システムソフトウェアと
オペレーティング・システム 87－11
（２００１．６．２８）

複数の割り込みを一括処理する一括割り込み処理機構

中島 耕太† 谷口 秀夫†† 雨宮 真人††

†九州大学大学院システム情報科学府
††九州大学大学院システム情報科学研究院

近年、計算機の高速化が著しく、それに伴い計算機周辺の入出力装置の処理速度も向上してきている。また、計算機の取り扱うデータ量の増加により、入出力の頻度も増加してきている。入出力の頻度の増加は、割り込み発生の回数の増加を伴う。割り込み発生の回数が増加すると、他の演算処理に対する性能低下といった影響を与える恐れがある。このため、割り込み処理のオーバヘッドを削減し、効率の高い割り込み処理を実行する必要がある。そこで、我々は、割り込み処理の前処理と後処理のオーバヘッドを削減する手法として、複数の割り込みを一括処理する一括割り込み処理機構を提案する。本論文では、一括割り込み処理機構の設計について述べ、実現した一括割り込み処理機構の基本性能について報告する。

The Package Interrupt Mechanism for Processing Multiple Interrupts

Kohta NAKASHIMA , Hideo TANIGUCHI and Makoto AMAMIYA

Graduate School of Information Science and Electrical Engineering, Kyushu University

As recently performance of peripheral I/O device improves and data handled by computer increases , so the frequency of interrupt increase. The increase in frequency of interrupt influences performance of execution. It is important to reduce overhead of interrupt handle and to handle interrupt more effectively. Thus, we propose the package interrupt mechanism that processing multiple interrupts that reduce overhead of interrupt. In this paper, we report design of the package interrupt mechanism and evaluation of its performance.

1 はじめに

近年、計算機の高速化が著しく、それに伴い計算機周辺の入出力装置の処理速度も向上してきている。また、計算機の取り扱うデータ量の増加により、入出力の頻度も増加してきている。入出力の頻度の増加は、割り込み発生回数の増加を伴う。割り込み発生回数が増加すると、他の演算処理の性能低下といった影響を与える恐れがある。このため、割り込み処理のオーバヘッドを削減し、効率の高い割り込み処理を実行する必要がある。

一般に、CPU は、入出力装置への入出力要求後、入出力装置からの結果が得られるまでの間、他の演算処理を実行し、入出力装置からの結果が得られたことを割り込みにより検知し、入出力処理を実行している。割り込み処理は、実行中の他の演算処理に割り込んで実行されるため、レジスタの退避や回復といった、前処理や後処理を必ず実行する必要がある。

そこで、我々は、割り込み処理の前処理と後処理のオーバヘッドを削減する手法として、複数の割り込みを一括処理する一括割り込み処理機構を提案する。従来の割り込み処理は、その実行毎に全汎用レジスタの退避や回復、割り込みコントローラへの設定処理といった前処理と後処理を伴う。割り込み発生毎に割り込み処理を行なわず、割り込み発生時には、割り込み発生の登録処理のみを行い、後に一括して登録情報をもとに割り込み処理を行う。このように一括して割り込みを処理することにより、従来は各割り込み発生毎に実行した前処理と後処理の削減をはかる。これにより、システム全体の処理性能の向上をはかる。

本論文では、一括割り込み処理機構の設計について述べ、実現した一括割り込み処理機構の基本性能について報告する。

2 逐次割り込み処理と一括割り込み処理

2.1 概要

従来の割り込み処理では、外部から CPU への割り込みが要求されると、これらの割り込みに対して、逐次、割り込みに対応する割り込み処理を行う。こ

の処理は、通常以下のように行われる。まず、割り込みが発生すると、割り込み処理ルーチンは、レジスタ群の退避や、割り込みコントローラへの設定処理といった前処理を行う。次に、各割り込みに対応する処理である本処理を行う。最後に、割り込みコントローラへの処理やレジスタの回復処理といった後処理を行う。このように、割り込みが発生すると、必ず本処理の前後に前処理と後処理を実行する。この処理方式を、以降、逐次割り込み処理方式と呼ぶ。

逐次割り込み処理方式では、割り込み発生毎に必ず前処理と後処理を実行する。このため、割り込みが頻発するような環境では、前処理や後処理の負荷がシステムの負荷全体に占める割合が増加する。

