Dealing with *Asymmetry* for Performance and Energy Efficiency in ARM big.LITTLE architectures

Enrique S. QUINTANA-ORTÍ
Motivation

- Moore’s law is alive, but Dennard’s scaling is over
Motivation

Welcome “dark silicon” and asymmetric architectures to tackle power/energy/utilization walls!

![Exynos 5422 System-on-Chip Diagram]

**Hardkernel Odroid XU3**
Samsung Exynos5422
Cortex-A15 quad core + Cortex-A7 quad core (sorry, also tiny GPU)
Motivation

- Fault tolerance
  - *Increase in #components with Moore’s Law:*
    - Sequoia @ LLNL (Petascale): MTBF is 1.5 days
    - Exascale requires increasing #components by $O(10^3)$
  - *Near-threshold voltage computing (NTVC)* may trade off energy for hardware errors
Motivation

- Numerical libraries for (dense) linear algebra (DLA)

Dealing with asymmetric multicore processors (AMPs):
- Energy efficiency
- Fault tolerance
Outline

- High performance BLAS (also multi-threaded)
- BLAS
- Dense matrix factorizations
- Fault tolerance (for energy efficiency)
Outline

- High performance BLAS (also multi-threaded)
- BLAS
- Dense matrix factorizations
- Fault tolerance (for energy efficiency)
High Performance BLAS

- Commercial libraries for BLAS

MKL
ACML
ESSL
CUBLAS
High Performance BLAS

- “Open” sw.: GotoBLAS, ATLAS, OpenBLAS, BLIS
High Performance BLAS

- “Open” sw.: GotoBLAS, ATLAS, OpenBLAS, BLIS
High Performance BLAS

- “Open” sw.: GotoBLAS, ATLAS, OpenBLAS, BLIS

![Graphs comparing performance of BLAS implementations](image-url)
High Performance BLAS

- **BLIS**
  - Software framework for instantiating high-performance BLAS-like dense linear algebra libraries
  - All BLAS: single/double, real/complex
  - New/modified/3-clause BSD license
  - https://code.google.com/p/blis/
High Performance BLAS: GEMM in BLIS

Loop 1: for $j_c = 0, \ldots, n - 1$ in steps of $n_c$
Loop 2: for $p_c = 0, \ldots, k - 1$ in steps of $k_c$
Loop 3: for $i_c = 0, \ldots, m - 1$ in steps of $m_c$

\[
C(i_c : i_c + m_c - 1, j_c : j_c + n_c - 1) \equiv C_c \; + = \; A_c \cdot B_c \quad \text{// Macro-kernel}
\]
High Performance BLAS: GEMM in BLIS
High Performance BLAS: GEMM in BLIS

Loop 4
for \( j_r = 0, \ldots, n_c - 1 \) in steps of \( n_r \)

Loop 5
for \( i_r = 0, \ldots, m_c - 1 \) in steps of \( m_r \)

\[
C_c(i_r : i_r + m_r - 1, j_r : j_r + n_r - 1) = A_c(i_r : i_r + m_r - 1, 0 : k_c - 1) \cdot B_c(0 : k_c - 1, j_r : j_r + n_r - 1)
\]
High Performance BLAS: GEMM in BLIS
High Performance BLAS: GEMM in BLIS
How to choose optimal blocking/register tiling parameters? 

$ m_c, n_c, k_c, m_r, n_r$

- Experimentally
  
  “BLIS: A Framework for Rapid Instantiation of BLAS Functionality”
  F. G. Van Zee, R. A. van de Geijn
  ACM Transactions on Mathematical Software (TOMS), Vol. 41(3), 2015
  http://www.cs.utexas.edu/users/flame

- Analytically

  “Analytical Modeling is Enough for High Performance BLIS”
  T. M. Low, F. D. Igual, T. M. Smith, E. S. Quintana-Ortí
  FLAME Working note #74. Submitted to ACM TOMS
  http://www.cs.utexas/edu/users/flame
### High Performance BLIS

- **What loops should be parallelized for multicore/manycore architectures?**
  - Loop 1 ($j_c$): Independent operations for multi-socket
  - Loops 2 or 6: race conditions!
  - Loop 3 ($i_c$): multicore with shared L3, private L2
  - Loops 4 ($j_r$):
    - Replicated $A_c$ if L2 cache is private. Single copy if shared. Single slice of $B_c$ if L1 is private
  - Loops 5 ($i_r$):
    - Fine-grained. Replicated slice of $B_c$ in each (private) L1 cache

"Anatomy of High-Performance Many-Threaded Matrix Multiplication"
T. M. Smith, R. van de Geijn, M. Smelyanskiy, J. R. Hammond, F. G. Van Zee
International Parallel and Distributed Processing Symposium - IPDPS, 2014
http://www.cs.utexas.edu/users/flame
Outline

- BLAS for ARM big.LITTLE AMPs

S. Catalán, E. S. Quintana-Ortí, J. R. Herrero, F. D. Igual, R. Rodríguez-Sánchez

“Architecture-Aware Configuration and Scheduling of Matrix Multiplication on Asymmetric Multicore Processors”
S. Catalán, F. D. Igual, R. Mayo, R. Rodríguez-Sánchez, E. S. Quintana-Ortí

“Multi-Treaded Dense Linear Algebra Libraries for Low-Power Asymmetric Multicore Processors”
S. Catalán, J. R. Herrero, F. D. Igual, R. Rodríguez-Sánchez, E. S. Quintana-Ortí
BLAS for ARM big.LITTLE AMPs

- Samsung Exynos 5422
  - Private L1 cache per core, private L2 cache per cluster
  - No L3 cache
BLAS for ARM big.LITTLE AMPs: GEMM

- Scheduling for multi-threaded asymmetric architecture?

Which loop(s) do we parallelize?
- Balanced workload distribution
- Exploit cache organization
BLAS for ARM big.LITTLE AMPs: GEMM

- Static symmetric scheduling between clusters
BLAS for ARM big.LITTLE AMPs: GEMM

- Static asymmetric scheduling between clusters
BLAS for ARM big.LITTLE AMPs: GEMM

- Cache-aware optimization with static asymmetric scheduling

Use different $m_c$, $k_c$ depending on the type of core
BLAS for ARM big.LITTLE AMPs: GEMM

- Cache-aware optimization with dynamic asymmetric scheduling

Dynamically distribute the iteration space for Loop 1 between the two clusters

```
Loop 1  for j_c = 0, ..., n - 1 in steps of n_c
Loop 2  for p_c = 0, ..., k - 1 in steps of k_c
        B(p_c : p_c + k_c - 1, j_c : j_c + n_c - 1) \rightarrow B_c
Loop 3  for i_c = 0, ..., m - 1 in steps of m_c
        A(i_c : i_c + m_c - 1, p_c : p_c + k_c - 1) \rightarrow A_c
Loop 4  for j_r = 0, ..., n_c - 1 in steps of n_r
Loop 5  for i_r = 0, ..., m_c - 1 in steps of m_r
Loop 6  for p_r = 0, ..., k_c - 1 in steps of 1
        C_c(i_r : i_r + m_r - 1, j_r : j_r + n_r - 1)
        += A_c(i_r : i_r + m_r - 1, p_r)
        \cdot B_c(p_r, j_r : j_r + n_r - 1)
        endfor
        endfor
        endfor
        endfor
        endfor
```
BLAS for ARM big.LITTLE AMPs: GEMM

- Performance...
BLAS for ARM big.LITTLE AMPs: GEMM

- ...and energy efficiency?
BLAS for ARM big.LITTLE AMPs
Concluding remarks

- All Level 3 BLAS available: GEMM, SYMM (HEMM), TRMM, TRSM, SYRK (HERK), SYR2K (HER2K)
- Easy to integrate support for AMPs into BLIS framework
- Significant increase in GFLOPS and GFLOPS/W with architecture-aware BLAS
Outline

