Block Lanczos-Montgomery Method over Large Prime Fields with GPU Accelerated Dense Operations

D. Zheltkov, N. Zamarashkin

INM RAS

September 24, 2018

Scalability of Lanczos method

Notations

- Matrix order N, average nonzero elements per row ρ.
- Number W of machine words in prime.
- Block size k.
- Number of nodes $p = ks$.

Complexity of block Lanczos method operations

- Sparse matrix by block multiplication — $O\left(\frac{\rho W N^2}{ks}\right)$.
- Dense algebra — $O\left(\frac{W^2 N^2}{ks} + \frac{W^2 k N}{s}\right)$.
- Communication — $O\left(\frac{W N^2}{k} + \frac{W N^2}{ks} + WNk\right)$.

Scalability

If dense algebra is fast method scales **almost perfectly**.

Scalability of Lanczos method

Notations

- Matrix order N, average nonzero elements per row ρ.
- Number W of machine words in prime.
- Block size k.
- Number of nodes $p = ks$.

Complexity of block Lanczos method operations

- Sparse matrix by block multiplication — $O\left(\frac{\rho W N^2}{ks}\right)$.
- Dense algebra — $O\left(\frac{W^2 N^2}{ks} + \frac{W^2 k N}{s}\right)$.
- Communication — $O\left(\frac{W N^2}{k} + \frac{W N^2}{ks} + WNk\right)$.

Scalability

If dense algebra is fast method scales **almost perfectly**.

Scalability of Lanczos method

Notations

- Matrix order N, average nonzero elements per row ρ.
- Number W of machine words in prime.
- Block size k.
- Number of nodes $p = ks$.

Complexity of block Lanczos method operations

- Sparse matrix by block multiplication — $O\left(\frac{\rho W N^2}{ks}\right)$.
- Dense algebra — $O\left(\frac{W^2 N^2}{ks} + \frac{W^2 k N}{s}\right)$.
- Communication — $O\left(\frac{W N^2}{k} + \frac{W N^2}{ks} + W N k\right)$.

Scalability

If dense algebra is fast method scales **almost perfectly**.

Dense algebra over large prime field

CPU implementation problems

Problems

- Relatively high latency and low throughput for integer multiplication.
- Low throughput for extended precision operations.
- No instructions for integer fused multiply-add with carry.
- No extended precision vector instructions.

Result

Rate of needed integer operations can be 64 **times lower** than flop rate or even worse.

Dense algebra over large prime field

CPU implementation problems

Problems

- Relatively high latency and low throughput for integer multiplication.
- Low throughput for extended precision operations.
- No instructions for integer fused multiply-add with carry.
- No extended precision vector instructions.

Result

Rate of needed integer operations can be 64 **times lower** than flop rate or even worse.

Dense algebra over large prime field

GPU advantages and problems

Advantages

- Higher flop rate.
- Instruction for integer fused multiply-add with carry (*madc*).

Problems

- Integer extended precision instructions are 32-bit — 4 times more operations are needed.
- Several (2 to 6) clocks to perform extended precision multiplication.
- Memory resources are more limited.

Result

Overall, GPU must be **several times faster** than CPU even with the same flop rate.

Dense algebra over large prime field

GPU advantages and problems

Advantages

- Higher flop rate.
- Instruction for integer fused multiply-add with carry (*madc*).

Problems

- Integer extended precision instructions are 32-bit — 4 times more operations are needed.
- Several (2 to 6) clocks to perform extended precision multiplication.
- Memory resources are more limited.

Result

Overall, GPU must be **several times faster** than CPU even with the same flop rate.

Dense algebra over large prime field

GPU advantages and problems

Advantages

- Higher flop rate.
- Instruction for integer fused multiply-add with carry (*madc*).

Problems

- Integer extended precision instructions are 32-bit — 4 times more operations are needed.
- Several (2 to 6) clocks to perform extended precision multiplication.
- Memory resources are more limited.

Result

Overall, GPU must be **several times faster** than CPU even with the same flop rate.

Numerical experiments

Implemented method

Implementation details

- Block Lanczos method.
- Naive matrix multiplication on GPU.
- Winograd matrix multiplication on CPU.
- Supported multiple GPU per node.
- Multiplication of dense blocks of sparse matrix on GPU.

GPU cluster of INM RAS (**T**)

- 4-core Intel Core i7-960 3.2 GHz per node.
- Two NVidia Tesla C2070 per node.
- Infiniband 10 Gbit/s.

Supercomputer "Lomonosov" (**L**)

- Two 4-core Intel Xeon X5570 2.93 GHz per node.
- NVidia Tesla X2070 per node.
- Infiniband 40 Gbit/s.

Supercomputer "Lomonosov-2" (**L2**)

- 14-core Intel Xeon E5-2697v3 2.6 GHz per node.
- NVidia Tesla K40M per node.
- Infiniband 56 Gbit/s.

GPU cluster of INM RAS (**T**)

- 4-core Intel Core i7-960 3.2 GHz per node.
- Two NVidia Tesla C2070 per node.
- Infiniband 10 Gbit/s.

Supercomputer "Lomonosov" (**L**)

- Two 4-core Intel Xeon X5570 2.93 GHz per node.
- NVidia Tesla X2070 per node.
- Infiniband 40 Gbit/s.

Supercomputer "Lomonosov-2" (**L2**)

- 14-core Intel Xeon E5-2697v3 2.6 GHz per node.
- NVidia Tesla K40M per node.
- Infiniband 56 Gbit/s.

GPU cluster of INM RAS (**T**)

- 4-core Intel Core i7-960 3.2 GHz per node.
- Two NVidia Tesla C2070 per node.
- Infiniband 10 Gbit/s.

Supercomputer "Lomonosov" (**L**)

- Two 4-core Intel Xeon X5570 2.93 GHz per node.
- NVidia Tesla X2070 per node.
- Infiniband 40 Gbit/s.

Supercomputer "Lomonosov-2" (**L2**)

- 14-core Intel Xeon E5-2697v3 2.6 GHz per node.
- NVidia Tesla K40M per node.
- Infiniband 56 Gbit/s.

Matrix 1 (**M1**)

- 64445×65541.
- $\rho = 24.65$.
- 5 dense blocks.

Matrix 2 (**M2**)

- 2097152×2085659;.
- $\rho = 86.84$.
- 5 dense blocks.

Numerical experiments

Matrices

Matrix 1 (**M1**)

- 64445×65541.
- $\rho = 24.65$.
- 5 dense blocks.

Matrix 2 (**M2**)

- 2097152×2085659;.
- $\rho = 86.84$.
- 5 dense blocks.

Time to compute one Krylov vector, ms for **M1**, s for **M2**

p	**P0, M1**	**P1, M1**	**P2, M1**	**P0, M2**	**P1, M2**	**P2, M2**
1	87.9 (1)	52.2 (1)	49.8 (1)	5.91 (1)	4.7 (1)	4.65 (1)
2	50.5 (1)	29.9 (2)	28.2 (2)	3.19 (1)	2.48 (2)	2.43 (2)
4	31.1 (1)	16.8 (4)	15.3 (4)	1.79 (2)	1.28 (4)	1.25 (4)

Acceleration compared to one node.

p	**P0, M1**	**P1, M1**	**P2, M1**	**P0, M2**	**P1, M2**	**P2, M2**
2	1.74	1.75	1.77	1.85	1.90	1.91
4	2.82	3.11	3.25	3.30	3.67	3.72

Numerical experiments

Results, cluster **T**

Time to compute one Krylov vector, ms for **M1**, s for **M2**

p	**P0, M1**	**P1, M1**	**P2, M1**	**P0, M2**	**P1, M2**	**P2, M2**
1	87.9 (1)	52.2 (1)	49.8 (1)	5.91 (1)	4.7 (1)	4.65 (1)
2	50.5 (1)	29.9 (2)	28.2 (2)	3.19 (1)	2.48 (2)	2.43 (2)
4	31.1 (1)	16.8 (4)	15.3 (4)	1.79 (2)	1.28 (4)	1.25 (4)

Acceleration compared to one node.

p	**P0, M1**	**P1, M1**	**P2, M1**	**P0, M2**	**P1, M2**	**P2, M2**
2	1.74	1.75	1.77	1.85	1.90	1.91
4	2.82	3.11	3.25	3.30	3.67	3.72

Time to compute one Krylov vector, ms for **M1**, s for **M2**

p	**P0, M1**	**P1, M1**	**P0, M2**	**P1, M2**
1	56.9 (1)	41.1 (1)	3.81 (1)	3.10 (1)
2	29.2 (1)	21.5 (2)	2.07 (1)	1.61 (2)
4	21.8 (1)	11.2 (4)	1.14 (1)	0.815 (4)
8	13.8 (2)	6.9 (8)	0.709 (1)	0.417 (8)
16	8.0 (4)	4.3 (16)	0.401 (4)	0.231 (16)
32	–	–	0.216 (8)	0.128 (32)

Acceleration compared to one node.

p	**P0, M1**	**P1, M1**	**P0, M2**	**P1, M2**
1	1	1	1	1
2	1.95	1.91	1.84	1.93
4	2.61	3.67	3.34	3.80
8	4.12	5.96	5.37	7.43
16	7.11	9.56	9.5	13.42
32	–	–	17.64	24.22

Time to compute one Krylov vector, ms for **M1**, s for **M2**

p	**P0, M1**	**P1, M1**	**P0, M2**	**P1, M2**
1	38.3 (1)	29.4 (1)	1.66 (1)	1.41 (1)
2	26.1 (1)	19.2 (2)	0.87 (1)	0.725 (2)
4	15.8 (1)	9.4 (4)	0.500 (1)	0.381 (4)
8	9.9 (1)	5.8 (8)	0.314 (2)	0.202 (8)
16	7.3 (1)	4.0 (16)	0.193 (4)	0.119 (16)
32	6.5 (2)	2.8 (16)	0.116 (8)	0.0698 (32)

Numerical experiments

Results, cluster **L2**

Acceleration compared to one node.

p	**P0, M1**	**P1, M1**	**P0, M2**	**P1, M2**
1	1	1	1	1
2	1.47	1.53	1.91	1.95
4	2.42	3.13	3.32	3.70
8	3.87	5.07	5.29	6.98
16	5.25	7.35	8.6	11.85
32	5.89	10.5	14.31	20.2

Questions?

THANK YOU!