Thoughts on software for science

Steve Plimpton
Sandia National Laboratories

ASCR Workshop on the
Science of Scientific Software Development and Use
December 2021

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

An initial observation plus some science humor

The **Science** of **Scientific** Software Development and Use

An initial observation plus some science humor

The **Science** of **Scientific** Software Development and Use

- That word doesn't mean the same thing both times
- First is about the **process** we use to create software
- Second says our software is used by **people who do science**

An initial observation plus some science humor

The **Science** of **Scientific** Software Development and Use

- That word doesn't mean the same thing both times
- First is about the **process** we use to create software
- Second says our software is used by **people who do science**

Richard Feynman, physics Nobel laureate:

"Any field which has to have **science** in its name isn't one"

An initial observation plus some science humor

The **Science** of **Scientific** Software Development and Use

- That word doesn't mean the same thing both times
- First is about the **process** we use to create software
- Second says our software is used by **people who do science**

Richard Feynman, physics Nobel laureate:
"Any field which has to have **science** in its name isn't one"

- physics, chemistry, biology, mathematics, statistics

An initial observation plus some science humor

The **Science** of **Scientific** Software Development and Use

- That word doesn't mean the same thing both times
- First is about the **process** we use to create software
- Second says our software is used by **people who do science**

Richard Feynman, physics Nobel laureate:
"Any field which has to have **science** in its name isn't one"

- physics, chemistry, biology, mathematics, statistics
- computer science, computational science, social science, cognitive science

An initial observation plus some science humor

The **Science** of **Scientific** Software Development and Use

- That word doesn't mean the same thing both times
- First is about the **process** we use to create software
- Second says our software is used by **people who do science**

Richard Feynman, physics Nobel laureate:
"Any field which has to have **science** in its name isn't one"

- physics, chemistry, biology, mathematics, statistics
- computer science, computational science, social science, cognitive science (verbatim from workshop **abstract**)

An initial observation plus some science humor

The **Science** of **Scientific** Software Development and Use

- That word doesn't mean the same thing both times
- First is about the **process** we use to create software
- Second says our software is used by **people who do science**

Richard Feynman, physics Nobel laureate:
"Any field which has to have **science** in its name isn't one"

- physics, chemistry, biology, mathematics, statistics
- computer science, computational science, social science, cognitive science (verbatim from workshop **abstract**)
- data science, information science, **political** science, ...

Kernel of truth in the humor

Whatever we call what we do and **however** we go about it ...
it's **not the same** as science that domain scientists do

Kernel of truth in the humor

Whatever we call what we do and **however** we go about it ...
it's **not the same** as science that domain scientists do

I think of myself as a **toolmaker**, not a scientist ...
I make tools (software) for people who do the science

Kernel of truth in the humor

Whatever we call what we do and **however** we go about it ...
it's **not the same** as science that domain scientists do

I think of myself as a **toolmaker**, not a scientist ...
I make tools (software) for people who do the science

So instead of asking:
How do we add more science to the process of making tools ?

Kernel of truth in the humor

Whatever we call what we do and **however** we go about it ...
it's **not the same** as science that domain scientists do

I think of myself as a **toolmaker**, not a scientist ...
I make tools (software) for people who do the science

So instead of asking:
How do we add more science to the process of making tools ?

Another way to **frame the challenge** for this workshop:

- (axiom 1) DOE's chief product is excellent domain science
- (axiom 2) Big computers and sci software are
  two (of many) tools used to that end
- **Challenge:** How do we make it as easy as possible for
  domain scientists to do excellent science with those 2 tools

Kernel of truth in the humor

Whatever we call what we do and **however** we go about it ...
it's **not the same** as science that domain scientists do

I think of myself as a **toolmaker**, not a scientist ...
I make tools (software) for people who do the science

So instead of asking:
How do we add more science to the process of making tools ?

