The Problem of Application Software

Paul C. Muzio

Director, CUNY HPC Center

paul.muzio@csi.cuny.edu

Comments expressed in this presentation are those of the author and do not necessarily reflect the position of the College of Staten Island or the City University of New York

Discussion Points

- Software development cycle
- Programming models and computing architectures
- Economic priorities
- Government policies/priorities
- Educational system
- What can we do? (WCWD)

Software Development Cycle

“An application engineer’s ideal petascale supercomputer would have one powerful processor with direct access to main memory. By contrast, a computer scientist’s ideal petascale system would have hundreds of thousands of processors of different types and an innovative, distributed memory architecture.”

“Unfortunately for many users,the computer scientist’s system may be the one that’s built in the near future, as the technology to do otherwise does not exist. The challenge will be to build this kind of system but make it look like the kind the applications the software engineer wants.”

Software Development Cycle

- Years – long term commitment/long term investment
- Focused/dedicated effort
- Domain expertise (not computer scientists)
- “Cottage industry”
- Computing platform/programming model stability
- Examples:
  - DYNA
  - EPIC
  - GAUSSIAN
  - NASTRAN
- The Golden Days: 1960s thru 1980s

Programming Models/Architectures

- System manufacturers
  - IBM and the BUNCH (and Digital, Cray, SGI, Thinking Machines, Denelcor, etc, etc)
    - All had their own system level compiler development teams
    - By 2005, the systems manufacturers that remained, for the most part, no longer had compiler groups
  - Commoditization of the market has reduced profit margins to the level that compiler groups are not affordable
  - Compiler design no longer taught in universities

Programming Models/Architectures

The golden days

- Fortran and serial computers
  - 1956 thru 1980s
- Fortran and vector computers
  - 1976 thru early 1990s
- Evolution not revolution
- Corollary: Vectorization became successful not just because of the increase in performance, but because the compiler could do most of the work for you!
- Ocam's Razor

The Brave New World

- Autotasking and OpenMP
- CM Fortran (1990) and HPF (1993) for SIMD computing
- PVM (1989)
- MPI (1994)
- MPI-2 (1996)
- shmem (1994)
- Partitioned Global Address Space Programming Models
  - Co-Array Fortran (1998)
  - Unified Parallel C
- OpenMPI
- CUDA
- OpenCL (???)
- Corollary: How many times do you want to rewrite your code?
- "It's a Mad, Mad, Mad World"

“We certify it to be cluster ready”

Cluster Ready is an architecture and program that makes it easier to gain the performance advantages of HPC clusters

Programming Models/Architectures

- Historically, we had one programming model for over 30 years, now over 7 different programming models in less than 20 years
- New HPC architectures may require software developers to use two or three different programming models
- What happens to my investment (human and $), if I select the wrong programming model?
- Validation and verification
  - Against what configuration(s)?
  - How many different systems do I need to support?
- Now, I have to spend 80% of my time worrying about coding techniques and with only 20% of my time left for science!!
- Rewriting software is sideways progress
- Where do I get the talent?

WCWD-Programming Models/Architectures

- We need system level compilers, not chip level compilers
  - Applications run on systems not just on chips
- Component manufacturers
  - Need to take a system level perspective, not just sub-system (because the system level providers can't afford to do it and are limited by the component technology)
    - Look at the Apple model, not the PC model
    - Forget the "cluster ready" marketing hype
    - But HPC is not the market driver, it's the fringe
    - But understand that the issues facing HPC today will affect the commodity PC and server market tomorrow

WCWD-Economic priorities

- Subsidies to systems providers/compiler companies for advanced system compiler development?
- Positive aspects
  - Re-investment in compiler development as part of the HPCS Program
    - Cray and IBM system level Co-Array Fortran and UPC compilers
    - How about some Government support for SGI compiler development?
    - Will Government support for this continue?
  - Intel's planned compiler support for Co-Array Fortran (Fortran 2008)
    - Is half a loaf better than none?
    - Intel - How about on-the-chip hardware address translation so that we can again think system-wide (at least for memory accesses)?
  - Portland Group's PGI Accelerator Programming Model
    - Better than CUDA and OpenCL
      - Directives and portability

WCWD-Programming Models/Architectures

- Chip level support for global addressing
  - Can't make the processors appear monolithic, but can make memory look shared
    - Provide hardware level support for Global Address Space Program Models such as Co-Array Fortran (CAF) and Unified Parallel C (UPC)
    - CAF and UPC are much easier programming models than MPI
    - Evolutionary extensions to existing languages (Fortran, C, C++)
    - CAF and UPC can support compiler optimization for communications
    - Better debugging tools because the compiler has a global view

Software Development Economics

- Industry invested in R&D and application software development
  - Seven airframe manufactures, now down to two
    - IRAD
  - Automobile companies?
  - Who are the new investors?
- In the past, a few Government agencies developed new, useful and innovative software
  - Missions with focused problems to solve
- Mission drift
    - Questions on NASA's mission in space (and even when it was clear, NASTRAN was externally developed)
    - DOE is not building nuclear weapons
    - Government agencies, in general, have a abysmal record for software development
    - Do people get fired for not meeting schedule/performance specs?
- Pleased to see NASA and DoD's focused efforts

Government priorities

- Petascale and Exascale
  - Where are the real-world applications?
  - Fill up the machine so we can justify a bigger one!!
- High Productivity Computing Systems
  - What happened to the "Productivity"?

- Petascale projects on Blue Waters
  - 6 - Astrophysics and turbulence
  - 1 – QCD
  - 2 – Computer science
  - 1 - Monte Carlo simulations
  - 5 - Chemistry/biology
  - 2 - Climate/weather
  - 2 – Earthquakes

WCWD-Government-Action Items

- Funding for software development of **<u>“critical applications”</u>** (not fluff applications)
- Defined, focused objective
- Hands-off approach
  - Define the need, not the methodology or the technology
  - Provide validation and verification data, if available
  - Provide access to system resources
- Partial funding
  - Developer must take partial risk
  - Competitive
  - Support multiple, competing efforts
  - Finite and defined period for development
- Bonus to best delivered product
- Companies retain IP
- Investment tax credits to spur R&D
  - Application software development
  - Compiler development

Educational System

- University departments are stove-piped
- Updating of curriculum takes years
- Computer science
  - Design the hammer, but don’t build anything
- Engineering and science curricula
  - Emphasis on theory, not problem solving
  - Not enough emphasis on computations in course work
- Universities/faculty need to recognize that computations is the third leg of research
  - You can not get a Ph. D. for writing software

WCWD-Educational System

- Change the undergraduate curriculum
  - Require engineering and science students to take courses in
    - Numerical algorithms
    - Computer architectures
    - Programming techniques for scientific computing
  - Require computer science students to take courses in
    - Numerical algorithms
    - Science and engineering
- Encourage interdisciplinary studies
- Recognize that software is as valuable as a research paper
- NSF continuing efforts to foster change
  - A big plus

Thank you

Questions?