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
• What are indirect copies?

```python
import numpy as np
x = ...
y = ...
ind = [P0, P1, P2..]
x = y[ind]
```
• What are indirect copies?

```python
import numpy as np
x = ...
y = ...
ind = [P0, P1, P2..]
x = y[ind]
```
What is an affine transformation?
- Any transformation that preserves collinearity.

\[
\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -100 \\ -100 \end{bmatrix}
\]
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.
• 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.

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

$$A \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} x + b$$

```
import numpy as np
x = ...
y = ...
...
x = y[::2]
```
Legate creates separate tasks to remove affine transformations
- Github Issue#705

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

```cpp
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

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

```cpp
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;
```

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

```cpp
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;
```

```cpp
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
• 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

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

```
../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);
```
PROFILING
Nsight
PROFILING

Nsight
REALM
Short-term (3-6 months)

- CUDA DMA
- Profiling
- UCX Network Module
• 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](https://example.com/issue621)
GPU CUDA DMA
Existing GPU DMA

IndexSpace<2> is(M, N)
IndexSpace<2> is(M, N)

GPU CUDA DMA
Existing GPU DMA

GPU D2D Copy
GPU CUDA DMA

Existing GPU DMA

IndexSpace<2> is(M, N)

cuMemcpyAsync
GPU CUDA DMA

Existing GPU DMA

IndexSpace<2> is(M, N)

M

N

cuMemcpyAsync

cuMemcpyAsync
GPU CUDA DMA

Existing GPU DMA

IndexSpace<2> is(M, N)

cuMemcpyAsync

cuMemcpyAsync

cuMemcpyAsync
GPU CUDA DMA

Existing GPU DMA

IndexSpace<2> is(M, N)

cuMemcpyAsync

...
### GPU CUDA DMA

**IndexSpace<2> is(M, N)**

<table>
<thead>
<tr>
<th>CUDA Threads</th>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GPU CUDA DMA

2022 Q4...

• How do we do it?

```cpp
...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

<table>
<thead>
<tr>
<th>File</th>
<th>Lines Coverage</th>
<th>Braches Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>activemsg.cc</td>
<td>75.8%</td>
<td>46.3%</td>
</tr>
<tr>
<td>activemsg.h</td>
<td>100.0%</td>
<td>0%</td>
</tr>
<tr>
<td>activemsg.inl</td>
<td>85.0%</td>
<td>42.6%</td>
</tr>
<tr>
<td>atomiccs.inl</td>
<td>96.4%</td>
<td>75.0%</td>
</tr>
<tr>
<td>bsgwork.cc</td>
<td>82.0%</td>
<td>41.9%</td>
</tr>
<tr>
<td>bsgwork.h</td>
<td>100.0%</td>
<td>0%</td>
</tr>
<tr>
<td>bsgwork.inl</td>
<td>100.0%</td>
<td>6.0%</td>
</tr>
<tr>
<td>bytearray.inl</td>
<td>96.5%</td>
<td>36.5%</td>
</tr>
<tr>
<td>circ_queue.inl</td>
<td>80.7%</td>
<td>36.7%</td>
</tr>
<tr>
<td>cmdline.cc</td>
<td>54.4%</td>
<td>37.6%</td>
</tr>
<tr>
<td>cmdline.h</td>
<td>83.3%</td>
<td>0%</td>
</tr>
<tr>
<td>cmdline.inl</td>
<td>67.9%</td>
<td>28.7%</td>
</tr>
<tr>
<td>codescsc.cc</td>
<td>67.7%</td>
<td>32.0%</td>
</tr>
<tr>
<td>codescsc.h</td>
<td>100.0%</td>
<td>0%</td>
</tr>
<tr>
<td>cuda/cuda_internal.cc</td>
<td>81.4%</td>
<td>48.9%</td>
</tr>
<tr>
<td>cuda/cuda_internal.h</td>
<td>73.8%</td>
<td>39.7%</td>
</tr>
<tr>
<td>cuda/cuda_module.cc</td>
<td>43.5%</td>
<td>21.2%</td>
</tr>
</tbody>
</table>

Legend: low: < 75.0% medium: >= 75.0% high: >= 90.0%
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!