Parallel Computing: Opportunities and Challenges

Victor Lee

Parallel Computing Lab (PCL), Intel

Who We Are: Parallel Computing Lab

- Parallel Computing -- Research to Realization
  - Worldwide leadership in throughput/parallel computing, industry role-model for application-driven architecture research, ensuring Intel leadership for this application segment
  - Dual Charter:
    - Application-driven architecture research and multicore/manycore product-intercept opportunities
- Workload focus:
  - Multimodal real-time physical simulation, Behavioral simulation, Interventional medical imaging, Large-scale optimization (FSI), Massive data computing, non-numeric computing
- Industry and academic co-travelers
  - Mayo, HPI, CERN, Stanford (Prof. Fedkiw), UNC (Prof. Manocha), Columbia (Prof. Broadie)
- Architectural focus:
  - "Feeding the beast" (memory) challenge, unstructured accesses, domain-specific support, massively threaded machines
- Recent accomplishments:
  - First TFlop SGEMM and highest performing SparseMVM on KNF silicon demo'ed at SC'09
  - Fastest LU/Linpack demo on KNF at ISC'10
  - Fastest search, sort, and relational join – Best Paper Award for Tree Search at SIGMOD 2010

Motivations

- Exponential growth of digital devices
  - Explosion of the amount of digital data

Motivations

- Exponential growth of digital devices
  - Explosion of the amount of digital data
- Popularity of World-Wide-Web
  - Changing the demographics of computer users

Motivations

- Exponential growth of digital devices
  - Explosion of the amount of digital data
- Popularity of World-Wide-Web
  - Changing the demographics of computer users
- Limited frequency scaling for single core
  - Performance improvement via increasing core count

