Scalable Software Infrastructure Project

A. Nishida, H. Kotakemori, T. Kajiyama, A. Nukada

http://ssi.is.s.u-tokyo.ac.jp/

SC06
November 11-17
Tampa, FL, USA

Abstract. The objectives of our project are (1) the development of a basic library of solutions and algorithms required for large scale scientific simulations, which have been developed separately in each field, and (2) its integration into a scalable software infrastructure.

Lis

Lis is a Library of Iterative Solvers for linear systems, providing 12 iterative methods, 10 preconditioners, and 11 sparse matrix storage formats. Both sequential and parallel computing environments are supported, and double and quadruple precisions can be used through a common interface.

Quadruple precision

Lis uses the "double-double" precision, together with Intel SSE2 instructions for high performance.

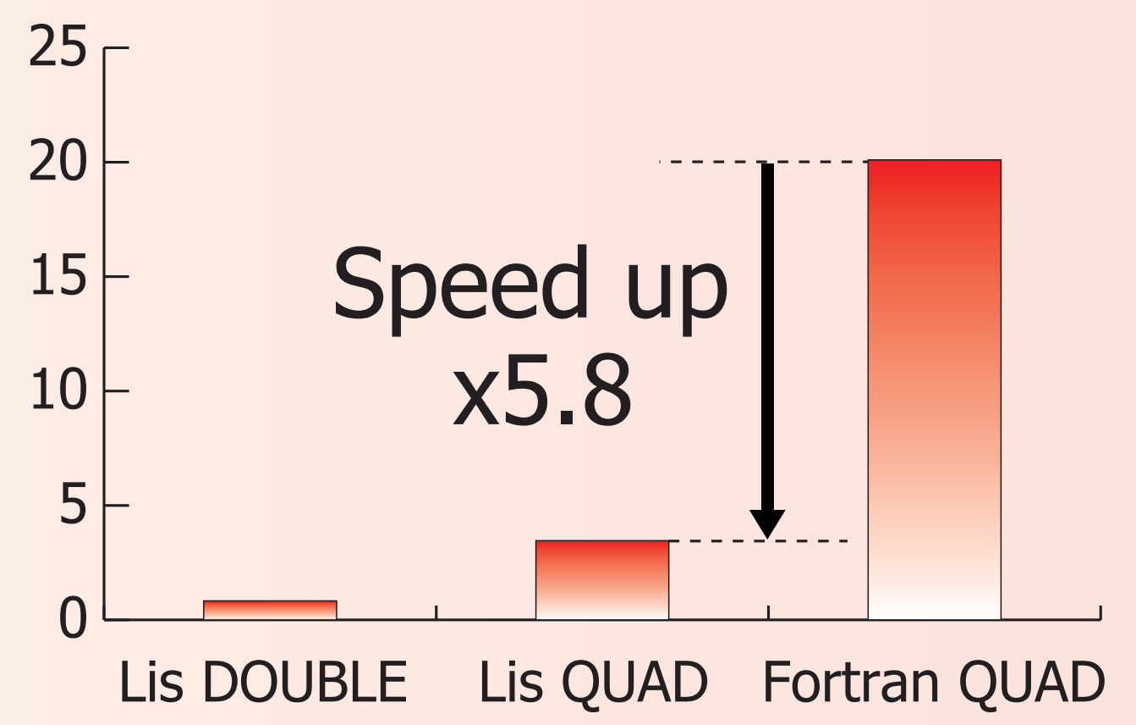
	Exp	Fraction		Exp	Fraction
	11 bits	52 bits		11 bits	52 bits

Toeplitz matrix (dimension 10^5)

$$A = \begin{pmatrix} 2 & 1 & & & & & \\ 0 & 2 & 1 & & & & \\ 1.4 & 0 & 2 & 1 & & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1.4 & 0 & 2 & 1 \\ & & & & 1.4 & 0 & 2 \end{pmatrix}$$

