Concepts and Models of Parallel and Data-centric Programming

Parallel Algorithms I

Lecture, Summer 2020

Dr. Christian Terboven <terboven@itc.rwth-aachen.de>

Outline

Parallel Algorithms

What is this Chapter about?

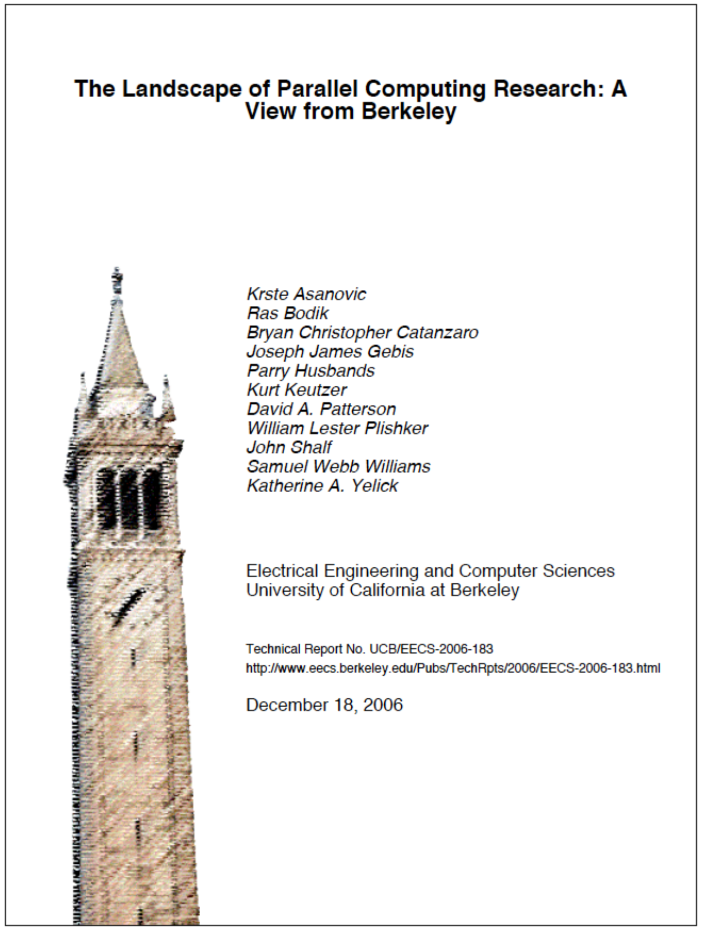
- Application of the concepts and methods for parallel programming learned in the first part of this lecture…

- … based on common classes of parallel algorithms defined by the Berkeley Dwarfs
  - Dense Linear Algebra
  - Sparse Linear Algebra
  - Monte Carlo Methods
  - Graph Traversal

- Decision-making on which parallel programming model to choose

Books & Links

- The Landscape of Parallel Computing Research: A View from Berkeley
  - → Asanovic et al., Technical Report No. UCB/EECS-2006-183, 2006
  - → https://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html



The Landscape of Parallel Computing Research: A View from Berkeley

Krste Asanovic
Ras Bodik
Bryan Christopher Catanzaro
Joseph James Gebis
Parry Husbands
Kurt Keutzer
David A. Patterson
William Lester Plishker
John Shalf
Samuel Webb Williams
Katherine A. Yelick

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-183
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

December 18, 2006

Recap

- Main parallel programming models
  - Shared memory programming
  - Hardware accelerator programming
  - Message passing programming
  - Distributed shared memory programming

- What problems can we solve with these programming models?

- Which programming model to chose for which kind of problem?

