

---

# **Merrimac Applications**

---

Kayvon Fatahalian  
January 13, 2004

# Purpose

---

- Provide a high-level overview of how each of the major Merrimac applications has been implemented in Brook (or will be).
- Identify common computational tasks among the apps, and compare existing solutions.
- Identify any existing problems in the Brook language/compiler that are hindering current application development.

# Applications

---

**StreamFLO**

**StreamFEM**

**StreamMD**

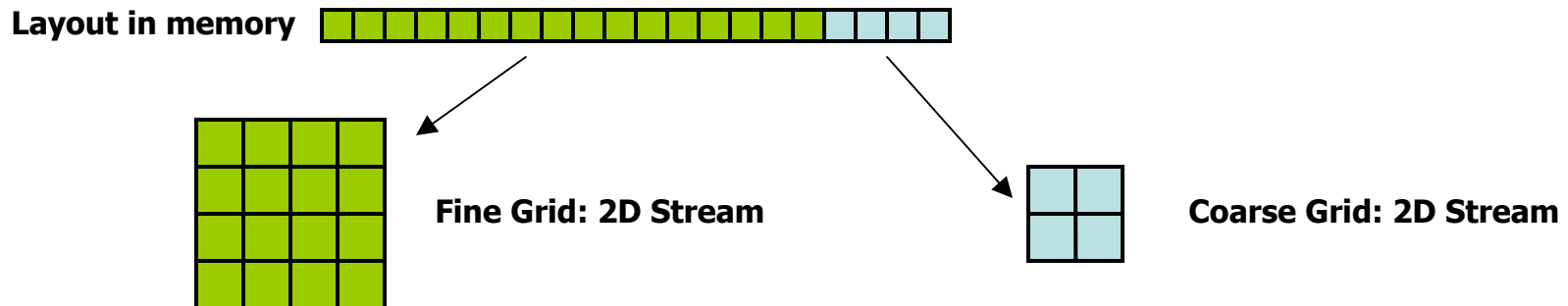
**StreamSPAS**

**Unstructured Multigrid**

# StreamFLO

---

- **Description:** finite volume 2D Euler solver. Utilizes non-linear multigrid algorithm on *regular* grids.
- **Data Structure:** 3 structs store information for each grid cell.
  - **Flow** { density, pressure, momentum }
  - **Cell:** { (x,y) position in space, *logical position in grid (i, j)* }
  - **Flux:** { density, momentum } (inefficiency here, flux computed twice)
- All grids stored as a single large consecutive 1D array in memory. Data loaded into a distinct 2D stream for each grid resolution.



# StreamFLO

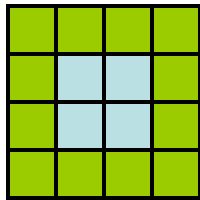
---

- **Stream Operations:**
  - **StreamStencil:** Max allowable time step and flux computations performed by kernels that operate on stencils. Mostly uses 3x3 and 5x5 stencils.
    - **Caveat:** Requires different boundary conditions on edges of grid. Periodic boundary conditions in the direction of the flow, fixed boundary along wall.

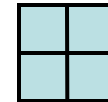
# StreamFLO

---

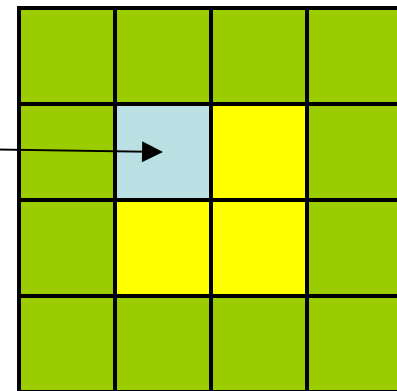
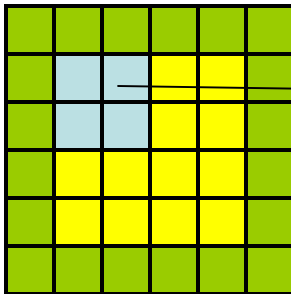
- **The most interesting use of Brook stream operators is in the transfer of flow between different grids:**
  - **StreamDomain/StreamMerge:** Separate interior grid cells from boundary cells for different processing (after computation, stick them back together).



Create stream of interior nodes with StreamDomain



- **StreamGroup:** Flow from a 2x2 group of fine grid cells (interior cells) is summed to produce flow in a single coarse grid cell. Interpolation is performed on 2x2 groups of coarse grid cells to transfer flow back up to the fine grid.



# StreamFLO

---

- **Difficulties:**

- Boundary conditions. Must operate differently on interior cells, near and far boundary cells.
  - **Solution:** Store logical position  $(i,j)$  of each cell in the cell struct.
  - Use conditionals in kernels depending upon value of  $cell.i$  and  $cell.j$
  - Rely on compiler to optimize out loads of these values in kernels that do NOT require this information.
- Brook unable to support streams of varying length. Essential for multigrid since algorithm required grids of varying size.

# StreamFLO

---

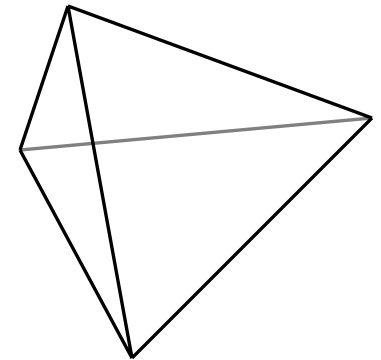
- **State of StreamFLO:**

- Multigrid Brook implementation exists (not Brook 0.2), but performance only measured using computation on **single** fine grid.
  - Hold up: lack of variable length streams.
  - Decreasing parallelism on coarse grids, interesting test of Merrimac performance.
- Brook 0.2 port does not yet exist. Most likely still only a single grid version.
- StreamFLO 3D is in progress (in BrookTran?)

# StreamFEM

---

- **Description:** Discontinuous Galerkin finite element solver for unstructured 2D/3D meshes.
- **Data Structures:** 2 main types of records
  - **Cell** { **index of 4 adjacent faces**, quadrature pts, simulation data...}
    - A cell is a tetrahedra in 3D, triangle in 2D
  - **Face** { **index of 2 adjacent cells**, boundary info, simulation data...}
  - Stored in memory:
    - Array of cell structs (size = number of cells)
    - Array of face structs (size = number of faces)
    - Array of flux terms (size = number of faces)

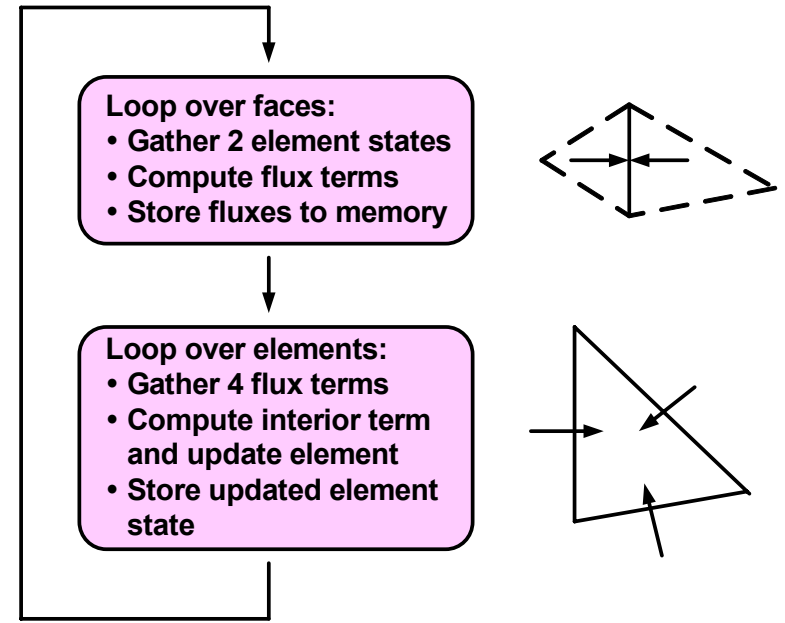


# StreamFEM

- **StreamOperators:**

- StreamGather:
  - During flux computation, use a gather to create 2 streams containing cells on each side of every face.
  - For each cell, gather flux terms for each face to update element
- StreamDomain/StreamMerge: Split into streams of interior and boundary nodes and send to different kernels.

