NNSA
National Nuclear Security Administration

UNIVERSITY OF
NOTRE DAME

ASC™

High Performance Computing for Science and Engineering

Karel Matouš

College of Engineering Collegiate Associate Professor of Computational Mechanics
Director of Center for Shock-Wave Processing of Advanced Reactive Materials

Outline

- Why Parallel Computing
  - Importance of parallel computing
  - Examples of parallel computing
- HPC architecture trends
  - Parallel architectures and their trends
  - Path towards Exascale
- On node, and node to node parallelism
  - Modern parallel programming languages, OpenMP, MPI, and HPX
  - Co-processor acceleration, CUDA and OpenCL
  - Core kernels packages, Kokkos

Outline

- Numerical Parallel Algorithms
  - Data parallelism
  - Scaleability
  - PDEs, Optimization tools, Statistical techniques, Multiscale modeling
  - Numerical Libraries, ScaLAPACK, PETSc, LAMMPS, Trilinos, HYPRE, ParMETIS
- Software Engineering Tools
  - Gitlab, Jenkins, Debugging, Profiling, Visualization

Solving Real World Problems — Beyond Academic Exercise

This Week

NEWS

PAGE 1

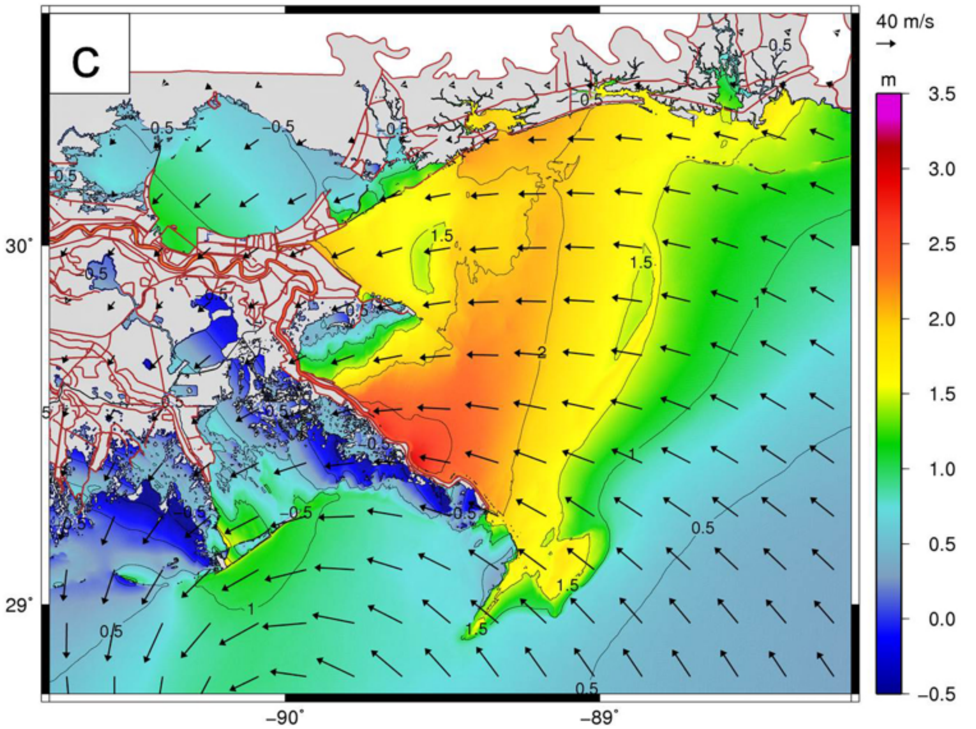
Japan's n
generatio
facilities

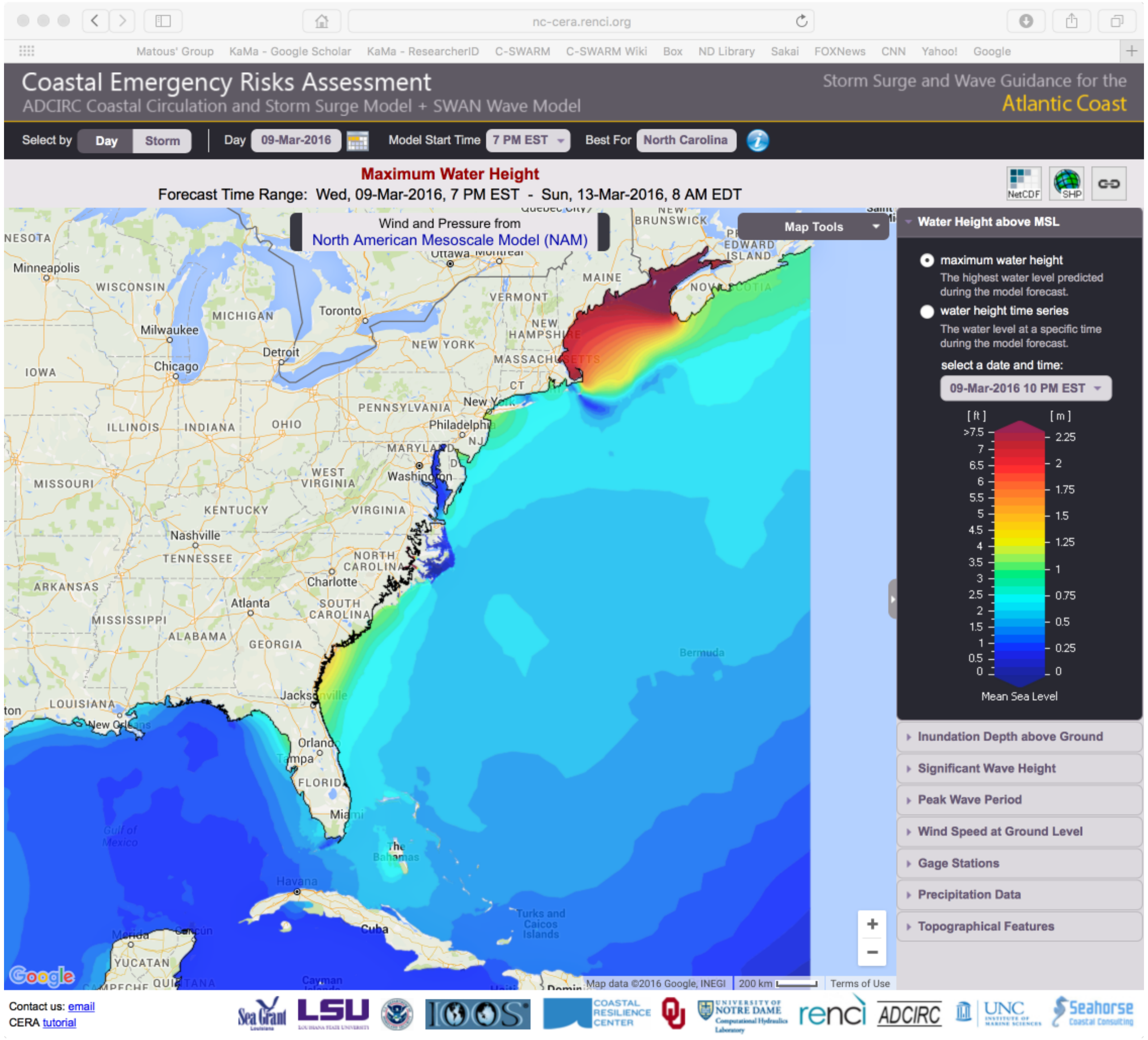
HURRICANE KATRINA

Scientists' Fears Come True as Hurricane Floods New Orleans

9 SEPTEMBER 2005 VOL 309 SCIENCE www.sciencemag.org

Published by AAAS





nc-cera.renci.org

Matous' Group KaMa - Google Scholar KaMa - ResearcherID C-SWARM C-SWARM Wiki Box ND Library Sakai FOXNews CNN Yahoo! Google

Coastal Emergency Risks Assessment

ADCIRC Coastal Circulation and Storm Surge Model + SWAN Wave Model

Storm Surge and Wave Guidance for the Atlantic Coast

Select by Day Storm | Day 09-Mar-2016 | Model Start Time 7 PM EST | Best For North Carolina

Maximum Water Height

Forecast Time Range: Wed, 09-Mar-2016, 7 PM EST - Sun, 13-Mar-2016, 8 AM EDT

Wind and Pressure from North American Mesoscale Model (NAM)

Water Height above MSL

● maximum water height
The highest water level predicted during the model forecast.

○ water height time series
The water level at a specific time during the model forecast.

select a date and time: 09-Mar-2016 10 PM EST

- Inundation Depth above Ground
- Significant Wave Height
- Peak Wave Period
- Wind Speed at Ground Level
- Gage Stations
- Precipitation Data
- Topographical Features

Contact us: email
CERA tutorial

▸ Computational Hydraulics Laboratory

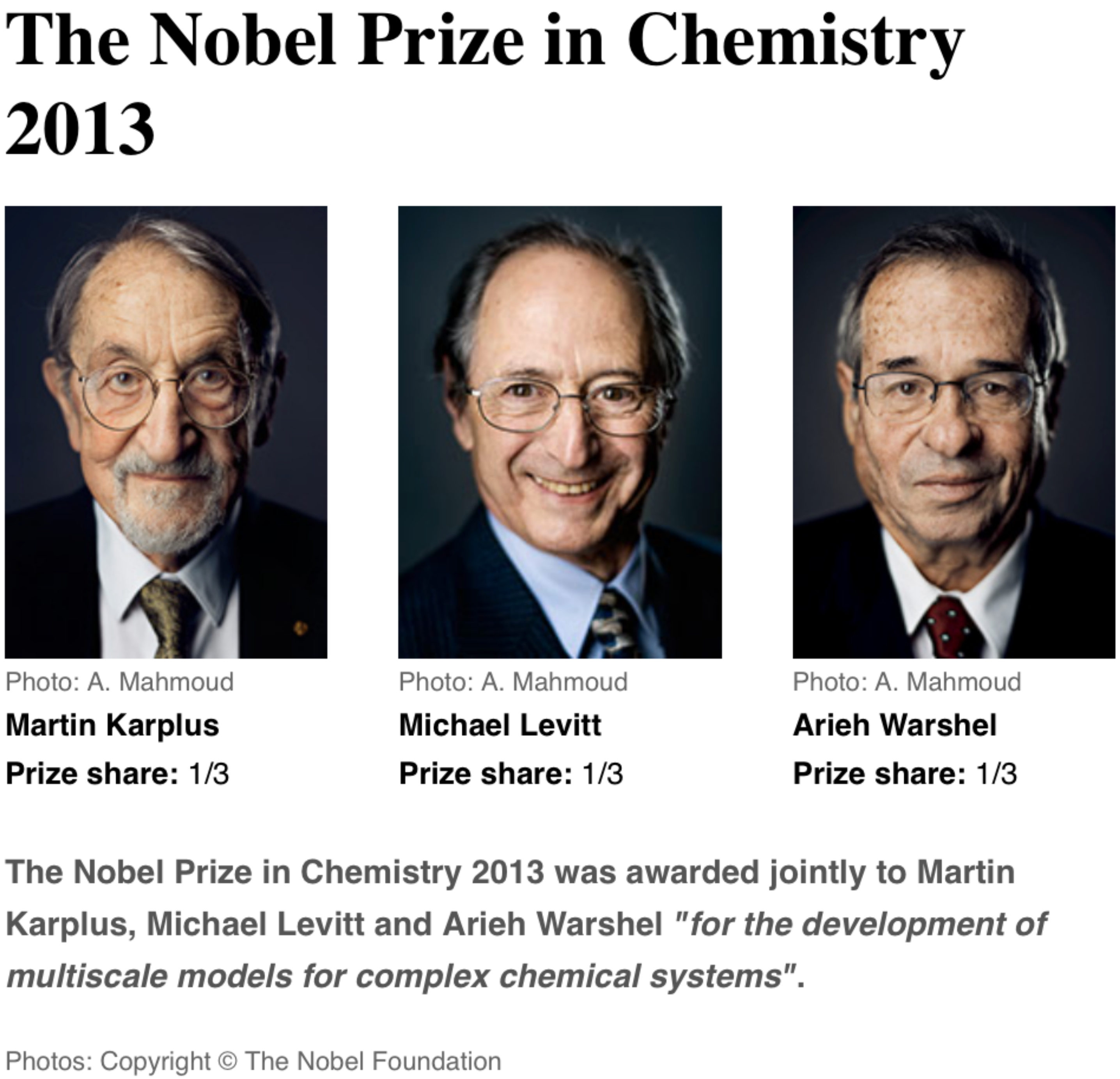
http://www.nd.edu/~coast/index.html

http://nc-cera.renci.org

UNIVERSITY OF NOTRE DAME

Solving Real World Problems — Beyond Academic Exercise

- (Chemistry at HARvard Macromolecular Mechanics)



The Nobel Prize in Chemistry 2013

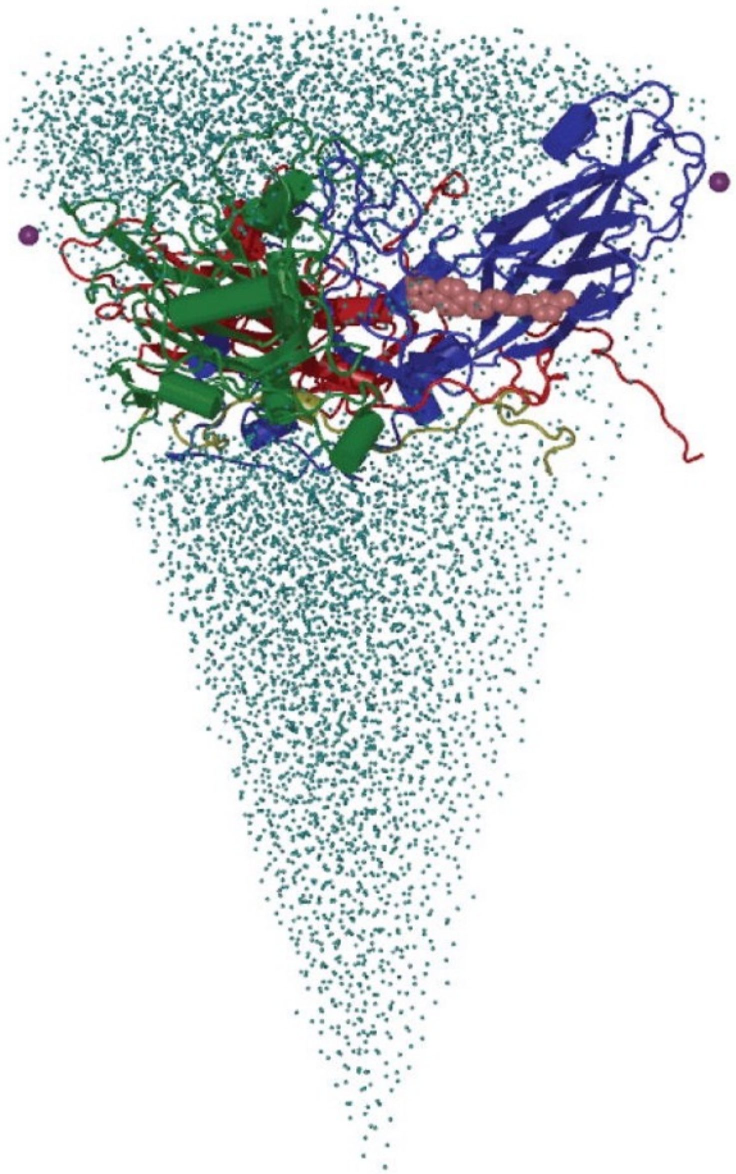
Photo: A. Mahmoud
Martin Karplus
Prize share: 1/3

Photo: A. Mahmoud
Michael Levitt
Prize share: 1/3

Photo: A. Mahmoud
Arieh Warshel
Prize share: 1/3

The Nobel Prize in Chemistry 2013 was awarded jointly to Martin Karplus, Michael Levitt and Arieh Warshel *"for the development of multiscale models for complex chemical systems"*.

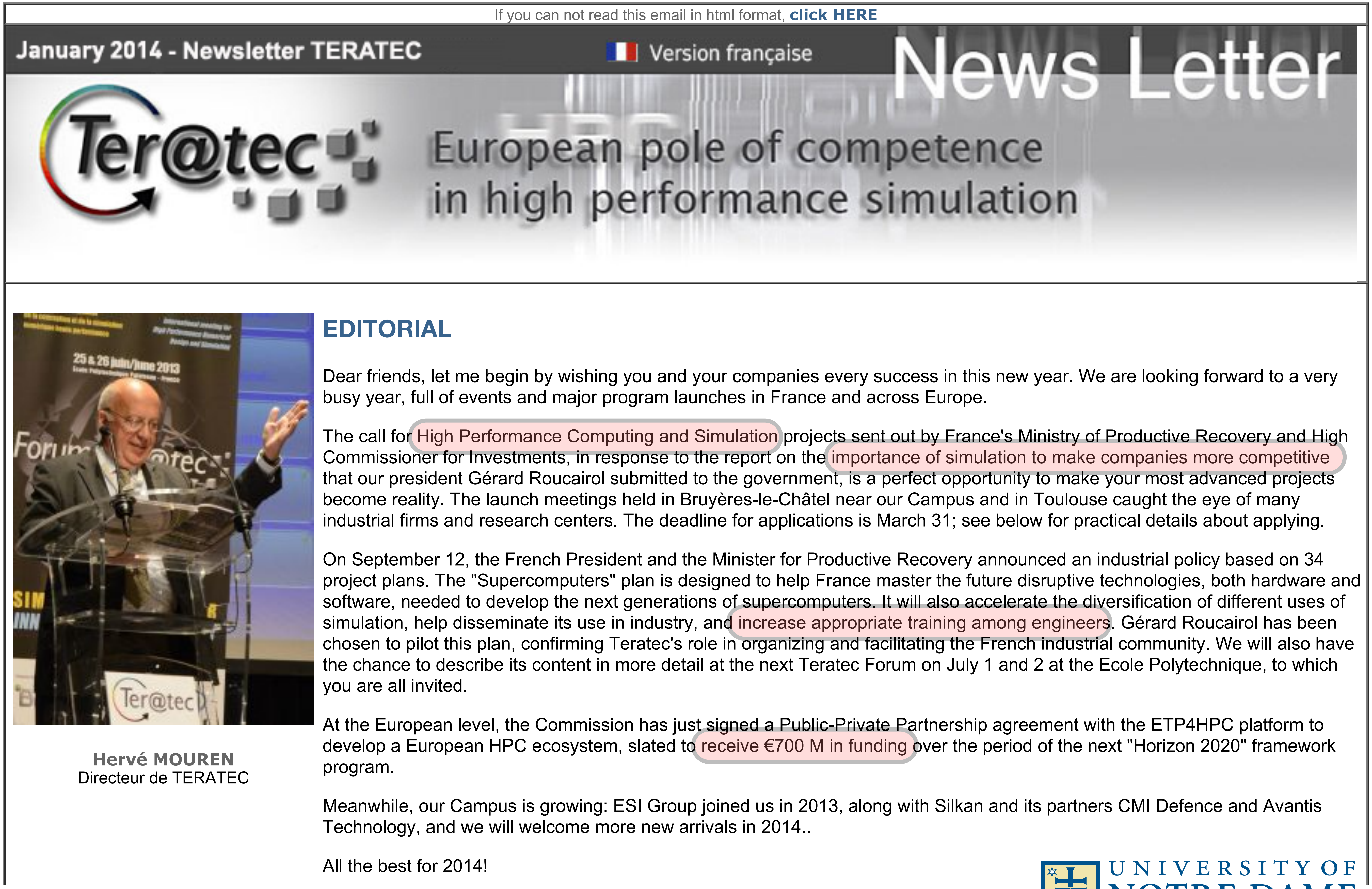
Photos: Copyright © The Nobel Foundation



The protomeric unit of HRV14 virus
CHARMM mode using 750,000 atoms

KADAU et al., MOLECULAR DYNAMICS COMES OF AGE:
320 BILLION ATOM SIMULATION ON BlueGene/L, 2006

High Performance Computing



If you can not read this email in html format, **click HERE**

January 2014 - Newsletter TERATEC

Version française

News Letter

European pole of competence
in high performance simulation

EDITORIAL

Dear friends, let me begin by wishing you and your companies every success in this new year. We are looking forward to a very busy year, full of events and major program launches in France and across Europe.

The call for High Performance Computing and Simulation projects sent out by France's Ministry of Productive Recovery and High Commissioner for Investments, in response to the report on the importance of simulation to make companies more competitive that our president Gérard Roucairol submitted to the government, is a perfect opportunity to make your most advanced projects become reality. The launch meetings held in Bruyères-le-Châtel near our Campus and in Toulouse caught the eye of many industrial firms and research centers. The deadline for applications is March 31; see below for practical details about applying.

On September 12, the French President and the Minister for Productive Recovery announced an industrial policy based on 34 project plans. The "Supercomputers" plan is designed to help France master the future disruptive technologies, both hardware and software, needed to develop the next generations of supercomputers. It will also accelerate the diversification of different uses of simulation, help disseminate its use in industry, and increase appropriate training among engineers. Gérard Roucairol has been chosen to pilot this plan, confirming Teratec's role in organizing and facilitating the French industrial community. We will also have the chance to describe its content in more detail at the next Teratec Forum on July 1 and 2 at the Ecole Polytechnique, to which you are all invited.

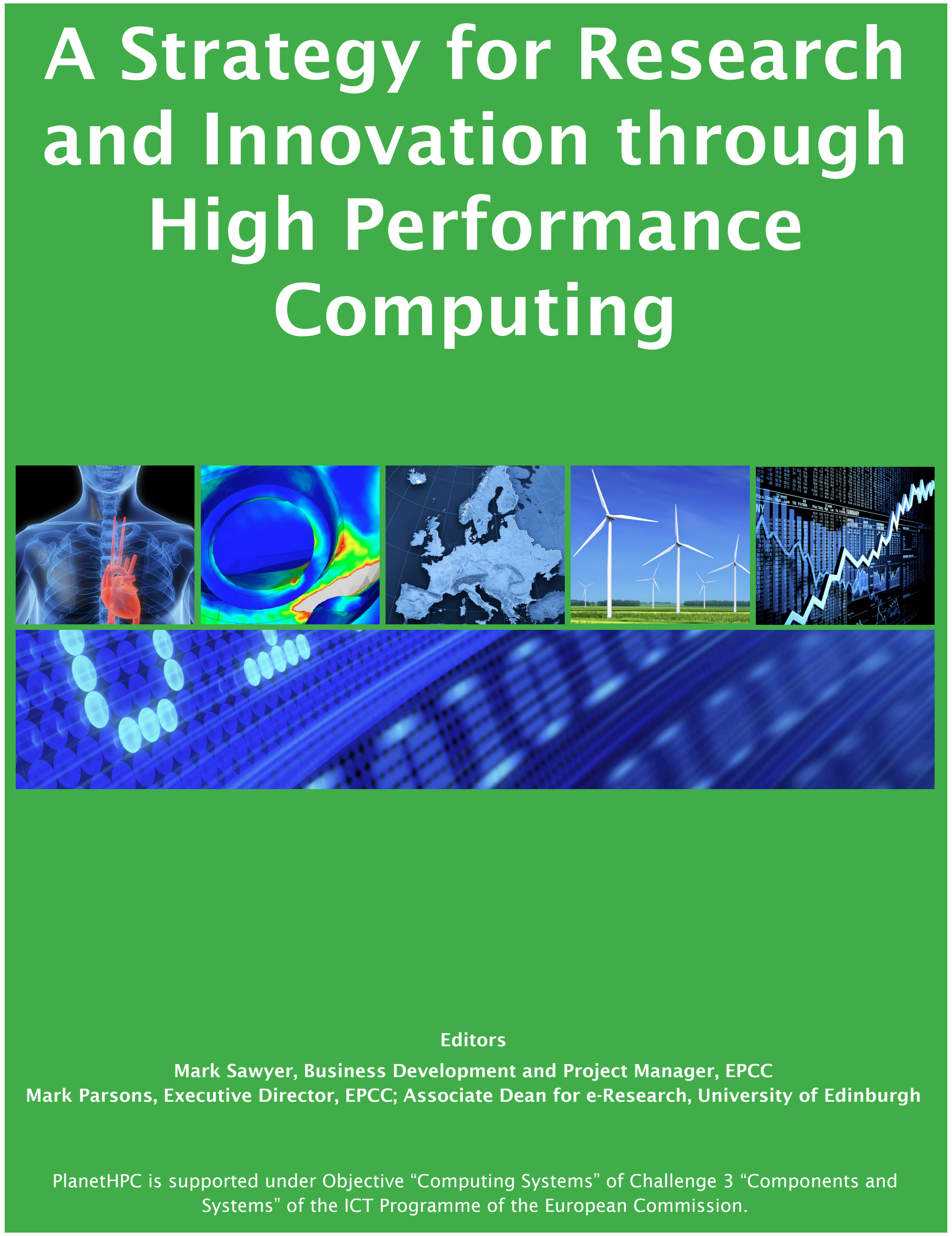
At the European level, the Commission has just signed a Public-Private Partnership agreement with the ETP4HPC platform to develop a European HPC ecosystem, slated to receive €700 M in funding over the period of the next "Horizon 2020" framework program.

Meanwhile, our Campus is growing: ESI Group joined us in 2013, along with Silkan and its partners CMI Defence and Avantis Technology, and we will welcome more new arrivals in 2014..

All the best for 2014!

Hervé MOUREN
Directeur de TERATEC

High Performance Computing



A Strategy for Research and Innovation through High Performance Computing

Editors

Mark Sawyer, Business Development and Project Manager, EPCC
Mark Parsons, Executive Director, EPCC; Associate Dean for e-Research, University of Edinburgh

PlanetHPC is supported under Objective “Computing Systems” of Challenge 3 “Components and Systems” of the ICT Programme of the European Commission.

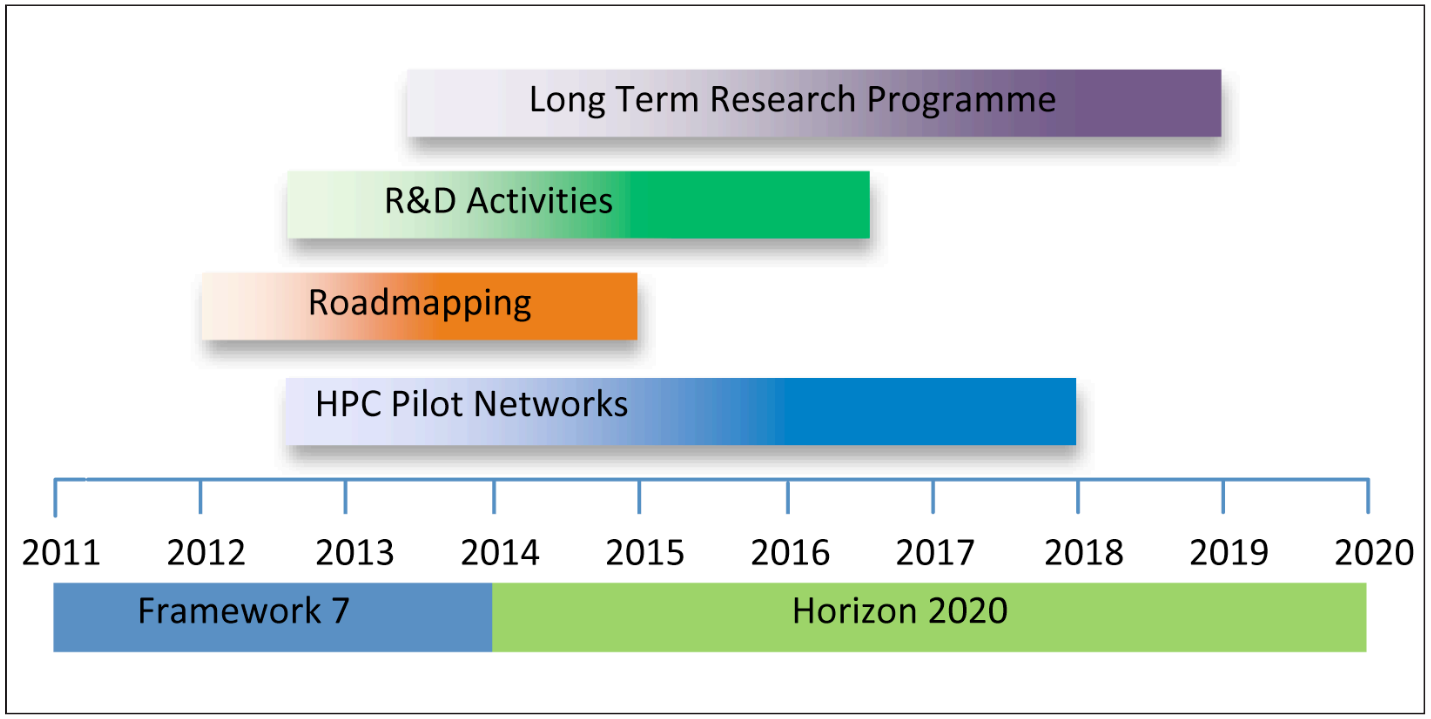
HPC is without doubt a key enabling technology for many technologically advanced nations in the 21st century. Many countries world-wide are investing in HPC and some, most notably the USA, China and Japan are investing vast sums of public money on related infrastructure.

The importance of HPC is summarised well by the statement contained in a recent report by industry experts IDC:

“Today, to Out-Compute is to Out-Compete”.

Timeline of activities

The preparation for this programme of research needs to start as soon as possible in Framework 7 and continue through Horizon 2020. The following timeline is proposed for actions:

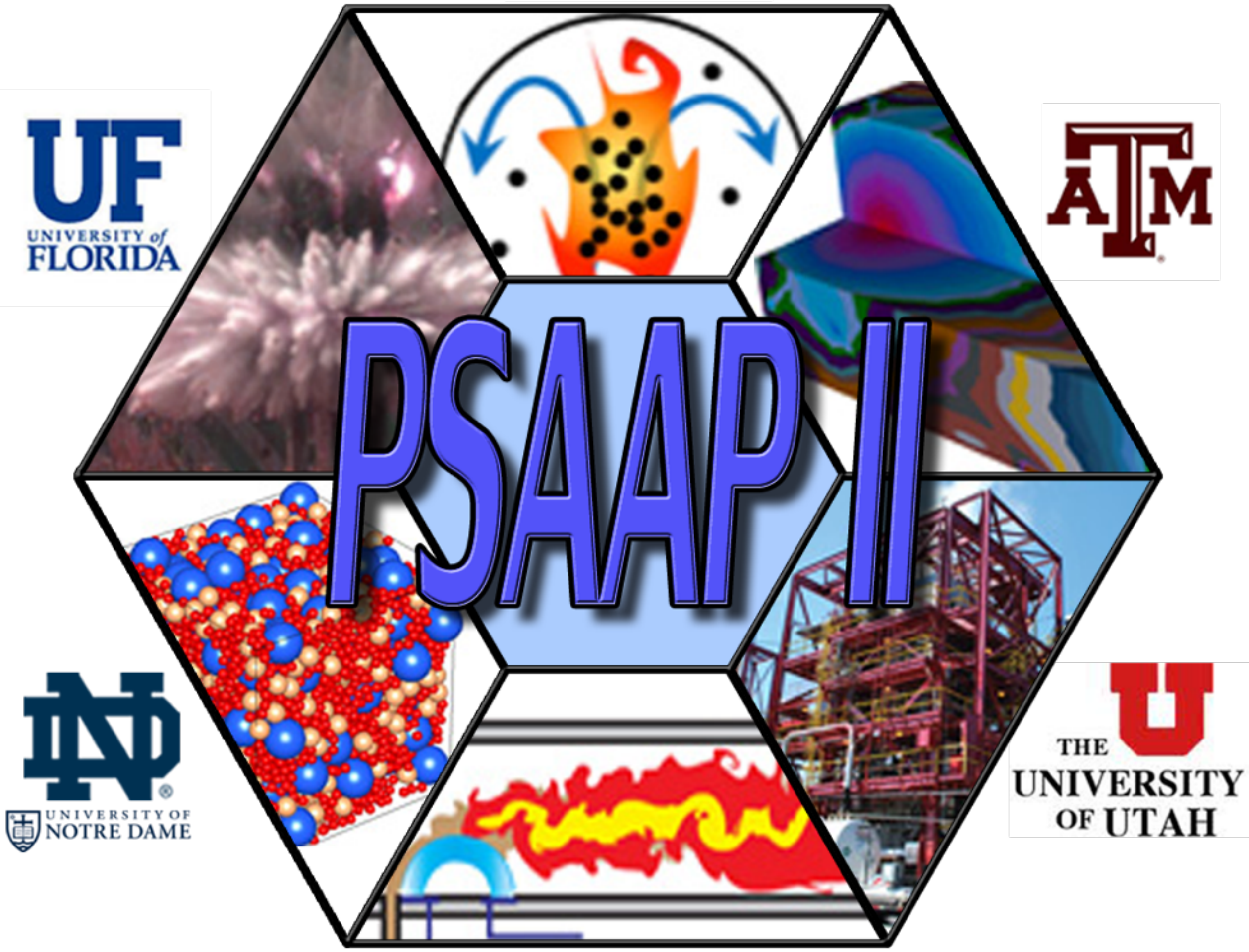


Predictive Science Academic Alliance Program (PSAAP II)

STANFORD UNIVERSITY

UNIVERSITY of FLORIDA

A&M



UNIVERSITY OF NOTRE DAME

THE UNIVERSITY OF UTAH

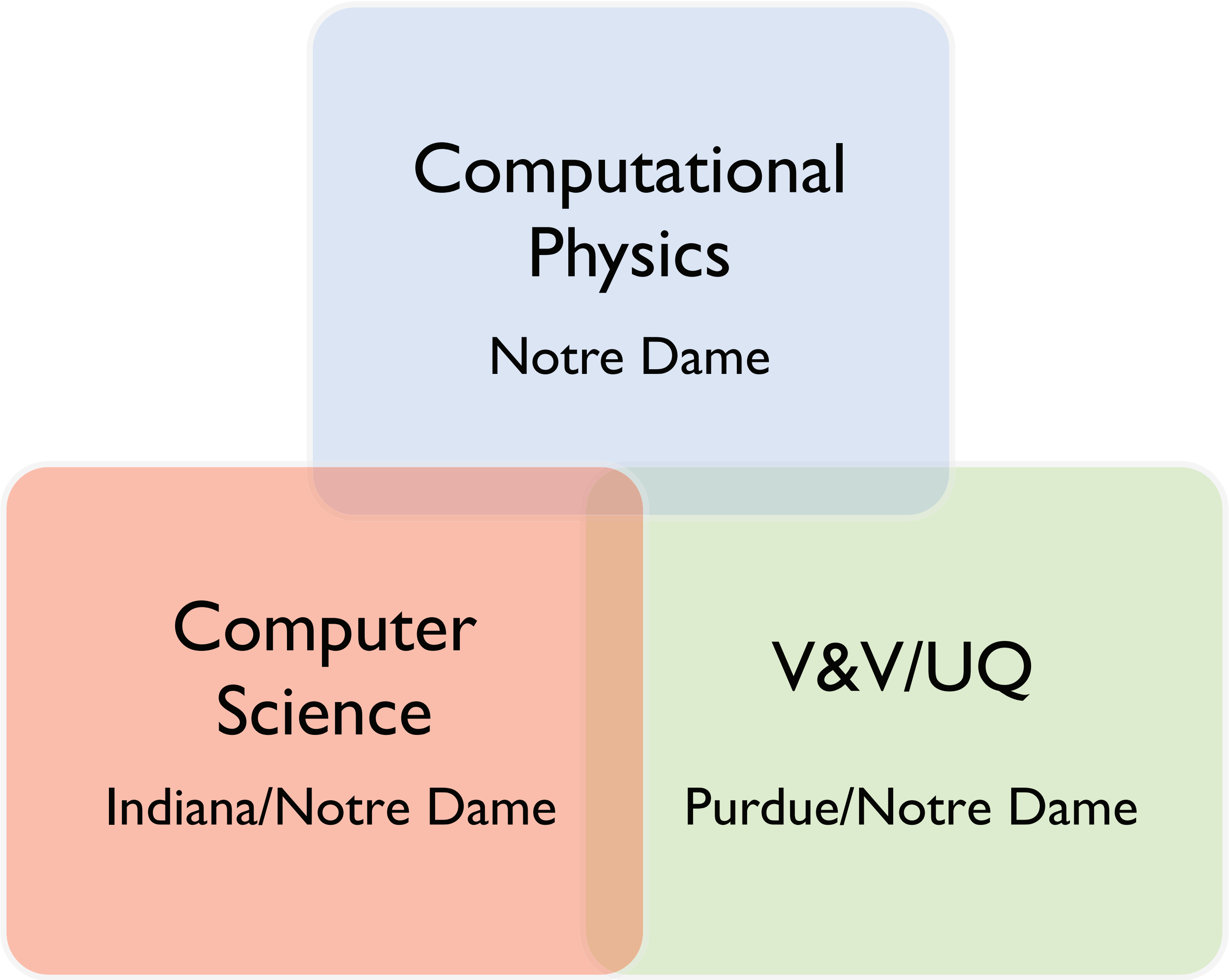
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

NNSA
National Nuclear Security Administration

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

Shock Wave-processing of Advanced Reactive Materials

K. Matous



Computational Physics

Notre Dame

Computer Science

Indiana/Notre Dame

V&V/UQ

Purdue/Notre Dame

A. Mukasyan

S. Paolucci

J. Powers

G. Tryggvason

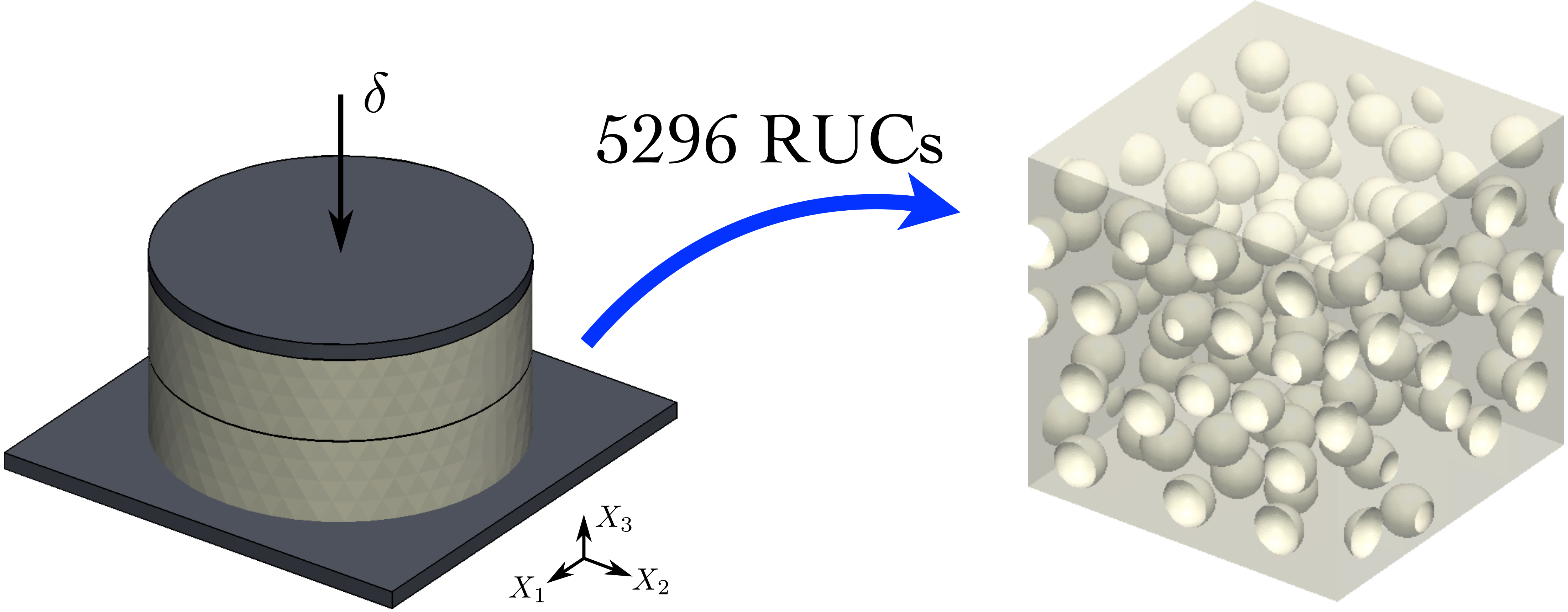
P. Kogge

A. Lumsdaine

T. Sterling

S. Son

Computational Homogenization



5296 RUCs

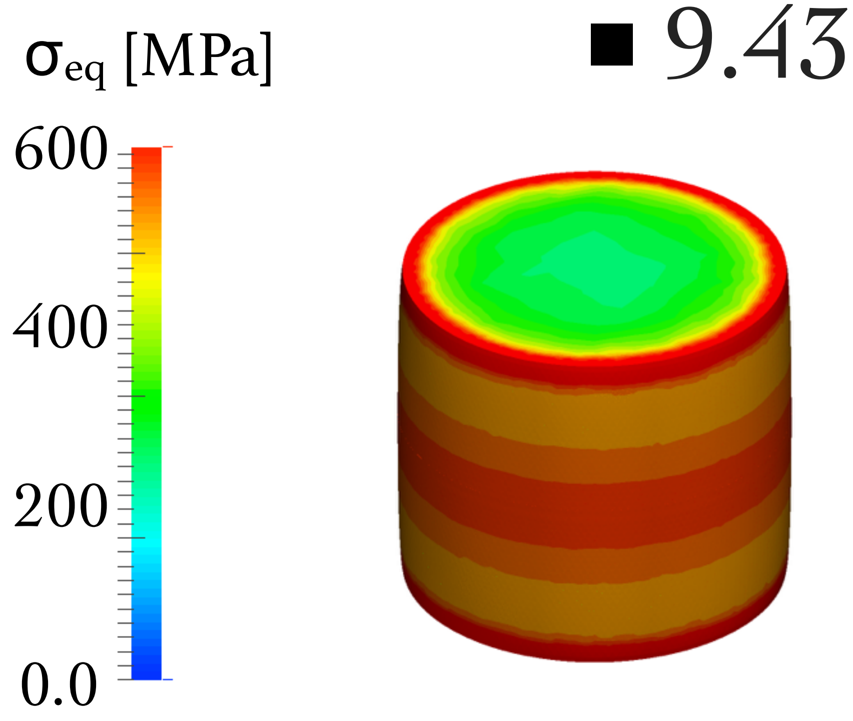
- Macro-scale
  - No-slip on top/bottom
  - h= 20 mm, d = 20 mm
- E = 205 GPa, ν = 0.25
- 320K elements in Macro

