平成６年７月

新プログラムによる研究の進捗状況報告書

研究課題　:専用並列計算機による　'場の物理」の研究

研究代表者　:筑波大学　･物理学系　･教授　　岩崎洋一

目 次

I 研究概要 **5**

1 研究目的 **5**

2 研究計画の概要 **5**

3 研究組織と研究テーマ **7**

4 計算物理学研究センター設置及びその整備 **9**

5 研究推進方法 **12**
5.1 センター研究員会議及びワーキング ・グループ 12
5.2 研究会開催 14

6 本プロジェクトの背景 **14**
6.1 計算物理学 14
6.2 スーパーコンピュータから超並列計算機へ 15
6.3 並列計算機 QCDPAX 16

II 研究進捗状況 **18**

7 計算物理学からの並列計算機に対する要求性能の検討 **18**
7.1 素粒子物理学における格子量子色力学 18
7.1.1 格子量子色力学の計算内容及び計算方法の概要 18
7.1.2 格子パラメタの選択 19
7.1.3 必要とされる計算能力の検討 20
7.1.4 乱数及び初等関数 25
7.1.5 結論 25
7.2 宇宙物理学からの並列計算機に対する要求性能の検討 26
7.2.1 宇宙流体力学 27
7.2.2 電磁流体力学 28
7.2.3 輻射流体力学 29
7.2.4 結論 29

8 超並列計算機 CP-PACS アーキテクチャ **31**
8.1 超並列計算機 CP-PACS システムの主要目標仕様 31
8.2 CP-PACS の アーキテクチャ 31

9 ノードプロセッサ ・アーキテクチャ **35**
9.1 従来のマイクロプロセッサの問題点 35
9.2 擬似ベクトルプロセッサ PVP-SW 36
9.3 PVP-SW における擬似ベクトル処理例 38

9.4 CP-PACS で採用予定のプロセッサ 40

10 ネットワーク **40**
10.1 データ転送ネットワーク 40
10.2 データの高速転送のための機構 43
10.3 メッセージのブロードキャスト転送 45
10.4 ストライド転送 46
10.5 同期ネットワーク 47

11 システム構成概念図 **47**

12 ソフトウエア **50**
12.1 並列実行 OS 機能 50
12.1.1 PU 群/プロセスに関する機能 50
12.1.2 ファイル関連機能 50
12.2 プロセス間通信機能 51
12.3 プログラミング環境 51
12.4 プログラム言語 51
12.4.1 最適化コンパイラ 52
12.5 プログラム開発 ・デバッグ機能 52
12.5.1 リモートデバッガ 52
12.5.2 トレーサ ・モニタ 52
12.5.3 シミュレータ 52
12.6 簡易コンパイラ 52

13 計算物理学の典型的問題に対する仮想ベンチマークの結果 **55**
13.1 ブロック ・ストライド転送 55
13.2 仮想ベンチマーク QCD MULT の評価結果 57
13.2.1 スライドウィンドウを用いた場合の QCD MULT のノードプロセッサあたり実効速度効率 57
13.2.2 QCD MULT の効率に対するメモリのバンクコンフリクトの影響 58
13.2.3 QCD MULT の効率に対する通信の影響 59
13.3 格子量子色力学シミュレーションに要する計算時間の評価 60
13.3.1 quenched QCD ハドロン質量計算 60
13.3.2 full QCD ハドロン質量計算 62
13.3.3 有限温度 QCD のシュミレーション 62
13.4 KS fermion の場合の実効性能評価 63
13.4.1 QCD MULT の効率 63
13.4.2 KS fermion の場合の QCD シミュレーション計算時間評価 64
13.5 評価 65

14 これまでの研究成果及び反省すべき点 **66**

15 当初計画からの変更点 **67**

III　今後の研究計画 **67**

16 平成 6 年度以降の研究計画 **67**
16.1 平成 6 年度 67
16.2 平成 7 年度 68
16.3 平成 8 年度 69
16.4 平成 9 年度以降 69

17 世界における専用並列計算機の開発状況 **69**

18 今後の目標 **70**

Part I
研究概要

1　研究目的

素粒子物理学、物性物理学、宇宙物理学の基本的重要問題の多くは、物理系が時空上の場によって記述される「場の物理」の問題である。近年の計算機の能力の急速な進歩により、これら「場の物理」の重要な問題において、解析的方法では解くことが困難な問題を数値計算により解決したり、あるいは実験室では実現不可能な条件を数値実験で実現することが可能な状況になってきた。しかし、スーパーコンピューターといえども、計算能力、メモリ容量がいまだ足りず、思うような計算ができない問題が数多く残されている。モデルを簡単化するか、近似を導入するか、計算精度が不十分でも我慢するか、研究者は各々工夫をしているのが現状である。

これらの問題は、いままでは主としてベクトル型汎用スーパーコンピュータを用いて計算されてきたが、この種のスーパーコンピュータの計算能力は技術的限界に近付いている。一方、「場の物理」の問題は並列計算機に非常に向いた問題である。実際、「場の物理」の問題に適した並列計算機のアーキテクチャを採用すれば、物理空間を計算機アレイに写像でき、近接相互作用の特徴を最大限に活用した高速計算が可能となる。さらに、この問題限定により、計算速度のみならずコストパーフォマンスも飛躍的に向上する。

物理研究の基本的手段として、ベクトル型スーパーコンピュータを凌駕する性能をもつ並列計算機を製作するアプローチは、先の QCDPAX の開発によって第一歩が踏み出された。この開発においては、平成 2 年度完成時点で、当時の最高速ベクトル型スーパーコンピュータの約 3 倍の速度を持つ並列計算機を製作することに成功し、それを用いて重要な物理結果を導いている。

本プロジェクトでは、QCDPAX での経験を生かし、「場の物理」の計算に適した超高性能専用並列計算機 CP-PACS (Computational Physics by Parallel Array Computer Systems) を新たに設計・製作し、それを用いて素粒子物理学、物性物理学、宇宙物理学の基本的重要問題に対して、第一原理に即した計算を行なう。これにより、基本法則の検証、新しい質の現象に対する原理的な理解、新しい現象の予言などに関して、従来の計算力不足に起因する曖昧さなしに、明快な結論を導くことを目指し、物理学の重要な難問題に対する飛躍的な進歩を図る。

さらに、この専用並列計算機の研究開発を通じて、並列計算機工学の研究を発展させ、併せて、物理学の数値的方法による研究を推進する上で欠くことのできない物理学と計算機工学の共同研究の基盤を確立する。

2　研究計画の概要

本計画は、(1) 「場の物理」の為の超高性能専用並列計算機 CP-PACS の設計・製作と、(2) それを用いた素粒子物理学、物性物理学、宇宙物理学の重要難問題の研究とからなる。

(1) 専用並列計算機の設計・製作に当たっては、物理学研究者と計算機工学者の緊密な共同研究を行ない、理論的ピークスピード 600 GFLOPS (1 秒間に 6 千億回の浮動小数点演算) 以上の計算性能の実現を図る。専用並列計算機として最も重要なことは理論的ピークスピードよりもむしろ「場の物理」の計算に対して実行スピードが充分高速なことである。この目的の為に、物理学研究者は、本研究で解決を図る物理学の諸問題に於いて重要な計算の典型的

なプログラムを分析・提示する。計算機工学者は、技術的に実現可能な並列計算機の演算ユニット（演算素子、記憶素子等）の構成とその結合方法等についての検討を基に、これらの計算を超高速に処理するために最も適した並列計算機アーキテクチャを選択し、物理学研究者との討議の上、システム全体の基本設計を纏める。

この基本設計を基に、さらに詳細設計を行なう。詳細設計の各段階においては、設計された並列計算機の物理学の具体的問題に対する実効性能評価のために、典型的な問題の仮想ベンチマークテストを行ない、その結果に応じて、必要な場合は設計の変更を行なう。この詳細設計に基づき、実装設計を行ない、CP-PACS の組立、調整、機能検証、評価を行なった後、本格稼働に入る。

これらと並行して、システムソフトウエア、プログラミング言語、コンパイラの設計・開発、デバッグ機能の開発など、並列計算機を使用する上で重要なソフトウェアを開発・製作する。

(2) 製作された超並列計算機 CP-PACS を用いて、理論的方法と実験的方法では解決困難な「場の物理」の基本的な重要難問題の解決を図る。素粒子物理学に於いては、強い相互作用の基礎理論である量子色力学に基づく素粒子の諸性質の解明、宇宙初期の物質の状態に関する予言、又、ヒッグス粒子の質量等弱い相互作用に絡む未解決の問題の究明を目指し、さらに、物性物理学に於ける高温超伝導機構の解明、宇宙物理学に於ける銀河、星の形成過程の解明、重力崩壊とブラックホールの形成の解明等の研究を行う。

これらの問題について高い計算能力を実現するために、典型的な問題に対して必要とされる計算能力の検討を当初から行なって、超並列計算機 CP-PACS の設計にその結果を反映させると共に、その基本設計・詳細設計の各段階で仮想ベンチマークテストを行なう。ベンチマークテストの結果を参照しながら、並列アルゴリズムの研究と応用プログラムの開発を行い、また CP-PACS のプロトタイプを用いたテスト計算を実行して、応用プログラム・解析プログラムを改良充実する。CP-PACS の本格稼働時迄に以上の計算準備を充分に行ない、本格稼働と同時に物理計算の実行を開始して、格子量子色力学等、「場の物理」の重要問題に対して世界に先進的な結果を得ることを目指す。

3　研究組織と研究テーマ

図 1: 研究組織機構図

研究班 1	「場の物理」専用並列計算機システムの研究
中澤喜三郎	研究班 「場の物理専用並列計算機システムの研究」の統括
星野 力	並列計算機アーキテクチャの研究 システム設計
白川友紀	
和田耕一	
朴 泰祐	
渡瀬芳行	並列計算機システム制御運用の研究
一井信吾	
中村宏	演算素子高速化の研究、ＣＰＵとメモリ構成の設計
中田育男	並列計算機のソフトウエアの研究
山下義行	並列計算機のコンパイラの研究
小柳義夫	並列計算機のアルゴリズムの研究
森 正武	「場の物理」の高速計算のための並列アルゴリズムの研究
川合敏雄	「場の物理」のプログラムの並列処理効率向上の研究

研究班 2	「場の物理」に於ける諸問題の研究
岩崎洋一	全計画及び研究班 「場の物理に於ける諸問題の研究」の統括
宇川 彰	格子量子色力学の研究 仮想ベンチ ・マーク、アルゴリズム開発、プログラム開発
大川正典	
金谷和至	
青木慎也	
吉江友照	
石塚成人	
観山正見	星、銀河系の生成過程及び天体の重力崩壊の研究 仮想ベンチ ・マーク、アルゴリズム開発、プログラム開発
中村卓史	
梅村雅之	
中本泰史	
宮下精二	スピン系の研究
今田正俊	仮想ベンチ ・マーク、アルゴリズム開発、プログラム開発
川合 光	量子重力の研究、モデル考察、アルゴリズム開発

表 1: 研究テーマ

学術振興会特別研究員
平成４年度
菊川芳夫　(現　:京都大学理学部助手)
石塚成人　(現　:筑波大学物理学系助手)
平成５年度
蔵増嘉伸

4　計算物理学研究センター設置及びその整備

新プログラムの研究推進母体として、計算物理学研究センターが平成４年４月１０日に全国共同利用施設として筑波大学に設置された。研究部門は計算素粒子物理学、計算物性物理学、計算宇宙物理学及び並列計算機工学の４部門から構成される。定員は教官定員１０人、客員教官２人である。センター建物は計算機棟　(図 2参照) である第１期分　(６９８ m^2) が平成５年夏竣工し、研究棟である第２期分　(１０７０ m^2) が平成６年度完成予定である。

研究内容は

1. 計算物理学の研究に適した超並列計算機 CP-PACS の開発製作。
2. 超並列計算機を用いて、素粒子物理、物性物理、宇宙物理に於ける、解析的方法及び実験的方法のみでは解決困難な重要難問の数値的な方法による研究。

である。新規開発する CP-PACS が完成するまで, 並列計算機 QCDPAX をセンターに移管し、有効に活用している。計算物理学の研究テーマとしては、

- 素粒子物理　:宇宙初期の物質の状態、素粒子の質量の計算、ヒッ グス粒子の性質の解明。
- 物性物理　　:高温超伝導機構の解明、結晶成長などのシミュレー ション。
- 宇宙物理　　:銀河、天体の生成過程の解明、重力崩壊とブラック ・ホールの生成過程の解明。

などが挙げられる。

新プログラムの研究メンバーをセンターの共同研究員に委嘱し、センターを中心に研究活動を行っている。文字通り、センターは新プログラム研究の推進母体としての役割を果たしている。

新プログラム研究によって新規に開発 ・製作する超並列計算機 CP-PACS を全国の共同研究者の使用に供するための計算機システムをセンター経費を用いて構築する。計算機システムは図 3 に示したように、CP-PACS, フロント ・計算機システム、QCDPAX から構成される。

フロント ・計算機システムはフロント ・システム ・ホスト、外部記憶装置、磁気テープ、プリンタ、グラフィクス系などから構成される。フロント ・システム ・ホストは、CP-PACS の制御、CP-PACS との間の多量のデータの入出力などを取り扱う。外部記憶装置は、ユーザーのデータを保管する。

フロント ・計算機システムの第一期分は、平成６年３月より稼働を開始し、システムソフトウエア開発、並列プログラム開発、物理学研究などのために 90%以上の稼働率で運用している。第２期分は、外部記憶装置の増強などを主たる目的として、平成７年３月よりレンタル開始の予定である。

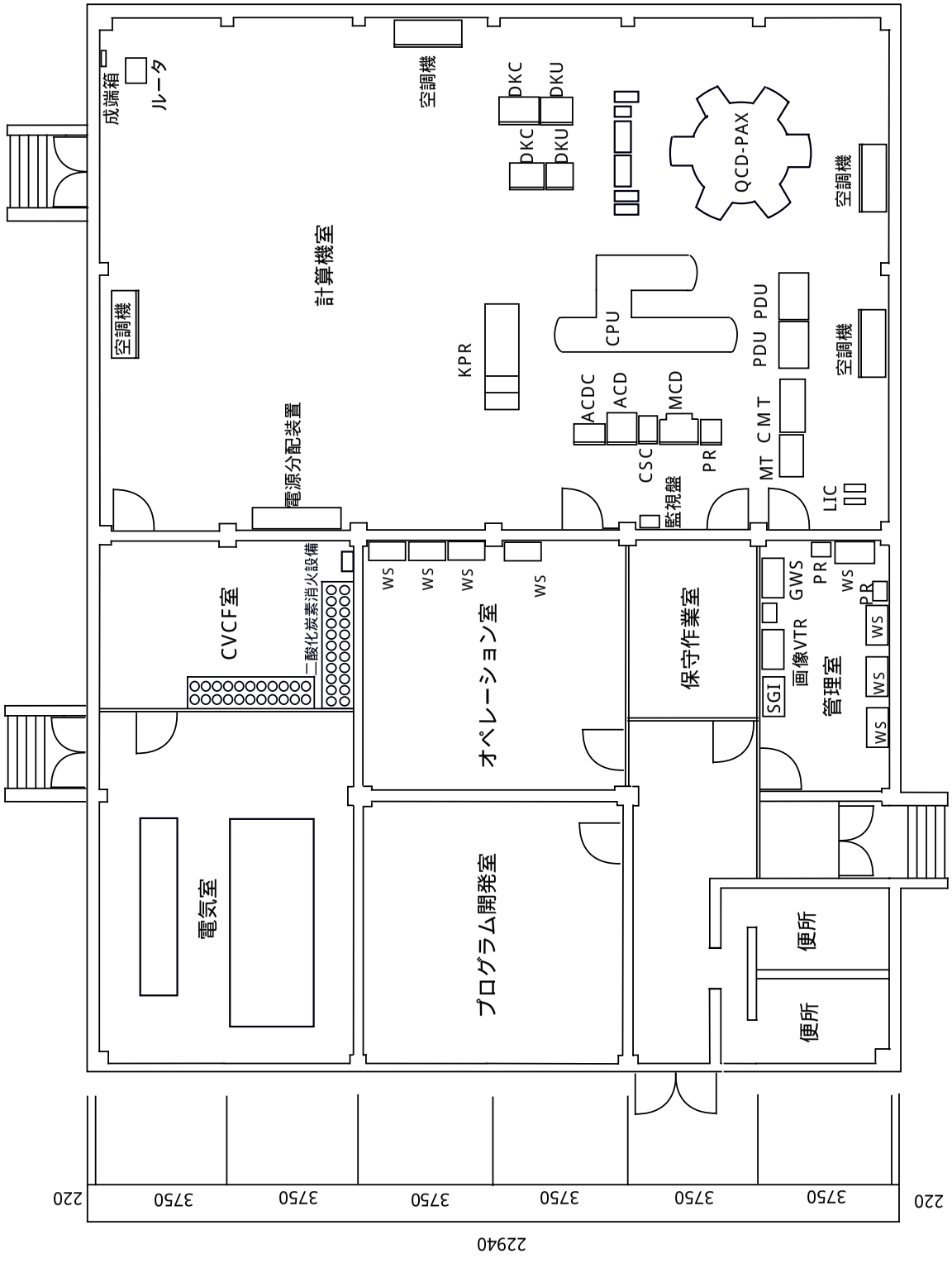

図 2: 計算物理学研究センター計算機棟平面図

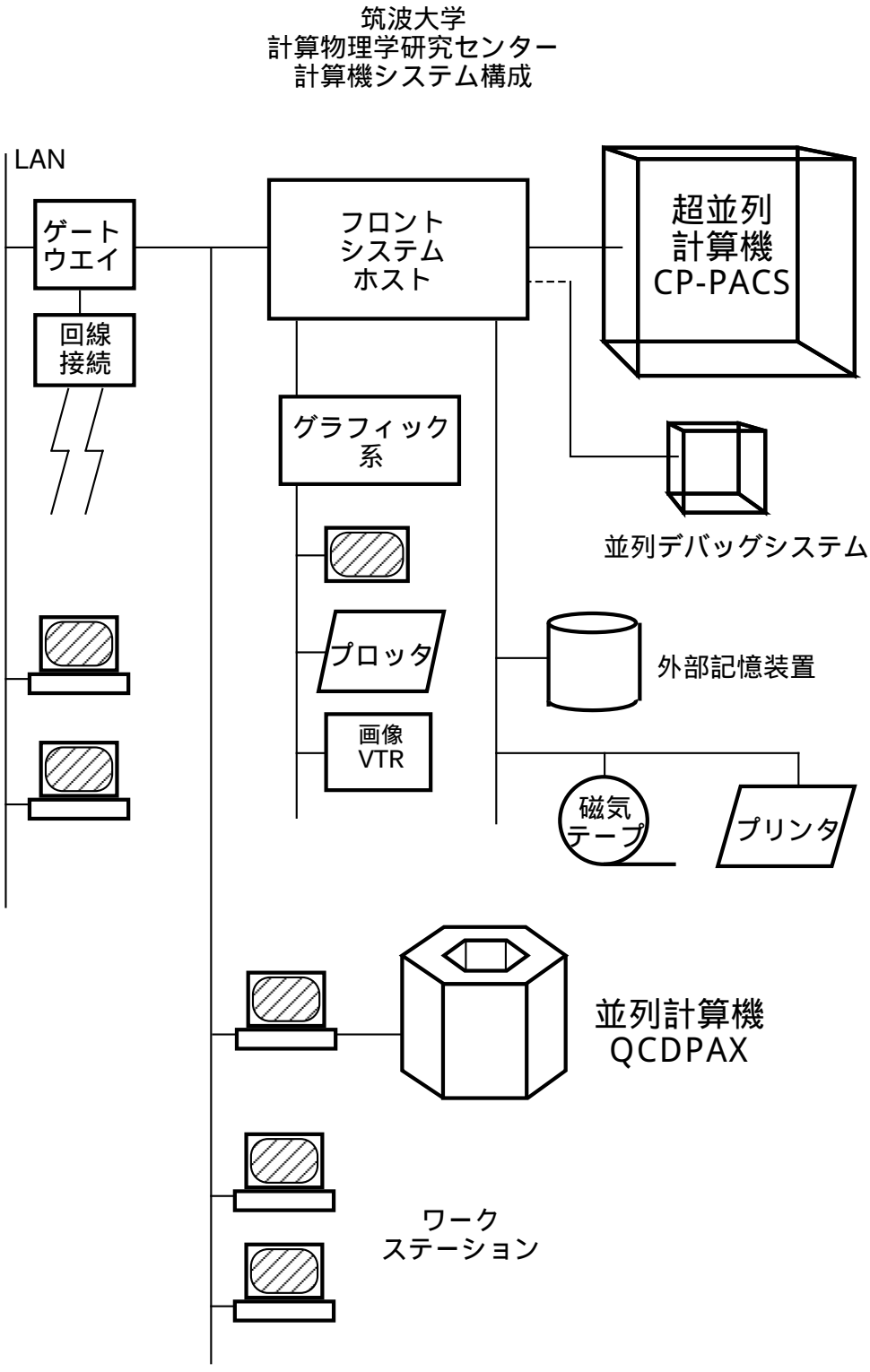

図 3: 計算物理学研究センター計算機システム構成図

5 研究推進方法

研究班は 2 つになっているが、物理学者と並列計算機工学者とが実質的に一つのプロジェクトに有機的に取り組んでいる。そのため、研究経費として班に分配することなく、全体で用いている。研究年次計画概要は図 4に示したとおりで、センター研究員会議及びワーキンググループを中心として研究 ・開発にあたっている。

5.1 センター研究員会議及びワーキング ・グループ

プロジェクトのメンバーは現在 26 名からなり、その中には、汎用大型機及びスーパーコンピュータの開発の中心的な役割をはたしてきた計算機工学者、世界に先駆け並列計算機システム PAX シリーズの開発を推進してきた計算機工学者、並列計算機 QCDPAX を用いて素粒子物理の研究を行なってきたグループ、スーパーコンピュータを用いて研究しているグループなどを擁している。各々がこれまでの経験を生かし、実効速度が速く、ユーザーに使い易く、信頼性の高いシステムを開発製作すべく, センター研究員会議、種々のワーキング ・グループなどの議論を中心として、色々な角度から検討を加え、CP-PACS の開発に取り組んでいる。

研究員会議									
平成 4 年度	4/10	4/18	5/9	5/23	6/8	6/20	7/13	7/22	9/5
	10/1	10/24	11/7	12/5	1/9	1/21	2/6	2/18	3/6
	3/22								
平成 5 年度	4/3	4/22	5/8	5/20	6/5	6/17	7/3	7/15	8/5
	8/26	9/16	10/1	10/22	11/6	11/18	12/4	12/16	1/17
	2/5	2/17	3/5	3/17					
ハードウェア WG									
平成 4 年度	7/9	7/23	8/6	9/5	9/19	10/5	10/27	11/30	12/22
	1/14	1/28	3/1						
平成 5 年度	4/2	5/7	6/17	8/4	9/24	10/28	12/6	1/28	2/28
ソフトウェア WG									
平成 4 年度	11/6	12/3	1/21	2/23					
平成 5 年度	4/28	5/31	7/5	8/4	8/31	9/16	10/22	12/16	2/7
	3/10								

表 2: 研究員会議、ハードウエア WG、ソフトウエア WG 開催日程

センター研究員会議は 2 週間に 1 度、ハードウエア及びソフトウエアのワーキング ・グループは各々 1 カ月に 1 度程度開催している。平成 4 年度、 5 年度の開催日を表 2に示した。研究員会議では、超並列計算機 CP-PACS の開発方針、ハードウエア WG 及びソフトウエア WG の報告、CP-PACS のフロント ・計算機システムの構成などを議題として通常 2 ~ 3 時間程度討論 ・報告が行なわれる。ハードウエア WG、 ソフトウエア WG では、CP-PACS の開発に関する方針から技術的な細部にまでわたる議論を通常 5 ~ 6 時間程度かけて行なう。さらに適宜、並列ネットワークの問題、プロセッサの問題のためなどの、サブワーキング ・グループを構成し、詳細な議論を重ね研究を進めている。

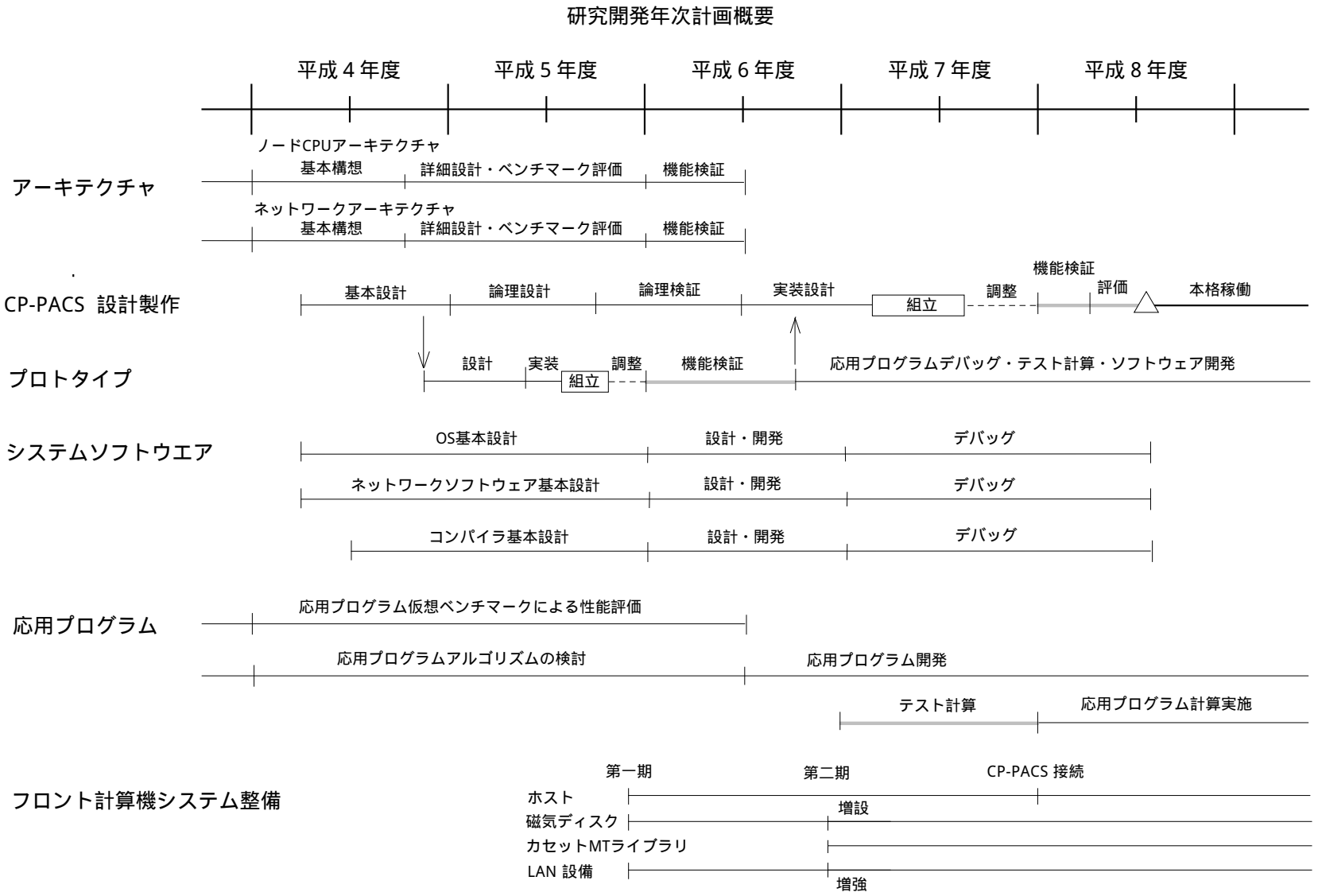

図 4: 研究開発年次計画概要

平成４年度と５年度、CP-PACS の基本設計、詳細設計に関して日立製作所と契約を結び、ハードウエア及びソフトウエアのワーキング ・グループは日立製作所の人も参加して行なっている。