そこで、割り込み発生毎に必要となる前処理、後処理の処理時間を削減する一括割り込み処理方式を提案する。

一括割り込み処理方式の概要を述べる。割り込み発生時には、割り込みが発生したことを記録する割り込み登録処理のみを行う。この登録処理は、割り込みの発生を記録するのみであるため、レジスタの退避/回復処理や割り込みコントローラへの設定処理の一部を省略できる。このように登録された割り込みは、一括割り込み処理の実行契機が訪れた時に処理される。この際、全ての登録されている割り込みについて各割り込み対応の本処理を行う。一括割り込みの実行契機としては、割り込みの登録個数が一定数に達した場合を契機とする方式や、一定時間毎に契機を与える方式が考えられる。このように、複数の割り込みを一括して処理することにより、逐次割り込み処理の際の、前処理、後処理の処理時間を削減する。

2.2 総実行時間の比較

逐次割り込み処理方式による場合と、一括割り込み処理方式による場合との割り込み処理の総実行時間の比較について述べる。逐次割り込み処理方式による場合の命令実行の様子を図 1、一括割り込み処理方式による場合の命令実行の様子を図 2 に示す。

逐次割り込み処理方式では、割り込み発生毎に、前処理、本処理、後処理を実行する。前処理の処理時間を t_1、本処理の処理時間を t_2、後処理の処理時間を t_3 とすると、N 回割り込みが発生した場合の割り込み処理の総実行時間は、

$$N(t_1 + t_2 + t_3) \tag{1}$$

である。

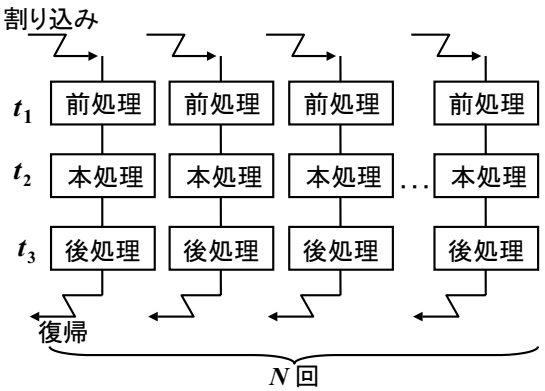

図 1 逐次割り込み処理方式での命令実行

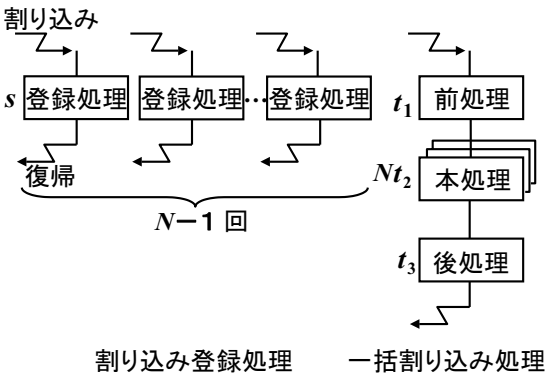

図 2 一括割り込み処理方式での命令実行

一方、一括割り込み処理方式では、N 回の割り込みを契機に一括割り込み処理を行うとすると、登録処理は $N-1$ 回行われる。そして、N 回目の割り込みが発生した際に、前処理を 1 回、本処理を N 回、後処理を 1 回実行する。登録処理の処理時間を s、前処理の処理時間を t_1、本処理の処理時間を t_2、後処理の処理時間を t_3 とすると、N 回割り込みが発生した場合の割り込み処理の総実行時間は、

$$(N-1)s + t_1 + Nt_2 + t_3 \tag{2}$$

である。また、一括処理の契機を一定期間毎とする場合も、一定期間毎に発生する割り込みの平均回数を N 回とすれば、一定期間に実行される割り込み処理の総実行時間は、式 (2) と同様である。

