Square Block Code for Positive Definite Symmetric Cholesky Band Routines

Fred G. Gustavson¹, José R. Herrero², E. Morancho²

¹IBM T.J. Watson Research Center, Emeritus, and Umeå University
fg2935@hotmail.com

²Computer Architecture Department
Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
{josepr,enricm}@ac.upc.edu

10th Int. Conf. on Parallel Processing and Applied Mathematics (PPAM’13)

Warsaw, Poland, Sep. 10th, 2013
Outline

Introduction

Upper Square Block Band Format

Ongoing work

Conclusions
Outline

Introduction
Upper Square Block Band Format
Ongoing work
Conclusions
Overview: Band matrix Storage
Symmetric Positive Definite Matrix

- Dense storage

- $j^{th}$ diagonal element in $A(j, j)$
Overview: Band matrix Storage
Symmetric Positive Definite Matrix

- Dense storage

  - $j^{th}$ diagonal element in $A(j, j)$
  - LAPACK Lower Band storage (UPLO = 'L')

  j-th column of $A$ is stored in the j-th column of $AB$ as
  - Diagonal element $A(j, j)$ stored in $AB(1, j)$
Overview: Band matrix Storage

Symmetric Positive Definite Matrix

- Dense storage

- $j^{th}$ diagonal element in $A(j,j)$

- LAPACK Lower Band storage (UPLO = 'L')

**j-th column of A is stored in the j-th column of AB as**

- Diagonal element $A(j,j)$ stored in $AB(1,j)$

The rectangle holds $P+T$:

- A parallelogram $P$

- A lower isosceles triangle $T$ of side equal to $kd$ ($kd$ is the half bandwidth)
Overview: Band matrix Storage

Details

- Dense storage

Access value in $i^{th}$ row and $j^{th}$ column of A as $A(i, j)$
- $j^{th}$ diagonal element in $A(j, j)$
Overview: Band matrix Storage

Details

- Dense storage

Access value in $i^{th}$ row and $j^{th}$ column of $A$ as $A(i, j)$
  - $j^{th}$ diagonal element in $A(j, j)$
- LAPACK Lower Band storage (UPLO = 'L')

$j$-th column of $A$ is stored in the $j$-th column of $AB$ as
  - $AB(1 + i - j, j) = A(i, j)$ for $j \leq i \leq \text{min}(n, j + kd)$
  - Diagonal element $A(j, j)$ stored in $AB(1, j)$
Overview: Previous Work


**Goal:**
Improve Programmability of LAPACK Lower Band Cholesky

- Change Leading Dimension of $AB$ from $LDAB$ to $LDAB - 1$ in the array declaration of $AB$
- Tells the compiler that the distance in the 2\textsuperscript{nd} dimension is one less.
- As a result one can write $AB(i, j)$ to access value $A(i, j)$
Goals

Can we do any better?

Improve Programmability . . . but also
  ▶ Improve data locality
  ▶ Ease parallelization

How?

Rearrange data so that
  ▶ No further data copies or transformations are necessary at computation time
  ▶ Data off-loading is more efficient
  ▶ Parallelization is easier and more efficient with an optimal amount of space
Outline

Introduction

Upper Square Block Band Format

Ongoing work

Conclusions
Related work

Lower Blocked Column Packed and Upper Square Blocked Packed Formats


An extended version can be retrieved as LAPACK Working Note 249
Related work

Lower Blocked Column Packed and Upper Square Blocked Packed Formats

Lower Blocked Packed Format

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>19</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>20</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>21</td>
<td>27</td>
</tr>
</tbody>
</table>
### Related work

**Lower Blocked Column Packed and Upper Square Blocked Packed Formats**

#### Upper Blocked Packed Format

<table>
<thead>
<tr>
<th></th>
<th>0 2</th>
<th>4 6</th>
<th>8 10</th>
<th>12 14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9 11</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>30</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>33</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Upper BPF is the preferred format**
## Transition

Upper Square Blocked Packed Format used to store a band matrix

A band matrix stored in Upper Blocked Packed Format

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>*</th>
<th>*</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>30</td>
<td>32</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>33</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A band matrix stored in Upper Blocked Packed Format

|   | 0  | 2  | 4  | 6  | *  | *  | *  | *  | 3  | 5  | 7  | 9  | *  | *  | 16 | 18 | 20 | 22 | *  | *  | 19 | 21 | 23 | 25 | *  | 28 | 30 | 32 | 34 | 31 | 33 | 35 | 36 | 38 | 39 |

