FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

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

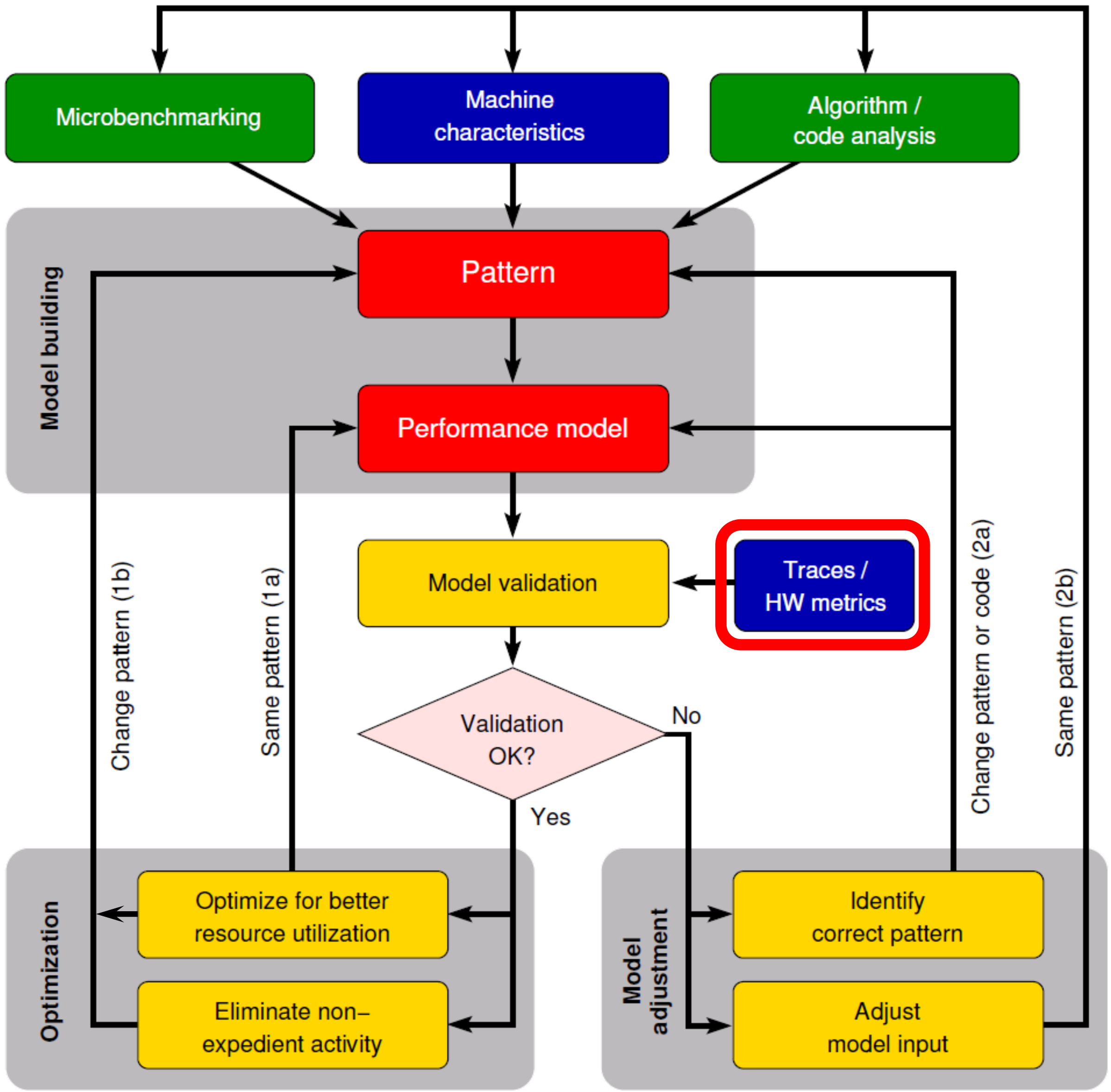
Outline

Performance Engineering (PE)

The GHOST library

Work planned for ESSEX-II

The whole PE process at a glance



Example: KPM

Kernel Polynomial Method

- Compute spectral properties of quantum system (Hamilton operator)
- Approximation of full spectrum
- Naïve implementation: SpMVM + several BLAS-1 kernels

for $r = 0$ to $R-1$ **do** — Application: Loop over random initial states
  $|v\rangle \leftarrow |\text{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do** — Algorithm: Loop over moments
    $\text{swap}(|w\rangle, |v\rangle)$
    $|u\rangle \leftarrow H|v\rangle$ ▷ spmv()
    $|u\rangle \leftarrow |u\rangle - b|v\rangle$ ▷ axpy()
    $|w\rangle \leftarrow -|w\rangle$ ▷ scal()
    $|w\rangle \leftarrow |w\rangle + 2a|u\rangle$ ▷ axpy()
    $\eta_{2m} \leftarrow \langle v|v\rangle$ ▷ nrm2()
    $\eta_{2m+1} \leftarrow \langle w|v\rangle$ ▷ dot()
  end for
end for

Building blocks: (Sparse) linear algebra library

Dot Product

- 1 **do**
$\rangle$
steps and computation of η_0, η_1
$M/2$ **do**
$, |v\rangle)$
$(H - b\mathbb{1})|v\rangle - |w\rangle$ &
$= \langle v|v\rangle$ &
$= \langle w|v\rangle$ ▷ aug_spmv()
$[:] = \langle w|v\rangle$ ▷ aug_spmv()

Augmented Sparse Matrix Vector Multiply

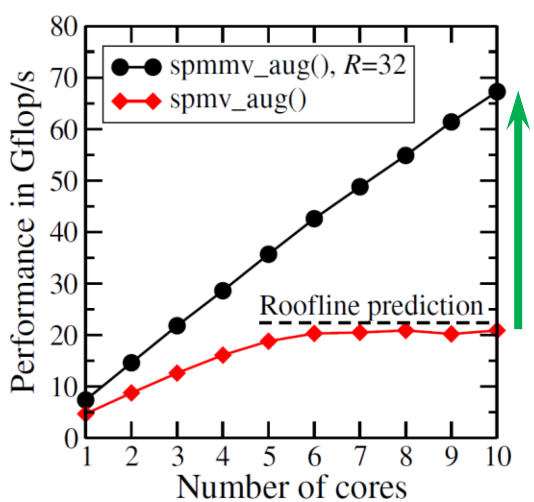
Augmented Sparse Matrix Multiple Vector Multiply

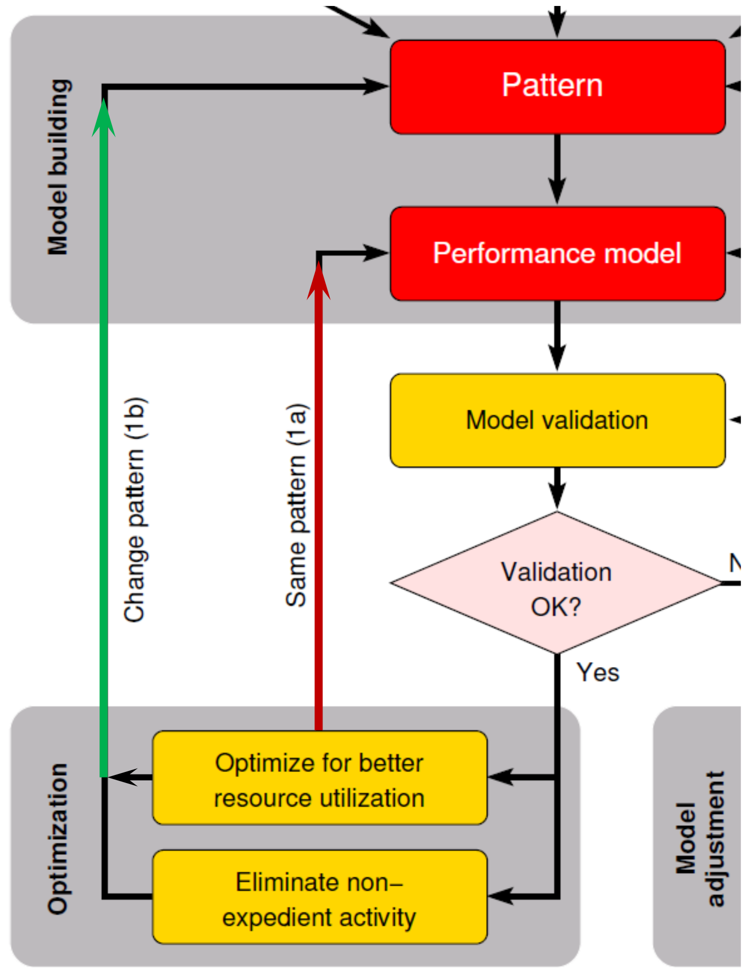
Step 1: naïve → augmented (fused) kernel

