VASP user guide in KISTI supercomputer

2014. 1.

김흥식[1], 강지훈[2]

(1) 한국과학기술원 자연과학연구소

(2) KISTI 국가슈퍼컴퓨팅연구소 슈퍼컴퓨팅서비스센터

< 목 차 >

1. VASP 소개

VASP (Vienna Ab initio Simulation Package)는 제일원리 (first principle) 계산을 통해 물질의 원자 구조를 계산하는 시뮬레이션 패키지입니다. VASP은 다체 슈뢰딩거 방정식 (many-body Schrödinger equation) 을 풀기 위해 DFT (Density Functional Theory)의 Kohn-Sham 방정식을 이용하는 방법과 HF(Hatree-Fock) 근사의 Roothaan 방정식을 이용하는 방법을 제공하고 있으며, DFT에 HF 방법을 이용한 hybrid 방법 또한 제공하고 있습니다. 또한 Green's function 방법(GW quasiparticles, ACFDT-RPA)와 다체 섭동이론 (many-body perturbation theory) 방법 또한 계산이 가능합니다. 표 1은 VASP이 제공하고 있는 방법을 나타내고 있습니다.

본 단락의 내용은 VASP 홈페이지에 소개된 내용의 일부이며 보다 상세한 내용은 VASP 홈페이지 (www.vasp.at)을 참고하시기 바랍니다. 또한 VASP은 상용 SW이기 때문에 KISTI에서는 설치 방법에 대한 안내를 하고 있으며, 사용에 필요한 라이선스는 직접 구입하셔야 이용이 가능합니다.

<표 1> VASP features

Simulation theory	Features
Functionals	- LDA, GGAs, metaGGAs - Hartree-Fock, Hartree-Fock/DFT hybrids
First derivatives	- Forces and stress tensor for DFT, Hartree-Fock, and hybrid functionals
Dynamics and relaxation	- Born-Oppenheimer molecular dynamics - Relaxation using conjugate gradient, Quasi-Newton or damped molecular dynamics - Nudged elastic band methods (transition states search) - Climbing dimer method (transition state search)
Magnetism	- Collinear and non-collinear - Spin-orbit coupling - Constrained magnetic moments approach
Linear response to electric fields	- Static dielectric properties - Born effective charge tensors - Piezoelectric tensors (including ionic contributions)
Linear response to ionic displacements	- Phonons - Elastic constants (including ionic contributions) - Internal strain tensors
Optical properties	- Frequency dependent dielectric tensors in the independent particle approximation - Frequency dependent tensors in the RPA and TD-DFT - Cassida's equation for TD-DFT and TD-Hartree-Fock
Berry phases	- Macroscopic polarization - Finite electric fields
Green's function methods	- GW quasiparticles - ACFDT total energies in the RPA
Many-body perturbation theory	- 2nd-order Møller-Plesset perturbation theory

2. VASP 설치 (I) - KISTI 슈퍼컴퓨터 4호기 Gaia 시스템

아래 내용은 VASP 5.3.3 버전을 기준으로 작성되었으며, 기타 버전에 대해서는 본 내용을 토대로 설치 위치 및 컴파일 환경에 대해 VASP이 제공하고 있는 가이드 내용을 반영하시기 바랍니다.

2.1. VASP 기본 라이브러리 컴파일

VASP 본 코드를 컴파일하기 위해서는, 먼저 VASP에서 공통적으로 사용되는 라이브러리들을 컴파일해야 합니다. VASP 소스 파일의 압축을 풀면, 다음과 같은 2개의 디렉토리가 만들어 집니다.

```
./vasp.5.3
./vasp.5.lib
```

먼저 vasp.5.lib 디렉토리로 들어가서, makefile 파일을 생성합니다. 몇 가지 시스템들에 대한 예제 makefile이 있기 때문에, 이 중 하나를 makefile로 복사한 후에, 컴파일러 관련 옵션을 다음과 같이 고치면 되며, 완전한 makefile은 부록을 참고하시기 바랍니다. FFLAGS option은 Gaia의 power 프로세서 버전에 맞게 고쳐도 무방합니다.

```
# C-preprocessor
CPP     = /usr/ccs/lib/cpp -P -C $*.F >$*.f
FC      = xlf

CFLAGS = -O
FFLAGS = -O3 -qarch=auto -qtune=auto -qstrict
FREE   = -qfree=f90
```

주의할 점은, make를 실행하여 라이브러리를 컴파일하기 전에, OBJECT_MODE 환경변수를 64비트로 설정해주어야 합니다. Bash shell 에서는 다음 명령어를 실행하면 됩니다.

```
$ export OBJECT_MODE=64
```

컴파일이 성공적으로 마무리되면, 디렉토리에 libdmy.a 파일이 만들어지고, VASP 본 코드를 컴파일 할 때 이 라이브러리를 불러오게 됩니다.

2.2. 본 코드 컴파일 : 행렬 연산에 ESSL을 사용

(Thanks to 서울대학교 물리천문학부 김영국 박사님)

VASP 컴파일과 실행에서 가장 중요한 라이브러리는 해밀토니안 행렬 연산에 사용되는 BLAS (Basic Linear Algebra Subroutines, http://www.netlib.org/blas/) 와 LAPACK (Linear Algebra PACKage, http://www.netlib.org/lapack/) 입니다. BLAS는 두 행렬 사이의 기본적인 연산이, LAPACK은 BLAS에서 정의된 함수들을 이용하여 대각화나 행렬식 등을 계산하는 함수들이 정의되어 있는 라이브러리들로서, BLAS와 LAPACK을 각 시스템에 맞게 최적화된 패키지를 사용함으로서 코드의 성능을 향상시킬 수 있습니다. Gaia 1,2차 시스템에는, IBM 시스템에 맞게 최적화되어 있는 ESSL (Engineering and Scientific Subroutine Library, http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=%2Fcom.ibm.cluster.essl.doc%2Fesslbooks.html) 이 설치되어 있습니다.

ESSL에는 LAPACK에서 정의된 함수들이 일부 포함되어 있는데, 두 라이브러리에서 정의된 함수의 형태가 같지 않아 서로 충돌하는 경우가 발생합니다. 이러한 충돌을 막기 위해, VASP을 Gaia 시스템에서 컴파일 할 때는 항상 LAPACK보다 ESSL을 먼저 링크할 것을 권장하고 있습니다. 다만, 현재 버전(5.3.3)의 spinsym.F에서 호출되는 DGEEV와 ZGEEV 서브루틴 같은 경우에는, 라이브러리 순서를 맞추는 것만으로는 라이브러리 사이의 충돌을 막을 수 없습니다. 따라서, 이러한 충돌을 막으려면 spinsym.F 안에서 호출되는 DGEEV와 ZGEEV 함수를 LAPACK에서 정의된 것과 같은 형태로 새로 정의하여 호출해야 합니다. 따로 첨부하는 spinsym.F를 source 경로에 붙여 넣고 컴파일하면 됩니다.

두 시스템에 대한 Makefile 파일은 부록에 덧붙이며, 여기에서는 Makefile에서 시스템별/포함되는 라이브러리 별 차이가 있는 부분만을 설명하도록 합니다.

< Gaia 1차 >

```
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr5\" -DMPI -Duse_collective \
      -Dessl  -DCACHE_SIZE=32768  -DPGF90 \
      -DUSE_ZHEEVX \
      -Davoidalloc \
      -DNGZhalf \
      $*.F >$*$(SUFFIX)

LAPACK  =  /applic/local/lib/liblapack-PWR5.a

LIB = -L/usr/lib -lessl \
       -L../vasp.5.lib -ldmy  \
       ../vasp.5.lib/linpack_double.o \
       $(MPI) $(LAPACK) -lm \
       /applic/local/lib/libfftw3.a
```

< Gaia 2차 >

```
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr6\" -DMPI -Duse_collective \
      -Dessl  -DCACHE_SIZE=32768  -DPGF90 \
      -DUSE_ZHEEVX \
      -Davoidalloc \
      -DNGZhalf \
      $*.F >$*$(SUFFIX)

LAPACK  =  /applic/local/64BIT/LAPACK/V342/lib/liblapack.a

LIB = -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lessl \
       -L../vasp.5.lib -ldmy  \
       ../vasp.5.lib/linpack_double.o \
       $(MPI) $(LAPACK) -lm \
       /applic/local/64BIT/FFTW/V322/lib/libfftw3.a
```

Spin-orbit coupling을 포함하여 계산하려 할 경우에는 CPP의 -DNGZhalf 옵션을 제거 후 컴파일해야 합니다. 자세한 것은 vasp 매뉴얼을 참조하시기 바랍니다.

2.3. 본 코드 컴파일 : 행렬 연산에 BLAS를 사용

앞서 설명한 것처럼, VASP에서 ESSL과 LAPACK을 사용하여 컴파일 할 때는, 두 라이브러리 사이의 충돌 때문에 컴파일이 안되며, 코드 상에서 충돌을 일으키는 부분을 수정해야 합니다. 코드를 건드리지 않고 컴파일하기 위해서, ESSL 대신 BLAS를 사용할 수 있습니다. 대신, Gaia 시스템에 최적화된 ESSL 라이브러리를 사용하지 않기 때문에 ESSL을 사용할 때에 비해서 성능 저하가 있을 수 있습니다.

< Gaia 1차 >

```
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr5\" -DMPI -Duse_collective \
      -DCACHE_SIZE=32768   -DPGF90 \
      -Davoidalloc \
      -DUSE_ZHEEVX \
      -DNGZhalf \
      $*.F >$*$(SUFFIX)

LAPACK = /applic/local/lib/liblapack-PWR5.a

LIB = -L/usr/lib -lblas \
      -L../vasp.5.lib -ldmy  \
      ../vasp.5.lib/linpack_double.o \
      $(MPI) $(LAPACK) \
      -L/usr/lib -lessl \
      /applic/local/lib/libfftw3.a
```

Gaia 1차 시스템에서는 LAPACK 라이브러리가 ESSL을 이용하여 컴파일 되어 있기 때문에, 추가로 ESSL을 링크해주어야 합니다. 이 때, BLAS와 ESSL의 충돌을 막기 위해서, ESSL은 BLAS와 LAPACK 뒤에 링크되어야 합니다.

< Gaia 2차 >

```
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr6\" -DMPI -Duse_collective \
      -DCACHE_SIZE=32768   -DPGF90 \
      -Davoidalloc \
      -DNGZhalf \
      $*.F >$*$(SUFFIX)

LAPACK = /applic/local/64BIT/LAPACK/V342/lib/liblapack.a

LIB = -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lblas \
       -L../vasp.5.lib -ldmy  \
       ../vasp.5.lib/linpack_double.o \
       $(MPI) $(LAPACK) \
       -L/applic/local/64BIT/FFTW/V322/lib -lfftw3
```

2.4. 본 코드 컴파일 : 행렬 연산에 BLACS/ScaLAPACK을 사용

표면/분자 상태처럼 계산하고자 하는 물리계의 크기가 커질 경우, 전체 해밀토니안 행렬의 크기 역시 그에 비례하여 커지게 됩니다. 거대한 크기의 행렬을 MPI를 통해 많은 수의 CPU를 사용하여 다루기 위한 라이브러리로서는 BLACS (Basic Linear Algebra Communication Subprograms, http://www.netlib.org/blacs/) 와 ScaLAPACK (Scalable Linear Algebra PACKage, http://www.netlib.org/scalapack/index.html) 이 있고, VASP을 이들을 사용하여 컴파일하여 사용할 수 있습니다. VASP에서 ScaLAPACK은 계산하고자 하는 subspace 의 해밀토니안 행렬(NBANDS-차원의 정사각 행렬)의 LU decomposition 및 대각화에 사용되며, 큰 규모의 물리계(~103개의 원자)를 많은 수의 CPU(~1000 코어)를 사용하여 계산할 때에 ScaLAPACK을 사용하여 보다 개선된 병렬화 효율을 얻을 수 있을 것이라 기대할 수 있지만, 그보다 작은 크기의 물리계를 다루거나 또는 작은 수의 CPU를 사용하는 경우는 오히려 추가적인 MPI 통신에 따른 속도 저하를 겪을 수 있습니다. (다음 링크된 문서의 Figure 9,10을 참조하시기 바랍니다.- http://www.hector.ac.uk/cse/distributedcse/reports/vasp01/vasp01_collectives/index.htm) 따라서, Gaia system에서는 이들을 사용하는 것을 권장하지 않습니다만, 참고 목적으로 ScaLAPACK을 사용하여 VASP을 컴파일하기를 원하시면 Makefile을 다음과 같이 변경하시면 됩니다. 본 문서에서는 Gaia 2차 시스템에서의 컴파일만을 설명하도록 합니다.

< Gaia 2차 >

```
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr6\" -DMPI -Duse_collective \
      -DCACHE_SIZE=32768   -DPGF90 \
      -Davoidalloc \
      -DUSE_ZHEEVX \
      -DNGZhalf \
      -DscaLAPACK \
      $*.F >$*$(SUFFIX)

LAPACK = /applic/local/64BIT/LAPACK/V342/lib/liblapack.a
SCA    = /applic/local/64BIT/SCALAPACK/V180/lib/libscalapack.a
BLACS  = /applic/local/64BIT/BLACS/V11_PATCH03/lib/blacsF77init_MPI-PWR6-0.a \
           /applic/local/64BIT/BLACS/V11_PATCH03/lib/blacs_MPI-PWR6-0.a\
       /applic/local/64BIT/BLACS/V11_PATCH03/lib/blacsF77init_MPI-PWR6-0.a

LIB    = -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lpessl -lessl \
         -L../vasp.5.lib -ldmy  \
         ../vasp.5.lib/linpack_double.o \
         $(MPI) $(LAPACK) $(SCA) $(BLACS) -lessl \
         -L/applic/local/64BIT/FFTW/V322/lib -lfftw3
```

3. VASP 설치 (II) - KISTI 슈퍼컴퓨터 4호기 Tachyon 시스템

3.1. VASP 기본 라이브러리 컴파일

VASP 본 코드를 컴파일하기 위해서는, Gaia 시스템과 마찬가지로 VASP에서 공통적으로 사용되는 라이브러리들을 컴파일해야 합니다. VASP 소스 파일의 압축을 풀면, 다음과 같은 2개의 디렉토리가 만들어집니다.

```
./vasp.5.3
./vasp.5.lib
```

먼저 vasp.5.lib 디렉토리로 들어가서, makefile 파일을 생성합니다. 몇 가지 시스템들에 대한 예제 makefile이 있기 때문에, 이 중 하나를 makefile로 복사한 후에, 컴파일러 관련 옵션을 다음과 같이 고치면 됩니다.

```
# C-preprocessor
CPP     = gcc -E -P -C $*.F >$*.f
FC      = ifort

CFLAGS = -O
FFLAGS = -O0 -FI
FREE   = -FR
```

컴파일이 성공적으로 마무리되면 Gaia 시스템과 동일하게 디렉토리에 libdmy.a 파일이 만들어지고, VASP 본 코드를 컴파일 할 때 이 라이브러리를 불러오게 됩니다.

3.2. 본 코드 컴파일 - BLAS/LAPACK 사용

Gaia 시스템에서의 컴파일에서 기술하였듯이 VASP 컴파일과 실행에서 가장 중요한 라이브러리는 해밀토니안 행렬 연산에 사용되는 BLAS (Basic Linear Algebra Subroutines, http://www.netlib.org/blas/) 와 LAPACK (Linear Algebra PACKage, http://www.netlib.org/lapack/) 입니다. BLAS는 두 행렬 사이의 기본적인 연산이, LAPACK은 BLAS에서 정의된 함수들을 이용하여 대각화나 행렬식 등을 계산하는 함수들이 정의되어 있는 라이브러리들로서, BLAS와 LAPACK을 각 시스템에 맞게 최적화된 패키지를 사용함으로서 코드의 성능을 향상시킬 수 있습니다. Tachyon 1,2차 시스템에는 이러한 BLAS와 LAPACK 라이브러리 뿐 아니라, Intel MKL (Math Kernel Library, http://software.intel.com/en-us/intel-mkl)과 AMD ACML (AMD Core Math Library, http://developer.amd.com/tools-and-sdks/cpu-development/amd-core-math-library-acml/) 등의 각 CPU 아키텍처에 맞게 최적화된 BLAS/LAPACK 라이브러리들이 설치되어 있습니다.

다만, Tachyon 1,2차 시스템에서 기본 컴파일러로 설정되어 있는 Intel composerxe 2013 버전을 사용해 컴파일 할 때, 2013년 버전으로 컴파일된 수치해석 라이브러리 중 문제가 없이 컴파일이 완료되는 경우는 LAPACK/BLAS 또는 ACML을 사용했을 때 뿐이고, 나머지 MKL이나 GotoBLAS를 사용하여 컴파일 할 경우는 컴파일은 문제없이 완료되나 실제 코드를 실행할 때 Segmentation fault 오류를 내며 코드가 실행 중지됩니다. 그러므로 본 문서에서는 Intel composerxe 2013년 버전과 LAPACK/BLAS와 ACML을 사용하여 컴파일하는 경우에 대해 설명하도록 합니다. Intel compiler 11.1 버전을 이용해 컴파일 했을 때에도 같은 문제가 발생하는 것으로 보아 현재 컴파일러 환경에 문제가 있을 것으로 판단되며, Intel 컴파일러와 Intel MKL을 이용하여 컴파일 및 코드 실행까지 성공하신 분은 관리자에게 알려주시면 추후 지침서 업데이트시 추가하도록 하겠습니다.

