Sun's HPCS Approach: Hero

John L. Gustafson, HPCS PI

Sun Labs, Mountain View, CA

john.gustafson@sun.com

Caveats

- I will present Sun's priorities, philosophy, and approach for HPCS system design.
- Sun Labs creates enabling technologies; it doesn't design products. This talk is a Sun Labs perspective. Products are years away.

Background/Perspective

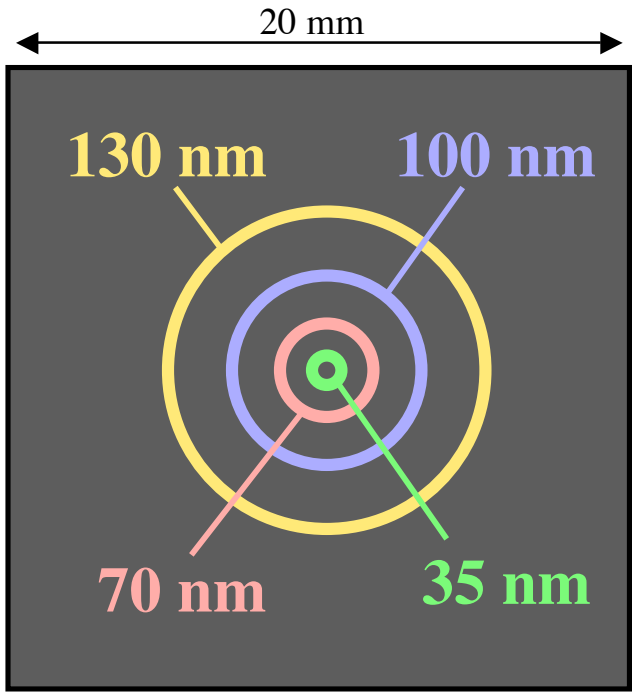
- Sun's boom in e-business and database apps has masked the HPC side of the company.
- During the boom, Sun absorbed pieces of Cray, Thinking Machines, Genias, Maximum Strategy, and other HPC strongholds.
- The DARPA HPCS program had an incredibly big effect steering Sun ($3M steering a $12B company).
- We are starting with a clean slate and designing from HPC *customer workloads*.

HPC Hardware Philosophy

- First-order constraints: size, heat, cost.
- Reduce size to reduce latency (speed of light).
- Heat removal=> Power density => lower latency and affordable high bandwidth.
- Make every watt count. (Note that this might go counter to speculative computing).
- More than ever, supercomputing means aggressive cooling design.
- Modular design allows greatly reduced cost.

The "Speed of Light" Isn't What It Used To Be

- The clock can't make it across a 20 mm die at current GHz rates
- Either we go to multiple cores or use Non-Uniform Cache Access (NUCA)
- This is a *physical* limit, not *technological*



[Source: Chuck Moore, UT-Austin]

Bandwidth Burns Energy; Maybe Our Measure of Work is… Joules?

Operation	**Energy** (130 nm, 1.2 V)
32-bit ALU operation	5 pJ
32-bit register read	10 pJ
Read 32 bits from 8K RAM	50 pJ
Move 32 bits across 10 mm chip	100 pJ
Move 32 bits off chip	1300 to 1900 pJ

[source: Bill Dalley, Stanford]