- Naïve kernel is clearly memory bound
- Better resource utilization
- B_C = 3.39 B/F → 2.23 B/F
- Still memory bound → same pattern

Step 2: augmented → blocked

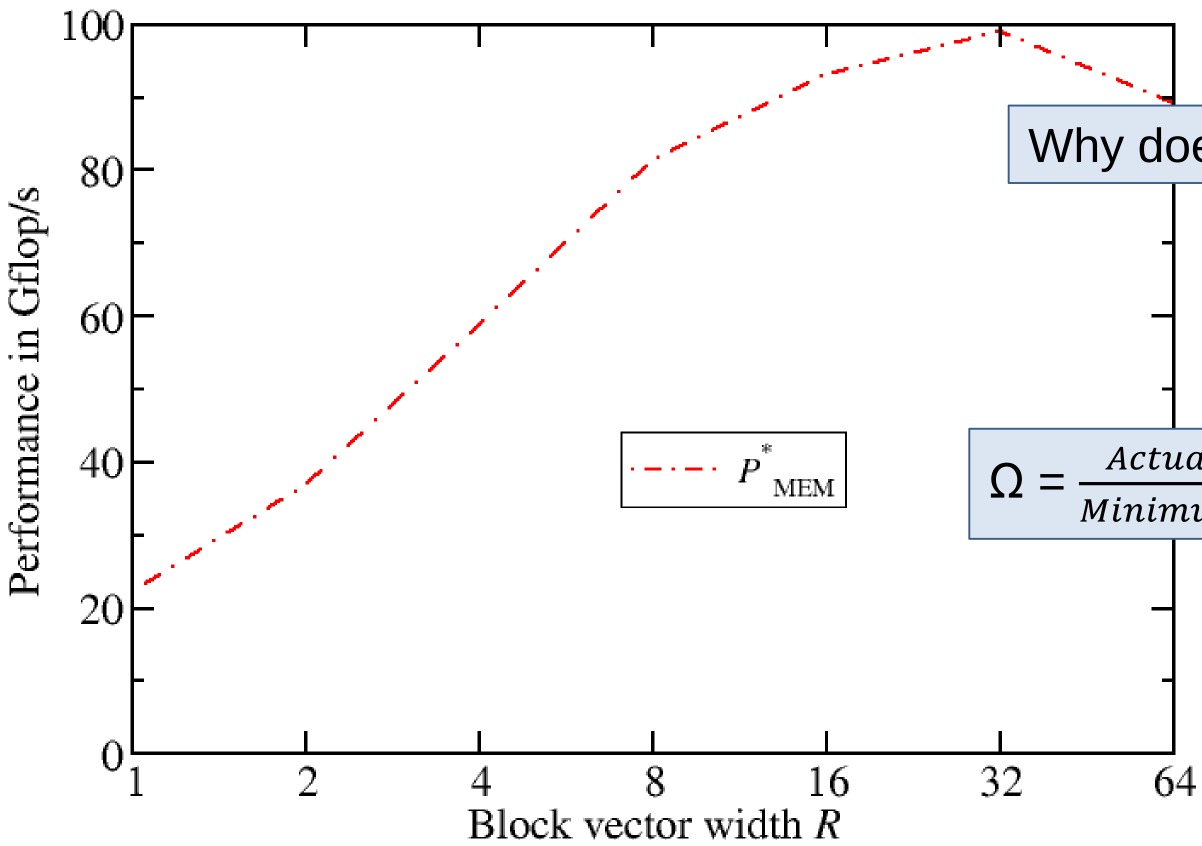
- Augmented kernel is memory bound
- R = # of random vectors
- B_C = 2.23 B/F → (1.88/R + 0.35) B/F
- Decouples from main memory BW





