

Iterative Compilation for Fast Inner Kernels for Linear Algebra Codes

José-Ramón Herrero

Computer Architecture Department
Universitat Politècnica de Catalunya

Currently on sabbatical leave at
Barcelona Supercomputing Center

E-mail: josepr@ac.upc.edu

URL: <http://personals.ac.upc.edu/josepr/>

Adaptive Compilation
(HiPEAK'08 Cluster Meeting)

Barcelona, Spain, June 3rd, 2008

Outline

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Specialized inner kernels

Results

Ideas

Specialized inner
kernels

Results

Ideas

Outline

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Specialized inner kernels

Results

Ideas

Specialized inner
kernels

Results

Ideas

Overview

Our context: Linear Algebra codes

- ▶ Dense
- ▶ Sparse [▶ More Info](#)

In search for high performance:

- ▶ **Efficiency of inner kernel** is of paramount importance.

Usual approach:

- ▶ Ad-hoc codes written in assembler.

Our approach:

- ▶ Compiler-optimized inner kernel for operation on small matrices

Compiler-optimized inner kernels

Our approach:

- ▶ Compiler-optimized inner kernel for operation on small matrices
 - ▶ Collection of codes written in high level language;
 - ▶ Use compiler to generate optimized object code.
 - ▶ Insert best code in library: Small Matrix Library (SML).

Use SML routines for general codes.

SML: Idea

- ▶ Write several variants of code
 - ▶ Loop order
 - ▶ Loop unrolling factors
- ▶ Use the *best* compiler available
 - ▶ Try several compiler optimization flags

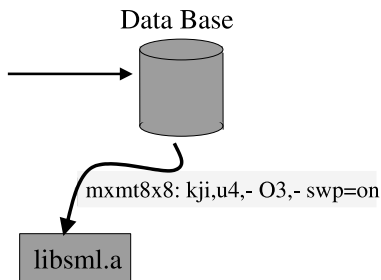
SML: Large search space

- ▶ Many combinations
 - ▶ Leading dimensions
 - ▶ Loop limits
 - ▶ Loop orders
 - ▶ Loop unrolling factors
 - ▶ Compiler flags
 - ▶ Target machines

- ▶ We need to automate the tests
 - ▶ Use a benchmarking tool

SML: Use a Benchmarking Tool

- foreach parameter combination
 - compile
 - execute
 - store results (Mflops)



- select best combination
- add object to library

Outline

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Specialized inner kernels

Specialized inner
kernels

Results

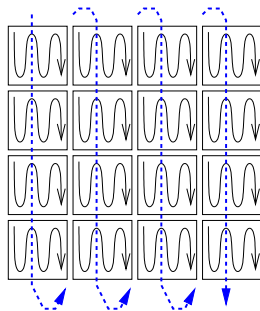
Results

Ideas

Ideas

Simple Square Block (SB) storage:

matrices aligned and stored by submatrices.



Simple SB storage: $C = C - A^t \times B$

Results on Power4

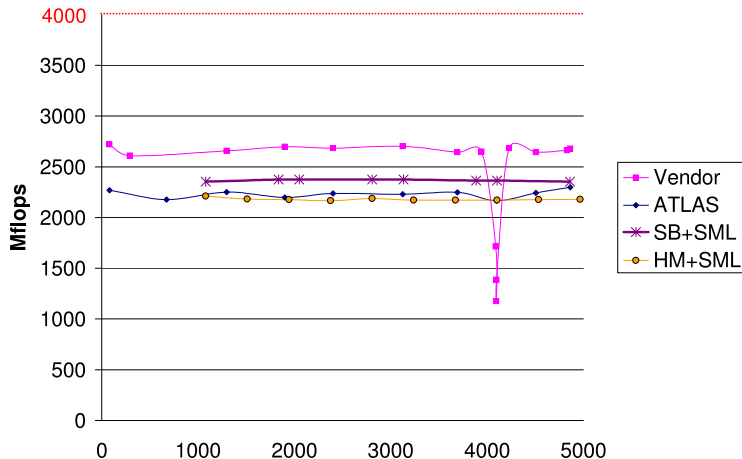
Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Specialized inner
kernels

