Many-Task Computing Tools for Multiscale Modeling

Daniel S. Katz (d.katz@ieee.org)

Senior Fellow, Computation Institute (University of Chicago & Argonne National Laboratory)

Adjunct Associate Professor, Electrical and Computer Engineering, Louisiana State University

Area Co-Director for Applications, Open Grid Forum

www.ci.anl.gov
www.ci.uchicago.edu

THE UNIVERSITY OF CHICAGO

Founded in 1892 as an institutional "community of scholars," where all fields and disciplines meet and collaborate

Computation Institute

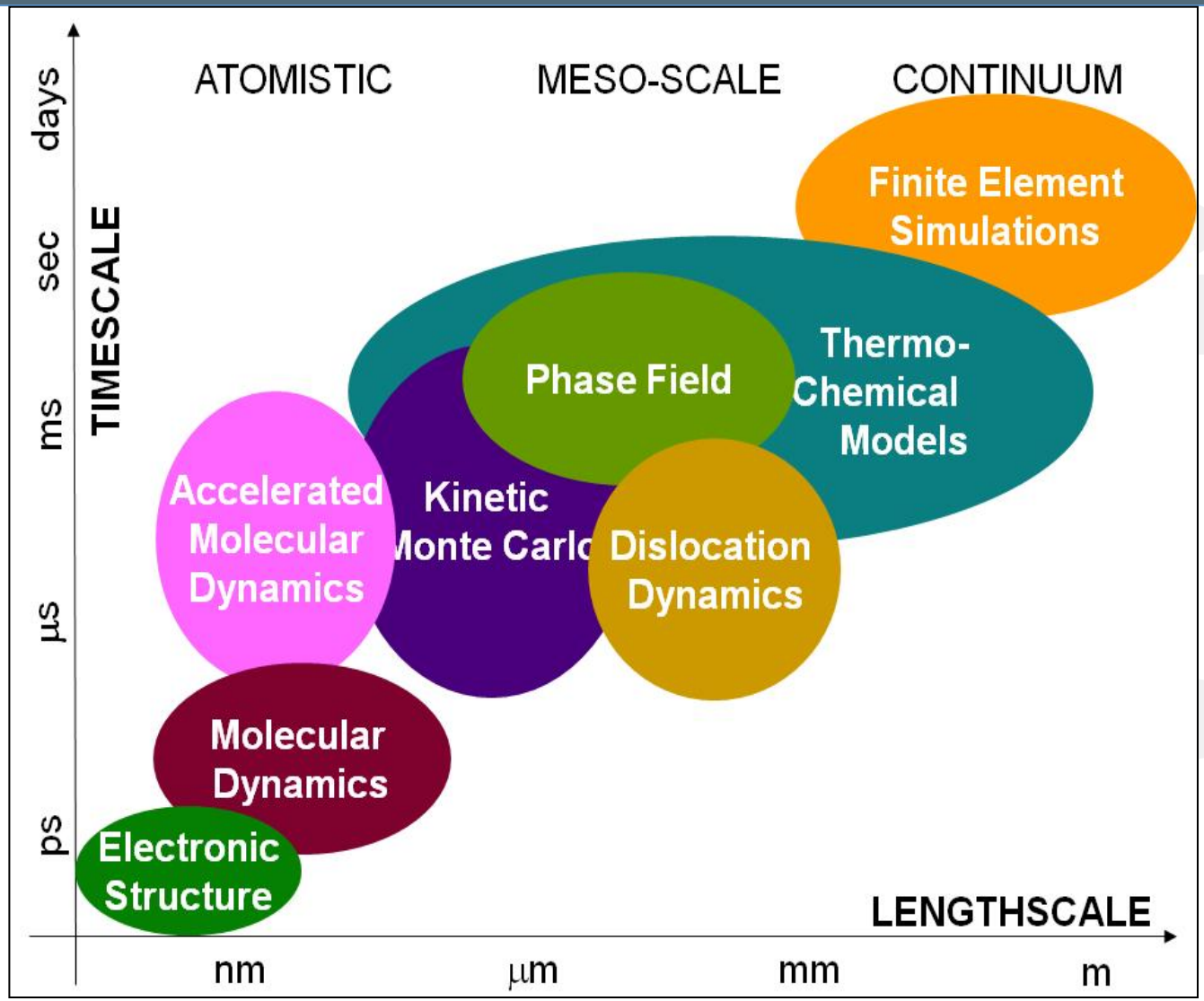
- Mission: address the most challenging problems arising in the use of strategic computation and communications
- Joint Argonne/UChicago institute, ~100 Fellows (~50 UChicago faculty) & ~60 staff
- Primary goals:
  - Pursue new discoveries using multi-disciplinary collaborations and computational methods
  - Develop new computational methods and paradigms required to tackle these problems, and create the computational tools required for the effective application of advanced methods at the largest scales
  - Educate the next generation of investigators in the advanced methods and platforms required for discovery

Multiscale Modeling

Multiscale

- The world is multiscale
- In modeling, a common challenge is determining the correct scale to capture a phenomenon of interest
  - In computer science, a parallel problem is describing a problem with the right level of abstraction
    - Capture the details you care about and ignore those you don't
- But multiple phenomena interact, often at different scales

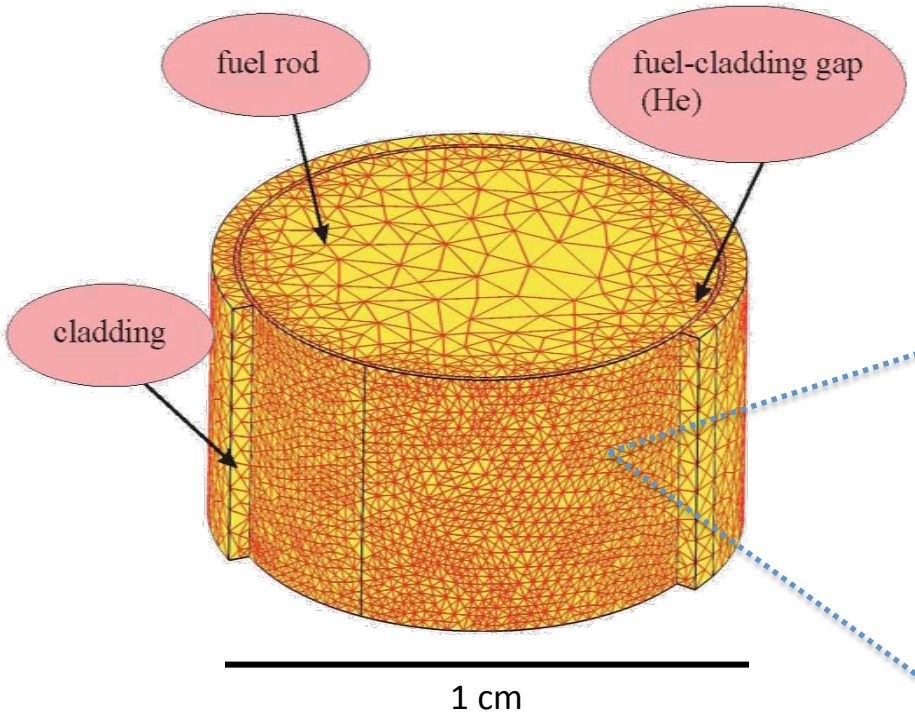
Material Science, Methods and Scales



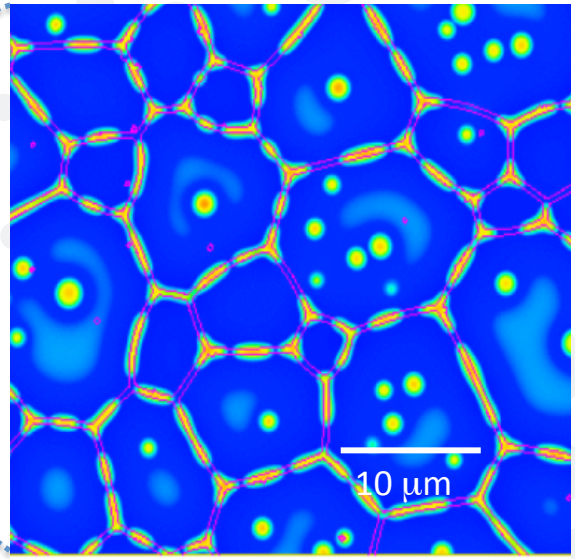
Credit: M. Stan, Materials Today, 12 (2009) 20-28

