Brook: A Parallel Streaming Language

Ian Buck

Computer Systems Laboratory
Stanford University
Brook Review

- C with streaming
- Streams
  - Contiguous 1D view of records in memory.
  - Operated on in parallel.
Brook Review

- **Kernels**
  - Functions which operate only on streams
    - Arguments are read-only, write-only, or reduce (associative operations only)
  - Called on every element of the input streams
  - Limited communication between elements
    - No “static” variables
    - No global memory access
UPC Data Partitioning

• Motivation:
  – Data partitioning and load balancing a hard problem!
  – Add primitives allow compiler hints, not rules.
  – Don’t reinvent

• UPC
  – Unified Parallel C
  – Industry, government, academic collaboration project
  – Formalized specification and open source compilers available
UPC Additions

• Data Partitioning
  – Partitioned across machine as defined in UPC spec
    shared [3] int A[THREADS][5];

  – Dynamic allocation
    shared void *upc_all_alloc (size_t nblocks, size_t nbytes)
    Same as:
    shared [nbytes] char[nblocks*nbytes]
UPC Additions

• Data partitioning dictates parallel kernel execution
  – Distributed data structures drives parallelism

```
shared [4] float a[12];
foo(a, a);
```
UPC Additions

• Same scalar code executed on all nodes
  – No mechanism for task parallelism (no UPC forall constructs, no MYTHREAD)
  – Each instance keeps a copy all non-shared memory (locals, heap memory, etc).
  – Read/Write to non-shared memory is resolved locally
  – Reads to shared memory is performed by all nodes
  – Writes to shared memory only occurs if memory resides on that node.

• Synchronization on Writes
  – Determined by UPC reference-type-qualifier.
    strict shared int y;
    • Strict ordering semantics (code inserted by compiler/runtime)
    • Worst case: sync on every write
    relaxed shared int x;
    • Unordered semantics
  – User inserts barriers/sync commands
    • upc_barrier(); (no upc_locks, upc_wait, upc_notify)
  – #pragma upc strict/relaxed determines default characteristics
Memory Consistency

• Motivation
  – Stream code and traditional C (“scalar”) code.
  – Permit parallel execution of kernels and scalar.
  – Present well defined memory model

• Formalization
  – Two kinds of streams
    • memstream: Streams backed by memory
    • stream: Temporal streams for connecting kernels (non-backed)
Memory Consistency

• Memstreams
  – Similar to C-arrays.
  – Arbitrary access within scalar code *NEW!*
    ```c
    memstream float a[1024];
    a[435] = 3.2f; // LEGAL!
    foo(a, a); // kernel call
    ```
  – No pointer aliasing
  – Writes and reads to a memstream are ordered
  – The name “a” always refers to memory “a” declared
MemStreams

- No address calculations or pointer arithmetic

```c
memstream float as [13][32];
float b = as[3][5];  \ Legal
as[2][5] = 4.3f;     \ Legal
float *p = &(as[3][5]);  \ Illegal: address computed
b = **(as+5);        \ Illegal: pointer arithmetic
```
MemStreams

- Declarations
  
  `memstream float val;`
  Stream of floats, one element

  `memstream float a[4][5][9];`
  Stream of floats, 4*5*9 elements.

  `typedef float mytype[3][2];`
  `memstream mytype c[304];`
  Stream of float[3][2], 304 elements

- A few more complex examples
  
  `memstream char *daytab [13];`
  Stream of char *, 13 elements

  `memstream char (*daytab)[13];`
  Stream of pointer to char[13], one element

  `memstream char (stringtable[3])[5];`
  Stream of char[5], 3 elements

See K&R sec 5.2
Memstream

• 3 kinds of stream shape
  – Constant: Fixed compiled time shape
    ```c
    memstream float as[30][20][5];
    ```
  – Static: Fixed runtime shape
    ```c
    memstream float as[n][m];
    ```
  – Dynamic: Variable shape
    ```c
    memstream float as[][][] = 
    alloc_memstream(sizeof(float), 8, 10);
    reshape(as, 4, 20);
    dim(as, 0); // value = 4
    ```

• Shape is part of type
• No changing of number of dimensions
Memstream

• Passing memstreams to functions
  ```
  memstream float as[3][2];
  memstream float bs[n][m];
  memstream float cs[][] = alloc_memstream(sizeof(float), 8, 5);
  
  void function (memstream float arg1[3][2],
                  memstream float arg2[n][m],
                  memstream float arg3[void][void],
                  memstream float arg4[][]);
  ```

• Shape promotion rules
  Constant -> Constant, Static, Dynamic
  Static    -> Static, Dynamic
  Dynamic   -> Static, Dynamic

• Shape passed by value

• Issue: cannot return created memstream
Streams

• Similar to memstreams
  – No array indexing
  – No associated memory

• Declarations
  
  `stream float s[][]; // 2D stream of floats`

• Shapes for streams
  – Similar to dynamic memstreams
  – Permit reshaping?
  – Shape inherited from first input argument of kernel (ugh)
Memory Consistency

memstream float a[1024], c[1024];
foo(a, c); // kernel
a[435] = 3.2f; // cannot affect foo, c
float x = c[385]; // sees foo

• Memstreams
  – All writes and reads to memstreams are fully ordered.
  – Behaves sequentially
Memory Consistency

```c
memstream float a[1024], c[1024];
stream float b[];
foo(a, b);       // kernel
a[435] = 3.2f;   // MAY affect b
bar(b, c);       // kernel
```

- **Streams**
  - Writes to b by kernels may be affected by writes to dependent memstreams
Memory Consistency

```c
memstream float a[1024], c[1024], d[1024];
stream float b[];
foo(a, b, c);    // kernel out:b, c
a[435] = 3.2f;  // MAY affect b
float x = c[342]; // Sees foo
bar(b, d);      // kernel
```

- **Streams**
  - Writes to b by kernels may be affected by writes to dependent memstreams
  - May recompute kernels
Stream Operators

- Deleted:
  - StreamLoad, StreamStore
  - FileStream
  - SelfProduct, Product
  - StreamShape
  - refstream
Stream Operators

- Same
  - StreamGroup
  - StreamStencil
  - StreamStride
  - StreamReplicate
  - StreamDomain
  - StreamCat
  - StreamMerge
Stream Operators

- Changed

StreamScatterOp(memstream base, memstream/stream offsets, memstream/stream values, OP);
StreamGatherOp(memstream/stream results, memstream base, memstream/stream offsets, OP);
Stream Operators

• Limbo
  – StreamGetLength/StreamSetLength
  – StreamRepeat
  – StreamZero
  – StreamFlatten
Stream Operators

- New (names not final)
  - alloc_memstream
  - dim
  - reshape
  - C ptr to memstream cast