MODÈLE DE PROGRAMMATION POUR NOYAUX DE CALCUL IRRÉGULIERS SUR ACCÉLÉRATEUR RECONFIGURABLE DANS UN SYSTÈME DISTRIBUÉ HÉTÉROGÈNE

Erwan Lenormand
Univ. Paris-Saclay
CEA, List
erwan.lenormand@cea.fr

Thierry Goubier
Univ. Paris-Saclay
CEA, List
thierry.goubier@cea.fr

Loïc Cudennec
DGA MI
Dept. of AI
loic.cudennec@def.gouv.fr

Henri-Pierre Charles
Univ. Grenoble Alpes
CEA, List
henri-pierre.charles@cea.fr

July 9, 2021

TOWARDS MORE HETEROGENEITY IN COMPUTER SYSTEMS
From supercomputer to System on Chip

Supercomputer

© Forschungszentrum Jülich

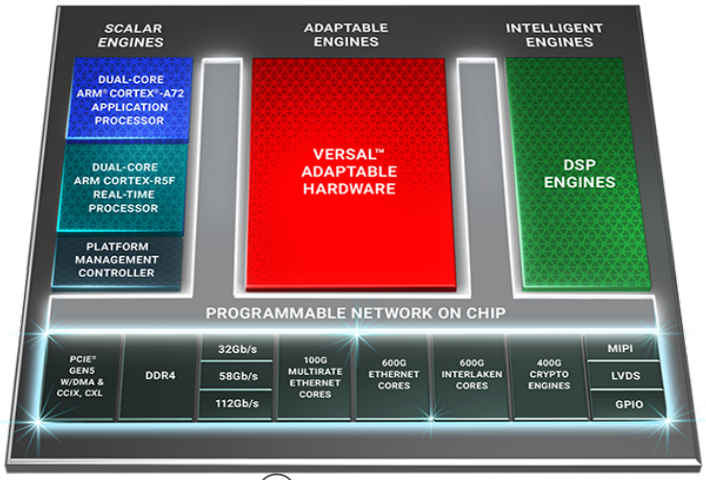
JUWELS

Most energy-efficient system in the TOP100 (Nov. 2020)

CPU – GPU

2020

System on Chip



© Xilinx

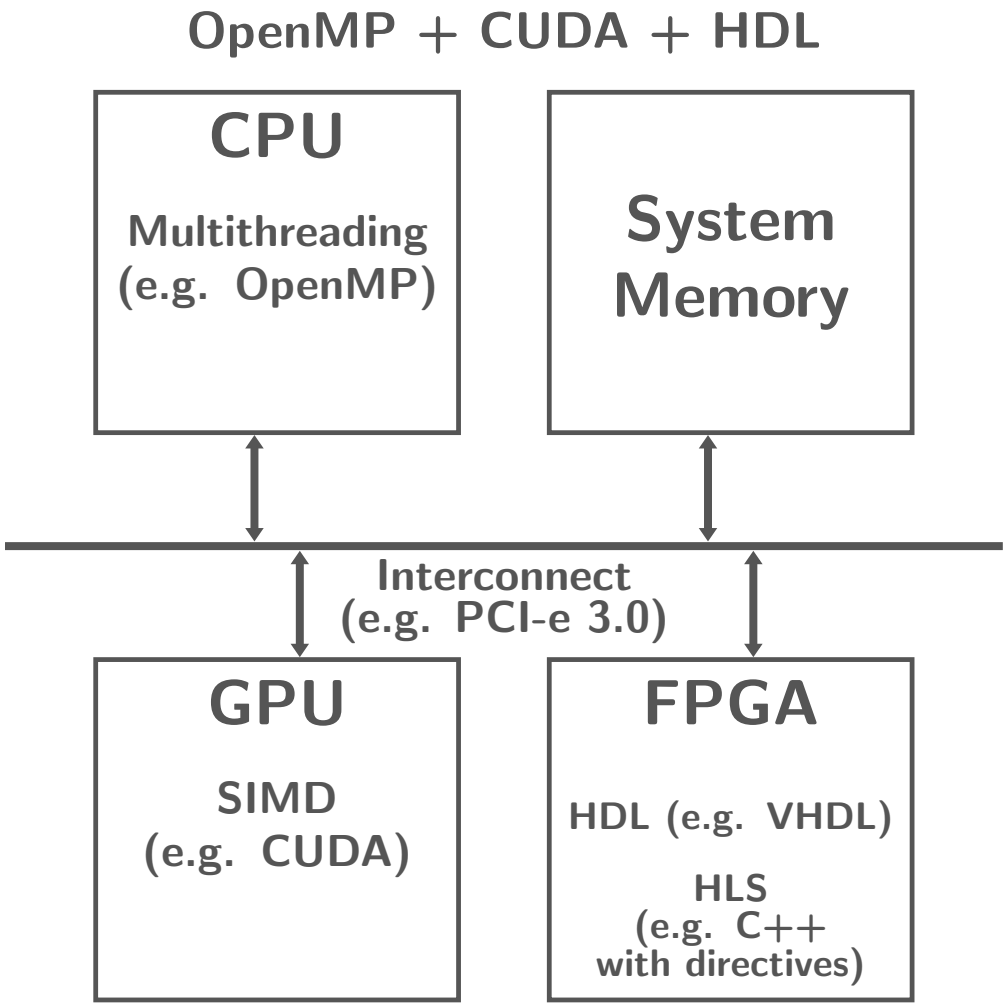
Xilinx Versal

Adaptive Compute Acceleration Platform

CPU – FPGA – AI DSP

2020

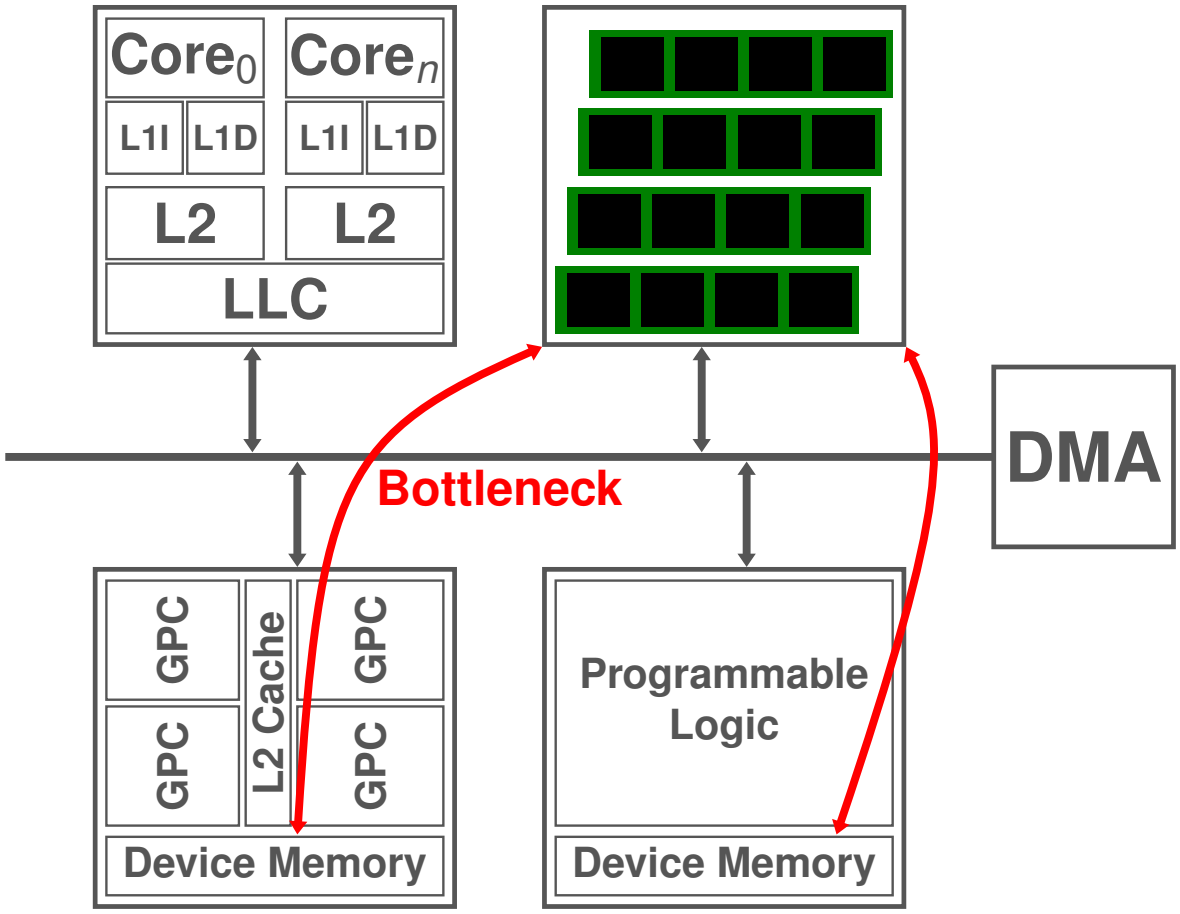
PROGRAMMING HETEROGENEOUS COMPUTING SYSTEMS
A complex and time-consuming task



Heterogeneous devices: Accelerator-specific programming models

PROGRAMMING HETEROGENEOUS COMPUTING SYSTEMS
A complex and time-consuming task

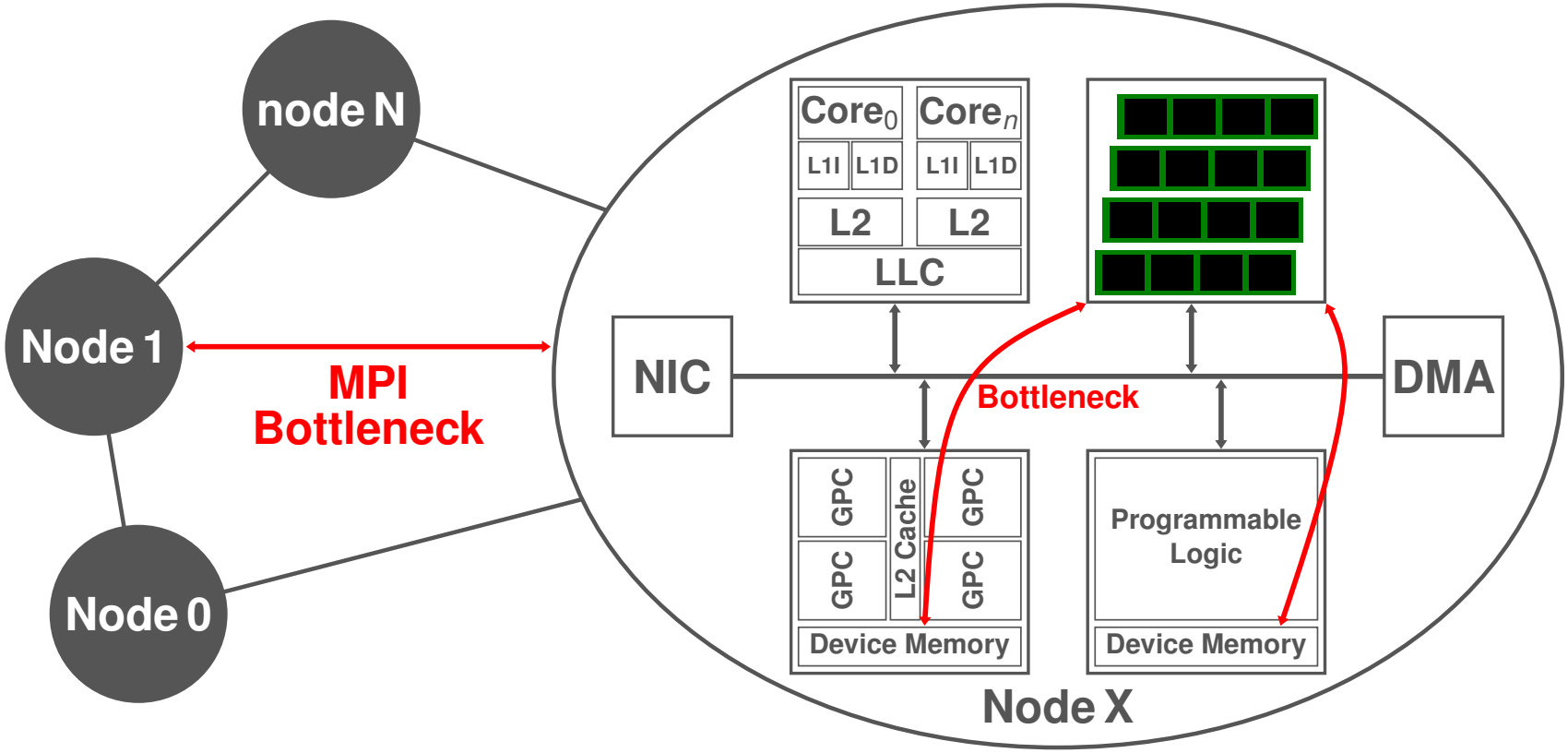
(OpenMP + CUDA + HDL)×(Memory management + Workload balancing)



Heterogeneous node: Hybrid programming model

PROGRAMMING HETEROGENEOUS COMPUTING SYSTEMS
A complex and time-consuming task

MPI×(OpenMP + CUDA + HDL)×(**Memory management** + **Workload balancing**)

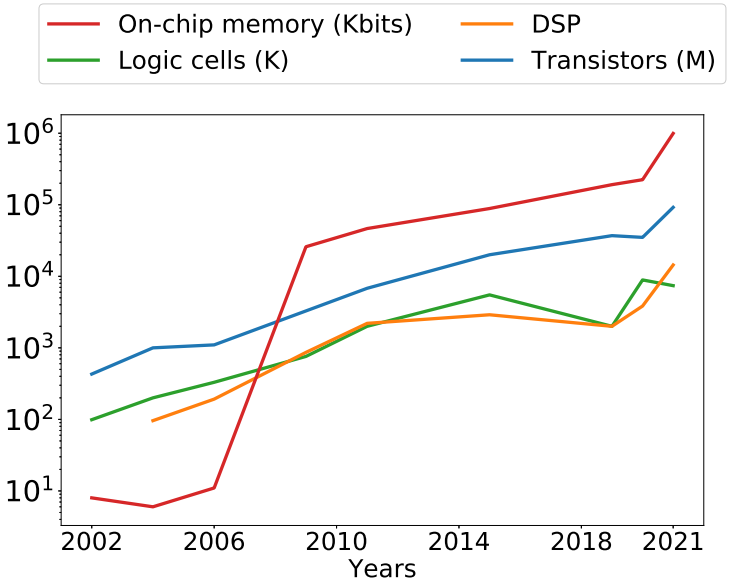


Distributed heterogeneous system: More hybrid programming model

SUITABILITY OF FPGAs FOR HIGH PERFORMANCE COMPUTING

Applications	Memory access	Current limitations
SLA: Direct (SuperLU), Iterative (Krylov)	Irregular	Area expensive FPU, Low FLOPS/access
Structured grids: Jacobi Gauss-Seidel	Repetitive, Shared	Area expensive FPU, Memory size, Synchronization, Network bandwidth
Unstructured grids: Jacobi, Graph algorithm	Repetitive, Shared	Area expensive FPU, OnChip memory size, Synchronization
Cryptographic	Regular	Logic resources, node count
Pattern detection decoding	Irregular Data dependent	OnChip memory size, design re-use

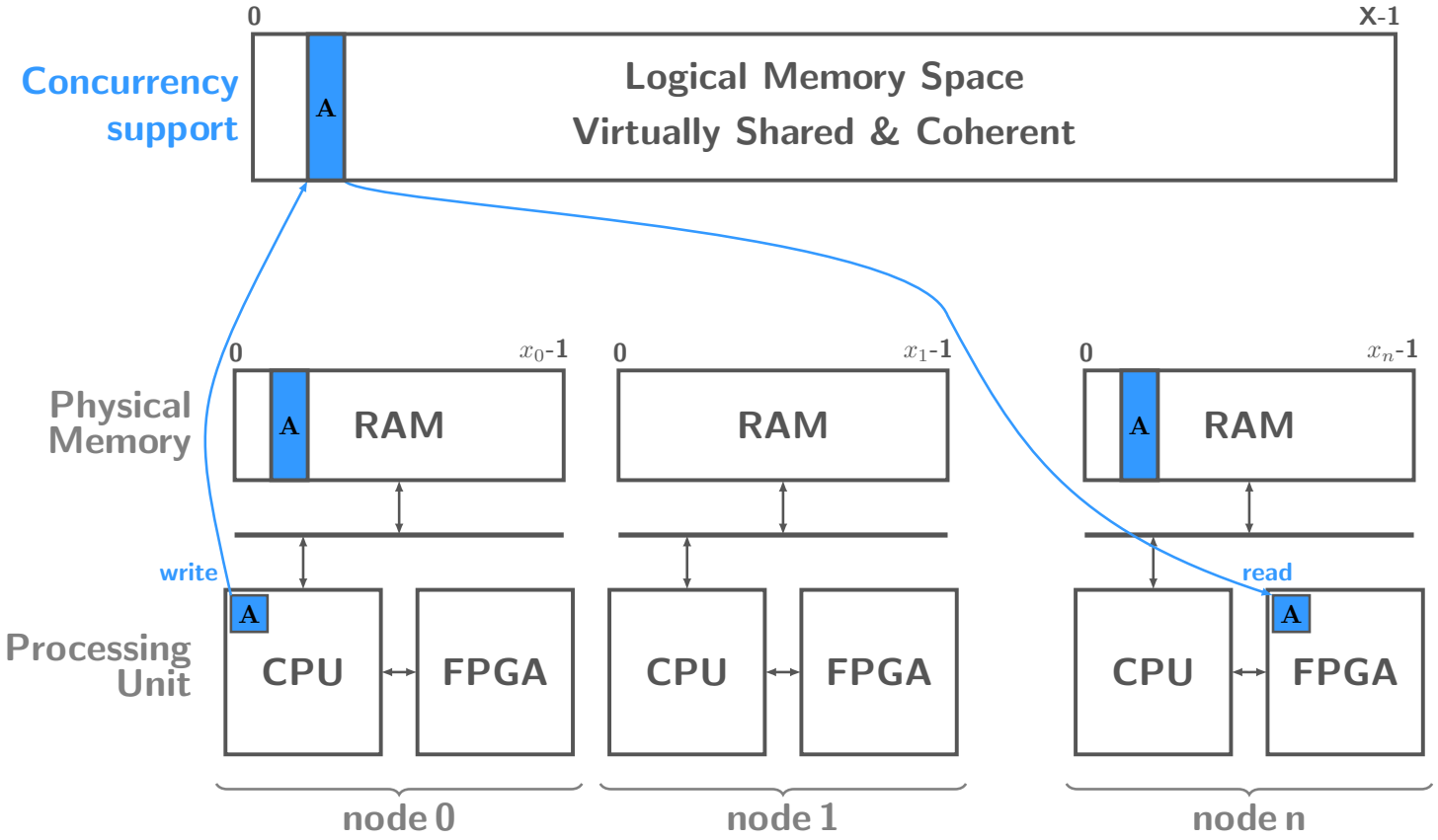
Fernando A. Escobar, Xin Chang, and Carlos Valderrama. "Suitability Analysis of FPGAs for Heterogeneous Platforms in HPC". In: *IEEE Transactions on Parallel and Distributed Systems* 27.2 (2016)



Evolution of Xilinx's FPGA capacity

WORK POSITIONING

Unifying distributed memories in a heterogeneous system with FPGA

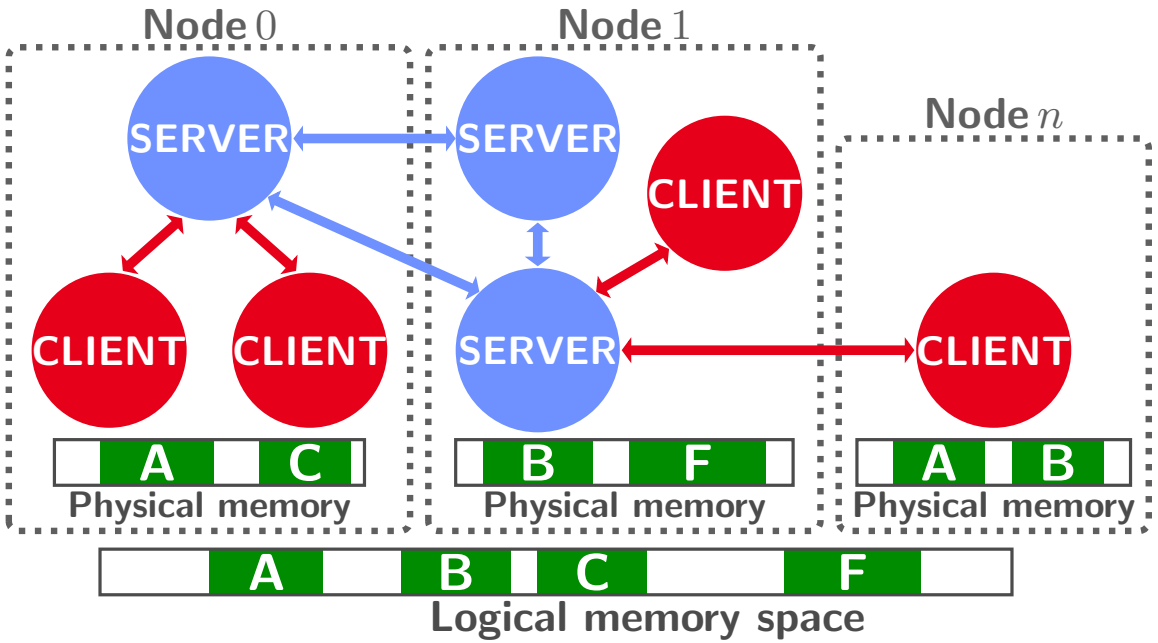


Erwan Lenormand, Loïc Cudennec, and Henri-Pierre Charles. "Unification des mémoires réparties dans un système hétérogène avec accélérateur reconfigurable : exposé de principe". In: Conférence d'informatique en Parallélisme, Architecture et Système (COMPAS). 2019

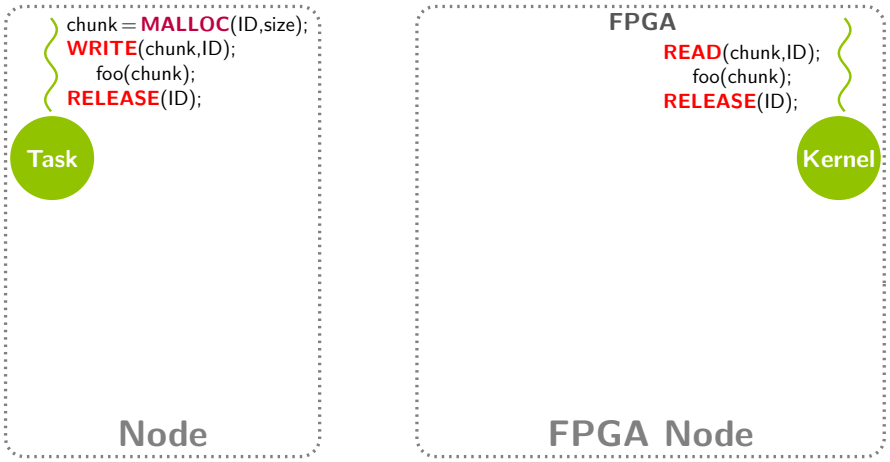
PLAN: 2 - INTEGRATION OF FPGA IN S-DSM

SOFTWARE-DISTRIBUTED SHARED MEMORY
Overview

- Loïc Cudennec. "Software-Distributed Shared Memory over heterogeneous micro-server architecture". In: *Euro-Par 2017: Parallel Processing Workshops*. 2017
- Clients run user code & Servers run API code
- Slicing large shared data into atomic chunks while locally preserving the pointer arithmetic
- Scope consistency programming model (acquire/release)



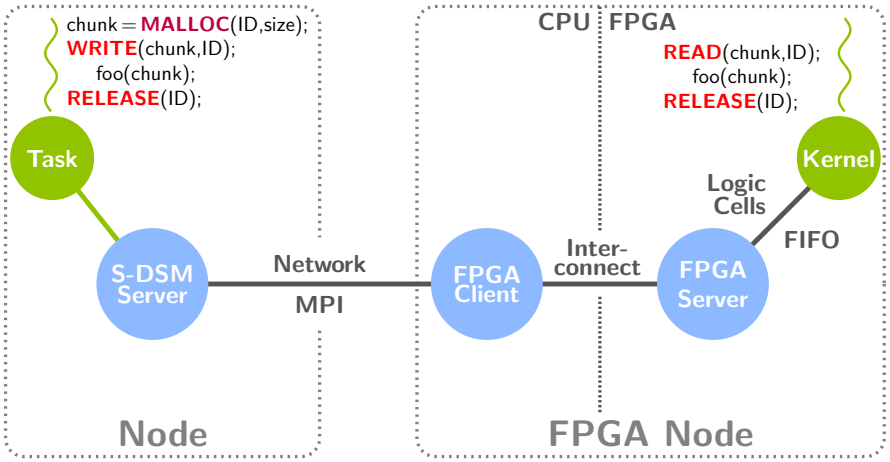
INTEGRATING FPGA INTO S-DSM
Breaking master-slave model



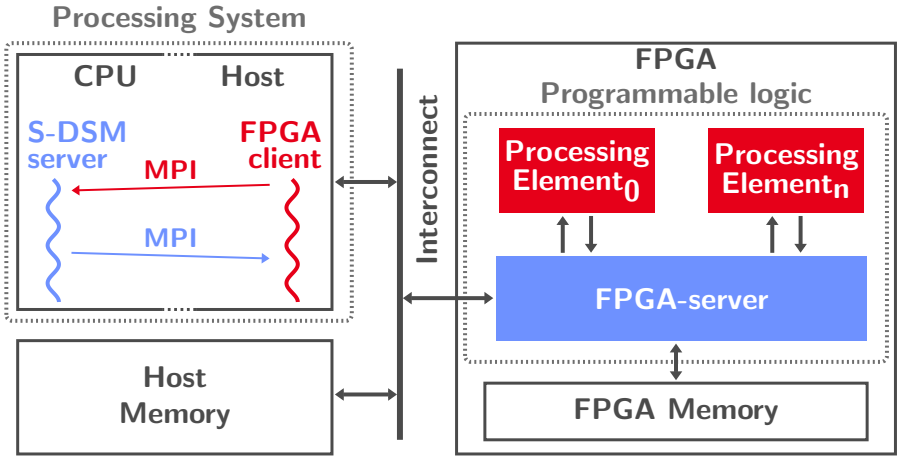
Goal

- Accelerators initiate remote memory access
- Supporting concurrent distributed data access
- Targeting irregular compute kernels

INTEGRATING FPGA INTO S-DSM
Breaking master-slave model



Integration topology



Overview of the reconfigurable accelerator integration architecture

Goal

- Accelerators initiate remote memory access
- Supporting concurrent distributed data access
- Targeting irregular compute kernels

PLAN: 3 - PROGRAMMING MODEL FOR IRREGULAR COMPUTE KERNELS

PROGRAMMING MODEL
Main idea

S-DSM may involve high data access latency	$\neq$	Accelerator need quick data access
Virtually shared memory is convenient for programming	$\rightarrow$	Enable to handle rich-pointer data structures
Hardware compute kernels are implemented as pipeline	$\rightarrow$	Dataflow is convenient for programming FPGAs

PROGRAMMING MODEL
Main idea

S-DSM may involve high data access latency	$\neq$	Accelerator need quick data access
Virtually shared memory is convenient for programming	$\rightarrow$	Enable to handle rich-pointer data structures
Hardware compute kernels are implemented as pipeline	$\rightarrow$	Dataflow is convenient for programming FPGAs

Chunks structure	$\rightarrow$	Pointer arithmetic in a contiguous space Irregular data structure abstraction
Dataflow model	$\rightarrow$	Hiding memory access latency
Data management	$\rightarrow$	Increasing data locality

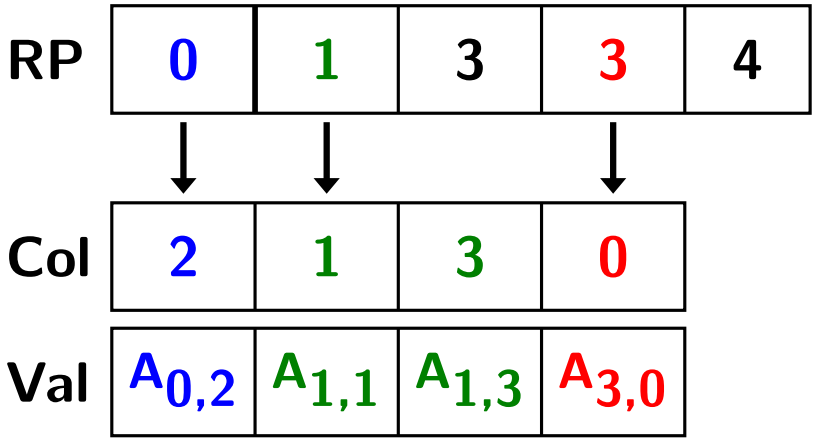
Easy transition between two models

Compute kernels generate data access pattern as chunk ID sequences, then access data through FIFOs

CHUNKS STRUCTURE
Compressed Sparse Row Format adaptation

$$\begin{bmatrix} 0 & 0 & A_{0,2} & 0 \\ 0 & A_{1,1} & 0 & A_{1,3} \\ 0 & 0 & 0 & 0 \\ A_{3,0} & 0 & 0 & 0 \end{bmatrix}$$

Dense format



Compressed sparse row format

Chunk adaptation

- Abstracting irregularity through metadata
- Improving locality by grouping data

ID	count	
0	1	$(2,A_{0,2})$
1	2	$(1,A_{1,1})$ $(3,A_{1,3})$
3	1	$(0,A_{3,0})$

Chunk Compressed sparse row format

SPARSE LINEAR ALGEBRA PROCESSING
from scalar processing to dataflow processing

```
for(row = 0; row < m; ++row)
  for(i = rp[row]; i < rp[row+1]; ++i)
    f(col[i],val[i]);
```

Code 1: CSR traversal

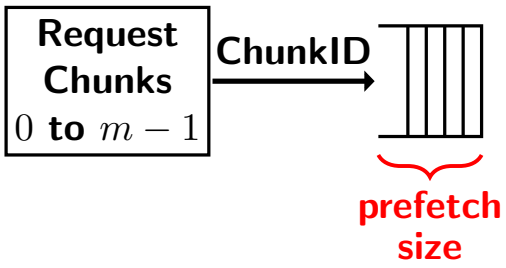
SPARSE LINEAR ALGEBRA PROCESSING
from scalar processing to dataflow processing

```
for(row=0; row < m; ++row)
{
  chunk = _SDSM_LOOKUP(row);




}
```

Code 2: Chunk CSR traversal



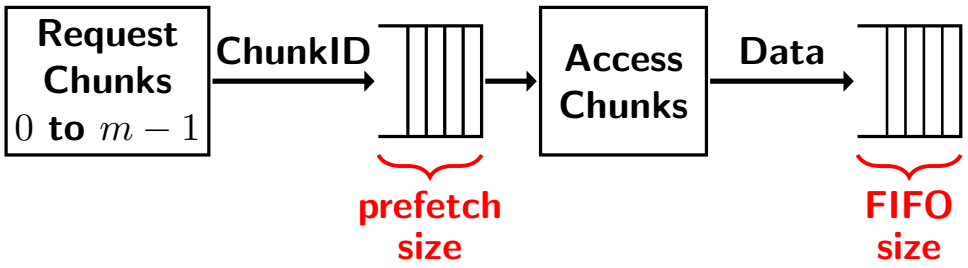
SPARSE LINEAR ALGEBRA PROCESSING
from scalar processing to dataflow processing

```
for(row=0; row < m; ++row)
{
  chunk = _SDSM_LOOKUP(row);
  _SDSM_READ(chunk);



}
```

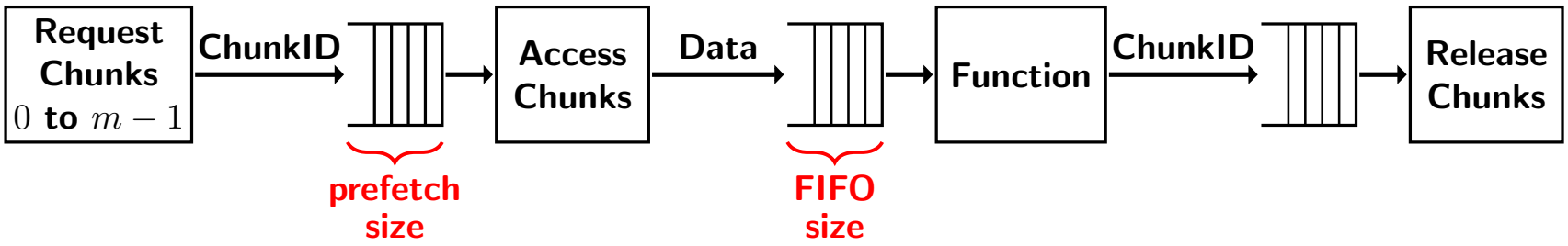
Code 3: Chunk CSR traversal



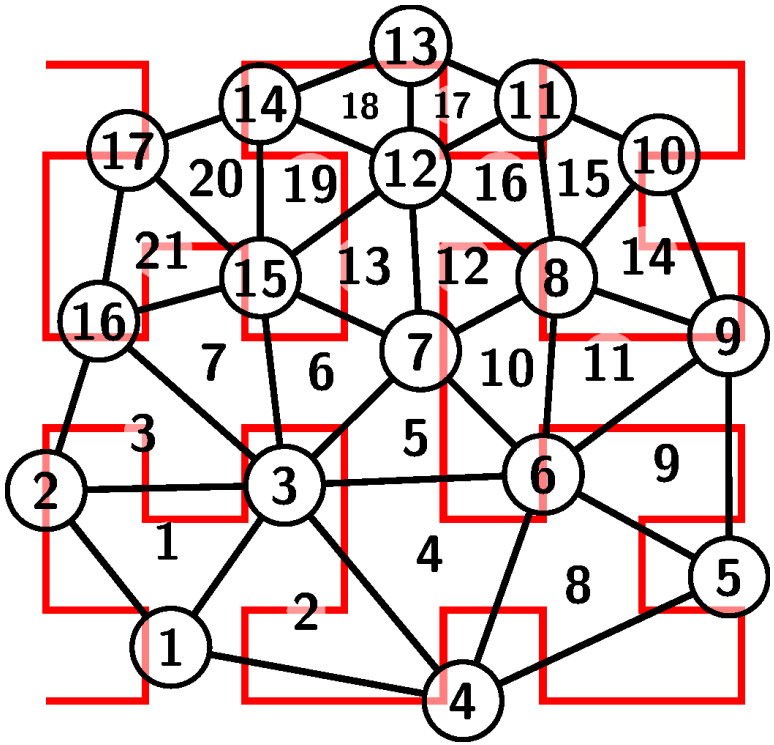
SPARSE LINEAR ALGEBRA PROCESSING
from scalar processing to dataflow processing

```
for(row=0; row < m; ++row)
{
  chunk = _SDSM_LOOKUP(row);
  _SDSM_READ(chunk);
  for(i=0;i < chunk->size/2; i+=2)
    f(chunk->data[i],chunk->data[i+1]);
  _SDSM_RELEASE(chunk);
}
```

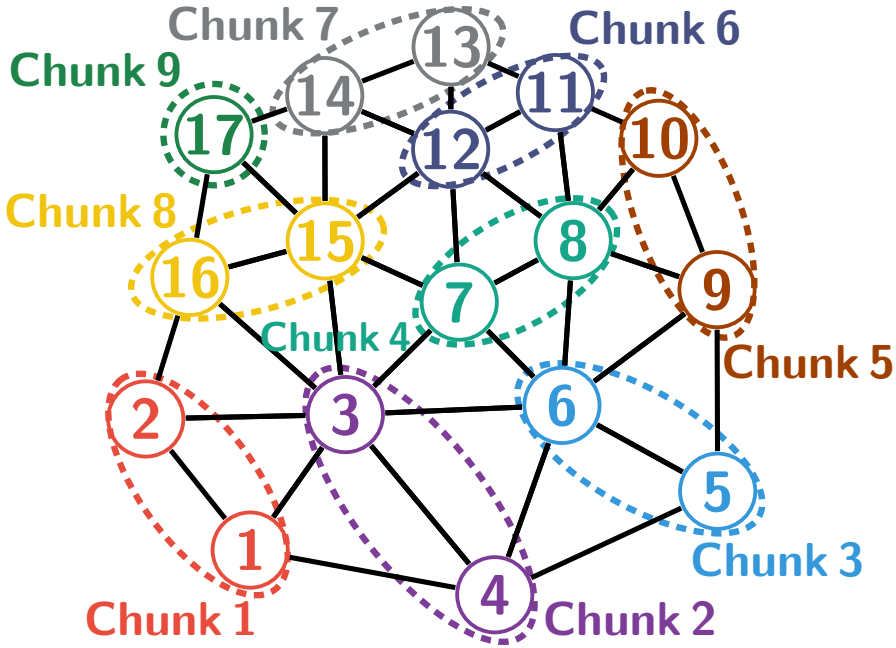
Code 4: Chunk CSR traversal



CHUNKS STRUCTURE
Mesh adaptation



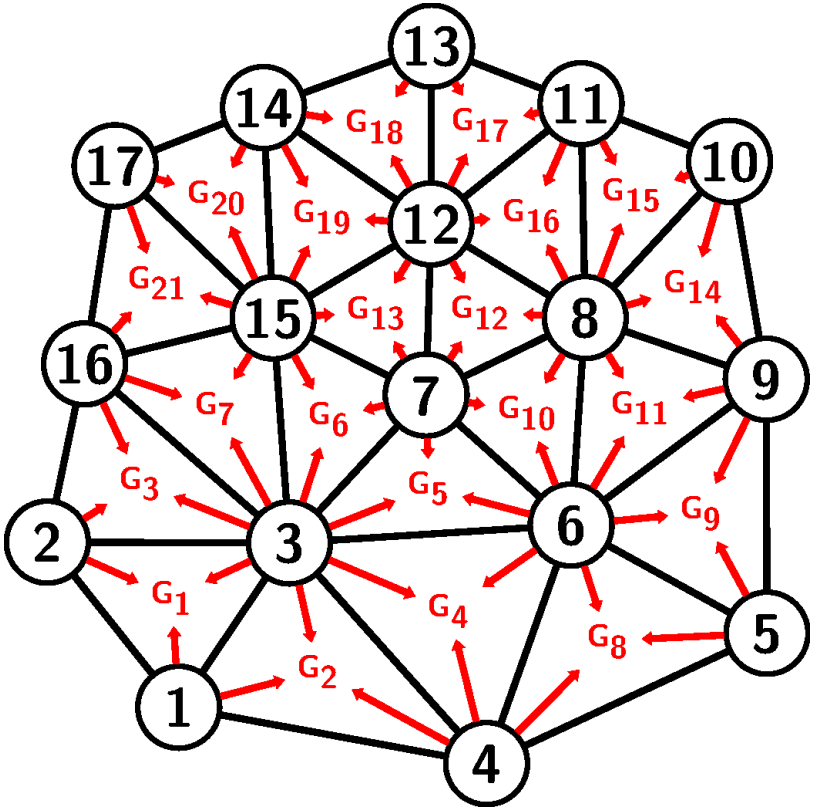
Hilbert curve over a 2D unstructured mesh.

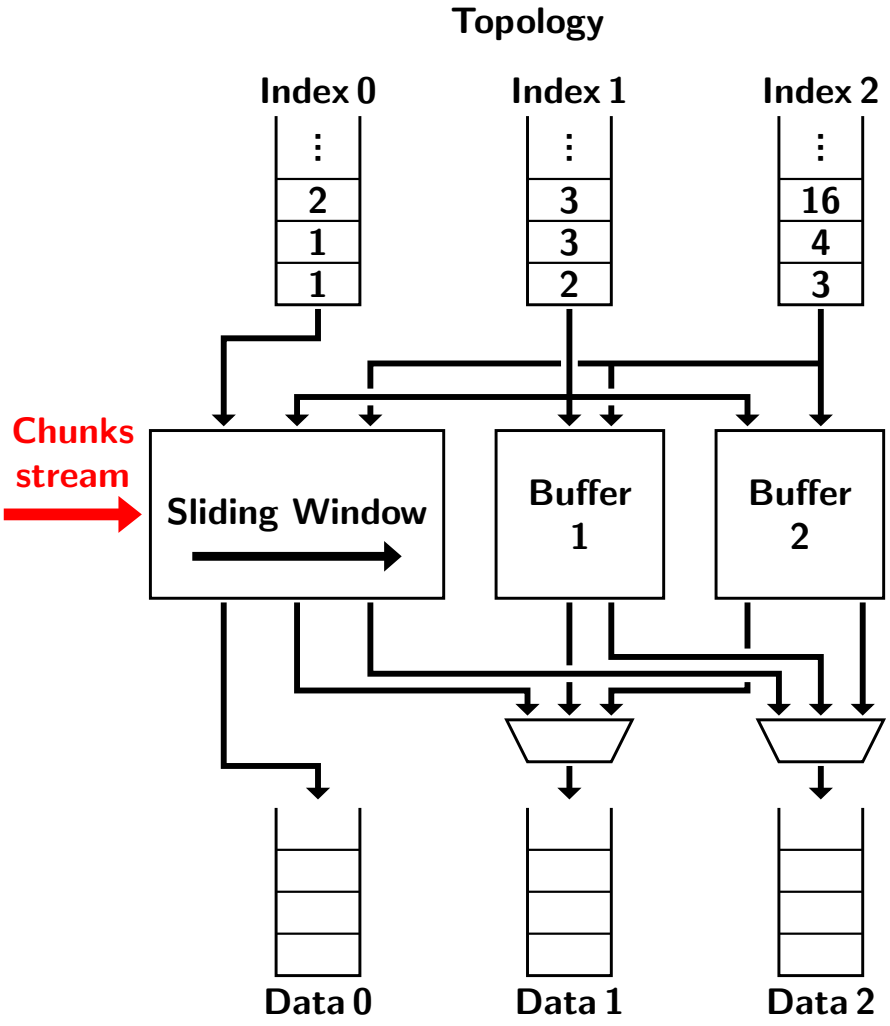


Partitioning of the mesh into chunks of two elements.

FINITE ELEMENT METHODE

Traverse mesh through a sliding window

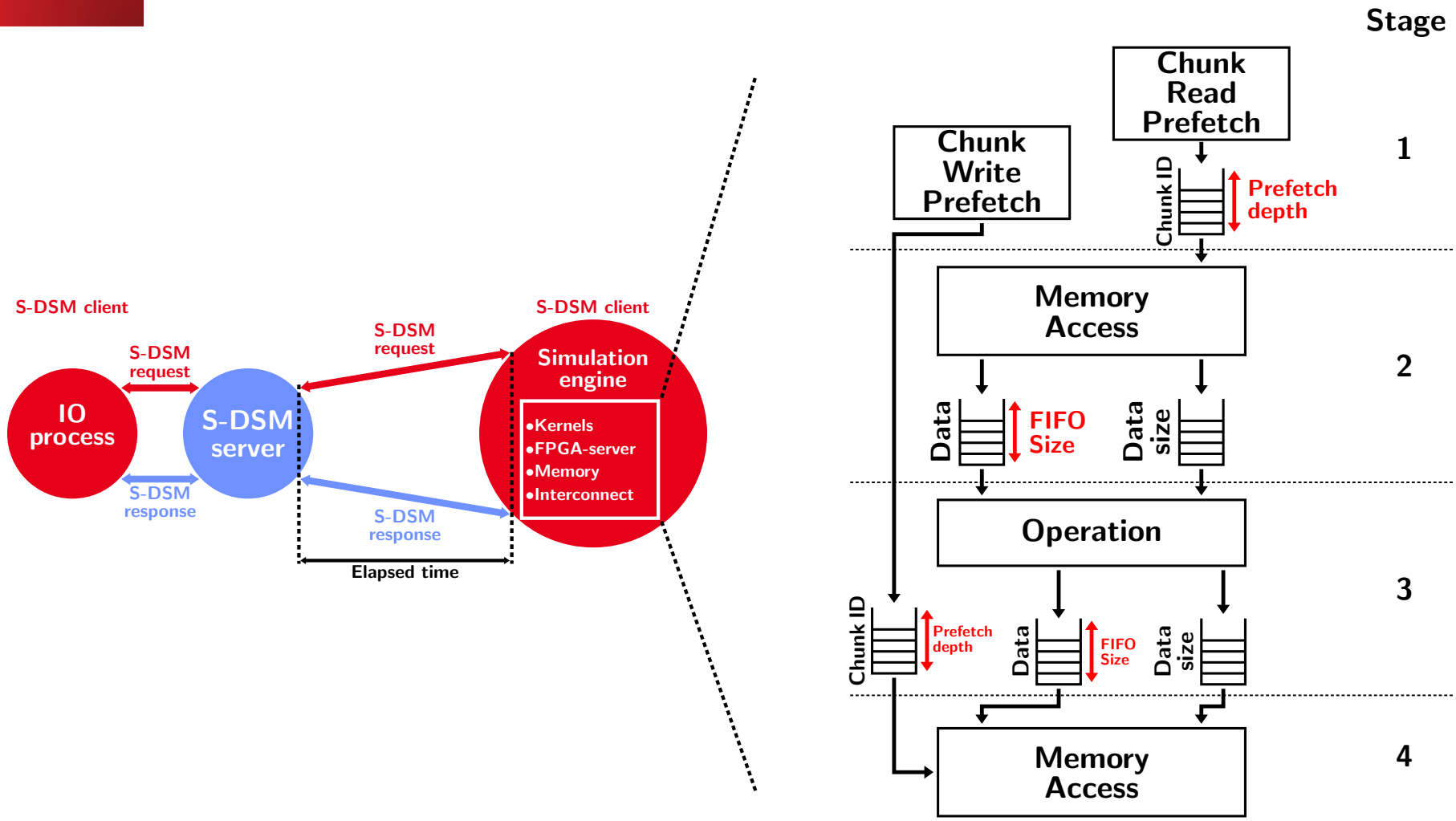




PLAN: 4 - PROGRAMMING MODEL VALIDATION

1. Context & motivation
2. Integration of FPGA in S-DSM
3. Programming model for irregular compute kernels
4. Programming model validation
   - Simulation methodology
   - Case study I: Tsunami simulation
   - Case study II: Sparse General Matrix-Matrix Multiplication
5. Conclusion & Future work

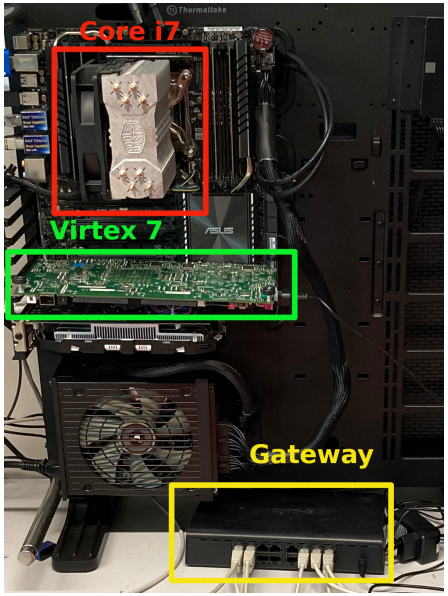
SIMULATION TOOL ARCHITECTURE OVERVIEW



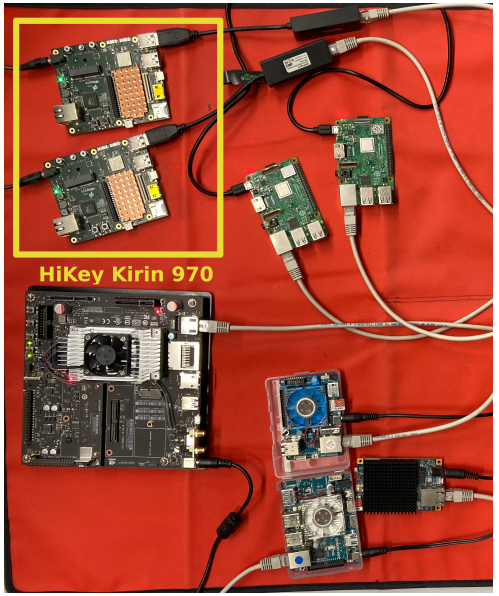
Erwan Lenormand et al. "A combined fast/cycle accurate simulation tool for reconfigurable accelerator evaluation: application to distributed data management". In: *2020 International Workshop on Rapid System Prototyping (RSP)*. 2020

SIMULATION PLATFORM

- In-house micro-cluster of heterogeneous development boards and a high-performance gateway



Main node



Heterogeneous nodes

Node	Processor	Latency		
		MPI ping-pong	S-DSM read	S-DSM write
Main	Intel Core i7 6800K	≈ 1 µs	≈ 383 µs	≈ 207 µs
HiKey Kirin 970	Arm Cortex A73/A53	≈ 313 µs	≈ 1311 µs	≈ 533 µs

Simulation platform description

PLAN: 4 - PROGRAMMING MODEL VALIDATION

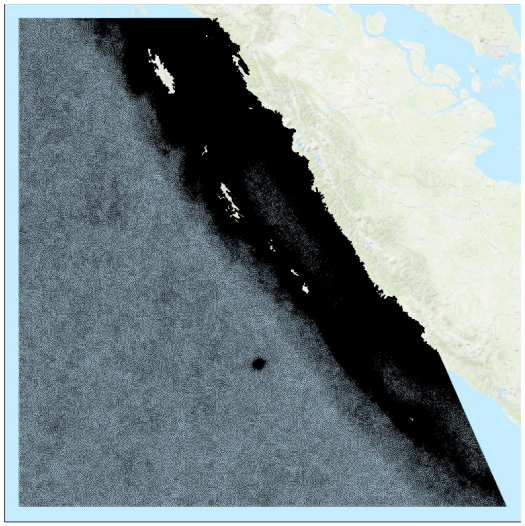
4 Programming model validation
- Simulation methodology
- Case study I: Tsunami simulation
- Case study II: Sparse General Matrix-Matrix Multiplication

CASE STUDY I: TSUNAMI SIMULATION
Compute kernel

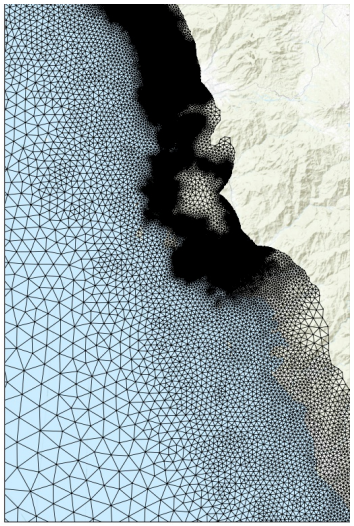
```
void compute_gradient(const size_t nTriangles,
                      const uint32_t ** elem2d,
                      const float * ssh,
                      const float ** bafu,
                      float * grad)
{
    for(size_t i = 0; i < nTriangles; ++i)
        grad[i] =  bafu[i][0] * ssh[elem2d[i][0]]
                 + bafu[i][1] * ssh[elem2d[i][1]]
                 + bafu[i][2] * ssh[elem2d[i][2]];
}
```

Code 5: Reference C code used for modeling the compute kernel. Adapted from the Fortran source code TsunAWI (https://gitlab.awi.de/tsunawi/tsunawi).

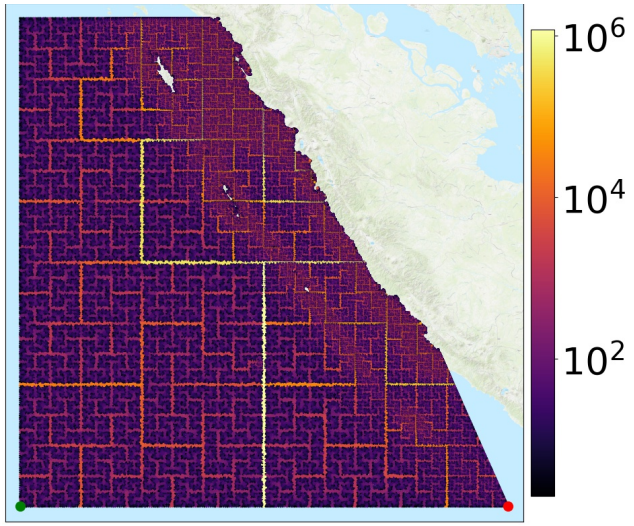
CASE STUDY I: TSUNAMI SIMULATION
Meshes dataset



Padang fine-grained mesh topology.



Zoom on coastal area.

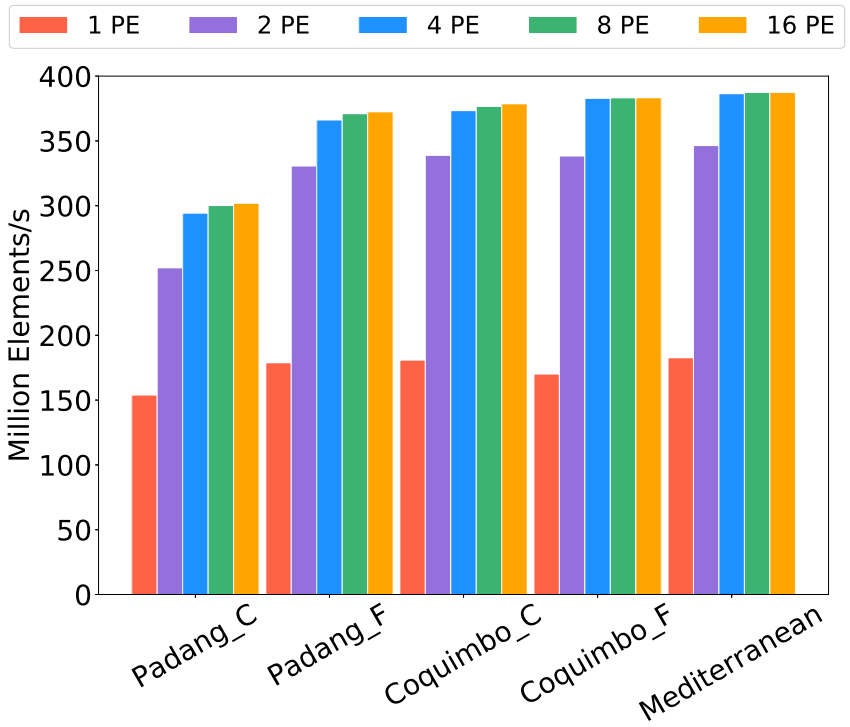


Logarithmic colormap of the maximum index difference of the neighbouring node.

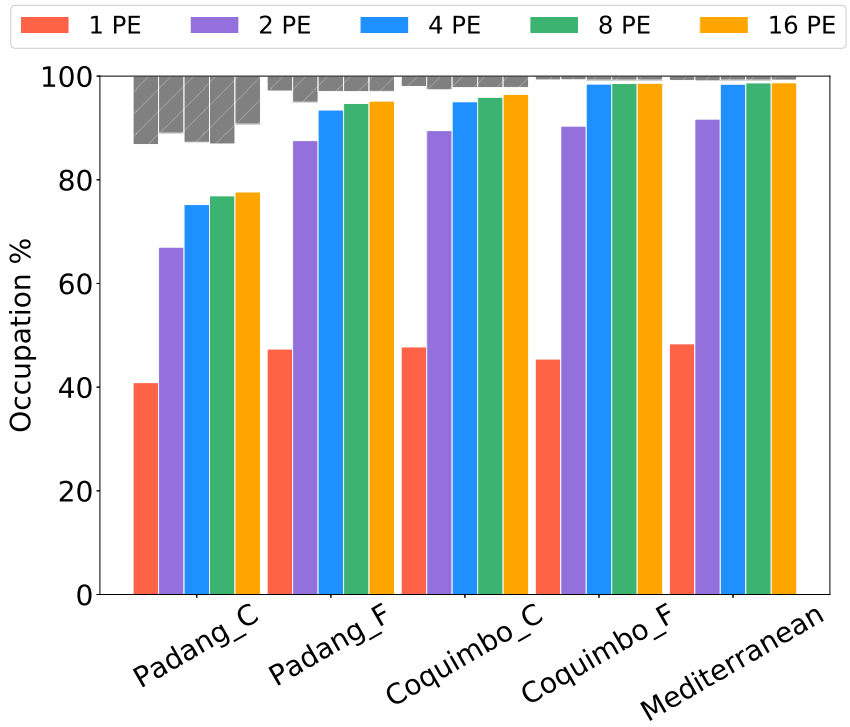
Name	# Elements	# Vertices	Size
Padang_C	460 119	231 586	14 Mo
Padang_F	2 470 345	1 242 653	74 Mo
Coquibo_C	3 396 755	1 709 506	102 Mo
Coquimbo_F	9 762 027	4 887 927	293 Mo
Mediterranean	9 917 645	4 999 404	298 Mo

Meshes dataset used for tsunami simulation.

CASE STUDY I: TSUNAMI SIMULATION
Performance according to the number of processing elements

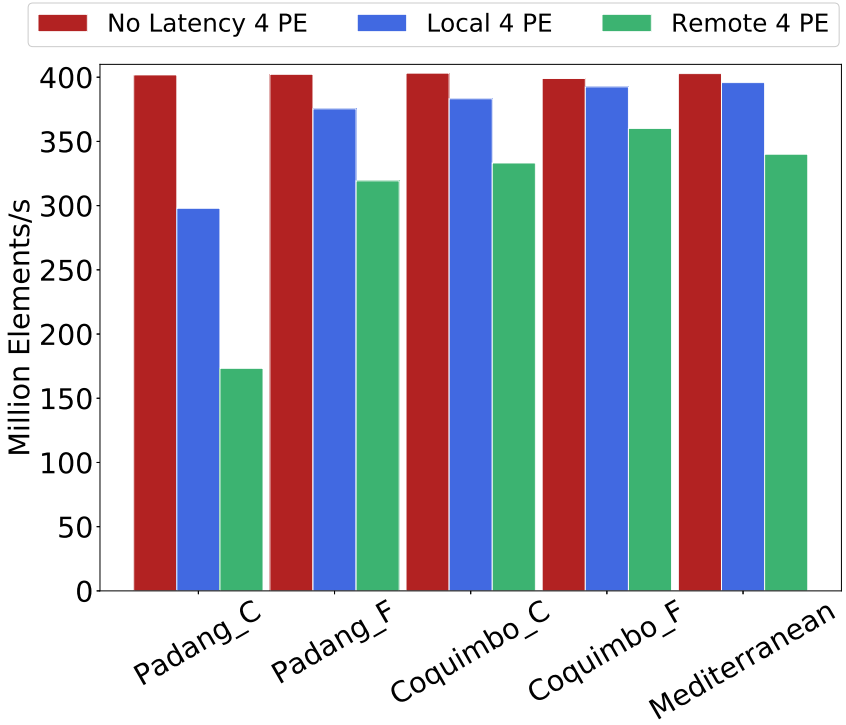


Computation speed in MElements/s according to the number of processing elements.

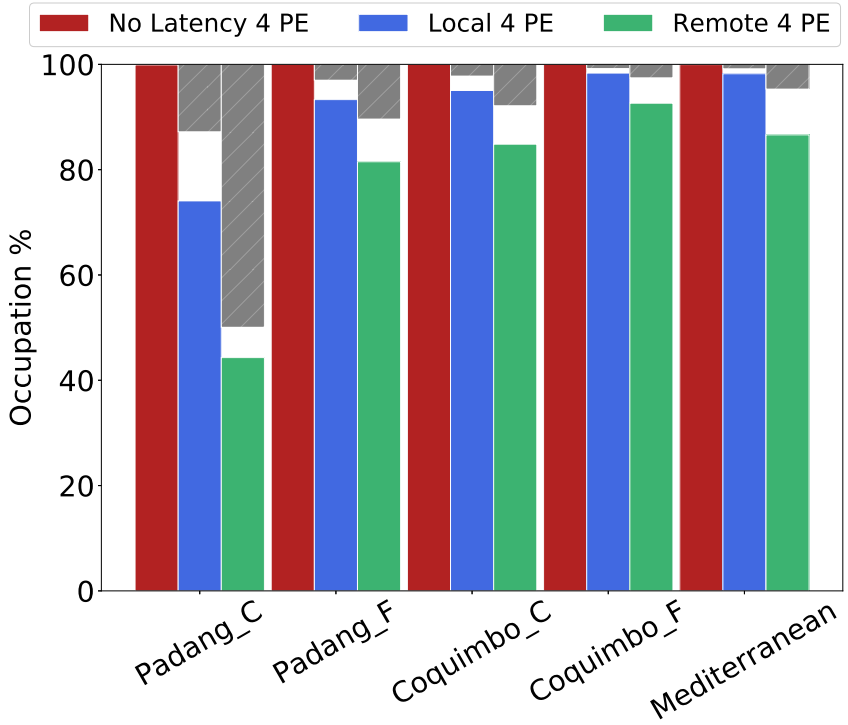


Memory controller activity (occupancy percentage) according to the number of processing elements.

CASE STUDY I: TSUNAMI SIMULATION
Performance according the system topology



Computation speed in MElements/s according the system topology.



Memory controller activity (occupancy percentage) according the system topology.

PLAN: 4 - PROGRAMMING MODEL VALIDATION

4. Programming model validation
   - Simulation methodology
   - Case study I: Tsunami simulation
   - Case study II: Sparse General Matrix-Matrix Multiplication

CASE STUDY II: SPGEMM
Compute kernel

Algorithm Gustavson's row-wise matrix multiplication algorithm.

$A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $C \in \mathbb{R}^{m \times p}$

for i = 0,...,m-1 **do**

For Each $A_{i,k}$ in row $A_{i,*}$ **do**

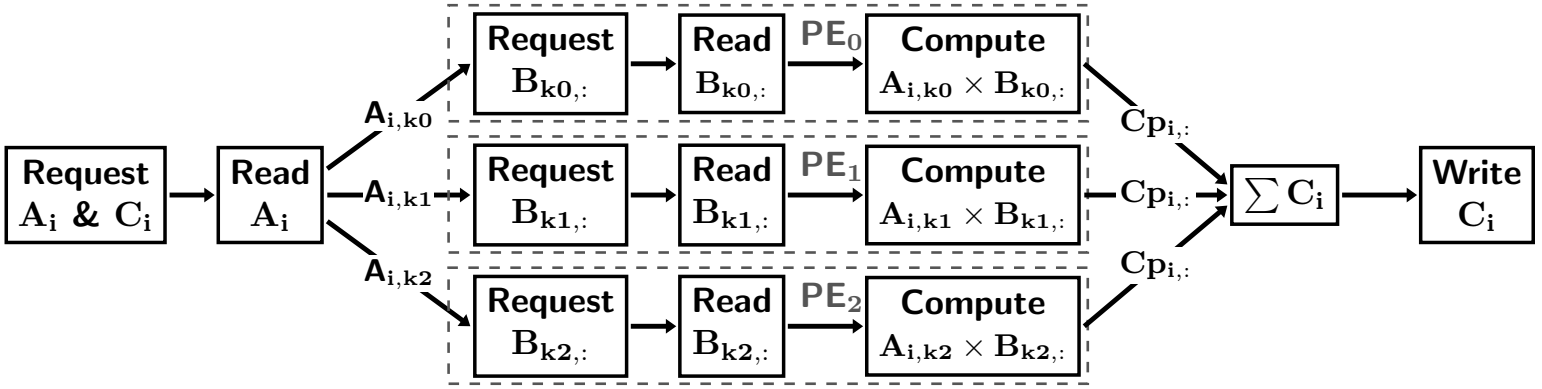
For Each $B_{k,j}$ in row $B_{k,*}$ **do**

if $C_{i,j} == 0$ **then**

$C_{i,j} = A_{i,k} \times B_{k,j}$

else

$C_{i,j} \mathrel{+}= A_{i,k} \times B_{k,j}$

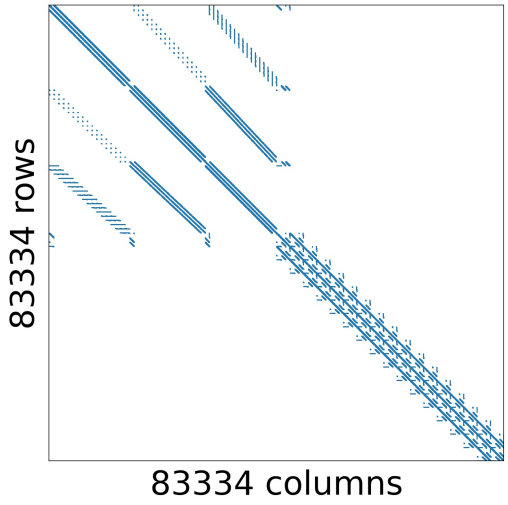


Dataflow overview of the compute kernel with 3 PEs.

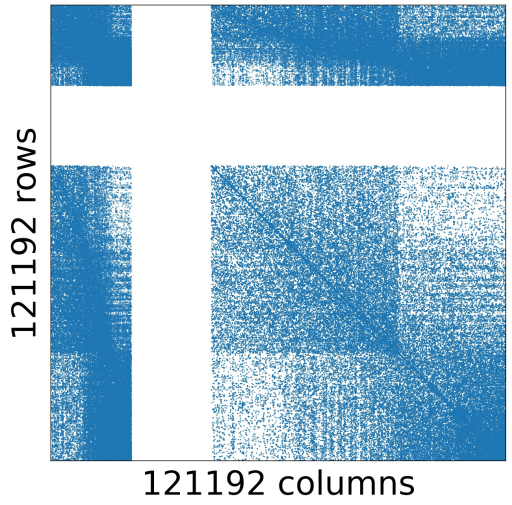
CASE STUDY II: SPGEMM

Matrix dataset

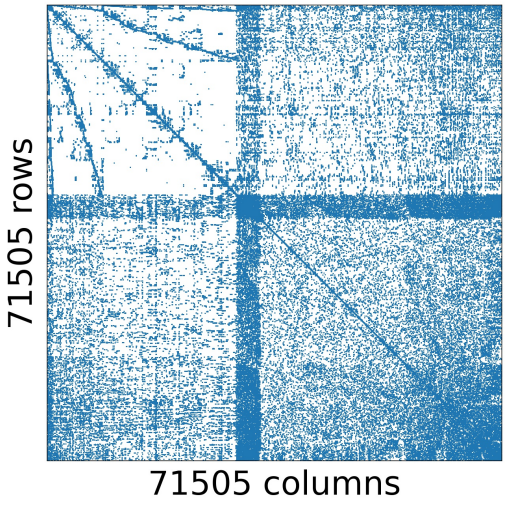
consph | 6.0M NNZ | DENS 0.087%



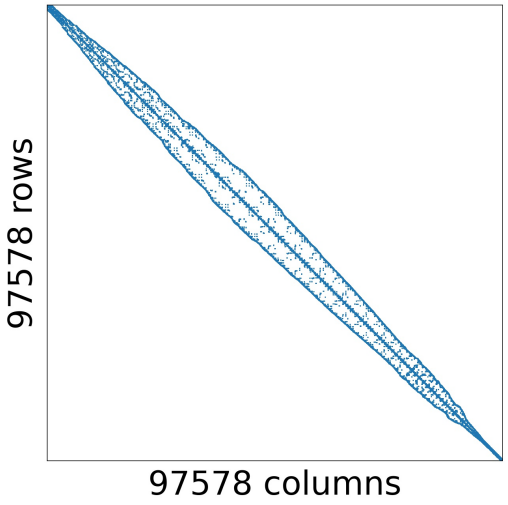
cop20k_A | 2.6M NNZ | DENS 0.018%



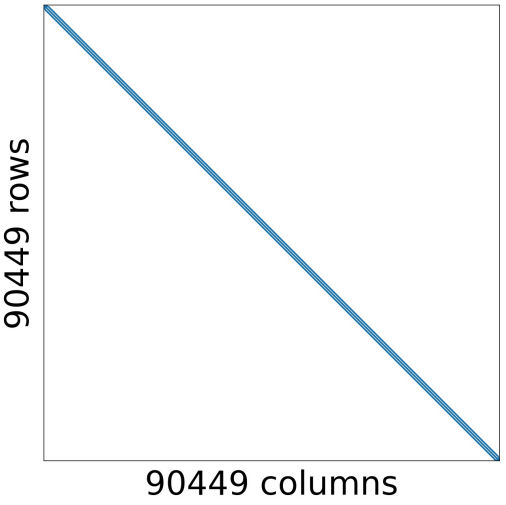
F2 | 5.3M NNZ | DENS 0.104%



m_t1 | 9.8M NNZ | DENS 0.102%

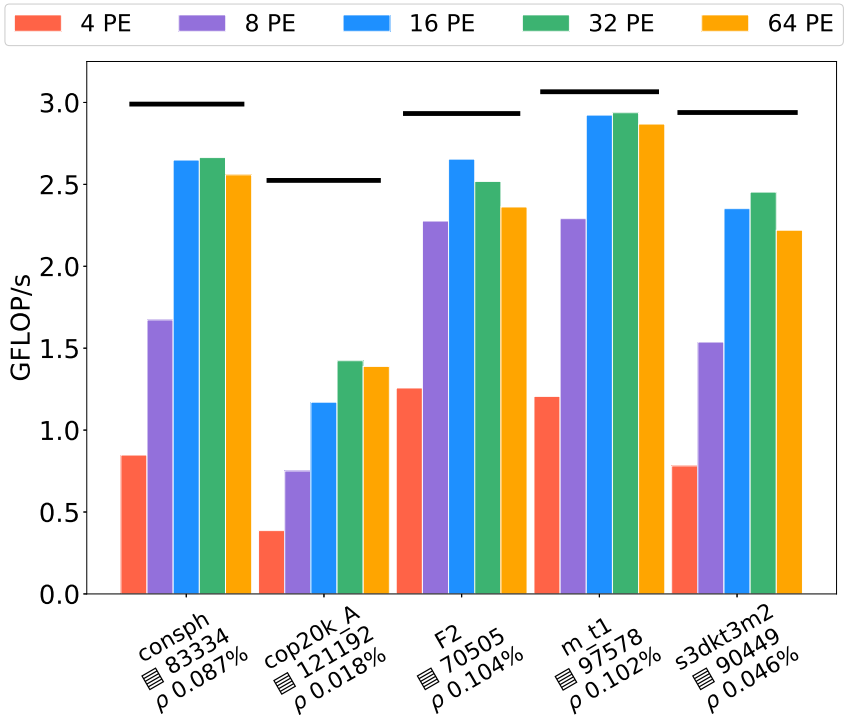


s3dkt3m2 | 3.8M NNZ | DENS 0.046%

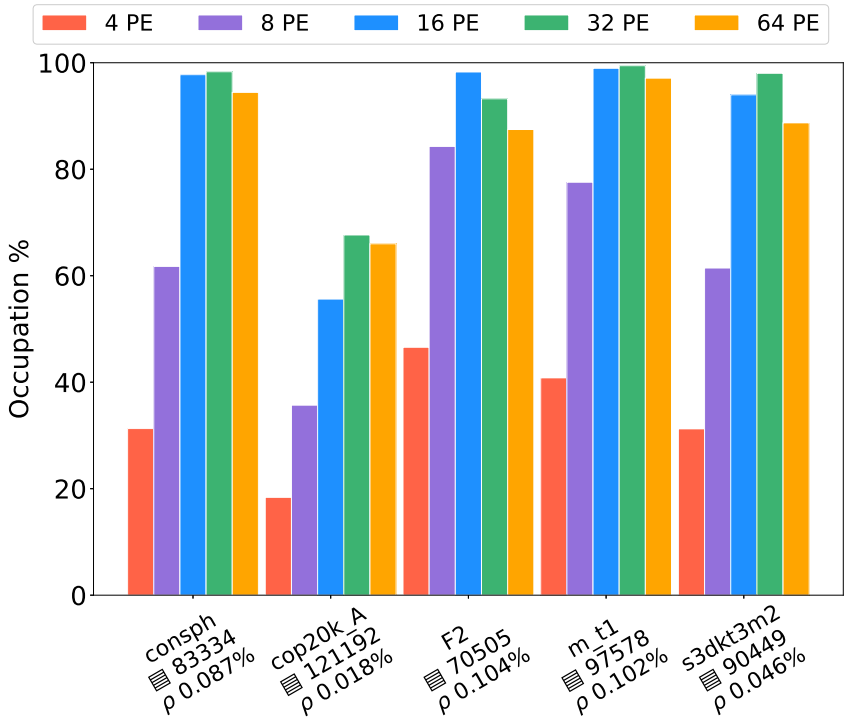


Square matrix dataset from the University of Florida Sparse Matrix Collection

CASE STUDY II: SPGEMM
Performance according to the number of processing elements

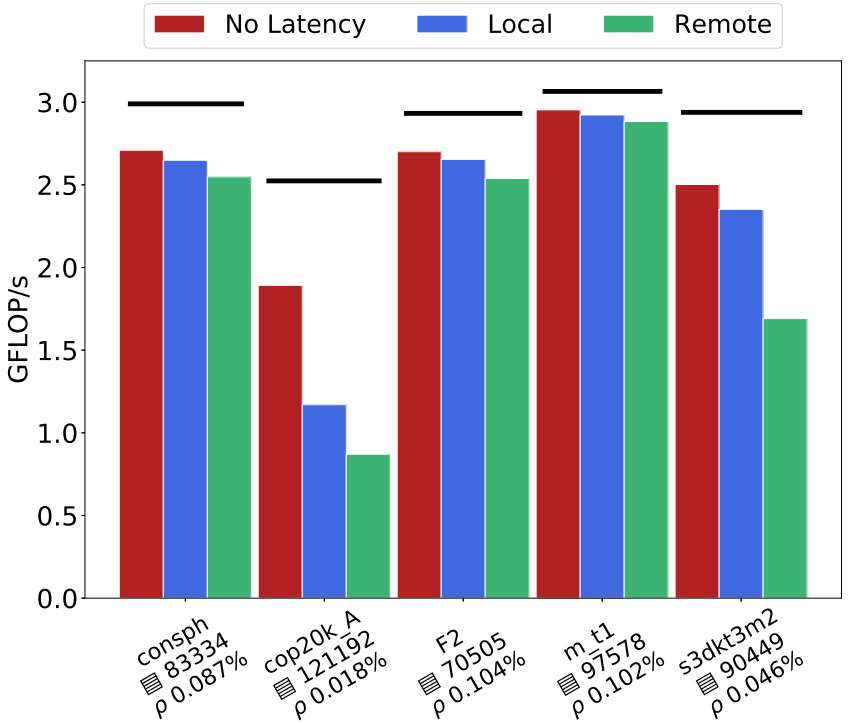


Computation speed in GFLOP/s according to the number of processing elements.

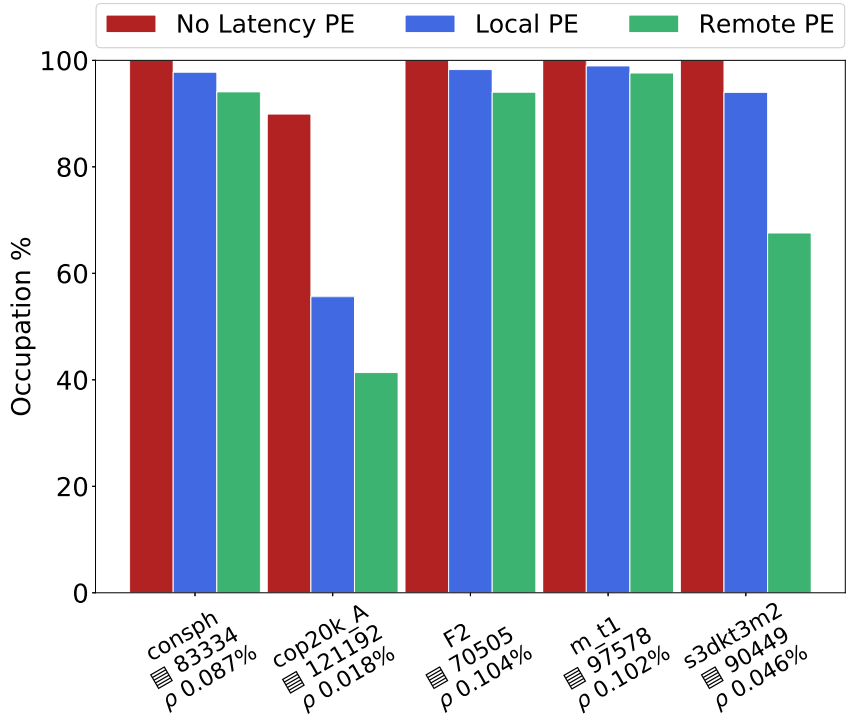


Memory controller activity (occupancy percentage) according to the number of processing elements.

CASE STUDY II: SPGEMM
Performance according the system topology



Computation speed in GFLOP/s according the system topology.



Memory controller activity (occupancy percentage) according the system topology.

PLAN: 5 - CONCLUSION & FUTURE WORK

CONCLUSION

- The increasing heterogeneity of computing systems leads to a more complex programming model.
- Shared memory is a convenient programming paradigm, especially for irregular computing.
- We are proposing a programming model for a S-DSM integrating FPGAs:
  - Targeting irregular compute kernels;
  - Based on chunk partitioning to abstract data structure;
  - Objectives : Accept data-dependent accesses, Addressing latency issue of S-DSM & Unifying memory access scheme for all processing units.
- Experimental results:
  - Efficiently hides distributed data access latency;
  - Almost reach the maximum performance allowed by the accelerator local memory interface.

FUTURE WORK

- Programming model:
  - Extend the programming model;
  - Enable HLS from C source code.
- Simulation method:
  - Model new type of FPGA;
  - Enable multiple FPGAs modeling
- Experimentation:
  - Vary the size or dimension of the stencil or convolution

Merci!

Commissariat à l'énergie atomique et aux énergies alternatives
Campus Saclay Nano-Innov | F-91191 Gif-sur-Yvette Cedex
www-list.cea.fr

Établissement public à caractère industriel et commercial | RCS Paris B 775 685 019