

# Managing Petascale Complexity

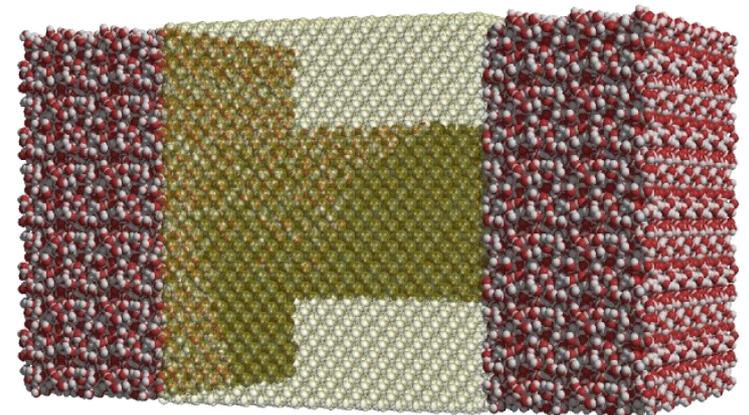
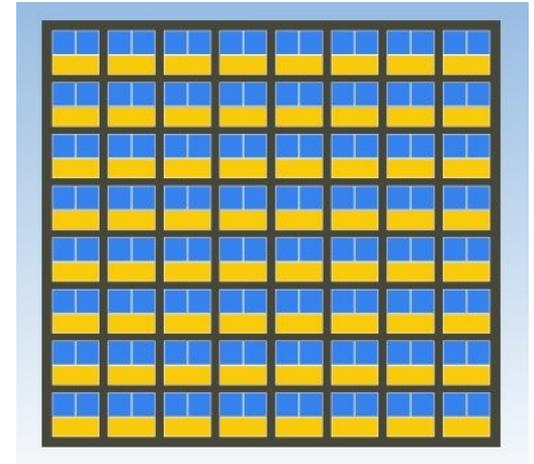
**Matt Leininger**

**Scalable Computing R&D Group  
Sandia National Laboratories  
Livermore, CA**

**23 October 2006**

# Petascale Complexity

- **System Level: scalability and performance issues (Computer Science)**
  - Scalability and reliability of 100K to 1M processors
  - Massively multi-core systems
  - OS features and scalability
  - Petascale I/O interface
  - Parallel file system
- **Human Level: productivity issues (Computational Scientist)**
  - Making people scalable
  - Dealing with complexity at the human level
  - Machine usability



How do we enhance scientific productivity in this highly complex environment?



# An interdisciplinary approach is used to maximize impact of research platforms

- All HPC elements are coupled to guide development of technologies

HPC	Hardware	FPGAs, InfiniBand	Sandia
	Middleware	MPI, CCA	
	Applications	MPQC, S3D, Sundance, PST, CTH, LAMMPS	
	Performance Tools, Benchmarks	VProf, mpiP, Vampir, ...	
	Computing	Institutional and R&D Computing	

# What are Components?

- No universally accepted definition in computer science research ...yet
- A unit of software development/deployment/reuse
  - i.e. has **interesting functionality**
  - Ideally, functionality someone else might be able to **(re)use**
  - Can be **developed independently** of other components
- Interacts with the outside world only through well-defined interfaces
  - **Implementation is opaque** to the outside world
- Can be composed with other components
  - "Plug and play" model to build applications
  - **Composition based on interfaces**