What these lead to

Massive data needs massive computing to process

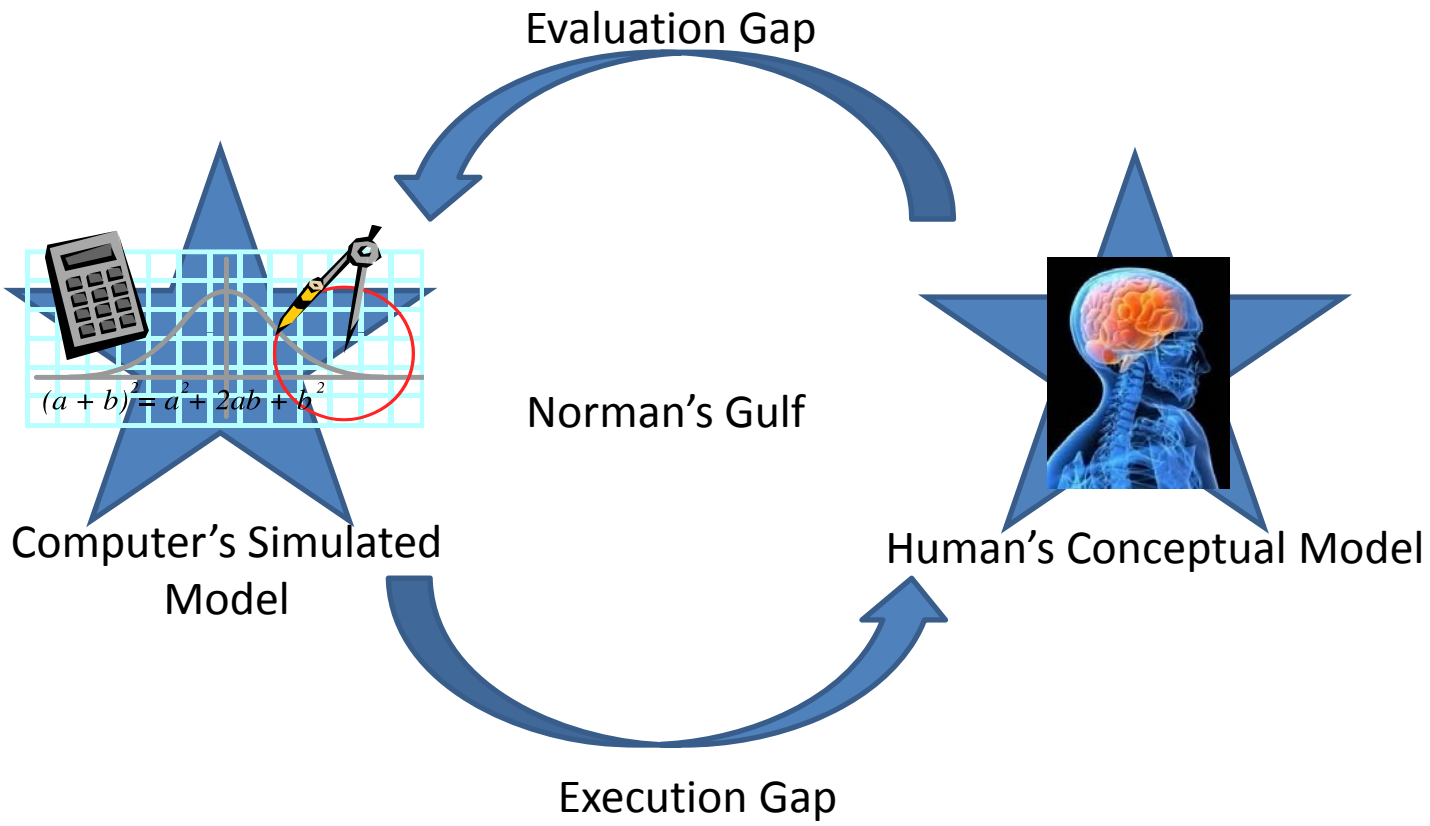
Birth of multi-/many-core architecture

Parallel computing

The Opportunities

What parallel computing
can do for us?

Semantic Barrier



- Lower semantic barrier => Make computers solve problems the human way => Makes it easier for human to use computers

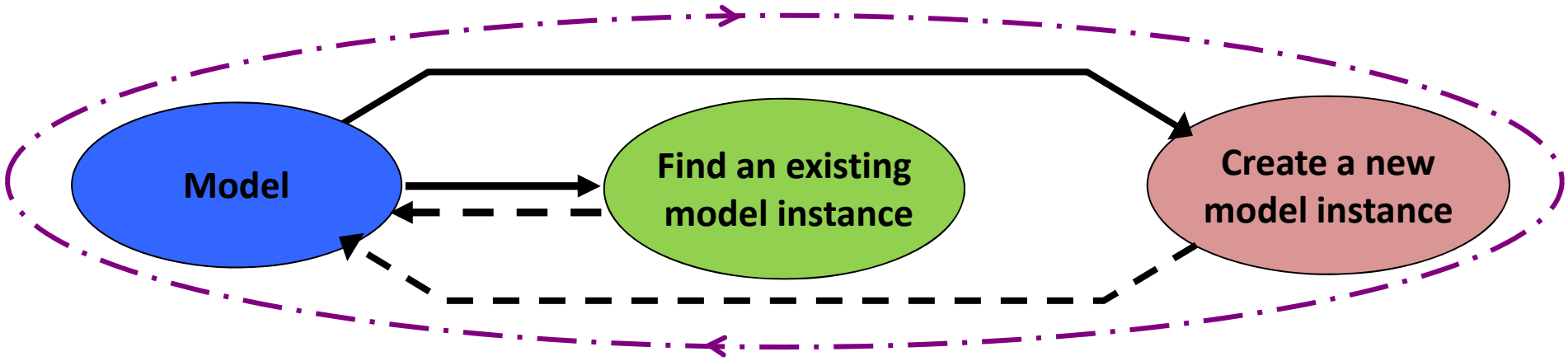
Model Driven Analytics

- Data-driven models are now tractable and usable
  - We are not limited to analytical models any more
  - No need to rely on *heuristics* alone for unknown models
  - Massive data offers new algorithmic opportunities
    - Many traditional compute problems worth revisiting
- Web connectivity significantly speeds up model-training
- Real-time connectivity enables continuous model refinement
  - Poor model is an acceptable starting point
  - Classification accuracy improves over time

