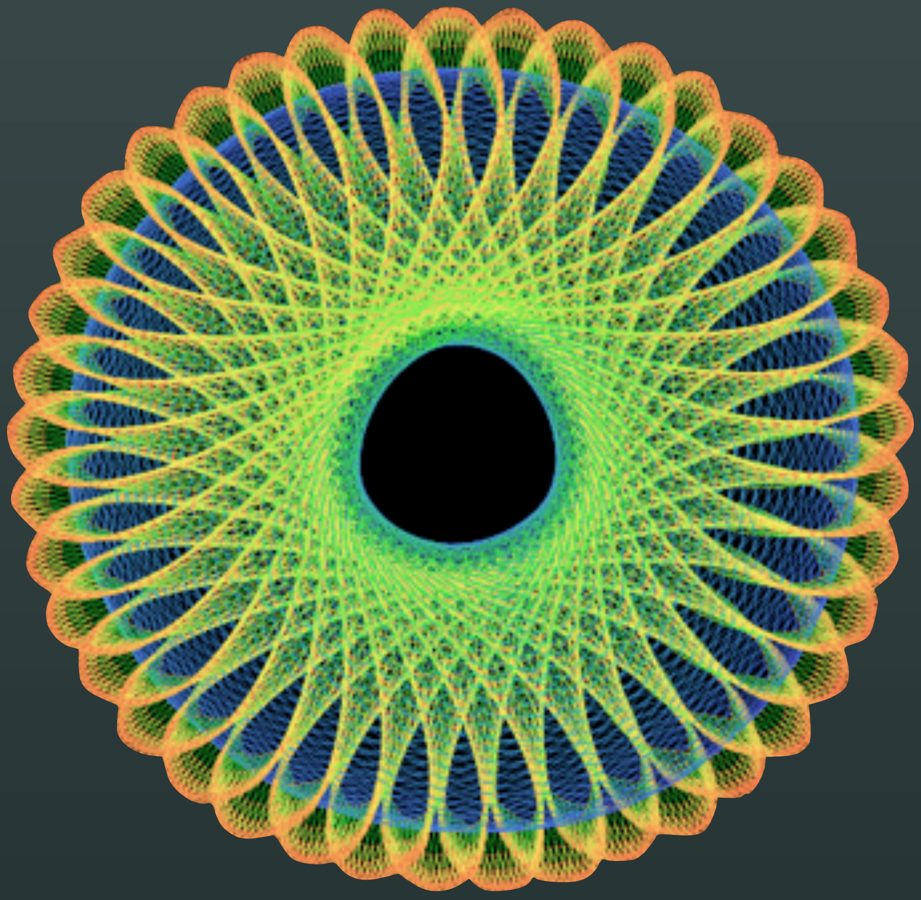
Hierarchical Diagonal Blocking

and Precision Reduction Applied to Combinatorial Multigrid



Kanat Tangwongsan

Carnegie Mellon University

Joint work with Guy Blelloch (CMU), Ioannis Koutis (CMU), and Gary Miller (CMU)

**image: GHS_indef, 40,000 nodes, 120,000 edges, courtesy of Yifan Hu*

Scalable Parallel SpMV Using

Hierarchical Diagonal Blocking

and Precision Reduction Applied to Combinatorial Multigrid

Sparse-Matrix Vector Multiply

the SpMV kernel

$$\begin{bmatrix} & & \\ & \textbf{sparse} & \\ & & \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Matrix ***A*** Vector ***x***

Compute *Ax, fast in parallel (shared memory)*

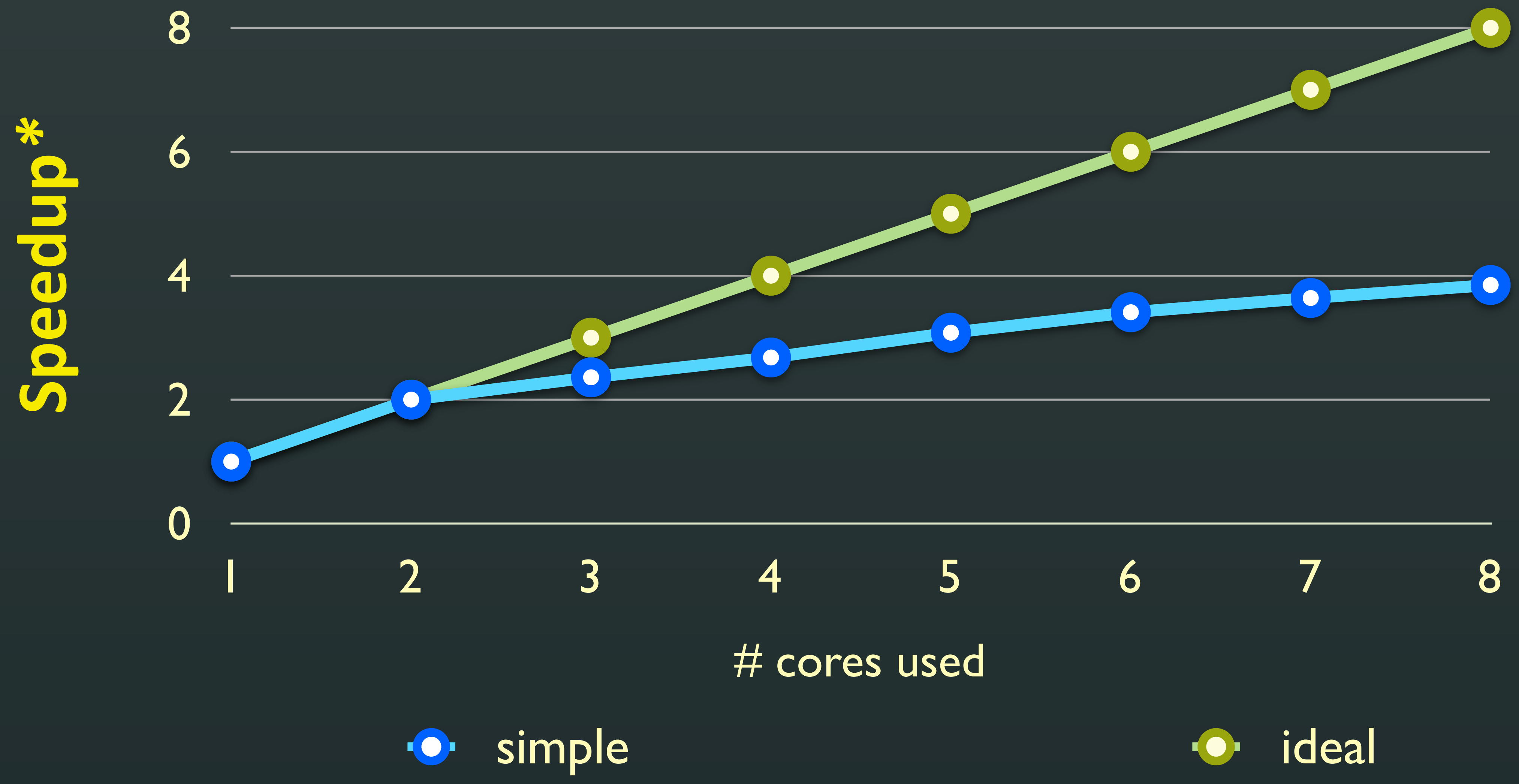
Numerous applications: iterative linear solver, eigenvalue, page rank, SVD, interior-point methods, ...

