Cholesky Factorization on Tile Low-Rank Matrices for Distributed-Memory Systems

Kadir Akbudak, Hatem Ltaief, Aleksandr Mikhalev, Ali Charara and David keyes
Extreme Computing Research Center
King Abdullah University of Science and Technology

IXPUG Workshop, KAUST

April 23-25, 2018
How does future of Dense Linear Algebra (DLA) look like?

Currently dense matrices arising in scientific applications. Dense matrices might be compressed.

Tile low rank (TLR) matrix format

Cholesky factorization (for distributed-memory architectures)

Huge performance improvement via cutting down flops

Significantly less memory via storing small $U_{ij}$ and $V_{ij}$ matrices instead of big $A_{ij}$, where $A_{ij} = U_{ij} V_{ij}$

Preserving the accuracy requirements of the scientific application

MKL ScaLAPACK

pdpotrf → pdpotrf?
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
Motivation

How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format

![Diagram of TLR matrix format]

K. Akbudak et al.

2 / 55
Motivation

How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops

\[ D_{1,1} \]
\[ D_{2,2} \]
\[ V_{2,1} \]
\[ V_{2,1} \]
\[ U_{2,1} \]
\[ V_{3,1} \]
\[ V_{3,1} \]
\[ U_{3,1} \]
\[ V_{4,1} \]
\[ V_{4,1} \]
\[ U_{4,1} \]
\[ V_{5,1} \]
\[ V_{5,1} \]
\[ U_{5,1} \]
\[ V_{6,1} \]
\[ V_{6,1} \]
\[ U_{6,1} \]
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops
  - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,j} = U_{i,j}V_{i,j}$
Motivation

How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops
  - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,j} = U_{i,j} V_{i,j}$
  - Preserving the accuracy requirements of the scientific application
Motivation

How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops
  - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,j} = U_{i,j} V_{i,j}$
  - Preserving the accuracy requirements of the scientific application
- MKL ScaLAPACK
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops
  - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,j} = U_{i,j} V_{i,j}$
  - Preserving the accuracy requirements of the scientific application
- MKL ScaLAPACK
  - pdpotrf → pdpotrf_tlr?
Computational statistics: multivariate large spatial data sets in climate/weather modeling:

\[ \ell(\theta) = -\frac{1}{2}Z^T\Sigma^{-1}(\theta)Z - \frac{1}{2}\log|\Sigma(\theta)| \]

(a) Problem Definition.

(b) Soil moisture.

(c) Wind speed.
Earth Science

Seismic imaging: Imaging the subsurface by solving the Helmholtz equation (acoustic wave equation):

\[
(-\triangle - k^2)u(x, w) = s(x, w)
\]

\[k = \frac{w}{v(x)},\] w is the angular frequency, v(x) is the seismic velocity field, and u(x, w) is the time-harmonic wavefield solution to the forcing term s(x, w).
Materials Science

- Structural and vibrational analysis to problems in computational physics and chemistry like electronic and band structure calculations

\[(A - \lambda B)x = 0\]

(a) Problem Definition. (b) Electronic structure.

Figure: Design of new materials.

w/ U. Schwingenschlogl
Common Properties

- Symmetric, positive-definite matrix
Common Properties

- Symmetric, positive-definite matrix
- (Apparently) dense matrices
Common Properties

- Symmetric, positive-definite matrix
- (Apparently) dense matrices
- Often data-sparse
Common Properties

- Symmetric, positive-definite matrix
- (Apparently) dense matrices
- Often data-sparse
  - Decay of parameter correlations with distance
Common Kernel: Cholesky Factorization (potrf)

The Cholesky factorization of an $N \times N$ real symmetric, positive-definite matrix $A$ has the form

$$A = LL^T,$$

where $L$ is an $N \times N$ real lower triangular matrix with positive diagonal elements.
Ranks after Compression into Tile Low Rank (TLR) Format

- TLR format with varying ranks

\[
\begin{align*}
D_{1,1} & \quad \gamma_{n1} \\
V_{2,1} & \quad k_1 \\
U_{3,1} & \\
V_{3,1} & \quad k_1 \\
U_{4,1} & \quad D_{4,4} \\
V_{5,1} & \quad k_1 \\
U_{6,1} & \\
V_{6,1} & \quad k_1 \\
\end{align*}
\]
Ranks after Compression into Tile Low Rank (TLR) Format

