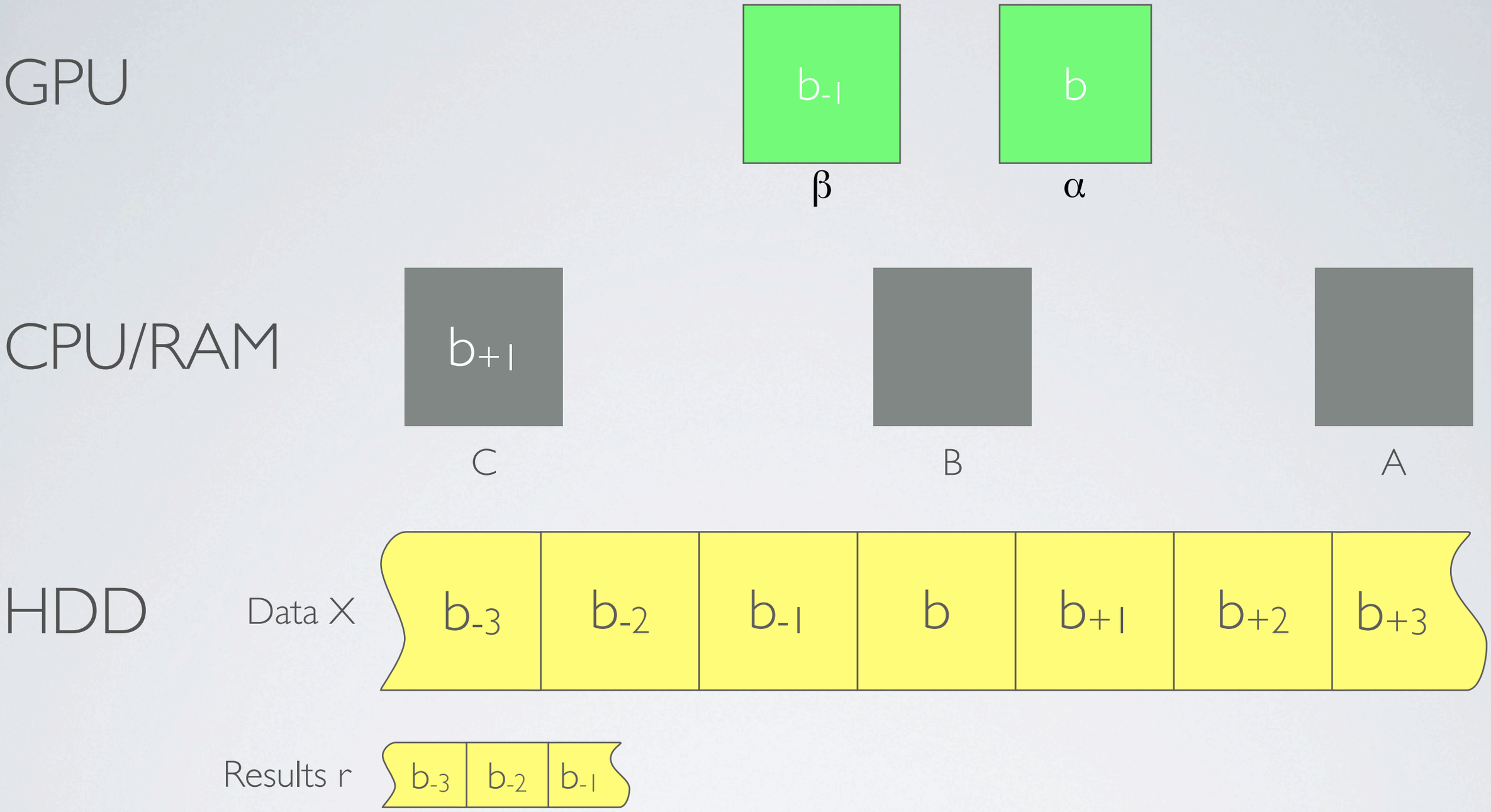
GPU



CPU/RAM

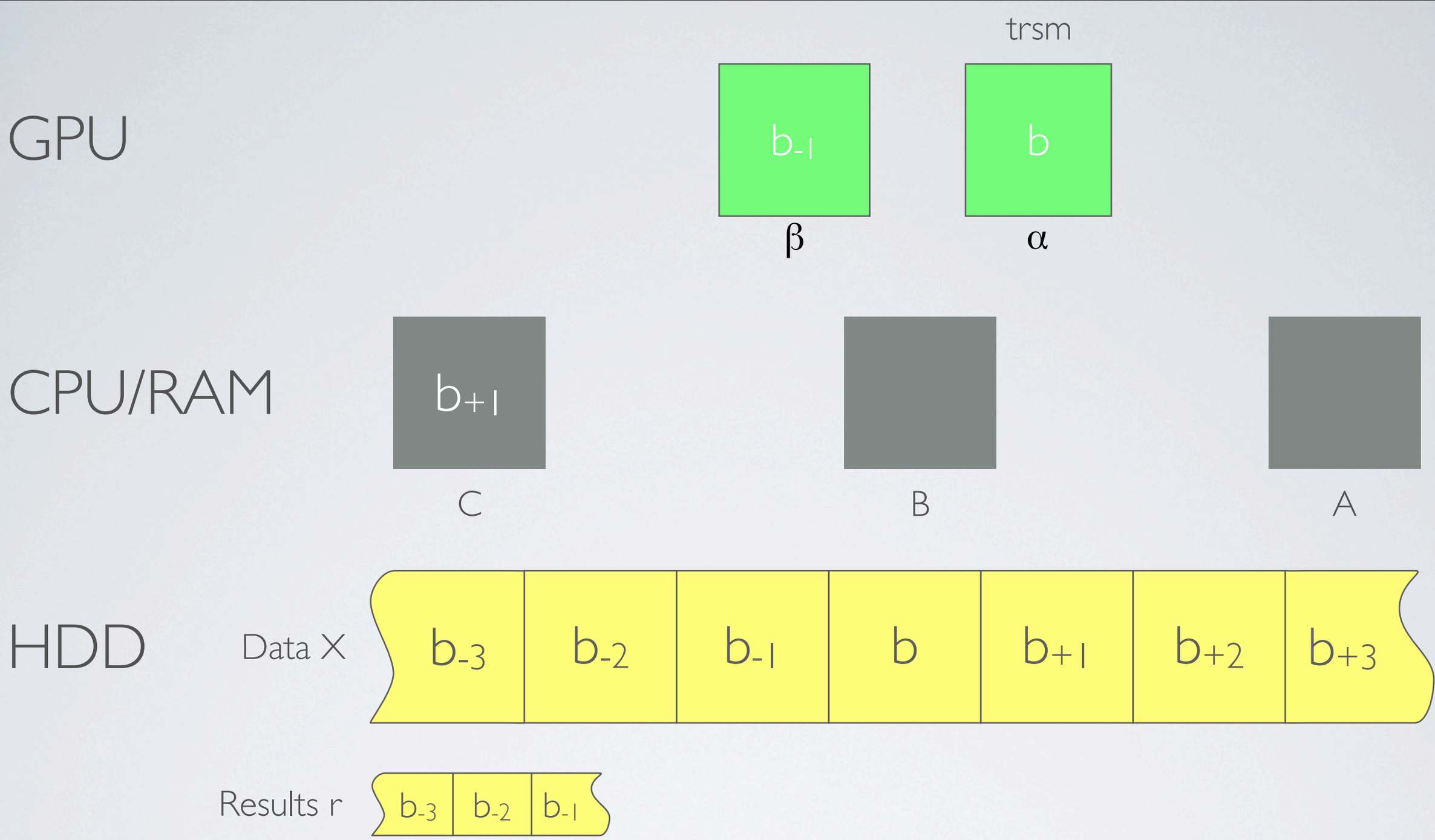
HDD

2-LEVEL TRIPLE-DOUBLE-BUFFERING



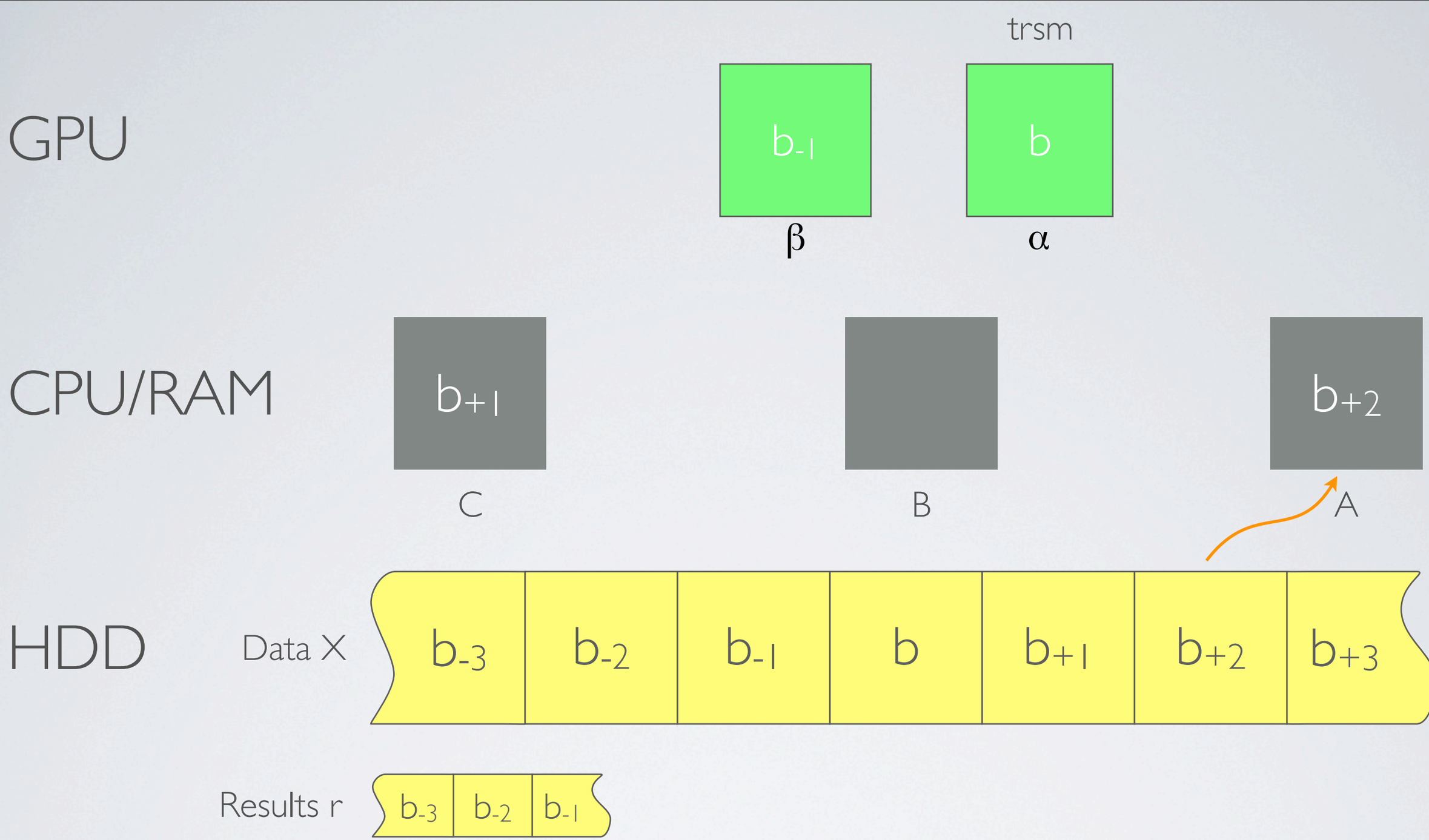
2-LEVEL TRIPLE-DOUBLE-BUFFERING

Assume the GPU holds b and is done with b-1



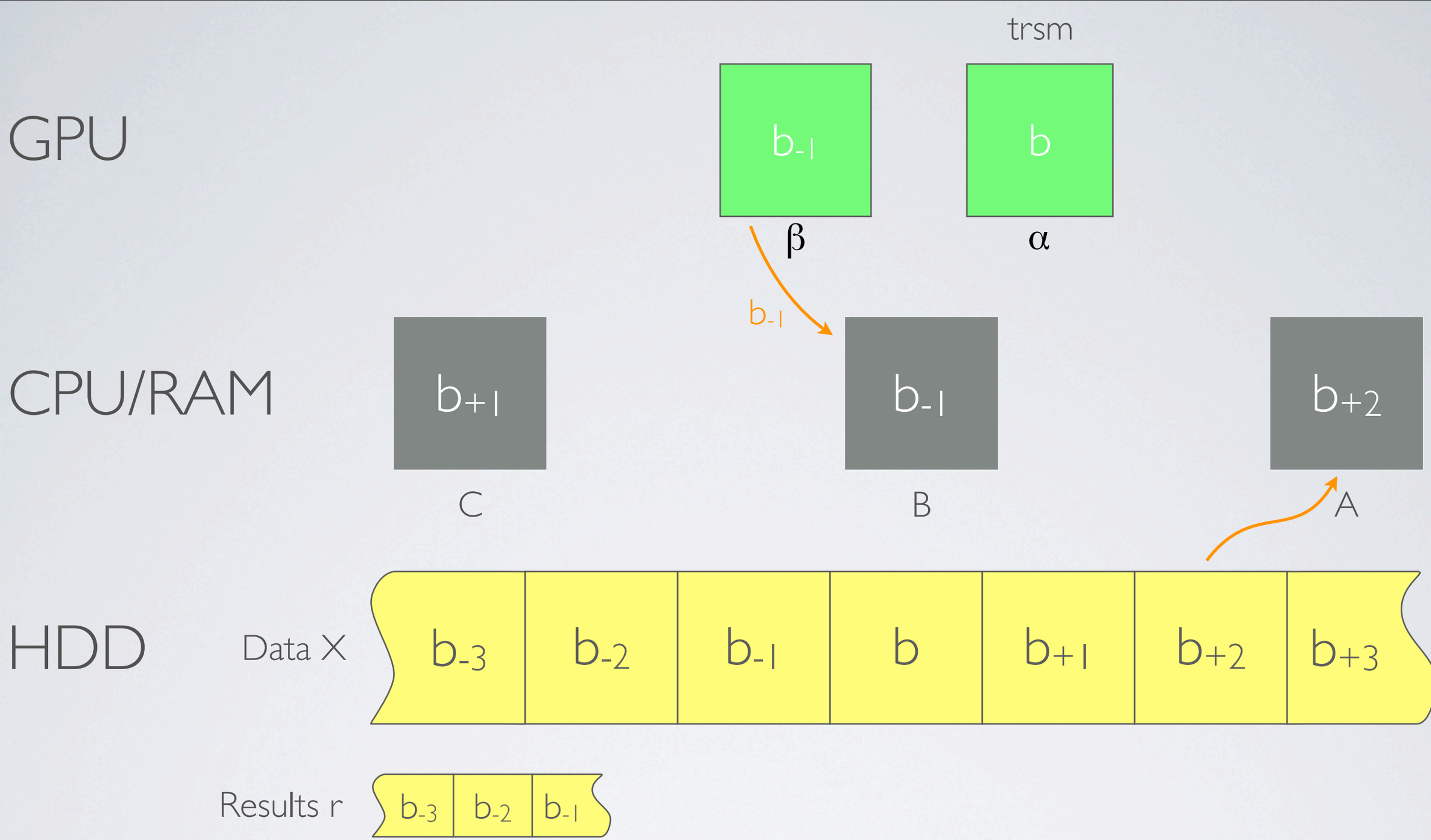