Modeling nuclear fuel rods

Fuel Element Model (FEM)



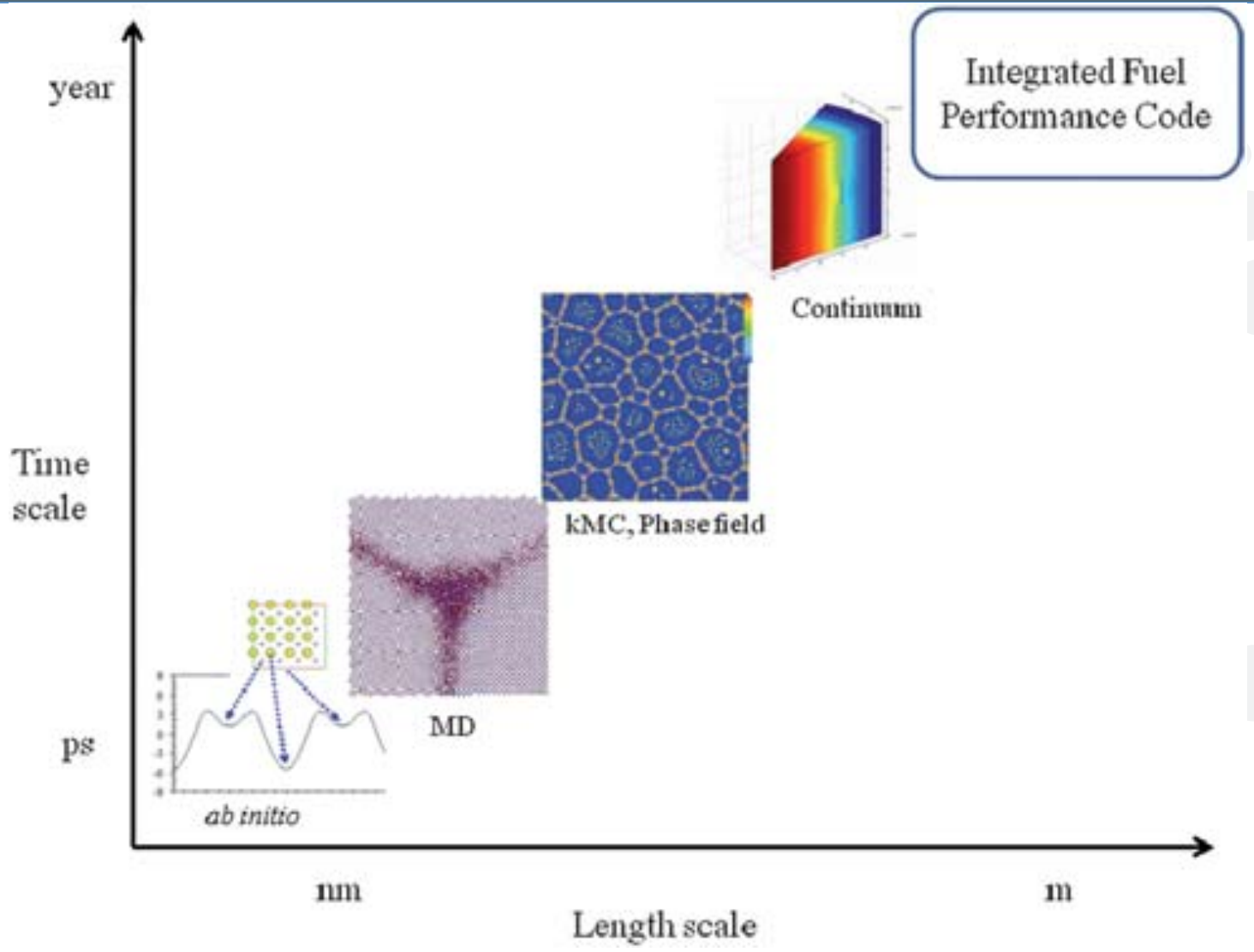
Microstructure Model (PF)



B. Mihaila, et al., J. Nucl. Mater., **394** (2009) 182-189

S.Y. Hu et al., J. Nucl. Mater. **392** (2009) 292–300

Sequential (information passing, hand-shaking, bridging)



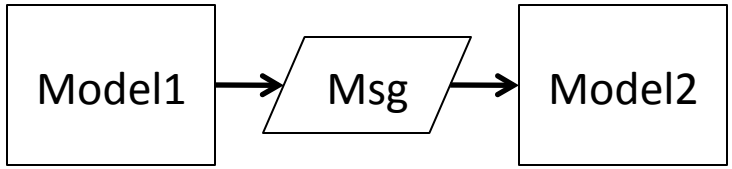
R. Devanathan, et al., Energy Env. Sc., **3** (2010) 1406-1426

Coupling methods

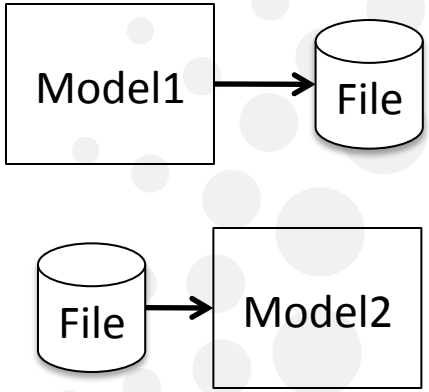
- We often know how to solve a part of the problem with sufficient accuracy, but when we combine multiple parts of the problem at various scales, we need to couple the solution methods too
- Must first determine the models to be run and how they iterate/interact
- Coupling options
  - "Manual" coupling (sequential, manual)
    - Inputs to a code at one scale are influenced by study of the outputs of a previously run code at another scale
    - Coupling timescale: hours to weeks
  - "Loose" coupling (sequential, automated) between codes
    - Typically performed using workflow tools
    - Often in different memory spaces
    - Coupling timescale: minutes
  - "Tight" coupling (concurrent, automated) between codes
    - e.g., ocean-atmosphere-ice-bio
    - Typically performed using coupling methods (e.g., CCA), maybe in same memory space
    - Hard to develop, changes in one code may break the system
    - Coupling timescale: seconds
- Boundary between options can be fuzzy
- Choice often depends on how frequently the interactions are required, and how much work the codes do independently

