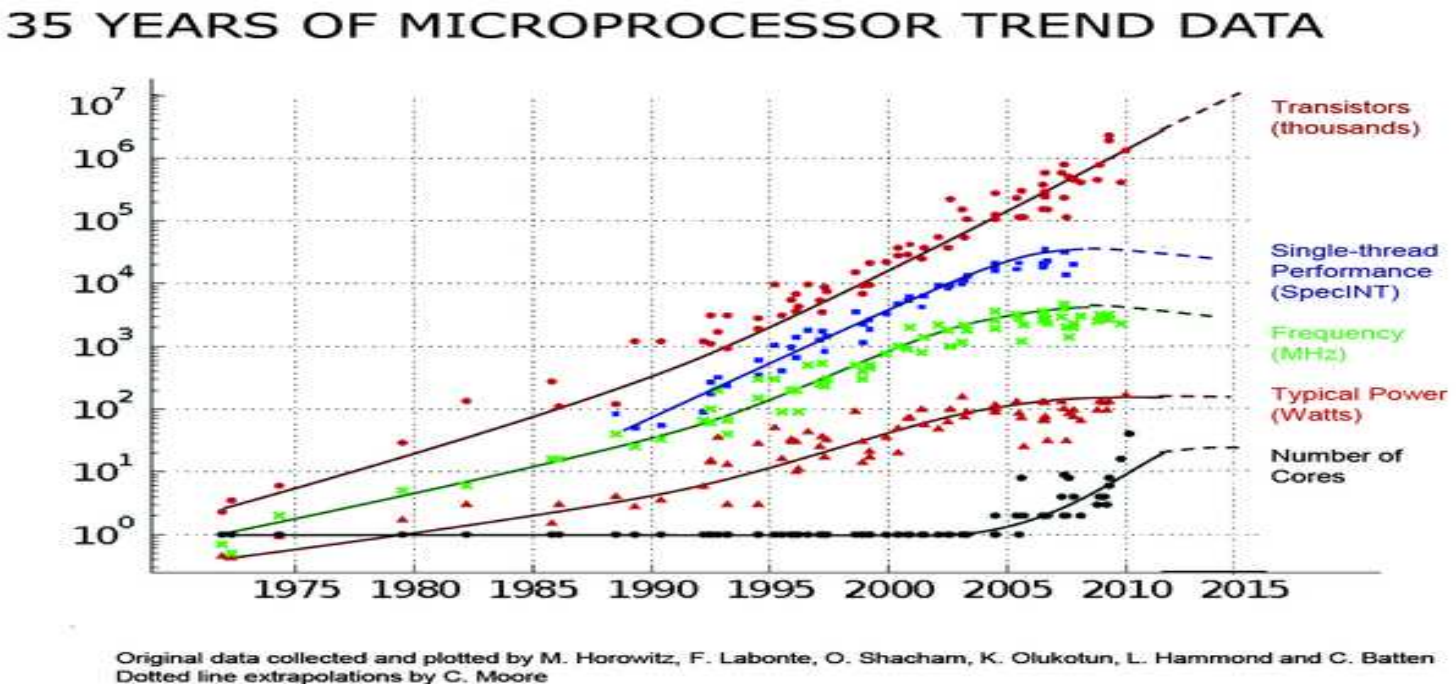
Dealing with *Asymmetry* for Performance and Energy Efficiency in ARM big.LITTLE architectures

Enrique S. QUINTANA-ORTÍ

UNIVERSITAT JAUME·I

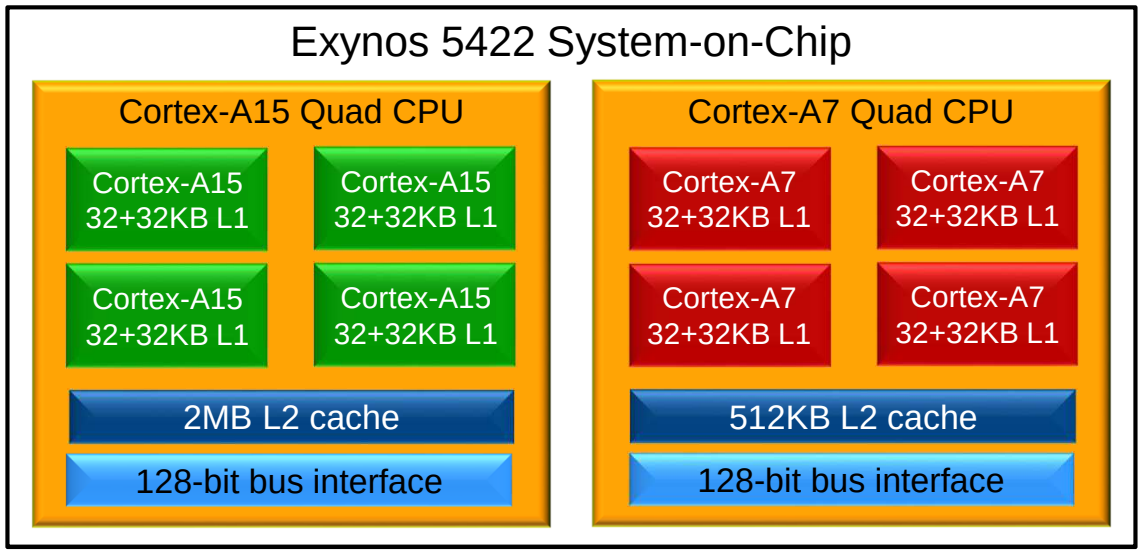
Motivation

- Moore's law is alive, but Dennard's scaling is over



Motivation

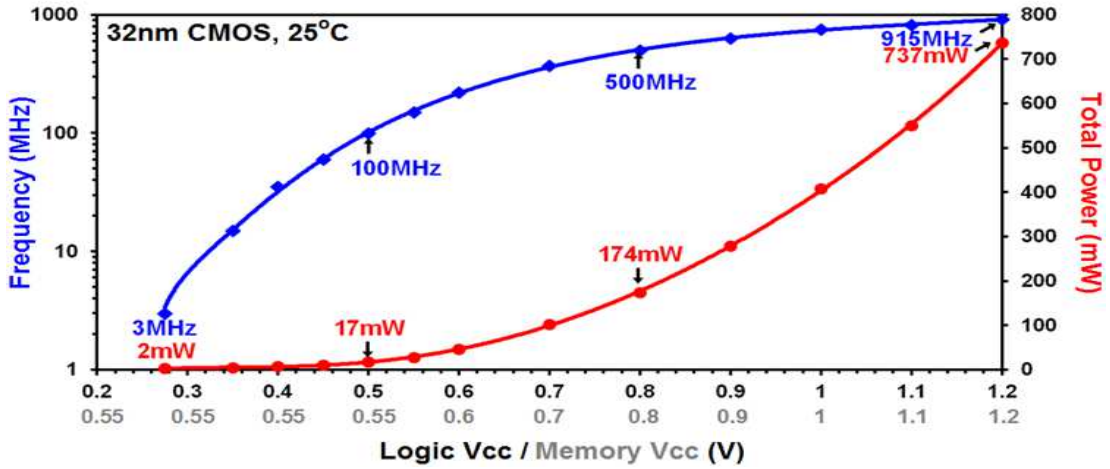
- Welcome “dark silicon” and asymmetric architectures to tackle power/energy/utilization walls!



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

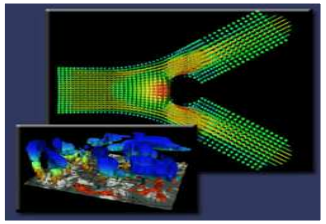
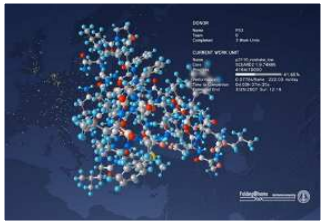
Motivation

- 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)

