NUMA-Aware Blocked Hessenberg Reduction Using Parallel Cache Assignment

Mahmoud Eljammaly, Lars Karlsson and Bo Kågström

Department of Computing Science

Overview

Hessenberg reduction is a similarity transformation:

$$Q^T * A * Q = H$$

where Q is an orthogonal matrix stored in compact WY representation.

Challenge

- The computation is ***memory-bound***, which means the performance is limited by the memory bandwidth rather than the computational power.

Solution requirements

- NUMA-aware algorithm.
- High utilization of low-level cache memory.

Blocked Hessenberg reduction: partition the matrix into column blocks and reduce the blocks one by one from left to right.

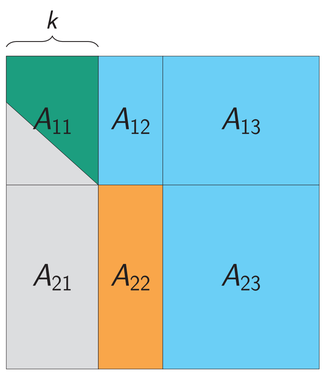


Figure : Partitioning of A after reducing k columns.

Blocked Hessenberg steps

begin
  foreach *block* **do**
    1. Reduce block (Level 2 BLAS) ***CRITICAL***.
       foreach *column in the block* **do**
         1.1 Update column.
         1.2 Construct reflector.
         1.3 Apply reflector / reduce column.
         1.4 Update compact WY representation.
       end
    2. Update compact WY representation (Level 3 BLAS).
    3. Update rest of the matrix (Level 3 BLAS).
  end
end

Level 2 BLAS: Matrix-Vector operations $O(n^2)$ computations and memory access.
Level 3 BLAS: Matrix-Matrix operations $O(n^3)$ computations and $O(n^2)$ memory access.

Parallel Cache Assignment (PCA)

Partition and store the matrix in local caches

- Data reuse $\rightarrow$ reduce cache - memory comm.
- Data locality $\rightarrow$ reduce cache - cache comm.
- NUMA aware $\rightarrow$ reduce remote memory comm.

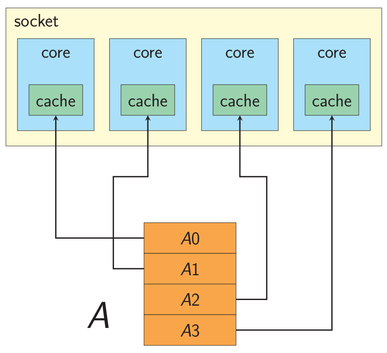


Figure : Data partitioning for PCA

Partitioning scheme

- Row block for inner loop (step 1).
- Column block for outer loop (steps 2 and 3).

Partitioning chosen to reduce synchronization points and to increase data locality and data reuse.

PCA used inside inner loop.

- Copy the submatrix to local buffer.
- Static assignment of data by dividing the submatrix into one block row per core.

Parallel cache assignment fulfills both solution requirements.

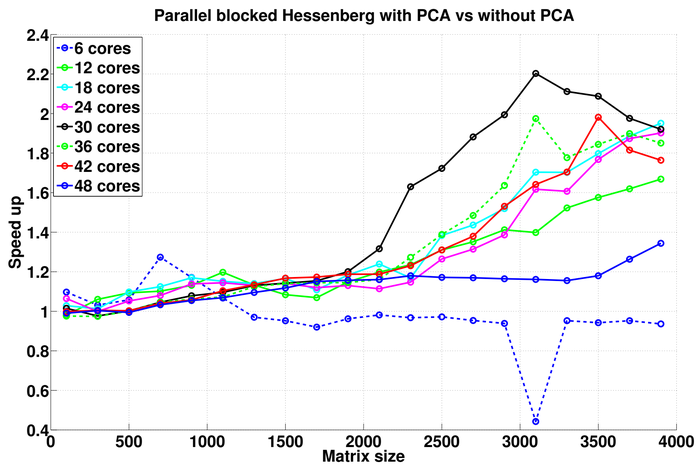
Experiments

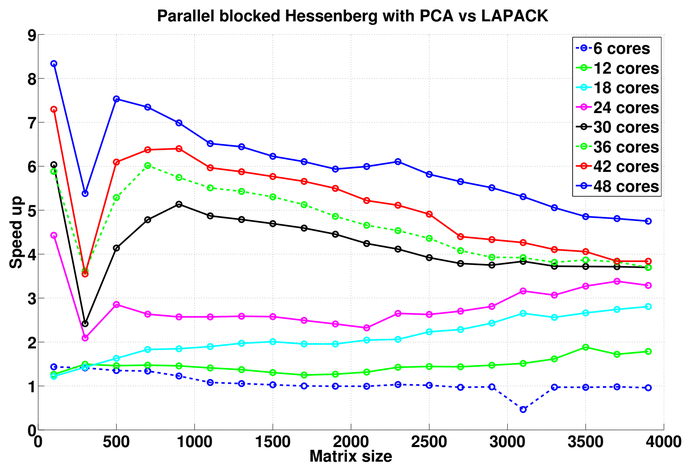
- Matrix sizes N $\times$ N: N between 100 and 4000.
- Number of cores between 6 and 48.
- Test against LAPACK.
- Test against ScaLAPACK.

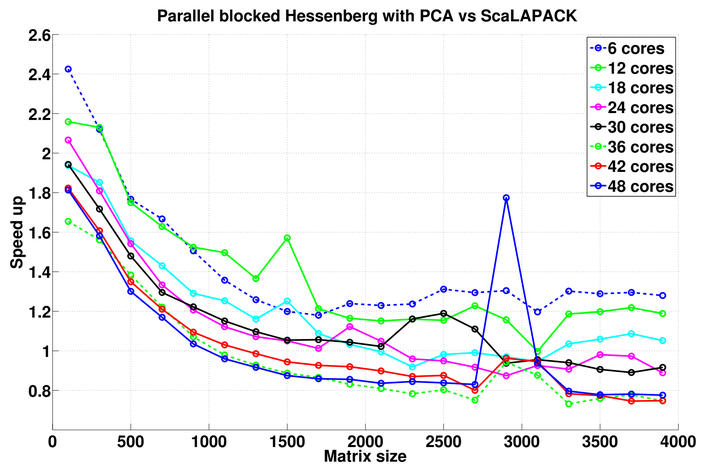
Key results

- **Up to 2.2 times faster than w/o PCA**
- **Up to 8.4 times faster than LAPACK**
- **Up to 2.4 times faster than ScaLAPACK**

Results





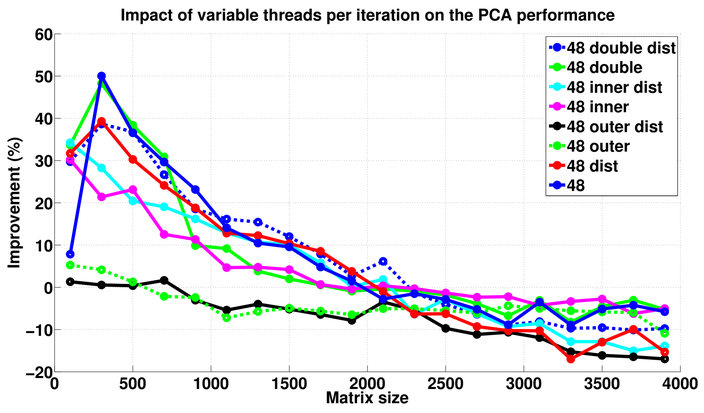


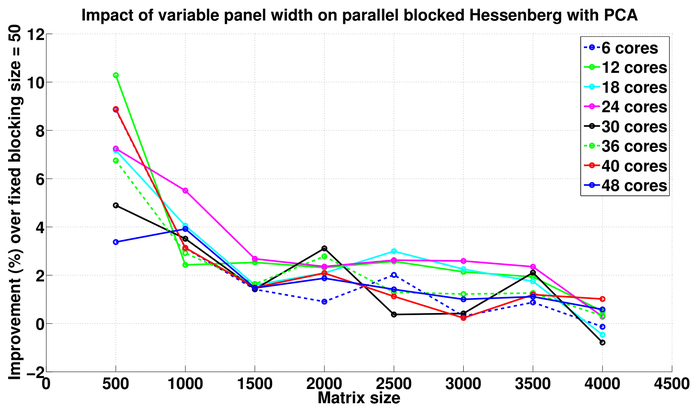
Future work

Tuning potential

- Number of threads per iteration.
  - Number of threads in inner loop.
  - Number of threads in outer loop.
  - Thread distribution.
  - **Up to 50% improvement for some cases.**
- Panel width per iteration.
  - Different panel widths give different performance.

Goal is to design an auto-tuning mechanism to find the best value for these parameters at run-time.





External funding

Vetenskapsrådet

eSSENCE THE E-SCIENCE COLLABORATION

Software developed and tested using the resources of HPC2N and UMIT Research Lab