Training
Code Performance and Scaling

Online Training 5/19/20
Ken Hackworth & Lars Koesterke
hackwort@psc.edu  lars@tacc.utexas.edu
XSEDE has an external code of conduct for XSEDE sponsored events which represents XSEDE’s commitment to providing an inclusive and harassment-free environment in all interactions regardless of gender, sexual orientation, disability, physical appearance, race, or religion. The code of conduct extends to all XSEDE-sponsored events, services, and interactions.

**Code of Conduct:** [https://www.xsede.org/codeofconduct](https://www.xsede.org/codeofconduct)

**Contact:**
- Event organizer: *Ken Hackworth – XSEDE Allocations Manager*
- XSEDE ombudspersons:
  - Linda Akli, Southeastern Universities Research Association, ([akli@sura.org](mailto:akli@sura.org))
  - Lizanne Destefano, Georgia Tech ([lizanne.destefano@ceismc.gatech.edu](mailto:lizanne.destefano@ceismc.gatech.edu))
  - Ken Hackworth, Pittsburgh Supercomputing Center, ([hackworth@psc.edu](mailto:hackworth@psc.edu))
  - Bryan Snead, Texas Advanced Computing Center ([jbsnead@tacc.utexas.edu](mailto:jbsnead@tacc.utexas.edu))
Extreme Science and Engineering Discovery Environment (XSEDE)’s Mission:

“To substantially enhance the productivity of a growing community of scholars, researchers, and engineers through access to advanced digital services that support open research; and coordinate and add significant value to the leading cyberinfrastructure resources funded by the NSF and other agencies.”
Digital Services

- Digital services provide users with seamless integration to NSF's high-performance computing and data resources. XSEDE's integrated, comprehensive suite of advanced digital services combined with other high-end facilities and campus-based resources, serve as the foundation for a national cyberinfrastructure ecosystem.
How to connect with these Digital Services, The Resources?

• A diverse portfolio of XSEDE-allocated resources is available to support your research.
  • HPC systems
  • HTC systems
  • GPU and AI systems
  • Visualization and data analysis systems
  • Large-memory systems
  • Data storage
  • Cloud systems
  • XSEDE Staff (ECSS and SGCI) “Best XSEDE Resource”

• Most US-based researchers are eligible for no-cost allocations via XSEDE. Get started in two weeks or less! Startup allocations.

• Large production allocations or also know Research Allocations, can be requested yearly, with available submission period every 3 months.
Why request a Startup Allocation?

• How to get one?
  • Requests accepted anytime, NO Deadlines!
  • Reviewed in less than two weeks!
  • Multiple resources can be requested!
  • Eligible PI/CV and Extended Abstract is all that is needed for submission!

• Purpose
  • Small-scale computational activities
  • Application development
  • Evaluation and experimentation on the various resources
  • Benchmarking – Code Performance and Scaling
Research Allocations

• For projects that have progressed beyond the Startup phase, either in purpose or scale of computation activities!
  • Reviewed quarterly by XRAC panel of peer reviewers
  • One award per PI/Research team
  • Highly competitive process, > 200 proposals per quarter considered
  • Proposal quality matters!
  • > 80% awarded, large number declined because?
    • Poor Code Performance and Scaling!
    • 32% of New proposals in last year rejected because of poor code performance and scaling information.
A well justified computational plan …

A well justified computational plan is the backbone of a successful XSEDE proposal

• Science is important, but the reviewers focus on the computational plan
• Funded projects (NSF, NASA, DoD, DoE, NIH, etc.)
  • Science is not reviewed per se
  • But computations should match science
• Demonstrating competent use of the expensive resources is paramount

We delve into the components of a ‘well justified computational plan’ in a minute …

Hands-on advice
What your proposal is not:
NSF proposal + two sentences presenting a number for the Service Units (SU)
Getting started with a Start-Up

Use a Start-Up to explore resources and to compile data for a successful proposal

• Request a Start-Up at any time
• Quick review and turn around
• 4-5 paragraphs detailing
  • Field of science
  • Your work
  • What you want to do. Relevant in the context of this talk: preparing a full proposal
  • Testing software, scaling studies, developing code, etc.
• Generous amount to get started
  • GPU’s also available
  • Specialized systems for Hadoop-style research and AI, etc.
From a Start-Up to a Full Proposal

Use common sense when outlining a full proposal

• Requests from new groups are under more scrutiny than proposals from well-established groups
• Well-established groups have a better chance of winning very large awards
• New PI’s should establish a track record first
• Compare your request size to the total availability of the resource(s)
  • Use available hours in a year
  • Use available number of nodes in resource
  • Limit yourself to a reasonable fraction of the target system
• Smaller requests certainly face less scrutiny than larger requests
• Focus on what you really need. One resource or a few. Do not target several similar resources without a good reason
  • ‘It is convenient’ or ‘What if a system is down?’ are not good reasons
Use common sense when preparing your proposal

Pick a reasonable request size

Acknowledge the fierce competition for the resources

Overly ambitious proposals (particularly from new PIs) do not fare well
What makes a good proposal?

