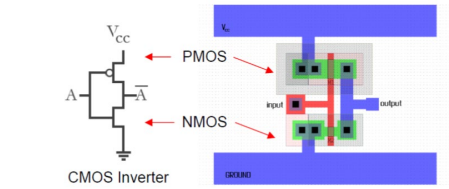
CA-01: 集成电路到数据中心

App<SW sys<HW sys < Module < Gates<Circuits<Devices<Physics

N-type:电子再留，高电压关闭



Active areas for thin oxide region

Polysilicon for the gate

Metal for interconnection

Contact (Via) for inter-layer connection

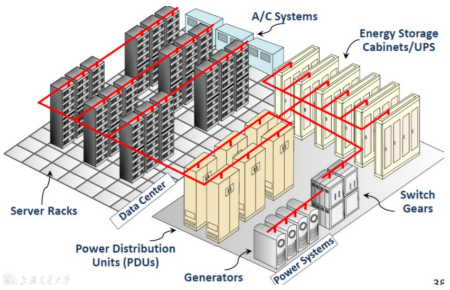
IC cost = Die cost + Testing cost + Packaging cost / Final test yield

Die cost = Wafer cost / Dies per Wafer * Die yield

Dies per wafer = π * (Wafer_diam / 2)² / Die Area − π * Wafer_diam / √2 * Die Area − Test dies

Die Yield = Wafer yield * {1 + Defects_per_unit_area * Die_Area}^−α

- *Die Cost ∝ die area*⁴



CA-02:指令集架构

ISA defines: data, control flow，编程者可见状态，instr format+semantic

Stack

Operations that perform i++

```
PUSH  &i ;
PUSH  0x01;
ADD;
```

Accumulator

Operations that perform i++

```
LDA  &i ;
ADDA  0x01;
STA  &i;
```

Register-Memory

Operations that perform i++

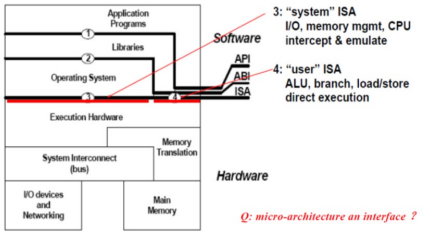
```
MOVE  d0, &i ;
INC  d0;
MOVE  &i, d0;
```

Register-Oriented

Operations that perform i++

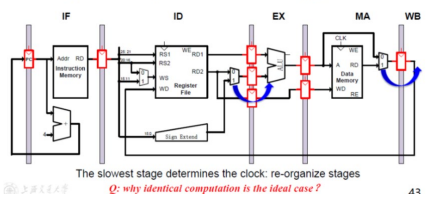
```
LD  r7, &i ;
ADD  r7, 0x01, r7;
ST  r7, &i;
```

CISC: less code, stack-oriented. Reg-mem, condition codes; RISC: reg-oriented, load-store. No con codes



User ISA: data flow, ALU ops, control flow; **sys ISA:** manage shared resource, privilege, mem/cpu/mem

- Uniform sub-computations => balancing pipeline states
- Identical computations => unifying instruction types
- Independent computations => minimizing pipeline stalls



Merge sub-computation: less reg+wait; subdivide: tinier bottleneck

More->better:no, +reg, +overhead

CA-03: 指令级并行(ILP)

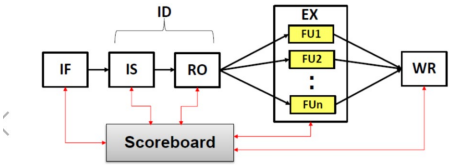
Dependencies: data, control, name

Data: decode 时可检查出来

Name: same reg/mem, no data 交换

- Anti-dependence
  - Instruction j writes a register or memory location that instruction I reads from (instruction I is executed first)
- Output dependence
  - Instruction I and instruction j write the same register or memory location (must preserve program order)

WAR. WAW: non in typical pipeline



- **Scoreboard**: A central control to prevent hazard
  - **Issue**: check for structural/WAW hazard; stall issue until clear
  - **Read operands**: read operands if no RAW hazards
  - **Execution**: followed by notification to scoreboard
  - **Write result**: checks for WAR; stall write until clear

Instruction Status

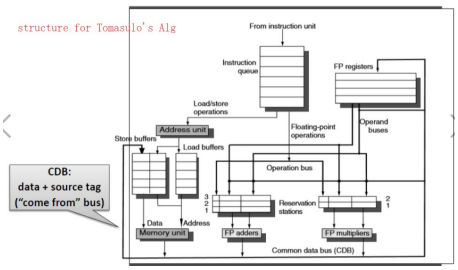
	j	k	IS	RO	EX	WR
1. LD	F6	34	R2	C1	C2	C3	C4
2. LD	F2	45	R3	C5	C6	C7	C8
3. MUL.D	F0	F2	F4	C6	C9	C19	C20
4. SUB.D	F8	F6	F2	C7	C9	C11	C12
5. DIV. D	F10	F0	F6	C8	C21	C61	C62
6. ADD.D	F6	F8	F2	C13	C14	C16	C22

Register Results Status

F0	
F2	
F4	
F6	
F8	
F10	Divide

Functional Unit Status

Time	Name	Busy	Op	Fi	Fk	Qj	Qk	Rj	Rk	Dest.	S1	S2	FU for j	FU for k	Fj ok?	Fk ok?
	Integer	No														
	Mult1	No														
	Mult2	No														
	Add	No														
0	Divide	No														



Reservation station: fetch and buffer an operand ASAP. When all operands read, enable associated FU

Both RS and L/S buffer have tag

Reg renaming: in RS，消除 name dep

More RS than reg,

- Instruction Status (Three stages)
  - Issue: get instruction from FP Op Queue (in-order)
  - Execution: operate on operand (may be out of order)
  - Write result: finish execution (may be out of order)
- Data flow approach: operations proceed as soon as their operands are available (instruction wake up)

Instruction Status

	j	k	IS	EX	WR
1. LD	F6	34	R2	C1	C2	C4
2. LD	F2	45	R3	C2	C3	C5
3. MUL.D	F0	F2	F4	C3	C6	C16
4. SUB.D	F8	F6	F2	C4	C6	C8
5. DIV. D	F10	F0	F6	C5	C17	C57
6. ADD.D	F6	F8	F2	C6	C9	C11

Register Results Status

F0	M*R
F2	M[A2]
F4	
F6	M-M+M
F8	M-M
F10	M*R/M

Reservation Stations

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

	Busy	Addr
Load1	No	
Load2	No	
Load3	No	

Load Buffer

Assumptions: Load (2 cycles), Add (2 cycles), Mult(10 cycles), Divi (40 cycles)order issue, out-of-order execution, out-of-order completion

Tomasulo: RS 分布式控制、CDB 广播所有 res, tags 对 value 进行认证

Tomasulo vs Scoreboard: RS 分布式，结果 bypass 到 FU，common bus

CA-04: 指令级并行 II

Multiissue: superscalar/ VLIW

1. simple scalar: IPC = 1, simple operation latency(SOL)=1,1 instr/cycle才能充分利用并行
2. superscalar pipeline (width=n): IPC = n, simple operation latency = 1, instr/cycle=n
3. superpipelined(degree=m): IPC = 1 (1/m of the base cycle), SOL=m, instr/cycle=m
4. VLIW machine: IPC = n instr/cycle = 1 VLIW /cycle, SOL=1, instr/cycle=n

- An exception is precise if
  - The exception is associated with a particular faulting instruction ***Fi***
  - All instructions (in **program order**) prior to ***Fi*** complete execution
  - ***Fi*** and all subsequent instructions **appear to have never started**
- Maintain precise exception implies in-order completion

precise exception和misprediction recovery是一个问题

- Need HW buffer for results of uncommitted instructions:
  - order in which instructions are completed
  - order in which memory is accessed
- A completion buffer managed as circular queue (FIFO)
  - Contains all the instructions that are in flight
  - Holds result between complete and commit

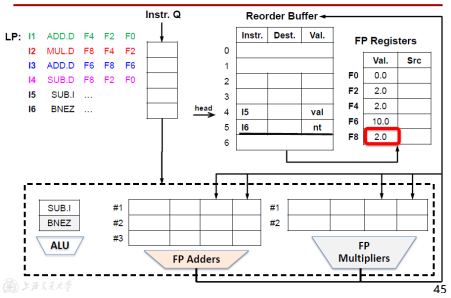
包括：type (L/S/ALU), destination(reg/mem), value(holds value直到指令

ROB-RS called instr window: its size determines the degree of ILP

ROB: instr type + dest + value

Tom ROB: issue+exe+write+commit

"Tomasulo+ROB" Example – Cycle 23



Explicitly Parallel Instruction Computing

EPIC (style of architecture) is an evolution of VLIW which has absorbed many of the best ideas of superscalar processors

EPIC Philosophy:

- Providing the ability to design the desired plan of execution (POE) at compile-time
- Providing features that permit the compiler to "play the statistics".
- Providing the ability to communicate the POE to the hardware.

