シミュレーション技術を取り巻く技術の変遷

技術解説

岡　　秀　樹*

Simulation technology change based on high performance computing

Key Words：simulation technology HPC

１．はじめに

筆者はHigh Performance Computing (HPC) によるシミュレーション技術に関して、ハード、ソフトの両面で内外の計算機メーカー（富士通、CRAY、シリコングラフィックス社）やCADベンダーに在職して、関わってきたが、この間のシミュレーション技術の変遷、及びHPCに関係する環境の変化について述べる事にする。

２．大型機（汎用機）の時代[1)] １９７０年代

カード、バッチ処理

筆者が富士通（株）に入社した当時の1970年代、System360の成功により、IBMの地位はゆるぎないものとなり、日本はIBMのフィーチャーシステム対抗する為に、共同組合を作った。筆者が計算を始めたのは富士通独自のFACOM230-35であった。その後、計算の主力は仮想記憶装置を初めて用いた230-58、後にIBM互換機Mシリーズを日立と共同開発したM380へと変わった。システムディスクを自分でインストールする時間占有のバッチ処理であった。高速計算機は日本初のベクトル計算機230-75/85が航空宇宙技術研究所にCRAY1に遅れること１年で納入されていた。230-75の前面パネルに並べられたランプの点滅はいかにも高速計算している風で憧れであった。シミュレーション計算は１次元から２次元の計算に移りつつあった。

TSS端末、インタラクティブ処理、グラフィックス端末

カードでバッチ処理から、TSS端末からジョブをサブミットできる様になった。ファイルを扱える様になったが、エディットはバッチ処理であった。計算は２次元での計算が本格化してきたが、まだ変数は２変数止まりだった。

２次元計算の結果、可視化が重要な課題となった。それまではペンプロットあるいはラインプリンタで、文字でグラフを書いていた。PLOT10-GKSの時代だった。グラフィック端末が出来、画面上でグラフを書ける様になった。Tektronixの高価なグラフィックス端末の代わりに安価なグラフィックス端末が利用できる様になった。この時の感激は、何物にも代えがたいものだった。

グラフィックスはその後ステラ、タイタンといった３次元グラフィックスワークステーションを経て、CG技術の進展と共に、PHIGS-OpenGLへと変わり、シリコングラフィックス社に引き継がれていった。

超LSI

半導体はこの時期、バイポーラトランジスタを用いて、IC（集積回路）を構成していた。RAMメモリでN型MOSが使われる様になり、それがロジック回路にも応用された。まだ集積度は高くなく、16Kbitが普及し、64Kbitは超LSIと呼ばれ、電電公社は超LSIプロジェクト、通商産業省は超LSI組合を作って開発を行った。

*Hideki OKA
1952年1月生
東京大学理学系研究科相関理化学（1977年）
現在、大阪大学　臨床医工学融合研究教育センター　特任教授　博士　ＨＰＣシミュレーション技術
TEL：06-6850-6531
FAX：06-6850-6530
E-mail：oka@bpe.es.osaka-u.ac.jp

3．ワークステーションと大型機と共存の時代　1980年代

ミニコン

大型機の隆盛に続いて、過渡的であったが、ミニコンピュータと呼ばれる計算機が現れた。DECの16bit PDP11が代表的であるが、メモリは64Kバイトと小さく、メモリを節約するプログラミングは制約の多いものだった。用途は露光用のマスクデータ処理、CAD専用機であった。このPDP11及び次の32bit VAX、そのOSのVMSはUNIXの普及に貢献した。ミニコンはマイクロプロセッサの台頭と共にワークステーションに取って代わられることになり、その後DEC社は消滅した。日本においてDEC社は計算機業界の優れた人材を輩出した。

ワークステーション

ワークステーションと呼ばれるUNIXマシンが台頭してきた、アポロが最初であったが、特にサンマイクロシステムズのSUN3の台数は、台頭著しい半導体のケイデンス社のECADとセットで販売した為、台数が一気に増大した。アポロもメンター社、機械系CADのプラットフォームで台数を伸ばした。SUNに代表されるシリコンバレーの台頭は世代交代を強く印象付けた。文化はIBM、DECに代表される東海岸、CDC、CRAYに代表される中部から、西海岸へと移ったのである。インテル、SUN、HP、Apple、SGIといった新興勢力が揃うシリコンバレーは活気を呈していた。

