Modified from `final_cs_2024_lhz.pdf`, as cheatsheet in final test.

1 DRAM

1.1 Relacement Policies

Insertion policy: when a block is first inserted into the cache.
Promotion policy: when a block is hit.
LRU: insert/promote to MRU,
LIP: insert to LRU, promote to MRU, BIP:insert with small probability at MRU, others at LRU
DIP: dynamically select between LRU and BIP.
Shadow/ghost tag arrays: simultaneously track the behaviors of accessingdata.
set dueling: assume hit/miss is uniform accross sets. delicate some ofsets to P0, others to P1, track which has smaller miss, othersfollow the winner.
LFU: least frequently used, use a counter.
FBR: frequency basedreplacement, do not increment counter for recently referenced.
LRFU:least recently frequently used, use a weighted value F(x) to each pastreference, x is the distance from that reference to the currentreference.

1.2 Non-Blocking Caches and Data Prefetching

track outstanding miss: when a miss occured, and when we are loadingthat block, if re-access that block, put it in MSHR.
MSHR(Miss Status Handling Register) keep track of each outstanding missto one cache block.
A MSHR include: valid bit, block address, pending load/store entries. Each entry has: valid bit, type&format, block offset, destinationsource register
prefetch choice: directly into cache, seprate stream buffer of Nblocks.
selective prefetch: only prefetch when detecting stream access patterns
stride detection: For each PC, remember the stride. Remember the last address used for this PC, Compare to currently used address for this PC
Track confidence using a two bit saturating counter.
temporal correlation: two addresses accessed nead each other in time arelikely to be accessed near each other in the future.
Markov prefetcher: pair-wise address correlation. prefetch width, prefetch depth.

1.3 Main memory - DRAM

physically seperated from processor chip, connected through memory channels.
DRAM: higher capacity, little bit slower than SRAM, high bandwith due to high parallelism.
Structure: 1.CPU socker 2.channels(CPU bin count, Fully independent access) 3.ranks(bus signal integrity, Independent access inside chips, butshare I/O) 4. chips(Fully synchronous access across all chips in a rank) 5. banks(chip area efficiency, Independent access inside banks, but share I/O (chip pins and internal bus)).
Access granularity:
1. same bank in all chips of a rank contributes a subset of dat(spatially)
2. bytes are transferred on the bus sequetially in multiple cycle(temporally)
DDR: double data rate, transfer data on both rising and falling edges ofthe clock.
Channel level parallelism, NUMA(non-uniform memory access).
Different ranks share command/address/data lines.
Different chips in a rank share command/address lines, but have different data lines.
Banks operate independently, but share command/address/data buses.
Concurrent access in Buses can overlap latencies.
Inside a DRAM bank, 2^n rows, 2^m columns, w bits per word.
Row buffer: $2^m \times w/8$ bytes, also called DRAM page, same size as arow.
Row and column share same address pins.
Burst transfer: each read/write transfers multiple consecutive columnsas a burst.

policy:
1. open-page policy: expect a hit access, leave row open afteraccess.
2. close-page policy: expect a conflict, PRECHAGE after access.
3. adaptive page policy: predict whether would conflict.
scheduling policies: 1. FCFS 2. FR-FCFS: first ready, first come, first serve. Row-hit first, then oldest first
DRAM refresh: DRAM cell capacitor charge leaks over time, need to touch each cell periodically, usually 64ms. Distributed refresh.

2 Memory Coherence

2.1 Cache Coherence

Coherence protocol defines a set of states and a sequence of actions needed to ensure caches are coherent in multi-core systems.
Basic rule of coherence: single-writer, multiple-readers (SWMR).
MSI:
1. modified(M): one cache has valid and latest copy,
2. shared(S): one or morecaches and memory have valid copy,
3. invalid(I): not latest version.
MR $\rightarrow$ SW: must invalidate all S copies before entering M state,
SW $\rightarrow$ MR: M state must downgrade to S when others ask for the data.
How to realize MSI: 1. snooping: all caches broadcast their misses through interconnect.
2. directory-based, use directory to track state of each block. Could reduce messages.
False Sharing: Different memory locations mapped to same cache block, traffic congestion among cores.