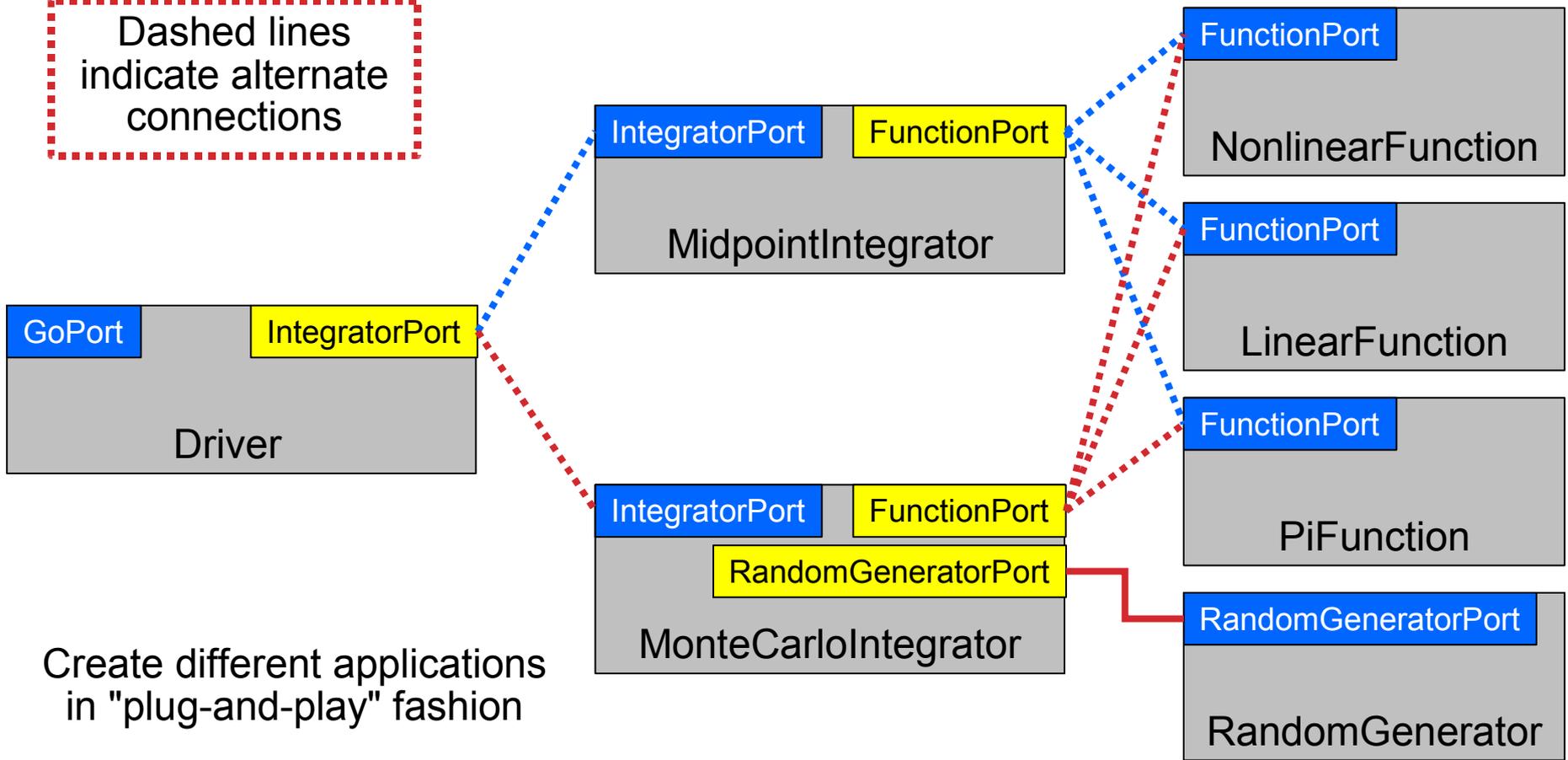


# Common Component Architecture (CCA) Addresses Human Scalability

- CCA is a *specification* of a component environment designed for high performance scientific computing
  - Specification is decided by the CCA Forum
    - CCA Forum membership open to all
  - “CCA-compliant” just means conforming to the specification
    - Doesn't require using any of our code!
- A *tool* to enhance the productivity of scientific programmers
  - Make the hard things easier, make some intractable things tractable
  - Support & promote reuse & interoperability
  - Not a magic bullet
    - i.e. hard problems are still hard

# The Goal: Mix and Match

Dashed lines indicate alternate connections



Create different applications in "plug-and-play" fashion

# Special Needs of Scientific HPC

- Support for legacy software
  - How much **change** required for component environment?
- Performance is important
  - What **overheads** are imposed by the component environment?
- Both parallel and distributed computing are important
  - What approaches does the component model support?
  - What **constraints** are imposed?
  - What are the **performance costs**?
- Support for **languages, data types, and platforms**
  - Fortran?
  - Complex numbers? Arrays? (as first-class objects)
  - Is it available on my parallel computer?

# The Common Component Architecture (CCA) Forum

- Combination of standards body and user group for the CCA
- Define Specifications for *High-Performance* Scientific Components & Frameworks
- Promote and Facilitate Development of Domain-Specific *Common Interfaces*
- Goal: *Interoperability* between components developed by different expert teams across different institutions
- Quarterly Meetings, Open membership...

Mailing List: [cca-forum@cca-forum.org](mailto:cca-forum@cca-forum.org)

<http://www.cca-forum.org/>



# CCA Interfaces Massively Parallel Quantum Chemistry Packages

- MPQC (Massively Parallel Quantum Chemistry) and CCA
  - Capable of utilizing multiple molecular electronic integral packages (cints and intv3)
  - Capable of cross-utilizing HF/DFT molecular orbitals between MPQC and NWChem
  - Capable of using optimization packages through CCA?
  - Comment on performance impact - 5-6% impact (Intel Woodcrest)
-

# Hiding latency with multi-threading in the MP2 algorithm

- Running more compute threads than processors gives processors work while threads are waiting on communications to complete:

- Results are from 2002

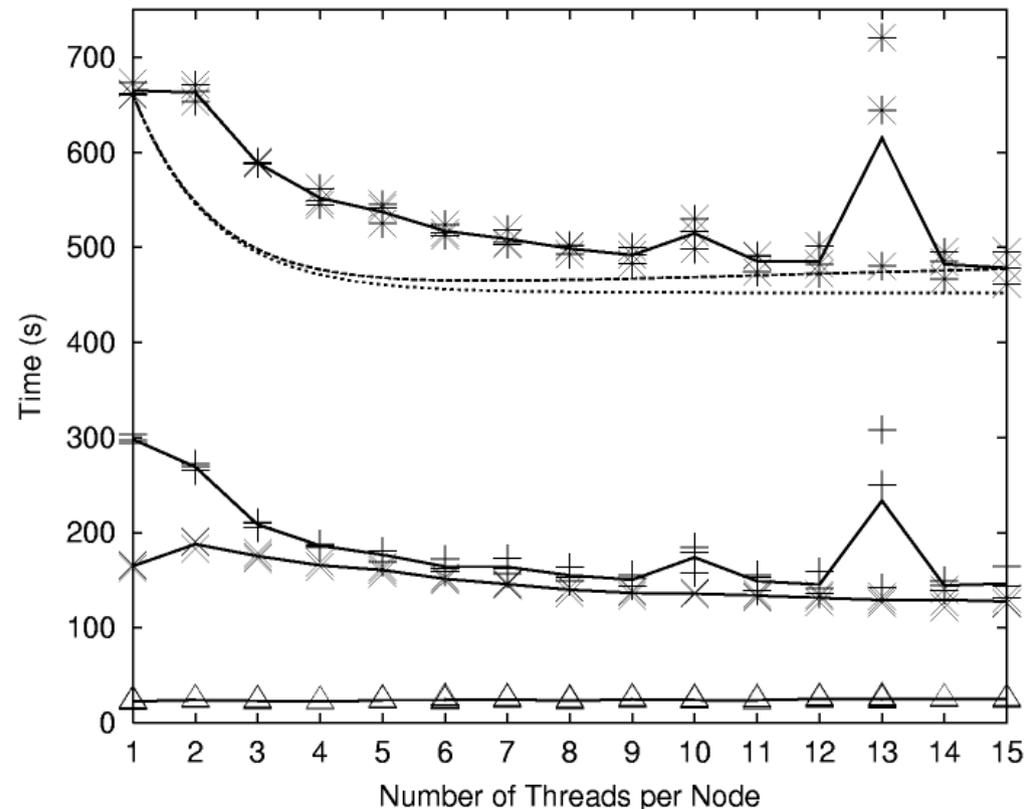
- Dual Pentium III nodes
- GigE interconnect

- Why interesting now?

- Lower latency PCIe interrupt mode
- PNI MWAIT might replace polling

- OpenMPI

MP2/6-31G\*\* Uracil Gradient Timings on 16 Uniprocessor Nodes





## Example 2: Sundance and PST

- High-level programming can produce simulation code that is more capable and more efficient than mid-level programming, at lower cost
- This goes against all conventional wisdom
- Many critical research thrusts do not have science-based engineering capabilities
  - Experimental groups want to move from “build-and-test” to “design-build-improve”

# The 5-floor elevator speech

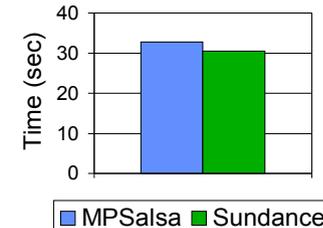
- Principal benefits of high-level tools

- Rapid development of high-performance multiphysics simulators
  - Let scientists be scientists, not programmers
    - Better ROI on staff
  - Improve tech transfer time for advanced algorithms
    - *Code development speed creates runtime speed*
- Enabling of accelerated optimization and UQ algorithms
  - Orders of magnitude improvement in overall computation time
- Automated performance tuning
  - Measurable improvements relative to legacy codes, w/o human intervention, immediately applicable to any problem domain

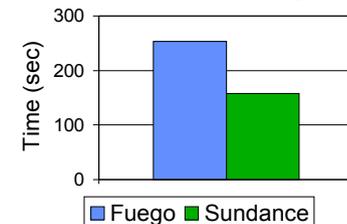
- Some success stories

- Even with little work so far on performance tuning, Sundance already has runtime efficiency at least as good as Ansys, Sierra, and MP-Salsa
- Very fast turnaround in problems in microfluidics, biophysics, microsystems, and large-scale inverse problems

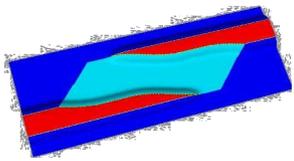
Fully-implicit 3D Navier-Stokes assembly time (Sundance vs MP-Salsa)



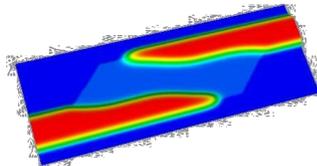
Semi-implicit pressure-projection 3D Navier-Stokes assembly times (Sundance vs Fuego)



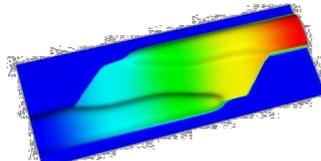
# Sundance is being used in an increasing number of simulation and optimization application problems