Another way to **frame the challenge** for this workshop:

- (axiom 1) DOE's chief product is excellent domain science
- (axiom 2) Big computers and sci software are
  two (of many) tools used to that end
- **Challenge:** How do we make it as easy as possible for
  domain scientists to do excellent science with those 2 tools

The rest of my talk takes that **point-of-view**: my bias != your bias

Hardware history of supercomputing on one slide

History = only $\sim$30 years (I'm old but not ancient)

Hardware history of supercomputing on one slide

History = only ∼30 years (I'm old but not ancient)

- **Faster and faster** for Linpack flops:
  - 2 Gflop in 1988 - Cray YMP
  - 1 Tflop in 1997 - ASCI Red (Cray)
  - 1 Pflop in 2011 - Roadrunner (IBM)
  - 1 Eflop in 2022-3 - Frontier (HP) and Aurora (Intel)
  - every machine at a DOE lab
  - every machine a one-off (except many YMPs)

Hardware history of supercomputing on one slide

History = only ∼30 years (I'm old but not ancient)

- **Faster and faster** for Linpack flops:
  - 2 Gflop in 1988 - Cray YMP
  - 1 Tflop in 1997 - ASCI Red (Cray)
  - 1 Pflop in 2011 - Roadrunner (IBM)
  - 1 Eflop in 2022-3 - Frontier (HP) and Aurora (Intel)
  - every machine at a DOE lab
  - every machine a one-off (except many YMPs)

Bottom line is a huge success story:
billion X faster in 34 years, doubling Linpack flops every 14 months

Hardware history of supercomputing on one slide

History = only ∼30 years (I'm old but not ancient)

- **Faster and faster** for Linpack flops:
  - 2 Gflop in 1988 - Cray YMP
  - 1 Tflop in 1997 - ASCI Red (Cray)
  - 1 Pflop in 2011 - Roadrunner (IBM)
  - 1 Eflop in 2022-3 - Frontier (HP) and Aurora (Intel)
  - every machine at a DOE lab
  - every machine a one-off (except many YMPs)

Bottom line is a huge success story:
billion X faster in 34 years, doubling Linpack flops every 14 months

Reasons:

1. Moore's law
2. large-scale distributed parallelism
   - Summit = 5600 ft^2, 340 tons, 13 MW, 4000 gal H_2O/min
3. GPUs for threaded parallelism

Software history of supercomputing on one slide

- **Harder** and harder to get good performance
- **Fewer** and fewer science apps achieve that performance
- **Harder** and harder to program the machines
  - hybrid nodes (CPU+GPU), **imbalance** in compute vs comm

Imbalance ratios over 30 years

- **Local balance** = flops to pay for 1 on-node word (8 bytes)
- **Remote balance** = flops to pay for 1 off-node word
- Thanks to **Si Hammond** at Sandia

Imbalance ratios over 30 years

- **Local balance** = flops to pay for 1 on-node word (8 bytes)
- **Remote balance** = flops to pay for 1 off-node word
- Thanks to **Si Hammond** at Sandia

Year	Machine	Linpack	Flops/ local	Flops/ remote
1988	Cray YMP	2.1 Gflops	**0.5**	**0.5**
1997	ASCI Red (SNL)	1.6 Tflops	8.3	20
2011	RoadRunner (LANL)	1.0 Pflops	6.7	170
2011	K-Computer (Japan)	11 Pflops	15	95
2012	Sequoia (LLNL)	17 Pflops	32	160
2013	Titan (ORNL)	18 Pflops	29	490
2016	Sunway TaihuLight (China)	93 Pflops	130	1500
2018	Summit (ORNL)	149 Pflops	45	10300
2021	Sunway Exascale (China)	∼1.5 Eflops	350	9200
2022	Frontier (ORNL)	∼1.5 Eflops	**250**	**29400**

Software history of supercomputing on one slide

- **Harder** and harder to get good performance
- **Fewer** and fewer science apps achieve that performance
- **Harder** and harder to program the machines
  - hybrid nodes (CPU+GPU), **imbalance** in compute vs comm

Software history of supercomputing on one slide

- **Harder** and harder to get good performance
- **Fewer** and fewer science apps achieve that performance
- **Harder** and harder to program the machines
  - hybrid nodes (CPU+GPU), **imbalance** in compute vs comm
- Developing sci software requires **larger** and larger teams
- Developing sci software requires **more** and more $$

Software history of supercomputing on one slide

- **Harder** and harder to get good performance
- **Fewer** and fewer science apps achieve that performance
- **Harder** and harder to program the machines
  - hybrid nodes (CPU+GPU), **imbalance** in compute vs comm
- Developing sci software requires **larger** and larger teams
- Developing sci software requires **more** and more $$

- Most of this is driven by hardware & chasing-flops mindset
- **None of this is optimal** for science and domain scientists

