On Parallelizing the MRRR Algorithm for Data-Parallel Coprocessors.

Christian Lessig, Dynamic Graphics Project, University of Toronto.

Paolo Bientinesi, AICES, RWTH Aachen.

Symmetric Eigenproblem

- Matrix form:

$$\mathbf{T}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}$$

Symmetric Eigenproblem

- Matrix form:

$$\mathbf{T}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda}$$

- Vector form:

$$\mathbf{T}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Numerical Symmetric Eigenproblem

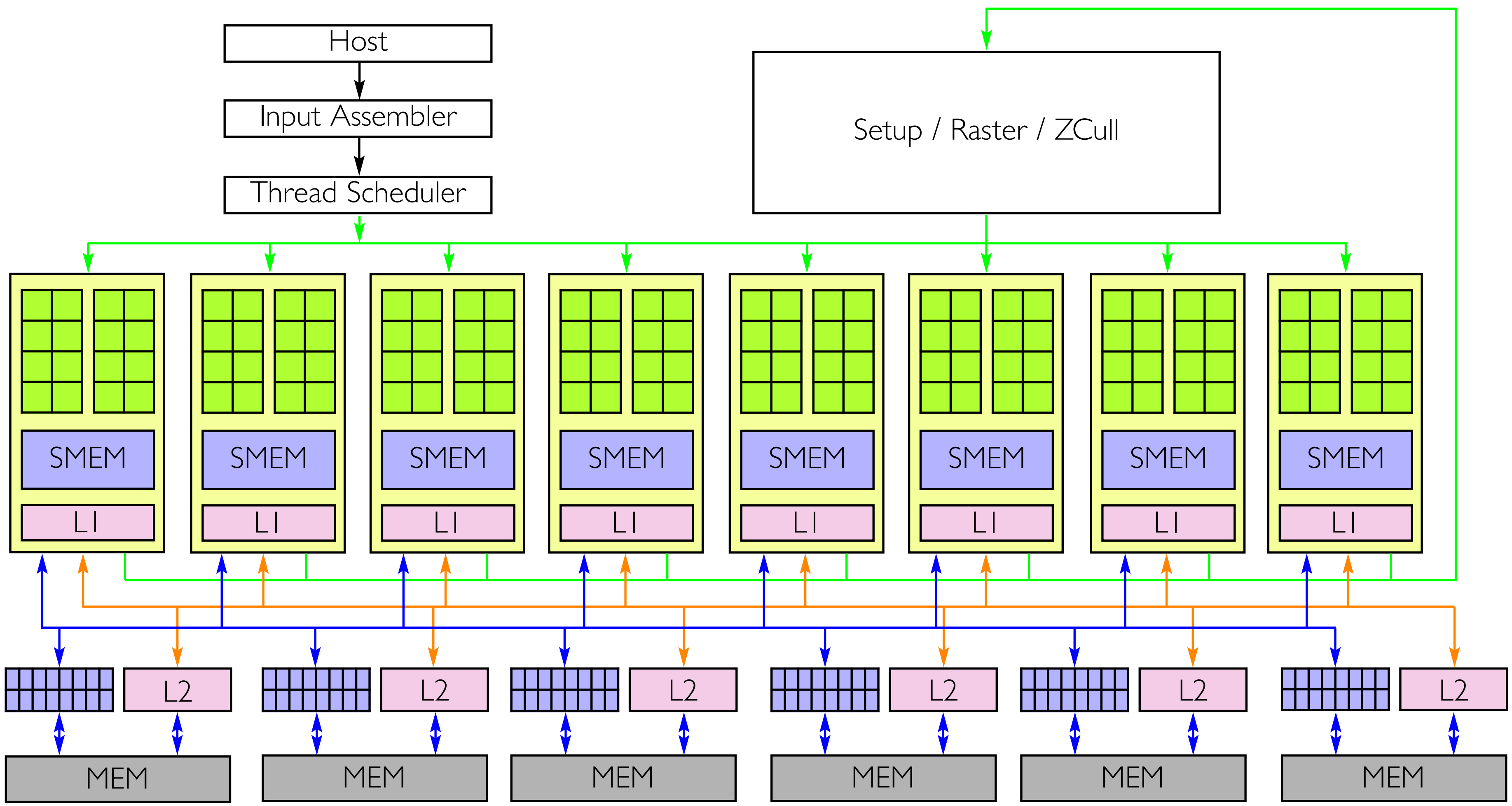
- Small residual:

$$\|\mathbf{T}\tilde{\mathbf{U}} - \tilde{\mathbf{U}}\tilde{\Lambda}\| = \mathcal{O}\left(n\,\epsilon\,\|\mathbf{T}\|\right)$$

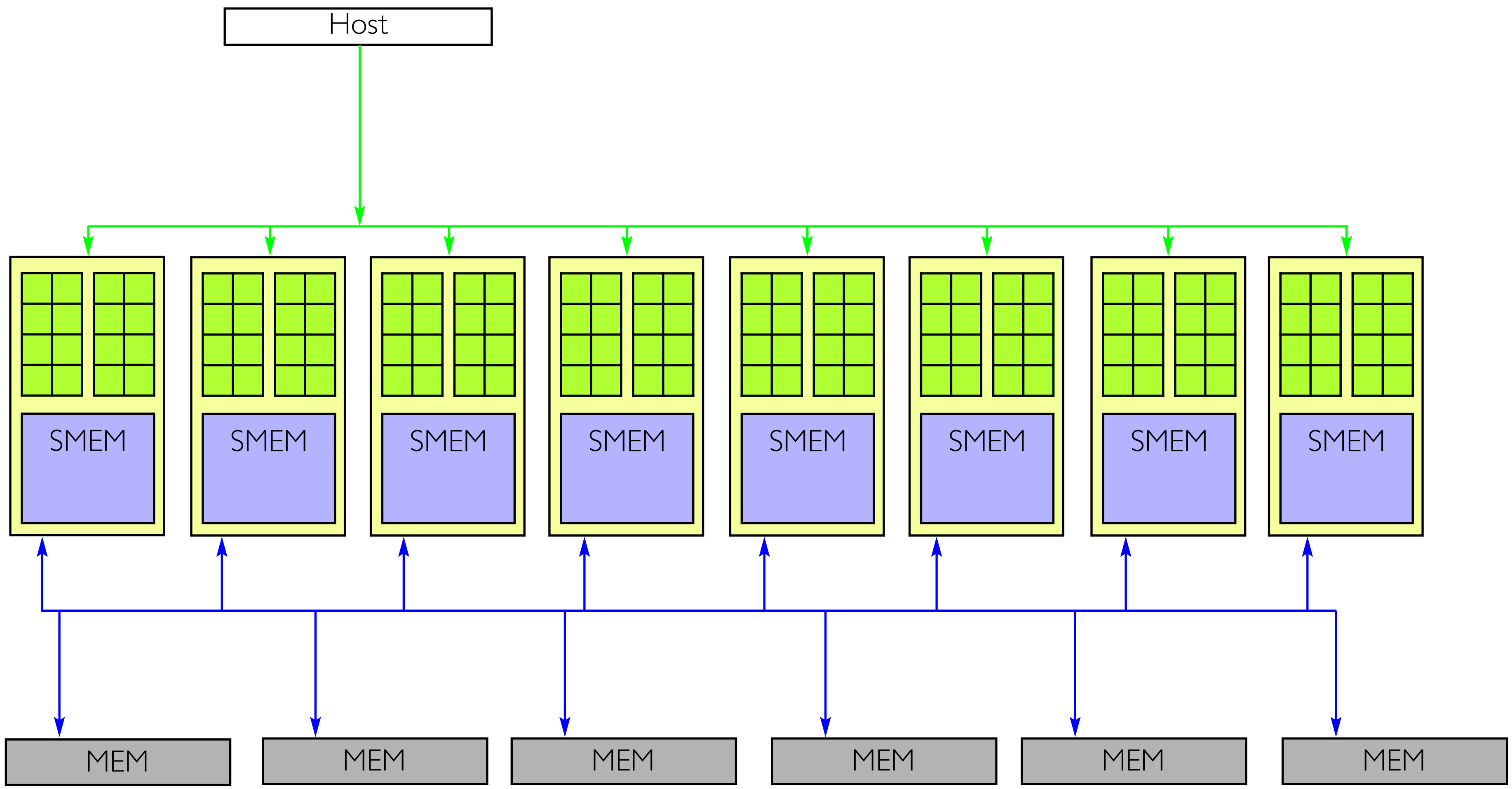
- Orthogonality of the eigenvectors:

