Keynote: Tackling Real World Problems with Mathematics and High-Performance Computing

Alice Koniges, University of Hawai‘i
Graduate Faculty Information and Computer Sciences and
Hawai‘i Data Science Institute, datasci@hawaii.edu

INFORMATION & COMPUTER SCIENCES
UNIVERSITY of HAWAI'I at MĀNOA

HAWAI‘I DATA SCIENCE

The Seventeenth International Conference on Advanced Engineering Computing and Applications in Sciences
ADVCOMP 2023

September 25 to 29, 2023 - Porto, Portugal

Dr. Alice Koniges, graduate faculty in Information and Computer Sciences at the University of Hawai'I, serves as computer scientist and research principal investigator for the Hawai'i Data Science Institute, a system-wide effort to support data science education, collaborative research and partnerships with industry. Koniges leads efforts in scientific computing with an emphasis on complex physics and mathematical models using differential equations, HPC performance modeling, and ML/DL/AI applications. She is associate editor of the International Journal of High Performance Computing Applications, and has published well over 100 refereed papers in both applications and parallel computing/programming languages, and has two books, "Industrial Strength Parallel Computing" and "OpenMP Common Core." Her PhD is from Princeton University in Mathematical Astrophysics, and she also has an MSE in Mechanical Engineering.

Alice smashing the Covid-19 virus

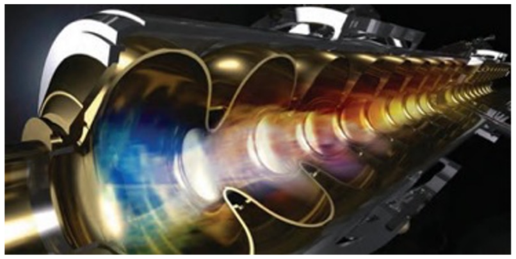
Koniges (PI) current grants/applications using the PISALE code
See our website, pisale.bitbucket.io for more information

Elements: ALE-AMR Framework and the PISALE Codebase

This project will apply the code for simulations of complex groundwater flow processes in Hawaiian islands characterized by highly heterogeneous volcanic rocks and dynamic interaction between freshwater and seawater.

An Extensible High Energy Density Modeling Tool for Extreme Regime

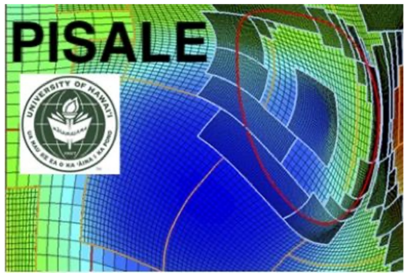
High Energy Density (HED) Physics implies the study of systems at very high pressures and temperatures. Our simulations will address a critical need to understand the interaction between HED material and surrounding liquid material for experiments at the X-ray Free Electron Laser (XFEL) located at the Stanford Linear Accelerator Center.

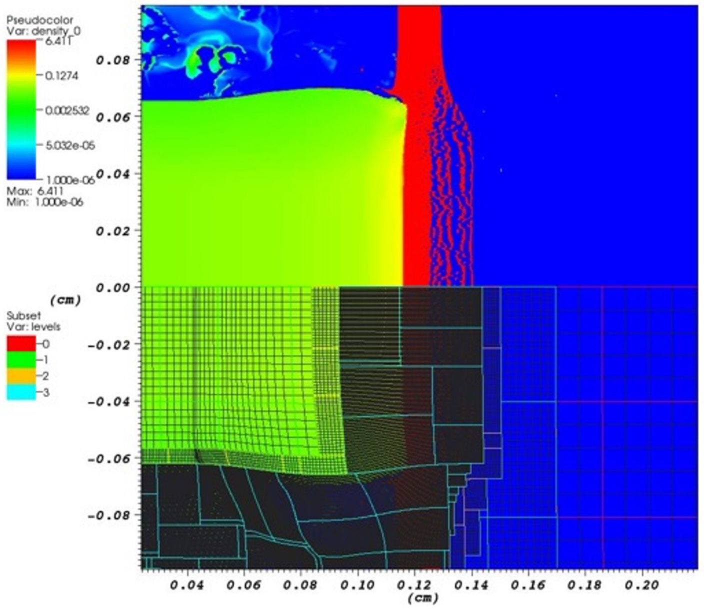


MURI: Faster than the speed of sound

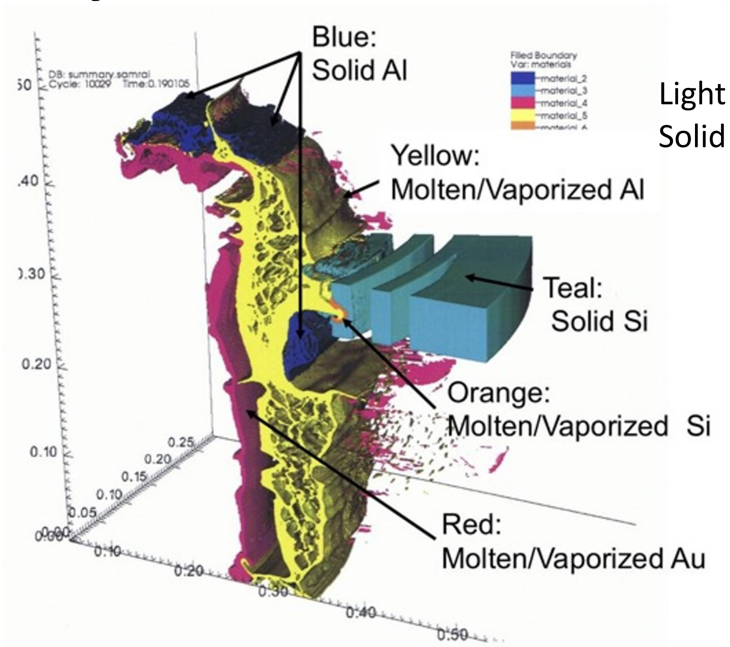
We use the code to study effects of rain, ice, and aerosols on hypersonic vehicles. This is a multi-university effort led by the University of Minnesota entitled Particulate and Precipitation Effects on High-speed Flight Vehicles.

PISALE (P̲acific I̲sland S̲tructured-AMR with A̲L̲E̲) combines Adaptive Mesh Refinement with Arbitrary Lagrangian Eulerian hydrodynamics to create a PDE framework written in C++, Python and F90

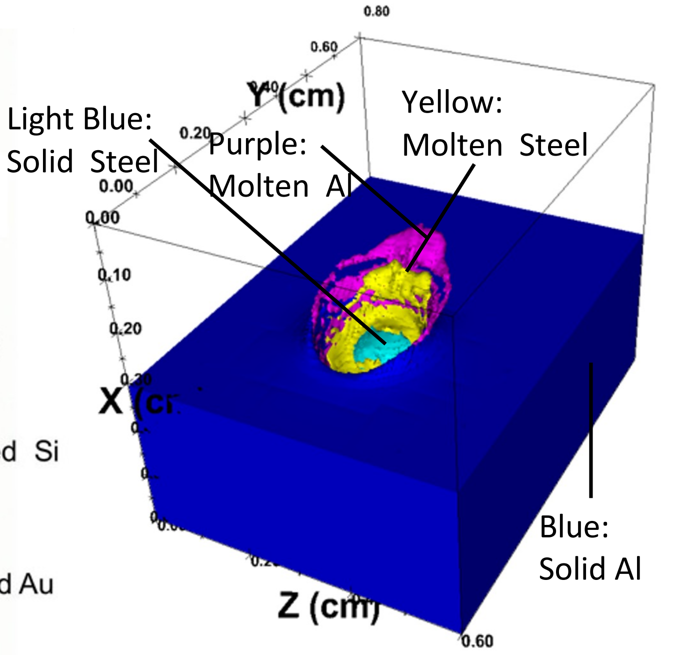




Laser heated plate with blowoff and rear spall. Mesh using AMR is shown in lower half of image.



A complex multimaterial simulation with use of void to create fragments. Model 1/32 of target.



Impact of steel sphere at 45^{o} on Al plate.

PISALE Solves Partial Differential Equations

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \vec{v}) = 0$$

Continuity equation

$$\rho \frac{\partial \vec{v}}{\partial t} = \nabla p + \nabla \bullet \Sigma' + \rho \vec{b}$$

Equations of motion

$$\rho \frac{\partial e}{\partial t} + p \nabla \bullet \vec{v} = 0$$

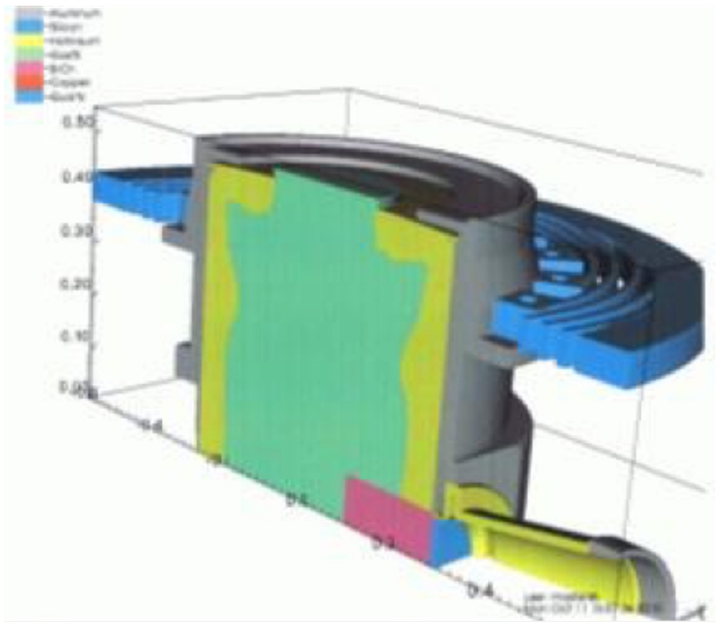
PdV work

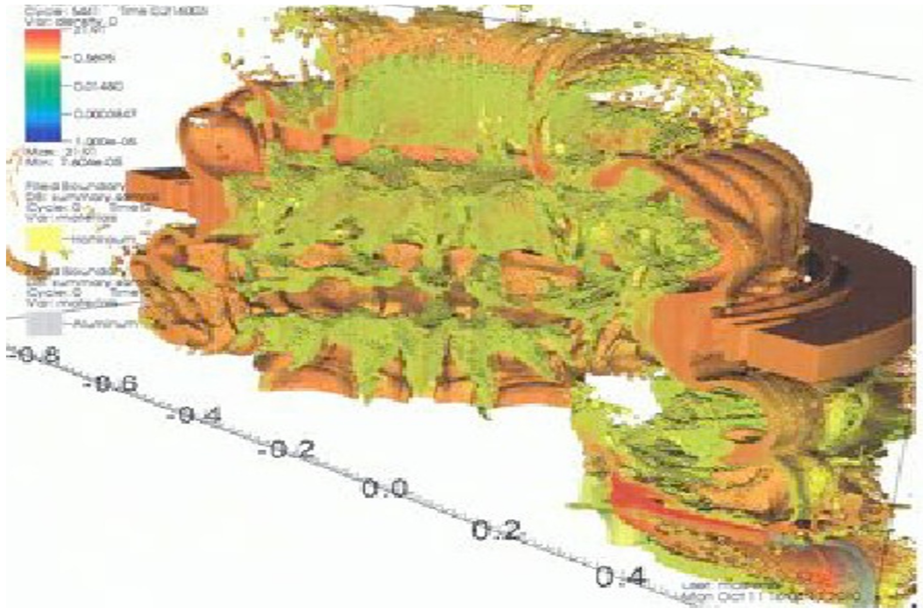
$$\sum^{n+1} = f(\Sigma^n, \rho, e, \vec{v}, p, T, \vec{h})$$

Material Stress Update

Partial differential equations are used to mathematically formulate problems involving functions of several variables, and describe physical phenomena such as fluid flows, electricity and magnetism, behavior of stars, propagation of virus particles

PISALE generates "Big Data" from simulations where zonal quantities, e.g., temperature, density, velocity, etc., are saved as a function of time in 3 dimensions





Students can use HPC supercomputers on their campus and at DOE and NSF centers through funded grants

