Installation of Wien2k, parallelization, large scale applications with WIEN2k

P. Blaha

Technical University of Vienna, Austria

WIEN2k- hardware

- WIEN2k runs on any Linux platform from PCs, Macs, workstations, clusters to supercomputers
- Intel **I7/I9** 6- or 8-core processors with **fast** memory bus (2-8 Gb/core, Gbit-network, 2-4 TB disk). 1000-1500€ /PC,
  - *with a few such PCs you have a quite powerful cluster (k-parallel)*
  - ***60 - 100 atom / cell**, requires 2-4 Gb RAM/core*
- Cluster of Intel Xeon/AMD based nodes with infiniband (too many cores per node useless because of memory access)
  - *mpi, Scalapack, **ELPA**, FFTW*
  - ***up to 1000 atoms/cell***
- Fortran90 (dynamical allocation, modules)
  - *real/complex version (inversion)*
  - *many individual modules, linked together with C-shell or perl-scripts*
- *web-based GUI – w2web (perl)*

Required / optional software

- f90 compiler: installation support for **ifort**(+slurm)/gfortran
- BLAS-library: **mkl**, (openblas+gcc 6.x) - most important for speed-up
  - ***mpi*** + ***Scalapack(mkl)*** + ***ELPA*** + ***FFTW*** *(for mpi-parallel version)*
- Linux utilities (not always installed by default)
  - *tcsh, bc, perl5, ghostscript, gnuplot (png support), pdf-reader*
  - *octave (structeditor)*
  - *python 3.x, numpy (BerryPI)*
  - *opendx (3D-plotting of NMR currents,…)*
- **Xcrysden** or/and **VESTA** (structure visualization, plotting)
- DFTD3/4 (van der Waals bonding)
- LIBXC: (https://libxc.gitlab.io/)
- Wannier90, PHONOPY
- "unsupported software" (see www.wien2k.at; phonopy, boltztrap2, fold2bloch, Skeaf, critic2, …)

Installation of WIEN2k

- Register via http://www.wien2k.at
- Create your $WIENROOT directory (e.g. ./WIEN2k)
- Download wien2k_XX.tar and examples (executables)
- Uncompress and expand all files using:
  - *tar –xvf wien2k_XX.tar*
  - *gunzip *.gz*
  - *./expand_lapw*
- This leads to the following directories:
  - *./SRC* *(scripts, ug.ps)*
  - *./SRC_aim* *(programs)*
  - *…*
  - *SRC_templates* *(example inputs)*
  - *…*
  - *SRC_usersguide_html* *(HTML-version of UG)*
  - *example_struct_files* *(examples)*
  - *TiC a*
- check_minimal_software_requirements.sh
  - *if errors occur, it does NOT make sense to continue*

siteconfig_lapw

- siteconfig_lapw to compile programs

```
    *                                W I E N                                *
    *                        site configuration                             *
    ***********************************************************************
       S   specify a system
       C   specify compiler
       O   specify compiler options, BLAS and LAPACK
       P   configure Parallel execution
       D   Dimension Parameters
       R   Compile/Recompile
       U   Update a package
       L   Perl path (if not in /usr/bin/perl)
       Q   Quit
```

D: define NMATMAX (adjust to your hardware/paging!):
NMATMAX=29000 ➔ 10Gb (real) or 20Gb (complex) ➔ ~100 atoms/unitcell
NUME=5000 ➔ number of eigenvalues (adjust to NMATMAX)

Compilation

- recommendation: **Intels ifort compiler** (includes **mkl, mpi**), can also be used for mpi compilation. free, now called "OneApi", www.intel.com
  - `which ifort` → *tells you if you can use ifort and which version you have*
    - usually installed in **/opt/intel/..../bin/intel64 (ls ....)**
    - **include compilervars.csh** (mklvars.csh) in your .bashrc/.cshrc file:
      - source /opt/intel/compilers_and_libraries_2017/linux/bin/compilervars.sh intel64
      - computing centers often use the "module" system: module avail; module load xxx
    - siteconfig has **default** options and libraries which should work for any modern **ifort** version
      - -O (-O1 in buggy versions); -traceback (to get line numbers for runtime errors)
      - -FR (free format); -assume buffered_io, -qopenmp
      - ifort versions are often buggy !!
- alternative: **gfortran** + **openblas** (at least **gcc** 6.x)
  - free
  - in sequential mode as fast as ifort+mkl (depends on hardware)
  - more complicated for mpi parallel version ??