Intel compiler와 MPI는 module 명령어를 통해서 원하는 버전으로 설정할 수 있습니다. 아래 예제에서 사용한 Intel 2013 버전과 OpenMPI 1.4.2 이외에 다른 컴파일러와 MPI를 사용해야 할 경우에는 먼저 module 명령어를 통해서 이를 변경한 후에 아래 컴파일러 경로 및 라이브러리 경로를 module 환경과 동일하게 수정하면 됩니다. (http://helpdesk.ksc.re.kr/4th/resource/hw/ 에 있는 TACHYON I,II 사용자 지침서를 참조)

< Tachyon 1차 >

```
FC       = mpif90 -I/applic/lib.intel/FFTW3/include
FCL      = $(FC)

CPP_     =  ./preprocess <$*.F | /usr/bin/cpp -P -C -traditional >$*$(SUFFIX)
CPP      = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
           -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
           -DMPI_BLOCK=8000 -Duse_collective

FFLAGS = -FR -lowercase -assume byterecl -heap-arrays
LAPACK = -L/applic/lib.intel/LAPACK -lblas -llapack
LIB      = -L../vasp.5.lib -ldmy \
             ../vasp.5.lib/linpack_double.o $(LAPACK) -lm \

FFT3D   = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o   \
            /applic/lib.intel/FFTW3/lib/libfftw3.a
```

ACML을 이용하여 컴파일 할 때는, 위의 LAPACK 부분을 다음과 같이 수정하면 됩니다.

```
LAPACK = -L/applic/lib.intel/ACML5/ifort64 -lacml
```

< Tachyon 2차 >

Tachyon 2차는 컴파일러/라이브러리 설치 경로 차이 때문에 달라지는 부분만을 설명합니다.

```
FC=mpif90 -I/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/include

LAPACK = -L/applic/compilers/intel/2013/applib1/LAPACK -lblas -llapack

FFT3D   = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o \
/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/lib/libfftw3.a
```

Tachyon 1차 시스템의 경우와 마찬가지로, ACML을 이용하기 위해서는 LAPACK 옵션을 다음과 같이 수정하면 됩니다.

```
LAPACK   = -L/applic/compilers/intel/2013/applib1/ACML/ifort64/lib -lacml
```

Gaia 시스템과 마찬가지로 Spin-orbit coupling을 포함하여 계산하려 할 경우에는 CPP의 -DNGZhalf 옵션을 제거 후 컴파일해야 합니다. 자세한 것은 vasp 매뉴얼을 참조하시기 바랍니다.

3.3. 본 코드 컴파일 - BLACS/ScaLAPACK 사용

표면/분자 상태처럼 계산하고자 하는 물리계의 크기가 커질 경우, 전체 해밀토니안 행렬의 크기 역시 그에 비례하여 커지게 됩니다. 거대한 크기의 행렬을 MPI를 통해 많은 수의 CPU를 사용하여 다루기 위한 라이브러리로서는 BLACS (Basic Linear Algebra Communication Subprograms, http://www.netlib.org/blacs/) 와 ScaLAPACK(Scalable Linear Algebra PACKage, http://www.netlib.org/scalapack/index.html) 이 있고, VASP을 이들을 사용하여 컴파일 하여 사용할 수 있습니다. VASP에서 ScaLAPACK은 계산하고자 하는 subspace 의 해밀토니안 행렬(NBANDS-차원의 정사각 행렬)의 LU decomposition 및 대각화에 사용되며, 큰 규모의 물리계(~103개의 원자)를 많은 수의 CPU(~1000 코어)를 사용하여 계산할 때에 ScaLAPACK을 사용하여 보다 개선된 병렬화 효율을 얻을 수 있을 것이라 기대할 수 있지만, 그보다 작은 크기의 물리계를 다루거나 또는 작은 수의 CPU를 사용하는 경우는 오히려 추가적인 MPI 통신에 따른 속도 저하를 겪을 수 있습니다. (다음 링크된 문서의 Figure 9,10을 참조하시기 바랍니다.- http://www.hector.ac.uk/cse/distributedcse/reports/vasp01/vasp01_collectives/index.htm) 따라서, Tachyon system에서는 Gaia 시스템과 마찬가지로 이들을 사용하는 것을 권장하지 않습니다만, 참고 목적으로 ScaLAPACK을 사용하여 VASP을 컴파일하기를 원하시면 Makefile의 다음 변수들을 다음과 같이 수정하시면 됩니다. (본 문서에서는 Tachyon 2차 시스템에서의 컴파일 방법을 설명하도록 하겠습니다.)

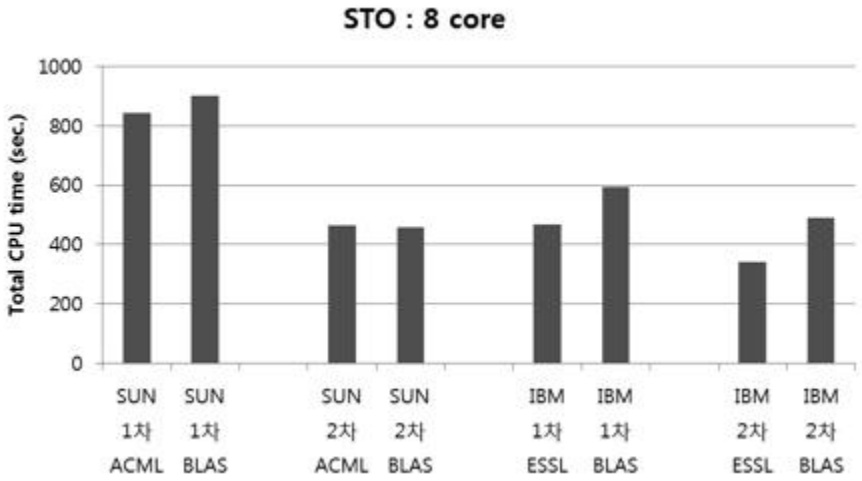
```
CPP     = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
          -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
          -DMPI_BLOCK=8000 -Duse_collective -DscaLAPACK

INTEL   = /applic/compilers/intel/2013
LAPACK = $(INTEL)/applib1/ACML/ifort64/lib/libacml.a
SCA     = $(INTEL)/mpi/openmpi/1.4.2/applib2/SCALAPACK/libscalapack.a
BLACS   = $(INTEL)/mpi/openmpi/1.4.2/applib2/BLACS/libmpiblacsF77init.a \
           $(INTEL)/mpi/openmpi/1.4.2/applib2/BLACS/libmpiblacs.a \
          $(INTEL)/mpi/openmpi/1.4.2/applib2/BLACS/libmpiblacsF77init.a

LIB     = -L../vasp.5.lib -ldmy \
          ../vasp.5.lib/linpack_double.o $(LAPACK) $(SCA) $(BLACS) $(LAPACK) -lm
```

4. VASP 성능 테스트

성능 테스트는 총 3가지 경우에 대해서 이루어졌으며, 마지막의 BLACS / ScaLAPACK을 포함한 경우는 Gaia 2차와 Tachyon 2차 시스템에서만 테스트를 진행했습니다. 각 물질에 대해서 서로 다른 시스템과 수치해석 라이브러리를 이용해서 계산된 결과들이 모두 1.0^{-9} eV의 오차 범위 안에서 같은 것을 확인하였습니다.

4.1. 8 core test - Cubic $SrTiO_3$

- 24X24X24 on Monkhorst-Pack grid, energy cutff 600 eV, LSORBIT off

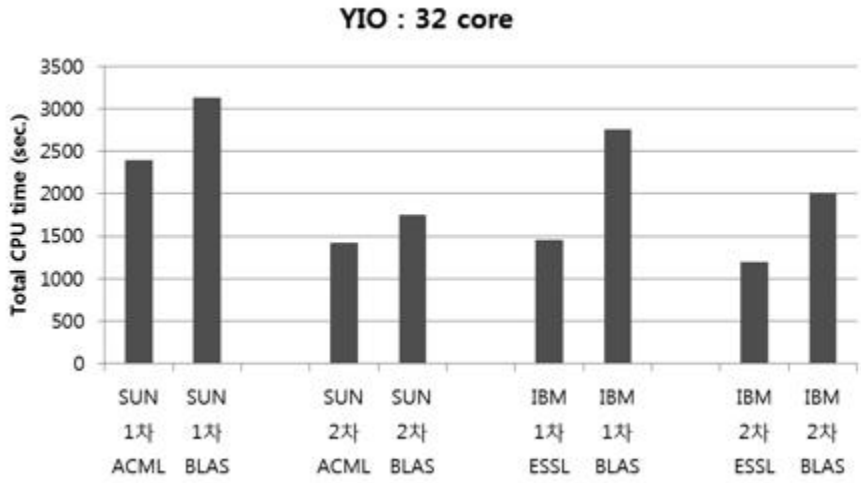


Tachyon 시스템의 경우, 1차의 경우 ACML을 이용한 결과가 근소하게 성능이 앞서며, 2차 시스템에서는 아주 작은 차이로 BLAS를 사용한 계산이 우세를 보입니다. 이는 ACML 라이브러리가 Tachyon 1차에 사용된 AMD cpu에 최적화 되었기 때문으로 보입니다. 전체적으로 2차 시스템이 1차에 비해서 같은 코어 수 대비 2배에 가까운 성능 향상을 보입니다.

Tachyon 시스템에서와 달리, Gaia 시스템에서는 ESSL을 사용했을 때 BLAS 대비 30~40% 내외의 두드러진 성능 향상을 보이며, 따라서 효율적인 계산을 위해서는 ESSL을 사용하는 것이 유리합니다.

4.2. 32 core test - Pyrochlore $Y_2Ir_2O_7$

- 8X8X8 on Monkhorst-Pack grid, energy cutoff 400 eV, LSORBIT off



BLAS와 ACML의 성능 차가 크지 않았던 8코어에서의 결과와는 달리, 계산의 크기가 커지는 문제에서는 ACML과 BLAS의 성능 차가 커집니다. AMD cpu를 사용한 Tachyon 1차에서 ACML과 BLAS 의 성능 차이가 두드러지며, Intel cpu를 사용한 Tachyon 2차 시스템에서도 ACML을 사용했을 때 15% 정도의 성능 향상이 보이며, 따라서 보다 Intel cpu에 최적화된 MKL 라이브러리를 사용할 경우 더 큰 성능 향상을 기대할 수 있습니다. 8 core 계산의 경우와 마찬가지로, Tachyon 2차 시스템의 경우 1차 시스템에 비해 같은 코어 수 대비 2배에 가까운 성능 향상을 보입니다.

Gaia 시스템의 경우, ESSL을 사용했을 때의 이득이 작은 시스템을 계산할 때 보다 더욱 커지는 것을 볼 수 있으며, Tachyon 시스템에서도 볼 수 있는 것처럼 대규모의 계산을 수행할 때 각 시스템에 최적화된 수치해석 라이브러리를 사용하는 것이 효율적인 계산을 위해서 중요한 것을 알 수 있습니다.

4.3. 64, 256 core test with BLACS/ScaLAPACK – $SrTiO_3$ N^3 supercell (N=2,3,4)

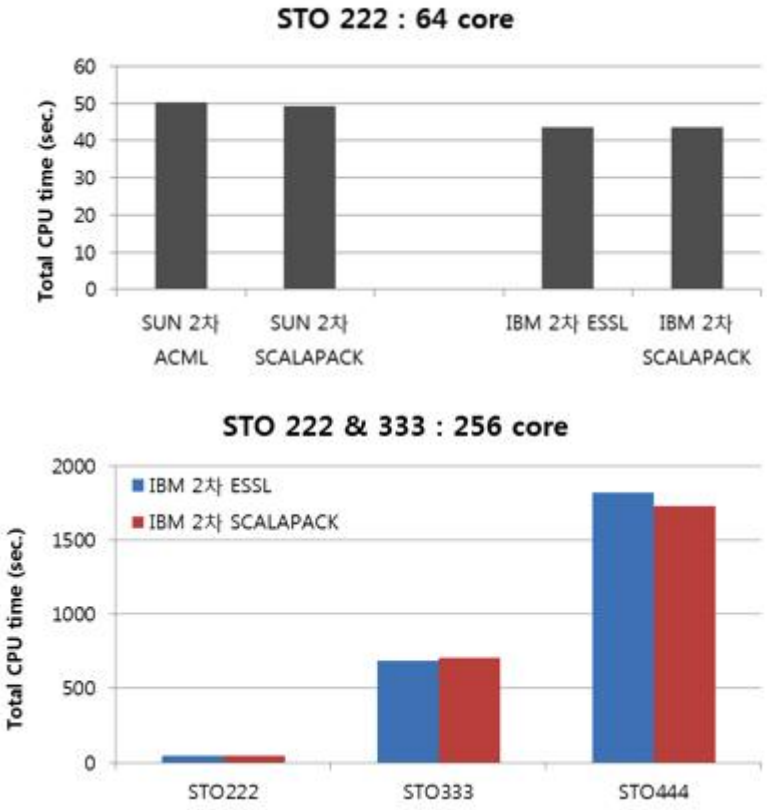
계산하고자 하는 물질의 크기에 따른 성능의 차이를 비교하기 위해서, 3가지의 서로 다른 크기를 가지는 물질을 계산한 결과입니다. 각 경우에 사용한 unit cell과 CPU 수, 그리고 계산에 사용된 변수 중 차이가 나는 부분은 다음과 같으며, 모든 계산에서 처음 12번의 DAV iteration만을 거친 시간을 측정했습니다.

I) $SrTiO_3$(STO) 2^3 supercell (원자 40개) : 64 core, KPOINTS는 Gamma-point만, NBANDS=256, ENCUT=400eV

ii) $SrTiO_3$(STO) 3^3 supercell (원자 135개) : 256 core, 2*2*2 Monkhorst-Pack grid, NBANDS=640, ENCUT=400eV

iii) $SrTiO_3$(STO) 4^3 supercell (원자 320개) : 256 core, Gamma-point, NBAND=1536, ENCUT=400eV

STO 3^3과 4^3 supercell은 Tachyon 2차 시스템에서는 테스트할 수 없었는데, 이는 INCAR의 NBANDS (VASP이 다루는 subspace의 basis 수)가 256을 넘게 되면 계산이 실행되지 않기 때문입니다. 따라서, ScaLAPACK을 사용한 계산을 실행하여야 할 경우는 Gaia 시스템을 사용할 것을 권장합니다.



결과에서 볼 수 있는 것처럼, 원자 수 ~400개, 256개의 CPU 수 까지는 ScaLAPACK을

사용하는 이득이 크지 않은 것을 볼 수 있습니다. 보다 많은 수의 CPU를 사용하는 경우는 ScaLAPACK을 사용한 경우에 병렬화 효율이 개선될 것이라 기대할 수 있지만, 그렇지 않은 경우에는 ScaLAPACK을 사용하지 않는 것을 권장합니다.

