

# Sparse Hypermatrix Cholesky: In Search for High Performance

Josep Ramon Herrero

Computer Architecture Department  
Universitat Politècnica de Catalunya

Talk at the School of Mathematics  
The University of Edinburgh  
May 12th, 2006

Introduction

Sparse  
Hypermatrix  
Cholesky

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Outline

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse Hypermatrix Cholesky Factorization

Compiler-optimized inner kernels

Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Outline

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

## Introduction

Sparse Hypermatrix Cholesky Factorization

Compiler-optimized inner kernels

Conclusions and future work

### Introduction

Sparse  
Hypermatrix  
Cholesky

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Motivation & General Goals

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

## Motivation

- ▶ Matrix computations lie at the heart of many applications.

## General Goal:

- ▶ Obtain efficient implementations of frequent matrix operations.

# Specific Goals

Identify the key points for obtaining high performance.

Obtain efficient implementations . . . . .

- ▶ of some frequent operations:
  - ▶ Sparse Cholesky factorization.
  - ▶ Dense Cholesky factorization.
  - ▶ Dense Matrix Multiplication.
  - ▶ Nearest Neighbor (NN) Classification.
- ▶ On different platforms.

Note:

- ▶ Focus on sequential code.

# Overview

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
  - ▶ 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.

# Outline

Introduction

## Sparse Hypermatrix Cholesky Factorization

SML kernels which operate on small data submatrices

Bit Vectors associated to data submatrices

Windows within data submatrices

Amalgamation

Ordering

Results

Compiler-optimized inner kernels

Conclusions and future work

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Hypermatrix (HM) Structure

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

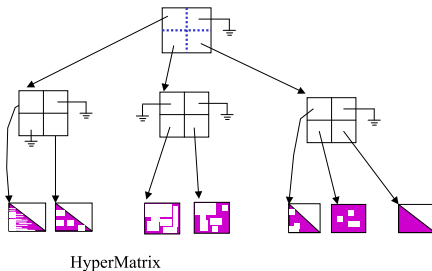
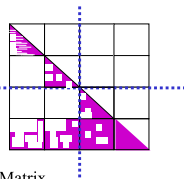
Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



# Hypermatrix (HM) Structure

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

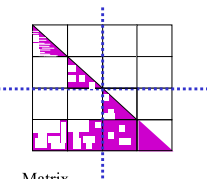
Amalgamation

Ordering

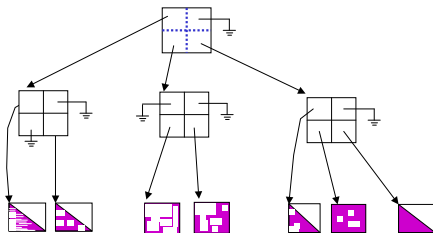
Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



Matrix



HyperMatrix

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

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

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

# Outline

## Introduction

## Sparse Hypermatrix Cholesky Factorization

SML kernels which operate on small data submatrices

Bit Vectors associated to data submatrices

Windows within data submatrices

Amalgamation

Ordering

Results

## Compiler-optimized inner kernels

## Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Hypermatrix Cholesky on problem PDS40

LP problem: Patient Distribution System (40 days)

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

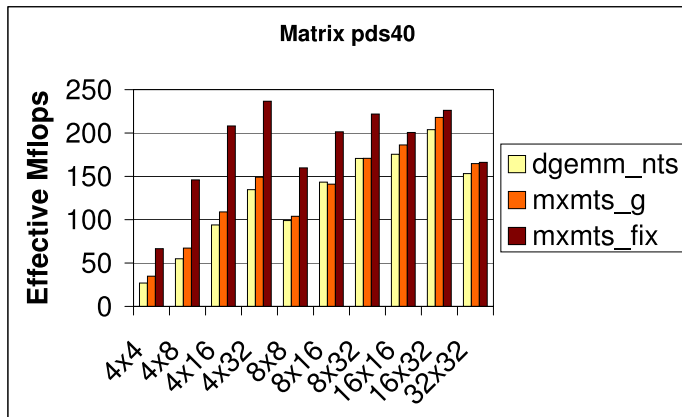
Sparse  
Hypermatrix  
Cholesky

SML routines

- Bit Vectors
- Dense Windows
- Amalgamation
- Ordering
- Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



$$\text{Effective Mflops} = \frac{\# \text{flops (excluding operations on zeros)} \cdot 10^{-6}}{\text{Time (including operations on zeros)}}$$

# Reducing Overhead & Increasing Performance

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Operation on small data submatrices ...  
still has overhead

Goal: reduce the overhead further

- ▶ Bit Vectors associated to data submatrices
- ▶ Windows within data submatrices

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Outline

## Introduction

## Sparse Hypermatrix Cholesky Factorization

SML kernels which operate on small data submatrices

**Bit Vectors** associated to data submatrices

Windows within data submatrices

Amalgamation

Ordering

Results

## Compiler-optimized inner kernels

## Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

**Bit Vectors**

Dense Windows

Amalgamation

Ordering

Results

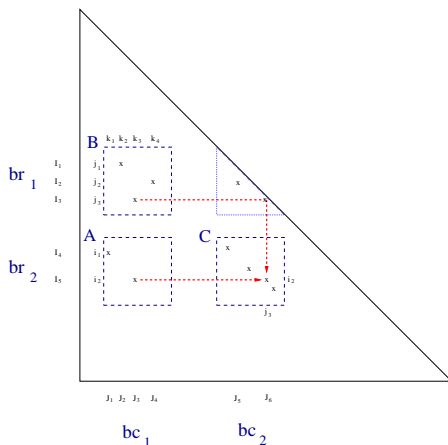
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