今やSUNにも過去の勢いは無く、Googleに代表される新興勢力が台頭しているのは時代の流れだろうか。

X Windowの出現

ワークステーションが普及したのはX Windowの出現が大きく貢献した。X Windowを扱える処理能力がプロセッサに要求された。大型機のX端末としての利用はプログラミングを格段に容易にした。

スーパーコンピュータの時代

何を持ってスーパーコンピュータとするのだろうか。アーキテクチャには依存しないだろうが、狭義ではベクトル型の技術計算専用コンピュータをスパコンと呼んだのが由来だろう。その時代の最高速の計算機をスーパーコンピュータと呼ぶのだろう。

筆者はCRAY Research社の日本法人に在籍していたが、今思えばこの時代は、計算機は最高に面白い激動の時代だった。カリスマ的な設計者セイモア・クレイがCDC社からCRAY社を興した話は伝説的な話として残っているが、最初のベクター計算機CRAY1からの後継機種はCRAY2、3への系列とX-MP、Y-MP、C90、T90への系列では異なった。クレイ自身は実装技術を生かす前者を選んだ。NTTに導入された液浸冷却方式の円筒形のCRAY2は魅力的であった。CRAY初のマルチプロセッサX-MPはスター設計者スチーブ・チェンが開発したものであるが、ベストセラーとなり、Y-MP、C90、T90へと進化した。一方クレイもCRAYコンピュータ社を設立して、CRAY3、4の開発を継続した。GaAsといった材料がまだ夢で語られていた時代である。一方同じCDCから出たETAシステムズ社は窒素温度冷却でのスパコンを開発して、これは東工大に納められた。こちらは必ずしも成功したマシンとは言えなかった。この時代、計算は3次元での計算があたり前となり、アーキテクチャ、実装技術、半導体技術がバランス良く調和して寄与していた時代だった。

日米摩擦

スパコンは日米摩擦のシンボル的な役割を果たした。当時米国の圧力の元で、自動車会社を始めとする産業界、官公庁に米国製品購入の圧力をかけた事は事実である。自動車会社がCRAYを導入したのは、米国輸出の代償であった事もあるが、CRAYでしか動作しなかったアプリケーションが寄与したのは事実である。逆に言えば日本が優れたアプリケーションを持たなかった事が致命的となった。得意な半導体技術を用いてベクトルレジスタの容量、ベクトルパイプラインの本数を増やした日本の重量級のスパコンにCRAYが苦戦を強いられた為、ソフトウェアの国産マシンへの移植を阻止して対抗した事も事実である。国産が性能差で非難をしても、過中にいた者からすれば、基本的アーキテクチャをCRAYに依存している限り、技術者として尊敬は得られなかった。その後、国内メーカーは欧米にアプリケーション開発、移植を行うセンターを設置した。

CRAYが独自の会社を設立して製造した時点で、半導体技術をバックに持つ日本の大会社に対抗することは困難であった。スパコンの様な統合技術は巨大な会社でしかもう製造できなくなってきたのである。その意味でもアーキテクチャの時代から半導体の時代へと時代は移ったのである。

スパコンの利用目的

軍事利用が無い日本でスパコンを牽引したのは、科学技術庁が中心となって、時代の順に①原子力（構造解析）（原研、動燃）、②宇宙、航空機（流体）（航空宇宙技術研究所、数値風洞）、③気象予測（流体）（海洋研、地球シミュレータ）、そして現在の④ナノバイオ技術（次世代計算機）を主たる応用分野とした。技術のピークは航技研の数値風洞であろう。この分野に投資される資金で特に①の時代に、欧米のアプリケーションを扱う会社、シンクタンク系のソフトウェア開発会社、あるいは計算機メーカーでの科学技術をサポートする部門が数多く設立され、計算技術の裾野を形成した。①、②の世代は現役を過ぎ、③は過渡期で④の世代はもう新しい世代が牽引すべきだろう。当時のスパコンの研究者はGrid、クラウドの研究に移っていった。

