Dense Triangular Solvers on Multicore Clusters using UPC

Jorge González-Domínguez*, María J. Martín,
Guillermo L. Taboada, Juan Touriño

Computer Architecture Group
University of A Coruña (Spain)
{jgonzalezd,mariam,taboada,juan}@udc.es

International Conference on Computational Science
ICCS 2011

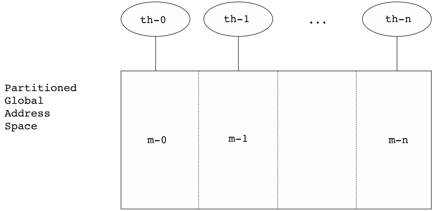
UPC: a Suitable Alternative for HPC in Multi-core Era

Programming Models:

- Traditionally: Shared/Distributed memory programming models
- Challenge: hybrid memory architectures
  - PGAS (Partitioned Global Address Space)

PGAS Languages:

- UPC -> C
- Titanium -> Java
- Co-Array Fortran -> Fortran



UPC Compilers:

- Berkeley UPC
- GCC (Intrepid)
- Michigan TU
- HP, Cray and IBM UPC Compilers

Studied Numerical Operations

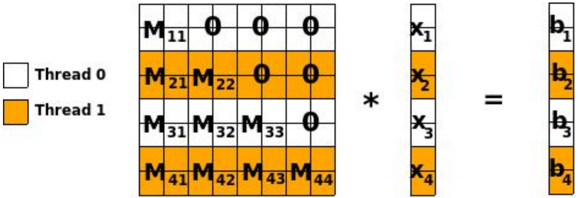
BLAS Libraries

- Basic Linear Algebra Subprograms
- Specification of a set of numerical functions
- Widely used by scientists and engineers
- SparseBLAS and PBLAS (Parallel BLAS)
- Development of UPCBLAS
  - *gemv*: Matrix-vector product ($\alpha * A * x + \beta * y = y$)
  - *gemm*: Matrix-matrix product ($\alpha * A * B + \beta * C = C$)

Studied Routines

- *trsv*: BLAS2 Triangular Solver ($M * x = b$)
- *trsm*: BLAS3 Triangular Solver ($M * X = B$)

Parallel BLAS2 Triangular Solver($M * x = b$) (I)



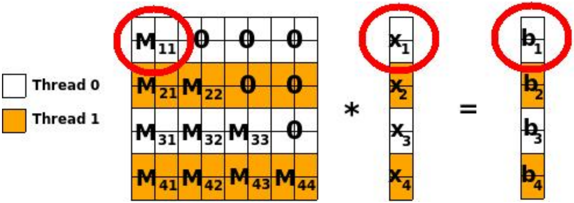
Example

- Matrix 8X8
- 2 Threads
- 2 Rows per block

Types of Blocks

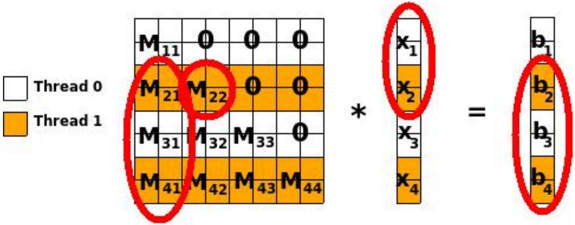
- $i < j$ Zero matrix
- $i = j$ Triangular matrix
- $i > j$ Square matrix

Parallel BLAS2 Triangular Solver($M * x = b$) (II)



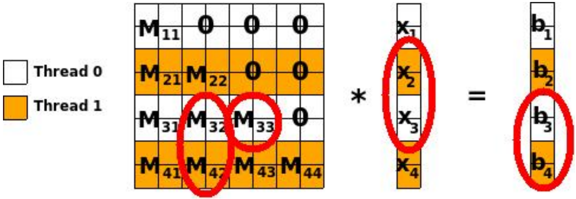
THREAD 0 $\rightarrow$ trsv(M_{11},x_1,b_1)

Parallel BLAS2 Triangular Solver($M * x = b$) (III)



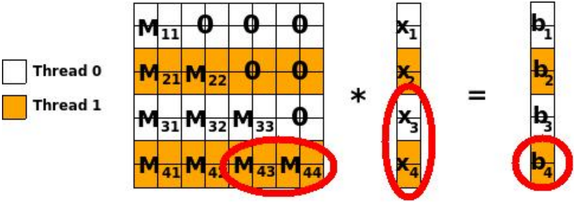
THREAD 0 → gemv(M_{31},x_1,b_3)
THREAD 1 → gemv(M_{21},x_1,b_2)
→ trsv(M_{22},x_2,b_2)
→ gemv(M_{41},x_1,b_4)

Parallel BLAS2 Triangular Solver($M * x = b$) (IV)



$$\text{THREAD 0} \rightarrow \text{gemv}(M_{32}, x_2, b_3)$$
$$\rightarrow \text{trsv}(M_{33}, x_3, b_3)$$
$$\text{THREAD 1} \rightarrow \text{gemv}(M_{42}, x_2, b_2)$$

Parallel BLAS2 Triangular Solver($M * x = b$) (V)



THREAD 1 → gemv(M_{43},x_3,b_4)
→ trsv(M_{44},x_4,b_4)

Parallel BLAS2 Triangular Solver($M * x = b$) (and VI)

Impact of the Block Size

The more blocks the matrix is divided in, the more ...

- computations can be simultaneously performed ($\uparrow$ perf)
- synchronizations are needed ($\downarrow$ perf)

Best block size automatically determined -> Paper

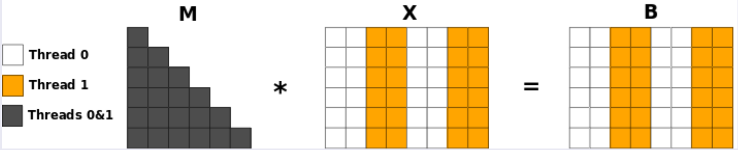
Parallel BLAS3 Triangular Solver($M * X = B$) (I)

Studied Distributions

- Triangular and dense matrices distributed by rows -> Similar approach than BLAS2 but changing sequential
  - *gemv* $\longrightarrow$ *gemm*
  - *trsv* $\longrightarrow$ *trsm*
- Dense matrices distributed by columns
- Triangular and dense matrices with 2D distribution (multicore-aware)

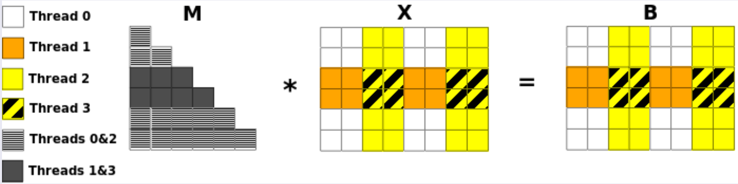
Parallel BLAS3 Triangular Solver($M * X = B$) (II)

Dense Matrices Distributed by Columns



Parallel BLAS3 Triangular Solver($M * X = B$) (and III)

Triangular and dense matrices with 2D distribution (multicore-aware)

