30 Years of HPC from Gigaflops to Exaflops: No science left behind or thinning the herd?

Steve Plimpton
Sandia National Laboratories

2018 Salishan Conference on High-Speed Computing

Impact of advancing HPC on particle simulations

- Many methods/models are $\sim O(N)$ cost in particle count
- Also scale as $\sim O(N/P)$ in parallel, for large enough $N/P$
- 1000x machine $\Rightarrow$ 1000x more particles or time or combo
Impact of advancing HPC on particle simulations

- Many methods/models are $\sim O(N)$ cost in particle count.
- Also scale as $\sim O(N/P)$ in parallel, for large enough $N/P$.
- 1000x machine $\Rightarrow$ 1000x more particles or time or combo.

30 yrs ago:
my thesis
1000 atoms
50K steps
Impact of advancing HPC on particle simulations

- Many methods/models are $\sim O(N)$ cost in particle count
- Also scale as $\sim O(N/P)$ in parallel, for large enough $N/P$
- $1000\times$ machine $\Rightarrow 1000\times$ more particles or time or combo

30 yrs ago:
- my thesis
- 1000 atoms
- 50K steps

Today:
- V Bulatov, et al (LLNL)
- 2.1B atoms
- 460M steps
Impact of advancing HPC on particle simulations

- Many methods/models are $\sim O(N)$ cost in particle count
- Also scale as $\sim O(N/P)$ in parallel, for large enough $N/P$
- 1000x machine $\Rightarrow$ 1000x more particles or time or combo

30 yrs ago:
my thesis
1000 atoms
50K steps

Today:
V Bulatov, et al (LLNL)
2.1B atoms
460M steps

Linpack: 1 BG/Q core / 1 Cray YMP proc = 41x !!
Impact of advancing HPC on particle simulations

- Many methods/models are $\sim O(N)$ cost in particle count
- Also scale as $\sim O(N/P)$ in parallel, for large enough $N/P$
- 1000x machine $\Rightarrow$ 1000x more particles or time or combo

30 yrs ago: my thesis
1000 atoms
50K steps

Today:
V Bulatov, et al (LLNL)
2.1B atoms
460M steps

- Linpack: 1 BG/Q core / 1 Cray YMP proc = 41x !!
- Cray YMP proc $\Rightarrow$ third of BG/Q Sequoia $\Rightarrow$ 21M faster
- MD atom-steps/s $\Rightarrow$ 8.5M faster
Newspapers and ukuleles

Ben Bagdikian (1920-2016): Pulitzer prize journalist, professor
Newspapers and ukuleles

Ben Bagdikian (1920-2016): Pulitzer prize journalist, professor

“Trying to be a first-rate reporter on the average American newspaper is like trying to play Bach’s *St. Matthew’s Passion* on a ukulele.”
“Trying to do first-rate science on the average petascale (soon exascale) machine is like trying to play Bach’s *St. Matthew’s Passion* on a million ukuleles.”
What I’m **NOT** trying to say

HPC is underfunded (like newspapers), this is one of best times in my memory for scientific HPC, lots of buzz about exascale, machine learning, beyond Moore Chip designers or machine architects are creating underpowered or deficient hardware (ukuleles). Current & future hardware is incredibly sophisticated. BTW, ukuleles can be amazing musical instruments. No one is doing first-rate science on petascale machines. Gordon Bell prizes at SC conference, INCITE allocations by DOE.
What I’m **NOT** trying to say

- HPC is **underfunded** (like newspapers)
  - this is one of best times in my memory for scientific HPC
  - lots of buzz about exascale, machine learning, beyond Moore
What I’m **NOT** trying to say

- HPC is **underfunded** (like newspapers)
  - this is one of best times in my memory for scientific HPC
  - lots of buzz about exascale, machine learning, beyond Moore

- Chip designers or machine architects are creating underpowered or **deficient hardware** (ukuleles)
  - current & future hardware is incredibly sophisticated
  - BTW, ukuleles can be amazing musical instruments
  - google **Jake Shimabukuro**
What I’m **NOT** trying to say

- **HPC is underfunded** (like newspapers)
  - this is one of best times in my memory for scientific HPC
  - lots of buzz about exascale, machine learning, beyond Moore

- Chip designers or machine architects are creating underpowered or **deficient hardware** (ukuleles)
  - current & future hardware is incredibly sophisticated
  - BTW, ukuleles can be amazing musical instruments
  - google **Jake Shimabukuro**

- No one is doing **first-rate science** on petascale machines
  - Gordon Bell prizes at SC conference
  - INCITE allocations by DOE
So what am I trying to say?
So what am I trying to say?