2.2 Networks On-Chip(NoC)

Cache Banking: like small caches in large cache block. Each cache block can only reside in one cache bank each time. Lower latency, more parallelism. a block is preferred to be placed in the bank closer to the core with the most accesses to it, and may also be migrated between banks(to minimize physical distance and access latency).
Set associate happens in banks, banking is another dimension on top of sets and ways.
1. Message: sofware/system level. arbitrary size
2. Packet: network-level. bounded size
3. Flit: switch-level. Fixed size, unit of control flow
4. Phit: link-level. Fixed size,unit of data transferred on link per cycle
Network Latency: Zero-load latency = header latency + serialization latency.
Header latency T_h = $H \times (t_r + t_l)$ (H hop count, t_r route delay, t_l link delay).
Serialization latency $T_s = L/B$ (L packet size, B link bandwidth).
Latency = Zero-load latency + queuing delay.

2.2.1 Topology

e.g. Linear (1D mesh), Ring (1D torus), 2D mesh, 2D torus.
Routing distance: number of hops on a route from source to destination
Diameter: maximum routing distance
Bisection bandwidth: bandwidth crossing a minimal cut that partitions the network in two equal-sized halves.

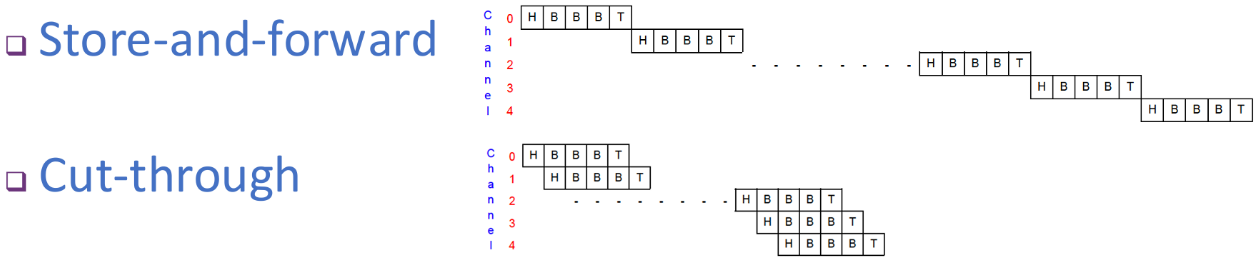
2.2.2 Routing

Possible Properties:
1. Deterministic: always select the same path every time
2. Minimal: only select a shortest path
3. Oblivious: route is decided without considering current network state such as traffic congestion
4. Adaptive (the opposite of oblivious): route is influenced by current network state,
5. Source: entire route is determined at the source

6. Incremental (the opposite of source): the route is incrementally determined at each hop
Examples:
e.g. **Dimension-order**: minimal&oblivious.
e.g. **Mesh**: deterministic.
e.g. **Torus**: 1.up/right/choice 2.down/left (deterministic) 3. randomly
e.g. **Adaptive**: 1.minimal adaptive: choose the lowest load among a minimal path. 2. Fully adaptive: choose lowest load at each hop, may encounter livelock.

2.2.3 Flow Control

Bufferless FC: 1. Dropping drop the packet if it losses resource arbitration. re-transmits: use NACK signal/timeout
2. Misrouting: route the losing package away. May livelock.
3. Circuit-Switching: use a request to set up a path and reserve links, tail flit release sources. Efficient if many data are sent, wasteful, longer latency
Buffered FC 1. Packet Granularity: allocate buffer and link in unit of pack ets. Store&forward, cut&through. No alloaction and control cost for each flits. Inefficient buffer capacity use, need large buffer, only support limited packet size, unfair, losing packege have long latency.
2. Wormhole: like cut-through, but buffer space allocated to flits. In unit of flits/packets. More efficient buffer space utilization. head-of-line blocking when waiting in the same buffer
3. Virtual-Channel: Provision multiple virtual channels (VCs, queues) per physical port. independent flits of different packets from the same port can use different VCs. Significanly reduce blocking. more complex.

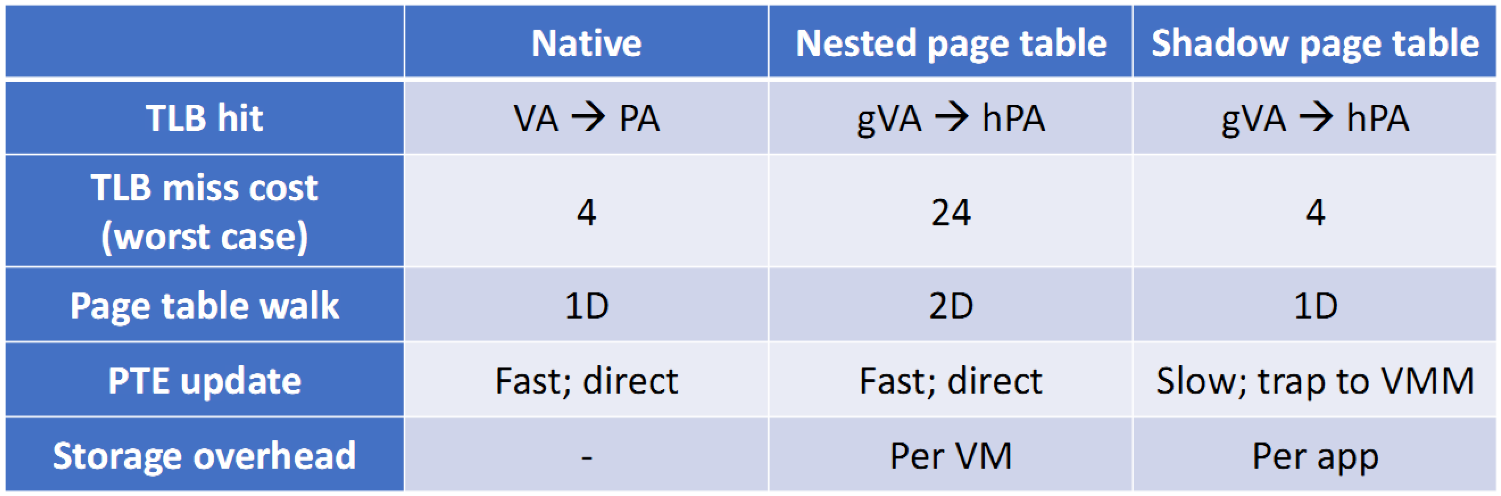


Backpressure:informs the upstream router to stop sending the next packet/flit.
e.g. Credit-based flow control: track space available in downstream. send 1 flit consume 1 credit. remove 1 from buffer, sent 1 upstream of credit.
Resource ordering: impose a partial order on resources, enforce allocating resource in non-descending order

3 Virtual

virtual page issue 1. homonym(same virtual address, different physical address): only one copy in cache, one process reads data of the other process. Be resolved by flushing cache on context switch/adding process ID to tags
2. synonym(different virtual address, same physical address): writes to one copy will not be reflected in the other copy
Physical Tagged Caches: translation, caches access in parralel, start index to cache with page offset only. tag check uses full physical address. need set size $\leq$ page size.
Virtual Machines:
Type 0 hypervisor: hardware-based solutions; hypervisor in firmware. Type 1 hypervisor: OS-like soft ware; 'the datacenter OS'. Type 2 hypervisor: simply a process on host OS.
VM options:
Paravirtualization: running transparent to apps but not OS, low over head, cannot be used by arbitrary OS.
Full virtualization: transparent to apps and OS, performance overhead.
VM Issue1: Priviledge mode.
Guest OS can only run in user mode of host machine. Divide user mode to virtual kernel mode and virtual user mode. trap and emulate: When wants to execute a priviledge instruction, trap to physical kernel mode, return to guest OS.
Problem: requires all sensitive instructions are privileged.
Solution: 1. use paravirtualization 2. binary translation 3. hardware support (new modes,new instructions, VMCS VM control structure).