ECP is perfect exemplar of both histories

7 year project (2016-2023), total budget around **$4 billion**

- $1.8B = hardware = 3 exascale machines at $600M each
- **$1.2B** = software = 30 apps + 30 software tech projects

ECP is perfect exemplar of both histories

7 year project (2016-2023), total budget around **$4 billion**

- $1.8B = hardware = 3 exascale machines at $600M each
- **$1.2B** = software = 30 apps + 30 software tech projects

A strategic **aside** for the workshop:

- DOE just spent $1.2B on Scientific Software for exascale
- We're saying **now** we need to apply scientific methods ?
- **Challenge #1**: What is this workshop proposing that wasn't included in ECP software $$ and why wasn't it ?

ECP is perfect exemplar of both histories

7 year project (2016-2023), total budget around **$4 billion**

- $1.8B = hardware = 3 exascale machines at $600M each
- **$1.2B** = software = 30 apps + 30 software tech projects

A strategic **aside** for the workshop:

- DOE just spent $1.2B on Scientific Software for exascale
- We're saying **now** we need to apply scientific methods ?
- **Challenge #1**: What is this workshop proposing that wasn't included in ECP software $$ and why wasn't it ?

Future is likely to be worse:

- SoC, neuromorphic computing, quantum computing, etc
- more specialized & narrowly applicable hardware, low-precision, harder to program

ECP is perfect exemplar of both histories

7 year project (2016-2023), total budget around **$4 billion**

- $1.8B = hardware = 3 exascale machines at $600M each
- **$1.2B** = software = 30 apps + 30 software tech projects

A strategic **aside** for the workshop:

- DOE just spent $1.2B on Scientific Software for exascale
- We're saying **now** we need to apply scientific methods ?
- **Challenge #1**: What is this workshop proposing that wasn't included in ECP software $$ and why wasn't it ?

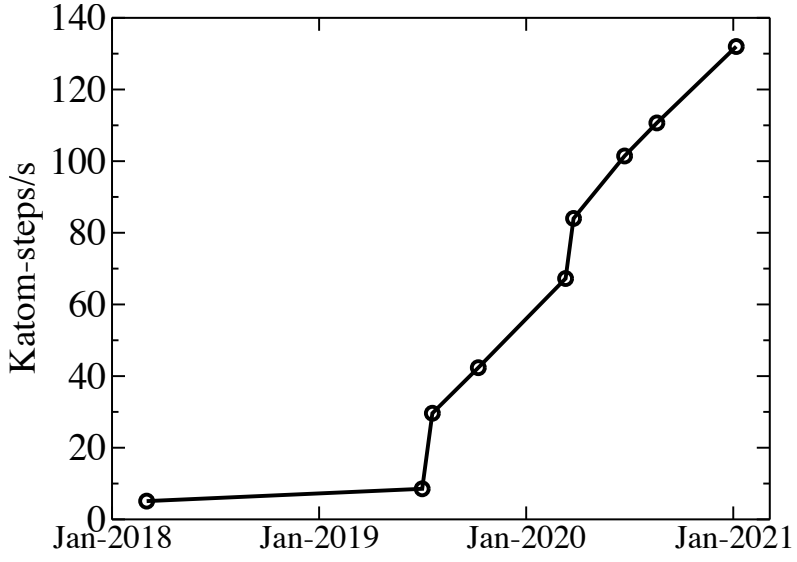
Future is likely to be worse:

- SoC, neuromorphic computing, quantum computing, etc
- more specialized & narrowly applicable hardware, low-precision, harder to program

Challenge #2: Can we exit this spending cycle of needing **more and more** $$ for software on new machines ?

