Scientific Computing

Some slides from **James Lambers**, Stanford

Dense Linear Algebra

- Scaling and sums
- Transpose
- Rank-one updates
- Rotations
- Matrix vector products
- Matrix Matrix products
- BLAS

Designing Numerical Software

- “Premature optimization is the root of all evil”
- First make the code produce the expected results
  - Validation
  - Then worry about performance
- Test individual components of the software separately
- Report and handle errors
- Validate input data

Mathematical Libraries

- BLAS – Basic Linear Algebra Subprograms
  - Three levels:
    - Level 1: Vector operations
    - Level 2: Matrix-Vector operations
    - Level 3: Matrix-Matrix operations
  - Originally written in F77, the standard has evolved to contain F77,F95,C,C++ libraries, each with their own version of the calling program. The dcopy routine is used to copy two vectors and is given by: f_dcopy for fortran users and c_dcopy for C users.
  - BLAS 1 was originally designed for vector processors. BLAS 2 and 3 were designed for cache-based computers
- LINPACK (Linear Algebra Package)
  - Linear equation and linear least squares solvers
  - Based on level 1 routines in BLAS

Mathematical Libraries, cont'd

- EISPACK (Eigenvalue Software Package)
  - Computes eigenvalues and eigenvectors
- LAPACK (Linear Algebra Package)
  - Originally designed to extend LINPACK and EISPACK on shared memory vector and parallel processors
  - Also implements highly-tuned system-dependend BLAS libraries, using BLAS levels 2 and 3 wherever possible
- SCALAPACK (Scalable Linear Algebra Package)
  - Designed to extend LAPACK routines to work on distributed memory parallel machines
  - Uses BLACS (Basic Linear Algebra Communications Subprograms) for interprocessor communication
- BLAS,LINPACK,EISPACK,LAPACK,SCALAPACK can be downloaded from http://www.netlib.org

Mathematical Libraries, cont'd

- PLAPACK (Parallel Linear Algebra Package)
  - Parallel linear algebra routines using high levels of abstraction
  - Allows implementation of advanced linear algebra routines
  - http://www.cs.utexas.edu/users/plapack
- FFTPACK (FFT Package)
  - Complex and real fft routines (available from netlib.org)
- VSIPL (Vector/Signal/Image Processing Library)
  - Attempt to create an fft standard
  - http://www.vsipl.org

Self-tuning Libraries

- PHIPAC (Portable High-performance ANSI C Linear Algebra Subprograms)
  - Run a performance tuner which stores information about particular machine to optimize BLAS routines
  - Performance tuner can take days because it creates system-specific versions of all BLAS routines
  - http://www.icsi.berkeley.edu/~bilmes/phipac/
- ATLAS (Automatically Tuned Linear Algebra Software)
  - System-specific information is placed into the design of one specific subroutine which can be called by all BLAS routines.
  - http://www.netlib.org/atlas/
- FFTW (Fastest Fourier Transform in the West)
  - System-specific information created at runtime using a "planner", which stores information used by "executor" functions during program execution
  - http://www.fftw.org

Commercial Libraries

- Hardware vendors sell highly-tuned versions of BLAS and LAPACK for their hardware
  - Intel Math Kernel Library (MKL)
    - http://www.intel.com/software/products/mkl/
  - HP MLIB (Mathematical Software Library)
  - IBM ESSL (Engineering and Scientific Subroutine Library)
  - SGI SCSL
  - SUN Performance Library
- Other libraries developed independently of BLAS, specifically for relevant hardware.
  - Visual Numerics' IMSL (Mathematical and Statistical Library)
    - http://www.vni.com
  - Numerical Algorithms Groups' Numerical Libraries
    - http://www.nag.co.uk

BLAS Structure

- Names are cryptic, follow a pattern

Prefix	Matrix Type	Name

- Example: DGEMM: double precision matrix multiplication of a general full matrix
- First versions written in Fortran 77
  - 1-based indexing
  - Call by reference
  - No dynamic memory allocation

BLAS, cont'd

Prefix	Fortran type
S	Real
D	Double precision
C	Complex
Z	Double precision complex

Type code	Matrix type
GE	General full
SY	Symmetric
HE	Hermitian
TR	Triangular
GB	General banded
SB	Symmetric banded