Interactive RMS Loop

Recognition | Mining | Synthesis

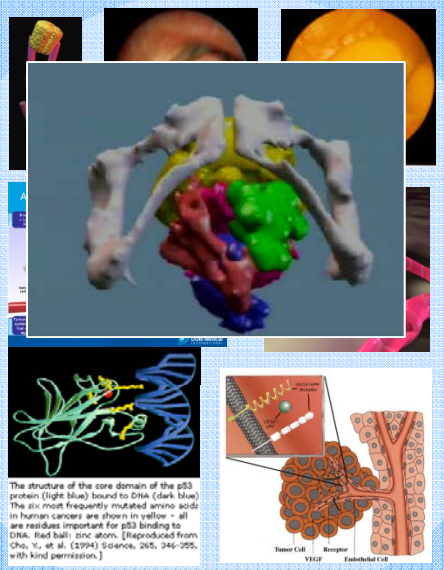
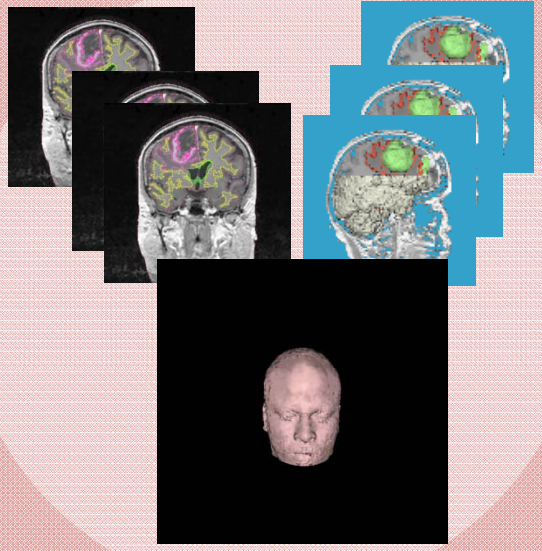
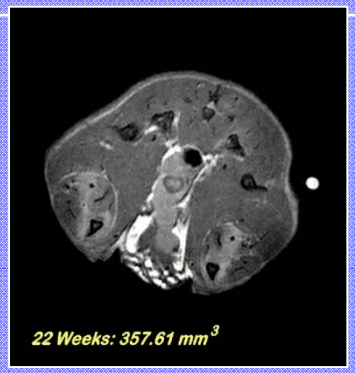
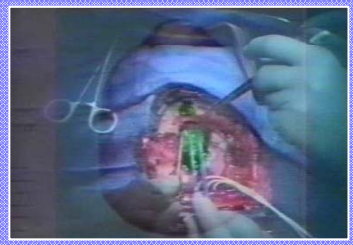
What is …? | Is it …? | What if …?



Most RMS apps are about enabling interactive (real-time) RMS Loop (iRMS)

Victor.W.Lee@intel.com

RMS Example: Future Medicine

Recognition	Mining	Synthesis
		 
What is a tumor?	**Is there a tumor here?**	**What if the tumor progresses?**

Images courtesy: http://splweb.bwh.harvard.edu:8000/pages/images_movies.html

It is all about dealing efficiently with complex multimodal datasets

RMS Example: Future Entertainment

Recognition	Mining	Synthesis
		
Who are Shrek, Fiona, and Prince Charming? What is the story-net?	**When does Shrek first meet Fiona's parents?**	**What if Shrek were to reach late? What if Fiona didn't believe Prince Charming?**

Tomorrow's interactions and collaborations: Interactive story-nets, multi-party real-time collaboration in movies, games and strategy simulations

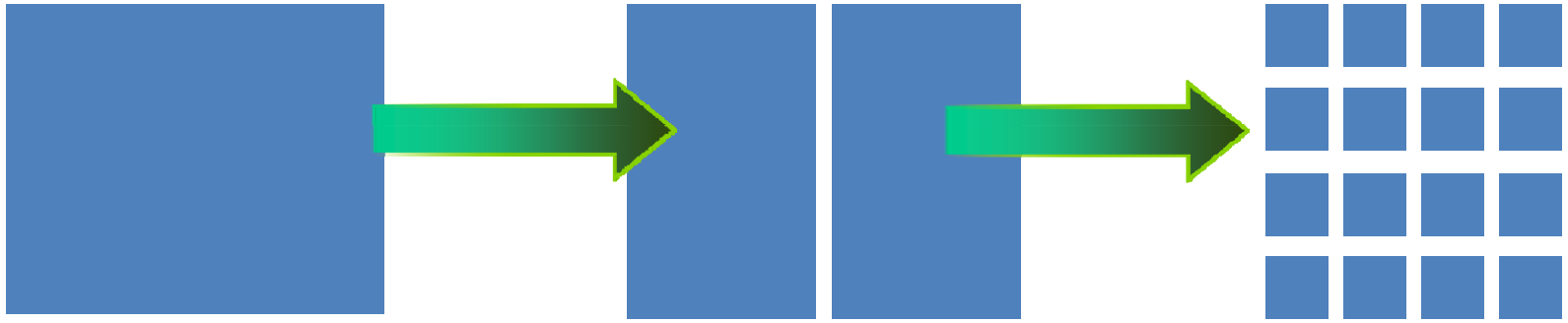
Opportunities (Summary)

- More data
  - Model-driven analytics
- More computing
  - Interactive RMS loops
- Lower computing barrier
  - Computer easier to use for the mass

The Challenges

Why Parallel Computing is hard?

Multi-Core / Many-Core Era



Multi-core / many-core provides more compute capability with the same area / power

Architecture Trends

- Rapidly Increasing Compute
  - ***Core* Scaling** (Nhm (4-cores) → Wsm (6-cores) →…→Intel Knights Ferry (32-cores) …)
  - ***Data-Level Parallelism (SIMD)* Scaling**
    - SSE (128-bits) → AVX (256-bits) →…→LRBNI(512-bits) → …
- Increasing Memory Bandwidth, But…
  - ***Not keeping pace with compute increase.***
  - *Used to be **1-byte/flop***
  - *Current: Wsm (**0.21 bytes/flop**); AMD Magny Cours: **(0.20 bytes/flop**); NVIDIA GTX 480 (**0.13 bytes/flop**)*
  - *Future: **0.05 bytes/flop (GPUs, 2017)**(ref: Bill Dally, SC'09)*

One clear trend: More cores in processors

Architecture Trend

	Intel Core i7 990X (a.k.a. Westmere)	Intel KNF
Sockets	2	1
Cores/socket	6	32
Core Frequency (GHz)	3.3	1.2
SIMD Width	4	16
Peak Compute	316 GFLOPS	1,228 GFLOPS

Increase in compute comes from more cores and wider SIMD

Implication: Need to start programming for Parallel Architecture

Parallel Programming

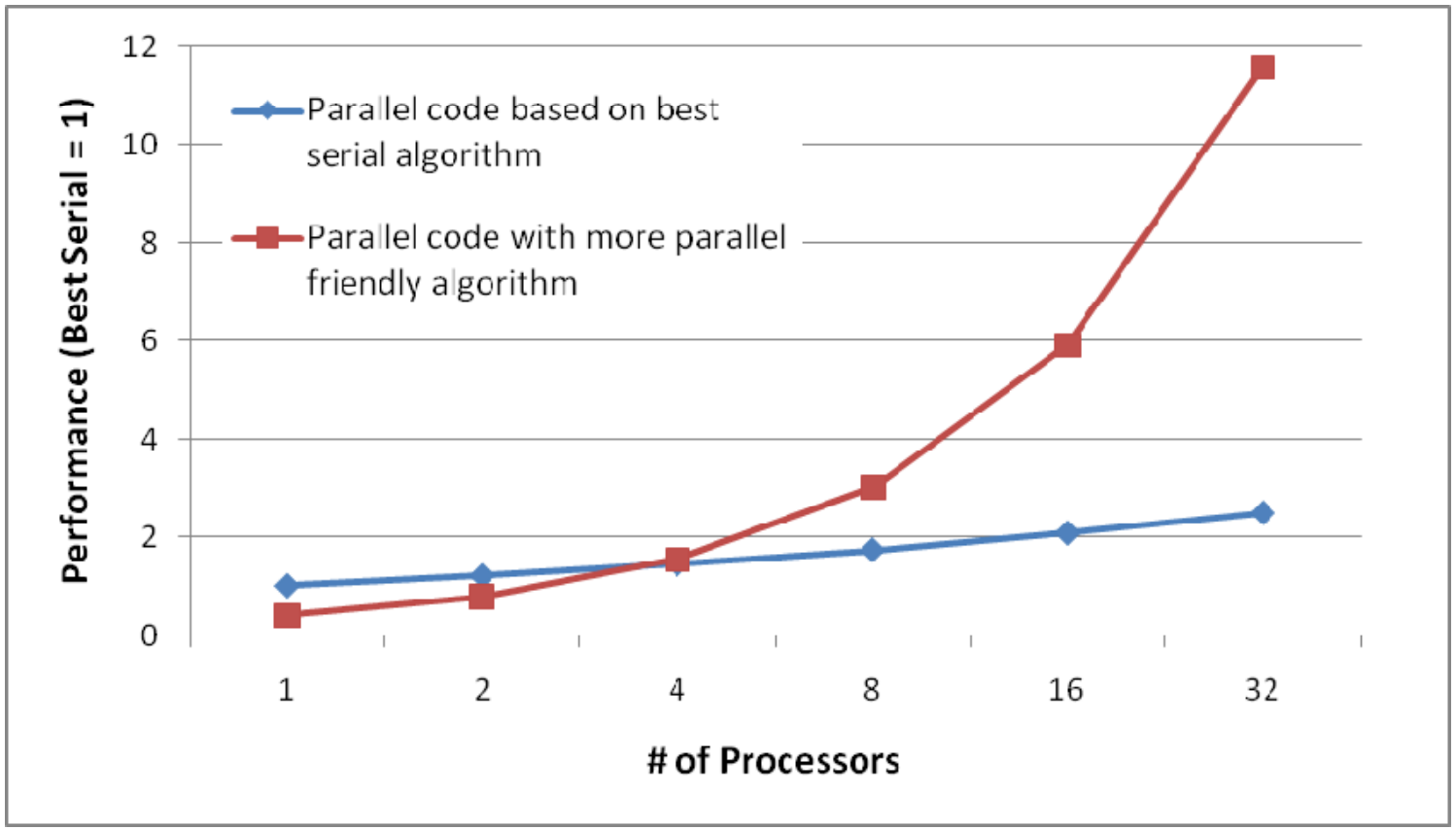
- What's hard about it?

