Performance Engineering for Algorithmic Building Blocks in the GHOST Library

Georg Hager, Moritz Kreutzer, Faisal Shahzad, Gerhard Wellein, Martin Galgon, Lukas Krämer, Bruno Lang, Jonas Thies, Melven Röhrig-Zöllner, Achim Basermann, Andreas Pieper, Andreas Alvermann, Holger Fehske

Erlangen Regional Computing Center (RRZE)
University of Erlangen-Nuremberg
Germany

ESSEX-II Minisymposium @ SPPEXA Annual Plenary Meeting
January 25, 2016
Garching, Germany
Outline

Performance Engineering (PE)

The GHOST library

Work planned for ESSEX-II
The whole PE process at a glance
Kernel Polynomial Method
• Compute spectral properties of quantum system (Hamilton operator)
• Approximation of full spectrum
• Naïve implementation: SpMVM + several BLAS-1 kernels

Example: KPM

for \( r = 0 \) to \( R - 1 \) do
    \( |v\rangle \leftarrow |\text{rand()}\rangle \)
    Initialization steps and computation of \( \eta_0, \eta_1 \)
for \( m = 1 \) to \( M/2 \) do
    swap\((|w\rangle, |v\rangle)\)
    \( |u\rangle \leftarrow H|v\rangle \)
    \( |u\rangle \leftarrow |u\rangle - b|v\rangle \)
    \( |w\rangle \leftarrow -|w\rangle \)
    \( |w\rangle \leftarrow |w\rangle + 2a|u\rangle \)
    \( \eta_{2m} \leftarrow \langle v|v\rangle \)
    \( \eta_{2m+1} \leftarrow \langle w|v\rangle \)
end for
end for

Application: Loop over random initial states
Algorithm: Loop over moments

Building blocks: (Sparse) linear algebra library

Augmented Sparse Matrix Vector Multiply
Multiple Vector Multiply
Augmented Sparse Matrix Vector Multiply
Dot Product

\( \triangleright \text{spmv}() \langle v|v\rangle \)
\( \triangleright \text{axpy}() \langle H - b|v\rangle - |w\rangle \& \)
\( \triangleright \text{scal}() = \langle v|v\rangle \& \)
\( \triangleright \text{axpy}() \)
\( \triangleright \text{nrm2}() \langle w|v\rangle \)
\( \triangleright \text{dot}() \)
Step 1: naïve → augmented (fused) kernel
- Naïve kernel is clearly memory bound
- Better resource utilization
- $B_C = 3.39 \text{ B/F} \rightarrow 2.23 \text{ B/F}$
- Still memory bound → same pattern

Step 2: augmented → blocked
- Augmented kernel is memory bound
- $R = \# \text{ of random vectors}$
- $B_C = 2.23 \text{ B/F} \rightarrow (1.88/R + 0.35) \text{ B/F}$
- Decouples from main memory BW

→ Performance portability becomes well defined!
What about the decoupled model?

\[ \Omega = \frac{\text{Actual data transfers}}{\text{Minimum data transfers}} \]

Why does it decrease?
The GHOST library

General Hybrid Optimized Sparse Toolkit

GHOST design guidelines

- Strictly support the requirements of the project
- Enable fully heterogeneous operation
- Limit automation
- Do not force dynamic tasking
- Do not force C++ or an entirely new language
- Stick to the well-known “MPI+X” paradigm
- Support data parallelism via MPI+X
- Support functional parallelism via tasking
- Allow for strict thread/process-core affinity
Task parallelism: Asynchronous checkpointing with GHOST tasks

Parent task

```c
ghost_task_create( ckpt_task_ptr, &CP_func, CP_obj,...)
```

```c
update_CP(CP_obj);
// async. copy of CP is updated
```

```c
ghost_task_wait(ckpt_task_ptr);
```

```c
ghost_task_enqueue(ckpt_task_ptr);
```

Checkpoint task

```c
CP_obj:
• void* to object of ckpt_t type
• ckpt_t class is defined by programmer
• checkpoint object contains the asynchronous copy of the checkpoint
```

```c
CP_func()
// This function takes an updated copy of CP_obj as argument and writes to PFS
```
Heterogeneous performance?

The need for hand-engineered kernels

Block vector times small matrix performance of GHOST and existing BLAS libraries (*tall skinny ZGEMM*)
SELL-C-σ

Performance portability for SpMVM
Constructing SELL-C-σ

1. Pick chunk size $C$ (guided by SIMD/T widths)
2. Pick sorting scope $σ$
3. Sort rows by length within each sorting scope
4. Pad chunks with zeros to make them rectangular
5. Store matrix data in “chunk column major order”

6. “Chunk occupancy”: fraction of “useful” matrix entries

$$\beta = \frac{N_{nz}}{\sum_{i=0}^{N_c} C \cdot l_i}$$

$$\beta_{\text{worst}} = \frac{N + C - 1}{CN} \approx \frac{1}{C}$$

$\beta = 0.66$
What is performance portability?
ESSEX-II and GHOST
1. **Building blocks development**
   - Improved support for *mixed precision kernels*
   - Fast *point-to-point sync* on many-core
   - High-precision *reductions*
   - *(Row-major storage TSQR)*
   - Full support for heterogeneous hardware *(CPU, GPGPU, Phi)*

2. **Optimized sparse matrix data structures**
   - Identify promising candidates *(ACSR, CSX)*
   - Exploiting *matrix structure*: symmetry, sub-structures

3. **Holistic power and performance engineering**
   - Comprehensive instrumentation of GHOST library functions
   - ECM performance modeling of SpMMVM and others
   - Energy modeling of building blocks
   - Performance modeling beyond the node

4. **Comprehensive documentation**
float sum = 0.0;
for (int i=0; i<N; i++) {
    sum = sum + a[i] * b[i]
}

1 ADD, 1 MUL

float sum = 0.0, c = 0.0;
for (int i=0; i<N; ++i) {
    float prod = a[i]*b[i];
    float y = prod-c;
    float t = sum+y;
    c = (t-sum)-y;
    sum = t;

4 ADD, 1 MUL

• No impact of Kahan if any SIMD is applied
• Compilers do not cut the cheese
• Method adaptable to other applications (e.g., other high-precision reductions, data corruption checks)
Example: Energy analysis of KPM

• Time to solution has lowest-order impact on energy

• Tailored kernels are key to performance (4.5x in runtime & energy)

• Energy-performance models yield correct qualitative insight

• Future: Large-scale energy analysis & modeling

\[ E(n) = F \cdot \frac{W_{00} + n(W_{01} + W_1 f + W_2 f^2)}{\min(nP_0(f), P_{max})} \]

Energy-performance model

IVB 2.2 GHz
Download our building block library and applications: http://tiny.cc/ghost

General, Hybrid, and Optimized Sparse Toolkit

Thank you.