VM Issue2: Address Translation.
guested virtual address(gva) → guested physical address(gpa)(page table in guest OS)
host virtual address(hva)=gpa → host physical address(hpa)(page table in VMM)
each vm page maps gva → gpa, but hardware uses hpa
TLB directly cache gva → hpa.
Shadow Page Table: gpa → hpa, keep shadow page table consistent with page table in guest OS.



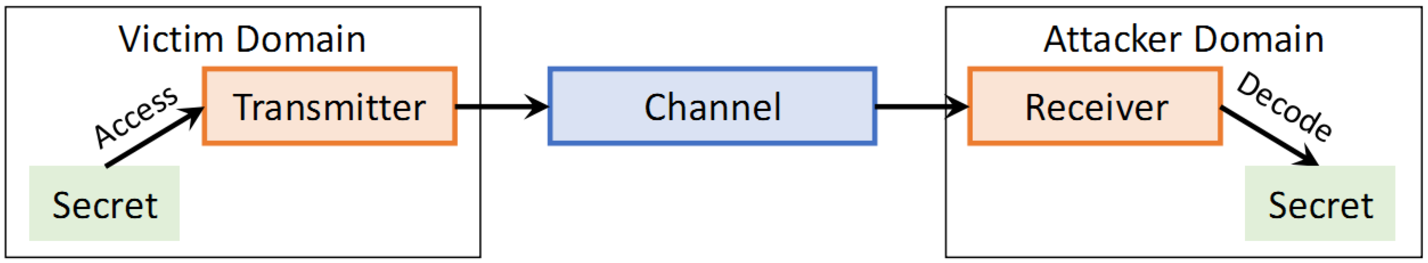
	Native	Nested page table	Shadow page table
TLB hit	VA → PA	gVA → hPA	gVA → hPA
TLB miss cost (worst case)	4	24	4
Page table walk	1D	2D	1D
PTE update	Fast; direct	Fast; direct	Slow; trap to VMM
Storage overhead	-	Per VM	Per app

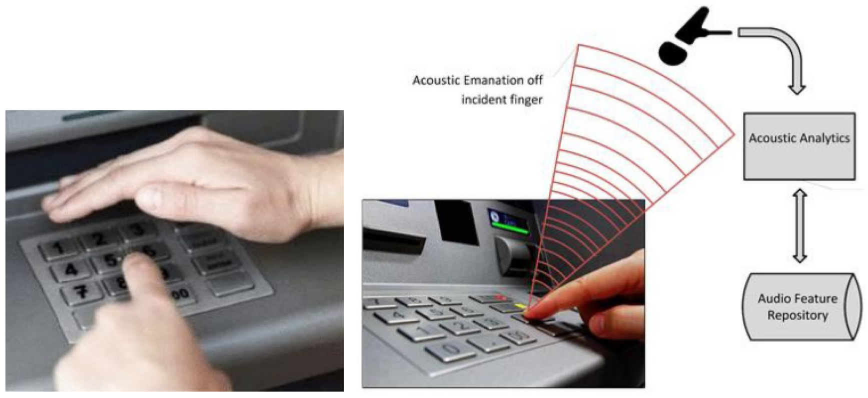
4 Security

4.1 Channels

covert channels: intended communication between parties(communicatingwithout let OS know).
side channels: unintended communication between parties.

