4.3.1 Transpose Matrix-Vector Multiplication

4.3.1 Transpose Matrix-Vector Multiplication

Start of transcript. Skip to the end.

Dr. Robert van de Geijn: Computational scientists

who use linear algebra are typically concerned about two things.

One is performance.

If you spend $30 million on a supercomputer, 0:00 / 5:56 1.0x

you don't want to only utilize 5% of that computer because of the way you

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Discussion

Hide Discussion

Topic: Week 4 / 4.3.1

Add a Post

Show all posts ▼ by recent activity ▼

There are no posts in this topic yet.

✖

Homework 4.3.1.1

1/1 point (graded)

```
Algorithm: $y := \text{MVMULT\_T\_UNB\_VAR1}(A, x, y)$
Partition $A \rightarrow \left( A_L \,\middle|\, A_R \right)$, $y \rightarrow \left( \begin{array}{c} y_T \\ \hline y_B \end{array} \right)$
  where $A_L$ is $m \times 0$ and $y_T$ is $0 \times 1$
while $m(y_T) < m(y)$ do
  Repartition
    $\left( A_L \,\middle|\, A_R \right) \rightarrow \left( A_0 \,\middle|\, a_1 \,\middle|\, A_2 \right)$, $\left( \begin{array}{c} y_T \\ \hline y_B \end{array} \right) \rightarrow \left( \begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array} \right)$
  $\psi_1 := a_1^T x + \psi_1$
  Continue with
    $\left( A_L \,\middle|\, A_R \right) \leftarrow \left( A_0 \,\middle|\, a_1 \,\middle|\, A_2 \right)$, $\left( \begin{array}{c} y_T \\ \hline y_B \end{array} \right) \leftarrow \left( \begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array} \right)$
endwhile
```

```
Algorithm: $y := \text{MVMULT\_T\_UNB\_VAR2}(A, x, y)$
Partition $A \rightarrow \left( \begin{array}{c} A_T \\ \hline A_B \end{array} \right)$, $x \rightarrow \left( \begin{array}{c} x_T \\ \hline x_B \end{array} \right)$
  where $A_T$ is $0 \times n$ and $x_T$ is $0 \times 1$
while $m(A_T) < m(A)$ do
  Repartition
    $\left( \begin{array}{c} A_T \\ \hline A_B \end{array} \right) \rightarrow \left( \begin{array}{c} A_0 \\ \hline a_1^T \\ \hline A_2 \end{array} \right)$, $\left( \begin{array}{c} x_T \\ \hline x_B \end{array} \right) \rightarrow \left( \begin{array}{c} x_0 \\ \hline \chi_1 \\ \hline x_2 \end{array} \right)$
  $y := \chi_1 a_1 + y$
  Continue with
    $\left( \begin{array}{c} A_T \\ \hline A_B \end{array} \right) \leftarrow \left( \begin{array}{c} A_0 \\ \hline a_1^T \\ \hline A_2 \end{array} \right)$, $\left( \begin{array}{c} x_T \\ \hline x_B \end{array} \right) \leftarrow \left( \begin{array}{c} x_0 \\ \hline \chi_1 \\ \hline x_2 \end{array} \right)$
endwhile
```

Write routines

- `[ y_out ] = Mvmult_t_unb_var1( A, x, y )`
- `[ y_out ] = Mvmult_t_unb_var2( A, x, y )`

that compute $y := A^T x + y$ using the algorithms in this unit.

Some links that will come in handy:

- Spark (alternatively, open the file `LAFF-2.0xM/Spark/index.html`)
- PictureFLAME (alternatively, open the file `LAFF-2.0xM/PictureFLAME/PictureFLAME.html`)

You may want to use the following scripts to test your implementations (these should be in your directory `LAFF-2.0xM/Programming/Week04/`):

- test_Mvmult_t_unb_var1.m

- test_Mvmult_t_unb_var2.m

✓ Done/Skip

✓

Submit

Homework 4.3.1.2

2/2 points (graded)

Implementations achieve better performance (finish faster) if one accesses data consecutively in memory. Now, most scientific computing codes store matrices in "column-major order" which means that the first column of a matrix is stored consecutively in memory, then the second column and so forth. Now, this means that an algorithm that accesses a matrix by columns tends to be faster than an algorithm that accesses a matrix by rows. That, in turn, means that when one is presented with more than one algorithm, one should pick the algorithm that accesses the matrix by columns.

Our FLAME notation makes it easy to recognize algorithms that access the matrix by columns.

• For the matrix-vector multiplication of $Ax + y$, would you recommend the algorithm that uses dot products or the algorithm that uses axpy operations?

axpy ▾ ✓

• For the matrix-vector multiplication of $A^T x + y$, would you recommend the algorithm that uses dot products or the algorithm that uses axpy operations?

dot ▾ ✓

Submit