Nominal depth

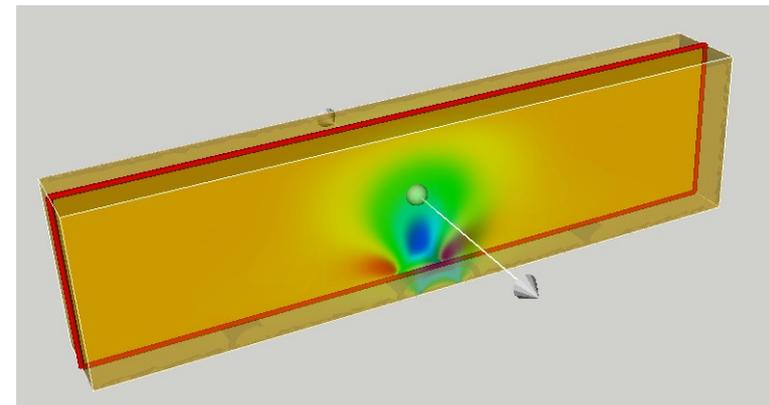


Etched depth

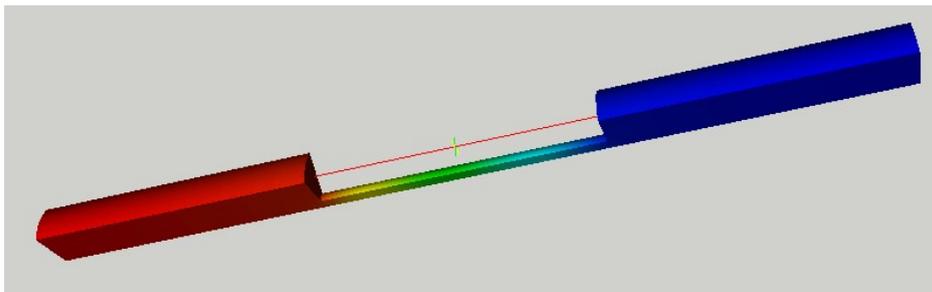


Travel time

Etch-compensated minimum-dispersion microfluidic displacer obtained through shape optimization. Sundance was used to simulate the device manufacturing and performance (KL, A. Skulan, S. Margolis, and P. Boggs, 2005)



Flow velocity above an electrode gap in a microfluidic channel. Sundance was used to develop a simulator coupling electrical, thermal, and fluid effects. Time between receipt of the paper describing the problem's physics to the creation of a working simulation code was under 24 hours (KL and B. van Bloemen Waanders, 2006)



Electric potential in a nanoscale pore in a biological membrane. Sundance was used as the continuum-level component in a coupled atomistic-continuum simulator (Debuschere and Adalsteinsson, 2006)

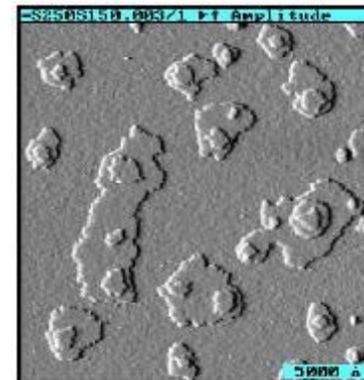
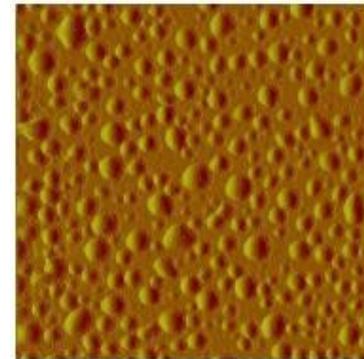
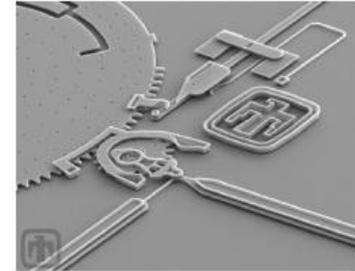


## “Elevator slide”

- Many critical research thrusts do not have science-based engineering capabilities
  - Experimental groups want to move from “build-and-test” to “design-build-improve”
- Multiscale simulations provide a way to extend the capabilities of existing tools
  - The problem is often not the physics, but the scales
- Sandia should be taking the lead in developing core multiscale capabilities

# Science at interfaces

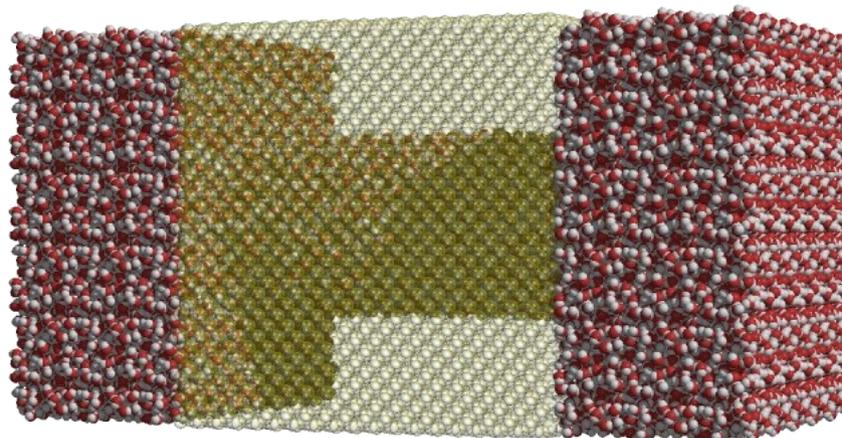
- Critical in a host of key applications
  - Design-and-build is currently the “state of the art”
  - MEMS, composites, electronics, separations, detectors, biology, next-generation materials
- Integration is key to interfacial science
  - CINT: Nano{mechanics, electronics, bio}
  - Materials research
- Interface behavior can affect system behavior far beyond interface region