BiCG without preconditioner
Convergence criteria 10^{-12}

Execution times (sec.) for 50 iterations



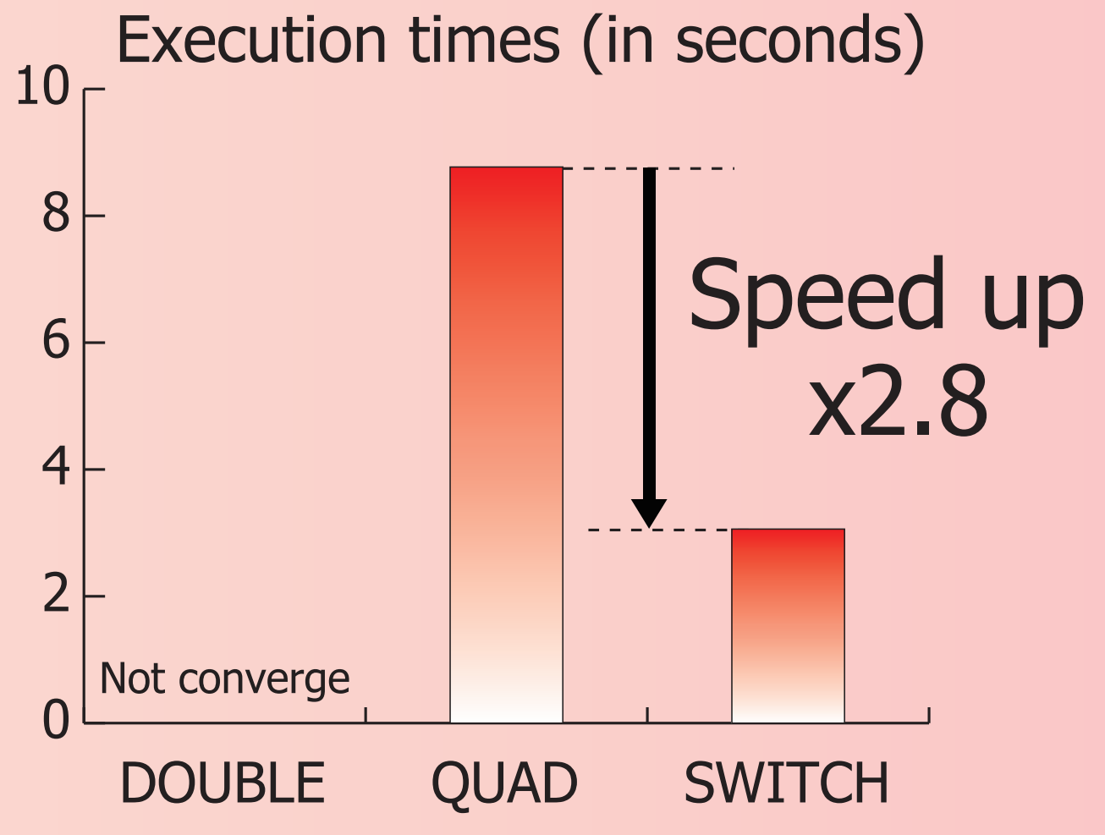
Mixed-precision iterative method

We proposed the SWITCH algorithm for further speed-ups by means of the double and quadruple precisions. The quadruple precision is not used for all the iterations, but only when it is necessary.

