DisCostiC: Digital Twin Performance Simulations Unlocking Hardware-Software Interplay

Ayesha Afzal[1], Georg Hager[1], Gerhard Wellein[2]
[1]Erlangen National High Performance Computing Center (NHR@FAU)
[2]Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany

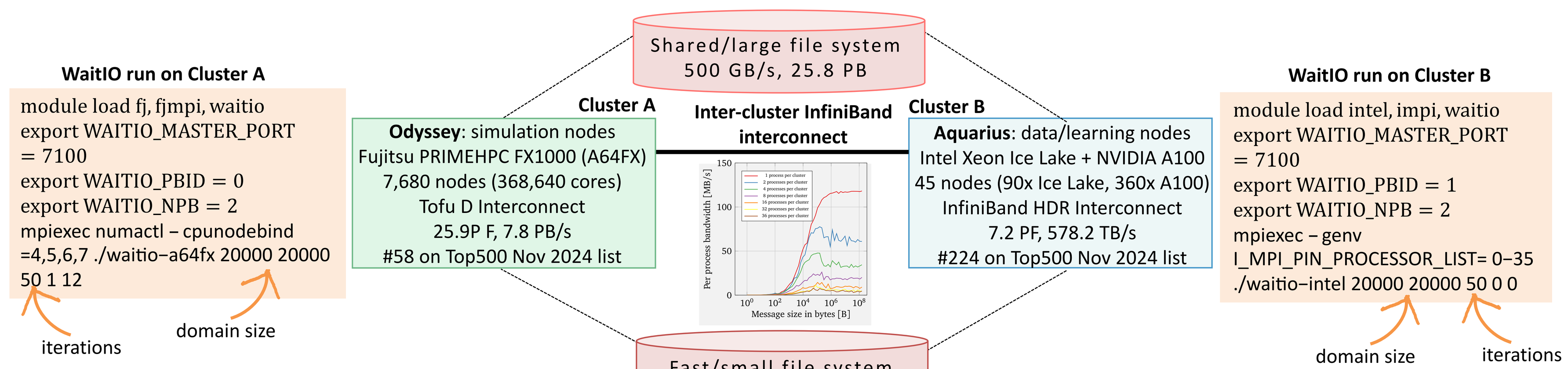
1 Problem and solution

- Hypothesis: predicted runtime = predicted computation runtime from models like Roofline and ECM + predicted communication runtime from models like Hockney and LogGP ➔ $t_{pred} = t_{comp} + t_{comm}$
- Reality: Hypothesis fails in certain cases ➔ $t_{actual} \lesseqgtr t_{pred}$
- Reason: Hardware-software interplay; complex interaction of noise, hardware bottlenecks, and overlapping contributions
- Existing techniques: Workflows require traces or running the application on a real system, adding ambiguous noise effects that obscure hardware-software interplay.
- Our technique: Full-scale simulator using first-principles analytic models across system hierarchies (cores, ccNUMA, chips, nodes, networks, clusters) ➔ DisCostiC
- Our technique strengths
  1. Distinctiveness: Uses DSEL-based application skeletons to capture inter-process dependencies, unlike trace-based simulators.
  2. Capabilities: Performance prediction, engineering, and design-space exploration for any parallel system.
  3. Accuracy: As accurate as the underlying white-box or gray-box models.
  4. Efficiency: Simulates hardware, code, and interactions with performance models, requiring no real hardware execution or trace collection, saving time and resources.
  5. Scalability: Supports large-scale studies, addressing hardware bottlenecks and effects not present at individual system hierarchy levels.

2 Capabilities: showcase 1

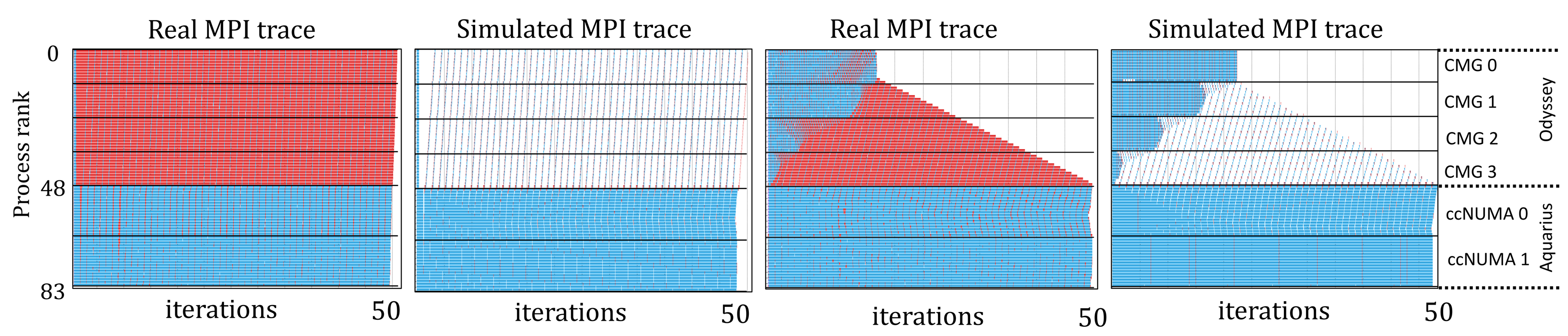
Wisteria/BDEC-01 system

- Odyssey with A64FX and Aquarius with Intel Ice Lake
- H3: Hierarchical, Hybrid, Heterogeneous
- h3-Open-SYS/WaitIO MPI library [1]



2D Four-Point Jacobi method

- Load imbalance between the two halves of the system (Odyssey & Aquarius), with one half in a lagging mode.
- Communication bottleneck across inter-cluster link, limiting scalability
- Simulation vs. real run of MPI-parallel Jacobi application on Wisteria using Vampir visualization; real trace: red (comm + idle times); simulation: red (comm), white (idle times)



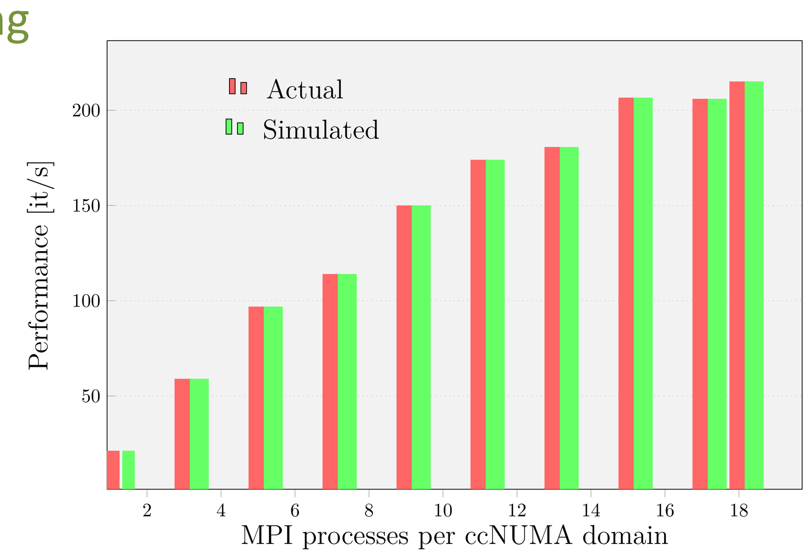
waitio_mpi_**recv** (.., bottom, ..)
➔ waitio_mpi_**send** (.., bottom, ..)
➔ waitio_mpi_**send** (.., top, ..)
➔ waitio_mpi_**recv** (.., top, ..)
➔ Jacobi $a_{i,j} = 0.25(a_{i-1,j} + a_{i,j-1} + a_{i+1,j} + a_{i,j+1})$

waitio_mpi_**irecv** (.., bottom, ..)
➔ waitio_mpi_**isend** (.., bottom, ..)
➔ waitio_mpi_**isend** (.., top, ..)
➔ waitio_mpi_**irecv** (.., top, ..)
➔ waitio_mpi_waitall (4, requests, statuses)
➔ Jacobi $a_{i,j} = 0.25(a_{i-1,j} + a_{i,j-1} + a_{i+1,j} + a_{i,j+1})$

