Performance Study of a Randomized Low-rank Approximation using multi-GPU

Théo, Ichi, Jakub, Piotr, Stan,

September 5, 2014

Low-rank Approximation

$$\underset{m \times n}{A} \approx \underset{m \times k}{B} \underset{k \times n}{C}$$

- If $\|A - BC\| \leq \varepsilon$, then $k =$ numerical rank of A.
- "Low-rank" $\Leftrightarrow k \ll \min(m, n)$.
  - Reduced computations and storage.

Pivoted QR Decomposition

- Pivoted QR decomposition

$$AP = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix}$$

  with
  - $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ an $m \times n$ orthogonal matrix;
  - $R = \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix}$ an $n \times n$ upper triangular matrix;
  - P a $n \times n$ pivot matrix.
- Truncated Pivoted QR Decomposition

$$\underset{m \times n}{AP} \approx \underset{m \times k}{Q_1} \underset{k \times n}{\begin{pmatrix} R_{11} & R_{12} \end{pmatrix}}$$

LAPACK's QP3

- Computes QR with column pivoting using BLAS-3.
- To compute truncated version $\rightarrow$ one line to change in the code.
- Limitations:
  - Not only BLAS-3: also BLAS-2;
  - Synchronization at every step to pick pivot;
  - Limited parallelism and data locality;
  - Communications.
  - Column norms get messed up $\rightarrow$ need to update.

Randomized Algorithm: Overview

- **Stage A**: generate Q, orthogonal subspace spanning the range of A, i.e.:

$$A \approx AQ^T Q$$

- **Stage B**: use Q to compute low-rank approximations of A (QR, SVD, ...) with standard deterministic methods.

for $i = 1, 2, \ldots, k$ **do**
    Draw random vector ω_i.
    Form the product $b_i = \omega_i A$.
end for

- $B = \{b_1, \ldots, b_k\}$ is **probably** linearly independent
  $\Rightarrow$ $Q = \text{orth}(B)$.

Stage A: Intuition

for $i = 1, 2, \ldots, k + p$ **do**
    Draw random vector ω_i.
    Form the product $b_i = \omega_i A$.
end for

- $B = \{b_1, \ldots, b_{k+p}\}$ is **probably** linearly independent
  $\Rightarrow$ $Q = \text{orth}(B)$.
- p is the oversampling.

Stage A: Sampling

- $\ell = k + p$

$$\underset{\ell \times n}{B} = \underset{\ell \times m}{\Omega} \underset{m \times n}{A}$$

Stage A: Sampling

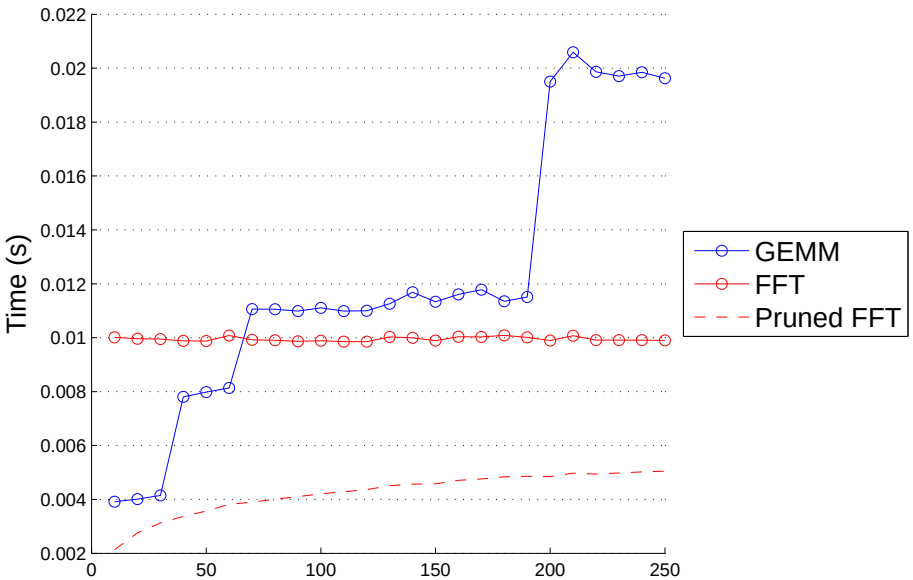
- $\ell = k + p$

$$\underset{\ell \times n}{B} = \underset{\ell \times m}{\Omega} \underset{m \times n}{A}$$