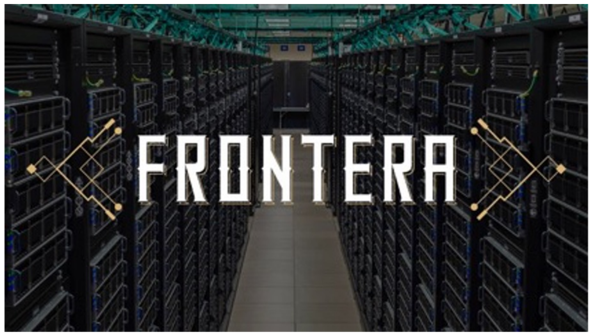
UH

345 compute nodes
120 GPUs
8,452 cores
62.9 TB of memory
1 PB of long term storage
61 TB of flash scratch storage
150 TB of standard scratch storage

NERSC - DOE

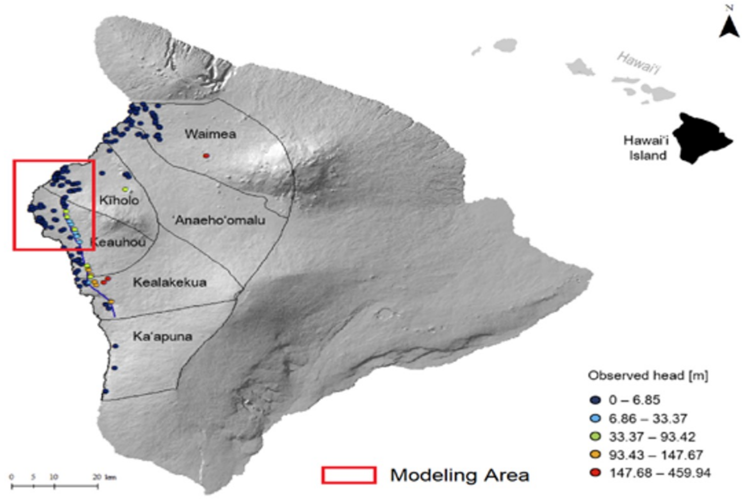
9,600 Intel Xeon Phi “KNL” manycore nodes
2,000 Intel Xeon “Haswell” nodes
700K cores, 1.2 PB memory
Cray XC40 / Aries interconnect
30 PB scratch file system
1.5 PB Burst Buffer @ 1.5 TB/s

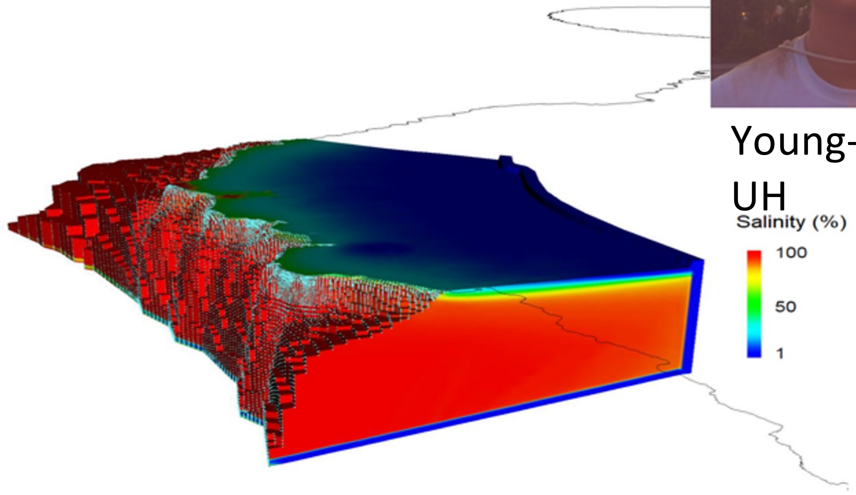
TACC - NSF

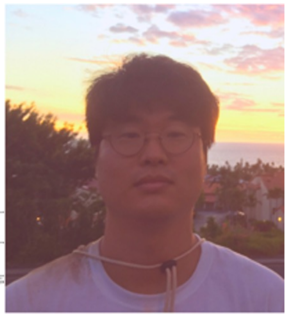


8008 compute nodes
Intel Xeon Platinum 8280
448,448 cores
DDR-4 memory, 192 GB/node
Local disk: 480 GB SSD drive
Network: Mellanox InfiniBand

NSF Elements: New Application for PISALE with Engineering Prof. Jonghyun Harry Lee





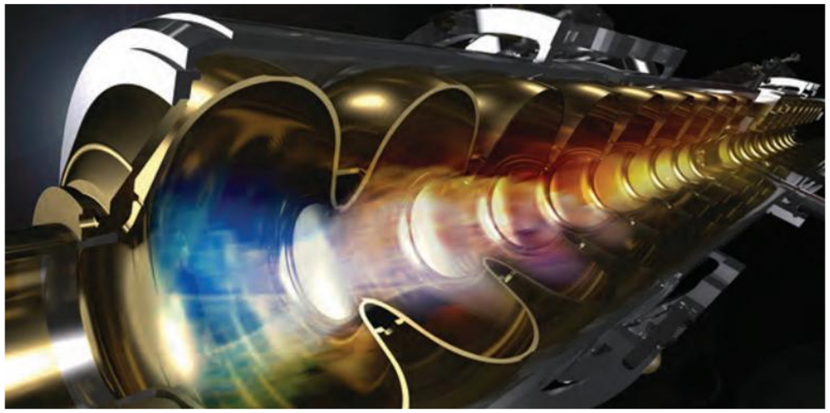


Young-Ho, UH

- Simulations of complex groundwater flow processes in Hawaiian islands characterized by highly heterogenous volcanic rocks.
- Most widely used Eulerian-based solvers cannot reproduce sharp freshwater-seawater interface observed in several monitoring locations.
- Water resources managers want to identify island-scale groundwater distribution, which have not been feasible due to poor scalability of conventional solvers.

DOE: An Extensible High Energy Density Modeling Tool for Extreme (HED) Regimes

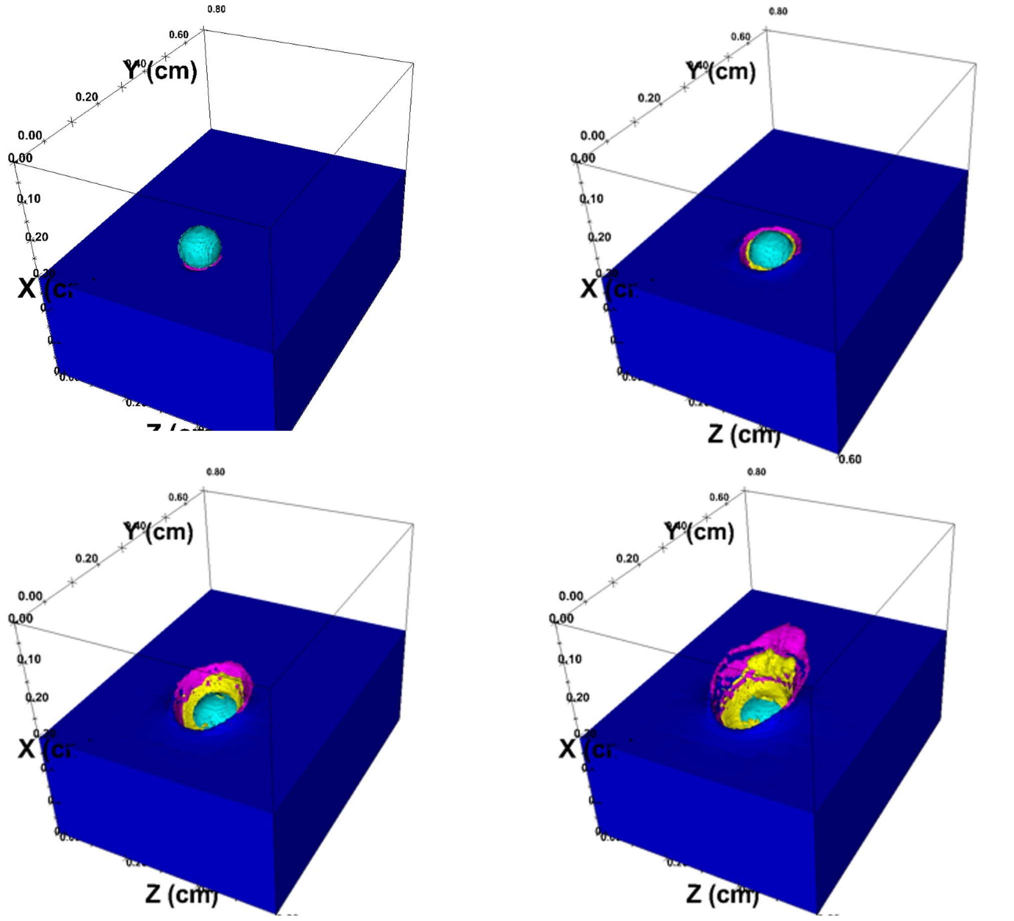
Claudia, Stanford



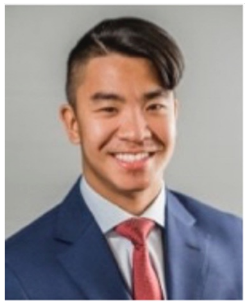
X-ray light sources are powerful tools for structural biology and can provide insight into viruses such as SARS-CoV-2 that may help to develop therapies to treat the disease.

- **The X-ray Free Electron Laser at Stanford will be capable of million shots/sec**
- **X-ray pulses interact with a series of droplets of various materials**
- **Used to study nanoscale materials dynamics, real-time biological reactions, quantum materials, HED physics, and chemical dynamics (via spectroscopy)**
- **PISALE's droplet dynamics models critical to allow operation at high rep rate**

DoD ONR MURI: Using PISALE to model rain impacts on hypersonic vehicles



Jack, UH

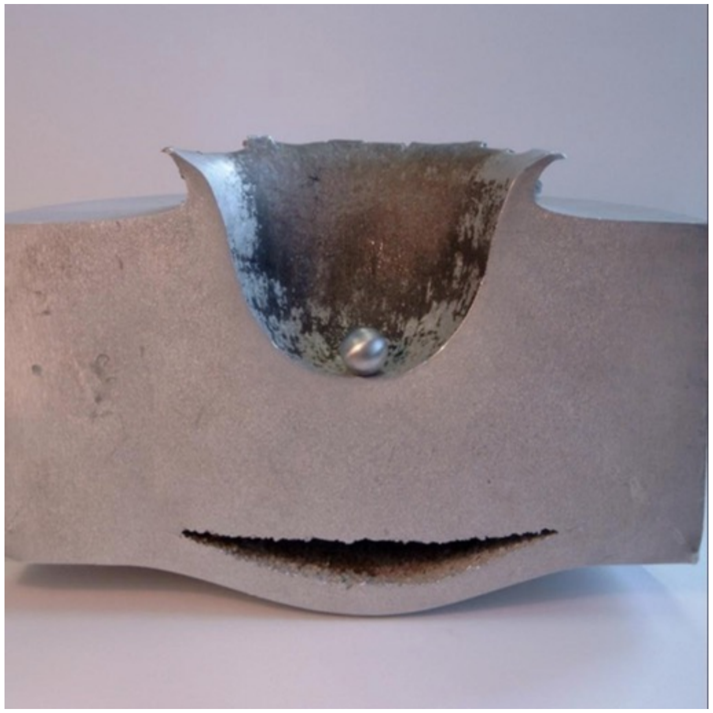

Peter, UMN

- Hypersonic is speeds of Mach 5 and above
- Impacts of rain can roughen and remove surface material affecting flow
- The work with U of MN and two other universities involves multiple professors, postdocs, and students doing modeling and experiments for the 5-year grant

Recent Experiments by European Space Agency Highlight Importance of Hydrocode Modeling

ESA space debris studies: hypervelocity impact sample

Photo Published on Twitter 12/4/21 (~2013 experiment)