3 Capabilities: showcase 2

Comparison of measured and predicted scaling

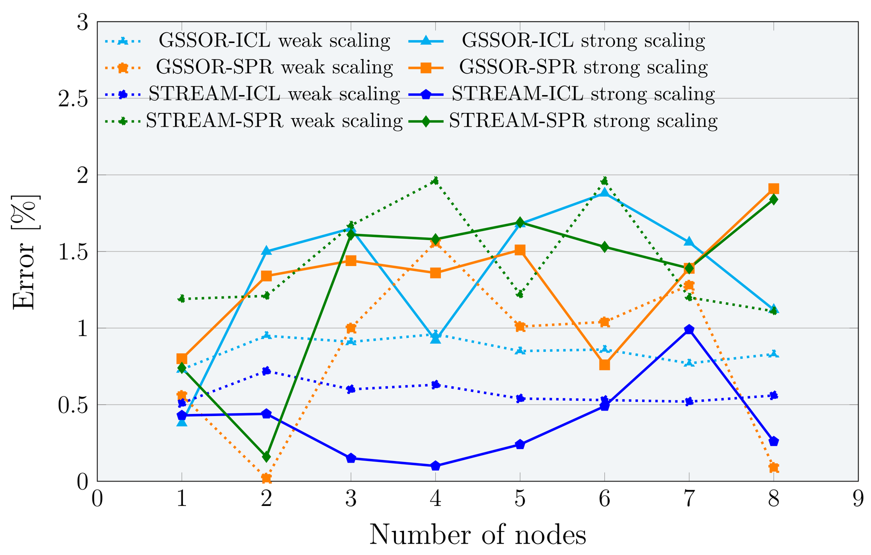
- Simulation vs. real-run performance of a memory-bound Jacobi application on 10 fixed nodes, varying the number of tasks per Ice Lake ccNUMA domain {1, 2, ..., 18} for 10K iterations on a $20k^2$ domain.
- Comparison reveals that DisCostiC accurately handles scaling, which occurs across ccNUMA domains, but not across cores within a single ccNUMA domain.



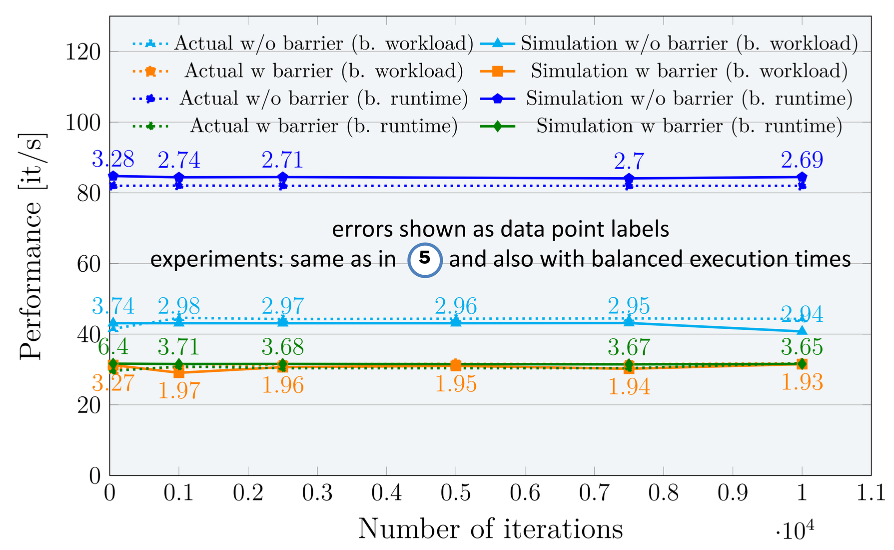
4 Accuracy

What's an effective metric for assessing the accuracy of a simulation? How can we determine it's accuracy?

- Is the error (actual run – simulated run) a reliable metric, considering the ambiguous noise effects in the actual run?
- Accurate ➔ **error < threshold**
- Tested various proxy structures of real-world applications, like Chebyshev filter diagonalization, Gauss-Seidel Successive Over-Relaxation (GSSOR), High Performance Conjugate Gradients, and Optical Flow Solver on Intel (Ice Lake ICL, Sapphire Rapids SPR) and non-Intel (Odyssey) systems.



