From crystal structure to properties of solids with the grid-enabled WIEN2k

Karlheinz Schwarz, Peter Blaha and Johannes Schweifer*

Abstract. *WIEN2k, a widely used application in materials science, is taken as a representative example for grid computing. The electronic structure of a given solid can be calculated using the program package WIEN2k on the basis of quantum mechanics using density functional theory (DFT). Once the DFT equations are solved the electronic wave functions and the corresponding energy states are known, from which various properties can be derived that are of importance for many materials science problems. A realistic simulation of modern materials requires large system sizes and thus significant computational power. A grid environment can provide a proper platform but must deal with the complexity of such an application. The newly developed W2GRID attempts to provide the necessary middleware with the desired functionality for WIEN2k.*

1. Introduction

In certain parts of materials sciences an atomistic approach becomes important, for example in electronic devices due to the increased miniaturization, for magnetic or optical storage media or sensors. In such cases the electronic structure dominates the properties of a given solid, its surface or chemical reactivity. The program package WIEN2k, which will be described further below, allows such studies on the basis of quantum mechanics using density functional theory (DFT). A materials scientist would like to focus on a problem of interest and does not want to consider the details, namely how and where the computations are done. A grid-enabled software should provide such a service. For the user it is desirable to have an automatic but transparent selection of computing resources. Such a scenario raises many challenges from finding efficient algorithms for parallel computing to developing stable middleware for the grid infrastructure. First steps are taken in this direction in order to make WIEN2k grid-enabled.

The complexity of this application shall be outlined below, where we start with general concepts of solids, describe the method of calculations with the mathematics involved, and explain the variety of properties of interest in materials sciences. This leads to an assessment of the computational needs and requirements that a grid infrastructure should support. A pragmatic solution is explored with W2GRID. A combination of experience from chemistry, physics, mathematics and computer sciences is needed to have success in this field.

*Institute for Materials Chemistry, Vienna University of Technology, Getreidemarkt 9/165-TC, A 1060 Vienna, Austria, email: kschwarz@theochem.tuwien.ac.at

2. Concepts of solids or condensed matter

As a first step a few concepts are listed below which set the scene for large-scale materials science computations. These details are needed in order to understand the complexity of the problem and to define the parameters that are crucial in various workflows that appear in WIEN2k applications.

2.1. Solids or condensed matter

We focus on an ideal crystal that consists of a well ordered periodic arrangement of atoms forming a solid. The translational symmetry and the periodic boundary conditions make it possible to just consider a single unit cell, which is (indefinitely) repeated in all three dimensions. The unit cell may contain only one atom as in an element, a trivial case that can easily be solved on a single laptop computer, or it may consist of up to several hundred atoms each having a nucleus and a certain number of electrons, a task leading to large-scale computations. The study of the electronic structure of such a system requires a quantum mechanical treatment. Such a complicated many-body problem can only be solved by making use of the translational symmetry, which cause the electronic wave functions to be of Bloch-type, labeled by a k-vector in reciprocal space, the quantum number of a solid. Thus the periodicity in real space is defined by the k-vector in reciprocal space, whose unit cell is called Brillouin zone (BZ). The latter becomes smaller the larger the real space unit cell gets.

2.2. Method of computation

The quantum mechanical treatment is provided by density functional theory (DFT), according to which the interacting electrons and nuclei can be represented by a series of one-electron Schrödinger-like (so called Kohn-Sham) partial differential equations [1]

$$[-\frac{1}{2}\nabla_i^2 + V_i]\Psi_i = \varepsilon_i \Psi_i. \tag{1}$$

In DFT each electron i moves in the field (potential) of the nuclei and all the other electrons. The effective potential V_i, in which a single electron moves, comes from the classical electrostatic Coulomb interaction (with the nuclei and the other electrons) as well as from the quantum mechanical exchange and correlation potential, which is a functional of the electron density. The corresponding partial differential equation is an eigenvalue problem with the eigenfunctions Ψ_i and the eigenvalues ε_i. This problem can only be solved iteratively, since the potential requires the knowledge of the density, but the density is computed from the sum $\Psi_i^*\Psi_i$ over all occupied electronic states i, which need the potential for obtaining the corresponding one-electron Kohn-Sham orbitals Ψ_i. This iterative scheme (see Fig. 1) is called self-consistent-field (SCF) cycle and is repeated until a fixed point is reached according to some convergence criterion. The details will be discussed later.

An approximate DFT solution can be obtained by constructing for orbital Ψ_i a trial wave function, which is often expanded in a basis set. In the present case we use a basis set of plane waves, each of which is augmented by atomic like functions around each atomic center. The linearized augmented plane wave (LAPW) method [2] is the basis of our WIEN2k program package [3] that was developed during the last 25 years. The energy expectation value computed with any trial wave function is an upper bound. That means the lower in energy one gets the closer one is to the (unknown) exact solution. According to the variational principle the energy can be minimized (to find the best solution

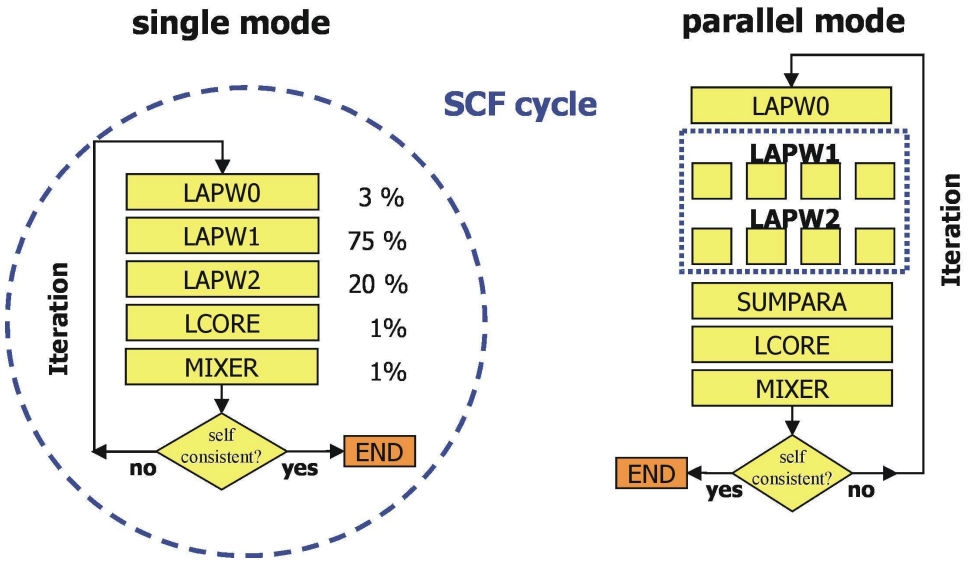

Figure 1. Self-consistent-field (SCF) cycle on (left panel) a single node (one CPU) or (right panel) in parallel mode distributing the k-points to different CPUs connected by a common file system. The fraction of the total computation time of a typical case is given in per cent. WIEN2k is composed of several independent programs, which are called one after the other in an SCF cycle. Since the whole program package with all its subprograms communicates with files, this scheme provides a high degree of freedom concerning the modification or replacement of certain parts. The single programs are linked with shell scripts. The figure represents two typical workflows.

for a given ansatz) by varying the coefficients of the basis set leading to a set of linear equations. In matrix notation this is the general eigenvalue problem HC=SCE, where H is the Hamiltonian, S the overlap matrix, and C the matrix containing the eigenvectors that specifies the relative weights of each plane wave for each of the eigenstates given in the diagonal matrix E.

2.3. Mathematical formalism and computational considerations

The SCF cycle starts with some input electron density (taken from atomic DFT calculations), from which the program LAPW0 calculates the corresponding potential of the crystal. The major computational effort (Fig. 1, left panel) comes from setting up the H and S matrices and solving the corresponding generalized eigenvalue problem for a series of k-vectors (Bloch states), which are given on a uniform mesh in the BZ. This is done in the module LAPW1. After diagonalization is completed for all the k-vectors, a synchronization step is needed, namely finding the Fermi energy, which separates the occupied from the unoccupied electronic states. Physically speaking all the electrons fill the lowest energy states so that a new electron density is obtained by summing (in LAPW2) the squared moduli $\Psi_i^*\Psi_i$ of the corresponding wave functions (Kohn-Sham orbitals) Ψ_i. This yields the corresponding valence electron density, to which the density from the core electrons (calculated in LCORE) must be added to obtain the total output density. The latter is used in MIXER with the densities of previous iterations to generate a new input density for the next SCF iteration. A sophisticated algorithm is implemented to minimize the number of SCF cycles. The SCF convergence is tested according to different criteria such as changes in total energy or charge density between the last iterations. If the convergence is better than the given threshold, the SCF cycle is stopped, otherwise the next cycle is started.

In the case of a parallel mode (Fig. 1, right panel), the set of k-points is distributed to different processors for both LAPW1 and LAPW2. In order to avoid too much communication (large eigenvector files) both of these programs should preferably be executed on the same processor for a given k-vector. At the end a summation of the densities that were computed on different CPUs must be done (in SUMPARA) to obtain the proper k-summation in the BZ. The other parts remain the same as in the single mode, since usually they cost less than 1 per cent of the CPU time. If the SCF cycle is split into smaller pieces, a significant overhead for various file transfers would occur, which should be avoided for an efficient use of resources.

In another scenario, namely when the matrices of the general eigenvalue problem do not fit into memory of a single CPU, the matrices must be distributed to several CPUs. In such a case another mathematical library (SCALAPACK) is required to solve the corresponding eigenvalue problem. That scheme has a lot of memory access to other processors and thus either a shared memory machine or at least fast communication (e.g. InfiniBand) is needed. Alternatively one can go for iterative eigensolvers which are better suited for distributed computing.

2.4. Materials science problems

The development of new modern materials (e.g. optical or magnetic storage devices, sensors or ferroelectrica, or smart materials like shape memory alloys) requires the knowledge of the electronic structure. The relation between the crystal structure, that is the arrangement of the atoms in the solid, and the electronic structure is vital for an understanding of a given material in terms of (mechanical, elastic, electric, optical, magnetic etc.) properties [4]. The systems of interest may vary from insulators, semiconductors, metals to magnetic or superconducting compounds. The solids (or surfaces) may contain light elements like hydrogen or heavy elements (like platinum for catalysis), for which relativistic effects must be included. In magnetic systems spin polarization must be taken into account. Therefore one is confronted with a high complexity due to the large variety in both, systems and properties, leading to modified work flows with respect to the two examples given above (Fig. 1).

3. Computational aspects and Grid

The application driven computational requirements vary by several orders of magnitude depending on the system and properties of interest. Consequently the computing resources that are needed to solve a given problem can be provided either on a single PC, on a medium sized PC-cluster, or may require a supercomputer (or a shared memory machine) or at least several CPUs that are tightly coupled with fast communication and high bandwidth (e.g. Myrinet or InfiniBand). From the input of a WIEN2k calculation an analytic performance model can roughly estimate the computer requirements. The selection of proper machines for a particular case is non-trivial for an unexperienced user and thus should be done automatically without user intervention, but made transparent. For high efficiency the WIEN2k program must be pre-installed on all machines that belong to the grid, since each hardware requires special compiler options and optimized mathematical libraries.

We follow the modern definition of grid computing: Grid computing is the distributed computing performed transparently across multiple administrative domains. Experience from other grid application projects has shown that the direct use of Globus tools places too heavy a burden on the application scientists [5]. However, they are willing to work with scripts or to edit them for job submission as long as these scripts do not contain special grid commands. This is one of the motivations to design middleware adapted to the needs of the user.

3.1. W2GRID

Recently Hayes et al. [6] suggested a lightweight grid services toolkit that is more likely to be used by applications groups, especially those whose programs are written in Fortran. The prototype client-side toolkit called GROWL (Grid Resources on Workstation Library) was developed at Daresbury Laboratory. It follows the plea for lightweight, usable middleware made by Chin and Conveney [7]. By the same guidelines we developed W2GRID basically from scratch. It is a perl-based lightweight middleware, that was designed to meet the special requirements of running large-scale WIEN2k jobs in a heterogeneous environment (see Fig. 2). At present it is available as a beta version. Given the fact that solids show a large variety in structure with a broad spectrum of properties, a computation with WIEN2k may significantly differ from case to case. Therefore a service for the materials science community must cope with this complexity and provide high functionality.

A typical WIEN2k user often has access to a pool of quite heterogeneous machines, on which only user-permission is granted and which belong to different administrative domains. The development of W2GRID aims to provide tools for the WIEN2k community to access and use such computing resources like a single uniform infrastructure. To meet this challenge we want to avoid the necessity of extended permissions (root access), which most of the common grid middleware like NetSolve [8], Condor [9],[10] or Unicore [11] require.

To run W2GRID on a certain node it usually is sufficient to have simple user permission. At present an ssh-login without password via public key authentication must be possible but this can be extended to a login via GSI (Grid Security Infrastructure) [12]. W2GRID does not explicitly require the presence of a queuing system but can benefit from it when it is available. In such a case the master gets the status information of all his compute nodes from the queuing system (e.g. PBS) and thus does not need the slave grid servers (the latter come into play only in systems without a queuing system). The infrastructure consists of several daemon processes as depicted in Fig. 2 and thus the user must be able to start them as user process.

W2GRID is built of a static core, which contains only the most basic routines to operate daemons and to provide the elemental functionality to interact with the operating system. All truly platform- and queue-specific functions are implemented by a plugin technology, which provides access to the uniquely configured resource, where W2GRID is installed. The plugins act as an interface between the W2GRID program-scripts and any kind of external program or systems call, which can not be known by the programmer in advance. An HPC application will need different system calls on different architectures, for example for submitting a job to a queuing systems (e.g. PBS or LoadLeveler, etc.). The program script, which is also written in perl, will just contain generic functions as for example exec_submit(). The specific function calls must be defined on each system that is added to the grid infrastructure. The application developer must provide the proper program scripts for all the many cases that are important for example in materials science. Due to this technology, W2GRID program scripts are fully portable to any other system, in which the infrastructure can be installed.

A WIEN2k job will first be evaluated by the grid client daemon with respect to its performance parameters (memory and total run-time). Various platforms, on which WIEN2k must be installed and which are available, can be found in the corresponding repository. These platforms are compared in order to make the proper selection and to submit the job to that system, which aims to offer optimal performance concerning the file-transfer, the overall calculation time and the queuing time.

