The 8th International Conference on
High Performance Computing in Asia-Pacific Region
(HPC Asia 2005)

LAPACK in SILC: Use of a Flexible Application Framework for Matrix Computation Libraries

Tamito KAJIYAMA, Akira NUKADA (JST)
Hidehiko HASEGAWA (University of Tsukuba)
Reiji SUDA, Akira NISHIDA (University of Tokyo)

Outline

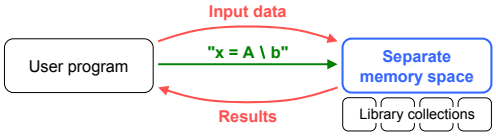
- Background
  - Ways of using matrix computation libraries
- Proposal of SILC
  - Simple Interface for Library Collections
- Use of LAPACK in SILC
- Experiments
- Summary and future work

Background

- Matrix computation libraries
- Source-level dependency on particular libraries
  - Resulting from direct use of library-specific APIs
  - You need to modify user programs when…
    - Porting them to other computing environments (using environment-specific special libraries)
    - Using alternative solvers, matrix storage formats, and precisions
  - Both portability of user programs and utility of libraries are limited

SILC: Simple Interface for Library Collections

- Basic ideas
  - **Data transfer** and **a request for computation**
  - **Mathematical expressions** for the request
  - **A separate memory space** for the computation



Solving a system of linear equations $Ax=b$

- In the traditional way (using LAPACK in C)

```
double *A, *b;
int kl, ku, lda, ldb, nrhs, info, *ipiv;

dgbtrf (N, N, kl, ku, A, lda, ipiv, &info);  /* LU factorization */
if (info == 0)
    dgbtrs ('N', N, kl, ku, nrhs, A, lda, ipiv, b, ldb, &info);  /* solve */
```

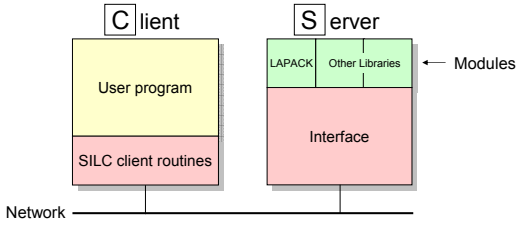
- In SILC

```
silc_envelope_t A, b, x;

SILC_PUT ("A", &A);
SILC_PUT ("b", &b);
SILC_EXEC ("x = A \ b");  /* call a solver (e.g., dgbtrf & dgbtrs) */
SILC_GET (&x, "x");
```

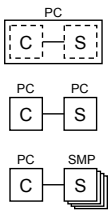
Design and implementation of SILC

- Based on a client-server architecture
- Client (sequential user program)
- SILC server (OpenMP-based parallel program)



Main benefits of using SILC

- Source-level independence between user programs and matrix computation libraries
  - Easy access to alternative solvers and matrix storage formats, possibly in other libraries
  - Instant porting to other computing environments without any modification in user programs
- You need to prepare only the smallest amount of data
  - Temporary buffers are automatically allocated
- Language-independent mathematical expressions
  - Applicable in many programming languages (C, Fortran, Python)



Functionalities

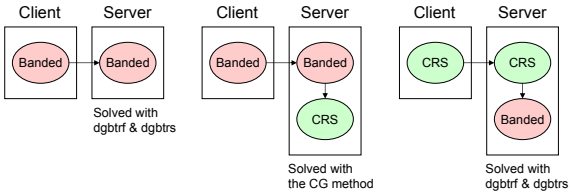
- Data structures
  - Data types: scalar, vector, matrix, cubic array
  - Precisions: integer, real, complex (single or double)
  - Matrix storage formats: dense, banded, CRS
- Mathematical expressions
  - Binary arithmetic operators (+, −, *, /, %)
  - Solutions of systems of linear equations (A \ b)
  - Conjugate transposes (A'), complex conjugates (A~)
  - Built-in functions
    - Ex. "sqrt(b' * b)" is the 2-norm of vector b
  - Subscript
    - Ex. "A[1:5,1:5]" is a 5×5 submatrix of A

