

Hidehiko Hasegawa

- 1983: University of Library and Information Science, the smallest National university
- March 1994: Visiting Researcher at iCL, University of Tennessee, Knoxville
- 1994-95 in Japan:
 - a bad harvest of rice,
 - Earth quake in Kansai-area,
 - terrorism with sarin
- Oct. 2002: University of Tsukuba at Kasuga Campus (Two national universities are merged)

1994

- i486 or Pentium
- Windows 3.1
- PVM
- MasPar
- Thinking Machine CM-5
- Intel Paragon
- IBM SP2
- telnet/ftp or rlogin

Before

- No experience to stay outside of Japan
- More than \$10,000 paid for English School
- Afraid of everything

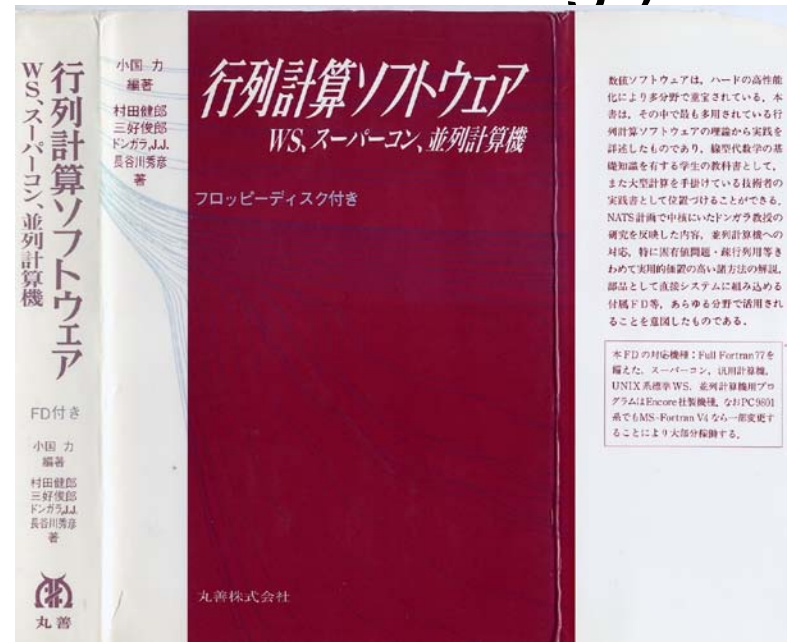
After

- Still poor English
- Not hesitate to ask
- Try to communicate

Research(Before)

- Convergence analysis of ICCG method by using eigenvalue and eigenvector
- Develop Linear Algebra Software for Vector-machine, Main-frame, and WS (Dense and Band, Direct solver and Eig.)

- Preconditioned iterative methods for 3D Convective Diffusion



Studied something during stay

- Used many types of machines -- 'being able to ask something easily to a skilled people' is very important
- To hear many talks -- 'Style is different in US and Japan'
- To see or to meet people -- 'Direct communication' is important.

A homework does not finish yet.....

Research(After)?

- Helped to make new parallel libraries
- Spread new things to many people:
‘Japanese’ intro. or sample
Education
- Support to make a relationship
between researchers
- Still research, however it’s
outside of main stream



One other approach

I believe Computing power may be used for

- Quality of Computation
- Ease of Use

Main ~~Another~~ stream of HPC

Computing Power should be used for

- Speed up
- ~~Quality of Computation~~
- Low Overhead
- Larger problem
- ~~Ease of Use~~

Recent research

- **Lis & Lis-test**: a Library of Iterative Solvers
by H. Kotakemori
- **SiLC**: Simple Interface of Library Collections
by T. Kajiyama
- **QuPAT**: Quadruple Precision Arithmetic Toolbox
by T. Saito, Tokyo University of Science

Motivation of Lis

- Krylov methods converge at most n iterations (n : dimension of Matrix) in exact computation.
- Diverge, stagnation, and many iterations occur because of round-off errors.
- The best choice depends on problem, usage, computing environment, etc.
- High Accuracy computation is one choice, however it is very costly.
(Double size of Memory; 20 times of Computation time)

Lis provides more than $10^3 \cdot 13 \cdot 11$ combinations

Precond.
Jacobi
SSOR
ILU(k)
Hybrid
I+S
SAINV
SA-AMG
Crout ILU
additive schwarz
User defined

Solvers
CG
BiCG
CGS
BiCGSTAB
BiCGSTAB(l)
GPBiCG
BiCGSafe
Orthomin(m)
GMRES(m)
TFQMR
Jacobi
Gauss-Seidel
SOR

Storage Format
CRS: Compressed Row
CCS: Compressed Column
MSR: Modified Compressed Sparse Row
DIA: Diagonal
ELL: Ellpack-Itpack gen. diag.
JDS: Jagged Diagonal
COO: Coordinate
DNS: Dense
BSR: Block Sparse Row
BSC: Block Sparse Column
VBR: Variable Block Row

Scenario of Lis and Lis-test

1) Fast Quadruple Arithmetic Operations

- Not standards
- double-double : use 2 double floating numbers
- Accelerate computation with SSE2
- Computation time: 3.2 times of Double