Problem:

Sparse matrix-vector product is **slow!**

Simple SpMV: for all rows, **in parallel**, compute $A_i x$

1M rows, 71M *nnz* ...**plenty of parallelism**



**Intel Nehalem X5550 2.66Ghz (8 cores), 8-core Bandwidth: 27.9 GBytes/sec*

Common observation:

Memory bandwidth is the limiting factor

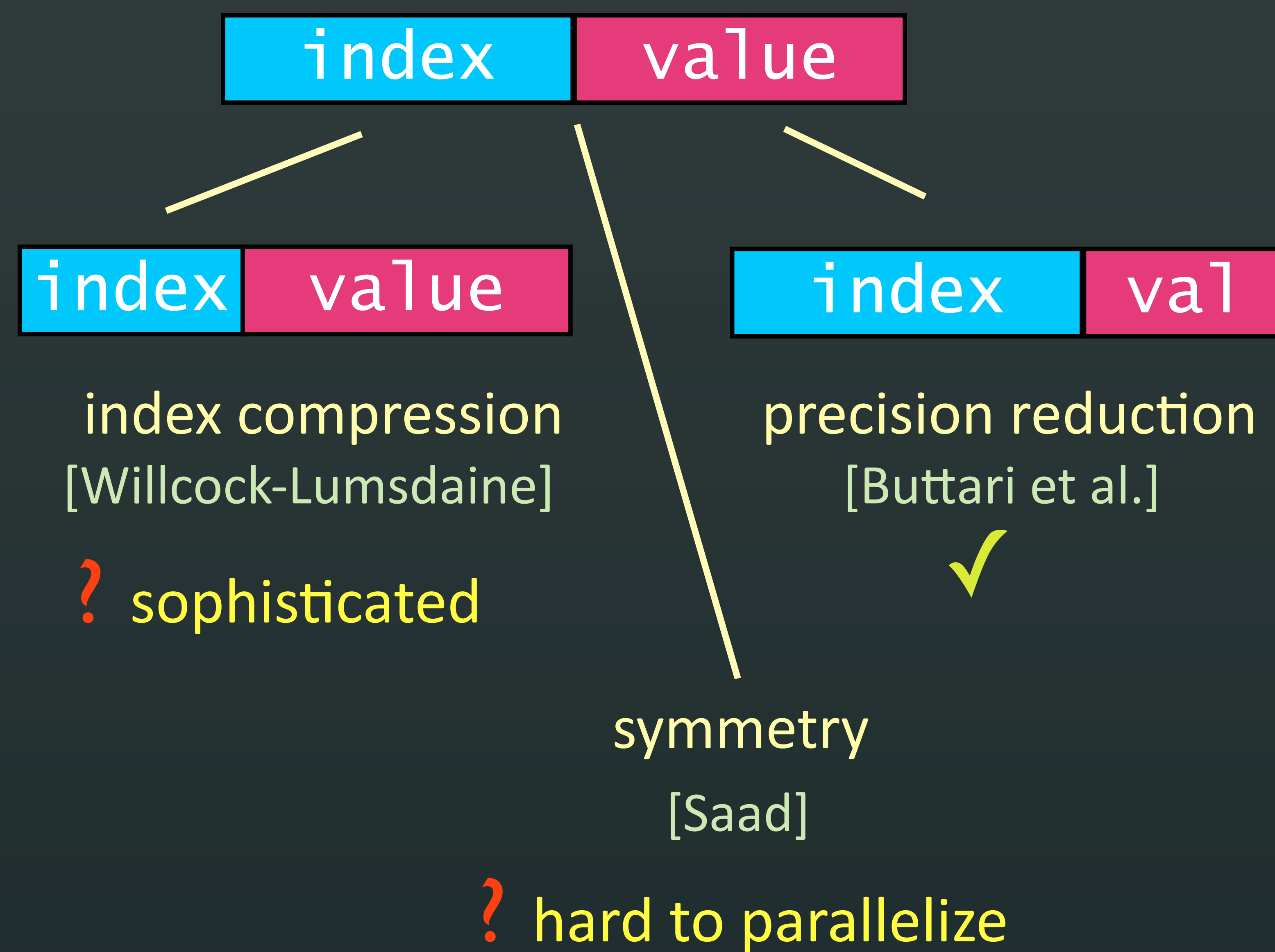
Memory bandwidth is the limiting factor

Previous Work:

enhance locality

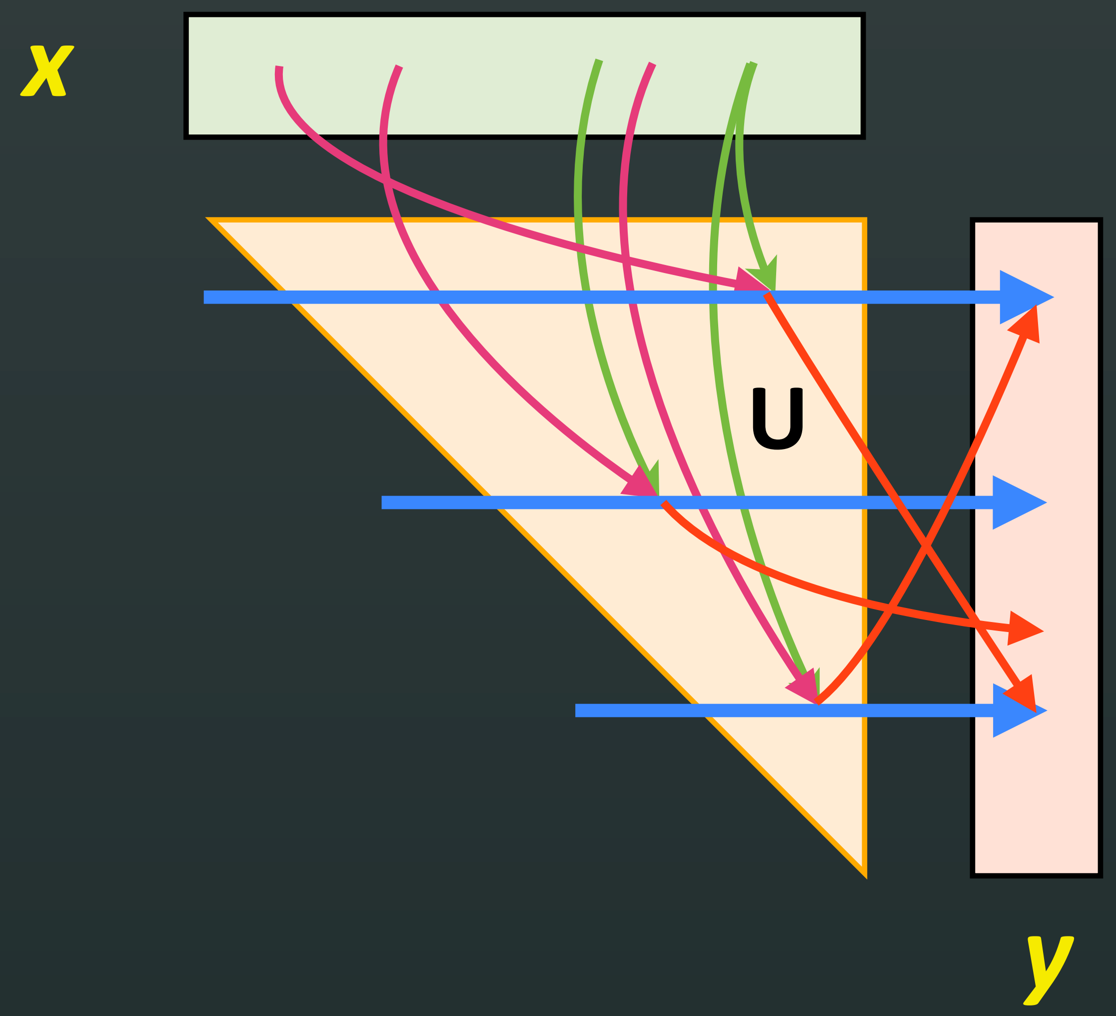
✓ row/column reordering
[Oliker et al., Pothen et al.]