L A P A C K
L -A P -A C -K
L A P A -C -K
L -A P -A -C K
L A -P -A C K
L -A -P A C -K

BLAS

GEMM

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

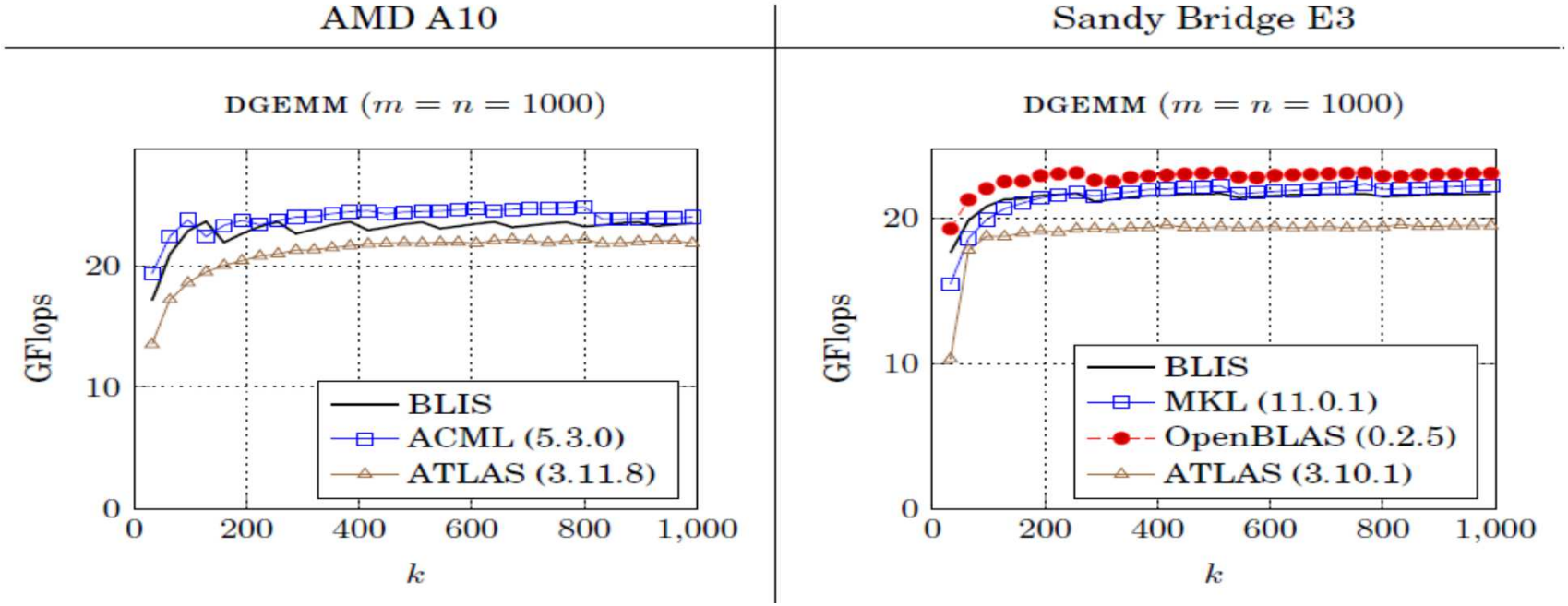
ACML

ESSL

CUBLAS

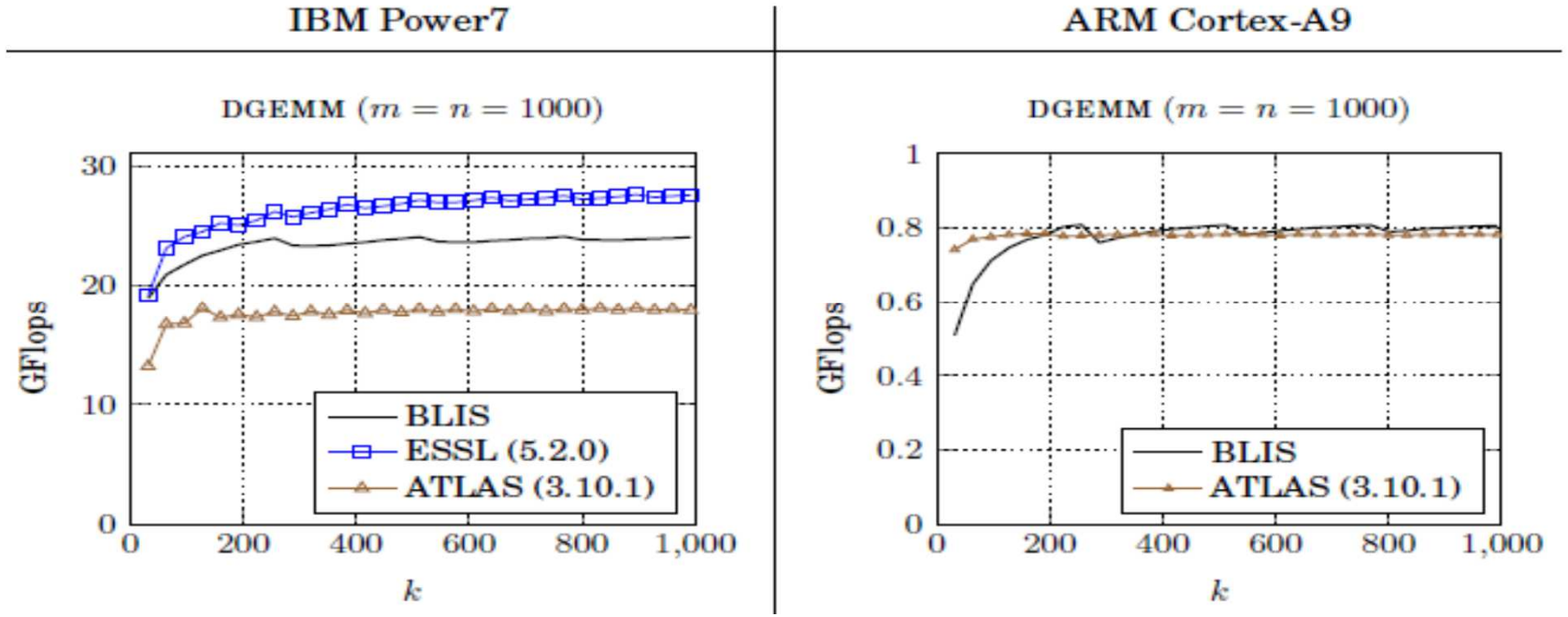
High Performance BLAS

- "Open" sw.: GotoBLAS, ATLAS, OpenBLAS, BLIS



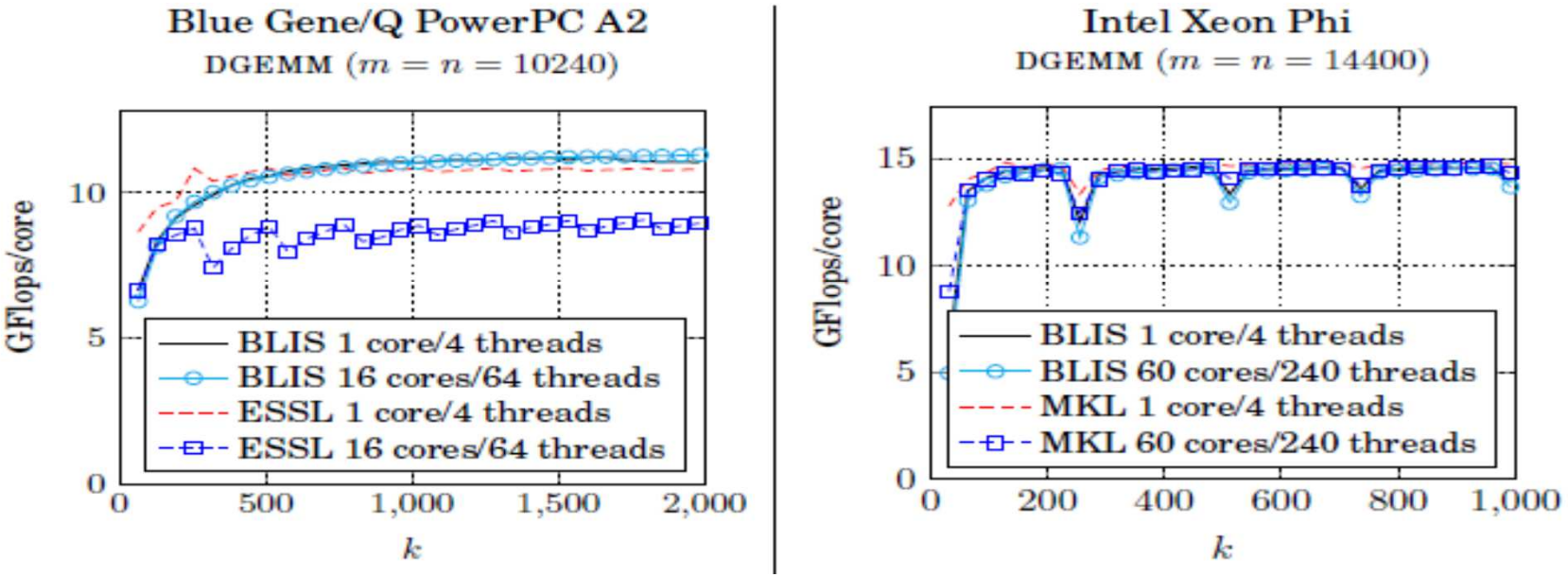
High Performance BLAS

- "Open" sw.: GotoBLAS, ATLAS, OpenBLAS, BLIS



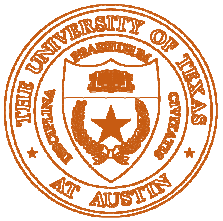
High Performance BLAS

- "Open" sw.: GotoBLAS, ATLAS, OpenBLAS, BLIS



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/