ベクトル化

ベクトル計算機向けのプログラミングはベクトル化というコードの書き直しが必要で（DOループの内側を長くする）様々なテクニックが必要とされたが、ある程度自動化ができる所でもあり、自動ベクトル化のコンパイラが利用できた。ベクトル化が旨く行った時の快感は、何物にも代えがたいものであった。今でも差分での流体計算といった計算では高いメモリバンド幅を活かしたベクトル計算機が有利である事は明らかである。

しかし、アーキテクチャに依存したプログラミングは避けなければならないと思う。一度依存したプログラムを作成すると、アーキテクチャの異なる機種に移行するのが大変になる。それを逆手にメーカーサイドで他機種に移行させない様な事もできたのである。ベクターとスカラの並列マシンではプログラムの構造そのものが異なる。移植性の観点からはアーキテクチャ依存は排除すべきである。

CRAYで感じた事

米国の各地に存在するスーパーコンピュータセンターを訪れたが、これらのセンターは地域に根付いたものであり、産業も含めた地域の活動、教育に貢献している。日本の計算センターはどうであろうか。CRAYで驚いたのはドキュメントの多さである。開発したマシンのHW/SWは正確にドキュメント化されて教育資料もWORKBOOKとして充実していた。CRAY1のマニュアルは今でも見る事ができる[2)]。教育という姿勢が日本とは異なるとの思いを持つ。

CMOSの時代

この時期、バイポーラトランジスタはますます複雑な構造を呈してきて、高速計算機にはECL回路を用いた高速な論理回路が用いられる様になったが、バイポーラは縦方向、横方向に寸法を取る3次元構造を持つ事と大量の電流を消費する為に、発熱の問題と集積度が上がらない事で、これに置き換わるトランジスタが望まれた。MOSトランジスタがそれに置き換わった。移動度の高いN型MOSは平面的な構造を持ち、集積度は高いもののやはり電力を食うことで、P型MOSと組み合わせてスイッチング時だけに電流が流れる相補型MOS（CMOS）が主流となっていった。CMOSは、面積は食うが、スケーリング則と呼ばれる法則により微細化を図れば、速度と集積度が同時に達成できた。露光技術を含めた微細加工技術の進展がCMOSの発展に大きな寄与をした。高速化と集積化は計算機の世界を変えることになるのである。

マイクロプロセッサ

マイクロプロセッサのビット数は8bitから32bitまで上がり、CISCプロセッサに代わりRISCプロセッサと呼ばれる命令を短縮して高速化を図ったプロセッサが登場してきた。CPUのクロック周波数もMHzからGHzへ迫る周波数へと向上した。これには微細化を図ったSRAMによる大容量キャッシュメモリの寄与が大きかった。集積度の向上により、大型機の機能をチップ上で実現できる様になってきた。

様々な試みの計算機

Si以外の材料が検討された時代があった。Siより

移動度の高い化合物半導体、特にGaAsは実際に計算機で採用された。またHe温度で動作するジョセフソン素子を用いた計算機もIBM、NTT、富士通で検討された。いずれも素子の安定性、集積度の点で消えていった。ETAシステムズ社は低温、窒素温度での動作を実現した。低温では移動度が向上する為である。結局窒素温度を維持する装置のコストが問題となった。CRAY3のGaAsを含めて可能性を追求した時代だった。
この流れは現在の量子コンピュータやナノテクノロジ、例えばCNTやフラーレンといったデバイスに受け継がれているが、集積度と信頼性の点でSiを置き換える事は簡単ではない。

