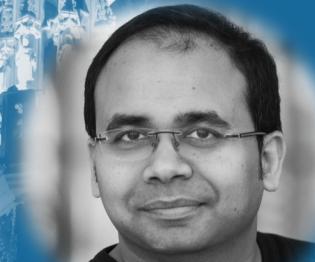


# Scalable System Simulation for Large Language Model Workloads



**Debjyoti Bhattacharjee**  
*Principal Member of Technical Staff*



**Arindam Mallik**  
*Department Director*



*Learn. Simulate. Scale.*

An aerial photograph of the Imec research campus in Leuven, Belgium. The image shows a large complex of modern buildings, including several large industrial facilities with flat roofs and multiple stories, surrounded by green trees and fields. A major highway runs through the foreground. The overall scene is a mix of industrial infrastructure and natural landscape.

Imec is more than just another R&D hub.

# What makes us the semiconductor lab of the world?



skilled  
people



world-class  
infrastructure



global  
ecosystem

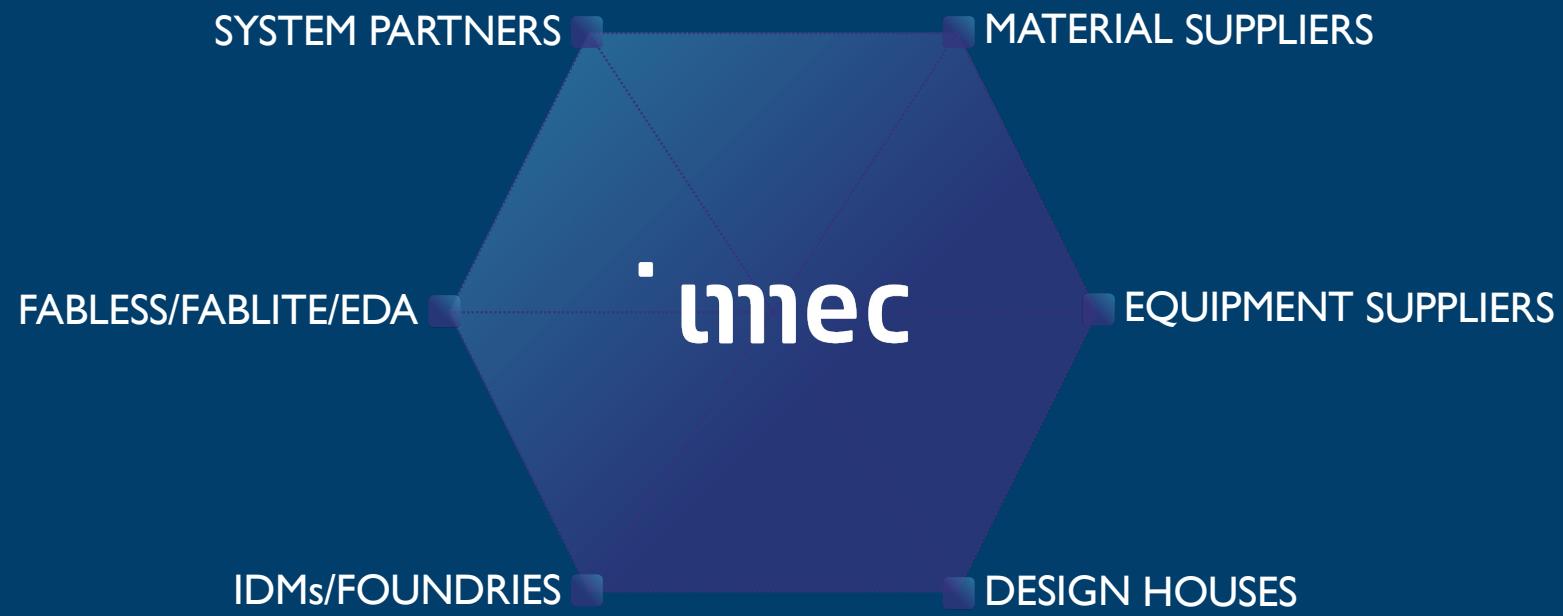
Zhen • John • Wilker • Feiyang • Roger • Xunqin • Adrien • Adarsham Solomon • Casper • Jaze • Thomas • Hien • Samman • Jie • Hyun • Kise • Koude • Jerryco • Julian • Ira • Taiwan • Venecio • Jan • Christophe • David • Ben • Christoph • Wout • Yaku • Kavie • Senja • Anal • Jakub • Seifallah • Eleftheria • Yusuke • Nayoung • Yu-Chun • Nils • Selika • Sarina • Han • Manya • Manya • Amrapali • Jonathan • Seongbin • Haki • Jonathan • Douglas Charles • Deanna • Dylan • Bas • Arian • Weijiang • Kartik • Asma • Tessa • Mahtab • Tom • Andre • Vera • Muhammad Raed • Stijn • Valerie • Javad • Alessandro • Yusuf • Neha • Peterl • Jan • Henrique • Jun • Shruthi • Jeroen • Sam • Danica • Soheila • Annie • Mohamed El Kordy • Emre • Jonas • Thomas • Franjo • Rishabh • Marwan • Hong Hai • Christian • Genis • Noemie • Laura • Sarah • Patricia • Guy • Giulia • Lea • Caro • Virginie • Karol • Morena • Valentine • Amber • Francesca • Nathan • Sien • Nele • Francesco • Sofie • Gaelle • Thomas • Jana • Wouter • Lennard • Hallie • Beatriz • Lucă • Yannick • Robert • Marie • Lize • Damon • Domenico • Alaa • Rick • Shu-Ngwa • Yosuke • Nicole • Hikaru • Michiel • Anna • James • Fernando • Natan • Julian • Yashovardhan • Jonas • Jasper • Vitaly • Irene • Arno • Bram • Sarah • Konstantin • Wafae • Gianpiero • Julian • Sang Cheol • Laila • Abdaoui • Preston • Wout • Thais • Onur • Woo Jin • Michiel • Ali • Peter • Maarten • Hasantha • Willi • Mehmet Kutay • Behzat Utku • Rostislav • Kurt • Evelien • Stijn • Roman • Brent • Seger • Erik • Tom • Marleen • Koen • Filip • Yoann • Thomas • Erik • Kristoffel • Mireille • Liesbeth • Ursula • Lamert • Nicholas • Yiming • Kishan • Vincent • Frederik • Dirk • Kevin • Veerle • Alex • Pieter • Aida • Cheng • Fabian • Jesu Kiran • Karen • Sathisha • Nick • Erik • Filip • Chidharth • Hajjin • Jesse • Francesco • Wenqi • Willy • Sander • Stephane • Sulakshna • Victor • Jonas • Hua • Xiaolong • Sandeep • Chenming • Benjamin • Makoto • Giulia • Celine • Daniel • Arthur • Giordano • Carlo • Nathalie • Jannes • Charlo • Frederic • Michiel • Liesbeth • Maxim • Bryan • James • Stephanie • Robbe • Wei2 • Kenny • Annelies • Kushagra Singh • Nishant • Jason • Yishu • Arno • Akshit • Ananya • Yiwei • Katrien • Bas • Thomas • Laura • Alexander • David • Varun • Jasper • Bo • kck • Anh Minh • Almudena • Shahriar • Jelle • Jerome • Luis Alberto • Jalal • Wei-Yu • Marion • Salma • Jonne • Charlotte • Michiel • Stuart • César • Saja • Carmen • Tanmoy • Vincent • Andrew • Reza • Charis • Hisashi • Kazutaka • Ian • Stefan • Hicham • Ju Hyuck • Hana • Theo • Ioannis • Nico • Marcel • Takashi • Aurore • Yann • Felipe Kenji • Igor • Rossa • Jack • Lisa • Hsiao-Lun • Jun-noh • Chen • Samson • Marco • Catalina • Meryem • Daria • Fatma • Amine • Maria Jose • Ihlas • Omid • Milica • Sin Fu • Jafal • Mathijs • Laura • Milad • Stephen • Javad Arian • Thanos • Chelsey • Ege • Safae • Marko • Hojjat • Sander • Sahar • Ahmed • Kanta • Shinya • Christoph • Nancy • Laetitia • Jonathan • Mohmadsadegh • Matthieu • Youngwook • Fawaz • Aladdin • Frans • Ann • Flora • Marion • Bruno • Sander • Delphine • Anthony • Joachim • Ricardo • Boris • Joonyoung • Stephanie • Karan • Lene • Benjamin • Zhengtao • Christoforos • Apostolia • Lucan • Vinod • Gerson • Yuchao • Emre • Asuncion • Santhosh • Nazar • Natasja • Matthias • Sylvia • Thomas • Olivier • Michiel • Bas • Ralf • Maarten • Hannes • Annelies • Ine • Sanne • Daniel • Isaac • Matthias • Kenzo • Laura • Jasper • Tom • Wanda • Ruben • Kyle • Dieter • Lien • Anniek • Emiel • Thomas • Ruben • Miel • Piete • Yan • Daniel • Mike • Yao • Ruben • Karien • Wim • Hubert • Yoann • Geert • Peter • Reginald • Jan • Dieter • Davinia • Martin • Christophe • Joni • Pascal • Kim • Stefaan • Pieter • Joël • Jan • Veerle • Werner • Jan • Anthony • Tom • Mario • Marlize • Gert • n • David • Nathalie • Arnaud • Stephane • Ben • Mohit • Bart • Alan • Bjorn • Rik • Jesse • Yannick • Piet • Kaushik • Niek • Wouter • Ine • Bart • Frank • Rik • Max • Yannick • Peter • Nik • Hemant Kumar • Willemien • Ali • Joshua • Werner • Christophe • Fran • Arnaud • Dongyang • Shirotori • Lukas • Hanna • Frederik • Geoffroy • Eric • David • Laura • Laura • Bart • Cedric • Xuelong • Gianluca • Ruben • Emmeric • Jens • Bram • Jeroen • Els • Eric • Wim • Patrick • Gijsbert • Marijn • Ellen • Simon • Alessandra • T Yusuke • Tom • Thomas • Thys • Mathieu • Huguette • Xiaohua • Liesbeth • Peter • Gilles • Stephen • Andrea • Xuening • Raees • Ilse • Jared • Olivier • Mathias • Francois • Robbe • Aurentje • Katrien • Philippe • Saartje • Stefan • Louis • Thomas • Jef • Jianhadur • Georgios • Elisabeth • Vladimir • Francesca • Andriy • Vincent • Rene • Douglas • Ziad • Michael • Gosia • Kathleen • Ken • Gavin • Paulius • Ainhoa • Amir-Hossein • Jonas • Sebastian • Pierre • Lucien • Karen • Pishko • Aftab • Jose Ignacio • Andrew • H

## We are a global team of over 6,500 talented employees from more than 100 nationalities.

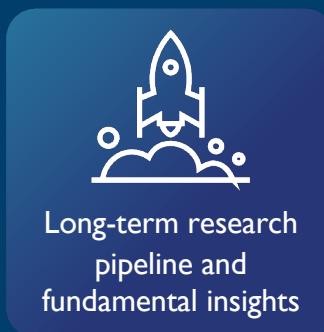
ebing • Kevin • Yuan • Paul • Jochem • Shenqi • Marnix • Jerald • Roy • Lei • Swathi • Cina • Mathijs • Murat • Alexandra • Ivo • Igor • Gabriela • Lucy • Leendert • Wenzhe • Saptarshi • Yannick • Roberto • Paul • Simone • Justin • Ulzhan • Daisuke • Sébastien • Ardo • Raj • Karolina • Rajendra Kumar • Maaike • Mehmet Bilgehan • Cornelie • Daniel • Valdy • Yunqi • Julien • Dogukan • Bas • Abhishek • Ward • Achintya • Istvan • Tamara • Yunfan • Jelle • Yang • Jan • Prafulla • Cassie • Linda • Bevita • Shuchi • Tobias • Lies • Kim • Erwin • Martine • Pieter • Utku • Jef • Jacob • Seunghak • Il Gyo • Karina • Hyukyun • Anoop • Warre • Julian • Shiqi • Hans • Jeroen • Swapnil • Tzu-Heng • Fengben • Maxime • Jarich • Gijs • Robert • Peter • Esma • Anurag • Hamed • Ashish • Pedrhis • Florian • Catherine • Steven • Arantxa • Shashikant • Chris • Rob • Antonietta • Tekin • David • Karen • Lorenza • Saransh • Bo • Neam • Mariachiara • Nick • Robbe • Wei-Hua • Benedikt • Muhammad Usama • Anadi • Soulyman • Mahsa • Jin San • Marc • Te • Junren • Junren • Baris • Lukas • Harish • Sriram • Kia Woon • Aislan • Wouter • Kazuki • Kaiwen • Jesper • Chia-Wei • Jiseok • Ivania • Andrew • Li • Hyun-Cheol • Jakob • Jordi • Bibi Zhara • Nicolas • Bas • Joost • Ali • Maarten • Hans • Jiwon • Annol • Johashika • Lauren • Neha • Martin • Brecht • Toon • Birte • Eveline • Thomas • Paola • Eduardo • Marika • Ching • Carlos • Thomas • Jacob • Ali • Andreas • Tianzi • Rufi • Esme • Andrei • Jori • Ewelina • Akane • Patricia • Pieter • Steven • Dara • Daigo • Saeed • Il Raja • Ulysse • Carina • Hannah • Douae • Maurice • Zeno • Felitsa • Eleni • Carla • Lealia • Laxman kumar • Pieter-Jan • Giselle • Abhinay • Priya • Yilun • Bart • Erfan • Yaren • Liam • Andrew • Greeshma • Jeroen • Ye • Sung Woo • Pankaj • Arantxa • Muhammed • Raj • Isaak • Aasiya Bano Abdul Rauf • Godfred • Aruzhan • Frederik • Hans • Jacobus • Arne • Lorin • Peter • Ying-Chun • Binghua • Tien Dat • Evangelos • Diego • Sacid • Gabriel • Gabriel • Blake • Matthias • Yusuf • Gaurav • Sahan • Serkan • Liwang • Mehdi • Tijs • Ward • Hareen • Gargya • Stijn • Julie • Krishna • Chunzhuo • Jonathan • Chao • Zhongtiao • Tibo • Lotte • Morgane • Marco • Gianpiero • Ernest • Meng • Jorik • Xiangyu • Yuqing • Siyuan • Anuj • Zhanwei • Martijn • Ilaria • Oreste • Gaoyuan • Jan • Xinrui • Marieke • Aditya • Hui • Aslihan • Samer • Anupam • Kamal • Chen • Ansar • Aarti • Martin • Vivek • Deepanjan • Tom • Shanxing • Mathijs • Tomas • Simone • Marco • Fabio • Viraj • Emad • Robbe • Gautam • Raphael • Federico • Arturo • Vic • Simon • Elise • Veronique • Johan • Karlien • Philippe • Maarten • Elise • Tim • Gerald • Kristof • Dieter • Thomas • Wendy • Ann • Bram • Wim • Simon • Marc • Ruben • Peter • Stefanie • Wouter • Ruben • Chris • Pieter • Dorien • Hadewijch • Steven • Matjason • Tom • Ben • Lars • Devesh • Aleksandra • Joseph Daniel • Lars • Andreia • Claire • Jay • David • Kwanyong • Tom • Mustafa • Merve • Hironori • Reze • Rene • Kanksha • Ivan • Adnan • Eslam • Sara • Carlos • Cagatay • Afzaal • Luis • Brendan • Yuta • Fuphan • Hussein • Filip • Constantine • Mike • Armand • Arash • Veronica Juliana • Mauricio • Ahmed • Sofia • Sara • Guido • Esmeralda • Thiago • Joonseok • Seunghwan • Pelin • Ludwika • Mihaela • Sonja • Pieter • Robert • Michael • Quentin • Ariz • Ata • Nikoleta • Ozgur • Silviu • Enrique • Sara • Corinna • Bryan-Elliott • Sanshiro • Rob • Dennis • Xhulio • Diogo • Antonio • Morteza • Leandro • Mohammadreza • Mohammad • Ioulia • Wen-Fu • Laura • Thayheng • Carlos • Gebrie • Alexandros • Tofistiane • Jef • Daniël • Lieven • Danielle • Rik • Vincent • Piet • Goedele • Serge • Bart • Erwin • Kristof • Albert • Marc • André • Ingrid • Luc • Nausika • Geert • Hilde • Hendrik • Eric • Greet • Steven • Dirk • Erwin • Wim • Johan • Wim • Tom • Philip • MaSabine • Bart • Martine • Steven • Johan • Guy • Filip • Jan • Johnny • Dries • Luc • Geert • Johan • Joost • Liesbet • Sigrid • Kristof • Gunther • Rudy • Ilse • Hans • Patrick • Monique • Benny • Paul • Dominique • Wendy • Hans • Luc • Koen • Bart • Nadine • Rita • Ann • Tom • Marc • Andre • Anne • Sara • Ann • Beatrijs • Christa • Eli • Paul • Jan • Piet • Franciska • Myriam • Nadine • Annouck • Peter • Geert • Ingrid • Dirk • Jan • Luc • Philip • Erik • Gregor • Joost • Hans • Paul • Nadia • Veronique • Johan • Nanc



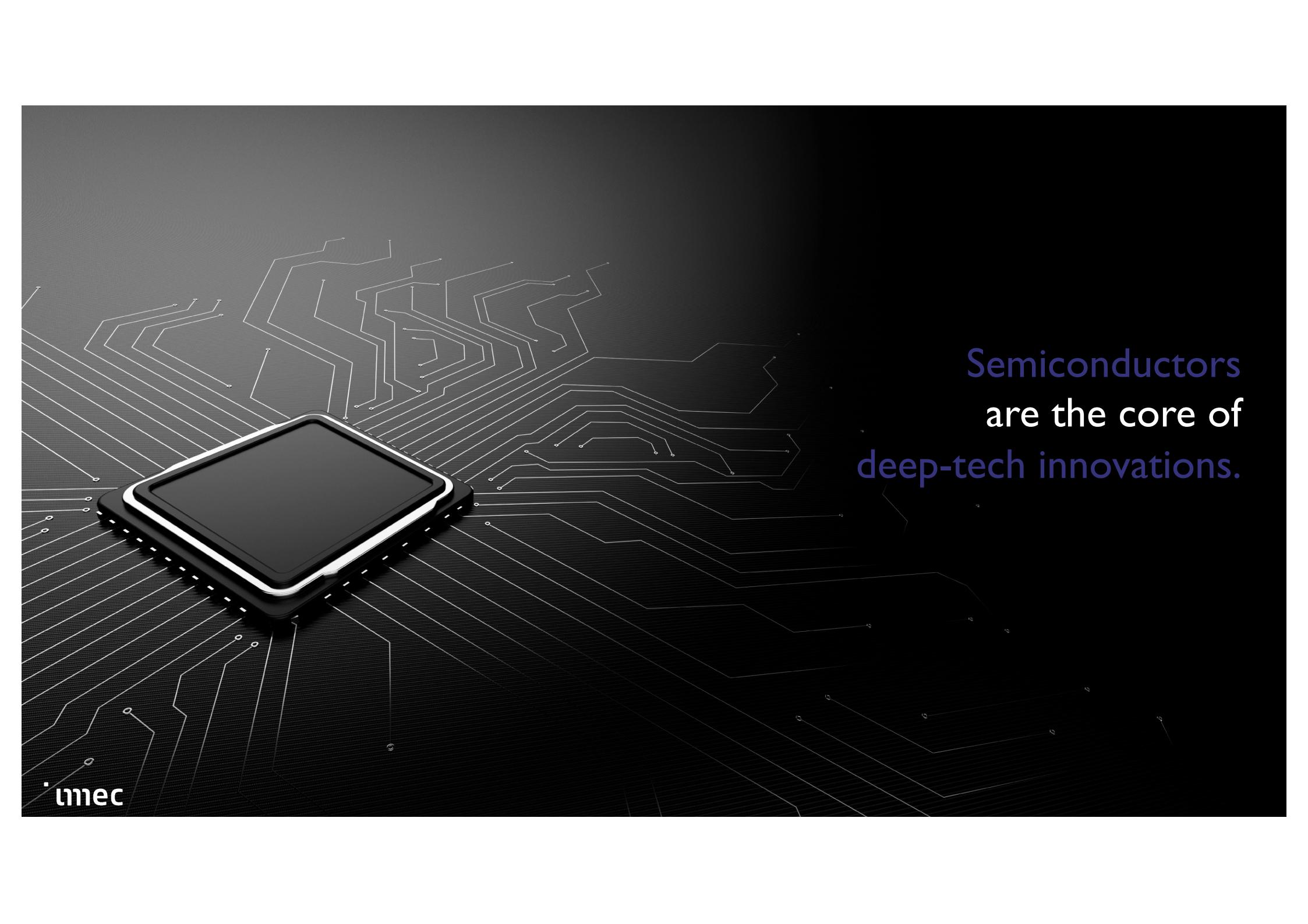
With a global ecosystem of over 600 partners,  
that work together in an open innovation model...



