並列ファイルシステムにおける ハードディスクの簡易的な温度トレース機能の実装

六車　英峰†，辻田　祐一†
近畿大学工学部電子情報工学科†

1. はじめに

PC クラスタにおける並列ファイルシステムを構築する際には，一般に入出力性能が重視されるが，クラスタの安定稼動の為には，特に入出力プログラムが動いている際の各ノードのハードディスク（以下 HDD）の温度を監視する必要があると我々は考えている．しかしながら，ノード毎に温度センサを設置するのは，費用や手間を要する．そこで我々は，並列ファイルシステム側の情報を MPI プログラムと同時にトレースする機能を持った PIOviz [1]に対して，HDD の S.M.A.R.T.情報をトレースする機能を追加することにより，並列ファイルシステムの振舞いと HDD の温度情報を同時にトレースする簡易的な機能の実装を行い，その動作検証を行った．

2. MPI プログラムのトレース機能

MPI プログラムの処理性能はプロセス同士の通信に大きく影響される．そのため，MPI プログラムの最適化には内部の動作をトレースする必要がある．よく知られた MPI プログラムのトレースツールに MPE2 [2]や Vampir [3]などがある．MPE2 は MPI ライブラリの一つである MPICH2 [4]で提供されているライブラリである．MPE2 によって得られたトレース情報は CLOG2 形式のフォーマットで出力され，更に，SLOG2 形式に変換した後に Jumpshot によって可視化することが出来る．Vampir は豊富な機能を持つ商用のトレースツールであり Open Trace Format [5]に準拠したデータフォーマットもサポートしている．このような一般的なトレースツールは MPI プログラムの動作をトレースすることは出来るがファイルシステム側の振舞いまでトレースすることはできない．我々が用いた PIOviz は，並列ファイルシステムの一つである PVFS2 [6]と MPICH2 を元に開発されたもので，MPI プログラムだけでなく PVFS2 ファイルシステムの動作や状態もトレース出来る．

Implementation of a simple temperature tracing mechanism for hard disk drives of a parallel file system
Hidetaka MUGURUMA, Yuichi TSUJITA
Department of Electronic Engineering & Computer Science, Faculty of Engineering, Kinki University

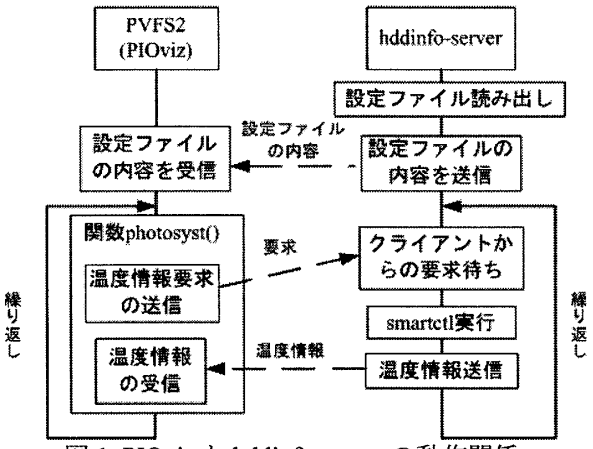

図 1. PIOviz と hddinfo-server の動作関係

3. 実装

本実装では，PIOviz の PVFS2 サーバとは独立に動くサーバプログラム hddinfo-server を介して温度情報を取得するようにした．図 1 に PIOviz と hddinfo-server の動作の関係を示す．

3.1 hddinfo-server の機能

hddinfo-server は起動時に，設定ファイルから smartctl コマンド[7]に渡すデバイス名などの引数を読み出す．その後設定ファイルの内容を PVFS2 サーバ側に送信する．hddinfo-server は PVFS2 サーバから温度情報を要求されるたびに smartctl コマンドを実行して S.M.A.R.T.情報の温度情報を取得して要求元へ返す．設定ファイルにデバイス名が複数行ある場合には，それら全ての温度情報を取得して PVFS2 サーバへ返す．

3.2 PIOviz 側の温度トレース機能の実装

PIOviz の PVFS2 サーバはデフォルトで 1 秒おきに関数 photosyst()を呼び出して PVFS2 サーバの状態のトレースを行うが，hddinfo-server への温度情報の取得要求は 10 秒に 1 回にした．毎秒行わないようにしたのは，smartctl コマンドが HDD に対してアクセスを行うことによる入出力性能への影響を抑えるためと，温度の変化は緩やかであるため頻繁に取得する必要が無いためである．HDD の温度情報が複数ある場合，その中の最も高い温度の値を自動的に選択してトレースファイルに記録するようにした．

3.3 PIOviz と hddinfo-server の通信

smartctl コマンド実行に要する時間は，我々の環境で計測したところ，他のプロセスによる HDD へのアクセスが無い状態で，約 220 ミリ秒であった．そのため，10 秒に 1 回の割合で，hddinfo-server からの応答を待つための遅延が生じ，トレース間隔が不均一になる問題があった．そこで我々は，hddinfo-server からの温度情報の受信を，要求を送った次のトレース実行で行うようにした．これにより， PVFS2 サーバによるトレース実行の合間に smartctl コマンドが実行されるので，他のプロセスによる HDD へのアクセスが無い場合には，遅延時間は通信に要する 3 ミリ秒程度に短縮出来ている．

