The Co-Design Process for Scientists and Project Leads

John Dennis and Cena Brown,
Application Scalability And Performance (ASAP) Group, CISL
Should I be interested in GPU-enabling my science?
What do we mean by co-design?

- Designing projects based on hardware characteristics, software constraints, and science objectives.
- What science could GPU-enablement really advance?
  - Some science objectives are well suited or GPU friendly
  - Other science objectives are not particularly GPU friendly
- This is not “Let’s do GPU-programming because everybody else is doing it”
What do we mean by co-design?

- Designing projects based on hardware characteristics, software constraints, and science objectives.
- What science could GPU-enablement really advance?
  - Some science objectives are well suited or GPU friendly
  - Other science objectives are not particularly GPU friendly
- This is not “Let’s do GPU-programming because everybody else is doing it”

back-of-the-envelope calculation ahead
Multiple successful earth system applications that have been GPU-enabled

- FastEddy
  - Large eddy simulation (LES) code for microscale flows
- MURaM
  - Multidimensional MHD to study solar magneto-convection and other related magnetic activities
- CM1
  - Mesoscale atmospheric model used for idealized process studies
- MPAS-A
  - Atmospheric component of the Model for Prediction Across Scales
- SAMURAI
  - Variational data assimilation of APAR observations
- HOMME++
  - Spectral element dynamical core used by the E3SM project
Common features of these GPU projects

• Compatible scientific objective
  – Have identified when a science objective is a good fit for GPU-enablement

• Knowledgeable, interdisciplinary team
  – Project design for GPU-enablement
  – Knowledge about how to perform the transformation
    • How to program in OpenACC, OpenMP offload, or CUDA

• Clearly defined achievable goals

• Significant stakeholder engagement

• Significant software engineering investments

Co-design for Scientists and Project Leads
Common features of these GPU projects

• Compatible scientific objective
  – Have identified when a science objective is a good fit for GPU-enablement
• Knowledgeable, interdisciplinary team
  – Project design for GPU-enablement
  – Knowledge about how to perform the transformation
    • How to program in OpenACC, OpenMP offload, or CUDA
• Clearly defined achievable goals
• Significant stakeholder engagement
• Significant software engineering investments

See other workshop sessions
Common features of these GPU projects

- Compatible scientific objective
  - Have identified when a science objective is a good fit for GPU-enablement
- Knowledgeable, interdisciplinary team
  - Project design for GPU-enablement
  - Knowledge about how to perform the transformation
    - How to program in OpenACC, OpenMP offload, CUDA
- Clearly defined achievable goals
- Significant stakeholder engagement
- Significant software engineering investments
Outline

• Motivation
• How to identify GPU friendly science objectives
• Estimating effort to achieve GPU-enablement
• Estimating return on investment (ROI)
A collection of scientific objectives

• Unlikely to be GPU friendly
  – Paleo-climate
  – Climate change

• Likely to be GPU friendly
  – Climate variability using large-ensembles
  – Ocean modeling process studies
  – High-resolution whole atmosphere modeling with Data Assimilation
  – Reanalysis
  – Compute-intensive post-processing
  – Data assimilation of observational data

• Very GPU friendly
  – Numerical weather prediction
  – Seasonal to sub-seasonal forecasting
  – Regional ocean modeling
  – LES modeling
  – High-resolution regional modeling with complex chemistry
  – Space-weather prediction
  – Magnetosphere modeling

Co-design for Scientists and Project Leads
How to determine if your science is GPU friendly

• Is it a computational demanding and why?
  – Potential scientific simulations
    • MPAS-A 3.75 km weather modeling
      – 38.6M x 56 ⇒ O(2162M) independent grid-points
      – ~300 GPUs per run: grid-points per GPU = O(7.2M)
      – O(1.22M) timesteps
    • CM1 ASD simulations
      – 2048x2048x1024 ⇒ O(4294M) independent grid-points
      – ~128 GPUs per run: grid-points per GPU = O(33M)
      – O(87K) timesteps
    • MURaM ASD simulations
      – 2352x2016x2016 ⇒ O(9559M) independent grid-points
      – ~252 GPUs per run: grid-points per GPU = O(37.9M)
      – O(250K) timesteps
How to determine if your science is GPU friendly

• Is it a computational demanding and why?
  – Potential scientific simulations
    • MPAS-A 3.75 km weather modeling
      – 38.6M x 56 $\Rightarrow$ O(2162M) independent grid-points
      – ~300 GPUs per run: grid-points per GPU = O(7.2M)
      – O(1.22M) timesteps
    • CM1 ASD simulations
      – 2048x2048x1024 $\Rightarrow$ O(4294M) independent grid-points
      – ~128 GPUs per run: grid-points per GPU = O(33M)
      – O(87K) timesteps
    • MURaM ASD simulations
      – 2352x2016x2016 $\Rightarrow$ O(9559M) independent grid-points
      – ~252 GPUs per run: grid-points per GPU = O(37.9M)
      – O(250K) timesteps
  – Computational demanding because of number of independent grid-points!

Co-design for Scientists and Project Leads
How to determine if your science is GPU friendly

• Is it a computational demanding and why?
  – Other potential configurations
    • 1-degree climate change:
      – 288x192x32 ⇒ O(1.7M) independent grid-points
      – 64 nodes per run: grid-points per node = O(27K)
      – ~O(17M) timesteps
      – Computationally demanding because of number of timesteps!
  – Does it perform a large amount of calculations between I/O?
    – Example
      
      ```
      read() Temp
      avgTemp = SUM(Temp(:,:,:));
      ```
    – Efficient use of GPU minimizes off device transfers
    – I/O bound problems are not typically a good match for GPUs
How to determine if your science is GPU friendly (con’t)

• Does the science have rate or throughput limitations?
  – If rate limitations
    • Execution rate GPU should match or exceed CPU rate ⇒ GPU friendly
    • Example:
      – Operational weather forecasting
      – Long climate simulations
  – If throughput limitations
    • Can more science be performed quicker or using less hardware
    • Example:
      – Large Eddy Simulation (LES)
      – large-ensemble climate modeling
      – seasonal to sub-seasonal forecasting
      – Magnetohydrodynamics (MHD)
Are your science objectives GPU friendly?  
[Student exercise: 13 minutes]

- Student exercise [5 minutes]
  
  - Determine the following
    - Total number of independent grid-points
    - \# \{nodes,GPU\} per run
    - \# grid-points per \{node,GPU\}
    - \# timesteps per run
  
  - Does it perform I/O frequently?
  
  - Do you have rate or throughput limitations?

- Discuss as a group any interesting realizations [7 minutes]
Outline

• Motivation
• How to identify GPU friendly science objectives
• Estimating effort to achieve GPU-enablement
• Estimating return on investment (ROI)
Estimating effort for GPU-enablement

- Does a GPU-enabled version of your code already exist?
  - Does this version of the code support all the necessary physics options?
- Is the code written in such a way that it is GPU-ready?
  - Is significant or full parallelism available at loop level?
  - Does a threaded (e.g. OpenMP) version of the code exist?
  - Does the code have some form of verification?
Needing to rewrite call structure to support significant parallelism at the loop level can be very time consuming.

Example: GPU ready loop arrangement

```
  do k=1,1024
    do j=1,128
      do i=1,1256
        wten(i,j,k)=wten(i,j,k)+(c1(i,j,k)*dum8(i,j,k-1)+c2(i,j,k)*dum8(i,j,k))
      enddo
    enddo
  enddo
```

Example: Loops in need of rearrangement

```
  do k=1,1024
    call radiation_solver()
    do j=1,128
      call lw_solve(a(1:256))
    enddo
  enddo
```
Needing to rewrite call structure to support significant parallelism at the loop level can be very time consuming.

Example: GPU ready loop arrangement

```fortran
do k=1,1024
  do j=1,128
    do i=1,256
      wten(i,j,k)=wten(i,j,k)+(c1(i,j,k)*dum8(i,j,k-1)+c2(i,j,k)*dum8(i,j,k))
    enddo
  enddo
enddo
```

Full parallelism available at loop level

Example: Loops in need of rearrangement

```fortran
do k=1,1024
  call radiation_solver()
  do j=1,128
    call lw_solve(a(1:256))
  enddo
enddo
```

Limited parallelism at loop level
OpenACC and OpenMP offload constructs are very similar to existing CPU-based threading.

Existing threaded version indicates that parallel “issues” have already been considered.

Existing threading approach may need to be reworked:
- GPUs need much larger level of concurrency.

**GPU-ready:**
Does a threaded version of the code already exist?
GPU-ready: Does the code provide verification?

• Code verification allows for incremental GPU-enablement
• Much easier to retain correctness than to regain correctness
• Addressing correctness bugs typically take majority of code conversion time
• Presence of well designed code verification simplifies the time spent debugging GPU-enabled code
Is your code GPU ready?  
[Student exercise: 13 minutes]

• Student exercise [5 minutes]
  – Does a GPU version of your code already exist?  
    • Yes [0 points]
      – Are the desired physics packaged GPU-enabled?  
        » Yes [1 points]
        » No [3 points]
    • No [4 points]
  – Is the code writing in such a way that it is GPU-ready?  
    • Is full parallelism is available at loop level?  
      – Yes [1 points]
      – No [7 points]
    • Does a threaded version of the code exist?  
      – Yes [1 point]
      – No [7 points]
    • Does the code have some form of verification?  
      – Yes [1 point]
      – No [7 points]

• Discuss with group any interesting realizations [7 minutes]
Is your code GPU ready? [Student exercise: 13 minutes]

- Student exercise [5 minutes]
  - Does a GPU version of your code already exist?
    - Yes [0 points]
    - Are the desired physics packaged GPU-enabled?
      - Yes [1 points]
      - No [3 points]
    - No [4 points]
  - Is the code writing in such a way that it is GPU-ready?
    - Is full parallelism is available at loop level?
      - Yes [1 points]
      - No [7 points]
    - Does a threaded version of the code exist?
      - Yes [1 point]
      - No [7 points]
    - Does the code have some form of verification?
      - Yes [1 point]
      - No [7 points]

- Discuss with group any interesting realizations [7 minutes]

NCAR UCAR | Co-design for Scientists and Project Leads
Outline

- Motivation
- How to identify GPU friendly science objectives
- Estimating effort to achieve GPU-enablement
- Estimating return on investment (ROI)
Estimating Return on Investment (ROI)

• What kind of capability GPU-enablement will deliver versus existing CPU code?
  – Serial versus parallel base case?
• Potential advantages to creation of a CPU and GPU enabled code
  – Reduced time-to-discovery for a particular science question
  – Access to broader collection of hardware
  – Ability to perform more science for a fixed resource cost
  – Ability to perform science not otherwise possible
• Advantage of GPU computing a result of better memory bandwidth and Floating-point (FP) rates
  – For Derecho: NVIDIA A100 versus AMD EPYC 7763
    • 3.8x increase in memory bandwidth
    • 1.9x increase in theoretical FP32 & FP64 rates
What is the working set size for a tightly nested loop?

- Consider typical loop in CM1:
  
  ```
  do k=1,1024
    do j=1,128
      do i=1,256
        wten(i,j,k)=wten(i,j,k)+(c1(i,j,k)*dum8(i,j,k-1)+c2(i,j,k)*dum8(i,j,k))
      enddo
    enddo
  enddo
  ```

- Loop accesses: 4 variables, 4-byte reals, of dimension 128x256x1024
- Total data access 512 MBytes which exceeds the 256 MB L3 cache on AMD EYPC
  - Memory bandwidth limited calculation → 3.8x potential speedup
- Measured overall CM1 speedup: 3.9x
What is the estimated ROI?  
[Student exercise: 13 minutes]

• Student exercise [5 minutes]
  – What is your working set size for inner loops?
  – What kind of Return on Investment (ROI) would you expect?
  – Would this kind of ROI have a meaningful impact on your science?

• Discuss with group any interesting realizations [7 minutes]
Questions: dennis@ucar.edu

Additional resources:

- [Co-Design in the Exascale Computing Project](#) (paper), Tim Germann 2021
- [ECP Co-Design Centers](#)
- [HPC Co-Design](#) (conference briefing by NNSA to DoD), Ronald Brightwell 2017
- [Workshop on Software Co-Design Actions in European Flagship HPC Codes](#), 2022
- [Resources for Co-Design](#) from POP Organization
  - [Webinar recording](#) by POP on this platform plus [slides](#)
- [A Blueprint for Success: Co-Design Approach for the Modular Supercomputing Architecture (MSA)](#), Intel 2020
- [Truly Heterogeneous HPC: Co-Design to Achieve What Science Needs from HPC](#) (slides), Smokey Mountain CSEC 2020 (focuses on neuromorphic computing)
- [On the Role of Co-Design in HPC](#) (paper), Barrett, et al 2013