- **TLR format with varying ranks**

![Diagram of TLR format with varying ranks]

- **Geospatial statistics, matrix size: 20000, block size: 500, accuracy threshold: $10^{-9}$, 2D problem**

```
<table>
<thead>
<tr>
<th>0</th>
<th>500</th>
<th>133</th>
<th>47</th>
<th>35</th>
<th>154</th>
<th>83</th>
<th>44</th>
<th>33</th>
<th>59</th>
<th>49</th>
<th>38</th>
<th>30</th>
<th>40</th>
<th>37</th>
<th>33</th>
<th>29</th>
<th>32</th>
<th>30</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>173</td>
<td>500</td>
<td>139</td>
<td>48</td>
<td>36</td>
<td>163</td>
<td>86</td>
<td>44</td>
<td>33</td>
<td>59</td>
<td>49</td>
<td>38</td>
<td>30</td>
<td>40</td>
<td>37</td>
<td>33</td>
<td>29</td>
<td>32</td>
<td>30</td>
<td>27</td>
</tr>
<tr>
<td>47</td>
<td>139</td>
<td>500</td>
<td>137</td>
<td>44</td>
<td>153</td>
<td>83</td>
<td>38</td>
<td>49</td>
<td>59</td>
<td>49</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>37</td>
<td>33</td>
<td>32</td>
<td>33</td>
<td>32</td>
<td>30</td>
</tr>
<tr>
<td>35</td>
<td>48</td>
<td>137</td>
<td>500</td>
<td>33</td>
<td>44</td>
<td>164</td>
<td>30</td>
<td>38</td>
<td>49</td>
<td>59</td>
<td>29</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>27</td>
<td>30</td>
<td>32</td>
<td>33</td>
<td>30</td>
</tr>
<tr>
<td>154</td>
<td>86</td>
<td>44</td>
<td>33</td>
<td>150</td>
<td>48</td>
<td>34</td>
<td>165</td>
<td>84</td>
<td>44</td>
<td>33</td>
<td>59</td>
<td>50</td>
<td>38</td>
<td>30</td>
<td>41</td>
<td>37</td>
<td>33</td>
<td>30</td>
<td>27</td>
</tr>
<tr>
<td>44</td>
<td>86</td>
<td>136</td>
<td>83</td>
<td>48</td>
<td>139</td>
<td>500</td>
<td>137</td>
<td>44</td>
<td>86</td>
<td>172</td>
<td>86</td>
<td>38</td>
<td>50</td>
<td>49</td>
<td>33</td>
<td>37</td>
<td>40</td>
<td>37</td>
<td>30</td>
</tr>
<tr>
<td>33</td>
<td>44</td>
<td>83</td>
<td>164</td>
<td>34</td>
<td>48</td>
<td>137</td>
<td>500</td>
<td>33</td>
<td>44</td>
<td>86</td>
<td>166</td>
<td>30</td>
<td>39</td>
<td>50</td>
<td>49</td>
<td>39</td>
<td>37</td>
<td>41</td>
<td>30</td>
</tr>
<tr>
<td>59</td>
<td>50</td>
<td>38</td>
<td>30</td>
<td>165</td>
<td>86</td>
<td>44</td>
<td>33</td>
<td>500</td>
<td>143</td>
<td>35</td>
<td>164</td>
<td>85</td>
<td>44</td>
<td>33</td>
<td>59</td>
<td>49</td>
<td>38</td>
<td>31</td>
<td>27</td>
</tr>
<tr>
<td>49</td>
<td>59</td>
<td>49</td>
<td>38</td>
<td>163</td>
<td>84</td>
<td>44</td>
<td>143</td>
<td>500</td>
<td>143</td>
<td>48</td>
<td>159</td>
<td>87</td>
<td>44</td>
<td>50</td>
<td>49</td>
<td>49</td>
<td>38</td>
<td>40</td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td>50</td>
<td>59</td>
<td>49</td>
<td>44</td>
<td>85</td>
<td>172</td>
<td>86</td>
<td>48</td>
<td>143</td>
<td>500</td>
<td>134</td>
<td>44</td>
<td>86</td>
<td>156</td>
<td>81</td>
<td>38</td>
<td>49</td>
<td>58</td>
<td>49</td>
</tr>
<tr>
<td>30</td>
<td>38</td>
<td>49</td>
<td>59</td>
<td>33</td>
<td>44</td>
<td>166</td>
<td>35</td>
<td>48</td>
<td>134</td>
<td>500</td>
<td>33</td>
<td>45</td>
<td>86</td>
<td>157</td>
<td>30</td>
<td>39</td>
<td>49</td>
<td>38</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>37</td>
<td>33</td>
<td>29</td>
<td>59</td>
<td>49</td>
<td>38</td>
<td>30</td>
<td>164</td>
<td>84</td>
<td>44</td>
<td>33</td>
<td>500</td>
<td>138</td>
<td>48</td>
<td>35</td>
<td>162</td>
<td>86</td>
<td>44</td>
<td>33</td>
</tr>
<tr>
<td>37</td>
<td>41</td>
<td>37</td>
<td>33</td>
<td>50</td>
<td>59</td>
<td>50</td>
<td>39</td>
<td>159</td>
<td>86</td>
<td>45</td>
<td>138</td>
<td>500</td>
<td>142</td>
<td>48</td>
<td>85</td>
<td>165</td>
<td>85</td>
<td>45</td>
<td>33</td>
</tr>
<tr>
<td>33</td>
<td>37</td>
<td>41</td>
<td>37</td>
<td>38</td>
<td>50</td>
<td>59</td>
<td>50</td>
<td>44</td>
<td>87</td>
<td>156</td>
<td>86</td>
<td>48</td>
<td>142</td>
<td>500</td>
<td>133</td>
<td>44</td>
<td>84</td>
<td>159</td>
<td>85</td>
</tr>
<tr>
<td>29</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>30</td>
<td>38</td>
<td>49</td>
<td>59</td>
<td>33</td>
<td>44</td>
<td>81</td>
<td>157</td>
<td>35</td>
<td>48</td>
<td>133</td>
<td>500</td>
<td>33</td>
<td>44</td>
<td>81</td>
<td>157</td>
</tr>
<tr>
<td>33</td>
<td>32</td>
<td>30</td>
<td>27</td>
<td>41</td>
<td>37</td>
<td>33</td>
<td>29</td>
<td>59</td>
<td>49</td>
<td>38</td>
<td>30</td>
<td>162</td>
<td>85</td>
<td>44</td>
<td>33</td>
<td>500</td>
<td>142</td>
<td>47</td>
<td>34</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>32</td>
<td>30</td>
<td>37</td>
<td>40</td>
<td>37</td>
<td>33</td>
<td>49</td>
<td>59</td>
<td>49</td>
<td>39</td>
<td>86</td>
<td>165</td>
<td>84</td>
<td>44</td>
<td>142</td>
<td>500</td>
<td>136</td>
<td>48</td>
</tr>
<tr>
<td>30</td>
<td>32</td>
<td>33</td>
<td>32</td>
<td>33</td>
<td>37</td>
<td>40</td>
<td>38</td>
<td>49</td>
<td>58</td>
<td>49</td>
<td>44</td>
<td>85</td>
<td>159</td>
<td>81</td>
<td>47</td>
<td>136</td>
<td>500</td>
<td>130</td>
<td>27</td>
</tr>
<tr>
<td>27</td>
<td>30</td>
<td>32</td>
<td>33</td>
<td>30</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>31</td>
<td>38</td>
<td>49</td>
<td>59</td>
<td>33</td>
<td>45</td>
<td>85</td>
<td>157</td>
<td>34</td>
<td>48</td>
<td>130</td>
<td>500</td>
</tr>
</tbody>
</table>
```
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
- x axis: different tile sizes
Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$

Heat map

- $x$ axis: different tile sizes
- $y$ axis: different accuracy thresholds
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with \( N = 128 \) and \( k = N \times 0.625 \) (10 grid points / wavelength) on \( \Omega = [0, 1]^3 \)

- Heat map
  - \( x \) axis: different tile sizes
  - \( y \) axis: different accuracy thresholds
  - value/color: maximum rank in TLR matrix for a specific tile size and accuracy
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$

- Heat map
- $x$ axis: different tile sizes
- $y$ axis: different accuracy thresholds
- value/color: maximum rank in TLR matrix for a specific tile size and accuracy
- lighter color: smaller maxrank
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$

- Heat map

- Even for accuracy of $1e^{-14}$, ranks are smaller than half of tile size.
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$

- Heat map
- Even for accuracy of $1e-14$, ranks are smaller than half of tile size.
- Even for tiles of size 2048, rank is smaller than $2048/2 = 1024$. 

![Heat map of maximum ranks after compression into TLR format.](image-url)
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map

Accuracy threshold vs. Tile size heat map

- Maximum observed rank

K. Akbudak et al.
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- x axis: different tile sizes

Data Sparsity

K. Akbudak et al.
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds
- value/color: maximum rank in TLR matrix for a specific tile size and accuracy

K. Akbudak et al.
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- $x$ axis: different tile sizes
- $y$ axis: different accuracy thresholds
- value/color: maximum rank in TLR matrix for a specific tile size and accuracy
- lighter color: smaller maxrank

<table>
<thead>
<tr>
<th>Tile size</th>
<th>Accuracy threshold</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.01</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>10</td>
<td>0.0001</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>15</td>
<td>1e-06</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>20</td>
<td>1e-08</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>25</td>
<td>1e-10</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>30</td>
<td>1e-12</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>35</td>
<td>1e-14</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>40</td>
<td>1e-16</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>45</td>
<td>1e-18</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>50</td>
<td>1e-20</td>
<td>5</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
</tbody>
</table>
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- Even for accuracy of $1e^{-14}$, maximum rank is 55 which is considerably smaller than tile size.
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- **Electronic structure**: Hamiltonian/Overlap matrix
- **Heat map**
- Even for accuracy of $1e^{-14}$, maximum rank is 55 which is considerably smaller than tile size.
- Even for larger tiles, maximum rank is 55.
The HiCMA Library: Hierarchical Computations on Manycore Architectures

Available at http://github.com/ecrc/hicma
Tile Low-Rank Cholesky-based Matrix Approximation

Dense Linear Algebra Renaissance

Fixed ranks
Preconditioners
Performance oriented

Fixed accuracy
Variable ranks
Dense/Sparse Direct Solvers

Figure: Tile Algorithms.
The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

- **hcore_dpotrf**: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.

- **hcore_dtrsm**: The operation applies an update to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the diagonal tile above it and overrides it with the final elements of the output matrix:
  \[ V(i,k) = V(i,k) \times D^{−1}(k,k) \]. The operation is a triangular solve.

- **hcore_dsyrk**: The kernel applies updates to a diagonal (lower triangular) tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it:
  \[ D(j,j) = D(j,j) - (U(j,k) \times V^T(j,k)) \times (U(j,k) \times V^T(j,k))^T \]. The operation is a symmetric rank-\(k\) update.

- **hcore_dgemm**: The operation applies updates to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it. The operation involves two QR factorizations, one reduced SVD (depending on the rank and/or the accuracy parameter) and two matrix-matrix multiplications.
The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

- **hcore_dpotrf**: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.

- **hcore_dtrsm**: The operation applies an update to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the diagonal tile above it and overrides it with the final elements of the output matrix: \( V(i,k) = V(i,k) \times D^{-1}_{(k,k)} \). The operation is a triangular solve.

- **hcore_dsym**: The kernel applies updates to a diagonal (lower triangular) tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it:

\[
D(j,j) = D(j,j) - (U(j,k) \times V^T(j,k)) \times (U(j,k) \times V^T(j,k))^T.
\]

The operation is a symmetric rank-\(k\) update.

- **hcore_dgemm**: The operation applies updates to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it. The operation involves two QR factorizations, one reduced SVD (depending on the rank and/or the accuracy parameter) and two matrix-matrix multiplications.
The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

**hcore dpotrf**: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.

**hcore dtrsm**: The operation applies an update to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the diagonal tile above it and overrides it with the final elements of the output matrix: \( \mathbf{V}(i,k) = \mathbf{V}(i,k) \times \mathbf{D}^{-1}_{(k,k)} \). The operation is a triangular solve.

**hcore dsyrk**: The kernel applies updates to a diagonal (lower triangular) tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it:
\[
\mathbf{D}(j,j) = \mathbf{D}(j,j) - (\mathbf{U}(j,k) \times \mathbf{V}^T(j,k)) \times (\mathbf{U}(j,k) \times \mathbf{V}^T(j,k))^T.
\]
The operation is a symmetric rank-\(k\) update.
HiCMA dpotrf

The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

- **hcore_dpotrf**: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.

- **hcore_dtrsm**: The operation applies an update to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the diagonal tile above it and overrides it with the final elements of the output matrix: $V_{(i,k)} = V_{(i,k)} \times D_{(k,k)}^{-1}$. The operation is a triangular solve.

