How to Write Fast Code

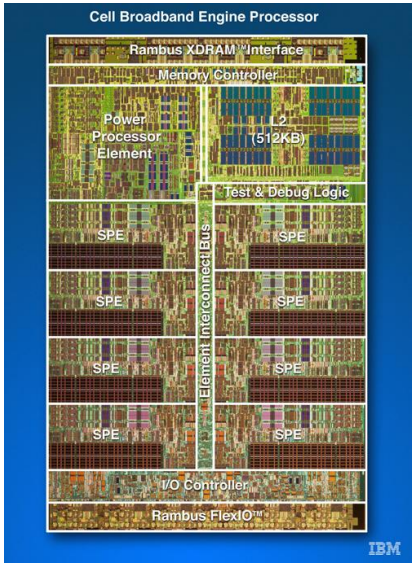
18-645, spring 2008
20th Lecture, Mar. 31st

Instructor: Markus Püschel

TAs: Srinivas Chellappa (Vas) and Frédéric de Mesmay (Fred)

Introduction

- **Parallelism: definition**
  - Carrying instructions out simultaneously
  - (Think multi-core)
- **Why parallelism?**
  - Duh (if you can make it faster, why not?)
  - More importantly: not many other options now
- **Why not parallelism?**
  - Difficult to build hardware
  - Difficult to program
    - Not intuitive to humans
  - No clear hardware or software model
    - Application specific



Lecture overview

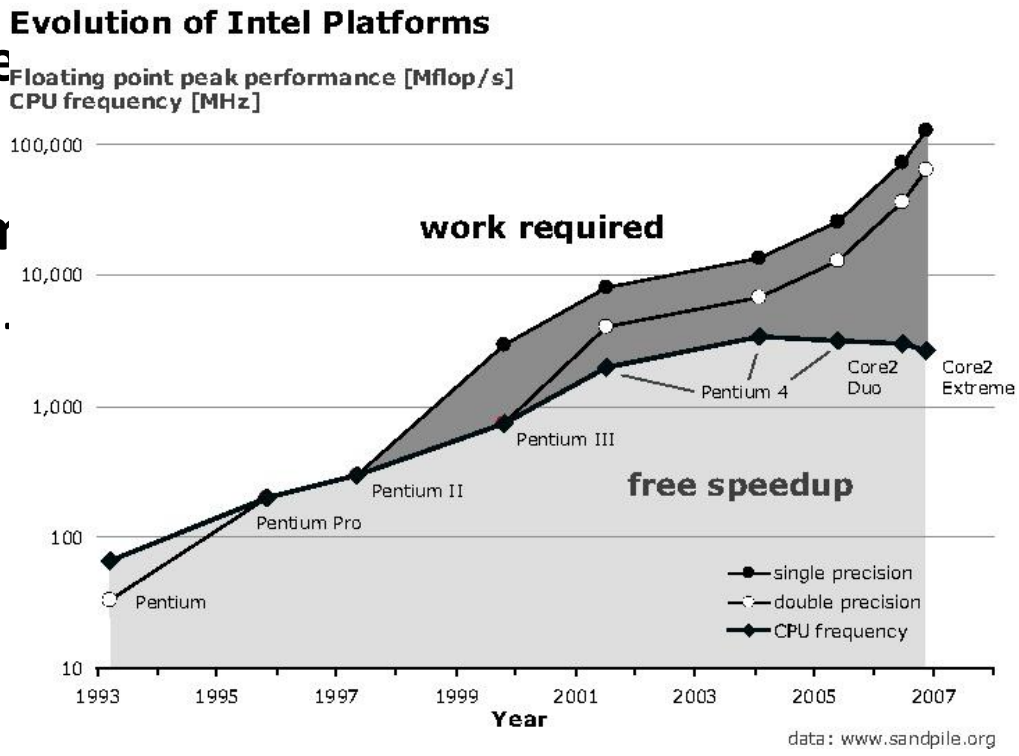
- Introduction / Background
- **Parallel platforms / hardware paradigms**
- **Parallel software paradigms**
- **Mapping algorithms to parallel platforms**

- **SIMD homework feedback**

- **Administrative stuff**
  - Feedback form
  - Schedule project meetings

Parallelism: motivation

- Idea has been around for a while. Why now?
- Frequency scaling no longe an option
- We're forced to parallelism
- Think Intel/AMD multi cores



Future (now) will be driven by parallelism

Parallel Programming

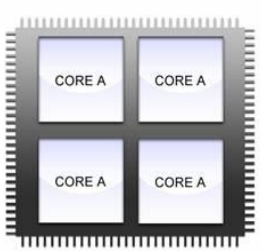
Problem
+
parallel algorithms

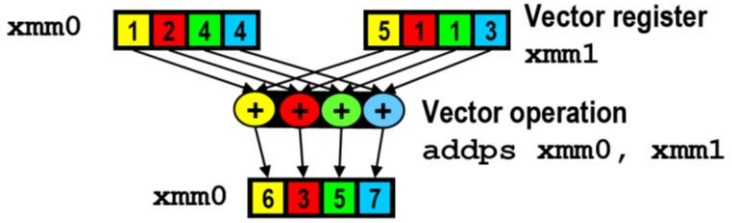
Paradigms:
Data parallel
Task parallel
Streaming
EPIC

Archs:
SMP
Cluster
Multi-core
SIMD
FPGA

Parallel Architectures

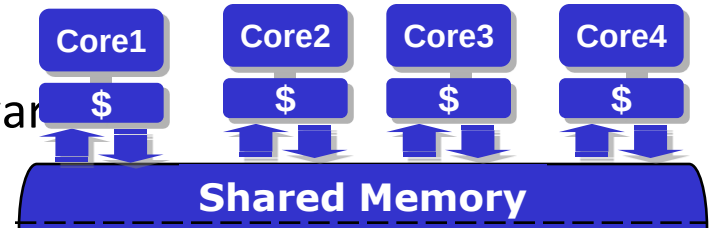
- **Parallelism: one size doesn't fit all**
  - SMP (symmetric multiprocessing): smaller CPUs
    - Multi-core
    - Multi-CPU, Hybrids, FPGAs etc.
  - Distributed/NUMA: larger systems
    - Cluster (including grid computing)
    - Supercomputers
  - Other types:
    - SIMD: fine-grained parallelism
    - Stream computing: pipelined parallelism
    - Instruction level parallelism (VLIW/EPIC/superscalar)



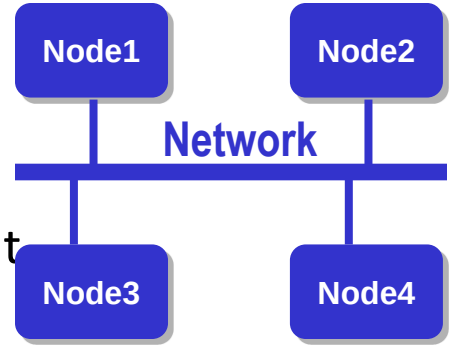


Parallel Architectures - Constraints

- **SMP:**
  - Easy to program
  - System complexity is pushed to hardwar design
  - Coherency protocols
  - Scalability



- **Distributed computing:**
  - Can never match SMP's node-to-node data transfer capability
  - Hardware architecture scales very well
  - Commodity hardware
  - Grid computing ("virtual supercomputers" built out of untrusted, unreliable nodes)
  - Programmer burden is higher



