List of proposal for the Master of HPC

Proposal 1: Improving strong scaling (i.e. at fixed problem size) for Quantum ESPRESSO

Proposed by: Paolo Giannozzi

Description of the problem

In spite of the many available parallelization levels, PWscf may not scale well for many systems of interest beyond a relatively small number of processors, especially for USPP and PAW where the lower cutoff makes the H\psi term less dominant (or even not dominant at all).

The main bottlenecks limiting strong scaling are, in decreasing order of importance:

- subspace (conventional) diagonalization in Davidson. ScaLAPACK and ELPA do not scale beyond N^2 processors when the number or rows or columns divided by N is smaller than a few hundreds (and they never scale well). In CP this is also true but CP makes a smaller usage of conventional diagonalization;
- Calculation of charge density and potentials. These calculations (which are replicated on different pools in case of k-point parallelization) may take a sizable amount of time, even after the recent speedup of the USPP and PAW parts. The *local-TF* mixing mode is also expensive;
- FFT's, not scaling on more than N processors with N in the order of the third FFT dimension. Task groups help for FFT's in H\psi but not for FFT's on charge density and potentials.

It is time to drop the paradigm from the 80's: "*H\psi and iterative diagonalization is where most of the time is spent, all the rest is irrelevant*". The rest is more and more relevant nowadays.

Proposed Work

- Study new diagonalization algorithms with reduced subspace diagonalization. Among the possible candidates:
  - FEAST (http://www.ecs.umass.edu/~polizzi/feast/)
  - LOPBCG (A. Knyazev, SIAM J. Sci.Comput. 23 (2) (2001) 517–541; https://en.wikipedia.org/wiki/LOBPCG)
  - PPCG (http://arxiv.org/pdf/1407.7506.pdf: a version exists for diagonalization without overlap matrix)
  - ParO: arxiv.org/pdf/1405.0260.pdf, http://arxiv.org/pdf/1510.07230.pdf
  - Tweaking Davidson (i.e. simple tweaking: set "diago_david_ndim=2")

The reduced subspace diagonalization is ideally absent but in practice must be done for numerical stability, but even reducing subspace diagonalization to one every 4-5 H\psi, on matrices Nb x Nb, Nb=number of states, would be a big improvement Even if none of these proves able to replace Davidson, a replacement of poorly-scaling but memory-sparing CG diagonalization would already be useful

- Parallelization over bands (preferably, distributed bands), in addition to (or in replacement of) task groups. This may also allow to remove the current (slow) mechanism that computes distributed <\beta_i|\psi_j>;
- Rethink k-point parallelization ("pools"): it should split the communicator for operations on rho and V. Currently all those operations are replicated on each pool;
- Extend OpenMP parallelization, still not present in many parts of the code;
- Add "spot" parallelization levels for parts of the code not currently parallelized, even if they take little time: everything becomes eventually a bottleneck.

Skills required

- Knowledge of Fortran 90, or willingness to learn it;
- Knowledge of MPI and OpenMP;
- Some knowledge of what PWscf does and which algorithms it uses, or willingness to learn them, is a plus.

Proposal 2: Automatic estimate of parallelization parameters

Proposed by: Paolo Giannozzi

Description of the problem

Most inexperienced users use PWscf (and likely also CP) with grossly inadequate parallelization parameters, by setting a large number of processors and expecting the code to cope with it. Even experienced users would appreciate a quick way to estimate the best parallelization options, or an automatic even if sometimes suboptimal estimate.

Proposed Work

- Decide and implement heuristic criteria. For instance, given a total number N of processes:
  - always use k-point parallelization if possible (Nk pools)
  - if Nfft=N/Nk > Nr3 use Nt task groups so that Nfft=N/Nk/Nt > Nr3/2
  - if still too many processes, use 4-8 OpenMP tasks if available
  - use Nd*Nd < Nfft processors for linear algebra so that Nbands/Nd is in the order of some reasonable number (250? 500?) Test the criteria in some cases, improve if needed
- Decide and implement criteria for analyzing the results, such as:

- too large Wall-CPU time difference (not explained by OpenMP) => something is not right
- too much time spent in FFT scatter => reduce number of processors for FFTs
- too much time spent in subspace diagonalization => change number of processors for linear algebra, use k-point parallelization
- Consider probing optimal parallelization, by performing selected tests of needed operations for different number of processors

Skills required

- Knowledge of Fortran 90, or willingness to learn it;
- Knowledge of MPI and OpenMP;
- Some knowledge of what PWscf does and which algorithms it uses, or willingness to learn them, is a plus.

