

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** 4-,6- or 8-core processors with **fast memory bus** (2-4 Gb/core, Gbit-network, SATA disks). **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 based nodes with infiniband (**probably 2x16 cores per node best 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



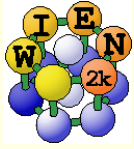
- **f90 compiler**: installation support for **ifort(+slurm)/gfortran**
- **BLAS-library: mkl**, (openblas+gcc 6.x) - most important for speed-up
 - *mpi + Scalapack(mkl) + ELPA + FFTW (for mpi-parallel version)*
- **Linux utilities** (not always installed by default)
 - *tcsh, perl5, ghostscript, gnuplot (png support), pdf-reader*
 - *octave (structeditor)*
 - *python 2.7.x, numpy (BerryPI)*
 - *opendx (3D-plotting of NMR currents,...)*
- **Xcrysden or/and VESTA** (structure visualization, plotting)
- DFTD3 (van der Waals bonding)
- LIBXC: (<http://www.tddft.org/programs/octopus/wiki/index.php/Libxc>)
- Wannier90, PHONOPY
- “unsupported software” (see www.wien2k.at; phonon, boltztrap, 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`
- `siteconfig_lapw` to compile programs (or: `tar -xvf SRC_executables.tar`)



siteconfig_lapw



- *****
- * 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=14000 → 2Gb (real) or 4Gb (complex) → 50-100 atoms/unitcell

NUME=1000 → number of eigenvalues (adjust to NMATMAX)



■ recommendation: **Intels Fortran compiler** (includes **mkl**)

not anymore free for non-commercial usage, 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`
- ***ifort 14 or later*** (*vers. 8.0, early 12.x are buggy and a bit different*)
 - **dynamic linking recommended** (depends on ifort version, requires system and compiler libraries at runtime, needs `$LD_LIBRARY_PATH`)
 - **Intel64** (em64t, IA32 bit, IA64 bit (Itanium) - version
 - **mkl-library: library-names change with every version, see:**
<http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor>
 - **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 !! (bug in reading unformatted files, `-O` crashes)



- **gfortran** + **openblas** + **new gcc 6.X** (gotolib, ATLAS-BLAS)
 - free
 - in sequential mode almost as fast as ifort+mkl (depends on hardware)
 - more **complicated** for mpi parallel version
- **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.*
 - *mpi installation requires that you **know** your mpi+scalapack+ELPA+fftw*
 - siteconfig can **search** for scalapack/fftw,...
- **if you have no compiler, you can use the precompiled executables**

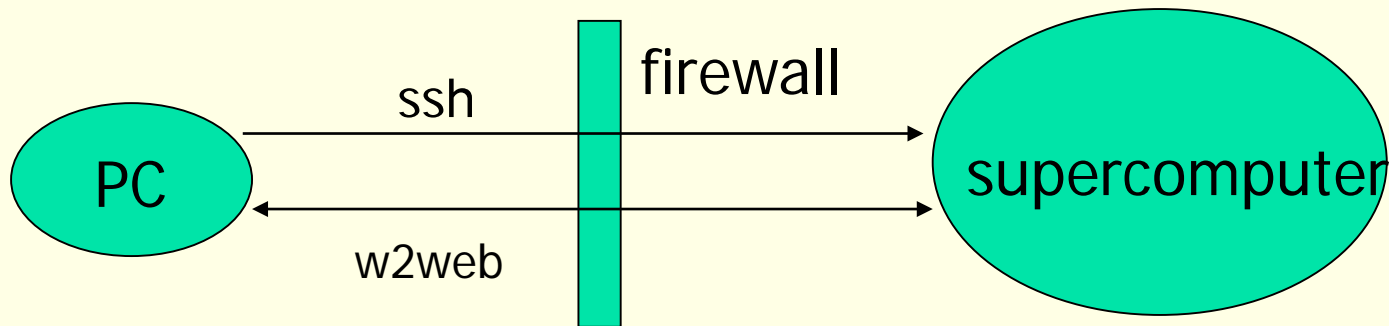


userconfig_lapw



- **Every user must run** `userconfig_lapw` (setup of environment)
 - *support for `tclsh` 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**: 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**



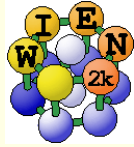
- `~/.w2web/hostname/conf/w2web.conf`: (configuration file)
 - `deny=*. *.*.*.*`
 - `allow=128.130.134.* 128.130.142.10`
 - define execution types: `NAME=commands` (eg.: `batch=batch < %f`)
- *xcrysdn* requires valid X-Windows when *w2web* was started
 - `kill_w2web`; restart `w2web`



