Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelorthesis

Optimization of the Quantum Espresso Density Functional Theory Code for parallel execution on the PHYSnet-Cluster

vorgelegt von

TJARK SIEVERS

Fakultät: Mathematik, Informatik und Naturwissenschaften
Fachbereich: Physik
Studiengang: Physik
Matrikelnummer: 7147558
Erstgutachter: Prof. Dr. Tim Wehling
Zweitgutachterin: Prof. Dr. Daria Gorelova

Todo list

density as calculated with KS states . . . 3
expansion of wave function in basis functions, also periodicity -> integrals over BZ . . . 4
not finished: kinetic energy -> fourier transform . . . 4
whole section . . . 4
functional/data parallelism? . . . 5
more details strong/weak scaling? . . . 6
what to look out for in configure output: when are the internal copies of blas and lapack used? . . . 7
finish . . . 7
is this good here? . . . 8
explanation of wait time, also this all would probably fit better in the computation chapter? so there is a central reference? . . . 10
analysis with right plots for ompi, ompi+openblas, maybe kick out scalapack? it should be the same . . . 12
absolute runtimes! so comparison is possible! (singlecore runtimes could also be differen!) 12
more analysis: difference between poolsizes . . . 13
more analysis: difference between poolsizes . . . 14
Linear Algebra . . . 15

Kurzzusammenfassung

Abstract

Contents

Motivation **vii**

I Many-body physics **1**
- I.1 The electronic structure problem . . . 1
- I.2 Density Functional Theory . . . 1
  - I.2.1 Hohenberg-Kohn theorems . . . 2
  - I.2.2 Kohn-Sham equations . . . 3
  - I.2.3 Pseudopotentials and basis set . . . 4
- I.3 Density Functional Perturbation Theory . . . 4

II Computational Details **5**
- II.1 Parallel computing . . . 5
  - II.1.1 On scalabilty . . . 5
- II.2 QUANTUM ESPRESSO . . . 6
  - II.2.1 Compilation of QUANTUM ESPRESSO . . . 6
  - II.2.2 Parallelization capabilities offered by QUANTUM ESPRESSO . . . 7
- II.3 Hardware configuration of the PHYSnet cluster . . . 8

III Parallelisation of electronic-structure calculations **9**
- III.1 First scaling tests . . . 9
- III.2 Testing different compilers and mathematical libraries . . . 12
- III.3 Using the parallelization parameters of QUANTUM ESPRESSO . . . 12
  - III.3.1 k point parallelization . . . 12
  - III.3.2 Linear algebra parallelization . . . 15
- III.4 Comparison with calculations on the HLRN cluster . . . 15
- III.5 Conclusion: Parameters for optimal scaling . . . 15

IV Parallelization of DFPT calculations **17**
- IV.1 Optimal parallelization parameters for DFPT calculations . . . 17
  - IV.1.1 k point parallelization . . . 17
  - IV.1.2 Linear algebra parallelization . . . 17
- IV.2 Image parallelization . . . 17
- IV.3 Conclusion: Parameters for optimal scaling . . . 17

Bibliography **19**

Listings **21**
- List of Figures . . . 21
- List of Tables . . . 21

Acknowledgement **23**

Motivation

I Many-body physics

I.1 The electronic structure problem

In solid state physics, the Hamiltonian describing the interacting nuclei and electrons in the solid is well known, as all interactions except Coulomb interaction can safely be ignored at the mass and energy scales at which the electrons and nuclei reside. This very general problem consisting of both the electronic and nuclei degrees of freedom can be simplified in a first step by employing the Born-Oppenheimer approximation. The approximation assumes the nuclei to be fixed point charges which create a potential for the N interacting electrons, so that the electronic part can be solved independently using the nuclei positions $\mathbf{R}_\alpha$ as a parameter.

This problem is described by the time-independent Schrödinger equation

$$\hat{H}\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N) = E\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N) \tag{I.1}$$

with the Hamiltonian in first quantization (i running over the electrons, α, β over the nuclei)

$$\begin{aligned} \hat{H} &= \hat{T}_e + \hat{U}_{e-e} + \hat{V}_{n-e} + \hat{W}_{n-n} & \text{(I.2)} \\ &= -\sum_i \frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i\neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_i \sum_\alpha \frac{Z_\alpha}{|\mathbf{r}_i - \mathbf{R}_\alpha|} + \frac{1}{2}\sum_{\alpha\beta} \frac{Z_\alpha Z_\beta}{R_{\alpha\beta}} & \text{(I.3)} \end{aligned}$$

where:

- $\hat{T}_e$ is the kinetic energy of the electrons
- $\hat{U}_{e-e}$ is the Coulomb interaction between the electrons and
- $\hat{V}_{n-e}$ is the potential energy of the electrons in the field of the nuclei
- $\hat{W}_{n-n}$ is the Coulomb interaction between the nuclei

The terms $\hat{V}_{n-e}$ and $\hat{W}_{n-n}$ can then be combined into an external potential V for the interacting electrons, so that the Hamiltonian reads

$$\hat{H} = \hat{T} + \hat{U} + \hat{V} \tag{I.4}$$

This Hamiltonian will be used in the further development of the underlying theory for this thesis.

I.2 Density Functional Theory

A direct solution to the electronic structure problem, this meaning obtaining the ground-state many-body wave function $\Psi(\mathbf{r_1}, \ldots, \mathbf{r}_N)$ for a given potential is analytically impossible even for a small number of electrons compared to the number of electrons in a macroscopic crystal. As such, the need for good approximations to obtain results for real world systems is high. One particularly successful approach is Density Functional Theory (DFT). In the following section, the theoretical framework of DFT will be developed, the outline of which can be found in any literature on solid state physics [1].