- We’re making machines that are **increasingly complex**
  - harder to program
  - harder to achieve high performance on
  - optimized for a narrower portion of computational science
So what am I trying to say?

- We’re making machines that are increasingly complex
  - harder to program
  - harder to achieve high performance on
  - optimized for a narrower portion of computational science

- Software developers (apps, libs, system) spend a lot of time & energy figuring out how to optimize for, and use these machines effectively
  - we suffer from Ukulele Syndrome
So what am I trying to say?

- We’re making machines that are **increasingly complex**
  - harder to program
  - harder to achieve high performance on
  - optimized for a narrower portion of computational science

- Software developers (apps, libs, system) spend a **lot of time & energy** figuring out how to optimize for, and use these machines effectively
  - we suffer from **Ukulele Syndrome**

- These two realities have a cost:
  - may be optimal for flops, but is **sub-optimal for science**
  - lots of time/$$ not spent on science
  - leaving many apps and scientists on the sidelines
  - there **ain’t that many** Jake Shimabukuros
What is Ukulele Syndrome?
Different than Ukulele Acquisition Syndrome (UAS)

- Google “ukulele syndrome”, will find real thing called UAS

- “What is UAS? It’s that insatiable desire to acquire the next ukulele for your need, to find that perfect ukulele for that favorite song, sound, or moment.”
Different than Ukulele Acquisition Syndrome (UAS)

- Google “ukulele syndrome”, will find real thing called UAS

- “What is UAS? It’s that insatiable desire to acquire the next ukulele for your need, to find that perfect ukulele for that favorite song, sound, or moment.”
- People **addicted** to buying more and more hardware
- Many of you and your orgs have serious cases of HAS
Different than Ukulele Acquisition Syndrome (UAS)

- Google “ukulele syndrome”, will find real thing called UAS

- “What is UAS? It’s that insatiable desire to acquire the next ukulele for your need, to find that perfect ukulele for that favorite song, sound, or moment.”
- People addicted to buying more and more hardware
- Many of you and your orgs have serious cases of HAS
- Sorry, there is no known cure for UAS or HAS
Five symptoms of Ukulele Syndrome

Do you or an app developer you care about experience these?

When you get a new, faster machine ...
Five symptoms of Ukulele Syndrome

Do you or an app developer you care about experience these?

When you get a new, faster machine ...

- Spend time and $$
  - re-writing your kernels or code for Nth time
  - writing papers for Mth time on how code adapted/perform
Five symptoms of Ukulele Syndrome

Do you or an app developer you care about experience these?

When you get a new, faster machine ...

- Spend **time and $$**
  - re-writing your kernels or code for Nth time
  - writing papers for Mth time on how code adapted/perform

- Use a **brute force algorithm** instead of more clever one
  - b/c it runs faster or more scalably, or at higher flop rate
Five symptoms of Ukulele Syndrome

Do you or an app developer you care about experience these?

When you get a new, faster machine ...

- Spend time and $$
  - re-writing your kernels or code for Nth time
  - writing papers for Mth time on how code adapted/performs
- Use a brute force algorithm instead of more clever one
  - b/c it runs faster or more scalably, or at higher flop rate
- See your code performance decrease:
  relative to PUP = peak ukulele performance
Five symptoms of Ukulele Syndrome

Do you or an app developer you care about experience these?

When you get a new, faster machine ...

- Spend **time and $$**
  - re-writing your kernels or code for Nth time
  - writing papers for Mth time on how code adapted/performs
- Use a **brute force algorithm** instead of more clever one
  - b/c it runs faster or more scalably, or at higher flop rate
- See your **code performance decrease**: relative to PUP = peak ukulele performance
- Run your code on **smaller fraction** of new machine:
  - b/c it doesn’t scale to more nodes
  - b/c running bigger problems for same short time isn’t useful
Five symptoms of Ukulele Syndrome

Do you or an app developer you care about experience these?