- **hcore_dsyrk**: The kernel applies updates to a diagonal (lower triangular) tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it: $D_{(j,j)} = D_{(j,j)} - (U_{(j,k)} \times V_{(j,k)}^T) \times (U_{(j,k)} \times V_{(j,k)}^T)^T$. The operation is a symmetric rank-$k$ update.

- **hcore_dgemm**: The operation applies updates to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it. The operation involves two QR factorizations, one reduced SVD (depending on the rank and/or the accuracy parameter) and two matrix-matrix multiplications.
Algorithm 1 hicma_dpotrf (HicmaLower, D, U, V, N, nb, rank, acc)

\[ p = \frac{N}{nb} \]

for \( k = 1 \) to \( p \) do

#task access(read\&write:D(k))

hcore_dpotrf (HicmaLower, D(k))

for \( i = k+1 \) to \( p \) do

#task access(read:D(k), read\&write:V(i,k))

hcore_dtrsm (V(i,k), D(k,k))

end for

for \( j = k+1 \) to \( p \) do

# task access(read:U(j,k), read:V(j,k), read\&write:D(j))

hcore_dsyrk (D(j), U(j,k), V(j,k))

for \( i = j+1 \) to \( p \) do

#task access(read:U(i,k),

read:V(i,k)), read:U(j,k), read:V(j,k),

read\&write:U(i,j), read\&write:V(i,j))

hcore_dgemm (U(i,k), V(i,k), U(j,k), V(j,k), U(i,j), V(i,j), rank, acc)

end for

end for

end for
Dense vs TLR Cholesky on a Single Node

Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
Smaller the better.
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.

- Application: Geospatial statistic, square exp. kernel
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: $10^{-9}$. This accuracy level is enough for a large amount of cases encountered in the geospatial statistics application.
**Experimental Results**

### Dense vs TLR Cholesky on a Single Node

- **Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).**
  - Smaller the better.
- **Application:** Geospatial statistic, square exp. kernel
- **Accuracy threshold:** $10^{-9}$. This accuracy level is enough for a large amount of cases encountered in the geospatial statistics application.
- **StarPU (task-based runtime)**

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>MKL-SNB</th>
<th>MKL-HSW</th>
<th>MKL-SKL</th>
<th>HiCMA-SNB</th>
<th>HiCMA-HSW</th>
<th>HiCMA-SKL</th>
</tr>
</thead>
<tbody>
<tr>
<td>27K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>108K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>176K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>230K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>297K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph showing execution times for different matrix sizes and runtime environments](image)
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.

- Application: Geospatial statistic, square exp. kernel

- Accuracy threshold: $10^{-9}$. This accuracy level is enough for a large amount of cases encountered in the geospatial statistics application.

- StarPU (task-based runtime)

- Three architectures:
  - Sandy Bridge (SNB): AVX, 2.0GHz
    2 sockets x 8 cores
  - Haswell (HSW): AVX2, 2.4GHz
    2 sockets x 18 cores
  - Skylake (SKL): AVX512, 2.1GHz
    2 sockets x 28 cores
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  
  **Smaller the better.**

- x axis (log-scale): different matrix sizes
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL \texttt{dpotrf} (dense) vs HiCMA \texttt{dpotrf} (TLR).
  Smaller the better.
- $x$ axis (log-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices
Dense vs TLR Cholesky on a Single Node

Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.

- x axis (log-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices
- y axis (log-scale): time of dpotrf in seconds
**Experimental Results**

**Dense vs TLR Cholesky on a Single Node**

- Execution times for MKL `dpotrf` (dense) vs HiCMA `dpotrf` (TLR).
  - *Smaller the better.*

- **x axis (log-scale):** different matrix sizes
  - Missing points due to not enough memory for larger matrices

- **y axis (log-scale):** time of `dpotrf` in seconds
  - Smaller time as chips get better
Experimental Results

Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.

- x axis (log-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices

- y axis (log-scale): time of dpotrf in seconds
  - Smaller time as chips get better
  - Solve larger problems
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  - Smaller the better.

