Brian's Peer-to-Peer Computing Network

Abstract

An architecture for a true peer-to-peer computing network is proposed. The network provides distributed, redundant data storage, directory services and computational resources. Data storage consists of a virtually unlimited number of virtual "Cells", copies of which may reside in any of the computer "Nodes" in the network. Computational requests are made to the network by placing Java applets within the network data space. These applets access data Cells and compute results which are placed in new Cells made available to the network.

A single supervisory program is envisioned. The supervisor is a compact, open-source implementation which allows the Node administrator to select limits for local storage allocation, connection bandwidth and processing priority.

The selection of appropriate computational tasks and their partitioning for efficient solution is beyond the scope of this document.

Introduction

Peer-to-peer file-sharing networks possess many of the requirements of a distributed computing network. Their key features include (1) direct, peer-to-peer data communication, (2) a file transfer mechanism, and (3) a distributed directory structure.

Existing file-sharing networks suffer from some significant limitations. The same file may exist with many different names. There is no annotation mechanism to allow independent nodes to confirm the contents of a given file, or to warn of spam or viruses. Files must be explicitly requested by a user and then must be explicitly shared with other users. There is no automatic mechanism to ensure the propagation of large or high-demand files to additional nodes; the presence of such a mechanism would act as a self-leveling traffic function and eliminate communication bottlenecks.

Adding a computation feature to the file-sharing architecture would result in a truly generalized peer-to-peer network. The current proposal represents an improved file-sharing core, supplemented by automatic execution of selected Java applets whose code and data traverse the network as shared files.

Definitions

The following terms are used in a particular technical sense when describing the operation of the network.

Node - A computer within the network. Nodes provide computing and storage resources to the network via their Internet connection.

Node Identifier - Nodes are identified by IPaddresses. Additional provisions may be required to allow this to work through firewalls, across Intranets and in the presence of dynamic IP address assignment.

Cell - A unique dataset. Cells may contain arbitrary amounts of data, up to 4 GB. Cells are identified by a unique binary Cell Identifier based on their contents. Cells may be copied from one Node to another and an individual cell may exist on many Nodes. Cells are fabricated at a particular Node and their existence and description is published in a Directory Entry. Once created, the contents of a cell never change; any change in the data would result in a different Cell Identifier.

Cell Identifier - Each cell has a unique binary identifier composed of the length of the data in the cell and the MD5 digest of that data. Use of a 32-bit data length field and the 128-bit MD5 digest yields a 20-byte binary value that makes a reasonable Cell Identifier.

Directory Entry - For each Cell a Directory Entry couples a text description with the Cell Identifier and the Node identifier that can be used to access that Cell. Since a Cell may reside on multiple Nodes, Nodes can make additions to the Directory Entry. Directory Entries are time-stamped when created or modified.

Directory - Cells may contain one or more Directory Entries. These cells are copied from Node to Node, and form the distributed Directory for the network. Processes running on individual Nodes choose which Directory Entries to publish and how often updated Cells containing Directory Entries are to be fabricated.

Operational Description

Network communication takes the form of IP Datagrams. It is unnecessary to establish sessions between Nodes. Each Datagram is one of two types: a request for a particular Cell (or portion thereof), or an actual portion of a Cell (up to 32 KB in a chunk). This structure allows for both "push" and "pull" transfer paradigms since data from Cells may be sent unsolicited or after receiving a request.

A Node knows that a complete Cell has been received when all of its blocks have been received and the MD5 digest matches. Note that the "chunks" may arrive in any order and from multiple sources. Once a complete Cell is received it becomes available for local processing operations. A new Directory Entry may also be created that shows this Node as a holder of the Cell. This new Directory Entry publishes the availability of the Cell for other Nodes.

System Features

The operating system for each Node must perform a standard set of functions to communicate with other Nodes, manage Cell storage, and process Java applets. Java extensions will allow access to selected operating system functions. A user-interface application will allow data Cells to be added to and copied from the network.

Cell Copying - In general, Cells are pulled from a host Node (A) to a new Node (B). This can be accomplished because Node B obtained a directory entry describing the Cell and listing Node A as a source. Node B

Directory Searches - Applets conduct searches of the directory entries contained in Cells on the local Node. These searches are essentially wildcard searches of the text description contained in the directory entries. Distributed searches are accomplished by sending directory search applets to other Nodes. Upon discovering the target directory entries these remote searches forward their results to the Node that made the original request.

Time Synchronization - The operation of the Directory Entry timestamp mechanism presupposes that the network maintains approximate synchronization across Nodes. This is also required to ensure that distributed searches expire gracefully.

Java Execution - The Node operating system will periodically scan the directory entries contained in the Cells it is hosting. Certain criteria will be used to identify candidate requests for scheduled execution. If the Cell described by a candidate directory entry is not present on the local Node a transfer request will be initiated. When the complete, valid Cell exists on the Node execution will commence.

Application Examples

A simple example of the utility of this network architecture is demonstrated by the peer-to-peer file sharing system.

Design Criteria: (1) True peer-to-peer with each node running the same software; (2) distributed directory; (3) arbitrary file sizes; (4) transfers from multiple sources; (5) elimination of duplicate files; (6) directory annotation.

(To be completed)