We don't think in parallel

Parallel algorithms are after-thoughts

Parallel Programming

- Best serial code doesn't always scale well for large # of processors



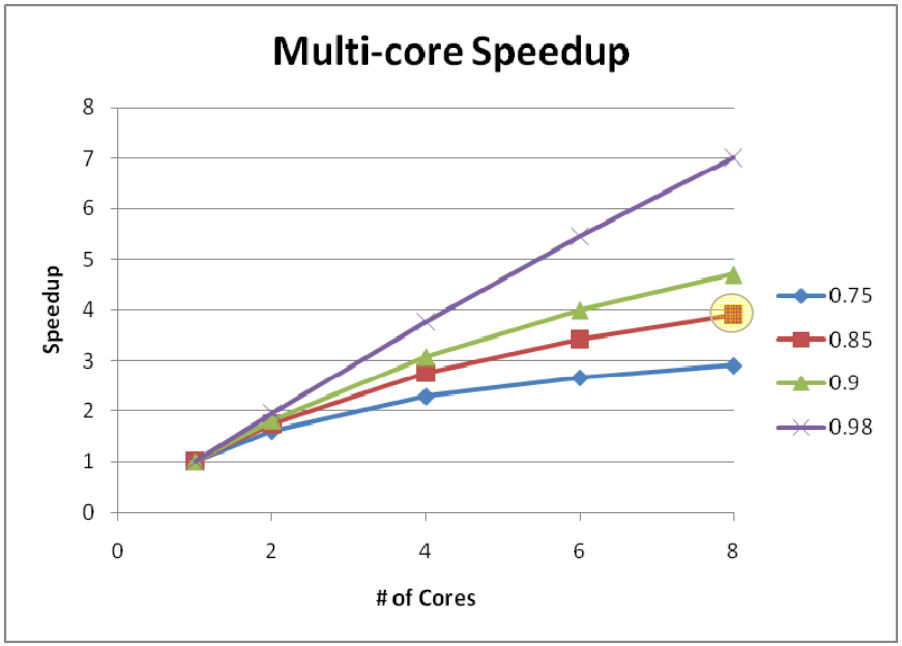
Scalability for Multi-Core

- Amdahl's law for multi-core architecture:

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

Serial component

Parallel component



Scalability of Many-Core

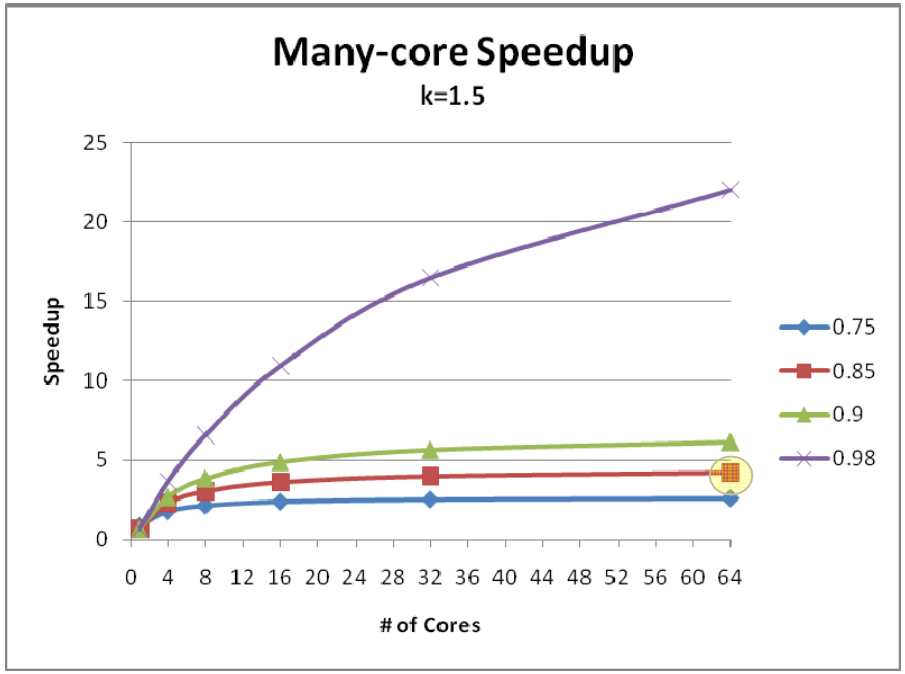
- Amdahl's law for many-core architecture:

$$S = \frac{1}{(1-P)\cdot K_N + \frac{P}{N}}$$

Serial component

Parallel component

Perf. ratio between 1 core in single-core processor and many-core processor



Significant portion of applications must be parallelized to achieve good scaling

Challenges (Summary)

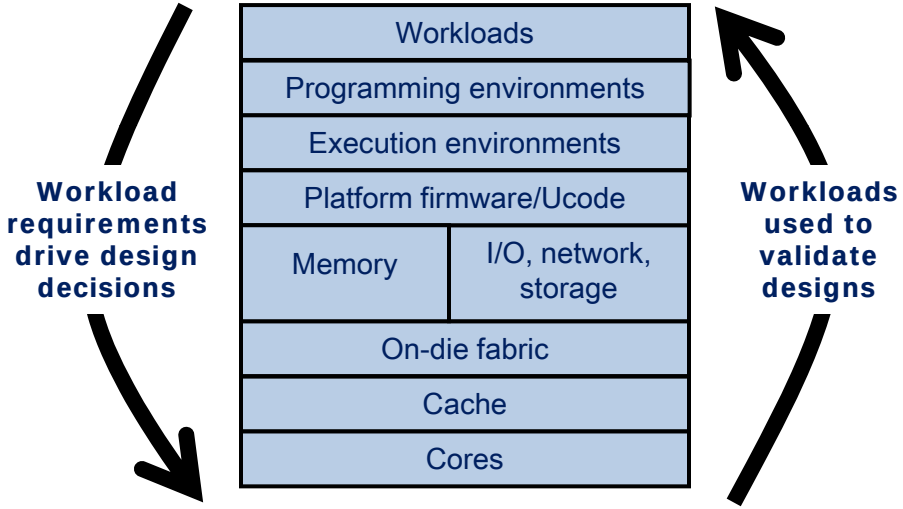
- Architecture changes for many-core
  - Compute density vs. compute efficiency
  - Data management: *Feeding the Beast*
- Algorithms
  - Is the best scalar algorithm suitable for parallel computing
- Programming model
  - Human tends to think in sequential steps. Parallel computing is not natural
  - Non-ninja parallel programming

Our approach

Application Specific
HW/SW Co-design

Our Approach: App-Arch Co-Design

Architecture-aware analysis of computational needs of parallel applications



Focus on specific co-travelers and domains:
HPC/Imaging/Finance/Physical Simulations/Medical/…

