Bulk-Synchronous Communication Mechanisms in Diderot

John Reppy, Lamont Samuels

University of Chicago

October 26th, 2015
Diderot

- Diderot is a parallel domain-specific language designed for biomedical image-analysis and visualization algorithms.

- Its design models the algorithmic structure of its application domain: independent strands computing over a continuous tensor field, which are reconstructed from discrete data using a separable convolution kernel $h$:

$$ F = V \ast h $$

---

Diderot overview

Diderot

- Diderot is a parallel domain-specific language designed for biomedical image-analysis and visualization algorithms.

- Its design models the algorithmic structure of its application domain: independent strands computing over a continuous tensor field, which are reconstructed from discrete data using a separable convolution kernel $h$:

\[ F = V \otimes h \]

---

Diderot Program Structure

Square roots of integers using Heron’s method.

```diderot
// global definitions
input int N = 1000;
input real eps = 0.000001;

// strand definition
strand SqRoot (real val)
{
    output real root = val;

    update {
        root = (root + val/root) / 2.0;
        if (|root^2 - val|/val < eps)
            stabilize;
    }
}

// initialization
initially [ SqRoot(real(i)) | i in 1..N ]
```
Diderot Parallelism Model

Bulk-synchronous parallel with deterministic semantics.
Diderot Parallelism Model (Cont’d)

- This model only supports applications that allow strands to act independently of each other, such as direct volume rendering or fiber tractography.

- Diderot is missing features needed for algorithms of interest, such as particle systems.
This model only supports applications that allow strands to act independently of each other, such as direct volume rendering or fiber tractography.

Diderot is missing features needed for algorithms of interest, such as particle systems.
Spatial Design

- Strands interact with each other based on their world coordinates.
- Strand uses special *query* functions to identify neighboring strands.
- Processing the queried sequence of strands is performed using a new mechanism called the *foreach* statement.
- Strand state is accessed using the *selection* operator.
Spatial Design

- Strands interact with each other based on their world coordinates.
- Strand uses special `query` functions to identify neighboring strands.
- Processing the queried sequence of strands is performed using a new mechanism called the `foreach` statement.
- Strand state is accessed using the `selection` operator.
Spatial Design

- Strands interact with each other based on their world coordinates.
- Strand uses special query functions to identify neighboring strands.
  - Processing the queried sequence of strands is performed using a new mechanism called the `foreach` statement.
- Strand state is accessed using the `selection` operator.
Spatial Design

- Strands interact with each other based on their world coordinates.
- Strand uses special **query** functions to identify neighboring strands.
- Processing the queried sequence of strands is performed using a new mechanism called the **foreach** statement.
- Strand state is accessed using the **selection** operator.

```c
strand Particle (vec2 initPos) {
    vec2 pos = initPos;
    update {
        foreach (Particle neighbor in sphere(5.0))
        {
            ...
        }
        ...
    }
}
```
Spatial Design

- Strands interact with each other based on their world coordinates.
- Strand uses special query functions to identify neighboring strands.
- Processing the queried sequence of strands is performed using a new mechanism called the foreach statement.
- Strand state is accessed using the selection operator.
Global Design

- Strands can share state information on a larger scale within the system.
- Global communication is performed by using common reduction operations.
- The reduction operations reside in a new definition block called `global`.
- A reduction is performed across a set of strands.
Global Design

- Strands can share state information on a larger scale within the system.
- Global communication is performed by using common reduction operations.
  - The reduction operations reside in a new definition block called `global`.
  - A reduction is performed across a set of strands.

Reductions: mean, max, min, all, exists, variance, sum, product
Global Design

- Strands can share state information on a larger scale within the system.
- Global communication is performed by using common reduction operations.
- The reduction operations reside in a new definition block called `global`.
- A reduction is performed across a set of strands.

Reductions: `mean`, `max`, `min`, `all`, `exists`, `variance`, `sum`, `product`
Global Design

- Strands can share state information on a larger scale within the system.
- Global communication is performed by using common reduction operations.
- The reduction operations reside in a new definition block called `global`.
- A reduction is performed across a set of strands.

Reductions: `mean`, `max`, `min`, `all`, `exists`, `variance`, `sum`, `product`
Spatial Implementation

- Spatial communication is implemented using a tree-based spatial partitioning scheme (k-d tree).
- The tree is built using a strand’s pos variable and is rebuilt before each iteration.
Global Implementation

The global computation phase groups reductions into execution phases. This process is done in two steps: lifting and phase insertion.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
</tr>
</tbody>
</table>
Strand Allocation

- Strands are dynamically allocated by the `new` statement.
- New strands will begin executing in the next iteration.
Strand Allocation

- Strands are dynamically allocated by the **new** statement.
- New strands will begin executing in the next iteration.
Live Demo (Particle-Based Isocontour Sampler)

Sampling Implicit Surfaces

Sampling Valley Lines
Conclusion

Summary

- Provided new mechanisms to our programming model.
  - Allows for more algorithms to be implemented within Diderot.
- Future Work
  - Additional query functions
  - Implementation of communication mechanisms on GPUs
Summary

▶ Provided new mechanisms to our programming model.
  ▶ Allows for more algorithms to be implemented within Diderot.

Future Work

▶ Additional query functions
▶ Implementation of communication mechanisms on GPUs
Summary

- Provided new mechanisms to our programming model.
  - Allows for more algorithms to be implemented within Diderot.

Future Work

- Additional query functions
- Implementation of communication mechanisms on GPUs
Summary

- Provided new mechanisms to our programming model.
  - Allows for more algorithms to be implemented within Diderot.
- Future Work
  - Additional query functions
  - Implementation of communication mechanisms on GPUs
Acknowledgements

The Diderot Project

Charisee Chiw  University of Chicago
Gordon Kindlmann  University of Chicago
John Reppy  University of Chicago
Nick Seltzer  University of Chicago
Questions?

http://diderot-language.cs.uchicago.edu