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(1) TRSM of b on GPU



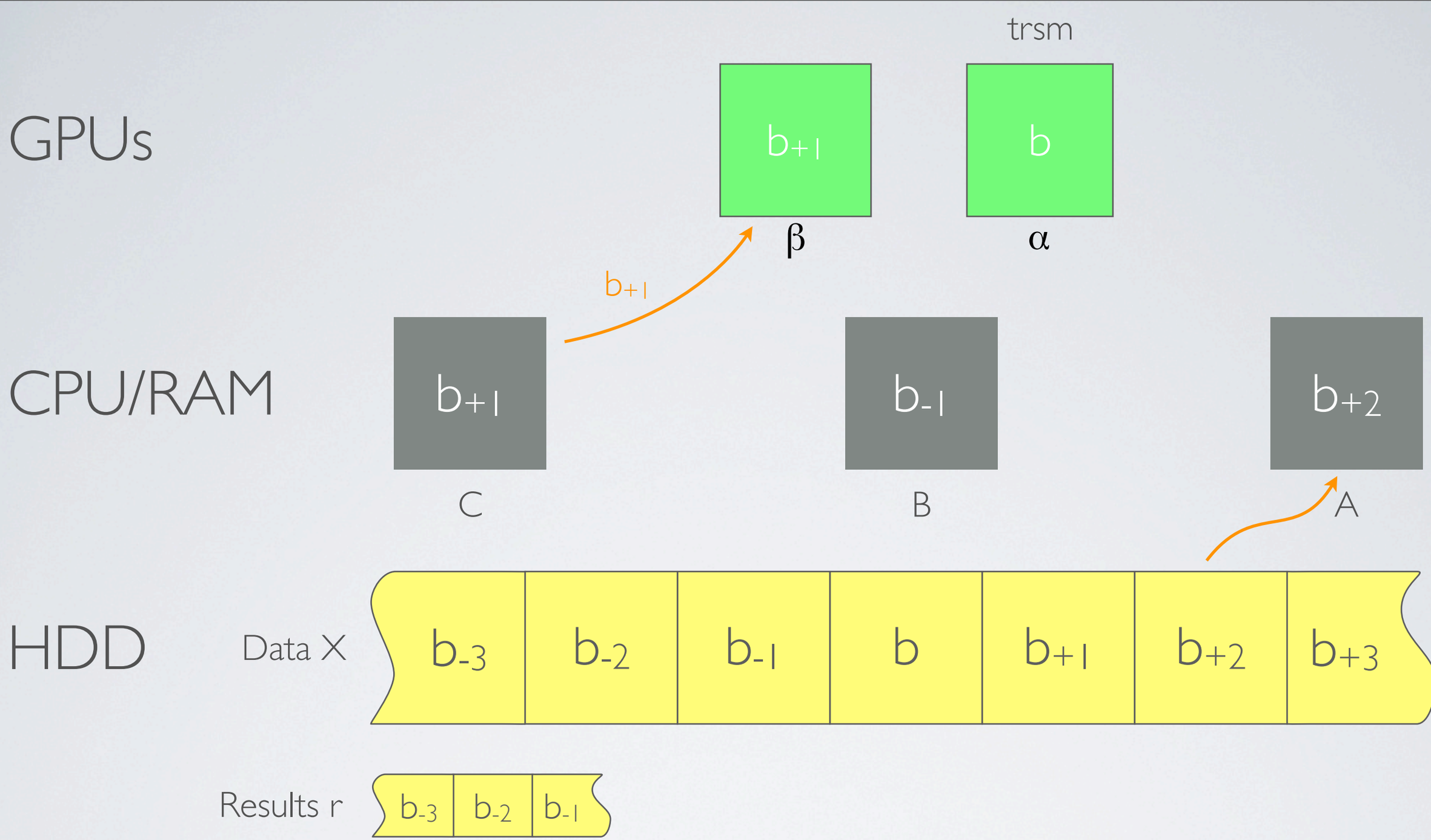
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(2) start reading second-next block from HDD



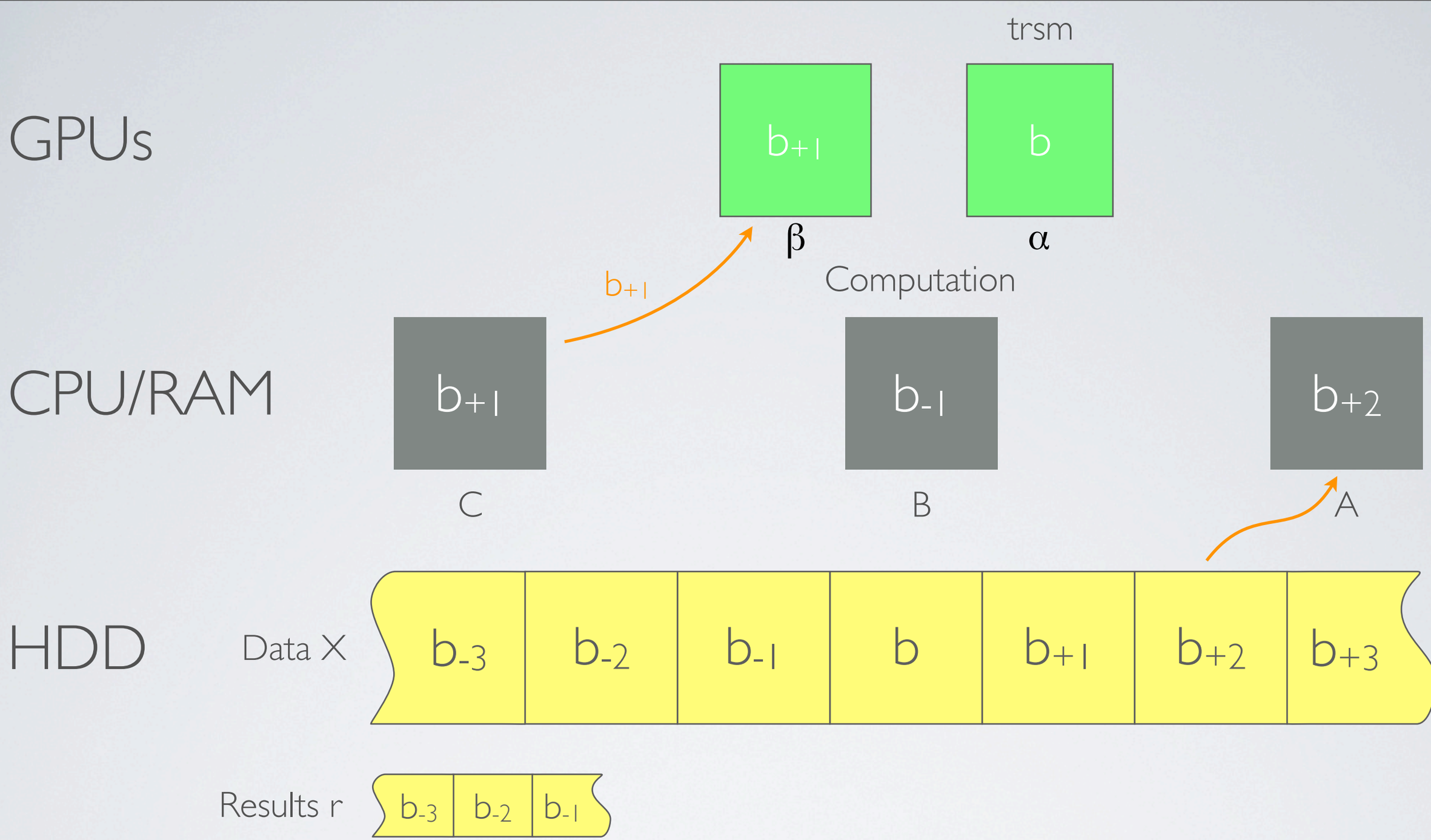
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(3) Retrieve previous results from GPU



