Profiling High Performance Dense Linear Algebra Algorithms on Multicore Architectures for Power and Energy Efficiency

Hatem Ltaief¹  Piotr Łuszczek²  Jack Dongarra²

¹KAUST Supercomputing Laboratory
Thuwal, Saudi Arabia

²Innovative Computing Laboratory
University of Tennessee Knoxville

EnaHPC’11 Conference Hamburg, Germany
Outline

1. The "K" Computer
2. A Look Back...
3. LAPACK: Block Algorithms
4. PLASMA: Tile Algorithms
5. Power Analysis
6. Summary and Future Work
Outline

1. The "K" Computer
2. A Look Back...
3. LAPACK: Block Algorithms
4. PLASMA: Tile Algorithms
5. Power Analysis
6. Summary and Future Work
1. The "K" Computer
2. A Look Back...
3. LAPACK: Block Algorithms
4. PLASMA: Tile Algorithms
5. Power Analysis
6. Summary and Future Work
Outline

1. The "K" Computer
2. A Look Back...
3. LAPACK: Block Algorithms
4. PLASMA: Tile Algorithms
5. Power Analysis
6. Summary and Future Work
Outline

1. The "K" Computer
2. A Look Back...
3. LAPACK: Block Algorithms
4. PLASMA: Tile Algorithms
5. Power Analysis
6. Summary and Future Work
Motivations

**K computer Specifications**

<table>
<thead>
<tr>
<th>CPU (SPARC64 VIIIfx)</th>
<th>Cores/Node</th>
<th>8 cores (@2GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>128GFlops</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>SPARC V9 + HPC extension</td>
<td></td>
</tr>
<tr>
<td>Cache</td>
<td>L1 (I/D) Cache : 32KB/32KB L2 Cache : 6MB</td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>58W (typ. 30°C)</td>
<td></td>
</tr>
<tr>
<td>Mem. bandwidth</td>
<td>64GB/s.</td>
<td></td>
</tr>
<tr>
<td>Node</td>
<td>Configuration</td>
<td>1 CPU/Node</td>
</tr>
<tr>
<td></td>
<td>Memory capacity</td>
<td>16GB (2GB/core)</td>
</tr>
<tr>
<td>System board (SB)</td>
<td>No. of nodes</td>
<td>4 nodes/SB</td>
</tr>
<tr>
<td>Rack</td>
<td>No. of SB</td>
<td>24 SBs/rack</td>
</tr>
<tr>
<td>System</td>
<td>Nodes/system</td>
<td>&gt; 80,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inter-connect</th>
<th>Topology</th>
<th>6D Mesh/Torus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>5GB/s. for each link</td>
<td></td>
</tr>
<tr>
<td>No. of link</td>
<td>10 links/node</td>
<td></td>
</tr>
<tr>
<td>Additional feature</td>
<td>H/W barrier, reduction</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>Routing chip structure (no outside switch box)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cooling</th>
<th>CPU, ICC*</th>
<th>Direct water cooling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other parts</td>
<td>Air cooling</td>
<td></td>
</tr>
</tbody>
</table>

**System**
LINPACK 10 PFlops over 1PB mem.
800 racks
80,000 CPUs
640,000 cores

**Node**
128 GFlops
16GB Memory
64 GB memory

**System Board**
512 GFlops
12.3 TFlops
15TB memory

* ICC : Interconnect Chip
Motivations

- 10 MW needed to feed the baby
- Exascale roadmap says up to 20 MW
- Huge challenge: achieving 2 orders of magnitude in performance by only doubling the power rate
- Co-designed Hardware and Software solutions
Motivations

- 10 MW needed to feed the baby
- Exascale roadmap says up to 20 MW
- Huge challenge: achieving 2 orders of magnitude in performance by only doubling the power rate
- Co-designed Hardware and Software solutions
Motivations

- 10 MW needed to feed the baby
- Exascale roadmap says up to 20 MW
- Huge challenge: achieving 2 orders of magnitude in performance by only doubling the power rate
- Co-designed Hardware and Software solutions
Motivations