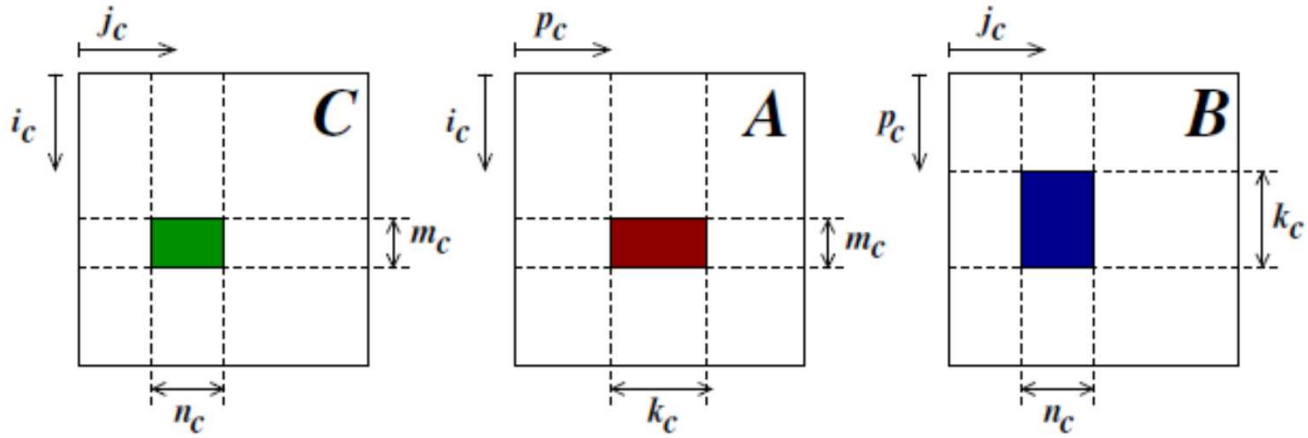
High Performance BLAS: GEMM in BLIS

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

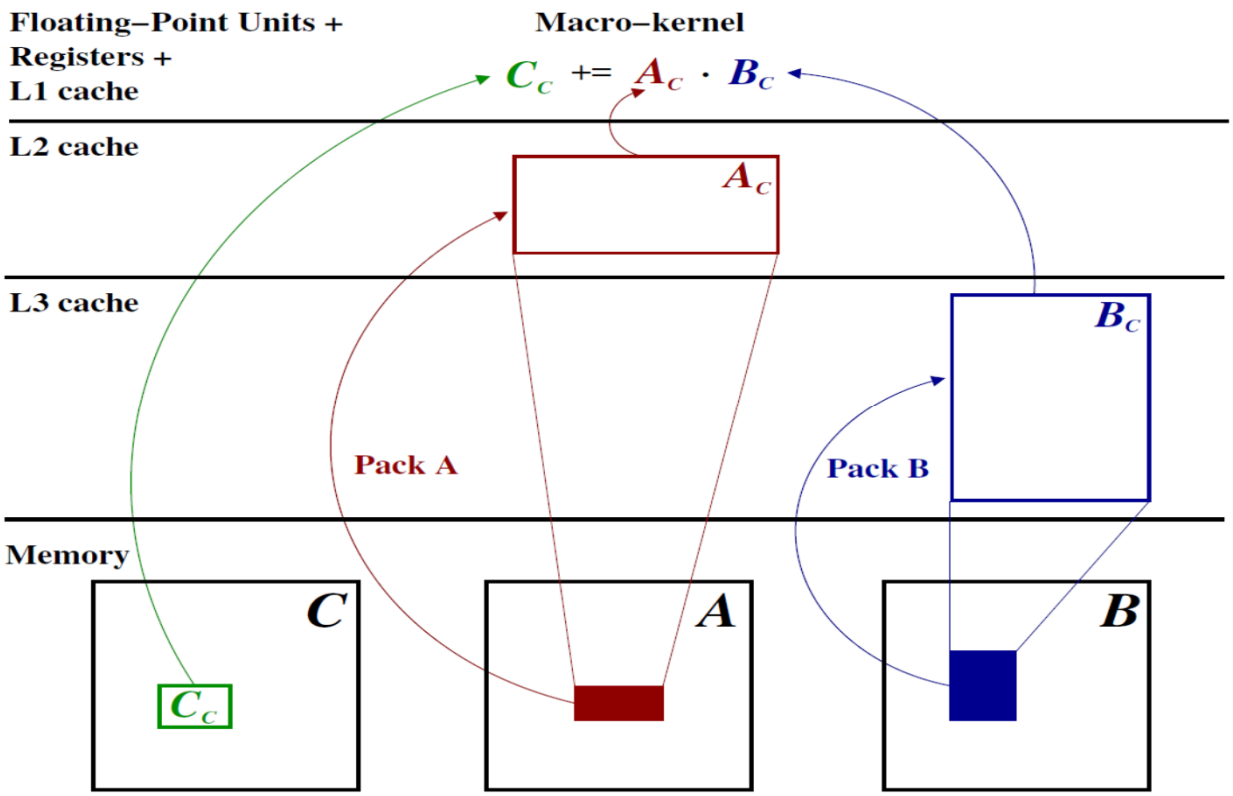
Loop 3: for $i_c = 0, \ldots, m-1$ in steps of m_c

$C(i_c : i_c + m_c - 1, j_c : j_c + n_c - 1) \equiv C_c \mathrel{+}= A_c \cdot B_c$ // Macro-kernel

endfor
endfor
endfor



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 // Macro-kernel

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

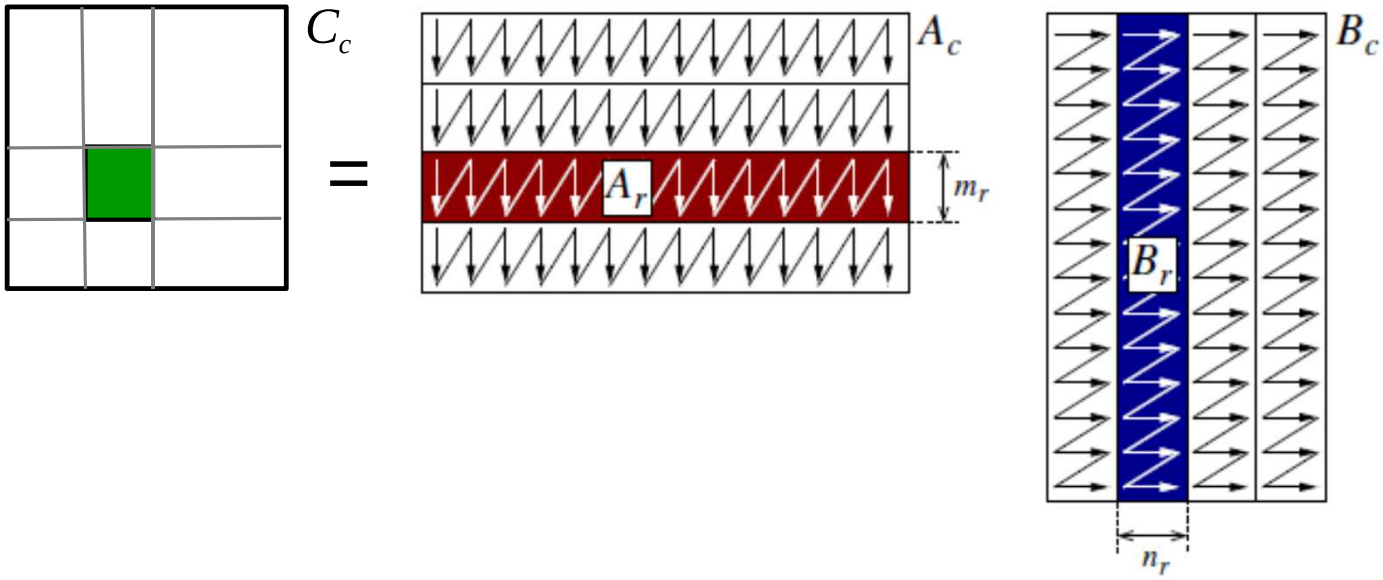
$C_c(i_r : i_r + m_r - 1, j_r : j_r + n_r - 1)$ // Micro-kernel

