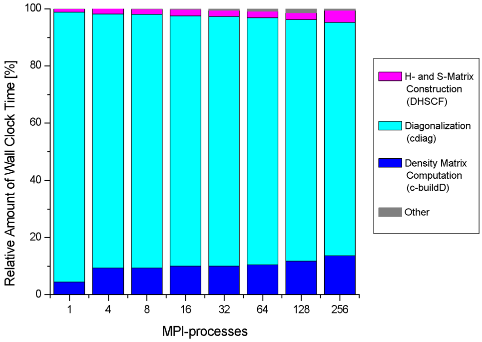
Sakurai-Sugiura algorithm based eigenvalue solver for Siesta

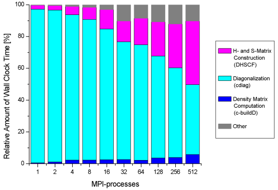
Georg Huhs

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Motivation

Timing analysis for one SCF-loop iteration:





left: CNT/Graphene, right: DNA

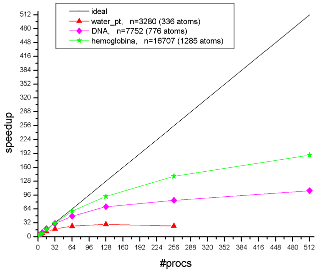
Siesta

Specifics

- High fraction of EVs needed (e.g. 30%)
- Sparse matrices

Actual solver: ScaLAPACK

- Dense solver
- Limited scaling



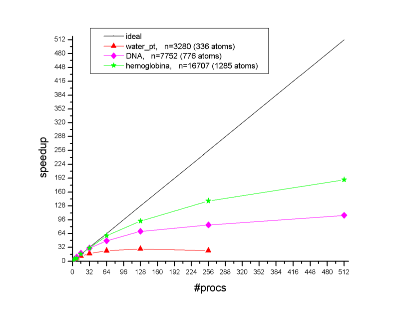
Siesta

Specifics

- High fraction of EVs needed (e.g. 30%)
- Sparse matrices

Actual solver: ScaLAPACK

- Dense solver
- Limited scaling



Characteristics

SS-method, Tetsuya Sakurai and Hiroshi Sugiura, 2002

- Solver for GEV $A\,\mathbf{x} = \lambda\,B\,\mathbf{x}$
  $A, B \in \mathbb{C}^{n \times n}$
- Projection method, based on complex contour integration
- Meant for finding EVs in a certain domain
- Suitable for
  - large, sparse matrices
  - parallelisation

Basic steps

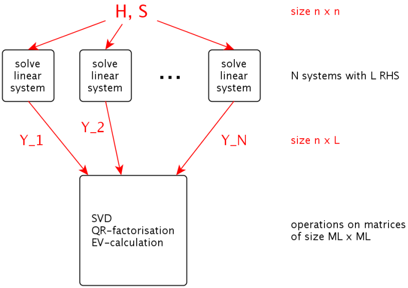
Divide domain of interest into subdomains, in each:

1. Solve a set of linear systems
2. Use results for numerical integrations
3. Construct subspace
4. Find eigenvalues/-vectors in subspace and backtransform
5. Select correct eigenpairs

Parallelisation

Three levels of parallelisation

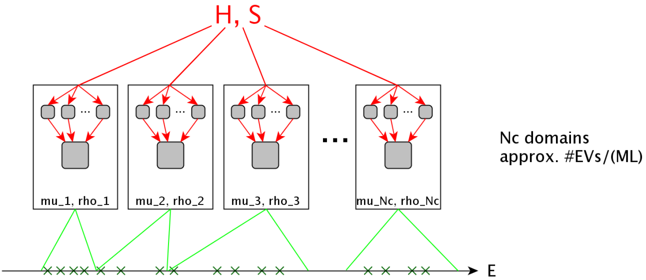
1. Handling subdomains in parallel
2. Solving N linear systems in parallel
3. Use parallel solver for each linear system



Parallelisation

Three levels of parallelisation

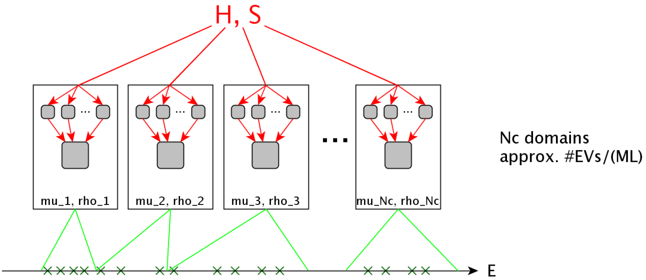
1. Handling subdomains in parallel
2. Solving N linear systems in parallel
3. Use parallel solver for each linear system



