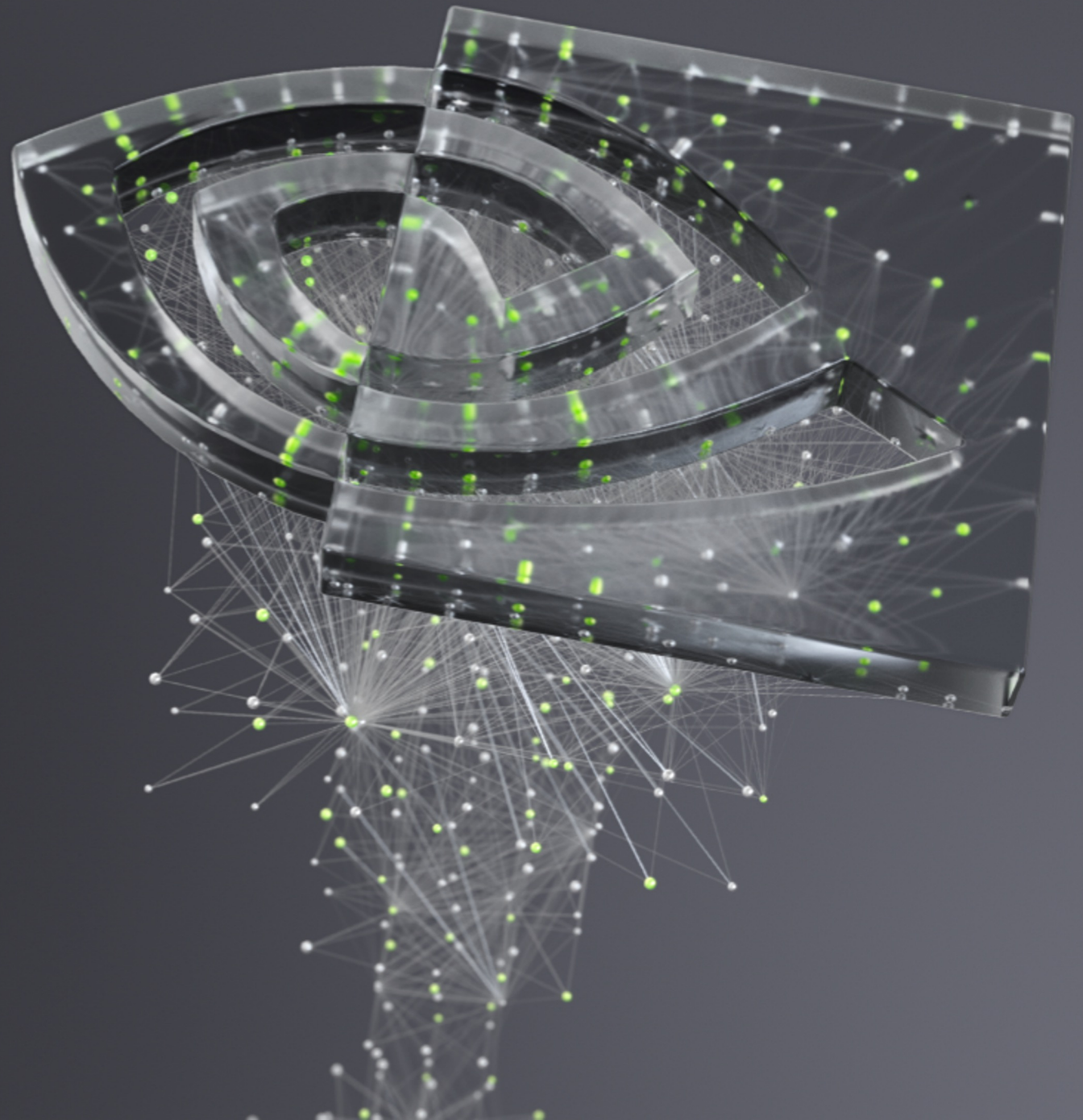


REALM ROADMAP

realm-dev@

REALM

Team

- Artem Priakhin
- Cory Perry
- Sean Treichler
- Wei Wu

REALM

Roadmap

- What’s been done 2022 Q1-Q4
- Short-term plans (3-6 months)
- Long-term plans (6-24 months)
- Projects under discussion
- Q/A

REALM

2022 Q1-Q4

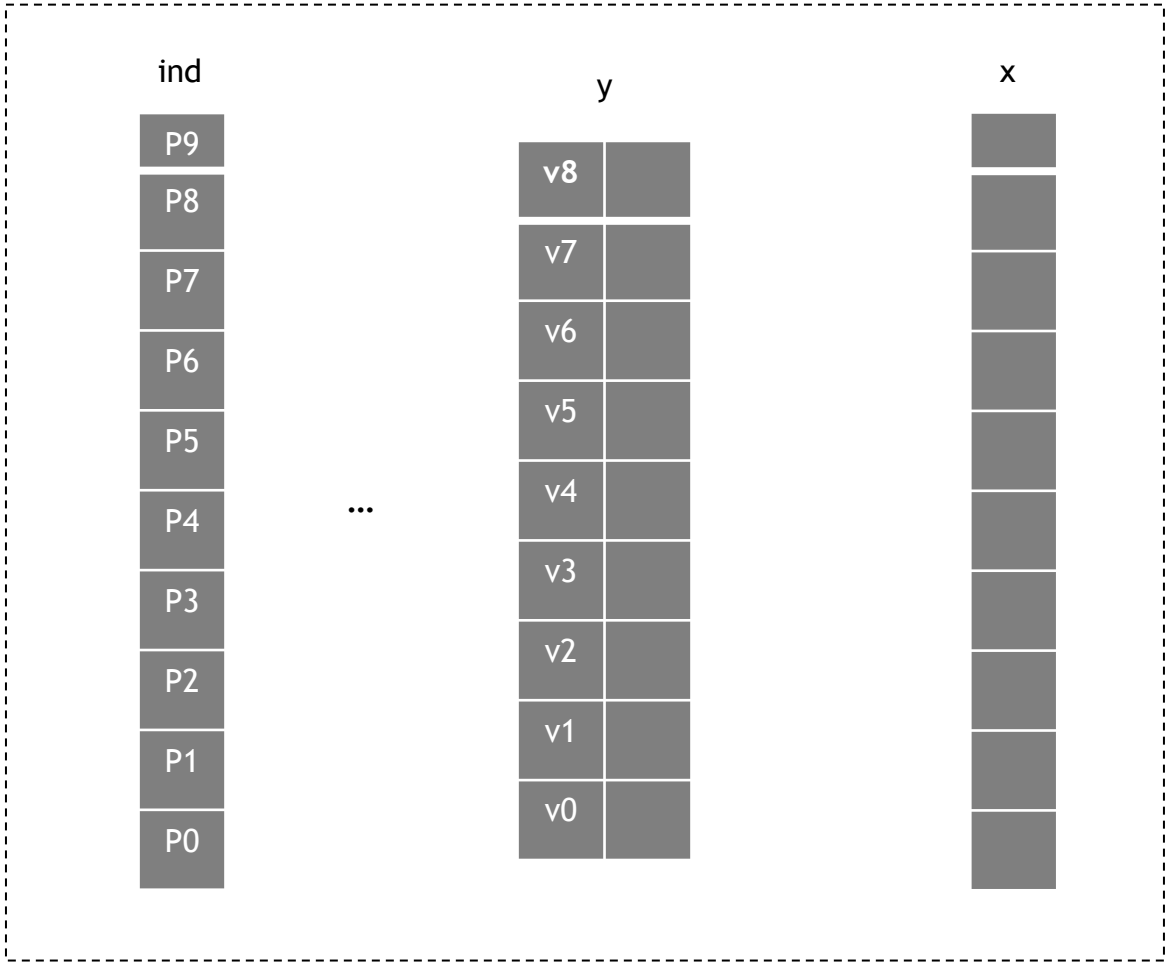
- Affine
  - Affine indirect copies
  - Affine image/preimage operations
- CUDA
  - CUDA array support
  - CUDA dynamic memory
- Faster path planning
- Priorities on copies/fills/partition operations
- Profiling

AFFINE INDIRECT COPIES

2022 Q1-Q4

- What are indirect copies?

```
import numpy as np
x = …
y = …
ind = [P0, P1, P2..]
x = y[ind]
```