Modes for using LAPACK in SILC

Mode (A)
- Both data transfer and computation with **LAPACK**

Mode (B)
- Data transfer with **LAPACK**
- Computation with **another library**

Mode (C)
- Data transfer with **another library**
- Computation with **LAPACK**

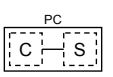


Benefits of using LAPACK in SILC

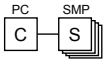
- Easy to use different solvers and matrix storage formats (possibly in other libraries)
  - The SILC server is free to change solvers and matrix storage formats
- The same mathematical expression "x = A \ b" for all precisions and matrix storage formats
  - The dgbtrf/dgbtrs pair is used only for band matrices of double precision
- Independent of vendor-specific C interfaces

Experiments

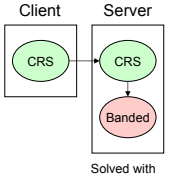
- Solution of systems of linear equations $Ax = b$
  - 5-point difference Laplacian on a uniform 2D grid
  - A is a sparse symmetric matrix (dimension 40,000)
- Program #1
  - Using LAPACK in Mode (C) with the request "X = band(A) \ B"
  - Tested with 2 different servers



Local SILC server



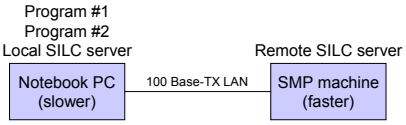
Remote SILC server



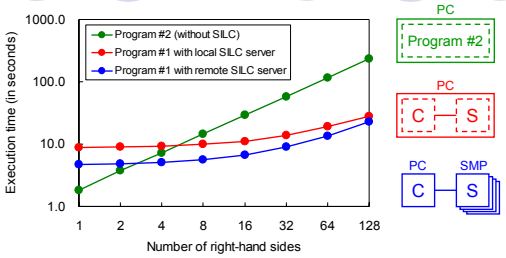
Mode (C)

Experiments (cont'd)

- Program #2 (**without** SILC)
  - Matrix A in the CRS format
  - Direct use of library function ssi_cg (the CG method)
- Computing environments
  - Notebook PC
    - CPU: Intel Pentium M 733 1.1 GHz, OS: Linux
  - A remote SMP machine in a 100 Base-TX LAN
    - CPU: dual Intel Xeon 2.8 GHz, OS: Linux



Experimental results



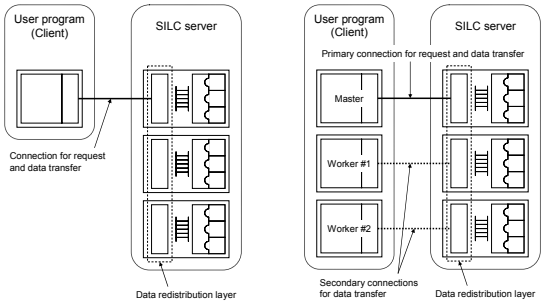
- Instant porting by means of the 2 SILC servers (without any modification in Program #1)
- Easy to use LAPACK from a user program of a different library (originally using ssi_cg with the CRS format)

Summary

- Proposal of SILC
  - Simple Interface for Library Collections
- Source-level independence between user programs and matrix computation libraries
  - Easy access to alternative solvers and matrix storage formats, possibly in other libraries
  - Instant porting to other computing environments without any modification in user programs

Future work: distributed SILC

- Sequential client with MPI-based server
- MPI-based client and server



Advertisement

- SILC version 1.1 is freely available
  - For UNIX environments
  - Sample programs in C, Fortran, and Python
  - Documentations
- Visit the SILC home page for more info

  http://ssi.is.s.u-tokyo.ac.jp/silc/
- I am willing to do a demonstration of SILC upon request