- How do we generate Ω?
  - Gaussian
  - FFT

GEMM vs FFT

$$\underset{l \times n}{B} = \underset{l \times m}{S} \quad \underset{m \times m}{\Pi} \quad \underset{m \times n}{A}$$



(m=50,000, n=500, $\ell = 10 \rightarrow 250$)

GEMM	$\mathcal{O}(mn\ell)$
FFT	$\mathcal{O}(mn\log(m))$
Pruned FFT	$\mathcal{O}(mn\log(\ell))$

Stage A: Power Method

- $\|A - AQ^T Q\| \leq C(\Omega, p)\sigma_{k+1}$
  - If $\{\sigma_i\}_{i=1,n}$ decay slowly, $\|A - AQ^T Q\|$ can be big.
- $B = \Omega\, A\, (A^T A)^q$
- $\|A - AQ^T Q\| \leq C(\Omega, p)^{1/(2q+1)}\sigma_{k+1}$

Stage A: Power Method

- $\|A - AQ^T Q\| \leq C(\Omega, p)\sigma_{k+1}$
  - If $\{\sigma_i\}_{i=1,n}$ decay slowly, $\|A - AQ^T Q\|$ can be big.
- $B = \Omega\, A\, (A^T A)^q$
- $\|A - AQ^T Q\| \leq C(\Omega, p)^{1/(2q+1)}\sigma_{k+1}$
- Round-off errors $\Rightarrow$ need to reorthogonalize.

$$
\begin{array}{lcl}
B_0 & = & \Omega\, A \\
\text{repeat } q \text{ times:} & & \\
\acute{Q}_0 & = & \text{orth}(B_0) \quad ; \quad \acute{B}_1 = \acute{Q}_0\, A^T \\
\acute{Q}_1 & = & \text{orth}(\acute{B}_1) \quad ; \quad B_1 = \acute{Q}_1\, A
\end{array}
$$

Randomized PQR

- Truncated PQR step:

$$
\begin{aligned}
BP &\approx \widehat{Q}_k \begin{pmatrix} \widehat{R}_{1:k} & \widehat{R}_{k+1:n} \end{pmatrix} \\
&= \widehat{Q}_k \widehat{R}_{1:k} \begin{pmatrix} I_k & \widehat{R}_{1:k}^{-1} \widehat{R}_{k+1:n} \end{pmatrix} \\
&= BP_{1:k} \begin{pmatrix} I_k & \widehat{R}_{1:k}^{-1} \widehat{R}_{k+1:n} \end{pmatrix} \\
\Rightarrow AP &\approx AP_{1:k} \begin{pmatrix} I_k & \widehat{R}_{1:k}^{-1} \widehat{R}_{k+1:n} \end{pmatrix}
\end{aligned}
$$

- QR step:

$$AP_{1:k} = Q\overline{R}$$

- Final approximation:

$$
\begin{array}{ccccc}
AP & \approx & Q & \overline{R} & \begin{pmatrix} I_k & \widehat{R}_{1:k}^{-1} \widehat{R}_{k+1:n} \end{pmatrix} \\
m \times n & & m \times k & & k \times n \\
 & & \mathbf{Q} & & \mathbf{R}
\end{array}
$$

Implementation

1: Input: $m \times n$ matrix A.
2: $B_0 = \Omega\, A$
3: **for** 1, 2, ..., q **do**
4: $\quad \acute{Q}_0 = \mathrm{orth}(B_0)$
5: $\quad \acute{B}_1 = \acute{Q}_0\, A^T$
6: $\quad \acute{Q}_1 = \mathrm{orth}(\acute{B}_1)$
7: $\quad B_1 = \acute{Q}_1\, A$
8: **end for**
9: $\widehat{Q}, \widehat{R}, P = \mathrm{TruncPQR}(B_q)$
10: $Q, \overline{R} = \mathrm{QR}(AP_{1:k})$
11: $R = \overline{R} \begin{pmatrix} I_k & \widehat{R}_{1:k}^{-1} \widehat{R}_{k+1:n} \end{pmatrix}$
12: Output: Q, R, P such that $AP \approx QR$.

Communications

- Two memory levels hierarchy: fast/slow (M = size of fast memory).

