Introduction to MCL

Roberto Gioiosa

Rizwan Ashraf, , Ryan Friese, Lenny Guo
Alok Kamatar, Gokcen Kestor
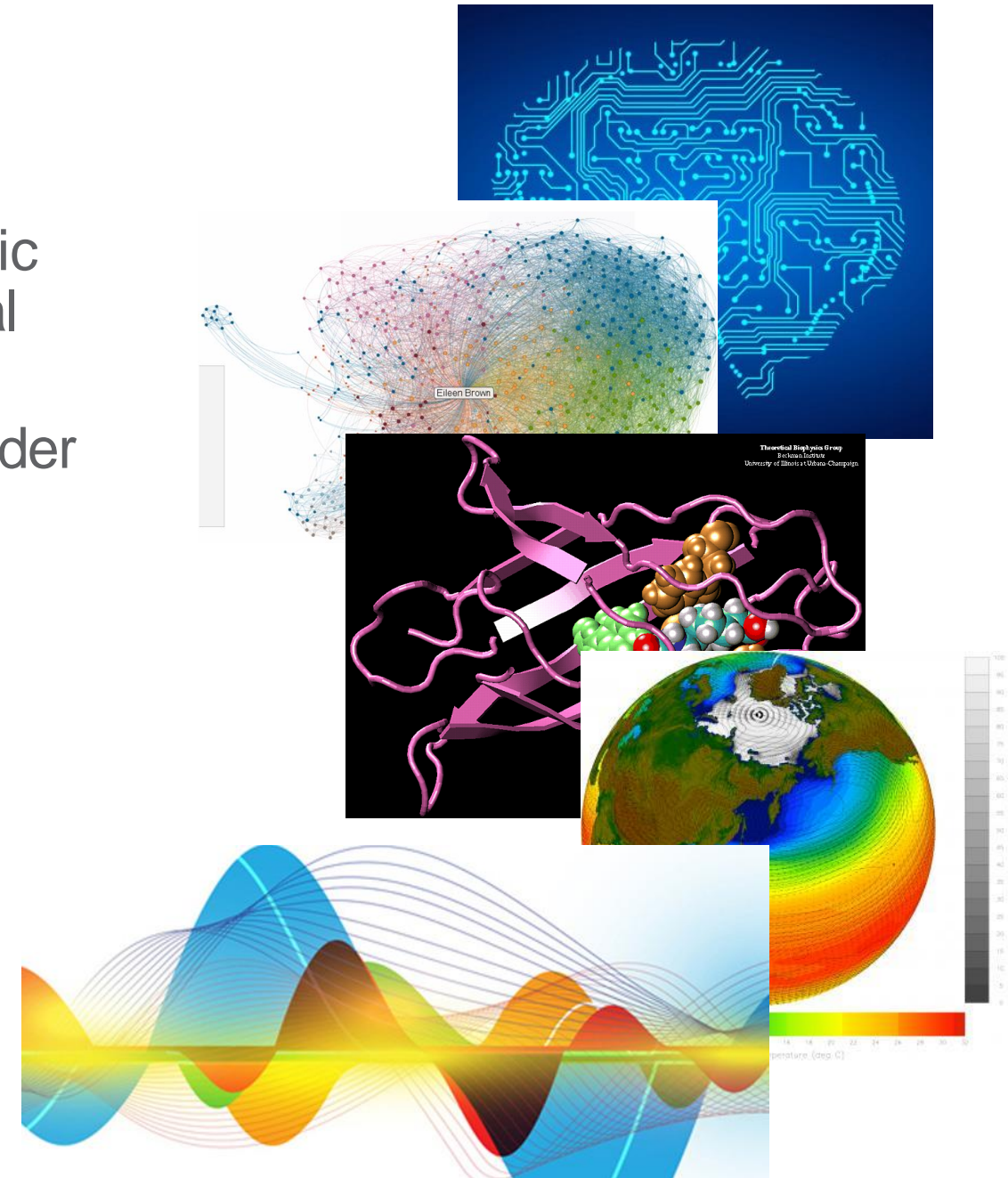
New Challenges

- New challenges
  - Emerging applications in different domains (scientific simulations, machine learning, data analytics, signal processing, etc.)
  - Unprecedented amount of data to be processed under strict real-time, power, and trust constraints

- Providing high-performance, scalable, and versatile solutions becomes a fundamental requirement

More and More Specialization

- Specialization has become a fundamental pillar for the design of future high-end computer systems:
  - In HPC: GP-GPUs, FPGAs,…;
  - In Military: ASICs, DSPs, …
  - Specialized hardware for machine- and deep-learning (Tensor Cores, NVDLA, SambaNova, Cerebras,…).