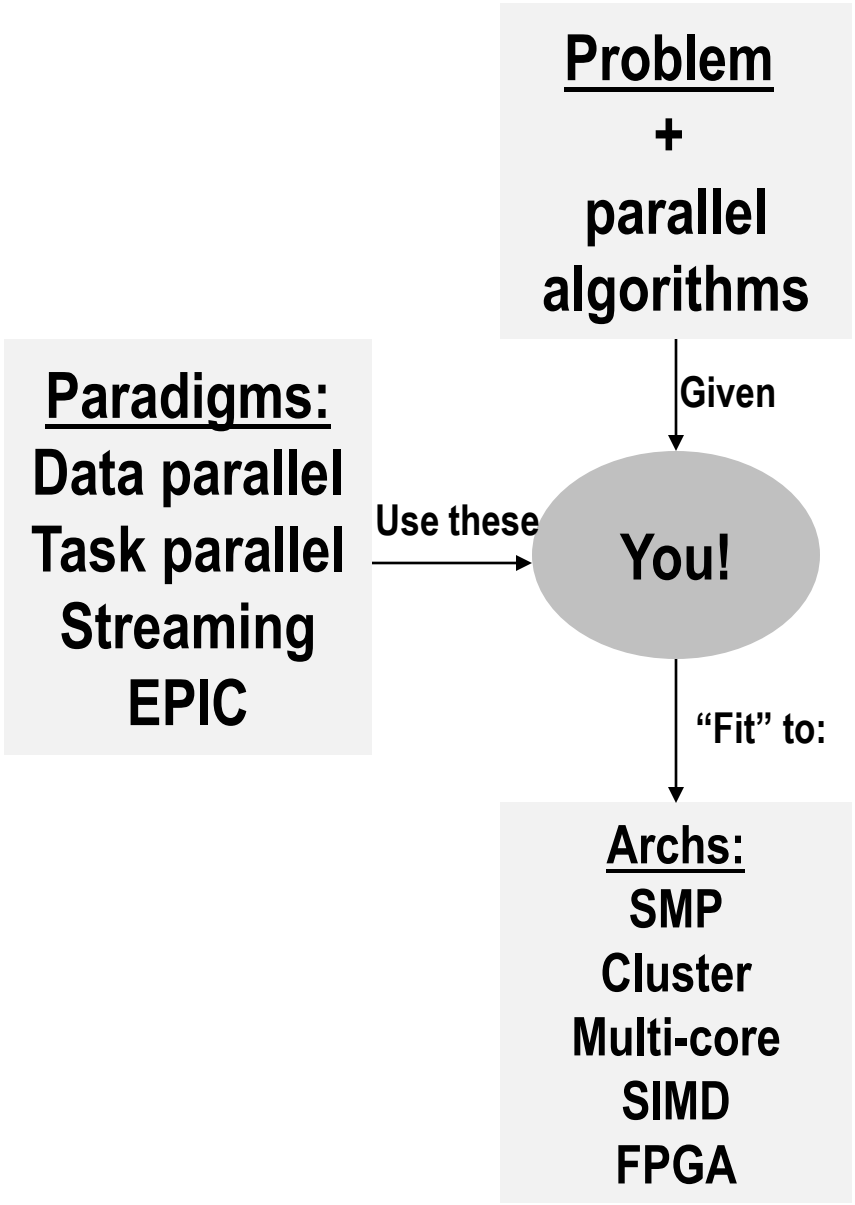
Solving a Problem in Parallel

- **So far, we've looked at parallel architectures**
  - To solve problems in parallel, we must identify and extract parallelism out of problems
- **Question 1: does it make sense to parallelize?**
  - Amdahl's law: predict theoretical maximum speedup
- **Question 2: what is the target platform/arch?**
  - Does the problem "fit" the architecture well?

Parallel Paradigms

- **Task parallelism**
  - Distribute execution processes across computing nodes
- **Data parallelism**
  - Distribute chunks of data across computing nodes
- **Many real-world problems lie on the continuum**
- **Other models: streaming**
  - Sequential tasks performed on the same data
  - Pipelined parallelism
  - Eg: DVD decoding

Mapping problems to machines



Challenge: To map the problem to the architecture using abstractions

- **Some problems match some architectures better**

Case studies

“Heavy metal”