	#flops	#words
Sampling	$\mathcal{O}(mn\ell)$	$\mathcal{O}(mn\ell/M^{1/2})$
Iter. (mult.)	$\mathcal{O}(mn\ell q)$	$\mathcal{O}(mn\ell q/M^{1/2})$
Iter. (orth.)	$\mathcal{O}((m+n)\ell^2)$	$\mathcal{O}((m+n)\ell^2/M^{1/2})$
TruncPQR	$\mathcal{O}(n\ell^2)$	$\mathcal{O}(n\ell^2)$
QR	$\mathcal{O}(m\ell^2)$	$\mathcal{O}(m\ell^2/M^{1/2})$
Total	$\mathcal{O}(mn\ell(1+q))$	$\mathcal{O}(mn\ell(1+q)/M^{1/2})$
TruncQP3	$\mathcal{O}(mnk)$	$\mathcal{O}(mnk)$

- Under assumption that p, q are constant, Randomized PQR converges towards communications lower bound.

Orthogonalization (I)

- Needed for power method and for QR(B) step.

	Stability		Performance
Householder QR	☺☺	ε	☹
Cholesky QR	☺	$\kappa(A)^2\varepsilon$	☺
CA QR	☺☺	ε	☺?
CGS	☺	$\kappa(A)^2\varepsilon$	☺
MGS	☺	$\kappa(A)\varepsilon$	☹

Orthogonalization (II)

- Cholesky QR Algorithm:
  1. Form $S = X^T X$.
  2. Compute Cholesky factorization $R = \text{chol}(S)$.
  3. Solve $Q = XR^{-1}$.

Orthogonalization (II)

- Cholesky QR Algorithm:
  1. Form $S = X^T X$.
  2. Compute Cholesky factorization $R = \text{chol}(S)$.
  3. Solve $Q = XR^{-1}$.
- Orthogonalization schemes
  - Repeat Cholesky QR multiple times.
  - Try Cholesky, if fail, do Householder.
  - For power method, for tall & skinny matrices: do Cholesky on bigger matrix, and Householder on smaller one.
  - For power method, orthogonalize only at given iterations.
  - Mixed-precision Cholesky QR.

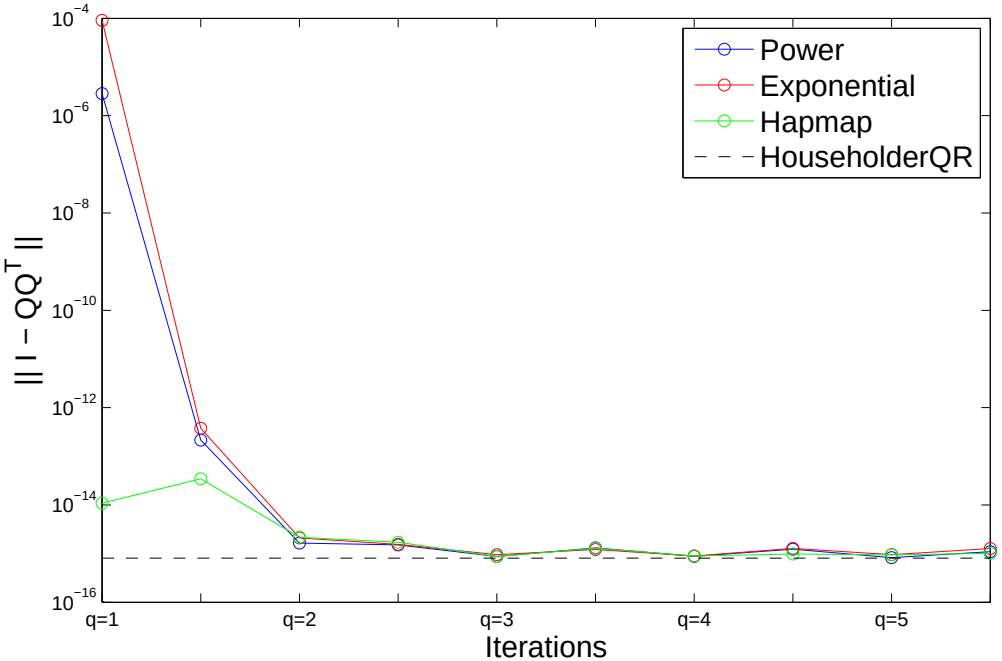
Experimental Setups

- Hardware: Bunsen (16 threads running on 16 cores Genuine Intel(R) CPU @ 2.60GHz + 3 GPUs Tesla K40c)
- Software: MAGMA.
- Test matrices:
  - power: $A = X\Sigma Y$, with $\sigma_i = -i^{\alpha}$, ($\alpha = 3$).
  - exponent: $A = X\Sigma Y$, with $\sigma_i = 10^{-i\gamma}$ ($\gamma = 0.1$).
  - hapmap.

	power	exponent	hapmap
m	500,000	500,000	503,783
n	500	500	506
k	50	50	50
p	10	10	10
ℓ	60	60	60
σ_1	1	1	9.9e+03
σ_{k+1}	8e-06	1.3e-05	5e+02
$\kappa(A)$	1.3e+05	7.9e+04	2e+01

Orthogonalization (III)

- Orthogonality norm: $\|I_\ell - \acute{Q}_0 \acute{Q}_0{}^T\|$ and $\|I_\ell - \acute{Q}_1 \acute{Q}_1{}^T\|$, at each iteration.
- $\kappa(B) \approx \kappa(A)$



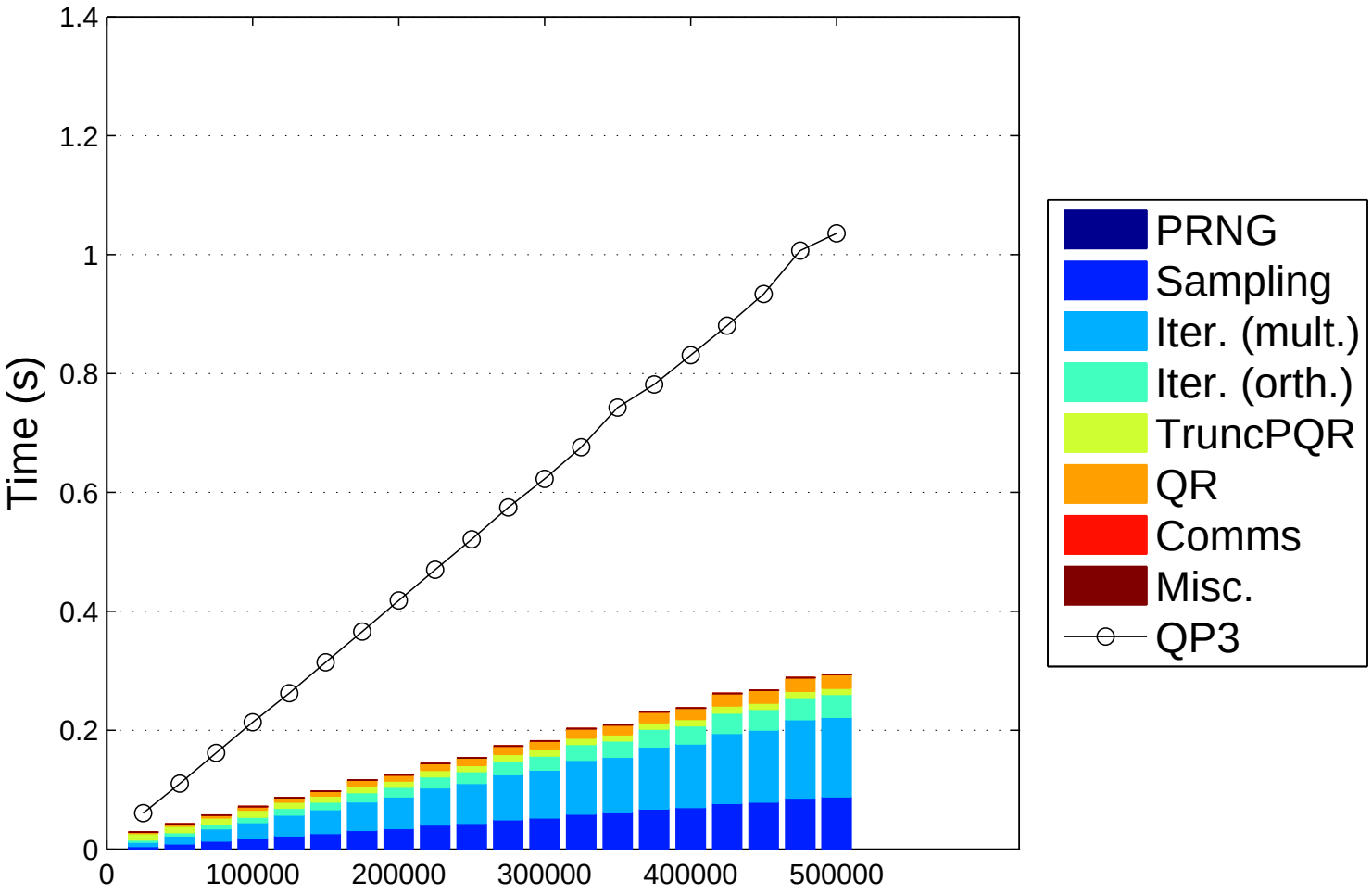
Convergence

- Approximation error $\|AP - QR\|/\|A\|$

	QP3	Rand $q = 0$	Rand $q = 1$	Rand $q = 2$
Power	4.4679e-05	9.0784e-05	4.5948e-05	4.4489e-05
Exponent	2.6892e-05	5.1795e-05	2.6899e-05	2.6898e-05
Hapmap	5.9887e-01	9.8563e-01	8.7363e-01	8.1816e-01

