# Profiling and Debugging Tools

Wonchan Lee

# Legion Prof

- Profiling tool for Legion and Regent applications

# Legion Prof

- Profiling tool for Legion and Regent applications
- Tracks timing information about tasks, copies, and instances
  - When a task started and when it finished
  - When a region instance was created and when it was destroyed
  - How long a low-level copy from one memory to another executed

# Legion Prof

- Profiling tool for Legion and Regent applications
- Tracks timing information about tasks, copies, and instances
  - When a task started and when it finished
  - When a region instance was created and when it was destroyed
  - How long a low-level copy from one memory to another executed
- Also tracks dependencies among tasks, copies, and instances
  - Which runtime tasks were initiated by which task
  - Which region instances were created by which task on which memory
  - Which copies were initiated by which task

# Legion Prof

- Profiling tool for Legion and Regent applications
- Tracks timing information about tasks, copies, and instances
  - When a task started and when it finished
  - When a region instance was created and when it was destroyed
  - How long a low-level copy from one memory to another executed
- Also tracks dependencies among tasks, copies, and instances
  - Which runtime tasks were initiated by which task
  - Which region instances were created by which task on which memory
  - Which copies were initiated by which task
- Generates a trace log for post-processing
  - Logging enabled when the application is passed `-hl:prof N`
  - Requires the logging level to be at least 2 (`-level legion_prof=2`)
  - Trace visualizer takes this log and generates a stand-alone visualizer

# Legion Prof Trace Visualizer

- Used to be one static SVG file
  - SVG is a vector image format supported by all major web browsers
  - Often becomes too long to fit on one page
  - Very slow to navigate with large numbers of tasks

# Legion Prof Trace Visualizer

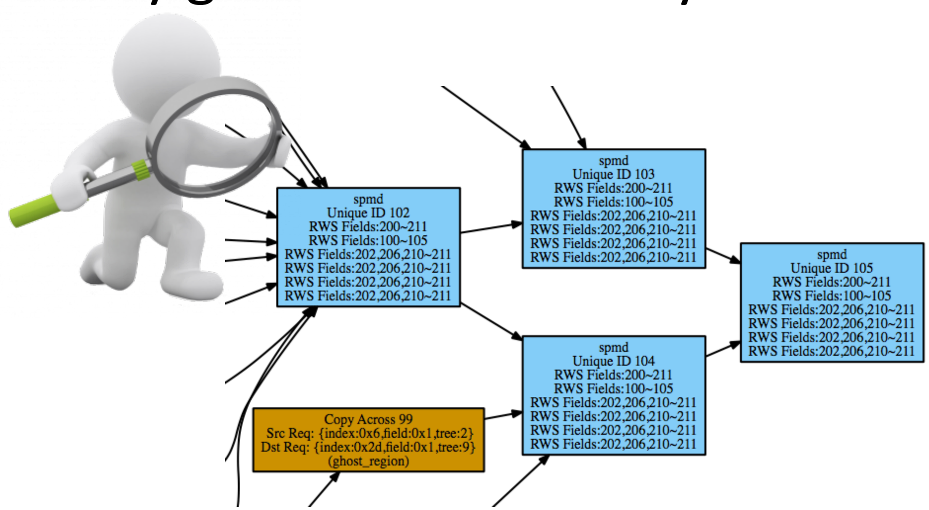
- Used to be one static SVG file
  - SVG is a vector image format supported by all major web browsers
  - Often becomes too long to fit on one page
  - Very slow to navigate with large numbers of tasks
- Developed a new web-based visualizer
  - Javascript program that dynamically renders SVG for trace data
  - Runs as a stand-alone visualizer (legion_prof.html)
  - Can zoom in and out for a particular time span
  - Renders only the objects that are big enough to be readable
  - Supports search with regular expressions

# Legion Prof Trace Visualizer

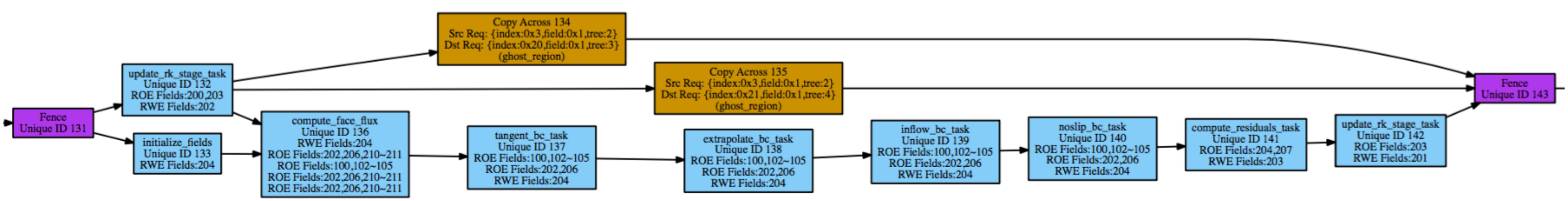
## Live Demo

# Operation Graphs

- Show data dependencies between tasks
- Hierarchically generated for every non-leaf task