Level 1 BLAS

	dim	scalar	vector	vector	scalars	5-element array		prefixes
SUBROUTINE xROTG (					A, B, C, S)		Generate plane rotation	S, D
SUBROUTINE xROTMG(					D1, D2, A, B,	PARAM)	Generate modified plane rotation	S, D
SUBROUTINE xROT (	N,		X, INCX,	Y, INCY,	C, S)		Apply plane rotation	S, D
SUBROUTINE xROTM (	N,		X, INCX,	Y, INCY,		PARAM)	Apply modified plane rotation	S, D
SUBROUTINE xSWAP (	N,		X, INCX,	Y, INCY)			$x \leftrightarrow y$	S, D, C, Z
SUBROUTINE xSCAL (	N,	ALPHA,	X, INCX)				$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
SUBROUTINE xCOPY (	N,		X, INCX,	Y, INCY)			$y \leftarrow x$	S, D, C, Z
SUBROUTINE xAXPY (	N,	ALPHA,	X, INCX,	Y, INCY)			$y \leftarrow \alpha x + y$	S, D, C, Z
FUNCTION xDOT (	N,		X, INCX,	Y, INCY)			$dot \leftarrow x^T y$	S, D, DS
FUNCTION xDOTU (	N,		X, INCX,	Y, INCY)			$dot \leftarrow x^T y$	C, Z
FUNCTION xDOTC (	N,		X, INCX,	Y, INCY)			$dot \leftarrow x^H y$	C, Z
FUNCTION xxDOT (	N,		X, INCX,	Y, INCY)			$dot \leftarrow \alpha + x^T y$	SDS
FUNCTION xNRM2 (	N,		X, INCX)				$nrm2 \leftarrow \|x\|_2$	S, D, SC, DZ
FUNCTION xASUM (	N,		X, INCX)				$asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$	S, D, SC, DZ
FUNCTION IxAMAX(	N,		X, INCX)				$amax \leftarrow 1^{st} k \ni \|re(x_k)\| + \|im(x_k)\| = max(\|re(x_i)\| + \|im(x_i)\|)$	S, D, C, Z

Level 2 BLAS

	options			dim	b-width	scalar	matrix	vector	scalar	vector		
xGEMV (		TRANS,		M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xGBMV (		TRANS,		M, N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xHEMV (	UPLO,			N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHBMV (	UPLO,			N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHPMV (	UPLO,			N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xSYMV (	UPLO,			N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSBMV (	UPLO,			N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSPMV (	UPLO,			N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xTRMV (	UPLO,	TRANS,	DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTBMV (	UPLO,	TRANS,	DIAG,	N,	K,		A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTPMV (	UPLO,	TRANS,	DIAG,	N,			AP,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTRSV (	UPLO,	TRANS,	DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTBSV (	UPLO,	TRANS,	DIAG,	N,	K,		A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTPSV (	UPLO,	TRANS,	DIAG,	N,			AP,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z

	options			dim	scalar	vector	vector	matrix		
xGER (				M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	S, D
xGERU (				M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	C, Z
xGERC (				M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + A, A - m \times n$	C, Z
xHER (	UPLO,			N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^H + A$	C, Z
xHPR (	UPLO,			N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^H + A$	C, Z
xHER2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xHPR2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xSYR (	UPLO,			N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^T + A$	S, D
xSPR (	UPLO,			N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^T + A$	S, D
xSYR2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D
xSPR2 (	UPLO,			N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D

Level 3 BLAS

	options				dim	scalar	matrix	matrix	scalar	matrix		
xGEMM (			TRANSA,	TRANSB,	M, N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha op(A)op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$	S, D, C, Z
xSYMM (	SIDE,	UPLO,			M, N,		ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$	S, D, C, Z
xHEMM (	SIDE,	UPLO,			M, N,		ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$	C, Z
xSYRK (		UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$	S, D, C, Z
xHERK (		UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$	C, Z
xSYR2K(		UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^T + \bar{\alpha} BA^T + \beta C, C \leftarrow \alpha A^T B + \bar{\alpha} B^T A + \beta C, C - n \times n$	S, D, C, Z
xHER2K(		UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C, C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C, C - n \times n$	C, Z
xTRMM (	SIDE,	UPLO,	TRANSA,	DIAG,	M, N,	ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A)B, B \leftarrow \alpha Bop(A), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z
xTRSM (	SIDE,	UPLO,	TRANSA,	DIAG,	M, N,	ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A^{-1})B, B \leftarrow \alpha Bop(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z

Simple Vector Operations (BLAS level 1)

- Setting to zero or a specified constant
  - bzero = set elements to zero

```
for(i=0;i<N;i++)
    y[i]=0;
```

  - memset = set elements to a specified constant

```
for(i=0;i<N;i++)
    y[i]=value;
```

- Vector copy
  - c_dcopy=Double precision vector copy y=x

```
for(i=0;i<N;i++)
    y[i]=x[i];
```