$\quad += A_c(i_r : i_r + m_r - 1, 0 : k_c - 1)$

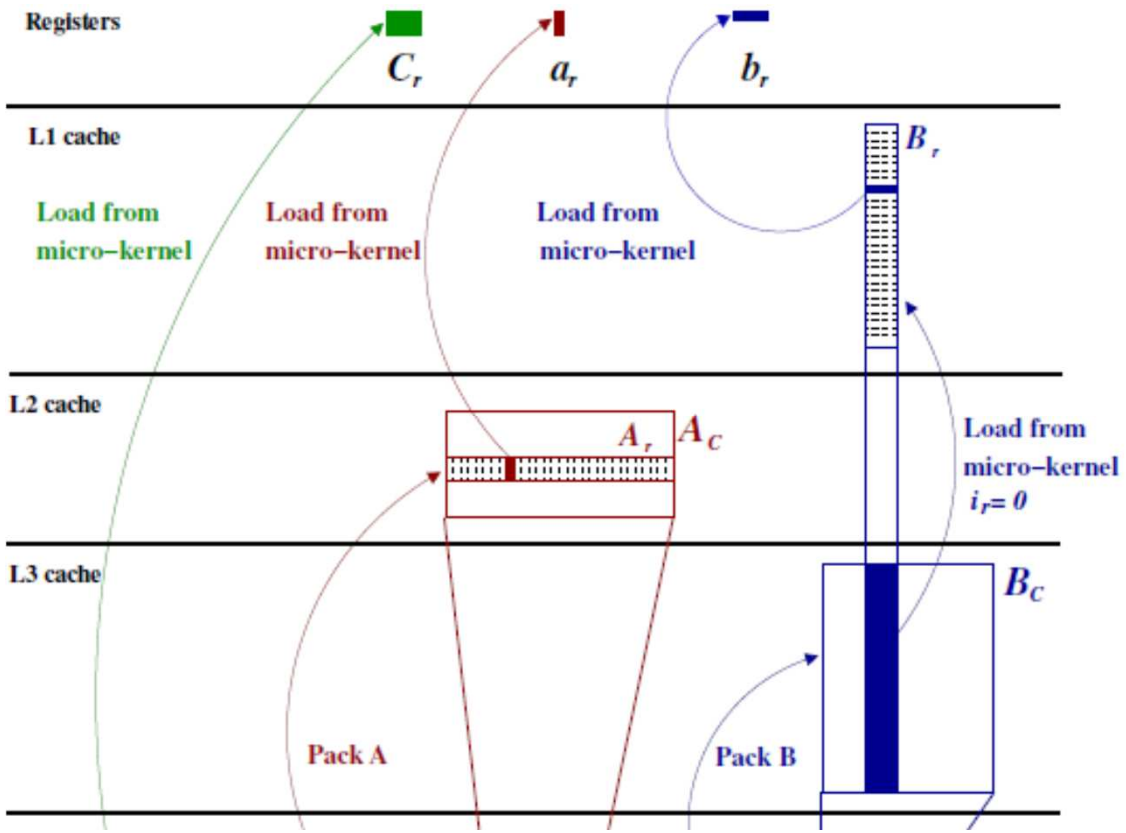
$\quad \cdot \quad B_c(0 : k_c - 1, j_r : j_r + n_r - 1)$

endfor

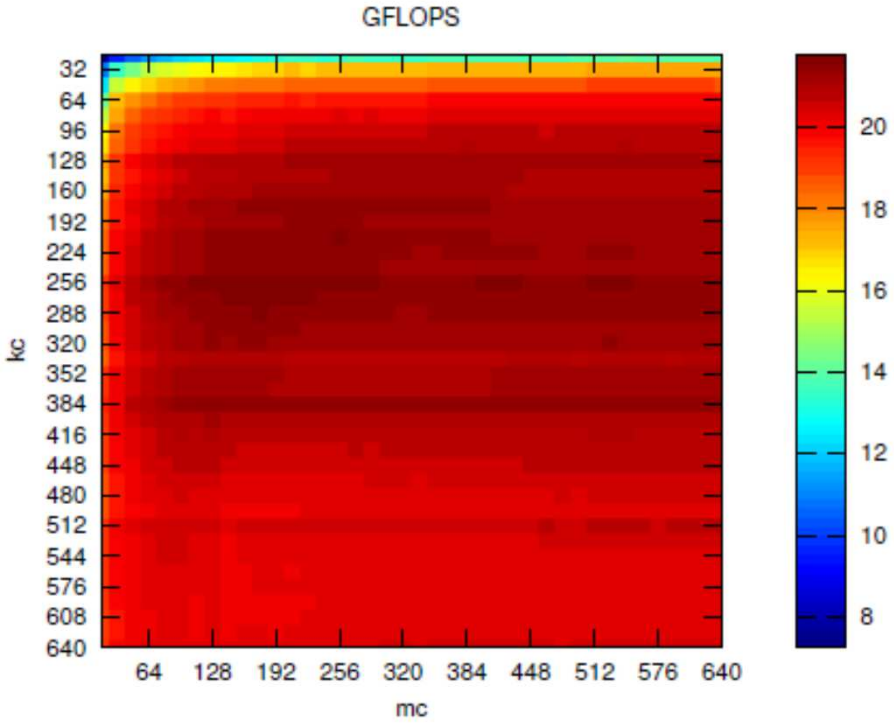
endfor

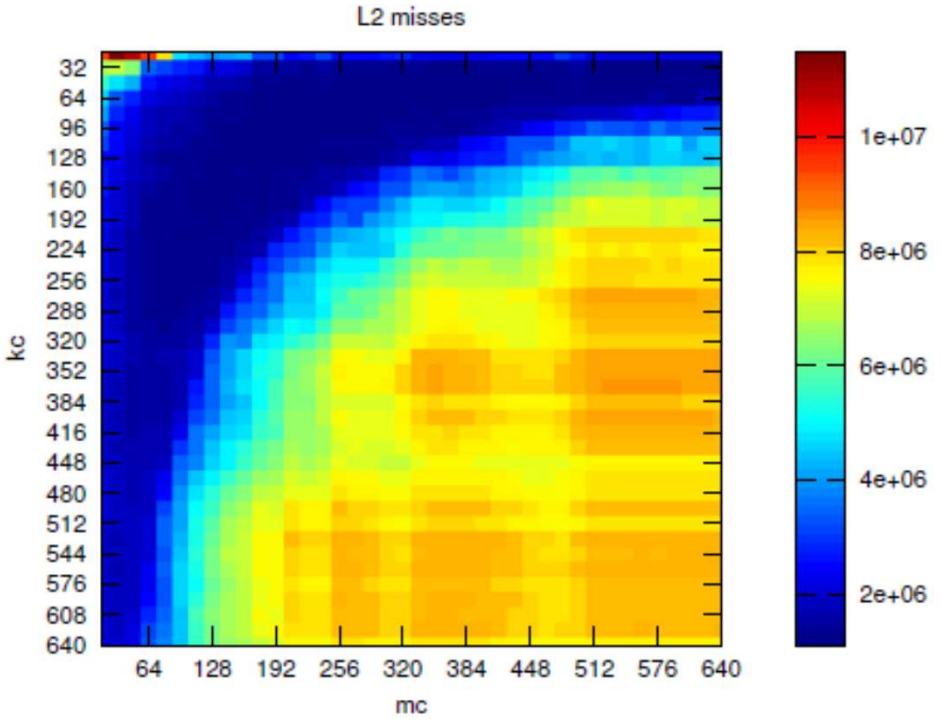


High Performance BLAS: GEMM in BLIS



High Performance BLAS: GEMM in BLIS





High Performance 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, R. Rodríguez-Sánchez

J. R. Herrero

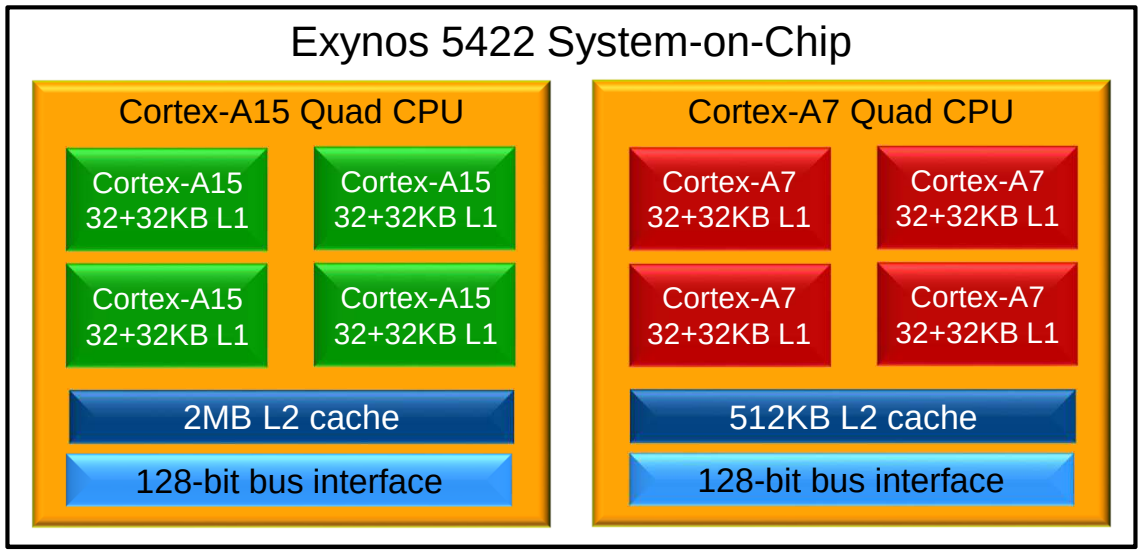
F. D. Igual

"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
arXiv:1506.08988 [cs.PF], June 2015. Submitted to Cluster Computing

"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
arXiv:1511.02171 [cs.MS], November 2015. Submitted to Journal of Computational Science

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?

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 $j_r = 0, \ldots, n_c - 1$ in steps of n_r
Loop 5 for $i_r = 0, \ldots, m_c - 1$ in steps of m_r
Loop 6 for $p_r = 0, \ldots, k_c - 1$ 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
endfor

Which loop(s) do we parallelize?

