Automatically Tuned Linear Algebra Software

R. Clint Whaley

whaley@cs.utsa.edu

www.cs.utsa.edu/∼whaley

University of Texas at San Antonio
Department of Computer Science

September 20, 2007

What is ATLAS?

- Provides high performance dense linear algebra routines:
  - BLAS, some LAPACK
- Automatically adapts itself to differing architectures using empirical techniques

Why is ATLAS needed?

- Well-tuned linear algebra routine runs orders of magnitude faster than generic implementation
- Hand-tuning is architecture specific
- No such thing as enough compute speed for many scientific codes

ATLAS Usage and Tech Transfer

- Scientific simulation:
  - physics, chemistry, biology, astronomy, engineering, math
- Almost all supercomputers
- Many OSes include ATLAS:
  - OS X, most Linux & BSDs
- Most PSEs:
  - Maple, Mathematica, Matlab, Octave
- Multitude of software:
  - GSL, HPL, SciPy, R, some compilers, etc.
- Host of research projects
  - You? – send link

ATLAS History and Funding

ATLAS History

- Originally developed at ICL/UTK
  - ICL – Jack Dongarra
  - First pub 1997, post-PHiPAC
  - Started by Clint Whaley
  - Joined by Antoine Petitet
    - Rec. gemm/ger/gemv-based BLAS
    - Pthreads, ifaces, etc.
  - Open source contrib
  - Left ICL 2001
- 3.6 rel 2003 at FSU (AMD)

ATLAS Present

- New funding 2006 at UTSA:
  - NSF CRI CNS-0551504 - 3/06
  - DoD ATLAS R&D contract - 7/06
- 37 developer releases
- 1$^{\text{st}}$ stable release in almost 4 years this month (heh)
- More than 58,000 direct downloads during award
  - Most users get ATLAS via repackegers
- Error analysis research

ATLAS philosophy: AEOS

Automated Empirical Optimization of Software (AEOS)

Key Idea

Probe machine empirically, accepting only those transforms that result in measurable improvements, just as scientific method probes nature. Automate the empirical probing so that code is tuned by computer (w/o extensive arch-specific info), rather than team of experts

Goal

Optimized, portable kernel (wt associated library) available in hours rather than months or years (or never).

AEOS
Overview and Requirements

Basic idea

- Mach srch opt space
- Finds app-apparent arch

AEOS Requires

- Define simplist & most reusable kernel(s)
- Sophisticated timers
- Robust search heuristic
- Method of software adapt.

ATLAS's Methods of Soft. Adapt.

1. **Parameterization**: vars provide differing implementations (eg., N_B).
   - Easy to implement, limited
2. **Mult Implem**: linear srch of rout list
   - Simple to implement, simple for external contribution
   - Low adaptability, ISA independent, kernel dependent
3. **Source generator**: heavily paramed prog generates varying impl.
   - Very complicated to program, search, and contribute
   - High adaptability, ISA independent, kernel dependent

ATLAS Introduction

Empirically tunes BLAS & some LAPACK for arbitrary machines

BLAS: Basic Linear Algebra Subprograms

Building block routines – must be optimized for each machine.

- Level 1 BLAS: vector-vector operations (∼50 routines)
  - $y \leftarrow x$, $dot \leftarrow x^T y$, $y \leftarrow \alpha x + y$, etc
  - $O(N)$ flops, $O(N)$ data $\Rightarrow$ bus-bound
- Level 2 BLAS: Matrix-vector operations (66 routines)
  - $y \leftarrow \alpha Ax + \beta y$, $A \leftarrow \alpha xy^T + A$, $x \leftarrow A^{-1}x$, etc.
  - $O(N^2)$ flops, $O(N^2)$ data $\Rightarrow$ bus-bound
- Level 3 BLAS: Matrix-Matrix operations (30 routines)
  - matmul, symmetric update, matrix solve, etc.
  - $O(N^3)$ flops, $O(N^2)$ data $\Rightarrow$ FPU-bound when optimized

LAPACK: Linear Algebra Package (BLAS provide perf)

Eigenvalues, factorizations, least-squares solve, iterative refin, etc.