逐次割り込み処理方式と、一括割り込み処理方式の総実行時間の差は、式 (1)− 式 (2) より、

$$(N-1)(t_1 + t_3 - s) \tag{3}$$

である。ただし、

$$t_1 + t_3 - s > 0 \tag{4}$$

であるとき、式 (3) は正である。つまり、式 (4) で示される条件が満たされる場合、すなわち、前処理と

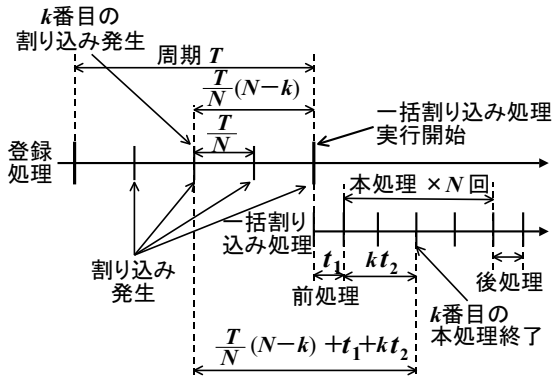

図 3　一括割り込み処理処理方式の処理の時間経過

後処理の処理時間の和より登録処理の処理時間が短ければ、一括割り込み処理方式の方が割り込み処理総実行時間は短く、割り込み処理を一括することによる利得が得られる。

2.3　割り込み平均遅延時間の比較

割り込みの平均遅延について述べる。

逐次割り込み処理方式の場合、割り込み発生直後に前処理、本処理、後処理を実行するため、割り込みが発生してから各割り込みの本処理が終了するまでの時間は、

$$t_1 + t_2 \tag{5}$$

である。

一方、一括割り込み処理方式の場合、割り込み処理実行の処理の時間経過を図 3 に示す。等間隔で時間 T の間に N 回割り込みが発生し、また、N 回目の割り込み発生時に一括割り込み処理を実行すると仮定する。尚、図 3 では、$N = 4$ の場合を示している。このように仮定すると、割り込み発生の間隔は、

$$\frac{T}{N} \tag{6}$$

となる。この期間に発生した割り込みはこの期間が終了する時に一括して処理される。この期間の第 k 番目に発生した割り込みに対する処理は、一括割り込み処理の開始まで待たされる。図 3 では、$k = 2$ の場合を示している。第 k 番目に発生した割り込みが発生してから、一括割り込み処理が起動されるまでの時間は、式 (6) より

$$\frac{T}{N}(N - k) \tag{7}$$

である。さらに、一括割り込み処理の開始から第 k 番目に発生した割り込みに対応する処理の実行が完了するまでにかかる時間は、

$$t_1 + kt_2 \tag{8}$$

である。すなわち、第 k 番目の割り込み発生から本処理が終了するまでの時間は、式 (7) と式 (8) より、

$$\frac{T}{N}(N - k) + t_1 + kt_2 \tag{9}$$

である。したがって、この期間に発生した割り込みからそれぞれに対応する割り込み処理が実行されるまでの平均時間は、式 (9) より、

$$\frac{1}{N}\sum_{k=1}^{N}\left\{\frac{T}{N}(N - k) + t_1 + kt_2\right\} \tag{10}$$

である。これを整理すると、

$$\frac{1}{2}\left(\frac{N-1}{N}\right)T + t_1 + \frac{1}{2}(N+1)t_2 \tag{11}$$

となる。

逐次割り込み処理方式と一括割り込み処理方式を比較すると、一括割り込み処理方式と逐次割り込み処理方式の平均的な遅延の差は、式 (11)− 式 (5) より、

$$\frac{1}{2}\left(\frac{N-1}{N}\right)T + \frac{1}{2}(N-1)t_2 \tag{12}$$

となる。

2.4　総合比較

逐次割り込み処理方式と一括割り込み処理方式の長短を表 1 に示す。逐次割り込み処理方式は、割り込みが発生してから即座に本処理を行うので応答時間が短い。一方、全ての割り込みに対して、全汎用レジスタの退避や回復、割り込みコントローラへの設定処理といった前処理や後処理を行う必要があり、総実行命令数は、一括割り込み処理方式より多い。

一括割り込み処理方式は、割り込み登録処理時に、逐次割り込み処理方式での前処理や後処理のオーバヘッドが削減できる。このため、総実行命令数は、逐次割り込み処理方式より少ない。一方、割り込みが発生してから一括割り込み処理を実行するまで遅延があるため、割り込みが発生してから本処理の実行までの応答時間が長い。

このように、逐次割り込み処理方式は、応答時間重視である。実時間処理を行う場合や、MPI(Message Passing Interface) を用いて複数計算機上で並列計算を行う場合といった応答時間が重要視される場合には、逐次割り込み処理方式の方が適している。

表 1　逐次割り込み処理方式と一括割り込み処理方式の比較

割り込み処理方式	長所	短所
逐次割り込み処理方式	割り込み発生から割り込み処理実行までの時間が短い	割り込み処理の総実行命令数が多い
一括割り込み処理方式	割り込み処理の総実行命令数が少ない	割り込み発生から割り込み処理実行までの時間が長い

図 4　プロセス状態の遷移

一方、一括割り込み処理方式は、スループット重視である。入出力回数が頻発するが、応答時間はあまり重要ではない場合には、一括割り込み処理方式の方が適している。例として、プロセススケジューラがタイムスライスのみをサポートし、プリエンプションをサポートしないシステムの場合を考察する。この様子を図 4 に示す。図 4 では、タイムスライス間隔が 1.0 秒で、優先度 5 のプロセス A は、1.0 秒プロセッサ処理を行い、入出力待ち 0.5 秒を繰り返すとする。優先度 3 のプロセス B は、プロセッサ処理を実行し続けるものとする。

このシステムで、逐次割り込み処理方式で割り込み処理を行った場合について述べる。まず、時刻 0.0 秒からプロセス A が Run 状態となり、プロセス B は Ready 状態となる。時刻 1.0 秒の時点で、プロセス A は入出力待ちとなり、Wait 状態に遷移する。また、タイムスライス処理が実行され、Ready 状態のプロセス B が Run 状態に遷移する。時刻 1.5 秒の時点で、入出力完了割り込みが発生し、割り込み処理によりプロセス A は Wait 状態から、Ready 状態に遷移する。時刻 2.0 秒の時点で、タイムスライス処理が実行され、プロセス A が Run 状態へ、プロセス B が Ready 状態へ遷移する。

次に、一括割り込み処理方式で割り込み処理を行った場合について述べる。ここでは、一括割り込み処理の契機をタイムスライス処理の実行直前とする。まず、時刻 0.0 秒からプロセス A が Run 状態となり、プロセス B は Ready 状態となる。時刻 1.0 秒の時点で、プロセス A は入出力待ちとなり、Wait 状態に遷移する。また、タイムスライス処理が実行され、Ready 状態のプロセス B が Run 状態に遷移する。時刻 1.5 秒の時点で、入出力完了割り込みが発生するが、ここでは割り込み登録処理のみが行われる。したがって、プロセスの状態は遷移しない。時刻 2.0 秒の時点で、一括割り込み処理が実行される。このため、プロセス A は Ready 状態に遷移する。さらに、タイムスライス処理が実行され、プロセス A が Run 状態へ、プロセス B が Ready 状態へ遷移する。この場合、割り込み処理が遅延することによって、プロセス A の走行が遅延することはなく、逐次割り込み処理の場合と同様にプロセス A が走行する。このように逐次割り込み処理方式の場合とタイムスライスを契機として一括割り込み処理を行った場合とでは、入出力待ちのプロセスにプロセッサが割り当てられるまでの時間は変わらない。したがって、一括割り込み処理方式の短所である割り込み発生から割り込み処理実行までの遅延時間が長いという問題が隠蔽され、この短所がシステムの応答時間に悪影響を与えていない。このような場合では、一括割り込み処理方式は大きく効果を発揮する。

UNIX では、タイムスライス機能とプリエンプション機能があるが、プリエンプション機能については、カーネル内を走行中はプリエンプション不可である。カーネル内には、TCP/IP といった通信プロトコル処理が実装されており、ソケットシステムコール内では非プリエンプション時間が長い。したがって、通信が頻発する場合には、プリエンプションされる場合が減少し、上記のように一括割り込み処理方式が大きく効果を発揮する可能性がある。

3　一括割り込み処理機構

3.1　一括割り込み処理機構の概要

一括割り込み処理機構の概要を図 5 に示す。一括割り込み処理機構は、各割り込み発生毎に割り込みを登録する割り込み登録処理と、一括して各割り込みに対応する処理を行う一括割り込み処理からなる。まず、割り込みが発生すると、割り込み登録処理が呼び出され、割り込み登録情報表に発生した割り込

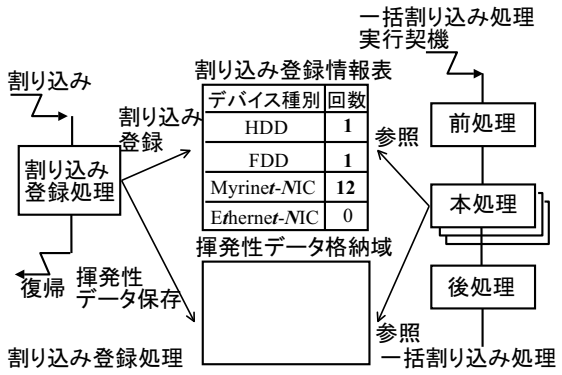

図 5 一括割り込み処理概要

みの登録を行う。また、次の割り込みが発生すると消失する揮発性データを揮発性データ格納域に保存する。そして、一括割り込みの契機で一括割り込み処理を実行する。一括割り込み処理では、前処理を行った後、割り込み登録処理で記録した情報を参照し、各割り込みに対応する本処理を一括して実行する。その後、後処理を実行する。