I.2.1 Hohenberg-Kohn theorems

The starting for DFT is the exact reformulation of the outlined electronic structure problem by Hohenberg and Kohn [2]. This reformulation uses the ground state density of the electronic system $n_o(r)$ as the basic variable. To achieve this, Hohenberg and Kohn [2] formulated two theorems, which demonstrate that the ground state properties of an electronic system can be described using the ground state density (the proof of those theorems is omitted here, but can be found in the original paper [2] or any publication on DFT [1]):

I The external potential (and via the Schrödinger equation also the ground state wave function and the ground state energy) is a unique functional of the ground state density (except for an additive constant).

II The ground state energy minimizes the energy functional,

$$E[n(\mathbf{r})] > E_0 \; \forall \; n(\mathbf{r}) \neq n_0(\mathbf{r})$$

The proof of those theorems show the existence of the energy functional $E[n(\mathbf{r})]$, but a concrete expression for it cannot be given. As the ground state wave function is a functional of the ground state density, a formal definition of the energy functional can be written as

$$\begin{aligned} E[n(\mathbf{r})] &= \langle \Psi | \hat{H} | \Psi \rangle \\ &= \langle \Psi | \hat{T} + \hat{U} + \hat{V} | \Psi \rangle \\ &= \langle \Psi | \hat{T} + \hat{U} | \Psi \rangle + \int \mathrm{d}\mathbf{r}' \, \Psi^*(\mathbf{r}') V(\mathbf{r}') \Psi(\mathbf{r}') \end{aligned}$$

Defining the universal functional $F[n(\mathbf{r})] = \langle \Psi | T + U | \Psi \rangle$, which is material independent and writing $n(\mathbf{r}') = \Psi^*(\mathbf{r}')\Psi(\mathbf{r}')$, the energy functional becomes

$$E[n(\mathbf{r})] = F[n(\mathbf{r})] + \int \mathrm{d}\mathbf{r}' \, V(\mathbf{r}') n(\mathbf{r}') \tag{I.5}$$

This is just a formal definition, as all the formerly mentioned complication of the Hamiltonian I.4 now lie in the functional $F[n(\mathbf{r})]$. With a known or well approximated universal functional $F[n(\mathbf{r})]$, the Hohenberg-Kohn theorems provide a great simplification for finding the ground state properties of a solid state system, as the problem is now only a variational problem with 3 spatial coordinates instead of $3N$ coordinates when trying to solve the full Hamiltonian.

I.2.2 Kohn-Sham equations

One way of approximating the functional $F[n]$ was given by Kohn and Sham [3]. The idea is to use a non-interacting auxiliary system of electrons

$$H_0 = \sum_{i}^{N_e} \frac{p_i^2}{2m} + v_{KS}(\mathbf{r}_i) \tag{I.6}$$

With a correction potential v_{KS} such that the ground state charge density for the auxiliary and the interacting system are the same. This introduces a new set of orthonormal wave functions, the solutions to the non-interacting problem $\mathit{\Psi}_i$. The kinetic energy of such a non-interacting problem can be easily calculated as sum over all electrons:

density as calculated with KS states

$$T_S[n(\mathbf{r})] = -\frac{1}{2} \sum_{i=1}^{N_e} \int \mathrm{d}^3r \, \mathit{\Psi}_i^*(\mathbf{r}) \mathit{\Delta} \mathit{\Psi}_i(\mathbf{r}) \tag{I.7}$$

With the help of this system and the classical electrostatic energy

$$E_H[n(\mathbf{r})] = \frac{1}{2} \int \int \mathrm{d}^3r_1 \, \mathrm{d}^3r_2 \, \frac{n(\mathbf{r}_1)n(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|} \tag{I.8}$$

an ansatz for the universal functional can be written as

$$F[n(\mathbf{r})] = T_S[n(\mathbf{r})] + E_H[n(\mathbf{r})] + E_{XC}[n(\mathbf{r})] \tag{I.9}$$

where now $E_{XC}[n(\mathbf{r})]$ is a functional of the density accounting for all exchange and correlation effects not present in the non-interacting electron system. The success of DFT lies in the fact that E_{XC} only contributes only a small part of the total energy and can be surprisingly well approximated.

Using that form of $F[n(\mathbf{r})]$, from the variational problem (with a Lagrange parameter introduced to ensure the orthonormality of the states $\mathit{\Psi}_i$)

$$\delta \left(E[n(\mathbf{r})] - \sum_j \lambda_j \left[\int \mathrm{d}^3r \, |\mathit{\Psi}_j|^2 - 1 \right] \right) = 0 \tag{I.10}$$

single particle, Schrödinger-like equations can be derived:

$$\left(-\frac{1}{2} \mathit{\Delta} + \frac{1}{2} \int \mathrm{d}^3r' \, \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + \frac{\delta E_{XC}}{\delta n(\mathbf{r})} + V \right) \mathit{\Psi}_i(\mathbf{r}) = \lambda_i \mathit{\Psi}_i(\mathbf{r}) \tag{I.11}$$

Schrödinger-like in this context means, that with the identification

$$v_{KS} = V_H + V_{XC} + V = \frac{1}{2} \int \mathrm{d}^3r' \, \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + \frac{\delta E_{XC}}{\delta n(\mathbf{r})} + V \tag{I.12}$$

eq. I.11 become the KS equations:

$$\left(-\frac{1}{2} \mathit{\Delta} + v_{KS} \right) \mathit{\Psi}_i(\mathbf{r}) = \epsilon_i \mathit{\Psi}_i(\mathbf{r}) \tag{I.13}$$

Importantly, the potential v_{KS} depends on the solutions $\mathit{\Psi}_i(\mathbf{r})$, as the Hartree potential V_H and the XC potential V_{XC} include the density $n(\mathbf{r})$, so the problem becomes a self-consistency problem.