4. 動作検証

本実装の動作検証を PVFS2 に付属の MPI-IO-TEST ベンチマークを用いて行った．このベンチマークで，入出力するデータサイズと書き込み回数を指定して，集団型 MPI-IO 関数で入出力を行い PVFS2 サーバに対して負荷をかけ，それに伴う温度上昇がトレースできるかを検証した．

4.1 実行環境

動作検証には 9 ノードで構成される PC クラスタを使用した．このクラスタの各ノードの仕様を表 1 に示す．ヘッドノードを含む 5 ノードで PIOviz の PVFS2 ファイルシステムを構築し，ヘッドノードをメタデータノード，残りの 4 ノードをデータノードにした．そして，その他の 4 ノードをクライアントプログラムを実行するクライアントノードとした．MPI-IO-TEST では，各プロセスが PVFS2 ファイルシステムに対して，40 MB のデータを 250 回書き込むように設定した．

表 1. PC クラスタの各ノードの仕様

CPU	Intel Pentium D 930 (3.2 GHz)
メモリ	DDR2-SDRAM 1.5 GB
HDD	ヘッドノード Western Digital WD1600JS-75N
	ヘッドノード以外 Seagate ST3160811AS
OS	CentOS4.4 Linux kernel2.6.9-42ELsmp

4.2 実行結果

トレース情報を Jumpshot で表示したものを図 2 に示す．説明のため，温度とクライアントプロセスの情報のみ表示している．上から 4 つのライン上にあるのがクライアントプロセスの動きである．その下にある 5 つの帯状のものは，一番上から順にメタデータノードと 4 つのデータノードの温度情報を示している．図中に書き込まれている数字は取得した HDD の摂氏温度である．各データノードで，ベンチマーク開始から約 120 秒程度経過後，温度が上昇し，終了してから約 120 秒程度経過後に温度が下がり始めていることがわかる．一方，ヘッドノードはベンチマークを開始する前から温度が上昇し始め，ベンチマーク終了後も下がっていないことがわかる．

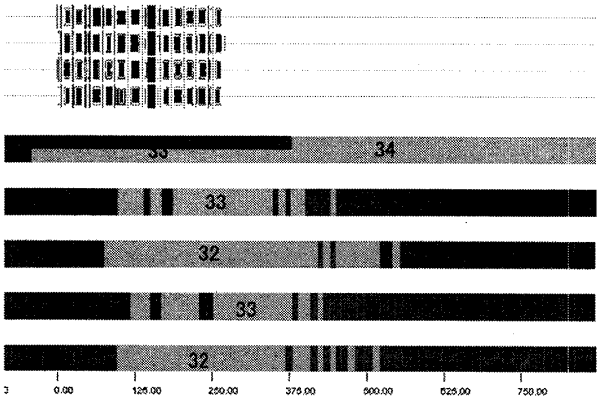

図 2. トレース情報の可視化の例

5. まとめ

今回の実装で，S.M.A.R.T.情報の温度情報がトレース出来ていることを，動作検証を行うことで確認できた．これにより，MPI プログラムによる入出力と HDD の温度の変化の関連性を調査する環境が構築できた．今回の実装では，まれに入出力の影響を受けて smartctl コマンドの実行に 1 秒以上の遅延が生じ，トレース情報を取得する間隔が不規則になることがあった．今後，この問題の改善に取り組むと共に，入出力と HDD の温度の変化の関連性について調査を行う予定である．

謝辞

PIOviz を提供して頂くと共に大変有用なアドバイスを頂いたドイツ気候計算センター並びにハンブルク大学の Thomas Ludwig 教授と Julian Kunkel 氏に感謝致します．

参考文献

[1] T. Ludwig, S. Krempel, M. Kuhn, J. M. Kunkel, and C. Lohse, “Analysis of the MPI-IO optimization levels with the PIOviz jumpshot enhancement,” LNCS 4757, pp.213-222, Springer, 2007
[2] A. Chan, W. Gropp, and E. Lusk, “An efficient format for nearly constant-time access to arbitary time intervals in large trace files,” *Scientific Programming*, vol.16, no.2-3, pp.155-165, 2008.
[3] Vampir. http://www.vampir.eu/
[4] MPICH2. http://www.mcs.anl.gov/research/projects/mpich2/
[5] A. Knüpfer, R. Brendel, H. Brunst, H. Mix, and W. E. Nagel, ”Intoroducing the open trace format (OTF),” LNCS 3992, pp. 526-533, Springer, 2006.
[6] PVFS2. http://www.pvfs.org/pvfs2/
[7] Smartmontools. http://sourceforge.net/apps/trac/smartmontools/