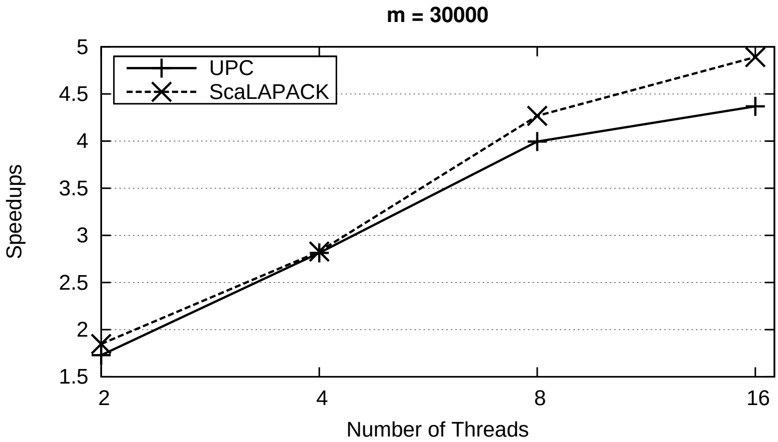


- Node 1 -> Cores 0 & 1
- Node 2 -> Cores 2 & 3

1. Introduction
2. BLAS2 Triangular Solver
3. BLAS3 Triangular Solver
4. **Experimental Evaluation**
5. Conclusions

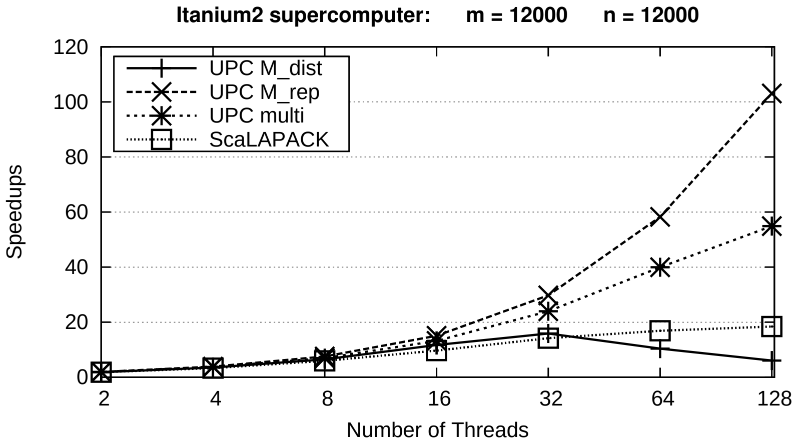
Evaluation of the BLAS2 Triangular Solver

Departamental Cluster; InfiniBand; 8 nodes; 2th/node



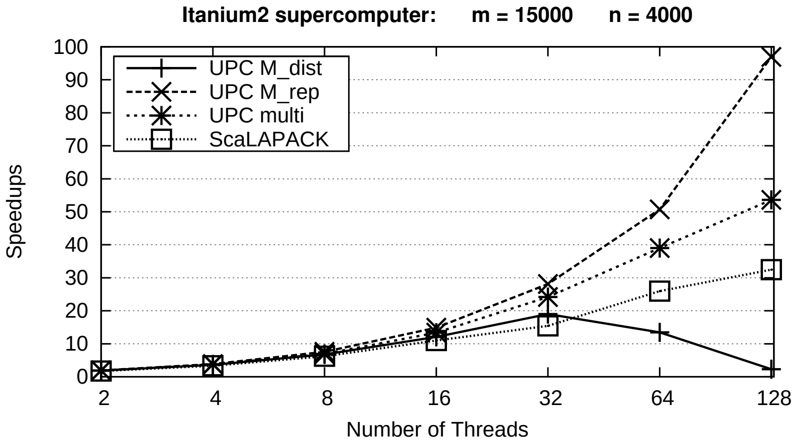
Evaluation of the BLAS3 Triangular Solver (I)

Finis Terrae Supercomputer; InfiniBand; 32 nodes; 4th/node



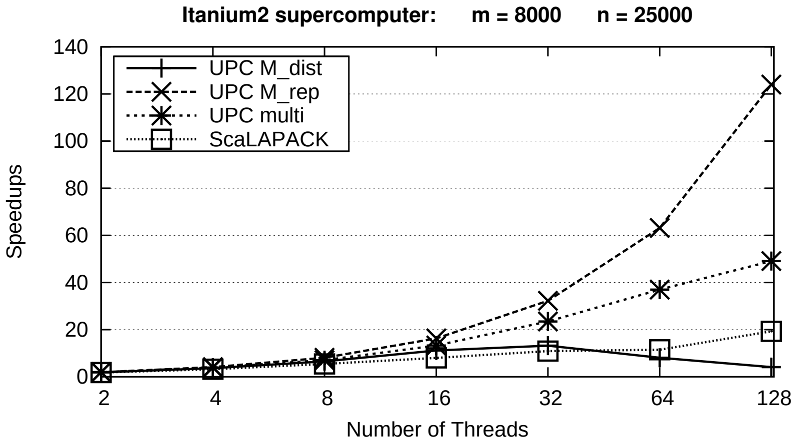
Evaluation of the BLAS3 Triangular Solver (II)

Finis Terrae Supercomputer; InfiniBand; 32 nodes; 4th/node



Evaluation of the BLAS3 Triangular Solver (and III)

Finis Terrae Supercomputer; InfiniBand; 32 nodes; 4th/node



Main Conclusions

Summary

- Implementation of BLAS triangular solvers for UPC
- Several techniques to improve their performance
- Special effort to find the most appropriate data distributions
    - BLAS2 → Block-cyclic distribution by rows
        - Block size automatically determined according to the characteristics of the scenario
    - BLAS3 → Depending on the memory constraints
- Comparison with ScaLAPACK (MPI)
    - UPC easier to use
    - Similar of better performance

Dense Triangular Solvers on Multicore Clusters using UPC

Jorge González-Domínguez*, María J. Martín,
Guillermo L. Taboada, Juan Touriño

Computer Architecture Group
University of A Coruña (Spain)
{jgonzalezd,mariam,taboada,juan}@udc.es

International Conference on Computational Science
ICCS 2011