$$\|\tilde{\mathbf{U}}^T\tilde{\mathbf{U}} - \mathbf{I}\| = \mathcal{O}\left(n\,\epsilon\right)$$

Data-Parallel Coprocessors



Data-Parallel Coprocessors



The MRRR Algorithm

The MRRR Algorithm

- Well separated:

$$\lambda_i \;\; : \;\; \min_{\lambda_j} \left(\mathrm{reldist}(\lambda_i, \lambda_j) \right) > \delta_c$$

The MRRR Algorithm

- Well separated:

$$\lambda_i \; : \; \min_{\lambda_j} \left(\mathrm{reldist}(\lambda_i, \lambda_j) \right) > \delta_c$$

- Relative distance:

$$\mathrm{reldist}(\lambda_i, \lambda_j) = \frac{|\lambda_i - \lambda_j|}{|\lambda_i|}$$

The MRRR Algorithm

- Matrix shifts:

$$\hat{\mathbf{T}} = \mathbf{T} - \sigma \mathbf{I}$$

The MRRR Algorithm

- Matrix shifts:

$$\hat{\mathbf{T}} = \mathbf{T} - \sigma\mathbf{I}$$

- Matrix shifts using Relatively Robust Representations (RRR's):

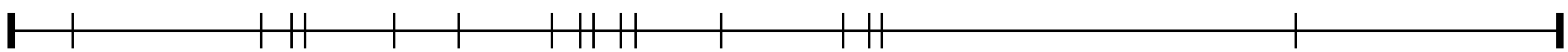
$$\hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^T = \mathbf{L}\mathbf{D}\mathbf{L}^T - \sigma\mathbf{I}$$