# Our strong connection with the academic world nurtures our innovative excellence.



Together, we develop new concepts and leading-edge technology



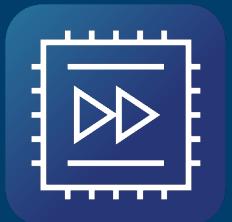
Semiconductors  
are the core of  
deep-tech innovations.

imec

# Computing power needs are exploding.



Data source: Sevilla et al. <https://doi.org/10.48550/arXiv.2202.05924>



increased  
performance



increased  
complexity



reduced  
cost

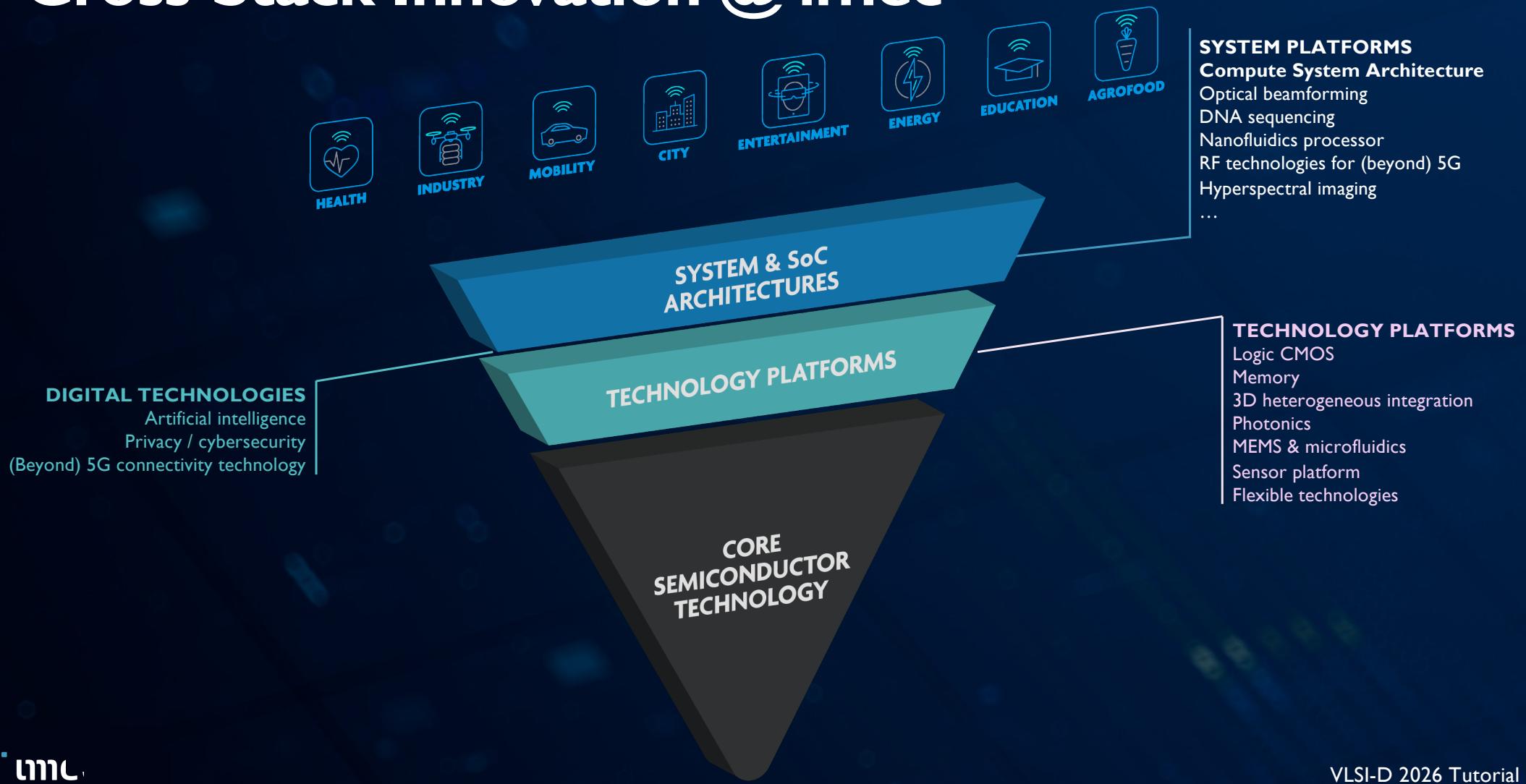


reduced  
power

To handle growing amounts of data,  
we need full-stack innovation



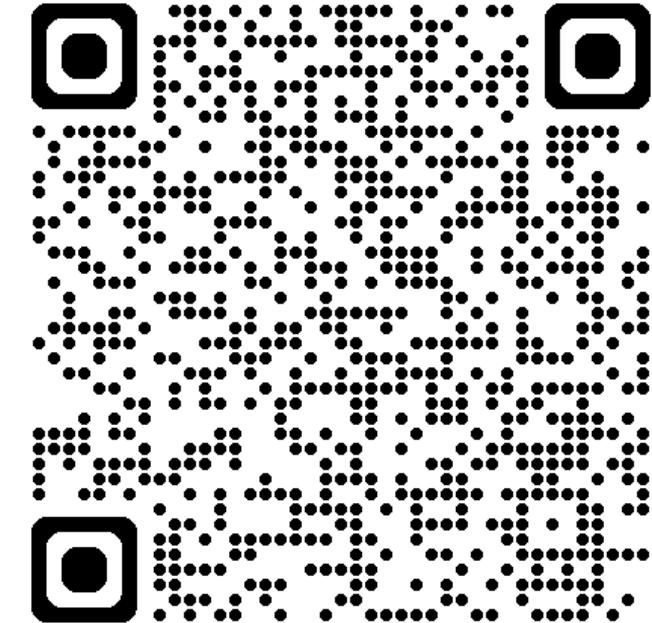
# Cross-Stack Innovation @ imec



# Primary objective of this tutorial

# Agenda

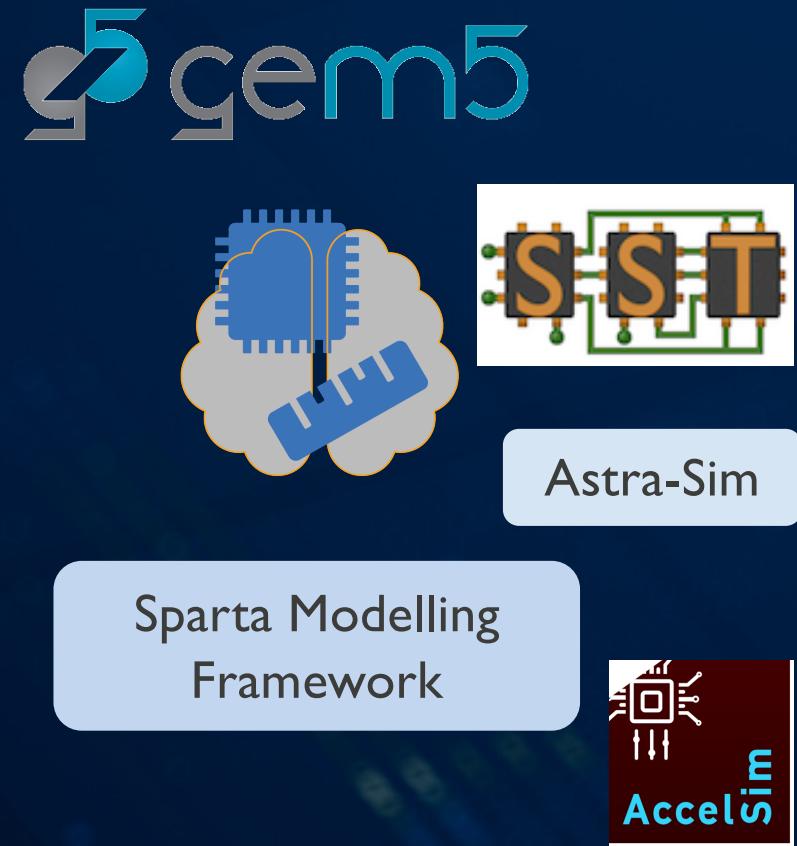
Topic	Timeline
Introduction to System Simulation	14:00-14:45
Large scale analytical system simulation	14:45-15:30
Break	15:30-16:00
Exploring the scaling of LLM training with Packet-level Simulation using SST	16:00-16:30
Cross-architecture x Multi-fidelity simulations	16:30-17:30



Tutorial Documentation

# System-Level Simulators

- Modern systems are increasingly complex
  - Requires accurate and scalable models to evaluate hardware-software interactions.
- Physical prototyping is expensive, time-consuming, and infeasible during early design stages.
- Simulators enable rapid exploration, performance analysis, and optimization of system architectures



<https://www.gem5.org/>

<https://sst-simulator.org/>

<https://sparcians.github.io/map/index.html> <https://accel-sim.github.io/>

<https://scalesim-project.github.io/> <https://astra-sim.github.io/>

# Performance Modelling for Large Scale System

## Role of Simulation in Hardware-System Co-Design

- Create an abstraction to capture the important aspects affecting performance at a system level.
  - Depends highly on the question we are trying to answer
- Use **system-level simulators** to model components (CPUs, memory, interconnects) and their interactions.
- Select/combine tools like **gem5** (detailed architectural modelling) and **SST** (scalable simulations) to analyse real-world scenarios
  - In most cases, a combination of multiple simulation tools are required to come to an “actionable” decision
- Use **representative benchmarks** and **design of experiments (DoE)** to ensure accuracy and scalability and **validation of the results**

# Multi-fidelity system architecture simulation

## Features and Limitations

- Typically allows modelling of a “hardware” component at high level of abstraction
  - Usually much less details than RTL
- Performance estimation and bottleneck analysis can be done
  - Eg: *Pipeline viewer to see stalls*
- Serves well to do “design space exploration” and answer “what-if” question(s)
  - Eg: *What if the processor clock could run at 30Ghz?*

How does “area”, “energy” change with “technology”?

**Direct estimation is not feasible.**  
Needs additional tools.

**It does not say if it is feasible to design a CPU with 30Ghz clock.**  
RTL validation and possible full mapping to technology library would be necessary.

# Design of Experiments (DoE)

## Choosing “Figure of Merit”

- Choosing the “figure of merit” is a very critical aspect in the DoE
- If a simulator cannot be used to report or derive an “figure of merit”
  - Choose a different simulator
  - Extend it or combine it with some other simulator → This is mostly what happens in practice!
- Often a combination of multiple metrics might be interesting
  - Eg: *Area-Delay Product, Throughput/mm<sup>2</sup>*.



# Design of Experiments (DoE)

Choosing “Figure of Merit” – In the context of LLMs

- Systems built for machine learning are ubiquitous now
  - Powering everything from data centers to edge devices.
- Operational costs and energy efficiency is a well accepted FoM
  - TOPs/W → Measure of “energy efficiency”
  - TOPs/(W mm<sup>2</sup>) → Ties performance, power, and area together
- Environmental impact is become an important FoM
  - Carbon Emissions related to manufacturing of the chips
  - <https://netzero.imec-int.com/>



# Design of Experiments (DoE)

## Benchmarks for Evaluation

- Choice of benchmarks is as important as choice of “FoM”
  - *Eg: If you want to evaluate a multi-node system, MPI workloads might be interesting*
- It always helps to use “well-accepted” benchmarks to compare across different systems of similar kind
  - *Eg: SPEC CPU® 2017, LINPACK benchmarks, etc.*

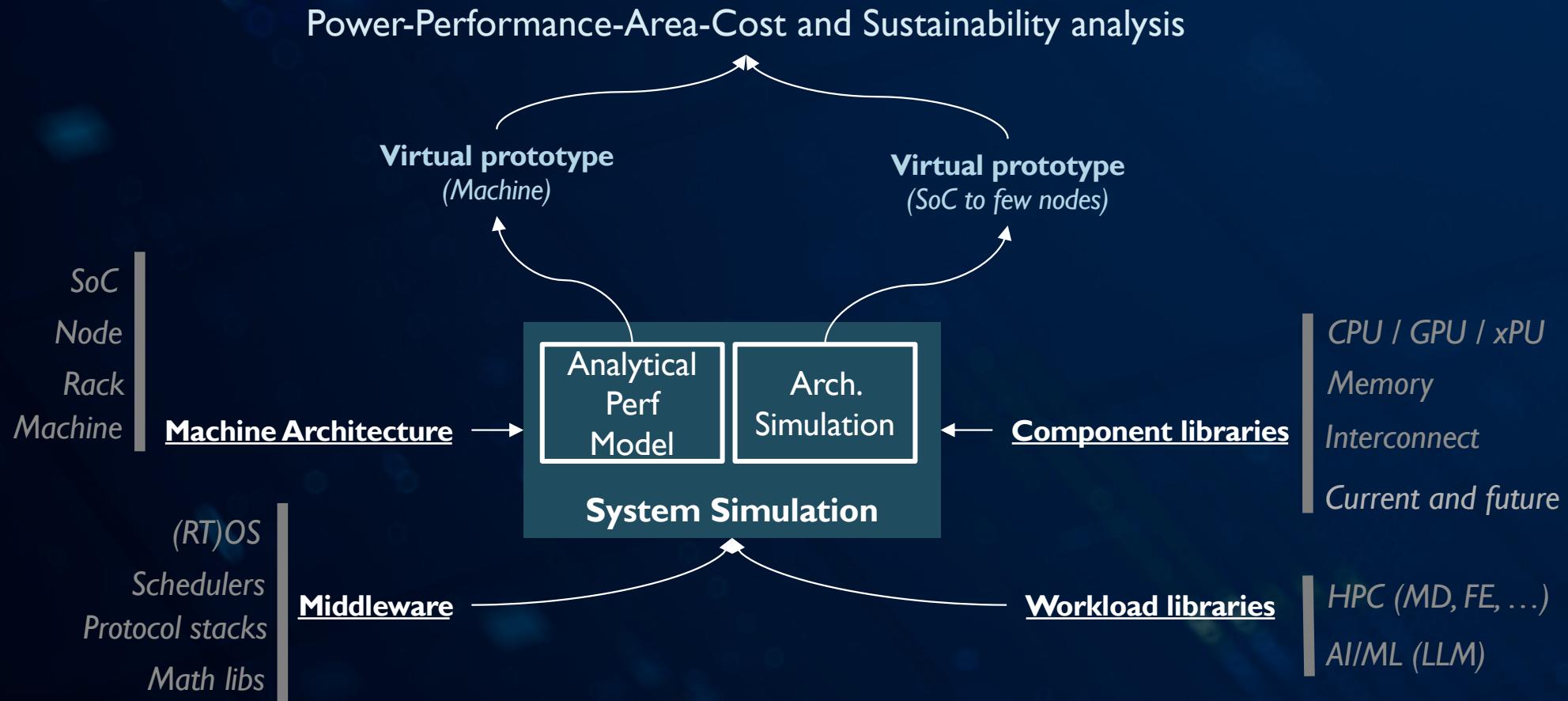
**Benchmarks need to be represented/compiled in a way that the simulator/system can ingest!**

# Multi-fidelity system simulation



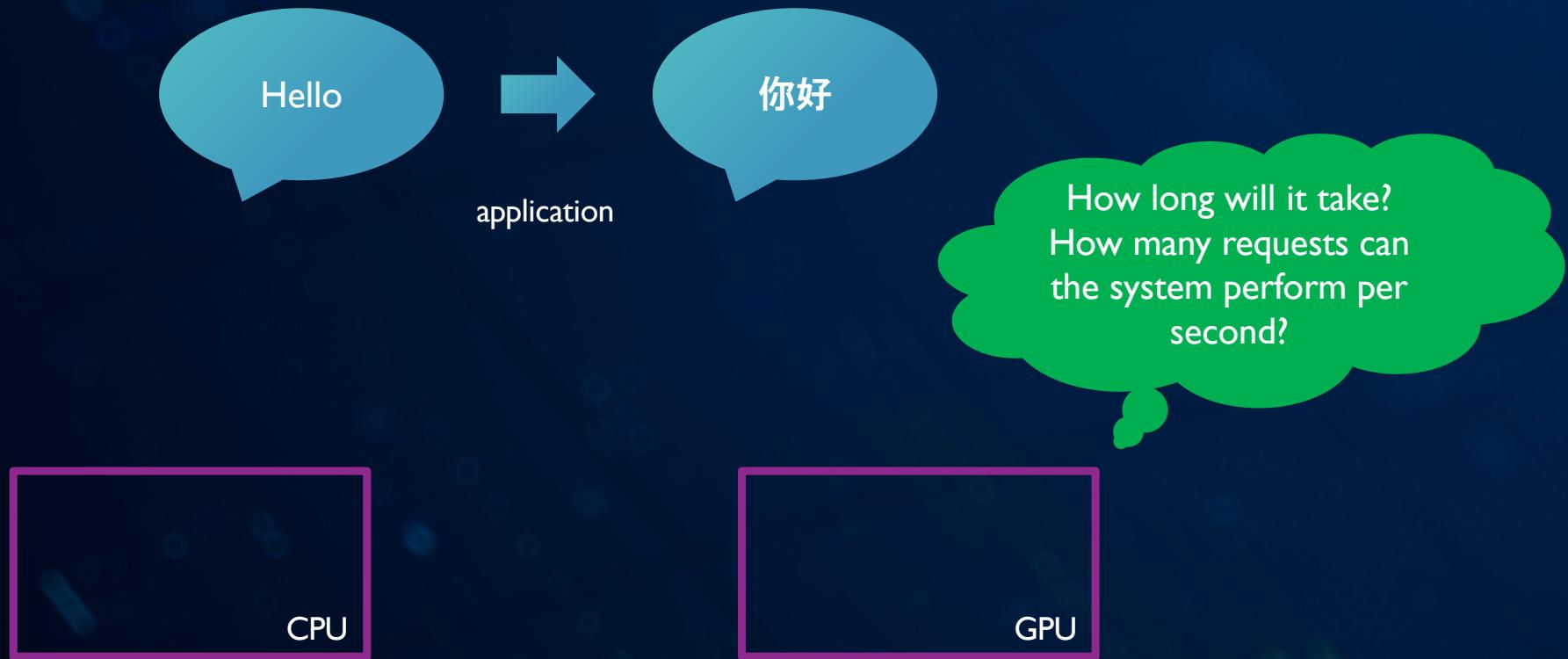
Virtual prototyping to de-risk early design stages

# Virtual prototyping using system simulation

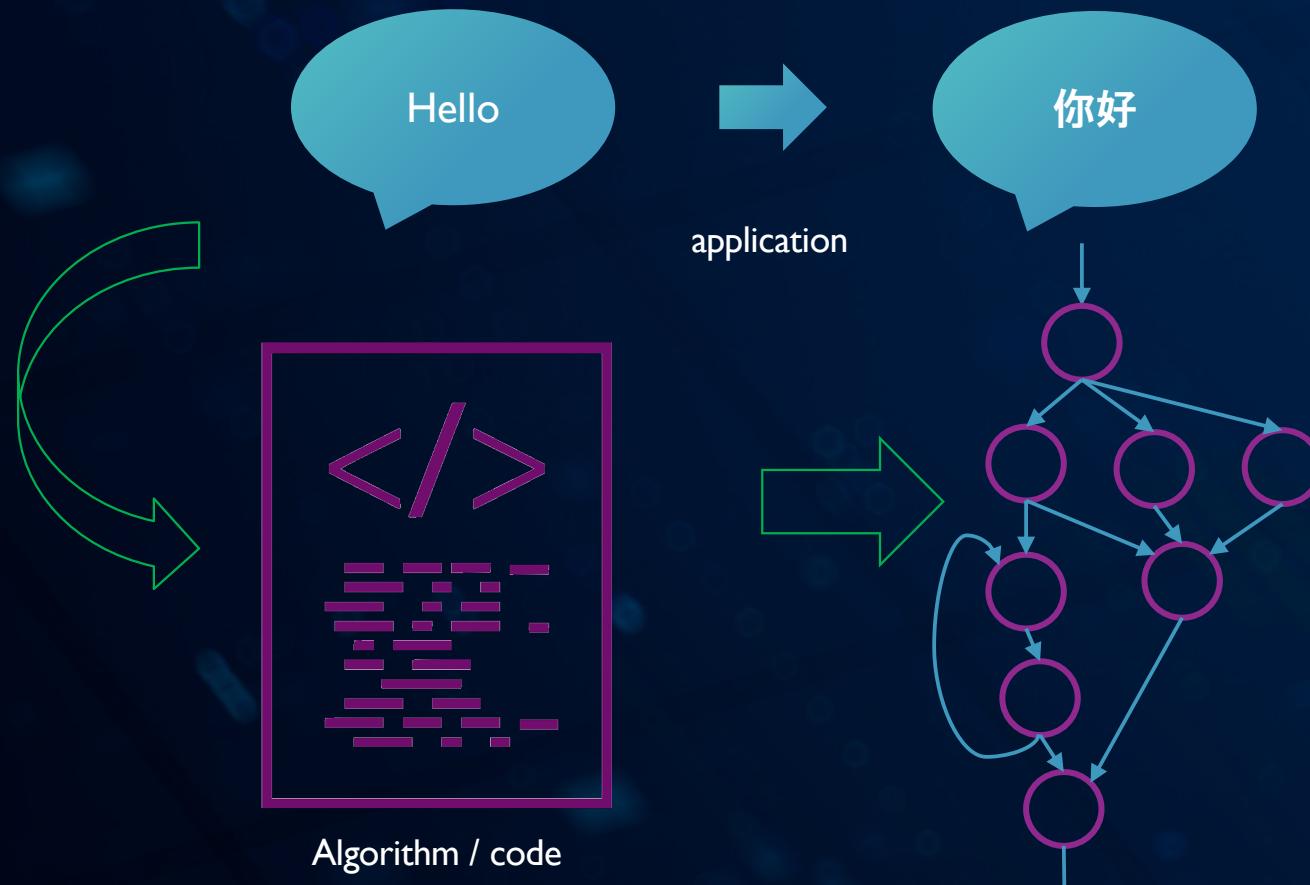


# What is Analytical Performance Modeling ?

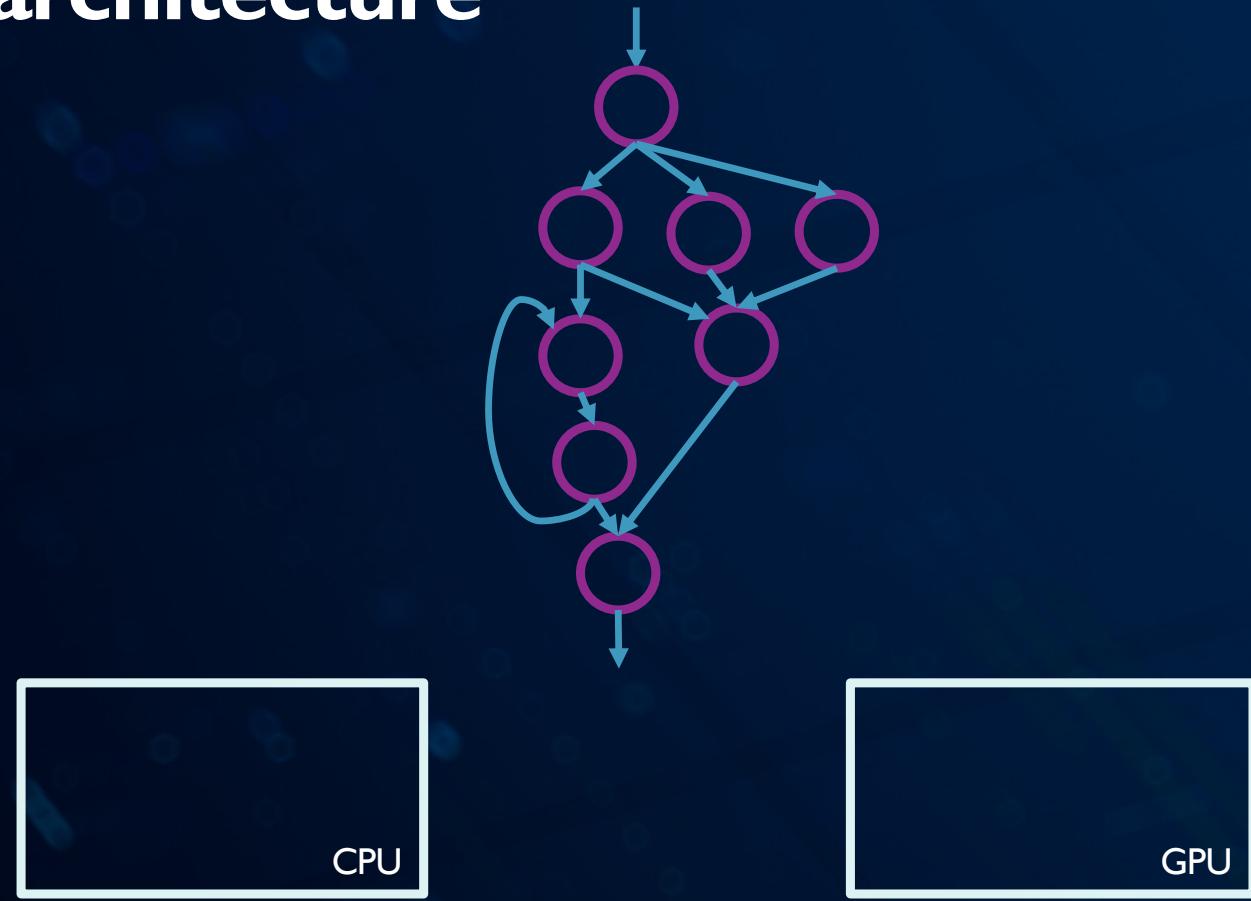
# Performance modeling of an application running on certain hardware



# Application - algorithm - dependence graph

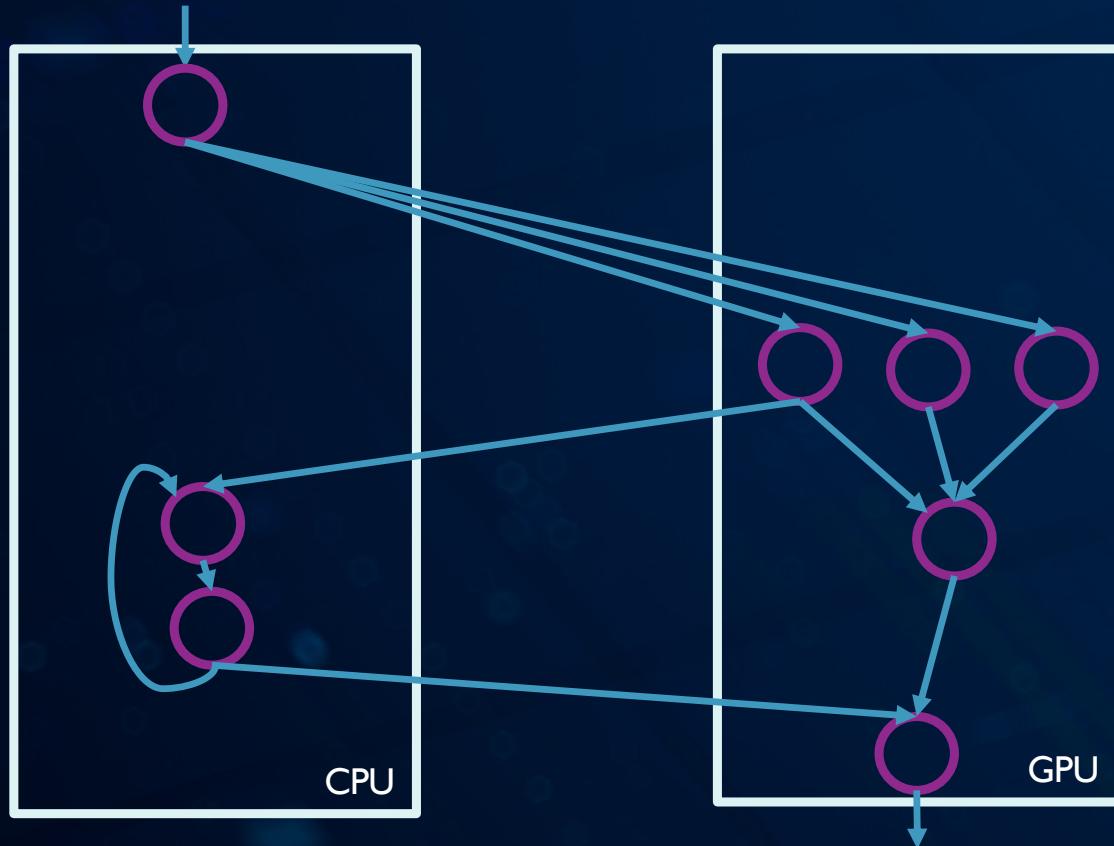


# System architecture



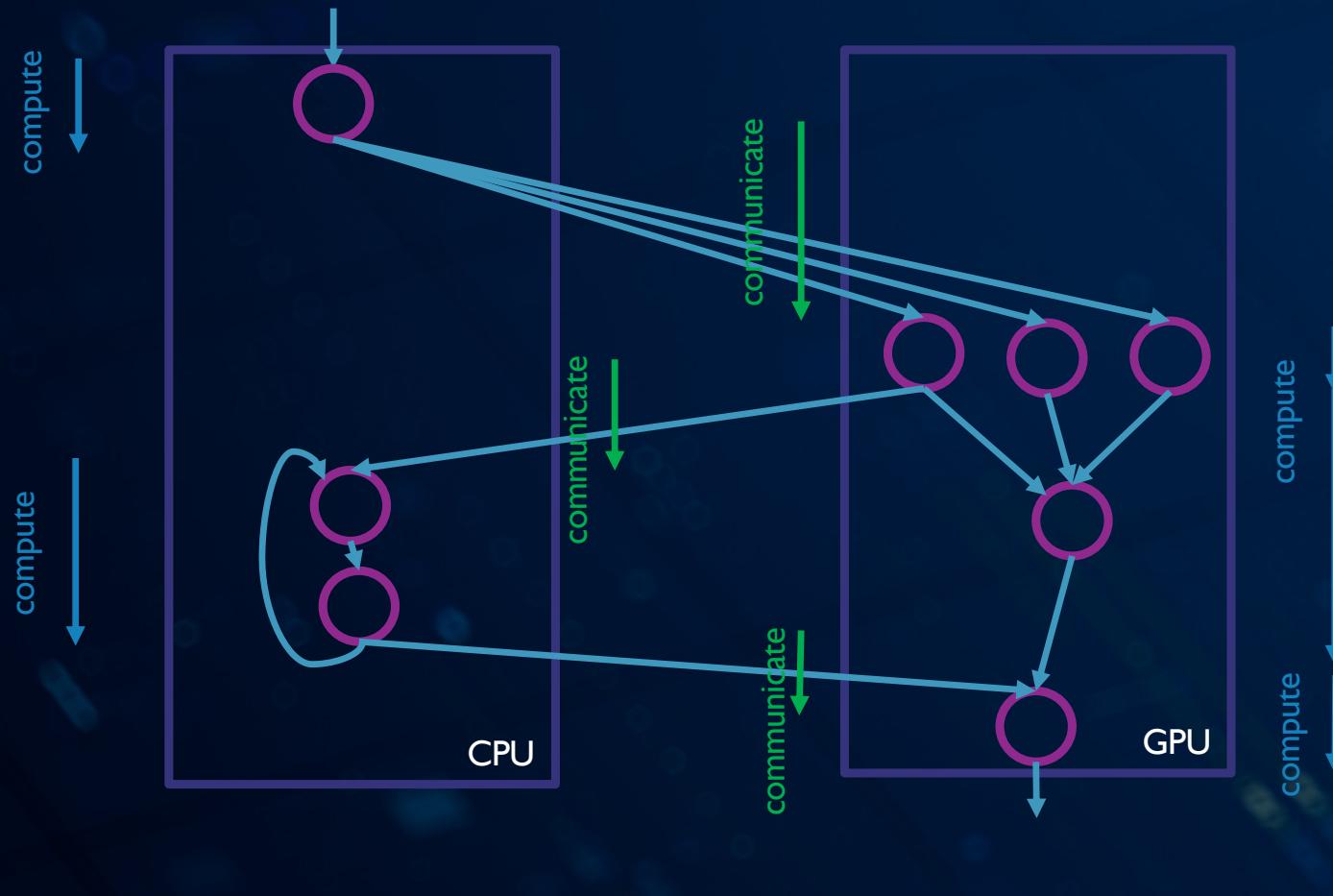
# Mapping application graph

On system architecture



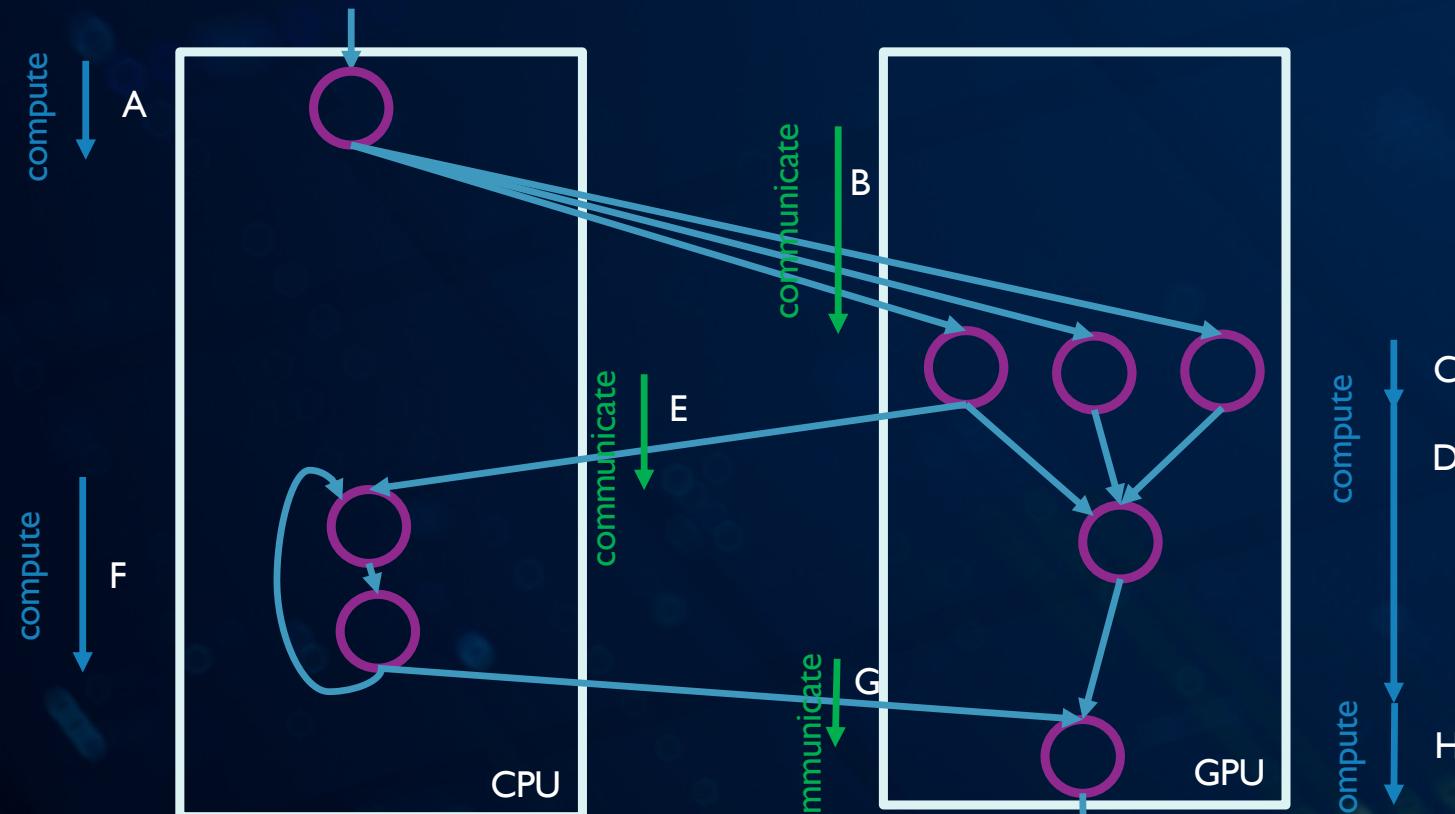
# Performance

## Execution time



# Performance

## Execution time



Time: compute time and communication time of longest path

$$A + B + C + \text{Max}(E+F+G, D) + H$$

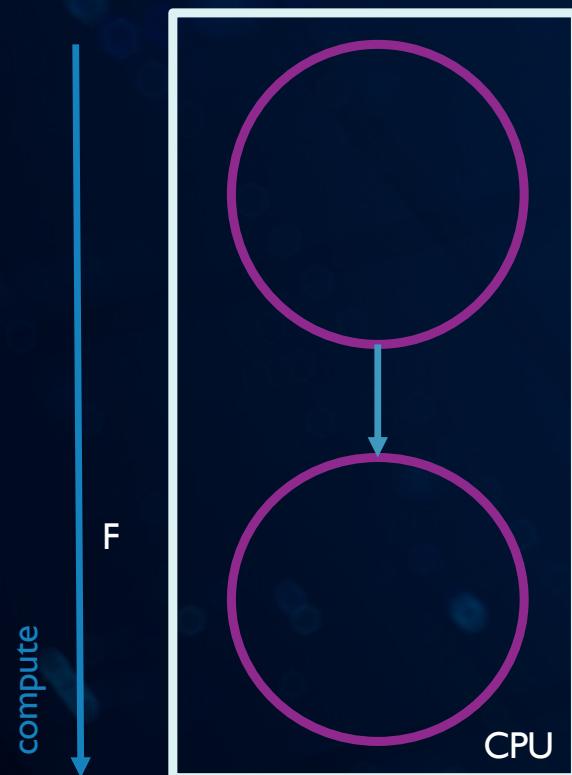
# Zooming in



Time

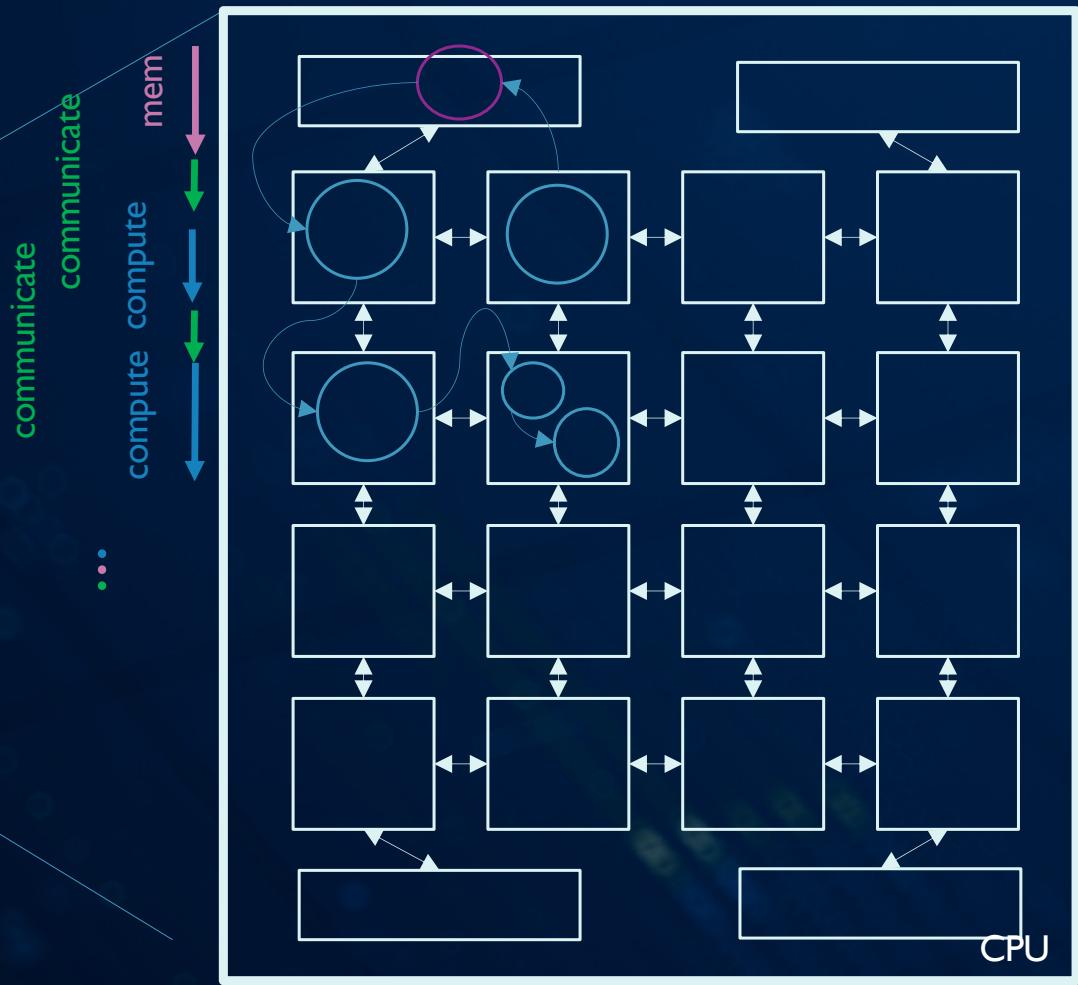
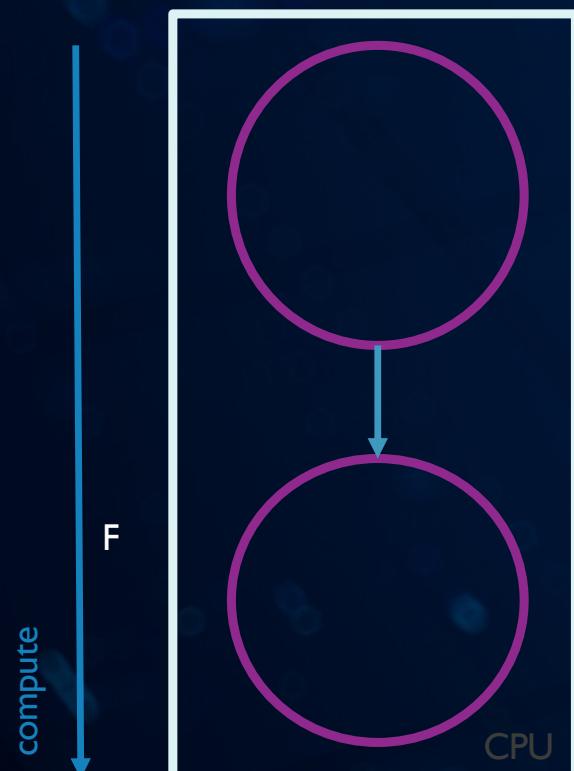
imec

# Zooming in



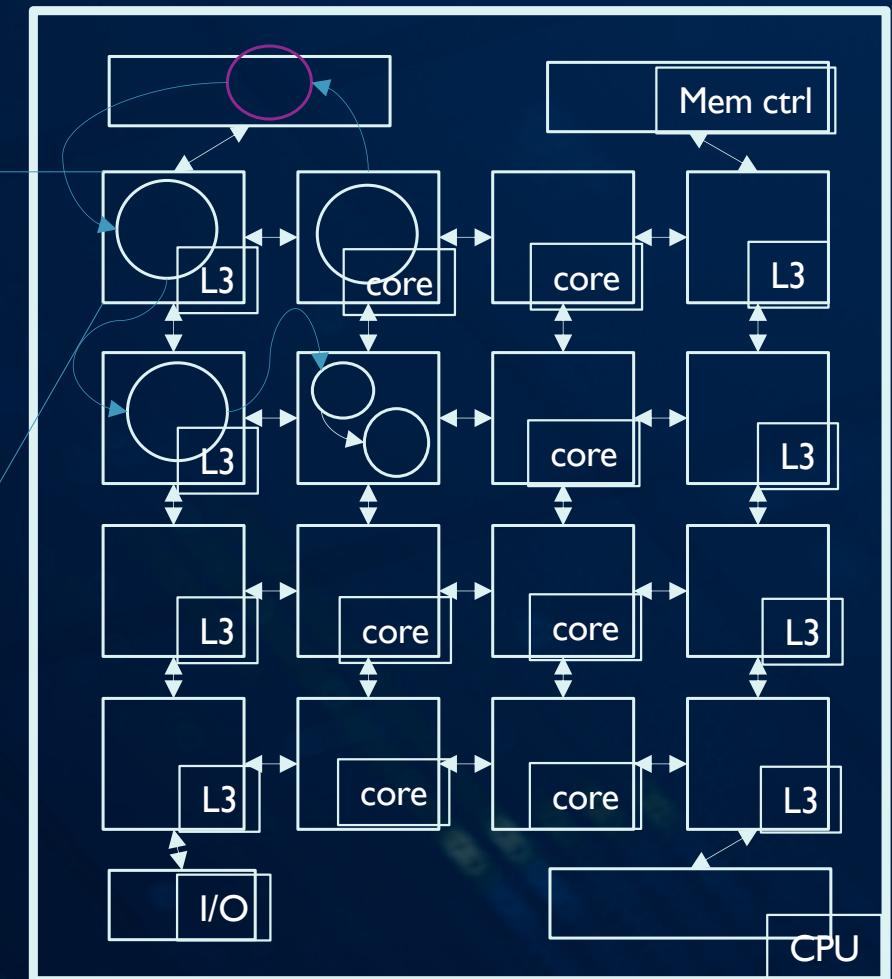
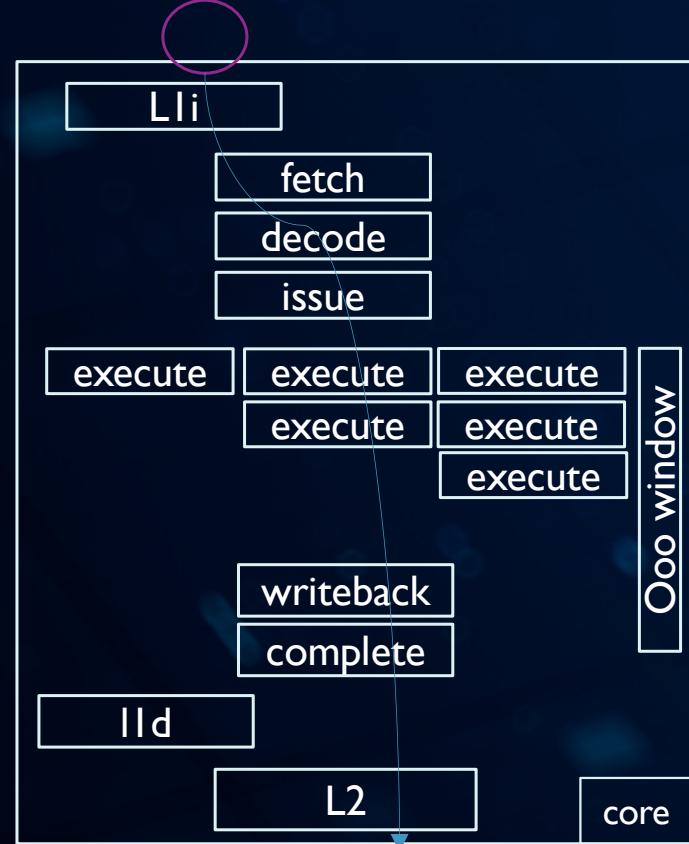
# Zooming in

## Same exercise - mapping on SoC architecture



# Zooming in more

Same exercise again - mapping on microarchitecture



# Analytical performance modeling: goals

- Cover as much of the HW abstraction stack
- Expose abstract knobs at each level in the stack,
  - Accuracy; Error function; Model parallelism; Scheduling; Network bandwidth and topology; Hybrid on-chip memory; Vector unit size
- Model how knobs affect application performance (energy, cost, ...) on the system
  - “Easy” analytical formula, quick to evaluate
  - Prefer build up from mechanics (how things work) vs fitting to experimental data

Problem
Algorithm
Program/Language
Runtime System (VM, OS, MM)
ISA (Architecture)
Microarchitecture
Logic
Devices
Electrons

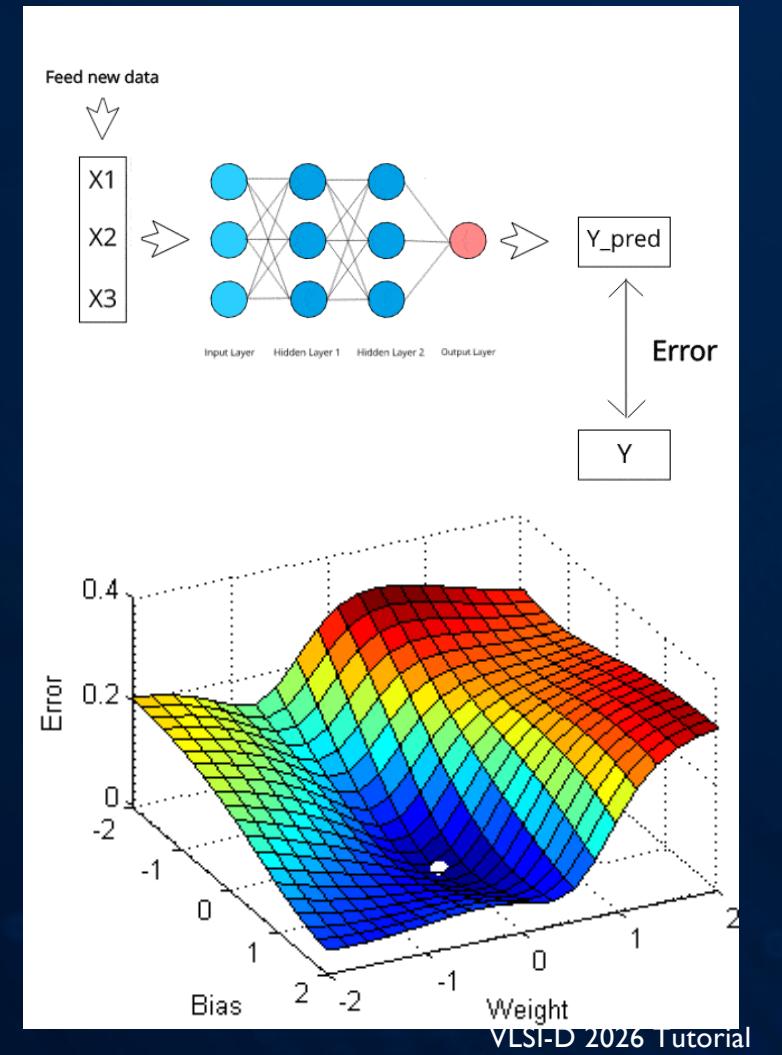
# LLM Training

# Steps in Neural Network Training

- Load input data (batch)
- Forward pass = calculate activations
- Loss calculation = compare activations with ground truth
- Backward pass = calculate gradients
- Weight updates

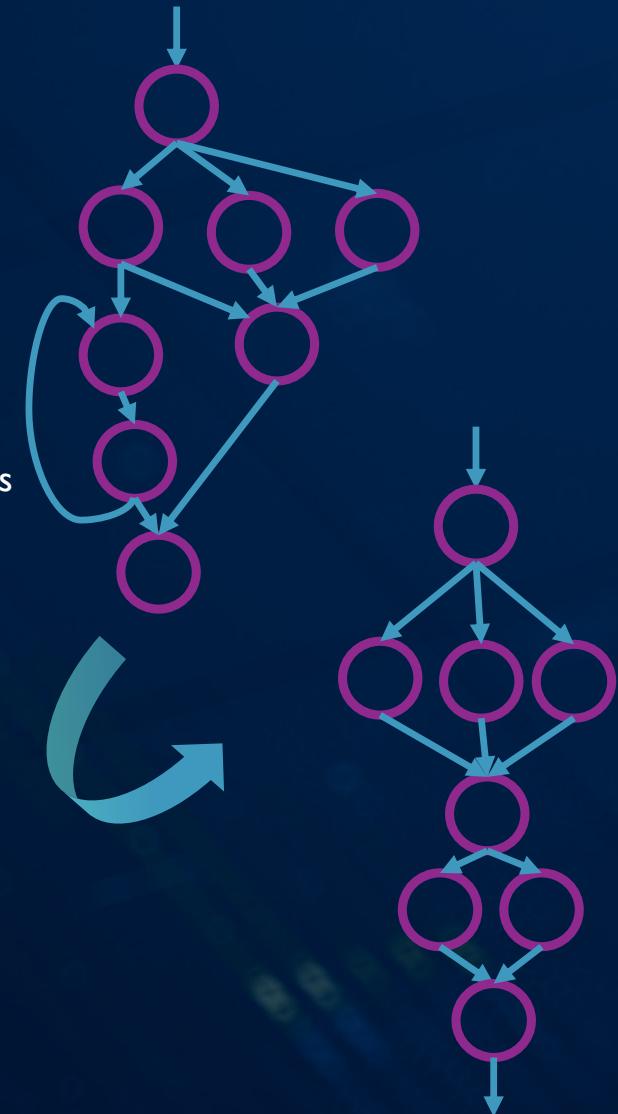
## Memory Requirements

- Residual states
  - Activations
  - Temp buffers
  - Fragmented memory
- Model states
  - Weights
  - Gradients
  - Optimizer states

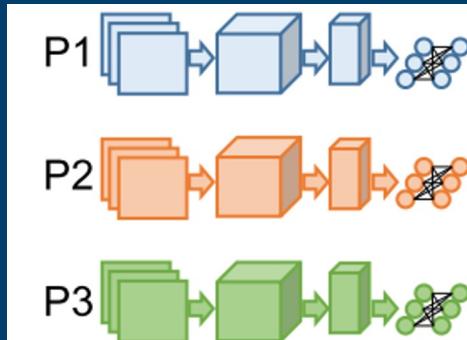


# Exploiting regularity in LLM

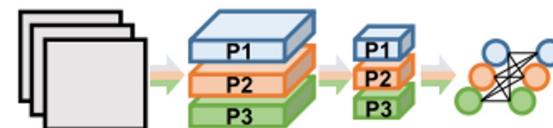
- Dependency graph is known up front by neural network architecture
  - Each step (Data loading, Forward pass, loss calculation, backward pass, weight updates) has known sizes to communicate and known amount of required operations
  - Compute time  $\sim$  operations required / effective throughput
  - Communication time  $\sim$  latency + data / effective bandwidth
- Significant impact of mapping
  - how much computation can be done in parallel
  - how much communication is required
  - how much memory is required where
- Assumption on using homogeneous compute node



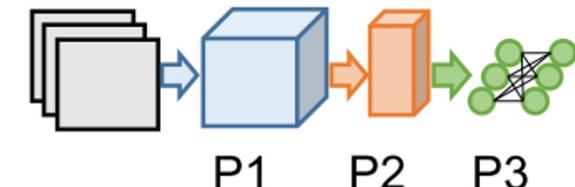
# Mapping Knobs: Data, Tensor (model), & Pipeline Parallelism



Data Parallelism:  
Distribute input samples.



Model Parallelism:  
Distribute network.



Pipeline Parallelism:  
Partition by layer.



3D parallelism:

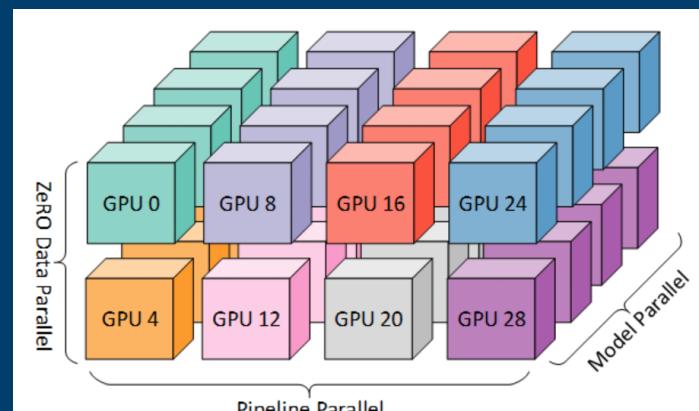


Figure 2: Mapping of workers in Figure 1 to GPUs on a system with eight nodes, each with four GPUs.  
Coloring denotes GPUs on the same node.