以降、割り込み登録処理と一括割り込み処理について述べる。

3.2 割り込み登録処理

割り込み登録処理について述べる。式 (4) を満たすために割り込み登録処理時間は、前処理と後処理の処理時間よりも短くなければならない。

逐次割り込み処理では、割り込み発生時の処理は以下のようになる。ここでは、例として PC/AT 互換機での場合について述べる。

(1) 全汎用レジスタの退避処理 (前処理)
(2) 割り込みコントローラへの割り込み受け取り通知処理 (前処理)
(3) 割り込みマスク変更処理 (前処理)
(4) 割り込み禁止解除処理 (前処理)
(5) 本処理
(6) 割り込み禁止処理 (後処理)
(7) 割り込みマスク復元処理 (後処理)
(8) 全汎用レジスタの回復処理 (後処理)

割り込みが発生し、登録されている割り込み処理ルーチンが起動されるときには、割り込み禁止状態である。割り込み禁止状態の間に、処理 (1)、(2) および (3) を行う。全汎用レジスタの退避処理を行うことにより、以降の処理で汎用レジスタを制限無く使用することが可能になる。割り込みコントローラへの割り込み受け取り通知処理を行うことにより、割り込みコントローラは、割り込みを再び受け付けることが可能になる。割り込みマスク変更処理により、割り込み禁止解除処理後に走行する本処理中で許可/不許可する割り込みを指定することが可能になる。

割り込み禁止処理、割り込みマスク回復処理、全汎用レジスタの回復処理は、前処理に対する後処理であり、割り込みマスクや汎用レジスタの状態を割り込み発生前の状態に復元するための処理である。

逐次割り込み処理方式における全汎用レジスタの退避/回復処理、割り込みマスク変更/復元処理は、本処理で、自由にレジスタを使用したり、優先度の高い割り込みの多重処理を可能にするために行う処理である。割り込み登録処理は、本処理と比較して処理が単純であるため、これらの処理の一部を削減することが可能である。

割り込み登録処理で必要となる処理は、以下の通りである。

(1) 割り込み登録処理で使用するレジスタの退避処理
(2) 割り込みコントローラへの割り込み受け取り通知処理
(3) 割り込み発生の記録処理
(4) ハードウェア上の揮発性データの保存処理
(5) 割り込み登録処理で使用するレジスタの回復処理

割り込み登録処理で使用するレジスタの退避処理では、割り込み登録処理で使用するレジスタのみの退避処理を行う。退避するレジスタの本数を削減することにより、レジスタ退避/回復処理の軽減をはかる。また、割り込み登録処理では、逐次割り込み処理の前処理と後処理での割り込みマスクの変更/復元処理を省略する。これは、割り込み登録処理中に他の割り込みの受け付けを許可する必要が無いためである。割り込み発生の記録処理では、割り込みが発生したことを記録する処理を行う。この処理では、各デバイス毎の割り込み発生回数を割り込み登録情報表に記録する。ハードウェア上の揮発性データの保存処理では、ハードウェア上のデータのうち、次の割り込みが発生すると消失する揮発性データを保存するため、ハードウェア上の揮発性データを揮発性データ格納域に保存する。

3.3 一括割り込み処理

一括割り込み処理について述べる。一括割り込み処理は、一括割り込み処理の契機により起動され、登録された割り込みを一括処理する。一括割り込み処

理では、前処理を実行し、本処理を一括する割り込みの回数分実行し、後処理を実行する。

前処理は、逐次割り込み処理の前処理同様の前処理を行う。具体的には、以下の処理を行う。

(1) 割り込み禁止処理
(2) 全汎用レジスタの退避処理
(3) 割り込みコントローラへの割り込み受け取り通知処理
(4) 割り込みマスク変更処理
(5) 割り込み禁止解除処理

割り込み処理の前処理中に他の割り込みの実行を禁止する必要がある。このため、一括割り込み処理起動時に割り込み可能状態である場合には、割り込み禁止処理を行う必要がある。処理 (2) から処理 (5) については、逐次割り込み処理の前処理と同様である。

これらの前処理の後、一括して実行する本処理を連続実行する。この処理は、割り込み登録情報表や、揮発性データ格納域を参照して実行する。

本処理終了後、後処理を実行する。以下の処理 (1) から処理 (3) は、逐次割り込み処理の後処理と同様である。後処理では、一括割り込み処理実行前の状態の復元する必要がある。このため、一括割り込み処理実行前が割り込み可能状態であった場合には、割り込み許可処理を行う必要がある。

(1) 割り込み禁止処理
(2) 割り込みマスク復元処理
(3) 全汎用レジスタの回復処理
(4) 割り込み許可処理

次に、一括割り込み処理を行う契機について述べる。一括割り込み処理を行う契機には、割り込み発生回数が一定数に達した場合に処理する方式と一定期間毎に処理する方式がある。

割り込み発生回数が一定数に達した場合に処理する方式では、割り込み登録処理時に、登録されている割り込みが一定個数に達していた場合に、一括割り込み処理を行う。この場合、一括割り込み処理を行う間隔は、割り込みの発生頻度に依存する。すなわち、割り込みの発生頻度が多い時は、一括割り込み処理が実行される間隔も短くなり、割り込みの発生頻度が少ない時は、一括割り込み処理が実行される間隔が長くなる。割り込み発生頻度と一括割り込み処理の頻度が一定の割合である場合は、割り込み発生頻度に準ずる一括割り込み処理を行うことが可能である。一方、この方式では、一定数の割り込み発生を待って一括割り込み処理を行うため、一定数未満の割り込み発生の後、割り込みが発生しなかった場合には、登録されている割り込みを実行することができない。このため、一定時間以上割り込みが発生しなかった場合には、タイムアウト処理として一括割り込み処理を行う機能が必要である。

一定期間毎に処理する方式では、タイマを用いて一定期間毎にその期間に割り込みが発生したかどうか調査し、発生していた場合には、その間に発生していた割り込みに対する処理を行う。この場合、一括割り込み処理は、定期的に行われる。従って、割り込み発生の頻度に関わらず、一定期間毎に一括割り込み処理が行われる。このため、一定期間以上割り込みに対する処理が遅れることが無いことを保証できる。一方、割り込み発生頻度が低い場合には、一括する割り込みの個数が少ないため、割り込みを一括することによる処理時間の削減効果が低下する。つまり、割り込み頻度によって、処理時間の削減効果が変動する。

このように一括割り込み処理の実行契機には、それぞれの方式について長短があり、どちらの方式が良いかは、扱う入出力の性質による。ディスク等の入出力では、ディスクへの入出力要求後に、必ず入出力完了割り込みが発生する。したがって、割り込みの発生状況を事前に把握することが可能であるため、一定数の割り込み発生を実行契機とする方法を用いることが可能である。一方、通信処理の入出力では、通信の受信処理を考えると、受信側計算機は、あらかじめデータの受信を把握することができない。つまり、データの受信回数を予測できないため、一定数の割り込み発生を実行契機とする方式を用いても、タイムアウト処理を行う可能性が高い。したがって通信処理では、一定期間毎に処理する方式を用いる方が良い。

4 実装と評価

4.1 実装内容

我々の研究室で開発を行っている ***Tender*** オペレーティングシステム [1] の割り込み制御部に一括割り込み処理機構を実装した。一括割り込み処理機構の実装対象デバイスを Myrinet[2] 通信を制御するハードウェア (以降、Myrinet-NIC と呼ぶ。) とした。理由は以下の通りである。Myrinet は、バンド幅 1.28Gbps(160MB/s) のハードウェア性能を持つ高速通信路であり、***Tender*** の Myrinet 制御についても、実測した結果バンド幅は 84.5MB/s であった。したがって、4KB のデー

タを連続受信する場合、Myrinet-NIC から 1,000μs 間に、約 20 回程度の割り込みが発生する可能性がある。このような理由により、Myrinet-NIC を一括割り込み処理機構の実装対象デバイスとした。

Myrinet-NIC は、受信完了時に割り込みを発生させる。したがって、割り込み処理では受信完了処理を行う。

割り込み登録処理では、割り込みの発生回数を記録する処理と、次の通信が発生すると前の通信時のデータが消失する揮発性のデータの保存処理を行う。Myrinet-NIC の場合、次の通信が発生すると消失する揮発性のデータは、Myrinet-NIC 上の記憶領域にある。Pentium II 450MHz の実測によると、Myrinet-NIC 上の領域から主記憶への 4B のデータ複写時間は 0.540μs であった。Myrinet-NIC 上にある次の通信が発生すると消失する揮発性のデータは、合計 8B である。

一括割り込み処理は、タイマ割り込みを契機に実行することとした。本実装では、タイマの周期は1,000μs とした。一括割り込み処理では、割り込み登録処理により登録されたデータを元に、本処理を一括して実行する。本処理では、受信完了処理を実行する。受信完了処理としては、受信サイズの取得処理、受信プロセスを Wait 状態から Ready 状態に変更する処理がある。この際、次の通信が発生すると消失する揮発性のデータは、登録処理時に主記憶に保存されているので、逐次割り込み処理を行う際よりも、処理が高速化する。

4.2 測定条件

Tender に実現した一括割り込み処理機構の基本性能の評価を行った。また、比較のため、逐次割り込み処理方式による割り込み処理の性能も評価を行った。

測定には、Pentium II 450MHz、主記憶 128MB の PC/AT 互換機 2 台を Myrinet で接続した環境を用いた。Myrinet-NIC は、PCI バス (32bit、33MHz) で計算機と接続されている。

一括割り込み処理機構、逐次割り込み処理機構共に、測定対象となる OS 内のモジュールの実行時間を測定した。測定は、2 台の計算機間で 4KB のデータを 1 回往復させる。一括割り込み処理機構については、その際に発生した割り込みに対する割り込み登録処理、一括割り込み処理の前処理、本処理 (1 回分)、後処理を測定した。また、逐次割り込み処理機構については、前処理、本処理、後処理を測定した。

表 2 基本性能と割り込み処理総実行時間

モジュール	一括割り込み処理 (μs)	逐次割り込み処理 (μs)
割り込み登録処理	2.80	—
前処理	3.00	3.00
本処理	16.17	16.77
後処理	1.82	1.82
N 回の処理時間の合計	$18.97N + 2.02$	$21.59N$
20 回の処理時間の合計	381.4	431.8

この測定を 50 回行い、1 回あたりの平均を算出した。

測定には、CPU のハードウェアクロックカウンタを用い、2 箇所のカウンタ値の差分をクロック数で割ることにより、処理時間を算出した。

4.3 基本性能

基本性能として、一括割り込み処理機構の登録処理と一括割り込み処理の前処理、本処理、後処理を測定した。本処理の測定は、1 回分の割り込みに対する処理を測定した。また、比較のため、逐次割り込み処理の前処理、本処理、後処理を測定した。結果を表 2 に示す。

4.1 節で述べたように、逐次割り込み処理の本処理と、一括割り込み処理の本処理とを比較すると、一括割り込み処理の本処理の方が高速である。また、逐次割り込み処理の前処理時間と後処理時間の和は、4.82μs であり、一括割り込み処理の割り込み登録処理時間は、2.80μs であるため、式 (4) を満たす。すなわち、一括割り込み処理方式の方が逐次割り込み処理方式よりも、割り込み処理の総実行命令時間が短くなることを示している。

4.4 割り込み処理総実行時間の比較

割り込み処理総実行時間の比較を表 2 に示す。N 回の処理時間の合計と 20 回の処理時間の合計については、表 2 で示した数値を元に、式 (1) と式 (2) を用いて算出した式または数値である。

N 回の処理時間の合計について一括割り込み処理方式と逐次割り込み処理方式を比較すると、N の係数の値が、一括割り込み処理方式の方が小さい。これは、一括する割り込みの個数が増加すると、逐次割り込み処理方式と比較した場合の一括割り込み処理方式の総処理時間の削減分が増加することを意味する。両者が同じ時間になる N は、$N = 1.25$ である。つまり、$N \geq 2$ の時、一括割り込み処理時間が短くなる。具体的には、20 回の割り込みを一括して処理すると、約 11.7%に相当する 50.4μs 処理時間が削減できることが期待できる。この時、1,000μs 間に

表 3 割り込み処理平均遅延時間の比較

場合	一括割り込み処理 (μs)	逐次割り込み処理 (μs)
一括割り込み処理周期 $T\mu s$、 一括割り込み処理数 N 回の場合	$\frac{1}{2}\left(\frac{N-1}{N}\right)T$+8.09$N$+11.09	19.77
一括割り込み処理周期 1,000μs、 一括割り込み処理数 20 回の場合	647.8	19.77

