日曜研究室

技術的な観点から日常を綴ります

[xv6 #58] Chapter 5 – File system – Log design

テキストの67〜68ページ

本文

ログは、ディスクの一番最後の決まった場所にある。
データブロック群に続くひとつのヘッダブロックから成る。
そのヘッダブロックは、セクタ番号の配列を含み、そのセクタ番号はそれぞれログ済みのデータブロックを指す。
ヘッダブロックは、ログ済みのブロックの数も含む。
xv6は、トランザクションがコミットされたとき（前ではない）にそのヘッダブロックに書き込み、ログ済みのブロックをファイルシステムへコピーした後に、そのログ済みのブロック数にゼロをセットする。
これまで述べたように、トランザクション途中のクラッシュの場合、そのヘッダブロックにあるログ済みのブロック数はゼロのままになるだろう。
コミット後のクラッシュの場合、何らかの正の整数が記録されている状態になっているだろう。

どのシステムコールのコードも、アトミックでなければならない一連の書き込みの最初と最後に注意を向ける。
そのような一連の書き込みを、（データベースのトランザクションよりはかなりシンプルだが）トランザクションと呼ぶ。
常に、一度にひとつのシステムコールだけがトランザクションを実行出来る。
他のプロセスは、実行中のトランザクションが完了するまで待たなければならない。
つまり、ログは多くとも一度に一つのトランザクションを持つ。

xv6は、次に述べるような種類の競合（並行なトランザクションが許可されている場合に起こりうる競合）を避けるために、一度に一つのトランザクションのみを許可する。
トランザクションXが、あるinodeへの変更をログに書き込んだと仮定する。
それと平行なトランザクションYは、同じブロックにある別のinodeを読み込み、そのinodeを更新し、そのinodeのブロックをログに書き込み、そしてコミットするとする。
Xはまだコミットしておらず、そしてYによるコミットは「Xが修正したinode」をファイルシステムに

書きこんでしまうので、これは大事故に繋がる。
この問題を解決する洗練された方法がある。
xv6は、並行トランザクションを禁止することによってこの問題を解決している。

xv6は、読み込みだけのシステムコールに、トランザクションを用いて平行に実行することを許可している。
inodeのロックは、読み込みだけのシステムコールに対して、アトミックなトランザクションと同じような効果をもたらす。

xv6は、ログを保持するためにディスク上の限られたスペースを割り当てる。
ログ用のスペース以外に、追加でよりたくさんのブロックを書き込めるようなシステムコールは無い。
これはほとんどのシステムコールにとって問題ではないが、そのうち2つは潜在的に多くのブロックを書き込む可能性がある。
その2つとは、writeとunlinkである。
巨大なファイルを書き込む事は、当然ひとつのinodeブロックは使うが、多くのデータブロックと多くのビットマップブロックを書き込むだろう。
巨大なファイルをunlinkする事は、ひとつのinodeと、多くのビットマップブロックを書き込むかもしれない。
xv6のwriteシステムコールは、巨大な書き込みをログの領域に収まるよう分解する。
xv6のファイルシステムは常に一つのビットマップブロックを使うので、unlinkは問題を引き起こさない。

感想

今回もログの概要です。
前回よりは少しつっこんだ内容です。
次回はログのコードを見ていくようです。

ログの領域サイズは固定で、これは一般的に巨大なファイルを書き込むときと消すときに問題になりうるのですが、消す場合（unlinkの場合）xv6ではビットマップブロックはひとつしかないので問題にならず、書き込む場合（writeの場合）は分割して書き込むよう実装されてるので問題にならないようです。

ビットマップブロックとはなんぞや。
まだその辺のソースを読んでないので完全に想像ですが、1ブロック=512バイト=4kビットであり、1ビットを1セクタに対応づけて4kセクタ分のディスクの利用状況を記録するようなブロックかなと思います。
（この方式だと2MBぐらいしか管理出来ないので、実際の対応付けは1ビット=1セクタじゃないかも）

ビットマップブロックが複数存在し得るシステムでは、巨大ファイルのunlinkによって、ディスクの利用状況が一度に大きく代わり、ビットマップブロックを複数変更しなければならなくなる可能性があり、限られたログの領域に収まらない可能性が出てきます。
xv6fsはビットマップブロックは1つに限られてるので、unlinkでは問題は起きないということですね。

カテゴリー: 技術 l タグ: xv6 l 投稿日: 2012/4/2 月曜日 [http://peta.okechan.net/blog/archives/1602] l