Results

Ideas



Simple SB storage: $C = C - A^t \times B$

Results on Pentium4

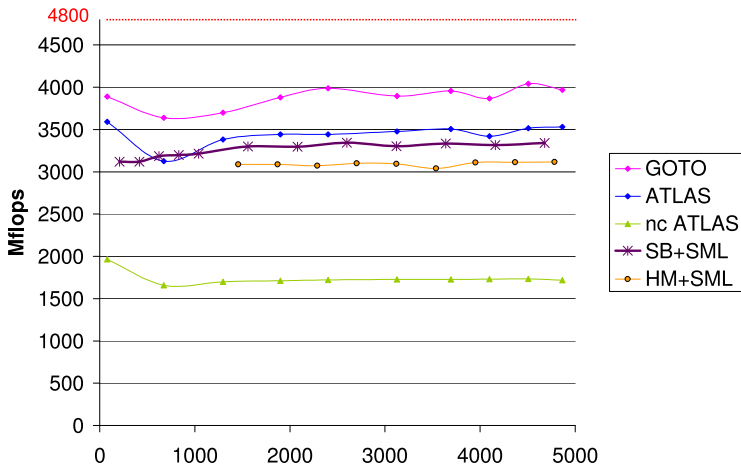
Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Specialized inner
kernels