The SWITCH Algorithm

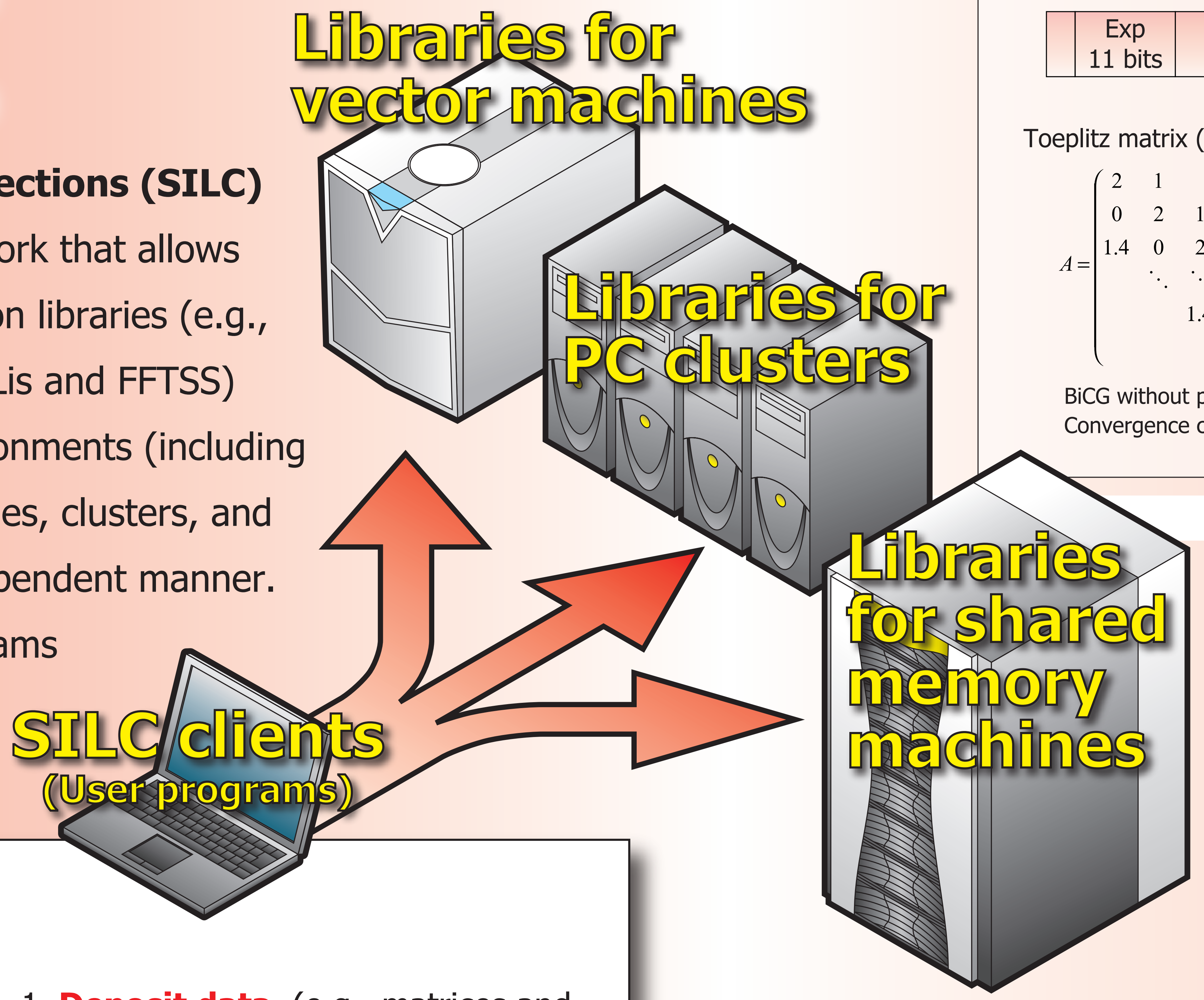
```
for (k=0; k<maximum iterations; k++) {
    Use **double precision** iterative method;
    if ( nrm2<restart_tol ) break;
}
Fill the work variables with zeros, except x;
for (k=k+1; k<maximum iterations; k++) {
    Use **quadruple precision** iterative method;
}
```

Execution times (in seconds)



SILC

Simple Interface for Library Collections (SILC) is an easy-to-use application framework that allows you to use various matrix computation libraries (e.g., BLAS/LAPACK, ScaLAPACK, and our Lis and FFTSS) in different kinds of computing environments (including desktop PCs, shared memory machines, clusters, and vector machines) in a language-independent manner. You don't need to modify user programs when using alternative libraries and computing environments.


Libraries for vector machines
Libraries for PC clusters
Libraries for shared memory machines
SILC clients
(User programs)

A user program in C for solving Ax = b in the SILC framework

```
silc_envelope_t A, b, x;
/* prepare matrix A and vector b */
SILC_PUT(”A”, &A);
SILC_PUT(”b”, &b);
SILC_EXEC(”x = A \\ b”);
SILC_GET(&x, ”x”);
```

1. **Deposit data** (e.g., matrices and vectors) to a SILC server.
2. **Send requests for computation** using mathematical expressions in the form of text.
3. **Fetch the results** of computation.

Benefits

SILC provides independence from libraries, environments, and programming languages. C, Fortran, Java, and Python are currently supported.

