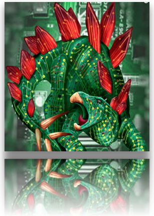


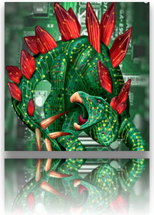
Parallel Applications (13 Dwarves)

Lecture 7

Several figures from:

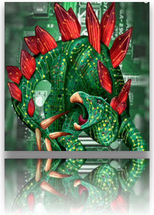
The Landscape of Parallel Computing Research: A View From Berkeley

13 Dwarves from the Hobbit



Parallel Applications

- ★ Paradigm shifts (revisit)
- ★ (13) Dwarfs
- ★ Programming Models
- ★ Embarrassingly Parallel (MapReduce)

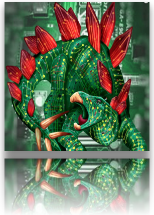


Paradigm shifts (Conventional Wisdom, Berkeley)

- ★ Was - Power is free, but transistors expensive
  Now - Power is expensive, but transistors are **free**.
- ★ Was - Only concern is dynamic power
  Now - leakage (static power) is 40% of total power
- ★ Was - Uniprocessors are reliable
  Now - below 65nm, soft/hard error rates
- ★ Was - build a chip to demonstrate architecture
  Now - simulate?

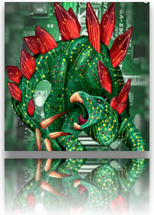
Paradigm shifts (ctd.) (Conventional Wisdom, Berkeley)

- ★ Was - ALU slow, loads/stores fast
  Now - ALU fast, loads/stores slow
- ★ Was - **More ILP via compilers and architecture innovation**
  Now - **ILP wall** (diminishing returns)
- ★ Was - 2x (CPU) performance every 18 months
  Now - **Power Wall + Memory Wall + ILP Wall**
  System on Chip (eg. **Apple M1**) Processor is the new transistor.
- ★ Was - Clock scaling
  Now - Processors Parallelism



Paradigm shifts (ctd.) (Conventional Wisdom, Berkeley)

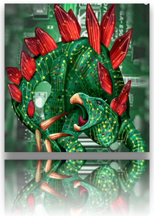
- ★ Was - Don’t parallelizing app, just use faster computer
  Now - No more 1 processor per chip. New programming model.
  Vectorization (CUDA), Thread-Level Parallelism
- ★ Was - Less than linear scaling is failure
  Now - Sublinear speedups are beneficial



