Technical Disclosure Commons

Defensive Publications Series

May 06, 2015

Dynamic Computational Workload-Based Co-Processing

David Zeuthen

Alejandro Deymonnaz

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation

Zeuthen, David and Deymonnaz, Alejandro, "Dynamic Computational Workload-Based Co-Processing", Technical Disclosure Commons, (May 06, 2015)
http://www.tdcommons.org/dpubs_series/73

This work is licensed under a Creative Commons Attribution 4.0 License.

This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.

Dynamic Computational Workload-Based Co-Processing

Abstract

A computing device may dynamically determine a strong coprocessor to perform peer-to-peer updates. The computing device may advertise its hardware capabilities as services to enable co-processing and peer-to-peer updates. Based on the information provided by other devices in the local network, the computing device may automatically send work to some devices with more processing capabilities. The computing device may be configured to automatically offload computation from less powerful devices (e.g., mobile phones) to other more powerful devices (e.g., personal computers) in a local network. The computing device may also receive at least a portion of one or more files from other devices from the local network and share at least a portion of one or more files to other devices in the local network. A dedicated computing device or a super node may be used to download auto-updates and perform peer-to-peer auto-updates.

Background

Software distributors often provide a software file, such as an operating system, firmware, or an application, to a large number of client machines over the Internet. The client machines can include computing devices such as personal computers, laptops, tablets, and mobile phones. Unfortunately, distributing files over the Internet to a large number of client machines may use a lot of data and potentially cause network congestion.

One solution to such a problem is to cache the needed files to the server. However, such a solution requires large files to be saved on the local network and requires a specialized network that allows caching. The space for caching may not support a large number of files, and the client machines may need to separately connect to the Internet to access the files.

Using local computing devices to share software files alleviates data used by a plurality of computing devices without relying on the capabilities of the network. A computing device may advertise its capabilities and offer services on the local network. With this technology, the user can share software files with other users in the local network, which means that only one user needs to obtain a copy of the file or an update over the Internet, instead of a plurality of users needing to access the Internet to receive the update. This alleviates other users from using data to obtain a copy of the file or an update, and reduces network traffic. In addition, users in the local network will be able to download the software files faster, because they are downloading from other users in the local network instead of downloading from the Internet.

Furthermore, the computing device may advertise computational services, such as factoring an integer or running a script. For example, other computing devices in the local network may use these computational services to help lessen their computational load. During a computationally extensive operation, such as factoring, the computing device may detect other available computing devices in the local network that are advertising computational services and may take advantage of the computational services. Other services that computing devices may advertise include: 1) analyzing recorded data; 2) recording data from local environment; 3) processing transactions; and 4) running a computer simulated world.

Features of the subject technology

The subject technology relates to a computing device that may advertise its sharing capabilities and offer services on the local network that it is connected to (e.g., via WiFi or Ethernet). Different protocols such as mDNS/DNS-SD protocols may be used to advertise based on the standards used by the computing devices. For example, the computing device may send an IP multicast query message that requests other computing devices in the same network to

identify itself. Each of the other computing devices in the same network responds by multicasting a message that includes, for example, its IP address, and its capabilities. Since multicast is used, every computing device may cache replies in order to answer future queries without sending out query messages. Cache invalidation may happen via both timeouts and explicit cache invalidation messages.

The computing device may also advertise a number of service endpoints or nodes using the same protocol. Each of the service-endpoints is identified by a textual string, and could be anything from file sharing (for example, via HTTP) to computational services such as factoring an integer or running a script. The computing device may also advertise a name (typically a random hexadecimal string) for identification.

For each service endpoint, the device may also advertise a scalar value expressing the load of the service on the device. For example, a low scalar value could indicate availability of spare processing cycles on the device, and a high scalar value could indicate a low availability of processing cycle. The computing device may have a high scalar value (for example, 10.0) indicating a high load on the computing device when the user is, for example, editing a document, watching a video or participating in a video conference. The computing device may have a low scalar value (for example 0.5 or 0) indicating a low load on the computing device when the computing device is idle.

The scalar value expressing the load of service on the device may further depend on the number of services being offered by the device to other computing devices on the local network. For example, a computing device may send a message to advertise that it has the file requested and a load of 0.0. If the load for the computing device changes, then the computing device may send a follow up message with the new load. The message may further comprise information

indicating how long the message is valid for. This message may be saved in cache in other computing devices and may be used. A message containing cache invalidation may be sent in response to a change in the load of the computing device to invalidate the previous message stored. This means that those other computing devices need to query the computing device to receive the latest information.

In another embodiment, the scalar value expressing the load of services on the device may be the number of active downloads or it may be the bandwidths currently being used. The person skilled in the art would be able to create an algorithm to calculate a scalar value for the load based on available hardware in the computing device and running application services.

Furthermore, each service end may expose additional data that may help other devices easily determine if the service is useful. For example, for a file sharing service, a list of shared files and the sizes of each file may be exposed. If a user of a computing device requires a particular file, then a message is sent out to other computing devices in the same local network. If the file is not in the list of shared files by the node or the size of the file provided by the node does not match the requested byte-range, then the node is not considered.

The computing device may be further configured to consider all nodes on the local network when it is choosing a device to offload workload to. For example, the computing device may consider all the available nodes that are offering computing services, and consider the 10% least loaded devices to utilize. The computing device may further determine the node to utilize by selecting randomly among the 10% least loaded device. In another embodiment, the computing device may choose the least loaded device to offload workload to.

In addition to regular end-user computing devices, it is possible to deploy “super”-nodes, which may be a computing device with additional computational resources (e.g. additional

network bandwidth, faster CPU etc.). The "super"-node may act like other regular nodes except that its advertised scalar value expressing the load will be lower due to the additional computational resources, and therefore has a higher likelihood of being chosen for a task.

An exemplary embodiment of the invention with four nodes is shown below as FIG. 1.

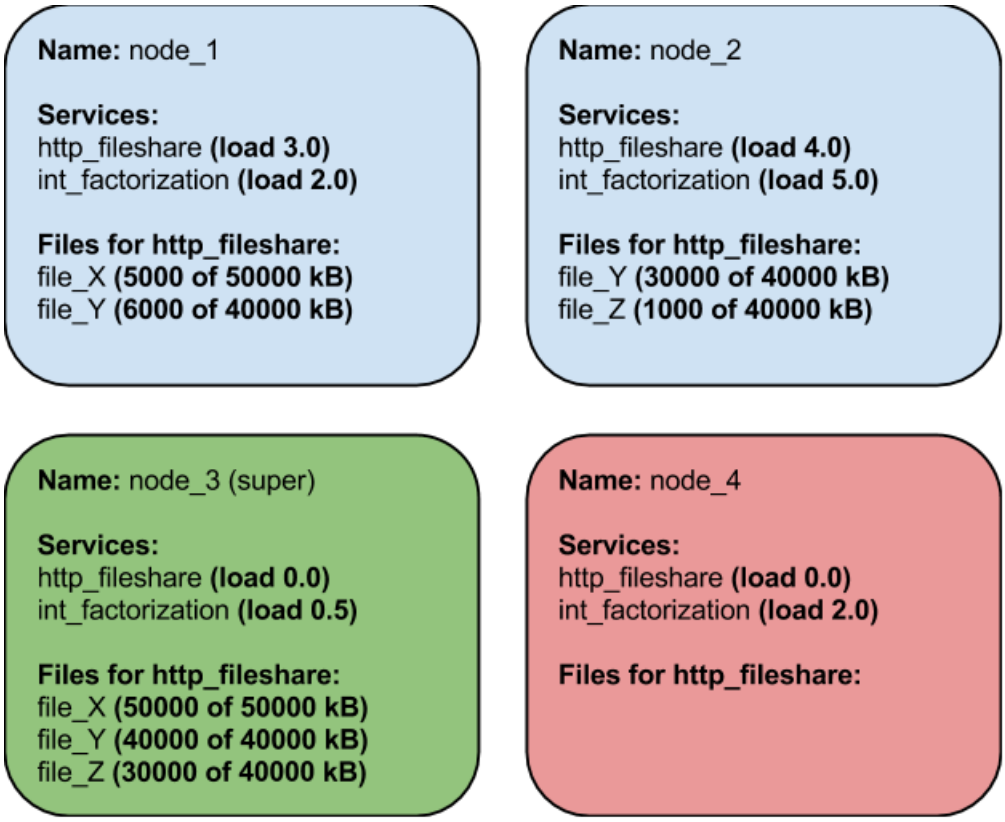

FIG. 1

Each node depicted in the figure is a computing device offering two services "http_fileshare" and "int_factorization." The http_fileshare service may additionally advertise which files it is hosting and how many bytes it has for each of the files. Figure 1 depicts four computing devices, node_1, node_2, and node_3 and node_4, and three files file_X, file_Y and file_Z being shared. Computing device node_4 is not sharing any files yet.

In one embodiment, node_1 through node_4 in the same local network are updating the operating system by retrieving file_X, file_Y, and file_Z. In another embodiment, node_1 through node_4 in the same local network are sharing files file_X, file_Y, and file_Z. Node_3 has downloaded all of file_X, file_Y, and file_Z by accessing the web. When the user of node_4 initiates an update or requests certain files, node_4 sends a multicast to other nodes in the same local network. Node_4 can use node_1 for the first 6000 kB and node_2 for the next 24000 kB but has to go to node_3 for the last 10000 kB.

While node_4 is downloading file_Y from each of the different nodes, it can start sharing this file immediately, which allows other nodes to access the file.