Projektaufgabe 1

WPF-Modul *Cluster Computing*

Parallel Sorting by Regular Sampling (PSRS)

Es handelt sich hierbei um einen parallelen Sortieralgorithmus der einen verträglichen Kommunikationsaufwand und gute Balance-Eigenschaften aufweist.
Der Algorithmus ist für diverse MIMD-Architekturen geeignet.

Der Algorithmus gliedert sich in vier Phasen:

Phase 1: Sequenzielles Sortieren lokaler Daten

Ausgangspunkt ist ein Feld von zufällig erzeugten Integer-Zahlen vom Umfang ***n*** und ein Parallelrechnersystem mit ***N*** Prozessoren (bzw. Knoten). Ist ***n*** ein Vielfaches von ***N***, erhält jeder Prozessor einen Block von ***n/N*** Elemente, sonst erhalten einzelne Prozessoren jeweils ein Element mehr bzw. sind Dummy-Elemente einzufügen. Die für jeden Prozessor nunmehr vorliegenden Blöcke werden mit einem sequenziellen, möglichst schnellen Algorithmus sortiert.
Anschließend selektieren die Prozessoren parallel jeder genau ***N*** Elemente an den lokalen Positionen

$$1,\ w + 1,\ 2\,w + 1,\ \ldots,\ (N - 1)\,w + 1 \quad \text{mit} \quad w = n / N^2$$

um eine repräsentative Auswahl des lokal sortierten Blockes zu erhalten. Die N^2 ausgewählten Daten, ***N*** von jedem der ***N*** Prozessoren, sind eine reguläre Auswahl des gesamten Datenarrays.

Phase 2: Balancierte Lade-Phase

Ein Prozessor wird festgelegt, die lokalen regulären Auswahlen zu sammeln und zu sortieren. Dann werden ***N-1*** Pivots aus der sortierten regulären Auswahl selektiert, und zwar mit den Indizes

$$N + t,\ 2N + t,\ \ldots,\ (N-1)N + t \quad \text{mit} \quad t = N/2 \text{ (ganzzahliger Teil).}$$

Jeder Prozessor erhält eine Kopie der Pivots und erzeugt ***N*** Blöcke aus seiner lokalen sortierten Liste.

Phase 3: Datenaustausch

Prozessor ***i*** erhält jeweils den Block ***i*** von allen Prozessoren ***(i = 1, …, N).***

Phase 4: Paralleles Mischen

Die Prozessoren mischen parallel ihre N Blöcke. Durch Zusammensetzen der erzeugten Listen erhält man die sortierte Folge.

Allgemein:

Falls kein Element doppelt vorkommt, dann garantiert PSRS, dass die Arbeit unter den Prozessoren so verteilt wird, dass keiner mehr als doppelt soviel wie die anderen zu tun hat, unabhängig von der anfangs vorliegenden Datenverteilung.

Die Zeitkomplexität des Algorithmus beträgt für den Fall $n >= N^3$: $O[(n/N)\log n]$.

Aufgabenstellung:

Setzen Sie den vorgegebenen Algorithmus in ein C-Programm unter Einsatz von MPI um. Dabei sollen folgende Punkte Berücksichtigung finden:

- die Zahlen sind von einem Prozessor zu erzeugen und an die beteiligten Prozessoren zu verteilen
- am Ende soll ein Prozessor das sortierte Feld zu Kontrollzwecken ausgeben können
- das Programm soll mit unterschiedlich großen Datensätzen getestet werden, z.B. 20000, 40000, 80000 Zahlen
- das Programm soll mit unterschiedlich vielen Prozessoren getestet werden.

Analysieren Sie das Laufzeitverhalten Ihres Programms, wobei insbesondere Aussagen zum Speedup von Interesse sind.
Des Weiteren sollen Aussagen darüber getroffen werden, welchen zeitlichen Anteil das anfängliche sequenzielle Sortieren an der Gesamtlaufzeit des Programms hat und wie groß der Kommunikationsoverhead ist.