- High level of specialization results in extremely heterogeneous systems that are complicated to design, test, validate, and program

Accelerators are only useful if they are accessible…

Scaling Up and Down

IBM Summit

NVIDIA DGX-1
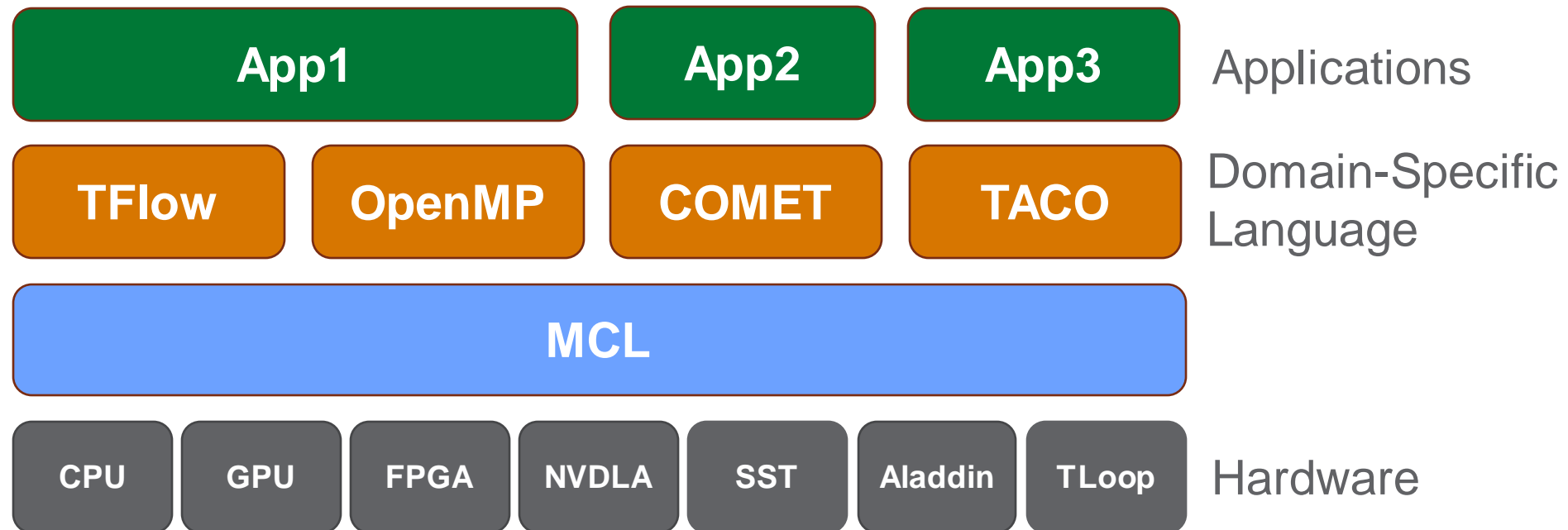(P100/V100)

Apple iMac Pro

Apple MacBook Pro

Xilinx MPSoC ZynQ ZCU 102/106

The Minos Computing Library (MCL)

- Framework for programming extremely heterogeneous systems
  - Programming model and programming model runtime
  - **Abstract low-level architecture** details from programmers
  - **Dynamic** scheduling of work onto available resources

- Key programming features:
  - Applications factored into tasks
  - **Asynchronous** execution
  - Devices are managed by the scheduler
  - Co-schedule **independent applications**
  - Simplified APIs and programming model (based on OpenCL)

- Flexibility:
  - Scheduling framework
  - **Multiple scheduling algorithms** co-exist
  - Code portability
  - Resources allocated **at the last moment**

Convergence of HPC/AI/Data Analytics

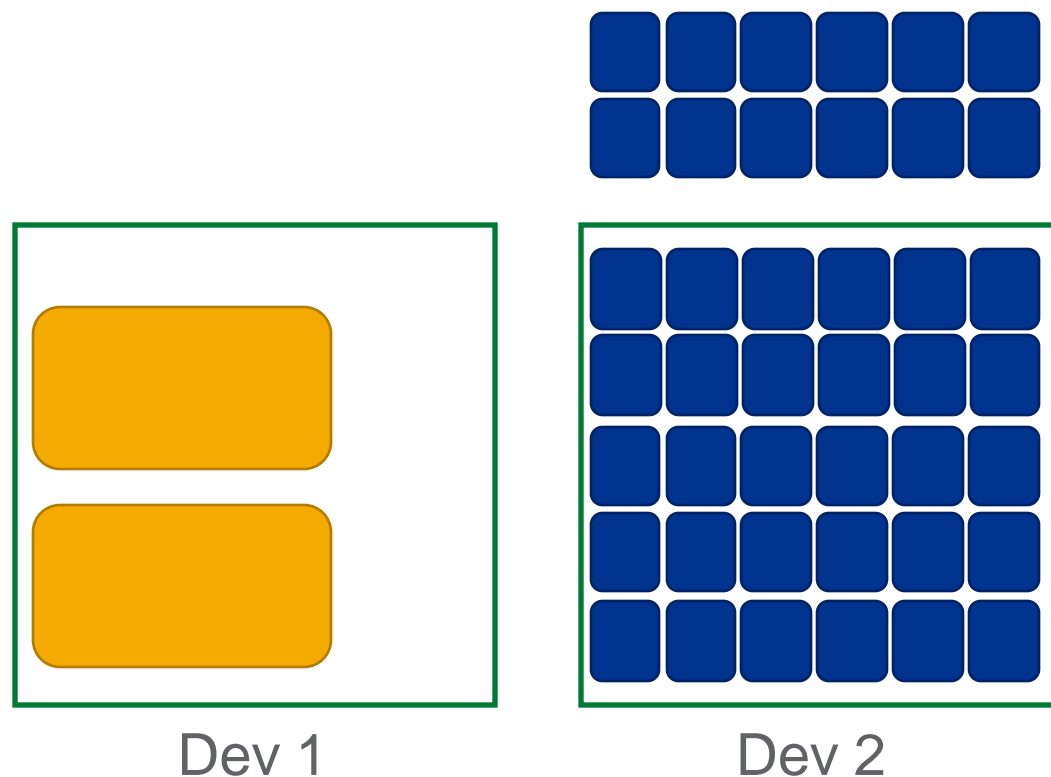| App1	App2	App3	Applications
TFlow **OpenMP**	**COMET**	**TACO**	Domain-Specific Language
MCL			
CPU GPU FPGA NVDLA SST Aladdin TLoop			Hardware

