

Athene HPC-Cluster Uni-Regensburg

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/hpc08.html

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/hpc08.pdf

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/hpc08.dvi

17. Juni 2011

KAPITEL 1

What's New — Neuigkeiten

13.6.2011: Neuer Knoten mit 2 Tesla M2070 GPGPU-Karten

10.1.2011: Neue Knoten 798, 801-816

11.2.2010: Neue Knoten 785-797

16.11.2009: Chapel von Cray, eine PGAS-Sprache mit Ziel hoher Produktivität, funktioniert.

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/hpccpl.html

10.11.2009: UPC, eine PGAS-Sprache, funktioniert.

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/pgasupc.html

30.10.2009: Titanium, eine PGAS-Sprache, funktioniert.

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/pgastit.html

KAPITEL 2

Probleme

module load parastation-gcc geht nicht

Schnelleinstieg

Zum Rechnen auf der Athene benötigt man

- einen Athene-Account
- den Login-Knoten Athene1 zum Einloggen
- einen Parallel-Compiler (FSF/Gnu, Intel)
- eine Parallel-Bibliothek (meist eine der MPI-Bibliotheken)
- das Torque-Batch-System zum Starten der Jobs auf den Knotenrechnern

1. Account

Bitte stellen Sie einen Antrag auf Clusterbenutzung im Rechenzentrum. Ihr normaler NDS-Account wird nach Genehmigung auf die Athene übertragen.

Ein elektronisches Antragsformular finden Sie in

<http://www-cgi.uni-regensburg.de/RZ/Dienste/Linux/HPC/index.phtml>

2. Login

Der sog. Loginknoten ist Athene1, d.h. zum Arbeiten müssen Sie sich dort einloggen. Als Account wird Ihr normaler NDS-Account verwendet. Beispiel mit ssh unter Linux:

```
ssh bbbxxxx@athene1 (aus dem Intranet)
ssh bbbxxxx@athene1.rz.uni-regensburg.de (sonst)
ssh athene1
```

Bildschirmprotokoll:

```
1 rex2:~> ssh athene1
2 The authenticity of host 'athene1 (132.199.252.31)' can't be established.
3 RSA key fingerprint is 2d:d4:6f:2d:f5:d3:9d:9d:e1:07:79:40:49:79:a0:25.
4 Are you sure you want to continue connecting (yes/no)? yes
5 Warning: Permanently added 'athene1,132.199.252.31' (RSA) to the list of known hosts.
6 Password:
7 Password:
8 Last login: Wed Mar 25 08:48:00 2009 from pc23520.rz.uni-regensburg.de
9 *****
10 *** User's Guide Documentation available at
11 *** /opt/parastation/doc/ClusterUserGuide-en.regensburg.pdf
12 *** (last updated Feb 04, 2009)
13 *****
14 ***          ssh into compute nodes is disabled          ***
15 ***          ssh auf die Rechenknoten ist abgeschaltet    ***
16 *****
17 ***          Torque has been upgraded to 2.3.6           ***
18 ***          Torque Version 2.3.6 eingespielt            ***
19 *****
20 athene1 /home/brf09510>
```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/ssh.txt)

3. Wahl von Compiler und MPI-Library

Es stehen mehrere Werkzeuge zur Programmentwicklung zur Verfügung. Um sie benutzen zu können, müssen diverse Umgebungsvariable gesetzt werden. Das kann direkt geschehen, wenn Sie sich auskennen oder mit dem *module*-System automatisiert erfolgen.

In jedem Fall wird ein Compiler benötigt. Derzeit sind zwei Hersteller (FSF/Gnu, Intel), drei Sprachen (Fortran, C, C++) und ihre Versionsvarianten (Intel 10, Intel 11, gnu 4.1, gnu 4.4) verfügbar.

Hersteller	Aufruf	Sprache	Compiler	Version (Stand Oktober 2009)
FSF/Gnu	gfortran	Fortran	gfortran	4.1.2, 4.4.1
FSF/Gnu	gcc	C	gcc	4.1.2, 4.4
FSF/Gnu	g++	C++	g++	4.1.2, 4.4.1
Intel	ifort	Fortran	ifort	10.0, 11.0
Intel	icc	C	icc	10.0, 11.0
Intel	icpc	C++	icpc	10.0, 11.0

Über den aktuellen Stand können Sie sich jederzeit mit dem Kommando `module avail` informieren.

Die wichtigste Parallelbibliothek ist MPI, die auf der Athene in derzeit drei Varianten vorliegt.

MPI-Bibliothek	Beschreibung	Webseite für weitere Informationen
Parastation MPI	MPICH für Infiniband	http://mvapich.cse.ohio-state.edu/
MVAPICH2	MPICH2 für IB mit VAPI	
Intel MPI (impi)		

Die vierte Variante openMPI ist derzeit nur für den internen Gebrauch vorgesehen.

3.1. Wahl von Compiler/MPI-Library mit *module*.

Eine ausführliche Beschreibung von *module* finden Sie unter

<http://modules.sourceforge.net/>
<http://modules.sourceforge.net/man/module.html>

module ist ein Alias für die folgende Kommandofolge mit dem Kommandokern `/usr/bin/modulecmd`

```

1 set  _prompt="$prompt ";
2 set  prompt="";
3 eval ` /usr/bin/modulecmd tsh !* `;
4 set  _exit=$status ;
5 set  prompt="$_prompt ";
6 unset _prompt ;
7 test 0 = $_exit ;
```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/modalias.txt)

Die häufigsten *module*-Kommandos sind:

```

module avail (Welche Module existieren?)
module load parastation-gcc (Laden eines Moduls - hier impi-intel)
module list (Welche Module sind geladen?)
module unload parastation-gcc (Entfernen eines Moduls - hier parastation-gcc)
```

Manche Module schließen sich wechselseitig aus. Beim Wechsel muss dann vorher das alte Modul erst entfernt werden.

<i>Module-Argument</i>	
dot	GraphViz: Tool zur Erzeugung von Graphen
gcc-4.4	Gnu Compiler Collection 4.4 (C/C++/Fortran unterstützt OpenMP V3)
impi-gcc	impi übersetzt mit gcc
impi-intel	impi übersetzt mit icc
intel-10.0-icc	Intel icc 10.0
intel-10.0-ifort	Intel ifort 10.0
intel-11.0	Intel icc/ifort 11.0
module-cvs	
module-info	
modules	
mvapich2-gcc	MVAPICH2 übersetzt mit gcc
mvapich2-intel	MVAPICH2 übersetzt mit icc
null	
parastation-gcc	Parastation MPI übersetzt mit gcc
parastation-intel	Parastation MPI übersetzt mit icc
use.own	

Zu OpenMP siehe <http://sites.google.com/site/gfortransite/>

Zu gfortran siehe <http://gcc.gnu.org/wiki/GFortran>

4. Übersetzung

Die Übersetzung normaler Programme und ihr Start sollte Linux-Kundigen keine Probleme bereiten. Nur die zusätzliche Verwendung paralleler Bibliotheken wie MPI erfordert Änderungen bei der gewohnten Bedienung.

In diesen Schnelleinstieg wird nur die Kombination FSF/Gnu mit Parastation-MPI beschrieben. Andere Kombinationen lesen Sie bitte in der ausführlicheren Dokumentation weiter unten nach.

Wir verwenden ein Hello-World-Programm mit MPI:

```

1 /* hello_mpi.c */
2
3 #include <stdio.h>
4 #include <mpi.h>
5
6 int main (int argc, char **argv)
7 {
8     int rank, size;
9     MPI_Init (&argc, &argv); /* starts MPI */
10    MPI_Comm_rank (MPI_COMM_WORLD, &rank); /* get current process id */
11    MPI_Comm_size (MPI_COMM_WORLD, &size); /* get number of processes */
12    printf ("_Hello_world_from_process_%d_of_%d\n", rank, size);
13    MPI_Finalize ();
14    return 0;
15 }

```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/helmpi.c)

Es wird mit gcc übersetzt, anschließend wird Parastation-MPI anmontiert. Wie das folgende Beispiel zeigt, sind trotz des Wrappers mpicc die normalen Compiler-Optionen erlaubt. Natürlich müssen Sie ein Programm nur einmal übersetzen. Die bei der Option Wall erscheinende Warnung in mpi.h kann ignoriert werden.

```

module load parastation-gcc
mpicc helmpi.c
mpicc -v helmpi.c
mpicc -ansi -pedantic -Wall -o hp helmpi.c
In file included from helmpi.c:3:
/opt/parastation/mpi2/include/mpi.h:344: warning: ISO C90 does not support 'long long'

```

Die C++-Variante lautet:

```

1 /* hello_mpi.cc */
3 #include <iostream>
  #include <mpi.h>
5
  using namespace std;
7
  int main (int argc, char **argv)
9 {
    int rank, size;
11 MPI::Init (argc, argv);           // starts MPI
    rank = MPI::COMM_WORLD.Get_rank (); // get current process id
13 size = MPI::COMM_WORLD.Get_size (); // get number of processes
    cout << "_Hello_world_from_C++-process_" << rank << "_of_" << size << endl;
15 MPI::Finalize ();                // finish MPI
    return 0;
17 }

```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/helmpi.cc)

Es wird mit g++ übersetzt, anschließend wird Parastation-MPI anmontiert. Wie das folgende Beispiel zeigt, sind trotz des Wrappers mpicxx die normalen Compiler-Optionen erlaubt. Natürlich müssen Sie ein Programm nur einmal übersetzen. Die bei der Option Wall erscheinende Warnung in mpi.h kann ignoriert werden.

```

module load parastation-gcc
mpicxx helmpi.cc
mpicxx -v helmpi.cc
mpicxx -ansi -pedantic -Wall -o hp helmpi.cc

```

5. Start von HPC-Jobs

Das übersetzte Programm sollte normalerweise als Batchjob zur parallelen Ausführung auf den Knotenrechnern gestartet werden. Dazu wird Torque verwendet. Es ermöglicht sowohl einen interaktiven Start von Jobs (z.B. zum Debuggen) als auch den Start von Batch-Jobs. Torque ist aus dem älteren OpenPBS hervorgegangen. Deshalb taucht das Kürzel PBS in den Submit-Scripts auf. Der Scheduler heißt Maui.

Beschreibung:

```

http://www.clusterresources.com/pages/resources/documentation.php
http://www.clusterresources.com/pages/products/torque-resource-manager.php
http://www-theor.ch.cam.ac.uk/IT/servers/maui/maui-pbs-introduction.html

```

Der Batchjob wird in einem modifizierten Shell-Script, einem sog. PBS-Script aufgerufen. Als Script-Prozessor eignen sich alle Linux-Script-Prozessoren (bash, csh, sh,...).

```

1 #!/bin/bash
2 #PBS -l nodes=4:ppn=8
3 #PBS -l walltime=00:01:00
4
5 #      cput=00:01:00 (hh:mm:ss) CPU-Zeit
6 #      walltime=00:01:00 (hh:mm:ss) Verweilzeit
7
8 module load parastation-gcc
9
10 cd ${PBS_O_WORKDIR}
11
12 mpiexec -np 32 ./hp
13

```

```
14 # end of this script
```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/batch.sh)

Mit der ersten Pseudokommentarzeile (Shebang `#!/bin/bash`) wird `bash` zur Ausführung gestartet. Die folgenden PBS-Direktiven (`#PBS directive`) sind mit dem Schlüsselwort `PBS` als Direktiven gekennzeichnet. Zeile 2 fordert 4 Knotenrechner (nodes) mit je 8 CPUs (`ppn` - processors per node, cores), also insgesamt 32 Cores an. Zeile 3 stellt eine maximale Verweilzeit (`walltime` - Wanduhr) des Jobs von 1 Minute ein. Verweilt der Job länger als die hier angegebene Zeit im System, wird er automatisch abgebrochen. Kleine Werte führen andererseits zu einem früheren Jobstart als größere. Wählen Sie also die Werte nach dem Prinzip *so klein wie möglich, so groß wie nötig, vielleicht mit einem moderaten Sicherheitszuschlag nach oben*. Diese Angabe wird also vom Batchsystem verwendet, um den Zeitpunkt des Jobstarts zu bestimmen und um zu lange (möglicherweise fehlerhafte) Jobs wieder zu entfernen. Der folgende Kommentar in Zeile 5 begrenzt die Rechenzeit (`cput` - CPU Time pro Core) auf 1 Minute. Dieser Wert kann angegeben werden, spielt jedoch keine Rolle, da er vom Batchsystem **nicht** verwendet wird. Der Rest der Datei besteht aus normalen Shellscript-Kommandos. Das `cd`-Kommando wechselt vom möglicherweise falschen Verzeichnis `${HOME}` in das Verzeichnis `${PBS_O_WORKDIR}`, aus dem das PBS-Script aufgerufen wurde. Hier werden auch die Resultate gespeichert. Das letzte Kommando `mpiexec` startet das Programm in der Datei `hp` auf 32 Cores (die Datei `hp` wurde wie in der letzten Übersetzung weiter oben erstellt).

Der Job wird mit dem Kommando

```
qsub batch.sh
```

in die Queue gestellt und abgearbeitet, sobald genügend Knoten frei geworden sind und der Job unter allen mit ihm konkurrierenden die höchste Priorität bekommen hat.

Auch im `qsub`-Kommando können begrenzende Limits als Option wie im PBS-Script selbst angegeben werden. Im folgenden Beispiel wird mit der Option `-l` (`limit`) die maximale `walltime` begrenzt. Von mehreren konkurrierenden Angaben gilt zunächst die `qsub`-Angabe, dann die PBS-Angabe und zuletzt die Voreinstellung.

```
qsub -l walltime=2:0:0 batch.sh (beim Start)
#PBS -l walltime=2:00:00 (im PBS-Script)
```

Mit dieser Beispiel-Option wird die maximale Verweilzeit (`walltime`) auf 2 Stunden begrenzt. Begrenzung der CPU-Zeit (`cput`) hat keine Wirkung.

Über den Status von Jobs, Queues und Server gibt das Kommando `qstat` Auskunft.

```
qstat
qstat job-id
qstat -f job-id
qstat -q Queue-Übersicht
qstat -Q Queue-Übersicht
```

Versehentlich gestartete Jobs können mit `qdel` wieder gelöscht werden:

```
qdel job-id
```

Nach dem normalen Ende eines Jobs werden die nach `stdout` und `stderr` geschriebenen Programmergebnisse in zwei Dateien geschrieben, deren Namen mit dem PBS-Scriptnamen beginnen und die Jobnummer enthalten. Im Beispiel sind das

```
1 brf09510@athene1:~> ls -altr
2 total 15516
3 ...
4 -rw----- 1 brf09510 rz          0 2009-03-31 13:43 batch.sh.e165093
5 -rw----- 1 brf09510 rz       1140 2009-03-31 13:43 batch.sh.o165093
6 drwxr-xr-x 3 brf09510 rz       4096 2009-03-31 13:43 .
7 brf09510@athene1:~>
```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/eofiles.txt)

Ein interaktiver Job wird gestartet mit

```
qsub -I
```

Queue-Übersicht (Jan'2011):

common
common32
striegnitz
rzkurs
spang_i
merkl
bocola
condmat
serial
develop
express
dick

KAPITEL 4

Allgemeines

1. Zeittafel

Abnahme: 26.2.2009

Testbetrieb: Februar 2009

Einweihung: 29.4.2009

Inbetriebnahme: 1.4.2009

Hardware

In der veralteten und zu groben Taxonomie nach Flynn ist die Athene als MIMD-Architektur aufgebaut. Die Struktur des gesamten Clusters ist eine Distributed Memory Architektur. Jeder Knoten hat seinen eigenen Speicher, die Kommunikation erfolgt über eines der beiden Netze (Infiniband, GigE). Jeder einzelne Knoten verfügt über zwei Prozessoren mit je eigenem Speicher, der jedoch prinzipiell von beiden Prozessoren genutzt werden kann, also eine ccNUMA-Architektur. Jeder der beiden Prozessoren besteht aus vier Rechenkernen mit gemeinsamem Speicher, bildet also eine SMP-Architektur.

1. Rechner

Bei der Erstinstallation war der Cluster fast homogen. Bei Erweiterungen kann diese Homogenität nicht erhalten bleiben.

1.1. Übersicht.

Login Node (Athene 1), Typ S1
 Master Node (Athene 2), Typ S1
 File Server (Athene 3-4), 2 Typ S1

Node 1-122 (Infiniband), 122 Typ O1
 Node 123-143 (Infiniband), 21 Typ O2
 Node 144-176 (Infiniband), 33 Typ O3
 Node 769-776, 780 (Gigabit Ethernet), 9 Typ O4
 Node 777-779 (Gigabit Ethernet), 3 Typ O5

Node 785-797 (Gigabit Ethernet), 13 Typ X1
 Node 798,801-816 (Gigabit Ethernet), 17 Typ X2
 Node 833-834 (Infiniband), 2 Typ X3

Zusammen 220 Knoten, $402 \times$ Quadcore-CPU, $38 \times$ Hexacore-CPU und 1836 Cores (1504 Opteron, 104+228 Xeon).

Gesamtspeicher: 4352 GiB

Nicht jeder *node* ist allgemein verfügbar.

1.2. Ausstattung der Rechenknoten Typ O1, 1-122.

CPU: $2 \times$ AMD K10 Quad-Core, 2,2 GHz (Opteron 2354, Barcelona B3)
 RAM: 16 GiB
 HDD: 250 GB
 Interconnect: GbE, 4X-DDR IB

1.3. Ausstattung der Rechenknoten Typ O2, 123-143.

CPU: $2 \times$ AMD K10 Quad-Core, 2,2 GHz (Opteron 2354, Barcelona B3)
 RAM: 16 GiB
 HDD: 500 GB
 Interconnect: GbE, 4X-DDR IB

1.4. Ausstattung der Rechenknoten Typ O3, 144-176.

CPU: 2 × AMD K10 Quad-Core, 2,2 GHz (Opteron 2354, Barcelona B3)
 RAM: 32 GiB
 HDD: 500 GB
 Interconnect: GbE, 4X-DDR IB

1.5. Ausstattung der Rechenknoten Typ O4, 769-776,780.

CPU: 2 × AMD K10 Quad-Core, 2,2 GHz (Opteron 2354, Barcelona B3)
 RAM: 16 GiB
 HDD: 500 GB
 Interconnect: GbE

1.6. Ausstattung der Rechenknoten Typ O5, 777-779.

CPU: 2 × AMD K10 Quad-Core, 2,2 GHz (Opteron 2354, Barcelona B3)
 RAM: 32 GiB
 HDD: 1 TB
 Interconnect: GbE

1.7. Ausstattung der Rechenknoten Typ S1, athene[1-4].

CPU: 2 × AMD K10 Quad-Core, 2,2 GHz (Opteron 2354, Barcelona B3)
 RAM: 32 GiB
 HDD: 146 GB SAS
 Interconnect: 10 GbE, 4X-DDR IB

1.8. Ausstattung der Rechenknoten, Typ X1, 785-797.

CPU: 2 × Intel Xeon, 2,4 GHz (E5530, Nehalem-EP "Gainestown" [4 Cores, 8 Threads])
 RAM: 24 GiB
 HDD: 250 GB
 Interconnect: GbE

1.9. Ausstattung der Rechenknoten, Typ X2, 798, 801-816.

CPU: 2 × Intel Xeon, 2,66 GHz (X5650, Westmere-EP "Gulftown" [6 Cores, 12 Threads])
 RAM: 24 GiB
 HDD: 250 GB
 Interconnect: GbE

1.10. Ausstattung der Rechenknoten, Typ X3, 833,834.

CPU: 2 × Intel Xeon, 2,8 GHz (X5660, Westmere-EP "Gulftown" [6 Cores, 12 Threads])
 GPU: 1 × M2070 (nVidia Tesla "Fermi"), 6 GiB ECC-RAM, 448 CUDA-Recheneinheiten
 (Organisation: 14 Shadercluster a 32 Streamprozessoren; theoretische Leistung: Rpeak_SP 1.03TFlops, Rpeak_DP 515GFlops)
 RAM: 24 GiB
 HDD: 250 GB
 Interconnect: GbE, 4X-QDR IB
 System: Debian

1.11. Zugriffsrechte.

public: 1-108, 144-176 (common queue)
 seriell: 769 - 780
 common32: 144-176 (4GiB pro Prozess)
 Striegnitz: 109-122
 Spang: 123-134, 769-776
 Merkl: 135-143
 Bocola: 777-779
 condmat (Richter/Grifoni): 785-797, 801-816
 dick: 798

tesla: 833,834 (currently in development)

1.12. Voreinstellung der Queues common, common32, serial, express und develop.

```
min.nodect = 2
max.nodect = 32
default.pmem = 1 GiB
max.pmem = 2 GiB
default.nodes = 2:ppn = 8
max.walltime = 48:00 h
max_user_queuable = 20
```

```
default.nodes = 1:ppn = 1 (serial queue)
```

```
default.pmem = 2 GiB (common32)
max.pmem = 4 GiB (common32)
```

```
max.walltime = 5:00 h (express)
max.nodect = 4 (express)
```

```
max.walltime = 0:10 h (develop)
```

Aktuelle Angaben mit `qstat -q` und `qstat -Q` erfragen!

2. Netz

Gigabit Ethernet (Broadcom Corporation NetXtreme Gigabit Ethernet PCI Express (rev 21))

Infiniband (DDR 4x, 20 GBit/s, InfiniBand: Mellanox Technologies MT25204 [InfiniHost III Lx HCA] (rev 20))

Zwei unabhängige Netze, die beide für Message Passing verwendet werden können. Das Ethernet wird eher für Verwaltungszwecke, Infiniband eher für Message Passing eingesetzt.

3. Dateisystem

3.1. Dateiserver.

Athene3

Athene4

Zwei NFS-Server für die Dateisysteme `$HOME` (quotiert, 3GiB/15000 files), `/scratch` und `/data`.

Derzeit existiert kein paralleles Filesystem.

3.2. Lokale Filesysteme: NFS. `/tmp` kann auch für lokale temporäre Daten verwendet werden; ist bei verteilter Anwendung mit vielen Knoten unter Umständen schneller als `/scratch`.

3.3. Umgang mit den Scratch-Verzeichnissen.

- Lokales Scratch wird nach jedem Jobende gelöscht. Benutzung unter `/scratch/${PBS_JOBID}`
- NFS-Scratch1 und -Scratch2: Hier existiert pro Benutzer ein Unterverzeichnis. Bei 80% Füllstand werden älteste Dateien automatisch gelöscht, bis ein Füllstand von 60% erreicht wird.

Achtung: Es ist unfair und wird überwacht, ob Dateien mit `touch` aufgefrischt werden.

KAPITEL 6

Betriebssystem

SLES10 SP2 Enterprise Linux (Novell)

Sprachen und Compiler

1. FSF/Gnu

Alle Gnu-Compiler stehen ohne weitere Vorbereitung direkt zur Verfügung (gfortran, gcc, g++). Für MPI-Programme sollte ein Wrapper zur Übersetzung benutzt werden (mpicc).

2. Intel

Intel muss mit `module load` aktiviert werden.

```
module load intel-11.0
module load intel-10.0-icc
module load intel-10.0-ifort
```

Die Compiler heißen mit `ifort`, `icc` und `icpc`.

```
/opt/intel/Compiler/11.0/074/Documentation
http://www.uni-r.de/EDV/kurs_info/brf09510/hpc/opt/intel/Documentation/getting_started_c.htm
http://www.uni-r.de/EDV/kurs_info/brf09510/hpc/opt/intel/Documentation/getting_started_f.htm
```

Es wird mit dem Compiler von Intel und der MPI-Bibliothek von MVAPICH übersetzt:

```
1 athene1 /home/brf09510> module load intel-11.0
2 athene1 /home/brf09510> module load mvapich2-intel
3 athene1 /home/brf09510> module list
4 Currently Loaded Modulefiles:
5   1) intel-11.0          2) mvapich2-intel
6 athene1 /home/brf09510> which icc
7 /opt/intel/Compiler/11.0/074/bin/intel64/icc
8 athene1 /home/brf09510> mpicc helmpi.c
9 athene1 /home/brf09510>
```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/mpicc.txt)

3. OpenMP

OpenMP (Open Multiprocessing) ermöglicht parallele Ausführung in mehreren Threads, die durch Compiler-Direktiven in C, C++ und Fortran gesteuert werden.

Derzeit mit den Intel- und mit Gnu-Compilern (Version 4.4) möglich.

Zu OpenMP siehe <http://sites.google.com/site/gfortransite/>

Zu gfortran siehe <http://gcc.gnu.org/wiki/GFortran>

Umgebungsvariable für OpenMP:

```
setenv OMP_NUM_THREADS 5 (c-shell)
export OMP_NUM_THREADS=5 (bash)
```

Übersetzung mit

```
gcc -fopenmp x.c
```

4. PGAS

4.1. co-Array Fortran. siehe http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/pgascaf.html

4.2. UPC: Unified parallel C. siehe http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/pgasupc

4.3. Titanium: PGAS-Sprache auf Java-Gundlage. siehe http://www.uni-regensburg.de/EDV/kurs_info/

Titanium ist eine in Berkeley entwickelte Java-basierte PGAS-Sprache. Webseite:
<http://titanium.cs.berkeley.edu/>

Im Verzeichnis
 /home/brf09510/tc

existiert eine clusterfähige Titaniuminstallation. Sie kann benutzt werden, wenn ihr Verzeichnisname in den Pfad eingetragen wird.

```
setenv PATH "${PATH}:/home/brf09510/tc/bin"
```

Der Compiler kann (fast) jedes Java-Programm und jedes Titanum-Programm übersetzen. Die umfangreiche Java-Bibliothek steht natürlich nur sehr eingeschränkt zur Verfügung. Der Compileraufruf ist

```
tcbuild --help
tcbuild --backend gasnet-ibv-uni hello.ti
tcbuild --backend mpi-cluster-uniprocess hello.ti
tcbuild --backend sequential hello.ti
tcbuild --backend smp hello.ti
```

Übersetzte Titanium-Programme werden mit normalen PBS-Jobs auf den Clusterknoten gestartet. Das folgende Beispiel startet ein Programm in der Datei ./hello.

```
1 #PBS -l walltime=30,nodes=2:ppn=8 -k eo
2 ls -l $PBS_NODEFILE
3 #cat $PBS_NODEFILE
4 pwd
5
6 #echo 'gasnet uni'
7 mpiexec -np 16 -f $PBS_NODEFILE ./hello
8 exit
```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/titjob.sh)

4.4. X10. <http://x10-lang.org/>

<http://dist.codehaus.org/x10/documentation/languagespec/x10-latest.pdf>

http://domino.research.ibm.com/comm/research_projects.nsf/pages/x10.index.html.

4.5. Chapel. <http://chapel.cray.com/spec/spec-0.795.pdf>

<http://chapel.cray.com/>.

4.6. Fortress.

Bibliotheken

1. Parallelrechnen

1.1. Parastation-MPI (modifiziertes MPICH mit Infiniband-Kommunikation).

1.2. mvapich.

1.3. Intel-MPI. [/opt/intel/impi/3.2.0.011/Doc_Index.html](http://opt/intel/impi/3.2.0.011/Doc_Index.html)

http://www.uni-r.de/EDV/kurs_info/brf09510/hpc/opt/intel/impi/3.2.0.011/Doc_Index.html

1.4. OpenMP.

2. Atlas

Die Bibliothek steht im Verzeichnis

`/opt/lapack/atlas-3.8.2`

Mit dem folgenden Beispielprogramm (Quelle: <http://math-atlas.sourceforge.net/faq.html>) wird die Versionsnummer der Bibliothek am Bildschirm ausgegeben:

```

#include <stdlib.h>
2 main ()
  /*
4  * Compile, link and run with something like:
  * gcc -o xprint_buildinfo -L[ATLAS lib dir] -latlas ; ./xprint_buildinfo
6  * if link fails, you are using ATLAS version older than 3.3.6.
  */
8 {
  void ATL_buildinfo(void);
10  ATL_buildinfo ();
  exit (0);
12 }

```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/atlasv.c)

Übersetzung des Beispiels und Start:

```

gcc -I /opt/lapack/atlas-3.8.2/include/ -c atlasv.c
2 gcc -o atlasv atlasv.o -L /opt/lapack/atlas-3.8.2/lib/ -latlas

```

```

4 athene1 /home/brf09510> ./atlasv

```

ATLAS version 3.8.2 built by root on Mon Feb 2 10:14:50 CET 2009:

```

6  UNAME      : Linux node0769 2.6.16.60-0.33-smp #1 SMP Fri Oct 31 14:24:07 UTC 2008 x86_64
  INSTFLG   : -1 0 -a 1
8  ARCHDEFS  : -DATL_OS_Linux -DATL_ARCH_AMD64K10h -DATL_CPUMHZ=2194 -DATL_SSE3 -DATL_SSSE3
  F2CDEFS   : -DAdd_ -DF77_INTEGER=int -DStringSunStyle
10  CACHEEDGE: 524288
  F77       : gfortran, version GNU Fortran (GCC) 4.3.2
12  F77FLAGS  : -fomit-frame-pointer -mfpmath=sse -msse3 -O2 -falign-loops=32 -m64
  SMC       : gcc, version gcc (GCC) 4.3.2
14  SMCFLAGS  : -fomit-frame-pointer -mfpmath=sse -msse3 -O2 -falign-loops=32 -m64

```

```
SKC      : gcc, version gcc (GCC) 4.3.2
16 SKCFLAGS : -fomit-frame-pointer -mfpmath=sse -msse3 -O2 -falign-loops=32 -m64
athene1 /home/brf09510>
```

(http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/atlasv.r)

3. MKL Math Kernel Library

```
/opt/intel/Compiler/11.0/074/Documentation/mkl
http://www.uni-r.de/EDV/kurs\_info/brf09510/hpc/opt/intel/Documentation/mkl/mkl\_documentation.htm
```

KAPITEL 9

Torque und Maui

1. Torque

1.1. Übersicht. siehe <http://www.clusterresources.com/products/torque/docs/a.a.commands.shtml>

1.2. qsub. siehe <http://www.clusterresources.com/products/torque/docs/commands/qsub.shtml>

1.3. Ressourcen. siehe <http://www.clusterresources.com/products/torque/docs/2.1.jobsubmission.shtml>

Ressource	Wert	Bedeutung
cput	[[hh:]mm:]ss	Maximale CPU-Zeit pro core des Jobs
walltime	[[hh:]mm:]ss	Maximale Verweilzeit des Jobs

1.4. qstat.

1.5. qdel.

2. PBS

2.1. PBS Umgebungsvariable.

Die folgenden Umgebungsvariable werden von der Umgebung des qsub-Kommandos weitergereicht:

Variable	Wert (Bsp)	Beschreibung
PBS_ENVIRONMENT	PBS_BATCH	set to PBS_BATCH to indicate that the job is a batch job; otherwise set to PBS_INTERACTIVE to indicate that the job is a PBS interactive job
PBS_JOBID	165139.Athene2	the job identifier assigned to the job by the batch system
PBS_JOBNAME	batch.sh	the job name supplied by the user
PBS_NODEFILE	/var/spool/torque/aux//165139.Athene2	the name of the file that contains the list of the nodes assigned to the job
PBS_QUEUE	common	the name of the queue from which the job is executed
PBS_TMPDIR		

Die Umgebungsvariable X aus der Umgebung des qsub-Kommandos wird als Variable PBS_O_X weitergereicht.

Variable	Wert (Bsp)	Beschreibung
PBS_O_HOME	/home/brf09510	HOME variable in the evvironm. in which qsub was executed
PBS_O_LANG	en_US.UTF-8	LANG variable in the evvironm. in which qsub was executed
PBS_O_LOGNAME	brf09510	LOGNAME variable in the evvironm. in which qsub was executed
PBS_O_PATH	/usr/bin:/bin:/	PATH variable in the evvironm. in which qsub was executed
PBS_O_MAIL	/var/mail/brf09510	MAIL variable in the evvironm. in which qsub was executed
PBS_O_SHELL	/bin/tcsh	SHELL variable in the evvironm. in which qsub was executed
PBS_O_TZ		TZ variable in the evvironm. in which qsub was executed
PBS_O_HOST	Athene1.uni-regensburg.de	the name of the host upon which the qsub command is running.
PBS_SERVER	Athene1.uni-regensburg.de	the hostname of the pbs_server which qsub submits the job to.
PBS_O_QUEUE	batch	the name of the original queue to which the job was submitted.
PBS_ARRAYID		each member of a job array is assigned a unique identifier (see -t)
PBS_O_WORKDIR	/home/brf09510	the abs. path of the current working dir. of the qsub command

Zusätzlich werden die folgenden Umgebungsvariable definiert:

Variable	Wert (Bsp)	Beschreibung
PBS_O_HOST	Athene1.uni-regensburg.de	the name of the host upon which the qsub command is running
PBS_O_QUEUE	batch	the name of the original queue to which the job was submitted
PBS_O_WORKDIR	/home/brf09510	the absolute path of the current working directory of the qsub command

3. Maui

KAPITEL 10

Debuggen

Intel

`/opt/intel/Compiler/11.0/074/Documentation/idb`

`/opt/intel/Compiler/11.0/074/idb/Getting_Started.html`

`http://www.uni-r.de/EDV/kurs_info/brf09510/hpc/opt/intel/Documentation/idb/Getting_Started.html`

`http://www.uni-r.de/EDV/kurs_info/brf09510/hpc/opt/intel/Documentation/idb/debugger_documentation.html`

KAPITEL 11

Weitere Details

Parastation zur Steuerung des Clusters (Process Management)

Gridmonitor

Webseite: <http://cluster-competence-center.com/support.php>

Glossar

AMD: CPU-Hersteller (K10).

ATLAS: Automatically Tuned Linear Algebra Software. Zur Zeit stehen C- und F77-Schnittstellen bereit, die effiziente parallele BLAS- und LAPACK-Funktionen aufrufen.

Bandbreite (*bandwidth*): Maximale Frequenz, mit der Daten über eine Verbindung des Netzes geschickt werden können. Einheit: Byte pro Sekunde. Daraus folgt die Übertragungszeit t für n Byte mit der Bandbreite b : $t = n/b$.

Barcelona: Quadcore-Chip von AMD mit vier K10-Architekturen.

BSP: Bulk Synchronous Programming; älteres Software-Modell mit Message Passing; <http://www.bsp-worldwide.org/>

CAF: siehe Co-Array Fortran

ccNUMA: Cluster-Architektur (Cache Coherent Nonuniform Memory Access). Jede CPU verfügt über einen Cache zum Speicher; manchmal teilen sich CPU-Gruppen einen gemeinsamen Cache. Jede CPU kann aber prinzipiell (schlimmstenfalls nach Wartezeit) auf jede Speicherzelle zugreifen. Die Caches werden automatisch synchronisiert. Der Cache beschleunigt die Speicherzugriffe im Vergleich zu NUMA.

Chapel: Eine PGAS-Sprache von Cray

<http://chapel.cray.com/spec/spec-0.795.pdf>

<http://chapel.cray.com/>.

CMF: nicht mehr aktuelles Software-Modell.

Co-Array Fortran: PGAS-Sprache;

<http://www.nag.co.uk/SC22WG5/>.

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/pgascaf.html (Skript)

CUDA: Software-Modell von NVidia (siehe GPU).

Core: CPU-Kern (Unabhängig agierende ALU/FPU, die ein Programm eigenständig abarbeiten kann).

DARPA: Defense Advanced Research Projects Agency (U.S Department of Defense institute) Urheber eines Projektes für den Entwurf neuer paralleler Programmiersprachen 2002. Seit 2006 konkurrieren noch die Entwürfe X10 (IBM), Chapel (Cray) und Fortress (Sun).

http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0706.pdf.

Distributed Memory: Cluster-Architektur. Jeder Prozessor hat seinen eigenen Speicher. Datenaustausch zwischen den Prozessoren erfolgt über ein unabhängiges Netz.

DM/DMEM: Distributed memory; viele Prozessoren haben je einen eigenen Speicher; vermeidet Flaschenhalse; erfordert Kommunikation.

DMP: Distributed Memory Processing; Jede CPU ist mit eigenem Speicher ausgestattet. Alternative zu SMP.

Fortress: PGAS-Sprache von Sun.

<http://projectfortress.sun.com/Projects/Community/>

<http://research.sun.com/projects/plrg/fortress.pdf>

<http://research.sun.com/projects/plrg/Publications/GSPx-Lecture2006public.pdf>.

GAS: Global address space

GASNET: Global-Address Space Networking. GASNet is a language-independent, low-level networking layer that provides network-independent, high-performance communication primitives tailored for implementing parallel global address space SPMD languages such as UPC, Titanium, and Co-Array Fortran. The interface is primarily intended as a compilation target and for use by runtime library writers (as opposed to end users), and the primary goals are high performance, interface portability, and expressiveness.

<http://gasnet.cs.berkeley.edu/>

GigE: Gigabit Ethernet.

GPU-Programming (siehe CUDA): Graphik-Prozessor.

HPF: High Performance Fortran.

IB: Infiniband.

IPMI: Intelligent Platform Management Interface - Protokoll.

K10: CPU-Architektur von AMD (auch AMD Opteron (K10)).

Latenz: Verweilzeit einer Nachricht im Netz. Latenz T für n Byte mit Bandbreite b bei einem Overhead o : $T = o + n/b$. Der Overhead besteht aus den meist konstanten Zeiten, die zum Start und Empfang der Nachricht benötigt werden und der Signallaufzeit für 1 Bit.

MIMD: (Flynn); Multiple Instruction Multiple Data; mehrere unabhängige Prozessoren bearbeiten mehrere Befehlsströme mit unabhängigen Daten. Je nach Speicherarchitektur unterscheidet man SM/SHMEM, DM/D-MEM und VSM.

MISD: (Flynn); Multiple Instruction Single Data; eigentlich überflüssige Einstufung, die nur zur Vervollständigung von SISD, SIMD, MISD, MIMD eingeführt wurde. Um dem Ausdruck nachträglich doch noch Sinn zu verleihen, wurden hier Pipeline-Architekturen, die ja hintereinander mit demselben Datum arbeiten, subsummiert. Eigentlich jedoch liegt aber nach jedem Pipeline-Schritt ein anderes modifiziertes Datum vor. Weiter werden hier fehlerreduzierende Mehrfacharchitekturen genannt, wo mehrere Prozessoren mit demselben Datum rechnen, deren Ergebnis nur akzeptiert wird, wenn es überall gleich ist.

MPI: Message Passing Interface (MPI-1, MPI-2); wichtigstes Software-Modell <http://www.mpi-forum.org/docs/>.

MPICH Das CH von MPICH kommt von CHameleon, eine Bezeichnung für die Portabilitätsschicht im ursprünglichen MPICH, die den Anschluss an beliebige Message-Passing-Systeme gewährleistet.

http://wiki.mcs.anl.gov/mpich2/index.php/Frequently_Asked_Questions

MVAPICH Der Name MVAPICH signalisiert, dass die zugrundeliegende MPI-Implementierung MPICH die InfiniBand VAPI-Schnittstelle benutzt. Die offizielle Aussprache ist Emwähpitsch.

<http://mvapich.cse.ohio-state.edu/>

<http://mvapich.cse.ohio-state.edu/support/faq.shtml>

Node: Knotenrechner mit ggf mehreren CPU-Kernen.

NUMA: Cluster-Architektur. (Non Uniform Memory Access). Vielen Prozessoren ist je ein eigener Speicher zugeordnet, aber prinzipiell kann jeder Prozessor, ggf. nach einer Wartezeit, jede Speicherzelle, auch die anderer Prozessoren, benutzen.

OpenMP: Open Multiprocessing; Thread-basiertes Software-Modell <http://openmp.org/wp/>.

Open MPI: Open Source MPI-Implementierung. Jünger als MPICH. Vorsicht: hat soviel mit OpenMP zu tun wie Javascript mit Java! <http://www.open-mpi.org/>

PBS: Portable Batch System.

PGAS: Partitioned global address space; Software-Modell mit global nutzbarem, aber bereichsweise den Prozessoren zugeteiltem Speicher.

PThreads: Thread-Modell für C von POSIX.

PVM: Parallel Virtual Machine; Software-Modell mit Message Passing <http://www.csm.ornl.gov/pvm/>.

Shebang, auch Magic Line: mit `#!` beginnende Scriptzeile, die als Kommando mit der Scriptdatei als Argument ausgeführt wird.

SISD: (Flynn); Single Instruction Single Data; **der** klassische sequentielle Rechner.

SIMD: (Flynn); Single Instruction Multiple Data; Ein Befehlswerk holt und analysiert Befehle. Die eigentliche Ausführung wird von einem Array mehrerer Prozessoren übernommen, die dann mit mehreren Datenströmen unabhängig dieselbe Befehlssequenz durchführen können.

SHMEM/SM: SHared MEMory; viele Prozessoren teilen sich einen gemeinsamen Speicher; Flaschenhals bei Speicherzugriff; Notwendigkeit der Synchronisation von Schreibzugriffen; kein Datenaustausch zwischen den Prozessoren nötig (keine Kommunikation). SHMEM ist auch eine Kommunikationbibliothek von Cray.

SM/SHMEM: siehe SHMEM.

SMP: Cluster-Architektur. (Symmetric MultiProcessing). Viele Prozessoren greifen gleichberechtigt auf alle restlichen Ressourcen, insbesondere den gemeinsamen Speicher, zu. Der gemeinsame Datenbus bildet hier einen extremen Flaschenhals. Alternative zu DMP.

Split-C: GAS-Sprache (PGAS ohne P) aus Berkeley <http://www.cs.berkeley.edu/~yelick/arvindk/splitc-super93>

Titanium: Java-basierte PGAS-Sprache.

<http://titanium.cs.berkeley.edu/>

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/pgastit.html (Skript)

Torque: Terascale Open source Resource and QUEUE manager.

UMA: Cluster-Architektur. (Uniform Memory Access): Viele Prozessoren greifen gleichberechtigt über mehrere Zugänge auf den gemeinsamen Speicher zu. Das beschränkt den Flaschenhals von SMP auf die Fälle, in denen auf Daten in derselben Speicherbank zugegriffen wird.

UPC: C-basierte PGAS-Sprache;

http://upc.gwu.edu/docs/upc_specs_1.2.pdf.

http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/pgasupc.html (Skript)

VSM: Virtual Shared Memory; Oberbegriff zu NUMA und PGAS

X10: PGAS-Sprache von IBM; Java-basiert;

<http://x10-lang.org/>

<http://dist.codehaus.org/x10/documentation/languagespec/x10-latest.pdf>

http://domino.research.ibm.com/comm/research_projects.nsf/pages/x10.index.html.

Inhalt

Inhaltsverzeichnis

Kapitel 1. What's New — Neuigkeiten	3
Kapitel 2. Probleme	5
Kapitel 3. Schnelleinstieg	7
1. Account	7
2. Login	7
3. Wahl von Compiler und MPI-Library	8
4. Übersetzung	9
5. Start von HPC-Jobs	10
Kapitel 4. Allgemeines	13
1. Zeittafel	13
Kapitel 5. Hardware	15
1. Rechner	15
2. Netz	17
3. Dateisystem	17
Kapitel 6. Betriebssystem	19
Kapitel 7. Sprachen und Compiler	21
1. FSF/Gnu	21
2. Intel	21
3. OpenMP	21
4. PGAS	21
Kapitel 8. Bibliotheken	23
1. Parallelrechnen	23
2. Atlas	23
3. MKL Math Kernel Library	24
Kapitel 9. Torque und Maui	25
1. Torque	25
2. PBS	25
3. Maui	26
Kapitel 10. Debuggen	27
Kapitel 11. Weitere Details	29
Kapitel 12. Glossar	31
Inhalt	35

TTH-Seite:

<http://hutchinson.belmont.ma.us/tth/>