compilation

- siteconfig has support for:
  - *ifort (LI)*
  - *ifort + SLURM batch system (LS)*
  - *gfortran (LG)*
  - *the standard siteconfig-options should work without modification for sequential (+ k-point parallel) compilation, once you have the proper **fftw** libraries*
  - *mpi installation requires that you **know** your mpi+scalapack+ELPA+fftw*
    - siteconfig can **search** for scalapack/fftw/libxc
- if you have no compiler, you can use the precompiled executables

installation of fftw (libxc, elpa, ...)

- install from your Linux distribution (only for gfortran !!!)
- install from http://www.fftw.org/download.html
  - *unzip and untar the downloaded file*
  - *Change into the expanded directories and configure the compilation.*
  - *Define your fortran compiler in the variables F77 (export F77=ifort)*
  - *./configure −−prefix=/installation-pathname −−enable-openmp" to configure compilation.*
  - *If you also want to use the mpi-version of WIEN2k, add the "−−enable-mpi" switch to the line above.*
  - *make*
  - *make install (if you specified a "system-directory" like /usr/local/fftw3 or /opt/fftw3 you must have proper permissions for this step, eg. become root user)*

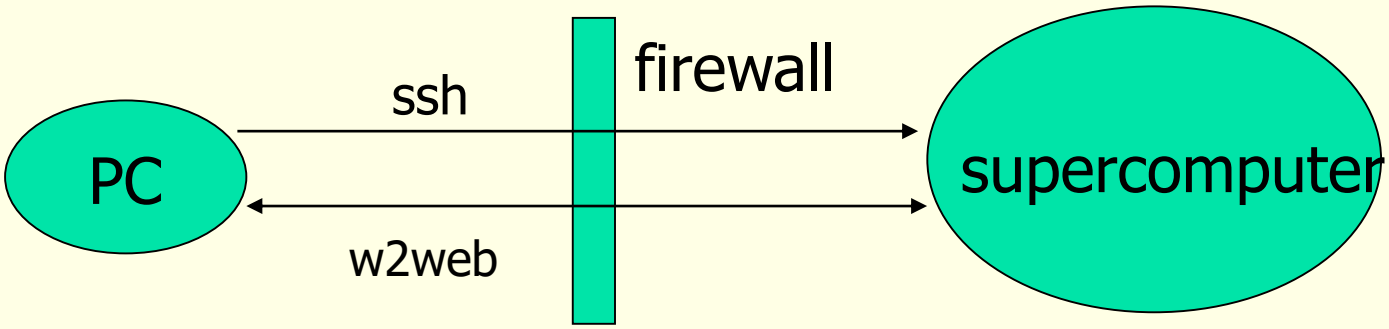
- for libxc or elpa see UG or the corresponding README files

userconfig_lapw

- Every user must run userconfig_lapw (setup of environment)
  - *support for tcsh and bash (requires .cshrc or .bashrc)*
  - *sets **PATH** to $WIENROOT, sets several variables and aliases*
    - $WIENROOT, $SCRATCH, $EDITOR, $PDFREADER, $STRUCTEDIT_PATH
    - **SCRATCH** directory (stores large files on local disks): **/scratch** or **./**
    - *pslapw: ps –ef | grep lapw*
    - *lsi: ls –als *.in** *lso: ls -als *.output**
    - *lss: ls –als *.scf** *lsc: ls -als *.clm**
- edit directly your .bashrc (.cshrc) file:
  - $OMP_NUM_THREADS = 1, **2** or 4 (for openmp+mkl shared mem. parallel)
  - set a suitable prompt: hostname:dir (export PS1='\h:$PWD>')
  - $LD_LIBRARY_PATH (on some systems)
  - source ifort configuration (if not done by system admin)
  - include configurations (VARIABLES and PATH) for optional products (XCRYSDEN, PYTHON, PHONOPY, ...)

