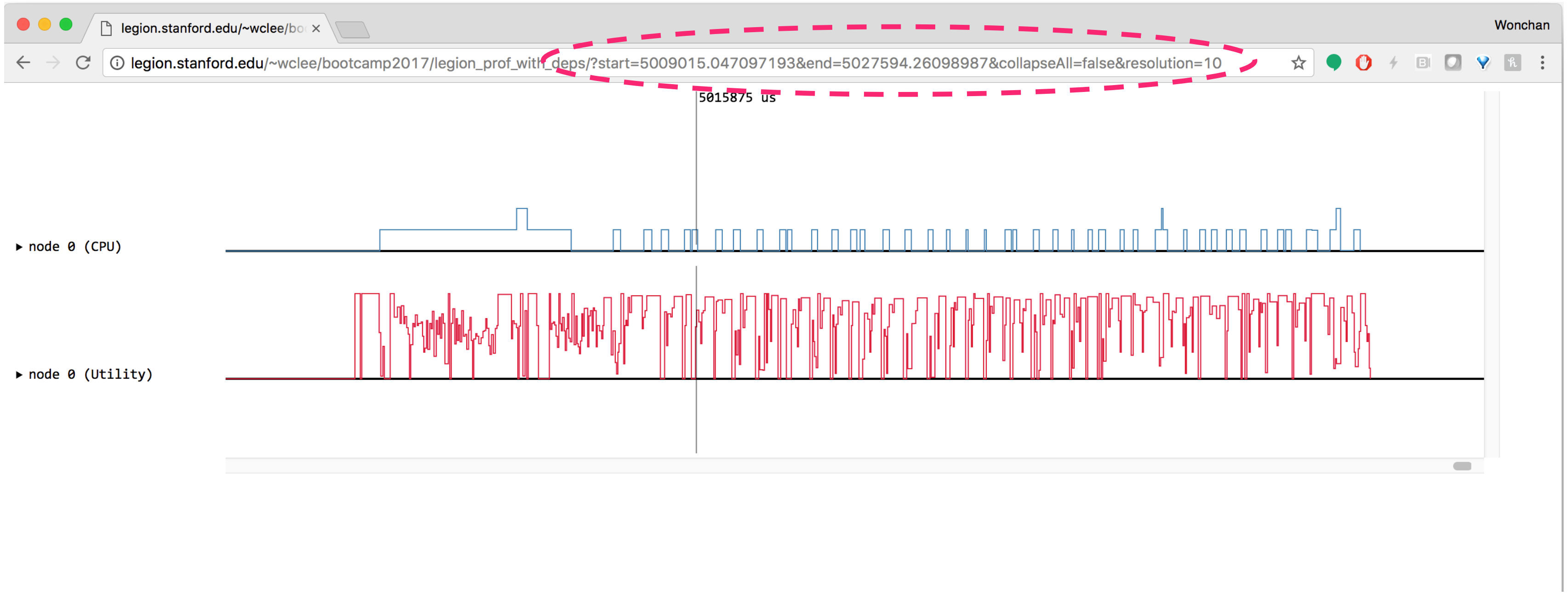
Features in Legion Prof

Wonchan Lee

URL String

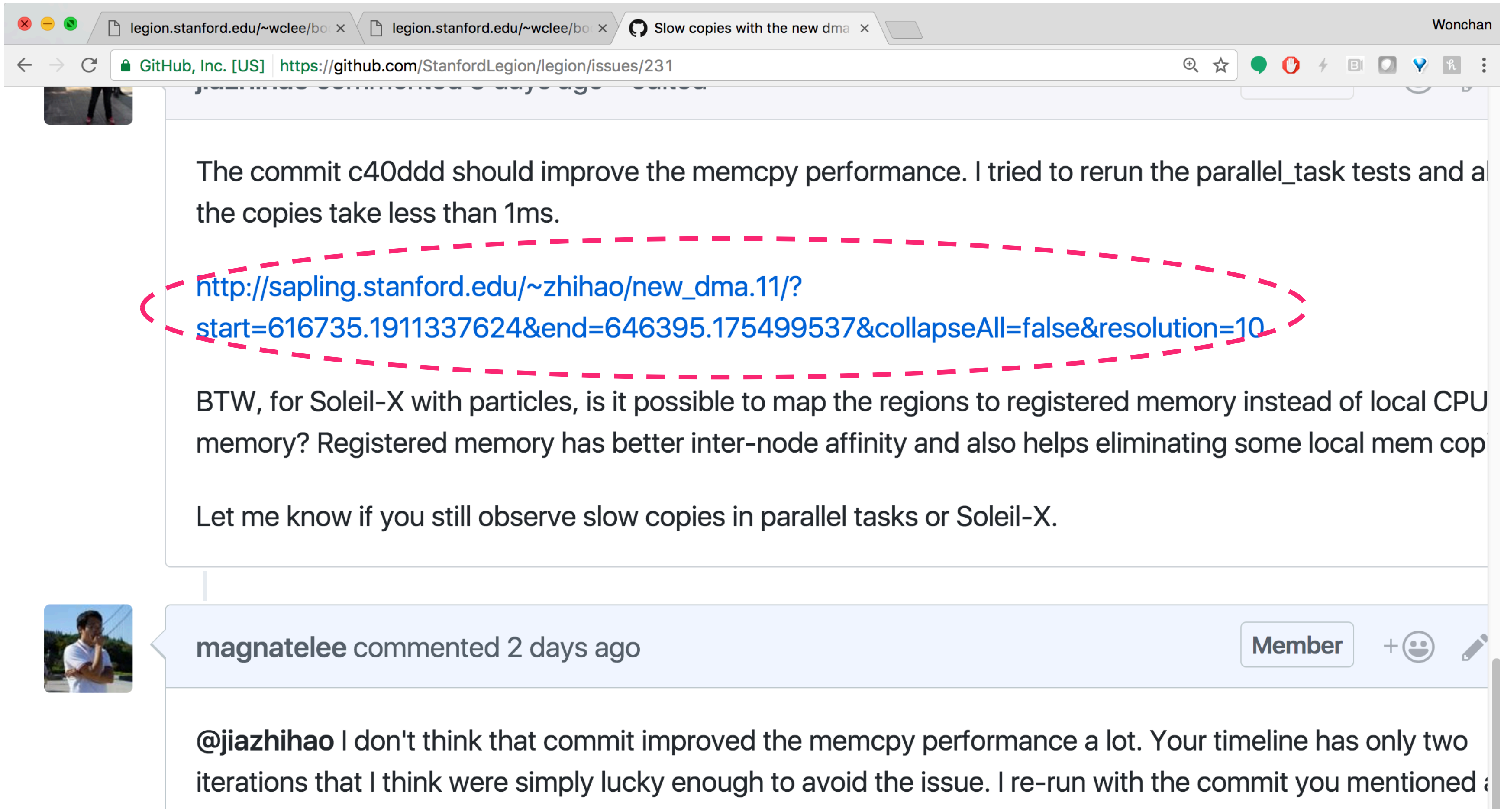
- Remembers which time period the viewer was showing



legion.stanford.edu/~wclee/bootcamp2017/legion_prof_with_deps/?start=5009015.047097193&end=5027594.26098987&collapseAll=false&resolution=10

URL String

- Useful to share selections of interest in timelines


legion.stanford.edu/~wclee/bo
legion.stanford.edu/~wclee/bo
Slow copies with the new dma
Wonchan
GitHub, Inc. [US]
https://github.com/StanfordLegion/legion/issues/231

The commit c40ddd should improve the memcpy performance. I tried to rerun the parallel_task tests and a
the copies take less than 1ms.