2) Mixed Precision Iterative method

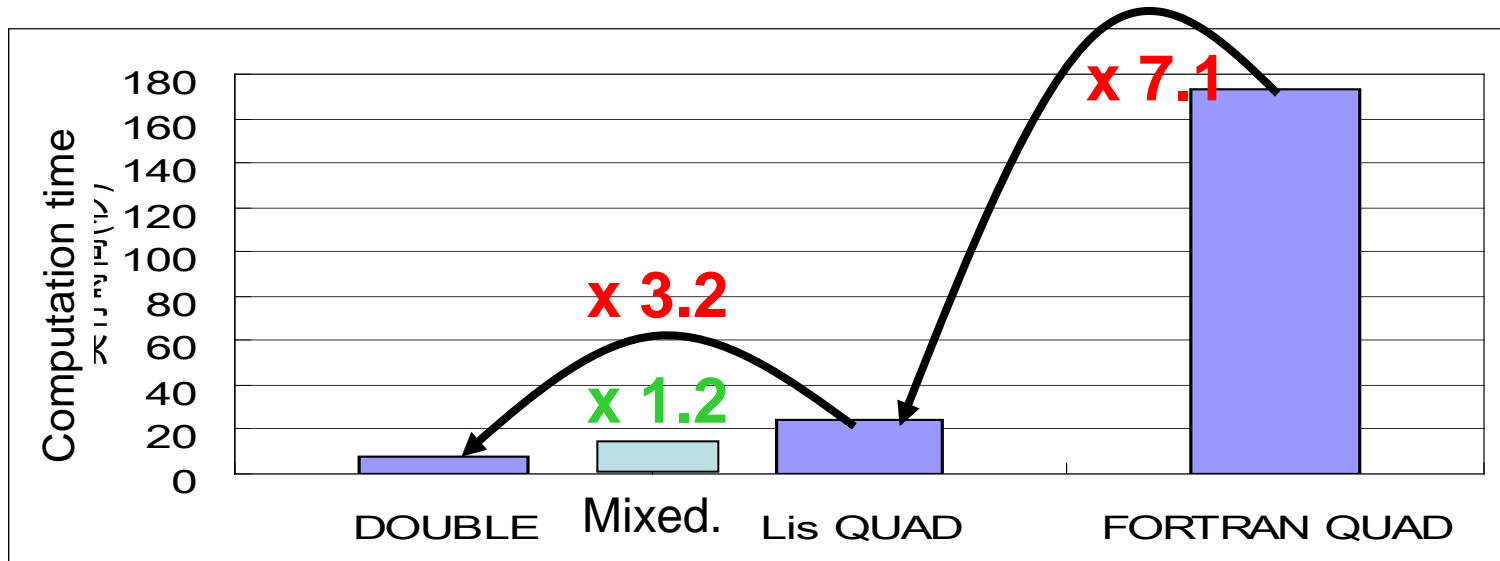
- Reduce Computation time with less Quadruple Arithmetic Operations
- Restart with different precision
- Computation time: 1.2 times of Double

3) Auto restart strategy (iterative refinement)

- Automatically changes precision

Computation time

Poisson ($n=10^6$, CRS) , Xeon 2.8GHz



Parallel

4

8

*3.74

*6.84

4

8

*3.88

*7.61

Parallel Issues for Fast Quad.

- Depends on the implementation of Ax , $A^T x$, $M^{-1}x$, $M^{-T}x$, and Matrix Storage Format
- Data transfer is almost same
- Heavy Computation
 - Suitable for Distributed Parallel
- Less round-off errors
 - light preconditioner (easy to parallelize)

Lis-test for evaluation

- Consists of two Windows files
- Not necessary to install. Run from USB
- Prepare Matrix data as text file with Matrix Market' exchange format
- Run in parallel if the PC is multi-core
- To click, solutions, history, etc are computed
- (new!) Input is acceptable from the Web

Lis-test for windows 0.1

Matrix A: C:\Documents and Settings\hasegar\ Open Cancel

RHS b: File b=(1,...,1)^T b=A * (1,...,1)^T

Dimension: 37054 x 37054
 Nonzeros : 544430
 Include b: Yes

Solvers

<input type="checkbox"/> CG	<input type="checkbox"/> CR	ALL	CLEAR
<input checked="" type="checkbox"/> BiCG	<input checked="" type="checkbox"/> BiCR		
<input type="checkbox"/> CGS	<input type="checkbox"/> CRS		
<input type="checkbox"/> BiCGSTAB	<input type="checkbox"/> BiCRSTAB		
<input checked="" type="checkbox"/> GPBiCG	<input checked="" type="checkbox"/> GPBiCR		
<input type="checkbox"/> BiCGSafe	<input type="checkbox"/> BiCRSafe		
<input type="checkbox"/> TFGMR	<input type="checkbox"/> BiCGSTAB(l) l = 2		
<input type="checkbox"/> Jacobi	<input type="checkbox"/> GMRES(m) m = 40		
<input type="checkbox"/> Gauss-Seidel	<input type="checkbox"/> FGMRES(m) m = 40		
<input type="checkbox"/> SOR w = 1.9	<input type="checkbox"/> ORTHOMIN(m) m = 40		