< 부록 >

A. Makefile for vasp 5.lib

A.1 Makefile for vasp 5.lib in Gaia1 and Gaia2

```
.SUFFIXES: .inc .f .F
#-----------------------------------------------------------------------
# Makefile for IBM GAIA2
#-----------------------------------------------------------------------
# C-preprocessor
CPP        = /usr/ccs/lib/cpp -P -C     $*.F >$*.f
FC=xlf
CFLAGS = -O
FFLAGS = -O3 -qarch=auto -qtune=auto -qstrict
FREE     = -qfree=f90
DOBJ =   preclib.o timing.o derrf.o dclock.o   diolib.o dlexlib.o drdatab.o
#-----------------------------------------------------------------------
# general rules
#-----------------------------------------------------------------------
libdmy.a: $(DOBJ) lapack_double.o linpack_double.o
        -rm libdmy.a
        ar vq libdmy.a $(DOBJ)
# files which do not require autodouble
lapack_double.o: lapack_double.f
        $(FC) $(FFLAGS) $(NOFREE) -c lapack_double.f
lapack_single.o: lapack_single.f
        $(FC) $(FFLAGS) $(NOFREE) -c lapack_single.f
linpack_double.o: linpack_double.f
        $(FC) $(FFLAGS) $(NOFREE) -c linpack_double.f
linpack_single.o: linpack_single.f
        $(FC) $(FFLAGS) $(NOFREE) -c linpack_single.f
.c.o:
        $(CC) $(CFLAGS) -c $*.c
.F.o:
        $(CPP)
        $(FC) $(FFLAGS) $(FREE) $(INCS) -c $*.f
.F.f:
        $(CPP)
.f.o:
        $(FC) $(FFLAGS) $(FREE) $(INCS) -c $*.f
```

A.2. Makefile for vasp 5.lib in Tachyon 1 and Tachyon 2

```
.SUFFIXES: .inc .f .F
#-----------------------------------------------------------------------
# Makefile for intel fortran compiler
# the makefile was tested only under Linux on Intel platforms
# however it might work on other platforms as well
## Mind: one user reported that he had to copy preclib.F diolib.F
#   dlexlib.F and drdatab.F to the directory vasp.4.4, compile the files
#   there and link them directly  into vasp
#   for no obvious reason these files could not be linked from the library
##-----------------------------------------------------------------------
# C-preprocessor
CPP        = gcc -E -P -C $*.F >$*.f
FC         = ifort
CFLAGS = -O
FFLAGS = -O0 -FI
FREE    =  -FR
DOBJ =   preclib.o timing_.o derrf_.o dclock_.o   diolib.o dlexlib.o drdatab.o
#-----------------------------------------------------------------------
# general rules
#-----------------------------------------------------------------------
libdmy.a: $(DOBJ) lapack_double.o linpack_double.o lapack_atlas.o
        -rm libdmy.a
        ar vq libdmy.a $(DOBJ)
# files which do not require autodouble
lapack_min.o: lapack_min.f
        $(FC) $(FFLAGS) $(NOFREE) -c lapack_min.f
lapack_double.o: lapack_double.f
        $(FC) $(FFLAGS) $(NOFREE) -c lapack_double.f
lapack_single.o: lapack_single.f
        $(FC) $(FFLAGS) $(NOFREE) -c lapack_single.f
lapack_atlas.o: lapack_atlas.f
        $(FC) $(FFLAGS) $(NOFREE) -c lapack_atlas.f
linpack_double.o: linpack_double.f
        $(FC) $(FFLAGS) $(NOFREE) -c linpack_double.f
linpack_single.o: linpack_single.f
        $(FC) $(FFLAGS) $(NOFREE) -c linpack_single.f
.c.o:
        $(CC) $(CFLAGS) -c $*.c
.F.o:
        $(CPP)
        $(FC) $(FFLAGS) $(FREE) $(INCS) -c $*.f
.F.f:
        $(CPP)
.f.o:
        $(FC) $(FFLAGS) $(FREE) $(INCS) -c $*.f
```

B. Makefile for Gaia

B.1. Makefile for vasp5.3 in gaia1 with ESSL

```
.SUFFIXES: .inc .f .F
SUFFIX=.f
FC =mpxlf90 -q64 -qfree=f90
F77=mpxlf
FCL=$(FC)
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr5\" -DMPI -Duse_collective \
           -Dessl -DCACHE_SIZE=32768 -DPGF90 \
           -DUSE_ZHEEVX \
           -Davoidalloc \
           -DNGZhalf \
           $*.F >$*$(SUFFIX)
OFLAG_0 = -O0
OFLAG_1 = -O
OFLAG_2 = -O2
OFLAG_3 = -O3 -qstrict
OFLAG_4 = -O4 -qstrict -qhot
DEBUG = -g -qfullpath
INCS =
OFLAG = $(OFLAG_2)
INLINE = $(OFLAG)
FFLAGS = -qmaxmem=-1 -qarch=auto -qtune=auto -qcache=auto
OFLAG = -O3 -qarch=auto -qstrict
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH = none
OBJ_NOOPT = none
DEBUG = -g -qfullpath
INLINE = $(OFLAG)
LINK = -q64 -bdynamic
MPI = -L/usr/lpp/ppe.poe/lib -lmpi
LAPACK = /applic/local/lib/liblapack-PWR5.a
LIB = -L/usr/lib -lessl \
         -L../vasp.5.lib -ldmy \
          ../vasp.5.lib/linpack_double.o \
          $(MPI) $(LAPACK) -lm \
          /applic/local/lib/libfftw3.a
FFT3D = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
BASIC= symmetry.o symlib.o lattlib.o random.o
SOURCE= base.o mpi.o smart_allocate.o xml.o \
           constant.o jacobi.o main_mpi.o scala.o \
           asa.o lattice.o poscar.o ini.o mgrid.o xclib.o vdw_nl.o xclib_grad.o \
           radial.o pseudo.o gridq.o ebs.o \
           mkpoints.o wave.o wave_mpi.o wave_high.o spinsym.o\
           $(BASIC) nonl.o nonlr.o nonl_high.o dfast.o choleski2.o \
           mix.o hamil.o xcgrad.o xcspin.o potex1.o potex2.o \
           constrmag.o cl_shift.o relativistic.o LDApU.o \
           paw_base.o metagga.o egrad.o pawsym.o pawfock.o pawlhf.o rhfatm.o hyperfine.o paw.o \
           mkpoints_full.o charge.o Lebedev-Laikov.o stockholder.o dipol.o pot.o \
           dos.o elf.o tet.o tetweight.o hamil_rot.o \
           chain.o dyna.o k-proj.o sphpro.o us.o core_rel.o \
           aedens.o wavpre.o wavpre_noio.o broyden.o \
           dynbr.o hamil_high.o rmm-diis.o reader.o writer.o tutor.o xml_writer.o \
           brent.o stufak.o fileio.o opergrid.o stepver.o \
           chgloc.o fast_aug.o fock_multipole.o fock.o mkpoints_change.o sym_grad.o \
           mymath.o internals.o npt_dynamics.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
           nmr.o pead.o subrot.o subrot_scf.o \
           force.o pwlhf.o gw_model.o optreal.o steep.o davidson.o david_inner.o \
           electron.o rot.o electron_all.o shm.o pardens.o paircorrection.o \
           optics.o constr_cell_relax.o stm.o finite_diff.o elpol.o \
           hamil_lr.o rmm-diis_lr.o subrot_cluster.o subrot_lr.o \
```

```
             lr_helper.o hamil_lrf.o     elinear_response.o ilinear_response.o \
             linear_optics.o \
             setlocalpp.o   wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
             mlwf.o        ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
             local_field.o ump2.o ump2kpar.o fcidump.o ump2no.o \
             bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
             rmm-diis_mlr.o   linear_response_NMR.o wannier_interpol.o linear_response.o ztrevc.o zlatrs.o
vasp: $(SOURCE) $(FFT3D) $(INC) main.o
        rm -f vasp.real
        $(FCL) -o vasp.real   main.o   $(SOURCE)    $(FFT3D) $(LIB) $(LINK)
makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
        $(FCL) -o makeparam   $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
zgemmtest: zgemmtest.o base.o random.o $(INC)
        $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
dgemmtest: dgemmtest.o base.o random.o $(INC)
        $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
        $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
        $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
clean:
        -rm -f *.g *.f *.o *.L *.mod ; touch *.F
main.o: main$(SUFFIX)
        $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c main$(SUFFIX)
xcgrad.o: xcgrad$(SUFFIX)
        $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
xcspin.o: xcspin$(SUFFIX)
        $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
makeparam.o: makeparam$(SUFFIX)
        $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
makeparam$(SUFFIX): makeparam.F main.F
base.o: base.inc base.F
mgrid.o: mgrid.inc mgrid.F
constant.o: constant.inc constant.F
lattice.o: lattice.inc lattice.F
setex.o: setexm.inc setex.F
pseudo.o: pseudo.inc pseudo.F
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
        $(CPP)
        $(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
        $(CPP)
        $(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
        $(CPP)
        $(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
        $(CPP)
        $(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
        $(CPP)
$(SUFFIX).o:
        $(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
fft3dfurth.o : fft3dfurth.F
        $(CPP)
```

```
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symlib.o : symlib.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symmetry.o : symmetry.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dynbr.o : dynbr.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
broyden.o : broyden.F
	$(CPP)
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
wave.o : wave.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
fftmpi_map.o : fftmpi_map.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
stockholder.o : stockholder.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
paircorrection.o : paircorrection.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
tet.o : tet.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dos.o : dos.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pseudo.o : pseudo.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
 paw.o : paw.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pot.o : pot.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
sydmat.o : sydmat.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
elinear_response.o : elinear_response.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
chi.o : chi.F
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -qalias_size=500000000 -c $*$(SUFFIX)
```

Makefile for vasp5.3 in Gaia2 with ESSL

```
.SUFFIXES: .inc .f .F
SUFFIX=.f
FC =mpxlf90 -q64 -qfree=f90
F77=mpxlf
FCL=$(FC)
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr6\" -DMPI -Duse_collective \
          -Dessl   -DCACHE_SIZE=32768   -DPGF90 \
          -DUSE_ZHEEVX \
          -Davoidalloc \
          -DNGZhalf \
          $*.F >$*$(SUFFIX)
OFLAG_0 = -O0
OFLAG_1 = -O
OFLAG_2 = -O2
OFLAG_3 = -O3 -qstrict
OFLAG_4 = -O4 -qstrict -qhot
DEBUG    =  -g -qfullpath
INCS     =
OFLAG    = $(OFLAG_2)
INLINE   = $(OFLAG)
#FFLAGS = -qmaxmem=-1 -qarch=auto -qtune=auto -qcache=auto
FFLAGS = -qmaxmem=-1 -qarch=pwr6 -qtune=pwr6
OFLAG  = -O3 -qarch=auto -qstrict
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH = none
OBJ_NOOPT = none
DEBUG  = -g -qfullpath
INLINE = $(OFLAG)
#LINK     = -q64 -bdynamic -bnoquiet
LINK     = -q64 -bdynamic
MPI      = -L/usr/lpp/ppe.poe/lib -lmpi
LAPACK   =  /applic/local/64BIT/LAPACK/V342/lib/liblapack.a
#LAPACK   = ../vasp.5.lib/lapack_double.o /home01/g062kyk/programs/espresso-4.1/flib/lapack.a
LIB      = \
                -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lessl \
                        -L../vasp.5.lib -ldmy  \
                        ../vasp.5.lib/linpack_double.o \
                $(MPI) $(LAPACK) -lm \
                /applic/local/64BIT/FFTW/V322/lib/libfftw3.a
#LIB      = \
#               -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lessl \
#                       -L../vasp.5.lib -ldmy  \
#                       ../vasp.5.lib/linpack_double.o \
#             /home01/g062kyk/programs/espresso-4.1/flib/lapack.a \
#               $(MPI) \
#               /applic/local/64BIT/FFTW/V322/lib/libfftw3.a
#FFT3D  = fftmpi.o fftmpi_map.o fft3dessl+furth.o fft3dlib.o
FFT3D  = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
```

```
BASIC=  symmetry.o symlib.o  lattlib.o  random.o
SOURCE=  base.o  mpi.o  smart_allocate.o  xml.o  \
        constant.o jacobi.o  main_mpi.o  scala.o  \
        asa.o  lattice.o  poscar.o  ini.o  mgrid.o  xclib.o  vdw_nl.o  xclib_grad.o \
        radial.o  pseudo.o  gridq.o  ebs.o  \
        mkpoints.o wave.o  wave_mpi.o  wave_high.o spinsym.o\
        $(BASIC)  nonl.o  nonlr.o  nonl_high.o dfast.o  choleski2.o \
        mix.o  hamil.o  xcgrad.o  xcspin.o  potex1.o  potex2.o  \
        constrmag.o cl_shift.o relativistic.o LDApU.o \
        paw_base.o metagga.o  egrad.o  pawsym.o  pawfock.o  pawlhf.o  rhfatm.o  hyperfine.o
paw.o  \
        mkpoints_full.o  charge.o  Lebedev-Laikov.o  stockholder.o dipol.o  pot.o \
        dos.o  elf.o  tet.o  tetweight.o hamil_rot.o \
        chain.o  dyna.o  k-proj.o  sphpro.o  us.o  core_rel.o \
        aedens.o  wavpre.o  wavpre_noio.o broyden.o \
        dynbr.o  hamil_high.o  rmm-diis.o reader.o  writer.o  tutor.o xml_writer.o \
        brent.o  stufak.o  fileio.o  opergrid.o stepver.o  \
        chgloc.o  fast_aug.o fock_multipole.o  fock.o  mkpoints_change.o sym_grad.o \
        mymath.o  internals.o  npt_dynamics.o  dynconstr.o  dimer_heyden.o  dvvtrajectory.o
vdwforcefield.o \
        nmr.o  pead.o  subrot.o  subrot_scf.o \
        force.o  pwlhf.o  gw_model.o optreal.o  steep.o  davidson.o  david_inner.o \
        electron.o rot.o  electron_all.o shm.o  pardens.o  paircorrection.o \
        optics.o  constr_cell_relax.o  stm.o  finite_diff.o elpol.o  \
        hamil_lr.o rmm-diis_lr.o  subrot_cluster.o subrot_lr.o \
        lr_helper.o hamil_lrf.o  elinear_response.o ilinear_response.o \
        linear_optics.o \
        setlocalpp.o  wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
        mlwf.o  ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
        local_field.o ump2.o ump2kpar.o fcidump.o ump2no.o \
        bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
        rmm-diis_mlr.o  linear_response_NMR.o wannier_interpol.o linear_response.o ztrevc.o zlatrs.o
vasp: $(SOURCE) $(FFT3D) $(INC) main.o
        rm -f vasp.real
        $(FCL) -o vasp.real  main.o  $(SOURCE)  $(FFT3D) $(LIB) $(LINK)
makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
        $(FCL) -o makeparam  $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
zgemmtest: zgemmtest.o base.o random.o $(INC)
        $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
dgemmtest: dgemmtest.o base.o random.o $(INC)
        $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
        $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
        $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
clean:
        -rm -f *.g *.f *.o *.L *.mod ; touch *.F
main.o: main$(SUFFIX)
        $(FC) $(FFLAGS)$(DEBUG)  $(INCS) -c main$(SUFFIX)
xcgrad.o: xcgrad$(SUFFIX)
```

```
	$(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
xcspin.o: xcspin$(SUFFIX)
	$(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
makeparam.o: makeparam$(SUFFIX)
	$(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
makeparam$(SUFFIX): makeparam.F main.F
base.o: base.inc base.F
mgrid.o: mgrid.inc mgrid.F
constant.o: constant.inc constant.F
lattice.o: lattice.inc lattice.F
setex.o: setexm.inc setex.F
pseudo.o: pseudo.inc pseudo.F
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symlib.o : symlib.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symmetry.o : symmetry.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dynbr.o : dynbr.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
broyden.o : broyden.F
	$(CPP)
```

```
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
wave.o : wave.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
fftmpi_map.o : fftmpi_map.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
stockholder.o : stockholder.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
paircorrection.o : paircorrection.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
tet.o : tet.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dos.o : dos.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pseudo.o : pseudo.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
 paw.o : paw.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pot.o : pot.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
sydmat.o : sydmat.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
elinear_response.o : elinear_response.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
chi.o : chi.F
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -qalias_size=500000000 -c $*$(SUFFIX)
```

Makefile for vasp5.3 in gaia1 with BLAS

```
.SUFFIXES: .inc .f .F
SUFFIX=.f
FC  = mpxlf90
F77 = mpxlf
FCL = $(FC)
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr5\" -DMPI -Duse_collective  \
           -DCACHE_SIZE=32768  -DPGF90 \
           -Davoidalloc \
           -DUSE_ZHEEVX \
           -DNGZhalf \
           $*.F >$*$(SUFFIX)
FFLAGS = -q64 -qmaxmem=-1 -qarch=auto -qtune=auto -qcache=auto -qinitauto
OFLAG_0 = -O0
OFLAG_1 = -O  -q64
OFLAG_2 = -O2 -q64
OFLAG_3 = -O3 -qstrict
OFLAG_4 = -O4 -qstrict -qhot
DEBUG   = -g -qfullpath
INCS    =
OFLAG   = $(OFLAG_2)
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH = none
OBJ_NOOPT = none
DEBUG  = -g -qfullpath
INLINE  = $(OFLAG)
LINK    = -q64 -bdynamic -bnoquiet
MPI     = -L/usr/lpp/ppe.poe/lib -lmpi
LAPACK  =  /applic/local/lib/liblapack-PWR5.a
LIB =  -L/usr/lib -lblas \
        -L../vasp.5.lib -ldmy  \
        ../vasp.5.lib/linpack_double.o \
        $(MPI) $(LAPACK) \
        -L/usr/lib -lessl \
        /applic/local/lib/libfftw3.a
FFT3D  = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
BASIC=   symmetry.o symlib.o   lattlib.o  random.o
SOURCE=  base.o      mpi.o       smart_allocate.o      xml.o  \
          constant.o jacobi.o    main_mpi.o  scala.o   \
          asa.o        lattice.o  poscar.o   ini.o  mgrid.o  xclib.o  vdw_nl.o  xclib_grad.o \
          radial.o    pseudo.o   gridq.o      ebs.o  \
          mkpoints.o wave.o      wave_mpi.o  wave_high.o  spinsym.o \
          $(BASIC)    nonl.o      nonlr.o     nonl_high.o dfast.o     choleski2.o \
          mix.o        hamil.o     xcgrad.o    xcspin.o    potex1.o    potex2.o  \
          constrmag.o cl_shift.o relativistic.o LDApU.o \
          paw_base.o metagga.o  egrad.o  pawsym.o  pawfock.o  pawlhf.o  rhfatm.o  hyperfine.o paw.o \
          mkpoints_full.o        charge.o    Lebedev-Laikov.o  stockholder.o dipol.o      pot.o \
          dos.o        elf.o        tet.o        tetweight.o hamil_rot.o \
          chain.o     dyna.o      k-proj.o     sphpro.o     us.o  core_rel.o \
```

```
              aedens.o    wavpre.o    wavpre_noio.o broyden.o \
              dynbr.o     hamil_high.o  rmm-diis.o reader.o    writer.o    tutor.o xml_writer.o \
              brent.o     stufak.o    fileio.o    opergrid.o stepver.o  \
              chgloc.o    fast_aug.o fock_multipole.o  fock.o   mkpoints_change.o sym_grad.o \
              mymath.o    internals.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
              nmr.o       pead.o      subrot.o    subrot_scf.o \
              force.o     pwlhf.o     gw_model.o optreal.o   steep.o     davidson.o  david_inner.o \
              electron.o rot.o  electron_all.o shm.o     pardens.o   paircorrection.o \
              optics.o    constr_cell_relax.o    stm.o     finite_diff.o elpol.o     \
              hamil_lr.o rmm-diis_lr.o  subrot_cluster.o subrot_lr.o \
              lr_helper.o hamil_lrf.o    elinear_response.o ilinear_response.o \
              linear_optics.o \
              setlocalpp.o  wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
              mlwf.o      ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
              local_field.o ump2.o  bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
              rmm-diis_mlr.o  linear_response_NMR.o wannier_interpol.o linear_response.o
   vasp: $(SOURCE) $(FFT3D) $(INC) main.o
              rm -f vasp
              $(FCL) -q64 -o vasp main.o  $(SOURCE)   $(FFT3D) $(LIB) $(LINK)
   makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
              $(FCL) -o makeparam  $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
   zgemmtest: zgemmtest.o base.o random.o $(INC)
              $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
   dgemmtest: dgemmtest.o base.o random.o $(INC)
              $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
   ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
              $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
   kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
              $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
   clean:
              -rm -f *.g *.f *.o *.L *.mod ; touch *.F
   main.o: main$(SUFFIX)
              $(FC) $(FFLAGS)$(DEBUG)  $(INCS) -c main$(SUFFIX)
   xcgrad.o: xcgrad$(SUFFIX)
              $(FC) $(FFLAGS) $(INLINE)  $(INCS) -c xcgrad$(SUFFIX)
   xcspin.o: xcspin$(SUFFIX)
              $(FC) $(FFLAGS) $(INLINE)  $(INCS) -c xcspin$(SUFFIX)
   makeparam.o: makeparam$(SUFFIX)
              $(FC) $(FFLAGS)$(DEBUG)  $(INCS) -c makeparam$(SUFFIX)
   makeparam$(SUFFIX): makeparam.F main.F
   ## MIND: I do not have a full dependency list for the include
   # and MODULES: here are only the minimal basic dependencies
   # if one strucuture is changed then touch_dep must be called
   # with the corresponding name of the structure
   #base.o: base.inc base.F
   mgrid.o: mgrid.inc mgrid.F
   constant.o: constant.inc constant.F
   lattice.o: lattice.inc lattice.F
   setex.o: setexm.inc setex.F
   pseudo.o: pseudo.inc pseudo.F
```

```
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symlib.o : symlib.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symmetry.o : symmetry.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dynbr.o : dynbr.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
broyden.o : broyden.F
	$(CPP)
	$(FC) -O0 -q64 -qmaxmem=-1 -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
wave.o : wave.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
fftmpi_map.o : fftmpi_map.F
	$(CPP)
```

```
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
stockholder.o : stockholder.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
paircorrection.o : paircorrection.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
tet.o : tet.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dos.o : dos.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pseudo.o : pseudo.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
 paw.o : paw.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pot.o : pot.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
sydmat.o : sydmat.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
elinear_response.o : elinear_response.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
```

