FINAL COMMENTS

Optimization Strategy

- During optimization, your goal is to minimize the amount of work the computer is required to do. The strategy we recommend is a two-step approach:

  1. Write code that is efficient from the start (e.g., use vectorizations instead of loops)

  2. After your code is debugged and working, try more aggressive optimization techniques (e.g., manipulating the mathematical formulas to reduce calls to built-in math functions).

Disadvantages?

- Optimizing code is time consuming

  Do not waste weeks optimizing code that will run once for 1 hour.

- Some optimizations can make the code harder to read and debug.

- Be aware that different architectures can respond in different ways.

  Just because code is optimized on your laptop does not necessarily mean that it is optimized on your colleague's computer.

- Some optimizations can adversely affect parallel scaling.

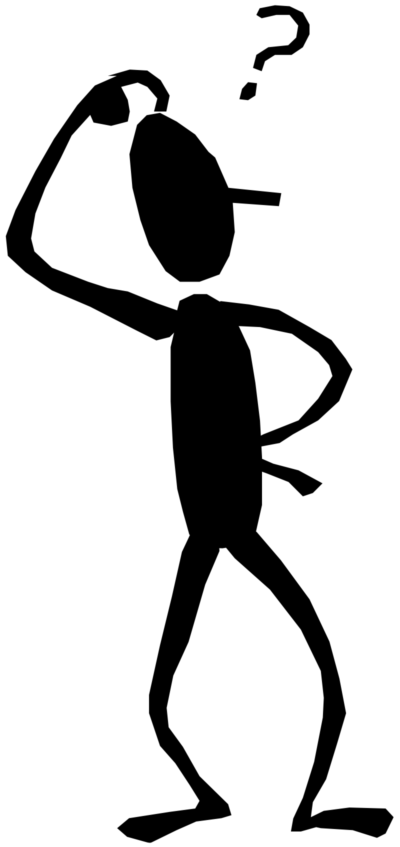
When to optimize?

- Code optimization is an iterative process requiring time, energy and thought. It is recommended for:
  - Codes that will be widely distributed and used often by the research community.
  - Projects that have limited allocation, so that you can maximize the available time on the compute resources.

When optimization isn’t enough

- When you have done everything possible to optimize your code, and it still isn’t fast enough, you can
  - Find a better algorithm (if one exists).
  - Look into parallelizing your code.

Questions?



UNIVERSITY of VIRGINIA
ADVANCED RESEARCH COMPUTING SERVICES