5.2 研究会開催

平成５年３月１６日に、筑波大学計算物理学研究センター主催の第１回研究会を、 超並列計算機による計算物理学」のテーマで開催した。新プログラム研究の進捗状況の報告と関連分野の研究状況の報告を、約９５名の参加者を得て行なった。この研究会の報告集 (講演の OHP のコピー) とセンターの概要を、センターの平成４年度の研究報告集にまとめた [1]。

平成６年３月１日から３月３日の３日間にわたって第２回研究会が、「2 １世紀への宇宙物理学シンポジウム :天体物理学における輻射輸送過程」のテーマで開催され、全国の大学等研究機関より約８０名の参加者があった。シンポジウムで取り上げたテーマである “輻射輸送過程” は、あらゆる宇宙現象に関わっている重要な基礎過程であるが、その取扱いの難しさの故に、これまで十分には調べられてこなかった。しかし、超並列計算機 CP-PACS を用いることで、これまで不可能とされてきた完全６次元 (空間３次元、方向２次元、エネルギー１次元) の輻射輸送を解くことが可能であると考えられ、宇宙物理学における飛躍的な進歩が見込まれる。この状況を視野に入れ、宇宙物理学の諸分野において、輻射輸送がどのように関わっているか及び数値計算法について具体的かつ総合的な議論を行なった。

6 本プロジェクトの背景

本節では、本プロジェクトの背景として、 (1) 計算物理学、 (2) 最近の計算機の発展、特にベクトル型スーパーコンピュータから超並列計算機への発展、 (3) このプロジェクトの前身である並列計算機 QCDPAX 各々の概略を述べる [2]。

6.1 計算物理学

近年のコンピュータの能力の急速な進歩によって、科学技術のあらゆる研究分野で、研究方法に構造的な大きな変革が起きている。従来からの研究方法としては、実験及び観測装置を用いた実験的方法と、数学的な手法による理論的方法の 2 つが主要な方法であった。これら 2 つの方法と並び第三の方法として、コンピュータを用いた研究方法が非常に重要になってきている。いままで、技術的、時間的、価格的な困難のため実行出来なかった実験をコンピュータ上で実現し、数学的に解けなかった方程式をコンピュータで解くことが可能となってきた。物理学に於いても同様で、計算機を主たる研究手段として用いて研究する物理を計算物理学という [3]。

例を素粒子物理にとる。クォーク ・グルーオンの基本法則と考えられている量子色力学 (QCD) から、摂動論を用いて、陽子などハドロンの質量を計算することができない。本質的に、強結合、非線形理論である。電子 ・光子の基本法則である量子電磁気学 (QED) に対しては、摂動論が成功裡に用いられ、“紙と鉛筆” で電子の磁気能率双極子などが計算できたのと対照的である。ハドロンの質量を計算する唯一の方法は、量子色力学を格子上に定義した格子量子色力学を用いた計算機による数値的手法である。また、宇宙初期に於いて温度が約 1 兆度の時に、クォーク ・グルーオン ・プラズマ状態からハドロン状態へ相転移を起こしたと考えられている。実験室でクォーク ・グルーオン ・プラズマ状態を実現させるのは容易ではないが、計算機上にその状態を作り出すこ

とが出来る。このように、“紙と鉛筆” で出来ない計算を計算機を用いて計算したり、実験室で実現困難な状態を計算機上に実現することにより、物理学の本質的な進歩が期待出来る。

量子色力学の場合、時間空間が 4 次元であり、1 格子点あたりの自由度も大きいために、計算速度、メモリ容量ともに、格段のものが要求される。現時点のスーパーコンピューターといえども満足のいく結果を得るにはまだ能力が足りない。他の分野でも似た状況である。

さらに高性能なコンピュータをどうやって実現するかが問題である。コンピューターの能力は年々向上していくことが期待できるので、能力の優れたコンピューターが商品として発売されるのを待つのがひとつである。普通はこうする。素粒子物理の実験の場合、加速器や測定器に出来合いの商品などはもちろんないので、自分たちで設計し製作を発注することが当たり前である。必要なら製作してしまおうという精神風土、基本法則から出発するのであるから近似は導入したくないという要求、これらが結び付いて、自前の並列計算機を製作するプロジェクトが、素粒子物理の分野で、日本、米国、欧州でそれぞれ発足し、色々と重要な成果を挙げている (節 17参照)。

6.2 スーパーコンピュータから超並列計算機へ

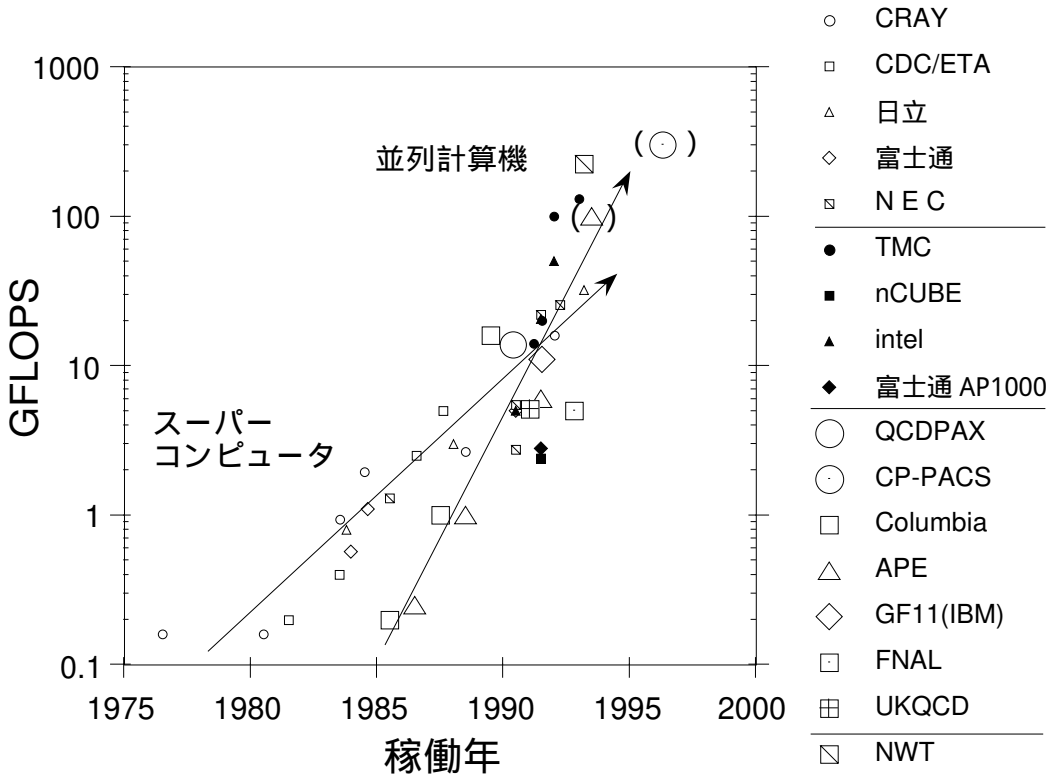

図 5: スーパーコンピュータと並列計算機の最高速度の発展 :1 Gflops は 1 秒間に 10^9 回の浮動小数点演算速度。大きな白抜きの記号は専用並列計算機。

従来は、最高性能の計算機は、いわゆるスーパーコンピュータといわれるベクトル型コンピュータであった。計算速度が 4、5 年に 10 倍になる割合で年々能力が向上してきていた。

しかし、図 5に示すように、1991 年頃に、最高性能の機種がベクトル型コンピュータから超並列コンピュータに取って代わられた。上記の日本、米国、欧州の素粒子専用並列計算機、及び

米国のシンキングマシン社などによる商用並列計算機の出現による [4]。

このクロスオバー現象は、一方で、ベクトル型のコンピュータによる計算速度の高速化がすでに限界に近付いていること、他方で、半導体技術の急速な進歩によって、CMOS 半導体チップが高速化、小型化、低価格化し、千台規模の超並列計算機が実装でき、全体の性能としてスーパーコンピュータの性能以上のものが製作できるようになったことによる。

ベクトル型のコンピュータの性能向上は、ECL 半導体による演算素子の計算速度を高速化することによって達成されてきた。マシンサイクルは 2 ns 程度である。これは、光が 60 cm 進む間に 1 演算を行なうスピードである。これに見あって、メモリからデータを読みだし、メモリに結果を書き込まなければならない。1 演算要素としてこれ以上格段にスピードを向上させるのは難しい。すでに、スーパーコンピュータでも、複数 (10 程度) の演算要素を並べ、同時に計算が行なえるようになっているし、最近は中央制御装置 (CPU) も複数個 (10 程度) 備えているものが当たり前になってきた。この並列性によって、性能向上を図っている。ECL 半導体は発熱量も多く、電力、冷却の問題で、これ以上大幅な並列化は困難である。

パソコン、ワークステーションなどに使われている CMOS 半導体チップは、ECL 半導体に比べると演算速度はいくらか遅いが、必要電力、発熱量は少なくすみ、千台程度備えたものを実装することが可能である。千台程度以上のものを超並列コンピュータと呼ぶ。これからは、最高性能の計算機は超並列コンピュータであることは衆目の一致するところとなった。

並列計算機のアーキテクチャには、 SIMD か MIMD か、メモリ構成は分散か共有か、ネットワークのトポロジー、これらの千差万別の組合せがある。この多様な可能性の中からどれを採用するかが問題である。計算機にさせる仕事の種類は多く、科学技術計算から、事務処理までである。これらどの仕事にも、並列効率のよい並列計算機のアーキテクチャを選択するとなると問題は簡単ではない。しかし、仕事の種類を 「場の物理」に限り、実際の応用プログラムで実効速度が充分に出るかを詰めていくことにより、並列計算機のアーキテクチャをしぼっていくことが出来る。この際、物理学の問題の詳細まで熟知している物理研究者と計算機のハードウエア ・ソフトウエアの細部まで熟知している計算機工学研究者との共同研究によって初めて、ユーザーにとって使いよい、実効性能のでる並列計算機を製作することができる。

並列計算機の理論的最高速度は、一つの演算要素 (PU) の演算速度×台数である。従って、最高速度を向上させるためには、PU の演算速度を向上させ、台数を増やすことが必要である。ただし、実際の計算に於いては、PU 間でデータのやりとりをしなくてはならないので、実効速度を高めるためには、演算速度に見合ったデータ転送速度の向上が必須である。また、実装、価格、電力、信頼性の問題などから、台数にも上限があり、現在の技術では、数千台が限度であると考えられる。

6.3 並列計算機 QCDPAX

1987 年から 1989 年までの 3 年間に約 3 億円の科学研究費補助金 (特別推進研究) を受け、並列計算機 QCDPAX[5] が開発製作された。筑波大学を中心とした素粒子物理学者と並列計算機工学者からなる QCDPAX グループで基本設計を行ない、アンリツ株式会社が製作した。演算要素 (PU) が 480 台から構成されており、2 次元トーラスのメッシュに結合されている。最高速度は約 14 GFLOPS であり、完成当時のスーパーコンピュータの $3-4$ 倍の性能である。それを用いて現在も素粒子物理学の研究を行なっている。完成後約 3 年間フル稼働しており、稼働時間の合計は 2 万時間を越えている。商用機に比べれば、故障も多いし、ソフトウエアも整備されていなくユーザーの苦労も多いが、格子 QCD を用いて、宇宙初期におけるクォーク ・グルーオン状態か

らハドロン状態への相転移の精密計算、ハドロンの質量の計算、クォークの種類数と閉じ込めの関係などに関する結果を導き出している [6]。

QCDPAX は、世界に先駆け 1977 年より筑波大学構造工学系星野力らによって開発が開始された、並列計算機 PAX シリーズの 5 代目に当たる [7]。PAX のアーキテクチャは MIMD、局所分散メモリ型近接通信、2 次元メッシュトーラス構造であり、格子 QCD に適している。QCD のもつ特徴、局所性、近接性、境界条件が、局所分散メモリ型近接通信、トーラス構造に対応している。また 4 次元を 2 次元写像することは何の問題もなく、ハードウェアの作りやすさ、PU 間の通信の速さなど総合的に判断して、従来の PAX シリーズと同じ二次元トーラス構造を採用した。図 6に QCDPAX の構成図を示す。

