PLDI'18 & ISMM'18

Trip Report

Jun 18 - 22 2018, Philadelphia

Fangzhou Liu

Fri Jul 06, 2018

Agenda

- Overview and Important Notes
- Enhancing Computation-to-Core Assignment with Physical Location Information

  Orhan Kislal, Jagadish Kotra, Xulong Tang, Mahmut Taylan Kandemir, Myoungsoo June

  Pennsylvania State University, Yonsei University
- Mapping Spiking Neural Networks onto a Manycore Neuromorphic Architecture

  Chit-Kwan Lin, Andreas Wild, Gautham N. Chinya, Tsung-Han Lin, Mike Davies, Hong Wang

  Intel Labs

Agenda

- **Overview and Important Notes**
- Enhancing Computation-to-Core Assignment with Physical Location Information
- Mapping Spiking Neural Networks onto a Manycore Neuromorphic Architecture

PLDI 2018
Overview

- 55 papers accepted out of 245 submitted (*22.4%*) — PLDI
  9 papers accepted out of 21 submitted (42.9%) — ISMM

- Co-located with ISMM, LCTES, ARRAY, DeepSpec, FMS, MAPL

- Covers Multiple Topics
  Hardware, Optimization, Concurrency, Synthesis and Learning, Transaction, Program Analysis, Verification, Probabilistic program *et al*.

https://pldi18.hotcrp.com/
https://ismm18.hotcrp.com/

PLDI 2019
Federated Computing Research Conference

- Hold at Phoenix Convention Center, Phoenix, Arizona
- Co-located with COLT, E-energy, EC, HPDC, ICS, ISCA, IWQoS, SIGMETRICS, SPAA and STOC
- June 22 - 28, 2019

https://fcrc.acm.org/

PLDI 2019
Important Dates

- Fri 16 Nov, 2018
  Research Papers Submission Deadline
- Tue 29 - Thu 31 Jan, 2019
  Author Response Period
- Fri 15 Feb, 2019
  Author Notification
- Tue 16 Apr, 2019
  Camera-Ready Deadline
- Mon 24 - Wed 26 Jun, 2019
  Main Conference

- Extends and/or applies programming-language concepts to advance the field of computing.
- Novel system designs, thorough empirical work, well-motivated theoretical results.
- New application areas.
- …