Scalar-Vector accumulation

- daxpy=Double precision y=a*x+y

```
for(i=0;i<N;i++)
  y[i]+=a*x[i];
```

- daxpby=Double precision y=alpha*x+beta*y

```
for(i=0;i<N;i++)
  y[i]=alpha*x[i]+beta*y[i];
```

- Compiler will unroll these loops

Dot Product: BLAS ddot

No unrolling:

```
for(i=0;i<N;i++)
  ddot+=x[i]*y[i];
```

Two-way loop unrolling:

```
for(i=0;i<nend;i+=2) {
    ddot1+=x[i]*y[i];
    ddot2+=x[i+1]*y[i+1];
  }
```

- Effect of loop unrolling with N=1024 (8 kb), in MFlops

gcc –O0:
0: 235.90
2: 538.41 (2.28x)
4: 941.68 (3.99x)
6: 1024.97 (4.34x)
8: 1071.04 (4.54x)
10: 1053.29 (4.46x)
12: 1063.94 (4.51x)
14: 1061.28 (4.50x)

gcc –O2:
0: 223.60
2: 553.34 (2.47x)
4: 1092.23 (4.88x)
6: 1449.38 (6.48x)
8: 1548.78 (6.93x)
10: 1594.80 (7.13x)
12: 1572.16 (7.03x)
14: 1529.57 (6.84x)

• Loop unrolling reduces overhead and allows for pipelining
• Too much unrolling leads to register spilling

Some BLAS Level 1 Routines

Routine	Operation	Memory ops per iteration	Floating point ops per iteration	F:M ratio
dcopy	yi=xi	2	0	0
daxpy	yi=yi+a*xi	3	2	0.67
daxpby	yi=β*yi+α*xi	3	3	1.00
ddot	ddot=ddot+xi*yi	2	2	1.00

BLAS Level 2 routines

- Level 2 routines consist of 2 loops and include matrix copy, matrix transpose, and matrix-vector routines.
- Matrix copy: dge_copy

  Performs well because of unit stride on inner-most loop:

```
for(i=0;i<N;i++)
  for(j=0;j<N;j++)
   b[i][j]=a[i][j];
```

  The transpose case is not as straightforward:

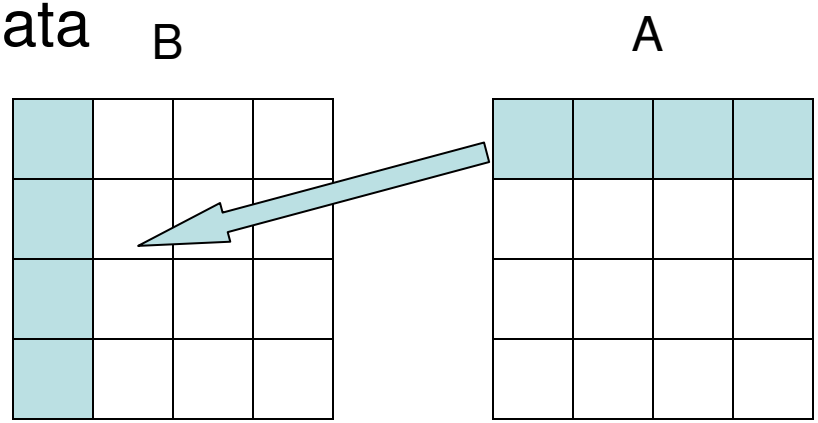
```
for(i=0;i<N;i++)
  for(j=0;j<N;j++)
   b[i][j]=a[j][i];
```

gcc –O2 (1024 X 1024 int)
Copy: 707.31 Mb/s
Transpose: 54.59 Mb/s

Transpose with 1-d blocking

```
1-d blocks (a and b ints):
int en = bsize*(n/bsize);
for(jj=0;jj<en;jj+=bsize)
  for(i=0;i<n;i++)
    for(j=jj;j<jj+bsize;jj++)
      b[i][j]=a[j][i];
```