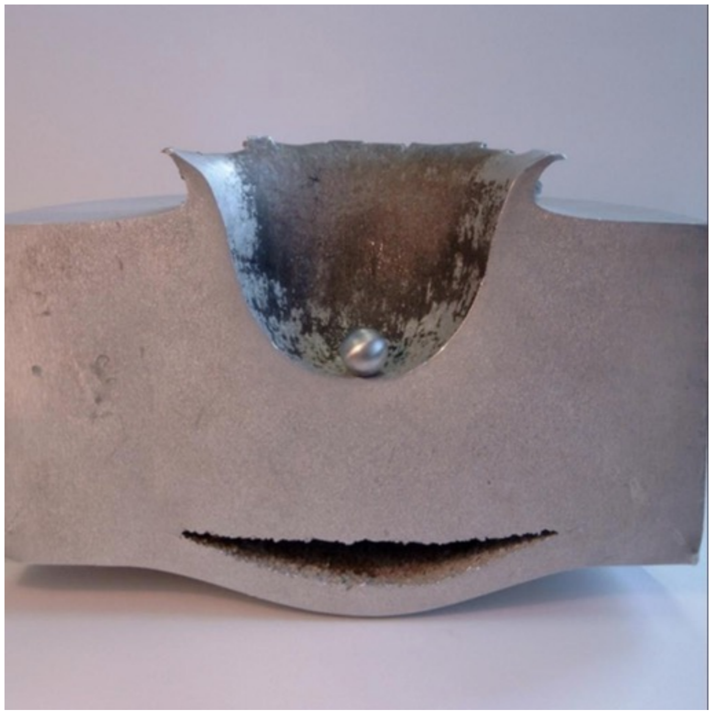


Destructive energy is a consequence of high impact velocities

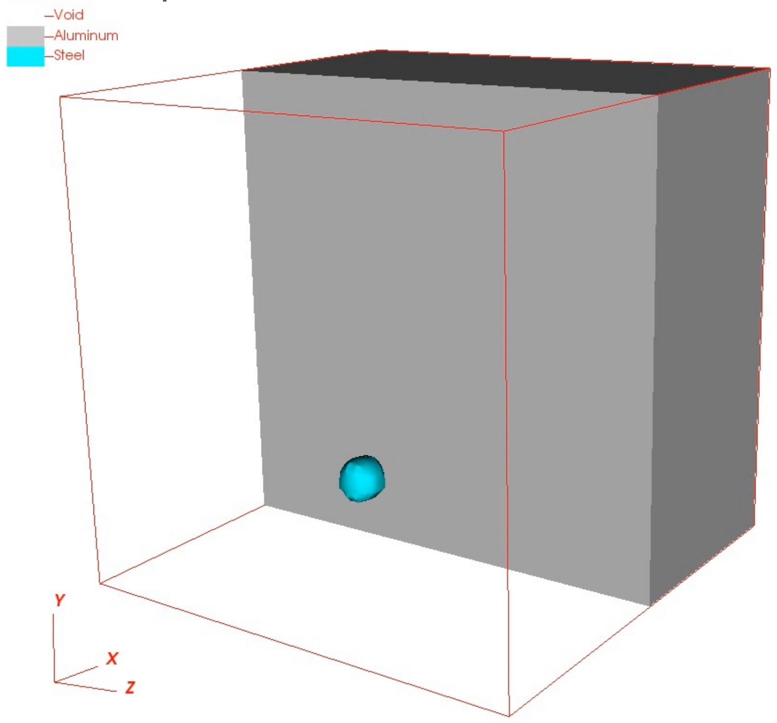
- The effects of hypervelocity impacts are a function of projectile and target material, impact velocity, incident angle and the mass and shape of the projectile.
- Beyond 4 km/s (depending on the materials), an impact will lead to a complete breakup and melting of the projectile. Typical impact velocities are around 14 km/s for space debris, and significantly higher for meteroids.
- At low velocities, plastic deformation normally prevails. With increasing velocity the impactor will leave a crater on the target. Beyond 4 km/s (depending on the materials), an impact will lead to a complete breakup and melting of the projectile, and an ejection of crater material to a depth of typically 2–5 times the diameter of the projectile.

PISALE gives capability to accurately model such phenomena (RHS is different params)

ESA space debris studies: hypervelocity impact sample
Photo Published on Twitter 12/4/21 (~2013 experiment)



Steel ball at 5km/s hitting Aluminum plate at 45 degrees angle. Approximately 3h run time with 32 Haswell processors on Cori.



New (more portable) build system enables PISALE installation on smaller clusters

- New build system:
  - Comprehensive documentation (in progress)
  - No hardcoded library paths
  - Using standard GNU Make and POSIX shell commands
  - Configure script and makefiles are tested and working
- Planned/in-progress work:
  - Consider SPACK build system
  - More comprehensive unit tests and test suites
    - Testing on systems besides Cori (TACC, Univ. clusters, UH Mana)
  - Integrate with current and future SAMRAI updates

Updating to latest SAMARI

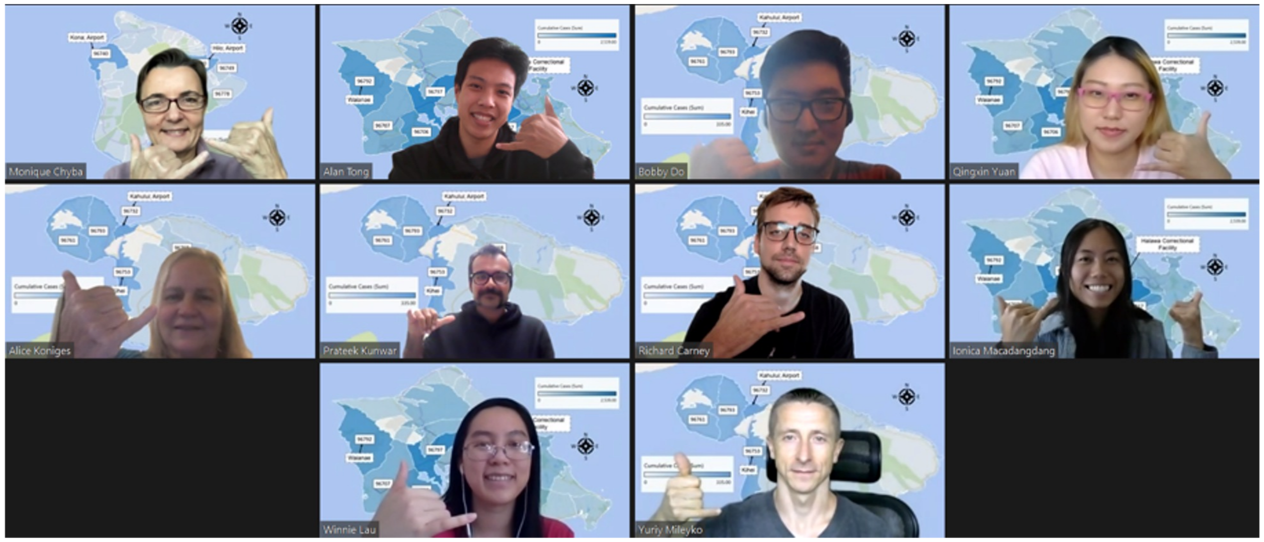
The current PISALE code uses SAMRAI v 3.2.

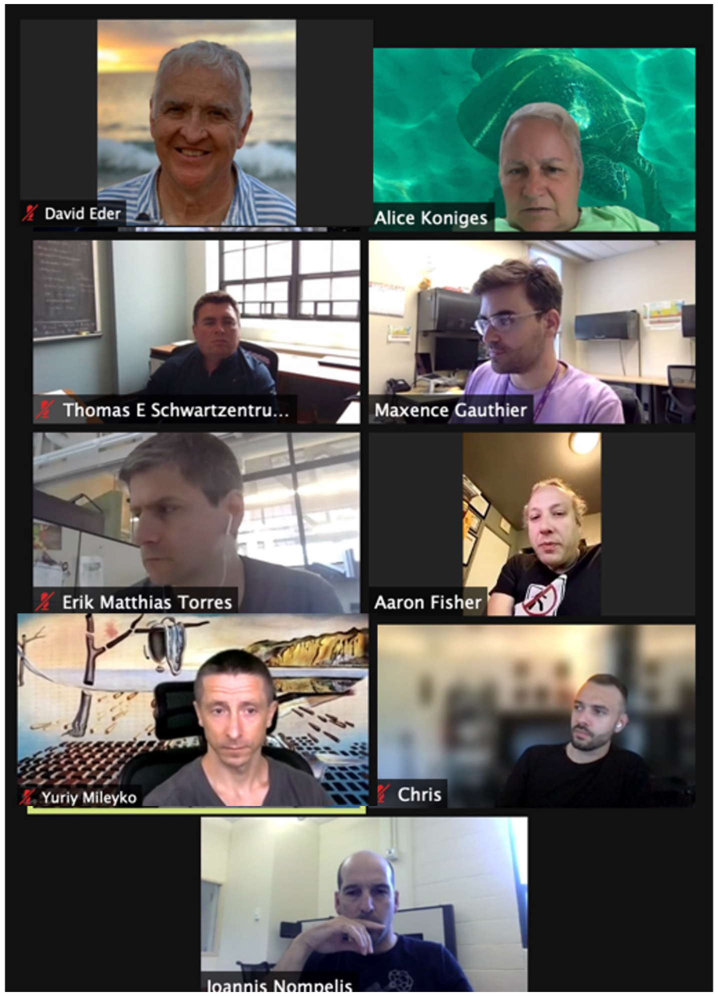
- SAMRAI v 3.4 saw a significant code overhaul that touched most of the files:
  - The outdated smart pointer class has been replaced with boost libs shared_ptr
  - Iterator classes have been modified
  - GridGeometry class hierarchy has been changed
- Most of the needed modifications to the PISALE code have been done:
  - The frequent use of smart pointers has been updated to work with shared_ptr
  - The omnipresent loops utilizing iterators have been re-written
  - Many other needed code changes have been implemented
  - A few more needed changes are pending

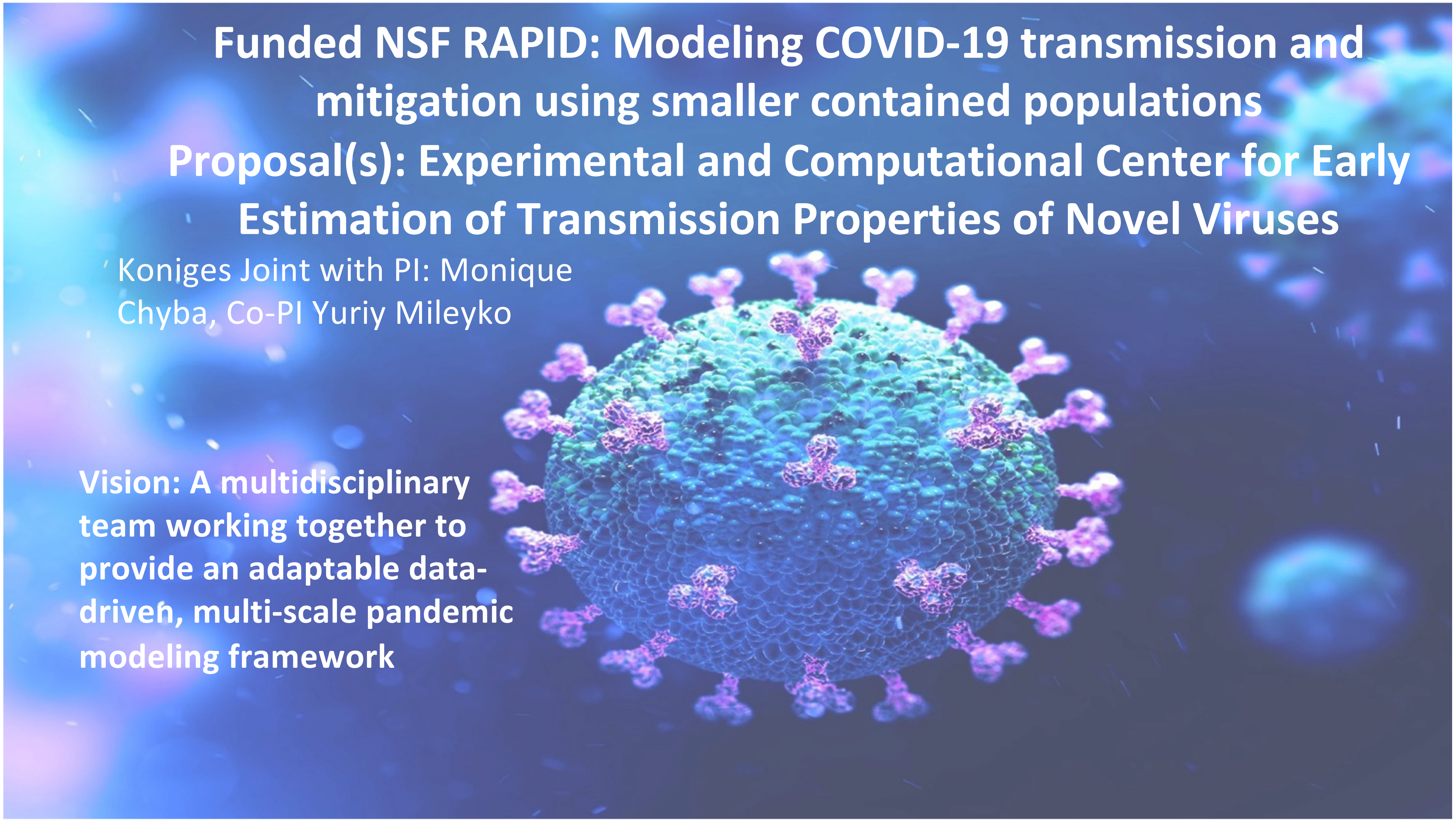
Updating to latest SAMARI (Cont)