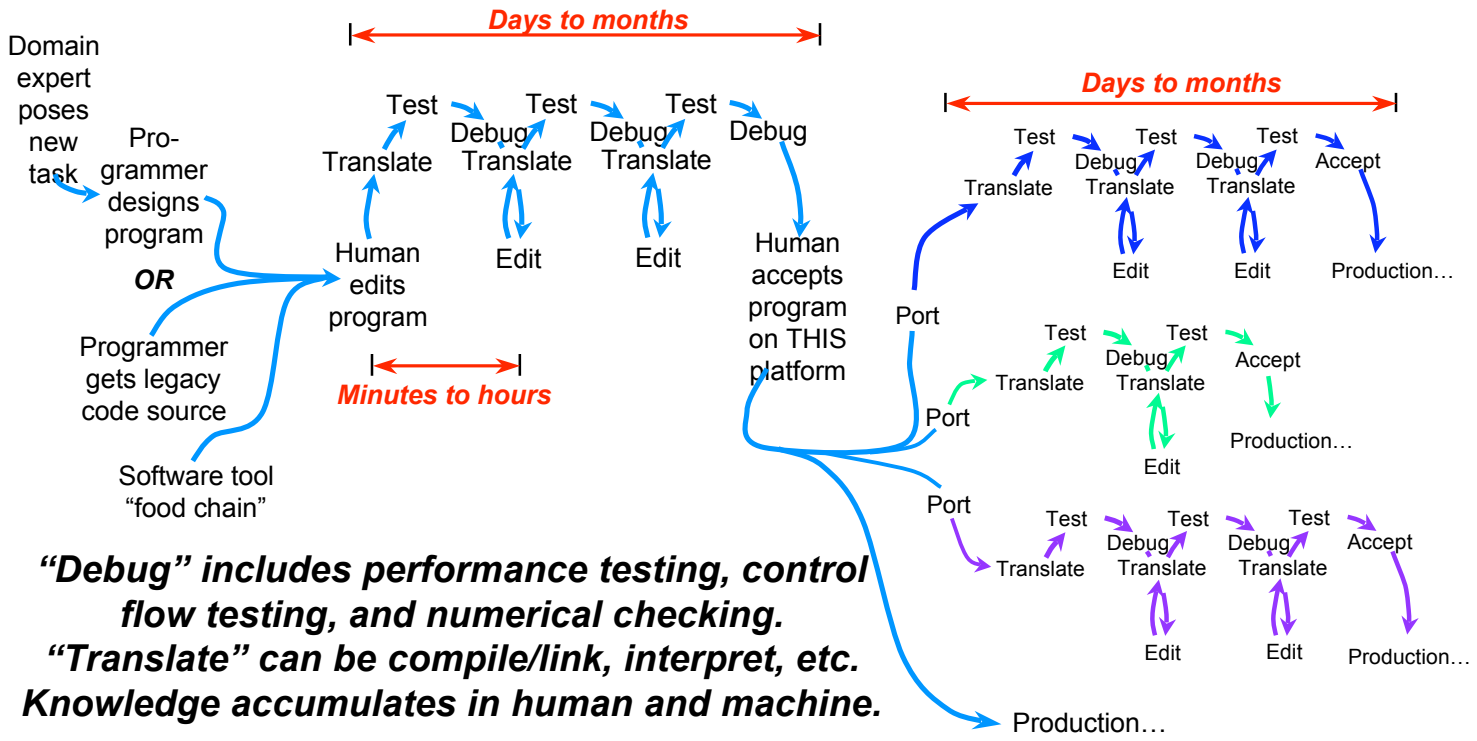
Asynchronous Computing

- You won't be able to ask "what's the clock rate?" much longer. There won't be one.
- Saves power, enables incredible bandwidth gains, relieves RF noise, and lets every part of a system go as *fast as it can*.
- Only drawback: hell to design and test.

HPC Software Philosophy

- Consider domain specialist, programmer, and user to be *different people* in general.
- Look for ways to free those people from having to do any job that's not theirs to do.
- Look for new things that the computer can do to unburden humans in the development task, especially in the *run-time environment*.

SW Development Cycles



Mom & Apple Pie Comment Page

- Big systems will always have some part in failure mode, and must have resilience built in.
- Must be *cheaper* or *more capable* than commodity Linux clusters, or why bother?
- Support legacy languages. Duh.
- Will we support MPI, OpenMP, or CAF-type program model? Yes.
- Libraries are for cleverness-intensive tasks.

Design for Hardest Case

- Design for sparse matrices, and dense matrices are easy.
- Design for irregular grids, and regular grids are easy.
- Design for dynamic loads and static loads are easy.
- Design for high-quality arithmetic, and low-precision tasks are easy. And if you do it right, the non-associativity problem goes away…
- Design for high I/O optionally, and those who don't need it don't have to pay for it.