- Balanced workload distribution
- Exploit cache organization

BLAS for ARM big.LITTLE AMPs: GEMM

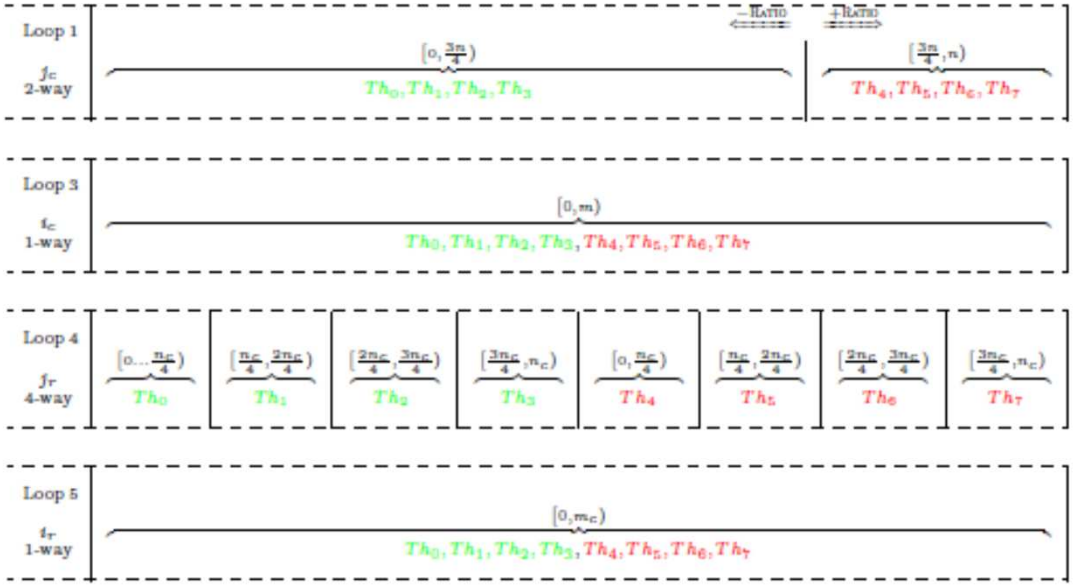
- Static symmetric scheduling between clusters

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 $j_r = 0, \ldots, n_c - 1$ in steps of n_r
Loop 5 for $i_r = 0, \ldots, m_c - 1$ in steps of m_r
Loop 6 for $p_r = 0, \ldots, k_c - 1$ 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

- Static asymmetric scheduling between clusters

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 $j_r = 0, \ldots, n_c - 1$ in steps of n_r
Loop 5 for $i_r = 0, \ldots, m_c - 1$ in steps of m_r
Loop 6 for $p_r = 0, \ldots, k_c - 1$ 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

- Cache-aware optimization with static asymmetric scheduling

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 $j_r = 0, \ldots, n_c - 1$ in steps of n_r
Loop 5 for $i_r = 0, \ldots, m_c - 1$ in steps of m_r
Loop 6 for $p_r = 0, \ldots, k_c - 1$ 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

Use different m_c, k_c depending on the type of core

BLAS for ARM big.LITTLE AMPs: GEMM

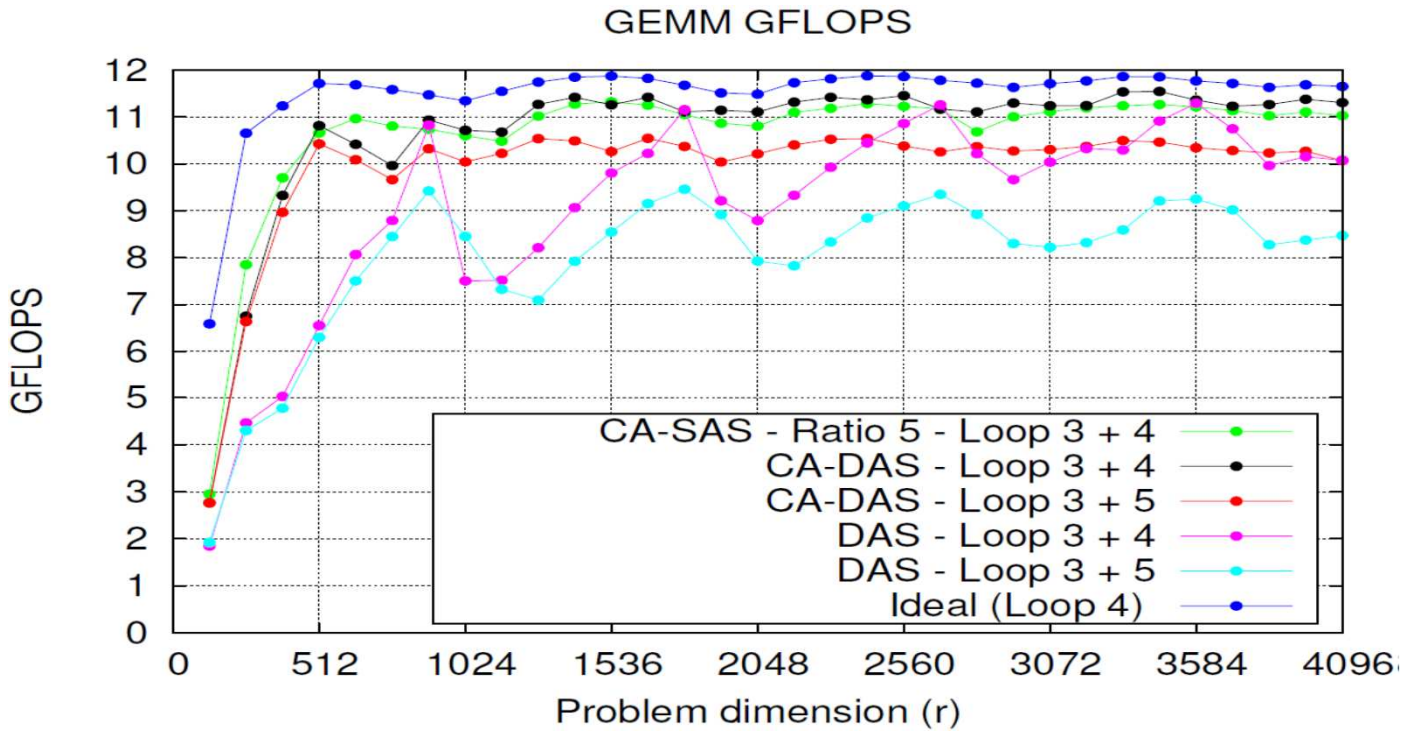
- Cache-aware optimization with dynamic asymmetric scheduling

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 $j_r = 0, \ldots, n_c - 1$ in steps of n_r
Loop 5 for $i_r = 0, \ldots, m_c - 1$ in steps of m_r
Loop 6 for $p_r = 0, \ldots, k_c - 1$ 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
endfor

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

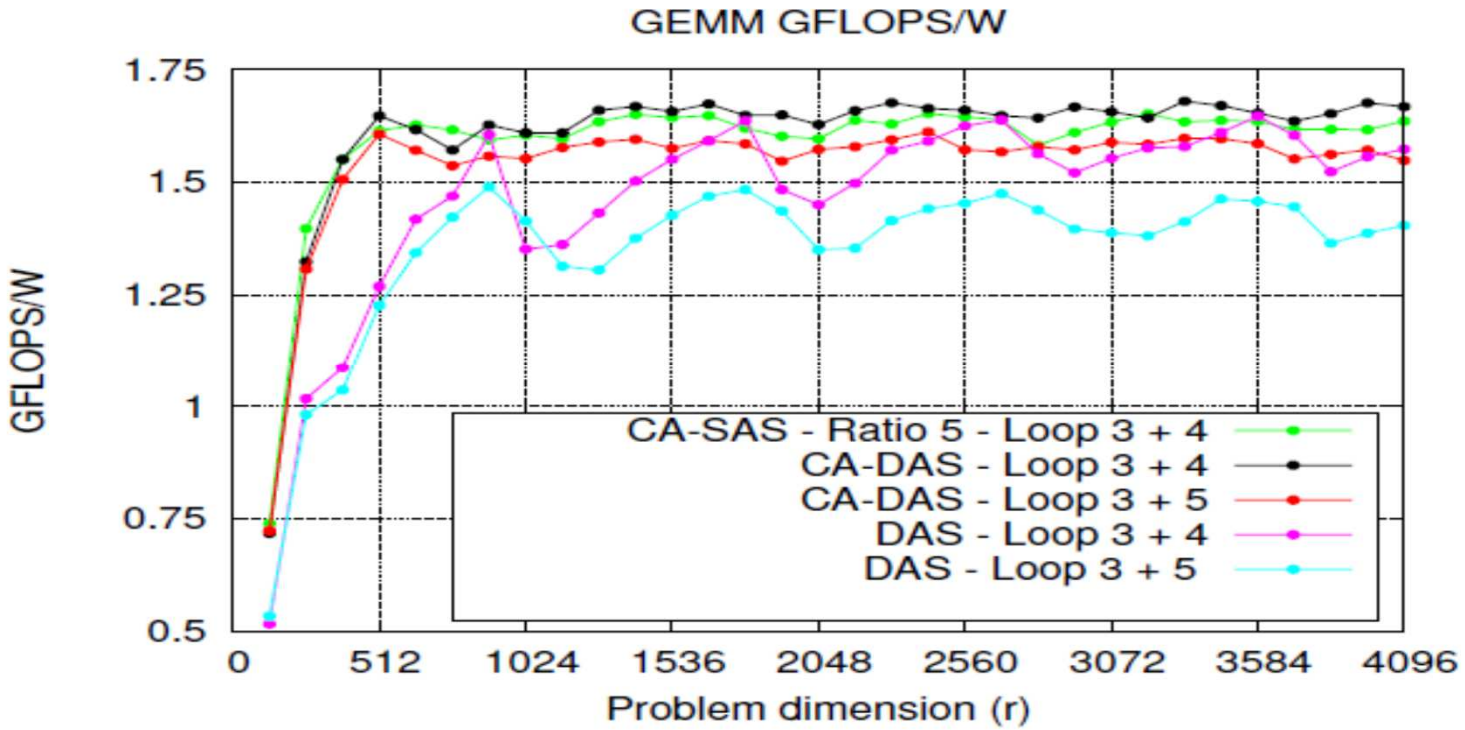
BLAS for ARM big.LITTLE AMPs: GEMM

- 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

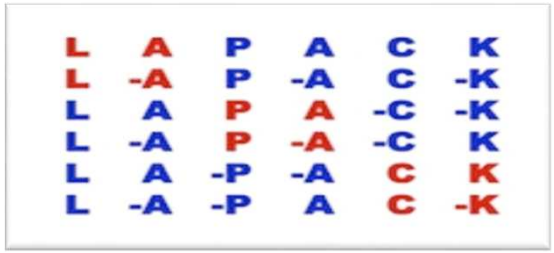
L. Costero, F. D. Igual
K. Olcoz

"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
axXiv: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

