Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA '06)
Umea, Sweden, June 18-21, 2006

Distributed SILC: An easy-to-use interface for MPI-based parallel matrix computation libraries

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

Outline

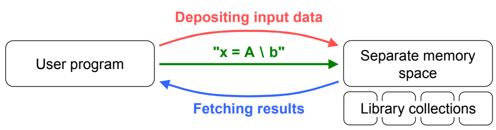
- Background
  - Ways of using matrix computation libraries
- Distributed SILC
  - An easy-to-use interface for MPI-based parallel matrix computation libraries
- Examples of SILC applications
  - Performance results
- Summary and future work

Background

- The burden of using matrix computation libraries
  - Incompatible application programming interfaces
  - Various computing environments with their own “special” libraries
- Modifications to user programs are needed
  - When using alternative libraries and computing environments
- Proposal of SILC
  - **S**imple **I**nterface for **L**ibrary **C**ollections
  - A framework for using matrix computation libraries in a language- and computing environment-independent manner

What is SILC ?

- Basic ideas
  - **Depositing input data** (such as matrices and vectors) to a separate memory space
  - **Making requests for computation** using mathematical expressions in the form of text
  - **Fetching the results** of computation



The traditional programming vs. SILC

- A program that solves $Ax = b$ using ScaLAPACK in C

```
double *A, *B;
int desc_A[9], desc_B[9], *ipiv, info;
/* create matrix A and vector B */
pdgesv(N, NRHS, A, IA, JA, desc_A, ipiv, B, IB, JB, desc_B,
       &info);
/* solution x is stored in B */
```

- A program that makes use of ScaLAPACK via SILC

```
silc_envelope_t A, b, x;
/* create matrix A and vector b */
SILC_PUT("A", &A);
SILC_PUT("b", &b);
SILC_EXEC("x = A \\ b"); /* call for pdgesv() for example */
SILC_GET(&x, "x");
```

Characteristics and benefits of SILC

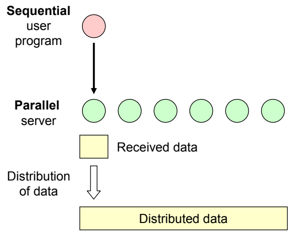
- Environment-independent
  - Sequential, shared-memory parallel, and distributed parallel environments
- Language-independent
  - Libraries and user programs in different languages
- Easy access to different libraries
  - Support for various solvers, matrix storage formats, and arithmetic precisions

MPI-based SILC system

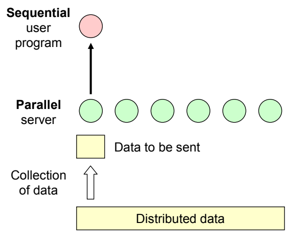
- Currently based on a client-server model
  - A SILC server is an MPI-based parallel program
  - Support for both sequential user programs and MPI-based parallel user programs
- Data redistribution mechanism
  - The server keeps data in a distributed manner
  - Support for various data distributions
    - 2D block-cyclic distribution,
    - 1D row-block and column-block distributions, etc.
  - In different matrix storage formats
    - Dense, band, the CRS format, etc.

Data transfer: the sequential case

- SILC_PUT

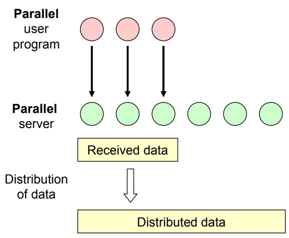


- SILC_GET

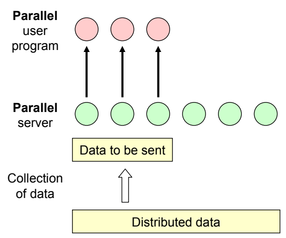


Data transfer: the parallel case

- SILC_PUT



- SILC_GET



Performance comparisons

- The traditional programming vs. SILC
- Examples of SILC applications
  1. Solution of a dense system with ScaLAPACK
     - MPI-based parallel user programs
  2. Solution of an initial-value problem of a PDE
  3. Cloth simulation
     - Sequential user programs

Solving $Ax = b$ with ScaLAPACK

- Traditional

```
pdgesv(N, NRHS, A, IA, JA,
  desc_A, ipiv, B, IB, JB,
  desc_B, &info);
```

- SILC

```
SILC_PUT("A", &A);
SILC_PUT("b", &b);
SILC_EXEC("x = A \\ b");
SILC_GET(&x, "x");
```

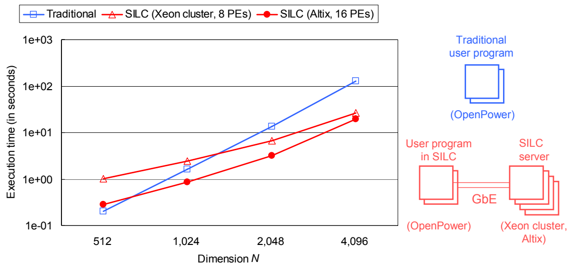
- MPI-based parallel user programs and SILC server
- Matrix A in the dense format (2D block-cyclic distribution)