Umm! There is unused space! Can we do any better?
Towards an Upper Square Block Band format

A band matrix stored in Upper Blocked Packed Format

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>*</th>
<th>*</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>*</td>
<td>25</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>30</td>
<td>32</td>
<td>34</td>
<td>36</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>33</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Towards an Upper Square Block Band format

A band matrix stored in Upper Blocked Packed Format

\[
\begin{array}{ccccccc}
0 & 2 & 4 & 6 & * & * & * \\
1 & 3 & 5 & 7 & 9 & * & * \\
16 & 18 & 20 & 22 & * & * & 9 \\
17 & 19 & 21 & 23 & 25 & * & \\
28 & 30 & 32 & 34 & 36 & 38 & 39 \\
31 & 33 & 35 & & & & \\
\end{array}
\]

Value at @9 can be stored in @1
Value at @25 can be stored in @17
Towards an Upper Square Block Band format

A band matrix stored in Upper Blocked Packed Format

<table>
<thead>
<tr>
<th>0 2</th>
<th>4 6</th>
<th>* *</th>
<th>* *</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
<td>5 7</td>
<td>9 *</td>
<td>* *</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16 18</th>
<th>20 22</th>
<th>* *</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 19</td>
<td>21 23</td>
<td>25 *</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>28 30</th>
<th>32 34</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>33 35</td>
</tr>
<tr>
<td>36 38</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
</tr>
</tbody>
</table>

Value at @9 can be stored in @1
Value at @25 can be stored in @17
Empty blocks need not be stored
Towards an Upper Square Block Band format

A band matrix stored in Upper Blocked Packed Format

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>*</th>
<th>*</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>30</td>
<td>32</td>
<td>34</td>
<td>36</td>
<td>38</td>
<td>39</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>33</td>
<td>35</td>
<td>36</td>
<td>38</td>
<td>39</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Value at @9 can be stored in @1
Value at @25 can be stored in @17
Empty blocks need not be stored
Space savings w.r.t. LAPACK band storage
So, what did you do with my panel?

From LAPACK Lower Band Format

into

Upper Square Block Band Format
Upper Square Block Band Format

Final layout P+T

P and T must be stored in compatible formats
- P stored in Upper Square Block Band Format
- T stored in Upper Square Block Packed Format
Data Transformation Process

Fast in-place data transformations based on

- Transposition
- Shuffle/Unshuffle

References

Outline

Introduction

Upper Square Block Band Format

Ongoing work

Conclusions
The new format stores submatrices as a set of square blocks. This

- Provides higher locality of memory accesses
- Results in regular access patterns
- Exposes parallelism

Consequently, this allows for efficient execution of kernels working on square blocks in parallel.

- No further data copies or transformations are necessary at computation time
- Data off-loading is more efficient
- Can use regular BLAS or LAPACK codes, or Specialized kernels
- Can be parallelized more easily
Code optimization and Parallelization

We are currently

- Creating efficient kernels for efficient execution on each core.
  - Previous work on creation of a Small Matrix Library (SML) with auto-tuning via iterative compilation
Code optimization and Parallelization

- Parallelizing the code using OMPSs
  - Dynamic Task Scheduling based on a Task Dependency Graph
Outline

Introduction

Upper Square Block Band Format

Ongoing work

Conclusions
Conclusions

The new format stores submatrices as a set of square blocks. This

- Reduces storage space
- Provides higher locality of memory accesses
- Results in regular access patterns
- Exposes parallelism
- Will yield optimal performance
Square Block Code for Positive Definite Symmetric Cholesky Band Routines

Fred G. Gustavson\textsuperscript{1}, José R. Herrero\textsuperscript{2}, E. Morancho\textsuperscript{2}

\textsuperscript{1}IBM T.J. Watson Research Center, Emeritus, and Umeå University  
fg2935@hotmail.com

\textsuperscript{2}Computer Architecture Department  
Universitat Politècnica de Catalunya, BarcelonaTech  
Barcelona, Spain  
{josepr,enricm}@ac.upc.edu

10th Int. Conf. on Parallel Processing and Applied Mathematics (PPAM’13)

Warsaw, Poland, Sep. 10th, 2013