# Software Challenges for Extreme-Scale Computing

William Gropp

www.cs.illinois.edu/~wgropp

## Quotes from "Enabling Technologies for Petaflops Computing" (MIT Press 1995)

- "**The software for the current generation of 100 GF machines is not adequate to be scaled to a TF…**"
- "The Petaflops computer is achievable at reasonable cost with technology available in about 20 years [2014]."
  - (estimated clock speed in 2004 — 700MHz)*
- "Software technology for MPP's must evolve new ways to design software that is portable across a wide variety of computer architectures. Only then can the small but important MPP sector of the computer hardware market leverage the massive investment that is being applied to commercial software for the business and commodity computer market."
- "To address the inadequate state of software productivity, there is a need to develop language systems able to integrate software components that use different paradigms and language dialects."

## Quotes from “Enabling Technologies for Petaflops Computing” (MIT Press 1995)

- “The software for the current generation of 100 GF machines is not adequate to be scaled to a TF…”
- “The Petaflops computer is achievable at reasonable cost with technology available in about 20 years [2014].”
  - (estimated clock speed in 2004 — 700MHz)*
- “Software technology for MPP’s must evolve new ways to design software that is portable across a wide variety of computer architectures.  Only then can the small but important MPP sector of the computer hardware market leverage the massive investment that is being applied to commercial software for the business and commodity computer market.”
- “**To address the inadequate state of software productivity, there is a need to develop language systems able to integrate software components that use different paradigms and language dialects.**”

## Top Three Challenges

- Algorithms for Extreme Scale
  - ♦ Must match/work with features of the architecture:
    - Heterogeneous, “vectors” (optimized hardware)
    - Latency hiding; communication/computation overlap
    - Small memory per “core”
    - High memory locality (in part to reduce energy use of algorithm)
    - Large degree of concurrency (~$10^6$ “cores”)
    - Resilience
  - ♦ Algorithm changes; e.g., higher-order approximations in numerical codes; more compact representations in general
  - ♦ Current software systems provide little support/help for programming these algorithms
    - ♦ Esp locality, performance, resilience

# Top Three Challenges

- Software for realizing algorithms
  - ♦ Must avoid the false choice of a single approach for everyone
    - Wrong on two levels – mismatch with skill and with needs
  - ♦ Low Level: Must allow skilled programmers to realize full potential of machine
    - These programmers need help; better tools to analyze performance and correctness; performance as a first-class property in the language/runtime
  - ♦ Higher Levels: Must provide higher levels of abstraction, <u>even at the cost of generality</u>

## Top Three Challenges

- An execution model that provides access to performance, correctness, reliability, and composability
  - ♦ And to performance (else get a smaller, simpler machine)
  - ♦ Composability to allow multiple software components to work *efficiently* together

# Are We Up to These Challenges?

- Those are three big challenges
- How much needs to change?
  - ♦ Algorithms
    - More dynamic, less BSP, more latency tolerant
    - Bonus: Also better for current machines
  - ♦ Execution models
    - Hardest part; depends on technology choices and implementation
    - To start:
      - Different kinds of parallelism (vector, SMT, core, chip, node)
      - Different kinds of memory (register, cache, shared, SRAM, DRAM, NVRAM)
      - I/O operations and semantics (POSIX will be a huge mistake here)
  - ♦ Software for programming
    - Actually the easiest
    - Plan A: New, general purpose, high productivity programming language and environment

# Plan B

- MPI + OpenMP + …
  - ♦ UPC, CAF, etc.
    - Exploit hierarchy to handle level of concurrency
    - E.g., UPC program is MPI "process"
  - ♦ Languages/models with dynamic concurrency
    - Provide adaptive load balance, latency hiding
  - ♦ MPI 3 extensions
    - "I don't know what it will be, but it will be called ~~Fortran~~ MPI"
- Algorithms are enhancements of current approaches
  - ♦ Adds hierarchy to avoid fine-grain data decomposition and larger numbers of MPI processes
  - ♦ Adds non-blocking collectives, RMA for latency hiding
  - ♦ Uses user-directed, in-memory encoded checkpoints for resiliency
- Higher level programming models through community-driven, domain (really algorithm)-specific tools

## Where Is MPI Today?

- Applications already running at large scale:

| System | Cores |
|---|---|
| Tianhe-2 | 3,120,000 (most in Phi) |
| Sequoia | 1,572,864 |
| Blue Waters | 792,064* + 59,136smx |
| Mira | 786,432 |
| K computer | 705,024 |
| JUQUEEN | 458,752 |
| Titan | 299,008* + 261,632smx |

* 2 cores share a wide FP unit

# MPI is not a BSP system

- BSP = Bulk Synchronous Programming
  - ♦ Programmers **like** the BSP model, adopting it even when not necessary (see "functionally irrelevant barriers")
  - ♦ Unlike most programming models, *designed* with a performance model to encourage *quantitative* design in programs
- MPI makes it easy to *emulate* a BSP system
  - ♦ Rich set of collectives, barriers, blocking operations
- MPI (even MPI-1) sufficient for dynamic adaptive programming
  - ♦ The main issues are performance and "progress"
  - ♦ Improving implementations and better HW support for integrated CPU/NIC coordination is the right answer

## Plan B and Algorithms

| Issue | Programming Support | Options |
|---|---|---|
| Heterogeneous processing elements | "Domain" specific languages, annotations for composition | DSL, s2s, Open{X}, … |
| Latency hiding/ overlap | Virtual tasks; non-blocking communication | MPI, OpenMP4, Charm++, … |
| Small memory/core | RMA features (avoid copies) | MPI, PGAS,… |
| Memory locality | Hierarchical models; explicit locality management | MPI+X, DSL, s2s, … |
| Concurrency | Hierarchical models | MPI+X+Y |
| Resilience | Hierarchical models | ? |

# Observations

- MPI can work on extreme (and Exascale) systems
  - ♦ Current MPI *implementations* will need some changes
  - ♦ Productivity limitations related mostly to lack of distributed data structure support
    - Fixing this doesn't require a new programming model
  - ♦ Real issues include library overhead, cost of abstraction, static partitioning/degree of concurrency
- Quest for a single universal software solution is the *single biggest reason that we have the software crisis* (at least in HPC)
- Algorithms need at least as much attention!