You are competing with many outstanding researchers
Writing a better proposal pays off

1. Description of science
2. Detailing the specific scientific questions
3. Description of the software
4. Computational plan
5. Estimate of Service Units (SU)
6. Supporting data, i.e. scaling and computational efficiency to your

Details

1. Proposal comparable to your NSF proposal, but with a different focus (computational plan). Keep in mind that reviewers are not necessarily domain experts. A broader approach is appreciated by the reviewers.

2. Connect your scientific questions to the computations that you are proposing.

3. What equations/algorithms are being applied? How is the code parallelized? Describe software tools and programming environment, if applicable. Did you do anything special to increase performance. Assure the reviewers that you know how to run the code on the target system.

4. Present a computational plan that details:
   • Number of calculations and run times
   • Node count, and how this relates to your scaling data

5. Use basic math and allow reviewers to verify your total SU estimate.

6. Scaling data ‘proves’ to the reviewer efficient use of the resource. Justify the size for an individual calculation based on:
   • Scaling efficiency
   • Total run-time
   • Reviewers will appreciate all efforts you document.
Hands-on advice

The computational plan takes center stage. Your computational plan and resource justification must not be limited to a terse

... and therefore we request x-million SUs on the resource ...

Reviewers want to be assured that you know what you are doing
Demonstrate efficient use of the resource through scaling

It is difficult to judge the efficiency of an application

The review panel is using ‘scaling efficiency’ as a proxy

Reviewers like to see scaling on the target system, or a similar XSEDE system

Present scaling for a typical case

Present timings for a few time steps
Present timings for a smaller sample
In complex situations, i.e. Adaptive Mesh Refinement (AMR), choose a simplified setup and discuss implications

The documented scaling should either
• cover the number of nodes targeted for production, or
• apply a well justified ‘extrapolation’ from a smaller sample or fewer time steps

Target a node count for production where the scaling is good
Balance wall-clock time and scaling/parallel efficiency (see also remarks later)
Speed-up and Parallel Efficiency (I)

Strong scaling

Workload is constant, even as more and more resources are applied. Ideally execution time decreases with number of resources. Resources are either cores or nodes.

- **Speed-up** is the ratio of execution times, i.e. $t_1$, $t_2$, ...
- Use $t_1/t_n$ in this setup, ideal strong scaling is linear, and hence easy to assess.
- Do not use $t_n/t_1$ in this setup, ideal strong scaling is not linear, but $1/x$.
- Usually linear axis are chosen.

Note that strong (and weak) scaling experiments on a single node will always flatten.

- Most likely you will run out of memory bandwidth before utilizing all cores on a node.

Strong scaling experiments are often conducted for smaller calculations on a single node.
Scaling Plots

Demonstrate either strong or weak scaling, or both

Strong scaling
Amount of work is fixed

Weak scaling
Amount of work scales linearly with number of cores/nodes

Make the reviewer’s job easy
Show a plot, where the ideal scaling is a straight line
• Either linear axis or log-log
Do not use execution time (1/x) in your plot (ideal scaling is a curve)

Example of ‘Strong Scaling’

Ideal scaling

Good scaling

Reasonable scaling

Bad scaling

Speed-up

cores/nodes

Do not pick ‘maximum’ speed, where the wall-clock is shortest, but the efficiency is low
The Problem with $1/x$

or any ‘technique’ that obscures the scaling or compute efficiency

Non-linear ‘target’

Deviations from the ‘ideal’ are hard to judge

Reviewers may severely cut or reject a proposal

• If it takes ‘too much effort’ to make sense of the data
• If a reviewer become suspicious

Easy to assess scaling

Difficult to assess scaling
Weak scaling

Workload is proportional to the amount of work as more and more resources are applied. Ideally, execution time stays constant with number of resources.

Resources are either cores or nodes.

- **Parallel efficiency** is the ratio of execution times, i.e. $t_1, t_2, \ldots$
- Use $t_1/t_n$. In this setup the ideal parallel efficiency is 1, and non-ideal is lower than 1.
- You may use a linear or logarithmic scale for the x-axis.

Weak scaling experiments are often conducted for larger calculations on multiple nodes.

In some fields it is common to use ‘progress’ rather than execution time.

- Time steps per hour
- Simulation time per second (fs/s in Molecular dynamics)

This is fine, but do not confuse reviewers. Make units accessible to non-domain experts.
Scaling Plots

Weak scaling
Amount of work scales with number of cores/nodes

Ideal ‘Weak Scaling’
Execution time is constant and ideally equal to the time on a single* node

Parallel efficiency =
‘ideal time’ / ‘actual time’
‘ideal time’ is the time on the smallest number of nodes

Present with a linear or log x-axis
* or the time on the smallest number of nodes that is feasible
**Hands-on advice**

Use common sense when picking a target scale

Particularly for large calculations, with a high SU consumption, do **not** target quick completion at the expense of efficiency

The goal is to demonstrate that you will use the resources efficiently

Review panel is not overly concerned with wall-clock time. The award period is a year and you expect to run through the whole year.