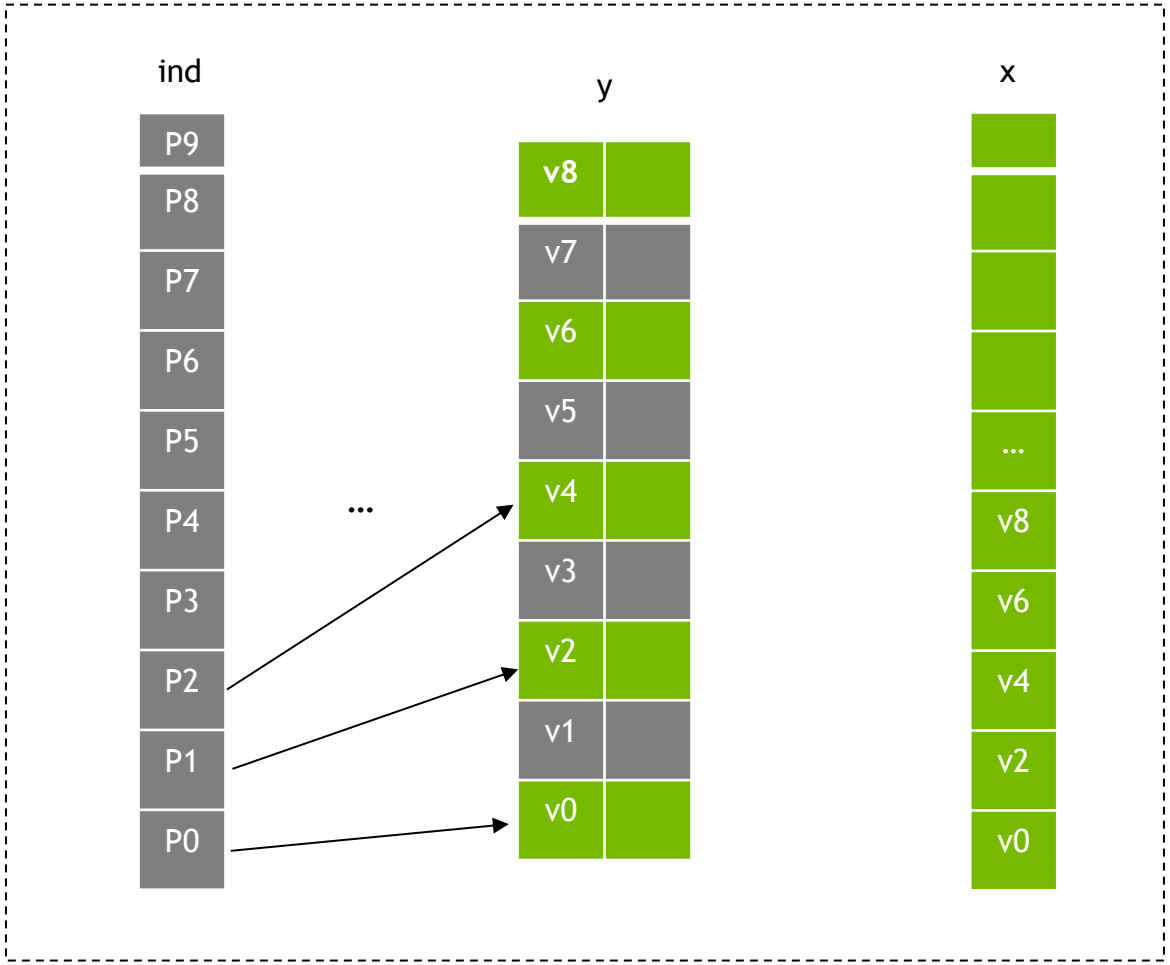


AFFINE INDIRECT COPIES

2022 Q1-Q4

- What are indirect copies?

```
import numpy as np
x = …
y = …
ind = [P0, P1, P2..]
x = y[ind]
```

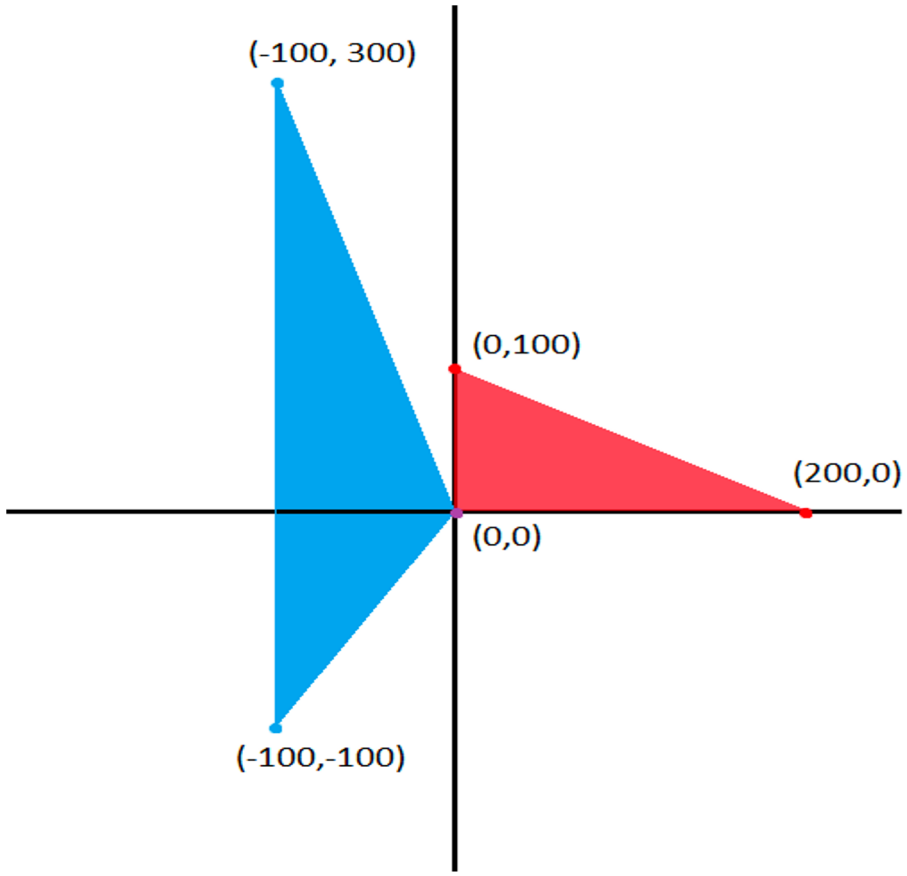


AFFINE INDIRECT COPIES

2022 Q1-Q4

- What is an affine transformation?
  - Any transformation that preserves collinearity..

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -100 \\ -100 \end{bmatrix}$$



y

AFFINE INDIRECT COPIES

2022 Q1-Q4

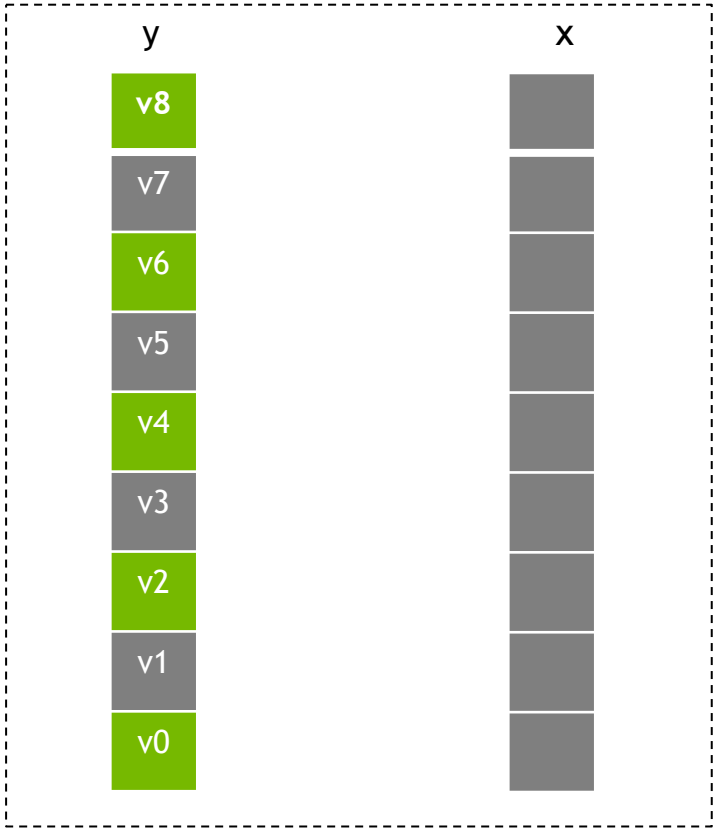
- Why affine indirect copies are interesting?
  - If a copy can be described with an affine transformation, it can be done as fast dense copy instead of regular indirect copy.

AFFINE INDIRECT COPIES

2022 Q1-Q4

- Why affine indirect copies are interesting?
  - If a copy can be described with an affine transformation it can be done as fast dense copy instead of regular indirect copy.

```
import numpy as
np
x = …
y = …
...
x = y[::2]
```



AFFINE INDIRECT COPIES

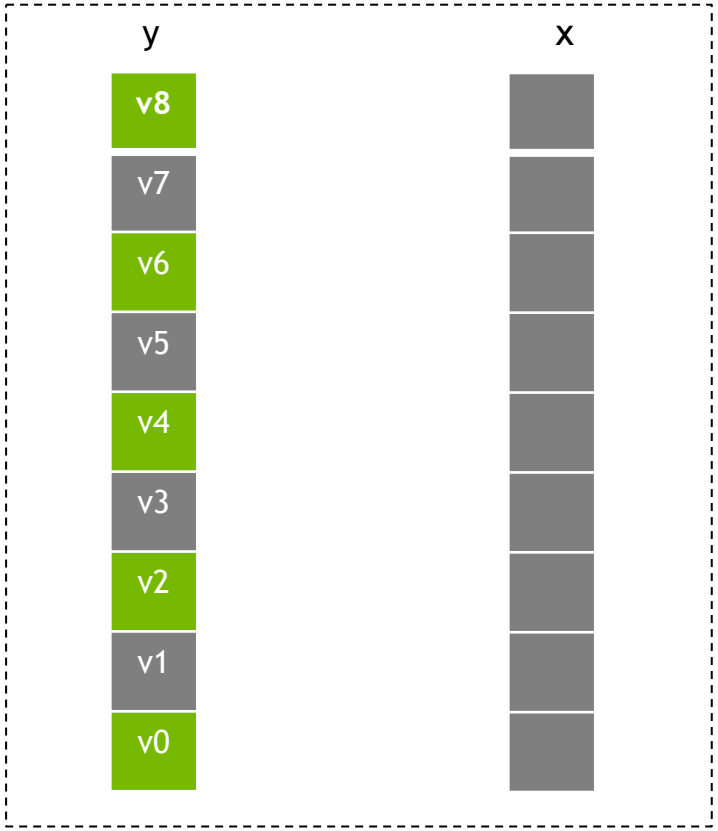
2022 Q1-Q4

- Why affine indirect copies are interesting?
  - If a copy can be described with an affine transformation it can be done as fast dense copy instead of regular indirect copy.

Scale

$$A\left(\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) x + b$$

```
import numpy as
np
x = …
y = …
...
x = y[::2]
```



AFFINE INDIRECT COPIES

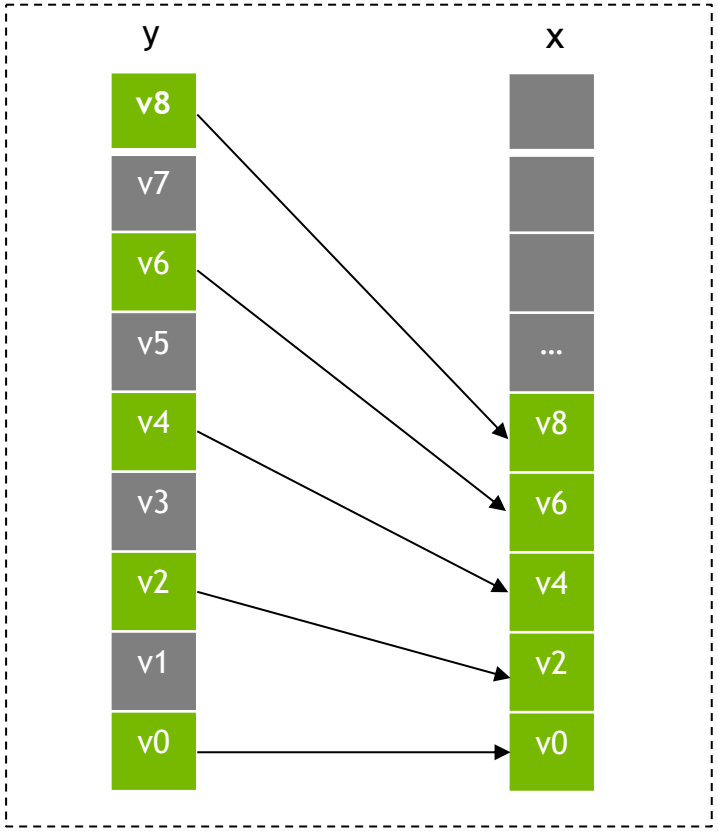
2022 Q1-Q4

- Why affine indirect copies are interesting?
  - If a copy can be described with an affine transformation it can be done as fast dense copy instead of regular indirect copy.

Scale

$$A\left(\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) x + b$$

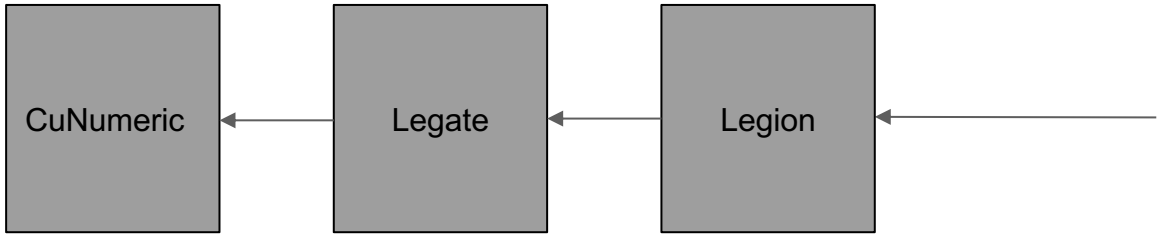
```
import numpy as
np
x = …
y = …
...
x = y[::2]
```



AFFINE INDIRECT COPIES

2022 Q1-Q4

- Legate creates separate tasks to remove affine transformations
  - Github Issue#705



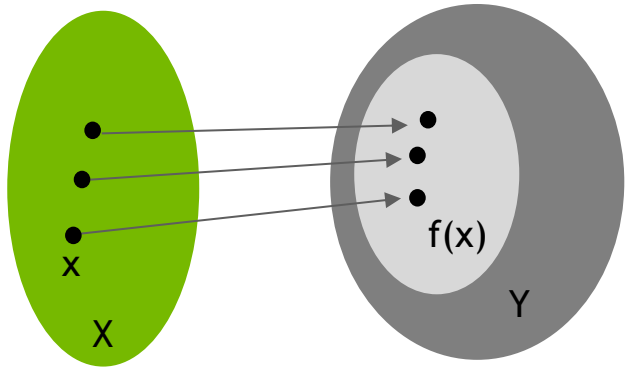
Realm interface ...runtime/realm/indexspace.h

```
template <int N2, typename T2 = int>
 class Unstructured : public CopyIndirection<N,T>::Base {
 public:
   FieldID field_id;
   RegionInstance inst;
   ...
 };

template <int N2, typename T2 = int>
 class Structured : public CopyIndirection<N,T>::Base {
 public:
   Matrix<N,N2,T2> transform;
   Point<N2,T2> offset;
   ...
 };
```

AFFINE IMAGE/PREIMAGE

2022 Q1-Q4

- Image/Preimage are deppart operations
  - Image computes element reachable via an affine transformation
  - Preimage opposite of image - computes elements that reach a given subspace



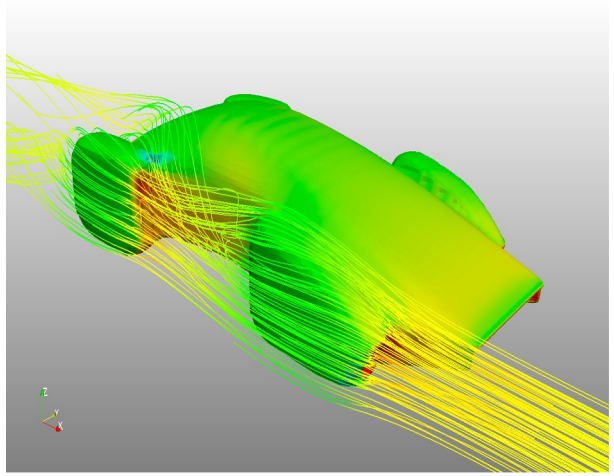

```
template <int N, typename T, int N2, typename T2>
class REALM_PUBLIC_API AffineTransform {
public:
 ...
 Matrix<N, N2, T2> transform;
 Point<N, T2> offset;
};
```