- Scientists express their algorithm with high-level DSLs that provide domain-specific programming abstractions

- Compiler lowers DSL code to device-specific, highly-optimized code

- Dynamic runtime coordinates access to computing resources and data transfers <u>across different applications</u>
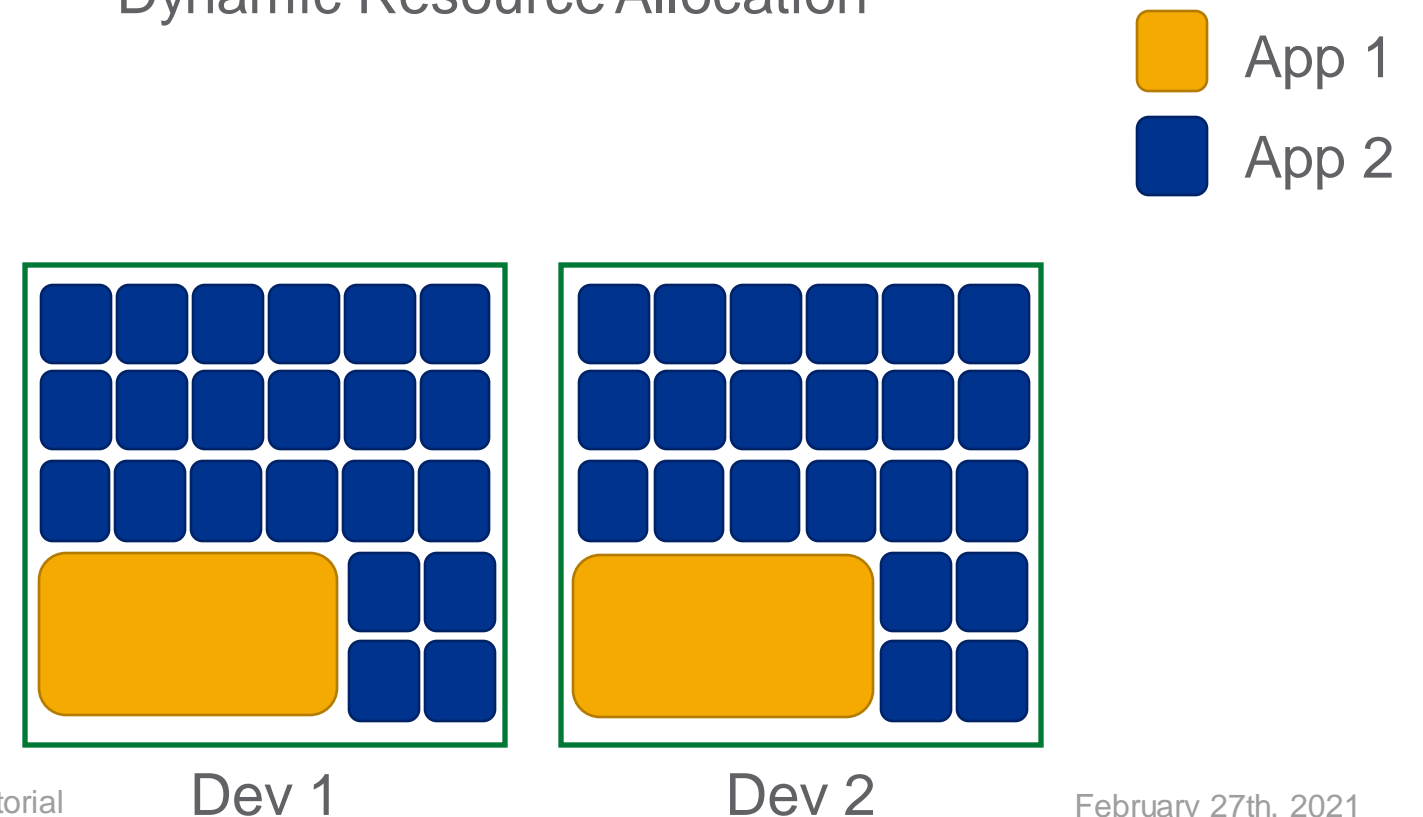
Multi-Process support

