5J-4

Java仮想マシンにおけるスレッド優先度の制御方式

篠本 一昌† 芝 公仁† 岡田 至弘†
†龍谷大学理工学部

1 はじめに

プログラムの実行環境を独自に構築する仮想マシンでは，プログラミング言語のレベルからスレッドをサポートする．マルチスレッドに関しては，マルチプロセッサを活用できるなどの利点から，オペレーティングシステムレベルでのスレッドを使用して仮想マシン上のスレッドを実現する場合が多くなっている．仮想マシン上にスレッドを実現する場合，仮想マシンには，オペレーティングシステムの持つスレッド管理機能をアプリケーションから有効に活用できるような仕組みを持つことが求められる．

最も利用されている仮想マシンの一つにJava[1]がある．本稿では，まず，Java仮想マシンの実装のひとつであるHotSpotにおいて，Javaスレッドの優先度がオペレーティングシステムのスケジューリングに反映されない場合があることについて述べる．またLinuxが持つ高度なスレッド管理機能をJava仮想マシン上でも活用できるようにするHotSpotの拡張についても述べる．本拡張によってJavaスレッドの優先度をオペレーティングシステムが行うスケジューリングに積極的に反映させることができるようになる．

2 Java仮想マシンにおけるスレッド

本章では，Java仮想マシンの評価に使用したHotSpotでのJavaスレッドの管理について述べる．本研究では，HotSpotの評価にオープンソースとして公開されているOpenJDKを使用している．

HotSpotは，他の多くのJava仮想マシン同様，オペレーティングシステムが提供するネイティブスレッド作成し，これにJavaスレッドを割り当てることでJavaスレッドを実行する．HotSpotでは，スレッドの優先度の扱いをThreadPriorityPolicyによって起動時に設定することができる．ThreadPriorityPolicyは，Javaスレッドの優先度とネイティブスレッドの優先度の割り当て方法について指定するものである．0の場合，Javaスレッドに対応するネイティブスレッドの優先度を変更しない．ThreadPriorityPolicyが1の場合，setpriorityシステムコールを用いてネイティブスレッドの優先度を対応するJavaスレッドの優先度に合わせて設定する．しかし，setpriorityシステムコールによる優先度の変更では，特権を持たないスレッドは優先度を下げることしかできないため，特権を持たない場合はThreadPriorityPolicyを0として動作する．

また，特権を持たせて起動した場合，ThreadPriorityPolicyを1とする設定は有効になるが，必ずしも優先度の高いJavaスレッドが優先されるとは限らない．setpriorityシステムコールで設定される優先度はnice値であり，スケジューリングにおいては，CPUの割り当て時間に差がつくのみである．

3 スレッド優先度の制御

Linuxでは，各々のスレッドに静的な優先度sched_priorityが対応付けられている．sched_priorityは0から99の値をとり，値が大きい方が優先順位が高い．スケジューリングにおいては，実行可能スレッドの中で最もsched_priorityの高いスレッドから実行される．通常のスレッドはsched_priorityが0であり，sched_priorityが1以上の実行可能なスレッドがない場合のみ，sched_priorityが0のスレッドの中だけで決定される動的な優先度に基づいて，実行されるスレッドが決められる．動的な優先度はスレッドのnice値を基に算出され，実行待ち時間によって再計算される．

以上のことを考慮し，本研究では，Linuxの機能のより有効な活用を目指し，次の2つの拡張を行った．

- 特権を持たない場合でも，Javaスレッド間の優劣を有効にする．
- Javaスレッドに優先度に応じてsched_priorityの値を変更する．

特権を持たない場合でも，setpriorityシステムコールを用いて優先度を下げることは可能である．しかし，HotSpotでは，特権を持たない場合，優先度を下げることを行わず，ネイティブスレッドの優先度を変更しない．この動作を変更し，低い優先度のJavaスレッドに対応するネイティブスレッドの優先度を下げるようにした．また，通常ではJavaスレッドの優先度Jに対し，ネイティブスレッドの優先度Nとして対応付けられているものを，表1のAのように変更することで，特権を持たない場合でも，Javaスレッドの優先度が有効に機能するようにした．Javaスレッドの優先度は1から10の値をとることができ，表1では:の左側がnice値，右側がsched_priorityを表す．ただし，この変更は，HotSpot上で動作するスレッドの優先度が全体的に低くなるため，スレッドの動作自体は相対的に鈍くなることが考えられる．そのため，特権の有無によらず，スレッド間の優劣をつけたい場合に有効に作用する．

Thread priority control method in the Java Virtual Machine

Kazumasa Shinomoto†, Masahito Shiba†and Yoshihiro Okada†
†Faculty of Science and Technology, Ryukoku University

