20th NEXT Workshop, Kyoto TERSA
2015 January 13

A Spike Algorithm for Solving Tridiagonal Block Matrix Equations on a Multicore Machine

Shinji TOKUDA

Research Organization for Information Science and Technology

Acknowledgement:
N. Aiba (JAEA), M. Yagi (JAEA)

Abstract

We are developing a recursive direct-solver for linear equations with the coefficient matrix of a tridiagonal block form by adopting the spike algorithm.
The solver is parallelized by open MP that supports sections and task constructs. The block structure in the coefficient matrix naturally leads to nested parallelism.
Some tests are reported on the performance of the parallelized solver, which shows the spike algorithm is expected to provide an efficient solver on modern multicore machines.

Contents

- **Introduction**

- **Spike Algortithm for Tridiagonal Block Matrix**
  - ◇ What is Spike?
  - ◇ Recursive Spike Algorithm and
    Its Parallelization by open MP
  - ◇ Nested Parallelism for Spike Algorithm

- **Summary**
  - ◇ Numerical Experiments of the parallelized solever
  - ◇ Future Works

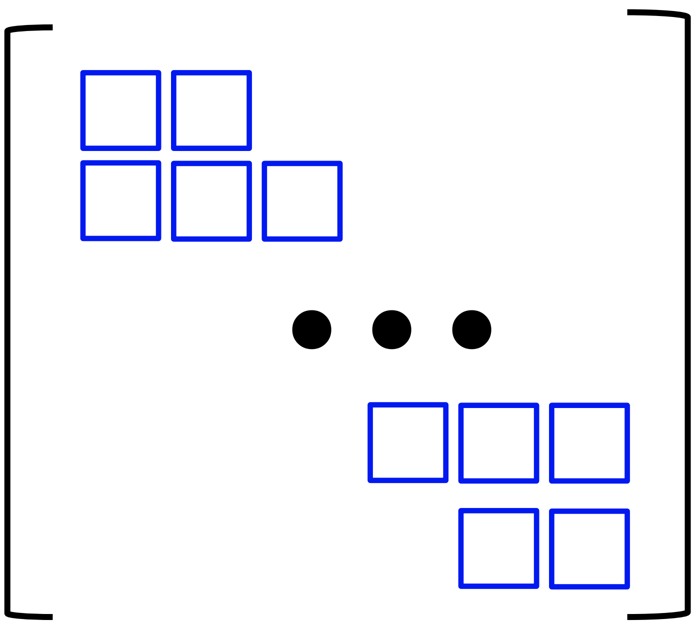
Introduction

We want to solve linear equations that have a coefficient matrix of a tridiagonal block form

(1) $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$

on a multicore machine (for example, a personal computer with an accelerator).

$\boldsymbol{A}$ =



Assumed Problem Sizes:

The size of each block (□)

(**Nblock**)^2 = $(128)^2$~ $(512)^2$

The block can be sparse or dense.

The number of diagonal blocks

msize0 = 512~1024

Such equations are familiar in computational physics. In tokamak fusion research they appear in MHD (MagnetoHydroDymanic) equilibrium and stability analyses and in nonlinear MHD simulations.
Algorithms for tridiagonal block matrix equations has been studied since 1970's, and several solvers have been developed[1].
We however consider that a new solver have to be developed in such a way that it realizes faster computaion by adopting recent developments in hardware and software; they are

①Development in hardware：multicore machines are now available (with accelerators) and are applied to image/accoustic processing and **scientific/ engineering computations**.

②Development in fortran ：fortran 90/95 has made available ***recursive*** procedure and array slice.

