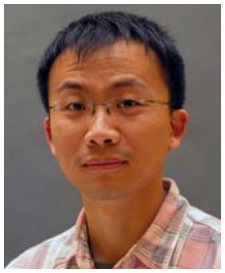
Scalable algorithms for kernel matrix approximation

with

Chenhan Yu, Bill March, and Bo Xiao





GEORGE BIROS

padas.ices.utexas.edu

INSTITUTE FOR COMPUTATIONAL ENGINEERING & SCIENCES

THE UNIVERSITY OF TEXAS AT AUSTIN

Kernel matrices

Input

N points in $\mathbb{R}^d$: $x_1, \ldots, x_N$

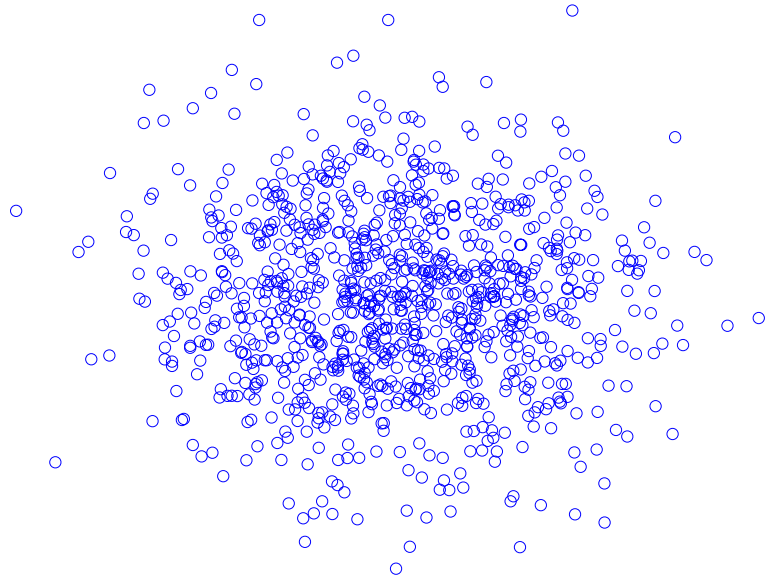
N densities in $\mathbb{R}$: $w_1, \ldots, w_N$

Output

N potentials in $\mathbb{R}$: $u_1, \ldots, u_N$

$$u_i = \sum_{j=1}^{N} G(x_i, x_j) w_j$$

$$G(x_i, x_j) = \exp\left(-\frac{1}{2}\frac{\|x_i - x_j\|_2^2}{h^2}\right)$$



Gaussian	$\exp(-\|x - x_j\|^2/(2h^2))$
Laplace	$\|x - x_j\|^{2-d},\ d > 2$
Matern	$\left(\sqrt{2\nu}\|x - x_j\|\right)^{\nu} K_\nu \left(\sqrt{2\nu}\|x - x_j\|\right)$
Polynomial	$\left(x^T x_j/h + c\right)^p$
Ornstein-Uhlenbeck	$\exp(-c\|x - x_j\|)$
Multiquadratic	$\sqrt{c^2 + \|x - x_j\|_2^2}$
Inverse multiquadratic	$1/\sqrt{c^2 + \|x - x_j\|_2^2}$

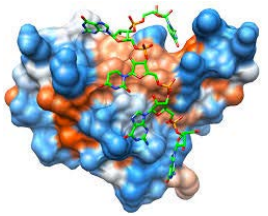
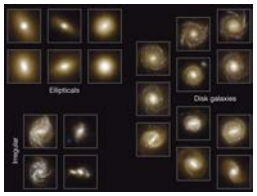
Applications

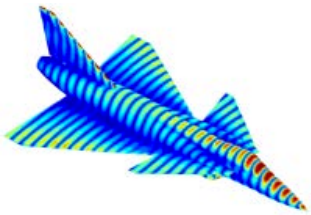
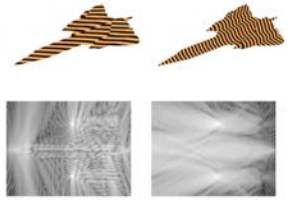
Simulation

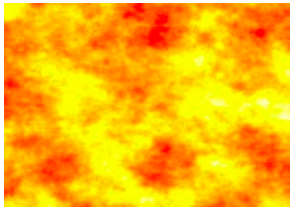
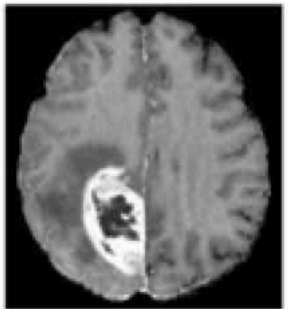
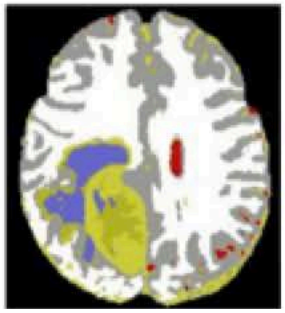
- Gravity & Coulomb
- Waves & scattering
- Fluids & transport

Data analysis

- Kernel methods in machine learning
- Approximation/Geostatistics
- Non-parametric statistics