2-LEVEL TRIPLE-DOUBLE-BUFFERING

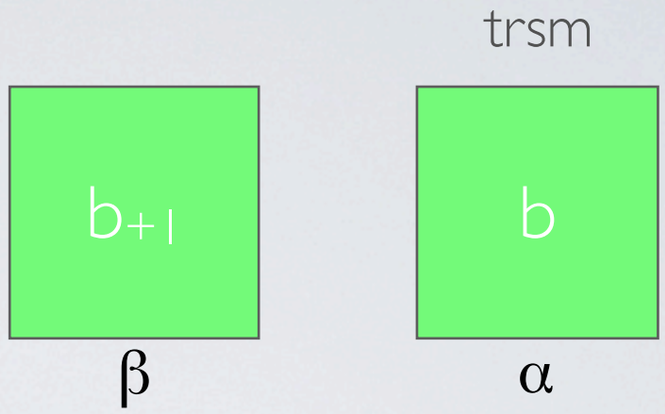
(4) Send next block to GPU



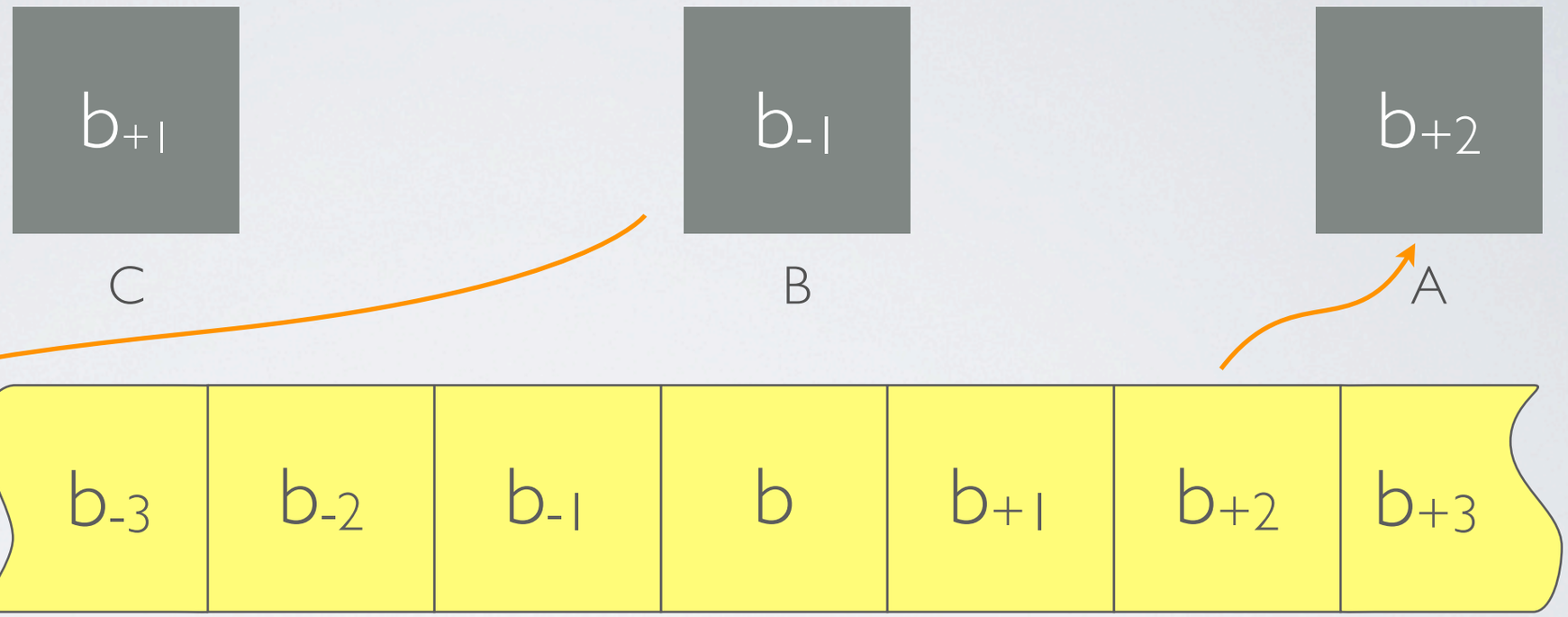
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(5) start CPU computation of least-squares