Top-level task

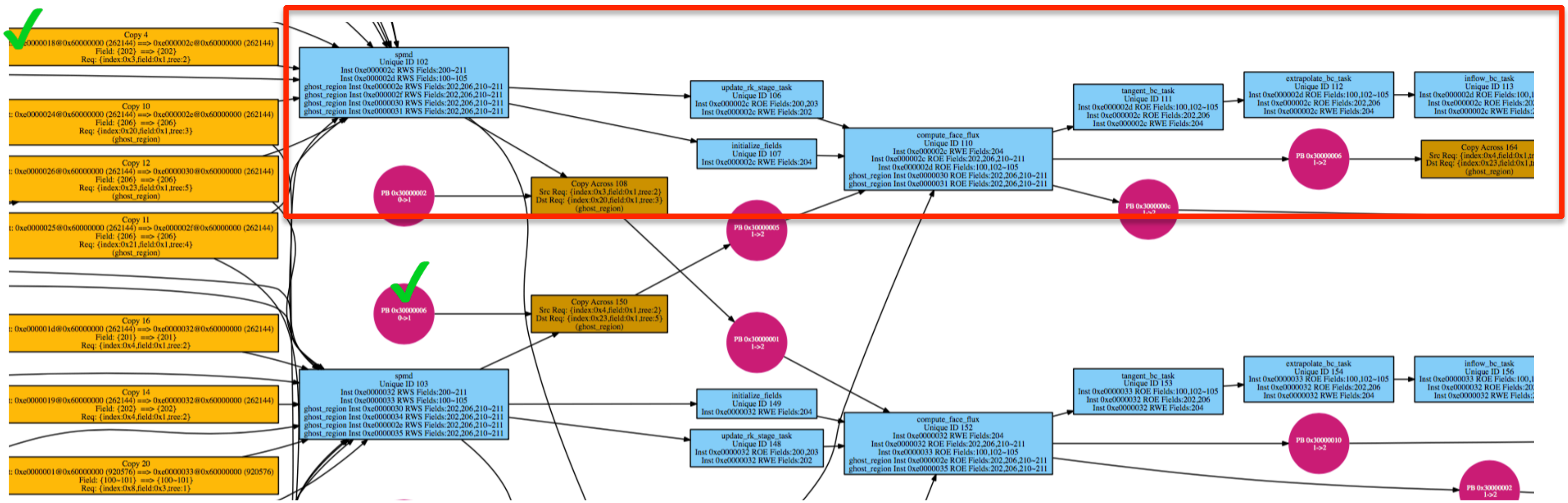



spmd task 102

# Operation Graphs

- Show data dependencies between tasks
- Hierarchically generated for every non-leaf task
  - c.f. event graphs show all operations and events:

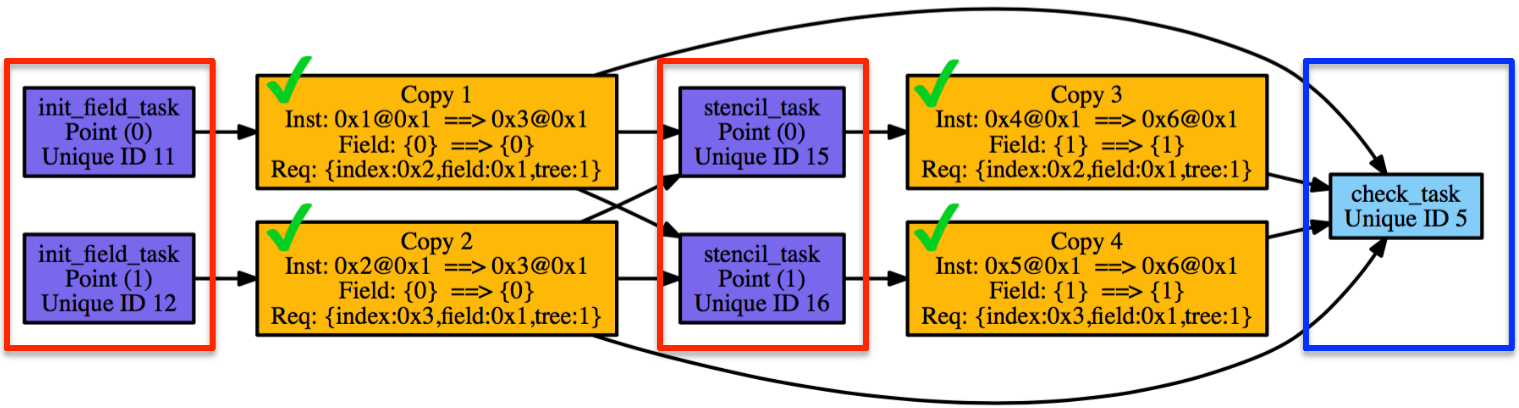
child task is flattened




more operations rendered

# Event Graphs

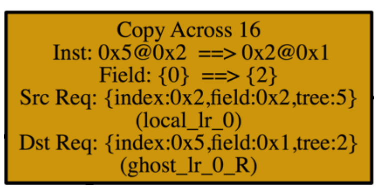
- Show all event dependencies in operations



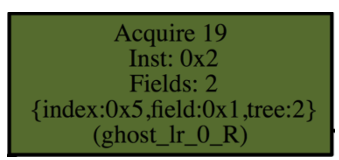

point tasks from indexspace task launch

tasks from single task launch

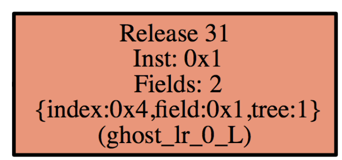
copy events inserted by runtime



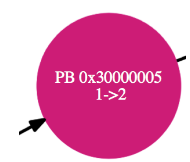

user-triggered copy operations




acquire operations



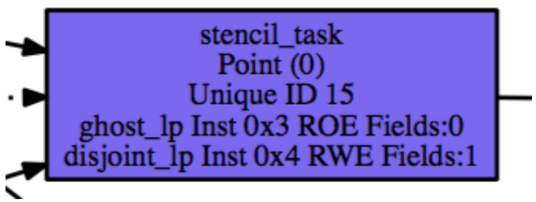

release operations




phase barriers

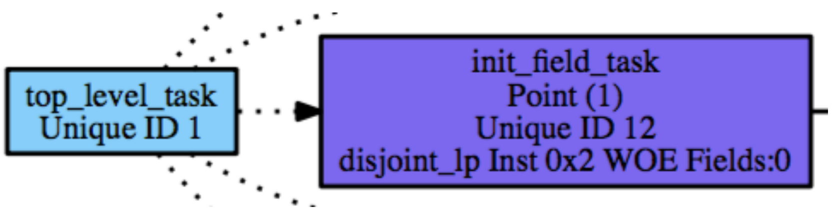
# Event Graphs

- Can show more information
  - A list of accessed physical instances with their privileges