- (Dense) Matrix factorizations on ARM big.LITTLE AMPs

S. Catalán, R. Rodríguez-Sánchez, E. S. Quintana-Ortí

L. Costero, F. D. Igual, K. Olcoz

“Revisiting Conventional Task Schedulers to Exploit Asymmetry in ARM big.LITTLE Architectures for DLA”
S. Catalán, L. Costero, F. D. Igual, K. Olcoz, E. S. Quintana-Ortí, R. Rodríguez-Sánchez
arXiv:1509.02058 [cs.DC], September 2015
Submitted to AsHES 2016
Matrix Factorizations in ARM big.LITTLE AMPs

- High performance matrix factorizations (LAPACK) via multi-threaded BLAS

```c
void cholesky (double *A[s][s], int b, int s) {
    for (int k = 0; k < s; k++) {
        po_cholesky (A[k][k], b, b);
        for (int j = k + 1; j < s; j++)
            tr_solve (A[k][k], A[k][j], b, b);
        for (int i = k + 1; i < s; i++) {
            for (int j = i + 1; j < s; j++)
                ge_multiply (A[k][i], A[k][j],
                             A[i][j], b, b);
            sy_update (A[k][i], A[i][i], b, b);
        }
    }
}
```

Straight-forward for ARM big.LITTLE AMPs using asymmetric-aware BLIS!
Matrix Factorizations in ARM big.LITTLE AMPs

- For some operations yields fair performance
Matrix Factorizations in ARM big.LITTLE AMPs

- ... but some others require a better solution
Matrix Factorizations in ARM big.LITTLE AMPs

- For complex DLA, exploiting task-parallelism delivers higher performance

```c
void cholesky (double *A[s][s], int b, int s)
{
    for (int k = 0; k < s; k++) {
        po_cholesky (A[k][k], b, b);
        for (int j = k + 1; j < s; j++)
            tr_solve (A[k][k], A[k][j], b, b);
        for (int i = k + 1; i < s; i++) {
            for (int j = i + 1; j < s; j++)
                ge_multiply (A[k][i], A[k][j],
                             A[i][j], b, b);
            sy_update (A[k][i], A[i][i], b, b);
        }
    }
}
```

Runtimes:
- Quark (UT@Knoxville)
- SuperMatrix (UT@Austin)
- OmpSs (BSC@Barcelona)
- StarPU (INRIA@Bordeaux)
Matrix Factorizations in ARM big.LITTLE AMPs

- Runtimes for AMPs?
  - Considerably more difficult because of tasks in the critical path
  - Botlev (OmpSs)
Matrix Factorizations in ARM big.LITTLE AMPs

- Exploit asymmetric-oblivious runtime + multi-threaded asymmetric-aware BLIS
  - 4 Symmetric virtual cores (VC)
Matrix Factorizations in ARM big.LITTLE AMPs

- Cholesky factorization via runtimes
Matrix Factorizations in ARM big.LITTLE AMPs

- Asymmetric-oblivious OmpSs + sequential BLIS
- Asymmetric-oblivious OmpSs + asymmetric-aware multi-threaded BLIS
- Botlev (asymmetric-aware runtime)
Matrix Factorizations in ARM big.LITTLE AMPs

Concluding remarks

- Possible to refactor existing asymmetric-oblivious runtimes
- Hide asymmetry inside BLIS
- Competitive performance, less programming burden
- Possible for other domains?
Outline

- Fault tolerance *(for energy efficiency)*

T. M. Smith, R. A. van de Geijn  
M. Smelyanskiy  
E. S. Quintana-Ortí

“Toward ABFT for BLIS GEMM”  
T. M. Smith, R. A. van de Geijn, M. Smelyanskiy, E. S. Quintana-Ortí  
Fault tolerance: GEMM

- Consider $C = A \cdot B$, and the augmented matrices

\[
A^* = \begin{pmatrix} A \\ v^T A \end{pmatrix}, \quad B^* = \begin{pmatrix} B \\ Bw \end{pmatrix}, \quad C^* = \begin{pmatrix} C \\ v^T C \\ v^T Cw \end{pmatrix},
\]

In absence of errors, then $C^* = A^* B^*$.