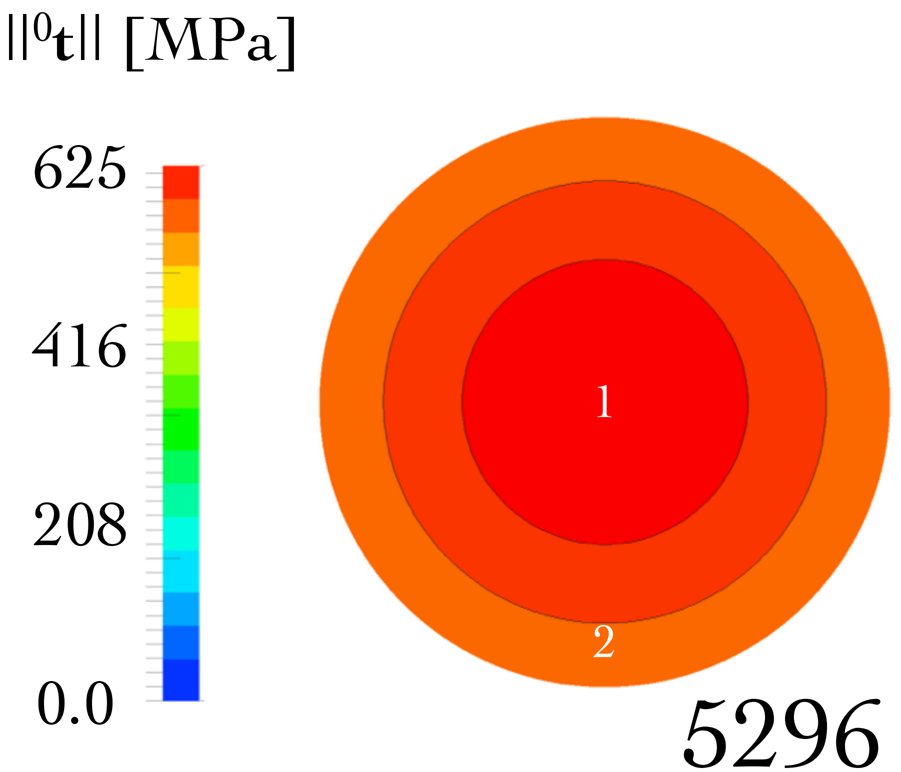
- Micro-scale
  - 210 x 210 x 210 μm^3
  - 98 voids, 30 μm diameter
- E = 3 GPa, ν = 0.29
- 10.2M elements in cell

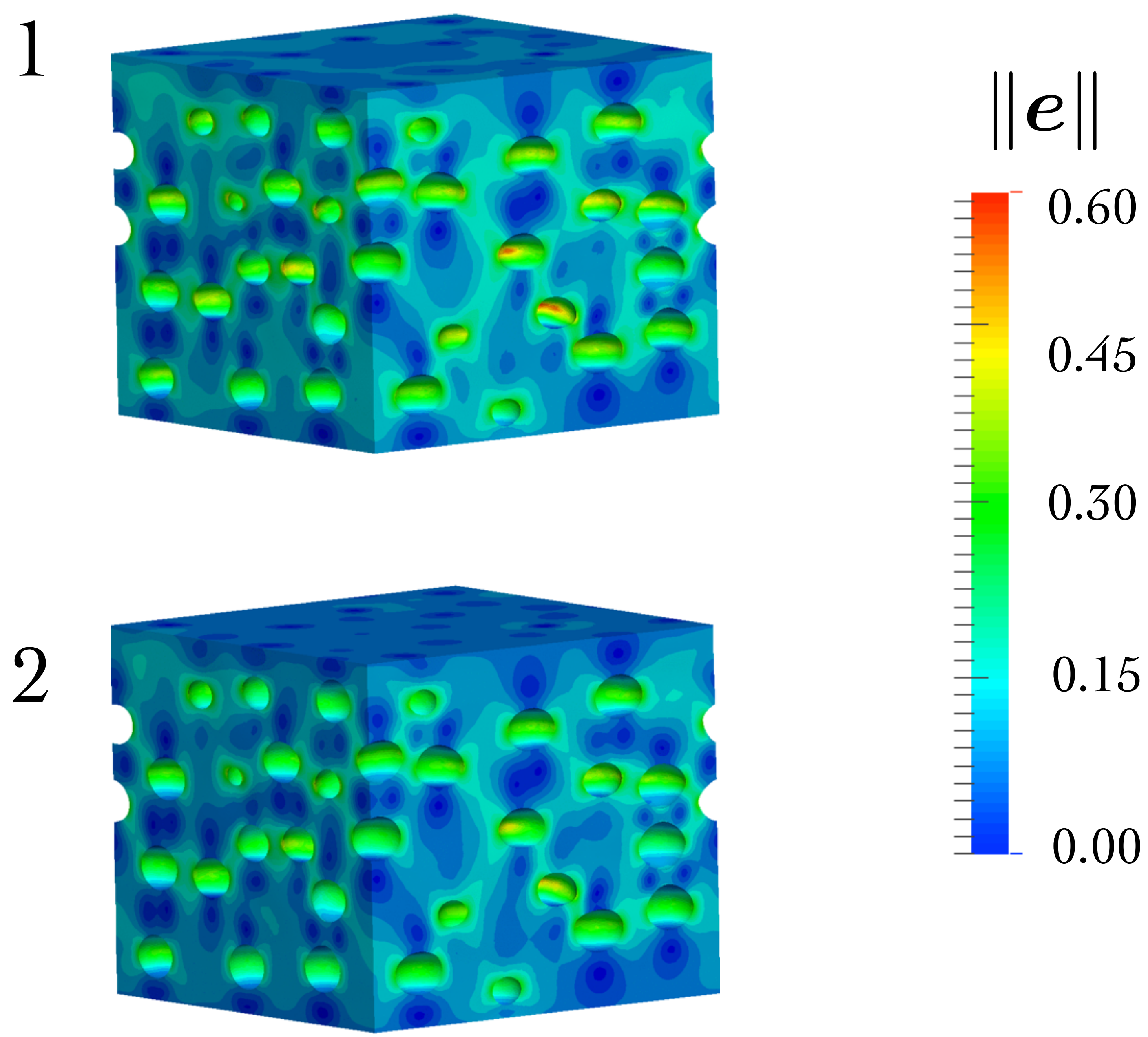
- Nonlinear hyperelastic constitutive model

Multi-scale Simulations, PGFem3D - CH

- Full system on LLNL Vulcan — 393,216 cores, 786,432 threads
  - 9.43B Node, 53.75B Elements, 28.08B DOF

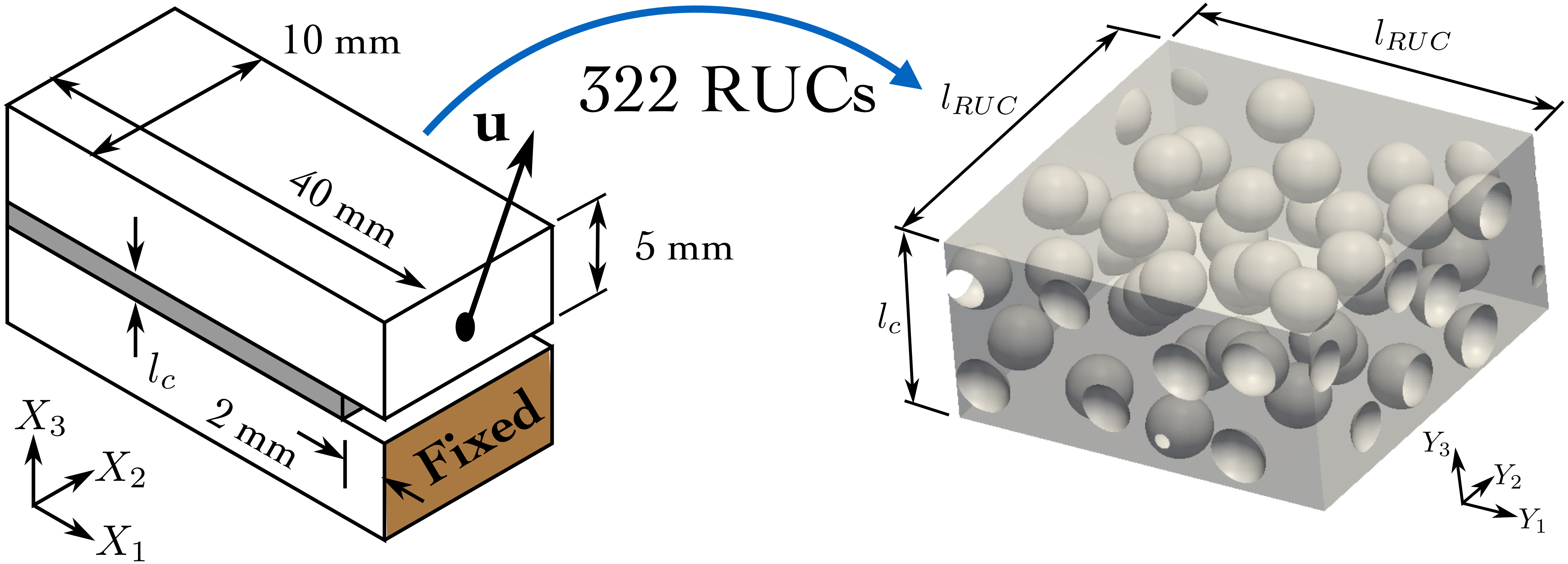






5296 RUCs $h_e(\min)$=191 nm

Multi-scale Simulations, PGFem3D - CH



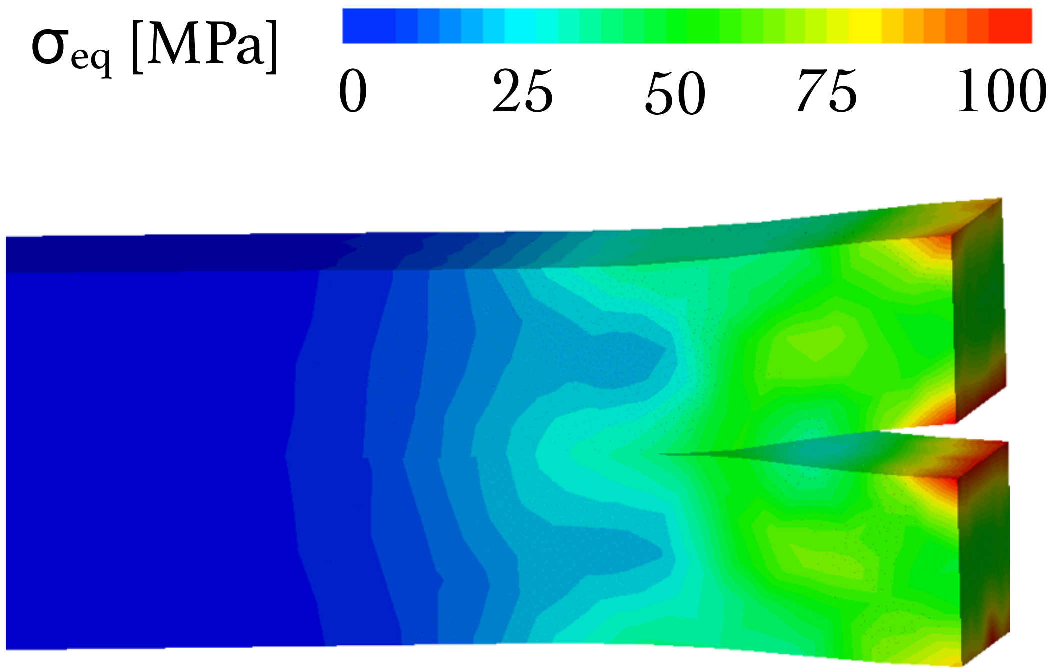
- Macroscale
  - Mode I loading
- E = 15 GPa, ν = 0.25
- 10K elements
- 322 cohesive elements

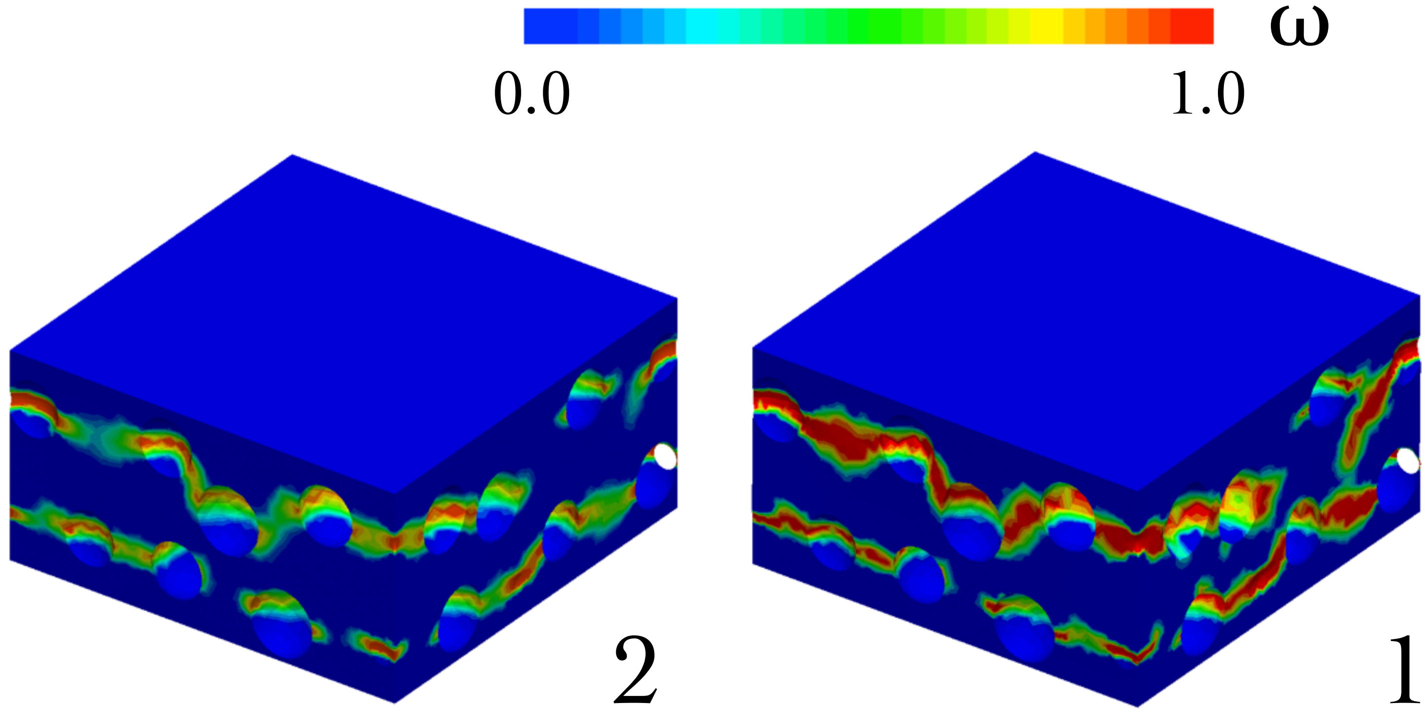
- Microscale
  - 250 x 250 x 125 μm^3
  - 40 voids, 40 μm diameter
- E = 5 GPa, ν = 0.34
- 249K elements in RUC

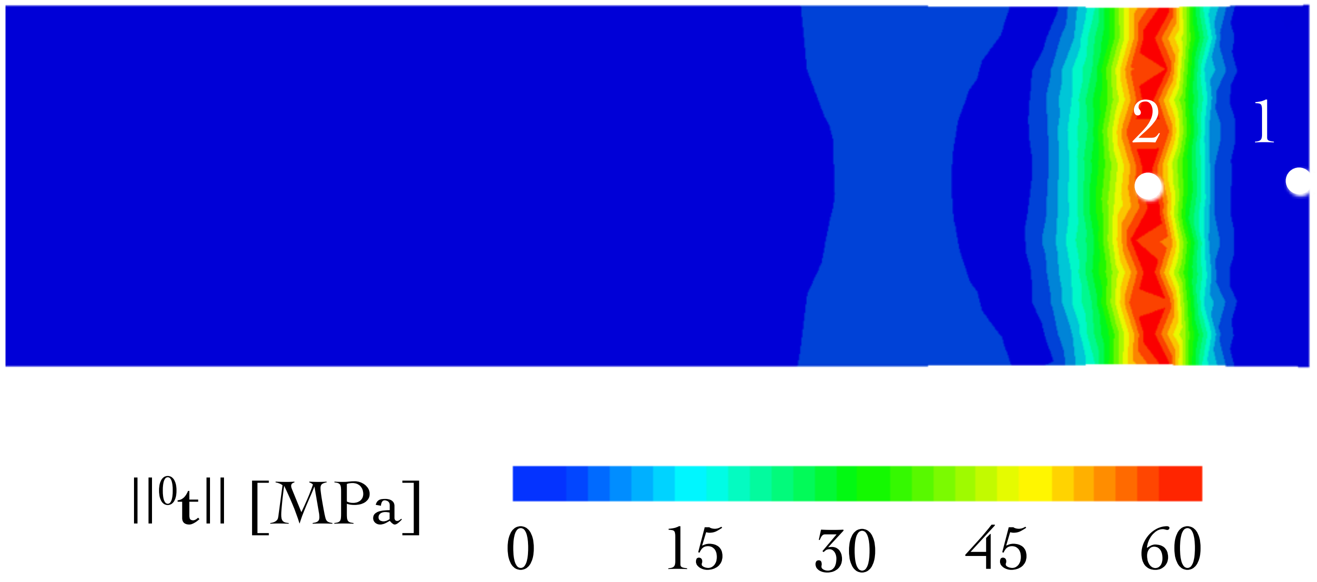
- 80M Elements, 42.5M DOFs

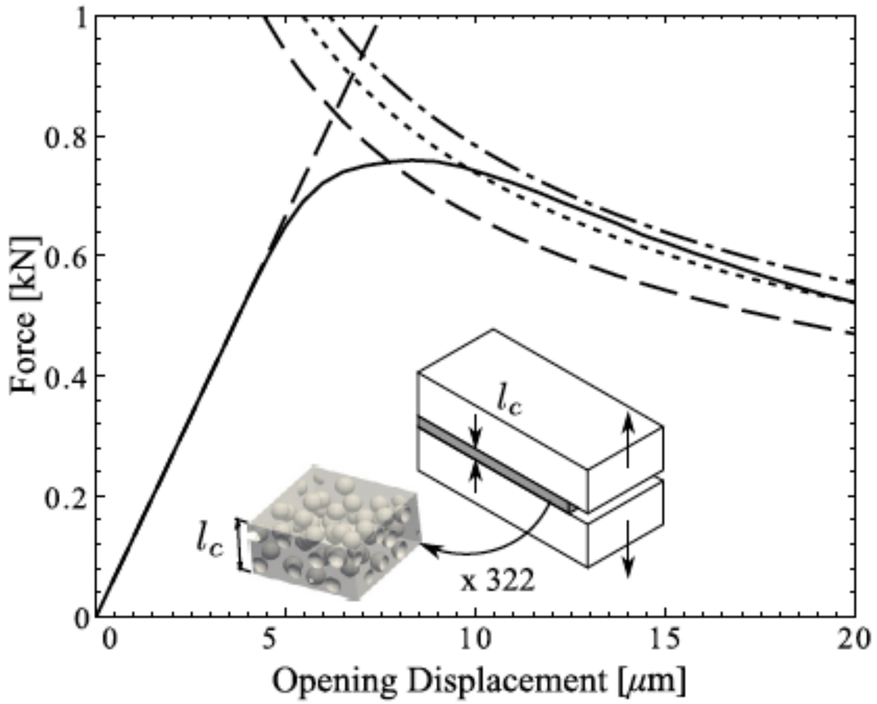
Fully Coupled Multi-scale DCB Failure: Mode-I

- Numerically resolve $O(10^5)$ scales (1 cm to 100 nm)

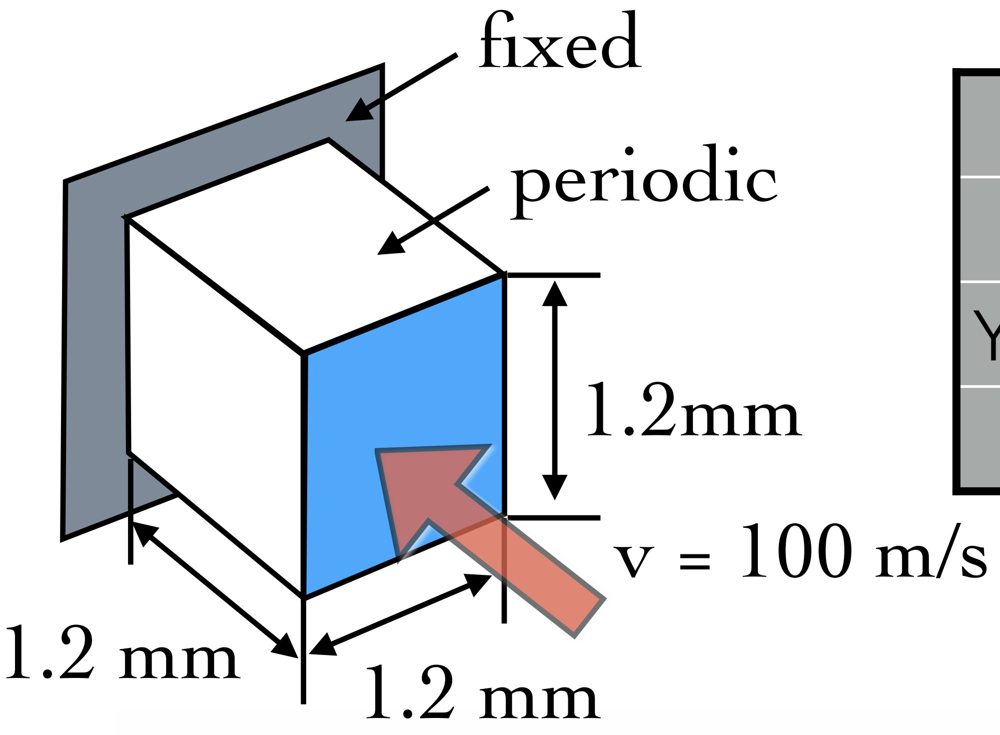








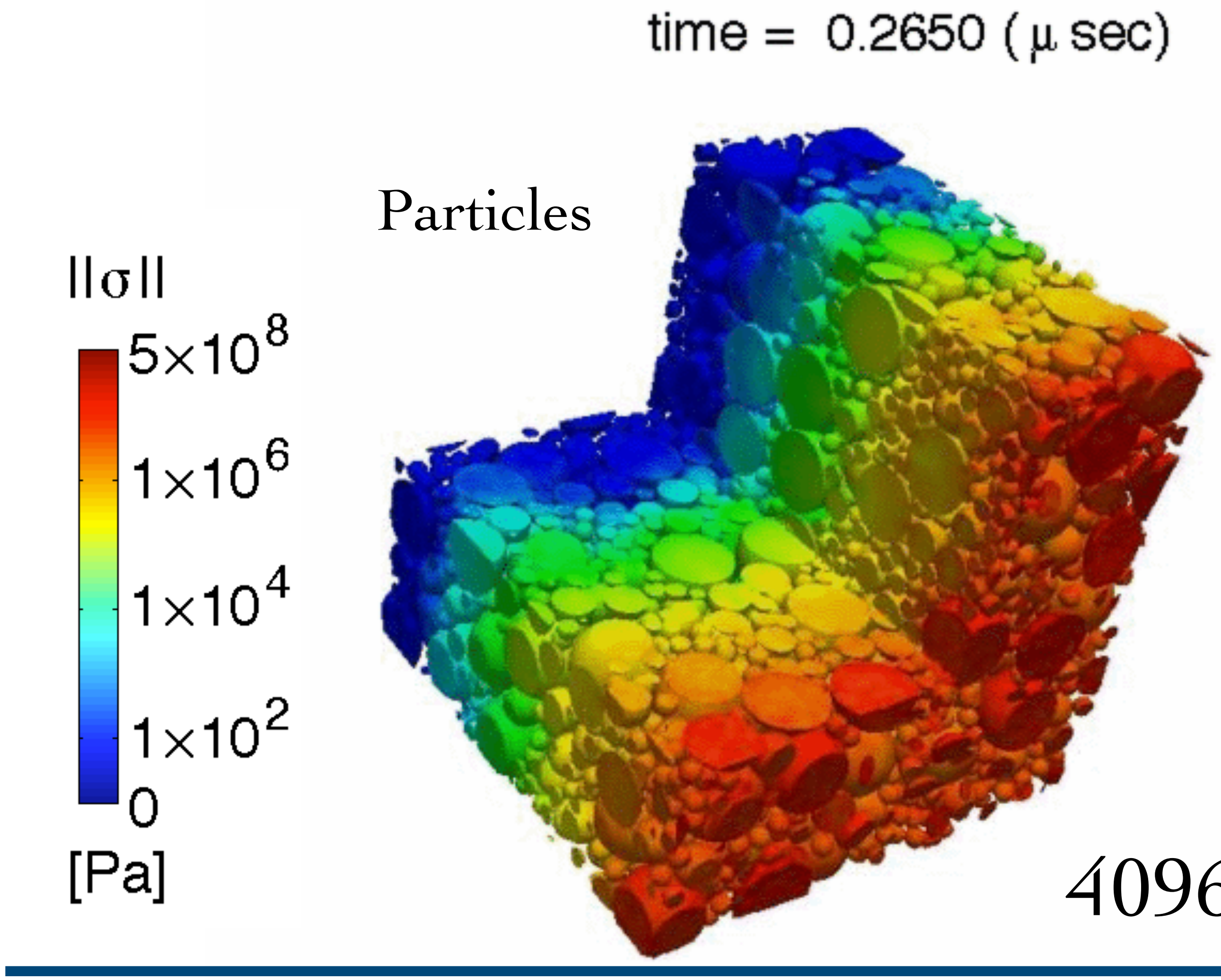
Shock Simulations in Heterogeneous Materials

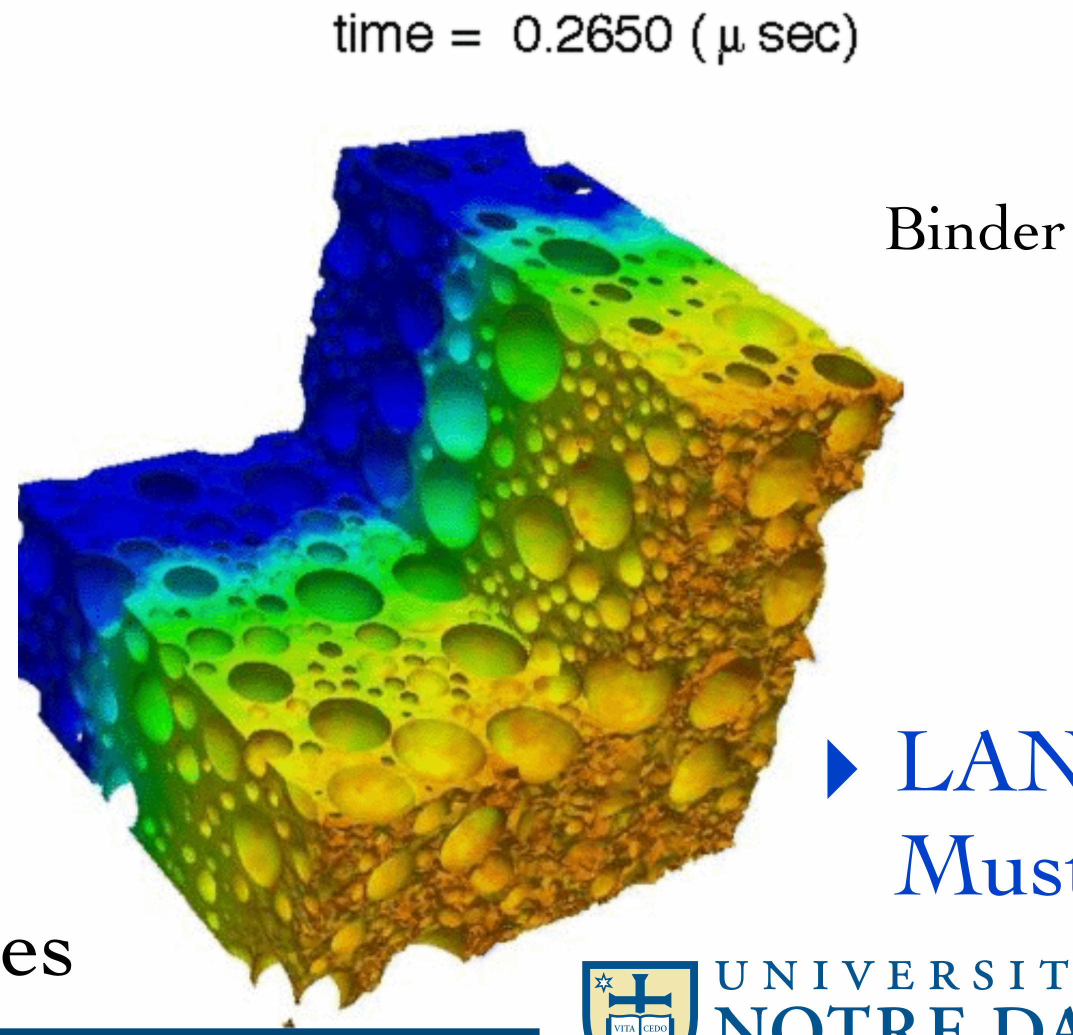


	Ni	Al	Matrix
Particle radius [μm]	100	50 / 25	-
Young's modulus [GPa]	225.9	100	0.1
Density [kg/m^3]	9000	2700	500

Number of elements : 18,227,610

Δt = 0.2 ns, c_{par} = 0.7





4096 cores

- LANL Mustang

UNIVERSITY OF NOTRE DAME

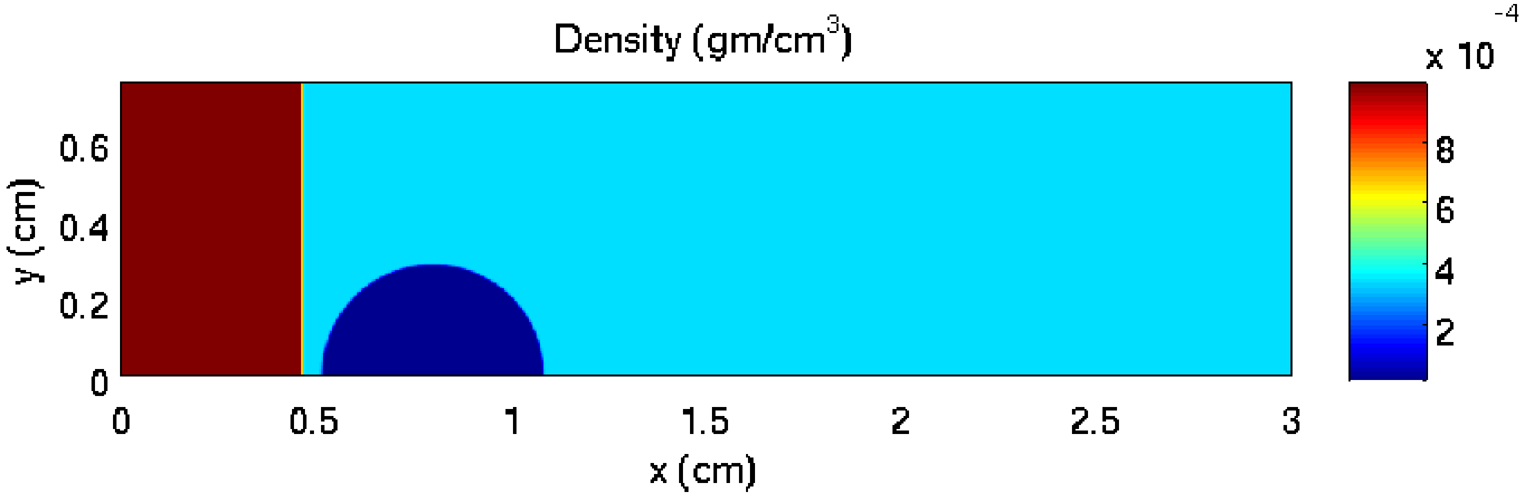
Macro-continuum Example - *WAMR*

- Domain
  $[0, 5] \times [0, 0.75]$ cm

- Ambient mixture
  $Y_{N2} = 0.868$, $Y_{O2} = 0.232$
  P = 101.3 kPa
  T = 1000 K

- Hydrogen bubble
  $Y_{H2} = 0.99$, $Y_{air} = 0.01$
  $x = 0.80$ cm

- Chemical model
  9 species, 38 reactions

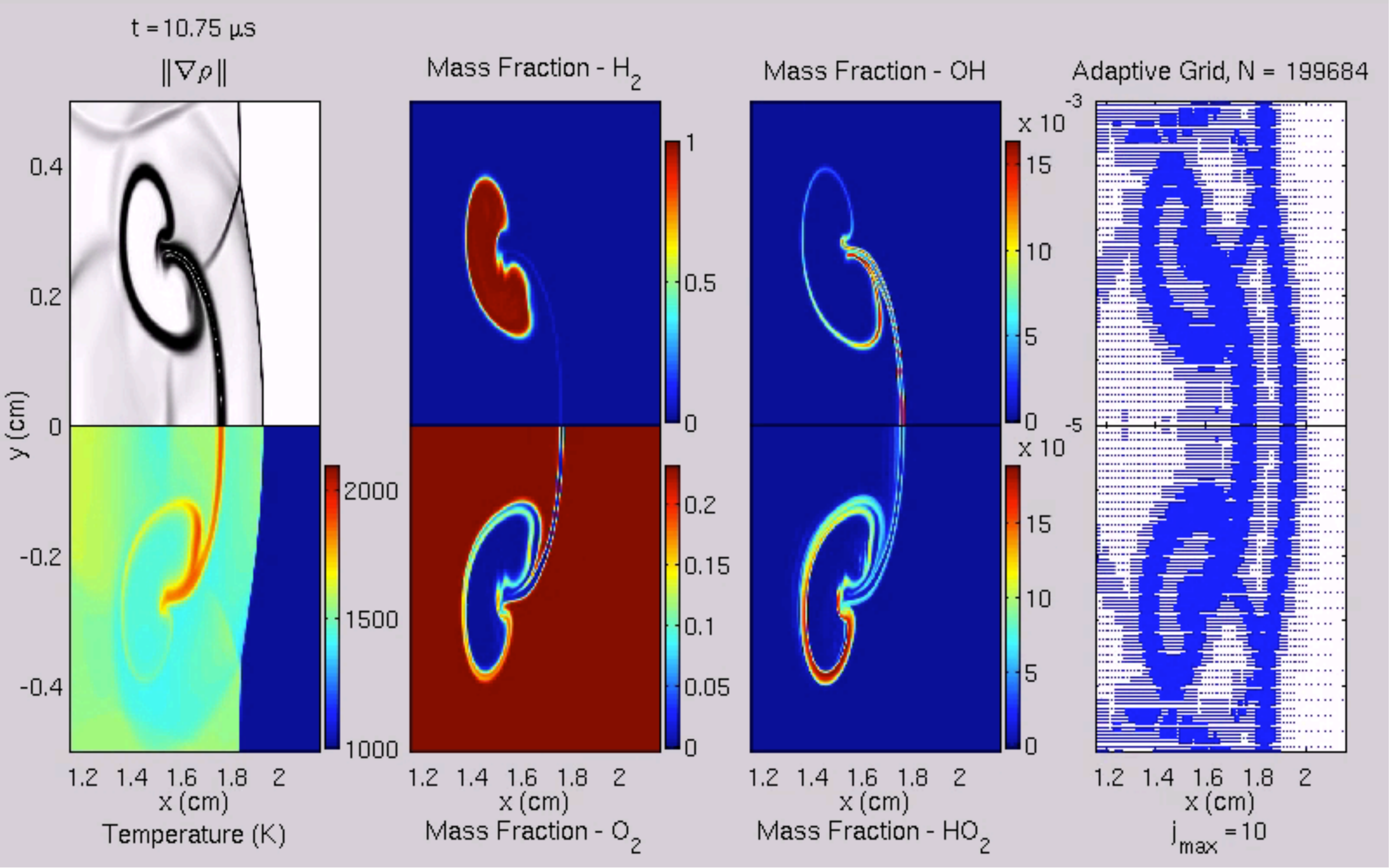


- Loaded by shock Ms = 2.0 at
  $x = 0.46$ cm

- Wavelet parameters
  $\varepsilon = 10^{-3}$
  p=6
  $[N_x \times N_y]_{coarse} = [50 \times 8]$
  J = 14