Results

Ideas

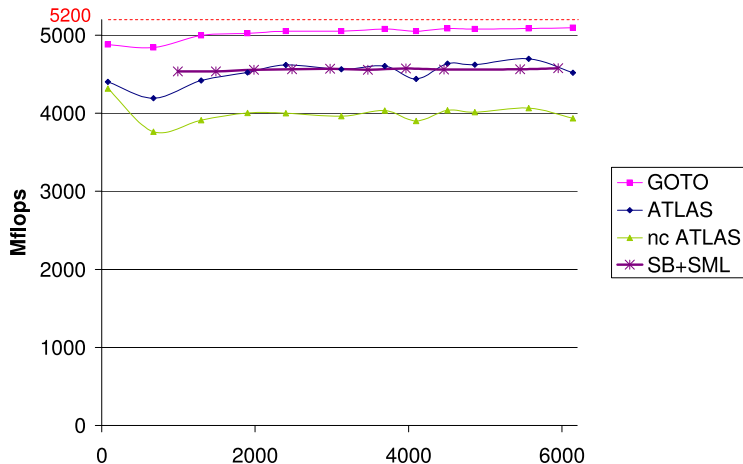


Simple SB storage: $C = C - A^t \times B$

Results on Itanium2

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero



Specialized inner
kernels

Results

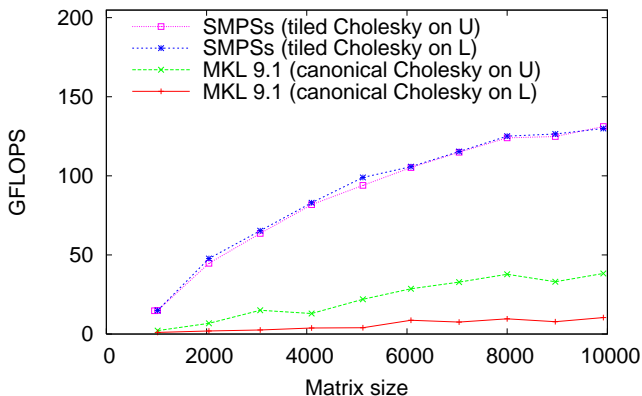
Ideas

Parallel Cholesky: Tiled vs Traditional

Need a flexible way to parallelize code and overlap iterations

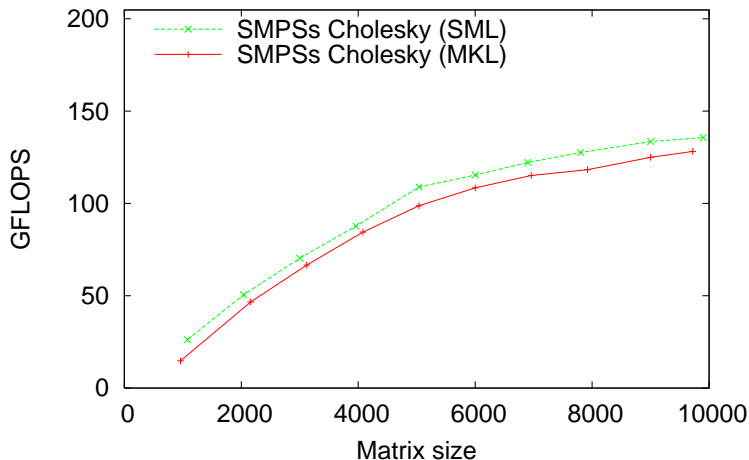
- ▶ Use a runtime system which schedules tasks
- ▶ SMPs: SMP Superscalar (BSC)

Cholesky factorization on 32 Intel Itanium 2 @ 1.6GHz



SMPSs + SML: Performance

Cholesky factorization on 32 Intel Itanium 2 @ 1.6GHz



Tile Size

SMPs and SML on 32 Itanium 2 @ 1.6 GHz

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

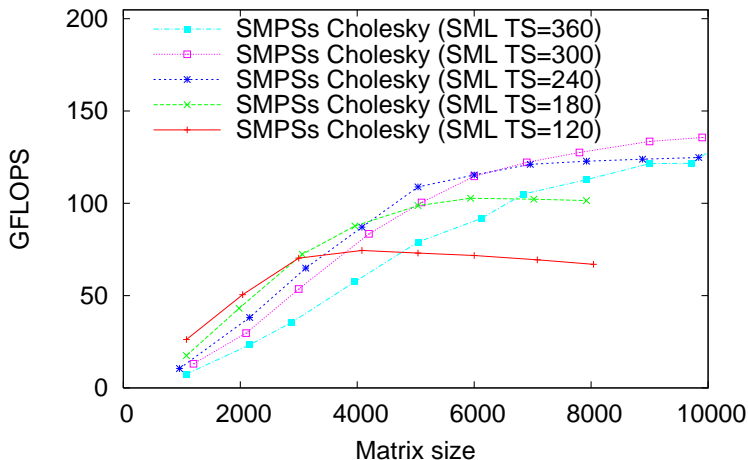
J.R. Herrero

Specialized inner
kernels

Results

Ideas

Cholesky factorization on 32 Intel Itanium 2 @ 1.6GHz



Conclusions

In search for high performance linear algebra codes:

- ▶ Multiple cores \Rightarrow exploit parallelism within the processor
- ▶ Requires efficient operation on small matrices.

Specialization of inner kernels

- ▶ Reduces overhead
- ▶ Exposes simple & regular codes which a compiler can optimize

SMPSs + specialized kernels

- ▶ Can outperform hand-optimized code written in assembler

Outline

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Specialized inner kernels

Results

Ideas

Specialized inner
kernels

Results

Ideas

Ideas for Future Work

- ▶ Improve search / prune exploration space
- ▶ Generate code transformations automatically
- ▶ Identify optimal storage for submatrices automatically
 - ▶ Column-wise vs Row-wise
 - ▶ Dimensions
- ▶ ...

Iterative Compilation for Fast Inner Kernels for Linear Algebra Codes

José-Ramón Herrero

Computer Architecture Department
Universitat Politècnica de Catalunya

Currently on sabbatical leave at
Barcelona Supercomputing Center

E-mail: josepr@ac.upc.edu

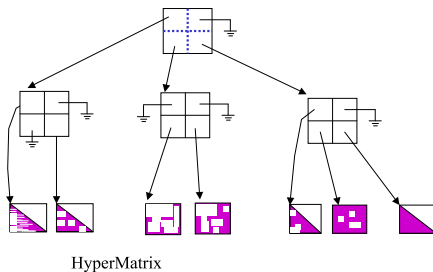
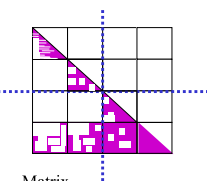
URL: <http://personals.ac.upc.edu/josepr/>

