Lomonosov Moscow State University
Institute of Mechanics

On the Role of Non-blocking Global Communications in Implementation of Linear Algebra Methods for HPC Systems

B. Krasnopolsky, A. Medvedev

krasnopolsky@imec.msu.ru

Supercomputing Technologies in Science, Education and Industry
MSU, Moscow, Russia, 25 February 2020

Outline of the Talk

Plan:

1. Motivation
2. Execution time model for Krylov subspace methods
3. IMB-ASYNC benchmarks
4. Theoretical model validation
5. Comparison of BiCGStab methods

Outline of the Talk

Plan:

1. **Motivation**
2. Execution time model for Krylov subspace methods
3. IMB-ASYNC benchmarks
4. Theoretical model validation
5. Comparison of BiCGStab methods

XAMG Library

Current project: development of library for solving SLAEs with multiple right-hand sides

- Focus on elliptic PDEs
- Large sparse systems with ~10^8 unknowns and ~1-32 RHS vectors
- Good scalability at least up to 10^4 CPU cores
- Hybrid programming models (MPI+Posix ShM+CUDA*)
- Vectorization of all basic operations

*B. Krasnopolsky, A. Medvedev, Acceleration of Large Scale OpenFOAM Simulations on Distributed Systems with Multicore CPUs and GPUs // ParCo-2015.

Numerical Methods

Methods, implemented in **XAMG**:

- Classical Algebraic MultiGrid (**hypre** - to construct the hierarchy)
- Chebyshev polynomials, Jacobi, Gauss-Seidel, etc.
- Krylov subspace iterative methods (BiCGStab)
  - Classical BiCGStab?..
  - Pipelined BiCGStab?..
  - Reordered BiCGStab?..
  - ...

Classical BiCGStab

x_0 = initial guess; $p_0 = r_0 = b - Ax_0$
$\rho_0 = (r_0, r_0)$
for $j = 0, 1, \ldots$ **do**
  $v_j = Ap_j$
  $\alpha_j = \frac{\rho_j}{(v_j, r_0)}$
  $s_j = r_j - \alpha_j v_j$
  $t_j = As_j$
  $\omega_j = \frac{(t_j, s_j)}{(t_j, t_j)}$
  $x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j$
  $r_{j+1} = s_j - \omega_j t_j$
  if $\|r_{j+1}\| < \varepsilon$ **then exit**
  $\rho_{j+1} = (r_{j+1}, r_0)$
  $\beta_j = \frac{\rho_{j+1}}{\rho_j} \frac{\alpha_j}{\omega_j}$
  $p_{j+1} = r_{j+1} + \beta_j (p_j - \omega_j v_j)$
end for

Basic operations:

- Matrix-vector multiplications
- Linear operations with vectors
- Dot products
  - 3 global synchronizations per each iteration

BiCGStab Methods

Classical BiCGStab*

$$c = Ab$$
$$\alpha = (c, d)$$
$$e = f + \alpha g$$

- 3 blocking global reductions
- $20N$ FLOPS ($22N$ reads/writes)

*H. A. van der Vorst, SIAM J. Sci. Stat. Comput., 13 (1992), 631–644.

Pipelined BiCGStab**

$$\alpha = (b, c)$$
$$h = Ad$$
$$e = f + \alpha g$$

- 2 non-blocking global reductions
- $38N$ FLOPS ($43N$ reads/writes)

**S. Cools, W. Vanroose, Parallel Comput., 65 (2017) 1–20.

Pipelined BiCGStab with Non-blocking Collectives

Pipelined BiCGStab

Non-blocking collectives from MPI-3:

$$\alpha = (b, c)$$
$$h = Ad$$
$$e = f + \alpha g$$

MPI_Iallreduce(..., &req);

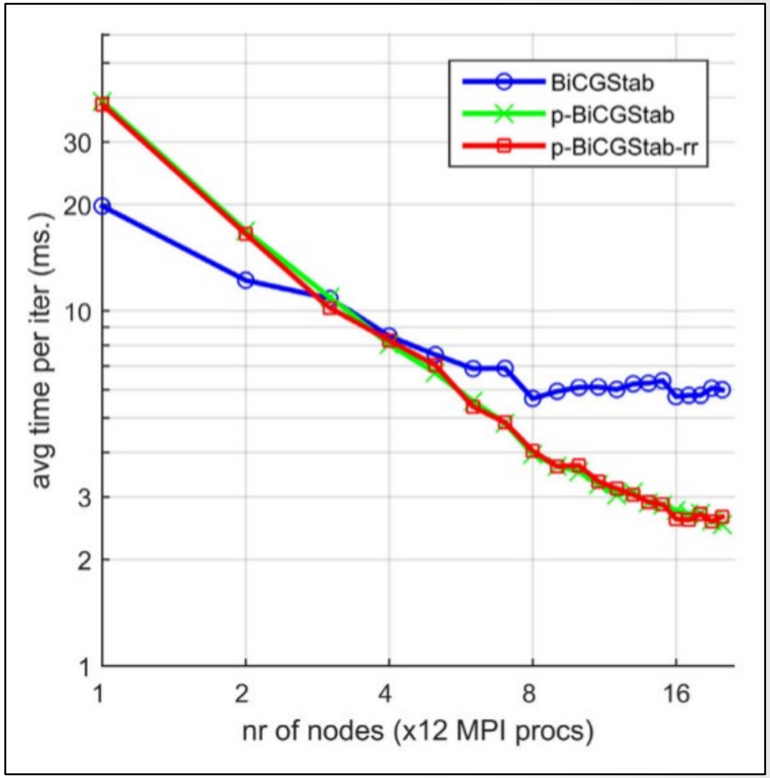
MPI_Wait(&req, ...);