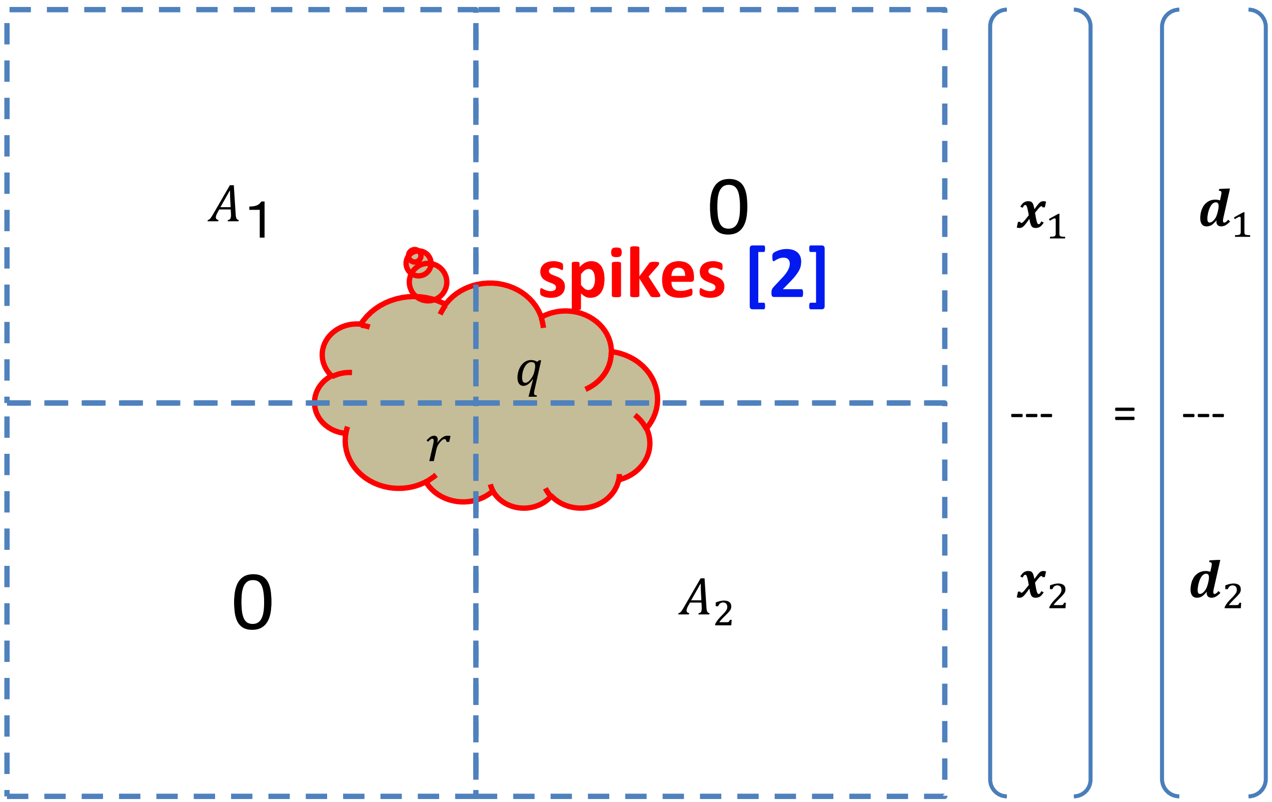
③ Development in Open MP：

Open MP 3.0 (since 2008) supports the task construct that facilitates parallelizing ***recursive*** procedures, and

Open MP 4.0 (since 2013) supports the programming of accelerators, SIMD programming and thread affinity.

In the present work we report a direct solver for the tridiagonal block matrix equations by adopting the ***spike*** algorithm[2] .
The spike is a kind of divide and conquer algorithms feasible for parallel computing.
Also the tridiagonal structure of equations enables us to continue applying the spike algorithm, resulting in a recursive solver that is coded by fortran90 and easily parallelized by the task construct in open MP.
Such features exploit the hardware and software developments and are expected to provide an efficient parallelized solver for the tridiagonal block matrix equations.

Spike Algorithm(1) What is spike?



(1)

Operate $\mathrm{Diag}(\mathbf{A}_1^{-1}, \mathbf{A}_2^{-1})$ on the both sides from the left

Spike Algorithm(2): Reduction in Equation

$$\left(\begin{array}{c|c} I_1 & \begin{matrix} v_1 \\ \vdots \\ \vdots \\ v_m \end{matrix} \quad 0 \\ \hline 0 \quad \begin{matrix} v_{m+1} \\ \vdots \\ \vdots \\ v_N \end{matrix} & I_2 \end{array}\right) \begin{pmatrix} \boldsymbol{x}_1 \\ --- \\ \boldsymbol{x}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}_1 \\ --- \\ \boldsymbol{g}_2 \end{pmatrix} \tag{2}$$

$$\boldsymbol{v} = (v_1, \cdots, v_m)^t, \quad \mathbf{w} = (v_{m+1}, \cdots, v_N)^t \tag{3}$$

$$A_1 \boldsymbol{v} = (0, \ \cdots, \ q)^t, \quad A_1 \boldsymbol{g}_1 = \boldsymbol{d}_1$$

$$A_2 \boldsymbol{w} = (r, \ \cdots, \ 0)^t, \quad A_2 \boldsymbol{g}_2 = \boldsymbol{d}_2$$

These equations can be solved in parallel by the ***sections construct*** of open MP.