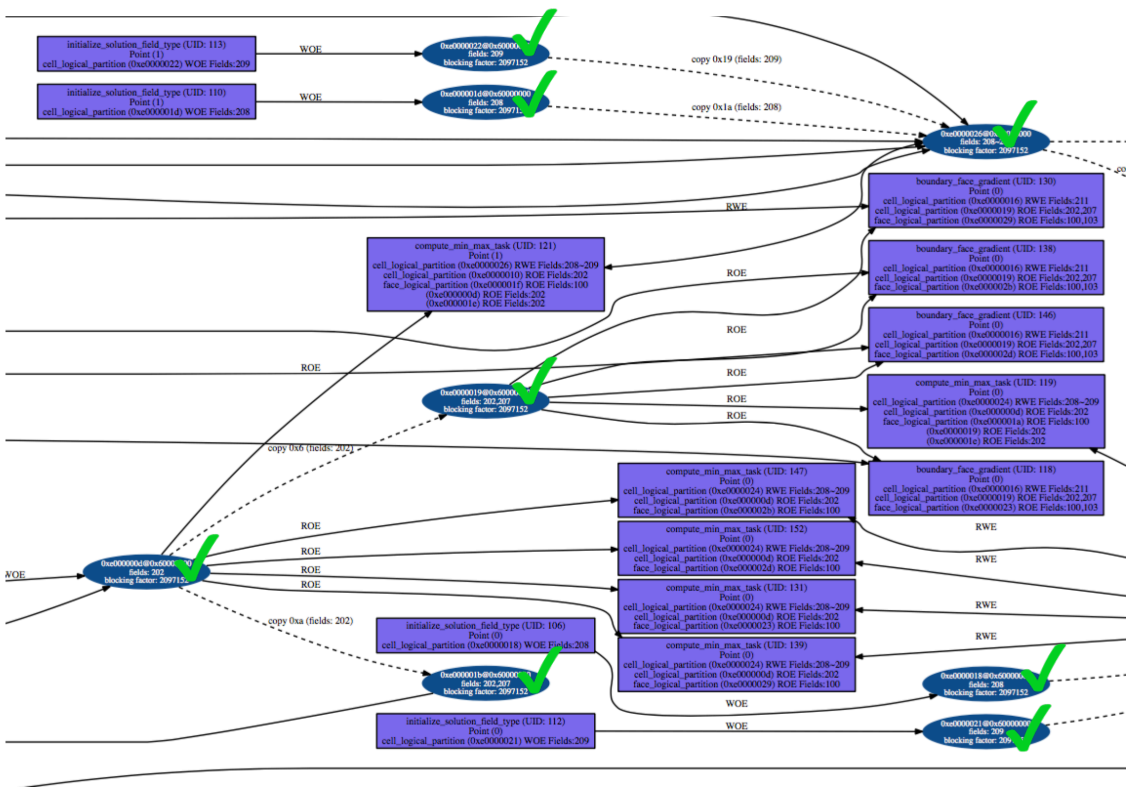

| Privileges | Coherence |
|---|---|
| RO: Read-Only | E: Exclusive |
| WO: Write-Only | S: Simultaneous |
| RW: Read-Write | A: Atomic |
| Red: Reduction | R: Relaxed |

  - Implicit dependences between parent tasks and their children



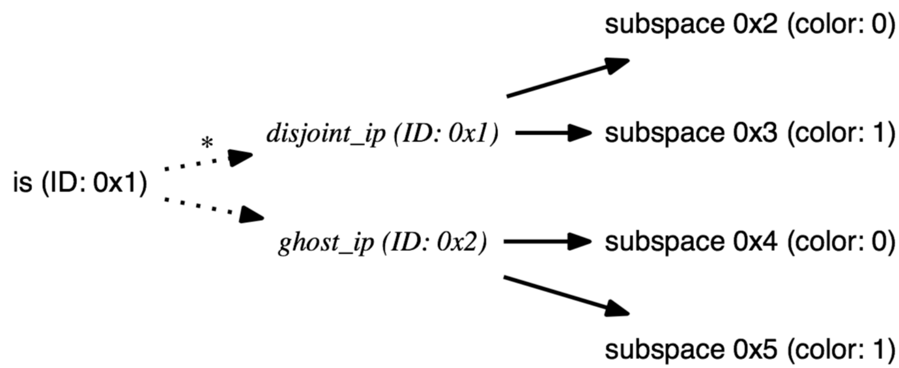
# Instance Graphs

- Provide an instance-centric view of operations
  - Which instances are used by which tasks
  - Which instances are copied to which

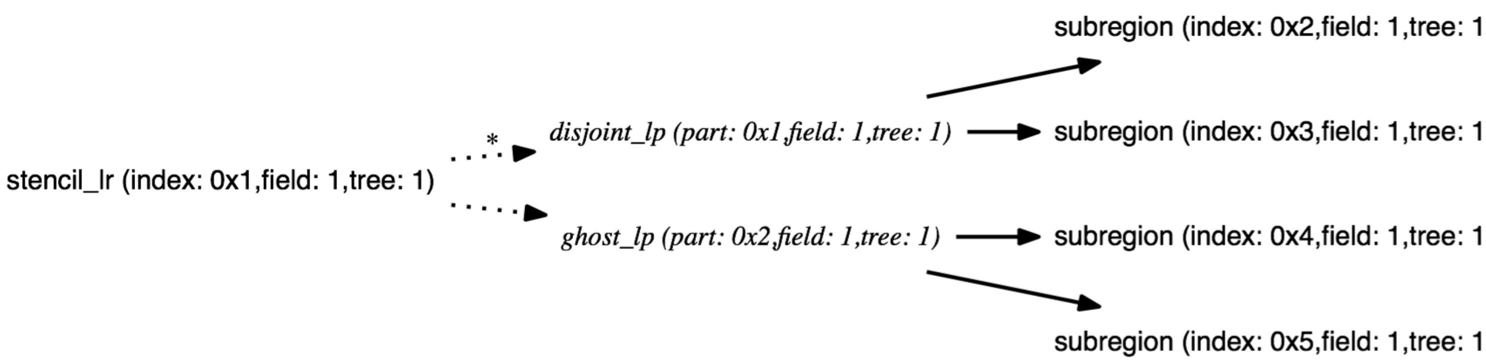


# Partitioning Graphs

- Show index spaces and region trees
  - Edge labelled '*' means a disjoint partition




Indexspace tree




Region tree

# Disjointness Checks in Legion

- Verify the disjointness of partitions claimed to be disjoint
  - Without checks, runtime and compiler just believe the code

```
// Legion C++  API
IndexPartition create_index_partition(Context ctx, IndexSpace parent,
                                      const Coloring &coloring,
                                      bool disjoint,
                                      int part_color = -1);
-- Regent
var rp = partition(disjoint, r, some_coloring)
```

  - Legion Runtime will check the disjointness with flag – `hl:disjointness`

```
$ ./partitioning -hl:disjointness
Running daxpy for 1024 elements...
Partitioning data into 3 sub-regions...
[0 - 1] {ERROR}{runtime}: ERROR: colors 0 and 1 of partition 1 are not disjoint when they
are claimed to be!
Assertion failed: (false), function create_index_partition, file /Users/wclee/Workspace/
stanford/projects/legion//runtime/runtime.cc, line 5348.
```

# Privilege Checks in Legion

- Check if the task abides by privileges stated in requirements
  - Enabled optionally by compile flag `-DPRIVILEGE_CHECKS`
  - E.g.

```
...
TaskLauncher launcher(...);
launcher.add_region_requirement(RegionRequirement(r, READ_ONLY, EXCLUSIVE, r));
runtime->execute_task(ctx, launcher);
...
void init_field_task(...)
{
  ...
  auto acc = regions[0].get_field_accessor(FID_X).typeify<double>();
  ...
   acc.write(...);
  ...
}
```

```
$ ./privileges
Running daxpy for 1024 elements...
Initializing field 0...
PRIVILEGE CHECK ERROR IN TASK init_field_task: Need WRITE-DISCARD privileges but
only hold READ-ONLY privileges
Assertion failed: (false), function check_privileges, file /Users/wclee/Workspace/
stanford/projects/legion//runtime/accessor.h, line 160.
```