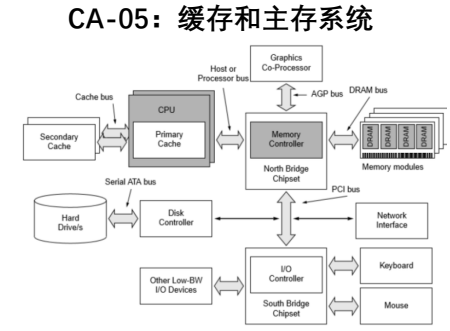
- Intel/HP Itanium (IA-64) follows EPIC philosophy
  - 6-wide, 10-stage pipeline at 800Mhz

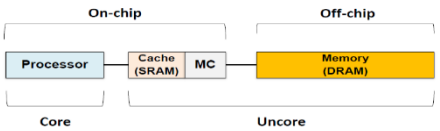
Category	CISC	RISC	VLIW
Inst Size	Varies	Fixed (typically 32bits	Fixed
Inst Format	Field placement varies	Regular, consistent placement of fields	
Inst Semantics	Complex; possibly many dependent ops per instr	Almost always one simple operation	Multiple, independent simple operations
Registers	Few, sometimes special	Many, general purpose	
Memory	Reg-Mem architecture	Load/Store architecture	
Hardware	Exploit microcode	Not microcoded implementation	
Example:			

- Complexity of scheduling is moved into the compiler
  - Compiler creates each VLIW word
  - Detects hazards and hides latencies
  - Optimizations to fill the slots in instruction

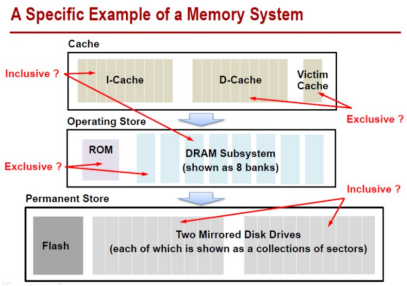
- **Structural hazards?**
  - No two operations to the same functional unit
  - No two operations to the same memory bank
- **Data hazards?**
  - no data hazards among instructions in a bundle
- **Control hazards?**
  - Predicated execution
  - Static branch prediction

CA-05：缓存和主存系统





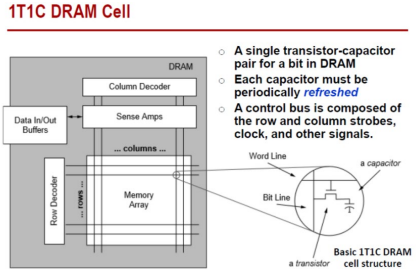
Directed map: block 唯一出现在某处
Fully associated: a block anywhere
Set associated: 限定位置
3C model: compulsory miss: first access to a block not in the cache;
Capacity miss: cache cannot contain all; **conflict miss:** if too many blocks mapped to its set



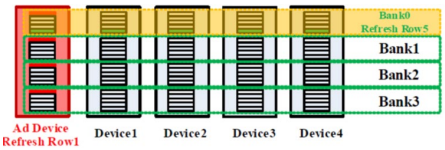
Oftentimes, a system is comprised of many independent DIMMs
-Each DIMM may contain one or more independent ranks
-Each rank is a set of DRAM devices that operate in unison
-Each DRAM device has one or more independent banks
-Finally, each bank is comprised of slaved memory arrays

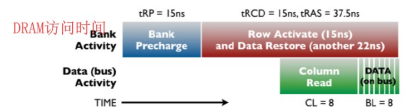
xN DRAM: number of output pins
x4 DRAM has at least 4 mem arrays (in a bank), col width is 4 bits (each column read/write transmits 4 bits).

- Rank is a separately addressable set of DRAM devices. It allows for DIMM-level independent operation.
- Each set of memory arrays (inside of a DRAM device) that operates independently is referred to as a bank.
- Multiple banks are provided so we can be simultaneously working on different requests (*interleaving*)
- Having concurrency at the rank and bank levels provides bandwidth through the ability to pipeline requests



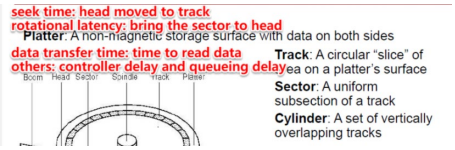
A row is the smallest refresh unit in bank. Per-rank/ per-bank refresh
Per-rank refresh 中, banks 没法访问
物理地址直接映射到内存大致包括: channel ID, rank ID, bank ID, row ID, column ID