More on coupling

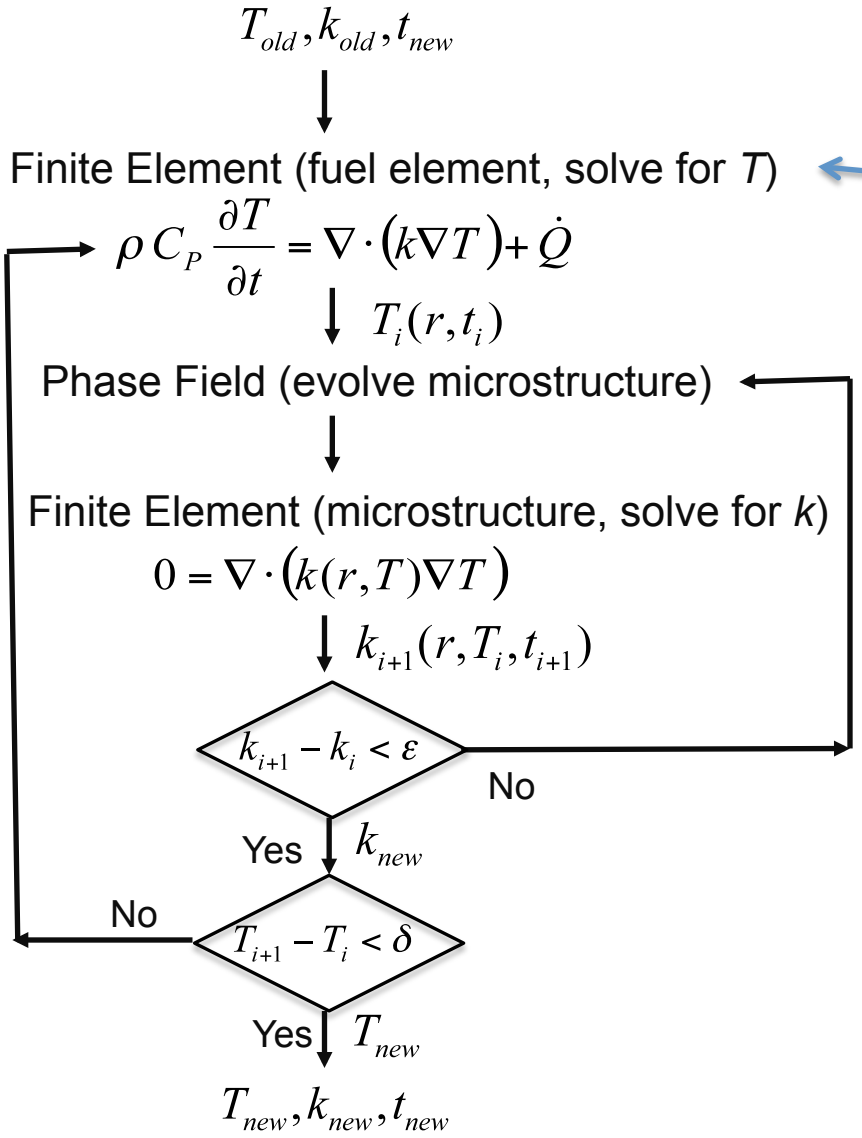
Tight

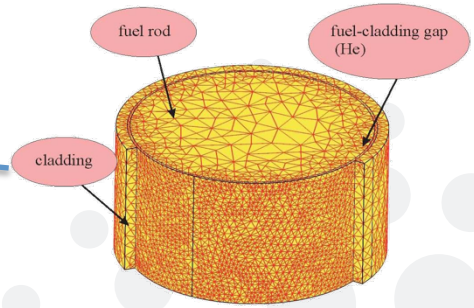


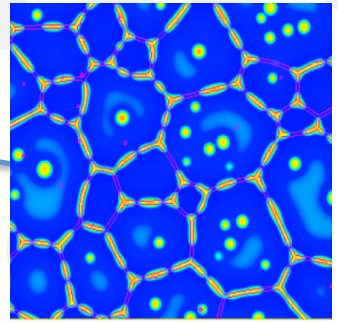
Loose



Coupling Models



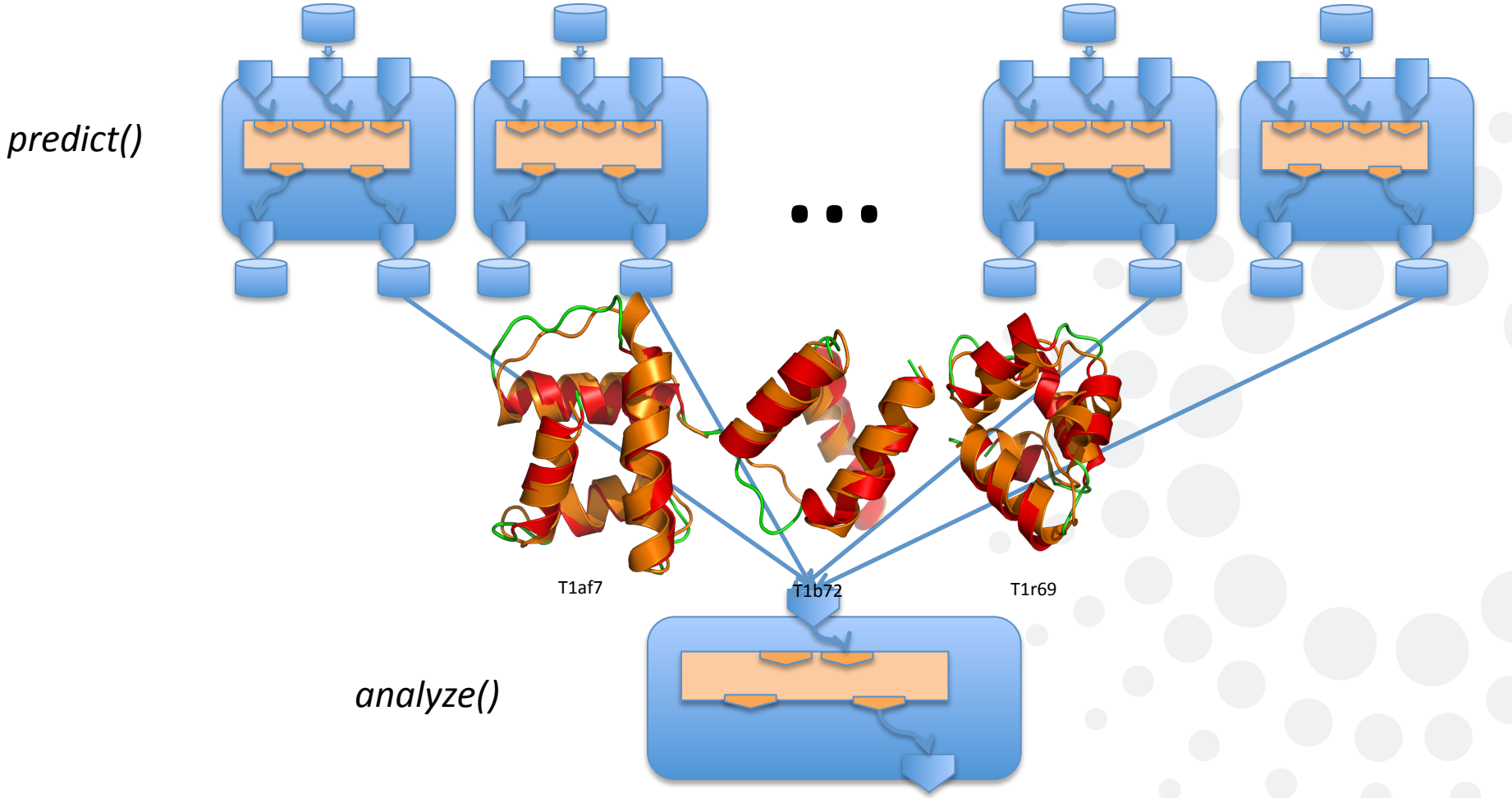




Credit: Marius Stan, Argonne

Swift (for Loose Coupling)

Workflow Example: Protein Structure Prediction



Want to run: 10 proteins x 1000 simulations x
3 MC rounds x 2 temps x 5 deltas = 300K tasks

Swift

- Portable workflows – deployable on many resources
- Fundamental script elements are external processes and data files
- Provides natural concurrency at runtime through automatic data flow analysis and task scheduling
- Data structures and script operations to support scientific computing
- Provenance gathered automatically

Portability: dynamic development and execution

- Separate workflow description from resource and component implementations

<sites.xml>
…

```
rawdata = sim(settings);
stats = analysis(rawdata);
…
```

select resources

write script

allocate resources

Execute
(Swift)

<tc.data>
…

define
components

Swift scripts

- C-like syntax, also Python prototype
- Supports file/task model directly in the language

