4A-2

Linux スレッドライブラリ NPTL の評価

荒木 裕[†] 熊谷 宏樹[†] 上床 克樹[†] 佐久間 毅[†]

株式会社 東芝[†]

1. 序論

豊富なミドルウェア群やオープンソースを特徴として、Linux はデジタル情報機器などの組込み機器向けの OS として注目されている。本稿では、Linux における中核技術のひとつであるスレッドライブラリに着目し、情報家電等の組込み機器で広く利用されているアーキテクチャの CPU を用いて、従来技術である LinuxThreads（以後 LT） と、現在も開発が進められている Native POSIX Thread Library (以後 NPTL)[1] について性能比較を実施し、その評価結果を報告する。また、上記 CPU と x86 系 CPU とを比較して、アーキテクチャ依存部による相違についても考察する。

2. Linux スレッドライブラリ

現在、Linux の標準スレッドライブラリとしては、NPTL が定着しており、現在も開発が続いている。ここでは、NPTL と、従来のスレッドライブラリである LT からの変更点及び、スレッドライブラリのアーキテクチャ依存部に関する特徴を述べる。

2.1 Native POSIX Thread Library (NPTL)

LT からの変更点として、LT では、スレッドの同期プリミティブとしてシグナル機構を用いており、速度を犠牲にすると共に、管理スレッドを必要とし、スレッド制御のオーバヘッドなどが問題とされている。NPTL では、以下の変更などにより、問題の改善を目指している。

- 管理スレッドの廃止
- シグナル機構の廃止
- Futex (Fast mutex)による同期プリミティブ採用

また、これらの変更を通して POSIX 互換性の向上も目指している。

2.2 アーキテクチャ依存部におけるスレッドライブラリの特徴

現在、NPTL や LT では、スレッド情報の高速な取り扱いのために、TLS(Thread Local Storage) を標準サポートしている。TLS はスレッドレジスタを必要とするため、このレジスタの Hardware によるサポートの有無が、スレッドライブラリでのスレッド情報取得に影響を与える。例えば、MIPS アーキテクチャ[2]にはスレッドレジスタが Hardware でサポートされているものはまだ無いため、カーネルによってエミュレートされている。この処理がボトルネックとなり、スレッドレジスタを Hardware でサポートする x86 などのアーキテクチャに比べ、MIPS での TLS 処理のオーバヘッドが大きくなっている。最近、この問題を解決すべく、エミュレート処理の改善や、コンパイラによる命令順序組換えによる改善が試みられている[3][4]。

Evaluation of the Linux thread library NPTL
†Embedded System Core Technology
Development Department, CORE TECHNOLOGY CENTER,
TOSHIBA CORPORATION

3. 評価

現状の NPTL 評価のために、処理時間を評価指標として、LT との比較評価を行う。また、対象のアーキテクチャとして、スレッドレジスタのサポート状況が異なる x86 と MIPS を用いて比較検討を行う。

3.1 評価項目

2 節より、NPTL の特徴やアーキテクチャの違いを評価するため、以下の処理に着目する。

- スレッド情報取得
- 同期機構の呼出し（Mutex、Semaphore）
- スレッド間コンテクストスイッチ（Mutex、Semaphore、Condition Variable）
- スレッド生成・削除

3.2 評価環境

評価環境は表 1、表 2 のとおり。

表 1：x86 及び MIPS の評価環境(1/2)

	x86	MIPS
CPU	Pentium4	TX49/H4
Freq	3372 MHz	333 MHz
RAM	1024 MB	256 MB
Cache	L1:16KB, L2:2MB	I:32KB,D:32KB

表 2：x86 及び MIPS の評価環境(2/2)

		x86	MIPS
kernel		2.6.11-1	2.6.16-14
gcc		4.0.0	4.1.1
binutil		2.15	2.17
Glibc	NPTL	2.3.5	2.4
	LT	2.3.5	2.3.6

x86 と MIP`S のライブラリ環境が多少異なるが、本評価に影響を及ぼすような違いは存在せず、問題ないと判断している。

3.3 評価結果

評価結果及び NPTL と LT の処理時間比を以下に示す。

表3：スレッド情報取得のオーバヘッド[nsec]

	NPTL	LT	NPTL/LT
X86	5.4	5.9	0.92
MIPS	150	170	0.88

表4：同期機構の呼出しのオーバヘッド[nsec]

		NPTL	LT	NPTL/LT
X86	Mutex	64	80	0.80
	Semaphore	400	150	2.7
MIPS	Mutex	570	520	1.1
	Semaphore	1870	960	1.9

表5：コンテクストスイッチのオーバヘッド[usec]

		NPTL	LT	NPTL/LT
X86	Mutex	1.53	2.74	0.56
	Semaphore	1.54	2.83	0.54
	Condvar	2.28	3.09	0.74
MIPS	Mutex	19.8	62.9	0.31
	Semaphore	19.7	59.8	0.33
	Condvar	29.1	64.6	0.45

表6：スレッド生成・削除のオーバヘッド[nsec]

		NPTL	LT	NPTL/LT
x86	Create	12.5	31.9	0.39
	Join	5.7	28.4	0.20
MIPS	Create	146	637	0.23
	Join	80.5	841	0.096

表 3 より、NPTL と LT の処理時間はほぼ同じ値となっている。これは、NPTL と LT でスレッド情報取得処理が似たような実装のためと考えられる。また、x86 と MIPS の結果を比較すると、CPU クロック周波数比が x86/MIPS=約 10 倍であるのに対し、NPTL や LT の処理時間比は MIPS/x86=約 30 倍となっている。

同期機構のオーバヘッドに関しては、どちらのアーキテクチャでも、semaphore の処理が LT に比べ、NPTL で遅くなっている。これは、semaphore の所有者などの情報が LT ではユーザ空間で共有されていたのに対し、NPTL ではカーネル空間側に切り離されたという変更が影響していると考えられる。この変更により、NPTL では、semaphore の所有者情報などを用いた処理を行う際に、ユーザ空間とカーネル空間のスイッチが発生する。このスイッチのオーバヘッドが処理速度低下の原因と考えられる。

コンテクストスイッチやスレッド生成・削除のオーバヘッドに関しては、全体的に NPTL が LT の処理時間よりも数倍高速である。高速化の要因としては、LT からの変更点である futex の採用や、管理スレッドの廃止などが影響していると考えられる。また、処理時間比 NPTL/LT に関しては、MIPS のほうが上回っているという結果となった。

4. 考察

Hardware でスレッドレジスタがサポートされた場合の、スレッド情報取得の処理時間の改善度について考察してみる。本評価とは別に、x86、MIPS 共に NPTL における TLS 処理以外の処理のオーバヘッドを計測してみた。計測の結果、x86 の場合、表 3 の測定値と同等であった。これより、スレッドレジスタの機能が Hardware でサポートされている場合、スレッド情報取得のための TLS の処理にほとんど時間がかからなくなると仮定できる。MIPS の場合、計測結果は約 60[nsec]であり、上記の仮定より、MIPS での NPTL におけるスレッド情報取得の処理時間は、最大約 90[nsec]程の高速化が期待できる。これにより、x86 と MIPS との TLS の処理時間比はクロック比と同程度になる。またこの考察より、mutex などの同期機構など、内部で TLS を利用している他の処理も高速化されることが期待できる。

5. 結論

本稿では、Linux のスレッドライブラリ NPTL について、従来技術である LT との比較評価を行った。

アーキテクチャ別の評価結果から、Hardware 的な違いが実装によって顕著に現れる TLS の処理は、Hardware の改善によって処理時間の向上が期待できる。また、スレッドライブラリでは、内部処理で TLS を多く利用しているため、TLS 処理時間の向上は、ライブラリ全体の処理の高速化にもつながるであろう。

NPTL の総評としては、LT に対し、semaphore のオーバヘッドが少々目立つものの、その他の高速化により、スレッドライブラリとして NPTL の優位性が確認出来た。NPTL は、現在も開発が進められており、ライブラリ自体の機能追加はもちろん、Hardware やコンパイラ等の改良によるライブラリの処理時間の向上など、スレッドライブラリとして更なる改善が期待できる。

参考文献

[1] The Native POSIX Thread Library for Linux、
http://people.redhat.com/drepper/nptl-design.pdf

[2] Gerry Kane 著、前川守監訳、"mips RISC アーキテクチャ"、共立出版株式会社、1992

[3] fast path for rdhwr emulation for TLS、
http://www.linux-mips.org/archives/linux-mips/2006-07/msg00039.html

[4] MIPS RDHWR instruction reordering、
http://gcc.gnu.org/ml/gcc/2006-06/msg00603.html