Present and Future Computing Requirements for **Numerical Algorithms for Electronic and Nuclear Structure Analysis**

Chao Yang

Computational Research Division

Lawrence Berkeley National Laboratory

NERSC ASCR Requirements for 2017
January 15, 2014
LBNL

1. Project Description

22 users, scientists, postdocs and students at LBL and other universities and research institutions

- The main goal is to develop enabling applied mathematics and numerical tools for improving the fidelity and throughput of computational materials science and chemistry research.
- Support several SciDAC3 institution and partnership projects that we are currently funded to work on:
  - FASTMath (eigensolver)
  - BES
  - NP

The problems we solve

Linear and Nonlinear eigenvalue problems

- Kohn Sham density functional theory (KSDFT) based electronic structure analysis
- First-principle molecular dynamics
- Wavefunction methods for electronic structure analysis, i.e., configuration interaction, multi-configuration methods, coupled cluster
- Green's function based excited states calculations
- Configuration interaction for nuclear structure

Our target in 2017

- Perform each KSDFT-based electronic structure calculation for 10,000-atom 3D systems in 1 min
- Perform Green's function based excited state calculation for 1,000-atom systems within days
- Perform CI, MCSCF, CCSD calculations for large molecules in accurate model space
- Perform light nuclei CI calculation in accurate model space with 3 or 4-body potentials