w2web

- w2web: acts as webserver on a userdefined (high) port.
  - *define user/password and port. (http://host.domain.xx:5000)*
  - *on remote system: ssh -X user@host; w2web*
  - *behind firewall create a „ssh-tunnel":*
    - **ssh -fNL 5000:host:5000 user@host**



  - *~/.w2web/hostname/conf/w2web.conf: (configuration file)*
    - deny=*.*.*.*
    - allow=128.130.134.* 128.130.142.10
    - define execution types: NAME=commands (eg.: batch=batch < %f)
  - *xcrysden requires valid X-Windows when w2web was started*
    - kill_w2web; restart w2web

openMP Parallelization

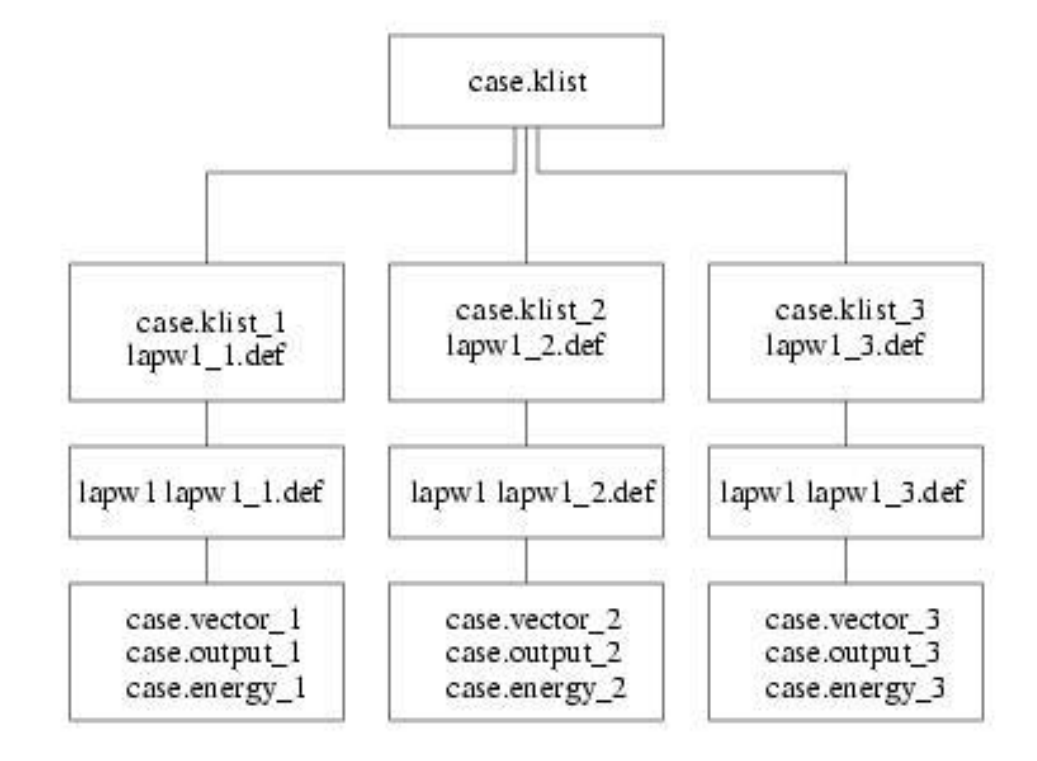
- very **efficient** on 2 (4) cores on a multicore PC (when it fits into memory) !
- requires **compilation** with openMP support (default), fftw-openmp
- export **OMP_NUM_THREADS=2** (in .bashrc file)
- .machines file:
  - *1:host1 (speed:hostname) for k-point parallelization*
  - *2:host2*
  - *....*
  - *# Uncomment for specific OMP-parallelization (overwriting a global OMP_NUM_THREADS)*
  - *#omp_global:4*
  - *# or use program-specific parallelization:*
  - *#omp_lapw0:4 # atoms, fftw*
  - *#omp_lapw1:4 # mkl efficiency limited to ~4*
  - *#omp_lapw2:4*
  - *#omp_lapwso:4*
  - *#omp_dstart:4 # atoms, PW*
  - *#omp_sumpara:4*
  - *#omp_nlvdw:4 # fftw*

