A portable DMA support of Level-3 BLIS for embedded manycore hardware

BLIS Retreat 2017 – September 18-19 UT Austin

Minh Quan Ho, Benoît Dupont De Dinechin - Kalray Inc.
Bernard Tourancheau - University of Grenoble Alps
Christian Obrecht - University Claude Bernard Lyon 1, CETHIL, INSA-Lyon
Francisco Igual Peña - Complutense University of Madrid (UCM)
Thank to:

Robert van de Geijn, Field Van Zee and Devin Matthews
for help and discussion
# AGENDA

1. Introduction

2. Challenge: BLIS for manycore

3. Solution: BLIS-RDMA on the Kalray MPPA2®-256

4. Conclusion
Introduction

MPPA®

The real-time extreme computing processor
Launched in 2008
Spin-off of the “CEA” (French Department of Energy)

Pioneer in manycore processors
• Focus on low latency, low-power, extreme processing
• Core market: Data Centers & Critical Embedded Applications

Comprehensive product offer
• MPPA® processors, acceleration cards, associated software
• Strategic partnership with TSMC

Adopted by FORTUNE 500 companies

1st COMMERCIAL PROCESSOR TO BREAK THE “100 COMPUTING CORE WALL” - 2013

TOP 20 “MOST PROMISING HPC SOLUTION PROVIDERS” – CIO REVIEW - 2016

21 NUMBER OF GRANTED PATENTS HELD BY KALRAY FOR ITS MANYCORE ARCHITECTURE

Three locations
Grenoble, France, Los Altos, CA, USA, Tokyo, Japan

Experienced technical team
World-leading multicore/manycore talent, with experience in both hardware & software
(M)assively
(P)arallel
(P)rocessor
(A)rray

MPPA®

The real-time extreme computing processor
MPPA® 2nd GENERATION: BOSTAN

**MANYCORE PROCESSOR**
- 16 compute clusters
- 4 quad-core host CPUs, DDR, and PCIe access
- 2 networks-on-chip
- 400-500 MHz

**COMPUTE CLUSTER**
- 16 user cores + 1 system core
- NoC Tx and Rx interfaces
- Debug & Support Unit (DSU)
- 2 MB shared memory

**VLIW CORE**
- 5-issue VLIW architecture
- MMU + 8KBx2 cache (I&D)
- 32-bit/64-bit IEEE 754 FPU
- Predictability & energy efficiency
MPPA® 3rd GENERATION: COOLIDGE
(available in 2018)

64-bit MANYCORE PROCESSOR
Can be tiled in same package or on-board
600 – 1200 MHz

COMPUTE CLUSTER

3RD GENERATION VLIW CORE
CHALLENGE:
BLIS FOR MANYCORE
BLIS FOR MANYCORE

INTRODUCTION

BLIS is officially cache-based

It is well-suited for multi-core applications
BLIS FOR MANYCORE
2 MAIN CHALLENGES

CHALLENGE 1

- Manycore often operates private caches and high-speed scratchpad memories
- Manycore suffers cache penalties during packing on large matrices (read and write)

CHALLENGE 2

- Can use DMA to copy blocks before packing => isolated from cache
- BLIS+DMA will be a good match for embedded manycore hardware

Size and Latency

Near Prefetch:
- HW prefetcher
- SW assembler
- Good improvement

Far Prefetch:
- HW prefetcher (!)
- SW RDMA
- Huge improvement

Core registers
L1 cache 1-2 cycles
L2 cache $O(1)$ cycles
L3 cache or Scratchpad memory or Tightly Coupled Memory - TCM $O(10)$ cycles
Off-chip memory, DDR, NVME etc. $O(10^3)$ - $O(10^9)$ cycles
### BLIS-RDMA FOR MANYCORE

**TWO OPTIONS FOR PORTING BLIS ON MANYCORE**

<table>
<thead>
<tr>
<th>1. Using cache (as for multi-core)</th>
<th>2. Using DMA on manycore (bypassing cache)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Out-of-the-box solution</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Advantage:</strong> Easy to set up <em>(out-of-the-box, gcc, C99/asm)</em></td>
<td><strong>Advantage:</strong> Higher performance</td>
</tr>
<tr>
<td><strong>Disadvantage:</strong> Poor performance</td>
<td><strong>Disadvantage:</strong> Time-consuming, complex</td>
</tr>
</tbody>
</table>
1. Asynchronous communication and overlapping in mind
BLIS-RDMA FOR MANYCORE CHALLENGES

1. Asynchronous communication and overlapping in mind
2. Scratchpad as an active « cache » not waiting for miss
3. Explicit data movement from global memory (DDR) to scratchpad
BLIS-RDMA FOR MANYCORE CHALLENGES

1. Asynchronous communication and overlapping in mind

2. Scratchpad as an active «cache» not waiting for miss

3. Explicit data movement from global memory (DDR) to scratchpad

4. Managed by software (API) to trigger DMA transfer, how to design it?

def struct point2d_s {
    int32_t xpos; // x-position from the buffer pointer
    int32_t ypos; // y-position from the buffer pointer
    int32_t xdim; // x-dim of the allocated buffer
    int32_t ydim; // y-dim of the allocated buffer
}