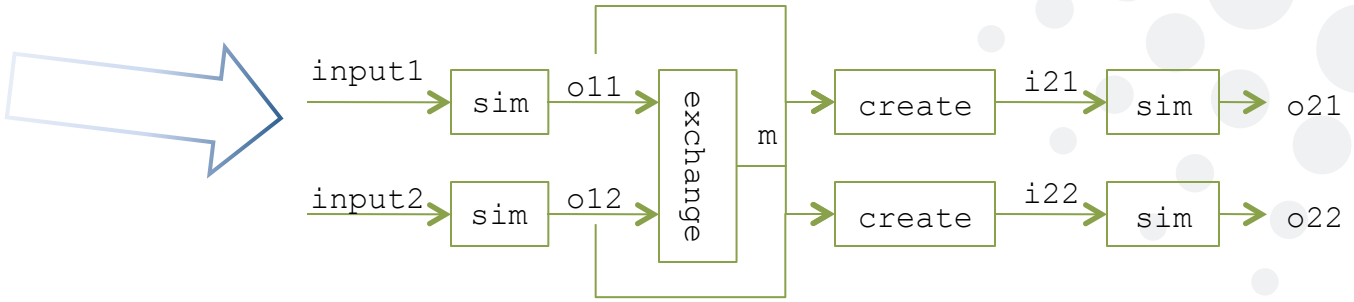
```
type file;

app (file output) sim(file input) {
  namd2 @input @output
}
```

- Historically, most tasks have been sequential applications

Data flow and natural concurrency

- Provide natural concurrency through automatic data flow analysis and task scheduling

```
file o11 = sim(input1);
file o12 = sim(input2);
file m   = exchange(o11, o12);
file i21 = create(o11, m);
file o21 = sim(i21);
...
```

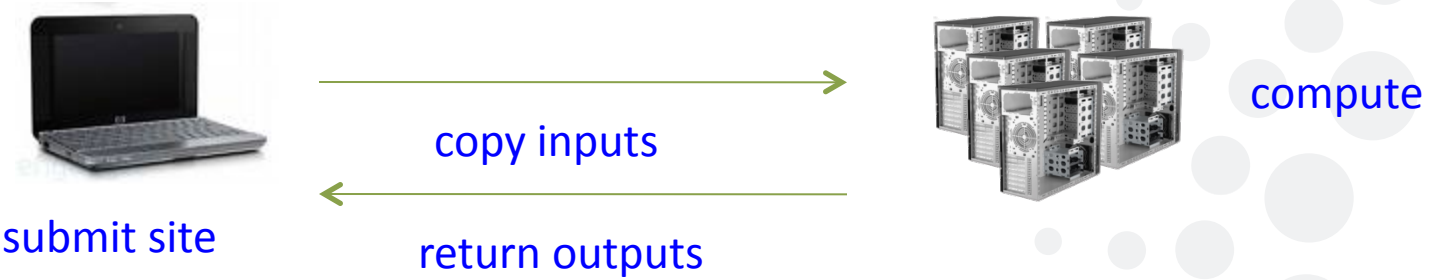


Variables, Tasks, Files, Concurrency

- Variables are single assignment futures
  - Unassigned variables are open
- Variables can represent files
  - When a file doesn't exist, the variable is open
  - When a file exists, the variable is closed
- All tasks found at runtime
- Tasks with satisfied dependencies (closed variables) are run on whatever resources are available
- These runs create files/variables that allow more tasks to run

Execution model

- In a standard Swift workflow, each task must enumerate its input and output files
- These files are shipped to and from the compute site

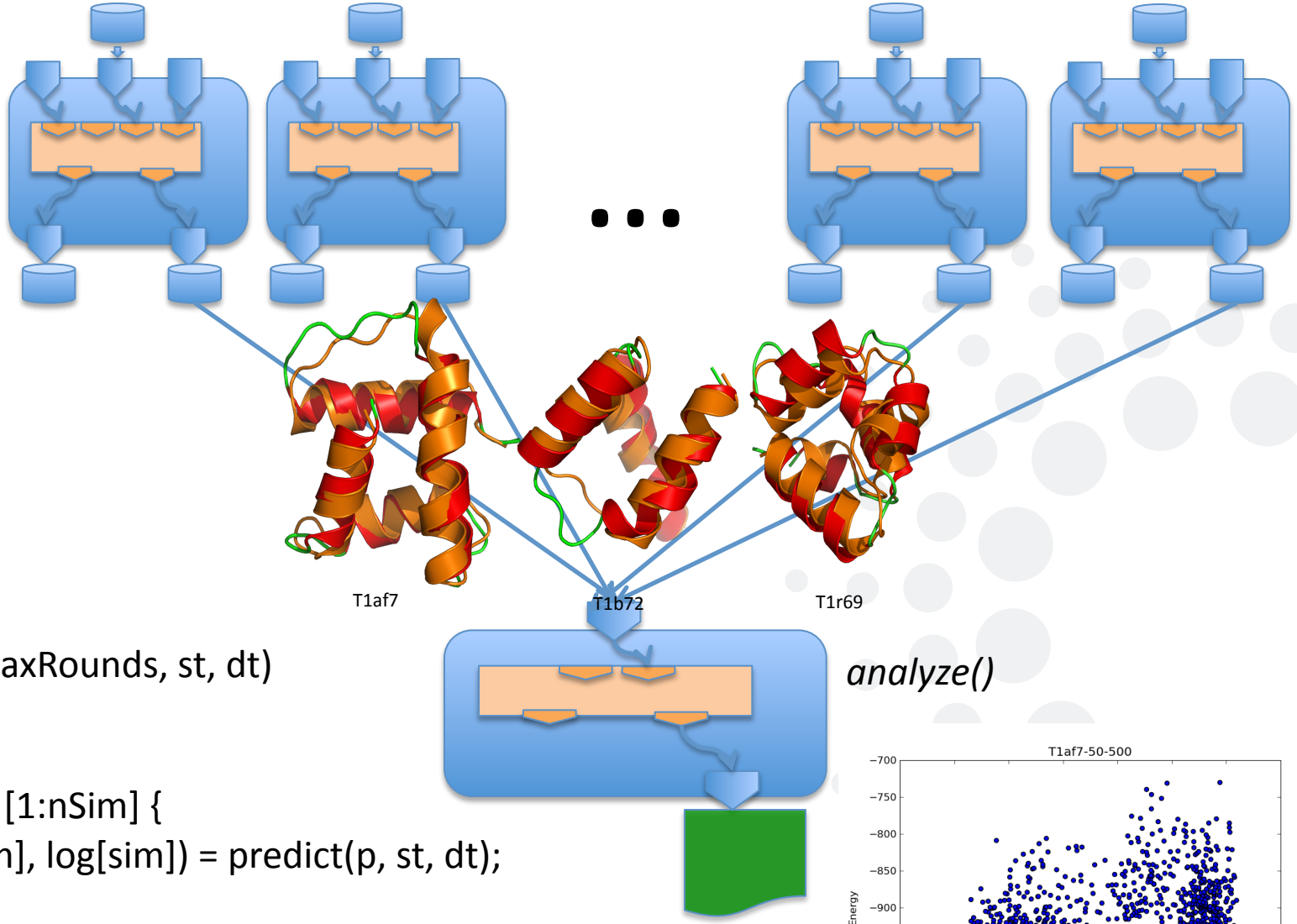


- RPC-like technique, can use multiple queuing systems, data services, and execution environments
  - Uses abstractions for file transfer, job execution, etc.
  - Allows use of local systems (laptop, desktop), parallel systems (HPC), distributed systems (HTC, clouds)
  - Supports grid authentication mechanisms
- Can use multi-level scheduling (Coasters) – alleviates need for reservations?