- **Clusters**
- **BlueGene/P (Supercomputer)**

Desktop

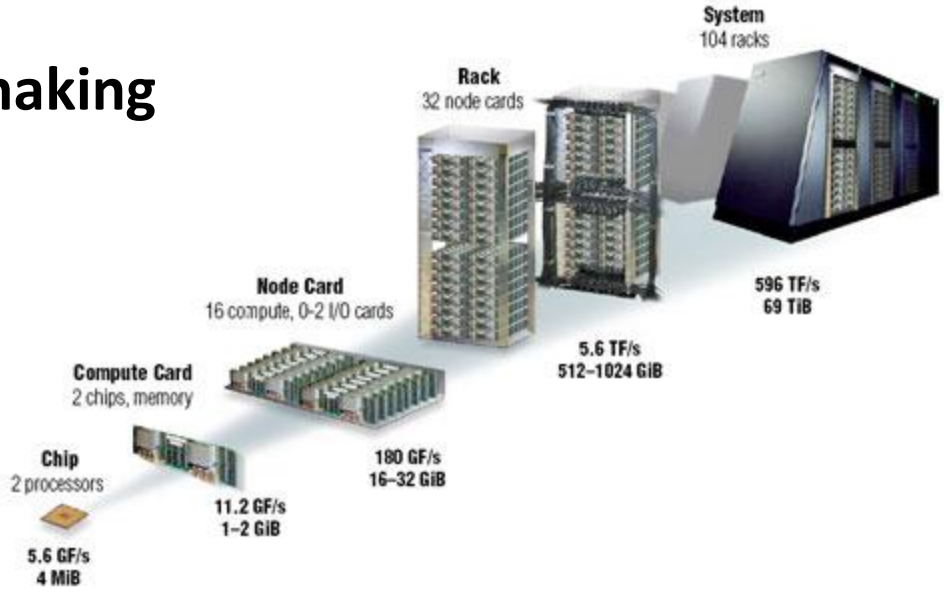
- **Future Intel (Nehalem)**
- **GPU (Graphics Processing Unit)**
- **Hybrid/accelerated (CPU+GPU+FPGA)**
- **Cell Broadband Engine**

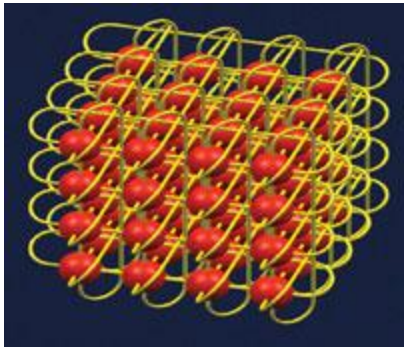
Cluster Computing

- **Designed for:**
  - Supercomputing applications
  - High availability (redundant nodes)
  - Load balancing
- **Architecture**
  - Multiple nodes connected by fast local area networks
  - Can use COTS nodes
- **Peak performance**
  - 100 Teraflop/s (Cluster Platform 3000. 14,000 Intel Xeons!)
- **How to program**
  - Specific to each cluster
- **Notes**
  - Power/heat concerns
  - Most supercomputers, grids

BlueGene/P: Cluster Supercomputer

- **IBM's supercomputer in the making**
- **Architecture**
  - PowerPC based
  - High-speed 3D toroidal network
  - Dual processors per node
- **Peak performance**
  - Designed for upto 1 Petaflop/s
  - 32,000 cpus for 111 Teraflop/s
- **Features**
  - Scalable in increments of 1024 nodes

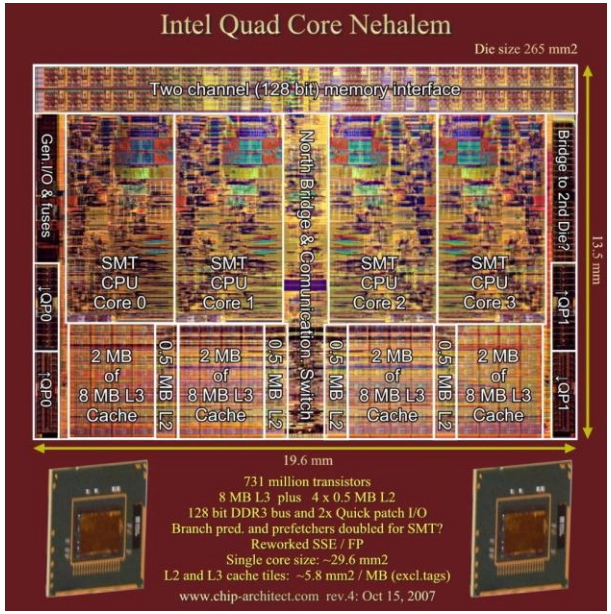




Future Intel: Nehalem

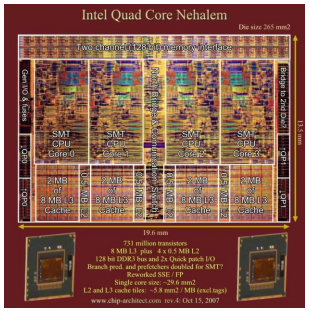
- **Designed for:**
  - Desktop/mainstream computing
  - Servers
  - Successor to Intel's "Core" architecture
  - (2008-09)