Conditions

$\|rk\|_2/\|r0\|_2 \leq 1.0e-012$ MaxIters = 1000

Storage: CRS Block Size = 2

Precision: Quadruple # of Threads: 1

Preconditioners

None **ALL** **CLEAR**

Jacobi

ILU(k) k = 0

ILUT drop = 0.1 rate = 10.0

Crout ILU drop = 0.1 rate = 10.0

SSOR

Hybrid SOR w = 1.5

tol = 1.0e-003 MaxIter = 25

I+S m = 3 alpha = 1.0

SAINV drop = 0.1

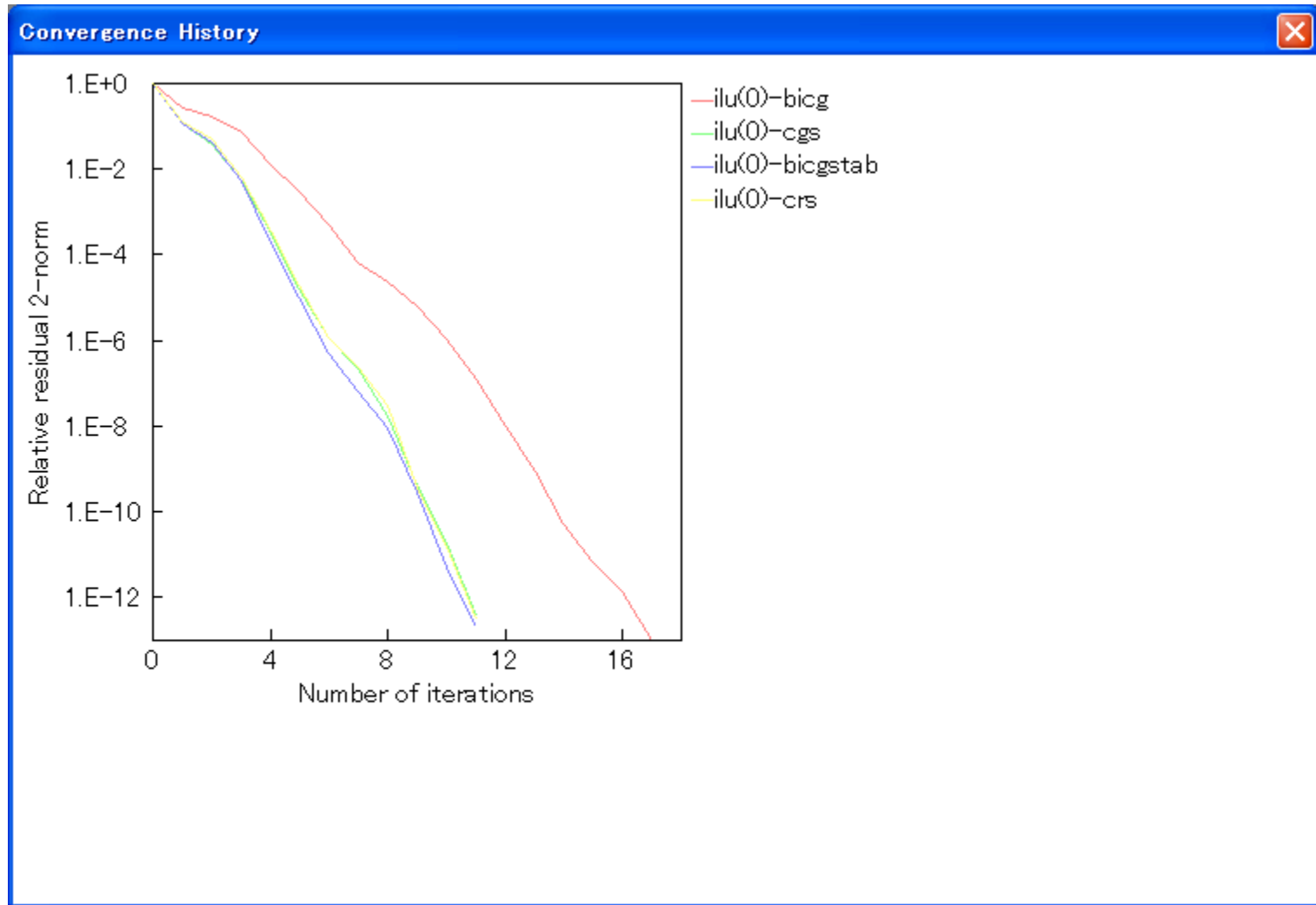
SA-AMG

SET **RUN** HISTORY

	Solver	Precon	P	T	Iter.	Sec.	p_cre	p_sol	i_sol	TRR	Stora...	Opt
RDY	bicg	ilu(0)	Q	1							CRS	"C"
RDY	epbicg	ilu(0)	Q	1							CRS	"C"
RDY	bicr	ilu(0)	Q	1							CRS	"C"
RDY	epbicr	ilu(0)	Q	1							CRS	"C"

Lis-test: GUI for Library Lis

Comparison is done easily!