arXiv:1802.09941

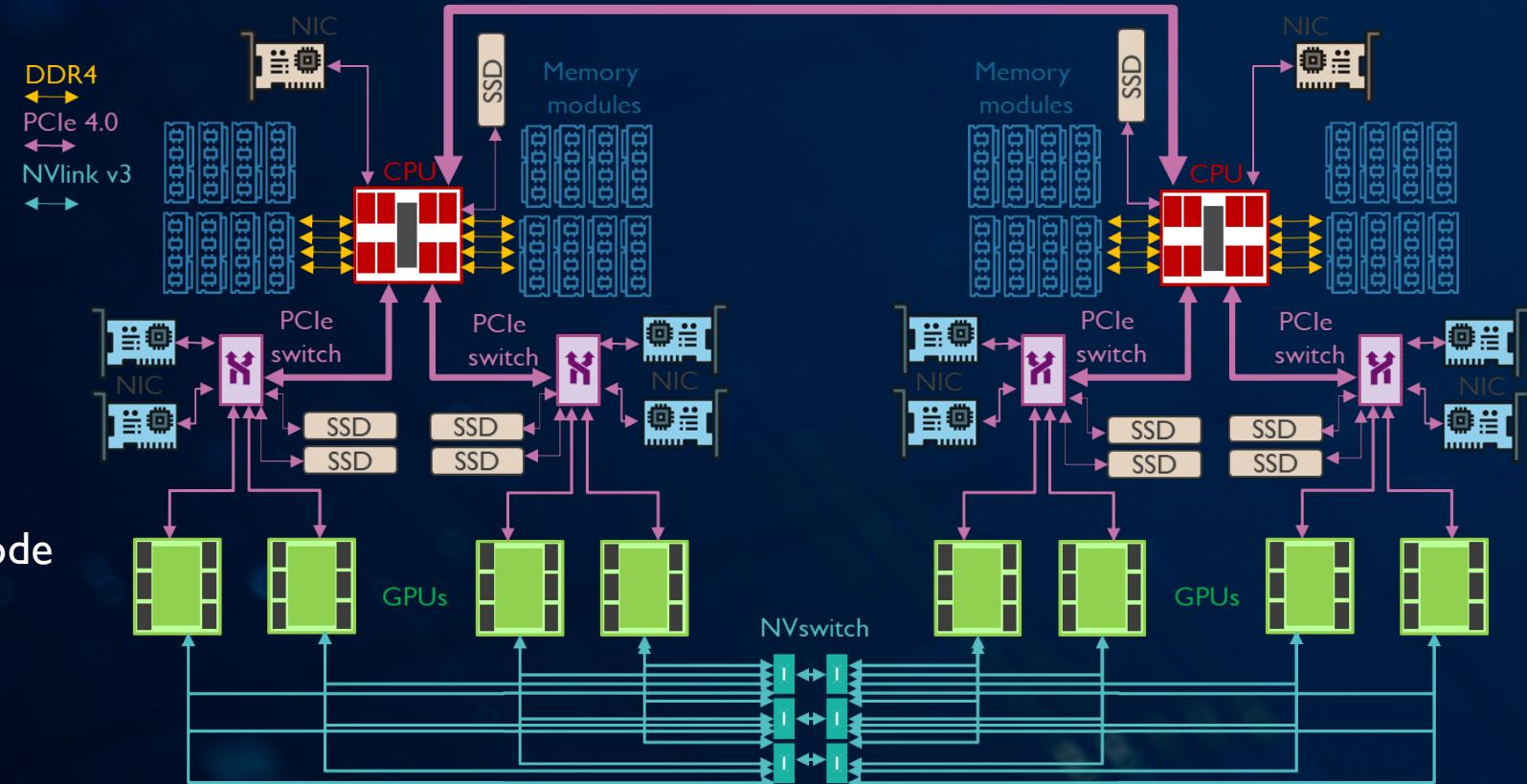
imec

VLSI-D 2026 Tutorial

# System Architecture Knobs

Abstracting the large scale system is a crucial parameter

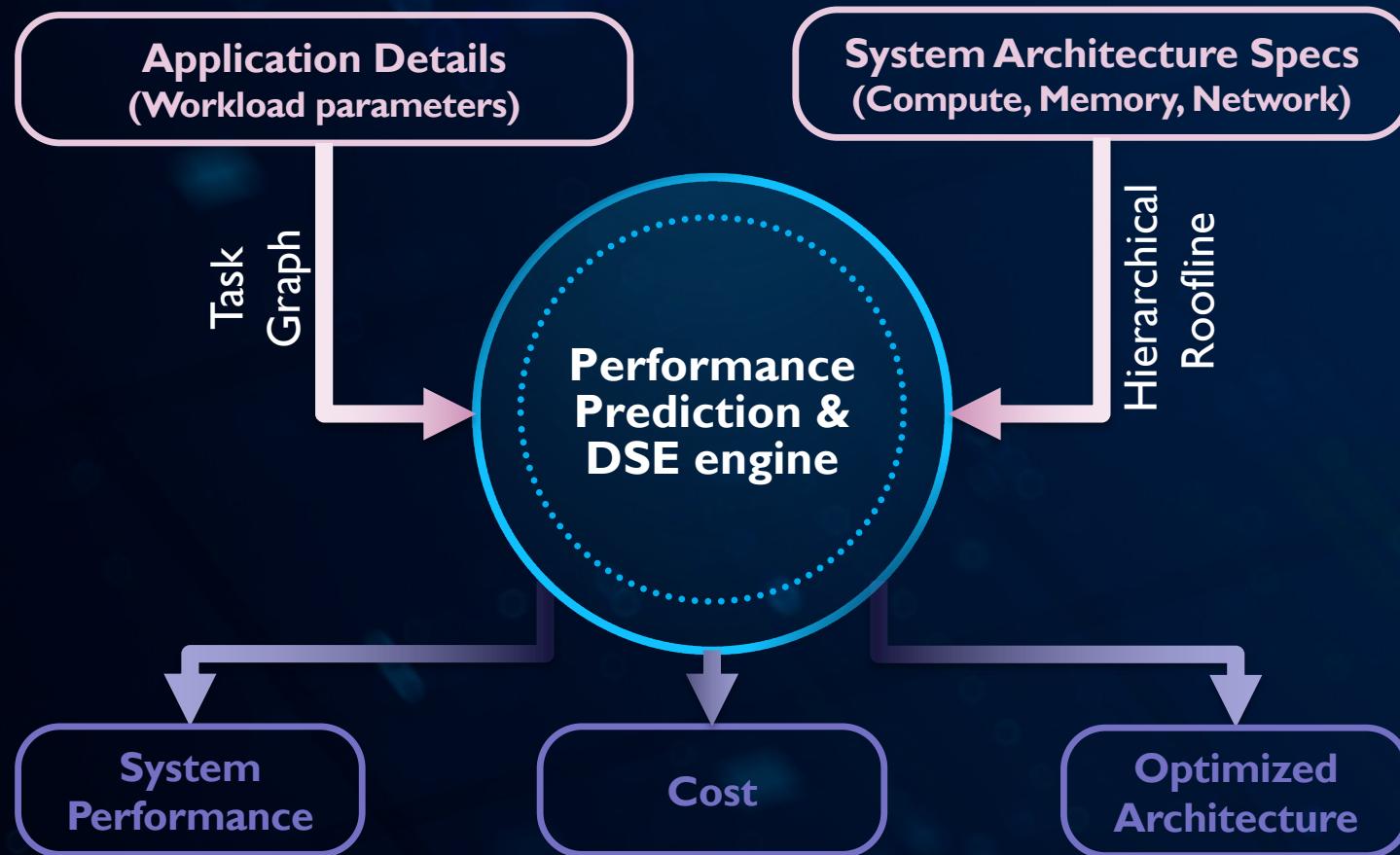
- Inter-node network
  - Total nodes
  - Latencies
  - Bandwidths
  - Topologies



- Intra-node network
  - Accelerators per node
  - Latencies
  - Bandwidths
  - Topologies

# Analytical Architecture Analysis Tool: *imec.kelis*

Fast, Accurate and Generalizable



## Supported Workloads

- LLM Training
- LLM Inference

## Supported Architectures

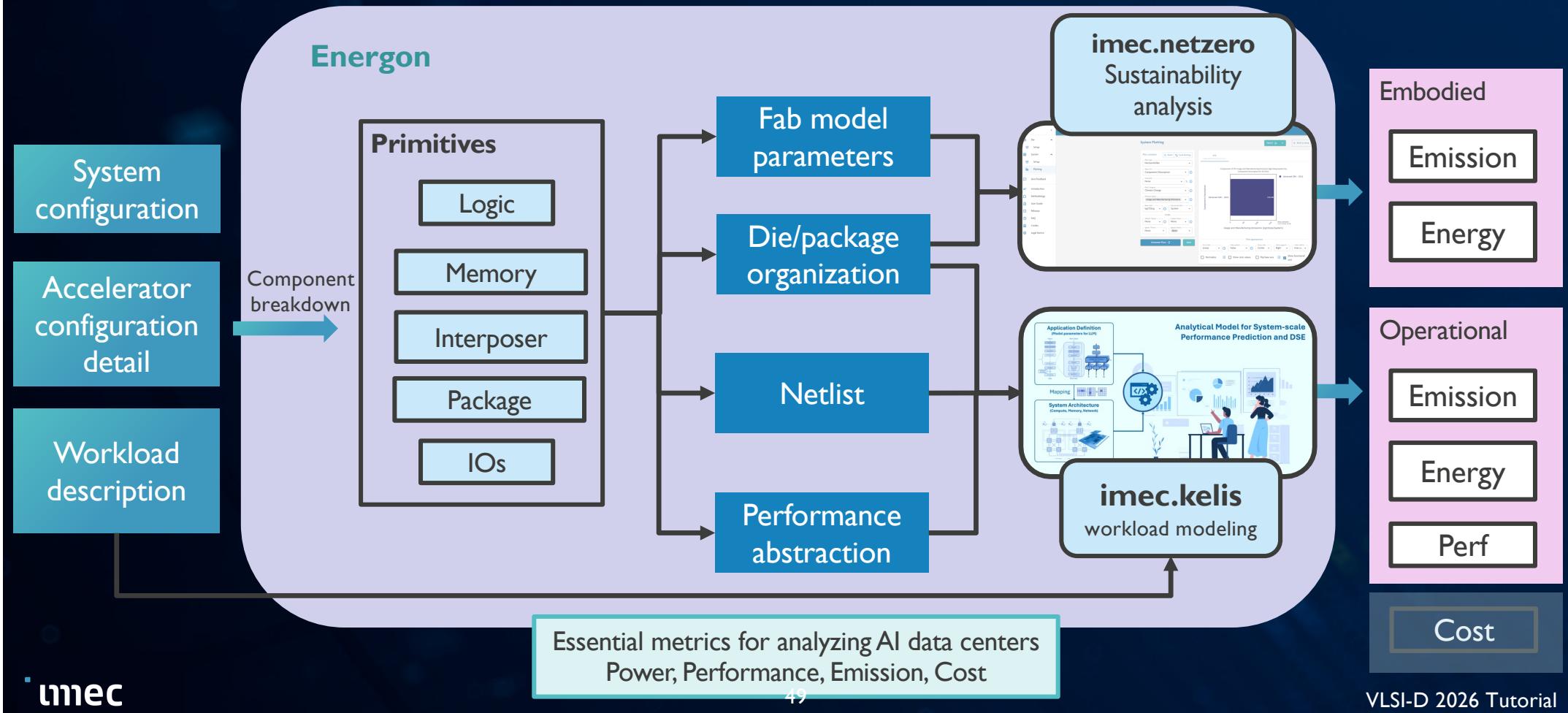
- GPU (Validated A/H100)
  - B200, H200, H100, A100
- Custom AI Accelerators
  - based on a general GPU architecture template

## Custom Testcases

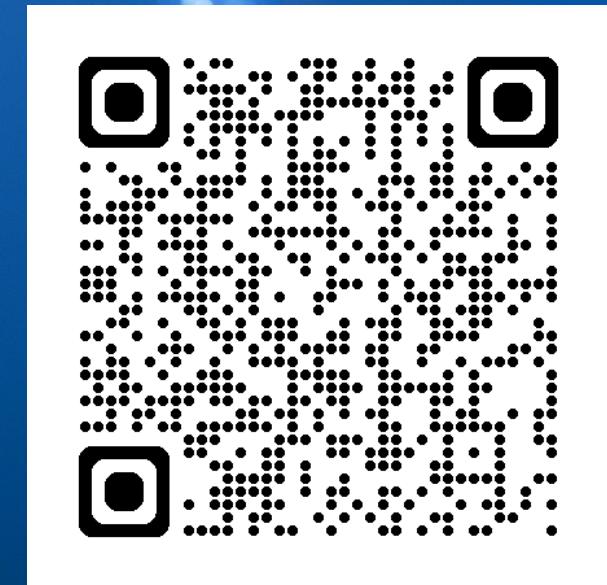
- RISC-V Accelerator
- Superconducting Processing Unit

# Optional link to imec.netzero (web only)

Augmenting with imec.netzero for embodied emissions



# Demonstration of the imec.KELIS and use-cases



[Analysis](#)[System Setup](#)

# System Setup

[Reset](#) [Load Setup](#)

Define a system in terms of an accelerator, a node architecture and system configuration. The system will be applied for further performance analysis.

Description: 

## Accelerator

Accelerator:



## Node Parameters

Number of Accelerators:

# Per Node

CPU:

Select CPU Type

▼ # CPUs

Host DRAM:

Select DRAM Size

▼ # DRAM

NIC:

Select NIC Type

▼ # For Storage # Per Accelerator

Intra-node Switch:

Select Switch Type

▼ # Intra-node Switches

PCIe Switch:

Select PCIe Switch Type

▼ # PCIe Switches

OS SSD:

Select OS SSD Type

▼ # OS SSDs

Storage SSD:

Select Storage SSD Type

▼ # Storage SSDs

## System Parameters

Number of Nodes:

[Save Setup](#)

Kalis UI 0.3.0-beta.1

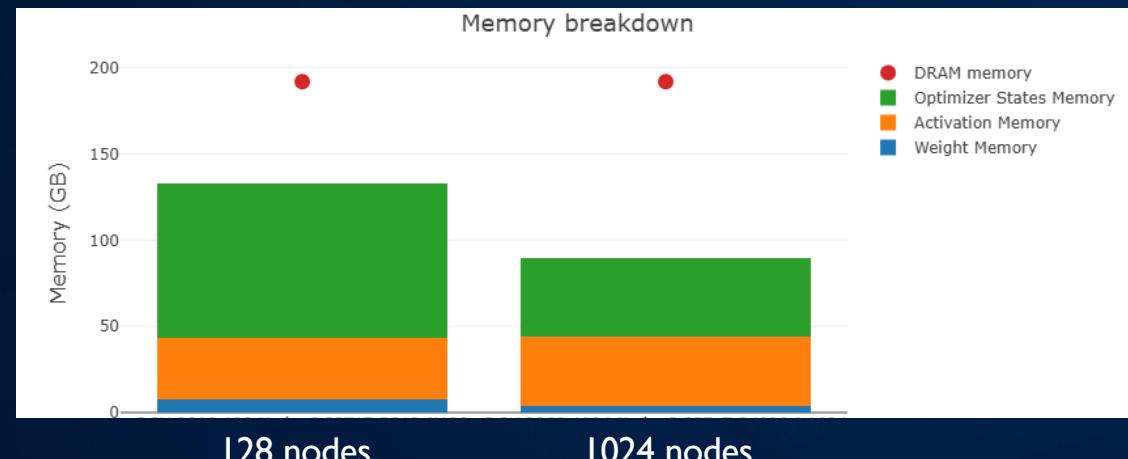
Kalis API 0.1.0



# System size scaling

## Pretraining on systems of different sizes

- System
  - DGX B200 systems
  - NVLink 5
  - ConnextX-8 InfiniBand NIC
- Model
  - GPT3 with 1T parameters
  - Batch size is tuned for the total system memory
- Parallelism mapping
  - Data, tensor and pipeline
  - Optimized for the total number of devices



# Workload scaling

Pretraining different models on the same system

➤ System

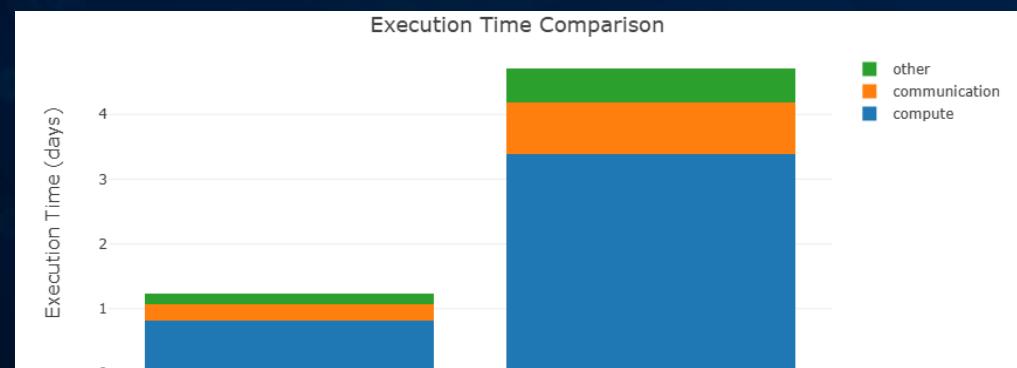
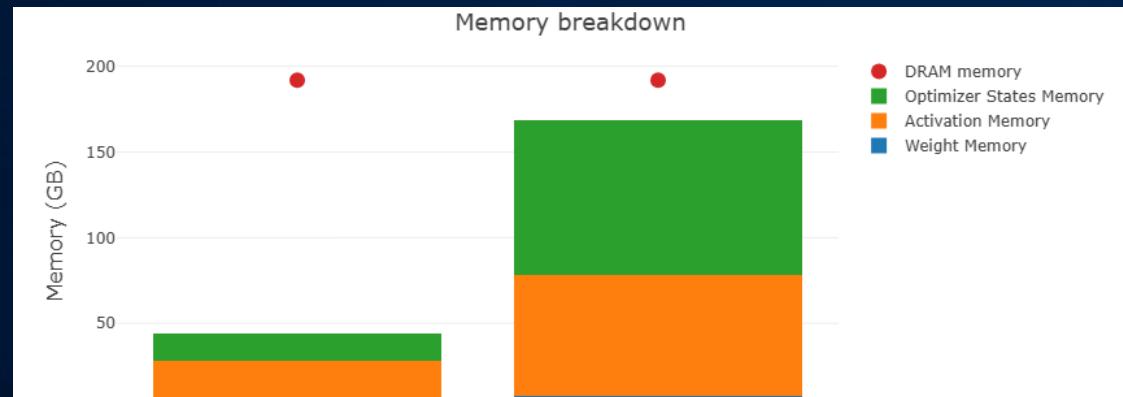
- DGX B200 systems with 128 nodes
- NVLink 5
- ConnextX-8 InfiniBand NIC

➤ Model

- GPT3
- Batch size is the same

➤ Parallelism mapping

- Data, tensor and pipeline
- Same mapping



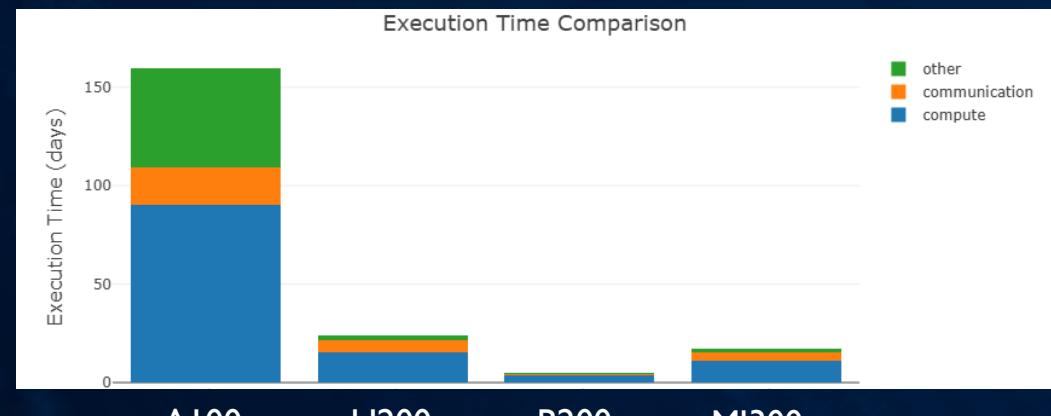
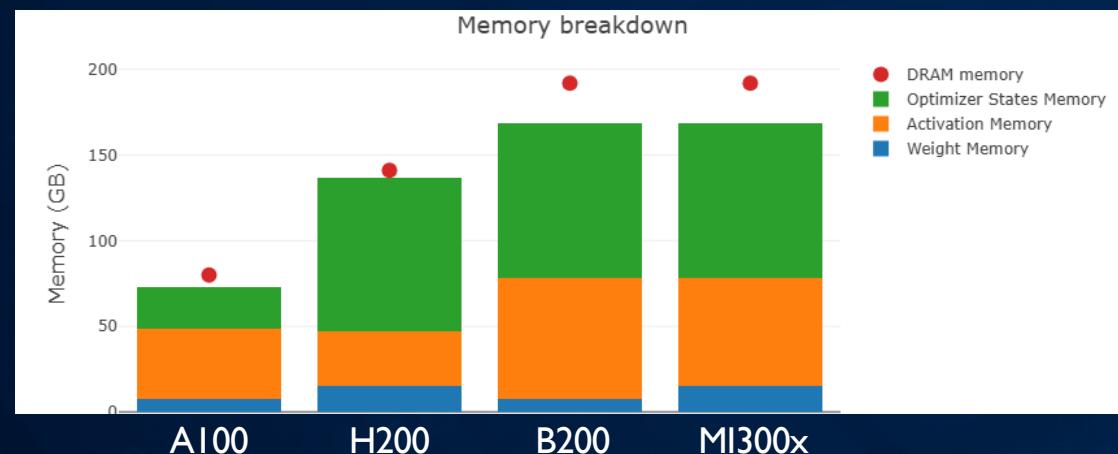
# System comparison