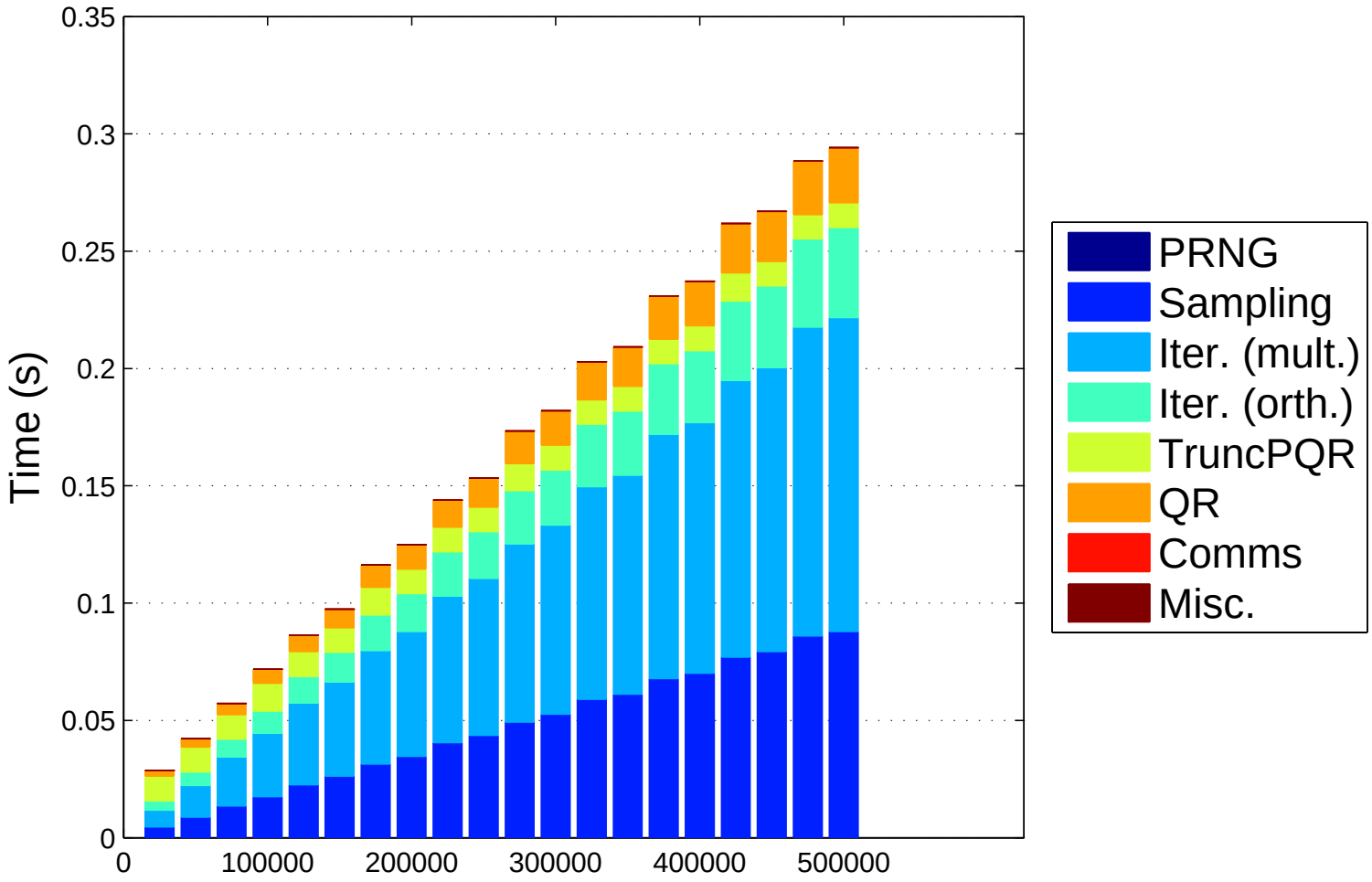
- Usefulness of oversampling: roughly an order of magnitude.
- Other error measurements?