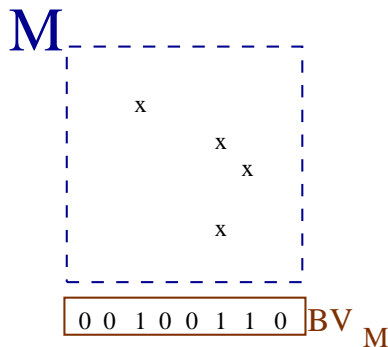
# Bit Vectors: Goal

Reduce unnecessary computation

- ▶ Avoid matrix multiplication when two submatrices produce no update upon a third one



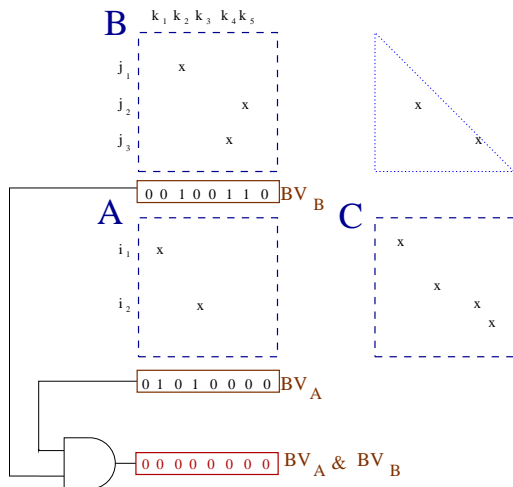
# Bit Vectors: Definition



One bit associated to a column in a data submatrix

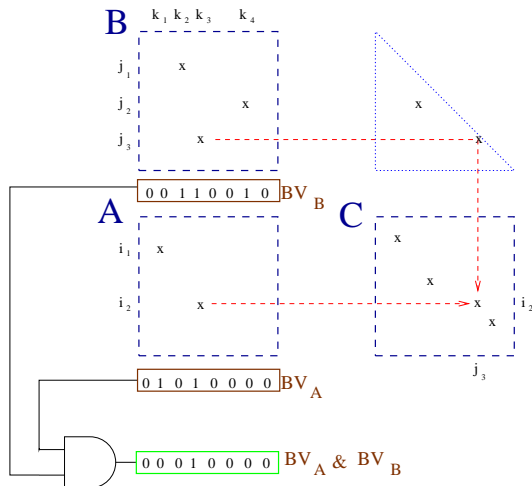
- ▶ Value = 0  $\Leftrightarrow$  column is full of 0's
- ▶ Value = 1  $\Leftrightarrow \exists \geq 1$  NZ in column

# Bit Vectors: Usage (I)



$BV_A \& BV_B = 0$ : Operation can be skipped

# Bit Vectors: Usage (II)



$BV_A \& BV_B \neq 0$ : Operation must be performed

# Outline

## Introduction

## Sparse Hypermatrix Cholesky Factorization

SML kernels which operate on small data submatrices

Bit Vectors associated to data submatrices

**Windows within data submatrices**

Amalgamation

Ordering

Results

## Compiler-optimized inner kernels

## Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

**Dense Windows**

Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Windows within data submatrices: Goal

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Store and use only a part of a data submatrix

- ▶ Reduce unnecessary computation
- ▶ Reduce storage

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

**Dense Windows**

Amalgamation

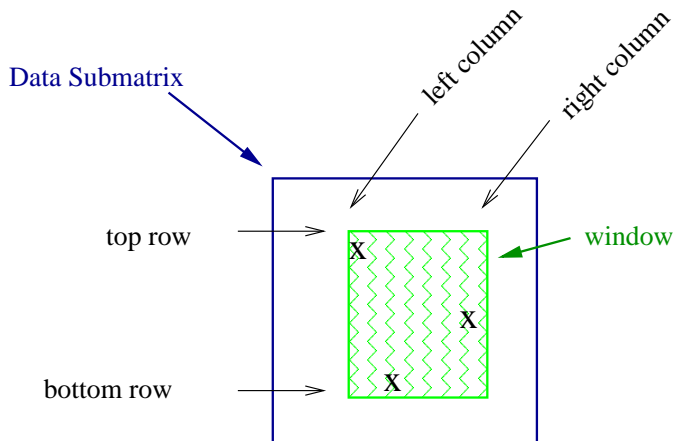
Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Windows: Definition

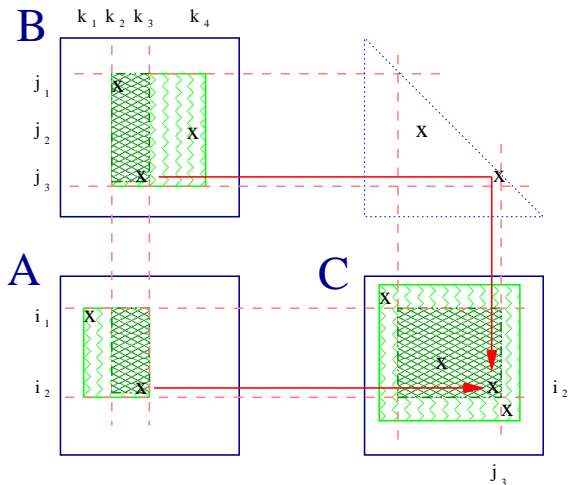


Window: subset of data submatrix

# Windows: Usage (I)

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

**Dense Windows**

Amalgamation

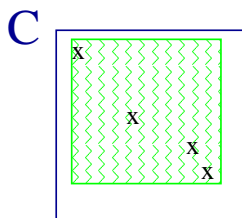
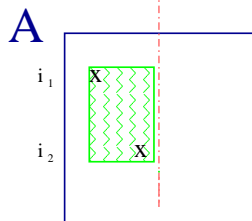
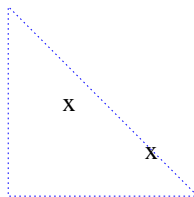
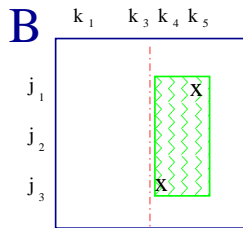
Ordering

