Hitachi SR16000 を利用する際の注意点

2014 年 11 月 28 日

理化学研究所仁科加速器研究センター 土井琢身

Hitachi SR16000 を利用する上で、問題となる可能性のある事象についてまとめる。

行った計算： 格子 QCD による核力の計算
使用言語、コンパイラ： C/C++, IBM XL C/C++ compiler for AIX

1. 可変長配列のメモリ確保のスコープについて

[問題現象]
可変長配列のメモリ確保/解放のタイミングがプログラマの期待と異なる振る舞いをするため、スタック領域を食い潰してプログラムがクラッシュする。

[発生環境]
SR16000 における IBM XL C/C++ compiler for AIX（おそらく、AIX 上であれば machine/compiler を問わない）

[具体例]
次のような、可変長配列をスタックに取るコードを考える。

```
----------
void func(int n)
{
    for(int i=0; i<1000; i++) { // scope-A
      int array[n];
      // hoge
    }
}
int main()
{
    for(int i=0; i<100; i++) {
      func(1);
    }
    // fuga
```

}

プログラマの期待は、可変長配列 array[] は、そのスコープ(i.e., scope-A) に対応してスタック領域に確保・解放が行われるというものであろう。実際、GCC on x86 ではそのように動作するようである。この場合、サイズ n があまり大きくない限りはスタックを食い潰すことはない。

しかし、compiler/OS によっては、以下のように、スタックメモリ確保・解放のタイミングが異なるため、問題が起こり得る。

サポートセンターとのやり取りにより、「AIX ではスタックが解放されるタイミングがプロセス終了時となります。」との仕様が判明した。従って、上記コードでは、プログラマの期待よりも 100*1000 倍スタックを異常消費してしまうことになる。

なお、報告者が検証したのは IBM XL C/C++ compiler on SR16000 だけであるが、上記仕様からは、AIX 上であれば machine/compiler によらず本問題に該当すると予想される。

なお、固定長配列では、上記のような問題は無い。 また、BlueGene/Q 上での IBM XL C/C++ compiler については(明示的な確認/検証は行っていないものの)、報告者の経験では上記のような問題は起こらず、正しいスコープで確保・解放しているようである。(これも、(2) については問題は compiler でなく AIX にあることと consistent である。)

[対策]
たとえサイズが小さくても、可変長配列を自動変数としてスタックに取るのはやめ、malloc/free, new/delete 等でヒープ領域に明示的に確保・解放する。 あるいは、可能であれば、自動変数のままでもループの外側で定義すれば問題は軽減される。 なお、C++ では(未検証ではあるが) vector コンテナ等を使っても解決するはずである。

AIX について仕様改善の希望を伝えたが(2013/07)、その後の具体的な動きについては聞いていない。

2. OpenMP + MPI hybrid 並列時のスタック領域について

[問題現象]
スレーブスレッド用にシステムがデフォルトで確保するスタック領域が小さすぎる為、スタックオーバーフローでプログラムがクラッシュする。

[発生環境]
SR16000 における IBM XL C/C++ compiler

[原因・対策」
日立サポートセンターによる調査の結果、以下が判明した。

[原因]
XL C/C++ compiler では、１スレッドあたりのスタックサイズのデフォルトが次のようになっている。
 マスタースレッド(メインスレッド)　　：4GB
 スレーブスレッド(プロセスの２つ目以降のスレッド)：4MB
従って、プログラムがこれ以上のスタックを使用すると、
セグメンテーション障害(スタックオーバーフロー)により異常終了する。

[対策 1]
malloc 及び free を用いて、配列をヒープ領域に確保及び解放するようにソースコードを修正する。

[対策 2]
ソースコードの変更をしない場合は、ジョブの JCF のプログラム実行前に
　OMP_STACKSIZE もしくは XLSMPOPTS stack
の環境変数を追加して、スレーブスレッドのスタックサイズを拡張する。

例えば、32MB を指定する場合、
　setenv OMP_STACKSIZE 32M
または
　setenv XLSMPOPTS stack=33554432

[スレーブスレッドのスタックサイズに指定できる値]
・32 ビット・モードの場合：　8KB 以上 256MB 以下
・64 ビット・モードの場合：16KB 以上　64GB 未満

なお、unlimited 相当の設定にすることはできない。また、上記範囲外の値を設定してジョブを実行した場合は、次のようなメッセージが標準エラーファイルに出力され、スレーブスレッドのスタックサイズはデフォルト値(4MB)に設定され、ジョブが実行される。

[スレーブスレッドのスタックサイズが不正な場合のメッセージ]

1587-106 The value [size] specified for option 'OMP_STACKSIZE' is not in the valid range 16384 to 68719476735. All SMP run-time options have been set to their default values.

ただし、OMP_STACKSIZE 環境変数を B 単位で 2GB 以上を指定した場合は、すべて 2GB と設定される。(K、M、G 単位で指定した場合は問題は無い。 また、XLSMPOPTS stack 環境変数を使用する場合は、B 単位でも 2GB 以上を指定可。)

[補足]

スレーブスレッドのスタックオーバーフローの検出方法としまして、XL C/C++ コンパイラの -qsmp=stackcheck オプションをご紹介させて頂きます。

コンパイルオプションに "-qsmp=stackcheck" を追加し、
ジョブの JCF に環境変数 "XLSMPOPTS=stackcheck=[num]"(num:バイト)を追加することで、スレーブスレッドの残りのスタックサイズが [num] バイトより小さくなると、下記のようなランタイム警告メッセージがジョブの標準エラーファイルに出力されます。

「1587-122 SMP runtime library error. The stack for thread 1 is within 524288 bytes of overflowing.」

(上記は JCF に "setenv XLSMPOPTS stackcheck=524288" を追加した場合です)

但し、上記の stackcheck のサイズ以上のスタックを一度に確保しようとし、スタックオーバーフローが発生した場合は、当該メッセージは出力されません。

また、本オプションを追加すると、チェックコードが挿入されるため、パフォーマンスに影響が出る可能性がございます。具体的にどのくらいの影響が出るかという指標はございません。もしご利用される際はお客様プログラムにおいてご検証して頂き、許容範囲かどうかご確認願います。

[参考情報]

コンパイラの仕様について：

[IBM XL C/C++ for AIX コンパイラ・リファレンス]

https://scwww.kek.jp/sc-man/aix-pdf/xlcccomref.pdf

・XLSMPOPTS 環境変数 (stack, stackcheck オプション) (pp.34-35) (v.12.1: p.34)

・OMP_STACKSIZE 環境変数 (pp.40-41) (v.12.1: p.44)

・-qsmp=stackcheck オプション (p.331) (v.12.1: p.359)
(報告者注：上記マニュアルを見るには KEK System A のアカウントが必要。
報告者参照時は バージョン 11.1, リンク先は最新版。
また、System A ユーザーは wiki
「要素並列化なしで正常に実行できるジョブを要素並列化して実行すると、セグメンテーション違反によりジョブが異常終了する。」
https://scwww.kek.jp/sc-man/wiki/index.php?FAQ#tc831ba5
も参照のこと。)

なお、上記コンパイラマニュアルには、スタックサイズのデフォルト値について、上の説明と異なる記述があるが、日立サポートセンターによる検証の結果、マニュアル不備と考えられる。日立から開発元(IBM)に修正を依頼した。

以上