I.2.3 Pseudopotentials and basis set

In order to represent the states and operators in eq. I.11, a basis set has to be chosen. A natural choice, at least for delocalized (higher energy) orbitals is the discrete set of plane waves $|\mathbf{G}\rangle$

$$\langle \mathbf{r} | \mathbf{k} + \mathbf{G} \rangle = \frac{1}{\sqrt{V}} e^{i(\mathbf{k}+\mathbf{G})\mathbf{r}} \tag{I.14}$$

expansion of wave function in basis functions, also periodicity -> integrals over BZ

where $\mathbf{G}$ is a reciprocal lattice vector. One problem of this approach lies in the fact that the lower energy core electrons are in contrast very localized around the cores and as such need a lot of basis functions. Because the higher orbitals have to be orthogonal to the lower orbitals, they must also have features on the length scale of the core electrons, which means more basis functions are needed to describe them accurately.

not finished: kinetic energy -> fourier transform

Importantly, the core electrons don't contribute much to the electronic structure problem, so an approach to make the calculations more economically is to introduce an effective screened potential which describes the nuclear potential as well as the states close to the nucleus. These potentials are called Pseudopotentials (PP) and the orbitals constructed with them are smooth functions and can be accurately represented with a small number of basis functions. This fact can be used in computation by introducing a *cutoff energy* E_{cutoff} which limits the number of basis functions by using only expansion coefficents with $|\mathbf{k} + \mathbf{G}| \leq E_{cutoff}$.

whole section

I.3 Density Functional Perturbation Theory

II Computational Details

II.1 Parallel computing

The following section will give an overview of the technical aspects of running computer code (such as QUANTUM ESPRESSO) on massively parallel computing environments (such as the PHYSnet compute cluster). The information presented can be found in any textbook on parallel or high-perfomance computing [4].

functional/data parallelism?

II.1.1 On scalabilty

In scientific computing, one can identify two distinct reasons to distribute workloads to multiple processors:

- The execution time on a single core is not sufficient. The definition of sufficient is dependent on the specific task and can range from “over lunch” to “multiple weeks”
- The memory requirements grow outside the capabilities of a single core

In order to judge how well a task can parallelized, usually some sort of scalabilty metric is employed, for example:

- How fast can a problem be solved with N processors instead of one?
- What kind of bigger problem (finer resolution, more particles, etc.) can be solved with N processors?
- How much of the resources is used for solving the problem?

The speedup by using N workers to solve a problem instead of one is defined as $S = \frac{T_1}{T_N}$, where T_1 is the execution time on a single processor and T_N is the execution time on N processors. In the ideal case, where all the work can be perfectly distributed among the processors, all processors need the same time for their respective workloads and don’t have to wait for others processors to finish their workload to continue, the execution time on N processors would be $\frac{T_1}{N}$, so the speedup would be $S = \frac{T_1}{\frac{T_1}{N}} = N$.

In reality, there are many factors either limiting or in some cases supporting parallel code scalability. Limiting factors include:

- *Algorithmic limitations*: when parts of a calculation are mutually dependent on each other, the calculation cannot be fully parallelized
- *Bottlenecks*: in any computer system exist resources which are shared between processor cores with limitations on parallel access. This serializes the execution by requiring cores to wait for others to complete the task which uses the shared resources in question

- *Startup Overhead*: introducing parallelization into a programm necessarily introduces an overhead, e.g. for distributing data across all the processors
- *Communication*: often solving a problem requires communication between different cores (e.g. exchange of interim results after a step of the calculation). Communication can be implemented very effectively, but can still introduce a big prize in computation time

On the other hand, faster parallel code execution can come from:

- *Better caching*: when the data the program is working with is distributed among processors (assuming constant problem size), it may enable the data to be stored in faster cache memory. Modern computers typically have three layers of cache memory, with level 1 cache being the smallest and fastest and level 3 being the largest and slowest, so smaller data chunks per processor can lead to the data not being stored in main memory, but completely in cache or in a faster cache level

more details strong/weak scaling?

II.2 Quantum ESPRESSO

Quantum ESPRESSO (opEn-Source Package for Research in Electronic Structure, Simulation, and Optimization) [5, 6] is a collection of packages implementing (among others) the techniques described in sec. I.2 and **??** to calculate electronic structure properties as well as phonon frequencies and eigenvectors.

II.2.1 Compilation of Quantum ESPRESSO

As the core of this thesis is an in depth examination of the Quantum ESPRESSO software and ways its performance can be optimized, a discussion of the way is is compiled is needed. The information in this section is taken from the Quantum ESPRESSO 7.0 user guide [7].

The Quantum ESPRESSO distribution is packaged with everything needed for simple, non-parallel execution, the only additional software needed are a minimal Unix environment (a shell like `bash` or `sh` as well as the utilities `make` `awk` and `sed`) and a Fortran compiler compliant with the F2008 standard. For parallel execution, also MPI libraries and an MPI aware compiler need to be provided.

Quantum ESPRESSO needs three external mathematical libraries, BLAS and LAPACK for linear algebra as well as an implementation of FFT for fourier transforms. In order to make the installation as easy as possible, Quantum ESPRESSO comes with a publicly available reference implementation of the BLAS routines, the publicly available LAPACK package and an older version of FFTW (Fastest Fourier Transform in the West, an open source implementation of FFT). Even though these libraries are already optimized in terms of the algorithms implemented, usage of libraries implementing the same routines which can use more specific CPU optimizations significantly improves performance (e.g. libraries provided by Intel can use CPU optimizations not present on AMD processors).

