7th IESP Meeting Working Group on:

Revolutionary Approaches

Punctuated Equilibrium of Continuous Evolution

Moderated by:

Thomas Sterling, IU

Bronis R. de Supinski, LLNL

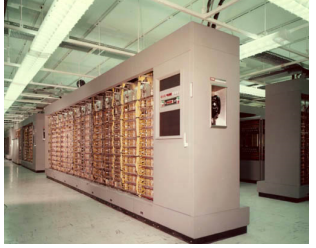
October 7, 2011

Revolutionaries

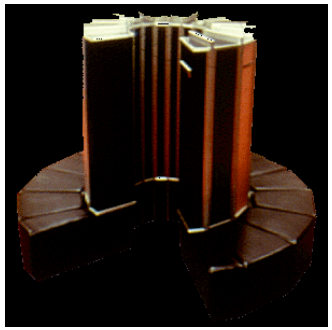
- Pete Beckman, ANL
- Ron Brightwell, SNL
- Barbara Chapman, UH
- Jack Dongarra, UTK
- Anshu Dubey, UofC
- Al Geist, ORNL
- Andrew Jones, NAG
- Alice Koniges, LBL
- Jesus Labarta, BSC
- Bob Lucas, USC ISI
- Paul Messina, ANL …
- Hiroshi Nakamura, UT
- Hiroshi Nakashimi, KU
- Thomas Sterling, IU
- Shinji Sumimoto, Fujitsu
- Bronis de Supinski, LLNL
- Kenjiro Taura, UT
- Rajeev Thaker, ANL
- Anne Trefethan,
- Vladimir Voevodin, MSU

Success through Incremental Innovation

A History of Revolution in HPC



- Phase I: Sequential instruction execution (1950)
- Phase II: Sequential instruction issue (1965)
  - pipeline execution,
  - reservation stations,
  - ILP

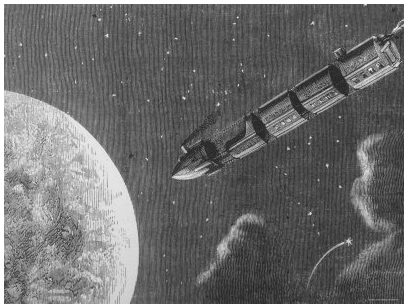


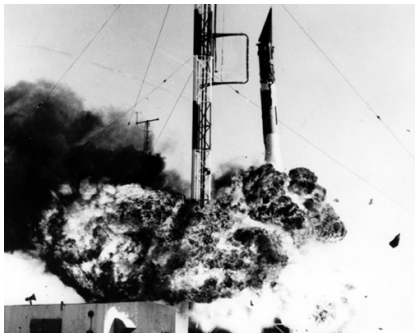
- Phase III: Vector (1975)
  - pipelined arithmetic, registers, memory access
  - Cray

- Phase IV: SIMD (1985)
  - MasPar, CM-2
- Phase V: Communicating Sequential Processes (1990)
  - MPP, clusters
  - MPI, PVM

Issues/Questions

1. Socialization of Change - Controversy
   a. Space Cadets: Dreaming, disconnected, “to infinity and beyond”
   b. Bricklayers: Fear, complacency, and loathing
2. Quantum Leap through Punctuated Equilibrium
   a. Exascale challenges that may require revolutionary approaches
   b. Potential distinct advantages/opportunities and constructive impact of revolution in Exascale
3. Candidate areas for revolution
   a. Hardware, software, algorithms
   b. Execution models & Paradigms
4. Disruption effects
   a. Nature & dangers of disruption
   b. Mitigating disruption of revolutionary methods





Challenges that may need Revolution

- Parallelism
  - Single thread performance
- Dependability
- Debugging
  - Reliability/resilience/fault tolerance (at all levels)
- Algorithmic changes
  - Need for partitioners; reordering is currently a sequential bottleneck
  - Mathematical bottlenecks (new math algorithms)
- Synchrony Synchronization/global coordination/within application scheduling
  - Asychronous versus Synchronous execution
- Productivity
- Complexity of software and hardware
- Exposure of details of hardware
- Treating software as a deterministic system
- Compiler needs changes
  - Currently does not explicitly reason about parallelism
  - How to understand, to express and to exploit locality
- Variability
- Inflexibility in system design

Opportunities of Possible Revolution (1)

