Using nonlinear array layouts in dense matrix operations

Josep Ramon Herrero & Juan J. Navarro

Computer Architecture Department
Universitat Politècnica de Catalunya

PARA’06
June 20th, 2006
Outline

Introduction: A bottom-up approach

Operation on dense matrices: Nonlinear array layouts

Conclusions
Outline

Introduction: A bottom-up approach

Operation on dense matrices: Nonlinear array layouts

Conclusions
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.
Inner $C = C - A \times B^T$ kernel
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.

<table>
<thead>
<tr>
<th></th>
<th>( A \times B^t )</th>
<th>( A^t \times B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>No align</td>
<td>3334</td>
<td>3220</td>
</tr>
<tr>
<td>Align</td>
<td>3457</td>
<td>3810</td>
</tr>
</tbody>
</table>
Inner kernels: Conclusions

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.

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

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

Different optimal block sizes for different processors
A bottom-up approach

First

- Produce inner kernel and determine best block size
Using nonlinear array layouts in dense matrix operations

J.R. Herrero

Introduction: A bottom-up approach

Operation on dense matrices

Conclusions

A bottom-up approach

Then

- Create structure based on best block size

First

- Produce inner kernel and determine best block size
Outline

Introduction: A bottom-up approach

Operation on dense matrices: Nonlinear array layouts
  Operation on dense matrices: Hypermatrix Storage
  Operation on dense matrices: Square Block Storage

Conclusions
Using nonlinear array layouts in dense matrix operations

J.R. Herrero

Outline

Introduction: A bottom-up approach

Operation on dense matrices: Nonlinear array layouts
  Operation on dense matrices: Hypermatrix Storage
  Operation on dense matrices: Square Block Storage

Conclusions
Hypermatrix (HM) Structure
Using nonlinear array layouts in dense matrix operations

J.R. Herrero

Introduction: A bottom-up approach

Operation on dense matrices

HM Storage
SB Storage

Conclusions

Results: MIPS R10000
Dense Cholesky and Matrix Multiplication
Orthogonal Blocks

Constructed so that the directions of the blocks of adjacent levels are different.
Results: Size 4507 on Itanium2
HM matrix multiplication using several loop orders

HM Storage
SB Storage

Conclusions
Using nonlinear array layouts in dense matrix operations

J.R. Herrero

Introduction: A bottom-up approach

Operation on dense matrices

HM Storage

SB Storage

Conclusions

Results: Itanium2
Dense Cholesky and Matrix Multiplication
Introduction: A bottom-up approach

Operation on dense matrices: Nonlinear array layouts
Operation on dense matrices: Hypermatrix Storage
Operation on dense matrices: Square Block Storage

Conclusions
Simple SB storage: matrices aligned and stored by submatrices.

m

mtofree

blkszy

dimx

blkszx

dimy

m

Operation on dense matrices

HM Storage

SB Storage

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

Results on Power4
Simple SB storage: $C = C - A^t \times B$

Results on Pentium4
Simple SB storage: $C = C - A^t \times B$

Results on Itanium2

![Graph showing results on Itanium2 for various storage methods.

Legend:
- GOTO
- ATLAS
- nc ATLAS
- SB+SML

Introduction: A bottom-up approach

Operation on dense matrices

HM Storage

SB Storage

Conclusions
Outline

Introduction: A bottom-up approach

Operation on dense matrices: Nonlinear array layouts

Conclusions
Conclusions

- Efficiency of inner kernel is of paramount importance.
  - Different optimal block sizes for different processors
  - Fundamental aspects to achieve high performance:
    - data accessed with stride one;
    - data properly aligned;
    - store operations removed from the innermost loop.

- Iterative+SB outperforms Recursive+HM
- Iterative+SB+SML provides competitive performance
Using nonlinear array layouts in dense matrix operations

Josep Ramon Herrero & Juan J. Navarro

Computer Architecture Department
Universitat Politècnica de Catalunya

PARA’06
June 20th, 2006
Results: Alpha 21264A
Compiler-optimized inner kernels
Compiler-optimized inner kernels

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
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...
Current compilers can generate very efficient codes when working on simple and regular codes.

Generation of efficient compiler-optimized inner kernels for different:
- Matrix sizes
- Platforms
Using nonlinear array layouts in dense matrix operations

J.R. Herrero

Compiler-optimized inner kernels

**Inner** \( C = C - A \times B^T \) kernel

**Alpha 21264A (ev67) @ 731 MHz**

**Itanium2 @ 1.3 GHz**