Spike Algorithm(3)
Construction of the solution

$$(I - v_m v_{m+1}) x_m = g_m - v_m g_{m+1}$$

$$x_{m+2} = g_{m+1} - v_{m+2} x_m$$

$$\vdots$$

$$x_N = g_N - v_N x_m$$

$$(I - v_{m+1} v_m) x_{m+1} = g_{m+1} - v_{m+1} g_m$$

$$x_1 = g_1 - v_1 x_{m+1}$$

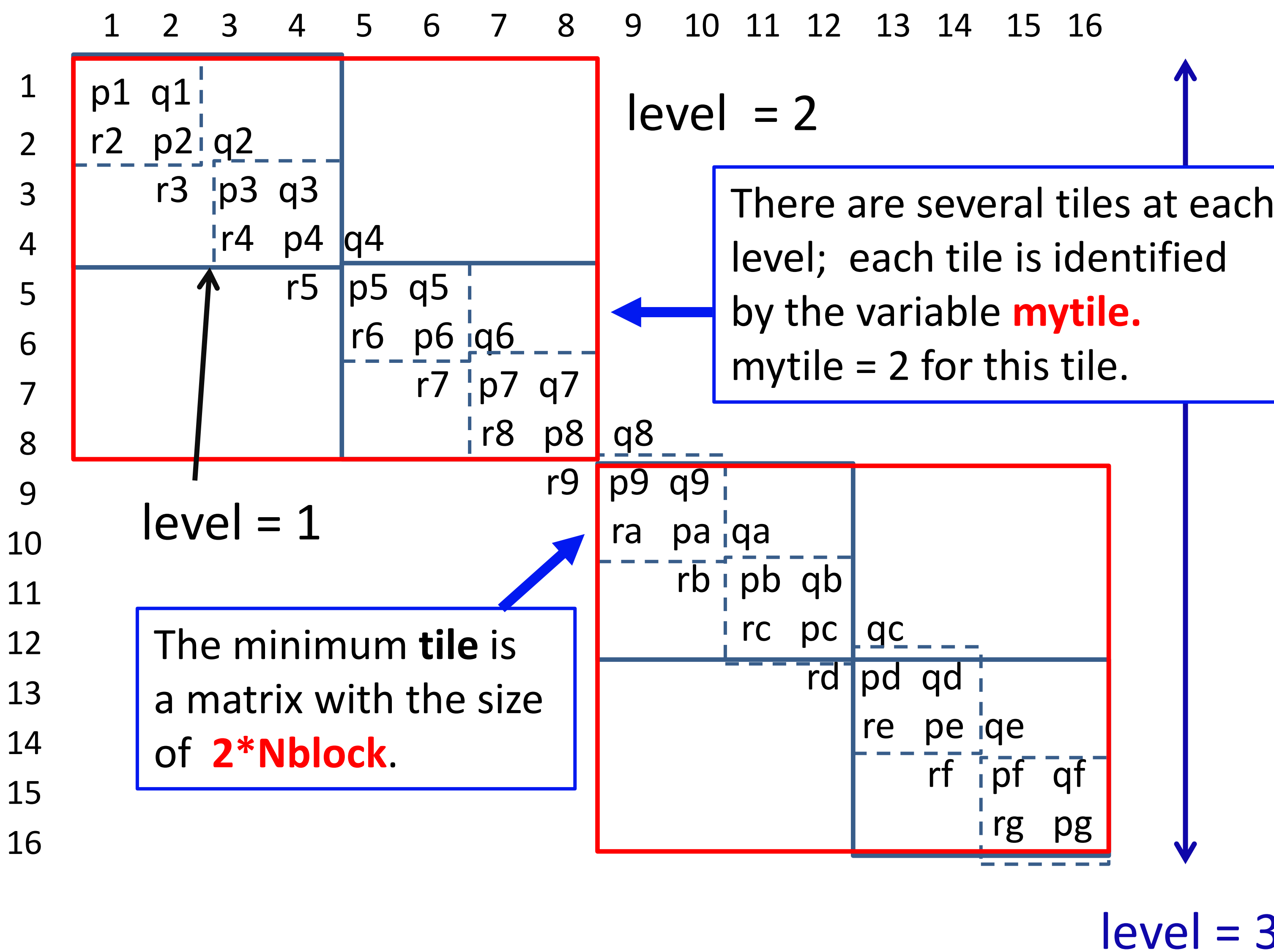
$$\vdots$$

$$x_{m-1} = g_{m-1} - v_{m-1} x_{m+1}$$

Parallelization is made available
by the sections construct in Open MP.