For each timestep:



- Question? Could separation of faces and cells be used to avoid repeated flux computation in StreamFLO?

# StreamFEM

---

- **App Characteristics and Performance**

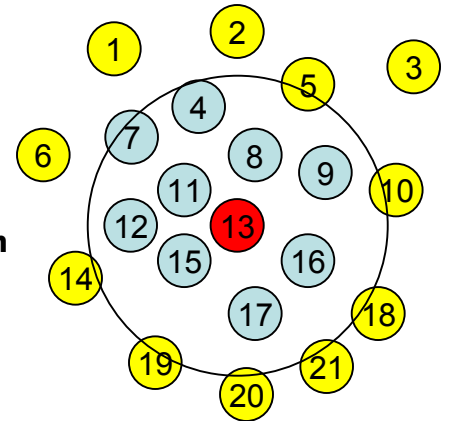
- Unstructured mesh results in irregular memory access pattern
  - But 2x reuse of flux data (cached).
  - Future work: look into optimizing traversal for improved cache coherency.
- Performance of **entire app** measured at ~50% of peak.
- Computation of flux terms is bandwidth limited.
- In current version, unnecessary write from stream back to memory between kernels due to inability to declare functions with stream arguments in C headers.

# StreamMD

---

- **Description:** Molecular dynamics simulation of water molecules.
- Ported “non-bonded” force computation (electrostatic and Lennard-Jones forces) to streaming model. Most costly part of computation.
- Non-bonded forces are only significant between nearby atoms. Apply distance cutoff to determine which atoms interact. For each atom, a list of “neighboring” atoms is computed **on the scalar processor** ever couple of time steps.

Blue-colored atoms will appear in atom 13's neighbor list.

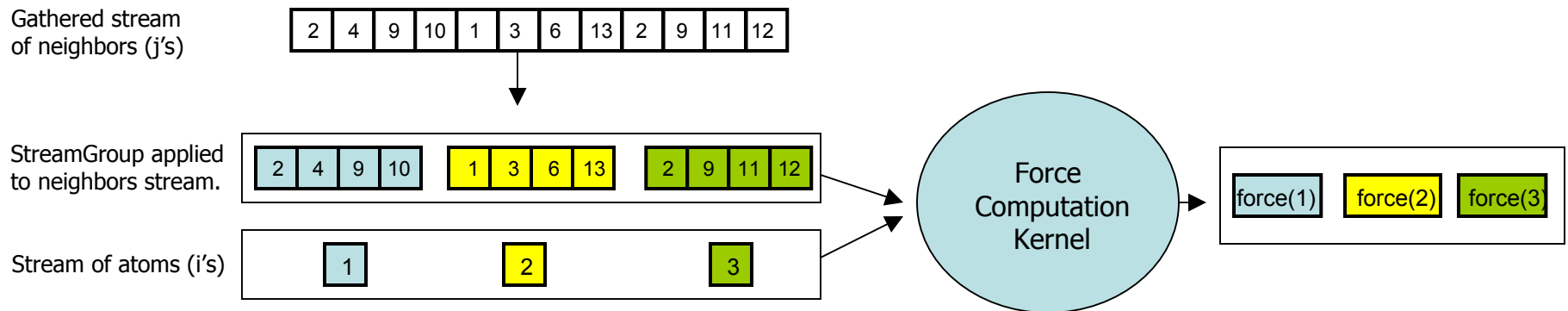


- **Primary Data Structure:**
  - **Atom** { position, velocity, force, **several fixed-size “neighbor” lists** }

# StreamMD

- **Force computation algorithm (simple version):**

- Gather and group a stream of atoms, given by indices in atom  $i$ 's neighbor list.
- Compute non-bonded forces between each atom  $i$  and all atoms  $j$  in the neighbor list.
- Output total force on each particle.