On the PHYSnet cluster, a variety of software packages are available as modules. The benchmark in this thesis were made using the following modules:

- `openmpi/4.1.1.gcc10.2-infiniband`: OpenMPI 4.1.0 (implies usage of Quantum ESPRESSO provided BLAS/LAPACK)

- `openmpi/4.1.1.gcc10.2-infiniband openblas/0.3.20`: OpenMPI 4.1.0 and OpenBLAS 0.3.20
- `scalapack/2.2.0`: OpenMPI 4.1.0, OpenBLAS 0.3.20 and ScaLAPACK 2.2.0
- `intel/oneAPI-2021.4`: Intel oneAPI 2021.4

QUANTUM ESPRESSO offers an configuration script to automatically find all required libraries. As the default options of the `configure` script work well in the use case of this thesis, all compilations were made using the minimal commands

```
module load <module names>
./configure --with-scalapack=no|yes|intel
```

with the scalapack options yes (when using `scalapack/2.2.0`), intel (when using `intel/oneAPI-2021.4`) and no otherwise.

what to look out for in configure output: when are the internal copies of blas and lapack used?

II.2.2 Parallelization capabilities offered by Quantum ESPRESSO

Fig. ??

finish

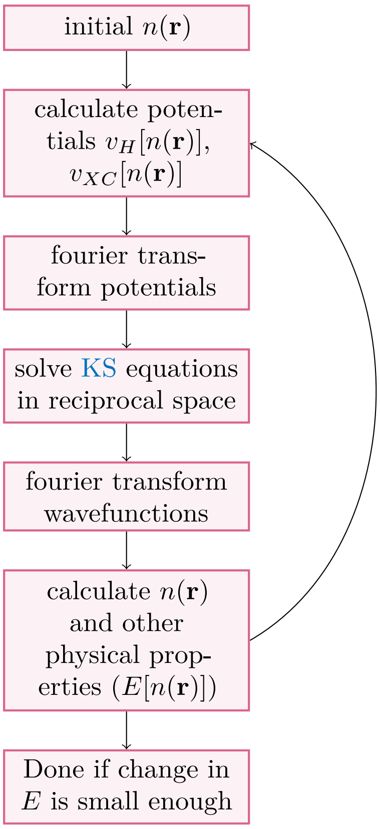

initial $n(\mathbf{r})$
calculate potentials $v_H[n(\mathbf{r})]$, $v_{XC}[n(\mathbf{r})]$
fourier transform potentials
solve KS equations in reciprocal space
fourier transform wavefunctions
calculate $n(\mathbf{r})$ and other physical properties ($E[n(\mathbf{r})]$)
Done if change in E is small enough

Figure II.1: *Flowchart of the algorithm to solve the KS equations*

II.3 Hardware configuration of the PHYSnet cluster

is this good here?

III Parallelisation of electronic-structure calculations

The `PWscf` (Plane-Wave Self-Consistend Field) package is one of the core modules of QUANTUM ESPRESSO , as many other modules need the variables ground state density or total energy as input. This chapter deals with examining the best ways to run `PWscf` calculations in the `scf` mode.

III.1 First scaling tests

The first step in analysing the scaling of the `PWscf` is to perform a baseline scaling test without any optimisations appplied. In Fig. III.1 to III.4 two scaling tests on the earlier mentioned benchmarking systems Si and TaS2 are pictured. The tests are run using QUANTUM ESPRESSO 7.0, compiled using the Fortran and C compilers in OpenMPI 4.1.0, without any of the compilation or runtime optimisation parameters mentioned in section II.2 used.

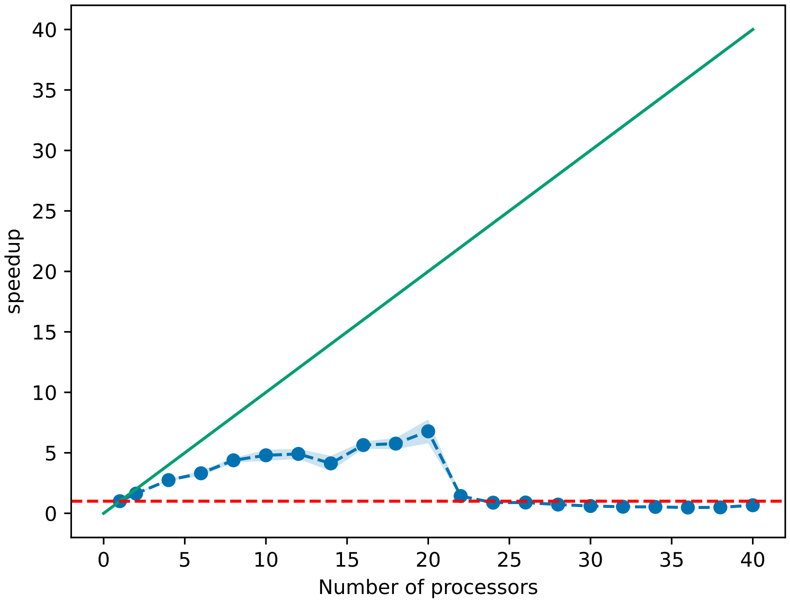


Figure III.1: *Baseline scaling test on the Si benchmarking system* QUANTUM ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`

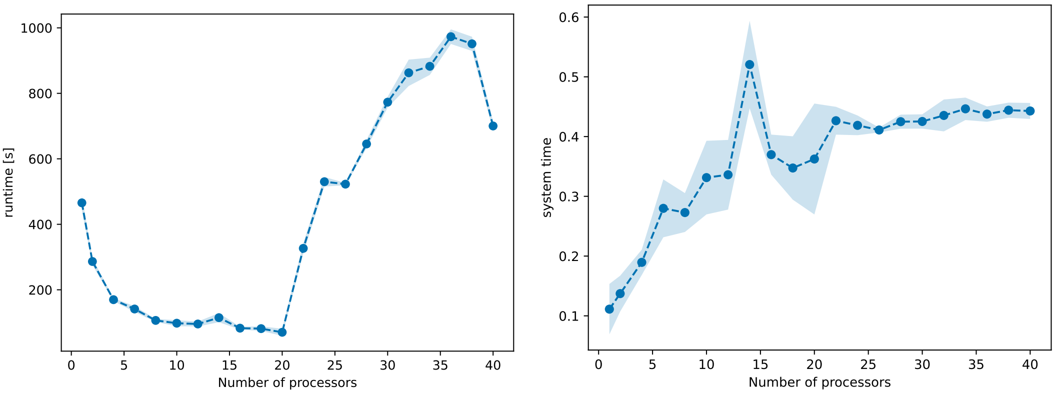


Figure III.2: *Baseline scaling test on the Si benchmarking system* QUANTUM ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`

Three different metrics of scalability are pictured in III.1 and III.2.

