Efficient Support for Sophisticated Interactions between Entities in Distributed Brokering Systems

Shrideep Pallickara and Geoffrey Fox
{spallick, gcf}@indiana.edu
Community Grid Laboratory
Indiana University

Applications and services are increasingly becoming network centric. The devices, with which applications and services need to interact, span a wide spectrum that includes desktops, PDAs, appliances, and other networked resources. Entities – which abstract users, resources and proxies thereto – within these systems communicate with each other through the exchange of events, which are essentially messages with timestamps. These events encapsulate information pertaining to interactions such as transactions, data interchange, system conditions and finally the search, discovery and subsequent sharing of resources.

Entities specify the type of interactions they are interested in. This usually involves a specification of the constraints that fields within the event (encapsulating the interaction) must satisfy prior to delivery to the entity in question. We refer to this evaluation of a constraint against a stored event as the matching process.

This is usually performed in a matching engine, which evaluates an event against a set of stored subscriptions. Depending on the type of the event and the complexity of the evaluated constraint there could be multiple matching engines residing within the system. For example, there could be a matching engine responsible for evaluating constraints specified using regular expressions and another for SQL-conforming constraints.

The matching process returns a set of destinations corresponding to the entities that specified the matching constraints. It is however infeasible to store have every matching engine residing at every messaging node to compute destinations based on calculations performed over the entire set of constraints present within the system. It is thus very important to efficiently organize these constraints.

Scaling, availability and fault tolerance requirements entail that the messaging infrastructure hosting these entities, and routing their interactions, be based on a distributed network of cooperating nodes. The processing and servicing of events is, thus in itself, a distributed problem that involves several nodes and the links that connect them.

We suggest that inefficient approaches to either the calculation of, or routing to, destinations can result in unacceptable network degradations. Inefficient utilizations of the communal resource – in this case the underlying networks and networked resources – degrades the value of the resource for other applications/infrastructures that utilize this resource. Entities, applications and services alike are then exposed to problems concomitant in such solutions. This usually leads to degraded response times at the entities and bottlenecks at nodes. It is clear that as interactions between entities get increasingly sophisticated and network bound the stress placed on the network fabric (comprising the CPU and network cycles available) increase considerably.

We can identify three very important factors in any system that seeks to provide support for sophisticated interactions between entities. First, the organization of the nodes within the distributed messaging infrastructure plays a very crucial role. A sophisticated node organization scheme can allow the messaging infrastructure to scale to support entity configurations of arbitrary size. Furthermore, such a scheme also facilitates efficient constraint organization strategies, which in turn can enable sophisticated real time interactions.

Second, the underlying solution should support sophisticated search/discovery and matching of events while allowing arbitrarily complex applications to be built upon these solutions. Resident matching engines need to provide support for increasingly complex and sophisticated qualifiers, for specifying constraints, that events should satisfy prior to being considered for delivery to applications. Matching engines need to evaluate events against stored constraints to compute target destinations.

Finally, destinations computed by the matching engines need to be routed by the messaging infrastructure. This routing should eschew paths that would involve failed or failure-suspected messaging nodes. Furthermore, the paths traversed to reach the final destinations must be along shorter routes with lower communication hops and lower overall communication latencies.

Solutions that address these issues obviate the need for every node to maintain the list of all constraints within the system. As a result every node does not perform matching, over the entire set of constraints, for every event that is received. This in turn implies that the computation costs are significantly lower. Furthermore, this cost of computing destinations is amortized over the nodes involved in the process of computing destinations.

We have investigated matching, routing and network utilization issues in the context of our research system NaradaBrokering [1,2]. NaradaBrokering addresses the three important factors to enable efficient dissemination of interactions. First, the cluster based architecture ensures that the system can scale to support entity configurations of arbitrary size. Second, NaradaBrokering incorporates support for multiple matching engines residing in the system. The queries could be SQL (from JMS) or XPath based and the topics could be Integers, "/" separated strings or <tag, value> tuples. The constraint organization scheme within the system ensures that destinations associated with interactions are computed efficiently and also that the number of operations performed to compute these destinations is low. Finally, there is the routing of events to the final destinations. The system does this routing by selectively deploying valid brokers to achieve the disseminations.

The organization of constraints, matching and finally the routing of events to valid destinations exploit the underlying cluster-based hierarchical topology deployed in NaradaBrokering.

References

[1]. The NaradaBrokering Project at Indiana University. http://www.naradabrokering.org .

[2]. Shrideep Pallickara and Geoffrey Fox. NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. (To appear) Proceedings of the ACM/IFIP/USENIX International Middleware Conference.