- The most recent version of SAMRAI is 4.1, with versions 3.5 through 3.15 and 4.0 in between
- Planned near future work:
  - Finish update for SAMRAI v 3.4
  - Develop a suite of test problems exercising various combinations of PSALE features like 2D/3D, ALE, AMR
  - Integrate with the new build system
- Planned further future work:
  - Update PISALE code to work with the latest SAMRAI version
  - Thoroughly test the new implementation with unit tests and the new test suite

Pandemic Team (LHS) and PISALE Team (RHS) meet regularly on zoom (not all members shown)







Funded NSF RAPID: Modeling COVID-19 transmission and mitigation using smaller contained populations

Proposal(s): Experimental and Computational Center for Early Estimation of Transmission Properties of Novel Viruses

Koniges Joint with PI: Monique Chyba, Co-PI Yuriy Mileyko

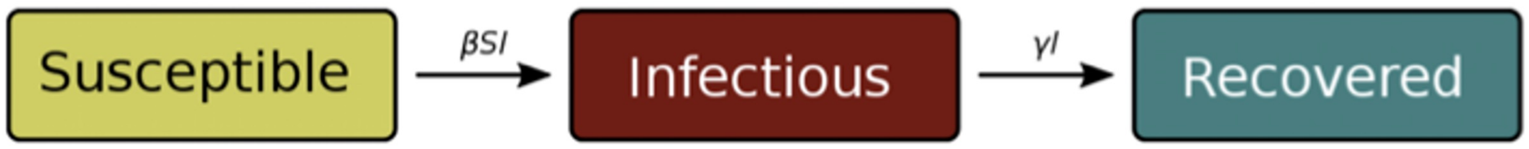
Vision: A multidisciplinary team working together to provide an adaptable data-driven, multi-scale pandemic modeling framework

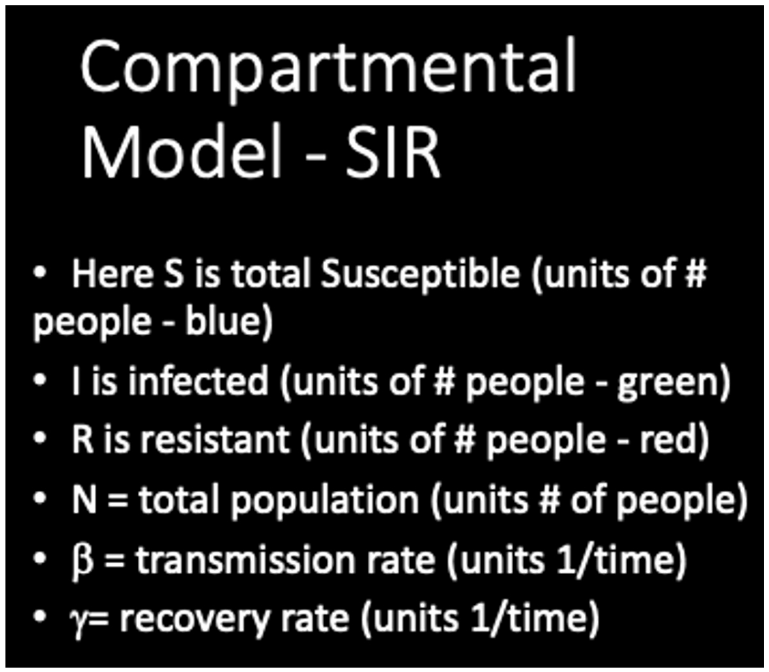
Two Different but Complementary Methods are used for Prediction and Understanding of Disease Spread

- There are primarily two different types of epidemiological models: differential equation-based (EBM) and agent based (ABM).
- We use two such models:
  - A discrete compartmentalized SEIR (Susceptible-Exposed-Infectious-Recovered) model that we developed (Mileyko, et al.)
  - COVID-Agent-based Simulator (Covasim) “Covasim: An agent-based model of COVID-19 dynamics and interventions,” (Kerr, et al. 2021) that we altered to include some specific attributes.
- These epidemiological models estimate the spread of the infectious disease across a population based on some core epidemiology determinants: incubation period, duration of infectious period, population size, and R0 (the reproductive value).
- Our specific implementations of these two models use more variables and complex interactions

Details: Kunwar, P., Markovichenko, O., Chyba, M., Mileyko, Y., Koniges, A., & Lee, T. (2021). A study of computational and conceptual complexities of compartment and agent based models. *arXiv preprint arXiv:2108.11546*.

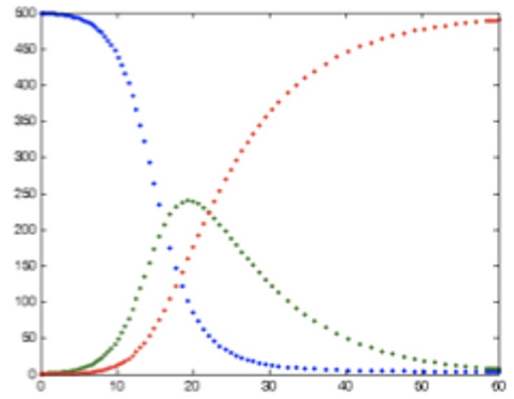
Simplest Compartmental Model





Compartmental Model - SIR

- Here S is total Susceptible (units of # people - blue)
- I is infected (units of # people - green)
- R is resistant (units of # people - red)
- N = total population (units # of people)
- β = transmission rate (units 1/time)
- γ= recovery rate (units 1/time)

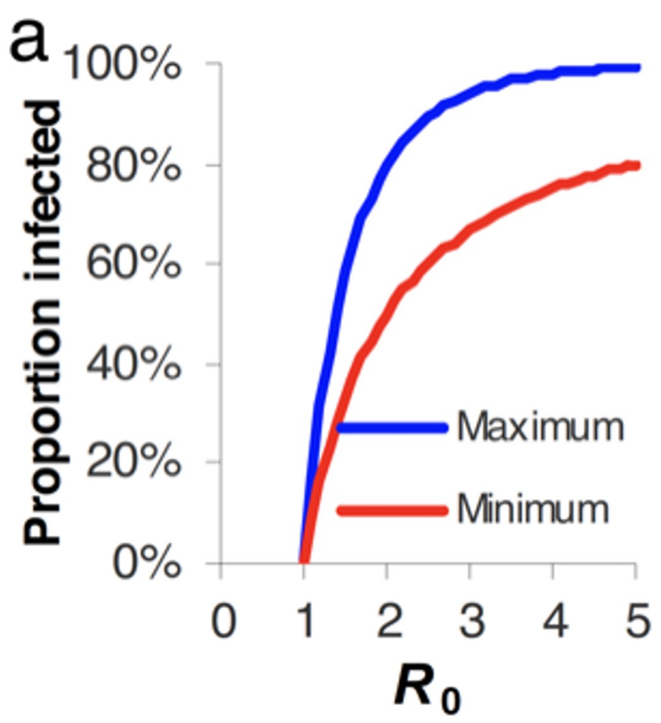


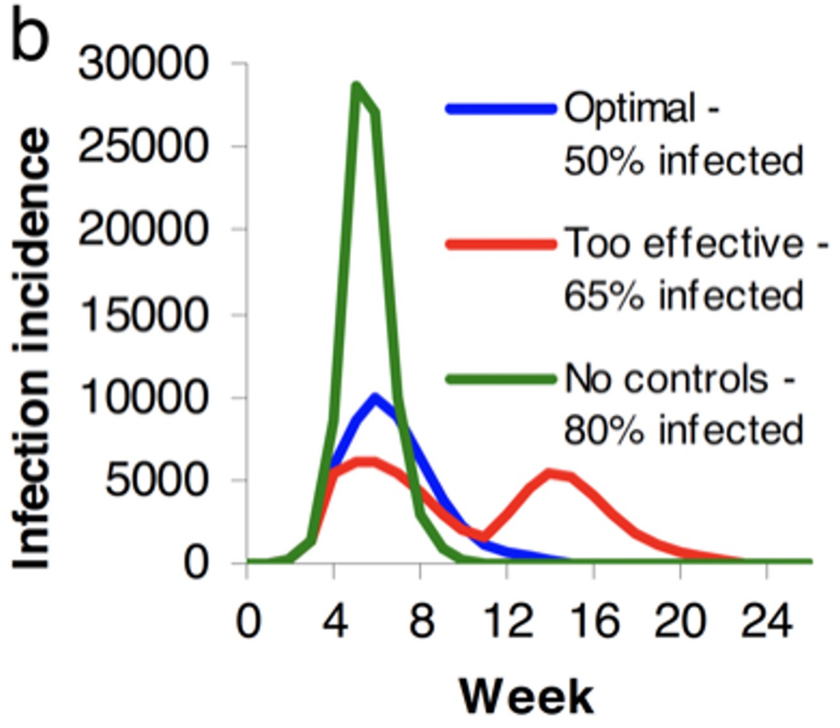
$$\frac{dS}{dt} = -\beta \frac{IS}{N}, \qquad \frac{dI}{dt} = \beta \frac{IS}{N} - \gamma I, \qquad \frac{dR}{dt} = \gamma I,$$

$R_0 = \beta/\gamma$ is dimensionless = reproduction number

This is an important dimensionless number for epidemics

Effects of transient imperfect health interventions on epidemic dynamics. *Bootsma & Ferguson PNAS 2007*





SIR model: Total proportion of the population infected in an epidemic in the absence of controls or reactive contact reduction compared with the minimal proportion needing to be infected to achieve herd immunity and therefore stop transmission, shown as a function of R_0.

Weekly infection incidence over 6 months from a SIR model with 3.5-day infectious period, R_0=2, 100,000 population, two seed infections at time 0, and controls imposed from day 25. Green curve, no controls; red curve, overeffective controls that reduce R by 40% and stop on day 75 (leading to a second wave); blue curve, controls that reduce R by 32.5% and stop on day 110 (giving the minimal possible epidemic size).

