

Software Connectors for Highly Distributed and Voluminous Data Intensive Systems

Chris A. Mattmann^{1,2}

¹*Computer Science Department
University of Southern California
Los Angeles, CA 90089
mattmann@usc.edu*

²*Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109
mattmann@jpl.nasa.gov*

Abstract

We describe a research agenda for selecting software connectors which quantifiably satisfy different scenarios for large volume data distribution. We outline the necessity for a framework which allows a user to select amongst the different distribution connectors available. The framework is based on a classification of distribution connectors along eight key dimensions of data distribution.

1. Research Problem and Importance

Consider the following realistic scenario in which a recent NASA mission, the Mars Reconnaissance Orbiter (MRO), is set to produce *hundreds* of terabytes of data. This mission *alone* will increase the overall size of NASA's planetary science archive, the Planetary Data System (PDS) [1], from the current 20 terabytes (TB) to over 200 TB. This data needs to be delivered *periodically* using *hundreds* of intervals of size anywhere between *tens of megabytes* to *tens of gigabytes*. The data needs to be received by *thousands* of scientists, and *thousands* of educators geographically distributed across *the U.S.* and *foreign countries* with *hundreds* of different network configurations and topologies and *hundreds* of different *access policies* at each user's institution.

The above scenario poses a significant challenge for the problem of selecting appropriate interconnection/distribution mechanisms (i.e., *software connectors*) for data delivery. Since there are many popular software connector *classes* for data distribution (e.g., P2P, grid technologies), the problem of selecting the most appropriate connector class (or combination thereof) that adheres to a particular distribution scenario, and that is compatible with existing data distribution architectures and systems, becomes very difficult.

To address the enormous projected volume increase from MRO, we recently completed an exploratory study [2] at the Jet Propulsion Laboratory (JPL) and the University of Southern California (USC). We

evaluated a set of existing connectors for their key properties in enabling large volume data distribution. These connectors included the basic file transfer protocol (FTP) [3], GridFTP [4], commercial [5] and open source [6] UDP data transfer technologies, and several "hard media" (e.g., storage brick, DVD) technologies. The major problem that we identified during our task was the lack of knowledge with respect to classifying and comparing how each of the available off-the-shelf software connectors are able to deal with different use-case scenarios for data distribution.

Our study led us to identify the need for a framework for selecting software connectors for large-volume data distribution. As part of this selection framework, we have targeted a set of data distribution dimensions of interest to end users. The proposed framework, given a set of constraints on the multi-dimensional data distribution problem space, will aid a user in understanding how the set of constraints affects the combination of software connectors to use. Part of the framework includes a dynamic software technology which provides the capability to deploy, adapt, combine and replace different software connectors in support of a given data distribution scenario.

2. Related Work

Several areas of existing literature, incl. classifying and combining software connectors, detecting architectural mismatches when combining software connectors, and middleware technologies, influence our work.

One of the seminal sources for the classification of software connectors to date has been the taxonomy of software connectors by Mehta et al. [7]. Although many other works provide a theoretical foundation for software connectors (e.g., [8, 9]), our preliminary work to date on classifying distribution connectors has drawn heavily from this taxonomy. Mehta et al.'s taxonomy has been employed and studied by Kandé and Strohmeier [10], who propose a UML connector

model based on the taxonomy, and by Bálek and Plášil [11], who utilize the taxonomy to address the *deployment anomaly* issue.

Spitznagel and Garlan [12] outline a framework for *composing* and *combining* software connectors focusing on a class of connectors called *wrappers*. They model a connector as a six-tuple of concrete areas comprising: application code, communication libraries, low level infrastructure services, data tables, policy, and formal behavior.

Arbab's work on the Reo [13] channel coordination system and the work by Lau et al. [14] are supplementary theoretical models for combining connectors. Both connector models utilize the notion of *exogenous* connectors. Exogenous connectors control the execution of a program, rather than just the communication. Each model provides the ability to construct more complex connectors out of a limited amount of primitive connector types. We hypothesize that the primitive connector types proposed in the two models can help us to combine distribution connectors.