- 10 MW needed to feed the baby
- Exascale roadmap says up to 20 MW
- Huge challenge: achieving 2 orders of magnitude in performance by only doubling the power rate
- Co-designed Hardware and Software solutions
Software infrastructure and algorithmic design follow hardware evolution in time:

- **70’s - LINPACK**, vector operations:  
  *Level-1 BLAS operation*

- **80’s - LAPACK**, block, cache-friendly:  
  *Level-3 BLAS operation*

- **90’s - ScaLAPACK**, distributed memory:  
  *PBLAS Message passing*

- **00’s - PLASMA**, many-cores friendly:  
  *DAG scheduler, tile data layout, some extra kernels*
Software infrastructure and algorithmic design follow hardware evolution in time:

- **70’s - LINPACK, vector operations:**
  *Level-1 BLAS operation*

- **80’s - LAPACK, block, cache-friendly:**
  *Level-3 BLAS operation*

- **90’s - ScaLAPACK, distributed memory:**
  *PBLAS Message passing*

- **00’s:**
  - **PLASMA, many-cores friendly:**
    *DAG scheduler, tile data layout, some extra kernels*
Software infrastructure and algorithmic design follow hardware evolution in time:

- **70’s - LINPACK, vector operations:**
  - *Level-1 BLAS operation*

- **80’s - LAPACK, block, cache-friendly:**
  - *Level-3 BLAS operation*

- **90’s - ScaLAPACK, distributed memory:**
  - *PBLAS Message passing*

- **00’s:**
  - PLASMA, many-cores friendly:
    - *DAG scheduler, tile data layout, some extra kernels*
Software infrastructure and algorithmic design follow hardware evolution in time:

- **70’s - LINPACK, vector operations:**
  *Level-1 BLAS operation*

- **80’s - LAPACK, block, cache-friendly:**
  *Level-3 BLAS operation*

- **90’s - ScaLAPACK, distributed memory:**
  *PBLAS Message passing*

- **00’s:**
  - **PLASMA, many-cores friendly:**
    *DAG scheduler, tile data layout, some extra kernels*
Principles

- Panel-Update Sequence
  - Transformations are blocked/accumulated within the Panel (Level 2 BLAS)
  - Transformations applied at once on the trailing submatrix (Level 3 BLAS)
  - Parallelism hidden inside the BLAS
  - Fork-join Model
Panel-Update Sequence

Transformations are blocked/accumulated within the Panel (Level 2 BLAS)

Transformations applied at once on the trailing submatrix (Level 3 BLAS)

Parallelism hidden inside the BLAS

Fork-join Model
Principles

- Panel-Update Sequence
- Transformations are blocked/accumulated within the Panel (Level 2 BLAS)
- Transformations applied at once on the trailing submatrix (Level 3 BLAS)
- Parallelism hidden inside the BLAS
- Fork-join Model
Principles

- Panel-Update Sequence
- Transformations are blocked/accumulated within the Panel (Level 2 BLAS)
- Transformations applied at once on the trailing submatrix (Level 3 BLAS)
- Parallelism hidden inside the BLAS
- Fork-join Model
Principles

- Panel-Update Sequence
- Transformations are blocked/accumulated within the Panel (Level 2 BLAS)
- Transformations applied at once on the trailing submatrix (Level 3 BLAS)
- Parallelism hidden inside the BLAS
- Fork-join Model
LU, QR, Cholesky

(a) First step.

(b) Second step.

(c) Third step.

Figure: Panel-update sequences for the LAPACK one-sided factorizations.
Hessenberg, TRD and BRD

Figure: Panel-update sequences for the LAPACK two-sided transformations.
Fork-Join Paradigm
PLASMA: Parallel Linear Algebra for Scalable Multi-core Architectures

- Parallelism is brought to the fore
- May require the redesign of linear algebra algorithms
- Tile data layout translation
- Remove unnecessary synchronization points between Panel-Update sequences
- DAG execution where nodes represent tasks and edges define dependencies between them
- Dynamic runtime system environment QUARK

⇒ http://icl.cs.utk.edu/plasma/
PLASMA: Parallel Linear Algebra for Scalable Multi-core Architectures