When you get a new, faster machine ...

- Spend **time and $$**
  - re-writing your kernels or code for Nth time
  - writing papers for Mth time on how code adapted/perform
- Use a **brute force algorithm** instead of more clever one
  - b/c it runs faster or more scalably, or at higher flop rate
- See your **code performance decrease**:
  - relative to PUP = peak ukulele performance
- Run your code on **smaller fraction** of new machine:
  - b/c it doesn’t scale to more nodes
  - b/c running bigger problems for same short time isn’t useful
- Write **allocation proposals** for computer time that:
  - jump thru hoops to show code can use 50%+ of the machine
  - run lots of small jobs (together) to boost machine usage
What is the impact on science, mission or otherwise?

- **Opportunity cost** for continually adapting apps to new machines
  - doesn’t advance the models
  - nor the numerical methods & algorithms
  - nor the science (at least directly)
What is the impact on science, mission or otherwise?

- **Opportunity cost** for continually adapting apps to new machines
  - doesn’t advance the models
  - nor the numerical methods & algorithms
  - nor the science (at least directly)

- Deploying computers that are **hard to use and code for**:
  - shrinks the number of apps that can use it
  - we do less multiphysics, less multiscale
  - shrinks the pool of people that can program them
  - **casual HPC users** are not Jake Shimabukuros, but they are often **better scientists**
  - if make machines hard for them to code & use, we lose their science contributions
Balance ratios on past, present, future HPC platforms

Thanks to Si Hammond (Sandia) for this data!
Why you might be singing the Ukulele Blues

Balance ratios on past, present, future HPC platforms

Thanks to Si Hammond (Sandia) for this data!

- **Local balance** = flops to pay for on-node word (8 bytes)
- **Remote balance** = flops to pay for off-node word
**Local balance** = flops to pay for on-node word (8 bytes)

**Remote balance** = flops to pay for off-node word

<table>
<thead>
<tr>
<th>Year</th>
<th>Machine</th>
<th>Linpack</th>
<th>Flops/local</th>
<th>Flops/remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Cray YMP</td>
<td>2.1 Gflops</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td>1997</td>
<td>ASCI Red (SNL)</td>
<td>1.6 Tflops</td>
<td>8.3</td>
<td>20</td>
</tr>
<tr>
<td>2011</td>
<td>RoadRunner (LANL)</td>
<td>1.0 Pflops</td>
<td>6.7</td>
<td>170</td>
</tr>
</tbody>
</table>
**Current blues**

- **Local balance** = flops to pay for on-node word (8 bytes)
- **Remote balance** = flops to pay for off-node word

<table>
<thead>
<tr>
<th>Year</th>
<th>Machine</th>
<th>Linpack</th>
<th>Flops/local</th>
<th>Flops/remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Cray YMP</td>
<td>2.1 Gflops</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td>1997</td>
<td>ASCI Red (SNL)</td>
<td>1.6 Tflops</td>
<td>8.3</td>
<td>20</td>
</tr>
<tr>
<td>2011</td>
<td>RoadRunner (LANL)</td>
<td>1.0 Pflops</td>
<td>6.7</td>
<td>170</td>
</tr>
<tr>
<td>2012</td>
<td>Sequoia (LLNL)</td>
<td>17 Pflops</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>2013</td>
<td>Titan (ORNL)</td>
<td>18 Pflops</td>
<td>29</td>
<td>490</td>
</tr>
</tbody>
</table>
Asian blues

- **Local balance** = flops to pay for on-node word (8 bytes)
- **Remote balance** = flops to pay for off-node word