k-point Parallelization (lapw1+lapw2)

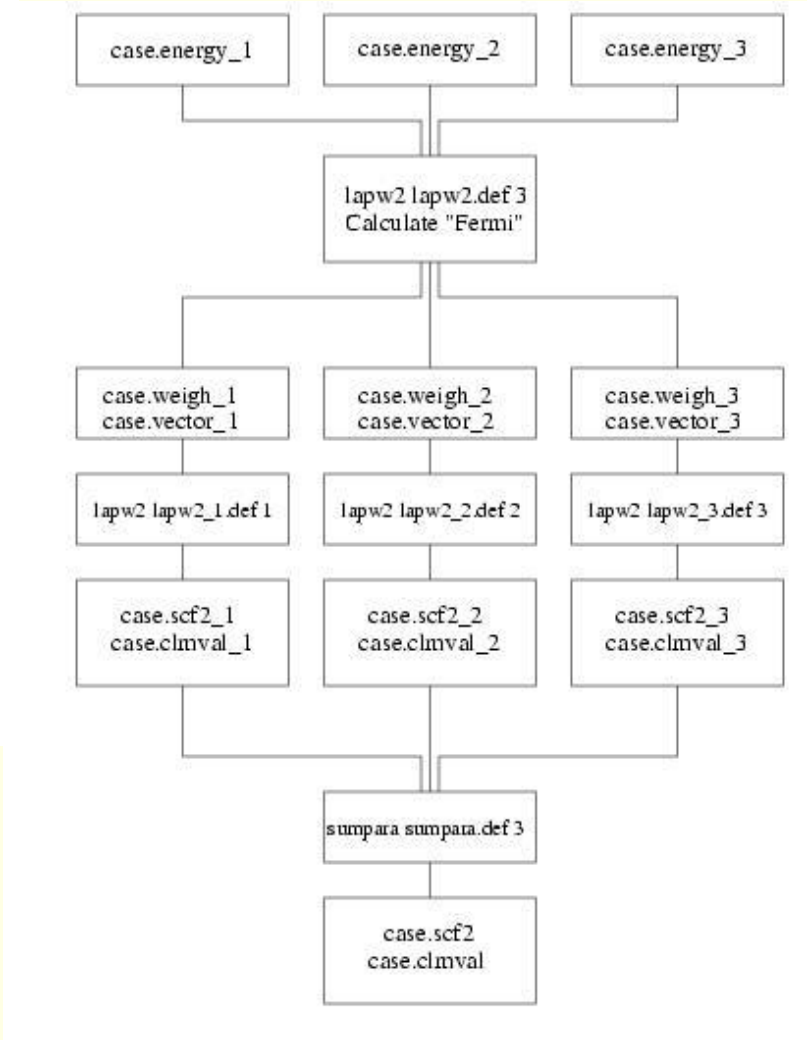
- very efficient parallelization even on loosely coupled PCs (slow network):
  - ***common NFS filesystem*** *(files must be accessible with the same path on all machines; use /host1 as data-directory on host1)*
  - ***ssh without password*** *(private/public keys)*
    - ssh-keygen –t rsa
    - append .ssh/authorized_keys on remote host with id_rsa.pub of local host
    - .machines file:
      - 1:host1 (speed:hostname)
      - 1:host1
      - 2:host2
      - ....
    - testpara (tests distribution); **run_lapw –p**
  - *cases must fit into memory of one PC !*
  - *high NFS load: use local $SCRATCH directory (beware of accidental overwriting); run_lapw -p -scratch /scratch/pblaha*
  - *couple with openMP parallelization*