The MRRR Algorithm

$$\boxed{\mathbf{LDL}^T_{1,1}}$$

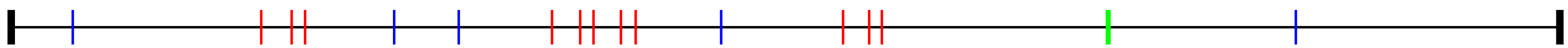
The MRRR Algorithm

$$\mathbf{LDL}^T_{1,1}$$

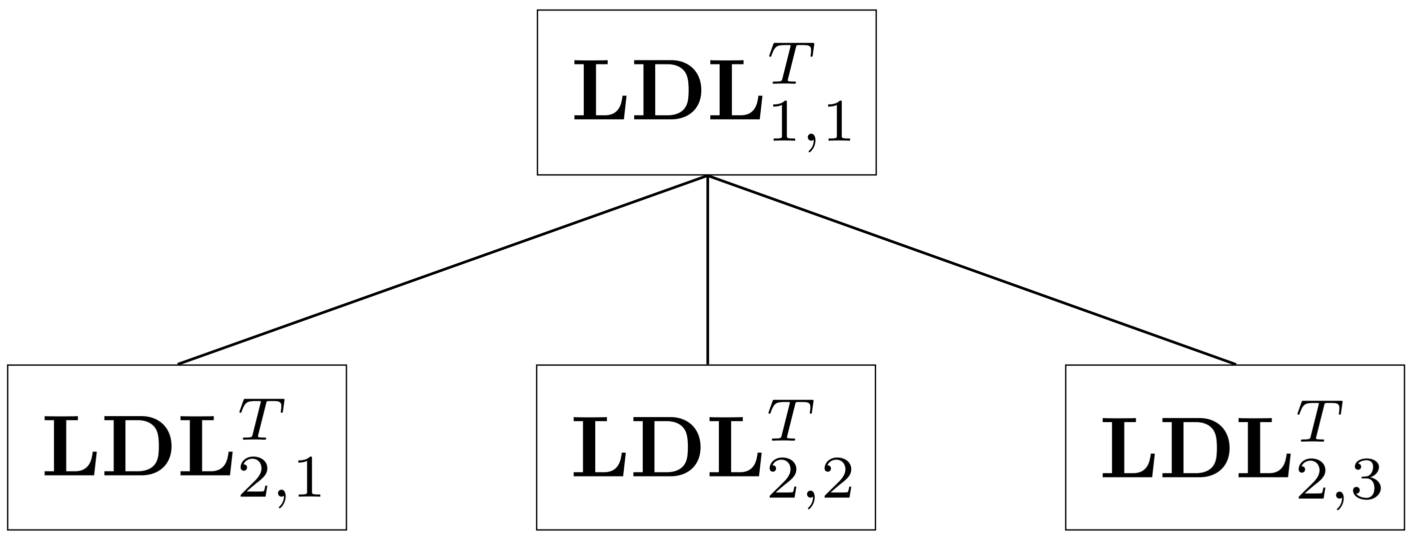


The MRRR Algorithm

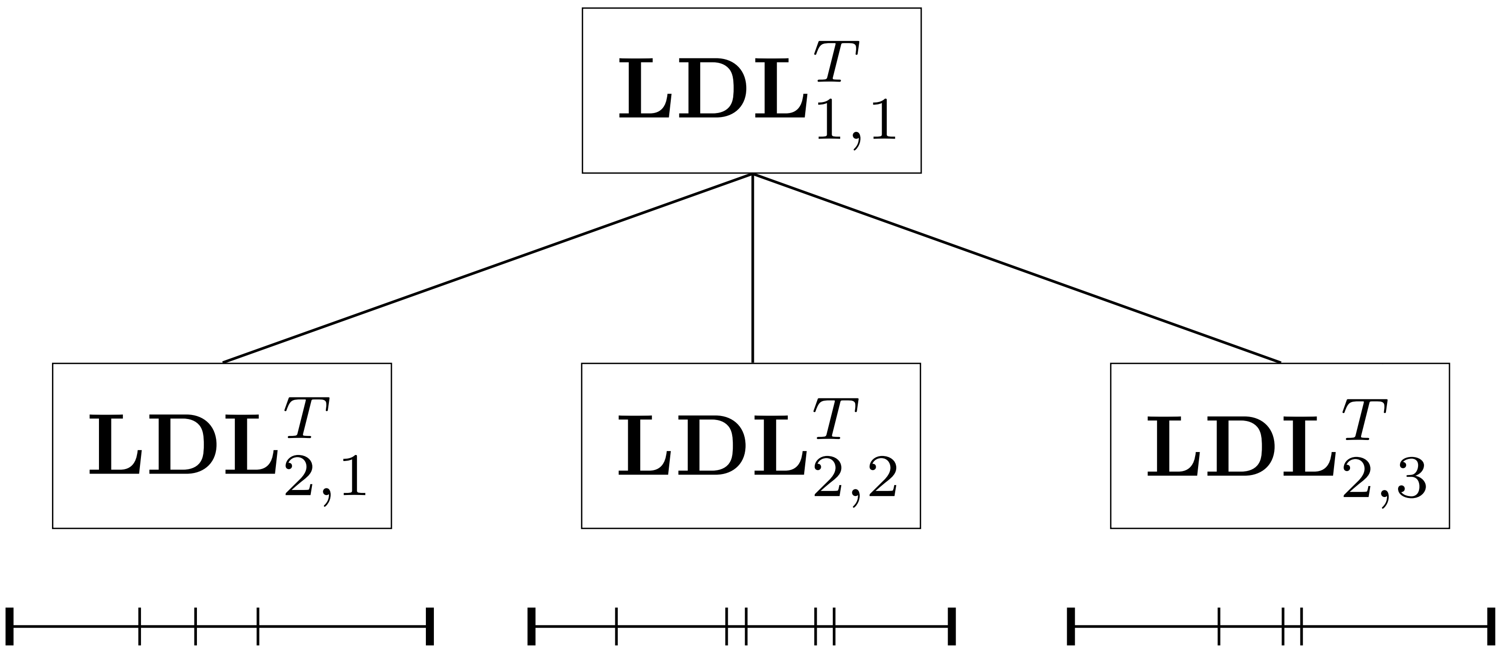
$$\boxed{\mathbf{LDL}^T_{1,1}}$$



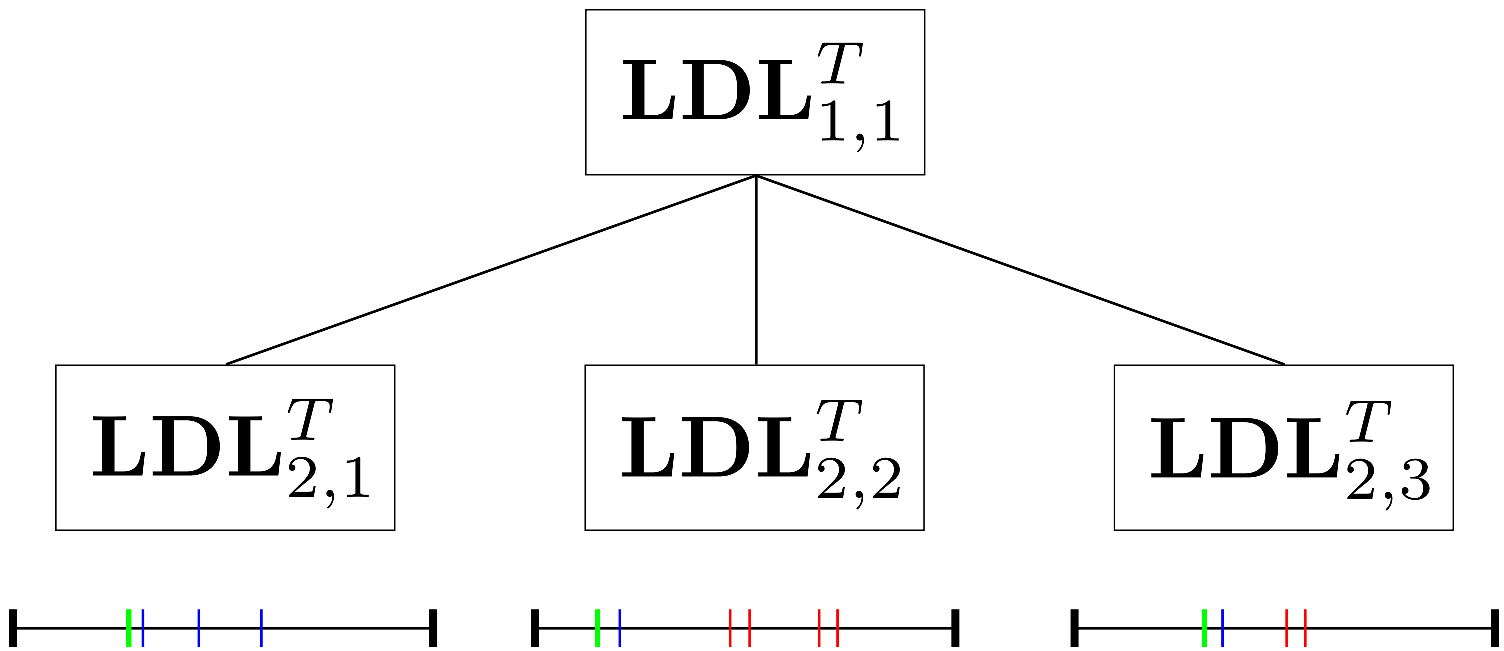
The MRRR Algorithm



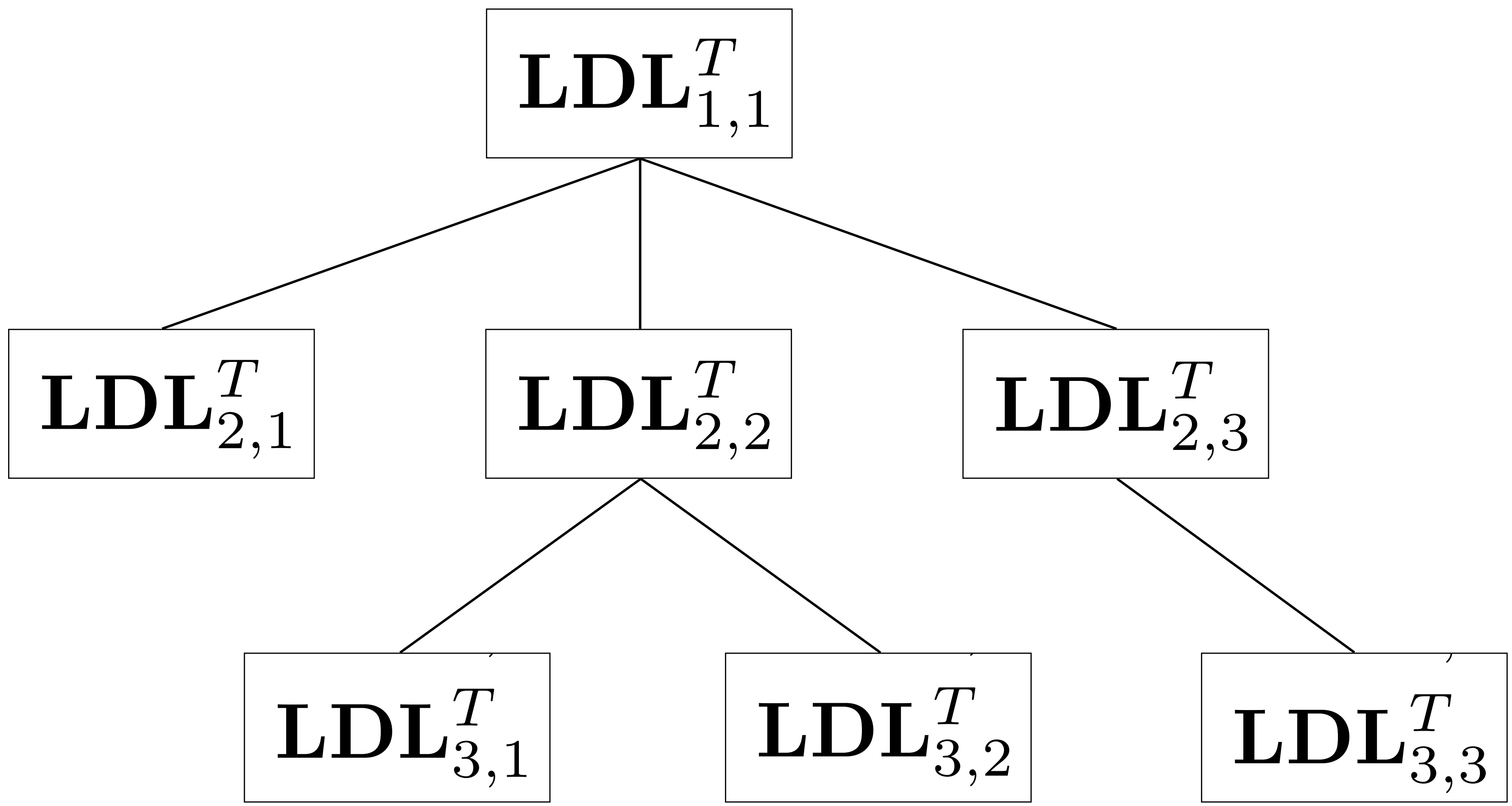
