Supercomputing 102: The Toolbox of a Successful Computational Scientist

20 Years of Excellence in Computational Science

OLCF

Oak Ridge Leadership Computing Facility

1992–2012

Presented by:

Judith Hill

Task Lead for Scientific Computing Liaisons

Oak Ridge Leadership Computing Facility (OLCF)

National Center for Computational Sciences (NCCS)

CSGF Program Review: HPC Workshop

July 17, 2014

Washington, DC

A few background questions …

- Who is a "new" fellow?

  1st year fellow?

  2nd year fellow?

- Who has heard of MPI?

  programmed in MPI?

  OpenMP (or other threading mechanisms)?

  CUDA (or other accelerator) programming language?

The ground rules for our discussion …

- We won’t embark on “religious” discussions
  - vi/vim vs emacs
  - GPUs vs many-core vs <insert favorite architecture here>
  - Fortran vs C vs C++ OR CUDA vs OpenACC vs OpenMP

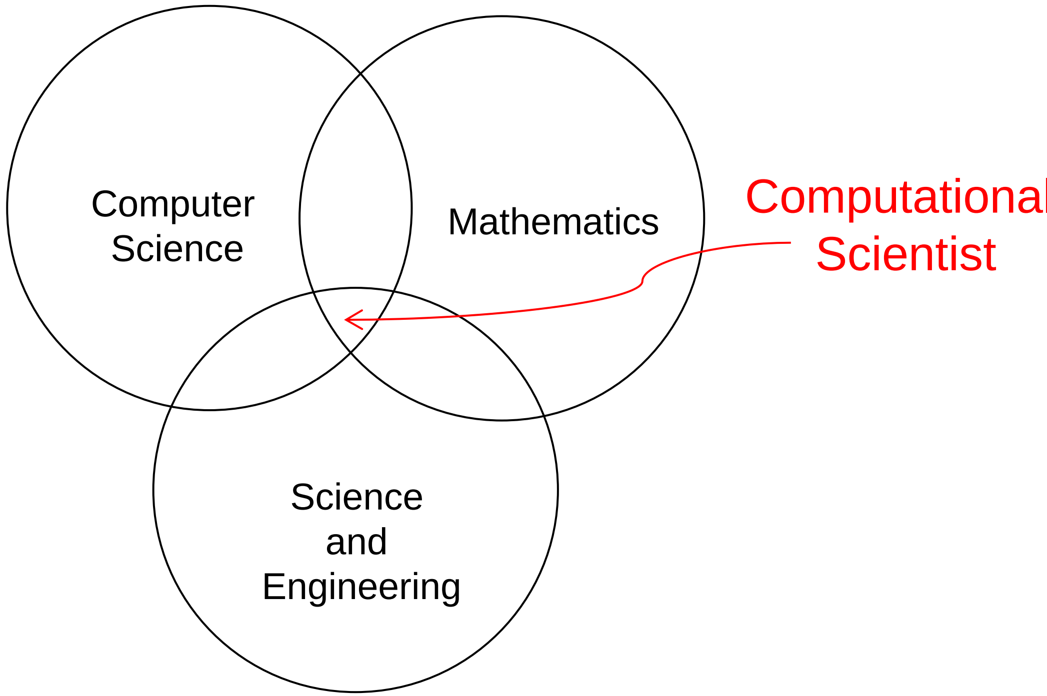
- Questions, interruptions, discussions are highly encouraged

- An hour isn’t long enough to cover all the possible topics that we could discuss
  - The following is just a starting point in hopes of sparking thought on broader topics related to computational science

A few assumptions we should make …

- I don't claim to be successful
  - But I've made enough mistakes that hopefully you can avoid a few of them
- This is not a talk I've given before (nor is it overly technical)
  - Feedback on topics you would have liked or were expecting to see would be appreciated
- My background is in numerical methods and algorithms applied to continuum PDEs
  - Our topics will be general and high level, but any examples I discuss will likely be from this viewpoint

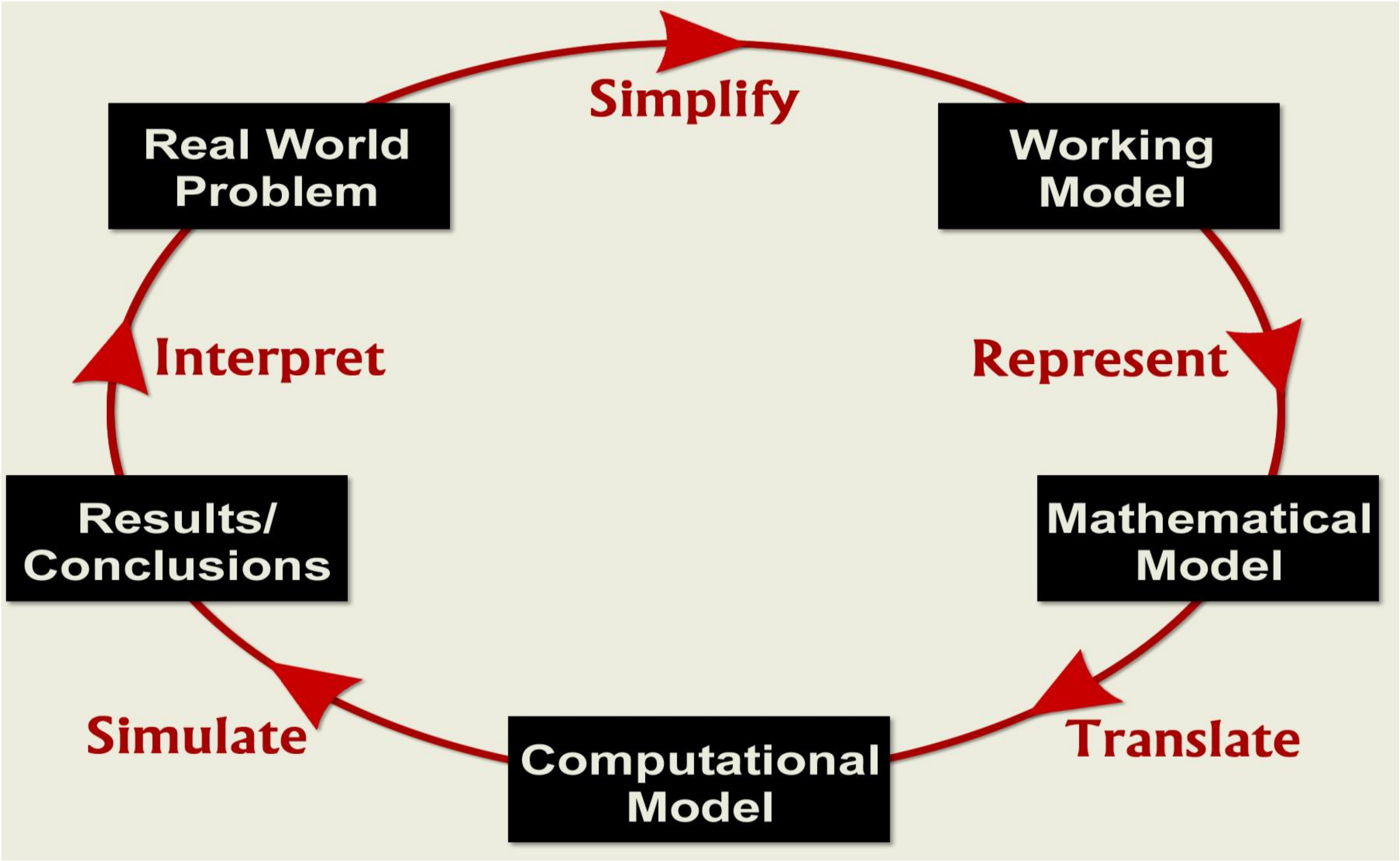
Our Goal: Define a successful computational scientist



Computational science is neither computer science, mathematics, some traditional field of science, engineering, a social science, nor a humanities' field.

It is a blend.

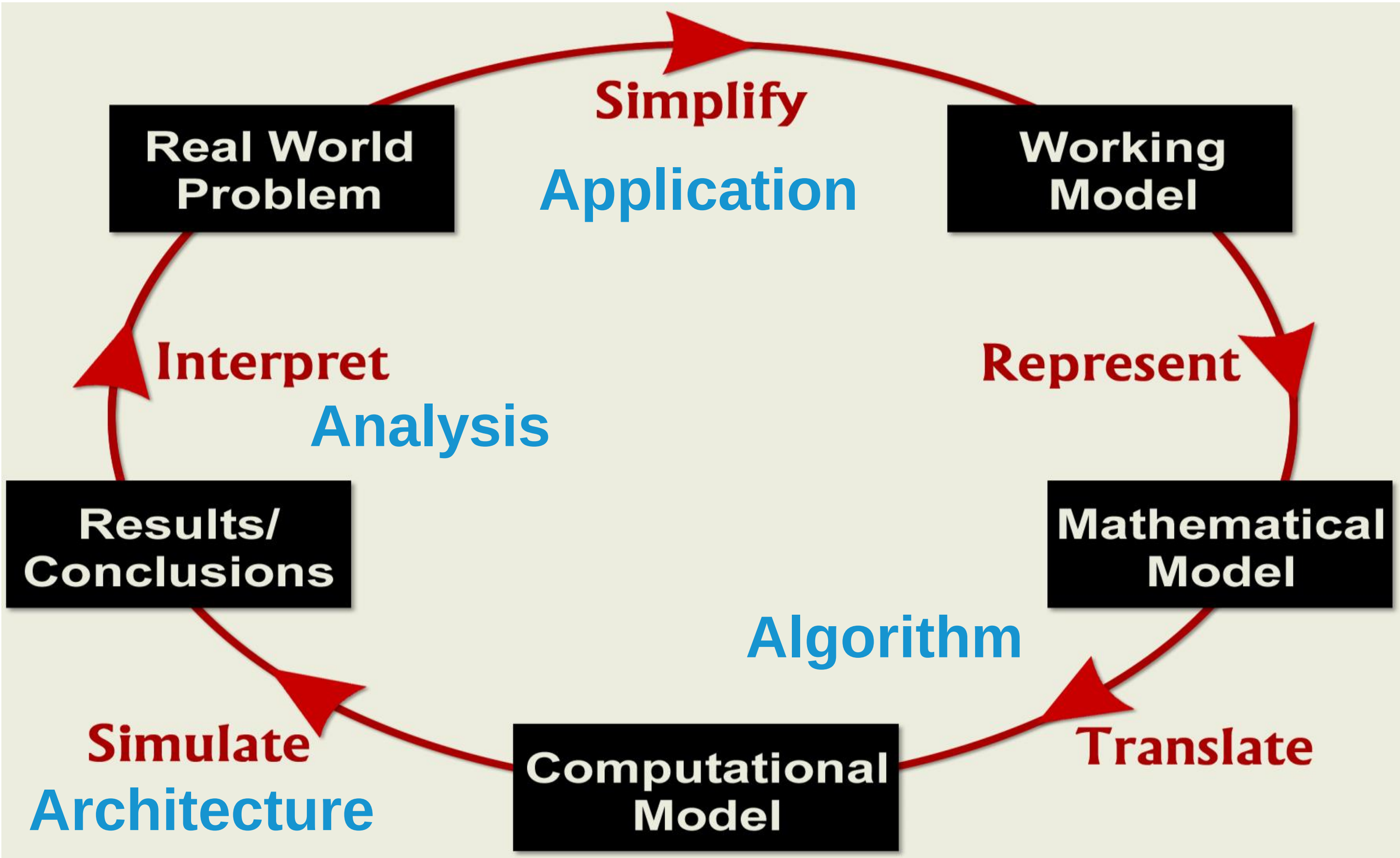
The Computational Science Process



The Three A's that are required ...

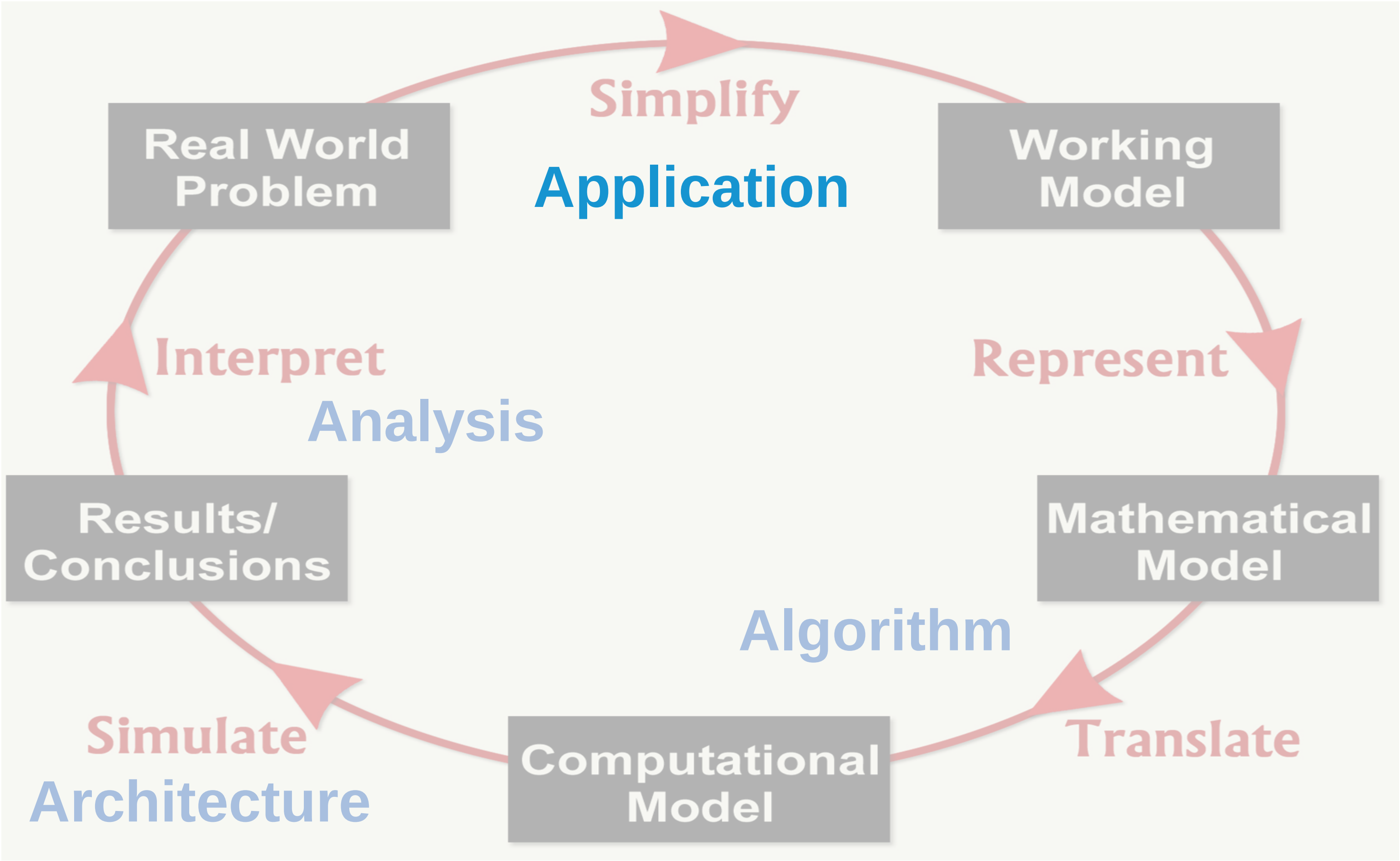
- **A computational science investigation should include**
  - **An Application -** a scientific problem of interest and the components of that problem that we wish to study and/or include.
  - **Algorithm -** the numerical/mathematical representation of that problem, including any numerical methods or recipes used to solve the algorithm.
  - ✓ **Architecture –** a computing platform and software tool(s) used to compute a solution set for the algorithm.
- A fourth "**A**" is increasingly becoming important: **Analysis** of the simulation results and comparison with experimental measurements or observations.