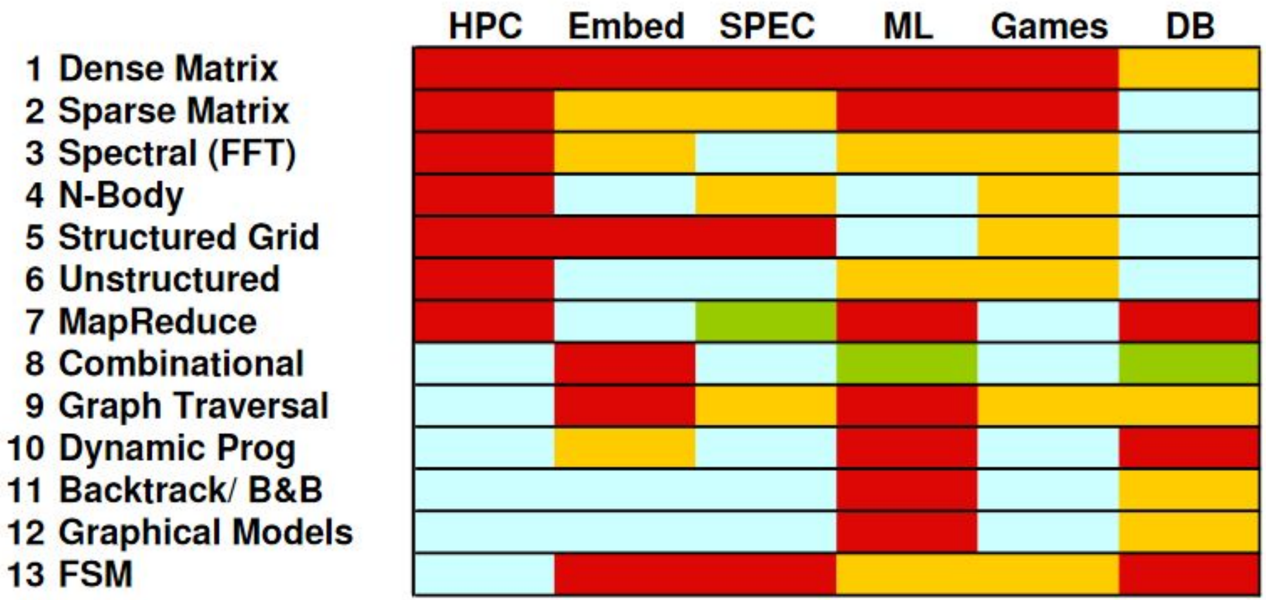
(13) Dwarfts

- ★ Dense Linear Algebra
- ★ Sparse Linear Algebra
- ★ Spectral Methods
- ★ N-Body Methods
- ★ Structured Grids
- ★ Unstructured Grids
- ★ MapReduce
  (aka. Embarrassingly Parallel)

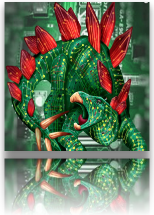
- ★ Combinational Logics
- ★ Graph Traversal
- ★ Dynamic Programming
- ★ Back-track/Branch & Bound
- ★ Graphical Model Inference
- ★ Finite State Machine



Dwarf Popularity (Red Hot → Blue Cool)



Taken from https://web.stanford.edu/class/ee380/Abstracts/070131-BerkeleyView1.7.pdf



Programming Models

Current Programming model, developers have to explicitly identify task.

Data Model is usually ignored.

Autotuners ? (Compilers)

Model	*Domain*	*Task Identification*	*Task Mapping*	*Data Distribution*	*Communication Mapping*	*Synchronization*
Real-Time Workshop [MathWorks 2004]	DSP	Explicit	Explicit	Explicit	Explicit	Explicit
TejaNP [Teja 2003]	Network	Explicit	Explicit	Explicit	Explicit	Explicit
YAPI [Brunel et al 2000]	DSP	Explicit	Explicit	Explicit	Explicit	Implicit
MPI [Snir et al 1998]	HPC	Explicit	Explicit	Explicit	Implicit	Implicit
