Invited Presentation to:
CCSGC 2010

EXAPHOBIA

Thomas Sterling

Arnaud & Edwards Professor, Department of Computer Science

Adjunct Faculty, Department of Electrical and Computer Engineering

Louisiana State University

Distinguished Visiting Scientist, Oak Ridge National Laboratory

CSRI Fellow, Sandia National Laboratories

September 8, 2010

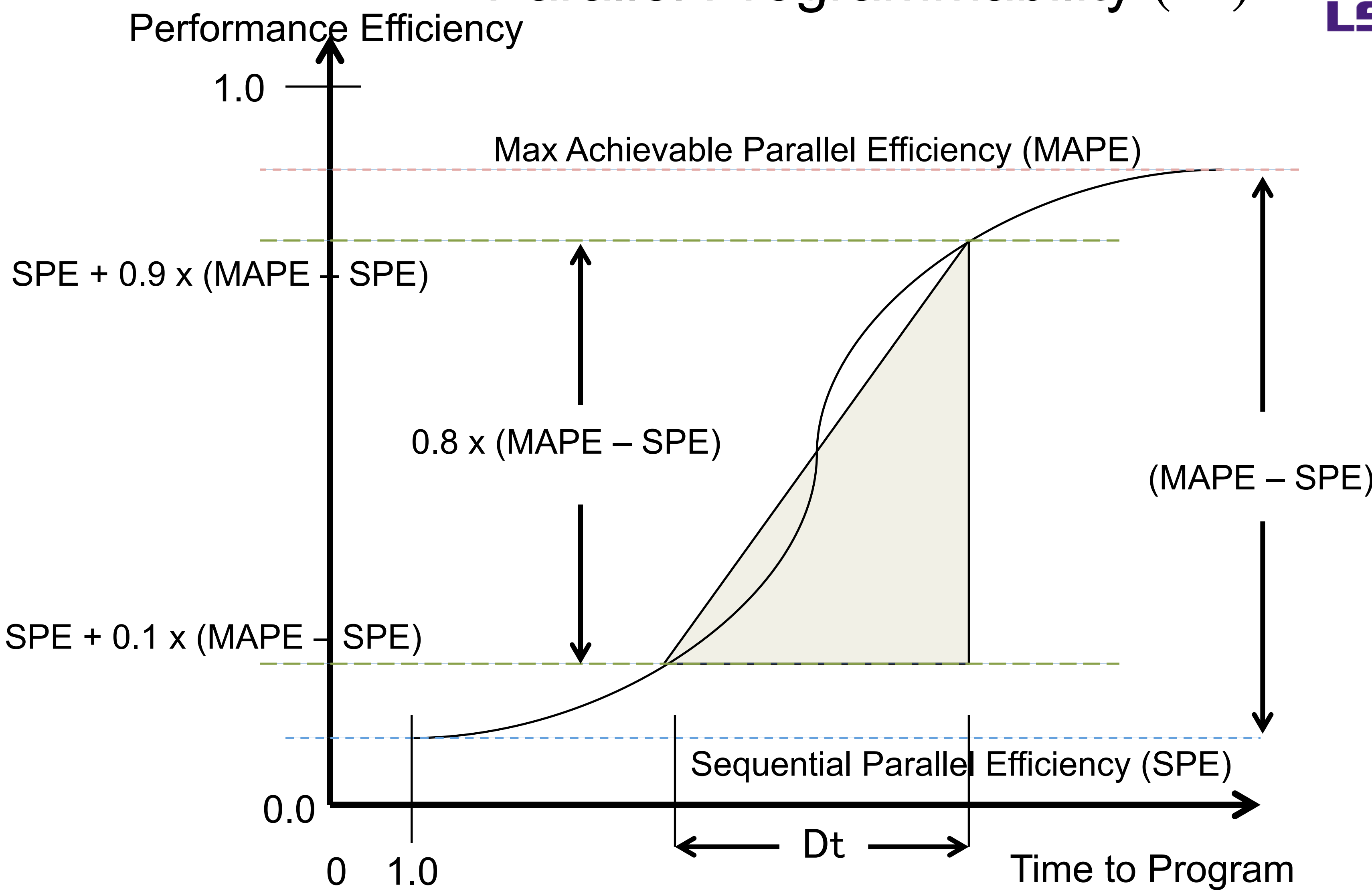
Fear and Loathing in 21st Century HPC

- Disagreement about need for revolution
  - Either – it won't work without new paradigm, i.e. change everything
  - Or – you guys said that about Pflops; how's that working out for you?
- Vendors:
  - Don't worry your silly little head about this, its under control, or
  - Users tell us what you want, as long as its commodity
- Application programmers:
  - We hate what we have, but
  - Don't change anything, cuz we got legacy
- Mission critical Agencies:
  - (1) We know we need long term research, but
  - (2) Since we didn't do that before, we can only fund short term work to catch up
  - (3) Repeat step 1

Quantifying Challenges

- Time to Completion
  - Strong scaling
  - Seconds
- Parallelism
  - Addresses Starvation
  - Machine & software
  - Granularity, average, variance, …
- Latency
  - Average distances for access and services (cycles, nanoseconds)
- Overhead
  - Extra critical-path work for managing concurrency (cycles, nanoseconds)
- (Waiting for) Contention
  - Waiting time in a queue for service request
- Energy
  - Joules
  - Ops, data movement, memory access
- MTTI
  - Seconds, mean/variance
- Programmability
  - I feel your pain