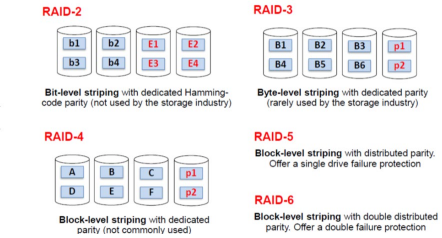


Bank 读 row 到 row buffer 中, bus 根据 col 找 data, row buffer 是 DRAM 的 buffer;

CA-06：数据存储和输入输出



ATA:并行短线,不同 data 传输 mode
Drive interface: simple protocol(短握手); high autonomy(少 cpu);high data rate(别太多); overlapping cmd(多 disk),cmd queueing(+throughout)
Data striping: factor/width (disk number), unit (fixed-size data block); size/depth (size of unit)



DAS: RAID+controller 在 server 下, client 和 server 用 LAN/WAN 传输
NAS: NAS 管 disk,不在 server 下
SAN: SAN 链接不同的 server/storage

	SAN	NAS
Usage Model	Mission-critical data	Serve files
Network	Fibre channel	Ethernet
Data Access	Blocks of data	File level
Cost	Very high	Cost-efficient

CA-07：性能分析和评测

$$S(n) = \frac{S(1)}{1-f+f/n} \qquad \lim_{n\to\infty}\frac{S(n)}{S(1)} = \frac{1}{1-f}$$

$$CPI = \sum \frac{IC_i \times CPI_i}{Instruction\ count}$$

$$CPU\ time = (\sum IC_i \times CPI_i) \times Clock\ cycle\ time$$

- Then the average length in the queuing system is:

$$L = \lambda W$$

\lambda: the average job arrival time W: the average wait time

On The Evaluation of Energy and Power

P_(total) = P_(dyn) * U + P_(idle)

- Dynamic Power
  - C: load capacitance
  - V: supply voltage (typically less than 1.5 V)
  - A: activity factor (a fraction between 0 and 1)
  - f: operating frequency

The CPU power-to-frequency relationship is cubic

$$P = a \times CV^2Af$$

Experimental results show that the power-to-speed relationship is almost linear
-Due to limited voltage/frequency levels
-Not applied to many components in the server

- Frequency typically scales linearly with voltage
  - *i.e.*, $f \propto V$, Thus, the combined V^2f portion of the dynamic power equation has a cubic impact on power dissipation

Sim metric:acc, eval t, dev t, coverage

Fuctional Simulation: Models only the functional characteristics of ISA, No timing issue is considered, Basically a instruction-set emulator, 这种方法也generate instr和addr trace，所以可以作为其他的simulation tools的输入
Trace-Driven Simulation: Takes program instruction and address traces into a detailed microarchitecture timing simulator. Separates functional simulation from the timing simulation, 劣势是需要保存trace file，而且再mis-prediction下的model很不准确
Execution-Driven Simulation: The de-facto simulation approach today. Combines functional with timing simulation. 比trace-driven simu的精度更高
Full system simulation: Trace-driven simulation + execution-driven simulation

	Function Simulation	Trace-Driven Simulation	Execution-Driven Simulation
Development Time	Excellent	Poor	Very Poor
Evaluation Time	Good	Poor	Very Poor
Accuracy	Excellent	Very good	Excellent
Coverage	Poor	Excellent	Excellent

Fast-forwarding: functional sim 建立 arch state, exe-driven sim 认真 sim
Checkpointing:保存 reg/mem 状态
Workload 设计: characterization, pca, cluster 分析
Workload:一个机器上的 user 程序和操作系统 instr 的混合

$$\langle \mathcal{B}, \mathcal{T}, \mathcal{F}, \mathcal{S}_0 \rangle$$

Multi-Program Benchmark Example:

- ***B*** - A set of single-program benchmarks
- ***T*** - The terminating condition for the benchmark
- ***F*** - The selection function with decides which member of ***B*** to run on a particular core when that core becomes idle
- ***S*** - the initial state for ***F***

$$SPECRatio = \frac{Execution\ Time_{reference}}{Execution\ Time_{rated}}$$

- Harmonic mean is meaningful when:
  - The metric is obtained by dividing A by B
  - A is weighed equally among the benchmarks

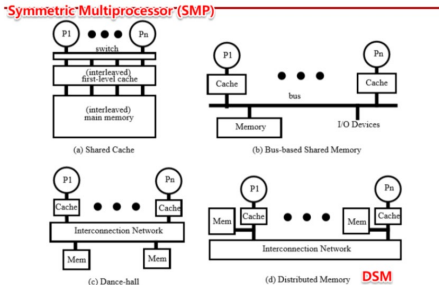
$$\frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} B_i} = \frac{n \cdot A}{\sum_{i=1}^{n} B_i} = \frac{n}{\sum_{i=1}^{n} \frac{B_i}{A}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{A/B_i}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{A_i/B_i}} = HM(A_i/B_i)$$

- Arithmetic mean is meaningful when:
  - The metric is obtained by dividing A by B
  - B is weighed equally among the benchmarks

$$\frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} B_i} = \frac{\sum_{i=1}^{n} A_i}{n \cdot B} = \frac{1}{n}\sum_{i=1}^{n}\frac{A_i}{B} = \frac{1}{n}\sum_{i=1}^{n}\frac{A_i}{B_i} = AM(A_i/B_i)$$

CA-08：多处理器和线程级并行

TL para: 多 chip 多核、多核单 chip
DL para：众核、interconnection net
RL para：多机集群，数据中心，warehouse-scale computer



UMA:多个处理器共享物理内存，访问事件相同，**NUMA:**内存分开
DSM 优点:scalable 的内存带宽核低延迟的 local access;**缺点**：复杂的通信和 node 之间的高延迟
Uniprocessor 的 coherence:DMA

- **Multiprocessors (both SMP and DSM)**
  - Tightly coupled architecture
  - Processors connected via bus or interconnect network
  - Consists of a few processors (2 ~ dozens)
  - A single shared address space
  - Communicate data implicitly via load and store
  - Thread-level parallelism
- **Multicomputers, Clusters (WSCs)**
  - Loosely coupled architecture
  - Individual computers connected on a local area network
  - Consists of large number of nodes
  - Multiple private address spaces
  - Explicitly passing messages among the processors
  - Request/Task level parallelism

Cache Coherence的定义：
1. 之后read的processor，可以读到在它之前写入的内容
2. 如果一个processor写入了一个内容，那么另一个processor总在之后可以读到写入的内容
3. 如果两个processor先后写入同一个位置，所有的processor都可以看到相同的顺序

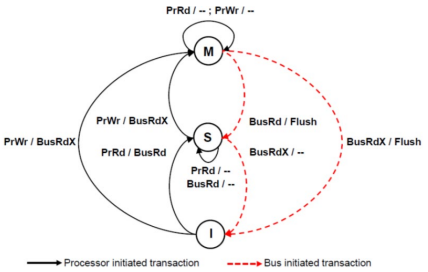
Enforcing Coherence

The key to implementing coherence is tracking the state of any sharing of a data block

Two Classes of Protocols

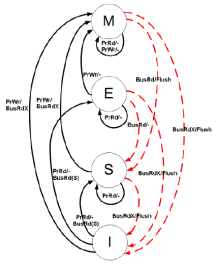
- Snooping Protocol
  - Monitoring all transactions on the interconnect (snooping)
  - Every cache has a copy of the sharing status
  - Normally faster if there is enough bandwidth
- Directory-based Protocol
  - Does not broadcast on the interconnect
  - The sharing status is kept in a single directory
  - Easier to support a large number of processors

Snoop: bus/processor-side controller
Cache 一致:update/invalid-based



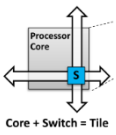
A 4-state (MESI) Write-Back Invalidation protocol

- An extension to MSI
- Its variants are used in many modern processors
- Four States:
  - M: Modified
  - E: Exclusive-Clean
  - S: Shared
  - I: Invalid
- The E State means only this cache has a copy and it has not been modified



CA-09：片上多处理器和多核系统

Fine-grained mt: 每个 clock 都 switch; **coarse-grained mt:** 长 latency stalls 触发; **SMT：**单 cycle 内有来自多 thread 的 instr issue.
多核 cpu: share-memory，MIMD
SMT: 共享 cache/TLB; **CMP:**都独立
heterogeneous/asymmetric CMP: 特性分配 re，不同 workload 需求提供更好的高效的 coverage; **single ISA heterogeneous CMP：**异构 core，同样 ISA; **heterogeneous-ISA CMP：**更灵活，需要：1. process scheduling, 2. page table mappings change 3. binary translation; 4. state transformation
many-core：核更多，并行高，单线程性能较差；**core + switch = tile**



