2013 年 10 月 7 日 情報システム基盤学合同輪講発表資料

Scalable Massively Parallel I/O to Task-Local Files

著者： Wolfgang Frings, Fwlix Wolf, Ventsislav Petkov
出典： *Proceedings of the Conference on High Performance Computing Networking(SC'09), pp.1-11, 2009.*
発表者： 高性能コンピューティング学講座 本多・近藤研究室 1353002 大坂隼平

1 はじめに

高性能コンピュータシステムの性能は、年々高くなっている。その背景として、クラスタやスパコンにおけるプロセッサコアの数の増加がある。この多数のコアを用いて並列処理を行うことで高い性能を得ることができるが、この際には並列に I/O を行うことの重要度が高い。しかし、並列アプリケーションおいて、checkpoint や、パフォーマンスデータなど各タスクがそれぞれ出力するファイルである task-local file を同一のディレクトリに大量に生成するとそれを管理するメタデータサーバが競合を起こしたり、並列ファイルシステムが不安定化してしまう。このため I/O の性能が頭打ちとなり、ファイル生成に時間がかかってしまうという問題がある。

本論文では、メタデータサーバの負荷を低減させるために、一つの物理ファイル上に仮想的に複数の task-local file のマッピングを行う層を設ける SIONlib を提案する。また、その効果を調査するために 2 つのアプリケーションに組み込むことにより評価実験を行う。

2 SIONlib

2.1 SIONlib の構成

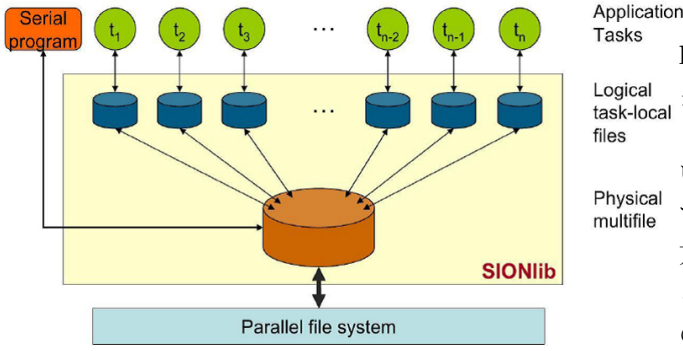

図 1: SIONlib の構成

並列ファイルシステムの最大の長所は、1 つのファイルに効率的に同時アクセスを行うことが可能であるという点である。しかし、ファイルシステムの制限上、多くのアプリケーションは同時にかつ大量に task-local file を生成するとディレクトリの管理が複雑になり、ファイル生成に必要以上に時間がかかっていた。

この問題の解決手法として図 1 に SIONlib の構成を示す。SIONlib は並列アプリケーションと、並列ファイルシステムの中間に 1 つの巨大なファイルを作り、それを仮想的に複数のファイルに見せることによってメタデータサーバの負荷の低減を狙うものである。

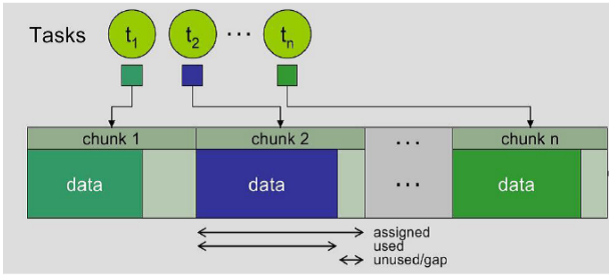

図 2: 物理ファイルの統合

SIONlib の実現方法を図 2 に示す。multifile と呼ばれる 1 つの物理ファイルをチャンクと呼ばれる区画に分割し、そのチャンク毎に各タスクの task-local file を配置する。チャンクの開始位置を把握するために metadata ブロックを作成し、チャンクの開始位置情報を格納する。これにより各タスクが task-local file を同時にオープンしようとした場合でも、1 回のメタデータサーバへのアクセスで済み、ファイルシステムの安定化に繋がる。

ここで、multifile の作成にはあらかじめ各タスクの task-local file の合計サイズを指定する必要がある。しかし、各タスクが multifile に書き込みを行う合計のサイズを知ることは一般的には難しく、各タスクが書き込む量が少ない場合でも multifile のチャンクサイズを大きくとらなくてはならない問題が発生する。そこで、ブロック化を行う方式を採用する。各タスクの書き込む合計のサイズを知る方法は難しいが、各タスクが一回の書き込みを行う際の最大サイズは容易に想定できるため、そのサイズをチャンクサイズとし、二回目以降の書き込みでチャンクサイズを超えるようであれば、新しいブロックを作成し、そこにデータを配置する。これにより、各タスクが書き込む合計量が少ない場合でも、multifile のサイズを大きくする必要がなくなる。

なお、アプリケーションとディスク間の並列性を利用するためにブロック毎に分割を行いつつユーザが指定した数だけの複数の multifile を生成することも可能である。