- runtime: absolute runtime (walltime) of the compute job
- speedup: runtime divided by runtime of the job on a single core
- system time: percentage of wall time used by system tasks, e.g. writing to disk, etc. (calculated as (walltime - cputime) / walltime)

For further analysis mainly speedup will be used as a metric of scalability, because it lends itself to easy interpretation: optimal scalability is achieved when the speedup scales linearly with the number of processors (with a slope of one), as discussed in ch. II.1. This on the one hand necessarily implies good parallelization and a lower runtime for more processors used, but the other two parameters should also always be considered.

As an example, for a problem with a single core runtime of 600 s, a speedup of 100 would mean a runtime of 6 s, whereas a speedup of 200 would mean a runtime of 3 s. Even with optimal scaling, the 100 processors needed for the speedup of 200 could be considered wasted for just 3 s of saved time. On the other hand, for a problem with a single core runtime of 2400 h, the difference between a speedup of 100 (runtime 24 h) and 200 (runtime 12 h) is the difference between needing to wait a whole day for the calculation and being able to let the job run overnight and continue working in the morning, so using 100 more processors for that can be considered a good use of resources.

explanation of wait time, also this all would probably fit better in the computation chapter? so there is a central reference?

On a single node, both the Si and TaS2 calculations show good, but not perfect scaling behavior: the speedup does approximately scale linearly with the number of processors, but the slope is closer to $\frac{1}{2}$, than 1. Even though the scaling behavior is not perfect, there is just a small, almost constant amount of runtime used by system calls, this speaks for good parallelization. As discussed in sec. II.1, startup time is an unavoidable part of every parallel program, so a constant amount of time used not for calculations is expected.

When using more than one node, not only does the scaling get worse, the execution needs longer than on a single core for the Si system (resulting in a speedup smaller than 1), with a marginally better, but still worse performance than on a single node for the TaS2 system. The

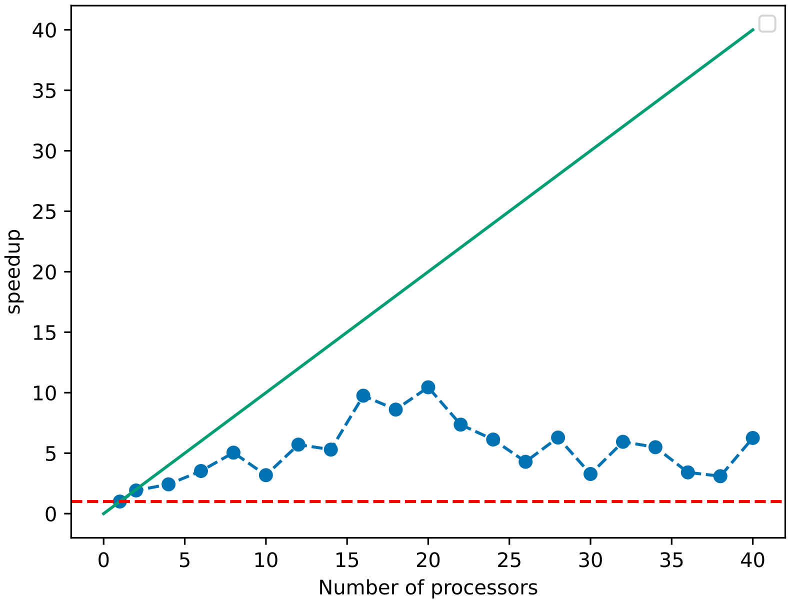


Figure III.3: *Baseline scaling test on the TaS2 benchmarking system* QUANTUM ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`

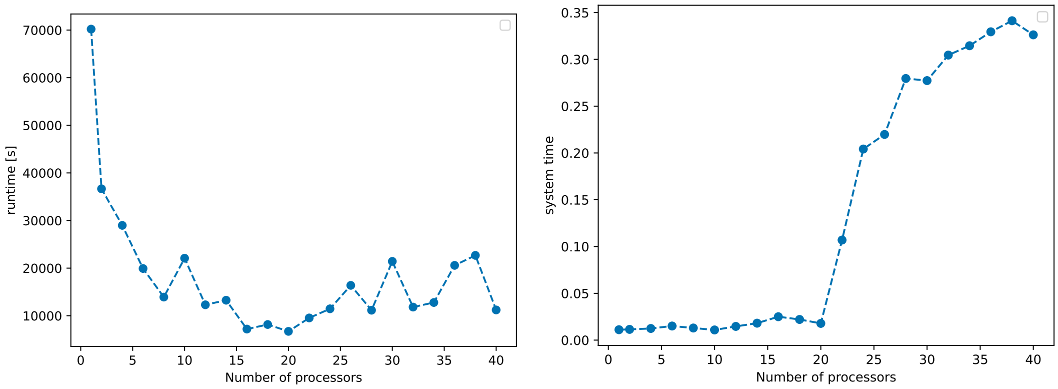


Figure III.4: *Baseline scaling test on the TaS2 benchmarking system* QUANTUM ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`

reason for this can be seen in the plots for system time, as bad parallelization shows itself by introducing waiting times between processors, which makes the system time in some way grow with the the number of processors, as can be seen especially well in III.4. Here the percentage of time used for tasks not directly related to calculations goes from a near constant value for under 20 processors to 35% of the execution time for the TaS2 system.