Performance and Usage

- Swift is fast
  - Uses Karajan (in Java CoG) as powerful, efficient, scalable, and flexible execution engine
  - Scaling close to 1M tasks; .5M in live science work, and growing
- Swift usage is growing (~300 users in last year):
  - Applications in neuroscience, proteomics, molecular dynamics, biochemistry, climate, economics, statistics, astronomy, etc.
  - And earthquake modeling (to be discussed in seasonal school Wed.)

nSim (1000) predict()s




```
ItFix(p, nSim, maxRounds, st, dt)
{
  ...
  foreach sim in [1:nSim] {
    (structure[sim], log[sim]) = predict(p, st, dt);
  }
  result = analyze(structure)
  ...
}
```

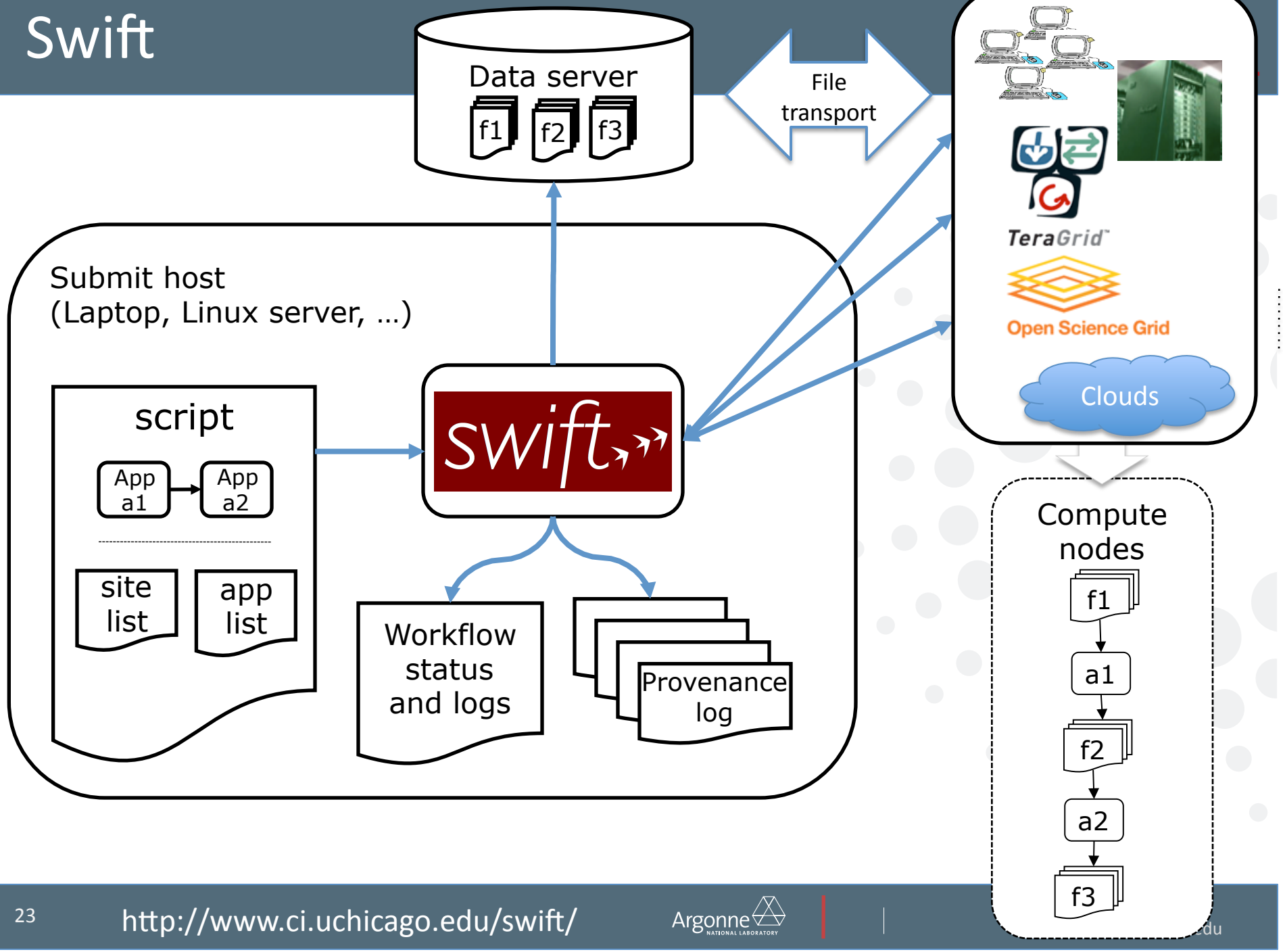
analyze()

Powerful parallel prediction loops in *Swift*

```
1.  Sweep( )
2.  {
3.    int nSim = 1000;
4.    int maxRounds = 3;
5.    Protein pSet[ ] <ext; exec="Protein.map">;
6.    float startTemp[ ] = [ 100.0, 200.0 ];
7.    float delT[ ] = [ 1.0, 1.5, 2.0, 5.0, 10.0 ];
8.    foreach p, pn in pSet {
9.      foreach t in startTemp {
10.       foreach d in delT {
11.         ItFix(p, nSim, maxRounds, t, d);
12.       }
13.     }
14.   }
15. }
```

10 proteins x 1000 simulations x
3 rounds x 2 temps x 5 deltas
= 300K tasks

Swift



http://www.ci.uchicago.edu/swift/

Swift summary

- Structures and arrays of data
- Typed script variables intermixed with references to file data
- Natural concurrency
- Integration with schedulers such as PBS, Cobalt, SGE, GT2, …
- Advanced scheduling settings
- A variety of useful workflows can be considered

Using Swift for Loose Coupling

Multiscale Molecular Dynamics

- Problem: many systems are too large to solve using all-atom molecular dynamics (MD) models
- Potential solution: coarse-grained (CG) models where each site represents multiple atoms
- In order to do this, have to decide how to coarsen the model
  - How many sites are needed?
  - Which atoms are mapped to which sites?
  - What is the potential energy as a function of coordinates of those CG sites?

Credit: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago

Building a CG model – initial data processing

- Stage 0 – run multiple short-duration trajectories of all-atom MD simulation, e.g., using NAMD, capture dcd files
  - Can require large run time and memory, so run on TeraGrid system
  - Download (binary) dcd files to local resources for archiving
  - Remove light atoms (e.g., water, H)
  - Performed manually
- Stage 1 – remove non α-Carbon atoms on a subset of the dcd files from each trajectory
  - Need to know how many steps were in each trajectory – not always what was planned, and final file may be corrupt, so some manual checking needed
  - Performed by fast Tcl script

Credit: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago

Building CG model – covariance matrix

- Stage 2 – join trajectory files together into ascii file
  - Requires trajectory length from previous stage
  - Performed by fast Tcl script
- Stage 3 – generate covariance matrix for each trajectory
  - Find deviation of each atom from its average position across all time steps
  - Covariance matrix determines which atoms can be grouped into rigid bodies (roughly)
  - Performed by shell script that runs a compiled C code
    - Takes several hours per trajectory

Credit: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago

Building a CG model – CG mapping

- Stage 4 – for a given number of sites (#sites), find best mapping for each trajectory
  - Pick 3 to 5 values for #sites that should cover the likely best value
  - For each #sites, can find χ^2 value for each mapping
  - Overall, want lowest χ^2 and corresponding mapping
  - Uses a group of random initial values and simulated annealing from each
  - Performed by shell script to launch compiled C code, O(50k) trials, takes several days on 100-1000 processors
- Stage 5 – check χ^2 values for each trajectory
  - χ^2 vs. #sites on a log-log plot should be linear
  - Performed by script
  - If a point is not close to the line, it's probably not a real minimum χ^2 for that #sites
    - Go back to Stage 4 – run more initial case to get a lower χ^2