If the original matrix ***A*** has no further structure of the slide #7, then the spike reduction in the equations finishes there.
However since the matrix ***A*** is tridiagonal, the reduced matrices $\boldsymbol{A_1}$, $\boldsymbol{A_2}$ are again tridiogonal and have the spike structures, which we call the ***self-similar*** spike structure of a tridigonal matrix.
Such *self-similarity* enables us to continue applying the spike algorithm.

We follow two technical terms: **level and tile [3].**



There are several tiles at each level; each tile is identified by the variable **mytile.** mytile = 2 for this tile.

The minimum **tile** is a matrix with the size of **2*Nblock**.

Recursive Spike Algorithm

The self-similar spike structure of a tridiagonal block matrix makes it possible to develop a recursive solver by fortran 90. Parallelization of the recursive procedure can be implemented by the task construct of open MP.

```
RECURSIVE subroutine MultiLevelSpike (...,  mlevel, mytile, ... )
     ...  ... ...
 mytile0 = mytile
 mytile= 2*(mytile0-1)+1
    call MultiLevelSpike (...,  mlevel-1, mytile, ... )

 mytile= 2*(mytile0-1)+2
    call MultiLevelSpike (...,  mlevel-1, mytile, ... )
  ...  ... ...
end subroutine MultiLevelSpike
```

Parallelization by the task construc of open MP

A Feature Characteristic of Block Matrices from the View Point of the Spike Algorithm

$$\left[\begin{array}{c|c} A_1 & 0 \\ & q \\ \hline r & \\ 0 & A_2 \end{array}\right] \begin{bmatrix} \boldsymbol{x}_1 \\ \text{---} \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_1 \\ \text{---} \\ \boldsymbol{d}_2 \end{bmatrix}$$

One application of the spike algorithm yields the following two systems of equations

$$A_1[\boldsymbol{v}, \boldsymbol{g}_1] = \begin{bmatrix} 0 & \\ \vdots & \boldsymbol{d}_1 \\ \boldsymbol{q} & \end{bmatrix}$$

q, r are matricies with the size of **Nblock**

$$A_2[\boldsymbol{w}, \boldsymbol{g}_2] = \begin{bmatrix} \boldsymbol{r} & \\ \vdots & \boldsymbol{d}_2 \\ 0 & \end{bmatrix}$$

If the level descends by one, two systems of equations are generated; the number of equations increases by Nblock for each system.

Nested Parallelism for Spike Algorithm

Let **NumEq0** be the number of equations to be solved at the first level = maxlevel.
The spike reduction finally yields **Ntile = $2^{(maxlevel-1)}$** tiles at level = 1, each of which has

$$\mathbf{NumEq = (maxlevel - 1)*Nblock + Numeq0}$$

equations with coefficient matrices of the size $(Nblock)^2$; they may be fine-grained.
However the number of equations can be used as the loop index of parallelized do loops in a task (nested parallelism); such do loops may be coarse-grained.
The set of Numeq equations at level = 1 will be solved by the ***threaded*** Intel lapack.

Example of the nested parallelism :
maxlevel = 9, Nblock = 128, Numeq0 = 1

	Ntile	Numeq
	256	***1025***
	open MP	**thread**
parallelism	**sections, task**	**Intel lapack**

Present Status of the Code Development

(1) Single Version has been developed.
(2) Parallelization by sections and task constructs has been finished.

Numerical Experiments

Computer

CPU (Dual) :
Intel Xeon E5-2697v2
(2.70GHz, **12** cores) **× 2**

OS : Linux (CentOS)
Fortran compiler : **Intel ifort**
Lapack routines: **Intel MKL**

Results of experiments

for Nblock = 128, Numeq0 = 2

	CPUtime (sec)		Performance
msize0	Single	Multi(*)	
256	6.748	3.076	2.19
1024	39.64	18.34	2.16

(*) generates only **two tasks** in parallel by the task construct

Future Works

(1) Performance tests of ***threaded*** Intel MKL lapack

(2) Adaptation of the solver to MIC (Many Integrated Core)

References

[1] *for example*, S.P. Hirshman, K.S. Perumalla, V.E. Lynch, R. Sanchez, BCYCLIC: A parallel block tridiagonal matrix solver, J. Computational Physics **229** (2010) 6392.

[2] E. Polizzi, A. Sameh, A parallel hybrid banded system solver: the SPIKE algorithm, Parallel Computing 32 (2006) 177.

[3] J.C. Brodman, et al., A Parallel Numerical Solver Using Hierarchically Tiled Arrays, in Proc. of the Intl. Workshop on Languages and Compilers for Parallel Computing, pp.46-61, Springer, Berlin, 2011.