These scaling tests pose now two questions to be answered:

- Is better scaling on a single node possible?
- How can acceptable scaling over more than one node be achieved?

III.2 Testing different compilers and mathematical libraries

A first strategy for solving issues with parallelization is trying different compilers and mathematical libraries. As discussed in sec. II.2.1, Quantum ESPRESSO can make use of a variety of software packages available on the PHYSnet cluster. The benchmarks in III.5 are run with the following software combinations:

- OpenMPI 4.1.0 and Quantum ESPRESSO provided BLAS/LAPACK, so the baseline test discussed in sec. III.1
- OpenMPI 4.1.0 and OpenBLAS 0.3.20
- OpenMPI 4.1.0, OpenBLAS 0.3.20 and ScaLAPACK 2.2.0
- Intel oneAPI 2021.4

analysis with right plots for ompi, ompi+openblas, maybe kick out scalapack? it should be the same

Fig. III.5 shows that using Intels Intel oneAPI packages provides optimal scaling for up to round about 6 processors, which means those calculations ran about twice as fast as the calculations with just OpenMPI. This result not only stands for itself as a statement about scaling on a single node, but also provides a basis for scaling beyond this apparent optimal range of 6 processors The k point parallelization explained in sec. II.2.2 can distribute the workload in such a way that processor pools of sizes within this range work on individual k points and as such can provide optimal scaling within one pool while also not losing performance because the pools do not need to communicate with each other in the same order of magnitude as the pools have to communicate within themselves.

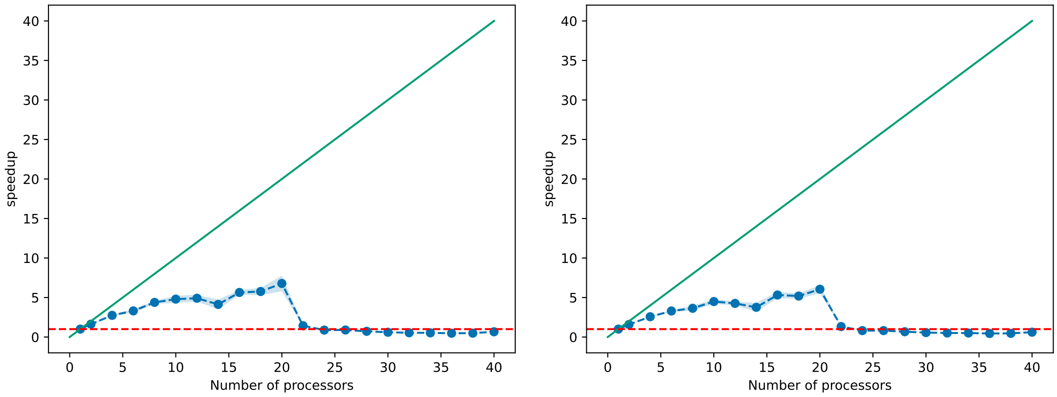
absolute runtimes! so comparison is possible! (singlecore runtimes could also be differen!)

III.3 Using the parallelization parameters of Quantum ESPRESSO

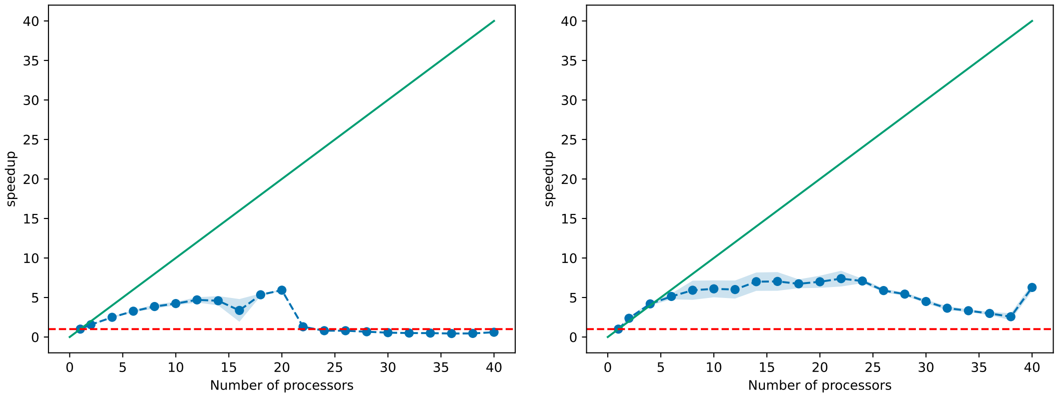
As detailed in section II.2.2, Quantum ESPRESSO offers ways to manage how the workload is distributed among the processors. In `pw.x` the default plane wave parallelization, k-point-parallelization and linear-algebra parallelization are implemented.

III.3.1 k point parallelization

The benchmark pictured in III.6 is set up as follows: for a given number of processors N_p, the parameter N_k splits the N_p processors into N_k processors pools. As the number of processors in one pool has to be a whole number, only certain combinations of N_p and N_k are possible, for example $N_p = 32$ could be split into processor pools of size 2 with $N_k = 16$, size 8 with $N_k = 4$ or size 16 with $N_k = 2$. This leads to choosing the size of the processor pools as a variable, not the parameter `nk`. Fig. III.6 shows the scaling for poolsizes 2, 8 and 16 for Quantum ESPRESSO being compiled with OpenMPI/Scalapack and Intel oneAPI. This choice of pool



(a) *OpenMPI* **(b)** *OpenMPI + OpenBLAS*



(c) *OpenMPI + ScaLAPACK* **(d)** *Intel oneAPI*

Figure III.5: *Baseline scaling test Si benchmarking system with different combinations of compilers and mathematical libraries*

sizes showcases the smallest pool size possibly (namely 2), as well as a bigger pool size with 16, that still gives rise to a few data points over the chosen range of processors.