✓ cache blocking
[Im et al., Williams et al.]



Symmetric Form: **Not Naturally Parallelizable**

To compute $y = Ax$



Concurrent Read

Concurrent Write

This Work:

Is there a **simple parallel algorithm** that offers the benefits of these optimizations using a **single, simple representation**?

HDB: Hierarchical Diagonal Blocking

can take advantage of

row/column reordering

index compression ← simplified

cache blocking

symmetry ← made possible w/o locking

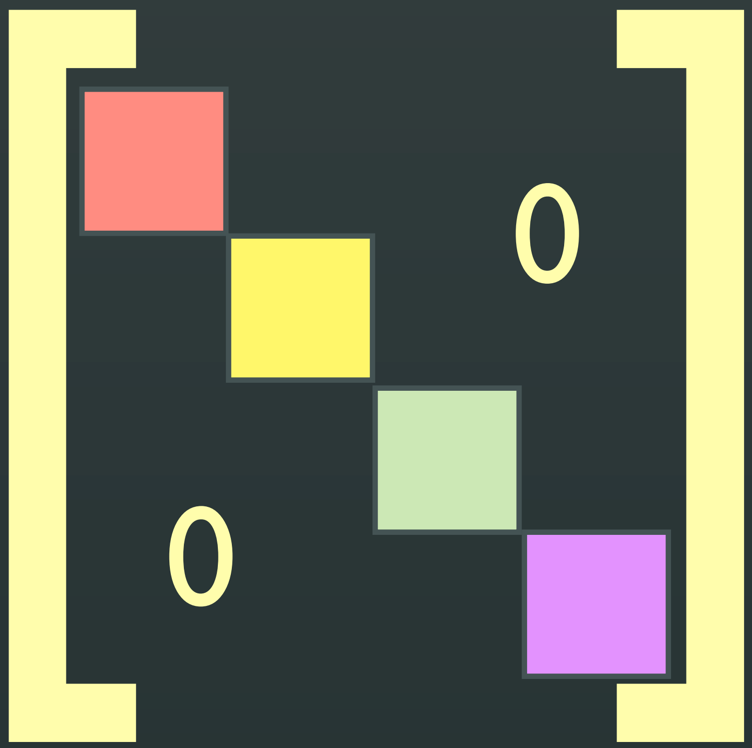
\+

mixed precision **+ parallelism**

in a single representation

A Simple Example

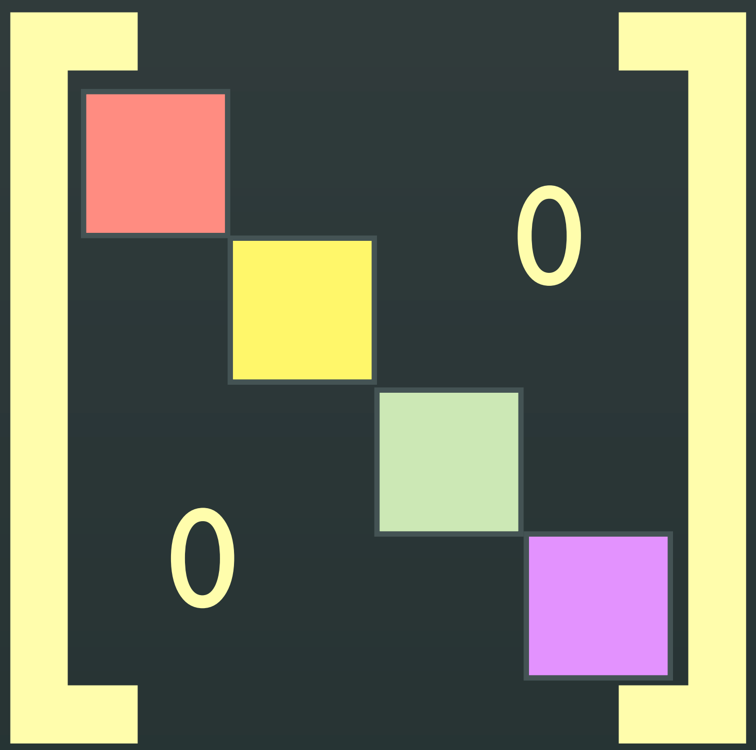
If a matrix can be ordered...



- natural parallelism
- good cache locality
- index compression
- symmetric form

But, matrices aren't always nice!

Decompose A into



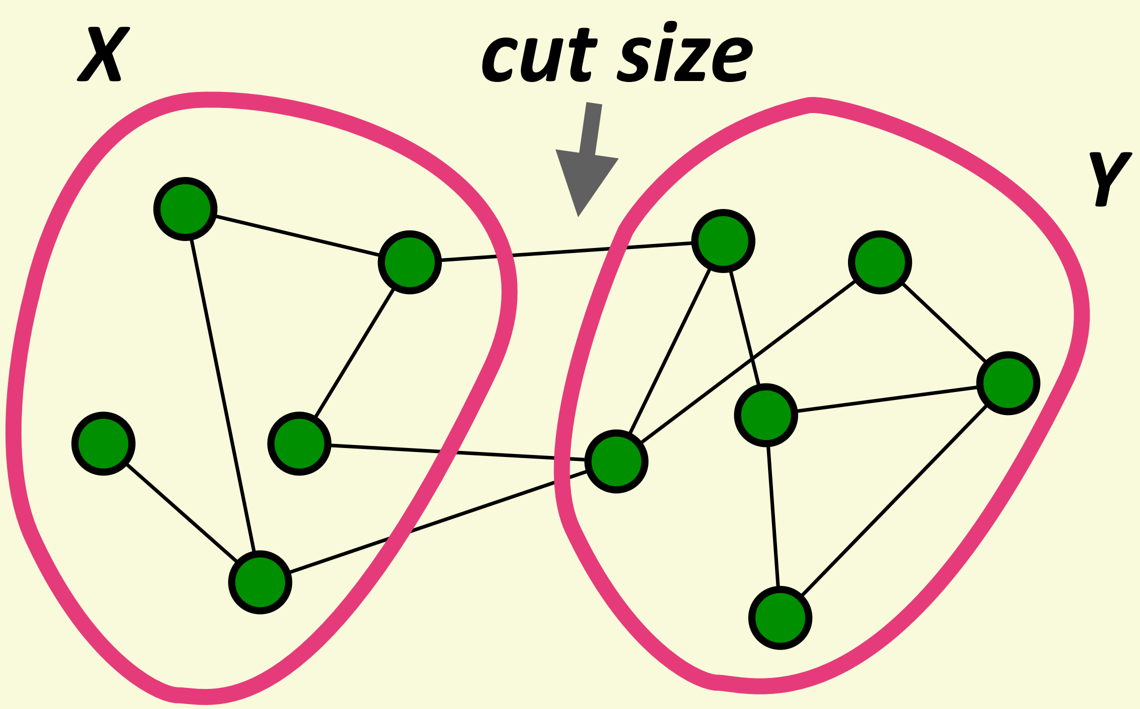
+ off diag.