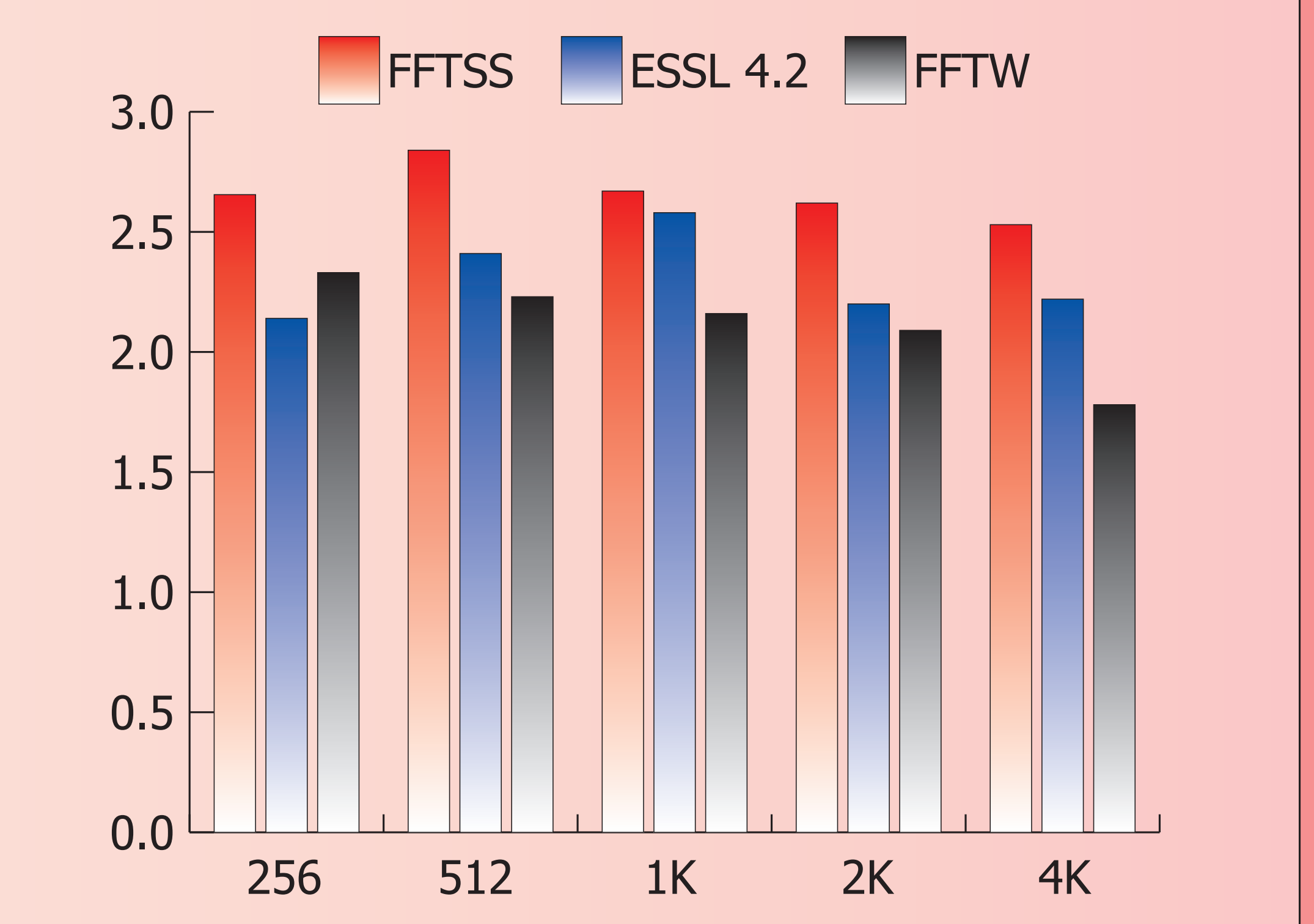
Costs in SILC

Primary costs are of data transfer between user programs and SILC servers, although some speedup is likely by means of fast matrix computation libraries.