The MRRR Algorithm



The MRRR Algorithm



The MRRR Algorithm



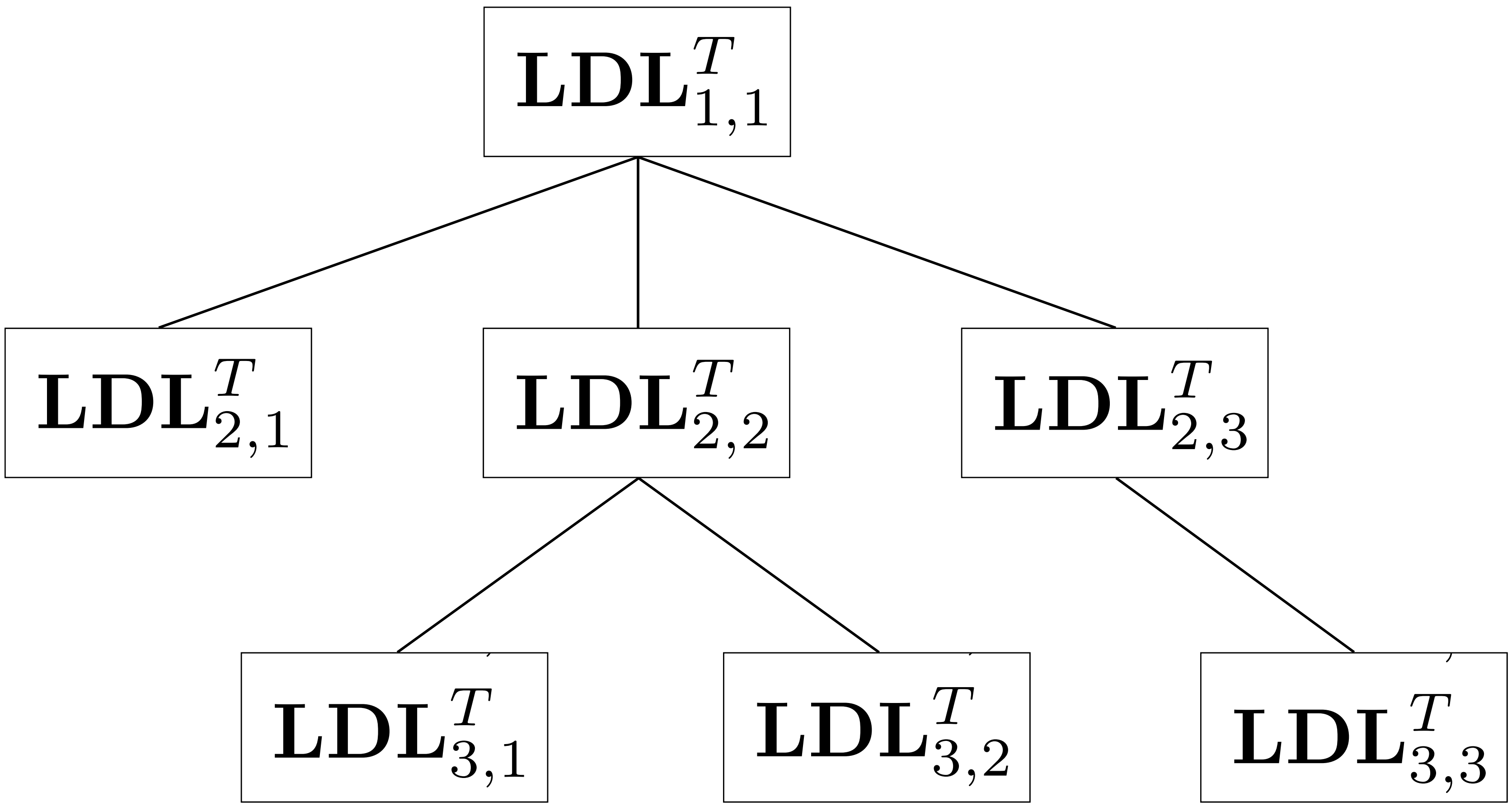
The MRRR Algorithm

For each node *(l,m)* of the representation tree:

1. Classify eigenvalues as singletons or clustered.
2. Compute eigenpairs for singletons.
3. Compute a shifted matrix and create a new tree node *(l+1,m)* for every cluster.