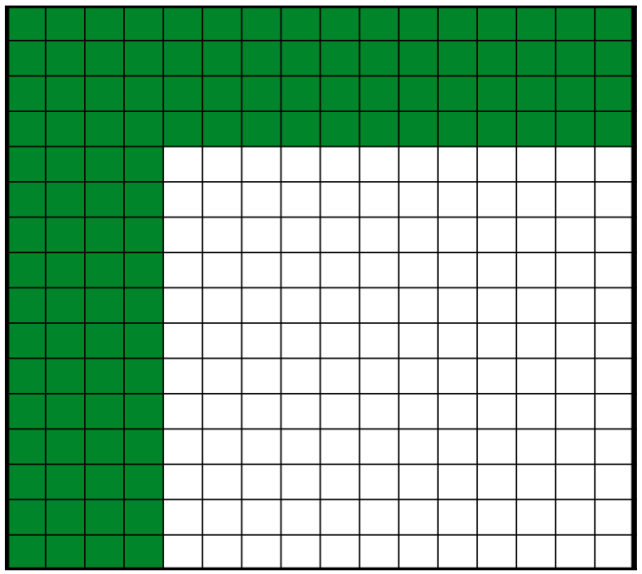



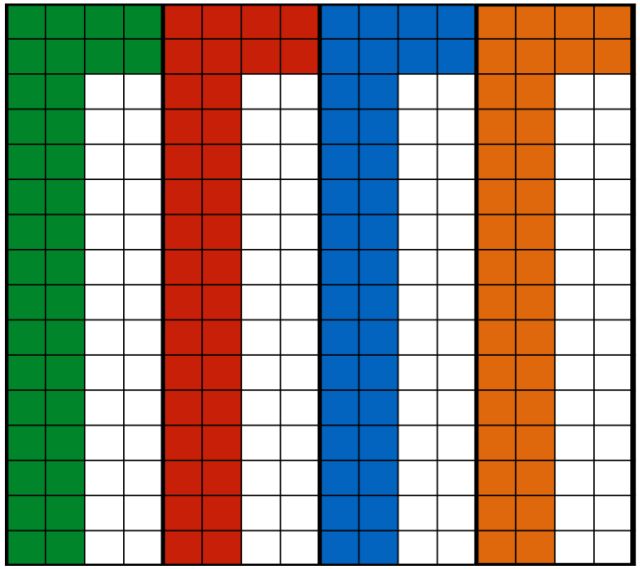
Computational challenges

- N points
  - N^2 work for matvec
  - N^3 work for factorization

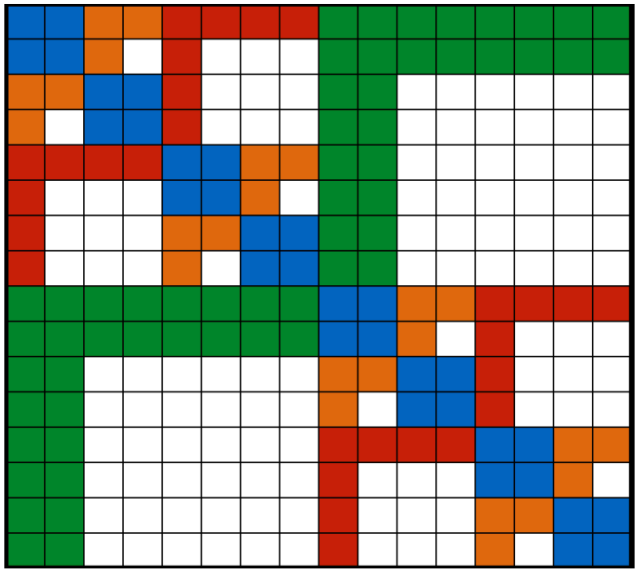
Achieving $O(N \log^a N)$ complexity



NYSTROM



ENSEMBLE NYSTROM



HIERARCHICAL MATRICES

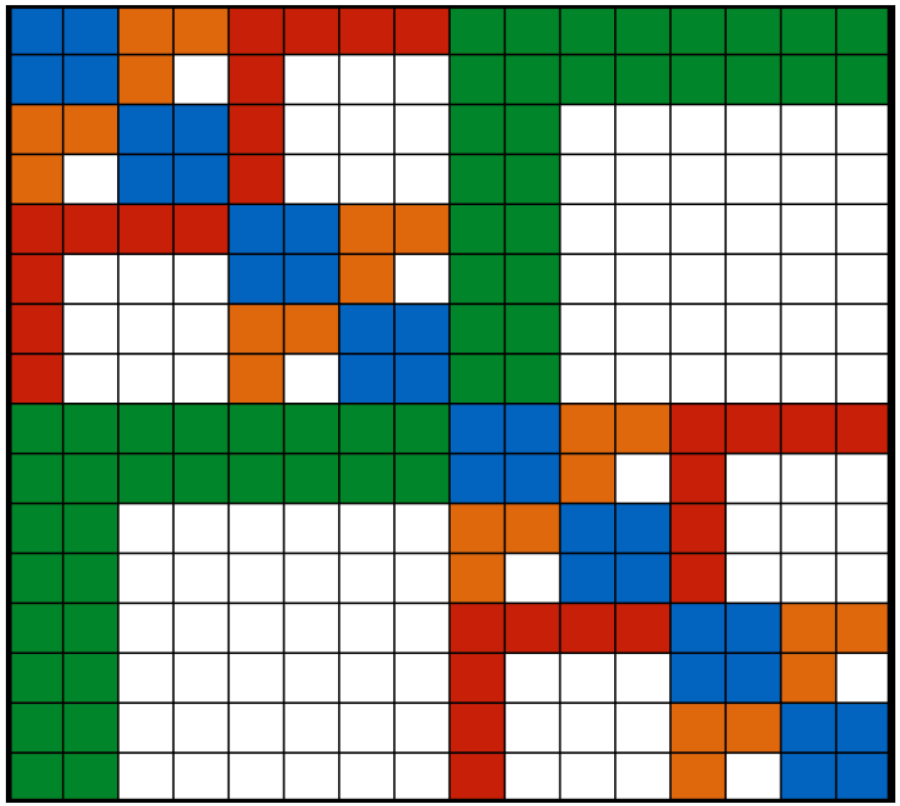
Hierarchical matrices, basic idea

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}$$

$$= \begin{bmatrix} G_{11} & 0 \\ 0 & G_{22} \end{bmatrix} + \begin{bmatrix} 0 & G_{12} \\ G_{21} & 0 \end{bmatrix}$$