Pthreads [Pthreads 2004]	General	Explicit	Explicit	Implicit	Implicit	Explicit
StreamIt [Gordon et al 2002]	DSP	Explicit	Implicit	Explicit	Implicit	Implicit
MapReduce [Dean and Ghemawat 2004]	Large Data sets	Explicit	Implicit	Implicit	Implicit	Explicit
Click to network processors [Plishker et al 2004]	Network	Implicit	Implicit	Implicit	Implicit	Explicit
OpenMP [OpenMP 2006]	HPC	Implicit (directives, some explicit)	Implicit	Implicit	Implicit	Implicit (directives, some explicit)
HPF [Koelbel et al 1993]	HPC	Implicit	Implicit	Implicit (directives)	Implicit	Implicit

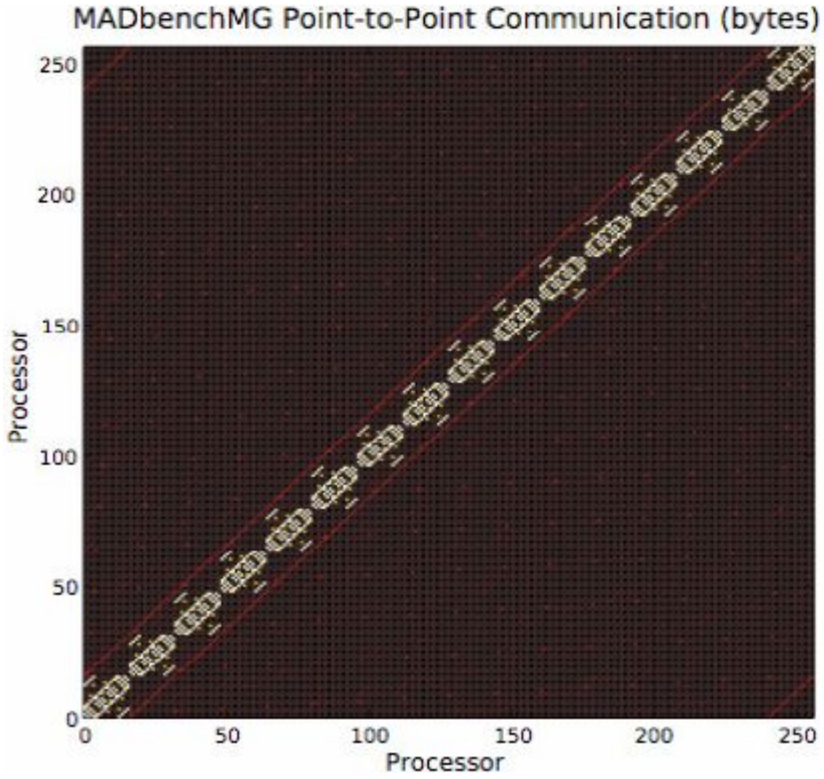
Figure 7. Comparison of 10 current parallel programming models for 5 critical tasks, sorted from most explicit to most implicit. High-performance computing applications [Pancake and Bergmark 1990] and embedded applications [Shah et al 2004a] suggest these tasks must be addressed one way or the other by a programming model: 1) Dividing the application into parallel tasks; 2) Mapping computational tasks to processing elements; 3) Distribution of data to memory elements; 4) mapping of communication to the inter-connection network; and 5) Inter-task synchronization.