# Privilege Checks in Regent

- Regent type system guarantees all typed Regent programs do not violate privileges
  - No runtime overhead for privilege checks
  - E.g.

```
task check(r : region(int))
where reads(r)
do
  for e in r do @e = 10 end -- needs write privilege as well
end
```

```
$ regent.py check.rg
legion/language/src/regent/std.t:1113: Errors reported during typechecking.
check.rg:21: invalid privilege writes($r) for dereference of ptr(int32, $r)
  for e in r do @e = 10 end -- needs write privilege as well
                ^
```

# Bounds Checks in Legion

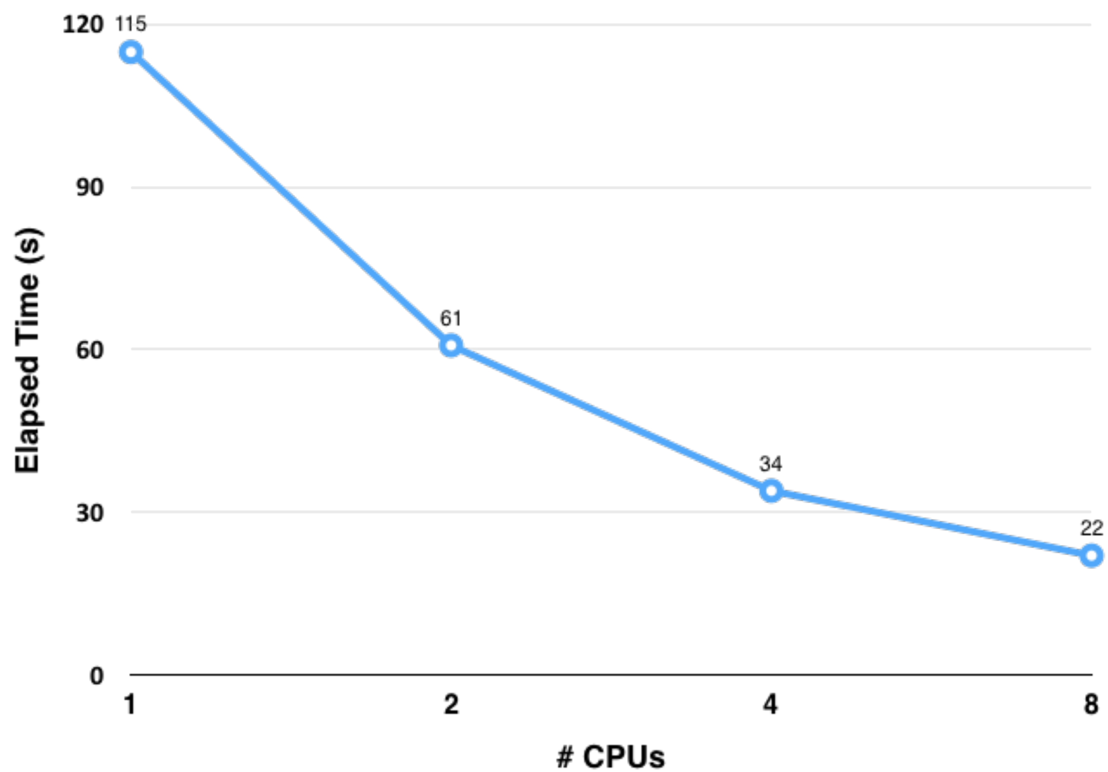
- Check if any region access was out of bounds
  - Only valid when the application used region accessors from the API
  - Enabled optionally by compile flag `-DBOUNDS_CHECKS`
  - E.g.

```
$ ./bounds
Running daxpy for 1024 elements...
Initializing field 0...
Initializing field 1...
Running daxpy computation with alpha 0.39646477...
BOUNDS CHECK ERROR IN TASK 3: Accessing invalid 1D point (1024)
Assertion failed: (false), function check_bounds, file /Users/wclee/Workspace/
stanford/projects/legion//runtime/runtime.cc, line 12368.
```