The Computational Science Process



A successful computational scientist will address this entire "life cycle"

The Computational Science Process



A successful computational scientist will address this entire "life cycle"

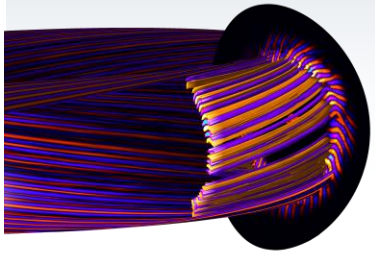
Why HPC? = Science drivers

"Computational simulation offers to enhance, as well as leapfrog, theoretical and experimental progress in many areas of science and engineering…"

A Science-Based Case for Large-Scale Simulation (SCaLeS Report), Office of Science, U.S. DOE, July 2003

Advanced energy systems

- Fuel cells
- Fusion



Biotechnology

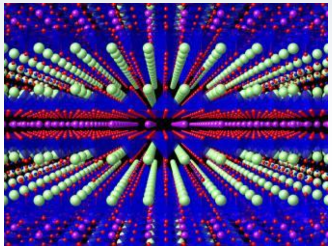
- Genomics
- Cellular dynamics

Environmental modeling

- Climate prediction
- Pollution remediation

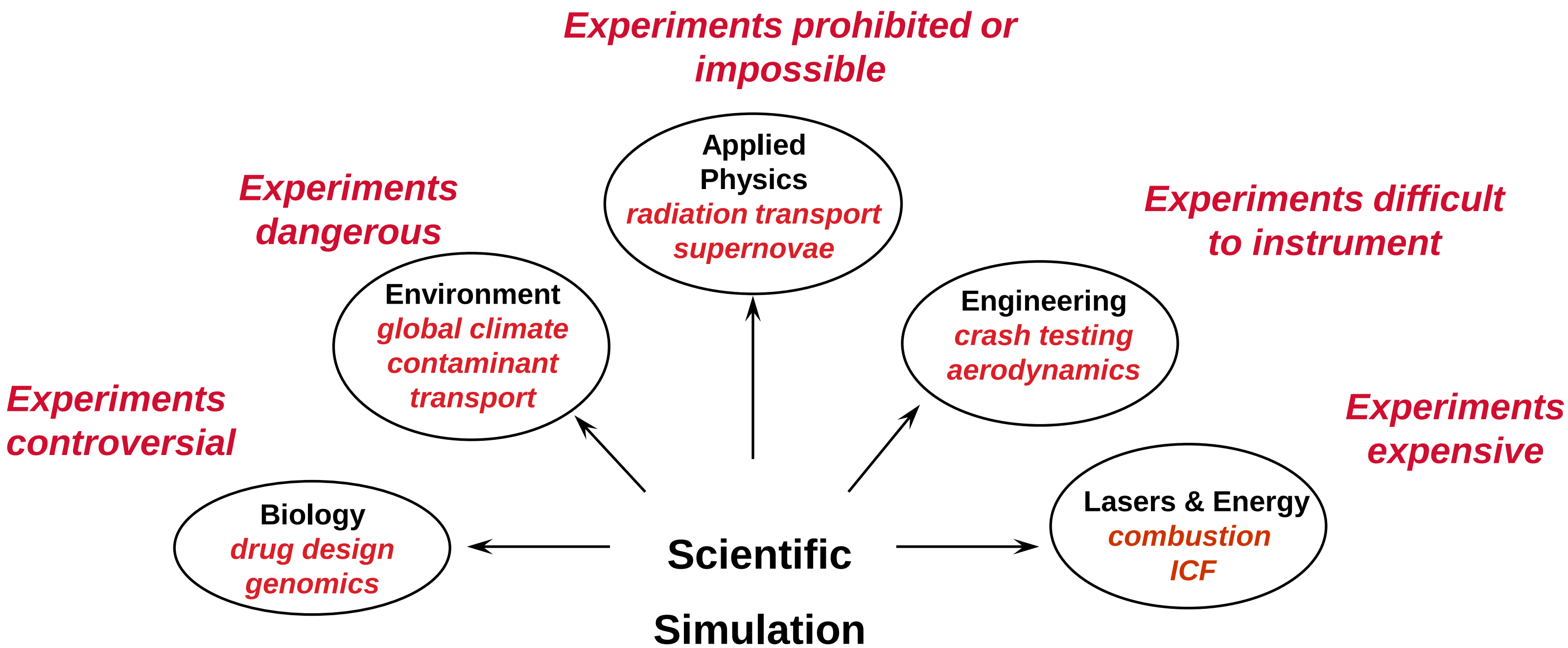
Nanotechnology

- Sensors
- Storage devices



Slide from D. Keyes

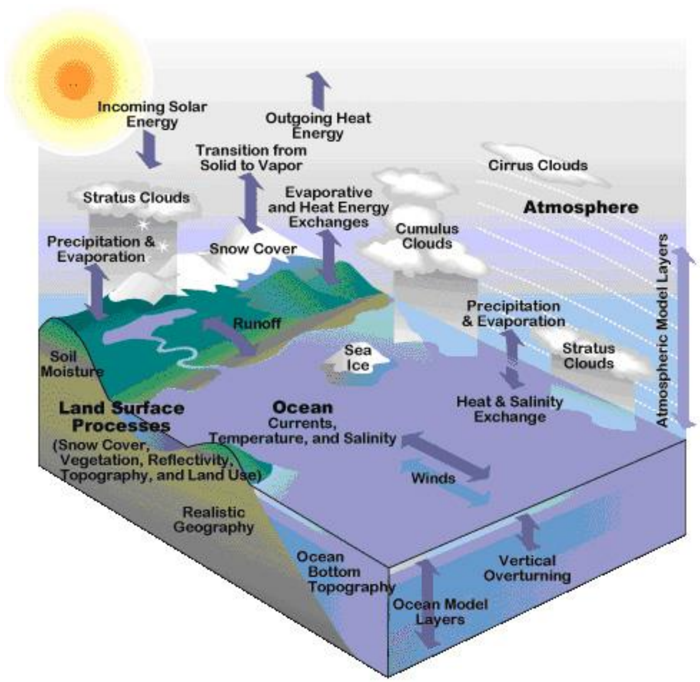
The imperative of simulation



In these, and many other areas, simulation is an important complement to experiment.

Challenges of Application Representation

- Too much simplification can lead to an unrealistic model
  - Trusting the results obtained with the model will be difficult



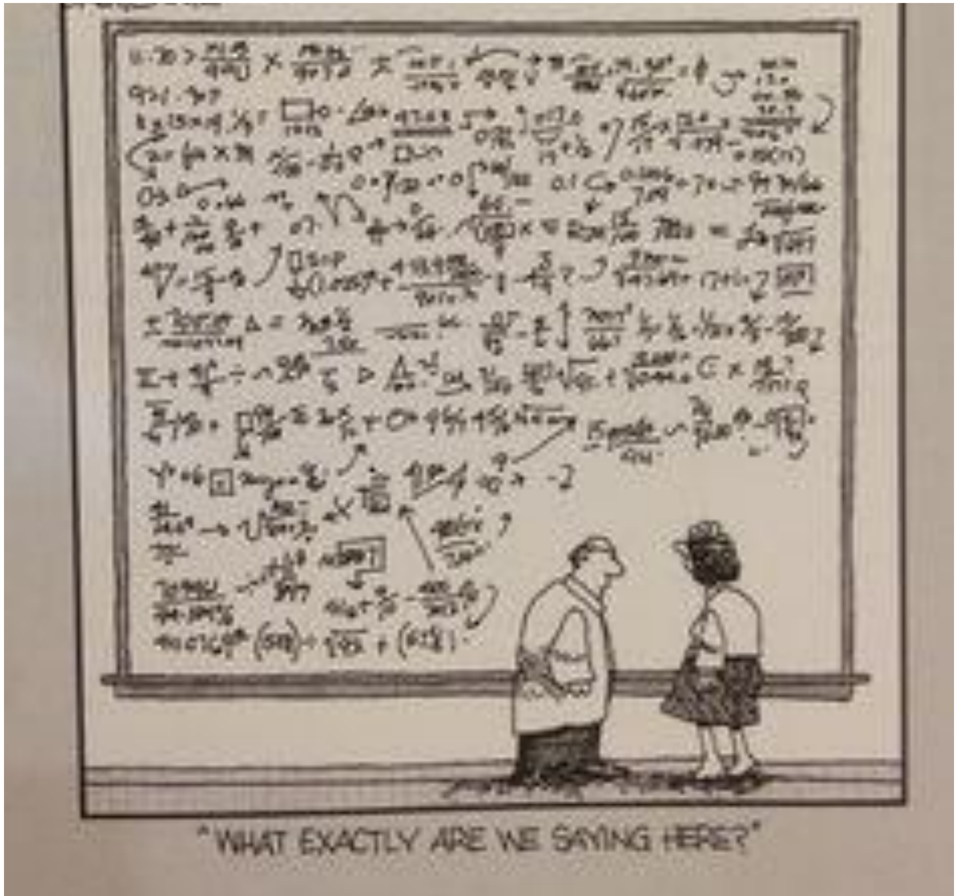
Physical/Chemical processes in climate

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

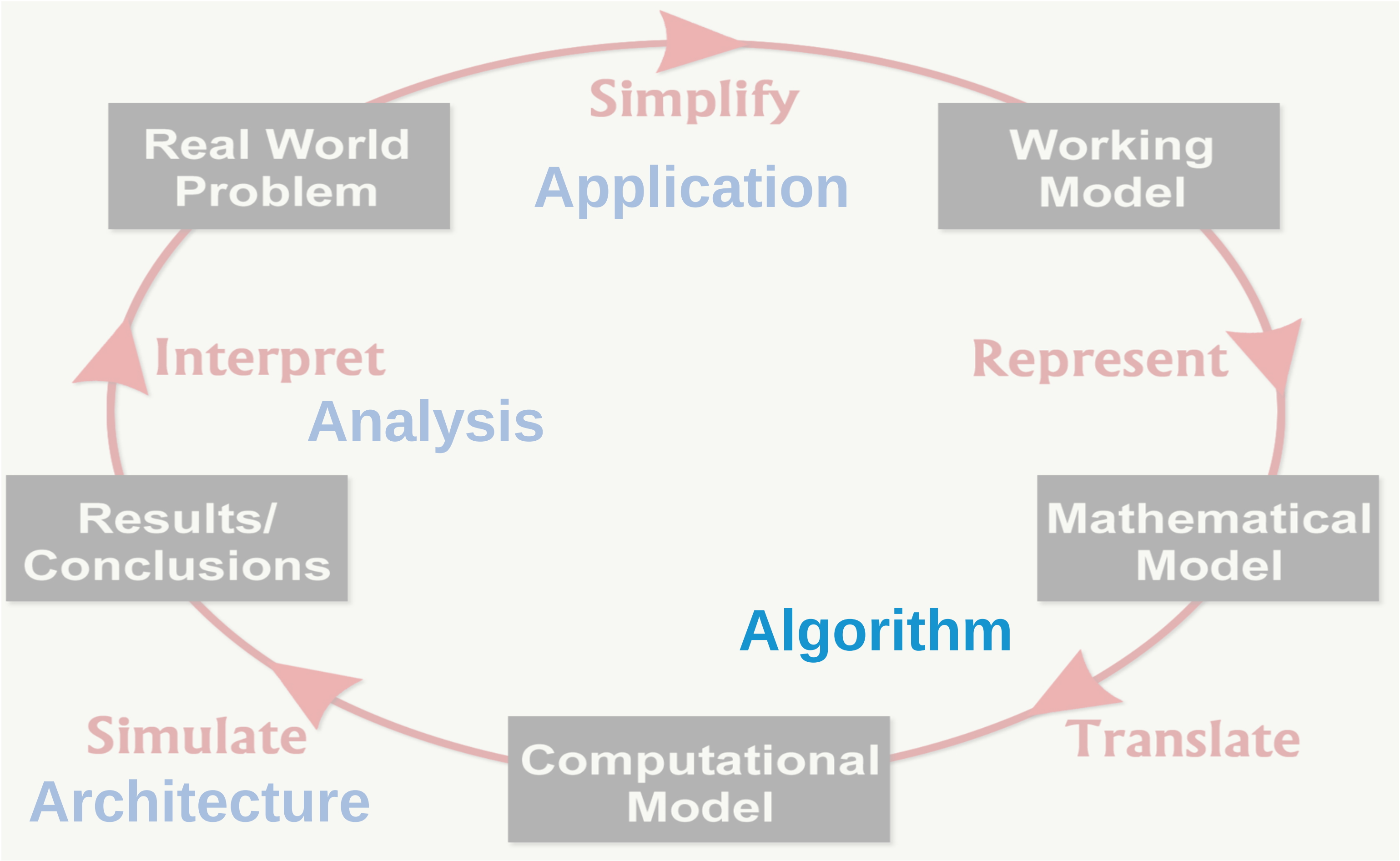
Mathematicians favorite equation

Challenges of Application Representation

- Too much detail leads to results that may be problematic to implement and debug, difficult to solve and impossible to analyze/interpret



The Computational Science Process



Algorithmic Challenges

- We've chosen a problem to study and created a mathematical model for it

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

How do we actually approach solving this equation?

Common Computational Motifs

- High-end simulation in the physical sciences can generally be represented by a few computational patterns
  - Dense Linear Algebra
  - Sparse Linear Algebra
  - Fast Fourier Transform
  - Particles
  - Monte Carlo approaches
  - Structured Grids/Meshes
  - Unstructured Grids/Meshes

Caution: This list is not all-encompassing. There have been modifications and additions proposed to cover a broader range of applications.

“Defining Software Requirements for Scientific Computing”, Phillip Colella, 2004

Common Computational Motifs

- High-end simulation in the physical sciences can generally be represented by a few computational patterns
  - Dense Linear Algebra
  - Sparse Linear Algebra
  - Fast Fourier Transform
  - Particles
  - Monte Carlo approaches
  - Structured Grids/Meshes
  - Unstructured Grids/Meshes

✓ **These motifs have a pattern of computation and communication shared amongst applications**

✓ **These motifs are well-defined targets from algorithmic, software, and architecture standpoints**

“Defining Software Requirements for Scientific Computing”, Phillip Colella, 2004

Why are motifs important to you?

- Gives you a vocabulary/organization to talk across disciplinary boundaries
- Define building blocks for creating libraries that cut across application domains

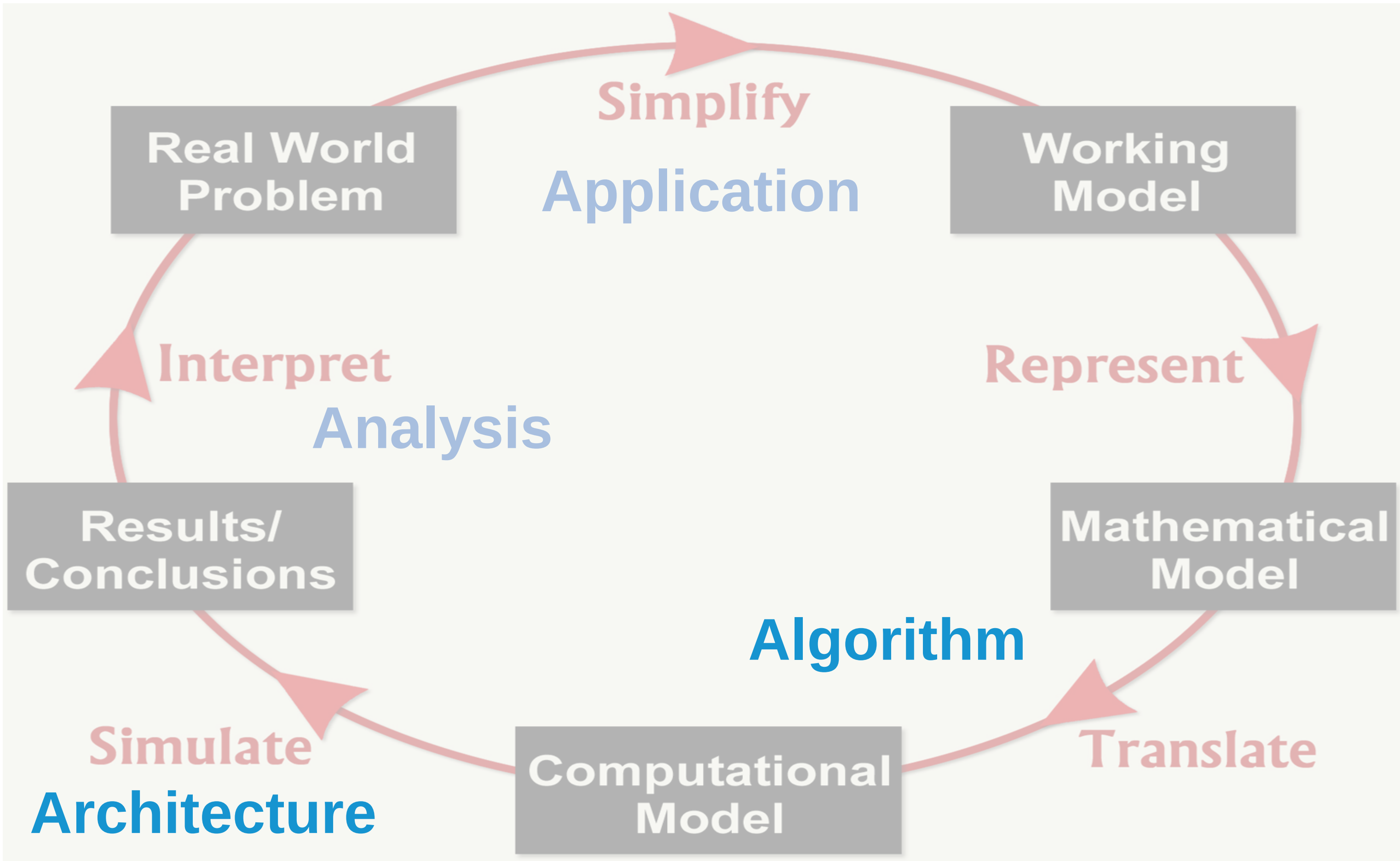
Dense Linear Algebra	Sparse Linear Algebra	FFT	Particles	Monte Carlo	Structured Grids	Unstructured Grids
ScaLAPACK SuperLU	PETSc Trilinos	FFTW			Overture Chombo	Cubit

List of software libraries is not complete nor all encompassing.

Why are motifs important to me?

- Define minimum set of necessary functionality for new hardware/software systems and help to ensure algorithm coverage for testing/acceptance
- “Anti-benchmarks” not tied to code or language artifacts ⇒ encourage innovation in algorithms, languages, data structures, and/or hardware
- They decouple research in computer science and mathematics without waiting years for full application implementation/development

That’s all I need to know about algorithms?



The implementation of a numerical method on an architecture

Programming Models

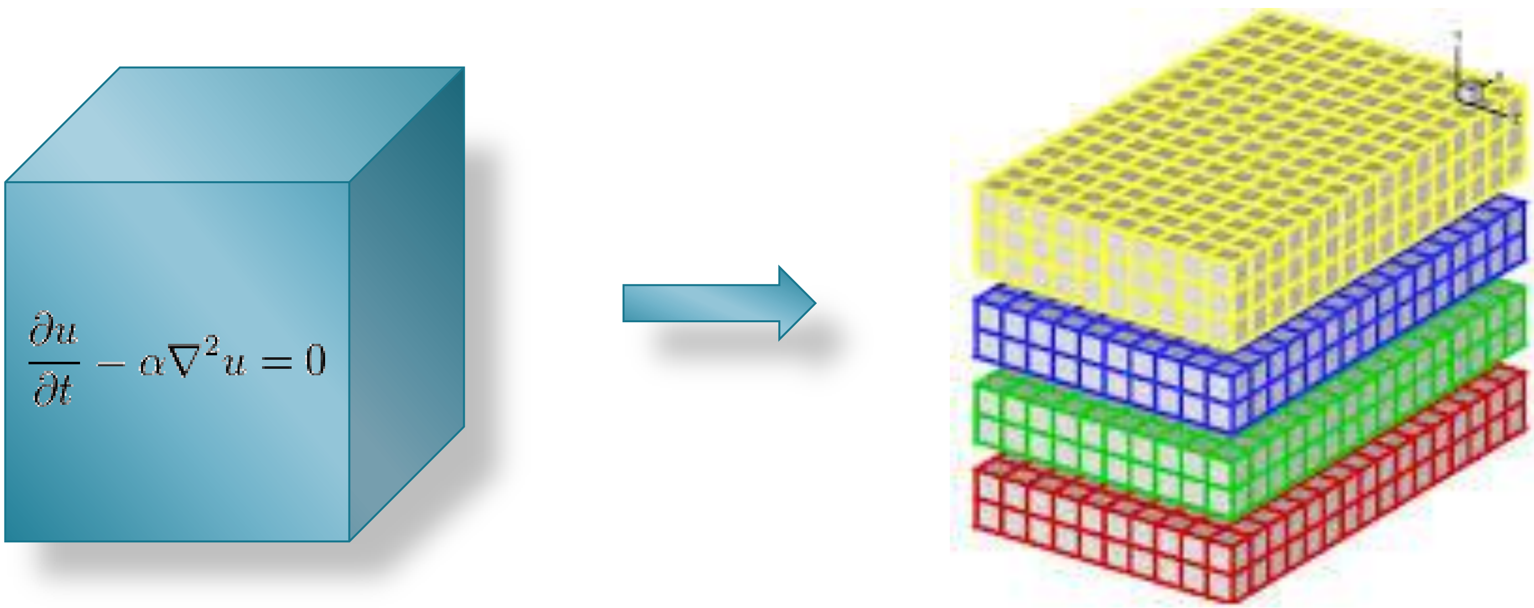
Matlab/Python
MPI
MPI + X
Accelerator Programming

Increasing Programming Complexity

Requires Increase in Exposed Parallelism

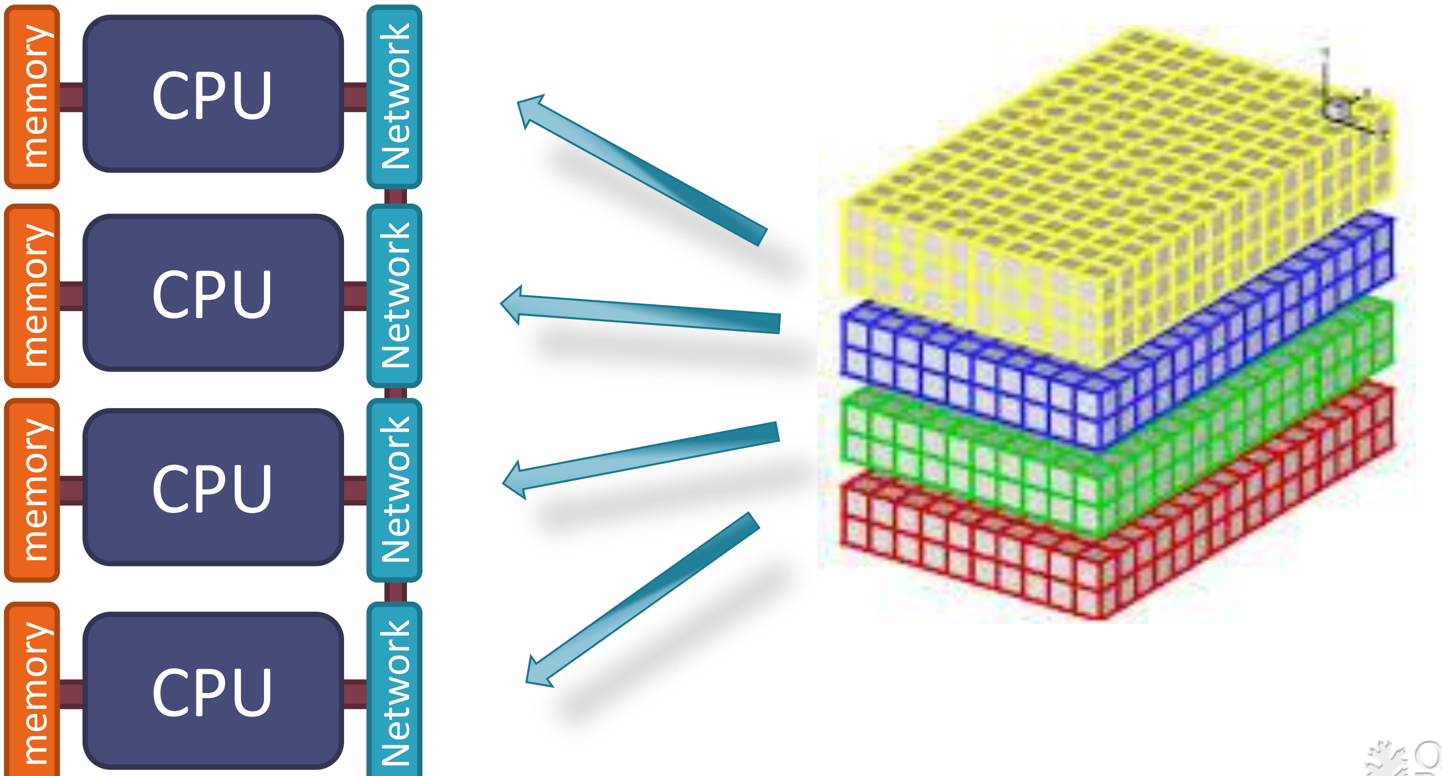
Distributed Programming with MPI in three slides

- Strategy: Domain Decomposition
  - Generally used for high(er)-fidelity simulations where the time-to-solution is too long on one processor or the resolution is not sufficient



Distributed Programming with MPI in three slides

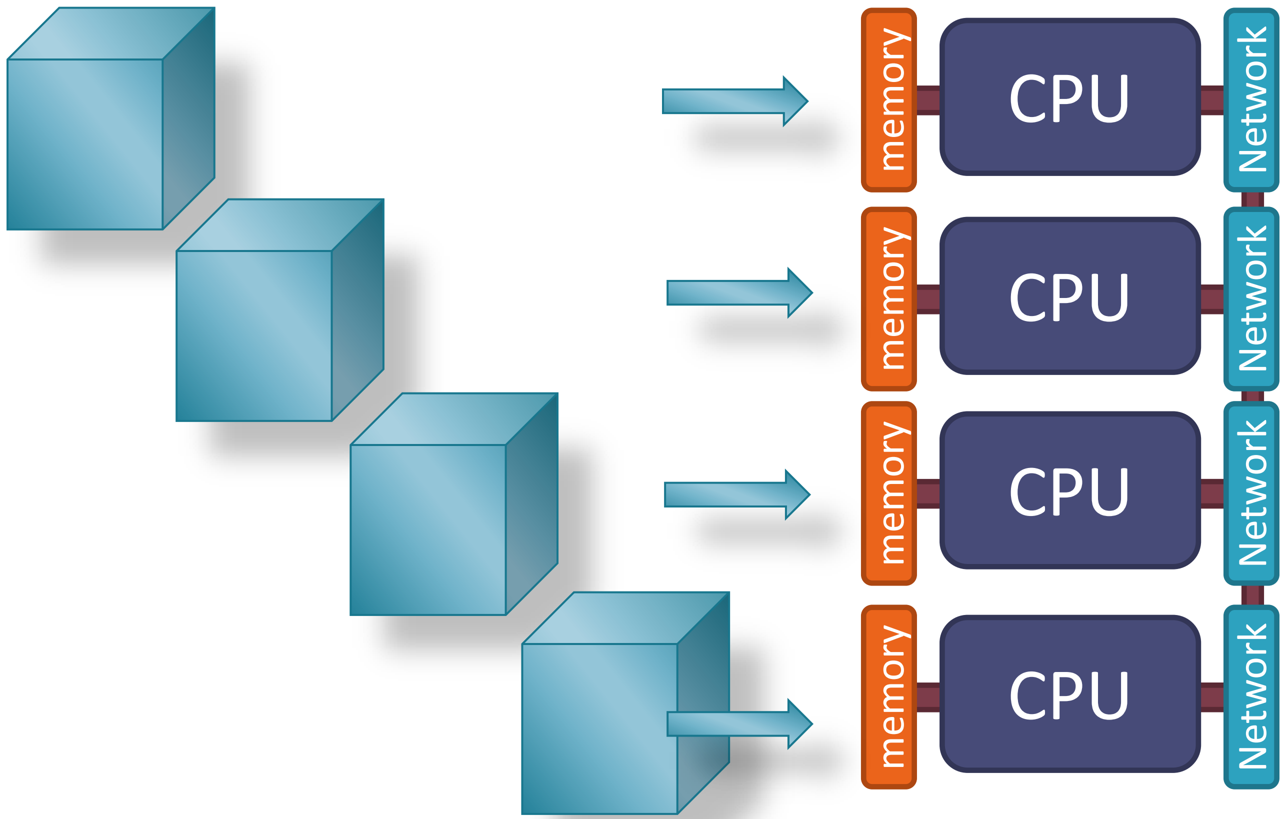
- Strategy: Domain Decomposition
  - Pass data between processes running on different nodes



Distributed Programming with MPI in three slides

- Strategy: "Naively" parallel simulations
  - Generally used for parameter sweeps or an ensemble of simulations that are very similar