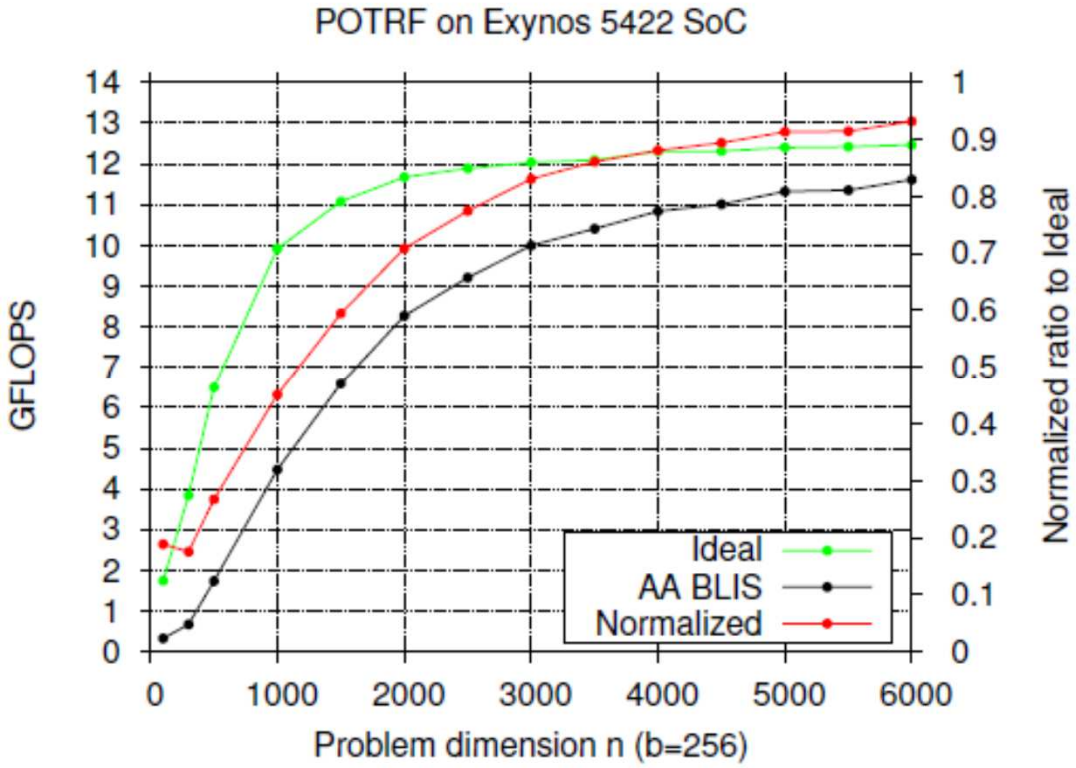
```
void cholesky (double *A[s][s], int b, int s)
{
   for (int k = 0; k < s; k++) {

      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!

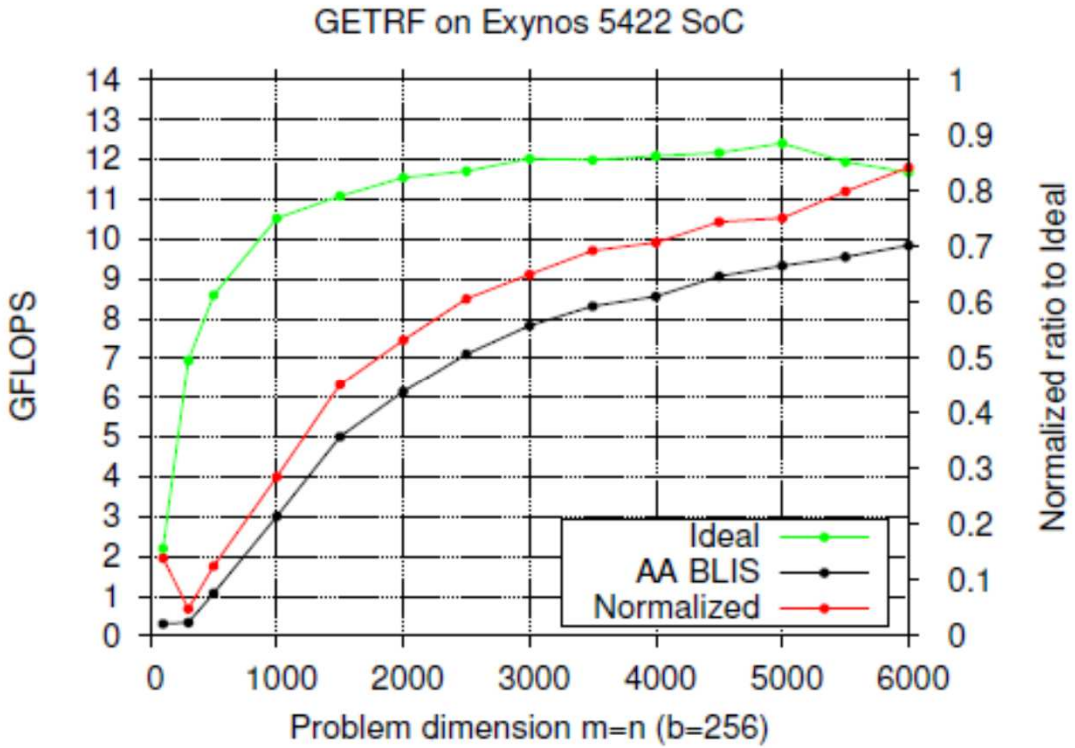
Matrix Factorizations in ARM big.LITTLE AMPs

- For some operations yields fair performance



Matrix Factorizations in ARM big.LITTLE AMPs

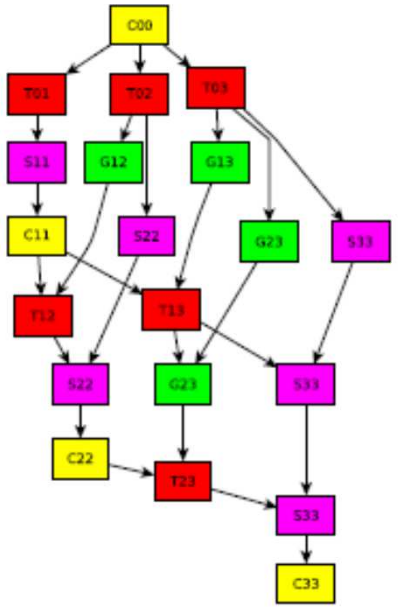
- … but some others require a better solution



Matrix Factorizations in ARM big.LITTLE AMPs

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

```
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);
      }

   }
}
```



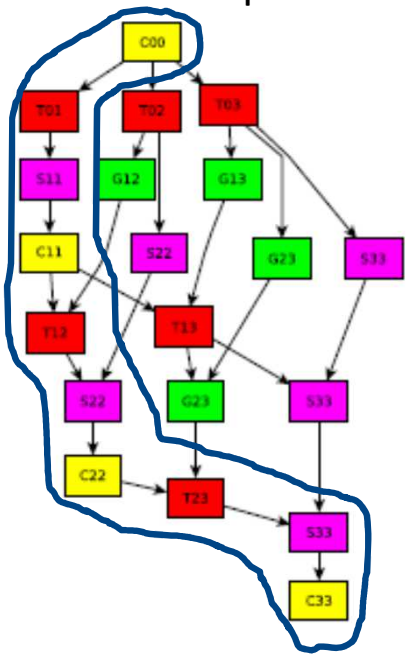
Runtimes:

- Quark (UT@Knoxville)
- SuperMatrix (UT@Austin)

- OmpSs (BSC@Barcelona)
- StarPU (INRIA@Bordeaux)

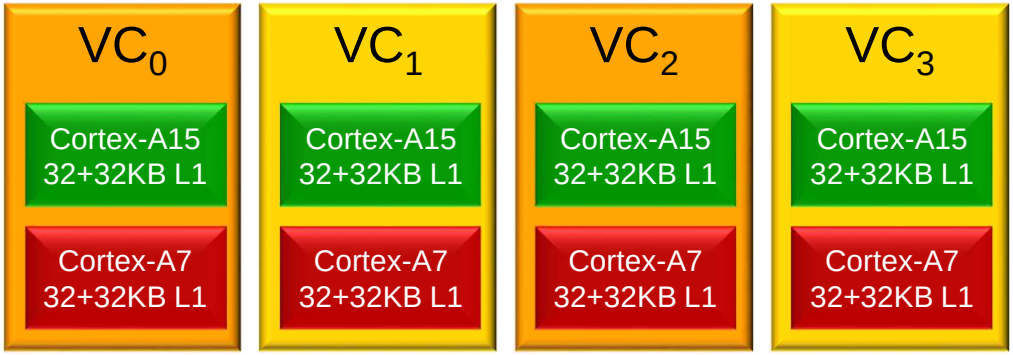
Matrix Factorizations in ARM big.LITTLE AMPs

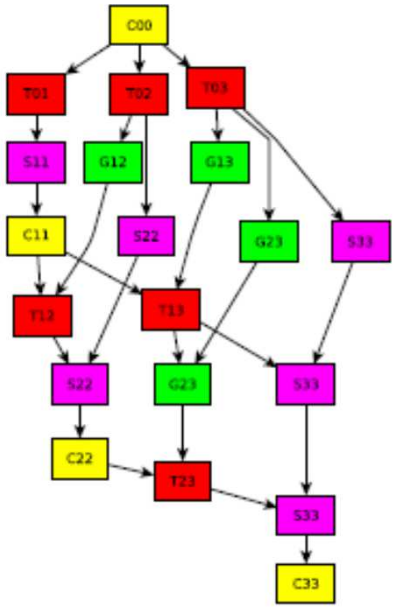
- Runtimes for AMPs?
  - Considerably more difficult because of tasks in the critical path
  - Botlev (OmpSs)



Matrix Factorizations in ARM big.LITTLE AMPs

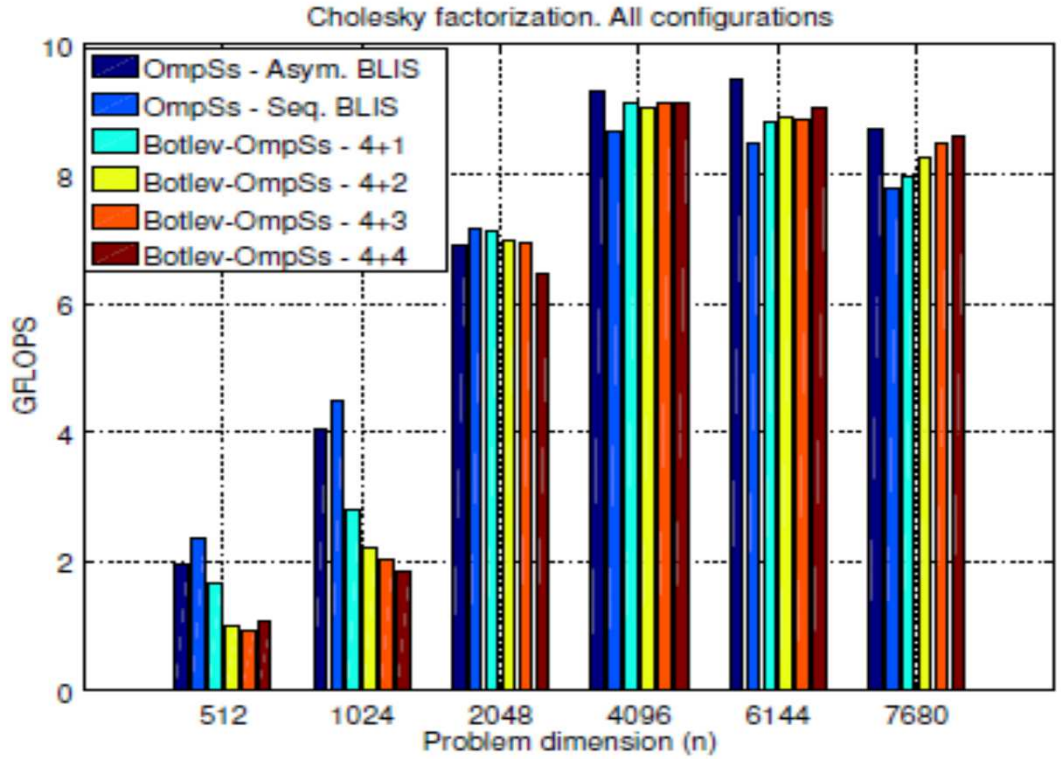
- Exploit asymmetric-oblivious runtime + multi-threaded asymmetric-aware BLIS
  - 4 Symmetric virtual cores (VC)





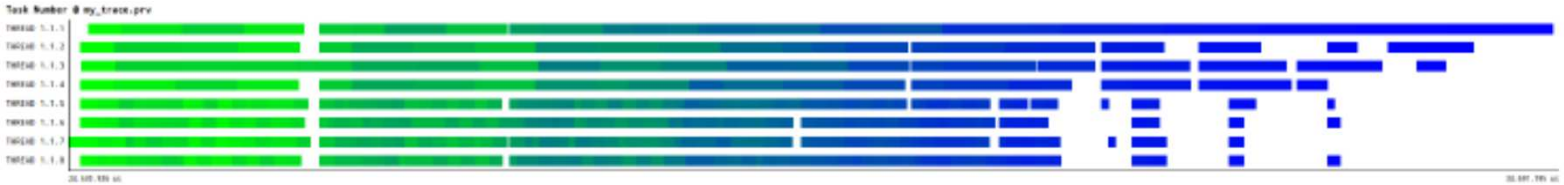
Matrix Factorizations in ARM big.LITTLE AMPs

- Cholesky factorization via runtimes



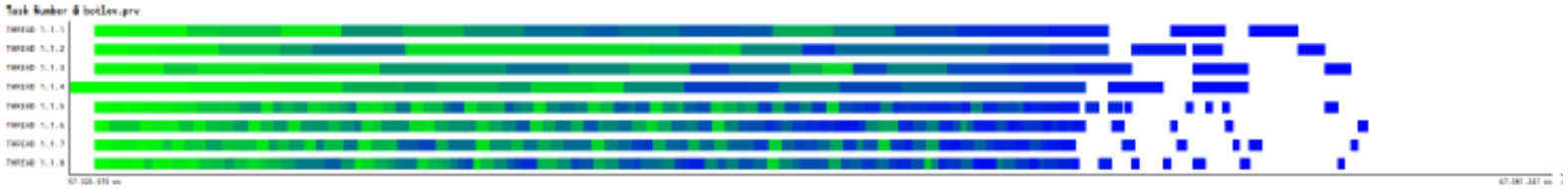
Matrix Factorizations in ARM big.LITTLE AMPs

- 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
FLAME Working note #76. Technical Report TR-15-05. Dept. of Computer Science. The University of Texas at Austin

Fault tolerance: GEMM

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

$$A^* = \left(\frac{A}{v^T A}\right), B^* = \left(\begin{array}{c|c} B & Bw \end{array}\right), C^* = \left(\begin{array}{c|c} C & Cw \\ \hline v^T C & v^T Cw \end{array}\right),$$

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

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** $j_r = 0, \ldots, n_c - 1$ **in steps of** n_r
Loop 5 **for** $i_r = 0, \ldots, m_c - 1$ **in steps of** m_r
$C_c(i_r : i_r + m_r - 1, j_r : j_r + n_r - 1)$
$+= A_c(i_r : i_r + m_r - 1, 0 : k_c - 1)$
$\cdot \; B_c(0 : k_c - 1, j_r : j_r + n_r - 1)$
endfor
endfor
endfor
endfor
endfor

Fault tolerance: GEMM

- Tune granularity

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** $j_r = 0, \ldots, n_c - 1$ **in steps of** n_r
Loop 5 **for** $i_r = 0, \ldots, m_c - 1$ **in steps of** m_r
$C_c(i_r : i_r + m_r - 1, j_r : j_r + n_r - 1)$
$+= A_c(i_r : i_r + m_r - 1, 0 : k_c - 1)$
$\cdot B_c(0 : k_c - 1, j_r : j_r + n_r - 1)$
endfor
endfor
endfor
endfor
endfor

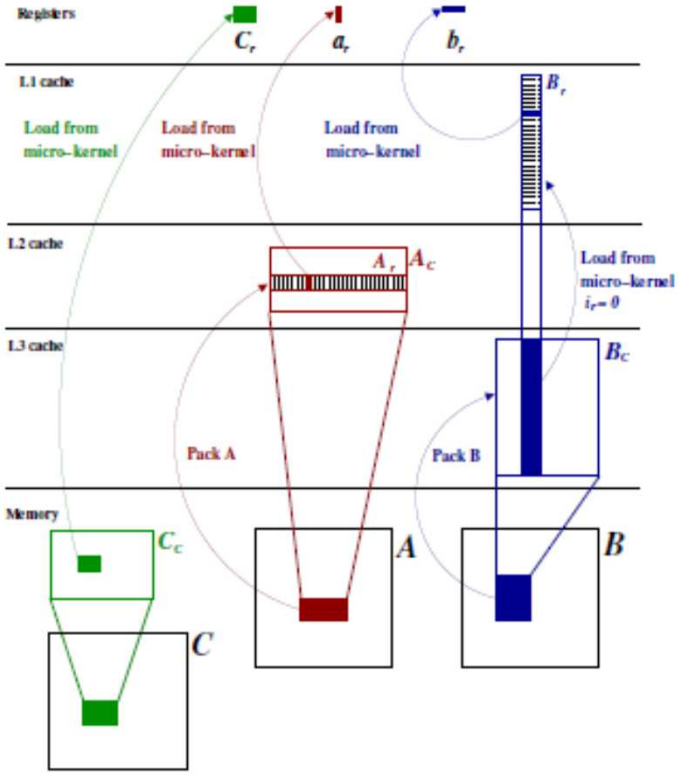
Loop index	Required workspace	$\mathcal{O}_d$ and $\mathcal{O}_c$ depend on
j_c	$m \times n_c$	(m, n_c, k)
p_c	$m \times n_c$	(m, n_c, k_c)
i_c	$m_c \times n_c$	(m_c, n_c, k_c)
j_r	$m_c \times n_r$	(m_c, n_r, k_c)
i_r	$m_r \times n_r$	(m_r, n_r, k_c)

$$\mathcal{O}_d = \frac{4m_c n_c + 5m_c k_c + 5k_c n_c}{2m_c n_c k_c}$$

$$\mathcal{O}_c = \frac{2m_r n_r k_c E_b}{2m_c n_c k_c} = \frac{m_r n_r E_b}{m_c n_c}.$$

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** $j_r = 0, \ldots, n_c - 1$ **in steps of** n_r
Loop 5 **for** $i_r = 0, \ldots, m_c - 1$ **in steps of** m_r
$C_c(i_r : i_r + m_r - 1, j_r : j_r + n_r - 1)$
$+= A_c(i_r : i_r + m_r - 1, 0 : k_c - 1)$
$\cdot B_c(0 : k_c - 1, j_r : j_r + n_r - 1)$
endfor
endfor
endfor
endfor
endfor



Fault tolerance: GEMM

- Right checksum:

$$d = \hat{C}_c \cdot w - A_c \cdot B_c \cdot w$$

for $p_c = 0, \ldots, k-1$ **in steps of** k_c, $\mathcal{P}_c = p_c : p_c + k_c - 1$
  for $j_c = 0, \ldots, n-1$ **in steps of** n_c, $\mathcal{J}_c = j_c : j_c + n_c - 1$
    $B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow B_c$
    $d_b := -B_c \cdot w$
    for $i_c = 0, \ldots, m-1$ **in steps of** m_c, $\mathcal{I}_c = i_c : i_c + m_c - 1$
      $A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow A_c$
      $d := A_c \cdot d_b \ (= A_c \cdot B_c \cdot d_b)$

---

      for $j_r = 0, \ldots, n_c - 1$ **in steps of** n_r, $\mathcal{J}_r = j_r : j_r + n_r - 1$
        for $i_r = 0, \ldots, m_c - 1$ **in steps of** m_r, $\mathcal{I}_r = i_r : i_r + m_r - 1$
          $T = A_c(\mathcal{I}_r, 0 : k_c - 1) \cdot B_c(0 : k_c - 1, \mathcal{J}_r)$
          $d(\mathcal{I}_r) \mathrel{+}= T \cdot w(\mathcal{J}_r)$

          $C_c(\mathcal{I}_r, \mathcal{J}_r) \mathrel{+}= T$
        endfor
      endfor

---

    endfor
  endfor

endfor

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

	Freq., f (MHz)	Voltage, V (V)	Performance, G (GFLOPS)	Power, P (W)	EE (GFLOPS/W)	Normalized w.r.t. $f_{\min}$ Freq., f_N	GFLOPS, G_N	Power, P_N	$(V^2 f)_N$
A15	200	0.912	3.48	0.49	7.05	1	1.00	1.00	1.00
	400	0.912	7.11	0.96	7.35	2	2.03	2.00	1.95
	600	0.912	10.45	1.41	7.37	3	2.99	3.00	2.86
	800	0.925	13.64	1.88	7.22	4	3.91	4.11	3.82
	1,000	0.973	17.10	2.63	6.49	5	4.90	5.68	5.32
	1,200	1.023	20.23	3.50	5.77	6	5.80	7.54	7.09
	1,400	1.062	23.00	4.41	5.21	7	6.59	9.48	8.92
	1,600	1.115	25.78	5.77	4.46	8	7.39	11.94	11.67
	1,800	1.191	28.29	7.62	3.70	9	8.11	15.33	15.43

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 V, 600 MHz)
  - Reference performance G_R = 10.45 GFLOPS
  - Reference power dissipation rate P_R = 1.41 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?

Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

- Consider an arbitrary "configuration":
  - (V_A, f_A) with performance G_A, power dissipation rate P_A, and energy efficiency $G_A / P_A \approx G_A / (\alpha_{A15} V_A^2 f_A)$
- Detection and correction overheads can be modeled as a decrease in the effective GFLOPS rate:

$$\frac{G_A}{(1+\mathcal{O}_d)(1+\mathcal{O}_c)}$$

$$\mathcal{O}_d = \frac{4m_c n_c + 5m_c k_c + 5k_c n_c}{2m_c n_c k_c}$$

$$\mathcal{O}_c = \frac{2m_r n_r k_c E_b}{2m_c n_c k_c} = \frac{m_r n_r E_b}{m_c n_c}.$$

Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

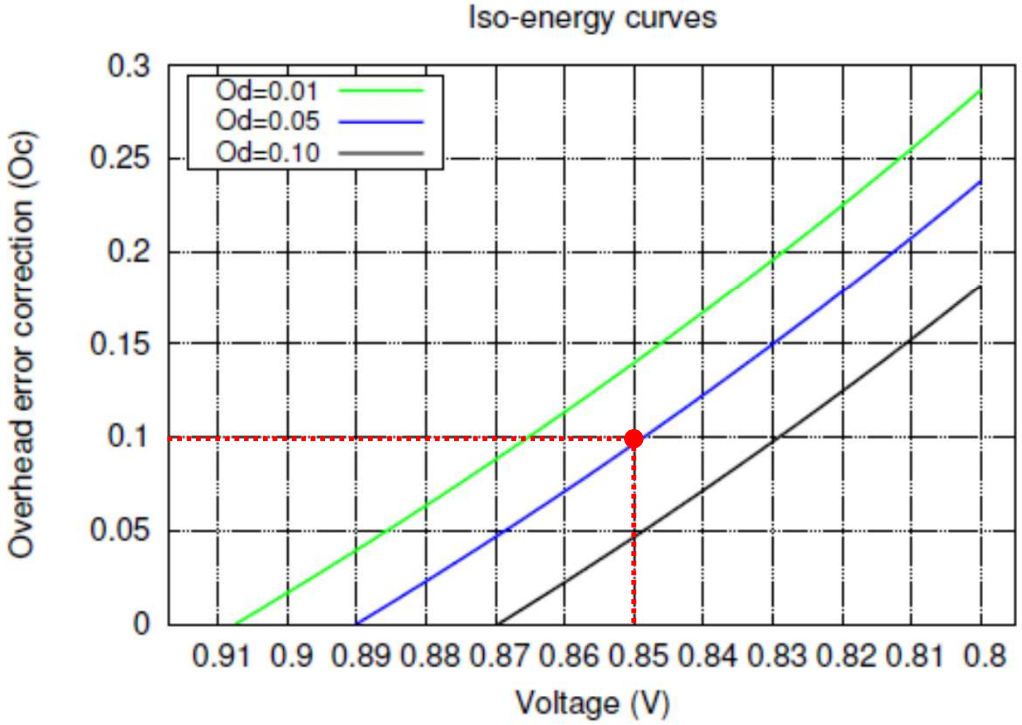
- Iso-energy:
  - Original BLIS GEMM executed with the reference configuration and no errors
  - FT-BLIS GEMM executed with the arbitrary configuration and errors that incur detection+correction overheads

$$\frac{G_R}{P_R} = \frac{G_R}{\alpha_{A15} V_R^2 f_R} =$$

$$\frac{G_A}{P_A} = \frac{G_A^{Eff}}{\alpha_{A15} V_A^2 f_A} = \frac{G_A}{\alpha_{A15} V_A^2 f_A (1+\mathcal{O}_d)(1+\mathcal{O}_c)}$$

Fault tolerance for energy efficiency in ARM big.LITTLE AMPs

- Under iso-energy conditions, then $\mathcal{O}_c^{iso} = \left(\frac{G_R}{V_R^2 f_R}\right)^{-1} \left(\frac{G_A}{V_A^2 f_A (1+\mathcal{O}_d)}\right) - 1.$



● → 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".
J. Aliaga, S. Catalán, C. Chalios, D. Nikolopoulos, E. S. Quintana.
ParCo2015. To appear

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

Enrique S. QUINTANA-ORTÍ

UNIVERSITAT JAUME·I