Parallel Formulation of a Matrix Factorization Algorithm

Lars Karlsson, Björn Adlerborn, Bo Kågström

Research Day
2015-02-25
The 3 obligatory questions

1. Why is our research important?
2. What can it be used for?
3. What is interesting from a research perspective?
Aim

- Give you some idea of what we do
- Give an example of a problem we have recently studied
The HT matrix factorization

\[(A, B) \mapsto (H, T) = (Q^T AZ, Q^T BZ)\]

- **Input:** General matrix pair
- **Output:** Hessenberg-triangular matrix pair
- **Purpose:** Get structured form that is easier to work with
Rotation

\[ x \mapsto y = Gx, \quad G = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, \quad c^2 + s^2 = 1. \]

- Rotation is a little $2 \times 2$ matrix
Rotation on matrix rows

Matrix

Rotation

Rows

Affected rows

\[
G = \begin{bmatrix}
I & c & -s \\
\frac{c}{I} & s & c \\
\frac{s}{I} & s & c
\end{bmatrix}, \quad A \mapsto GA
\]

- Rotation affects a pair of adjacent rows
Sequence of overlapping rotations

\[ A \mapsto G_{n-1} G_{n-2} \cdots G_1 A \]

- Sequence of rotations affecting all rows
- From the bottom up
- Subsequent rotations overlap with one row
Distributed matrix

- Blocks of rows distributed like a deck of cards
Goal

- Apply a sequence of rotations...
- ...to a distributed matrix...
- ...using parallel processing...
- ...as fast as possible.
Local vs border rotations

- Most rotations are **local**: Involve one processor
- Some rotations cross a **border**: Involve two processors
Problems

- Each local rotation (many) involves only one processor
- Each border rotation (few) involves only two processors
- Overlapping rotations: dependence from one rotation to the next
Split a rotation into smaller parts

1. Read
2. Compute
3. Write

- Matrix
  - Rotation
  - Columns updated independently
  - Dependencies within but not between columns
Idea!

- Split columns into fragments
- Apply sequence of rotations to each fragment separately
A closer look at one fragment

- $L = \text{Local rotation}$
- $B = \text{Border rotation}$
- Rotations applied in sequence of the form
  
  \[
  \cdots LL \cdots LLBLL \cdots LLBLL \cdots LL \cdots
  \]

- After merging adjacent $L$’s:
  
  \[
  \cdots LBLBLL \cdots
  \]

- State halfway through:
  
  \[
  LBLBLL \underline{LBLBLL} LBLBLL
  \]

- Clearly defined next action (underlined) or completed (striked)
- Similarly for every fragment independently
Scheduling one fragment

- One action at a time on one or two processors
- Next action one step in the counter-clockwise direction
Tracking the location of all fragments

- A fragment can be located either at or between processors
- Zero or more fragments in each slot
Scheduling idea

We can do one of the following things in parallel:
- One local \((L)\) action per processor, or
- One border \((B)\) action per adjacent pair

The slots are a great tool to:
- Decide which option to take, and
- Keep track of the dynamic scheduling state
Questions to consider:

- Which option to take (L or B actions)?
- Which fragment to choose from each slot?
Scheduling algorithm

Create fragments;
Place each fragment in its initial slot;
while there are remaining fragments do
  Select which type of action to perform ($L$ or $B$);
  Select all slots with fragments of the chosen type;
  Select one fragment from each selected slot;
  Perform an action on all selected fragments in parallel;
  Move (or remove) each selected fragment;
Then what?

Adaptations:
- From block row to 2D block-cyclic distribution
- From rows to columns
- From general to triangular matrix

Extensions:
- Rotations not known in advance (reductions)

Applications:
- Complete HT reduction algorithm
- Other related reduction algorithms
Are there any questions?
Are there any questions?
Are there any questions?

Raise your hand if you could understand!