Main Issues

• Increased parallelism
• Need for locality
• Heterogeneity
• Resilience
• Variability
• Virtualization
• Socialization
Managing 1B threads

• Increased parallelism
• Need for locality
• Heterogeneity
• Resilience
• Variability
• Virtualization
• Socialization
Scaling Applications

Weak scaling: use more powerful machine to solve larger problem
   – increase application size and keep running time constant; e.g., refine grid
• Larger problem may not be of interest
   – Identify problems that require petascale performance
• May want to scale time, not space (e.g., molecular dynamics)
   – Study parallelism in time domain
• Cannot scale space without scaling time (iterative methods): granularity decreases and communication increases
Scaling Iterative Methods

- Assume that number of cores (and compute power) increases by factor of $k$
- Space and time scales are refined by factor of $k^{1/4}$
- Mesh size increases by factor of $k^{3/4}$
- Per core cell **volume** decreases by factor of $k^{1/4}$
- Per core cell **area** decreases by a factor of $k^{1/4 \times 2/3} = k^{1/6}$
- **Area to volume ratio** (communication to computation ratio) **increases** by factor of $k^{1/4} / k^{1/6} = k^{1/12}$
- Per core computation is finer grained and needs relatively more communication
- (Per chip computation is coarser grained and and needs relatively less communication if most increase in # cores is per chip)
Debugging and Tuning: Observing 1B Threads

- Scalable infrastructure to control and instrument 1B threads
- On-the-fly sensor data stream mining to identify “anomalies”
- Need to ability to express “normality” (global correctness and performance assertions)
Locality

• Increased parallelism
• Need for locality
• Heterogeneity
• Resilience
• Variability
• Virtualization
• Socialization
It’s the Memory, Stupid

• CPU performance is determined, within 10%-20%, by trace of memory accesses [Snavely]

☞ Algorithm design should focus on data accesses, not operations
  – Temporal locality: cluster accesses in time
  – Spatial locality: match data storage to access order (*not vice-versa*); use partially-constrained iterators
  – Processor locality: cluster accesses in processor space
Heterogeneity

• Increased parallelism
• Need for locality
• Heterogeneity
• Resiliency
• Variability
• Virtualization
• Socialization
Hybrid Communication

- Multiple levels of caches and of cache sharing
- Different communication models intra and inter node
  - Coherent shared memory inside chip (node)
  - rDMA (put/get/update) across nodes
- Communication architecture changes every HW generation
- Need to easily adjust number of cores & replace inter-node communication with intra-node communication
- Easy to “downgrade” (use shared memory for message passing); hard to “upgrade”; hence tend to use lowest commonality (message passing)
- No good interoperability between shared memory (e.g., OpenMP) and message passing (MPI)
Do You Trust Your Results?

- Increased parallelism
- Need for locality
- Heterogeneity
- Resilience
- Variability
- Virtualization
- Socialization
Resilience

• Transient error are more frequent:
  – More transistors
  – Smaller transistors
  – Lower voltage
  – More manufacturing variance

• Error detection is expensive (e.g., nVidia vs. Power 7)

• Checkpoint/restart, as currently done, does not scale

• Need, new, more scalable error recovery algorithms

• Supercomputers built of low-cost commodity components may suffer from (too) high a rate of undetected errors.
  – Will need software error detection or fault-tolerant algorithms
Plus ça change, moins c’est la même chose

• Increased parallelism
• Need for locality
• Heterogeneity
• Resilience
• Variability
• Virtualization
• Socialization
Bulk Synchronous

• Many parallel applications are written in a “bulk-synchronous style”: alternating stages of local computation and global communication
• Models implicitly assume that all processes advance at the same compute speed
• Assumptions break down for an increasingly large number of reasons
  – Black swan effect
  – OS jitter
  – Application jitter
  – HW jitter
Jitter Causes

• Black swan effect
  – If each thread is unavailable (busy) for 1 msec once a month, than most collective communications involving 1B threads take > 1 msec

• OS jitter
  – Background OS activities (daemons, heartbeats...)

• HW jitter
  – Background error recovery activities (e.g., memory error correction, memory scrubbing, reexecution); power management; management of manufacturing variability; degraded operation modes

• Application jitter
  – Input-dependent variability in computation intensity

• Need to move away from bulk synchronous model
On the Need for Culture Change

• Increased parallelism
• Need for locality
• Heterogeneity
• Resilience
• Variability
• Virtualization
• Socialization
Big Systems Are Expensive

• 1% performance gain on a 4 week run = $100,000. Are we willing to invest a man-year to get it?
• Would we have our undergraduate students implement a major experiment at CERN?
• Major supercomputing application codes should be developed by professional teams that include specialized engineers – including a performance engineer and a SW architect
  – Incentives should encourage this model
Good Engineers Need Good Tools

Need integrated development environments
• Expert friendly tools for good engineers – a steep learning curve is necessary (no easy way to learn brain surgery)
• Analysis, debugging and performance tools are fully integrated in development environment at all levels of code creation/refactoring
  – correctness/performance information is presented in terms of programmer’s interface
  – compiler analysis and performance information available for refactoring
• Support a systematic methodology for performance debugging
  – Requires a performance model
• Will not come from industry – no market – but can leverage industrial infrastructure
• Performance programming can be made easier, but will never be easy – we have not automated bridge building, either
International Politics of Supercomputing

• An exascale system
  – Will cost ~$1B
  – Will consume 20-50MW
  – May use much less commodity technology than current supercomputers
  – May not have any military application

• Should supercomputing be done by international consortia?