Data Reorganization and Future Embedded HPC Middleware

Ken Cain  
The MITRE Corporation

James Lebak  
MIT Lincoln Laboratory

Anthony Skjellum  
MPI Software Technology, Inc.

Abstract

In recent years, the embedded HPC community has seen the establishment of new standard software interfaces such as the Real-Time Message Passing Interface (MPI/RT) and the Vector, Signal, and Image Processing Library (VSIPL) API. Embedded HPC hardware vendors have also teamed with third party software vendors to provide support for the standard Message Passing Interface (MPI). Although each of these APIs will provide the building blocks for developing portable and high performance applications, two very important areas need to be addressed as HPEC middleware evolves:

1. Reducing the difficulty of developing data-parallel signal processing software

Currently, writing data-parallel signal processing software involves explicit management of complex data relationships, including manual calculation of data partitioning and invocation of low-level communication primitives (e.g., DMA transfers, point-to-point and collective message passing calls). Vendor-provided libraries provide some relief, but still require substantial programming effort when application requirements do not exactly match available library functionality. The data reorganization API approach provides a higher level of abstraction, treating many individual peer-to-peer parallel communications as a single logical operation.

2. Integrating communication and processing oriented functions to provide a more automated software optimization process.

To date, middleware standards and proprietary libraries have strictly separated communication and processing operations. However, with the integration of advanced architectures like the PowerPC G4 in embedded computers, applications will be faced with tradeoffs—organizing data "optimally" for processing, but at the expense of a more costly communication phase (and vice versa). Currently, advanced communication-only APIs like MPI/RT rely on a deferred early-binding approach (planning real-time communications in advance, at initialization time). Future middleware will need to apply the early-binding concept to an entire signal processing chain, and not just to its individual components. Currently, the data reorganization API approach considers only the communication components, but does address optimization-motivated integration with other communication interfaces like MPI and MPI/RT (so-called "co-layering").

This talk will highlight the work of the Data Reorganization Forum, a volunteer group consisting of embedded HPC vendors, defense contractors, users, and researchers. The forum’s work to produce a high level, language-independent data reorganization API will be highlighted. Features of this API that support high performance will be emphasized, including integration of user-space and library-space memory buffers and outer loop (early) resource reservation. Additionally, the group’s preliminary work to define an MPI co-layer will be presented. Finally, near-term API interoperability approaches and long term advanced API integration approaches will be presented.
Data Reorganization and Future Embedded HPC Middleware

Ken Cain, The MITRE Corporation (Presenter)
Anthony Skjellum, MPI Software Technology Inc.
James Lebak, MIT Lincoln Laboratory†
20 September 2000

† This author sponsored by the U.S.Navy under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.
The Data Reorganization Forum

http://www.data-re.org
Join the mailing list discussion!

Goal: Final specification by June 2001

- Broad community participation includes:
  - FFRDCs and Government/Defense Laboratories
  - Defense integrators
  - Commercial embedded multicomputer vendors
  - Commercial HPC tool vendors
- Examining API’s, algorithms, and application requirements
What Problems Does Data Reorg Try To Solve?
Data Partitioning and Redistribution Issues for Signal/Image Processing (SIP) Applications

- Block partitioning is most common
  - Whole problems stored in 1 memory for performance

- Data redistribution communication is “severe”
  - Prototypical example is matrix transpose in 2DFFT/SAR
Interface Scalability

State of the Art (current standard APIs)
- Programmer manually computes data partitioning
- Programmer manually redistributes data (MPI or MPI/RT)
- Compute using VSIP

Future Practice (with Data Reorg API)
- Programmer uses high-level partitioning services
- Middleware handles data partitioning details
- Data redistribution with a single high-level call
- Compute using VSIP

Long-term future: higher-level / integrated / OO ???

Easier to scale programming effort

Hard to scale programming effort to large systems

Long-term future: higher-level / integrated / OO ???
Data Reorg Interface Example

Example: Data movement with block to cyclic redistribution

This example shows a one-dimensional data set stored in block fashion mapped on three nodes being re-distributed in cyclic fashion over two nodes.

- Application programmer uses DRI to move data
- DRI hides complex data movement from programmer
Model-Year Portability

Portable software leverages inevitable advances in COTS HPC technology

Defense system lifetimes: long
COTS HPC system lifetimes: short

“Point” solutions specific to a single vendor are long-term cost ineffective

Portable software with high performance is a powerful tool and is the ultimate goal
Challenges to Achieving Consensus In A Committee Context
Three Areas of Concern

Research
- Allow integration of research approaches in API implementations
- Enable optimized implementations for a broad class of HPC architectures

Operational
- Will this API make it easier to write SIP applications?
- Does API support most common data reorgs for SIP?

Overlap with other APIs
- Common user / library buffers
- VSIP, MPI, MPI/RT
- Which API allocates data?

Scoped / Prioritized to satisfy most SIP application needs
Data Reorg
Committee Status
Data Reorg
Objects and Implementation Approaches

CORE
- Uniquely part of Data Reorganization API
- Must be provided in all Data Reorg implementations
- Objects:
  - DRI_Global_Data
  - DRI_Partition
  - DRI_Distribution
  - DRI_Layout
  - DRI_View
  - DRI_Overlap
Data Reorg
Objects and Implementation Approaches

Standalone

- Functionality overlaps with other middleware
- Full implementation (without Middleware Adapter) gives a “pure” data reorg programming environment

- Objects:
  - Datatypes: DRI_Dataspec
  - Process Sets: DRI_Group
  - User and Library Memory: DRI_Bufferset
  - User and Library Memory: DRI_Buffer_Id
  - Data Transmission Constructs: DRI_Channel
Middleware Adapter

- Defines a hybrid interface that leverages supporting middleware
  - MPI
  - MPI/RT
  - Mercury PAS
  - Sky SCL

- Objects:
  - Selected from “Standalone”, depending on supporting middleware
Data Re-org Forum Plan

- Two more official meetings
- Several informal “working” meetings
  - Resolve issues with buffers and buffersets
  - Resolve issues with memory layouts and distributions
- Near-Term activities:
  - Establish CORE and Standalone Interfaces
  - Define MPI Middleware Adapter for Data Reorg
  - Final document detailing ideas and lessons learned

In the long term, the forum feels that a larger effort in this area would have substantial benefits for the high-performance embedded computing community.