Question: When can we decompose a matrix into diagonal blocks with only few off-diagonal entries?

Key Observation:

a surprising number of real-world graphs have **small separators**!

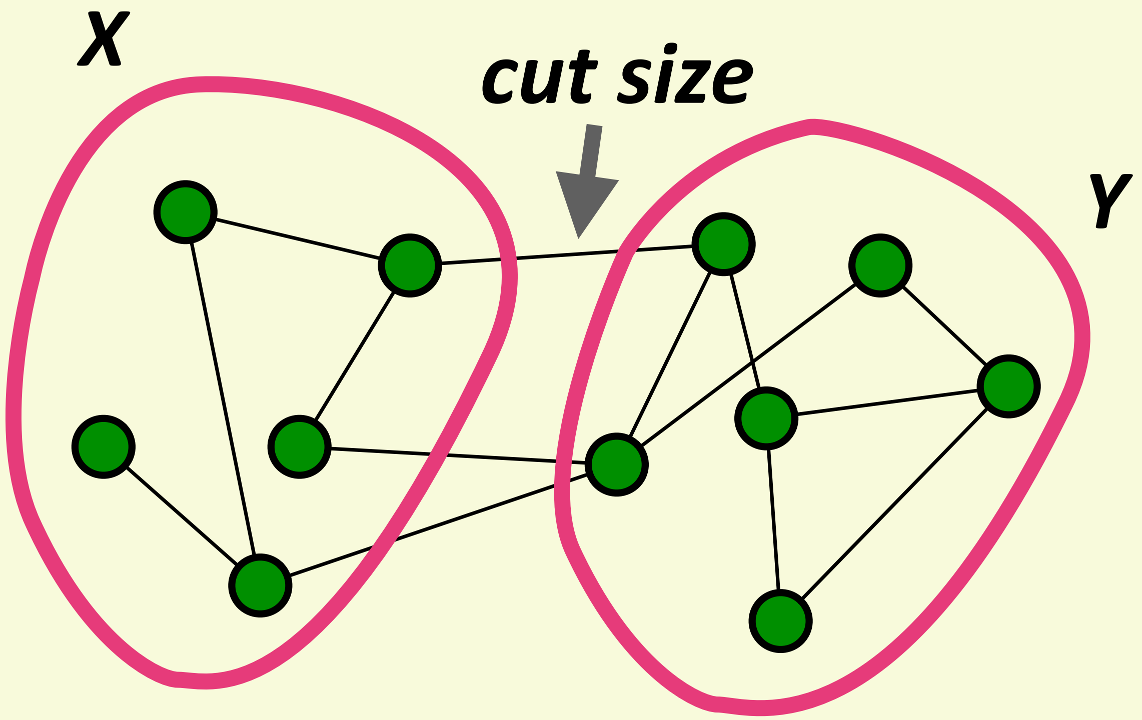
Graph ⇔ Matrix



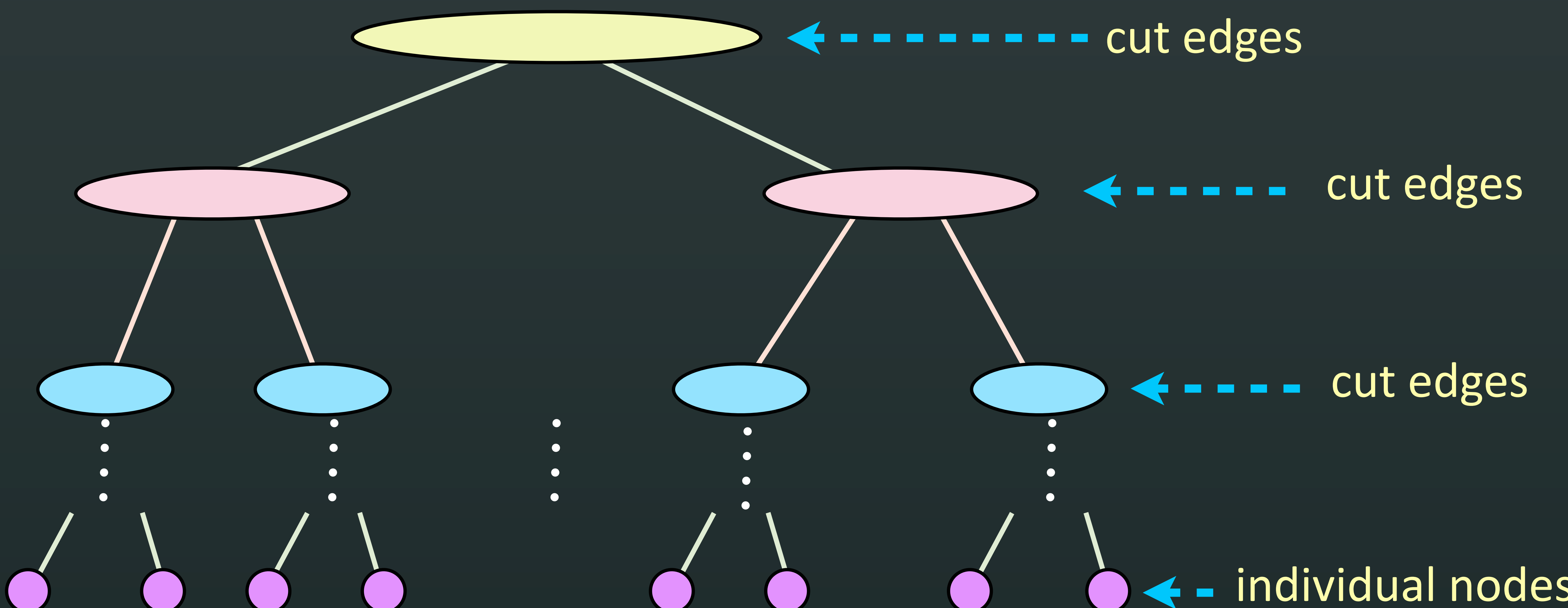
small separators:
can be recursively partitioned into roughly equal-sized parts with cut size $\leq O(n^{1-c})$