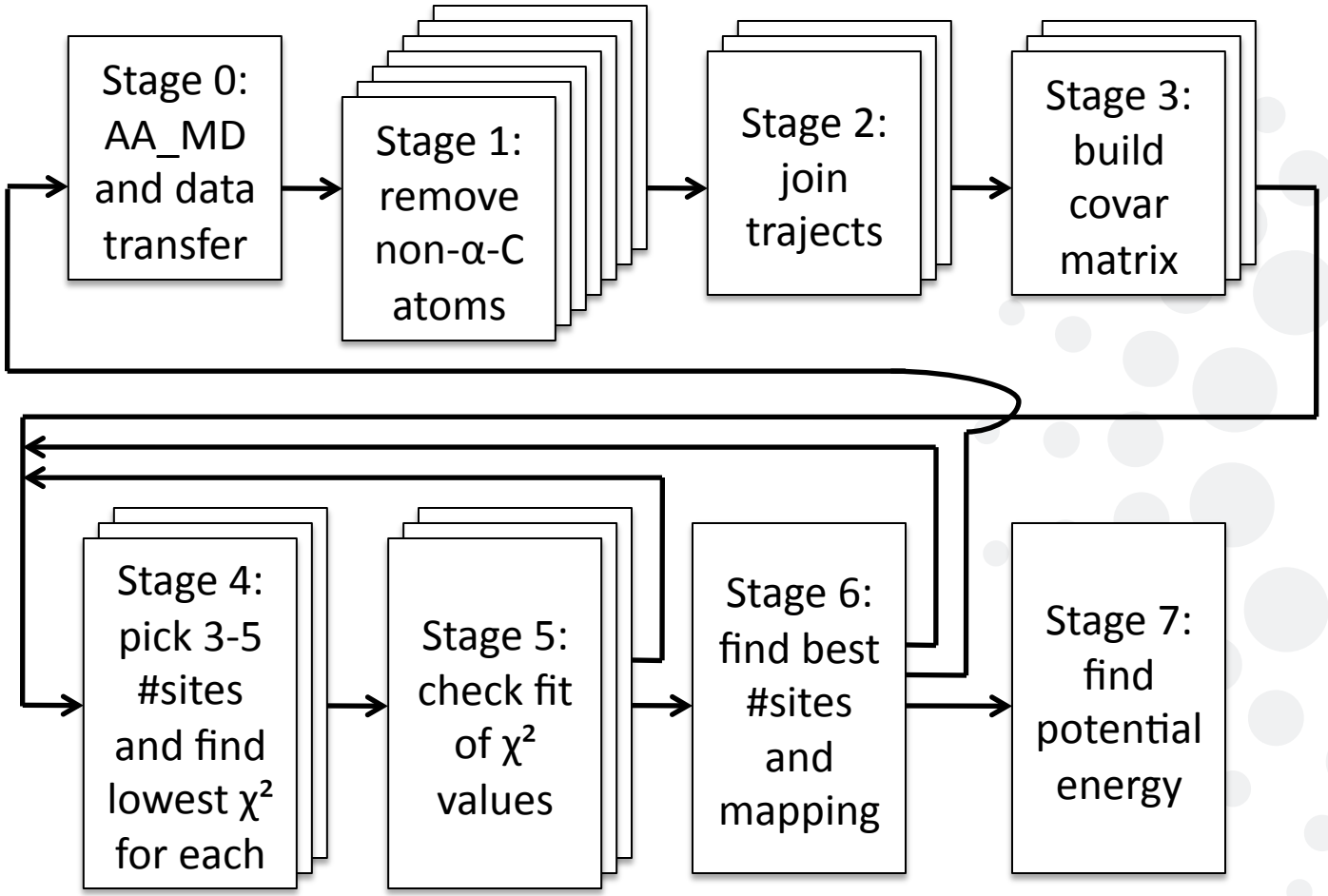
Credit: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago

Building a CG model – finding #sites

- Stage 6 – determine #sites
  - Estimate best #sites (b#sites) from slope/intercept of line in stage 5, and compare results of all trajectories
  - Performed by script
  - If results for each trajectory are different, trajectories didn't sample enough of the phase space – go back to Stage 0 and run more/longer trajectories
  - If b#sites is outside the range of #sites that have been calculated, add to initial range and go back to Stage 4
  - If b#site is inside the range, create a smaller range around b#sites and go back to Stage 4
    - b#sites is an integer, so don't have to do this too much
  - Outputs final b#sites and corresponding mapping
- Stage 7 – building potential energy as function of site coordinates
  - Can be done by different methods, e.g., Elastic Network Models (ENM)
    - Currently under construction

Credit: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago

Bio workflow: AA->CG MD



Credit: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago

Multiscale?

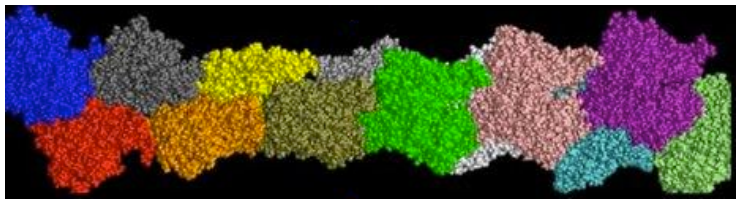
- So far, this isn't really multiscale
- It has just used fine grain information to build the best coarse grained model
- But it's a needed part of the process
- Overall, can't run AA_MD as much as desired.
  - Here, limited AA_MD simulations -> structural information for a rough CG model of the internal molecular structure
  - With rough CG model, user can parameterize interactions for CG "atoms" via targeted all-atom simulations -> determine average energies and forces etc. for the CG beads
- Doing this automatically is a long-term goal

Credit: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago

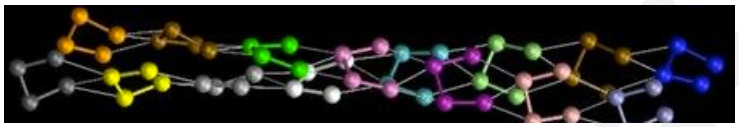
NSF Center for Chemical Innovation Phase I award: "Center for Multiscale Theory and Simulation"

- ... development of a novel, powerful, and integrated theoretical and computational capability for the description of biomolecular processes across multiple and connected scales, starting from the *molecular scale and ending at the cellular scale*
- Components:
  - A theoretical and computer **simulation capability to describe biomolecular systems at multiple scales** will be developed, includes atomistic, coarse-grained, and mesoscopic scales ... all scales will be connected in a multiscale fashion so that key information is passed upward in scale and vice-versa
  - Latest generation scalable computing and **a novel cyberinfrastructure** will be implemented
  - A high profile demonstration projects will be undertaken using the resulting theoretical and modeling advances which involves the **multiscale modeling of the key biomolecular features of the eukaryotic cellular cytoskeleton (i.e., actin-based networks and associated proteins)**
- Core CCI team includes a diverse group of leading researchers at the University of Chicago from the fields of theoretical/computational chemistry, biophysics, mathematics, and computer science:
  - Gregory A. Voth (PI, Chemistry, James Franck Institute, Institute for Biophysical Dynamics, Computation Institute); Benoit Roux (co-PI, Biochemistry and Molecular Biology, Institute for Biophysical Dynamics); Nina Singhal Hinrichs (co-PI, Computer Science and Statistics); Aaron Dinner (co-PI, Chemistry, James Franck Institute, Institute for Biophysical Dynamics); Karl Freed (co-PI, Chemistry, James Franck Institute, Institute for Biophysical Dynamics); Jonathan Weare (co-PI, Mathematics); Daniel S. Katz (Senior Personnel, Computation Institute)

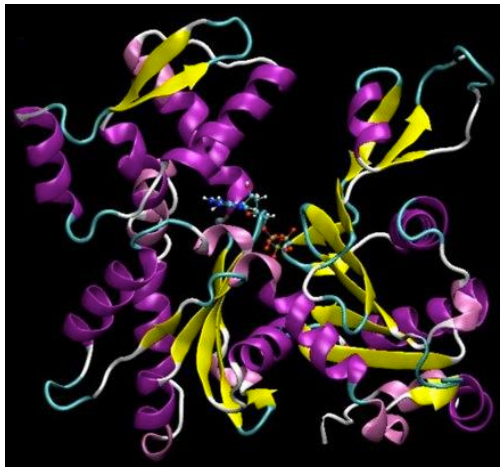
Actin at multiple scales



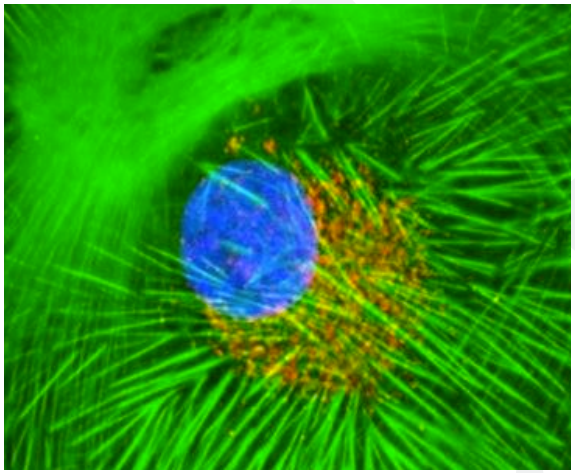
Actin filament (F-Actin) – complex of G-Actins



