Hessisches Kompetenzzentrum
für Hochleistungsrechnen

HKHLR–How To
run VASP on an HPC-Cluster

Version of 16 of January 2019

1 What is VASP and how can it be obtained

VASP is an acronym for "Vienna Ab Initio Simulation Package". It is a software package that is used for quantum mechanical (and parameter free – also known as first principles or *ab-inito*) atomic scale simulations of material properties (mostly solid state materials). VASP implements density functional theory (DFT) and extensions thereof such as GW or BSE. Common types of applications are electronic structure calculations or *ab-inito* molecular dynamics.

Some properties that can be computed are:

- electronic (quasi-particle) band structures and density of states
- optical properties such as the dielectric functions (in various approximations schemes)
- lattice dynamics (phonons)
- molecular dynamics simulations

A VASP license must be purchased from the VASP Software GmbH. For further information of how to obtain VASP please refer to the following website: `https://www.vasp.at/index.php/faqs`.

2 Compiling VASP

An introduction to the (new) VASP build system (as introduced with release version 5.4.1) can be found on the VASP website: `https://cms.mpi.univie.ac.at/wiki/index.php/The_VASP_Manual` (see section "Installation and Validation").

The build system will output three binary files named "vasp_std", "vasp_gam", and "vasp_ncl" for the standard, gamma-only, and the non-collinear version of VASP. These three versions are build generated by specifying certain preprocessor flags to the Fortran source files. In earlier versions of VASP these had to be given by hand but the new build system will do the job for you. The three versions can be generated by typing in the command line:

- `make std` (standard version),
- `make gam` (gamma-only version),
- `make ncl` (non-collinear version), or simply
- `make all` (or just `make`) for all versions to be build one after the other.

Note that the VASP makefiles generally do *not* support parallel compilation; hence things like e.g. `make -j4 std` will result in the compilation to break.

The gamma-only version of VASP is very useful for large systems where only one $\vec{k}$-point is needed for the simulation which then must be chosen to be the Γ-point at the centre of the Brillouin zone. Using this version instead of the standard version can result in significant speed up of the whole calculation. The non-collinear version of VASP must be used in case effects like spin-orbit coupling are of relevance.

3 Theoretical foundation and some technical details

An overview of the methods implemented in VASP and the underlying theory can be found on `https://cms.mpi.univie.ac.at/wiki/index.php/The_VASP_Manual`, in section "Getting started". Further, some very detailed video lectures/talks were given at NERSC (by Dr. Martijn Marsman, one of the developers of the VASP code); these lectures can be found on Youtube (search for "vasp workshop at nersc" to the a list of all lectures).

It is also recommended to have a look at the section "Input" where the files (INCAR, KPOINTS, POSCAR, POTCAR, ...) used by VASP are mentioned. This section also links to the list of input parameters that govern the type of calculations (as set in the INCAR file).

For some example calculations please refer to the section "Tutorials and Examples". Please note that some tutorials are yet to be put online and hence is not complete. Further, some of the INCAR files are rather minimalist and only contain the parameters strictly necessary in the calculation of current interest. In order to be consistent with other calculations (upon which the current calculation builds), you might have to add some parameters to your INCAR file.

4 Additional topics

4.1 Parallelism

The following bullet points provide a summary (with some additional comments) of the "Parallelisation" lecture given at NERSC by Dr. Martijn Marsman (refer to Youtube for the video lecture).

The following parallelisation levels are available in VASP:

- $\vec{k}$-points (optional)
  - was added rather recently
  - KPAR $\neq 1$ (should be a divisor of #MPI-ranks)
  - work over k-points is distributed over KPAR groups of MPI-ranks
  - $\vec{k}$-dependent data (wavefunctions) is **replicated** over the groups and hence the memory demand rises!!!
  - although each group of MPI-ranks only works on a subset of $\vec{k}$-points it holds the wavefunctions of all $\vec{k}$-points in memory

    $\rightarrow$ only makes sense if the extra memory can be afforded

    $\rightarrow$ it is strongly recommended to only use this parallelisation scheme in case compute nodes can be used *exclusively* (this setting might not be available on all HPC machines)
  - not much communication needed because $\vec{k}$-points are independent (Kohn-Sham (KS) equations only couple to other $\vec{k}$-points through the density); only at very few occasions information from other $\vec{k}$-points are needed
- bands (default parallelisation level; influenced by NBANDS)
  - bands and data are distributed over the MPI-ranks
  - each MPI-rank only holds the data for the bands it is supposed to work on
  - only works efficiently if the number of bands is "sufficiently large" (this for example is the case if your calculation involves many atoms)

- plane wave (PW) coefficients (lowest level of parallelisation)
  - PW coefficients that belong to one band are distributed over MPI-ranks
  - can be combined with parallelism over bands
  - controlled by NCORE parameter (#MPI-ranks that share the PW coefficients of a single band); as an alternative: use NPAR
  - can be used in case only a few bands are to be calculated but number of plane waves used to represent the band is large enough (the number of plane waves is governed by the input parameter ENCUT which is the cut-off energy for the plane wave expansion of the KS-orbitals)
  - meaning of NCORE is easier to understand than that of NPAR
  - default: NCORE $= 1 \rightarrow$ each band gets worked on by one MPI-rank (FFT is calculated locally within the scope of one MPI-rank)
  - if NCORE $\neq 1$: sets the number of MPI-ranks that share the PW coefficients of one band (FFT is calculated in parallel)

Recommendations for parallelisation parameters:

- In case you have some $\vec{k}$-points and the memory is available use KPAR (often a good choice is to set KPAR $=$ #number of nodes in since then - in case nodes can be used exclusively - the whole memory of a node is available)
- NCORE $\leq$ #cores/socket (in many cases nodes have two sockets); this can be useful for architectures where cores within one socket have very fast access to a memory associated with that socket but slower access to the memory associated with the other socket
- The parallel performance of the code might depend on the type of calculation you actually want to carry out. Hence, in order to obtain maximum the benefit of certain parallelisation schemes, a decent amount of testing will be required. In this context it might be worth to consider using the "Extra-P" (`http://www.scalasca.org/software/extra-p/download.html` developed by Alexandru Calotoiu. For the use, please contact us: staff@hpc-hessen.de) tool that allows you to estimate the scaling behaviour (strong scaling as well as week scaling) of a parallel application.

Remark

The present "How To" is based on the experience obtained in the HKHLR group. Any questions, suggestions and possible improvements are welcome as well as the feedback, if the guide is useful. Please, contact us by email: staff@hpc-hessen.de.