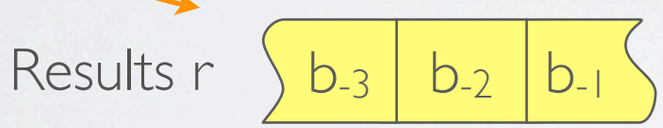
GPUs



CPU/RAM



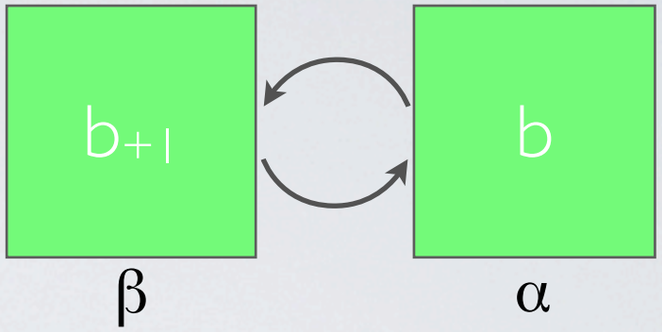
HDD



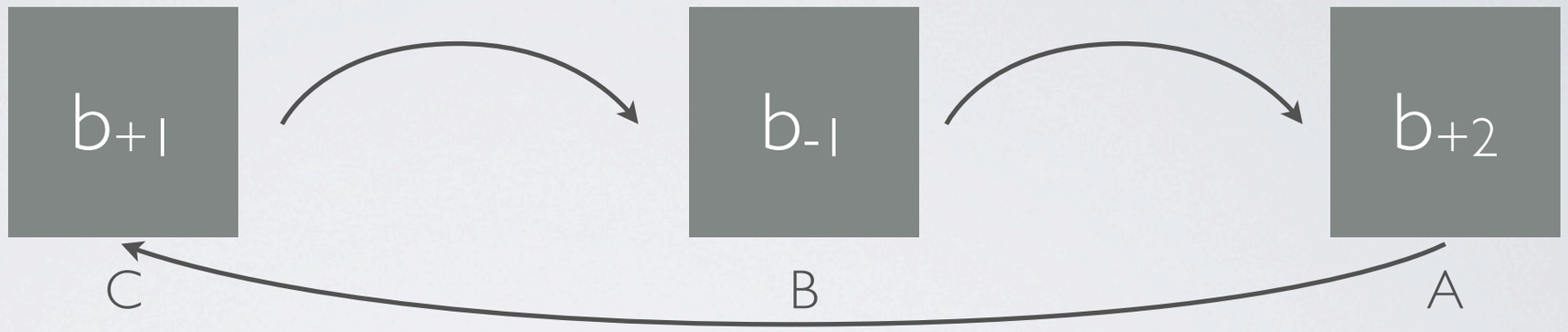
2-LEVEL TRIPLE-DOUBLE-BUFFERING

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

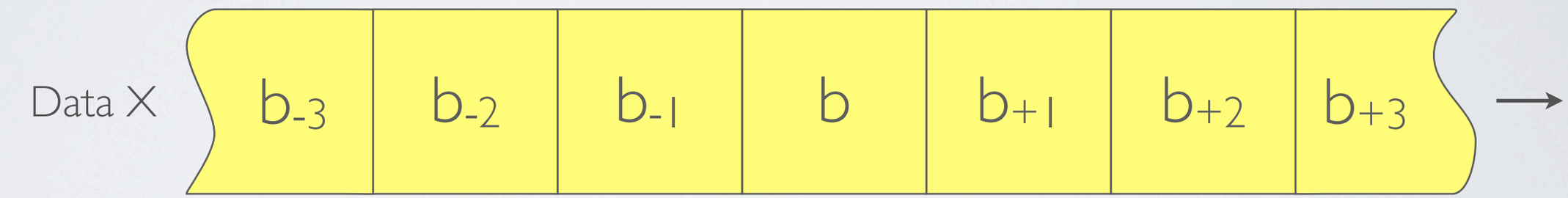
GPUs



CPU/RAM

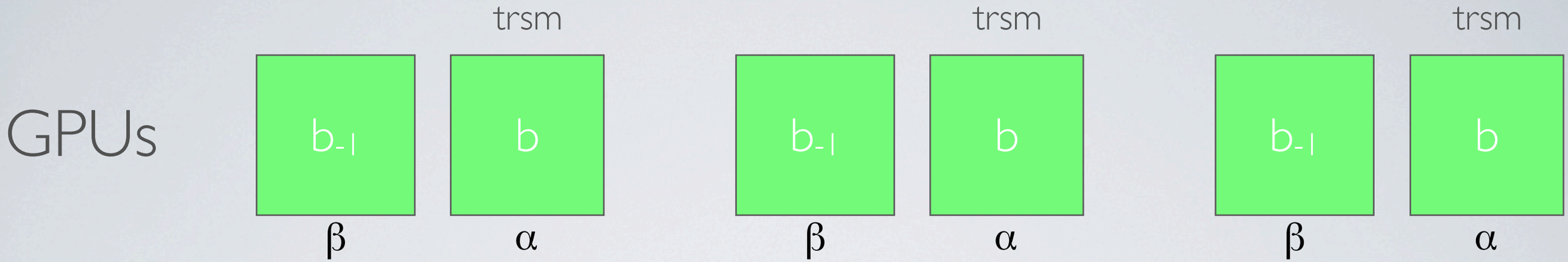


HDD



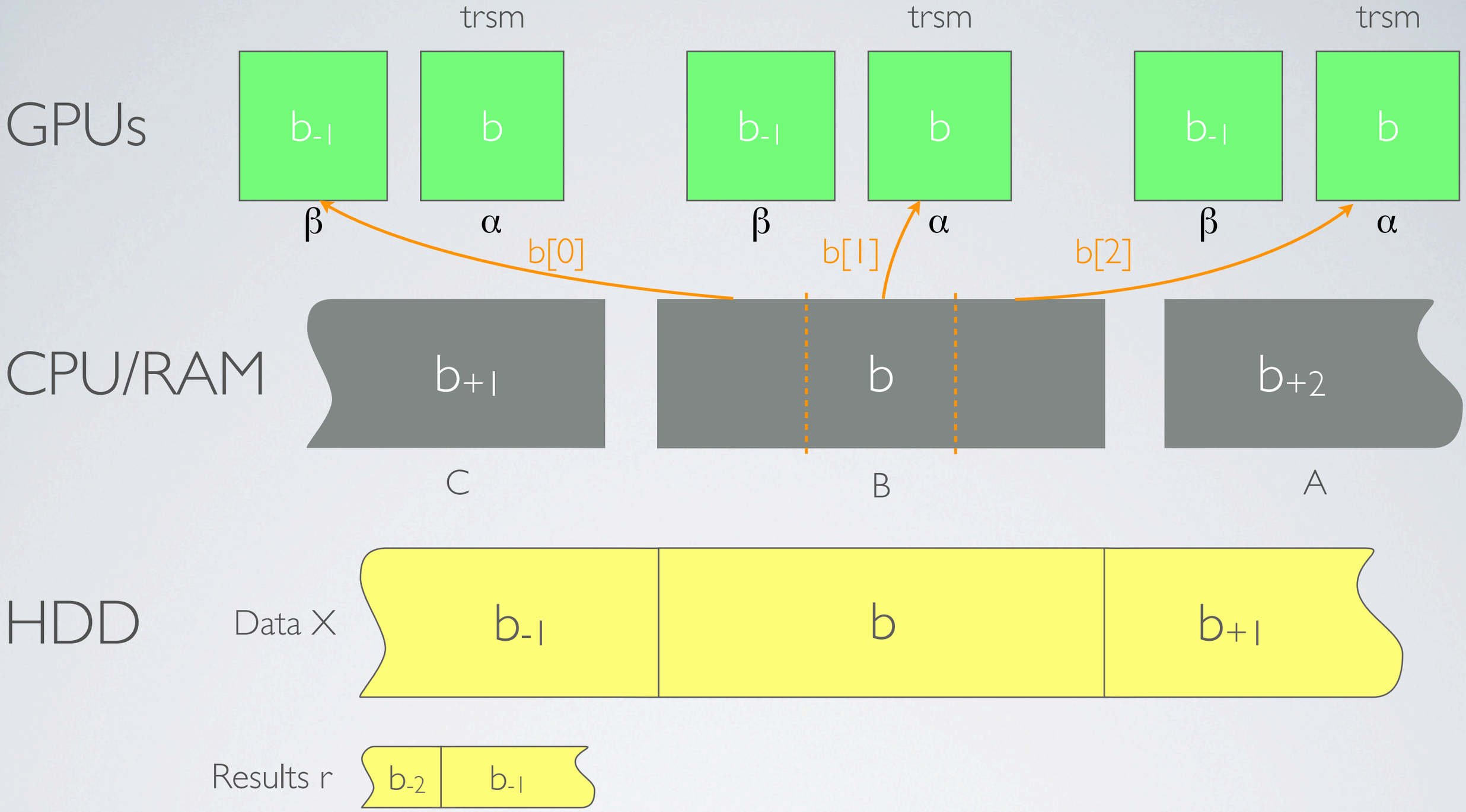
2-LEVEL TRIPLE-DOUBLE-BUFFERING

(7) Rotate buffers, iterate, smile



WITH MULTIPLE GPUs

Increase CPU's block size



WITH MULTIPLE GPUs

Increase CPU's block size

TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

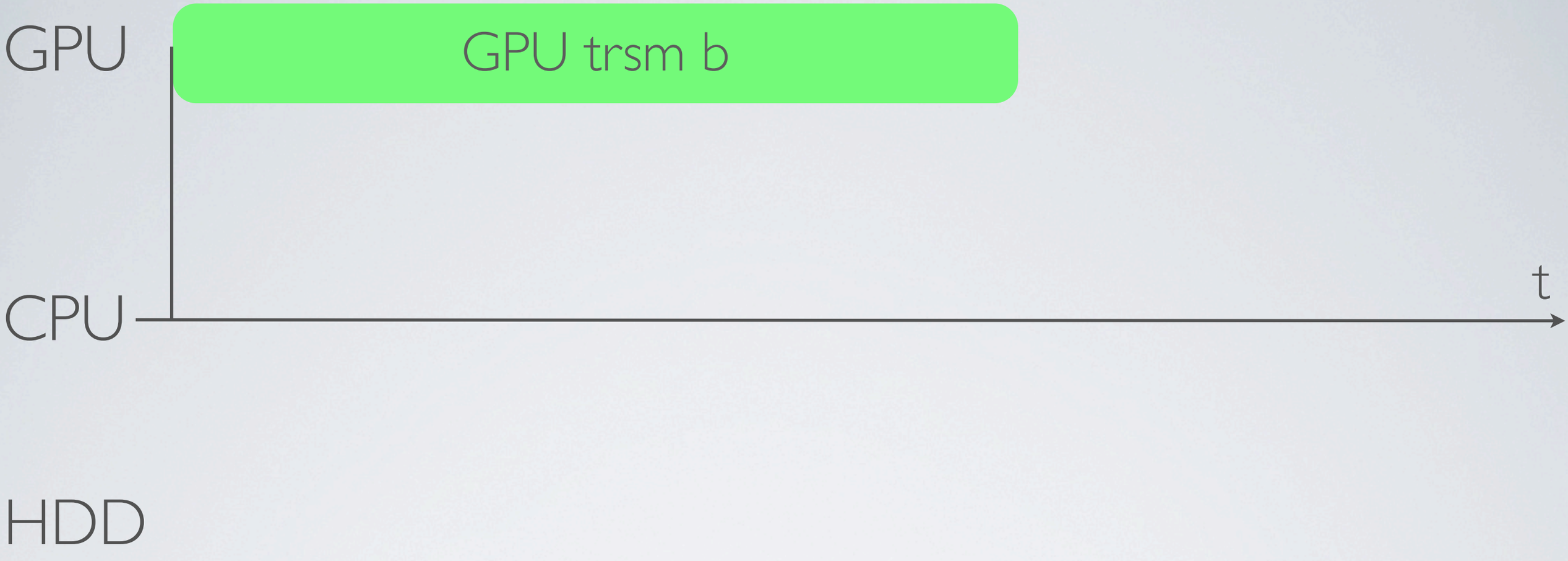
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

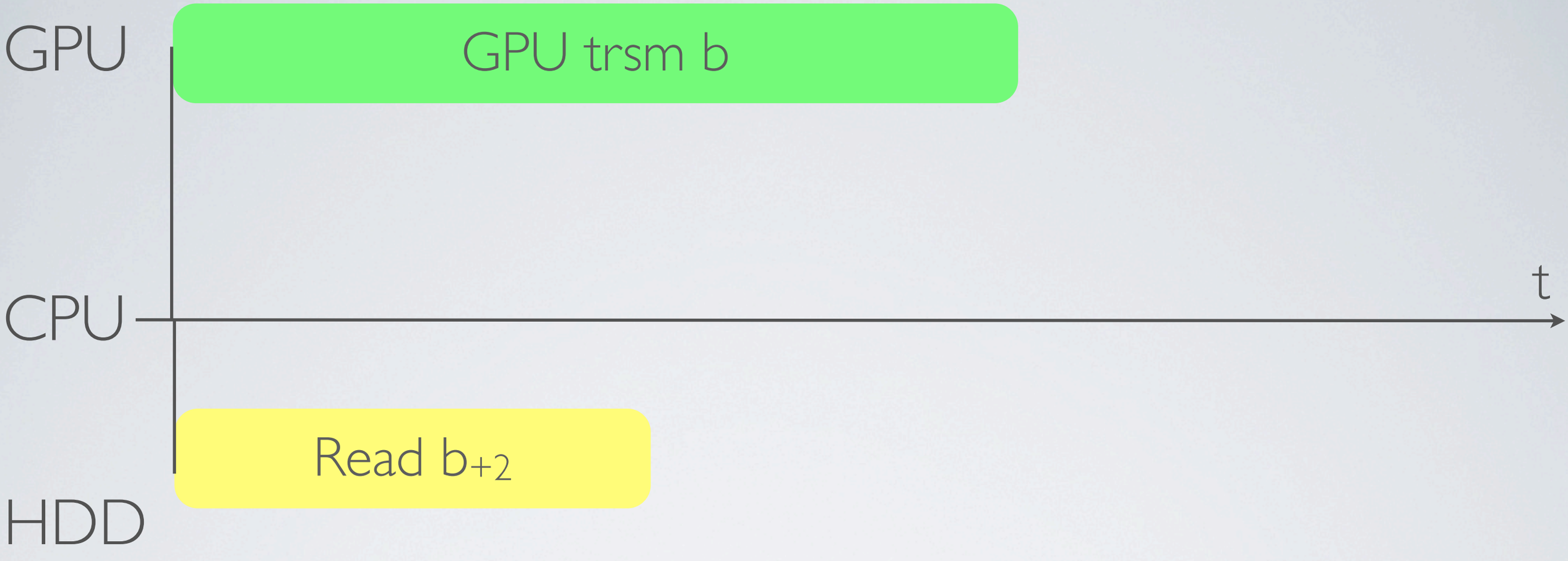
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

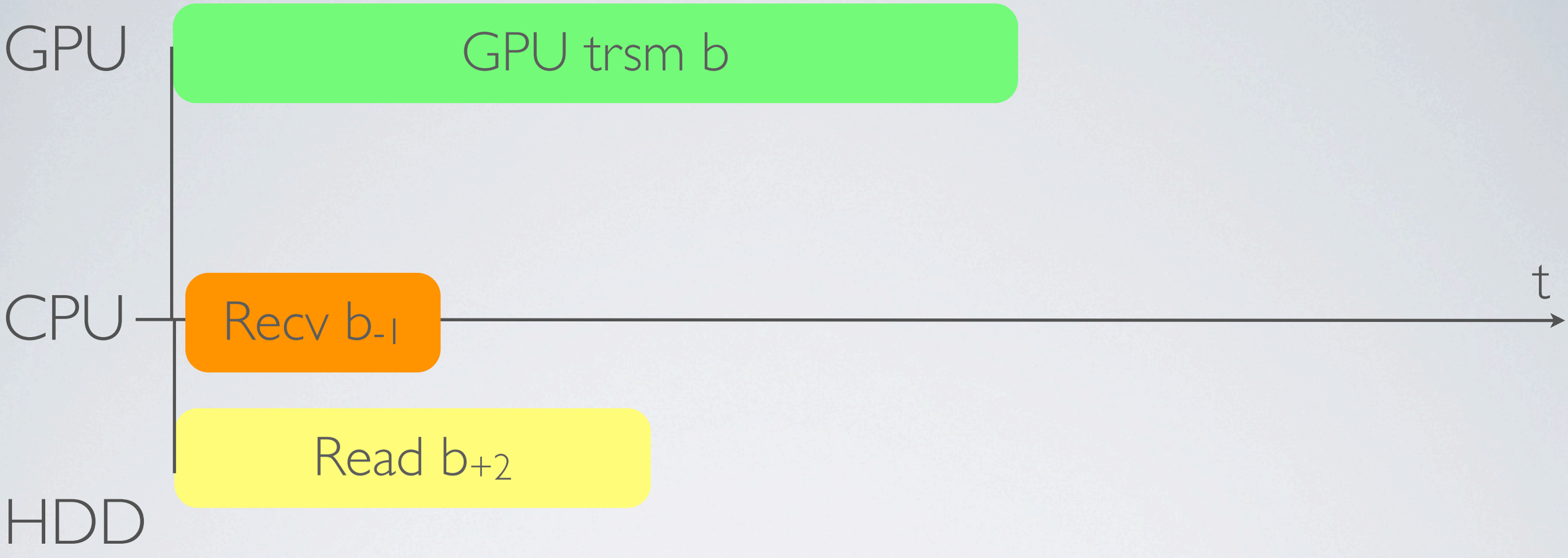
GPU computation

Data dependencies

HDD ⇄ CPU transfer

CPU computation

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

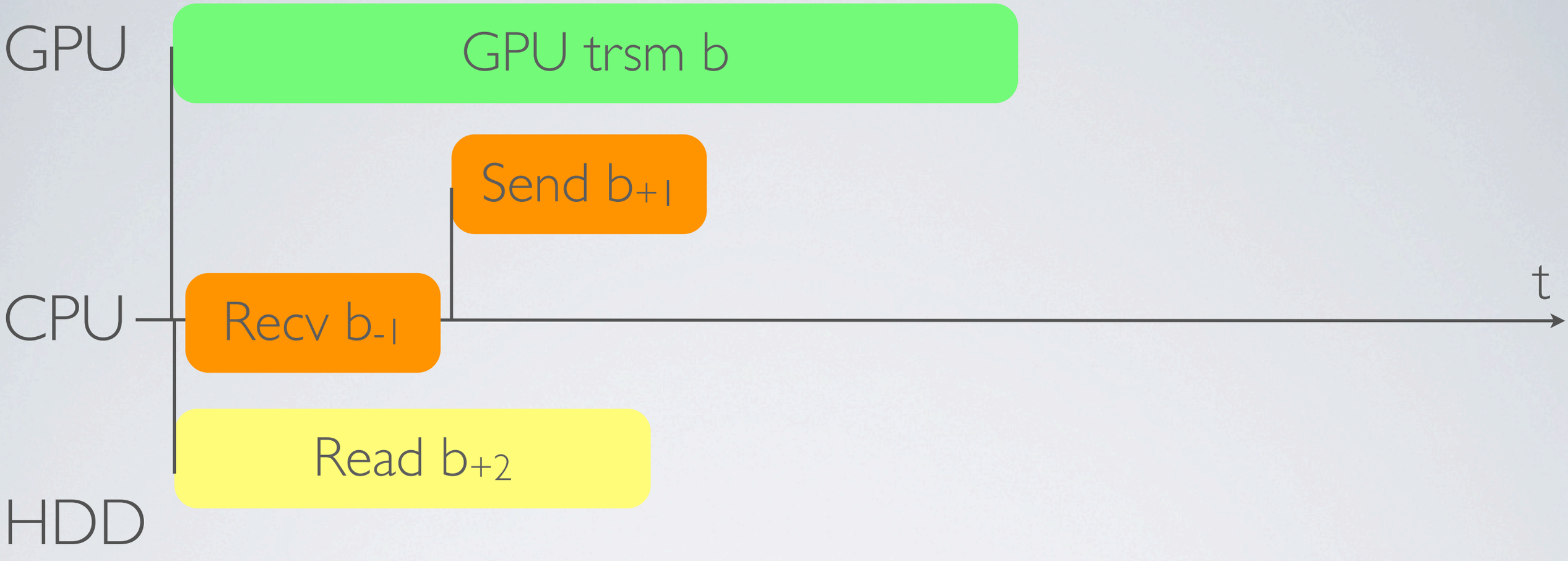
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