Actin filament (F-Actin) – CG representation



Single Actin monomer (G-Actin) – all-atom representation



Actin in cytoskeleton – mesoscale representation

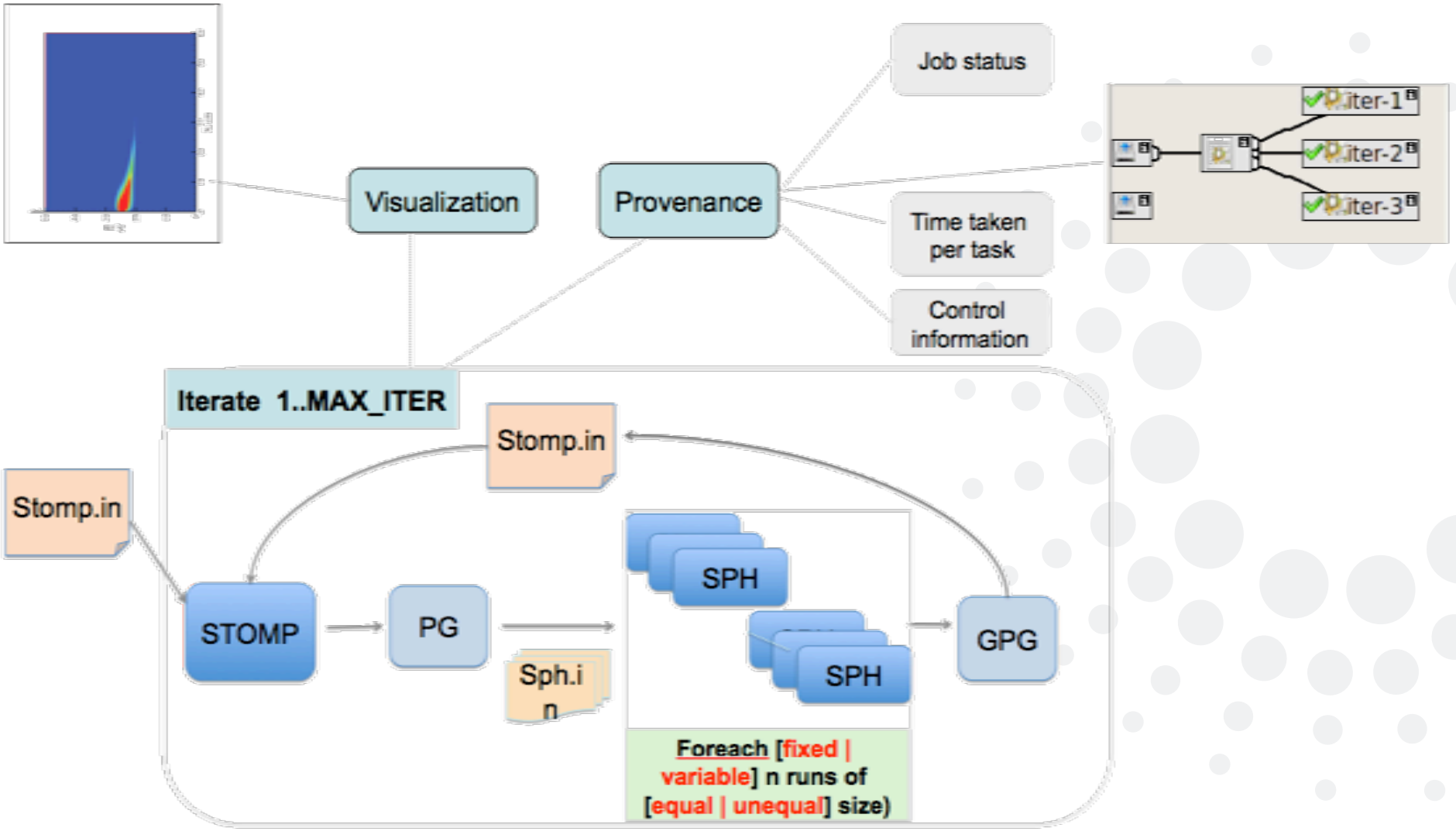
Credit: Greg Voth, U. Chicago

Geophysics Application

- Subsurface flow model
- Couples continuum and pore scale simulations
  - Continuum model: exascale Subsurface Transport Over Multiple Phases (eSTOMP)
    - Scale: meter
    - Models full domain
  - Pore scale model: Smoothed Particle Hydrodynamics (SPH)
    - Scale: grains of soil (mm)
    - Models subset of domain as needed
- Coupler codes developed
  - Pore Generator (PG) – adaptively decides where to run SPH and generates inputs for each run
  - Grid Parameter Generator (GPG) – uses outputs from SPH to build inputs for next eSTOMP iteration

Credit: Karen Schuchardt , Bruce Palmer, Khushbu Agarwal, Tim Scheibe, PNNL

Subsurface Hybrid Model Workflow



Credit: Karen Schuchardt , Bruce Palmer, Khushbu Agarwal, Tim Scheibe, PNNL

Swift code

```
// Driver
file stompIn <"stomp.in">;

iterate iter {
        output = HybridModel(inputs[iter]);
        inputs[iter+1] = output;
        capture_provenance(output);

} until(iter >=  MAX_ITER);
```

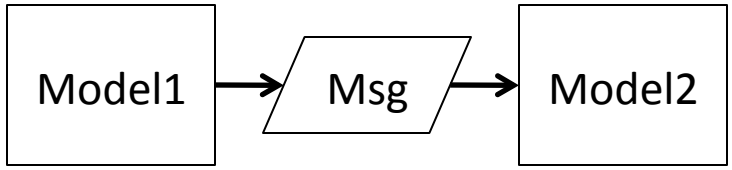
```
//Hybrid Model

(file  simOutput) HybridModel (file input) {
    …
     stompOut = runStomp(input);

    (sphins, numsph) = pg(stompOut, sphinprefix);

     //Find number of pore scale runs
     int n = @toint(readData(numsph));
      foreach i in [1:@toint(n)]  {
         sphout[i]= runSph(sphins[i],  procs_task)
      }

    simOutput = gpg(sphOutArr, n, sphout);
}
```

Credit: Karen Schuchardt , Bruce Palmer, Khushbu Agarwal, Tim Scheibe, PNNL

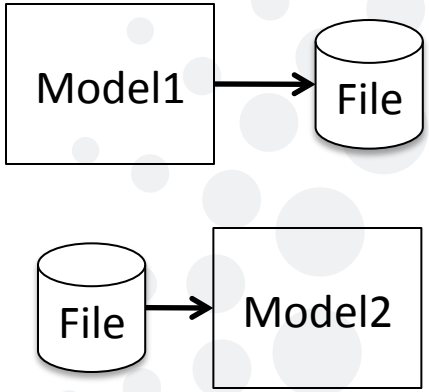
Towards Tighter Coupling

More on coupling

Tight



Loose



More on coupling

- Message vs. file issues:
  - Performance: overhead involved in writing to disk vs. keeping in memory
  - Semantics: messages vs. Posix
  - Fault tolerance: file storage provides an automatic recovery mechanism
  - Synchronicity: messages can be sync/async, files must be async
- Practical issues:
  - What drives the application?
    - Loose case: A driver script calls multiple executables in turns
    - Tight case: No driver, just one executable
  - What's the cost of initialization?
    - Loose case: Executables initialized each time
    - Tight case: All executables exist at all times, only initialized once
  - How much can components be overlapped?
    - Loose case: If all components need the same number of resources, all resources can be kept busy all the time
    - Tight case: Components can be idle waiting for other components

