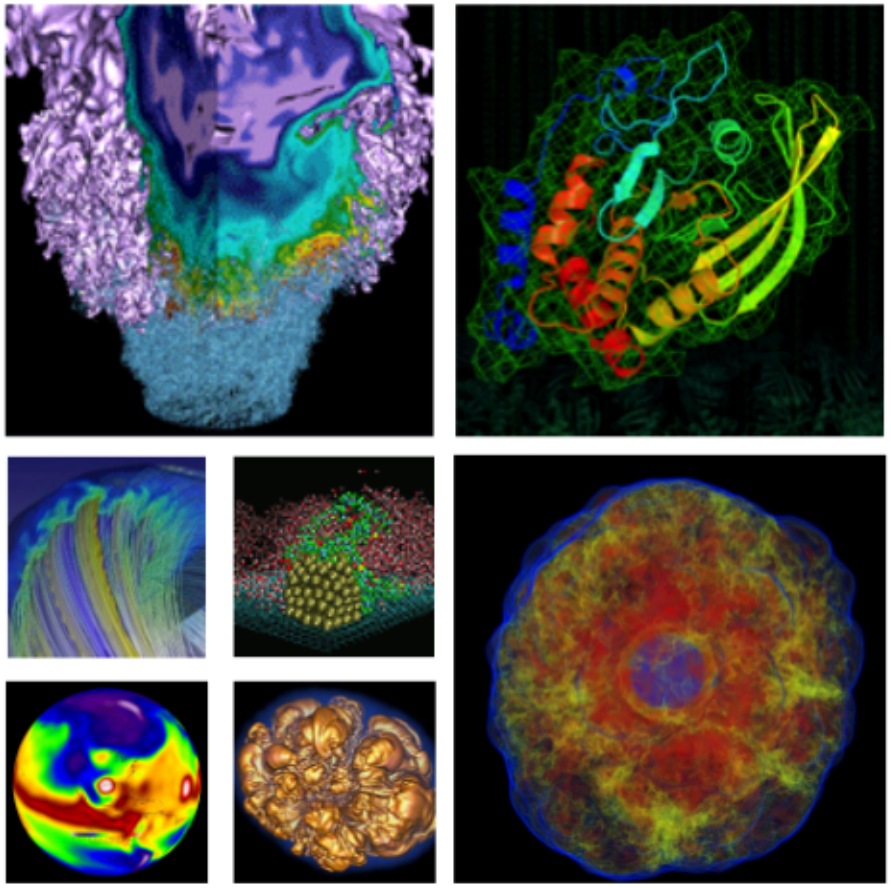
# BerkeleyGW: Evolution of a Materials Science Code



Jack Deslippe (NERSC)

HPC Forum

U.S. DEPARTMENT OF ENERGY | Office of Science

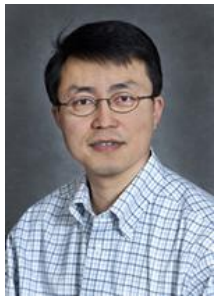
BERKELEY LAB
Lawrence Berkeley National Laboratory

## Jack Deslippe

- Ph.D. In Physics from Berkeley in 2012
- NERSC User Services Group (Materials Science / Chemistry Consultant)
- **Developer in BerkeleyGW project**
- NESAP (NERSC's exascale readiness program)
- NERSC liaison/developer for NERSC - DOE Light-source big data integration efforts
- Developer of MyNERSC real-time web portal

