Anleitung für zwei C++ - Openmp - Beispiele auf der NWZSuperdome

(Timo Heinrich, t_hein03@uni-muenster.de)

Inhaltsverzeichnis:

0.Einleitung:

In dieser Beschreibung wird an zwei einfachen Beispielen der Quellcode eines Openmp-Programms, programmiert in C++, erläutert und explizit mit einigen Screenshots die Ausführung auf der NWZSuperdome (dem Shared-Memory-Rechner der IVV4) demonstriert. Alle Informationen findet man auch auf der Seite http://www.uni-muenster.de/IVVNWZ/computing/superdome/index.html

Ziel dieses Dokuments ist alle notwendigen Schritte für die Ausführung eines mit Openmp parallelisierten Programms zu erlernen.

1.Teil: Helloworldprogramm:

1.1 Quellcode: Helloworld.cpp

Zunächst muss der Quellcode des auszuführenden Programms erstellt werden. Hierzu kopiert man einfach den folgenden Programmcode in einen Editor (z.B. Proton, Emacs,...) und speichert die Datei in einem Ordner auf dem Lauwerk I: . In diesem Artikel wird der Speicherpfad: "I:\Superdomedemonstration" verwendet.

Quellcode des Helloworld-Programms:

```
#include <iostream>
#include <omp.h>
using namespace std;

int main()
{

        int NUMTHREADS, THREADID;

        #pragma omp parallel private(THREADID,NUMTHREADS)  //Initialisierung der parallelen Umgebung
        {

        THREADID= omp_get_thread_num();
        //Auslesen der Nummer des Threads mit Hilfe der function OMP_GET_THREAD_NUM()

        NUMTHREADS=omp_get_num_threads();
        //Auslesen der Anzahl der Threads mit Hilfe der function OMP_GET_THREAD_NUM()

        cout << "Das ist Thread Nummer:" << THREADID << "von insgesamt" << NUMTHREADS << "Threads" <<
endl;

        }//Beenden der parallelen Umgebung

return (0);
}
```

Zur Erklärung des Quellcodes:

Das Programm öffnet in Zeile 10 eine parallele Umgebung und schließt diese in Zeile 21. Innerhalb dieses Bereichs liest jeder Thread seine Nummer und die Anzahl aller Threads aus. Die Kommentare im Quellcode beschreiben das Programm noch detaillierter.

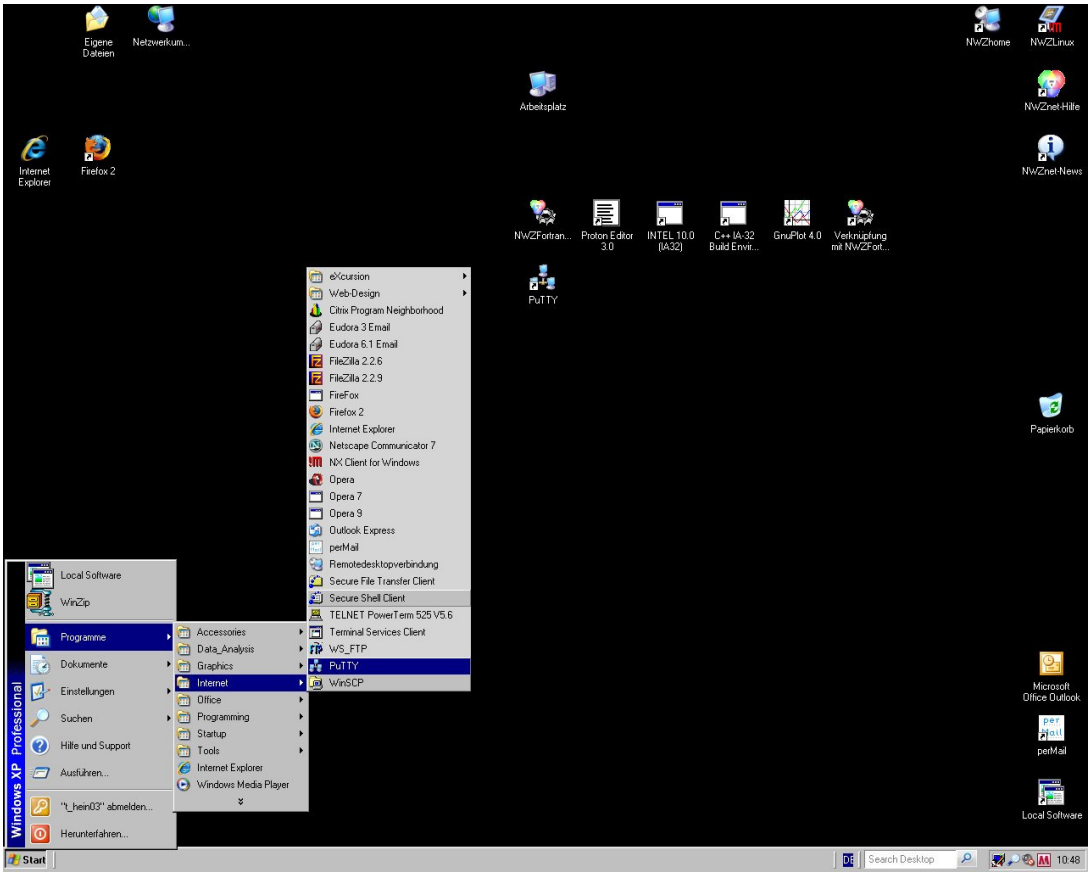
Als Ausgabe ist zu erwarten, wenn 4 die Anzahl der Threads ist:

```
Das ist Nummer 0 von 4
Das ist Nummer 1 von 4
Das ist Nummer 2 von 4
Das ist Nummer 3 von 4
```