Parallelizing the MRRR Algorithm

Parallelizing the MRRR Algorithm



Parallelizing the MRRR Algorithm

For each node *(l,m)* of the representation tree:

1. Classify eigenvalues as singletons or clustered.
2. Compute eigenpairs for singletons.
3. Compute a shifted matrix and create a new tree node *(l+1,m)* for every cluster.

Parallelizing the MRRR Algorithm

For each node *(l,m)* of the representation tree:

1. ***For each*** eigenvalue, classify it as singletons or part of a cluster.
2. ***For each*** singleton, compute the eigenpair.
3. ***For each*** cluster, compute a shifted matrix and create a representation tree node *(l+1,m)*.

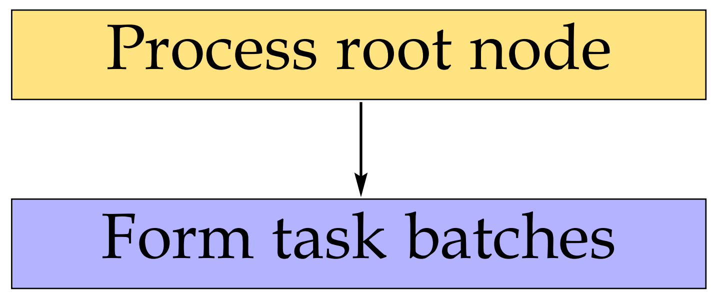
Parallelizing the MRRR Algorithm

- **Task parallelism:** Representation tree allows to process nodes on the same level or in different subtrees independently.
- **Data parallelism:** Computations per cluster are data-parallel.