- x axis (log-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices

- y axis (log-scale): time of dpotrf in seconds
- Smaller time as chips get better
- Solve larger problems
- Smaller slope: $O(n^3) \rightarrow O(kn^2)$
Experimental Results

ScaLAPACK vs TLR Cholesky \( (Shaheen-2, \text{ HSW, acc}=10^{-9}) \)

- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.

\[ \begin{array}{cccccccccc}
\text{Matrix size} & 10^0 & 10^1 & 10^2 & 10^3 & 10^4 \\
\text{Time(s)} & \text{ScaLAPACK 16 nodes} & \text{HiCMA-TLR Cholesky-16} \\
\end{array} \]

K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.
ScaLAPACK vs TLR Cholesky \((Shaheen-2, HSW, \, acc=10^{-9})\)

- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel

\[\begin{array}{cccccc}
100 & 101 & 102 & 103 & 104 & & & & \\
\end{array}\]

\(10^1\)
\(10^2\)
\(10^3\)
\(10^4\)

Matrix size
Time(s)
ScaLAPACK 16 nodes vs HiCMA-TLR Cholesky-16

\(K.\ \text{Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.}\)
ScaLAPACK vs TLR Cholesky *(Shaheen-2, HSW, acc=10\(^{-9}\))*

- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). *Smaller the better.*
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10\(^{-9}\).

---

## ScaLAPACK vs TLR Cholesky (Shaheen-2, HSW, acc=10⁻⁹)

- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10⁻⁹.
- StarPU (task-based runtime)

![Graph showing execution times for ScaLAPACK 16 nodes vs HiCMA-TLR Cholesky-16](image)

Experimental Results

ScaLAPACK vs TLR Cholesky \((Shaheen-2, HSW, \text{acc}=10^{-9})\)

- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: \(10^{-9}\).
- StarPU (task-based runtime)
- \(x\) axis (log-scale): different matrix sizes

Experimental Results

ScaLAPACK vs TLR Cholesky \((Shaheen-2, HSW, acc=10^{-9})\)

- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: \(10^{-9}\).
- StarPU (task-based runtime)
- \(x\) axis (log-scale): different matrix sizes
- \(y\) axis (log-scale): time of dpotrf in seconds

---

K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.
Experimental Results

ScaLAPACK vs TLR Cholesky \((Shaheen-2, HSW, \text{acc}=10^{-9})\)

- Execution times for ScaLAPACK \text{dpotrf} (dense) vs HiCMA \text{dpotrf} (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: \(10^{-9}\).
- StarPU (task-based runtime)
- \(x\) axis (log-scale): different matrix sizes
- \(y\) axis (log-scale): time of \text{dpotrf} in seconds
- 85x speedup

\[\begin{array}{c|c|c}
\text{Matrix size} & \text{ScaLAPACK 16 nodes} & \text{HiCMA-TLR Cholesky-16} \\
54K & 81K & 108K 135K 189K 270K 351K 459K 594K
\end{array}\]

\(K.\ Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,\) 
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.
ScaLAPACK vs TLR Cholesky \((Shaheen-2, HSW, \text{acc}=10^{-9})\)

- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: \(10^{-9}\).
- StarPU (task-based runtime)
- \(x\) axis (log-scale): different matrix sizes
- \(y\) axis (log-scale): time of dpotrf in seconds

---

K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.
Experimental Results

TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices.
  - Smaller the better.

![Graph showing memory footprint for dense vs TLR matrices.](image)

- Full rank: $N^2$
- TLR: $N^2 k_{nb}$

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>54K</td>
<td>10^0</td>
</tr>
<tr>
<td>81K</td>
<td>10^1</td>
</tr>
<tr>
<td>108K</td>
<td>10^2</td>
</tr>
<tr>
<td>135K</td>
<td>10^3</td>
</tr>
<tr>
<td>189K</td>
<td></td>
</tr>
<tr>
<td>270K</td>
<td></td>
</tr>
<tr>
<td>351K</td>
<td></td>
</tr>
<tr>
<td>459K</td>
<td></td>
</tr>
<tr>
<td>594K</td>
<td></td>
</tr>
</tbody>
</table>

K. Akbudak et al. 37 / 55
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel

![Graph showing memory footprint comparison between Full rank and TLR matrices for different matrix sizes. The x-axis represents matrix size, and the y-axis represents memory required in gigabytes. The graph shows a clear upward trend for both Full rank and TLR, with Full rank having a higher memory requirement at larger matrix sizes.]
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
TLR Cholesky: Memory Footprint \((\text{acc}=10^{-9})\)

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: \(10^{-9}\)
- x axis \((\log\text{-scale})\): different matrix sizes \((N)\)

![Graph showing memory footprint comparison between Full rank and TLR Cholesky]