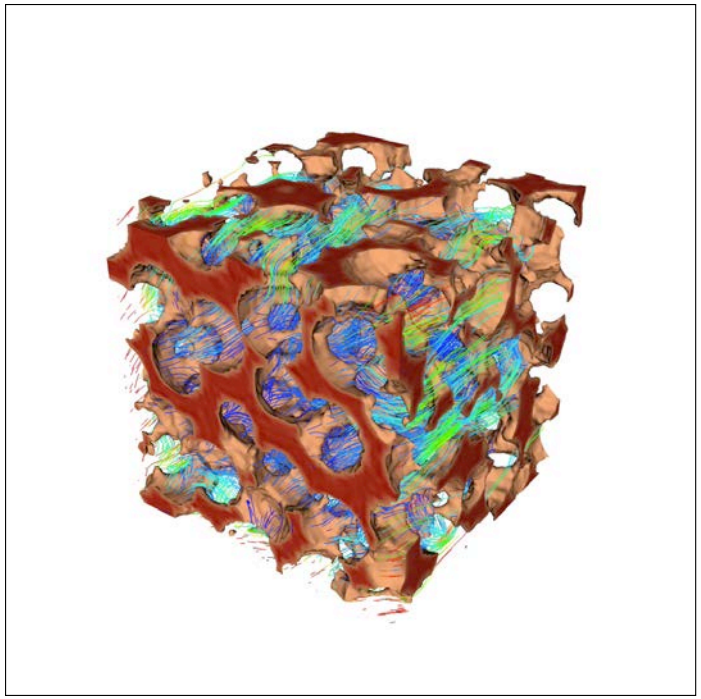
$$D + UV$$

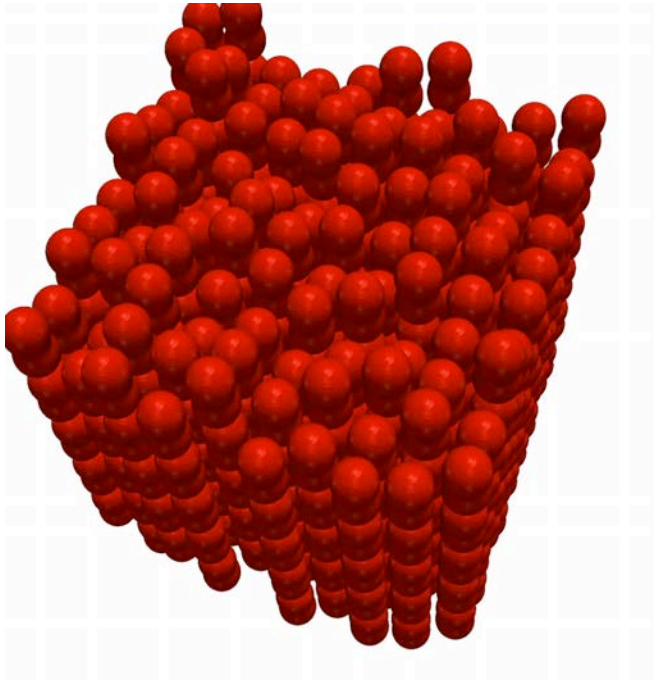
$$D + UV$$



$$\mathcal{O}(N^2) \rightarrow \mathcal{O}(N \log N)$$

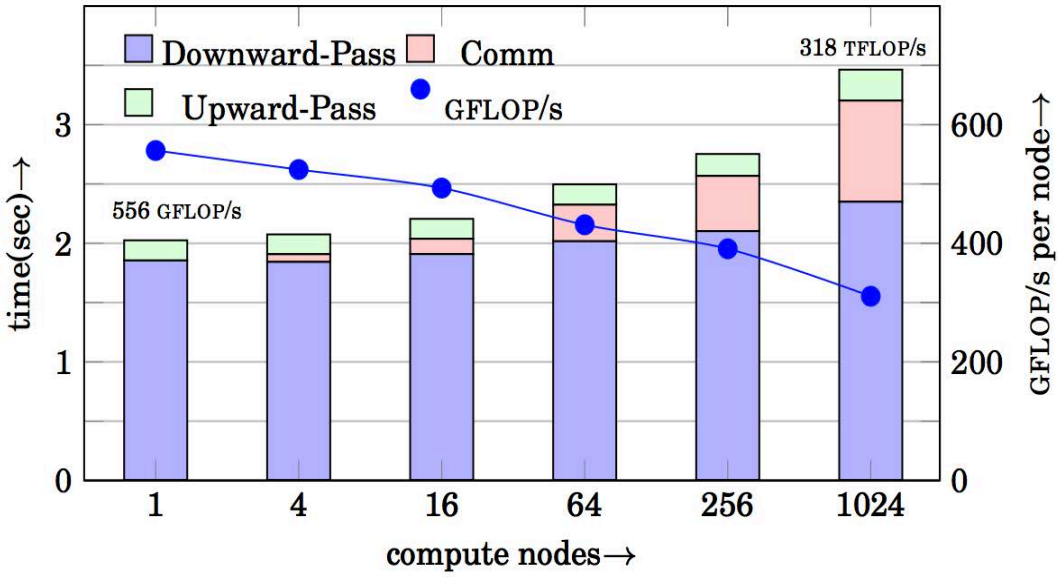
Complex fluids





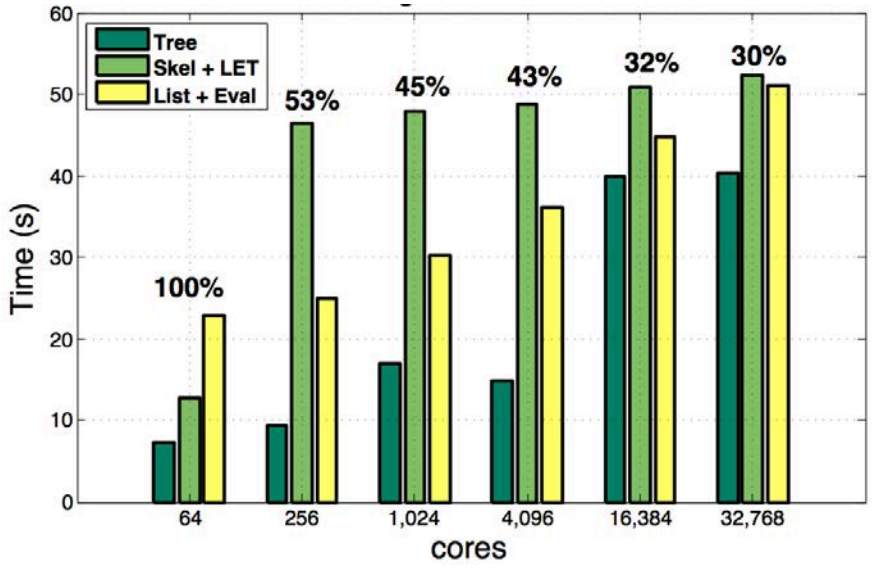
Highlights

CFD:12B/3D ~ 700 GB



Malhotra, Gholami, & B' SC14

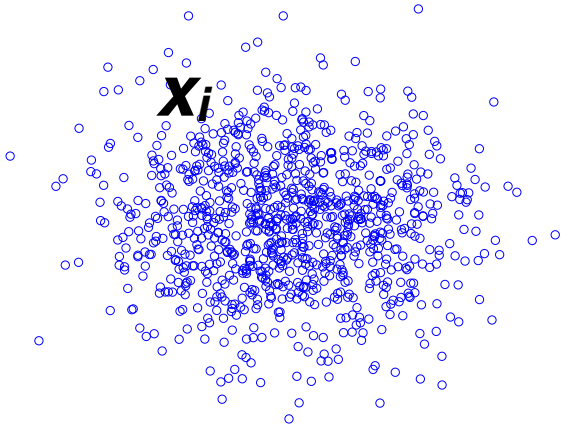
ML: 1B/128D ~ 1TB

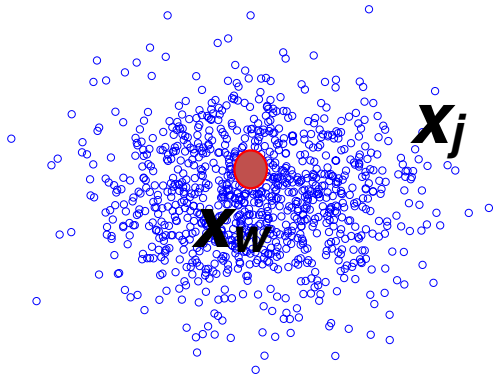


March, Yu, Xiao, B. KDD'15

Constructing the approximation

Idea I: far-field —> low rank

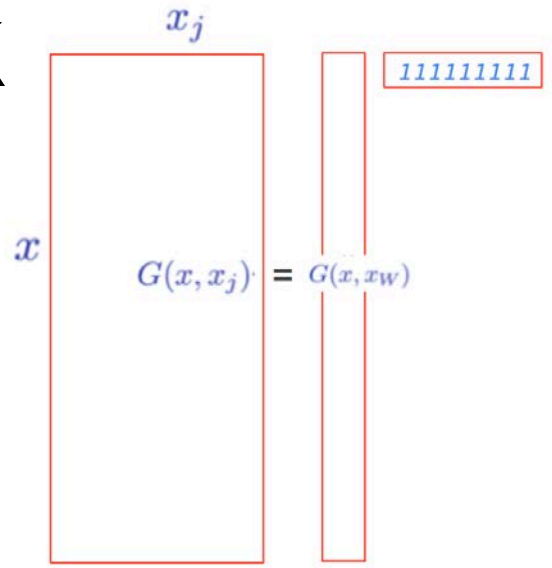


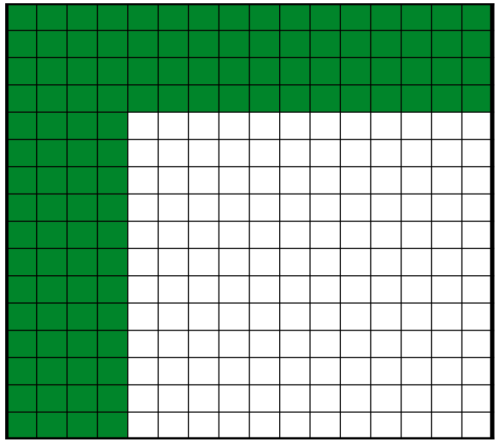


x: Target, x_j: sources
w_j: weights

$$u(x) = \sum_j G(x, x_j) w_j$$

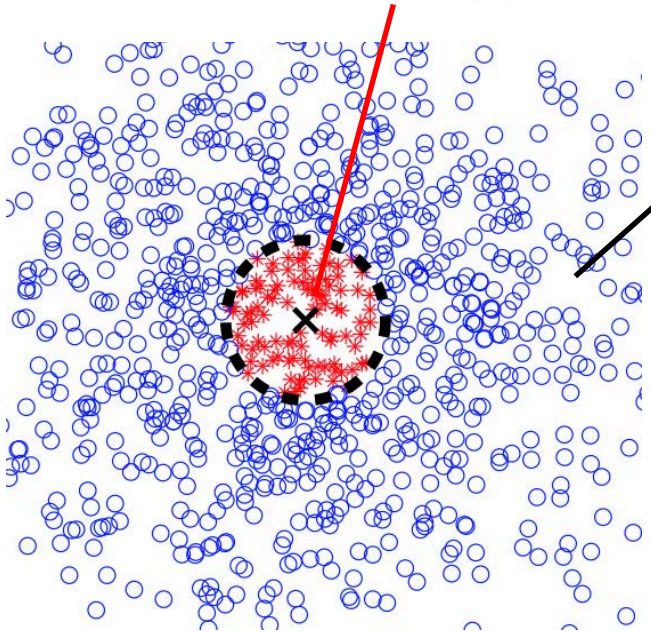
1. compute $W = \sum_i w_j$
2. choose x_W
3. $u(x) \approx G(x, x_W) W$