IndexSpace<N, T>::

```
template <int N2, typename T2, typename TRANSFORM>
Event create_subspace_by_image(const TRANSFORM& transform,
                               const IndexSpace<N2, T2>& source,
                               IndexSpace<N, T>& image..) const;

template <int N2, typename T2, typename TRANSFORM>
Event create_subspaces_by_image(const TRANSFORM& transform..);


template <int N2, typename T2, typename TRANSFORM>
Event create_subspace_by_preimage(const TRANSFORM& transform,
                                  const IndexSpace<N2, T2>& target,
                                  IndexSpace<N, T>& preimage..) const;

template <int N2, typename T2, typename TRANSFORM>
Event create_subspaces_by_preimage(const TRANSFORM& transform..);
```

CUDA ARRAYS

- CUDA arrays - opaque memory layouts optimized for texture fetching
  - Write kernels that use CUDA arrays
  - Use HPC simulation data for visualization
    - Paraview/VTK
  - CUDA-graphics interop in render engines
    - Omniverse



CUDA DYNAMIC ALLOCATIONS

2022 Q1-Q4

- Realm reserves storage for memories at startup
  - To provide reliable capacity numbers for mappers
  - Eliminate allocation-related OS/driver overhead during execution
  - Attempt network registration for rdma-based data transfers
  - Provide deferred allocations to let Legion schedule further into the future

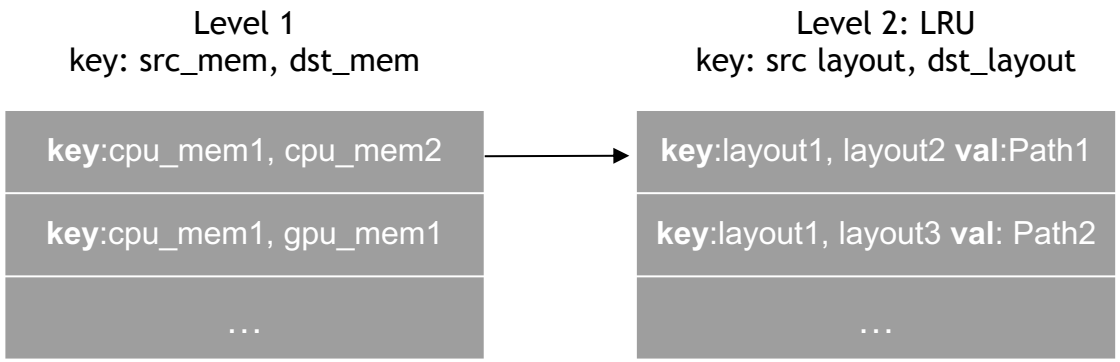
- When composing realm-based application code with non-realm-based code, it may be hard to determine the resources

- To allow this flexibility, realm has added GPU_DYNAMIC_MEM
  - Should be seen as a fallback rather than preferred option

Caching of Optimal Path

2022 Q1-Q4

- Path Planning
  - Required for each Realm copy
  - Expensive
    - Need to check all pairs of src/ib and ib/dst memories
    - Cost increases with the number of memories per node (rank-per-node > rank-per-gpu)
- Caching Optimal Paths
  - Path depends on
    - src mem, dst mem, redop_id, src layout, dst layout
  - One level cache could be too large
  - Solution: two level LRU cache
    - LRU size: -ll:path_cache_size (16 by default)



INTEGRATION WITH NSIGHT SYSTEMS

2022 Q1-Q4

- Integration with Nsight
  - Nsight Systems support NVTX (annotation language)
    - Realm provides a library to use NVTX tags

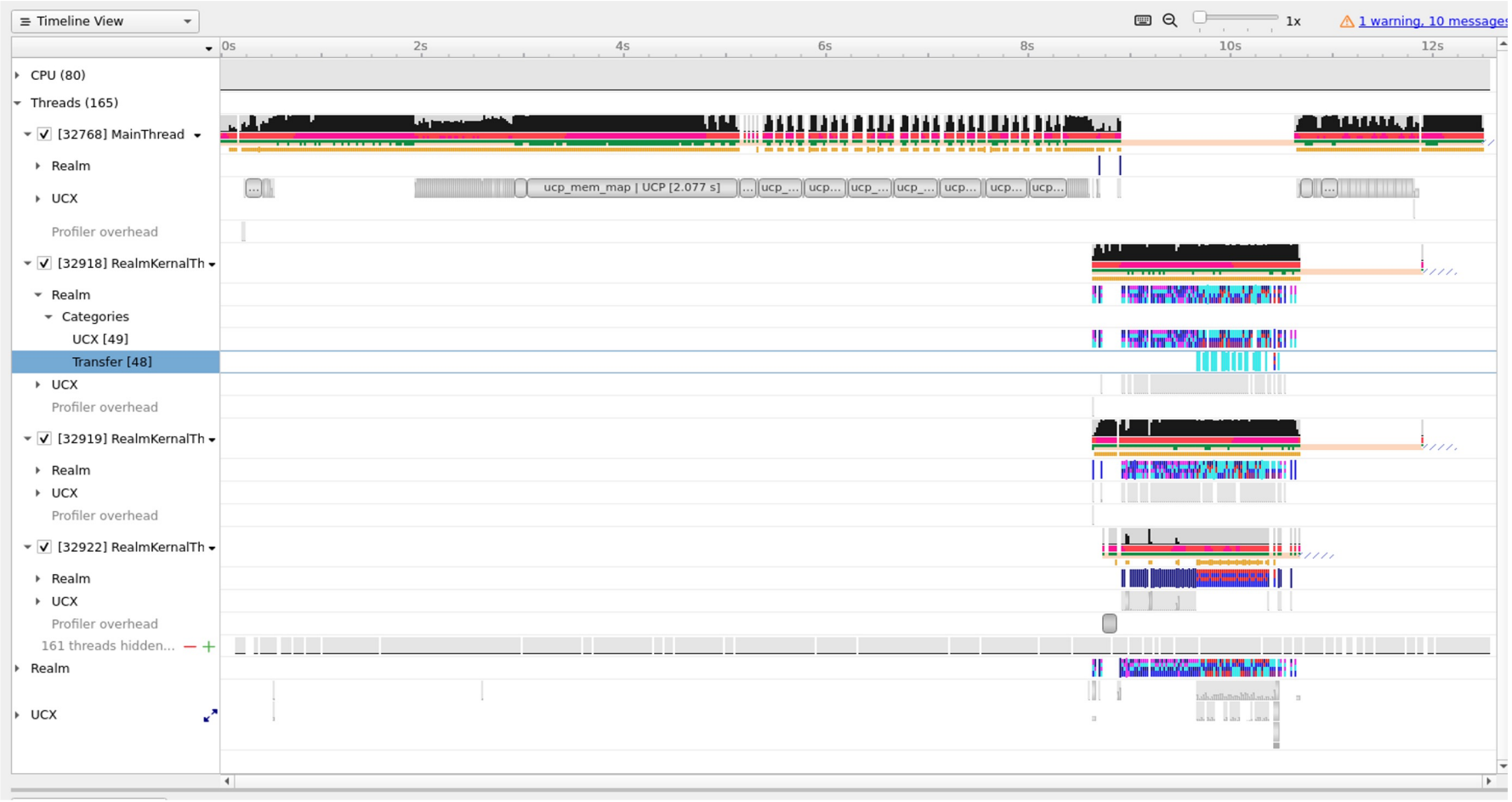
../runtime/realm/nvtx.h

```
void nvtx_range_push(NvtxCategory *category, const char *message,
                     uint32_t color = nvtx_color::white, int32_t payload);
void nvtx_range_pop(void);
```

```
bool BackgroundWorkManager::Worker::do_work(...) {
   nvtx_range_push(NvtxCategory::bgwork, "bgwork", color, payload);
   ...
   nvtx_range_pop();
}
```

