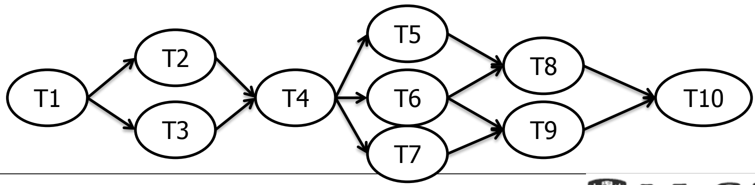
Case: Showing Work for Parallel Execution

- Divide work into tasks that can be executed concurrently
- Many different decompositions possible for any computation
- Tasks may be same, different, or even unknown sizes
- Tasks may be independent or have non-trivial order
- Conceptualize tasks and ordering as task dependency graph
  - A *directed acyclic graph (DAG)*
  - Node = task
  - Edge or arc = control dependence



Example: Database Query

- Task: set of elements that satisfy a predicate -> table
- Edge: output of one task serves as input to the next

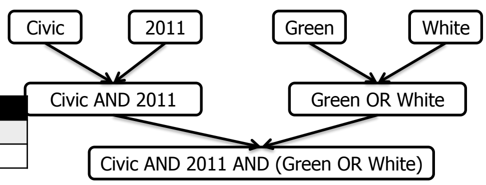
MODEL="CIVIC" AND YEAR=2011 AND (COLOR="GREEN" OR COLOR="WHITE")

ID#	Model
4523	Civic
6734	Civic
4395	Civic
7352	Civic

ID#	Year
7623	2011
9834	2011
6734	2011
5342	2011
3845	2011
4395	2011

ID#	Color
7623	Green
9834	Green
5342	Green
8354	Green

ID#	Color
3476	White
6734	White



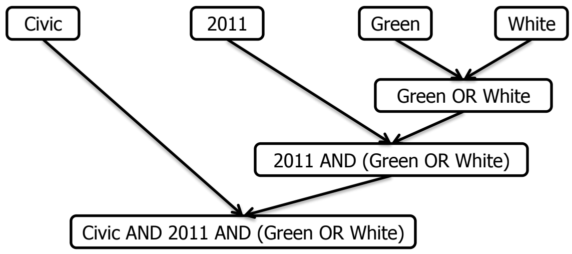
ID#	Color
3476	White
7623	Green
9834	Green
6734	White
5342	Green
8354	Green

ID#	Model	Year
6734	Civic	2011
4395	Civic	2011

ID#	Model	Year	Color	Dealer	Price
6734	Civic	2011	White	BC	$17,000

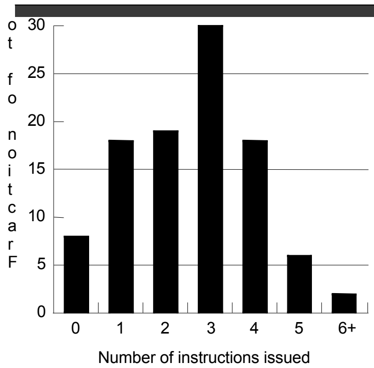
Alternative Query Decomposition

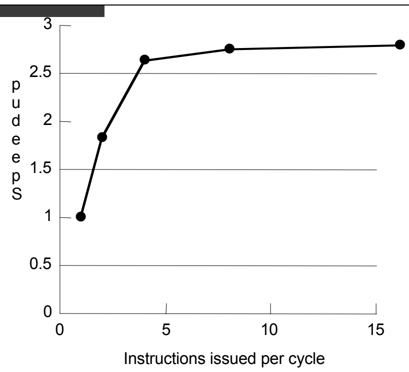
MODEL="CIVIC" AND YEAR=2011 AND
(COLOR="GREEN" OR COLOR="WHITE")



- Different decompositions may yield different parallelism

How Far ILP Goes?

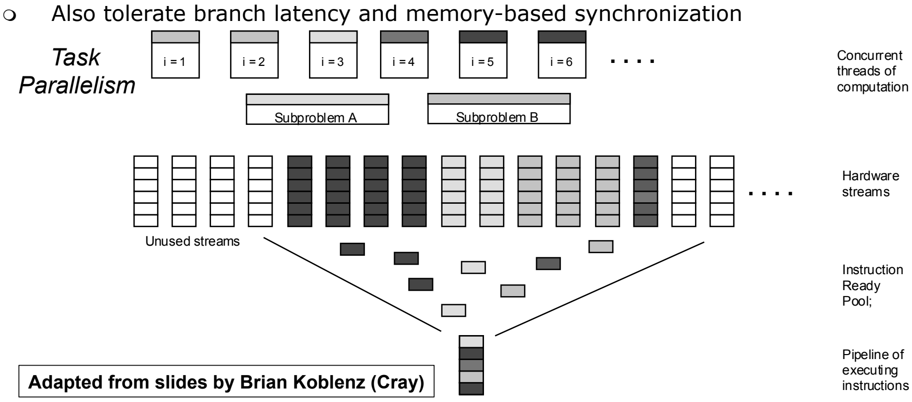




- Assumptions: infinite resources and fetch bandwidth, perfect branch prediction and renaming

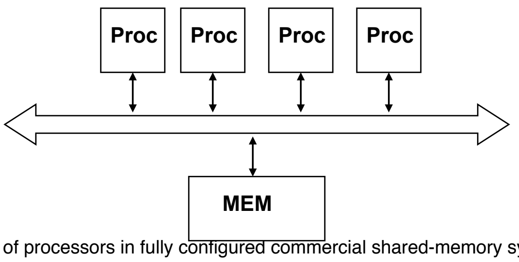
Tolerating Latency - Multithreading

- Multiple active threads - long latency filled by instructions in other threads
- Tolerate latency and keep pipelines full
- Also tolerate branch latency and memory-based synchronization



Adapted from slides by Brian Koblenz (Cray)

Thread Level Parallelism Use



No. of processors in fully configured commercial shared-memory systems

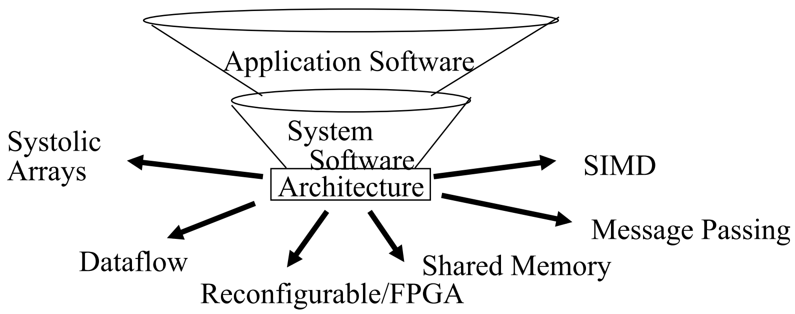
- Multiple CPUs with shared memory
  - dominates server and enterprise market, moving down to desktop
- More responsive with multiple CPUs
  - OS Kernel just picks another processor for next thread

Some Taxonomies

- Number of Instructions/Data
  - SISD – serial
  - SIMD
  - MISD
  - MIMD
- Parallelism planes
  - Control-flow
  - Data
- Others: Single Program, Multiple Data (SPMD)

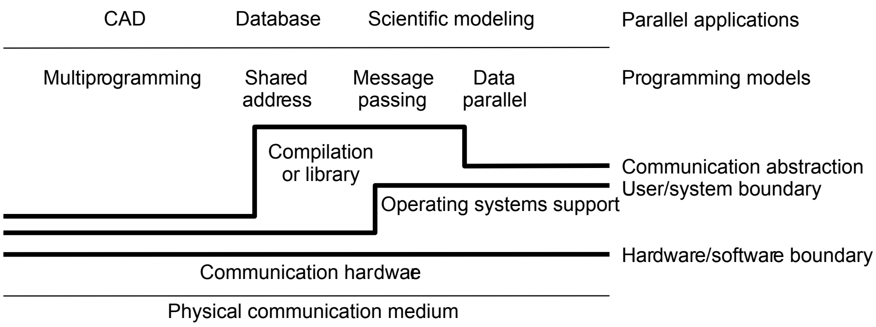
Parallel Architectures so Far

In Past: Divergent architectures, no predictable pattern of growth.