New! Get matrix data from Web

Open Matrix Market format file

MatrixMarket

- Harwell-Boeing Collection
 - ACOUST
 - AIRTFC
 - ASTROPH
 - MCCA**
 - MCFE
 - BCSPWR
 - BCSSTRUC1
 - BCSSTRUC2
 - BCSSTRUC3
 - BCSSTRUC4
 - BCSSTRUC5
 - BCSSTRUC6
 - CANNES
 - CEGB
 - CHEMIMP
 - CHEMWEST
 - CIRPHYS
 - COUNTERX
 - DWT
 - ECONAUS
 - ECONIEA
 - FACSIMILE
 - GEMAT
 - GRENOBLE
 - JAGMESH
 - LANPRO
 - LAPLACE
 - LAPU
 - LNS
 - LOCKHEED

MATRIX MARKET Home Search Browse Resources
Version 3.0

MCCA: Nonlinear radiative transfer and statistical equilibrium in astrophysics
Atom=Ca2, Atmos=VAL3C

from set [ASTROPH](#), from the [Harwell-Boeing Collection](#)

[\[Download\]](#) [\[Visualizations\]](#) [\[Matrix Statistics\]](#) [\[Set Information\]](#)

Download as

- Compressed [MatrixMarket format](#) file: [mcca.mtx.gz](#) (22864 bytes)
- Compressed [Harwell-Boeing format](#) file: [mcca.rua.gz](#) (18514 bytes)

Help: My browser can't read the compressed data files. [What now?](#)

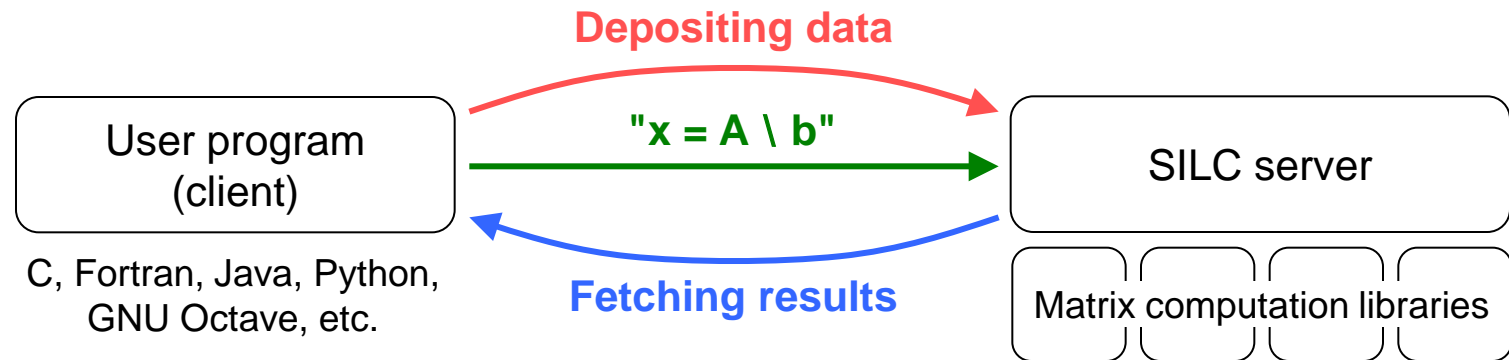
File URL:

Files of type:

