GOES-R Ground System and Algorithm Implementation Design

Alexander Werbos*, John Baldwin*, Adam Copeland*, Jared Donbocht†, Joshua Downer, Robert Kaiser†, Elizabeth Lundgren*, Andy Tarpley† and T. Scott Zaccheo*

*AER, Inc, Lexington, MA USA

†Harris Corporation GCSD, Melbourne, FL USA

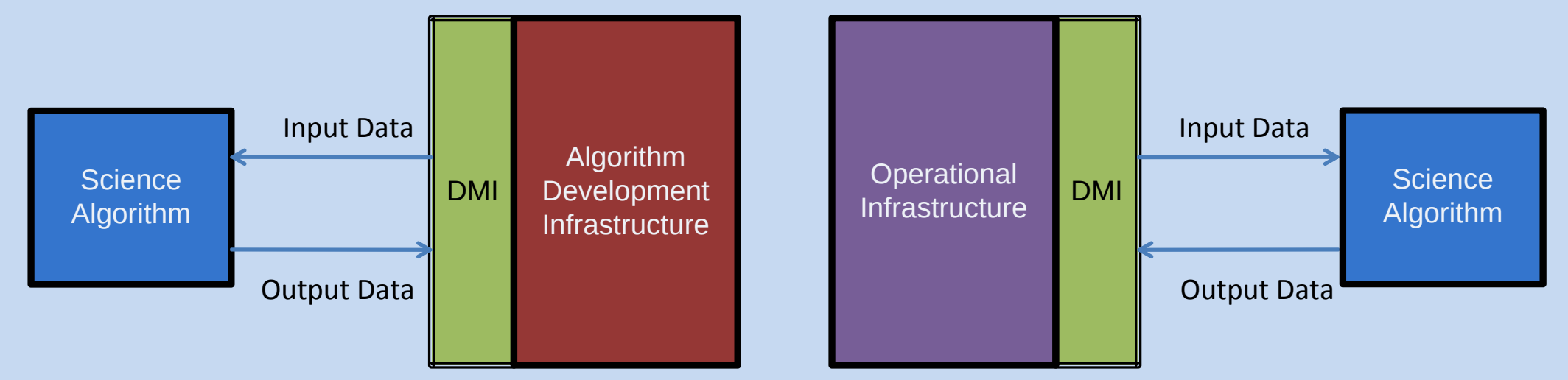
Introduction

The next generation of NOAA's Geostationary Operational Environmental Satellite system, Series R (GOES-R) provides continuity of the GOES mission and improvement of its remotely-sensed environmental data. The GOES-R system consists of the Space and Ground Segments. The Space Segment consists of spacecraft bus, its remote-sensing instruments, and communications payloads; while the Ground Segment consists of all Earth-based functions, provides satellite operations and instrument product generation and distribution. The GOES-R Ground Segment operates from three sites: the NOAA Satellite Operations Facility (NSOF) in Suitland, MD; the Wallops Command and Data Acquisition Station (WCDAS), located in Wallops, VA; and a geographically diverse remote backup facility (RBU) located at Fairmont, WV. The architecture has been developed to allow integrated operation within a geographically distributed information systems framework. GOES-R will provide advanced products, based on government-supplied algorithms, that describe the state of the atmosphere, land, and oceans over the Western Hemisphere as well as products for monitoring the local space environment and the solar state. The Harris GOES-R Core Ground Segment (GS) Team will provide the software and engineering infrastructures to produce and distribute these next-generation data products both directly to users and to the archival systems. Within the Ground Segment, the Product Generation Element (PG) is responsible for the software implementations of scientific algorithms. In this presentation we provide an overview of how Product Generation software works with the other elements of the Ground Segment to produce Level 2+ end-products. We discuss the specific software structures used to implement Level 2+ algorithms, and how those structures interface with other components in a way that meets the needs of a distributed, high-performance computational environment.

