Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Prof. Dr. T. Ludwig, Dr. Jannek Squar

Übungsblatt 06 zur Vorlesung
Hochleistungsrechnen
im WiSe 2026/2027
Abgabe: 21.11.2026, 23:59

Dieses Übungsblatt soll Ihnen die Möglichkeit geben, Ihre ersten Schritte in der Programmierung und Fehlersuche mit MPI zu machen. Die erworbenen Fertigkeiten werden auf späteren Übungsblätter für komplexere Aufgaben benötigt.

1 Das erste MPI-Programm (100 Punkte)

Erweitern Sie das gegebene Programm `timempi.c` um MPI-Kommunikation damit es eine ähnliche Ausgabe erzeugt wie das parallele Skript von Übungsblatt 03. Dabei sind folgende Vorgaben zu beachten:

- Bei n Prozessen sollen die Prozesse mit den Rängen 0 bis n-2 die folgende Zeichenkette bei sich erzeugen: `[Rang] HOSTNAME // TIMESTAMP` und per MPI an den Prozess mit dem letzten Rang (n-1) senden, welcher die komplette Ausgabe übernimmt. Der letzte Prozess generiert selbst keine eigene Zeichenkette (siehe Beispiel).
- Diese Ausgabe soll nach Rang der Prozesse geordnet erfolgen.
- Die Prozesse sollen alle erst beenden, wenn die Ausgabe aller Prozesse komplett erfolgt ist. Das Programm ist falsch, wenn ein Prozess zu früh beenden könnte! *Hinweis:* Aufgrund von Bufferung kann es aber sein, dass trotz korrekter Synchronisation die Ausgabe nicht getrennt ausgegeben wird. Entscheidend ist die korrekte Trennung in der Implementierung.
- Sobald der letzte Rang alle Ausgaben ausgegeben hat, müssen alle Prozesse in beliebiger Reihenfolge terminieren. Direkt vor dem Beenden soll jeder (auch der letzte) Prozess selbst einen Text ausgeben: „[Rang] beendet jetzt!“
- Das Programm muss mit beliebig vielen Prozessen lauffähig sein. Überlegen Sie dabei eine Handhabung für 1 Prozess.
- Das Programm muss ohne Warnungen und Fehler kompilieren. Entfernen Sie keine Flags aus dem Makefile.

Die Ausgabe mit 3 Prozessen könnte wie folgt aussehen:

```
[0] west1 // 2026-11-20 23:15:57.968558
[1] west2 // 2026-11-20 23:15:56.968557
[1] beendet jetzt!
[2] beendet jetzt!
[0] beendet jetzt!
```

Beachten Sie, dass Prozess 3 die eigene Zeichenkette nicht erzeugt und nur die Ausgaben der anderen Prozesse sammelt und ausgibt. In der allgemeinen Form gilt dies für den Prozess mit dem letzten Rang (n-1).

Hinweis: Binden Sie den Header `mpi.h` ein und benutzen Sie den Compiler `mpicc`.

2 Ergebnisse sammeln im MPI-Programm (50 Punkte)

Erweitern Sie Ihr MPI-Programm um folgende Funktion:

- Direkt nach der Ausgabe der empfangenen Daten soll der Prozess mit dem letzten Rang noch den kleinsten Mikrosekunden-Anteil sowie die größte Differenz der Mikrosekundenwerte zwischen allen Prozessen ausgeben. Da der letzte Prozess selbst keine Zeichenkette generieren sollte, ist er nur für die Bestimmung zuständig und steuert selbst keinen Mikrosekunden-Anteil bei.
- Hierfür bieten sich kollektive Operationen an.
- Erst nach der Ausgabe der größten Differenz sollen alle Prozesse beginnen, in die Endphase zu gehen. Beachten Sie den Hinweis zur Bufferung des Terminal-Outputs.
- Überlegen Sie, welcher Umgang bei Ausführung mit nur einem Prozess sinnvoll ist.