Time with increasing number of rows



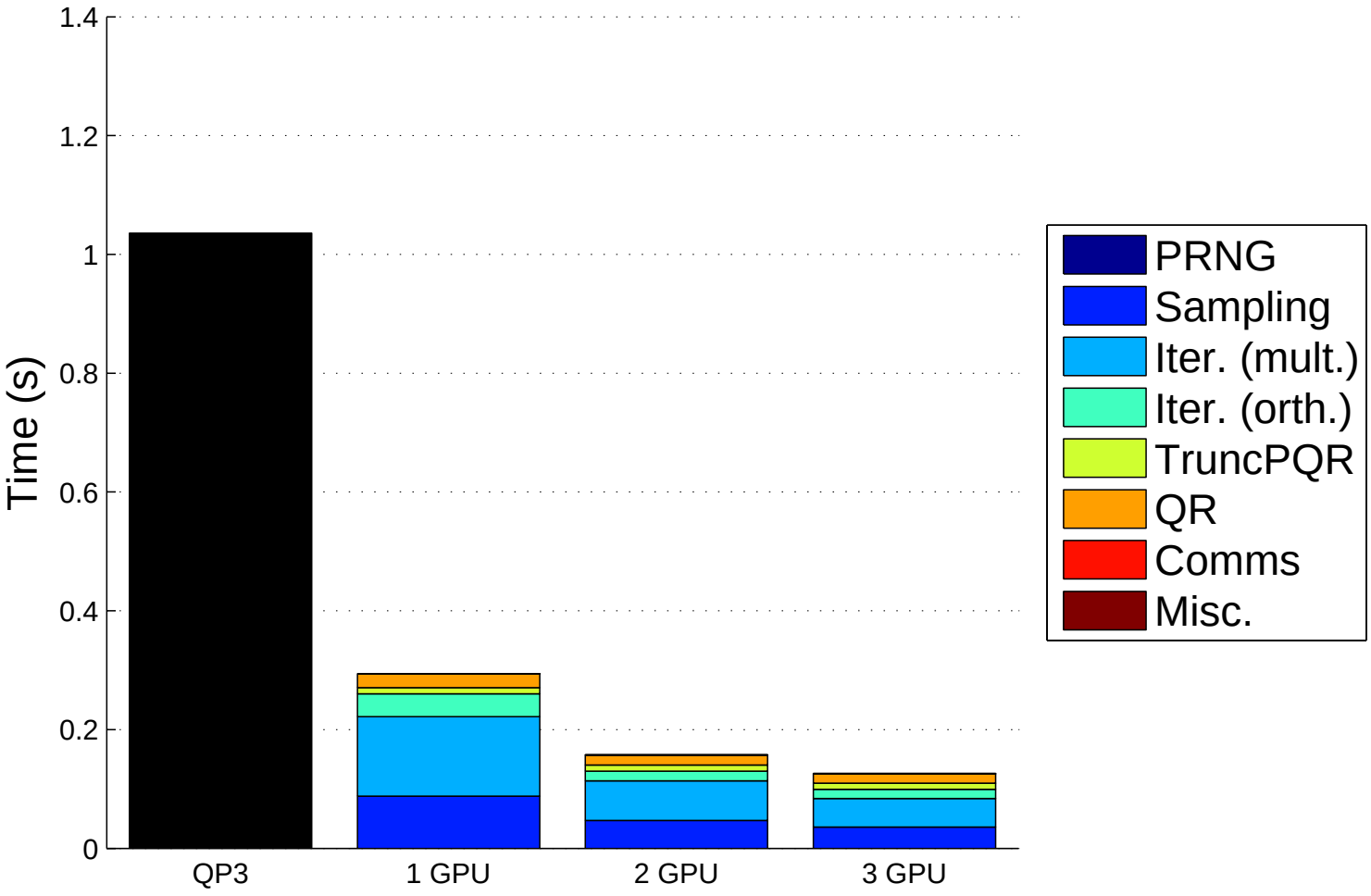
$(m = 25,000 \rightarrow 500,000;\ n = 500;\ \ell = k + p = 50 + 10;\ q = 1)$

