Circuit: A Regent Application

CS315B
Lecture 6

Circuit

- Electrical simulation
- A graph
  - Wires are edges
  - Nodes are places where wires meet

Circuit.rg

- Iterative simulation with three phases:
  - calculate_new_currents
  - distribute_charge
  - update_voltages
- New features
  - Structs
  - Permissions on multiple fields
  - wait_for(…)
  - __demand(…)

Circuit_dep_par.rg

- New features
  - Pointers to region unions
  - Reduce privilege
  - __demand(__parallel)
  - Tracing

Circuit Dependent Partitioning

```
var pn_equal = partition(equal, rn, colors)
var pw_outgoing = preimage(rw, pn_equal, rw.in_ptr)
var pw_incoming = preimage(rw, pn_equal, rw.out_ptr)
var pw_crossing_out = pw_outgoing - pw_incoming
var pw_crossing_in = pw_incoming - pw_outgoing
var pn_shared_in = image(rn, pw_crossing_in, rw.out_ptr)
var pn_shared_out = image(rn, pw_crossing_out, rw.in_ptr)
var pn_private = (pn_equal - pn_shared_in) - pn_shared_out
var pn_shared = pn_equal - pn_private
var pn_ghost = image(rn, pw_crossing_out, rw.out_ptr)
```

Mapping

Mapping

- Mapping is the process of assigning resources to Regent/Legion programs
- Conceptually
  - Assign a processor to each task
    - The task will execute in its entirety on that processor
  - Assign a memory to each region argument
- Can also control other things
  - But these are the most important

The Legion Mapping API

- A *mapper* implements the Legion mapping API
  - A set of C++ callbacks
- Bishop is a high-level language for mapping
  - Compiles to the mapping API
  - Not as general, but easier to learn and use

High-Level Overview

- An instance of the Legion runtime runs on every node
- When a task is launched the local runtime
  - Makes mapper calls to pick a processor for the task
  - Makes mapper calls to pick memories for the region arguments
  - … and other mapper calls as well …

New Concepts

- There are a number of concepts in mapping that don't exist in Regent
- Machine models
- Variants
- Physical Instances

Machine Model

- To pick concrete processors & memories, the runtime must know:
- How many processors/memories there are
  - And of what kinds
- And where the processors/memories are
  - At least relative to each other

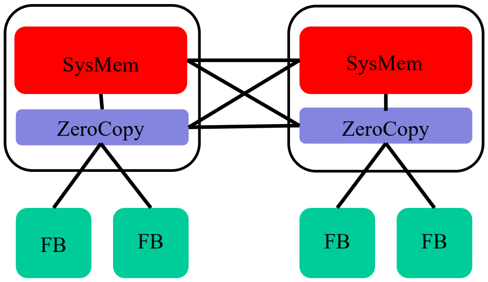
Machine Model

- Processors
  - LOC
  - TOC
  - PROC_SET
  - UTILITY
  - IO
- Memories
  - GLOBAL
  - SYSTEM
  - RDMA
  - FRAME_BUFFER
  - ZERO_COPY
  - DISK
  - HDF5

Affinities

- Processor -> Memory
  - Which memories are attached to a processor
- Memory -> Memory
  - Which memories have channels between them
- Memory -> Processor
  - All processors attached to a memory
- Affinities are provided as a list of *(proc,mem)* and *(mem,mem)* pairs

An Example Machine Model (Simplified)



Task Variants

- A task can have multiple *variants*
  - Different implementations of the same task
  - Multiple variants can be registered with the runtime
- Examples
  - A variant for LOC
  - Another variant for TOC
  - Variants for different data layouts

Physical Instances

- A *region* is a logical name for data
- A *physical instance* is a copy of that data
  - For some set of fields
- There can be 0, 1 or many physical instances of a specific field of a region at any time

Physical Instances

- Can be *valid* or *invalid*
  - Is the data current or not?
- Live in a specific memory
- Have a specific layout
  - Column major, row major, blocked, struct-of-arrays, array-of-structs, …
- Are allocated explicitly by the mapper
- Are deallocated by the runtime
  - Garbage collected

Index Launches

- A normal task call launches a single task
- An *index task call* launches a set of tasks
  - One for each point in a supplied index space
- Index launches are more efficient than launching many tasks individually
  - Regent automatically transforms loops of single task launches into index task launches

Example

```
for x in prt.colors do
   task(prt[x])
```

becomes

```
index_launch(task,prt,prt.colors)
```

(if there are no dependencies)

A Mapper

- The circuit custom mapper, circuit_bishop.rg

Miscellaneous Mapping Topics …

Controlling Processor Choice in Regent

- Place immediately before a task declaration
  - __demand(__cuda)
- Causes both CPU and GPU task variants to be produced
- And the default mapper always prefers to pick a GPU variant if possible

Layout Constraints

- Tasks can have layout constraints on physical instances
  - "This task requires data in row major order"
- Constraints are just that
  - Don't specify an exact layout
  - Multiple instances may satisfy the constraints

Reduction Instances

- A *reduction instance* is a special instance used for reductions
- Pattern

```
for i in R do
    i.field += val1
    i.field += val2
```

```
fill(R', 0)
for i in R.indices do
    R'[i] += val1
    R'[i] += val2

… later …

R += R'
```

Virtual Mappings

- It is also possible for a mapper to map a region to *no* instance
  - If the task does not use the region itself
  - E.g., only passes it to subtasks
- This is a *virtual mapping*

Summary

- Mapping
  - Selects processors for tasks
  - Selects memories for physical instances
    - Satisfying region requirements of tasks
- Choices
  - Use Bishop – easy to write a custom mapper
  - Default mapper does reasonable things
    - Overriding methods in the default mapper provides more options than Bishop mappers