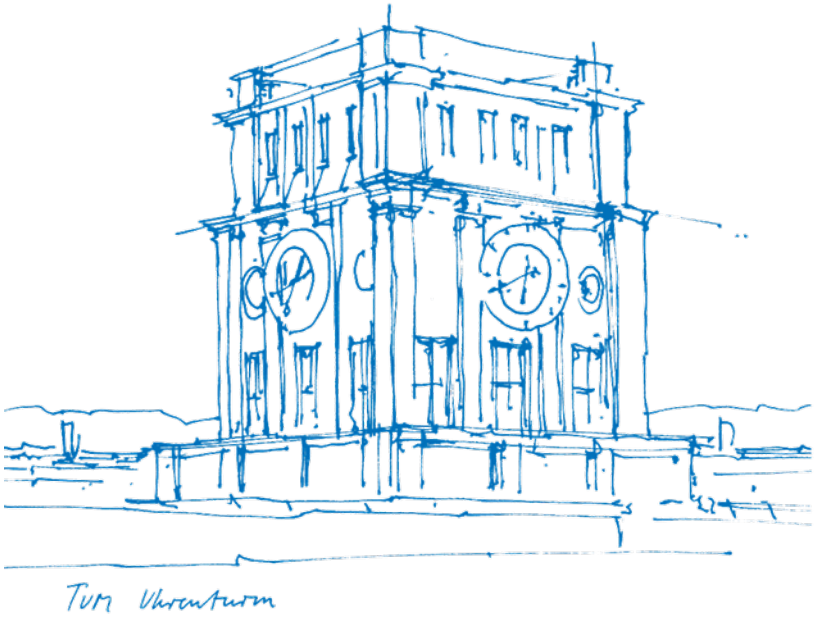
Programming Fully Disaggregated Systems

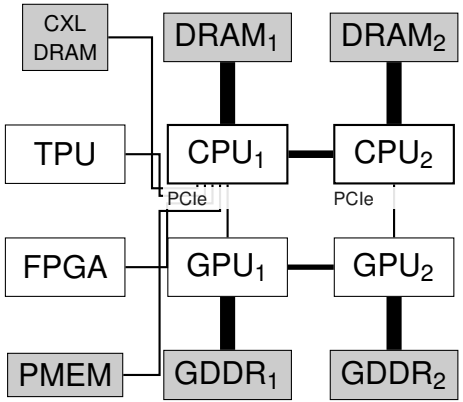
Christoph Anneser, Lukas Vogel, Ferdinand Gruber, Maximilian Bandle, Jana Giceva
School of Computation, Information and Technology
Technical University of Munich

19th Workshop on Hot Topics in Operating Systems
Providence, Rhode Island, USA
22–24 June 2023

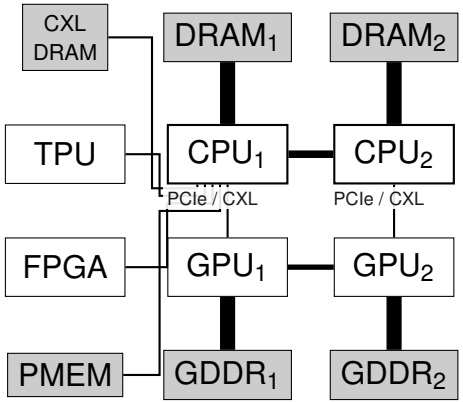


Motivation

Motivation


CXL DRAM
DRAM1
DRAM2
TPU
CPU1
CPU2
PCIe
PCIe
FPGA
GPU1
GPU2
PMEM
GDDR1
GDDR2

Motivation


CXL DRAM
DRAM1
DRAM2
TPU
CPU1
CPU2
PCIe / CXL
PCIe / CXL
FPGA
GPU1
GPU2
PMEM
GDDR1
GDDR2

Motivation

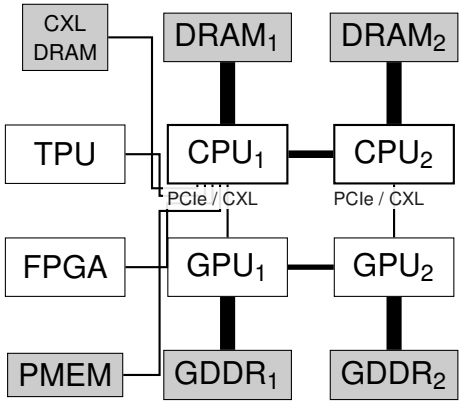


Table: Memory device properties as seen from a CPU.

Name	Bw.	Lat.	Gran.	Attached	Sync	Persist.
Cache	++	++	1 B	CPU	✓	✗
HBM	++	+	64 B	CPU	✓	✗
DRAM	+	+	64 B	CPU	✓	✗
PMem	○	○	256 B	CPU	✓	✓
CXL-DRAM	○	○	64 B	PCIe	✓/✗	✓/✗
Disagg. Mem.	○	–	?	NIC	✗	✓/✗
SSD	–	–	4 KiB	PCIe	✗	✓
HDD	– –	– –	4 KiB	SATA	✗	✓

Motivation

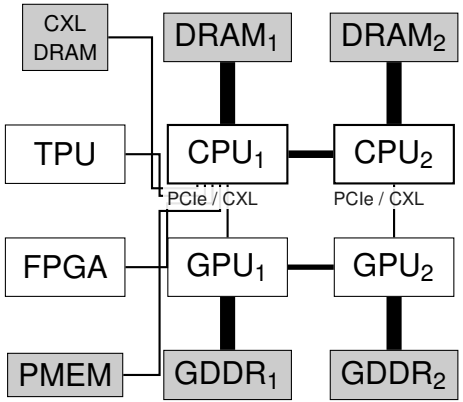
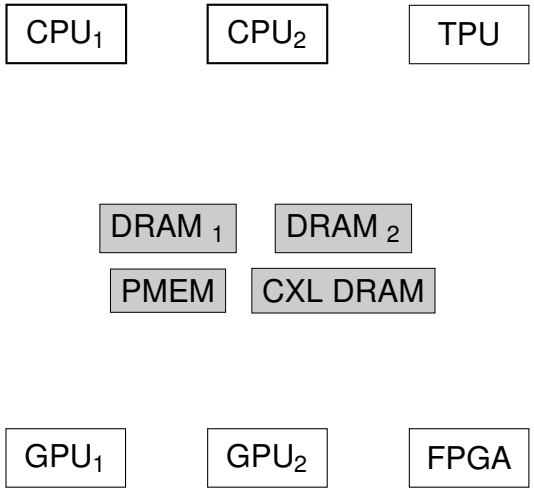


Table: Memory device properties as seen from a CPU.

