依頼内容

格子 QCD シミュレーションにおいて、クォーク伝播関数を読み込んで Ω-バリオンの４点相関関数を計算したい。
現在は全く並列化されていないプログラムを MPI を実装させて早くしたいので、どういういう手順を踏んで並列化をすればいいか教えてほしい。

使用する計算機：KEK Blue Gene/Q
質問など：KEK Blue Gene/Q と PC の並列化で特筆すべき違いがもしあれば教えてください。

依頼者の意図：
　現在、依頼者が PC で行っている格子 QCD の計算をスーパーコンピュータによる大規模計算に適用するため、依頼者が所有しているプログラムを大規模並列計算用に拡張し、高速化したいとのことである。
計算したい量は
$\sum_{\vec{x}} \langle B_1(\vec{x}+\vec{r},t) B_2(\vec{x},t) \bar{B}_2(\vec{y},t_0) \bar{B}_1(\vec{y},t_0) \rangle$
であり、クォーク伝播関数 $Q^{-1}(\vec{x},t;\vec{y},t_0)$ の積で記述される量である。
(複雑さを避けるため、スピンなどの情報は落とした。)

回答：

今回、プログラムを並列化する目的として、
（1）多数の計算ノードを用いて並列に計算を進め、計算に要する時間を短縮する。
（2）計算に必要なメモリを多数の PC に分散し、大容量のメモリ領域を必要とするシミュレーションへの展開を図る。
の２つがあると思われます。
このうち(1)については、マルチプロセッサー型のコンピュータにおいて単純に処理を分散する事によって達成できると考えられます。

目的(2)は将来的に大規模計算への展開を考慮した場合に、プログラムを MPI 並列化し分散メモリー型並列計算機に載せる事で達成できると考えられる。
今回の依頼の回答として、これらの目的を達成するために分散メモリー型並列計算機を想定し、並列化の方針を助言しました。

バリオンの４点相関関数は、クォーク伝播関数の積の和で書き下すことが出来ます。そのため、計算に必要となる要素はクォーク伝播関数６個の積で

$$\sum_{\vec{x}} \langle B_1(\vec{x}+\vec{r},t)\, B_2(\vec{x},t)\, \bar{B}_2(t_0)\, \bar{B}_1(t_0) \rangle$$
$$= \sum_{sfc} Q^{-1}(\vec{x}+\vec{r},t;t_0) Q^{-1}(\vec{x}+\vec{r},t;t_0) Q^{-1}(\vec{x}+\vec{r},t;t_0) Q^{-1}(\vec{x},t;t_0) Q^{-1}(\vec{x},t;t_0) Q^{-1}(\vec{x},t;t_0)$$

で与えられます。和は spin, flavor, color の適当な組について取られます。
この量は更に３個のクォーク伝播関数の積に分解できて次の量を定義します。

$$\tilde{B}(\vec{x},t-t_0) = \sum_{sfc} Q^{-1}(\vec{x},t;t_0) Q^{-1}(\vec{x},t;t_0) Q^{-1}(\vec{x},t;t_0)$$

あらかじめこの量を格子空間の全点で計算しておく事で、求めるべき量の計算の下準備を行います。
この量はノード間通信を必要としない計算なので、まずはこの量の計算方法について方針を示します。

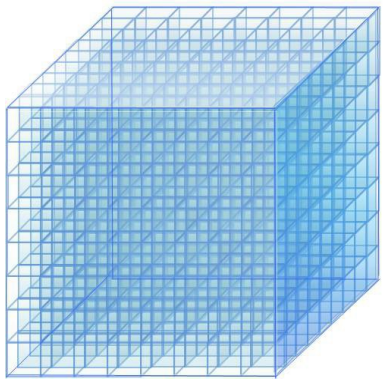

図 1:物理格子

まず、計算機の構成と格子を対比させ、１つの CPU が担当する格子領域を分割し並列化のイメージをつくります。

たとえば、8x8x8 のサイズの格子を例として、図１のように 2x2x2 のトーラスネットワークによって計算ノードが接続されているクラスター計算機上で計算をする場合には、格子を 4x4x4 の８つの部分格子に分解し、それぞれの計算を計算ノードに分けて並列的に計算を行います。
今回の依頼については分割した格子内の計算量に偏りが少ないと思われるので、上記のような単純分割で最適なパフォーマンスが得られると考えられます。

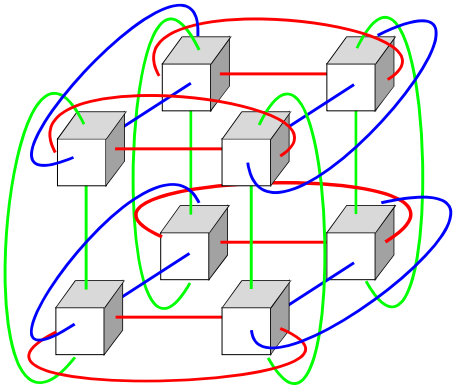

図 2:2x2x2 トーラスネットワーク (Wikipedia)