他の演算処理に与えることが可能な演算時間は、一括割り御子処理方式の場合 618.6μs であり、逐次割り込み処理方式の場合 568.2μs である。このことから、他の演算処理の性能が約 8.87%向上することが期待できる。

4.5 割り込み処理平均遅延時間の比較

割り込み処理平均遅延時間の比較を表 3 に示す。表 3 において、一括割り込み周期 $T\mu$s、一括割り込み処理数 N 回の場合は、一括割り込み処理方式については、一括割り込み周期 $T\mu$s、一括割り込み処理数 N 回の場合の割り込み処理平均遅延時間を示し、逐次割り込み処理方式については、割り込み処理の平均遅延時間を示している。一括割り込み周期 1,000μs、一括割り込み処理数 20 回の場合は、一括割り込み処理方式については、一括割り込み周期 1,000μs、一括割り込み処理数 20 回の場合の割り込み処理平均遅延時間を示し、逐次割り込み処理方式については、割り込み処理の平均遅延時間を示している。これらの割り込み処理平均遅延時間は、式 (5) と式 (11) を用いて算出した式または数値である。

一括割り込み周期 T や、一括割り込み処理数 N が増加すると、一括割り込み処理時の割り込み処理平均遅延時間は増加する。具体的には、一括割り込み周期が 1,000μs であり、一括割り込み処理数が 20 回の場合には、割り込み処理平均遅延時間は、647.8μs となり、逐次割り込み処理の遅延時間 (19.77μs) と比較すると非常に大きい。このため、一括割り込み処理方式では、実時間処理といった応答遅延時間に厳しい制約を求められるシステムには不向きである。一方、プリエンプション機能をサポートしないような、応答遅延時間の制約が比較的緩いシステムでは、あまり大きくシステム全体の性能に影響を与えないものと考えられる。例えば、プリエンプション機能をサポートしないシステムでタイムスライス間隔が 1,000μs である場合、逐次割り込み処理方式では、割り込み発生から入出力待ちのプロセスに CPU が割り当てられるまでの時間は、平均 500μs である。この場合と一括割り込み処理方式の平均遅延時間 (647.8μs) を比較すると、大きな遅延ではない。

5 おわりに

複数の割り込みを一括処理する一括割り込み処理機構を提案した。一括割り込み処理方式を用いることで、割り込み処理の前処理や後処理といったオーバヘッドを削減できる。一方、一括割り込み処理方式を用いると、割り込み発生から割り込み処理が実行されるまでの遅延時間が増大する問題点がある。逐次割り込み処理方式と一括割り込み処理方式では、それぞれ長短があり、応答時間重視の処理においては逐次割り込み処理方式、スループット重視の処理においては一括割り込み処理方式が優れている。

また、一括割り込み処理機構の設計について述べた。一括割り込み処理機構は、割り込み登録処理と一括割り込み処理から構成され、割り込み登録処理での処理内容、一括割り込み処理での処理内容、および、一括割り込み処理での処理契機について述べた。

さらに、一括割り込み処理方式を ***Tender*** の Myrinet 制御に適用し、基本性能を評価した。基本性能から、前処理や後処理のオーバヘッドの削減量や前処理や後処理のオーバヘッドの削減による他の演算処理の性能向上性について評価し、1,000μs 間に 20 回の割り込みが発生した場合、割り込み処理時間を約 11.7%削減し、他の演算処理の性能が約 8.87%向上することが期待できると述べた。また、基本性能から、割り込み処理の平均遅延時間を評価し、1,000μs 間に 20 回の割り込みが発生した場合、一括割り込み処理方式では 672.8μs、逐次割り込み処理方式では、19.77μs の遅延が生じると述べた。

残された課題として、一括割り込み処理機構の様々な入出力装置への適用や一括割り込み処理方式による入出力処理実行時の他の演算処理の影響評価がある。

参考文献

[1] 谷口秀夫, 青木義則, 後藤真孝, 村上大介, 田端利宏 : “資源の独立化機構による ***Tender*** オペレーティングシステム”, 情報処理学会論文誌, Vol.41 , No.12 , pp.3363-3374(2000)

[2] Nanette J.Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su: “Myrinet – A Gigabit-per-Second Local-Area Network”, IEEE-Micro,Vol.15,No.1,February 1995, pp.29-36