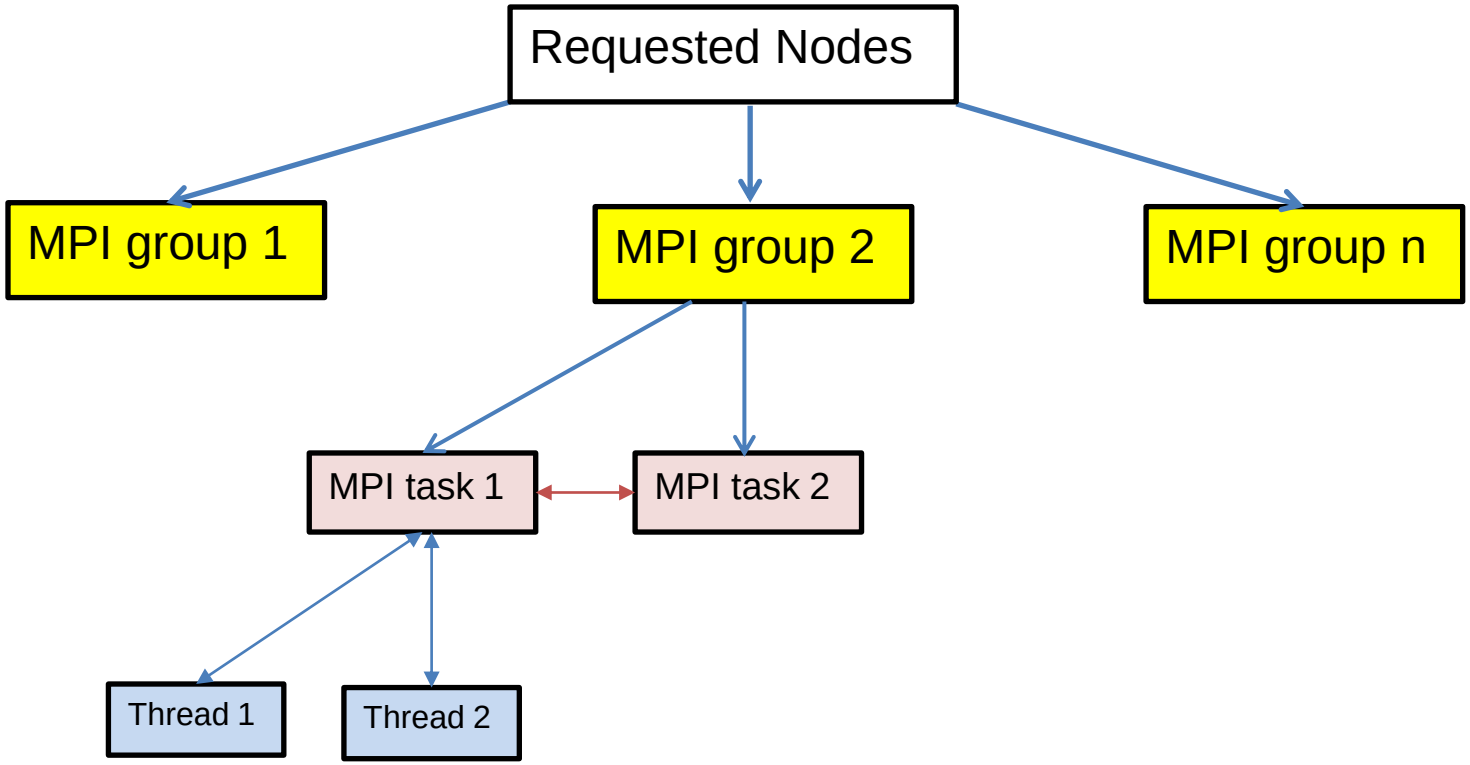
2. Computational Strategies

- KSDFT:
  - Reduce the complexity of electron density evaluation by sparse matrix techniques
  - Reduce the prefactor of electron density calculation by alternative discretization scheme (local adaptive basis + discontinuous Galerkin)
  - Accelerate convergence of nonlinear eigensolver through preconditioning
- Green's function method for excited states:
  - Dielectric matrix calculation without explicit construction of polarizability
  - Efficient GW self-energy integration for full-frequency calculations
  - Efficient eigensolvers for Bethe-Salpeter equation
- Wave function methods
  - Efficient matrix-vector multiplications (tensor contraction, compression)
  - Preconditioner for eigensolver

The code we work with

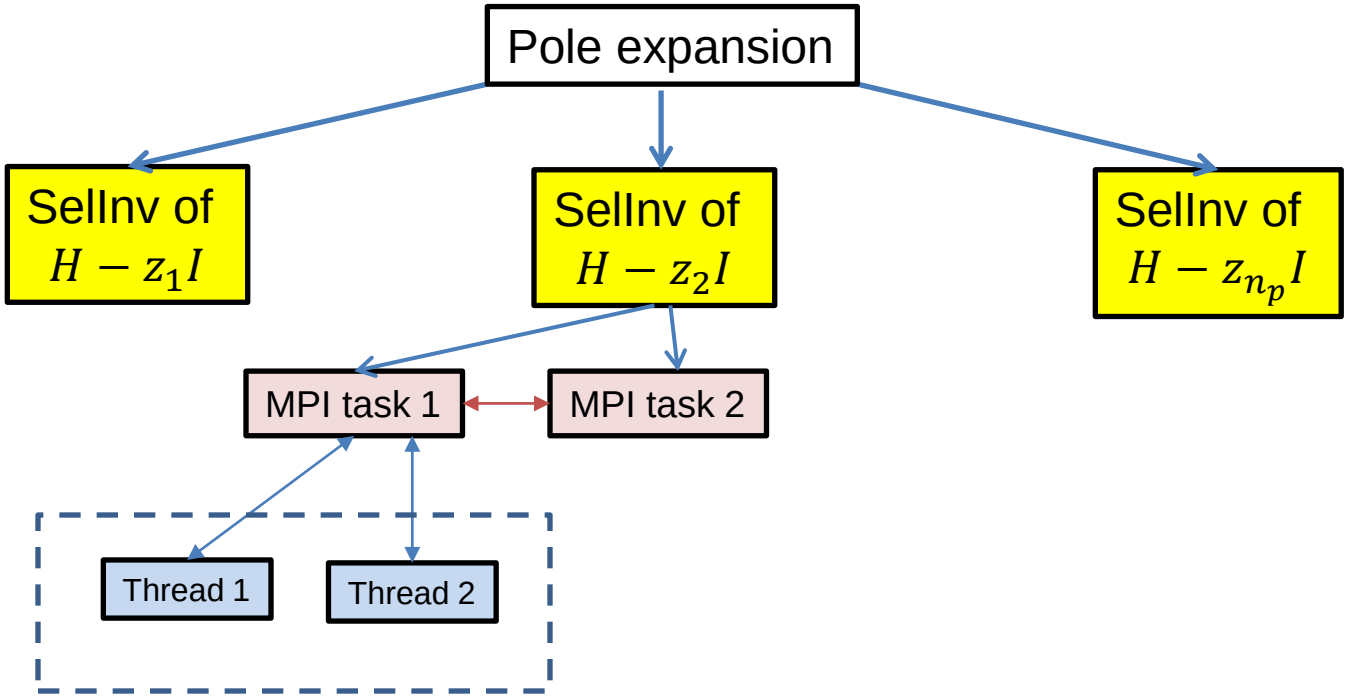
- PPEXSI (parallel pole expansion and selected inversion)
- DGDFT (discontinuous Galerkin based DFT)
- Eigensolvers
- CP2K, SIESTA, Quantum-espresso, Qbox
- BerkeleyGW
- NWCHEM, Qchem
- MFDn

Implementation Strategy

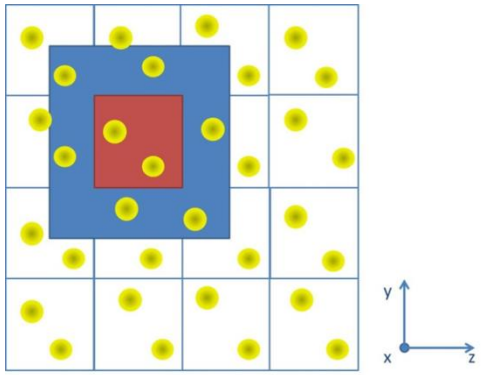


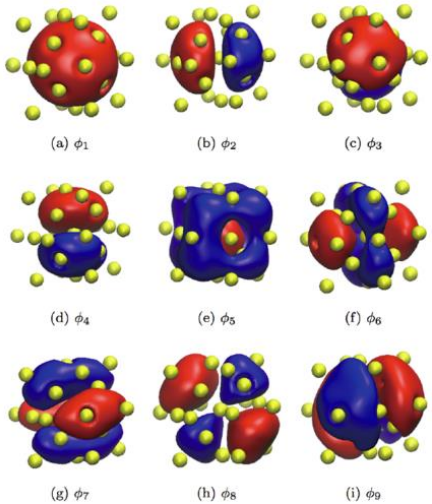
Example 1: PPEXSI

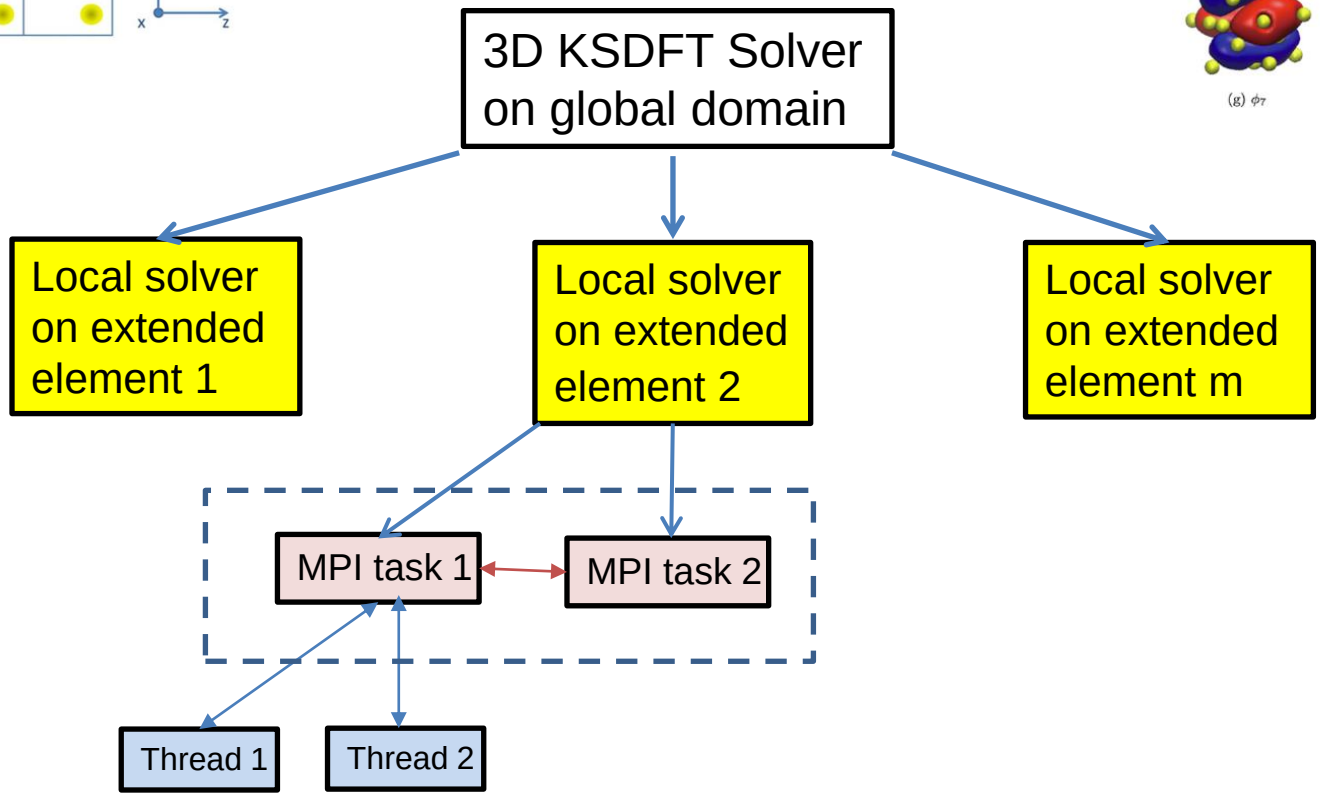
$$\rho \approx Im\left[\sum_{i=1}^{n_p} \omega_i \mathrm{diag}((H - z_i I)^{-1})\right]$$