- 256 cores

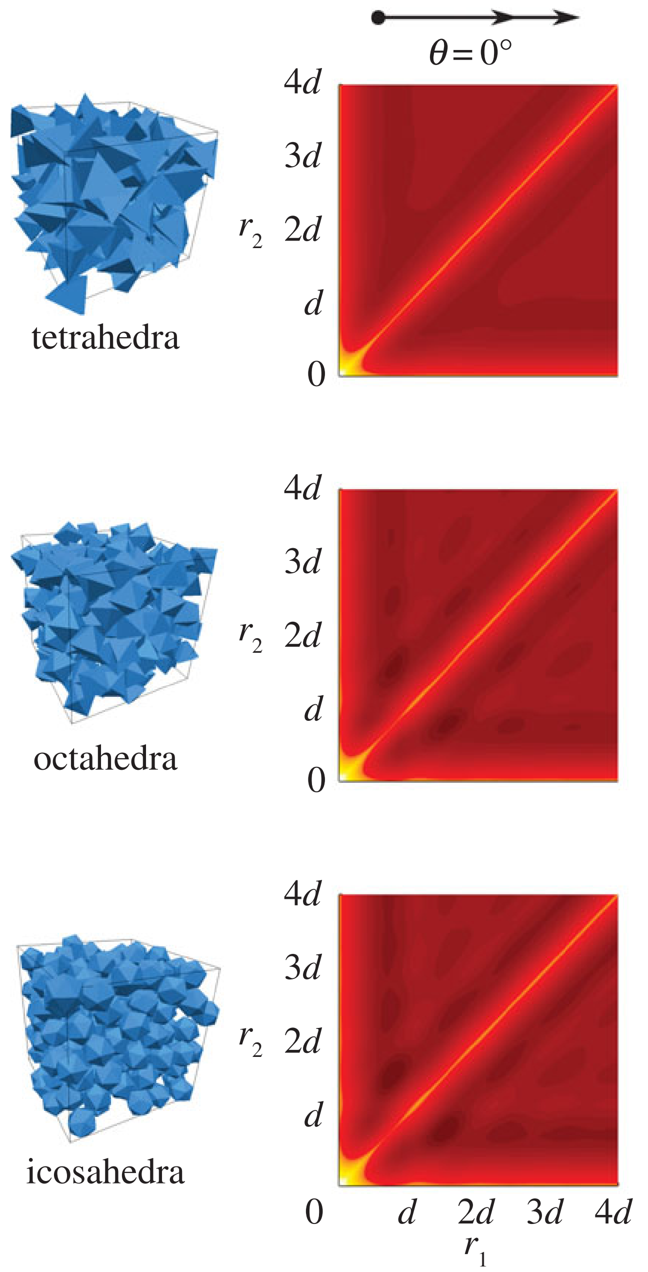
Macro-continuum Example - *WAMR*

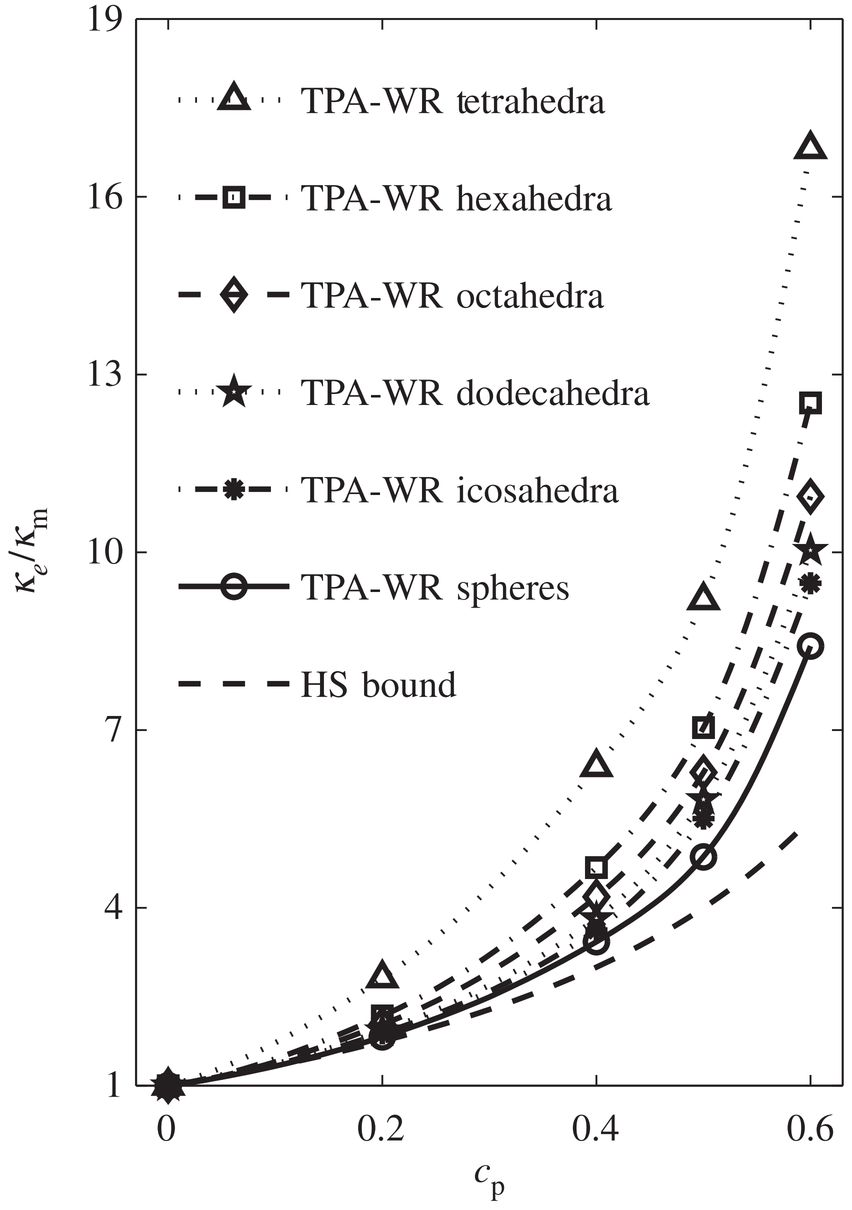


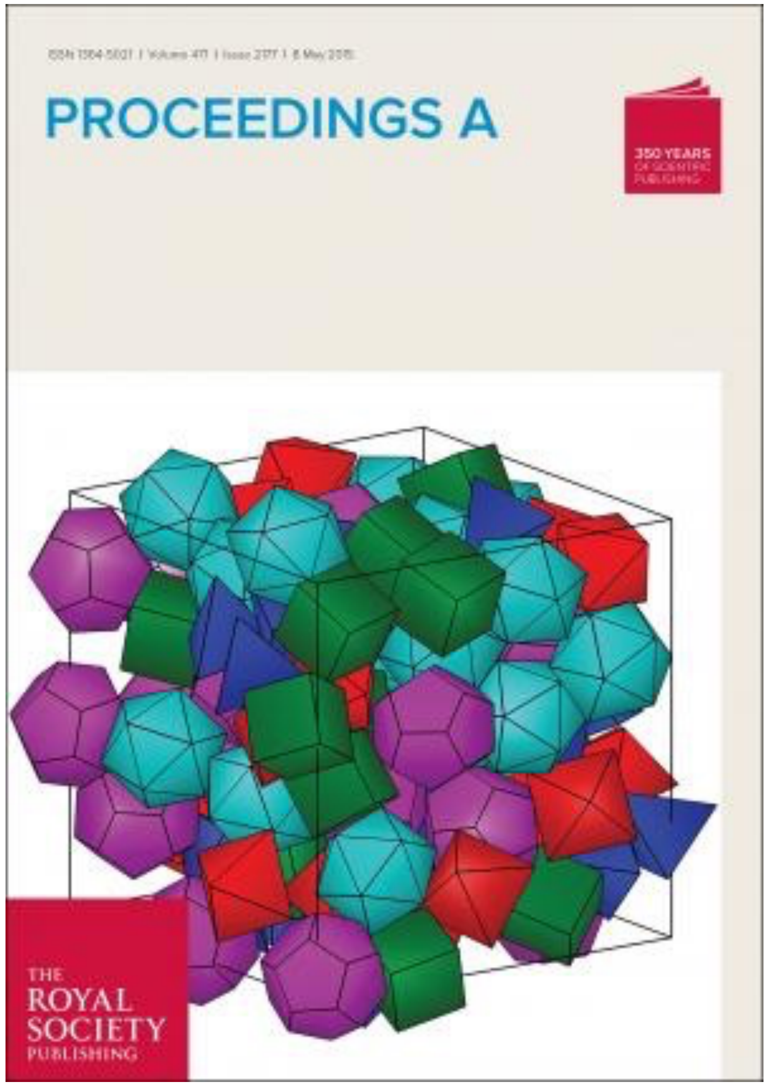
- Resolution required < 1 micron

Statistical Micromechanics

Third-order statistics



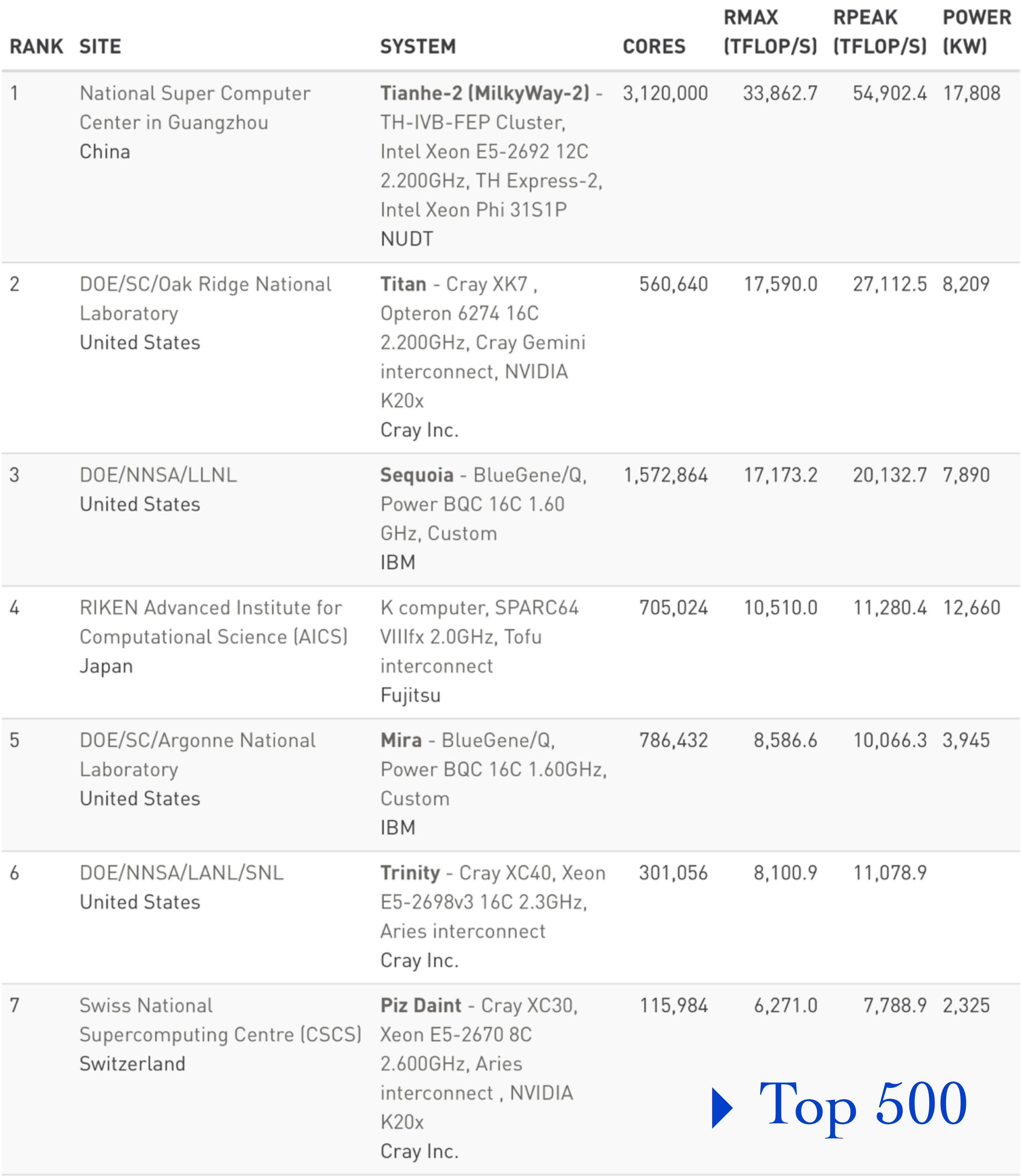




PROCEEDINGS A
350 YEARS OF SCIENTIFIC PUBLISHING
THE ROYAL SOCIETY PUBLISHING

- Morphology is important
- LANL Mustang, 7200 cores

HPC architecture trends

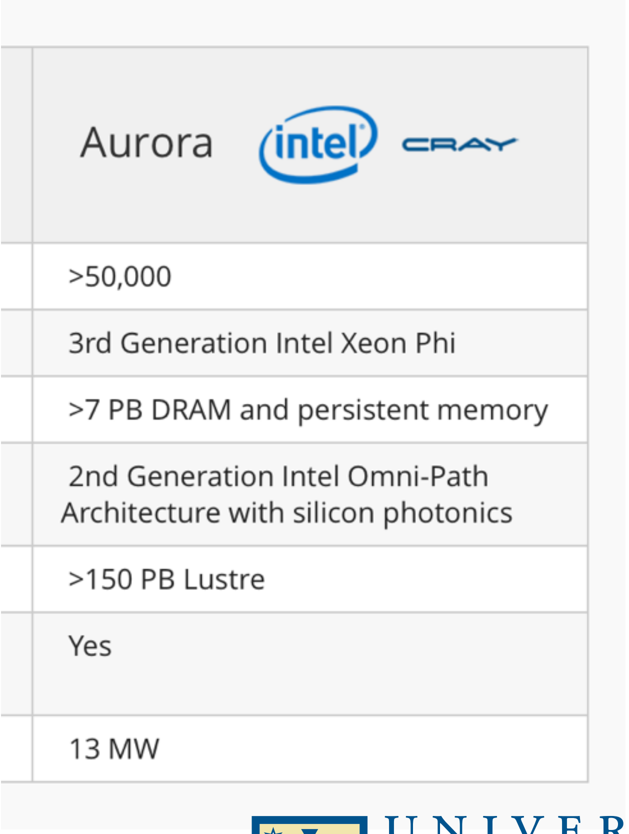


RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
7	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325

- Top 500

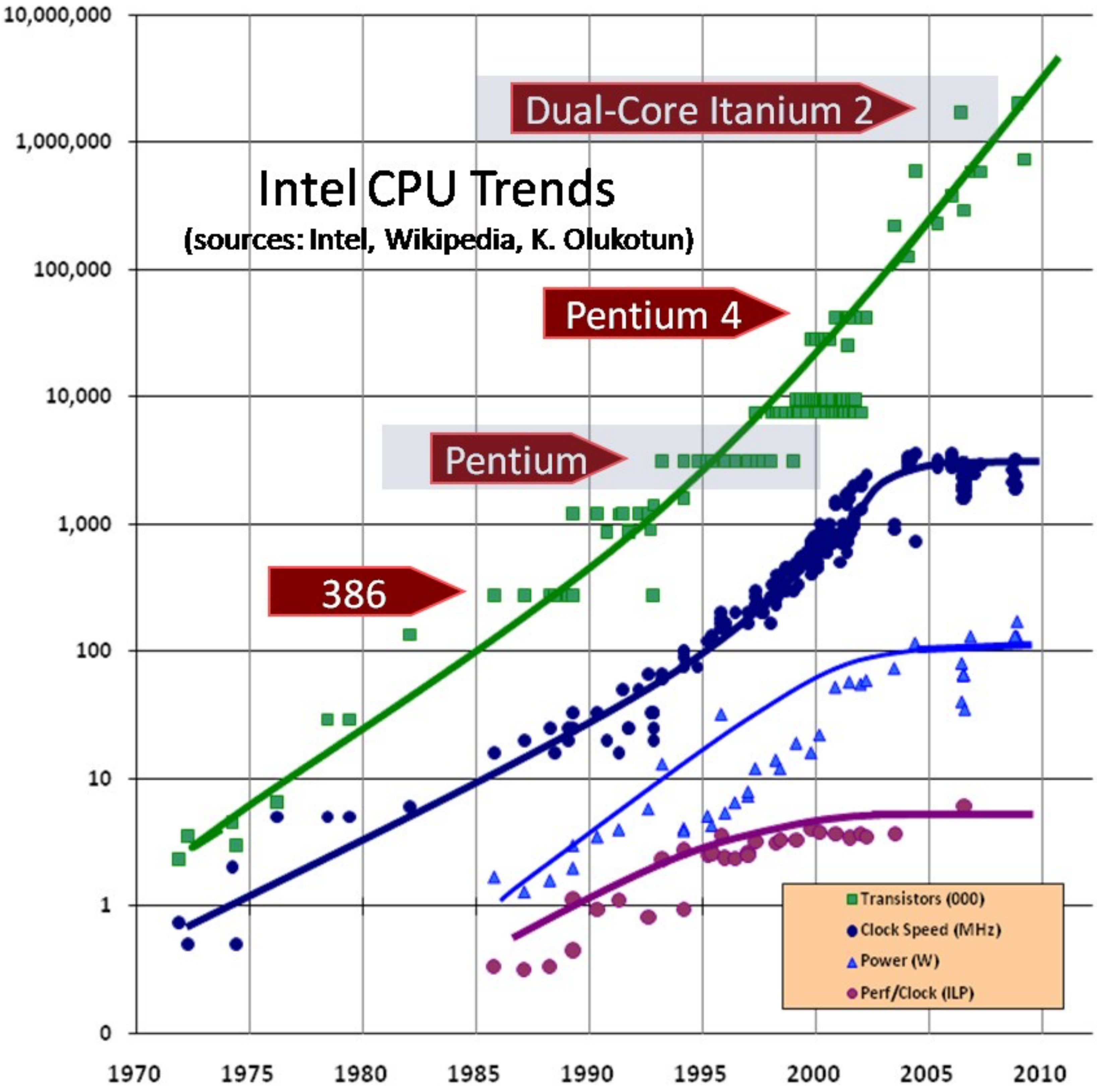
Collaboration of Oak Ridge, Argonne, and Lawrence Livermore (CORAL) — $525 Million

- SUMMIT — Oak Ridge
- SIERRA — LLNL
- AURORA — Argonne



Aurora intel CRAY
>50,000
3rd Generation Intel Xeon Phi
>7 PB DRAM and persistent memory
2nd Generation Intel Omni-Path Architecture with silicon photonics
>150 PB Lustre
Yes
13 MW

Technology Demands new Response



The Free Lunch Is Over

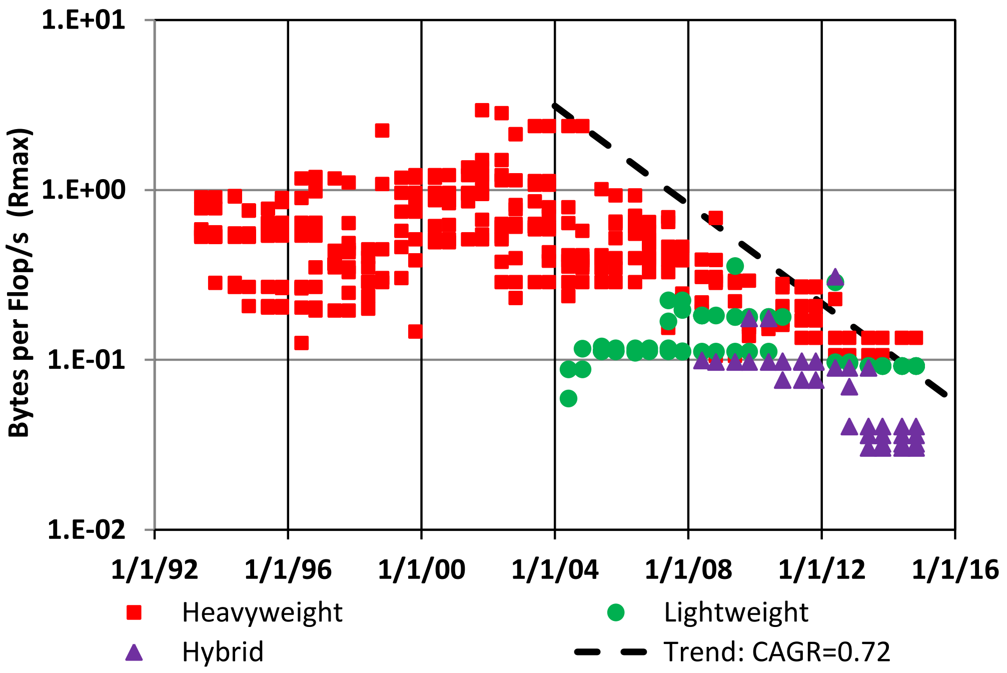
A Fundamental Turn Toward Concurrency in Software

By Herb Sutter, 2005

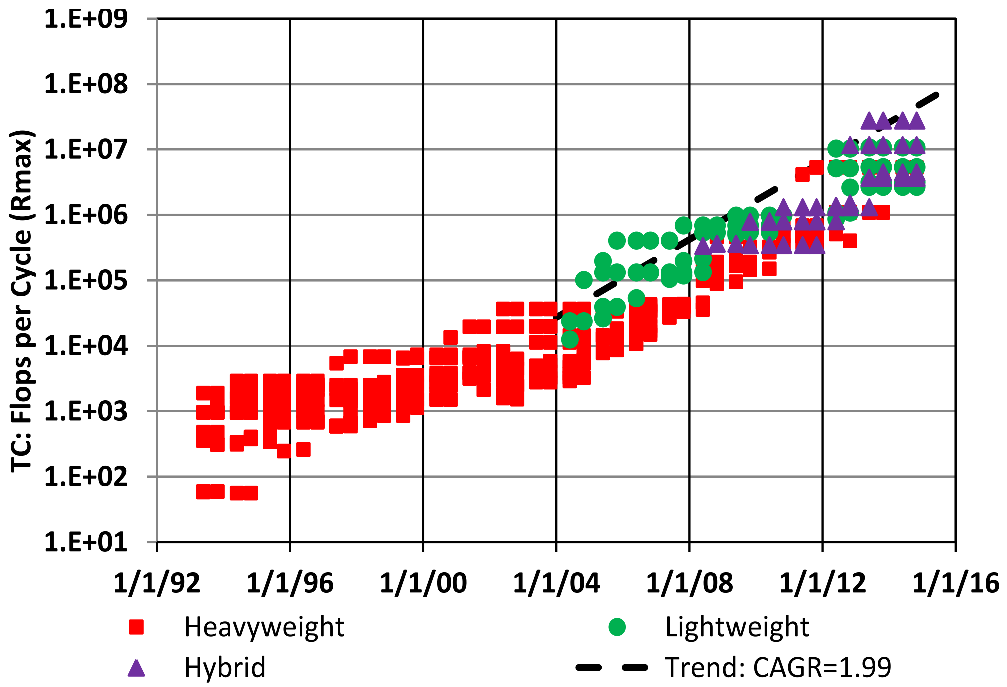
"There ain't no such thing as a free lunch." —Robert A. Heinlein
The Moon Is a Harsh Mistress

Extreme Scale Architectures

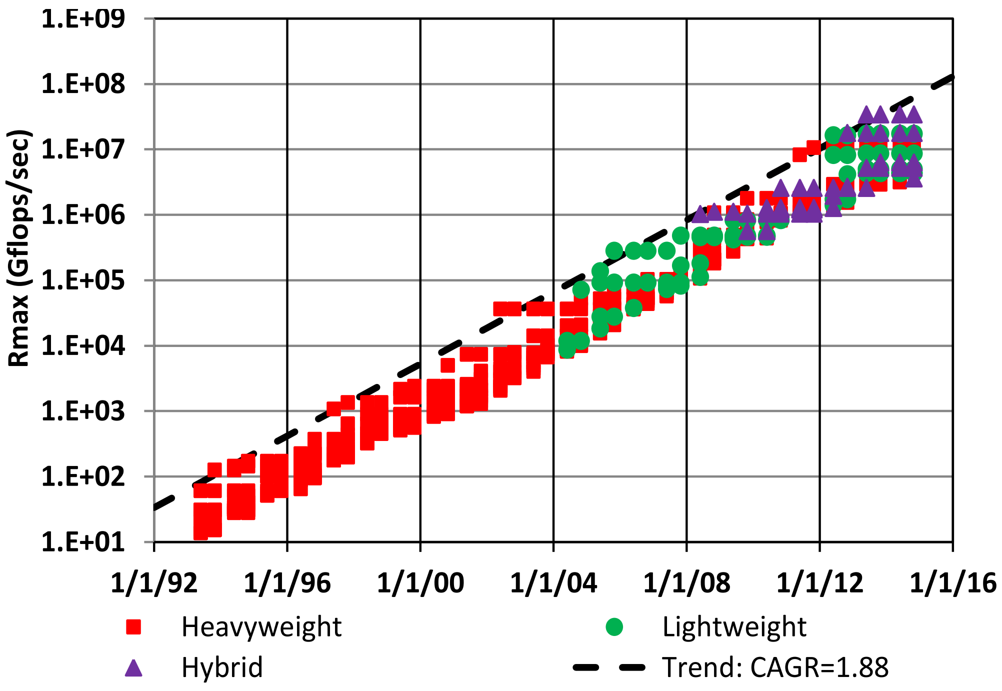
Memory/Flop/s Is Cratering

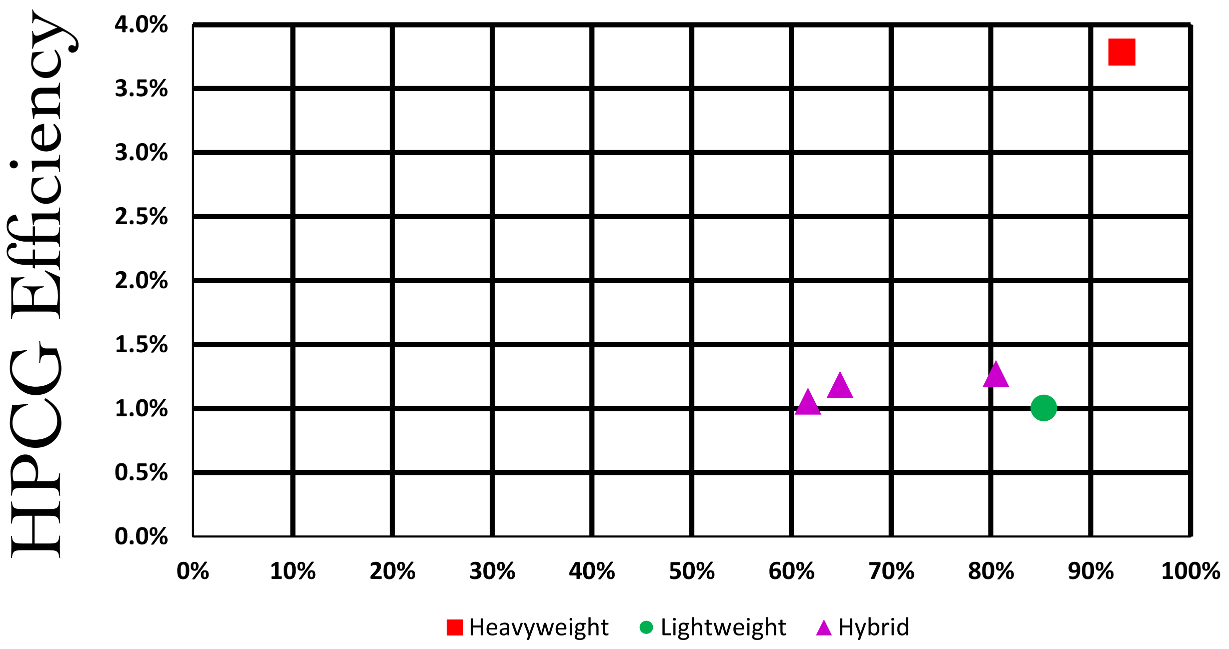


Concurrency Is Skyrocketing



LINPACK (TOP500) Still Increasing



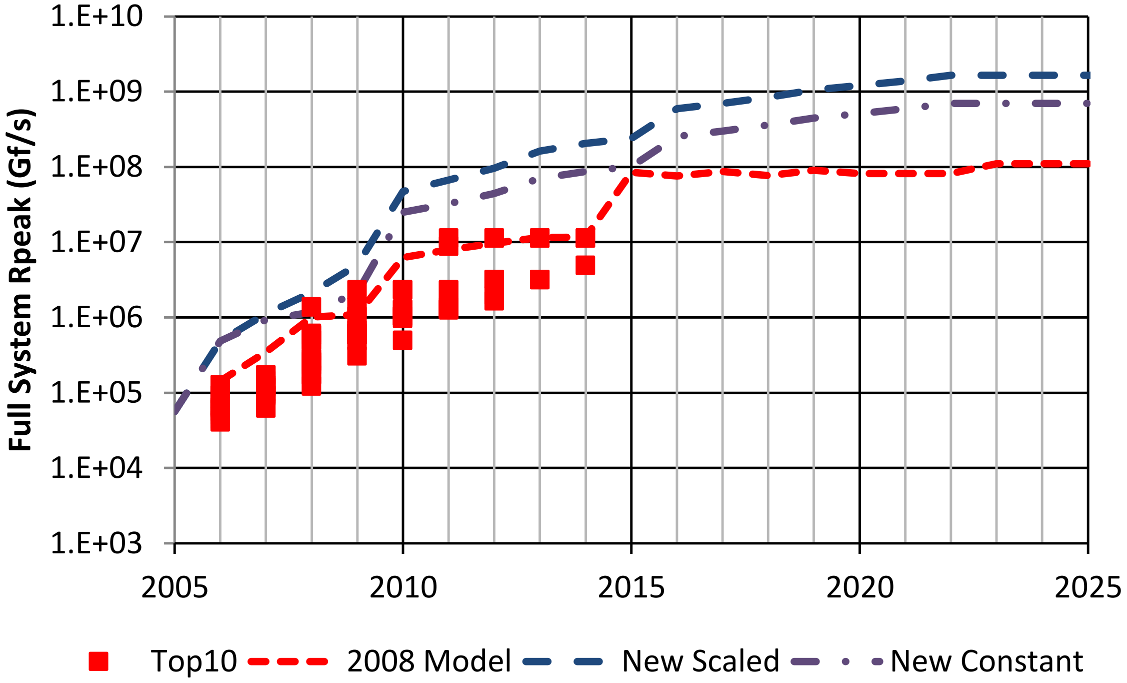


LINPACK Efficiency

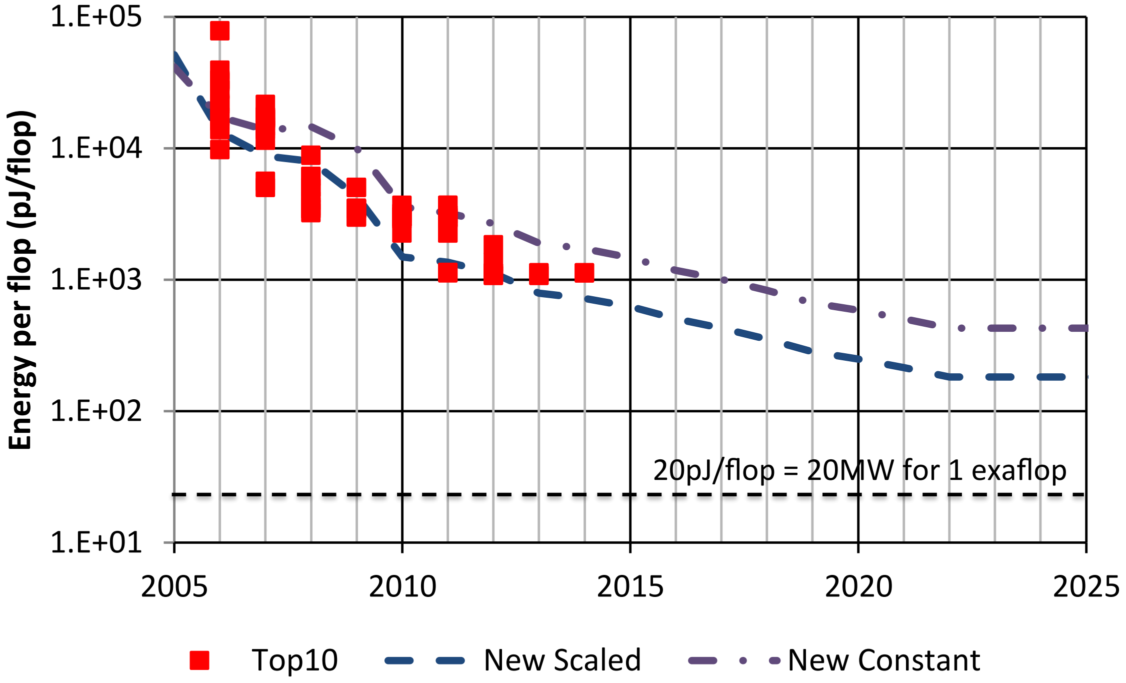
UNIVERSITY OF NOTRE DAME

Extreme Scale Architectures

We Might Get a HeavyWeight Exaflop/s



Energy/Flop Predicts 0.5 GW



▸Looking Forward

- Flops are not the question (esp. dense)
- Conventional heavyweight multi-cores with statically located threads are not the answer
- Problem lies in
  - Memory systems (bandwidth, latency)
  - Handling massive concurrency, asynchrony
- Exascale architectures must be memory-centric, with ability for threads to move

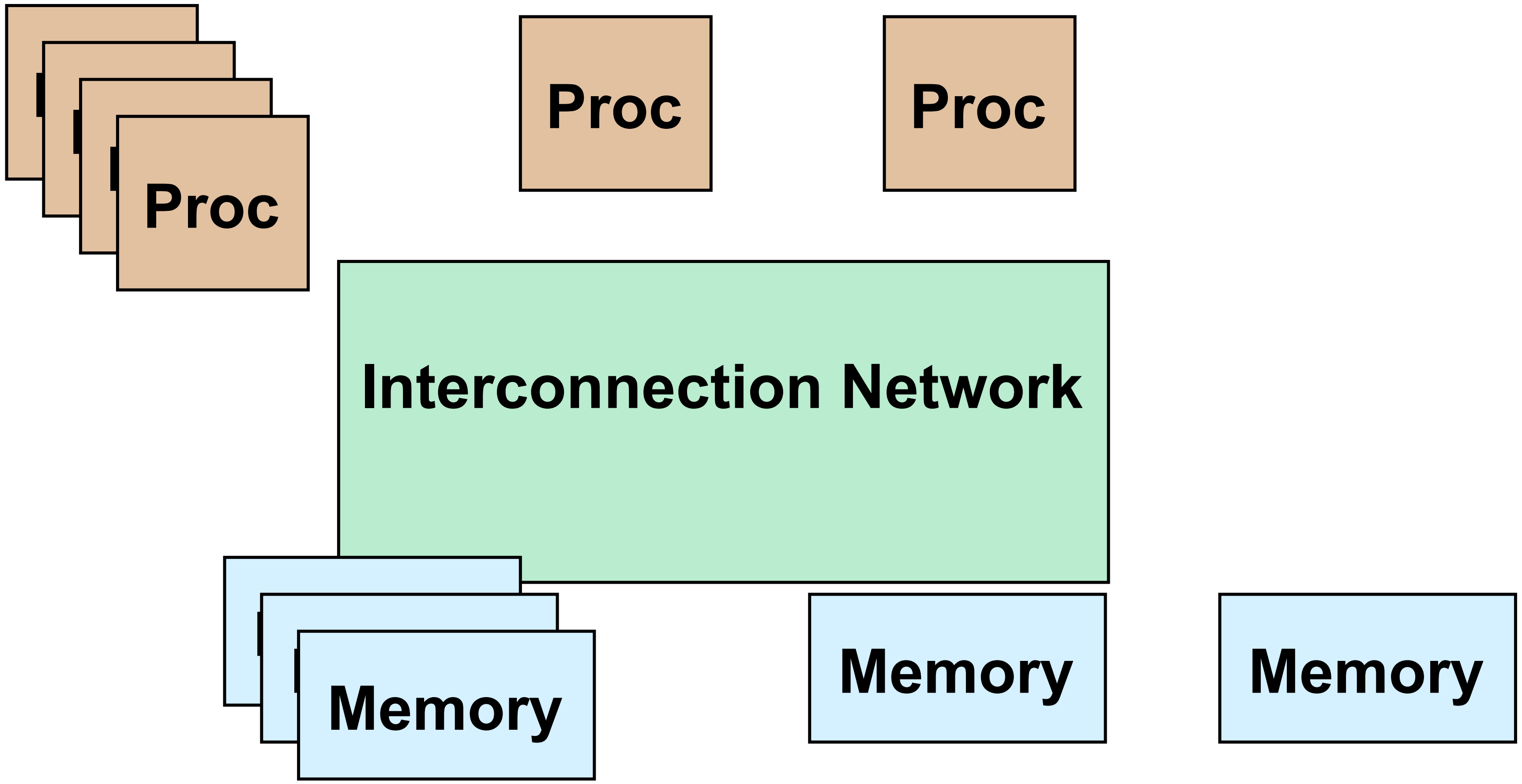
▸It is the Memory, not the Core

▸It is the Data, not the Computations

UNIVERSITY OF NOTRE DAME

On node, and node to node parallelism

- A generic parallel architecture



- Where is the memory physically located?
- Is it connected directly to processors?
- What is the connectivity of the network?

Parallel Programming Models

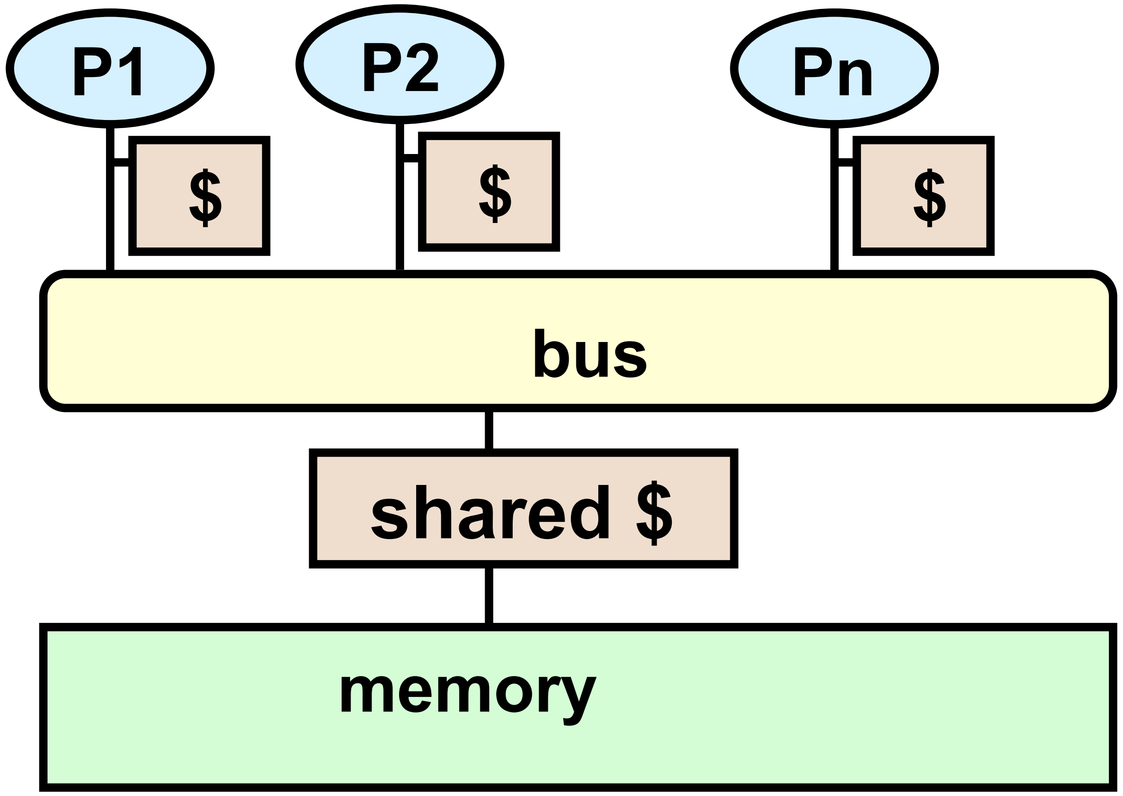
- Programming model is made up of the languages and libraries that create an abstract view of the machine
- Control
  - How is parallelism created?
  - What orderings exist between operations?
- Data
  - What data is private vs. shared?
  - How is logically shared data accessed or communicated?
- Synchronization
  - What operations can be used to coordinate parallelism?
  - What are the atomic (indivisible) operations?
- Cost
  - How do we account for the cost of each of the above?

Parallel Programming Models

Programming Models	Machine Models
1. Shared Memory	1a. Shared Memory 1b. Multithreaded Procs. 1c. Distributed Shared Mem.
2. Message Passing	2a. Distributed Memory 2b. Internet & Grid Computing
2a. Global Address Space	2c. Global Address Space
3. Data Parallel	3a. SIMD (Single Instruction Multiple Data) 3b. Vector
4. Hybrid	4. Hybrid

Programming Model 1a: Shared Memory

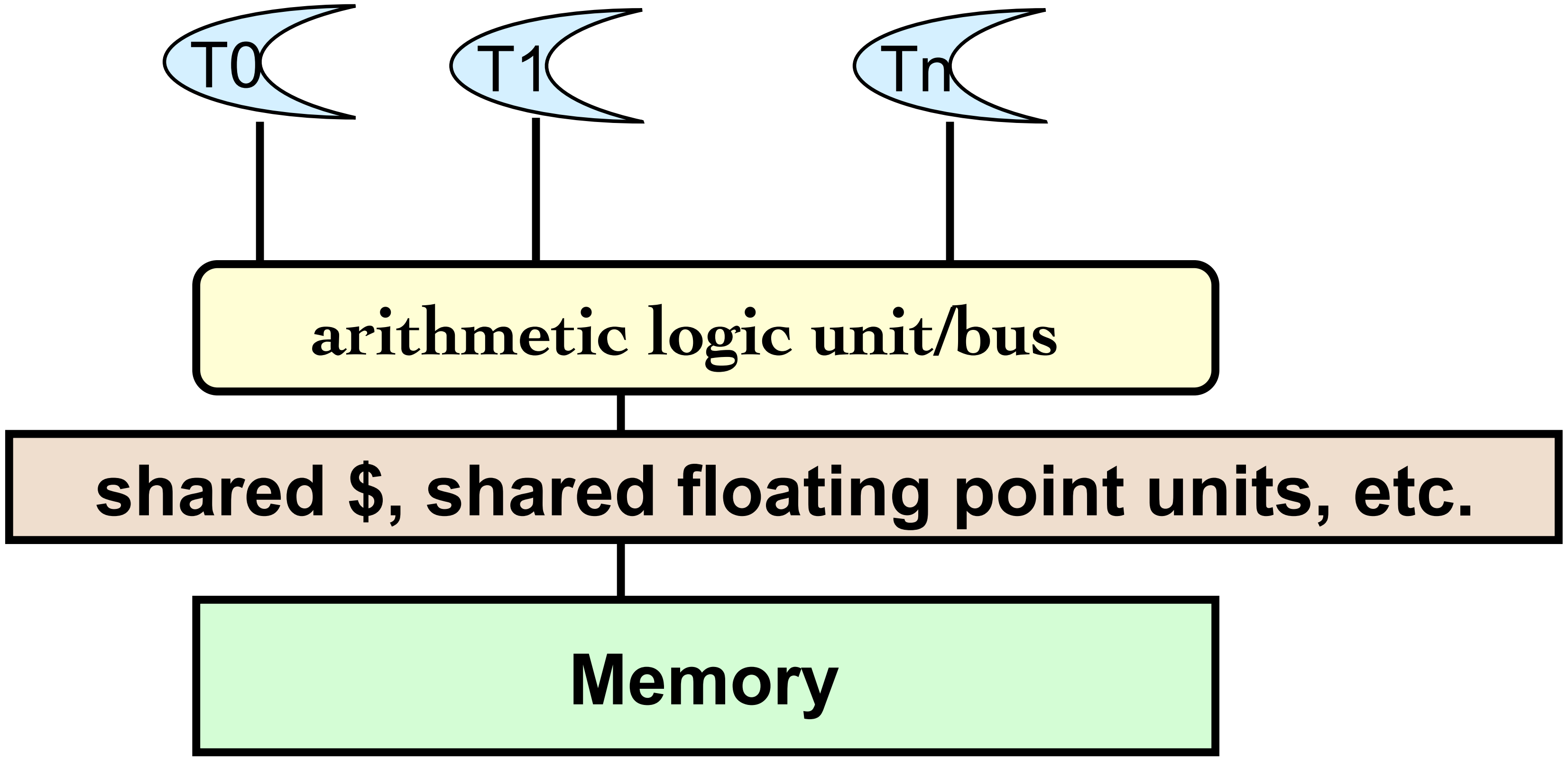
- Processors all connected to a large shared memory.
  - Typically called Symmetric Multiprocessors (SMPs)
  - SGI, Sun, HP, Intel, IBM SMPs
  - Multicore chips, except that all caches are shared
- Advantage: uniform memory access (UMA)
- Cost: much cheaper to access data in cache than main memory
- Difficulty scaling to large numbers of processors
  - <= 32 processors typical



Note: $ = cache

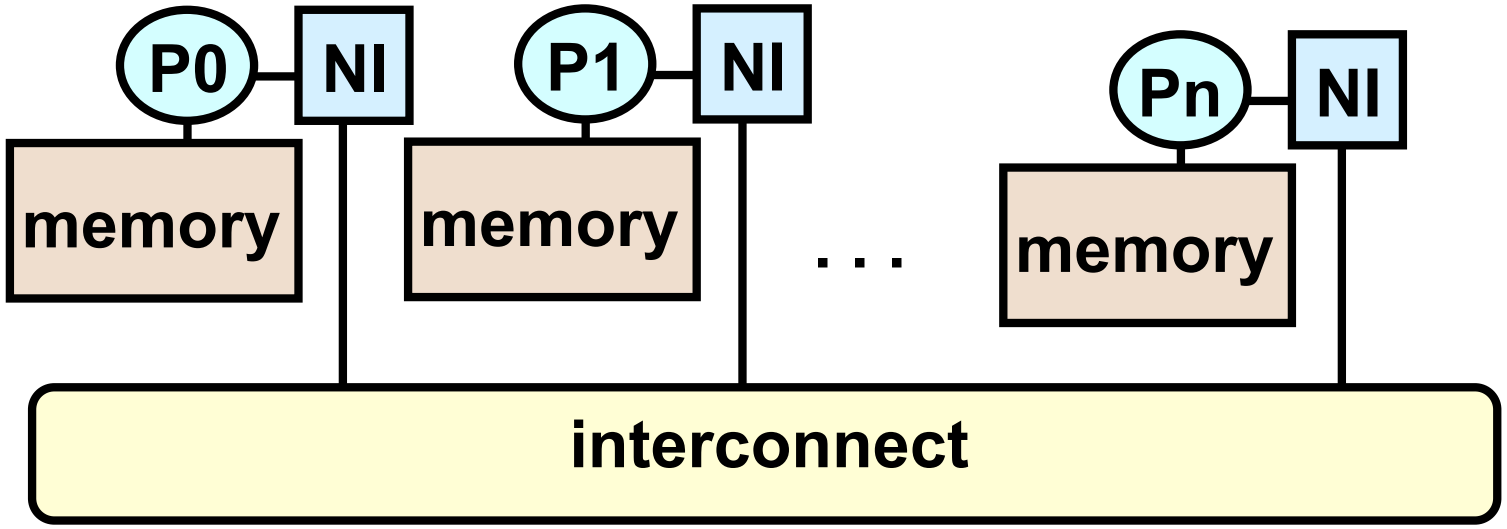
Programming Model 1b: Multithreaded Processor

- Multiple thread “contexts” without full processors
- Memory and some other state is shared
- Sun Niagra processor (for servers)
  - Up to 64 threads all running simultaneously (8 threads x 8 cores)
  - In addition to sharing memory, they share floating point units
  - Why? Switch between threads for long-latency memory operations
- Cray MTA and Eldorado processors (for HPC)


T0
T1
Tn
arithmetic logic unit/bus
shared $, shared floating point units, etc.
Memory

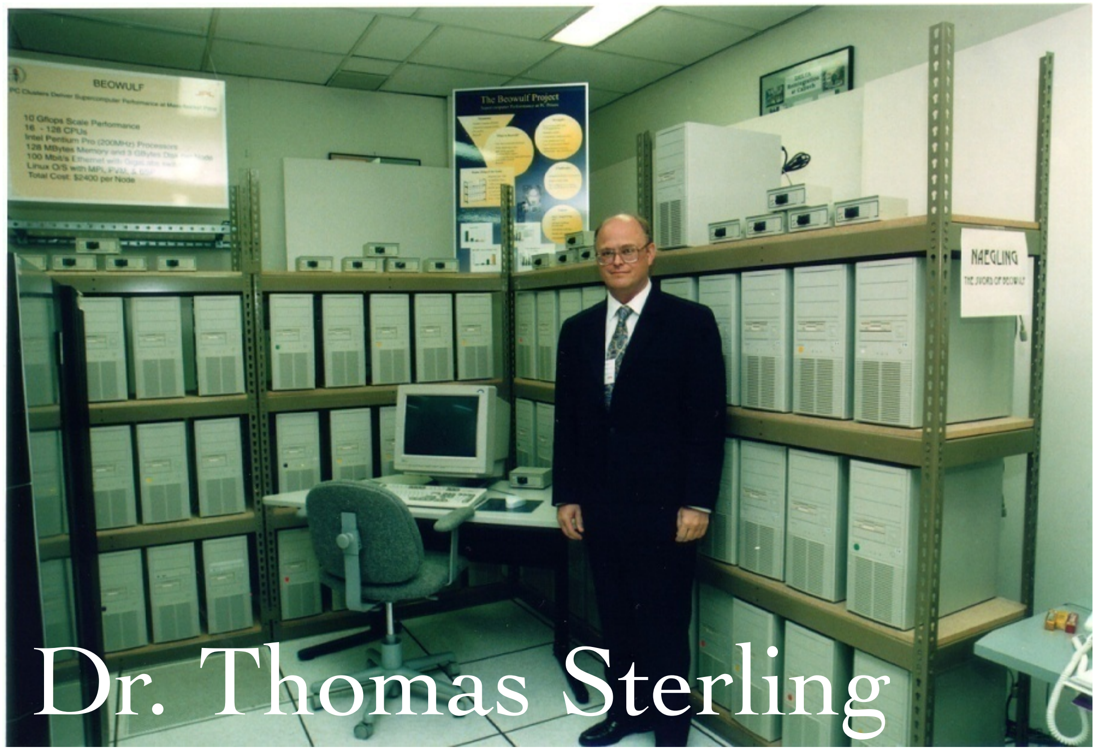
Programming Model 2a: Distributed Memory

- Cray XE6 (Hopper), Cray XC30 (Edison)
- PC Clusters (Berkeley NOW, Beowulf)
- Edison, Hopper, most of the Top500, are distributed memory machines, but the nodes are SMPs.
- Each processor has its own memory and cache but cannot directly access another processor’s memory.
- Each “node” has a Network Interface (NI) for all communication and synchronization.