Open (O) Cancel

SILC: Simple Interface for Library Collections

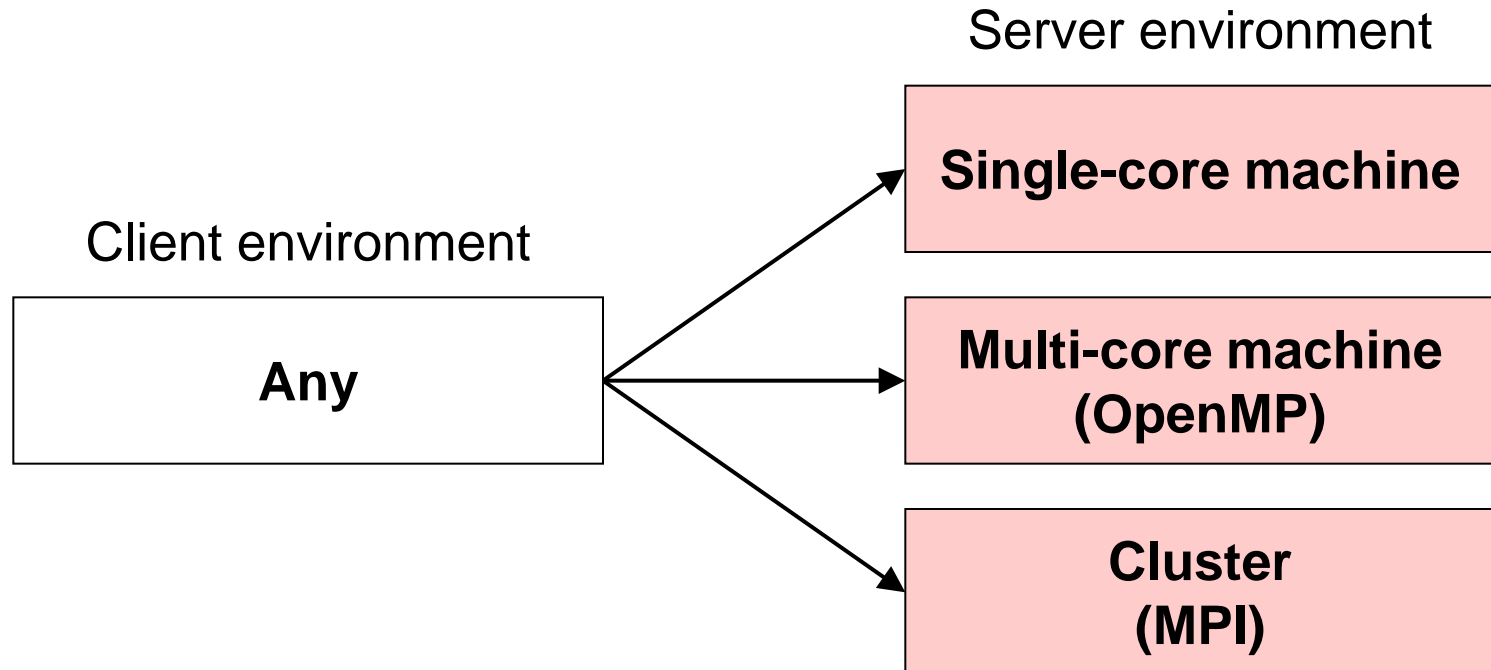
- Based on a client-server architecture



- Computing environment independent
- Language independent
 - User programs in C, Fortran, Java, Python, etc.
 - A SILC client API is required for each language

