4A-1

Linux カーネルにおける性能情報取得機構

松沢　敬一†　揚妻　匡邦†　中野　隆裕†　岩嵜　正明†

（株）日立製作所　システム開発研究所†

1　はじめに

今日，エンタープライズ用途で Linux の利用が増えており，Linux サーバの一層の高性能化が求められている．近年，CPU の高性能化手法は単体性能の向上からマルチプロセッサ，マルチコア化による並列化へと変化している．しかし Linux は CPU の性能情報として，精度が粗い CPU 利用率しか提供しない．CPU 並列化の影響を大きく受けるマルチスレッド環境において性能をより正確に把握し，高性能化を行うには，スレッド毎での正確な利用率測定に加え，キャッシュミス率や CPI(Clock Per Instruction)値の測定が必要である．そこで我々は，高精度の CPU 利用率に加え，L2 キャッシュミス率（以後キャッシュミス率），CPI 値をスレッド毎に測定する機構を x86 用 Linux2.6 カーネルに組み込んだ．

2　従来方式

Linux2.6 は，1ms 毎のタイマ割込み時に実行中のスレッドが，その 1ms 間 CPU を占有したとして CPU 利用率を算出する．しかしこの手法では，多数のスレッドが 1ms 以下の間隔で頻繁にタスクスイッチを行いながら動作するサーバでは十分な精度の CPU 利用率を得られない．例えば 4 つのスレッド A～D が動作している**図 1**の例では，明らかに実行時間が異なる A，B，D の CPU 利用率が等しく 33%で，C の利用率が 0%と判断されてしまう．この手法では CPU 利用率を正しく求めることができない．

他の問題点として，割込み処理中にタイマ割込みが発生した場合，元の割込み発生時のスレッドが 1ms 間 CPU を利用したと判断される点がある．例えば，**図 1**ではスレッド A 実行中の一部は他の割込み処理を行っているにも関わらず，その処理時間を考慮せず，CPU 利用率は 33%となる．後に示すように IO や NIC による割込みが多く発生するサーバ環境では，この要因による CPU 利用率の精度への影響は大きい．

一方 1ms以下の間隔で割込みを起こし，かつ他の割込みを考慮して精度の高い性能情報を得るツールにはOprofile[2] がある．このツールは，割込み頻度を増すためツール自体のオーバヘッドが大きくなる問題がある．

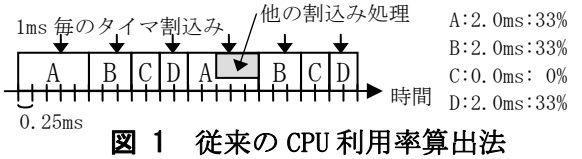

図 1　従来の CPU 利用率算出法

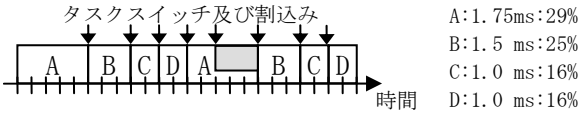

図 2　提案する CPU 利用率算出法

利用率以外のCPU性能情報を得る手法として，Intel CPUの性能モニタリングカウンタを用いるPerfctr[3] やRabbit[4] がある．これらはシステム全体のキャッシュミス率やCPI値を得ることができるが，スレッド毎の情報取得は行えない．

3　提案手法

我々は低オーバヘッド，高精度で，CPU 利用率とキャッシュミス率，CPI 値測定を可能にするため，以下の手法を提案し，x86_64 用 Linux カーネル 2.6.12 に実装した．

3.1 高精度な CPU 利用率算出

我々の手法はカーネル空間，ユーザ空間を問わず全スレッドの CPU 利用率を正確に測定する汎用的な機構を提供する．本機構ではカーネルの改造により，各スレッドが個別に性能測定の機構を導入することなく CPU 利用率を取得できる．また，CPU 動作時間をクロック単位でカウントする TSC(Time Stamp Counter)を用いた粒度の細かい CPU 利用率を取得できる．

Linux はスレッドを task_struct 構造体で管理する．この構造体に前回割込み及びタスクスイッチ時の TSC 値と累計 TSC 値のフィールドを追加した．割込みやタスクスイッチが発生した時，カーネルはその時点の TSC 値と前回発生時の TSC 値の差分をとり，スレッド毎の実行時間として累計 TSC 値に加算する（**図 2**）．この手法では，実際にスレッドが動作した時間を測定するため正しい実行時間を取得できる．

3.2 キャッシュミス率及び CPI 値の測定

利用率以外にキャッシュミス率やCPI値を採取するため，Intelのプロセッサが持つ性能モニタリング機構[1] を利用する．この機構はL2 キャッシ

A Performance Monitoring Facility for Linux Kernel
† Systems Development Laboratory, Hitachi, Ltd.

%CPU	%nCPU	%irqCPU	TSC	L2MsR	CPI	COMMAND
70.12	59.591	7.723	1698319658	0.009	0.842	perl
25.68	24.894	1.018	709640148	0.282	5.695	dd
12.84	14.050	0.015	398686435	0.016	18.775	kblockd
5.93	4.901	1.196	139099415	1.286	6.035	xfslogd
4.94	2.305	0.462	65665110	1.775	4.970	nfsd
3.95	2.396	0.500	68210772	1.763	4.926	nfsd
3.95	3.540	0.741	100762417	1.735	4.921	nfsd
2.96	2.368	0.288	67428760	1.765	5.081	nfsd
2.96	1.693	0.298	48217757	1.737	4.881	nfsd
1.98	1.849	0.174	52468087	0.910	4.185	kswapd0
1.98	1.414	0.308	40243462	1.787	5.001	nfsd
1.98	2.229	0.433	63470974	1.710	4.940	nfsd
0.99	0.932	0.150	26527893	1.710	5.035	nfsd
0.99	1.314	0.257	37403219	1.708	4.884	nfsd

図 3　提案手法を適用した top コマンド

ュミスや実行命令をカウントする性能カウンタを提供する．この値をTSC値同様に割込み及びタスクスイッチ時にスレッド毎 に累計する．累計TSC値を実行命令数で割ることでCPI値を，L2 キャッシュミス回数を累計TSC値で割ることでキャッシュミス率を算出する．

3.3 性能情報の提供方法

カーネルが持つ TSC 値及び性能カウンタ値の情報をユーザ空間に提供するため，本実装では procfs を用いた．procfs ではカーネル内の情報をテキストファイルの形で提供できるため，利用者による取得，加工が容易と言う利点がある．

4　実行結果

4.1 キャッシュミス率及び CPI 値の測定

多数のスレッドが頻繁にタスクスイッチを行う例として NFS サーバの実行結果を示す．NFS サーバは複数の NFSD スレッドが IO 処理を行うため，タスクスイッチ間隔が数十μs 程度と非常に短く，既存のツールでは正確な CPU 利用率が測定できない．ここでは NFSD との動作の違いを見るためシングルスレッドで動作する perl や dd を同時に実行し，2CPU 環境で測定した．

図 3は本機構により得た情報を表示するよう改造した top コマンドの出力結果である．1 行が 1 つのスレッドを表し，各項目は左から従来手法で算出した CPU 利用率，提案手法で算出した CPU 利用率，各スレッド実行中の割込み処理による CPU 利用率，クロック単位の実行時間，1000 倍にスケーリングしたキャッシュミス率，CPI 値，コマンド名を示す．いずれも 1 秒間のカウント数が表示される．網掛け部からわかるように，従来手法による CPU 利用率が離散的であることに比べ，提案手法ではより細かい値が測定できた．提案方式では割込みの影響も考慮した CPU 利用率が取得できており，従来手法で求めた CPU 利用率にはかなりの誤差があったこともわかった．

4.2　キャッシュミス率及び CPI 値の測定

表 1　CPU 数による性能情報の変化

CPU Num	Avg CPU Load	Avg L2Miss Ratio	Avg CPI	Total Inst-ructions
2	27.188%	0.821%	4.62	329,820K
1	45.948%	0.691%	4.00	321,317K

CPU構成の影響を調査する例として，CPU数が 1 と 2 の場合について性能測定を行った．結果は**表 1**の通りである．表の各列は左からCPU数，平均CPU利用率，命令あたり平均L2 キャッシュミス率，平均CPI値，合計実行命令数を示している．一般にCPUの数が増えた場合，ロックや資源の競合が発生するため性能は線形には上がらない[4]．実際この表では， 1CPUと 2CPUを比較すると実行命令数は 2.6%増加にとどまっているが， CPI値は 16%増加しており，実行効率が低下していることがわかる．また，実行命令数に対するキャッシュミス率もが 19%増えており，実行効率低下の原因の一つにキャッシュミス率の増加があることがわかる．

5　おわりに

本論文では，x86 用 Linux のための高精度な CPU 利用率，キャッシュミス率，CPI 値の測定機構を提案し，実装した．本機構では割込み時及びタスクスイッチ時にスレッド毎の TSC 値及び性能カウンタ値累計を行うことで低オーバヘッド及び高精度な情報取得を可能とした．

また，本機構を実装したカーネル上で NFS サーバを動作させ，実際に測定を行った．測定結果から，我々の手法が従来手法に比べ高い精度で CPU 利用率を測定でき，加えてキャッシュミス率や CPI 値を測定できることを確認した．本手法で得られた結果から，サーバで動作させるプログラムの特性に応じて，CPU 数が性能に与える影響を調べることができた．

参考文献

1) Intel: The IA-32 Intel Architecture Software Developer’s Manual Volume 3, System programming Guide, Number 256668, 2005.
2) Oprofile - A System Profiler for Linux: http://oprofile.sourceforge.net/news/
3) perfctr: http://user.it.uu.se/~mikpe/linux/perfctr/
4) Rabbit: http://www.scl.ameslab.gov/Projects/Rabbit/
5) 岩嵜　正明, 高本　良史, 吉住　誠一: 密結合マルチプロセッサにおけるオーバヘッドの解析的一評価手法, 情報処理学会論文誌, 第 31 巻, 第 11 号, pp.1627-1635, 1990.