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
- **\$OMP_NUM_THREADS=2** (in .bashrc file)
- export **OMP_NUM_THREADS=2**
- .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)



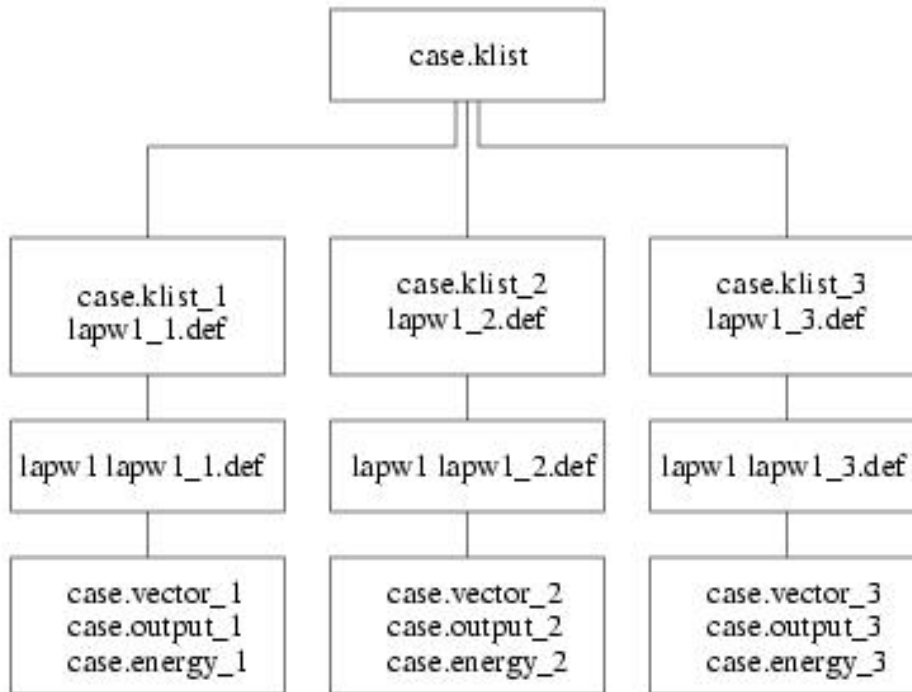
- **very efficient parallelization** even on loosely coupled PCs (**slow** network):
 - *common NFS filesystem* (files must be accessible with the same path on all machines; use **/host1** as data-directory on **host1**)
 - *ssh without password* (private/public keys)
 - ssh-keygen -t rsa
 - append **.ssh/authorized_keys** 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*