1. Compute (possibly left and) right checksum vector(s):

\[
\|d\|_\infty = \|Cw - A(Bw)\|_\infty
\]

2. Error if

\[
\|d\|_\infty > \tau \|A\|_\infty \|B\|_\infty
\]

with $\tau = \max(m, n, k) \cdot u$

Cost of detection is fixed

Cost of correction depends on error rate

3. Recompute if error detected
Fault tolerance: GEMM

- Tune granularity
Fault tolerance: GEMM

- Tune granularity

<table>
<thead>
<tr>
<th>Loop index</th>
<th>Required workspace</th>
<th>$O_d$ and $O_c$ depend on</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j_c$</td>
<td>$m \times n_c$</td>
<td>$(m, n_c, k)$</td>
</tr>
<tr>
<td>$p_c$</td>
<td>$m \times n_c$</td>
<td>$(m, n_c, k)$</td>
</tr>
<tr>
<td>$i_c$</td>
<td>$m_c \times n_c$</td>
<td>$(m_c, n_c, k_c)$</td>
</tr>
<tr>
<td>$j_r$</td>
<td>$m_c \times n_r$</td>
<td>$(m_c, n_r, k_c)$</td>
</tr>
<tr>
<td>$i_r$</td>
<td>$m_r \times n_r$</td>
<td>$(m_r, n_r, k_c)$</td>
</tr>
</tbody>
</table>

$$
O_d = \frac{4m_c n_c + 5m_c k_c + 5k_c n_c}{2m_c n_c k_c}
$$

$$
O_c = \frac{2m_r n_r k_c E_b}{2m_c n_c k_c} = \frac{m_r n_r E_b}{m_c n_c}
$$
Fault tolerance: GEMM

Loop 1 for \( j_c = 0, \ldots, n - 1 \) in steps of \( n_c \)
Loop 2 for \( p_c = 0, \ldots, k - 1 \) in steps of \( k_c \)
\[
B(p_c : p_c + k_c - 1, j_c : j_c + n_c - 1) \rightarrow B_c
\]
Loop 3 for \( i_c = 0, \ldots, m - 1 \) in steps of \( m_c \)
\[
A(i_c : i_c + m_c - 1, p_c : p_c + k_c - 1) \rightarrow A_c
\]
Loop 4 for \( j_r = 0, \ldots, n_r - 1 \) in steps of \( n_r \)
for \( i_r = 0, \ldots, m_r - 1 \) in steps of \( m_r \)
\[
C_c(i_r : i_r + m_r - 1, j_r : j_r + n_r - 1) = A_c(i_r : i_r + m_r - 1, 0 : k_c - 1) \\
\cdot B_c(0 : k_c - 1, j_r : j_r + n_r - 1)
\]
endfor
endfor
endfor
endfor
Fault tolerance: GEMM

- Right checksum:

\[ d = \hat{C}_c \cdot w - A_c \cdot B_c \cdot w \]
Dealing with Asymmetry for Performance and Energy Efficiency in ARM big.LITTLE architectures

Outline

- Fault tolerance for energy efficiency

S. Catalán, E. S. Quintana-Ortí, R. Rodríguez-Sánchez  
J. R. Herrero
T. M. Smith, R. A. van de Geijn

"On the Energy Costs of Fault Tolerance for Matrix Multiplication on Low-Power Multicore Architectures"
S. Catalán, J. R. Herrero, E. S. Quintana-Ortí, R. Rodríguez-Sánchez, T. M. Smith, R. A. van de Geijn
Submitted to AsHES 2016
Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