# Other Developers

Steven G. Louie
UC Berkeley, LBNL



Chao Yang
LBNL

David Strubbe
MIT

Jeff Neaton
LBNL

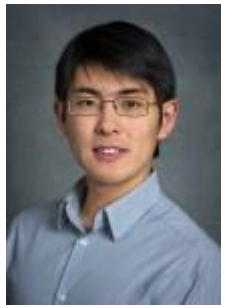
Felipe Jornada
UC Berkeley

Derek Vigil
UC Berkeley

Fang Liu
LBNL

Jamal Mustafa
UC Berkeley

Johannes Lischner
LBNL



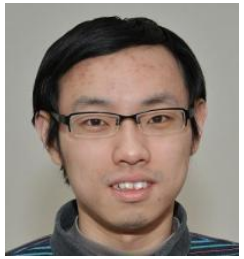
Lin Lin
LBNL

Georgy Samsonidze
BOSCH

Brad Barker
UC Berkeley

Andrew Canning
LBNL

Sahar Sharifzadeh
LBNL

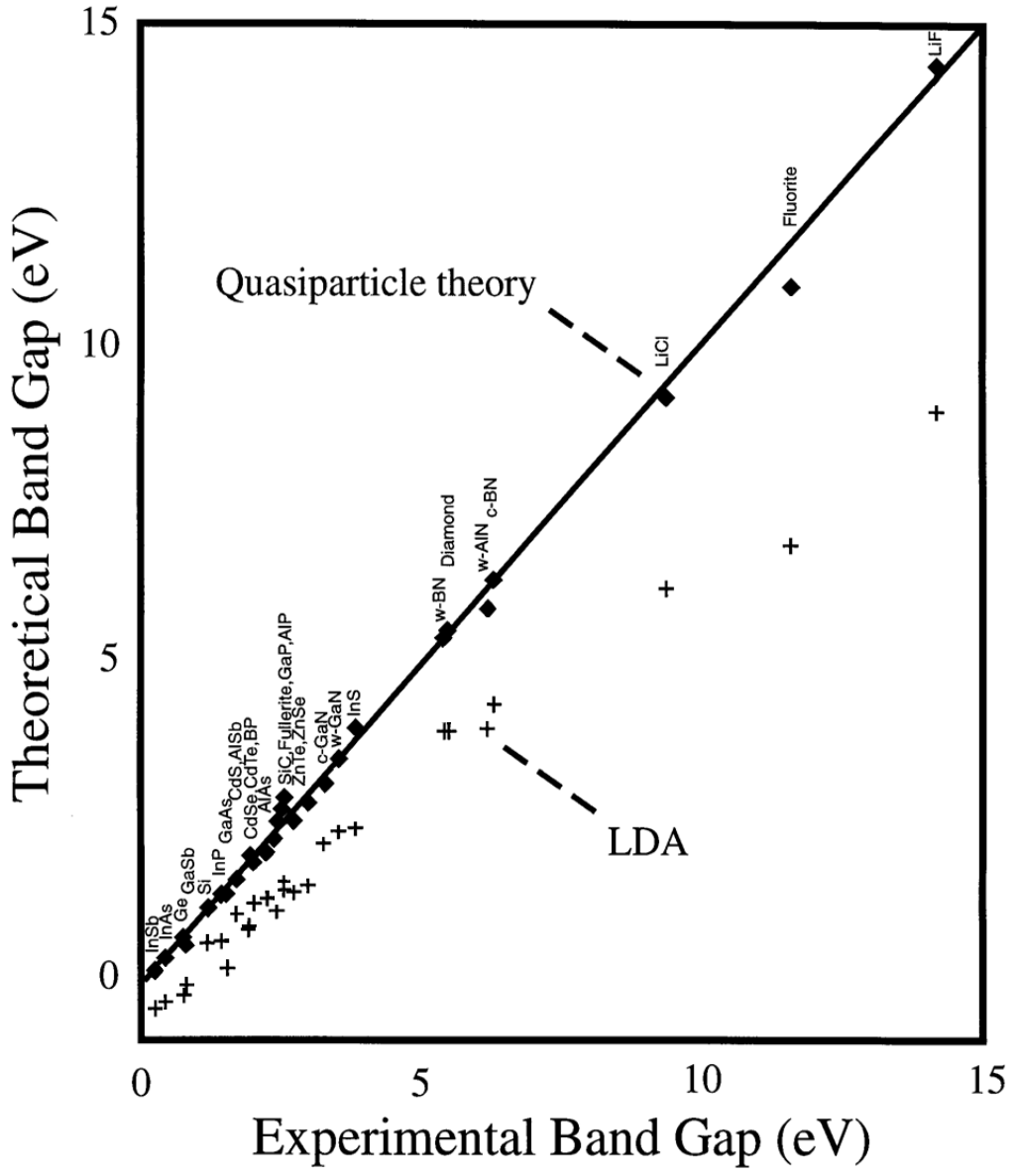


Meiyue Shao

# What is GW

**Materials**:

InSb, InAs
Ge
GaSb
Si
InP
GaAs
CdS
AlSb, AlAs
CdSe, CdTe
BP
SiC
$C_{60}$
GaP
AlP
ZnTe, ZnSe
c-GaN, w-GaN
InS
w-BN, c-BN
diamond
w-AlN
LiCl
Fluorite
LiF

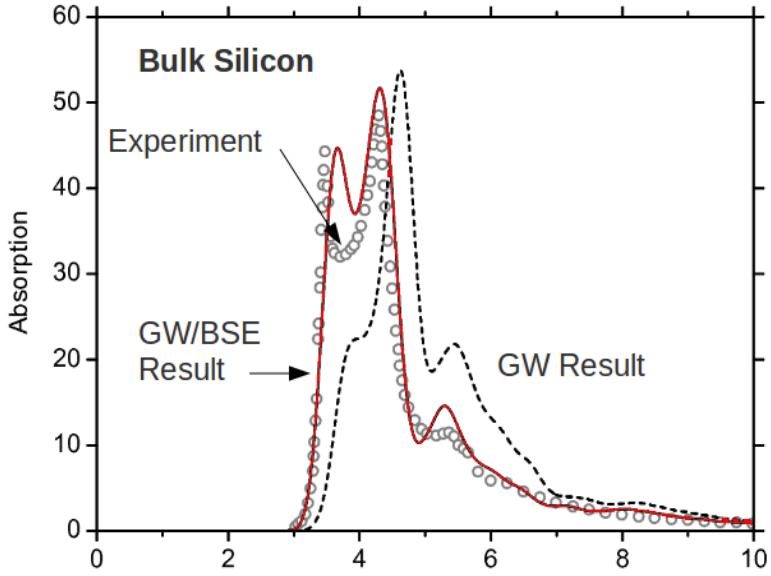


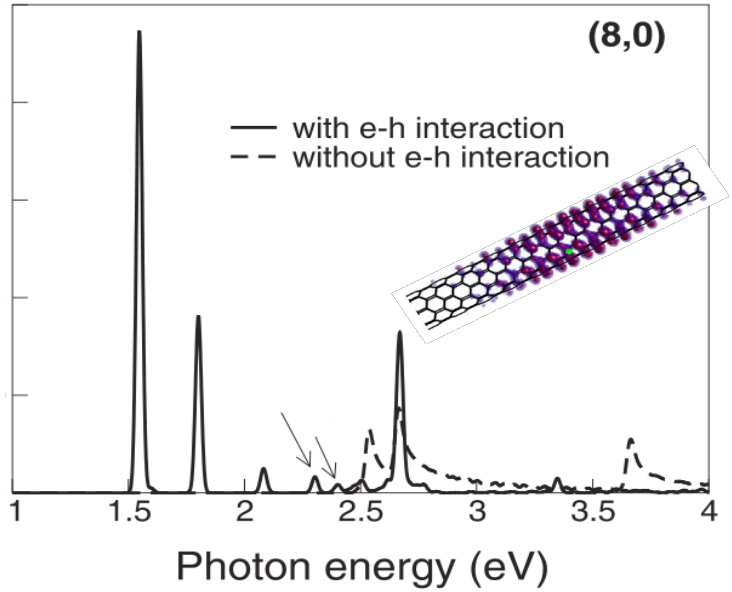
# Why We Need GW

Many-body effects extremely important in **Excited-State properties** of Complex Materials.

Accurately describes properties important for:

- Photovoltaics
- LEDs
- Junctions / Interfaces
- Defect Energy Levels
- ….



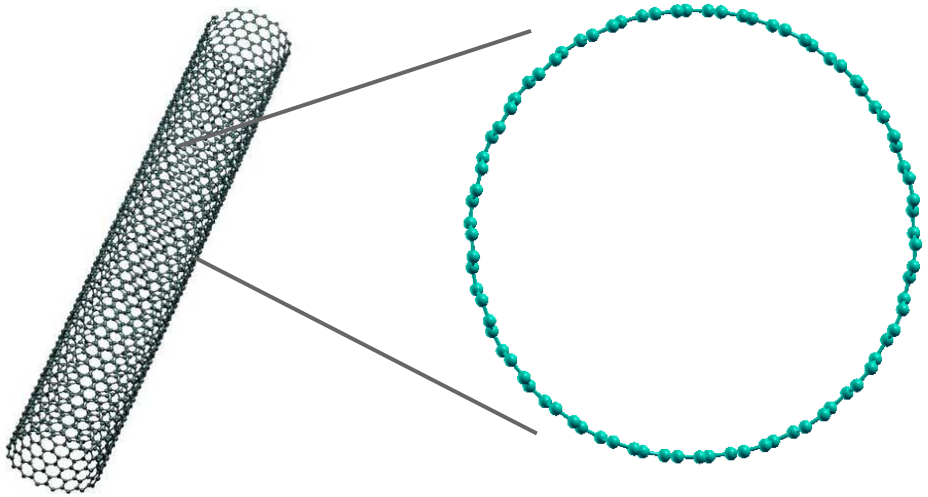
# BerkeleyGW in the Rush to Publish Era

# The State of The Code in 2006

- Originates in 1980's

- Over next 20 years, develops organically
  - ~20 developers
  - 10's of different versions (each with different features/issues/bugs)

- Some versions had "basic" parallelization with MPI
  - Communication done with Disk-IO
  - Significant Serial Bottlenecks
  - Hardwall's on system size due to non-distributed arrays



**The Good:**

Quantitatively accurate for quasiparticle properties in a wide variety of systems.

