Composable HPC Curricula: Embracing the UNIX Development Paradigm and Leveraging Core Practices from Linux Kernel Development in HPC Training Marterial Development

Michael Alexander
Austrian Academy of Sciences

SC23 EducHPC Lightning Talk

November 13, 2023

Issue Statement

- Landscape of HPC courses and training materials heavily emphasize
  - foundational concepts, especially those related to MPI/OpenMP and CUDA
  - site-specific intros
  - commonplace platforms such as JupyterLab i.a.
- Apparent gap to mid-tier topics
  - such as sparse linear algebra using HPC
  - given the breadth of subject matter, such courses are less commonly developed
  - mid-tier course developments may be isolated from prerequisite courses
    - leading to potential gaps or large courses with excessive overlap in content

Motivation from Course Development Observation

Having seen a past that points to possible for routes in this context {explain}

Purpose, Towards:

- a purpose being to produce HPC materials specifically focused to mainstream subjects such as sparse linear algebra numerics or part thereof, while also:
- ensuring up-to-date coverage of rapidly evolving HPC topics
  - this may enable to integrate and cover fast-changing topics that today are often operationalized in Python frameworks, some of which might not have their roots in traditional HPC.

"Make each program do one thing well." [D. McIlroy]

```
@article{unixphilo,
  author = {McIlroy, M. D. and Pinson, E. N. and Tague, B.
  journal = {The Bell System Technical Journal},
  number = {6, part 2},
  pages = {p.1902},
  title = {Unix Time-Sharing System Forward},
  volume = 57,
  year = 1978
```

Composability

"Expect the output of every program to become the input to another, as yet unknown, program.", id.

- Modularity and the:
  - First-principles trap
- Lots of utility in adaptability to different educational needs
- Composability may be facilitated by drawing on category theory with objects, morphisms, composability and functors

Composability II

Difficult domain decomposition needs flexible approaches

- junctions
- aura
- couplings
- interfaces
- boundary layer

Collaboration: Towards Kernelized Development Paradigm

A proposed solution leans on the UNIX philosophy in meaning of doing one thing and doing it well

- replicate the collaborative ethos and tools that have driven the success of open-source projects
  - exemplified by the Linux kernel
- borrow core principles of modularity, community collaboration, transparent governance, and iterative development from successful open source projects
- Going towards open source as default training material model

Collaboration II: Fostering Openess

- No course should ever be finished/frozen/vaulted
- Facilitate boarding of potential new contributors
- Forking being acceptable practize
- Relxing the notion of ownership
- Allowing for 'breaking the ABI' for agility

Collaboration Tools and Technologies

Theorem

A greatest common divisor of collaborative technolgies is sufficient for effective collaboration, think git.

Lemma

Less being more given a garden of remembrance of e-Learning middleware technologies, such as learning objects.

Pluratlity of Tools and Approaches

- Stack tandardization may not be community positive
- Keeping degrees of freedom on materials formats?
- TBD possiblity for overlay tags/decroators
  - neutral element-type DSL
  - transformable structured training materials?

BOF-Type Questions

- Complex collaborations are not self-organizing
- Needs topics, skills $\rightarrow$ architected composable topics map
- Discussion topic: is it possible to do without elaborate
  - curriculum design
  - skills matrices
  - ontologies of learning objectives
  - coverage maps
    - albeit this might be good for funding rationales
    - possible positive for creating transparent composable materials market showing gaps for potential contributors
  - Some form of non-self organizing coordination is probably needed: e.g. how are mainline pulls done?