https://pldi19.sigplan.org/dates

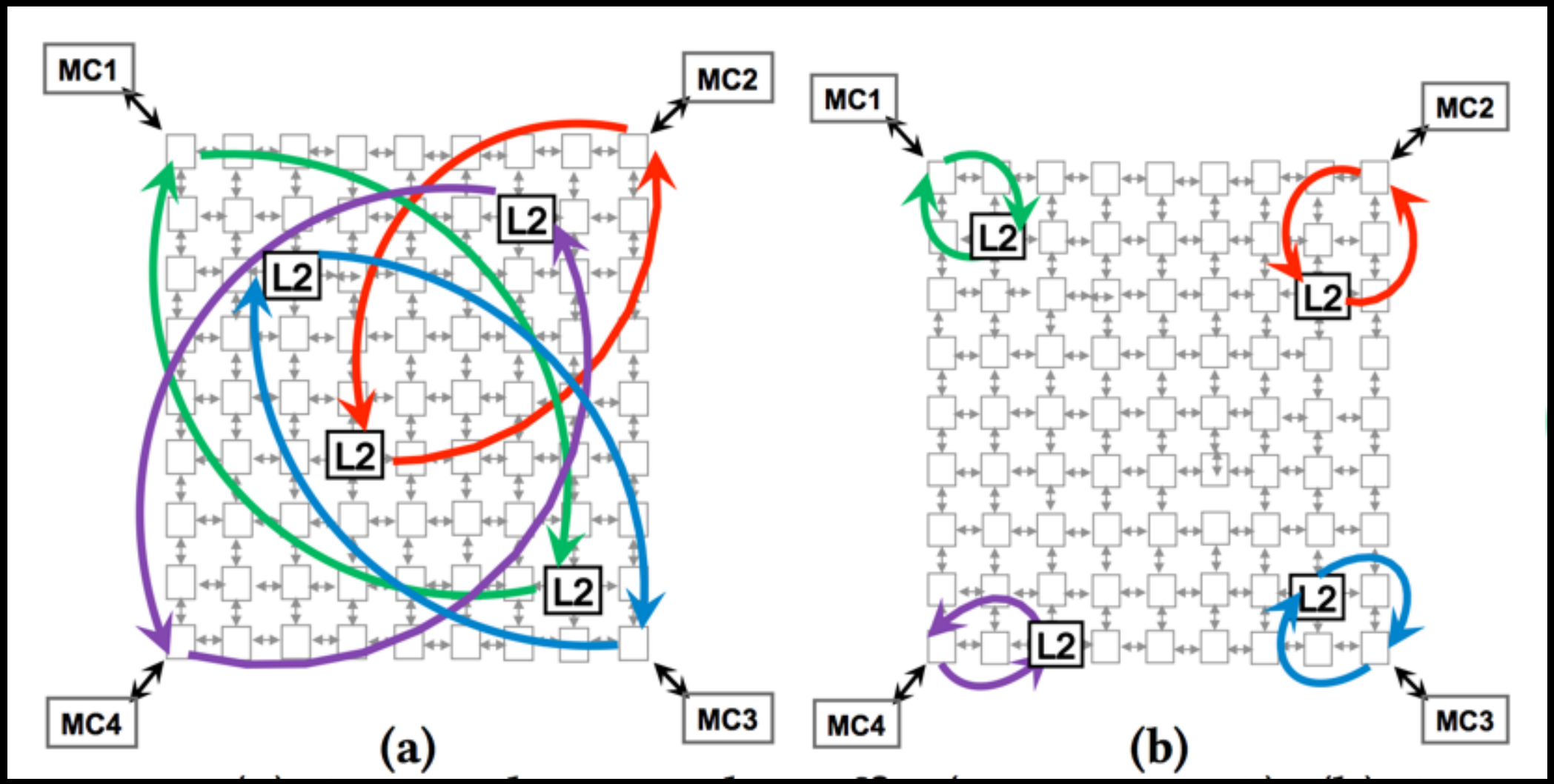
Agenda

- Important Notes
- **Enhancing Computation-to-Core Assignment with Physical Location Information**
- Mapping Spiking Neural Networks onto a Manycore Neuromorphic Architecture

Introduction

- GOAL: Reduce the On-chip Network Latency in NUCA system
- METHOD: Compiler-aid Computation-to-Core Mapping
  - The execution time can be saved (**14%** and **17.1%** for Private LLCs and Shared LLCs on average) when maximizing the on-chip network performance
- PLATFORM: gem5 simulator and Intel KNL system

Computation to Core Mapping
Example (P-NUCA)



Computation to Core Mapping
Definition

- Memory affinity of iterations (MAI)
  Fraction of the data requests of iteration set *I* to all memory controller in the system

- Memory affinity of cores (MAC)
  The Manhattan Distance between core and memory controller

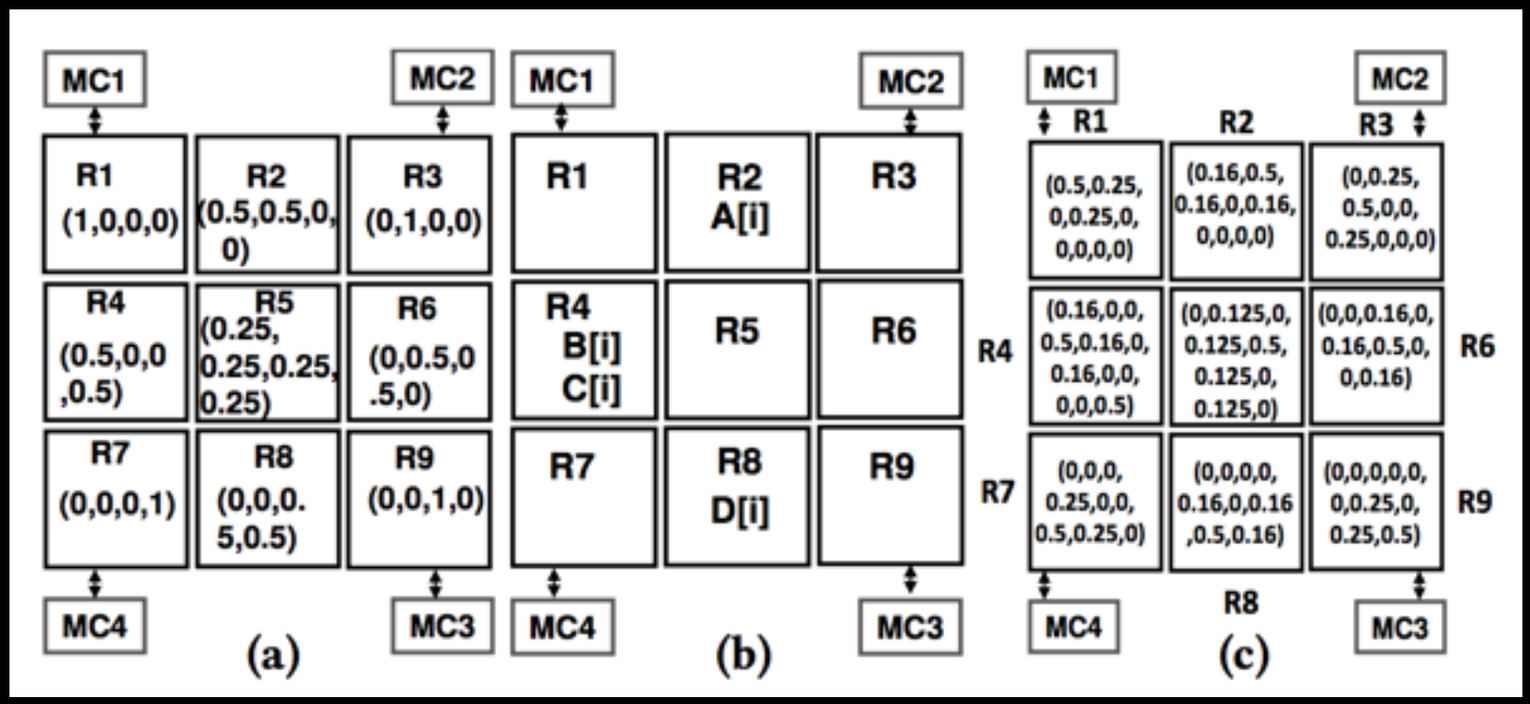
- Cache affinity of iterations (CAI) [S-NUCA only]

- Cache affinity of cores (CAC) [S-NUCA only]

- Vector Similarity Coefficient
  $\eta_m(\delta, \delta') = \Sigma_k |\delta_k - \delta'_k| / m$ $\eta = \alpha \cdot \eta_c + (1 - \alpha) \cdot \eta_m$

MAI, MAC, CAI, CAC Calculation

Example



	if misses (MC)	if hits (region)
A(i)	**MC 3**	**R2**
B(i)	**MC 1**	**R4**
C(i)	**MC 1**	**R4**
D(i)	**MC 2**	**R8**

For i = 1, 2, 3, 4 … N

A[i] = B[i] + C[i] + D[i]

MAI (0.5, 0.25, 0.25, 0)

CAI (0, 0.25, 0, 0.5, 0, 0, 0, 0.25, 0)

Computation to Core Mapping
Methodology

- **STEP 1: Compute MAI, MAC, CAI, CAC**
  Refined using Cache Miss estimation strategy via PLUTO compilation framework.

- **STEP 2**: Determine the **α** parameter

- **STEP 3**: Calculate the vector similarity **η_m(MAI, MAC)**, **η_c (CAI, CAC)** and the overal all error **η**

- **STEP 4**: Do the computation-to-core assignment based on **η**

Evaluation
Setup

- gem5 Simulator
  36 Cores(6*6), 16KB L1, 512K L2, Iteration Set Size 0.25% of iterations

- 21 multi-threaded benchmarks (OpenMP)

- Compare their result to default computation mapping

Evaluation
Speed Up

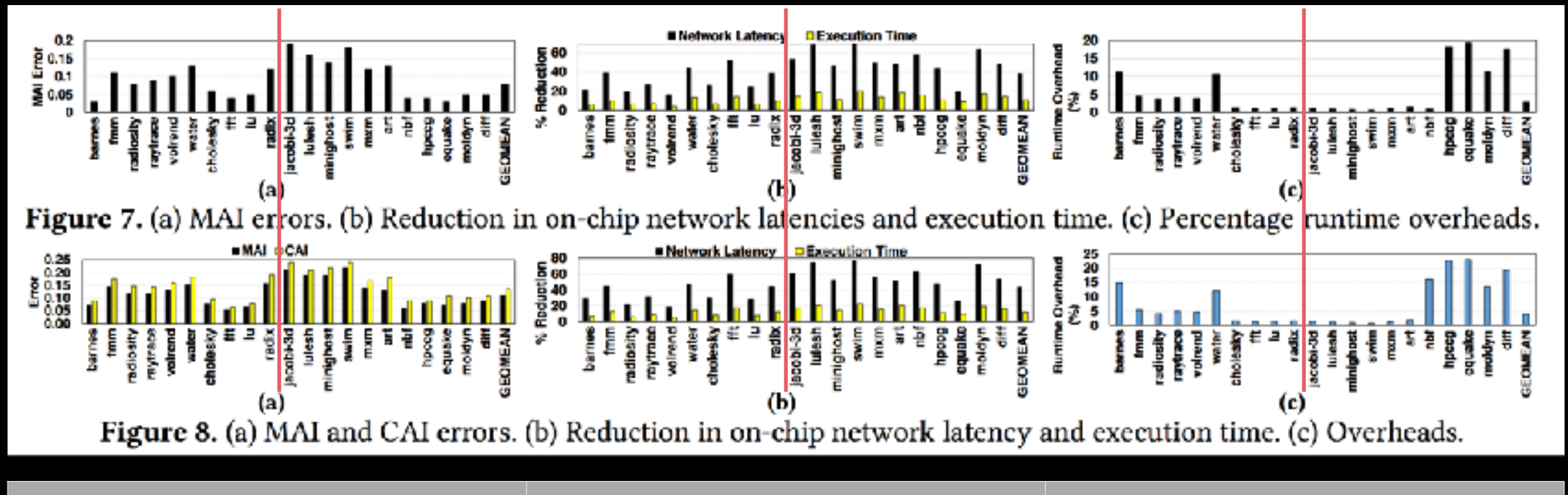


Figure 7. (a) MAI errors. (b) Reduction in on-chip network latencies and execution time. (c) Percentage runtime overheads.

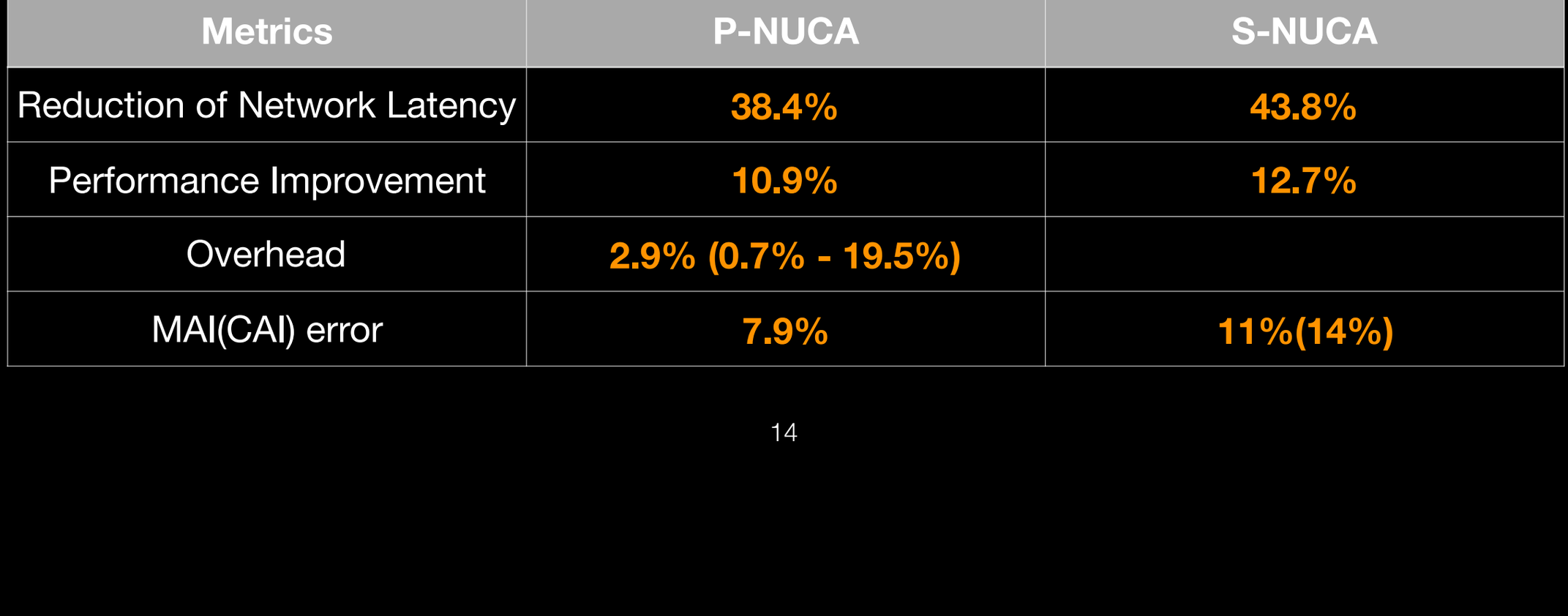


Figure 8. (a) MAI and CAI errors. (b) Reduction in on-chip network latency and execution time. (c) Overheads.

Metrics	P-NUCA	S-NUCA
Reduction of Network Latency	**38.4%**	**43.8%**
Performance Improvement	**10.9%**	**12.7%**
Overhead	**2.9% (0.7% - 19.5%)**	
MAI(CAI) error	**7.9%**	**11%(14%)**

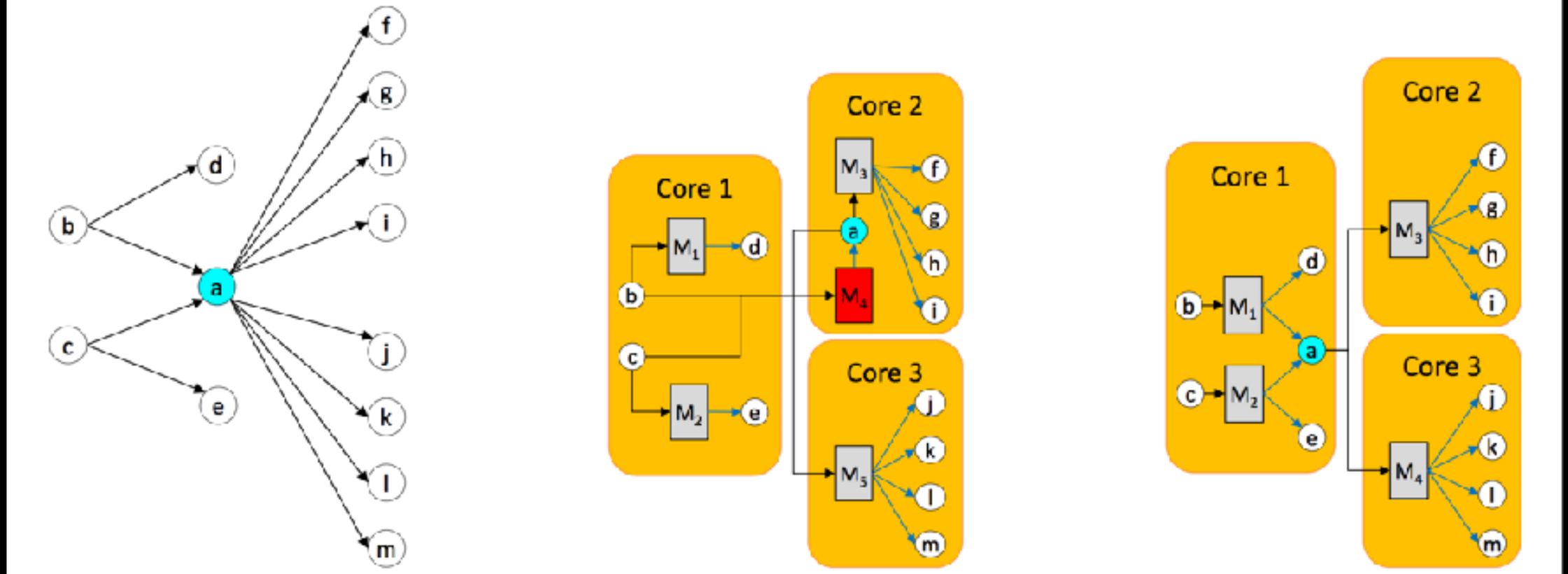
Agenda

- Important Notes
- Enhancing Computation-to-Core Assignment with Physical Location Information
- **Mapping Spiking Neural Networks onto a Manycore Neuromorphic Architecture**

Introduction

- GOAL: Reduce the Power Consumption
- METHOD: Neurons/Nodes to Cores Mapping
  - Energy cost for propagating a spike **4pJ** [Previous Work]
  - Energy cost for local update **50pJ**
- PLATFORM: Intel Loihi processor

SNN Mapping
Example



(a) Network graph

(b) Suboptimal partitioning

(c) Optimal partitioning

SNN Mapping
Mapping Algorithm

- Cut Penalty Matrix

  In which each elements $p_{\alpha\beta}$ represent the cost of assigning **compartments**(a basic building block of nodes) α and β to different cores.

SNN Mapping
Mapping Algorithm

- **STEP 1**: Initialization
  Calculate the cut-penalty matrix, put all available cores, compartments into set.

- **STEP 2**: Find an “anchor” compartment **c***
  The anchor compartment should be the one with the highest “fan-in” (most inbound edges)

- **STEP 3**: Find the first available core **k**

- **STEP 4**: Assign **c*** to **k** and co-locate all its one-hop neighbors
  The neighbor with the higher cut-penalty will be considered first

- **STEP 5**: Repeat STEP 2 to 5 until no more un-assigned compartments left

- **STEP 6**: Check if the core can support the requested number of Input Mapping
  If the core assigned cannot support the number of Input Mapping required, remove the compartments with the least cut-penalty and repeat STEP 2-5

Evaluation
Setup

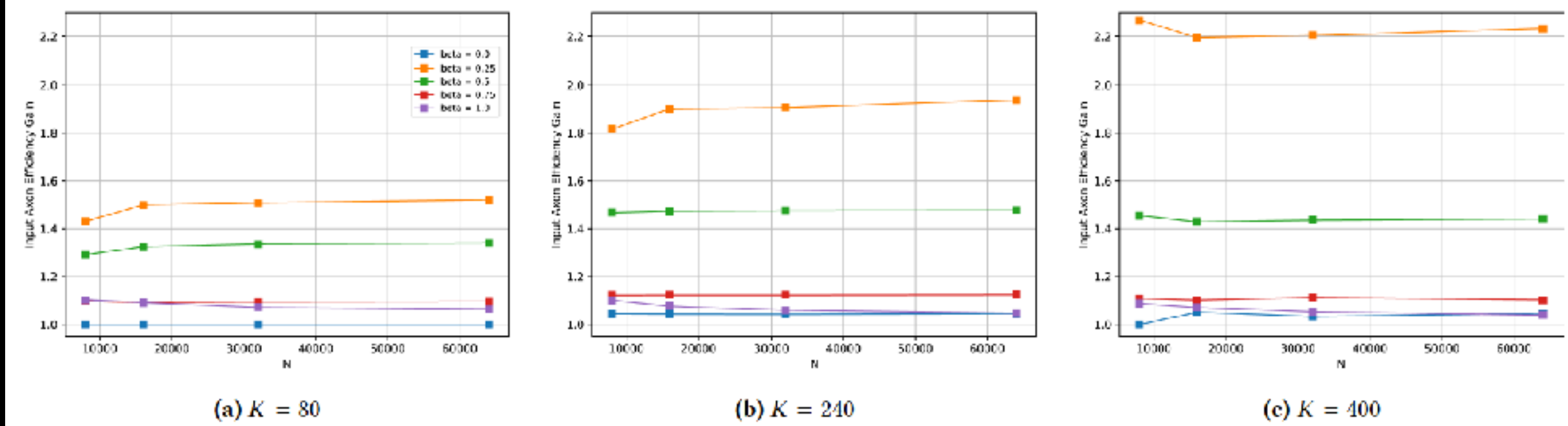
- Intel Loihi
  Announced in September 2017, contains 128 neuromorphic cores, each implements 1,024 primitive spiking neural units (compartments)

- Watts-Strogatz model with randomness parameter ß
  small-world network, rich class of graphs for evaluation

- Compare their result to default mapping

https://en.wikichip.org/wiki/intel/loihi/

Evaluation
Efficiency Gain



(a) $K = 80$ (b) $K = 240$ (c) $K = 400$

100 Graphs was generated

REFERENCE

Enhancing Computationto-Core Assignment with Physical Location Information
Orhan Kislal, Jagadish Kotra, Xulong Tang, Mahmut Taylan Kandemir, and Myoungsoo Jung
In Proceedings of 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'18).
https://doi.org/10.1145/ 3192366.3192386

Mapping Spiking Neural Networks onto a Manycore Neuromorphic Architecture
Chit-Kwan Lin, Andreas Wild, Gautham N. Chinya, Tsung-Han Lin, Mike Davies, and Hong Wang
In Proceedings of 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'18)
https://doi.org/10.1145/3192366.3192371

Loihi: A neuromorphic manycore processor with on-chip learning
Davies, Mike, Srinivasa, Narayan, Lin, Tsung-Han, Chinya, Gautham, Cao, Yongqiang, Choday, Sri Harsha , Dimou, Georgios, Joshi, Prasad, Imam, Nabil, Jain, Shweta, and others
IEEE Micro, 2018

Questions or Suggestions?

Spiking Neural Network
Introduction

- Node -> Neuron
- Edge -> Synapse
- Each neuron and synapse has its own local state
- Local state will be evolved according to local rules, which is a function of spike arrival and departure times