SIF8041 OPERATIVSYSTEMER OG DATABASER

våren 2001

ØVING 6: MPI (OS)

INNLEVERINGSFRIST : Mandag 30. april , 2001

GRUPPESTØRRELSE : 2 -3. Lever svarene på papir i boks 2 eta. E-blokka.

Generelt

Dere skal prøve å lage et lite MPI-program som leser igjennom bildefila "elster.xbm" linje for linje. Programmet skal lese fila og telle opp antall svarte og hvite tegn (pixler) i fila ved å tolke octal bmp encoding verdier. Dette er beskrevet i den amerikanske oppgaveteksten under.

"elster.xbm"-fila kan lastes ned fra web her: http://www.idi.ntnu.no/emner/sif8041/elster.xbm

Poenget med denne siste øvinga er at dere skal se litt på og prøve ut MPI. Dessverre kommer denne øvinga av ulike årsaker for sent i semesteret, så det er ikke noe absolutt krav at alt skal fungerer for å få godkjent. Alle grupper som har forsøkt å få til noe som fungerer får godkjent!

Hvordan kjøre MPI

Det blir reservert maskiner i nettet som dere kan bruke. MPI kan kun brukes fra maskiner på VM-brakka. Info. om dette kommer mandag. For å kompilere/kjøre programmet må dere bruke 'mpirun'. Info. om hvilke parametre dere bør bruke kommer også (spør ellers und.ass'ene).

Lenker til info. om MPI:

http://www-unix.mcs.anl.gov/mpi/tutorial/mpiintro/index.htm

http://www-unix.mcs.anl.gov/mpi/tutorial/

Som en myk start kan dere evt. prøve 'Hello World' programmet som forklart her:

http://www.ece.utexas.edu/projects/courses/fall_00/ee360p/proj3/proj3-1.html

---

Amerikansk oppgavetekst

You will be computing some statistics of an image. These statistics include counting the no. of white vs. black pixes in each line and/or the number of octal pixel groups within some threshold ranges.

You will be working on a pretty small image (48-by-48 octal bmp encoding "elster.xbm"), so

that we do not crash the systems.

Your task will be to

1. Distribute this image among the available MPI processes MPI_INIT gives you on the respective clusters, e.g. on a 8-process cluster you will put 48/8 = 6 lines on each cluster.
2. Each process should then **in parallel** process each line as stated in the next section.
3. The resulting 48-entry global vector should then be gathered on process 0
4. and displayed as a histogram.

LINE PROCESSING

Process each line using the octal encoding

1. Set ranges for "white" (close to 0x00), "black" (close to 0xff) and "gray" (range inbetween).
2. Count for each line the number of "white", "gray" and "black" pixel-octets there are in the image, and
3. Store the result in a local array
4. Distribute the result back to process 0.

EXTRA CREDIT: Process the underlying pixels

Instead of looking at "octets", find a way to decompose the "xbm"-octets into a bitmap and then count in parallel how many white pixels the image has per line.

Timing

Add in MPI_Wtime to the program.

MPI_Wtime() returns a double-precision floating-point number of seconds since some arbitrary point in the past. Thie point is guaranteed not to change during the lifetiem of the process. Thus, a time interval can be measured by calling this routine at the begining and the end of the program segment and subtracting the values returned.

MPI_Wtime() should be called at the begining of your program and after each point you want to time where you can print something like:

```
"Timestamp  ww from Process nn"
```

where ww = difference between first and second MPI_Wtime() returned.

Run your program on each of the 3 (4?) Linux cluster configurations available for MPI (INFO. KOMMER!).

- List a table of the 3 timing results (with timestamps from each processor on each run.)
- Comment on whether your results agree with your expectations -- why and why not?
- How would you expect the results to be different if a truely large image was processed on a decicated cluster?

TO HAND IN:

- Hard-copy of all MPI code (incl. comments!!)
- Hard-copy of results from PART 4 above.
- Online-submission of sources and makefile