PROFILING

Nsight



PROFILING

Nsight

REALM

Short-term (3-6 months)

- CUDA DMA
- Profiling
- UCX Network Module

GPU CUDA DMA

2022 Q4…

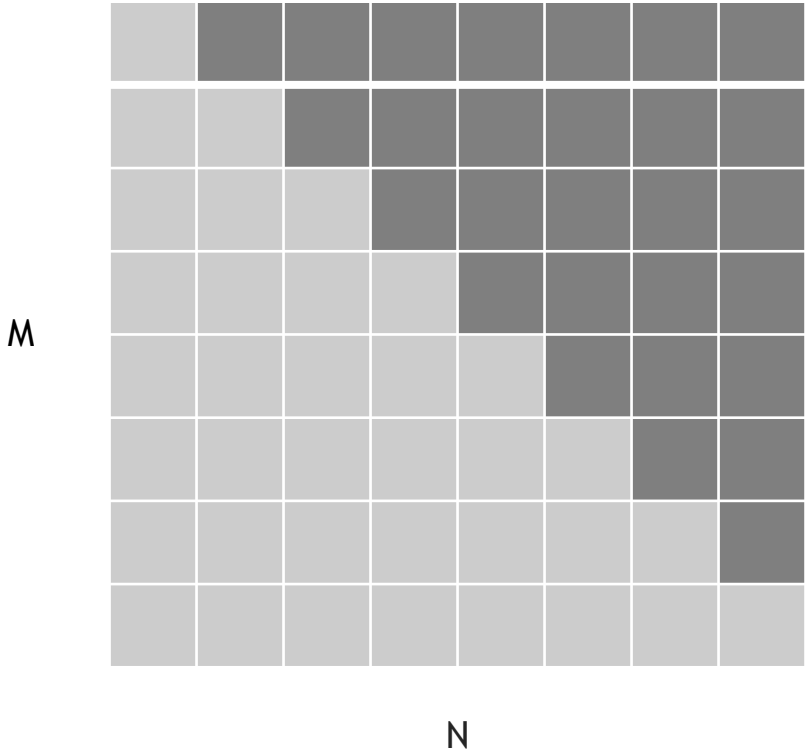
- What is CUDA DMA?
  - An improvement over the current DMA mechanism to achieve lower latency on:
    - sparse memory transfers
    - dense transfers which are not cudaMemcpy friendly
  - Use SM instead of CE

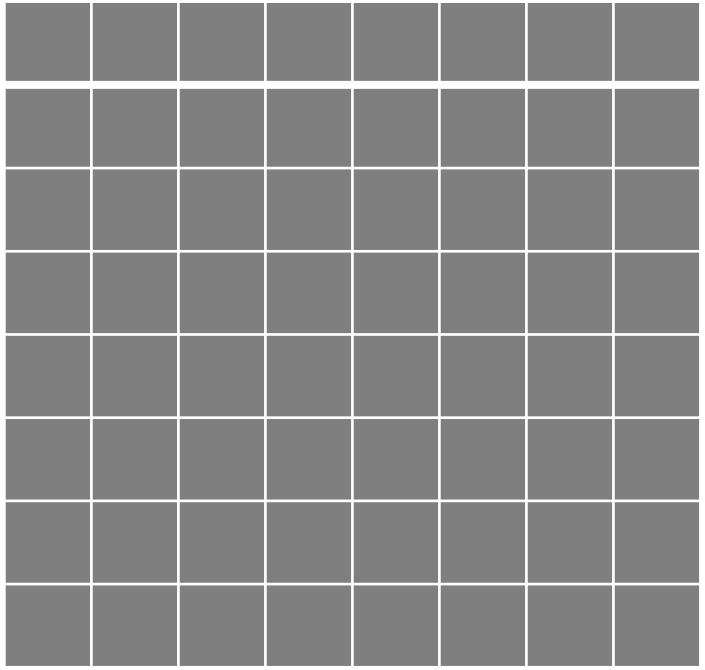
- Github Issue#621

GPU CUDA DMA

Existing GPU DMA

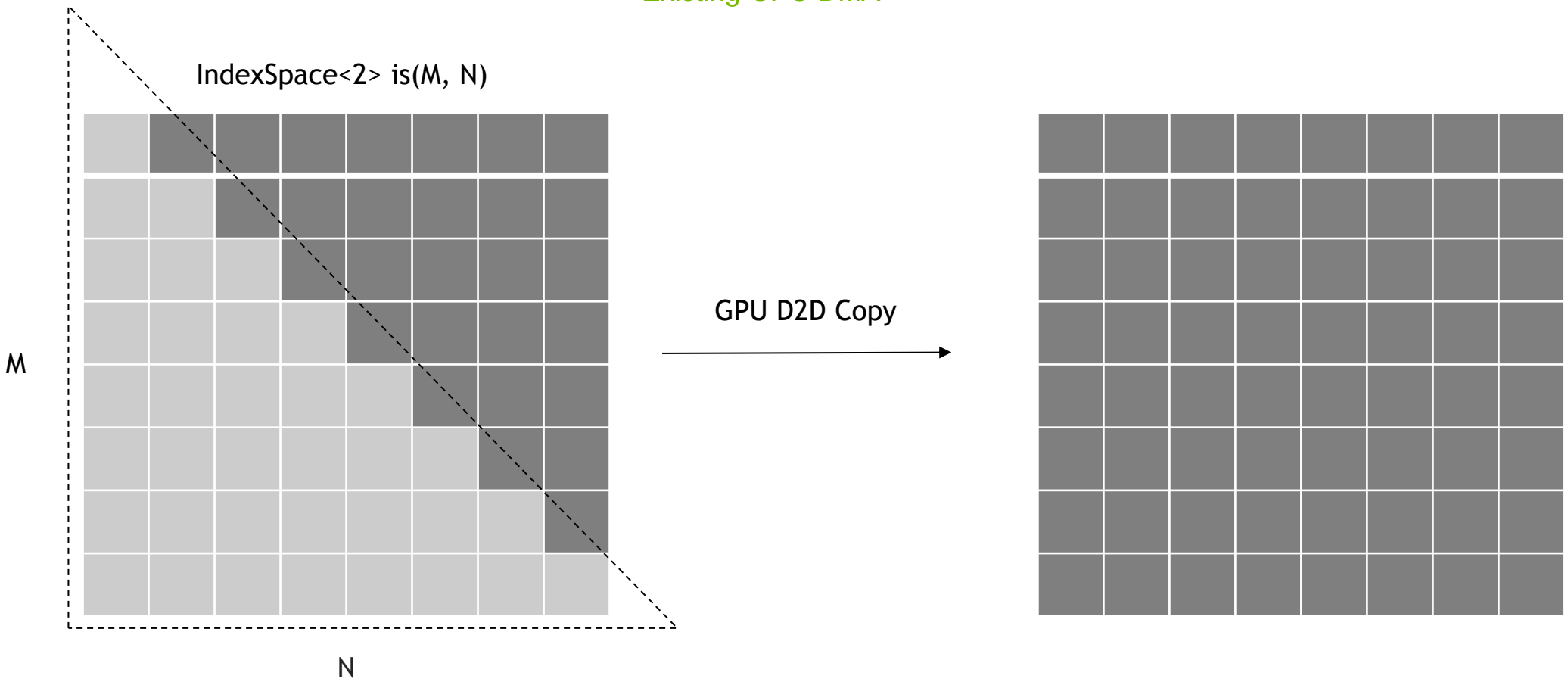
IndexSpace<2> is(M, N)