Accurately describes dielectric screening important in excited state properties.

**The Bad:**

Prohibitively slow for large systems. Usually thought to cost orders of magnitude more time that DFT.

Memory intensive and scales badly. Exhausted by storage of the dielectric matrix and wavefunctions. Limited ~50 atoms.

**The Good:**

Quantitatively accurate for quasiparticle properties in a wide variety of systems.

Accurately describes dielectric screening important in excited state properties.

**The Bad:**

Prohibitively slow for large systems. Usually thought to cost orders of magnitude more time than DFT.

Memory intensive and scales badly. Exhausted by storage of the dielectric matrix and wavefunctions. Limited ~50 atoms.

# BerkeleyGW in the MPP Era

1. Pick a logo


2. Use version control (SVN) Create testsuite / buildbot

3. Profile (with tools, IPM/craypat/hpctoolkit), utilize timers throughout, wrap allocate statements etc…

4. Parallelize, distribute memory etc...

1. Compute via *nxn'* FFTs ($N^3$ Step. Big Prefactor.):

$$M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G}) = \langle n\mathbf{k}+\mathbf{q}|\, e^{i(\mathbf{q}+\mathbf{G})\cdot\mathbf{r}}\, |n'\mathbf{k}\rangle$$

$$M_{nn'}(\mathbf{k},\mathbf{q},\{\mathbf{G}\}) = FFT^{-1}\left(\phi_{n,\mathbf{k}+\mathbf{q}}(\mathbf{r}) * \phi^*_{n',\mathbf{k}}(\mathbf{r})\right)$$

**2. Compute sum via large ZGEMM ($N^4$ Step. Small Prefactor. All to All Communication Done):**

$$\chi_{\mathbf{GG'}}(\mathbf{q};0) = \mathbf{M}(\mathbf{G},\mathbf{q},(n,n',\mathbf{k}))\cdot\mathbf{M}^T(\mathbf{G'},\mathbf{q}(n,n',\mathbf{k}))$$

Where, $\mathbf{M}(\mathbf{G},\mathbf{q},(n,n',\mathbf{k})) = M_{nn'}(\mathbf{k},\mathbf{q},\mathbf{G})\cdot\dfrac{1}{\sqrt{E_{n\mathbf{k}+\mathbf{q}}-E_{n'\mathbf{k}}}}$

3. Matrix Inversion. ScaLAPACK

## (Sigma GPP Option)

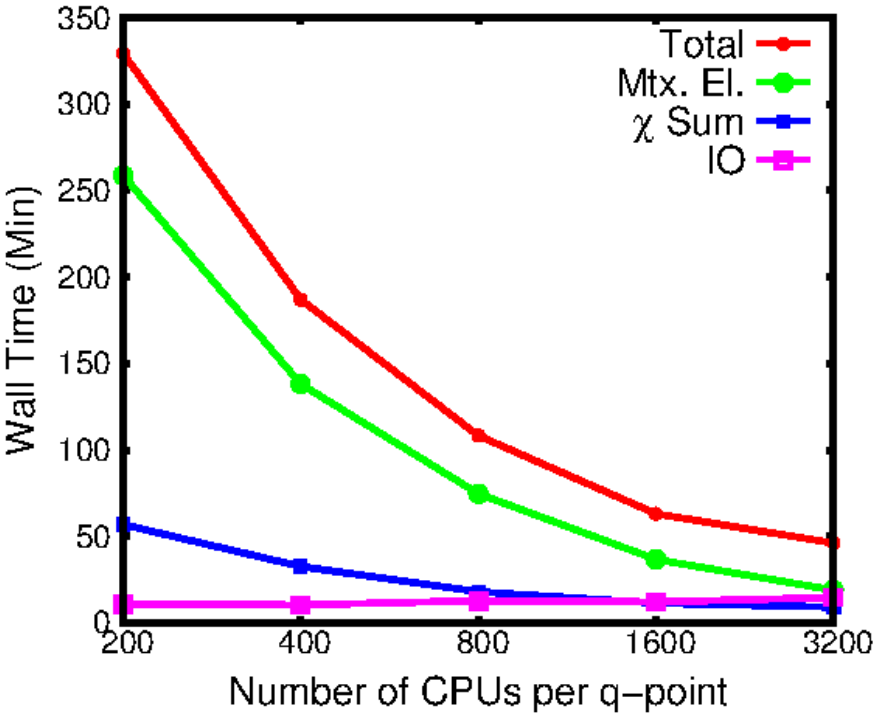
4. Manual loop reductions to compute sum for self-energy.