- Execution model
  - Addresses: Starvation, Latency, Overhead, and Waiting for resolution of contention
  - Impacts co-design across programming models, architectures,
  - Not clear what right one is, some people don't know what it is (informational graph)
  - Come from semantics of application information graph, not just hardware side
- Changing the way we think about system and programming merger
  - Dynamic behavior including aspects of non-determinacy, variability
  - Can't control the *horizontal* or *vertical* – we don't control the machine
  - Change of culture, attitude, not just bottom up – impacts algorithms
- Intelligence
  - Decision making at multiple levels, the decision chain
  - On the fly, control choices and data analysis
  - Spatially aware from data perspective requires hardware support for operation knowledge, predictive
  - Information backplane, FT backplane – protocol between system layers for mutual dynamic introspective behavior
- OS
  - Optimization point for programming model and the architecture
  - Reactive: Driven by revolutions in other areas, can't keep old OS w new architecture

Opportunities of Possible Revolution (2)

- Programming models
  - Levels of abstraction
    - Where are we falling down?
      - Inertia of comfort zone, familiarity breads complacency
    - Semantics of parallelism
      - Control of sequencing (or not)
      - Forms of parallelism, synchronization constructs
    - Relationship of data structure and control
      - One sided access
      - Dependency for automatic control and data move
  - MPI
    - provides locality control, permits human intervention of placement and movement of data
    - Ease of reasoning about program execution (or not)
    - Legacy of codes and skill sets
    - Don't want to know where everything is – suffering from tsunami of complexity
  - Intermix programming/execution modes
    - "Right tool for the right job"
- Correctness and debugging
  - Tool for verification and boundedness for quasi reproducability
  - Program itself knows that it is correct – how does it know what correct is?

Final Comments

- Managing revolution
  - Gradual steps, Otherwise cannot plan, Is there a concept of an “incremental revolution”
  - Starting with something credible
  - Don’t throw out everything
    - Some changes but …
  - Something has to die: Ron Brightwell
  - Education and culture change
  - Revolution is being imposed, even for next generation machines
    - Libraries may mitigate transition
      - If you can read this you’re too close!
    - Repeated changes to architecture generations
- Can we identify destination before choosing the road
  - Key is to determine the right answer and then worry about how we get there…
    - Often can take a quantum leap or a gradual path to the same destination
- Where do we put intelligence in the systems
  - Revolutionizing the libraries
  - Changing algorithms
- Worry about reliance on compiler technology
  - Abstractions could be too high, doesn’t convey to scientific computing
- Persistent storage – a hierarchy that is likely to change; the revolution is coming
  - Out of core calculations

QUESTIONS/COMMENTS?

- Disruptive to the application users or to the people in IESP or everybody?
- Key is to determine the right answer and then worry about how we get there…
  - Often can take a quantum leap or a gradual path to the same destination
- What is our goal to predict or to make change?
- How small does the space of applications have to go before support and funding dry up?
- Could the era of exascale look like the era of supersonic aircraft?
- Should we review the IESP roadmap to determine if we need revolutionary changes in each?
- Difficult to say what is and what is not a revolution

What are the problems we need to address

- Parallelism
- Debugging
- Single thread performance
- Algorithmic changes
  - Need for partitioners; reordering is currently a sequential bottleneck
  - Mathematical bottlenecks (new math algorithms)
- Reliability/resilience/fault tolerance (at all levels)
- Synchronization/global coordination/within application scheduling

More problems

- Productivity
- Exposure of details of hardware
- Asychronous execution
- Synchronous execution
- Complexity of software
- Treating software as a deterministic system
- Compiler needs changes
  - Currently does not explicitly reason about parallelism
  - How to understand, to express and to exploit locality

Still more problems

- Variability
- Inflexibility in system design

Potential changes/solutions

- Integrating parts of the system (e.g., OS and compiler runtime)
- Build on conventional systems
  - Need to communicate in some way; could potentially build solution on (hidden) MPI
- Optimize the interface between hardware and software
- Execution model
- Asychronous execution

More solutions

- Synchronous execution
- Need to express properties of algorithms
  - How to extract them
  - How to describe them
- Feedback from the compiler and runtime system
- Look ahead

The Execution Model Imperative

- HPC in the 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 $O(N^2)$ to $O(N)$
- Empowers discrimination, commonality, portability
  - Establishes a phylum of UHPC class systems
- Decision chain
  - For reasoning towards optimization of design and operation