ATLAS Support by Level

Overview of ATLAS's Degree of Support for Various Classes of Routine

Level 3 BLAS Well Optimized

- Pthreads for parallel support
- Level 3 use recursive gemm-based BLAS
- Performance based on gemm kernel
  - Source generation + parameterizion
  - Multiple implementation + parameterization

Level 1 and 2 BLAS Not-So-Well Optimized

- No threading
- Level 2 use ger-/gemv-based recursive/blocked BLAS
- Multiple implementation + parameterization – only a few kernels

Limited LAPACK Support: LU, Cholesky, ILAENV

Code Generator Details

Present Code Generator

- Written in ANSI C, output ANSI C
- Unrolling on all three loops
  - outer loops are jammed, imply register block
- Load C at top or bottom of loop
- Two prefetch strategies:
  - On comp w/o pref or asg support, empty macros
  - pref nxt blk of A while wrking on this one
  - pref C when not loaded at top
- Peel of 1st k iteration for $\beta = 0$
- Software pipelining of mul and add

Present Code Gen

- Reg blking
- L1 blk, N_b
- MAC or mul+add
- Crude ld sched
- Various C src optimizations

Lacking

- Ld/use pipelining
- More pref sched
- SIMD Vectorization

Multiple Implementaton Details

- Developers ‘scratch own itch’ & help community
- Standard tester/timer reads index file for list of routs
  - Testers rule out kernels spec to other archs
  - Search linear except for ruling out classes (SSE, comp/flg)
  - Further tuned via param (L1 and L2 blk, etc), polyalgorithmic
- Always combined wt parameterization:
  - L3 BLAS : tiling for L2 and L1, extensively polyalgorithmic
  - L2 BLAS : tiling for one level of blocking, rec for very lrg mat
  - L1 BLAS : limited complex/real reuse
- Use hand kernel tuning as proving ground for new optimizations

Lessons from ATLAS

Why Multiple Implementation and iFKO?

- Compiler changes make kernels very fragile
- ATLAS+code generator lost to hand-tuning overwhelmingly because of lack of backend-optimizations – not search or tile
  - SIMD Vectorization (2-8$\times$)
  - Reg asg (gcc spill in loops)
  - Front-end optimizations (code alignment, decoding, inst win):
    - Athlon classic (75% $\rightarrow$ 92%): align, nop, group
    - Core2Duo (71% $\rightarrow$ 78%): CISC compaction, align
    - PowerPC970FX (69% $\rightarrow$ 86%): flights-of-four, align
    - Opteron $\sim$5% speedup for LU: CISC compaction
  - Differing prefetch strategies and schedules

$\Rightarrow$ For DLA, need a host of tunable opt phases, many of which don't exist in present compilers, and cannot be done src-to-src

New in Forthcoming Stable

ATLAS3.8.0: Now with stone tools & Fire!

- Code generator improvements:
  - prefetch, ld top/bott, peel-K
- Extensive arch improv
  - P4, Eff, P4E, C2D, Opt, MIPS, SSE3, AV
- More assembly support (32 & 64 bits)
  - x86, x8664, PPC, PARISC, MIPS
- New config & build mech
- Improved ILAENV helps non-ATLAS LAPACK
- Improved error on some archs
- Improved shape handling:
  - Long-K, small M,N ($\times 2$)
  - Rank-[1-4] K update
- Improved complex perf on some arch

Further details

- ATLAS homepage:

  `http://math-atlas.sourceforge.net/`
- My homepage:

  `http://www.cs.utsa.edu/~whaley/`
- How to use ATLAS's gemm kernel to speed up packed (rec): "Minimizing Development and Maintenance Costs in Supporting Persistently Optimized BLAS", *Software: Practice & Experience*, Volume 35, Number 2, pp 101-121, February, 2005.
- Error analysis technical report:

  `http://www.cs.utsa.edu/research/tr/2007/CS-TR-2007-002.pdf`