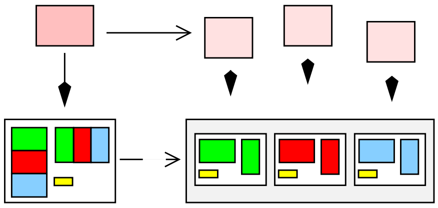
ADAPTOR
Installation Guide
Version 6.0 (June 1998)

T. Brandes
R. Höver-Klier



Report ADAPTOR 1

June 24, 1998

GMD – German National Research Center for Information Technology
Institute for Algorithms and Scientific Computing, SCAI
Schloss Birlinghoven
D-53754 Sankt Augustin
Germany

Tel.: +49 (0)2241 / 14-2492
E-mail: brandes@gmd.de, hoever-klier@gmd.de

This page is left empty

ADAPTOR
Installation Guide
Version 6.0 (June 1998)

T. Brandes
R. Höver-Klier

GMD – German National Research Center for Information Technology
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany

Abstract

ADAPTOR (Automatic DAta Parallelism TranslatOR) is a public domain High Performance Fortran (HPF) compilation system developed at the SCAI institute (GMD) during the last years. The tool transforms data parallel programs written in High Performance Fortran into message passing programs. The latest version incorporates the feedback of many users and contains already a lot of optimization techniques. Beside some restrictions, it supports now the new base language HPF 2.0 as well as most of the approved extensions within this new standard. Version 6.0 now also supports the compilation of HPF programs for vector architectures like the NEC SX-4.

This installation guide describes how this tool can be installed for different parallel architectures and how the installation is tested and tuned for the given target machine.

1 Overview

The *ADAPTOR* package consists of:

- the source to source transformation tool $\boxed{\text{fadapt}}$,
- the distributed array library $\boxed{\text{DALIB}}$. This is the HPF runtime system that handles descriptors for arrays, sections and distributions and implements communication routines,
- the compiler driver $\boxed{\text{gmdhpf}}$,
- documentation files in PostScript format,
- and many example programs.

Figure 1 shows how the different components work together. The compiler driver `gmdhpf` invokes the source-to-source translation `fadapt` that generates an SPMD program with message passing from the HPF program. Afterwards, it invokes a native FORTRAN 77 or Fortran 90 compiler to compile the generated code. Finally, the compiled codes are linked with the DALIB runtime system and the utilized message passing library.

Note: There is a modified ADAPTOR version 6.0 available that supports also the automatic vectorization capabilites of the NEC SX compilers. This version is not available in the public domain.

2 System Requirements

The ADAPTOR source to source translation $\boxed{\text{fadapt}}$, the $\boxed{\text{DALIB}}$ runtime system, and the compile driver $\boxed{\text{gmdhpf}}$ are available as C programs and require only a C compiler.

For the generated code to be compiled, linked and run, the following components are required:

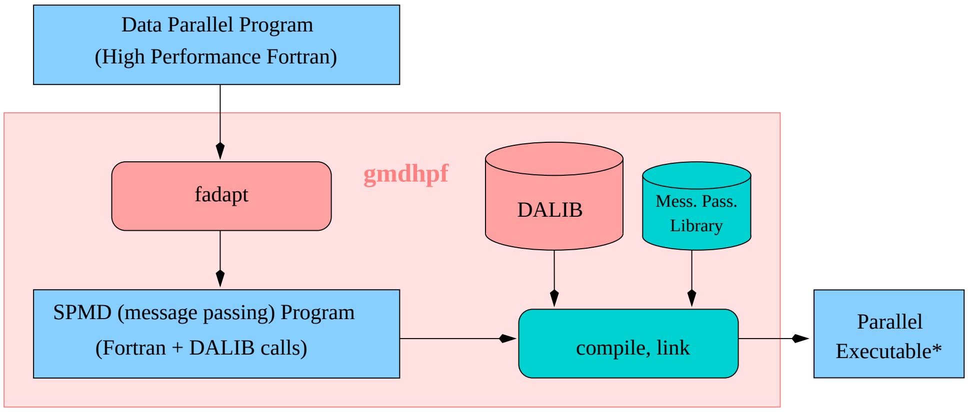


Figure 1: Overview of the ADAPTOR tool.

- A MPI or PVM message-passing subsystem, for some systems (IBM SP, Intel Paragon, CM 5 and Meiko CS-2) the native message passing subsystem can be used directly.
- A FORTRAN 77 compiler is sufficient for compiling the data parallel program based on usual HPF array syntax. If the HPF program contains modules, derived data types, pointers or some other special features of Fortran 90 are used, a Fortran 90 compiler is required.
- A parallel programming environment in which parallel programs can be loaded and executed on a parallel machine.

3 Installation Procedure

3.1 Installation Directory

Before downloading ADAPTOR you should create a directory for the installation. This installation directory will be referred to as PHOME.

```
...
mkdir adaptor
cd adaptor
pwd         < this is PHOME >
$PHOME=‘pwd‘
```

3.2 Downloading of ADAPTOR

ADAPTOR is available via anonymous ftp and should be downloaded directly in the installation directory.

```
cd $PHOME
ftp ftp.gmd.de   (129.26.8.90)
 user: anonymous
 password: <your e-mail address>
cd gmd/adaptor
binary
get adp_6.0.tar.Z                  or get adp_6.0.tar.gz
quit
```

At first the tar file has to be uncompressed before all the files of ADAPTOR can be extracted.

```
uncompress adp_6.0.tar.Z            or gunzip adp_6.0.tar.gz
tar xvf adp_6.0.tar
rm adp_6.0.tar
```

After extraction, these files and subdirectories should be present in the ADAPTOR-directory:

```
 bin             (directory for binaries)
 src             (source files for fadapt)
 src1            (source files for gmdhpf)
 dalib           (runtime system used for ADAPTOR)
 man             (manual pages for ADAPTOR)
 docs            (documentation for ADAPTOR)
 hpf_examples    (HPF example programs)
```

The installation of ADAPTOR should be proceeded in the following intended way:

1. Compile and link the transformation tool $\boxed{\text{fadapt}}$ (see section 4).
2. Compile and install the $\boxed{\text{DALIB}}$ for your parallel machine (see section 5).
3. Compile and link the compiler driver $\boxed{\text{gmdhpf}}$ (see section 6).
4. Set and verify the correct configuration (see section 6.2).
5. Test the installation (see section 7).

3.3 Example Programs

A lot of example programs are provided. All the codes are HPF programs that have been translated successfully with the ADAPTOR system and some other HPF compilers.

```
cd $PHOME
ftp ftp.gmd.de   (129.26.8.90)
   user: anonymous
   password: <your e-mail address>
  cd gmd/adaptor
  binary
  get hpf_examples.tar.Z
  quit
```

The tar file has to be uncompressed in the same way as the ADAPTOR package.

Many example programs sorted in different directories are given to demonstrate the translation with ADAPTOR.

3.4 Documentation

Some documentation for the ADAPTOR tool is provided.

```
cd $PHOME
ftp ftp.gmd.de   (129.26.8.90)
 user: anonymous
 password: <your e-mail address>
cd gmd/adaptor
binary
get docs.tar.Z
quit
```

The tar file has to be uncompressed in the same way as the ADAPTOR package.

The following documents should be available:

- Installation Guide (**iguide.ps**) is this document,
- the User's Guide [BHK98] (**uguide.ps**) explains how to use ADAPTOR,
- the Programmer's Guide [Bra98b] (**pguide.ps**) describes the supported features of ADAPTOR and how to use them efficiently.

Some further documentation (e.g. for experiences and results with ADAPTOR) is available via anonymous ftp (`ftp.gmd.de:gmd/adaptor/docs`).

4 Installation of fadapt

The sources for the transformation tool fadapt are stored in the subdirectory **src**. The sources are all written in C and should run on all UNIX machines.

Depending on the machine where the transformation tool is installed it might be necessary to make some changes in the Makefile (see section 4.2).

The executable **fadapt** for the tool fadapt will be generated by simply typing make.

```
cd $PHOME/src
make
mv fadapt ../bin
make clean           ! after successful installation
```

4.1 Information about the Source Files

Most files of the source to source transformation have been generated with compiler tools (more information at `http://www.gmd.de/SCAI/lab/adaptor/cocktail.html`). The tar file provides only the generated sources and not the compiler specifications. Otherwise, it would be necessary to install the compiler tools in the correct version (the GMD compiler tools are also available in the same anonymous ftp area in the directory **gmd/cocktail**).

We do not provide the compiler specifications of ADAPTOR by public domain. Beside some copyrights, we do not want to have different versions of the ADAPTOR tool. Nevertheless, we can provide the ADAPTOR frontend and backend specifications for people who want to build other tools based on this compiler technology.

4.2 Changes in the Makefile

The `Makefile` delivered for the installation of `fadapt` should work directly on machines where the GNU C compiler is available. For other C compilers, it might be necessary to make some changes for the variables.

Check the following variables within the `Makefile`:

- `CC` for your C-Compiler
- `CFLAGS` for compiler options
- `LFLAGS` for the link options

These variables have to be set correctly. On some machines it might be necessary to increase the sizes of internal compiler tables (e.g. `-Wf,-XNg1500` must be included in `CFLAGS` for SGI machines with IRIX 5.x).

The `Makefile` contains already some flags that have been tested on some machines.

```
SUN_CFLAGS =
IBM_CFLAGS = -qlanglvl=ansi
SGI_CFLAGS = -Wf,-XNg1500
T3D_CFLAGS =
T3E_CFLAGS = -h tolerant
SX4_CFLAGS = -h0 -hnovector
...
CFLAGS  = -I./include -I$(XINCLUDE) -w -O $(SUN_CFLAGS)
```

Unfortunately, the generated source files of the ADAPTOR translation are not very clean and reqire a rather tolerant C compiler.

4.3 Known Problems

The following problems may occur depending on the machine where the tool is installed:

1. For some machines, in the source file `Scanner.c` the redeclaration of `malloc` in yyInitialize has to be removed.
2. It could be necessary to undefine an environment variable for the compilation. Therefore insert in the file `Tree.h` in the subdirectory `include` the following line:

   ```
   #undef __STDC__
   ```

3. There might be problems for the redefinition of Macros.

   ```
   "Tree.c", line 33.0: 1506-213 (S)
       Macro name begintDefinitions cannot be redefined.
   "Tree.c", line 37.0: 1506-213 (S)
       Macro name beginint cannot be redefined.
   ```

   In this case, use a compiler flag that allows ANSI extensions (e.g. for the IBM xlc compiler the flag `-qlanglvl=extended`).

4. There might be problems for dynamic linking.

```
picie: Error: Illegal j/jal for dynamic link in function 'FindTheMainNode'.
  1. You can't Jump to the function. and
  2. You can't call to non-function.
```

In this case, find the file containing the routine (with fgrep) and remove the static definition.

```
  > static tCallGraph FindTheMainNode
  < tCallGraph FindTheMainNode
```

4.4 Predefined Options

Though all options for the translation can be changed via flags (see the *Users Guide of ADAPTOR* [BHK98]), some changes in the include file **global.h** in the subdirectory **src/include** are useful to have the correct predefined options for your parallel machine.

Relevant Columns

```
#define predefined_fix_length           72
#define predefined_strip_length         132
```

- The value of `predefined_fix_length` specifies the number of relevant columns for the input source. This value is only of importance if the source code is in fixed format. The default value should not be changed as it is the usual default value for all Fortran compilers.
- The value `predefined_strip_length` specifies the number of relevant columns for the output source. Usually, every Fortran compiler expects 72 relevant columns. But if the target compiler supports extended lines, the value 132 should be used to get better readable output files.

Sizes of FORTRAN Datatypes

The ADAPTOR source to source translation has to know how many bytes are used for the numerical data types `INTEGER` and `REAL` in Fortran. Furthermore, the number of bytes used for an address is needed.

These values are *very important* to have a correct interface to the HPF runtime system. The best way is to use the `sizeof` operator for the datatypes of C that correspond the FORTRAN datatypes.

```
#define predefined_default_int_size  (sizeof(int))
#define predefined_default_real_size (sizeof(float))
#define predefined_default_addr_size (sizeof(int *))
```

Attention: The above settings might not work on machines where cross-compilation is used and the compile server has different values from the parallel machine.

Source and Target Language

The source language is usually full HPF. It can be set to `HPF_SUBSET` or `HPF_BASE` (HPF 2.0, base language only) to restrict the language. But this should usually not be done as the predefined value.

The target language is usually `FORTRAN_90` for Fortran 90. If there is no Fortran 90 compiler available for the target architecture, it should be set to `FORTRAN_77`.

```
#define predefined_source_language      HPF
#define predefined_target_language      FORTRAN_90
```

Vectorization

ADAPTOR also supports vector architectures by generating better vectorizable code. In this case, the vectorization should be switched on by setting the value `VECTORIZE`. As there might be some conflicts for cache optimizations, it should be set to `NO_VECTOR` for all other architectures.

```
#define predefined_vectorization        NO_VECTOR
```

Parallelization

ADAPTOR also supports multiprocessor architectures that support thread parallelism for independent loops. By default, this kind of multiprocessing on a single node is not supported `NO_MP`.

```
#define predefined_sm_parallelization   NO_MP
```

For certain machines, parallelization directives can be generated.

- `OPEN_MP` inserts parallelization directives of Open MP.
- `SUN_MP` inserts parallelization directives for the SUN Fortran compiler.
- `CRAY_MP` inserts the CRAY parallelization directives.
- `SGI_MP` inserts the parallelization directives of the SGI compiler.

Other Options

The following options should not be changed.

```
#define predefined_target_model          ONLY_NODE
#define predefined_ddefault_kind         DDEFAULT_REPLICATED

#define predefined_MaxDistributedDims    3
#define predefined_optimization          4
#define predefined_safety                2
```

Section 7 describes how to test the compilation of the generated programs. Before, it is necessary to install the ADAPTOR runtime systems DALIB (for parallel SPMD execution) and UNILIB (for serial execution).

5 Installation of the DALIB and the UNILIB

5.1 Overview

The Fortran programs generated by the source-to-source translation have to be linked with the HPF runtime system. Therefore, the distributed array library DALIB must be installed on your machine. For serial machines or for running data parallel programs on a single node, the library UNILIB should also be installed.

The libraries have been implemented in C. Most source files are machine-independent. But every machine has some special machine-dependent realizations of communication functions and of the initialization routines as well as timing functions and random numbers [Bra98a].

- Concerning the message-passing routines of ADAPTOR, there are different versions available in different subdirectories.
- Concerning the different machines there are configuration files available in the directory `$PHOME/dalib/conf`.

5.2 Supported Message Passing Systems

Concerning the message-passing routines of ADAPTOR, there are different versions available that are based on wide-spread message passing systems.

The most wide-spread communication systems are certainly PVM [GBD+94] and MPI [GLS94]. If one of these message passing systems is available for your parallel platform, the DALIB can be easily adapted. Due to its high functionality and high availability, MPI is recommended.

Some versions are available that are directly based on native communication systems.

Table 1 shows the different versions of the DALIB that are currently available. Please use the corresponding sub-directory when generating the corresponding DALIB.

Directory	Message Passing System
`PVM3`	PVM (Parallel Virtual Machine), Oak Ridge National Laboratory
`MPI`	MPI (Message Passing Interface), Argonne Nat. Laboratory
`SHM`	Shared Memory (System V)
`CM5`	native implementation for the CM-5 based on CMMD 3.0 or higher
`PGON`	native implementation for the Intel Paragon based on NX
`CS`	native implementation for the Meiko CS-2
`SP`	native implementation for the IBM SP based on IBM's EUI interface
`CRAY`	special version based on PVM for the Cray T3D/T3E

Table 1: Supported message passing systems.

- The DALIB version for the Paragon XP/S works also for the Intel iPSC/860.
- For the DALIB on a Cray T3D a special PVM realization has been implemented where the processes are not dynamically created.

If there is not a native or special implementation available for the machine on which ADAPTOR should be installed, please use one of the general message passing systems.

5.3 Supported Machines

Unfortunately, the runtime system is not completely machine-independent. For this reason, conditional compiling and the macro preprocessor `m4` are utilized. Therefore the user must specify the architecture for which the DALIB is to be generated.

Table 2 lists the machines on which the DALIB has already been tested and which architecture name must be used for it (for very good reasons the names are identical to the names used for the public domain communication system PVM).

In the Makefile the corresponding variable `ARCH` has to be set to a correct item.

```
ARCH = PGON
COPT = ... -D$(ARCH)
```

ARCH	Machine
CM5	Thinking Machines' CM-5
PGON	Intel Paragon X/PS
CS	Meiko CS 2 (Computing Surface)
KSR	Kendall Square KSR
RS6K	IBM RS 6000 series workstations, IBM SP
SUN4	Sparc-1, Sparc-2 or Sparc-10 workstations, running Sun OS
SUN4SOL2	SUN Sparc workstations, running Solarix 2.x
SGI	Silicon Graphics workstations, SGI Origin, SGI Onyx
I860	Intel iPSC/860
HPPA	HP workstations
ALPHA	DEC alpha workstations
PPC	Parsytec Power GC
CENJU3	NEC Cenju-3
CRAY_T3D	Cray T3D
CRAY_T3E	Cray T3E
LINUX	PC's running LINUX
SX4	NEC SX-4
VPP	Fujitsu VPP architectures

Table 2: List of tested DALIB implementations

Please note that a corresponding configuration file `$ARCH.m4` must be available in the directory `$PHOME/dalib/conf`.

5.4 Support of System V Features

The latest version of ADAPTOR is able to take advantage of shared memory segments and of semaphores. These features can only be used on shared memory multiprocessor systems with these System V extensions.

If these features are available the following flags should be used:

```
COPT =  ... -DSHM -DSEM
```

The preprocessor will then include some special routines for shared memory segments to allow the use of shared arrays in HPF programs. By this feature, one can speed up the execution of certain HPF programs.

5.5 Vectorization of the Runtime System

On vector architectures, certain runtime routines should be vectorized to gain better performance. Currently, the DALIB contains pragma directives for the NEC SX-4 that are helpful for the automatic vectorization capabilities of the SX compiler.

5.6 Building the Libraries

For the installation of the DALIB go into the corresponding subdirectory belonging to your message passing system. If a native message passing system is used, just type make. If a general message passing system is used, the Makefile has to be edited to set correct entries for the architecture and installation directory of the general message passing system (see section 5.2).

After the compilation there should be at least two libraries in the directory:

- `unilib.a` is the library with some subroutines that are used for single node programs (timing, random numbers, array operations).
- `dalib.a` is the library with subroutines for message passing, reductions, topologies and all the other stuff.

Of course it is possible to have different versions of the DALIB available on one machine using different message passing systems, e.g. a PVM version, and a version based on the native implementation (see 6.2.6).

5.6.1 Known Problems

- For some machines it is required to call `ranlib` for the runtime system libraries. For other machines the corresponding system call should be removed from the `Makefile`.
- For some machines, in the soure file `dalib.m4` the redeclaration of `malloc` has to be removed.

5.7 Using Other Machines

If you have a parallel machine that is not supported until now, please read the documentation of DALIB carefully in order to port one of the existent codes onto your machine.

The DALIB is already very portable. Only some modules have to be changed. These are the modules for the initialization (`init.m4`), for the point-to-point communication (`pt2pt.m4`), for timing (`timing.m4`) and for the realization of some intrinsics (`intrinsics.m4`).

Broadcasting `broadcast.m4` and reductions `reduction.m4` are realized based on point-to-point communication using a tree communication pattern. It might be better to optimize these modules for the available machines though it is not really necessary.

The example directory provides many programs that can be used for testing the implementation of the DALIB on a new machine.

6 Installation of gmdhpf

The sources for the compiler driver gmdhpf are stored in the subdirectory **src1**. The sources are all written in C and should run on all UNIX machines.

The executable `gmdhpf` for the driver and `genlist` for producing listings will be generated by simply typing make.

```
cd $PHOME/src1
make
mv gmdhpf ../bin
mv genlist ../bin
make clean           ! after successful installation
```

Depending on the machine where ADAPTOR is installed it might be necessary to make some changes in the Makefile (see section 4.2).

6.1 Configuration of the Compiler Driver

You get the actual configuration settings of `gmdhpf` with the command:

```
gmdhpf -settings
```

```
Default settings read in from following configuration files:
./.adprc

ADAPTOR Home Directory : /home/resi/adaptor
target_machine:        : Sun OS
target_communication   : PVM 3.x
DALIB:                 : /home/resi/adaptor/dalib/PVM3/dalib.a
Mess.-Pass.-Lib.       : $PVM_ROOT/lib/$PVM_ARCH/libpvm3.a

Verbose = 0,    Dryrun = 0,   Keep = 0

invoke cpp for cpp input-files
       CPP       : cpp
       CPP_OPT   : -P

invoke fadapt for fadapt input-files
       FADAPT     : /home/resi/adaptor/bin/fadapt
       FADAPT_OPT : -strip 72

invoke fc for fc input-files
       FC        : f77
       FC_OPT    : -O -e -f -w

LINK STEP WILL BE INVOKED if no error occurres at compilation step.
LD will be taken for generation of SPMD program a.out

Generation of SPMD program with:
       LD        : f77
       LD_OPT    : -lnsl -lsocket

Generation of SINGLE NODE program with:
       LD1       : f77
       LD1_OPT   :
```

6.2 Default Settings

The compiler driver uses default settings for the different tools and their options.

These can be changed by using **configuration files** (see 6.3) or by changing the predefined settings in the file `include/gmdhpfglobal.h` as described by the following:

6.2.1 Choosing Target Machine and Target Communication

The default settings for the compilers, linkers, runtime library and message passing library depend on the chosen target machine and target communication (see 3 and 4).

These predefinitions should be changed to generate a corresponding compiler driver:

```
#define predefined_target_machine        SUN4
#define predefined_target_communication  PVM3
```

- The value of the predefined target machine should be the name of the machine for which ADAPTOR will be used directly. The name must be an item of table 2.
- The predefined target communication should be the most efficient one that is available on the target machine. Beside `PVM` and `MPI` some native implementations can be used (see table 1). In the last case, `NATIVE` should be the predefined value for the constant `predefined_target_communication`.

6.2.2 Default Settings for the Message Passing Libraries

```
#define predefined_MPI_LIB "/home/hlrz/mpich/lib/solaris/ch_p4/libmpi.a"
#define predefined_PVM_LIB "$PVM_ROOT/lib/$PVM_ARCH/libpvm3.a"
#define predefined_SHM_LIB "$SHM_ROOT/lib/$ARCH/libshm.a"
```

In most cases you will have to change the names of the libraries according to your environment. You may either write down the approriate names of your libs, or the appropriate -l options to make the linker search the library search path, or an empty string, indicating that the linker knows all needed libs.

6.2.3 Default Settings for the DALIB

```
#define predefined_DALIB   "$PHOME/dalib"
```

sets the search path for the DALIB. The filename of the DALIB is: `predefined_DALIB/<MP>/dalib.a` rsp. `predefined_DALIB/<MP>/unilib.a`, MP is one of the directories in table1, e.g.: `$PHOME/dalib/SHM/dalib.a`.

If you don't name the dalib according to this rule, you can specify the complete name in a configuration file (see 6.3).

6.2.4 Default Settings for fadapt

```
#define predefined_FADAPT "$PHOME/bin/fadapt"
```

6.2.5 Default Settings for the Fortran Compiler

The generated programs of ADAPTOR can be compiled with every FORTRAN 77 compiler. Only if some certain Fortran 90 features are used, a Fortran 90 compiler is required.

When choosing the right compiler options the following should be observed:

- switch off the warnings (for most compilers this is the option `-w`),
- take advantage of the best optimization (usually `-O`),
- use the option that supports extended source lines (e.g. `-e` or `-Mextend`).

```
f77 -w -O -e -c cube.f
f77 -w -O -e -c node1.f
```

- `FC` is the Fortran compiler for the generated program and

- OPT are the options used for this compilation,
- As the front end system of the Cray T3D is a Cray Y-MP, it should be observed that the DALIB will be compiled for the T3D (using the flag -Tcray-t3d).

Table 3 gives an overview about the options that should be used with the different Fortran compilers on different machines (already preset in include/gmdhpfglobal.h).

target_machine	FC	FC_OPT	FC_MP
cm5	f77	-O -e -f -w	
pgon	if77	-O -Mvect -Knoieee -Mextend	
cs	f77	-O -e -f -w	
ksr	f77	-O2 -i4 -r8	
rs6k	xlf	-O3 -qstrict -w	
sun4	f77	-O -e -f -w	
sun4sol2	f77	-O -e -f -w	-explicitpar
sgi	f77	-O3 -mips4 -extend_source -w -static	-cray_mp
i860	if77	-O -Mvect -Knoieee -Mextend	
hppa	fort77	-O +es -w +U77	
alpha	f77	-O	
ppc	f77.px	-O -e -f -w	
cenju3	cjf77	-O -f -w -Zextend_source	
t3d	cf77	-Wf -e -v -Oscalar3 -Ccray-t3d	
t3e	f90	-O3 -N132	
linux	f77	-O -e -f -w	
sx4	f90	-float0	
vpp	frt	-w	
unknown	f77	-O -e -f -w	

Table 3: Fortran compilers and their options invoked by default for all supported target machines

6.2.6 Default Settings for the Linker

Every **parallel program** must be linked with the DALIB runtime system and with the corresponding message passing library.

```
f77 -o cube cube.o $PHOME/dalib/PVM3/dalib.a \
                   $PVM_ROOT/lib/SUN4/libpvm3.a
```

On some machines the linker itself might find automatically the message passing library (e.g. mpxlf on the IBM SP for using MPL or mpixlf for using MPI).

Now compare the values of the following variables to the values needed for your machine:

- LD is the linker and LD_OPT are the link options,
- LD1 is the linker for a serial version of the HPF program and LD1_OPT the options used for this linker.

In the file include/gmdhpfglobal.h the default values can be reset for the different machines.

If ADAPTOR has generated a **serial program** from the data parallel HPF program, the library unilib.a must be linked with the compiled program. It is not necessary to link any message passing library.

```
f77 -o node1 node1.o $PHOME/dalib/PVM3/unilib.a
```

If `gmdhpf` is called with option `-1` this will be done automatically.

Table 4 gives an overview about the options that should be used with the different Fortran linkers on different machines.

target_machine	ld	ld_opt	ld1	ld1_opt
cm5	f77		f77	
pgon	if77		if77	
cs	f77	-L/opt/MEIKOcs2/lib -lmpsc -lew-lelan	f77	
ksr	f77		f77	
rs6k	mpxlf		xlf	
sun4	f77		f77	
sun4sol2	f77		f77	
sgi	f77		f77	
i860	if77		if77	
hppa	fort77		fort77	
alpha	f77		f77	
ppc	f77.px		f77.px	
cenju3	cfj77		cfj77	
t3d	f77	-Ccray-t3d	f77	-Ccray-t3d
t3e	f90	-Xm	f90	
linux	f77		f77	
sx4	f90		f90	
vpp	frt		frt	
unknown	f77		f77	

Table 4: Linkers and their options invoked by default for all supported target machines

6.3 Customization Files

The predefined settings for the translation can be redefined in customization files as well as by changing `$PHOME/src1/include/gmdhpfglobal.h`.

Customization file must be named `.adprc`. The compiler driver searches these files in the following directories:

- first in the installation directory of ADAPTOR, thus `installation wide settings` should be made in `$PHOME/.adprc`.
- then in the home directory of the user `$HOME/.adprc`,
- at last in the working directory `./.adprc`.

If the driver finds a customization file, the settings defined there replace the corresponding default settings from `$PHOME/src1/include/gmdhpdglobal.h`. If more than one customization file exists, the definitions of the next one found override the hence valid settings.

The filename for the `DALIB` and for the message passing libraries (`MPI_LIB`, `PVM_LIB` or `SHM_LIB`) can be set directly.

Example for possible settings:

```
  TARGET_MACHINE = SUN4
  TARGET_COMMUNICATION = MPI
#
  CPP  = cpp
  CPP_OPT = -P
  FADAPT = $PHOME/bin/fadapt
  FADAPT_OPT =
  FC = f77
  FC_OPT = -O -w
  LD  = f77
  LD_OPT =
  LD1 = f77
  LD1_OPT =
  DALIB   = /home/brandes/adaptor/dalib/MPI/dalib.a
  PVM_LIB = $PVM_ROOT/lib/$PVM_ARCH/libpvm3.a
  MPI_LIB = -lmpi
  SHM_LIB =
```

7 Testing the Installation

7.1 Setting the Environment

You should include the **bin** subdirectory of the ADAPTOR installation directory in your environment variable PATH and the **man** subdirectory in your MANPATH:

```
setenv PHOME ...
setenv PATH $PHOME/bin:$PATH
setenv MANPATH $PHOME/man:$MANPATH
```

or in the following way:

```
set path  ($PHOME/bin $path)          export PATH=$PHOME/bin:$PATH
set manpath ($PHOME/man $manpath)     export MANPATH=$PHOME/man:$MANPATH
```

7.2 Testing the fadapt Installation

Should change to the directory with the example programs:

```
cd $PHOME/hpf_examples/simple
```

The file `simple.hpf` can now be translated with the following command:

```
fadapt simple.hpf
```

After the translation the following file should have been generated:

- `cube.f` (program running on all nodes),

It is also possible to generate a FORTRAN 77 program that will run on a single node:

```
fadapt -1 simple.hpf
```

In this case a program called `node1.f` will be generated. In the generated program the array operations are translated to serial loops and the dynamic arrays become pseudo-dynamic arrays. By this way HPF programs can also be tested on serial machines.

7.3 Testing the gmdhpf Installation

```
cd $PHOME/hpf_examples/simple
```

The file `simple.hpf` can now be translated with the following commands:

- for generation of parallel program:

  ```
  gmdhpf -v -o simple simple.hpf
  ```

- for generation of single node program:

  ```
  gmdhpf -v -1 -o simple simple.hpf
  ```

7.4 Running the Parallel Applications

The execution of the parallel programs depend on the available parallel platform and the parallel programming environment provided on them. Please refer to the corresponding documentation of your parallel platform.

8 Using Libraries

8.1 Installation of the X-Windows Library

In the subdirectory `$PHOME/dalib/X` some functions for X-Windows are implemented. This library is not really necessary hence it is required for some example programs. It realizes the display of two-dimensional arrays with pixels on a screen of X-Windows.

```
cd $PHOME/dalib/X
make
```

Afterwards the library `libadpx.a` should be available in the directory `$PHOME/dalib`.

9 Announcing the Installation

After the successful installation the following announcement should be made for all other interested users:

```
ADAPTOR has been installed on this machine in the installation
directory '<PHOME>'

a) Step 1 : Set the environment variable PHOME to this directory

   setenv PHOME <PHOME>

b) Step 2 : Include the bin and man directory in your path

   setenv PATH $PHOME/bin:$PATH
   setenv MANPATH $PHOME/man:$MANPATH
```

```
c) You will find many examples programs in $PHOME/hpf_examples
   (make local copies in your own directory).

d) You will find documentation files in $PHOME/docs.
   It is recommended to start with the Users Guide (uguide.ps).
```

10 Maintenance and Problems

The authors would be grateful to all hints for problems and errors that users have had with this tool (please write a mail describing the problem to `brandes@gmd.de`). Your comments are welcome and will help to improve the quality of this free software product.

Acknowledgement

We are specially indebted to the many people whose help was very useful to remove the bugs out of the former versions. We thank Falk Zimmermann (NEC, Germany) for his support on the NEC SX-4 architecture. The help of David Singleton (Fujitsu, Austria) was very valuable for porting ADAPTOR to the Fujitsu VPP architecture.

References

[BHK98] T. Brandes and R. Höver-Klier. ADAPTOR User's Guide (Version 6.0). Technical documentation, GMD, June 1998. Available via anonymous ftp from `ftp.gmd.de` as `gmd/adaptor/docs/uguide.ps`.

[Bra98a] T Brandes. ADAPTOR Distributed Array Library (Version 6.0). Technical Documentation Adaptor 4, GMD, May 1998. Available via anonymous ftp from `ftp.gmd.de` as `gmd/adaptor/docs/dalib.ps`.

[Bra98b] T. Brandes. ADAPTOR Programmer's Guide(Version 6.0). Technical documentation, GMD, June 1998. Available via anonymous ftp from `ftp.gmd.de` as `gmd/adaptor/docs/pguide.ps`.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing.* Scientific and Engineering Computation Series. The MIT Press, Cambridge, MA, 1994.

[GLS94] W. Groop, E. Lusk, and A. Skjellum. *Using MPI : Portable Parallel Programming with the Message-Passing Interface.* Scientific and Engineering Computation Series. The MIT Press, Cambridge, MA, 1994.