Proposal 3: Analysis and interpolation of the ground state energy of solids as a function of the crystal parameters

Proposed by: Andrea Dal Corso

Description of the problem

The size and shape of the unit cell of a crystalline solid is given by the crystal parameters, the lengths of the principal vectors and their angles. Depending on the crystal, one parameter might suffice to describe the cell as in cubic crystals while six parameters are necessary in triclinic crystals. Codes such as Quantum ESPRESSO can calculate the energy of the solid as a function of the crystal parameters and also minimize it with automatic tools. The actual crystal parameters at zero temperature are those that minimize the energy.

The purpose of this project is to analyze the energy function about the minimum for systems that require three or more parameters. We plan to interpolate the energy by a second or fourth order polynomial and to determine the main properties of the polynomial interpolation. Moreover we are interested in finding optimal sets of points in parameter space that minimize the number of energy evaluations needed to find the polynomial coefficients. Finally another goal of the project is to write useful software to visualize the energy surface or its polynomial interpolation.

Proposed Work

- Choose an orthorombic solid and compute the total energy of the system for a uniform mesh of points in the three dimensional space of the crystal parameters remaining close to the experimental values of these parameters;
- Fit the total energy function with a second or fourth order polynomial and find the minimum of these polynomials;

- Discuss the accuracy of the determination of the minimum as a function of the number of points of the grid and of the ranges chosen for the parameters;
- Find and discuss methods to reduce as much as possible the size of the three dimensional grid needed to fit the polynomials;
- Plot the total energy about the minimum finding ways to visualize a function depending on three parameters;
- Choose a monoclinic solid whose crystal structure depends on four parameters and discuss the same questions for its energy surface;
- Finally (if possible) choose a triclinic solid whose structure depends on six parameters and discuss the same questions for its energy surface.

Skills required

- Knowledge of Fortran 90, or willingness to learn it;
- Ability to use the Quantum ESPRESSO package in an HPC environment, or interest to learn its use;
- Interpolation of multidimensional functions;
- Interest in computer graphics and data visualization.

Proposal 4: Pools optimization of Quantum Espresso and thermo_pw

Proposed by: Andrea Dal Corso

Description of the problem

Thermo_pw is a set of Fortran drivers for the parallel and/or automatic computation of material properties using Quantum ESPRESSO (QE) routines as the underlying engine. See: http://www.qe-forge.org/gf/project/thermo_pw/

The QE codes have several levels of parallelization: G-vector, k-points, etc. The parallelization scheme is decided by the user at the beginning of the run by using input flags given as arguments to the code. According to this scheme the processors are divided in different pools whose size is fixed.

There are several cases in which this fixed division of processors might limit the scalability of the codes. For instance, thermo_pw might call several instances of the QE routines and different routines might require very different types of parallelizations. A typical example is the calculation of the density of electronic states where a self-consistent run is followed by a band structure calculation. The number of k points of the self-consistent run is much smaller than the number of k points of the band structure calculation so the former limits the maximum number of pools that can be used. It would be useful to be able to run the two calculations within the same code but

with different numbers of pools. Similarly for different q-vectors the phonon code might require very different numbers of k points and therefore different numbers of pools to optimize the run.

The purpose of this project is to remove the present limitations writing the support software that, calling MPI routines, allows to change the division into pools of the available processors during the same run. Moreover, the usefulness of the approach has to be demonstrated by running several benchmark examples.