http://sapling.stanford.edu/~zhihao/new_dma.11/?start=616735.1911337624&end=646395.175499537&collapseAll=false&resolution=10

BTW, for Soleil-X with particles, is it possible to map the regions to registered memory instead of local CPU memory? Registered memory has better inter-node affinity and also helps eliminating some local mem cop

Let me know if you still observe slow copies in parallel tasks or Soleil-X.

magnatelee commented 2 days ago Member

@jiazhihao I don't think that commit improved the memcpy performance a lot. Your timeline has only two iterations that I think were simply lucky enough to avoid the issue. I re-run with the commit you mentioned

Zoom

- Keyboard shortcuts
  - **Ctrl-Alt -** or **1** – Zoom out (y-axis)
  - **Ctrl-Alt +** or **2** – Zoom in (y-axis)
  - **Ctrl -** or **3** – Zoom out (x-axis)
  - **Ctrl +** or **4** – Zoom in (x-axis)
  - **Ctrl 0** or **0** – Reset zoom (x-axis)

- Drag-select to zoom in for a particular range
  - Will show only the time span if CMD is pressed
  - Can be undone with **U**

Search

- Find matches on the names of tasks with regex

- Keyboard shortcuts
  - **S** – Start a new search
  - **T** – Toggle search
  - **N** – Switch to the next search
  - **P** – Switch to the previous search
  - **H** – Show the search history
  - **C** – Clear the search history

- Search query is also encoded in URL string

Dependency Tracking

- Show dependencies of each operation in the timeline

- Require both Legion Prof and Legion Spy outputs
  - Legion Spy might introduce some overhead

- Critical path analysis will be coming up shortly!

Mapper DSL

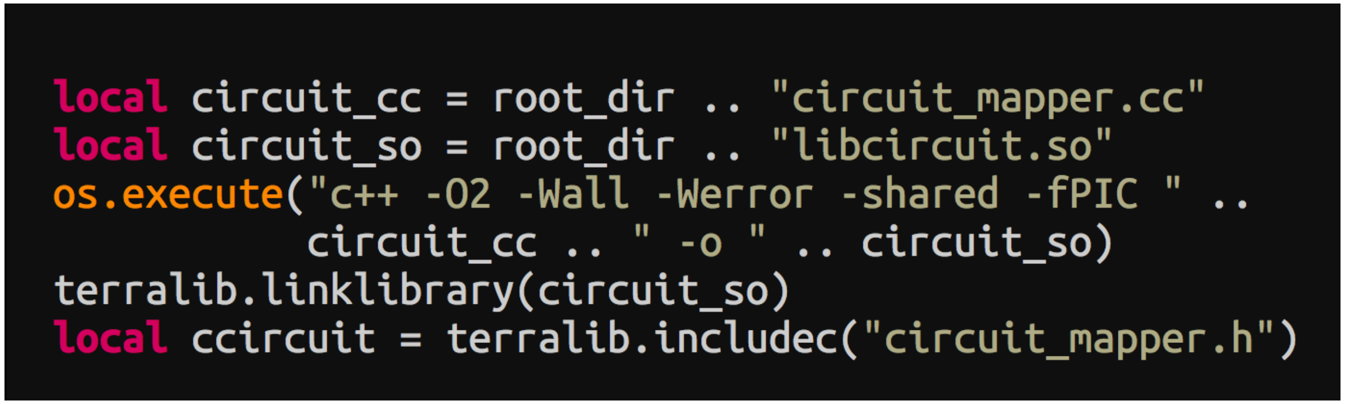
Wonchan Lee

Writing Mappers is Tedious

- Verbosity in the C++ API
- Differences between Regent and Legion
  - Region names vs. region requirements
  - Field names vs. field IDs
  - Compiler optimizations that generate non-user tasks