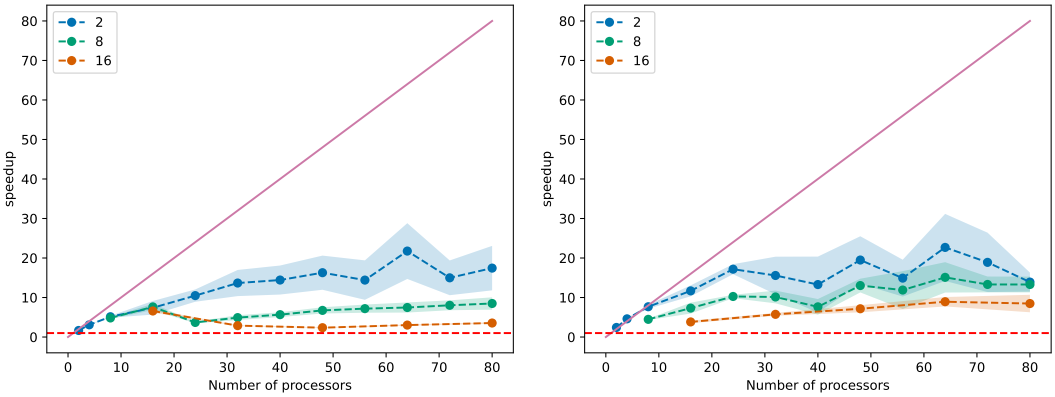
Fig. III.6 shows that using k parallelization with a pool size of 2 significantly improves the scaling behavior, not only on one node, but especially over more than one node.

more analysis: difference between poolsizes

Another important conclusion to draw out of fig. III.6 is the impact of using Intels compiler instead of OpenMPI, as that factor alone speeds up the calculation by a factor of 2 over the whole range of processors.

The same scaling test is applied to the TaS2 system in fig. III.7, with a similar list of pool sizes, but over a wider range of processors.

Remarkably, the scaling behavior is swapped in comparison to III.6, as the pool size 2 saturates fast and the bigger pool sizes show way better scaling behavior. Furthermore, there are instances of better than linear scaling, which according to Quantum ESPRESSO docs can be attributed to better caching of data.



(a) OpenMPI + ScaLAPACK **(b)** Intel oneAPI

Figure III.6: *Benchmark with k-point parallelization for the Si benchmarking system with 3 different sizes of processor pools*

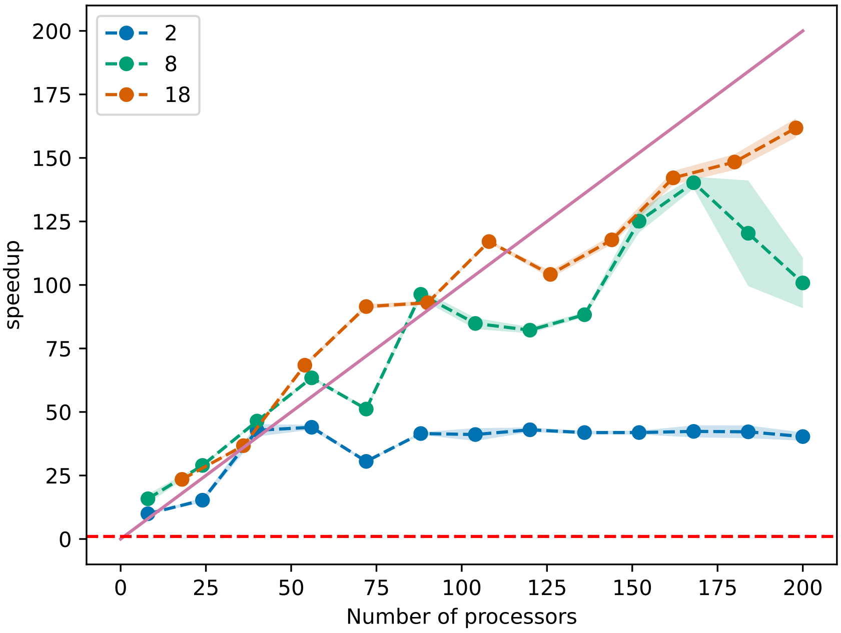


Figure III.7: *Benchmark with k-point parallelization for the TaS2 benchmarking system*

It can also be instructive to look at the idle time for this benchmark to judge the quality of parallelization.

Fig. III.8 shows a distribution of idle times between about 4 % and 6 % of the whole wall time, without any kind of systemic increase over any range of processors.

more analysis: difference between poolsizes

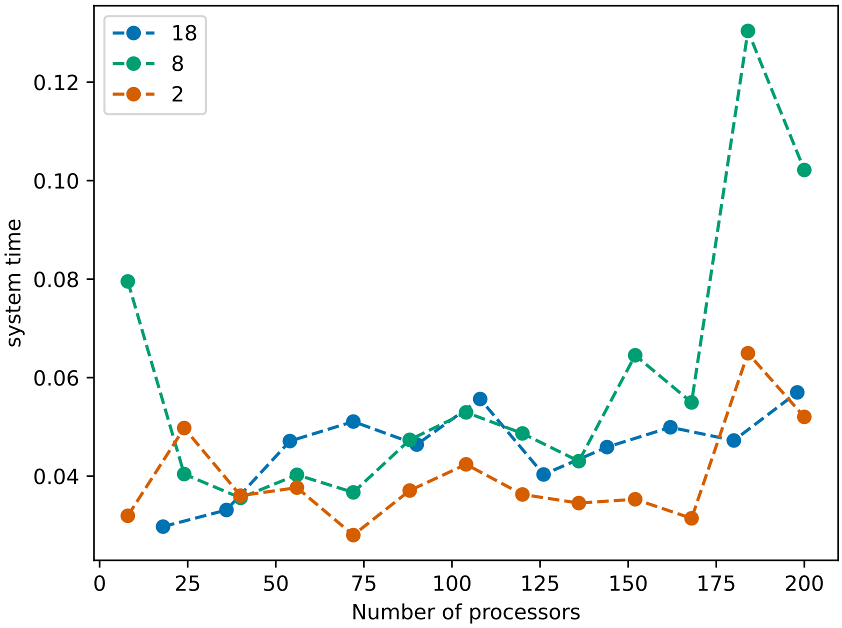


Figure III.8: *Idle time for the k point parallelization benchmark for the TaS2 system*