## Pretraining on different systems

- DGX System
  - 128 nodes
  - NVLink
  - ConnextX InfiniBand NIC
- MI300X system
  - 128 nodes
  - Infinity Fabric 4
  - ConnextX InfiniBand NIC

- Model
  - GPT3 with 1T parameters
  - Batch size is tuned for the total system memory

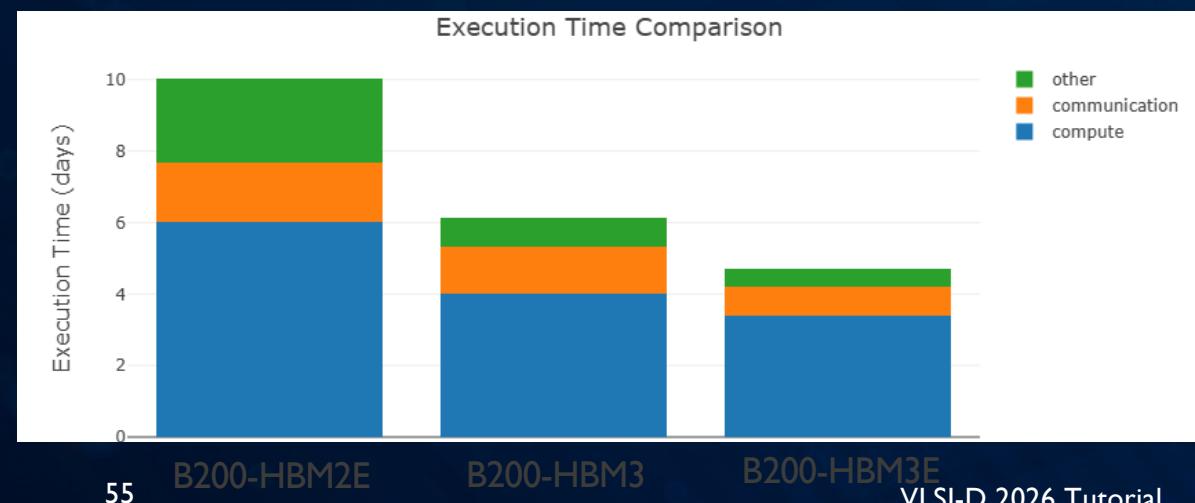
- Parallelism mapping
  - Data, tensor and pipeline
  - Optimized for the system capacity



# Memory technology scaling

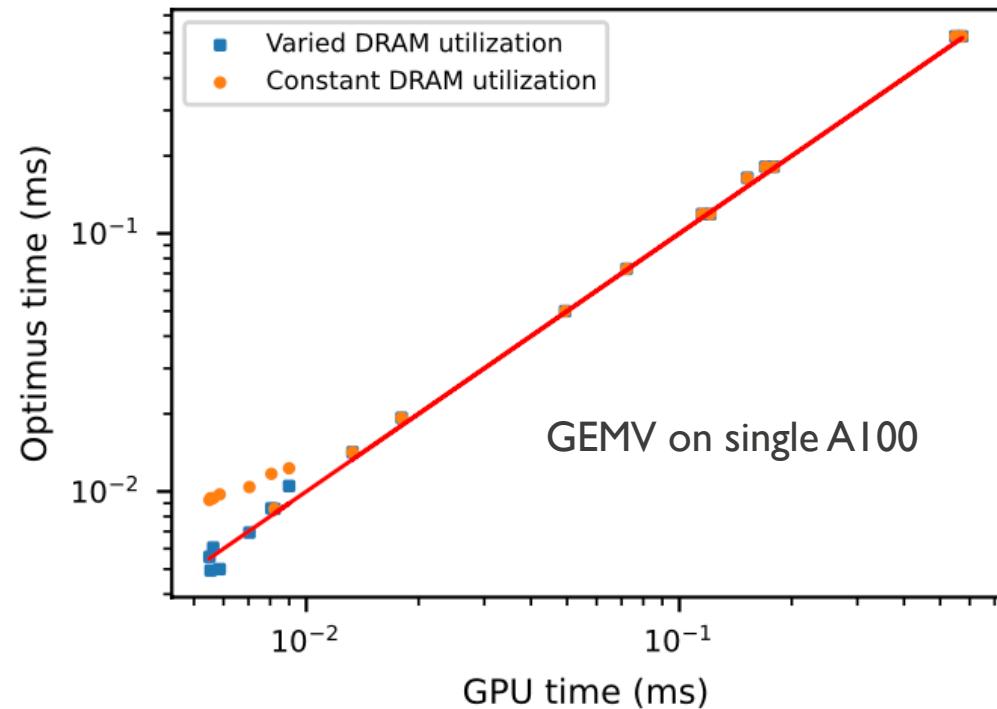
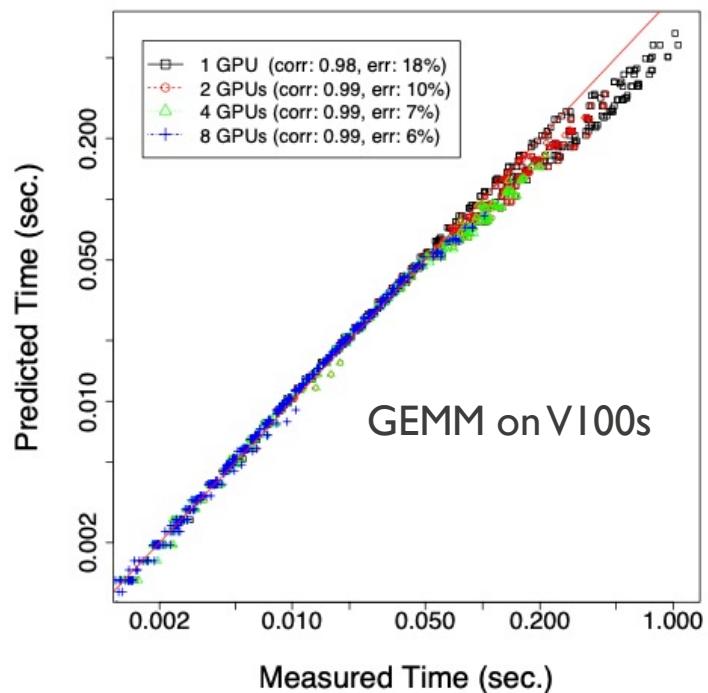
## Pretraining on systems with different memory technologies

- DGX B200 System
  - 128 nodes
  - NVLink 5
  - ConnextX-8 InfiniBand NIC
- Memory technology
  - HBM2E: 80 GB, 1.95TB/s
  - HBM3: 141 GB, 4.8 TB/s
  - HBM3E: 182 GB, 8 TB/s
- Model
  - GPT3 with 1T parameters
  - Batch size is tuned for the total system memory
- Parallelism mapping
  - Data, tensor and pipeline
  - Optimized for the system capacity



# Validation

# GEMM and GEMV Validation



# Performance validation for Nvidia GPUs

Compare predictions with published performance

Training A100

Model	# GPUs	Batch size	DP-TP-PP-SP	Activation recomputation	$t_{ref}$ in s from [28] [14]	$t_{pred}$ in s	$\delta E$ (%)
Only TP and PP							
GPT-22B	8	4	1-8-8-1	full	1.4	1.4	2.1
GPT-175B	64	64	1-8-8-1	full	18.1	16.9	6.9
GPT-530B	280	280	1-8-35-1	full	49.1	46.8	4.6
GPT-1008B	512	512	1-8-64-1	full	94.4	87.9	6.9
TP, PP and SP							
GPT-22B	8	4	1-8-8-8	selective	1.1	1.1	0.0
GPT-175B	64	64	1-8-8-8	selective	13.8	12.9	5.9
GPT-530B	280	280	1-8-35-8	selective	37.8	35.5	6.2
GPT-1008B	512	512	1-8-64-8	selective	71.5	69.1	3.4
DP, TP and PP							
GPT-310B	1920	2160	15-8-16-1	full	37.6	34.1	9.5
GPT-530B	2520	2520	9-8-35-1	full	54.2	51.2	5.5
GPT-1008B	3072	3072	6-8-64-1	full	102.4	100.7	1.6

Inference A100 + H100

Model	# GPUs	TP	$t_{nvidia}$ in ms (A100)	$t_{pred}$ in ms (A100)	$\delta E$ (%)	$t_{nvidia}$ in ms (H100)	$t_{pred}$ in ms (H100)	$\delta E$ (%)
Llama2-70B	8	8	4735	4284	9.5	3202	3147	1.7
Llama2-70B	4	4	6403	6019	6.0	4116	3986	3.2
Llama2-70B	2	2	10500	10042	4.4	6267	6186	1.3
Llama2-13B	8	8	1693	1514	10.6	1201	1209	0.7
Llama2-13B	4	4	1894	1748	7.7	1431	1258	12.1
Llama2-13B	2	2	2499	2492	0.2	1717	1617	5.8
Llama2-13B	1	1	3884	4263	9.7	2396	2599	8.5
Llama2-7B	8	8	1187	1096	7.7	828	899	8.6
Llama2-7B	4	4	1280	1166	8.9	924	869	5.6
Llama2-7B	2	2	1544	1526	1.2	1143	1016	11.1
Llama2-7B	1	1	2190	2472	12.9	1440	1522	5.7

# Relevant Publications

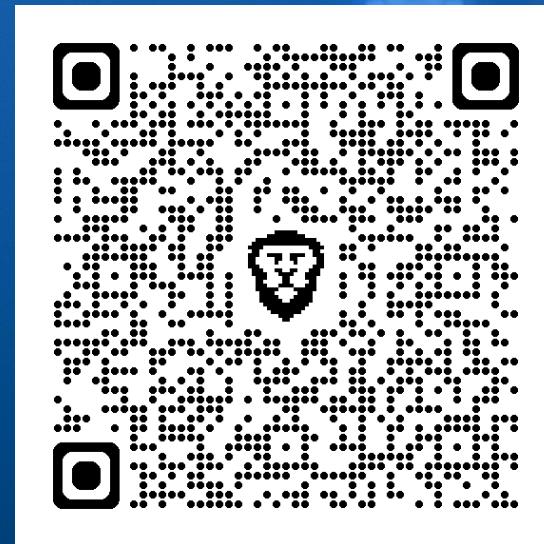
- D. Moolchandani\*, J. Kundu\*, F. Ruelens, P. Vrancx, T. Evenblij and M. Perumkunnil, "AMPeD: An Analytical Model for Performance in Distributed Training of Transformers," 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Raleigh, NC, USA, 2023, pp. 306-315, doi: 10.1109/ISPASS57527.2023.00037.
- J. Kundu\*, W. Guo\*, A. BanaGozar\*, U. De Alwis, S. Sengupta, P. Gupta, A. Mallick, "Performance Modeling and Workload Analysis of Distributed Large Language Model Training and Inference", 2024 IEEE International Symposium on Workload Characterization (IISWC), Vancouver, BC, Canada, 2024, pp. 57-67, doi: 10.1109/IISWC63097.2024.00015
- W. Guo\*, J. Kundu\*, U. Tos, W. Jiang, G. Sisto, T. Evenblij, M. Perumkunnil, "Keeping up with Large Language Models: A Holistic Methodology of Compute, Memory, Communication, and Cost Modeling," in 2025 IEEE International Symposium on Workload Characterization (IISWC), Irvine, CA, USA, 2025, pp. 116-126, doi: 10.1109/IISWC66894.2025.00019.
- J. Kundu\*, Debjyoti Bhattacharjee, Nathan Josephsen, Ankit Pokhrel, Udara De Silva, Manu Perumkunnil, Quentin Herr, Anna Herr, "A System Level Performance Evaluation for Superconducting Digital Systems", Accepted in DATE 2025, <https://arxiv.org/abs/2411.08645>

# Challenges in Analytical Performance Modeling

-  **Oversimplification**  
Static equations can't capture dynamic runtime optimizations (gradient checkpointing, adaptive batching)
-  **Communication Complexity**  
Real distributed training shows variable network behavior that bandwidth-latency models miss
-  **Memory Hierarchy in heterogeneous environment**  
Cascading effects across L1→L2→L3→HBM→Network are extremely difficult to model analytically when the nodes are heterogeneous in nature
-  **Tail Latencies Matter**  
Stragglers and edge cases dominate training time but disappear in average-case analysis

# COFFEE BREAK

15:45-16:10



Scan this to ask your question

# Design Space Exploration

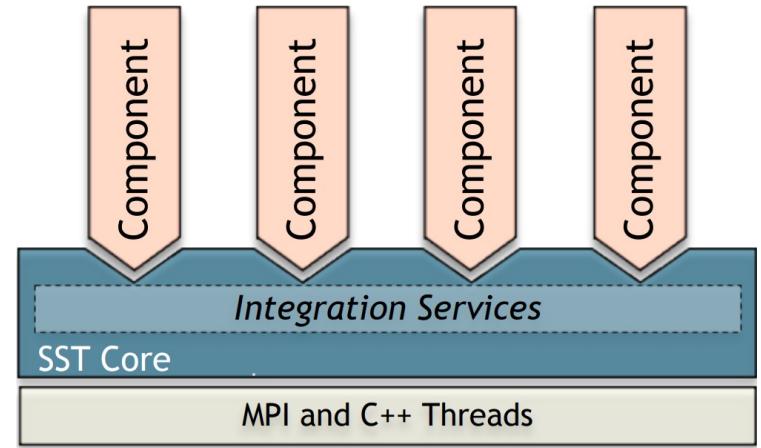
Profile real-life workloads for optimal design choices based on performance vs. PPAC analysis, across parameters from:

- ✓ Microarchitecture
- ✓ Memory and storage
- ✓ Network configuration and topology
- ✓ Many more

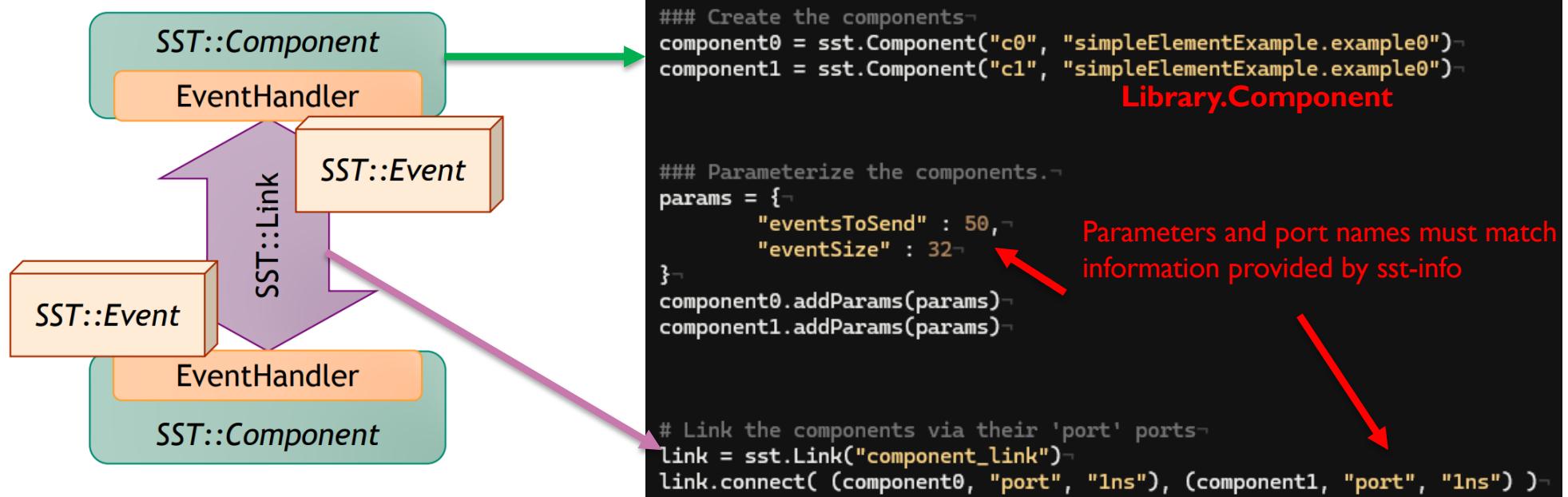


# Introduction to Structural Simulation Toolkit (SST):

- SST is a Parallel discrete-Event Simulator Framework
  - Developed by Sandia National Laboratories
  - Uses MPI for parallelization: **Allows large scale simulation**
- Divided into two projects:
  - **SST-core framework:**
    - The backbone of the simulator
    - Provides utilities and interfaces for components
    - **You don't need to edit it**
  - **SST-Elements Libraries:**
    - Libraries of components that model compute system
    - You can add new components or libraries
    - **More information on <https://sst-simulator.org>**
- Components are modeled in C++
- **Simulations are built with python script**



# Introduction to Structural Simulation Toolkit (SST): Building a system

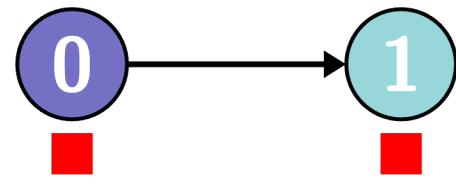


Source: [A-SST Initial Specification](#)

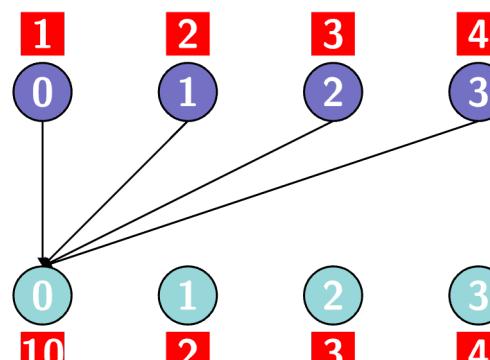
Python script snippet

# Background on Parallel Programming on Distributed Memory System

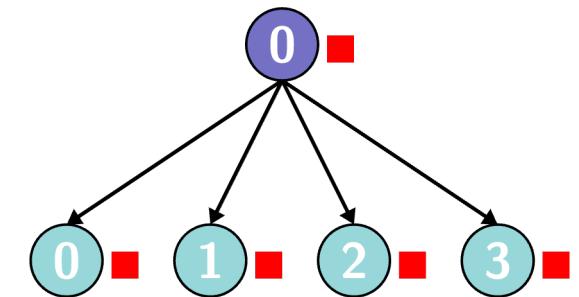
## Message Passing Interface (MPI) collective communication



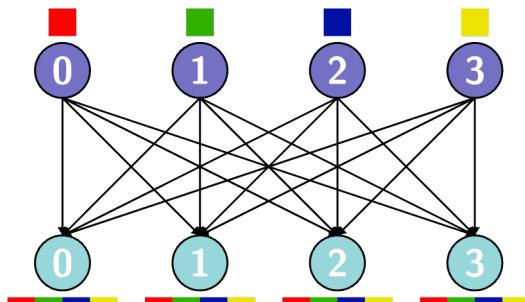
Send-Recv communication



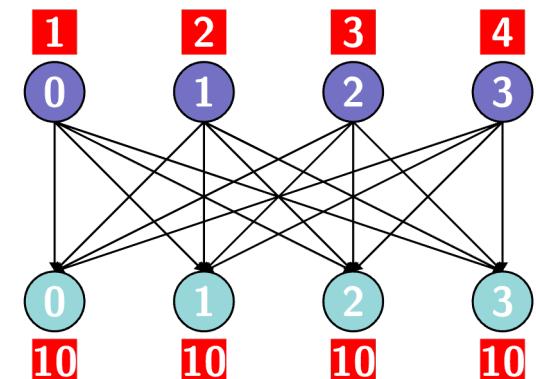
Reduce (sum) communication



Broadcast (bcast) communication

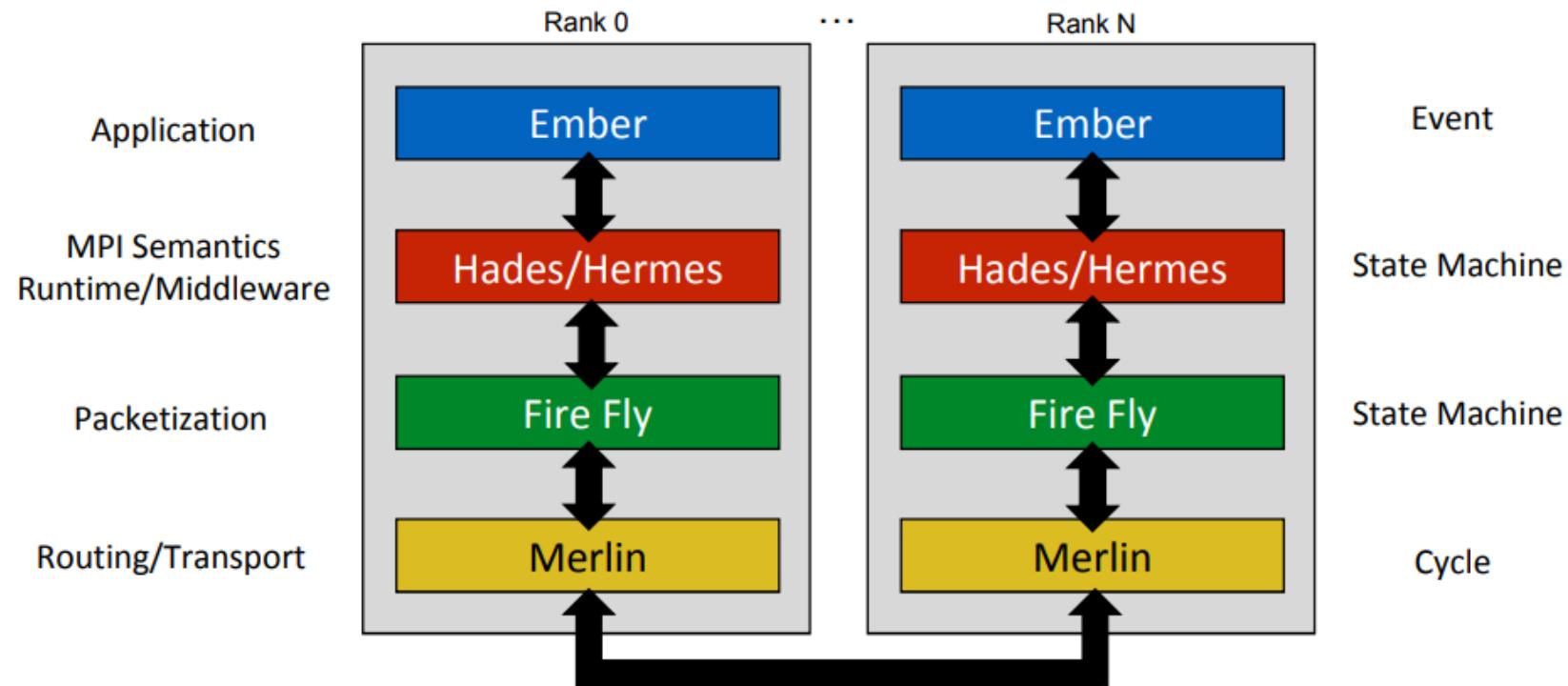


All gather communication



All reduce (sum) communication

# Packet-level simulation with SST/Ember



From: "Ember: Reference Communication Patterns for Exascale. S.D. Hammond et al." ([osti.gov/servlets/purl/1307276](https://osti.gov/servlets/purl/1307276))

## Packet-level simulation with SST/Ember

More complex motifs can be implemented by using a finite state machine

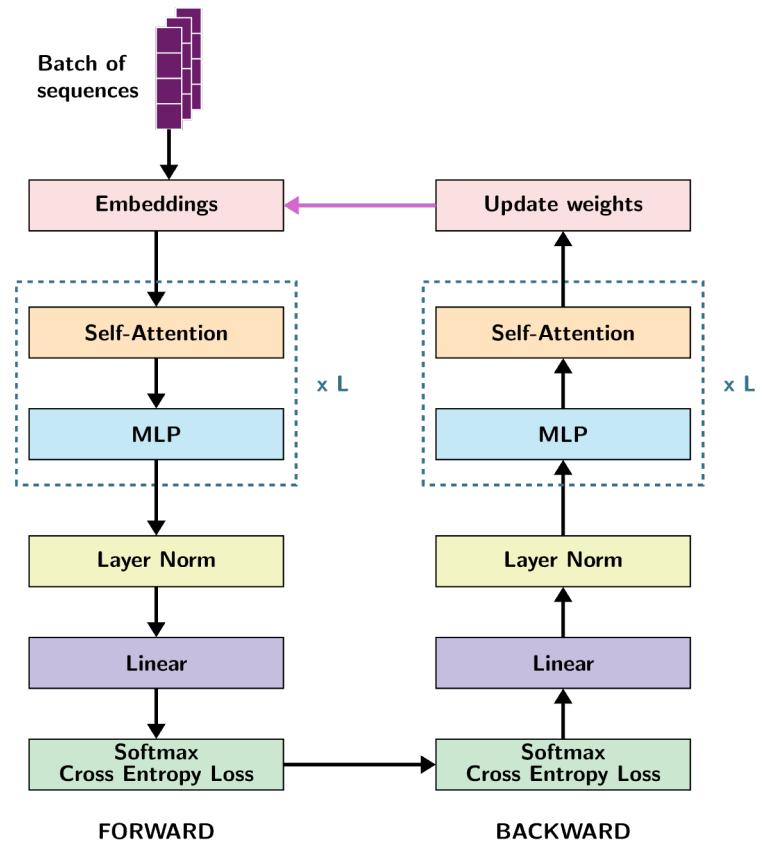
```
bool EmberDummyGenerator::generate(queue<EmberEvent*>& evQ) {  
    bool finish = false;  
  
    sendBuf = memAlloc(sizeof(double)*count);  
    recvBuf = memAlloc(sizeof(double)*count);  
  
    enQ_compute( evQ, 1e6 );  
    enQ_allreduce(evQ, sendBuf, recvBuf, count, DOUBLE, Hermes::MP::SUM, GroupWorld);  
  
    loop_index++;  
  
    if(loop_index == num_iter)  
        finish = true;  
  
    return finish;  
}
```

- Generates *num\_iter* times the same motif → generation stops when the function return true.
- After 1ms ( $1e6$  ns) of compute time → *all reduce* communication is scheduled.
- When *all reduce* communication is completed → next compute phase is scheduled.

# Background on training parallelism

## Decoder-only transformers

- The input is a batch of token sequences → converted into embeddings tensor by the embeddings layer
- Embeddings tensor crosses L hidden layers includes
  - Self-Attention block
  - Multilayer perceptron (MLP)
- Embeddings tensor is processed by a normalization function → projected onto the vocabulary (linear).
- The loss is calculated with a SoftMax and a cross-entropy function.
- Finally, the loss is backpropagated to update the weights



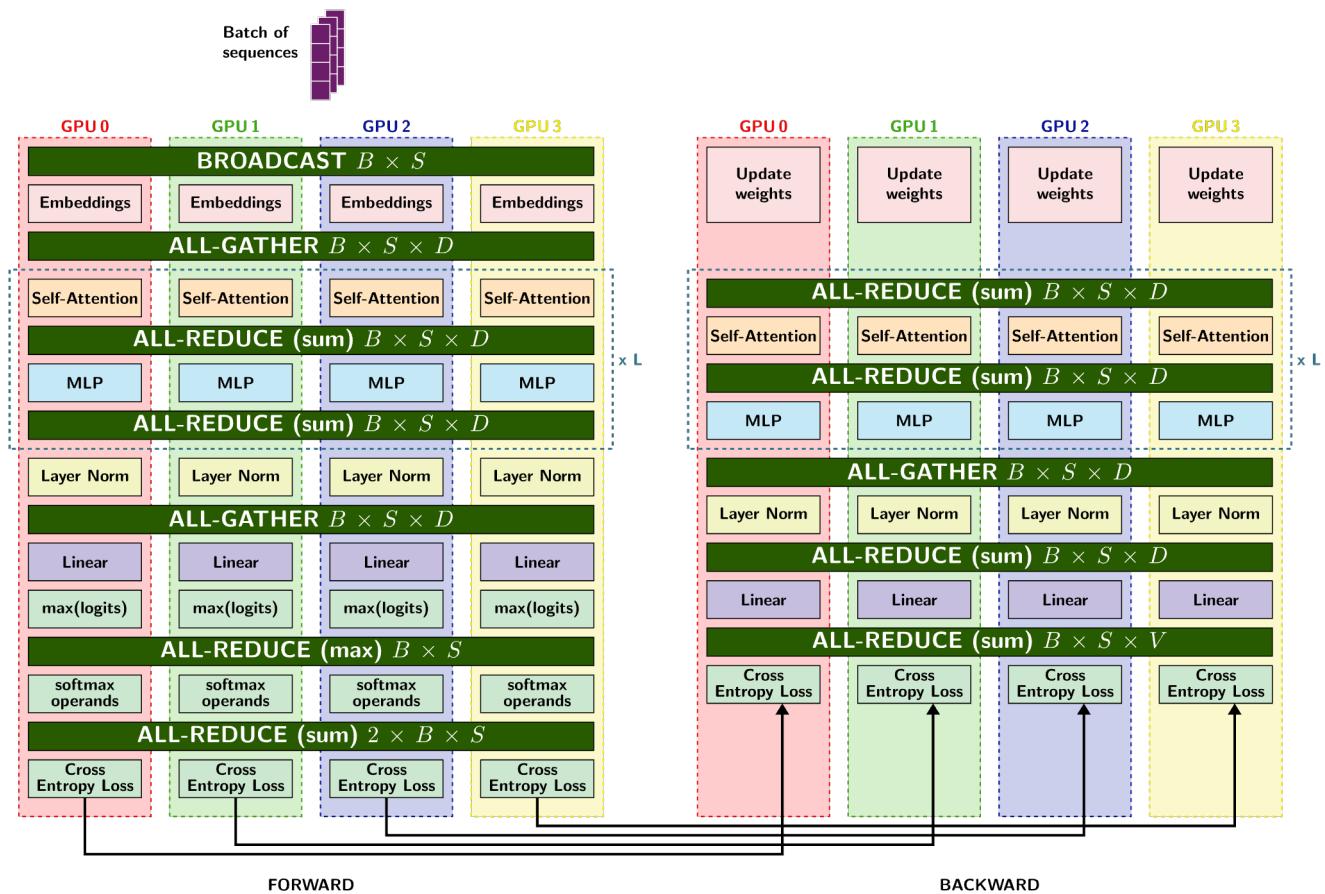
Architecture overview of decoder-only transformers

# Background on training parallelism

## Tensor parallelism

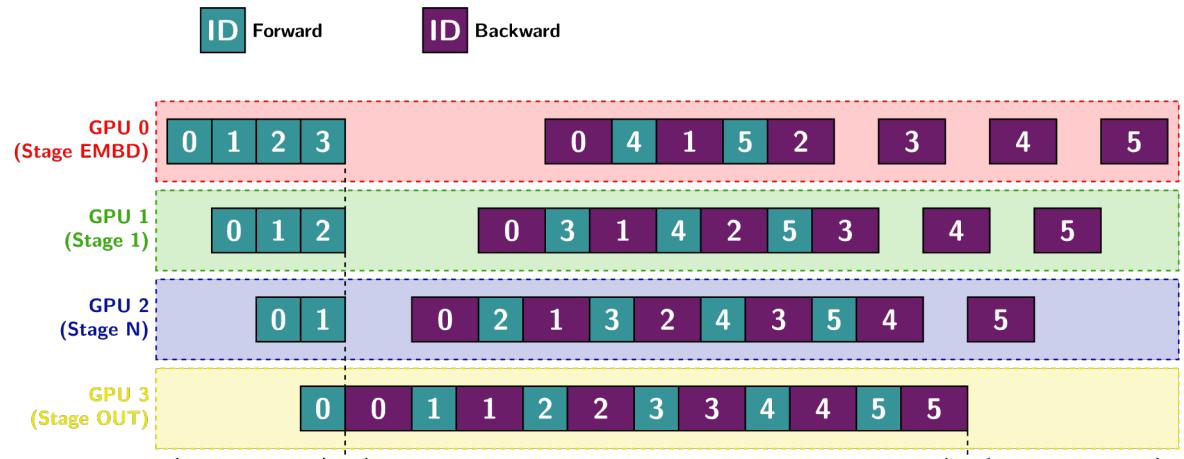
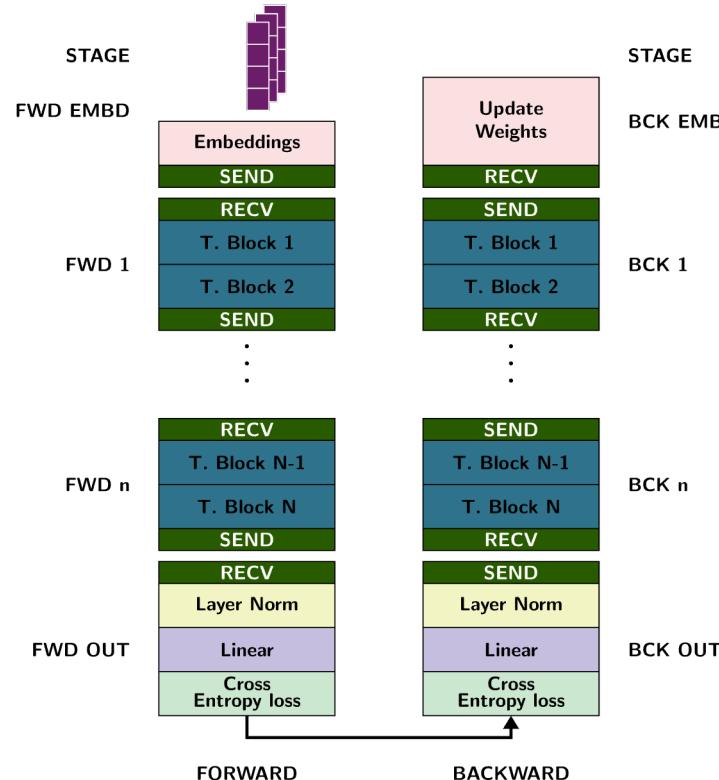
Symbol	Name	Typical value
B	Batch size	32
S	Sequence length (# tokens per sequence)	8192
D	Model dimension (hidden size)	8192
V	Vocabulary size	128256

Weight matrices are split and distributed across the GPUs



# Background on training parallelism

## Pipeline parallelism



one-forward-one-backward (1F1B) pipeline parallelism

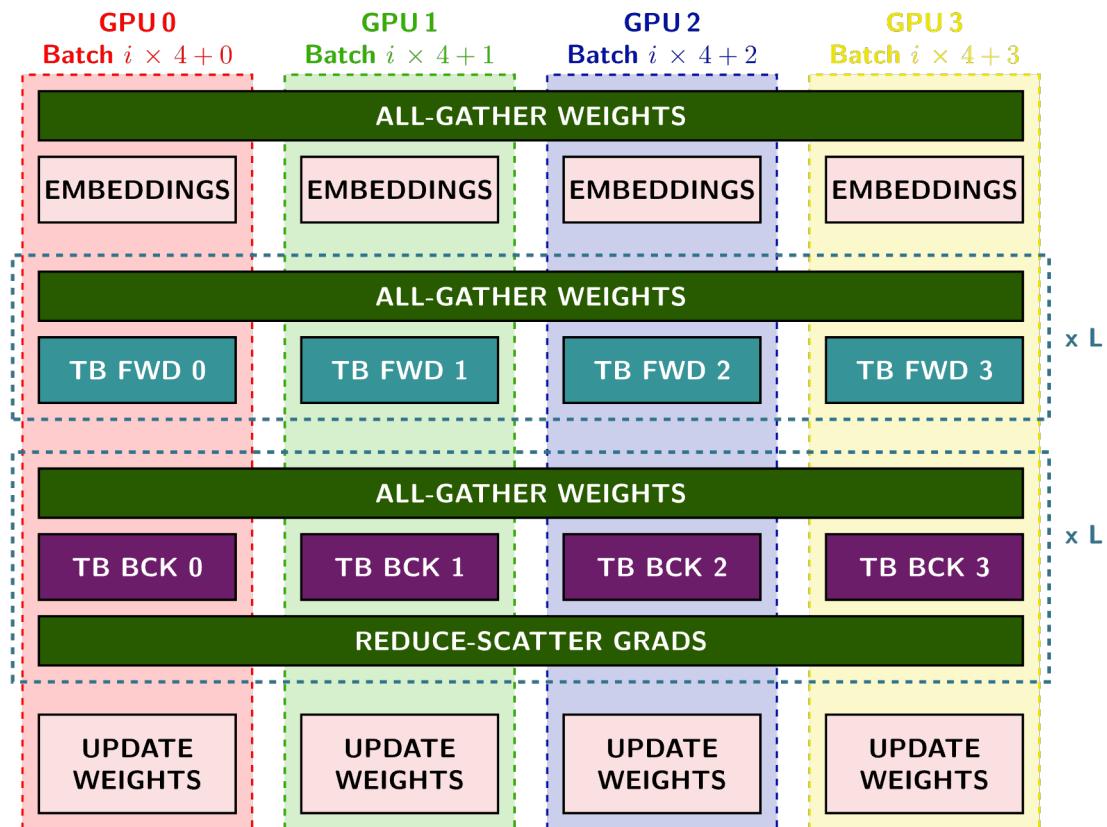
**The execution is split into stages which are mapped to a GPU:**

- The same mapping must be applied for the forward and backward path
- In this tutorial, we assume that 1 GPU is dedicated to embeddings stages and another one to the output stage
- The Transformer blocks are distributed across the remaining GPUs
- Requires at least 3 GPUs

# Background on training parallelism

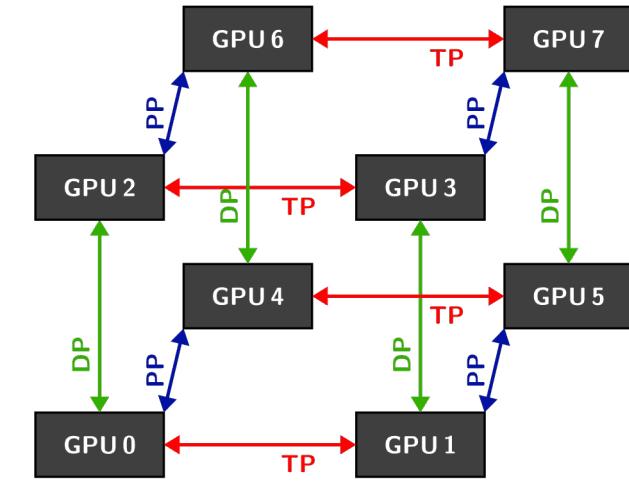
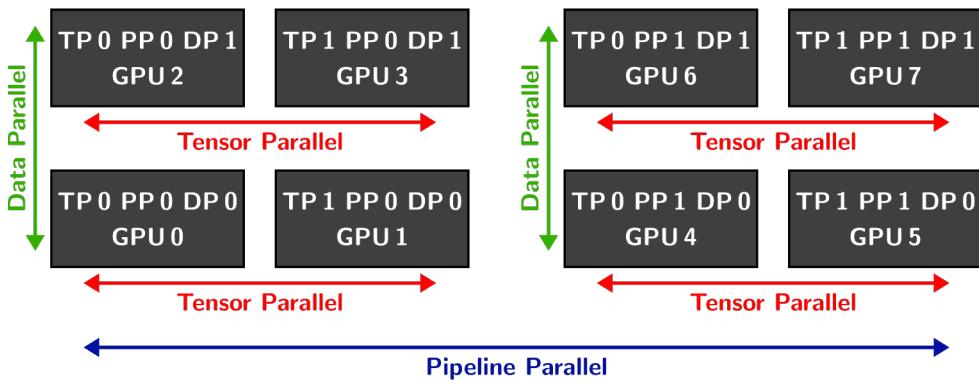
## Data parallelism