• Productivity crisis: software development suffers with dissimilar models

Modern Layered Framework

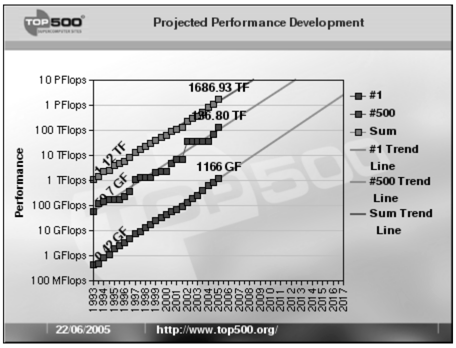


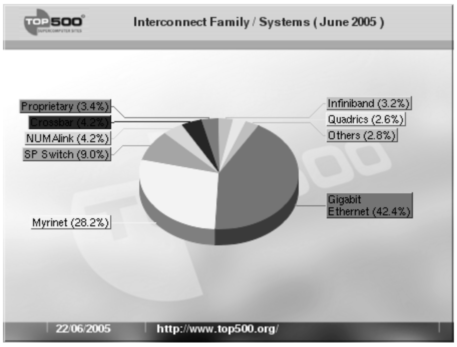
Module 4: Machines, SW Eng. Issues

- Fastest Machines
  - Current state, trends
- High Performance Computing Demands
- Software Engineering of HPC
  - Challenges in large scale programming
  - Workflows, lifecycle
- Performance: speedup