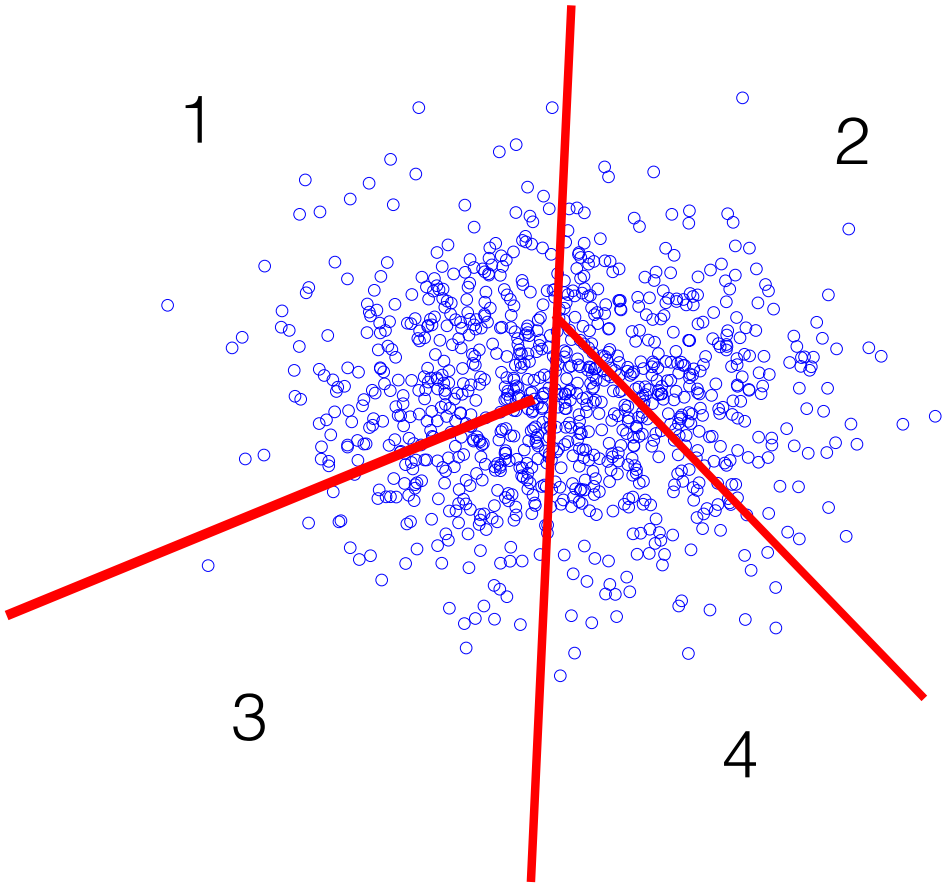


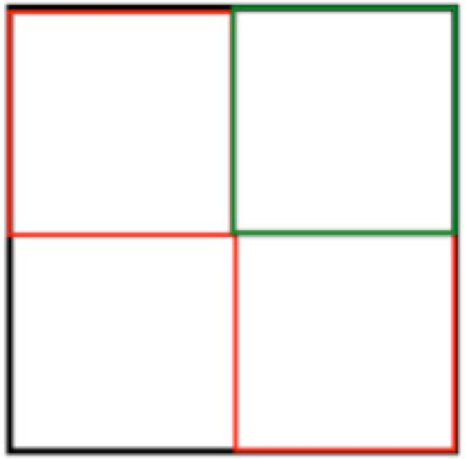
Idea II: Near/Far field split

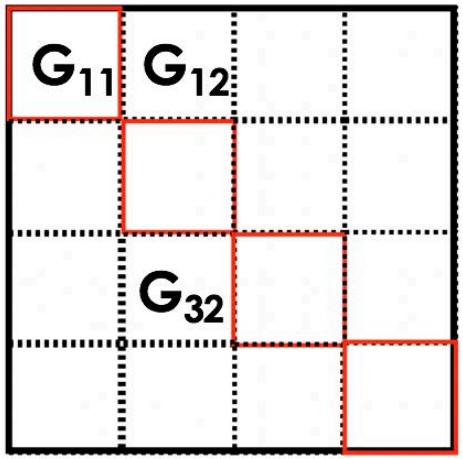
$$u_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} G(x_i, x_j) w_j = \sum_{j \in \text{near}(i)} G_{ij} w_j + \sum_{j \in \text{far}(i)} G_{ij} w_j$$



Idea III: recursion







Questions

- Accurate far-field approximation
- Optimal complexity
- Error bounds
- HPC
- **For d=4 these have been answered**

Related work — low dimensions

- Barnes & Hut’86 — treecodes
- Greengard & Rokhlin’87 — FMM
- Rokhlin’90 — high-frequency FMM
- Hackbush & Novak’89 — panel clustering
- Benderdorf’08 & Hackbush’99,’15 — H-matrices
- Greengard & Gropp’91 — parallel shared memory
- Warren & Salmon’93 — parallel distributed memory

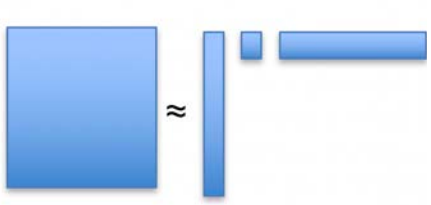
Related work — high dimensions

- Griebel et al'12 — Fast Gauss transform
- Duraiswami'06 — Improved Fast Gauss transform
- Lee, Vuduc & Gray'12 — Treecode (parallel)
- Kondor et al'16 — Wavelets in high dimensions
- Mahoney and Darve'15 — HSS matrices

Challenges in high-dimensions

- Constructing the far-field approximations
  polynomial in ambient-D
- Near-far field decomposition
  polynomial in ambient-D
- No scalable algorithms (other than Nystrom)
- Nystrom method assumes low rank
  provably not the case with increasing *N*

Randomized linear algebra — Nystrom method



- **Low-rank** decomposition of *G*
- Random sampling of ***O(s)*** points, **s:** target **rank**

$$G \approx \tilde{G} = G_{Ns} G_{ss}^{-1} G_{sN}$$

- Work $Ns + s^3$
- Error $\|G - \tilde{G}\| \leq \sqrt{1 + 6N/s}\, \sigma_{s+1}(G)$

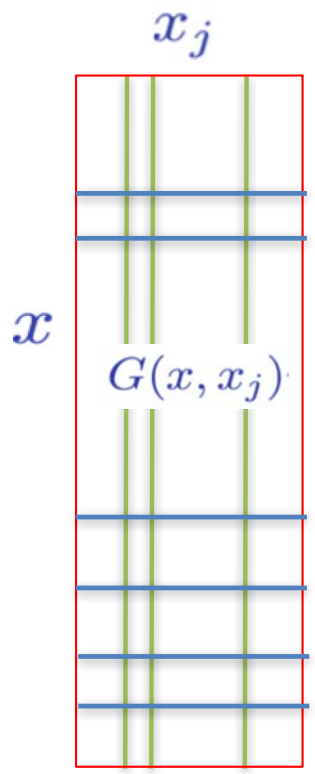
ASKIT

- Randomized Linear Algebra — far field approximation
- Parallel binary trees — permutation, partitioning
- Nearest neighbors — pruning and sampling
- Treecode / FMM
- MPI / OpenMP / SIMD / GPU acceleration
- Inspired by
  Ying & B. & Zorin'03
  Haiko & Martinsson & Tropp'11
  Drineas & Kahan & Mahoney'06

SISC'15,16
ACHA'15
KDD'15
SC'15
IPDPS'15,16,17

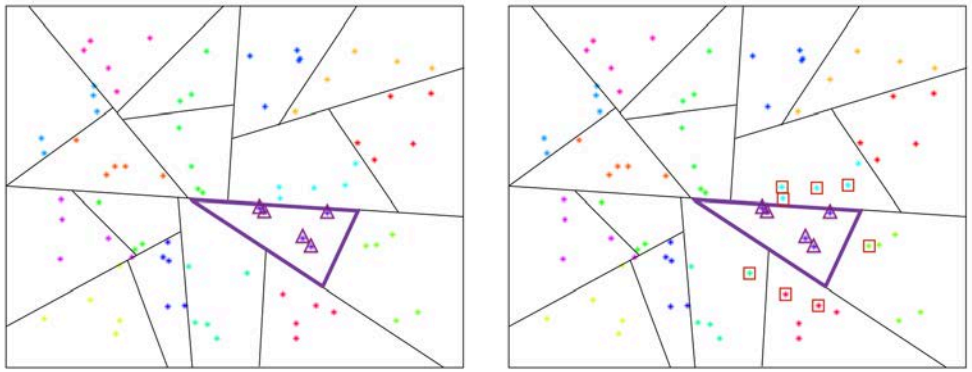
Far-field *s-rank* approximation

$$G(x, x_j) = G_{x,s}(G_{\ell,s})^{\dagger} G_{s,x_j}$$



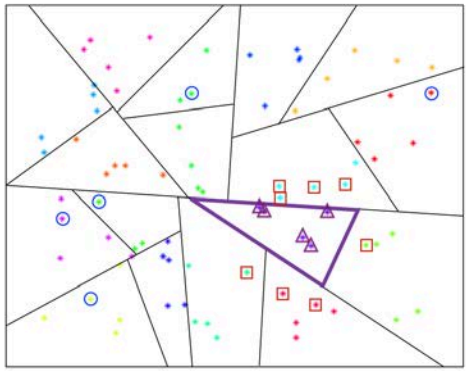
- SVD is too expensive — use sampling
- Sample rows
  leverage, norm, range-space
- Interpolative decomposition
- ASKIT: approximate norm *adaptive* sampling
  using nearest-neighbors + *adaptive* rank selection

Skeletonization (far field approximation)

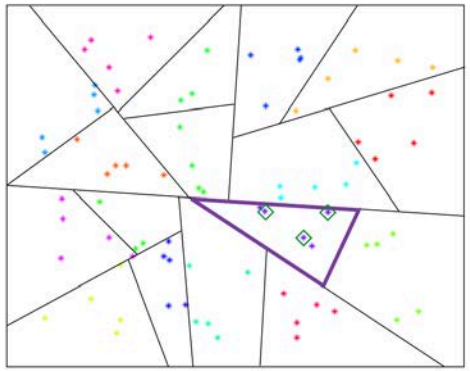


(a) A leaf node.