2.2 ソースコードの変更

```
sid=sion_paropen_mpi();          集団的にopenを行う

if(!sion_feof()){
 sion_ensure_free_space();       現在のチャンクサイズよりも大きいサイズを書き込みたい場合、新しいチャンクが配分される
 fwrite();                       データの書き込み
                                 (また、sion_fwrite();で上記の2行を1行で書くことも可能)
}
sion_parclose_mpi();             集団的にcloseを行う
```

図 3: Parallel write モード

SIONlib においてアプリケーションを multifile へアクセスさせるためには、ファイルのオープン、クローズ用のコードを変える必要がある。例として、Parallel write モードを図 3 に示す。書き込みを行う際 sion_ensure_free_space() を追加し、コードを書き変えることで、実装可能である。

3 定量評価

3.1 評価環境

評価には 2 つのベンチマーク Juelich Blue Gene(Jugene) [1] と Jaguar[2] を用いて行った。Jugene はドイツのユーリッヒ研究センターに設置された計算機で、ノード数 1024、コアの合計は 294,912 である。一方 Jaguar はアメリカのオークリッジ研究所に設置された計算機で、コアの合計は 31,328 である。

3.2 オーバーヘッドの評価結果

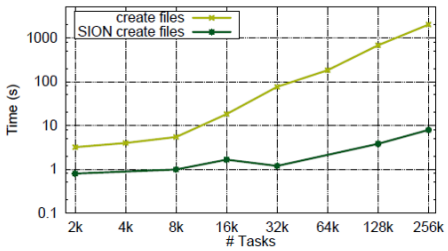

図 4: Jugene におけるファイル生成時間

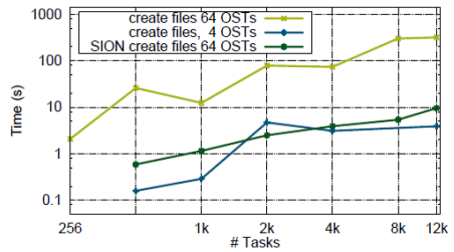

図 5: Jaguar におけるファイル生成時間

SIONlib を適用したことにより、ファイル生成によるオーバーヘッドのが改善されたか、検証を行った。

図 4、図 5 に、それぞれ Jugene、Jaguar を用いて task-local file の数と、実行時間の関係を表したグラフを示す。実行時間は、最初にファイルオープンを行ってから、クローズするまでの時間をとった。SIONlib 適用前と適用後で比較を行った結果、両方の計算機で、SIONlib を適用したことで、オーバーヘッドが改善されたことが判明した。

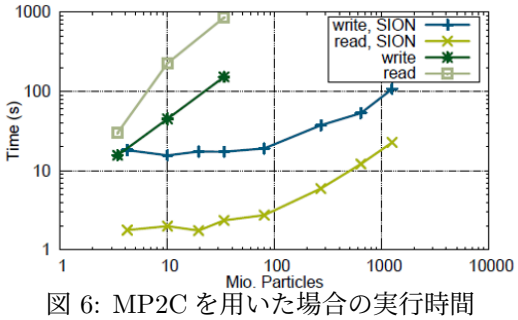

図 6: MP2C を用いた場合の実行時間

表 1: Scalasca を用いた場合の測定結果

I/O type	#tasks	trace size	activation	write BW
Task-local	65536	1105 GB	253.5 s	3476 MB/s
SIONlib	65536	1105 GB	25.9 s	3589 MB/s

また、multifile 生成によるバンド幅の影響もほとんど受けないことが分かっている。

4 事例研究

SIONlib の効果を示すため、実際に 2 つのアプリケーションに SIONlib を組み込み、使用した場合と使用しなかった場合の測定をそれぞれ行った。

MP2C

MP2C は流体力学の粒子動的シミュレータである。図 6 に MP2C における Jugene 上で 1,000 コアを用いて書き込み、読み取りを行った場合の SIONlib 使用前後の実行時間の比較をグラフで表している。実験結果より、SIONlib を組み込んだことで実行時間の短縮、より大きな粒子数の測定が可能となった。

Scalasca

Scalasca は、並列アプリケーション用のプロファイリングツールである。表 1 に Jugene 上で 50 行にわたる C で書かれたコードのトレーシングを行った結果を示す。activation における時間が SIONlib を使用したことによって、大幅に短くなっている。これは、待ち状態が発生している箇所が減少したためである。

5 まとめ

本研究では、SIONlib を実装することによって task-local file を大量に生成する際に発生するメタデータサーバの競合や、並列ファイルシステムの不安定化の改善を行った。評価結果は、実験を行った全ての計算機に対して、オーバーヘッドが改善され、SIONlib を組込んだアプリケーションにおいても、性能向上がみられた。

参考文献

[1] Juelich Supercomputing Center. JUGENE. http://www.fz-juelich.de/jsc/jugene.

[2] Oak Ridge National Laboratory. Jaguar. http://www.nccs.gov/computing-resources/jaguar/.