Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Dr. J. Kunkel, Hans Ole Hatzel

Übungsblatt 1 zum Praktikum
Paralleles Programmieren
im SoSe 2015
Abgabe: 22.04.2015

In diesem Übungsblatt wird die Benutzung des Clusters bzw. Linux nochmal erwähnt, sofern Sie sich damit noch nicht auseinander gesetzt haben. Erste Grundlagen der Programmierung mit C werden behandelt. Schließlich sollen kleine Programme mit MPI erstellt werden und somit erste parallele Anwendungen.
Obligatorische Aufgaben sind mit *OBLIGATORISCH* gekennzeichnet. Diese Übungen sind in Einzelarbeit zu bearbeiten und abzugeben, gerne können Sie natürlich gemeinsam die notwendigen Kentnisse erarbeiten. Ziel ist es dennoch, dass Sie etwas lernen. Diese Aufgaben sind mit einer Punktzahl versehen welche der erwarteten Bearbeitungszeit in Minuten entspricht. 50% der Punkte müssen auf jedem Übungsblatt erzielt werden, welche Aufgaben bearbeitet werden obligt euch. Schicken Sie bis zum Abgabedatum eine Email mit Ihrem Namen und den abzugebenden Artefakten in einem TAR-Archiv an: `papo-abgabe@wr.informatik.uni-hamburg.de`[1].
Bitte schickt keine vom Compiler erstellten Dateien mit!
Da dieser Übungszettel die Grundlagen legt, sollten Sie alle Aufgaben bearbeiten.
Als kleiner Hinweis: ein stures Abarbeiten der Aufgaben ist durchaus möglich, gerne könnt ihr aber etwas mehr lesen und nicht nur den wenigsten Aufwand betreiben. Aufgetretene Probleme bzw. Lösungsvorschläge werden kurz im Praktikum besprochen, aber ihr solltet durchaus eigene Erfahrungen mit der Programmierung von C und MPI sammeln.
Sollten Probleme auftauchen, wenden Sie sich bitte an die Mailingliste der Veranstaltung:

`papo-15@wr.informatik.uni-hamburg.de`

1 Cluster-Kennung (30 P, OBLIGATORISCH)

Im Folgenden sollen Sie sich auf dem Cluster einloggen und das Navigieren in einer Shell üben.
Bitte melden Sie sich auf dem Cluster an und machen Sie sich ein wenig mit den grundlegenden Linux-Befehlen vertraut. Informationen dazu finden Sie in unserem Wiki in der Sektion Cluster:

`http://wr.informatik.uni-hamburg.de/teaching/ressourcen/start`

Beachten Sie insbesondere den Link „Beginners' Guide".
Die folgenden konkrete Aufgaben haben Sie zu bewältigen:

1. Installieren Sie Putty oder ein vergleichbaren SSH-Client (siehe Beginners' Guide)
2. *Einloggen*

   Loggen Sie sich auf dem Cluster mit ihrem Benutzer und Passwort ein.
3. *Bewegen im CLI (Command Line Interface)*
   a) Machen Sie sich mit der Verwendung von Manual-Pages vertraut: `$> man man`
   b) Lassen Sie sich den Namen des aktuellen Arbeitsverzeichnisses anzeigen: `$> man pwd`
   c) Lassen Sie sich den Inhalt Ihres Homeverzeichnisses anzeigen: `$> man ls`
   d) Erzeugen Sie ein neues Verzeichnis mit dem Namen `testdir`: `$> man mkdir`
   e) Ändern Sie das Arbeitsverzeichnis in das neue Verzeichnis: `$> cd testdir`
   f) Lassen Sie sich noch einmal das aktuelle Arbeitsverzeichnis anzeigen.
   g) Erzeugen Sie eine leere Datei mit dem Namen `testfile`: `$> man touch`

[1] Die Erstellung des TAR-Archivs lernen Sie bei der Bearbeitung dieses Übungszettels.

h) Benennen Sie die neue Datei um in `testfile2`: `$> man mv`

i) Kopieren Sie die umbenannte Datei in `testfile3`: `$> man cp`

j) Löschen Sie die Datei `testfile2`: `$> man rm`

Frage: Warum trägt die Manual-Page zu `cd` den Titel „POSIX Programmer's Manual“ wohingegen die anderen Manual-Pages aus dieser Aufgabe den Titel „User Commands“ haben? (Tipp: Sehen sie sich die Manual-Page zu `cd` genau an.)

4. *Packen eines Archiv*

a) Erstellen Sie ein Verzeichnis mit dem Namen `testarchiv`.

b) Erzeugen Sie darin eine Datei mit zufälligem Inhalt:
`$> dd if=/dev/urandom of=testarchiv/zufallsdatei bs=1k count=256`

c) Lassen Sie sich die Größe der Datei anzeigen:
`$> ls -lh testarchiv/zufallsdatei`

d) Lassen Sie sich die Größe des Verzeichnisses anzeigen:
`$> ls -ldh testarchiv`

e) Erzeugen Sie ein `tar`-Archiv, welches das Verzeichnis enthält:
`$> tar -cf testarchiv.tar testarchiv`

f) Lassen Sie sich die Größe des Archives `testarchiv.tar` ausgeben.
Was fällt Ihnen auf?

g) Komprimieren Sie das Archiv:
`$> gzip testarchiv.tar`

Das Archiv ist nun erstellt. `gzip` hat das Archiv automatisch in `testarchiv.tar.gz` umbenannt.

h) Lassen Sie sich die Größe des gepackten Archives `testarchiv.tar.gz` ausgeben.

Frage: Es ist möglich, ein gepacktes Archiv (`.tar.gz`) mit einem Aufruf von `tar` zu erzeugen? Wie hätte dieser Aufruf lauten müssen?

i) Lassen Sie sich den Inhalt des gepackten Archives ausgeben.

Abgabe:

Dokumentieren Sie Ihre Shell Befehle und die Antworten durch Copy&Paste der Ausgabe in der TXT Datei “1.txt“.

2 Versionsverwaltung mit Git (30 P, OBLIGATORISCH)

Ein Versionsverwaltungssystem ist für die Entwicklung größerer Projekte unbedingt erforderlich. Für die Programmieraufgaben im Team sind sie ein Werkzeug, welches die parallele Entwicklung am Quellcode ermöglicht.

- Setzen Sie mit `git config --global` ihre Benutzerinformation.
- Schauen Sie mit `git help` die Hilfe an. Mit `git help <Kommando>` können Sie mehr über ein Kommando erfahren.
- Erstellen Sie ein Verzeichnis und initialisieren Sie ein Repository darin.
- Erstellen Sie eine Datei und fügen Sie diese zum Index hinzu.
- Committen Sie die Daten ins Repository.
- Ändern Sie den Dateiinhalt und committen Sie erneut.
- Betrachten Sie die Protokolle der Veränderungen, welchen Hash haben die Commits.

Abgabe:

Dokumentieren Sie Ihre Shell Befehle und die Antworten durch Copy&Paste der Ausgabe in der TXT Datei "2.txt".

3 Erste Schritte mit C (120 P, OBLIGATORISCH)

1. *Workshop*

   Downloaden Sie auf dem Cluster den C-Workshop Code von
   `http://wr.informatik.uni-hamburg.de/teaching/wintersemester_2012_2013/c-workshop`
   und entpacken sie die Datei (Tipp: `wget` und `unzip`). Versuchen Sie Lektionen 0-*.c bis 9-*.c durchzuarbeiten und zu verstehen. Es ist nicht wichtig, dass Sie alles auf Anhieb verstehen. Notieren Sie sich die dabei entstehenden Fragen.