Design Objectives

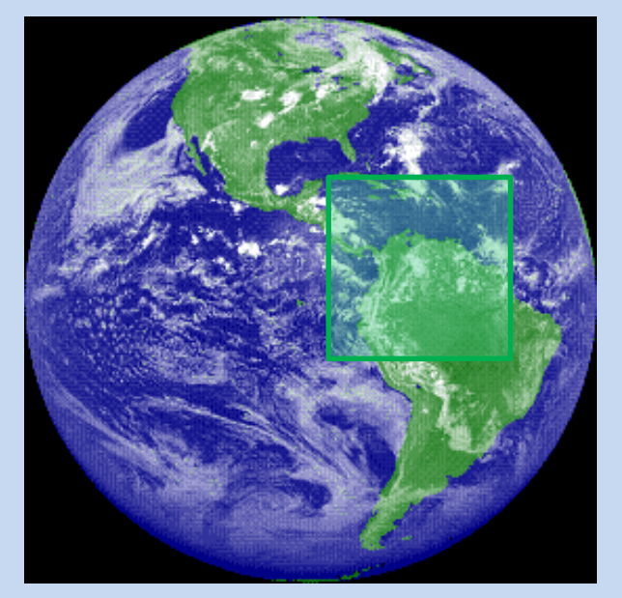
Pluggable Algorithms

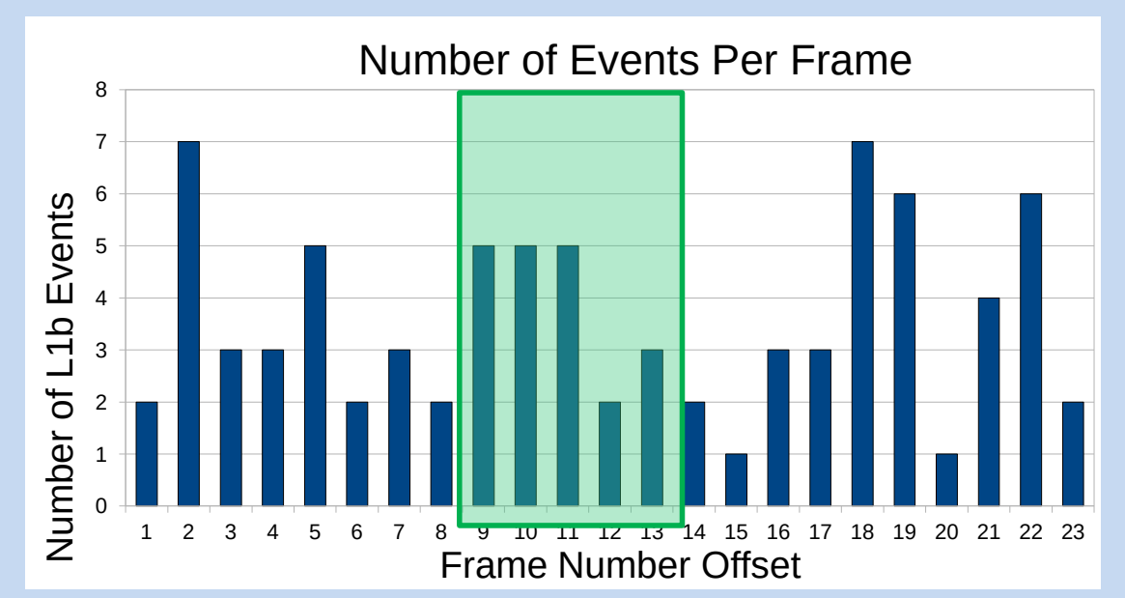
- GOES-R scientific algorithm software and operational infrastructure are developed in separate environments with different needs
- Algorithms need to be transitioned between these two environments without altering their internal functionality
- Infrastructure must provide algorithms with a common **Data Model Interface** (**DMI**) that abstracts data I/O from the environment-specific infrastructure



Variable Algorithm Execution Area

- Scientific algorithms run on a particular area. For grid algorithms, this is a pixel rectangle. For time-series algorithms, it can be a range of times to be processed.
- The size of algorithm execution area greatly impacts its performance. Larger areas may have less overhead, while smaller areas may be parallelized to more processing units.
- The processing hardware configuration is the key factor in determining the best processing block size. This size is generally independent from the scientific functionality of the algorithm.
- To give maximum flexibility in operational hardware configurations, infrastructure must allow algorithms to have variably sized execution areas, known as **Contexts**

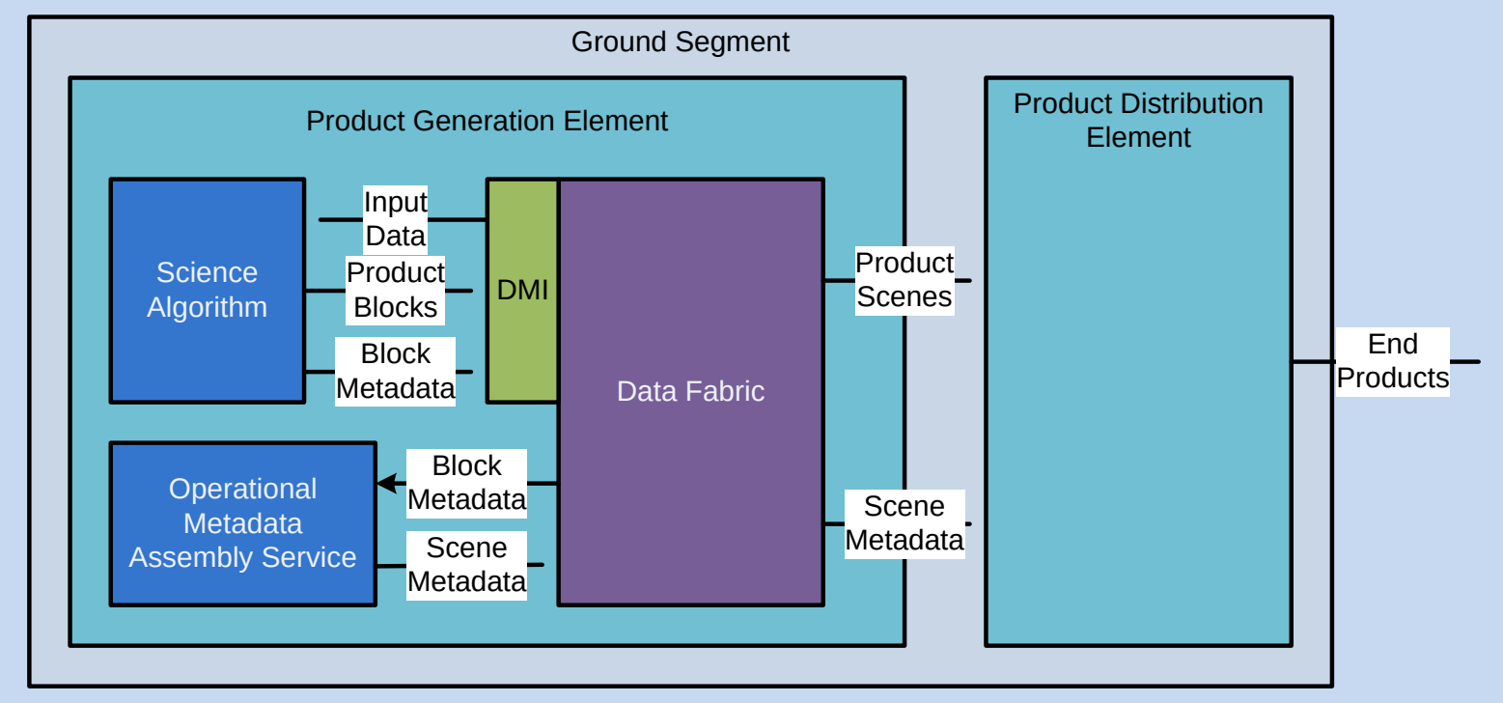

Grid Algorithm Execution Context



Time-series Algorithm Execution Context

Shared Product Generation Responsibilities

- Algorithm software is narrowly focused on generating output data, not concerned with final output product format or encoding. This decouples the scientific logic involved in generating products from end-user requirements, allowing both to change more easily.
- The **Data Fabric**, a high-performance data access and storage layer, fills the role of operational infrastructure. It automatically assembles image pieces like a jigsaw puzzle as they are processed in blocks.
- The **Operational Metadata Assembly Service** (**OMAS**) builds block-level metadata into scene metadata
- **Product Distribution** assembles raw image data, scene metadata, and supporting semi-static data into final product files



Design Decisions

Algorithms as Passive Components

- Algorithms are conceived of as objects that must be run by a set of other components that collectively fulfill the role of **Algorithm Executor**
- The Algorithm Executor instantiates algorithm objects, schedules and runs them
- The Algorithm Executor will provide the necessary concrete DMI objects to algorithms so they can run in its environment

Narrow Infrastructure Interface

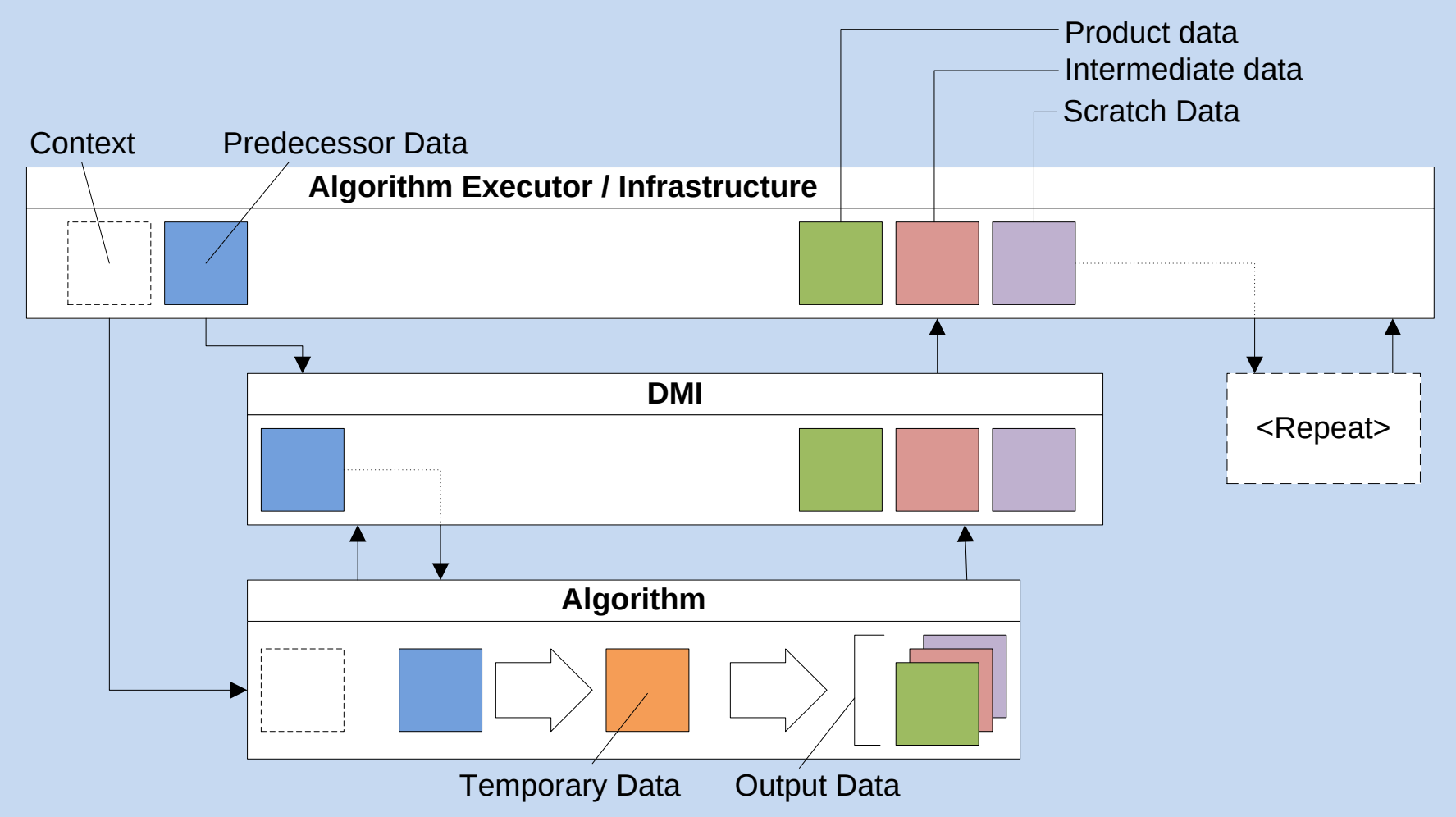
- A single common DMI, termed **DMI Generic**, must exist between algorithms and Infrastructure that is capable of handling all the diverse types of algorithm data
- This DMI Generic interface is implemented by all environments that will run algorithms
- DMI Generic works with two types of data: Imagery and Atomic Blobs
  - Imagery are gridded data that can be retrieved in smaller sections, allowing large images to be processed in smaller chunks
  - Atomic Blobs are data that must be retrieved in one piece, such as algorithm configuration parameters or lookup tables
- Algorithms can retrieve Imagery and Atomic Blobs in several ways, such as all images falling in a certain time range, the most recent set of configuration parameters, etc.
- This simple interface means that new algorithms with new data do not require changes to the DMI
- Data are translated from generic formats on read and write using **DMI Specifics**
- DMI Specifics provide/accept data in their most convenient form (strongly-typed blocks of floating point numbers, parameter data structures, etc)

Algorithm Execution Parameterization

- We developed the general notion of an Algorithm Execution **Context**, a data structure describing the spatial and/or temporal region on which an algorithm is to be run
- The Algorithm Executor generates contexts, and runs the algorithms on them
- Algorithms read in the necessary data to generate output for the entire context
- Statistical metadata, such as mean product values, are calculated on a per-block basis by the algorithms

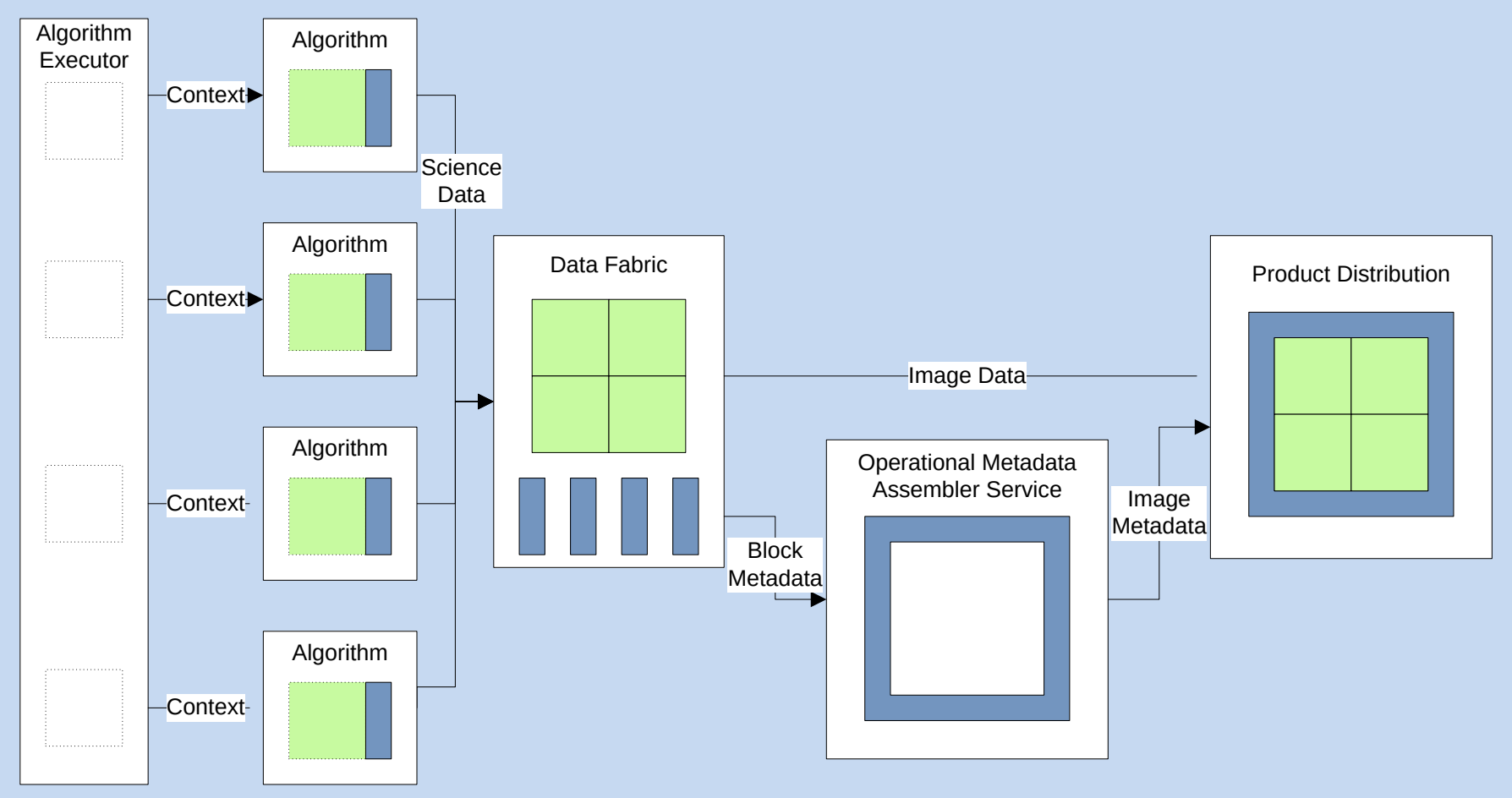
Algorithm Execution Model

- Algorithms are instantiated with an instance of the DMI Generic, which will be used to read and write all data
- All data can be preloaded into the DMI before algorithm execution begins, instead of waiting for sequential access-and-retrieve
- Algorithms use DMI Specifics to retrieve data in immediately-usable formats
- Computations are performed
- All data that must be persisted, including temporary state to be used subsequent runs, must be written to the DMI
- When execution terminates, algorithm and DMI instances can be destroyed



Summary

- Algorithms are passive components
- An environment-specific Algorithm Executor creates contexts, and schedules and runs algorithm on those contexts.
- The Algorithm Executor provides a concrete implementation of the DMI Generic interface to algorithms to read and write all of their data
- Data written to the DMI Generic is sent to the Data Fabric, where images are automatically put together
- Block-Level metadata are assembled by the OMAS into product-level summaries
- Algorithm data and metadata are combined and formatted by Product Distribution



Future Work

Algorithm Operationalization

- Algorithms are currently being developed and tested in the Algorithm Development Infrastructure
- Future work will involve integrating the algorithms into the Operational Infrastructure, including the Data Fabric and OMAS
- Operationalized algorithms will be optimized for high-performance operation
- Algorithms will be wrapped in services designed to run in a distributed computational environment