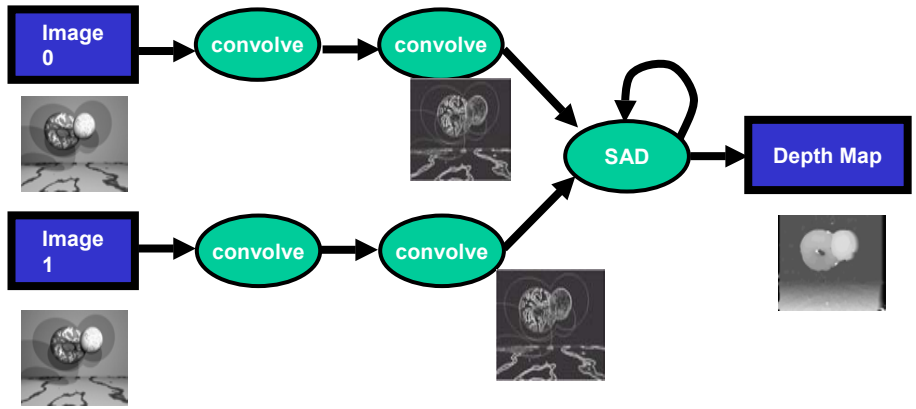
Stream Processor Architecture

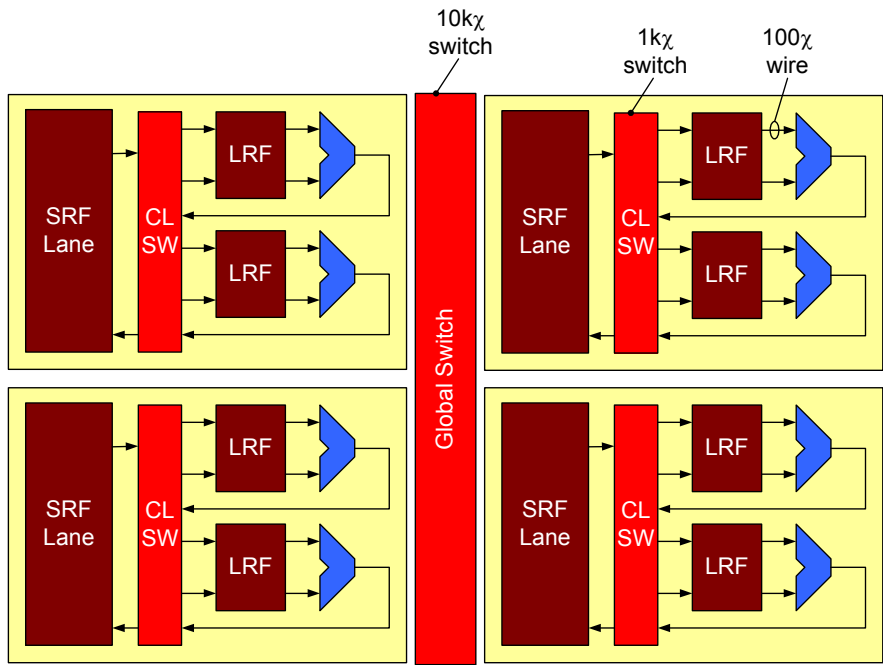
William J. Dally
Stanford University
August 22, 2003
Streaming Workshop