MPI 並列化に必要な素材について。

並列化のイメージが出来れば、各 CPU が担当する計算部分に必要なクォーク伝播関数の配列を分配し、個々の計算ノードを並列的に計算させることになります。

今回の依頼に限ると、MPI 並列化に際し必要となる関数は

- プログラムの初期化・終了関数

  MPI_INIT()　　：MPI の起動

  MPI_COMM_RANK()：自身のプロセスランクを取得

  MPI_COMM_SIZE()：全プロセス数を取得

  MPI_BARRIER()　　：バリア同期

  MPI_FINALIZE()　　：MPI の終了

- 並列計算を行う全ノードに同じデータを送信するブロードキャスト通信

  MPI_SCATTER()　　：全てのノードにデータを送信

  MPI_GATHER()　　：全てのノードからデータを取得

- 特定の１対１のノード間における通信

  MPI_SEND()　　：送信

  MPI_RECV()　　：受信

- 全てのノード間での集団通信

  MPI_ALLGATHER()：全てのノードにある変数データの共通化

  MPI_ALLREDUCE()：全てのノードにある変数データの和をとり共通化

であり、以上の関数の使用法を理解すれば、基本的な MPI 並列プログラムを作成することができると考えられます。

具体例：

今回の依頼については、全格子空間を１つの計算ノードに任せていたプログラムを上記のイメージのように分割し、担当する計算ノード毎に並列化しただけなので、プログラム開発についてはそれ程難しくは無いと想像できます。

上記の例に従って考えると、全体が

(Xsites, Ysites, Zsites) = (8, 8, 8)

の格子を空間の３次元の軸に対して

(Xnodes, Ynodes, Znodes) = (2, 2, 2)

のように分割を設定すします。この際、クォーク伝播関数 $Q^{-1}(x, y, z, t; t_0)$ も、各計算

ノードが担当する部分 $Q_i^{-1}(i=0,\cdots,7)$ に分解します。それぞれにクォーク伝播関数の利用部分を分配し計算をすることになります。

この際、各計算ノードが担当する部分は全プロセス数と自身のランク(Rank)から、例えば、以下のルールによって決める事ができます。

Rank=Xc+Xnodes*(Yc+Ynodes*Zc)

上記で例とした 2x2x2 トーラスネットワークの場合は

Rank	0	1	2	3	4	5	6	7
Xc	0	1	0	1	0	1	0	1
Yc	0	0	1	1	0	0	1	1
Zc	0	0	0	0	1	1	1	1

となり、決めたルールに従って分割されたクォーク伝播関数を分配する事になります。

並列計算の終了後に、各ノードで計算した値を元の配列に戻す事で計算終了となります。

畳み込み計算：

次の段階では、この前の段階で準備した量に対して、 $\vec{r}$ だけ離れた２つの積の前空間の和 $\sum_{\vec{x}} \tilde{B}_1(\vec{x}+\vec{r})\tilde{B}_2(\vec{x})$ を計算する必要があり、ここで計算ノード間の通信が重要になります。

また、この量は畳み込みになっている事がわかり、これを計算する際に直接定義式を用いると計算量は $O(n^2)$ となる。しかし、一旦 DFT をしてから掛け算を行い、IDFT を行い戻す方法もあります。この場合、DFT の高速アルゴリズムである FFT を用いることで計算量が $O(n\log n)$ で済むことが分かります。

今回求めたい量のフーリエ変換を見ると、

$\sum_{\vec{x}} \tilde{B}_1(\vec{x}+\vec{r})\tilde{B}_2(\vec{x}) \Rightarrow^{FT} \sum_{\vec{q}} \tilde{B}_1(\vec{q})\tilde{B}_2(-\vec{q})e^{-i\vec{q}\cdot\vec{r}}$

で定義され、運動量空間での $\tilde{B}_i(\vec{q})$

$\tilde{B}(\vec{x},t-t_0) \Rightarrow^{forward} \tilde{B}(\vec{q},t-t_0) \quad \tilde{B}(\vec{x},t-t_0) \Rightarrow^{backward} \tilde{B}(-\vec{q},t-t_0)$

を求めた後に、フーリエ逆変換を実行する事で目的の量が得られることが分かります。この方法を用いることにより、計算の高速化が可能になると考えれますので、参考にしてください。

KEK Blue Gene/Q での並列計算について：

　この点に関して、KEK-SystemB のスーパーコンピューター内部のネットワークを、現在問題とするサイズへ変換する必要があります。詳しくは

https://scwwwb.kek.jp/index.html

をご覧下さい。