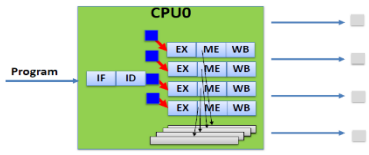
CA-10: 数据级并行化和 GPGPU

vector instr的好处： 1. 更少的instr fetch带宽 2. 更少的用于data hazard checking的硬件 3. 更稳定的mem access延迟
vector processor类型： 1. mem-mem vector processor（所有的vector op都是mem2mem）2. vector-reg processor（vector相当于load-store架构）
vector processor包括vector units (fixed-size bank holding a single vector, determines the maximum vector length (MVL)，有8-32个vector regs)和scalar units
vector执行时间取决于： 1. operand vector长度 2. structure hazard 3. data dependency

$$T_{vector} = T_{scalar} + (\text{vector length} / \text{lane number}) - 1$$

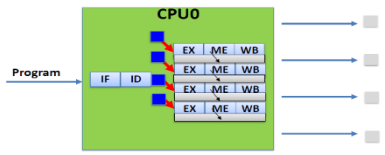
Chaining: the vector version of register bypassing
Convoy: Vector instructions that may execute together。convoy中没有结构hazard和数据hazard（或者已经被chaining解决了）
Vector Length Register (VLR): The value is no greater than the maximum vector length (MVL)
对于超过最大长度的vector: The first loop do short segment (n mod MVL);All the subsequent segments use VL = MVL
Vector–mask control: Based on a Boolean vector (vector mask register). Instruction operates only on vector elements whose corresponding entries in the mask register are 1

SIMD/Vector Approach: 1 heavy thread + data parallel ops
(single PC, single register file)



Split identical, independent task over multiple execution units

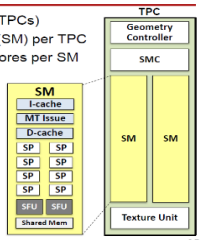
SIMT Approach:
multiple threads + scalar ops (single PC, multiple register file)



Split identical, independent task over multiple lockstep threads

Case Study: Tesla GPU Architecture

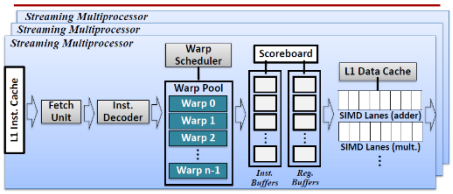
- 8 texture/processor clusters (TPCs)
- 2 streaming multiprocessors (SM) per TPC
- 8 streaming processor (SP) cores per SM
- SM is a unified graphics and computing engine
  - SM executes thread blocks
- Streaming processor (SP)
  - Scalar ALU for a single CUDA thread
  - SP core is a SIMD lane



35

CUDA thread block: an array of cooperative threads,每个都有 unique ID; **grid:** 执行同一个核上的 an array of thread blocks; SM 执行 a pool of threads **(warp).** warp 的 thread 从同地址开始，相同 PC，同 SM 调度。
调度：1. 线程以 warp 为粒度进行；2.scheduler 从 pool 中选一个 warp 执行 3. 尽量 SIMD lanes 都被占用
Warp divergence: re-covergence stack merge
动态 warp 构成：从 pool 中选择 pc 相同的组成新的 warp，这样就不会有上面的顺序执行的问题
GPU 最大的并行数常被寄存器文件（large SRAM，更大可以容纳更多线程，做更快速的线程切换，更充分利用内存带宽）的容量限制（因为太大能耗高 占用面积大），寄存器文件通常比 L1 和 L2 cache 还大

Warp Scheduling (Cont'd)



- Potential factors that can delay a warp's execution
  - Scheduling policies
  - Instruction/Data cache miss
  - Structural/Control/Data hazard
  - Synchronization primitives

更大的 reg file：可以容纳更多的 active threads，高效进行更多的 context switch，GPU 更充分地利用内存带宽；但面积更大，更耗电
SIMT 和 SIMD 的区别：SIMT（多线程+scalar ops（单 pc,多 regfile），SIMD（1 个重线程+data parallel（单 pc，单寄存器文件））

CA-11：片上互联网络

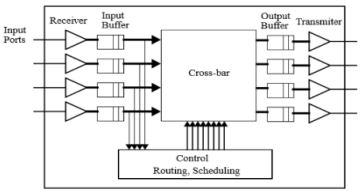
network 可以被描述：Topology (physical interconnection); **Routing Algorithm** (data to which route); **Switching Strategy** (how data in a message traverses its route); **Flow control mechanism** (when the message, or portions of it, move)*
Minimum unit transferred across a link is called a flow control unit: flit

Switch Network Topologies

- View switched network as a graph
  - Vertices: nodes or switches
  - Edges: communication paths
- Describes the structure of the network graph
  - e.g. Direct network have a host node connected to each switch
    - a.k.a. distributed switch
  - e.g. Indirect network have hosts connected only to certain switches
    - a.k.a. centralized switch
- Regular vs Irregular
  - Regular network are widely used graphs such as grid and tree, etc.

direct network：每个 switch 都和一个 node 相连；indirect network：node 与部分 switch 相连
Switch 策略：circuit switching (has better bandwidth) & packet switch (has longer setup time); **Latency (lower bound)**=sending overhead+routing t+arbitration t+switching t+packet size/bandwidth+receiving overhead

- Network switches implement the routing, arbitration, and switching functions of switched-media networks



internal structure of a network switch

Crossbar switch: 所有 node 互联，O(N)带宽,互联 cost O(N^2)
Array:diameter=N-1,average dis=2N/3, bisection width=1
Ring:d=N-1,AD=N/2,BW=1
2D Mesh: nodes 之间多 route，node 间 latency 不尽相同，cost O(N)
2D Torus: Slightly lower latency while higher cost
Tree: routing distance 按照对数增长
Multistage 互联网络: Log(N)的 stage，N/2 的 switching units

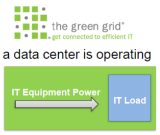
CA-12：数据中心即计算机

DC 功能: processing (request 级别的并行), storage(大规模、高可靠), communication (高 BBW)
数据中心的关键部分：ICT equipment, power 系统，cooling 系统，其他的照明、安全等系统

Data Center Operation Efficiency

- Power Usage effectiveness (PUE)
  - Provides a top-level view of how efficient a data center is operating

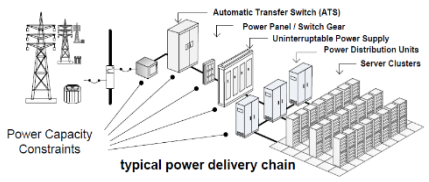
$$PUE = \frac{Total\ Facility\ Power}{IT\ Equipment\ Power}$$



- More metrics from "The Green Grid" …
  - WUE: water usage effectiveness (L/kWh).
  - CUE: carbon usage effectiveness (kgCO$_2$eq/kWh)

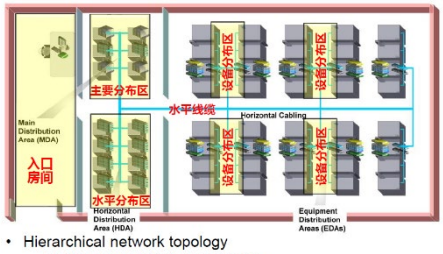
$$WUE = \frac{Annual\ water\ usage}{IT\ Equipment\ Energy} \qquad CUE = \frac{Total\ Carbon\ Emissions}{IT\ Equipment\ Energy}$$

Datacenter Infrastructure – Power System



- ATS: basically a fast, mechanical switch
- STS: basically a superfast, electronic switch
- UPS: basically a battery with control interfaces
- PDU: a power converter and distribution unit

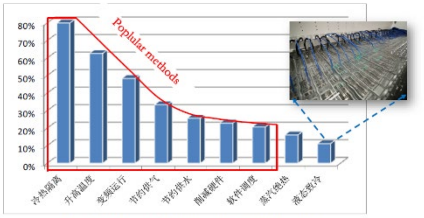
Datacenter Infrastructure – ICT System



- Hierarchical network topology
  - Entrance facility + MDA + HAD + EDA

Over-provisioning: server nameplate power<data center power capacity, nameplate power + worst-case numbers for a safety margin
Over-subscribing: can't run all
data center 不同的 tier standard： tier4 要求最高的可用度，备用能源，2N+1 的备份；tier3 要求低一些，备用能源，N+1 备份；tier2 要求更低，无备用能源，但对关键组件要备份；tier1 只有可用度要求
WSC，设计要考虑性能，能效，可用性，网络，service level agreement（service 如何提供），service level objectives（评估 service 的性能）

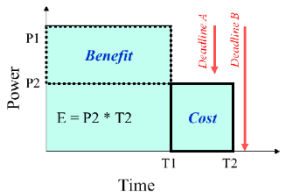
Discussion: Cooling Approach



Based on the data from the Uptime Institute, 2014

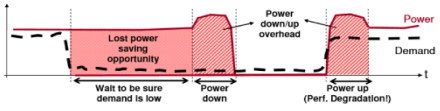
CA-13:面向功效和节能的设计

Frequency Scaling



- Benefit vs. Cost
  - power demand ∝ overall performance

S-State Transition Latency

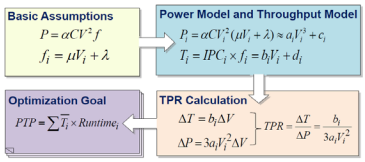


- Resume times from S3 can be an order of magnitude better than those with S4 or S5
- The power-off times for S3 are significantly better than for S4 and S5