<table>
<thead>
<tr>
<th>Year</th>
<th>Machine</th>
<th>Linpack</th>
<th>Flops/local</th>
<th>Flops/remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Cray YMP</td>
<td>2.1 Gflops</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td>1997</td>
<td>ASCI Red (SNL)</td>
<td>1.6 Tflops</td>
<td>8.3</td>
<td>20</td>
</tr>
<tr>
<td>2011</td>
<td>RoadRunner (LANL)</td>
<td>1.0 Pflops</td>
<td>6.7</td>
<td>170</td>
</tr>
<tr>
<td>2012</td>
<td>Sequoia (LLNL)</td>
<td>17 Pflops</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>2013</td>
<td>Titan (ORNL)</td>
<td>18 Pflops</td>
<td>29</td>
<td>490</td>
</tr>
<tr>
<td>2011</td>
<td>K-Computer (Japan)</td>
<td>11 Pflops</td>
<td>15</td>
<td>95</td>
</tr>
<tr>
<td>2013</td>
<td>Tianhe-2 (China)</td>
<td>34 Pflops</td>
<td>22</td>
<td>2100</td>
</tr>
<tr>
<td>2016</td>
<td>Sunway TaihuLight (China)</td>
<td>93 Pflops</td>
<td>130</td>
<td>1500</td>
</tr>
</tbody>
</table>
### Exascale blues

<table>
<thead>
<tr>
<th>Year</th>
<th>Machine</th>
<th>Linpack</th>
<th>Flops/local</th>
<th>Flops/remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Cray YMP</td>
<td>2.1 Gflops</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td>1997</td>
<td>ASCI Red (SNL)</td>
<td>1.6 Tflops</td>
<td>8.3</td>
<td>20</td>
</tr>
<tr>
<td>2011</td>
<td>RoadRunner (LANL)</td>
<td>1.0 Pflops</td>
<td>6.7</td>
<td>170</td>
</tr>
<tr>
<td>2012</td>
<td>Sequoia (LLNL)</td>
<td>17 Pflops</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>2013</td>
<td>Titan (ORNL)</td>
<td>18 Pflops</td>
<td>29</td>
<td>490</td>
</tr>
<tr>
<td>2011</td>
<td>K-Computer (Japan)</td>
<td>11 Pflops</td>
<td>15</td>
<td>95</td>
</tr>
<tr>
<td>2013</td>
<td>Tianhe-2 (China)</td>
<td>34 Pflops</td>
<td>22</td>
<td>2100</td>
</tr>
<tr>
<td>2016</td>
<td>Sunway TaihuLight (China)</td>
<td>93 Pflops</td>
<td>130</td>
<td>1500</td>
</tr>
<tr>
<td>~2021</td>
<td>TBD Door #1</td>
<td>1.0 Eflops</td>
<td>80</td>
<td>3200</td>
</tr>
<tr>
<td>~2021</td>
<td>TBD Door #2</td>
<td>1.0 Eflops</td>
<td>300</td>
<td>10000</td>
</tr>
</tbody>
</table>
## Exascale blues

### Congratulations! billion X speed-up in 30 years!

<table>
<thead>
<tr>
<th>Year</th>
<th>Machine</th>
<th>Linpack</th>
<th>Flops/local</th>
<th>Flops/remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Cray YMP</td>
<td>2.1 Gflops</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td>1997</td>
<td>ASCI Red (SNL)</td>
<td>1.6 Tflops</td>
<td>8.3</td>
<td>20</td>
</tr>
<tr>
<td>2011</td>
<td>RoadRunner (LANL)</td>
<td>1.0 Pflops</td>
<td>6.7</td>
<td>170</td>
</tr>
<tr>
<td>2012</td>
<td>Sequoia (LLNL)</td>
<td>17 Pflops</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>2013</td>
<td>Titan (ORNL)</td>
<td>18 Pflops</td>
<td>29</td>
<td>490</td>
</tr>
<tr>
<td>2011</td>
<td>K-Computer (Japan)</td>
<td>11 Pflops</td>
<td>15</td>
<td>95</td>
</tr>
<tr>
<td>2013</td>
<td>Tianhe-2 (China)</td>
<td>34 Pflops</td>
<td>22</td>
<td>2100</td>
</tr>
<tr>
<td>2016</td>
<td>Sunway TaihuLight (China)</td>
<td>93 Pflops</td>
<td>130</td>
<td>1500</td>
</tr>
<tr>
<td>~2021</td>
<td>TBD Door #1</td>
<td>1.0 Eflops</td>
<td>80</td>
<td>3200</td>
</tr>
<tr>
<td>~2021</td>
<td>TBD Door #2</td>
<td>1.0 Eflops</td>
<td>300</td>
<td>10000</td>
</tr>
</tbody>
</table>
Commentary on the blues

- You might be saying ...

