Cerium による並列処理向け I/O の設計と実装

085726C 古波倉正隆 指導教員：河野真治

1 はじめに

1.1 研究背景

近年、CPU 1 コア当たりのクロック数が頭打ちとなっているので、シングルコアでの処理能力はほとんど上がっていない。それを解決した結果、シングルコアからマルチコアへの移行によって CPU 性能が向上している。しかし、マルチコア CPU を最大限に活かすためには、プログラムの並列度を向上させなければならない。そこで当研究室では Cerium Library を提供することによって並列プログラミングを容易にしている。

1.2 研究目的

先行研究による Task の並列化によって、プログラム全体の処理速度は飛躍的に向上しているが [1] 、ファイル読み込み等の I/O と Task が並列で動作するようには実装されていない。ファイル読み込みと Task を並列化させることにより、さらなる処理速度の向上が見込まれる。I/O と Task が並列に動作し、高速かつ容易に記述できるような API を Cerium Library が提供することにより、様々な人が容易に並列プログラミングが記述できるようになるであろうと考えている。

本研究では、 I/O と Task の並列化の設計・実装によって既存の正規表現の処理速度、処理効率を上げることを目指す。

2 Cerium Task Manager

Cerium Task Manager は、並列処理を Task 単位で記述する。関数やサブルーチンを Task として扱い、その Task に対して Input Data、Output Data 及び依存関係を設定する。そして、それに基づいた設定の元で Task に管理し、実行される。本稿で述べる Input Data とは、検索対象となるテキストファイルのことである。

Cerium Task Manager は PlayStation 3/Cell、Mac OS X 及び Linux 上で利用することが可能で、近年では GPU への利用も可能となった。[2]

3 I/O を含む Task の概要

ファイルを読み込んで一定の大きさでファイルを分割し (File Read)、それらに対してそれぞれ文字列検索等の処理 (Run Tasks) を行う。そしてそれぞれの処理から返されたの結果 (Output Data) を最後に集計をして結果を返す (Run resultTask)。(図 1)

図 1

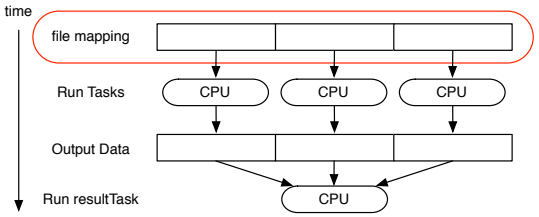

図 1: I/O を含む Task

先行研究では、File Read の部分は mmap 関数を使用して実装していた。mmap 関数での実装の場合はコードの記述が容易である。

4 並列処理向け I/O の設計と実装

4.1 mmap での実装の問題点

mmap でファイルを読み込むタイミングは、mmap 関数が呼ばれたときではなく、mmap した領域に対して何らかのアクセスをしたときに初めてファイルが読み込まれる。つまり、分割された Task は文字列検索をすぐに行うのではなく、文字列検索を行おうとした時に初めてファイルが格納される。Task は複数一斉に実行されることが望ましいが、mmap だとそれぞれの Task で読み込みが起こってしまうので、I/O ネックによる Task の待ちが発生する恐れがある。

4.2 Blocked Read の設計と実装

Blocked Read とは、あるサイズずつで読み込む処理と、それらに文字列検索行う処理を分離させるための実装方法である。この方法では、読み込み専用の Blocked Read と、文字列検索を行う Task Blocks をを別々に生成し処理を行う。Read Task はファイル全体を一度に読み込むのではな

く、ある程度の大きさで分割を行い、読み込みされ次第それぞれの文字列検索が行われる。

Task は 1 つずつ起動すると、起動した Task でメモリを圧迫してしまうため、Task を複数まとめたブロック単位で起動を行う。この 1 つのブロックで処理されるテキストファイルを、Blocked Read で読み込んでいき、読み込みが終わったら読み込まれた範囲の Task Blocks を起動する。もし、Blocked Read で読み込まれる前にその範囲を担当する Task が起動してしまうと、正しい結果が返ってこない。それを防止するために、Task Blocks は必ず Blocked Read が行われてから起動するように wait をかけている。(図 2)

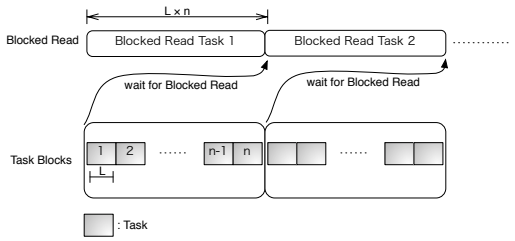

図 2: Wait for Blocked Read

4.3 I/O 専用 thread の実装

Cerium Task Manager では Task 単位で CPU Type の設定を変更することができる。SPE_ANY という CPU Type を設定すると、Cerium Task Manager 側が自動的に CPU を割り振ってくれる。しかし、今回の実装でこの CPU Type を使用してしまうと、Blocked Read Task の隙間時間に Task が割り振られてしまう問題がある。(図 3)

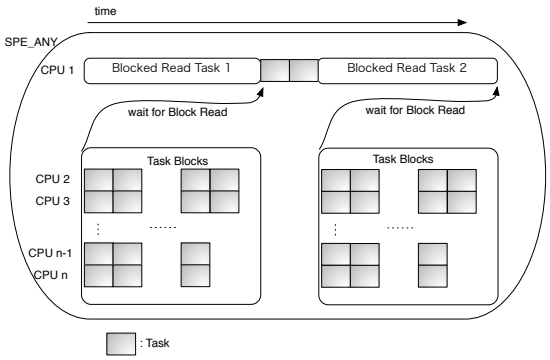

図 3: SPE_ANY での設定時

その問題を解決するために、IO_0 という CPU Type を新しく実装した。IO_0 は他の CPU Type よりも priority を高く設定しているので、他の Task に割り込まれることがないようにした。(図 4)

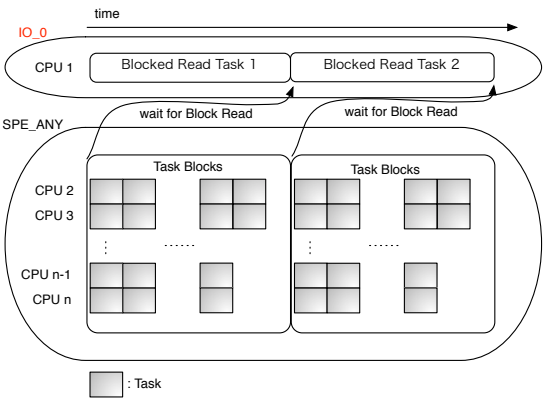

図 4: IO_0 での実装時

5 ベンチマーク

5.1 実験環境

- Mac OS X Mavericks (10.9.1)
- HDD 1TB、Memory 16GB、CPU 2*2.66 GHz 6-Core Intel Xeon
- ファイルサイズ 10GB に対して検索文字列がいくつ含まれるのかカウント

5.2 実験結果

読み込み方法	実行速度 (s)
mmap	XX.XXX
Blocked Read & SPE_ANY	XX.XXX
Blocked Read & IO_0	XX.XXX

表 1: file read の実行結果

6 まとめと今後の課題

参考文献

[1] 金城裕、河野真治、多賀野海人、小林佑亮 (琉球大学)
ゲームフレームワーク Cerium Task Manager の改良
情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS), April 2011

[2] 渡真利 勇飛、河野 真治 (琉球大学)
Cerium Task Manager の GPGPU への対応
情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS)、May 2013