Numerical Experiments*

Test problem: 5-diagonal matrix, with 10^6 unknowns

HPC system: 20 nodes, 2x6-core Intel Xeon X5660, IB QDR

MPI: MPICH-3.1.3
MPICH_ASYNC_PROGRESS = 1
MPICH_MAX_THREAD_SAFETY = multiple



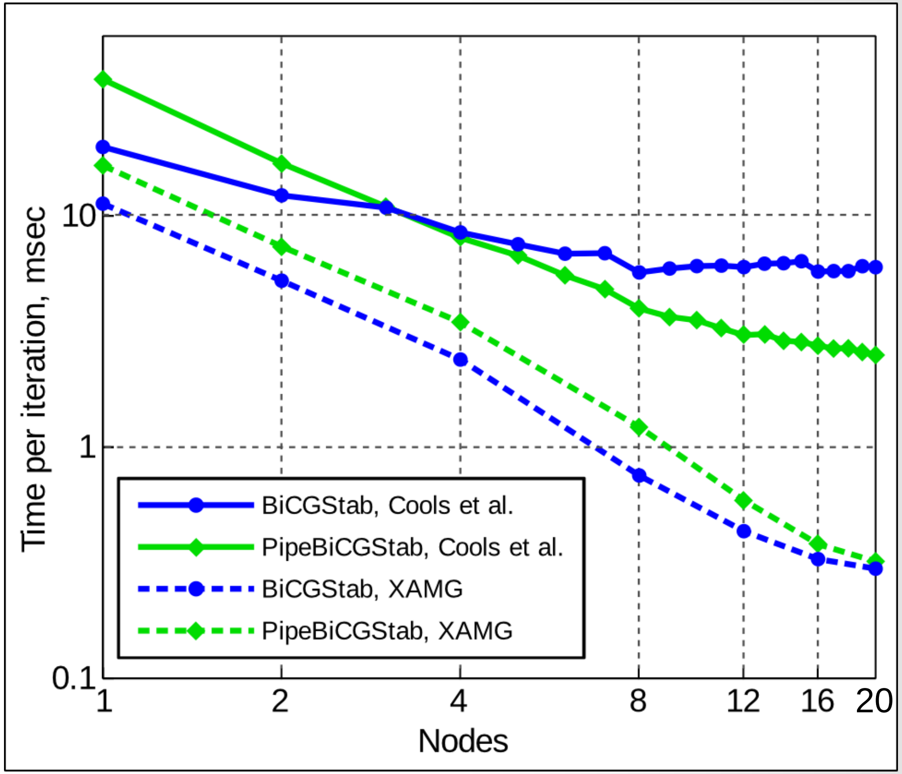
*S. Cools, W. Vanroose, Parallel Comput., 2017.

Our Numerical Experiments

Test problem: 5-diagonal matrix, with 10^6 unknowns

HPC system: 20+ nodes, 2x4-core Intel Xeon X5570, IB QDR (Lomonosov supercomputer)

MPI: Intel MPI 2017
I_MPI_ASYNC_PROGRESS = 0



Outline of the Talk

Plan:

1. Motivation
2. **Execution time model for Krylov subspace methods**
3. IMB-ASYNC benchmarks
4. Theoretical model validation
5. Comparison of BiCGStab methods

Analytical Model: Basics

Three basic aspects:

- Computations time
  - matrix-vector multiplications
  - vectors updates & dot products
- Communications time
  - local communications with neighbours (SpMV)
  - global reductions (dot products)
- Overlap of communications and computations

Execution Time Model

Classical BiCGStab method:

$$T^{BiCGStab}(p) = 22T_{vec}(p) + 2T_{SpMV}(p) + 3T_G(p)$$

Pipelined BiCGStab method:

$$T^{PipeBiCGStab}(p) = 43T_{vec}(p) + 2\max\left(T_{SpMV}(p) + \Delta_G, T_G(p)\right)$$

SpMV time:

$$T_{SpMV}(p) = \max\left(T_{mul}(p) + \Delta_L, T_L(l)\right)$$

Computation Time Estimates

Computation time:

$$T_{calc} = \frac{\Sigma}{B}$$

- matrix-vector multiplications (CSR):

$$T_{mul} = \frac{N(8(C+1) + 4(3C+1))}{b\,p}$$

- vector updates & dot products:

$$T_{vec} = \frac{8N}{b\,p}$$

Σ – total memory traffic

B – total memory bandwidth

b – single node bandwidth

p – number of nodes

N – matrix size

C – avg. nonzeros per row

Communication Time Estimates

Specific microbenchmarks to measure the communication times:

- Global communications time: generalized IMB *Iallreduce* benchmark

$$T_G(p,l) = 3.5 \cdot 10^{-6} + 1.7 \cdot 10^{-6}\, l^{0.21}\, p^{0.54}$$

- Local communications time: generalized IMB *Exchange* benchmark

$$T_L(l) = \begin{cases} 2.4 \cdot 10^{-6} + 6.9 \cdot 10^{-8}\, l^{0.56} & ,\ l \leq 2048 \text{ bytes} \\ 3.2 \cdot 10^{-6} + 2 \cdot 10^{-9}\, l & ,\ l > 2048 \text{ bytes} \end{cases}$$

*Correlations for Lomonosov supercomputer

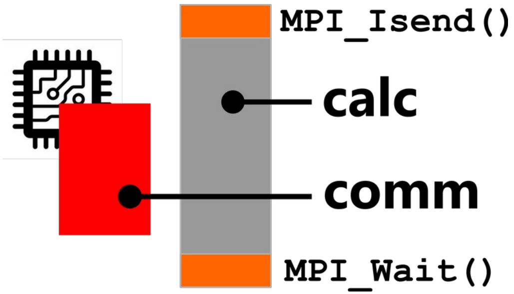
Outline of the Talk

Plan:

1. Motivation
2. Execution time model for Krylov subspace methods
3. **IMB-ASYNC benchmarks**
4. Theoretical model validation
5. Comparison of BiCGStab methods

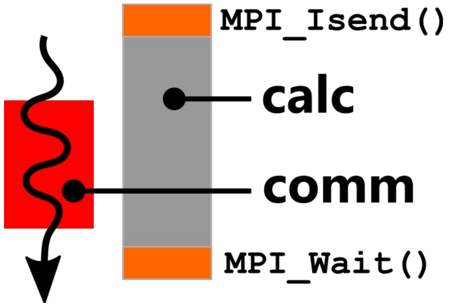
Communications Overlapping: Message Progression

Hardware supported



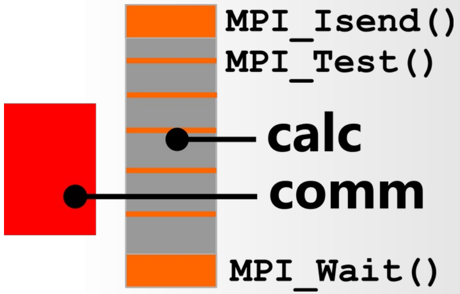
«automatic progress»

Software based



«progress threads»

Manual progress



periodical MPI_Test calls

Communications Overlapping: Efficiency estimation (1)

Which message progress for asynchronous communication is better?

- The answer depends on both hardware and software aspects
- Benchmark it to make right choice on a specific supercomputer
- No full and precise standard benchmarks exist in both IMB* and OSU suites

* IMB-NBC suite doesn't include point-to-point modes, no manual progress support and the overlap efficiency estimation methodology doesn't seem fully correct

Communications Overlapping: Efficiency estimation (2)

New **IMB-ASYNC** benchmark suite:

- Benchmarks **point-to-point** communication with non-blocking MPI_Isend/MPI_Irecv/MPI_Wait pattern
- Benchmarks **collective Allreduce** communication with non-blocking MPI_Iallreduce/MPI_Wait pattern
- To be added: non-blocking **neighborhood** communication, non-blocking **RMA** communication
- Source code:
  - https://github.com/a-v-medvedev/mpi-benchmarks

Communications Overlapping: Efficiency estimation (3)

“What is the overlap efficiency and how to correctly estimate it?”

- Always compare with the same blocking communication
- **100% efficiency** means: all the communications are hidden behind the computations
- **0% efficiency** means: no benefit if compared to blocking communication
- **negative** values: non-blocking communication makes things slower

More details:

A. Medvedev. Towards benchmarking the asynchronous progress of non-blocking MPI point-to-point and collective operations // Proceedings of ParCo conference, 2020 (in press).