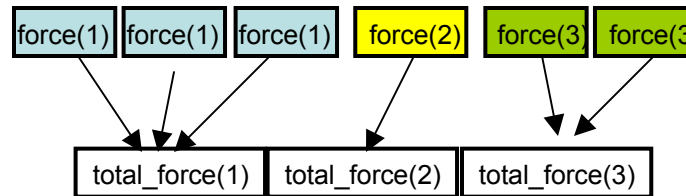
# StreamMD

---

- **Performance Optimizations:**

- Fixed size neighbor lists for performance, but number of actual neighbors varies throughout simulation.
  - If too few actual neighbors, pad with index of a fictitious infinitely distant atom.
  - If too many neighbors, **duplicate atom** in the simulation (this is done when building neighbor list)!
  - Now force computation computes a partial force for each atom. ScatterAdd resulting forces back to memory to get total force.

Partial forces computed by force kernel.



- Different types of force computations between different atoms.
  - Solution: Avoid conditionals in kernels, use different kernels and maintain different neighbor lists for each interaction type.
- Periodic boundary conditions in the simulation. Distant atoms may interact across boundary (screws up distance computation).
  - Solution: Use different kernels & neighbor lists for interactions over boundaries.

# StreamMD

---

- **Important Stream Operators**
  - StreamGather
  - StreamGroup
  - ScatterOp
- **Twice the Necessary Computation is Performed:**
  - Each force computation must be done twice.  $(i,j)$  and  $(j,i)$
  - But writing force to both  $i$  and  $j$  requires scatterAdd to  $j$ 's force accumulator.
  - Neighbor list approach saves bandwidth at cost of extra computation.
- **Interesting multinode issues**
  - Atoms migrate over time, interaction lists unknown until runtime.
- **StreamMD Performance:**
  - Simulated at 21 GFLOPS on Merrimac
  - Ran at 20% of peak on Imagine

# StreamSPAS

---

- **Description:** Benchmarking performance of sparse linear algebra operations on Merrimac.
- Sparse Matrix, Dense Vector Multiplication
  - Various representations of sparse matrices:
    - Compressed Sparse Row
      - Variable length rows, gather performed on X vector
    - Compressed Sparse Column
      - Variable length columns, scatterAdd to output (linear combination of columns)
    - Hypergraph Edge Storage
      - Requires matrices with symmetric nonzero pattern, utilizes gathers and scatterAdds
    - Element-by-Element Storage
      - store matrix in small dense blocks (corresponding to FEM elements), requires gathers and scatterAdds.
      - Matrix never has to be assembled completely in memory.

# StreamSPAS

---

- **Example: Padding CSR representation:**

- Variable number of nonzero elements per row is difficult to stream.
- Fix maximal number of elements, and pad small rows.

$$\begin{bmatrix} A & 0 & 0 & 0 \\ 0 & 0 & C & D \\ 0 & E & 0 & 0 \\ F & 0 & 0 & 0 \end{bmatrix}$$

CSR Representation:

Anz: A C D E F

AIdx: 0 2 3 1 0

Rows: 0 1 3 4 5

Padded Representation:

Anz: A 0, C D, E 0, F 0

AIdx: 0 0, 2 3, 1 0, 0 0

- Similar to neighbor lists in StreamMD
  - Could reduce extra padding by “duplicating rows” and scatterAdd partial row sums
  - Could order rows by number of NZ elements, apply different kernels for different sizes

# StreamSPAS

---

- **Low arithmetic intensity:**
  - Minimum one memory fetch for every 2 float ops.
  - Simulated performance of Kernel/StreamC implementation
    - Achieved 4 GLFOPS (EBES on large matrices)
  - Current Work: Reduce bandwidth requirements of computation.
    - For EBES matrices, compute **element matrix on the fly** from FEM element state.

# StreamSPAS

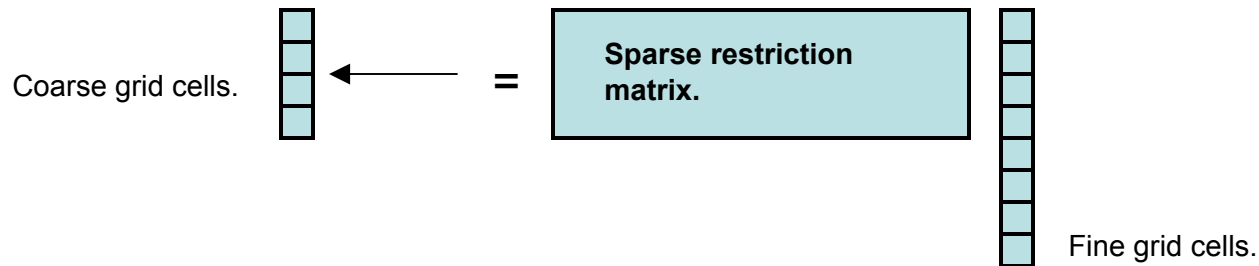
---

- **StreamSPAS goal:** Iterative solution to Poisson equation for arbitrary order finite elements on 3D meshes.
- Efficient Iterative Solvers require effective preconditioning of matrix.
  - Common preconditioners use recursive algorithms. (cannot stream)
  - Current work: Attempting to develop Sparse Approximate Inverse (SPAI) preconditioner and EBES preconditioner in Brook.
    - Requires significant sparse matrix data-structure manipulation
    - Creates streams of large record sizes ( $\sim 1000$  floats)

# Unstructured Multigrid

---

- **Description:** Unstructured geometric multigrid is a component of CDP (combustion simulation on unstructured meshes)
- Can no longer use stencil operations as in StreamFLO.
- Must perform gathers using index lists of neighboring cells.
- Flow transfer between grids can be expressed as sparse matrix-vector product (non-square matrix).



- Smoothing expressible as a sparse matrix-vector product (square matrix)
- Performance Concerns:
  - Decreasing parallelism at coarser grid levels.

# Applications Summary

---

- Merrimac applications can be characterized by the graph of their principal data structure (ordered by increasing complexity):

	<b>Graph</b>	<b>Num Edges</b>	<b>Num Nodes</b>
StreamFLO	Structured	Constant	Constant
StreamFEM StreamLES	Unstructured	Constant	Constant
StreamSPAS	Unstructured	Const/Variable	Constant
StreamMD	Unstructured	Variable	Constant
??	Unstructured	Const/Variable	<b>Variable</b>

- StreamMD handles variable number of graph edges by periodically adjusting the amount of per-atom storage for neighbor lists. (duplication of atoms with long interaction lists done following neighbor list update).
- We currently have no applications that handle a variable number of graph nodes (adaptive refinement of elements, grid cells, ect.)

# Usage of Brook Operators

---

	StreamFLO	StreamFEM	StreamSPAS	StreamMD
Multi-D streams (StreamShape)	X			
StreamDomain/Merge/Cat	X	X		
StreamStencil	X			
StreamGroup	X			X
ScatterOp			X (ScatterAdd)	X (ScatterAdd)
Gather		X	X	X
GatherOp				
Variable Output Kernels				
Reduction Kernels	X	X	X (inner product)	X

- Can any current applications benefit from the lesser used Brook features?
- GatherOp can be used to build more complex data structures. (mimic pointers)

## In General...

---

- Everyone is waiting for variable length streams.
- Most applications would benefit from Brook support for a variable length group operation. (stream of variable-sized lists)
  - Is this streams of streams?