- Energy efficiency of BLIS GEMM

To a large extent, \( P = \alpha_{A15} V^2 f \)
Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

- Assume GFLOPS depend linearly on $f$
- Most energy-efficient configuration as the reference:
  - $(V_R, f_R) = (0.912 \text{ V}, 600 \text{ MHz})$
  - Reference performance $G_R = 10.45 \text{ GFLOPS}$
  - Reference power dissipation rate $P_R = 1.41 \text{ W}$
  - In an error-free execution, the reference energy efficiency is then
    $G_R / P_R \approx G_R / (\alpha_{A15} V_R^2 f_R)$

What is the error rate we can accommodate into FT GEMM via voltage-frequency scaling (VFS) while delivering the same/higher energy efficiency as the reference case?
Consider an arbitrary “configuration”:
- \((V_A, f_A)\) with performance \(G_A\), power dissipation rate \(P_A\), and energy efficiency
  \[
  \frac{G_A}{P_A} \approx \frac{G_A}{(\alpha_{A15} V_A^2 f_A)}
  \]
- Detection and correction overheads can be modeled as a decrease in the effective GFLOPS rate:

\[
\frac{G_A}{(1+O_d)(1+O_c)}
\]

\[
O_d = \frac{4m_c n_c + 5m_c k_c + 5k_c n_c}{2m_c n_c k_c}
\]

\[
O_c = \frac{2m_r n_r k_c E_b}{2m_c n_c k_c} = \frac{m_r n_r E_b}{m_c n_c}
\]
Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

- Iso-energy:
  - Original BLIS GEMM executed with the reference configuration and no errors
  - FT-BLIS GEMM executed with the arbitrary configuration and errors that incur detection+correction overheads

\[
\begin{align*}
\frac{G_R}{P_R} &= \frac{G_R}{\alpha_{A15} V_R^2 f_R} \\
\frac{G_A}{P_A} &= \frac{G_A}{\alpha_{A15} V_A^2 f_A} = \frac{G_A}{\alpha_{A15} V_A^2 f_A (1+\delta_d)(1+\delta_c)}
\end{align*}
\]
Dealing with Asymmetry for Performance and Energy Efficiency in ARM big.LITTLE architectures

Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

- Under iso-energy conditions, then

\[ O_{c}^{iso} = \left( \frac{G_R}{V^2 f_R} \right)^{-1} \left( \frac{G_A}{V^2 f_A (1 + O_d)} \right) - 1. \]

→ Detection overhead = 5%
Voltage scaled down to 0.85 V
If correction overhead < 10%: higher energy efficiency than the reference case
Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

Concluding remarks

- NTVC promises higher energy efficiency at the cost of exposing unreliable hardware
- Difficult to predict the error rate due to NTVC (missing error model!)
- Overcome by analyzing the trade-off between energy efficiency and resilience:

  What is the error rate we can accommodate into FT GEMM via voltage-frequency scaling (VFS) to deliver the same/higher energy efficiency as the reference case?
Dealing with *Asymmetry* for Performance and Energy Efficiency in ARM big.LITTLE architectures

- ...and sparse linear algebra?

  C. Chalios, D. Nikolopoulos
  J. I. Aliaga,
  S. Catalán,
  E. S. Quintana-Ortí

"Evaluating Asymmetric Multicore Systems-on-Chip and the Cost of Fault Tolerance using Iso-metrics"
C. Chalios, D. Nikolopoulos, S. Catalán, E. S. Quintana.
IET Computers & Digital Techniques, 2016. To appear

"Performance and Fault Tolerance of Preconditioned Iterative Solvers on Low-Power ARM Architectures".
ParCo2015. To appear
Dealing with *Asymmetry* for Performance and Energy Efficiency in ARM big.LITTLE architectures

*Enrique S. QUINTANA-ORTÍ*