# HTCondor Vanilla Universe Remote IO

## Problem Statement

There are two primary features in the standard universe missing from the vanilla universe:

1. **Remote IO**: ability for the job to access the submit node through POSIX functions.
2. **Checkpointing**: ability to stop a process on one node and restart it from another.

The implementation of these features is complex, costly to maintain, and not used by a majority of HTCondor users as standard universe places several restrictions on the processes run.

While DMTCP and CRIU have provided a path forward to provide generic checkpointing without as many limitations, not much work has been done to provide POSIX remote IO.

This document describes a mechanism for providing remote IO to arbitrary vanilla universe jobs without relying on recompilation or other interposition techniques.

## Proposed Architecture

I propose combining three feature to provide POSIX remote IO for vanilla universe jobs:

1. **Chirp IO**: Condor already provides a built-in protocol for accessing the submitters filesystem. Chirp provides a POSIX-like API; combined with parrot’s ptrace-based interposition, provides remote IO for jobs run under parrot.
2. **FUSE**: FUSE, or filesystems-in-userspace, provides a mechanism for implementing POSIX filesystems in a user process.
3. **Mount namespaces**: A set of processes (such as a job) can have a set of mounts different from the system mounts. HTCondor uses mount namespaces in several different features. When all the processes in a job exits, any filesystem unique to that namespace is automatically unmounted.

We will implement a new FUSE filesystem with chirp IO. Any accesses to the filesystem will be translated into Chirp calls and authenticated by the existing mechanisms.

The job will be put in a separate mount namespace so the filesystem is only visible to the job and automatically unmounted at job end.

## Implementation

The implementation is a straightforward combination of existing components; an initial implementation is posted to V7\_9-vanilla\_remote\_io-branch. The implementation adds a new executable, condor\_remoteio, which implements a FUSE filesystem and links against libfuse. This executable is run before the user process execs. If the mount fails, the job is put on hold. To prevent /etc/mtab from being modified, it is mounted as read-only before running condor\_remoteio.

Condor must be run as root in order to use filesystem namespaces.

There are two new job ad knobs:

* **WantIOProxy**: When set to TRUE, enables the IO proxy in the shadow and configures a chirp.config cookie so the client can authenticate. The submitter’s root filesystem will be available at /condor/submitter in the job.
* **WantRemoteRoot**: When this and WantIOProxy is set to TRUE, the submitter’s root filesystem will also be the root filesystem in the job.

The configuration knob **ALLOW\_REMOTE\_IO\_MOUNT** controls whether the filesystem is available at all.

## Development Plan

* Do appropriate cmake work to enable this for UW and non-UW builds [1 day].
* Submit to NMI and work through bugs [1 day].