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>Full rank (GB)</th>
<th>TLR (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>54K</td>
<td>10^0</td>
<td>10^0</td>
</tr>
<tr>
<td>81K</td>
<td>10^1</td>
<td>10^1</td>
</tr>
<tr>
<td>108K</td>
<td>10^2</td>
<td>10^2</td>
</tr>
<tr>
<td>135K</td>
<td>10^3</td>
<td>10^3</td>
</tr>
<tr>
<td>189K</td>
<td>10^3</td>
<td>10^3</td>
</tr>
<tr>
<td>270K</td>
<td>10^4</td>
<td>10^4</td>
</tr>
<tr>
<td>351K</td>
<td>10^4</td>
<td>10^4</td>
</tr>
<tr>
<td>459K</td>
<td>10^5</td>
<td>10^5</td>
</tr>
<tr>
<td>594K</td>
<td>10^6</td>
<td>10^6</td>
</tr>
</tbody>
</table>
**TLR Cholesky: Memory Footprint (acc=10^{-9})**

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (log-scale): different matrix sizes ($N$)
- y axis (log-scale): memory required for storing matrix in terms of giga bytes

![Graph showing memory footprint comparison between Full rank and TLR methods]
Experimental Results

TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (log-scale): different matrix sizes (N)
- y axis (log-scale): memory required for storing matrix in terms of giga bytes
- Full rank: N^2

![Graph showing memory footprint comparison between Full rank and TLR]

\[
\begin{array}{cccccc}
\text{Matrix size} & 54K & 81K & 108K & 135K & 189K & 270K & 351K & 459K & 594K \\
\hline
\text{Memory (GB)} & 10^0 & 10^1 & 10^2 & 10^3 & & & & & \\
\end{array}
\]
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (log-scale): different matrix sizes (N)
- y axis (log-scale): memory required for storing matrix in terms of giga bytes
- Full rank: $N^2$
- TLR: $\frac{N^2 k}{nb}$
TLR Cholesky: Memory Footprint (acc=$10^{-9}$)

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: $10^{-9}$
- $x$ axis (log-scale): different matrix sizes ($N$)
- $y$ axis (log-scale): memory required for storing matrix in terms of giga bytes
- Full rank: $N^2$
- TLR: $\frac{N^2k}{nb}$
  - $k$: average rank
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (log-scale): different matrix sizes (N)
- y axis (log-scale): memory required for storing matrix in terms of giga bytes
- Full rank: $N^2$
- TLR: $\frac{N^2 k}{nb}$
  - $k$: average rank
  - $nb$: tile size
Impact of Accuracy Thresholds, 1M Matrix Size, nb=2700

- Memory footprint for dense vs TLR matrices. Smaller the better.

![Graph showing memory footprint vs accuracy threshold for full rank and HiCMA-TLR Cholesky methods.](image)

Impact of Accuracy Thresholds, 1M Matrix Size, nb=2700

- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel

![Graph showing memory footprint vs accuracy threshold for different methods.](image)

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- $x$ axis (log-scale): different accuracy thresholds

![Graph showing memory footprint vs accuracy threshold]

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different accuracy thresholds
- y axis (log-scale): memory required for storing matrix in terms of giga bytes

![Graph showing memory footprint with accuracy thresholds on the x-axis and memory required on the y-axis.](image)

Impact of Accuracy Thresholds, 1M Matrix Size, nb=2700

- Memory footprint for dense vs TLR matrices. **Smaller the better.**
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different accuracy thresholds
- y axis (log-scale): memory required for storing matrix in terms of giga bytes
- Being more accurate requires higher ranks so larger memory is used.

Impact of Accuracy Thresholds, **1M Matrix Size, nb=2700, 64 nodes, Shaheen-2**

- Execution times for HiCMA dpotrf (TLR)

---

Experimental Results

Impact of Accuracy Thresholds, 1M Matrix Size, nb=2700, 64 nodes, Shaheen-2

- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different accuracy thresholds

---

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different accuracy thresholds
- y axis (log-scale): time of dpotrf in seconds

![Graph showing execution times for HiCMA dpotrf (TLR) with accuracy thresholds on the x-axis and time in seconds on the y-axis.](image)

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different accuracy thresholds
- y axis (*log*-scale): time of dpotrf in seconds
- Being more accurate requires higher ranks so more flops are done.