PC clusters — Contributions of Beowulf

- An experiment in parallel computing systems (1994)
- Established vision of low cost, high end computing
  - Cost effective because it uses off-the-shelf parts
- Demonstrated effectiveness of PC clusters for some (not all) classes of applications
- Provided networking software
- Conveyed findings to broad community (great PR)
- Tutorials and book
- Design standard to rally community!
- Standards beget: books, trained people, software … virtuous cycle

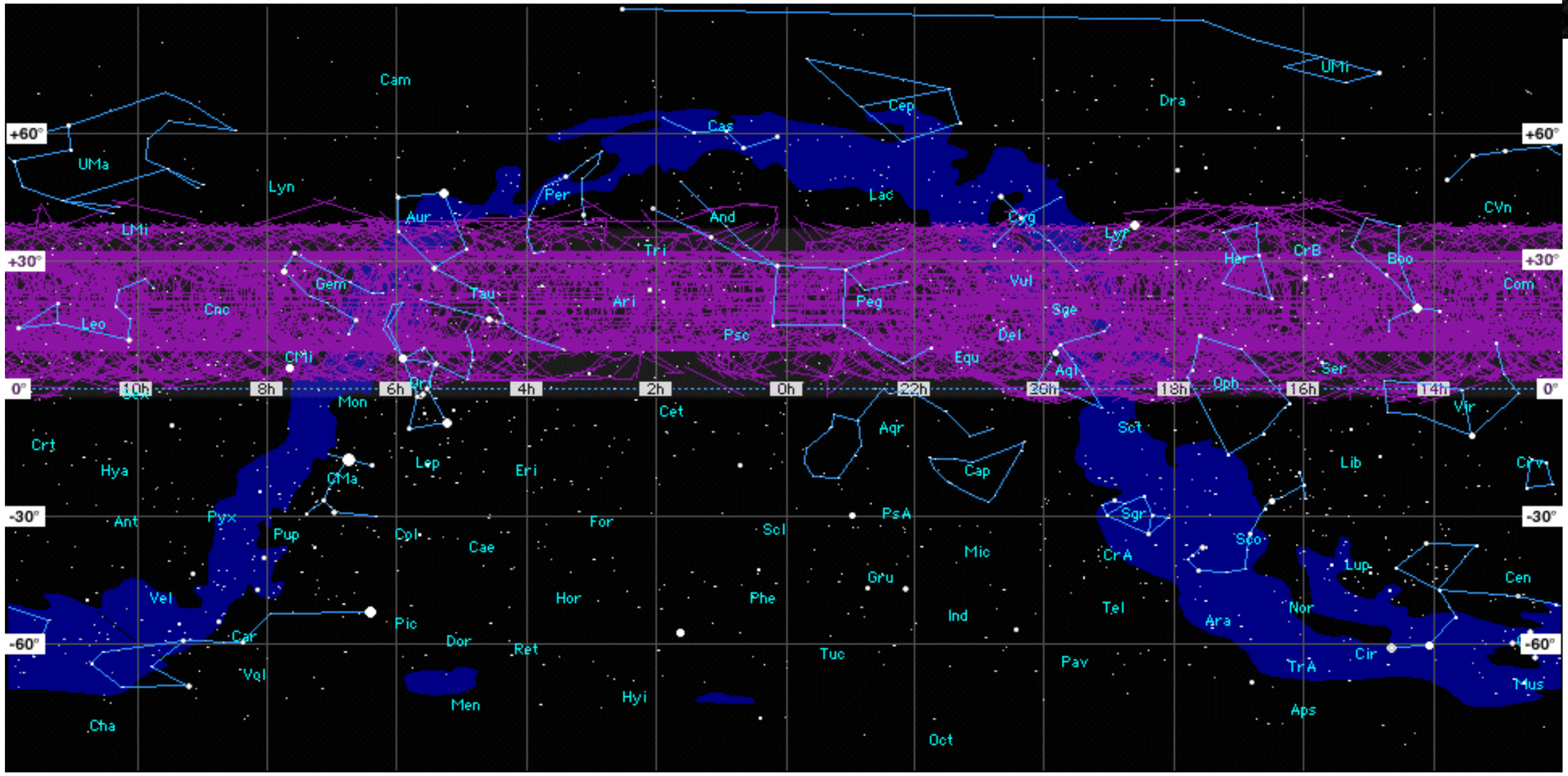


Dr. Thomas Sterling

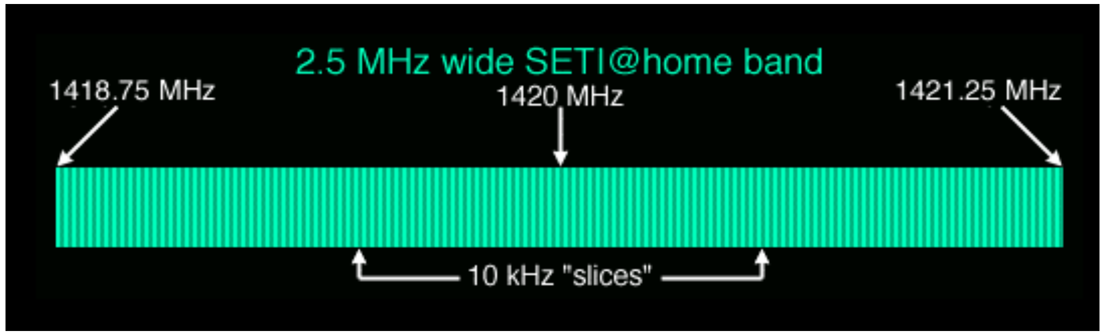
Adapted from Gordon Bell, presentation at Salishan

Programming Model 2a: Internet/Grid Computing

- SETI@Home: Running on 3.3M hosts, 1.3M users (1/2013)
  - ~1000 CPU Years per Day (older data)
  - 485,821 CPU Years so far
- Sophisticated Data & Signal Processing Analysis
- Distributes Datasets from Arecibo Radio Telescope



Next Step-
Allen Telescope Array



Google
"volunteer computing"
or "BOINC"

Introduction to OpenMP

- What is OpenMP?
  - Open specification for Multi-Processing, latest version 4.0, July 2013
  - "Standard" API for defining multi-threaded shared-memory programs
  - openmp.org – Talks, examples, forums, etc.
  - computing.llnl.gov/tutorials/openMP/
  - portal.xsede.org/online-training
  - www.nersc.gov/assets/Uploads/XE62011OpenMP.pdf

- High-level API (application programming interface)
  - Preprocessor (compiler) directives (~ 80%)
  - Library Calls (~ 19%)
  - Environment Variables (~ 1%)

A Programmer's View of OpenMP

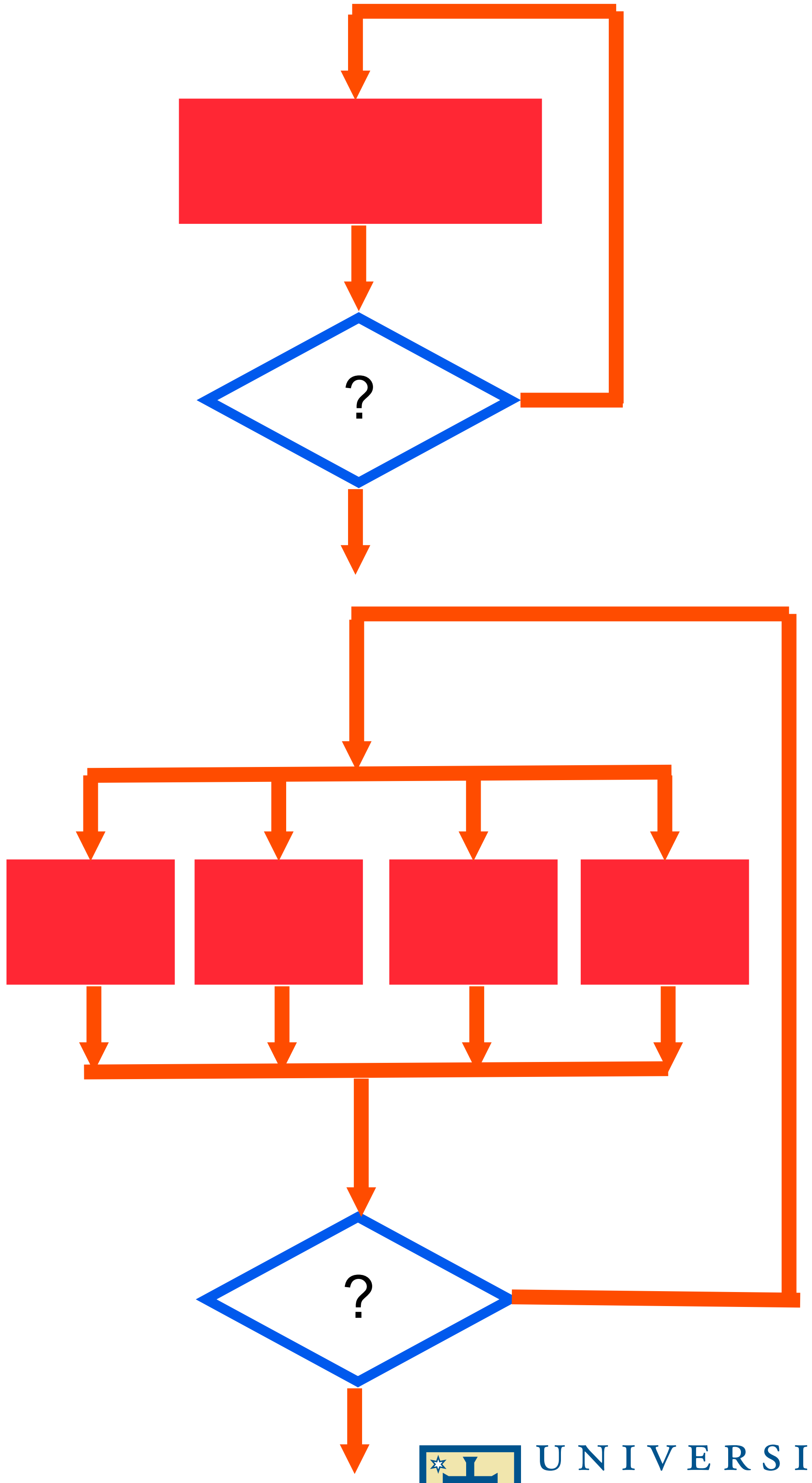
- OpenMP is a portable, threaded, shared-memory programming *specification* with "light" syntax
  - Exact behavior depends on OpenMP *implementation*!
  - Requires compiler support (C, C++ or Fortran)

- OpenMP will:
  - Allow a programmer to separate a program into *serial regions* and *parallel regions,* rather than T concurrently-executing threads*.*
  - Hide stack management
  - Provide synchronization constructs

- OpenMP will not:
  - Parallelize automatically
  - Guarantee speedup
  - Provide freedom from data races (concurrent memory access)

Programming Model – Concurrent Loops

- OpenMP easily parallelizes loops
    - Requires: No data dependencies (reads/write or write/write pairs) between iterations!
- Preprocessor calculates loop bounds for each thread directly from *serial* source



```
#pragma omp parallel for

 for( i=0; i < 25; i++ )
 {

   printf(“Foo”);

 }
```

OpenMP Summary

- OpenMP is a compiler-based technique to create concurrent code from (mostly) serial code
- OpenMP can enable (easy) parallelization of loop-based code
  - Lightweight syntactic language extensions
- OpenMP performs comparably to manually-coded threading
  - Scalable
  - Portable
- Not a silver bullet for all (more irregular) applications
- Lots of detailed tutorials/manuals on-line

Distributed Memory Machines and Programming
Network Analogy

- To have a large number of different transfers occurring at once, you need a large number of distinct wires
  - Not just a bus, as in shared memory
- Networks are like streets:
  - Link = street.
  - Switch = intersection.
  - Distances (hops) = number of blocks traveled.
  - Routing algorithm = travel plan.
- Properties:
  - Latency: how long to get between nodes in the network.
    - Street: time for one car = dist (miles) / speed (miles/hr)
  - Bandwidth: how much data can be moved per unit time.
    - Street: cars/hour = density (cars/mile) * speed (miles/hr) * #lanes
    - Network bandwidth is limited by the bit rate per wire and #wires

Message Passing Interface — MPI

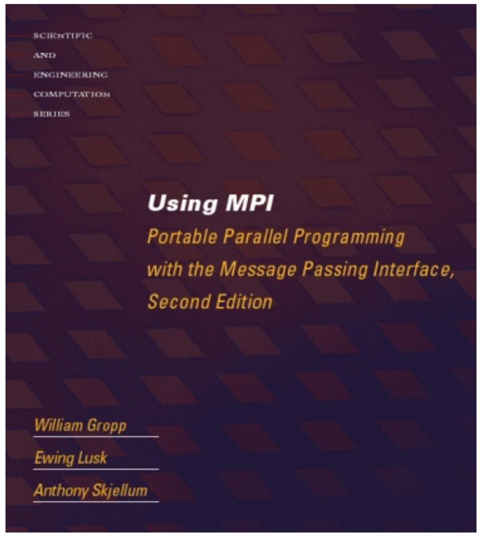
- All communication, synchronization require subroutine calls
  - No shared variables
  - Program run on a single processor just like any uniprocessor program, except for calls to message passing library
- Subroutines for
  - Communication
    - Pairwise or point-to-point: Send and Receive
    - Collectives all processor get together to
      - Move data: Broadcast, Scatter/gather
      - Compute and move: sum, product, max, prefix sum, … of data on many processors
  - Synchronization
    - Barrier
    - No locks because there are no shared variables to protect
  - Enquiries
    - How many processes? Which one am I? Any messages waiting?

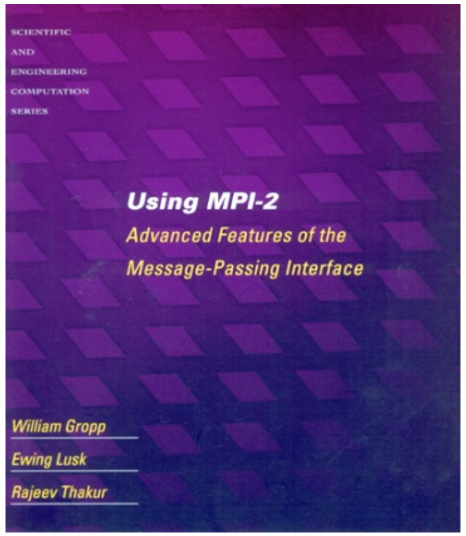
Message Passing Interface — MPI

- The Standard itself:
  - at http://www.mpi-forum.org
  - All MPI official releases, in both postscript and HTML
  - Latest version MPI 3.1, released June 2015

- Other information on Web:
  - at http://www.mcs.anl.gov/mpi
  - pointers to lots of stuff, including other talks and tutorials, a FAQ, other MPI pages

Message Passing Interface — MPI





- *Using MPI: Portable Parallel Programming with the Message-Passing Interface (2nd edition)*, by Gropp, Lusk, and Skjellum, MIT Press, 1999.
- *Using MPI-2: Portable Parallel Programming with the Message-Passing Interface*, by Gropp, Lusk, and Thakur, MIT Press, 1999.
- *MPI: The Complete Reference - Vol 1 The MPI Core,* by Snir, Otto, Huss-Lederman, Walker, and Dongarra, MIT Press, 1998.
- *MPI: The Complete Reference - Vol 2 The MPI Extensions*, by Gropp, Huss-Lederman, Lumsdaine, Lusk, Nitzberg, Saphir, and Snir, MIT Press, 1998.
- *Designing and Building Parallel Programs*, by Ian Foster, Addison-Wesley, 1995.
- *Parallel Programming with MPI*, by Peter Pacheco, Morgan-Kaufmann, 1997.

CS267 Lectures & Bill Gropp, UIUC

Hello World (C)

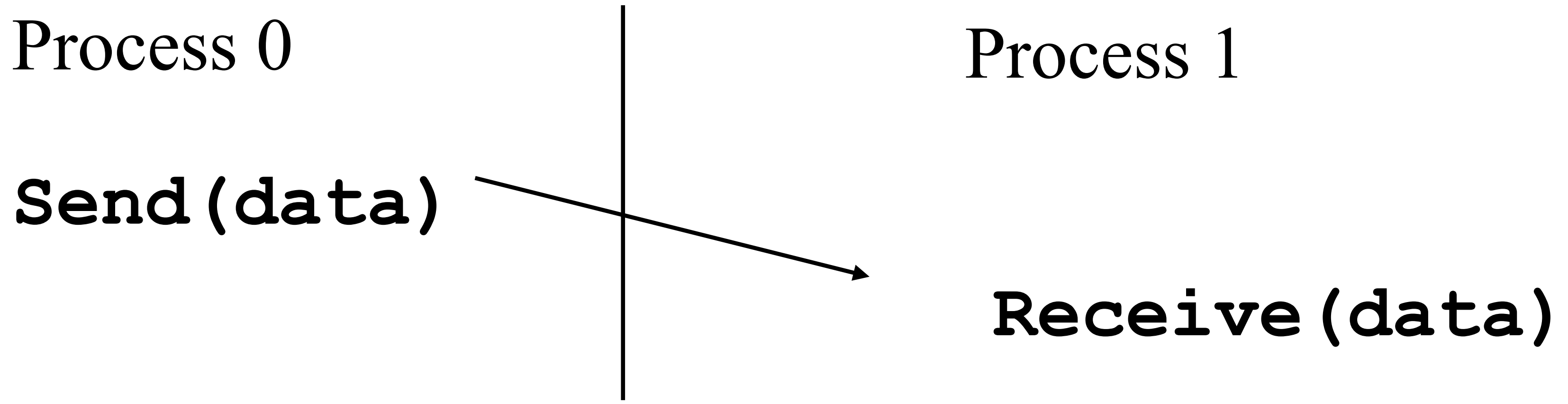
```
#include "mpi.h"
#include <stdio.h>

int main( int argc, char *argv[] )
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    printf( "I am %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

```
mpirun –np 4 a.out
```

CS267 Lectures & Bill Gropp, UIUC

MPI Basic Send/Receive

- We need to fill in the details in



Process 0

`Send(data)`

Process 1

`Receive(data)`

- Things that need specifying:
  - How will “data” be described?
  - How will processes be identified?
  - How will the receiver recognize/screen messages?
  - What will it mean for these operations to complete?

Parallel Environment

- Two important questions that arise early in a parallel program are:
    - How many processes are participating in this computation?
    - Which one am I?

- MPI provides functions to answer these questions:
    - **`MPI_Comm_size`** reports the number of processes.
    - **`MPI_Comm_rank`** reports the *rank*, a number between 0 and size-1, identifying the calling process

Some Basic Concepts

- Processes can be collected into groups
- Each message is sent in a context, and must be received in the same context
  - Provides necessary support for libraries
- A group and context together form a communicator
- A process is identified by its rank in the group associated with a communicator
- There is a default communicator whose group contains all initial processes, called **`MPI_COMM_WORLD`**

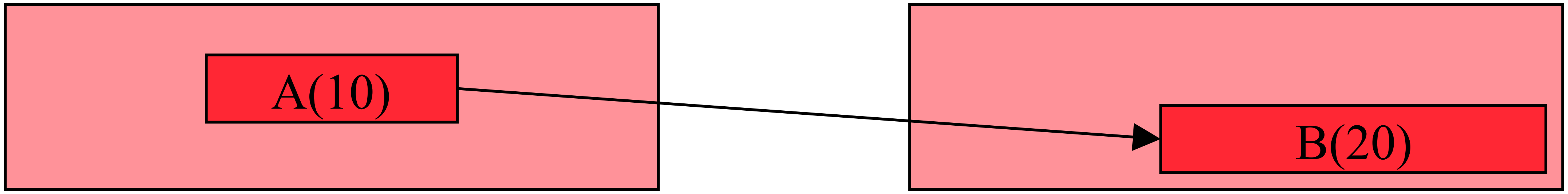
MPI Datatypes

- The data in a message to send or receive is described by a triple (address, count, datatype), where
- An MPI datatype is recursively defined as:
  - predefined, corresponding to a data type from the language (e.g., MPI_INT, MPI_DOUBLE)
  - a contiguous array of MPI datatypes
  - a strided block of datatypes
  - an indexed array of blocks of datatypes
  - an arbitrary structure of datatypes
- There are MPI functions to construct custom datatypes, in particular ones for subarrays
- May hurt performance if datatypes are complex

MPI Tags

- Messages are sent with an accompanying user-defined integer tag, to assist the receiving process in identifying the message
- Messages can be screened at the receiving end by specifying a specific tag, or not screened by specifying MPI_ANY_TAG as the tag in a receive
- Some non-MPI message-passing systems have called tags "message types". MPI calls them tags to avoid confusion with datatypes

MPI Basic (Blocking) Send



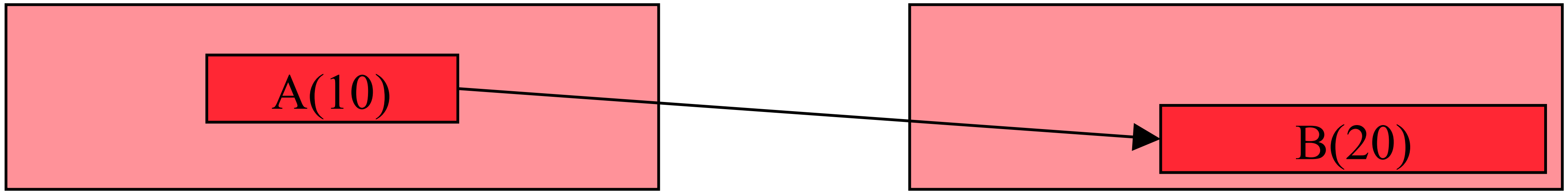
MPI_Send(A, 10, MPI_DOUBLE, 1, …)

MPI_Recv(B, 20, MPI_DOUBLE, 0, …)

`MPI_SEND(start, count, datatype, dest, tag, comm)`

- The message buffer is described by (**`start, count, datatype`**).
- The target process is specified by **`dest`**, which is the rank of the target process in the communicator specified by **`comm`**.
- When this function returns, the data has been delivered to the system and the buffer can be reused. The message may not have been received by the target process.

MPI Basic (Blocking) Receive



MPI_Send(A, 10, MPI_DOUBLE, 1, …)

MPI_Recv(B, 20, MPI_DOUBLE, 0, …)

`MPI_RECV(start, count, datatype, source, tag, comm, status)`

- Waits until a matching (both **`source`** and **`tag`**) message is received from the system, and the buffer can be used
- **`source`** is rank in communicator specified by **`comm`**, or **`MPI_ANY_SOURCE`**
- **`tag`** is a tag to be matched or **`MPI_ANY_TAG`**
- receiving fewer than **`count`** occurrences of **`datatype`** is OK, but receiving more is an error
- **`status`** contains further information (e.g. size of message)

A Simple MPI Program

- Many parallel programs can be written using just these *six functions*, only two of which are non-trivial:

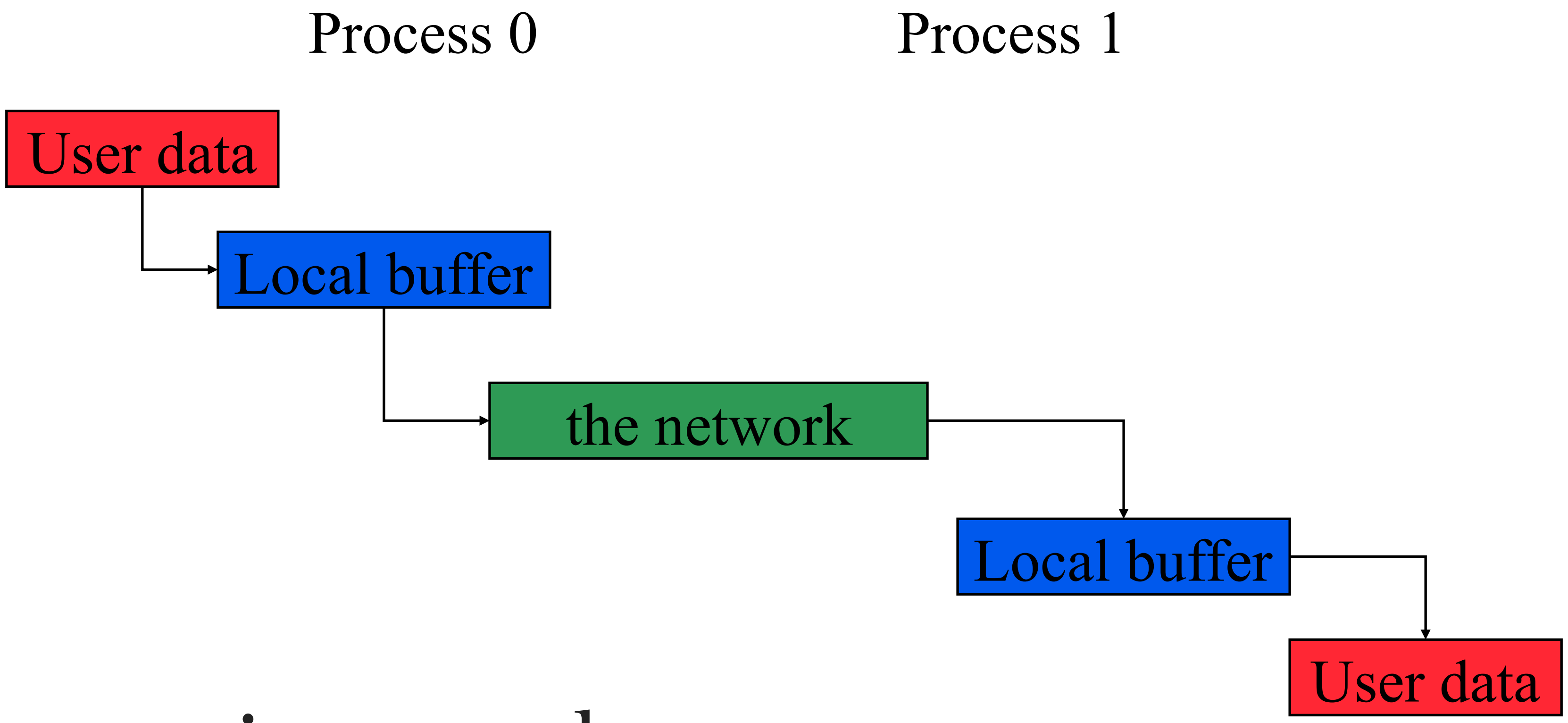
```
#include “mpi.h”
#include <stdio.h>
int main( int argc, char *argv[])
{
  int rank, buf;
  MPI_Status status;
  MPI_Init(&argv, &argc);
  MPI_Comm_rank( MPI_COMM_WORLD, &rank );

  /* Process 0 sends and Process 1 receives */
  if (rank == 0) {
    buf = 123456;
    MPI_Send( &buf, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
  }
  else if (rank == 1) {
    MPI_Recv( &buf, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
              &status );
    printf( “Received %d\n”, buf );
  }

  MPI_Finalize();
  return 0;
}
```

- **MPI_INIT**
- **MPI_FINALIZE**
- **MPI_COMM_SIZE**
- **MPI_COMM_RANK**
- **MPI_SEND**
- **MPI_RECV**

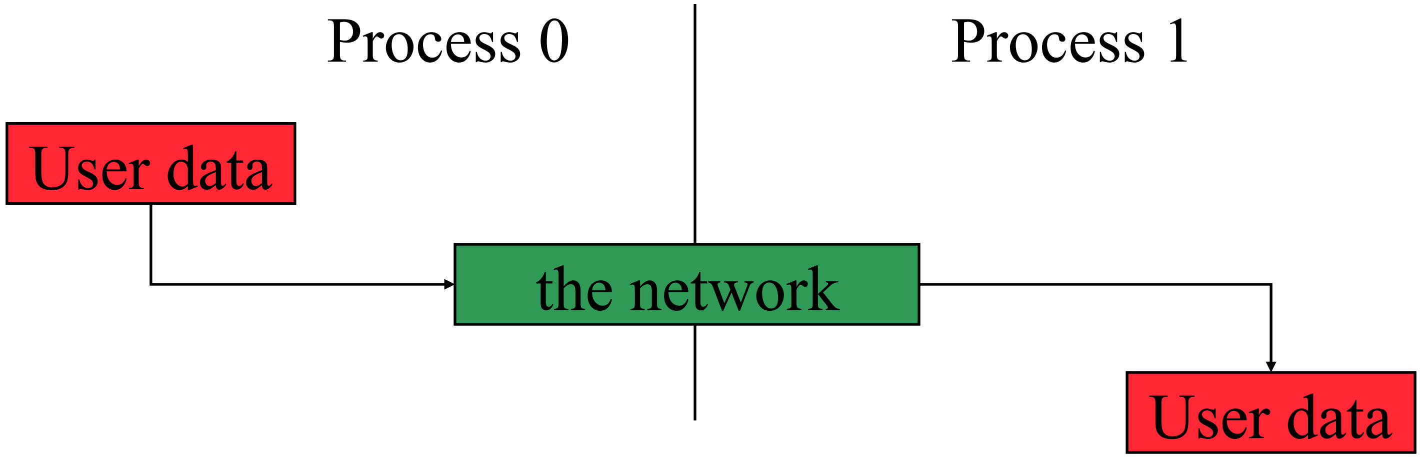
CS267 Lectures & Bill Gropp, UIUC

Buffers

- When you send data, where does it go?



- Avoiding copies uses less memory
- May use more or less time



Deadlocks

- Send a large message from process 0 to process 1
  - If there is insufficient storage at the destination, the send must wait for the user to provide the memory space (through a receive)
- What happens with this code?

Process 0	Process 1
`Send(1)`	**`Send(0)`**
`Recv(1)`	**`Recv(0)`**

- Order the operations more carefully

Process 0	Process 1
`Send(1)`	**`Recv(0)`**
`Recv(1)`	**`Send(0)`**

- This is called "unsafe" because it depends on the availability of system buffers in which to store the data sent until it can be received

CS267 Lectures & Bill Gropp, UIUC

MPI — Non-blocking Operations

- Non-blocking operations return (immediately) “request handles” that can be tested and waited on:

```
MPI_Request request;
MPI_Status status;
 MPI_Isend(start, count, datatype,
     dest, tag, comm, &request);
 MPI_Irecv(start, count, datatype,
     dest, tag, comm, &request);
 MPI_Wait(&request, &status);
```

(each request must be Waited on)

- One can also test without waiting:

```
MPI_Test(&request, &flag, &status);
```

- Accessing the data buffer without waiting is undefined

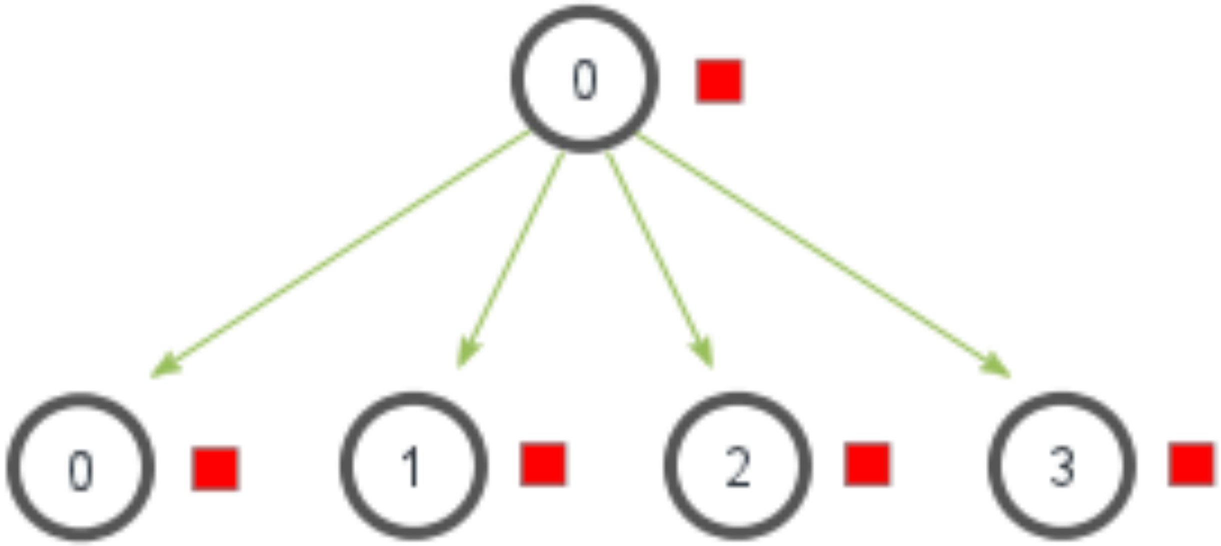
Collective Operations in MPI

- Collective operations are called by all processes in a communicator
- **`MPI_BCAST`** distributes data from one process (the root) to all others in a communicator
- **`MPI_REDUCE`** combines data from all processes in communicator and returns it to one process
- In many numerical algorithms, **`SEND/RECEIVE`** can be replaced by **`BCAST/REDUCE`**, improving both simplicity and efficiency

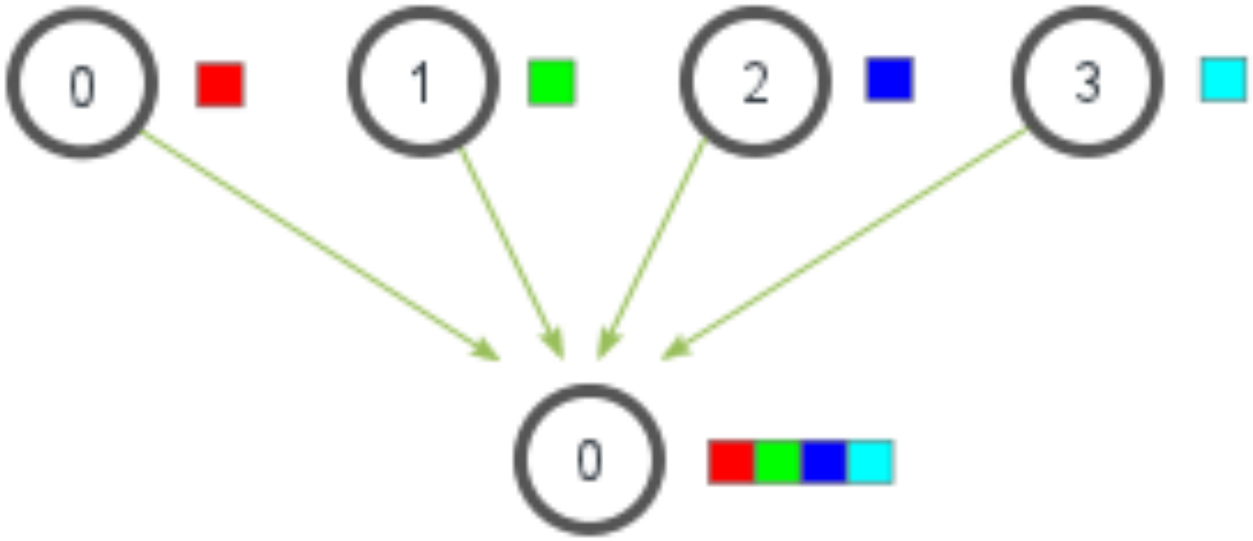
CS267 Lectures & Bill Gropp, UIUC

Collective Operations in MPI

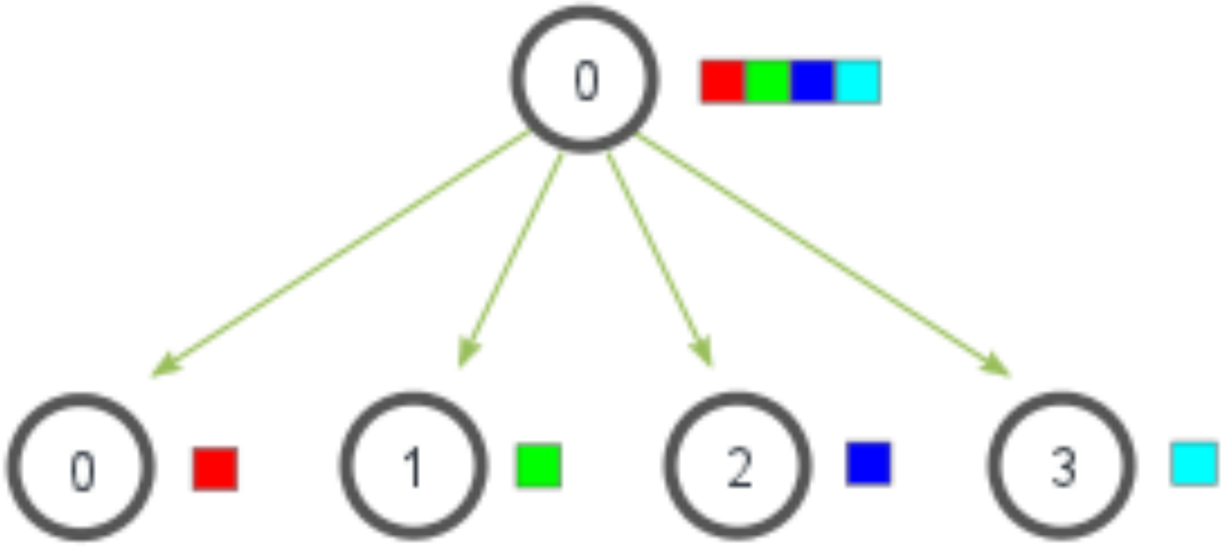
MPI_Bcast



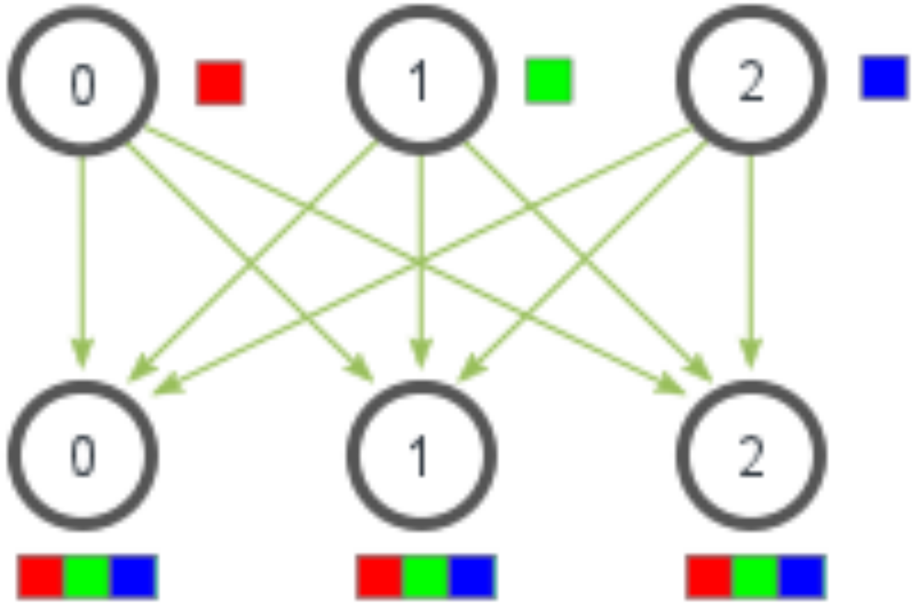
MPI_Gather



MPI_Scatter



MPI_Allgather



http://mpitutorial.com

MPI is Simple

- Claim: most MPI applications can be written with only 6 functions (although which 6 may differ)

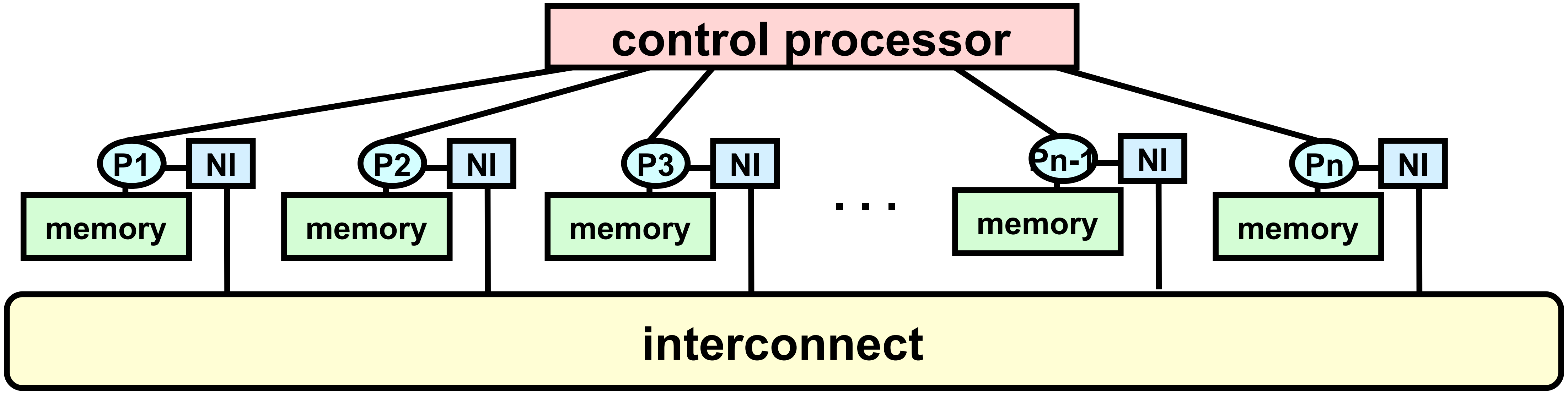
- Using point-to-point:
  - **MPI_INIT**
  - **MPI_FINALIZE**
  - **MPI_COMM_SIZE**
  - **MPI_COMM_RANK**
  - **MPI_SEND**
  - **MPI_RECEIVE**

- Using collectives:
  - **MPI_INIT**
  - **MPI_FINALIZE**
  - **MPI_COMM_SIZE**
  - **MPI_COMM_RANK**
  - **MPI_BCAST**
  - **MPI_REDUCE**

- You may use more for convenience or performance

Programming Model 3a: SIMD & GPU

- A large number of (usually) small processors.
  - **A single "control processor" issues each instruction.**
  - **Each processor executes the same instruction.**
  - **Some processors may be turned off on some instructions.**
- Originally machines were specialized to scientific computing, few made (CM2, Maspar)
- Programming model can be implemented in the compiler
  - **mapping n-fold parallelism to p processors, n >> p, but it's hard (e.g., HPF)**



Programming Model : GPU

- GPU's big performance opportunity is data parallelism
  - Most programs have a mixture of highly parallel operations, and some not so parallel
  - GPUs provide a threaded programming model (CUDA) for data parallelism to accommodate both
  - Current research attempting to generalize programming model to other architectures, for portability (OpenCL)
- CUDA is a programming model designed for
  - Heterogeneous architectures, Wide SIMD parallelism, Scalability
- CUDA provides
  - Synchronization & data sharing between small thread groups A thread abstraction to deal with SIMD

Programming Model 4: Hybrid machines

- Multicore/SMPs are a building block for a larger machine with a network
- Old name:
  - CLUMP = Cluster of SMPs
- Many modern machines look like this:
  - Edison and Hopper (2x12 way nodes), most of Top500
- What is an appropriate programming model #4 ???
  - Treat machine as "flat", always use message passing, even within SMP (simple, but ignores an important part of memory hierarchy).
  - Shared memory within one SMP, but message passing outside of an SMP.
- GPUs may also be building block
  - Nov 2014 Top500: 14% have accelerators, but 35% of performance

Programming Model : Hybrids

- Programming models can be mixed
  - Message passing (MPI) at the top level with shared memory within a node is common
  - New DARPA HPCS languages mix data parallel and threads in a global address space
  - Global address space models can (often) call message passing libraries or vice verse
  - Global address space models can be used in a hybrid mode
    - Shared memory when it exists in hardware
    - Communication (done by the runtime system) otherwise
- For better or worse
  - Supercomputers often programmed this way for peak performance

Kokkos —Dr. H. Carter Edwards
Sandia National Laboratories

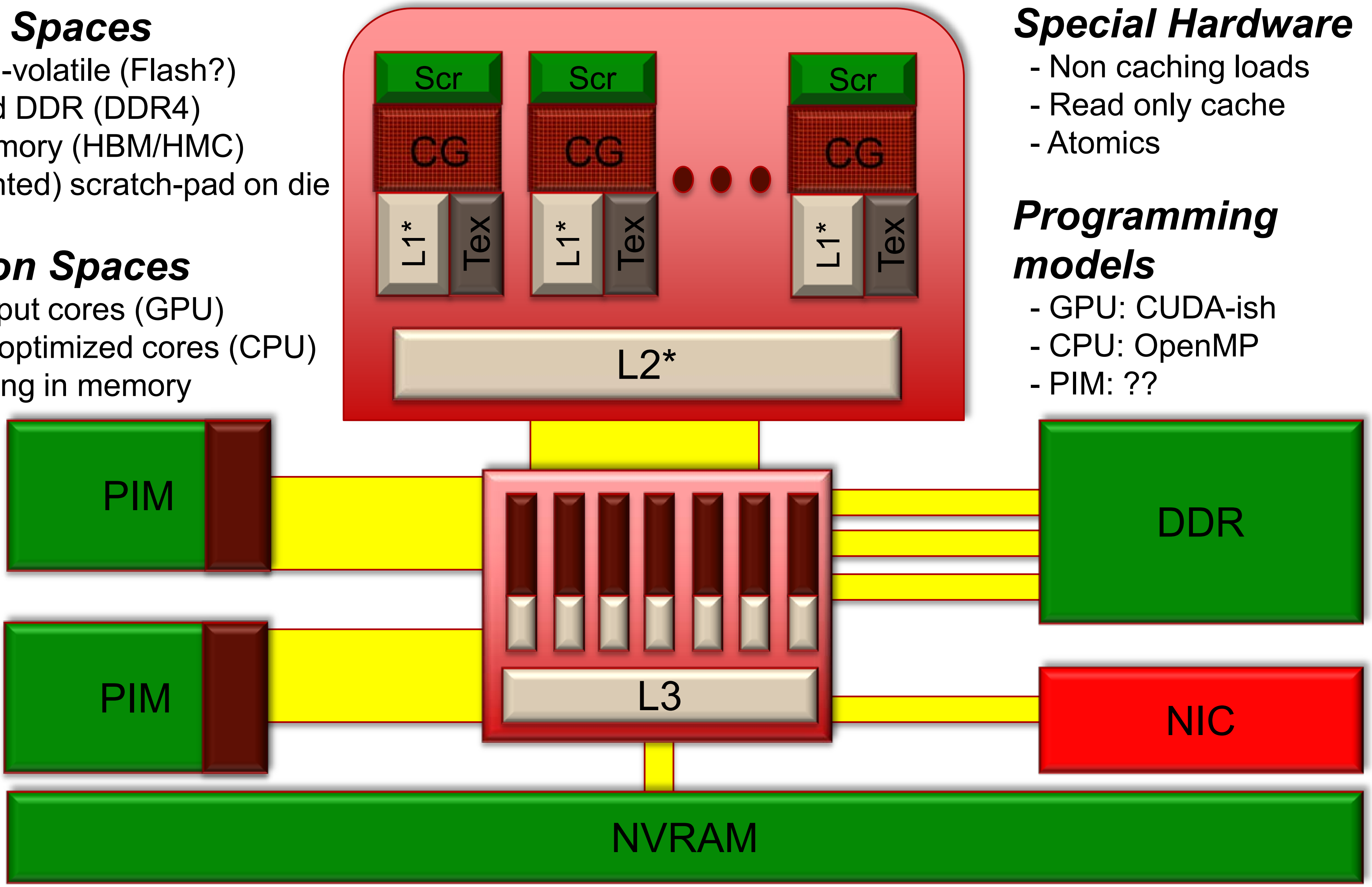
Increasingly Complex Heterogeneous Future

Memory Spaces

- Bulk non-volatile (Flash?)
- Standard DDR (DDR4)
- Fast memory (HBM/HMC)
- (Segmented) scratch-pad on die

Execution Spaces

- Throughput cores (GPU)
- Latency optimized cores (CPU)
- Processing in memory



Special Hardware

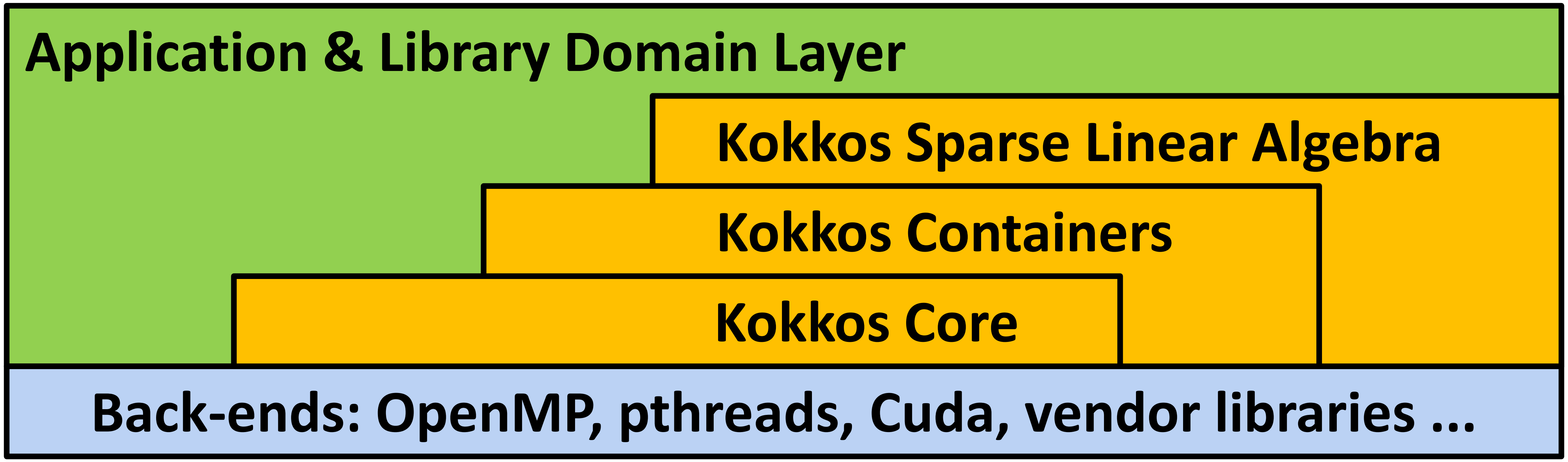
- Non caching loads
- Read only cache
- Atomics

Programming models

- GPU: CUDA-ish
- CPU: OpenMP
- PIM: ??

Kokkos — A Layered Collection of Libraries

- **Standard C++, Not a language extension**
  - In *spirit* of Intel's TBB, NVIDIA's Thrust & CUSP, MS C++AMP, ...
  - *Not* a language extension: OpenMP, OpenACC, OpenCL, CUDA
- **Uses C++ template meta-programming**
  - Currently rely upon C++1998 standard (everywhere except IBM's xlC)
  - Prefer to require C++2011 for lambda syntax
    - Need CUDA with C++2011 language compliance



Application & Library Domain Layer

Kokkos Sparse Linear Algebra

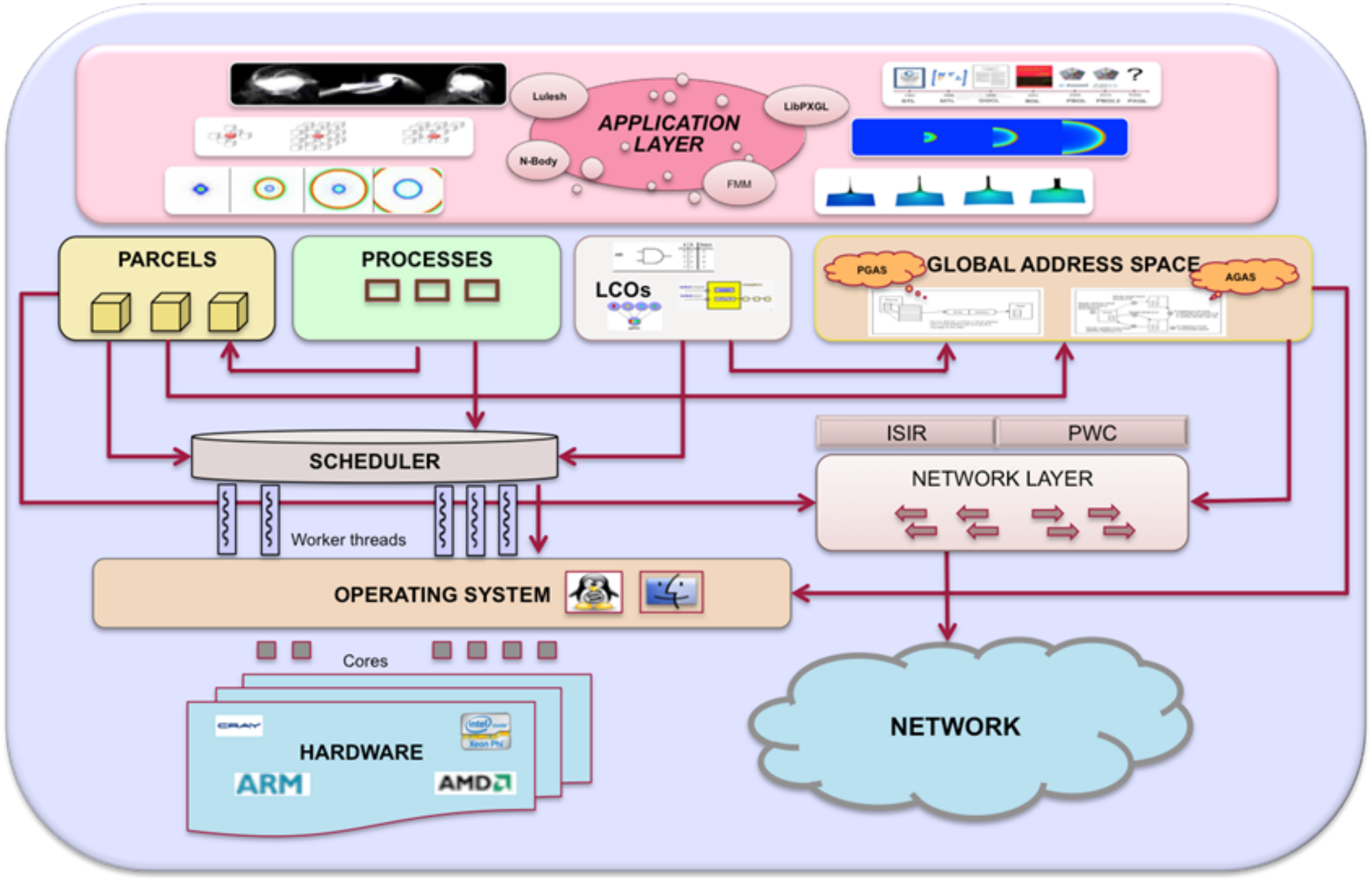
Kokkos Containers

Kokkos Core

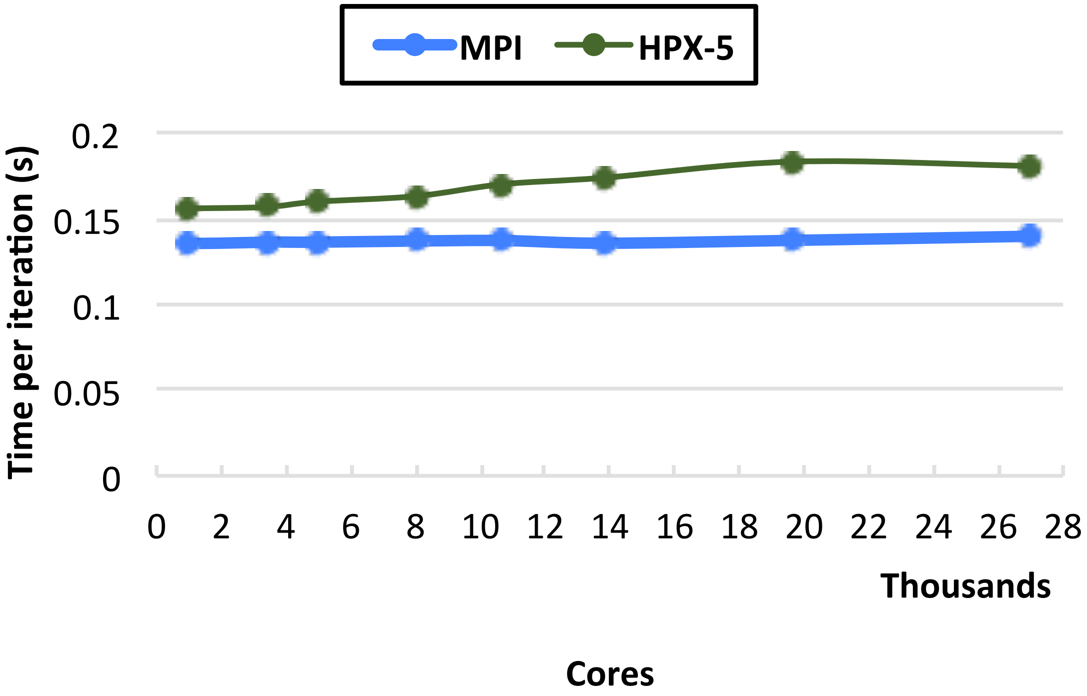
Back-ends: OpenMP, pthreads, Cuda, vendor libraries ...

High Performance ParalleX - HPX

HPX-5 v2.1.0



HPX-5 LULESH

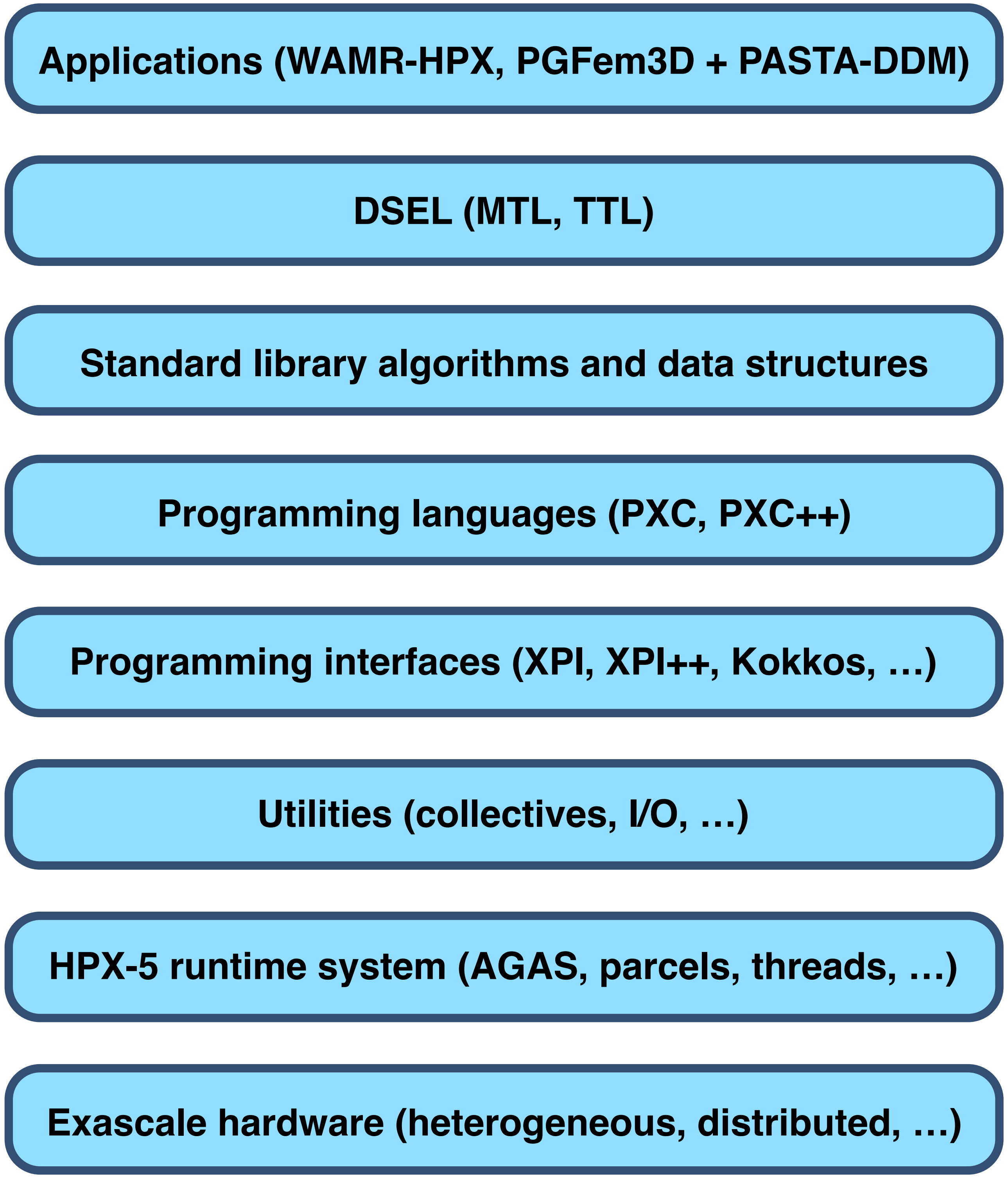


Cray XC30 weak scaling

- The HPX runtime system reifies the ParalleX execution model to support large-scale irregular applications:
  - Localities, ParalleX Processes, Complexes (ParalleX Threads and Thread Management)
  - Parcel Transport and Parcel Management (simple parcel continuations)
  - Local Control Objects (LCOs), Networking (MPI ISIR and PWC)

C-SWARM Framework

- C-SWARM codes are written in a DSEL that sits atop a multi-tiered software stack
- Progressive abstraction of applications away from the runtime system
- Ultimately, everything runs on HPX
- Unified by ParalleX execution model



Applications (WAMR-HPX, PGFem3D + PASTA-DDM)

DSEL (MTL, TTL)

Standard library algorithms and data structures

Programming languages (PXC, PXC++)

Programming interfaces (XPI, XPI++, Kokkos, ...)

Utilities (collectives, I/O, ...)

HPX-5 runtime system (AGAS, parcels, threads, ...)

Exascale hardware (heterogeneous, distributed, ...)

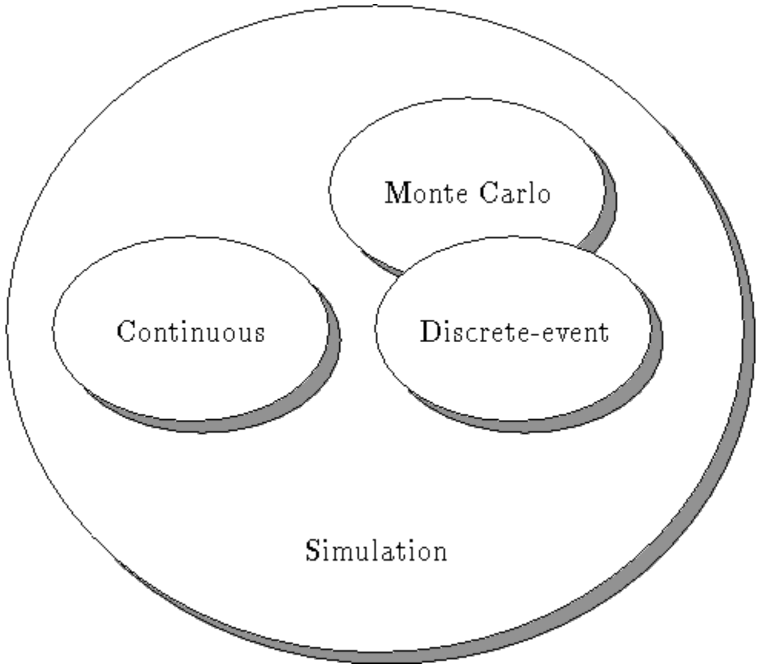
Numerical Parallel Algorithms
Parallelism and Locality in Simulation

- Parallelism and data locality both critical to performance
  - Recall that moving data is the most expensive operation
- Real world problems have parallelism and locality:
  - Many objects operate independently of others.
  - Objects often depend much more on nearby than distant objects.
  - Dependence on distant objects can often be simplified.
    - Example of all three: particles moving under gravity
- Scientific models may introduce more parallelism:
  - When a continuous problem is discretized, time dependencies are generally limited to adjacent time steps.
    - Helps limit dependence to nearby objects (eg collisions)
  - Far-field effects may be ignored or approximated in many cases.
- Many problems exhibit parallelism at multiple levels

Parallelism and Locality in Simulation

- Types of simulations
  - Discrete Event Systems
  - Particle Systems
  - Ordinary Differential Equations (ODEs)
