Persistent Storage for the Determinator OS

Jonathan MacMillan

Project Proposal - 9/22/11

I. Overview

Determinator is a distributed, multiprocessor OS. It guarantees that the same computation run multiple times will return the same result. (See [1] for a more detailed overview of the Determinator project.) Presently, Determinator has no method of persistent storage - whenever a machine running the Determinator kernel is shut down (intentionally or otherwise) all of the data in processes is lost. This, of course, is an undesirable state of affairs - a method of persisting data between shutdowns would enhance the utility of Determinator.

The internal structure of the Determinator kernel is not conducive to using a typical file system structure - “files” in Determinator exist in every process’s address space. Since each process has its own virtual address space, different from that of every other process, a file-system paradigm focused on persisting files (that is, single data containers that can be accessed and modified by different processes) will not be the most effective solution. A better paradigm is the snapshot paradigm, where the current running state of the machine is saved onto the persistent storage, allowing the machine to begin executing from exactly that point when it restarts. Determinator’s special process management system (namely, process synchronization through address-space merging) is also an important aspect to take into account during the design of the persistent storage system.
II. Project Goals

The primary goal of this project is to develop a persistent storage mechanism for Determinator. This storage system (I refrain from calling it a file system, because it is not storing files in the same manner as a traditional file system, e.g. ext2 or FAT) will contain serialized versions of the process hierarchy. In essence, it will contain snapshots of the operational state of the system at given points, allowing the user to restart the system and return to nearly the same place in his computations, in case of either intentional or accidental shutdown of the system.

The initial goal, then, is to create a serialization mechanism that is able to save the state of the running machine to disk, and then restore it upon boot. This will require modifying the Determinator kernel to support interaction with persistent storage disks; modifying the Determinator kernel to support serializing data to the disk and loading data from the disk; and implementing a mechanism to regularly serialize the process tree to disk, so that only small amounts of work are lost during a system failure.

Once a working persistence mechanism is created, I will do some more fine-tuning work to improve the performance of the system. Determinator has an interesting method of process synchronization. It gives child processes a copy of their parent’s full address space, and implements copy-on-write in order to avoid copying the entire memory space each time a child is forked. When a child process terminates, its memory space is “diff”ed with its parent’s, and the changes are merged back together. Ideally, the storage system will take this into account and be optimized to allow for fast diffing and merging of its contents. As a second optimization, and considering the prevalence of flash-based storage devices in today’s market, I will attempt to optimize the storage system to use flash-based storage. This is a stretch goal, in that simply creating a functional serialization mechanism will likely be difficult enough without the added trouble of figuring out how best to optimize it for solid state storage.

III. Tentative schedule

This schedule is highly tentative - it is provided as an early way to pace work over the course of the term.

- 10/1 - conceptual design of the Determinator storage system; map of additions and changes that will need to be made to the kernel
• 10/15 - functional snapshotting mechanism

• 10/28 - largely-correct functional snapshotting mechanism with test suite

• 11/4 - automated checkpointing and restoring

IV. Deliverables

The deliverables for this project will be:

• Code base for implementation of persistent storage system in Determinator
  
  – This may take the form of a checkout of a GIT branch; or, if the code is stable enough to be integrated back into the Determinator master branch, a checkout of that branch

• Project report

References