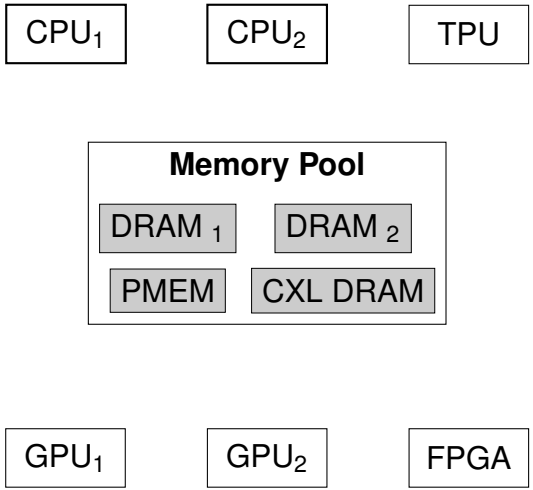
Name	Bw.	Lat.	Gran.	Attached	Sync	Persist.
Cache	++	++	1 B	CPU	✓	✗
HBM	++	+	64 B	CPU	✓	✗
DRAM	+	+	64 B	CPU	✓	✗
PMem	○	○	256 B	CPU	✓	✓
CXL-DRAM	○	○	64 B	PCIe	✓/✗	✓/✗
Disagg. Mem.	○	–	?	NIC	✗	✓/✗
SSD	–	–	4 KiB	PCIe	✗	✓
HDD	– –	– –	4 KiB	SATA	✗	✓

⇒ **How can we develop & optimize applications for heterogeneous, disaggregated environments?**

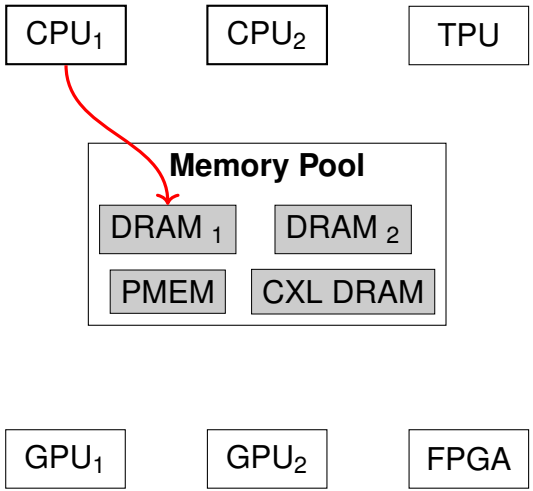
CXL Enables a Memory-Centric View



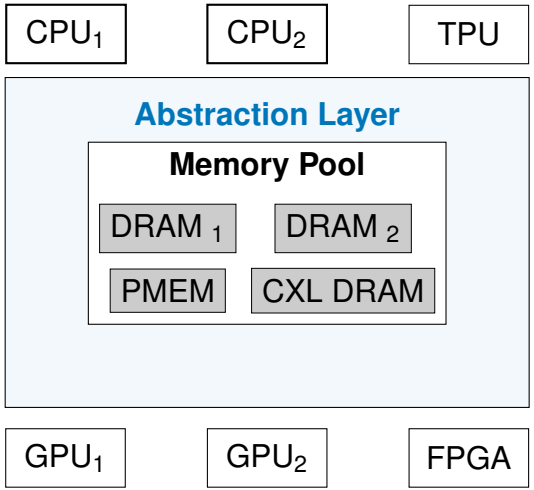
CXL Enables a Memory-Centric View



CXL Enables a Memory-Centric View



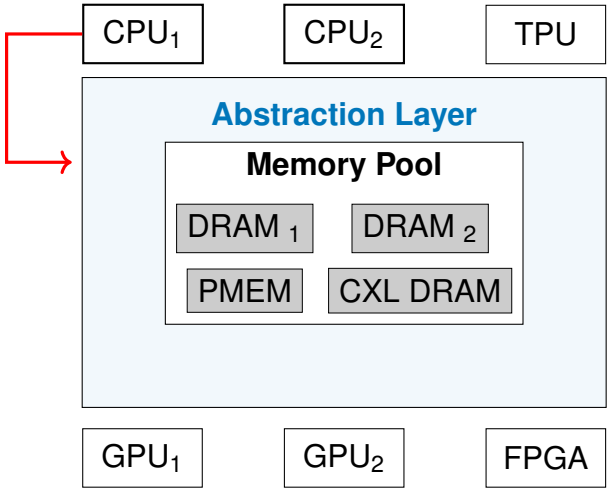
CXL Enables a Memory-Centric View



[1] Gay and Aiken: “Memory Management with Explicit Regions” (1998)

[3] Tofte and Talpin: “Region-based Memory Management” (1997)

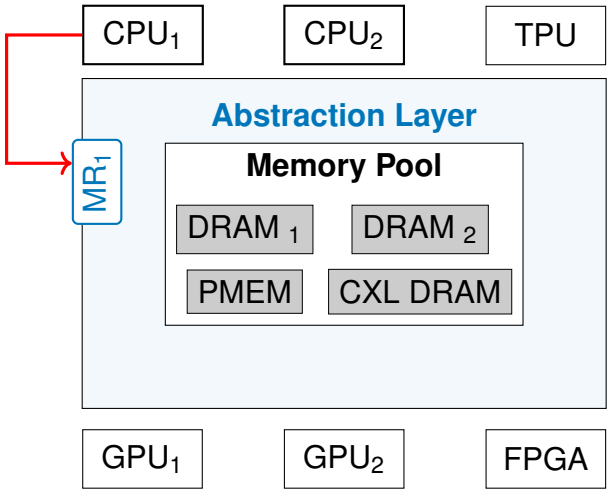
CXL Enables a Memory-Centric View



[1] Gay and Aiken: “Memory Management with Explicit Regions” (1998)
[3] Tofte and Talpin: “Region-based Memory Management” (1997)

CXL Enables a Memory-Centric View

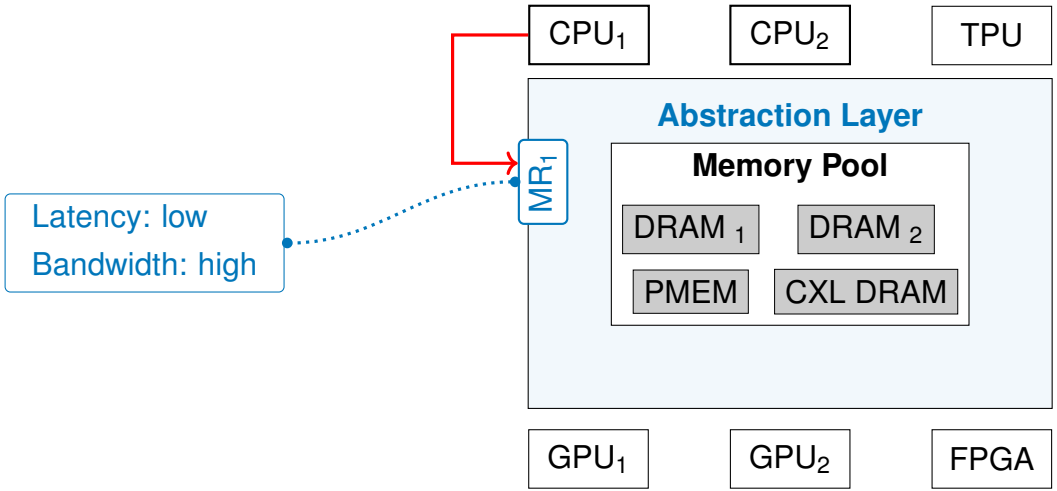
– Leverage **memory regions** [1, 3] as abstraction layer for disaggregated memory!



[1] Gay and Aiken: “Memory Management with Explicit Regions” (1998)
[3] Tofte and Talpin: “Region-based Memory Management” (1997)

CXL Enables a Memory-Centric View

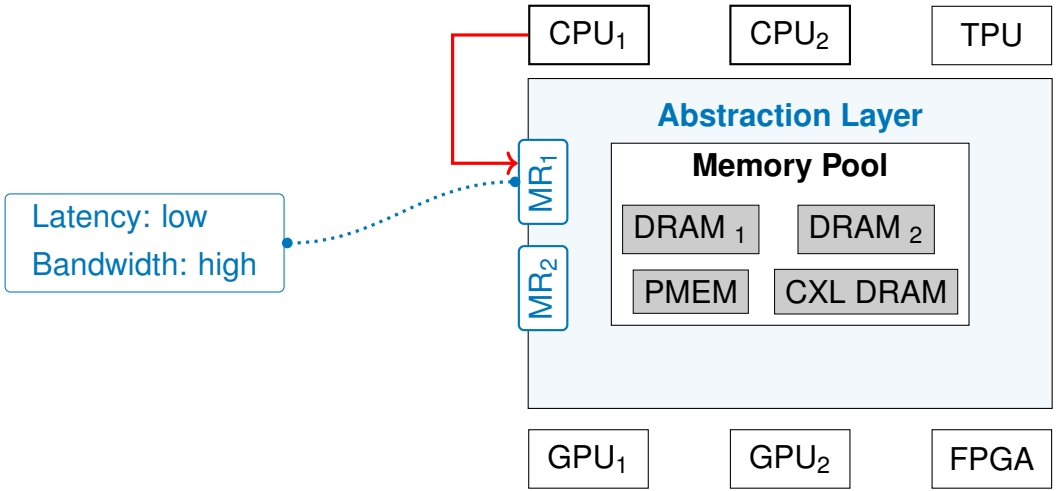
– Leverage **memory regions** [1, 3] as abstraction layer for disaggregated memory!



[1] Gay and Aiken: “Memory Management with Explicit Regions” (1998)
[3] Tofte and Talpin: “Region-based Memory Management” (1997)

CXL Enables a Memory-Centric View

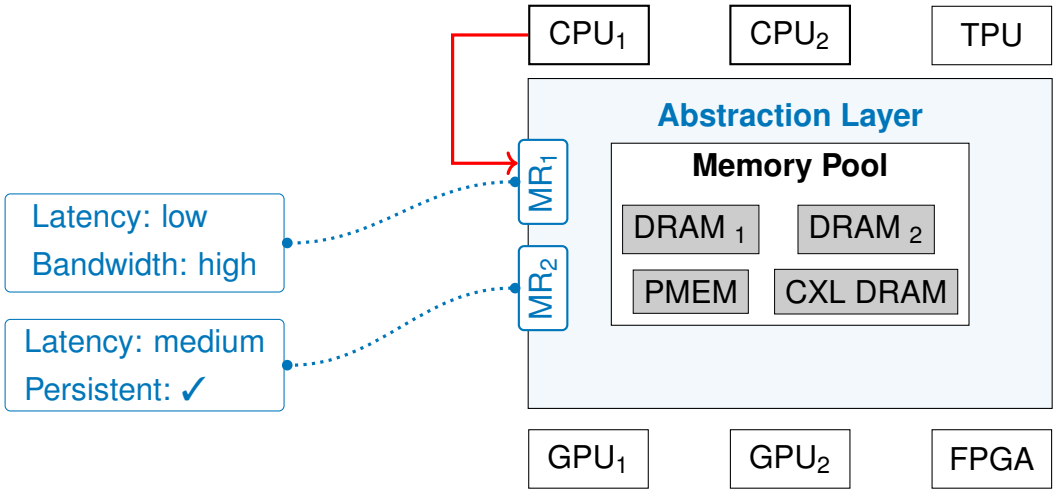
– Leverage **memory regions** [1, 3] as abstraction layer for disaggregated memory!



[1] Gay and Aiken: “Memory Management with Explicit Regions” (1998)
[3] Tofte and Talpin: “Region-based Memory Management” (1997)

CXL Enables a Memory-Centric View

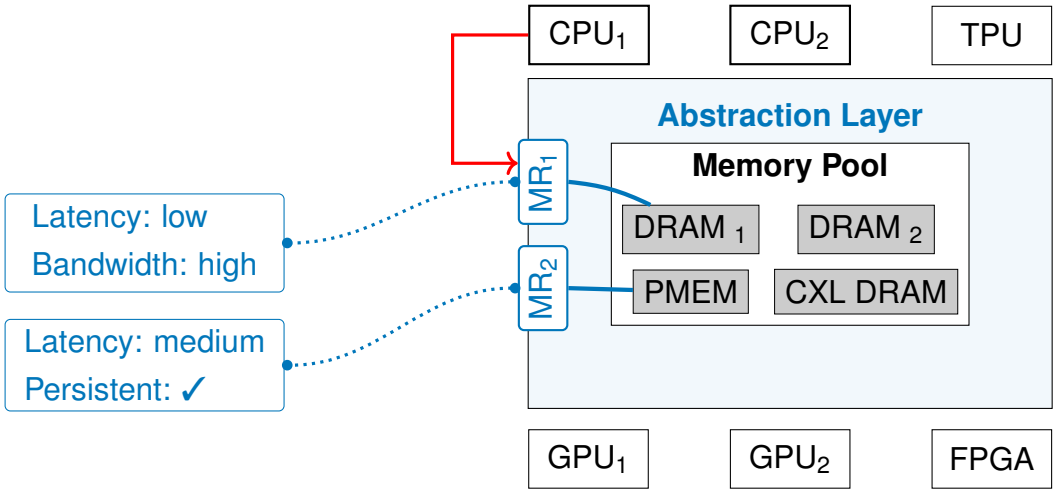
– Leverage **memory regions** [1, 3] as abstraction layer for disaggregated memory!



[1] Gay and Aiken: “Memory Management with Explicit Regions” (1998)
[3] Tofte and Talpin: “Region-based Memory Management” (1997)

CXL Enables a Memory-Centric View

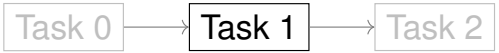
– Leverage **memory regions** [1, 3] as abstraction layer for disaggregated memory!
– Memory Regions are logical view on physical memory!



[1] Gay and Aiken: “Memory Management with Explicit Regions” (1998)
[3] Tofte and Talpin: “Region-based Memory Management” (1997)

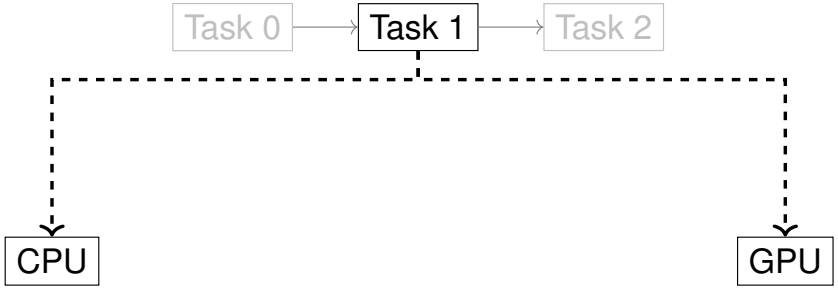
Mapping Memory Regions to Devices

Mapping Memory Regions to Devices



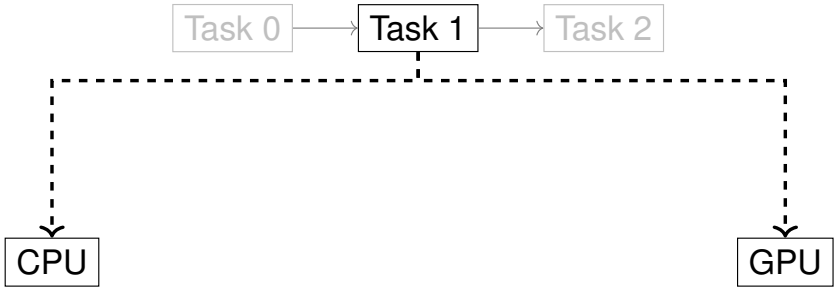
Mapping Memory Regions to Devices

– **Task Placement**



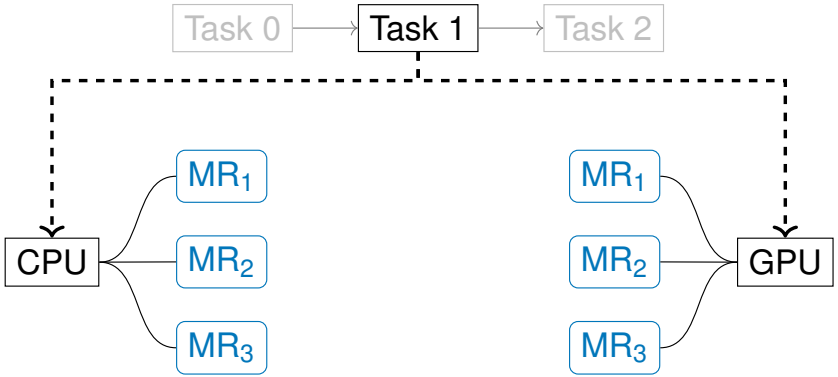
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync



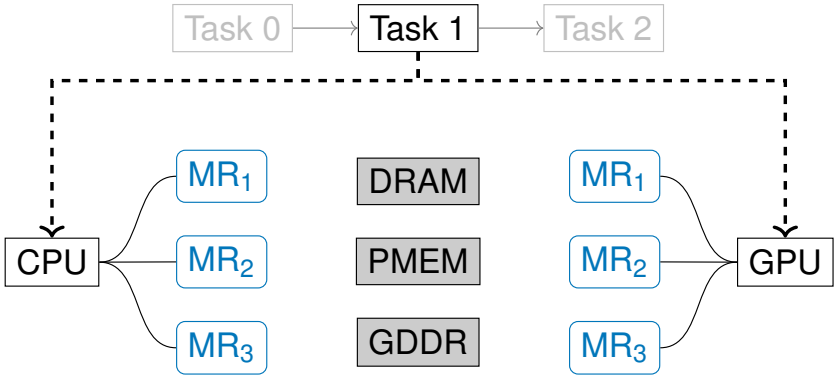
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync



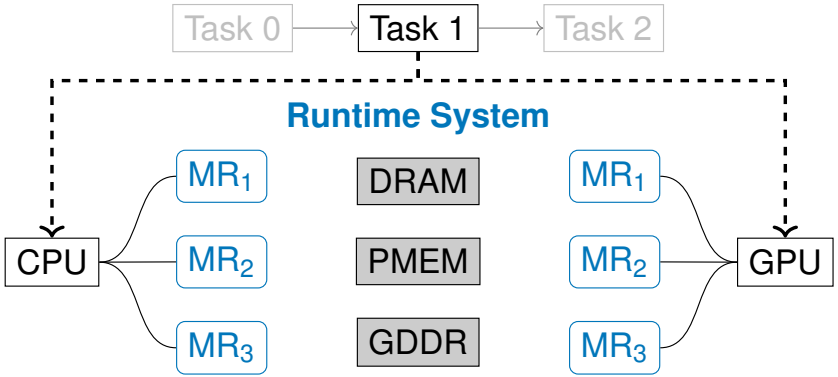
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync



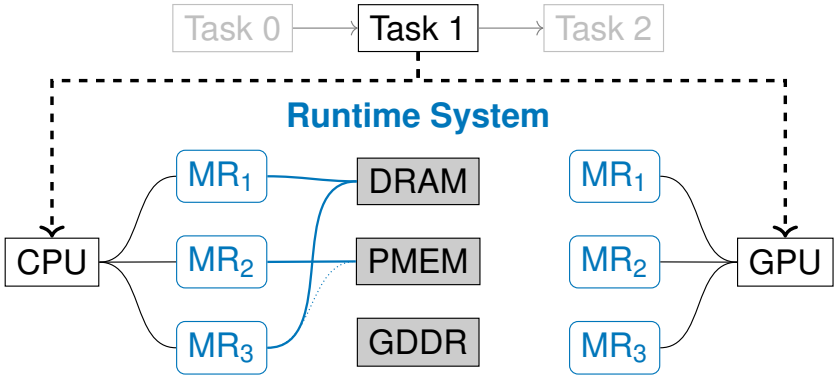
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync



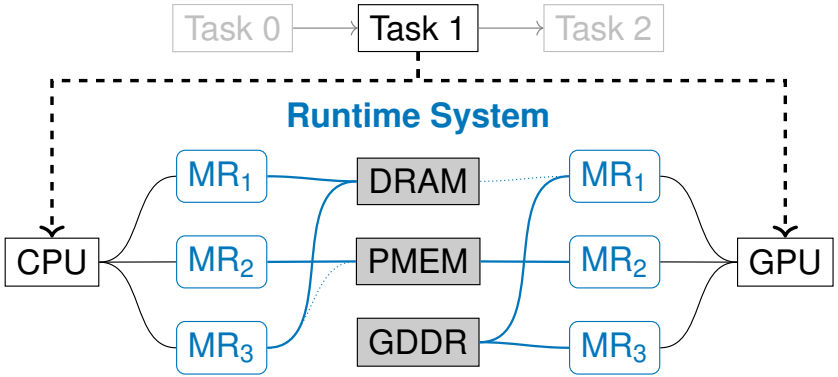
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync



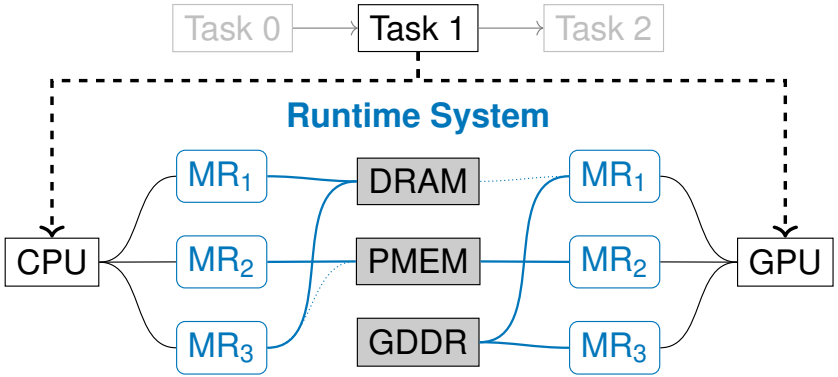
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync



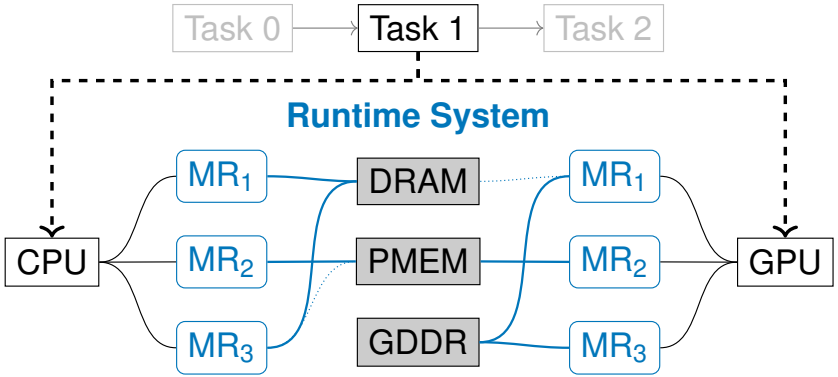
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync
- **Handovers:**
  - MR_1 : T0 Output, T1 Input
  - MR_2: T1 Output, T2 Input



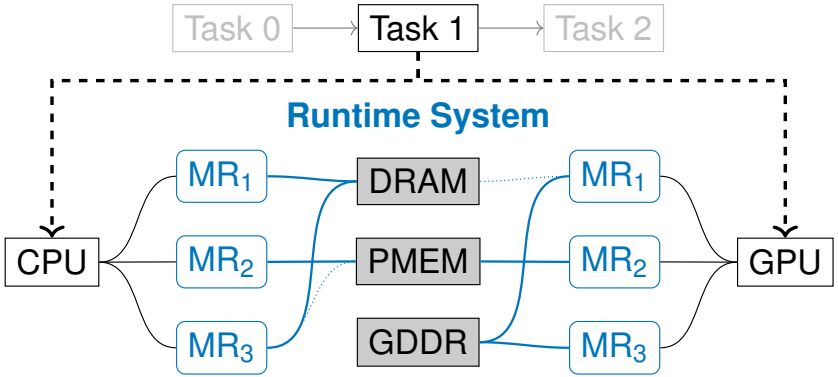
Mapping Memory Regions to Devices

- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync
- **Handovers:**
  - MR_1 : T0 Output, T1 Input
  - MR_2: T1 Output, T2 Input
- **Device Utilization**



Mapping Memory Regions to Devices

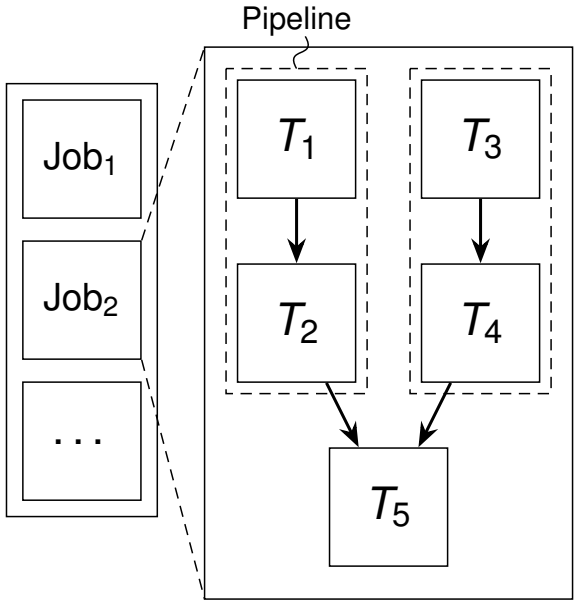
- **Task Placement**
- **Memory Region Properties:**
  - MR_1 : low lat., sync
  - MR_2: low lat., persistent, async
  - MR_3: low lat., high bandwith, sync
- **Handovers:**
  - MR_1 : T0 Output, T1 Input
  - MR_2: T1 Output, T2 Input
- **Device Utilization**



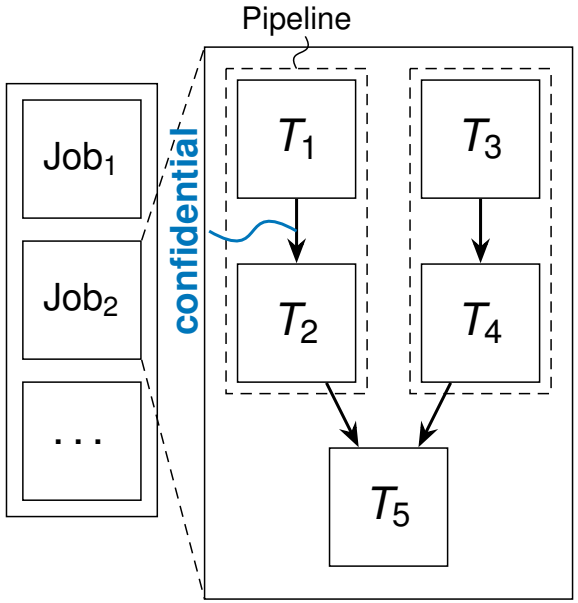
Flexible mapping at runtime → Late Binding

Dataflow Systems on Disaggregated Systems

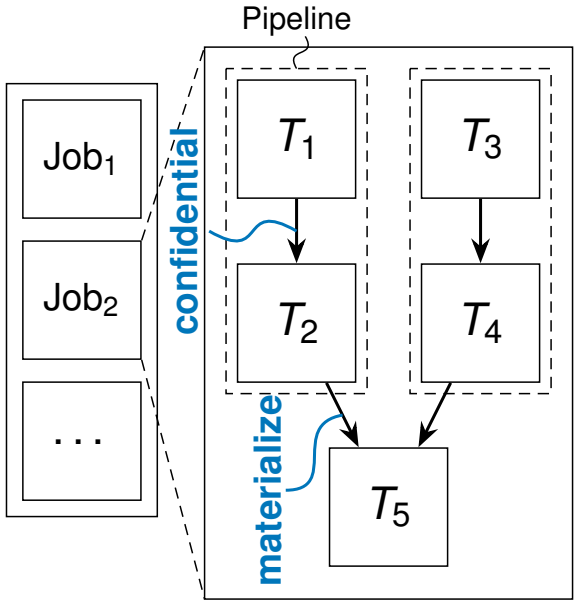
Dataflow Systems on Disaggregated Systems



Dataflow Systems on Disaggregated Systems

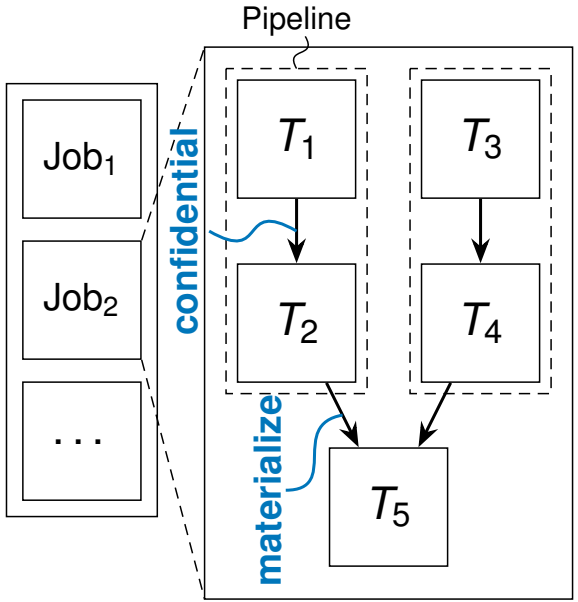


Dataflow Systems on Disaggregated Systems



Dataflow Systems on Disaggregated Systems

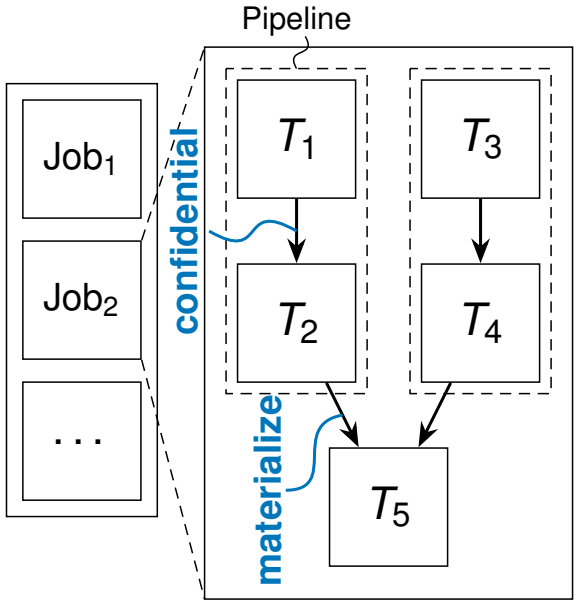
Declaratively attach properties to



Dataflow Systems on Disaggregated Systems

Declaratively attach properties to

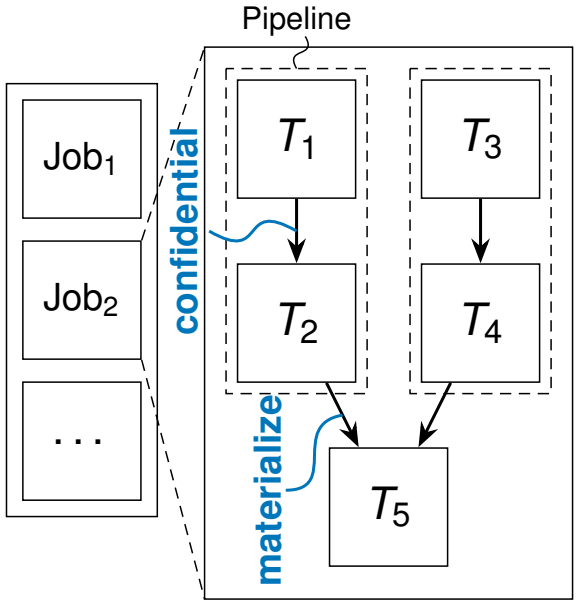
- Memory Regions



Dataflow Systems on Disaggregated Systems

Declaratively attach properties to

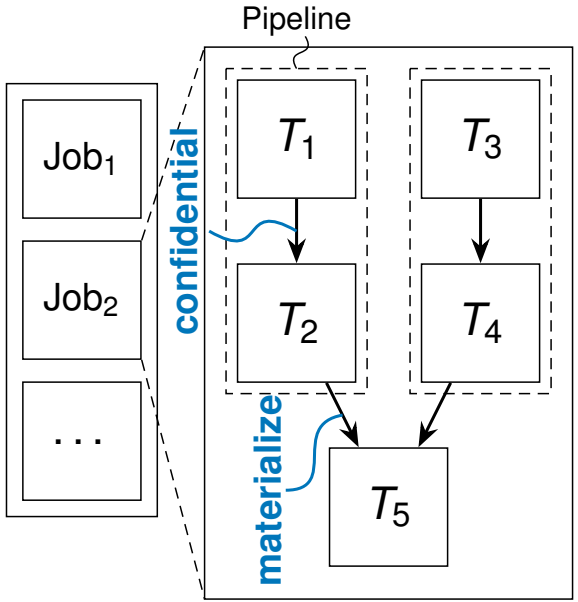
- Memory Regions
- Tasks



Dataflow Systems on Disaggregated Systems

Declaratively attach properties to

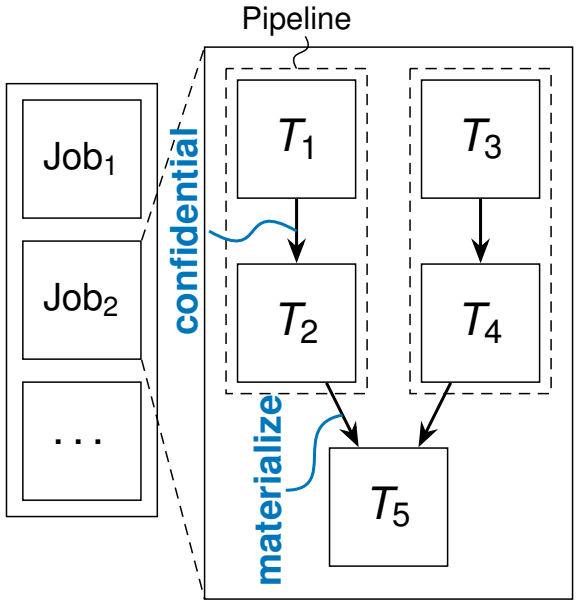
- Memory Regions
- Tasks
- Pipelines



Dataflow Systems on Disaggregated Systems

Declaratively attach properties to

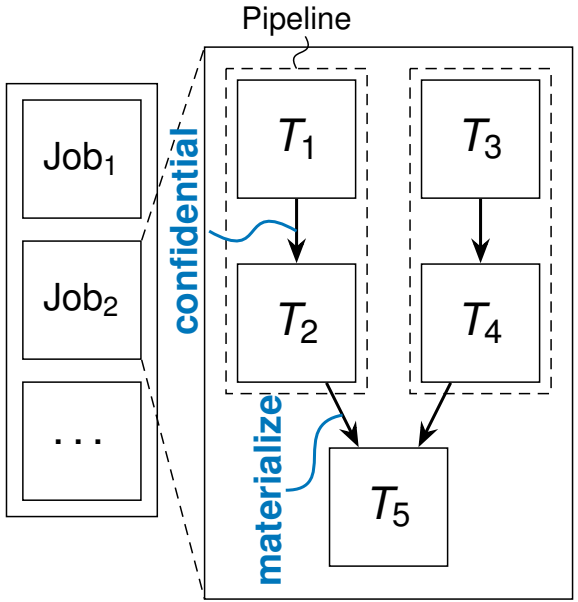
- Memory Regions
- Tasks
- Pipelines
- Jobs



Dataflow Systems on Disaggregated Systems

Declaratively attach properties to

- Memory Regions
- Tasks
- Pipelines
- Jobs
- Applications



Typed Memory Regions [2]

Dataflow applications use memory for . . .

[2] Gog et al.: “Broom: Sweeping Out Garbage Collection from Big Data Systems” (2015)

Typed Memory Regions [2]

Dataflow applications use memory for . . .

. . . **Communication**
- Purpose: Syncing tasks, message passing, . . .
- Properties: coherent, sync

[2] Gog et al.: “Broom: Sweeping Out Garbage Collection from Big Data Systems” (2015)

Typed Memory Regions [2]

Dataflow applications use memory for . . .

. . . **Communication**
- Purpose: Syncing tasks, message passing, . . .
- Properties: coherent, sync

$\Rightarrow$ **Global State**

[2] Gog et al.: “Broom: Sweeping Out Garbage Collection from Big Data Systems” (2015)

Typed Memory Regions [2]

Dataflow applications use memory for . . .

. . . **Communication**
- Purpose: Syncing tasks, message passing, . . .
- Properties: coherent, sync

. . . **Exchanging Data** (e.g. Caching)
- Properties: coherent, async

$\Rightarrow$ **Global State**

[2] Gog et al.: “Broom: Sweeping Out Garbage Collection from Big Data Systems” (2015)

Typed Memory Regions [2]

Dataflow applications use memory for . . .

. . . **Communication**
- Purpose: Syncing tasks, message passing, . . .
- Properties: coherent, sync

⇒ **Global State**

. . . **Exchanging Data** (e.g. Caching)
- Properties: coherent, async

⇒ **Global Scratch**

[2] Gog et al.: “Broom: Sweeping Out Garbage Collection from Big Data Systems” (2015)

Typed Memory Regions [2]

Dataflow applications use memory for . . .

. . . **Communication**
- Purpose: Syncing tasks, message passing, . . .
- Properties: coherent, sync

⇒ **Global State**

. . . **Exchanging Data** (e.g. Caching)
- Properties: coherent, async

⇒ **Global Scratch**

. . . **Thread-local State**
- Properties: non-coherent, sync, fast

[2] Gog et al.: “Broom: Sweeping Out Garbage Collection from Big Data Systems” (2015)

Typed Memory Regions [2]

Dataflow applications use memory for . . .

. . . **Communication**
- Purpose: Syncing tasks, message passing, . . .
- Properties: coherent, sync

⇒ **Global State**

. . . **Exchanging Data** (e.g. Caching)
- Properties: coherent, async

⇒ **Global Scratch**

. . . **Thread-local State**
- Properties: non-coherent, sync, fast

⇒ **Private Scratch**

[2] Gog et al.: “Broom: Sweeping Out Garbage Collection from Big Data Systems” (2015)

Typed Memory Regions – cont'd

How will different application types use the Typed Memory Regions?

	Priv. Scratch	Glob. State	Glob. Scratch

Typed Memory Regions – cont'd

How will different application types use the Typed Memory Regions?

	Priv. Scratch	Glob. State	Glob. Scratch
DBMS	operator state (hashtables, . . .)	synchronization (locks, . . .)	(temp) indexes, caches

Typed Memory Regions – cont'd

How will different application types use the Typed Memory Regions?

	Priv. Scratch	Glob. State	Glob. Scratch
DBMS	operator state (hashtables, . . .)	synchronization (locks, . . .)	(temp) indexes, caches
ML/AI	model training state	metadata, worker state	input data, cached transf. data

Typed Memory Regions – cont'd

How will different application types use the Typed Memory Regions?

	Priv. Scratch	Glob. State	Glob. Scratch
DBMS	operator state (hashtables, . . .)	synchronization (locks, . . .)	(temp) indexes, caches
ML/AI	model training state	metadata, worker state	input data, cached transf. data
HPC	node-local working mem.	job metadata, node states	object/blob storage

Typed Memory Regions – cont'd

How will different application types use the Typed Memory Regions?

	Priv. Scratch	Glob. State	Glob. Scratch
DBMS	operator state (hashtables, . . .)	synchronization (locks, . . .)	(temp) indexes, caches
ML/AI	model training state	metadata, worker state	input data, cached transf. data
HPC	node-local working mem.	job metadata, node states	object/blob storage
Streaming	cache/buffer (send, recv.)	cluster/worker state	result/data cache

Lifetime Management of Memory Regions

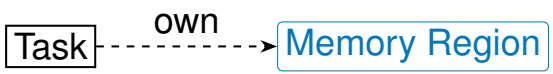
Lifetime Management of Memory Regions

– **Challenge**: Memory Regions might outlive CPU/GPU/. . . Tasks/Processes

Lifetime Management of Memory Regions

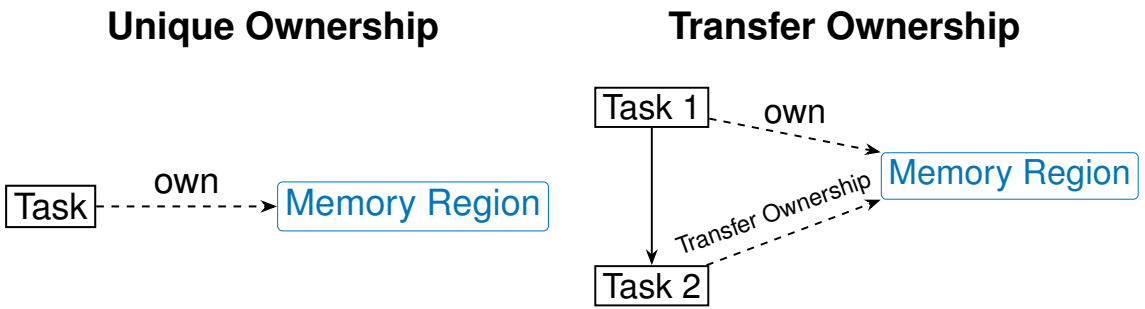
– **Challenge**: Memory Regions might outlive CPU/GPU/. . . Tasks/Processes
– **Ownership** of Memory Regions:

Unique Ownership



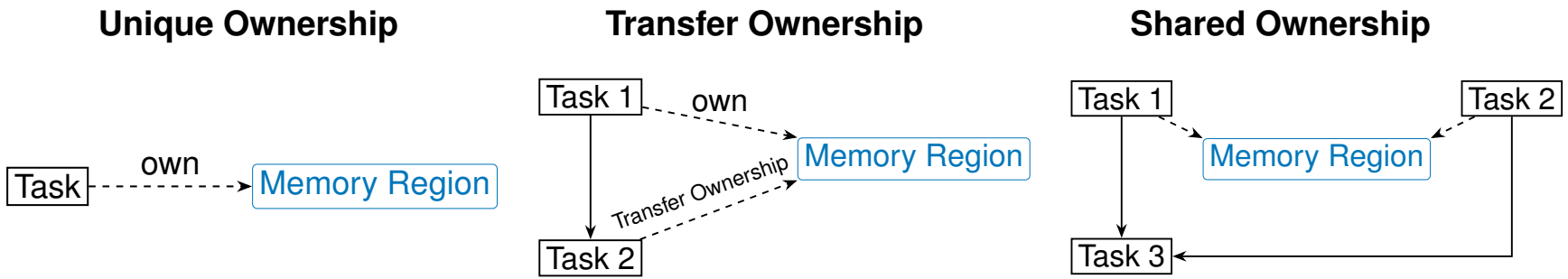
Lifetime Management of Memory Regions

– **Challenge**: Memory Regions might outlive CPU/GPU/. . . Tasks/Processes
– **Ownership** of Memory Regions:



Lifetime Management of Memory Regions

– **Challenge**: Memory Regions might outlive CPU/GPU/. . . Tasks/Processes
– **Ownership** of Memory Regions:



Our Vision

Challenge: Developing dataflow applications for fully disaggregated systems

We propose a new programming model:

(1) A memory-centric view based on *logical memory regions*

Our Vision

Challenge: Developing dataflow applications for fully disaggregated systems

We propose a new programming model:

(1) A memory-centric view based on *logical memory regions*

(2) Requesting memory *declaratively* based on properties

Our Vision

Challenge: Developing dataflow applications for fully disaggregated systems

We propose a new programming model:

(1) A memory-centric view based on *logical memory regions*

(2) Requesting memory *declaratively* based on properties

(3) *Typed memory regions*

Our Vision

Challenge: Developing dataflow applications for fully disaggregated systems

We propose a new programming model:

(1) A memory-centric view based on *logical memory regions*

(2) Requesting memory *declaratively* based on properties

(3) *Typed memory regions*

(4) A *runtime system* that co-optimizes data- and compute-placement

The Way Forward

How can we turn our vision into a programing model?

(1) The Runtime System . . .
  – What functionality is required from the RTS?

The Way Forward

How can we turn our vision into a programing model?

(1) The Runtime System . . .
- What functionality is required from the RTS?
- Where should the RTS/control plane be placed?

The Way Forward

How can we turn our vision into a programing model?

(1) The Runtime System . . .
- What functionality is required from the RTS?
- Where should the RTS/control plane be placed?
- Page- or object-based memory allocations?

The Way Forward

How can we turn our vision into a programing model?

(1) The Runtime System . . .
- What functionality is required from the RTS?
- Where should the RTS/control plane be placed?
- Page- or object-based memory allocations?
- What layer supports the RTS memory deployment?

The Way Forward

How can we turn our vision into a programing model?

(1) The Runtime System . . .
- What functionality is required from the RTS?
- Where should the RTS/control plane be placed?
- Page- or object-based memory allocations?
- What layer supports the RTS memory deployment?

(2) The Programming Model . . .

The Way Forward

How can we turn our vision into a programing model?

(1) The Runtime System . . .
- What functionality is required from the RTS?
- Where should the RTS/control plane be placed?
- Page- or object-based memory allocations?
- What layer supports the RTS memory deployment?

(2) The Programming Model . . .
- Can we combine declarative and iterative concepts?

The Way Forward

How can we turn our vision into a programing model?

(1) The Runtime System . . .
- What functionality is required from the RTS?
- Where should the RTS/control plane be placed?
- Page- or object-based memory allocations?
- What layer supports the RTS memory deployment?

(2) The Programming Model . . .
- Can we combine declarative and iterative concepts?

Thank you for your attention!

anneser@in.tum.de

Sources

[1] David Gay and Alexander Aiken. “Memory Management with Explicit Regions”. In: *PLDI*. ACM, 1998, pp. 313–323.

[2] Ionel Gog et al. “Broom: Sweeping Out Garbage Collection from Big Data Systems”. In: *HotOS*. USENIX Association, 2015.

[3] Mads Tofte and Jean-Pierre Talpin. “Region-based Memory Management”. In: *Inf. Comput.* 132.2 (1997), pp. 109–176.

Roadmap

- Set-up the testbed
  - Machine with CXL-support (e.g. Intel's Sapphire Rapids)
  - Identify suitable Workloads for CXL
    - Dataflow over heterogeneous hardware
    - Memory extension for large intermediate state
  - Understand CXL's performance implications through benchmarks

- Design and implement the RTS key components & building blocks
  - CXL-enabled vmcache, overcoming the accelerator's limited capacities
  - Optimizer for data- and compute-placement