超並列マシン（MPP）
超並列マシンが隆盛した時期があった。元来MITのプロジェクトをコネクションマシンとして実現したのが端緒であるが、後にThinking Machine社を設立して製品化した。今のクラスターとは思想が異なり、性能というより超並列のコンセプトが重要だった。AI、記号処理、データ検索を目的としていた。商用ベースで分散メモリに限定するとThinking MachineのCM5、IntelのParagon、iWarp、nCUBE、KSR、MasPar、アライアント、国産もAP3000、Cenju、ADENART、チップではTransputer、IPSC/860といった特徴ある並列計算機が出てきた。各機種で計算ノードを繋ぐネットワークは異なり、各々得意な応用分野があった。今は全て無くなってしまった。その原因は、計算機としての未成熟さによる低信頼性と並列処理の使い勝手の悪さが原因であった。一方で、この当時、半導体技術の進歩は目覚ましく、単一CPUの性能はムーアの法則により毎年2倍向上するのに反し、超並列マシンはその設計を始める段階で、一世代前のCPUからスタートする事から、マシンが完成する頃には、その時期のCPUはすでに以前の数CPUの性能を簡単に出すことができ、面倒な並列処理のプログラミングを始めで性能を上げるより、最新のCPUを待った方が高い性能を単一のプログラムで得られたのである。
筆者はCRAY社で、CRAY初のマイクロプロセッサを用いた超並列計算機CRAY-T3D[3)]のプロジェクトに参加した。当時クロック数が最速のDECのアルファチップを改良したプロセッサを用い、ネットワークに3次元トーラス構造といったノード間の距離が最少になるネットワークを採用し、ネットワークのlatencyを短縮した結果、高速な並列性能が得られた。日本でも3台納入した。しかしながら、納入先の既存のプログラムのMPIを用いた並列マシンへの移植には大変苦労した。最初から並列性を意識したプログラムにしておかないと、並列性は十分に生かせない。特に企業ではこの様なスキルを持つエンジニアを抱えることは困難であることから、ベクター以上にプログラミングのサポートは欠かせなかった。並列処理マシンの普及には未だ問題があった。その後、プロセッサの改良によりT3Dは自立型のT3Eへと進化した。T3EはJAISTに納入された。

共有メモリ
共有メモリはメモリバンクとCPUを繋ぐネットワークの複雑さ、またHWでの同期を取る困難さの故にCRAYでT90-32CPUで限界となった。この当時のCRAYのディレクティブ挿入によるマクロタスキング、マイクロタスキングによる並列化プログラミングスタイルは、OpenMPとなって現在に受け継がれている。このディレクティブ挿入はオートタスキングである程度自動化できた。完全な自動並列化はやはり現在でも実用化には遠い。この後、前述の数値風洞の様に共有メモリをクラスター化したマシンが現れる事になった。

分散共有メモリ
CRAYがシリコングラフィックス社（SGI）に買収され、SGIに移ったが、SGIもOriginという計算サーバーを作った。これはccNUMAという、分散メモリであるが、共有メモリとしてのアドレスを持つマシンでありプログラミング上は使い易いサーバーだった。T3DもNUMAマシンであるが、キャッシュコヒーレンシは取っていなかった。

計算アルゴリズムを固定すれば、並列化をおこなう事は可能である。演算器を多数並べて特定の用途での並列処理を可能にする事は以前から行なわれている。グラフィックスエンジンやFFTを用いた信号処理専用機とかはその例である。またGRAPEやMD-GRAPE、QCD-PAXの様に、用途を限定して、

そのアルゴリズムに合わせたネットワーク、プログラムを用意すれば、安価に高い性能を出すことも可能である。現在のFPGAやGPGPUも同様であるが、用途は限定される。安くて、性能が出て、使いやすくて、何にでも使える、という計算機は無い。使い易ければ高い、安ければ使いにくい、というのが常識である。高いものは信頼性やネットワークにコストをかけている。昨今安価なクラスターが入手できる様になった。数千万のシステムでTFlopsの性能は入手できる。只、信頼性に関して言えば、価格に見合ったものでしかない。目的に合わせて選択すべきだろう。
現在、プロセッサの処理能力とネットワークの通信能力の差が乖離して来た為、性能を出す為に通信を抑える事に腐心しているが、本来通信を活かすのがMPPの目的だった筈で、ネットワークに見合ったプロセッサを用意してMPPの特徴を生かしたバランスの取れたマシンを構成すべきと筆者は思う。

４．PCの時代　1990年代～

RISCプロセッサの乱立からワークステーションメーカーは潰れていった。それは多数存在する必然性が無いからである。各メーカーの異なるUNIXは混乱を招いた。SUN、IBM、DEC、HP、CRAYといった多くのプラットフォームにアプリケーションを対応させるのはアプリケーションベンダーには重荷であった。その対応には順番があり、台数の多いSUNは、性能は遅いもののマーケットの台数が多い故に、最初のプラットフォームとなる点で有利であった。性能の良悪ではなく台数が、また利用できるアプリケーションの数が勝負を決めた。
オープンな結果、乱立を招いた業界は、OSはLinux、プロセッサはインテル、AMDに収束されてきた。PCの普及と共にWindows上でも計算が可能になってきた。技術は巨大化し、Microsoft、Intelといった巨大企業に集中していった。寡占化、統一化の時代にまたなったのである。

RISCプロセッサの淘汰

RISCプロセッサは元来シンプルさが特徴だった筈が、高速化、効率化の為に、集積化が上がる事を利用してスーパースケーラ、スーパーパイプライン、投機実行、分岐予測といった様々な大型機の機構を取り込んで肥大化していった。結果として面積を食う割に効果は上がらない事が判った。そこで１コアをシンプルにして、集積度を活かしてマルチコアにする様になった。インテルのCISCプロセッサは内部で命令をRISCに変換し、RISCプロセッサと遜色ない性能を出す様になり、他のRISCプロセッサを追いやった。

半導体の行き詰まり

半導体は最小寸法がミクロンからnmの世代になってきた。光露光の限界に近付いたのである。物理的な動作限界（スイッチングエネルギー、論理を構成する最小電圧）工学的な限界（加工ばらつき）、発熱、冷却能力の限界、材料特性の限界、製造コストの限界が見えてきた。製造装置は高額になり、計算機メーカーは単独で半導体生産工場を持つことはできなくなった。設備投資を継続することができる会社は巨大企業、あるいは製造専用の会社のみとなり、企業の分業化、集中化、寡占化が始まった。
高速化、集積化の限界に直面した。集積度の面では、特性ばらつきの低減、信頼性の確保が最大の課題となってきた。高速化は、デバイスの性能はもう向上せず、微細化の恩恵を受けることができなくなってきて、頭打ちになった。
集積化は平面的な２次元は限界であり、３次元的な集積化が研究段階で検討されているが、加工技術は一層困難になり、益々製造できる会社は限られるだろう。今後システムを載せることはできる様になるかもしれないが、冗長性やフォールトトレラント性を持たさないと信頼性、歩留まりが確保できなくなるだろう。
半導体が性能を上げる時代は終わりを告げ、再度プロセッサを集積する並列マシン（クラスター）に注目せざるを得なくなったのである。アーキテクチャ、半導体と移り、利用技術の時代へと移ってゆくのである。

企業の製品の変化

企業の製品も変わった。利益を生み出す製品は、PCからゲーム機、携帯電話を含むモバイル商品へと移行してきた。プロセッサメーカーは組み込みプロセッサへとシフトした。ユーザインターフェース

が重要なテーマとなった。ソフト、システムの時代への移行である。半導体は高機能、低電力、ワンチップ化によるコスト削減がテーマとなった。かつてのスパコンは撤退が相次ぎ、UNIX サーバーに代わった。重厚長大な製品は製品も開発者も影を潜めた。次世代計算機で開発辞退を引き起こしたのも企業環境の変化と無縁では無く、資金というよりも利益を生み出す分野に人的リソースを集中したい、という事なのだろう。

５．今後の展開　利用技術の時代

プログラミング

プログラミングも以前は FORTRAN 一辺倒だったが、マイクロプロセッサの隆盛とともに低レベルでのプログラミングが可能な C 言語での計算が現れ、オブジェクト指向の流れから C++ が出てきた。数値計算はオブジェクト指向を取る必要も無く、配列処理に向いた FORTRAN が技術計算ライブラリの利用や最適化、並列化の容易さからも最も向いている事は変わらない。分散並列では、最新の FORTRAN 2008 では並列処理向けに CAF(coarray fortran) が規格化されており、G95 でも利用可である。C/C++ は UPC（Unified parallel C)、また X10 や Chapel といった並列処理言語も IBM/CRAY で使われている [4]。早く標準化されて MPI や OpenMP で山の様な API を書くことを避ける事が今後必要である。

インターネットの普及に伴い、システムプログラミングの人口が増え、プラットフォーム非依存の Java や Python、Perl、Ruby とかスクリプト言語系も現れた。一方でもっと優しい MATLAB、Maple の様なツールを用いる人も増えている。システムプログラミングと技術計算の世代の gap が広がった事は好ましい傾向ではないが、最近の言語でも例えば mpiJava、pyMPI といった様に FORTRAN/C/C++ と同じ機能を実現しようとする試みは見られる。

アカデミックでは難しい言語、新しい言語を好むが、標準化されたものを使ってゆく事が必要と思う。プログラミングが個人のスキルに強く依存する事は排除しないといけない。CRAY にいた時は異能というべきプラグラミングに長けたプログラマーがいた。彼らにかかると魔法の様にプログラムが早くなった。今は誰もが一定水準のプログラムを作成できる様な環境が求められている。

ミドルウェアの必要性

研究者は最適化や並列化には元来興味が無いし割く時間も無い。自分の研究の目的に計算したいことを簡単に実行してくれるものが欲しい。しかし計算したい人と計算する人が別れていいのかと言うと、計算は実際に計算した人でしか判らない。ここにジレンマがある。計算のテーマを持っている人、計算する技術を持っている人、計算のツールを作る人、可視化が得意な人、これらが集まらねばならない。計算の技術を持っている人の代わりを計算機が肩代わり、あるいは吸収できる様なミドルウェアなりが今後必要になる。

数値精度

これだけメモリが使えて大規模計算が可能になったが、数値精度は倍精度 64bit から向上していない。これは早晩問題になると思う。倍々精度 128bit は必要無いにせよ（ソフトウェア処理なので遅い)、96bit の精度は必要になると思われる。ソルバーの精度確保が困難になりつつある。精度が十分であれば、発散や収束が改善される。32bit を 48bit に拡張して、それを倍精度にすればいいのだが、バイトマシンの制約や HW/SW の新規開発の困難さが足枷なのだろう。昔は 36bit のワードマシンもあって IBM より精度が良くて重宝したが、IEEE754 の規格で決定的になった。CRAY の旧数値フォーマットは IEEE より有効桁数が少なく、反復解の収束が悪くベンチマークには不利な場合もあった。数値精度の問題はもっと真剣に議論すべきである。次世代計算機を作るなら、96bit の精度を持つといった他に無い特徴があれば次世代計算機で計算したいという研究者も現れると思う。

計算機の利用形態

１）企業での計算機利用

ほとんどが商用コードの利用になる。そこではツールとシミュレーションの差がある。NASTRAN に代表される構造解析や SPICE に代表される回路解析は設計ツールとなっている。そこで用いる入力パラメータは品質を表現する重要な指標である。一方、

例えば流体解析や衝突解析はあくまでシミュレーション、模擬実験である。長くシミュレーションは正当な評価は受けてこなかった。形状モデル作成に時間がかかり、計算機に載せるまで時間がかかる。シミュレーションに用いるパラメータの決定、精度確保にもユーザ毎の予備実験が必要である。この様な地道な基礎実験に時間を割く余裕は無いのが実情である。自動車の様に設計期間が長くて高価なものだと実機での検証をシミュレーションに肩代わりさせる事は意味を持つ。電子機器の様に設計時間が短くて、試作品を短時間で作成出来るものは、実機での試験で済ませることが多い事になる。シミュレーションをいかにしてツール化できるかが最大の課題である。

過去の設計データの蓄積が重要とされ、計算精度の維持が必要であり、新しい計算機の導入には保守的である。計算精度が並列 CPU 数に依存する様な並列処理計算の導入は難しい。単一 CPU で高速な計算機が求められ、マルチプロセッサはマルチジョブを処理できるシステムのスループット性能が求められる。

2）アカデミックでの利用

ここでは利用分野が多岐にわたるのが特徴である。自作コードが多いが、言語は多岐に渡り、研究者は特に日本では公開の意識、制約の多い維持、品質管理、商品化に対する意識が薄いのが実情である。メーカーは計算機センターでのプログラミング支援が必要だった。今はローカルで計算サーバーを立てている所が多くなったが、管理は研究者の負担になっている。プログラミングは個人のスキルに依存し、チームでの作成は稀である。計算対象は企業と比べると新しい現象の解明、仮想的な環境での解析が多い。シミュレーションに対する自由度は多いが、実証、検証が難しい。

アカデミックは企業の品質管理を、企業はアカデミックの新しい計算手法を、アプリケーションベンダーを介在させる事も考えて、企業内に取りこんでゆく試みが必要だろう。

ネットワークへとシフト

プロセッサ開発が米国の独占状態になった事もあり、アーキテクトの関心は計算機内外のネットワークへと移った。共有メモリマルチプロセッサが、チップ内で実現されてマルチコアとなり、計算ノード、更にはノードを繋ぐネットワークの設計が関心事となった。特にノード数が巨大になるとこの設計は性能を決める決定的な要因となる。