Real-World Example: ATM Acoustic Channel





- Secret: Pin
- Transmitter: keypad
- Channel: air
- Modulation: acoustic waves
- Receiver: microphone
- Decoder: a simple classifier

type of channels:
1. physical:power,sound, etc.
2. timing:execution time(data dependent round loop leads to different running time)

3. microatchitectural:prime-probe, prime access cache blocks, secret access same cacheline, access again to test whether previous data still in
4. transient attack: sepculative execution might bring unexpected data into cache

```
BR:     if (x < size_array1) {
LD1:        secret = array1[x];
LD2:        y = array2[secret * 64];
        }
```

Precondition: train branch predictor to predict executing LD1/2.
Transmit: out-of-bound x values, let array1[x] points to desired address.
Receive: probe cache to infer which cache line of array2 was fetched

```
BR:     if (...) { ... }
        ...
LD1:    secret = array1[x];
LD2:    y = array2[secret * 4096];
```

Precondition: train BTB properly, use an instruction aliasing to BR to train BTB to jump to LD1.
Transmit&Receive: Similar to the last example.
Temporal partition: execute sequentially, clear and reset states in between.
Spatial partition: divide resources and as signs to each app
Intel Cache partition Technology(CAT): Divide the cache by ways or by sets, only can use its own partition. Underutilization, Scalability(apps $\leq$ num).
Noises: Pros: simple, no performance overhead; Effective. Cons: no effect for no-relative time attack. no effect to big-timing difference. Affect usability if benign app need good timer.

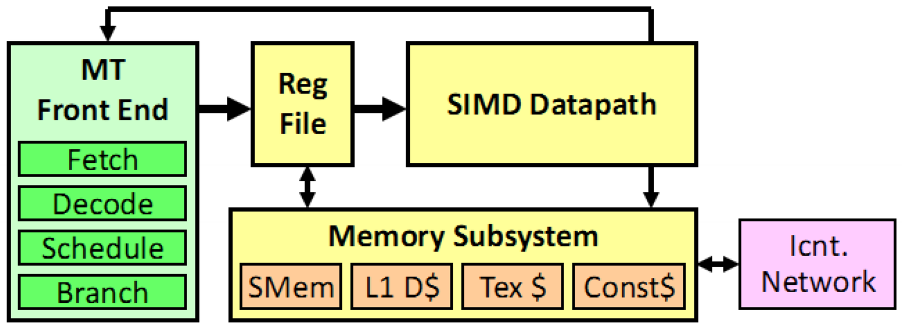
4.2 TEE and Secure Processor

Focus on Intel SGX. Pogram in the enclave is trusted, all other software is untrusted.
Need to ensure non-enclave code cannot access enclaved data.
Attestation, Isolation, Off-chip data protection.
Platform attestation:"authentic platform"(public-key cryptography) public key and private key.
Enclave measurement: "right application", use digital signature. send back hash digest.
Hardware manages a reversed page table (RPT), stored securily in "enclave page cache mapping" (EPCM). Entry: PA $\rightarrow$ VA, enclaved ID, attributes.
Memory encryption engine(MEE). Confidentially:encryption, integrity: MAC, Freshness:MAC + counter, Merkle tree.

5 GPU

Warp: unit of scheduling in hardware. size is implementation-specific, typically 32 threads.
Different warps have different progress (PC value). All warps keep their state on-chip for fast switch. Need large files and on-chip memories. Lower utilization and performance loss when threads in a warp branch differently. Basic blocks with no active threads are not executed
Atomic operations to L2/global memory. Synchronization at grid level.
Barrier synchronization within a thread block. Synchronization within grid level, atomic operations to L2/global memor
Streaming Multiprocessor: one thread block is mapped to one SM. GPU relies on
1. large off-chip memory bandwith
2. Tolerate long latency by switching between warps
3. Optimize software program to maximize aligned and coalesced access

GPU Microarchitecture



- MT frontend, SIMD backend
- Frontend: multi-threading
  - Track dependencies and thread readiness
  - Interleave warp execution to hide latency
  - Handle branch divergence
- Datapath: multiple SIMD functional units
- Large register files: register values of all threads stay inside core

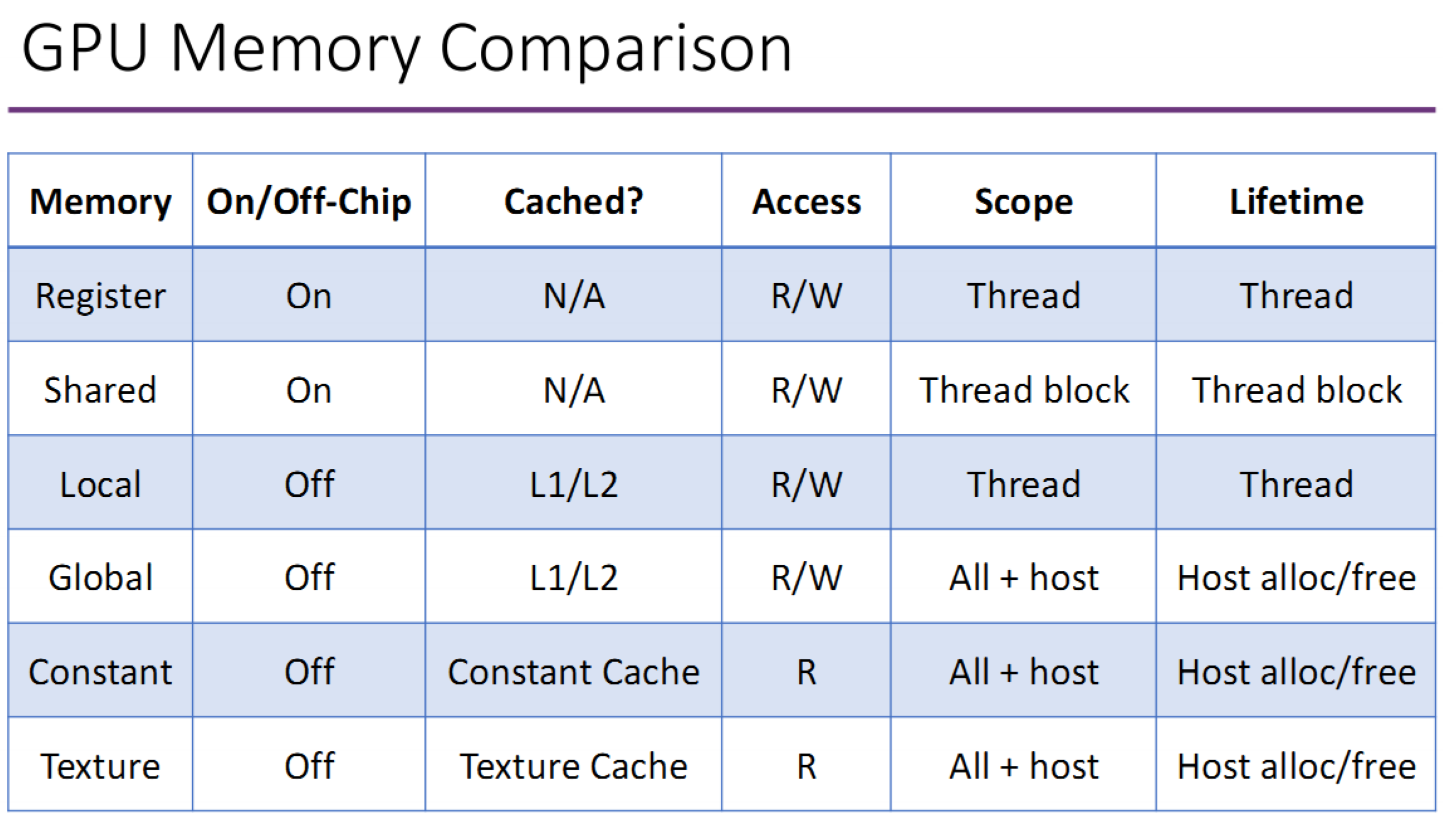
CPU and GPU have physically and logically seperate address space. Recent GPUs support unified virtual address space. Performance overhead is significant.
Optimization: overlap computations and communications with asynchronous transfer.
Partition workloads on a problem.