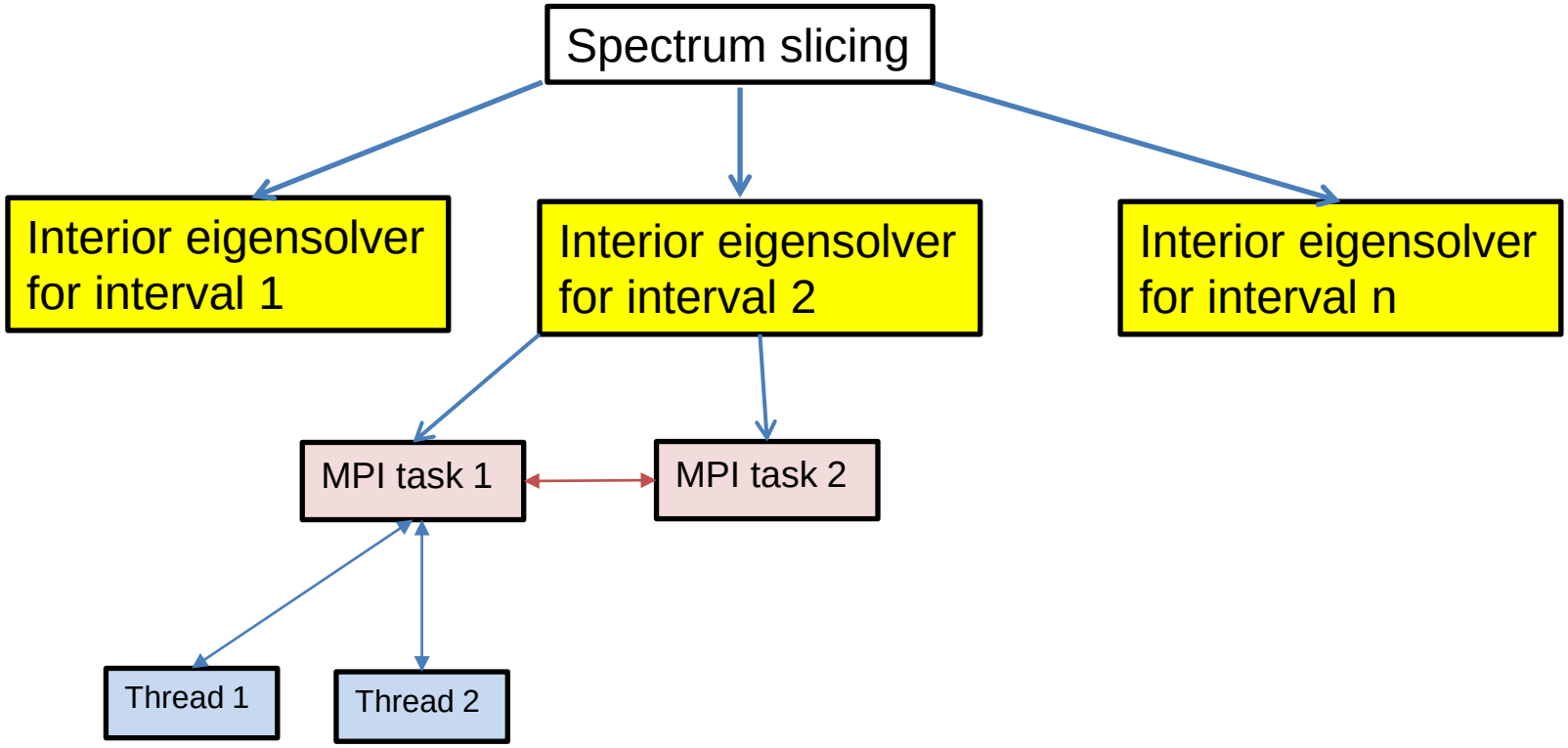
Example 2: DGDFT







Example 3: spectrum slicing based eigensolver

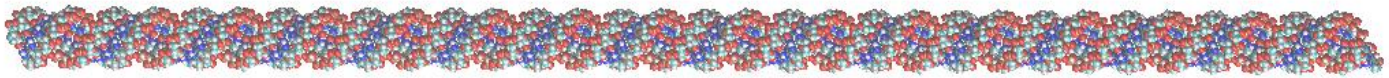


challenges

- Communication overhead in fine grained parallelism
- High thread overhead and NUMA issues within a node
- Variation in runtime due to node topology (topology-aware is possible is some cases but not all)