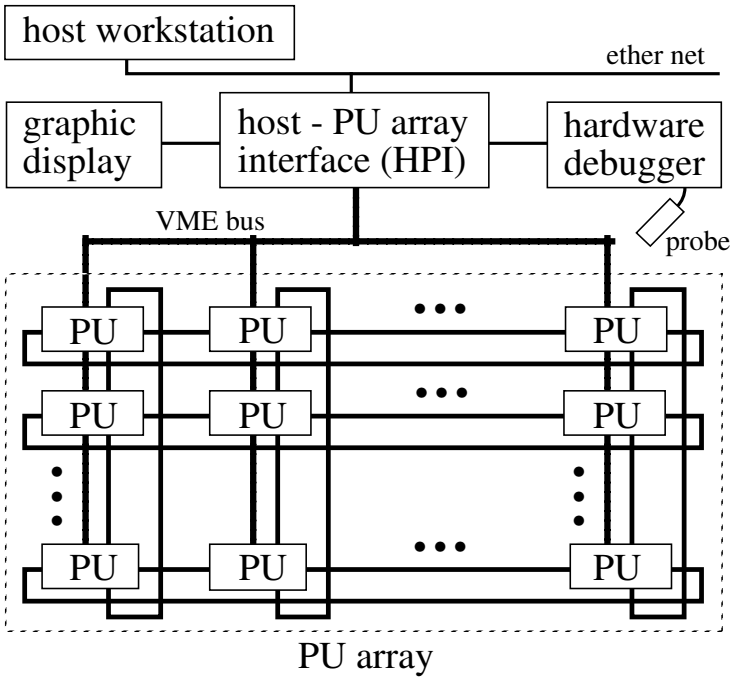

図 6: QCDPAX システム構成図

この基本的アーキテクチャに基づき、1 演算要素あたりの計算速度と、通信速度をそれまでの PAX シリーズより格段に速くして、その当時のスーパーコンピュータより高速の計算機を製作した。一台の PU の速度は約 30MFLPOS であり、全体の PU 台数は 480 台であるので最高速度は約 14GFLOPS となる。

今回の超並列計算機 CP-PACS の開発において、QCDPAX の開発 ・製作過程での経験を生かして, 「場の物理」の計算に適し、かつ使用者にとって使い勝手のよい並列計算機システムの構築を目指している。

Part II
研究進捗状況

平成４年度において、物理学研究者と計算機工学者の共同研究により、「場の物理」の計算に適した専用並列計算機の基本設計を行った。この目的の為に、物理学研究者は、本研究で解決を図る物理学の諸問題に於いて重要な計算の典型的なプログラムを分析・提示した。計算機工学者は、技術的に実現可能な並列計算機の演算ユニット（演算素子、記憶素子等）の構成とその結合方法等についての検討を基に、これらの計算を超高速に処理するために最も適した並列計算機アーキテクチャを選択し、物理学研究者との討議の上、システム全体の基本設計を纏めた。

平成５年度において、基本設計を基に、専用並列計算機の詳細設計を行った。この段階において、ノードプロセッサ・結合ネットワーク・システム制御・入出力制御の各部の論理設計を行い、演算ユニット構成素子（演算素子、記憶素子等）の性能、構成方式等の技術的詳細を選択決定し、それらに基づく並列計算機が「場の物理」の研究に必要な計算性能を満たすかどうかを検討し、必要な場合は設計の変更を行った。併せてこれらの確認用にプロトタイプの部分試作も行った。また、並列処理システム用システムプログラムの基本設計を開始した。物理学研究者は、詳細設計の各段階において、設計された並列計算機の物理学の具体的問題に対する実効性能評価のために、典型的な問題の仮想ベンチマークテストを行った。

以下に、これらの点を詳述する。

7　計算物理学からの並列計算機に対する要求性能の検討

素粒子物理学、宇宙物理学、物性物理学の典型的な問題に対して、並列計算機に対する要求性能を検討した。ここでは、一番詳しく調べた素粒子物理学の格子量子色力学の場合を詳述する。宇宙物理学の計算能力に対する要求は、素粒子物理学と同程度である。ただし、要求する計算能力とメモリ容量との割合が異なるので、その点を主として記述する。物性物理学は、多様なモデルを数多くのパラメータに対して計算を計算を行なうことが多く、計算能力, メモリ容量、ディスク容量に対する要求は素粒子物理学、宇宙物理学と同程度またはそれ以下である。

7.1　素粒子物理学における格子量子色力学

本プロジェクトの主目的の一つは、格子量子色力学、即ち素粒子の強い相互作用の基礎理論を、専用並列計算機を駆使した数値シミュレーションによって解き、その正当性を検証すると同時に、新たな定量的予言を引き出すことにある。本章に於いては、この目的の為必要とされる並列計算機の計算能力に関する検討結果を述べる。

7.1.1　格子量子色力学の計算内容及び計算方法の概要

格子量子色力学 (lattice QCD) は、ハドロンの基本構成粒子であるクォーク（従来から５種類が知られていたが、最近６種類目の証拠が見い出された）と、それらを結び付けるグルオンを基本的自由度とする四次元時空格子上の場の理論である。これらの自由度を表す基本変数は、四次元時空格子の各格子点 n に定義された 3x4 次元複素グラスマンベクトル $q = q_n^{a\alpha}(a = 1, 2, 3, \beta = 1, 2, 3, 4)$ 及び格子上の最小線分 ℓ 各々に定義された３行３列の特殊ユニタリ行列 $U = U_\ell^{ab}(a, b = 1, 2, 3)$ に

より与えられる。前者がクォークの自由度を、後者がグルオンの自由度を表す。変数 q の 3 次元分は変数 U の 3 次元に対応し、カラーの自由度と呼ばれる。残り 4 次元はスピン自由度である。

理論を定める基本的量は、作用と呼ばれる U と q の関数である。格子量子色力学の作用は次式で与えられる。

$$S_{QCD} = \frac{\beta}{6}\sum_{p}\mathrm{tr}\,(U_{\ell_1}U_{\ell_2}U_{\ell_3}U_{\ell_4}) + \sum_{n,n'}\bar{q}_{n'}D_{n,n'}(U)q_n \tag{1}$$

但し、第一項は、格子上の最小正方形 p 全てについてその四辺上のグルオン変数 U の積の行列トレースを取ることを意味する。β は格子量子色力学の結合定数 g^2 と $\beta = 6/g^2$ なる関係にあるパラメタである。第二項はクォーク作用であり、その取り方には、Wilson fermion 法及び Kogut-Susskind (KS) fermion 法と呼ばれる二つの代表的方法がある。Wilson fermion 法の場合、$D = D^{a\alpha,b\beta}_{n,n'}(U)$ は変数 U について線形の $12V \times 12V$ 次元 (V は格子点数) の行列である。本稿では、 特に断らない限り、Wilson fermion 法を中心に評価を行なう。クォーク質量 m_q は行列 D にパラメタとして現れる。

量子色力学の物理量は、作用より定まる重み付きの U と q に関する次の積分平均により与えられる。

$$< \mathcal{O}(U,q) >= \frac{1}{Z}\int\prod_{ell}dU_\ell\prod_n dq_n d\bar{q}_n\mathcal{O}(U,q)\exp S_{QCD}(U,q) \tag{2}$$

但し、Z は $< 1 >= 1$ とするための規格化定数である。

格子量子色力学の数値シミュレーションでは、結合定数とクォーク質量の値を選択した後、適当な確率過程により、積分の配位空間の点 (配位) を重み $\exp S_{QCD}$ に従って生成し、各配位上での物理量の値を平均することにより物理結果を得る。

シミュレーションは、配位の生成法により二種類に大別される。「クェンチ近似」(quenched QCD) は、作用の中の変数 q を無視する近似である。生成法は、変数 U に対する、メトロポリス法・熱浴法等のモンテカルロ法であり、基本演算は、格子の最小正方形各々について、その回りの三辺に沿っての変数 U の行列積 $U_1 \cdot U_2 \cdot U_3$の計算、その値に基づく残り一辺上の U の変更値の確率的棄却・採用である。

「完全な量子色力学」(full QCD) ではこのような近似は行なわない。生成法は、ハイブリッドモンテカルロ法を基本とする。配位分布に対する変数 q の効果を取り入れるには、正規乱数を元とする $12V$次元ベクトル Y 及び与えられた U の配位に対して、行列 D についての $12V$次元連立一次方程式 $Dx = Y$を解くことが必要となり、これが計算時間の大半を占める。解法としては、共役傾斜法・最小残差法・共役残差法等が用いられ、不完全 LU 分解等の前処理も有効である。基本演算は、$12V$ 次元複素ベクトル p と行列 D の積 $p \to Dp$ である。

生成された配位上での物理量の計算は様々であるが、最も計算時間を要するのは生成された Uの各配位上でクォーク伝播関数 $G^{a\alpha,b\beta}_{n,n'} = (D(U)^{-1})^{a\alpha,b\beta}_{n,n'}$を求めることである。これは連立一次方程式 $Dx = b$ を、12 種類の右辺 b に対して解くことにより得られる。ハドロンの質量や遷移行列要素等、強い相互作用諸量は、このクォーク伝播関数を組み合わせて得られるハドロン伝播関数に、χ^2フィット等の処理を行なって抽出される。

7.1.2 格子パラメタの選択

格子量子色力学の基本的対象は複数のクォークからなるハドロンである。ハドロンの空間的拡がりは約 $r_H = 2\mathrm{fm}$ 程度 ($1\mathrm{fm}=10^{-15}\mathrm{m}$)、またクォークの質量は最も軽い u, d クォークの $5 - 10\mathrm{MeV}$ から現在知られている中では最も重い t クォークの $175\mathrm{GeV}$ 迄広い巾がある。この

	現状			目標		
	La/a(fm)	m_π/m_ρ	格子サイズ	La/a(fm)	m_π/m_ρ	格子サイズ
ハドロン質量	2-3/0.2-0.1	≥ 0.5	$\leq 32^3 \times 64$	3-4/0.1-0.05	$0.4 - 0.3$	$\geq 64^3 \times 64$
重いクォークの物理	2-3/0.1	≥ 0.9	$\leq 24^3 \times 48$	2-3/0.05-0.03	≥ 0.9	$\geq 64^3 \times 64$
CP 非保存等	2-3/0.2-0.1	≥ 0.5	$\leq 24^3 \times 48$	5-6/0.2-0.1	$0.5 - 0.4$	$\geq 64^3 \times 64$
有限温度相転移	2-3/0.3-0.2	≥ 0.3	$\leq 16^3 \times 8$	5-6/0.2-0.1	$0.3 - 0.2$	$\geq 64^3 \times 16$

表 3: 格子量子色力学シミュレーションの現状と目標

ような系の物理的性質を離散且つ有限な時空格子上での数値シミュレーションにより求めるには、統計誤差に加え、格子を用いたことに起因する各種の系統誤差を小さくしなければならない。このための基本的条件は、(1) 格子間隔 a (最近接格子点間の距離) が r_H に比べて充分小さく、且つ (2) 格子のサイズ La (L は一辺当たりの格子点数) が r_H に比べ充分大きいこと、さらに (3) クォーク質量を現実の値に近く取りうること、の三点である。

条件 (1) と (2) を満たすには一辺あたりの格子点数 L を大きく取ればよいが、格子量子色力学は 4 次元時空上理論であり、その数値シミュレーションは、必要な記憶容量、基本的演算数、出力データ容量のいずれも L^4 に比例して増大するため、異論の余地のないほど大きな L (例えば $L = 1000$) を取ることは現実的ではない。また、条件 (3) に関しては、軽いクォークに対して前節に述べた連立方程式 $Dx = b$ を解く際の逐次近似解法の必要反復回数が増え、且つ統計的揺らぎが増大するとの困難がある。これは、クォーク質量 m_q が行列 D の定数項として現れ、D の条件数が $1/m_q$ に比例するためである。特に、full QCD シミュレーションに於いては、配位更新の度に連立方程式を解く必要があり、物理量の計算に於いてのみこのことが必要な quenched QCD に比して、著しく計算量が膨大になる。以上の理由のため、格子間隔 a 及び格子サイズ L をどのように取れば有限格子間隔と有限格子サイズに起因する系統誤差が充分に小さい物理結果を得られるか、またその際クォーク質量をどこまで下げうるかを知ることは極めて重要である。

これらの条件の詳細は問題とする個々の強い相互作用物理量に応じそれぞれ多少異なっている。最も基本的なハドロン質量の計算を例に取れば、現在迄に、$a \approx 0.2 - 0.1$fm、$La = 2 - 3$fm (即ち $L \approx 20 - 30$)、またクォーク質量については π 中間子と ρ 中間子の質量で見て $m_\pi/m_\rho \geq 0.5$ (現実の値は $m_\pi/m_\rho = 0.18$ であり、これはより軽いクォークに対応する) の領域が、quenched QCD により詳しく調べられている。その結果を基礎として、格子パラメタについては $a \approx 0.1 - 0.05$fm 及び $La \approx 3 - 4$fm (即ち $L \geq 60$)、クォーク質量については $m_\pi/m_\rho \approx 0.3 - 0.4$ への計算拡張を行なえば系統誤差数%以下の精度が得られると考えられる。

表 3に、格子量子色力学に於ける中心的課題それぞれについて、現在行なわれているシミュレーションのパラメタ値及び信頼しうる結果を得るために目標とすべきと思われるパラメタ値を示す。

7.1.3 必要とされる計算能力の検討

格子量子色力学の数値シミュレーションは表 3に挙げたごとく多岐にわたるが、その中で最も基本的課題は、π 中間子、ρ 中間子、核子等軽いクォークからなるハドロンの質量スペクトルの

計算及びその実験値との比較による量子色力学の検証である。この計算は必要とされる計算能力の面からも最も大規模にわたるものの一つであり、従って、その検討は、格子量子色力学シミュレーション全般にとり基本的である。以下、quenched QCD と full QCD それぞれについて、計算の各段階について考慮すべき点をまとめた後、(1) 計算速度、(2) 主記憶容量、(3) 中間結果保持のための一時記憶容量、(4) 最終結果保持のための外部記憶容量、についての具体的評価を示す。

quenched QCD の計算手続きは、(1) 変数 U に対するモンテカルロ法による配位の生成、(2) 生成された配位上で、$Dx = b$ を解くことによるクォーク伝播関数の計算、(3) クォーク伝播関数を組み合わせたハドロン伝播関数の計算、(4) 以上の繰り返し、からなる。各ステップでの計算時間を左右するパラメタは次のとおりである。

1. 配位間 sweep 数

   変数 U の生成には通常各最小線分上の U_ℓ に逐次確率的変更を加える。各 U_ℓ に於いて必要な浮動小数点演算数は 5000 程度である。全ての U_ℓ の変更を行なうことを 1 sweep とよぶ。統計的に独立な配位を得る為に必要な sweep 数 N_{sweep} は、モンテカルロ法の詳細にも依存するが、少なくとも 3000 sweep 以上を実行する必要がある。

2. 配位数

   統計誤差を充分に制御するために必要な独立な配位数 N_{conf} については充分に知られていない。現在迄で最大の $24^4 - 32^4$ 格子 ・配位数 200 の計算結果によれば、ハドロン質量の統計精度を 1% におさえるのに必要な配位数は、24^4 格子で 10000 程度以上、64^4 格子で 500 程度以上と考えられる。

3. クォーク伝播関数の計算法

   大行列連立一次方程式 $Dx = b$ の逐次近似解法には様々なものがある。繰り返し一回あたりの基本演算は既に述べたごとく $p \to Dp$ であり、方法によりこれを行なう回数が異なる。効率は、繰り返し一回当たりの計算量と繰り返し回数の積により決まる。ここでは格子量子色力学に適し且つ並列計算機上での実現が容易なものとして red/black 最小残差法を例にとる。この場合、$p \to Dp$ は二回必要であり、繰り返し一回あたりの浮動小数点演算総数は格子点数を $V = L_x \times L_y \times L_z \times L_t$ として、$1728V$ となる。

   逐次近似解法の必要繰り返し回数 N_{iter} はクォーク質量等格子パラメタに依存する。現在までの経験によれば、繰り返し回数は物理的クォーク質量の逆数に比例し、格子サイズ等他のパラメタにはあまり依存しない。物理的クォーク質量は π 中間子と ρ 中間子の質量比 m_π/m_ρ と直接に関係する。表 4に最小残差法を用いた場合、解 x の全ての要素が 0.01% 精度で求まるのに要する繰り返し回数経験値を示す。

4. PU 実効速度と PU 間通信性能

   並列計算機に於ける計算時間は、CPU の実効速度と CPU 相互間の通信速度により決まる。ここでは、実効速度はパラメタとし、通信時間は演算時間の 20% と仮定する。

5. 主記憶容量

   主記憶容量は、基本変数に計算アルゴリズム上必要な補助変数及び並列計算効率化のためのいわゆる 「袖」等を加えたもので決まる。後者は並列計算機の PU 台数と格子サイズの比率等に依存するためここでは考察から除外する。quenched QCD の配位生成に於ける基本変

m_π/m_ρ	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.18
m_q(MeV)	713	563	107	61	37	21	11	3.8
繰り返し回数	200	200	360	630	1060	1830	3470	10130

表 4: red/black 最小残差法の繰り返し必要回数 (各要素の精度 0.01%) ($m_\pi/m_\rho = 0.18$ は実験値に対応)。

数はU、その次元 (実数換算) は$72V$、プログラム上必要な変数総数はその約 2 倍である。クォーク伝播関数の計算においては、基本変数はU及び $Dx = b$ の解 x (複素 $12V$次元)、変数総数は逐次近似解法により多少異なり、最小残差法の場合、変数Uの二倍である。

6. 一時記憶容量

ハドロン伝播関数はクォーク伝播関数を組み合わせて得られるため計算時間は大きくないが、クォーク伝播関数は必要な分だけ並列計算機に結合した分散ディスク等一時記憶に保管しておく必要がある。クォーク伝播関数は実数換算で $144V$ 次元の大配列であり、各配位上で計算後一時保管する必要のあるクォーク伝播関数の数 N_qは物理の要請から 5–7 程度に上る。このため、一時記憶容量、一時記憶への書きこみ・読みだし時間は、無視できない要素になる。クォーク伝播関数を計算後、その一時記憶への書き込みは N_q回、またハドロン伝播関数計算に際しての読みだし回数も N_qである。一時記憶への I/O 時間を規定する I/O スループットについては 300MByte/sec を仮定する。

7. 外部記憶容量

格子量子色力学計算の基本となるのは変数Uの配位 (ゲージ配位) である。ひとたびこれが生成されれば、その上で、表 3 に示した多様な計算を行なうことができる。この意味で、外部記憶装置に保管する必要があるのは、配位生成により得られた N_{conf}個の配位及びハドロン伝播関数その他の最終データである。クォーク伝播関数は一配位あたりで既にその容量が膨大になるため外部記憶装置への記録は現実的でない。

full QCD に於いては、配位の生成法が、ハイブリッドモンテカルロ法等に変更される。これに付随する変更点は以下のとおりである。

1. 配位生成

ハイブリッドモンテカルロ法では、変数Uに共役な運動量 Xを導入し、作用をポテンシャルとする分子動力学により二つの変数 U と q をある時間τだけ時間発展させた後、変数Uの確率的棄却・採用を行なう。これが quenched QCD の 1 sweep に相当し、通常 1 trajectory と呼ばれる。この方法は、quenched QCD の場合のモンテカルロ法と異なり、全ての U_ℓを同時に更新する。更新の大きさは分子動力学時間発展のステップサイズ$\delta\tau$により決まり、この各ステップで Yを正規乱数ベクトルとして $Dx = Y$ を解くことが必要になる。

更新された配位の採用率を 80%程度以上に保つためには、経験的にステップサイズを$\delta\tau \approx 0.02(8/L)$ 程度に取る必要がある。従って、trajectory 長を$\tau = 1$ とした場合、1 trajectory

の生成には、$50(L/8)$ 程度のステップが必要である。$Dx = Y$ の解は基本的にクォーク伝播関数の計算と同じであり、これが「完全な量子色力学」シミュレーションが多大の計算時間を要する原因である。逐次近似解の収束に必要な繰り返し回数はクォーク質量に依存し、最小残差法を用いた場合には表 4に与えられる。

2. 配位間隔及び配位数

   ハイブリッドモンテカルロ法による配位生成に於いて、trajectory 数を幾つに取れば独立な配位が得られるかについては信頼すべき資料がない。配位生成のための計算時間が膨大であるので、できうる限り多数の配位上で物理量の計算を行なっているのが現状であるが、少なくとも$\tau \approx 5$ 程度（即ち、trajectory 長が 1 ならば 5 trajectory）は間隔を置くべきであると考えられる。

   full QCD の大規模計算としては、現在迄に、$16^4 - 20^4$ 格子上で総 trajectory 数 1000-2000 が行なわれている。その経験を基にすれば、統計誤差を有意味に評価するために必要な総 trajectory 数は、24^4 格子で 10000 程度以上（即ち、5 trajectory 間隔にて、配位数 2000 以上）、32^4格子で 3000 程度以上が一つの目安と考えられる。

3. 主記憶容量

   基本変数は U と共役運動量 X 及び Y、補助変数には $Dx = Y$ を解くためのもの等が含まれる。プログラム上必要な変数総数は U の 5 倍程度である。

格子量子色力学のハドロン質量計算に於いて、一配位を処理するために必要とされる浮動小数点演算数及び記憶容量を表 5 にまとめる。但し、ハドロン伝播関数計算部分は、浮動小数点演算数は計算内容に応じ変動が大きいこと、記憶容量はクォーク伝播関数計算の為に必要な容量にて充分であることから省略した。

以上の考察を基に、実効速度 400GFLOPS を仮定した場合、ハドロン質量計算に要する計算時間及び記憶容量の評価を、幾つかの格子サイズに対して、表 6(a) に示す。前提条件は同表 (b) にまとめてある。また、実効速度を変数とした場合の計算時間についての評価を図 7（quenched QCD）と図 8（full QCD）に与える。

計算ステップ	配位生成		クォーク伝播関数
	quenched QCD	full QCD	
演算数	$2 \cdot 10^4 V N_{sweep}$	$1728 V N_{iter} \tau / \delta\tau$	$1728 V N_{iter}$
主記憶	$144V$	$368V$	$144V$
一時記憶	–	–	$288 V N_q$
外部記憶	$72 V N_{conf}$	$72 V N_{conf}$	–

表 5: ハドロン質量計算に於ける浮動小数点演算数及び記憶容量（ワード単位）

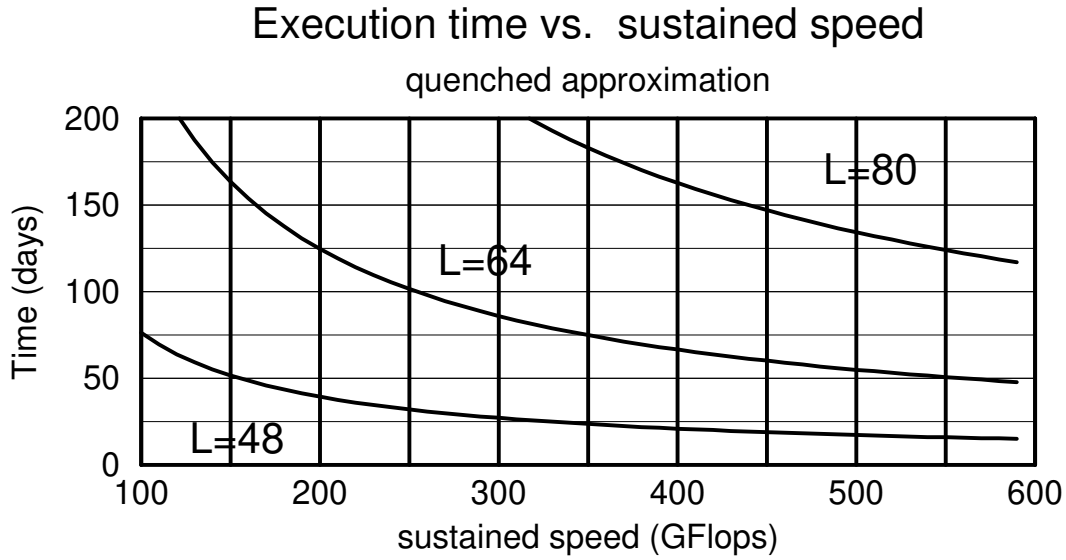

図 7: 格子サイズ L^4 の quenched QCD ハドロン質量計算に必要な CPU 日数の実効速度依存性。前提条件は表 6(b) に同じ。

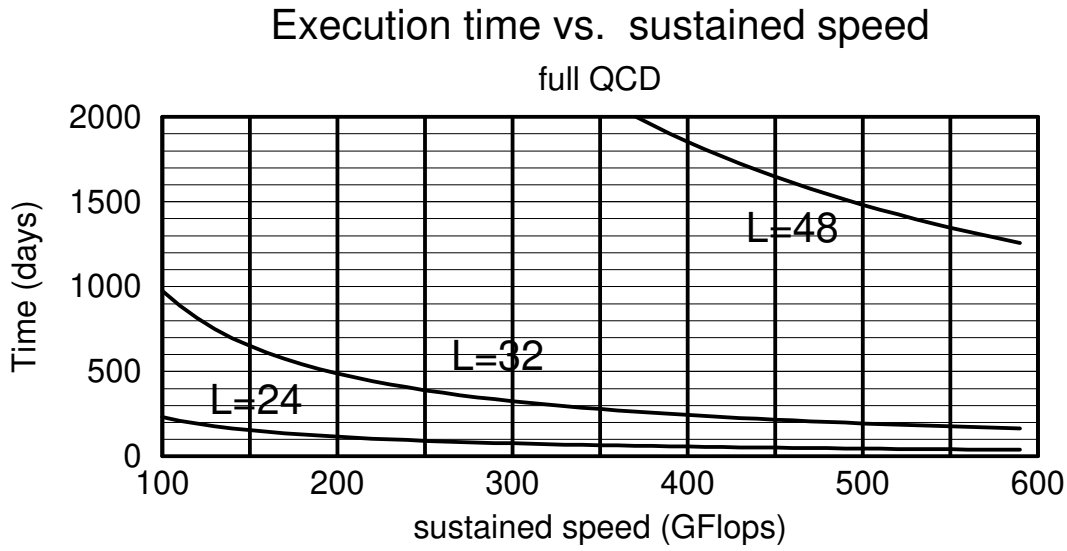

図 8: 格子サイズ L^4 の full QCD ハドロン質量計算に必要な CPU 日数の実効速度依存性。前提条件は表 6(b) に同じ。

(a) ハドロン質量計算の計算時間及び必要記憶容量

	quenched QCD			full QCD		
	48^4	64^4	80^4	24^4	32^4	48^4
計算時間 (日)	21.1	66.7	163	58.0	244	1854
主記憶 (GB)	5.7	18	44	0.91	2.88	14.6
一時記憶 (GB)	68.3	216	527	2.85	9.0	45.6
外部記憶 (GB)	1420	4500	10990	89	281	1424