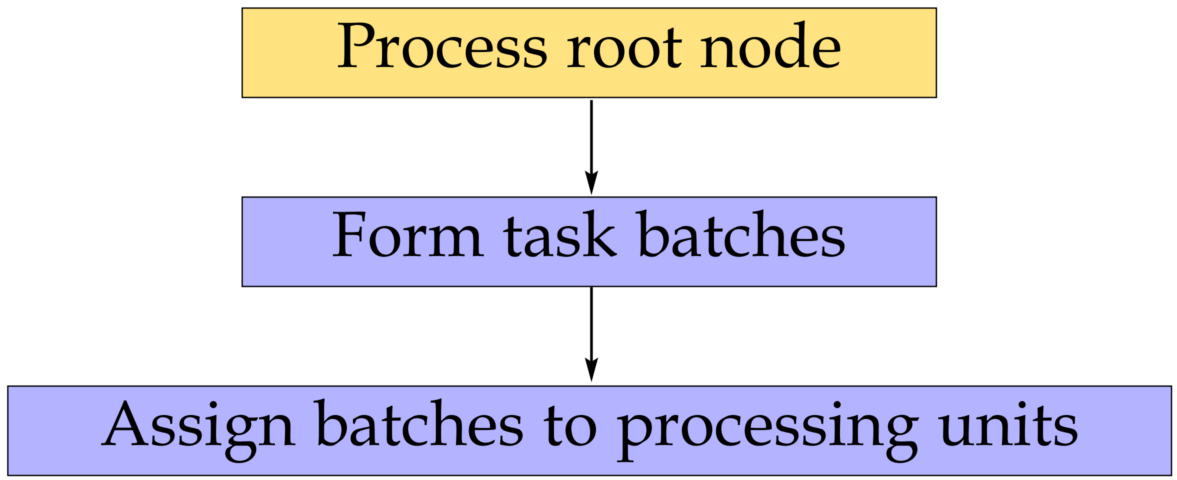
Implementation

Process root node

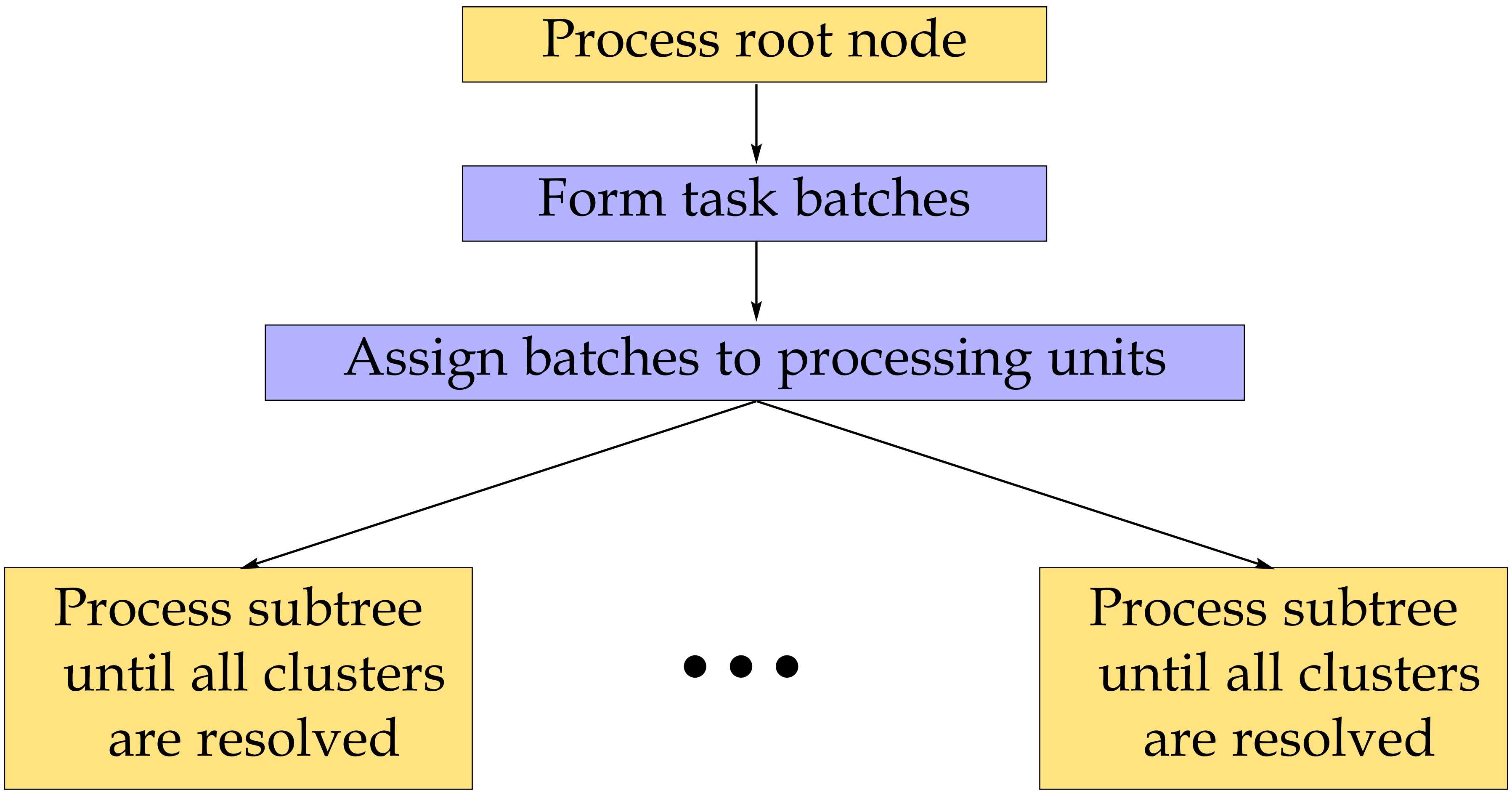
Implementation



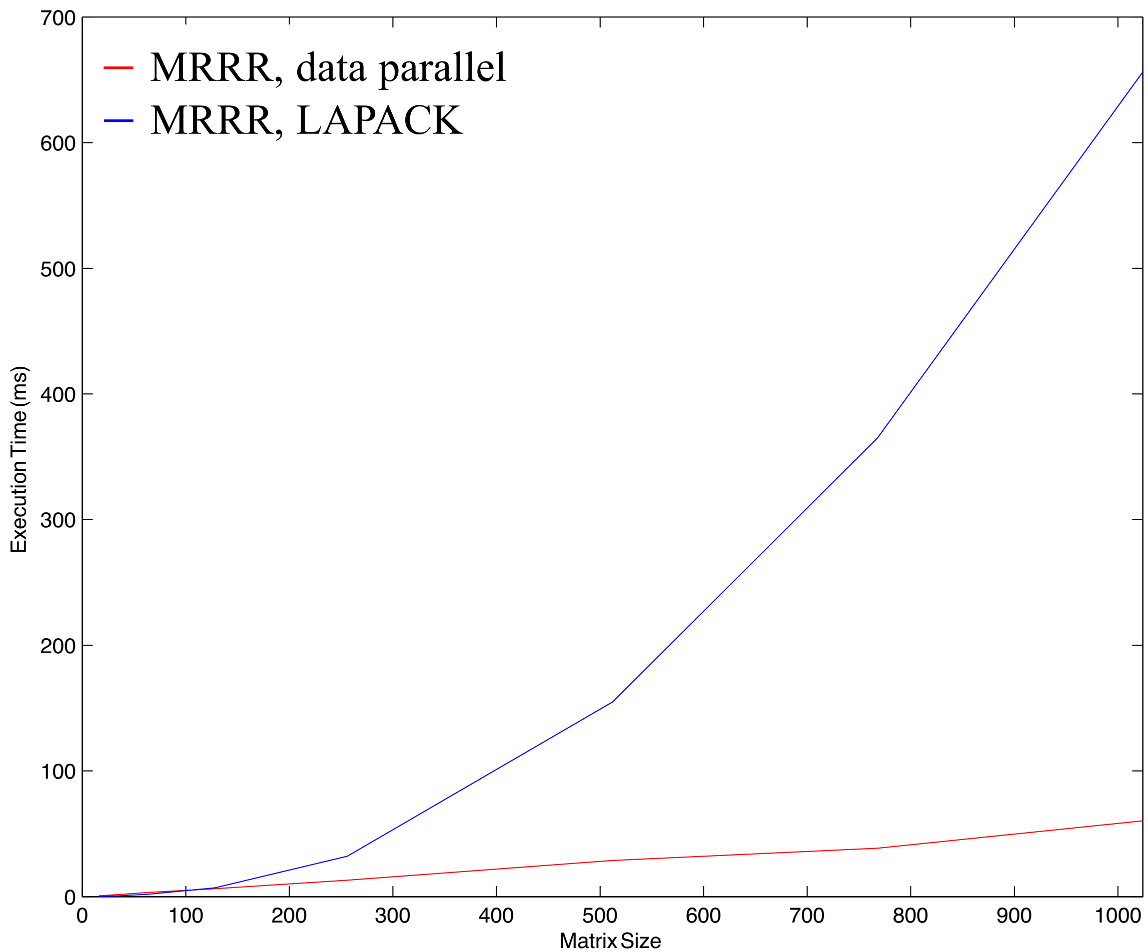
Implementation



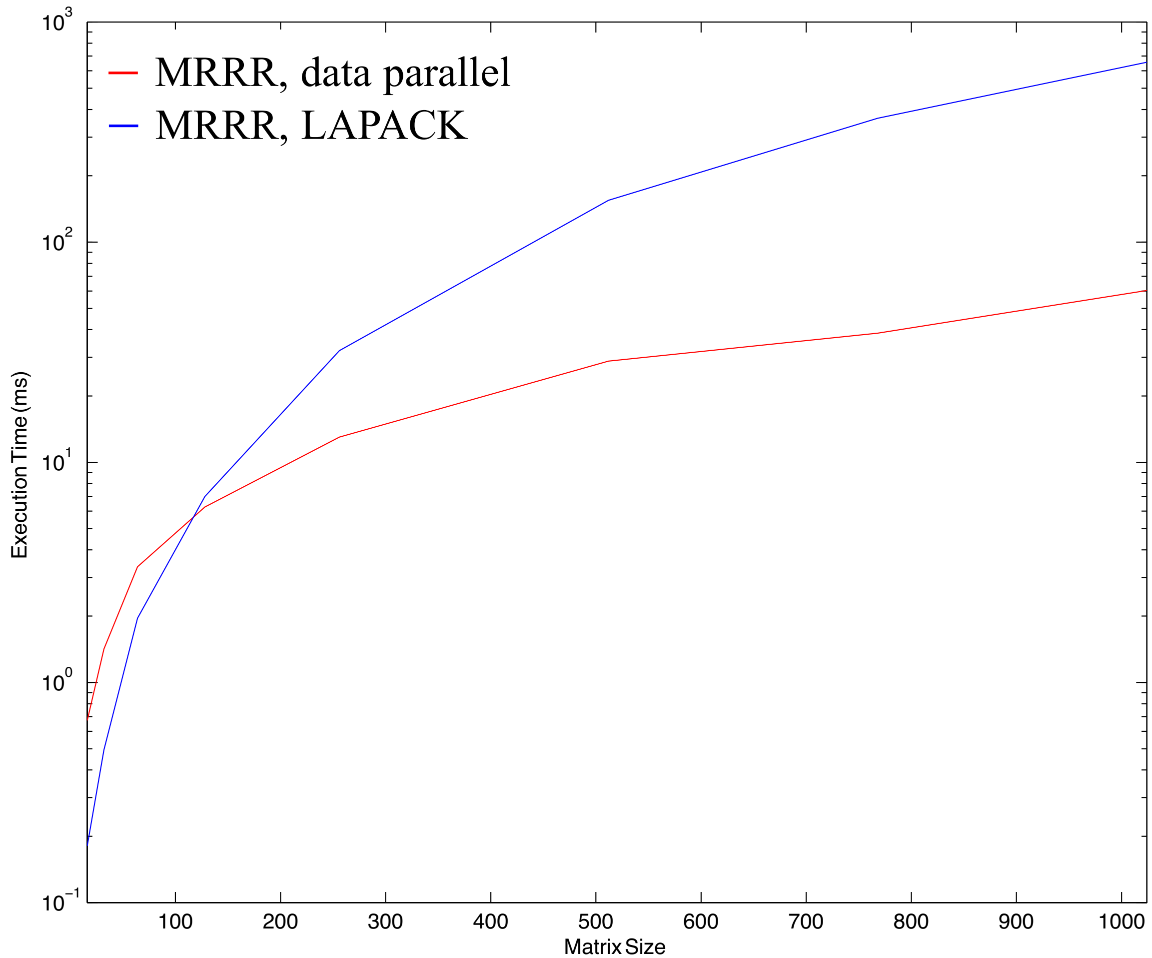