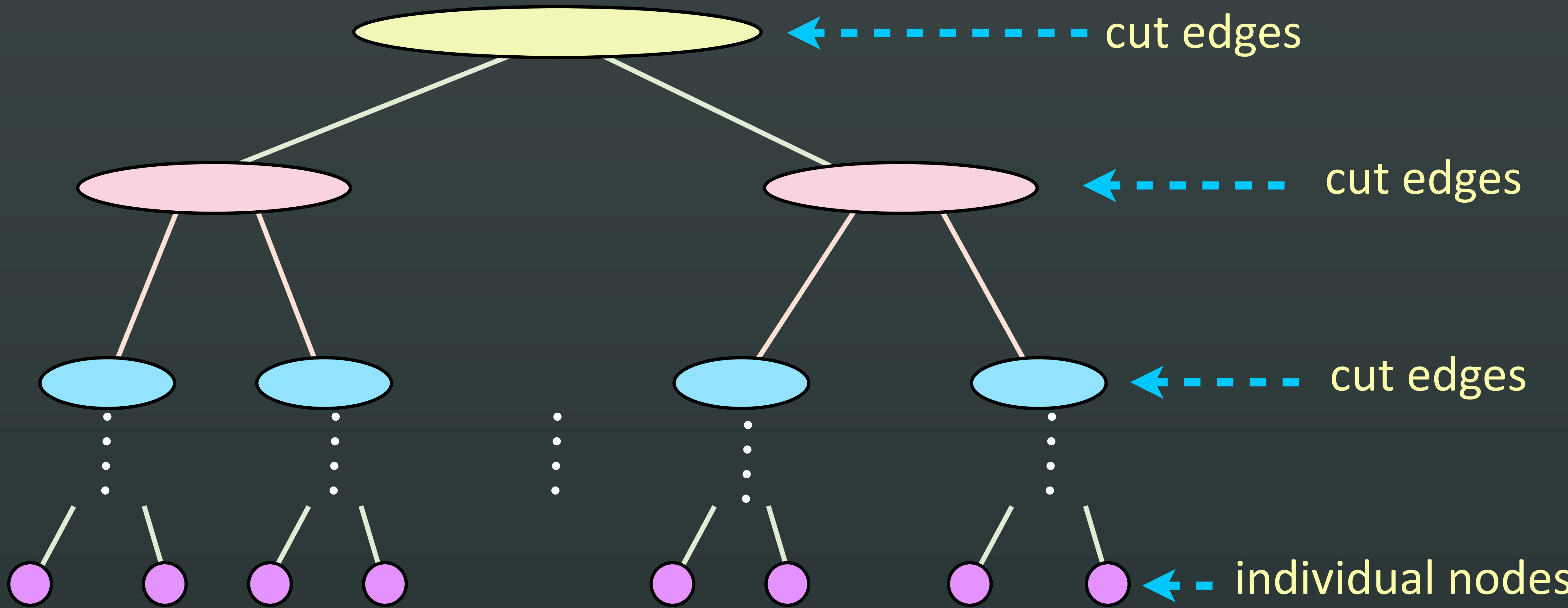
Examples: planar graphs, finite element meshes, google link graph, social networks, US road networks, etc.

e.g. [Ungar'51, Lipton-Tarjan'79, Blanford et al.'04]



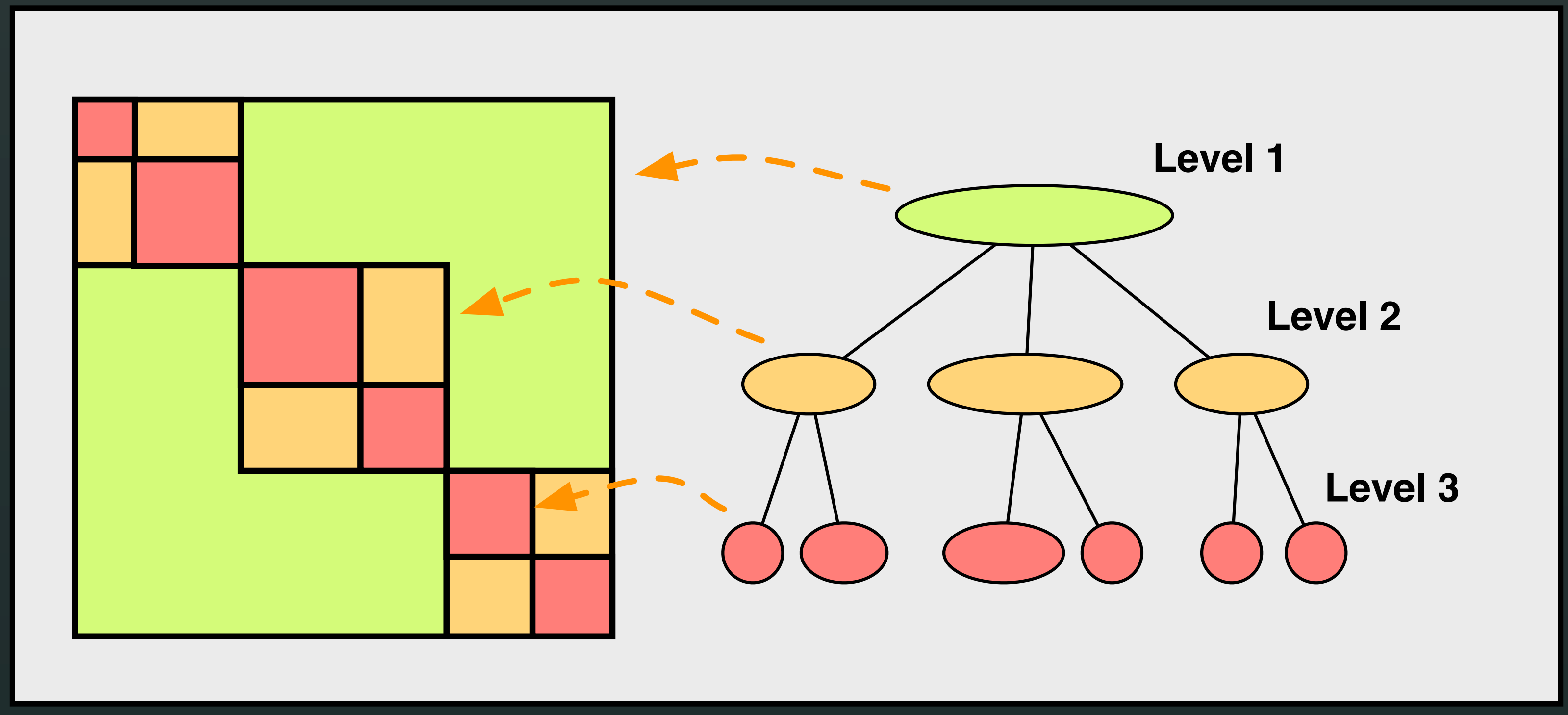
small separators:
can be recursively partitioned into roughly equal-sized parts with cut size $\leq O(n^{1-c})$





“separator tree” ordering

+ hierarchy of submatrices



HDB: Theoretical Guarantees

w - word size *B - block size* *M - cache size*

Theorem:

If an *n*-by-*n* matrix has small separators (n^{1-c}), then

(1) HDB $\#nnz + O(n/w)$ words

(2) Cache oblivious algorithm with misses at most

$$\#nnz/B + O(1 + n/(Bw) + n/M^c)$$

(3) Algorithm has polylog depth and is work efficient

This Talk: How does this perform in practice?