- I/O overhead in some codes

Scaling

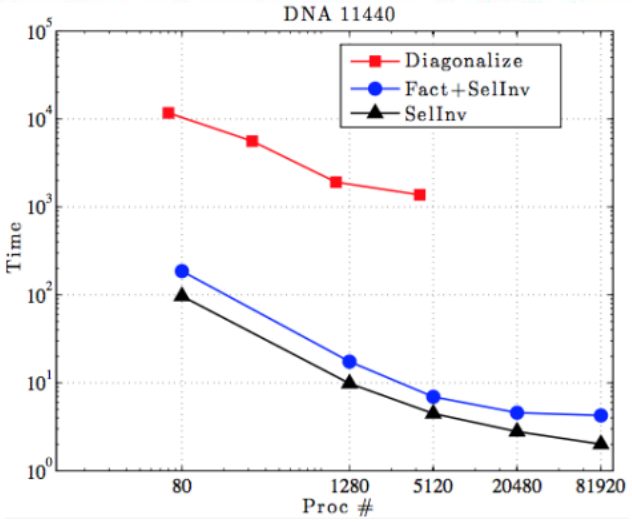
- For sufficiently large problems, our codes exhibit reasonable strong scaling to 100,000 cores
- Weak scaling is important for applications



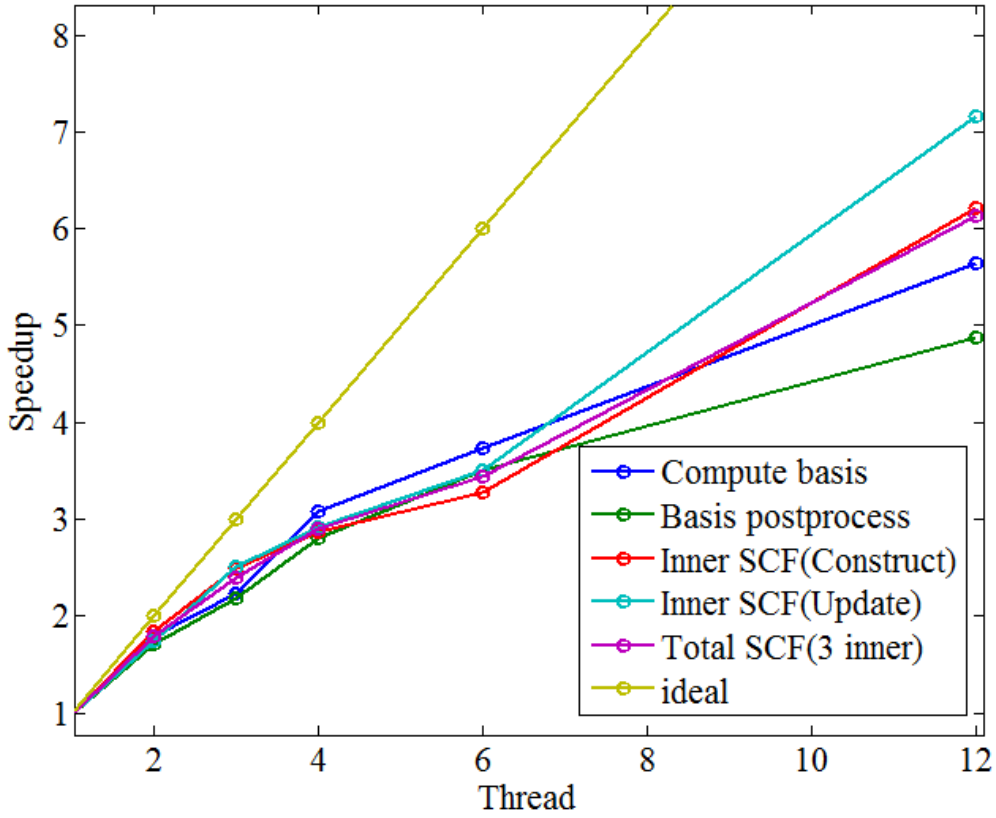
Example: PPEXSI for 11,000-atom DNA, Hamiltonian generated by SIESTA

Selected inversion at one pole

Typically 40 and 80 poles are needed



OpenMP scaling



Constructing local basis on a single extended element by solving a local Kohn-Sham eigenvalue problem

3. Current HPC Usage

- Machines currently using (NERSC or elsewhere)
  - Hopper and Edison
- Hours used in 2012-2013 (list different facilities)
  - 7.5M hours
- Typical parallel concurrency and run time, number of runs per year
  - 500 – 100,000
  - A few large runs to test scaling and correctness on large data
  - Many (thousands or more) interactive small jobs for code development and testing
- Data read/written per run
  - Most code reads and writes a small amount of data (under several GB)
  - Some codes (BerkeleyGW, DGDFT-based MD) can read/write several TB
- Memory used per (node | core | globally)
  - Typically can use all 32 GB per node
  - Can use several TB aggregate memory
- Necessary software, services or infrastructure
  - BLAS/LAPACK/ScaLAPACK, FFTW, quantum espresso,SIESTA,ABINIT,QCHEM, NWCHEM, BerkeleyGW, SuperLU_DIST,Pardisol,MUMPS,Parmetis,PT-SCOTCH,DDT,CrayPat

4. HPC Requirements for 2017

- Compute hours needed (in units of Hopper hours)
  - 75M hours
- Changes to parallel concurrency, run time, number of runs per year
  - 10x increase expected
- Changes to data read/written
  - 10x increase
- Changes to memory needed per (core | node | globally)
  - Two orders of magnitude increase per node, three orders of magnitude increase aggregate
- Changes to necessary software, services or infrastructure
  - Be able to run larger jobs (with more than 10,000 cores) interactively without waiting for too long

5. Strategies for New Architectures (1 of 2)

- Our code currently does not contain CUDA/OpenCL directives. We simply do not have enough manpower to develop CUDA/OpenCL versions of the code. Our codes are mostly memory bound and is unlikely to benefit significantly from GPUs that have limited amount of fast memory.
- We do not run on Titan using the GPUs.
- Some of our codes (e.g. DGDFT, MFDn, BerkeleyGW) have OpenMP directives now. There are used to support loop level parallelism within an MPI tasks. We also use multi-threaded BLAS/LAPACK in some cases.
- We current do not run on Mira or Sequoia using threading?
- There is no plan to port to the Intel MIC architecture in the near future.

5. Strategies for New Architectures (2 of 2)

- We closely work with our SciDAC application partners to benchmark and test on the existing architecture. Preparation and planning will begin once the roadmap for the new architecture is clear.
- Our main strategy at the moment for preparing for future architecture are:
  - to develop algorithms that have multiple levels of parallelism with little or no communication at the coarse-grain level even if they converge a bit slower (e.g., domain decomposition, block algorithms)
  - To develop algorithms that trades more highly optimized kernels (e.g. BLAS3) for fewer kernels that are difficult to scale (e.g., dense eigensolver) even if that means we have to perform more flops
- NERSC can help with the transition to these architectures by
  - Help with porting existing codes on future architectures
  - Perform benchmark testing
  - Share experience by presentation and tutorials
- What role should DOE and ASCR play in the transition to these architectures?
  - Provide more sustained funding support for algorithm development

5. Special I/O Needs

- Our code currently do not have checkpoint/restart capability now. But we plan to develop that in the future. This is especially important for MD simulations in which quantities from multiple snapshots should be saved.

- Burst buffer architecture can potentially be beneficial for I/O

6. Summary

- Improvement in NERSC computing hardware, software and services will allow scientists to perform unprecedented simulation for novel materials to study the effects of doping, defects on the electronic, mechanical and optical properties of materials
- Scalable architecture, with high quality system and application software support (compilers, MPI etc.) are important for enabling breakthroughs in science
- With a 10x increase in computational resources, we believe we can achieve the goals we set out in our SciDAC project
  - Perform electronic structure calculation for a 10,000-atom 3D system (Li-ion battery) under 1 min. This will allow scientists to run longer trajectories in first principle MD simulation
  - Perform full-frequency GW calculation for 1000-atom systems