Folk Memory

A Minimalist Architecture for Adaptive Federation of Object Servers

Ward Cunningham
June 20, 1997

This memo describes an adaptive architecture for federating peer object servers. Both the architecture and its implementations are called *Folk Memory*. This term alludes to the social process by which folk tales or folk songs are remembered and propagated within a culture. The folk servers, like people within a culture, share their interests with their associates and associate with those who share their interests. Folk servers stream domain objects over simultainous connections to peers dynamically chosen to shorten access to the authoritative sources for objects deemed *interesting*. Conflicts are resolved by a collective process that may not settle to identical state in all parts of a network. Folk memory favors scaleability and robustness over accuracy and determinism.

Principles

While listening to papers presented at COOTS'97 I became frustrated that so many requirements were heaped on every distributed object architecture. Surely there is a class of application where delay and uncertainty are acceptable or even desirable. What would be a suitable infrastructure for these applications? I doodled, sketched and crumpled. Then, during the final panel discussion, my thoughts for an alternative came together with these eight principles:

- A server contains pools of interconnected objects.
- Connecting to peer server initiates bi-directional stream of objects without query.
- Inflow suggests schedule for outflow.
- Objects carry trail to more authoritative servers.
- Update implies elevation of authority and therefore trail reversal.
- Trail reversal guides the establishment of new connections.
- Transportation of objects is by breadth first traversal pruned by trail length.
- Pruning is by substitution of inert proxy (a.k.a. Ellipsis).

Concepts

Here I elaborate the above principles by describing the purpose or responsibility of various conceptual components. These collectively form a distributed object infrastructure upon which high-volume, world-wide, non-stop and incremental applications can be built.

Object: The unit of communication between servers. Objects may refer to other objects, but those references may be temporarily elided as a consequence of transmission. A transmitted copy of an object is a less authoritative vestige of its source.

Trail: Objects maintain a trail that records a sequence of servers through which the object has been copied. The root of a trail is presumed to be the server hosting the most authoritative copy of an object, the primary authority.

Ellipsis: A mark that stands in place of a reference within an object on a server. Unlike a proxy, the ellipsis offers no access to the object it replaces. An ellipsis will spontaneously transform into a native reference should any vestige of the elided object appear on the server. The transformation event can trigger computation within an object or its observers.

Promotion: An object may spontaneously raise its own authority, perhaps due to locally acquired new state that supersedes that of the source copy. Promotion initiates a process of trail reversal where the revised object is distributed to previously authoritative servers. Promotion is to prime authority, and is quick. Shorn trails then persist only as required by connection and distribution heuristics.

Resolution: An object can be assembled from various sources of various and changing authority. The resolution process guarantees that only one copy of an object will exist per server, and that updates to its non-reference state is atomic. {Should the object participate in its own resolution?} Identity confusion is possible, thought unlikely, among objects of the same kind.

Server: A server hosts objects of various authority and shares them through bi-directional connections with other peer servers. Servers have a network identity which is recorded in the trail of objects it distributes. Peers may connect with a server in the hope of recovering more authoritative information about an object, for example: revised state or copies of elided references. A server does not respond to queries. Rather, it responds to a connection by sending copies of objects by a schedule of its own devising.

Connection: Servers establish bi-directional symmetrical point-to-point connections for the purpose of sharing objects. A server simultaneously distributes outbound objects and collects inbound objects. A server may terminate a connection at any time and will often do so to conserve system and CPU resources or because of apparent blockage or other failure of the channel. Heuristics may depend on knowing which objects have been collected from which channels (hence pools).

Extinction: An object that is no longer distributed with primary authority is in danger of extinction. A server has some obligation to continue distributing objects for which it is the

prime authority. Should such a server fail, an object may still be saved by promotion on another server. An object cannot be deleted--only forgotten which may lead to eventual extinction.

Transmission: Objects are transmitted using an open-loop wire protocol on top of a stream protocol such as tcp/ip. Class definitions (bytecodes) are transmitted with objects unless they are known to already exist on the receiving end. {Or should the receiver request missing classes thus introducing closed-loop behavior?}

Heuristics

The architecture includes several heuristics for allocating limited resources. I think it likely that simple heuristics can provide robust and effective total system behavior, though the actual selection of these heuristics may require substantial analysis or experimentation.

Connection Heuristic: Chooses when to open, accept and close connections. Guided largely by "trail climbing" of interesting objects, and average channel performance. Allocates sockets, total bandwidth, and processing time devoted to transmission.

Distribution Heuristic: Chooses which objects, and in what order (schedule), to distribute through each open channel. Guided by authority, apparent interest from hosted application, and interest shown by peer servers. Allocates channel bandwidth to individual objects. Strongly effects distribution latency.

Preservation Heuristic: Chooses which objects will be kept in the server. Guided by authority, network and application interest, connection behavior and available space. {Can a native reference be elided by this heuristic?}

Applications

Here I consider some applications that might find such a platform desirable. Each requires a large, structured and continuously growing repository for information, much of which is provided by the users themselves. I expect the ambiguity and non-determinism of the platform to be reflected through the application to these users, who are well prepared to deal with this reality.

Users would enter the system when they launch the application with its embedded server and providing the freshly started server with the network name of a currently running, but otherwise undistinguished, peer.

Many-User Game: A MUD or MOO could represent rooms, characters and possessions as distributed objects. Connections would be made or broken as players inserted their characters into new rooms. Possessions could be left or dragged along behind a character by authority promotion as they are manipulated in each new environment. The natural

desire to avoid overcrowded rooms would prevent server traffic concentration. Since each player contributes additional server capacity, their participation in very large games would still be welcome.

Wiki-Nature Documents: Hyperlinked documents could span many subjects, authors and usage patterns. Trail climbing would lead toward a distribution of objects on servers congruent with the hypertext structure of the document. Agent mediated searches would be breadth first from the point of initiation, and would yield continuously growing results limited only by the initiator's patience. {What about RecentChanges?}

Customer and Market Information: A world-wide securities sales force could track market sectors and customer preferences in an environment where one records many facts that are expected to obsolesce quickly.

Related Work

Tom Ray's Digital Preserve is about as biological of a network as one can imagine. If my use of statistical properties seems far fetched, then read what Tom is doing.
http://www.hip.atr.co.jp/~ray/pubs/reserves/reserves.html

Chip Morningstar's company, Electric Communities, has developed Global Communication Protocols which support secure, peer to peer networking. Chip's game, *Habitat*, is the sort I imagine deploying on folk memory.
http://www.communities.com/

Jim Waldo and others at Sun are building a layer on RMI called JavaSpaces which offers a narrow interface through which applications pass objects. Their middleware need not understand the application or the interfaces it might use. There are linkages to security and transactions.
http://chatsubo.javasoft.com/

Cache-Only Memory Architectures (COMA) are the latest refinement of *Attraction Memory*, first proposed in a rather complicated hierarchical and hardware assisted form. Saulsbury, et.al. describe how to use standard MMUs to gain much of the same effect.
http://playground.sun.com/pub/S3.mp/simple-coma/isca-94/paper.html