PAW Installationsanleitung

4. Dezember 2007

Inhaltsverzeichnis

Kapitel 1

Installationsanleitung

1.1 PAW Distribution ablegen

Zunächst lädt man eine PAW Distribution herunter. Man findet sie auf der PAW Webseite

`https://orion.pt.tu-clausthal.de/paw`

wenn man dem Link "Download" folgt. Der Zugang zu den Distributionen erfordert eine gültige Lizenz, die man auf derselben Webseite beantragen kann. Mit einem gültigen Passwort erhält man Zugang zu den Programmen und den sogennannten "Setups". Letztere sind Datenfiles, die für das Ausführen von PAW benötigt werden.

Unter der Rubrik "Sources" liegt die eigentliche Distribution. Wir bezeichnen das File im Folgenden mit ***paw-distribution***. Man sollte die neueste Version herunterladen. (Obwohl die vorliegenden Versionen mit Beta bezeichnet werden, ist dies das verlässlichste Niveau. Wir wollen uns nur die Möglichkeit einer stringenter getesteten Version vorbehalten.)

Die PAW Distribution liegt als komprimiertes TAR Archiv vor. mit dem Kommando

`tar tvfz` ***paw-distribution***

erhält man eine Aufstellung der in der Distribution enthaltenen Files. Das Archiv wird mit dem Befehl

`tar xvfz` ***paw-distribution***

in das aktuelle Verzeichnis entpackt. Das dabei erzeugte Verzeichnis "paw-beta" bezeichnen wir im Folgenden als ***paw-directory***.

Auf Anderen als Linuxsystemen muss man die Distribution erst umbenennen, sodass die Endung `.tar.gz` und nicht `.tgz` heißt. Anschließend unzipped man das file mit

`gunzip` ***paw-distribution***.tar.gz

und entpackt es anschließend mit

`tar xvf` ***paw-distribution***.tar

Das Verzeichnis ***paw-directory*** sollte etwa wie folgt aussehen:

```
configure
configure.ac
dx
Makefile_bare.in
Makefile_targets.in
parameters
parms.example
parms.g95
README
src
```

Anschließend lädt man eine Distribution der Setupfiles von obiger Webseite herunter und entpackt dieses File entsprechend. ***paw-setupdir*** Diese Files werden erst beim Ausführen des PAW Programms wieder benötigt.

1.2 Compiler und Bibliotheken installieren

Man benötigt

- einen Fortran90 Compiler,
- eine Bibliothek für Lineare Algebra, welches der BLAS entspricht (BLAS, ACML, MKL, ESSL)
- eine Bibliothek für Lineare Algebra, welches der LAPACK entspricht (LAPACK, ACML, MKL, ESSL)
- eine Bibliothek für Fouriertransformationen (FFTW, ACML, MKL, ESSL)
- ein Message Passing Interface (MPICH, MVAPICH)
- evtl eine Utility Bibliothek (Compilerabhängig)

Der Leser sollte die Installationen vornehmen. Unsere Erfahrungen mit dem Compilern und den Bibliotheken sind in Kapitel 2 beschrieben.

1.3 Parameterfile anpassen

Als nächstes muss die Distribution configuriert werden. Bei der Konfiguration werden an das System angepasste make-files für die Installation der Distribution erstellt. Etwas Hintergrund zum Konfigurationsprozess im Allgemeinen findet man in Kapitel 5.17.

Für die Konfiguration wird ein Parameterfile benötigt, das zunächst angepasst werden muss. Wir nennen es im Folgenden "***parmfile***". Das Parameterfile der PAW Distribution ist vergleichsweise explizit und durchsucht das System nur geringfügig.

Am besten geht man von Beispielen aus, die auf der PAW Downloadpage zu finden sind, bzw im Anhang B für verschiedene Beispielkonfigurationen angegeben sind. Eine Beschreibung der Parameter findet man in Kapitel 3

Man sollte darauf achten, dass sich der Aufbau der Parameterfiles mit der Zeit noch verändern werden.

1.4 Konfigurieren

Mit dem Parameterfile kann man die Konfiguration mit dem Befehl

`./configure --with-parmfile=`***parmfile***

in der ***paw-directory*** durchführen. Dies erzeugt unter anderem Makfiles und enstrechenden Unterverzeichnisse.

Eine erfolgreiche Konfiguration wird mit der Meldung:

```
----done!---configuration completed successfully!--------------
```

beendet. Ansonsten erscheint eine entsprechende Fehlermeldung. Einen typischen Ausdruck findet man in Anhang A.

Das Configure Skript erzeugt ein Verzeichnis

paw-directory`/bin/`***arch***

Der Name ***arch*** sollte für jedes Parameterfile eindeutig gewählt werden. Dies erlaubt es verschiedene Installationen nebeneinander zu halten.

1.5 Make

Indem man in der PAW-Directory das Kommando

`make`

ausführt, wird die PAW Distribution übersetzt und installiert.

Es ist ratsam die Installation nicht wie eben beschrieben in einem Schritt durchzuführen, sondern beim ersten mal die folgende Reihenfolge einzuhalten, um mögliche Probleme frühzeitig zu erkennen.

1. Man erzeugt die Dokumentation in ***paw-directory***`/doc` durch

   `make docs`

2. als Nächstes erzeugt man ein Binary ohne spezielle Compileroptionen

   `make none`

   Das Binary wird als ***paw-directory/arch***`/paw.x` abgelegt

3. Nun erzeugt man sich eine Installation zur Produktion.

   `make fast`

   Das binary wird als ***paw-directory/arch***`/paw_fast.x` abgelegt.Es ist ratsam die Resultate von `paw_fast.x` mit denen von `paw.x` zu vergleichen, weil die Optimierungen unter Umständen die Resultate verfälschen.

4. Will man einen Parallelrechner nutzen, erzeugt man sich eine Installation zur Produktion auf Parallelrechnern

   ```
   make fast_parallel
   ```

   Das binary wird als ***paw-directory/arch***/`ppaw_fast.x` abgelegt

5. Schließlich erzeugt man sich Analyseprogramme mit

   ```
   make tools
   ```

6. Möchte man eine Vollinstallation, kann man nun auch

   ```
   make
   ```

   ausführen, um die anderen Binaries zu erzeugen.

Kapitel 2

Benötigte Software

Neben dem PAW Programm und Systemfunktionen wird für die Installation wird

- ein Fortran compiler (Siehe Kapitel 2.1)

benötigt.
Des weiteren benötigt man einen Satz von Bibliotheken (Siehe Kapitel 2.3)

- Bibliothek für Lineare Algebra
- Bibliothe für Fouriertransformationen
- Message Passing Interface (MPI)

Für bestimmte Compiler benötigt man eine

- Utility Library (siehe Kapitel 2.2)

Hinzu kommen

- ein GNU C-Preprocessor `cpp`
- Das GNU Make Tool. (Auf AIX wird das GNU Make mit `gmake` bezeichnet. Das Normale `make` wird bei unseren Makefiles crashen.)

2.1 Fortran Compiler

Zum Compilieren von PAW wird ein Fortran90 Compiler benötigt.(Spätere Versionen wie Fortran 95 etc. können entsprechend genutzt werden.) Nachfolgend werden einige Compiler aufgeführt, soweit wir damit Erfahrung gesammelt haben.

2.1.1 G95

- Quelle: *http://www.g95.org*
- Probleme: keine bekannten Probleme.
- Anmerkungen:

  - GNU Lizenz (open source)
  - Um PAW für 64 Bit Maschinen der Architektur x86_64 oder EMT64 zu compilieren, wurde bisher die Version benutzt, welche auf der Downloadseite mit

    Linux x86_64/EMT64 (32 bit D.I.)

    bezeichnet wird. Das ist wahrscheinlich nicht mehr nötig.

- Beim G95 muss die g2c Bibliothek gelinkt werden. (Angeblich ist das jetzt nicht mehr nötig. Das muss überprüft werden.)

2.1.2 PGI Fortran Compiler

- Hersteller: The Portland Group
- Quelle:*http://www.pgroup.com/*
- Lizenz: kommerziell
- Probleme: keine bekannten Probleme
- Anmerkungen:
  - enthält einige Bibliotheken wie LAPACK,ACML,BLAS,MPICH

2.1.3 IFORT

- Hersteller: Intel
- Quelle:*https://welcome.intel.com/Login.aspx*. Dort muss man sich registrieren. Anschließend nach "Intel Fortran Compiler" suchen.
- Anmerkungen:
  - Sehr schnell.
  - Produziert mitunter besonders großen Code. Dies kann zu einem Überschreiten des Stackgröße führen, was sich in unvorhersagbarem Verhalten äußert. In diesem Fall sollte die Stackgröße mit `ulimit -s unlimited` vergrößert werden. Die Stackgröße kann mit `ulimit -a` kontrolliert werden.
- Probleme: keine bekannten Probleme.

2.1.4 XLF Compiler

- Hersteller: IBM
- Quelle: *http://www-306.ibm.com/software/awdtools/fortran/xlfortran/features/xlf-linux.html*
- Lizenz: kommerziell
- Anmerkung: Nur für Power Architekture von IBM
- Probleme: Keine bekannten Probleme

2.1.5 gfortran

- Hersteller: Free Software Foundation
- Quelle:*http://gcc.gnu.org/fortran/*
- Probleme: keine erfolgreiche Compilierung.

2.1.6 Absoft Fortran

- Hersteller: Absoft
- http://www.absoft.com/
- Probleme: Es ist uns nicht gelungen eine lauffähige PAW Version zu erzeugen.

2.2 Utility Bibliothek

Es gibt einige Routinen die nicht im Fortran Standard enthalten sind, aber bei allen Compilern in ähnlicher Form als Bibliothek eingebunden werden können oder direkt als Fortranerweiterungen behandelt werden.

Die Utility Bibliothek wird ganz unterschiedlich benannt. IBM nennt sie "Service and Utility Bibliothek", Intel nennt sie "Portability Routinen", Traditionell heißt die Utility Bibliothek U77, wobei 77 auf den Fortran77 Standard verweist.

Obwohl die Bibliotheksaufrufe fast identisch sind, unterscheiden sie sich dennoch im Kind-Parameter der Argumente. Deshalb besitzt PAW Interfaces zu den Utility Bibliotheken der einzelnen Compiler. Diese werden über den C-Preprozessor ausgewählt. Hierzu müssen die entsprechenden Parameter im Parameterfile für die Konfiguration gesetzt sein.

Ob eine Utility Bibliothek speziell eingebunden werden muss erfährt man im User Guide des enstprechenden Conmpilers.

2.3 Numerische Bibliotheken

Es werden Bibliotheken mit drei unterschiedlichen Funktionen benötigt, die manchmal in größeren Paketen zusammengefasst sind. Diese sind:

- Basic Linear Algebra Subroutines (BLAS): Mathematische Bibliothek mit elementaren Operationen der linearen Algebra wie Vektor- und Matrixmultiplikationen.
- Lineare Algebra Bibliothek (LAPACK): Komplexere Matrixoperationen wie z.B. Eigenwetprobleme
- Fourier Transformations Bibliothek (FFT): Fouriertransformationen.

Diese Bibliotheken sind für das Programm geschwindigkeitsbestimmend.

Für die Linearen Algebra Routinen gibt es Standardpakete, nämlich das Linear Algebra PACKage (LAPACK) und die Basic Linear Algebra Subprograms (BLAS) . Die Beschreibung der LAPACK-Routinen findet man unter

`http://www.netlib.org/lapack/`

und die von BLAS unter

`http://www.netlib.org/blas/`

Diese stellen sozusagen einen Standard für entsprechende Bibliotheken dar.

Diese Pakete sind zum Teil in speziellen Bibliotheken enthalten.

- IBM Engineering and Scientific Software Library (ESSL)
- AMD Core Math Library (ACML)
- Intel Math Kernel Library (MKL)

Es ist ratsam, nicht die mit dem Operating System mitgeliefereten Bibliotheken zu nutzen, sondern diese möglichst selber auf dem Zielsystem zu Kompilieren. Dies erzeugt schnellere Binaries.

Wir legen selbst erzeugte Bibliotheken in einer Directory `/home/tools` auf dem jeweiligen Rechner ab. Dies hat Vorteile bei der Installation von Parallelrechnern, bei denen die Bibliotheken nur auf dem Frontendrechner vorgehalten werden.

LAPACK	BLAS	FFT
LAPACK	ATLAS	FFTW
ACML	ACML	FFTW
ACML	ACML	ACML
MKL	MKL	FFTW(MKL)
MKL	MKL	FFTW
ESSL	ESSL	ESSL

2.3.1 ATLAS-BLAS

- Hersteller: Open Source
- Name: ATLAS=Automatically Tuned Linear Algebra Software
- Lizenz: Freie Software
- Quelle: *http://math-atlas.sourceforge.net/*
- Anmerkung: Diese Bibliothek wird nicht empfohlen. Anstelle dieser Bibliothek sollten die entsprechende Bibliotheken der Hardware Hersteller verwendet werden, die speziell auf ihre Hardware optimiert sind.
- Funktionen: BLAS
- Installation:
  - After unpacking the ATLAS distribution type make and follow the instructions. Always use the default value ([y] or [n] by just typing ENTER) until you arrive at

    `use express setup? [y]:`

    Enter `no` and proceed taking the defaults if you like so. Use `f90` as FORTRAN77 compiler (just needed to compile the wrappers). As `F77 FLAGS` use

    `-YEXT_NAMES=LCS -YEXT_SFX=_ -O`

    to ensure, that the linking works out.
    Again take the default values until you reach the

    `Enter C Flags (CCFLAGS) [-fomit-frame-pointer -O3 -funroll-all-loops]:`

    prompt. Just use `-fomit-frame-pointer -O` and proceed accepting the defaults.
    If you have compiled an ATLAS BLAS for different architectures (e.g. Pentium and ATHLON), the corresponding libraries will be in different subdirectories of the ATLAS distribution. You find these subdirectories in `ATLAS/lib`. If there is just one, the configure script will chose it automatically.

2.3.2 LAPACK

- Name: LAPACK=Linear Algebra PACKage
- Quelle: http://www.netlib.org/lapack/
- Funktionen: BLAS
- Anmerkung: Diese Bibliothek wird nicht empfohlen. Anstelle dieser Bibliothek sollten die entsprechende Bibliotheken der Hardware Hersteller verwendet werden, die speziell auf ihre Hardware optimiert sind.
- Installation:

1. Download *lapack-lite-3.1.1.tgz* von http://www.netlib.org/lapack/index.html
2. gunzip and untar the file
3. Copy and edit the file LAPACK/make.inc.example to LAPACK/make.inc.
4. Edit the file LAPACK/Makefile (see http://www.netlib.org/lapack/lawn81/node13.html)
5. type make

2.3.3 FFTW

- Hersteller: Massachusetts Institute of Technology (MIT)
- Name: FFTW=Fastest Fourier Transform in the West
- Quelle: *http://www.fftw.org/download.html*
- Anmerkungen
  - Die Versionen FFTW2 und FFTW3 sind nicht kompatibel. Gegenwärtig wird nur FFTW2 in PAW unterstützt.
- Installation:
  1. FFTW 2.1.5 von `http://www.fftw.org/download.html` herunterladen
  2. Archiv mit `tar -xvzf Dateiname` entpacken
  3. Den Befehl `./configure --enable-i386-hacks` ausführen. The option `--enable-i386-hacks` takes advantage if rge gcc/x86 specific performance hacks. (Beim core2duo muss die Option `--enable-i386-hacks` weggelassen werden!)
  4. Folgende Zeile in der *fftw/config.h* ändern (am Ende der Datei):

     ```
     \#define F77_FUNC_(name,NAME) name ## __
     ```

     dort den letzten Unterstrich entfernen, so dass daraus die folgende Zeile wird:

     ```
     \#define F77_FUNC_(name,NAME) name ## _
     ```
  5. Den Befehl *make* ausführen

2.3.4 ACML

- Name: ACML=AMD Core Math Library
- Hersteller: AMD
- Quelle *http://developer.amd.com/acml.jsp*
- Enthält: FFT, LAPACK, BLAS
- Installation: Zum Installieren einfach das Archiv entpacken und den Installationshinweisen folgen.

2.3.5 MKL

- Hersteller: Intel
- Name: MKL=Math Kernel Library
- Quelle: http://www.intel.com/cd/software/products/asmo-na/eng/307757.htm
- Installation:
  1. Anmeldung unter *https://welcome.intel.com/Login.aspx*
  2. http://www.intel.com/cd/software/products/asmo-na/eng/307757.htm
- Mit der MKL muss die Bibliothek "pthread" gelinkt werden. (Siehe Anhang 5.6)

FFTW der MKL

Die MKL enthält ein Interface für FFTW Aufrufe, und stellt damit auch die Routinen für Fouriertransformationen zu Verfügung.

Um die FFTW von der MKL nutzen zu können, muss man in das Unterverzeichnis *interfaces/fftw2xf* der MKL wechseln und dort das Kommando *make* ausführen. Benötigt wird dafür der Intel Fortran Compiler und der Intel C Compiler. Es wird dann die Bibliothek *libfftw2xf_intel.a* im entsprechenden *mkl-libs* Verzeichnis erzeugt, welche in PAW eingebunden werden kann, indem man die fftw Pfade im Parameterfile anpasst. Desweiteren muss unter den Punkten *LIBS_SCALAR* und *LIBS_PARALLEL* der Wert *fftw* durch *fftw2xf_intel* ersetzt werden.

2.3.6 ESSL

- Hersteller: IBM
- Quelle: *http://www-03.ibm.com/systems/p/software/essl/index.html*
- Anmerkung:
  - Nur für IBM Hardware. Nur für AIX operating system oder Linux für Power achitekture.
  - ESSL verwendet nicht dieselben Calling sequences wie LAPACK und BLAS. PAW besitzt aber spezielle Interfaces für ESSL.
- Covers: FFT, LAPACK, BLAS

2.4 Message Passing Interface

Das MPI (Message Passing Interface) ist ein Protokoll, um ein verteiltes (paralleles) Rechnen zu ermöglichen. Das MPI muss so compiliert sein, das seine Funktiosaufrufe mit einem Underscore funktionieren!

Die meisten Hardware Hersteller bieten eigene MPI Implementierungen an. Es gibt aber auch freie MPI implementierungen wie MPICH und Open MPI*http://www.open-mpi.org/*. Unsere Erfahrungen beschränken sich auf MPICH bzw MVAPICH, sowie die MPI von IBM.

2.4.1 MPICH

MPICH ist eine freie Implementierung des MPI für Ethernet.

- Hersteller: Argonne National Laboratory and Missisippi State University
- Quelle: http://www-unix.mcs.anl.gov/mpi/mpich1/
- Anmerkungen:
  - Inzwischen existitert eine neue Implementierung MPICH2, die heute empfohlen wird. Sie unterstützt Cluster aus Single und SMP Knoten. MPICH2 kann unter *http://www-unix.mcs.anl.gov/mpi/mpich2/index.htm* heruntergeladen werden.
- Installation:
  1. Download der neuesten version unter

     http://www-unix.mcs.anl.gov/mpi/mpich1/

  2. Nach dem Entpacken kann man mit folgendem Script MPI für PAW compilieren:

     ```
     #!/bin/sh
     export F90=g95
     export F90FLAGS=-fno-second-underscore
     export FC=g95
     export FFLAGS=-fno-second-underscore
     export FLINKER=g95
     export RSHCOMMAND=ssh
     ./configure --enable-f77
     ```

     Für Pathscale

     ```
     #!/bin/sh
     export CC=pathcc
     export FC=pathf90
     export F90=pathf90
     export F90FLAGS=-fno-second-underscore
     export FFLAGS=-fno-second-underscore
     export FLINKER=pathf90
     export RSHCOMMAND=ssh
     ```

```
./configure -c++=pathcc -opt=-O3 --enable-f90 --enable-f90modules \
            --with-romio --disable-weak-symbols
```

3. *make* ausführen

2.4.2 MVAPICH

Freie MPI Implementierung für ein Infiniband Netzwerk. Sie basiert auf MPICH.

- Hersteller: Ohio State University
- Quelle: *http://mvapich.cse.ohio-state.edu/*

Kapitel 3

Das Parameterfile

Das Parameter-File dient zur Konfiguration der Compilation von PAW. In der Datei wird der Compiler, dessen Optionen, die genutzten mathematischen Bibliotheken, die MPI und andere Dinge festgelegt, die zum Compilieren notwendig sind. Ein Grundgerüst für ein Parameter-File findet man in der Datei parms.example. Das Beispiel ist so ausgelegt, dass man es mit dem G95 auf einem 64 Bit Rechner verwenden kann, wenn im Verzeichnis */opt/libs/acml/gnu64* die ACML für GNU64 und in */opt/libs/fftw/* die fftw in der Version 2.1.5 vorliegt. Alle Variablen die angepasst werden müssen, sind im Folgendem unterstrichen.

3.1 Beispiel für ein Parameterfiles

```
#############################################################################
##________________________architecture (arbitrary name)_____________________##
ARCH="g95_guam"
##________________________flag for uppercase or lower case module file names____##
TUPPERCASEMOD="F"
##________________________flag for parallelization_____________________________##
TPARALLEL="T"
##_______special rules for the configure script and f90 preprocessor_________##
SPECIAL="none"
##___________________DIRECTORIES containing LIBRARIES___________________________##
BLASDIR=""
LAPACKDIR="/home/tools/libs/acml3.0.0_64/gnu64/lib/"
FFTDIR="/home/tools/libs/fftw-2.1.5_no-second-underscore/"
MPIDIR="/home/tools/libs/mpich-1.2.6/"
##_____________________________ include file for fftw_____________________________##
FFT_HEADER="${FFTDIR}/fortran/fftw_f77.i"
##_____________________________________ include file for mpi "mpif.f90"___________##
MPI_HEADER="${MPIDIR}/include/mpif.h"
##_______F90 compiler and linker for scalar executables_____________________##
COMPILER_SCALAR="g95 -fno-second-underscore "
##_______F90 compiler and linker for parallel executables___________________##
COMPILER_PARALLEL="g95 -fno-second-underscore "
```

```
##__________________standard compiler flags___________________________________##
FCFLAGS_NONE="-c "
#_____________________compiler flags for optimization_____________________##
FCFLAGS_OPT="-c -O3 -fshort-circuit -funroll-loops -fomit-frame-pointer -msse2"
##_____________________compiler flags for  profiling_____________________##
FCFLAGS_PROF="-c -pg -O3 -fshort-circuit -funroll-loops  -msse2"
#_____________________compiler flags for debugging_____________________##
FCFLAGS_DBG="-c -g -std=f95 -Wall -ftrace=full  -fimplicit-none -fbounds-check"
#_____________________flags for linking_____________________##
LDFLAGS_SCALAR="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR}  \
                -L${FFTDIR}/fftw/.libs/"
#_____________________flags for linking_____________________##
LDFLAGS_PARALLEL="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
                 -L${FFTDIR}/fftw/.libs/ -L${MPIDIR}/mpe/lib"
#_____________________external libraries (sequential)_____________________##
LIBS_SCALAR="-Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lg2c"
#_____________________external libraries (parallel)_____________________##
LIBS_PARALLEL="-Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lg2c -Wl,-dy -lmpe \
              -Wl,-dy -llmpe"
#_____________________preprocessor variables_____________________##
CPPFLAGS="-DCPPVAR_COMPILER_G95 -DCPPVAR_FFT_FFTW \
          -DCPPVAR_LAPACK_LAPACK -DCPPVAR_BLAS_BLAS"
#_____________________file extension_____________________##
FEXT="f90"
###############################################################################
```

3.2 Erklärung der Einträge im Parameterfile

3.2.1 ARCH

Dient zur Namensgebung der Verzeichnisse, in dem die lauffähigen Programme erzeugt werden. In der PAW Directory wird ein Unterordner ***paw-directory***`/bin/`***arch*** erzeugt. In diesem Ordner befinden sich anschließend die lauffähige Programme, Objekt files, Module files etc.

3.2.2 TUPPERCASEMOD

Je nach Compiler werden die module files mit Groß- oder Kleinbuchstaben geschrieben. (`TUPPERCASEMOD='T'` für Großschreibung und `TUPPERCASEMOD='F'` für Kleinschreibung). Die Makefiles können mit dieser Information die die Abhängigkeiten berücksichtigen und vermeiden damit unnotwendiges Neucompilieren.

Ist der Wert zunächst nicht bekannt, kann man mit einem beliebigen Wert starten. Anschließend kann man die Konvention mit

`ls` ***paw-directory***`/bin/`***arch***`/Objects/fast/*.mod`

nachsehen und den Parameter entsprechend anpassen.

3.2.3 TPARALELL

Gibt an, dass mit MPI eine parallel arbeitende Version vom PAW erzeugt werden soll. Voraussetzung für ein paralleles PAW ist eine entsprechenden MPI Bibliothek. Um nur sequentielle Programme zu erzeugen, setzt man den Wert `TPARALELL='F'`.

3.2.4 SPECIAL

Setze `SPECIAL=' '`. Dies ist ein Flag für Ausnahmeregeln in den Makefiles. Nichts für "gewöhnliche" Nutzer.

3.2.5 BLASDIR,LAPACKDIR,FFTDIR,MPIDIR

Dieser Block gibt den Pfad der verschiedenen Bibliotheken an. Diese Variablen haben keine eigene Bedeutung, sondern sind Kürzel für den Rest des Parameterfiles. Configure erkennt keine Zwischenvariablen im Parameterfile. Die hier genannten Variablen können jedoch belegt und wieder verwendet werden. Bibliotheken, die nicht benötigt werden, müssen hier auch nicht belegt werden.

Beachte nochmals: PAW ist NICHT kompatibel zur fftw 3.* !!!

3.2.6 FFT_HEADER

Pfad zum Includefile `fftw_f77.i` der FFTW Bibliothek. Dieses File enthält gewisse Konstanten, welche den FFTW routinen als Argumente übergeben werden können um deren Verhalten zu beeinflussen.

Bei der Nutzung der fftw2 wie in Kapitel 2.3.3 beschrieben, muss hier nichts angepasst werden.

3.2.7 MPI_HEADER

Pfad zur Datei `mpif.h`. Dieses File muss mit der MPI Distribution mitgeliefert werden. Für MPICH muss hier nichts angepasst werden.

Was ist wenn das file nicht existiert, aber auch nicht benötigt wird?

Neuere Versionen (MPICH-2) bieten ein Modulfile an, was eher dem Fortran90 Standard entspricht. Darin werden auch explizite Interfaces für die Fortan aufrufe angegeben. Allerdings sind hierbei nicht alle Aufrufe unterstützt, die von PAW genutzt werden. Deshalb verwenden wir die herkömmliche Methode mit dem Includefile, um die Parameter einzubinden.

3.2.8 COMPILER

Der Aufruf für den Compiler für die sequentiellen und parallelen Binaries. Optionen die generell genutzt werden können in Ausnahmefällen bereits hier berücksichtigt werden.

Der g95 benö tigt die Option *-fno-second-underscore* um die Bibliotheken richtig linken zu können.

3.2.9 FCFLAGS_NONE,FCFLAGS_OPT, FCFLAGS_DBG,FCFLAGS_PROF

Hier werden die Compilervariablen gesetzt. Es werden vier unterschiedliche Parametersätze verwendet, um unterschiedliche Executables zu erzeugen.

- FCFLAGS_NONE: Der einfachste Parametersatz. Er wird nur verwendet, um die Abhängigkeit von den Parametern zu untersuchen. Die entsprechenden Executables sind paw.x und ppaw.x. Die Objectfiles liegen in dem Ordner "***paw-directory***`/bin/`***arch***`Objects/none`" bzw "***paw-directory***`/bin/`***arch***`Objects/none_parallel`".

- FCFLAGS_OPT: Der Parametersatz FCFLAGS_OPT entspricht der höchstmöglichen Optimierungsstufe. Das entsprechende Executable wird für den Normalgebrauch verwendet, bei dem es um Effizienz geht. Die entsprechenden Executables sind ***paw-directory***`/bin/`***arch***`/paw_fast.x` und ***paw-directory***`/bin/`***arch***`/ppaw_fast.x`. Die Objectfiles liegen in dem Ordner "***paw-directory***`/bin/`***arch***`Objects/fast`" bzw "***paw-directory***`/bin/`***arch***`Objects/fast_parallel`".

- FCFLAGS_DBG: Dies ist der Parametersatz zum Debuggen. Es sollen, neben dem Parameter -g, auch alle vernünftigen Tests, wie Array-bound checking, ausgeführt werden. Es soll keine Optimierung stattfinden. Die entsprechenden Executables sind paw_dbg.x und ppaw_dbg.x. Die Objectfiles liegen in dem Ordner Objects/dbg bzw Objects/dbg_parallel.

- FCFLAGS_PROF: Dieser Parametersatz wird zum Profiling genutzt. Die Parameter entsprechen dem Satz für die optimierte Version und "-pg". Die entsprechenden Executables sind paw_prof.x und ppaw_prof.x. Die Objectfiles liegen in dem Ordner Objects/prof bzw Objects/prof_parallel.

Die Parametersätze sind vom Compiler abhängig und manchmal (besonders bei der Optimierung) auch von der Rechnerarchitektur und CPU.

3.2.10 LDFLAGS_SCALAR,LDFLAGS_PARALLEL

`LDFLAGS_PARALLEL` wird nur für benoötigt wenn `TPARALLEL=''T''`.

Parameter für den Linker. Hier werden besonders die Suchpfade für die Bibliotheken angegeben. Die Parametersätze werden für die sequentiellen und parallelen Binaries individuell angegeben.

3.2.11 LIBS_SCALAR,LIBS_PARALLEL

`LIBS_PARALLEL` wird nur für benoötigt wenn `TPARALLEL=''T''`.

Hier werden die Bibliotheken angegeben die beim Linken eingebunden werden sollen.

Mit -Wl,-dn und -Wl-dy wird festgelegt ob die Bibliotheken statisch oder dynamisch gelinkt werden sollen. Eine statische gelinkte Bibliothek -lfoo zeigt auf ein File libfoo.a. Eine dynamisch gelinkte Bibliothek zeigt auf libfoo.so. Wenn möglich, sollte man dynamisch linken. Will man ein Binary auf einem anderen Rechner nutzen, ist es notwendig, statisch zu linken. Manche Bibliotheken können nicht dynamisch gelinkt werden.

Die Bibliothek g2c wird für den G95 benötigt, welcher bei SUSE im Paket *compat-g77* enthalten ist.

Die MKL Bibliothek benötigt auch die Bibliotheken libguide and pthread. pthread is a native linux library used by libguide. libguide provides multithreading support within MKL. It is important that pthread is specified as last item in the link line.

```
-lmkl_lapack -lmkl_em64t -lguide -lpthread
```

When linking the atlas blas library, one does need to specify `-lf77blas -latlas`. ATLAS is a C-library and f77blas is a Fortran interface to the C-subroutines.

3.2.12 CPPFLAGS

Flags für den C-Preprozessor. Der C-Preprozessor wählt entsprechend dieser Flags bestimmte Programmsegmente aus. Dies geschieht besonders bei den Interfaces zu externen Bibliotheken. Letztere sind in dem File ***PAW-directory***`/src/paw_library.f90` enthalten.

- Die Variable `-DCPPVAR_COMPILER_`*foo* wählt das Interface zur entsprechenden Utility Bibliothek aus. Die Utility Bibliothek enthält Fortran Interfaces zu System Routinen, welche in C-geschrieben sind und im Allgemeinen Teil des Systems sind. Mögliche Werte sind
  - `CPPVAR_COMPILER_G95`
  - `CPPVAR_COMPILER_IFC`
  - `CPPVAR_COMPILER_IFC7`
  - `CPPVAR_COMPILER_ABSOFT`
  - `CPPVAR_COMPILER_XLF`
  - `CPPVAR_COMPILER_PGI`
  - `CPPVAR_COMPILER_PATHSCALE`
- Die Variable `CPPVAR_FFT_foo` wählt die Interfaces zur Bibliothek für Fouriertransformationen aus. Mögliche Werte sind:
  - `CPPVAR_FFT_FFTW`
  - `CPPVAR_FFT_ESSL`
  - `CPPVAR_FFT_ACML`
- Die Variable `CPPVAR_LAPACK_foo` wählt die Interfaces zur LAPACK routinen, bzw. äquivalenter Bibliotheksroutinen aus. Wird kein entsprechender Parameter angegeben, dann werden die Aufrufe zur LAPACK Bibliothek verwendet. Da die meisten numerischen Bibliotheken LAPACK identisch nachbilden, muss dieser Parameter nur selten angegeben werden. Es ist dennoch ratsam einen Parameter einzufügen, um das Verhalten explizit zu machen.

  Mögliche Werte für `CPPVAR_LAPACK_foo` sind:
  - `CPPVAR_LAPACK_ESSL`
  - `CPPVAR_LAPACK_LAPACK`: Default Verhalten. Es werden die LAPACK aufrufe verwendet.
- Die Variable `CPPVAR_BLAS_foo` wählt die Interfaces zur BLAS routinen, bzw. äquivalenter Bibliotheksroutinen aus. Wird kein entsprechender Parameter angegeben, dann werden die Aufrufe zur BLAS Bibliothek verwendet. Da die meisten numerischen Bibliotheken BLAS identisch nachbilden, muss dieser Parameter nur selten angegeben werden. Es ist dennoch ratsam einen Parameter einzufügen, um das Verhalten explizit zu machen.

  Mögliche Werte für `CPPVAR_BLAS_foo` sind:
  - `CPPVAR_BLAS_ESSL`
  - `CPPVAR_BLAS_BLAS`: Default Verhalten. Es werden die BLAS aufrufe verwendet.

3.2.13 FEXT

Endung für die Quellcode files die dem Compiler präsentiert werden. Fortran Compiler verlangen dass die files für den Quellcode eine bestimmte Endung haben, wie z.B. foo.f90. Für Fortran90 ist das meistens f90. Für Fortran77 war es .f und für Fortran95 ist es manchmal f95.

Kapitel 4

Liste der Targets des Make files

Das Makefile erkennt die folgenden Targets:

- none Erzeugt ein sequentielles Binary ohne Optimierungen. Hiermit sollte beim erstmalligen installieren angefangen werden.
- dbg
- fast Mit dem Kommando *make fast* werden im Verzeichnis *bin/$arch* die Objekte und die Programmdateien für ein sequentielles Binary erzeugt.
- prof
- none_parallel
- dbg_parallel
- fast_parallel
- prof_parallel
- clean
- clean_none
- clean_dbg
- clean_fast
- clean_prof
- tools
- docs

Kapitel 5

Probleme und Hinweise

5.1 Stack-size exceeded

Unpredictable behavior can occur if the stack-size is exceeded. In the bash shell the stack size can be increased using the command `ulimit -s unlimited`. This is apparently a problem of the operating system. It can be caused by certain optimizations such as inlining. The latter increases the code size.

5.2 No core dump

If the code crashes without creating a core dump, the limit for the core file size must be increased using `ulimit -c unlimited`

5.3 Second underscore

Der Compiler muss Fortran Namen wie die von Subroutinen oder Variablen in Symbole umwandeln. Dabei wird an jeden Namen üblicherweise ein underscore angehängt. Manche Compiler weichen von dieser Konvention ab, wenn der Name bereits einen underscore enthält, und hängen dann zwei underscores an. Letzeres ist die "second-underscore" Konvention.

"G95 follows the f2c convention of adding an underscore to public names, or two underscores if the name contains an underscore." Diese Konvention kann üblicherweise durch Compilerflags geändert werden.

In der folgenden Tabelle wird die "second-underscore" Konvention als su und die "no-second-underscore" Konvention mit nsu bezeichnet.

Fortran Name	Symbol su	Symbol nsu
`abc`	`abc_`	`abc_`
`a_b_c`	`a_b_c__`	`a_b_c_`
`abc_`	`abc___`	`abc__`
`a_b_c_`	`a_b_c___`	`a_b_c__`

Offensichtlich entstehen Probleme wenn man eine Bibliothek einbinden möchte, welche mit einer anderen underscoring Konvention erzeugt wurde. Viele Bibliotheken sind eigentlich in C geschrieben und besitzen zusätzlich einen Fortran wrapper. Damit ein Bibliotheksaufruf erkannt wird muss die Bibliothek mit derselben underscoring Konvention übersetzt worden sein.

5.4 Symbollisten von Object files und Libraries

Um zu sehen, welche Routinen aufgerufen, bzw. in Bibliotheken vorhanden sind, kann man die Symbollisten vergleichen.

Mit dem Kommando `nm foo.o` kann man die Symbolliste eines Objectfiles foo.o einsehen und damit erkennen, welche Underscoringkonvention jeweils verwendet wurde. Aehnlich kann man die Symbolliste einer Bibliothek, z.B. `nm foo.a`, einsehen.

5.5 Runtime Error in viacheck.c, `code=VAPI_RETRY_EXC_ERR`

The routine viacheck.c is part of MVAPICH, the MPI implementation for Infiniband networks. The error means that the Infiniband Reliable Connection retry Coundf was exceeded. This may occur if there is a bad cable or port on the hardware. It may also occur if the code undergoes a segmentation fault, so that the job is not stopped on all nodes. On those nodes the job fails, because it does not receive the required response.

5.6 Fehlende Bibliothek pthread

Bei `pthread` handelt es sich um eine Systembibliothek welche von der MKL benötigt wird. Sie kann bei manchen Systemen zu Problemen während des Compilierens führen. Speziell bei Suse-Systemen ist dies auffällig gewesen. Um dieses Problem zu umgehen, benötigt man entweder eine pthread Bibliothek von einem anderem System oder man linkt sie beim Compilieren dynamisch.

The static SUSE pthread library (`/usr/lib/libpthread.a`) is buggy. The problem is usually solved by linking the pthread library dynamically, that is with "`-Wl,-dy -lpthread`". Another workaround has been to use the pthread library from debian or redhat. This library is copied to , for example, `/usr/lib/libpthread-debian.a`, and then linked using `-lpthread-debian` instead of `-lpthread`.

5.7 Fehlende Bibliothek g2c

Falls der Linker "undefined references" in "`xerbla.o`" anzeigt, ist dies ein Zeichen dafür, dass die g2c Bibliothek fehlt oder nicht korrekt eingebunden ist.

Bei *g2c* handelt es sich um den Gnu Fortran-to-C converter, der die Grundlage des G77 compilers bildet. Er hiess früher f2c (libf2c). Die Bibliothek g2c ist meistens im G77 oder gfortran Paket enthalten.

Fogende Bemerkung vom Internet ohne Kommentar...

"Library g2c is the Fortran 77 shared library needed to run Fortran 77 dynamically linked programs. The library is no longer needed with the new family of Fortran (native) compilers like g95. To correct the error, we searched first of all for the shared version of g2c in /usr/lib*:"

5.8 MPI: rsh versus ssh

The parallelization requires a method to communicate between different machines. Two possibilities exist, namely via rsh and ssh. rsh is a bit faster and ssh is a lot safer. Therefore the use of ssh is strongly recommended.

5.9 Dynamic versus static linking

Libraries can be linked dynamically or statically. A statically linked library is completely integrated with the binary. If the binary is to be used on another computer, it is important to link the libraries statically.

The code size is smaller when the libraries are linked dynamically, that is during runtime. In that case only an interface to a shared library is integrated into the binary. If one copiesthe executable to another machine, it is important that the shared libraries are available and identical to those on the original machine.

The preferred mode is dynamic linking.

It is possible to convert a static library into a dynamic library. Use teh command

```
ls -z allextract *.a *.so
```

5.10 Multiple definition of ...

- (To `paw_...` This problem should be absent in the current implementation. The possible cause is that paw routines are linked twice, once directly and once as part of the paw librart libpaw.a. This may happen when linking the PAW tools.
- Do not link both, blas and mkl (or blas and acml). Both contain the BLAS routines

5.11 Cannot find lf77blas

Set the correct path to the library libf77blas.a. It is part of the ATLAS package.

5.12 PMPI_Allreduce

This problem is related to the PATHSCALE compiler:

If the linker complains that it does not find routines starting with PMPI it helps if one also links the library `libpmpich`. The library commands are the "`-lfmpich -lmpich -lpmpich`". (Solution obviously for MPICH only.)

5.13 Linker flag -I does not work

Some time ago the ABSOFT compiler did not accept the -l flag to set the search path include files. The current installation procedure takes care of this.

5.14 cannot find -lvapi

vapi is a library used by the infiniband drivers. (Infiniband is a network protocoll that can be used by MPI).

The solution is to link vapi dynamically. `-Wl,-dy -lvapi`.

5.15 Cannot find library gcc_s

This problem is related to the PATHSCALE compiler:

Simply add "`-lgcc_s`" to the list of libraries.It may be necessary to supply also the path. On my system itr is in "`-L/usr/lib64/gcc/x86_64-suse-linux/4.1.2/`".

5.16 Informationen über das System

Um zum die richtigen Compiler und Bibliotheken herunterzuladen und für bestimmte Lizenzen, sind einige Informationen uber das aktuelle System notwendig.

5.16.1 Rechnername

Das Kommando "`hostname`" lieferet den Rechnernamen.

Die IP Adresse erhält man durch "`hostname -i`" Den Domainennamen erhält man durch "`hostname -d`"

5.16.2 MAC Adresse

Die Media Access Control (MAC) Adresse oder Ethernet ID wird manchmal benötigt um eine Lizenz speziell für einen Rechner ausstellen zu können. Sie ist eine eindeutige Nummer für jede Netzwerkkarte. Die MAC Adresse besteht aus 6 Byte und wird häufig hexadezimal geschrieben, z.B. `08:00:20:AE:FD:7E`.

Um die MAC Adresse zu erhalten, führt man das Kommando

```
/sbin/ifconfig -a
```

aus. Man erhält information zu allen Netzwerkkarten. Die Ethernet Hardware Adresse ist die MAC Adresse.

5.16.3 Rechnerarchitektur

Das Linux Kommando "`arch`" oder "`uname -m`" lieferet die Rechnerarchitektur. z.B. x86_64.
Die wichtigsten Rechnerarchitekturen sind:

- IA-32 (Intel Architecture 32bit), i386, x86 ist die seit 87 von Intel verwendete Prozessorarchitektur bis zur Einführung der 64bit Rechner.
- x86-64, AMD64, EM64T, IA-32e, x64 Architektur der ersten 64-bit Prozessoren von AMD. EM64T steht für "Extended Memory 64 Technology".
- IA-64 ist die volkommen neue 64 Architektur von Intel. Sie kommt bei den Itanium Prozessoren zum Einsatz. IA-64 steht für "Intel Architecture 64".
- POWER ist die Architektur der RISC Prozessoren von IBM. Power steht hier für Performance optimized with enhanced RISC)

5.16.4 Linux version

Unter Suse erhält man die Versionsnummer durch `cat /etc/SuSE-release`. Die Kernel version erhält man durch `uname -r`

5.17 Hintergrund zum Configure Skript

Es mag von Interesse sein sein, zu verstehen, woher das configure script kommt, welches die Konfiguration vornimmt.

5.17.1 What does the configure script do

The configuration helps the user to compile the PAW-Code without having to find out about a lot of parameters himself. The configure script explores the availability of the compilers and libraries, and it sets the corresponding compiler flags and preprocessor variables. In the current version, however, we have to specify most parameters explicitely in a parameter file.

The configure script uses the parameter file ***parmfile*** in order to construct the necessary Makefiles from corresponding templates. The templates in use are

- `Makefile_targets.in` is converted in the `Makefile` located in the ***PAW-directory***. This is the primary makefile executed by the user.
- `Makefile_bare.in` is converted into `Makefile_bare` located in the ***PAW-directory***. However this make file is never executed itself, but it is itself only a template for the makefiles located in the directories

  paw-directory`/bin/`***arch***`/Objects/`***type***

  where ***type*** is one of `none`,`dbg`, `fast`, `prof`, `none_parallel`, `dbg_parallel`, `fast_parallel`, `prof_parallel`.

The configure script makes a copy of the templates for the Makefiles and replaces strings of the type `@VARIABLE@` in the copy of the template by the corresponding value. The variables are identifies by the two `@` at the beginning and the end.

In addition, the configure script constructs a directory Tree that will hold the specific makefiles, objects, binaries etc.

5.17.2 How the configure script is constructed

The configure script is constructed with the help of the GNU `autoconf` tool . The `autoconf` tool uses an input file `configure.ac`, which describes the configuration process in the `autoconf` macro language.

Especially important is the first part with the *user adaptable variables*. Here all the values can be set - the rest of the script just uses the variables. This is the place to make permanent changes to the installation scheme (e.g. change the default compiler flags).

By invoking

```
autoconf
```

in the PAW directory the `configure` file – which is a `/bin/sh` script – will be generated.

Anhang A

Resultat des Konfigure scripts

Ist configure erfolgreich, findet man ein Resultat wie das Folgende Beispiel:

```
checking for parms.g95_guam... yes
check for make
checking for gmake... gmake
checking for cpp... cpp
checking for g95... yes
checking for xlf90... no
checking for ifort... no
checking for ifc... no
checking for f90... yes
checking for fort... no
resolve parms.g95_guam
${MPI_HEADER}

============================
check MPI directory
checking for /home/tools/libs/mpich-1.2.6/... yes
check FFT directory
checking for /home/tools/libs/fftw-2.1.5_no-second-underscore/... yes
copy parms.g95_guam to parms.in_use
creating subdirectories and copying shell scripts
configure: creating ./config.status
config.status: creating /home/ptpb/Tree/PAW/main/bin/g95_guam/f90pp
config.status: creating Makefile
config.status: creating Makefile_bare
modify makefile_bare for lowercase module files
creating Makefile in none
creating Makefile in none_parallel
creating Makefile in fast
creating Makefile_parallel in fast
creating Makefile in dbg
creating Makefile_parallel in dbg
```

```
creating Makefile in prof
creating Makefile_parallel in prof
------------------------------------------------------------
----------------------SUMMARY-------------------------------
------------------------------------------------------------
directory of distribution      : /home/ptpb/Tree/PAW/main
directory with binaries        : /home/ptpb/Tree/PAW/main/bin/g95_guam
architecture                   : g95_guam
preprocessor variables         : -DCPPVAR_COMPILER_G95 -DCPPVAR_FFT_FFTW  -DCPPVAR_LAPACK
architecture name              : g95_guam
parallel environment           : T
compile command (scalar)       : g95 -fno-second-underscore
F90 file extension             : f90
compile flags (none)           : -c
compile flags (fast)           : -c -O3 -fshort-circuit -funroll-loops -fomit-frame-point
compile flags (dbg)            : -c -g -std=f95 -Wall -ftrace=full  -fimplicit-none -fbou
compile flags (prof)           : -c -pg -O3 -fshort-circuit -funroll-loops  -msse2
link command w.flags (scalar)  : g95 -fno-second-underscore  -Wl,-dy -I${OBJDIR} -L${OBJD
external libraries (scalar)    : -Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lg2c
uppercase module names?        : F
blas library                   :
lapack library                 : /home/tools/libs/acml3.0.0_64/gnu64/lib/
libs for Fourier transforms    : /home/tools/libs/fftw-2.1.5_no-second-underscore/
parallel envirnment considered : yes
compile command (parallel)     : g95 -fno-second-underscore
link command w. flags(parallel): g95 -fno-second-underscore  -Wl,-dy -I${OBJDIR} -L${OBJD
external libraries (parallel)  : -Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lg2c -Wl,-dy -lfm
mpi library                    : /home/tools/libs/mpich-1.2.6/
MAKE command                   : gmake
CPP command                    : cpp  -traditional
------------------------------------------------------------
----done!---configuration completed successfully!--------------
------------------------------------------------------------
```

Anhang B

Beispiele für Parameterfiles

B.0.3 Beispiel eines Parameterfiles für eine einfache Installation

Dies ist ein einfaches Beispiel für ein Parameterfile.

```
ARCH="G95_mycomputer"
TPARALLEL="T"
TUPPERCASEMOD="F"
TPARALLEL="T"
SPECIAL="none"
BLASDIR=""
LAPACKDIR="/opt/acml4.0.1/libs/acml3.0.0_64/gnu64/lib/"
FFTDIR=""
MPIDIR="/opt/mpich-1.2.6/"
FFT_HEADER=" "
MPI_HEADER="${MPIDIR}/include/mpif.h"
COMPILER_SCALAR="g95 -fno-second-underscore "
COMPILER_PARALLEL="g95 -fno-second-underscore "
FCFLAGS_NONE="-c "
FCFLAGS_OPT="-c -O3 -fshort-circuit -funroll-loops -fomit-frame-pointer -msse2"
FCFLAGS_PROF="-c -pg -O3 -fshort-circuit -funroll-loops  -msse2"
FCFLAGS_DBG="-c -g -std=f95 -Wall -ftrace=full  -fimplicit-none -fbounds-check"
LDFLAGS_SCALAR="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} "
LDFLAGS_PARALLEL="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR}  -L${MPIDIR}/lib"
LIBS_SCALAR="-Wl,-dn -lacml -Wl,-dy -lg2c"
LIBS_PARALLEL="-Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lg2c \
            -Wl,-dy -lfmpich -Wl,-dy -lmpich"
CPPFLAGS="-DCPPVAR_COMPILER_G95 -DCPPVAR_FFT_ACML  \
    -DCPPVAR_LAPACK_LAPACK "-DCPPVAR_BLAS_BLAS "
FEXT="f90"
```

B.1 Parameterfiles für G95

```
ARCH="g95_guam"
```

```
TUPPERCASEMOD="F"
TPARALLEL="T"
SPECIAL="none"
BLASDIR=""
LAPACKDIR="/home/tools/libs/acml3.0.0_64/gnu64/lib/"
FFTDIR="/home/tools/libs/fftw-2.1.5_no-second-underscore/"
MPIDIR="/home/tools/libs/mpich-1.2.6/"
FFT_HEADER="${FFTDIR}/fortran/fftw_f77.i"
MPI_HEADER="${MPIDIR}/include/mpif.h"
COMPILER_SCALAR="g95 -fno-second-underscore "
COMPILER_PARALLEL="g95 -fno-second-underscore "
FCFLAGS_NONE="-c "
FCFLAGS_OPT="-c -O3 -fshort-circuit -funroll-loops -fomit-frame-pointer -msse2"
FCFLAGS_PROF="-c -pg -O3 -fshort-circuit -funroll-loops  -msse2"
FCFLAGS_DBG="-c -g -std=f95 -Wall -ftrace=full  -fimplicit-none -fbounds-check"
LDFLAGS_SCALAR="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR}\
      -L${FFTDIR}/fftw/.libs/"
LDFLAGS_PARALLEL="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
        -L${FFTDIR}/fftw/.libs/ -L${MPIDIR}/lib"
LIBS_SCALAR="-Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lg2c"
LIBS_PARALLEL="-Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lg2c \
         -Wl,-dy -lfmpich -Wl,-dy -lmpich"
CPPFLAGS="-DCPPVAR_COMPILER_G95 -DCPPVAR_FFT_FFTW  \
         -DCPPVAR_LAPACK_LAPACK "-DCPPVAR_BLAS_BLAS "
FEXT="f90"
```

B.2 Parameterfiles für IFORT

```
ARCH="ifc10_guam"
TUPPERCASEMOD="F"
TPARALLEL="T"
SPECIAL="none"
BLASDIR=""
LAPACKDIR="/opt/intel/mkl/9.1/lib/em64t/"
FFTDIR="/home/tools/libs/fftw-2.1.5_no-second-underscore/"
MPIDIR="/home/tools/libs/mpich-1.2.6_ifc10/"
FFT_HEADER="${FFTDIR}/fortran/fftw_f77.i"
MPI_HEADER="${MPIDIR}/include/mpif.h"
COMPILER_SCALAR="ifc10"
COMPILER_PARALLEL="${MPIDIR}/bin/mpif90 -choicemod "
FCFLAGS_NONE="-c "
FCFLAGS_OPT="-c -O2 -fast -finline-functions -finline-limit=50"
FCFLAGS_PROF="-c -pg -O3 "
FCFLAGS_DBG="-c -g -check bounds -check format -check pointers \
            -check uninit -debug full -debug-parameters all \
            -fp-stack-check -ftrapuv -stand f95 -traceback \
             -warn declarations"
LDFLAGS_SCALAR="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
                -L${FFTDIR}/fftw/.libs/"
LDFLAGS_PARALLEL="-Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
                  -L${FFTDIR}/fftw/.libs/"
LIBS_SCALAR="-Wl,-dn -lfftw -Wl,-dn -lmkl_lapack -Wl,-dn -lmkl_em64t \
             -Wl,-dn -lguide  -Wl,-dy -lpthread  -Wl,-dy -lg2c"
LIBS_PARALLEL="-Wl,-dn -lfftw -Wl,-dn -lmkl_lapack -Wl,-dn -lmkl_em64t \
               -Wl,-dn -lguide  -Wl,-dy -lpthread -Wl,-dy -lg2c "
CPPFLAGS="-DCPPVAR_COMPILER_IFC -DCPPVAR_FFT_FFTW \
           -DCPPVAR_LAPACK_LAPACK -DCPPVAR_BLASK_BLAS "
FEXT="f90"
```

B.3 Parameterfile für PGI

```
ARCH="pgi_guam"
TUPPERCASEMOD="F"
TPARALLEL="T"
SPECIAL="none"
BLASDIR=""
LAPACKDIR="/opt/pgi/linux86-64/7.1/lib"
FFTDIR="/home/tools/libs/fftw-2.1.5_no-second-underscore/"
MPIDIR="/opt/pgi/linux86-64/7.1/mpi/mpich/"
FFT_HEADER="${FFTDIR}/fortran/fftw_f77.i"
MPI_HEADER="${MPIDIR}/include/mpif.h"
COMPILER_SCALAR="pgf90 -fpic"
COMPILER_PARALLEL="pgf90 -fpic"
FCFLAGS_NONE=" -c "
FCFLAGS_OPT="-c -fast -fastsse -Mipa=fast,inline"
FCFLAGS_PROF="-c -pg -fast -fastsse -Mipa=fast,inline"
FCFLAGS_DBG="-c -g -Mlist -m -C -Mbounds "
LDFLAGS_SCALAR=" -g77libs -Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
                -L${FFTDIR}/fftw/.libs/"
LDFLAGS_PARALLEL="-g77libs -Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
                 -L${FFTDIR}/fftw/.libs/ -L${MPIDIR}/lib"
LIBS_SCALAR="-Wl,-dn -lfftw -Wl,-dy -lacml  "
LIBS_PARALLEL="-Wl,-dn -lfftw -Wl,-dy -lacml  -Wl,-dy -lfmpich -Wl,-dy -lmpich"
CPPFLAGS="-DCPPVAR_COMPILER_PGI -DCPPVAR_FFT_FFTW \
          -DCPPVAR_LAPACK_LAPACK -DCPPVAR_BLAS_BLAS"
FEXT="f90"
```

B.4 Parameterfile für Pathscale

```
ARCH="pathscale_guam"
TUPPERCASEMOD="T"
TPARALLEL="T"
SPECIAL="none"
BLASDIR=""
LAPACKDIR="/opt/acml4.0.1/pathscale64/lib/"
FFTDIR="/home/tools/libs/fftw-2.1.5_no-second-underscore/"
MPIDIR="/opt/mpich-1.2.7p1_pathscale_ssh"
FFT_HEADER="${FFTDIR}/fortran/fftw_f77.i"
MPI_HEADER="${MPIDIR}/include/mpif.h"
COMPILER_SCALAR="pathf95 -fno-second-underscore  "
COMPILER_PARALLEL="pathf95 -fno-second-underscore  "
FCFLAGS_NONE="-c "
FCFLAGS_OPT="-c -O3   -OPT:Ofast -fno-math-errno -ffast-math "
FCFLAGS_PROF="-c -pg -O3 -profile "
FCFLAGS_DBG="-c -C -g -Wall "
LDFLAGS_SCALAR="-ipa -Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
              -L${FFTDIR}/fftw/.libs/ -L/usr/lib64/gcc/x86_64-suse-linux/4.1.2/ "
LDFLAGS_PARALLEL="-ipa -Wl,-dy -I${OBJDIR} -L${OBJDIR} -L${LAPACKDIR} \
           -L${FFTDIR}/fftw/.libs/  -L/usr/lib64/gcc/x86_64-suse-linux/4.1.2/ \
            -L${MPIDIR}/lib "
LIBS_SCALAR="-Wl,-dn -lfftw -Wl,-dn -lacml -Wl,-dy -lgcc_s "
LIBS_PARALLEL="-Wl,-dn -lfftw -Wl,-dn -lacml  -Wl,-dy -lfmpich \
               -Wl,-dy -lmpich -lpmpich -Wl,-dy -lgcc_s"
CPPFLAGS="-DCPPVAR_COMPILER_PATHSCALE -DCPPVAR_FFT_ACML  \
          -DCPPVAR_LAPACK_LAPACK -DCPPVAR_BLAS_BLAS "
FEXT="f90"
```

Anhang C

Input Datenfiles

C.1 Control Input File

```
!CONTROL
 !GENERIC  TRACE=f DT=10.0 NSTEP=180 NWRITE=100 START=t !END
 !DFT    TYPE=10 !END
 !FOURIER EPWPSI=20 CDUAL=2 !END
 !PSIDYN  FRIC=0.005
    !AUTO FRIC(-)=0.3 FACT(-)=0.97
          FRIC(+)=0.3 FACT(+)=1. !END
 !END
!END
!EOB
```

C.2 Structure Input File

```
!STRUCTURE
  !GENERIC LUNIT=10.26 !END
  !KPOINTS DIV=1 1 1 !END
  !OCCUPATIONS EMPTY=5 NSPIN=1  !END
  !LATTICE T= 0.00000  0.50000  0.50000
             0.50000  0.00000  0.50000
             0.50000  0.50000  0.00000 !END
  !SPECIES NAME= 'Si' ZV=4. M=5.  NPRO= 2 2 1 lrhox=2
           FILE='si_.75_6.0.out'
  !END
  !ATOM   NAME= 'Si_1'  R= 0.00  0.00 0.00 !END
  !ATOM   NAME= 'Si_2'  R= 0.25  0.25 0.25 !END
!END
!EOB
```

Der String `'si_.75_6.0.out'` muss angepasst werden. Er verweist auf ein Setupfile.