# Performance Tuning Example: Reducing Copies in MiniAero

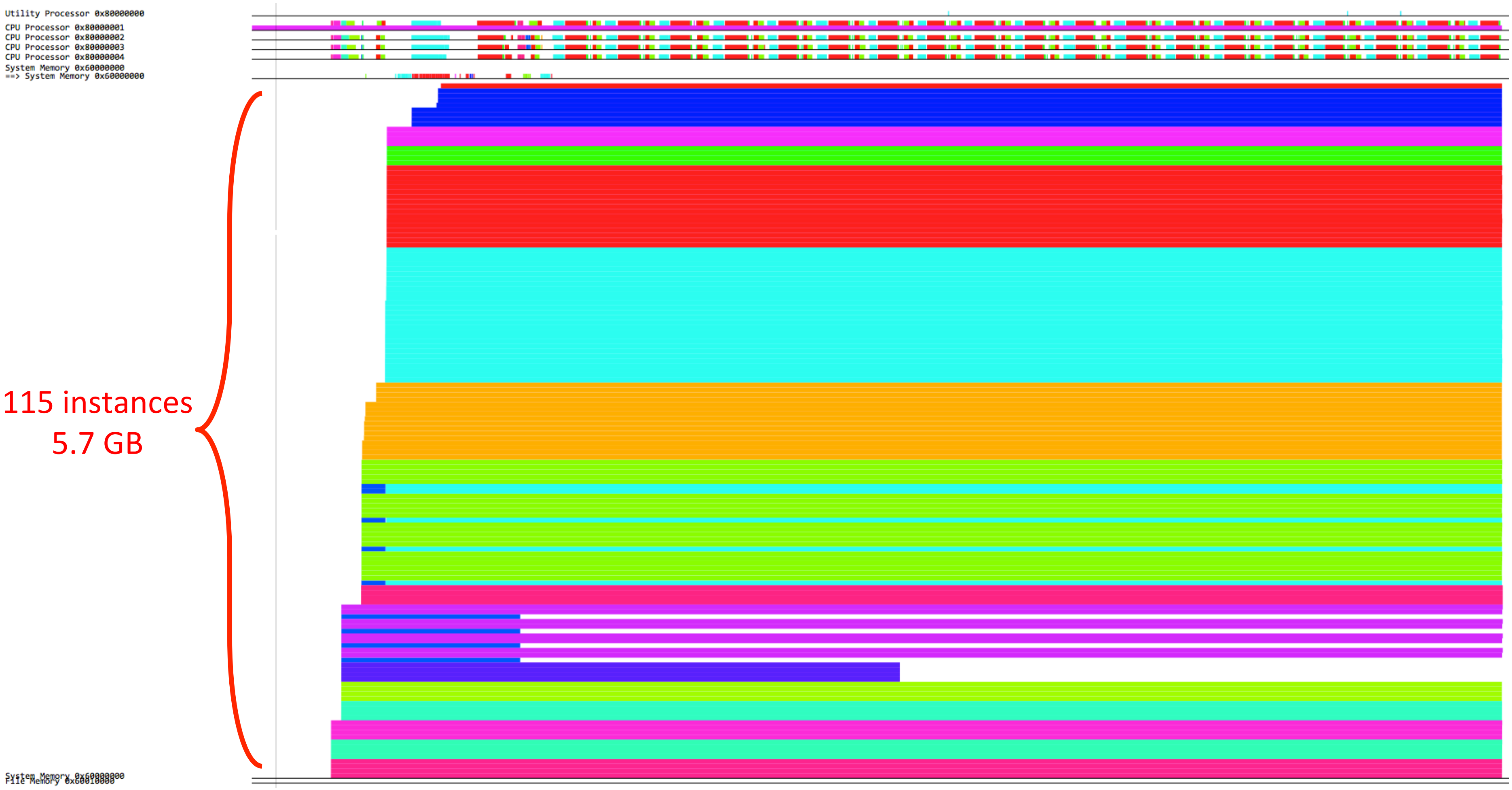
# Measuring Performance

- Mesh size: 2M cells, 7M faces (512x1024x4)
- Run 5 iterations with 1, 2, 4, and 8 cpus on a single node



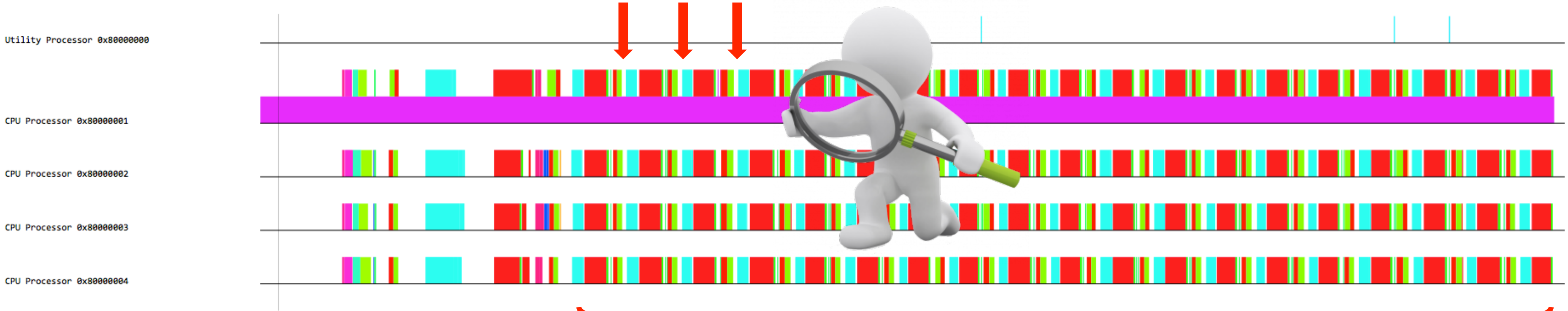

5x speed up on 8 CPUs
Is this the best speed-up we can get?

