日曜研究室

技術的な観点から日常を綴ります

[xv6 #57] Chapter 5 – File system – Logging layer

テキストの67ページ

本文

ファイルシステムのデザインにおけるとても興味深い側面は、クラッシュリカバリーである。
この問題は、たくさんのファイルシステムの操作がディスクへの多数の書き込みを伴う事に起因し、その書き込みの一部を残した状態でクラッシュすると、ディスク上のファイルシステムは、矛盾した状態のままになってしまうだろう。
例えば、ディスクに書き込む順番に依存するが、ファイル削除中のクラッシュは、解放されたinodeを指し示すディレクトリエントリや、割り当てらたけど参照されないinodeを残すだろう。
後者は比較的問題が少ないが、解放されたinodeを参照するディレクトリエントリは、再起動後に深刻な問題を引き起こす可能性がある。

xv6は、このようなファイルシステム操作中のクラッシュによる問題を、ロギングの簡易版で解決する。
xv6のシステムコールは、ディスク上のファイルシステムのデータ構造を直接変更しない。
その代わり、ディスク上のログに、行いたいディスクへのすべての書き込みに関する説明を配置する。
一度システムコールがそこで発生した書き込みに対するログを残したら、スペシャルコミットレコード（special commit record）を、完全な操作を含むログを指し示すディスクへ書き込む。
その時点で、そのシステムコールはその書き込みを、ディスク上のファイルシステムのデータ構造へコピーする。
一連の書き込みが終わった後、そのシステムコールはディスク上のログを消す。

システムがクラッシュし再起動せざるを得ない場合、ファイルシステムのコードは、他のプロセスが実行される前に、次の手順でクラッシュから回復する。
ログが完全な操作を含んでいるとマークされている場合、回復のためのコードは、ログに残ってるその書き込みをディスク上のファイルシステム内の本来あるべき場所にコピーする。

ログが完全な操作を含んでいるとマークされていない場合、そのログを無視する。
どちらの場合でも、ログを消去することによって回復処理は完了となる。

ファイルシステム操作中のクラッシュによる問題を、xv6のログがなぜ解決できるのだろうか？
操作が完了する前にクラッシュが発生した場合、ディスク上のログは完了したとはマークされておらず、回復のためのコードはそのログを無視し、そしてディスクの状態はまだ操作が開始されてない状態と同じになるだろう。
操作が完了した後にクラッシュが発生した場合、回復処理はログに残された操作を再現し、おそらく「その書き込みをディスク上のデータ構造へ書き込むための操作が開始された場合」と同じ操作を繰り返す。
どちらの場合でも、ログは、クラッシュに対する配慮をしながら、操作をアトミックにする。
回復処理の後は、一連の操作による書き込みがすべてディスク上に反映されるか、もしくは全く反映されないかのどちらかになる。

感想

ロギングの層の概要です。
ファイルシステムの章に入ってすぐのころはもう少し低機能なものかなと想像してました。
かなりシンプルではありますが、ジャーナリングファイルシステムの一種と言えます。

カテゴリー: 技術 | タグ: xv6 | 投稿日: 2012/3/31 土曜日 [http://peta.okechan.net/blog/archives/1595] |