Experiments

large, sparse, symmetric matrices from various domains (> 1M non-zeros) e.g., FEM, vision (TV denoising)

Intel Nehalem X5550: **two** 4-core chips, 2.66Ghz

- How much bandwidth is saved?
- How does that translate into performance gain?
- What is the effect of separator quality?

Representation Footprint

How much could we save?

> 1.5x saving with blocking

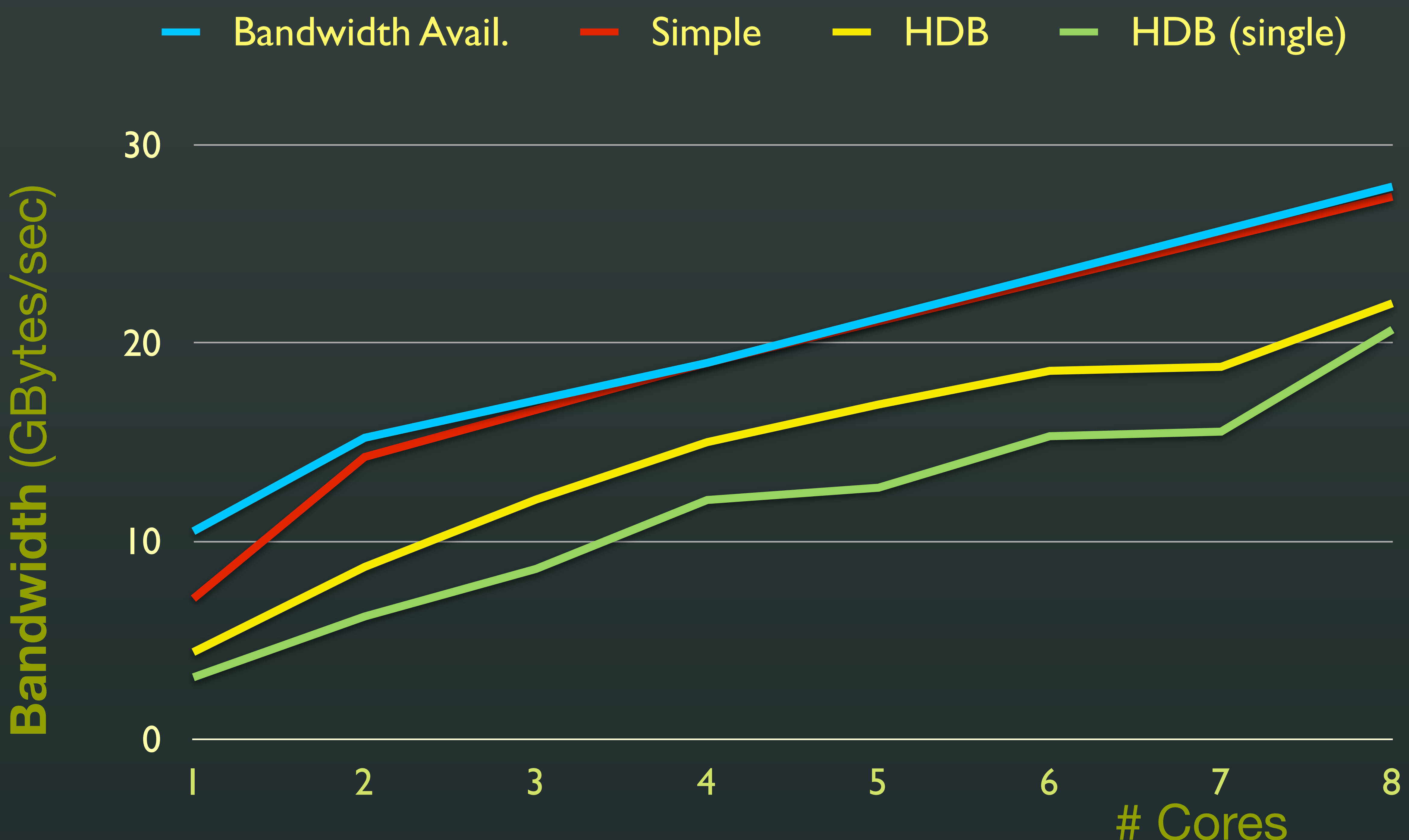
more (~3x) with precision reduction

Memory Access (MBytes)

Matrix	CSR/dbl	HDB/dbl	HDB/singl
2d-A (*1M rows, 50M nnz*)	80	56	36
3d-A (*1M rows, 69M nnz*)	103	67	43
af_shell10 (*1.5M rows, 53M nnz*)	657	313	193
audikw_1 (*.9M rows, 78M nnz*)	951	426	261
bone010 (*1M rows, 72M nnz*)	880	404	251
ecology2 (*1M rows, 50M nnz*)	80	56	36
nd24k (*72K rows, 29M nnz*)	346	164	106
nlpkkt120 (*3.5M rows, 97M nnz*)	1,212	589	367
pwtk (*3.5M rows, 97M nnz*)	143	65	40

Bandwidth Analysis

Matrix: bone010 - 1M rows, 72M nonzeros

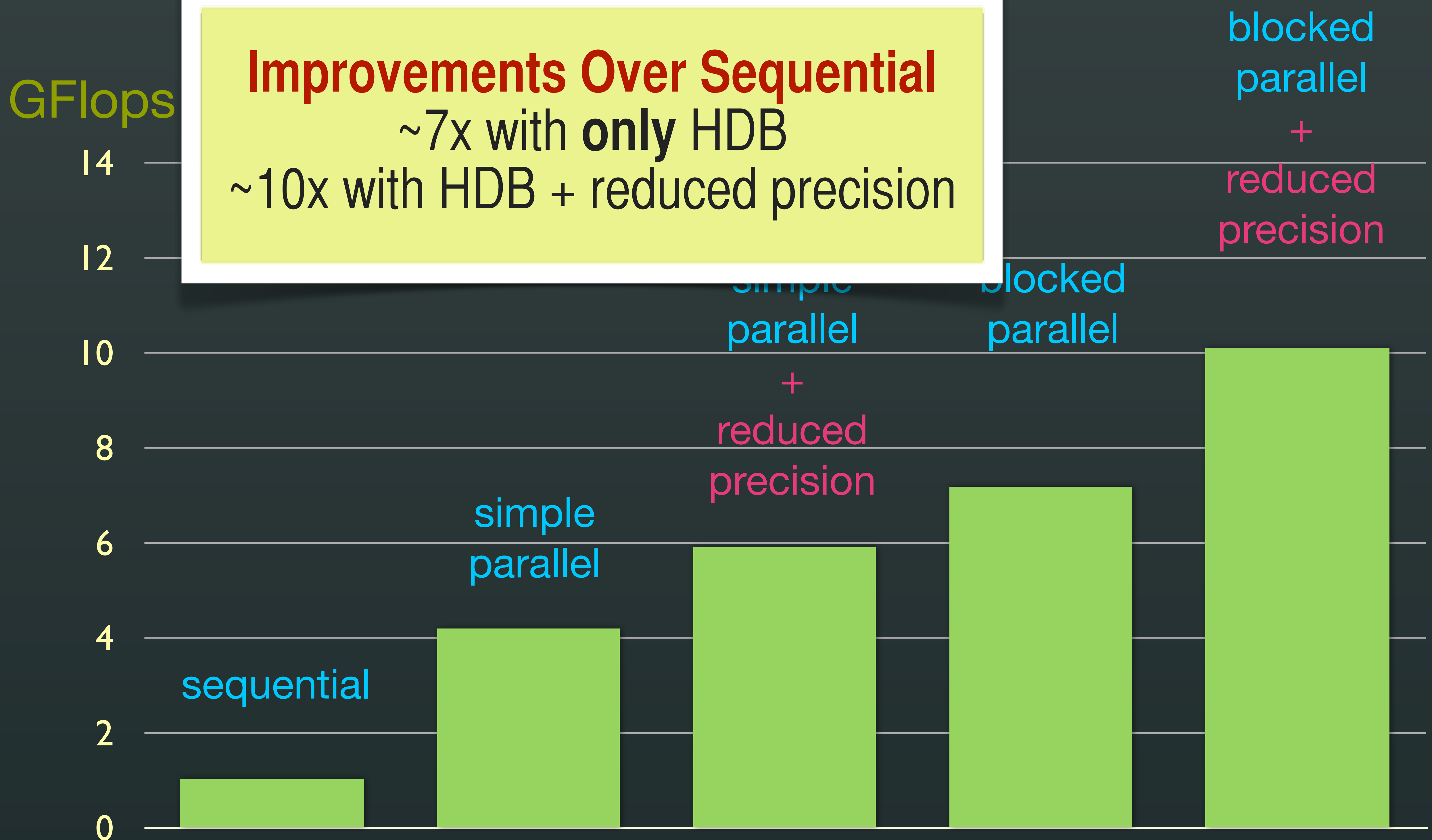


Performance Analysis: **Median**

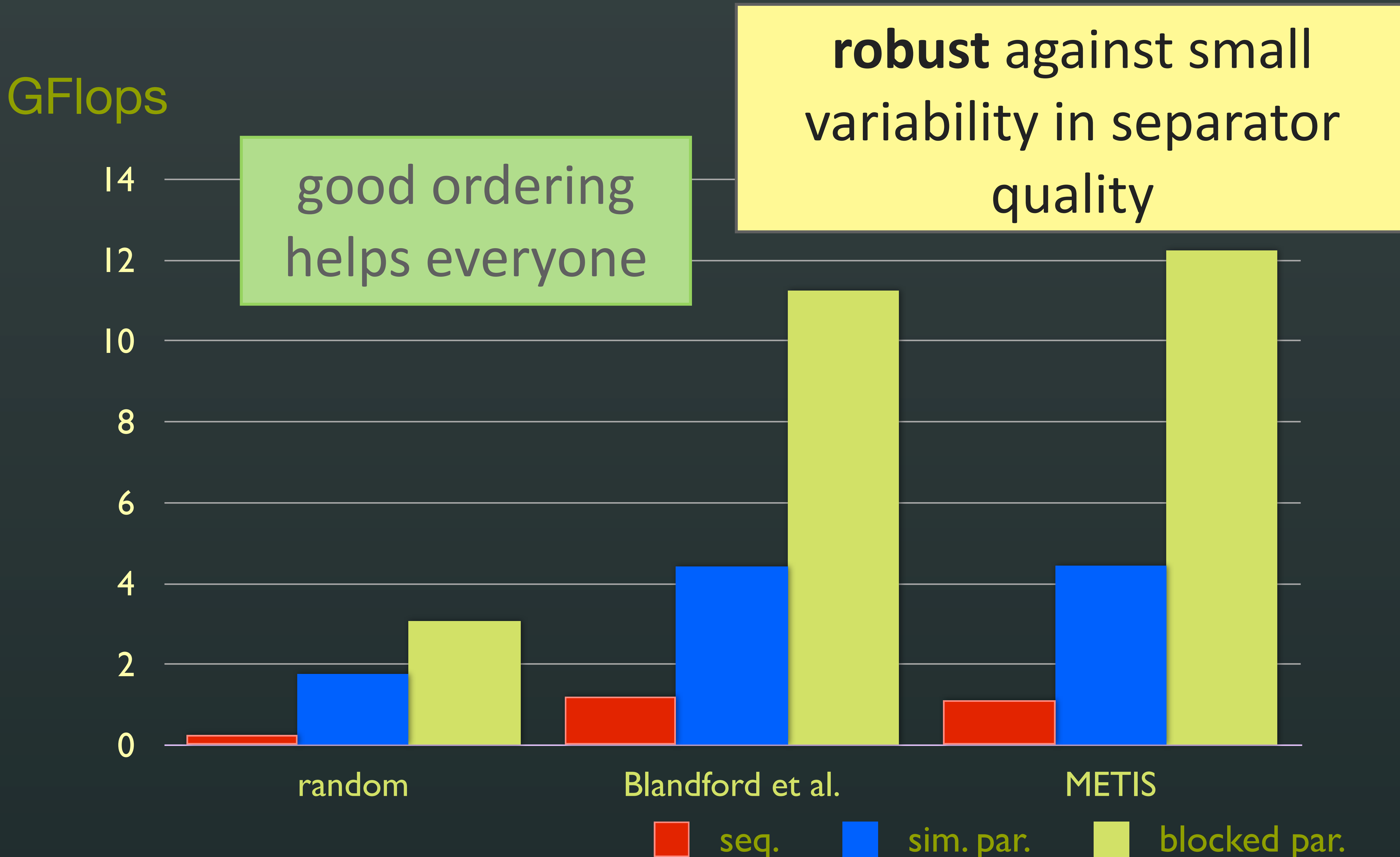
Improvements Over Sequential

~7x with **only** HDB

~10x with HDB + reduced precision



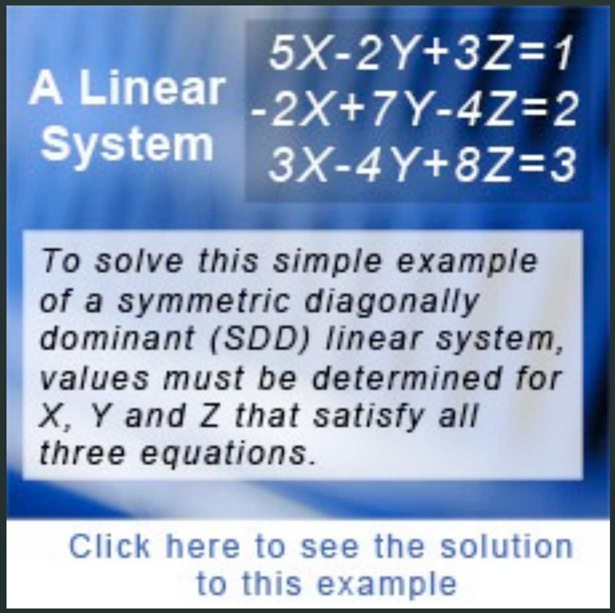
Effects of Separator Quality



**Matrix: audikw_1 - 1M rows, 78M nonzeros*

When is precision reduction viable?

low-precision “raw” data



use approximate answers in intermediate steps to derive full-precision final solutions

Combinatorial Multigrid (CMG) Solvers

symmetric diagonally dominant (SDD) = each diagonal element is larger than the sum of the absolute values of other elements in that row