Implementation



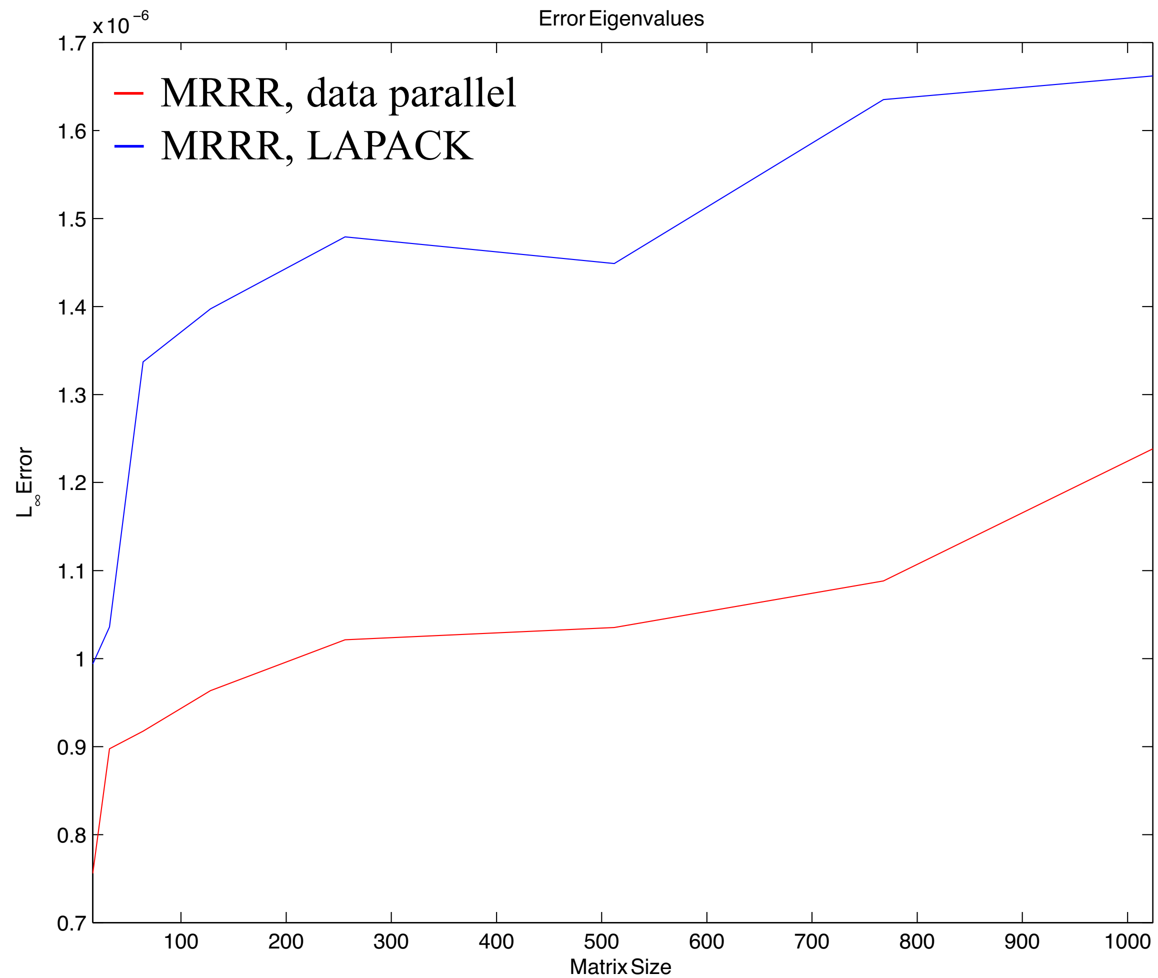
Results


700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000
Execution Time (ms)
Matrix Size
MRRR, data parallel
MRRR, LAPACK