Dense Matrix

Block Triadiagonal Matrix, Lower Upper

Symmetric Gauss-Seidel /

Vector computers, Array computers



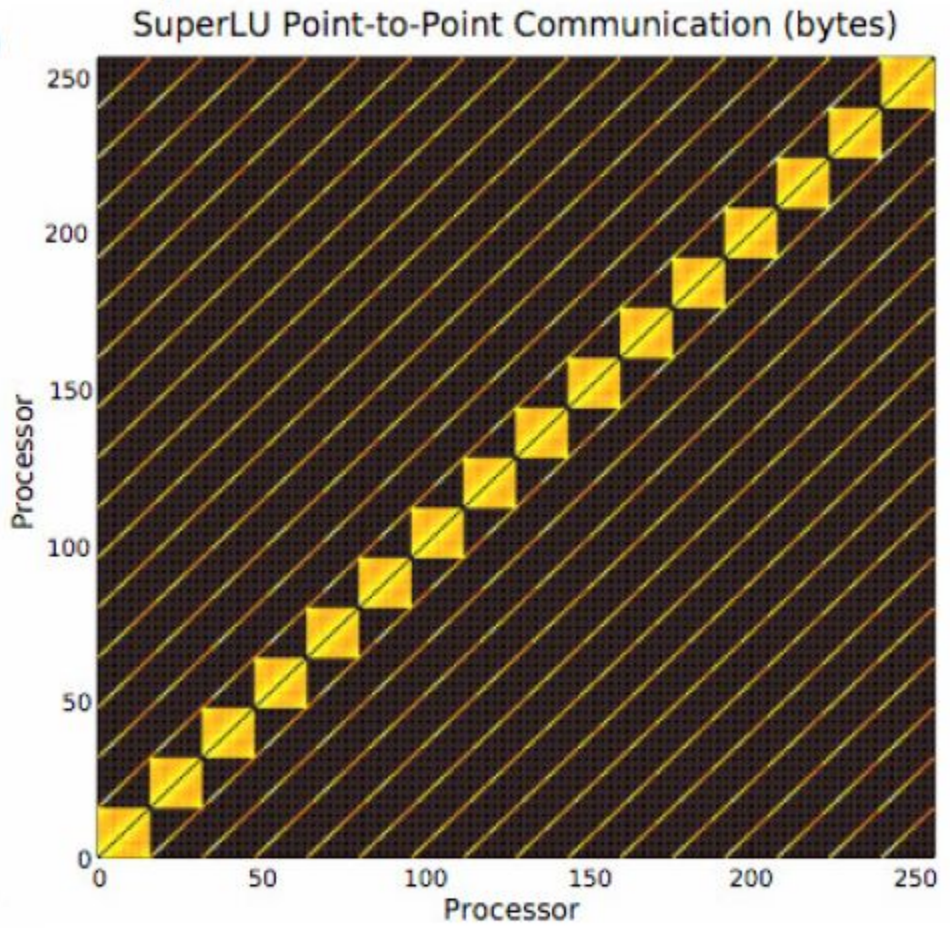
Sparse Matrix

Conjugate

Gradient / Vector

computers with

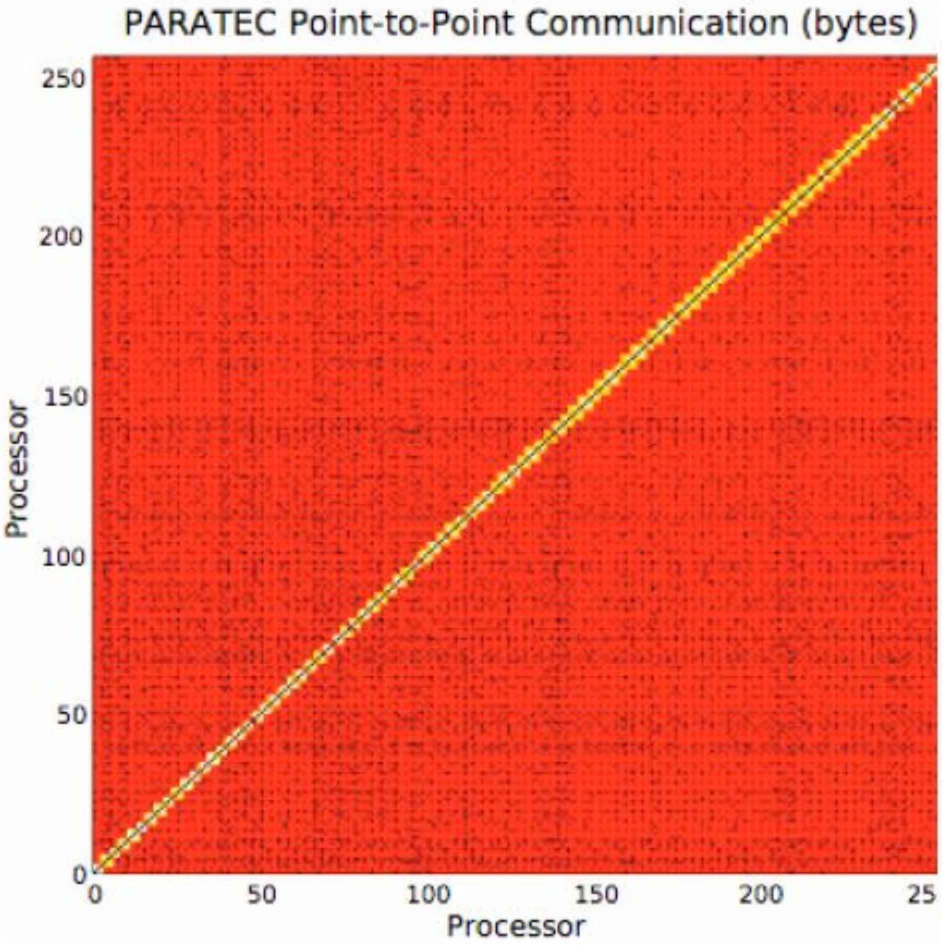
gather/scatter



Spectral

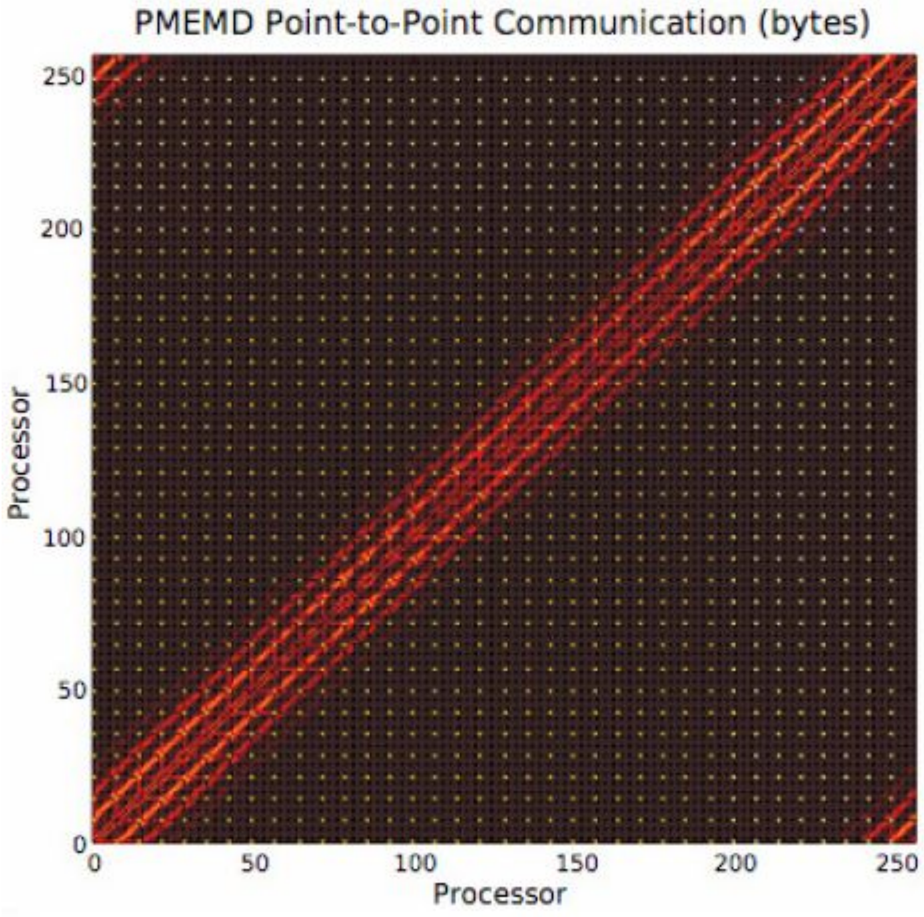
Fourier Transform / DSPs,

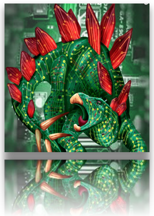
Data are in the frequency domain, as opposed to time or spatial domains. Typically, spectral methods use multiple butterfly stages, which combine multiply-add operations and a specific pattern of data permutation, with all-to-all communication for some stages and strictly local for others.



N-Body Methods

Depends on interactions between many discrete points. Variations include particle-particle methods, where every point depends on all others, leading to an O(N^2) calculation, and hierarchical particle methods, which combine forces or potentials from multiple points to reduce the computational complexity to O(N log N) or O(N).

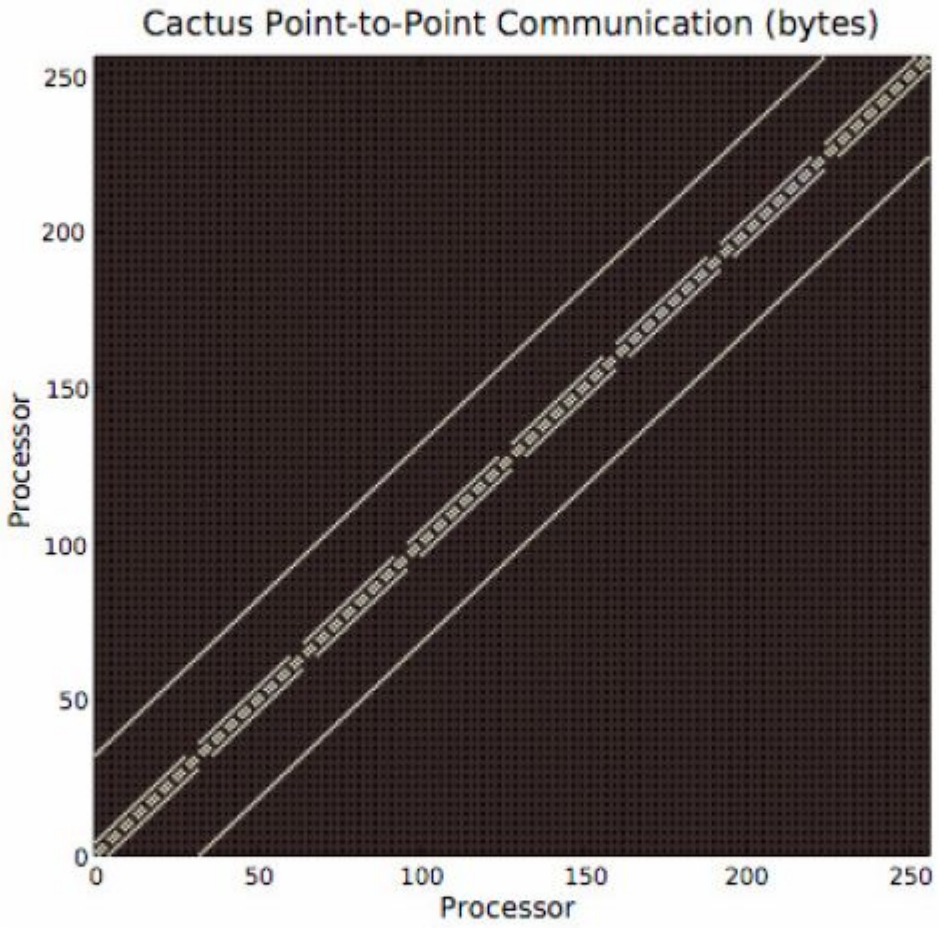




Structured Grid

Multi-Grid, Scalar Pentadiagonal / QCDOC

Represented by a regular grid; points on grid are conceptually updated together. It has high spatial locality. Updates may be in place or between 2 versions of the grid. The grid may be subdivided into finer grids in areas of interest (“Adaptive Mesh Refinement”); and the transition between granularities may happen dynamically.

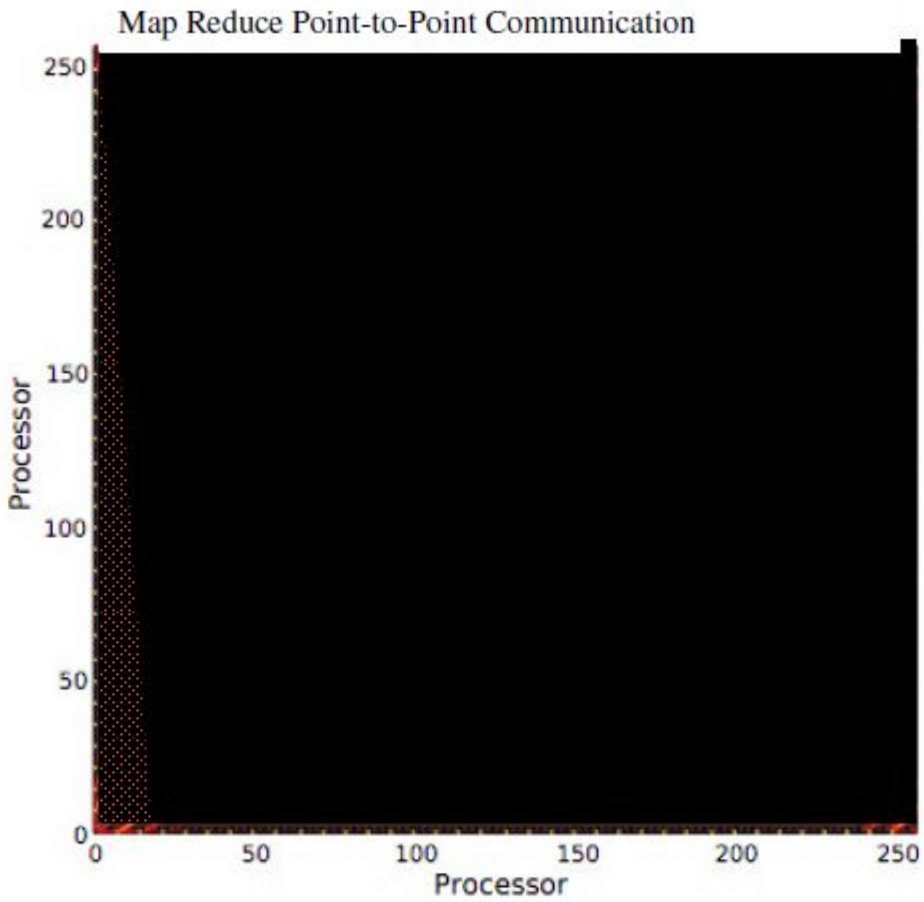


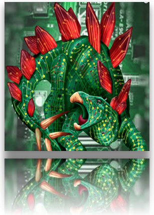
Map Reduce

Embarrassingly Parallel / NSF Teragrid

Calculations depend on statistical results of repeated random trials. Considered embarrassingly parallel.

(No dependency/ no need for communication between tasks)





Others

- ★ Combinational Logic
  - Encryption, CRC, ...
- ★ Graph Traverse
  - Indirect table lookups
- ★ Graphical Models
  - Bayesian Networks, Hidden Markov Models
- ★ Finite State Machines
- ★ Machine Learning
  - Dynamic Programming
  - Backtrack and Branch and Bound

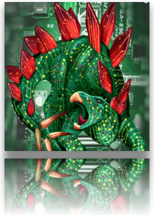
FYI

Support Vector Machine - Dense-Linear Algebra

Principal Component Analysis - Dense/Sparse Linear Algebra

Decision Trees - Graph Traversal

Hashing - Combinational Logic



Dwarf	*Embedded Computing*	*General Purpose Computing*	*Machine Learning*	*Graphics / Games*	*Databases*	*Intel RMS*
1. Dense Linear Algebra (e.g., BLAS or MATLAB)	*EEMBC Automotive*: iDCT, FIR, IIR, Matrix Arith; *EEMBC Consumer*: JPEG, RGB to CMYK, RGB to YIQ; *EEMBC Digital Entertainment*: RSA MP3 Decode, MPEG-2 Decode, MPEG-2 Encode, MPEG-4 Decode; MPEG-4 Encode; *EEMBC Networking*: IP Packet; *EEMBC Office Automation*: Image Rotation; *EEMBC Telecom*: Convolution Encode; *EEMBC Java*: PNG	*SPEC Integer*: Quantum computer simulation (libquantum), video compression (h264avc) *SPEC Fl. Pl.*: Hidden Markov models (sphinx3)	Support vector machines, princpal component analysis, independent component analysis		Database hash accesses large contiguous sections of memory	Body Tracking, media synthesis linear programming, K-means, support vector machines, quadratic programming, *PDE*: Face, *PDE*: Cloth*
2. Sparse Linear Algebra (e.g., SpMV, OSKI, or SuperLU)	*EEMBC Automotive*: Basic Int + FP, Bit Manip, CAN Remote Data, Table Lookup, Tooth to Spark; *EEMBC Telecom*: Bit Allocation; *EEMBC Java*: PNG	*SPEC Fl. Pt.*: Fluid dynamics (bwaves), quantum chemistry (gamess; tonto), linear program solver (soplex)	Support vector machines, principal component analysis, independent component analysis	Reverse kinematics; Spring models		Support vector machines, quadratic programming, *PDE*: Face, *PDE*: Cloth* *PDE*: Computational fluid dynamics
3. Spectral Methods (e.g., FFT)	*EEMBC Automotive*: FFT, iFFT, iDCT; *EEMBC Consumer*: JPEG; *EEMBC Entertainment*: MP3 Decode		Spectral clustering	Texture maps		*PDE*: Computational fluid dynamics *PDE*: Cloth
4. N-Body Methods (e.g., Barnes-Hut, Fast Multipole Method)		*SPEC Fl. Pt.*: Molecular dynamics (gromacs, 32-bit; namd, 64-bit)				
5. Structured Grids (e.g., Cactus or Lattice-Boltzmann Magneto-	*EEMBC Automotive*: FIR, IIR; *EEMBC Consumer*: HP Gray-Scale; *EEMBC Consumer*: JPEG; *EEMBC Digital Entertainment*: MP3 Decode, MPEG-2 Decode, MPEG-2 Encode, MPEG-4 Decode; MPEG-4 Encode; *EEMBC Office Automation*:	*SPEC Fl. Pt.*: Quantum chromodynamics (milc),magneto hydrodynamics (zeusmp), general relativity (cactusADM), fluid dynamics (leslie3d-AMR; lbm), finite element methods (dealII-AMR; calculix), Maxwell's E&M		Smoothing; interpolation		

Dwarf	*Embedded Computing*	*General Purpose Computing*	*Machine Learning*	*Graphics / Games*	*Databases*	*Intel RMS*
hydrodynamics)	Dithering; *EEMBC Telecom*: Autocorrelation	eqns solver (GemsFDTD), quantum crystallography (tonto), weather modeling (wrf2-AMR)				
6. Unstructured Grids (e.g., ABAQUS or FIDAP)			Belief propagation			Global illumination
7. MapReduce (e.g., Monte Carlo)		*SPEC Fl. Pt.*: Ray tracer (povray)	Expectation maximization		MapReduce	
8. Combinational Logic	*EEMBC Digital Entertainment*: AES, DES ; *EEMBC Networking*: IP Packet, IP NAT, Route Lookup; *EEMBC Office Automation*: Image Rotation; *EEMBC Telecom*: Convolution Encode		Hashing		Hashing	
9. Graph Traversal	*EEMBC Automotive*: Pointer Chasing, Tooth to Spark; *EEMBC Networking*: IP NAT, OSPF, Route Lookup; *EEMBC Office Automation*: Text Processing; *EEMBC Java*: Chess, XML Parsing		Bayesian networks, decision trees	Reverse kinematics, collision detection, depth sorting, hidden surface removal	Transitive closure	Natural language processing
10. Dynamic Programming	*EEMBC Telecom*: Viterbi Decode	*SPEC Integer*: Go (gobmk)	Forward-backward, inside-outside, variable elimination, value iteration		Query optimization	
11. Back-track and Branch +Bound		*SPEC Integer*: Chess (sjeng), network simplex algorithm (mcf), 2D path finding library (astar)	Kernel regression, constraint satisfaction, satisfiability			
12. Graphical Models	*EEMBC Telecom*: Viterbi Decode	*SPEC Integer*: Hidden Markov models (hmmer)	Hidden Markov models			

Dwarf	*Embedded Computing*	*General Purpose Computing*	*Machine Learning*	*Graphics / Games*	*Databases*	*Intel RMS*
13. Finite State Machine	*EEMBC Automotive*: Angle To Time, Cache "Buster", CAN Remote Data, PWM, Road Speed, Tooth to Spark; *EEMBC Consumer*: JPEG; *EEMBC Digital Entertainment*: Huffman Decode, MP3 Decode, MPEG-2 Decode, MPEG-2 Encode, MPEG-4 Decode; MPEG-4 Encode; *EEMBC Networking*: QoS, TCP; *EEMBC Office Automation*: Text Processing; *EEMBC Telecom*: Bit Allocation; *EEMBC Java:* PNG	*SPEC Integer*: Text processing (perlbench), compression (bzip2), compiler (gcc), video compression (h264avc), network discrete event simulation (omnetpp), XML transformation (xalancbmk)		Response to collisions		

Figure 6. Mapping of EEMBC, SPEC2006, Machine Learning, Graphcs/Games, Data Base, and Intel's RMS to the 13 Dwarfs. *Note that SVM, QP, PDE:Face, and PDE:Cloth may use either dense or sparse matrices, depending on the application.

Limitations

Dwarf	Performance Limit: Memory Bandwidth, Memory Latency, or Computation?
1. Dense Matrix	Computationally limited
2. Sparse Matrix	Currently 50% computation, 50% memory BW
3. Spectral (FFT)	Memory latency limited
4. N-Body	Computationally limited
5. Structured Grid	Currently more memory bandwidth limited
6. Unstructured Grid	Memory latency limited
7. MapReduce	Problem dependent
8. Combinational Logic	CRC problems BW; crypto problems computationally limited
9. Graph traversal	Memory latency limited
10. Dynamic Programming	Memory latency limited
11. Backtrack and Branch+Bound	?
12. Construct Graphical Models	?
13. Finite State Machine	Nothing helps!

Figure 9. Limits to performance of dwarfs, inspired by an suggestion by IBM that a packaging technology could offer virtually infinite memory bandwidth. While the memory wall limited performance for almost half the dwarfs, memory latency is a bigger problem than memory bandwidth

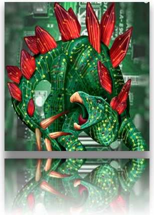
MapReduce

MapReduce is also referred to as Embarrassingly Parallel. (Explicitly Parallel)

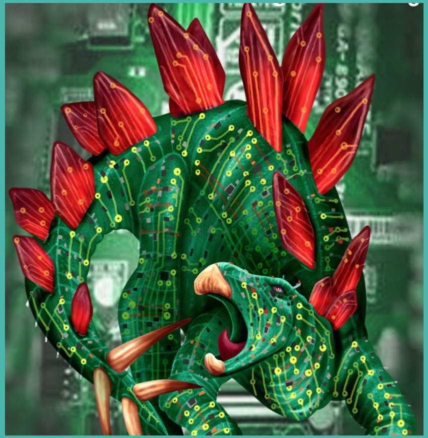
See this video.

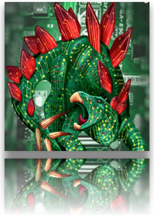
https://drive.google.com/file/d/0B6p-RWKTApyUWHRoWldDMGNnTzg/view?usp=sharing

See a live demo from couchdb



Exercises

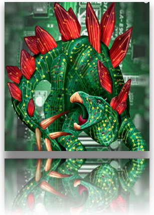




Exercises

★ Use CouchDB, to demonstrate an aggregate query/view (eg. sum, average) in MapReduce manner. Explain how such design can be useful in cluster environment.

In particular, emulate this SQL statements
SELECT age, average(salary) FROM employee GROUP BY age;



End of Lecture 7