IFIP International Conference on Network and Parallel Computing (NPC 2006)
October 2–4, 2006, in Tokyo, Japan

A Performance Evaluation Model for the SILC Matrix Computation Framework

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

Purpose

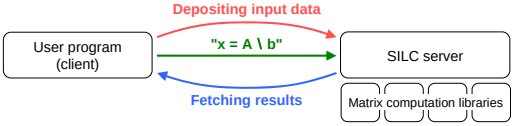
- A model-based analysis of the benefits and cost in the SILC framework
  - SILC: A simple interface for matrix computation libraries based on a **client-server architecture**
  - 1st benefit: Independence from libraries, computing environments, programming languages
  - 2nd benefit: Speedups
  - Cost: Data transfer between a client and a server
- A performance evaluation model for SILC
  - **How much speedup** is expected if a fast library is used via SILC **at the cost of data transfer**

Outline

- Overview of the SILC framework
- Performance evaluation model for SILC
- Experiments
  - Verify the effectiveness of the model
  - Observations on the model's utility
- Concluding remarks

Overview of the SILC framework

- **S**imple **I**nterface for **L**ibrary **C**ollections
  - Currently based on a client-server architecture
- 3 steps to use a library
  - **Depositing data** (e.g., matrices and vectors) to a server
  - **Making requests for computation** by means of textual mathematical expressions
  - **Fetching the results** of computation requests



Example: Solving a linear system $Ax = b$ in SILC

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

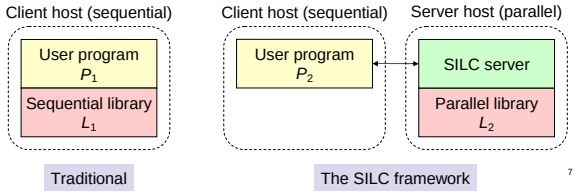
- Independence from libraries
  - E.g., solvers are specified by server configurations
- Independence from computing environments
  - Servers run in both sequential and parallel environments
- Independence from programming languages
  - By means of textual mathematical expressions

Why a performance evaluation model?

- Because use of fast libraries via SILC results in some speedups even at the cost of data transfer
- Matrix computations tend to take more time than data communications of input/output data
  - LU factorization (dense): $O(N^3)$ flop vs. $O(N^2)$ data
    - N: dimension
  - The CG method (sparse): $O(\alpha N)$ flop vs. $O(N)$ data
    - In the case of a sparse matrix with several non-zero diagonals
    - α: iteration count
  - Communication time is proportional to the amount of data
- How much speedup at how much cost?
  - A performance evaluation model answers this question

Performance evaluation model for SILC

- Estimates the performance ratio of P_1 to P_2
  - P_1 uses L_1 in the traditional programming style
  - P_2 uses L_2 through a SILC server
  - Both P_1 and P_2 perform the same matrix computations to solve a given problem



Traditional | The SILC framework

Estimated performance ratio T_c/T_s

- T_c : Execution time of P_1 in a sequential client host
- T_s : Execution time of P_2 in the same client host together with a SILC server in a parallel server host

$$T_c = X/C$$

$$T_s = X/S + Y/B + ZD$$

  - C, S : Performance rates of the client & server hosts (flops)
    - S depends on the degree of parallelism in the server host
  - B : Bandwidth (bps)
  - D : Latency (sec.)
  - X : Problem size (flop)
  - Y : Amount of data to be transferred (bits)
  - Z : Minimum number of pairs of send/recv system calls

How to determine parameter values

- By running P_1 (using L_1) in the client host
  - C : Performance rate of the client host (flops)
- By running P_1 (using L_2) in the server host
  - S : Performance rate of the server host (flops)
- By means of a network performance benchmark
  - B : Bandwidth (bps)
  - D : Latency (sec.)
- Determined by a given problem
  - X : Problem size (flop)
  - Y : Amount of data transfer (bits)
  - Z : Minimum number of pairs of send/recv calls

Performance of SILC_PUT & SILC_GET

- Comparable to the maximum data transfer rate if the data size is large
  - Example: Performance results in the case of transferring a vector of dimension 10^7
  - The proportions of the PUT/GET data transfer rates to the maximum data transfer rate (measured by Netperf) shown in parentheses