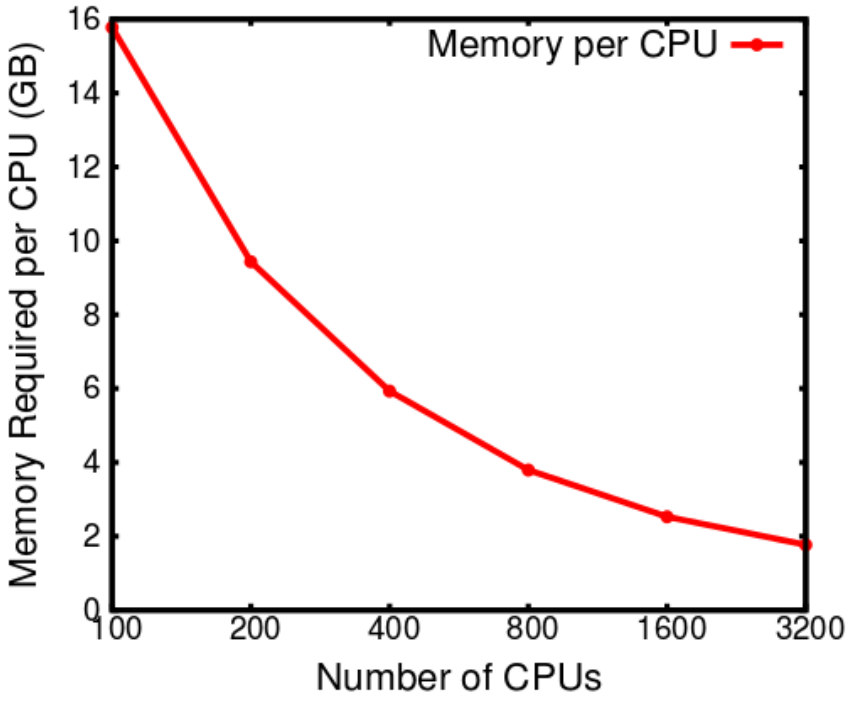
$N^3$ x <number of bands of interest>

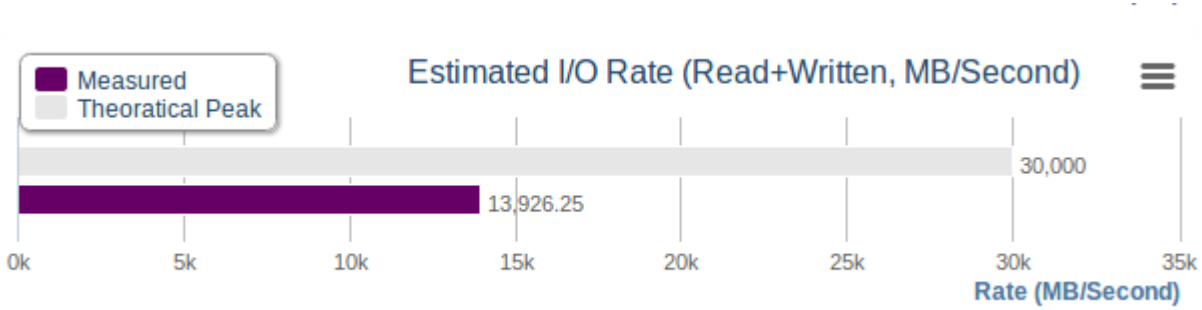
$$\langle n\mathbf{k}|\,\Sigma_{\mathrm{SX}}(E)\,|n'\mathbf{k}\rangle = -\sum_{n''}^{\mathrm{occ}}\sum_{\mathbf{q}\mathbf{G}\mathbf{G}'} M^*_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G}') \times\left[\delta_{\mathbf{G}\mathbf{G}'}+\frac{\Omega^2_{\mathbf{G}\mathbf{G}'}(\mathbf{q})\,(1-i\tan\phi_{\mathbf{G}\mathbf{G}'}(\mathbf{q}))}{(E-E_{n''\mathbf{k}-\mathbf{q}})^2-\tilde{\omega}^2_{\mathbf{G}\mathbf{G}'}(\mathbf{q})}\right]v(\mathbf{q}+\mathbf{G}')$$

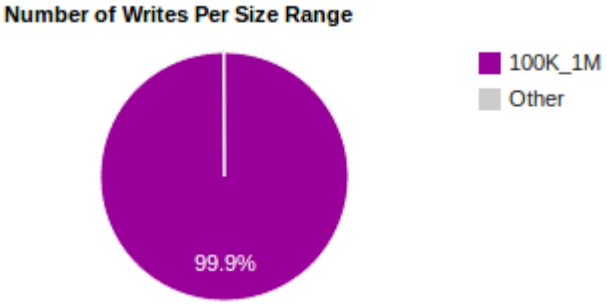
$$\langle n\mathbf{k}|\,\Sigma_{\mathrm{CH}}(E)\,|n'\mathbf{k}\rangle = \frac{1}{2}\sum_{n''}\sum_{\mathbf{q}\mathbf{G}\mathbf{G}'} M^*_{n''n}(\mathbf{k},-\mathbf{q},-\mathbf{G})M_{n''n'}(\mathbf{k},-\mathbf{q},-\mathbf{G}') \times\frac{\Omega^2_{\mathbf{G}\mathbf{G}'}(\mathbf{q})\,(1-i\tan\phi_{\mathbf{G}\mathbf{G}'}(\mathbf{q}))}{\tilde{\omega}_{\mathbf{G}\mathbf{G}'}(\mathbf{q})\,(E-E_{n''\mathbf{k}-\mathbf{q}}-\tilde{\omega}_{\mathbf{G}\mathbf{G}'}(\mathbf{q}))}\,v(\mathbf{q}+\mathbf{G}')$$

# MPI Scaling of Epsilon Code:



# BerkeleyGW in the Many-Core Era

# The NERSC-8 System: Cori

- **Cori will begin to transition the workload to more energy efficient architectures**

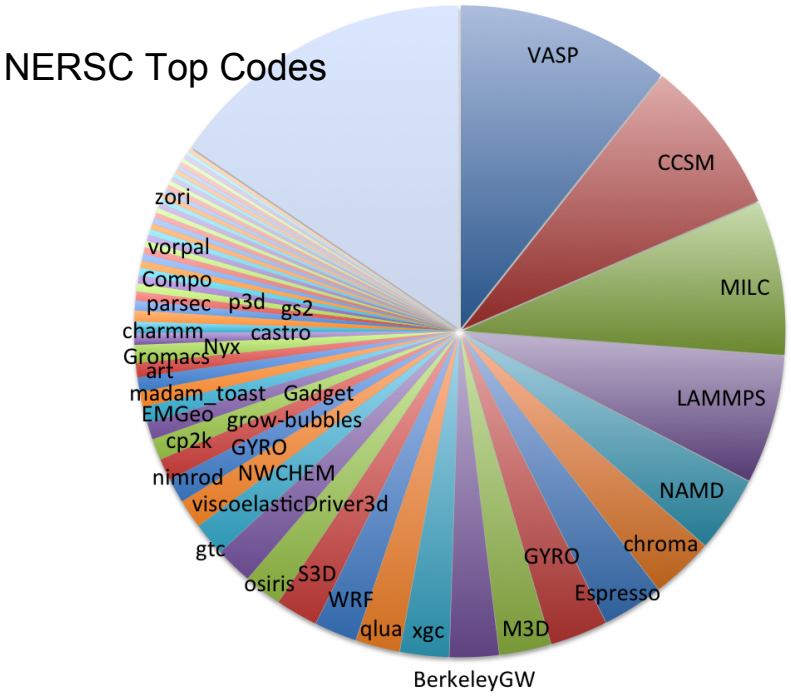
- **Cray XC system with over 9300 Intel Knights Landing compute nodes**
  - Self-hosted, (not an accelerator) manycore processor with over 60 cores per node (support for four hardware threads)
  - MPI + OpenMP programming model
  - AVX512 Vector pipelines with a hardware vector length of 512 bits (eight double-precision elements)
  - Up to 16GB On Package Memory High Bandwidth Memory (96 GB DDR)


Image source: Wikipedia

System named after Gerty Cori, Biochemist and first American woman to receive the Nobel prize in science.