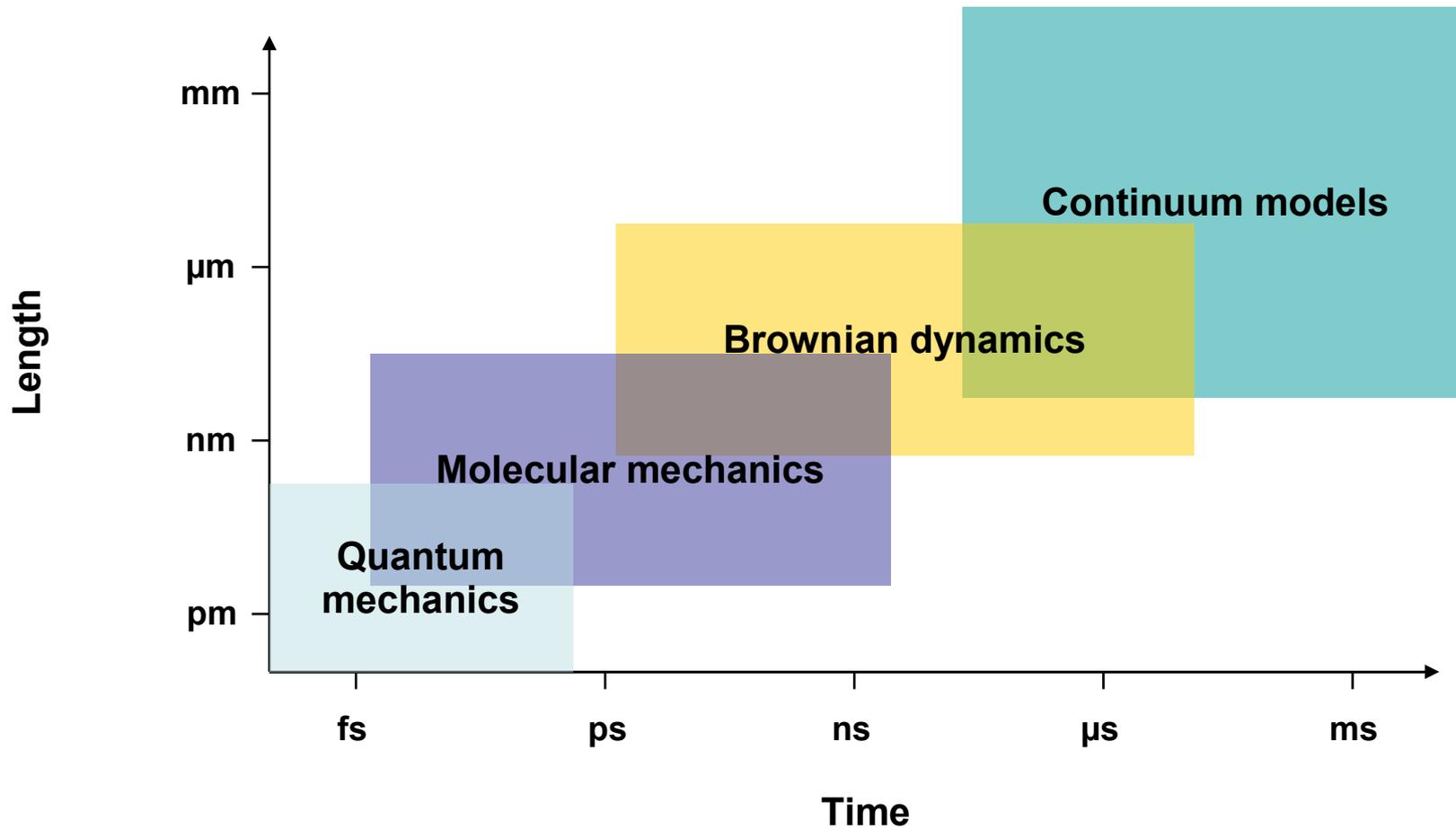


# Water-to-solid interfaces

- Critical in many important application domains
  - Separation and purification systems
  - Detectors and chemical analysis
  - Nanomaterials and biological systems
- Particularly hard to model or analyze
  - Near-wall water has significantly different properties
  - Interface boundary is highly dynamic
  - In nanoscale systems, the key system may be “interface”



# Multiscale simulations can, in theory, be used to couple scales





# ...but there are precious few good multiscale models

- Implementations
  - Interoperability between different software modules
  - Parallelization and performance
  - Obtaining relevant (experimental) data
- Algorithms
  - Translating data between disjoint models
  - System partitioning
  - Handling under-resolved systems
  - Uncertainty quantification
  - V&V

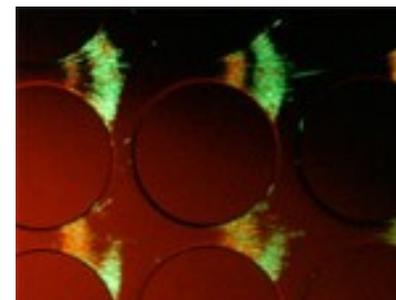
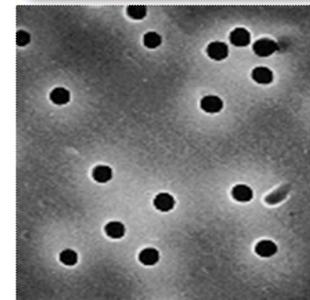
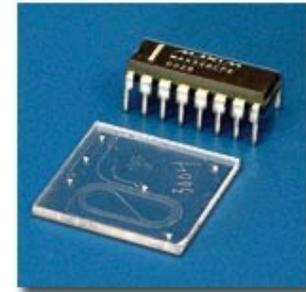