  - Partial Differential Equations (PDEs)
- Common problems:
  - Load balancing
    - May be due to lack of parallelism or poor work distribution
    - Statically, divide grid (or graph) into blocks
    - Dynamically, if load changes significantly during run
  - Locality
    - Partition into large chunks with low surface-to-volume ratio
      - To minimize communication
    - Distributed particles according to location, but use irregular spatial decomposition (e.g., quad tree) for load balance
  - Constant tension between these two
    - Particle-Mesh method: can't balance particles (moving), balance mesh (fixed) and keep particles near mesh points without communication

Discrete Event Systems



- Systems are represented as:
  - finite set of variables.
  - the set of all variable values at a given time is called the state.
  - each variable is updated by computing a transition function depending on the other variables.

- System may be:
  - synchronous: at each discrete timestep evaluate all transition functions; also called a state machine.
  - asynchronous: transition functions are evaluated only if the inputs change, based on an "event" from another part of the system; also called event driven simulation.

Statistical Micromechanics

indicator function: $\chi_r\left(\mathbf{x};\alpha\right) = \left\{ \begin{array}{ll} 1 & \text{if } \mathbf{x} \text{ is in } r \\ 0 & \text{otherwise} \end{array} \right\}$

ensemble average: $\overline{\chi_r\left(\mathbf{x}\right)} = \int_{\mathcal{E}} \chi_r\left(\mathbf{x};\alpha\right) p\left(\alpha\right) d\alpha$

$$S_{r_1 r_2 \ldots r_n}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\right) = \overline{\chi_1\left(\mathbf{x}_1\right) \chi_2\left(\mathbf{x}_2\right) \ldots \chi_{r_n}\left(\mathbf{x}_n\right)}$$

assuming ergodicity, statistical homogeneity

$$S_r(\mathbf{x}) = c_r$$

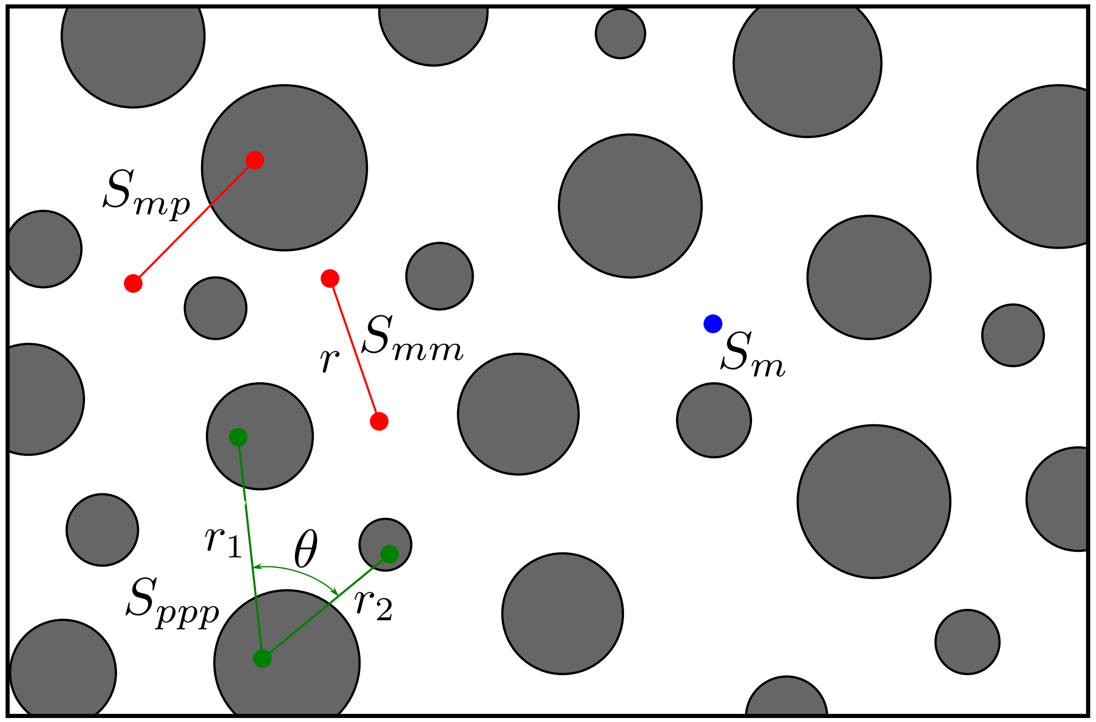
$$S_{rs}(\mathbf{x}, \mathbf{x}') = S_{rs}(\mathbf{x} - \mathbf{x}')$$

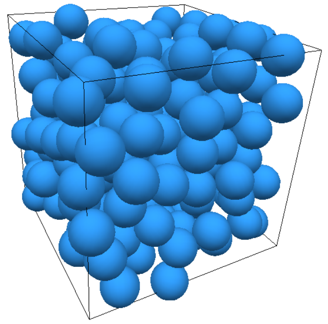
$$S_{rsq}(\mathbf{x}, \mathbf{x}') = S_{rsq}(\mathbf{x} - \mathbf{x}', \mathbf{x} - \mathbf{x}')$$

for statistically isotropic system

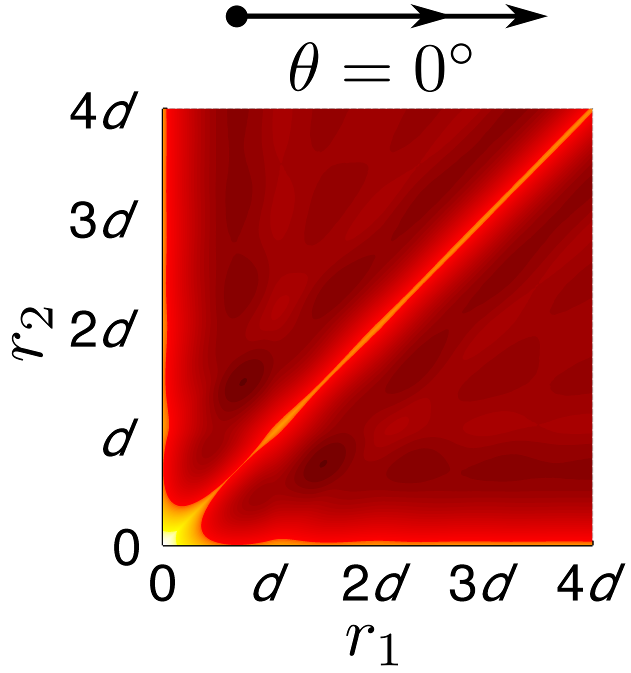
$$S_{rs}(\mathbf{x}, \mathbf{x}') = S_{rs}(|\mathbf{x} - \mathbf{x}'|)$$

$$S_{rsq}(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'') = S_{rsq}(|\boldsymbol{x} - \boldsymbol{x}'|, |\boldsymbol{x} - \boldsymbol{x}''|, \theta)$$

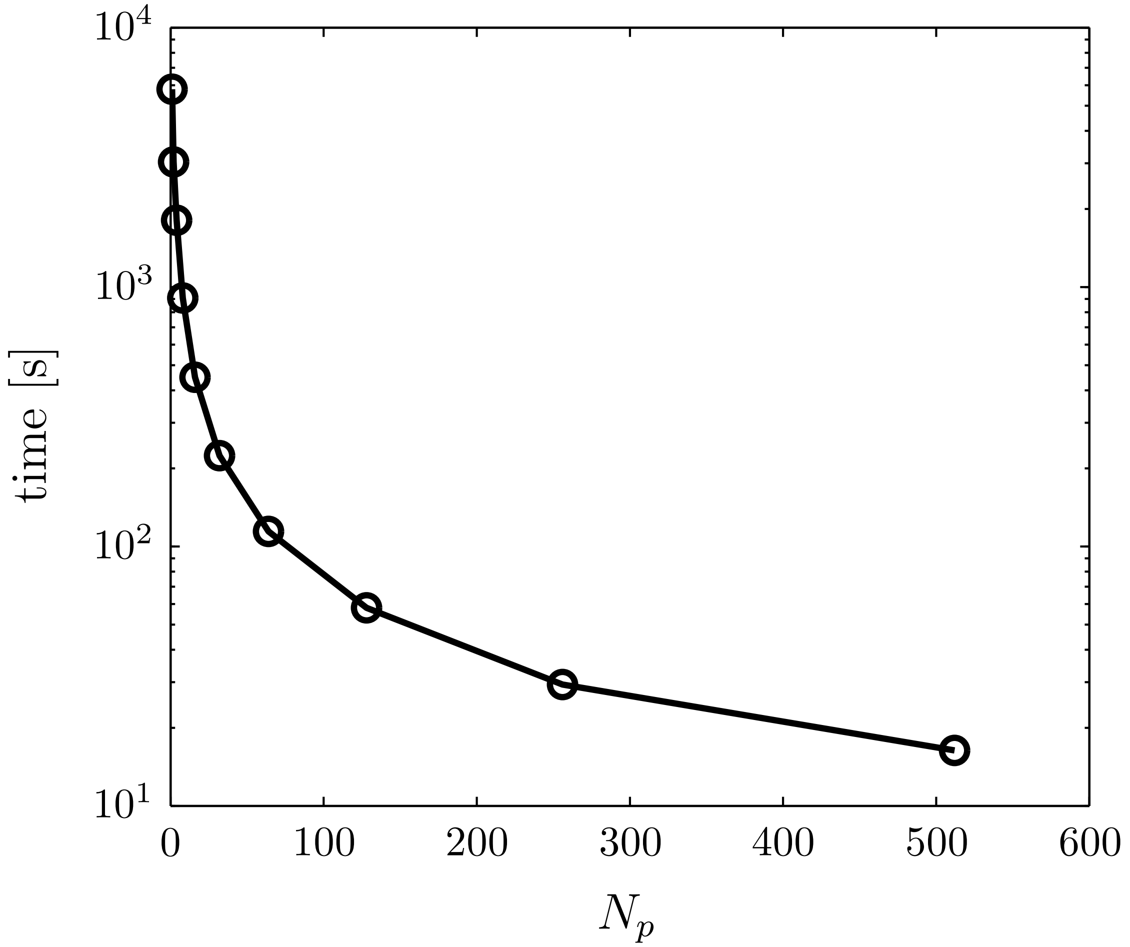


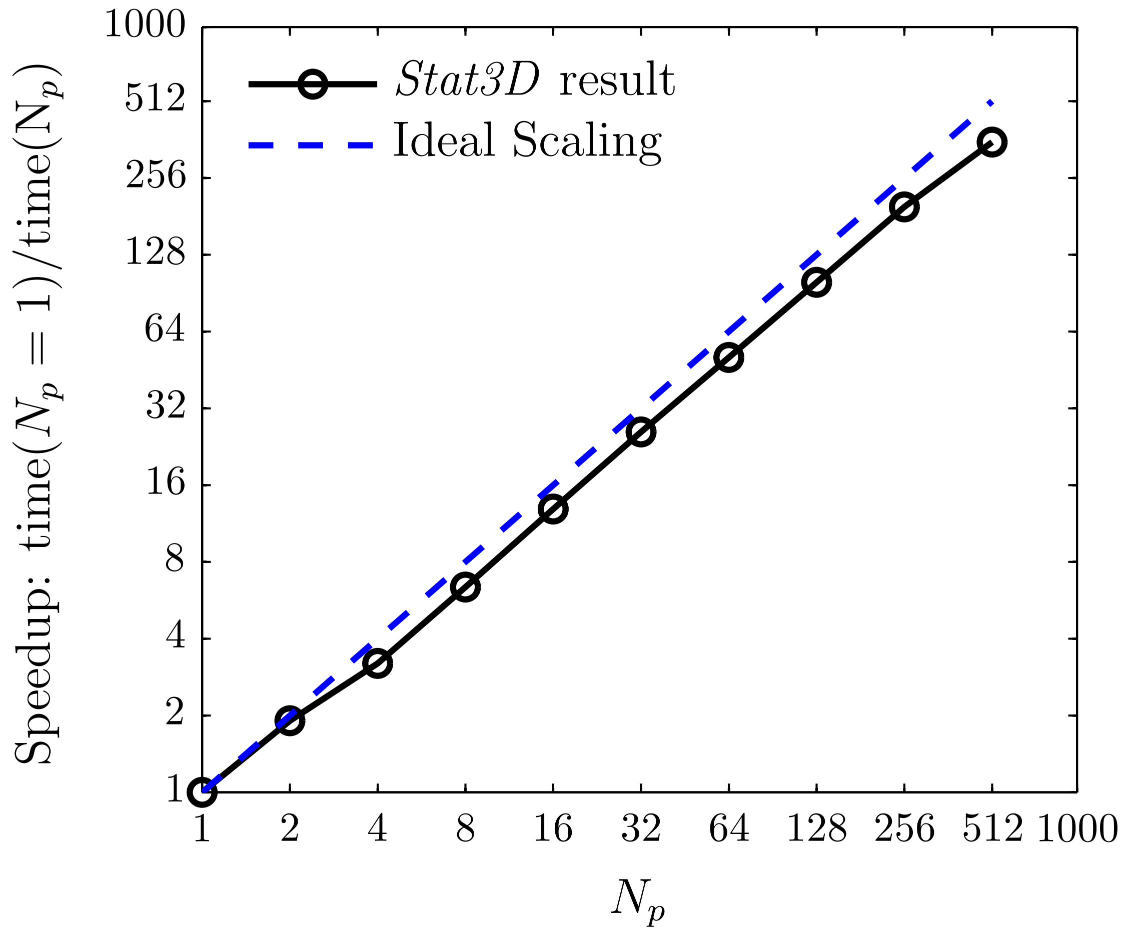


Spheres



Statistical Micromechanics — Stat3D





- Embarrassingly parallel
- Simple parallel domain decomposition, N/p
- Efficient three-based search algorithm

Statistical Micromechanics

Governing equations

$$\begin{aligned}\nabla \cdot \boldsymbol{q}(\boldsymbol{x}) &= 0 \quad \text{in} \quad \Omega, \\ T(x) &= Q_0 \cdot \boldsymbol{x} \quad \text{on} \quad \partial\Omega, \\ \boldsymbol{q}(\boldsymbol{x}) &= \boldsymbol{\kappa}(\boldsymbol{x}) \cdot \boldsymbol{Q}(\boldsymbol{x}) \\ \text{where } \boldsymbol{Q}(\boldsymbol{x}) &= -\nabla T(\boldsymbol{x}) \\ \boldsymbol{\kappa}(\boldsymbol{x}) &= \sum_{i=1}^{N} \boldsymbol{\kappa}_i \chi_i(\boldsymbol{x})\end{aligned}$$

- Third order bounds (Beran)

$$\kappa^L = c_p\kappa_p + c_m\kappa_m - \frac{c_m c_p(\kappa_p - \kappa_m)^2}{c_m\kappa_p + c_p\kappa_m + 2\left(\frac{\zeta_p}{\kappa_p} + \frac{\zeta_m}{\kappa_m}\right)^{-1}}$$

$$\kappa^U = c_p\kappa_p + c_m\kappa_m - \frac{c_m c_p(\kappa_p - \kappa_m)^2}{c_m\kappa_p + c_p\kappa_m + 2\left(\zeta_p\kappa_p + \zeta_m\kappa_m\right)}$$

- Third order approximation (Torquato)

$$\frac{\kappa_e}{\kappa_m} = \frac{1 + 2c_p\beta_{pm} - 2c_m\zeta_p\beta_{pm}^2}{1 - c_p\beta_{pm} - 2c_m\zeta_p\beta_{pm}^2}, \text{ where } \beta_{pm} = \frac{\kappa_p - \kappa_m}{\kappa_p + 2\kappa_m}$$

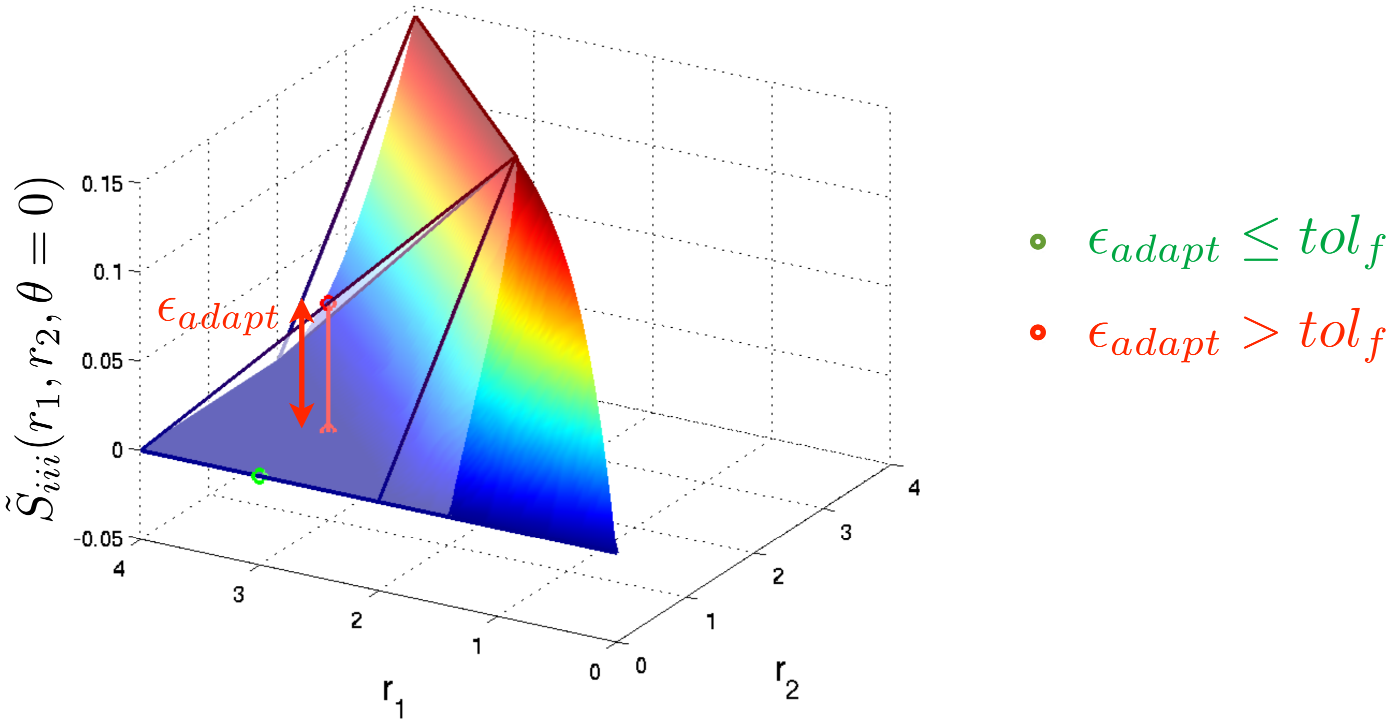
- Microstructural parameter involving three-point statistics

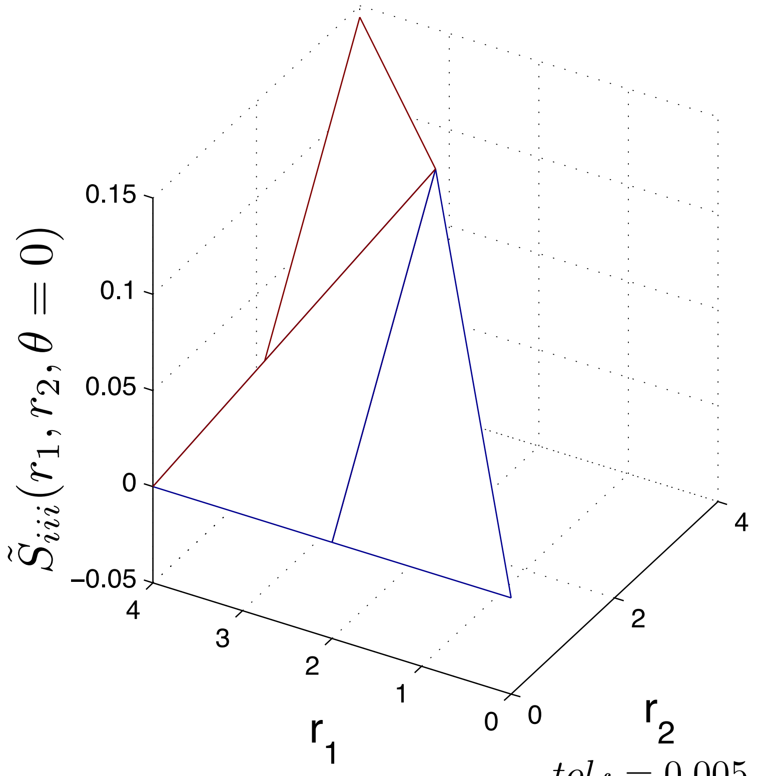
$$\zeta_i = \frac{9}{c_p c_m}\int_0^\infty\int_0^\infty\int_{-1}^{1} \frac{P_2(\cos\theta)}{r_1 r_2}\tilde{S}_{iii}(r_1, r_2, \theta)\mathrm{d}(\cos\theta)\mathrm{d}r_1\mathrm{d}r_2$$

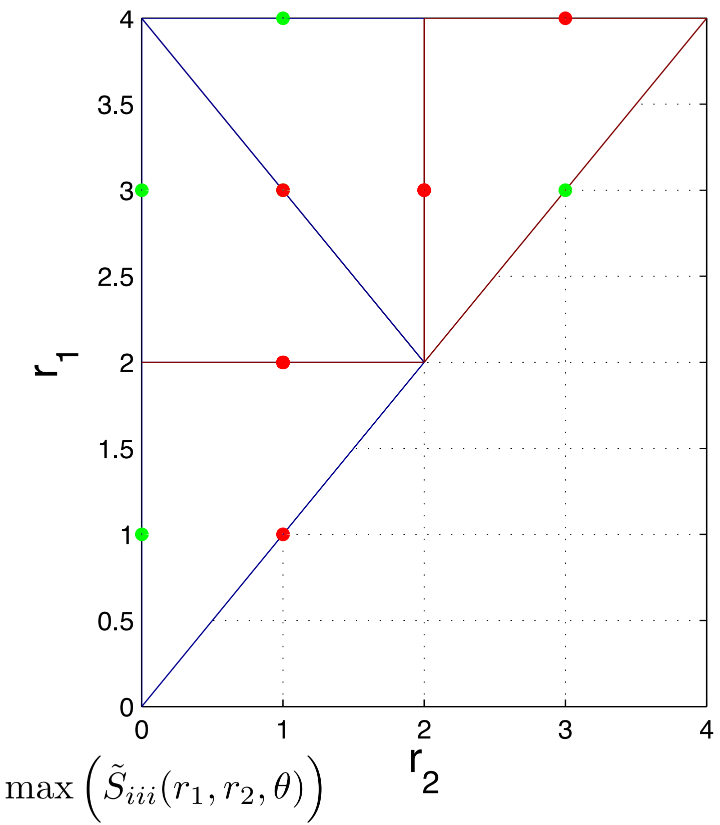
$$\tilde{S}_{iii}(r_1, r_2, \theta) = S_{iii}(r_1, r_2, \theta) - \frac{S_{ii}(r_1)S_{ii}(r_2)}{c_i}$$

Numerical Methods

Iteratively construct a Delaunay triangulation with local linear interpolation (C^0 continuity) to create an interpolant of $\tilde{S}_{iii}(r_1, r_2, \theta)$.







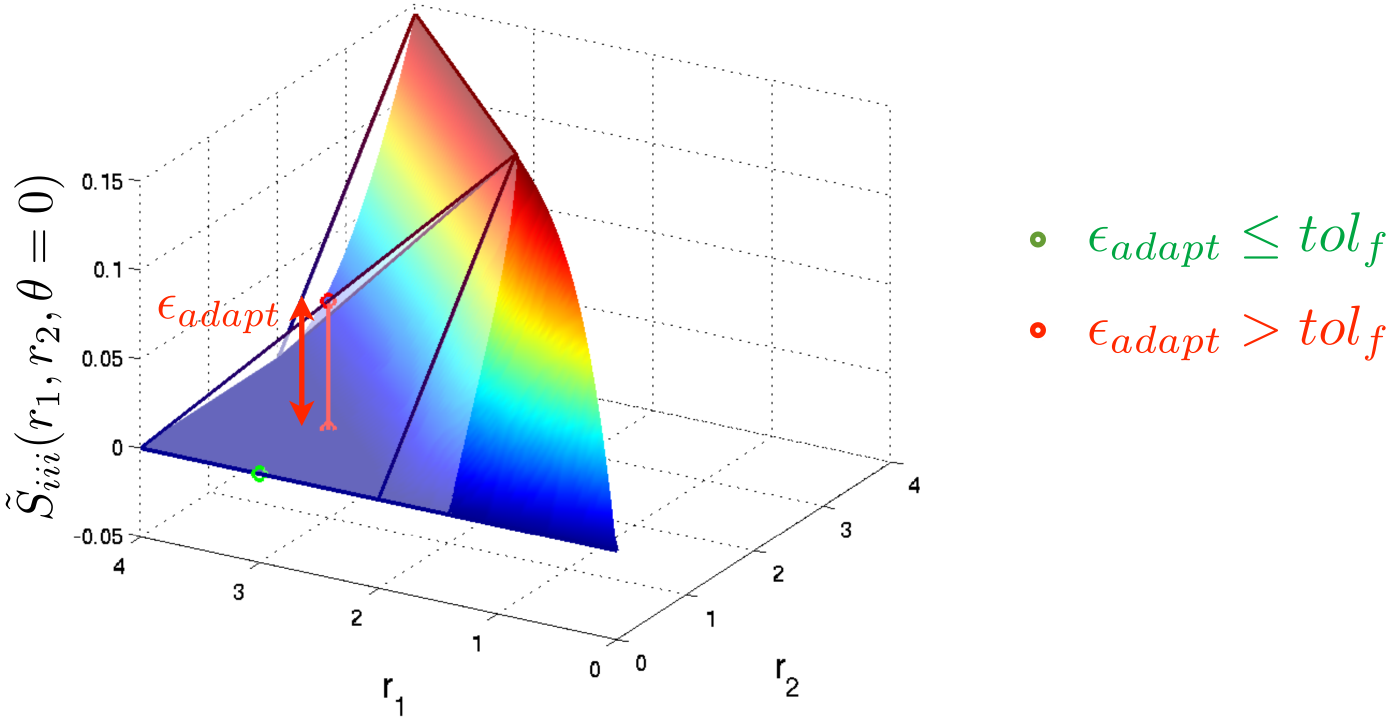
$tol_f = 0.005 \max\left(\tilde{S}_{iii}(r_1, r_2, \theta)\right)$

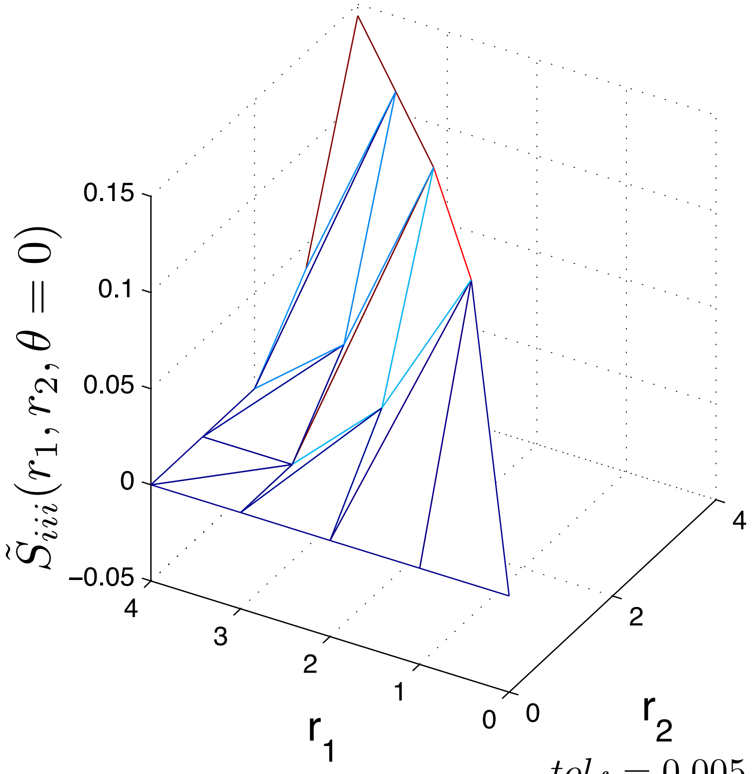
Algorithm Summary

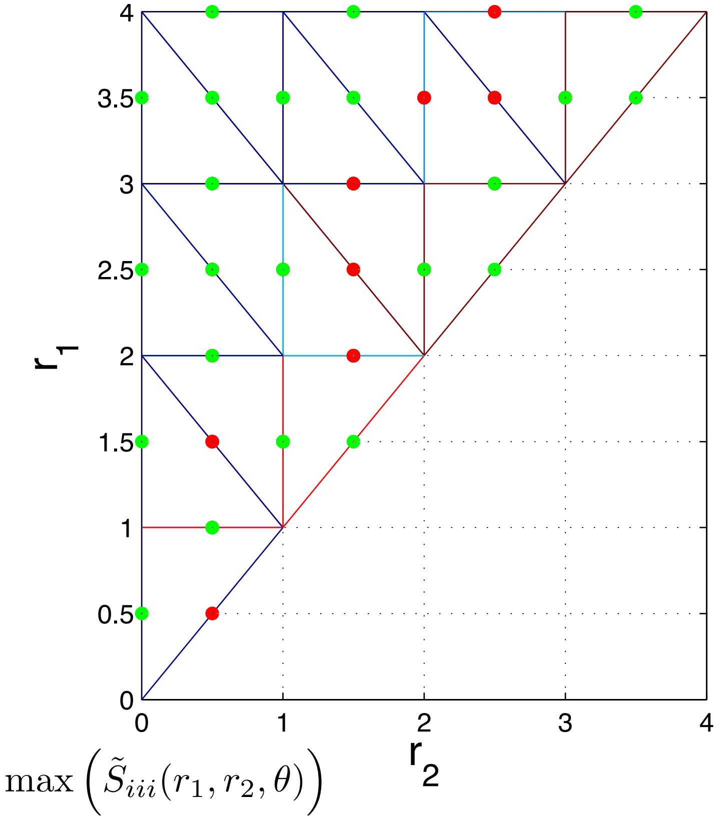
- Construct initial regular tetrahedral grid for domain $[r_1 = 0, r_1 = r^\infty] \times [r_2 = 0, r_2 = r_1] \times [\theta = 0, \theta = \pi]$. Initial triangulation and associated function values, $\tilde{S}_{iii}(r_1, r_2, \theta)$, define $\mathcal{T}_{l=0}$ (l is adaptive iteration level).
- A bisection method is then used to refine the interpolant based on the local error of each tetrahedron:
  - For all tetrahedron midpoints in $\mathcal{T}_l$, evaluate the error indicator function, $\epsilon_{adapt} = \left|\tilde{S}_{iii}(r_1, r_2, \theta) - \mathcal{T}_l(r_1, r_2, \theta)\right|$.
  - If $\epsilon_{adapt} > tol_f$ for given midpoint, each edge of the tetrahedron is bisected and added to $\mathcal{T}_{l+1}$.
  - If all midpoints in a tetrahedron satisfy $\epsilon_{adapt} \leq tol_f$, the tetrahedron is added to $\mathcal{T}_{l+1}$ unchanged.
  - Repeat until all midpoints satisfy $\epsilon_{adapt} \leq tol_f$
- After constructing $\tilde{S}_{iii}(r_1, r_2, \theta)$, MC integration is utilized for computing ζ_i.

Numerical Methods

Iteratively construct a Delaunay triangulation with local linear interpolation (C^0 continuity) to create an interpolant of $\tilde{S}_{iii}(r_1, r_2, \theta)$.







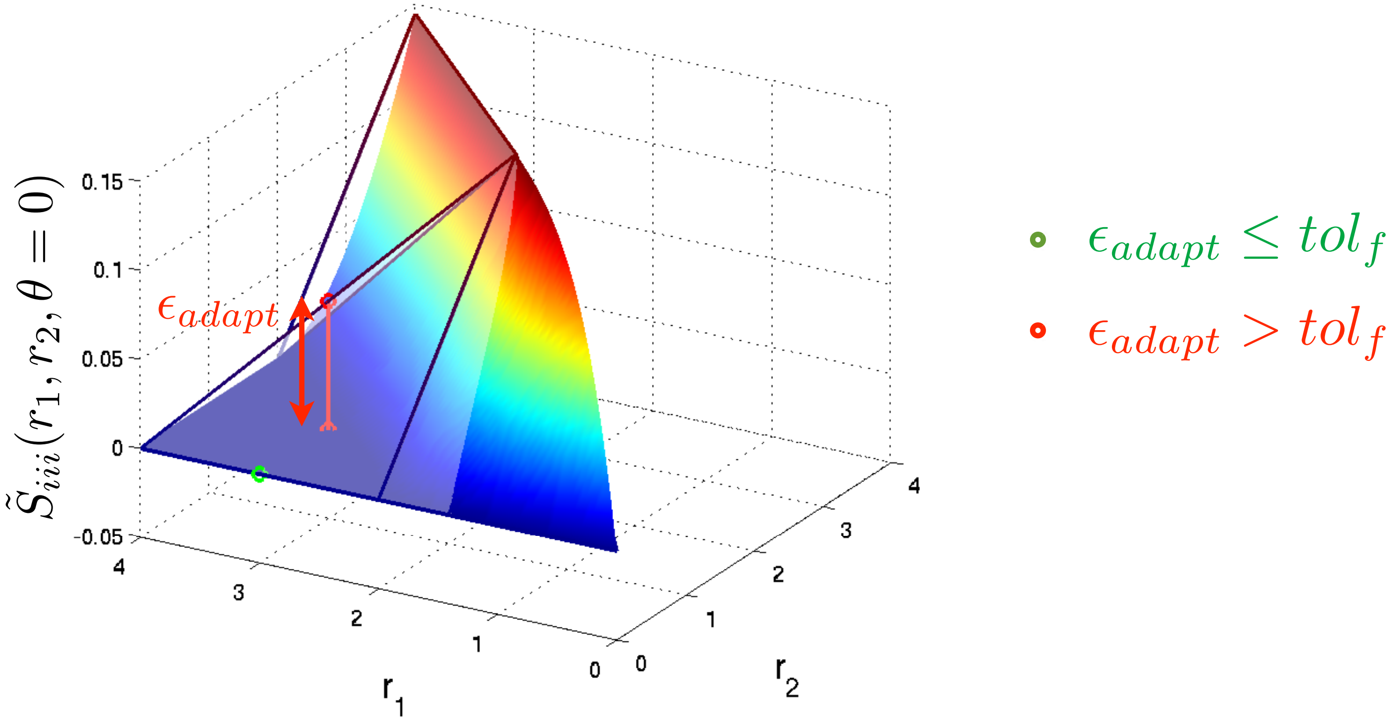
$tol_f = 0.005 \max\left(\tilde{S}_{iii}(r_1, r_2, \theta)\right)$

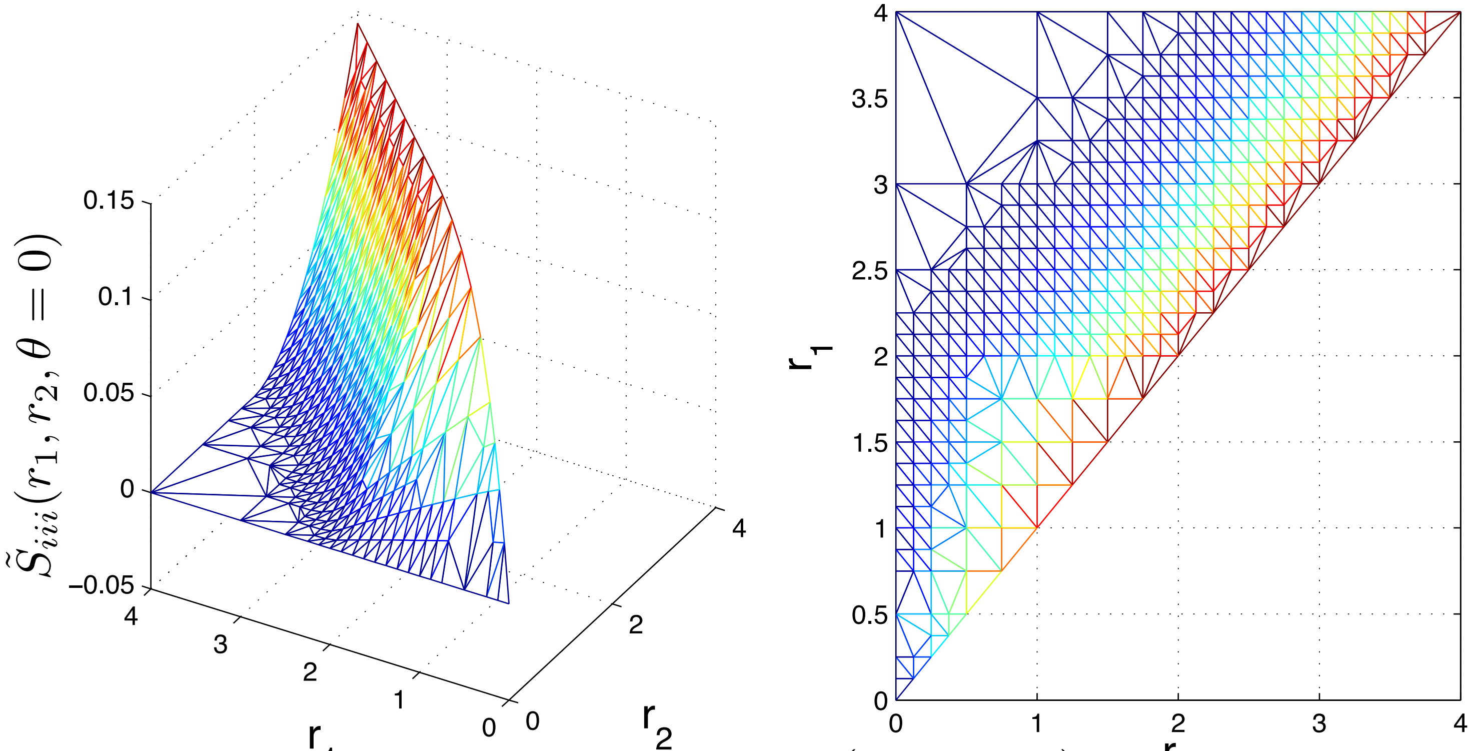
Algorithm Summary

- Construct initial regular tetrahedral grid for domain $[r_1 = 0, r_1 = r^\infty] \times [r_2 = 0, r_2 = r_1] \times [\theta = 0, \theta = \pi]$. Initial triangulation and associated function values, $\tilde{S}_{iii}(r_1, r_2, \theta)$, define $\mathcal{T}_{l=0}$ (l is adaptive iteration level).
- A bisection method is then used to refine the interpolant based on the local error of each tetrahedron:
  - For all tetrahedron midpoints in $\mathcal{T}_l$, evaluate the error indicator function, $\epsilon_{adapt} = \left|\tilde{S}_{iii}(r_1, r_2, \theta) - \mathcal{T}_l(r_1, r_2, \theta)\right|$.
  - If $\epsilon_{adapt} > tol_f$ for given midpoint, each edge of the tetrahedron is bisected and added to $\mathcal{T}_{l+1}$.
  - If all midpoints in a tetrahedron satisfy $\epsilon_{adapt} \leq tol_f$, the tetrahedron is added to $\mathcal{T}_{l+1}$ unchanged.
  - Repeat until all midpoints satisfy $\epsilon_{adapt} \leq tol_f$
- After constructing $\tilde{S}_{iii}(r_1, r_2, \theta)$, MC integration is utilized for computing ζ_i.

Numerical Methods

Iteratively construct a Delaunay triangulation with local linear interpolation (C^0 continuity) to create an interpolant of $\tilde{S}_{iii}(r_1, r_2, \theta)$.





$tol_f = 0.005 \max\left(\tilde{S}_{iii}(r_1, r_2, \theta)\right)$

Algorithm Summary

- Construct initial regular tetrahedral grid for domain $[r_1 = 0, r_1 = r^\infty] \times [r_2 = 0, r_2 = r_1] \times [\theta = 0, \theta = \pi]$. Initial triangulation and associated function values, $\tilde{S}_{iii}(r_1, r_2, \theta)$, define $\mathcal{T}_{l=0}$ (l is adaptive iteration level).

- A bisection method is then used to refine the interpolant based on the local error of each tetrahedron:
  - For all tetrahedron midpoints in $\mathcal{T}_l$, evaluate the error indicator function, $\epsilon_{adapt} = \left|\tilde{S}_{iii}(r_1, r_2, \theta) - \mathcal{T}_l(r_1, r_2, \theta)\right|$.
  - If $\epsilon_{adapt} > tol_f$ for given midpoint, each edge of the tetrahedron is bisected and added to $\mathcal{T}_{l+1}$.
  - If all midpoints in a tetrahedron satisfy $\epsilon_{adapt} \leq tol_f$, the tetrahedron is added to $\mathcal{T}_{l+1}$ unchanged.
  - Repeat until all midpoints satisfy $\epsilon_{adapt} \leq tol_f$

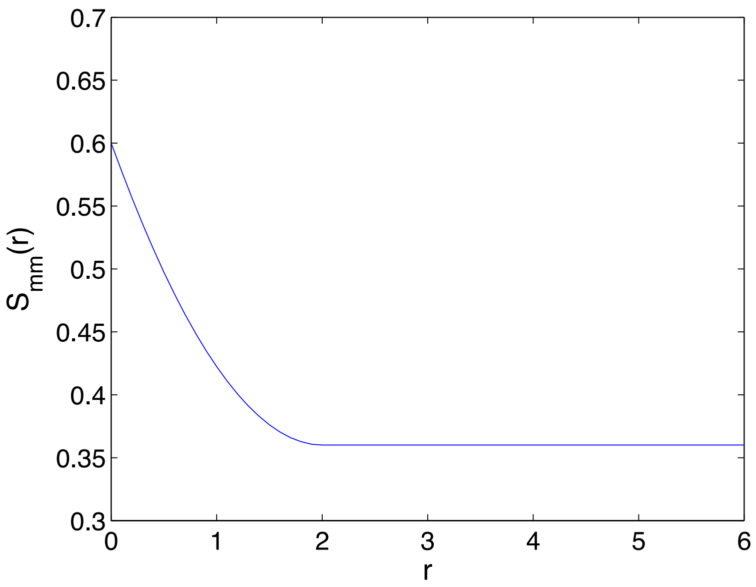
- After constructing $\tilde{S}_{iii}(r_1, r_2, \theta)$, MC integration is utilized for computing ζ_i.

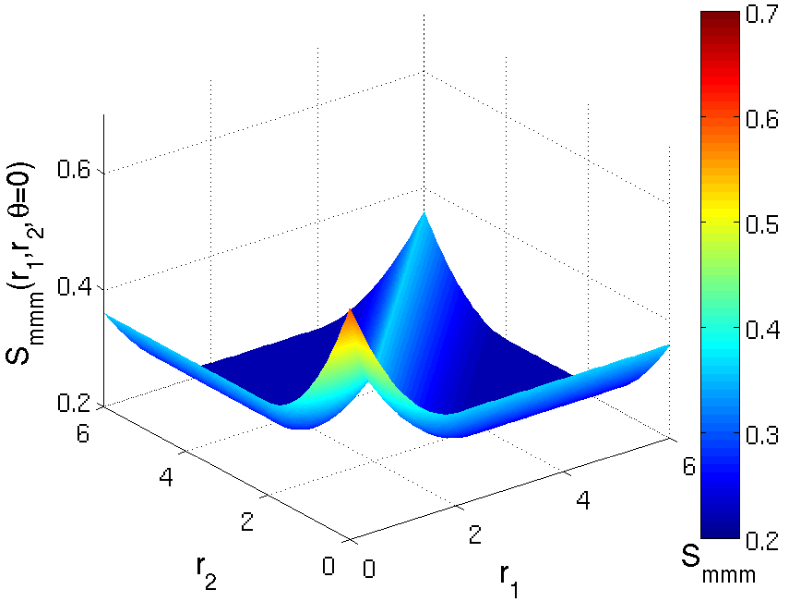
Penetrable Sphere Model (Verification Example)

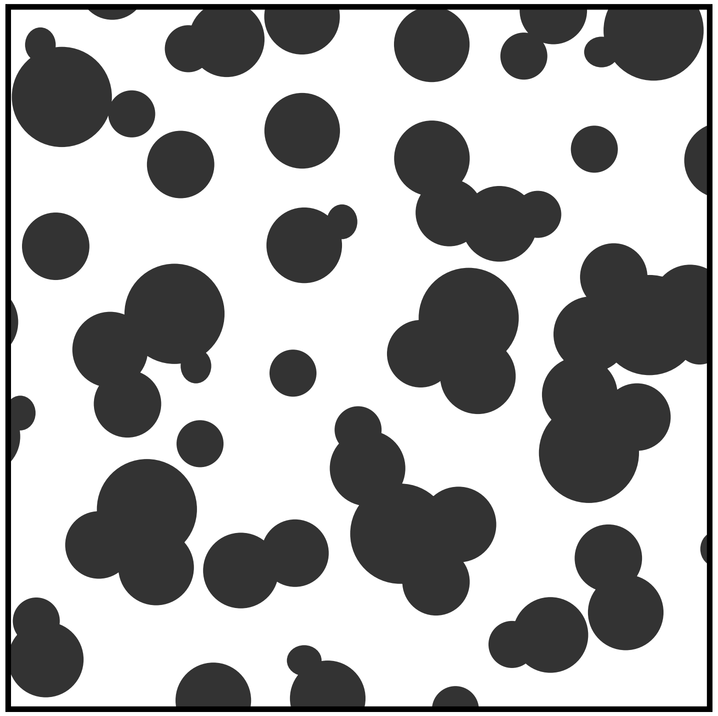
- n-point probability functions can be formulated analytically for penetrable sphere model

$$S_{m\cdots m}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) = \exp\left(-\rho V_n\right)$$

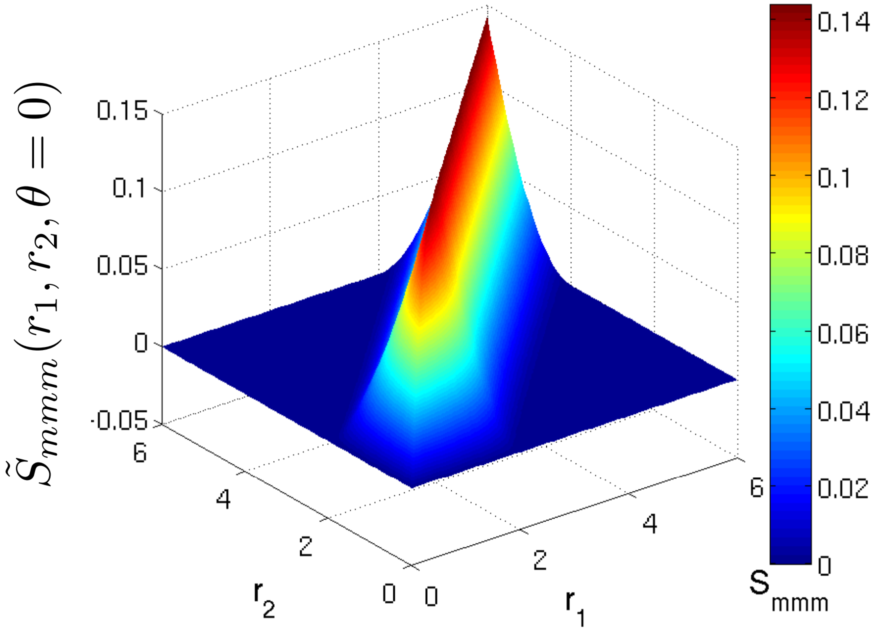
$V_n -$ union volume of n spheres, $\rho -$ number density of spheres







Penetrable sphere microstructure

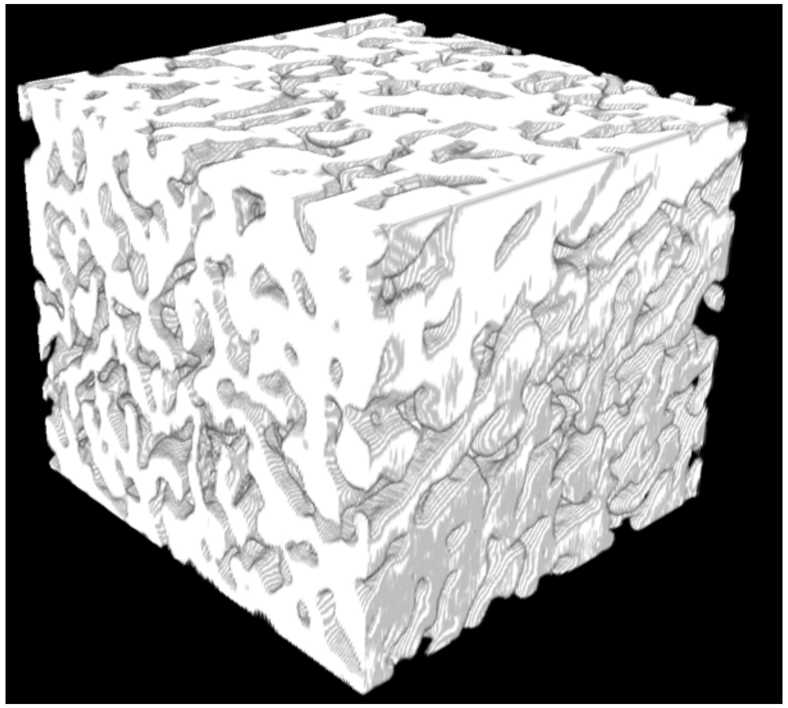


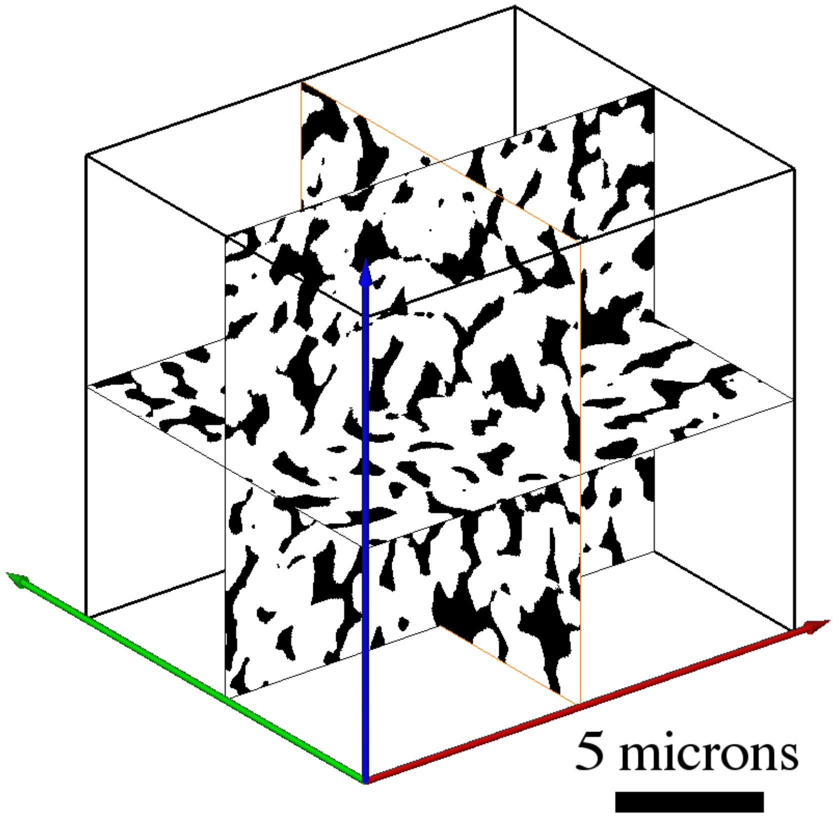
- Comparison of our microstructural parameters to literature

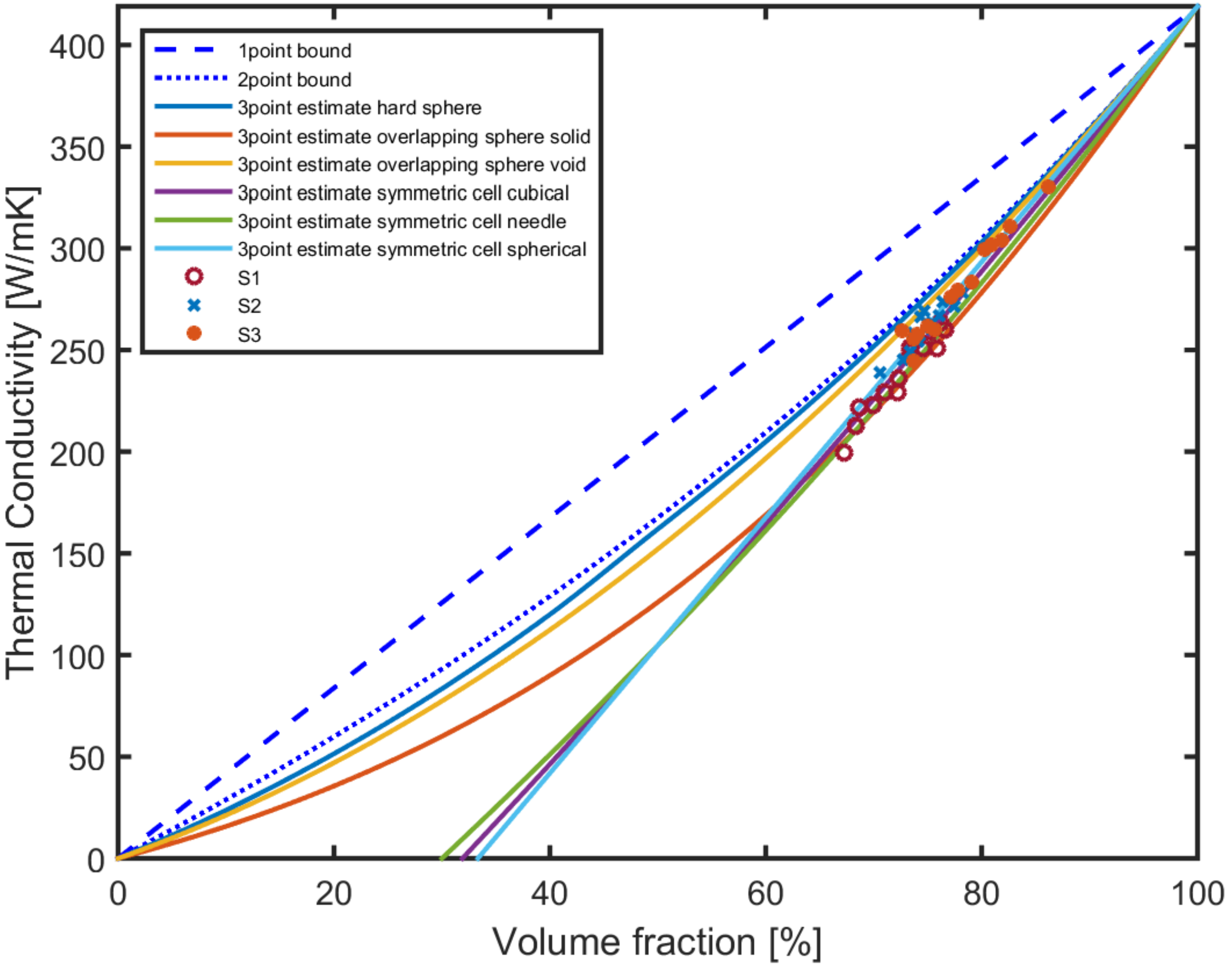
c_p	ζ_m	ζ_m^{R1}	ε_{PS}^{ζ}	η_m	η_m^{R1}	ε_{PS}^{η}
0.2	0.5187	0.5174	0.24%	0.4178	0.4163	0.35%
0.4	0.6571	0.6489	1.26%	0.5579	0.5604	0.44%
0.6	0.7743	0.7702	0.54%	0.6987	0.7050	0.89%

[R1] - Helte, *Proc. R. Soc. A.* 79(3), 1983.

Porous Silver Paste

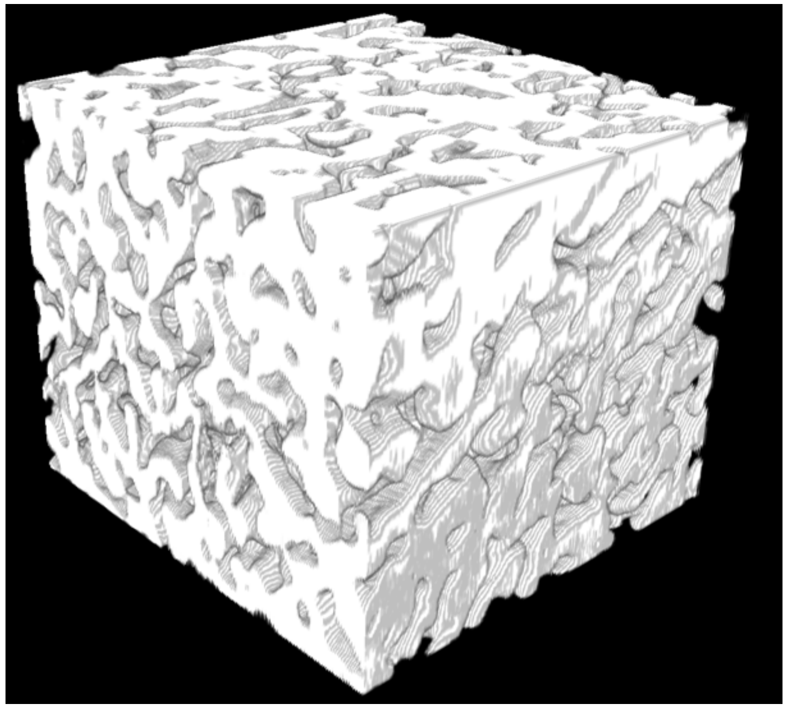


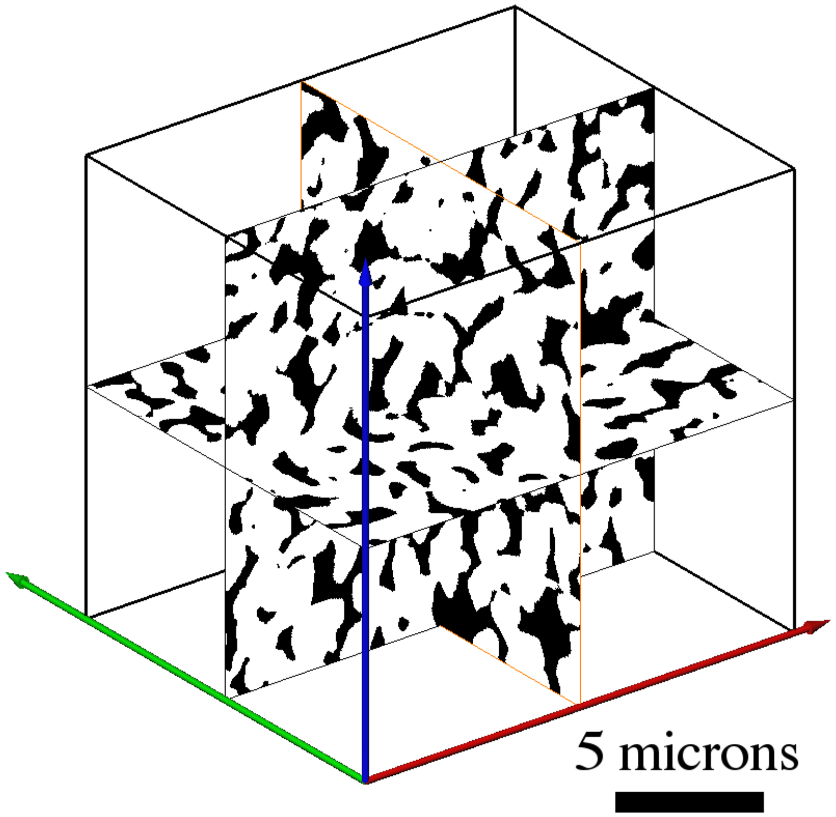


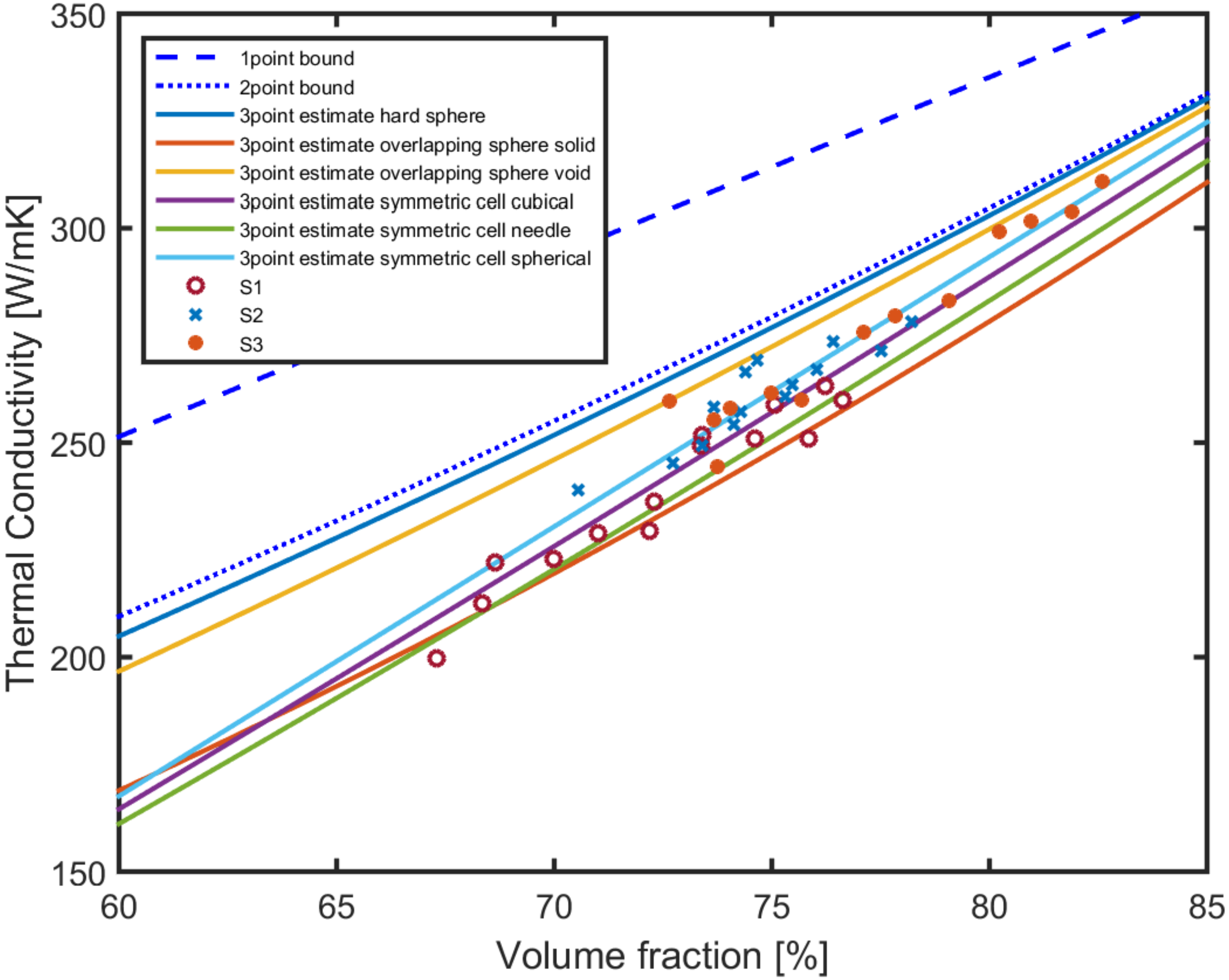


Mark Roelofs, Andrew Gillman, Varvara Kouznetsova, Karel Matous, Marc Geers

Porous Silver Paste







Mark Roelofs, Andrew Gillman, Varvara Kouznetsova, Karel Matous, Marc Geers

Ordinary Differential Equation (ODE)

- Explicit methods for ODEs need sparse-matrix-vector mult.
- Implicit methods for ODEs need to solve linear systems
- Direct methods (Gaussian elimination)
  - Called LU Decomposition, because we factor A = L*U.
  - Future lectures will consider both dense and sparse cases.
  - More complicated than sparse-matrix vector multiplication.
- Iterative solvers
  - Will discuss several of these in future.
    - Jacobi, Successive over-relaxation (SOR) , Conjugate Gradient (CG), Multigrid,...
  - Most have sparse-matrix-vector multiplication in kernel.

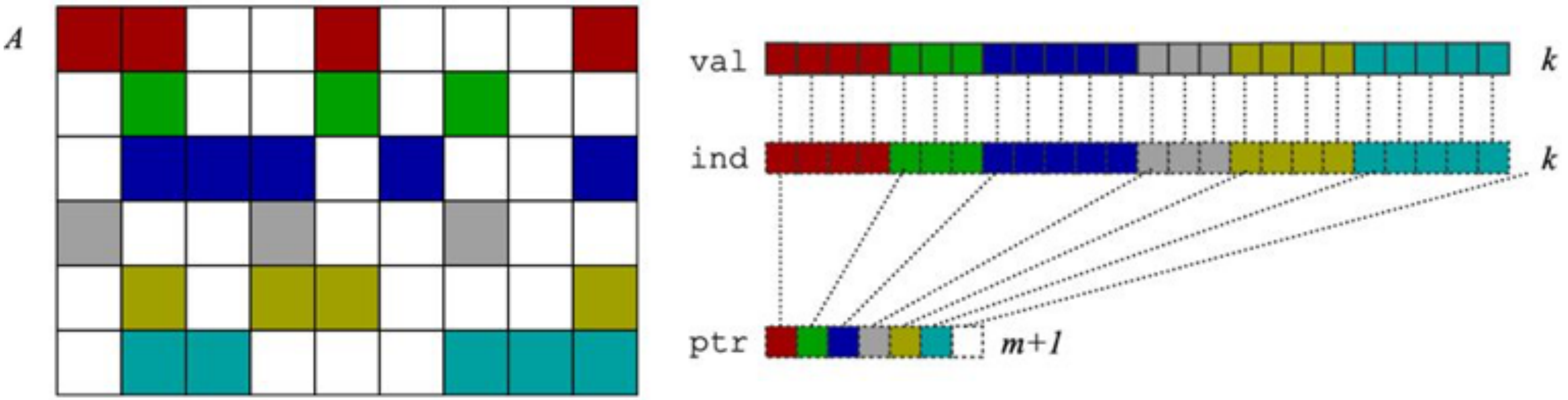
- Eigenproblems
  - Also depend on sparse-matrix-vector multiplication, direct methods.

Partial Differential Equation (PDE)

- As with ODEs, either explicit or implicit approaches are possible
  - Explicit, sparse matrix-vector multiplication
  - Implicit, sparse matrix solve at each step
    - Direct solvers are hard
    - Iterative solves turn into sparse matrix-vector multiplication
      - Graph partitioning

- Graph and sparse matrix correspondence:
  - Sparse matrix-vector multiplication is nearest neighbor "averaging" on the underlying mesh
- Not all nearest neighbor computations have the same efficiency
  - Depends on the mesh structure (nonzero structure) and the number of Flops per point.

SMP — Compressed Sparse Row (CSR) Format



Matrix-vector multiply kernel: $y(i) \leftarrow y(i) + A(i,j)*x(j)$

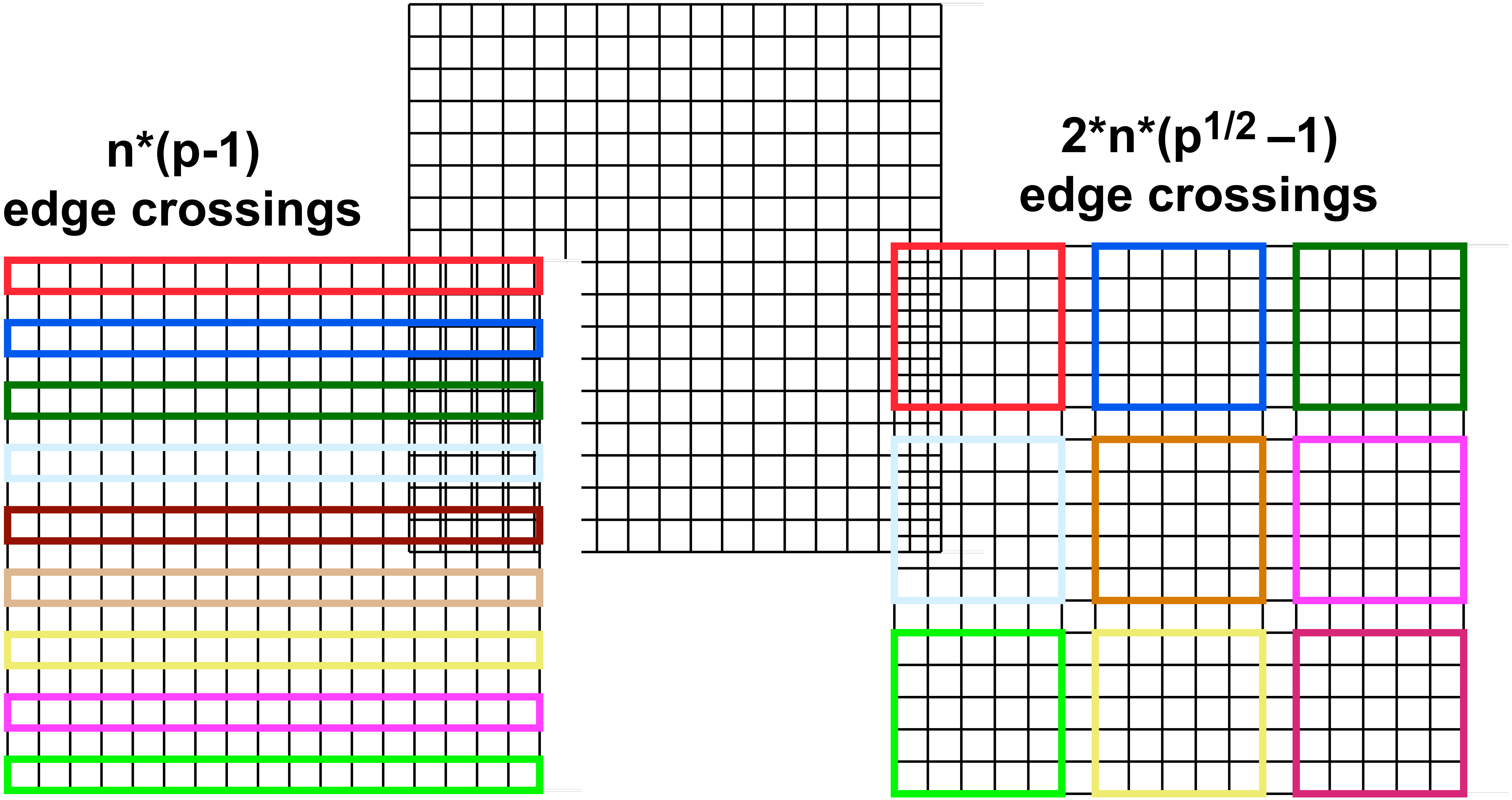
```
for each row i
   for k=ptr[i] to ptr[i+1] do
       y[i] = y[i] + val[k]*x[ind[k]]
```

CS267 Lectures

Domain Decomposition

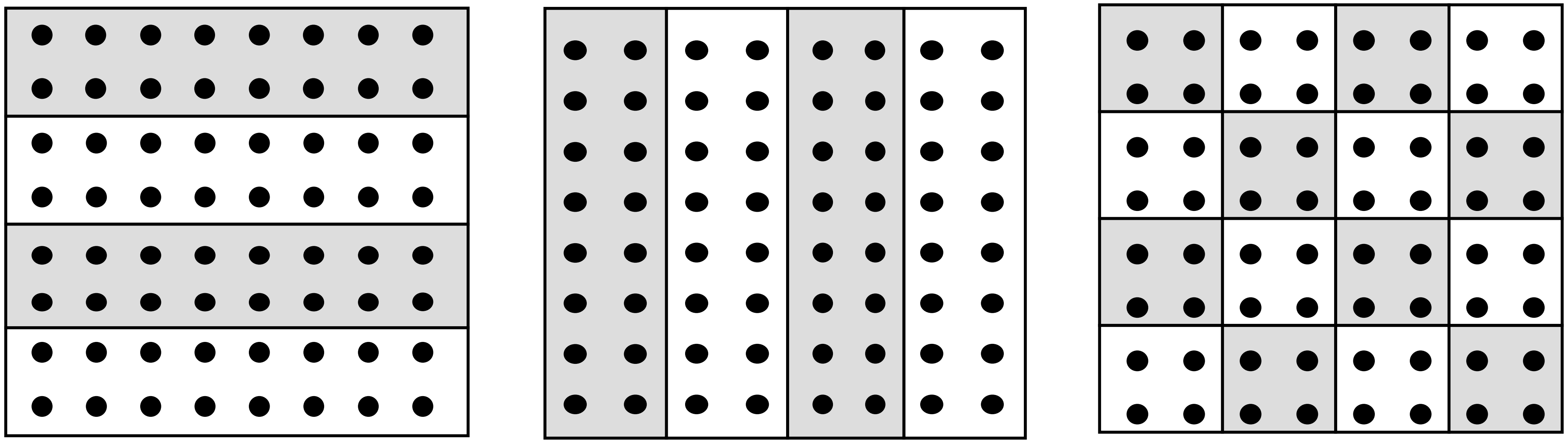
- Suppose graph is nxn mesh with connection neighbors
- Which partition has less communication? (n=18, p=9)

- Minimizing communication on mesh ≡ minimizing “surface to volume ratio” of partition

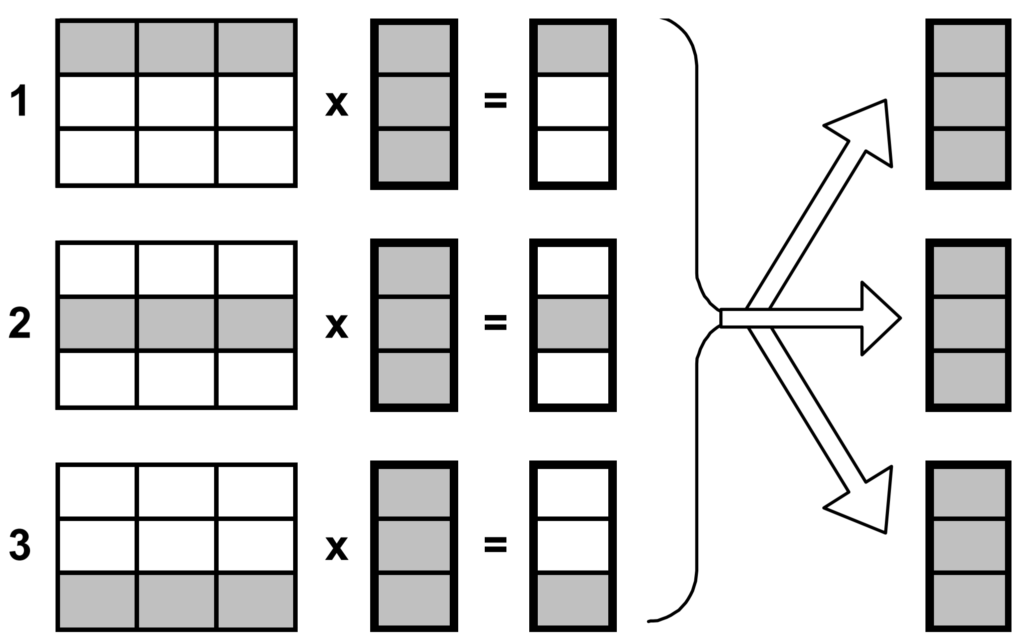


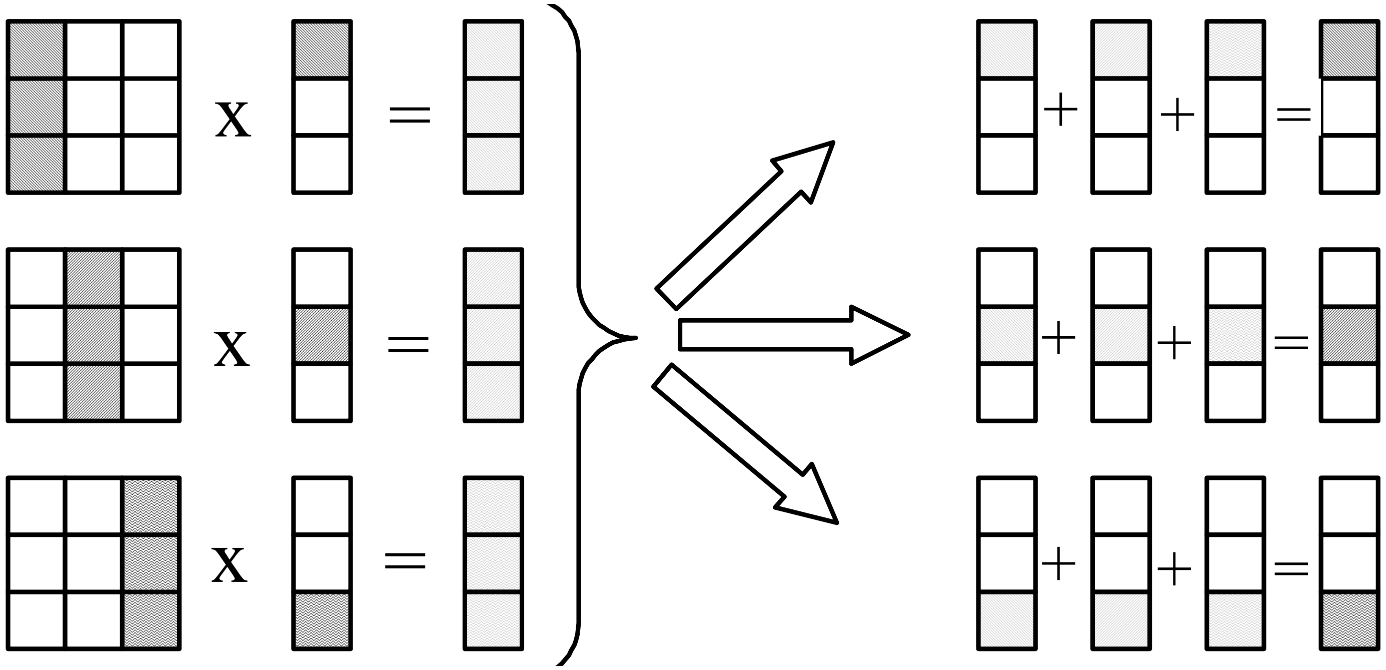
Parallel Sparse Matrix-vector multiplication

- Matrix decomposition



- Row-wise decomposition
- Column-wise decomposition

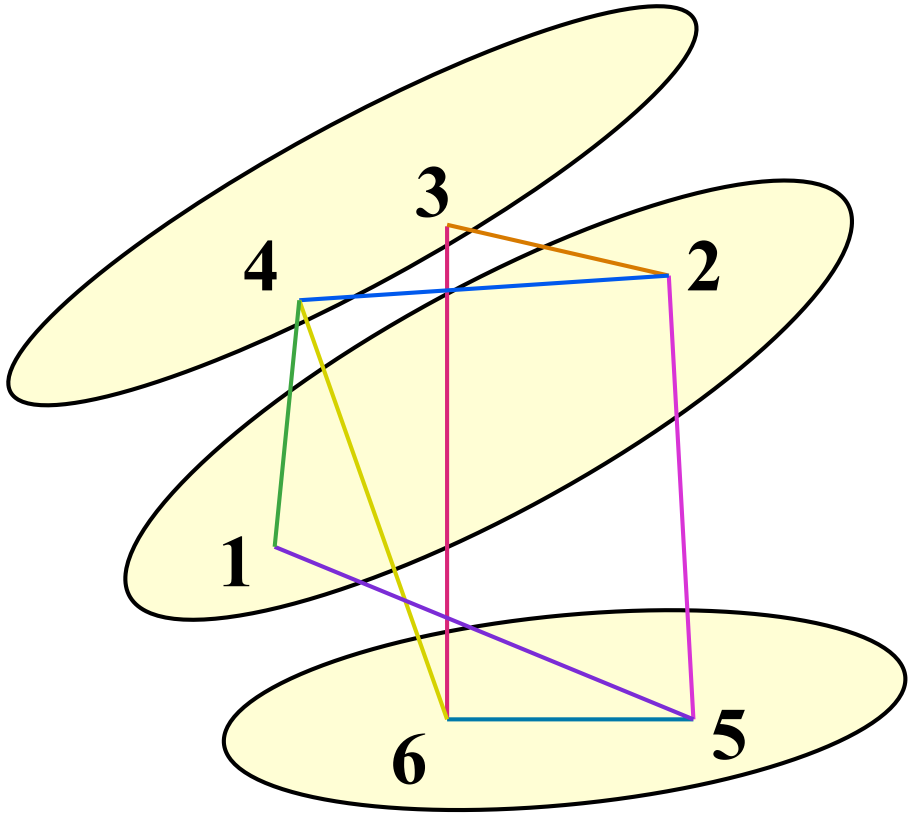




Graph Partitioning and Sparse Matrices

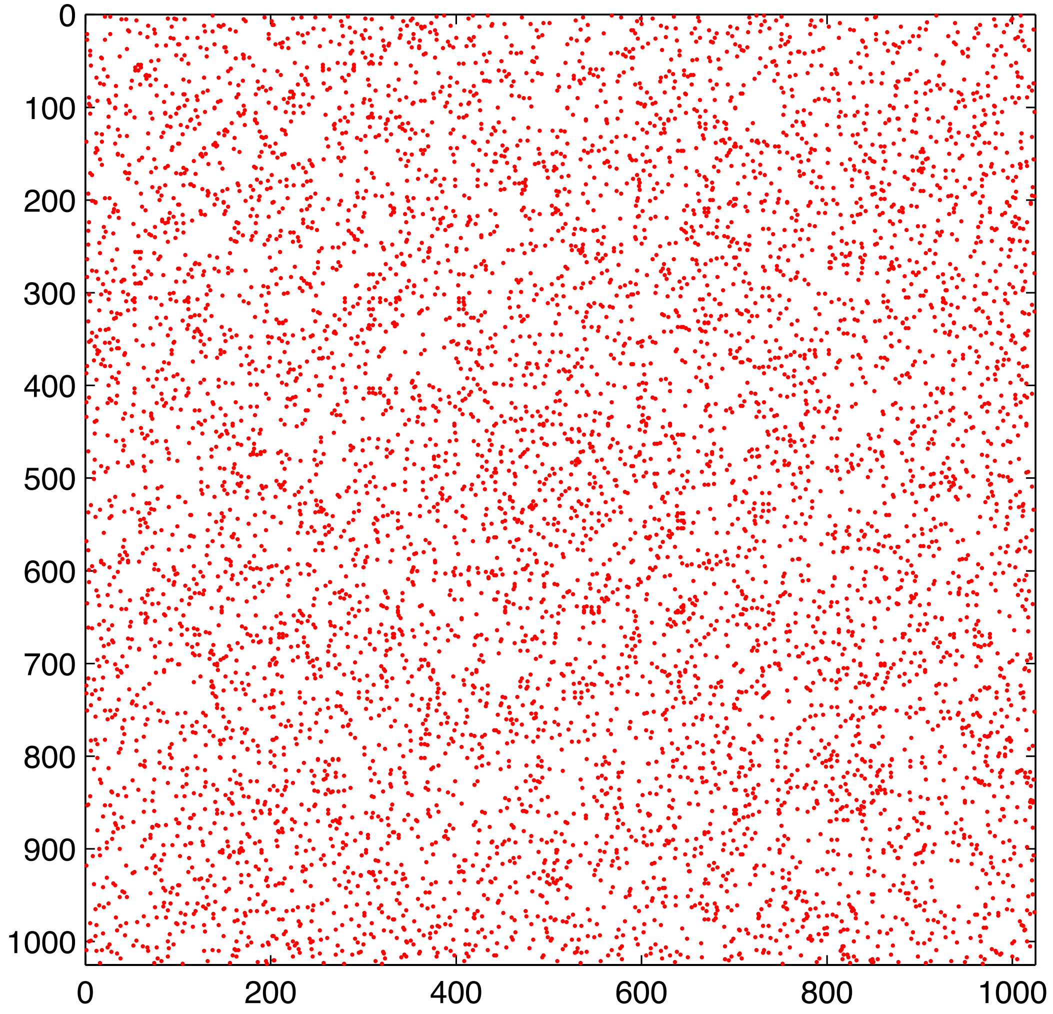
- Relationship between matrix and graph

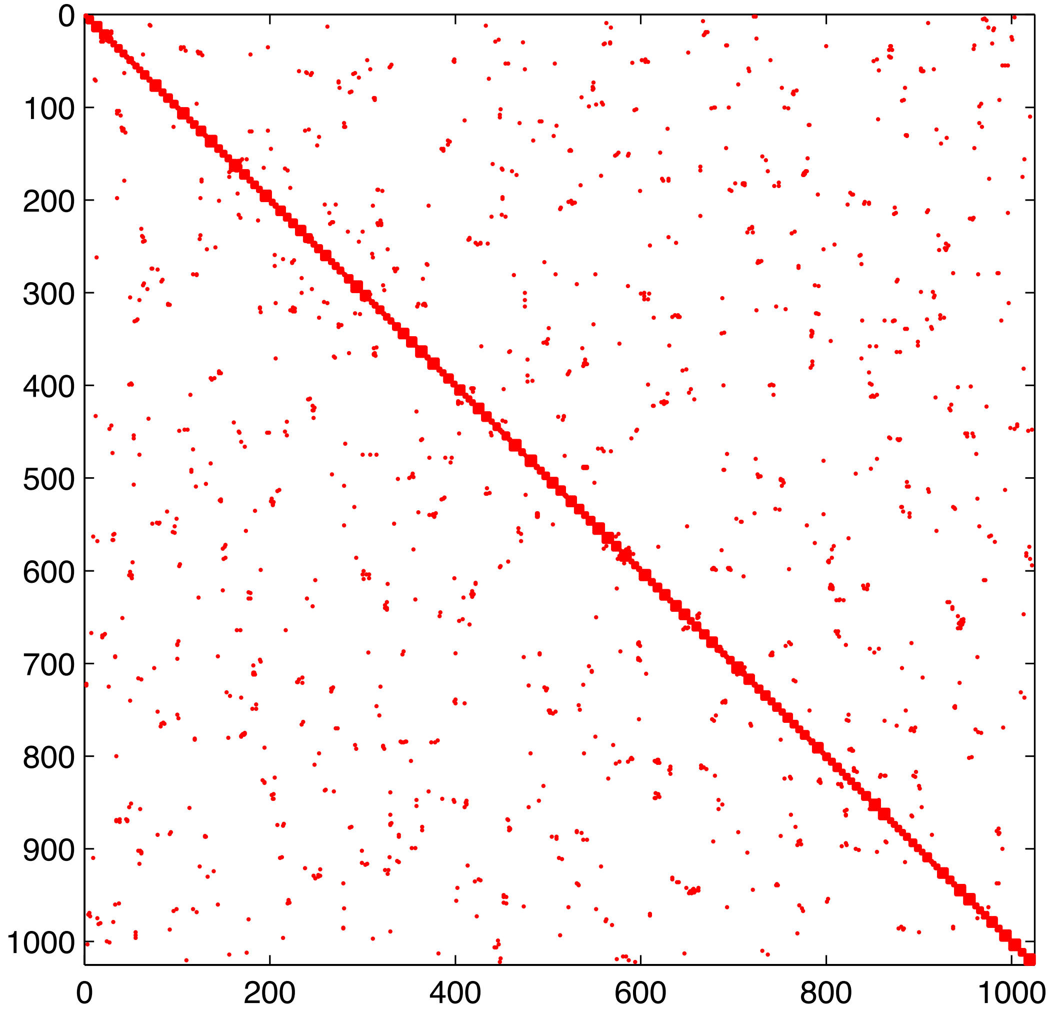
$$\begin{array}{c|cccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 1 & & & 1 & 1 & \\
2 & & 1 & 1 & 1 & 1 & \\
3 & & 1 & 1 & & & 1 \\
4 & 1 & 1 & & 1 & & 1 \\
5 & 1 & 1 & & & 1 & 1 \\
6 & & & 1 & 1 & 1 & 1 \\
\hline
\end{array}$$



- Edges in the graph are nonzero in the matrix: here the matrix is symmetric (edges are unordered) and weights are equal (1)
- If divided over 3 procs, there are 14 nonzeros outside the diagonal blocks, which represent the 7 (bidirectional) edges

Graph Partitioning and Sparse Matrices

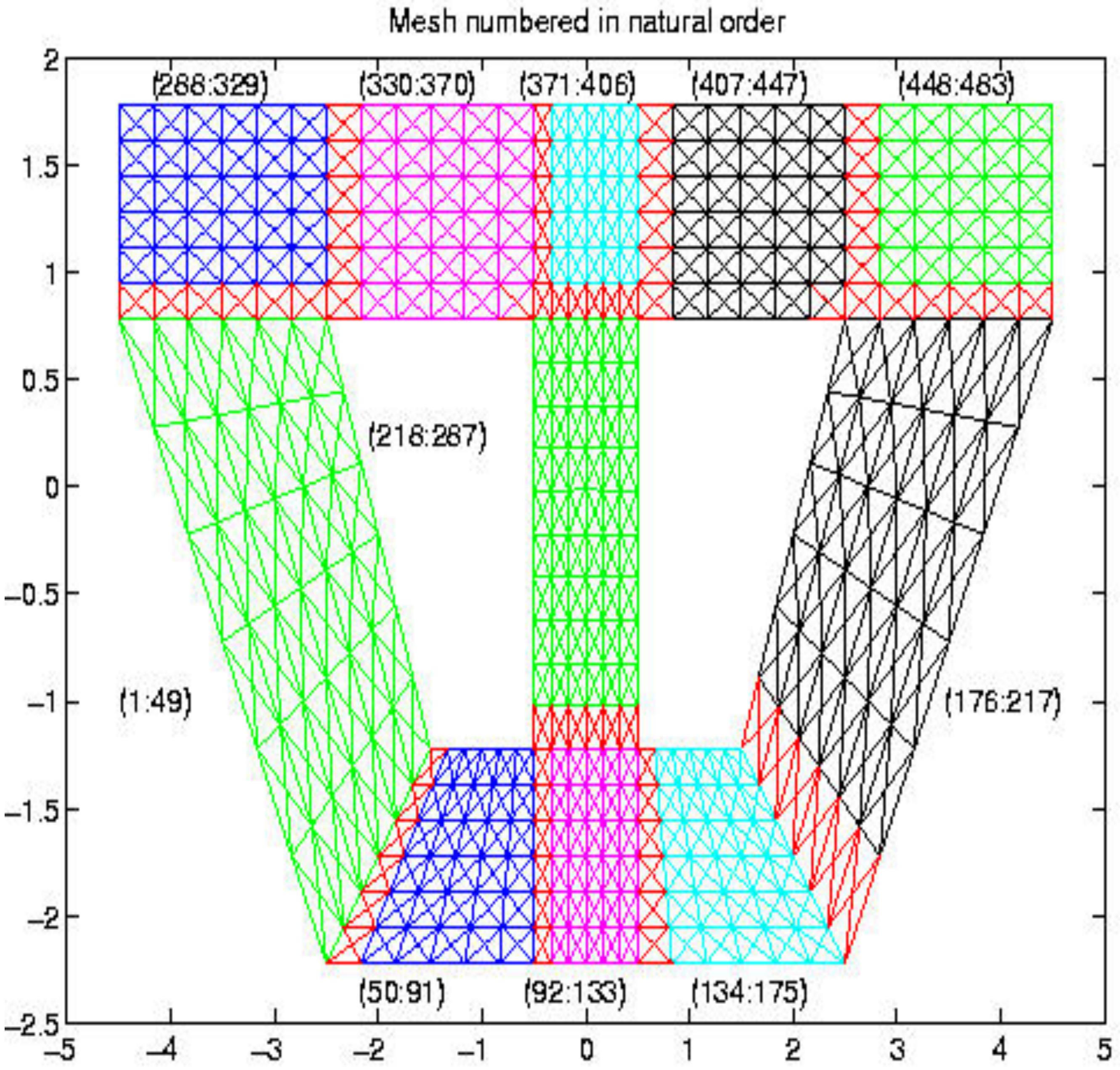


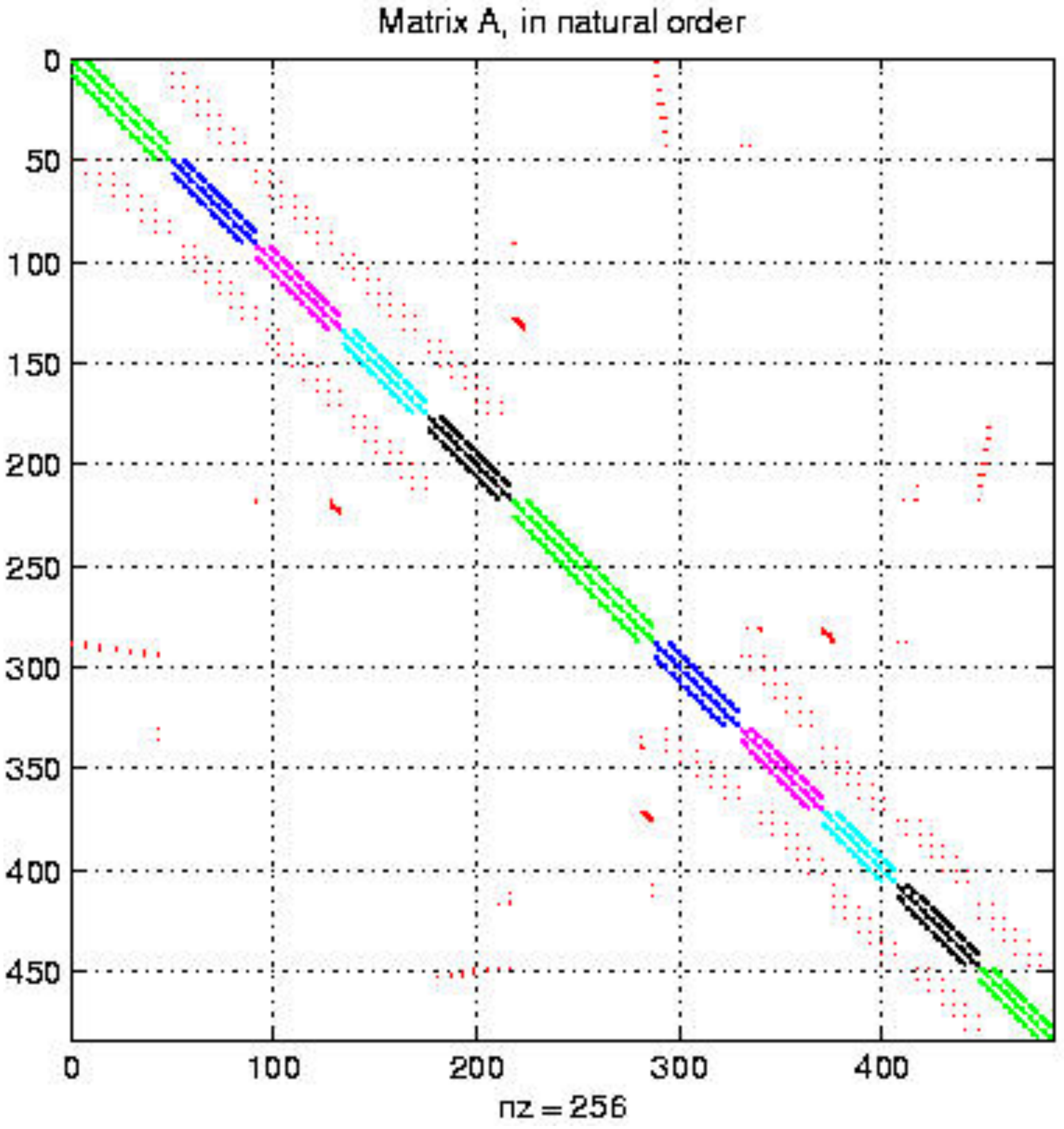


- High–performance graph algorithms from parallel sparse matrices, Gilbert et al. 2006

- Zoltan — Parallel Partitioning, Load Balancing and Data-Management Services (http://www.cs.sandia.gov/zoltan/)
- ParMETIS — Parallel Graph Partitioning and Fill-reducing Matrix Ordering (http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview)
- PT-SCOTCH —Graph and mesh partitioning, static mapping, and sparse matrix ordering (http://www.labri.fr/perso/pelegrin/scotch/)

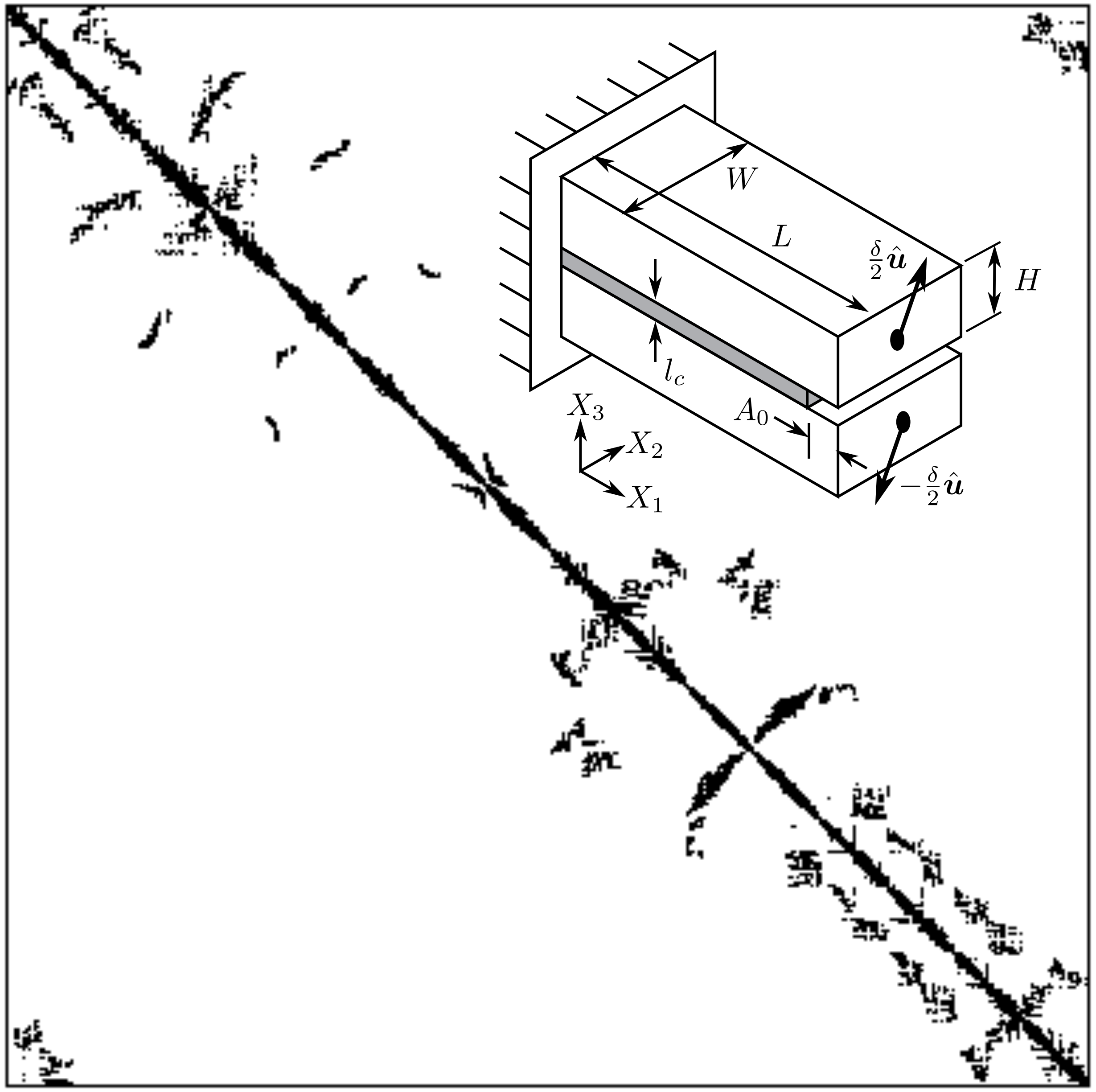
Meshes in Computational Mechanics



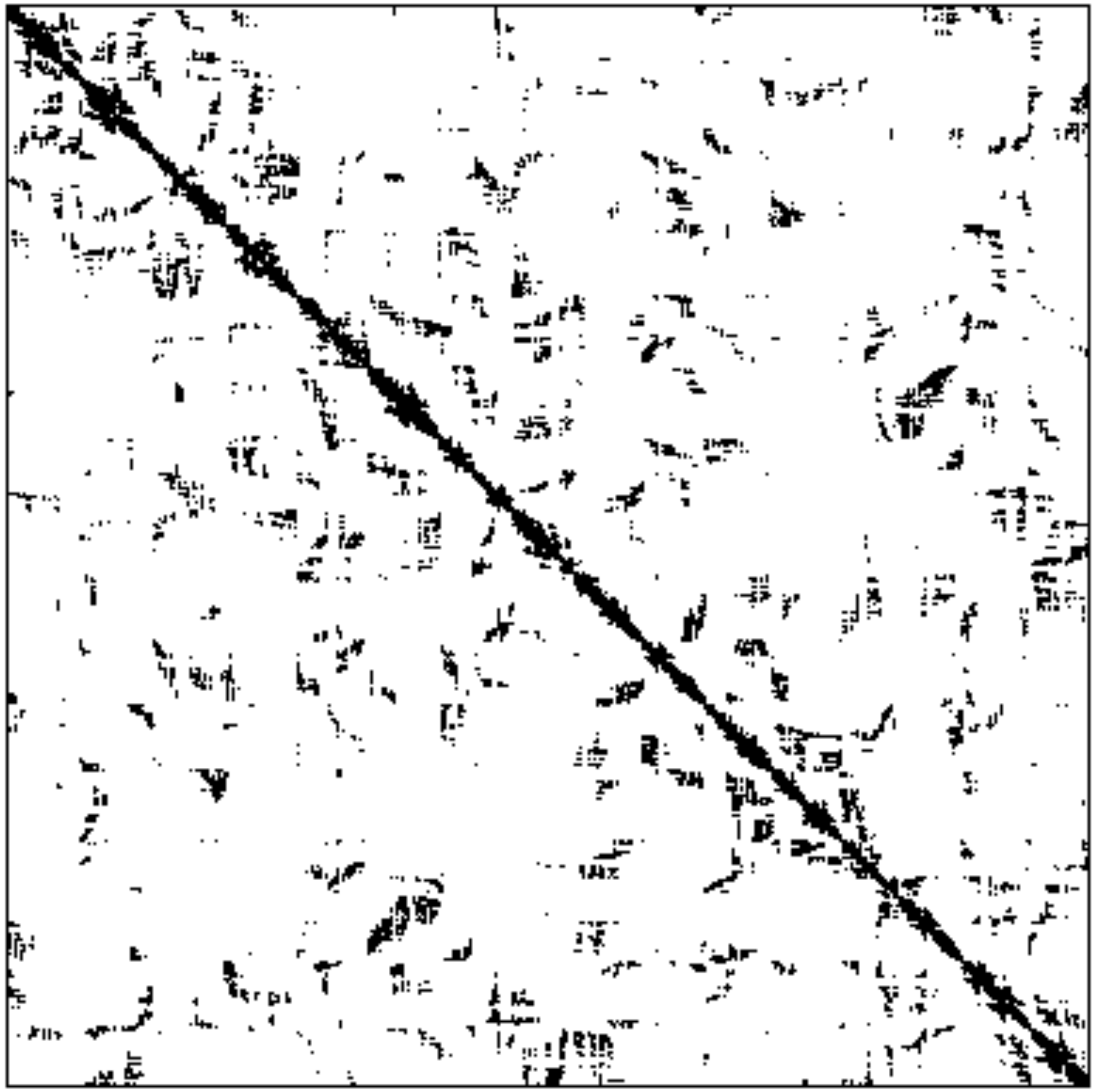


Matrices in Computational Mechanics

Sparcity (Np/Ndof/NZ): (16/1049202/60686684)



Sparcity (Np/Ndof/NZ): (16/125984/7771256)



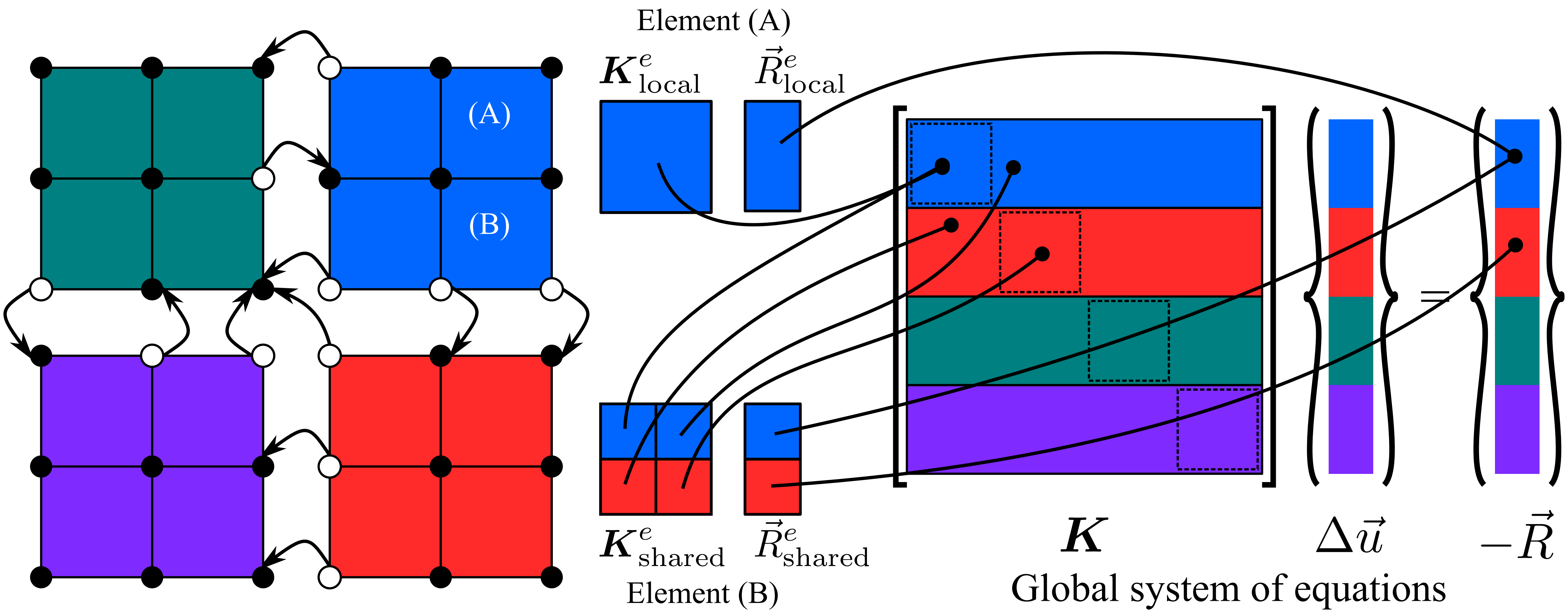
- Volumetric + Cohesive Elements
- Periodic Unit Cell

Challenges of Irregular Meshes

- How to generate them in the first place
  - Start from geometric description of object
  - 2D hard!
  - 3D harder!!
- How to partition them
  - ParMetis, a parallel graph partitioner
- How to design iterative solvers
  - PETSc, a Portable Extensible Toolkit for Scientific Computing
  - Prometheus, a multigrid solver for finite element problems on irregular meshes
- How to design direct solvers
  - SuperLU, parallel sparse Gaussian elimination

- These are challenges to do sequentially, more so in parallel

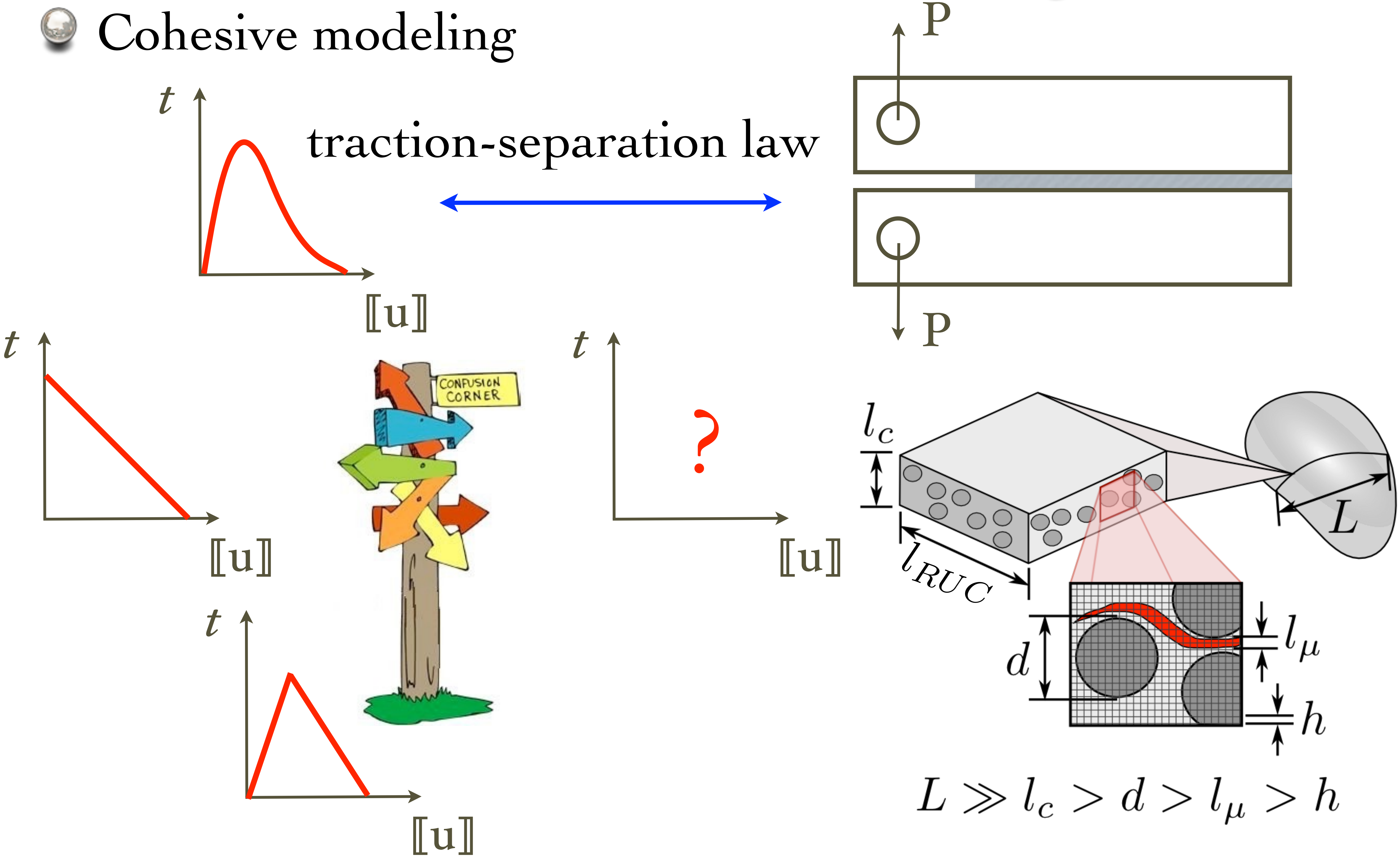
Stiffness Matrix Assembly — Finite Element Method


(A)
(B)
Element (A)
$\boldsymbol{K}^e_{\text{local}}$
$\vec{R}^e_{\text{local}}$
$\boldsymbol{K}^e_{\text{shared}}$
$\vec{R}^e_{\text{shared}}$
Element (B)
$=$
$\boldsymbol{K}$
$\Delta\vec{u}$
$-\vec{R}$
Global system of equations

- Use local and global numbering for efficient operations
- Assemble first border elements
- Overlay computation and communication

Multiscale Materials Modeling

- Cohesive modeling

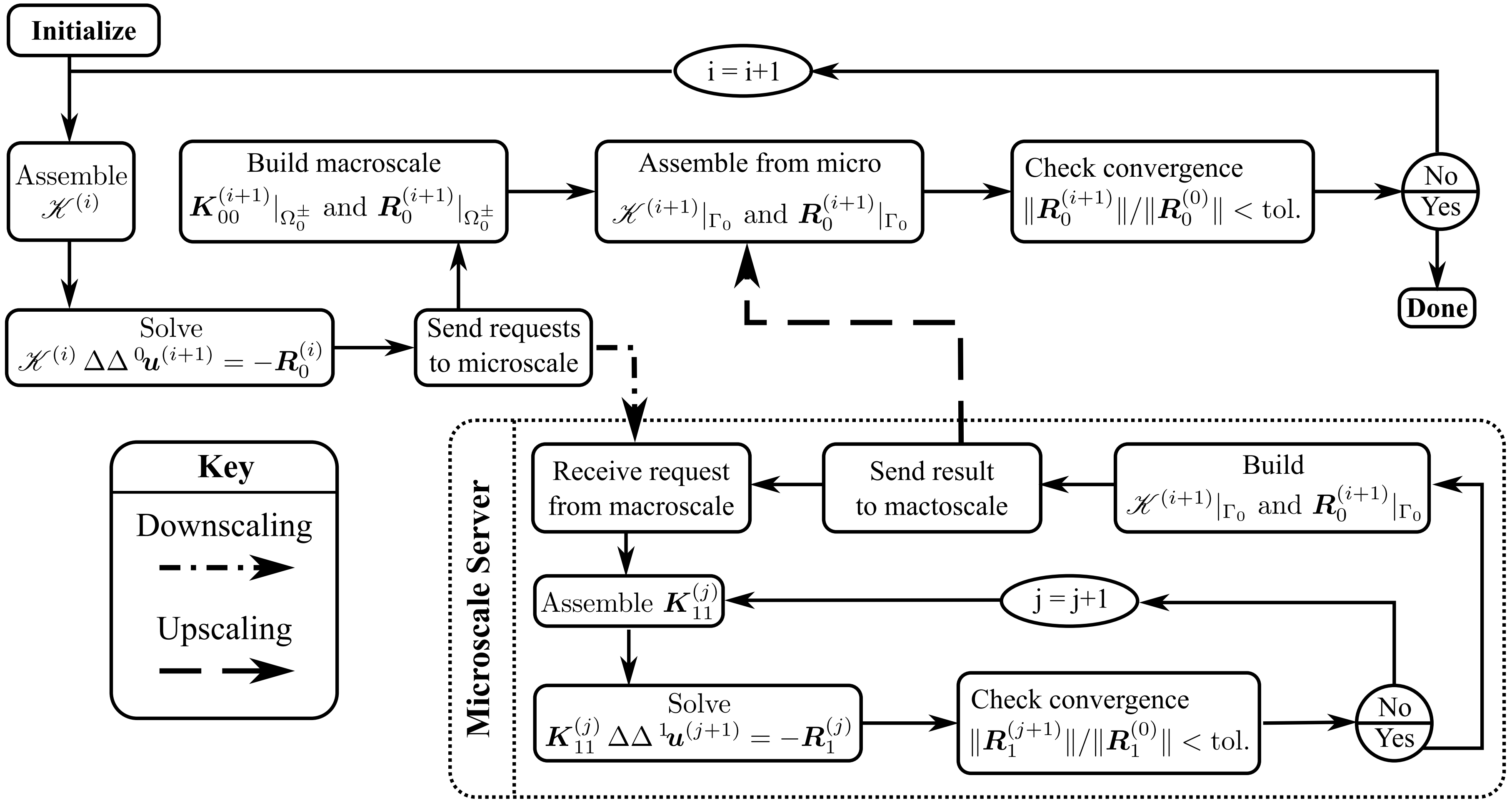

traction-separation law

$$L \gg l_c > d > l_\mu > h$$

- Cohesive law based on lower scale physics

Hierarchically Parallel Multiscale Solver

$$ {}^0\mathcal{R} = \int_{\Omega_0^\pm} {}^0\boldsymbol{P} : \nabla_{\boldsymbol{X}}(\delta^0\boldsymbol{u})\,\mathrm{dV} \quad - \int_{\Omega_0^\pm} \boldsymbol{f}\cdot\delta^0\boldsymbol{u}\,\mathrm{dV} - \int_{\partial\Omega_0^{\mathrm{t}}} \boldsymbol{t}^{\mathrm{p}}\cdot\delta^0\boldsymbol{u}\,\mathrm{dA} + \int_{\Gamma_0} {}^0\boldsymbol{t}\cdot[\![\delta^0\boldsymbol{u}]\!]\;\mathrm{dA} = 0 $$



Initialize

i = i+1

Assemble $\mathscr{K}^{(i)}$

Build macroscale $\boldsymbol{K}_{00}^{(i+1)}|_{\Omega_0^\pm}$ and $\boldsymbol{R}_0^{(i+1)}|_{\Omega_0^\pm}$

Assemble from micro $\mathscr{K}^{(i+1)}|_{\Gamma_0}$ and $\boldsymbol{R}_0^{(i+1)}|_{\Gamma_0}$

Check convergence $\|\boldsymbol{R}_0^{(i+1)}\|/\|\boldsymbol{R}_0^{(0)}\| <$ tol.

No

Yes

Done

Solve $\mathscr{K}^{(i)}\,\Delta\Delta\,{}^0\boldsymbol{u}^{(i+1)} = -\boldsymbol{R}_0^{(i)}$

Send requests to microscale

Key

Downscaling

Upscaling

Microscale Server

Receive request from macroscale

Send result to mactoscale

Build $\mathscr{K}^{(i+1)}|_{\Gamma_0}$ and $\boldsymbol{R}_0^{(i+1)}|_{\Gamma_0}$

Assemble $\boldsymbol{K}_{11}^{(j)}$

j = j+1

Solve $\boldsymbol{K}_{11}^{(j)}\,\Delta\Delta\,{}^1\boldsymbol{u}^{(j+1)} = -\boldsymbol{R}_1^{(j)}$

Check convergence $\|\boldsymbol{R}_1^{(j+1)}\|/\|\boldsymbol{R}_1^{(0)}\| <$ tol.

No

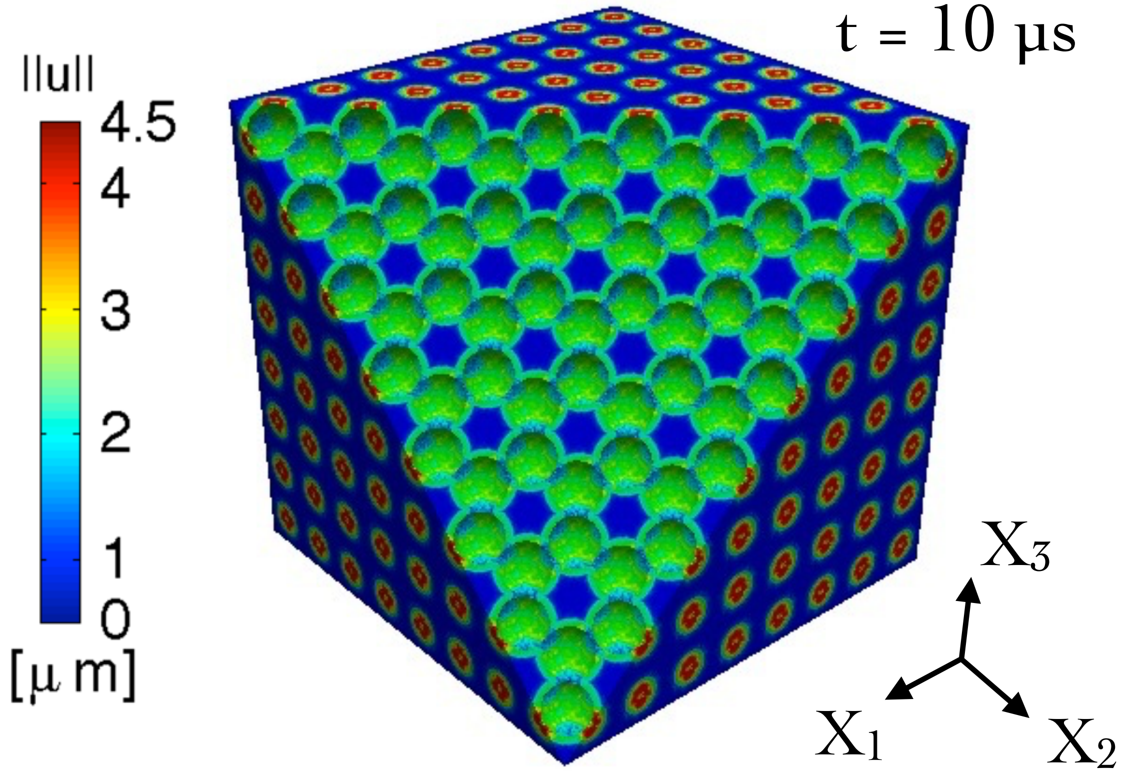
Yes

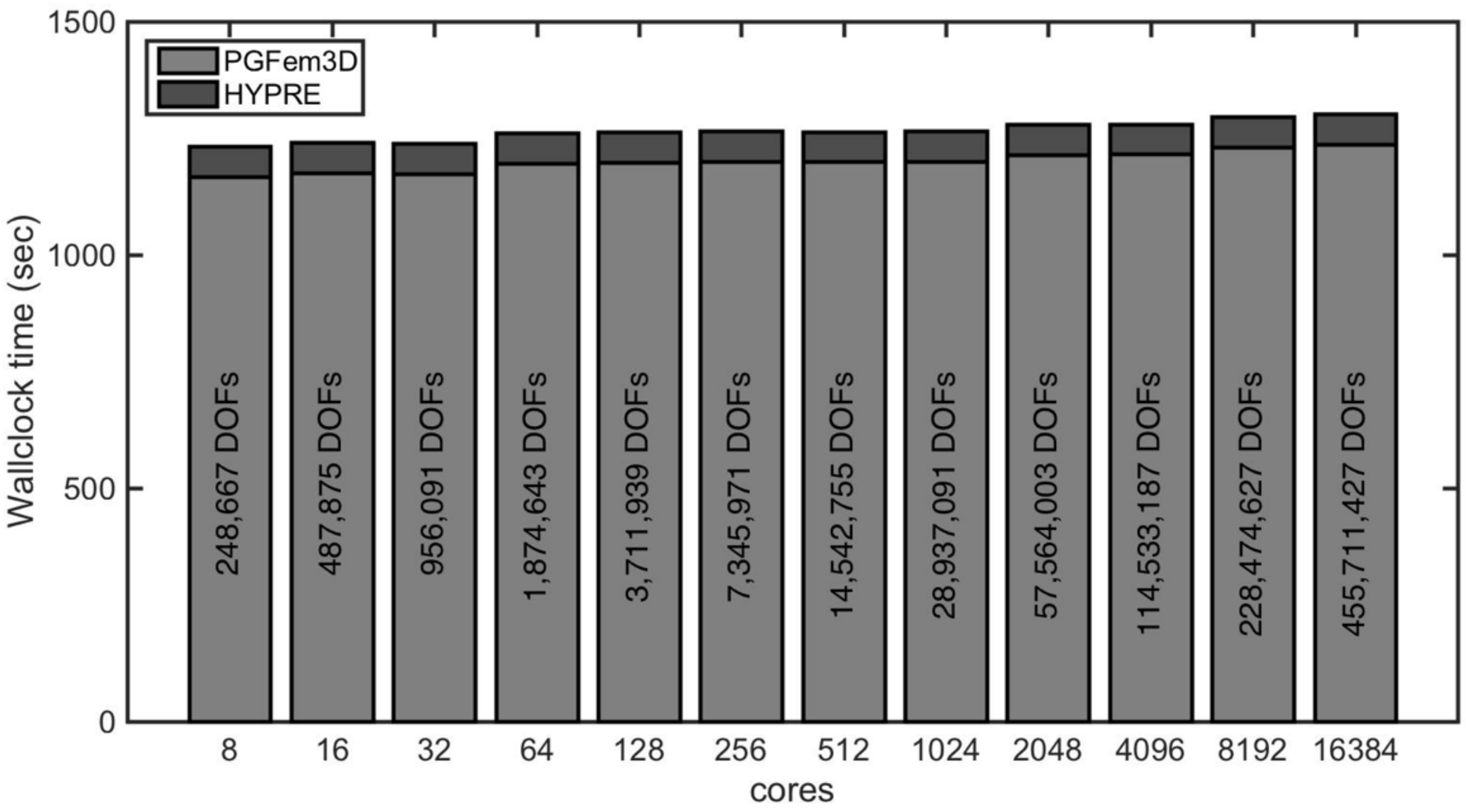
$$ {}^1\mathcal{R} = \frac{l_c}{|\Theta_0|}\int_{\Theta_0} {}^1\boldsymbol{P} : \nabla_{\boldsymbol{Y}}(\delta^1\boldsymbol{u})\,\mathrm{dV} = 0 $$

$$ {}^{[\![{}^0\boldsymbol{u}]\!]}\mathcal{R} = \left({}^0\boldsymbol{N}\cdot\frac{1}{|\Theta_0|}\int_{\Theta_0} {}^1\boldsymbol{P}\,\mathrm{dV} - {}^0\boldsymbol{t}\right)\cdot[\![\delta({}^0\boldsymbol{u})]\!] = 0 $$

PGFem3D solver

- Highly scalable finite strains *PGFem3D* solver
- Multiscale FE^2 capability
- Quasi-steady or transient analysis
- Complex constitutive Eq.



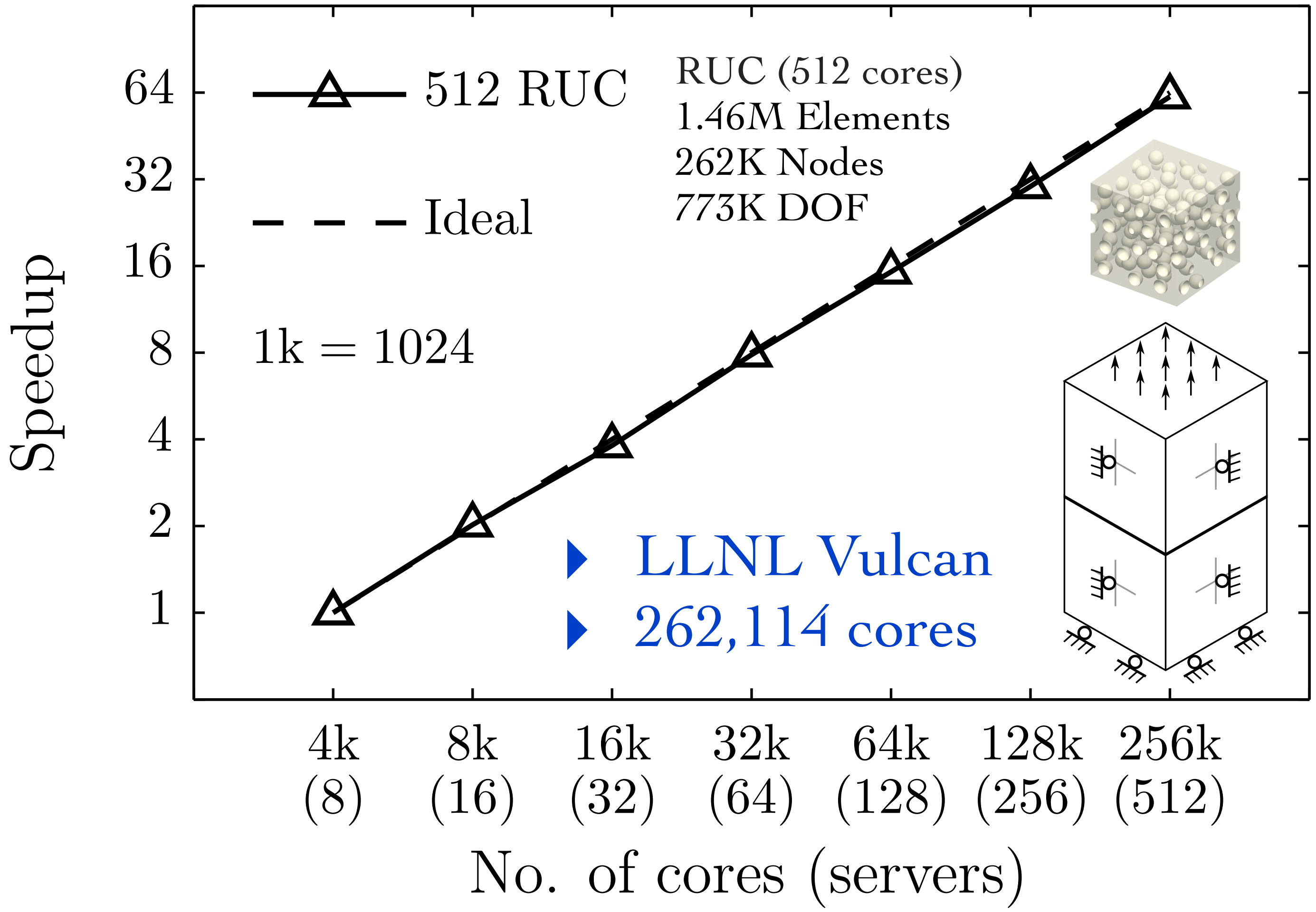


- N_e = 123,168,768
- N_n = 28,366,848
- 10 Nonlinear time steps
- 4 Linear iterations

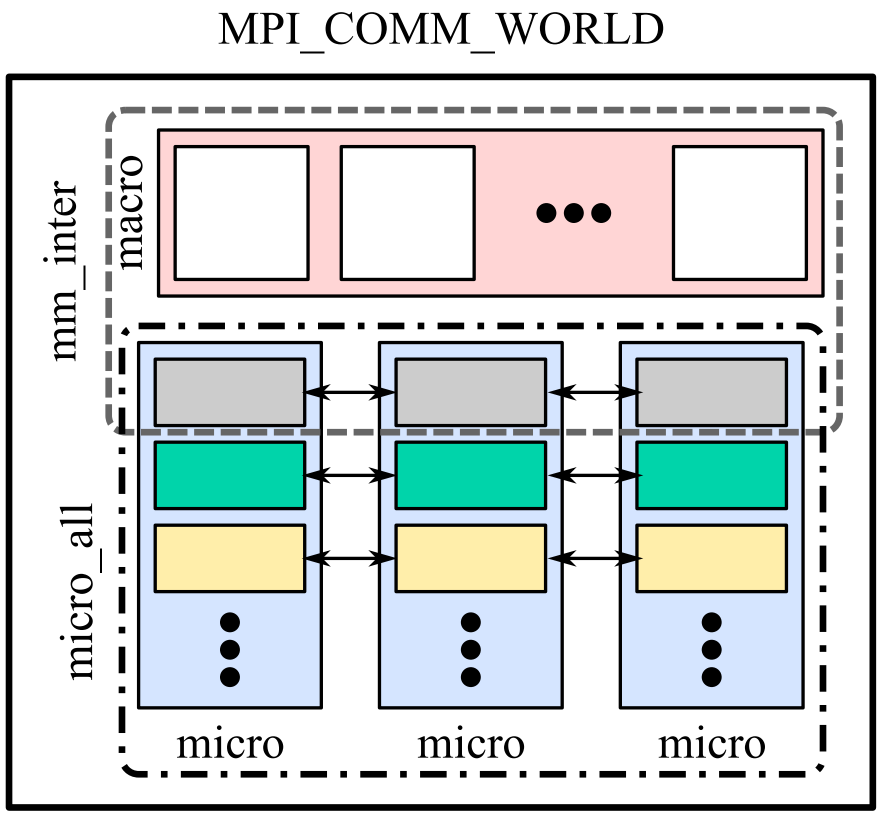
Δt = 1 μs v = 1 m/s

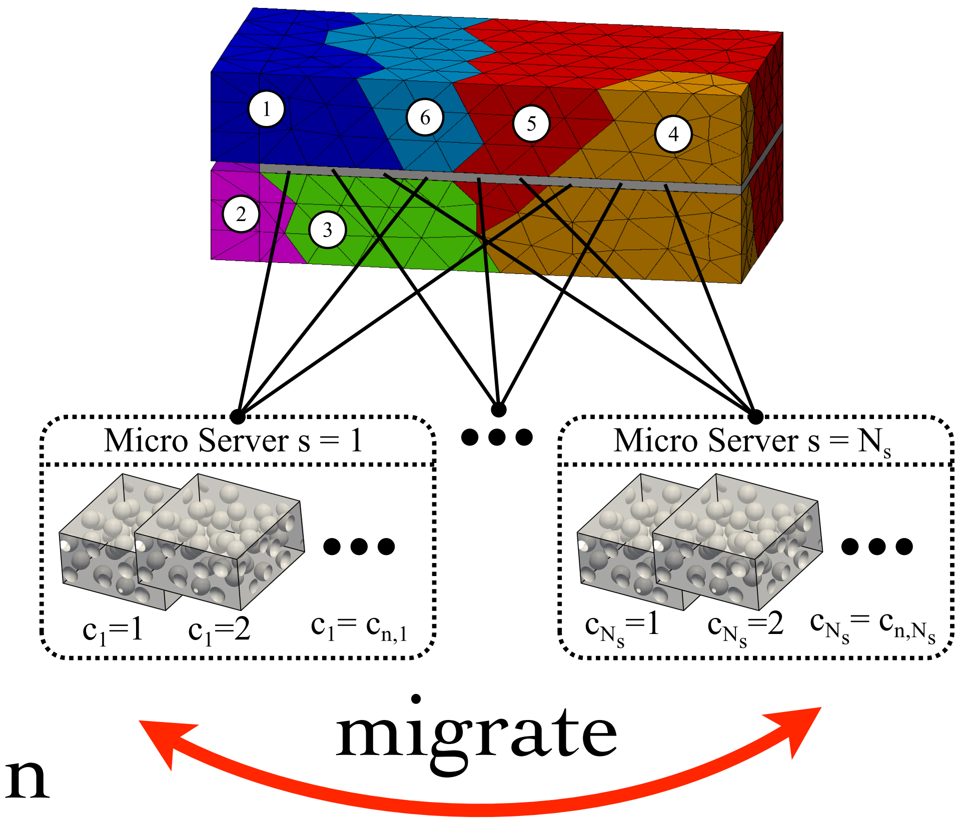
UNIVERSITY OF NOTRE DAME

Multi-scale Simulations, PGFem3D - GCTH



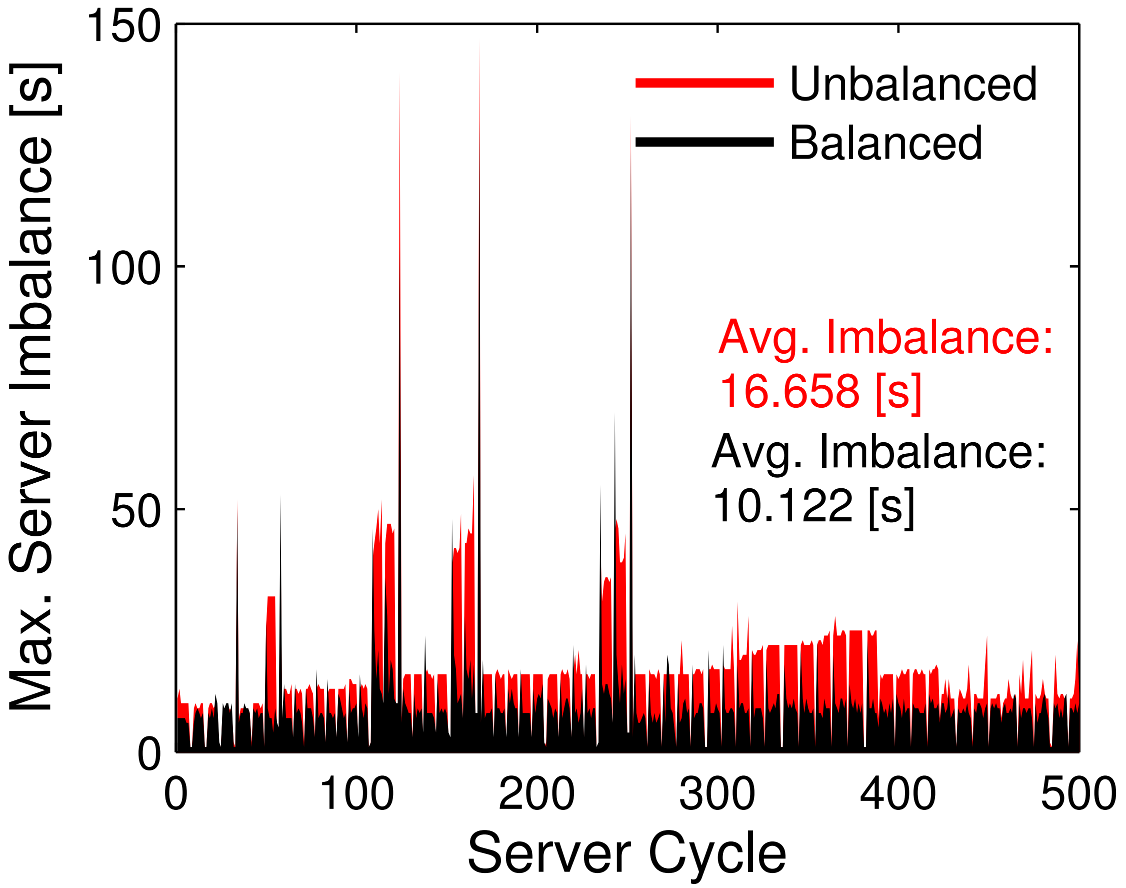
- LLNL Vulcan
- 262,114 cores

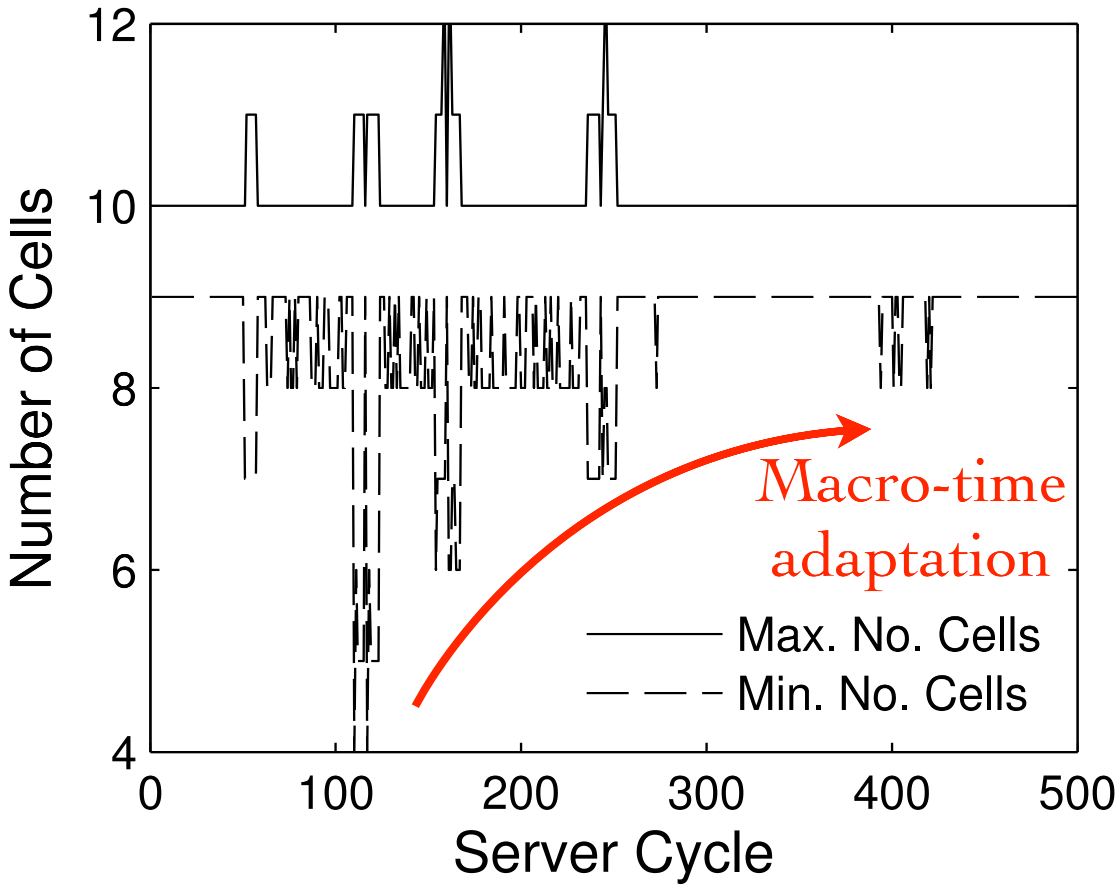




- Load-balancing for microscale simulations
  - Time-based metrics for adaptation, load-balancing heuristics
  - Non-blocking data migration among servers/computing nodes
  - Overlay computations with data migration

Micro-server Rebalancing





- Accounts for physics-induced imbalance
- Utilizes resources better
- Faster time-to-solution
- Highly-scalable

- C-SWARM, 1136 cores
- 16 Clients, 35 micro-servers
- 32 cores per micro-server

- 12% speedup when rebalancing the micro servers

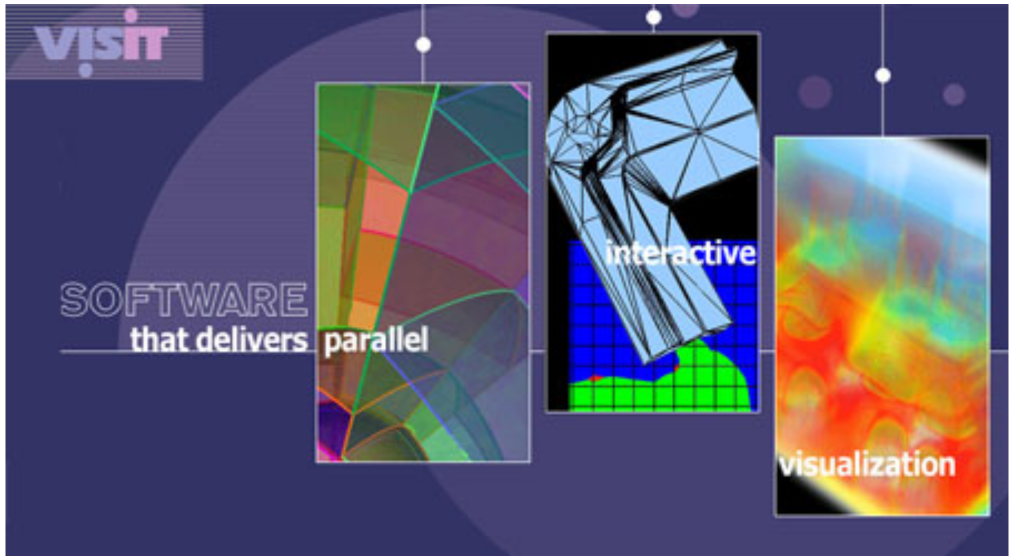
Numerical Libraries

- PETSc – Portable, Extensible Toolkit for Scientific Computation (www.mcs.anl.gov/petsc/)
- HYPRE – library for solving large, sparse linear systems of equations on massively parallel computers (computation.llnl.gov/project/linear_solvers/index.php)
- ScaLAPACK – Scalable Linear Algebra PACKage (www.netlib.org/scalapack/)
- LAMMPS – Large-scale Atomic/Molecular Massively Parallel Simulator (http://lammps.sandia.gov)
- Trilinos – large-scale, complex multi-physics engineering and scientific problems (https://trilinos.org)

Parallel Visualization & Data Analysis

ParaView

The ParaView project started in 2000 as a collaborative effort between Kitware Inc. and Los Alamos National Laboratory. (http://www.paraview.org)



VisIt was originally developed by the Department of Energy (DOE) Advanced Simulation and Computing Initiative (ASCI) to visualize and analyze the results of terascale simulations. (wci.llnl.gov/simulation/computer-codes/visit)

Debugging and Profiling

- TAU – profiling and tracing toolkit for performance analysis of parallel programs (www.cs.uoregon.edu/research/tau/home.php)
- APEX – Autonomic Performance Environment for eXascale (www.nic.uoregon.edu/~khuck/apex_docs/doc/html/index.html)
- Allinea DDT – Parallel debugger (www.allinea.com/products/ddt)
- TotalView – Parallel debugger (www.roguewave.com/products-services/totalview)

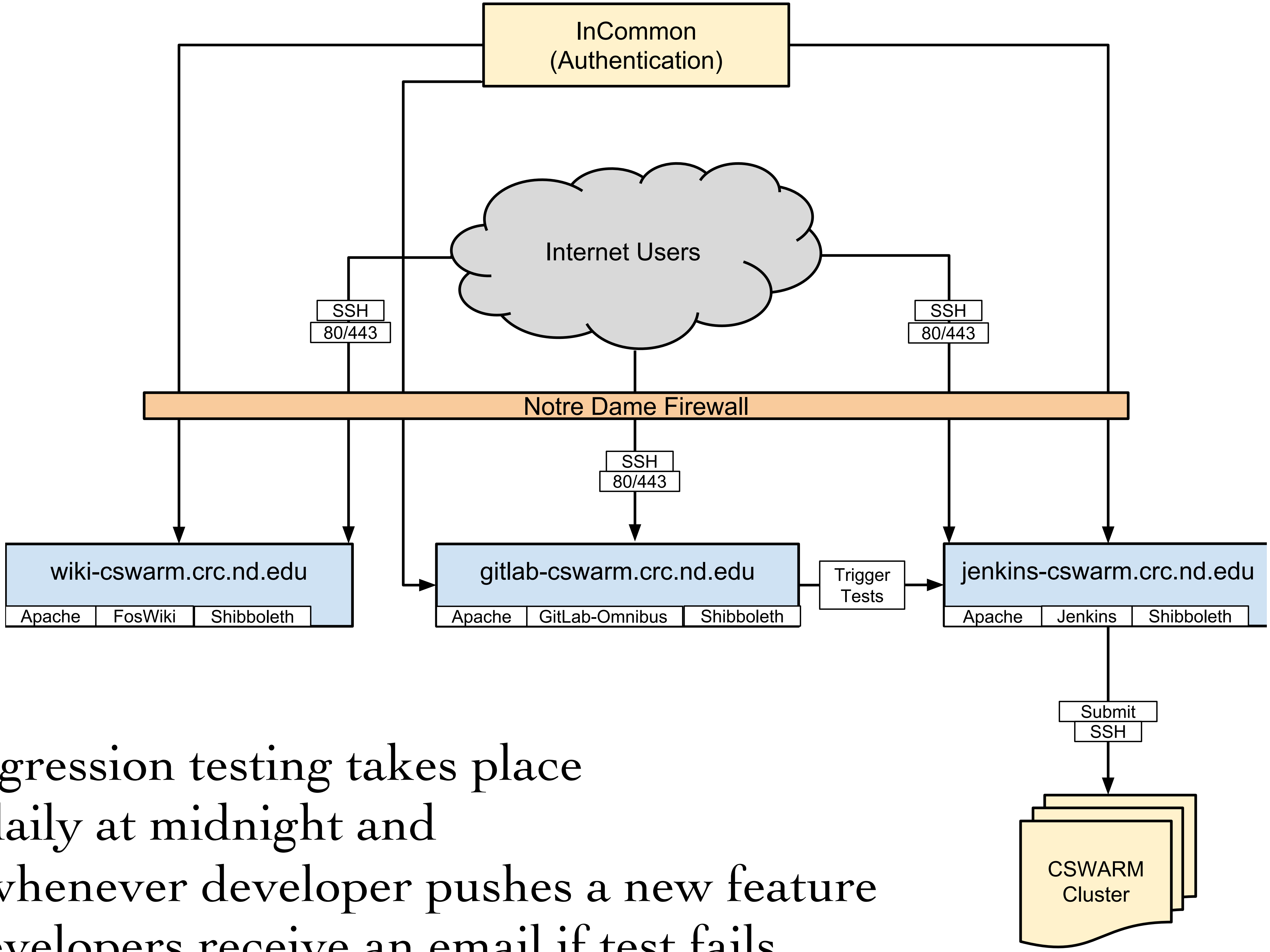
Software Engineering Tools

- Source Code and Data Control
  - C-SWARM maintained through wiki tracking system
  - Open-source control system GitLab
- Automated Builds, Full-System, and Regression Testing
  - Frequent builds on C-SWARM cluster, Jenkins
- Maintain Third-party Software
  - HYPRE, ParMETIS, ParaView, TAU, etc.
- Documentation, Release, and Support
  - Doxygen documentation system
  - PDF User's and Developer's guides

Gitlab and Jenkins

- Repository manager with wiki and issue tracking features (gitlab.com)
  - repository management, code reviews, issue tracking, activity feeds and wikis, code statistics
- Continuous integration tool (jenkins.io)
  - Builds can be started by various means, including being triggered by commit in a version control system (gitlab), by scheduling

Software Engineering Tools



- Regression testing takes place
  - daily at midnight and
  - whenever developer pushes a new feature
- Developers receive an email if test fails

Conclusions

- High performance computing is here to stay
- Need in initial investment
- Use existing libraries
- Software engineering is important component
- It is never to late to start

- Thanks to Colleagues, Research Staff, Students
- Kokkos Documentations, Dr. H. Carter Edwards

- Material Taken from:

- U.C. Berkeley CS267, Applications of Parallel Computers, Dr. James Demmel

Karel Matouš

College of Engineering Collegiate Associate Professor of Computational Mechanics
Director of Center for Shock-Wave Processing of Advanced Reactive Materials

Department of Aerospace & Mechanical Engineering
University of Notre Dame

367 Fitzpatrick Hall of Engineering
Notre Dame, IN 46556
Email: kmatous@nd.edu
www.nd.edu/~kmatous
www.cswarm.nd.edu