File Transfer Using a Web Caching Proxy
Version 2
by Carl Vuosalo

The File Transfer (FT) class provides functionality to transfer files between machines. It includes the capability to transfer files via a web server.

During the HTCondor job life cycle, FT objects transfer input files from the submit machine to the execute machine. Input files are transferred for every single job. Even if some of those files are identical, every job gets its own copy of the files. This process wastes network and I/O resources by repeatedly transferring identical files.

A method to reduce this waste is to perform file transfer via web server and employ a web caching proxy like Squid. The first time a file is transferred, a copy will be stored by the caching proxy. Then, all subsequent requests for the file will be served from the cache, thereby reducing the load on the submit machine.

This approach of using a web caching proxy allows HTCondor to leverage the advanced functionality of the caching proxy in managing caches. Issues regarding the optimization of cache usage are left to the administrator of the caching proxy rather than being explicitly addressed by HTCondor itself. A more comprehensive approach, where all the caching functionality would reside in HTCondor itself, is a possible enhancement in the future if the caching proxy method proves to be inadequate.

Access to input files will be provided by a web server. Any standard web server can be used, as long as it can respond to If-Modified-Since queries and fill in the “Expires” header in HTTP responses. The web server should allow configuration of a default expires time for files, and a recommended value for this time is 20 minutes. After this time, the caching proxy will perform an If-Modified-Since query to the web server when a file is requested. A longer expires time prevents unnecessary queries of the web server, but if it is too long, a cache file may become out of date without the proxy refreshing it. The web server must be configured to listen on the “localhost” IP address. The HTCondor configuration parameter WEB_SERVER_PORT will specify the port number.

To set up file transfer caching, the ENABLE_CACHE_TRANSFERS, ENABLE_URL_TRANSFERS, and WEBSVR_FOLLOW_SYMLINKS configuration parameters must be set to TRUE. Also, the WEB_ROOT_DIR parameter must be set to point to the directory that will hold the input files to transfer. This directory can be the same as the spooling directory.

The web caching proxy needs to be provided sufficient hardware resources to handle anticipated loads. The site administrator needs to consider the number and sizes of files to be cached, and the maximum number of jobs that will start up at once and request files from the caching proxy. Other than adequate hardware resources, no special configuration of the caching proxy is required. The appropriate environment variable must be set with the caching proxy's IP address and port number.

In the submit file, the user specifies with the PublicInputFiles parameter those files that can be cached. If any of these files also appear on the TransferInput list, they will be ignored on the latter list. These files must accessible and readable by any user on the system so that the web server can access them. Files that are unique to each job should not be on the public list, since there is little point in caching such files. Since the public files will be accessible via a web server, they would be potentially available to anyone, though their file names will be opaque. The caching proxy and web server will refer to

these files by an MD5 MAC hash of their file names. These 16-character hash names are expected to be always different for different file names.

As the shadow prepares to transfer a public input file, it will create a link in the web root directory to the actual public input file using its hash name. If the link already exists and correctly points to the file, its modified time will be updated. If the actual file is not world accessible and readable, a “not world readable” error will be returned for this file. condor_preen will include the web root directory in the list of directories that it examines, and it will delete any links that are more than a week old or that point to non-existent files. It is expected that the number of public files used at a site will not be large enough to fill up a filesystem in a period shorter than a week. There is no protection against the user changing the input files while the task is running.

In the file transfer process via web server, the starter will issue an HTTP request using the hash file name via the curl plugin. The request will go to the caching proxy, which may serve it directly if it has a valid cache copy, or the proxy will pass on the request. The request will then go to the Shared Port Daemon (SPD), which will route it to a fixed named pipe configured for the web server. A connector daemon will be listening on this pipe, and it will pass the request to the “localhost” IP address, where the web server is listening.

Several alternative designs were considered. Putting a gSOAP web server into the shadow was an option rejected because it would significantly increase the memory footprint of the shadow, and shrinking the size of the shadow has been a major priority. Another idea of having the shadow fork a process that would then load the web server libraries was rejected as being too complex even though it would overcome the problem of bloating the shadow. Alternatives to a public input file list were considered, but since the files could be cached by a caching proxy that could be running on a separate machine, the user needs to be clearly informed that the file contents are not completely secure. Any sort of automatic determination of which files to cache or privileged access to input files could, without the user's knowledge, expose private files or create security holes. The extra burden of listing the public files and making them world-readable seems the simplest and most consistent way to ensure that only truly public files can end up in the cache.