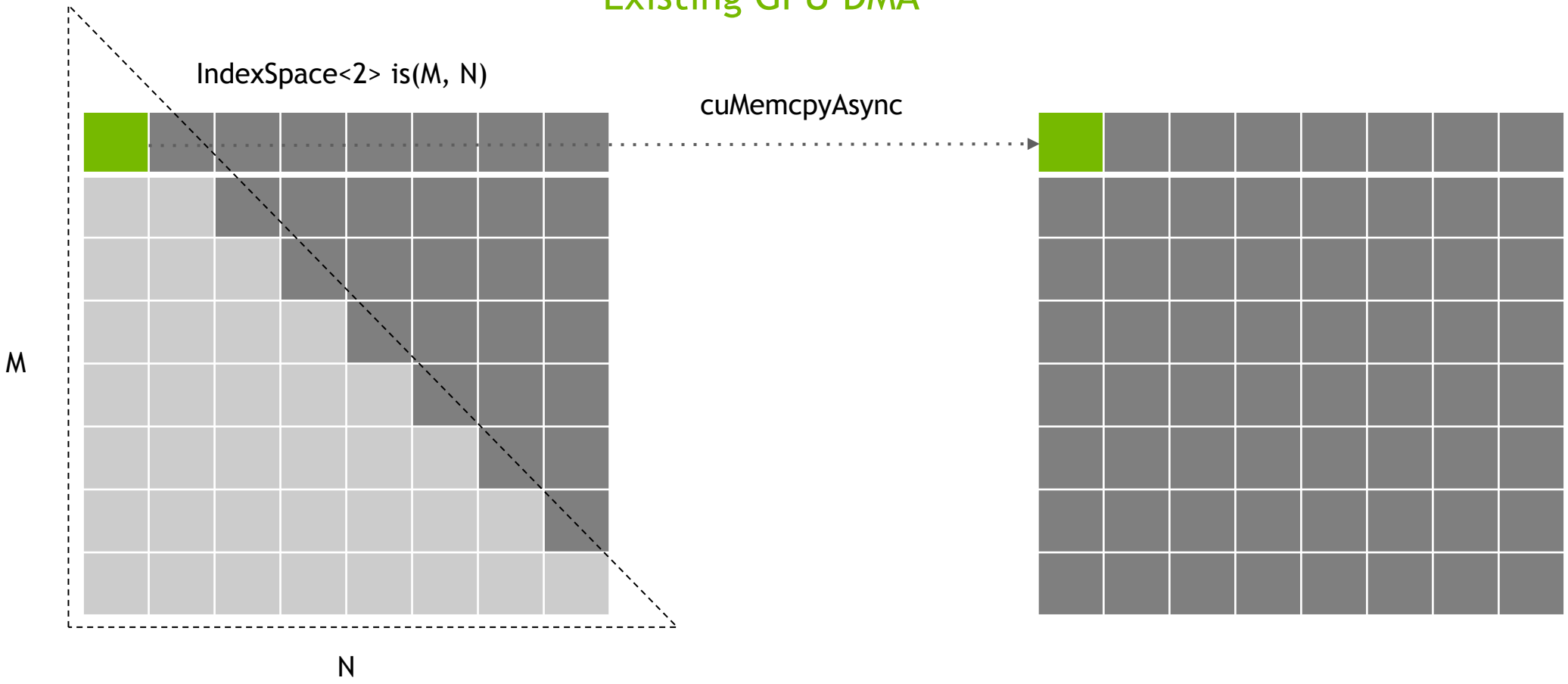
GPU CUDA DMA

Existing GPU DMA



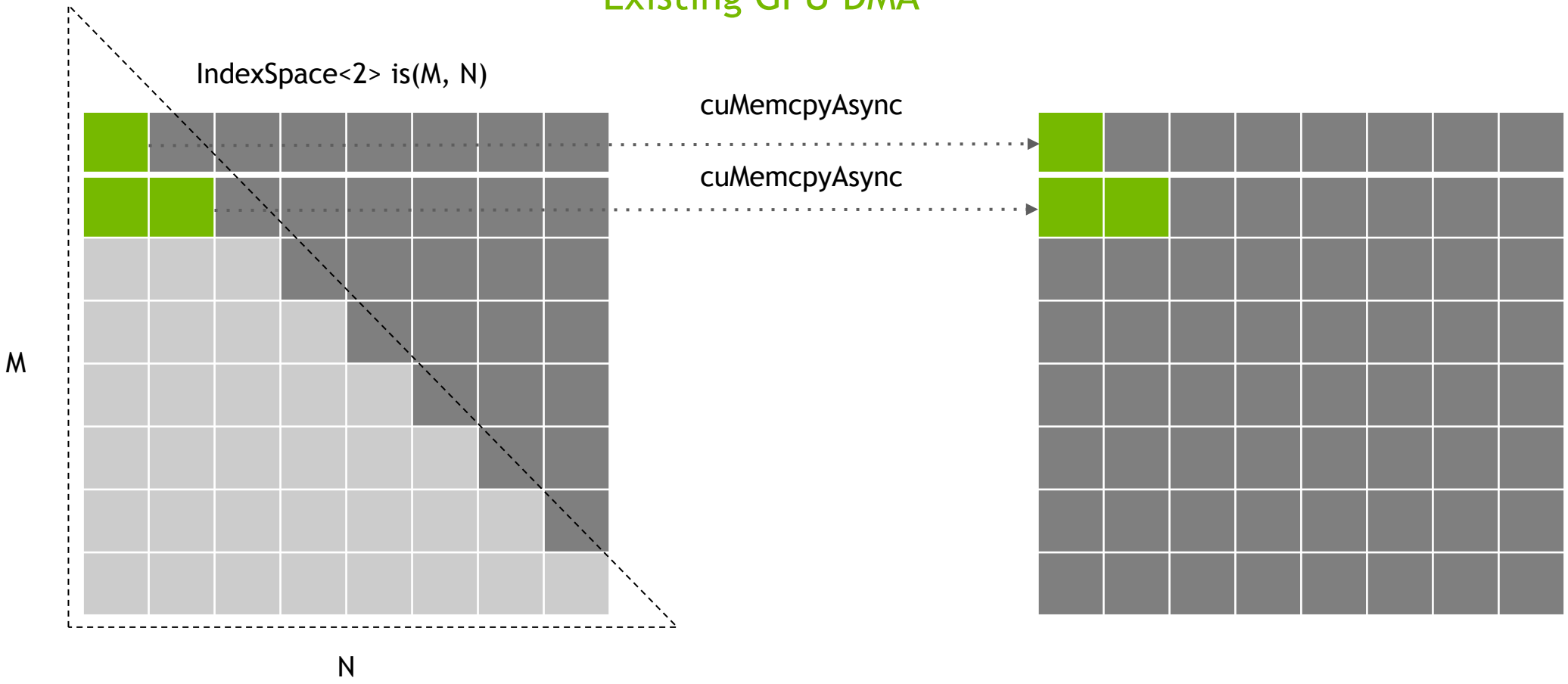
GPU CUDA DMA

Existing GPU DMA



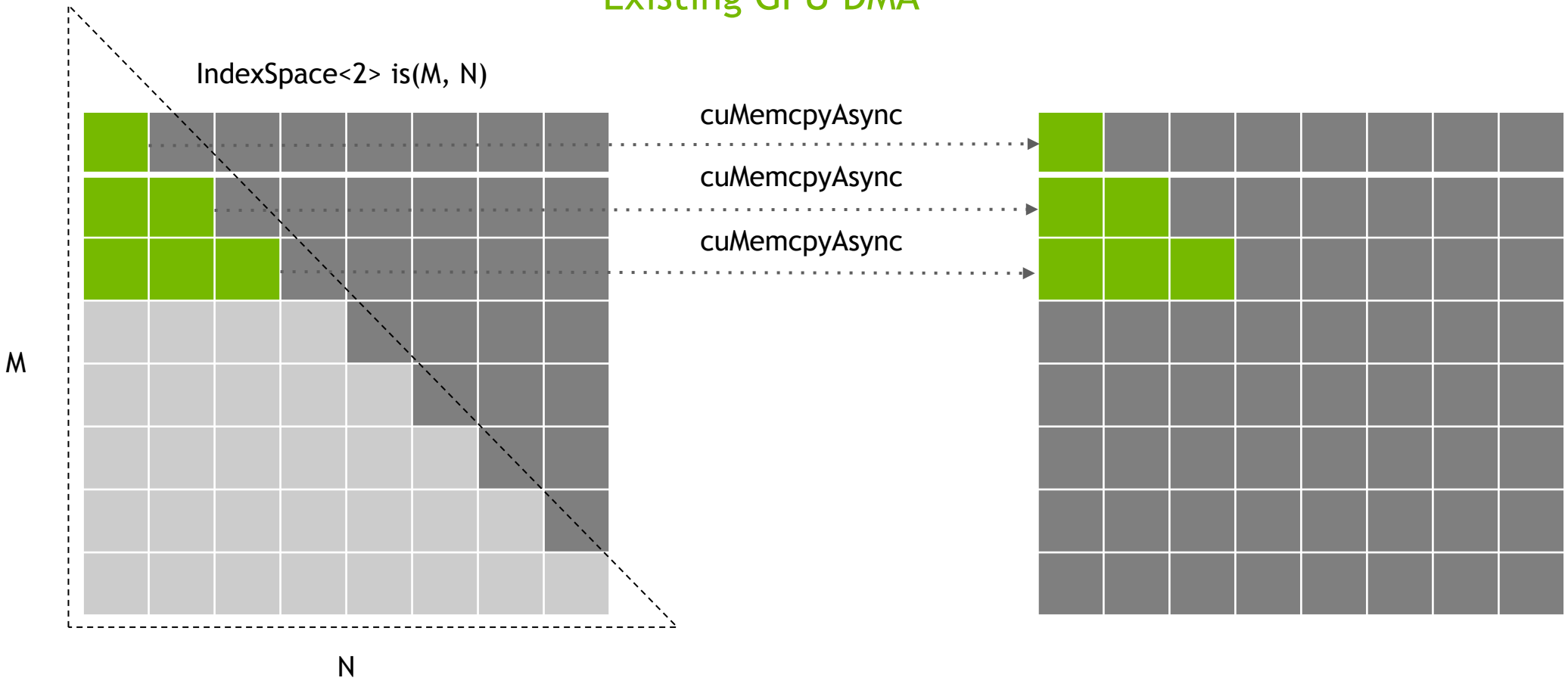
GPU CUDA DMA

Existing GPU DMA