6 Customized Hardware & Accelerators

$$\text{EPI} = \frac{\text{Perform}}{\text{power}} = 1/\frac{\text{Energy}}{\text{Operation}}$$

L1 cache, shared memory, gloabl L2, read only data cache.
Line buffer: a specialized cache for stencil.
Keep the lines containing the input stencil.
In steady slides, fetch one new pixel in and an old out.
Double Buffer: two banks switch roles between fetching data from memory and serving data to computation. Overlap computation with memory access. Cost:$2\times$ capacity.
Reconfigurable Architecture. Configuration:realizing a specific design. Different configurations implement different custom hardware.
FPGA: 2D array of reconfigurable logic elements: Look-up tables (LUTs), flip flops, DSP blocks, adders, block RAM (BRAMs). Static programmable interconnect.
LUT: all possible calues of output bits, which are selected by input bits. A$m \to n$LUT can be implemented by an SRAM of $2^n \times m$bits, n-bit input is address. Or a set of $2^n m$ bits registers and a 2^n way MUX. n bits input is MUX selection.
Operator Mapping: Suppoer various operators, with different data dimensions. Dataflow is for maximize utilization and reuse.

7 Appendix



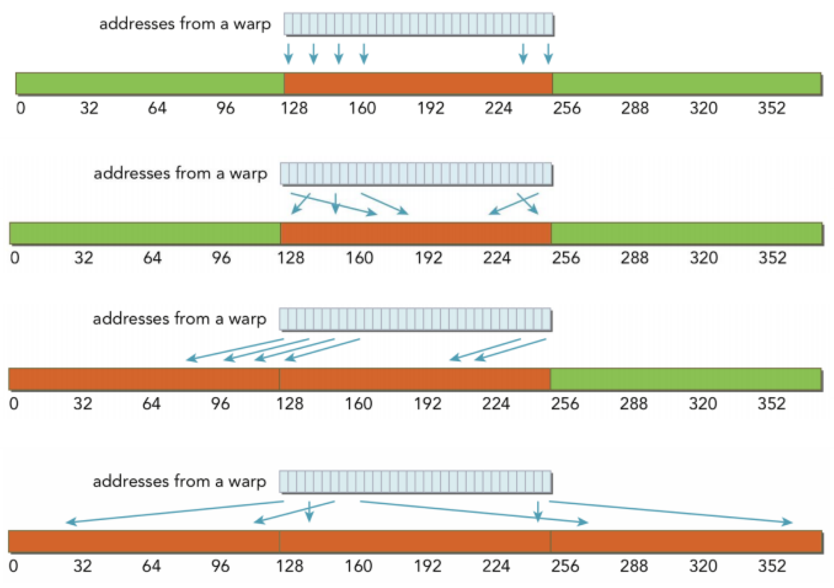
GPU Memory Comparison

Memory	On/Off-Chip	Cached?	Access	Scope	Lifetime
Register	On	N/A	R/W	Thread	Thread
Shared	On	N/A	R/W	Thread block	Thread block
Local	Off	L1/L2	R/W	Thread	Thread
Global	Off	L1/L2	R/W	All + host	Host alloc/free
Constant	Off	Constant Cache	R	All + host	Host alloc/free
Texture	Off	Texture Cache	R	All + host	Host alloc/free

GPU parrallelism is limited by both computational and storage resources. Registers and shared memories are limited per SM resources

Aligned and Coalesced Access

- Aligned and coalesced
  - Require only a single access to global memory
- Coalesced and misaligned
  - Require two accesses, and only half of the fetched data are used
- Misaligned and uncoalesced
  - Many more accesses and bytes
  - Even the other data may still be used by later accesses, cache may not be large enough



Comparison for CPU and GPU hierarchy: larger register files, smaller and shallow cache hierarchy. Main memory: limited capacity, but extremely high bandwidth.
GPU and CPU is connected by a fast bus(PCIe) offloading: copying between device and host memory