RU-Spaces: Enabling Data-Intensive Applications Workflows at Extreme Scale

Center for Autonomic Computing

Ciprian Docan, Fan Zhang, Tong Jin, Manish Parashar

Abstract

Large scale parallel and distributed applications (*e.g.*, data analytics, science and engineering simulations, financial/business analytics, etc.) typically involve complex workflows that are data intensive. In these applications, independent heterogeneous application components need to collaborate and exchange information following dynamic and complex interaction patterns. These application components often run on heterogeneous resources and progress independently at different rates. They thus require a flexible and scalable underlying interaction and coordination framework, which can support the interaction patterns while having minimal impact on the execution of the applications themselves. Implementing these interactions using typical workflow systems that impose stricter coordination and synchronization constraints can be inefficient.

Motivation and Objectives

Motivations

Large-scale applications running on large-scale HPC machines generate large data volumes, which are complex and expensive to manage

- Application requirements
  - Scalable data-intensive computations and collaboration
  - Dynamic and heterogeneous interactions and coordination
  - Efficient exchange of large data sizes
- System complexities
  - Multi/many-cores architectures
  - Hybrid systems, e.g., clusters of CPUs & GPUs
  - Deep/multi-level memory hierarchies

Challenges

- Efficient data transport at scale on HPC machines and across WAN
- Online data mapping and indexing for quick data lookup and retrieval
- Exploiting data locality between parallel applications running on hybrid resources
- Minimizing data movement between cooperating applications

Objectives

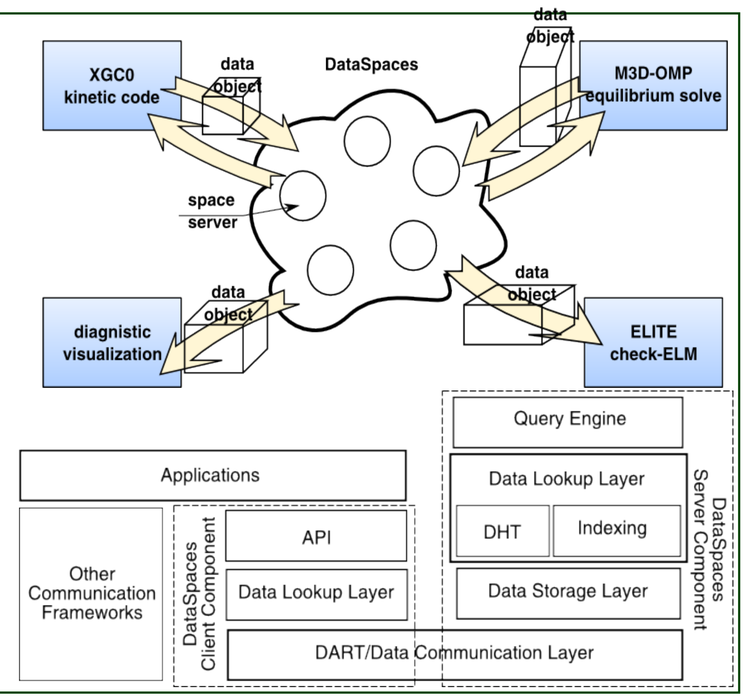
- Design and deploy a distributed, robust and scalable framework to support asynchronous coordination and data exchanges in a heterogeneous environment
- Provide simple and flexible high-level programming abstractions for coordination, data access, and data exchange.

RU-Spaces Project

- **DART: Asynchronous data transfer and communication substrate**
- **DataSpaces: Scalable framework for applications interaction and coordination**
  - Semantically specialized shared-spaces abstraction spanning hybrid resources – multi-cores (CPU & GPU)
  - Online metadata indexing for fast access, data retrieval and query processing
  - Scalable, robust in-memory data storage for data exchanges
  - In-situ workflow scheduling and application co-location for data locality
- **ActiveSpaces: Dynamic code deployment and execution at the data source**
- **High-level programming, *e.g.*, PGAS, Workflows, Database**

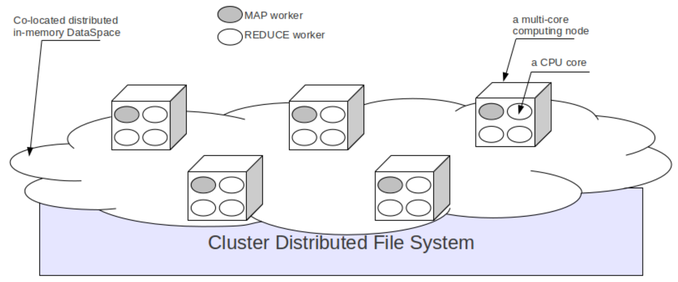
DataSpaces

- **DataSpaces overview**
  - Provide the abstraction of a semantically specialized virtual shared-space that can be associatively and asynchronous accessed by multiple applications
  - Builds on a dynamic/hybrid set of nodes, e.g., CPUs, GPUs, Cloud instances, etc.
- **Architecture**
  - *Storage layer*: implements a distributed in-memory temporary storage
  - *Indexing & DHT layer*: online indexing and distribution of the metadata for fast retrieval
  - *Query engine*: supports flexible, semantically specialized metadata queries, including queries involving ranges and wildcards
- **Programming model:**
  - Simple and flexible API for inserting or retrieving data



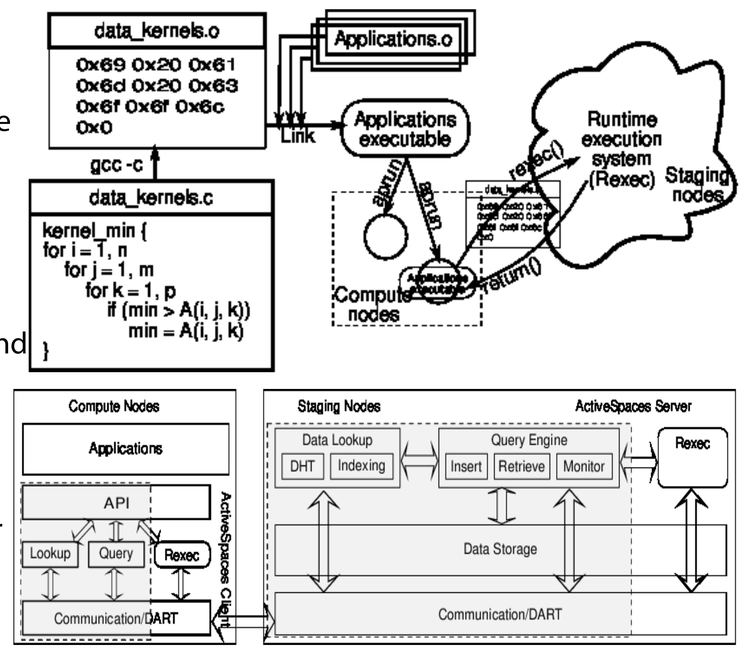
In-situ Workflows

- **In-situ workflows overview**
  - Build on a Hybrid DataSpaces that runs inline with the applications on multi/many-core systems
    - Exploits computing heterogeneity to efficiently map applications to resources
  - Runtime layer schedules and co-locates workflows components based on application interaction patterns to maximize data locality



ActiveSpaces

- **ActiveSpaces overview:**
  - A service built on DataSpaces to enable custom transformation and filtering of data by providing runtime execution environment for dynamic code deployment and execution
- **Architecture:**
  - Runtime execution layer (Rexec) defines the application programming support to develop customized data kernels
  - Provides a runtime execution system to deploy and execute the kernels on the data in DataSpaces
- **Programming support:**
  - All "C" language constructs can be used to define custom data kernels
  - Binary code is loaded into DataSpaces in a similar manner as data
  - Supports map/reduce style operations for distributed data transformation operations



Status and results

- Complements existing workflow engines with efficient data exchange mechanisms
- Used by real scientific application codes [1] as part of major US projects such as the **Fusion Simulation Project** and **Combustion Exascale Co-Design**
- Efficiency and scalability demonstrated at very large scales [2]
- Application acceleration and online analytics support by customized data kernel deployment [3]

Publications

1. Enabling Multi-Physics Coupled Simulations within the PGAS Programming Framework, CCGrid' 11
2. Moving Code to the Data – Dynamic Code Deployment using ActiveSpaces, IPDPS' 11
3. DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows, HPDC' 10

URL: http://nsfcac.rutgers.edu/spaces/

China – US Software Workshop, September 2011

The Center for Autonomic Computing is supported by the National Science Foundation under Grant No. 0758596.

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY