PR0683

準汎用計算機 GRAPE-DR による DFT 計算

（理研[a]　名大院情報[b] 国立天文台[c]）○松原裕樹[a] 安田耕二[b]　戎崎俊一[a]　牧野淳一郎[c]

【はじめに】

電子状態計算は物理、化学分野で非常に重要な位置を占めるようになった。計算機の高速化とアルゴリズムの発展により、最近では生体分子までもが電子状態計算の対象になり、今後大規模計算の需要は益々拡大すると思われる。今日大規模計算は、多数の一般商用 PC を多数ネットワーク結合したクラスタで行われる。これが最良の価格性能比を与えると信じられているからである。しかし科学技術計算のみを対象とするなら、価格性能比がより優れた、準汎用計算機を設計できる。我々も準汎用機 GRAPE-DR を開発しているが、これは汎用性を多少犠牲にして、単一チップに多数のプロセッサを集積する事で、ある種の計算を極めて高速に実行できる。加速ボードとして提供されるこの準汎用機のピーク性能は 1TFLOPS に達する。今後の科学技術計算では、この種の計算機が使われるだろうが、独特のアーキテクチャーに最適なアルゴリズムを、新規に考案する必要がある。本研究では GRAPE-DR で密度汎関数(DFT)計算を実現するアルゴリズムを開発し、電子状態計算を誰でも超高速に実行できる事を目的とする。

【汎用計算機の現状と課題】

プロセスルールの微細化や、多段パイプラインアーチテクチャにより、より多くの素子をより高速動作させる事が可能になり、その結果 CPU 速度は 2 年で 2 倍（ムーアの法則）という指数関数的発展を遂げている。これに対して主記憶(DRAM)の速度は大幅に遅く、その進歩も遅いため、高速な CPU にデータを十分供給できない状況になっている。そのため、通常 CPU 内部に高速メモリー(cache)を置き、頻繁に使うコード/データを保存するが、現在これがチップ面積のかなり(90%)を占める。又パイプラインハザードを回避する複雑な制御回路も必要である。これらは多種類のプログラムを高速実行するために、汎用プロセッサには必須の機能である。これに対し戎崎や牧野らは、天体や古典 MD シミュレーションを高速に行える GRAPE 型専用計算機を開発してきた。つまり古典シミュレーションで最も時間のかかる 2 体力計算を、専用回路で行い高速化した。この方法は極めて高速だが汎用性に乏しい事が欠点であった。

【準汎用計算機 GRAPE-DR】

科学技術計算では、プログラムの特定箇所が計算時間の殆どを占め、また演算量に比べて通信量が少ない例がかなりある（古典 MD の 2 体力計算や密行列の乗算等）。この場合 CPU に cache や制御機構は殆ど不要である。そこで GRAPE-DR プロジェクトでは、cache などの替わりに多数の演算器をチップ上に集積し、科学技術計算を高速に行える、プログラム可能な専用計算機を開発している。GRAPE-DR チップは、内部に 512 個の単純なプロセッサ (Processor Element, PE) を持つ並列計算機である（図 1)。各 PE は浮動小数の加算器と乗算器、256 語の局所メモリー、33 語のレジスタで構成される（図 2, 1 語は 72 bit)。ホスト計算機が与えた単一の命令流が、各 PE の動作を指定して、汎用性を実現する。つまり、異なるデータに対して同じ動作をする SIMD 型並列計算機である。一般にこのアーキテクチャーでは、各 PE に異なるデータを供給する必要があるため、外部メモリーとの通信速度がボトルネックになる。ところが科学技術計算では、演算量に比べて通信量が少ない例がかなりある。アルゴリズムを工夫してデータ供給速度を小さくできる問題には、GRAPE-DR は非常に強力な並列プロセッサとなる。

現在開発中の GRAPE-DR カードは動作クロック 500MHz、チップあたり 512 PE で、4 チップ版ボードのピーク性能は 1TFLOPS である。この加速カードをホスト PC の PCI バスに挿す事で、一般の研究室でも TFLOPS 級の性能が手に入る。またプロジェクト全体[1]では、この加速カードを搭載した Linux cluster を超高速ネットワークで結合し、2008 年の時点で 2 PFLOPS を超える計算能力を持つシステムを完成させる予定である。

【SCF アルゴリズム】

あるプログラムを GRAPE-DR 計算機で高速実行するには、(i)細粒度で等質に数千 PE に並列化し (ii) 通信量を計算量の 1/10～1/100 に抑え (iii)中間結果（作業領域）を 256 語に抑える必要が有る。汎用プロセッサ用のソースコードから、この 3 条件を満たす GRAPE-DR の機械語を自動生成するコンパイラは､現在の技術では実現できそうに無い(2 体力計算等の定型プログラムはコンパイル可能)。そこでこの 3 条件を満たすようアルゴリズムを再設計する必要が有る。また各 PE は単純化のため構