⇒ http://icl.cs.utk.edu/plasma/

- Parallelism is brought to the fore
- May require the redesign of linear algebra algorithms
- Tile data layout translation
- Remove unnecessary synchronization points between Panel-Update sequences
- DAG execution where nodes represent tasks and edges define dependencies between them
- Dynamic runtime system environment QUARK
PLASMA: Parallel Linear Algebra for Scalable Multi-core Architectures

⇒ http://icl.cs.utk.edu/plasma/

- Parallelism is brought to the fore
- May require the redesign of linear algebra algorithms
- Tile data layout translation
  - Remove unnecessary synchronization points between Panel-Update sequences
  - DAG execution where nodes represent tasks and edges define dependencies between them
  - Dynamic runtime system environment QUARK
PLASMA: Parallel Linear Algebra for Scalable Multi-core Architectures

⇒ http://icl.cs.utk.edu/plasma/

- Parallelism is brought to the fore
- May require the redesign of linear algebra algorithms
- Tile data layout translation
- Remove unnecessary synchronization points between Panel-Update sequences
  - DAG execution where nodes represent tasks and edges define dependencies between them
- Dynamic runtime system environment QUARK
Parallelism is brought to the fore

May require the redesign of linear algebra algorithms

Tile data layout translation

Remove unnecessary synchronization points between Panel-Update sequences

DAG execution where nodes represent tasks and edges define dependencies between them

Dynamic runtime system environment QUARK
PLASMA: Parallel Linear Algebra for Scalable Multi-core Architectures

⇒ http://icl.cs.utk.edu/plasma/

- Parallelism is brought to the fore
- May require the redesign of linear algebra algorithms
- Tile data layout translation
- Remove unnecessary synchronization points between Panel-Update sequences
- DAG execution where nodes represent tasks and edges define dependencies between them
- Dynamic runtime system environment QUARK
Data Layout Format

LAPACK: column-major format

PLASMA: tile format
Dynamic Scheduling QUARK

- Conceptually similar to out-of-order processor scheduling because it has:
  - Dynamic runtime DAG scheduler
  - Out-of-order execution flow of fine-grained tasks
  - Task scheduling as soon as dependencies are satisfied
  - Producer-Consumer
Dynamic Scheduling QUARK

- Conceptually similar to out-of-order processor scheduling because it has:
  - Dynamic runtime DAG scheduler
  - Out-of-order execution flow of fine-grained tasks
  - Task scheduling as soon as dependencies are satisfied
  - Producer-Consumer
Dynamic Scheduling QUARK

- Conceptually similar to out-of-order processor scheduling because it has:
  - Dynamic runtime DAG scheduler
  - Out-of-order execution flow of fine-grained tasks
  - Task scheduling as soon as dependencies are satisfied
  - *Producer-Consumer*
Dynamic Scheduling QUARK

- Conceptually similar to out-of-order processor scheduling because it has:
  - Dynamic runtime DAG scheduler
  - Out-of-order execution flow of fine-grained tasks
  - Task scheduling as soon as dependencies are satisfied
  - Producer-Consumer
Dynamic Scheduling QUARK

- Conceptually similar to out-of-order processor scheduling because it has:
  - Dynamic runtime DAG scheduler
  - Out-of-order execution flow of fine-grained tasks
  - Task scheduling as soon as dependencies are satisfied
  - *Producer-Consumer*
Dynamic Scheduling QUARK

- Conceptually similar to out-of-order processor scheduling because it has:
  - Dynamic runtime DAG scheduler
  - Out-of-order execution flow of fine-grained tasks
  - Task scheduling as soon as dependencies are satisfied
  - *Producer-Consumer*
PLASMA: Tile Algorithms

PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- **Dense Linear Algebra**
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- **Multicore Systems**
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra** for **Scalable Multi-core Architectures**
- Numerical software library
- **Dense Linear Algebra**
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- **Multicore Systems**
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- Parallel Linear Algebra for Scalable Multi-core Architectures
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
    - multi-socket
    - shared memory
    - possibly NUMA
PLASMA In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
PLASMA In a Nutshell