Results

Compiler-  
optimized inner  
kernels

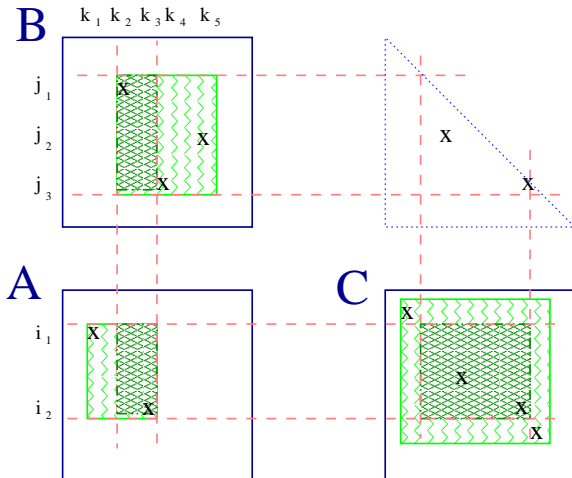
Conclusions and  
future work

# Windows: Usage (II)



Operation can be skipped

# Windows: Usage (III)



Unnecessary operation performed (Could be avoided with BVs)

# Results: Context information

- ▶ MIPS R10000 @ 250 MHz (500 Mflops peak)
- ▶ Sequential code
- ▶ Large problems solved In-Core
- ▶ Ordered using METIS
- ▶ Post-order of Elimination Tree
- ▶ Linear Programming problems [▶ Details](#)
  - ▶ NetLib
  - ▶ Multicommodity Network Flow generators
- ▶ Applications of Finite Element Method [▶ Details](#)
  - ▶ NetLib
  - ▶ PERMAS
  - ▶ PARASOL

# Performance: Block Size vs BVs vs Windows

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

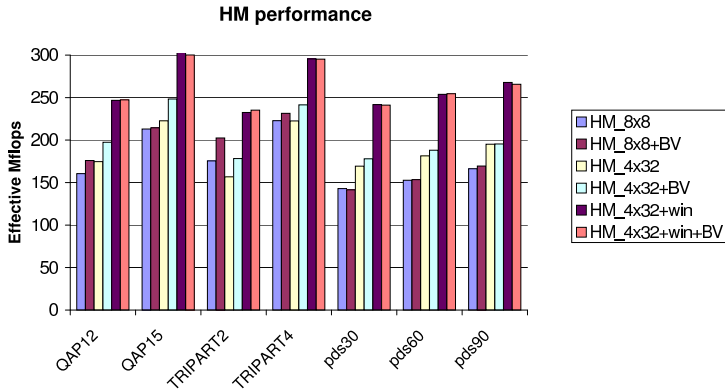
Amalgamation

Ordering

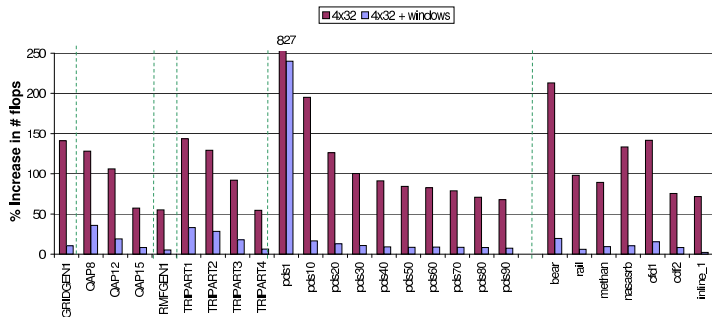
Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



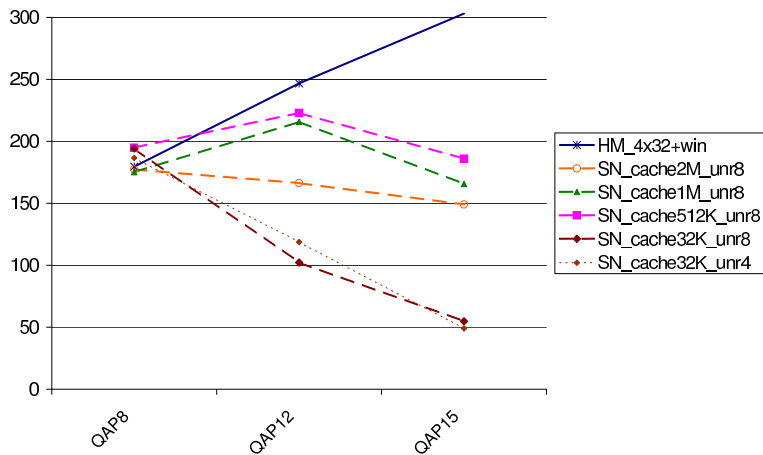
# Increase in number of floating point operations in sparse HM Cholesky w.r.t. the minimum: windows reduce the number of operations on zeros.



