



SSS Software Update

Ian Buck

Mattan Erez

August 2002



Outline

- **Applications**
 - StreamMD
 - StreamFlo
- **Compilation**
- **Compiler Status**
- **Multinode notes**
- **Results (what we want)**



StreamMD Overview

- **Water simulation**
- **Gridded for accelerated force calculation**
 - Only adjacent grid cells interact
- **Complex force calculation**
 - Tens of instructions per word transferred
- **Status**
 - Reference C++ version working
 - Mostly complete Brook version



StreamMD Details

- **Grid structure**
 - The grid cells are a stream
 - Each stream record holds a number of molecules
 - A separate stream holds the number of molecules in each cell
- **Gridding operation**
 - For every molcl – compute the appropriate cell
 - Use a **GatherAdd** operation to assign locations
 - Each molcl cell's num_molcls is atomically incremented
 - The resulting indices are returned in a stream
 - The return stream is used to **scatter** the molecules
- **Force calculation**
 - Redundant forces are calculated assuming max number of molecules per cell



StreamFlo

- **2D multi-grid solver**
- **Data structure**
 - Data is stored in a 1D stream which holds all the levels
 - Each level is selected using **domain** and then **shaped** to a 2D stream
- **Operation**
 - In each level a **stencil** is used to calculate the flux
 - A **group/stencil** is used to transfer the field up/down the hierarchy



Compiling StreamMD

- **Gridding operation supported by hardware stream Fetch&Op**
- **Force calculation**



Compiling StreamFlo

- **Simply need to generate code for the different stream operators used**
 - StreamShape
 - StreamStencil
 - StreamDomain/StreamMerge
 - StreamSetLength
- **The general approach is to delay the operation until required by a kernel**
 - Register the operator parameters in a compile time and/or runtime structure and wait



StreamShape

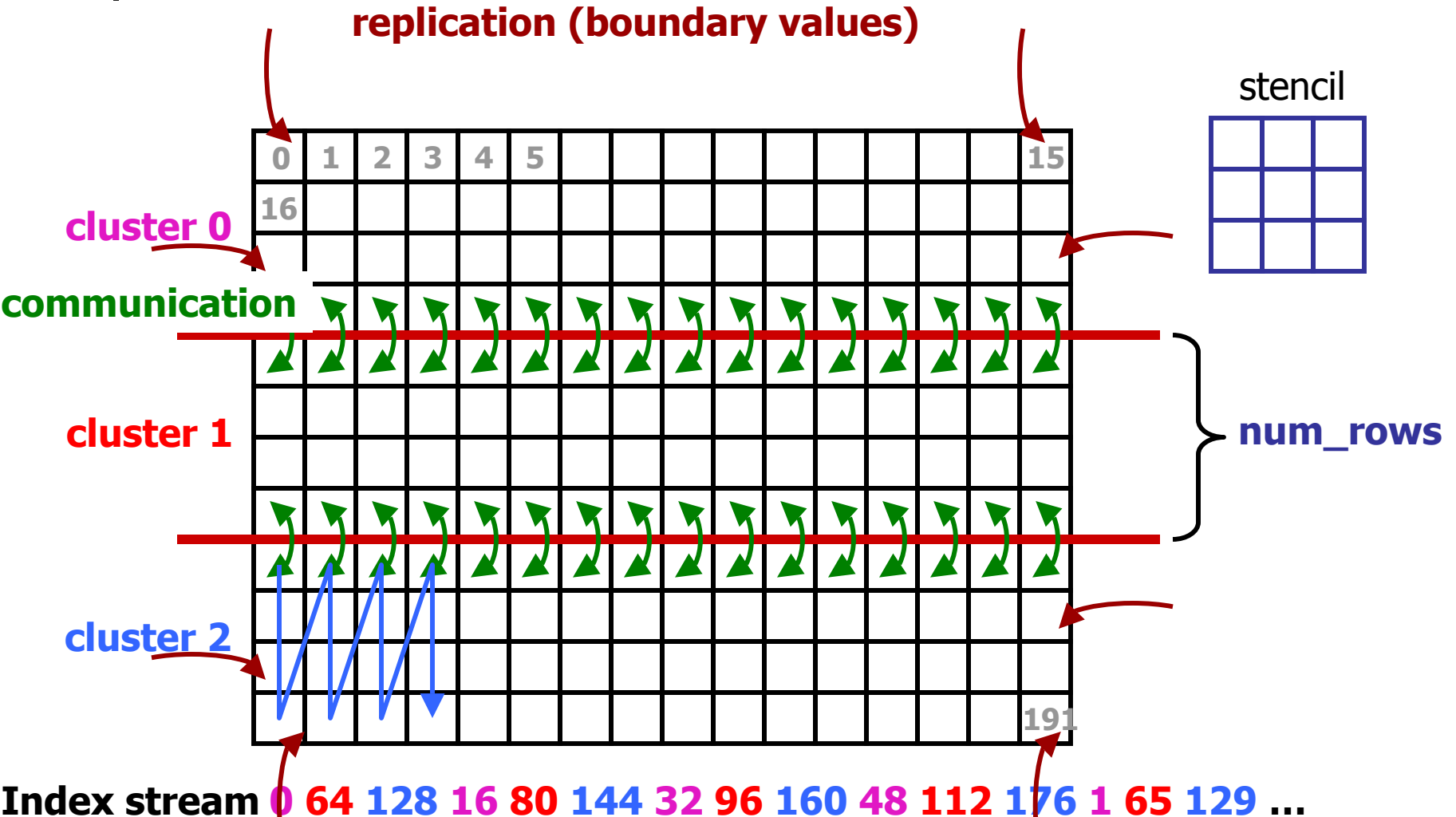
- **Register shape in runtime table**
 - Dimension (currently 2D only)
 - 1D streams are not tracked
 - Size
 - $M \times N$



StreamStencil (1)

- **Map the 2D stencil into 1D streams**
 - Suitable for both StreamC/KernelC and the SVM
- **Minimize cost**
 - Replication (re-reading stream elements)
 - Communication
 - Utilize LRF and Scratchpad
- **Current implementation relies on specific order of execution in the stream unit**

StreamStencil (2)





StreamStencil (3)

- **Communication to computation ratio proportional to**
$$\frac{\text{row length}}{\text{num rows}}$$
- **Replication only for boundary values**
- **Number of rows determined by amount of local storage**
 - Scratchpad
 - LRF



StreamGroup

- **Handled exactly like streamStencil**
- **No need for communication**
- **No need for boundary conditions**
- **num_rows == group_length (since no comm)**



StreamDomain

- **Domain use:**

- **Another domain** – propagate boundaries (do nothing)
- **Kernel input** – generate an index stream for the domain and gather the domain (unless 1D)
- **Kernel output** – wait for **StreamMerge** and then use index stream to copy result back
- **Stencil input** – generate the stencil index stream starting at the appropriate offset
- **Problems** – can only work if the domain portion of the source stream is not modified



Compiler Status

- **Most features implemented**
 - Parts of streamStencil and SetLength are currently being coded
- **Annoyances**
 - Can't handle multiple .br files
 - No automatic type casting in kernels
 - $1 \neq 1. \neq 1.0 == 1f$
 - -1 doesn't work (use 0 - 1)
 - ...
 - No global variables allowed in kernels
 - Shaky error reporting
 - Some editing of .h files required



Compiler Status (2)

■ Deficiencies

- Can't handle many stream functions – if the function changes the properties of the stream (length, shape, stencil, domain, ...)
- Can't handle arrays as record fields or kernel params
 - Vec3 must be .x, .y , and .z instead of [0], [1], and [2]
- No optimizations
 - We did our best implementing efficient strip-mining and stream operators
 - Strip parameters must be inserted manually
 - No merging/splitting of kernels
- No multi-node support
- Compiles to StreamC/KernelC and not SVM



Multi-Node Notes

- **Simple partitioning**
 - Both apps are basically regular meshes
- **Simple synchronization**
 - Very obvious synch points between force/flux calculations and integration/multi-grid iterations
 - StreamMD requires refstream and user synchronizations
- **Will use C library mutex type synchronization**



Results (what we need)

- **Single node**

- Run-time
- LRF/SRF/MEM bandwidth
- Cluster utilization
- Stall time (due to memory/host)

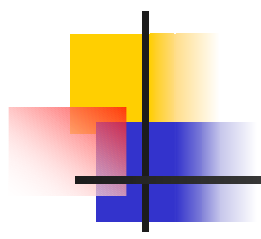
- **Multi node**

- All of the above
- Network BW
- Synch stalls

- **Comparison**

- Need to identify, define, and collect base line numbers for the 100:1 comparison







StreamSetLength

- **New length < old length**
 - Rename all future references to new_stream
 - Derive new_stream from old_stream
- **New length > old length**
 - Rename all future references to new_stream
 - Create new_stream with the new length
 - Copy old data to the beginning of the new stream