**Adaptive Compilation
(HiPEAK'08 Cluster Meeting)**

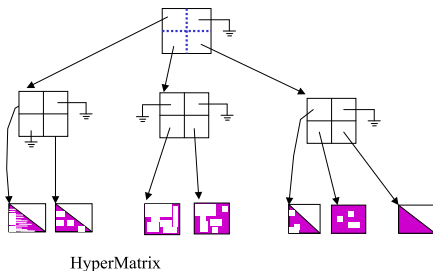
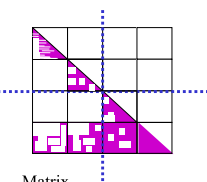
Barcelona, Spain, June 3rd, 2008

Sparse Hypermatrix Cholesky Factorization

Hypermatrix (HM) Structure



Hypermatrix (HM) Structure



Can store 0's within
data submatrices

- ▶ Storage
- ▶ Computation

Trade-off in data sub-
matrix size

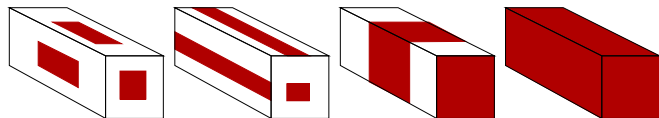
- ▶ BLAS3 efficiency
- ▶ (Useless)
operation on 0's

Reducing Overhead & Increasing Performance

- ▶ Efficient kernels which operate on small data submatrices
- ▶ Bit Vectors associated to data submatrices
- ▶ Windows within data submatrices
- ▶ Amalgamation

Matrix multiplication: efficiency of codes

Our sparse HM Cholesky uses 4 routines:



Less efficient

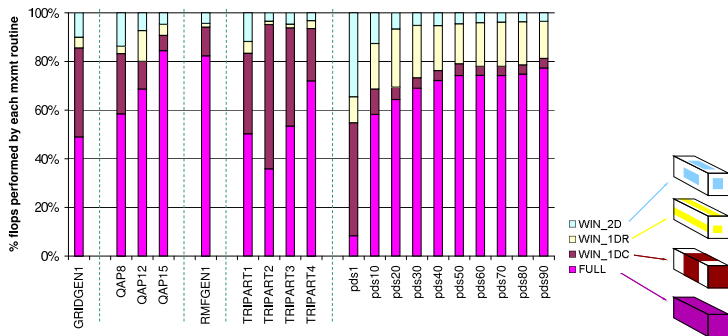
Most efficient

HM flops per $A \times B^T$ subroutine type

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Sparse
Hypermatrix
Cholesky

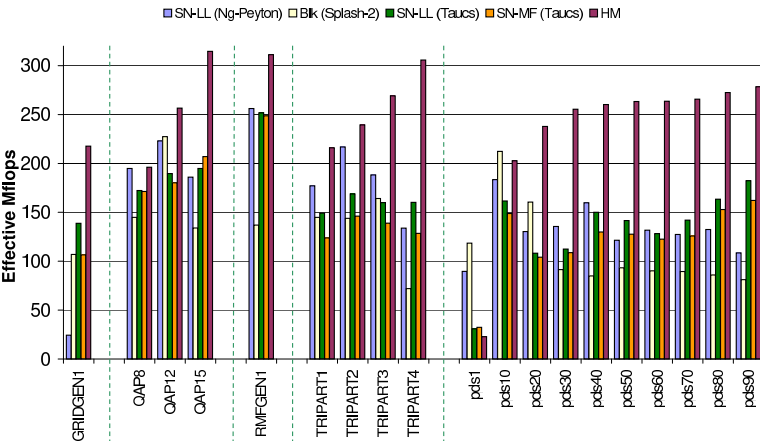


Performance of several sparse Cholesky codes: IPM

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Sparse
Hypermatrix
Cholesky



Sparse HM Cholesky vs WSSMP

Iterative
Compilation for
Fast Inner Kernels
for Linear Algebra
Codes

J.R. Herrero

Sparse
Hypermatrix
Cholesky

