The Case for Malleable Stream Architectures

Christopher Batten\textsuperscript{1,3}, Hidetaka Aoki\textsuperscript{2}, Krste Asanović\textsuperscript{3}

\textsuperscript{1} Department of Electrical Engineering and Computer Science
  Massachusetts Institute of Technology, Cambridge, MA

\textsuperscript{2} Central Research Laboratory
  Hitachi, Ltd., Tokyo, Japan

\textsuperscript{3} Department of Electrical Engineering and Computer Science
  University of California, Berkeley, CA

Workshop on Streaming Systems
  November 8, 2008
The Case for Malleable Stream Architectures

Key Characteristics of Stream Programs

Types of Parallelism

- DLP: Data-Level Parallelism
- KLP: Task-Level Parallelism
- KLP: Pipeline Parallelism

Other Characteristics

- Data-dependent control flow
- Communication patterns
- Real-time constraints

MIT/UCB
Christopher Batten
The Case for Malleable Stream Architectures

Mapping Stream Programs to Stream Architectures

Temporal Data-Level Parallelism

Temporal Kernel-Level Parallelism
Mapping Stream Programs to Stream Architectures

Temporal Data-Level Parallelism | Spatial Data-Level Parallelism
---|---
Temporal Kernel-Level Parallelism

The Case for Malleable Stream Architectures
Mapping Stream Programs to Stream Architectures

Temporal Data-Level Parallelism
Temporal Kernel-Level Parallelism

Spatial Data-Level Parallelism
Spatial Kernel-Level Parallelism
### Comparison of Stream Program Mappings

#### Temporal DLP
- Temporally amortize control & synchronization overheads
- Efficiently saturate off-chip memory bandwidth

#### Spatial DLP
- Spatially amortize control & synchronization overheads
- Efficiently saturate off-chip memory bandwidth
- Trivial load-balancing assuming no data-dependent control flow

#### Temporal KLP
- Exploit producer-consumer locality to reduce buffering
- Reduce per-element latency

#### Spatial KLP
- Exploit producer-consumer locality to reduce buffering
- Reduce per-element latency
- Easy to map data-dependent control flow
- Good utilization for stateful kernels
Example Stream Processors

**NVIDIA GTX 200**
- 30 Cores
- 8 Lane Vector Units
- Inter-kernel buffering usually stored in DRAM
- Difficult to exploit KLP spatially

**SPI Storm-1**
- 1 Core
- 16 Lane Vector Unit
- 32b Subword SIMD
- Inter-kernel buffering blocked in stream register file
- Cannot exploit KLP spatially

**Tilera TILE64**
- 64 Cores
- 32b Subword SIMD
- Inter-kernel buffering routed through static network

The Case for Malleable Stream Architectures
Programmers and architects should first leverage DLP execution whenever possible.

Energy Efficiency • Memory Bandwidth Utilization • Load Balancing
Our Position: Exploit DLP First Then KLP

Programmers and architects should first leverage DLP execution whenever possible

Energy Efficiency • Memory Bandwidth Utilization • Load Balancing

Programmers and architects must still be able to efficiently exploit KLP, but only after DLP

Minimize Buffering • Reduce Latency • Data-Dependent Conditionals
The Case for Malleable Stream Architectures

Maven: Malleable Array of Vector-Thread Engines

MIT/UCB

Christopher Batten