Übungen zur Vorlesung

Wissenschaftliches Rechnen – Paralleles Höchstleistungsrechnen

Prof. Dr. C. Engwer, S. Westerheide
http://wwwmath.uni-muenster.de/num/Vorlesungen/WissenschaftlichesRechnen_SS11/

Abgabe 30. 5. 2011. Abg. der Programmieraufgaben per Email an `sebastian.westerheide@uni-muenster.de`, schriftliche Abgabe Dienstags in der Vorlesung.

- Alle Programmierübungen müssen per Email und in ausgedruckter Form abgegeben werden.
- Achten sie darauf, ihr Programm ordentlich zu formatieren und gut zu kommentieren.
  In Zukunft wird die Form mit in die Bewertung eingehen.

Auf der Vorlesungsseite haben wir Ihnen "Hinweise zur Verwendung von MPI im Computerpool" zusammengestellt. Lesen Sie diese vor der Bearbeitung der nächsten Aufgabe und führen Sie den Punkt "System einrichten" durch. Wie im Punkt "Erste Schritte mit MPI" beschrieben, können Sie anschließend testen, ob alles ordnungsgemäß funktioniert.

Übung 1 MPI Punkt-zu-Punkt Kommunikation

Laden Sie sich das Programmskelett "MPI Ring Kommunikation" von der Vorlesungsseite und ergänzen Sie es um den fehlenden Code an den durch "`...`" markierten Stellen. Das fertige Programm soll eine Nachricht für eine vom Benutzer festzulegende Anzahl von Durchläufen durch einen Ring von Prozessen schicken, wobei die Nachricht jeweils den noch durchzuführenden Durchläufen entspricht.

- Erweitern Sie das Programm derart, dass der Prozess mit dem Rang 0 die mittlere Zeit für das verschicken einer Nachricht von Prozess zu Prozess berechnet und ausgibt. Benutzen Sie dafür die Methode `MPI_Wtime()`, die in der MPI Kurzeinführung aus der Vorlesung beschrieben wird.
- Führen Sie das Programm mehrmals für eine steigende Anzahl an Prozessen auf einem einzelnen Computer aus. Plotten Sie die ermittelten Zeiten gegen die Anzahl der Prozesse.
- Starten Sie das Programm nun wiederholt für eine steigende Anzahl an Prozessen, die auf verschiedenen Computern laufen. Lassen Sie sich dafür zuvor über das bereitgestellte Skript `gen-machines` eine Liste von Computern generieren. Plotten Sie wieder die ermittelten Zeiten gegen die Anzahl der Prozesse.
- Diskutieren Sie die beiden Plots sowohl im Einzelnen als auch im Vergleich. Wie lassen sich ihre Beobachtungen erklären?

Hinweis: Die beiden Plots gehören zum schriftlichen Teil der Abgabe dazu.

6 Punkte

In der nächsten Aufgabe betrachten wir das Problem, dass jeder Prozess ein individuelles Datum an alle anderen Prozesse senden muss, so dass am Ende jeder Prozess alle Daten kennt. Genauer betrachten wir zwei verschiedene Algorithmen, mit denen sich dieses Problem lösen lässt.

Algorithmus 1 (Alle an alle austeilen im Ring)

Alle Prozesse schieben das Datum, das sie zuletzt erhalten haben, zyklisch im Ring weiter (Fig. 1). Es wird asynchrone Kommunikation verwendet, um das Schieben parallel durchzuführen.

```
parallel all-to-all-ring
{
    const int P;
    process Π[int p ∈ {0, ..., P − 1}]
    {
        void all_to_all_broadcast(msg m[P])
        {
            int i,
                item = p;                          // M[item] wird geschickt/empfangen
            msgid ids, idr;
            for (i = 0; i < P; i++)                // P − 1 mal schieben
            {
                ids =asend(Π_{(p+1)%P},m[item]);   // an nächsten
                item = (item + P − 1)%P;           // zähle rückwärts
                idr =arecv(Π_{(p+P−1)%P},m[item]);
                while(!success(ids));
                while(!success(idr));
            }
        }
        ...
        m[p] ="'Das ist von p!"';
        all_to_all_broadcast(m);
        ...
    }
}
```

Algorithmus 2 (Alle an alle austeilen im Hypercube)

Die Prozesse senden sich die Daten über den sogenannten *Dimensionsaustausch* im Hypercube: Für $d = 1$ tauschen die beiden Prozesse ihre Daten trivial aus (Fig. 2). Für $d = 2$, also vier Prozesse, tauschen erst 00 und 01 bzw 10 und 11 ihre Daten aus, dann tauschen 00 und 10 bzw 01 und 11 jeweils zwei Informationen aus (Fig. 3). Allgemein kennt nach dem i-ten Schritt ein Prozess p die Daten aller Prozesse q, die sich genau in den letzten i Bitstellen von ihm unterscheiden.

```
parallel all-to-all-hypercube
{
    const int d, P = 2^d;
    process Π[int p ∈ {0, ..., P − 1}]
    {
        void all_to_all_broadcast(msg m[P]) {
            int i, mask = 2^d − 1, q;
            for (i = 0; i < d; i++) {
                q = p ∧ 2^i;
                if (p < q) {                       // wer zuerst?
                    send(Π_q,m[p&mask],...,m[p&mask + 2^i − 1]);
                    recv(Π_q,m[q&mask],...,m[q&mask + 2^i − 1]);
                }
                else {
                    recv(Π_q,m[q&mask],...,m[q&mask + 2^i − 1]);
                    send(Π_q,m[p&mask],...,m[p&mask + 2^i − 1]);
                }
                mask = mask ∧ 2^i;
            }
        }
        ...
        m[p] ="'Das ist von p!"';
        all_to_all_broadcast(m);
        ...
    }
}
```

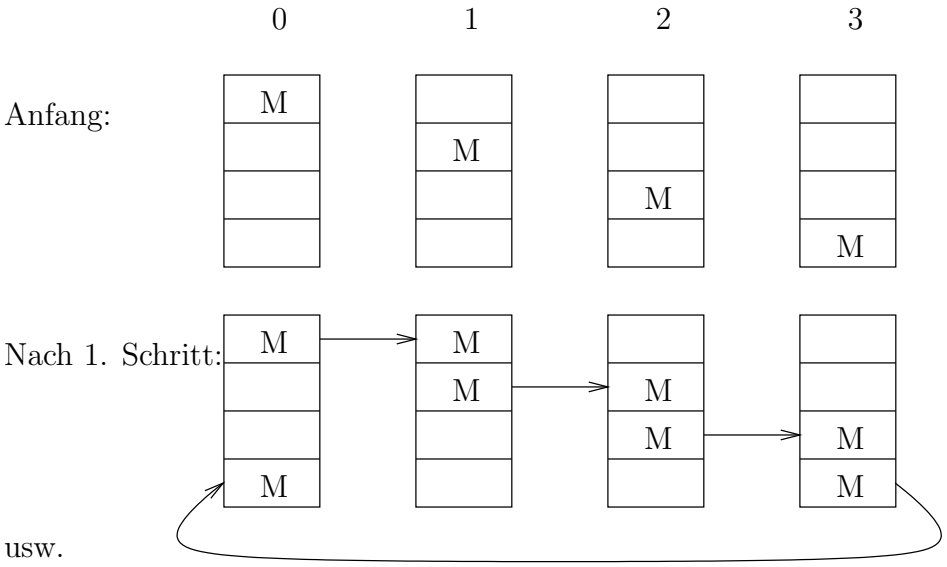

Figure 1: Alle an alle austeilen im Ring

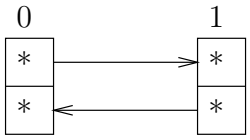

Figure 2: Alle an alle austeilen im Hypercube ($d = 1$)

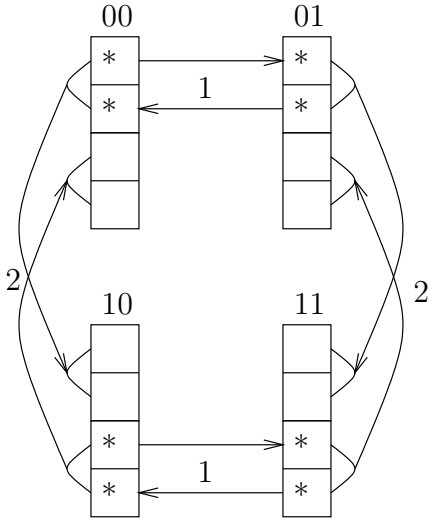

Figure 3: Alle an alle austeilen im Hypercube ($d = 2$)

Übung 2 Globale Kommunikation - alle an alle austeilen

Betrachten Sie folgende Kombinationen aus Algorithmus, Routing-Strategie und der *physikalischen* Netzwerktopologie, die den Prozessen zugrundeliegt:

a) Algorithmus 1 auf einem Ring mit store-and-forward routing

b) Algorithmus 2 auf einem Hypercube mit store-and-forward routing

c) Algorithmus 2 auf einem Ring mit cut-through routing

- Bestimmen Sie für jede dieser Kombinationen die Laufzeit des Algorithmus' und begründen Sie detailliert, wie Sie zu diesem Ergebnis gekommen sind. Gehen Sie dabei wie in der Vorlesung davon aus, dass das Versenden jeder einzelnen Nachricht in einem Paket der Länge n Byte geschieht und benutzen Sie die entsprechenden Notationen t_s, t_h, t_b.

  Bedenken Sie bitte, dass eine physikalische Kommunikationsleitung auch beim cut-through routing zu jedem Zeitpunkt höchstens einmal belegt sein kann. Bei der Berechnung der Laufzeit von Kombination c) müssen sie daher die Größe von n berücksichtigen.

- Erläutern Sie, welche der beiden Kombinationen a) und c) bei einem sehr großen n zu bevorzugen ist und welche bei einem sehr kleinen n (z.B. ein Byte).

6 Punkte