Die Ausgabe könnte dann wie folgt aussehen:

```
[0] west1 // 2026-11-20 23:15:57.968558
[1] west2 // 2026-11-20 23:15:56.968557
[2] Kleinster us-Anteil: 968557
[2] Größte Differenz: 1
[2] beendet jetzt!
[0] beendet jetzt!
[1] beendet jetzt!
```

Hinweis: Da Terminal-Output gebuffert ist, kann es bei der Ausgabe zu unerwartetem Verhalten bzgl des Timings kommen, wenn mehrere Prozesse involviert sind. Wichtig ist nur, dass im Code die Prozesse korrekt synchronisiert werden.

3 Paralleles Debugging mit DDT (60 Punkte)

In dieser Aufgabe sollen Sie sich mit dem parallelen Debugger DDT vertraut machen. Verwenden Sie dazu Ihr gerade geschriebenes paralleles Programm.

Unter folgendem Link finden Sie eine kleine Einführung **und Hinweise zum Starten von DDT**:

```
https://wr.informatik.uni-hamburg.de/teaching/ressourcen/debugging#ddt
```

Hinweis: Für die Benutzung von DDT benötigen Sie X-Forwarding. Alternativ können Sie X2Go nutzen, welches üblicherweise eine höhere Leistung erreicht.

Die Lizenz auf dem Cluster ist für 16 gleichzeitige Prozesse gültig. Bedenken Sie bei Ihren Tests, dass sich alle Benutzer diese Anzahl teilen! Das Aufrufen von DDT ist nur vom Login-Knoten möglich, wobei ein Programm durchaus auf mehreren Knoten analysiert werden kann.

Machen Sie sich ein wenig mit DDT vertraut, indem Sie Ihr Programm mit vier Prozessen debuggen. Dokumentieren Sie folgende Punkte mit jeweils einer kurzen Beschreibung und einem Screenshot; markieren Sie die relevanten Stellen der Screenshots:

1. Wie kann man in DDT die Programmparameter angeben? Geben Sie zwei Wege an.
2. Setzen Sie in einer Zeile einen Breakpoint. Welche Step-Möglichkeiten gibt es und wie unterscheiden sich diese?
3. Schauen Sie sich die Werte der Variable an, in der Sie den Rang des aktuellen Prozesses gespeichert haben. Erklären Sie die Linien in der Darstellung und vergleichen Sie die Werte aller Prozesse mit Hilfe des Kontextmenüs.
4. Machen Sie sich mit der Funktion des Evaluate-Fensters in der rechten unteren Ecke vertraut.
5. In der oberen Leiste finden Sie eine Übersicht aller Prozesse und Threads Ihres Programmes. Wechseln Sie zwischen den einzelnen Prozessen und beobachten Sie das Evaluate-Fenster.
6. Erweitern Sie Ihr Programm um ein Array und initialisieren Sie es mit beliebigen Zahlenwerten. Lassen Sie sich die Werte anzeigen. Visualisieren Sie diese Array in DDT. (**Hinweis:** Das Array wird nur für diese Aufgabe benötigt und soll im abgegebenen Programmen `timempi` **nicht** enthalten sein.)

Abgabe

Als Abgabe erwarten wir ein gemäß den Vorgaben benanntes komprimiertes Archiv, das ein gemäß den Vorgaben benanntes Verzeichnis mit folgendem Inhalt enthält:

- Den Sourcecode und Makefile zum C-Programm `timempi.c`.
- Ein PDF mit den Antworten und Screenshots.

Achten Sie darauf, dass Ihre Programme ohne Warnungen oder Fehler bauen und Valgrind keine Probleme anzeigt. Packen Sie ein komprimiertes Archiv (`.tar.gz`) aus dem sauberen Verzeichnis (**ohne Binärdateien oder versteckte Dateien**). Benennen Sie das Archiv nach den Nachnamen der Gruppenmitglieder (z. B. `MustermannMusterfrau.tar.gz`).
Ein Mitglied Ihrer Gruppe legt das Archiv dann in

```
/home/USERNAME/HR-Abgaben-2627/Blatt-06
```

ab und schickt nach erfolgter Abgabe eine E-Mail an `jannek.squar@uni-hamburg.de`, in der Sie einfach den absoluten Pfad zu Ihrem Archiv angeben:

```
/home/USERNAME/HR-Abgaben-2627/Blatt-06/MustermannMusterfrau.tar.gz.
```

Setzen Sie Ihre Gruppenmitglieder auf CC.