Flow of parallel execution

lapw1para

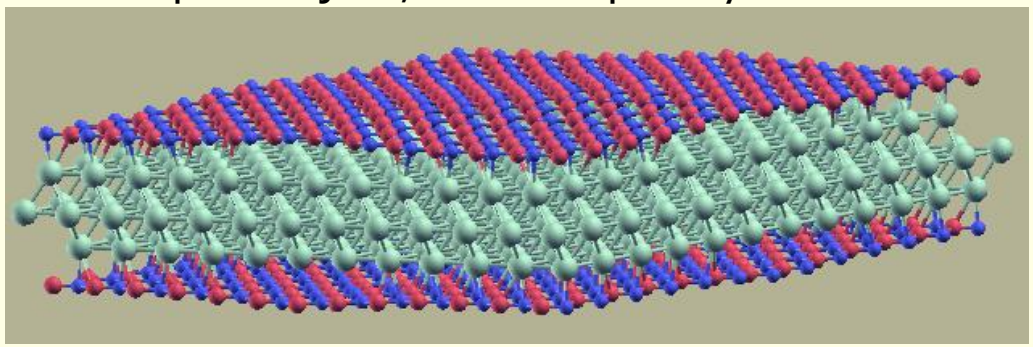


lapw2para



fine-grain mpi-parallelization

- for **bigger** cases (> 50 atoms) and **more** than **4 cores**
- fast network (~~Gbit~~, **Infiniband**, big shared memory machines)
- **mpi** (you need to know which mpi is installed (mpich, open-mpi, **intel-mpi**, ..)
  - ***mpif90*** *or* ***mpiifort***
- **scalapack** (included in ifort): **blacs-library** depends on your mpi!!
  - *llibmkl_blacs_lp64.a or libmkl_blacs_openmpi_lp64.a or libmkl_blacs_intelmpi_lp64.a*
- **FFTW** (version 3 ; mpi and sequ. version needed)
- **ELPA** (use most recent version; optional, but **faster** than scalapack)
- .machines file:
  - 1:host1:4 host2:4 — 8 mpi-parallel jobs on host1 and host2
  - lapw0:host1:4 host2:4 — 8 parallel jobs; atom-loops only + fft !!!
- simultaneous k-point and mpi-parallelization possible
  - *BN/Rh(111) nanomesh: cell with* ***1100*** *atoms*
    - NMAT=45000-100000; 64 cores, 30min / iteration; scales up to 1024 cores



case.dayfile

- check how your computer is performing:

```
>   lapw1  -p          (07:09:28) starting parallel lapw1 at Sat Jun 21 07:09:2
4 number_of_parallel_jobs
     ne(1) 197.017u 1.750s 1:46.71 186.2%      0+0k 0+119520io 0pf+0w
     ne(1) 198.383u 1.943s 1:47.88 185.6%      0+0k 0+105192io 0pf+0w
     eos(1) 188.838u 1.553s 1:49.79 173.4%      0+0k 17288+106456io 0pf+0w
     eos(1) 187.964u 1.849s 1:42.29 185.5%      0+0k 24+106872io 0pf+0w
>   lapw2 -p          (07:11:38) running LAPW2 in parallel mode
      ne 60.015u 0.621s 1:10.52 85.9% 0+0k 0+21088io 0pf+0w
      ne 60.686u 0.634s 1:08.63 89.3% 0+0k 0+17688io 0pf+0w
      eos 60.428u 0.689s 1:18.04 78.2% 0+0k 14152+17688io 0pf+0w
      eos 59.942u 0.598s 1:18.60 77.0% 0+0k 24+17696io 0pf+0w
```