計算機外では、超高速ネットワークを介した Grid と呼ばれる複数の計算機を利用する仕組みに取り組んでおり、地域を超えた有機的かつ透過的なシステム構築が考えられている。かつてのスパコンの研究者は今は Grid 研究に移っている人が多い。Grid はクラウドコンピューティングに包含され様としている。

情報技術との連携

今までは、計算をして結果を可視化するだけであったが、現在は情報化技術との連携が進んでいる。それはバイオの分野で最初に起こり、バイオインフォマティクスと称される DNA の遺伝子情報を解析するヒトゲノムの解析へと結実した。バイオから普及してフルードインフォマティクス[5]、メディカルインフォマティクス、ハイドロインフォマティクスとコンベンショナルな分野にも広がりつつある。

モデル化の限界

シミュレーションが現実的な問題を再現できるかどうかはモデル化に依存するが、モデル化は目的とする現象を単純化、抽象化したものであり、そこには何らかの仮定が含まれる。この仮定はモデル化した人は理解しているかもしれないが、背景は伝わらない。何を落としたのかを理解しておかないと、何を計算しているのかを理解できない。計算が合わないのはどうしてなのか、と真面目に聞かれるが、シミュレーションはモデル化の範囲でしか結果を生まない。非経験的手法があるが、そこにも大きな仮定があることを理解しておかねばならない。また物事は平行して同時に起こっているが、これを計算機に載せることは困難である。計算アルゴリズムのあり方についても検討が必要だろう。

マルチスケール、マルチフィジックス

対象とするスケールでのシミュレーションを結合して、ミクロレベルからマクロレベルの機能を解明し

ようとする試みは最近話題になっている。バイオの世界では、細胞から人体の機能までのマルチスケールなシミュレーションを Physiome と呼んでポストゲノムを推進するテーマの１つとなっている。これは量子力学的なミクロな世界のシミュレーションが計算能力の向上により現実的な時間で計算ができる様になってきた事も背景にある。ミクロな世界は非経験的手法が主流であり、確率論的な世界である。ここからマクロな決定論的な世界での方程式に繋げるインターフェースが重要である。各階層でのフィジックスが異なる為、マルチフィジックスに対応しなければならない。ただミクロなレベル、原子分子レベルの現象が解明すれば全てが解明できるのか、という訳ではないことは理解すべきである。一方でアプリケーションベンダーは連成計算を指向している。これに合わせてベンダーの統合が始まっている。今までスケールを限定してきた理解してきた単一の現象から、幅広いスケール、あるいは複合的な現象として捉えようとする試みへの移行は今後益々重要になってゆくだろう。

統合環境

シミュレーションモデルデータベースの作成、統合、シミュレーションツールの起動、ポスト処理まで一貫した環境を、インフォマティクスを用いて構築する試みがなされている。

我々は insilicoIDE という統合環境を開発中である[6]。マクロな機能レベルのシミュレーションに有限要素法（FEM）を用いているが、ここでは FreeFem++ というツール[7]を用いている。独自のフォーマットを用いて簡単なスクリプトを作成するだけで、シミュレーション形状作成から実行、ポスト処理まで一貫して処理できる。解く方程式は、弱形式を記述するだけで、いかなる種類の方程式も解くことが可能である。バイオという、商用コードの利用頻度が小さく、解析分野が多岐に渡るという特徴に向いている。insilicoIDE から吐きだす insilicoML という XML を用いたモデル記述言語を読み、FreeFem++ のコードを自動生成する様にしてある。ユーザは普通の強形式の方程式を入力するだけで、自動的に弱形式に変換され、計算を実行する。並列処理も MPI ベースで行える。

図１に insilicoIDE による心臓のリエントリ現象をシミュレーションする例を示す。心臓の形状データ[8]と興奮伝播の式、および最初の興奮の位置情報を基に insilicoIDE モデルを作成すると、insilico