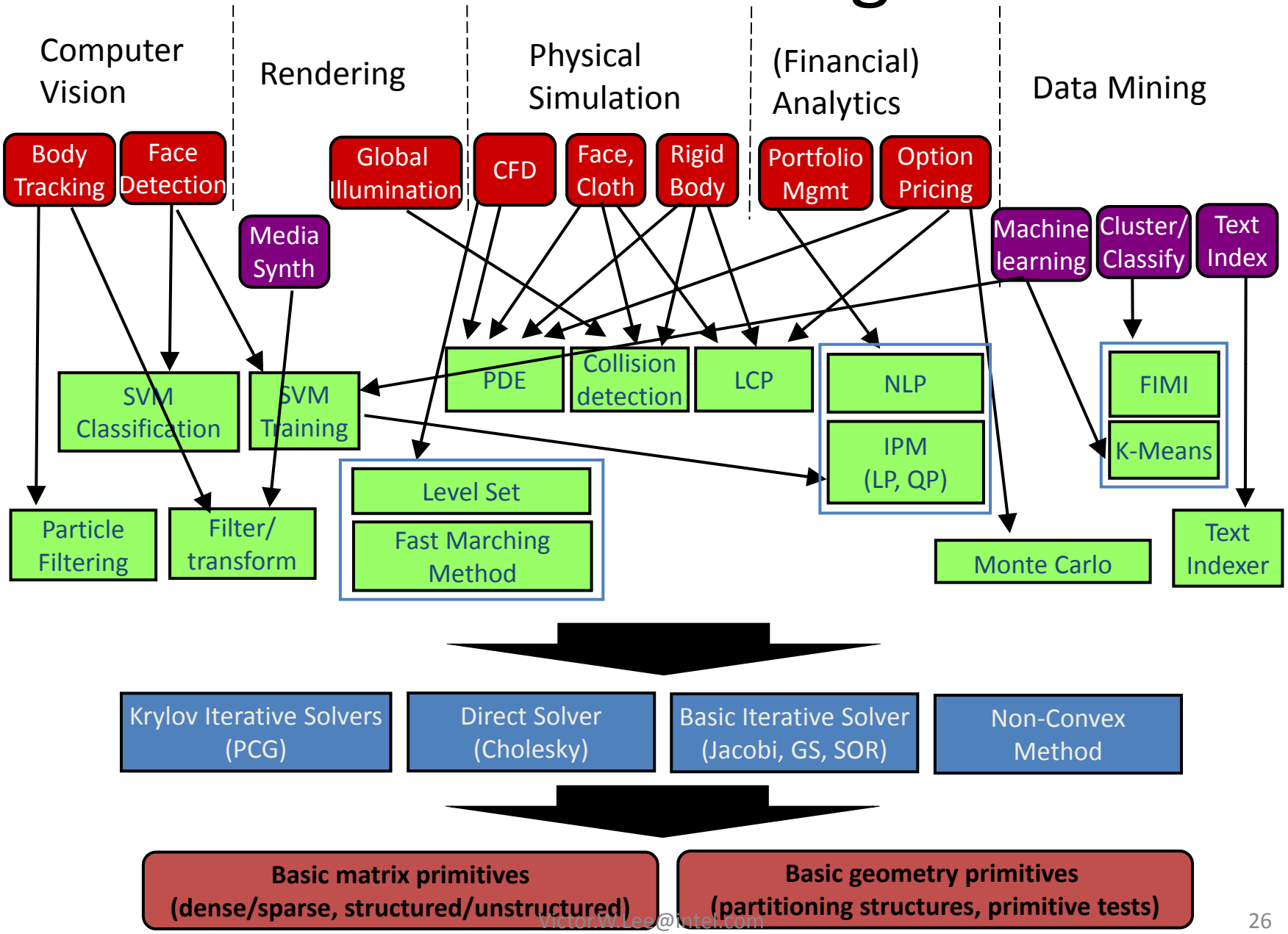
Multi-/Many-core features that accelerate applications in a power-efficient manner (bonus point: simplify programming)

Steps

1. Understand algorithm behind applications
2. Analysis characteristics of key kernels for algorithms
3. Evaluate the sensitivities to various architecture parameters
4. Develop architecture straw-man
5. Adjust algorithm to target architecture