# Application Readiness

- **NERSC partnering with selected projects (~20) to help prepare application codes for Cori**
- **The program will provide:**
  - early access to NERSC-8 hardware and testbed systems
  - special vendor (Cray + Intel) training and optimization sessions
  - NERSC Staff support and training
- **NERSC will advertise eight two-year postdoctoral research staff positions in a variety of fields**



# 20 NESAP Codes

**ASCR (2)**

Almgren (LBNL) – **BoxLib AMR Framework** used in combustion, astrophysics

Trebotich (LBNL) – **Chombo-crunch** for subsurface flow

**BES (5)**

Kent (ORNL) – **Quantum Espresso**
Deslippe (NERSC) – **BerkeleyGW**
Chelikowsky (UT) – **PARSEC** for excited state materials
Bylaska (PNNL) – **NWChem**
Newman (LBNL) – **EMGeo** for geophysical modeling of Earth

**BER (5)**

Smith (ORNL) – **Gromacs** Molecular Dynamics
Yelick (LBNL) – **Meraculous** genomics
Ringler (LANL) – **MPAS-O** global ocean modeling
Johansen (LBNL) – **ACME** global climate
Dennis (NCAR) – **CESM**

**HEP (3)**

Vay (LBNL) – **WARP & IMPACT-** accelerator modeling
Toussaint (U Arizona) – **MILC** Lattice QCD
Habib (ANL) – **HACC** for *n*-Body cosmology

**NP (3)**

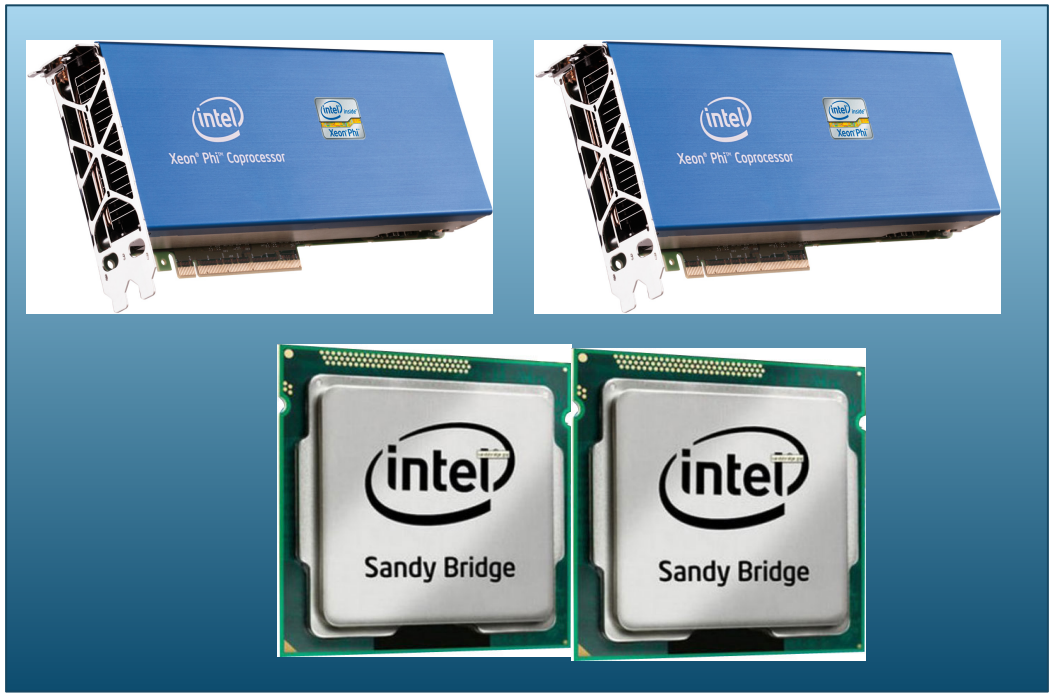
Maris (U. Iowa) – **MFDn** *ab initio* nuclear structure
Joo (JLAB) – **Chroma** Lattice QCD
Christ/Karsch (Columbia/BNL) – **DWF/HISQ** Lattice QCD

**FES (2)**

Jardin (PPPL) – **M3D** continuum plasma physics
Chang (PPPL) – **XGC1** PIC plasma

# Babbage

- NERSC's Xeon-Phi (KNC) testbed. 50 Nodes, each with two KNC and two Sandybridge.
- Used to investigate code performance in "Native Mode". On card performance.