Makefile for vasp5.3 in gaia2 with BLAS

```
.SUFFIXES: .inc .f .F
SUFFIX=.f
FC  = mpxlf90
F77 = mpxlf
FCL = $(FC)
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr6\" -DMPI -Duse_collective  \
          -DCACHE_SIZE=32768  -DPGF90 \
          -Davoidalloc \
          -DNGZhalf \
          $*.F >$*$(SUFFIX)
#         -Dessl  -DCACHE_SIZE=32768  -DPGF90 \
FFLAGS = -q64 -qmaxmem=-1 -qarch=auto -qtune=auto -qcache=auto -qinitauto
OFLAG_0 = -O0
OFLAG_1 = -O  -q64
OFLAG_2 = -O2 -q64
OFLAG_3 = -O3 -qstrict
OFLAG_4 = -O4 -qstrict -qhot
DEBUG   = -g -qfullpath
INCS    =
OFLAG   = $(OFLAG_2)
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH = none
OBJ_NOOPT = none
DEBUG  = -g -qfullpath
INLINE  = $(OFLAG)
LINK    = -q64 -bdynamic
MPI     = -L/usr/lpp/ppe.poe/lib -lmpi
LAPACK  =  /applic/local/64BIT/LAPACK/V342/lib/liblapack.a
LIB     =  -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lblas \
           -L../vasp.5.lib -ldmy  \
           ../vasp.5.lib/linpack_double.o \
           $(MPI) $(LAPACK) \
           /applic/local/64BIT/FFTW/V322/lib/libfftw3.a
#           -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lessl \

FFT3D  = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
BASIC=   symmetry.o symlib.o  lattlib.o  random.o
SOURCE=  base.o      mpi.o       smart_allocate.o       xml.o  \
         constant.o jacobi.o   main_mpi.o  scala.o   \
         asa.o       lattice.o  poscar.o   ini.o  mgrid.o  xclib.o  vdw_nl.o  xclib_grad.o \
         radial.o    pseudo.o   gridq.o     ebs.o  \
         mkpoints.o wave.o      wave_mpi.o  wave_high.o  spinsym.o \
         $(BASIC)   nonl.o      nonlr.o     nonl_high.o dfast.o     choleski2.o \
         mix.o       hamil.o    xcgrad.o    xcspin.o    potex1.o    potex2.o  \
         constrmag.o cl_shift.o relativistic.o LDApU.o \
         paw_base.o  metagga.o  egrad.o       pawsym.o     pawfock.o    pawlhf.o     rhfatm.o
hyperfine.o paw.o   \
         mkpoints_full.o       charge.o   Lebedev-Laikov.o  stockholder.o dipol.o     pot.o \
         dos.o       elf.o      tet.o      tetweight.o hamil_rot.o \
         chain.o     dyna.o     k-proj.o   sphpro.o    us.o   core_rel.o \
         aedens.o    wavpre.o   wavpre_noio.o broyden.o \
         dynbr.o     hamil_high.o  rmm-diis.o reader.o   writer.o   tutor.o xml_writer.o \
         brent.o     stufak.o   fileio.o   opergrid.o stepver.o  \
         chgloc.o    fast_aug.o fock_multipole.o  fock.o  mkpoints_change.o sym_grad.o \
         mymath.o    internals.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
         nmr.o       pead.o     subrot.o    subrot_scf.o \
         force.o     pwlhf.o    gw_model.o optreal.o  steep.o     davidson.o  david_inner.o \
         electron.o rot.o  electron_all.o shm.o     pardens.o  paircorrection.o \
         optics.o    constr_cell_relax.o   stm.o     finite_diff.o elpol.o     \
```

```
              hamil_lr.o rmm-diis_lr.o   subrot_cluster.o subrot_lr.o \
              lr_helper.o hamil_lrf.o    elinear_response.o ilinear_response.o \
              linear_optics.o \
              setlocalpp.o   wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
              mlwf.o       ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
              local_field.o ump2.o   bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
              rmm-diis_mlr.o   linear_response_NMR.o wannier_interpol.o linear_response.o
   vasp: $(SOURCE) $(FFT3D) $(INC) main.o
            rm -f vasp
            $(FCL) -q64 -o vasp main.o   $(SOURCE)    $(FFT3D) $(LIB) $(LINK)
   makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
            $(FCL) -o makeparam   $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
   zgemmtest: zgemmtest.o base.o random.o $(INC)
            $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
   dgemmtest: dgemmtest.o base.o random.o $(INC)
            $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
   ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
            $(FCL) -o ffttest $(LINK) ffttest.o  mpi.o  mgrid.o  random.o  smart_allocate.o  base.o  $(FFT3D)
$(LIB)
   kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
            $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
   clean:
            -rm -f *.g *.f *.o *.L *.mod ; touch *.F
   main.o: main$(SUFFIX)
            $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c main$(SUFFIX)
   xcgrad.o: xcgrad$(SUFFIX)
            $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
   xcspin.o: xcspin$(SUFFIX)
            $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
   makeparam.o: makeparam$(SUFFIX)
            $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
   makeparam$(SUFFIX): makeparam.F main.F
   ## MIND: I do not have a full dependency list for the include
   # and MODULES: here are only the minimal basic dependencies
   # if one strucuture is changed then touch_dep must be called
   # with the corresponding name of the structure
   #base.o: base.inc base.F
   mgrid.o: mgrid.inc mgrid.F
   constant.o: constant.inc constant.F
   lattice.o: lattice.inc lattice.F
   setex.o: setexm.inc setex.F
   pseudo.o: pseudo.inc pseudo.F
   mkpoints.o: mkpoints.inc mkpoints.F
   wave.o: wave.F
   nonl.o: nonl.inc nonl.F
   nonlr.o: nonlr.inc nonlr.F
   $(OBJ_HIGH):
            $(CPP)
            $(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
   $(OBJ_NOOPT):
            $(CPP)
            $(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
   fft3dlib_f77.o: fft3dlib_f77.F
            $(CPP)
            $(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
   .F.o:
            $(CPP)
            $(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
   .F$(SUFFIX):
            $(CPP)
   $(SUFFIX).o:
            $(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
```

```
# special rules
#-----------------------------------------------------------------------
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symlib.o : symlib.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symmetry.o : symmetry.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dynbr.o : dynbr.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
broyden.o : broyden.F
	$(CPP)
	$(FC) -O0 -q64 -qmaxmem=-1 -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
wave.o : wave.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
fftmpi_map.o : fftmpi_map.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
stockholder.o : stockholder.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
paircorrection.o : paircorrection.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
tet.o : tet.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dos.o : dos.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pseudo.o : pseudo.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
 paw.o : paw.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pot.o : pot.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
sydmat.o : sydmat.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
elinear_response.o : elinear_response.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
```

Makefile for vasp5.3 in gaia2 with scalapack

```
# makefile for gaia2 using scalapack
.SUFFIXES: .inc .f .F
SUFFIX=.f
FC =mpxlf90 -q64 -qfree=f90
F77=mpxlf
FCL=$(FC)
CPP = /usr/ccs/lib/cpp -P -DHOST=\"pwr6\" -DMPI -Duse_collective   \
            -Dessl   -DCACHE_SIZE=32768   -DPGF90 \
            -DUSE_ZHEEVX \
            -Davoidalloc \
            -DNGZhalf \
            -DscaLAPACK \
            $*.F >$*$(SUFFIX)
OFLAG_0 = -O0
OFLAG_1 = -O
OFLAG_2 = -O2
OFLAG_3 = -O3 -qstrict
OFLAG_4 = -O4 -qstrict -qhot
DEBUG    =  -g -qfullpath
INCS     =
OFLAG    = $(OFLAG_2)
INLINE   = $(OFLAG)
#FFLAGS = -qmaxmem=-1 -qarch=auto -qtune=auto -qcache=auto
FFLAGS = -qmaxmem=-1 -qarch=pwr6 -qtune=pwr6
OFLAG   = -O3 -qarch=auto -qstrict
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH = none
OBJ_NOOPT = none
DEBUG   = -g -qfullpath
INLINE = $(OFLAG)
#LINK     = -q64 -bdynamic -bnoquiet
LINK     = -q64 -bdynamic
MPI      = -L/usr/lpp/ppe.poe/lib -lmpi
LAPACK   = /applic/local/64BIT/LAPACK/V342/lib/liblapack.a
SCA      = /applic/local/64BIT/SCALAPACK/V180/lib/libscalapack.a
BLACS    = /applic/local/64BIT/BLACS/V11_PATCH03/lib/blacsCinit_MPI-PWR6-0.a \
           /applic/local/64BIT/BLACS/V11_PATCH03/lib/blacsF77init_MPI-PWR6-0.a \
           /applic/local/64BIT/BLACS/V11_PATCH03/lib/blacs_MPI-PWR6-0.a
#LAPACK   = ../vasp.5.lib/lapack_double.o /home01/g062kyk/programs/espresso-4.1/flib/lapack.a
LIB      = \
                -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -pessl -lessl \
                     -L../vasp.5.lib -ldmy   \
                     ../vasp.5.lib/linpack_double.o \
                $(MPI) $(LAPACK) $(SCA) $(BLACS) -lessl \
                /applic/local/64BIT/FFTW/V322/lib -lfftw3
#LIB      = \
#                -L/usr/llp/xlf/lib -lxlopt -L/usr/lib -lessl \
#                     -L../vasp.5.lib -ldmy   \
#                     ../vasp.5.lib/linpack_double.o \
#             /home01/g062kyk/programs/espresso-4.1/flib/lapack.a \
#                $(MPI) \
#                /applic/local/64BIT/FFTW/V322/lib/libfftw3.a
#FFT3D   = fftmpi.o fftmpi_map.o fft3dessl+furth.o fft3dlib.o
FFT3D   = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
BASIC=    symmetry.o symlib.o    lattlib.o   random.o
SOURCE=   base.o       mpi.o        smart_allocate.o        xml.o   \
          constant.o jacobi.o    main_mpi.o   scala.o    \
          asa.o        lattice.o   poscar.o    ini.o   mgrid.o   xclib.o   vdw_nl.o   xclib_grad.o \
          radial.o    pseudo.o    gridq.o       ebs.o   \
```

```
              mkpoints.o wave.o      wave_mpi.o   wave_high.o spinsym.o\
              $(BASIC)    nonl.o      nonlr.o      nonl_high.o dfast.o      choleski2.o \
              mix.o       hamil.o     xcgrad.o     xcspin.o     potex1.o    potex2.o  \
              constrmag.o cl_shift.o relativistic.o LDApU.o \
              paw_base.o metagga.o   egrad.o      pawsym.o     pawfock.o   pawlhf.o     rhfatm.o   hyperfine.o
paw.o    \
              mkpoints_full.o        charge.o    Lebedev-Laikov.o  stockholder.o dipol.o      pot.o \
              dos.o       elf.o       tet.o       tetweight.o hamil_rot.o \
              chain.o     dyna.o      k-proj.o    sphpro.o     us.o   core_rel.o \
              aedens.o    wavpre.o    wavpre_noio.o broyden.o \
              dynbr.o     hamil_high.o  rmm-diis.o reader.o    writer.o    tutor.o xml_writer.o \
              brent.o     stufak.o    fileio.o    opergrid.o stepver.o  \
              chgloc.o    fast_aug.o fock_multipole.o  fock.o   mkpoints_change.o sym_grad.o \
              mymath.o          internals.o   npt_dynamics.o          dynconstr.o   dimer_heyden.o   dvvtrajectory.o
vdwforcefield.o \
              nmr.o       pead.o      subrot.o    subrot_scf.o \
              force.o     pwlhf.o     gw_model.o optreal.o   steep.o     davidson.o   david_inner.o \
              electron.o rot.o   electron_all.o shm.o      pardens.o   paircorrection.o \
              optics.o    constr_cell_relax.o    stm.o      finite_diff.o elpol.o     \
              hamil_lr.o rmm-diis_lr.o   subrot_cluster.o subrot_lr.o \
              lr_helper.o hamil_lrf.o    elinear_response.o ilinear_response.o \
              linear_optics.o \
              setlocalpp.o   wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
              mlwf.o      ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
              local_field.o ump2.o ump2kpar.o fcidump.o ump2no.o \
              bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
              rmm-diis_mlr.o   linear_response_NMR.o wannier_interpol.o linear_response.o ztrevc.o zlatrs.o
   vasp: $(SOURCE) $(FFT3D) $(INC) main.o
              rm -f vasp.real
              $(FCL) -o vasp.real   main.o   $(SOURCE)    $(FFT3D) $(LIB) $(LINK)
   makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
              $(FCL) -o makeparam   $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
   zgemmtest: zgemmtest.o base.o random.o $(INC)
              $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
   dgemmtest: dgemmtest.o base.o random.o $(INC)
              $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
   ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
              $(FCL)  -o  ffttest  $(LINK)  ffttest.o  mpi.o  mgrid.o  random.o  smart_allocate.o  base.o  $(FFT3D)
$(LIB)
   kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
              $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
   clean:
              -rm -f *.g *.f *.o *.L *.mod ; touch *.F
   main.o: main$(SUFFIX)
              $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c main$(SUFFIX)
   xcgrad.o: xcgrad$(SUFFIX)
              $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
   xcspin.o: xcspin$(SUFFIX)
              $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
   makeparam.o: makeparam$(SUFFIX)
              $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
   makeparam$(SUFFIX): makeparam.F main.F
   base.o: base.inc base.F
   mgrid.o: mgrid.inc mgrid.F
   constant.o: constant.inc constant.F
   lattice.o: lattice.inc lattice.F
   setex.o: setexm.inc setex.F
   pseudo.o: pseudo.inc pseudo.F
   mkpoints.o: mkpoints.inc mkpoints.F
   wave.o: wave.F
   nonl.o: nonl.inc nonl.F
   nonlr.o: nonlr.inc nonlr.F
```

```
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symlib.o : symlib.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
symmetry.o : symmetry.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dynbr.o : dynbr.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
broyden.o : broyden.F
	$(CPP)
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
wave.o : wave.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) $(OFLAG_2) -c $*$(SUFFIX)
fftmpi_map.o : fftmpi_map.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
stockholder.o : stockholder.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
paircorrection.o : paircorrection.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
tet.o : tet.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
dos.o : dos.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pseudo.o : pseudo.F
	$(CPP)
```

```
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
paw.o : paw.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
pot.o : pot.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
sydmat.o : sydmat.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
elinear_response.o : elinear_response.F
	$(CPP)
	$(FC) $(OFLAG_1) -c $*$(SUFFIX)
chi.o : chi.F
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -qalias_size=500000000 -c $*$(SUFFIX)
```