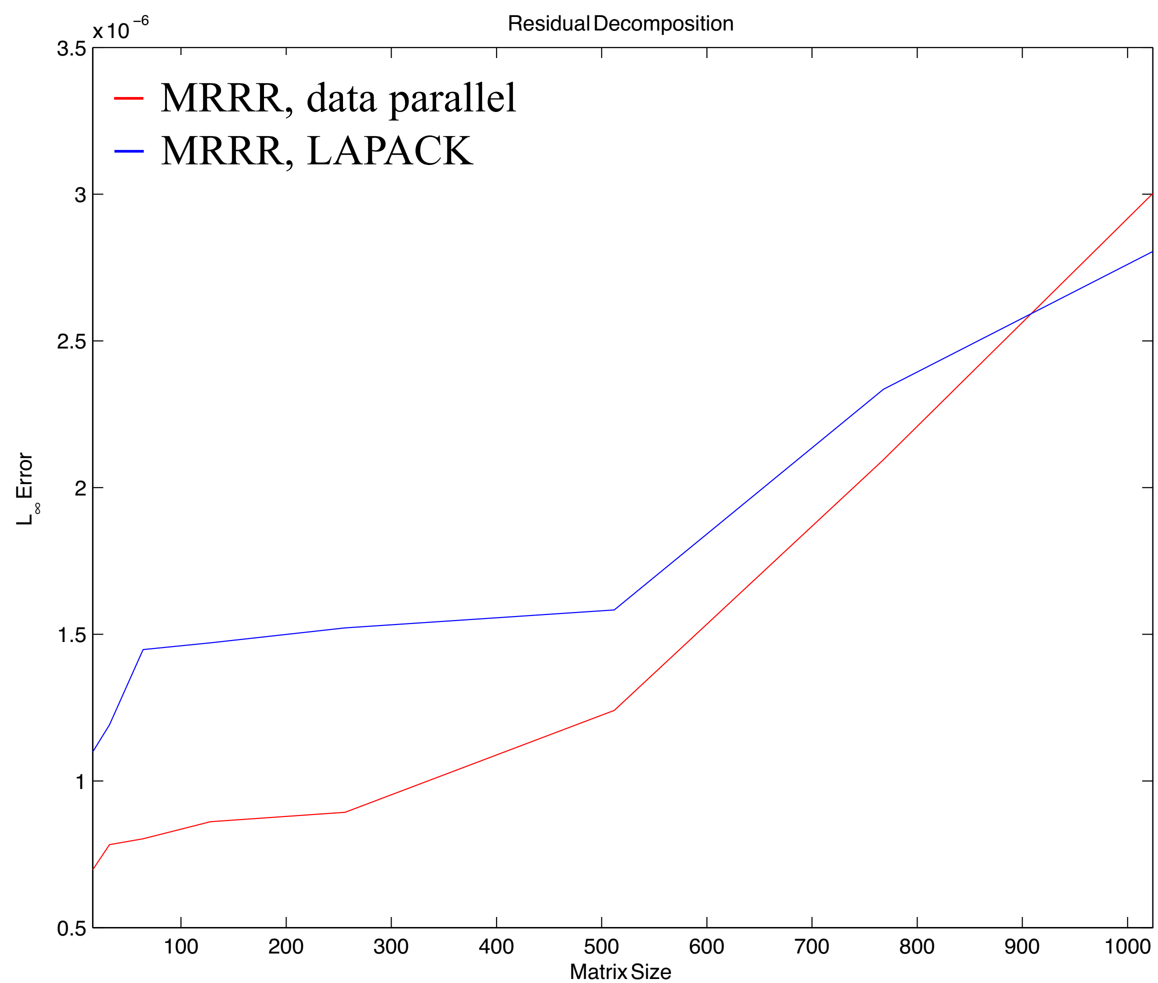
Results


MRRR, data parallel
MRRR, LAPACK
Execution Time (ms)
Matrix Size
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
100
200
300
400
500
600
700
800
900
1000

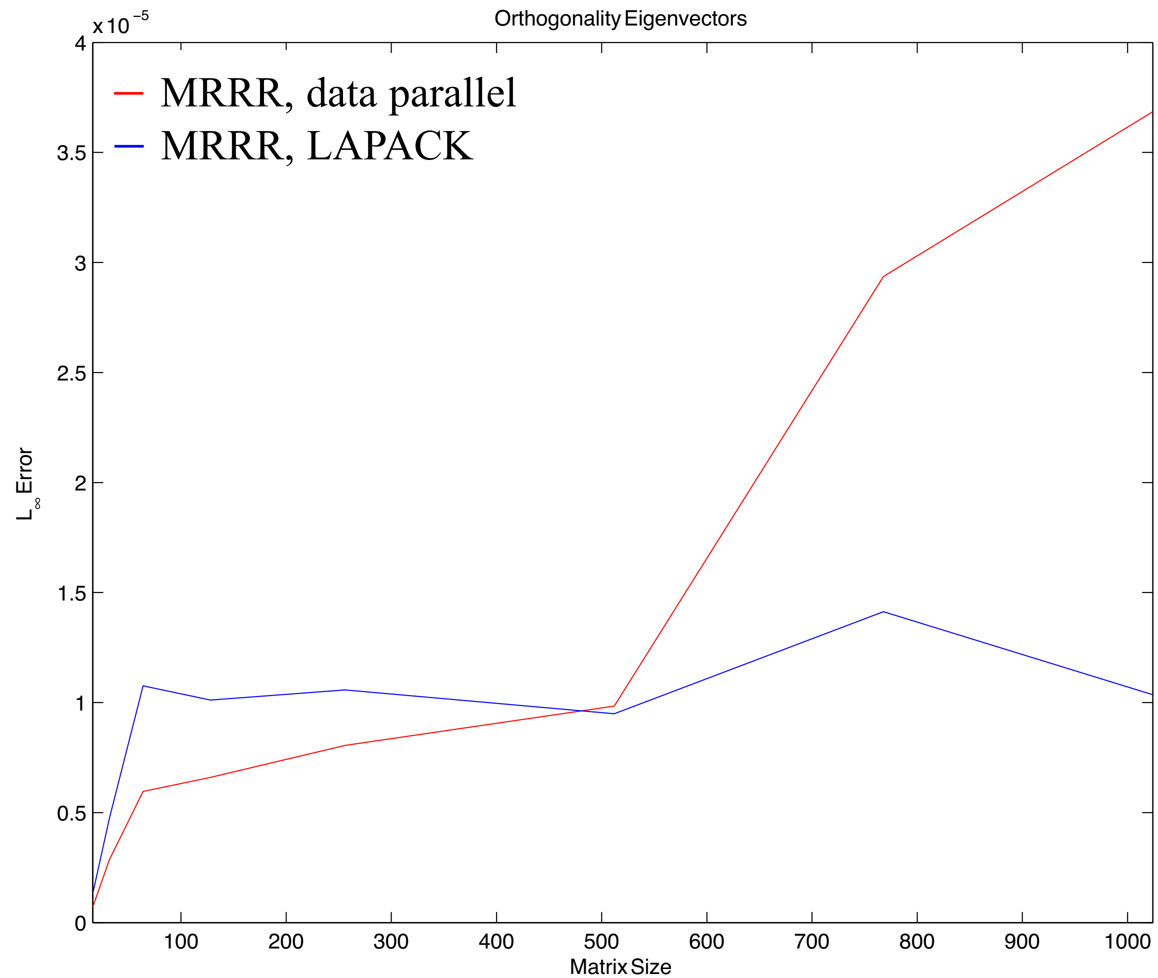
Results


Error Eigenvalues
x 10^-6
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
L∞ Error
100
200
300
400
500
600
700
800
900
1000
Matrix Size
MRRR, data parallel
MRRR, LAPACK

Results


Residual Decomposition
x 10^-6
3.5
3
2.5
2
1.5
1
0.5
L∞ Error
100
200
300
400
500
600
700
800
900
1000
Matrix Size
MRRR, data parallel
MRRR, LAPACK

Results


Orthogonality Eigenvectors
x 10^-5
MRRR, data parallel
MRRR, LAPACK
L∞ Error
Matrix Size
0
0.5
1
1.5
2
2.5
3
3.5
4
100
200
300
400
500
600
700
800
900
1000

Conclusion

- MRRR algorithm can be mapped efficiently onto data parallel coprocessors.
  - Representation tree provides task parallelism.
  - Computations for each tree node provide data parallelism.

Conclusion

- MRRR algorithm can be mapped efficiently onto data parallel coprocessors.
  - Representation tree provides task parallelism.
  - Computations for each tree node provide data parallelism.
- Significant speedups over single-threaded CPU implementation possible.

Open Problems and Future Work

- Resolve remaining accuracy issues.
- Load balancing between processing units.
  - For nodes on the first representation tree level.
  - Load balancing at every level of the representation tree?
- Load balancing between host and device.
- Port to OpenCL and Larrabee.

More details:

www.dgp.toronto.edu/people/lessig/mrrr/