- Parallel Linear Algebra for Scalable Multi-core Architectures
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
Plasma: Tile Algorithms

Plasma In a Nutshell

- **Parallel Linear Algebra for Scalable Multi-core Architectures**
- Numerical software library
- Dense Linear Algebra
  - linear systems of equations
  - least square problems
  - singular value problems
  - eigenvalue problems
  - all precisions (S, D, C, Z)
  - Linux, Windows, Mac OS, AIX
- Multicore Systems
  - multicore
  - multi-socket
  - shared memory
  - possibly NUMA
AX = B

- **LU-based solver**: Gaussian elimination, non-symmetric
- Cholesky-based solver: symmetric positive definite
- \( LDL^T \)-based solver: non-symmetric positive definite
- QR/LQ-based solver: least squares
- Matrix inversion using LU and Cholesky (statistics)
- Tall and skinny factorizations using tree reductions
- Mixed precision iterative refinement
$\mathbf{AX} = \mathbf{B}$

- LU-based solver: Gaussian elimination, non-symmetric
- Cholesky-based solver: symmetric positive definite
- $LDLT$-based solver: non-symmetric positive definite
- QR/LQ-based solver: least squares
- Matrix inversion using LU and Cholesky (statistics)
- Tall and skinny factorizations using tree reductions
- Mixed precision iterative refinement
$AX = B$

- LU-based solver: Gaussian elimination, non-symmetric
- Cholesky-based solver: symmetric positive definite
- $LDLT$-based solver: non-symmetric positive definite
- QR/LQ-based solver: least squares
- Matrix inversion using LU and Cholesky (statistics)
- Tall and skinny factorizations using tree reductions
- Mixed precision iterative refinement
AX = B

- LU-based solver: Gaussian elimination, non-symmetric
- Cholesky-based solver: symmetric positive definite
- $LDL^T$-based solver: non-symmetric positive definite
- QR/LQ-based solver: least squares
- Matrix inversion using LU and Cholesky (statistics)
- Tall and skinny factorizations using tree reductions
- Mixed precision iterative refinement
PLASMA: Tile Algorithms

\[ AX = B \]

- LU-based solver: Gaussian elimination, non-symmetric
- Cholesky-based solver: symmetric positive definite
- \( LDL^T \)-based solver: non-symmetric positive definite
- QR/LQ-based solver: least squares
- Matrix inversion using LU and Cholesky (statistics)
- Tall and skinny factorizations using tree reductions
- Mixed precision iterative refinement
\[ \mathbf{AX} = \mathbf{B} \]

- LU-based solver: Gaussian elimination, non-symmetric
- Cholesky-based solver: symmetric positive definite
- $LDL^T$-based solver: non-symmetric positive definite
- QR/LQ-based solver: least squares
- Matrix inversion using LU and Cholesky (statistics)
- Tall and skinny factorizations using tree reductions
- Mixed precision iterative refinement
\[ AX = B \]

- LU-based solver: Gaussian elimination, non-symmetric
- Cholesky-based solver: symmetric positive definite
- \( LDL^T \)-based solver: non-symmetric positive definite
- QR/LQ-based solver: least squares
- Matrix inversion using LU and Cholesky (statistics)
- Tall and skinny factorizations using tree reductions
- Mixed precision iterative refinement
\[ A = X \land X^T \] and \[ A = U\Sigma V \]

- Symmetric Eigenvalue Problem (Two-stage reduction + QR Iteration)
- Singular Value Problem (Two-stage reduction + QR Algorithm)
- Generalized Symmetric Eigenvalue Problem... more later!
\[ A = X \wedge X^T \text{ and } A = U\Sigma V \]

- Symmetric Eigenvalue Problem (Two-stage reduction + QR Iteration)
- Singular Value Problem (Two-stage reduction + QR Algorithm)
- Generalized Symmetric Eigenvalue Problem... more later!
\[ A = X \wedge X^T \text{ and } A = U\Sigma V \]

