Towards ATLAS 4.0

R. Clint Whaley
rcwhaley@lsu.edu
www.csc.lsu.edu/∼whaley
Rakib Hasan (rhasan@cct.lsu.edu)
Majedul Sujon (msujon@cct.lsu.edu)

Louisina State University
Computer Science and Engineering Division (CSC) &
Center for Computation & Technology (CCT)

May 19, 2016

Towards ATLAS 4.0

NSF CAREER OCI-1149303, "Empirical Tuning for Extreme Scale"

Improving Kernels and Exploiting SIMD Vectorization

1. Generic vectorization using `atlas_simd.h` & `atlas_cplxsimd.h`
2. New tuning framework supporting SIMD-friendly storage
3. Extending tuning framework for additional kernels
4. Using FKO/iFKO for backend compilation

Improving Thread-Level Parallelism

5. Cache-based communication (CBC) for hardware-speed comms
   → req careful R&D for weakly-ordered caches
6. Thread-pool rather than launch & join to reduce overhead
7. Investigate blocking & scheduling for extreme-scale shared mem

? heterogenuous (big/little), fast scheduling (gatmcctr, gbitvec)

Generic SIMD vectorization

atlas_simd.h & atlas_cplxsimd.h

atlas_simd.h: Generic VLEN-agnostic SIMD primitives

Must avoid kernel rewrite when VLEN or machine changes:

- Macro ATL_VLEN tells you how long each vector is
- ATL_vvrsumI (I all pwr2 $\leq VL$; eg., vvrsum4/2/1) provides parallel reduction for accumulators (needed for vectorization along dot product dim)
- Basic ops supported for: ARM32(NEON)/ARM64(AdvSIMD), POWER (VSX), x86 (AVX[2]/SSE[1-3]), and gcc's builtins
- Model may break down in some cases, so generic macros can lose performance in edge cases
  - Unaligned access handled differently to optimize, esp. POWER

  $\rightarrow$ I need more experience on PWR & ARM

New GEMM tuning framework

Key Ideas

1. All timing/tunings in parallel
2. Huge # of NBs: sml probs; target last private/LLC
3. Presently supports M- or K-vec access-major storage
4. Code gens (cpy&amm) using `atlas_simd.h`

$\Rightarrow$ Allow user to build custom μkernels and call directly (PCA)

Challenges

1. Search not yet optimized
2. Serial haswell: MKL:92.1%, ATL:89.2%
3. 24-core haswell-e: MKL:86.6%; ATL:82.8%/85.6%

$\Rightarrow$ Multilvl blking, bad tuning, diff format?

$\rightarrow$ looking at supporting arbitrary formats

Extending GEMM μkernel to other BLAS

GEMM μkernel can probably help all L3 except TRSM

- With very large NB, diagonal blks remain important for perf
- Add ability to generate register-blocked block-lower storage
- Now SYRK/TRMM/SYRK supported by changing one loop param in amm μkern
- Have support for SYRK using k-vec storage $M_u = N_u$

$\rightarrow$ Need more info to see if this is worth complexity

TRSM

- Idea is recursive/multilvl gemm-based
- Not clear we can keep overhead sufficiently low

iterative Floating Point Kernel Optimizer

iFKO is an iterative and empirical compiler framework

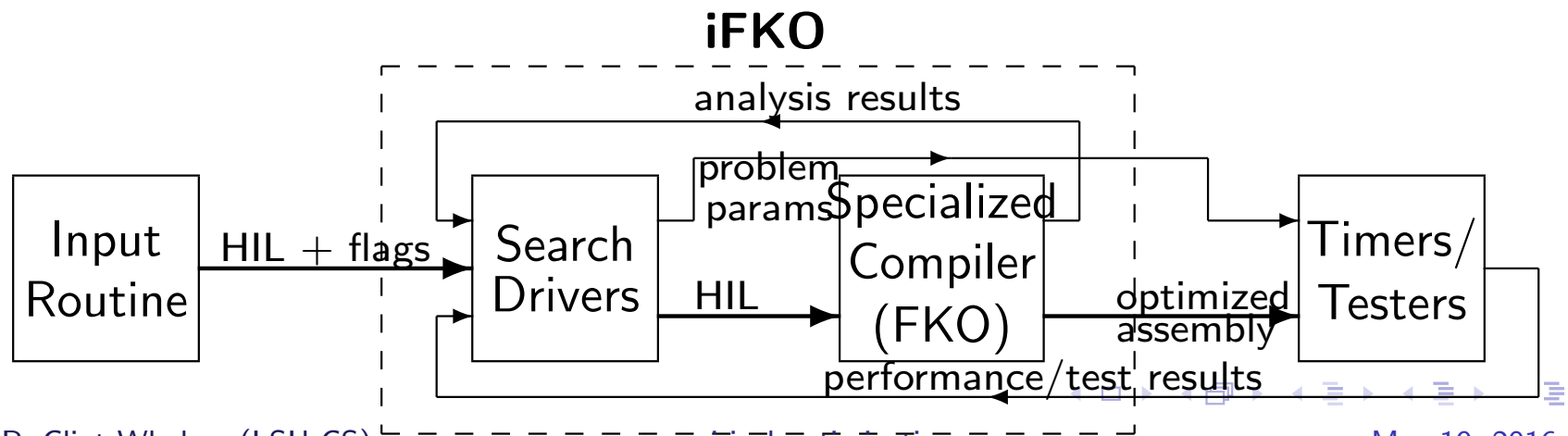
Attempt to extend impact past LA to all HPC computation.

iFKO composed of:

1. A collection of search drivers,
2. a compiler specialized for empirical floating point kernel optimization (FKO)
   - Specialized in analysis, HIL, type & flexibility of transforms

Key design decisions:

- Iterative & empirical for auto-adaptation
- Transforms done at backend, allowing arch exploitation (eg. SIMD vect, CISC inst formats)
- Kernel annotation markup
- Narrow & deep focus



Status of iFKO

Recently published work

- Majedul Haque Sujon, R. Clint Whaley and Qing Yi, "Vectorization Past Dependent Branches Through Speculation", in *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 353-362, Edinburgh, Scotland, September, 9-11, 2013.

Ongoing & recent work in iFKO

- Auto-vectorize mixed-type inst (eg., IAMAX)
- Effectively handle nested loops
- Support for 2-D arrays
- Improved support for register exhaustion
- Outer loop vectorization & vector reduction parallelization
- Half incorporated into ATLAS framework