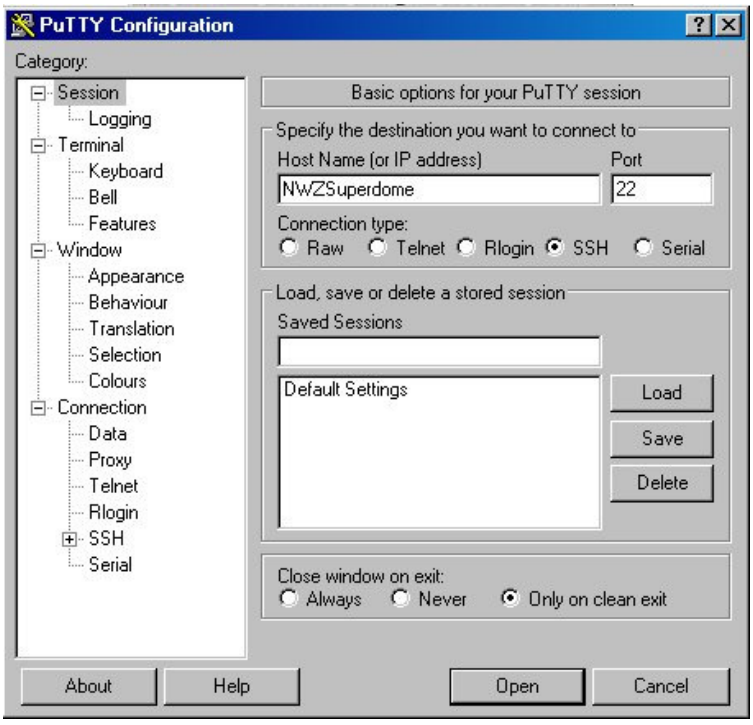
Die Nummerierung beginnt per Definition bei 0, wobei Thread Nummer 0 der Masterthread ist. Die Reihenfolge kann unter Umständen auch variieren.

1.2 Einloggen auf der NWZSuperdome

Nachdem jetzt das Programm unter I:\Superdomedemonstration\Helloworld.cpp gespeichert ist soll das Programm auf der Superdome ausgeführt werden.
Deshalb muss man sich als nächstes auf der NWZSuperdome einloggen. Vorteilhaft ist, dass das I:-Laufwerk automatisch angebunden wird, so dass das gespeicherte Programm auf der Superdome direkt verfügbar ist.
Auf der Superdome muss man sich mit dem Client Putty per SSH einloggen. Diesen findet man unter: ***Start > Programme > Internet > Putty*** :