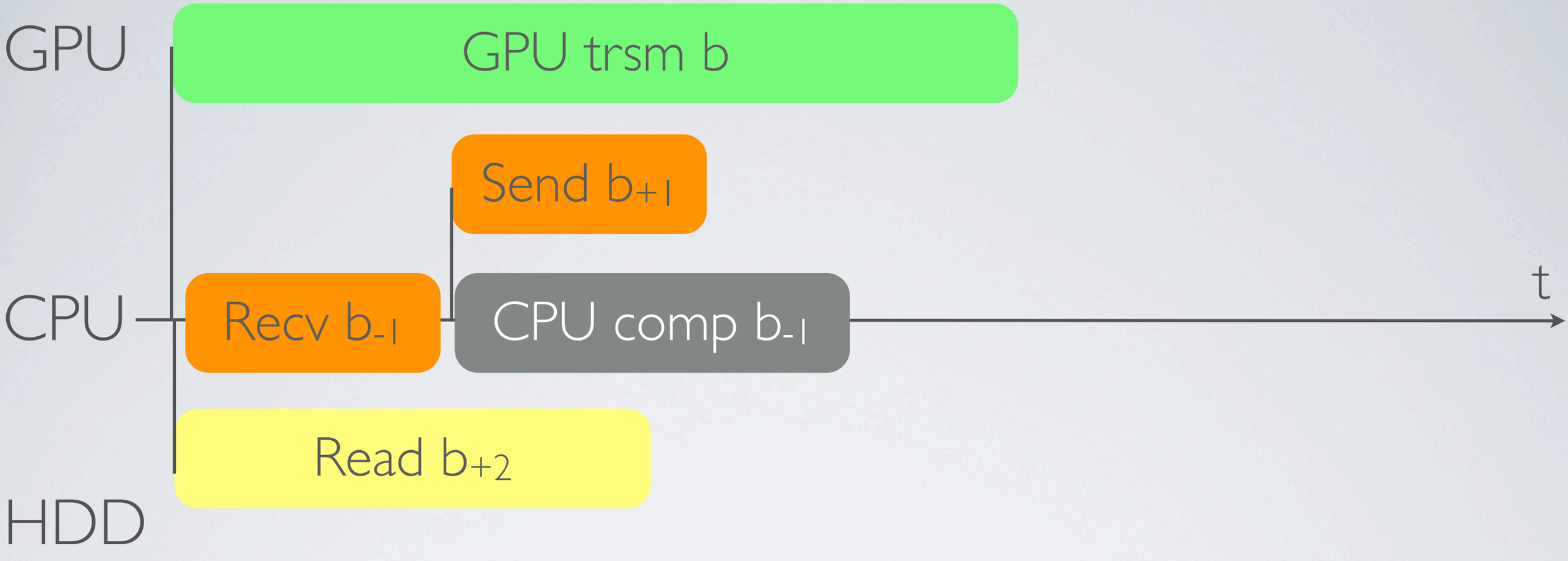
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

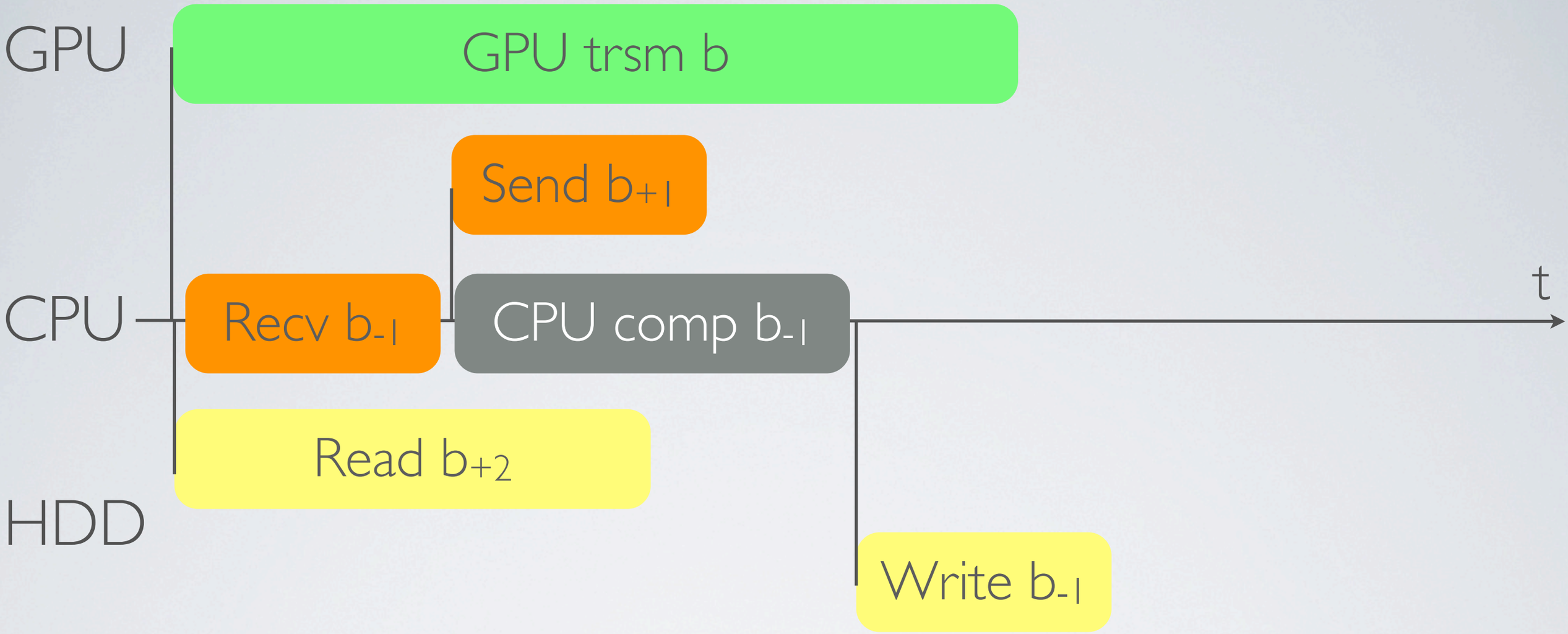
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

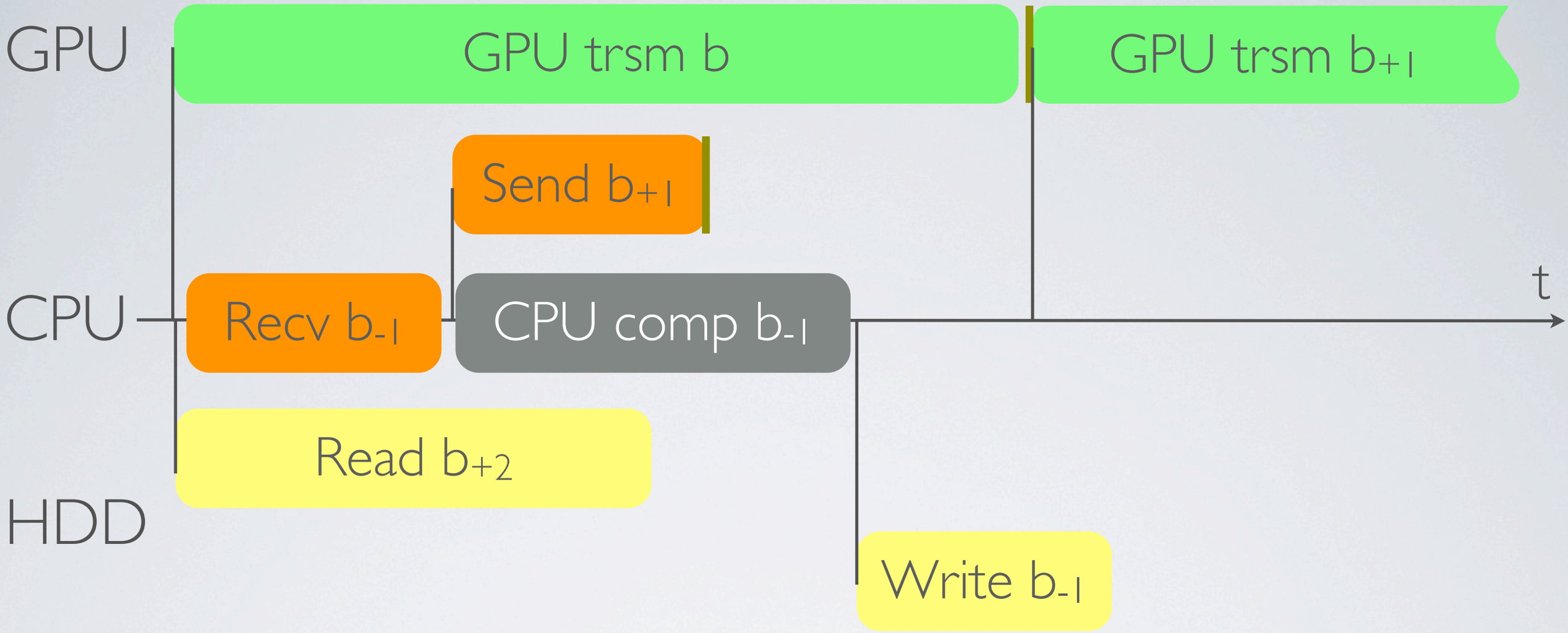
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

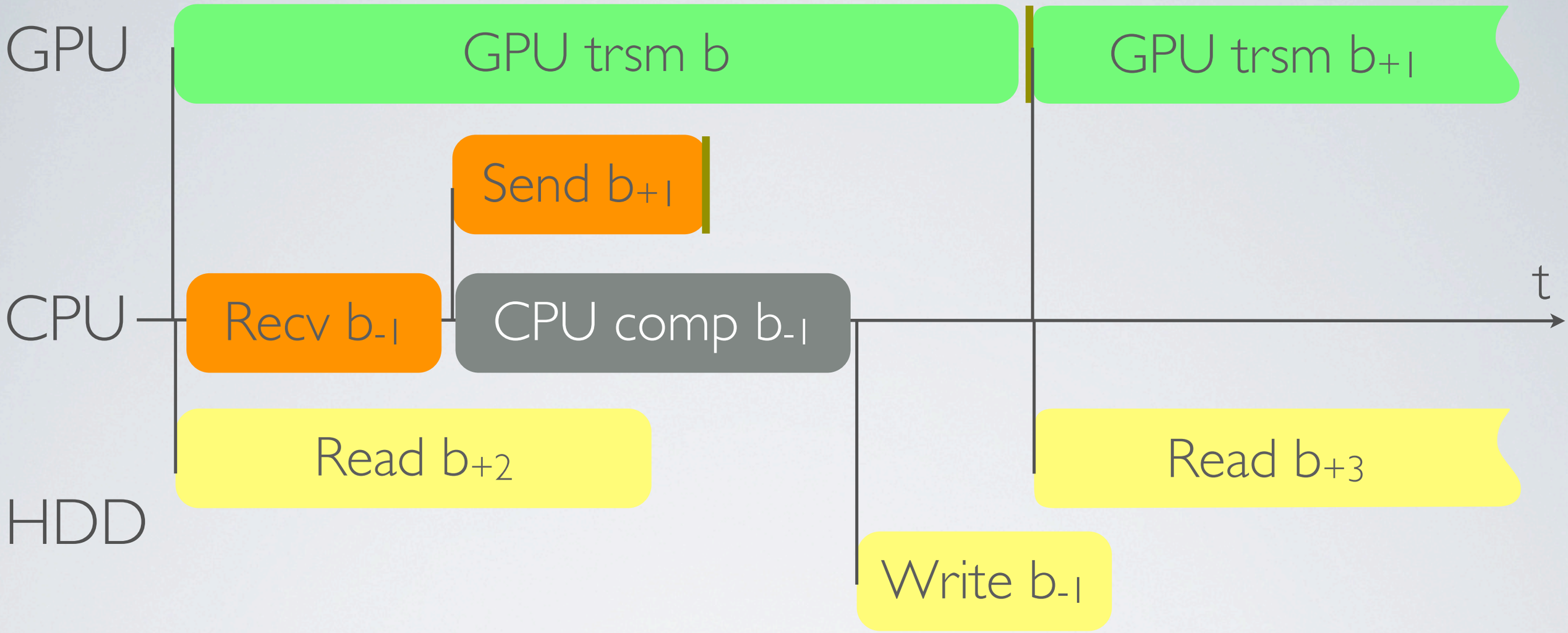
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