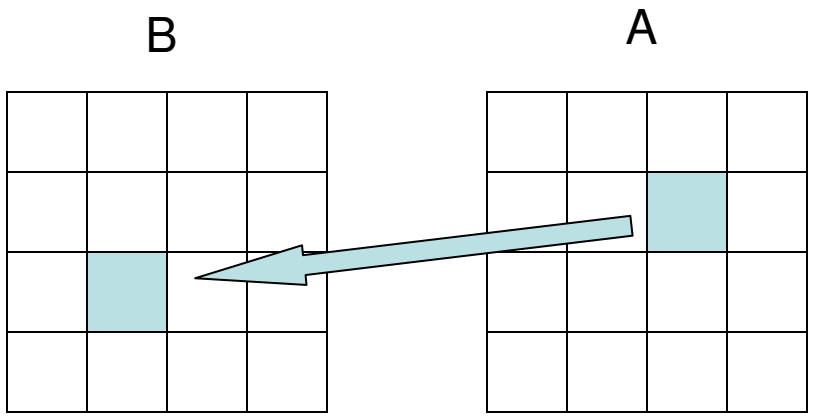
- After i=0 inner loop (assuming 32-byte cache blocks):
  - 8*bsize elements of a in L1 cache (a[0:bsize-1][0:7])
- Next 7 loops in i will hit a[][] data



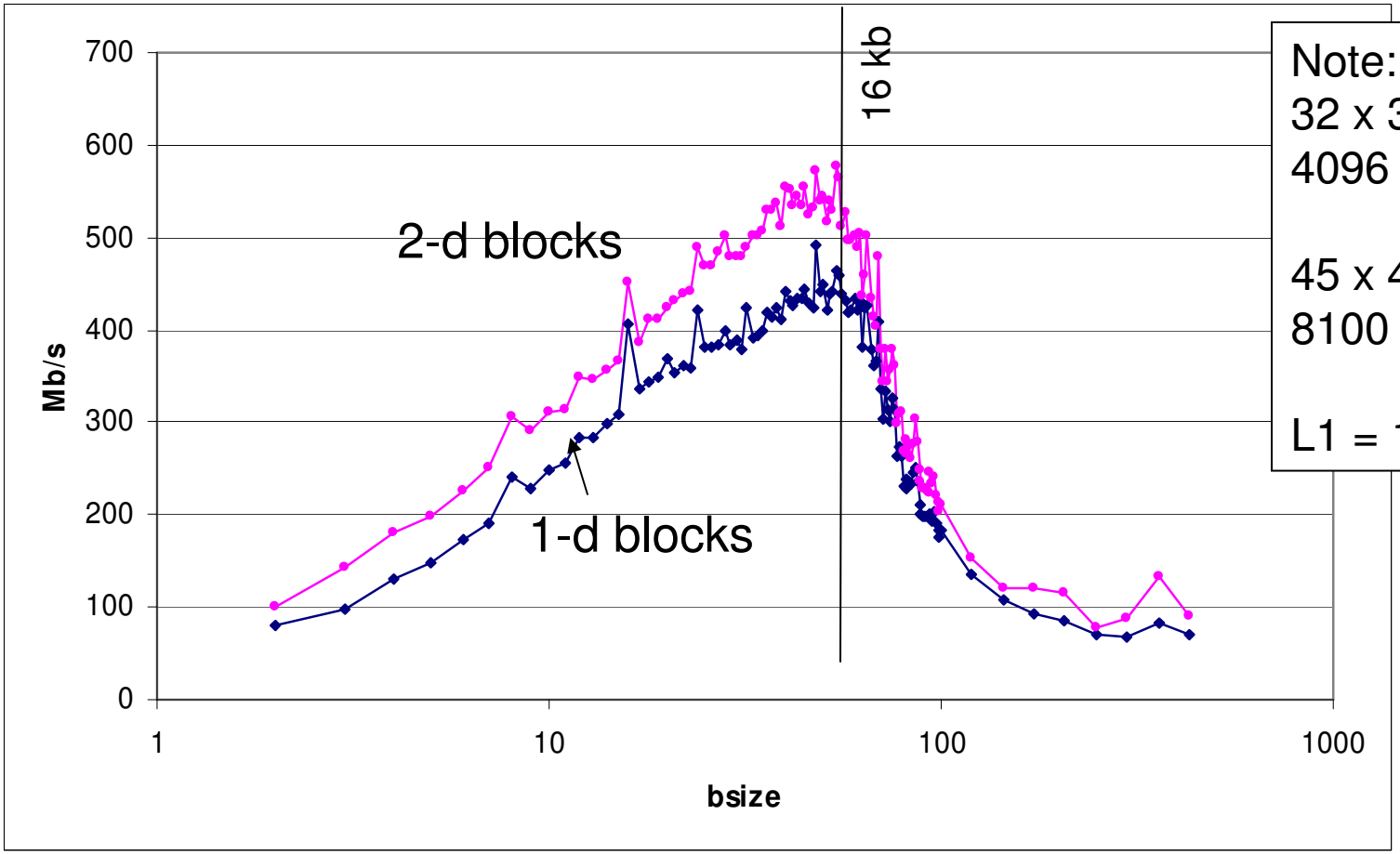
Transpose with 2-d blocking

2-d blocks (a and b ints):

```
int en = bsize*(n/bsize);
for(ii=0;ii<en;ii+=bsize)
   for(jj=0;jj<en;jj+=bsize)
     for(i=ii;i<ii+bsize;i++)
       for(j=jj;j<jj+bsize;j++)
         b[i][j]=a[j][i];
```