- A key distinctive features of MCL is that it supports multi-process, multi-threaded workloads
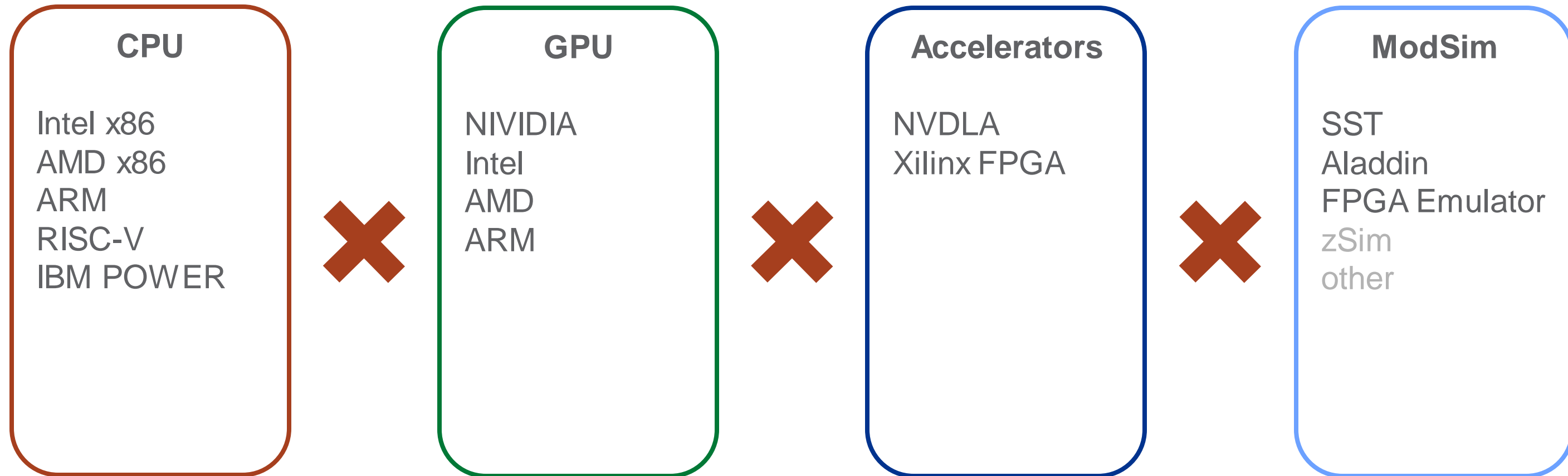 - Global optimization across the workload
  - Dynamic resource allocation

Static Resource Allocation

Dynamic Resource Allocation

App 1
App 2

Dev 1   Dev 2   Dev 1   Dev 2

Supported Architectures

CPU		**GPU**		**Accelerators**		**ModSim**
Intel x86 AMD x86 ARM RISC-V IBM POWER	✖	NIVIDIA Intel AMD ARM	✖	NVDLA Xilinx FPGA	✖	SST Aladdin FPGA Emulator zSim other

MCL works and/or can be integrated with other technologies commonly used in HPC, Data analytics, and ML.