makefile for vasp5.3 in tachyon1 with BLAS/LAPACK

```
.SUFFIXES: .inc .f .f90 .F
#-----------------------------------------------------------------------
# Makefile for Intel Fortran compiler for Pentium/Athlon/Opteron
# bases systems
# we recommend this makefile for both Intel as well as AMD systems
# for AMD based systems appropriate BLAS and fftw libraries are
# however mandatory (whereas they are optional for Intel platforms)
## The makefile was tested only under Linux on Intel and AMD platforms
# the following compiler versions have been tested:
#   - ifc.7.1   works stable somewhat slow but reliably
#   - ifc.8.1   fails to compile the code properly
#   - ifc.9.1   recommended (both for 32 and 64 bit)
#   - ifc.10.1 partially recommended (both for 32 and 64 bit)
#                 tested build 20080312 Package ID: l_fc_p_10.1.015
#                 the gamma only mpi version can not be compiles
#                 using ifc.10.1
#   - ifc.11.1 strongly recommended (we use this to compile vasp)
#                 Build 20090630 Package ID: l_cprof_p_11.1.046
## it might be required to change some of library pathes, since
# LINUX installation vary a lot
# Hence check ***ALL*** options in this makefile very carefully
#-----------------------------------------------------------------------
## BLAS must be installed on the machine
# there are several options:
# 1) very slow but works:
#    retrieve the lapackage from ftp.netlib.org
#    and compile the blas routines (BLAS/SRC directory)
#    please use g77 or f77 for the compilation. When I tried to
#    use pgf77 or pgf90 for BLAS, VASP hang up when calling
#    ZHEEV  (however this was with lapack 1.1 now I use lapack 2.0)
# 2) more desirable: get an optimized BLAS
## the two most reliable packages around are presently:
# 2a) Intels own optimised BLAS (PIII, P4, PD, PC2, Itanium)
#       http://developer.intel.com/software/products/mkl/
#    this is really excellent, if you use Intel CPU's
## 2b) probably fastest SSE2 (4 GFlops on P4, 2.53 GHz, 16 GFlops PD,
#       around 30 GFlops on Quad core)
#    Kazushige Goto's BLAS
#    http://www.cs.utexas.edu/users/kgoto/signup_first.html
#    http://www.tacc.utexas.edu/resources/software/
##-----------------------------------------------------------------------
# all CPP processed fortran files have the extension .f90
SUFFIX=.f90
#-----------------------------------------------------------------------
# fortran compiler and linker
#-----------------------------------------------------------------------
FC=mpif90 -I/applic/lib.intel/FFTW3/include
# fortran linker
FCL=$(FC)
#-----------------------------------------------------------------------
# whereis CPP ?? (I need CPP, can't use gcc with proper options)
# that's the location of gcc for SUSE 5.3
##  CPP_    =   /usr/lib/gcc-lib/i486-linux/2.7.2/cpp -P -C
## that's probably the right line for some Red Hat distribution:
##  CPP_    =   /usr/lib/gcc-lib/i386-redhat-linux/2.7.2.3/cpp -P -C
##  SUSE X.X, maybe some Red Hat distributions:
CPP_ =  ./preprocess <$*.F | /usr/bin/cpp -P -C -traditional >$*$(SUFFIX)
# this release should be fpp clean
# we now recommend fpp as preprocessor
```

```
# if this fails go back to cpp
# CPP_=fpp -f_com=no -free -w0 $*.F $*$(SUFFIX)
#-----------------------------------------------------------------------
# possible options for CPP:
# NGXhalf                 charge density    reduced in X direction
# wNGXhalf                gamma point only reduced in X direction
# avoidalloc              avoid ALLOCATE if possible
# PGF90                   work around some for some PGF90 / IFC bugs
# CACHE_SIZE              1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV            use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV            use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                   MD package of Tomas  Bucko
#-----------------------------------------------------------------------
CPP     = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
      -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
      -DMPI_BLOCK=8000 -Duse_collective
#-----------------------------------------------------------------------
# general fortran flags  (there must a trailing blank on this line)
# byterecl is strictly required for ifc, since otherwise
# the WAVECAR file becomes huge
#-----------------------------------------------------------------------
FFLAGS =  -FR -lowercase -assume byterecl -heap-arrays
#-----------------------------------------------------------------------
# optimization
# we have tested whether higher optimisation improves performance
# -axK  SSE1 optimization,  but also generate code executable on all mach.
#       xK improves performance somewhat on XP, and a is required in order
#       to run the code on older Athlons as well
# -xW   SSE2 optimization
# -axW  SSE2 optimization,  but also generate code executable on all mach.
# -tpp6 P3 optimization
# -tpp7 P4 optimization
#-----------------------------------------------------------------------
# ifc.9.1, ifc.10.1 recommended
OFLAG=-O2 -ip
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH =
OBJ_NOOPT =
DEBUG  = -FR -O0
INLINE = $(OFLAG)
#-----------------------------------------------------------------------
# the following lines specify the position of BLAS  and LAPACK
# VASP works fastest with the libgoto library
# so that's what we recommend
#-----------------------------------------------------------------------
# mkl.10.0
# set -DRPROMU_DGEMV  -DRACCMU_DGEMV in the CPP lines
#BLAS=-L/opt/intel/mkl100/lib/em64t -lmkl -lpthread
# even faster for VASP Kazushige Goto's BLAS
# http://www.cs.utexas.edu/users/kgoto/signup_first.html
# parallel goto version requires sometimes -libverbs
# LAPACK, simplest use vasp.5.lib/lapack_double
#LAPACK= ../vasp.5.lib/lapack_double.o
# use the mkl Intel lapack
#LAPACK= -lmkl_lapack
#-----------------------------------------------------------------------
#LAPACK= /applic/lib.intel/ACML5/ifort64/lib/libacml.a
LAPACK= -L/applic/lib.intel/LAPACK -lblas -llapack
LIB  = -L../vasp.5.lib -ldmy \
      ../vasp.5.lib/linpack_double.o $(LAPACK) -lm \
# options for linking, nothing is required (usually)
LINK     =
```

```
#-----------------------------------------------------------------------
# fft libraries:
# VASP.5.2 can use fftw.3.1.X (http://www.fftw.org)
# since this version is faster on P4 machines, we recommend to use it
#-----------------------------------------------------------------------
FFT3D    = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o \
           /applic/lib.intel/FFTW3/lib/libfftw3.a
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D    = fftw3d.o fft3dlib.o    /opt/libs/fftw-3.1.2/lib/libfftw3.a
#=======================================================================
# MPI section, uncomment the following lines until
#      general  rules and compile lines
# presently we recommend OPENMPI, since it seems to offer better
# performance than lam or mpich
#
# !!! Please do not send me any queries on how to install MPI, I will
# certainly not answer them !!!!
#=======================================================================
#-----------------------------------------------------------------------
# fortran linker for mpi
#-----------------------------------------------------------------------
#FC=mpif77
#FCL=$(FC)
#-----------------------------------------------------------------------
# additional options for CPP in parallel version (see also above):
# NGZhalf                 charge density    reduced in Z direction
# wNGZhalf                gamma point only reduced in Z direction
# scaLAPACK                use scaLAPACK (usually slower on 100 Mbit Net)
# avoidalloc              avoid ALLOCATE if possible
# PGF90                   work around some for some PGF90 / IFC bugs
# CACHE_SIZE              1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV            use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV            use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                   MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#CPP      = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
#       -DCACHE_SIZE=4000  -DPGF90  -Davoidalloc  -DNGZhalf \
#       -DMPI_BLOCK=8000  -Duse_collective
##      -DRPROMU_DGEMV   -DRACCMU_DGEMV
#-----------------------------------------------------------------------
# location of SCALAPACK
# if you do not use SCALAPACK simply leave that section commented out
#-----------------------------------------------------------------------
#BLACS=$(HOME)/archives/SCALAPACK/BLACS/
#SCA_=$(HOME)/archives/SCALAPACK/SCALAPACK
#SCA=  $(SCA_)/libscalapack.a   \
#                  $(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a                  $(BLACS)/LIB/blacs_MPI-LINUX-0.a
$(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a
SCA=
#-----------------------------------------------------------------------
# libraries for mpi
#-----------------------------------------------------------------------
#LIB       = -L../vasp.5.lib -ldmy   \
#         ../vasp.5.lib/linpack_double.o $(LAPACK) \
#         $(SCA) $(BLAS)
# FFT: fftmpi.o with fft3dlib of Juergen Furthmueller
#FFT3D    = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D    = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o   /opt/libs/fftw-3.1.2/lib/libfftw3.a
#-----------------------------------------------------------------------
# general rules and compile lines
```

```
#-----------------------------------------------------------------------
BASIC=    symmetry.o symlib.o    lattlib.o   random.o
SOURCE=   base.o       mpi.o        smart_allocate.o        xml.o   \
          constant.o jacobi.o    main_mpi.o   scala.o    \
          asa.o        lattice.o   poscar.o    ini.o   mgrid.o   xclib.o   vdw_nl.o   xclib_grad.o \
          radial.o    pseudo.o    gridq.o       ebs.o   \
          mkpoints.o wave.o       wave_mpi.o   wave_high.o   spinsym.o \
          $(BASIC)    nonl.o       nonlr.o      nonl_high.o dfast.o      choleski2.o \
          mix.o        hamil.o     xcgrad.o    xcspin.o      potex1.o    potex2.o   \
          constrmag.o cl_shift.o relativistic.o LDApU.o \
          paw_base.o metagga.o   egrad.o       pawsym.o     pawfock.o   pawlhf.o     rhfatm.o   hyperfine.o
paw.o    \
          mkpoints_full.o          charge.o    Lebedev-Laikov.o   stockholder.o dipol.o      pot.o \
          dos.o        elf.o        tet.o        tetweight.o hamil_rot.o \
          chain.o      dyna.o       k-proj.o      sphpro.o      us.o   core_rel.o \
          aedens.o    wavpre.o    wavpre_noio.o broyden.o \
          dynbr.o      hamil_high.o   rmm-diis.o reader.o    writer.o    tutor.o xml_writer.o \
          brent.o      stufak.o    fileio.o    opergrid.o stepver.o   \
          chgloc.o    fast_aug.o fock_multipole.o   fock.o   mkpoints_change.o sym_grad.o \
          mymath.o    internals.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
          nmr.o        pead.o       subrot.o    subrot_scf.o \
          force.o      pwlhf.o      gw_model.o optreal.o   steep.o      davidson.o   david_inner.o \
          electron.o rot.o   electron_all.o shm.o      pardens.o   paircorrection.o \
          optics.o    constr_cell_relax.o    stm.o      finite_diff.o elpol.o      \
          hamil_lr.o rmm-diis_lr.o   subrot_cluster.o subrot_lr.o \
          lr_helper.o hamil_lrf.o    elinear_response.o ilinear_response.o \
          linear_optics.o \
          setlocalpp.o   wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
          mlwf.o       ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
          local_field.o ump2.o   bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
          rmm-diis_mlr.o   linear_response_NMR.o wannier_interpol.o linear_response.o
vasp: $(SOURCE) $(FFT3D) $(INC) main.o
          rm -f vasp
          $(FCL) -o vasp main.o   $(SOURCE)    $(FFT3D) $(LIB) $(LINK)
makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
          $(FCL) -o makeparam   $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
zgemmtest: zgemmtest.o base.o random.o $(INC)
          $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
dgemmtest: dgemmtest.o base.o random.o $(INC)
          $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
          $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
          $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
clean:
          -rm -f *.g *.f *.o *.L *.mod ; touch *.F
main.o: main$(SUFFIX)
          $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c main$(SUFFIX)
xcgrad.o: xcgrad$(SUFFIX)
          $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
xcspin.o: xcspin$(SUFFIX)
          $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
makeparam.o: makeparam$(SUFFIX)
          $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
makeparam$(SUFFIX): makeparam.F main.F
## MIND: I do not have a full dependency list for the include
# and MODULES: here are only the minimal basic dependencies
# if one strucuture is changed then touch_dep must be called
# with the corresponding name of the structure
#base.o: base.inc base.F
mgrid.o: mgrid.inc mgrid.F
```

```
constant.o: constant.inc constant.F
lattice.o: lattice.inc lattice.F
setex.o: setexm.inc setex.F
pseudo.o: pseudo.inc pseudo.F
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
# these special rules have been tested for ifc.11 and ifc.12 only
fft3dlib.o : fft3dlib.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpi.o : fftmpi.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpiw.o : fftmpiw.F
	$(CPP)
	$(FC) -FR -lowercase -O1 $(INCS) -c $*$(SUFFIX)
wave_high.o : wave_high.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
# the following rules are probably no longer required (-O3 seems to work)
wave.o : wave.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
paw.o : paw.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
cl_shift.o : cl_shift.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
```