(b) 前提条件

全体に関する条件	quenched QCD に関する条件	full QCD に関する条件
演算実効速度=400GFLOPS	$m_\pi/m_\rho = 0.9, 0.8, 0.7, 0.6, 0.5, 0.4^\dagger$	$m_\pi/m_\rho = 0.7, 0.6, 0.5, 0.4^\dagger$
通信時間 / 演算時間=20%	配位数 500	配位数 500
一時記憶 I/O スループット	配位間 sweep 数 3000	配位間隔 $\tau = 5$
=300MByte/sec		$\delta\tau = 0.02(8/L)$

$^\dagger Dx = b, Y$ の解法は前処理つき最小残差法、収束回数は表 4 と同じ。

表 6: 格子量子色力学に於けるハドロン質量計算の計算時間及び必要記憶容量

7.1.4 乱数及び初等関数

格子量子色力学のシミュレーションに質の良い乱数の効率的生成は必須の要件である。表 7に熱浴法を用いた quenched QCD での配位生成及びハイブリッドモンテカルロ法による full QCD での配位生成に於いて必要な乱数の種類と個数、さらにその使用に付随して必要となる初等関数の種類を挙げておく。

7.1.5 結論

7.1.1節 「格子量子色力学の計算内容及び計算方法の概要」の表 3 に挙げたように、格子サイズ 64^4、独立配位数 500 の計算は、量子色力学数値シミュレーションに於いて信頼し得る結果を得るための一つの大きな目標である。

quenched QCD に於いては、前節表 6 及び図 7 によれば、実効演算速度 400GFLOPS を達成できれば、この計算は 60CPU 日程度で実行することが可能である。格子間隔についての系統誤差の評価の為には、格子の物理的サイズ La を一定に保って、一辺当たりの格子点数 L を三種類以上変えた計算が必要である。これについては、最大格子サイズを 64^4 に取り、より小さな格子サイズを用いてより大きな格子間隔の計算を行なうことにより、三種類の格子間隔に対して総計 180CPU 日程度で実行することができる。さらに、部分的に 64^4 より大きな格子サイズあるいは配位数 500 より高統計が必要な場合、クォーク質量等のパラメタを限定してその点に集中した

	種類	個数	初等関数
熱浴法	$[0,1]$ 一様乱数	$(24N_{hit}+48)V^{\dagger}$/sweep	$\exp, \sqrt{}, \cos, \sin, \log$
ハイブリッド法	正規乱数	$56V$/trajectory	$\tan, \sqrt{}, \cos, \sin$

† N_{hit} は $SU(2)$ 部分行列あたりのヒット回数。

表 7: 量子色力学数値シミュレーションに必要とされる乱数の種類と個数及び使用する初等関数

計算を行なうことにより、充分実行可能と思われる。必要主記憶容量については、プログラム上必要な「軸」の部分、また上記の部分的により大きな格子サイズの計算の可能性、等を考慮すると、80GByte 以上が必要と考えられる。また、一時記憶及び外部記憶については、多岐にわたる計算結果を保持する必要性等を考慮すると、一時記憶 1000GByte 以上、外部記憶 50TByte 以上が必要条件と考えられる。

full QCD については、red/black 最小残差法により $Dx = b$ を解く限り、実効速度 400GFLOPS の場合の現実的計算は 32^4 格子程度に留まるが、この場合でも、現在の quenched QCD 計算を凌駕する結果が期待される。また、不完全 LU 分解共役傾斜法など、繰り返し回数の大幅縮減が可能な解法の効率的な並列化が実現できれば、48^4 格子の計算も現実味を帯びる。従って、計算アルゴリズムの開発も並列計算機の有効活用の面から力を注ぐべきテーマである。必要記憶容量に関しては、表 6からもわかるように、quenched QCD 計算の場合から評価した値が実現されれば充分である。

以上をまとめて、

実効演算速度 400GFLOPS 以上、
主記憶 80GByte 以上、
一時記憶 1000GByte 以上、
外部記憶 50TByte 以上

を実現することができれば、quenched QCD の枠内では多くの問題について最終的解決に充分の見込があり、full QCD に於いても、現在の quenched QCD 計算以上の精度の達成が可能であると考えられる。

7.2 宇宙物理学からの並列計算機に対する要求性能の検討

計算宇宙物理学の眼目は、宇宙における様々な非線形現象、例えば銀河形成や星・惑星系の誕生、宇宙大規模構造の発生といった複雑な過程を、数値シミュレーションによって解析し、その本質的物理機構を解明するところにある。この目的のためにこれまで用いられてきた手法は、主として宇宙流体力学である。宇宙流体力学は、様々な数値流体計算法と共に発展してきたが、最近までの計算は、空間的に 1 次元、 2 次元が主流であった。これは、計算機の能力が足りなかったためである。しかし、昨今のスーパーコンピュータの発達により、現実的な 3 次元の計算を行なうことも可能になってきている。さらには、ブラックホール周りのガス運動や、銀河中心核の物理を解明するために、流体力学と電磁場を結合させた電磁流体力学を数値的に解く試みもなされ成果を上げ始めている。しかし、これらの 3 次元計算は、空間分解能の点で、まだまだ不十分なものであり、 1 桁近く線形分解能を上げることが要求されている。

一方、このような計算機の発達にあっても、いまだに扱えない分野がある。それは、輻射流体力学である。輻射は宇宙に普遍的に存在するものであり、あらゆる宇宙現象に密接に関係している。そもそも、天体の観測は、そこで起こっている物理過程の結果として生まれた輻射を見るものであり、この情報を通してその物理機構を解明していく。すべての天体現象で、輻射場と物質場の相互作用が重要な役割を果たしていることは間違いないが、この相互作用を完全な形で扱うことはこれまで不可能とされてきた。それは、この問題が空間 3 次元、輻射場 3 次元の 6 次元問題になるからである。しかし、超並列計算機を用いれば、その主記憶容量、処理速度の点から言って、この 6 次元問題を解くことが可能になる。 (具体的な評価は後に述べる。) また、この問題は輻射場の 3 次元については完全に並列処理を行なうことが可能であり、並列計算に極めて適している。

ここでは、宇宙流体力学、電磁流体力学、輻射流体力学の具体的数値計算法について、現状で行なわれている計算の分解能と、超並列計算機によって目標とする分解能、及びその場合の必要メモリ容量、ディスク容量、標準的な 1 つの計算を行なうのに必要な CPU 時間をまとめ、必要性能を評価する。

7.2.1 宇宙流体力学

基礎となる宇宙流体力学を解く方法は大きく分けて 2 つある。それは、オイラー法とラグランジ法である。オイラー法は、空間に固定グリッドを張りめぐらし、そのグリッドでの物理量の変化を追いかけていく方法で、通常、格子法と呼ばれる。これに対し、ラグランジ法は、流体の運動と共に物理量の変化を見る方法である。空間 3 次元の計算でこれを実現する方法に、流体粒子法がある。この 2 つの方法は、どのような物理変化を見ていきたいかによって使い分けされるべきものである。

宇宙流体力学の特徴は、多くの場合、自己重力系の流体力学であることである。また、しばしばダークマターなどの無衝突粒子系も含んだ 2 成分系の計算になることもある。宇宙流体力学で扱う系は、 2 次元以上の構造を持つことがほとんどである。この場合、自己重力のポアッソン方程式をどう解くかによって計算コストは異なってくる。全ての要素の重力相互作用を直接的に計算するのが最も精度がよいが、これは要素の数の 2 乗の計算コストがかかり非常に重いため、通常は重力計算時間の短縮のための様々な便法が用いられる。

格子法　格子法で用いられる重力計算法の一つは、格子に対して FFT を用いる PM(Particle-Mesh) 法で、計算量は $O(N_G)$ である。ここで、N_Gは一辺当たりの格子点数を L として $N_G = L^3$である。ただし、この場合の重力計算精度は、格子間隔の数倍になることが知られている。そこで、精度を上げる為に $\mathrm{P^3M}$(Particle-Particle Particle-Mesh) 法を用いることがある。$\mathrm{P^3M}$ 法は、長距離は FFT で計算し、短距離は直接法で計算するもので、計算コストは $N_G \log N_G$に比例する。

流体部分の方程式系は双極型の偏微分方程式である。その上宇宙物理では、衝撃波が現れる問題を解くことが多い。このような系を数値的に解く方法として最近は、衝撃波を含んだ解の非線形安定性の条件 (TVD 条件と呼ばれる) を満たす TVD 法が主流である。TVD 法には空間 2 次精度の Roe 法や TVD-ES 法, 3 次精度の PPM 法などがある。これまでの計算では、線形空間分解能 ($\equiv 1/L$) で 0.01 程度の計算が限界であったが、この分解能を 1 桁上げることが目標である。この場合、時間発展方程式を安定に解く為の条件 (CFL 条件) を満たなければならないため、時間間隔を分解能に比例してとる必要がある。従って、空間分解能を 1 桁上げることは、 1 job のステップ数が 10 倍になり、CPU コストは格子の増加分より 1 桁大きくなる。表 8に $\mathrm{P^3M}$-TVD 法

の現状と目標を、必要メモリ容量、ディスク容量、標準的な１つの計算を行なうのに必要な CPU 時間、データ転送時間で示す。この表を見てわかるように、現在の空間分解能を１桁上げる為には 100GB 近い大容量メモリが必要である。また、この計算をいくつかのパラメータについて行なう為には、500Gflops 以上の CPU 性能が必要である。

	分解能 $(1/L)$	メモリ (GB)	ディスク容量 (GB)	CPU 時間 (/job)	データ転送時間 (/job/(10MB/s))
現状	0.01	0.08	0.4	6 hr/Gflops	3.7 min
目標	0.001	80	400	120 hr/500Gflops	12.4 hr

表 8: 宇宙流体力学シミュレーション (格子法) の現状と目標

流体粒子法 (SPH) 流体粒子法は流体のラグランジ計算をするため、格子法の場合のような FFT を用いることができない。そこで、粒子法の重力計算には Tree 法と呼ばれる高速計算法が用いられる。Tree 法は、空間をおのおの一つの粒子が占有するテリトリーに分割し、近くは高分解能で計算し、遠くは低分解能で計算する方法で、ダークマターや流体粒子のような粒子系の自己重力計算に適している。Tree 法の演算量は、流体粒子の数を N_S として $O(N_S \log N_S)$ である。分解能を格子法と比較すると、$N_S = O(10) {N_G}^{\gamma},\ 1/2 \leq \gamma \leq 2/3$ の関係がある。従って，$N_G = 10^6\ (L = 100)$ の格子法と $N_S =$ 数 10^4の粒子法がほぼ同じ空間分解能の計算ということになる。ここで、流体粒子法の分解能とは線形分解能 $(\equiv 1/\sqrt[3]{N_S})$ である。Tree-SPH 法の現状と目標を表 9にまとめる。流体粒子法は格子法に比べると、メモリ、ディスク容量共に負荷は小さい。従って、この方法では現在の分解能を１桁以上上げることが容易である。

	分解能	メモリ (GB)	ディスク容量 (GB)	CPU 時間 (/job)	データ転送時間 (/job/(10MB/s))
現状	0.05	0.004	0.2	2.6 hr/Gflops	25 sec
目標	0.005	4	200	110 hr/500Gflops	7 hr

表 9: 宇宙流体力学シミュレーション (粒子法) の現状と目標

7.2.2 電磁流体力学

電磁流体力学は、宇宙流体力学と電磁場の方程式を結合したものである。電磁流体力学では、これまで流体計算に関しては非 TVD 法を用いたもの (Lax-Wendolf MHD 法) が多く用いられてきたが、今後は TVD 法を使った Roe MHD 法などの計算が望まれる。ここでは、Lax-Wendolf MHD 法を用いた現状と、Roe MHD 法を用いた目標を表 10に示す。電磁流体力学では、電磁場の物理量が加わるため、分解能を１桁上げようとすると、100GB を越えるメモリが必要になってくる。また、物理的に意味のある計算を行なうためには CPU 性能は 500Gflops 以上なければならない。

	分解能 $(1/L)$	メモリ (GB)	ディスク容量 (GB)	CPU 時間 (/job)	データ転送時間 (/job/(10MB/s))
現状	1.25×10^{-2}	0.14	0.7	2 hr/Gflops	8.3min
目標	1.25×10^{-3}	136	683	250 hr/500Gflops	19 hr

表 10: 電磁流体力学シミュレーションの現状と目標

7.2.3 輻射流体力学

輻射流体力学は、宇宙流体力学と輻射輸送方程式を結合したものである。輻射流体力学は、空間 3 次元 (L^3)、輻射方向 2 次元 $(N_\theta \times N_\phi)$、輻射エネルギー (波長) 空間 1 次元 (N_ν) の 6 次元問題である。このような高次元性のため、輻射流体力学は計算時間よりも主記憶容量の方がまず問題になる。