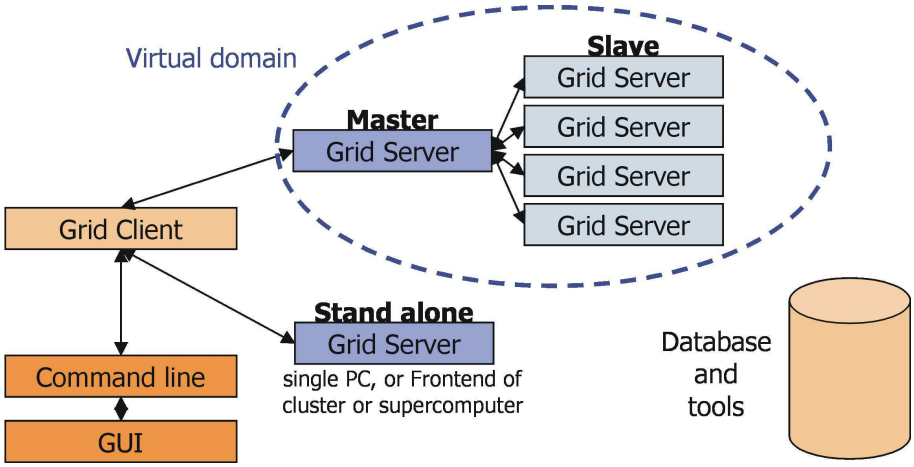

Figure 2. W2GRID, a lightweight middleware: The user communicates – either by using a graphical user interface (GUI) or via command line – with a single grid client daemon that keeps contact with the servers in its registry and distributes the tasks. A grid server daemon is running waiting for remote calls to start or check a computation. Two scenarios are shown, a master with slaves (a system without a queuing systems) or a stand alone system that can be a supercomputer with a queuing system (e.g.PBS). Intermediate data concerning servers, communications, job status are kept in a simple database, which also contains the necessary information to communicate with the underlying platform. In addition some simple tools for administration are available.

3.2. Analytic performance model

Before starting a WIEN2k calculation [3] on the grid one needs to know – as mentioned above – some resource parameters, which can be estimated from the input, e.g.

- $\mathbf{k_{BZ}}$, the number of k-points in the Brillouin zone that are needed for a well converged calculation,
- $\mathbf{n_{at}}$, the number of inequivalent atoms,
- $\mathbf{RK_{max}}$, another important parameter in WIEN2k, which determines the basis set and thus the matrix size M [3],
- **inversion symmetry**, whose existence can be checked by a program for the specified crystal structure.

These parameters can be extracted from the given input files by W2GRID and allow to estimate the required computational resources using a rather simple analytic performance model. For example the memory needed in solving the general eigenvalue problem may vary from 100 MB to several 10 GBs and is determined by M, the size of the matrices. If the system under consideration does have inversion symmetry, the matrices are real, otherwise they are complex and thus need twice as much memory. The following formula gives an estimate of the memory size in bytes

$$\text{size} = M^2 \cdot 8 \cdot c \cdot p, \qquad c = \begin{cases} 1 & \text{real, (inversion)} \\ 2 & \text{complex, (no inversion),} \end{cases} \tag{2}$$

where the factor 8 comes from the double precision word and $p \approx 1.4$ is the approximate 40 per cent overhead for the programs and auxiliary arrays. The minimum memory requirement may already rule out certain machines, on which the present case can not be run efficiently. The remaining subset of machines can be taken into further consideration.

The second crucial performance parameter is the estimated computation time per SCF cycle, $time_{scf}$, which mainly depends on the matrix size and linearly on the number of k-points. It may vary from seconds to several hours of CPU time. The main computer time (for each k-point) is spent in setting up the matrices (scales roughly with M^2 and the number of atoms) and solving the general eigenvalue problem (scales with M^3 in the case of direct diagonalization). This time can roughly be estimated using the formula

$$\text{time}_{SCF} = f \cdot k_{BZ} \cdot (\alpha \cdot n_{at} \cdot M^2 + \beta \cdot M^3) \cdot c, \qquad c = \begin{cases} 1 & \text{real} \\ 4 & \text{complex.} \end{cases} \tag{3}$$

The proportionality factor f combines all platform- and processor related performance numbers in a single value specific for a particular processor. It can be adjusted together with the parameters α and β from a set of test calculations. Depending on inversion symmetry, linear algebra packages for real or complex matrices must be used, which differ in CPU effort by a factor of about 4.

The simple examples that were given already illustrate that each solid state problem for WIEN2k can be analyzed and the corresponding task can be put in the proper category concerning memory requirement, data locality, coarse or fine grain parallelization (using MPI and SCALAPACK) and rough estimates of required CPU time. However, it can not be know in advance, how many SCF cycles it takes to reach convergence but experience shows that often about 20 cycles will be needed. This allows a very crude estimate of required computing costs.

This is the part that presently is already implemented in the beta version of W2GRID. In the future a more sophisticated performance model can include further aspects, like the algorithms (full or iterative diagonalization), spin–polarization or spin–orbit coupling etc. In the latter cases the calculation must be done for both, spin-up and spin-down electrons, doubling the effort.