…*but*

Don't try to do HPC and commercial applications with a single architecture!

Inflammatory Comments Page

- Current HPC benchmarks are grossly inadequate for guiding vendors to make the right choices.
- Complaining about “efficiency” as a fraction of peak FLOPS is a throwback to times when FLOPS dominated cost. Get real. Over 95% of cost is the memory system!
- It’s going to get hard to count “processors” as things get weird. Thread/node/processor/module, whatever.
- Anything below one memory op per flop puts too much pressure on programmers to be clever about data placement and timing of that placement.

An HPC Taxonomy

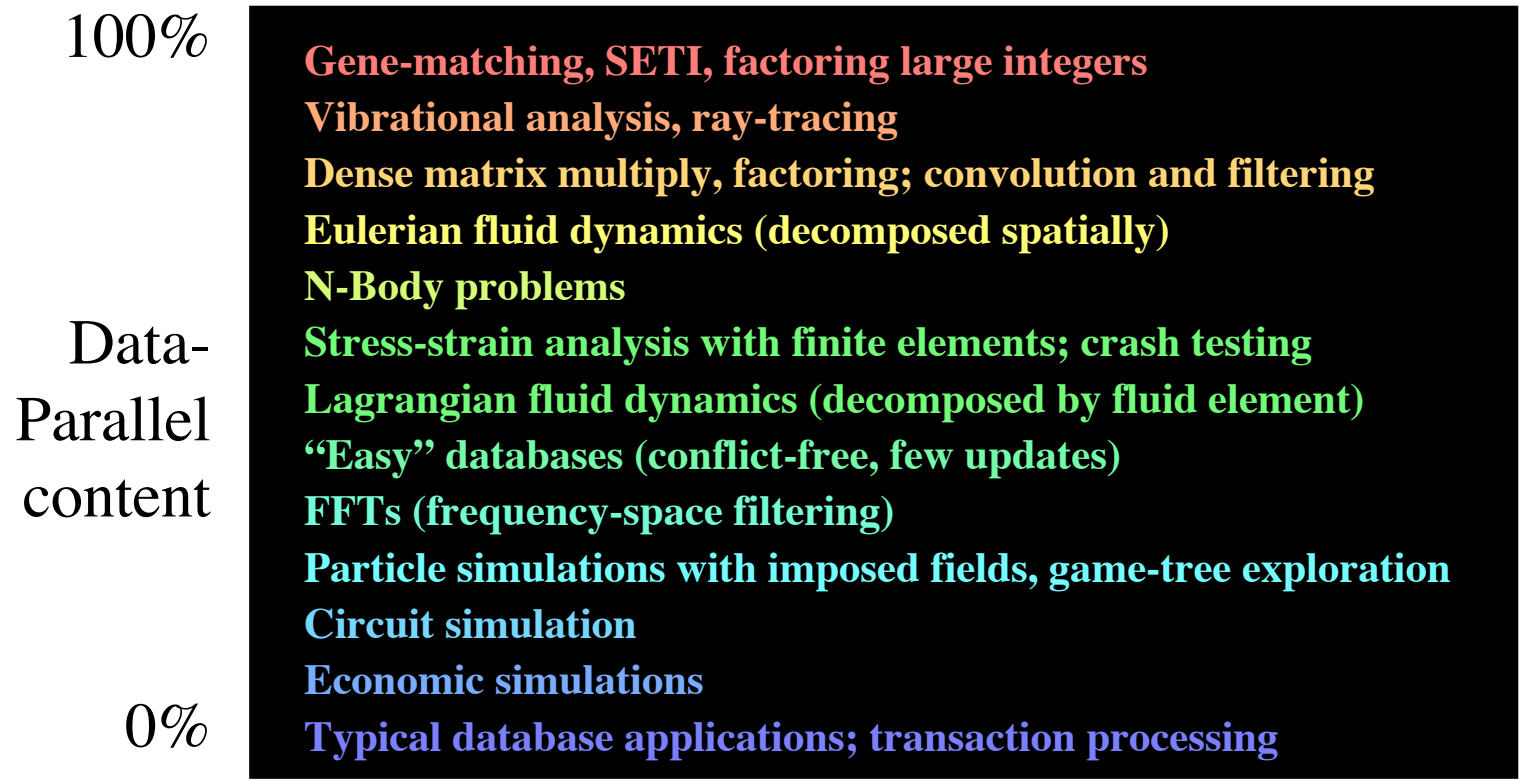
Electronic Design
Nuclear Applications
Mechanical Design
Mechanical Engineering
Radar Cross-Section
Crash Simulation
Fluid Dynamics
Weather, Climate modeling
Signal/Image Processing
Encryption
Life Sciences
Financial Modeling
Petroleum