2. *Kompilieren*

   Machen Sie sich schlau, was ein Makefile ist. (Tipp: The GNU Make Manual)
   Erstellen Sie ein Verzeichnis und schreiben sie in diesem Verzeichnis ein kleines C-Programm (z. B. `helloworld.c`). Erstellen Sie ebenfalls in dem Verzeichnis ein Makefile, so dass durch den Aufruf von `$> make` in diesem Verzeichnis Ihr C-Programm kompiliert wird (Tipp: `make`, `gcc`). Durch den Aufruf von `$> make clean` soll das Verzeichnis wieder in den Zustand von vor dem Kompilieren versetzt werden.

Abgabe:

Dokumentieren Sie Ihre Fragen in der TXT Datei "3.txt". Übersenden Sie die C-Datei und Makefile in einem Order "3".

4 Erste Schritte mit MPI (30 P, OBLIGATORISCH)

Hier wollen wir erste MPI Anwendungen in C entwickeln. Diese kleinen Testprogramme dienen dafür erste Erfahrungen zu sammeln und kleinere Aufgaben zu lösen. Für jede Aufgabe: erstellen Sie das dazu gehörige MPI Programm (in einer oder mehrerer Source-Code Dateien), ein Makefile welches eine ausführbare Datei erstellt und ein dazu passendes Jobskript um das Programm ausführen zu können.
Als Namensgebung für die Teilaufgaben erstellen Sie am besten einen Ordner: *Zettelnummer/Section_Subsection*. Somit ist uns auch immer klar wo ihr Code auf dem Cluster liegt, sofern es Rückfragen gibt.

4.1 Hello World

Alle Prozesse sollen „Hello World *Prozess* von *Prozesszahl*". Wobei die konkrete Prozessnummer (Rang) bzw. die gesamte Prozesszahl verwendet werden sollte. Nutzen sie `mpirun` um ihr Programm zu testen.

Abgabe:

Übersenden Sie das Makefile und die Quelldatei im Ordner "4"

5 Erste Schritte mit OpenMP (30 P, OBLIGATORISCH)

5.1 Hello World

Alle Prozesse sollen „Hello World *Prozess* von *Threadanzahl*". Wobei die konkrete Threadnummer bzw. die gesamte Threadanzahl verwendet werden sollte.

Abgabe:

Übersenden Sie das Makefile und die Quelldatei im Ordner "5"

6 Slurm (30 P, OBLIGATORISCH)

Slurm ist ein Batchsystem (Portable Batch System - PBS), welches die verfügbaren (Rechen)-Ressourcen verwaltet und Jobs auf diese verteilt. Ein Benutzer übergibt Slurm ein Jobskript (PBS-Skript), welches Information enthält was gestartet soll und welche Ressourcen benötigt sind. Die verfügbaren Ressourcen werden unter allen Benutzern aufgeteilt und somit der einzelne Job warten, bis er Ressourcen zugeteilt bekommt. Sobald der Job ausgeführt wird, werden die Ausgaben in entsprechenden Dateien dokumentiert. Siehe auch: `http://wr.informatik.uni-hamburg.de/teaching/ressourcen/beginners_guide#job_managing`

1. Erstellen Sie ein Jobskript für 4 Prozesse: Nutzen Sie den Befehl `srun` um den Befehl `hostname` auf jedem der Knoten auszuführen.

2. Starten Sie das Jobskript mehrfach.
   Frage: Was fällt Ihnen auf? Versuchen Sie Ihre Beobachtung(en) zu erklären!

3. Erstellen Sie Jobskripte für Ihre HelloWorld-Anwendungen für MPI (2 Knoten / 4 Prozesse) und OpenMP (1 Prozess / 8 Threads) und starten Sie Ihre Anwendungen damit.

Abgabe:

Übersenden Sie die Jobskripte im Ordner “6“ und legen Sie dazu die Datei “beobachtung.txt“ bei.