Proposed Work

- Learn how the Quantum ESPRESSO code splits the processors into different pools and generate communicators for these pools using MPI calls;
- Using this knowledge write a routine able to reinitialize these communicators with a different division of processors among pools without interrupting the run;
- Write a FORTRAN demonstrative code able to initialize the Quantum ESPRESSO parallel environment using routines provided by the package and make a single self consistent run of the code pw.x;
- Using the new routine reinitialize the communicators and using a different set of pools in the same run make a non self-consistent run of pw.x reading the data produced by the first run as an input;
- Insert the new routines in the thermo_pw code and discuss the advantages or disadvantages of using the dynamical division of the processors.

Skills required

- Knowledge of Fortran 90 and MPI, or willingness to learn it;
- Some knowledge of the parallelization schemes of QE or interest to learn them.

Proposal 5: Improving real-space implementation of non-local pseudo-potentials in Quantum ESPRESSO

Proposed by: Stefano de Gironcoli

Description of the problem

In modern electronic structure codes based on a plane-wave wave-function representation, like PWscf and CP in Quantum ESPRESSO, the interaction between valence electrons and ions is represented by non-local pseudo-potentials of Norm-conserving (NC), Ultra Soft (US) or Projector-Augmented Wave (PAW) type.

The action of these pseudo-potentials on trial wave-functions can be represented by a series of projections on localized orbitals that can be computed either in real or reciprocal space. The

wave-function contribution to the system charge density involves, for US and PAW pseudo-potentials, the incorporation of localized augmentation charges that again can be added in real or reciprocal space.

The real space implementation is in principles preferable because the spacial localization of the pseudo-potential projectors and augmentation charges can be exploited for better performance however its current realization in PWscf suffers from severe aliasing errors whenever the Fourier expansion of the needed ingredients is not complete.

Due to this problem the reciprocal space implementation is most often used in practical calculations with significant efficiency loss.

Proposed Work

- Analysis of the current implementation in real and reciprocal space for the augmentation charge calculation and pseudo-potential projectors;
- Review of methods proposed in the literature to cope with the aliasing problem;
- Implementation of one or more alternatives;
- Optimization of the code and integration with different parallelization strategies in the code.

Skills required

- Knowledge of Fortran 90, or willingness to learn it;
- Knowledge of MPI and OpenMP;
- Some knowledge of what PWscf does and which algorithms it uses, or willingness to learn them, is a plus.

Proposal 6: Implementation of a general toolkit for wave-function operations

Proposed by: Pietro Delugas

Description of the problem

State vectors in plane wave codes like Quantum Espresso are represented by arrays of complex numbers. The effective meaning of these arrays may anyway change, depending on the definition of the plane-wave basis set and the type of pseudo-potentials which are used.

For this reason basic operations on wave-functions such as fourier transforms, scalar products and the application of local and non-local operators are specific for each representation. This introduces additional complexity in reading, maintaining and reusing the code. For this reason we need to develop domain-specific libraries which implement wave function operations in the most

efficient and transparent way.

Moreover, when wave-functions are saved for further processing, one has to bring along a significant amount of additional information and the development of post-processing software is often burdened by the necessity of collecting such information from different files and writing initialization routines specific for each representation; one has also to cope with the non-uniformity in the format with which state vectors are saved.

It would be important to uniform the format and the modality for saving the state vectors and related information and provide the community of a standard set of subroutines allowing the reading and manipulation of the state-vectors transparently from the representation.

Proposed Work

- Implementation of general wrappers and specific routines for wave-function scalar products and distributed matrix-matrix operations for the general and the gamma-only basis set representation;
- Implementation of general wrappers and specific routines for backward Fourier transforms of wave-functions. Possibly with case specific parallelism;
- Devise data structures which contain all the information needed for initializing operators on state vectors and organise them as HDF5 groups and data-sets;
- Write Fortran and/or Python reusable software for writing and reading the state vector information.

Skills required

- Knowledge of Fortran 90 or willingness to learn it;
- Willingness to learn what PWscf does and which algorithms it uses;
- Knowledge of HDF5 Fortran interface or willingness to learn it;
- Knowledge of python HDF5 interface or willingness to learn it will be a plus.