Communications Overlapping: Benchmark Scenarios

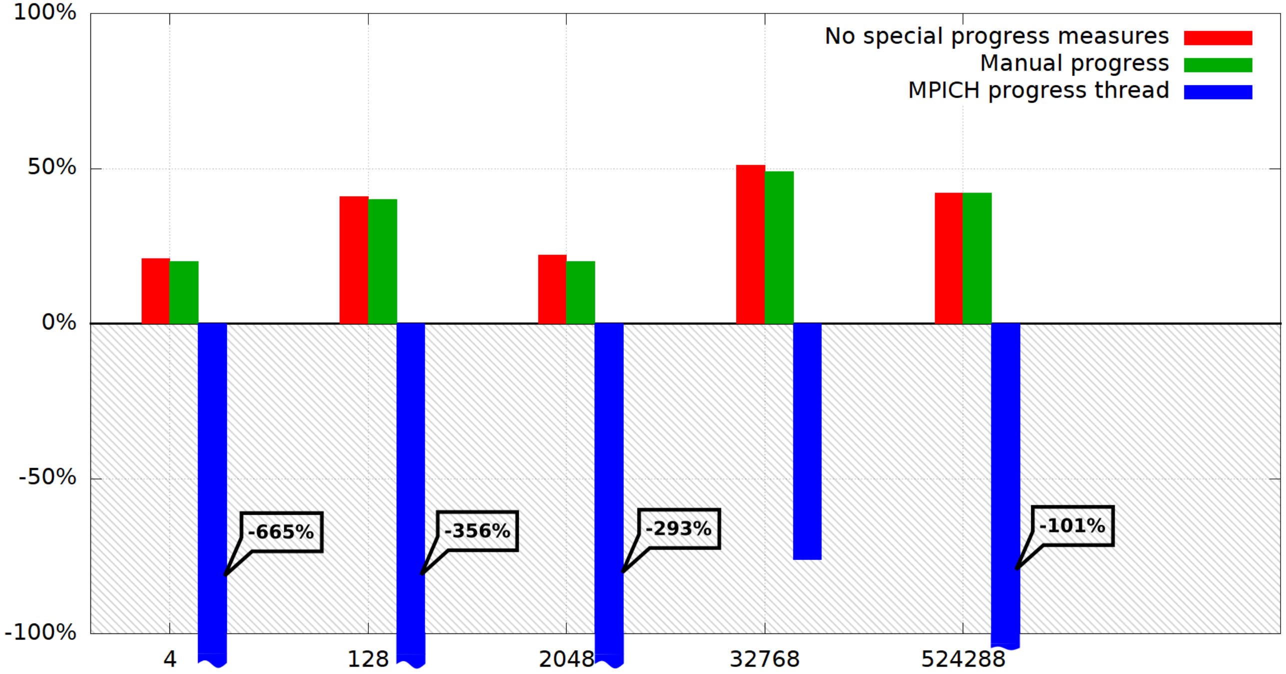
Experiments:

- **IMB-ASYNC** on Lomonosov-2, Intel MPI 2017
- 64 nodes, 14 ranks per node (full subscription)
- Message sizes from 16 bytes to 4 Mbytes

“No special progress measures”	just non-blocking communication as it is, overlapped with some simple computational load loop
“Manual progress”	benchmark code includes periodical **MPI_Tests()** calls inside the computational load loop
“MPICH progress thread”	MPICH-specific implementation of “progress thread” is turned on by setting **MPICH_ASYNC_PROGRESS=1**

Communications Overlapping: Local Communications

IMB-ASYNC (sync_p2p/async_p2p), Lomonosov-2, 64 nodes/14ppn, Intel MPI 2017



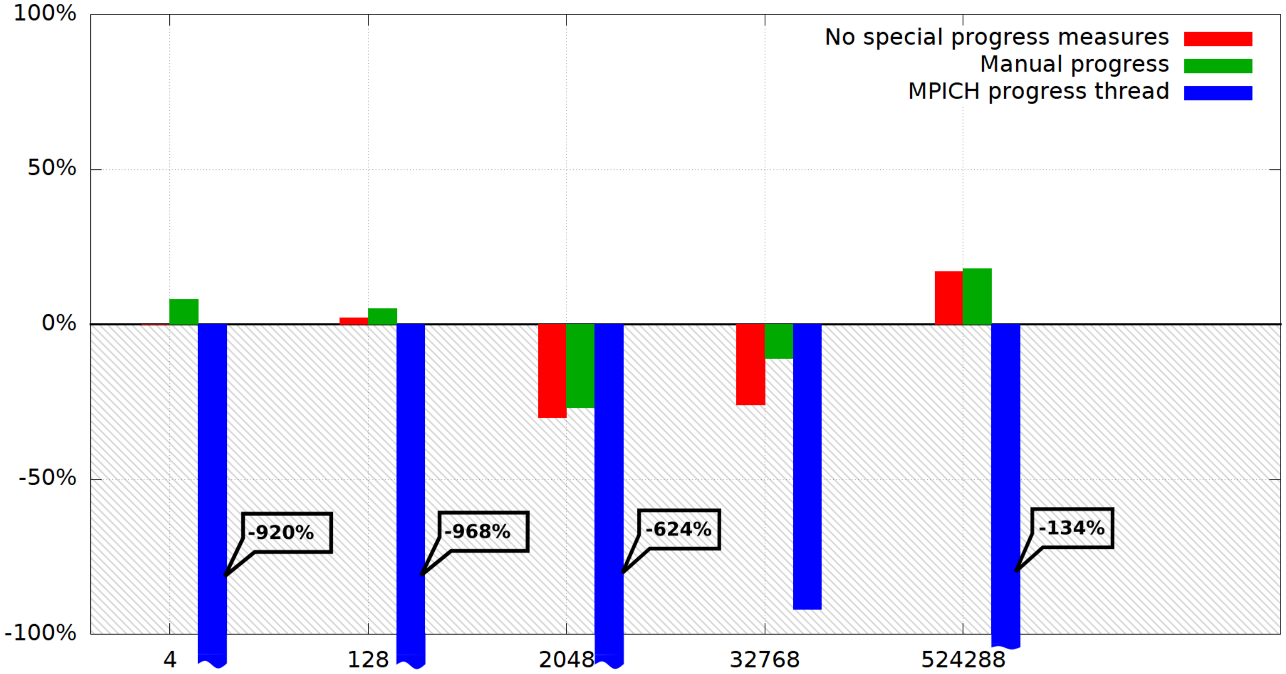
p2p communications provide efficient overlap of communications and computations without special efforts and overhead (**20-50% overlap efficiency**).