Game Changer – Runtime System

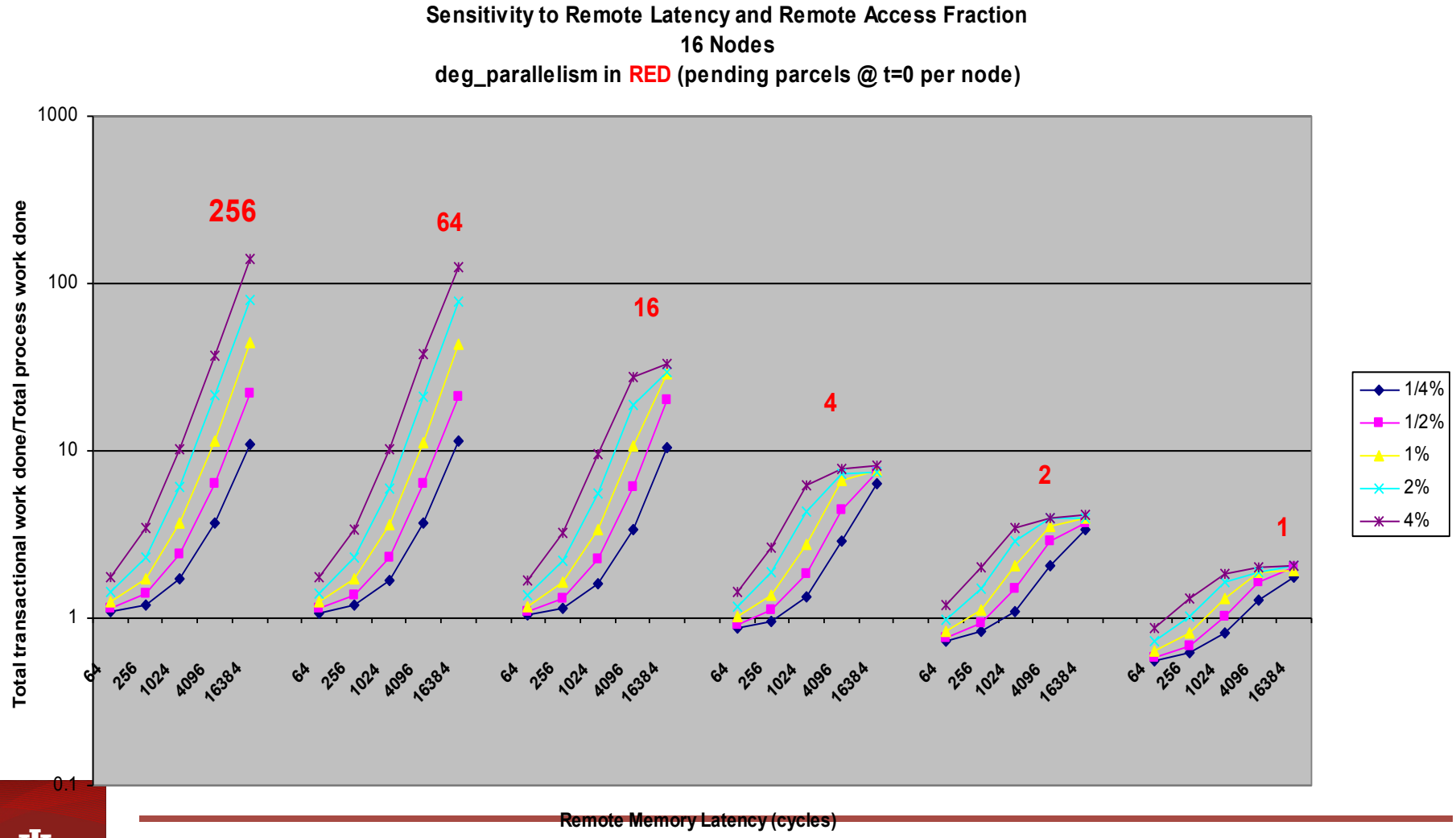
- Runtime system
  - is: ephemeral, dedicated to and exists only with an application
  - is not: the OS, persistent and dedicated to the hardware system
- Moves us from *static* to *dynamic* operational regime
  - Exploits situational awareness for causality-driven adaptation
  - Guided-missile with continuous course correction rather than a fired projectile with fixed-trajectory

- Based on foundational assumption
  - Untapped system resources to be harvested
  - More computational work will yield reduced time and lower power
  - Opportunities for enhanced efficiencies discovered only in flight
  - New methods of control to deliver superior scalability
- "Undiscovered Country" – adding a dimension of systematics
  - Adding a new component to the system stack
  - Path-finding through the new trade-off space

Concepts towards a new Paradigm

- Motivated by dynamic directed graphs
  - STEM
  - Knowledge management and understanding
- Split-phase transactions
  - Avoid blocking
- Message-driven computation
  - Move work to data
  - Parcels and Percolation
- Constraint-based synchronization
  - Declarative criteria for work
  - Event driven
- Data-directed execution
  - Merger of flow control and data structure
- Shared name space

Latency Hiding with Parcels
with respect to System Diameter in cycles



What is “Revolutionary”?