Advanced Configuration and Power Interface (ACPI): 开放的标准，介于 Drivers 和 OS power management 之间，操作系统可以通过它进行 power 的管理
Global System States/ G-state: high-level description of the platform states:1. **G0**-working. 2. **G1**-sleeping(no computation is performed) 3. **G3**-soft off (powered down but can be restarted by interrupts). 4. **G4** - hard off
Sleeping States / S-state: set in the BIOS and configed by sys: 1.**G0-S0:** normal operation 2.**G1: S1:** processor clock is off; **S2:** processor is off; **S3:** suspend to RAM; S4: suspend to disk 2. **G2-S5:** halt state
Processor Power States / C-States: G0 and S0 define a working platform state, at which a range of C-states are defined to save power: 1. **C0:** normal operating state 2.**C1**: auto halt, the clock is gated 3. **C3:** sleep, maintain arch state but flush data to shared LLC, shut down clock generator 4. **C6:** arch states saved to dedicated SRAM, core voltage reduced to 0
Processor Perf States / P-state: different operational modes
1.**不同 clock freq,** P0 最高
2. **sub states of C0**, define **dynamic voltage and frequency scaling (DVFS)** steps ; 3.**Switching latencie**s are negligible for most purposes
Thermal design power (TDP): The max sustained power used for design of the processor thermal solution

Calculate Throughput-Power Ratio

- Allocate precious power to high productive cores
  - Predict the return on investment (ROI)

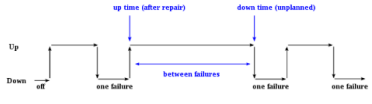


CA-14：面向可靠性和可用性设计

fault: physical flaw; **error:** a fault can manifest itself as an error. Or, fault can be masked and not manifested as error; An error can be masked or it can result in a failure, **failure is user-visible incorrect behavior**

MTTF, MTBF, MTTR

- Mean Time to Failure (MTTF)
- Mean Time Between Failures (MTBF)
- Mean Time to Repair (MTTR)



$$Availability = \frac{MTTF}{MTBF} = \frac{MTTF}{MTTF+MTTR}$$

Availability and Reliability

- The availability of a system at time t
  - is the probability that the system is operating correctly at time t.
  - units for availability are often the "number of nines"
- The reliability of a system at time t
  - is the probability that the system has been operating correctly from time zero until time t.

Is it possible for a low-reliability system to have high availability?

Architecturally correct execution (ACE) bit: correctness required for the program <-> Un-ACE bits
Architectural vulnerability factor (AVF): captures the probability that a fault in a structure will manifest as an error in the program output

- The AVF of a hardware structure H containing B bits over a period of N cycles can be expressed as:

$$AVF_{structure} = \frac{1}{N}\sum_{i=0}^{N}\left(\frac{ACE\ bits\ in\ H\ at\ cycle\ i}{B}\right)$$

- Design considerations?
  - Appropriate microarchitecture + Application characteristics

知识点

1.L2/3cache 消耗大量功率，作为静态功率;
2.增大 cache 虽然降低 miss 率，但是 hit 时间变大，功率也变大;
3.DRAM 读取之后要写回，所以周期时间长于访问时间;
4. HPC: throughput matters;
5.Internet Data Center(IDC):latency matters;
6.Toma 的核心是通过 reg renaming 把数据放到 RS 中, CDB 上有 value 和 tag，RS 的数据 ready 后丢到 bus，下一个 cycle 被读到;
7.Superscalar 基于硬件,VLIW 基于编译;
8. Branch Target Buffer(BTB): Branch Instr Addr(BIA)+Branch Target Addr(BTA);
9. NAND flash cell:加电压使得浮栅上吸附电子，因为没有放电回路，所以去掉电压的时候电子不会跑掉，所以不用经常刷新.
10. 多线程的必要性: ILP 没法解决 mem stall 的问题;
11.Single-Chip Cloud Computer(SCC):一个 chip 多个 core，没有 cache coherence，使用 message passing 来处理.
12. Swich/router 是片上网络的主要成本，他连的 input channel 数量叫做 switch degree 或 radix.
13.一个 route/path 上的 switch 数目叫做 hop count.
14. channel 带宽=信号频率*link width
15. per core load adaptation policy: 三种方法：IC(一直调单个，知道到达最佳水平)，RR(用 round-robin 的方式挨个调)，Opt(基于 TPR 来选择)
16. VLIW 不支持 out-of-order
17. DDRx 是 double data rate，在时钟上升和下降沿都读取数据
18. VLIW 使用 reduced ISA