LSU | Center for Computation & Technology

STE||AR GROUP

Asynchronous Execution of Python Code on Distributed Systems

@rtohid

11/07/2019

Phylanx

- Targets distributed and parallel HPC systems
  - Managed by Docker, Singularity, ... / dynamically built and deployed dockerfiles
- Distributed programming in Python
  - Multi stage distributed compilation.
  - Implemented as PhySL
- Asynchronous execution
  - Focusing on distributed concurrency
  - In-node concurrency is managed by hardware
- Optimization / Performance
  - Collected by APEX
  - Visualization by Traveler
  - Micro-scheduling: concurrent primitives (through Batched BLAS)
  - Macro-scheduling: concurrent tasks (a.k.a. subtrees)

Highlights of Phylanx (STE||AR@LSU)

- Distributed algorithms
  - Maxwell, Wei, Avah, Guoli
- Higher-dimension NumPy (Blaze Tensor)
  - Bita, Hartmut
- Linear Algebra on GPUs (cuBlaze)
  - Jules
- Optimization of linear algebra operations (Blaze tiling and scheduling)
  - Shahrzad, Gabriel
- CNN (First steps to Keras' Phylanx backend)
  - Hartmut, Bita, Shahrzad, myself
- Data collection
  - Parsa, Hartmut

Highlights of Phylanx (STE||AR@LSU)

- Scheduling algorithms (greedy and optimal)
  - Guoli, myself
- hpxMP
  - Tianyi
- Solvers for peridynamic simulations
  - Nanmiao, Bamba, Patrick
- Docker
  - Steve
- Alternative decorator paths
  - Phyll => Steve
  - OpenSCoP => Ye, myself
  - Experimental LLVM IR optimizations through Numba

Highlights of Phylanx (University of Oregon)

- Policies
  - Task in-lining
    - Monil, Bibek, Kevin
  - Message coalescing
    - Bibek, Monil, Kevin
- Task dependency analysis
  - Kevin
- Additional HW/OS monitoring
  - Kevin

Highlights of Phylanx (The University of Arizona)

- Traveler-Integrated
  - Alex, Kate, Sayef, Katy
- Expression Trees
  - Katy
- Gantt Charts
  - Kate
- Jupyter integration
  - Jesse, Katy
- Swagger & ReDoc APIs
  - Alex
- Charts for Counters
  - Sayef
- Section 508 Compliance
  - Alex, Kate

Laying Out the Suggested Solution

1. Make PhySL available in Python
   a. Drop-ins Replace Python constructs and modules (NumPy, Keras, ...) with the associated PhySL implementations.
   b. Decorators Expose and extend Phylanx by decorating Python functions.
2. Canonize the program
   a. Dropped-in doesn't need canonization!
   b. Decorated if needed, through AutoGraph, decorated functions are simplified (transformed to one or more basal functions)
   c. If canonized, transform to PhySL- based on transformation rules (built-in, or provided by the user)
3. Create the task graph
   a. Stitch PhySL nodes together

Laying Out the Suggested Solution

4. Tile the data
    a. By looking at the task graph
5. Optimize the task graph
    a. Make changes to the task graph to reflect the new data layout
    b. Partition the graph to assign tasks to nodes
    c. Configure lower level optimization
        i. Inlining
        ii. batch BLAS
        iii. Message coalescing
6. Debugging
    a. Trace data
    b. Visualization

Summary / Demo / Conclusion

- https://hub.docker.com/r/stevenrbrandt/phylanx.devenv/tags

Thank you!