One example from within ECP

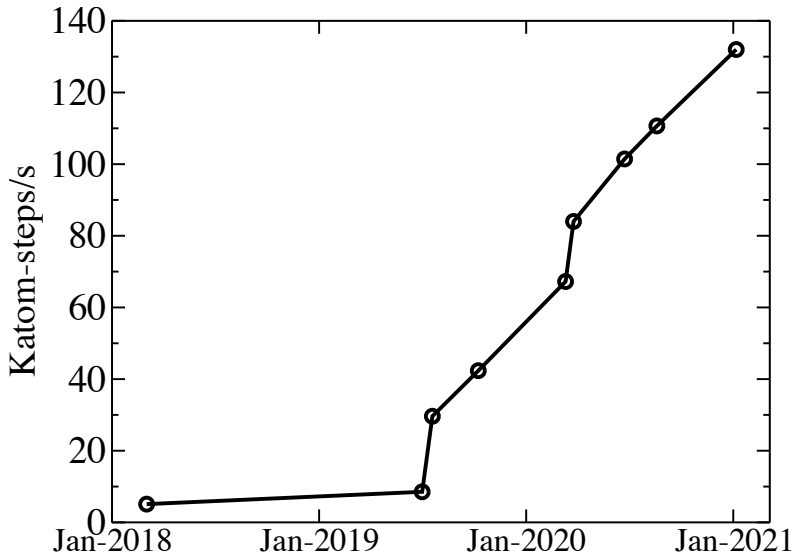
LAMMPS/SNAP **performance** on a NVIDIA V100 GPU



- LAMMPS = classical MD code for materials modeling
- SNAP = expensive, accurate ML potential
- **Kokkos-ized** for portable GPU performance
- **Team**: 2 ECP projects, NERSC, OLCF, ALCF, NVIDIA
- Algorithm refactoring, code restructuring, opt, tuning

One example from within ECP

LAMMPS/SNAP **performance** on a NVIDIA V100 GPU



- LAMMPS = classical MD code for materials modeling
- SNAP = expensive, accurate ML potential
- **Kokkos-ized** for portable GPU performance
- **Team**: 2 ECP projects, NERSC, OLCF, ALCF, NVIDIA
- Algorithm refactoring, code restructuring, opt, tuning
- Bottom line: **26x** speed-up in 3 years

Was the SNAP optimization effort a success or not ?

Was the SNAP optimization effort a success or not ?

Success !

- Just look at that plot
- EXAALT can now run larger or longer simulations or both

Was the SNAP optimization effort a success or not ?

Success !

- Just look at that plot
- EXAALT can now run larger or longer simulations or both

But it was also a **non-scalable** success !

- Large multi-institution team, several years, lots of $$
- Optimized **one** model in **one** code in **one** science field - **yikes**!
- Early AMD MI100 perf = **2.5x slower** than NVIDIA V100

Was the SNAP optimization effort a success or not ?

Success !

- Just look at that plot
- EXAALT can now run larger or longer simulations or both

But it was also a **non-scalable** success !

- Large multi-institution team, several years, lots of $$
- Optimized **one** model in **one** code in **one** science field - **yikes**!
- Early AMD MI100 perf = **2.5x slower** than NVIDIA V100

Challenge #3:
Can our software efforts & tools have broader impact ?

Issue #1: We live in a scientific software bubble

Issue #1: We live in a scientific software bubble

If you work at a DOE lab, or are funded by ASCR or ECP ...

- You live in an **insulated, myopic** bubble
- No malice, I've happily lived in that bubble for 30 years !

Issue #1: We live in a scientific software bubble

If you work at a DOE lab, or are funded by ASCR or ECP ...

- You live in an **insulated, myopic** bubble
- No malice, I've happily lived in that bubble for 30 years !

Most computational science is **NOT**

- Done on petascale/exascale machines or even supercomputers
- Done by people within DOE
- Done on GPUs
- And the customers for our software are often **each other**

Issue #1: We live in a scientific software bubble

If you work at a DOE lab, or are funded by ASCR or ECP ...

- You live in an **insulated, myopic** bubble
- No malice, I've happily lived in that bubble for 30 years !

Most computational science is **NOT**

- Done on petascale/exascale machines or even supercomputers
- Done by people within DOE
- Done on GPUs
- And the customers for our software are often **each other**

Challenge #4: To maximize impact on science,
can we focus on customers in the **much, much larger**
worldwide scientific computing community ?

Issue #2: As toolmakers, we like being needed

Who doesn't? It's human nature! Also funding and job security!

Issue #2: As toolmakers, we like being needed

Who doesn't? It's human nature! Also funding and job security!

The **old days**:

- One domain scientist wrote a small Fortran code
- Ran it themselves to do some science

Issue #2: As toolmakers, we like being needed

Who doesn't? It's human nature! Also funding and job security!

The **old days**:

- One domain scientist wrote a small Fortran code
- Ran it themselves to do some science

Today our culture has become (see the workshop abstract):

- Sci software requires **big teams** of cross-disciplinary experts
- Domain/computational/computer/social/cognitive scientists
- Complex software is created, some science is done
- If tool is complex $\Rightarrow$ team needs to **persist in perpetuity**

Issue #2: As toolmakers, we like being needed

Who doesn't? It's human nature! Also funding and job security!