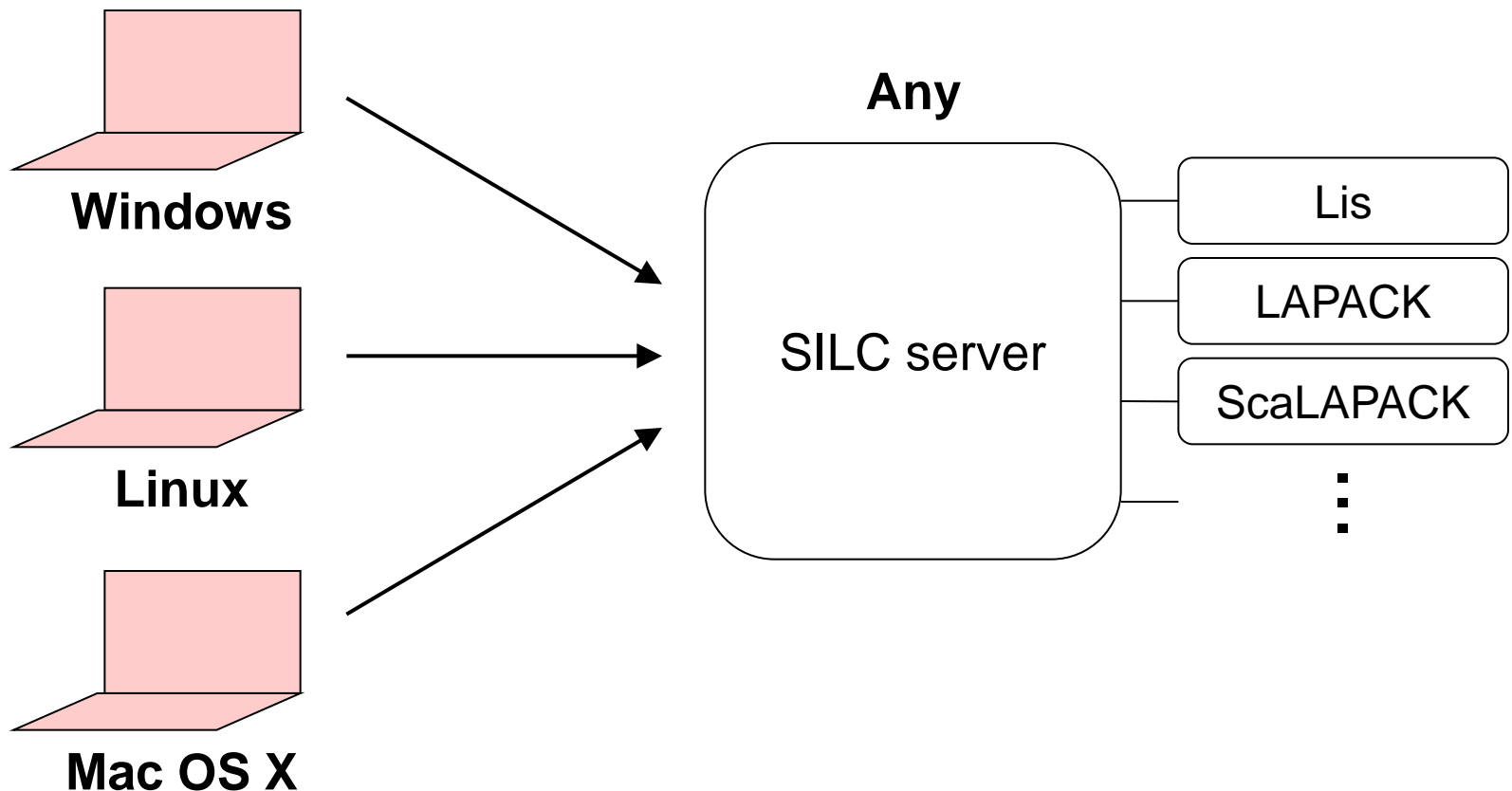
Independence from environments

- Computing environments can be single- and multi-core machines and clusters



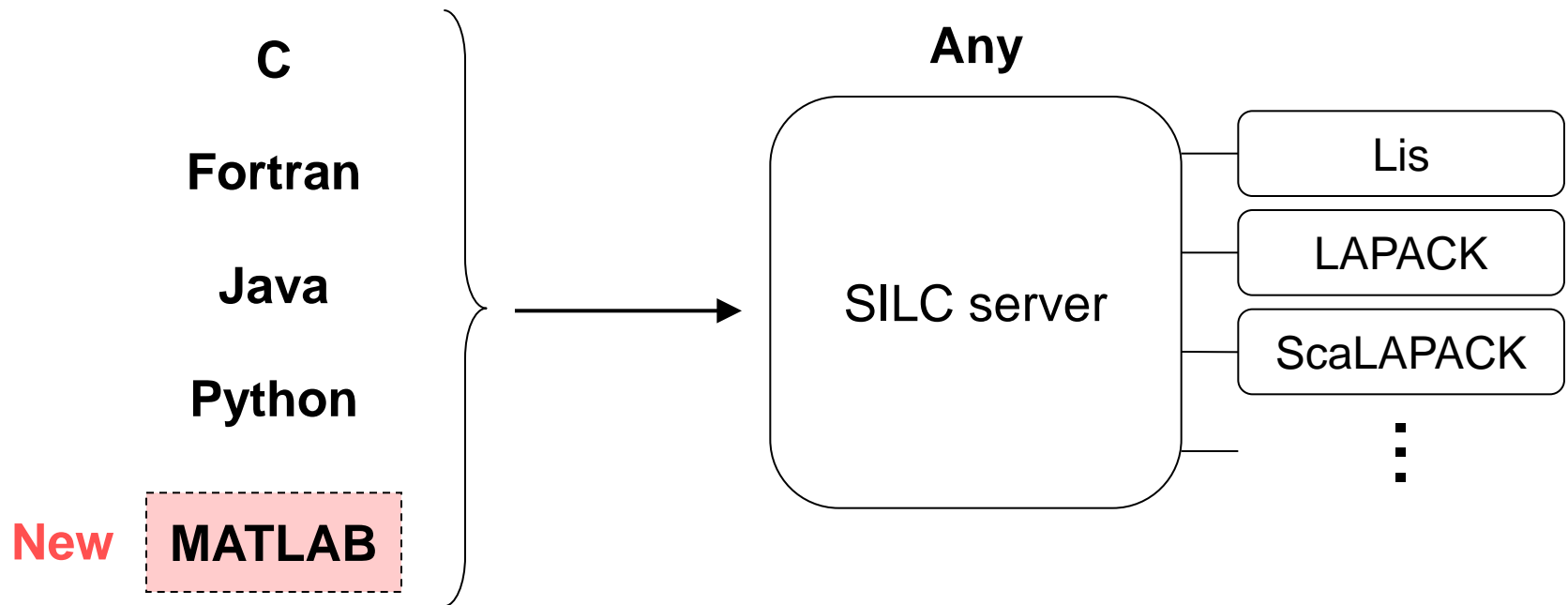
Independence from environments (cont'd)

- Various operating systems are supported



Independence from languages

- Various programming languages can be used to write user programs



Objective of the present research

- Implementation of **MATLAB-SILC**, a SILC client API for MATLAB, allowing users to
 - Write user programs for SILC in MATLAB
 - Simple binding to various matrix computation libraries
 - Utilize remote parallel computers (possibly interactively)
 - Reduce programming effort in different computing and programming environments

API of MATLAB-SILC

- `SILC_INIT()`
 - Establish a connection to a SILC server
- `SILC_PUT(name, data)`
 - Deposit input data into the SILC server with a name for later reference
- `SILC_EXEC(expr)`
 - Send computation requests in the form of mathematical expressions
- `data = SILC_GET(name)`
 - Fetch the results of computation from the SILC server
- `SILC_FINALIZE()`
 - Close the connection with the SILC server

