L4 QUINN p108 PERFORMANCE

BENCHMARKING PARALLEL

(wall clock time.... btw execution, termination)

we ignore time spent initiating MPI processes, establishig communications, performing I/O.

we measure the efficiency of the parallel program, against its sequential counterpart.

MPI_Wtime returns #seconds that have elapsed since some point in time in the past

MPI_Wtick. returns the precision of the result returned by MPI_Wtime.

headers:

```
double MPI_Wtime (void)
double MPI_Wtick (void)
```

we benchmark a section of the code by putting a pair of calls to MPI_Wtime before and after the section.

the difference btw the 2 values is the number of seconds elapsed.

pb: MPI processes executing on different processors may begin execution in different points in time, seconds apart.

this can throw off timings

sol: introduce a barrier synchronization before the first call to MPI_Wtime.

no process can proceed beyond a barrier until all processes have reached it.

a barrier ensures that all processes will enter the measured section of the code at the same time.

header:

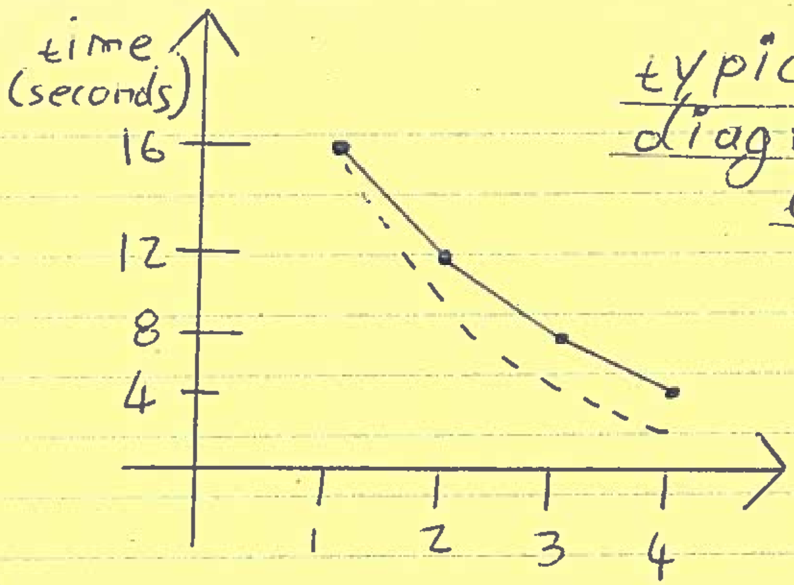
```
int  MPI_Barrier (MPI_Comm comm)
```

the arg. to MPI_Barrier indicates the communicator participating in the barrier.

main: double elapsed_time;

```
    MPI_Init ...
    MPI_Barrier(MPI_COMM_WORLD);
    elapsed_time = - MPI_Wtime();
          ⋮
  elapsed_time += MPI_Wtime();
```

typical diagram looks like this:



SOLID LINE: as we add proc^s, execution time decreases

DASHED LINE: "perfect" speed improvement ~> p proc^s execute the program p times as fast

(2 proc^s take half the time
3 proc^s take 1/3 of the time)

typical reason for disparity btw dashed/solid line: reduction operations

(overhead not incurred in sequential program)

#proc^s ↑ ~> this overhead grows too.