The **old days**:

- One domain scientist wrote a small Fortran code
- Ran it themselves to do some science

Today our culture has become (see the workshop abstract):

- Sci software requires **big teams** of cross-disciplinary experts
- Domain/computational/computer/social/cognitive scientists
- Complex software is created, some science is done
- If tool is complex $\Rightarrow$ team needs to **persist in perpetuity**

I don't think that's a **feature**, I think it's a **bug**

- Better if domain scientists can adapt & extend our tools
  without us to do new science

Issue #2: As toolmakers, we like being needed

Who doesn't? It's human nature! Also funding and job security!

The **old days**:

- One domain scientist wrote a small Fortran code
- Ran it themselves to do some science

Today our culture has become (see the workshop abstract):

- Sci software requires **big teams** of cross-disciplinary experts
- Domain/computational/computer/social/cognitive scientists
- Complex software is created, some science is done
- If tool is complex $\Rightarrow$ team needs to **persist in perpetuity**

I don't think that's a **feature**, I think it's a **bug**

- Better if domain scientists can adapt & extend our tools **without us** to do new science

Challenge #5: As toolmakers, can we write software so simple and good, we become **obsolete** (once the tool is deployed) ?

Complex software is sub-optimal for domain science

Software **complexity** is not a **simple** issue (pun intended)

- Modern computers & science have unavoidable complexity
- **One hand**: shielding users from complexity is often good
- **OTOH**: domain scientists often need to know what's going on
- Payoff for hiding complexity varies with kind of software:
  - **application** vs library vs system software

Complex software is sub-optimal for domain science

Software **complexity** is not a **simple** issue (pun intended)

- Modern computers & science have unavoidable complexity
- **One hand**: shielding users from complexity is often good
- **OTOH**: domain scientists often need to know what's going on
- Payoff for hiding complexity varies with kind of software:
  - **application** vs library vs system software

Sources of complexity for domain scientists:

- C++: templating, overloading, convoluted data structs, etc
- Hardware optimizations:
  compler directives, Cuda kernels, Kokkos abstractions, etc
- **ML**: perfectly designed to make path-to-results
  opaque to end users

Effects of sci software complexity on domain scientists

- Harder to **understand** what code is doing
- Harder to **implement** their new ideas w/out expert help
- Harder to **interpret** results
- Easier to misuse code or **make mistakes**

Effects of sci software complexity on domain scientists

- Harder to **understand** what code is doing
- Harder to **implement** their new ideas w/out expert help
- Harder to **interpret** results
- Easier to misuse code or **make mistakes**

Challenge #6: Can we reduce complexity of understanding & modifying scientific software so average domain scientist can still do it ?

Effects of sci software complexity on domain scientists

- Harder to **understand** what code is doing
- Harder to **implement** their new ideas w/out expert help
- Harder to **interpret** results
- Easier to misuse code or **make mistakes**

Challenge #6: Can we reduce complexity of understanding & modifying scientific software so average domain scientist can still do it ?

David Parnas: *"**Complexity is not a goal**. I don't want to be remembered as an engineer of complex systems."*

Example of a scientific mistake due to software complexity

2018: Emails from Doug Kothe (head of ECP) and Mike Heroux (ECP Software), asking about this article:

- Sub-title: How a **hidden coding error** fueled a seven-year dispute between two of condensed matter's top theorists
- Article says group that made a mistake was using **LAMMPS**

Some details of controversy

- D Chandler (UC Berkeley) and P Debenedetti (Princeton)
- Stat mech Q about 1 vs 2 phases of supercooled water
- **Half-day panic**: Was this caused by a bug in LAMMPS ?