# We are up to the challenge

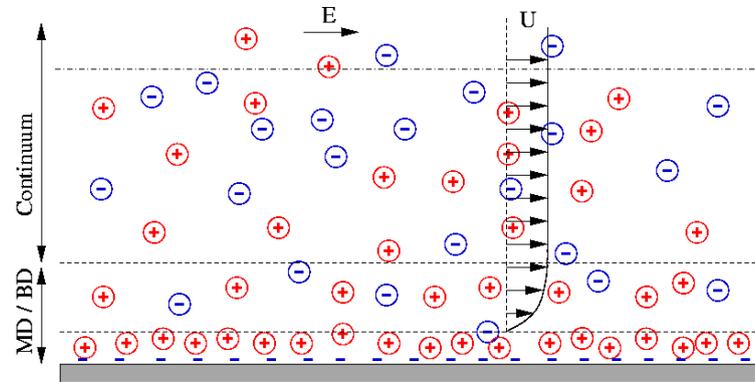
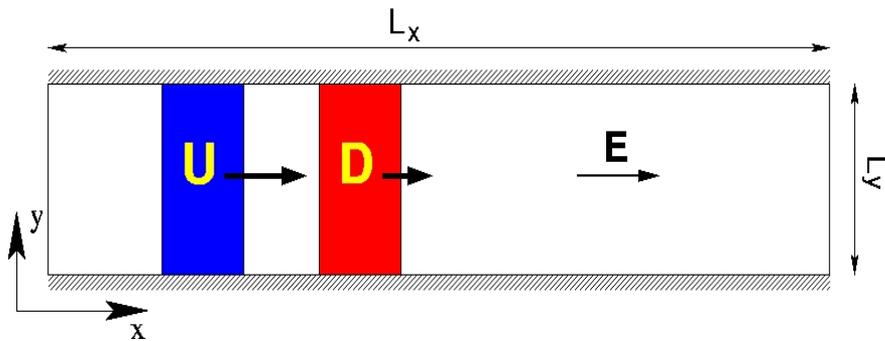
- Modular software architectures
  - Sundance, PST, MPQC, CCA, ...
- Uncertainty quantification and V&V
- Mathematics of multiscale systems
- Extensive experience in HPC
- Access to world-class experimental collaborators

# Focus: Water-solid systems with tightly-coupled physics

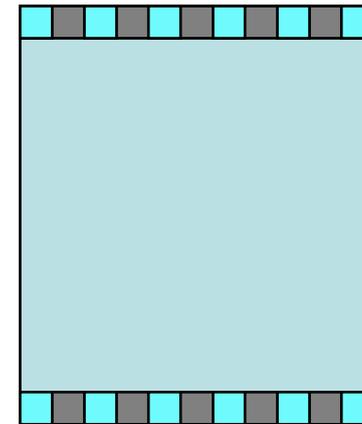
- Microfluidics
  - Key technology in detectors, sensors, and portable analysis systems
  - Performance limited by band spreading
- Nanoseparations
  - Key systems in purifications and analysis, new systems in water surety
  - Need for predictive simulation tools
- Dielectrophoresis
  - Emerging technology in nanoparticle manipulation



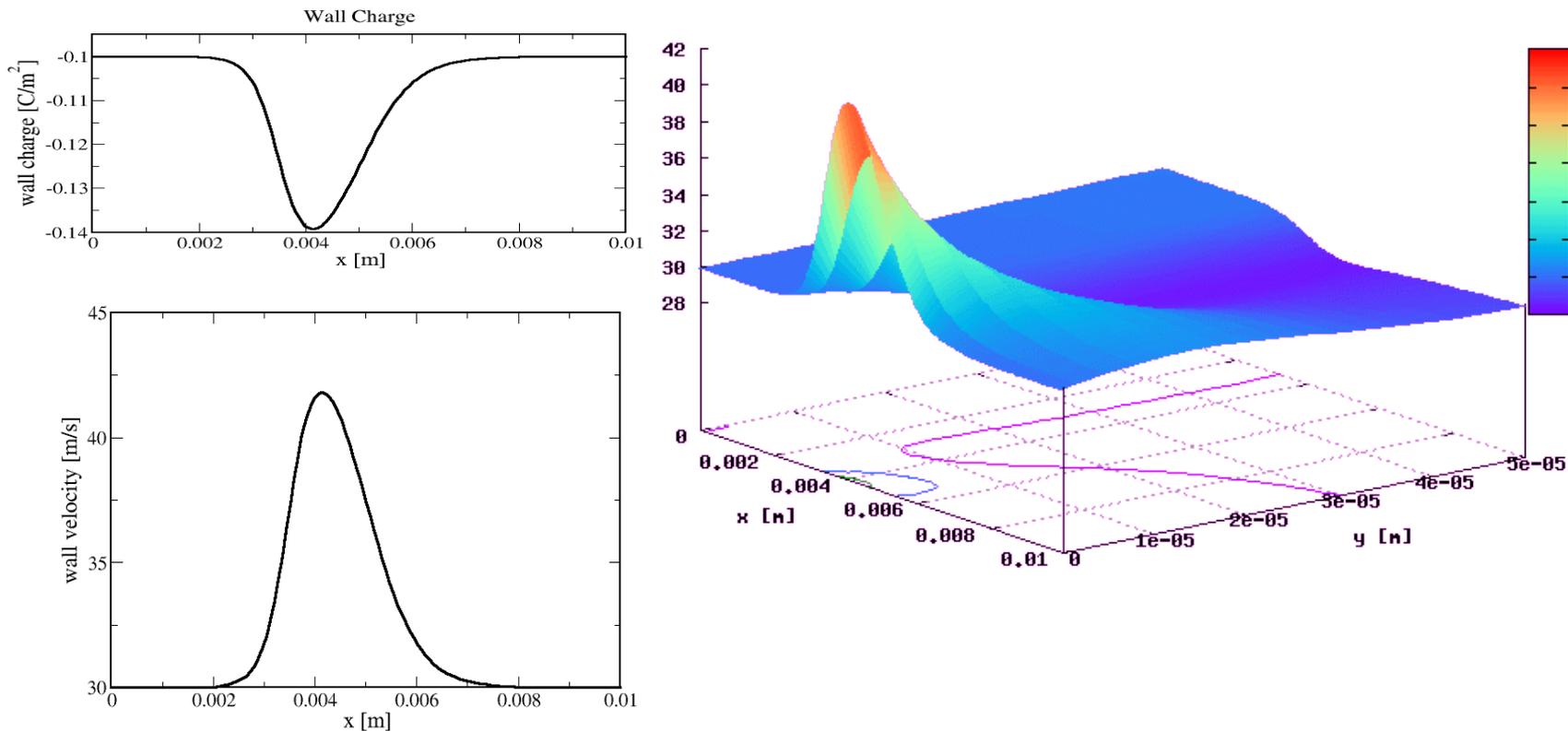
# Example: Chemical variations in a microfluidic channel