- Decoupled from conventional practices
- Exploitive of ideas (new?) not incorporated in traditional thinking
- Benefits from new opportunities that can only be exploited through non-typical means
- Dramatic change, with potentially disruptive consequences
- Inappropriate if near equivalent results may be realized through incremental progressions
- Essential if offers only viable path to achieving critical goals
- Controversial, risky, and unpopular
- May be less risky than ineffective application of common strategies

Strategic Challenges

- Performance
  - Efficiency
  - Scalability
- Energy
  - Bounded power
  - Minimized energy
- Reliability
  - Continued operation in the presence of faults
- Programmability
  - System transparency
  - Portability across system classes, scales, and generations
- Generality
  - STEM
  - Knowledge management and understanding

Performance Challenge Factors

- Starvation
  - Insufficiency of concurrency of work
  - Impacts scalability and latency hiding
  - Effects programmability
- Latency
  - Time measured distance for remote access and services
  - Impacts efficiency
- Overhead
  - Critical time additional work to manage tasks & resources
  - Impacts efficiency and granularity for scalability
- Waiting for contention resolution
  - Delays due to simultaneous access requests to shared physical or logical resources

Technology Demands new Response

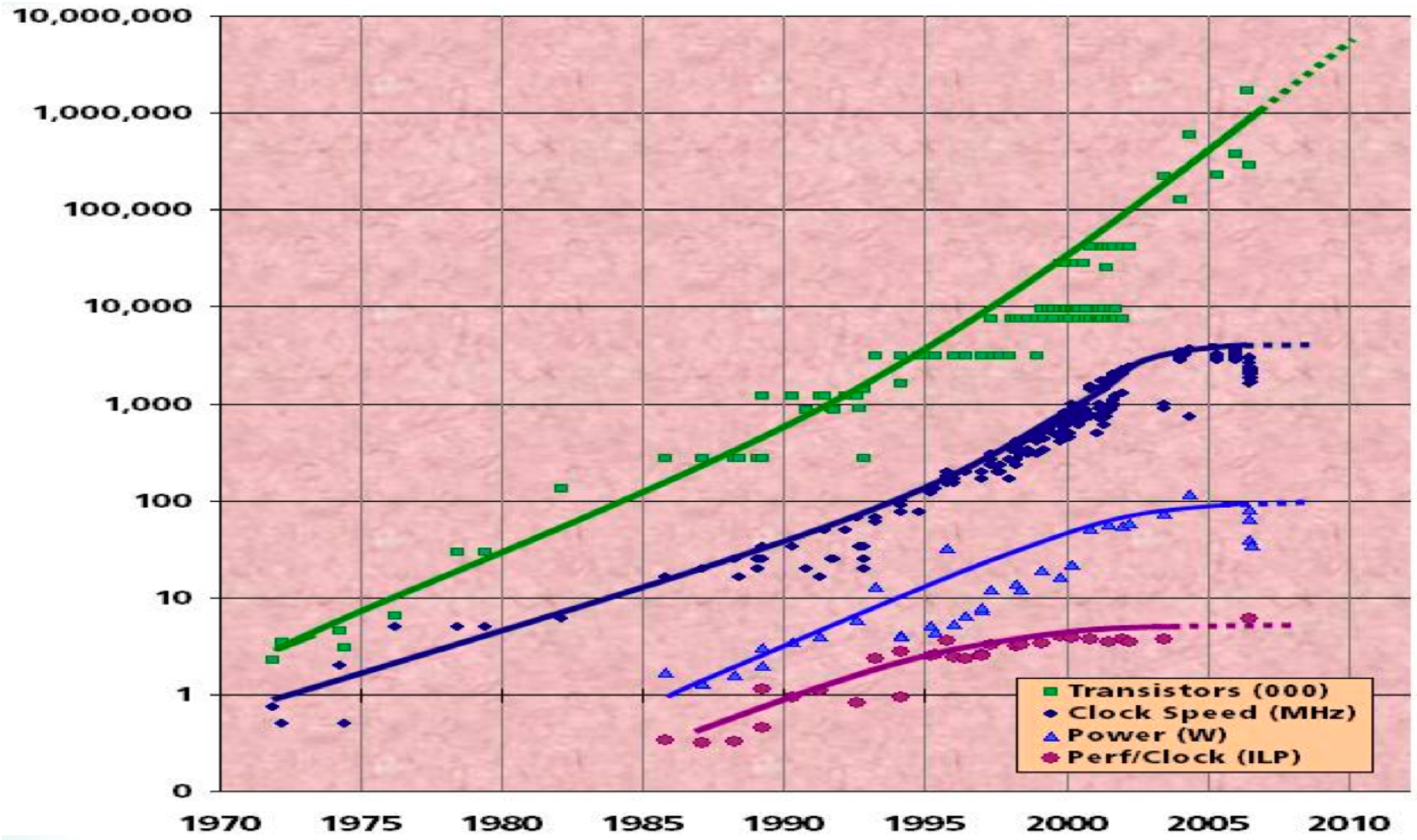


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith