Are their more appropriate domain-specific performance metrics for science and engineering HPC applications available then the canonical "percent of peak" or "parallel efficiency and scalability? If so, what are they? Are these metrics driving for weak or strong scaling or both?

- Value of HPL: Marketing
- Percentage of peak flops: Bad metric.
- Percentage of peak bandwidth: Better.
- Practical metric:
  - 8X performance improvement over previous system.

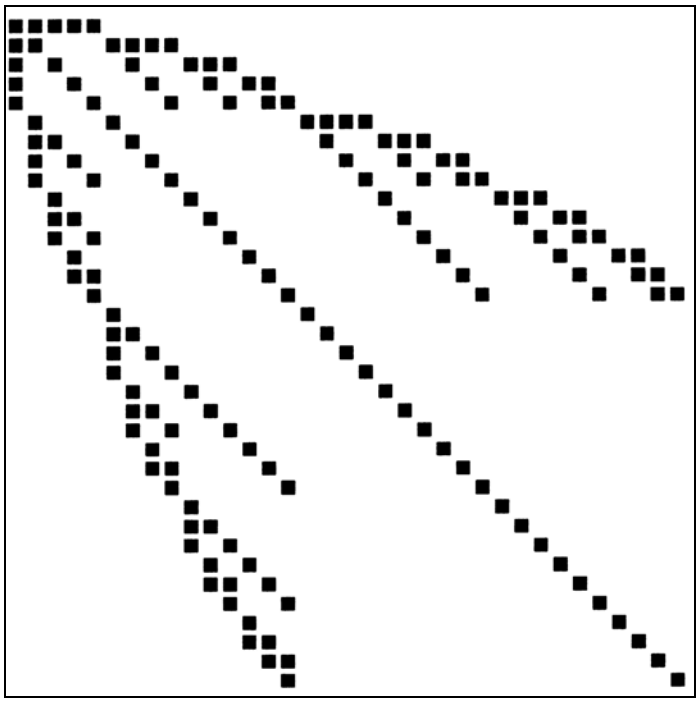
Michael A. Heroux Sandia National Laboratories

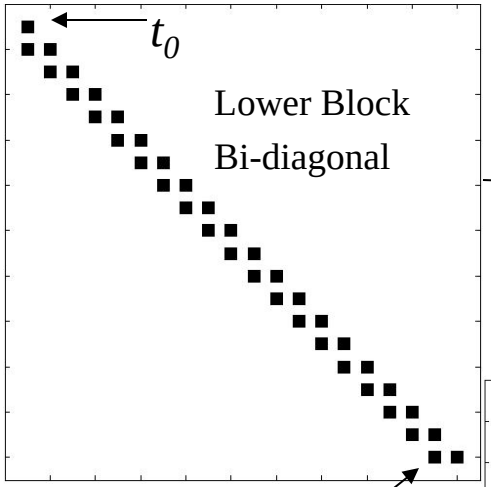
Similar to HPC "disruptive technologies" (memory, power, etc.) thought to be needed in many H/W roadmaps over the next decade to reach exascale, are their looming computational challenges (models, algorithms, S/W) whose resolution will be game changing or required over the next decade?

Advanced Modeling and Simulation Capabilities: Stability, Uncertainty and Optimization

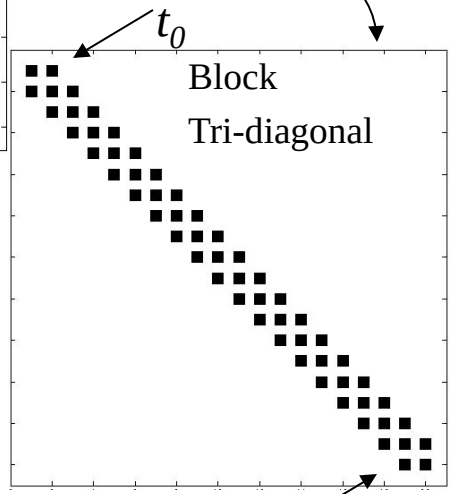
- Promise: 10-1000 times increase in parallelism (or more).

SPDEs:





Transient Optimization:



- Pre-requisite: High-fidelity "forward" solve:
  - Computing families of solutions to similar problems.
  - Differences in results must be meaningful.

■ - Size of a single forward problem

Advanced Capabilities: Derived Requirements

- Large-scale problem presents collections of related subproblems. Common theme: Increased efficiency.

- Linear Solvers: $Ax = b \rightarrow AX = B,\ Ax^i = b^i,\ A^i x^i = b^i$
  - Krylov methods for multiple RHS, related systems.

- Preconditioners: $A^i = A_0 + \Delta A^i$
  - Preconditioners for related systems.

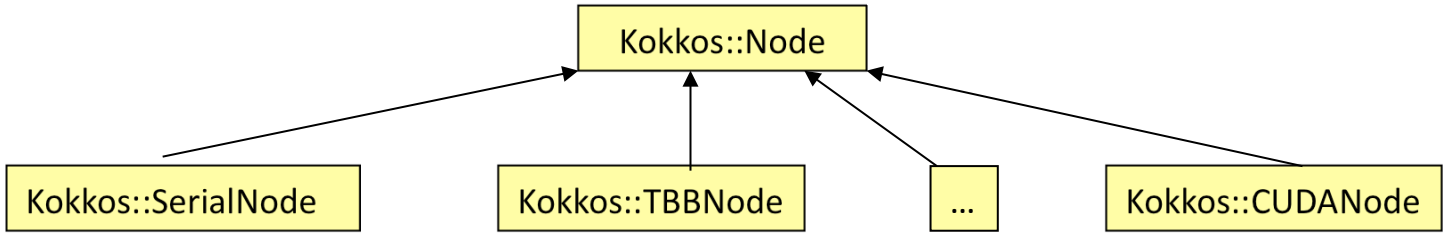
- Data structures/communication:
  - Substantial graph data reuse. $pattern(A^i) = pattern(A^j)$

What is the role of local (node-based) floating point accelerators (e.g., cell, GPUs, etc.) for key science and engineering applications in the next 3-5 years? Is there unexploited or unrealized concurrency in the applications you are familiar with? If so, what and where is it?

- Single GTX285 ≈ Dual socket quad core.
  - Sparse, unstructured, implicit
  - GPU data: device-resident (not realistic).
- Unexploited parallelism?
  - Yes, a lot.
  - 2-level parallel required to extract it all.

Should applications continue with current programming models and paradigms?

- Fortran, C, C++: Yes.
- PGAS:
  - Not needed in my app space.
  - Might use if MPI-compatible and commercially available.
- Flat MPI: Yes, for a while.
- MPI+Manycore: Required for future, useful now.
- We are already moving in this direction.



- Trilinos/Kokkos: Trilinos compute node package.
- Generic Node object defines:
  - Memory structures for parallel buffers
  - Parallel computation routines

    (e.g., **`parallel_for`**, **`parallel_reduce`**)
- This is an API, not yet-another-ad-hoc-parallel-programming model. (We don't need any more!)
- Needed because there is no portable user-level node programming model (My #1 need).