SEIR model

$$\begin{tikzcd}\text{Susceptible} \arrow[r] & \text{Exposed} \arrow[r] & \text{Infectious} \arrow[r] & \text{Recovered}\end{tikzcd}$$

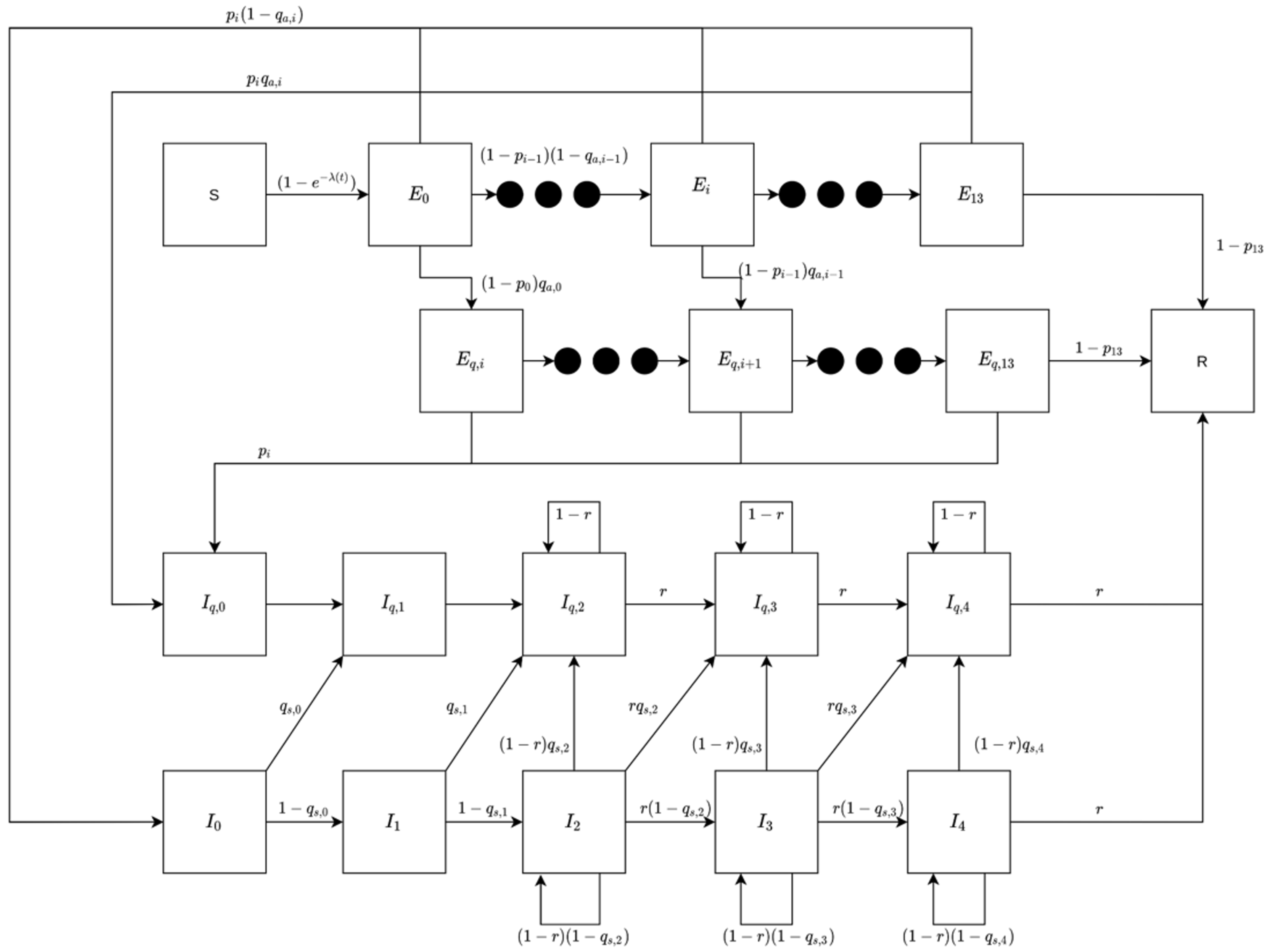
In its simplest form

$$\frac{dS}{dt} = -\beta \frac{IS}{N}, \qquad \frac{dE}{dt} = \beta \frac{IS}{N} - aE,$$

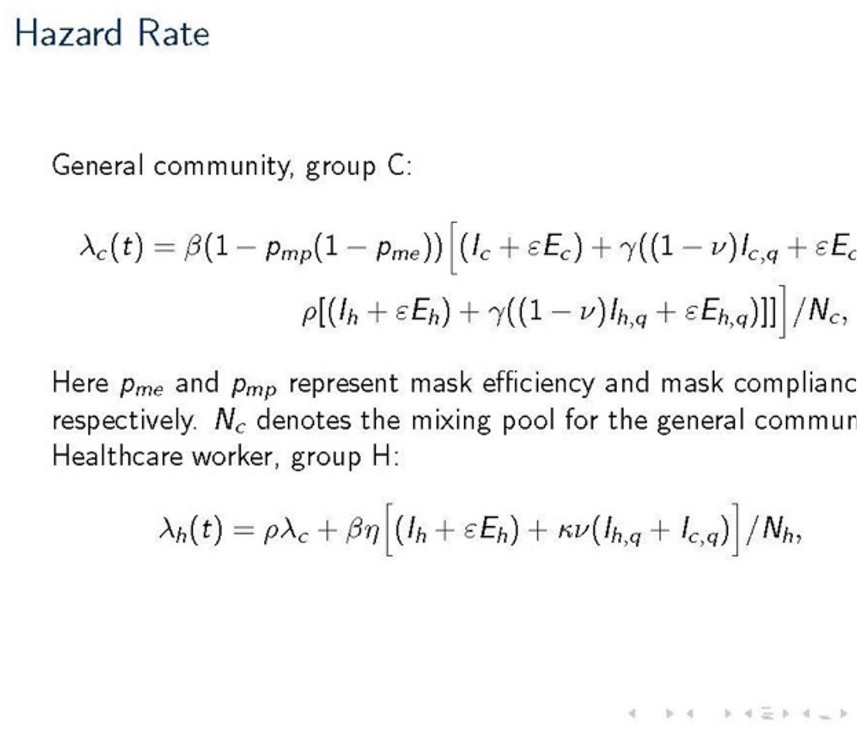
$$\frac{dI}{dt} = aE - \gamma I, \qquad \frac{dR}{dt} = \gamma I.$$

Note that by dividing the first equation by the last we have $dS/dR = -R_0\, S/N$ - same equation as for SIR. So long time dynamics of S and R will be the same for SEIR as for SIR.

Generalized SEIR



Equations



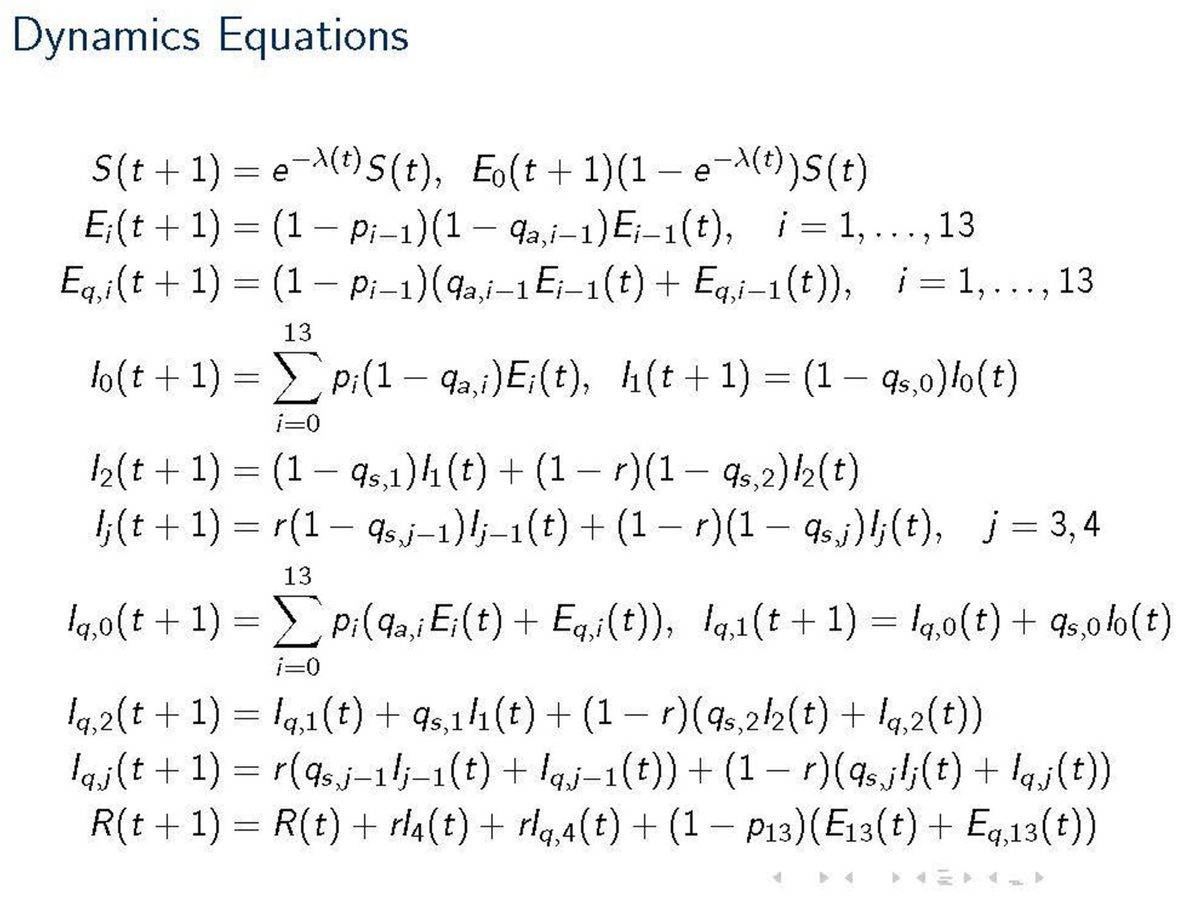
Hazard Rate

General community, group C:

$$\lambda_c(t) = \beta(1 - p_{mp}(1 - p_{me}))\Big[(I_c + \varepsilon E_c) + \gamma((1-\nu)I_{c,q} + \varepsilon E_c$$
$$\rho[(I_h + \varepsilon E_h) + \gamma((1-\nu)I_{h,q} + \varepsilon E_{h,q})]]\Big]/N_c,$$

Here p_{me} and p_{mp} represent mask efficiency and mask complianc
respectively. N_c denotes the mixing pool for the general commur
Healthcare worker, group H:

$$\lambda_h(t) = \rho\lambda_c + \beta\eta\Big[(I_h + \varepsilon E_h) + \kappa\nu(I_{h,q} + I_{c,q})\Big]/N_h,$$

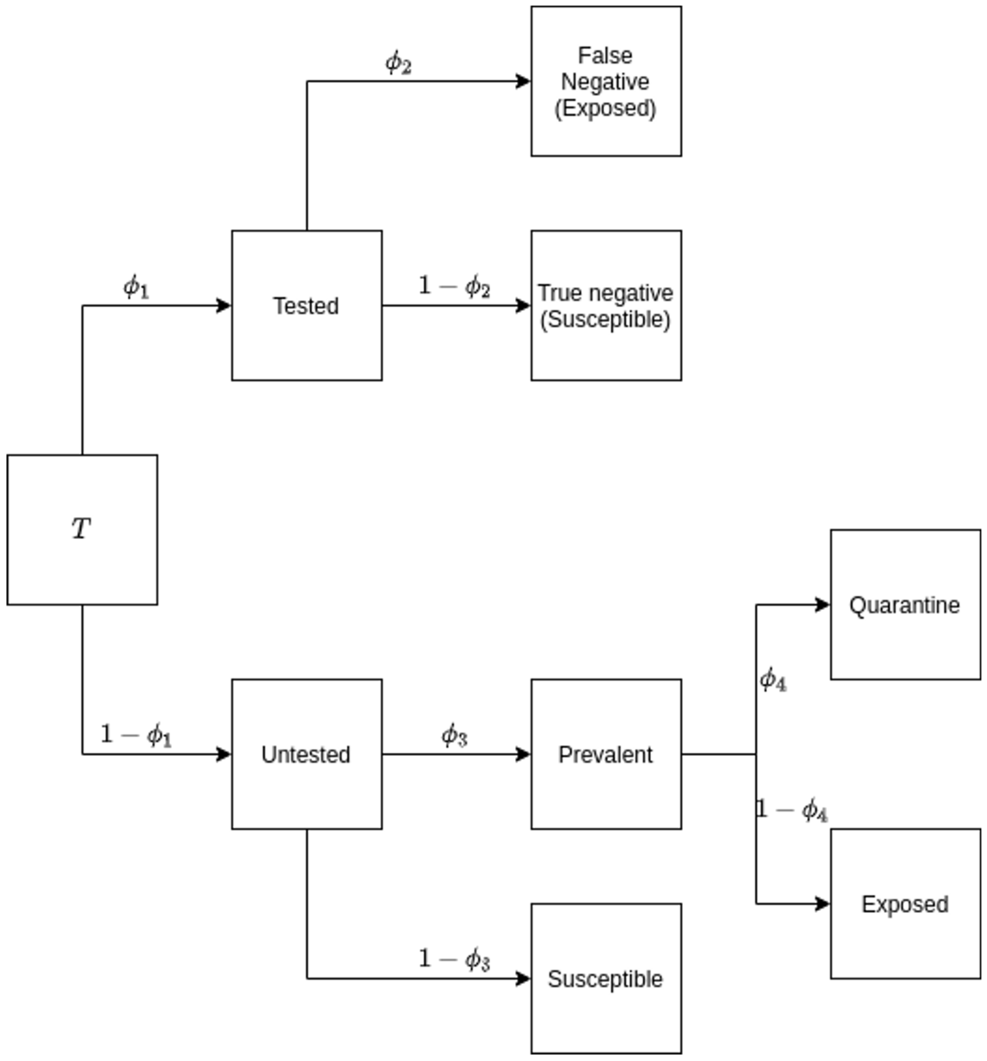


Dynamics Equations

$$
\begin{aligned}
S(t+1) &= e^{-\lambda(t)}S(t), \quad E_0(t+1)(1 - e^{-\lambda(t)})S(t) \\
E_i(t+1) &= (1 - p_{i-1})(1 - q_{a,i-1})E_{i-1}(t), \quad i = 1, \dots, 13 \\
E_{q,i}(t+1) &= (1 - p_{i-1})(q_{a,i-1}E_{i-1}(t) + E_{q,i-1}(t)), \quad i = 1, \dots, 13 \\
I_0(t+1) &= \sum_{i=0}^{13} p_i(1 - q_{a,i})E_i(t), \quad I_1(t+1) = (1 - q_{s,0})I_0(t) \\
I_2(t+1) &= (1 - q_{s,1})I_1(t) + (1 - r)(1 - q_{s,2})I_2(t) \\
I_j(t+1) &= r(1 - q_{s,j-1})I_{j-1}(t) + (1 - r)(1 - q_{s,j})I_j(t), \quad j = 3, 4 \\
I_{q,0}(t+1) &= \sum_{i=0}^{13} p_i(q_{a,i}E_i(t) + E_{q,i}(t)), \quad I_{q,1}(t+1) = I_{q,0}(t) + q_{s,0}I_0(t) \\
I_{q,2}(t+1) &= I_{q,1}(t) + q_{s,1}I_1(t) + (1 - r)(q_{s,2}I_2(t) + I_{q,2}(t)) \\
I_{q,j}(t+1) &= r(q_{s,j-1}I_{j-1}(t) + I_{q,j-1}(t)) + (1 - r)(q_{s,j}I_j(t) + I_{q,j}(t)) \\
R(t+1) &= R(t) + rI_4(t) + rI_{q,4}(t) + (1 - p_{13})(E_{13}(t) + E_{q,13}(t))
\end{aligned}
$$

Travelers

On October 15, 2020, the State of Hawaii implemented the Safe travel program. To implement travelers, we introduce daily travelers (T) and consider two broad categories of travelers - tourists and residents.



Vaccines

December 2020, the world started vaccinating!

The accelerated research, production, and distribution of the COVID-19 vaccines have had an impact on slowing the spread of COVID-19 in the areas that had access to enough supply of the vaccine.

As of Dec 3, 2021, more than 4.32 billion people worldwide have received a dose of a Covid-19 vaccine, equal to about 56.2 percent of the world population.. Hawaii as of Dec 2, 2021, as 71.6% fully vaccinated (Honolulu: 75%, Kauai: 69%, Maui and Hawaii: 64%) and 77.4% at least one shot.

We added vaccines to both of our computational models

Agent-Based Model

We use and modified to our needs COVID-19 Agent-based Simulator (Covasim) Developed by The Institute for Disease Modeling (part of the Bill & Melinda Gates Foundation's Global Health Division)

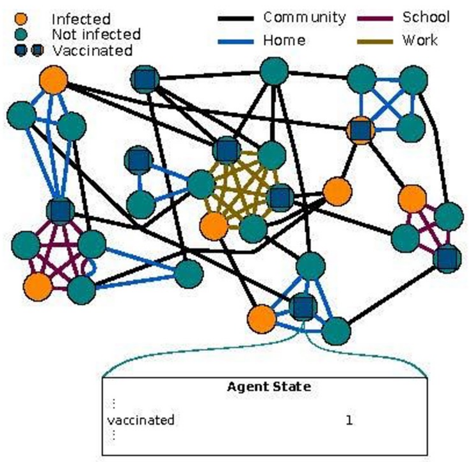
Compartmental models focus on directly capturing the collective behavior of groups of people, and are typically derived using estimates of aggregate (or limiting) behavior of a large number of individuals under (many) simplifying assumptions.

In contrast, agent based models (ABMs) focus on capturing the behavior of a single individual, referred to as an agent. Such individual behavior can often be described using fairly simple rules, however the collective behavior may still exhibit complicated phenomena.

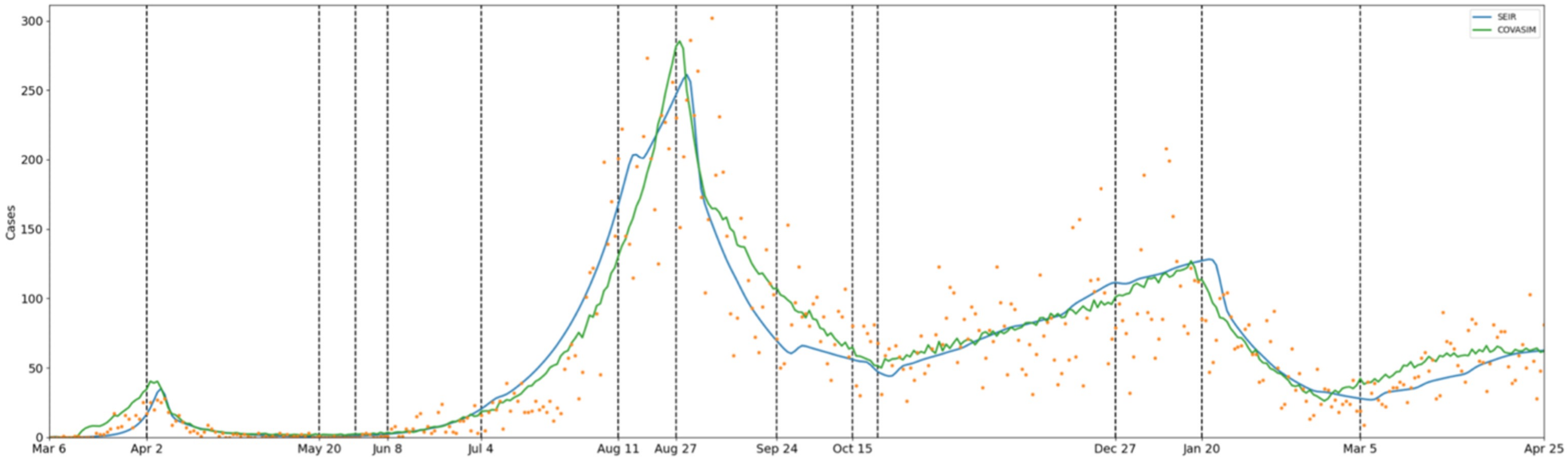
Popularity of ABMs exploded in the 90s, when the computational power significantly increased and became widely available.

Vaccines

Once the network construction is done, handling changes to agents' states due to newly available information can be readily implemented as tweaks to the interaction network and/or states of agents. Adding vaccinated individuals is conceptually quite simple. Of course, the process governing state transitions also needs to be updated.



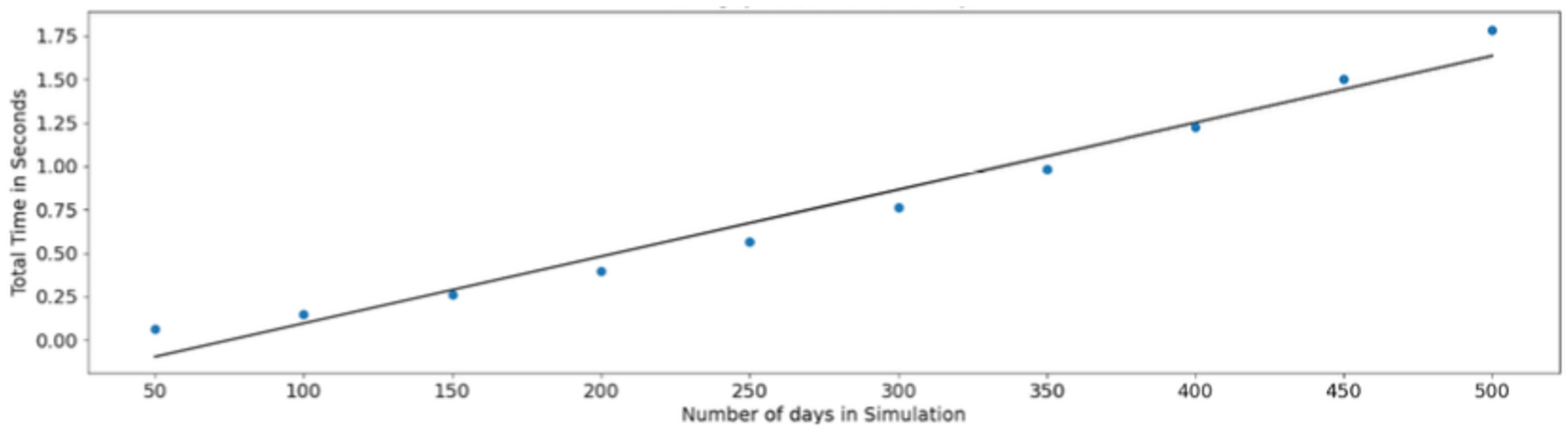
Benchmark Comparison of Models is Able To Capture Relevant Features



Blue: SEIR, Green:Covasim

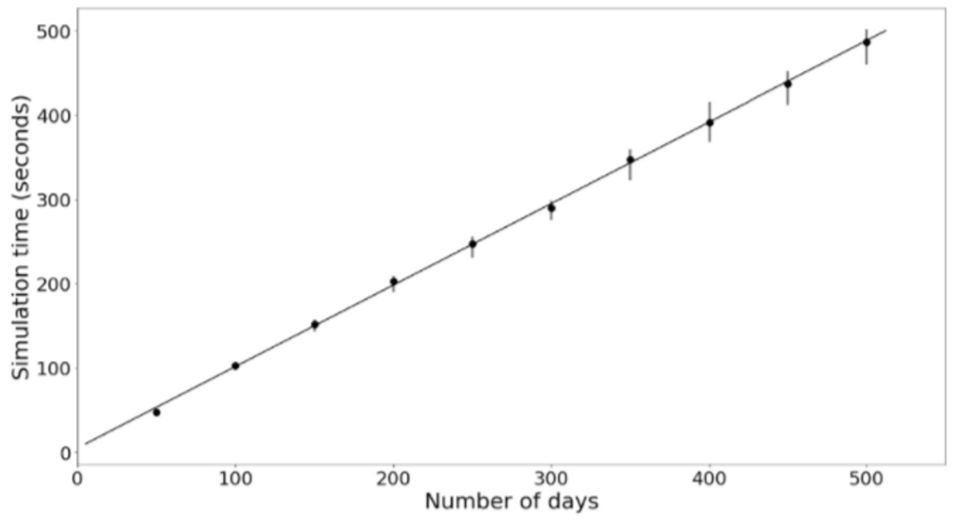
Sample computational cost of compartmental model

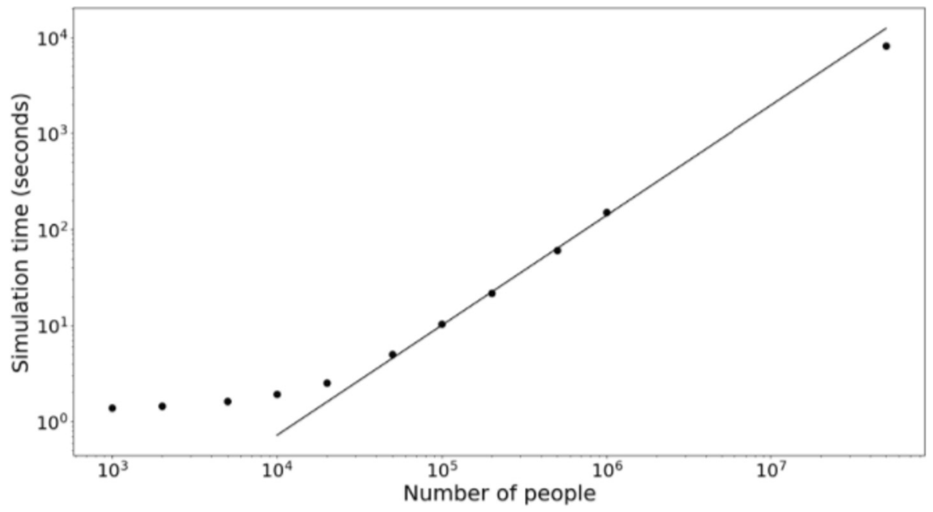
- Computational cost of compartmental models (for SEIR) is typically O(NC), where NC is the number of compartments, which is significantly less computationally intensive than agent based models.
- We see a roughly linear increase in the computational time with respect to the number of days. For this benchmark, a basic model with community and healthcare compartments is used without any interventions, travelers, or vaccinations.
- These simulations run easily on workstations or laptops.



- For ABMs each time step requires an iteration over each interaction edge, and the state of each agent needs to be evaluated and updated.
- Computational cost is O(N(|E| + |A|)), where N is number of time steps, |E| is the number of interaction edges and |A| is the number of agents.
- For top graph, set the population size equal to 1 million and run 15 simulations for each scenario with a different number of days starting from 50 to 500.
- For the bottom graph one we set the number of days equal to 150 and run 15 simulations for each scenario with different population sizes.
- The largest simulation was 150 days with 50 million people which took 8098 seconds.
- Larger simulations require HPC (High Performance Computing) techniques and more memory per node. Some of these are already available in Covasim, and others are becoming available with OpenMP.

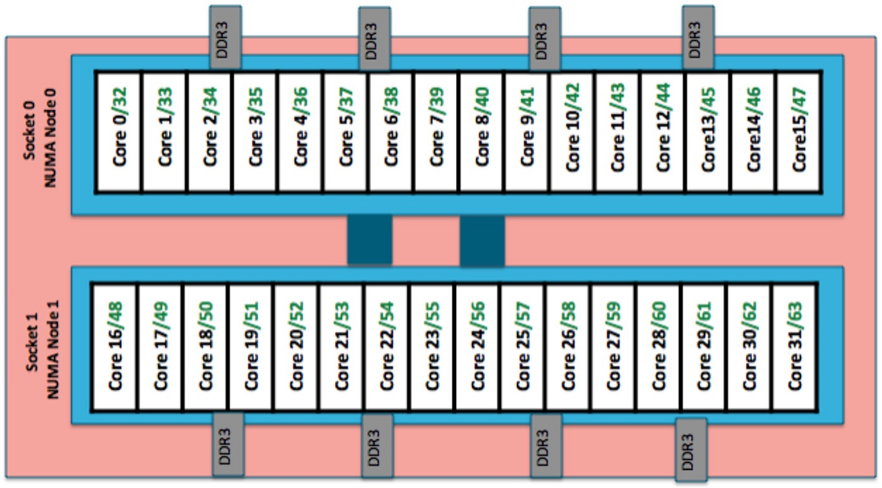
Sample computational cost of ABM





For the more computationally expensive ABM, simulations are run on NERSC/LBNL Cori (Cray XC40) system

- **9,688 Intel Knights Landing (KNL)** compute nodes
  - 68 cores per node, 4 hardware threads per core
- es: 32 cores/node



Thanks NERSC for providing resources under OpenMP development allocation

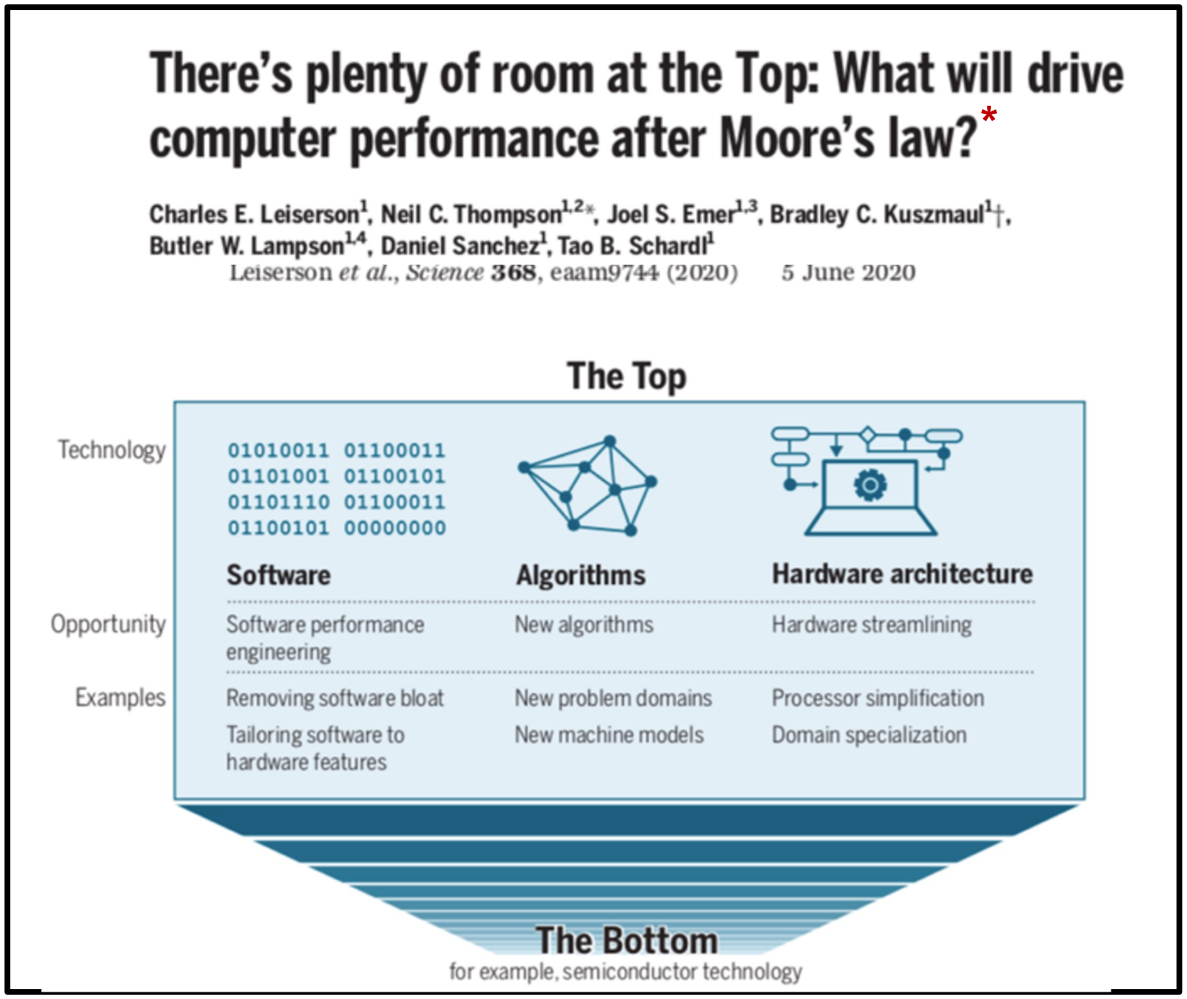
*KNL: Intel® Xeon Phi™ processor 7250 with 68 cores @ 1.4 GHz

*Haswell: Intel® Xeon™ processor E5-2698 v3 @ 2.3 GHz

Software vs. Hardware and the nature of Performance

Up until ~2005, performance came from semiconductor technology

From: SciPy'21 Anderson and Mattson



There's plenty of room at the Top: What will drive computer performance after Moore's law?*

Charles E. Leiserson[1], Neil C. Thompson[1,2]*, Joel S. Emer[1,3], Bradley C. Kuszmaul[1]†, Butler W. Lampson[1,4], Daniel Sanchez[1], Tao B. Schardl[1]

Leiserson *et al.*, *Science* **368**, eaam9744 (2020) 5 June 2020

Since ~2005 performance comes from "the top"

Better software Tech.
Better algorithms
Better HW architecture#

#HW architecture matters, but dramatically LESS than software and algorithms

The view of Python from an HPC perspective

From: SciPy'21 Anderson and Mattson and "Room at the top" paper

```
for I in range(4096):
   for j in range(4096):
      for k in range (4096):
            C[i][j] += A[i][k]*B[k][j]
```

A proxy for computing over nested loops … yes, they know you should use optimized library code for DGEMM

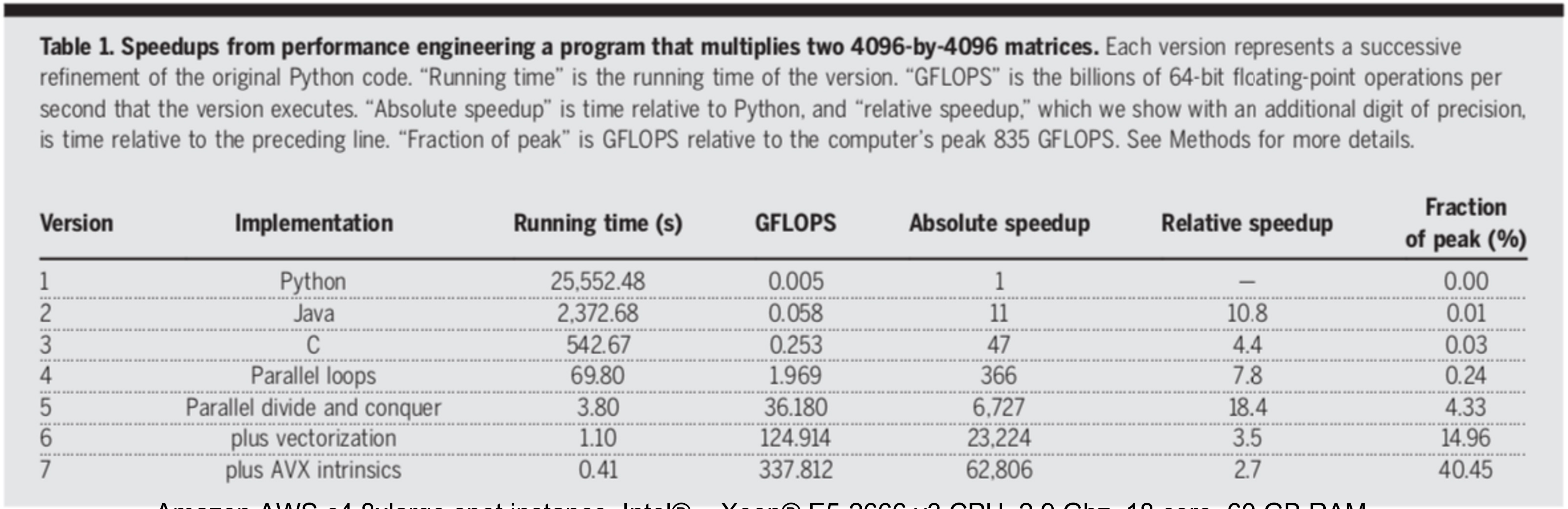


Table 1. Speedups from performance engineering a program that multiplies two 4096-by-4096 matrices. Each version represents a successive refinement of the original Python code. "Running time" is the running time of the version. "GFLOPS" is the billions of 64-bit floating-point operations per second that the version executes. "Absolute speedup" is time relative to Python, and "relative speedup," which we show with an additional digit of precision, is time relative to the preceding line. "Fraction of peak" is GFLOPS relative to the computer's peak 835 GFLOPS. See Methods for more details.

