Approach 3: block algorithms (nb = 256)

\[
\begin{align*}
\text{for } j &= 1 : nb : n \\
\text{for } k &= 1 : nb : n \\
\text{for } jj &= j : j + nb - 1 \\
\text{for } kk &= k : k + nb - 1 \\
\text{c}(:, jj) &= \text{a}(:, kk) * \text{b}(kk, jj) + \text{c}(:, jj);
\end{align*}
\]
In MATLAB, 1:256:1025 means 1, 257, 513, 769

- Note that we calculate

$$
\begin{bmatrix}
A_{11} & \cdots & A_{14} \\
\vdots & \ddots & \vdots \\
A_{41} & \cdots & A_{44}
\end{bmatrix}
\begin{bmatrix}
B_{11} & \cdots & B_{14} \\
\vdots & \ddots & \vdots \\
B_{41} & \cdots & B_{44}
\end{bmatrix}
= \begin{bmatrix}
A_{11}B_{11} + \cdots + A_{14}B_{41} & \cdots \\
\vdots & \ddots & \ddots
\end{bmatrix}
$$
Memory Management III

- Each block: \( 256 \times 256 \)

\[
\begin{align*}
C_{11} &= A_{11} B_{11} + \cdots + A_{14} B_{41} \\
C_{21} &= A_{21} B_{11} + \cdots + A_{24} B_{41} \\
C_{31} &= A_{31} B_{11} + \cdots + A_{34} B_{41} \\
C_{41} &= A_{41} B_{11} + \cdots + A_{44} B_{41}
\end{align*}
\]

- For each \((j, k)\), \(B_{k,j}\) is used to add \(A_{:,k} B_{k,j}\) to \(C_{:,j}\)
Example: when $j = 1, k = 1$

\[
\begin{align*}
C_{11} & \leftarrow C_{11} + A_{11}B_{11} \\
C_{21} & \leftarrow C_{21} + A_{21}B_{11} \\
    & \vdots \\
C_{41} & \leftarrow C_{41} + A_{41}B_{11}
\end{align*}
\]

Use Approach 2 for $A_{:,1}B_{11}$

$A_{:,1}$: 256 columns, $1024 \times 256/65536 = 4$ pages.

$A_{:,1, \ldots, :,4}$: $4 \times 4 = 16$ page faults in calculating $C_{:,1}$

For $A$: $16 \times 4$ page faults

$B$: 16 page faults, $C$: 16 page faults
BLAS defines only operations such as matrix-matrix products. How about operations like LU factorization for solving linear systems?

LAPACK – Linear Algebra PACKage, based on BLAS

Routines for solving
Systems of linear equations
Least-squares solutions of linear systems of equations
Eigenvalue problems, and
Singular value problems.

- Subroutines in LAPACK classified as three levels:
  - Driver routines: each solves a complete problem, for example solving a system of linear equations
  - Computational routines: each performs a distinct computational task, for example an LU factorization
  - Auxiliary routines: subtasks of block algorithms, commonly required low-level computations, a few extensions to the BLAS

- LAPACK provides both single and double versions
Naming: All driver and computational routines have names of the form XYYZZZ

- X: data type, S: single, D: double, C: complex, Z: double complex
- YY, indicate the type of matrix, for example
  - GB: general band
  - GE: general (i.e., unsymmetric, in some cases rectangular)
Band matrix: a band of nonzeros along diagonals

\[
\begin{bmatrix}
\times & \times & & & \\
\times & \times & \times & & \\
& \times & \times & \times & \\
& & \times & \times & \times \\
& & & \times & \times
\end{bmatrix}
\]

- ZZZ indicates the computation performed. For example,
SV  simple driver of solving general linear systems
TRF  factorize
TRS use the factorization to solve $Ax = b$ by forward or backward substitution
CON estimate the reciprocal of the condition number

- SGESV: simple driver for single general linear systems
- SGBSV: simple driver for single general band linear systems
Now optimized BLAS and LAPACK available on nearly all platforms
For example, Intel MKL (Math Kernel Library)
From LAPACK manual Third edition; Table 3.7
http://www.netlib.org/lapack/lug
LU factorization DGETRF: $O(n^3)$
Speed in megaflops ($10^6$ floating point operations per second)
## Block Algorithms in LAPACK II

<table>
<thead>
<tr>
<th></th>
<th>No. of CPUs</th>
<th>Block size</th>
<th>n 100</th>
<th>n 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec Alpha Miata</td>
<td>1</td>
<td>28</td>
<td>172</td>
<td>370</td>
</tr>
<tr>
<td>Compaq AlphaServer DS-20</td>
<td>1</td>
<td>28</td>
<td>353</td>
<td>440</td>
</tr>
<tr>
<td>IBM Power 3</td>
<td>1</td>
<td>32</td>
<td>278</td>
<td>551</td>
</tr>
<tr>
<td>IBM PowerPC</td>
<td>1</td>
<td>52</td>
<td>77</td>
<td>148</td>
</tr>
<tr>
<td>Intel Pentium II</td>
<td>1</td>
<td>40</td>
<td>132</td>
<td>250</td>
</tr>
<tr>
<td>Intel Pentium III</td>
<td>1</td>
<td>40</td>
<td>143</td>
<td>297</td>
</tr>
<tr>
<td>SGI Origin 2000</td>
<td>1</td>
<td>64</td>
<td>228</td>
<td>452</td>
</tr>
<tr>
<td>SGI Origin 2000</td>
<td>4</td>
<td>64</td>
<td>190</td>
<td>699</td>
</tr>
<tr>
<td>Sun Ultra 2</td>
<td>1</td>
<td>64</td>
<td>121</td>
<td>240</td>
</tr>
<tr>
<td>Sun Enterprise 450</td>
<td>1</td>
<td>64</td>
<td>163</td>
<td>334</td>
</tr>
</tbody>
</table>
Block Algorithms in LAPACK III

- 100 to 1000: number of operations 1000 times
- Block algorithms are not very effective for small-sized problems
- Clock speed of Intel Pentium III: 550 MHz
- Thus by block algorithms good performance can be achieved
Web page: http://math-atlas.sourceforge.net/

Programs specially compiled for your architecture
That is, things related to your CPU, size of cache, RAM, etc. are considered