Effect of Blocking on Transpose Performance (1024 x 1024 int)

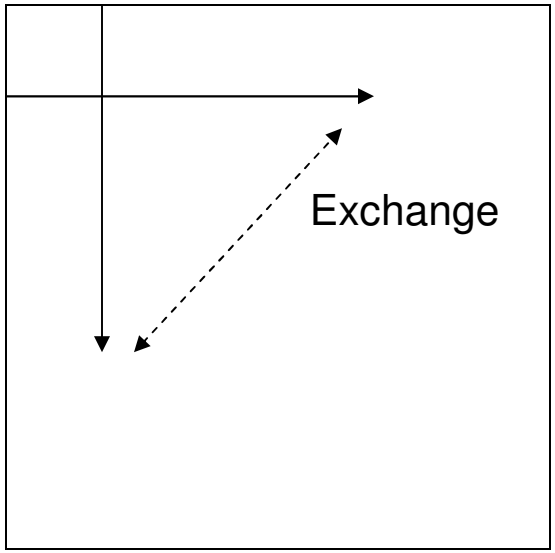


Peak throughput occurs when bsize=45
(when two 45x45 matrices fit into L1 cache!)

In-place Transpose: dge_trans

- Previous examples involved a copy and transpose
  - This is termed *out-of-place matrix transpose*
- Consider the *in-place matrix transpose* (method 1):

```
for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++) {
      temp = a[i][j];
      a[i][j] = a[j][i];
      a[j][i] = temp;
    }
```

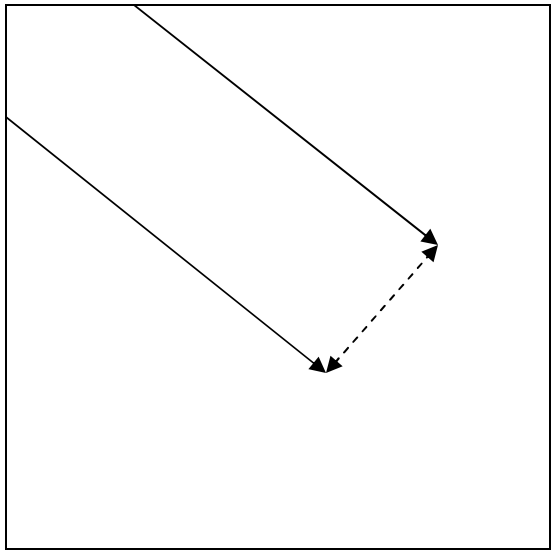


Move along a column and swap elements.

suffers from possible cache thrashing when n is a power of 2 since a[i*n+j] and a[j*n+i] can map to same cache location

In-place Transpose: Method 2

```
for(i=0;i<n;i++)
   for(j=i;j<n;j++) {
     temp = a[j-i][j];
     a[j-i][j] = a[j][j-i];
     a[j][j-i] = temp;
   }
```



Move along diagonals and swap elements.

- This method is less likely to incur cache thrashing when n is a power of 2 since stride is n+1 in j-loop:
- `a[j-i][j]=a[(j-i)*n+j]=a[(n+1)*j - i*n]`
- The BLAS routines employ blocking for in-place transpose

Summary of BLAS routines

- BLAS level 1: single-loop
  - Set to zero (bzero), set to a constant (memset), copy vectors (dcopy), inner-product (ddot), ax+y (daxpy), ax+by (daxpby)
- BLAS level 2: double-loop
  - matrix copy (dge_copy), matrix transpose (deg_trans), outer-product (dger), matrix-vector product (dgemv)
- BLAS level 3: triple-loop
  - matrix-matrix product (dgemm): optimized in-cache matrix solver. For use with dge_copy to create a blocked matrix-matrix multiply.

LAPACK

- Builds on BLAS for more linear algebra
- Linear Systems
  - LU, Cholesky
- Least Squares
  - QR
- Eigenproblems
- Singular Value Decomposition (SVD)

Dense Linear Algebra

- Matrix Decompositions
  - Linear systems
    - Symmetric,Nonsymmetric
  - Least squares
    - QR, SVD
- Eigenvalues
  - Rayleigh iteration
  - QR algorithm
- Standard libraries
  - Linpack, Lapack, Atlas, PetSC, MKL