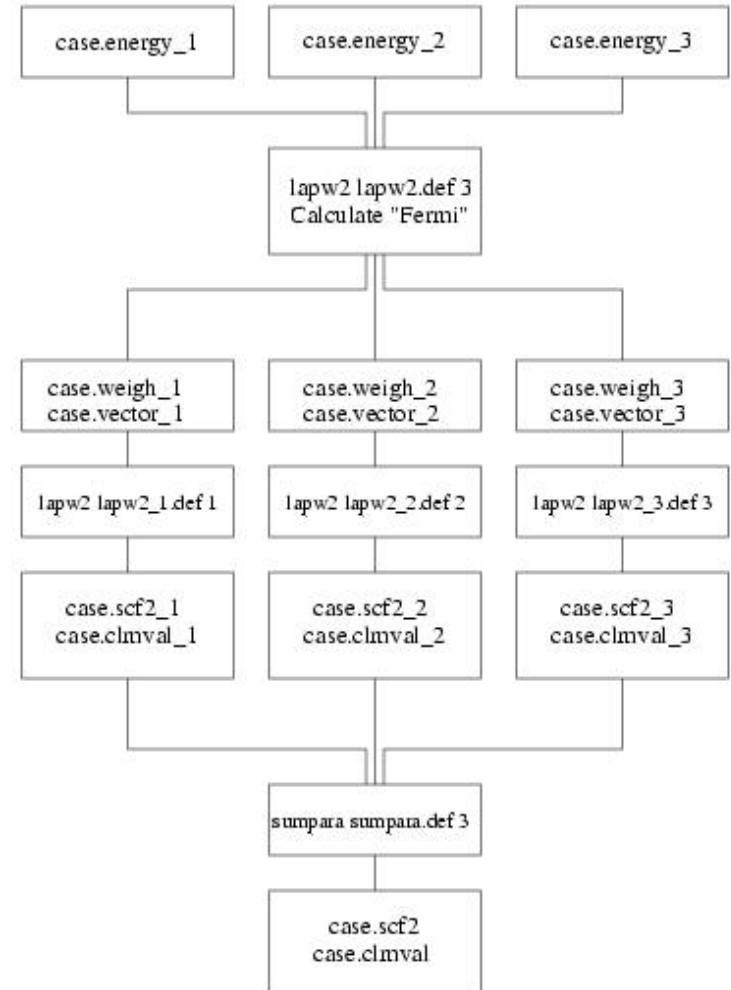
Flow of parallel execution



lapw1para



lapw2para





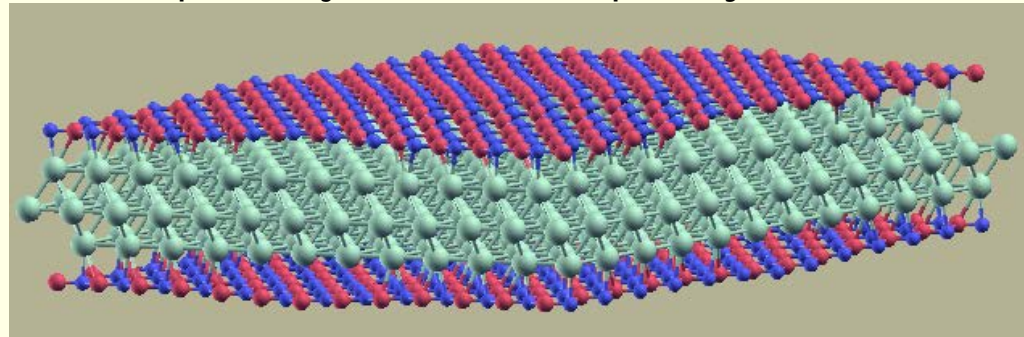
fine-grain mpi-parallelization



- 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!!
 - *libmkl_blacs_lp64.a* or *libmkl_blacs_openmpi_lp64.a* or *libmkl_blacs_intelmpi_lp64.a*
- **FFTW** (v. 2 or 3 ; mpi and sequ. version needed, -DFFTW2/3 in Makefiles)
- **ELPA** (at present use version 2018.11.001; optional, but **faster** than scalapack)
- .machines file:
 - 1:host1:4 host2:4
 - lapw0:host1:4 host2:4
- **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

8 mpi-parallel jobs on host1 and host2

8 parallel jobs; atom-loops only + fft !!!





case.dayfile



■ check how your computer is performing:

```
> lapw1 -p (07:09:28) starting parallel lapw1 at Sat Jun 21 07:09:28
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
```

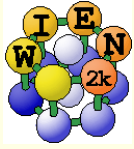
OMP_NUM_THREADS=2

```
> 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
```

```
> 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)



iterative diagonalization for surfaces:

■ `run_lapw -p -it (-noHinv # less I/O)`

cycle 1 (Thu Oct 31 07:20:53 CET 2013) (40/99 to go)

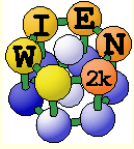
```
> lapw0 -p (07:20:53) starting parallel lapw0 at Thu Oct 31 07:20:53 2013
----- .machine0 : 64 processors
264.604u 21.742s 0:40.76 702.5% 0+0k 591784+49768io 369pf+0w
```

```
> lapw1 -up -p -orb (07:21:34) starting parallel lapw1 at Thu Oct 31
-> starting parallel LAPW1 jobs at Thu Oct 31 07:21:34 CET 2013
running LAPW1 in parallel mode (using .machines)
r09n30 r09n30 r09n30 ....
6.558u 1.796s 29:08.54 0.4% 0+0k 16+520io 0pf+0w
```

....
cycle 3 (Thu Oct 31 07:50:53 CET 2013) (40/99 to go)

```
...
> lapw1 -it -up -p -orb -noHinv (09:31:52) starting parallel lapw1 at ...
3.411u 0.908s 14:18.31 0.5% 0+0k 72+536io 0pf+0w
```

...



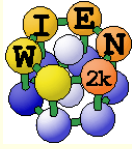
batch systems (see also our faq-page)



- 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
```

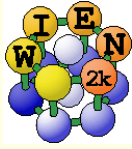
- you can combine k- and mpi-parallelization (`$mpisize_per_k`)
 - 32 cores: 2 k-points, 16 mpi-jobs/k-point



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.



- always use latest version (**bug fixes**, improved performance, new features, **better** and **new utilities**)
 - *improved siteconfig_lapw*
 - **better support for gfortran+openblas+gcc**
 - *openMP support for all time consuming steps !*
 - *lapw1mpi: new ELPA support !*
 - *non-local van der Waals option much faster*
 - *new PES module (valence band UPS/XPS)*
 - *new 3DDENS module (uses FFT, replacing prepare_xsf)*
 - *more efficient mixer*
- 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