SIRIUS: domain specific library prototype for electronic structure calculations

Trieste, January 14th 2015
Anton Kozhevnikov, CSCS
Introduction

PRACE-2IP WP8: how to approach the problem of community code refactoring?
SIRIUS is a prototype low level library developed under PRACE-2IP work package 8 (community codes refactoring and optimization) as a general-purpose solution for optimization and scaling of both Exciting and Elk full-potential LAPW codes.
Introduction

SIRIUS C++ library

- MPI + OpenMP parallel model with GPU acceleration
- LAPW specific functionality
- Pseudopotential specific functionality
- Common objects: unit cell description, reciprocal lattice description, FFT mesh, G-vector indexing, radial functions (local orbitals or beta projectors) indexing, XC potential generation, etc.

Low level libraries

- GNU scientific library
- HDF5
- spglib
- LibXC
- LAPACK and BLAS
- ScaLAPACK and PBLAS
- FFTW
- ELPA
- MAGMA
Research topics

- Wave-function distribution
- Iterative solvers:
  - Davidson solver (serial & parallel, CPU & GPU version)
  - Chebyshev-filtered subspace
  - RMM-DIIS
- Real-space beta projectors
- Second-variational approach to magnetism
Wave-function distribution

2D BLACS grid

[0, 0]  [0, 1]

[1, 0]  [1, 1]

Index of G-vectors

Index of bands

Wave-functions
Wave-function distribution

Split bands

Split G-vectors

Split bands and G-vectors

2D block-cyclic distribution
2D block cyclic distribution

- “ScaLAPACK friendly”

\[ \langle \phi_i | \hat{H} | \phi_j \rangle = \langle \phi_i | \hat{H} | \phi_j \rangle \]

- Task groups are introduced naturally
- Allows fast change of wave-function distribution
Changing wave-function distribution

MPI all-to-all along rows

MPI all-to-all along columns

split bands

split G-vectors
First, block-cyclic distributed wave-functions are gathered into ‘slices’ on individual nodes and then each node executes the whole FFTs on a subset of full wave-function vectors.
SiO ANA zeolite benchmark

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of atoms</td>
<td>144</td>
</tr>
<tr>
<td>Number of k-points</td>
<td>8 (nosym)</td>
</tr>
<tr>
<td>Number of bands</td>
<td>461</td>
</tr>
<tr>
<td>Number of beta-projectors</td>
<td>2112</td>
</tr>
<tr>
<td>( \text{ecutwfc} )</td>
<td>30 Ry</td>
</tr>
<tr>
<td>( \text{ecutrho} )</td>
<td>400 Ry</td>
</tr>
<tr>
<td>(</td>
<td>G+k</td>
</tr>
<tr>
<td>Coarse FFT mesh</td>
<td>90x90x90</td>
</tr>
<tr>
<td>Fine FFT mesh</td>
<td>180x180x180</td>
</tr>
</tbody>
</table>
Platform and code

Piz Daint: Intel 8-core SB @2.6 GHz + Tesla K20X, Aries interconnect

Latest svn version of QE (+ELPA_2013.11 +SIRIUS)

Latest git version of QE+GPU
Performance benchmark

Scaling

- QE (sirius)
- QE (many threads)
- QE (many ranks)
- Ideal

"Many ranks" run

<table>
<thead>
<tr>
<th>ranks per k-point</th>
<th>% of ranks for diagonalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>50%</td>
</tr>
<tr>
<td>32</td>
<td>78%</td>
</tr>
<tr>
<td>72</td>
<td>89%</td>
</tr>
<tr>
<td>128</td>
<td>94%</td>
</tr>
</tbody>
</table>
Performance benchmark

Time to solution

- QE (many threads, CPU)
- QE (many ranks, CPU)
- QE (sirius, CPU)
- QE+GPU
- QE (sirius, GPU)

Number of CPU sockets (cores):
- 8 (64)
- 32 (256)
- 72 (576)
- 128 (1024)

Time to solution (sec):
- 0
- 1240
- 2480
- 3100

Values for different configurations:
- 8 (64):
  - QE (many threads, CPU): 3011
  - QE (many ranks, CPU): 2639
  - QE (sirius, CPU): 1243
  - QE (sirius, GPU): 365
  - QE+GPU: 1081
- 32 (256):
  - QE (many threads, CPU): 1240
  - QE (many ranks, CPU): 777
  - QE (sirius, CPU): 546
  - QE (sirius, GPU): 377
  - QE+GPU: 359
- 72 (576):
  - QE (many threads, CPU): 359
  - QE (many ranks, CPU): 217
  - QE (sirius, CPU): 222
  - QE (sirius, GPU): 207
- 128 (1024):
  - QE (many threads, CPU): 133
  - QE (many ranks, CPU): 154
Valence charge density has to be augmented with:

$$\tilde{\rho}(G) = \sum_{\alpha} \sum_{\xi \xi'} q_{\xi \xi'}^{\alpha} Q_{\xi' \xi}^{\alpha}(G)$$

where

$$q_{\xi \xi'}^{\alpha} = \sum_{jk} \langle \beta_{\xi}^{\alpha} | \Psi_{jk} \rangle n_{jk} \langle \Psi_{jk} | \beta_{\xi'}^{\alpha} \rangle$$

Relation between plane-wave coefficients of the Q-operator for a given atom $\alpha$ and the corresponding coefficients of the Q-operator for a given atom type $A$:

$$Q_{\xi' \xi}^{\alpha(A)}(G) = e^{-iG\tau_{\alpha(A)}} Q_{\xi' \xi}^{A}(G)$$

Split the sum over atoms into sum over atom types and inner sum over atoms of the same type:

$$\tilde{\rho}(G) = \sum_{A} \sum_{\xi \xi'} Q_{\xi' \xi}^{A}(G) \sum_{\alpha(A)} q_{\xi \xi'}^{\alpha(A)} e^{-iG\tau_{\alpha(A)}} = \sum_{A} \sum_{\xi \xi'} Q_{\xi' \xi}^{A}(G) d_{\xi \xi'}^{A}(G)$$
D-operator evaluation

- treat as matrix-vector multiplication

\[ D^\alpha_{\xi \xi'} = \sum_G Q^\alpha_{\xi \xi'}(G) V(G) \]

- (even better) treat as matrix-matrix multiplication (to be tested)

\[ D^\alpha_{\xi \xi'} = \sum_G Q^A_{\xi \xi'}(G) e^{-iG \tau\alpha(A)} V(G) \]
Thank you for your attention.