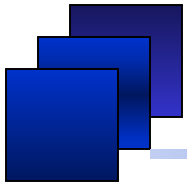


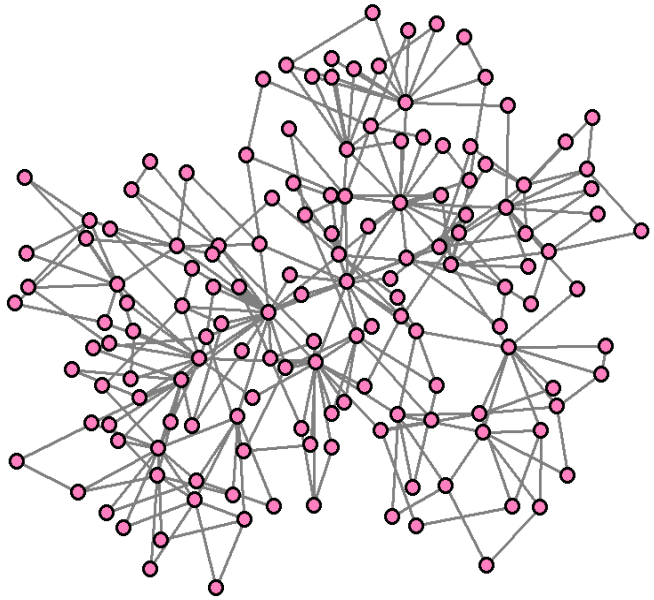
Improving Unstructured Mesh Application Performance on Stream Architectures

Nuwan Jayasena, Francois Labonte,

Yangjin Oh, Hsiao-Heng Lee,

Anand Ramalingam

Unstructured Mesh Applications



- Nodes have some values that are updated based on neighbors' values
- Nodes have different number of neighbors
- Accesses are not regular
- Lots of data reuse but irregular

1. Current Imagine

- Each node has different number of neighbors on which its update depends
- How to pack SRF?

Maximum Neighbors

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

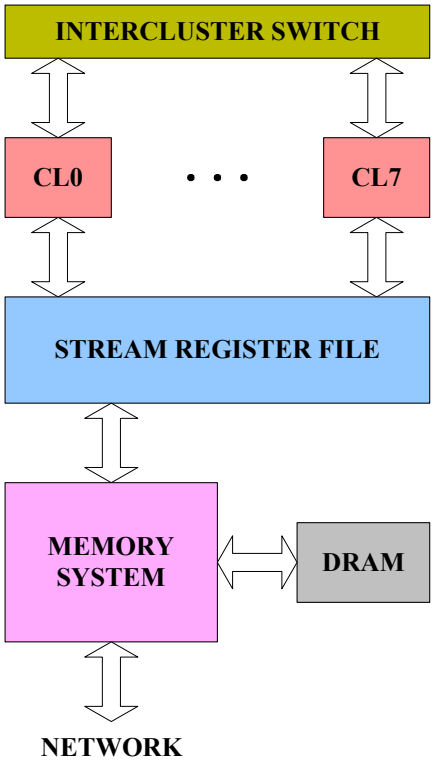
0	1	2	3	4	5	6	7
	1	2	3	4	5	6	7
		2	3	4		6	7
			3			6	
			3			6	
			3				
0	1	2	3	4	5	6	7
	1	2	3	4	5	6	7

Conditional input stream

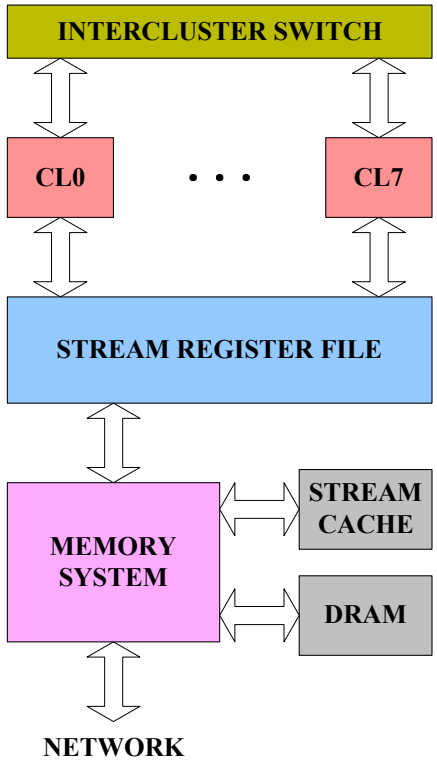
0	1	2	3	4	5	6	7
0	0	1	5	…	...	...	...

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
			3			6	
			3			6	
			3				

2. Imagine with Cache

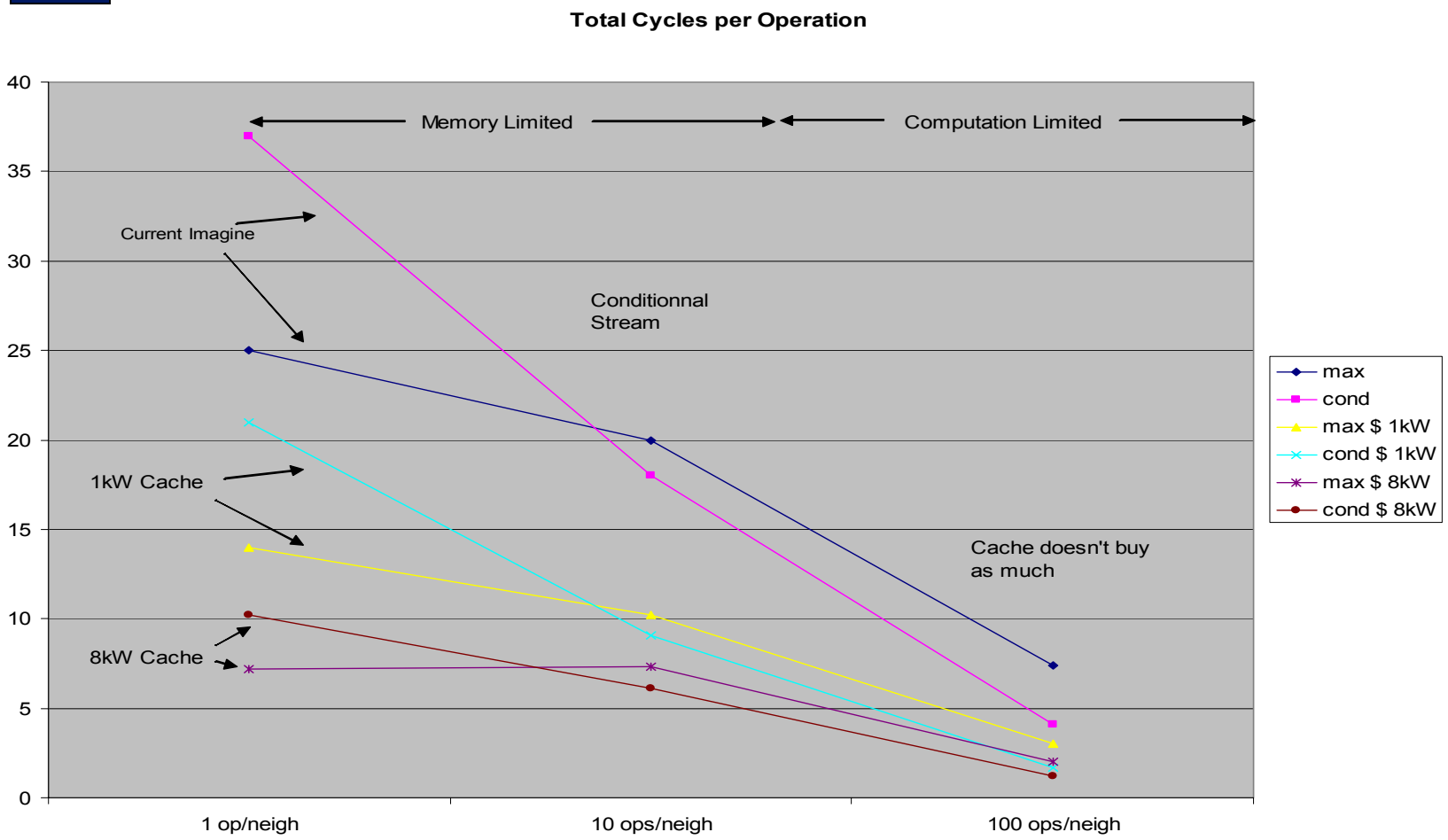


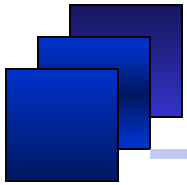
ORIGINAL IMAGINE SYSTEM



WITH STREAM CACHE

Results Imagine and Cache





SRF Indexing Tradeoffs

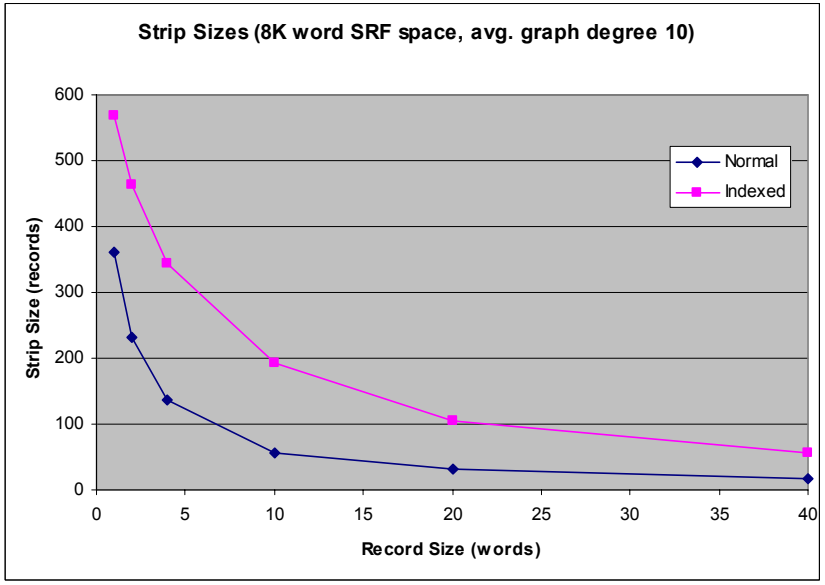
+ Reduce data replication in SRF → larger strip sizes
+ Reduce bandwidth demands on memory system
+ Data-dependant accesses within kernel (e.g. hashing)
+ Improve performance for memory-bound applications

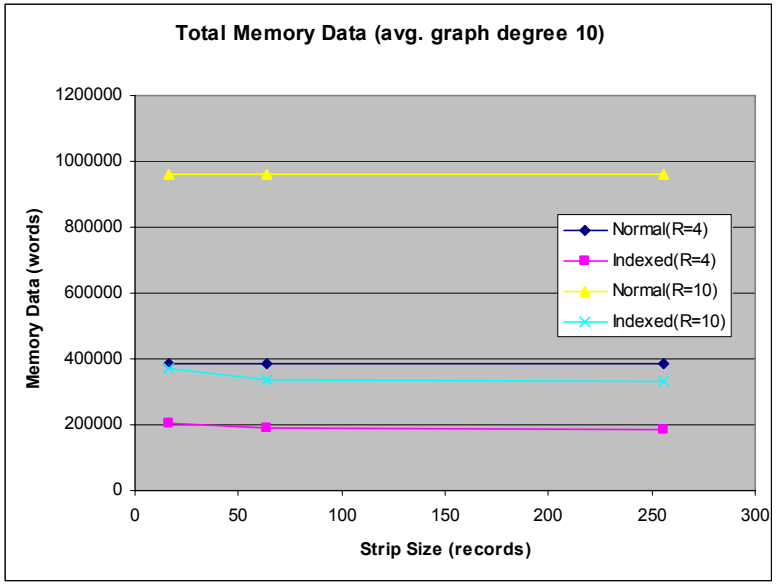
– 4:1 reduction in SRF bandwidth due to indexing
– Further SRF BW reduction due to bank conflicts
– Extra hardware
– Deadlock issues

Reducing Bank Conflicts by Memory Placement

- Use Hashing in static mesh to reduce bank conflicts
- Application dependent preprocessing, *pseudo hashing*.
- Application independent hashing
  - Multiplication Method
    - $h(k) = \lfloor m\ (kA - \lfloor kA \rfloor) \rfloor$
      - m = number of memory banks.
      - k = flat index of the vertex
      - For a constant A, $0 < A < 1$,
- Result: Both give a similar sort of performance on an average. Stick with independent hashing.

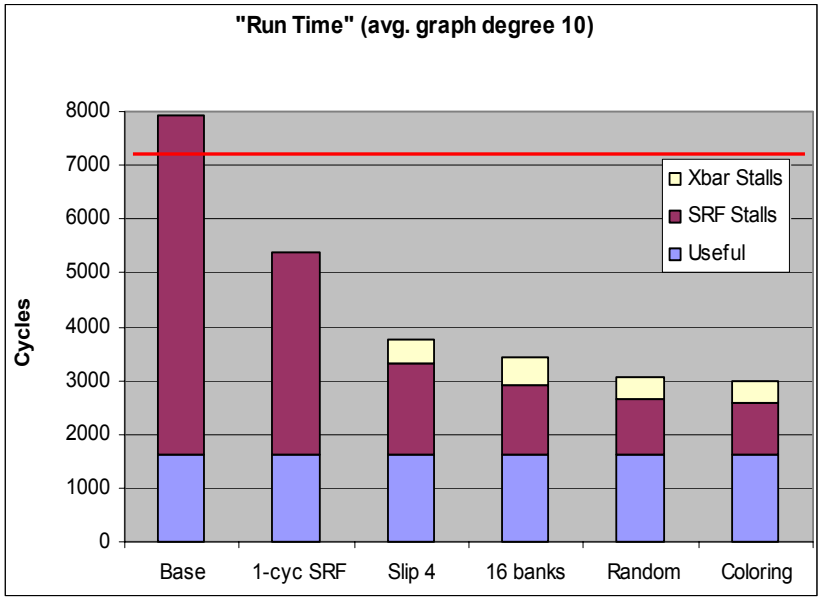
Application-level Impact of Indexing

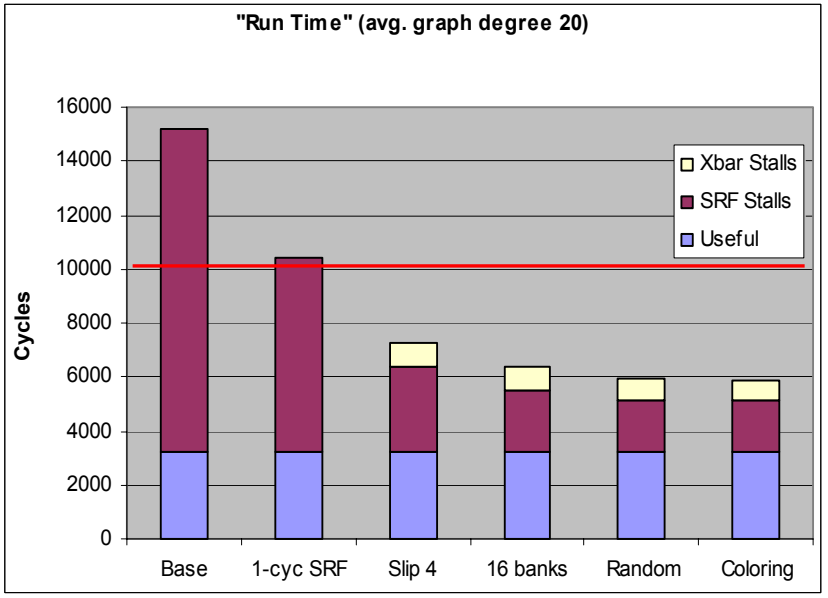




- *Normal* strip sizes reduce with increasing node degree, *Indexed* case fairly static
- *Normal* memory traffic increases with node degree *Indexed* traffic fairly static

Kernel-level Impact of Indexing

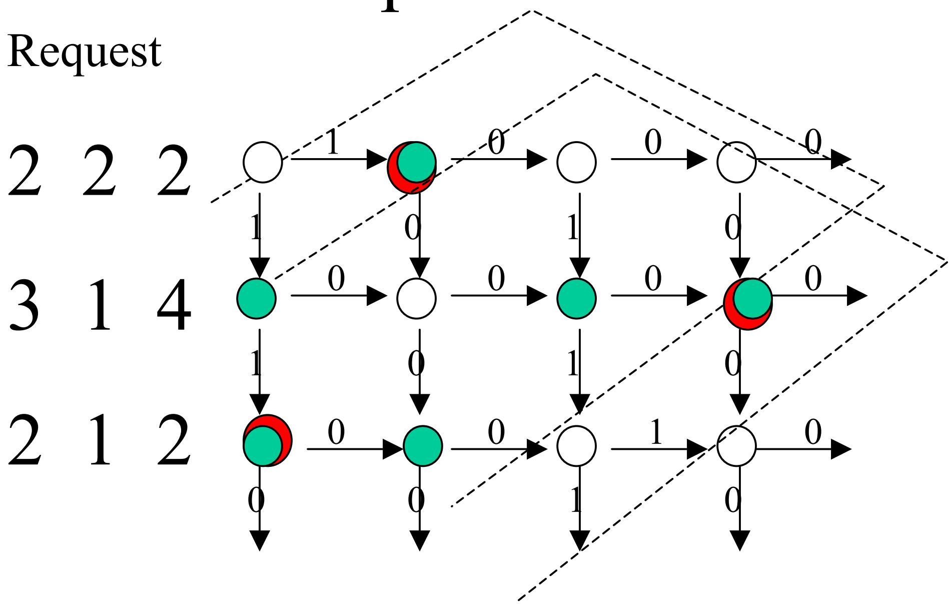




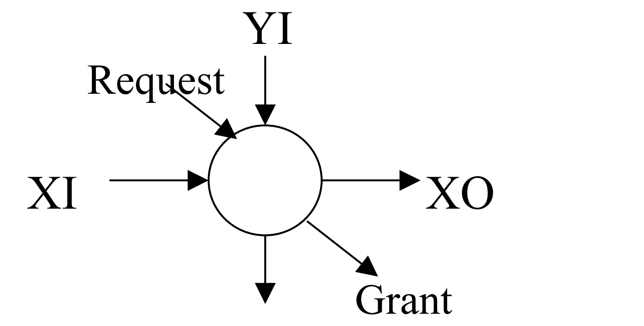
- Trace-driven simulation
- No computation (SRF accesses only)
- Ideal memory system (Imagine parameters)
- Application still memory-limited

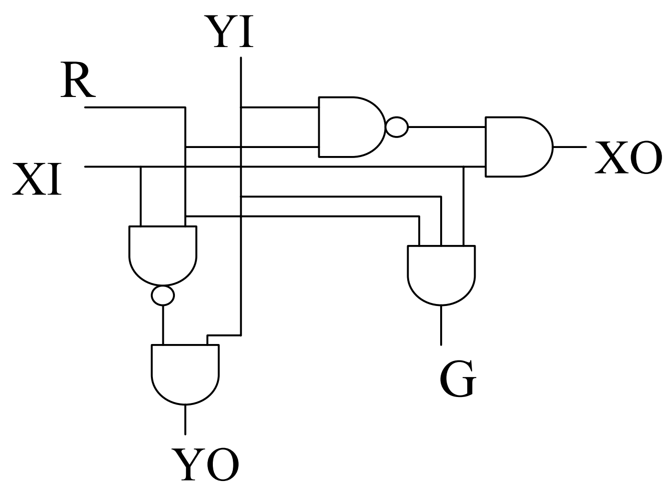
Reducing Bank Conflicts in Hardware: Wave Front Allocator

- Concept



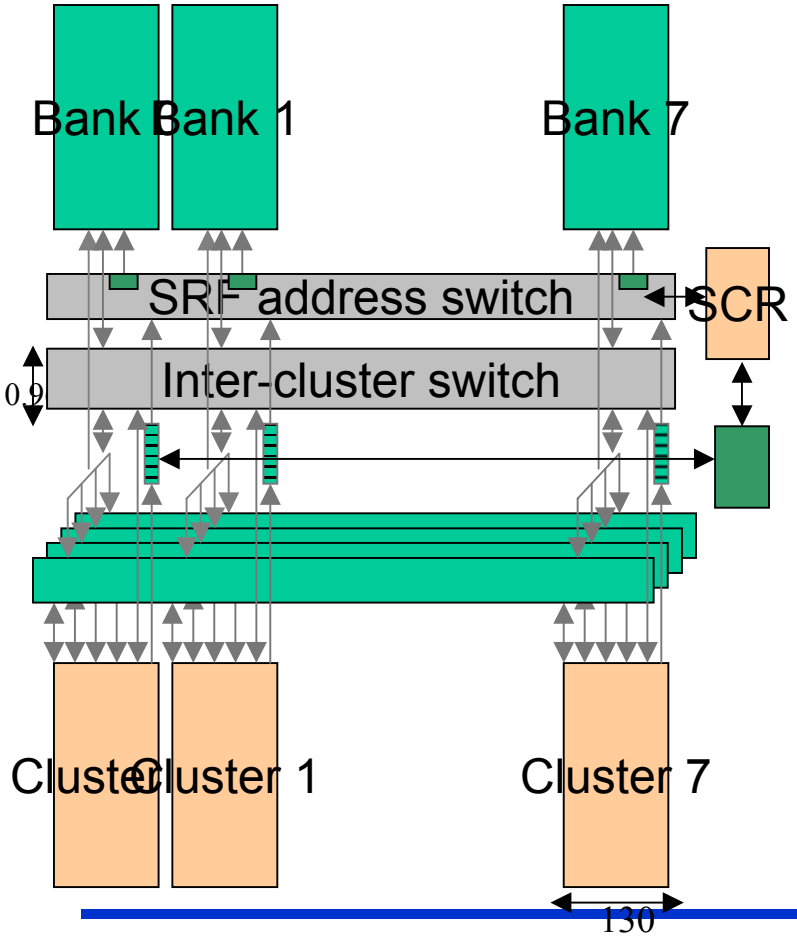
- Cell





YO

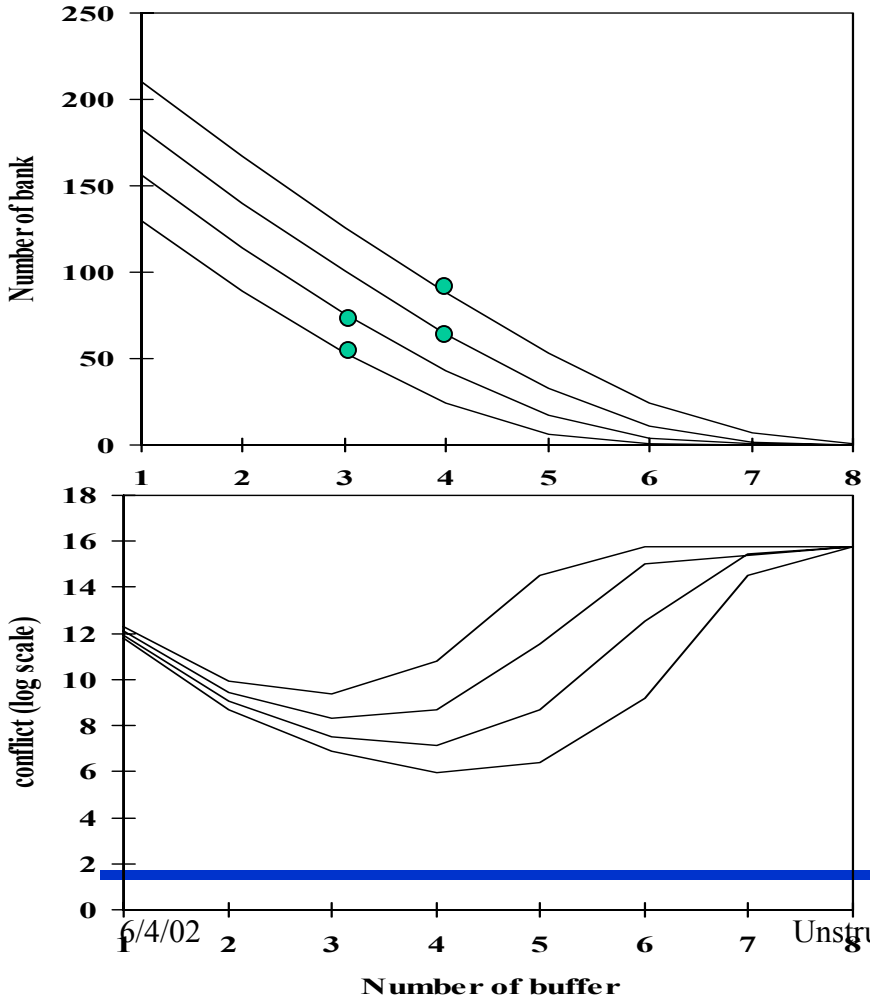
Cost Estimation



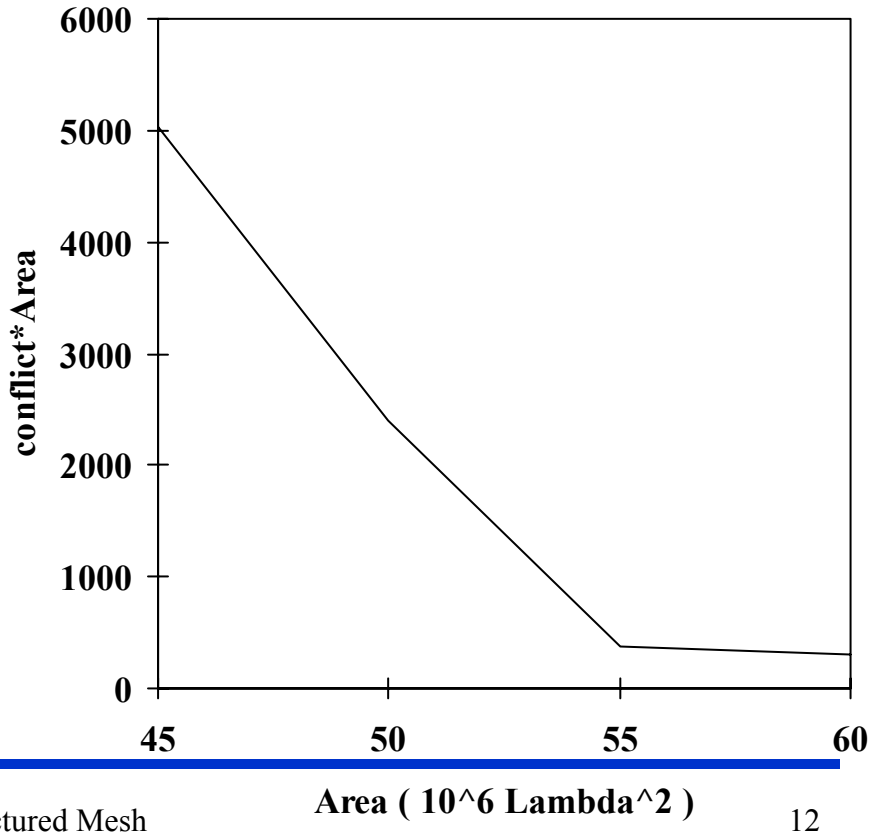
- Added cost component
  - ✓ Bus between buffer and allocator
  - ✓ Allocator
  - ✓ Switch and Switch control
  - ✓ Etc(request buffer, register for book-keeping…)
- Cost = 7280 (# of buffer) + 166(# of bank) + 1040 log(# of buffer) + 333 log(# of buffer)

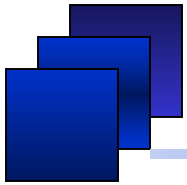
Cost and performance

- Equal cost line



- Cost vs Conflict





Conclusion

- Unstructured mesh have lots of data replication which hog memory bandwidth
  - Cache addition effectively multiplies memory bandwidth
  - SRF Indexing improves bandwidth and strip size
    - Banks conflicts can be reduced by hashing and hardware arbitration
- Open areas of explorations
  - Dynamic meshes