造ハザード（資源競合）があり、パイプラインを埋めた最適化コードを自動生成する事は現在難しい。つまりアセンブラコードを手作業で最適化する必要がある。

幸い密度汎関数法では、Fock 行列中の近接 Coulomb、交換、相関項の計算と、Fock 行列対角化が、計算時間の殆ど(95%以上)を占めるので、この部分を再設計すれば良い。使い勝手を考え、ホスト上では Gaussian03 を実行し、これらの部分だけ加速ボードで行う事とした。近接 Coulomb 項は Hermite Gauss 基底と Rys 求積法を用いた direct J 法で、Hartree-Fock 交換項は Saunders 法や DRK 法で計算する。交換相関項では、3 次元 grid 上での電子密度生成(GenRho)と、grid 上の potential 値から Fock 行列の生成（DgstEM)のみ加速ボードで行う。これにより Gaussian03 にある任意の汎関数をサポートする。Fock 行列対角化は McWeeny の密度行列最小化で行うが、ここで最も時間がかかる行列積のみ加速ボードで行う事とした。

基底関数や grid は沢山あるので、細粒度並列化は問題無くできる。通信量を減らすため近接 Coulomb 項計算では次の方法を使う。各 PE は自分担当の shell pair P と、全 PE に一斉放送された共通の shell pair Q に対して、2 電子積分を並列に計算する。その後一斉放送された密度行列値を掛け、局所メモリー上の Fock に加算する。全ての Q shell pairs を送ったら、積算した Fock 行列をホストに回収する。この場合積分 screening の効率が下がるため、計算時間は 2〜5 割増えるが、SIMD プロセッサではこの程度の増加はやむを得ない。2 電子積分計算では、演算量に比べて通信量はかなり少ないため、データ供給速度はボトルネックにならない。例えば(dd|dd)型では、放送すべき Q shell pair のデータは 39 語で、積分計算には 2 万〜3 万演算が必要である。交換項や行列積でも、一斉放送を使い通信量を減らした並列アルゴリズムを考案した。

各 PE の作業領域（256 語）は、近接 Coulomb 項に対しては十分ある。Primitive Hermite Gauss 基底と Rys 求積法を使えるため、f 関数までは高い効率で、g 関数は多少効率が下がるが計算可能である。指数関数や Rys 多項式の根や重みなどの補間表は全 PE で共通なので、最適化し共通メモリー(BM)に置く。256 語の作業領域は、交換相関項や行列積に対しても十分である。他方 Hartree-Fock 交換項では短縮 Gauss 基底で計算する事が望ましいが、高角運動量では作業領域が不足する。そこで我々は基底関数の型と短縮度に応じて、最適なアルゴリズムで計算する。つまり s 関数を複数含む時、中間積分は局所メモリーに格納可能なので、普通の DRK 法を使う。他方 d 型関数を多数含む時は primitive Gauss 関数で計算すれば良い。

要するに準汎用機 GRAPE-DR 上で DFT 計算を高速実行可能である。力の計算や時間依存 DFT も実行可能である。これまでに我々は、近接 Coulomb 項と純粋 DFT での交換相関項を計算するアセンブラコードを作成し、チップシミュレータ上でテストした。更に Gaussian03 から加速ボードにデータを送るインターフェースを作成した。又 GRAPE-DR チップは試作を終え、ボードを試作しテストを行っている。順調に行けばこの秋には試作ボードで DFT 計算が行える予定なので、当日はその結果を報告したい。

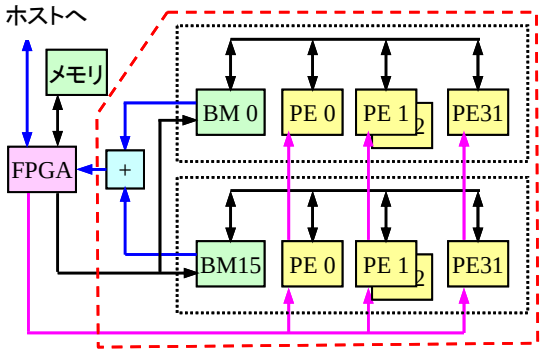

図 1：GRAPE-DR チップの構造。チップ（赤破線）は 16 個の放送ブロック（BB, 緑点線）からなる。各 BB は 1024 語の共通メモリ(BM)と 32 個の PE からなる。FPGA はホスト計算機から命令とデータを受け取り、チップに供給する。全 PE は同じコードを実行する。

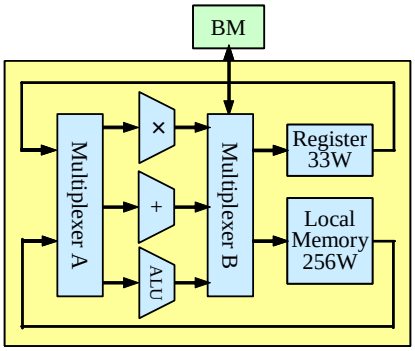

図 2：Processor element (PE)の構造。浮動小数加算器、乗算器、256 語の局所メモリ、33 語のレジスタを持つ。4 段パイプライン構成で、命令は 2 個の multiplexer を制御して計算を行う。倍精度加算、乗算のスループットは 1, 2 cycles である。

[1] GRAPE-DR プロジェクト，http://grape-dr.adm.s.u-tokyo.ac.jp/