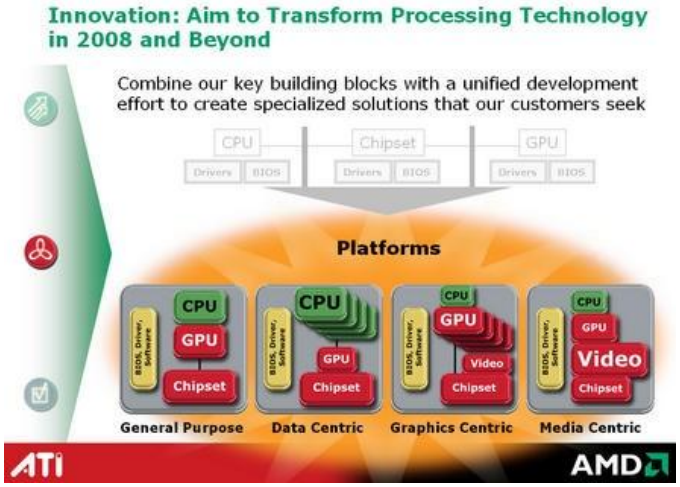
- **Microarchitecture**
  - Your standard parallel CPU of the future
  - 1-8 cores native (single die)
  - Out-of-order
  - 8MB on-chip L3 cache
  - Hyperthreading



Intel Nehalem

- **Peak performance**
  - ~ 100 Gflop/s (excluding GPU)
- **Features**
  - 45nm
  - SSE4
  - Intergrated Northbridge (combat memory wall)
  - GPU on chip (off-die)
- **Programming:**
  - Not very different from current CPUs
- **Notes**
  - AMD + ATI = something similar
  - What next?





GPU (Graphics Processing Unit)

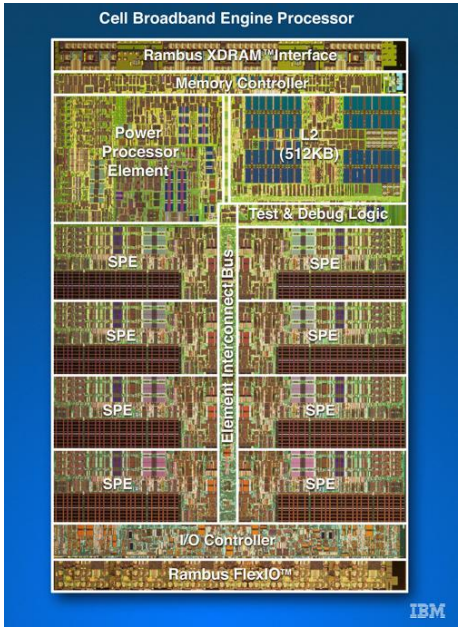
- **Designed for:**
  - Originally/largely: graphics rendering
  - Also: as a massive floating-point compute unit. General Purpose GPU (GPGPU)
- **Microarchitecture**
  - Multiple “Shader units”: vertex/geometry/pixel
  - Typically a PCI/PCIe card
  - Can be on-chip
- **Peak performance**
  - 0.5 – 1 TeraFlop/s

GPU

- **Programming**
  - Programmable shader units designed for matrix/vector operations
  - Stream processing
  - Grid computing (ATI / Folding@Home)
- **Notes**
  - Numeric computation only
  - Shipping data from and to destination (main memory)

Cell Broadband Engine ("Cell")

- **Designed for:**
  - Heavy desktop numeric computing
  - General purpose
  - Gaming / heavy multimedia vector processing
  - Supercomputer node
  - Bridge gap: desktop and GPU
- **Microarchitecture**
  - Hybrid multi-core CPU
  - 1 PPE (general-purpose core)
  - 8 SPEs (specialized vector cores)
  - Core Interconnect
- **Peak performance**
  - ~ 230 Gflop/s (3.2GHz, using all 9 cores)
  - 204.8 Gflop/s (3.2GHz, using just the SPEs)



Cell BE

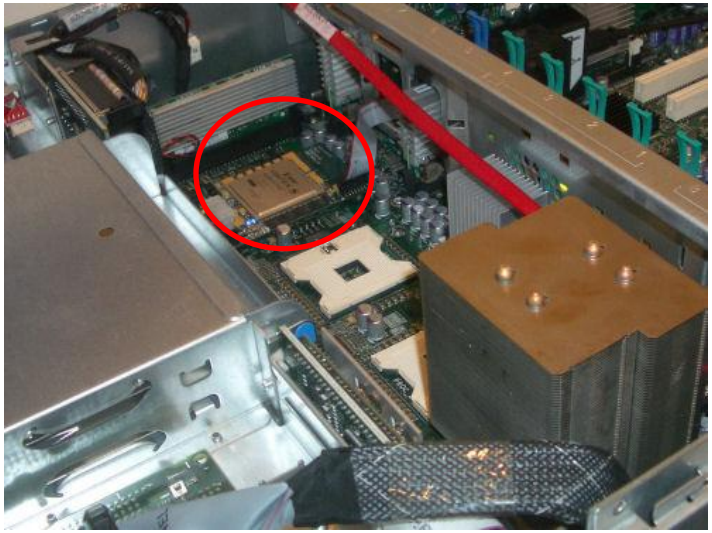
- **PPE (Power Processor Element) x1**
  - Power architecture based
  - Out-of-order , dual-threaded core
  - For general purpose computing (OS)
  - 32+32KB L1, 512KB L2
  - AltiVec vector unit
- **Synergistic Processing Elements x8**
  - For numeric computing
  - Dual-pipelined
  - 256kb “Local Stores” – 1 per SPE
  - Interconnect: EIB – (SPE Interconnect Bus)
- **Explicit DMA instead of caching!**
  - Programmer has to manually ship around data (overlap with comp.)
  - Advantage: fine-grained control over data, deterministic run times

Cell

- **How to program**
  - SMP or DMP model?
  - Parallel model: task parallel / data parallel / stream
  - Pushes some complexity to programmer
  - Libraries can handle DMA. (do not use for high-performance computing)
- **Available as:**
  - Already in use in the PlayStation 3
  - IBM BladeCenter
  - Add-on accelerator cards