Writing Mappers is Tedious

- Verbosity in the C++ API
- Differences between Regent and Legion
  - Region names vs. region requirements
  - Field names vs. field IDs
  - Compiler optimizations that generate non-user tasks
- Mapper is not part of the Regent language



```
local circuit_cc = root_dir .. "circuit_mapper.cc"
local circuit_so = root_dir .. "libcircuit.so"
os.execute("c++ -O2 -Wall -Werror -shared -fPIC " ..
           circuit_cc .. " -o " .. circuit_so)
terralib.linklibrary(circuit_so)
local ccircuit = terralib.includec("circuit_mapper.h")
```

Bishop: A High-level Mapper DSL

- CSS-like syntax

```
<html>
  <body>
    <p id="block1">Bigger</p>

    <p id="block2">Smaller</p>
  </body>
</html>
```

HTML

```
p#block1 {
  font-size: 30pt;
}

p#block2 {
  font-size: 10pt;
}
```

CSS

Bishop: A High-level Mapper DSL

- CSS-like syntax

```
task child(r : region(...))
  ...
end

task parent(r : region(...))
  child(r)
end
```

Regent

```
task#parent {
  target-kind: x86;
}

task#child {
  target-kind: cuda;
}

task#child region#r {
  target-kind: fbmem;
}
```

Bishop

Bishop: A High-level Mapper DSL

- CSS-like syntax

```
task child(r : region(...))
  ...
end

task parent(r : region(...))
  child(r)
end
```

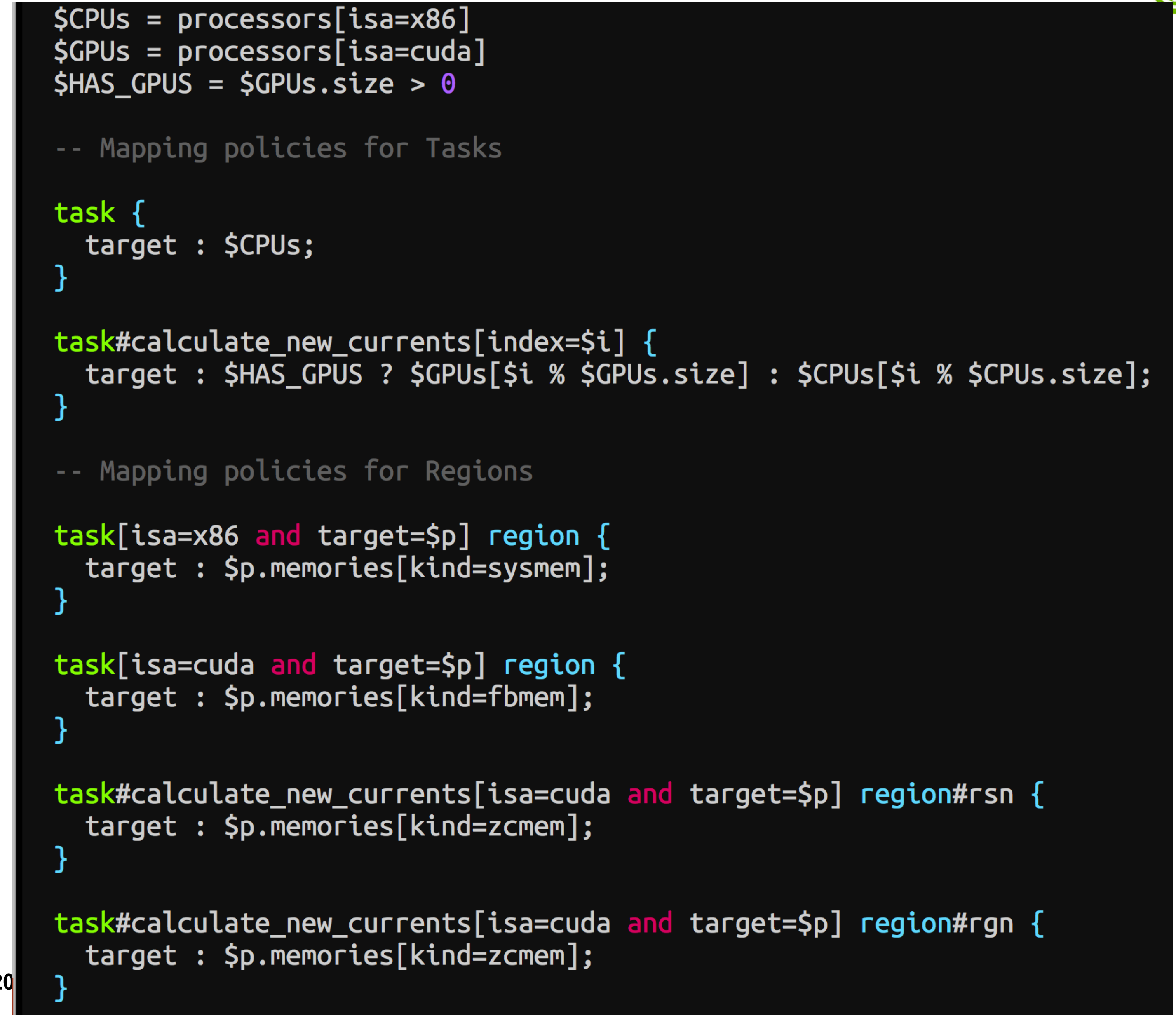
Regent

```
task#parent {
  target-kind: x86;
}

task#child {
  target-kind: cuda;
}

task#child region#r {
  target-kind: fbmem;
}
```