Server host	Client host	PUT		GET	
		Server side	Client side	Server side	Client side
ssixc0	ssixc1	856.1 (90.9%)	880.4 (93.5%)	869.9 (92.4%)	868.3 (92.2%)
ssixc0	altix	728.7 (98.4%)	720.5 (97.3%)	884.4 (94.2%)	890.3 (94.8%)

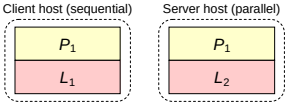
(Data transfer rates in Mbps; measured in the same GbE LAN.)

Experiments

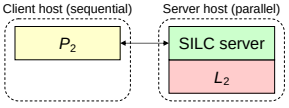
- Determine how accurately the proposed model estimates the performance ratio of P_1 to P_2
- Test problems
  1. Solution of a linear system with the CG method
  2. Dot product of two vectors
  3. Solution of a linear system with LAPACK
  4. Estimation of the condition number of a band matrix
  5. The CG method in SILC's mathematical expressions

Experimental procedure

- Run P_1 to obtain the estimated performance ratio T_c/T_s



- Run P_2 to obtain the actual performance ratio of P_1 to P_2



- Examine several cases of a problem to find a correlation between estimated and actual performance ratios

Test environments

Environment	Client host	Server host	Interconnect	B (Mbps)	D (sec.)
E3	t42	ssixc0 (2 PEs)	GbE	700.31	1.25e-04
E4	t42	ssixc0 (2 PEs)	Fast Ethernet	94.13	1.25e-04
E5	t42	altix (16 PEs)	GbE	709.04	1.24e-04

Host	Specifications
t42	IBM ThinkPad T42, Intel Pentium M 735 1.7 GHz, Memory: 512 MB, L2 cache: 2 MB, Fedora Core 4
ssixc0	IBM eServer xSeries 335, dual Intel Xeon 2.8 GHz, Memory: 1 GB, L2 cache: 512 KB, Red Hat Linux 8.0
altix	SGI Altix 3700, Intel Itanium2 1.3 GHz × 32, Memory: 32 GB, Red Hat Linux Advanced Server 2.1

(All these hosts are in the same Gigabit Ethernet LAN.)

13

Problem 1: Solving a linear system $Ax = b$ with the CG method

- Solve the following PDE using a finite difference approximation on a uniform grid

$$-u''(x) + 3u(x) = \cos(\pi x),\ 0 < x < 1,\ u(0) = u(1),\ u'(0) = u'(1)$$

  - The resulting linear system $Ax = b$ is SPD, so that the CG method is used
  - A is an $N \times N$ sparse matrix with $3N$ non-zero elements (stored in the CRS format)
- Used libraries
  - L_1: A sequential version of Lis
  - L_2: An OpenMP-based parallel version of Lis
    - Lis: An iterative solvers library (free software)
  - With the maximum iteration count m specified

14

Problem 1: Solving a linear system $Ax = b$ with the CG method (cont'd)

- Program P_1 (traditional)

```
lis_solve(A, b, x, params, options, status);
```

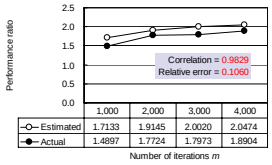
- Program P_2 (for SILC)

```
SILC_PUT("A", &A);
SILC_PUT("b", &b);
SILC_EXEC("x = A \\ b");  /* call for lis_solve */
SILC_GET(&x, "x");
```

- Test cases
  - Problem 1a. $N = 10^4$, in E4 (with Fast Ethernet)
  - Problem 1b. $N = 10^4$, in E3 (with GbE)
  - Problem 1c. $N = 10^5$, in E3 (with GbE)

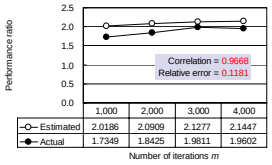
15

- Problem 1a ($N = 10^4$, FE)