FFTSS

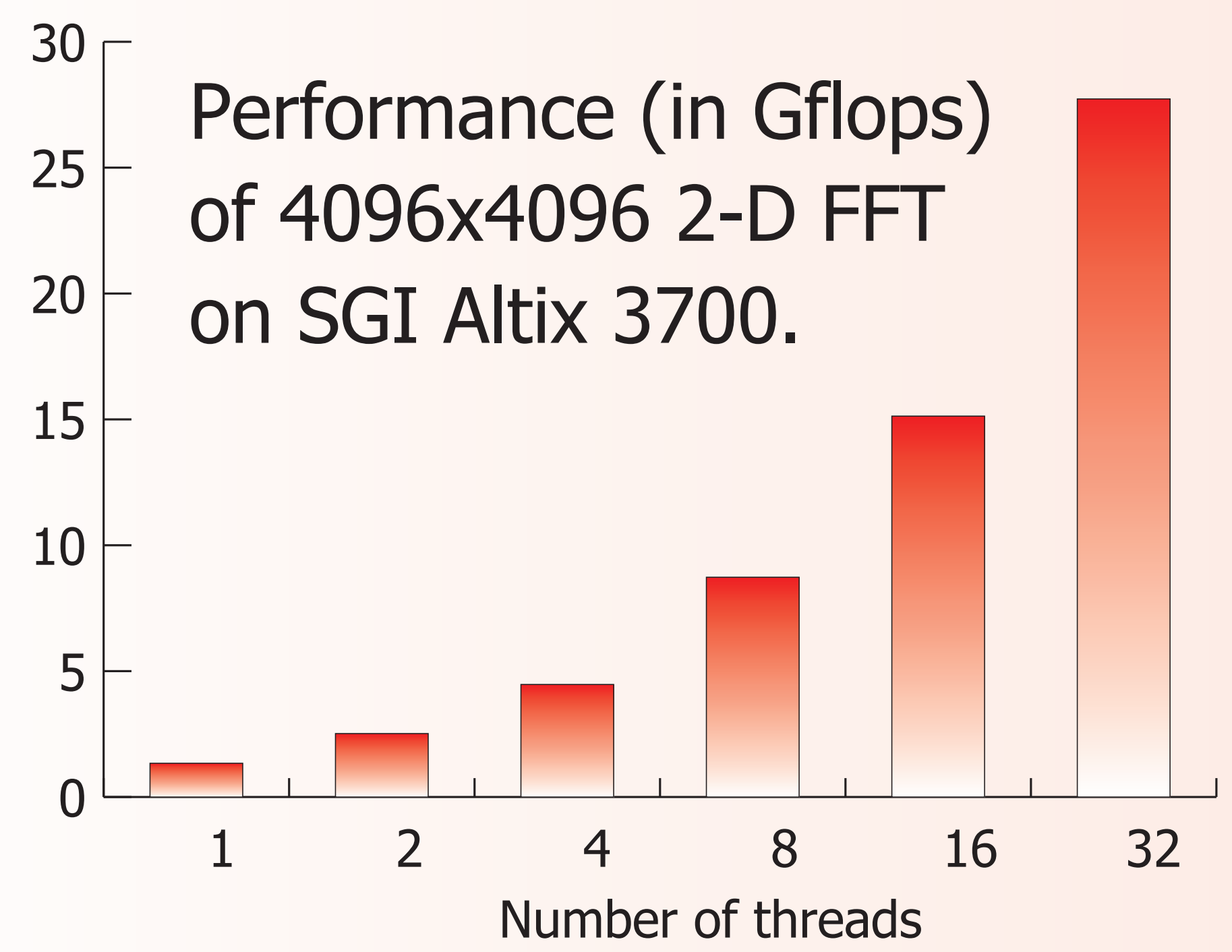
FFTSS is a fast Fourier Transform library for various computing environments. To achieve high performance, some processor specific instructions such as FMA, SSE2&3, and BlueGene/L SIMOMD are supported.

1-D FFT on POWER5 1.65GHz



Scalable Performance with OpenMP

Performance (in Gflops) of 4096x4096 2-D FFT on SGI Altix 3700.



Explicit data copy between the cache and main memory for efficient use of the cache memory.

Auto-tuning of FFT kernels, radices, and their order

Some of the FFT kernels are unrolled and software pipelined to assist compilers' optimization. As a result, efficient binary codes are generated and they outperform some vendors' FFT libraries. The interface of the library is almost compatible with FFTW3 which is the de facto standard library. This makes it easy to port your applications written for FFTW.