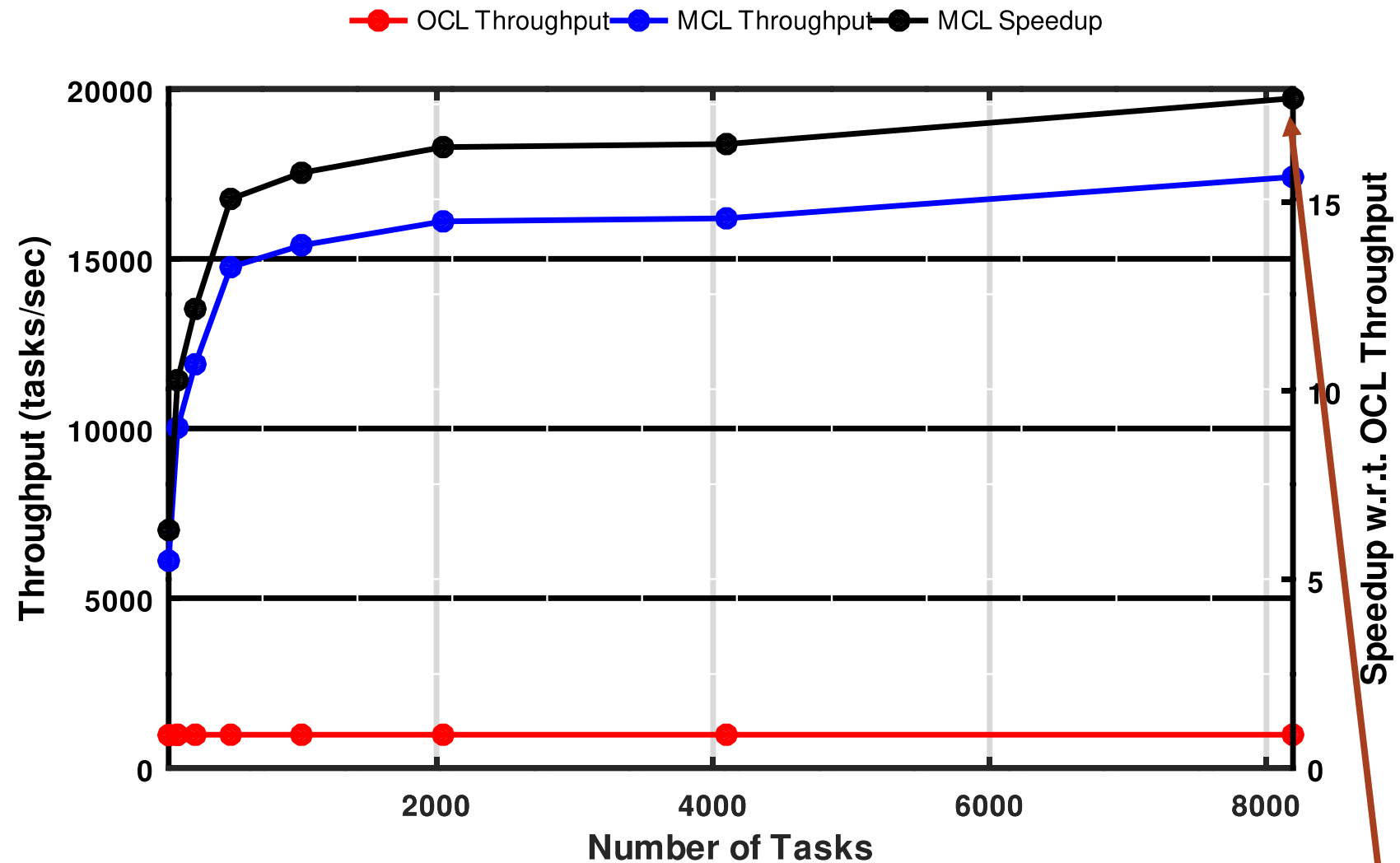# Throughput for Large Computations
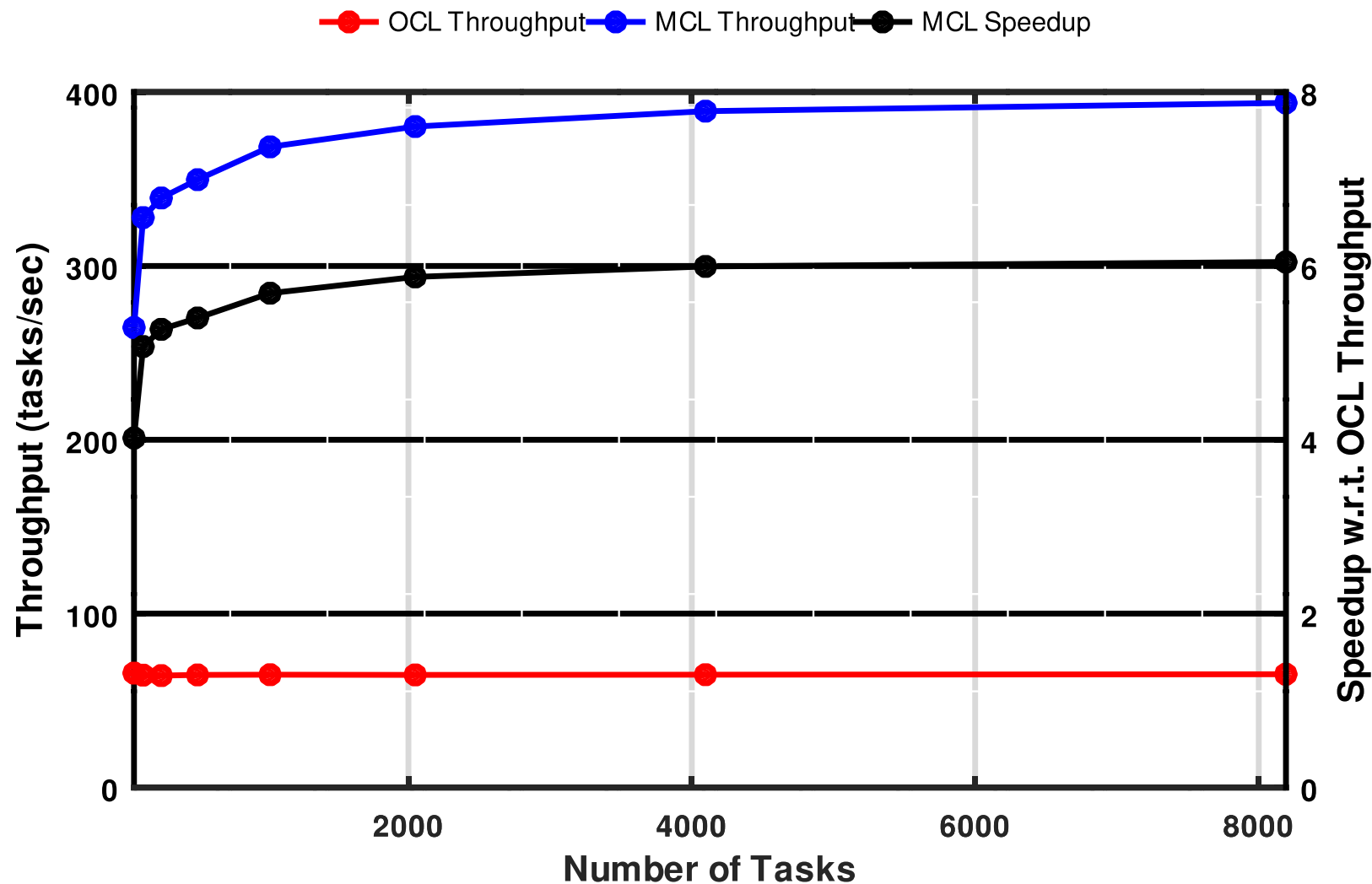
- DGX-1 V100
  - 2x 20-core Intel Xeon
  - 256 GB RAM
  - 8x NVIDIA V100, 16GB
- Benchmark:
  - DGEMM kernel
  - 64-8k tasks
  - 1024x1024
  - Compute-bound