---

Experimental Results

TLR Cholesky up to 11M (Shaheen-2, HSW, Statistics - SqExp kernel, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes

![Graph showing execution times for HiCMA dpotrf (TLR) for different number of nodes](image)

Experimental Results

TLR Cholesky up to 11M (*Shaheen-2, HSW, Statistics - SqExp kernel, acc=10^{-9}*)

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel

![Graph showing execution times for HiCMA dpotrf (TLR) for different number of nodes and matrix sizes.](image)

Experimental Results

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10\(^{-9}\*))

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes

TLR Cholesky up to 11M (*Shaheen-2, HSW, Statistics - SqExp kernel, acc=10^{-9}*)

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of dpotrf in minutes

---

*K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.*
Experimental Results

TLR Cholesky up to 11M (Shaheen-2, HSW, Statistics - SqExp kernel, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes
- y axis (log-scale): time of dpotrf in minutes
- Running ScaLAPACK for larger cases is not feasible because of time and memory.

Experimental Results

TLR Cholesky up to 11M (*Shaheen-2, HSW, Statistics - SqExp kernel, acc=10^{-9}*)

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes
- y axis (log-scale): time of dpotrf in minutes
- Running ScaLAPACK for larger cases is not feasible because of time and memory.
- Missing points for larger matrices: not enough memory

*K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.*
Experimental Results

TLR Cholesky up to 11M (*Shaheen-2, HSW, Statistics - SqExp kernel, acc=10^{-9}*)

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes
- y axis (log-scale): time of dpotrf in minutes
- Running ScALAPACK for larger cases is not feasible because of time and memory.
- Missing points for larger matrices: not enough memory
- Missing points for smaller matrices: not enough work

---

Experimental Results

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes

![Graph showing execution times for HiCMA dpotrf (TLR) for different number of nodes.]

Experimental Results

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel

![Graph showing execution times for HiCMA dpotrf (TLR) for different number of nodes.]

Experimental Results

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes

![Graph showing execution times for HiCMA dpotrf (TLR) for different number of nodes and matrix sizes.](image_url)

Experimental Results

**TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})**

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- $x$ axis (log-scale): different matrix sizes
- $y$ axis (log-scale): time of dpotrf in minutes

---

Experimental Results

**TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})**

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes
- y axis (log-scale): time of dpotrf in minutes
- We ran HiCMA on the latest Intel deployment.

---

![Graph showing execution times for HiCMA dpotrf (TLR) for different number of nodes.](image)

*K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.*
Experimental Results

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes

\[ \begin{array}{cccccc}
1M & 2M & 4M & 5M & 6M & 8M & 11M \\
Matrix size & 3 & 6 & 7 & 8 & 13 & 17 & 20 & 30 & 42 & 50 & 133
\end{array} \]

Time (minutes)

- HiCMA-16
- HiCMA-32
- HiCMA-64
- HiCMA-128
- HiCMA-256

Experimental Results

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel

Experimental Results

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes

Experimental Results

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- $x$ axis (log-scale): different matrix sizes
- $y$ axis (log-scale): time of dpotrf in minutes

Experimental Results

**TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})**

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (log-scale): different matrix sizes
- y axis (log-scale): time of dpotrf in minutes
- Not big difference between turbo on and off due low arithmetic intensity

![Graph showing execution times for HiCMA dpotrf (TLR) for different number of nodes.](image)

Experimental Results

Traces Chameleon: Dense dpotrf time = 18.1s on 4 nodes of Shaheen-2 with a matrix size of 54K
Experimental Results

Traces HiCMA: Data-sparse dpotrf time=1.8s on 4 nodes of Shaheen-2 with a matrix size of 54K
We cut down flops through compression.
We use less memory.
Accuracy requirement is satisfied.
HiCMA is for both shared and distributed memory systems.

HODLR/H (non-nested bases) data compression format.
More matrix computation algorithms (LU/QR, eigenvalue solvers/SVD, matrix inversion, etc.)
Support for more large-scale scientific applications.
The Hourglass Revisited
The Hourglass Revisited
Students/Collaborators/Vendors

- Tokyo Institute of Technology, Japan: R. Yokota
- Innovative Computing Laboratory @ UTK, USA: PLASMA/MAGMA/PaRSEC Teams
- INRIA/INP/LaBRI Bordeaux, France: Runtime/HiePACS Teams
- Max-Planck Institute@Leipzig, Germany: R. Kriemann
- American University of Beirut, Lebanon: G. Turkiyyah
- KAUST Supercomputing Lab
- Intel Parallel Computing Center
- Cray Center of Excellence
Thank You!

Questions?