# Profiling MiniAero

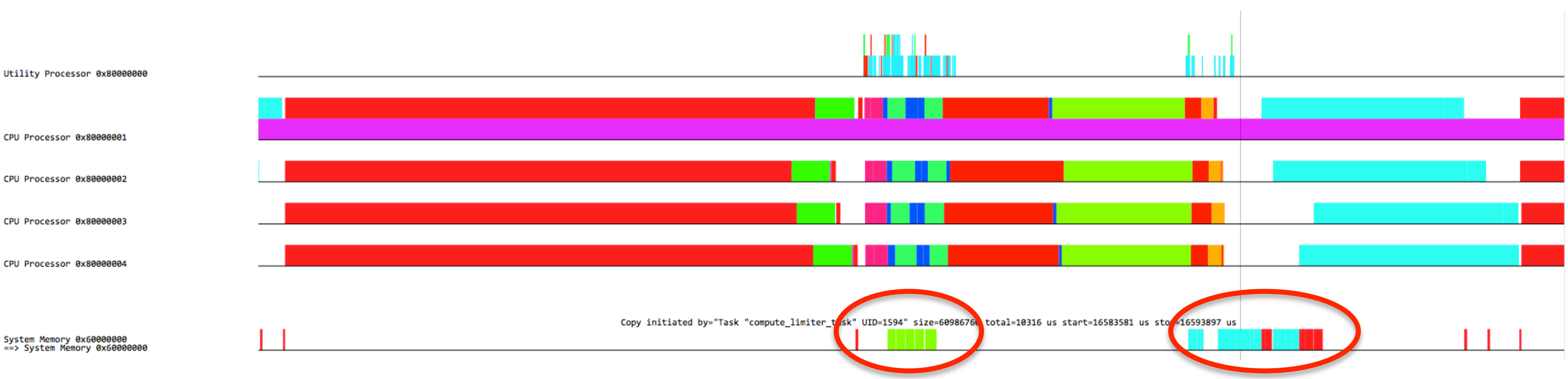


# Profiling MiniAero

Some gaps found between tasks



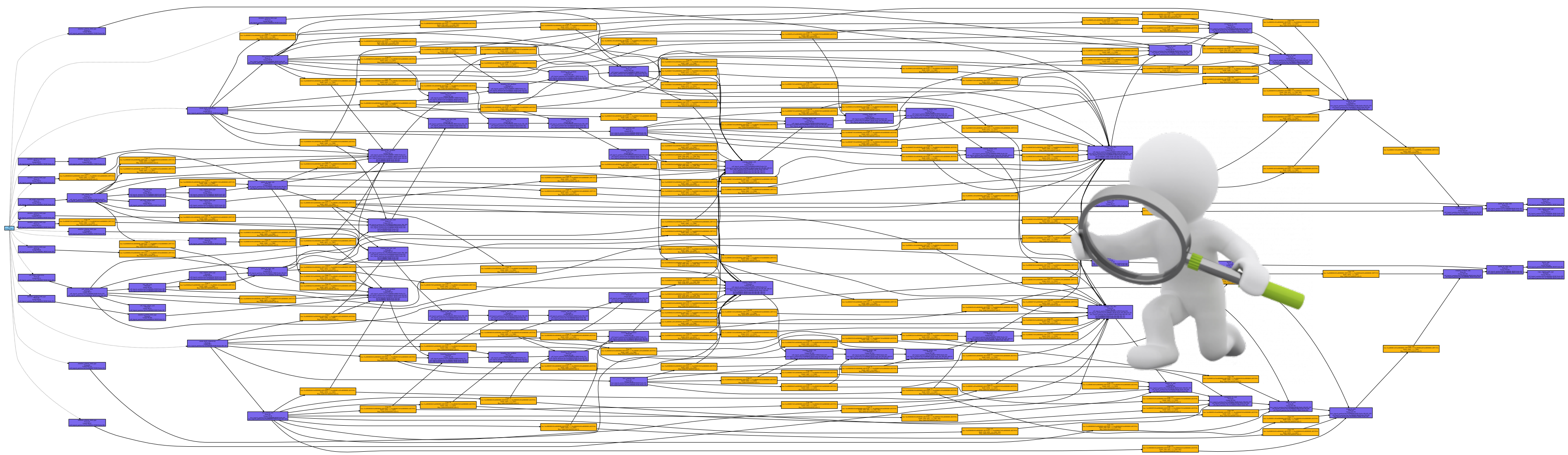

20 RK steps (5 * 4 steps)




low-level copies on the critical path

# Finding Copies in the Event Graph

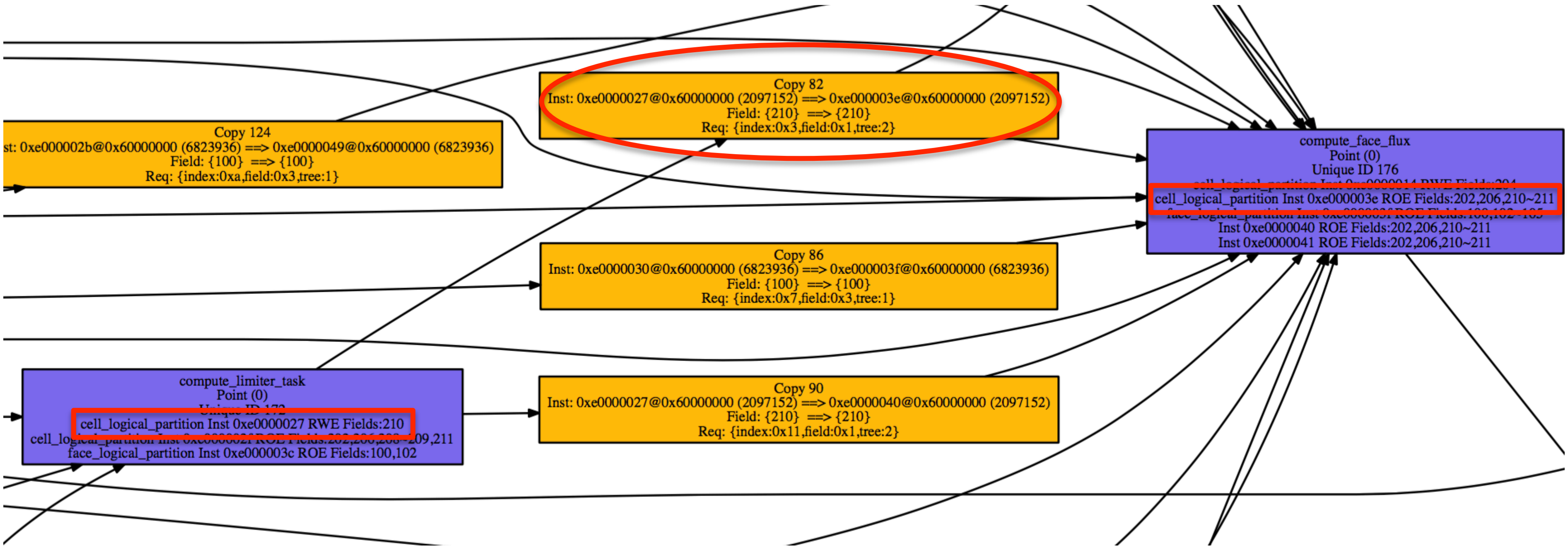
1 RK step on 2 partitions



84 tasks, 142 copies

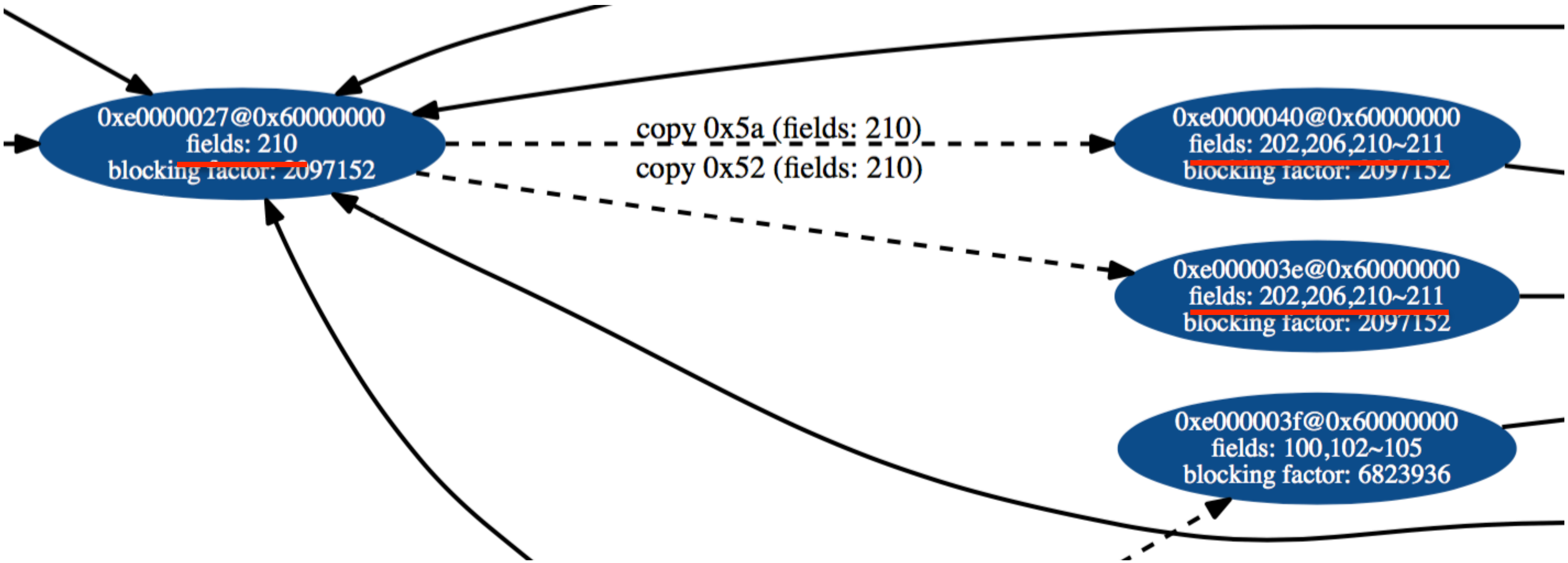
→ This shouldn't be the case as all updates are made only to disjoint partitions and all regions are mapped to the same memory

# Finding Copies in the Event Graph




mapped to two different instances
even though the first one is only a subset of the second

# Finding Copies in the Instance Graph

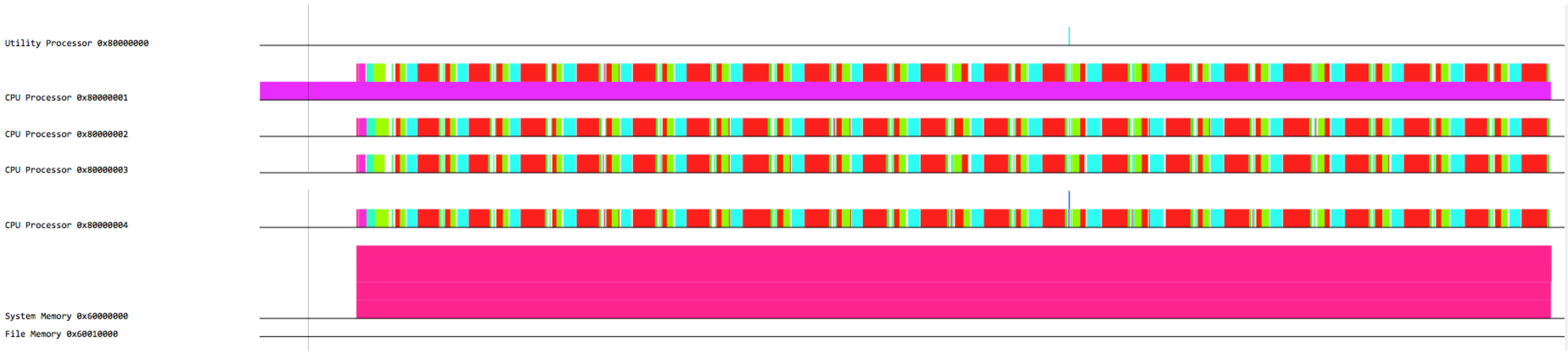



mapped to two different instances
even though the first one is only a subset of the second

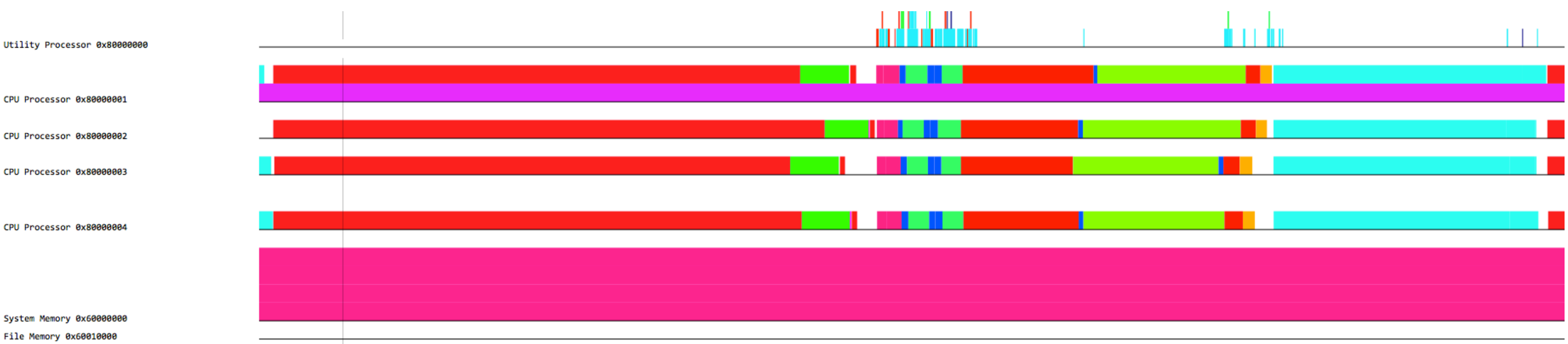
# Writing a Custom Mapper

- Creating instances with all necessary fields upfront
  - Add the fields to `additional_fields` of the region requirement
- Still needs some copies between ghost regions on one system memory to owned regions on the other memory
  - Choice remains between system and RDMA memory
- New Mapper API will allow us to map one region to a union of physical instances that satisfies the requirement