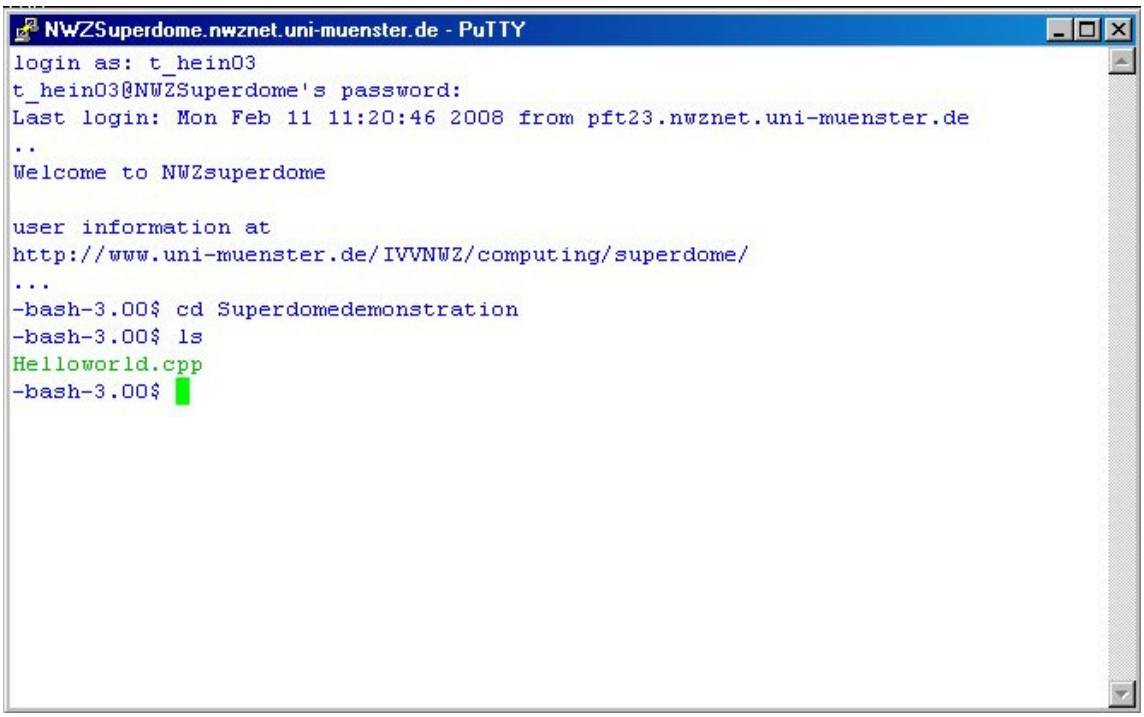


Bei dem sich öffnenden Fenster muss man bei Hostname ***NWZSuperdome*** eintragen. Außerdem muss darauf geachtet werden, dass als Connection type SSH gewählt wird und der Port auf 22 eingestellt ist:



Hat man diese Einstellungen vorgenommen muss man auf *Open* klicken. Es öffnet sich die Konsole, bei der man sich noch mit seiner NWZ-Kennung und dem zugehörigen Passwort einloggen muss:

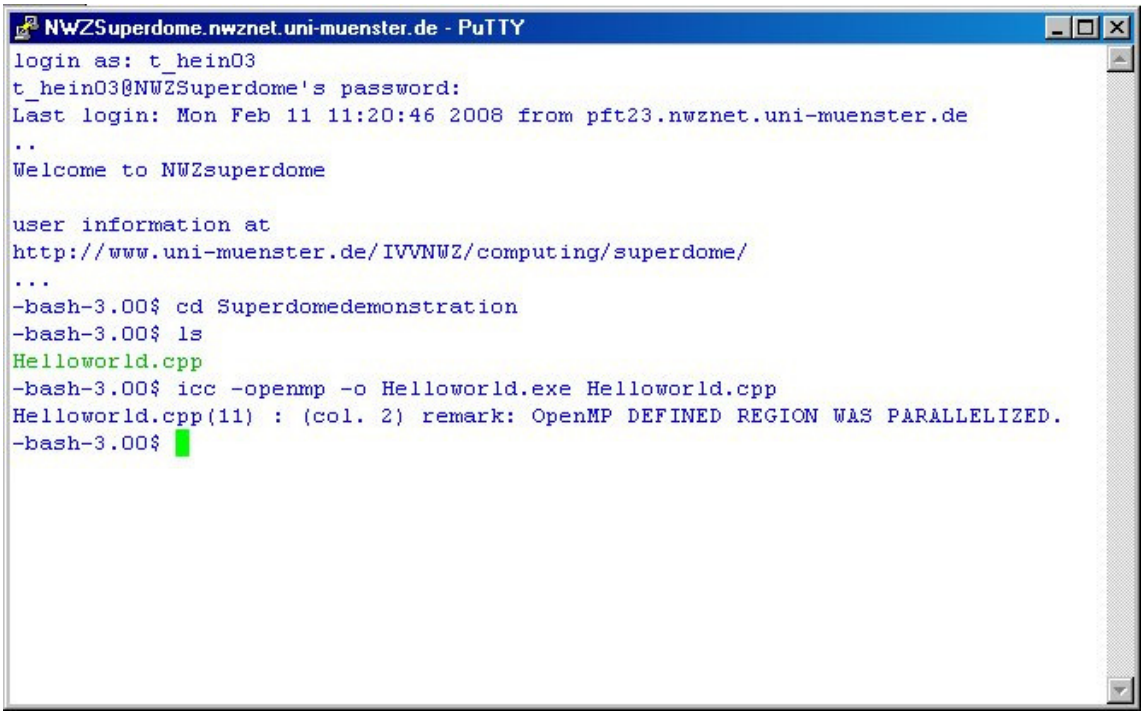
Als nächstes wechselt man in das Verzeichnis, in dem sich die Datei Helloworld.f90 befindet. In meinem Fall geschieht dies mit *cd Superdomedemonstration.* Hierbei ist darauf zu achten, dass Linux “case sensitive“ ist, das heißt Groß- und Kleinschreibung wird unterschieden.
Mit dem Befehl ls werden alle Dateien in diesem Ordner angezeigt, wie man sieht, befindet sich dort nur die bisher erstellte Datei Helloworld.cpp.



1.3 Kompilieren des Fortranprogramms mit Openmp:

Das Programm wird kompiliert durch das Kommando:

icc -openmp -o Helloworld.exe Helloworld.cpp



Wenn das Kompilieren erfolgreich war wird angezeigt welche Region parallelisiert wurde. Die Auszuführende, hier Helloworld.exe, befindet sich dann im gleichen Ordner wie das Programm.

1.4 Erstellen eines PBS-Skripts zur Ausführung des Jobs:

Bis jetzt befinden sich das C++ - Programm und die zugehörige Auszuführende in unserem Ordner. Um das Programm starten zu können, muss man ein PBS-Skript erstellen.

```
#PBS -q test
#PBS -N Helloworld
#PBS -e error.txt
#PBS -o output.txt
#PBS -m abe
#PBS -M abc@uni-muenster.de
#PBS -l nodes=1:ppn=4
source /opt/intel/fc/9.1.036/bin/ifortvars.sh
PARNODES=`wc -l $PBS_NODEFILE |gawk '{print $1}'`
export OMP_NUM_THREADS=$PARNODES
~/Superdomedemonstration/Helloworld.exe
```

Der Einfachheit halber kopiert man sich die oben stehenden Zeilen in einen Editor (Proton, Emacs,…) und speichert die Datei z.B. unter Helloworld.PBS im selben

Ordner in dem sich ihr Fortranprogramm befindet, in diesem Fall also “I:\Superdomedemonstration“ .

Zur Erklärung des Skripts:

-q test - hier wird festgelegt auf welcher Partition gerechnet werden soll, in diesem Fall auf der NWZSupertest-Partition

-N Helloworld - gibt den Namen des Jobs an, der auf der Superdome gestartet Wird. Dieser Name dient nur der Übersicht und hat sonst keine weitere Bedeutung

-e error.txt - Die Fehlerausgabe wird in die Datei error.txt umgeleitet

-o output.txt - Die Standard(Bildschirm-)ausgabe wird in die Datei output.txt Umgeleitet

-m abe - Bei Start, Ende oder Abbruch des Jobs wird eine e-mail …

-M abc@uni-muenster.de - ... an die hier stehende Adresse versandt

-l nodes=1:ppn=4 - hier wird die Anzahl der verwendeten Prozessoren festgelegt nodes muss immer auf 1 gesetzt werden, ppn ist die Anzahl der Prozessoren, die je nach Partition frei gewählt werden kann

source /opt/intel/fc/9.1.036/bin/ifortvars.sh – muss durchgeführt werden, damit dynamische Bibliotheken eingebunden werden

