DISCOSTIC: A DSL-BASED PARALLEL SIMULATION FRAMEWORK USING FIRST-PRINCIPLES ANALYTIC PERFORMANCE MODELS

Ayesha Afzal <ayesha.afzal@fau.de>, Georg Hager (advisor), Gerhard Wellein (advisor)

Erlangen National High Performance Computing Center | Department of Computer Science | Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

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-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

COMPARISON

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.

CHALLENGES

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

2. RELATED WORK AND FRAMEWORK FLOW

Try it yourself

RELATED WORK

Existing simulators: PSINS, xSim, HeSSE, DIMEMAS, LogGOPSim, MPI-SIM, SimGrid, COTSon, BigSim, SILAS, SST, MARS

Existing parallel simulators concentrate on communication aspects and ignore the analytical modelling of hierarchical structure below the node level with its component-level bottlenecks

DisCostiC | LIKWID | KERNCRAFT | Layer Conditioner | Pycachesim | OSACA

DISCOSTIC WORKFLOW

SIMULATOR DESIGN

In-core OSACA tool

Multi-core KERNCRAFT tool

layer condition analysis and cache simulator fallback

Likwid-perfctr of LIKWID suite

Parallel program running on cluster

a. User input — Cluster configuration and DSL-based parallel program

b. Analytic modelling — Analytical performance models at computation and communication levels

c. Simulated results — Simulation log file and visualization of trace file (JSON object data format) via Chromium browser (chrome://tracing or ui.perfetto.dev)

d. Testing and validation — Validation of models using measurement of derived hardware metrics

e. Parameter study with validation — Identification of governing principles concerning the interplay of parallel programs and components of supercomputers to target modelling and optimization challenges

3. SIMULATOR DESIGN

Simple two-bottleneck Roofline model for bottleneck identification and "light speed" estimates

Refined multi-bottleneck multi-core ECM model in the core and cache hierarchy,

Roofline → Increase prediction accuracy → Execution-Cache-Memory (ECM)

Node attributes (chip-level saturation, overlap, etc.) → Analytic runtime computation models (serial core and parallel chip level)

Prediction of computation time for arithmetic instructions and the amount of data loaded and stored in each memory level

Cluster topology, network-level bottlenecks and structure and domain knowledge of application (communication matrix, etc) → Management of input configuration, modelled predictions and MPI message semantics via generating an abstract syntax tree

Network attributes (eager limit, latency, bandwidth, etc.) → Analytic runtime communication models (network level)

Prediction of communication time for the latency and bandwidth of all P2P messages

Simple two parameter Hockney model, comparing the latency and bandwidth to decide latency or bandwidth bound code

Hockney → Increase prediction accuracy → LogGOPS

Refined multi-parameter LoGOPS model

4. TESTING AND VALIDATION: DSL CODE EXAMPLE

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

```
DisCostiC::Indextype left, right;
DisCostiC::Datatype phi[1002][1002];
DisCostiC::Event irecv, send, wait1, wait2 comp, comp2;
// all events initialize to DisCostiC::InvalidID
for (auto rank : DisCostiC::getRange(100))
{
    DisCostiC->Rank_Init(rank);
    left = rank - 1;
    right = rank + 1;
    if (left < 0)
        left = Discotic::NULLtype;
    if (left > 999)
        right = DisCostiC::NULLtype;
    for (auto timestep : DisCostiC::getRange(10000))
    {
        comp = DisCostiC->Comp(”LBL: JACOBI2D", comp2);
        if (rank > 0)
        {
        irecv = DisCostiC->Irecv(&phi[0][1], 1000, left, comp,
&req_left);
        send = DisCostiC->Send(&phi[1][1], 1000, left, comp);
        wait1 = DisCostiC->Wait(&req_left,&status_left);
        }
        else if  (rank < 99)
        {
        irecv = DisCostiC->Irecv(&phi[999][1], 1000, right, comp,
&req_right);
        send = DisCostiC->Send(&phi[998][1], 1000, right, comp);
        wait2 = DisCostiC->Wait(&req_right, &status_right);
        }
        comp2 = DisCostiC->Comp(”LBL: COPY", wait1, wait2);
    }
    DisCostiC->Rank_Finalize();
}
```

Comparison of simulated runtime prediction with experiments on the Meggie cluster (Intel Xeon Broadwell EP-E5-2630 v4, 2.2 GHz, two 10 cores NUMA domains per node, Omni-Path interconnect of fat-tree topology and 100 Gbits^{-1} raw bandwidth per link and direction)

DisCostiC

Distributed Cost in Clusters

Modeling complexity

Simulation accuracy, scalability, efficiency, features, modularity and portability

User-friendliness

5. TESTING AND VALIDATION: DSL MODES AND SIMULATED PERFORMANCE

Computation modes — Different ways of specifying a kernel in DSL

LBL mode — Kernel perf. by config parser

```
DisCostiC->Comp("LBL:JACIBI2D",
DisCostiC::Event DEPN_OP);
```

EXEC mode — Scaling from single-core execution properties

```
DisCostiC->Comp(”EXEC:
TOL=2.0||TnOL=1.0
|TL1L2=3.0|TL2L3=6.0|TL3Mem=14.2",
DisCostiC::Event DEPN_OP);
```

FILE mode — Run KERNCRAFT on kernel source file

```
DisCostiC->Comp("FILE:
jacobi2D.c//BREAK:JACIBI2D//
./broadwell.yml//20//-D N 1000",
DisCostiC::Event DEPN_OP);
```

SRC mode — Run KERNCRAFT on inline source

```
DisCostiC->Comp("SRC:
DisCostiC::Datatype
a[M][N],b[M][N],c[M][N],s;\n\n
for(DisCostiC::Indextype j=0;j<N-
1;j++)\n\tfor(DisCostiC::Indextype
i=0;i<M-1;i++)\n\tb[j][i]=(a[j][i-
1]+a[j][i+1]+a[j-1][i]+
a[j+1][i])*s;\n//BREAK:JACIBI2D//
./broadwell.yml//20//-D M 1000 N
1000", DisCostiC::Event DEPN_OP);
```

Input: string with node and kernel information (e.g., source code, machine file, number of cores, kernel parameters, etc.)

Script for KERNCRAFT interface

Output: Overwrite data of computation models in the config file of DisCostiC

Blocking/non-blocking modes of pair-wise communication in DSL

Communication modes — Sending or receiving

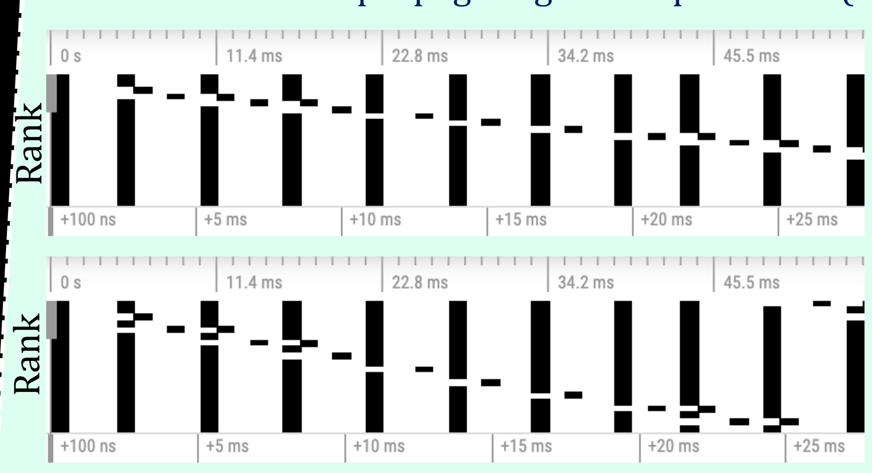
Irecv | Recv | Isend | Send

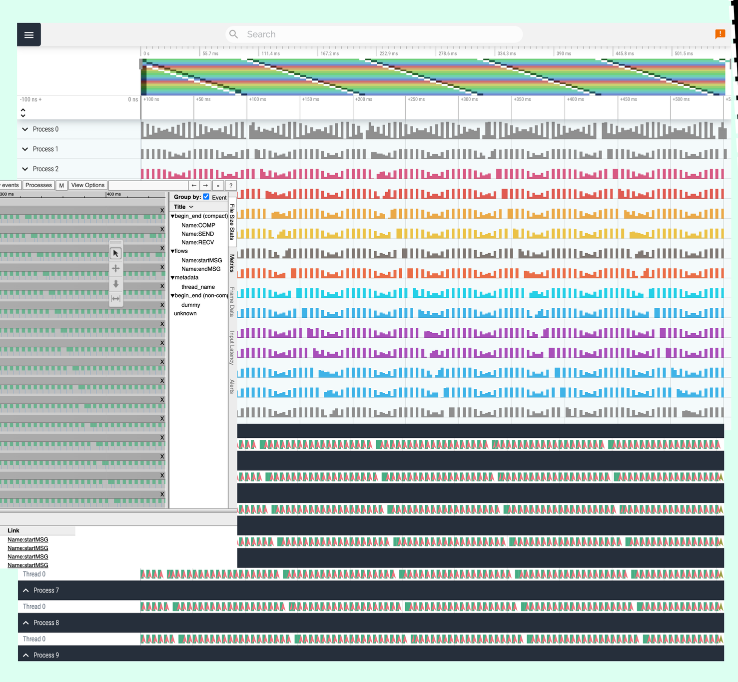
```
DisCostiC->X(DisCostiC::Datatype
Msg_in_bytes, DisCostiC::Indextype Dest_rank,
DisCostiC::Event DEPN_OP);
```