Tested environments

- For both user programs
  - IBM OpenPower 710 (Power5 1.65 GHz × 4)
- For SILC servers
  - Xeon cluster (Intel Xeon 2.8 GHz × 8)
  - SGI Altix 3700 (Intel Itanium2 1.3 GHz × 16)
- Gigabit Ethernet (1 Gbps)
- Computation in double precision real

Solving $Ax = b$ with ScaLAPACK (results)

- Traditional: elapsed time in `pdgesv`
- SILC: elapsed time from connection until SILC_GET
- Speedups (N = 4,096): 4.88 (Xeon cluster), 6.46 (Altix)



An initial-value problem of a PDE

- Solve the 1D time-dependent diffusion equation

$$\frac{\partial \varphi}{\partial t} = \frac{\partial^2 \varphi}{\partial x^2} \quad (t \ge 0, 0 \le x \le \pi)$$

under the initial condition $\varphi = \sin x \; (t = 0, 0 \le x \le \pi)$ and boundary conditions $\varphi = 0 \; (t > 0, x = 0)$ and $\varphi = 0 \; (t > 0, x = \pi)$

- By the Crank-Nicolson method
  - Solution of a sparse linear system $Ax = b$ for each time step using the CG method in Lis (an iterative solvers library)
  - Matrix A is an $N \times N$ sparse matrix with $3N - 2$ non-zero elements, stored in the CRS format

An initial-value problem of a PDE (cont'd)

- Traditional

Prepare A and x
For each time step {
  Construct b from x
  Solve $Ax = b$ with `lis_solve`
}

- SILC

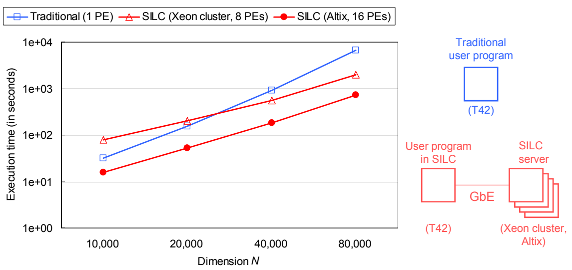
Prepare A and x
```
SILC_PUT("A", &A);
```
For each time step {
  Construct b from x
```
  SILC_PUT("b", &b);
  SILC_EXEC("x = A \\ b");
  SILC_GET(&x, "x");
```
}

Tested environments

- For both user programs
  - IBM ThinkPad T42 (Intel Pentium M 1.7 GHz)
- For SILC servers
  - Xeon cluster (Intel Xeon 2.8 GHz × 8)
  - SGI Altix 3700 (Intel Itanium2 1.3 GHz × 16)
- Gigabit Ethernet (1 Gbps)
- Computation in double precision real

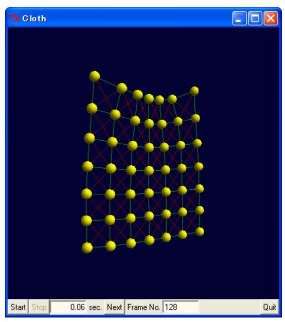
An initial-value problem of a PDE (results)

- Execution time (in seconds) of the first 20 time steps
- Speedups (N = 80,000): 3.38 (Xeon cluster), 9.12 (Altix)



Cloth simulation

- A simulator of cloth based on the mass-spring model
- An implicit integrator by Baraff & Witkin (1998)
- Code written in Python
- SciPy for solving a sparse linear system $A\Delta v = b$
- OpenGL for rendering the results of simulation
- GUI for controlling the simulation interactively



Cloth simulation (cont'd)

- Traditional

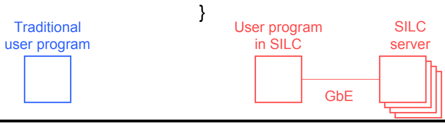
For each time step {
    Compute force f_0
    Construct A and b
    Solve $A\Delta v = b$ with **SciPy**
    Update velocity v
    Update position x
}

- SILC

For each time step {
    Compute force f_0
    Construct A and b

```
SILC_PUT("A", &A);
SILC_PUT("b", &b);
SILC_EXEC("d = A \\ b");
SILC_GET(&d, "d"); /* Δv */
```

    Update velocity v
    Update position x
}



Cloth simulation (results)

- Execution time of the first 100 time steps
  - In the case of 8^2 particles (dimension 192)
  - Matrix A consists of 5,652 non-zero elements, stored in the CRS format

		Time (sec.)	Speedup
Traditional	T42	121.74	1.00
SILC	T42 / Xeon cluster (8 PEs)	39.51	3.08
	T42 / Altix (16 PEs)	23.71	5.14

Summary and future work

- Distributed SILC: An easy-to-use interface for MPI-based parallel matrix computation libraries
  - Good speedups even at the cost of data transfer
  - Support for sequential and parallel user programs
  - Easy access to alternative libraries and computing environments (no need to modify user programs)
- Future work
  - Ready-made modules for various MPI-based parallel matrix computation libraries
  - Performance evaluation of the system