Bishop

- Keep the separation between description and execution

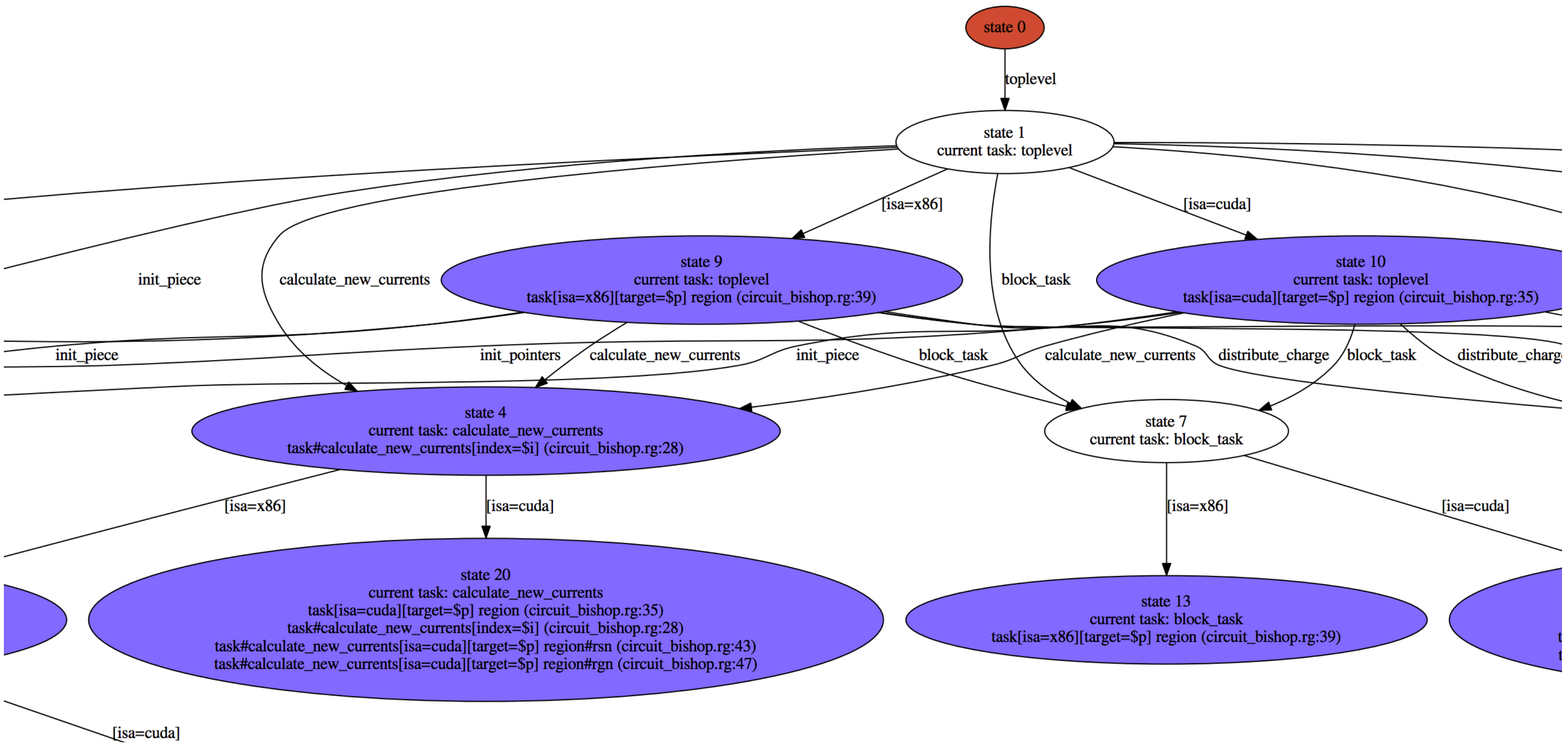
Circuit Example



```
$CPUs = processors[isa=x86]
$GPUs = processors[isa=cuda]
$HAS_GPUS = $GPUs.size > 0

-- Mapping policies for Tasks

task {
  target : $CPUs;
}

task#calculate_new_currents[index=$i] {
  target : $HAS_GPUS ? $GPUs[$i % $GPUs.size] : $CPUs[$i % $CPUs.size];
}

-- Mapping policies for Regions

task[isa=x86 and target=$p] region {
  target : $p.memories[kind=sysmem];
}

task[isa=cuda and target=$p] region {
  target : $p.memories[kind=fbmem];
}

task#calculate_new_currents[isa=cuda and target=$p] region#rsn {
  target : $p.memories[kind=zcmem];
}

task#calculate_new_currents[isa=cuda and target=$p] region#rgn {
  target : $p.memories[kind=zcmem];
}
```

Circuit Example

- Selectors are implemented as a distributed state machine



state 0
toplevel
state 1
current task: toplevel
[isa=x86]
[isa=cuda]
init_piece
calculate_new_currents
state 9
current task: toplevel
task[isa=x86][target=$p] region (circuit_bishop.rg:39)
block_task
state 10
current task: toplevel
task[isa=cuda][target=$p] region (circuit_bishop.rg:35)
init_piece
init_pointers
calculate_new_currents
init_piece
block_task
calculate_new_currents
distribute_charge
block_task
distribute_charg
state 4
current task: calculate_new_currents
task#calculate_new_currents[index=$i] (circuit_bishop.rg:28)
state 7
current task: block_task
[isa=x86]
[isa=cuda]
[isa=x86]
[isa=cuda]
state 20
current task: calculate_new_currents
task[isa=cuda][target=$p] region (circuit_bishop.rg:35)
task#calculate_new_currents[index=$i] (circuit_bishop.rg:28)
task#calculate_new_currents[isa=cuda][target=$p] region#rsn (circuit_bishop.rg:43)
task#calculate_new_currents[isa=cuda][target=$p] region#rgn (circuit_bishop.rg:47)
state 13
current task: block_task
task[isa=x86][target=$p] region (circuit_bishop.rg:39)
[isa=cuda]

Plan

- Make the language feature complete
    - Copy operations
    - Layout constraints for physical instances
    - Error handling
- Static analysis to generate mappers for Regent programs
- Optimize Regent task variants based on mapping policies