- Cray YMP main memory was like small/fast cache memory today
  - 1 core of YMP: 512 MB of memory, balance factor = 0.52
  - 1 core of modern Intel CPU: 32 KB of L1 cache, bfactor = 1.0

Growing imbalance ratios mean:
- fewer codes achieve high single-node performance
- fewer codes achieve good scalability

Bottom line: we're selecting for certain kinds of apps that can withstand these high imbalance ratios
Commentary on the blues

- You might be saying ...
- Cray YMP main memory was like small/fast cache memory today
  - 1 core of YMP: 512 MB of memory, balance factor = 0.52
  - 1 core of modern Intel CPU: 32 KB of L1 cache, bfactor = 1.0

- Growing imbalance ratios mean:
  - fewer codes achieve high single-node performance
  - fewer codes achieve good scalability
You might be saying ...

Cray YMP main memory was like small/fast cache memory today
- 1 core of YMP: 512 MB of memory, balance factor = 0.52
- 1 core of modern Intel CPU: 32 KB of L1 cache, bfactor = 1.0

Growing imbalance ratios mean:
- fewer codes achieve high single-node performance
- fewer codes achieve good scalability

Bottom line: we’re **selecting** for certain kinds of apps that can withstand these high imbalance ratios
But hey ... growing imbalance is good news for particles

- Particle apps:
  - lots of flops per memory access (expensive models)
  - particle/particle interactions are local (comm is local)
  - zillions of particles $\Rightarrow$ lots of threads

So I shouldn't be complaining ... we're thinning the herd of apps, less competition for cycles

But ... particles don't represent broad swath of computational science, or majority of apps that need HPC

physics often isn't short-range hard to reach long timescales with explicit timestepping
But hey ... growing imbalance is good news for particles

- Particle apps:
  - lots of flops per memory access (expensive models)
  - particle/particle interactions are local (comm is local)
  - zillions of particles $\Rightarrow$ lots of threads

- So I shouldn’t be complaining ...
  we’re **thinning the herd** of apps, less competition for cycles
But hey ... growing imbalance is good news for particles

- Particle apps:
  - lots of flops per memory access (expensive models)
  - particle/particle interactions are local (comm is local)
  - zillions of particles ⇒ lots of threads

- So I shouldn’t be complaining ... we’re thinning the herd of apps, less competition for cycles

- But ...
  - particles don’t represent broad swath of computational science, or majority of apps that need HPC
  - physics often isn’t short-range
  - hard to reach long timescales with explicit timestepping
Coding apps for the bleeding-edge of HPC

- **Vectorize** for YMP (medium vector length)
- **Vectorize** for SIMD (deja vu, long vectors)
- **Vectorize** for CPU/KNL (deja deja vu, short vectors)
- Learn **MPI** (distributed memory)
- Add **OpenMP** directives (modest threading)
- Learn **CUDA** for GPUs (massive threading)
- Overlap comp and comm (hide latencies)
- Manage memory for CPUs (4 level caches and growing)
- Hybrid nodes (CPU + multiple GPUs)
- Make codes **fault tolerant** (what?)
- Convert to **asynchronous multi tasking** (really?)
- **MPI may vanish** (#@!% really??)
Coding apps for the bleeding-edge of HPC

- **Vectorize** for YMP (medium vector length)
- **Vectorize** for SIMD (deja vu, long vectors)
- **Vectorize** for CPU/KNL (deja deja vu, short vectors)
- Learn **MPI** (distributed memory)
- Add **OpenMP** directives (modest threading)
- Learn **CUDA** for GPUs (massive threading)
- Overlap comp and comm (hide latencies)
- Manage memory for CPUs (4 level caches and growing)
- Hybrid nodes (CPU + multiple GPUs)
- Make codes **fault tolerant** (what?)
- Convert to **asynchronous multi tasking** (really?)
- **MPI may vanish** (#@%! really??)

- Hardware/Architects: this is the **price** apps have to pay to keep up with our amazing hardware
Coding apps for the bleeding-edge of HPC

- **Vectorize** for YMP  
  (medium vector length)
- **Vectorize** for SIMD  
  (deja vu, long vectors)
- **Vectorize** for CPU/KNL  
  (deja deja vu, short vectors)
- Learn **MPI**  
  (distributed memory)
- Add **OpenMP** directives  
  (modest threading)
- Learn **CUDA** for GPUs  
  (massive threading)
- Overlap comp and comm  
  (hide latencies)
- Manage memory for CPUs  
  (4 level caches and growing)
- Hybrid nodes  
  (CPU + multiple GPUs)
- Make codes **fault tolerant**  
  (what?)
- Convert to **asynchronous multi tasking**  
  (really?)
- **MPI may vanish**  
  (#@!% really??)

- Hardware/Architects: this is the **price** apps have to pay to keep up with our amazing hardware
- App developers: this is a ton of not-so-useful **work**
Coding apps for the bleeding-edge of HPC

- **Vectorize** for YMP (medium vector length)
- **Vectorize** for SIMD (deja vu, long vectors)
- **Vectorize** for CPU/KNL (deja deja vu, short vectors)
- Learn **MPI** (distributed memory)
- Add **OpenMP** directives (modest threading)
- Learn **CUDA** for GPUs (massive threading)
- Overlap comp and comm (hide latencies)
- Manage memory for CPUs (4 level caches and growing)
- Hybrid nodes (CPU + multiple GPUs)
- Make codes **fault tolerant** (what?)
- Convert to **asynchronous multi tasking** (really?)
- **MPI may vanish** (#@%! really??)

- Hardware/Architects: this is the **price** apps have to pay to keep up with our amazing hardware
- App developers: this is a ton of not-so-useful **work**
- Scientists: this is a **barrier to the science** I want to do
Qualitative history of apps on evolving HPC platforms

X-axis = paradigm shifts in HPC node hardware
Y-axis = percentage of scientific apps that adapt
Qualitative history of apps on evolving HPC platforms

X-axis = paradigm shifts in HPC node hardware
Y-axis = percentage of scientific apps that adapt

Y-axis = percentage of apps that adapt and run efficiently on full machine
“I find the three major administrative problems on a campus are ...
Hijacking another quote to explain HPC

Clark Kerr
president of UC Berkeley
later of UC system

“I find the three major administrative problems on a campus are ...

sex for the students,
athletics for the alumni, and
parking for the faculty.”
Supercomputers also have 3 constituencies (at least)

"I find the three major administrative problems when building a new supercomputer are ... flops for the funders, branding for the vendors, and access (parking) for the scientists."
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ...
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ...

**flops** for the funders,
**branding** for the vendors, and
**access** (parking) for the scientists.”
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ... 

**flops** for the funders, 
**branding** for the vendors, and 
**access** (parking) for the scientists.”

- Funders (students): flops are sexy, grab the headlines
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ... 

**flops** for the funders,
**branding** for the vendors, and
**access** (parking) for the scientists.”

- Funders (students): flops are sexy, grab the headlines
- Vendors (alumni): our technology is in the fastest machines
Thinking outside-the-box for branding ...

Introducing the new

Chick-fil-A®  IBM®  NVIDIA®

Summit supercomputer at ORNL
Introducing the new Sierra supercomputer at LLNL

“A new way to get high (flop rates)”
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ...

- **flops** for the funders,
- **branding** for the vendors, and
- **access** (parking) for the scientists.”

- Funders (students): flops are sexy, grab the headlines
- Vendors (alumni): our technology is in the fastest machines
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ... 

flops for the funders, 
branding for the vendors, and 
access (parking) for the scientists.”

- Funders (students): flops are sexy, grab the headlines
- Vendors (alumni): our technology is in the fastest machines
- Scientists (faculty):
  - access is more than time on the machine
  - also ease-of-use, ability to do many kinds of science
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ...

flops for the funders, 
branding for the vendors, and 
access (parking) for the scientists.”

- Funders (students): flops are sexy, grab the headlines
- Vendors (alumni): our technology is in the fastest machines
- Scientists (faculty):
  - access is more than time on the machine
  - also ease-of-use, ability to do many kinds of science

Can you make all 3 happy?
Supercomputers also have 3 constituencies (at least)

“I find the three major administrative problems when building a new supercomputer are ...

flops for the funders, 
branding for the vendors, and 
access (parking) for the scientists.”

- Funders (students): flops are sexy, grab the headlines
- Vendors (alumni): our technology is in the fastest machines
- Scientists (faculty):
  - access is more than time on the machine
  - also ease-of-use, ability to do many kinds of science

Can you make all 3 happy?
- Maybe at a rich university, but not with a supercomputer
- Because the 3 constituencies have competing interests
Cell biology

- PCR (1983) = polymerase chain reaction, DNA replication
- Microarray chips (1995) = parallel gene expression (millions)
- DNA sequencing (2001) = $10K/Mb $\Rightarrow$, few $0.01/Mb$
- CRISPR (2012) = genome editing in living cells
Cell biology

- PCR (1983) = polymerase chain reaction, DNA replication
- Microarray chips (1995) = parallel gene expression (millions)
- DNA sequencing (2001) = $10K/Mb ⇒, few $0.01/Mb
- CRISPR (2012) = genome editing in living cells

- All these technologies rapidly became ubiquitous
- Any lab, any grad student can use them
- Don’t need add-on experts to write an NIH proposal
PCR (1983) = polymerase chain reaction, DNA replication
Microarray chips (1995) = parallel gene expression (millions)
DNA sequencing (2001) = $10K/Mb \Rightarrow$, few $0.01/Mb$
CRISPR (2012) = genome editing in living cells

All these technologies rapidly became **ubiquitous**
Any lab, any grad student can use them
Don’t need add-on experts to write an NIH proposal
Could we aspire to that **ease-of-use** for HPC machines?
User facilities with billion $ instruments

- Hubble telescope (NASA/ESA),
  SNS (ORNL), Z-machine (Sandia)

What if 20x new HPC machine just gave all users 20x more?
User facilities with billion $ instruments

- Hubble telescope (NASA/ESA),
  SNS (ORNL), Z-machine (Sandia)

- Hubble: 1.3M observations, SNS: 20K users, Z: 3160 shots
User facilities with billion $ instruments

- Hubble telescope (NASA/ESA), SNS (ORNL), Z-machine (Sandia)
- Hubble: 1.3M observations, SNS: 20K users, Z: 3160 shots
- All solicit user proposals (Hubble from amateurs!)
- Facilities shield users from nearly all complexity
User facilities with billion $ instruments

- Hubble telescope (NASA/ESA),
  SNS (ORNL), Z-machine (Sandia)

- Hubble: 1.3M observations, SNS: 20K users, Z: 3160 shots
- All solicit user proposals (Hubble from amateurs!)
- Facilities shield users from nearly all complexity
- What if 20x new HPC machine just gave all users 20x more?
High-energy particle physics

- CERN, FermiLab, etc

Every new accelerator requires **one-of-a-kind new detectors** to be useful
- Detector = 100s of people, $100 million or more
- Performs handful of (high-impact, highly complex) science experiments in a narrow sub-field of physics
Every new accelerator requires one-of-a-kind new detectors to be useful.

Detector = 100s of people, $100 million or more

Performs handful of (high-impact, highly complex) science experiments in a narrow sub-field of physics

Is HPC more like cell bio, user facilities, or HE physics?
No silver bullet solutions ...

- Can we create easy-to-use machines & software for the 99% of mere-mortal computational scientists across many fields to do amazing science?
Can we create easy-to-use machines & software for the 99% of \textbf{mere-mortal computational scientists} across many fields to do amazing science?

\textbf{Metric}: A tool is most powerful, when users don't need experts to use it.
Can we create easy-to-use machines & software for the 99% of mere-mortal computational scientists across many fields to do amazing science?

**Metric**: A tool is most powerful, when users don’t need experts to use it.

Hope you view my remarks as **inducements** to:
- insulate users from growing complexity of HPC machines
- make life easier for the apps and the science
Can we create easy-to-use machines & software for the 99% of mere-mortal computational scientists across many fields to do amazing science?

**Metric**: A tool is most powerful, when users don’t need experts to use it.

Hope you view my remarks as inducements to:
- insulate users from growing complexity of HPC machines
- make life easier for the apps and the science

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