**Intel Xeon Phi coprocessor**

- 60 cores (4 Hardware Threads)
- 8 GB GDDR5

**Intel Xeon CPU E5-2670 0**

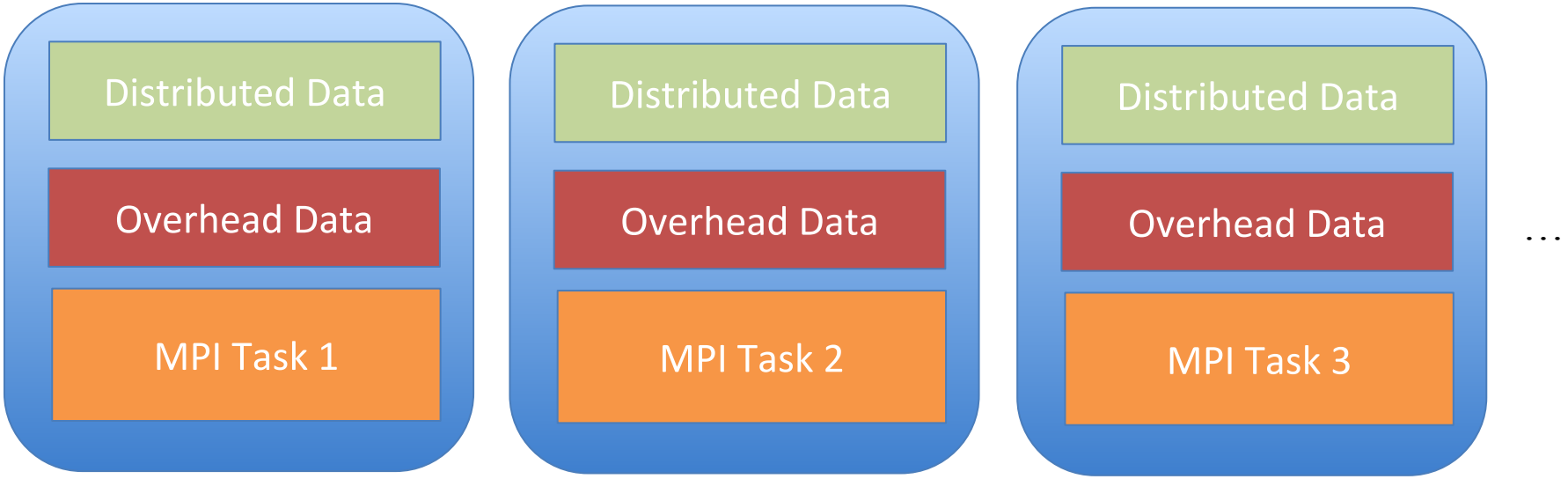
- 8 cores
- 64 GB DRAM

# Failure of the MPI-Only Programming Model in BerkeleyGW

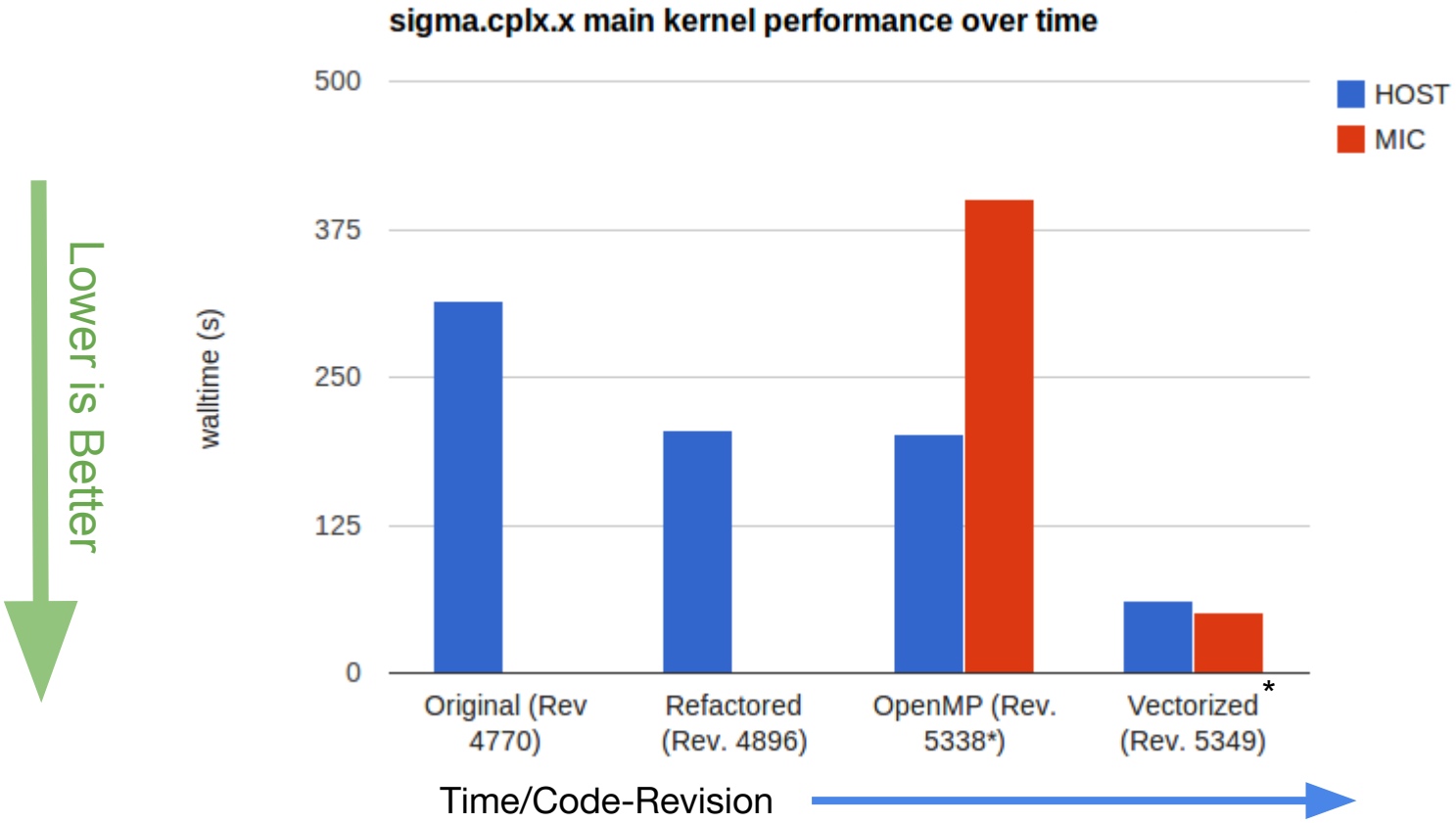
★ Big systems require more memory. Cost scales as $N_{atm}$^2 to store the data.

★ In an MPI GW implementation, in practice, to avoid communication, data is duplicated and **each MPI task has a memory overhead**.

★ On Edison, users sometimes forced to use 1 of 24 available cores, in order to provide MPI tasks with enough memory. **90% of the computing capability is lost**.