Some details of controversy

- D Chandler (UC Berkeley) and P Debenedetti (Princeton)
- Stat mech Q about 1 vs 2 phases of supercooled water
- **Half-day panic**: Was this caused by a bug in LAMMPS ?
- Used their own MC code to wrap LAMMPS as a library & perform short MD simulations on a rigid-body water model
  - their MC-swap of velocities between water molecules $\Rightarrow$ not compatible with rigid-body motion
  - LAMMPS corrected for that $\Rightarrow$ too-cold molecules
  - their code added back heat to only translational DOFs
  - **result**: **hot** translational DOFs, **cold** rotational DOFs
- Messed up their thermodynamic stats & their science

Some details of controversy

- D Chandler (UC Berkeley) and P Debenedetti (Princeton)
- Stat mech Q about 1 vs 2 phases of supercooled water
- **Half-day panic**: Was this caused by a bug in LAMMPS ?
- Used their own MC code to wrap LAMMPS as a library & perform short MD simulations on a rigid-body water model
  - their MC-swap of velocities between water molecules $\Rightarrow$ not compatible with rigid-body motion
  - LAMMPS corrected for that $\Rightarrow$ too-cold molecules
  - their code added back heat to only translational DOFs
  - **result**: **hot** translational DOFs, **cold** rotational DOFs
- Messed up their thermodynamic stats & their science
- **Bottom line**: bug in their code, not LAMMPS
  - were clues in LAMMPS manual, but who reads the manual ?
- **Key point**: Our code was complicated enough that expert domain scientists got confused and made a science mistake

Another practical lesson we learned with LAMMPS

- New science often require **new models** and **new code**
  - different material (alloys, mixtures, interfaces, etc)
  - different conditions (state points, non-equil driving forces)
  - different scales: atomic to meso to continuum (particles)

Another practical lesson we learned with LAMMPS

- New science often require **new models** and **new code**
  - different material (alloys, mixtures, interfaces, etc)
  - different conditions (state points, non-equil driving forces)
  - different scales: atomic to meso to continuum (particles)

Goal: Make it **easy** for domain scientists to add new models

Another practical lesson we learned with LAMMPS

- New science often require **new models** and **new code**
  - different material (alloys, mixtures, interfaces, etc)
  - different conditions (state points, non-equil driving forces)
  - different scales: atomic to meso to continuum (particles)

Goal: Make it **easy** for domain scientists to add new models

1. LAMMPS has 20 **styles**: potentials, diagnostics, etc
   - parent class: defines API to rest of code
   - child classes: written/contributed by anyone (90% of code)

Another practical lesson we learned with LAMMPS

- New science often require **new models** and **new code**
  - different material (alloys, mixtures, interfaces, etc)
  - different conditions (state points, non-equil driving forces)
  - different scales: atomic to meso to continuum (particles)

Goal: Make it **easy** for domain scientists to add new models

1. LAMMPS has 20 **styles**: potentials, diagnostics, etc
   - parent class: defines API to rest of code
   - child classes: written/contributed by anyone (90% of code)
2. Average domain scientist is not a **coding wizard**
   - have to read/understand existing code to create new code
   - keep the code simple: data structs, little templating
   - most of LAMMPS is more C-like code than C++

Another practical lesson we learned with LAMMPS

- New science often require **new models** and **new code**
  - different material (alloys, mixtures, interfaces, etc)
  - different conditions (state points, non-equil driving forces)
  - different scales: atomic to meso to continuum (particles)

Goal: Make it **easy** for domain scientists to add new models

1. LAMMPS has 20 **styles**: potentials, diagnostics, etc
   - parent class: defines API to rest of code
   - child classes: written/contributed by anyone (90% of code)
2. Average domain scientist is not a **coding wizard**
   - have to read/understand existing code to create new code
   - keep the code simple: data structs, little templating
   - most of LAMMPS is more C-like code than C++

Result: Contributions by 100s of users, some we never imagined

However we also learned this lesson

Child styles also used for **CPU & GPU acceleration**:
variants for OpenMP, Intel, Cuda, Kokkos

Vanilla CPU	Accelerated CPU/GPU styles		
Pair+Fix styles	coverage	by developers	by users
529	33%	275	**23**

However we also learned this lesson

Child styles also used for **CPU & GPU acceleration**:
variants for OpenMP, Intel, Cuda, Kokkos

Vanilla CPU	Accelerated CPU/GPU styles		
Pair+Fix styles	coverage	by developers	by users
529	33%	275	**23**

- We're now 14 years into the GPU revolution
- Average comp/domain scientists aren't writing GPU code
- LAMMPS falling **further behind** in portion GPU-ized
- Not sure if this is good news or bad news, but ...

