Phylanx Kickoff Meeting
August 24, 2017

- Chris Taylor, Kevin Huck, Hartmut Kaiser, Adrian Serio, Steve Brandt, Stefan van Zwam, Rod Tohid, Alireza Kheirkhahan, Parsa Amini, Bibek Wagle
- Welcome-
  - Chris- Department of Defense
- Hartmut’s HPX Presentation-
- Kevin’s APEX Talk
  - APEX_TASKGRAPH_OUTPUT
    - Task graph conversation
  - I want to focus on policies
  - Nick’s work
    - Find tasks which are blocking
- Hartmut- Let us know if you want your DoD agency needs or desires mention
- Stefan’s Algorithms Talk
  - Matroid theory
    - Related to linear algebra
    - Connectivity
      - Connectivity gives you structure
      - Tangles
  - Algorithmic Consequences-
    - Small branch width => thin class of graphs, dynamic programming
    - Large branch width => large grid minor => redundant vertex
    - These concepts allow us to approach graph problems computationally
      - Can help with big data sets
        - Image processing
  - I study low order structures
    - Error-correcting codes
      - Asymptotically good codes
    - Computational matroid theory
      - SageMath
    - Chris- You are a target audience
  - I was brought in to analyze tiling algorithms
    - Spartan system
      - Tiling heuristic: greedy (Tile node with most neighbors first)
      - The Spartan Problem Tiling (K)
        - Input:
          - Acyclic expression digraph
          - Node groups for each call to an operator
          - Cost function on edges
        - Problem- minimize cost
          - Min-tiling
  - Pick an optimal data layout
- Ali’s PXFS Talk
  - Smart IO
    - Keep data in memory
- Move work to data
- Prefetching
  - Components
    - HPXIO
    - Interposition Library
    - Workflow Manager
  - File views
- Chris’ Talk
  - NSCI
    - Nsf.gov/cise/nscl
    - Supercomputing is in the national interest
  - Cloud Software Stack
    - Convinced hardware people to expose the attached hardware
  - “Statistical Computing” (Machine Learning)
    - Applications-
      - Theano
      - Keras
      - Tensorflow
      - Dask
    - Uses Acyclic graphs
      - “Directed Acyclic Graph Technology”
    - High level languages
      - Scalable and performant
    - I want to be able to write a script which produces a graph that is executed on the cloud
    - NumPy
      - More general purpose
  - I am not worried about data storage
  - My bosses want to use commodity nodes to lower barriers to HPC
    - Keeps the domain scientist focused on domain work
    - Abstracts the hardware from the implementation
- Why HPX
  - Little’s Law
    - $L = \lambda \cdot w$
    - Arrival wait times
    - I think that HPX is doing this
      - Fast context switching
  - Standards conformant
    - Technology Transfer
      - This is what differentiates you
    - Have acyclic graphs built in
    - Built synergistically
  - Taking an intro to HPC class
    - Talking about acyclic graphs, and critical paths, etc.
  - I realized I need a math person
    - I think the math will have implications on all other project areas
  - I have two users in mind
- High level domain scientist
- Lower level performance optimizer

  - Hartmut-
    - I see three parts
      - High level interface
        - Produces a representation of the execution graph
      - Optimization
      - Execution engine that can read graph representations
        - Interpreter
    - These parts must be developed together

  - Adrian-
    - Do we need to think about cloud computing?
      - Or can we focus on tightly coupled systems
    - Chris- we can ignore the cloud stuff

  - Stefan
    - Where does the DoD come in?
      - NCSI is led by DoD

  - Kevin-
    - Where do these other tools fail?
      - TensorFlow
    - Chris- All of the optimizations are baked into the platform

- Lunch
- Project Planning
  - What are the goals of the project
    - What will we do for the current grant
    - What will we do for the future grant
  - What are the concrete use cases
  - We will write the proposal for the next grant now
  - Project Components
    - Python component
      - Primitives (numpy?)
      - Bindings?
    - Execution Graph
      - Analysis
      - Data representation
      - Visualization (Sage support)
    - Execution Engine
      - Interpreter on HPX
      - APEX
      - Storage
    - Rely on Spartan
    - What array sizes are we talk about?

- What we are proposing to build?
  - Custom NumPy (produces an execution graph)
  - Write an execution graph analyzer
    - Do this in the next step
  - Adapt HPX to take an Execution Graph
- In step one
  - Storage
    - We will need to have some load store primitives
  - Licenses-
  - Python 2 or 3?
    - We can do 3
  - Project Steps
    - Minimal Example
      - Spartan front-end
      - Serialize (pickle)
      - Simple execution engine
      - Primitives
        - I/O
        - Trivial ops +/-*/%
        - Map, filter, fold, scan, join_update
    - APEX
      - Chunking policy
      - Python Bindings
      - Multi-objective optimization
  - HPX interpreter
    - Takes the graph and executes
    - Magic is that you are only exposing predefined types
    - Zahra’s work
  - High Level User (Python)
  - Low Level User (C++)
  - Stefan-
    - Approximation algorithm with two tilings and a 0/1 cost function
  - Logistics
    - Minimal Product
      - Algorithm: Logistic Regression
      - All one repository, private, STE||AR Organization
      - Hartmut- will do the minimal setup
  - Timeline
    - Year 1:
      - Minimal NumPy implementation
      - 1 and 2 dimensional arrays of doubles
      - Low level C++ API
      - Serialization
      - Approximation Algorithm with 2 tilings and 0/1 cost function
      - Primitives
        - Including I/O
      - Single node
      - Chunking
      - Python Bindings
      - Data Representation
      - Applications: Logistic Regression (First), ALS
      - Minimal Interpreter
- Buildbot
- Regression Testing
- Performance Regression Test
- Spartan
- CMake

  o Year 2:
    - Complex Numbers
    - 3+ Dimensions
    - Approximating n-tiling with any cost function
    - Minimal IR example
    - Accelerators
    - Tiled I/O
    - Algorithms: NN, BFS, LDA, Logistic Regression (Distributed)
    - Parcel Coalescing
    - Algorithm Policies
    - Critical Path Analysis

  o Year 3:
    - 3+ dimensions
    - FPT
    - Bi-Level tiling
    - Misc. Optimizations (Learning models + RTS Decisions)
    - Additional Operations
    - Task Cancelation
    - Applications: Chelosky, SVD, CG
    - Multi-objective opt.

- Break for the Day