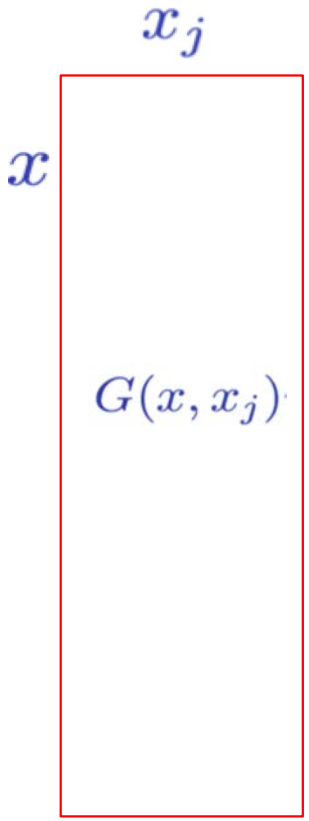
(b) The node's nearest neighbors.



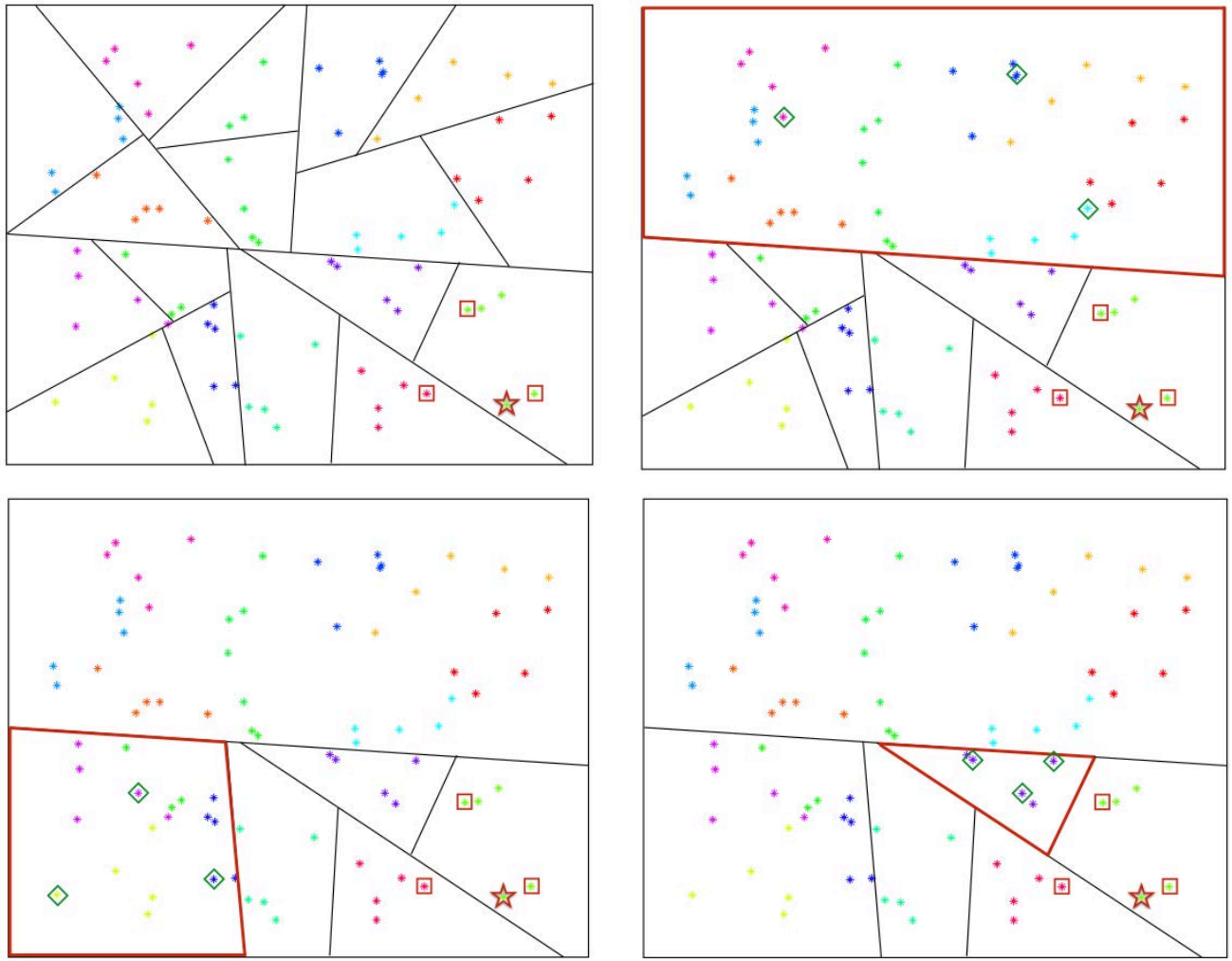
(c) Sampling distant points.



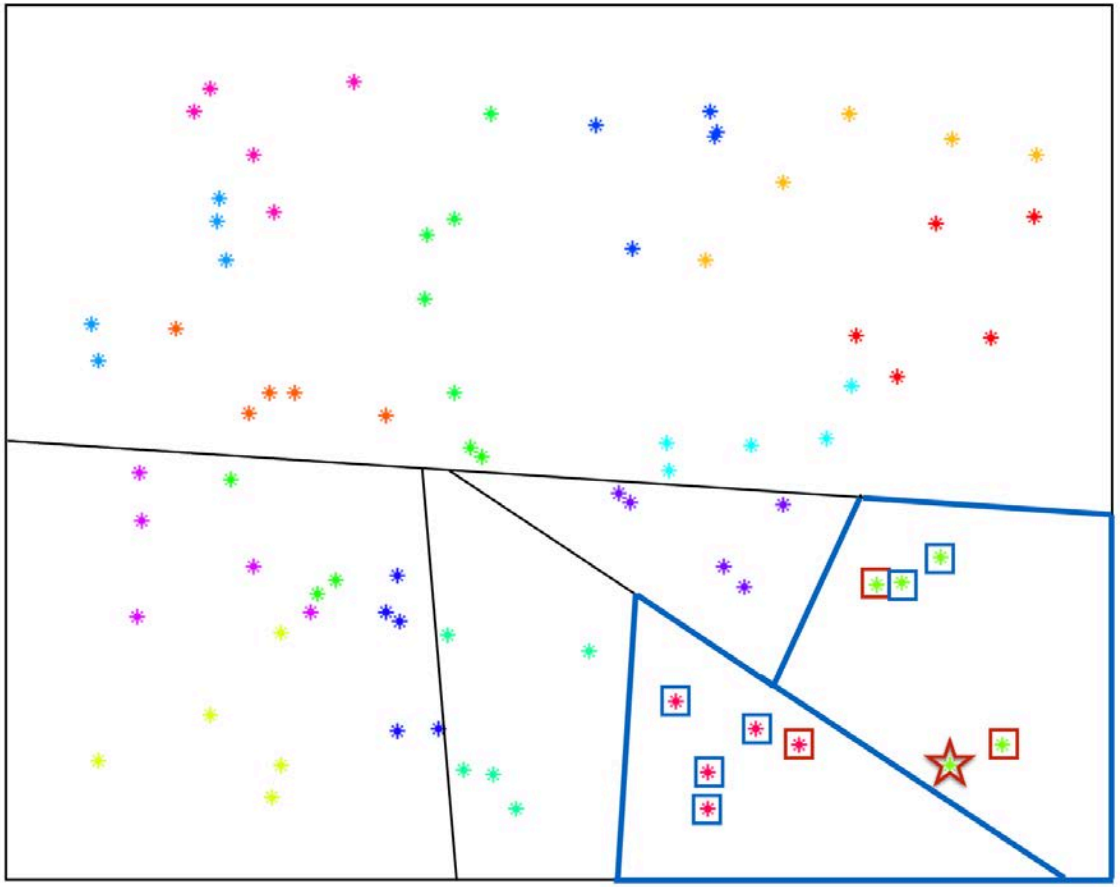
(d) The resulting skeleton.



Evaluation



Evaluation



Complexity and error

- Work

RAM	skeletonize	evaluate
$(d+\kappa)N$	Ns^2	$dNs\kappa\log(\frac{N}{s})$

- Error $\|G-\tilde{G}\| \leq \sqrt{1+6N/s}\ \log(N/s)\ \gamma_{s+1}\ \sigma_{s+1}$ off-diagonal

- Nystrom $\|G-\tilde{G}\| \leq \sqrt{1+6N/s}\ \sigma_{s+1}$

$Ns+s^3$ diagonal

Parallel complexity

Points per MPI task $n = \frac{N}{p}$ Tree depth $D = \log \frac{N}{s}$

$$\text{Tree construction} \leq (t_s + t_w) \log^2 p \log N + (t_w \log p)(d + k) n$$

$$\text{Skeletonization} \leq t_f \left(\frac{n}{s} + \log p \right) s^3$$

$$\text{Evaluation} \leq t_s p + (t_w + t_f) d\, k\, s\, D\, n$$

Summary of ASKIT features

- Binary tree for matrix perturbation
- Approximate randomized nearest neighbors
- Nearest neighbors for skeletonization
- Bottom-up recursive low-rank approximation
- Top-down pass for fast evaluation
- Adaptive sampling and rank selection

Gaussian

3D, 1M points

ϵ_2	T_S	T_{LET}	T_L	T_E	$\%K$
5E-10	439	53	7	4	2.1%
5E-05	73	16	1	1	0.6%
2E-04	29	15	1	1	0.4%
1E-03	14	15	1	1	0.3%
6E-03	10	15	1	1	0.2%

64D/20D intr, 1M points

ϵ_2	T_S	T_{LET}	T_L	T_E	$\%K$
9E-06	1068	395	149	260	56%
4E-04	486	67	11	29	6.2%
5E-03	57	30	1	9	1.6%

Kernel regression

Train:

$\{x_i \in \mathbb{R}^d, c_i \in \{-1, 1\}_{i=1}^N\}$

$\{w_j\}_{j=1}^N : \sum_{j=1}^N G(x_i, x_j) w_j = c_i, \quad \forall i.$

Classify: $c(x) = \text{sign} \sum_{j=1}^N G(x, x_j) w_j$

low rank ↑

full rank ↓

COVTYPE		SUSY		MNIST2M	
h	ϵ_c	h	ϵ_c	h	ϵ_c
0.35	71.6	0.50	65.7	4	95.0
0.22	74.0	0.15	72.1	2	97.4
0.14	79.8	0.09	75.0	1	100
0.02	95.4	0.05	76.7	0.1	99.5
0.001	6.4	0.01	64.3	0.05	13.6

Kernel acceleration

Data	N	d	ϵ_2	$\%K$
Uniform	1M	64	5E-3	1.6%
Covtype	500K	54	8E-2	2.7%
SUSY	4.5M	18	5E-3	0.4 %
HIGGS	10.5M	28	1E-1	11%
BRAIN	10.5M	246	5E-3	0.9%

Nystrom vs ASKIT (8M/784D)

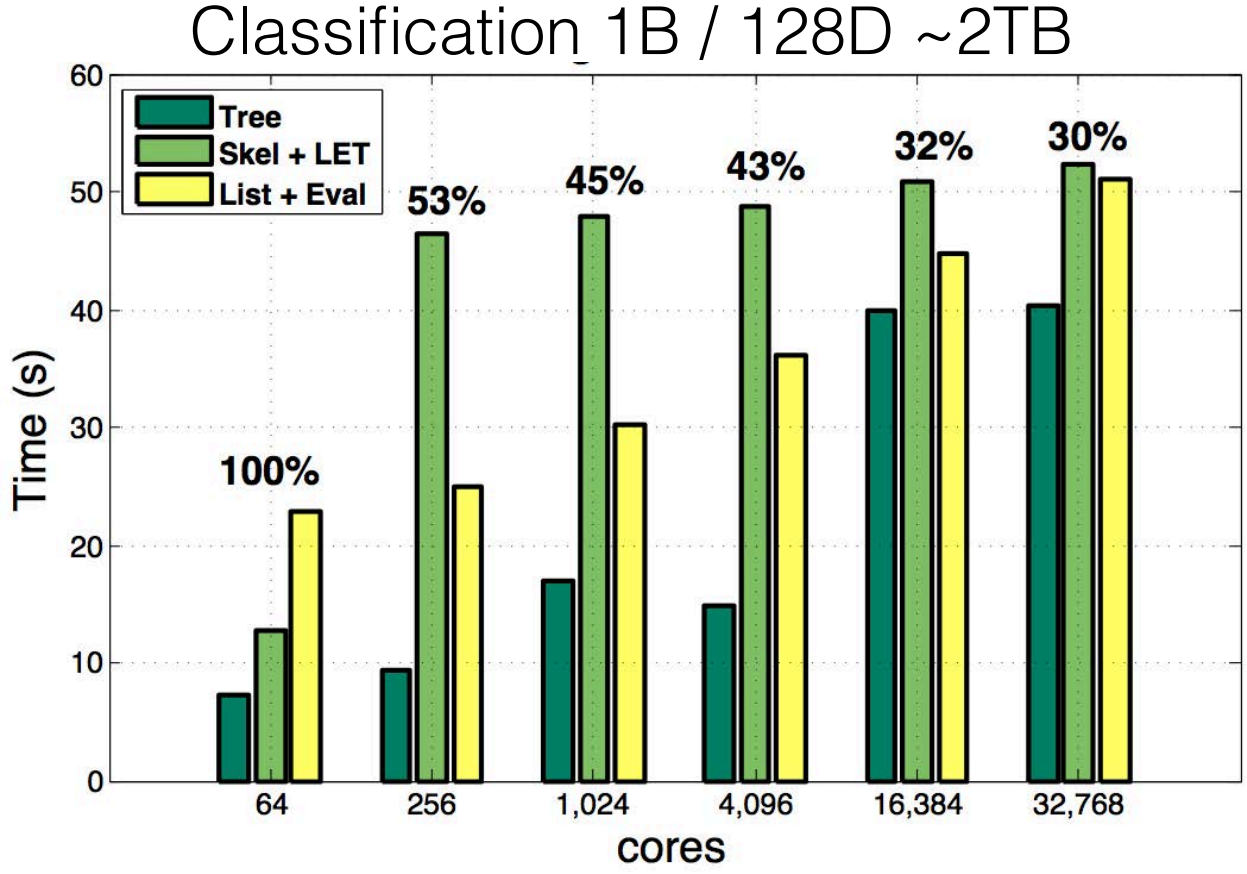
	Param	$h = 0.5$			$h = 1$		
		ϵ_2	T	T_E	ϵ_2	T	T_E
NYSTROM	$r = 1024$	>9E-1	63	<1	>9E-1	63	<1
	$r = 2048$	>9E-1	122	<1	>9E-1	120	<1
	$r = 4096$	>9E-1	299	<1	>9E-1	301	<1
	$r = 8192$	mem	–	–	mem	–	–
ASKIT	$\kappa = 256$	1E-4	226	32	3E-2	154	31
	$\kappa = 512$	3E-5	243	39	2E-2	181	38
	$\kappa = 1024$	5E-6	306	50	2E-2	239	47
	$\kappa = 2048$	9E-7	410	65	8E-3	370	62

Kernel regression scaling

MNIST dataset for OCR
strong scaling, 8M points d=784

#cores	512	2,048	4,096	8,192	16,384
Skel. (Alg. 2)	1,295	465	370	305	269
Lists (Alg. 3)	729	177	87	46	23
LET (Eq. 7)	273	136	107	87	71
Eval. (Eq. 14)	157	67	42	28	23
Total	2,471	862	621	483	394
Efficiency	1.00	0.72	0.50	0.32	0.20

Weak scaling



TACC's Stampede
Largest run 144s
200 TFLOPs
30% peak

Fast factorization

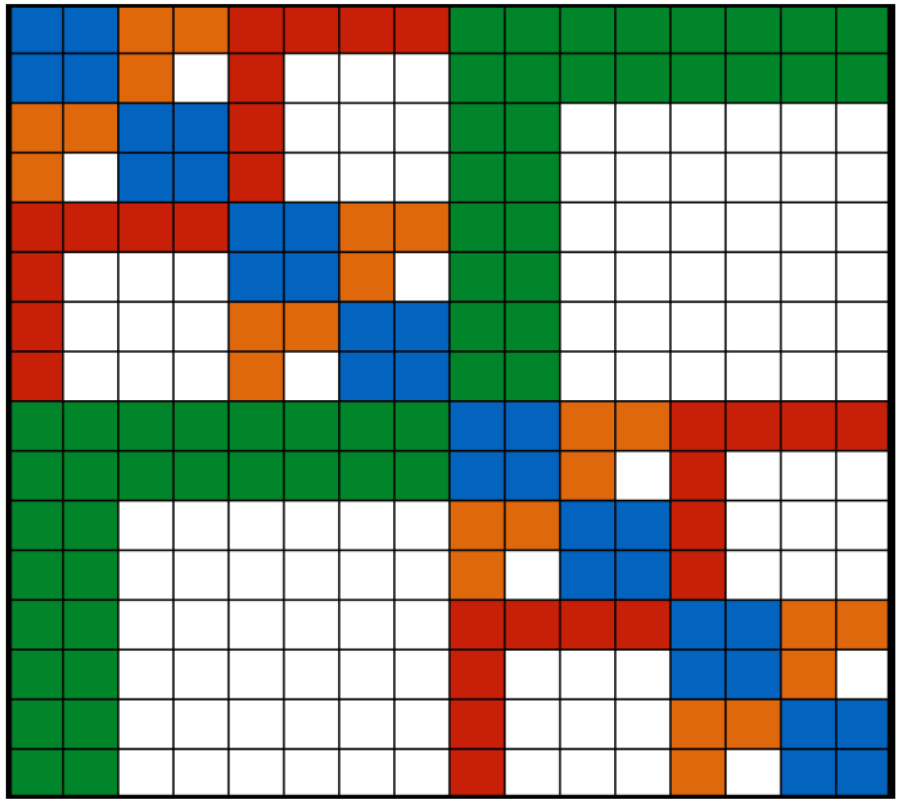
$$\begin{bmatrix} G_{11} & 0 \\ 0 & G_{22} \end{bmatrix} + \begin{bmatrix} 0 & G_{12} \\ G_{21} & 0 \end{bmatrix}$$

$$D + UV = D(I + D^{-1}UV)$$

$$D(I + WV), \text{ where } W = D^{-1}U.$$

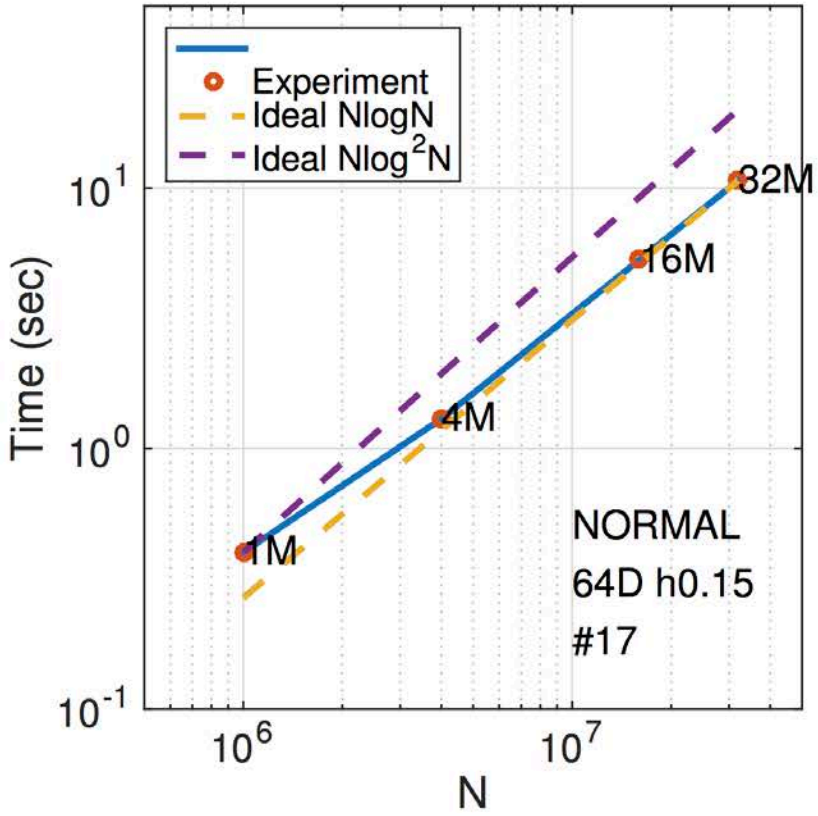
$$(I + WV)^{-1}D^{-1}$$

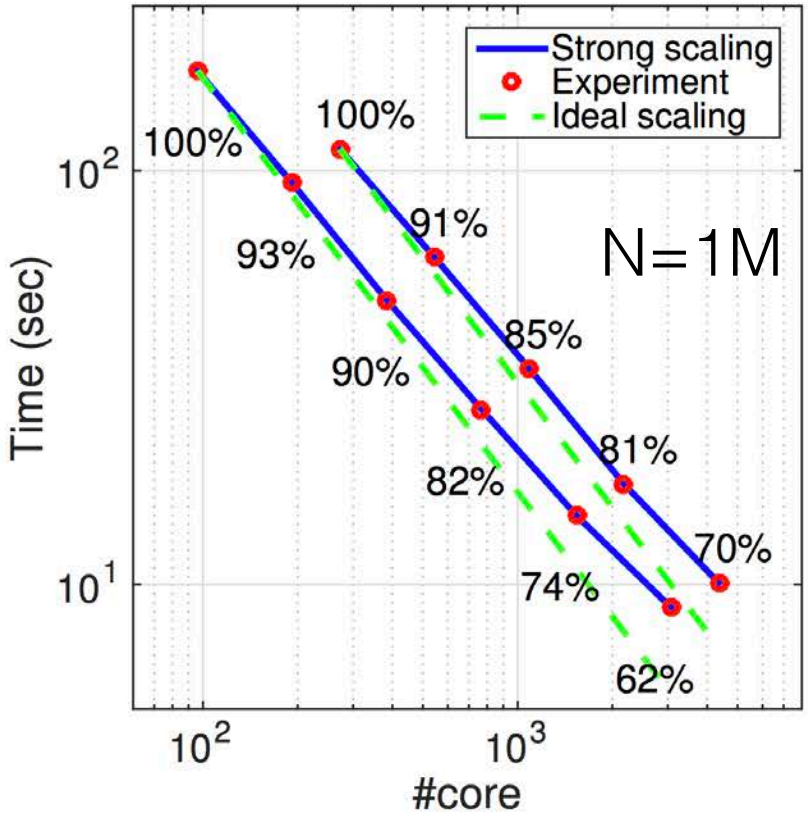
$$(I - W(I + VW)^{-1}V)D^{-1}$$



$$\mathcal{O}(N^2) \rightarrow \mathcal{O}(N \log N)$$

Scalability of factorization





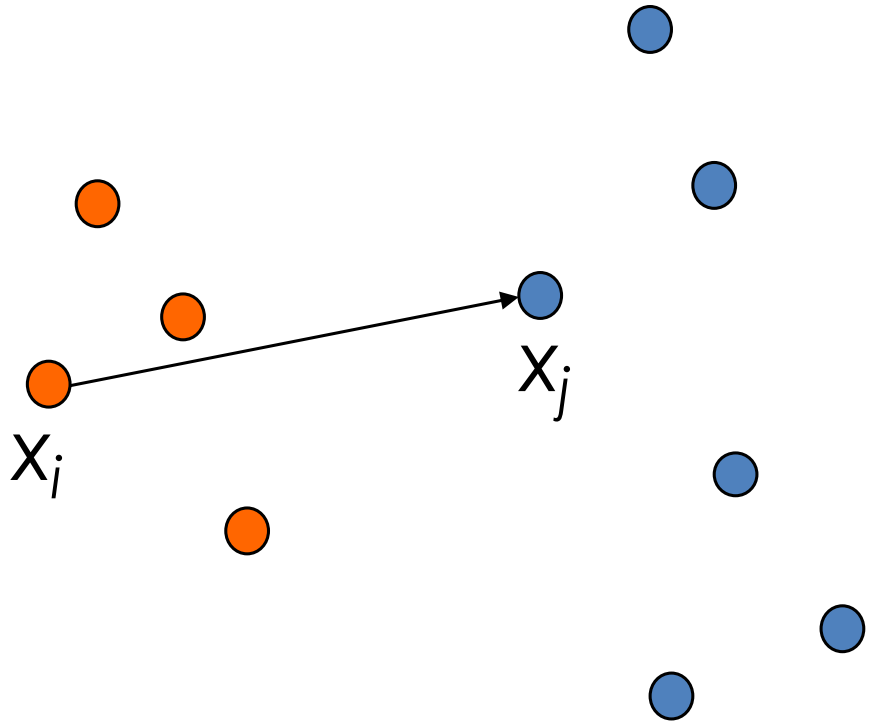
Computational primitives

- Geometric — distance calculation / projections
- Analytic — special functions / fast transforms
- Algebraic — BLAS / QR / SVD / Cholesky
- Combinatorial — hash / sort / merge / select / search
- Memory — permute / pack / gather / scatter

Nearest-neighbor primitive (exhaustive)

Compute all pairwise distances

Sort them and select ***k*** neighbors



Nearest-neighbor kernel

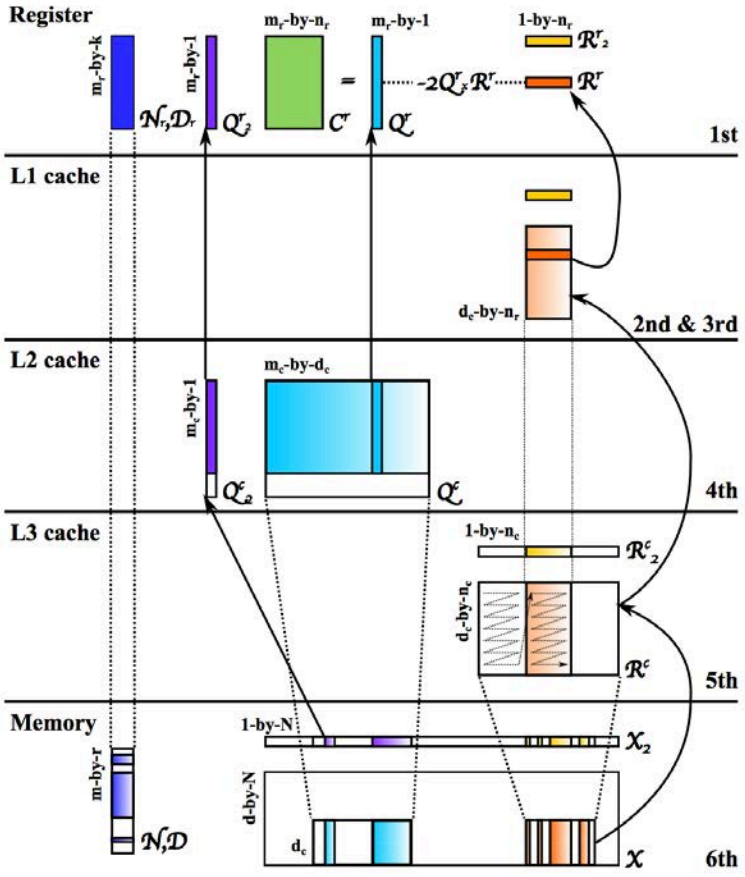
$$\|x_i - x_j\|_2^2 = \|x_i\|_2^2 + \|x_j\|_2^2 - 2x_i^T x_j$$

Algorithm 2.1 `GEMM` Approach to k-Nearest Neighbor

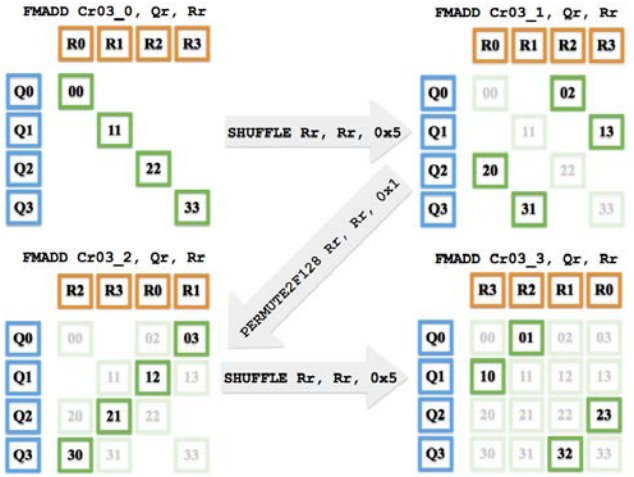
$C = -2Q^T R$; //GEMM$(-2, Q^T, R, 0, C)$;
for each query i and reference j **do**
  $C(i,j) += Q_2(i) + R_2(j)$; //$-2x_i^T x_j + \|x_i\|_2^2 + \|x_j\|_2^2$;
end for
for each query i **do**
  Update $<\mathcal{N}(i,:),\mathcal{D}(i,:)>$ with $<r(:),C(i,:)>$
end for

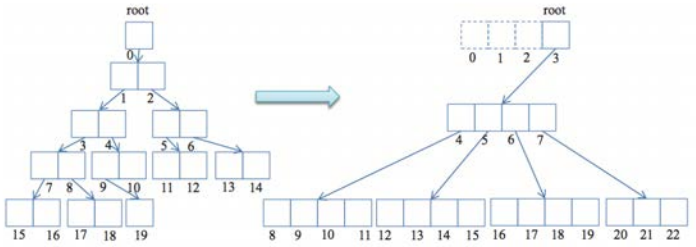
Input: *O(Nd)*
Output: *O(Nk)*
Calculations: *O(N²d + Nk log k)*
Intermediate storage: *O(N²)*

Custom GEMM with heap-selection

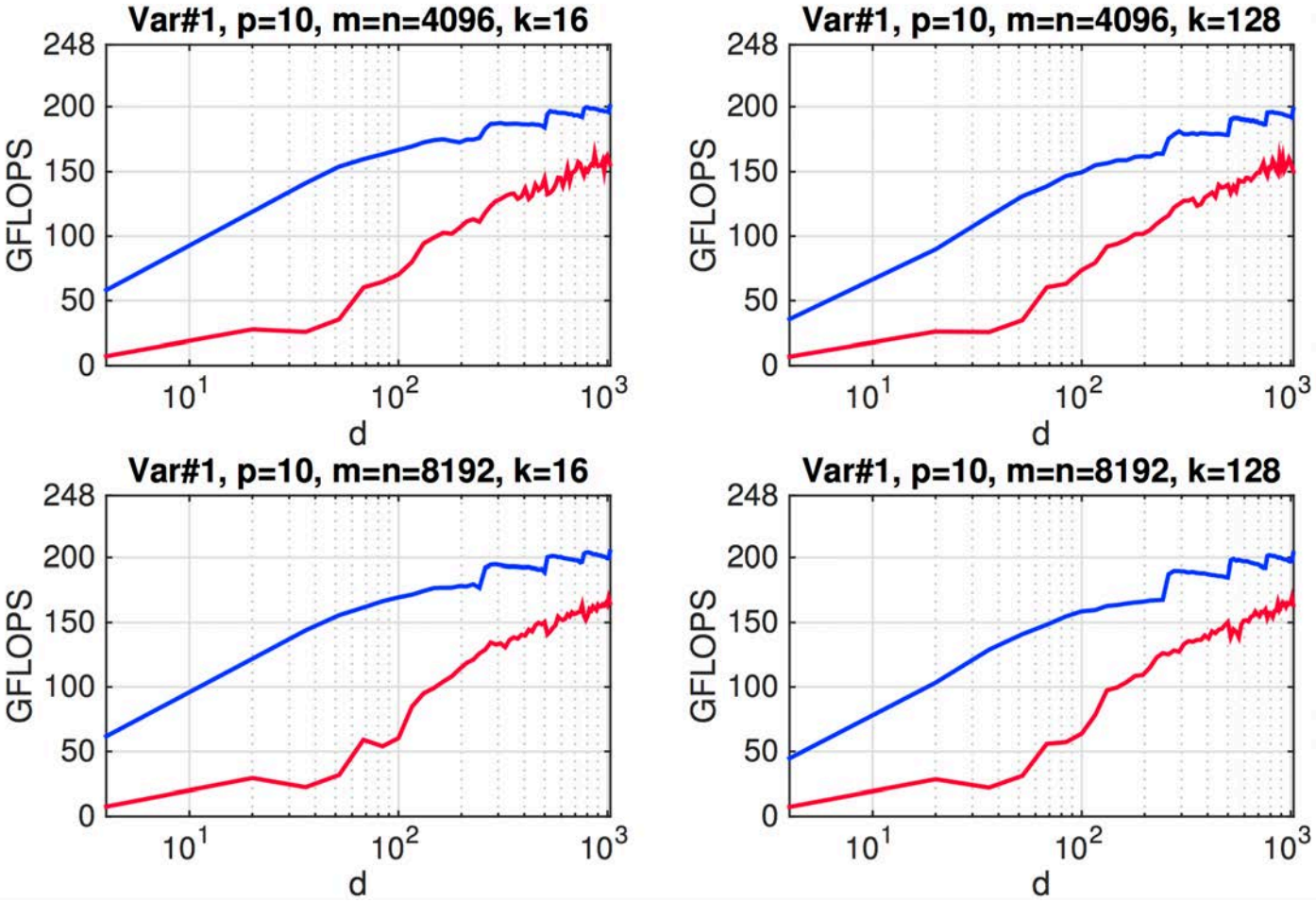


SC'15





Comparison with MKL



Summary

- ASKIT scheme for kernel matrix approximation
- Nearest neighbors and randomized linear algebra
- Performance depends on ambient intrinsic dimension and scale
- Supports shared/distributed memory and GPU acceleration
- Algorithms: kernel regression, kernel logistic regression, kernel clustering, kernel PCA, kernel density estimation, nearest-neighbors
- **padas.ices.utexas.edu/libaskit**