
Kathy Yelick
Associate Laboratory Director for Computing Sciences
Lawrence Berkeley National Laboratory
and
Professor of Electrical Engineering and Computer Science
UC Berkeley
Why is Change Necessary?

1. **Lightweight cores** will have all/most of the system performance
   - Need fine-grained parallelism; avoid unnecessary synchronization
   - Cores not powerful enough for complex communication protocols?

2. **On-chip interconnect** offers opportunities for performance
   - New models of communication may be essential

3. **Hardware is heterogeneous**: no single ISA
   - Portability and performance portability are challenging

4. **New levels of memory hierarchy**, possibly software-controlled
   - Locality and communication-avoidance paramount

5. **Performance variability** may increase
   - Software or hardware control clock speeds

6. **Resilience** will be paramount at scale
   - Failures grow with the number of components and connections
Don’t Overstake Claims – Prove them!

THE SKY IS FALLING!!
The Trade-Off Is With Us

Locality

Load Balance
Target new applications, not rewriting old ones
  - The motivation is higher (all that code to be written)
  - The barrier to entry is lower (no code to throw away)

Target parts of applications, not full rewrites
  - Support for incremental adoption is key
  - Need to leverage existing libraries and parts of existing codes

Target new programming patterns
  - Irregular-in-space communication (random access to large data)
  - Irregular-in-time (one-sided access to shared state)
  - Irregular workload (dynamic load balancing)

Look for low hanging performance fruit
  - Do you rooflining homework
  - What’s the threshold? 15% - no, 15x - yes
Approach to Adoption

• Target new applications, not rewriting old ones
  – The motivation is higher (all that code to be written)
  – The barrier to entry is lower (no code to throw away)

• Target parts of applications
  – Support for incremental adoption is key
  – Need to leverage existing libraries and parts of existing codes

• Target new programming patterns
  – Irregular-in-space (random access to large data)
  – Irregular-in-time (one-sided access to shared state)
  – Irregular workload (dynamic load balancing)

• Look for low hanging performance fruit
  – Do you rooflining homework
  – What’s the threshold? 15% - no, 15x - yes

Work with applications people to understand what they need!
Application Challenges (Opportunities?)

Random Access to large state

Dynamic load balancing

Fusion of observation into simulation

Unstructured DAG algorithms

Speedup

Infinite

15-20%

Infinite

2x

Programming mode role: encourage a way of thinking
Lightweight mechanisms, flexible policies

- Atomic remote memory operations (compare\&swap,...)
- 1-way put + signal (2 addresses, data + increment)
- Software-managed memory as an *option*
- Fault detection and resilience job schedulers, etc.
- Transparent performance information
Give an application some optimizations and you feed it for a machine.

Teach a computer to generate optimized code for an application and you feed it for a lifetime
Thank you!
• What kind of programming models will be good for Exascale systems? Will it be evolutionary or revolutionary?
• Will MPI+X (together with PGAS and Task-based models) will be best suited for Exascale systems?
• Will newer programming models like HPC++, Legion, HPX, etc. gain more traction?
• How will the upcoming hardware trend (accelerators, coprocessors, high bandwidth memory, on-chip networking, etc.) have impact on the suitability of programming models?