Building Blocks for Sparse Linear Algebra on Heterogeneous Hardware

Moritz Kreutzer, Georg Hager, Georg Wellein, Jan Treibig

ICS
Munich, 10.6.2014
Selective Challenges

We are facing a list of challenges which we have to deal with:

1. Increasingly heterogeneous hardware
   - Well-known x86 CPUs are working together with accelerators/co-processors
   - Inherently different programming paradigms
   - Few transparently heterogeneous libraries
Selective Challenges

We are facing a list of challenges which we have to deal with:

2. Increasing level of hardware parallelism
   - Higher hardware performance only due to more parallelism
   - Application may have limited scalability with standard approaches (e.g., data parallelism)
   - Novel levels of parallelism (e.g., task parallelism) may be cumbersome to implement by application developers in an efficient way
Selective Challenges

We are facing a list of challenges which we have to deal with:

3. Library performance is often limited due to generality
   - Application knowledge is a key to high library performance
     - E.g., we can fuse kernels instead of calling them sequentially
   - Established libraries may not perform well in specific cases
     - Prominent example: Calling GEMM with tall skinny matrices may deliver poor performance even for highly-optimized BLAS libs
A library which delivers highly efficient building blocks for sparse linear algebra ("General, Hybrid and Optimized Sparse Toolkit")

- Several levels of parallelism: MPI, OpenMP, CUDA, SIMD
- Transparent use of heterogeneous hardware
- Generic interface for hardware-affine task-level parallelism
- Highly-optimized low-level kernels (e.g. SELL-C-σ SPMVM, BLAS1)

Work supported by DFG through Priority Programme 1648 “Software for Exascale Computing” (SPPEXA) under project ESSEX (“Equipping Sparse Solvers for Exascale”)
GHOST Overview

- Upcoming (initial release for 2014):
  - General: Sparse building blocks
  - Hybrid: Various matrix formats incl. SELL-C-σ
  - Optimized: Hybrid/heterogeneous MPI+X parallelism
  - Sparse: Communication hiding
  - Toolkit: Built-in threading & tasking model
Sparse Matrix Format Jungle
SpMVM in the Heterogeneous Era

- Compute clusters are getting more and more heterogeneous

- A special format per compute architecture
  1. hampers runtime exchange of matrix data
  2. complicates library interfaces

- CRS (CPU standard format) may be problematic
  - Vectorization along matrix rows
  - Bad utilization for short rows and wide SIMD units (Intel MIC: 512 bit)

➤ We want to have a unified, SIMD-friendly, and high-performance sparse matrix storage format.
Sliced ELLPACK

- Well-known sparse matrix format for GPUs

- Entries and column indices stored column-wise in chunks
- One parameter:
  1. C: Chunk height
Minimizing the storage overhead $\Rightarrow$ SELL-C-$\sigma$

- Sort rows within a range $\sigma$ to minimize the overhead
  - $\sigma$ should not be too large in order to not worsen the RHS vector access pattern

- Two parameters:
  1. $C$: Chunk height
  2. $\sigma$: Sorting scope
Using a unified storage format comes with little performance penalty in the worst case and up to a 3x performance gain in the best case for a wide range of test matrices.
Example Node Partitioning

- Minimal amount of MPI processes on this node: 3
- GPU is managed by a full core on the nearest socket
- CPU process spans two NUMA nodes
- Xeon Phi operated in native mode
  - one MPI process running on the coprocessor
SpMVM Performance in a Heterogeneous System

Two 10-core sockets of Intel Xeon Ivy Bridge and two Nvidia Tesla K20m GPUs

- 2x Ivy: 12 Gflops/s
- K20m: 22 Gflops/s
- 2x Ivy + K20m: 33 Gflops/s
- 2x K20m: 43 Gflops/s
- 2x Ivy + 2x K20m: 53 Gflops/s
- 2x Ivy + 2x K20m (ideal): 56 Gflops/s

(ML_Geer matrix, 64-bit values, 32-bit indices, ECC=1)
Conclusion

- Wide SIMD/SIMT architectures pose challenges for spMVM
  - Short loops (CRS)
  - Fill-in (ELLPACK)
  - Reduction overhead (CRS)
  - Low vectorization ratio (CRS)
- SELL-C-σ alleviates or eliminates most of these problems
- GHOST addresses major challenges in current and future systems to enable high performance parallel sparse solver applications
THANK YOU.