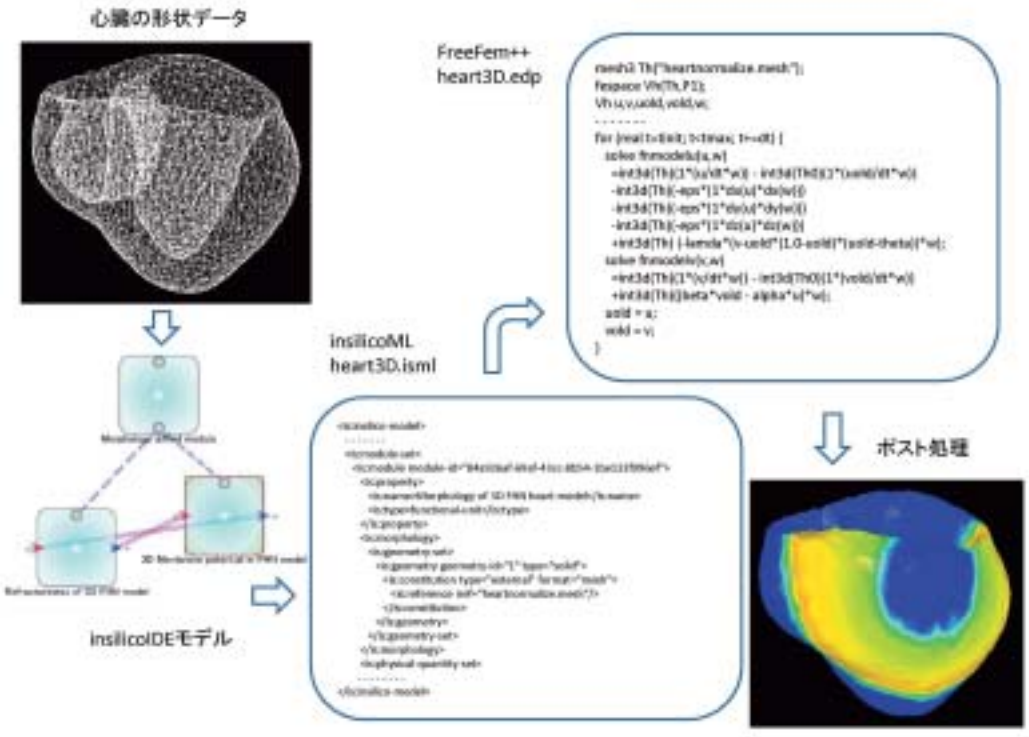

図１．insilicoIDE による心臓リエントリ現象のシミュレーション

IDE は insilicoML ファイルを作成する。これを解釈して FreeFem++ のコードを自動生成する。これを実行すると図に示す様な興奮伝播の動画が得られる。形状データや insilicoML データをデータベース化する事により、統合シミュレーションを容易に行える環境を構築中である[9]。

今後、FreeFem++ の様に、シミュレーションプログラムを一から書く様な事は減り、コードを自動生成する試みや、予め用意されたライブラリ（並列化対応）を組み合わせたり、あるいは MATLAB、SCILAB といったツールを取り込む、といった様にプログラミングの手間を軽減する試みが主流となるだろう。

６．おわりに

筆者が携わってきた HPC をベースとしたシミュレーション環境の変化について述べた。多様化の時代を経て、統合化の時代に移りつつあるが、今後、標準化と共に、より簡単にシミュレーションが行える環境が整備され、シミュレーション技術が我々の生活を豊かにする事に貢献する事を願ってやまない。

７．参考文献

(1) Computer History Museum：
http://www.computerhistory.org/
コンピュータ博物館情報処理学：
http://museum.ipsj.or.jp/
富士通のスパコン設計者内田氏のレビュー
http://www.info.kanagawa-u.ac.jp/~uchida/pdf/SupercomputerDevl&Pros.pdf
(2) http://ed-thelen.org/comp-hist/CRAY-1-HardRefMan/CRAY-1-HRM.html
(3) http://en.wikipedia.org/wiki/Cray_T3D
(4) Cray Japan 三上氏、private communication
(5) 日本機械学会、「フルードインフォマティクス」、技報堂出版 2010
(6) 浅井、「フィジオーム・システムバイオロジーのためのオープンプラットフォーム insilicoML & insilicoIDE」生産と技術 Vol.62 No.1
(7) FreeFem++ manual：http://www.freefem.org
(8) http://www-sop.inria.fr/members/Maxime.Sermesant/gallery.php
(9) http://www.physiome.jp