- Flow irregularities result in band spreading, causing reduced sensitivity
- Overall flow profile is a result of interplay between bulk solution and wall layer
- Characteristic size- and time scales span roughly nine orders of magnitude.



# Flow profile for chemical variation at microchannel wall

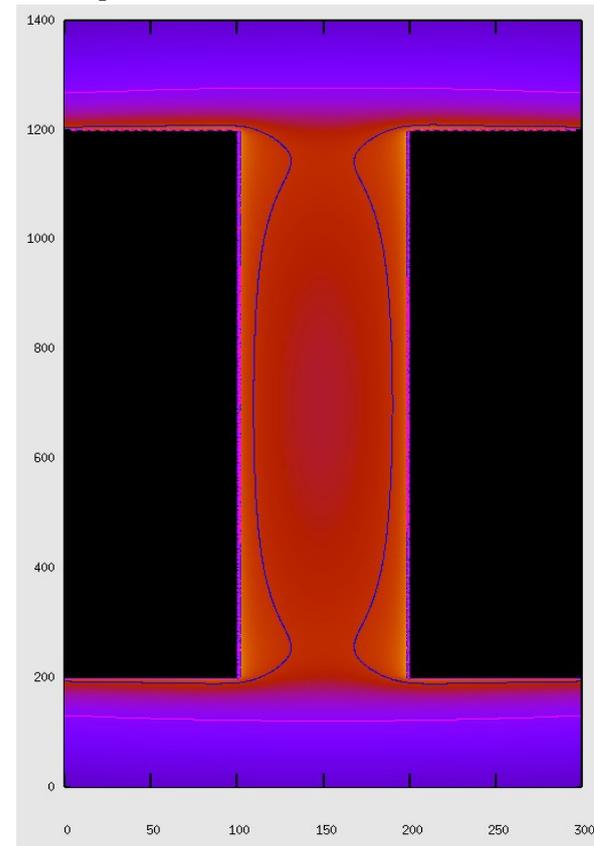
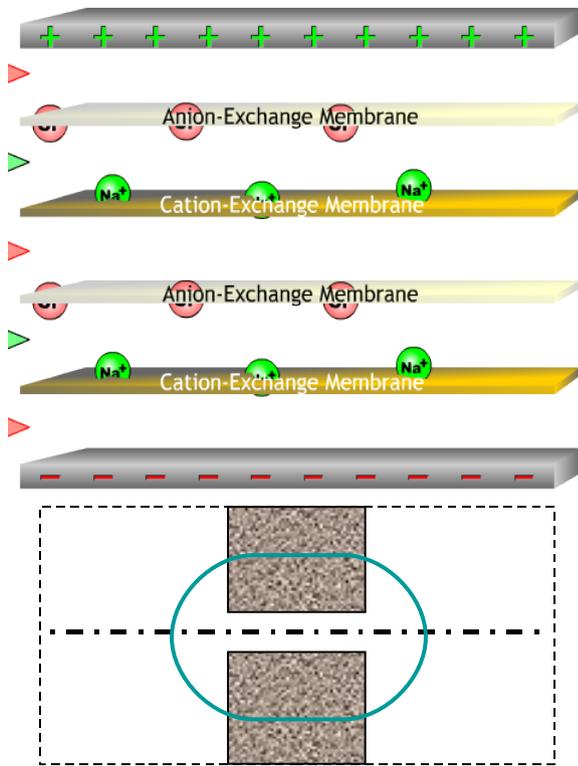


Whole-system flow profile obtained particle simulations

System-scale propagation handled by continuum modeling

New simulation capability for scientific insight into electro-osmotic flow

# Example: Nanopore separations



- Transport efficiency and selectivity arise from membrane chemistry, electrostatic fields, and ion distribution in solution
- Characteristic size- and time scales span six to nine orders of magnitude



