High-Performance Decoupling of Tightly Coupled Data Flows

Halo particles, Voronoi tessellation, and 2D density estimation

Tom Peterka
Argonne National Laboratory

Jay Lofstead
Sandia National Laboratory

Franck Cappello
Argonne National Laboratory

SDMAV Kickoff PI Meeting
Executive Summary

To achieve high performance, programmers tightly couple the data analysis with data generation -- making the analysis interdependent and closely coordinated with the computation, but limiting modularity and reuse. To address this issue, this project will explore a hybrid approach that combines both types of coupling---tight and loose---in effect decoupling tightly coupled applications.

Key Ideas

• Inject a separate dataflow between producer and consumer that enables:
  • Aggregation, deep data permutations
  • Automatic buffering
  • Data redistribution and pipelining
  • Resilience to faults
• Implement generic coupling between producers to consumers in a lightweight library that other tools can use
• Target extreme-scale architectures, workflows, and applications
Analysis Workflow

Analysis = Any data transformation, or a network or transformations. Can be visual, analytical, statistical, or data management. Anything done to original data beyond its original generation.

Generic analysis data flow graph, primarily for simulation data, single or ensemble sources and multiple users. Results are written to persistent storage at the cyan line that partitions the graph into operations done at run time (in situ) and post hoc.
Custom Coupling of Software

- Today, we write analysis tasks as libraries with a different driver for each combination of analysis tasks.
- Writing custom one-off main programs for each combination of producer and consumer is not a scalable approach.
- Neither is tuning the producer (number of nodes, output size, etc.) to the consumer and vice versa.
- Producers and consumers ought to be written independently, and generic coupling software should manage their connection.
# A More Generic Approach

## Applications
Mission-driven simulations, experiments, observations, ensembles, parameter sweeps

## User Libraries and Tools
Custom libraries, standard visualization/analysis packages, scripting and workflow

## Common Libraries
Statistical, math, vis, ML, graph analytics

## Data Movement
- **Intracode**
  (distr. data parallelism)
- **Intercode**
  (coupling dataflows)

Intra- and intercode data movement building blocks, data as a service data layer

## Optimized System Libraries
Run time, programming model, I/O

## System Services
Storage systems, resource managers, schedulers
Decaf: Decoupling Tightly Coupled Data Flows

Decoupling by converting a single link into a dataflow enables new features such as fault tolerance and improved performance.

We are building a generic coupling library out of 4 primitives that can be used for many purposes.
Major Decaf modes include aggregation, pipelining, and automatic buffering while potentially permuting data in an N:M and direct coupling of parallel codes.
Three abstract communicators—producer, dataflow, and consumer—are used to couple producer to consumer.

The dataflow can be a simple noop or a complete parallel program performing complex data transformations.
Resilience to Faults

One of our resilience efforts attempts to detect silent data corruption by validating analysis tasks with an auxiliary method, usually less expensive and less accurate, but hopefully good enough to detect soft errors.

Another research topic is modeling the dataflow and optimally adding replication and rollback mechanisms to recover from hard (fail stop) errors and soft errors detected above.
The above table summarizes the state of the art by describing various tools along different dimensions with respect to the capability needed for Decaf. A dark check mark indicates that the tool has all the capability that we need in Decaf. A light check mark indicates less than complete coverage compared with our projected need.
Wrapping Up

Decaf is a new project to couple analysis tasks together with simulations and with each other.

**Knowns**

- We already know how to write individual data analysis tasks with scalable intracode data movement
- We are designing an intercode data movement layer featuring:
  - A separate scalable dataflow between producer and consumer
  - Automatic buffering
  - Data redistribution and pipelining
  - Resilience to faults

**Challenges**

- Synergy with existing coupling tools and transport layers
- High-level interface: workflow representation and execution
- Data model representation
Acknowledgments:

Facilities
Argonne Leadership Computing Facility (ALCF)
National Energy Research Scientific Computing Center (NERSC)

Funding
US DOE SDMAV2 Exascale Research

People
Jay Lofstead, Patrick Widener, Franck Cappello, Florin Isaila, Lokman Rahmani,
Hadrien Croubois, Guillaume Aupy

“The purpose of computing is insight, not numbers.”
–Richard Hamming, 1962