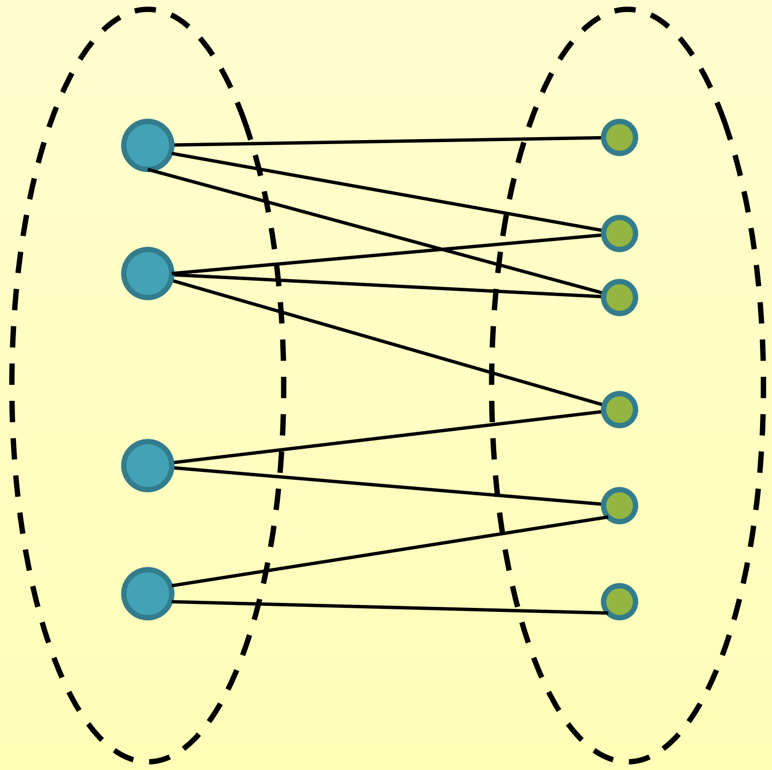
$$Ax = b$$

combinatorial preconditioning

SDD Problem Examples

Data Mining/Recommender

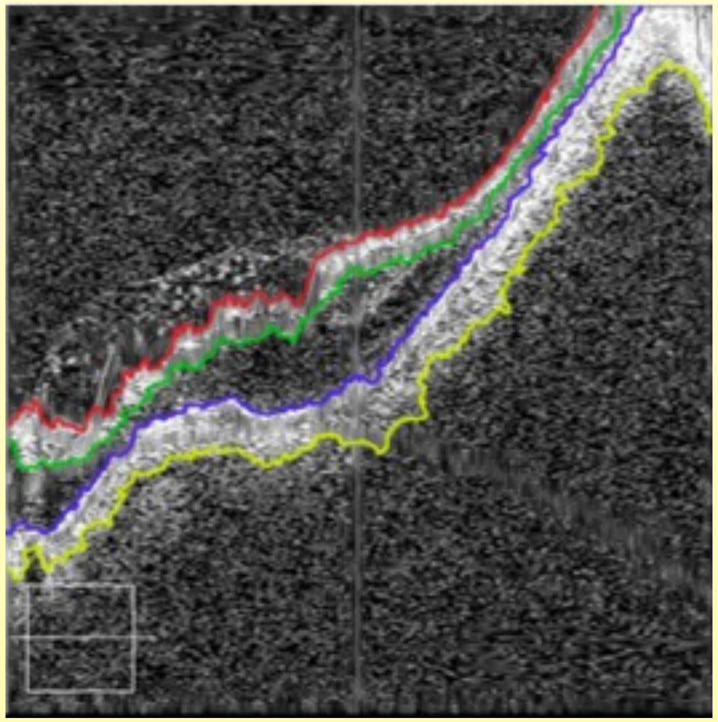
Compute electrical flow



Movie-Subscriber Graph

Optical Coherence Tomography

Compute few eigenvectors

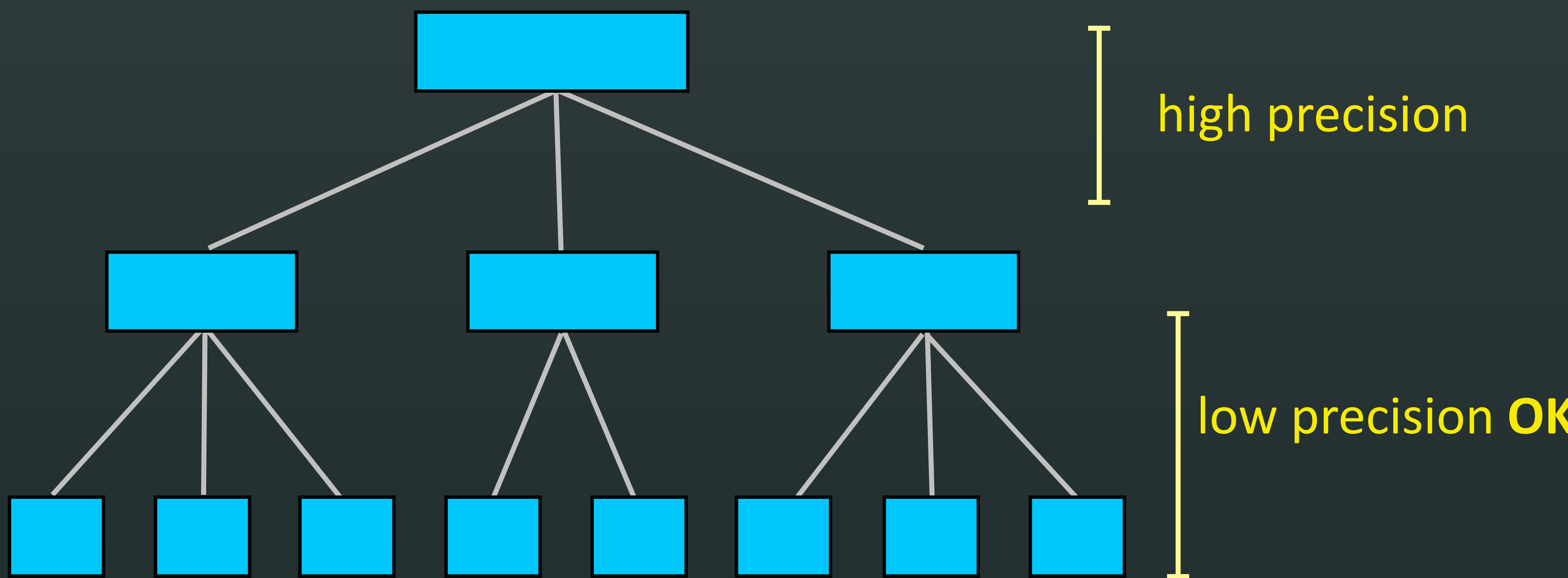


Retina Image

CMG Overview

Improvements Over Seq.
> 7x on 8 cores (max)
5.2x (median)

- hierarchical/recursive solver
- most work: SpMV + vector-vector ops



Take-Home Points

Thank you!

- Memory Bandwidth Bottleneck
- Hierarchical Diagonal Blocking (HDB)
  - simple, compact, cache-friendly
- CMG using Low-Precision Guide
  - full-precision answer from low-precision hints