Efficient Synchronization of Linux Memory Regions over a Network: A Comparative Study and Implementation (Notes)

A user-friendly approach to application-agnostic state synchronization

Felicitas Pojtinger (Stuttgart Media University)

2023-08-04

Rough Structure

Rough Structure

- Abstract: A comparative analysis and implementation of various methods for synchronizing Linux memory options over a network
- Introduction
  - Examining Linux's memory management and relevant APIs
  - Use cases for memory region synchronization
- Option 1: Handling page faults in userspace with userfaultfd
  - Introduction to userfaultfd
  - Implementing userfaultfd handlers and registration in Go
  - Transferring sockets between processes
  - Examples of handler and registration interfaces (byte slice, file, S3 object)
  - Performance assessment of this approach
- Option 2: Utilizing mmap for change notifications
  - Concept: mmap a memory region with MMAP_SHARED to track changes in a file
  - Method 1 for detecting file changes: inotify
  - Limitations: mmap does not generate WRITE events

Sections/Research Questions/Ideas Brainstorming

Sections/Research Questions/Ideas Brainstorming

- Usecases: Direct Mount vs. Managed Mount vs. Migration
- Effects of high latency on different pull methods (esp. direct vs. managed)
- Effects of slow local disks or RAM on pull methods
- The asynchronous background push method (for mounts); how chunks are marked as dirty when they are being written to before the download has finished completely
- Mount backend API vs. seeder API
- Preemptive pulls and parallelized startups (n MB saved)
- Background pulling system and interface (rwat), % of availability
- Chunking system/non-aligned reads and writes, checking for correct chunking behavior
- Local vs. remote chunking
- Backend implementations, performance and usecases: File, memory, directory, dudirekta, gRPC, fRPC, Redis, S3, Cassandra

Alternative Outline

Alternative Outline

1. Abstract
    - A comparative analysis and implementation of various methods for synchronizing Linux memory options over a network
2. Introduction
    - 2.1 Background: Examining Linux's memory management and relevant APIs
    - 2.2 Purpose: Use cases for memory region synchronization (Direct Mount, Managed Mount, Migration)
        - Discuss potential effects of high latency, slow local disks or RAM on different pull methods
3. Methodologies
    - 3.1 Option 1: Handling page faults in userspace with userfaultfd
        - 3.1.1 Explanation of userfaultfd and its implementation
        - 3.1.2 Description of userfaultfd handlers and registration in Go
        - 3.1.3 The process of transferring sockets between processes
        - 3.1.4 Examples of handler and registration interfaces
        - 3.1.5 Performance assessment of this approach, with focus on pull methods and effects of system constraints