Supported data types and precisions

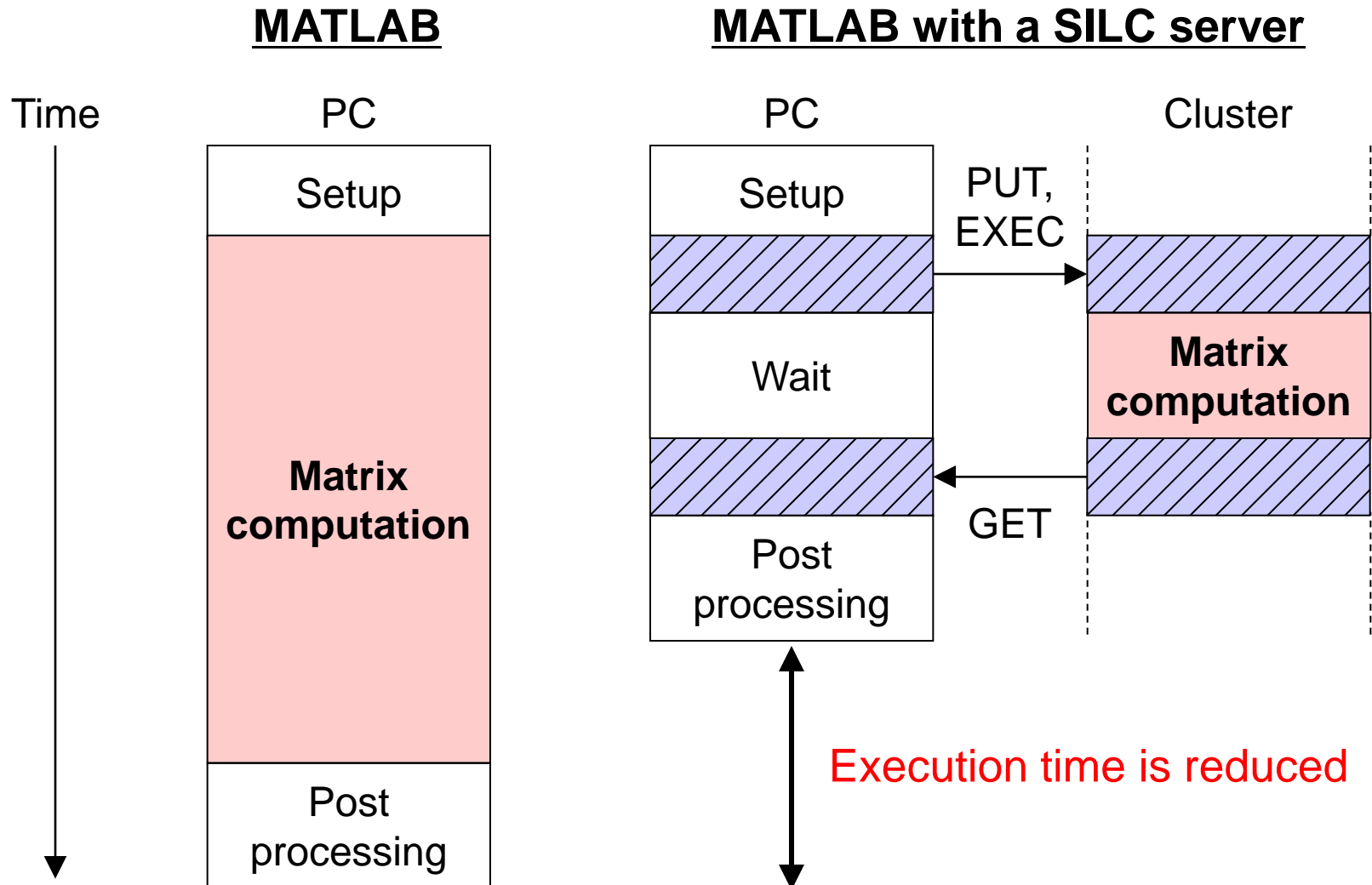
- Automatic format conversion occurs before/after data transfer
- Scalar, vector, dense matrix
 - MATLAB: the same “array” type, SILC: separate types
 - Supported data precisions
 - 32-bit & 64-bit integers
 - Single precision real & complex
 - Double precision real & complex
- Sparse matrix
 - MATLAB: Compressed Column Storage (CCS) format, SILC: Compressed Row Storage (CRS) format
 - Only double precision real is supported

Solve $Ax = b$ using MATLAB-SILC

```
% create matrix A and vector b
m = 100;
I = sparse(1:m, 1:m, 1);
U = sparse(1:m-1, 2:m, 1, m, m);
D = 4 * I - U - U';
A = kron(I, D) - kron(U, I) - kron(U', I);
b = A * ones(m^2, 1);

% solve the linear system Ax = b using MATLAB-SILC
SILC_INIT();
tic; % start a timer
SILC_PUT(' A', A);
SILC_PUT(' b', b);
SILC_EXEC(' x = A \ b');
x = SILC_GET(' x');
t = toc; % stop the timer
SILC_FINALIZE();
disp(sprintf('%d sec.', t)); % print the execution time
```

Main benefit of using MATLAB-SILC



Pros and cons of SILC

- Pros
 - Various computing environments can be used
 - Different solvers can be used
 - No modifications in user programs
 - Asynchronous parallelism
- Cons
 - SILC-based programming
 - Communication overhead

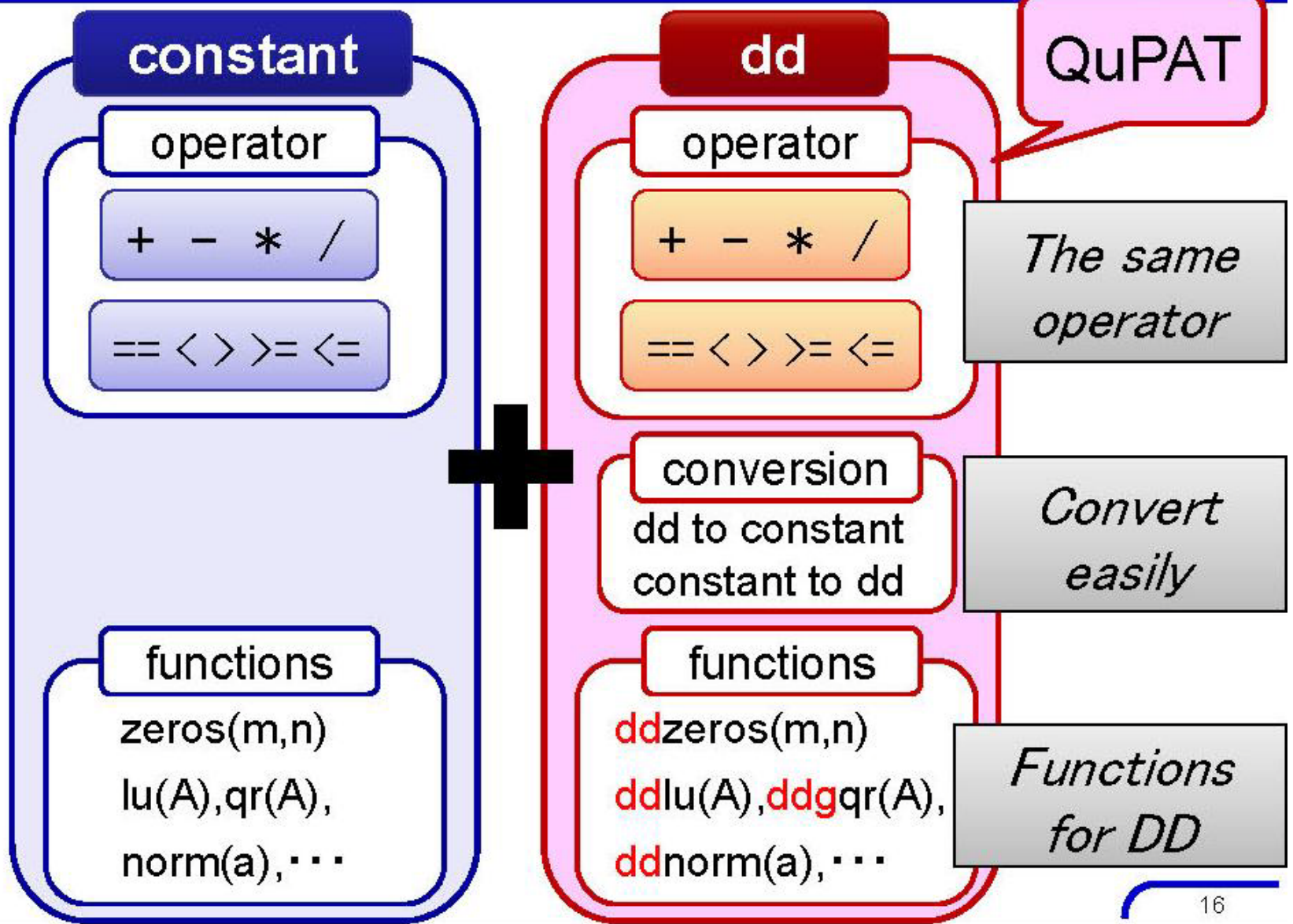
Collaborators and Acknowledgement in 2005

- TiS on ITBL
 - Y. Fukui (RIKEN)
 - K. Suzuki (Fujitsu)
 - Y. Sakaguchi (Fujitsu)
- Lis
 - H. Kotakemori
(JST/U Tokyo)
 - A. Fujii (Kogakuin U)
 - K. Nakajima (U Tokyo)
 - A. Nishida(U Tokyo/JST)
- SILC
 - T. Kajiyama
(JST/U Tokyo)
 - A. Nukada (JST/U Tokyo)
 - R. Suda (U Tokyo)
 - A. Nishida (U Tokyo/JST)
- Lis and SILC are parts of SSI project which is funded by JST/CREST

What is QuPAD?

- A toolbox of Scilab
- Double-Double used for Quadruple Arithmetic
- A new data type is introduced
- Operator overloading for $+: -: *:/$
- Only Scilab function is used

Relationship between Scilab and QuPAT



Computing area of rectangle

The image shows a screenshot of the SciPad 7.18.1 interface with a file named 'herons_formula.sce'. The code is as follows:

```
1 a = 1.0;
2 b = 10^(-16);
3 c = sqrt(a*a+b*b);
4 s = 0.5*(a+b+c);
5 area = sqrt(s*(s-a)*(s-b)*(s-c));
6
7 A = d2dd(a);
8 B = d2dd(b);
9 C = ddsqrt(A*A+B*B);
10 S = 0.5*(A+B+C);
11 AREA = ddsqrt(S*(S-A)*(S-B)*(S-C));
12 ddprint(AREA)
```

Annotations on the left side:

- A purple box labeled 'double' points to lines 1-5.
- A purple box labeled 'DD' points to lines 7-12.

Annotations on the right side:

- A light blue box labeled 'conversion' with a bracket points to lines 7 and 8.
- A light blue box labeled 'Modified function for DD' has arrows pointing to 'ddsqrt' in lines 9 and 11.
- A light blue box labeled 'Output for DD' has an arrow pointing to 'ddprint' in line 12.

Characteristics of QuPAD

- Simple Usage
 - A new data type for Quad.
 - same operators for Quad. And Double
 - new function name for Quad.
- Powerful tool for Numerical Analysis (Double, Quadruple, and Mixed Precision)
- Independent of Hardware and OS
- Free and Open Source

Professor and Tour guide



Welcome to Japan

To get codes Lis-test/Lis

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

lis ssi

検索



version 1.1.0•2

SILC is available!

- MATLAB-SILC is included in SILC v1.4
- SILC v1.4 are freely available at

<http://www.ssisc.org/silc/>

- Source (Unix/Linux, Windows, Mac OS X)
- Precompiled binary package for Windows
- Documentation, sample programs

QuPAT is available at

<http://www.mi.kagu.tus.ac.jp/qupat.html>