Time with increasing number of rows



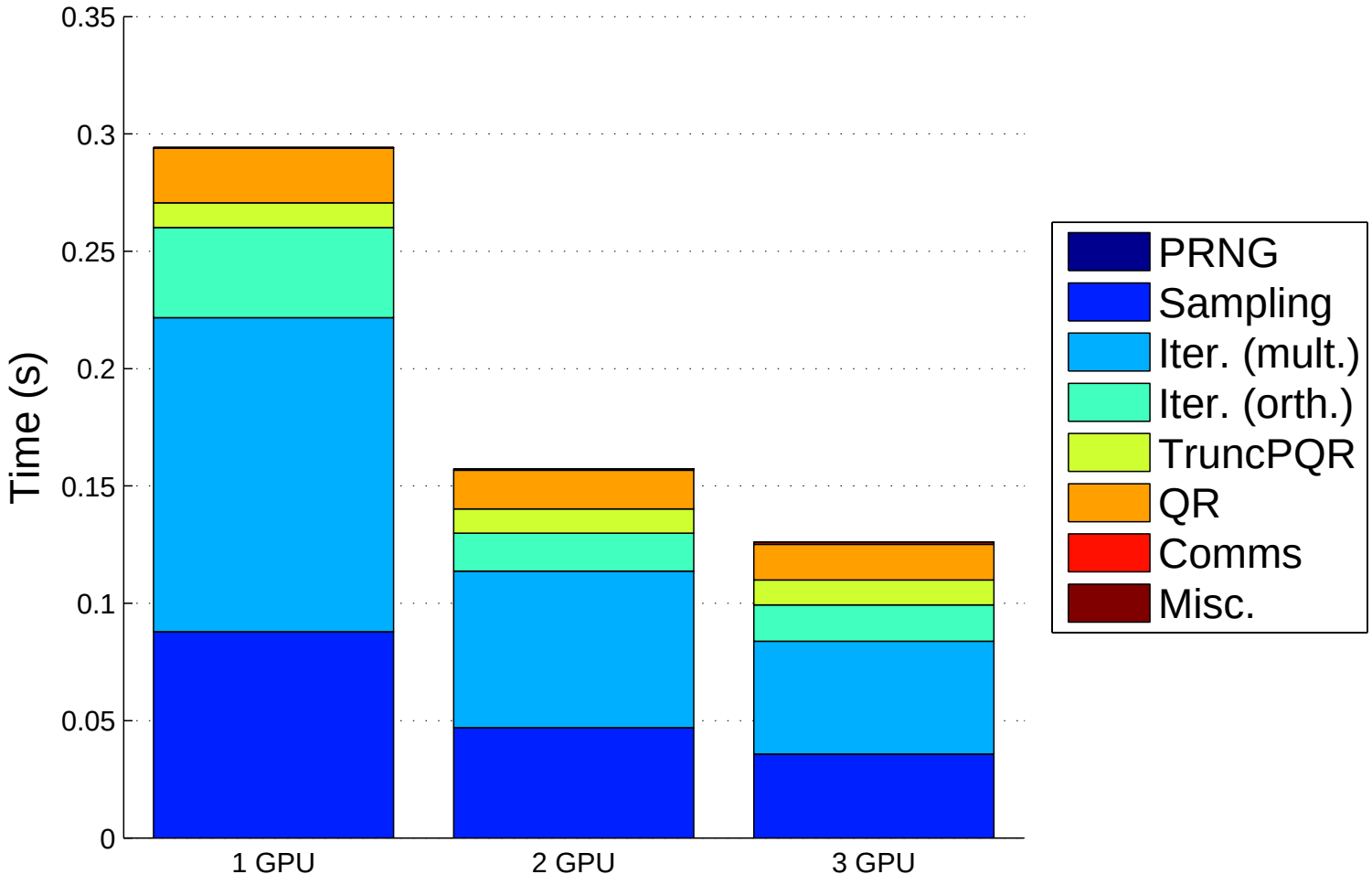
$(m = 25,000 \rightarrow 500,000;\ n = 500;\ \ell = k + p = 50 + 10;\ q = 1)$

