Anleitung für PVM, Glasgow parallel Haskell (GpH) und GranSim:

1. PVM

PVM (Parallel Virtual Machine) ist eine Bibliothek, die das Versenden und Empfangen von Nachrichten zwischen verschiedenen Programmen auf einem Rechnerverbund ermöglicht. Diese Bibliothek wird von den parallelen Haskell-Versionen benutzt, die wir in der Vorlesung verwenden.

A. Konfiguration:[1]

Die Shell-Konfiguration (`~/.tcshrc`, bitte leer anlegen, falls nicht vorhanden) muss um die folgenden Zeilen ergänzt werden:

```
<.tcshrc>
setenv PVM_ROOT /app/lang/parallel/pvm3
setenv PVM_ARCH `$PVM_ROOT/lib/pvmgetarch`
setenv PVM_DPATH $PVM_ROOT/lib/pvmd
```

und evtl. (nur für die Man-Pages):

```
<.tcshrc>
setenv MANPATH ${MANPATH}:${PVM_ROOT}/man
```

Außerdem ist ein Alias für die PVM-Konsole sinnvoll:

```
<.tcshrc>
alias pvm ${PVM_ROOT}/lib/pvm
```

B. Bedienung:

Die PVM-Konsole startet (bzw. bedient) einen PVM-Rechnerverbund.

- Start des Systems (mit dem definierten Alias):

```
berthold@windhoek:~> pvm
pvm>
```

(Wir sind in der PVM-Konsole, mit der das System bedient werden kann)

- Aktuelle Konfiguration anzeigen:

```
pvm> conf
conf
1 host, 1 data format
                    HOST     DTID     ARCH   SPEED       DSIG
                 windhoek    40000    LINUX    1000 0x00408841
```

- Neue Rechner hinzufügen:

```
pvm> add mombasa lome tunis
add mombasa lome tunis
3 successful
```

[1] Alle beschriebenen Konfigurationen beziehen sich auf die TC-Shell. Wer eine andere benutzt, kennt sich bestimmt aus.

```
                    HOST     DTID
                 mombasa    80000
                    lome    c0000
                   tunis   100000
pvm> conf
conf
4 hosts, 1 data format
                    HOST     DTID     ARCH   SPEED       DSIG
                 windhoek    40000    LINUX    1000 0x00408841
                  mombasa    80000    LINUX    1000 0x00408841
                     lome    c0000    LINUX    1000 0x00408841
                    tunis   100000    LINUX    1000 0x00408841
```

Es ist auch möglich (nur bei erstem Start), ein Hostfile für PVM anzugeben. Es enthält die Rechner, die beim Start in den Verbund aufgenommen werden sollen (Kommentare: #…).

```
berthold@windhoek:~> cat pvmhosts
# SR auf D5:
asmara
algiers
# SR auf D4:
tripoli
lome
kampala
tunis
lusaka
rabat
berthold@windhoek:~> pvm pvmhosts
pvm> conf
conf
10 hosts, 1 data format
                    HOST     DTID     ARCH   SPEED       DSIG
                 windhoek    40000    LINUX    1000 0x00408841
                   asmara    80000    LINUX    1000 0x00408841
                  algiers    c0000    LINUX    1000 0x00408841
                  tripoli   100000    LINUX    1000 0x00408841
                     lome   140000    LINUX    1000 0x00408841
                  kampala   180000    LINUX    1000 0x00408841
                    tunis   1c0000    LINUX    1000 0x00408841
                   lusaka   200000    LINUX    1000 0x00408841
                    rabat   240000    LINUX    1000 0x00408841
                  mombasa   280000    LINUX    1000 0x00408841
pvm>
```

- Die Konsole verlassen (PVM-System läuft weiter):

```
pvm> quit
quit
Console: exit handler called
pvmd still running.
berthold@windhoek:~>
```

- Noch eine Konsole (System läuft bereits):

```
berthold@windhoek:~> pvm
pvmd already running.
pvm>
```

- System anhalten:

```
pvm> halt
halt
Terminated
berthold@windhoek:~>
```

Die Datei `/tmp/pvml.<Benutzernummer>` enthält Ausgaben des PVM-Systems auf anderen Rechnern (insbes. Ausgaben der gestarteten Programme). Das sollte aber für uns egal sein, nur bei Fehlern interessant. `/tmp/pvmd.<Benutzernummer>` ist eine Art Semaphor, damit nicht

mehrere PVMs auf einem Rechner laufen. (Falls der Rechner bei laufendem PVM Probleme bekommt (z.B. Reboot), bleibt diese Datei stehen => Fehler beim Start vom PVM)

2. GHC

GHC (Glasgow Haskell Compiler) ist die Basis aller behandelten parallelen Varianten von Haskell. Der aktuellste GHC auf dem Netz des Fachbereichs befindet sich in

```
/app/lang/functional/ghc-5.00.2-eden/bin
```

Das Verzeichnis `/app/lang/functional` enthält weitere (ältere) GHCs. Auf einen davon werden wir noch zurückkommen.