However we also learned this lesson

Child styles also used for **CPU & GPU acceleration**:
variants for OpenMP, Intel, Cuda, Kokkos

Vanilla CPU	Accelerated CPU/GPU styles		
Pair+Fix styles	coverage	by developers	by users
529	33%	275	**23**

- We're now 14 years into the GPU revolution
- Average comp/domain scientists aren't writing GPU code
- LAMMPS falling **further behind** in portion GPU-ized
- Not sure if this is good news or bad news, but ...

Challenge #7: Can our tools enable average domain scientists to more easily write performant code ?

Examples of tool success stories

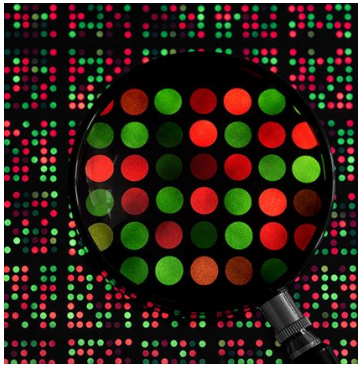
3 flavors

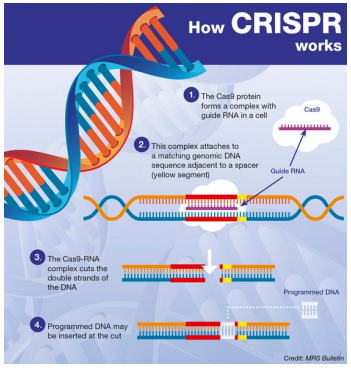
- Outside software
- Outside scientific software
- Within scientific software

Tool success story outside software

Experimental cell biology

- PCR (1983) = polymerase chain reaction, DNA replication
- Microarray chips (1995) = parallel gene expression (millions)
- DNA sequencing (2001) = \$10K/Mb $\Rightarrow$ 2 dimes/Mb
- CRISPR (2012) = genome editing in living cells



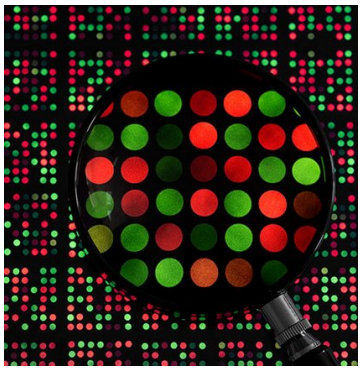


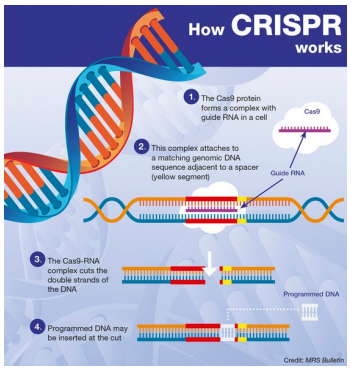
- 3 Nobel prizes on this list
- All these technologies rapidly became **ubiquitous**
- Any lab, any grad student can use them

Tool success story outside software

Experimental cell biology

- PCR (1983) = polymerase chain reaction, DNA replication
- Microarray chips (1995) = parallel gene expression (millions)
- DNA sequencing (2001) = \$10K/Mb $\Rightarrow$ 2 dimes/Mb
- CRISPR (2012) = genome editing in living cells





- 3 Nobel prizes on this list
- All these technologies rapidly became **ubiquitous**
- Any lab, any grad student can use them
- **Challenge #8**: Aspire to that **ease-of-use** for our software ?

Tool success story outside scientific software

Google Mapreduce + Hadoop

- 2004 by Jeffrey Dean and Sanjay Ghemawat
- Goal: make it easy for anyone within Google to use
  their **huge distributed computers** to analyze big data
- Within a few years 1000s of MR codes running at Google
- Spawned wildly popular open-source Hadoop project

Tool success story outside scientific software

Google Mapreduce + Hadoop

- 2004 by Jeffrey Dean and Sanjay Ghemawat
- Goal: make it easy for anyone within Google to use their **huge distributed computers** to analyze big data
- Within a few years 1000s of MR codes running at Google
- Spawned wildly popular open-source Hadoop project

Challenge #9: Can we make it similarly easy for **anyone** to write scientific software for big parallel computers ?

Another tool success story outside scientific software

iPhone or Android apps

- 2.2M iOS apps at Apple store
- 3.5M Android apps at Google Play
- Ecosystem evolved $\Rightarrow$ anyone can write a phone app
- Don't need an Apple employee or Android expert on your team

Challenge #10: Why not 10K or 1M apps on our big computers ?
Challenge #11: Create similar ecosystem for scientific software ?

Tool success story #1 within scientific software

Python language

- Created around 1990 by Guido van Rossum
- Initial audience was not computational science
- Now ubiquitous in computational science and ML
- **Easy** to understand, **easy** to write, **easy** to extend

Tool success story #2 within scientific software

FFTW library for discrete Fourier transforms

- Late 1990s by Matteo Frigo and Steven Johnson
- Wilkinson prize for numerical software
- Example of a perfect black box (admittedly in narrow space)
- One of first **auto-tuning** numerical codes
- Hides the right amount of complexity: hardware, FFT size
- Easy for domain scientists to use,
  they know models amenable to solution via FFTs

Possible tool success story #3 within scientific software

PETSc and Trilinos sparse iterative solver packages

- Large teams at 2 DOE labs: Argonne and Sandia
- Goal: hide solver complexity from domain scientists
- Goal: enable use of complicated solvers as a black box
- Goal: make it easy to plug into any PDE model

Possible tool success story #3 within scientific software

PETSc and Trilinos sparse iterative solver packages

- Large teams at 2 DOE labs: Argonne and Sandia
- Goal: hide solver complexity from domain scientists
- Goal: enable use of complicated solvers as a black box
- Goal: make it easy to plug into any PDE model

Disclaimer: I'm not a user, so don't know ...

- How well are these goals met ?
- Are they easy for average domain scientists to use & extend ?

Possible tool success story #3 within scientific software

PETSc and Trilinos sparse iterative solver packages

- Large teams at 2 DOE labs: Argonne and Sandia
- Goal: hide solver complexity from domain scientists
- Goal: enable use of complicated solvers as a black box
- Goal: make it easy to plug into any PDE model

Disclaimer: I'm not a user, so don't know ...

- How well are these goals met ?
- Are they easy for average domain scientists to use & extend ?

Challenge #12:

Measure ease-of-use or learning curve for sci software ?

Challenge #13:

Measure for average domain scientists who both use & extend ?

Final thoughts

From new book: The Genesis of Technoscientific Revolutions by Venki Narayanamurti and Jeff Tsao (DOE people!)

Final thoughts

From new book: The Genesis of Technoscientific Revolutions
by Venki Narayanamurti and Jeff Tsao (DOE people!)

*"**Research** is a deeply human endeavor that must be **nurtured** to achieve its full potential. As with tending a garden, care must be taken to organize, plant, feed, and weed - and the manner in which this nurturing is done must be consistent with the **nature** of what is being nurtured."*

Final thoughts

From new book: The Genesis of Technoscientific Revolutions
by Venki Narayanamurti and Jeff Tsao (DOE people!)

*"**Research** is a deeply human endeavor that must be **nurtured** to achieve its full potential. As with tending a garden, care must be taken to organize, plant, feed, and weed - and the manner in which this nurturing is done must be consistent with the **nature** of what is being nurtured."*

Just replace **Research** with **Creating scientific software**

Final thoughts

From new book: The Genesis of Technoscientific Revolutions
by Venki Narayanamurti and Jeff Tsao (DOE people!)

*"**Research** is a deeply human endeavor that must be **nurtured** to achieve its full potential. As with tending a garden, care must be taken to organize, plant, feed, and weed - and the manner in which this nurturing is done must be consistent with the **nature** of what is being nurtured."*

Just replace **Research** with **Creating scientific software**

This workshop proposes to nurture how sci software is developed

- Need to recognize the nature of what is being nurtured
- Nature = domain scientists use our software to do science
- If we don't make it easier for them to do it,
  we're not nurturing sci software development **effectively**

Final thoughts

From new book: The Genesis of Technoscientific Revolutions
by Venki Narayanamurti and Jeff Tsao (DOE people!)

*"**Research** is a deeply human endeavor that must be **nurtured** to achieve its full potential. As with tending a garden, care must be taken to organize, plant, feed, and weed - and the manner in which this nurturing is done must be consistent with the **nature** of what is being nurtured."*

Just replace **Research** with **Creating scientific software**

This workshop proposes to nurture how sci software is developed

- Need to recognize the nature of what is being nurtured
- Nature = domain scientists use our software to do science
- If we don't make it easier for them to do it,
  we're not nurturing sci software development **effectively**

DOE labs are the perfect place to try and do this