OMP_NUM_TREADS=2

```
>   lapw1  -p          (09:11:14) starting parallel lapw1 at Mon Jun 23 09:11:14
4 number_of_parallel_jobs
     susi(1) 254.613u 2.783s 2:16.95 187.9%      0+0k 0+119736io 0pf+0w
     susi(1) 257.553u 3.650s 2:18.71 188.3%      0+0k 0+107144io 0pf+0w
     planck(1) 299.348u 2.369s 3:03.88 164.0%    0+0k 13760+109696io 0pf+0w
     planck(1) 303.426u 2.783s 3:05.92 164.6%    0+0k 1664+107616io 0pf+0w
>   lapw2 -p -vresp     (09:25:17) running LAPW2 in parallel mode
      susi 23.078u 0.562s 0:13.24 178.4% 0+0k 0+34984io 0pf+0w
      susi 25.343u 0.552s 0:14.23 181.9% 0+0k 0+31584io 0pf+0w
      planck 22.181u 0.491s 1:54.13 19.8% 0+0k 56+31608io 0pf+0w
      planck 22.334u 0.476s 1:53.93 20.0% 0+0k 88+31608io 0pf+0w
```

somebody else is using planck
or the network is overloaded
(local SCRATCH)

supercomputers (see also our faq-page)

- submit a script to a queuing system (PBS, SGE, SLURM, …)
- you can only specify total number of cores:
  - ***#$ -pe mpich 32*** *(specify 32 cores, but you don't know the hosts)*
- get the machine names and write **.machines** on the fly:

```
set mpisize_per_k=16
set proclist=`cat $hostfile_tacc`          #  this will be different on your computer
set nproc=`cat $hostfile_tacc | wc -l`     #  we have some templates on our
set i=1                                    #  faq-page (wien2k.at)
while ($i <= $nproc )
echo -n '1:' >>.machines
@ i1 = $i + $mpisize_per_k
@ i2 = $i1 - 1
echo $proclist[$i-$i2] ':1' >>.machines
set i=$i1
end
echo 'granularity:1' >>.machines
echo 'extrafine:1' >>.machines
```

- you can combine k- and mpi-parallelization ($mpisize_per_k)
  - *32 cores: 2 k-points, 16 mpi-jobs/k-point*

Getting help

- *_lapw –h „help switch“ of all WIEN2k-scripts
- help_lapw:
  - *opens usersguide.pdf; Use ^f keyword to search for an item („index")*
- html-version of the UG: ($WIENROOT/SRC_usersguide/usersguide.html)
- http://www.wien2k.at/reg_user
  - *FAQ page with answers to common questions*
  - *Update information: When you think the program has an error, please check newest version*
  - *Textbook section: DFT and the family of LAPW methods by S.Cottenier*
  - *Mailing-list:*
    - subscribe to the list (always use the same email)
    - full text search of the „digest“ (your questions may have been answered before)
    - posting questions: Provide sufficient information, locate your problem (case.dayfile, *.error, case.scf, case.outputX).
    - „My calculation crashed. Please help.“ This will most likely not be answered.

WIEN2k_23.2 (soon 24.1)

- always use latest version (**bug fixes**, improved performance, new features, **better** and **new utilities**)
  - *improved* ***siteconfig_lapw [-update OLD_W2k-dir]***
  - ***init_lapw –prec 0-3 !***
  - ***for further changes see the "update" section on wien2k.at***
- if you find a problem, please check the mailing list, maybe it has been fixed by now.
- a new Wien2k version is usually coming at least once a year
- references:
- **P. Blaha, K.Schwarz, F. Tran, R. Laskowski, G.K.H. Madsen and L.D. Marks, J. Chem. Phys. 152, 074101 (2020) and**
- P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka, J. Luitz, R. Laskowski, F. Tran and L. D. Marks, *WIEN2k*, An Augmented Plane Wave + Local Orbitals Program for Calculating Crystal Properties (Karlheinz Schwarz, Techn. Universität Wien, Austria), 2023. ISBN 3-9501031-1-2