In all scaling cases: error < 2%

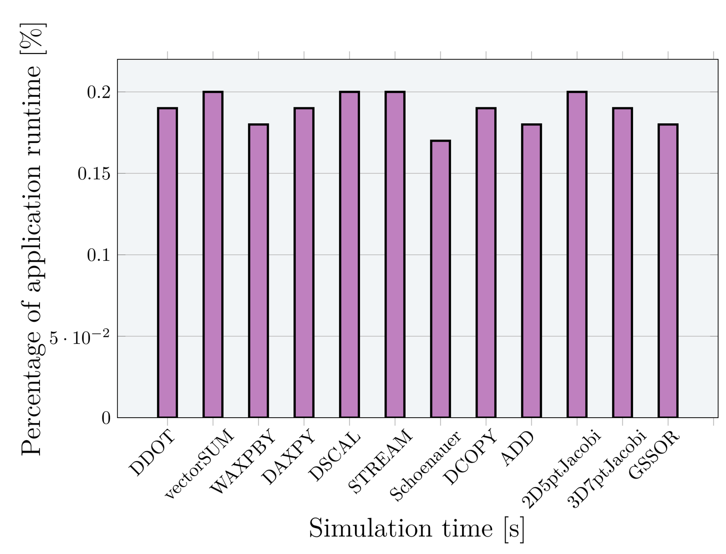


error remains same across number of iterations
→ simulations as accurate as the underlying models

5 Efficiency

How does the simulator perform?

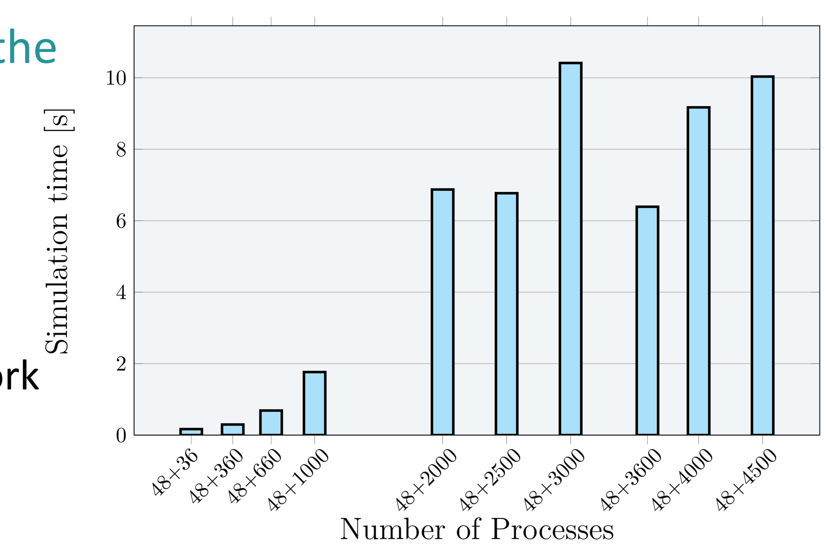
- Twelve benchmarks and applications were executed for approximately 1500 seconds on a single Sapphire Rapids node.
- For longer runs, the simulation time accounts for less than 1-2% of the total application runtime.
- For very short runs lasting less than a second, the real executions are more efficient, as expected.



6 Scalability

How does simulator performance vary with the number of processes?

- Simulation of 4548 processes: 48 Odyssey processes and 1 to 4500 Aquarius processes
- Simulator running with 2 to 4501 processes (simulation processes + 1) on Fritz hardware, utilizing Ice Lake CPUs and an InfiniBand network
- The simulator's performance trend with the number of processes indicates its scalability.



References

[1] Sumimoto et al., 2023. A System-Wide Communication to Couple Multiple MPI Programs for Heterogeneous Computing. DOI: 10.1007/978-3-031-29927-8_25
[2] Afzal et al., 2020. Desynchronization and Wave Pattern Formation in MPI-Parallel and Hybrid Memory-Bound Programs. DOI: 10.1007/978-3-030-50743-5_20
[3] Afzal et al., 2022. Analytic Performance Model for Parallel Overlapping Memory-Bound Kernels. DOI: 10.1002/cpe.6816
[4] Ujeniya, 2024. Extending a Simulation Framework for Performance Assessment of Parallel Applications
[5] Afzal et al., 2023. The Role of Idle Waves, Desynchronization, and Bottleneck Evasion in the Performance of Parallel Programs. DOI: 10.1109/TPDS.2022.3221085

Outlook

- Capabilities unlocking hardware-software interplay
- Good accuracy
- High efficiency
- Good scalability

Future work

- Experimentation with accelerated and non-accelerated programs
- Support for additional performance bottlenecks, e.g., caches bottleneck
- Support for energy consumption modelling

https://github.com/RRZE-HPC/DisCostiC-Sim

Acknowledgement

NHR FAU JHPCN Bundesministerium für Bildung und Forschung ITC FAU KONWIHR