Concept Document:  
Shared Data for HTCondor

Author: Brian Bockelman

# Use Case and Goals

On OSG, we have repeatedly observed users in the following situation: a user has a set of jobs (say, 1,000) submitted at once. Each job has a small input sandbox (1KB) and all have a large common input file (1GB). We assume the user has access to 200 machines.

Currently, we use HTCondor to schedule jobs and will transfer the input data and sandbox for each job. So, the input data is always transferred 1,000 times to complete all the jobs. Any decrease from 1,000 transfers of input data is considered an improvement

Our goal is to minimize the number of times the input data is transferred from shadow to starter by introducing the concept of *shared data* and prestaging this data to nodes. This is the primary concern for our use case is the cost delivering input data over the network – further reduction of resource usage is secondary.

We consider this an HTCondor problem and requires a solution completely within HTCondor.

# Introduction

For each job, we divide the input sandbox in two:

* *Private*: This is the set of unique input files required to run the job. Each job will have a different input sandbox.
* *Shared*: This is the set of input files required to complete the job, but not necessarily unique to that job.

Currently, we do not differentiate between private and shared data – all input data is transferred from submitter to worker node for each job.

We assume the user has authorization to pre-stage files to half of the machines, but has would like to run on all. We currently do not manage the space on the machines. We do not plan on having the scheduler aware of the nodes with the prestaged input versus those without.

# Concepts

We introduce a new HTCondor concept – *shared data*. The schedd and startd will create and own a local cache directory. Jobs will be required to differentiate between the data considered *private* and the *shared*. Shared data is explicitly considered to be public – any user has the ability to read any other user’s shared data.

The startd shared data directory will be owned by the startd but, for now, we assume it is managed by the owner of the machine.

### Before Job Submission

Before the user submits their jobs, they will prestage the data to the schedd. The files will be spooled to the schedd; the spooled version of the file is considered “correct” for any subsequent jobs. Any later updates the user does to the file will be ignored.

The schedd goes through the following steps to prestage:

* The file is spooled from the user to the schedd.
* The schedd will compute the hash of the file and store the association between the filename and the hash.
* For each worker node matching a given constraint, the schedd will attempt to prestage the file to the cache on the worker node.
* Each machine will verify the hash.
* The prestaging process will create a lease for the spool on the remote machine; the lease will be renewed for as long as the spool exists on the submit host.

We believe the caching logic on the submitter and the worker node is symmetric – the same implementation (“condor\_cached”) should work on both.

Each shared data directory instance should be:

* **Monitored**: usage statistics should be reported to the collector.
* **Managed**: have a limit on the volume of files allowed in the cache. Currently, we do not plan on doing per-user limits.
* **Cleaned**: When leases expire, files should be deleted from the cache.

### Job Submission

When a job cluster is submitted, the user can specify:

* shared\_input\_files = LOGICAL=LOCAL

Where LOGICAL is the filename registered in the schedd spool and LOCAL is the desired name of the file in the worker sandbox. The schedd will lookup the file name’s hash at submit time and record it in the jobs’ classad.

### Job Startup

When the shadow starts a job, the shared data files will be referenced by their hash. If a shared data file is already present in the startd’s shared data directory, the local copy of the file will be copied to the job sandbox. If the file is not present, it will be transferred from shadow to starter.

The name of the file in the job sandbox will be set to LOCAL.

For calculating transfer queue statistics (between the schedd and starter), we will assume that all jobs will always copy the shared data to the worker node.

### Spool Layout

The startd and schedd will have a sub-directory (“*shared\_data*”), owned by the condor user. The sub-directory structure of *shared\_data* will be such that, given a unique ID, we can deterministically calculate the associated file in the *shared\_data* directory. We also will hash the file path to avoid more than 1,000 files per sub-directory.

The cache code will also maintain a mapping between LFN and hash in this directory.