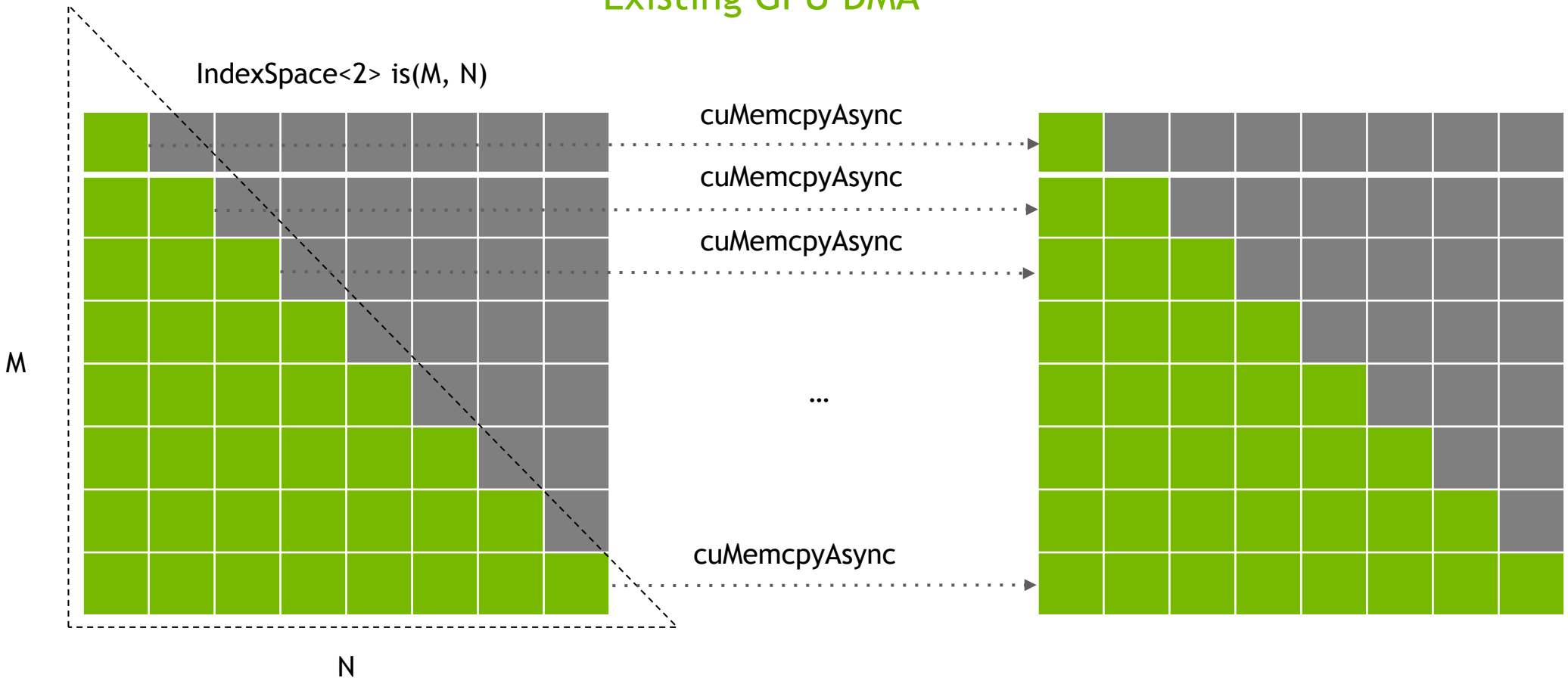
GPU CUDA DMA

Existing GPU DMA



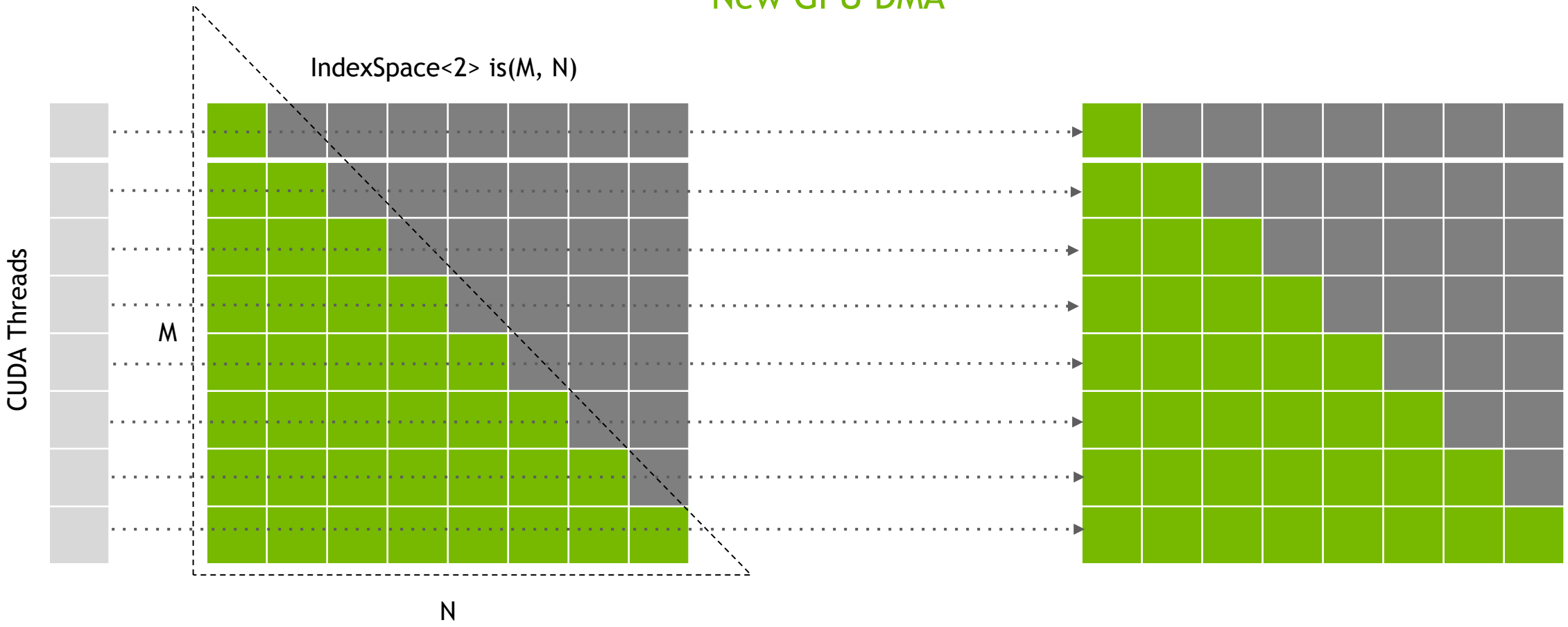
GPU CUDA DMA

Existing GPU DMA



GPU CUDA DMA

New GPU DMA



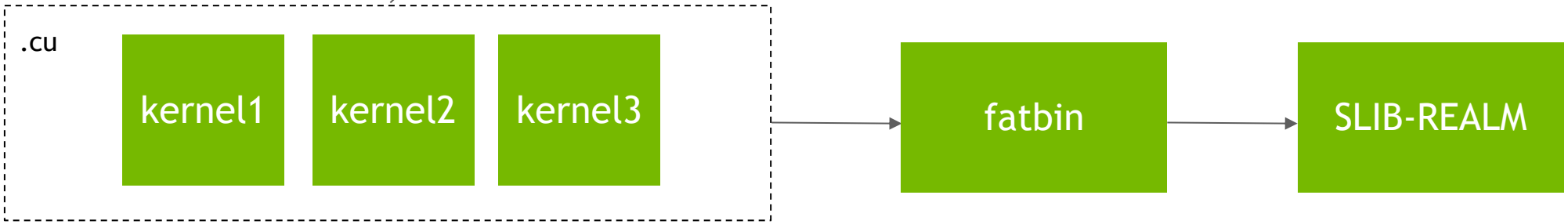
GPU CUDA DMA

2022 Q4…

- How do we do it?