- **The Weights and gradients are distributed over the GPUs:**
- Each GPU processes a different batch
- Each GPU stores a subset of the weights
- Before executing a new transformer block, all GPU share their subset to acquire all the weights pertaining to the block
- In the backward path, after executing a block, each GPU performs a reduce to acquire a subset of the distributed gradients



# Background on training parallelism

## 3D parallelism



**3D parallelism combines Tensor Parallelism, Pipeline Parallelism and Data Parallelism:**

- GPU stores smaller subset of the weights:
  - GPU 0, 1, 2 & 3 store each only  $\frac{1}{4}$  of the embedding weights and need to acquire only half of the weights
- For Data parallelism GPU 0 communicates only with GPU 2
- For Tensor parallelism GPU 0 communicates only with GPU 1
- For Pipeline parallelism GPU 0 communicates only with GPU 4

# Exploring scaling of LLM training

## Tensor parallelism

**Let's explore the scalability of tensor parallelism by sweeping the level of parallelism:**

Command to run:

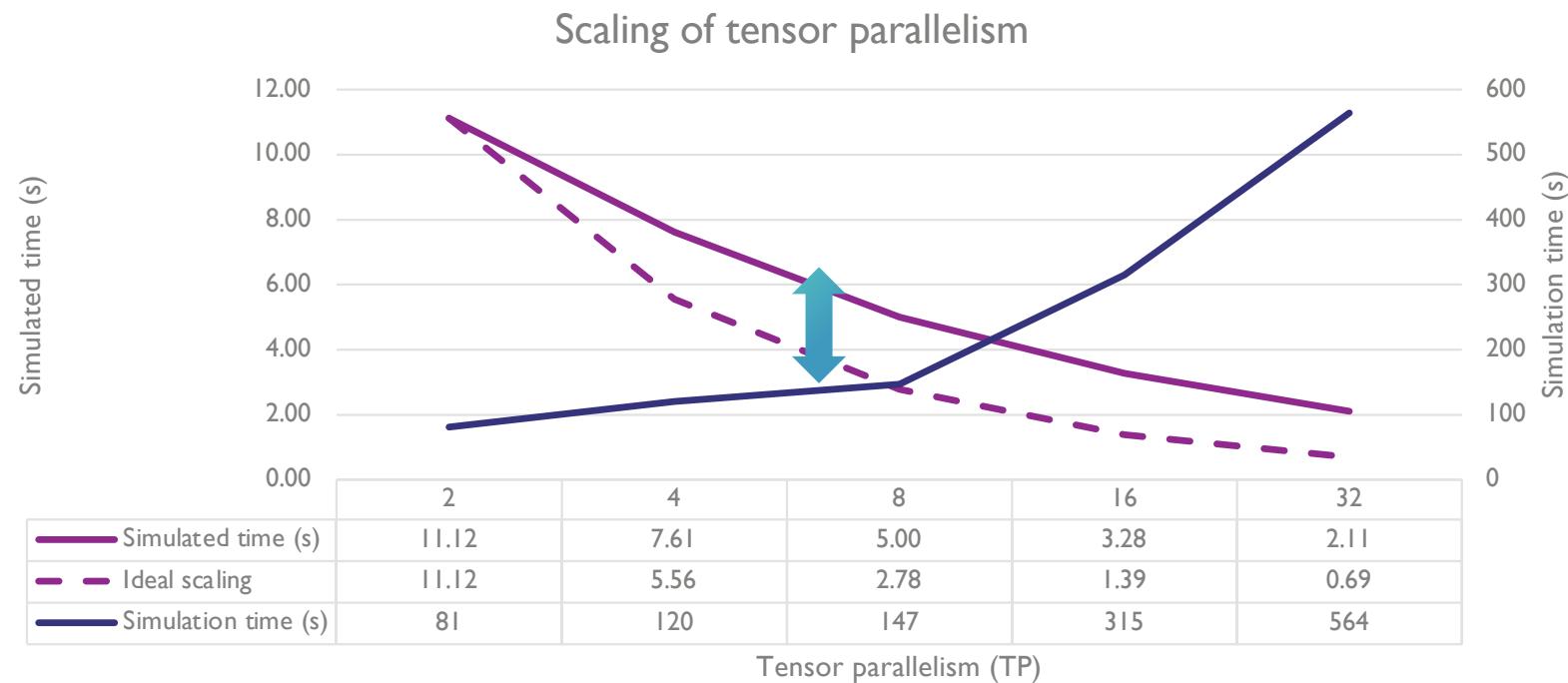
```
sst --print-timing-info training_llm.py --tp TP --pp 1 --dp 1 --batch_size 16 --sequence_len 1024 --n_batch 8 --llm_config small_config.json --log logger --stats stats.csv --topology single --verbose 10
```

You can fill the table below:

Tensor Parallelism (TP)	2	4	8	16	32
Simulated time (s)					
Simulation time(s)					

# Exploring scaling of LLM training

## Tensor parallelism



Gap between analytical and event-driven  
simulation prediction

# Exploring scaling of LLM training

## Pipeline parallelism

**Let's explore the scalability of pipeline parallelism by swiping the level of parallelism:**

Command to run:

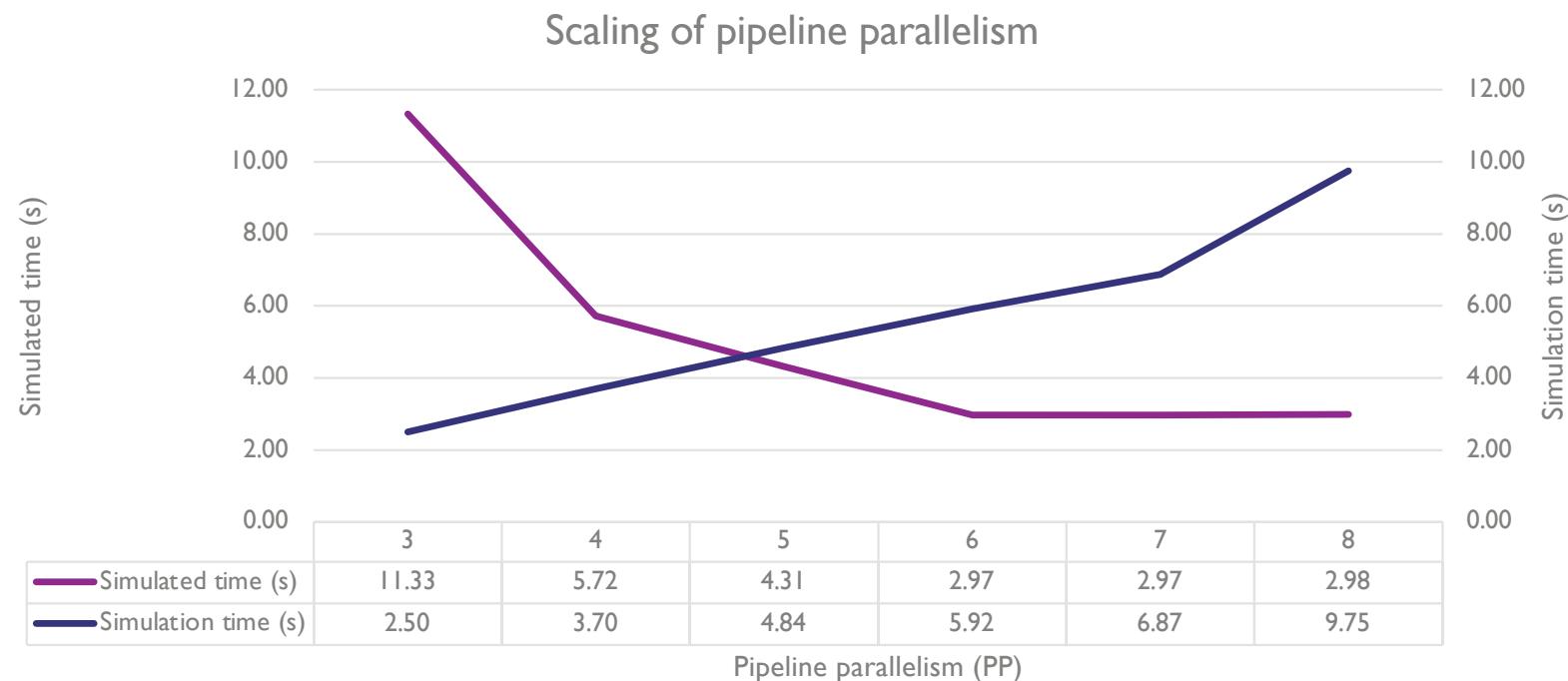
```
sst --print-timing-info training_llm.py --tp 1 --pp PP --dp 1 --batch_size 16 --sequence_len 1024 --n_batch 8 --llm_config small_config.json --log logger --stats stats.csv --topology single --verbose 10
```

You can fill the table below:

Pipeline Parallelism (PP)	3	4	5	6	7	8
Simulated time (s)						
Simulation time(s)						

# Exploring scaling of LLM training

## Pipeline parallelism



# Exploring scaling of LLM training

## Data parallelism

**Let's explore the scalability of data parallelism by swiping the level of parallelism:**

Command to run:

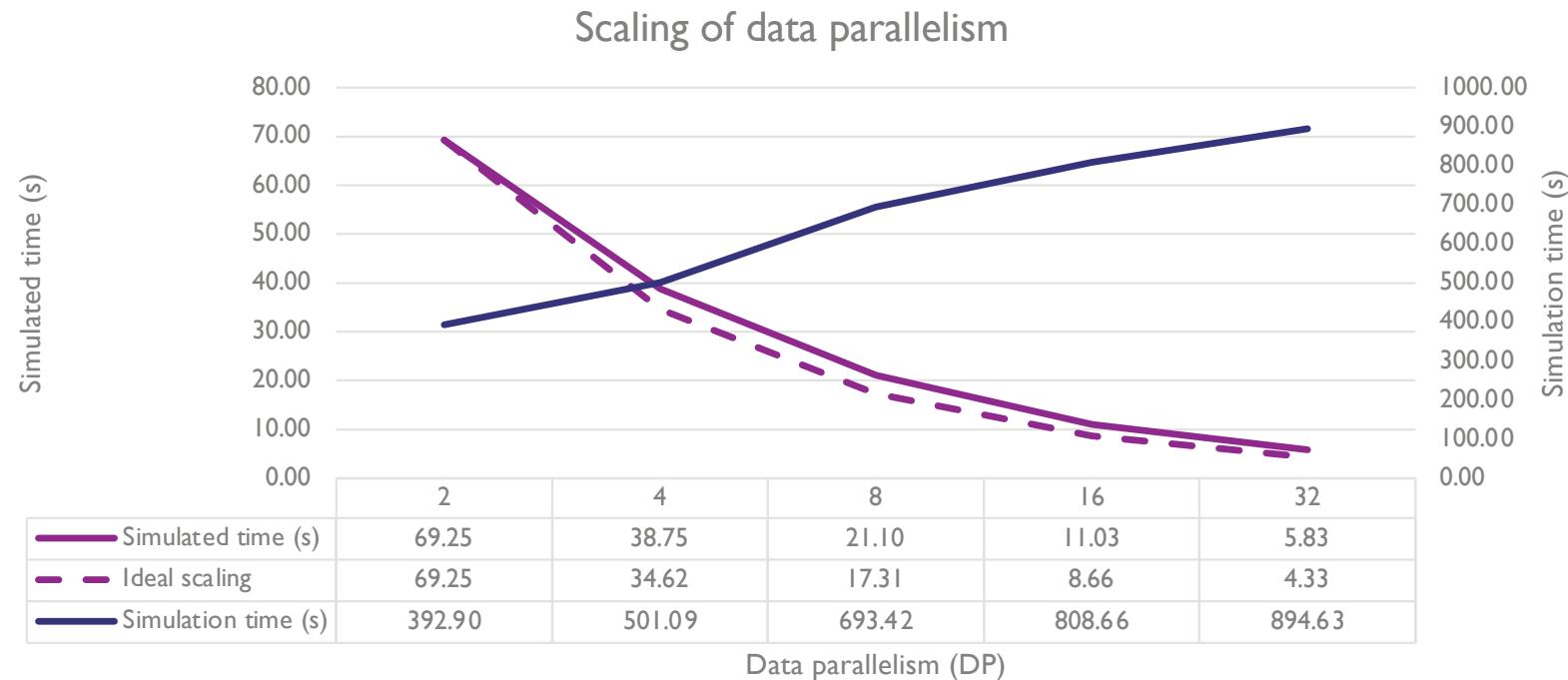
```
sst --print-timing-info training_llm.py --tp 1 --pp 1 --dp DP --batch_size 16 --sequence_len 1024 --n_batch 64 --llm_config small_config.json --log logger --stats stats.csv --topology single --verbose 10
```

You can fill the table below:

Data Parallelism (DP)	2	4	8	16	32
Simulated time (s)					
Simulation time(s)					

# Exploring scaling of LLM training

## Data parallelism



# Exploring scaling of LLM training

## 3D parallelism

**Let's explore the impact of the topology on the performance:**

Command to run:

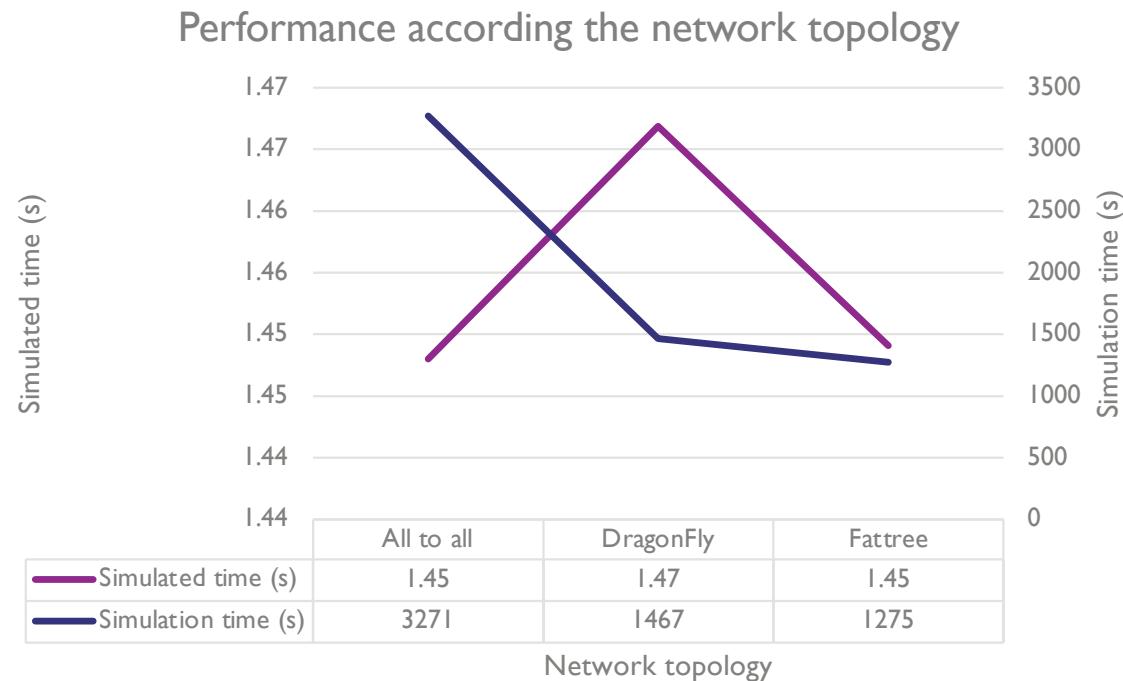
```
sst --print-timing-info training_llm.py --tp 8 --pp 6 --dp 8 --batch_size 16 --sequence_len 1024 --n_batch 32 --llm_config small_config.json --log logger --stats stats.csv --topology TOPOLOGY --verbose 10
```

You can fill the table below:

Topology	single	dragonfly	fattree
Simulated time (s)			
Simulation time(s)			

# Exploring scaling of LLM training

## 3D parallelism



## Wrap-up

SST provides an open-source framework for simulating large-scale HPC systems:

- Discrete event simulation parallelized with threads and MPI ranks
- Library of computer system elements → Offers a good starting point to model “target systems”
- Python library to build *scale-out* system with complex network topologies

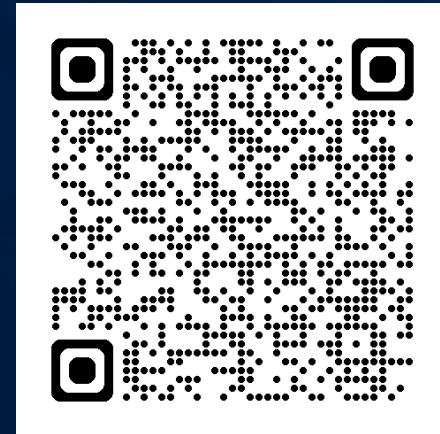
SST also allows connecting gem5 compute core with its uncore infrastructure.

## Parting thoughts

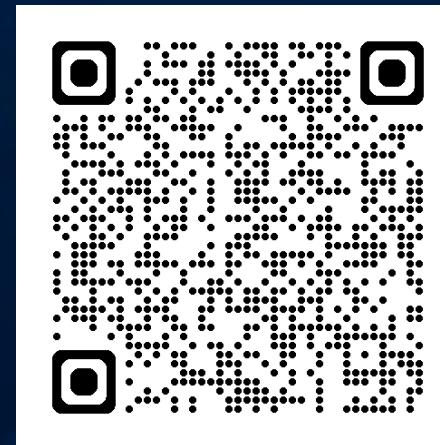
- HW-SW Codesign is becoming important for efficient system design.
- Multi-fidelity, hierarchical system simulation is a pragmatic approach to translate customer requirements to functional specs of a compute system.
- Workload representation is crucial element to enable multi-fidelity simulation at large scale

# Acknowledgements

- Wenzhe Guo and imec.KELIS team
- Uras Tos and imec.DARE team
- Jonas Svedas and the imec.HESPAS team



VLSI-D 26 Tutorial

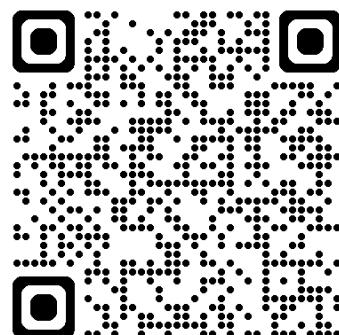


VLSI-D 25 Tutorial

VLSI-D 2026 Tutorial

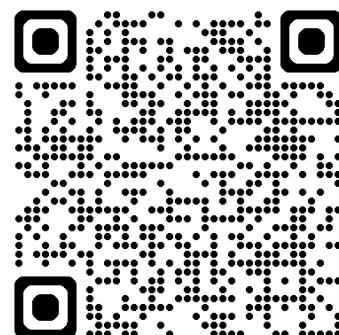
# A glimpse of the real workforce

## CSA@imec



Research@CSA

<https://www.imec-int.com/en/expertise/compute-system-architecture>



Jobs@imec

<https://www.imec-int.com/en/work-at-imec/job-opportunities>

# Our future starts with you.

We are looking for over 10+ new colleagues in our department

Discover your future on [www.imec-int.com/careers](http://www.imec-int.com/careers)

imec



לומֶק

embracing a better life