Accurate simulation of the execution of a massively parallel Jacobi code; any computation mode of Jacobi kernel in the DSL result in the same runtime prediction, with or without integration of external tools, like KERNCRAFT

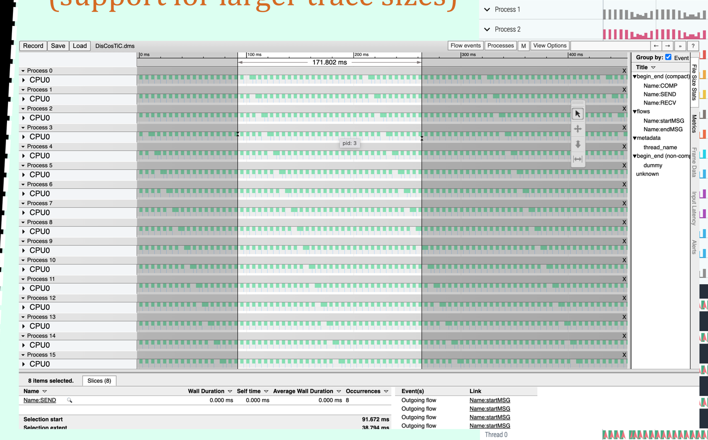
6. EVALUATION AND IMPLICATIONS

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





Latest ui.perfetto.dev browser (support for larger trace sizes)



Initial chrome://tracing browser (responsive trace navigation)

Visualization: massively parallel Jacobi code simulating for 10,000 iterations

Afzal et al.

[1] DOI: 10.1109/CLUSTER.2019.8890995

[2] DOI: 10.1007/978-3-030-78713-4_19

[3] DOI: 10.13140/RG.2.2.28259.50726

[4] DOI: 10.1007/978-3-030-50743-5_20

[5] DOI: 10.1002/cpe.6816

7. FUTURE WORK AND REFERENCES

ADVANCED MODELS

Including multi-core energy models, more MPI functionality including MPI 3.0 features, automatic generation of DSL code

HYBRID MODE

Extending the simulation framework to hybrid applications (MPI+X), incorporating X=OpenMP, CUDA, etc. including X-specific overheads

CONTENTION

Support network topologies and network-level contention

ADVANCED CASE STUDIES

End-user exploration tool for large space of proxy and real applications

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

[1] Ayesha Afzal, Georg Hager, and Gerhard Wellein. 2019. Propagation and Decay of Injected One-Off Delays on Clusters: A Case Study. In Proceedings - IEEE International Conference on Cluster Computing, ICCC.
[2] Ayesha Afzal, Georg Hager, and Gerhard Wellein. 2021. Analytic Modeling of Idle Waves in Parallel Programs: Communication, Cluster Topology, and Noise Impact. In Lecture Notes in Computer Science, vol 12728. Springer International Publishing, Cham.
[3] Ayesha Afzal, Georg Hager, and Gerhard Wellein. 2019. Delay Flow Mechanisms on Clusters. Poster at EuroMPI 2019.
[4] Ayesha Afzal, Georg Hager, and Gerhard Wellein. 2020. Desynchronization and Wave Pattern Formation in MPI-Parallel and Hybrid Memory-Bound Programs. In Lecture Notes in Computer Science, vol 12151. Springer International Publishing, Cham.
[5] Ayesha Afzal, Georg Hager, and Gerhard Wellein. 2022. Analytic Performance Model for Parallel Overlapping Memory-Bound Kernels. Concurrency and Computation: Practice and Experience. 34(10), e6816.

8. SUMMARY OF SIGNIFICANCE

RESOURCE EFFICIENT

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

STANDALONE

Low entry cost tool since
1. Optional dependency on other tools (LIKWID, KERNCRAFT, OSACA), only as an add-on feature
2. No tool dependency for trace analysis and visualization

USER-FRIENDLINESS

Convenient, compact and practically usable application programming interface (API)

UNIQUE SELLING POINT

Enables in-depth architectural exploration via analytic modelling of node-level bottlenecks without executing the code, in contrast to existing trace-based parallel simulators

Supported by KONWIHR OMI4papps | NHR@FAU | FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

EMBEDDED CONTENT http://tiny.cc/PASC22_Talk

POSTER ARTIFACT http://tiny.cc/PASC22-AD