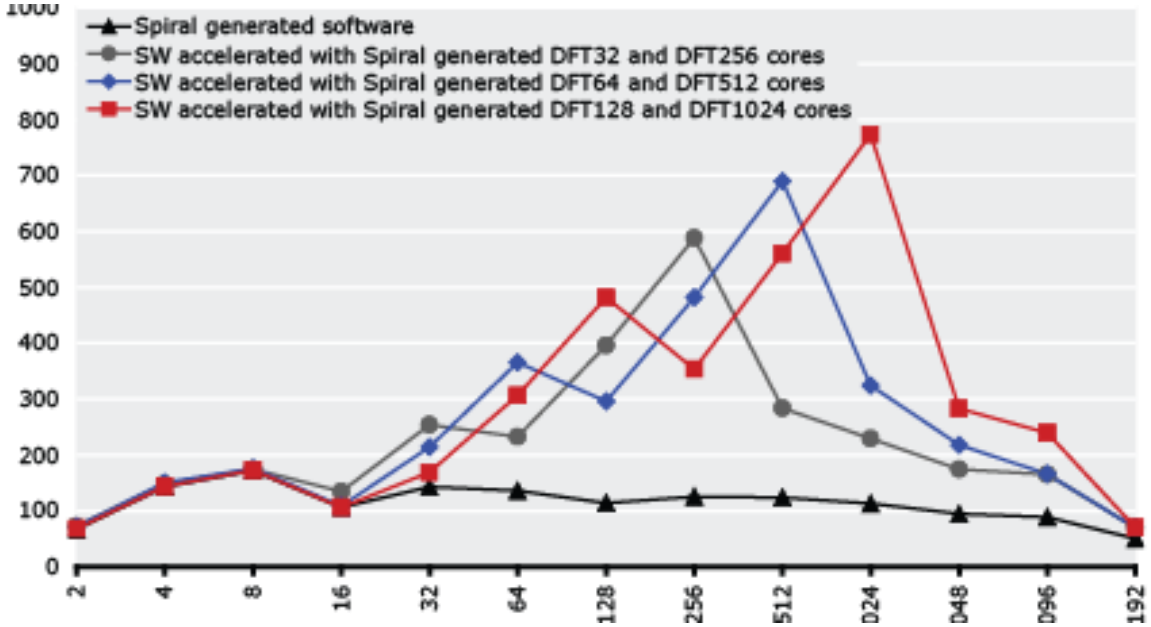
Hybrid/Accelerated

- **Designed for:**
  - Customized applications/supercomputing node
  - Main use: numeric/specialized computing
- **Microarchitecture**
  - CPU, FPGA on the same motherboard
  - High speed data interconnect
- **Peak performance**
  - CPU's + FPGA's
- **Features**
  - Benefits of FPGAs

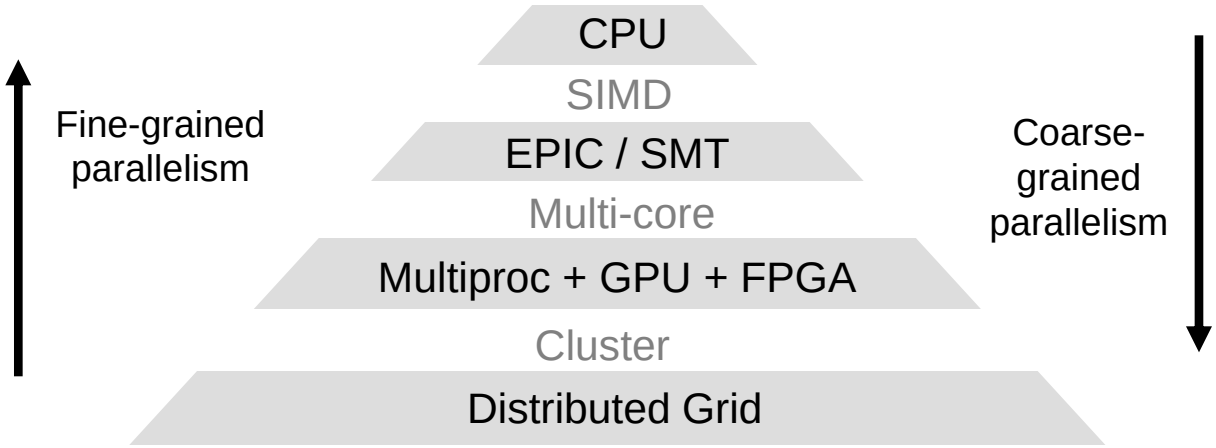


Hybrid / Accelerated

- **How to program**
  - Specialized: depends on setup, program
  - Need to generate FPGA design (non-trivial)
  - Then, generate software
- **Spiral generated FPGAs:**



Parallel Hierarchy



- **Parallelism works on multiple levels (nested?)**
- **Numeric fast code must explicitly address many/most levels**
- **A major difference between levels: cost of shipping data between nodes**

Parallel Mathematical Constructs

- Blackboard

Conclusion

- **Parallelism is the future**
- **Extracting/using parallelism: ongoing challenge**
  - Hardware is ahead of software:
  - Producing parallel hardware currently easier than producing parallelized software
- **“Our industry has bet its future on parallelism(!)”**

  - David Patterson, UC Berkeley
- **Challenge: how to “map” a given problem to a parallel architecture/platform**

Meetings Apr 7 (next Monday)

Markus	
11 – 11:45	8
11:45 – 12:30	7
1:30 – 2:15	9
2:15 - 3	16
3 – 3:45	14
	12
4:30 – 5:15	6
5:15 – 6	13

Fred	
4:30 – 5:15	1
5:15 – 6	2
6 – 6:45	3

Franz	
1 – 1:45	17
2 – 2:45	11
4:30 – 5:15	5

Vas	
3:45 – 4:30	4
4:30 – 5:15	10
5:15 - 6	15