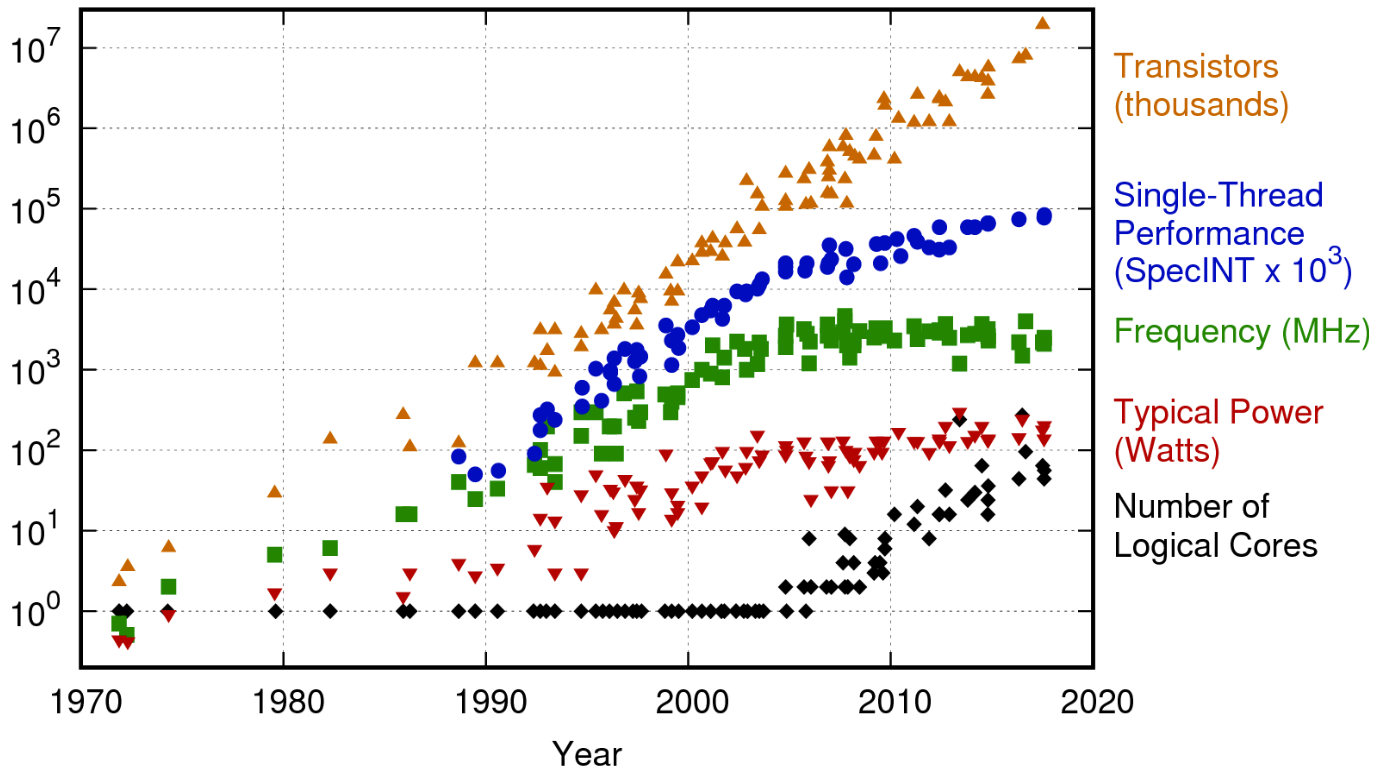
Motivation for the definition of the DWARFS / 1

- The free lunch is over [1]
  - 2x CPU Performance every 18 months until ~2005

[1] H. Sutter, The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software, Dr. Dobb's Journal 30(3), March 2005.

- Frequency stagnates
- Instruction-level parallelism per core stagnates
- Number of cores is growing

42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Motivation for the definition of the DWARFS / 2

- Major change in chip design towards multicore designs

- Berkeley researchers from many backgrounds met between February 2005 and December 2006 to discuss these changes and their impact on software and hardware
  - Topics included: Circuit design, computer architecture, massively parallel computing, computer-aided design, embedded hardware and software, programming languages, compilers, scientific programming, and numerical analysis

- Led to 7 questions to frame parallel research

7 Questions for Parallelism

- Applications:
  1. What are the apps?
  2. What are kernels of apps?
- Hardware:
  3. What are the HW building blocks?
  4. How to connect them?
- Programming Model & Systems Software:
  5. How to describe apps and kernels?
  6. How to program the HW?
- Evaluation:
  7. How to measure success?

7 Questions for Parallelism

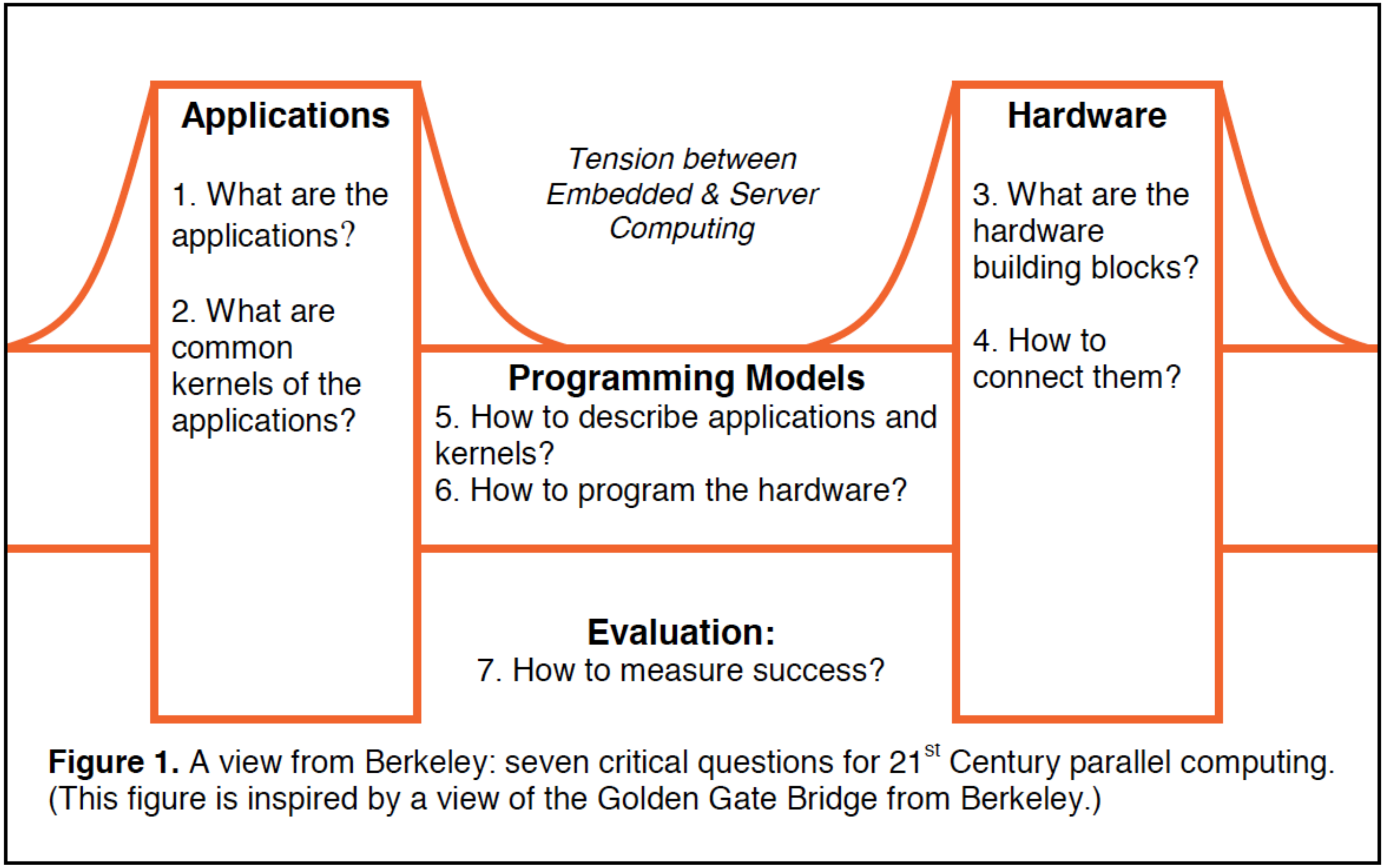

Applications
1. What are the applications?
2. What are common kernels of the applications?
Tension between Embedded & Server Computing
Hardware
3. What are the hardware building blocks?
4. How to connect them?
Programming Models
5. How to describe applications and kernels?
6. How to program the hardware?
Evaluation:
7. How to measure success?

Figure 1. A view from Berkeley: seven critical questions for 21st Century parallel computing. (This figure is inspired by a view of the Golden Gate Bridge from Berkeley.)

"A View from Berkeley", Asanovic et al., 2006

Phillip Colella's "Seven Dwarfs"

- A dwarf is a pattern of computation and communication

- Dwarfs are well-defined targets from algorithmic, software, and architecture standpoints

- Seven dwarfs (seven numerical methods)
  1. Dense Linear Algebra
  2. Sparse Linear Algebra
  3. Spectral Methods
  4. N-Body Methods
  5. Structured Grids
  6. Unstructured Grids
  7. Monte Carlo