**Non-ideal scaling is a fact of life**

Find a good compromise and present your data in a favorable, yet transparent, way
Hands-on advice

Balance efficiency and runtime (wall clock)
- When in doubt stay on the ‘efficient’ side
- Allocation is for a year. Do not push for early completion at the expense of efficiency
- Reduce wall-clock time, by advancing multiple calculations simultaneously (if applicable)
- For very long calculations also consider ‘time in queue’. 24-48 hours in queue for 24 hours running

Special considerations for time critical projects
- Time critical project, e.g. severe weather forecast
- Push a bit farther, but a 20% wall-clock reduction for 2x resources is difficult to justify

Pushing into inefficient territory
- You will be told to optimize your application, and may receive a severely reduced allocation

Small, but numerous calculations
- Bundle multiple calculations on a single node (often nodes are not shared)
- Investigate and present most efficient setup
Estimation of SUs

Present ‘simple’ math to estimate the total request
Use the timing from the scaling experiments
Extrapolate towards the total number of time steps or individual experiments

Hands-on Advice

Present details for all planned experiments
Allow the reviewer to easily follow your estimate
Give the reviewer reassurance that your estimate is not bloated
Add a table to summarize the individual experiments
For parameter studies
• Highlight that you have selected a reasonable/small number of parameter combinations
• Potentially discuss how first results may inform the selection of other parameter combinations

Do not forget to justify storage
Example

Simple math for every experiment and method

• ‘# of nodes’ \( \times \) ‘Total runtime’ \( \times \) ‘# of simulations = ‘SU’

Scaling

• Justification of ‘# of nodes’

Scaling and performance

• Use performance data to map ‘Simulation time’ to ‘Total runtime’

Resolution and simulation time is justified in the proposal (sciency bits)

<table>
<thead>
<tr>
<th>Exp 1 Method</th>
<th>Size Resolution</th>
<th>Simulation time</th>
<th># of Nodes</th>
<th>Total runtime</th>
<th># of simulations</th>
<th>SU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluid 1 Explicit 3D</td>
<td>1000^3</td>
<td>5 min</td>
<td>10</td>
<td>100 h</td>
<td>3</td>
<td>3,000</td>
</tr>
<tr>
<td>Fluid 1 Implicit 3D</td>
<td>1000^3</td>
<td>5 min</td>
<td>10</td>
<td>20 h</td>
<td>3</td>
<td>600</td>
</tr>
<tr>
<td>Fluid 2 Implicit 3D</td>
<td>10000^3</td>
<td>10 min</td>
<td>100</td>
<td>2400 h</td>
<td>1</td>
<td>240,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exp 2 Method</th>
<th>Size Resolution</th>
<th>Simulation time</th>
<th># of Nodes</th>
<th>Total runtime</th>
<th># of simulations</th>
<th>SU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluid 1 Explicit 1D</td>
<td>3000</td>
<td>1 h</td>
<td>1/2</td>
<td>5 min</td>
<td>36,000</td>
<td>1,500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exp 3 Method</th>
<th>Size Resolution</th>
<th>Simulation time</th>
<th># of Nodes</th>
<th>Total runtime</th>
<th># of simulations</th>
<th>SU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protein MD</td>
<td>10,000 part</td>
<td>200 fs</td>
<td>100</td>
<td>192 h</td>
<td>5</td>
<td>96,000</td>
</tr>
</tbody>
</table>

Provide relevant information for all methods and experiments
Scaling: Community Codes and ‘Well-known’ Codes

Scaling for these codes is often well documented

Nevertheless you should document scaling to demonstrate

- Scaling for your specific setup (community codes often cover a wide range of numerical setups) and size
- Scaling on the target machine (or on a machine with a very similar architecture)
- Accurate estimate of SU request size

Present scaling for a typical case

Use your results for your estimate of the SU costs
Do not refer to external documents where scaling for your widely used application is documented
Reviewers want to read about your experiences on a specific system
Additional Considerations (I)

Your request may not just be about compute SUs
Add and justify any additional requirements
• Storage, either long-term (tape) or medium-term (disk)
• Special queueing demands (only for very well justified requests)

Disclose any problems that you may anticipate
Resource providers are happy to work with you before you cause trouble on the system
You will not be ‘penalized’ by adding special requirements to the proposals

Discuss bottlenecks that you may anticipate
• I/O: users often overwhelm the file systems
• File transfer
• etc
Additional Considerations (II)

Hands-on Advice

Add things that you have done in the past to improve performance
A little bit of bragging goes a long way …

Have you done any of these?
• Efforts to understand code and workflow performance characteristics
• Profiling of code to measure performance
• Code performance optimization
• Scaling optimization
• Efforts to minimize and optimize I/O

Write about your efforts!
Summary

Design your proposal around the computational plan
Demonstrate efficient use through scaling (proxy)
Detail basic math to estimate total SU request size
Be reasonable with your request size
Ramp up over time

Accept that there is tough competition
A little bit of effort goes a long way …

Questions?