表 1　HotSpot における優先度の対応

J	N	A	B
1	4:0	9:0	9:0
2	3:0	8:0	6:0
3	2:0	7:0	3:0
4	1:0	6:0	0:0
5	0:0	5:0	-3:0
6	-1:0	4:0	-6:0
7	-2:0	3:0	-9:0
8	-3:0	2:0	0:1
9	-4:0	1:0	0:2
10	-5:0	0:0	0:3

グラフィックツールなどで，compute-bound な処理をしている場合でもユーザ操作に対する応答性を確保したいときなど，高い優先度の Java スレッドを必ず優先して動作させたい場合がある．これを行うために，Java スレッドに優先度に応じて sched_priority の値を変更するよう HotSpot のスレッド管理部を拡張した．ただし，sched_priority は他のスレッドの動作に強い影響を及ぼすため，通常のスレッドと両立させる形で実装を行った．この拡張における優先度の対応を表 1 の B に示す．

スケジューリングに関する変更は既存のアプリケーションへの影響が大きいため，以上の拡張は，既存の機能の置き換えではなく，ThreadPriorityPolicy の一つとして追加する形で行った．A に示す優先度の対応は，ThreadPriorityPolicy を 2 とした場合に有効になる．B に示す優先度の対応は，ThreadPriorityPolicy を 3 とした場合に有効になる．

4　性能評価

ThreadPriorityPolicy を 2 とした場合の Java スレッドの動作を確認するため，以下の処理を行うスレッドを複数動作させ評価を行った．

(1) 初期値が 0 であるカウンタを 1 ずつ増加させる．
(2) カウンタが 7000 万を越えると処理を終了する．

本実験では，5 個の Java スレッド，T1，T2，T3，T4，T5 を動作させた．このとき，各 Java スレッドの優先度はそれぞれ，1，3，5，7，9 である．通常の HotSpot で動作させた場合のカウンタの値の変化を図 1 に示す．通常の HotSpot で動作させた場合，全てのスレッドにおいて同様にカウンタの値が変化している．このことから，Java スレッドの優先度は，Java スレッドの動作に影響していないことが分かる．

ThreadPriorityPolicy を 2 として動作させた場合のカウンタの値の変化を図 2 に示す．この場合，各スレッドのカウンタの値の増加に差が現れている．特に，スレッド T5 は，最も優先して CPU が割り当てられるため，700ms の時点でカウンタの値が 7000 万を越え，終了している．この後は，スレッド T4 が優先して実行され，1000ms の時点ではカウンタの値が 7000 万になっている．このことから Java スレッドの優先度が有効に機能していることが分かる．

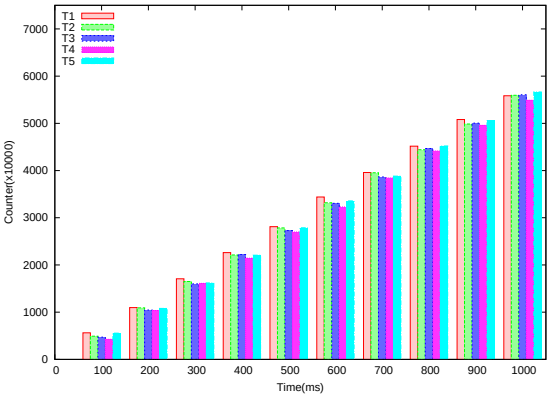

図 1　通常の HotSpot で動作させた場合の Java スレッドのカウンタの値の変化

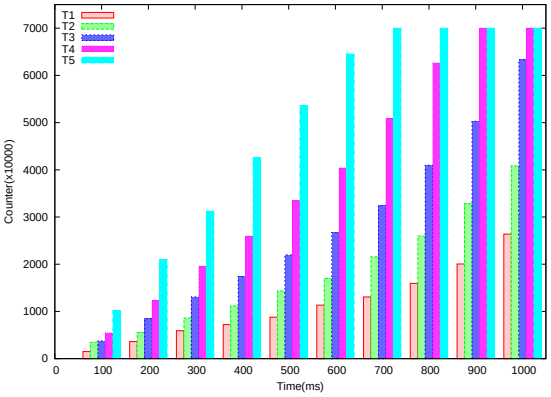

図 2　ThreadPriorityPolicy を 2 として動作させた場合の Java スレッドのカウンタの値の変化

5　おわりに

本稿では，Java 仮想マシンの実装の一つである HotSpot において，Java スレッドの優先度がオペレーティングシステムのスケジューリングに反映されない場合があることについて述べた．また，Linux が持つ高度なスレッド管理機能を Java 仮想マシン上でも活用できるようにする HotSpot の拡張についても述べた．本拡張により，Java スレッドの優先度をオペレーティングシステムが行うスケジューリングに積極的に反映させることができるようになる．

参考文献

[1] Lindholm, Tim and Yellin, Frank : Java Virtual Machine Specification, Addison-Wesley Longman Publishing Co., Inc., (1999)