Prof. Dr. R. Loogen, M. Dieterle
Fachbereich Mathematik und Informatik
Hans-Meerwein-Straße
D-35032 Marburg

12.Dezember 2008

Philipps Universität Marburg

Übungen zu „Parallele Programmierung“, WS 2008/09

Nr. 8, Abgabe der Aufgaben: 16.Januar 2009 in der Übung

Frohe Weihnachten und alles Gute in 2009

Benutzung von OpenMPI am Netz des Fachbereichs

OpenMPI ist am Fachbereich unter Linux installiert. Zur Benutzung fügen Sie */app/lang/parallel/bin* zu Ihrem Suchpfad hinzu. Zusätzlich können Sie den `MANPATH` noch um */app/lang/parallel/openmpi-1.2.8/share/man* erweitern, um mit `man mpi...` die Dokumentation der einzelnen Befehle zu bekommen. Sie können die Pfade permanent erweitern, je nach Shell:

bash: in der Datei *~/.bashrc* durch die Zeile:

```
export PATH=${PATH}:/app/lang/parallel/bin
export MANPATH=${MANPATH}:/app/lang/parallel/openmpi-1.2.8/share/man
```

tcsh: in der Datei *~/.tcshrc* durch die Zeile:

```
setenv PATH ${PATH}:/app/lang/parallel/bin
setenv MANPATH ${MANPATH}:/app/lang/parallel/openmpi-1.2.8/share/man
```

Benutzen Sie `mpicc` zum Kompilieren ihrer auf C basierenden MPI-Programme. `mpicc` können Sie weitere Optionen (z.B. `-o <progName> -O2` ...) übergeben, die an `gcc` weitergeleitet werden. Starten Sie die Programme mit `mpirun [options] progName`, wobei mit `-np <n>` die Anzahl `n` der Prozesse und `-hosts <list>` eine kommaseparierte Liste an benutzbaren Rechnern angegeben wird. Alternativ kann die Rechnerauswahl auch mit der Option `-hostfile <file>` vorgenommen werden, wobei die Rechner dann Zeilenweise in der Textdatei `file` stehen.
Benutzbare Rechner (in den PC-Arbeitsräumen) sind aktuell:

```
annaba boende kananga likasi owando sakania tanga wase
```

Aufgaben

8.1 Kollektive Reduktion in MPI

6 Punkte

Der MPI-Standard spezifiziert die Reduktion eines über mehrere Prozessoren verteilten Feldes mit einer assoziativen (und evtl. kommutativen) binären Operation (etwa der Summation zweier Elemente) als kollektive Operation MPI_Reduce. Wird das Ergebnis auf alle Prozessoren verteilt, so handelt es sich um eine MPI_Allreduce-Operation.

```
MPI_Allreduce(void* sendbuf,void* recvbuf, int count,
              MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
```

Bitte wenden!

Gleichzeitig können mehrere Elemente pro Prozessor kombiniert werden. Das an MPI_AllReduce übergebene Array `sendbuf` der Länge `count` wird dann elementweise kombiniert.

Die Implementierung von MPI_AllReduce kann als Kombination von Reduce (als Binärbaum) und nachfolgendem Broadcast oder mit einem Butterfly-Schema erfolgen. Falls `count` groß ist, kann außerdem im Butterfly-Schema das zu bearbeitende Array sukzessiv halbiert werden, um die Datenmenge zu reduzieren, wobei dann nach der Reduktion die Teilergebnisse wieder verteilt werden müssen.[1]

Wir vergleichen die genannten Alternativen mit einem *Kostenmodell*, welches den Zeitbedarf für Kommunikation und binäre Operation berücksichtigt. Eine einzelne binäre Operation benötige die Zeit γ, p sei die Zahl der beteiligten Prozessoren und n die Arraylänge (Paramter `count`). Die Zeit zur Kommunikation von m Daten von einem Prozessor zum anderen sei mit $\alpha + m \cdot \beta$ abgeschätzt (α ist konstanter Anteil pro Nachricht, β der Zeitbedarf pro Datenelement).

(a) Drücken Sie den Zeitbedarf der drei Implementierungen als Funktion der genannten Parameter aus. Zur Vereinfachung seien p und `count` Zweierpotenzen.

(b) Formulieren Sie mit Hilfe der Parameter im Kostenmodell Bedingungen, unter denen jeweils eine bestimmte Alternative optimal ist.

8.2 K-Means mit OpenMP(I)

12 Punkte

K-Means ist ein einfacher Algorithmus zum Clustern von m-dimensionalen Datensätzen. Auf *Wikipedia* findet sich folgende kurze Beschreibung des Verfahrens:

i) Bestimmung von k Clusterzentren.

ii) Jedes Objekt wird demjenigen Cluster zugeordnet, dessen Zentrum ihm am nähesten liegt.

iii) Für jedes Cluster wird das Zentrum neu berechnet.

iv) Basierend auf den neu berechneten Zentren werden die Objekte wieder wie in Schritt ii) auf die Cluster verteilt.

v) Wurde bei der Neuverteilung mindestens ein Objekt einem anderen Cluster zugeordnet, dann wiederhole den Algorithmus beginnend mit Schritt iii).

In Schritt i) sollen k zufällig gewählte Datensätze der Eingabedaten als Clusterzentren bestimmt werden. In Schritt iii) wird der Mittelpunkt der jeweils einem Cluster zugeordneten Datensätze als neues Zentrum gewählt. Generell ist vom euklidischen Abstand als Distanzmaß auszugehen.

(a) Implementieren Sie den Algorithmus in *C*. Die Problemgröße n, die Clusteranzahl k und die Anzahl m der Datendimensionen sollen parametrisierbar sein. Erzeugen Sie als Eingabe zufällig n m-dimensionale Datensätze.

(b) Parallelisieren Sie Ihr Programm mit *OpenMP*.

(c) Schreiben Sie eine Version, in der die Parallelisierung mit *MPI* erfolgt.

(d) Vergleichen Sie Laufzeiten der drei Versionen und diskutieren Sie kurz die Ergebnisse.

Benutzen Sie parallele Konstrukte wo möglich, jedoch nicht zur Berechnung der Distanz zweier Vektoren.

[1] Mehr zu diesem Thema: Rolf Rabenseifner: Optimization of Collective Reduction Operations. *http://www.hlrs.de/people/rabenseifner/publ/publications.html*