III.3.2 Linear algebra parallelization

Linear Algebra

III.4 Comparison with calculations on the HLRN cluster

III.5 Conclusion: Parameters for optimal scaling

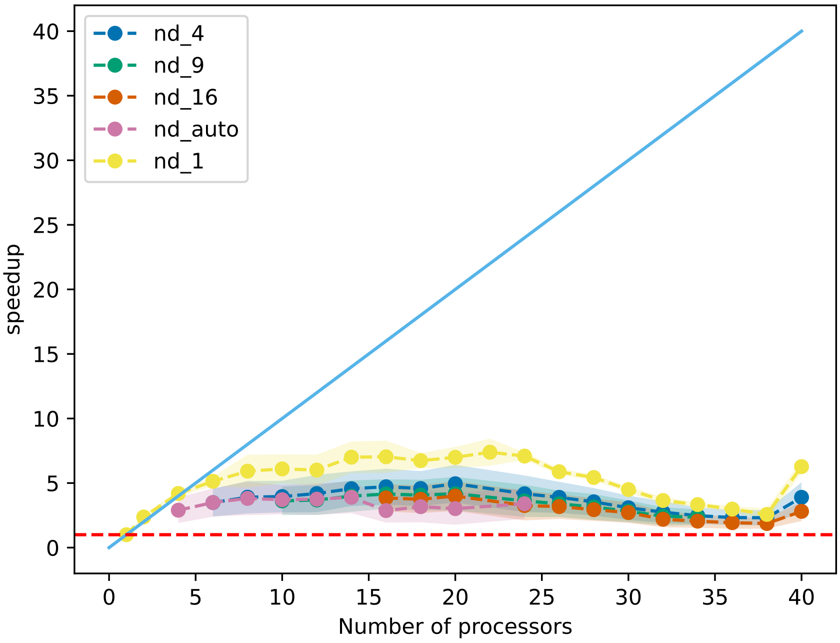


Figure III.9: *Benchmark with linear algebra parallelization for the silicon benchmarking system*

IV Parallelization of DFPT calculations

IV.1 Optimal parallelization parameters for DFPT calculations

IV.1.1 k point parallelization

IV.1.2 Linear algebra parallelization

IV.2 Image parallelization

IV.3 Conclusion: Parameters for optimal scaling

Bibliography

[1] N. Marzari. "Ab-initio Molecular Dynamics for Metallic Systems". PhD thesis. University of Cambridge, 1996.

[2] P. Hohenberg and W. Kohn. "Inhomogeneous Electron Gas". In: *Phys. Rev.* 136.3 (Nov. 1964). Publisher: American Physical Society, B864–B871. DOI: `10.1103/PhysRev.136.B864`.

[3] W. Kohn and L. J. Sham. "Self-Consistent Equations Including Exchange and Correlation Effects". In: *Phys. Rev.* 140.4 (Nov. 1965). Publisher: American Physical Society, A1133–A1138. DOI: `10.1103/PhysRev.140.A1133`.

[4] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers.* 0th ed. CRC Press, July 2, 2010. ISBN: 978-1-4398-1193-1. DOI: `10.1201/EBK1439811924`.

[5] P. Giannozzi et al. "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials". In: *Journal of Physics: Condensed Matter* 21.39 (Sept. 2009). Publisher: IOP Publishing, p. 395502. DOI: `10.1088/0953-8984/21/39/395502`.

[6] P. Giannozzi et al. "Advanced capabilities for materials modelling with Quantum ESPRESSO". In: *Journal of Physics: Condensed Matter* 29.46 (Oct. 2017). Publisher: IOP Publishing, p. 465901. DOI: `10.1088/1361-648x/aa8f79`.

[7] *Quantum ESPRESSO User's Guide (v. 7.0).* URL: `https://www.quantum-espresso.org/documentation/package-specific-documentation/` (visited on 05/23/2022).

Listings

List of Figures

II.1 Flowchart of the algorithm to solve the KS equations 7

III.1 Baseline scaling test on the Si benchmarking system *Quantum ESPRESSO 7.0, OpenMPI 4.1.0,* `nk 1` *and* `nd 1` 9
III.2 Baseline scaling test on the Si benchmarking system *Quantum ESPRESSO 7.0, OpenMPI 4.1.0,* `nk 1` *and* `nd 1` 10
III.3 Baseline scaling test on the TaS2 benchmarking system *Quantum ESPRESSO 7.0, OpenMPI 4.1.0,* `nk 1` *and* `nd 1` 11
III.4 Baseline scaling test on the TaS2 benchmarking system *Quantum ESPRESSO 7.0, OpenMPI 4.1.0,* `nk 1` *and* `nd 1` 11
III.5 Baseline scaling test Si benchmarking system with different combinations of compilers and mathematical libraries 13
III.6 Benchmark with k-point parallelization for the Si benchmarking system with 3 different sizes of processor pools 14
III.7 Benchmark with k-point parallelization for the TaS2 benchmarking system . . 14
III.8 Idle time for the k point parallelization benchmark for the TaS2 system 15
III.9 Benchmark with linear algebra parallelization for the silicon benchmarking system 16

List of Tables

Acknowledgement