(MPICH_PROGRESS_THREAD=1 → great slowdown!)

$$T_{SpMV} = \max\left(T_{mul}(p) + \Delta_L, T_L(l)\right) \quad : \qquad \Delta_L \to 0$$

Communications Overlapping: Global Communications

IMB-ASYNC (sync_allreduce/async_allreduce), Lomonosov-2, 64 nodes/14ppn, Intel MPI 2017



No observable overlap for non-blocking global communications (**0% or negative overlap efficiency** mostly)

(MPICH_PROGRESS_THREAD=1 → great slowdown!)

$$T^{PipeBiCGStab} = ... + \max\left(T_{SpMV}(p) + \Delta_G, T_G(p)\right) \quad : \quad \Delta_G \rightarrow \gamma T_G$$

Outline of the Talk

Plan:

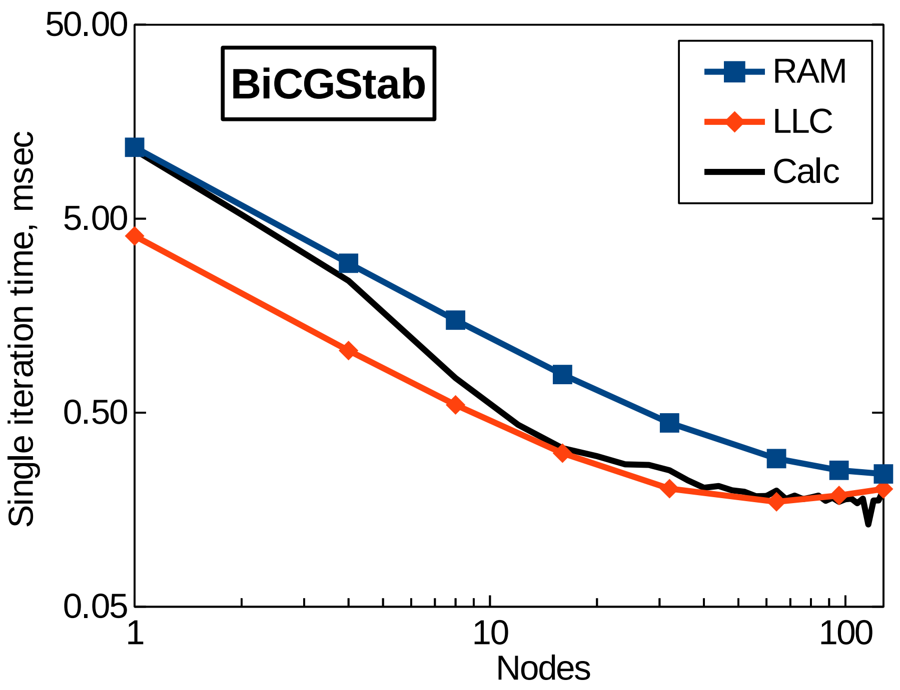
1. Motivation
2. Execution time model for Krylov subspace methods
3. IMB-ASYNC benchmarks
4. **Theoretical model validation**
5. Comparison of BiCGStab methods

Test Platforms

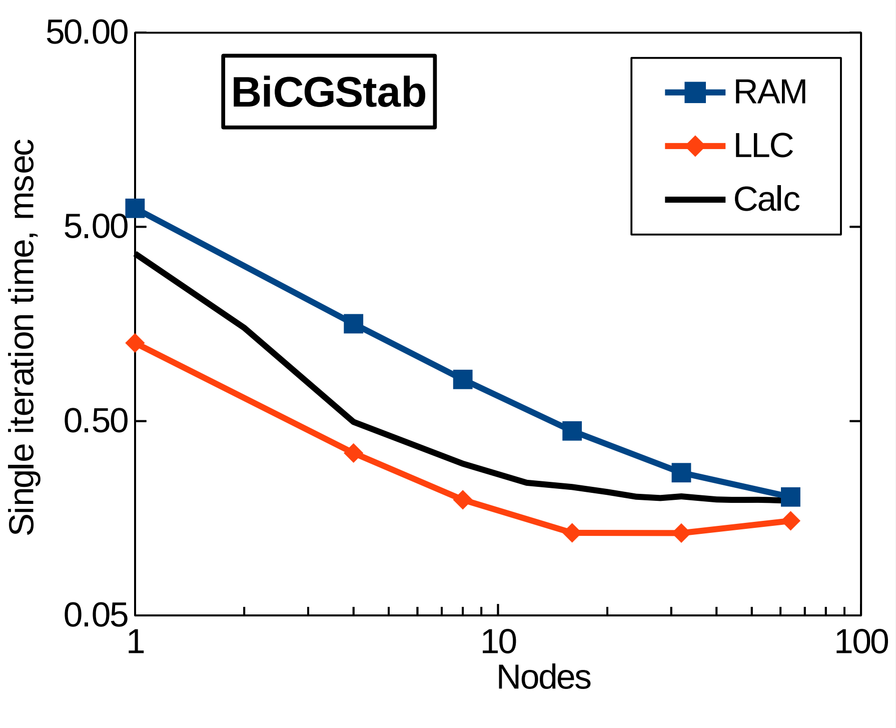
Supercomputer	Lomonosov	HPC5
Processors	Intel Xeon X5570	Intel Xeon E5-2650 v2
Cores	2 x 4 cores	2 x 8 cores
LLC size, MB	8	20
RAM bandwidth, GB/s*	16	40
LLC bandwidth, GB/s*	46	170
Interconnect	IB QDR	IB FDR
MPI library	Intel MPI 2017	Intel MPI 2017

