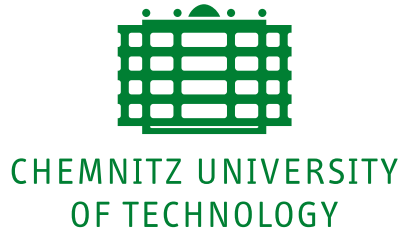
Computer Architecture Technical Report

TUC / RA-TR-2006-01

Date: 11th Jan 2006
(Last Build: 17th January 2006)

A short Performance Analysis of Abinit under different build environments

Robert Kullmann, Torsten Hoefler
{robk,htor}@informatik.tu-chemnitz.de



Chemnitz University of Technology
Department of Computer Science
Computer Architecture
Prof. Dr. W. Rehm

A short Performance Analysis of Abinit under different build environments

Robert Kullmann, Torsten Hoefler

{robk,htor}@informatik.tu-chemnitz.de

Keywords

Abinit, automated benchmarking, compiler comparison

1 Introduction

Abinit [1] uses the Density Functional Theory (DFT) to calculate the electronic structure of atomic systems. The code can be build for sequential as well as parallel execution under shared or distributed memory environments. It is tested with the Athlon MP and Intel Xeon CPU architecture with different compilers and math libraries (see section 2 for details) trying to find out which combination of compiler/math library is performing best under which architecture.

2 Compiling

For both the sequential and the parallel executable Abinit is compiled with the Intel [4] and the GNU G95 compiler [5]. Math libraries used are Abinit's internal math routines (which seem to be the reference implementation from netlib.org), the Intel Math Kernel Library (IMKL) [6], AMD's Core Math Library (ACML) [7], the optimized BLAS library from Kazushige Goto (Goto BLAS) [8] and the Atlas Math Library [9]. For parallelization Abinit is compiled with the MPICH2 MPI library [2]. With the Intel compiler a specific compiler flag is used to utilize shared memory parallelism via threading through the auto-parallelizer.

The following software versions are used:

- Abinit: 4.5.2
- Intel compiler: 8.1
- GNU G95 compiler: 2005-06-26
- Intel MKL: 7.2.1-3
- AMD ACML: 2.6.0
- Goto BLAS: 0.99-3
- Atlas: 3.7.10
- MPICH2: 1.0.1

All these combinations are tested on an Athlon MP and Intel Xeon processor architecture. Parts of the `makefile_macros` used for configuring and compiling Abinit are listet in the Appendix section A.1 and A.2.

2.1 Specific Problems

1. Intel compiler -O3 bug:
   The `-O3` optimization has to be disabled for several directories, due to compiler bugs which would lead to endless compiling.
2. Trying to include shared Goto BLAS library into statically build binary:
   When using the Goto BLAS library the Abinit executable cannot be build statically because when running such an executable it would complain about a "bad ELF interpreter":

   ```
   /usr/lib/libc.so.1: bad ELF interpreter: No such file or directory
   ```

It is, however, possible to link Abinit against the shared Goto BLAS library while all other libraries are supplied in static form. The resulting executable will then depend only on a few shared libraries (the Goto BLAS library plus `libm.so.6`, `libc.so.6` and `/lib/ld-linux.so.2`).

3. OpenMP cannot be used:
   Even so OpenMP statements are included within Abinit they cannot be utilized as they are incorrect. When specifying the relevant compiler flags the compiler will complain about the OpenMP statements.

2.2 Compilation Matrix

For each CPU architecture (Athlon MP and Intel Xeon) the following combinations of compiler/math library are tested with executables built for sequential and parallel run. Each cell contains the compilers used for that specific combination of math library and parallelization.

Math Library \ Parallelization	sequential	MPICH2	Intel Autoparallel
Abinit Internal	Intel / G95 compiler	Intel / G95 compiler	Intel compiler
Intel MKL	Intel / G95 compiler	Intel / G95 compiler	Intel compiler
AMD ACML	Intel / G95 compiler	Intel / G95 compiler	Intel compiler
Goto BLAS	Intel / G95 compiler	Intel / G95 compiler	Intel compiler
Atlas	Intel / G95 compiler	Intel / G95 compiler	Intel compiler

3 Benchmark Environment

Two local clusters are used for benchmarking. The first consisting of 8 Dual Xeon 2.4Ghz with 2GB main memory per node and an I/O disk system with an approximate bandwidth of 50MB/s. The second cluster consists of 8 Dual Athlon MP 1.4GHz. Each node has 512MB main memory and the I/O disk system achieves approximately 50MB/s. The nodes are interconnected with Gigabit Ethernet.

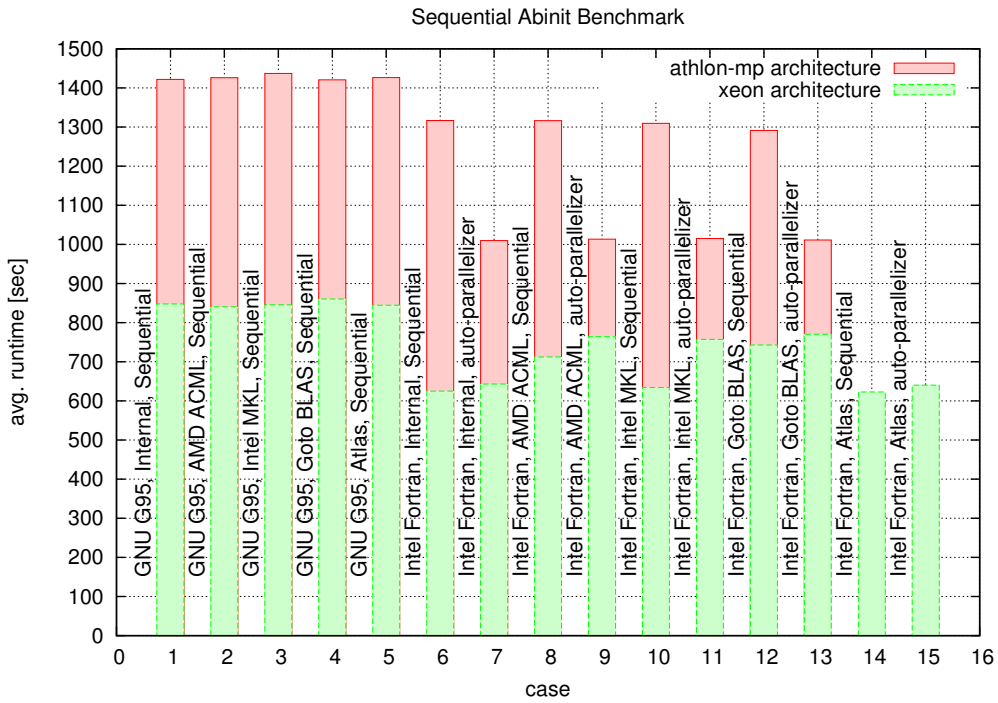
4 Benchmark Results

The following sections list all aquired benchmark results divided into the sequential and parallel variant of Abinit.

4.1 Sequential Variant

For the sequential variant the important parameters of the Abinit run are set to the following values:

- natom = 14
- nkpt = 2
- nband = 56
- npw = 7714

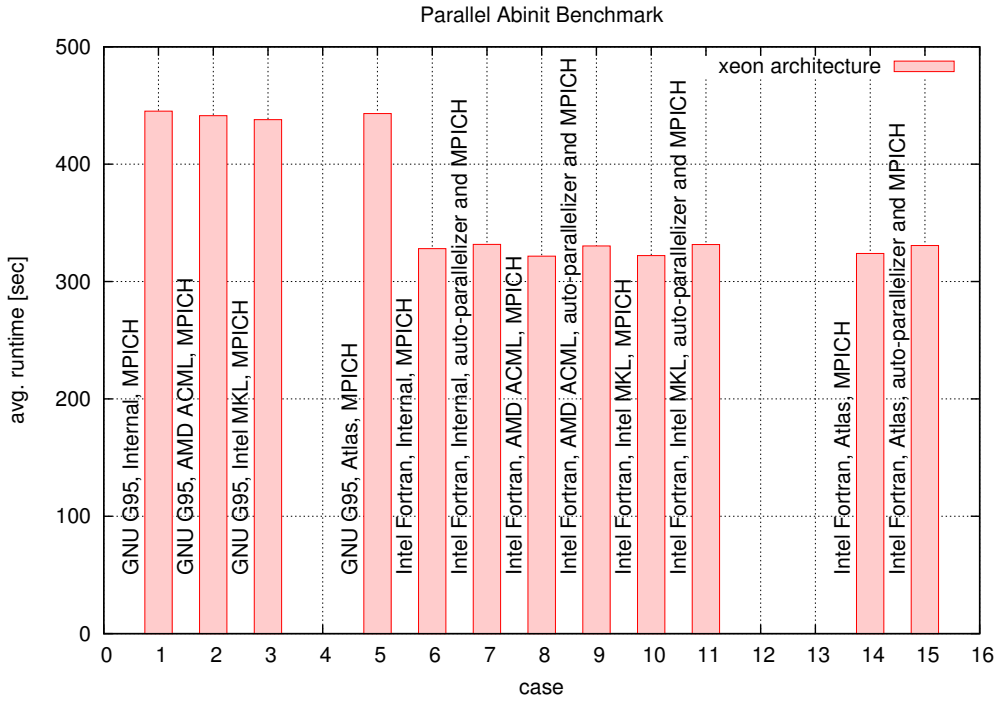


The diagram above shows the execution times of an Abinit run on the Athlon MP and Xeon architecture. There is no significant difference in runtime between the various math libraries when compiled with the GNU G95 compiler. Execution time difference between the math libraries in combination with the Intel compiler is greater than with the G95 compiler, but not to a great extent. Is is, however, noticeable that the auto-parallelizer feature does make a difference on SMP machines.

4.2 Parallel Variant

For the parallel variant the important parameters of the Abinit run are set to the following values:

- natom = 14
- nkpt = 2
- nband = 56
- npw = 7714



As it is with the sequential variant of Abinit, there is no significant difference in runtime between the math libraries with parallel execution of Abinit either. The overall execution time of the parallel variant is only about half of the time of the sequential one, as it was to be expected.

5 Summary and Conclusion

This short analysis shows that there is no significant gain in prefering a particular math library over another. That is mainly because we discovered in another study [3] that Abinit uses only about 3% of the respective math library's functions.

Different compilers can make a difference in execution time. The benchmarks show that the Intel compiler is capable of reducing the runtime especially when combined with parallelization features such as the proprietary auto-parallelizer. OpenMP should produce a similar difference once it works correctly.

Appendix

A Important parts of the makefile_macros

The following sections show all important parts of the `makefile_macros` files that were used to configure Abinit for the different CPU architectures.

A.1 Compiler flags and math library configurations for Athlon MP

A.1.1 Compiler flags

1. GNU G95 Fortran compiler flags:

```
# Fortran optimized compilation
FC=g95

COMMON_FLAGS=-O3 -floop-optimize2 -ftree-loop-linear -ftree-loop-im \
-ftree-loop-ivcanon -fivopts -ftree-vectorize -funroll-all-loops \
-freorder-blocks-and-partition -march=athlon-mp -mfpmath=sse -mmmx -msse -m3dnow
FFLAGS=$(COMMON_FLAGS) -ffree-form
```

```
FFLAGS_LIBS=$(COMMON_FLAGS) -ffixed-form

# link the binaries statically
FLINK=-static
```

2. GNU C compiler flags:

```
CC=gcc
CFLAGS=-O3 -funroll-all-loops -march=athlon-mp -mfpmath=sse -mmmx -msse -m3dnow
```

3. Intel Fortran compiler flags:

```
# Fortran optimized compilation
FC=ifort

COMMON_FFLAGS=-arch pn4 -axW -cpp -ip -tpp6
FFLAGS = $(COMMON_FFLAGS) -free -O3

# optimization does not work in these dirs (endless loop)
FFLAGS_Src_2psp=$(COMMON_FFLAGS) -O0 -free
FFLAGS_Src_3iovars=$(COMMON_FFLAGS) -O0 -free
FFLAGS_Src_9drive=$(COMMON_FFLAGS) -O0 -free

FFLAGS_LIBS=$(COMMON_FLAGS) -O3 -fixed

# link the binaries statically
FLINK=-static
```

4. Intel C compiler flags:

```
CC=icc
CFLAGS=$(COMMON_FLAGS) -O3
```

For the Intel compiler's auto-parallelizer the following `COMMON_FFLAGS` variable was used:

```
COMMON_FFLAGS=-arch pn4 -axW -cpp -ip -tpp6 -parallel
```

A.1.2 Math Library Configurations

1. Abinit's internal math functions:

   ```
   LIBS = $(LAPACK) $(BLAS)
   ```

   It is perfectly possible to comment or leave out the `LIBS` line at all.

2. AMD ACML math library:

   ```
   MATHLIBDIR=<path to the 32-bit ACML>/gnu32_nosse2/lib
   MY_LIBS=$(MATHLIBDIR)/libacml.a /usr/lib/gcc-lib/i386-pc-linux/3.2.3/libg2c.a
   LIBS = $(MY_LIBS)
   ```

3. Intel MKL math library:

   ```
   MATHLIBDIR=<path to the IMKL>/lib/32
   MY_LIBS=$(MATHLIBDIR)/libmkl_lapack.a $(MATHLIBDIR)/libmkl_ia32.a \
   $(MATHLIBDIR)/libguide.a /usr/lib/libpthread.a
   LIBS = $(MY_LIBS)
   ```

4. Goto BLAS math library:

   ```
   MATHLIBDIR=<path to Goto BLAS>
   MY_LIBS=-L$(MATHLIBDIR) -lgoto_coppermine-32-r0.99-3
   LIBS = $(MY_LIBS) $(LAPACK) /usr/lib/libpthread.a /usr/lib/libc.a
   ```

   For the (shared library) Goto BLAS it is necessary to not use `FLINK=-static`, but leave the `-static` out (`FLINK=`)! To minimize the used shared libraries `/usr/lib/libc.a` is additionally included here.

5. Atlas math library:

   ```
   MATHLIBDIR=<path to the Atlas installation>/lib
   ```

```
MY_LIBS=$(MATHLIBDIR)/liblapack.a $(LAPACK) $(MATHLIBDIR)/libf77blas.a \
$(MATHLIBDIR)/libcblas.a $(MATHLIBDIR)/libatlas.a
LIBS = $(MY_LIBS)
```

For the Intel Compiler it is necessary to add the following library after all others: `libsvml.a`.
For example:

```
LIBS = $(MY_LIBS) <path to the Intel compiler>/lib/libsvml.a
```

where `${MY_LIBS}` includes all other (necessary) libraries.

A.1.3 MPI Configuration

```
#####################################################################
# For the parallel version : MPICH / MYRINET

MPI_LIB_DIR=<path to MPICH2>

# Compiler flags and definitions
FFLAGS_PAR= $(FFLAGS) -I $(MPI_LIB_DIR)/include

# List of machine-dependent routines
MACHINE_DEP_C_PAR_SUBS_LIST=etime.par

# Location of the MPI library
MPI_A=$(MPI_LIB_DIR)/lib/libmpich.a $(MPI_LIB_DIR)/lib/libfmpich.a

# Include blas, lapack, and any other libraries here
LIBS_PAR=$(LIBS) $(MPI_A)
```

A.2 Compiler flags and math library configurations for Xeon

A.2.1 Compiler flags

1. GNU G95 Fortran compiler flags:

```
# Fortran optimized compilation
FC=g95

COMMON_FLAGS=-O3 -floop-optimize2 -ftree-loop-linear -ftree-loop-im \
-ftree-loop-ivcanon -fivopts -ftree-vectorize -funroll-all-loops \
-freorder-blocks-and-partition -march=pentium4 -mfpmath=sse -mmmx -msse -msse2
FFLAGS=$(COMMON_FLAGS) -ffree-form

FFLAGS_LIBS=$(COMMON_FLAGS) -ffixed-form

# link the binaries statically
FLINK=-static
```

2. GNU C compiler flags:

```
CC=gcc
CFLAGS=-O3 -funroll-all-loops -march=pentium4 -mfpmath=sse -mmmx -msse -msse2
```

3. Intel Fortran compiler flags:

```
# Fortran optimized compilation
FC=ifort

COMMON_FLAGS=-axN -cpp -w -tpp7 -ip
FFLAGS=$(COMMON_FLAGS) -FR -O3

# optimization does not work in these dirs (endless loop)
```

```
FFLAGS_Src_2psp = $(COMMON_FLAGS) -FR -O0
FFLAGS_Src_3iovars = $(COMMON_FLAGS) -FR -O0
FFLAGS_Src_9drive = $(COMMON_FLAGS) -FR -O0

FFLAGS_LIBS=$(COMMON_FLAGS) -O3

# link the binaries statically
FLINK=-static
```

4. Intel C compiler flags:

```
CC=icc
CFLAGS=$(COMMON_FLAGS) -O3
```

For the Intel compiler's auto-parallelizer the following `COMMON_FFLAGS` variable was used:

```
COMMON_FLAGS=-axN -cpp -w -tpp7 -ip -parallel
```

A.2.2 Math Library Configurations

1. Abinit's internal math functions:

   ```
   LIBS = $(LAPACK) $(BLAS)
   ```

   It is perfectly possible to comment or leave out the `LIBS` line at all.

2. AMD ACML math library:

   ```
   MATHLIBDIR=<path to the 32-bit ACML>/gnu32/lib
   MY_LIBS=$(MATHLIBDIR)/libacml.a /usr/lib/gcc-lib/i386-pc-linux/3.2.3/libg2c.a
   LIBS = $(MY_LIBS)
   ```

3. Intel MKL math library:

   ```
   MATHLIBDIR=<path to the IMKL>/lib/32
   MY_LIBS=$(MATHLIBDIR)/libmkl_lapack.a $(MATHLIBDIR)/libmkl_ia32.a \
   $(MATHLIBDIR)/libguide.a /usr/lib/libpthread.a
   LIBS = $(MY_LIBS}
   ```

4. Goto BLAS math library:

   ```
   MATHLIBDIR=<path to Goto BLAS>
   MY_LIBS=-L$(MATHLIBDIR) -lgoto_northwood-32-r0.99-3
   LIBS = $(MY_LIBS) $(LAPACK) /usr/lib/libpthread.a /usr/lib/libc.a
   ```

   For the (shared library) Goto BLAS it is necessary to not use `FLINK=-static`, but leave the `-static` out (`FLINK=`)! To minimize the used shared libraries `/usr/lib/libc.a` is additionally included here.

5. Atlas math library:

   ```
   MATHLIBDIR=<path to the Atlas installation>/lib
   MY_LIBS=$(MATHLIBDIR)/liblapack.a $(LAPACK) $(MATHLIBDIR)/libf77blas.a \
   $(MATHLIBDIR)/libcblas.a $(MATHLIBDIR)/libatlas.a
   LIBS = $(MY_LIBS)
   ```

For the Intel Compiler it is necessary to add the following library after all others: `libsvml.a`.

For example:

```
LIBS = $(MY_LIBS) <path to the Intel compiler>/lib/libsvml.a
```

where `${MY_LIBS}` includes all other (necessary) libraries.

A.2.3 MPI Configuration

```
#####################################################################
# For the parallel version : MPICH / MYRINET

MPI_LIB_DIR=<path to MPICH2>
```

```
# Compiler flags and definitions
FFLAGS_PAR= $(FFLAGS) -I $(MPI_LIB_DIR)/include

# List of machine-dependent routines
MACHINE_DEP_C_PAR_SUBS_LIST=etime.par

# Location of the MPI library
MPI_A=$(MPI_LIB_DIR)/lib/libmpich.a $(MPI_LIB_DIR)/lib/libfmpich.a  # -Vaxlib

# Include blas, lapack, and any other libraries here
LIBS_PAR=$(LIBS) $(MPI_A)
```

References

[1] X. Gonze, J.-M. Beuken, R. Caracas, F. Detraux, M. Fuchs, G.-M. Rignanese, L. Sindic, M. Verstraete, G. Zerah, F. Jollet, M. Torrent, A. Roy, M. Mikami,Ph. Ghosez, J.-Y. Raty and D.C. Allan: *First-principles computation of material properties: the ABINIT software project, Computational Materials Science 25, 478-492 (2002), see www.abinit.org*

[2] William Gropp: *A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard, Intl. Journal of Parallel Computing, Vol. 22, Nr. 6, pp 789-828, 1996*

[3] Torsten Hoefler, Rebecca Janisch, Wolfgang Rehm: *Performance Analysis of Abinit on a Cluster System, in Parallel Algorithms and Cluster Computing, 2006*

[4] *Intel Compilers, http://www.intel.com/cd/software/products/asmo-na/eng/compilers/index.htm, up at Jan 12 2006*

[5] *GNU G95 Compiler, http://g95.sourceforge.net/, up at Jan 12 2006*

[6] *Intel Math Kernel Library, http://www.intel.com/cd/software/products/asmo-na/eng/perflib/219769.htm, up at Jan 12 2006*

[7] *AMD Core Math Library, http://developer.amd.com/acml.aspx, up at Jan 12 2006*

[8] *High-Performance BLAS by Kazushige Goto, http://www.tacc.utexas.edu/ kgoto/, up at Jan 12 2006*

[9] *Automatically Tuned Linear Algebra Software, http://math-atlas.sourceforge.net/, up at Jan 12 2006*