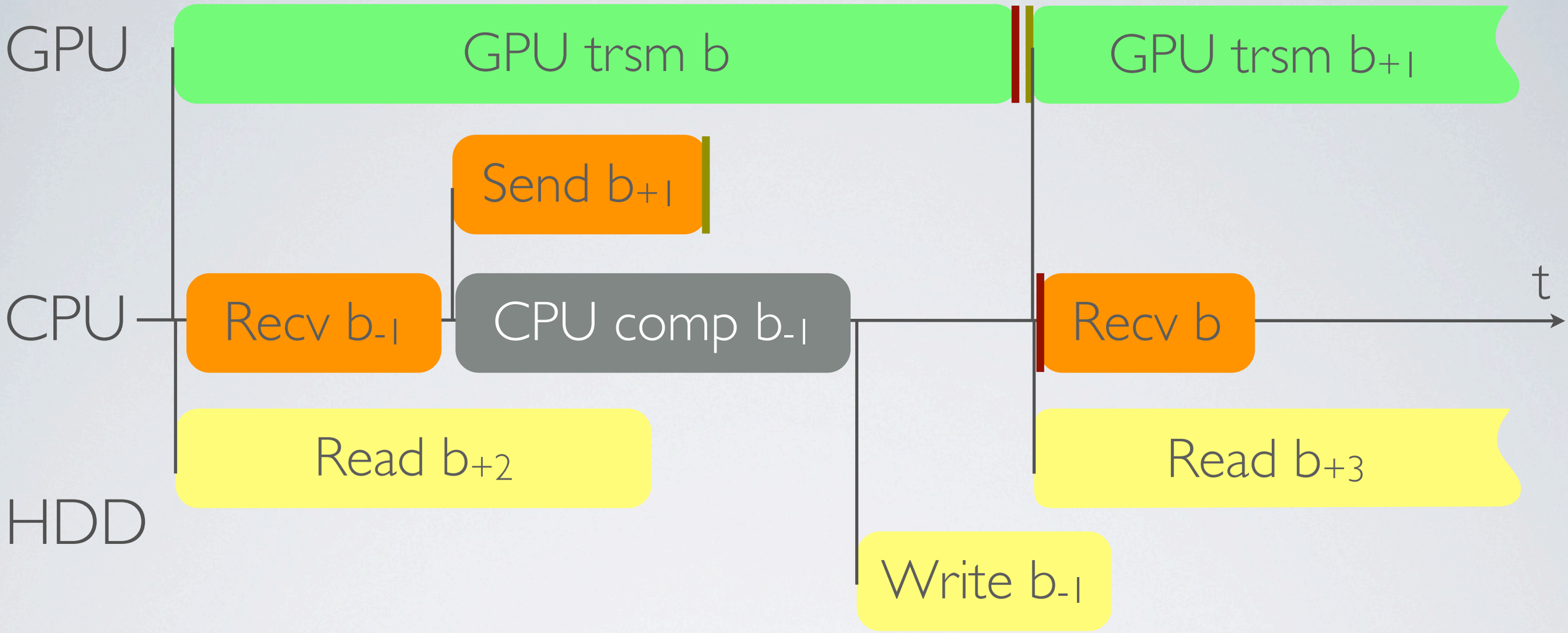
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

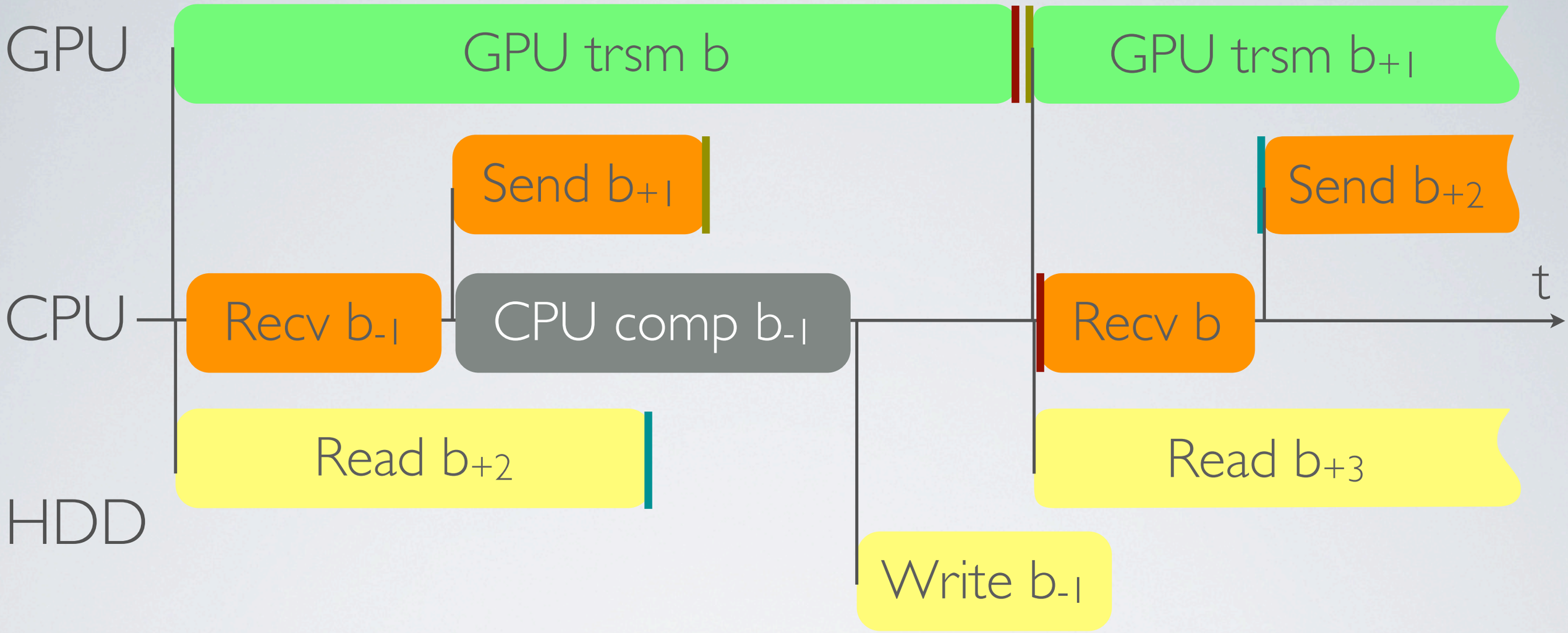
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

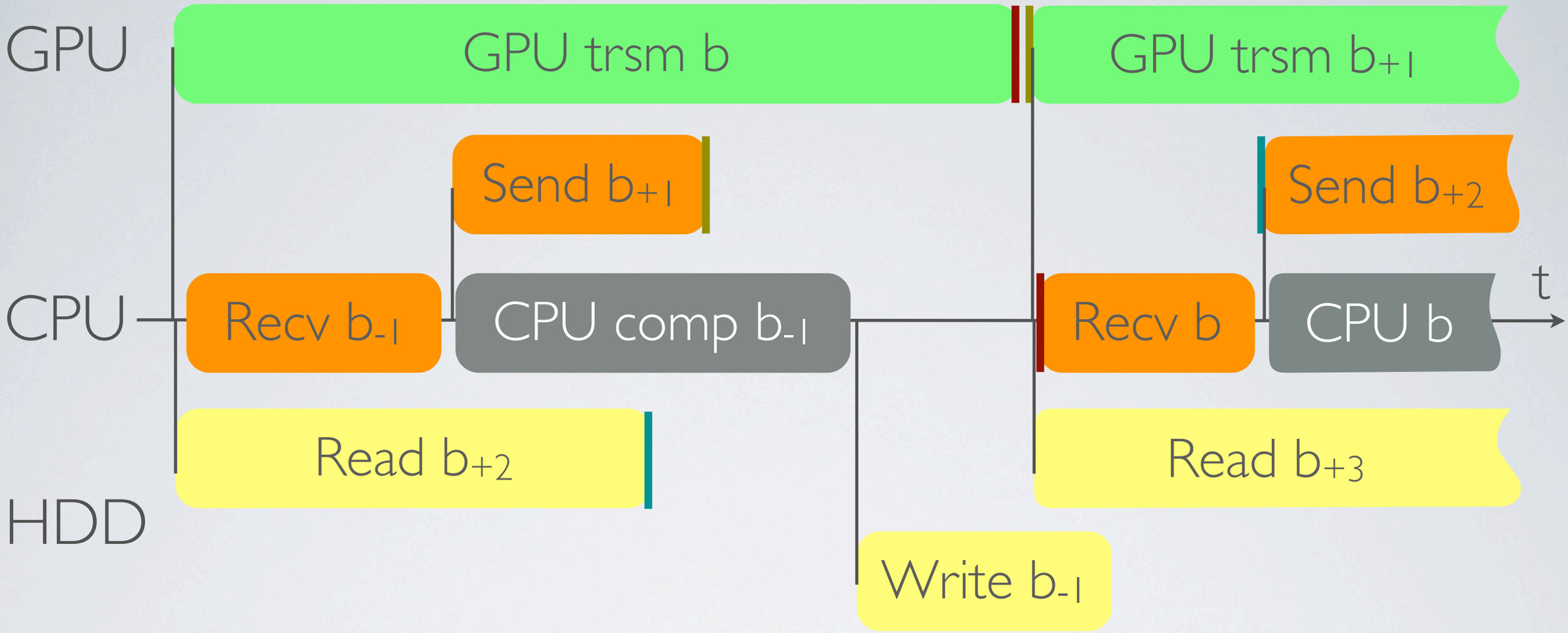
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



TIMELINE

Parallelism on the vertical axis

Heavy use of asynchronous dispatching

CPU ⇄ GPU transfer

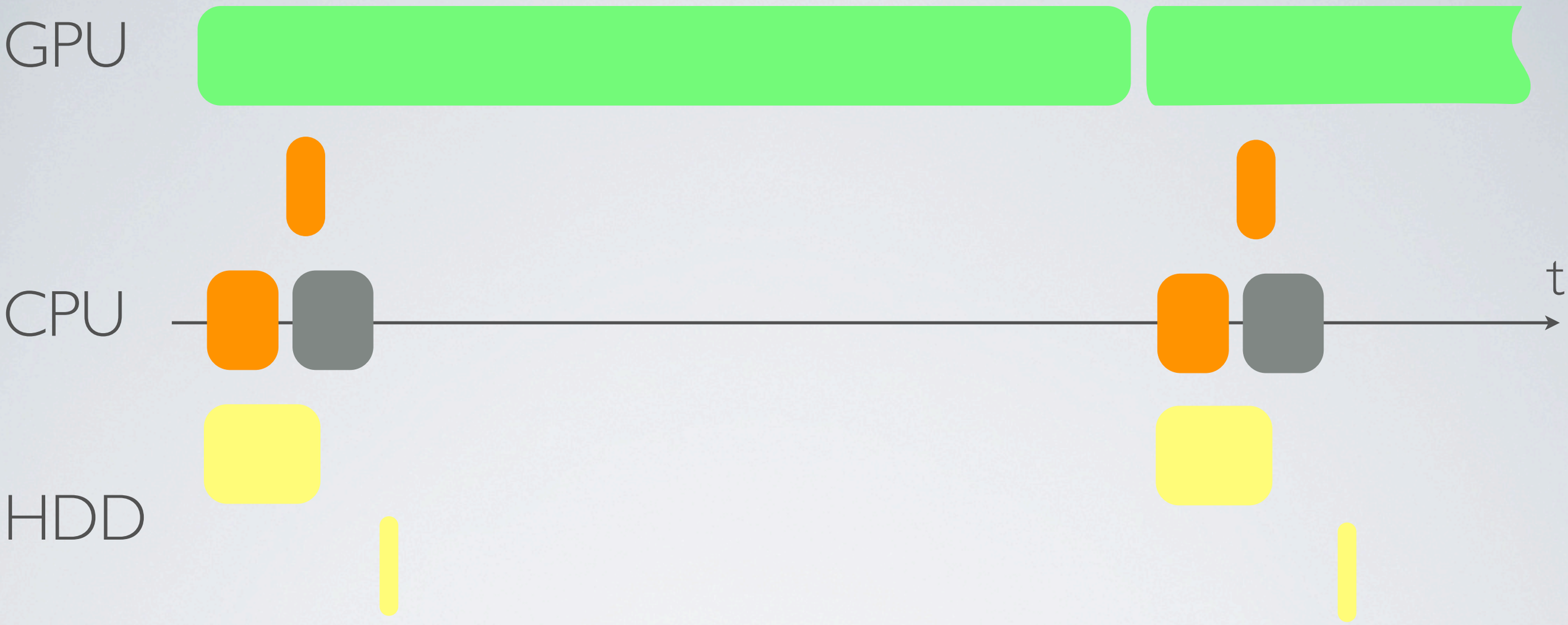
HDD ⇄ CPU transfer

GPU computation

CPU computation

Data dependencies

Asynchronous dispatch



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

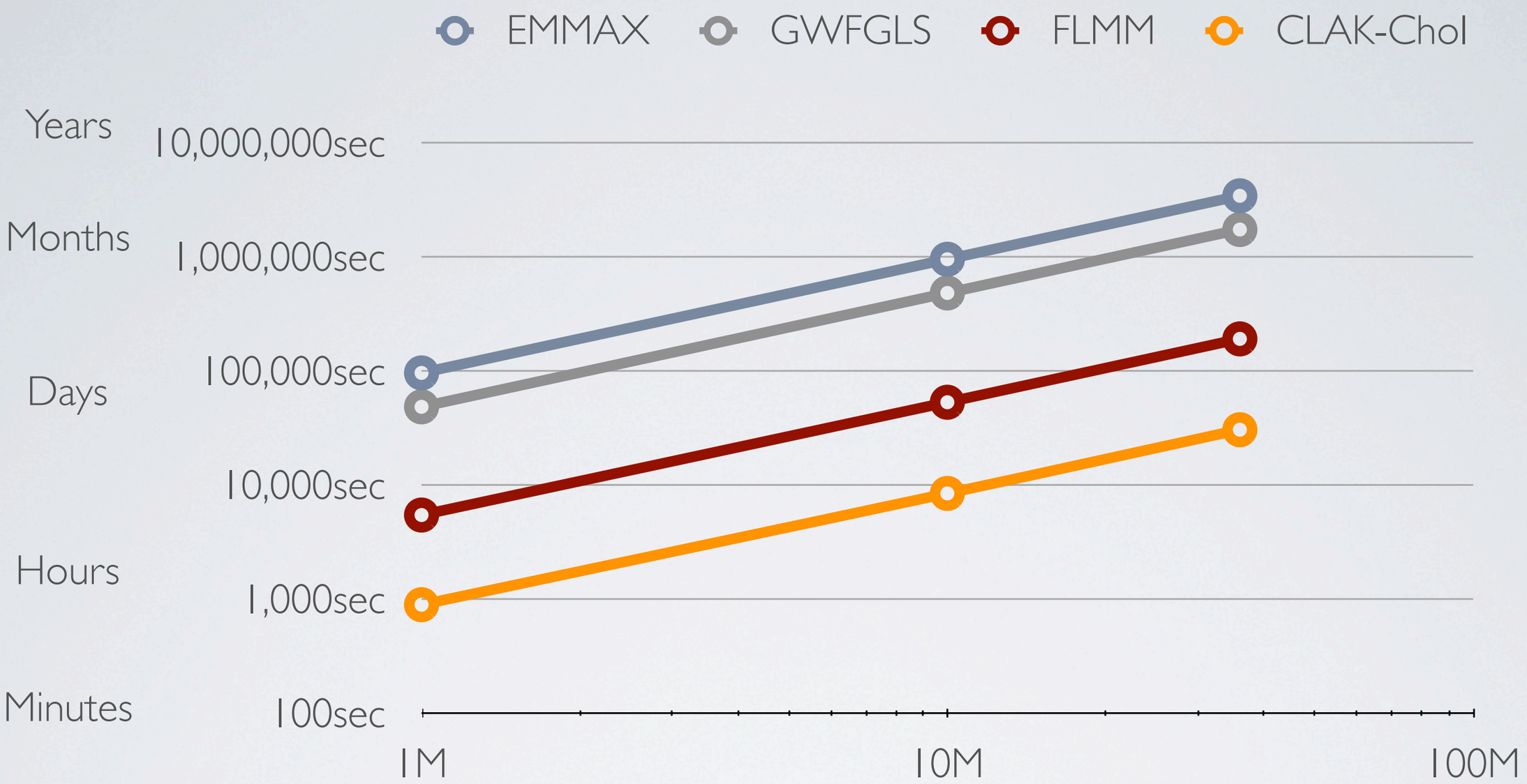
GPU computation

CPU computation

Blas: Intel MKL 10.2
Compiler: icc 12.1

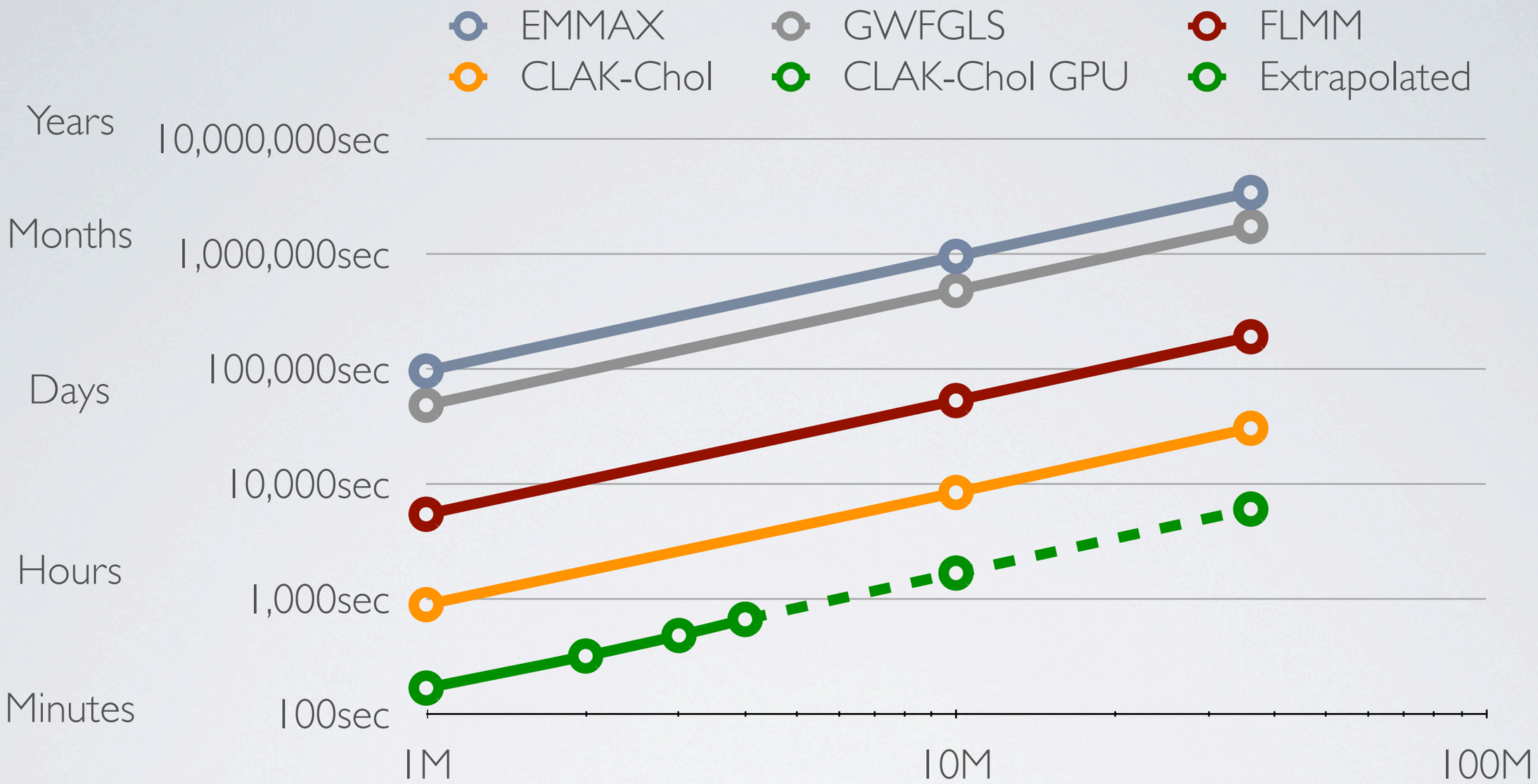
OUTLINE

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



PERFORMANCE

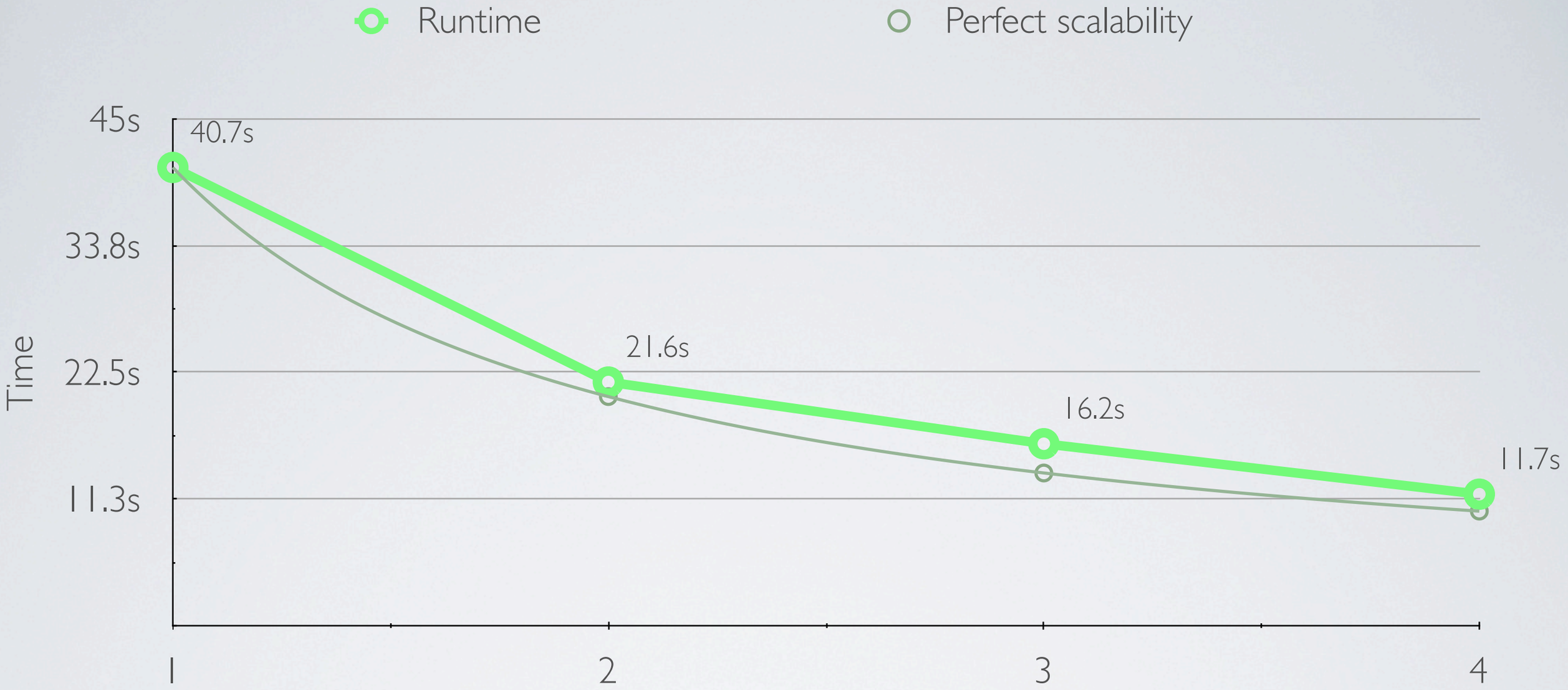
From years/months down to hours



PERFORMANCE

From years/months down to hours

12 CPU cores vs. 12 CPU cores + 4 GPUs



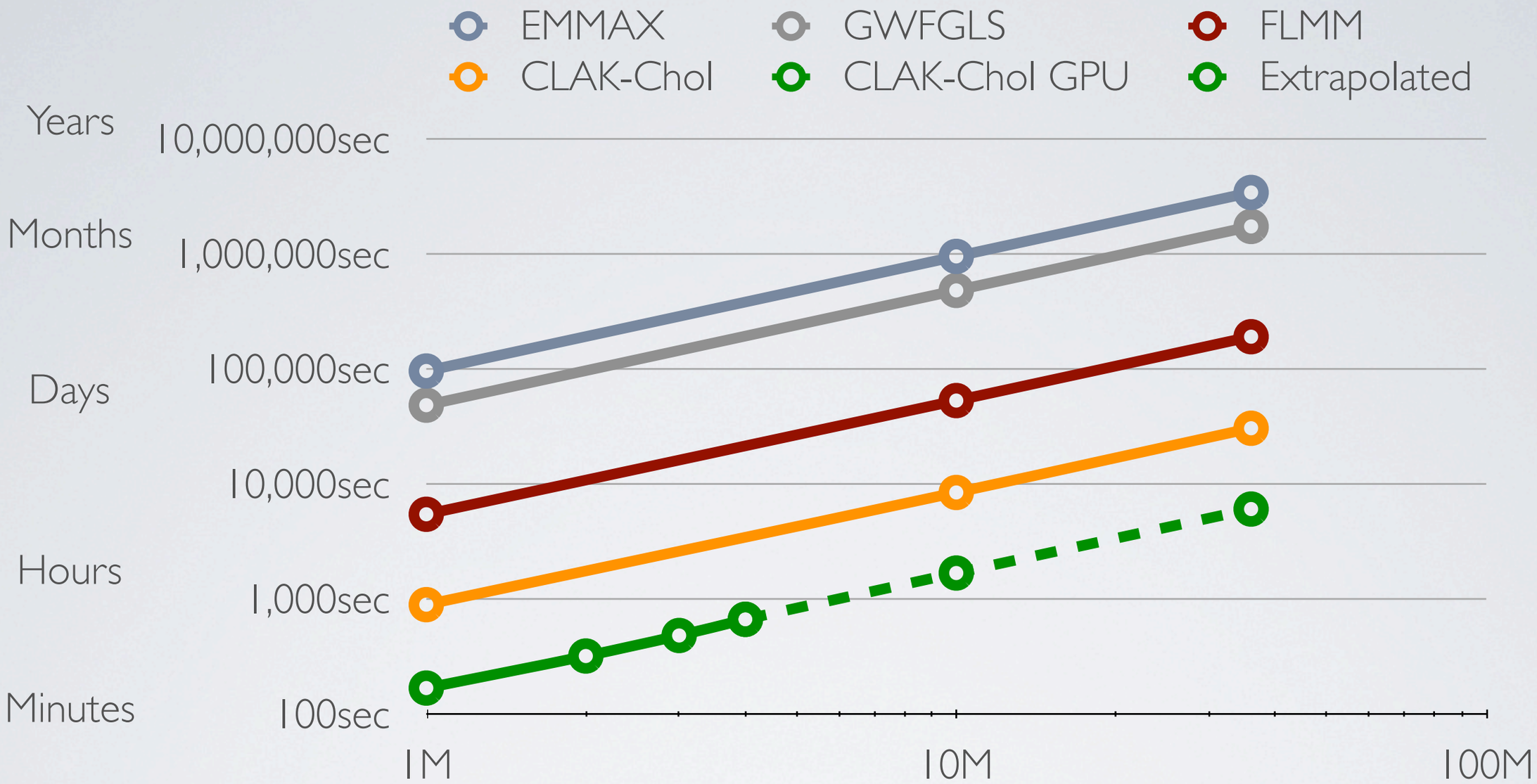
SCALABILITY