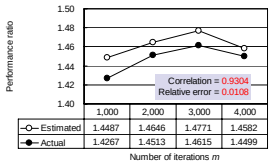
	1,000	2,000	3,000	4,000
Estimated	1.7133	1.9145	2.0020	2.0474
Actual	1.4897	1.7724	1.7973	1.8904

- Problem 1b ($N = 10^4$, GbE)



	1,000	2,000	3,000	4,000
Estimated	2.0186	2.0909	2.1277	2.1447
Actual	1.7349	1.8425	1.9811	1.9602

- Problem 1c ($N = 10^5$, GbE)



	1,000	2,000	3,000	4,000
Estimated	1.4487	1.4646	1.4771	1.4582
Actual	1.4267	1.4513	1.4615	1.4499

# of iterations m	1,000	2,000	3,000	4,000
Z	16	16	16	16
Problems 1a (E4) & 1b (E3)				
X (Mflop)	171.70	343.37	515.03	686.70
Y (Mbits)	4.27	4.27	4.27	4.27
S (Mflops)	808.33	820.72	833.99	838.07
C (Mflops)	385.73	385.06	386.89	386.95
Problem 1c (E3)				
X (Mflop)	1,717.00	3,433.61	5,150.23	6,866.85
Y (Mbits)	42.72	42.72	42.72	42.72
S (Mflops)	257.94	259.77	259.43	257.51
C (Mflops)	176.38	176.52	175.08	176.18

Summary of experimental results

Problem	Correlation	Error
1. Solution of a linear system with the CG method	0.9304 ~	~ 0.1181
2. Dot product of two vectors	0.9995 ~	~ 0.2340
3. Solution of a linear system with LAPACK	0.9987 ~	~ 0.0847
4. Estimation of the condition number of a band matrix	0.9827 ~	~ 0.1025
5. The CG method in SILC's mathematical expressions	0.9977 ~	~ 0.2099

- A clear correlation of more than 0.93 between estimated and actual performance ratios
  - Relative errors of less than 0.23
- The proposed model can accurately estimate the performance ratio of P_1 to P_2

17

Observations

- Communication overhead p (in seconds)

$$p = Y/B + ZD$$

- The ratio p/T_s
  - The proportion of the communication overhead to the execution time of P_2

	p (sec.)	Number of iterations m			
		1,000	2,000	3,000	4,000
Problem 1a ($N = 10^4$, FE)	0.4559	68.22%	52.15%	42.47%	35.75%
Problem 1b ($N = 10^4$, GbE)	0.0081	3.67%	1.90%	1.29%	0.98%
Problem 1c ($N = 10^5$, GbE)	0.0630	0.94%	0.47%	0.32%	0.24%

Observations (cont'd)

- The ratio S/C that satisfies $T_s = T_c$

$$S/C = 1 + pS/X$$

  - A server host needs to be faster than a client host by the factor of S/C in order to cancel the communication overhead

	Number of iterations m			
	1,000	2,000	3,000	4,000
Problem 1a ($N = 10^4$, FE)	1.0685	1.0345	1.0230	1.0171
Problem 1b ($N = 10^4$, GbE)	1.0381	1.0194	1.0131	1.0099
Problem 1c ($N = 10^5$, GbE)	1.0095	1.0048	1.0032	1.0024

19

Concluding remarks

- A performance evaluation model for SILC
  - Estimates the speedup to be achieved by using a fast library via SILC
  - Allows a quantitative analysis of the communication overhead in SILC
  - Predicts how faster a server host needs to be than a client host to offset the communication overhead
- A reasonable description of the cost and benefits in the SILC framework

20

Future work: Application of the model

- Forthcoming computing environments
  - 10 Gigabit Ethernet, faster machines, etc.
- WAN & Grid environments
  - The presented experiments dealt with only LAN (low latency) & simple data communications
- Distributed SILC
  - An MPI-based implementation of SILC for distributed parallel computing environments

21

Acknowledgment

- This research was supported by a grant from the CREST program of Japan Science and Technology Agency (JST)

SILC version 1.1 is freely available at
http://ssi.is.s.u-tokyo.ac.jp/silc/

22