makefile for vasp5.3 in tachyon1 with ACML

```
.SUFFIXES: .inc .f .f90 .F
#-----------------------------------------------------------------------
# Makefile for Intel Fortran compiler for Pentium/Athlon/Opteron
# bases systems
# we recommend this makefile for both Intel as well as AMD systems
# for AMD based systems appropriate BLAS and fftw libraries are
# however mandatory (whereas they are optional for Intel platforms)
## The makefile was tested only under Linux on Intel and AMD platforms
# the following compiler versions have been tested:
#  - ifc.7.1  works stable somewhat slow but reliably
#  - ifc.8.1  fails to compile the code properly
#  - ifc.9.1  recommended (both for 32 and 64 bit)
#  - ifc.10.1 partially recommended (both for 32 and 64 bit)
#               tested build 20080312 Package ID: l_fc_p_10.1.015
#               the gamma only mpi version can not be compiles
#               using ifc.10.1
#  - ifc.11.1 strongly recommended (we use this to compile vasp)
#               Build 20090630 Package ID: l_cprof_p_11.1.046
## it might be required to change some of library pathes, since
# LINUX installation vary a lot
# Hence check ***ALL*** options in this makefile very carefully
#-----------------------------------------------------------------------
## BLAS must be installed on the machine
# there are several options:
# 1) very slow but works:
#    retrieve the lapackage from ftp.netlib.org
#    and compile the blas routines (BLAS/SRC directory)
#    please use g77 or f77 for the compilation. When I tried to
#    use pgf77 or pgf90 for BLAS, VASP hang up when calling
#    ZHEEV  (however this was with lapack 1.1 now I use lapack 2.0)
# 2) more desirable: get an optimized BLAS
## the two most reliable packages around are presently:
# 2a) Intels own optimised BLAS (PIII, P4, PD, PC2, Itanium)
#       http://developer.intel.com/software/products/mkl/
#    this is really excellent, if you use Intel CPU's
## 2b) probably fastest SSE2 (4 GFlops on P4, 2.53 GHz, 16 GFlops PD,
#       around 30 GFlops on Quad core)
#    Kazushige Goto's BLAS
#    http://www.cs.utexas.edu/users/kgoto/signup_first.html
#    http://www.tacc.utexas.edu/resources/software/
##-----------------------------------------------------------------------
# all CPP processed fortran files have the extension .f90
SUFFIX=.f90
#-----------------------------------------------------------------------
# fortran compiler and linker
#-----------------------------------------------------------------------
FC=mpif90 -I/applic/lib.intel/FFTW3/include
# fortran linker
FCL=$(FC)
#-----------------------------------------------------------------------
# whereis CPP ?? (I need CPP, can't use gcc with proper options)
# that's the location of gcc for SUSE 5.3
##  CPP_    =  /usr/lib/gcc-lib/i486-linux/2.7.2/cpp -P -C
## that's probably the right line for some Red Hat distribution:
##  CPP_    =  /usr/lib/gcc-lib/i386-redhat-linux/2.7.2.3/cpp -P -C
##  SUSE X.X, maybe some Red Hat distributions:
CPP_ =  ./preprocess <$*.F | /usr/bin/cpp -P -C -traditional >$*$(SUFFIX)
# this release should be fpp clean
# we now recommend fpp as preprocessor
# if this fails go back to cpp
# CPP_=fpp -f_com=no -free -w0 $*.F $*$(SUFFIX)
```

```
#-----------------------------------------------------------------------
# possible options for CPP:
# NGXhalf               charge density   reduced in X direction
# wNGXhalf              gamma point only reduced in X direction
# avoidalloc            avoid ALLOCATE if possible
# PGF90                 work around some for some PGF90 / IFC bugs
# CACHE_SIZE            1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV          use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV          use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                 MD package of Tomas  Bucko
#-----------------------------------------------------------------------
CPP     = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
      -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
      -DMPI_BLOCK=8000 -Duse_collective
#-----------------------------------------------------------------------
# general fortran flags  (there must a trailing blank on this line)
# byterecl is strictly required for ifc, since otherwise
# the WAVECAR file becomes huge
#-----------------------------------------------------------------------
FFLAGS =  -FR -lowercase -assume byterecl -heap-arrays
#-----------------------------------------------------------------------
# optimization
# we have tested whether higher optimisation improves performance
# -axK  SSE1 optimization,  but also generate code executable on all mach.
#       xK improves performance somewhat on XP, and a is required in order
#       to run the code on older Athlons as well
# -xW   SSE2 optimization
# -axW  SSE2 optimization,  but also generate code executable on all mach.
# -tpp6 P3 optimization
# -tpp7 P4 optimization
#-----------------------------------------------------------------------
# ifc.9.1, ifc.10.1 recommended
OFLAG=-O2 -ip
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH =
OBJ_NOOPT =
DEBUG  = -FR -O0
INLINE = $(OFLAG)
#-----------------------------------------------------------------------
# the following lines specify the position of BLAS  and LAPACK
# VASP works fastest with the libgoto library
# so that's what we recommend
#-----------------------------------------------------------------------
# mkl.10.0
# set -DRPROMU_DGEMV  -DRACCMU_DGEMV in the CPP lines
#BLAS=-L/opt/intel/mkl100/lib/em64t -lmkl -lpthread
# even faster for VASP Kazushige Goto's BLAS
# http://www.cs.utexas.edu/users/kgoto/signup_first.html
# parallel goto version requires sometimes -libverbs
# LAPACK, simplest use vasp.5.lib/lapack_double
#LAPACK= ../vasp.5.lib/lapack_double.o
# use the mkl Intel lapack
#LAPACK= -lmkl_lapack
#-----------------------------------------------------------------------
#BLAS  = /applic/lib.intel/ACML5/ifort64/lib/libacml.a
LAPACK= /applic/lib.intel/ACML5/ifort64/lib/libacml.a
LIB   = -L../vasp.5.lib -ldmy \
      ../vasp.5.lib/linpack_double.o $(LAPACK) -lm \
# options for linking, nothing is required (usually)
LINK    =
#-----------------------------------------------------------------------
# fft libraries:
```

```
# VASP.5.2 can use fftw.3.1.X (http://www.fftw.org)
# since this version is faster on P4 machines, we recommend to use it
#-----------------------------------------------------------------------
FFT3D   = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o   /applic/lib.intel/FFTW3/lib/libfftw3.a
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D   = fftw3d.o fft3dlib.o   /opt/libs/fftw-3.1.2/lib/libfftw3.a
#=======================================================================
# MPI section, uncomment the following lines until
#     general  rules and compile lines
# presently we recommend OPENMPI, since it seems to offer better
# performance than lam or mpich
#
# !!! Please do not send me any queries on how to install MPI, I will
# certainly not answer them !!!!
#=======================================================================
#-----------------------------------------------------------------------
# fortran linker for mpi
#-----------------------------------------------------------------------
#FC=mpif77
#FCL=$(FC)
#-----------------------------------------------------------------------
# additional options for CPP in parallel version (see also above):
# NGZhalf                 charge density   reduced in Z direction
# wNGZhalf                gamma point only reduced in Z direction
# scaLAPACK               use scaLAPACK (usually slower on 100 Mbit Net)
# avoidalloc              avoid ALLOCATE if possible
# PGF90                   work around some for some PGF90 / IFC bugs
# CACHE_SIZE              1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV            use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV            use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                   MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#CPP     = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
#       -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
#       -DMPI_BLOCK=8000 -Duse_collective
##      -DRPROMU_DGEMV  -DRACCMU_DGEMV
#-----------------------------------------------------------------------
# location of SCALAPACK
# if you do not use SCALAPACK simply leave that section commented out
#-----------------------------------------------------------------------
#BLACS=$(HOME)/archives/SCALAPACK/BLACS/
#SCA_=$(HOME)/archives/SCALAPACK/SCALAPACK
#SCA= $(SCA_)/libscalapack.a  \
#                $(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a                $(BLACS)/LIB/blacs_MPI-LINUX-0.a
$(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a
SCA=
#-----------------------------------------------------------------------
# libraries for mpi
#-----------------------------------------------------------------------
#LIB      = -L../vasp.5.lib -ldmy  \
#        ../vasp.5.lib/linpack_double.o $(LAPACK) \
#        $(SCA) $(BLAS)
# FFT: fftmpi.o with fft3dlib of Juergen Furthmueller
#FFT3D   = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D   = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o   /opt/libs/fftw-3.1.2/lib/libfftw3.a
#-----------------------------------------------------------------------
# general rules and compile lines
#-----------------------------------------------------------------------
BASIC=   symmetry.o symlib.o   lattlib.o  random.o
SOURCE=  base.o      mpi.o       smart_allocate.o       xml.o  \
```

```
             constant.o jacobi.o     main_mpi.o   scala.o    \
             asa.o          lattice.o   poscar.o     ini.o   mgrid.o   xclib.o   vdw_nl.o   xclib_grad.o \
             radial.o     pseudo.o     gridq.o        ebs.o   \
             mkpoints.o wave.o        wave_mpi.o   wave_high.o   spinsym.o \
             $(BASIC)     nonl.o        nonlr.o       nonl_high.o dfast.o       choleski2.o \
             mix.o          hamil.o      xcgrad.o     xcspin.o       potex1.o     potex2.o   \
             constrmag.o cl_shift.o relativistic.o LDApU.o \
             paw_base.o metagga.o    egrad.o        pawsym.o      pawfock.o    pawlhf.o      rhfatm.o    hyperfine.o
paw.o    \
             mkpoints_full.o          charge.o     Lebedev-Laikov.o   stockholder.o dipol.o       pot.o \
             dos.o          elf.o          tet.o          tetweight.o hamil_rot.o \
             chain.o       dyna.o        k-proj.o       sphpro.o       us.o   core_rel.o \
             aedens.o     wavpre.o     wavpre_noio.o broyden.o \
             dynbr.o       hamil_high.o   rmm-diis.o reader.o     writer.o     tutor.o xml_writer.o \
             brent.o       stufak.o     fileio.o     opergrid.o stepver.o   \
             chgloc.o     fast_aug.o fock_multipole.o   fock.o   mkpoints_change.o sym_grad.o \
             mymath.o     internals.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
             nmr.o          pead.o        subrot.o     subrot_scf.o \
             force.o       pwlhf.o       gw_model.o optreal.o   steep.o      davidson.o   david_inner.o \
             electron.o rot.o   electron_all.o shm.o       pardens.o   paircorrection.o \
             optics.o     constr_cell_relax.o     stm.o       finite_diff.o elpol.o      \
             hamil_lr.o rmm-diis_lr.o   subrot_cluster.o subrot_lr.o \
             lr_helper.o hamil_lrf.o     elinear_response.o ilinear_response.o \
             linear_optics.o \
             setlocalpp.o   wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
             mlwf.o        ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
             local_field.o ump2.o   bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
             rmm-diis_mlr.o   linear_response_NMR.o wannier_interpol.o linear_response.o
   vasp: $(SOURCE) $(FFT3D) $(INC) main.o
             rm -f vasp
             $(FCL) -o vasp main.o   $(SOURCE)    $(FFT3D) $(LIB) $(LINK)
   makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
             $(FCL) -o makeparam   $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
   zgemmtest: zgemmtest.o base.o random.o $(INC)
             $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
   dgemmtest: dgemmtest.o base.o random.o $(INC)
             $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
   ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
             $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
   kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
             $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
   clean:
             -rm -f *.g *.f *.o *.L *.mod ; touch *.F
   main.o: main$(SUFFIX)
             $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c main$(SUFFIX)
   xcgrad.o: xcgrad$(SUFFIX)
             $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
   xcspin.o: xcspin$(SUFFIX)
             $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
   makeparam.o: makeparam$(SUFFIX)
             $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
   makeparam$(SUFFIX): makeparam.F main.F
   ## MIND: I do not have a full dependency list for the include
   # and MODULES: here are only the minimal basic dependencies
   # if one strucuture is changed then touch_dep must be called
   # with the corresponding name of the structure
   #base.o: base.inc base.F
   mgrid.o: mgrid.inc mgrid.F
   constant.o: constant.inc constant.F
   lattice.o: lattice.inc lattice.F
   setex.o: setexm.inc setex.F
```

```
pseudo.o: pseudo.inc pseudo.F
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
# these special rules have been tested for ifc.11 and ifc.12 only
fft3dlib.o : fft3dlib.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpi.o : fftmpi.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpiw.o : fftmpiw.F
	$(CPP)
	$(FC) -FR -lowercase -O1 $(INCS) -c $*$(SUFFIX)
wave_high.o : wave_high.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
# the following rules are probably no longer required (-O3 seems to work)
wave.o : wave.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
paw.o : paw.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
cl_shift.o : cl_shift.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
```

makefile for vasp5.3 in tachyon2 with BLAS/LAPACK

```
.SUFFIXES: .inc .f .f90 .F
#-----------------------------------------------------------------------
# Makefile for Intel Fortran compiler for Pentium/Athlon/Opteron
# bases systems
# we recommend this makefile for both Intel as well as AMD systems
# for AMD based systems appropriate BLAS and fftw libraries are
# however mandatory (whereas they are optional for Intel platforms)
## The makefile was tested only under Linux on Intel and AMD platforms
# the following compiler versions have been tested:
#  - ifc.7.1  works stable somewhat slow but reliably
#  - ifc.8.1  fails to compile the code properly
#  - ifc.9.1  recommended (both for 32 and 64 bit)
#  - ifc.10.1 partially recommended (both for 32 and 64 bit)
#               tested build 20080312 Package ID: l_fc_p_10.1.015
#               the gamma only mpi version can not be compiles
#               using ifc.10.1
#  - ifc.11.1 strongly recommended (we use this to compile vasp)
#               Build 20090630 Package ID: l_cprof_p_11.1.046
## it might be required to change some of library pathes, since
# LINUX installation vary a lot
# Hence check ***ALL*** options in this makefile very carefully
#-----------------------------------------------------------------------
## BLAS must be installed on the machine
# there are several options:
# 1) very slow but works:
#    retrieve the lapackage from ftp.netlib.org
#    and compile the blas routines (BLAS/SRC directory)
#    please use g77 or f77 for the compilation. When I tried to
#    use pgf77 or pgf90 for BLAS, VASP hang up when calling
#    ZHEEV  (however this was with lapack 1.1 now I use lapack 2.0)
# 2) more desirable: get an optimized BLAS
## the two most reliable packages around are presently:
# 2a) Intels own optimised BLAS (PIII, P4, PD, PC2, Itanium)
#       http://developer.intel.com/software/products/mkl/
#    this is really excellent, if you use Intel CPU's
## 2b) probably fastest SSE2 (4 GFlops on P4, 2.53 GHz, 16 GFlops PD,
#       around 30 GFlops on Quad core)
#    Kazushige Goto's BLAS
#    http://www.cs.utexas.edu/users/kgoto/signup_first.html
#    http://www.tacc.utexas.edu/resources/software/
##-----------------------------------------------------------------------
# all CPP processed fortran files have the extension .f90
SUFFIX=.f90
CPP      = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
       -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
       -DMPI_BLOCK=8000 -Duse_collective
##      -DRPROMU_DGEMV   -DRACCMU_DGEMV
#-----------------------------------------------------------------------
# fortran compiler and linker
#-----------------------------------------------------------------------
FC=mpif90 -I/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/include
# fortran linker
FCL=$(FC)
#-----------------------------------------------------------------------
# whereis CPP ?? (I need CPP, can't use gcc with proper options)
# that's the location of gcc for SUSE 5.3
##  CPP_    =  /usr/lib/gcc-lib/i486-linux/2.7.2/cpp -P -C
## that's probably the right line for some Red Hat distribution:
##  CPP_    =  /usr/lib/gcc-lib/i386-redhat-linux/2.7.2.3/cpp -P -C
##  SUSE X.X, maybe some Red Hat distributions:
```

```
CPP_ =  ./preprocess <$*.F | /usr/bin/cpp -P -C -traditional >$*$(SUFFIX)
# this release should be fpp clean
# we now recommend fpp as preprocessor
# if this fails go back to cpp
# CPP_=fpp -f_com=no -free -w0 $*.F $*$(SUFFIX)
#-----------------------------------------------------------------------
# possible options for CPP:
# NGXhalf                 charge density    reduced in X direction
# wNGXhalf                gamma point only reduced in X direction
# avoidalloc              avoid ALLOCATE if possible
# PGF90                   work around some for some PGF90 / IFC bugs
# CACHE_SIZE              1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV            use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV            use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                   MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
# general fortran flags  (there must a trailing blank on this line)
# byterecl is strictly required for ifc, since otherwise
# the WAVECAR file becomes huge
#-----------------------------------------------------------------------
FFLAGS =  -FR -lowercase -assume byterecl -heap-arrays
#-----------------------------------------------------------------------
# optimization
# we have tested whether higher optimisation improves performance
# -axK  SSE1 optimization,  but also generate code executable on all mach.
#       xK improves performance somewhat on XP, and a is required in order
#       to run the code on older Athlons as well
# -xW   SSE2 optimization
# -axW  SSE2 optimization,  but also generate code executable on all mach.
# -tpp6 P3 optimization
# -tpp7 P4 optimization
#-----------------------------------------------------------------------
# ifc.9.1, ifc.10.1 recommended
OFLAG=-O2 -ip
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH =
OBJ_NOOPT =
DEBUG  = -FR -O0
INLINE = $(OFLAG)
#-----------------------------------------------------------------------
# the following lines specify the position of BLAS  and LAPACK
# VASP works fastest with the libgoto library
# so that's what we recommend
#-----------------------------------------------------------------------
# mkl.10.0
# set -DRPROMU_DGEMV  -DRACCMU_DGEMV in the CPP lines
#BLAS=-L/opt/intel/mkl100/lib/em64t -lmkl -lpthread
# even faster for VASP Kazushige Goto's BLAS
# http://www.cs.utexas.edu/users/kgoto/signup_first.html
# parallel goto version requires sometimes -libverbs
#BLAS=  /opt/libs/libgoto/libgoto.so
# LAPACK, simplest use vasp.5.lib/lapack_double
#LAPACK= ../vasp.5.lib/lapack_double.o
# use the mkl Intel lapack
#LAPACK= -lmkl_lapack
#-----------------------------------------------------------------------
#MKLROOT = /applic/compilers/intel/11.1/mkl
#BLAS    = -L$(MKLROOT)/lib/em64t -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lpthread -lm
LAPACK    = -L/applic/compilers/intel/2013/applib1/LAPACK -lblas -llapack
#LAPACK = /applic/compilers/intel/2013/applib1/LAPACK/liblapack.a
#LAPACK = -L$(MKLROOT)/lib/em64t -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lpthread -lm
```

```
LIB  = -L../vasp.5.lib -ldmy \
       ../vasp.5.lib/linpack_double.o $(LAPACK) -lm
# options for linking, nothing is required (usually)
LINK     =
#-----------------------------------------------------------------------
# fft libraries:
# VASP.5.2 can use fftw.3.1.X (http://www.fftw.org)
# since this version is faster on P4 machines, we recommend to use it
#-----------------------------------------------------------------------
FFT3D                                =          fftmpiw.o          fftmpi_map.o          fftw3d.o          fft3dlib.o
/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/lib/libfftw3.a
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D    = fftw3d.o fft3dlib.o    /opt/libs/fftw-3.1.2/lib/libfftw3.a
#=======================================================================
# MPI section, uncomment the following lines until
#      general  rules and compile lines
# presently we recommend OPENMPI, since it seems to offer better
# performance than lam or mpich
#
# !!! Please do not send me any queries on how to install MPI, I will
# certainly not answer them !!!!
#=======================================================================
#-----------------------------------------------------------------------
# fortran linker for mpi
#-----------------------------------------------------------------------
#FC=mpif77
#FCL=$(FC)
#-----------------------------------------------------------------------
# additional options for CPP in parallel version (see also above):
# NGZhalf                    charge density    reduced in Z direction
# wNGZhalf                   gamma point only reduced in Z direction
# scaLAPACK                   use scaLAPACK (usually slower on 100 Mbit Net)
# avoidalloc               avoid ALLOCATE if possible
# PGF90                    work around some for some PGF90 / IFC bugs
# CACHE_SIZE                 1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV             use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV             use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                    MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#CPP      = $(CPP_) -DMPI   -DHOST=\"LinuxIFC\" -DIFC \
#       -DCACHE_SIZE=4000  -DPGF90  -Davoidalloc  -DNGZhalf \
#       -DMPI_BLOCK=8000  -Duse_collective
##      -DRPROMU_DGEMV   -DRACCMU_DGEMV
#-----------------------------------------------------------------------
# location of SCALAPACK
# if you do not use SCALAPACK simply leave that section commented out
#-----------------------------------------------------------------------
#BLACS=$(HOME)/archives/SCALAPACK/BLACS/
#SCA_=$(HOME)/archives/SCALAPACK/SCALAPACK
#SCA=  $(SCA_)/libscalapack.a   \
#                 $(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a                 $(BLACS)/LIB/blacs_MPI-LINUX-0.a
$(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a
SCA=
#-----------------------------------------------------------------------
# libraries for mpi
#-----------------------------------------------------------------------
#LIB       = -L../vasp.5.lib -ldmy   \
#         ../vasp.5.lib/linpack_double.o $(LAPACK) \
#         $(SCA) $(BLAS)
# FFT: fftmpi.o with fft3dlib of Juergen Furthmueller
#FFT3D    = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
```

```
  # alternatively: fftw.3.1.X is slighly faster and should be used if available
  #FFT3D    = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o   /opt/libs/fftw-3.1.2/lib/libfftw3.a
  #-----------------------------------------------------------------------
  # general rules and compile lines
  #-----------------------------------------------------------------------
  BASIC=    symmetry.o symlib.o    lattlib.o   random.o
  SOURCE=   base.o       mpi.o        smart_allocate.o        xml.o   \
            constant.o jacobi.o    main_mpi.o   scala.o    \
            asa.o        lattice.o   poscar.o    ini.o   mgrid.o   xclib.o   vdw_nl.o   xclib_grad.o \
            radial.o    pseudo.o    gridq.o       ebs.o   \
            mkpoints.o wave.o       wave_mpi.o   wave_high.o   spinsym.o \
            $(BASIC)    nonl.o       nonlr.o      nonl_high.o dfast.o      choleski2.o \
            mix.o        hamil.o     xcgrad.o    xcspin.o      potex1.o    potex2.o   \
            constrmag.o cl_shift.o relativistic.o LDApU.o \
            paw_base.o metagga.o   egrad.o      pawsym.o     pawfock.o   pawlhf.o     rhfatm.o    hyperfine.o
paw.o    \
            mkpoints_full.o          charge.o    Lebedev-Laikov.o   stockholder.o dipol.o      pot.o \
            dos.o         elf.o         tet.o         tetweight.o hamil_rot.o \
            chain.o      dyna.o       k-proj.o      sphpro.o      us.o   core_rel.o \
            aedens.o    wavpre.o    wavpre_noio.o broyden.o \
            dynbr.o      hamil_high.o   rmm-diis.o reader.o     writer.o     tutor.o xml_writer.o \
            brent.o      stufak.o    fileio.o     opergrid.o stepver.o   \
            chgloc.o    fast_aug.o fock_multipole.o   fock.o   mkpoints_change.o sym_grad.o \
            mymath.o    internals.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
            nmr.o        pead.o       subrot.o    subrot_scf.o \
            force.o      pwlhf.o      gw_model.o optreal.o   steep.o      davidson.o   david_inner.o \
            electron.o rot.o   electron_all.o shm.o      pardens.o   paircorrection.o \
            optics.o    constr_cell_relax.o    stm.o      finite_diff.o elpol.o      \
            hamil_lr.o rmm-diis_lr.o   subrot_cluster.o subrot_lr.o \
            lr_helper.o hamil_lrf.o    elinear_response.o ilinear_response.o \
            linear_optics.o \
            setlocalpp.o   wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
            mlwf.o       ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
            local_field.o ump2.o   bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
            rmm-diis_mlr.o   linear_response_NMR.o wannier_interpol.o linear_response.o
  vasp: $(SOURCE) $(FFT3D) $(INC) main.o
            rm -f vasp
            $(FCL) -o vasp main.o   $(SOURCE)    $(FFT3D) $(LIB) $(LINK)
  makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
            $(FCL) -o makeparam   $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
  zgemmtest: zgemmtest.o base.o random.o $(INC)
            $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
  dgemmtest: dgemmtest.o base.o random.o $(INC)
            $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
  ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
            $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
  kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
            $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
  clean:
            -rm -f *.g *.f *.o *.L *.mod ; touch *.F
  main.o: main$(SUFFIX)
            $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c main$(SUFFIX)
  xcgrad.o: xcgrad$(SUFFIX)
            $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
  xcspin.o: xcspin$(SUFFIX)
            $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
  makeparam.o: makeparam$(SUFFIX)
            $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
  makeparam$(SUFFIX): makeparam.F main.F
  ## MIND: I do not have a full dependency list for the include
  # and MODULES: here are only the minimal basic dependencies
```

```
# if one strucuture is changed then touch_dep must be called
# with the corresponding name of the structure
#base.o: base.inc base.F
mgrid.o: mgrid.inc mgrid.F
constant.o: constant.inc constant.F
lattice.o: lattice.inc lattice.F
setex.o: setexm.inc setex.F
pseudo.o: pseudo.inc pseudo.F
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
# these special rules have been tested for ifc.11 and ifc.12 only
fft3dlib.o : fft3dlib.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpi.o : fftmpi.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpiw.o : fftmpiw.F
	$(CPP)
	$(FC) -FR -lowercase -O1 $(INCS) -c $*$(SUFFIX)
wave_high.o : wave_high.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
# the following rules are probably no longer required (-O3 seems to work)
wave.o : wave.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
paw.o : paw.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
cl_shift.o : cl_shift.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
```