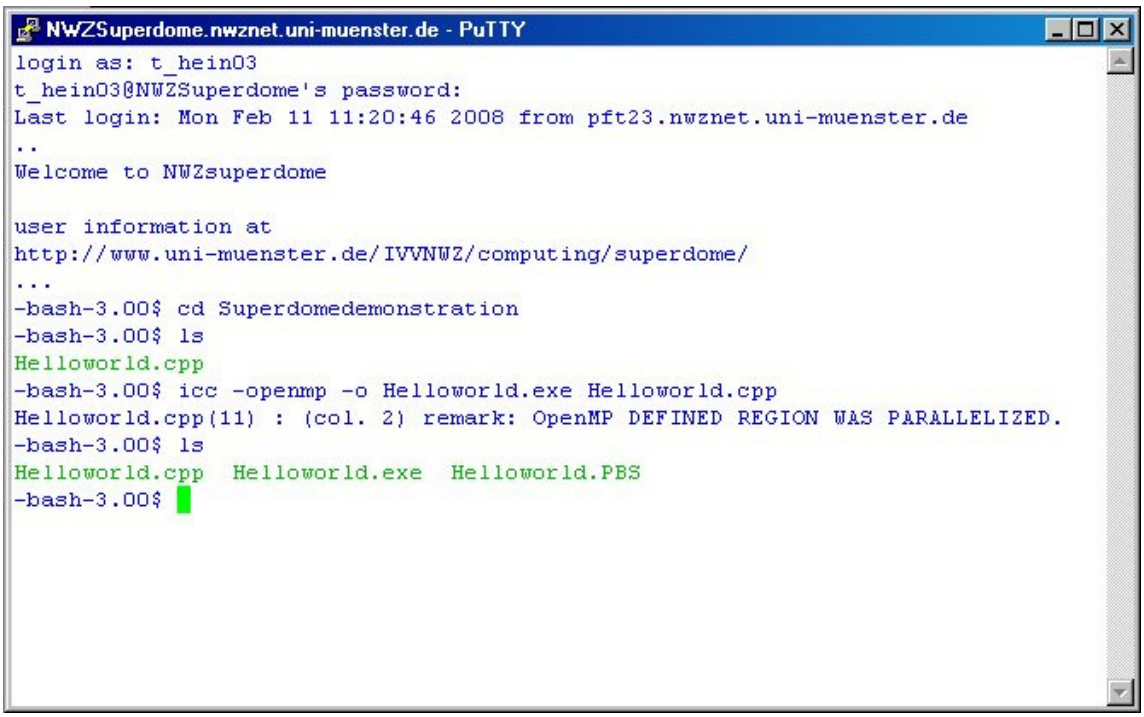
PARNODES=`wc -l $PBS_NODEFILE |gawk '{print $1}'`
export OMP_NUM_THREADS=$PARNODES - liest die Anzahl der Prozessoren aus und setzt die Anzahl der Threads auf diesen Wert

~/Superdomedemonstration/Helloworld.exe - startet das Programm

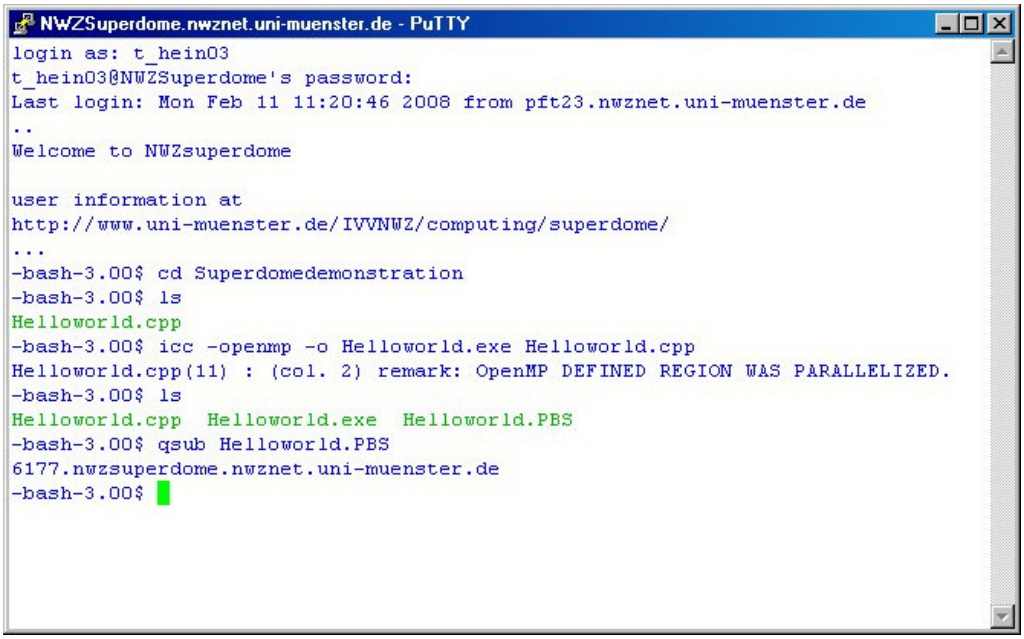
´

1.5 Starten des Jobs auf der Superdome:

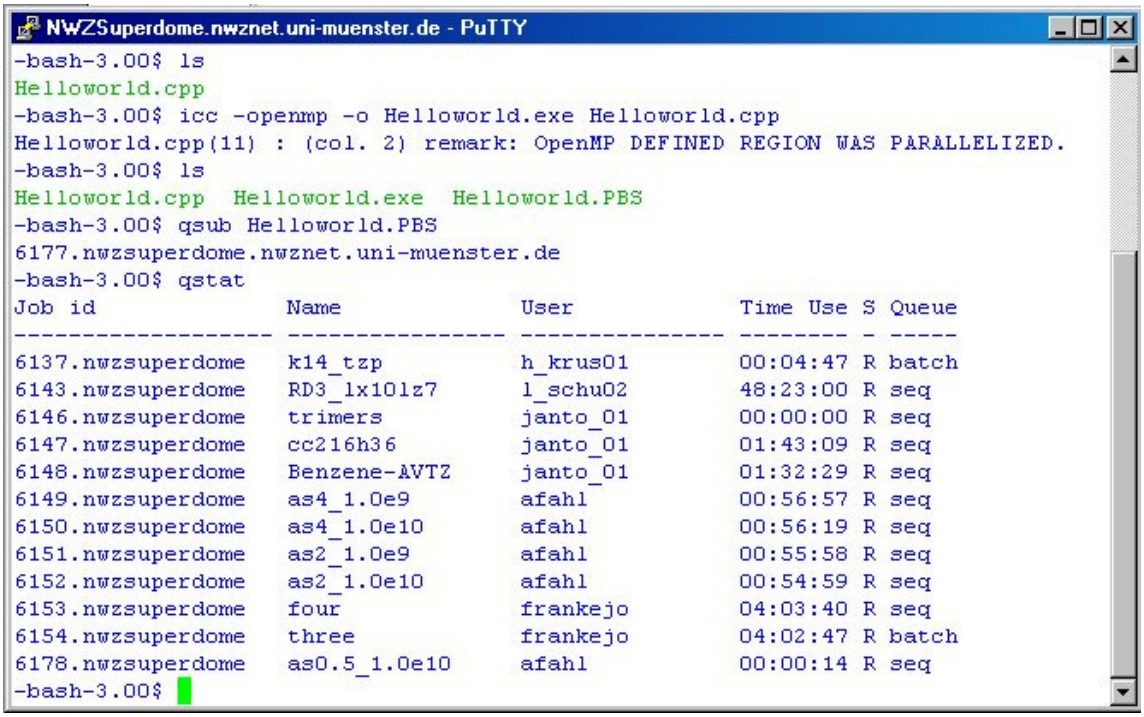
Zunächst vergewissern wir uns, dass sich die Auszuführende (Helloworld.exe) und das PBS-Skript im gleichen Ordner befinden:



Um den Job zu starten, was implizit in unserem PBS-Skript geschieht, muss das PBS-Skript ausgeführt werden. Dies geschieht mit qsub und dem Namen der PBS-Datei, also: *qsub Helloworld.PBS*
Danach wird dem Job eine Kennung zugewiesen.

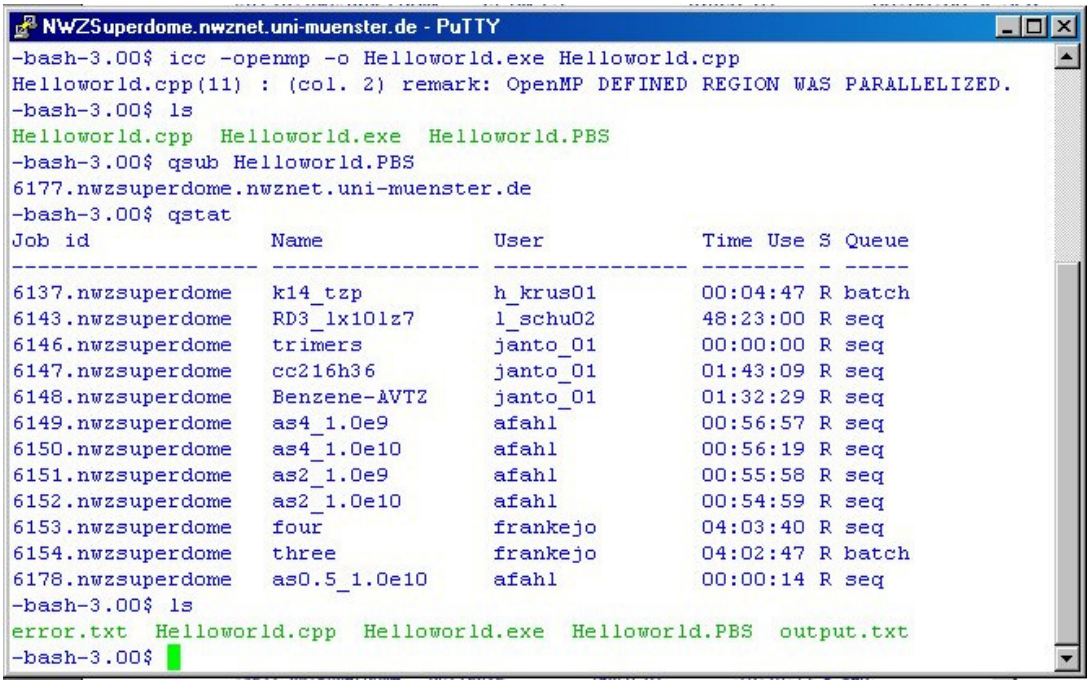


Der Status des Jobs kann mit dem Befehl qstat eingesehen werden.