Garlan et al. [15] defined the problem of *architectural mismatch*. Architectural mismatch is observed to be a type of mismatch in which a "reusable part makes assumptions about the structure of the application in which it is to appear". The authors classify the four major forms of architectural mismatch encountered: *component* assumptions, *connector* assumptions, *topology* assumptions, and *construction process* assumptions.

Complementary to Garlan et al.'s work on architectural mismatch was the dissertation work by Gacek [16]. Gacek delves into the problem of architectural mismatch when composing systems (and their subsystems). To combine distribution connectors, we will require a method for identifying mismatches; we plan on leveraging the foundational work by Gacek to develop a similar approach.

Middleware and software bus [17] technologies have often been thought of as implementation-level artifacts of architectural connection [18] evidenced by the fact that many data distribution connectors (e.g., OODT [19], GridFTP) are implemented by leveraging capabilities from existing middleware technologies. Although there have been several recent attempts to classify and compare middleware technologies (see Emmerich [20]/Zarras [21]), that work to date has been very broad. Our work will entail a detailed comparison and classification of how different connectors affect the data distribution application family along its unique dimensions such as *total volume*, *number of sites*, and *number of users*.

There are many other areas of related work that inform our research including *information*

dissemination systems, and *software connector testing*; due to space limitations we omit their full treatment here and point the reader to the author's qualifying exam report for a more detailed survey [22].

3. Problem Definition

We will address the problem of identifying and selecting suitable software connectors for data distribution that satisfy user specified data distribution constraints, or scenarios. So far, we have identified eight key dimensions of data distribution based on a review of existing literature (see [22] for a detailed survey) and on our own experience in the context of planetary science and cancer research at JPL [19]. A set of constraints on any combination of these eight dimensions identifies a data *distribution scenario*. Although we acknowledge the existence of other dimensions of data distribution (e.g., see [23]), we believe these eight dimensions to be a unique and representative set that is very capable of modeling the wealth of distribution scenarios that exist. The eight dimensions are:

1. *Total Volume* - the total amount of data that needs to be transferred from providers of data to consumers of data.
2. *Delivery Intervals* - the number, size and frequency (timing) of intervals wherein which the volume of data should be delivered.
3. *Performance Requirements* - any constraints and requirements on the consistency, efficiency, scalability and dependability of the distribution scenario.
4. *Number of Users* - the amount of unique users to whom data volume needs to be delivered.
5. *Number of User Types* - the amount of unique user types (e.g., scientists, students) to whom the data volume needs to be delivered.
6. *Data Types* - The number of different data types (e.g., data, metadata) that are part of the total volume to be delivered.
7. *Geographic Distribution* - The geographic distribution of the data providers and consumers.
8. *Access Policies* - The number and types of access policies in place at each producer and consumer of data.

4. Hypotheses

H1. Given a set of ≥ 2 OTS distribution connectors, and distribution scenarios and performance requirements that must be satisfied, an efficient algorithm can calculate (1) what connectors to select to meet the requirements and (2) if the selected connectors can be combined.

H2. By dynamically combining at least 2 existing off-the-shelf (OTS) distribution connector classes, a

representative set of distribution scenarios precluded by the scalability limitations of existing client-server distribution connectors (e.g., REST/HTTP) can be satisfied.

H3. At least 2 OTS distribution connectors can be temporally combined on the fly, ensuring at least the lower bound dependability and efficiency measurement of the constituent connectors.

5. Approach

Our framework includes a *classification*, *categorization*, *integration* and *testing* step (shown from left to right in Figure 1). The classification step entails classifying a set of distribution connector classes (shown in the left side of Figure 1) along our eight dimensions of data distribution. So far, we have identified four major classes of distribution connectors to classify: *event*-based, *grid*-based, *P2P*-based and *client/server*-based. Several implementations of these connector classes are widely deployed and pervasive in large-scale data distribution tasks in research, industry and government projects. The classification component is independent of the rest of the framework and performed offline, carefully separating the generation of what is essentially training data to the framework, from that of the decision making portion of it. The classification activity step will result in *distribution connector profiles (DCPs)*, which will be used as input to the next step in the approach.

After the classification activity is performed, the framework moves on to the categorization step. The inputs to the categorization step include DCPs, user-specified distribution scenarios, and a set of user preferences on performance requirements (e.g., “the connector should be *at least* scalable to N producers and consumers”). We separated the components that deal with each type of input to allow the evaluation of each type of input to occur independently of the other, e.g., if the user preferences are not satisfied, then there is no need to send data to the *Selector* component. The categorization activity will leverage the profiles to map the distribution scenario dimensions to candidate connectors that can satisfy the scenario constraints. As part of this mapping, the *Categorizer* will need to understand how to group and manage DCPs for use in search and retrieval by the *Selector* component. The *Selector* component will then use the information from the *Categorizer* and the user preferences to further discriminate against candidate connectors. The *Selector* component will need to detect potential architectural mismatches for the candidate connectors, which will be a challenging problem. Initially, we plan on investigating a simple pair-wise mismatch

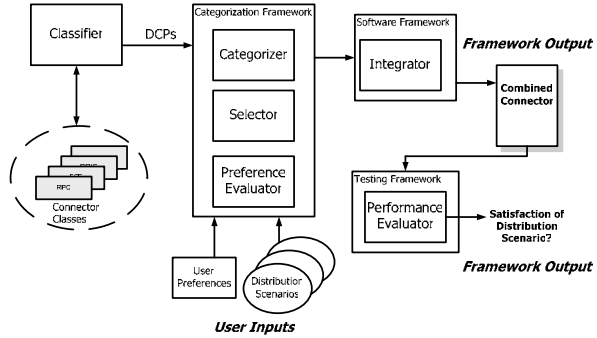


Figure 1. The proposed distribution connector framework

algorithm (based on the foundational work by Gacek [16]) that compares two data distribution connectors along the eight data distribution dimensions we have identified. For each dimension, the algorithm would detect potential mismatch areas and decide whether the (set of) mismatches identified were severe enough to prevent connector combination. For example, there may be a mismatch in the *Number of Users* dimension of two distribution connectors, A and B. If connector A supports fewer users than connector B, then A may become the bottleneck in the distribution.

The connector list output from the categorization framework will be sent to a dynamic software technology which will provide the capability to deploy, adapt, combine and replace different software connectors. The crux of the framework will be the *Integrator* component. It will need to support four major types of connector integration:

- *Wrapper* – take ≥ 2 distribution connectors and provide a wrapper in between them to allow them to communicate (does not modify connectors)
- *Switch/Alternator* - take ≥ 2 distribution connectors and determine how to divide the data up between them to distribute data over a time interval (does not modify the connectors)
- *Extractor* - integrate $x\%$ of distribution connector A and $y\%$ of distribution connector B to arrive at a new distribution connector C (may modify the connectors)
- *Combiner* - completely integrate and combine the functionality of two distribution connectors (may modify the connectors).

Although the goal of the proposed *Integrator* component is to support all four connector integration methods, initially we will focus on *Wrapping* and *Switching*, leaving the *Extractor* and *Combiner* methods to be investigated last. By necessity *Extractors* and *Combiners* are ad-hoc, and existing literature [18] has shown these two integration methods are very difficult to realize. We hypothesize

that at least the *Extractor* method can be synthesized using combinations of *Wrappers* and *Switchers*. The output of the software framework will be the actual distribution connector code (or modifications to it). The distribution connector code will be sent to the testing framework (i.e., the *Performance Evaluator* component) for ultimately validating the satisfaction of the distribution scenario, and the user preferences.

6. Validation Strategy

The approach will be validated empirically using large (terabyte-scale) amounts of data derived from three data-intensive information systems incl. PDS, the Orbiting Carbon Observatory (OCO) earth science mission at JPL, and the National Cancer Institute's Early Detection Research Network (EDRN), on which JPL leads informatics efforts. We will perform controlled experiments against scientist-specified distribution scenarios validating (or disproving) that the combination of connectors selected delivers the data in such a way that it satisfies the distribution scenario. To target our evaluation, we will focus on the four representative performance properties of data distribution (recall section 3, dimension 3). We will quantifiably measure consistency (data delivered is data sent), efficiency (memory footprint and data throughput), scalability (data volume and number of hosts) and dependability (uptime, number of faults). Since we have access to large amounts of data from all three systems, we will further validate that the approach is independent of the scientific domain in which it is used. We will compare our results to the state-of-the-art off-the-shelf distribution connector solutions.

7. Research Contribution and Conclusion

Our contributions include: (1) A comprehensive, classification framework to identify software connectors to satisfy large-scale data distribution problems, (2) A software framework for deploying, adapting and combining connectors to deliver large volumes of data, and (3) Composable, adaptable and replaceable data-intensive software connectors.

8. References

- [1] J. S. Hughes, et al., "The Planetary Data System. A Case Study in the Development and Management of Meta-Data for a Scientific Digital Library.," in *Proc. ECDL*, pp. 335-350, 1998.
- [2] C. Mattmann, et al., "A Classification and Evaluation of Data Movement Technologies for the Delivery of Highly Voluminous Data Products," in *Proc. MSST2006*, pp. 131-135, 2006.
- [3] J. Postel, et al., "File Transfer Protocol (FTP) RFC Document," RFC 1985.
- [4] W. Allcock, et al., "GridFTP: Protocol Extensions to FTP for the Grid," RFC Draft Document 2001.
- [5] "Aspera Software <http://www.asperasoft.com>," 2005.
- [6] D. Bush, "UFTP - UDP based FTP with multicast. <http://www.tcnj.edu/~bush/uftp.html>," 2005.
- [7] N. Mehta, et al., "Towards a Taxonomy of Software Connectors," in *Proc. ICSE*, pp. 178-187, 2000.
- [8] R. J. Allen, et al., "A Formal Basis for Architectural Connection," *TOSEM*, vol. 6, pp. 213-239, 1997.
- [9] M. Shaw, "Procedure Calls are the Assembly Language of Software Interconnections: Connectors Deserve First-Class Status," in *Proc. Workshop on Studies of Software Design*, pp. 17-32, 1993.
- [10] M. M. Kandé, et al., "Modeling Crosscutting Concerns using Software Connectors," in *Proc. OOPSLA'2001 Workshop on Advanced Separation of Concerns*, pp. 2001.
- [11] D. Balek, et al., "Software Connectors: A Hierarchical Model," Charles University, Prague, Tech. Report 2000/2, 2000.
- [12] B. Spitznagel, et al., "A Compositional Formalization of Connector Wrappers," in *Proc. ICSE*, pp. 374-384, 2003.
- [13] F. Arbab, "A Channel-based Coordination Model for Component Composition," CWI Technical Report SEN-R0203, 2002.
- [14] K. Lau, et al., "Exogenous Connectors for Software Components," in *Proc. CBSE-8*, pp. 90-106, 2005.
- [15] D. Garlan, et al., "Architectural Mismatch or Why it's hard to build systems out of existing parts," in *Proc. ICSE*, pp. 179-185, 1995.
- [16] C. Gacek, "Detecting Architectural Mismatch During Systems Composition," Ph.D. Dissertation, Univ. of Southern California, 1998.
- [17] M. Maybee, et al., "Q: A Multi-Lingual Interprocess Communications System for Software Environment Implementation," *The First Collected Arcadia Papers* 1993.
- [18] E. Dashofy, et al., "Using Off-the-Shelf Middleware to Implement Connectors in Distributed Software Architectures," in *Proc. ICSE*, pp. 3-12, 1999.
- [19] C. Mattmann, et al., "A Software Architecture-Based Framework for Highly Distributed and Data Intensive Scientific Applications," in *Proc. ICSE*, pp. 721-730, 2006.
- [20] W. Emmerich, "Software Engineering and Middleware: A Roadmap," in *The Future of Software Engineering*, A. Finkelstein, Ed., 2000, pp. 117-129.
- [21] A. Zarras, "A Comparison Framework for Middleware Infrastructures," *J. of Object Technology*, vol. 3, pp. 103-123, 2004.
- [22] C. Mattmann, "Software Connectors for Highly Distributed and Voluminous Data Intensive Systems," Tech. Report USC-CSE-2006-601, 2006.
- [23] M. Franklin, et al., "A Framework for Scalable Dissemination-based systems," in *Proc. OOPSLA*, pp. 94-105, 1997.