Exposing Inner Kernels and Nonlinear Storage for Fast Dense 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/

9th Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA’08)

Trondheim, Norway, May. 14th, 2008
Outline

Introduction

Specialized inner kernels

Results

Conclusions
Outline

Introduction

Specialized inner kernels

Results

Conclusions
Overview

In search for high performance dense linear algebra codes:

- Multiple cores ⇒ exploit parallelism within the processor
- Requires efficient operation on small matrices.
SBPF Cholesky

Cholesky factorization with Square Blocked Lower Packed Format.

Based on subroutine DPSTRF as appears in Fig.10, page 44 in:

Overview

Need a flexible way to parallelize code and overlap iterations

- Use a runtime system which schedules tasks
- SMPSs: SMP Superscalar
  (Barcelona Supercomputing Center)
Parallel Cholesky: Tiled vs Traditional
SMPSs and MKL 9.1 on 32 Itanium 2 @ 1.6 GHz

Cholesky factorization on 32 Intel Itanium 2 @ 1.6GHz

GFLOPS

Matrix size

SMPSs (tiled Cholesky on U)
SMPSs (tiled Cholesky on L)
MKL 9.1 (canonical Cholesky on U)
MKL 9.1 (canonical Cholesky on L)
Overhead

In search for high performance portability:

- Linear Algebra codes call BLAS

Within BLAS:

- Parameter checking
  - Robustness
- Data copies
  - Exploit locality & ease data streaming
- Repeated in every call ⇒ Overhead
DGEMM vs DPOTRF vs DPSTRF
Using ATLAS (Itanium 2 @ 1.5 GHz)
DPOTRF vs DPSTRF
Using Goto's library (Itanium 2 @ 1.5 GHz)
Outline

Introduction

Specialized inner kernels

Results

Conclusions
Compiler-optimized inner kernels

Fact:

- 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.
Creation of inner kernels

Approach:

- Profiling
  - Detect and optimize parts of code which take up most computation time.

- Specialization
  - Simplify code to do only what is strictly necessary.

- Compiler-optimized inner kernel for operation on small matrices
  - Small Matrix Library (SML)

Use SML routines for general codes.

- Bottom-up approach
  - 2nd: store matrix according to the underlying block size
  - 1st: determine adequate block size
Practical application

Application to routine DPSTRF on an Itanium 2

- **Profiling:** Focus on . . .
  - 1st: Matrix Multiplication
  - 2nd: SYRK & TRSM
  - Finally: POTRF

- **Specialization:** e.g. $A^T \times B$

- Create optimized matrix multiplication kernel
  - Use compiler to generate optimized object code
  - $\Rightarrow$ Chose best block size
  - Insert best code in library: Small Matrix Library (SML).

- Create optimized kernels for SYRK, TRSM and POTRF
Outline

Introduction

Specialized inner kernels

Results

Conclusions
Performance

Cholesky factorization on 32 Intel Itanium 2 @ 1.6GHz

GFLOPS

Matrix size

SMPSs Cholesky (SML)
SMPSs Cholesky (MKL)
Tile Size
SMPSs and MKL 9.1 on 32 Itanium 2 @ 1.6 GHz

Cholesky factorization on 32 Intel Itanium 2 @ 1.6GHz

GFLOPS vs. Matrix size graph showing performance comparison for different tile sizes.
Exposing Inner Kernels and Nonlinear Storage for Fast Dense Linear Algebra Codes

J.R. Herrero

Introduction
Specialized inner kernels
Results
Conclusions

Tile Size
SMPSs and SML on 32 Itanium 2 @ 1.6 GHz

Cholesky factorization on 32 Intel Itanium 2 @ 1.6GHz

GFLOPS
Matrix size
Scalability and Tile Size
SMPSs and MKL on 32 Itanium 2 @ 1.6 GHz

Cholesky factorization of a 6000 x 6000 matrix on 32 Intel Itanium 2 @ 1.6GHz
Exposing Inner Kernels and Nonlinear Storage for Fast Dense Linear Algebra Codes

J.R. Herrero

Introduction

Specialized inner kernels

Results

Conclusions

Scalability and Tile Size
SMPSs and SML on 32 Itanium 2 @ 1.6 GHz

Cholesky factorization of a 6000 x 6000 matrix on 32 Intel Itanium 2 @ 1.6GHz

GFLOPS vs. Number of CPUs for different tile sizes and SMPSs/SMALLs configurations.
Outline

Introduction

Specialized inner kernels

Results

Conclusions
Conclusions

Parallelization with SMPSs
► Quick and intuitive
► Efficient

Specialization of inner kernels
► Reduces overhead
► Exposes simple & regular codes which a compiler can optimize

SBPF + SMPSs + specialized kernels
► Reduced storage
► Can outperform hand-optimized code written in assembler
Acknowledgements

Thanks to...

- Ministerio de Educación y Ciencia of Spain for funding this project (Grant TIN2007-60625).

- Barcelona Supercomputing Center (BSC) for providing some of the resources used in this work.
Exposing Inner Kernels and Nonlinear Storage for Fast Dense 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/

9th Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA’08)

Trondheim, Norway, May. 14th, 2008