Automatic scaling to 8 GPUs ☺

Sub-linear speedup up (6x) ☹

* Same programming effort, not same hardware (MCL>OCL)
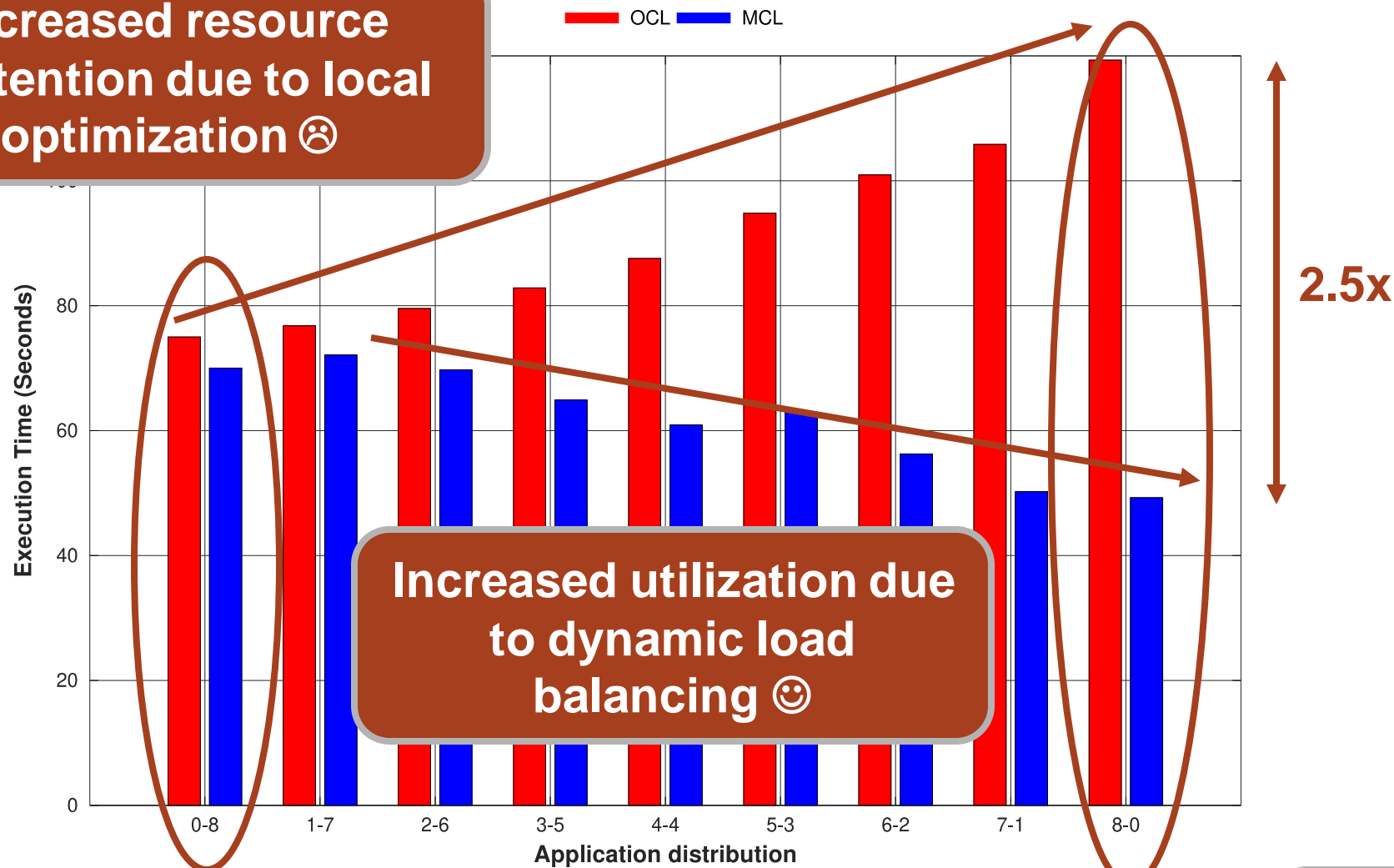
Throughput for Small Computations

- DGX-1 V100
  - 2x 20-core Intel Xeon
  - 256 GB RAM
  - 8x NVIDIA V100, 16GB
- Benchmark:
  - DGEMM kernel
  - 64-8k tasks
  - 64x64
  - I/O-boud (CPU-GPU) -> most challenging case

Automatic scaling to 8 GPUs ☺

Super-linear speedup GPUs ☺

Application Composition

Increased resource contention due to local optimization ☹

Increased utilization due to dynamic load balancing ☺

All large

All small

2.5x
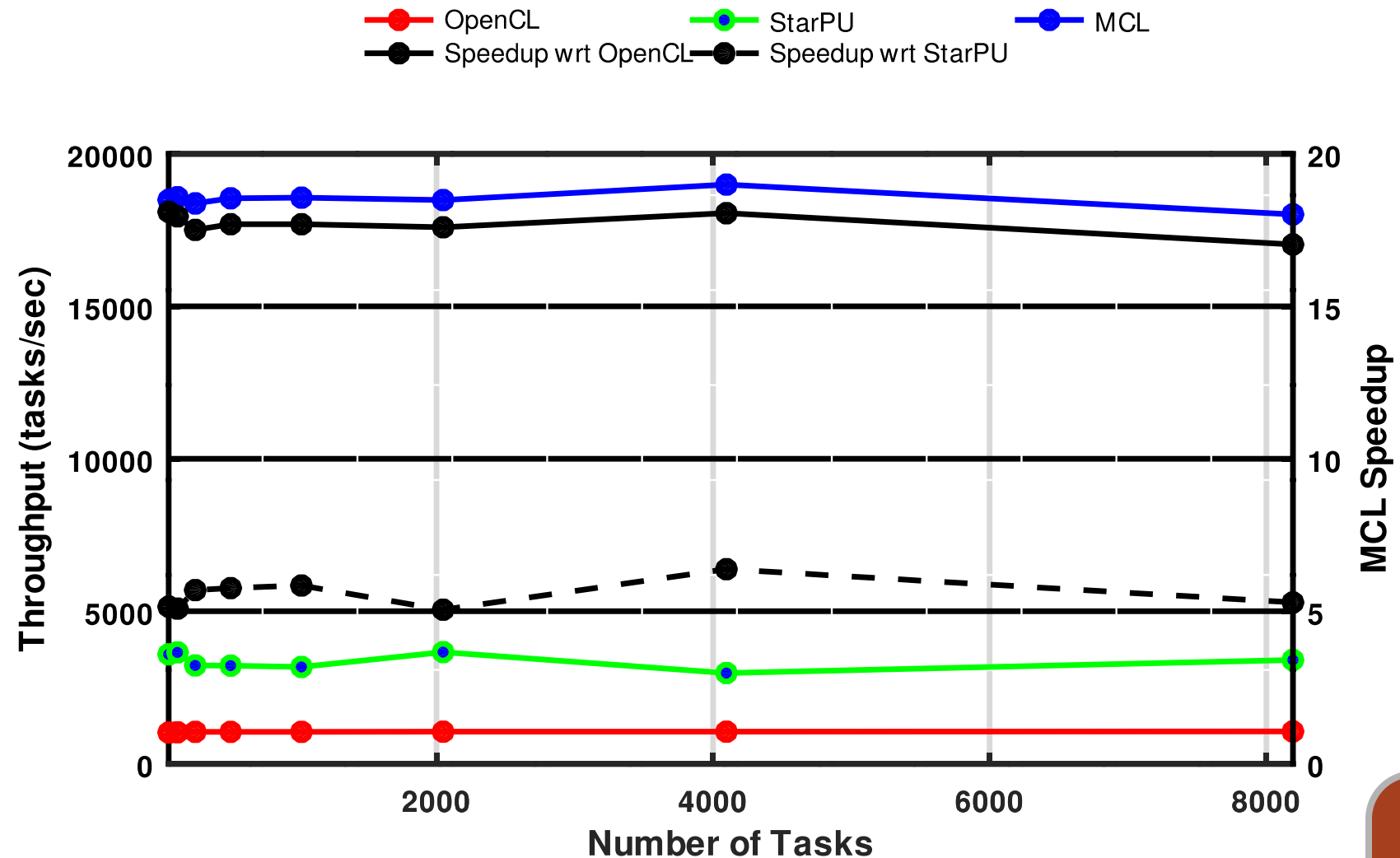
* Same hardware, not same programming effort (OCL>MCL)

- DGX-1 V100
  - 2x 20-core Intel Xeon
  - 256 GB RAM
  - 8x NVIDIA V100, 16GB

- Benchmark:
  - DGEMM kernel
  - Mix of small and large kernels (8 processes)
  - OpenCL modified for static resource allocation
  - No modification to MCL code

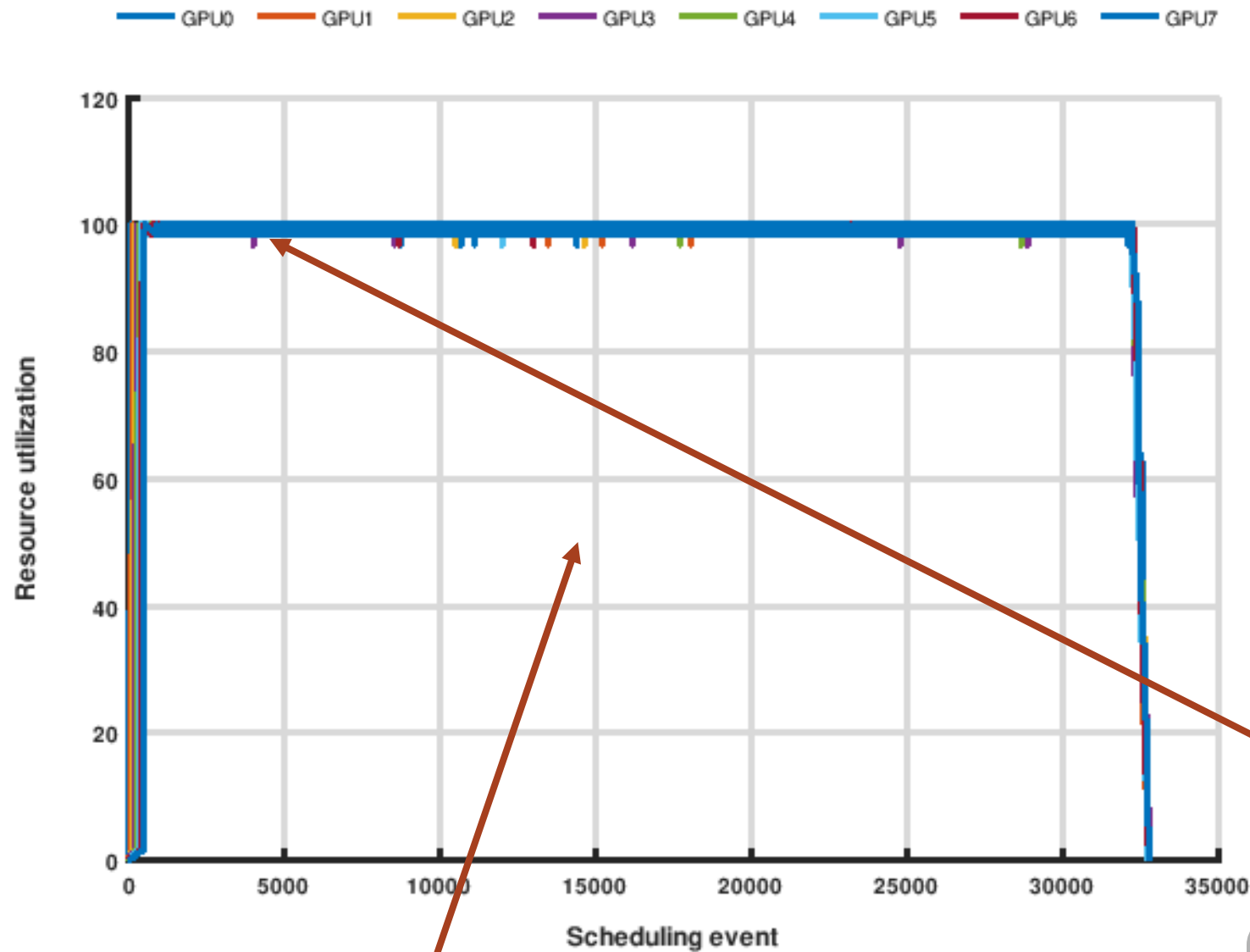**Dynamic resource allocation = faster execution** ☺

Comparison with StarPU

- DGX-1 V100
  - 2x 20-core Intel Xeon
  - 256 GB RAM
  - 8x NVIDIA V100, 16GB

- Benchmark:
  - DGEMM kernel
  - 64-8k tasks
  - 64x64

- Same kernel
- Similar code effort

LOC	
OpenCL:	160
StarPU:	120
MCL:	60

System utilization

- DGX-1 V100
  - 2x 20-core Intel Xeon
  - 256 GB RAM
  - 8x NVIDIA V100, 16GB
- Benchmark:
  - DGEMM kernel
  - 16k tasks
  - 1024x1024

Full utilization after transient phases

Effective dynamic load balancing!

MCL internal tracing

Publications

1. R. Gioiosa, B. Mutlu, S. Lee, J. Vetter, G. Picierro, M. Cesati. 2020. The Minos Computing Library: Efficient Parallel Programming for Extremely Heterogeneous Systems. In General Purpose Processing Using GPU (GPGPU '20), February 23, 2020, San Diego, CA, USA

2. G. Kestor, R. Gioiosa, M. Raugas. Towards Performance Portability through an Integrated Programming Eco-System for Tensor Algebra. In Performance, Portability, and Productivity in HPC Forum, Virtual, September 2020

3. A. V. Kamatar, R. D. Friese and R. Gioiosa, "Locality-Aware Scheduling for Scalable Heterogeneous Environments," 2020 IEEE/ACM International Workshop on Runtime and Operating Systems for Supercomputers (ROSS), GA, USA, 2020

Thank you