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$



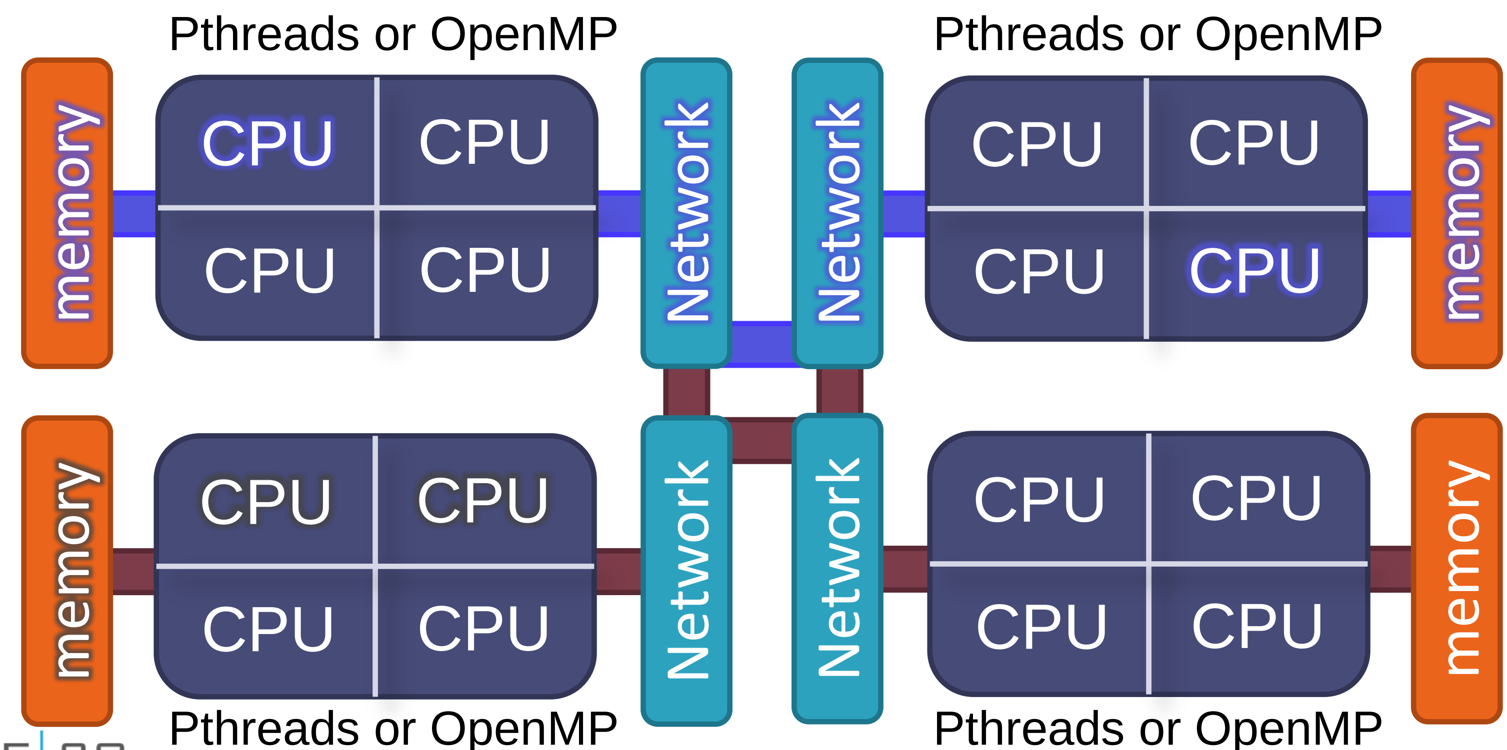
Distributed Programming with MPI in three slides

- Challenges you may encounter
  – As the number of MPI "ranks" (processes) grows, communication across the network can become contentious
    - Design algorithms to avoid "all-to-all" communication patterns if possible

  – As your ensembles come quite large, managing your "**workflow**" can be difficult

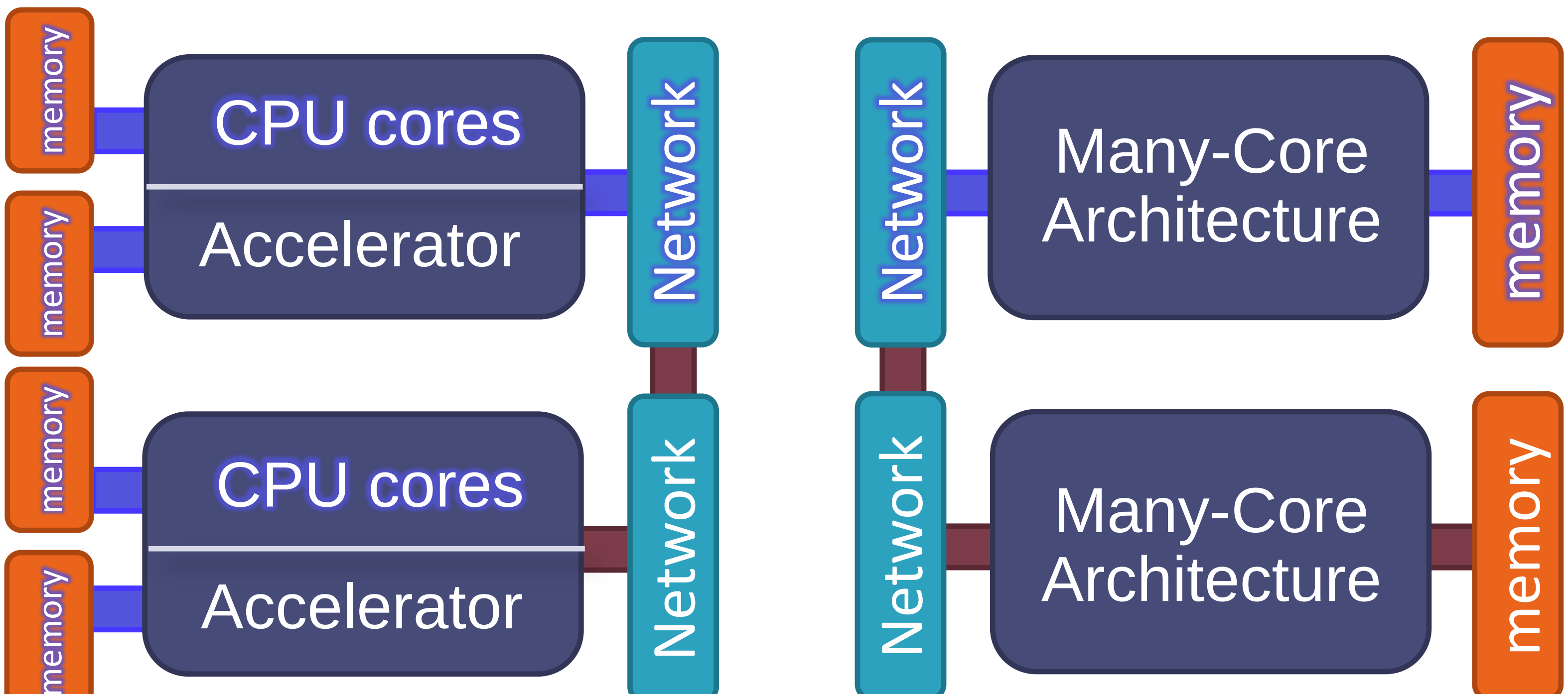
Shared Memory Programming in one slide

- Modern hybrid shared/distributed memory systems using multi-core processors
  - Inter/intra node communication
  - Each thread handles a subset of the calculations