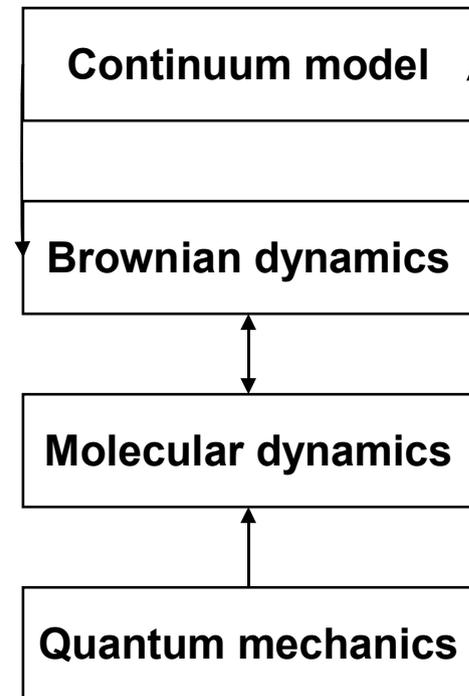
# Summary

- We need multiscale simulations
  - There is a host of research waiting for suitable simulation capabilities
- We are better suited than most for this field
  - We have the people, the algorithms, and the experience
- We should be building toward a multi-disciplinary, multi-center research thrust similar to what was done for combustion, materials, and MEMS



# Heterogeneous multiscale simulations

- **Concurrent**
  - Models are evolved together
  - Scales overlap appreciably
- **Cooperative**
  - Dynamic parameterization
  - Scales may differ significantly
- **Sequential**
  - Up-front parameterization
  - Separation of scales unimportant



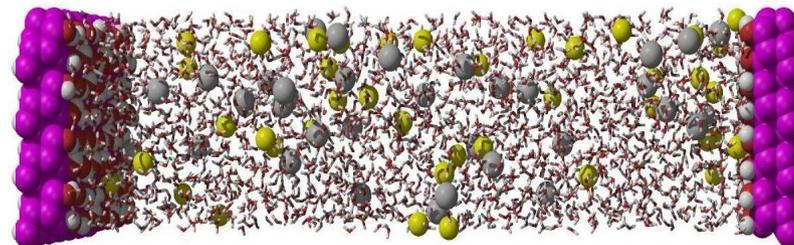
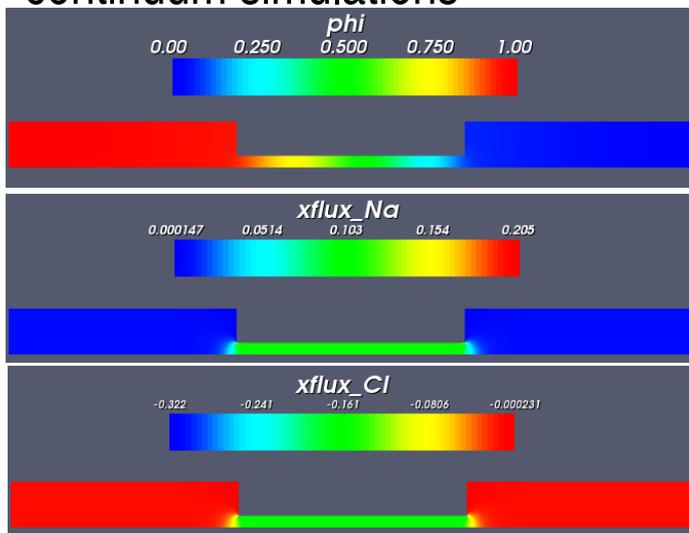
# Nanopore efficiency, selectivity, and performance limitations

Wall effects modeled using  
molecular dynamics

Detailed flow through Brownian  
dynamics

Whole-system evolution through  
continuum simulations

QuickTime® and a  
decompressor  
are needed to see this picture.





# Summary

- Petascale complexity
- Better asynchronous message passing models
- Mixed programming models MPI+threading
- Chip level parallel challenges are not unique to HPC community
- Component based SW models reduce complexity at the human level
- Develop common API's to enable multiscale multi-disciplinary simulations
- Develop higher level scientific tools (Sundance)
- Take advantage of massively parallel chips with new threaded programming models

# Notes

- Petascale complexity is ... system and human scalability and performance issue
- Manage complexity by dealing with scalability problems
  - Computer science (scalable architecture and system software)
  - Scientists perspective (human scalability)
- Intel 80 proc chip 1 Tflop
- Programming models
  - Active messages with Qlogic
  - Examples of MPQC with mpipro
  - Cray MTA?
- Rapid prototyping and development environments (Sundance example)
- Common Component Architecture (CCA) to interface between multiscale apps
  - Making people scalable rather than processors; deals with complexity at human level
- Python
  - and other scripting languages are examples of managing complexity by gluing stuff together
  - Ruby, perl, bash, etc
- What is locality? Locality now has multiple levels (is hierarchical)
- Sundance
  - Example of how complexity can be reduced
- Make as simple as it possibly can be but no simpler
- MPQC Blue Gene port?
- How do we design scalable algorithms as the notion of data locality continues to be blurred