A. Konfiguration:

Für GHC 5.00.2 muss nichts Spezielles an der Konfiguration geändert werden (für die parallelen Varianten muss aber die PVM-Umgebung so eingerichtet sein wie oben beschrieben).

Um den GHC zu benutzen, setzt man am besten den Suchpfad darauf:

```
<.tcshrc>
setenv PATH /app/lang/functional/ghc-5.00.2-eden/bin:$PATH
```

oder auch temporär:

```
berthold@windhoek:~> set path=( /app/lang/functional/ghc-5.00.2-eden/bin $path )
berthold@windhoek:~> which ghc
/app/lang/functional/ghc-5.00.2-eden/bin/ghc
```

B. Bedienung:

Der GHC lässt sich wie jeder übliche Compiler von der Kommandozeile aus bedienen.

Für "stand-alone"-Programme (eine einzelne Haskell-Datei):

```
berthold@windhoek:~/haskell/test> ghc -o mandelbrot1 -O2 mandelbrot.hs
berthold@windhoek:~/haskell/test>
```

Dieses Beispiel benutzt dieOption

`-o <name>`: Name der Ausgabedatei

`-O2`: Optimierung Stufe 2 beim Übersetzen

Für Programme, die aus mehreren Modulen bestehen, bietet GHC seit Version 5 eine Option "`--make`" an. Damit werden automatisch die Modulabhängigkeiten geprüft und alle nötigen Module übersetzt (z.B. beim Programm aus dem ersten Tutorium):

```
berthold@windhoek:test/intro> ghc -o showprime -O2 ShowPrime.hs  --make
ghc-5.00.2: chasing modules from: ShowPrime.hs
Compiling Types             ( Types.hs, Types.o )
Compiling Main              ( ShowPrime.hs, ./ShowPrime.o )
ghc: linking ...
berthold@windhoek:test/intro>
```

Andere "schöne" Optionen (siehe auch GHC-Doku, Teil 4.18):

`-W`	"normale" Warnungen
`-Wall`	sämtliche Warnungen (z.B. unbenutzte Variablen, fehlender Typ)
`-cpp`	Parsing mit dem C-Präprozessor, bevor übersetzt wird (z.B. #ifdef-direktiven)
`-D<var>`	setzt eine Variable für den C-Präprozessor
`-prof`	Version für Profiling-Informationen bauen

Die erzeugten Programme können bei Aufruf neben ihren Programm-Parametern auch Parameter für das Laufzeitsystem (RTS) erhalten. Diese Parameter (z.B. Profiling) werden in `+RTS…-RTS` eingeschlossen. Je nach Übersetzungsoption sind verschiedene Parameter möglich, z.B.:

`-G<n>`	Anzahl Generationen in der garbage collection
`-M<size>`	max. Heapgröße
`-s<file>`	garbage collection Statistik in `<file>` schreiben
`-P`	detailliertes Profiling (falls mit Profiling-Option `–prof` übersetzt)

```
berthold@windhoek:test/intro> showprime +RTS -G6 -P -M2M -sstderr -RTS 50000
showprime 50000 +RTS -G6 -P -M2M -sstderr
Heap exhausted;
Current maximum heap size is 1998848 bytes;
use `+RTS -M<size>' to increase it.
berthold@windhoek:test/intro>
berthold@windhoek:test/intro> showprime +RTS -G6 -P -M8M -sstderr -RTS 50000
showprime 50000 +RTS -G6 -P -M8M -sstderr
Primzahl Nr. 50000 ist 611953
262,000,152 bytes allocated in the heap
  2,492,200 bytes copied during GC
  1,036,572 bytes maximum residency (2 sample(s))

        999 collections in generation 0 (  0.06s)
          2 collections in generation 1 (  0.00s)
          2 collections in generation 2 (  0.00s)
          2 collections in generation 3 (  0.00s)
          2 collections in generation 4 (  0.00s)
          2 collections in generation 5 (  0.02s)

          3 Mb total memory in use

  INIT  time    0.00s  (  0.01s elapsed)
  MUT   time    3.86s  (  3.90s elapsed)
  GC    time    0.08s  (  0.09s elapsed)
  EXIT  time    0.00s  (  0.00s elapsed)
  Total time    3.94s  (  4.00s elapsed)

  %GC time       0.0%  (0.0% elapsed)
  Alloc rate    67,875,687 bytes per MUT second
  Productivity  98.0% of total user, 96.5% of total elapsed
```

3. Glasgow Parallel Haskell

Gph-Programme (die sich durch bestimmte Spracherweiterungen auszeichnen, siehe VL) werden mit der Option "`-parallel`" übersetzt. Das Modul `Parallel` muss im Programm importiert werden, um die parallelen Erweiterungen auszunutzen.

Als Beispiel kann für's erste ein (schlecht) parallelisiertes Quicksort dienen, das die binäre Rekursion parallel berechnet.

```
berthold@windhoek:~> ghc -parallel -O2 -o parquicksort parquicksort.hs -cpp -DPARVERSION
/app/lang/parallel/pvm3/lib/LINUX/libpvm3.a(lpvm.o): In function `pvm_tc_conreq':
```

```
lpvm.o(.text+0x897): the use of `tmpnam' is dangerous, better use `mkstemp'
berthold@windhoek:~>
```

Das Laufzeitsystem dieser Programme benutzt PVM und lässt sich über RTS-Parameter steuern:

`-q...` leitet eine GpH-Option ein

`-qp<n>` Programm auf n (virtuellen) PVM-Knoten laufen lassen

`-qQ<n>` max. Paketgröße für Kommunikation

`-qt<n>` Anzahl Threads auf einem Knoten

(weitere: siehe GpH-Anleitung)

```
berthold@windhoek:~> pvm ~/public_work/pvmhosts
pvm> conf
conf
10 hosts, 1 data format
                     HOST     DTID     ARCH   SPEED       DSIG
                  windhoek    40000    LINUX    1000 0x00408841
                    asmara    80000    LINUX    1000 0x00408841
                   algiers    c0000    LINUX    1000 0x00408841
                   tripoli   100000    LINUX    1000 0x00408841
                      lome   140000    LINUX    1000 0x00408841
                   kampala   180000    LINUX    1000 0x00408841
                     tunis   1c0000    LINUX    1000 0x00408841
                    lusaka   200000    LINUX    1000 0x00408841
                     rabat   240000    LINUX    1000 0x00408841
                   mombasa   280000    LINUX    1000 0x00408841
pvm> quit
quit

Console: exit handler called
pvmd still running.
berthold@windhoek:~> parquicksort bsp.txt +RTS -qp8 -qt2
==== Starting parallel execution on 8 processors ...
Lese bsp.txt
Sortiere
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0

00001111222233334444555566667777888899999
berthold@windhoek:~>
```

4. GranSim

GranSim (Granularity Simulation) ist ein Simulator für GpH, mit dem man grafische Aktivitätsprofile erzeugen kann. Da GranSim in Version 4ff des GHC nicht mehr enthalten ist, müssen wir den alten GHC in `/app/lang/functional/ghc-3.02/bin` benutzen. In diesem Verzeichnis befindet sich statt „`ghc`“ nur der versionierte Compiler „`ghc-3.02`“.[2]

```
berthold@windhoek:haskell/test> set path=( /app/lang/functional/ghc-3.02/bin $path )
berthold@windhoek:haskell/test>
```

GranSim-Programme (im Prinzip Programme für GpH) werden mit den Optionen `-gransim -fvia-C` übersetzt.

[2] Dieses Verzeichnis kann man auch als Standard in den Suchpfad legen. Da sich dort aber ein (uralter !) gcc befindet, ist das nicht zu empfehlen. (besser immer explizit „zuschalten“).

```
berthold@windhoek:~> ghc-3.02 -gransim -O2 -o gran.parqs parquicksort.hs -cpp -
DPARVERSION

NOTE: Simplifier still going after 4 iterations; bailing out.

NOTE: Simplifier still going after 4 iterations; bailing out.
ghc: module version changed to 1; reason: no old .hi file
berthold@windhoek:~>
```

Das übersetzte Programm wird (mindestens) mit der RTS-Option "`-bP`" (Profil erzeugen) ausgeführt; hier zusätzlich "`-bp<n>`" (n Prozessoren). Resultat ist eine Datei *.gr, die mit dem Script `gr2ps` in eine PostScript-Datei mit Grafik umgewandelt und dann (zum Beispiel) mit `ghostview` betrachtet werden kann.

```
berthold@windhoek:~> gran.parqs bsp.txt +RTS -bP -bp8
Lese bsp.txt
Sortiere
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0

00001111222233334444555566667777888899999
Simulation finished after @ 746181 @ cycles. Look at gran.parqs.gr for details.
berthold@windhoek:~> ls *gr
gran.parqs.gr
berthold@windhoek:~> gr2ps gran.parqs.gr
berthold@windhoek:~> gv -seascape gran.parqs.ps
berthold@windhoek:~>
```

GranSim bietet zahlreiche andere Optionen, etwa um die Kommunikationskosten festzulegen (siehe Anleitung). Wir werden aber in der Regel nur die Prozessorzahl variieren.

Nochmal auf die Schnelle:

<home>/.tcshrc sollte enthalten:

```
setenv PVM_ROOT /app/lang/parallel/pvm3
setenv PVM_ARCH `$PVM_ROOT/lib/pvmgetarch`
setenv PVM_DPATH $PVM_ROOT/lib/pvmd
setenv MANPATH ${MANPATH}:${PVM_ROOT}/man
alias pvm ${PVM_ROOT}/lib/pvm
setenv PATH /app/lang/functional/ghc-5.00.2-eden/bin:$PATH
```

Um mit GranSim zu arbeiten, setzt man (temporär) den Pfad um: (Aufruf dann „ghc-3.02“)

```
%Shell Prompt%~> set path=( /app/lang/functional/ghc-3.02/bin $path )
```