makefile for vasp5.3 in tachyon2 with ACML

```
.SUFFIXES: .inc .f .f90 .F
#-----------------------------------------------------------------------
# Makefile for Intel Fortran compiler for Pentium/Athlon/Opteron
# bases systems
# we recommend this makefile for both Intel as well as AMD systems
# for AMD based systems appropriate BLAS and fftw libraries are
# however mandatory (whereas they are optional for Intel platforms)
## The makefile was tested only under Linux on Intel and AMD platforms
# the following compiler versions have been tested:
#  - ifc.7.1  works stable somewhat slow but reliably
#  - ifc.8.1  fails to compile the code properly
#  - ifc.9.1  recommended (both for 32 and 64 bit)
#  - ifc.10.1 partially recommended (both for 32 and 64 bit)
#               tested build 20080312 Package ID: l_fc_p_10.1.015
#               the gamma only mpi version can not be compiles
#               using ifc.10.1
#  - ifc.11.1 strongly recommended (we use this to compile vasp)
#               Build 20090630 Package ID: l_cprof_p_11.1.046
## it might be required to change some of library pathes, since
# LINUX installation vary a lot
# Hence check ***ALL*** options in this makefile very carefully
#-----------------------------------------------------------------------
## BLAS must be installed on the machine
# there are several options:
# 1) very slow but works:
#    retrieve the lapackage from ftp.netlib.org
#    and compile the blas routines (BLAS/SRC directory)
#    please use g77 or f77 for the compilation. When I tried to
#    use pgf77 or pgf90 for BLAS, VASP hang up when calling
#    ZHEEV  (however this was with lapack 1.1 now I use lapack 2.0)
# 2) more desirable: get an optimized BLAS
## the two most reliable packages around are presently:
# 2a) Intels own optimised BLAS (PIII, P4, PD, PC2, Itanium)
#       http://developer.intel.com/software/products/mkl/
#    this is really excellent, if you use Intel CPU's
## 2b) probably fastest SSE2 (4 GFlops on P4, 2.53 GHz, 16 GFlops PD,
#       around 30 GFlops on Quad core)
#    Kazushige Goto's BLAS
#    http://www.cs.utexas.edu/users/kgoto/signup_first.html
#    http://www.tacc.utexas.edu/resources/software/
##-----------------------------------------------------------------------
# all CPP processed fortran files have the extension .f90
SUFFIX=.f90
CPP      = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
       -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
       -DMPI_BLOCK=8000 -Duse_collective
##      -DRPROMU_DGEMV   -DRACCMU_DGEMV
#-----------------------------------------------------------------------
# fortran compiler and linker
#-----------------------------------------------------------------------
FC=mpif90 -I/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/include
# fortran linker
FCL=$(FC)
#-----------------------------------------------------------------------
# whereis CPP ?? (I need CPP, can't use gcc with proper options)
# that's the location of gcc for SUSE 5.3
##  CPP_    =  /usr/lib/gcc-lib/i486-linux/2.7.2/cpp -P -C
## that's probably the right line for some Red Hat distribution:
##  CPP_    =  /usr/lib/gcc-lib/i386-redhat-linux/2.7.2.3/cpp -P -C
##  SUSE X.X, maybe some Red Hat distributions:
CPP_ =  ./preprocess <$*.F | /usr/bin/cpp -P -C -traditional >$*$(SUFFIX)
# this release should be fpp clean
```

```
# we now recommend fpp as preprocessor
# if this fails go back to cpp
#CPP_=fpp -f_com=no -free -w0 $*.F $*$(SUFFIX)
#-----------------------------------------------------------------------
# possible options for CPP:
# NGXhalf               charge density   reduced in X direction
# wNGXhalf              gamma point only reduced in X direction
# avoidalloc            avoid ALLOCATE if possible
# PGF90                 work around some for some PGF90 / IFC bugs
# CACHE_SIZE            1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV          use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV          use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                 MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
# general fortran flags  (there must a trailing blank on this line)
# byterecl is strictly required for ifc, since otherwise
# the WAVECAR file becomes huge
#-----------------------------------------------------------------------
FFLAGS =  -FR -lowercase -assume byterecl -heap-arrays
#-----------------------------------------------------------------------
# optimization
# we have tested whether higher optimisation improves performance
# -axK  SSE1 optimization,  but also generate code executable on all mach.
#       xK improves performance somewhat on XP, and a is required in order
#       to run the code on older Athlons as well
# -xW   SSE2 optimization
# -axW  SSE2 optimization,  but also generate code executable on all mach.
# -tpp6 P3 optimization
# -tpp7 P4 optimization
#-----------------------------------------------------------------------
# ifc.9.1, ifc.10.1 recommended
OFLAG=-O2 -ip
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH =
OBJ_NOOPT =
DEBUG  = -FR -O0
INLINE = $(OFLAG)
#-----------------------------------------------------------------------
# the following lines specify the position of BLAS  and LAPACK
# VASP works fastest with the libgoto library
# so that's what we recommend
#-----------------------------------------------------------------------
# mkl.10.0
# set -DRPROMU_DGEMV  -DRACCMU_DGEMV in the CPP lines
#BLAS=-L/opt/intel/mkl100/lib/em64t -lmkl -lpthread
# even faster for VASP Kazushige Goto's BLAS
# http://www.cs.utexas.edu/users/kgoto/signup_first.html
# parallel goto version requires sometimes -libverbs
#BLAS=  /opt/libs/libgoto/libgoto.so
# LAPACK, simplest use vasp.5.lib/lapack_double
#LAPACK= ../vasp.5.lib/lapack_double.o
# use the mkl Intel lapack
#LAPACK= -lmkl_lapack
#-----------------------------------------------------------------------
#MKLROOT = /applic/compilers/intel/11.1/mkl
#BLAS    = -L$(MKLROOT)/lib/em64t -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lpthread -lm
#BLAS    = -L/applic/compilers/intel/2013/applib1/LAPACK -lblas -llapack
LAPACK    = -L/applic/compilers/intel/2013/applib1/ACML/ifort64/lib -lacml
#LAPACK = -L/applic/compilers/intel/2013/applib1/LAPACK -llapack
#LAPACK = /applic/compilers/intel/2013/applib1/LAPACK/liblapack.a
#LAPACK = -L$(MKLROOT)/lib/em64t -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lpthread -lm
```

```
LIB  = -L../vasp.5.lib -ldmy \
       ../vasp.5.lib/linpack_double.o $(LAPACK) -lm
# options for linking, nothing is required (usually)
LINK    =
#-----------------------------------------------------------------------
# fft libraries:
# VASP.5.2 can use fftw.3.1.X (http://www.fftw.org)
# since this version is faster on P4 machines, we recommend to use it
#-----------------------------------------------------------------------
FFT3D                                =          fftmpiw.o          fftmpi_map.o          fftw3d.o          fft3dlib.o
/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/lib/libfftw3.a
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D   = fftw3d.o fft3dlib.o   /opt/libs/fftw-3.1.2/lib/libfftw3.a
#=======================================================================
# MPI section, uncomment the following lines until
#     general  rules and compile lines
# presently we recommend OPENMPI, since it seems to offer better
# performance than lam or mpich
#
# !!! Please do not send me any queries on how to install MPI, I will
# certainly not answer them !!!!
#=======================================================================
#-----------------------------------------------------------------------
# fortran linker for mpi
#-----------------------------------------------------------------------
#FC=mpif77
#FCL=$(FC)
#-----------------------------------------------------------------------
# additional options for CPP in parallel version (see also above):
# NGZhalf                 charge density   reduced in Z direction
# wNGZhalf                gamma point only reduced in Z direction
# scaLAPACK                use scaLAPACK (usually slower on 100 Mbit Net)
# avoidalloc             avoid ALLOCATE if possible
# PGF90                  work around some for some PGF90 / IFC bugs
# CACHE_SIZE             1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV           use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV           use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                  MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#CPP     = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
#       -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
#       -DMPI_BLOCK=8000 -Duse_collective
##      -DRPROMU_DGEMV  -DRACCMU_DGEMV
#-----------------------------------------------------------------------
# location of SCALAPACK
# if you do not use SCALAPACK simply leave that section commented out
#-----------------------------------------------------------------------
#BLACS=$(HOME)/archives/SCALAPACK/BLACS/
#SCA_=$(HOME)/archives/SCALAPACK/SCALAPACK
#SCA=  $(SCA_)/libscalapack.a  \
#                $(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a                $(BLACS)/LIB/blacs_MPI-LINUX-0.a
$(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a
SCA=
#-----------------------------------------------------------------------
# libraries for mpi
#-----------------------------------------------------------------------
#LIB      = -L../vasp.5.lib -ldmy  \
#        ../vasp.5.lib/linpack_double.o $(LAPACK) \
#        $(SCA) $(BLAS)
# FFT: fftmpi.o with fft3dlib of Juergen Furthmueller
#FFT3D   = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
```

```
  # alternatively: fftw.3.1.X is slighly faster and should be used if available
  #FFT3D    = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o   /opt/libs/fftw-3.1.2/lib/libfftw3.a
  #-----------------------------------------------------------------------
  # general rules and compile lines
  #-----------------------------------------------------------------------
  BASIC=    symmetry.o symlib.o    lattlib.o   random.o
  SOURCE=   base.o       mpi.o        smart_allocate.o        xml.o   \
            constant.o jacobi.o    main_mpi.o   scala.o    \
            asa.o        lattice.o  poscar.o    ini.o   mgrid.o   xclib.o   vdw_nl.o   xclib_grad.o \
            radial.o    pseudo.o    gridq.o       ebs.o   \
            mkpoints.o wave.o       wave_mpi.o   wave_high.o   spinsym.o \
            $(BASIC)    nonl.o       nonlr.o      nonl_high.o dfast.o      choleski2.o \
            mix.o        hamil.o     xcgrad.o    xcspin.o      potex1.o    potex2.o   \
            constrmag.o cl_shift.o relativistic.o LDApU.o \
            paw_base.o metagga.o   egrad.o       pawsym.o     pawfock.o   pawlhf.o     rhfatm.o   hyperfine.o
paw.o    \
            mkpoints_full.o          charge.o    Lebedev-Laikov.o   stockholder.o dipol.o      pot.o \
            dos.o         elf.o         tet.o         tetweight.o hamil_rot.o \
            chain.o      dyna.o       k-proj.o      sphpro.o      us.o   core_rel.o \
            aedens.o     wavpre.o     wavpre_noio.o broyden.o \
            dynbr.o      hamil_high.o  rmm-diis.o reader.o     writer.o     tutor.o xml_writer.o \
            brent.o      stufak.o     fileio.o     opergrid.o stepver.o   \
            chgloc.o     fast_aug.o fock_multipole.o   fock.o   mkpoints_change.o sym_grad.o \
            mymath.o     internals.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
            nmr.o         pead.o       subrot.o     subrot_scf.o \
            force.o      pwlhf.o      gw_model.o optreal.o   steep.o      davidson.o   david_inner.o \
            electron.o rot.o   electron_all.o shm.o      pardens.o   paircorrection.o \
            optics.o     constr_cell_relax.o     stm.o      finite_diff.o elpol.o      \
            hamil_lr.o rmm-diis_lr.o   subrot_cluster.o subrot_lr.o \
            lr_helper.o hamil_lrf.o     elinear_response.o ilinear_response.o \
            linear_optics.o \
            setlocalpp.o   wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
            mlwf.o        ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
            local_field.o ump2.o   bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
            rmm-diis_mlr.o   linear_response_NMR.o wannier_interpol.o linear_response.o
  vasp: $(SOURCE) $(FFT3D) $(INC) main.o
            rm -f vasp
            $(FCL) -o vasp main.o   $(SOURCE)    $(FFT3D) $(LIB) $(LINK)
  makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
            $(FCL) -o makeparam   $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
  zgemmtest: zgemmtest.o base.o random.o $(INC)
            $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
  dgemmtest: dgemmtest.o base.o random.o $(INC)
            $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
  ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
            $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
  kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
            $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
  clean:
            -rm -f *.g *.f *.o *.L *.mod ; touch *.F
  main.o: main$(SUFFIX)
            $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c main$(SUFFIX)
  xcgrad.o: xcgrad$(SUFFIX)
            $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcgrad$(SUFFIX)
  xcspin.o: xcspin$(SUFFIX)
            $(FC) $(FFLAGS) $(INLINE)   $(INCS) -c xcspin$(SUFFIX)
  makeparam.o: makeparam$(SUFFIX)
            $(FC) $(FFLAGS)$(DEBUG)   $(INCS) -c makeparam$(SUFFIX)
  makeparam$(SUFFIX): makeparam.F main.F
  ## MIND: I do not have a full dependency list for the include
  # and MODULES: here are only the minimal basic dependencies
```

```
# if one strucuture is changed then touch_dep must be called
# with the corresponding name of the structure
#base.o: base.inc base.F
mgrid.o: mgrid.inc mgrid.F
constant.o: constant.inc constant.F
lattice.o: lattice.inc lattice.F
setex.o: setexm.inc setex.F
pseudo.o: pseudo.inc pseudo.F
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
# these special rules have been tested for ifc.11 and ifc.12 only
fft3dlib.o : fft3dlib.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpi.o : fftmpi.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpiw.o : fftmpiw.F
	$(CPP)
	$(FC) -FR -lowercase -O1 $(INCS) -c $*$(SUFFIX)
wave_high.o : wave_high.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
# the following rules are probably no longer required (-O3 seems to work)
wave.o : wave.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
paw.o : paw.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
cl_shift.o : cl_shift.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
```

makefile for vasp5.3 in tachyon2 with ACML

```
.SUFFIXES: .inc .f .f90 .F
#-----------------------------------------------------------------------
# Makefile for Intel Fortran compiler for Pentium/Athlon/Opteron
# bases systems
# we recommend this makefile for both Intel as well as AMD systems
# for AMD based systems appropriate BLAS and fftw libraries are
# however mandatory (whereas they are optional for Intel platforms)
## The makefile was tested only under Linux on Intel and AMD platforms
# the following compiler versions have been tested:
#  - ifc.7.1  works stable somewhat slow but reliably
#  - ifc.8.1  fails to compile the code properly
#  - ifc.9.1  recommended (both for 32 and 64 bit)
#  - ifc.10.1 partially recommended (both for 32 and 64 bit)
#               tested build 20080312 Package ID: l_fc_p_10.1.015
#               the gamma only mpi version can not be compiles
#               using ifc.10.1
#  - ifc.11.1 strongly recommended (we use this to compile vasp)
#               Build 20090630 Package ID: l_cprof_p_11.1.046
## it might be required to change some of library pathes, since
# LINUX installation vary a lot
# Hence check ***ALL*** options in this makefile very carefully
#-----------------------------------------------------------------------
## BLAS must be installed on the machine
# there are several options:
# 1) very slow but works:
#    retrieve the lapackage from ftp.netlib.org
#    and compile the blas routines (BLAS/SRC directory)
#    please use g77 or f77 for the compilation. When I tried to
#    use pgf77 or pgf90 for BLAS, VASP hang up when calling
#    ZHEEV  (however this was with lapack 1.1 now I use lapack 2.0)
# 2) more desirable: get an optimized BLAS
## the two most reliable packages around are presently:
# 2a) Intels own optimised BLAS (PIII, P4, PD, PC2, Itanium)
#       http://developer.intel.com/software/products/mkl/
#    this is really excellent, if you use Intel CPU's
## 2b) probably fastest SSE2 (4 GFlops on P4, 2.53 GHz, 16 GFlops PD,
#       around 30 GFlops on Quad core)
#    Kazushige Goto's BLAS
#    http://www.cs.utexas.edu/users/kgoto/signup_first.html
#    http://www.tacc.utexas.edu/resources/software/
##-----------------------------------------------------------------------
# all CPP processed fortran files have the extension .f90
SUFFIX=.f90
CPP      = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
      -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
      -DMPI_BLOCK=8000 -Duse_collective -DscaLAPACK
##     -DRPROMU_DGEMV  -DRACCMU_DGEMV
#-----------------------------------------------------------------------
```

```
# fortran compiler and linker
#-----------------------------------------------------------------------
FC=mpif90 -I/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/include
# fortran linker
FCL=$(FC)
#-----------------------------------------------------------------------
# whereis CPP ?? (I need CPP, can't use gcc with proper options)
# that's the location of gcc for SUSE 5.3
##  CPP_    =  /usr/lib/gcc-lib/i486-linux/2.7.2/cpp -P -C
## that's probably the right line for some Red Hat distribution:
##  CPP_    =  /usr/lib/gcc-lib/i386-redhat-linux/2.7.2.3/cpp -P -C
##  SUSE X.X, maybe some Red Hat distributions:
CPP_ =  ./preprocess <$*.F | /usr/bin/cpp -P -C -traditional >$*$(SUFFIX)
# this release should be fpp clean
# we now recommend fpp as preprocessor
# if this fails go back to cpp
#CPP_=fpp -f_com=no -free -w0 $*.F $*$(SUFFIX)
#-----------------------------------------------------------------------
# possible options for CPP:
# NGXhalf               charge density    reduced in X direction
# wNGXhalf              gamma point only reduced in X direction
# avoidalloc            avoid ALLOCATE if possible
# PGF90                 work around some for some PGF90 / IFC bugs
# CACHE_SIZE            1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV          use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV          use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                 MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
# general fortran flags  (there must a trailing blank on this line)
# byterecl is strictly required for ifc, since otherwise
# the WAVECAR file becomes huge
#-----------------------------------------------------------------------
FFLAGS =  -FR -lowercase -assume byterecl -heap-arrays
#-----------------------------------------------------------------------
# optimization
# we have tested whether higher optimisation improves performance
# -axK  SSE1 optimization,  but also generate code executable on all mach.
#       xK improves performance somewhat on XP, and a is required in order
#       to run the code on older Athlons as well
# -xW    SSE2 optimization
# -axW  SSE2 optimization,  but also generate code executable on all mach.
# -tpp6 P3 optimization
# -tpp7 P4 optimization
#-----------------------------------------------------------------------
```

```
# ifc.9.1, ifc.10.1 recommended
OFLAG=-O2 -ip
OFLAG_HIGH = $(OFLAG)
OBJ_HIGH =
OBJ_NOOPT =
DEBUG  = -FR -O0
INLINE = $(OFLAG)
#-----------------------------------------------------------------------
# the following lines specify the position of BLAS  and LAPACK
# VASP works fastest with the libgoto library
# so that's what we recommend
#-----------------------------------------------------------------------
# mkl.10.0
# set -DRPROMU_DGEMV  -DRACCMU_DGEMV in the CPP lines
#BLAS=-L/opt/intel/mkl100/lib/em64t -lmkl -lpthread
# even faster for VASP Kazushige Goto's BLAS
# http://www.cs.utexas.edu/users/kgoto/signup_first.html
# parallel goto version requires sometimes -libverbs
#BLAS=   /opt/libs/libgoto/libgoto.so
# LAPACK, simplest use vasp.5.lib/lapack_double
#LAPACK= ../vasp.5.lib/lapack_double.o
# use the mkl Intel lapack
#LAPACK= -lmkl_lapack
#-----------------------------------------------------------------------
#MKLROOT = /applic/compilers/intel/11.1/mkl
#BLAS    = -L$(MKLROOT)/lib/em64t -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lpthread -lm
#BLAS    = -L/applic/compilers/intel/11.1/applib1/LAPACK -lblas -llapack
#LAPACK = /applic/compilers/intel/11.1/applib1/LAPACK/liblapack.a
#LAPACK = -L$(MKLROOT)/lib/em64t -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lpthread -lm
INTEL = /applic/compilers/intel/2013
LAPACK = $(INTEL)/applib1/ACML/ifort64/lib/libacml.a
SCA = $(INTEL)/mpi/openmpi/1.4.2/applib2/SCALAPACK/libscalapack.a
BLACS = $(INTEL)/mpi/openmpi/1.4.2/applib2/BLACS/libmpiblacsF77init.a \
      $(INTEL)/mpi/openmpi/1.4.2/applib2/BLACS/libmpiblacs.a \
      $(INTEL)/mpi/openmpi/1.4.2/applib2/BLACS/libmpiblacsF77init.a
LIB  = -L../vasp.5.lib -ldmy \
      ../vasp.5.lib/linpack_double.o $(BLAS) $(SCA) $(BLACS) $(LAPACK) -lm
# options for linking, nothing is required (usually)
LINK    =
#-----------------------------------------------------------------------
# fft libraries:
# VASP.5.2 can use fftw.3.1.X (http://www.fftw.org)
# since this version is faster on P4 machines, we recommend to use it
#-----------------------------------------------------------------------
FFT3D                  =  fftmpiw.o  fftmpi_map.o  fftw3d.o  fft3dlib.o
/applic/compilers/intel/2013/mpi/openmpi/1.4.2/applib2/FFTW3/lib/libfftw3.a
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D   = fftw3d.o fft3dlib.o  /opt/libs/fftw-3.1.2/lib/libfftw3.a
#=======================================================================
```

```
# MPI section, uncomment the following lines until
#      general  rules and compile lines
# presently we recommend OPENMPI, since it seems to offer better
# performance than lam or mpich
#
# !!! Please do not send me any queries on how to install MPI, I will
# certainly not answer them !!!!
#=======================================================================
#-----------------------------------------------------------------------
# fortran linker for mpi
#-----------------------------------------------------------------------
#FC=mpif77
#FCL=$(FC)
#-----------------------------------------------------------------------
# additional options for CPP in parallel version (see also above):
# NGZhalf                 charge density    reduced in Z direction
# wNGZhalf                gamma point only reduced in Z direction
# scaLAPACK                use scaLAPACK (usually slower on 100 Mbit Net)
# avoidalloc              avoid ALLOCATE if possible
# PGF90                   work around some for some PGF90 / IFC bugs
# CACHE_SIZE              1000 for PII,PIII, 5000 for Athlon, 8000-12000 P4, PD
# RPROMU_DGEMV            use DGEMV instead of DGEMM in RPRO (depends on used BLAS)
# RACCMU_DGEMV            use DGEMV instead of DGEMM in RACC (depends on used BLAS)
# tbdyn                   MD package of Tomas  Bucko
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#CPP     = $(CPP_) -DMPI  -DHOST=\"LinuxIFC\" -DIFC \
#       -DCACHE_SIZE=4000 -DPGF90 -Davoidalloc -DNGZhalf \
#       -DMPI_BLOCK=8000 -Duse_collective
##      -DRPROMU_DGEMV   -DRACCMU_DGEMV
#-----------------------------------------------------------------------
# location of SCALAPACK
# if you do not use SCALAPACK simply leave that section commented out
#-----------------------------------------------------------------------
#BLACS=$(HOME)/archives/SCALAPACK/BLACS/
#SCA_=$(HOME)/archives/SCALAPACK/SCALAPACK
#SCA= $(SCA_)/libscalapack.a  \
#               $(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a              $(BLACS)/LIB/blacs_MPI-LINUX-0.a
$(BLACS)/LIB/blacsF77init_MPI-LINUX-0.a
SCA=
#-----------------------------------------------------------------------
# libraries for mpi
#-----------------------------------------------------------------------
#LIB       = -L../vasp.5.lib -ldmy  \
#         ../vasp.5.lib/linpack_double.o $(LAPACK) \
#         $(SCA) $(BLAS)
# FFT: fftmpi.o with fft3dlib of Juergen Furthmueller
#FFT3D    = fftmpi.o fftmpi_map.o fft3dfurth.o fft3dlib.o
# alternatively: fftw.3.1.X is slighly faster and should be used if available
#FFT3D    = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o   /opt/libs/fftw-3.1.2/lib/libfftw3.a
```

```
#-----------------------------------------------------------------------
# general rules and compile lines
#-----------------------------------------------------------------------
BASIC=  symmetry.o symlib.o  lattlib.o  random.o
SOURCE=  base.o  mpi.o  smart_allocate.o  xml.o  \
         constant.o jacobi.o  main_mpi.o  scala.o  \
         asa.o  lattice.o  poscar.o  ini.o  mgrid.o  xclib.o  vdw_nl.o  xclib_grad.o \
         radial.o  pseudo.o  gridq.o  ebs.o  \
         mkpoints.o wave.o  wave_mpi.o  wave_high.o  spinsym.o \
         $(BASIC)  nonl.o  nonlr.o  nonl_high.o dfast.o  choleski2.o \
         mix.o  hamil.o  xcgrad.o  xcspin.o  potex1.o  potex2.o  \
         constrmag.o cl_shift.o relativistic.o LDApU.o \
         paw_base.o  metagga.o  egrad.o  pawsym.o  pawfock.o  pawlhf.o  rhfatm.o
hyperfine.o paw.o  \
         mkpoints_full.o  charge.o  Lebedev-Laikov.o  stockholder.o dipol.o  pot.o \
         dos.o  elf.o  tet.o  tetweight.o hamil_rot.o \
         chain.o  dyna.o  k-proj.o  sphpro.o  us.o  core_rel.o \
         aedens.o  wavpre.o  wavpre_noio.o broyden.o \
         dynbr.o  hamil_high.o  rmm-diis.o reader.o  writer.o  tutor.o xml_writer.o \
         brent.o  stufak.o  fileio.o  opergrid.o stepver.o  \
         chgloc.o  fast_aug.o fock_multipole.o  fock.o  mkpoints_change.o sym_grad.o \
         mymath.o  internals.o dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o \
         nmr.o  pead.o  subrot.o  subrot_scf.o \
         force.o  pwlhf.o  gw_model.o optreal.o  steep.o  davidson.o  david_inner.o \
         electron.o rot.o  electron_all.o shm.o  pardens.o  paircorrection.o \
         optics.o  constr_cell_relax.o  stm.o  finite_diff.o elpol.o  \
         hamil_lr.o rmm-diis_lr.o  subrot_cluster.o subrot_lr.o \
         lr_helper.o hamil_lrf.o  elinear_response.o ilinear_response.o \
         linear_optics.o \
         setlocalpp.o  wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o \
         mlwf.o  ratpol.o screened_2e.o wave_cacher.o chi_base.o wpot.o \
         local_field.o ump2.o  bse_te.o bse.o acfdt.o chi.o sydmat.o dmft.o \
         rmm-diis_mlr.o  linear_response_NMR.o wannier_interpol.o linear_response.o
vasp: $(SOURCE) $(FFT3D) $(INC) main.o
        rm -f vasp
        $(FCL) -o vasp main.o  $(SOURCE)  $(FFT3D) $(LIB) $(LINK)
makeparam: $(SOURCE) $(FFT3D) makeparam.o main.F $(INC)
        $(FCL) -o makeparam  $(LINK) makeparam.o $(SOURCE) $(FFT3D) $(LIB)
zgemmtest: zgemmtest.o base.o random.o $(INC)
        $(FCL) -o zgemmtest $(LINK) zgemmtest.o random.o base.o $(LIB)
dgemmtest: dgemmtest.o base.o random.o $(INC)
        $(FCL) -o dgemmtest $(LINK) dgemmtest.o random.o base.o $(LIB)
ffttest: base.o smart_allocate.o mpi.o mgrid.o random.o ffttest.o $(FFT3D) $(INC)
        $(FCL) -o ffttest $(LINK) ffttest.o mpi.o mgrid.o random.o smart_allocate.o base.o $(FFT3D)
$(LIB)
kpoints: $(SOURCE) $(FFT3D) makekpoints.o main.F $(INC)
        $(FCL) -o kpoints $(LINK) makekpoints.o $(SOURCE) $(FFT3D) $(LIB)
clean:
        -rm -f *.g *.f *.o *.L *.mod ; touch *.F
```

```
main.o: main$(SUFFIX)
	$(FC) $(FFLAGS)$(DEBUG)  $(INCS) -c main$(SUFFIX)
xcgrad.o: xcgrad$(SUFFIX)
	$(FC) $(FFLAGS) $(INLINE)  $(INCS) -c xcgrad$(SUFFIX)
xcspin.o: xcspin$(SUFFIX)
	$(FC) $(FFLAGS) $(INLINE)  $(INCS) -c xcspin$(SUFFIX)
makeparam.o: makeparam$(SUFFIX)
	$(FC) $(FFLAGS)$(DEBUG)  $(INCS) -c makeparam$(SUFFIX)
makeparam$(SUFFIX): makeparam.F main.F
## MIND: I do not have a full dependency list for the include
# and MODULES: here are only the minimal basic dependencies
# if one strucuture is changed then touch_dep must be called
# with the corresponding name of the structure
#base.o: base.inc base.F
mgrid.o: mgrid.inc mgrid.F
constant.o: constant.inc constant.F
lattice.o: lattice.inc lattice.F
setex.o: setexm.inc setex.F
pseudo.o: pseudo.inc pseudo.F
mkpoints.o: mkpoints.inc mkpoints.F
wave.o: wave.F
nonl.o: nonl.inc nonl.F
nonlr.o: nonlr.inc nonlr.F
$(OBJ_HIGH):
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG_HIGH) $(INCS) -c $*$(SUFFIX)
$(OBJ_NOOPT):
	$(CPP)
	$(FC) $(FFLAGS) $(INCS) -c $*$(SUFFIX)
fft3dlib_f77.o: fft3dlib_f77.F
	$(CPP)
	$(F77) $(FFLAGS_F77) -c $*$(SUFFIX)
.F.o:
	$(CPP)
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
.F$(SUFFIX):
	$(CPP)
$(SUFFIX).o:
	$(FC) $(FFLAGS) $(OFLAG) $(INCS) -c $*$(SUFFIX)
# special rules
#-----------------------------------------------------------------------
# these special rules have been tested for ifc.11 and ifc.12 only
fft3dlib.o : fft3dlib.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
fft3dfurth.o : fft3dfurth.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftw3d.o : fftw3d.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
```

```
fftmpi.o : fftmpi.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
fftmpiw.o : fftmpiw.F
	$(CPP)
	$(FC) -FR -lowercase -O1 $(INCS) -c $*$(SUFFIX)
wave_high.o : wave_high.F
	$(CPP)
	$(FC) -FR -lowercase -O1 -c $*$(SUFFIX)
# the following rules are probably no longer required (-O3 seems to work)
wave.o : wave.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
paw.o : paw.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
cl_shift.o : cl_shift.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
us.o : us.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
LDApU.o : LDApU.F
	$(CPP)
	$(FC) -FR -lowercase -O2 -c $*$(SUFFIX)
```