Some Definitions

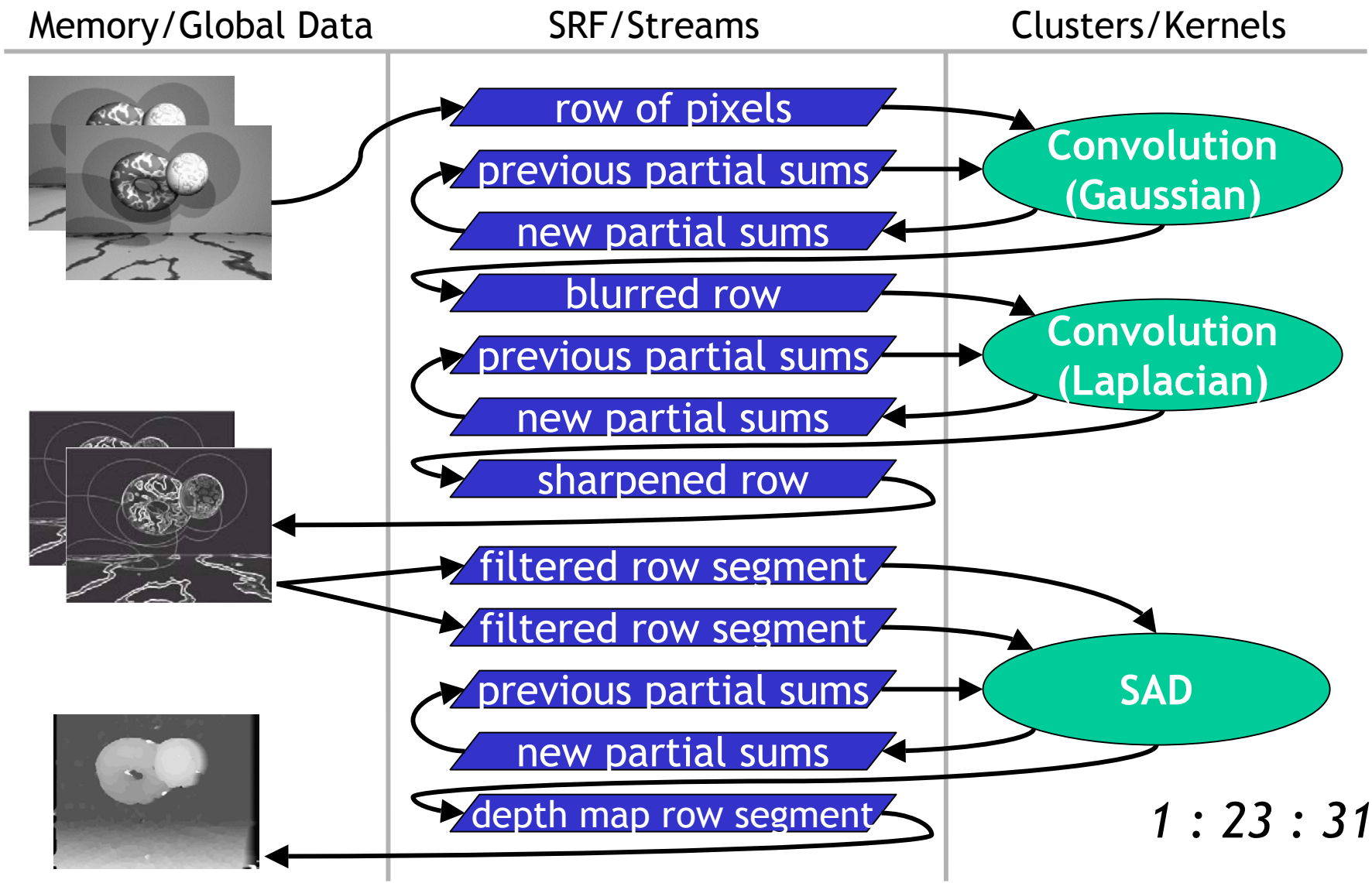
- A *Stream Program* expresses a computation as *streams* flowing through *kernels*