Version	Implementation	Running time (s)	GFLOPS	Absolute speedup	Relative speedup	Fraction of peak (%)
1	Python	25,552.48	0.005	1	–	0.00
2	Java	2,372.68	0.058	11	10.8	0.01
3	C	542.67	0.253	47	4.4	0.03
4	Parallel loops	69.80	1.969	366	7.8	0.24
5	Parallel divide and conquer	3.80	36.180	6,727	18.4	4.33
6	plus vectorization	1.10	124.914	23,224	3.5	14.96
7	plus AVX intrinsics	0.41	337.812	62,806	2.7	40.45

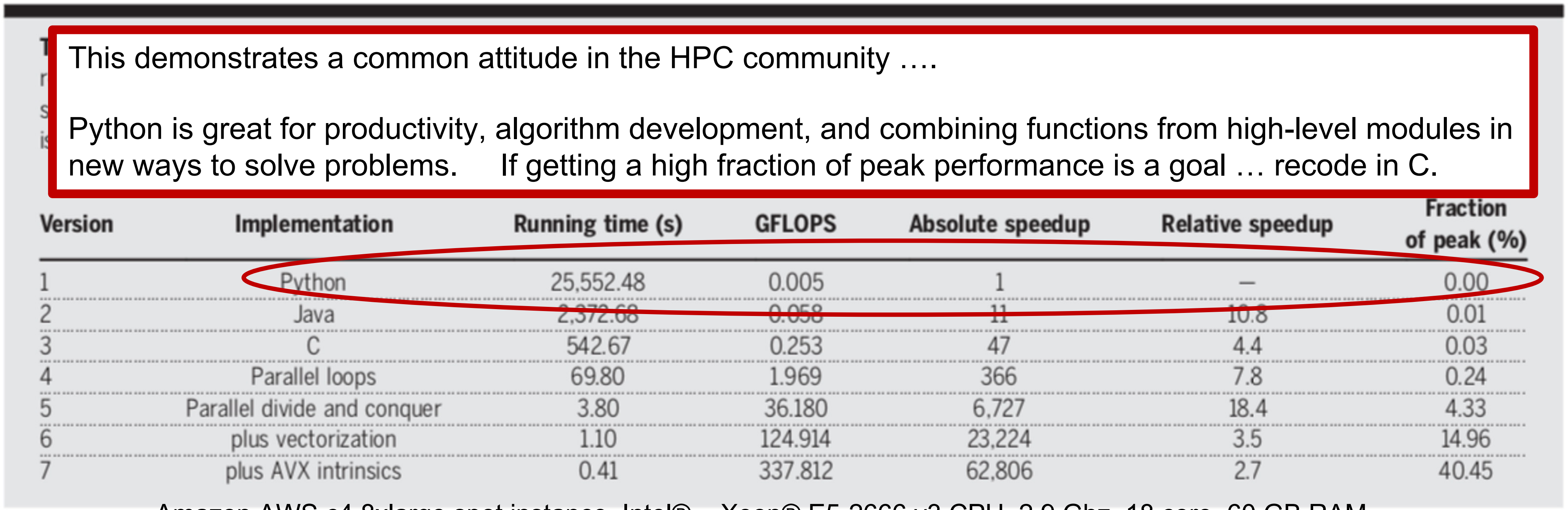
Amazon AWS c4.8xlarge spot instance, Intel® Xeon® E5-2666 v3 CPU, 2.9 Ghz, 18 core, 60 GB RAM

The view of Python from an HPC perspective

From: SciPy'21 Anderson and Mattson and "Room at the top" paper

```
for I in range(4096):
   for j in range(4096):
      for k in range (4096):
            C[i][j] += A[i][k]*B[k][j]
```

A proxy for computing over nested loops … yes, they know you should use optimized library code for DGEMM



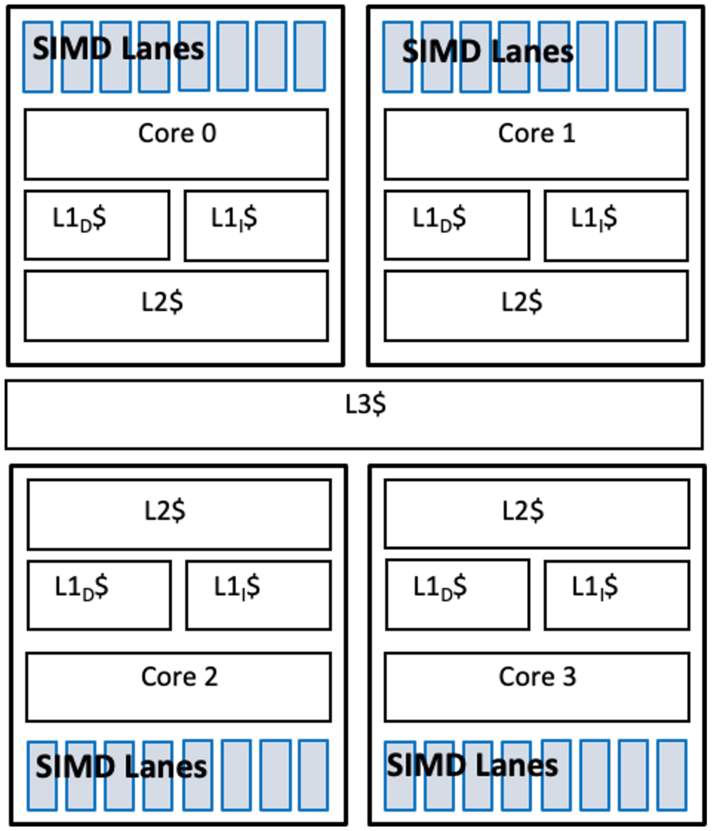
This demonstrates a common attitude in the HPC community ….

Python is great for productivity, algorithm development, and combining functions from high-level modules in new ways to solve problems.   If getting a high fraction of peak performance is a goal … recode in C.

Version	Implementation	Running time (s)	GFLOPS	Absolute speedup	Relative speedup	Fraction of peak (%)
1	Python	25,552.48	0.005	1	—	0.00
2	Java	2,372.68	0.058	11	10.8	0.01
3	C	542.67	0.253	47	4.4	0.03
4	Parallel loops	69.80	1.969	366	7.8	0.24
5	Parallel divide and conquer	3.80	36.180	6,727	18.4	4.33
6	plus vectorization	1.10	124.914	23,224	3.5	14.96
7	plus AVX intrinsics	0.41	337.812	62,806	2.7	40.45

Amazon AWS c4.8xlarge spot instance, Intel® Xeon® E5-2666 v3 CPU, 2.9 Ghz, 18 core, 60 GB RAM

How do you get high performance for a modern CPU?



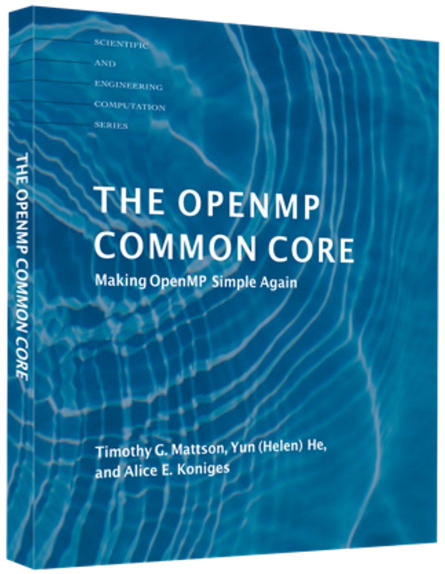
From: SciPy'21 Anderson and Mattson

Three simple principles:
- Lots of threads … at least one per hardware thread (often two hardware threads per core)
- Exploit SIMD lanes form each thread
- Maximize cache utilization

What is the most common way in HPC to create multithreaded code? Something called OpenMP

To learn OpenMP:

- New book that Covers the Common Core of OpenMP
- Common Core focuses on the 21 items that are most used
- Available from MIT Press

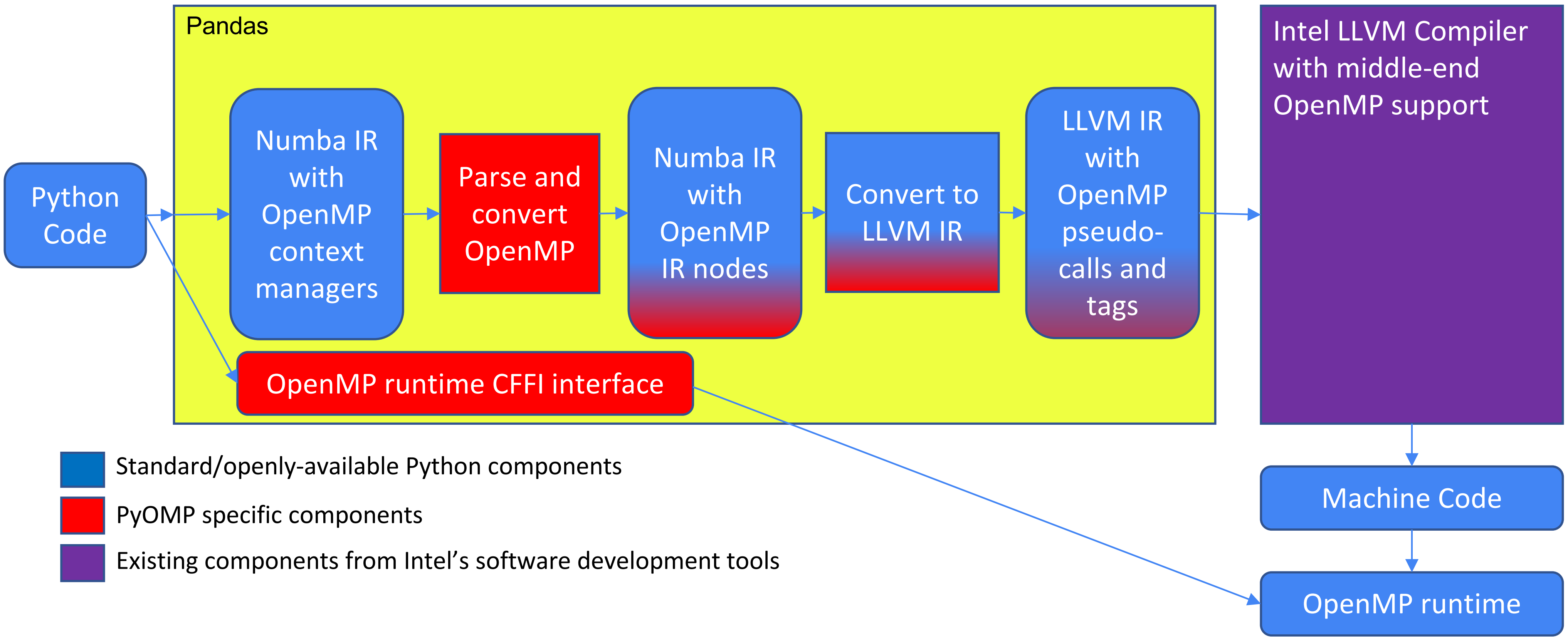

SCIENTIFIC AND ENGINEERING COMPUTATION SERIES
THE OPENMP COMMON CORE
Making OpenMP Simple Again
Timothy G. Mattson, Yun (Helen) He, and Alice E. Koniges
THE OPENMP COMMON CORE

Python and therefore Covasim can now gain OpenMP parallelism: new research area

- Covasim was developed for Python 3.8 using the SciPy (scipy.org) ecosystem [81]. It uses NumPy (numpy.org), Pandas (pandas.pydata.org), and Numba (numba.pydata.org) for fast numerical computing; Matplotlib (matplotlib.org) and Plotly (plotly.com) for plotting; and Sciris (sciris.org) for data structures, parallelization, and other utilities. **Covasim: An agent-based model of COVID-19 dynamics and interventions, Kerr, et al. 2021**
- An on-node parallelization methodology using OpenMP is being developed for Numba and Pandas which should improve on-node performance: Introducing PyOMP: Multithreaded parallel Python through OpenMP support in Numba (Todd Anderson and Tim Mattson, Intel Parallel Computing Lab)

Thanks: Institute for Disease Modeling for providing Open Source access to Covasim

PyOMP Implementation in Numba: Anderson and Mattson, SciPy21


Pandas
Python Code
Numba IR with OpenMP context managers
Parse and convert OpenMP
Numba IR with OpenMP IR nodes
Convert to LLVM IR
LLVM IR with OpenMP pseudo-calls and tags
OpenMP runtime CFFI interface
Intel LLVM Compiler with middle-end OpenMP support
Machine Code
OpenMP runtime

🟦 Standard/openly-available Python components

🟥 PyOMP specific components

🟪 Existing components from Intel's software development tools

Summary

Opportunities in PISALE group (code development, visualization, parallelism)

Opportunities in Pandemic Modeling group (model optimization and parallelism)

New proposals on horizon

Websites:

pisale.bitbucket.io

uhmdatascienceteam.com

From Alice’s Adventures in Wonderland:
“And yet you incessantly stand on your head—
Do you think, at your age, it is right?”

Acknowledgements

The PISALE code applications are supported by the National Science Foundation, under Office of Advanced Cyber Infrastructure Award Number 2005259, the U.S. Department of Energy, Office of Science, under Fusion Energy Sciences Research Division Award Number DE-SC0021374, DE-SC0023475, and DE-SC0024403 and the Office of Naval Research, under ONR MURI Award Number N00014- 20-1-2682.

COVID-19 modeling supported by the National Science Foundation under Grant No. 2030789, “Modeling COVID-19 transmission and mitigation using smaller contained populations.”

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC awards ERCAP0020882 and ERCAP0020890. Additional technical support and advanced computing resources are provided by the University of Hawaii Information Technology Services – Cyberinfrastructure, funded in part by the National Science Foundation MRI award Number 1920304.