# HM vs SN (Ng-Peyton): QAP matrix family

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

Ordering

Results

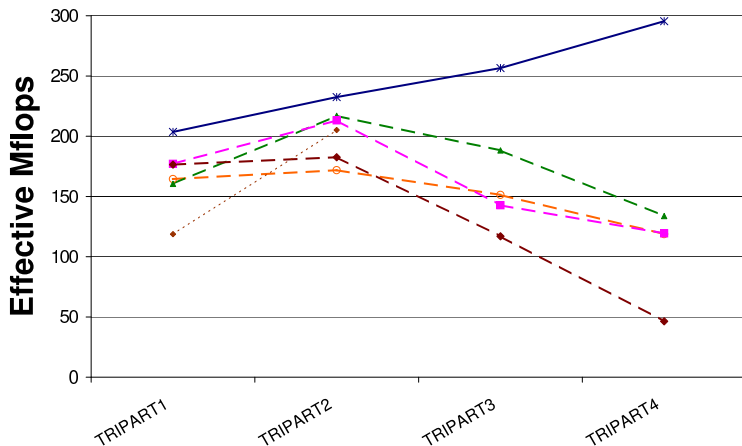
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# HM vs SN: TRIPART matrix family

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

Ordering

Results

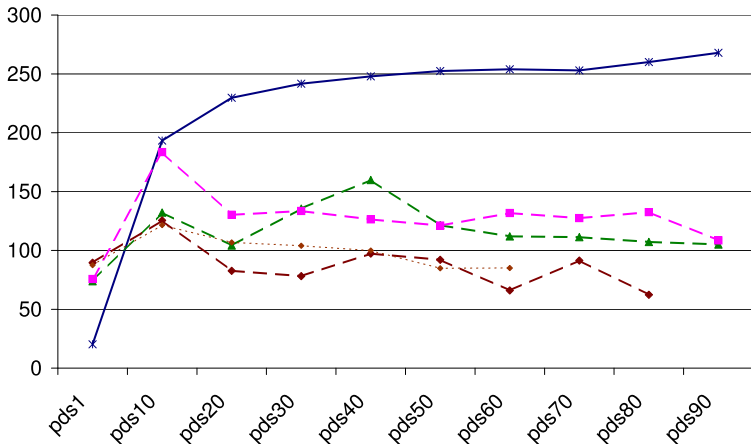
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# HM vs SN: PDS matrix family

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

**Dense Windows**

Amalgamation

Ordering

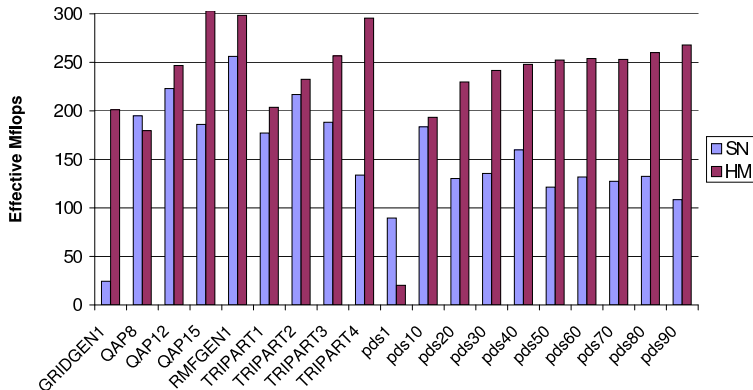
Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# HM vs SN performance: summary

SN vs HM



# Matrix multiplication: efficiency of codes

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

**Dense Windows**

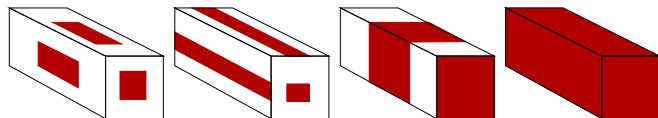
Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



Less efficient

Most efficient

# Outline

## Introduction

## Sparse Hypermatrix Cholesky Factorization

SML kernels which operate on small data submatrices

Bit Vectors associated to data submatrices

Windows within data submatrices

### Amalgamation

Ordering

Results

## Compiler-optimized inner kernels

## Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

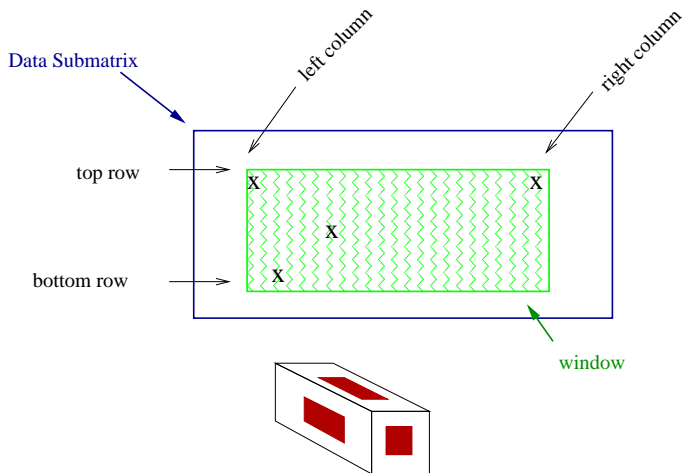
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Intra-Block Amalgamation: Original window

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

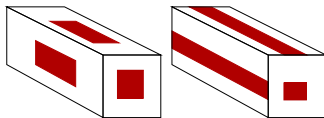
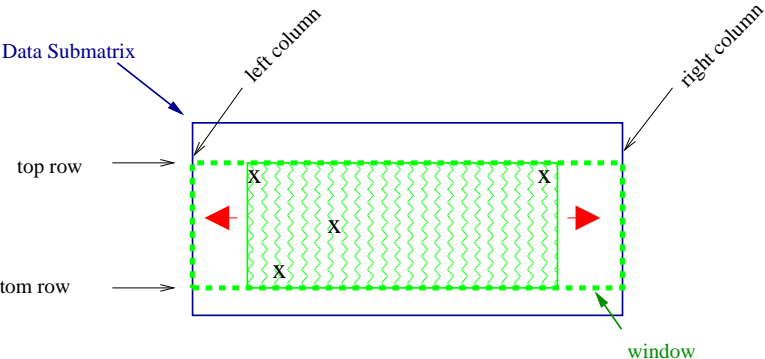
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Intra-Block Amalgamation: column-wise

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

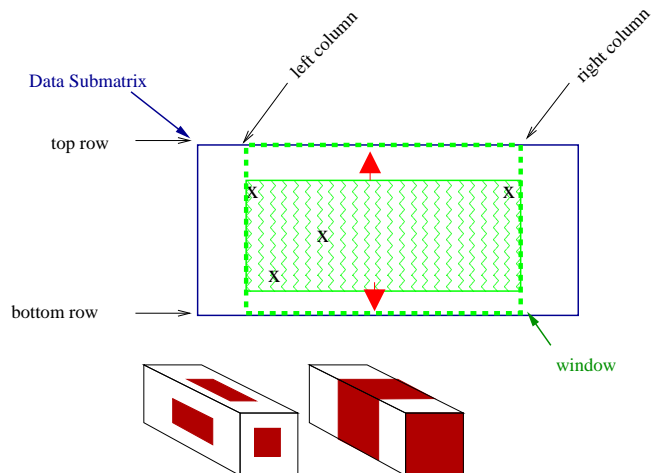
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Intra-Block Amalgamation: row-wise

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

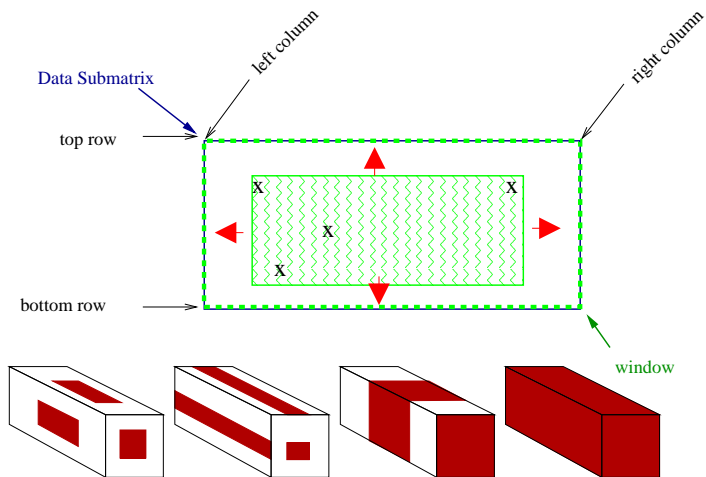
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Intra-Block Amalgamation: row and column-wise

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

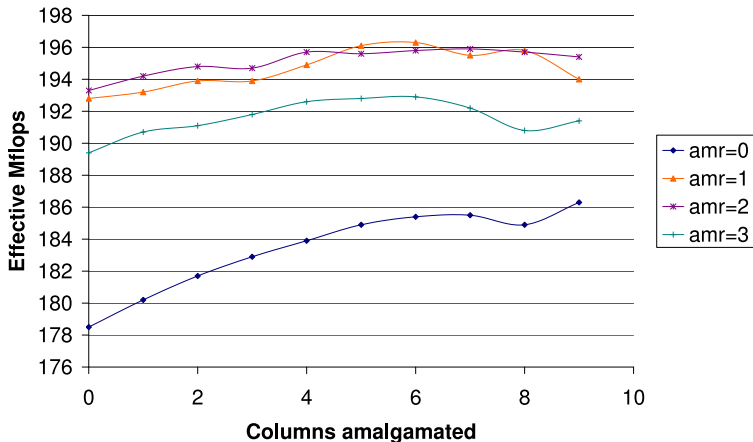
Conclusions and  
future work

# Results: QAP8

## Intra-Block Amalgamation

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

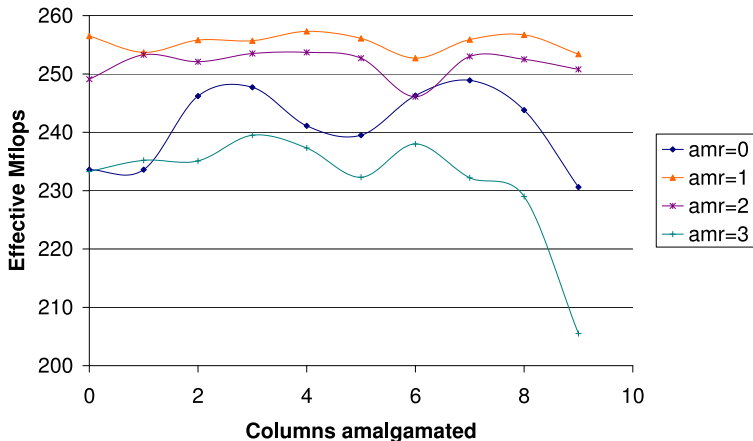
Conclusions and  
future work

# Results: QAP12

## Intra-Block Amalgamation

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

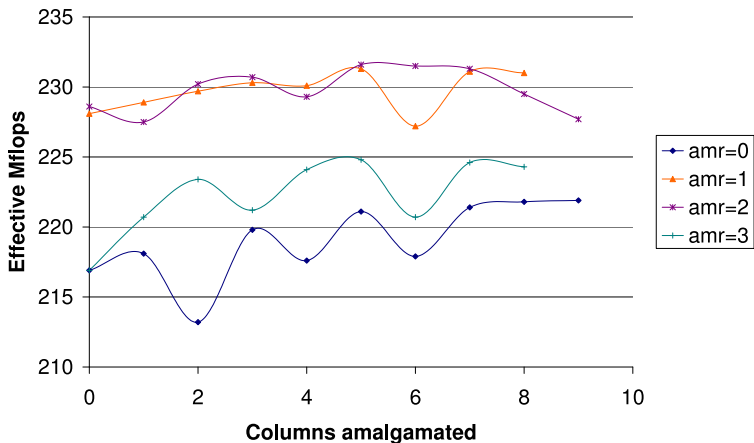
Conclusions and  
future work

# Results: TRIPART1

## Intra-Block Amalgamation

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Results: TRIPART2

## Intra-Block Amalgamation

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

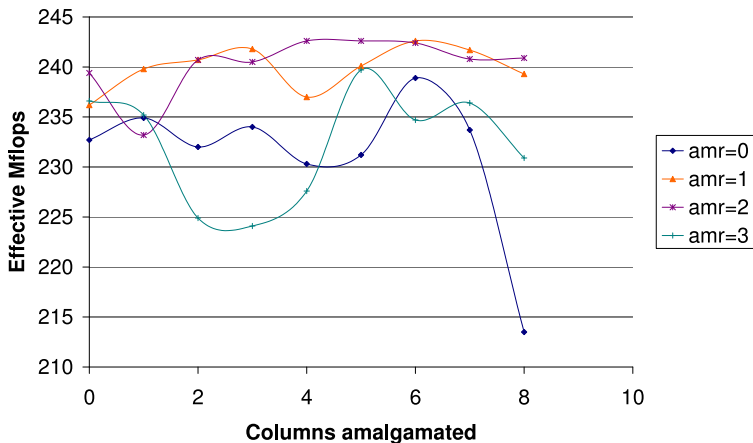
**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

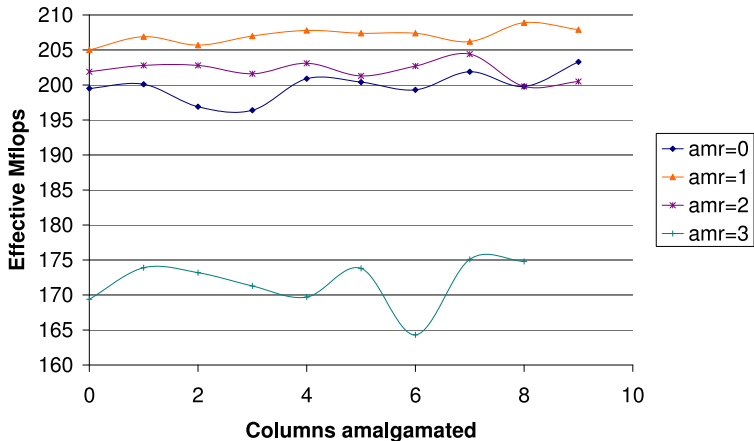


# Results: pds10

## Intra-Block Amalgamation

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

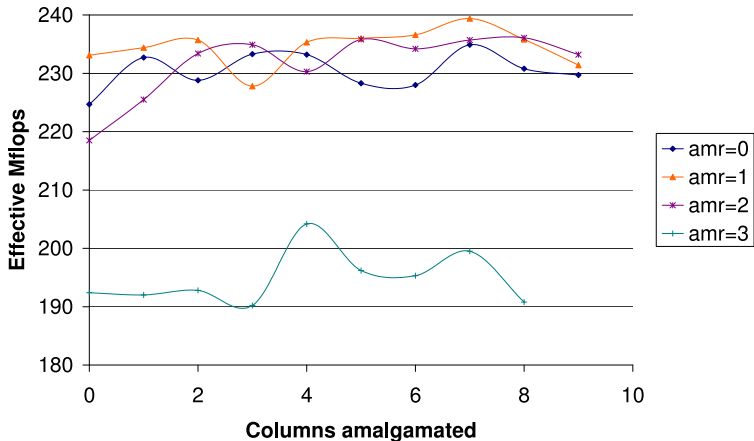
Conclusions and  
future work

# Results: pds20

## Intra-Block Amalgamation

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

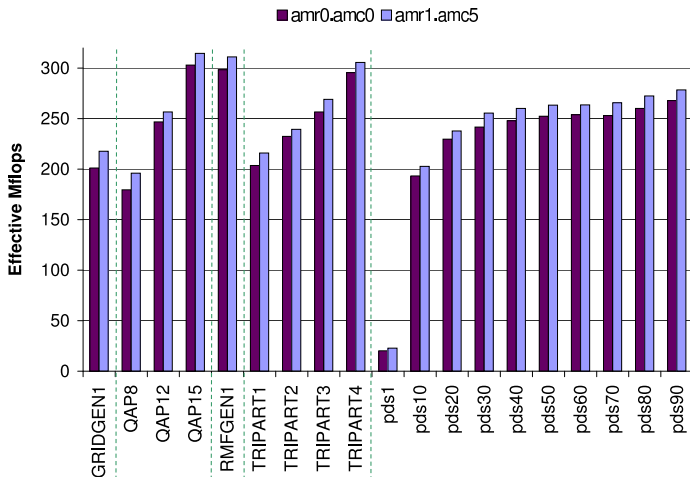
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Results: Original (without amalgamation) vs Intra-block amalgamation

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Intra-Block Amalgamation on Itanium2: Matrix pds20

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

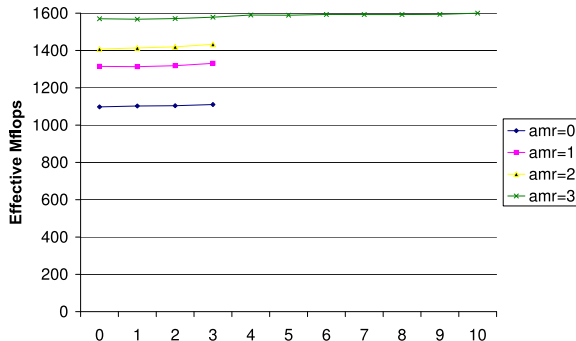
**Amalgamation**

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



# Outline

## Introduction

## Sparse Hypermatrix Cholesky Factorization

SML kernels which operate on small data submatrices

Bit Vectors associated to data submatrices

Windows within data submatrices

Amalgamation

**Ordering**

Results

## Compiler-optimized inner kernels

## Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

**Ordering**

Results

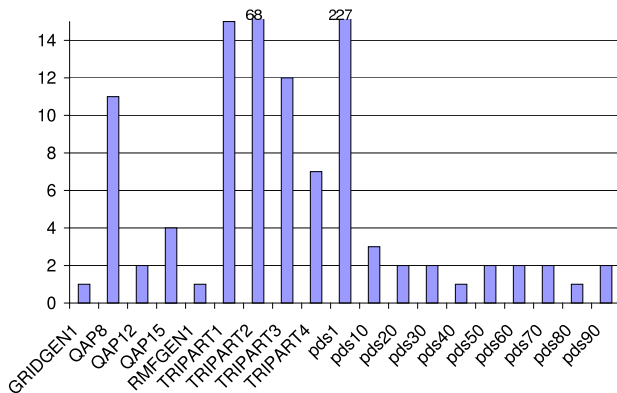
Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Ordering sparse matrices with METIS

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero



Number of iterations necessary to amortize cost of improved ordering.

