Monday 12/19/11

Last time: Pacheco Ch. 5

MPI_Bcast, MPI_Reduce, MPI_Allreduce

Today: Example of Allreduce that divides up vector on several parallel processes.

Sec. 5.5 Dot Product:

Mathematically, \( \mathbf{A}, \mathbf{y} \in \mathbb{R}^n \), column vectors

How to split them onto parallel processes?

One option: cyclic distribution

\[
\begin{bmatrix}
X_0 \\
X_1 \\
X_2 \\
\vdots
\end{bmatrix} \leftarrow p_0
\]

\[
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
\vdots
\end{bmatrix} \leftarrow p_1
\]

\[
\begin{bmatrix}
X_2 \\
X_3 \\
X_4 \\
\vdots
\end{bmatrix} \leftarrow p_2
\]

\[
\vdots
\]

\[
X = \begin{bmatrix}
X_0 \\
X_1 \\
X_2 \\
\vdots
\end{bmatrix}
\]

\[
\text{for } n = 3
\]

Note: Consecutive access to memory is the key to good performance on today's computers that have (L1, L2, L3) caches.

Caching means that the CPU assumes that it will need \( x_{i+1} \) next, if you accessed \( x_i \) now; so it pre-loads \( x_{i+1} \) ("caching").
therefore, block distribution is both easier and more efficient.

\[
\begin{bmatrix}
x_0 \\
\vdots \\
x_{l-n-1} \\
x_{l-n} \\
\end{bmatrix}
\begin{cases}
\mathbf{P}_0 \\
\mathbf{P}_1 \\
\vdots \\
\mathbf{P}_{l-1-n} \\
\end{cases}
\begin{align*}
\mathbf{X} &= \mathbf{P} \mathbf{X} \\
\mathbf{L} &= \mathbf{X} \\
\end{align*}
\]

\[\begin{align*}
\mathbf{L} &= \frac{n}{p} \\
\mathbf{L} &= \text{local - n elements of } x \in \mathbb{R}^n \text{ on each process.}
\end{align*}\]

Issues: \(x, y \in \mathbb{R}^n\) \(\Rightarrow\) we really only want to define their local portions \(\mathbf{L}_x, \mathbf{L}_y \in \mathbb{R}^{l-n}\)

Static:

\[
\begin{align*}
\text{double } \mathbf{L}_x[\mathbf{L}_n], \mathbf{L}_y[\mathbf{L}_n];
\end{align*}
\]

Dynamic:

\[
\begin{align*}
\text{double } *\mathbf{L}_x, *\mathbf{L}_y; \\
\mathbf{L}_x &= \text{malloc} (\mathbf{L}_n * \text{sizeof (double)}) \\
\text{number of bytes to be allocated}
\end{align*}
\]

\text{malloc does not initialize the memory.} \quad \Rightarrow \text{call} \text{free} \text{at end}

\text{call} \text{malloc sets it to 0 and has different syntax:}

\[
\begin{align*}
\mathbf{L}_x &= \text{calloc} (\mathbf{L}_n, \text{sizeof (double)}); \\
\rightarrow \text{check man pages}
\end{align*}
\]

\text{also notice any header files to be included?}
Then use either of these vectors as $l_x \ [ l_y ]$
for dynamically allocated vectors, remember to free them:

$$\text{free} ( l_x )$$

**Problem:** $x, y \in \mathbb{R}^n$, want $d = x^T y = \sum_{i=0}^{n-1} x_i y_i$

**Serial code:**

```c
double serial_dot ( double *x, double *y, int n ) {
    double d = 0;
    for ( i = 0; i < n; i++ ) {
        d += x[i] * y[i];
    }
    return d;
}
```

```c
double parallel_dot ( double *l_x, double *l_y, int l_n ) {
    double l_d; l_d = serial_dot ( l_x, l_y, l_n );
    d = MPI_Allreduce ( &l_d, &d, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD );
    return d;
}
```
Sec. 5.7 Gather, Scatter:

Have \( x \in \mathbb{R}^m \) locally, want to assemble \( x \) on Process 0 for text output.

```c
MPI_Gather(
    void *send_data,
    int send_count,
    MPI_Datatype send_type,
    void *recv_data,
    int recv_count,
    MPI_Datatype recv_type,
    MPI_Comm comm,
    MPI_Comm comm)
```

```
MPI_Gather( 
    l_x, l_n, MPI_DOUBLE,
    x, [l_n], MPI_DOUBLE, 0,
    MPI_COMM_WORLD )
```

Notes:
- `recv_cnt` = number of elements from each of the processes here: \( l_n \) (and not \( n \), despite \( x \in \mathbb{R}^m \))
- This is a collective communication \( \Rightarrow \) All processes must execute this command \( \Rightarrow \) \( x \) must be defined on all processes, even though it is used and valild on the root process \( \Rightarrow \) only allocate memory for \( x \) on Process 0:
  ```c
double *x;
if (id == 0) 
  x = malloc(n * sizeof(double))
```
MPI_Scatter is inverse operation of Gather =>

    calling sequence same with reversed roles of send, recv.

What if you want to assemble \( l \times x \) into \( x \) and

have it available on all processes?

Option 1: MPI_Gather to P O, as above.

MPI_Bcast \((x, x, MPI_DOUBLE, 0, MPI_COMM_WORLD)\)

Option 2:

MPI_Allgather \(\rightarrow\) just like Gather, but without

root process, since result goes to all processes.

Next time: How to define and use matrices,

not following Pacheco, since we want to interface

with other software (BLAS, LAPACK) and take

full control of memory allocation ourselves.
Lab 3 tomorrow = HW 4 and 5:

- Make utility: Unix command "make" to compile,
- link, etc., of all your code to get an executable.

- Download files with HW 4.

How to compile and link if your code is contained in 2 files:

```
make file1.c -c
make file2.c -c
```

```
/* -c = compile only.  
output file name file1.o, file2.o automatically */
make file1.o file2.o -o executable-name
```