/* To describe a 2D copy */
void* global;
void* local;
int32_t width;
int32_t height;
size_t size;
point2d_t global_point;
point2d_t local_point;
BLIS-RDMA FOR MANYCORE CHALLENGES

1. Asynchronous communication and overlapping in mind

2. Scratchpad as an active « cache » not waiting for miss

3. Explicit data movement from global memory (DDR) to scratchpad

4. Managed by software (API) to trigger DMA transfer, how to design it?

5. Not the same address space, must copy all we need (e.g SYMM)
PORTABLE BLIS-RDMA EXTENSION

To avoid repeating the same process on new platforms
typedef vendor_dma_event_t dma_event_t;

void bli_dma_backend_init(void);

void bli_dma_backend_finalize(void);

// 2D (asynchronous) copy between on-chip (SNEM) and off-chip (DDR) memory
void bli_dma_backend_trigger_get2D(
    void* global,  // begin address of global buffer
    void* local,   // begin address of local buffer
    size_t size,   // size of an element in byte
    int width,     // block width in element
    int height,    // block height in element
    point2d_t *global_point, // global_point
    point2d_t *local_point,  // local_point
    dma_event_t *event      // DMA event. If NULL: blocking,
                          // else return immediate and we must later call
                          // wait() on this event
);

// 2D (asynchronous) copy between on-chip (SNEM) and off-chip (DDR) memory
void bli_dma_backend_trigger_put2D(
    void* local,   // begin address of local buffer
    void* global,  // begin address of global buffer
    size_t size,   // size of an element in byte
    int width,     // block width in element
    int height,    // block height in element
    point2d_t *local_point,  // local_point
    point2d_t *global_point, // global_point
    dma_event_t *event      // DMA event. If NULL: blocking,
                          // else return immediate and we must later call
                          // wait() on this event
);

// Wait for termination of a DMA transfer
void bli_dma_backend_event_wait(dma_event_t *event);

// Return 1 if addr is in global memory, 0 otherwise.
int bli_dma_backend_addr_in_global_mem(void* addr);

./config/arch/bli_kernel.h :
#define BLIS_DMA_BACKEND_INIT bli_dma_backend_init
#define BLIS_DMA_BACKEND_FINALIZE bli_dma_backend_finalize
#define BLIS_DMA_BACKEND_TRIGGER_GET2D bli_dma_backend_trigger_get2D
#define BLIS_DMA_BACKEND_TRIGGER_PUT2D bli_dma_backend_trigger_put2D
#define BLIS_DMA_BACKEND_EVENT_WAIT bli_dma_backend_event_wait
#define BLIS_DMA_BACKEND_ADDR_IN_GLOBAL_MEM bli_dma_backend_addr_in_global_mem

./config/arch/kernels/dma/bli_dma_backend.c : Vendor implementation

./frame/base/dma/bli_dma.c : Reference memcpy-based implementation
PORTABLE BLIS-RDMA EXTENSION
WHAT HAS BEEN DONE?

1. New *bli_gemm_blk_var[1|3]_dma* and *bli_trsm_blk_var[1|3]_dma* functions injected into control trees

2. Some workarounds used to keep minimal footprint:
   - Square-rization of blocksize in case of *herm_or_symm* (sub-optimal)
   - Dynamic scratchpad allocation instead of using internal pools

3. DMA backend API and a reference memcpy-based implementation

4. Added user-defined scratchpad allocator
   ```
   #define BLIS_MALLOC_SCRATCHPAD utask_smem_malloc
   #define BLIS_FREE_SCRATCHPAD utask_smem_free
   ```

5. Passed the testsuite using memcpy on CPU on TravisCI
   ```
   ./configure --enable-dma auto
   make test
   ```
BLIS-RDMA on MPPA

Performance benchmark
BLIS-RDMA ON MPPA® 2-256
BOSTAN2

- BLIS running in single-cluster mode, upto 8 cores, 70-75 % peak
• BLIS running in single-cluster mode, up to 8 cores, 70-75% peak
FUTURE BLIS-RDMA ON MPPA®3 COOLIDGE

- Either single-cluster mode or full-chip as an [80|160]-core SMP (NUMA-like)
- 90-95 % peak
Either single-cluster mode or full-chip as an [80|160]-core SMP (NUMA-like)
90-95 % peak

**BLIS-RDMA (GFLOPS) (estimation)**

| MPPAv3® 160 | SGEMM @ 1200Mhz | 1382 | 22 GFLOPS/W |
| DGEMM @ 1200MHz | 345 | 6 GFLOPS/W |
| MPPAv3® 80 | SGEMM @ 1200Mhz | 691 | 20 GFLOPS/W |
| DGEMM @ 1200MHz | 172 | 5 GFLOPS/W |
| MPPAv2® 256 | SGEMM @ 500MHz | 409 | 9 GFLOPS/W |
| DGEMM @ 500MHz | 204 | 5 GFLOPS/W |
CONCLUSION

BLIS-RDMA IS THE MISSING PUZZLE PIECE

BLIS-RDMA can make BLIS useful everywhere

MANY OPTIMIZATION POSSIBILITIES

KALRAY CONTRIBUTES

Kalray is ready to contribute to BLIS to keep it open-source