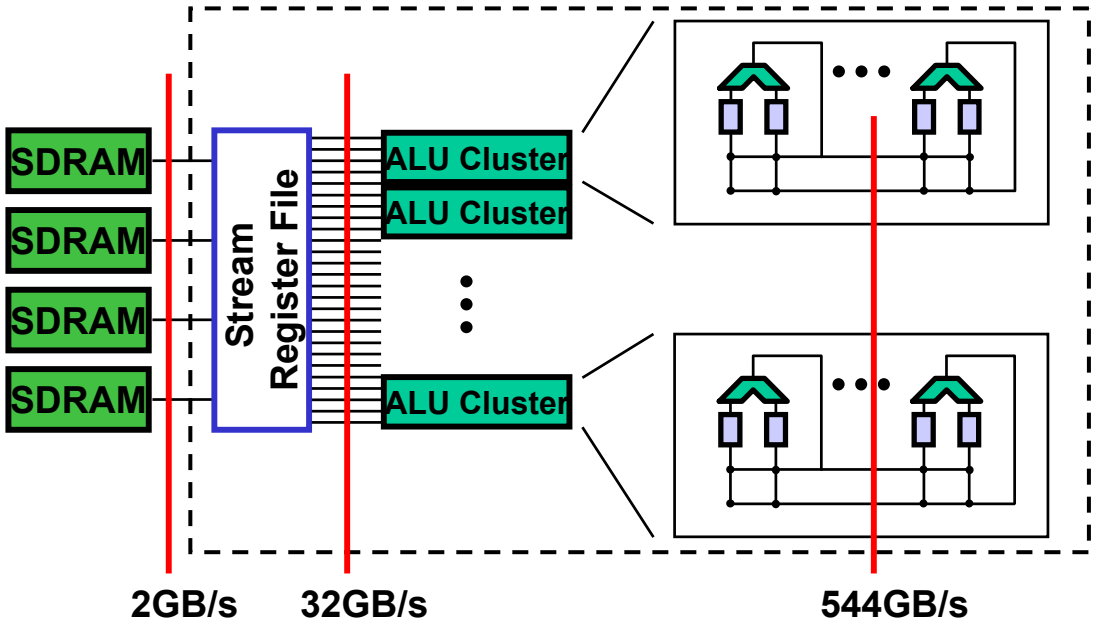
- A *Stream Processor* exploits the locality and concurrency in a stream program to use lots of ALUs with little communication



Producer-Consumer Locality in the Depth Extractor

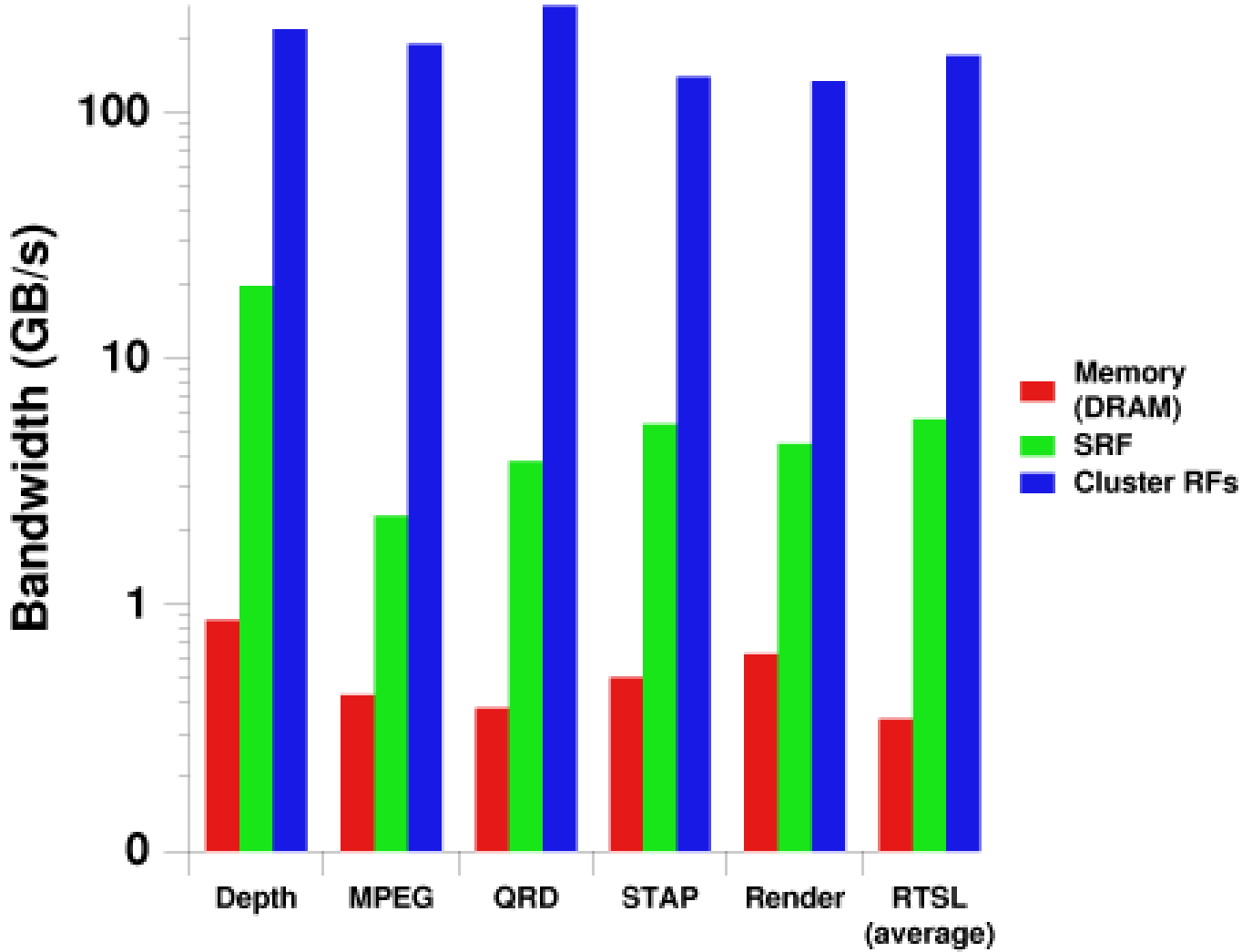


A Bandwidth Hierarchy exploits kernel and producer-consumer locality



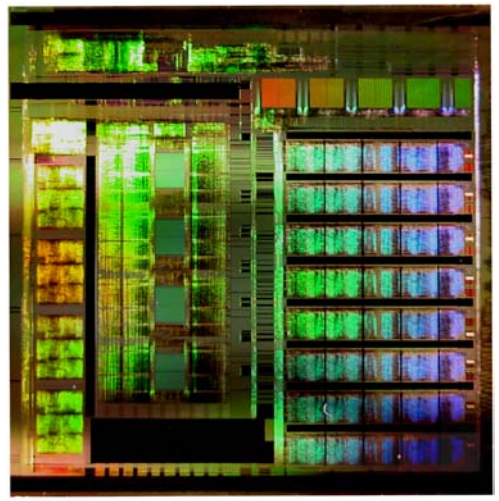
	Memory BW	*Global RF BW*	*Local RF BW*
Depth Extractor	0.80 GB/s	18.45 GB/s	210.85 GB/s
MPEG Encoder	0.47 GB/s	2.46 GB/s	121.05 GB/s
Polygon Rendering	0.78 GB/s	4.06 GB/s	102.46 GB/s
QR Decomposition	0.46 GB/s	3.67 GB/s	234.57 GB/s

Bandwidth demand of stream programs fits bandwidth hierarchy of architecture

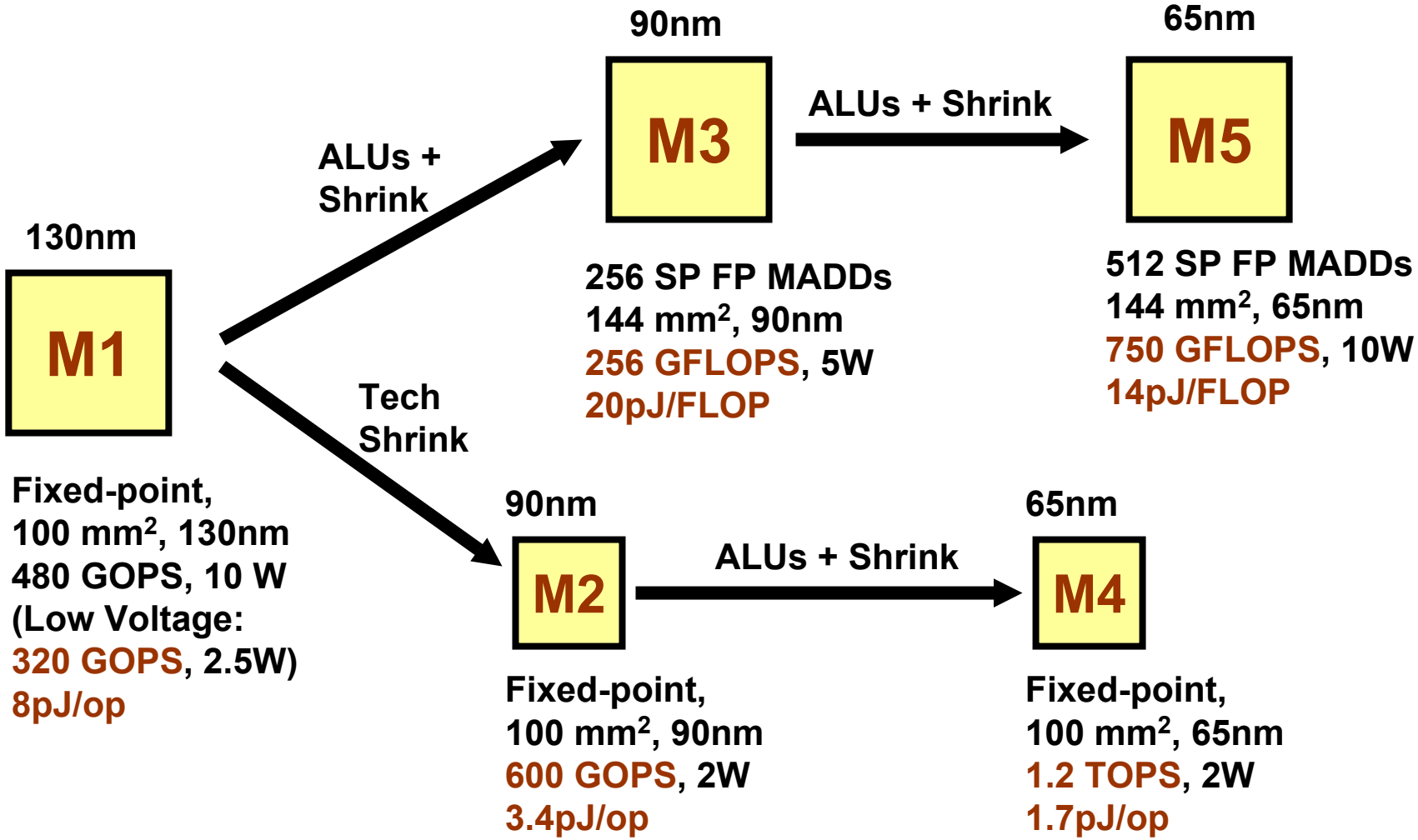


Prototype HW and SW

- Prototype of Imagine architecture
  - Proof-of-concept 2.56cm^2 die in 0.15um TI process, 21M transistors
  - Collaboration with TI ASIC
- Dual-Imagine development board
  - Platform for rapid application development
  - Test & debug building blocks of a 64-node system
  - Collaboration with ISI-East
- Software tools based on Stream-C/Kernel-C
  - Stream scheduler
  - Communication scheduling
- Many Applications
  - 3 Graphics pipelines
  - Image-processing apps – depth, MPEG
  - 3G Cellphone (Rice)
  - STAP



Stream Processor Roadmap



Streaming Scientific Applications

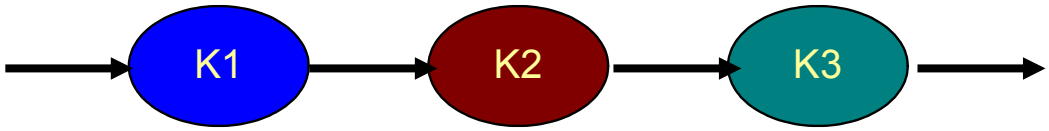
Application	GFLOPS (out of 64[1])	FLOPs/ Mem ref	LRF Refs	SRF Refs	Mem Refs
StreamFEM (Euler, quad)	32.2	23.5	169,505,648 (93.6%)	10,299,776 (5.7%)	1,354,448 (0.7%)
StreamFEM (MHD, cubic)	33.5	50.6	733,294,080 (94.0%)	43,762,752 (5.6%)	3,165,280 (0.4%)
StreamMD (gridded)	23.3[2]	14.3	427,743,216 (96.5%)	9,505,088 (2.1%)	5,978,848 (1.4%)
StreamFLO (key kernels[3])		50	(96%)	(2%)	(2%)

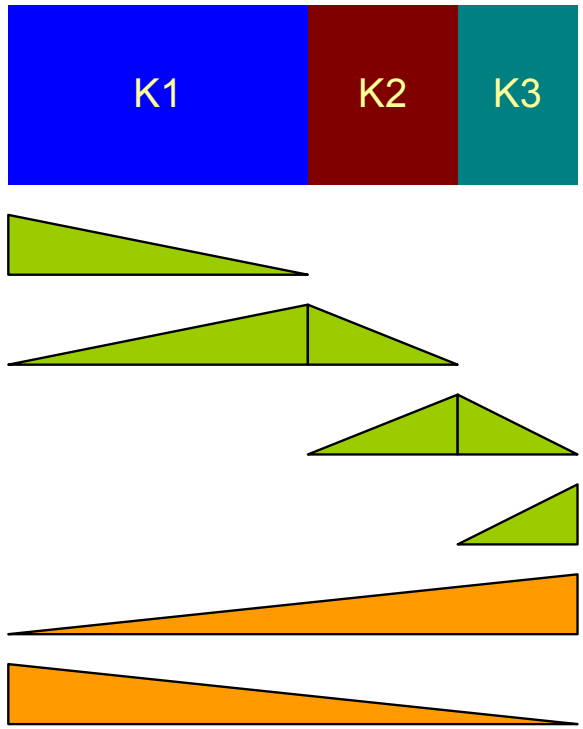
[1]Simulations run on version of simulator with 64GFLOPS nodes.

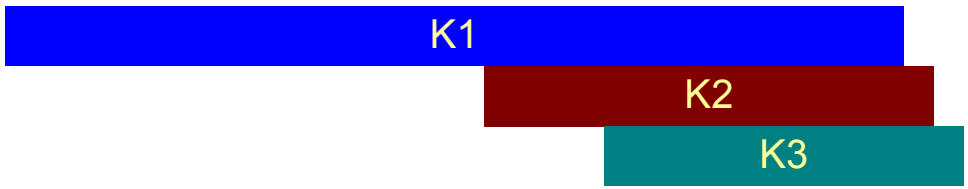
[2]Stream MD performance limited by false dependency.

[3]Estimated from key kernels.

Streaming in Time and Space



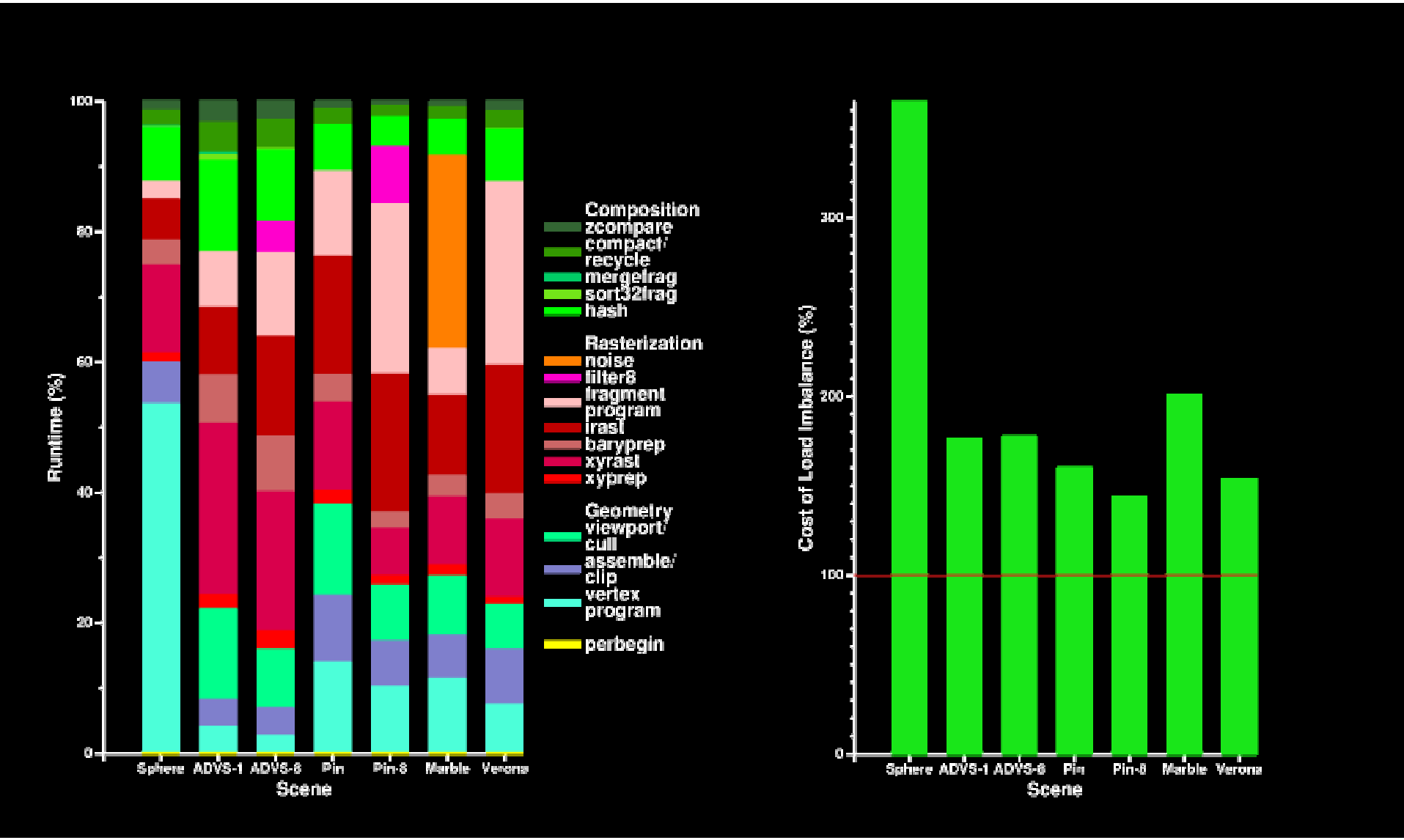




Space Multiplexing
+ Little storage required
+ Exploits control parallelism
- Load imbalance
- MIMD control
- Requires IPC

Time Multiplexing
+ Perfectly load balanced
+ Exploits data parallelism
+ SIMD control (power & area)
- Requires storage (SRF)

Load Imbalance in OpenGL Pipeline vs Scene



Some Interesting Questions & Topics

- Streamifying compiler
  - Automatically convert "C" or Fortran to kernels and streams
- Locality enhancement
  - Program transformations to enhance use of SRF
- What applications do and don't stream well?
  - All applications with data parallelism do stream well (dependence distance)
  - For those that don't, why don't they – (no DP, dependences, control…)
- Aspect ratio
  - How much DP vs ILP vs TLP
- Storage architecture
  - SRF – indexing, switching, virtualization
  - LRF – partitioning, switching
- Conditionals
  - How much MIMD is needed?

Conclusion

- Stream programs expose locality and concurrency
- Stream processors exploit these properties
  - Concurrency uses lots of ALUs and hides latency
  - Locality reduces communication and makes it explicit
  - Partition kernels in time, space, or both
- Imagine demonstrates stream processing for *media* applications
  - Many applications demonstrated
  - pJ/op can be made <2x that of special-purpose systems
- Merrimac exploring stream processing for scientific applications
  - 1/0.3TFLOPS peak/sustained per node vs. 10/0.5 GFLOPS
  - Global memory bandwidth is still an issue
- Many challenging questions and topics remain
  - Compilation, Architecture, and Applications

My project is a stream processor too…