…runtime/realm/cuda/cuda_memcpy.cu

```
template<typename T, size_t N, typename Offset_t = size_t>
static __device__ void memcpy_affine_batch(
   Realm::Cuda::AffineCopyPair<N, Offset_t> *info, size_t nrects) {
   …
}
```



Profiling

2023Q1…

- More profiling work…
  - Consolidate various profiling infrastructures in Realm

UCX Network Module

2022 Q1-Q4

- Motivation to add UCX
  - UCX is a lower-level abstraction than other APIs used by Realm
  - Have fewer external dependencies and more unified stack
    - NCCL/NVSHMEM/DASK use UCX too
  - Better support for future Realm requirements?

- Future plans
  - Finish functional and performance testing and merge into main branch
  - Provide support for prioritized communications
  - Provide support for elasticity

REALM

Long-term (6-24 months)

- Affine structured-unstructured indirect copies
- Improve scalability
- Realm automatic resource discovery
  - +Programmatic API
- Realm hardening

REALM HARDENING

Short-term (3-6 months)

- Improve test coverage
  - ~69% code coverage and ~25% branch coverage today

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	runtime/realm/	Lines:	23200	33210	69.9 %
Date:	2021-12-14 15:53:08	Branches:	26631	105464	25.3 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %				

File	Lines			Branches	
activemsg.cc		75.8 %	347 / 458	46.3 %	201 / 434
activemsg.h		100.0 %	10 / 10	- %	0 / 0
activemsg.inl		85.0 %	164 / 193	46.2 %	403 / 872
atomics.inl		96.4 %	53 / 55	75.0 %	6 / 8
bgwork.cc		82.0 %	264 / 322	41.9 %	198 / 472
bgwork.h		100.0 %	1 / 1	- %	0 / 0
bgwork.inl		100.0 %	27 / 27	50.0 %	6 / 12
bytearray.inl		96.5 %	82 / 85	63.5 %	33 / 52
circ_queue.inl		80.7 %	92 / 114	36.7 %	61 / 166
cmdline.cc		54.4 %	86 / 158	37.6 %	53 / 141
cmdline.h		83.3 %	5 / 6	- %	0 / 0
cmdline.inl		67.9 %	57 / 84	28.7 %	47 / 164
codedesc.cc		67.7 %	128 / 189	32.0 %	63 / 197
codedesc.h		100.0 %	5 / 5	- %	0 / 0
codedesc.inl		81.4 %	219 / 269	48.9 %	176 / 360
cuda/cuda_internal.cc		73.8 %	569 / 771	39.7 %	480 / 1209
cuda/cuda_internal.h		75.0 %	15 / 20	- %	0 / 0
cuda/cuda_module.cc		43.5 %	664 / 1528	21.2 %	685 / 3238

REALM HARDENING

Long-term (6-24 months)

What we want to achieve?

- Increase developer productivity
  - e.g. time/ease to test/debug/profile

- Maintain/Increase the quality bar
  - by increasing the test code coverage
  - by improving our code-review process (static analysis..)
  - by improving an integration with CI
    - more functional/performance test

REALM

Projects under discussion

- Elasticity

REALM ELASTICITY

Projects under discussion

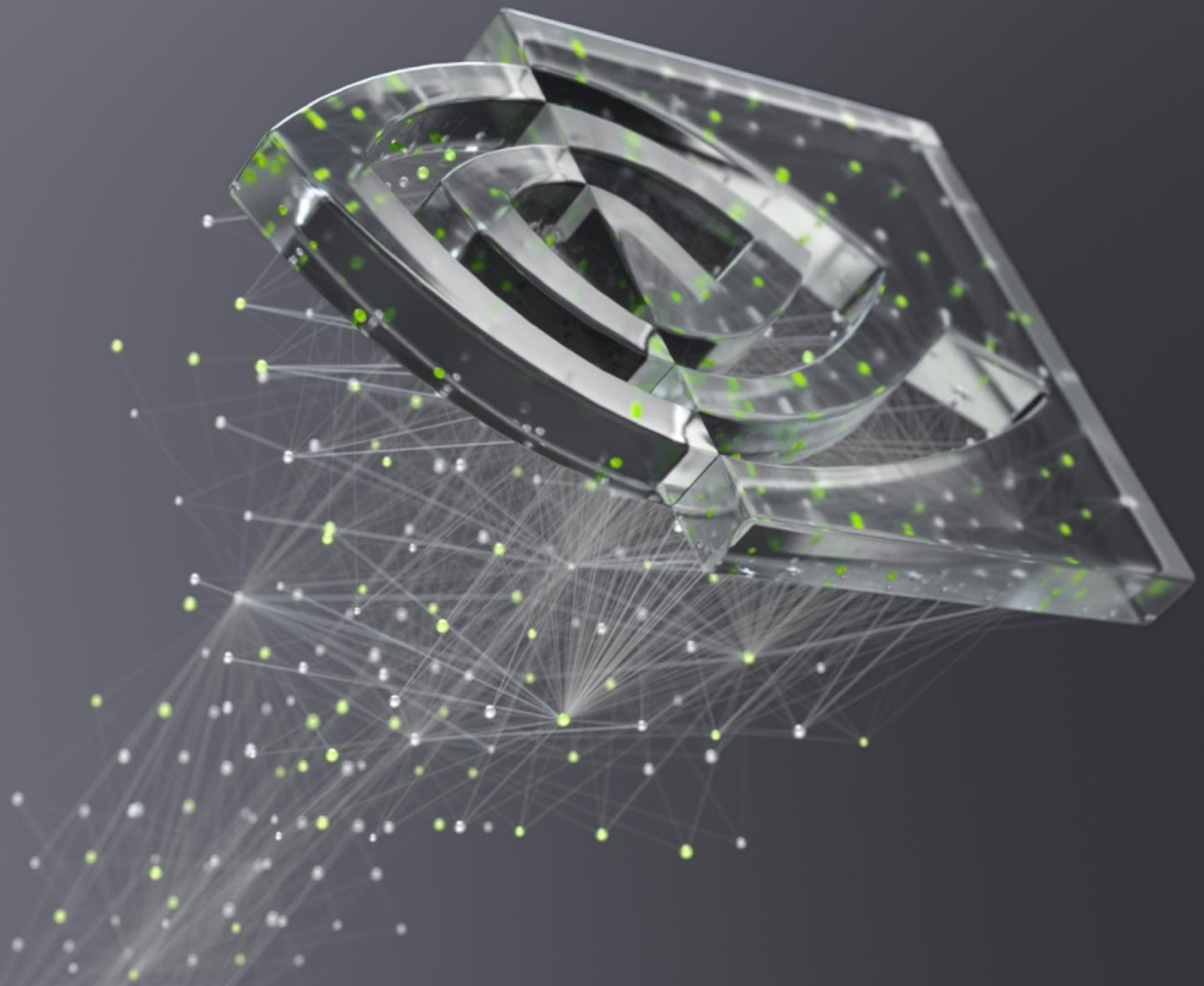
- Existing requests
  - Dynamically grow/shrink realm's machine model
  - Have interfaces to get clients notified when resources are added/removed

- Can we mock up a cunumeric example that wants to start with a few nodes and then require realm to grow resources?
- Question to audience -
  - Do you have cases that would benefit from elasticity?

REALM

Q/A

- Thank you!



NVIDIA®