- Symmetric Eigenvalue Problem (Two-stage reduction + QR Iteration)
- Singular Value Problem (Two-stage reduction + QR Algorithm)
- Generalized Symmetric Eigenvalue Problem... more later!
Machine Description

- dori.cs.vt.edu from Virginia Tech (K. Cameron)
- Cluster of 8 nodes
- Each node contains a dual core AMD Opteron dual processors with 6 GB RAM and each core has 1 MB cache.
- "Power to the people, not the chips" P. Luszczek
Machine Description

- dori.cs.vt.edu from Virginia Tech (K. Cameron)
- Cluster of 8 nodes
  - Each node contains a dual core AMD Opteron dual processors with 6 GB RAM and each core has 1 MB cache.
- ”Power to the people, not the chips” P. Luszczek
Machine Description

- dori.cs.vt.edu from Virginia Tech (K. Cameron)
- Cluster of 8 nodes
- Each node contains a dual core AMD Opteron dual processors with 6 GB RAM and each core has 1 MB cache.

"Power to the people, not the chips" P. Luszczek
Machine Description

- dori.cs.vt.edu from Virginia Tech (K. Cameron)
- Cluster of 8 nodes
- Each node contains a dual core AMD Opteron dual processors with 6 GB RAM and each core has 1 MB cache.
- "Power to the people, not the chips" P. Luszczek
Power Monitoring with PowerPack
Power Rate of LAPACK Cholesky

![Power Analysis Graph]

- **System**
- **CPU**
- **Memory**
- **Motherboard**
- **Fan**

*Power (Watts) vs. Time (seconds) for different system components.*
Power Rate of PLASMA Cholesky

![Power Analysis Diagram](image-url)

- System
- CPU
- Memory
- Motherboard
- Fan

Power (Watts) vs. Time (seconds)
Power Analysis

Power Rate of LAPACK QR

![Power Analysis Graph]

- **System**
- **CPU**
- **Memory**
- **Motherboard**
- **Fan**

Ltaief, Łuszczyński, Dongarra (KAUST, UTK)
Power Rate of PLASMA QR

![Graph showing power analysis for different components (System, CPU, Memory, Motherboard, Fan) over time (seconds)].

- **System**
- **CPU**
- **Memory**
- **Motherboard**
- **Fan**

**Power (Watts)** vs **Time (seconds)**

Ltaief, Łuszczek, Dongarra (KAUST, UTK)

Energy Profiling of DLA Algorithms

EnaHPC 2011
Power Analysis

Power Rate of PLASMA TRD

System
CPU
Memory
Motherboard
Fan

Power (Watts)

Time (seconds)

Energy Profiling of DLA Algorithms
EnaHPC 2011 23 / 28
Power Analysis

Power Rate of LAPACK BRD

![Graph showing power consumption over time for various components of a system.]
Power Analysis

Power Rate of PLASMA BRD

<table>
<thead>
<tr>
<th>Power (Watts)</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>CPU</td>
</tr>
<tr>
<td>Memory</td>
<td>Motherboard</td>
</tr>
<tr>
<td>Fan</td>
<td></td>
</tr>
</tbody>
</table>

Ltaief, Łuszczek, Dongarra (KAUST, UTK)

Energy Profiling of DLA Algorithms

EnaHPC 2011
QUARK and DVFS
What’s next?

- Non-symmetric eigenvalue problem
- Eigenvector computations
- RTE systems will have to do more than just scheduling (performing DVFS on-the-fly)
- Power analysis with MAGMA and DPLASMA
What’s next?

- Non-symmetric eigenvalue problem
- Eigenvector computations
  - RTE systems will have to do more than just scheduling (performing DVFS on-the-fly)
  - Power analysis with MAGMA and DPLASMA
What’s next?

- Non-symmetric eigenvalue problem
- Eigenvector computations
- RTE systems will have to do more than just scheduling (performing DVFS on-the-fly)
- Power analysis with MAGMA and DPLASMA
What’s next?

- Non-symmetric eigenvalue problem
- Eigenvector computations
- RTE systems will have to do more than just scheduling (performing DVFS on-the-fly)
- Power analysis with MAGMA and DPLASMA
Thank you!

Thank You!