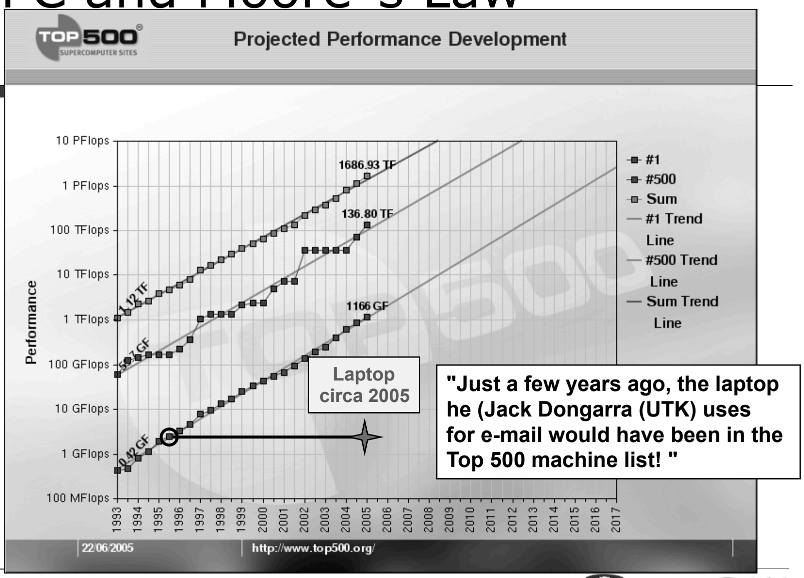
Fastest Computers

- www.top500.org – trends, makers, applications, users

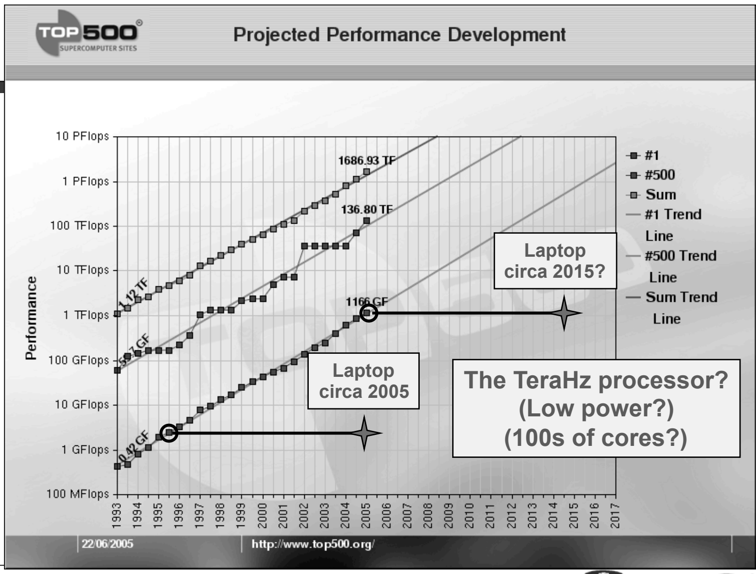




HPC and Moore's Law



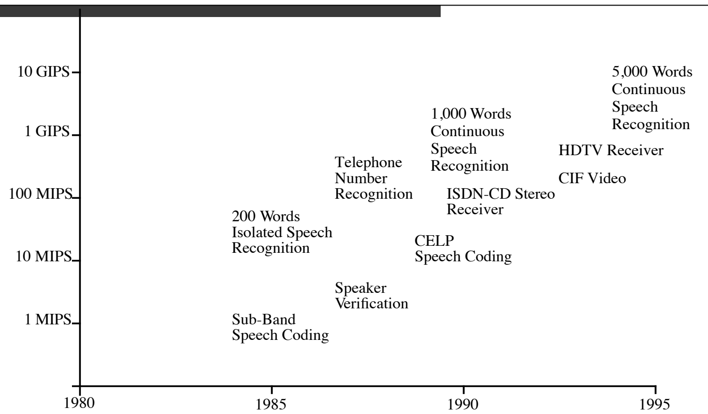
HPC and Moore's Law



Engineering Computing Demand

- Large parallel machines a must in many industries
  - Petroleum (reservoir analysis)
  - Automotive (crash simulation, drag analysis, combustion efficiency),
  - Aeronautics (airflow analysis, engine efficiency, structural mechanics, electromagnetism),
  - Computer-aided design
  - Pharmaceuticals (molecular modeling)
  - Visualization
    - in all of the above
    - entertainment (films like Toy Story)
    - architecture (walk-throughs and rendering)
  - Financial modeling (yield and derivative analysis)

Applications: Speech and Image Processing



- Also CAD, Databases, . . .
- *100 processors gets you 10 years, 1000 gets you 20* !

Module 4: SW. Eng. Aspects

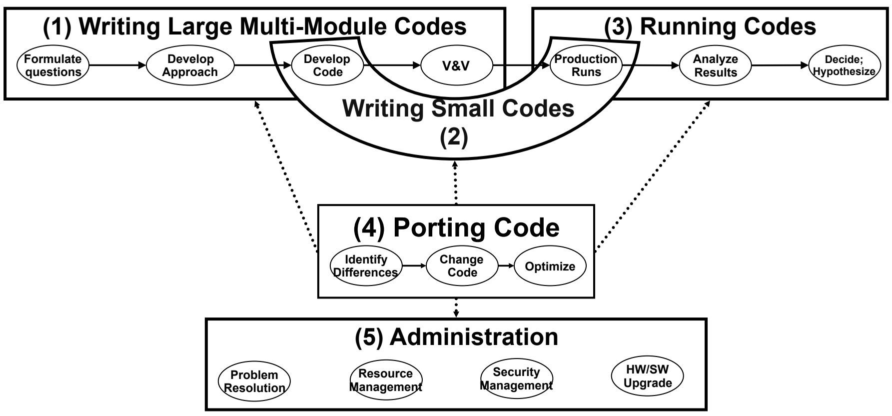
- Challenges in HPC code design
- HPC Workflows
- HPC SW Lifecycle
- Large scale HPC programming
- Performance: speedup

HPC Programming - Challenges

```
// Gaussian Elimination
for (k = 0 to n-1)
{
        for (i = k+1 to n)
        {
                z = A[i][k] / A[k][k];
                for (j = k+1 to n)
                        A[i][j] = A[i][j] - z * A[k][j]; y[i] = y[i] - z * y[k];
        }
}
```

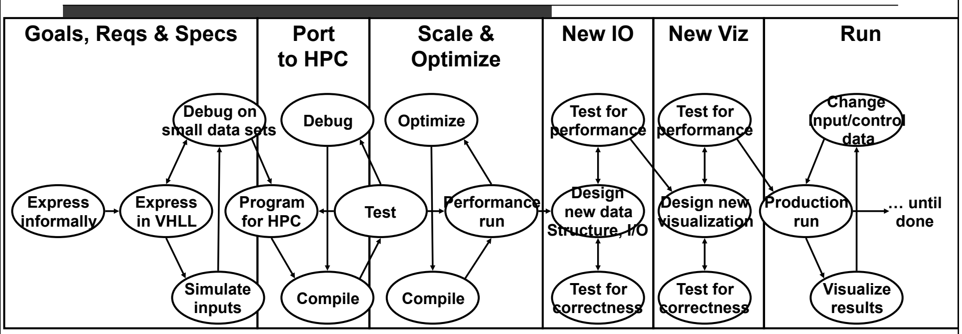
- Top500 High Performance Linpack (HPL)
  - Parallel, portable, block oriented
  - Exploit high-levels of temporal locality (reuse data)
  - Code size — >10,000 lines of FORTRAN + MPI
  - One of the most highly optimized codes for performance!

US Government HPC Workflows



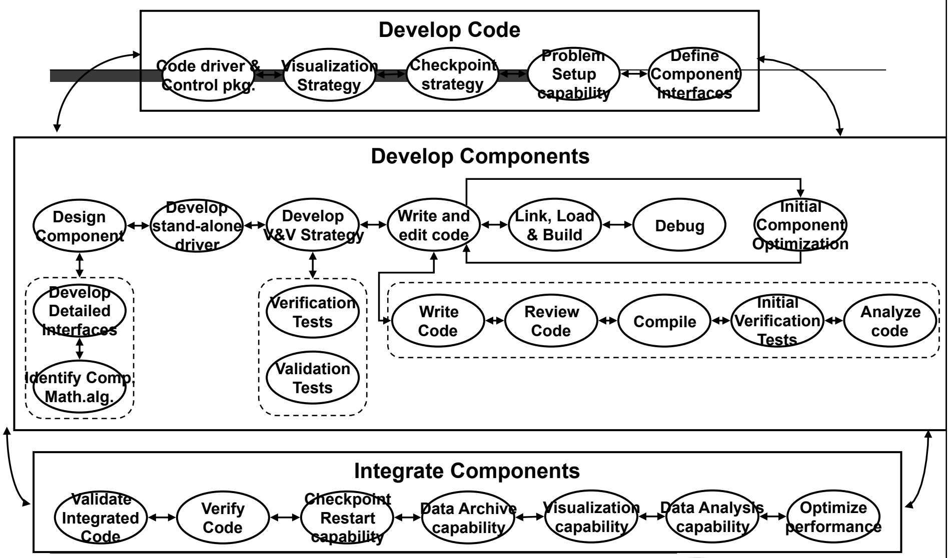
- Many overlapping steps. Item in red - HPC specific interest

Coding Programs Quickly

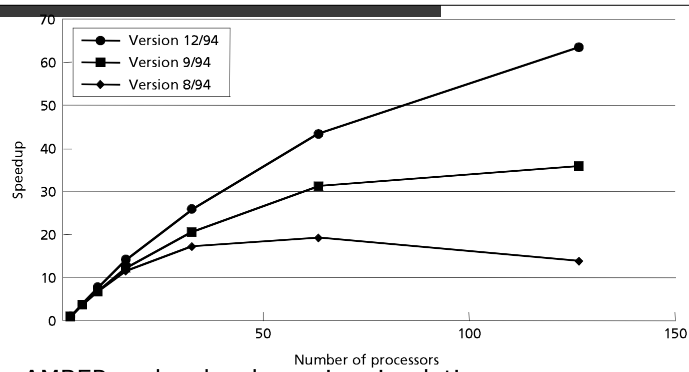


- Allows vendors to specifically identify which steps they are addressing with their technologies

Multi-Module Development



Is better parallel arch enough?

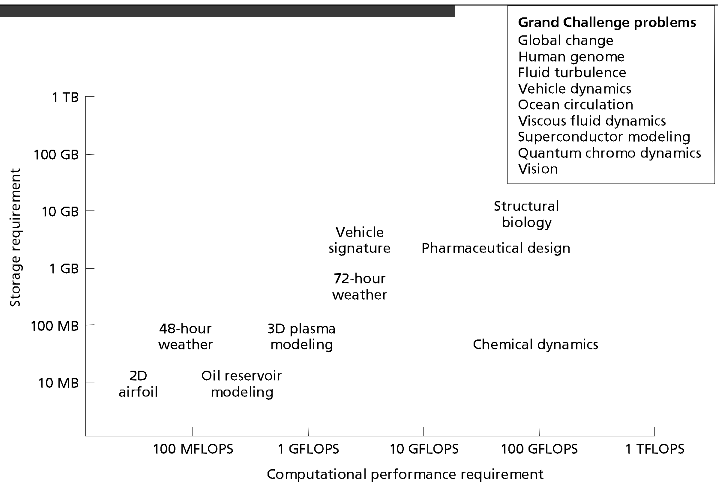


- AMBER molecular dynamics simulation program
- Starting point was vector code for Cray-1
- 145 MFLOP on Cray90, 406 for final version on 128-processor Paragon, 891 on 128-processor Cray T3D

Summary of Application Trends

- Transition to parallel computing has occurred for scientific and engineering computing
- In rapid progress in commercial computing
  - Database and transactions as well as financial
  - Usually smaller-scale, but large-scale systems also used
- Desktop also uses multithreaded programs, which are a lot like parallel programs
- Demand for improving throughput on sequential workloads
  - Greatest use of small-scale multiprocessors
- Solid application demand exists and will increase

Scientific Computing Demand



Main Measure: Speedup

- Speedup (p processors) = $\frac{\textit{Performance (N processors)}}{\textit{Performance (1 processor)}}$
- For a fixed problem size (input data set), Perf = 1/time
- Speedup (N processors) = $\frac{\textit{Time (1 processor)}}{\textit{Time (N processors)}}$
- Issue: comparison to uniprocessor version

Tale of Two Laws

- Amdahl – control-flow parallelism
  - Sequential part -> SP
- $S=\frac{T1}{TN}=\frac{T1}{SP*T1+(1-SP)T1/N}=\frac{N}{SP*N+(1-SP)}$
- Pessimistic – no data parallelism
  - Does not apply to SIMD, SPMD
- Gustafson-Barsis
  - Normalized: *TN*=1
- *S=T1=N-(N-1)*SP*

Finally: Grading Scheme

- 15% homeworks (3)
- 30% labs (3)
- 25% midterm exam
- 30% project (teams of 1-2)

Acknowledgments

- D. Koester, MITRE
- NUMAchine group
- Authors of recommended textbooks