→ **Performance portability** becomes well defined!

What about the decoupled model?



Why does it decrease?

$$\Omega = \frac{Actual\ data\ transfers}{Minimum\ data\ transfers}$$

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

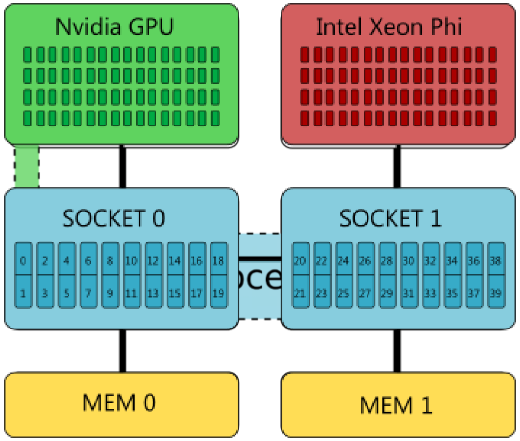
The GHOST library

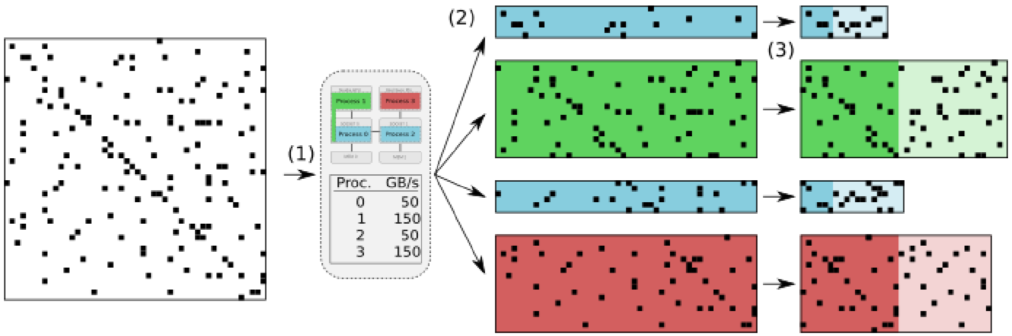
General Hybrid Optimized Sparse Toolkit

M. Kreutzer et al.: GHOST: Building blocks for high performance sparse linear algebra on heterogeneous systems.
Preprint arXiv:1507.08101

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

```
ghost_task_create( ckpt_task_ptr,&
     CP_func, CP_obj,…)

update_CP(CP_obj);
```

// async. copy of CP is updated

```
ghost_task_wait(ckpt_task_ptr);

ghost_task_enqueue(ckpt_task_ptr);
```

iterative

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

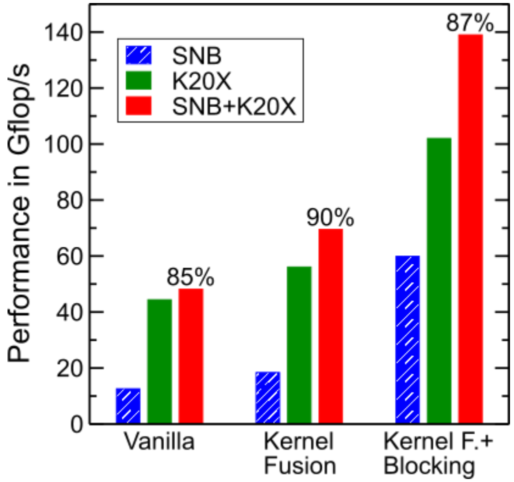
Checkpoint task

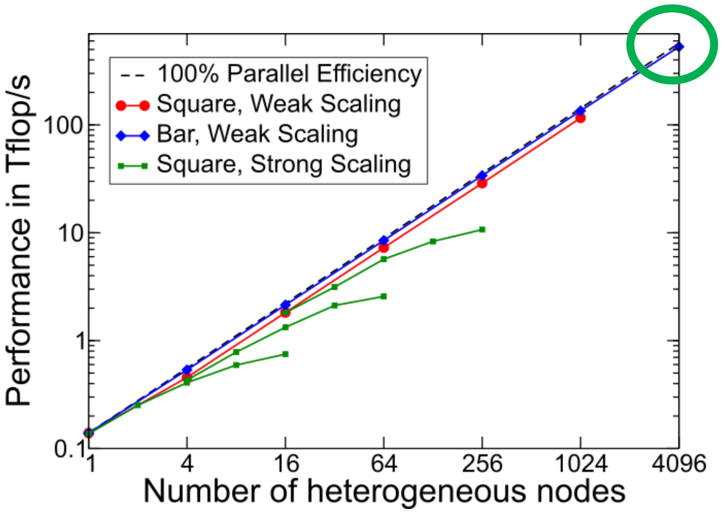
CP_func()

// This function takes an updated copy of **CP_obj** as argument and writes to PFS

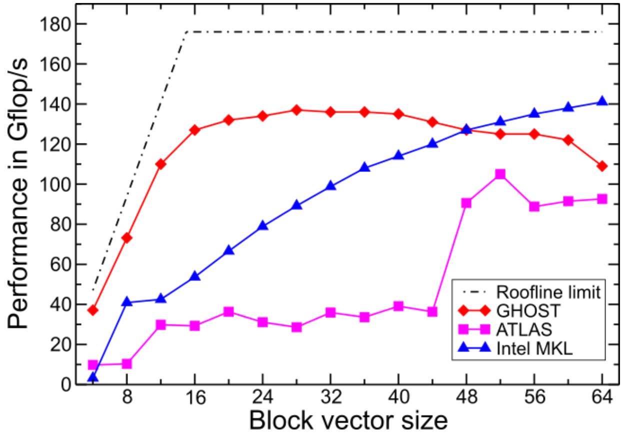
Heterogeneous performance?