#GPUs x2 ⇒ time x0.54

Almost perfect

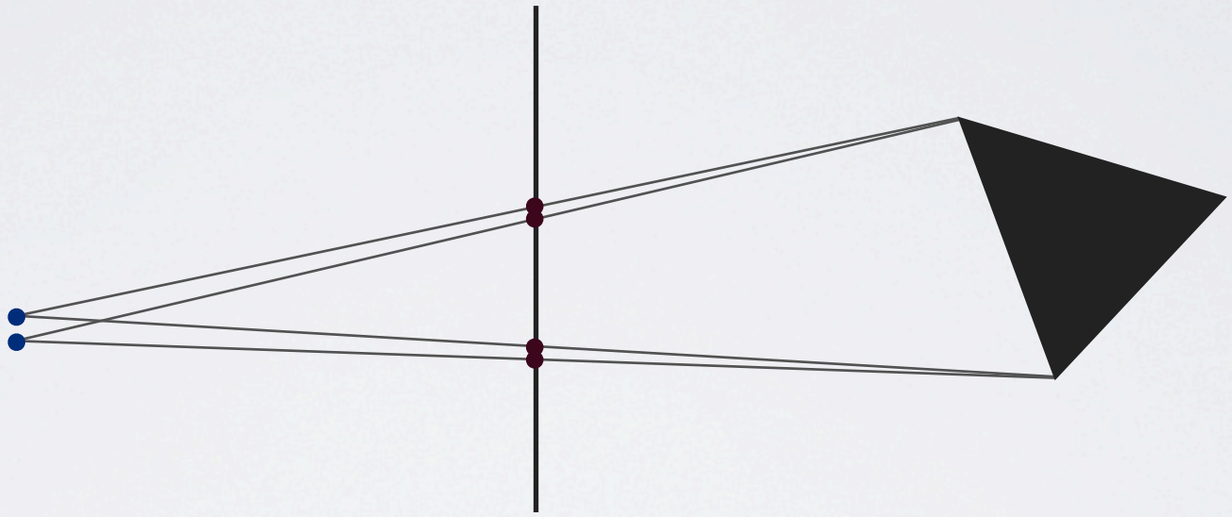
CONCLUSION

- ~2.6x Speedup per *additional* GPU (~10x for 4 GPUs)
- Don't replace the CPU by GPU; combine them
- Hide data transfer latency by overlapping with computation
  - Double/triple-buffering: GPU never stops computing
- one GPU order of magnitude faster?
  - Faster than what?
  - Victor W. Lee et al. («Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU», 2010)

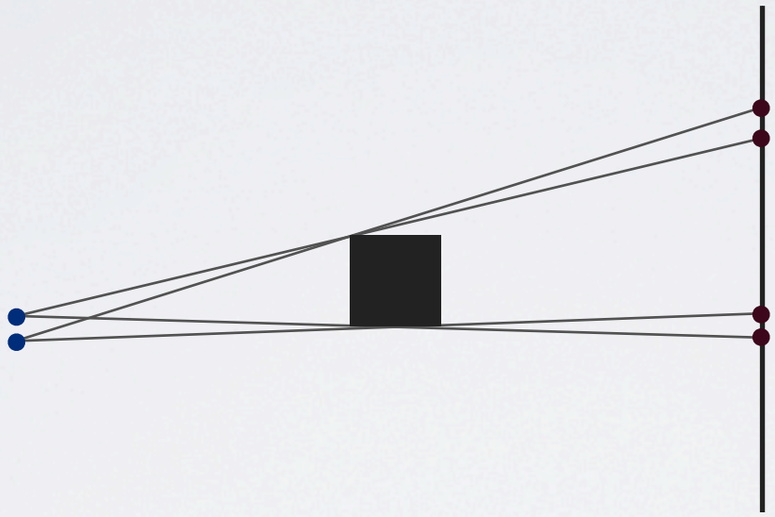


QUESTIONS?

lucasb.eyer.be

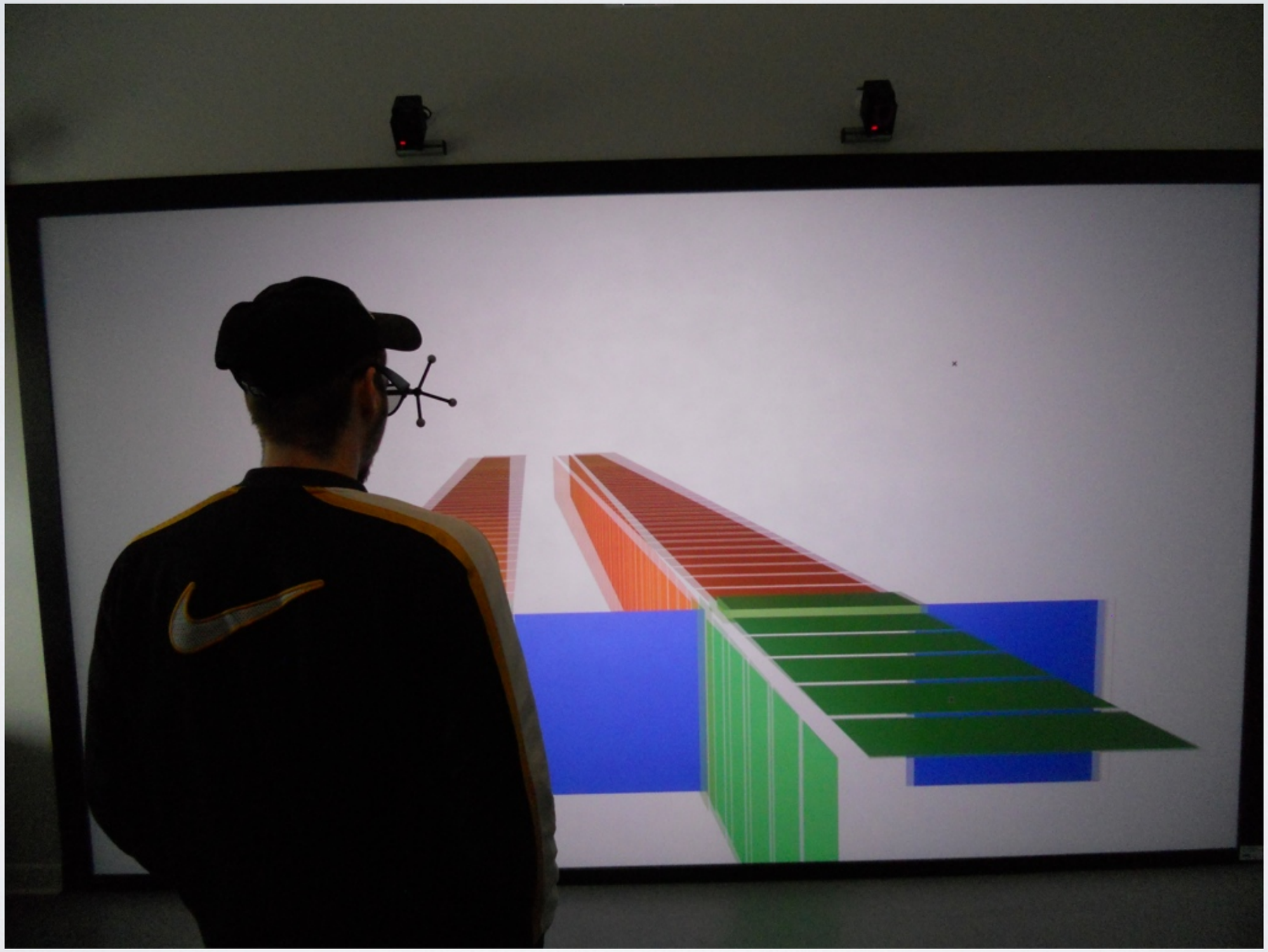


Viewer's eyes Projection surface 3D scene



Viewer's eyes 3D scene Projection surface

VISUALIZATION



FUTURE WORK

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