1. INTRODUCTION AND MOTIVATION

Lightweight message passing simulation toolkit

GOAL

Simulation of large-scale applications by taking the socket-level performance properties of the hardware and software interaction into account

USE-CASES

Reproducibility and exploration of the dynamics of parallel programs on current and future supercomputers in a well-controlled environment, thereby saving resources and time

Domain-specific language (DSL) based simulator, traces in existing trace-based simulators do not comprise reliable inter-process dependency information and are superimposed by many effects coming from real systems, MPI library, etc.

Using analytical, first-principle models for execution and communication time predictions, taking socket-level bandwidth contention into account and no dependencies on external libraries

Comparison of simulated runtime prediction with experiments on the Magpie cluster (Intel Broadwell E5-2630 v4 2.2 GHz, 12 cores node) and Omni-Path interconnect of fat-tree topology and 100 Gbit (350 Mbps bandwidth per link and direction)

4. TESTING AND VALIDATION: DSL CODE EXAMPLE

Test case: MPI parallel 2D finite point Jacobi with 1D decomposition

5. TESTING AND VALIDATION: DSL MODES AND SIMULATED PERFORMANCE

Different ways of specifying a kernel in DSL

6. EVALUATION AND IMPLICATIONS

Use-case: idleness propagating across processes (idle wave speed)

Zoom-in view

7. FUTURE WORK AND REFERENCES

Beyond the exploration of future supercomputers, DisCostic can reproduce the case-studies performed on current systems in following references:

8. SUMMARY OF SIGNIFICANCE

Resource Efficient

1. No intermediate trace files unlike any offline trace-driven tools
2. Low memory requirement no need of target architecture code execution

Convenient, compact and practically usable programming interface (API)

Enables in-depth architectural exploration via analytic modeling of node-level bottlenecks without accounting the cost of the code contrast to existing trace-based parallel simulators

Low entry cost tool

1. Optional dependency on other tools (LRWID, KERNCRAFT, OSACA only as an add-on feature
2. No tool dependency for trace analysis and visualization

Enables in-depth architectural exploration via analytic modeling of node-level bottlenecks without accounting the cost of the code contrast to existing trace-based parallel simulators