Parallelisation

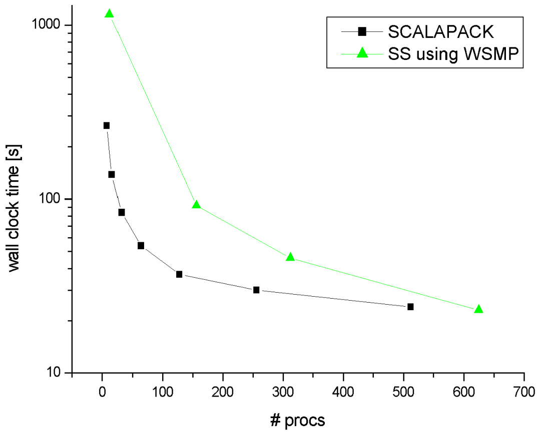
Three levels of parallelisation

1. Handling subdomains in parallel
2. Solving N linear systems in parallel
3. Use parallel solver for each linear system



Timing expectations

Example: DNA, 776 atoms, $n = 7752$



Using GPUs

Implemented: Solving the linear systems with CUSP.

Algorithm: gmres without preconditioner!

Gain: Speedup of 3-4 compared to WSMP (for the smaller examples)

Timing solving all lin. systems in parallel (per subdomain)

Problem	Setup	Solve	Comm	total	ScaLAPACK
water_pt ($n = 3280$)	4.0	3.7	0.1	11	7
DNA ($n = 7752$)	3.3	7.6	0.5	13	∼20
hemoglobina ($n = 16707$)	8.0	10	1.0	**22**	**∼70**

All times in seconds

Conclusions

Issues

- Inefficient in serial mode
- Memory parallelisation only in linear solver

Benefits

- Robust algorithm, parameterisation can be hidden from user
- **Actual problems:** SS can be faster than ScaLAPACK, depending on scaling of lin. solver
- **Future problems:** high potential for larger numbers of processors
- Possibility to use many GPUs efficiently

Conclusions

Issues

- Inefficient in serial mode
- Memory parallelisation only in linear solver

Benefits

- Robust algorithm, parameterisation can be hidden from user
- **Actual problems:** SS can be faster than ScaLAPACK, depending on scaling of lin. solver
- **Future problems:** high potential for larger numbers of processors
- Possibility to use many GPUs efficiently

Thanks to

Alberto García, Pablo Ordejón (Siesta)
Tetsuya Sakurai (Algorithm)
Anshul Gupta (WSMP)

BSC support team
José María Cela (BSC)

More details (1)

1. Define EV-range $\Rightarrow \gamma, \rho$ and a set of vectors $V \in \mathbb{C}^{n \times L}$.
2. Set quatrature points:

$$z_j = \gamma + \rho e^{\left(2\pi \mathrm{i}\left(j+\frac{1}{2}\right)\frac{1}{N}\right)}, \qquad j = 0, \ldots, N-1$$

3. Solve the linear systems

$$(z_j B - A)\, Y_j = B\, V$$

   - N systems, L right hand sides each
   - independent!

More details (2)

4 Numerical integration

$$S_k = \frac{1}{N} \sum_{j=0}^{N/2-1} \left(\frac{z_j - \gamma}{\rho} \right)^{k+1} Y_j, \quad k = 0, \ldots, M-1$$

Matrix $S = [S_0, S_1, \ldots, S_{M-1}] \quad \in \mathbb{C}^{n \times ML}$

5 Find the rank K of S with an SVD and shrink it to $n \times K$

6 QR-decomposition of $S \Rightarrow$ orthogonal basis Q

7 $\tilde{A} = Q^T (A - \gamma B)\, Q$
$\tilde{B} = Q^T B\, Q$ $\qquad \tilde{A}, \tilde{B} \in \mathbb{C}^{K \times K}$

8 Eigenpairs of $\tilde{A}\tilde{\mathbf{x}} = \tilde{\lambda}\tilde{B}\tilde{\mathbf{x}}$

More details (3)

8. Eigenpairs of $\tilde{A}\tilde{\mathbf{x}} = \tilde{\lambda}\tilde{B}\tilde{\mathbf{x}}$
9. Eigenpairs of the original problem

$$\lambda_j = \tilde{\lambda}_j + \gamma$$
$$\mathbf{x}_j = Q \cdot \tilde{\mathbf{x}}_j$$

10. Remove exterior eigenvalues and ghosts.