"Defining Software Requirements for Scientific Computing", Phillip Colella, 2004

Overview of the 7 Dwarfs

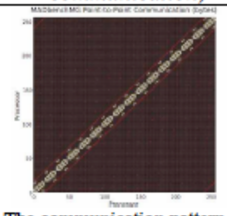
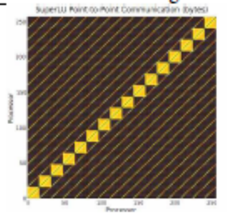
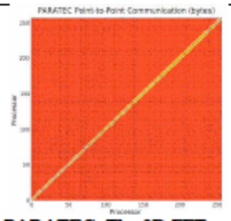
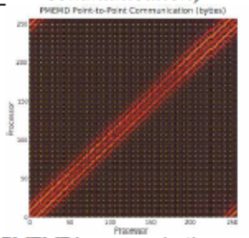
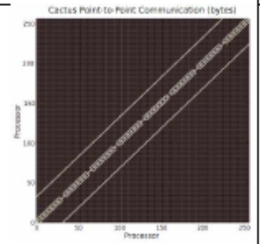
Dwarf	*Description*	*Communication Pattern (Figure axes show processors 1 to 256, with black meaning no communication)*	*NAS Benchmark / Example HW*
1. Dense Linear Algebra (e.g., BLAS [Blackford et al 2002], ScaLAPACK [Blackford et al 1996], or MATLAB [MathWorks 2006])	Data are dense matrices or vectors. (BLAS Level 1 = vector-vector; Level 2 = matrix-vector; and Level 3 = matrix-matrix.) Generally, such applications use unit-stride memory accesses to read data from rows, and strided accesses to read data from columns.	 The communication pattern of MadBench, which makes heavy use of ScaLAPACK for parallel dense linear algebra, is typical of a much broader class of numerical algorithms	Block Triadiagonal Matrix, Lower Upper Symmetric Gauss-Seidel / Vector computers, Array computers
2. Sparse Linear Algebra (e.g., SpMV, OSKI [OSKI 2006], or SuperLU [Demmel et al 1999])	Data sets include many zero values. Data is usually stored in compressed matrices to reduce the storage and bandwidth requirements to access all of the nonzero values. One example is block compressed sparse row (BCSR). Because of the compressed formats, data is generally accessed with indexed loads and stores.	 SuperLU (communication pattern pictured above) uses the BCSR method for implementing sparse LU factorization.	Conjugate Gradient / Vector computers with gather/scatter
3. Spectral Methods (e.g., FFT [Cooley and Tukey 1965])	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.	 PARATEC: The 3D FFT requires an all-to-all communication to implement a 3D transpose, which requires communication between every link. The diagonal stripe describes BLAS-3 dominated linear-algebra step required for orthogonalization.	Fourier Transform / DSPs, Zalink PDSP [Zarlink 2006]
4. N-Body Methods (e.g., Barnes-Hut [Barnes and Hut 1986], Fast Multipole Method [Greengard and Rokhlin 1987])	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).	 PMEMD's communication pattern is that of a particle mesh Ewald calculation.	(no benchmark) / GRAPE [Tokyo 2006], MD-GRAPE [IBM 2006]
5. Structured Grids (e.g., Cactus [Goodale et al 2003] or Lattice-Boltzmann Magneto-hydrodynamics [LBMHD 2005])	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.	 Communication pattern for Cactus, a PDE solver using 7-point stencil on 3D block-structured grids.	Multi-Grid, Scalar Penta-diagonal / QCDOC [Edinburg 2006], BlueGeneL
6. Unstructured Grids (e.g., ABAQUS [ABAQUS 2006] or FIDAP [FLUENT 2006])	An irregular grid where data locations are selected, usually by underlying characteristics of the application. Data point location and connectivity of neighboring points must be explicit. The points on the grid are conceptually updated together. Updates typically involve multiple levels of memory reference indirection, as an update to any point requires first determining a list of neighboring points, and then loading values from those neighboring points.		Unstructured Adaptive / Vector computers with gather/scatter, Tera Multi Threaded Architecture [Berry et al 2006]
7. Monte Carlo (e.g., Quantum Monte Carlo [Aspuru-Guzik et al 2005])	Calculations depend on statistical results of repeated random trials. Considered embarrassingly parallel.	Communication is typically not dominant in Monte Carlo methods.	Embarrassingly Parallel / NSF Teragrid

Figure 3. Seven Dwarfs, their descriptions, corresponding NAS benchmarks, and example computers.

Extension of the Seven Dwarfs

- Further discussion to include a broader array of computational methods
  - Embedded Computing, Desktop/Server Computing, Machine Learning, Games/Graphics/Vision, Data Base Software
- Main questions:
  - How well do the Seven Dwarfs of high performance computing capture computation and communication patterns for a broader range of applications?
  - What dwarfs need to be added to cover the missing important areas beyond high performance computing?
- Result: Added 7 more dwarfs, revised 2 original dwarfs, renumbered list

"A View from Berkeley", Asanovic et al., 2006

Current Set of Dwarfs

1. Dense Linear Algebra
2. Sparse Linear Algebra
3. Spectral Methods
4. N-Body Methods
5. Structured Grids
6. Unstructured Grids
7. MapReduce
8. Combinational Logic
9. Graph Traversal
10. Dynamic Programming
11. Back-track/Branch & Bound
12. Graphical Model Inference
13. Finite State Machine

"A View from Berkeley", Asanovic et al., 2006

Why are Dwarfs important?

- Common terminology/ organization to talk across disciplines
- Building blocks for libraries, applications, tools, etc.
- Claim: If parallel architecture, language, compiler run dwarfs well they will run parallel apps of future well
  - Set of requirements for new HW/ SW
- Not tied to code (higher level of abstraction)
  - Drive innovation based on pattern of computation and communication and not a specific implementation