# Legion Prof Trace after Tuning

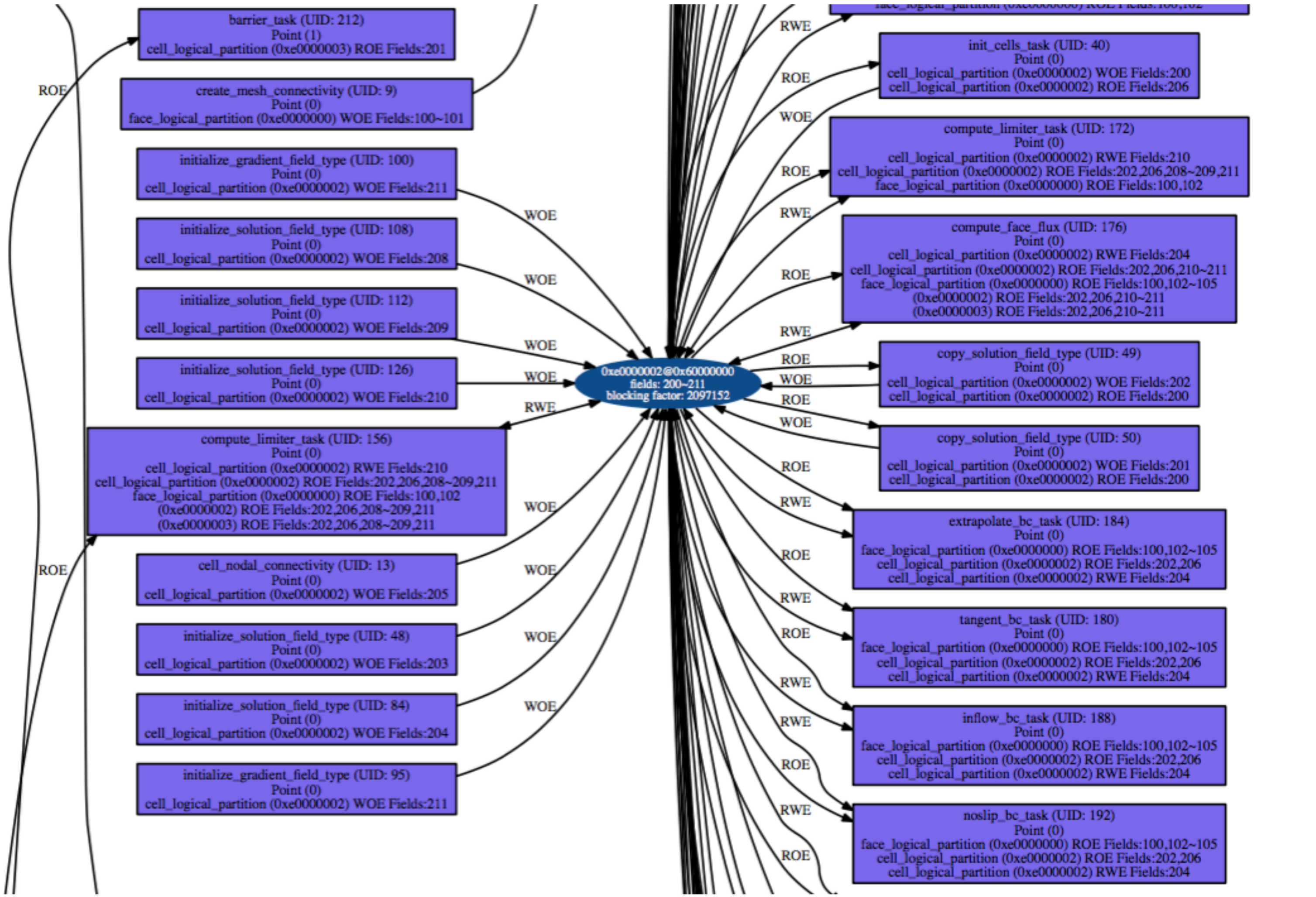



**Only 4 physical instances**
**No low-level copies exists**



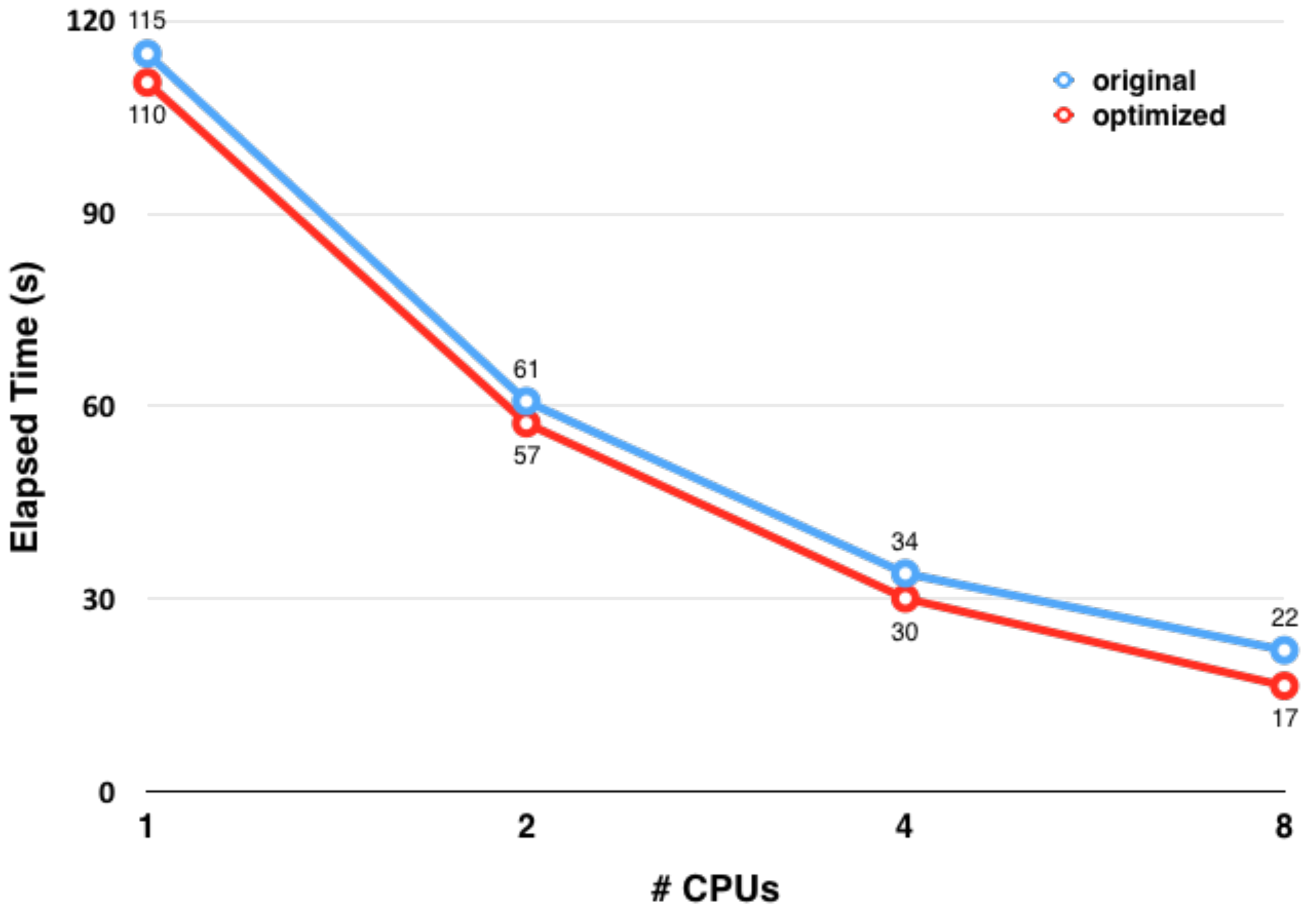

**Smaller gaps between tasks**

# Instance Graph after Tuning




**Tasks are now using the same instance**

# Performance after Tuning




We now get 6.5x speed up on 8 CPUs

# Questions?

# Bounds Checks in Regent

- Check if any region access or access to a statically sized array was out of bounds
  - Regent compiler inserts the bounds checks to the compiled code with flag `-fbounds-checks 1`
  - E.g.

```
task foo()
  var x : int[1] return x[1]
end
```

```
$ regent.py bounds.rg -fbounds-checks 1
assertion failed: array access to int32[5] is out-of-bounds
```