Time on 1,2,3 GPUs



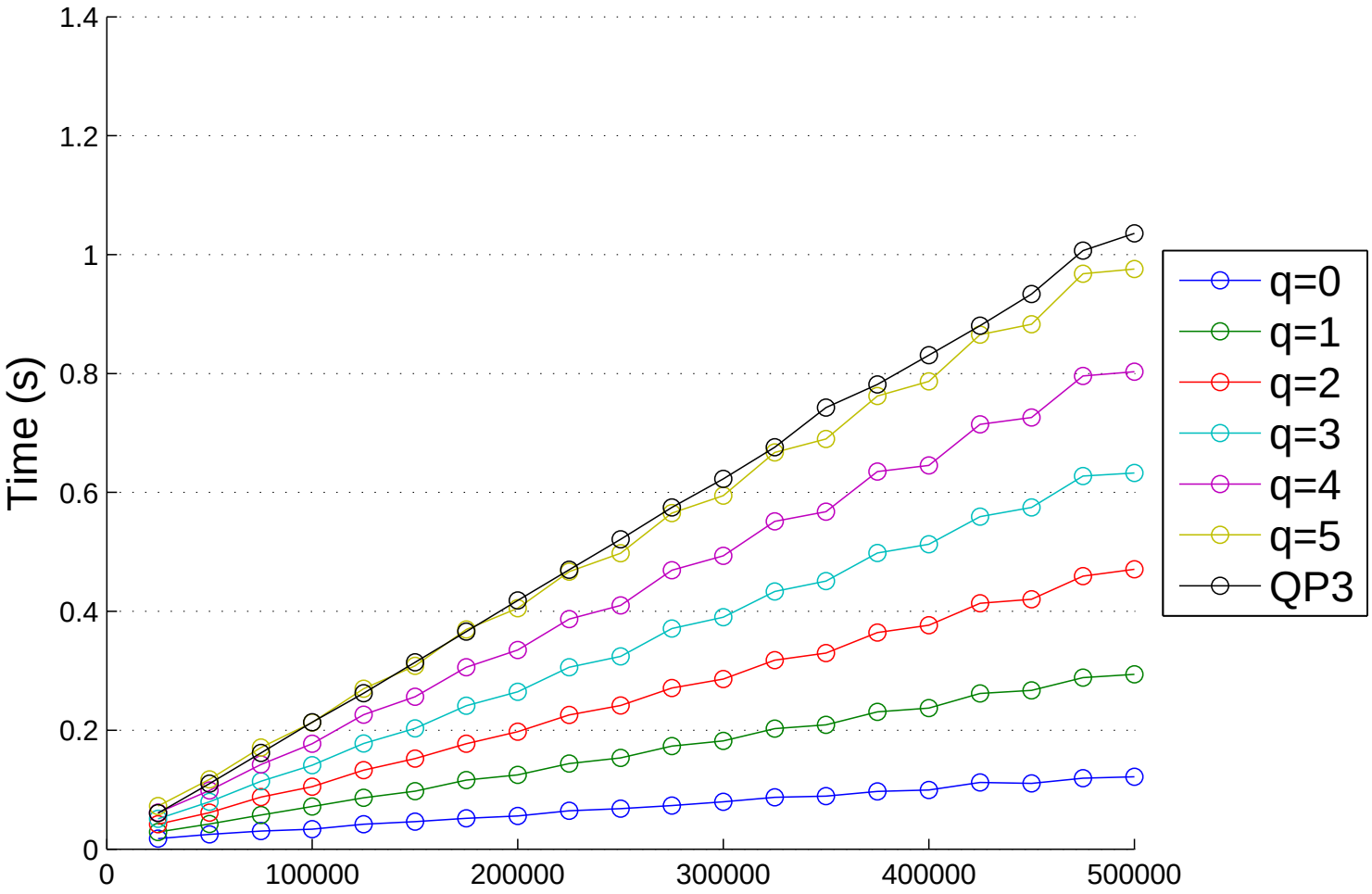
$(m = 500,000;\ n = 500;\ \ell = k + p = 50 + 10;\ q = 1)$

Time on 1,2,3 GPUs



$(m = 500,000;\ n = 500;\ \ell = k + p = 50 + 10;\ q = 1)$

Time with increasing number of iterations



$(m = 25,000 \to 500,000;\ n = 500;\ \ell = k + p = 50 + 10;\ q = 0 \to 5)$

Conclusion

Summary

- Substitute QP3 with Randomized, which is dominated by matrix-matrix multiplications $\Rightarrow$ very nice properties: data locality, higher parallelism, no synchronizations, minimized communications.
- Comparable accuracy on the test matrices used.
- Performance speedups above 5 and scaling on multiple GPUs.

For the future

- Use other test matrices, with more difficult properties ($\kappa(A) > 10^8 \Rightarrow$ possible mixed-precision CholQR application).
- Use other error measurements (applications e.g. clustering).
- Use FFT instead of GEMM.
- Test (and optimize) scalability for greater number of GPUs.
- Compare against a multiGPU QP3 (CA-QP3?) on multi-GPU.

Thank you and farewell! :)