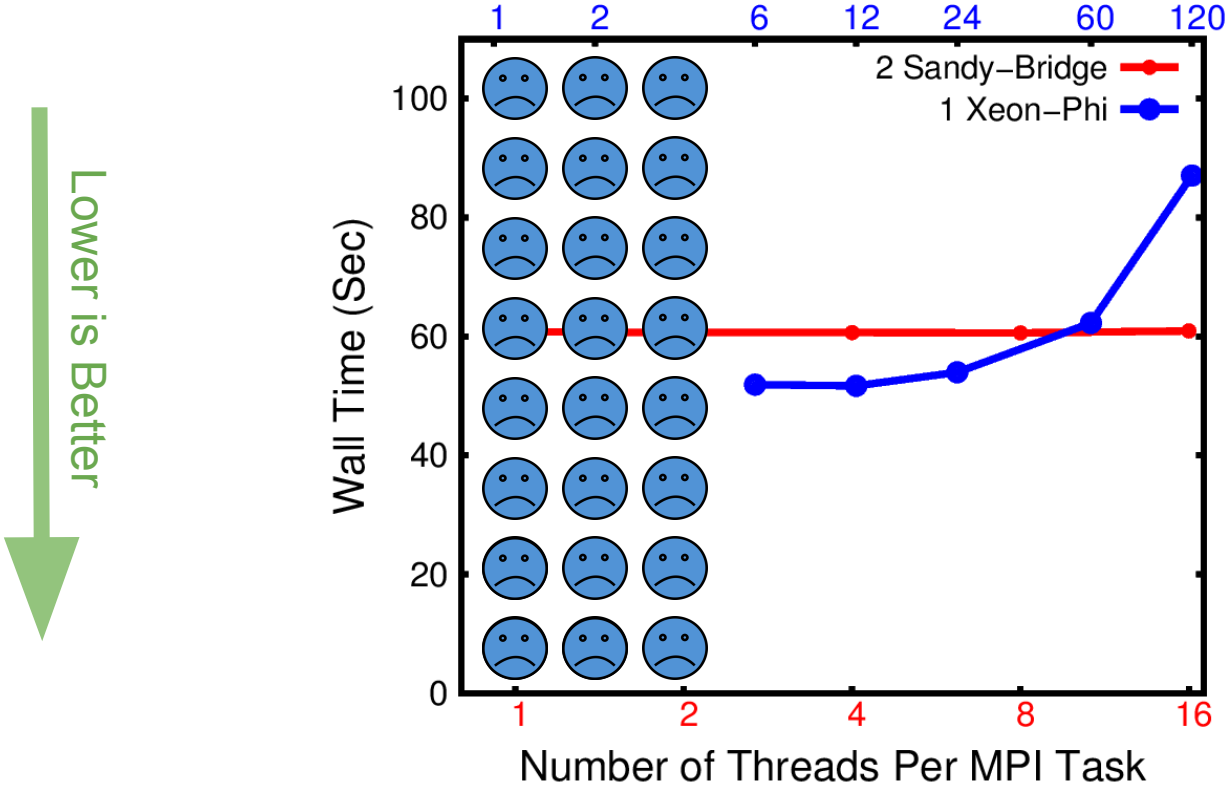
# Steps to Optimize BerkeleyGW on Babbage




1. Refactor to create hierarchical set of loops to be parallelized via MPI, OpenMP and Vectorization and to improve memory locality.
2. Add OpenMP at as high a level as possible.
3. Make sure large innermost, flop intensive, loops are vectorized

* - eliminate spurious logic, some code restructuring simplification and other optimization

# Running on Many-Core Xeon-Phi Requires OpenMP Simply To Fit Problem in Memory




Example problem cannot fit into memory when using less than 5 OpenMP threads per MPI task.

★ **Conclusion: you need OpenMP to perform well on Xeon-Phi in practice**

# Simplified Final Loop Structure

```
!$OMP DO reduction(+:achtemp)
  do my_igp = 1, ngpown

    ...

    do iw=1,3

      scht=0D0
      wxt = wx_array(iw)

      do ig = 1, ncouls

        !if (abs(wtilde_array(ig,my_igp) * eps(ig,my_igp)) .lt. TOL) cycle

        wdiff = wxt - wtilde_array(ig,my_igp)
        delw = wtilde_array(ig,my_igp) / wdiff

        ...

        scha(ig) = mygpvar1 * aqsntemp(ig) * delw * eps(ig,my_igp)

        scht = scht + scha(ig)

      enddo ! loop over g

      sch_array(iw) = sch_array(iw) + 0.5D0*scht

    enddo

    achtemp(:) = achtemp(:) + sch_array(:) * vcoul(my_igp)

  enddo
```

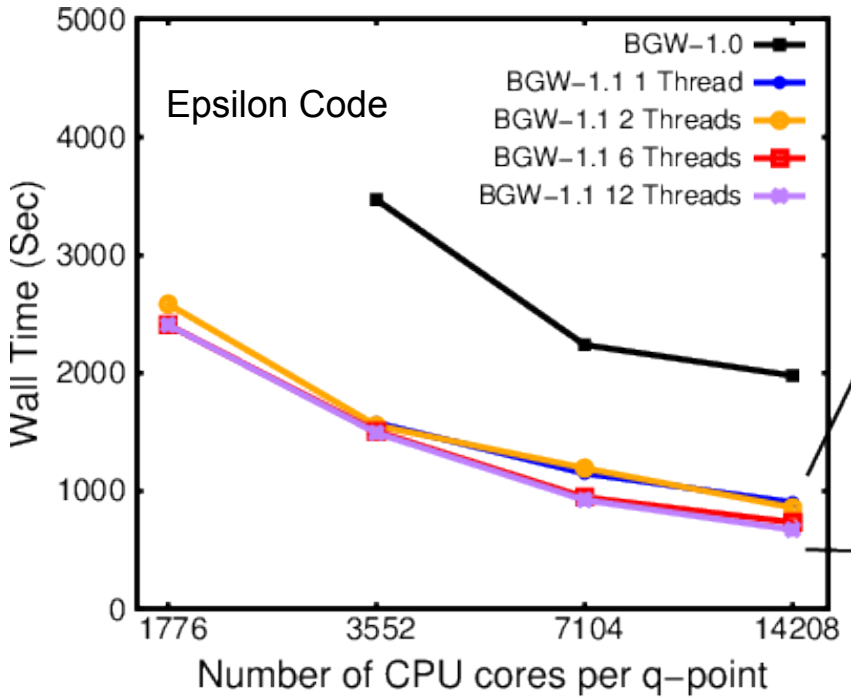
ngpown typically in 100's to 1000s. Good for many threads.

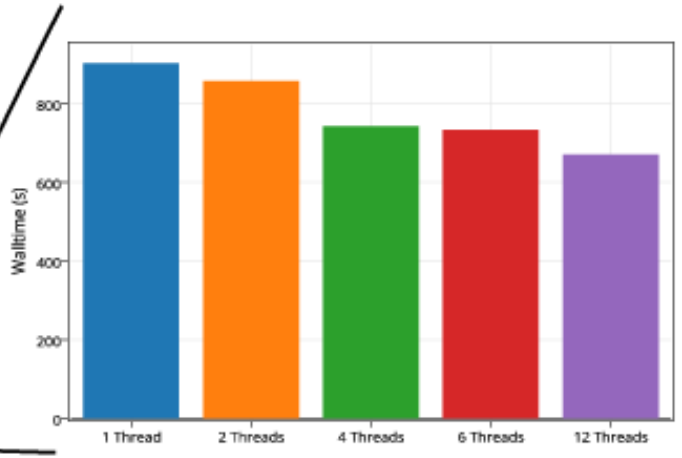
Original inner loop. Too small to vectorize!