Parallel Programmability (Dt)



The only thing we have to change is change itself

- Architecture
  - Microprocessor cores
  - Memory hierarchy
- Programming models
  - Languages
  - Legacy codes
- Operating systems
  - Manage massive resources
  - Lightweight runtime systems

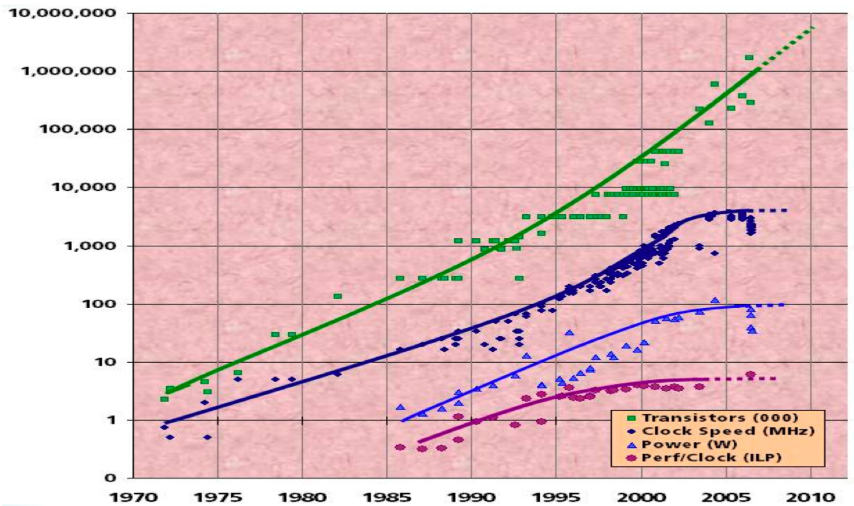


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Changing Change

- Phase I: Sequential instruction execution

- Phase II: Sequential instruction issue
  - pipeline execution,
  - reservation stations,
  - ILP

- Phase III: Vector
  - pipelined arithmetic, registers, memory access
  - Cray

- Phase IV: SIMD
  - MasPar, CM-2
- Phase V: Communicating Sequential Processes
  - MPP, clusters
  - MPI

Co-Design

- Objective
  - Optimal system design and operation
  - Modulated by workload characteristics
- Challenge
  - Division of roles and responsibilities across system layers
  - Efficiency
    - Minimize overhead, starvation, and latency effects
    - User productivity
    - Energy, reliability
- Methodology
  - Each system layer tuned with respect to needs of the others
  - Execution model as operational and design paradigm
    - for governing principles of operation
    - Interoperability among layers
    - Reduction of design complexity

The Execution Model Imperative

- HPC in 6th Phase Change
  - Driven by technology opportunities and challenges
  - Historically, catalyzed by paradigm shift
- Guiding principles for governing system design and operation
  - Semantics, Mechanisms, Policies, Parameters, Metrics
- Enables holistic reasoning about concepts and tradeoffs
  - Serves for Exascale the role of *von Neumann architecture* for sequential
- Essential for co-design of all system layers
  - Architecture, runtime and operating system, programming models
  - Reduces design complexity from O(N2) to O(N)
- Empowers discrimination, commonality, portability
  - Establishes a phylum of UHPC class systems
- Decision chain
  - For reasoning towards optimization of design and operation

Decision Chain

- Axiom: an operation is performed at a certain place at a certain time to achieve a specified effect
- How did this happen?
- Every layer of the system contributed to the tim /function event – the *decision chain*

- A program execution comprises the ensemble of such events across the system space and throughout the execution epoch
- There are many such paths that lead to a final result
- But not all minimize time and energy
- Understanding of the decision chain required for optimization
- Execution model required for understanding the decision chain

Conclusions – A Convergence?

- Global Address Space
  - PGAS
  - AGAS
- User multithreaded
  - Lightweight
  - Dynamic scheduling
  - Complexes
  - Codelets
- Message-driven
  - Active messages
  - Parcels
- Diversity of lightweight synchronization
  - Local control objects
- Runtime Software as key stack component
  - Support dynamic resource & task management

Panel Background

Is there a fear of Exascale? There is concern about faults, scaling, performance (latency), complex processing model (e.g., heterogeneous elements), cost (power, memory, $), impact on algorithms, narrowness of application domain, data handling for Exabyte data sets. In addition, does Exascale imply a discontinuity in programming, algorithms, debugging, etc.?

Panel Questions

- How can we overcome the fear?
  - Which fears are mistaken (after all, many were convinced that petascale systems would be impossible without new programming models)?
  - Conversely, which problems apply at a smaller scale, and hence can be addressed now and provide near-term benefits?
  - Which problems are (nearly) unique to Exascale? How do we build/test /improve algorithms, software, and applications? For example, do we need to build a much more sophisticated simulation environment?
- How can we build real excitement?
  - How do we provide evidence that Exascale systems will work well with applications?
  - How do we demonstrate that Exascale systems can enable new application areas (after all, Exascale systems may be greatly different in architecture - will that be a virture)?
- In all of the above, how do we move past qualitative statements to quantitative predictions?