Repeat Step 1 if necessary

Workload Convergence



Case-Study-I (3-D Stencil Operations)[1]

Algorithm/Optimization	Incremental Speedup
SIMDification	1.8X
Multi-threading (Non-blocked version is bandwidth bound)	2.1X

Perform Cache-blocking (2.5D Spatial + 1D Temporal)[2]

Blocking Optimization	1.7X
Multi-threading (Blocked version is compute-bound and scales further)	1.8X
SIMD Further scaling of compute-bound code	1.9X
ILP Optimization	1.1X
Overall Speedup	**24.1X**

1. Performance data on Intel Core i7 975, 4c at 3.33 GHz
2. Details in SC'10 paper (3.5-D Blocking Optimization for Stencil Computations on Modern CPUs and GPUs by Nguyen et al.)

Case-Study-II (FFT)[1]

Algorithm/Optimization	Incremental Speedup
Algorithm (Radix-4 Vs/ Radix-2)	1.72X
Multi-threading (Naïve Partitioning)	3.05X
Multi-threading (Intelligent Partitioning: load balanced)	1.23X
SIMDfication (Full V/s Partial SIMD)	1.18X
Memory Management (Double Buffering)	1.32X
Overall Speedup	**10.1X**

1. Performance data on Intel Core i7 975, 4c at 3.33 GHz

Case-Study-III
(Sparse Matrix Vector Multiplication)[1]

Algorithm/Optimization	Incremental Speedup
Multi-threading (Naïve Partitioning)	1.72X
Multi-threading (Intelligent Partitioning: load balanced)	2.2X
SIMDfication	1.13X
Cache Blocking	1.15X
Register Tiling	1.2X
Overall Speedup	**6.0X**

1. Performance data on Intel Core i7 975, 4c at 3.33 GHz

Case-Study-IV (Graph Traversal)[1]

Algorithm/Optimization	Incremental Speedup
Efficient Layout (Cache-Line Friendly)	10.1X
Hierarchical Blocking (Cache/TLB Friendly)	3.1X
SIMD	1.29X
ILP	1.35X
Multi-threading (Linear Scaling for compute-bound code)	3.9X

Overall Speedup	**212.6X**

1. Performance data on Intel Core i7 975, 4c at 3.33 GHz

Case-Study-V (Tree Search)[1,2]

Algorithm/Optimization	Incremental speedup
Efficient Layout (Memory Page-Blocking)	1.53X
Cache-Line Blocking	1.4X
SIMD	1.8X
ILP	2X
Multi-threading	3.9X

Overall Speedup	**30.1X**

1. Performance data on Intel Core i7 975, 4c at 3.33 GHz
2. Details in SIGMOD'10 paper (FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs by Kim et al.)

Case-Study-VI (Matrix Multiply)[1, 2]

Algorithm/Optimization	Incremental Speedup
Loop Inversion	9X
Cache-Tiling	1.33X
Multithreading	2.4X
SIMD	2.2X

Overall Speedup	**64X**

1. Performance data on Intel Core i7 975, 4c at 3.33 GHz
2. HiPC'2010 (Goa, India) Tutorial "Architecture Specific Optimizations for Modern Processors" by Dhiraj Kalamkar et.al.

Learning

- Parallel algorithms offer best speedup-effort RoI
  - Algorithmic core needs to evolve from pre-multicore era
- Technology-aware algorithmic improvements offer the next best speedup-effort RoI
  - Increasing compute density and data-parallelism
- Special attention to the least-scaling part of modern architectures: BW/op will be increasingly more critical to performance
  - Locality aware transformations
- Architecture-specific speedup is orders of magnitude less than commonly believed
  - *100-1000x CPU-GPU speedup myth*

Summary

Massive Data Computing
- Insatiable appetite for compute
- It's all about three C's:
  - Content – Connect -- Compute

Algorithmic Opportunity
- Algorithmic core needs to evolve from serial to parallel
- Massive data approach to traditional compute problems
  - *Data … data everywhere, … not a bit of sense …* ☺

Performance Challenge
- Performance variability on the rise with parallel architectures
- *Feeding the Beast: increasingly a performance bottleneck*
- Programmer productivity key to market success