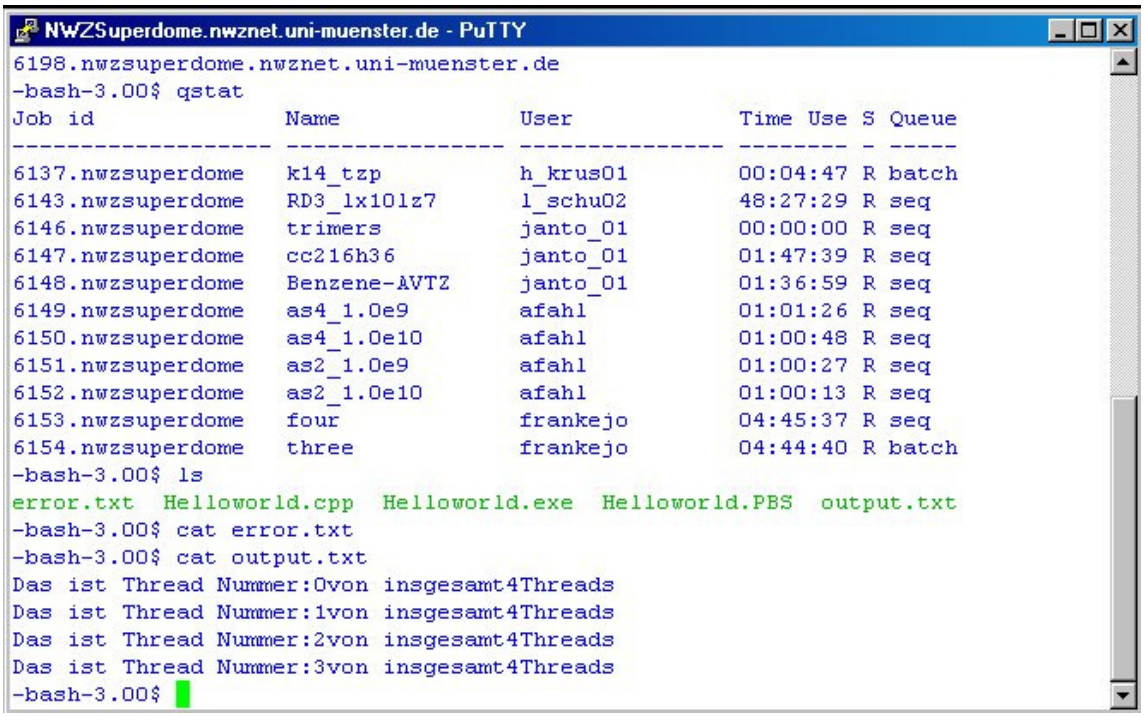


Der Job taucht hier allerdings nicht auf. Dies liegt daran, dass die Rechenzeit des Programms sehr klein ist, so dass das der Job bereits beendet wurde. Je nach Länge des Programms ergeben sich natürlich größere Rechenzeiten. Außerdem können Wartezeiten für den Job auftreten, wenn die Superdome gerade stark benutzt wird.

Nachdem der Job berechnet wurde findet man die error.txt und output.txt im selben Ordner wieder:



Die Ausgabe bzw. die Fehlerausgabe kann man sich zum Beispiel mit dem Befehl *cat ouput.txt* bzw. *cat error.txt* ansehen:



Die error.txt-Datei ist leer, da kein Fehler aufgetreten ist.

Das Ergebnis in der output.txt Datei stimmt mit der erwarteten Ausgabe überein. Damit ist das erste Beispiel beendet.

2.Teil: parallelisierte Schleife

Das Vorgehen ist genau dasselbe wie im ersten Teil, deshalb wird dieser Abschnitt etwas knapper sein:

2.1 Quellcode: Schleife.cpp
(Quelle: http://www.openmp.org/drupal/mp-documents/spec25.pdf)

```
#include <iostream>
#include <omp.h>
using namespace std;


int main()
{
        float a[100],b[100];
        int i,n, THREADID;
        n=100;
        for (i=0; i<n; i++)
        {
                a[i]=1/static_cast<float>(i+1);          //Definition von a

        }


        #pragma omp parallel for
        for (i=0;i<n; i++)
        {
        b[i] = (a[i] + a[i-1]) / 2.0;
        THREADID= omp_get_thread_num();

        cout << "Threadnummer" << THREADID << "Schleifendurchlaufnummer" << i << "Ergebnis" << b[i] << endl;
        }

  return 0;
}
```

In dem Programm wird zunächst ein Vektor a willkürlich definiert und hieraus in einer Schleife ein neuer Vektor b berechnet. Um diese Schleife wurde eine parallele Umgebung gesetzt, so dass die Schleife nun parallel berechnet wird.
Ausgegeben wird die Threadnummer, die Anzahl der Schleifendurchläufe und das Ergebnis b(I). Somit lässt sich nachvollziehen welcher Thread welche Komponente berechnet hat.

2.2 PBS-Datei zum Starten des Jobs:

Das PBS-Skript ändert sich nur geringfügig, nämlich die Bezeichnung des Jobs und der Pfad der exe-Datei:

```
#PBS -q test
#PBS -N loop
#PBS -e error2.txt
#PBS -o output2.txt
#PBS -m abe
#PBS -M abc@uni-muenster.de
#PBS -l nodes=1:ppn=4
source /opt/intel/fc/9.1.036/bin/ifortvars.sh
PARNODES=`wc -l $PBS_NODEFILE |gawk '{print $1}'`
export OMP_NUM_THREADS=$PARNODES
~/Superdomedemonstration/Schleife.exe
```

Diese Zeilen kopieren und in eine Datei z.B. mit dem Namen Schleife.PBS speichern.

2.3 Starten des Jobs:

Nachdem man sich wieder auf der Superdome eingeloggt hat, startet man den Job mit qsub Schleife.PBS

Hier ein kleiner Auszug des Ergebnisses: