Cache Refill/Access Decoupling for Vector Machines

Christopher Batten, Ronny Krashinsky, Steve Gerding, Krste Asanović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
December 8, 2004
Cache Refill/Access Decoupling for Vector Machines

• Intuition
  – Motivation and Background
  – Cache Refill/Access Decoupling
  – Vector Segment Memory Accesses

• Evaluation
  – The SCALE Vector-Thread Processor
  – Selected Results
Turning access parallelism into performance is challenging.
Turning access parallelism into performance is challenging

Target application domain
- Streaming
- Embedded
- Media
- Graphics
- Scientific
Turning access parallelism into performance is challenging.

Applications with Ample Memory Access Parallelism

Processor Architecture

Modern High Bandwidth Memory Systems

Target application domain
- Streaming
- Embedded
- Media
- Graphics
- Scientific

Techniques for high bandwidth memory systems
- DDR interfaces
- Interleaved banks
- Extensive pipelining
Turning access parallelism into performance is challenging. Many architectures have difficulty turning memory access parallelism into performance since they are unable to fully saturate their memory systems.
Turning access parallelism into performance is challenging.

- Applications with Ample Memory Access Parallelism
- Processor Architecture
- Modern High Bandwidth Memory Systems

Memory access parallelism is poorly encoded in a scalar ISA.

Supporting many in-flight accesses is very expensive.
Turning access parallelism into performance is challenging

Applications with Ample Memory Access Parallelism

Vector Architecture

Modern High Bandwidth Memory Systems

Supporting many in-flight accesses is very expensive
Turning access parallelism into performance is challenging

- Applications with Ample Memory Access Parallelism
- Vector Architecture
- Non-Blocking Data Cache
- Modern High Bandwidth Main Memory

A data cache helps reduce off-chip bandwidth costs at the expense of additional on-chip hardware.
Each in-flight access has an associated hardware cost.
Each in-flight access has an associated hardware cost
Saturating modern memory systems requires many in-flight accesses.

Main Memory Bandwidth-Delay Product

\[
\begin{align*}
\text{1 element} & \quad \times \quad 100 \text{ cycles} \\
\text{cycle} & \quad \implies \quad 100 \text{ in-flight elements}
\end{align*}
\]
Caches increase the effective bandwidth-delay product

2 elements per cycle × 100 cycles = 200 in-flight elements
Goal For This Work

Reduce the hardware cost of non-blocking caches in vector machines while still turning access parallelism into performance by saturating the memory system.
In a basic vector machine a single vector instruction operates on a vector of data.
In a basic vector machine a single vector instruction operates on a vector of data.

\[ vlw \ vr2, \ r1 \]
In a basic vector machine a single vector instruction operates on a vector of data.

vadd vr0, vr1, vr2
In a basic vector machine a single vector instruction operates on a vector of data.
In a decoupled vector machine the vector units are connected by queues.
Non-blocking caches require extra state to manage outstanding misses
Control processor issues a vector load command to vector units
Vector load unit reserves storage in the vector load data queue.
If request is a hit, then data is written into the VLDQ.
VEU executes writeback command to move data into architectural register.
On a primary miss, cache allocates a new miss tag and replay queue entry.

- Target register specifier
- Cache line offset
- Other management state
On a primary miss, cache allocates a new miss tag and replay queue entry.
On a secondary miss, cache just allocates a new replay queue entry.
Processor is free to continue issuing requests which may hit in the cache.
When the refill returns from memory, the cache replays each pending access.
When the refill returns from memory, the cache replays each pending access.
Expensive hardware is required to support many in-flight accesses.
Effective decoupling requires command and data queuing
Effective decoupling requires command and data queuing
Effective decoupling requires command and data queuing.

Diagram showing the relationship between the CPU (CP), VLIU (VLU), VLIU-Command Queue (VLU-CmdQ), VLIU-Execute Unit (VEU), VLIU-Command Queue (VEU-CmdQ), and VSU (VSU) with their respective data and tags.
Effective decoupling requires command and data queuing
Saturating memory system with many misses requires additional queuing.
Saturating memory system with many misses requires additional queuing
Saturating memory system with many misses requires additional queuing.
Saturating memory system with many misses requires additional queuing.
Refill/access decoupling
prefetches lines into cache
Refill/access decoupling prefetches lines into cache

- Acts as **inexpensive** and **non-speculative** hardware prefetch
- Only need one prefetch per cacheline
- Prefetch requests are cheaper than the actual accesses
The vector refill unit brings lines into the cache before the VLU accesses them.
The vector refill unit brings lines into the cache before the VLU accesses them.
The vector refill unit brings lines into the cache before the VLU accesses them.
The vector refill unit brings lines into the cache before the VLU accesses them.
The vector refill unit brings lines into the cache before the VLU accesses them.
The vector refill unit brings lines into the cache before the VLU accesses them.
VRU reduces need for hardware which scale with number of in-flight elements.
VRU reduces need for hardware which scale with number of in-flight elements
VRU reduces need for hardware which scale with number of in-flight elements

Decoupled Vector Machine

Decoupled Vector Machine with VRU
VRU reduces need for hardware which scale with number of in-flight elements
Vector segment memory accesses explicitly capture two-dimensional access patterns and thus make the memory system more efficient.
Using **multiple strided accesses** for 2D access patterns is inefficient

```assembly
la r1, A
li r2, 3
vlbst vr0, r1, r2
addu r1, r1, 1
vlbst vr1, r1, r2
addu r1, r1, 1
vlbst vr2, r1, r2
```
Using multiple strided accesses for 2D access patterns is inefficient.

```
la   r1, A
li   r2, 3
vlbst vr0, r1, r2
addu r1, r1, 1
vlbst vr1, r1, r2
addu r1, r1, 1
vlbst vr2, r1, r2
```
Using **multiple strided accesses** for 2D access patterns is inefficient.
Using multiple strided accesses for 2D access patterns is inefficient

```
la   r1, A
li   r2, 3
vlbst vr0, r1, r2
addu r1, r1, 1
vlbst vr1, r1, r2
addu r1, r1, 1
vlbst vr2, r1, r2
```

Multiple strided access do not capture the spatial locality inherent in the 2D access pattern.
Vector segment accesses perform the 2D access pattern more efficiently

```assembly
la r1, A
vlbseg 3, vr0, r1
```
Vector segment accesses perform the 2D access pattern more efficiently

```asm
la    r1, A
vlbseg 3, vr0, r1
```
Vector segment accesses perform the 2D access pattern more efficiently.

```assembly
la r1, A
vlbseg 3, vr0, r1
```
Vector segment accesses perform the 2D access pattern more efficiently.

```assembly
la   r1, A
vlbseg 3, vr0, r1
```
Vector segment accesses perform the 2D access pattern more efficiently

``` Assembly
la r1, A
vlbseg 3, vr0, r1
```
Vector segment accesses perform the 2D access pattern more efficiently.

```assembly
la r1, A
vlbseg 3, vr0, r1
```
Vector segment accesses perform the 2D access pattern more efficiently

Efficient encoding
- More compact command queues
- VRU process commands faster

Captures locality
- Reduces bank conflicts
- Moves data in unit-stride bursts

```
la r1, A
vlbseg 3, vr0, r1
```
Cache Refill/Access Decoupling for Vector Machines

• Intuition
  – Motivation
  – Background
  – Cache Refill/Access Decoupling
  – Vector Segment Memory Accesses

• Evaluation
  – The SCALE Vector-Thread Processor
  – Selected Results
**SCALE Vector Processor**

**Key Features**
- 4 lanes, 4 clusters
- Cluster for indexed accesses
- 4 segment address generators
- 4 VLDQs
- VRU includes throttle logic, refill address generator

**Diagram:**
- Control Proc
- Vector Execution Unit
  - Lane 0
  - Lane 1
  - Lane 2
  - Lane 3
- VRU
  - Throttle Logic
  - Refill
  - Unit Stride
- SEG
SCALE Cache

Key Features
- Unified I/D cache
- Two cycle hit latency
- Four 8 KB banks
- 32 way associative
- 32B cache lines
- 16B/cycle per bank
- Four 16B segment buffers per bank
Methodology and Kernels

• Simulation methodology
  – Microarchitectural C++ simulator of SCALE vector processor and non-blocking multi-banked cache
  – Main memory is modeled with a simple pipelined magic memory
  – Benchmarks were compiled for the control processor with gcc and key kernels were coded by hand in assembly

• 14 kernels with varying access patterns
  – vvaddw: Add two word element vectors and store result
  – hpg: 2D high pass filter on image with 8 bit pixels [EEMBC]
  – rgbyiq: RGB to YIQ color conversion with segments [EEMBC]
Normalized performance for \texttt{vvaddw} with varying queue sizes

### Configuration

- Limit study with very large queue sizes except for queue under consideration
- 8B/cycle bandwidth and 100 cycle latency main memory
- Normalized performance with and without the vector refill unit
Normalized performance for \texttt{vvaddw} with varying queue sizes

Vector Load Data Queues

Replay Queues

Maximum Queue Size

Decoupled Vector Machine

Decoupled Vector Machine with Vector Refill Unit
Normalized performance for hpg with varying queue sizes

Vector Load Data Queues

Replay Queues

Decoupled Vector Machine
Decoupled Vector Machine with Vector Refill Unit
Normalized performance for \texttt{rgbyiq} with varying queue sizes

- **Vector Load Data Queues**
- **Replay Queues**

- **Decoupled Vector Machine**
- **Decoupled Vector Machine with Vector Refill Unit**
Normalized performance for rgbYiq with varying queue sizes

Vector Load Data Queues

Replay Queues

Maximum Queue Size

Decoupled Vector Machine
Decoupled Vector Machine with Vector Refill Unit

Dashed lines indicate segments are turned into strided accesses
Performance with refill/access decoupling scales well with longer memory latencies

Configuration
- Includes the VRU
- Reasonable queues and buffering
- 8B/cycle mem bandwidth
- VLDQ and replay queues are a constant size
- Command queues and miss tags are scaled linearly with latency
Paper includes additional results and analysis

- 14 kernels with varying access patterns
- Performance versus number of miss tags
- Performance versus memory latency and bandwidth
- Comparison with an approximation of a scalar machine
- Various VRU and VLU throttling schemes
Related Work

• **Refill/Access Decoupling**
  - Software prefetching
  - Second-level vector register files [NEC SX, Imagine]
  - Speculative hardware prefetching [Jouppi90, Palacharla94]
  - Run-ahead processing [Baer91, Dundas97, Mutlu03]

• **Vector Segment Memory Accesses**
  - Streaming loads/stores [Khailany01, Ciricescu03]
Conclusions

• Saturating large bandwidth-delay memory systems requires many in-flight accesses and thus a great deal of access management state and reserved element data storage.

• Refill/access decoupling and vector segment accesses are simple techniques which reduce these costs and improve performance.