Work in progress towards tighter coupling in Swift: Collective Data Management (CDM)

- Data transfer mechanism is: transfer input, run, transfer output
- Fine for single node systems, could be improved to take advantage of other system features, such as intermediate file system (or shared global file system on distributed sites)
- Define I/O patterns (gather, scatter, broadcast, etc.) and build primitives for them
- Improve support for shared filesystems on HPC resources
- Make use of specialized, site-specific data movement features
- Employ caching through the deployment of distributed storage resources on the computation sites
- Aggregate small file operations into single larger operations

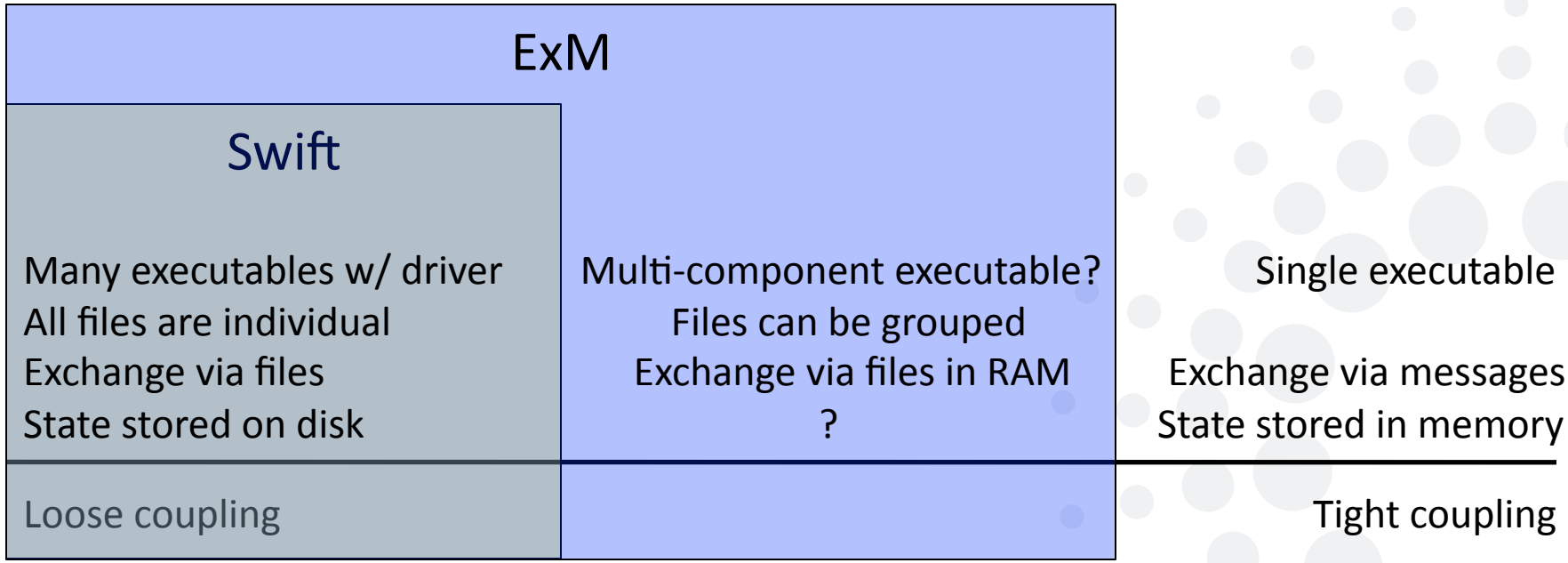
CDM examples

- Broadcast an input data set to workers
  - On Open Science Grid, just send it to the shared file system of each cluster once, the let worker nodes copy it from there
  - On IBM BG/P, use intermediate storage on I/O nodes on each pset similarly
- Gather an output data set
  - Rather than sending each job's output, if multiple jobs are running on a node and sufficient jobs are already runnable, wait and bundle multiple output files, then transfer bundle

Work in progress towards tighter coupling after Swift: ExM (Many-task computing on extreme-scale systems)

- Deploy Swift applications on exascale-generation systems
- Distributed task (and function management)
  - Break the bottleneck of a single execution engine
  - Call functions, not just executables
- JETS: Dynamically run multiple MPI tasks on an HPC resource
  - Allow dynamic mapping of workers to resources – both ways
  - Add resilience – allow mapping of workers to dynamic resources
- MosaStore: intermediate file storage
  - Use files for message passing, but stripe them across RAMdisk on nodes (single distributed filesystem w/ shared namespace), backing store in shared file system, potentially cache in the middle
- AME: intermediate file storage
  - Use files for message passing, but store them in RAMdisk on nodes where written (multiple filesystems w/ multiple namespaces), copy to new nodes when needed for reading

Increased coverage of scripting



- Questions:
  - Will we obtain good-enough performance in ExM?
  - How far can we go towards the tightly-coupled regime without breaking the basic Swift model?

Conclusions

- Multiscale modeling is important now, and use will grow
- Can think of multiscale modeling instances on a spectrum of loose to tight coupling
- Swift works for loose coupling
  - Examples shown for nuclear energy, biomolecular modeling, and subsurface flows
- Improvements in Swift (and ExM) will allow it to be used along more of the spectrum

Acknowledgments

- Swift is supported in part by NSF grants OCI-721939, OCI-0944332, and PHY-636265, NIH DC08638, DOE and UChicago SCI Program

- http://www.ci.uchicago.edu/swift/
- The Swift team:
  - Mike Wilde, Mihael Hategan, Justin Wozniak, Ketan Maheshwari, Ben Clifford, David Kelly, Allan Espinosa, Ian Foster, Ioan Raicu, Sarah Kenny, Zhao Zhang, Yong Zhao, Jon Monette, Daniel S. Katz

- ExM is supported by the DOE Office of Science, ASCR Division
  - Mike Wilde, Daniel S. Katz, Matei Ripeanu, Rusty Lusk, Ian Foster, Justin Wozniak, Ketan Maheshwari, Zhao Zhang, Tim Armstrong, Samer Al-Kiswany, Emalayan Vairavanathan
- Scientific application collaborators and usage described in this talk:
  - Material Science: Marius Stan, Argonne
  - Biomolecular Modeling: Anton Sinitskiy, John Grime, Greg Voth, U. Chicago
  - Subsurface Flows: Karen Schuchardt , Bruce Palmer, Khushbu Agarwal, Tim Scheibe, PNNL

- Thanks! – questions now or later – d.katz@ieee.org

8-12 October 2012 Chicago, IL, USA

http://www.ci.uchicago.edu/escience2012/

THE UNIVERSITY OF CHICAGO

Argonne NATIONAL LABORATORY

eScience Conference main events Wednesday – Friday
(keynotes, papers, panels, posters)

Paper submissions due 11 July; posters due 24 Aug

Microsoft eScience Workshop Monday – Tuesday

Additional eScience workshops Monday – Tuesday

Open Grid Forum OGF36 Monday – Wednesday

GLIF Annual Meeting Thursday – Friday

General Chair Ian Foster

Program Co-Chairs Daniel S. Katz, Heinz Stockinger

Program Vice Co-Chairs David Abramson, Gabrielle Allen, Rosa M. Badia, Geoffrey Fox

Early Results and Works-in-Progress Posters Chair Roger Barga

Workshops Chair Ruth Pordes

Sponsorship Chair Charlie Catlett

Conference Manager and Finance Chair Julie Wulf-Knoerzer

Publicity Chairs Kento Aida, Ioan Raicu, David Wallom

Conference Tracks:

eScience Algorithms and Applications

- eScience application areas, including:
  - Physical sciences
  - Biomedical sciences
  - Social sciences and humanities
- Data-oriented approaches and applications
- Compute-oriented approaches and applications
- Extreme scale approaches and applications

Cyberinfrastructure to support eScience

- Novel hardware
- Novel uses of production infrastructure
- Software and services
- Tools