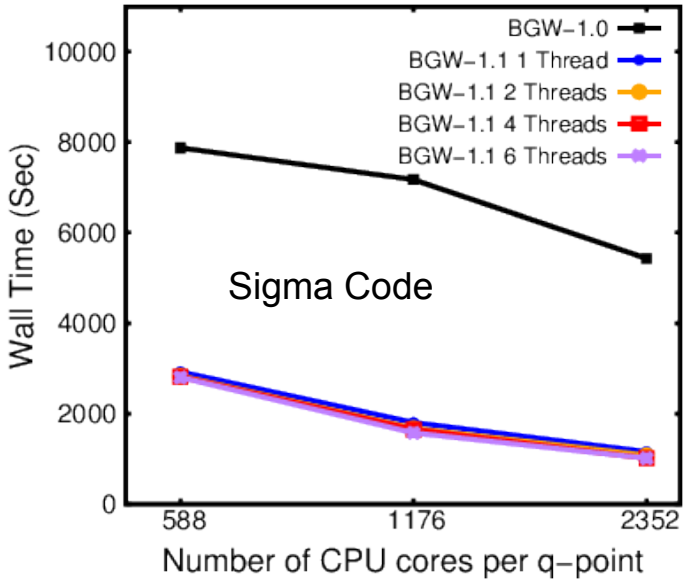
ncouls typically in 1000s - 10,000s. Good for vectorization. Don't have to worry much about memory. alignment.

Attempt to save work breaks vectorization and makes code slower.

# Hybrid MPI-OpenMP Scaling Improvements.



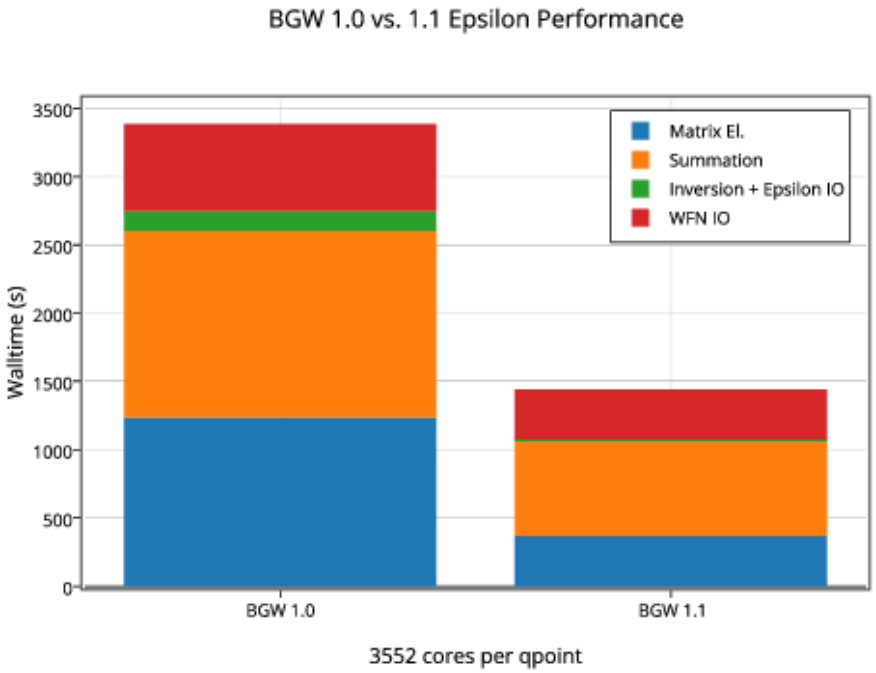



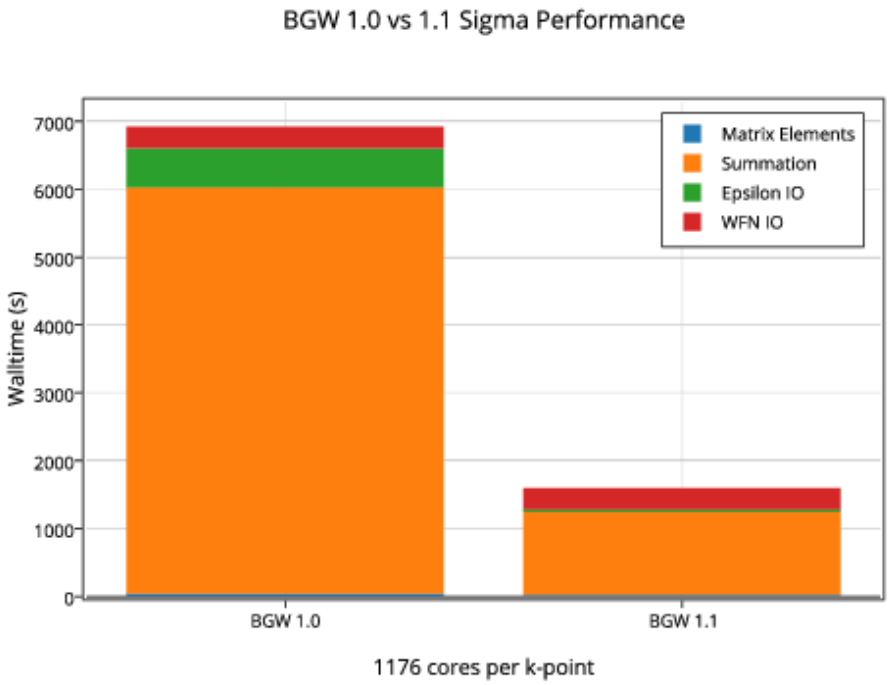




* Major Improvement between 1.0 and 1.1

* Trading MPI tasks for OpenMP threads, yields improved performance (mostly in MPI communication costs) and allows scaling to higher core counts.

# Epsilon/Sigma Improvements




- Performance improvements from:
  - Parallel IO (HDF5)
  - Vectorization
  - Memory-locality improvements
  - Implementation

# My Wishlist

- **Robust code changes.** I don't want to add things in only to take them out again two years later.

- **Performance portability.** Changes made today for one platform should help on all. To the extent possible, don't want multiple branches for each architecture.