3.3. Steering

For time-demanding WIEN2k jobs a monitoring of some intermediate results is highly desirable in order to allow steering of the job during runtime. For example, if oscillations (charge flushing) occur, one may want to change an input file (e.g. increase the damping) or properly terminate the running SCF step, so that the calculation can be started again after analyzing the problem and modifying some parameters. Thus it is desirable to interactively influence the flow of a current calculation. Since the corresponding communication is done with simple ASCII files, the steering requires a continuous transfer of the appropriate input- and output-files from the remote host back to the user and to synchronize modified input files with the remote hosts, where the calculation is running. This synchronization is automatically done with W2GRID and thus the user only needs to specify the corresponding file on the user side but not on the actual hosts (see file transfer section).

3.4. Resource management

The whole program-package allows a large variety of workflows that will be invoked. The SCF cycle will be the basic building block of most calculations that might be scheduled to the grid. Depending on the materials science problem, a series of SCF cycles may be needed to solve a problem leading to quite different workflows. Two representative examples are illustrated in Fig. 3.

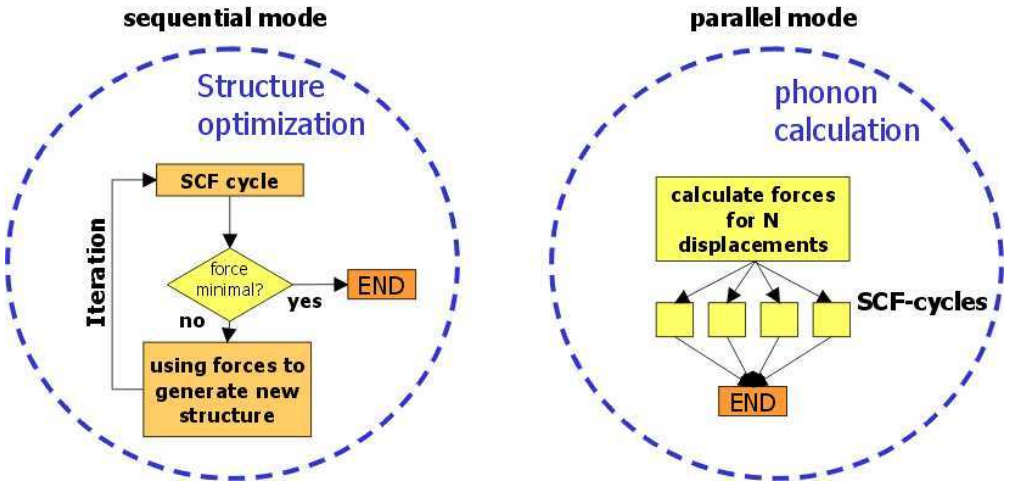

Figure 3. Workflow of two representative examples using SCF cycles as building blocks: (left) sequential mode (e.g. structure optimization), (right) parallel mode (e.g. the calculation of phonons).

i **Optimization of a structure:** For a given crystal structure with a set of atomic positions, first an SCF calculation is done leading to forces acting on the atoms. In order to find the equilibrium geometry, the atoms are moved (within the unit cell) according to the forces acting on them till they reach their equilibrium (i.e., the forces are below a threshold). This task is implicitly sequential, since each SCF cycle must be completed before the atoms can be moved to new positions and the next SCF cycle can be started. Therefore one should select those machines on which an SCF cycle runs the fastest. Such sequential calculations should presently not be run on physically remote machines without a shared file-system.

ii **Phonon calculations** by the "direct method" require a series of (independent) calculations, where in each case one atom is displaced by a small distance in a predetermined direction and the resulting forces acting on all other atoms in the unit cell are computed in an SCF cycle with a tight convergence criterion on the forces. From the results of these independent tasks the dynamical matrix can be constructed, which then allows phonon calculations. This can be described as a parameter case study that can be done on loosely coupled computers. Only at the end of all these runs small output files (with the resulting forces) must be collected to do the final analysis. This would be a very favorable case for grid computing using many available machines, which are appropriate for efficiently running SCF cycles.

The two examples given above illustrate that choosing the appropriate resource to run a given calculation is a non-trivial task. Despite the crucial memory consideration, the job performance depends on

the whole workflow, which can consist of either a single SCF cycle or a number of SCF cycles, which either depend on each other or can run independently in parallel as illustrated above (see Fig. 3). Furthermore scalability is not guaranteed but depends on the workflow or algorithm that needs to be used. The present resource management of W2GRID works on the basis of a static load-limit. Jobs are scheduled to those nodes whose current load is within a given range. In the future it will be important to incorporate dynamical load balancing into this system and to allow migrating jobs to better suited compute nodes.

3.5. File transfer

W2GRID was designed to meet the needs of WIEN2k concerning file transfer and to provide the functionality for specific applications. Some files help the steering of a calculation, some are continuously growing, and others are overwritten and thus each file must have a special attribute, which specifies how it should be handled by a file transfer mechanism. A few examples are given below.

In simple applications the input files are sent at the beginning and the output files after the calculation has finished, but in WIEN2k – in addition to the standard case – some files should be sent in between, for example for monitoring or error detection. In large-scale applications a few files can become rather large (several GB) and thus file transfer can be a bottleneck. A more specific WIEN2k aspect is that certain files are continuously appended and may get quite large, for example the SCF file which keeps the main output results of previous iterations for monitoring the convergence of the SCF cycles.

Another peculiar aspect is the way in which the sometimes large eigenvectors are written, namely as binary file in order to retain machine precision. Since the binary representation varies with hardware, such files can not easily be transferred to another type of machine without overhead. Converting these data to transferable format is feasible but would lead to substantial communication requirements and thus is not implemented at present.

A list of typical file attributes and the corresponding action concerning file transfer is given below. There are four types specifying the time at which the file transfer should occur:

- Input: transfer such a file at the very beginning of a calculation.
- Check: such files will be checked and transferred periodically during a job. They may contain intermediate results from the remote host, which the user may need for steering, or updated local input files, which must be transferred to the remote host.
- Finished: transfer this file when the job has been successfully completed.
- Error: transfer this file provided the job step has terminated with an error.

In addition the following three flags specify the details of the file transfer:

- Update: for such a file a hash key is generated that is compared with the remote key. Provided these two keys do not match this file is synchronized depending on the time stamps of the remote and local version of this file.
- Append: only the most recent data are transferred and appended to the already existing file accessible to the user.

- Sync: This flag will create or remove a file on the remote host if the file is created or removed locally (and the other way around).

These attributes and flags can be combined and allow the developer to define meta data for the files, which may differ in various applications and in their corresponding workflows. The application scientist does not need to get involved in these details. The logic of a file with its flags, options and purposes is independent of the actual mechanism for file transfer, which can employ any method for which a proper plug-in is provided. The type of transfer is specified when a remote host is added to the grid client registry. Third party programs for file transfer can be integrated.

4. Conclusion

The main purpose of this paper is to provide a framework for interdisciplinary research. A mutual understanding of the problems in all disciplines is essential but not easy to achieve. The material scientist, the mathematician, and the computer scientist must all appreciate the detailed experience of the others. No single group alone can solve the problems. The attempt is made to describe the complexity of materials problems, sketch the main mathematical problems (and algorithmic challenges) and indicate the computer science aspects in connection with a grid infrastructure in order to efficiently solve such applications.

There are many scenarios in materials sciences where a grid environment looks promising. The WIEN2k community with about 1000 groups worldwide can act as a forum for testing such an environment and may eventually lead to a corresponding virtual organization (VO) that can carry out new research topics. The latter may require both, heavy computations and sophisticated analysis tools including visualization. In an international co-operation one must rely on standards in grid environment (basics and middleware) from security, authentication, all the way to licenses and accounting. Some aspects such as legal obligations or quality of service (QoS) are not crucial for materials sciences applications, in contrast to medical applications, for which these issues are extremely important.

Nevertheless the present case has illustrated the high complexity of real applications, which can only be solved by joint efforts among chemists, physicists, mathematicians and computer scientists. The goal is to provide an infrastructure that can largely be hidden from the application group, which would like to focus just on the materials sciences problem. An efficient use of all the (hardware and human) resources requires a broad spectrum of details which must be combined in a concerted action, from new application requirements, algorithms for parallel or distributed computing, to efficient libraries (LAPACK or SCALAPACK, BLAS, MPI), and all the way to computer architecture with data locality, file transfer, communication and software components in all layers from the system to the portal.

The developers (like us for WIEN2k) can provide the program specific details that are separated from the grid infrastructure which must be defined in terms of mode of file transfer or type of queuing system. A widespread use can only be achieved by building on international standards wherever possible. With W2GRID we have explored the necessary functionality to make WIEN2k grid-enabled. So far W2GRID is already available as beta version, it can be interfaced to external programs and may interoperate with existing grid middleware. At present the corresponding plugins for PBS, LoadLeveler and the SunGridEngine are already available. Further work will include the development of plugins for other middleware (Condor).

5. Acknowledgments

Part of this work was supported by the Austrian Research Fund (SFB AURORA F1108) and the Austrian Grid Project.

References

[1] KOHN W., Sham L.S., "Self-consistent equations including exchange and correlation effects" in *Phys. Rev. B*, **Vol. 140**, 1133 (1965).

[2] BLAHA P., Schwarz K., Madsen G.K.H., "Electronic structure calculations of solids using the WIEN2k package for materials sciences" in *Comp. Phys. Commun.*, **Vol. 147**, 71 (2002).

[3] BLAHA P., Schwarz K., Madsen G.K.H., Kvasnicka D., Luitz J., "An Augmented Plane Wave Plus Local Orbital Program for Calculating Crystal Properties" in "WIEN2k" *ISBN* , **3-9501031-1-2** (2001).

[4] SCHWARZ K., "DFT calculations of solids with LAPW and WIEN2k" in *Solid State Commun.*, **Vol. 176**, 319 (2003).

[5] BRUIN R.P., Dove M.T., Calleja M., Tucker M.G., "Building and managing the eMinerals cluster" in *Computing in Science and Engineering*, **Vol. 7, Number 6**, 30 (2005).

[6] HAYES M., Morris L., Crouchley R., Grose D., van Ark T., Allan R., Kewley J., "GROWL: A Lightweight Grid Services Toolkit and Applications", www.allhands.org.uk/2005/proceedings/papers/460.pdf (2005)

[7] CHIN J., Coveney P.V., "Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware", www.nesc.ac.uk/technical_papers/UKeS-2004-01.pdf (2004)

[8] AGRAWAL S., Dongarra J., Seymour K., Vadhiyar S. "NetSolve: past, present and future – a look at a Grid enabled server" in F. Bernan, G.C. Fox, A.J.G. Hey (Ed.), "Grid Computing", **John Wiley & Sons Inc.**, West-Sussex (2003).

[9] THAIN D., Livny M., "Building Reliable Clients and Servers" in I. Foster, C. Kesselman (Eds.), *The Grid: Blueprint for a New Computing Infrastructure*, **Morgan Kaufmann**, (2003).

[10] THAIN D., Tannenbaum T., Livny M., "Condor and the Grid" in F. Bernan, G.C. Fox, A.J.G. Hey (Eds.), "Grid Computing", **John Wiley & Sons Inc.**, West-Sussex (2003).

[11] SNELLING D., "Unicore and the Open Grid Services Architecture" in F. Bernan, G.C. Fox, A.J.G. Hey (Eds.), "Grid Computing", **John Wiley & Sons Inc.**, West-Sussex (2003).

[12] WELCH V., Siebenlist F., Foster I., Bresnahan J., Czajkowski K., Gawor J., Kesselman C., Meder S., Pearlman L. and Tuecke S., "Security for grid services","Proceedings of the Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)",IEEE Press (2003).