For each of these, we seek to match

- Problem size
- Data locality
- Predominate data type
- Dynamic behavior in time
- Spatial irregularity
- Demands on I/O

To avoid the "toy benchmark" problem.

Example of Workload Dimension: Data Parallelism (estimated)



Purpose-Based Benchmarks

- A *purpose-based* benchmark states an objective function that has direct interest to humans.
- An *activity-based* benchmark states computer operations to be performed, usually defined by source code.

Either can be made to scale in multiple dimensions, but it's harder to do with activity-based benchmarks.

2-Way Benchmark Taxonomy

	Fixed-Size	*Scalable*
Activity-Based	**LINPACK** **NAS Parallel** **SPEC (any year)**	**LINPEAK** **Streams** **TableToy**
Purpose-Based	**NSA suites** **Many RFP tests**	**TPC-*x*, ECPerf** **Sun's PBB Suite**

HPC Benchmarks-I

HPC Segment	Task	
Electronic Design	**Design/test an *N*-bit adder**	***Clock speed, area***
Nuclear Applications	**Optimize radiation transfer**	***Evenness of radiation***
Mechanical Engineering	**Design truss to bear given load**	***Minimum weight of truss***
Crash Simulation	**Minimize deformation of elastic body past crush zone**	***Minimum deformation***

HPC Benchmarks-II

HPC Segment	Task	Objective
Fluid Dynamics	**For shape-constrained vehicle, minimize drag coefficient**	***Minimal air-drag***
Weather, Climate modeling	**Predict weather for *N* days**	***Minimal error in forecast***
Life Sciences	**Fold chain of *N* peptides in a given amount of time**	***Largest N (closest to real protein)***
Financial Modeling	**Predict and act on actual stream of financial events**	***Profit***
Petroleum	**Manage petroleum reservoir**	***Total petroleum extracted***

Innovations in Hero

- Memory bandwidth. No tricks, no apologies.
  - Uniform, low-latency petascale memory matched to floating-point speed
  - Full speed no matter where operands and processes reside
  - 100x historical I/O speeds relative to flops allow real-time, embedded uses
  - Checkpoint/restart can be done in under 5 seconds
  - Removes pressure on programmers to be “clever,” improving productivity

Innovations in Hero

- Portable Object Code and Run-Time Environment
  - New, industry-standard (open) high-level HPC model
  - Zero porting cost; tuning burden entirely on system instead of programmer
  - Legacy languages, codes, and parallel models fully supported
  - A collection of innovations assure *correctness* and *clarity*.

Innovations in Hero

- Hardware protection helps security, debugging, admin, OS scaling
- Interval Arithmetic assures validity and provides parallelism
  - Makes uncertainty in floating-point math visible and controllable
  - Enables new, massively parallel approaches for nonlinear optimization
- Purpose-Based Benchmarks measure productivity explicitly

SUMMARY

- Sun will contribute several disruptive technologies to HPC in the coming decade:
  - Asynchronous computing and new chip technology
  - Higher-quality floating-point computation
  - Much better memory-to-flops ratios
- We will take pressure off programmers by providing a simpler, more capable model.
- Sun will provide more meaningful benchmarks and share them with the HPC community. ■