*STREAM benchmark estimates

Model Validation, BiCGStab

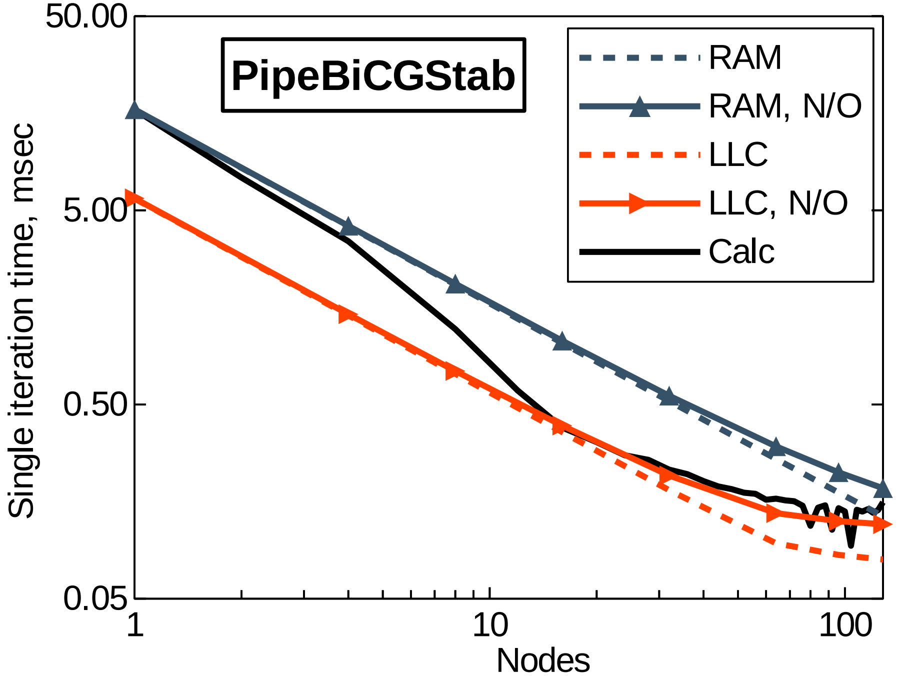


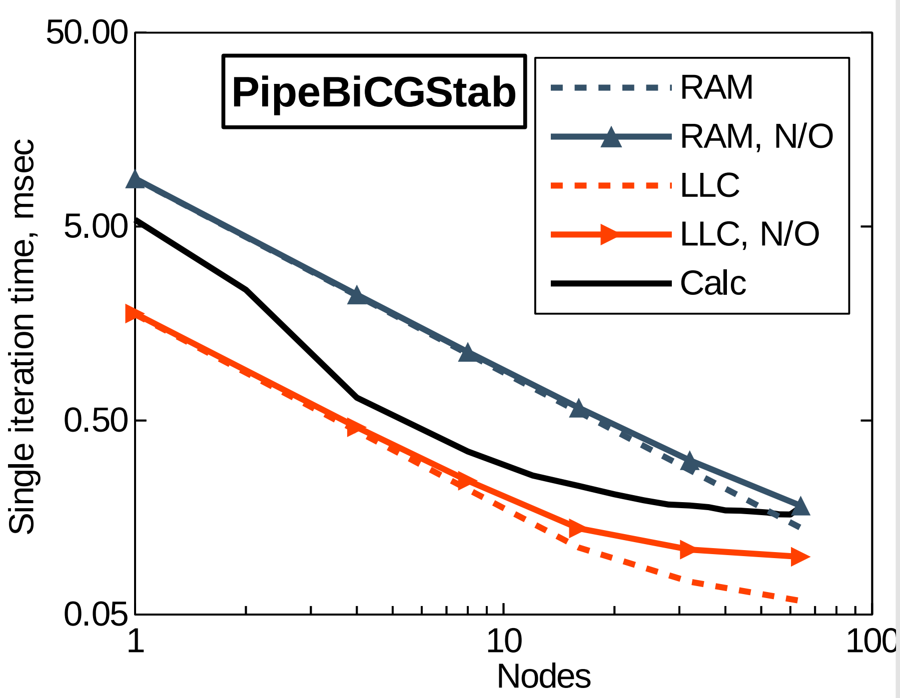
Lomonosov



HPC5

Model Validation, PipeBiCGStab





*no special message progression

Lomonosov

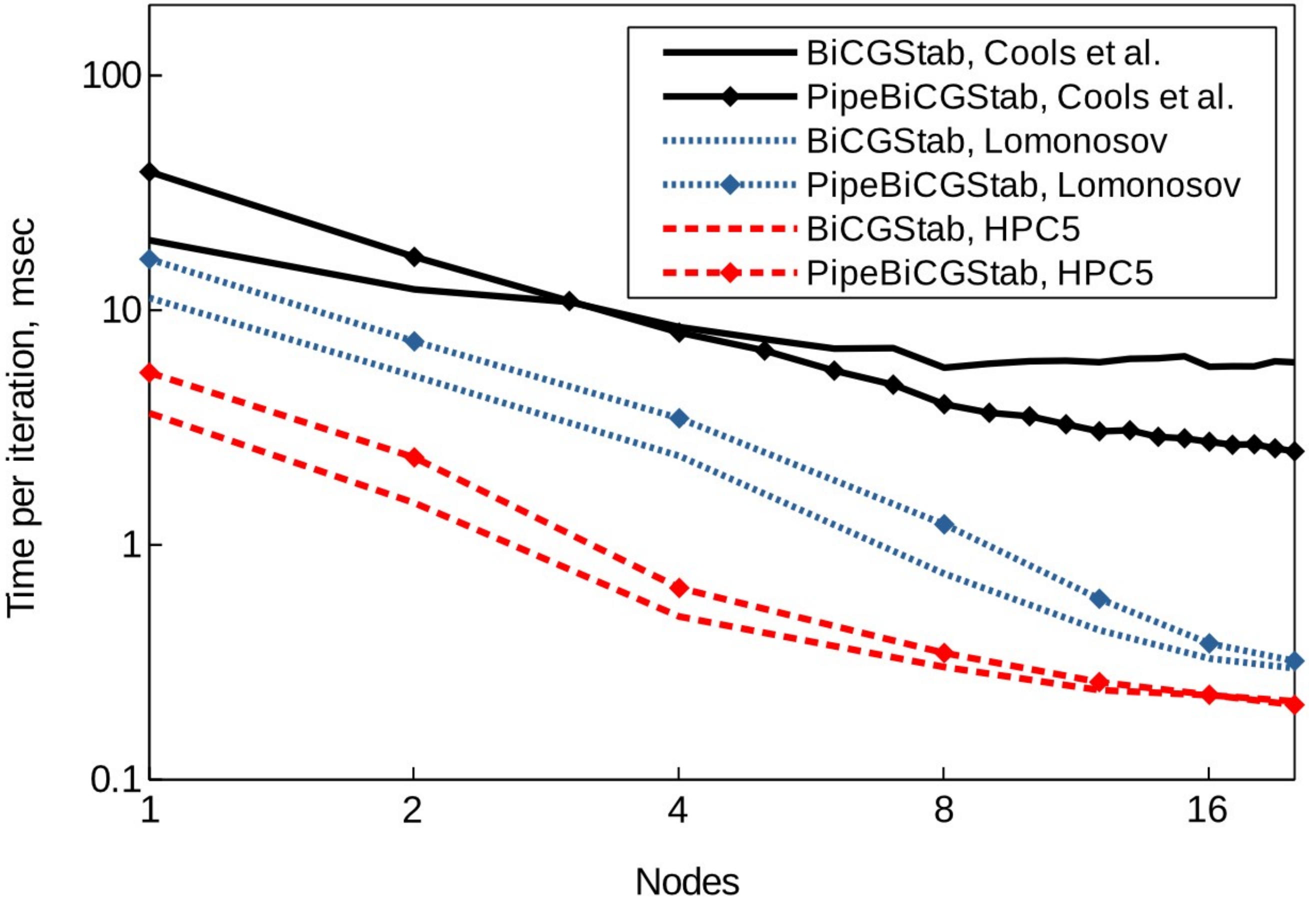
HPC5

Outline of the Talk

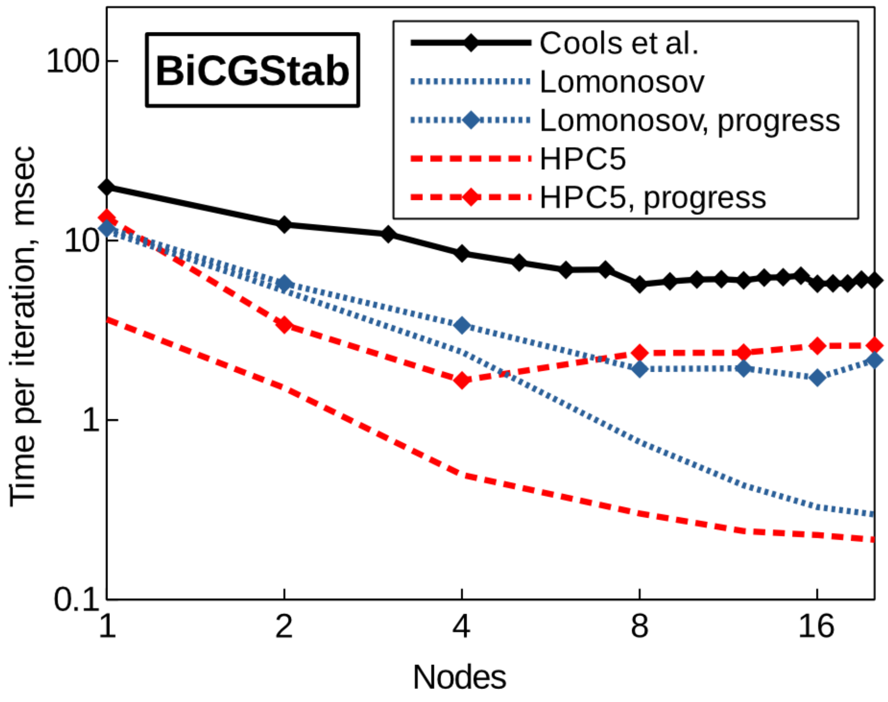
Plan:

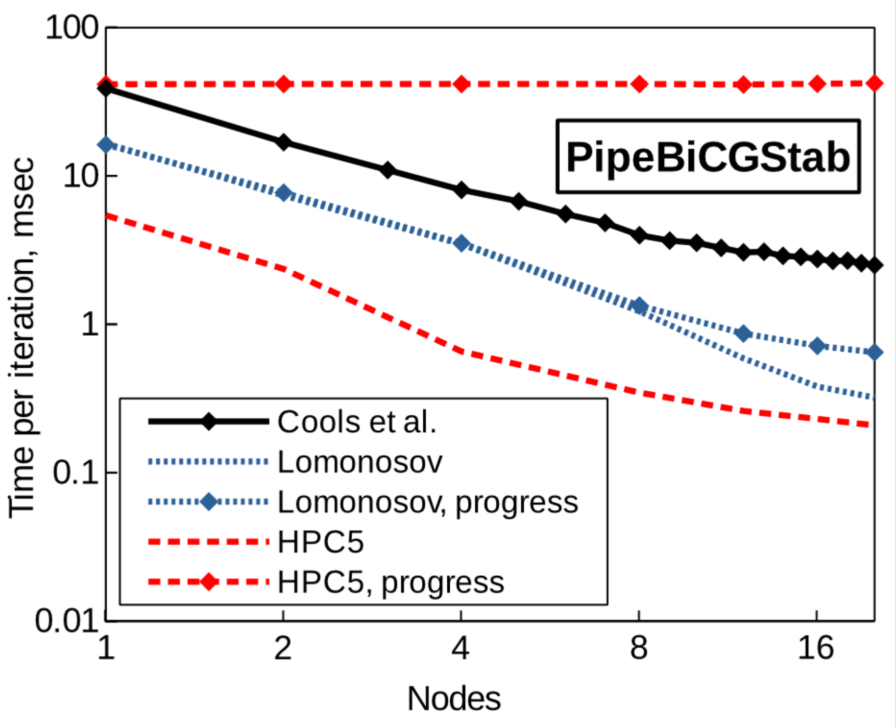
1. Motivation
2. Execution time model for Krylov subspace methods
3. IMB-ASYNC benchmarks
4. Theoretical model validation
5. **Comparison of BiCGStab methods**

“Default” Message Progression



Message Progression Threads





I_MPI_ASYNC_PROGRESS = 1

Conclusions