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

**Ordering**

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Outline

## Introduction

## Sparse Hypermatrix Cholesky Factorization

SML kernels which operate on small data submatrices

Bit Vectors associated to data submatrices

Windows within data submatrices

Amalgamation

Ordering

**Results**

## Compiler-optimized inner kernels

## Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

Amalgamation

Ordering

**Results**

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Performance of several sparse Cholesky codes: IPM

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

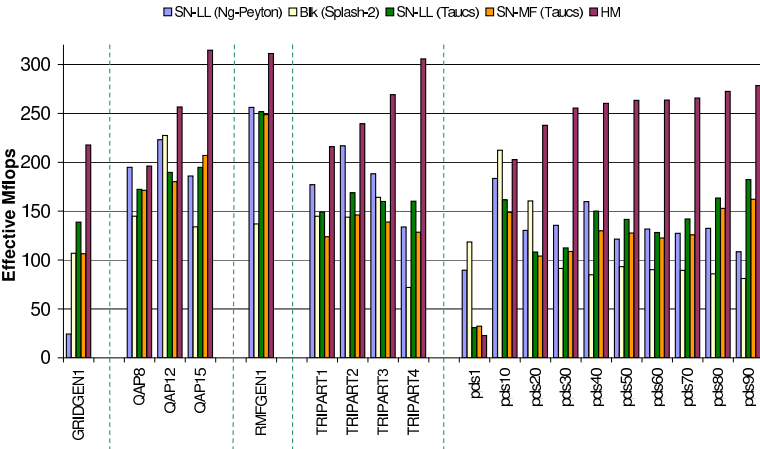
Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



# Performance of several sparse Cholesky codes: FEM

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

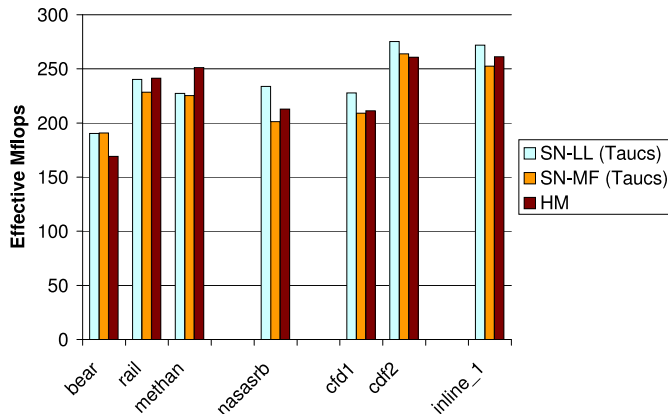
Amalgamation

Ordering

Results

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



► Details

# Overhead in number of operations in sparse HM Cholesky (4x32 + windows)

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse  
Hypermatrix  
Cholesky

SML routines

Bit Vectors

Dense Windows

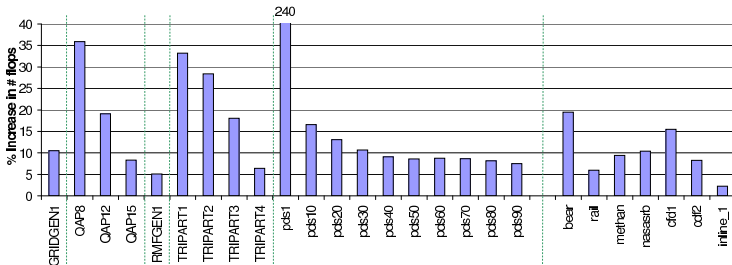
Amalgamation

Ordering

**Results**

Compiler-  
optimized inner  
kernels

Conclusions and  
future work



► Details

# Outline

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse Hypermatrix Cholesky Factorization

**Compiler-optimized inner kernels**

Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

**Compiler-  
optimized inner  
kernels**

Conclusions and  
future work

# Compiler-optimized inner kernels

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

## Facts:

- ▶ Need for high performance inner kernels
- ▶ High cost in creation of such kernels by hand
- ▶ Compiler Optimization is a mature field

## Approach:

- ▶ Smooth the way to the compiler
  - ▶ Creation of a Small Matrix Library (SML)

Introduction

Sparse  
Hypermatrix  
Cholesky

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Generalization

$$C = \beta C + \alpha \text{op}(A) \times \text{op}(B)$$

- ▶  $\alpha$  and  $\beta$  are scalars
- ▶  $\text{op}(A)$  is  $A$  or  $A^t$ .

**Table:** Peak Mflops of inner kernel on a Pentium 4 Xeon Northwood.

	$A \times B^t$	$A^t \times B$
No align	3334	3220
Align	3457	3810

# Efficient Inner Kernels: Conclusions

$C = C + \alpha A^t \times B$  is appealing:

- ▶ access to all three matrices with stride one;
- ▶ stores to matrix C can be hoisted from the inner loop

Compilers can perform efficient optimizations on regular codes. We can facilitate this by:

- ▶ Providing matrix leading dimensions and loop trip counts at compilation time;
- ▶ Trying several variants of code: different loop orders, unroll factors. . .

The resulting code can be more efficient if:

- ▶ Matrices are aligned;
- ▶ All matrices are accessed with stride one;
- ▶ Store operations are removed from the inner kernel.

# Outline

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Introduction

Sparse Hypermatrix Cholesky Factorization

Compiler-optimized inner kernels

Conclusions and future work

Introduction

Sparse  
Hypermatrix  
Cholesky

Compiler-  
optimized inner  
kernels

Conclusions and  
future work

# Conclusions

- ▶ Current compilers can generate very efficient codes when working on simple and **regular codes**.
- ▶ There is a trade-off between the speed of an algorithm and:
  - ▶ computation of non productive operations.
  - ▶ more regular codes
- ▶ Fundamental aspects to achieve high performance:
  - ▶ data accessed with stride one;
  - ▶ data properly aligned;
  - ▶ store operations removed from the innermost loop.

## Sparse hypermatrix Cholesky factorization

- ▶ The most effective overhead reduction techniques have been:
  - ▶ Use of SML routines working on rectangles
  - ▶ Use of dense windows within data submatrices
  - ▶ Use of Intra-Block Amalgamation

# Future Work

- ▶ Improve creation of SML routines using more realistic access patterns;
- ▶ Implement HM Cholesky on matrix  $U$  to increase number of accesses with stride one;
- ▶ Allow for storage of data submatrices as supernodes;
- ▶ Experiment with an Incomplete Cholesky factorization;
- ▶ ...

# Sparse Hypermatrix Cholesky: In Search for High Performance

Josep Ramon Herrero

Computer Architecture Department  
Universitat Politècnica de Catalunya

Talk at the School of Mathematics  
The University of Edinburgh  
May 12th, 2006

# Further Details I

## Further Details

# IPM: Matrix Characteristics

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Appendix  
Further Details

Matrix	Dimension	NZs	NZs in L <sup>a</sup>	Density	Flops to factor <sup>b</sup>
GRIDGEN1	330430	3162757	130586943	0.002	278891
QAP8	912	14864	193228	0.463	63
QAP12	3192	77784	2091706	0.410	2228
QAP15	6330	192405	8755465	0.436	20454
RMFGEN1	28077	151557	6469394	0.016	6323
TRIPART1	4238	80846	1147857	0.127	511
TRIPART2	19781	400229	5917820	0.030	2926
TRIPART3	38881	973881	17806642	0.023	14058
TRIPART4	56869	2407504	76805463	0.047	187168
pds1	1561	12165	37339	0.030	1
pds10	18612	148038	3384640	0.019	2519
pds20	38726	319041	10739539	0.014	13128
pds30	57193	463732	18216426	0.011	26262
pds40	76771	629851	27672127	0.009	43807
pds50	95936	791087	36321636	0.007	61180
pds60	115312	956906	46377926	0.006	81447
pds70	133326	1100254	54795729	0.006	100023
pds80	149558	1216223	64148298	0.005	125002
pds90	164944	1320298	70140993	0.005	138765

▶ <sup>a</sup>Number of non-zeros in factor L (matrix ordered using METIS).

▶ <sup>b</sup>Number of floating point operations (in Millions) necessary to obtain L from the original matrix (ordered with METIS).

# FEM: Matrix Characteristics

Matrix	Dimension	NZs	NZs in L <sup>a</sup>	Density	Flops to factor <sup>b</sup>
bear	25906	412447	3278225	0.009	847
rail	11783	799545	3768886	0.054	1594
methan	48162	1234332	16631801	0.014	10493
nasasrb	54870	1366097	10489476	0.007	3496
cf1	70656	949510	20910296	0.008	13523
cf2	123440	1605669	37696869	0.005	31218
inline_1	503712	18660027	174608135	0.001	150974

<sup>a</sup>Number of non-zeros in factor L (matrix ordered using METIS).

<sup>b</sup>Number of floating point operations (in Millions) necessary to obtain L from the original matrix (ordered with METIS).

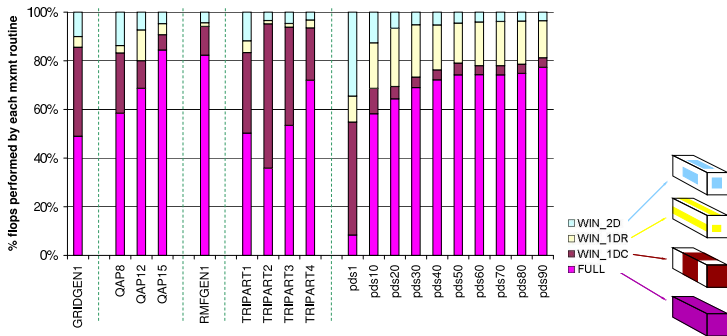
# HM flops per MxMt subroutine type

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Appendix

Further Details



# Calls, flops & time per $A \times B^T$ subroutine type

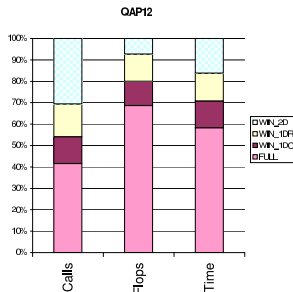
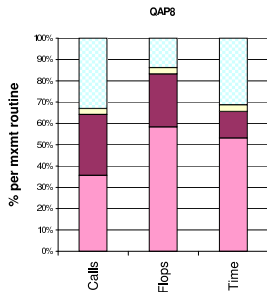
QAP8 and QAP12

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Appendix

Further Details



► IPM matrices

# Sparse HM Cholesky: flops per $M \times M^t$ subroutine type

IPM & FEM

Sparse  
Hypermatrix  
Cholesky:  
In Search for High  
Performance

J.R. Herrero

Appendix

Further Details