Accelerator Programming in one slide

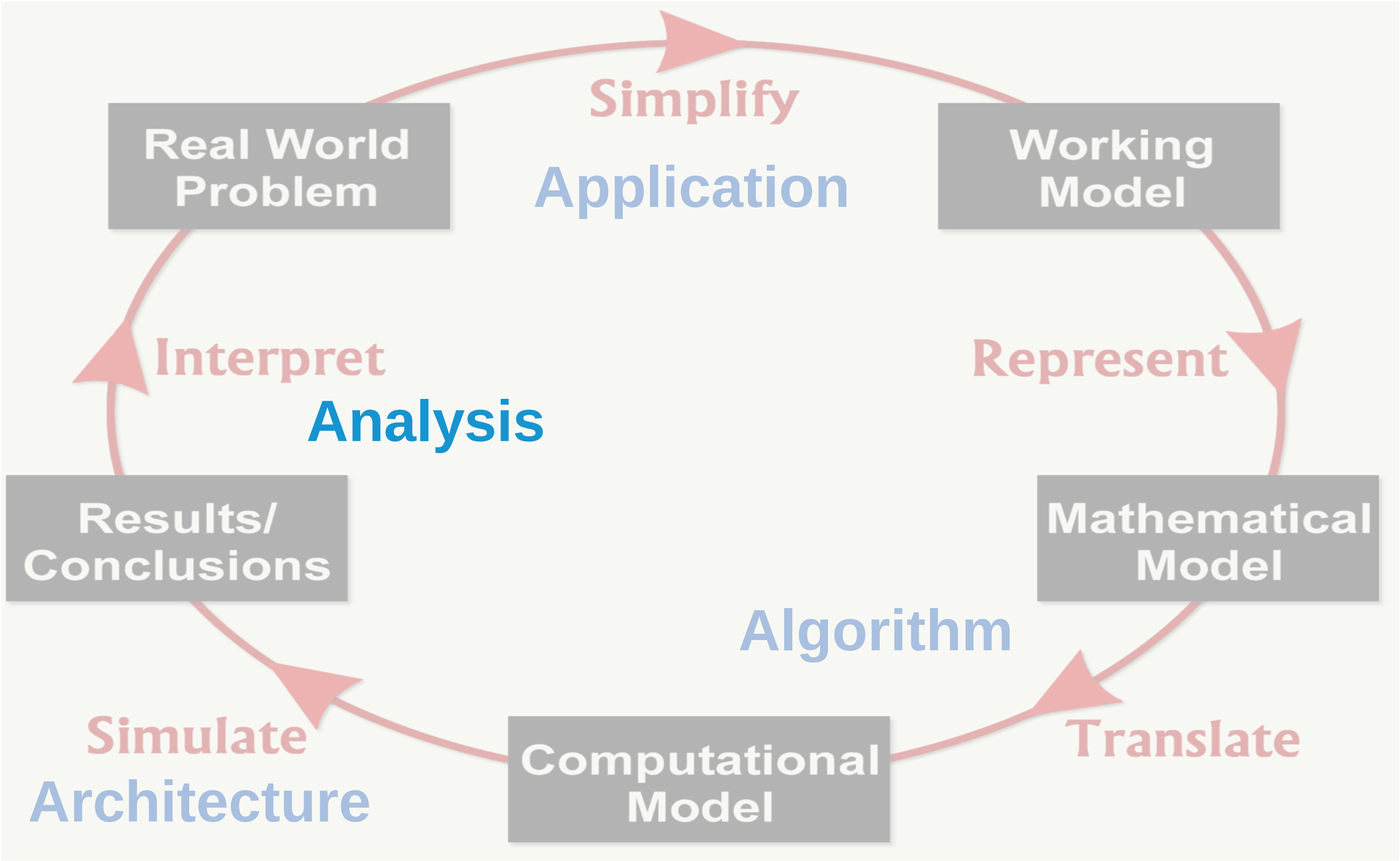
- Modern heterogeneous systems are using accelerators
  - The cost of data movement becomes the bottleneck
  - Keeping the accelerator "fed" to take advantage of the large number of flops available is challenging



OLCF's "Titan" system

Nersc's future "Cori" system

The Computational Science Process



Analysis can equal Big Data

That's a whole 'nother hour …

Our Goal: Define a successful computational scientist

- We've talked about the life cycle of computational science and abstract "tools" at each stage, but

How do we measure success?

The largest (in core count) simulation in the world?

The highest-resolution?

The most efficient (in terms of flops)?

Advantage: Very quantifiable

The most impactful scientific result → **Admittedly nebulous**

What I'm Thinking About These Days …
and hopefully you will too

- Changing architectural landscape
  - (Software) Application Portability
    - The disruptive transition with the advent of accelerators and many-core chips and resulting challenges for application developers to use all available supercomputers
  - (Software) Application Readiness
    - Ensuring that applications are prepared to take advantage of coming architectures
    - Ensuring that tools are available to application developers (compilers, debuggers, profilers, etc)

What I'm Thinking About These Days …

and hopefully you will too

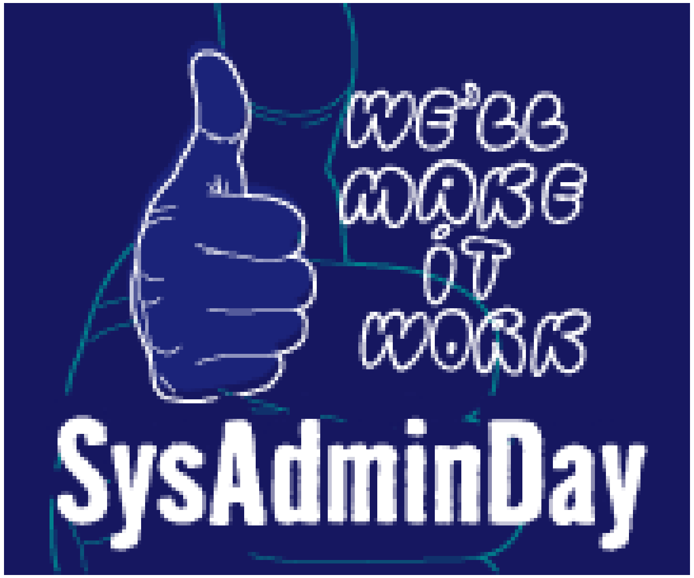
- Algorithmic innovations needed for emerging architectures
  - Fault tolerance and resilience
    - Increasingly larger computers generally means the "Mean Time to Failure" of hardware decreases.
    - Application developers have to be ready for this and guard against it themselves for the foreseeable future
- "Big Data"
  - Challenges for applications to appropriately (and efficiently) read/write large amounts of data
  - Computing needs for analytics of simulation data and how that differs from our traditional "Big Compute" approach
  - Integration of (large) experimental data into our simulation frameworks

Take Home Thoughts

- ✓ Identify an appropriate mathematical model that accurately represents the physical phenomena without being overly complicated
- ✓ Leverage existing software libraries for common computational kernels
- ✓ Exposing parallelism in your algorithms and implementation is continuing to grow in importance
- ✓ Future application development challenges due to architecture evolution and growth of simulation and experimental data

Final Advice and an Advertisement

- Computational science is not a field that is practiced alone. It generally requires teamwork and recognition of others' expert knowledge and skills.



Friday July 25, 2014

Thanks to Heather Mayes for reminding me every year!

Recommended Reading

- Krste Asanovíc, Rastislav Bodik, Bryan Catanzaro, Joseph Gebis, Parry Husbands, Kurt Keutzer, David Patterson, William Plishker, John Shalf, Samuel Williams, and Katherine Yelick. The Landscape of Parallel Computing Research: A View from Berkeley. Electrical Engineering and Computer Sciences University of California at Berkeley. Technical Report No. UCB/EECS-2006-18
  – http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

- Prakash Prabhu, Thomas B. Jablin, Arun Raman, Yun Zhang, Jialu Huang, Hanjun Kim, Nick P. Johnson, Feng Liu, Soumyadeep Ghosh, Stephen Beard, Taewook Oh, Matthew Zoufaly, David Walker, and David I. August. **A Survey of the Practice of Computational Science.** *Proceedings of the 24th ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, November 2011.

Questions?