- The proposed analytical model allowed to validate the calculation results and compare efficiency of BiCGStab-like methods
- Message progression is inapplicable for the algorithms based on asynchronous global communications with short messages, at least for interconnects w/o corresponding hardware support
- All the calculation results must somehow be validated...

Publications

1. B. Krasnopolsky, Revisiting Performance of BiCGStab Methods for Solving Systems with Multiple Right-Hand Sides // **Computers & Mathematics with Applications**, 2019, doi:10.1016/j.camwa.2019.11.025 (arXiv:1907.12874)
2. A. Medvedev, Towards benchmarking the asynchronous progress of non-blocking MPI point-to-point and collective operations // Proceedings of ParCo conference, 2020 (in press).
3. B. Krasnopolsky, Predicting Performance of Classical and Modified BiCGStab Iterative Methods // Proceedings of ParCo conference, 2020 (in press).
4. A. Medvedev. IMB-ASYNC benchmark. https://github.com/a-v-medvedev/mpi-benchmarks
5. B. Krasnopolsky. An Approach for Accelerating Incompressible Turbulent Flow Simulations Based on Simultaneous Modelling of Multiple Ensembles // **Computer Physics Communications**, 2018, doi:10.1016/j.cpc.2018.03.023
6. B. Krasnopolsky, A. Medvedev. Acceleration of Large Scale OpenFOAM Simulations on Distributed Systems with Multicore CPUs and GPUs // Parallel Computing: On the Road to Exascale, 2016, doi:10.3233/978-1-61499-621-7-93

Welcome to join us if you're interested in HPC & CFD!