0.5 Pflop/s





The need for hand-engineered kernels



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

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

SELL-C-σ

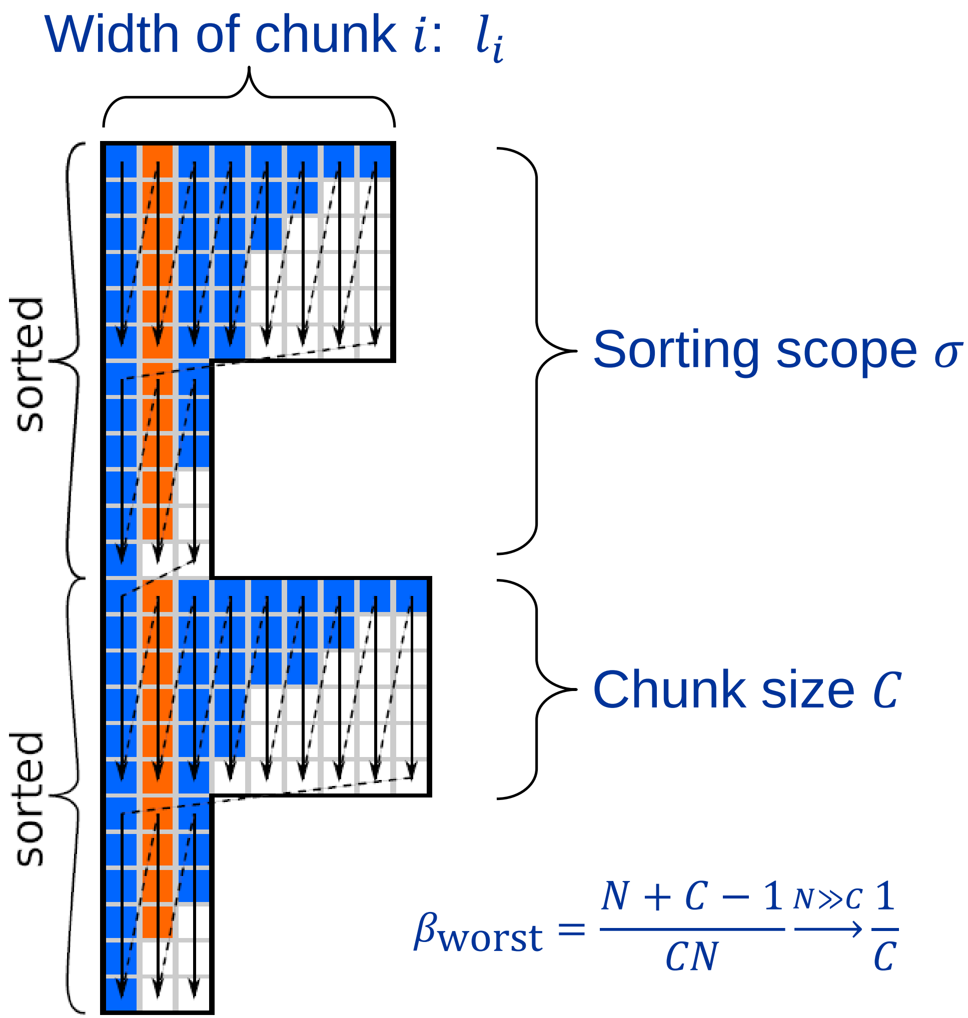
Performance portability for SpMVM

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

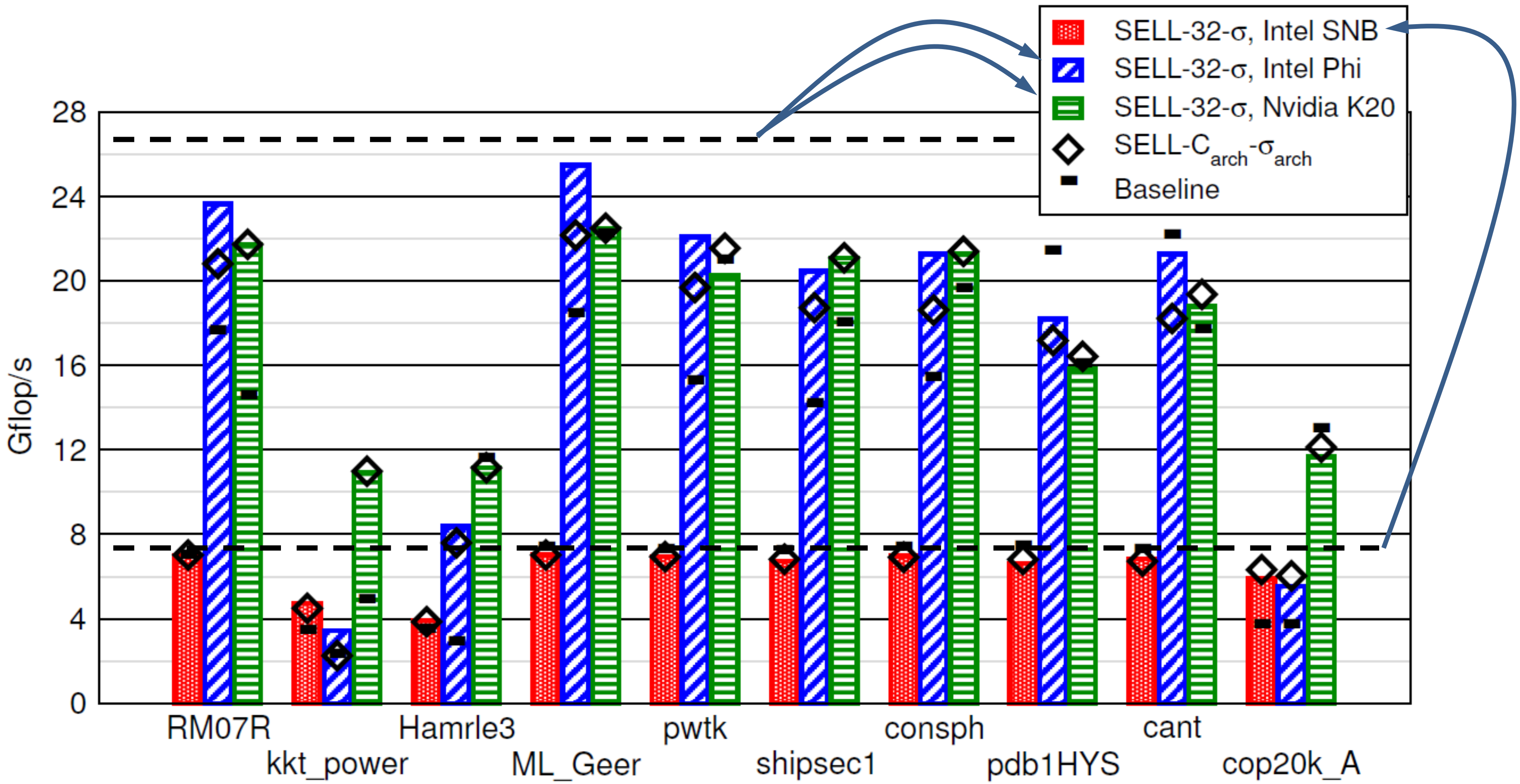
$$\beta = \frac{N_{nz}}{\sum_{i=0}^{N_c} C \cdot l_i}$$



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

SELL-6-12
β=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**

Example: performance impact of the Kahan-augmented dot product

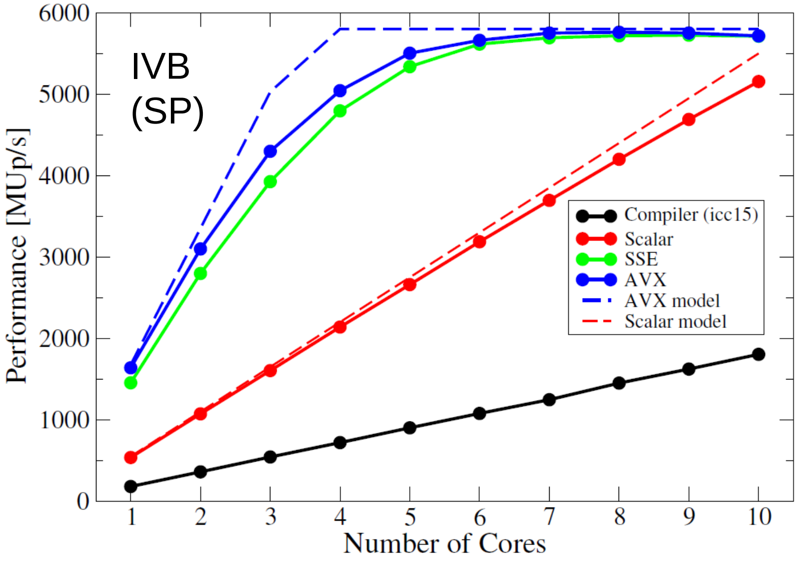
```
float sum = 0.0;
  for (int i=0; i<N; i++) {
  sum = sum + a[i] * b[i]
}
```

1 ADD, 1 MULT

```
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;
}
```

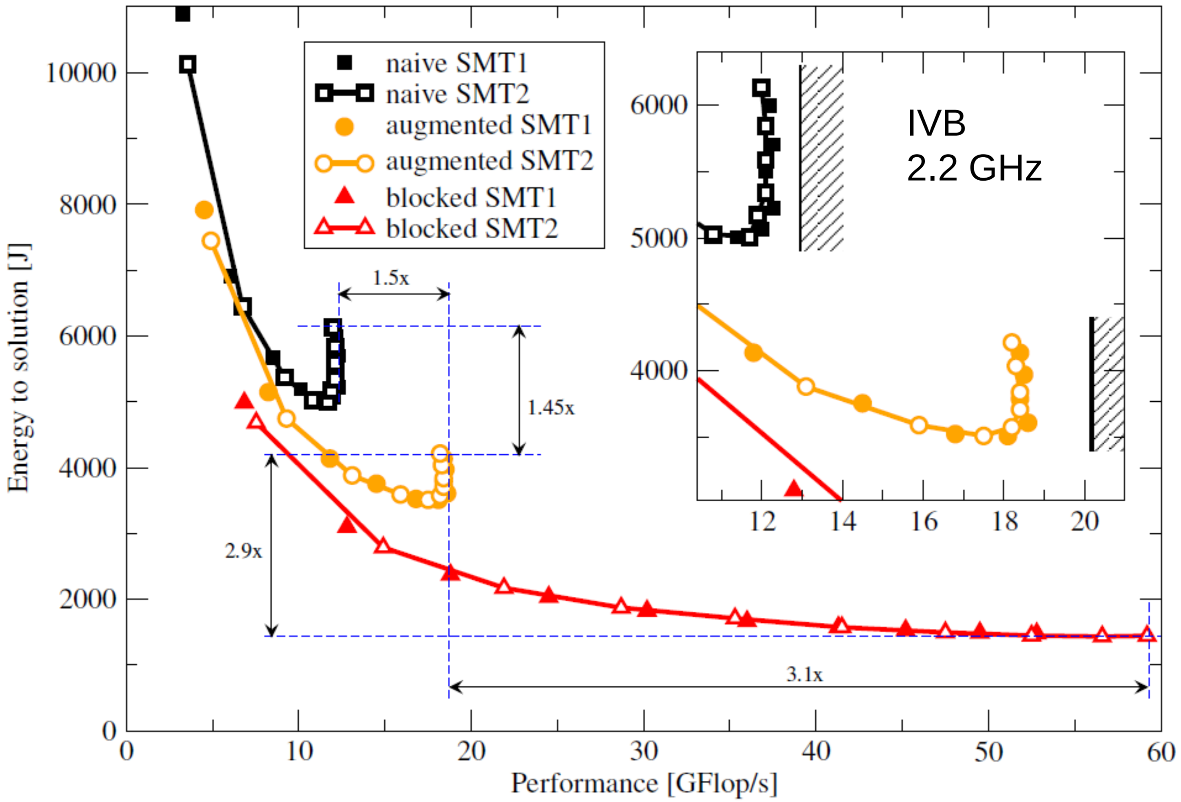
4 ADD, 1 MULT

- 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

Download our building block library and applications:

`http://tiny.cc/ghost`

GHOST *General, Hybrid, and Optimized Sparse Toolkit*

Thank you.

SPPEXA