輻射輸送方程式の解法には、大別して Long Characteristic 法と Short Characteristic 法がある。前者は、計算領域の端から端にわたる長い光線をとり、この光線上で輻射輸送を解く方法である。この場合は、流体の各格子点で、一般に光線の通る位置が異なるため、コーディングが複雑になる上、並列化が難しい。一方、Short Characteristic 法は、各空間メッシュで同等の光線を張ることにより、並列化を可能にする方法である。とはいうものの、流体の物理量と輻射の 6 次元の物理量すべてを計算機のメモリーにおく余裕はない。そこで、次のような方法を考える。流体運動を決める際に必要なのは、各点での輻射強度そのものではなく、これを方向について積分した輻射のエネルギーとフラックスである。輻射輸送を解く際には輻射場の境界条件がわかっていればよいから $O(L^2 N_\theta N_\phi N_\nu)$ の 5 次元分で済むことになる。この境界条件の輻射強度を補助記憶におけば、主記憶では輻射輸送計算に必要な空間格子点での物理量分の容量が必要となる。これは、空間分解能を $L = 100$ にとった場合 172MB になる。輻射の各方向について並列計算を行なうためには、各 PU にこれだけのメモリがなければならない。また、この計算法では、必要 PU 数は方向分解能の逆数 $(N_\theta N_\phi)$ に等しい。表 11に Short Characteristic 法による輻射流体力学の目標値をいくつかの分解能について示す。ここで、メモリは PU 当たりのメモリ容量、ディスクはシステム全体のディスク容量である。$L = 100$ は現在格子法で行なわれている空間分解能であり、少なくとも空間的にはこの値以上を実現し、これに輻射計算を加えることを目指す。

空間分解能 $(1/L)$	波長分解能 $(1/N_\nu)$	PU 数 $(= N_\theta N_\phi)$	メモリ (MB/PU)	ディスク (GB)	CPU 時間 (/job/500Gflops)	データ転送時間 (/job/(10MB/s))
0.01	0.01	1000	176	250	75 hr	35 hr
0.01	0.01	2000	176	500	150 hr	69 hr
0.01	0.005	2000	176	1000	300 hr	139 hr

表 11: 輻射流体力学シミュレーションの目標

7.2.4 結論

超並列計算機を使用する際に、計算宇宙物理に共通して言えることは、得られる結果の分解能は CPU 性能よりもメモリ容量に強く依存するということである。宇宙流体力学、電磁流体力

学では、これまでの分解能を 1 桁上げるためには、全メモリとして 100GB 以上が必要である。また、CPU 性能は 500Gflops 以上が望まれる。これらの値を実現するためには 2048 台以上の PU が必要である。

また、輻射流体力学では PU 台数だけでなく各 PU のメモリ容量が重要な要素であり、意味のある計算達成のためには PU 当たり 180MB 以上のメモリが必要である。さらにこの計算では、ディスクと PU 間のデータ転送時間が CPU 時間に匹敵する大きさになる。全計算時間軽減のためには、ディスク-PU 間で 10MB/s 以上の実効転送率を実現する必要がある。

8 超並列計算機 CP-PACS アーキテクチャ

超並列計算機 CP-PACS (Computational Physics — Parallel Array Computer System) に対する前節に述べられている必要性能評価を基礎に，システムの主要目標仕様を下記のように定めた．以下先ず主要項目を掲げ，後にそのアーキテクチャにつき簡単な説明を行う．

8.1 超並列計算機 CP-PACS システムの主要目標仕様

(1)	全体の理論的な peak speed	:	600 GFLOPS 以上 (64 ビットデータ) (当初の 300 GFLOPS を改定した目標値)
(2)	全体の主記憶容量	:	128 GB 以上 (当初 64 GB の改定目標値)
(3)	並列計算機を構成する node processor (PU)	:	汎用の高性能マイクロプロセッサの 機能拡張したもの (PA-RISC 1.1 に PVP-SW 機能付加)
(4)	並列相互結合用ネットワーク	:	科学・技術計算に適合したもの (3 次元 HXB (hyper-cross-bar network))
(5)	並列方式	:	分散記憶型 MIMD 方式
(6)	補助記憶装置	:	500 GB 以上を分散配置
(7)	全体を小部分に分割した運用	:	可能なこと
(8)	ソフトウェア	:	Fortran, C, assembler 言語， 結合網通信用 library, OS kernel など
(9)	FCS (front computer system) との接続	:	peak 100 MB/s 以上の 高速インターフェースで接続
(10)	稼働開始	:	1995 年度
(11)	並列計算機システムの信頼度	:	MTBF $>$ 2000 Hr

8.2 CP-PACS の アーキテクチャ

(1) CP-PACS システム全体の理論的な peak speed は，並列計算機を構成する個々の PU (node processor) の理論的な peak speed と全体の PU 台数の積で表される．

最近の半導体技術の進歩および計算機設計技術の進歩により，本研究スタート時点では 1 台の汎用の高性能 micro-processor で peak speed 150 MFLOPS のものが実用の域に達し始め，研究期間中に 300 MFLOPS 以上の性能のものの実現が充分可能なものと予測されたので，1 台の PU の性能目標は 300 MFLOPS 以上を狙うこととした．

並列計算機を構成する PU の総台数については，CP-PACS ではその目標としている全体の peak speed を出来る限り高くしたいので，PU 台数も出来る限り大きくしたいのであるが，その反面，

- 設置された時の装置の規模が現実的なものであるか否か，
- 装置全体の信頼度が計算物理学の計算に耐えられるものに成り得るか否か，
- 必要とされる費用の問題

なども勘案して台数を決める必要がある．CP-PACS では全体の PU を論理的に 3 次元に配置するイメージから，千台以上の PU から成る構成を検討することとし，当初は 1,024 台構成のものを中心に各種の検討を行った．検討の結果相互結合ネットワークの構成方式が明確となり，加えて物理的な実装の可能性も明らかとなってきたので，2,048 台の PU から成るシステムを目標とすることに目標仕様の再設定を行った．実現可能性という意味からは，4,096 台の構成のものまでは可能であるが，予算上の要因から上記のような線となったものである．

したがって並列計算機としての CP-PACS の目標性能は 600 GFLOPS 以上 (64 ビットデータ) である．

(2) 全体の主記憶容量については，予想される計算物理学の計算に必要となる総記憶容量に対して，使用可能な半導体記憶素子，各 PU での主記憶装置の構成方法，PU の高速性を損なわないための主記憶の高速性の確保などを勘案して，64 MB/PU に設定した．

(3) 並列計算機を構成する PU (node processor) については，(1) 項にも述べた如く当初は汎用の高性能マイクロプロセッサの使用を考えていたが，実際の計算物理学の応用での性能を机上で検討した結果，汎用のマイクロプロセッサそのままでは大規模な計算の場合に充分な性能を発揮できないことが判明した．これは主として汎用のマイクロプロセッサは高性能ワークステーションなどへの組み込みを意識したものであって，cache memory が有効に機能することを前提として，高性能を達成することを狙ったものであるためである．

そこで CP-PACS としては，プロセッサアーキテクチャとして次節に述べるような PVP-SW (pseudo vector processor based on slide window, スライドウィンドウ機能による擬似ベクトルプロッセッサ) を新たに考案し，メーカの協力も得てこの機能をも実現できるように汎用のマイクロプロセッサを拡張したものを新たに開発することとした．

この機能の付加により，各 PU は実際の問題の計算時に peak speed の 70 % 以上の性能を発揮できることが期待されている．

(4) 並列計算機用の相互結合ネットワークに関しては，隣接した PU の間のデータの転送のみならず，多様な科学・技術計算に適合したものを念頭に置き，各種の結合網につき検討を行った．その場合結合網の主要部分は半導体の特種専用 VLSI で構成されることになるので，その VLSI の実現可能性，および相互結合網全体の実装方法と性能との関係を検討した．その結果 3 次元の HXB (hyper-cross-bar network) を採用することとした．1 本の物理的な転送リンクの転送性能 (transfer rate) については，パイプライン的転送を行うことにより，200 MB/sec 以上の高速の転送性能を確保することを目論んでいる．ネットワークのデータ転送方式に関してはウォームホール (wormhole) 方式を採り，経路選択については固定的選択方式により，相互結合網でのデッドロック (dead-lock) を回避する方式を採っている．

計算中のプログラムからのデータ転送要求に対しては，それぞれの PU に存在する OS (operating system kernel) が関与する通常の転送プロトコル (protocol) の他に，計算プログラムから直接転送を行うことが出来る高速転送プロトコルも設定することとし，並列処理のデータ転送に伴うオーバヘッドの低減を狙っている．

また高効率の並列処理を行わせるために，PU 間で必要となる同期をとるための同期ネットワーク，および多数の PU へ同時に転送を行わせるための放送用の機能も用意することにしている．

(5) 並列方式については，物理的な構成から考えて，主記憶装置は各 PU に分散して実装する必要があるので，必然的に分散記憶型となる．物理的には分散して実装するにしても，プログラムから見たときにはあたかも全ての PU が共有している記憶装置があるように見える分散 ･共有メモリ型の並列機の研究が最近行われているが，未だ充分な成果を上げるに至っていない．そこで CP-PACS では send/receive 型のメッセージパッシング (message passing) 方式を採用することとしている．

並列処理の方法に関しては，特に初期においては，SPMD (single program multiple data) 方式での処理が多いものと思われ，そうであるならば SIMD 型の並列機でも良いのではないかという議論もあるが，SPMD は SIMD とは異なる点もあり，また多種多様な計算物理学上の応用にも応えられ，しかも OLTP (on-line transaction processing) やデータベース (data base) 処理などの並列計算機工学上の問題にも対処できるようにするため，MIMD 方式を採用することとした．

上記 (3), (4), (5) を基本とする並列処理方式の採用により，CP-PACS では計算物理学上の主要な計算 (例えば QCD 計算) に関し，全体として実計算時の実効速度 (sustained speed) が peak speed の 50 % 程度を達成できることを狙っている．

(6) 補助記憶装置については，長時間に及ぶ計算の中間結果などのデータの一時記憶用，バックアップ (backup) 用，その他のために，並列計算機 CP-PACS から直接使用することの出来る補助記憶装置，具体的には磁気ディスク (disk) 記憶装置を分散して接続することとした．

必要とされる総記憶容量は 500 GB 以上と見積もられている。このような大量の容量のものを実際に設置するにはいくつかの問題がある．

その 1 つはこれだけの容量のものを，コスト/パフォーマンス的に効率の良い 3.5 インチの小型磁気ディスクで実現するとしても，CP-PACS 全体としての総ディスク装置台数は 500 台を超えるものとなり，1 台の磁気ディスク装置の寿命が 20 年としても，約 0.5 か月に 1 台の故障 (unreadable error) が発生することになり，補助記憶装置としては大いに問題となる．この点に関しては，当初は他に方法がなく，ユーザは使用時にこのような事を充分注意しながら使用するようにするしかないと思われた．その後この方面の技術が進展し，RAID (redundant array of inexpensive disk, 4 台のディスクに対して 1 台の冗長ディスクを付加することにより 5 台中の 1 台のディスクの故障に対しては自動的に記憶情報の回復が行われるもの) が開発され，これを使用することとして信頼性問題は解決した．

もう 1 つの問題は大量のデータの読み書きを行うに要する時間である．この問題は本来読み書きするデータは各 PU に保持されているものであるので，それぞれの PU が並列に同時に読み書きを行うことが出来るようにするによって，解決される．そのため 3 次元の HXB ネットワークの $n_x \times n_y \times n_z$ 台から成る直方体の 1 つの側面にもう 1 つの IOU (input/output

unit) 用のプロセッサ平面を追加して，$n_x \times n_y \times (n_z + 1)$ の構成とするようにする．そしてこれらの IOU のそれぞれに RAID ディスクを接続する分散配置を行うこととした．

(7) 全体を小部分に分割した運用を可能としたのは，独立な小規模の問題などの場合にあたかも複数個の並列計算機が設置されているようなイメージで，運用することを考慮したためである．

(8) ソフトウェア関係については，応用プログラムはユーザが各自作成するのは当然のことである．しかしそのための OS，言語処理系などのシステムソフトウェアを，商用機のように完備したものとして CP-PACS 向きに揃えることは，開発期間から云っても，開発力・予算から云っても到底無理であることが当初から明らかであった．そこで必要最低限のシステムプログラムを開発する方針とした．

最低必要となるものとして，まずプログラミング言語，およびそのコンパイラ (compiler) がある．言語としては Fortran, C, および assembler 言語のみを取り上げることとし，少なくとも超並列計算機を構成する PU のアーキテクチャに追加された擬似ベクトルプロセッサ (PVP-SW) 機能をサポートした 1 台の PU 向きのコンパイラ (およびクロスコンパイラ (cross compiler)) は実現することを目指している．

並列処理向きの記述法 (言語仕様などの拡張) については，現在米国などにおいて並列処理ユーザ・メーカ間で広く精力的に仕様が論議されている HPF (high performance Fortran) を取り上げることにしている．

並列向きのプログラム開発用の環境としては，一般にも使用が始まりつつあると思われる Express に近いもののサブセット (subset) を想定し，まずこれの整備を目指す．ワークステーション上でのデバッグのためのシミュレータ (simulator) の用意も行う予定である．

OS に関しては，これも必要最低限の UNIX 系の OS カーネル (kernel) を備える予定である．これと結合ネットワーク通信用のライブラリ (library) 群を逐次整備していくことで，当初は対処することを考えている．

また磁気ディスク補助記憶装置を使用するためのファイル (file) の取扱い手段，FCS (front computer system) との間のファイル転送手段，ジョブ (job) やプロセス (process) の制御手段，運用上必要となる運用・管理ソフトウェアについても簡単なものを用意することを想定している．

(9) FCS (front computer system) との接続については，超並列計算機 CP-PACS 側に大容量の補助記憶装置を備えることにしているので，この補助記憶装置と FCS の間での高速のデータ転送手段が先ず必要とされる．このためピーク 100 MB/s 以上の転送を行う高速インターフェースでの接続を行い，ソフトウェアオーバヘッドも考慮しても実効的に 50 MB/s 以上のファイル間転送が可能なものとした．このため超並列計算機側は IOU の内の 1 台に高速の接続部を設け，FCS 側にも高速転送接続部を特に設けることとした．

さらに超並列計算機を使用するセンターシステムの全体の運用・管理が FCS より一元的に行えるようにし，ジョブの投入・実行制御・結果の取り出しなどの制御を行うために，FCS と超並列計算機の中の 1 台の IOU との間に Ethernet などの LAN をも用意することを予定している．

(10) 稼働開始時期については，基本設計，詳細設計およびパイロットボード (pilot board) などによる検討，実装設計および先行試作，全体装置の製作・調整などの必要期間を考慮して，1995 年度末 (1996 年 3 月) にハードウェアの据付けを行い，1996 年度に稼働を開始することを予定している．その後並行して準備されていたソフトウェアのデバッグを行い，1996 年度中には計算物理学の計算に試用可能なものとすることを目論んでいる．

(11) 並列計算機システムの信頼度については，先に補助記憶装置に関して述べたように，本並列計算機のように大規模なシステムでは信頼性を確保することはかなり難しい問題ではあるが，最近のコンピュータ技術の発展によるスーパコンピュータや大規模汎用システムの実績を考慮して，ハードウェアの MTBF (mean time between failure) を 2000 Hr 以上を目標とすることとした．

9 ノードプロセッサ・アーキテクチャ

9.1 従来のマイクロプロセッサの問題点

超並列計算機の各 PU のプロセッサとして有力な候補の一つは，一般のワークステーションに用いられているような高速の汎用 RISC マイクロプロセッサである．近年の汎用 RISC プロセッサの処理能力は，集積回路技術の進歩によるクロック周波数の向上と，スーパースカラ方式等に代表される命令レベルの並列性抽出による高速処理方式の実用化により，飛躍的に向上している．しかし，その高速性は「キャッシュが有効に働く」という仮定が成立する時においてのみ達成される。大規模な科学技術計算においては，データ領域が非常に大きく，またデータの時間的局所性が少ないという性質があるため，データキャッシュのミス率が大きい．このため，通常の汎用 RISC プロセッサの性能は，主記憶アクセスレーテンシによるペナルティのために大きく低下する．また，半導体技術の進歩により，プロセッサのクロック周波数は今後も飛躍的に向上することが予想される一方で，主記憶のアクセスレーテンシを決める大きな要因であるメモリ素子のアクセスタイムは，半導体技術の進歩によってもそれほど短縮しない．そのため，主記憶アクセスペナルティによる性能低下は，今後の半導体技術の進歩によっても改善されず，むしろさらに深刻な問題になっていくことが予想される．したがって，大規模な科学技術計算を高速に処理するためには，主記憶アクセスレーテンシを隠蔽する手法が必要不可欠である．

この問題に対する 1 つの解決法として，プリフェッチ機能をキャッシュに持たせ，必要なデータをあらかじめキャッシュにプリフェッチする手法が考えられる．しかしこの手法には，非連続アドレスアクセス時に同一キャッシュライン中の不要なデータもフェッチされるため主記憶とキャッシュの間に無駄なトラフィックが生ずる，データ参照に時間的局所性がない場合にはキャッシュのトラフィックが非常に厳しくなる，データプリフェッチの際に line conflict により有効なデータがキャッシュから追い出されてしまう，という 3 つの大きな問題点がある．

スカラプロセッサとベクトルプロセッサとの大きな相違点は，主記憶のデータ供給能力にある．ベクトルプロセッサでは主記憶アクセスがパイプライン化されておりそのスループットは高い．また，多数のベクトルレジスタを持つことによりベクトルロード / ストア処理のベクトル長を長くでき，チェイニングの機構によりこれらの処理と演算処理とを平行に行うことができる．これにより，レジスタへのプリロード機能を実現でき，主記憶アクセスレーテンシが性能に与える影響をかなり隠蔽できる．しかしながら，スカラプロセッサの主記憶はパイプライン化されておらず，スループットは低い．そしてキャッシュミス時には主記憶アクセスペナルティにより実効

性能は大きく低下する．しかしながら，両者とも複数のパイプライン化された演算ユニットを持ち，複数命令同時発行 (スカラ) あるいはチェイニング (ベクトル) といった，複数の演算ユニットを有効に働かせる機構を持つ点では類似している．従ってスカラプロセッサにおいても，主記憶アクセスペナルティを隠蔽することができれば，1 つのベクトル命令の処理内容を，複数のスカラ命令によって垂直マイクロプログラミング的に処理することにより，高速にベクトル処理を行うことができると考えられる．

9.2 擬似ベクトルプロセッサ PVP-SW

CP-PACS のプロセッサ・アーキテクチャは，科学技術計算を極めて高い効率で並列処理できるように考えられている．

CP-PACS では，前節で述べたようなベクトルプロセッサが実現する機能 / 特徴を，スーパスカラ方式のプロセッサに機能拡張を施して実現することを考える．この処理方式を我々は擬似ベクトル処理 (Pseudo Vector Processing) と呼ぶ．擬似ベクトル処理を実現するためには，前節で述べたスカラプロセッサとベクトルプロセッサの相違点である以下の 3 点を実現する必要がある．

- 多数の浮動小数点レジスタ
- レジスタへのプリロード機能の強化
- 主記憶アクセスのパイプライン化

我々の設計方針は，ソフトウェアの開発コストを抑えるべく，既存の RISC アーキテクチャとの上位互換性を保ちながらこの 3 つの機能 / 特徴を実現することである．第 3 の点は，主記憶を多数のバンクに分割し，インターリーブメモリを構成することにより擬似的に実現できる．しかしながら，第 1 の点である多数のレジスタを実現するためには，命令フォーマット中のレジスタ指定フィールドのビット数を増やす必要があるため，通常は命令セットアーキテクチャの互換性は失われる．

そこで，既存の RISC アーキテクチャとの上位互換性を維持しながら，多数のレジスタを利用できる「スライドウィンドウ方式に基づく擬似ベクトルプロセッサ **PVP-SW** (**P**seudo **V**ector **P**rocessor based on **S**lide-**W**indowed registers)」を提案する [8]. PVP-SW のアーキテクチャは既存のスカラプロセッサに対する拡張という形で定義される．拡張点は，浮動小数点レジスタのスライドウィンドウ化と数種類の命令追加である．

スライドウィンドウ構成　図 9に，浮動小数点レジスタのスライドウィンドウ構成を示す．物理的なレジスタ空間は複数の論理的なウィンドウに分割される．1 つの論理的なウィンドウ内のレジスタ数は，拡張前のアーキテクチャで定義される数 (図 9では 32 とした) と等しくなければならない．ウィンドウの位置は **window-offset** で与えられる．各ウィンドウは global 部と local 部にわかれ，global 部のレジスタは全てのウィンドウに共通である．図 9では，global 部のレジスタは g 個とした．

ある時点ではただ 1 つのウィンドウのみがアクティブであり，このアクティブウィンドウを指定するポインタ **FWSTP**(**F**loating-point **W**indow **ST**art **P**ointer) を導入する．拡張前のアーキテクチャにおける命令は全て FWSTP で指されるアクティブウィンドウ内のレジスタのみを用いる．一方，ソフトウェアで FWSTP の値を変更することにより，このアクティブウィンドウを物理的ウィンドウ空間内で自由に移動 (スライド) することが可能である．ウィンドウ内のレジス

タ数は拡張前のアーキテクチャに依存するが，物理レジスタの数 m はハードウェアが許す限り任意の個数実装することが可能である．このようにして，上位互換性を維持しながら多数のレジスタを利用可能となる．

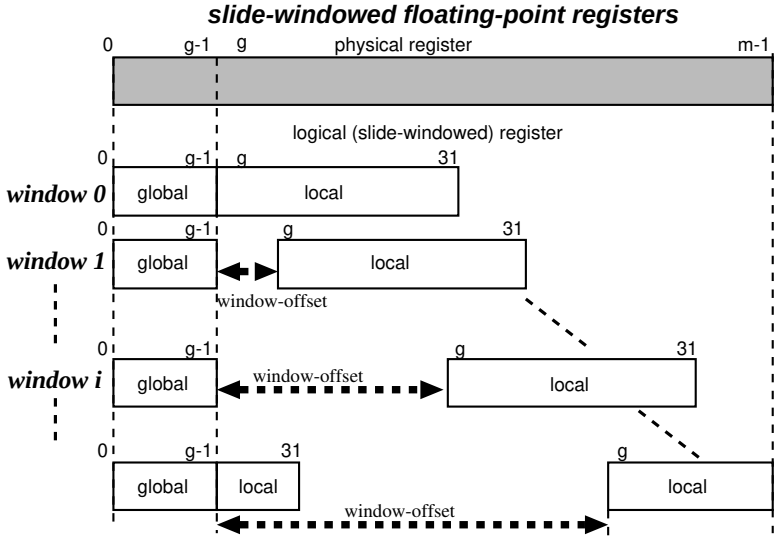

図 9: PVP-SW における浮動小数点レジスタの構成

追加命令　以下の命令が追加される．

FWSTPset アクティブウィンドウを指す FWSTP の値を変更し，アクティブウィンドウを切り替える．

FRPreload データをメモリから任意に指定されるウィンドウ内のレジスタにロードする．

FRPoststore データを任意に指定されるウィンドウ内のレジスタからメモリにストアする．

FRPreload 及び FRPoststore 命令は主記憶とレジスタとのデータ転送を行う命令であるが，以下の 3 つの特徴を持つ．

1. 置いてきぼり処理」により，データ転送の完了を待たずに後続命令は実行される
2. アクティブウィンドウに関係なく，全物理レジスタをデータ転送の対象として指定できる
3. キャッシュミス時には主記憶とレジスタの間で直接データを転送し，キャッシュ内データを変更しない

1 番目の特徴により，主記憶のアクセスレーテンシの影響でデータのロードやストアの実行に長い時間を要したとしても，それらの動作を放っておき，後続命令を実行できる．このため，将来必要となるデータに対するプリロード命令を先行発行することで，他の命令実行を妨げることなく該当データに対する主記憶アクセスレーテンシを隠蔽できる．より長いレーテンシを隠蔽するためにはそれだけプリロード命令を早く発行する必要があり，必要なレジスタ数が増加する．こ

の問題は物理的に多数のレジスタを用意すること，及び 2 番目の特徴で解決する．3 番目の特徴は，キャッシュのスループットが十分で無い場合に性能が低下するのを防ぐため，及びキャッシュコヒーレンシ問題に対処するためである．これらの特徴により，PVP-SW 方式が主記憶アクセスレーテンシを効果的に隠蔽し極めて高い性能を示すことを，各種ベンチマークの評価によって確認している．

9.3 PVP-SW における擬似ベクトル処理例

DAXPY ループ '$Y(i) = a \times X(i) + Y(i)$' を例として，PVP-SW における擬似ベクトル処理を説明する．図 10にオブジェクトコードを示す．説明を簡単にするため，分岐命令は省略してある．図 10中の '`FRPreload Y(i+1) -> r30<+2>`' は，現在アクティブであるウィンドウの 2 つ先のウィンドウ (そのウィンドウの window-offset は，アクティブウィンドウの window-offset より 2 大きい) の論理レジスタ `r30` に `Y(i+1)` のデータをプリロードすることを表す．また，最後にある '`FWSTPset +2`' は FWSTP の値を 2 増やすことを表す．

```
Loop:  FRPreload     Y(i+1) -> r30 <+2>
       ADD           r29 + r30 -> r29        % aX(i) + Y(i)
       FRPreload     X(i+2) -> r31 <+2>
       MULT          r7*r31  -> r31          % a * X(i+1)
       FRPoststore   r29 <-0> -> newY(i)
       FWSTPset +2   (FWSTP <- FWSTP + 2)
       % branch is omitted
```

図 10: PVP-SW における DAXPY ループのコード例

図 10のオブジェクトコードを実行する時のレジスタ割り当てを図 11に示す．図 11ではループ処理が時間の経過と共に展開されて示されている．6cycle ごとの点線は，アクティブウィンドウが切り替る事を表す．影付きの部分は `Y(i+2)` の計算に関係する部分である．

図 11よりわかるように，ループ毎にウィンドウを切り替えていくことにより各ループは異なるウィンドウ上で処理される．従って同一の論理番号を持つレジスタも，ループ毎に物理的には異なるレジスタに対応することになる．このようにして 32 レジスタしか指定できない演算命令が多数のレジスタを用いて処理を行なうことができる．

主記憶アクセスレーテンシの隠蔽　あるデータに対するロード要求から，そのデータが演算命令により最初に使用されるまでの時間間隔を，そのデータに対する *permitted latency* と呼ぶ．例えば図 11では `X(i+2)` に対する *permitted latency* は 7cycle である．全てのデータに対する *permitted latency* が主記憶アクセスレーテンシ以上であれば，主記憶アクセスレーテンシによる性能低下を隠蔽することができる．図 11において，*permitted latency* の最小値は 7cycle であるので，7cycle 以下の主記憶アクセスレーテンシを隠蔽できることになる．より長い主記憶アクセスレーテンシを隠蔽するためには，*permitted latency* を長くする必要がある．そのためにはできるだけ遠くのウィンドウにできるだけ早くデータをプリロードすれば良い．具体的には，図 10における '`FRPreload Y(i+1) ->r30<+2>`' 及び '`FRPreload X(i+2) ->r31<+2>`' の下線部を例えば，'`FRPreload Y(i+2) ->r30<+4>`' 及び '`FRPreload X(i+3) ->r31<+4>`' に変更すれば良い．この変更では，プリロード命令がさらに 1 ループ先行して発行されることになるため，結果

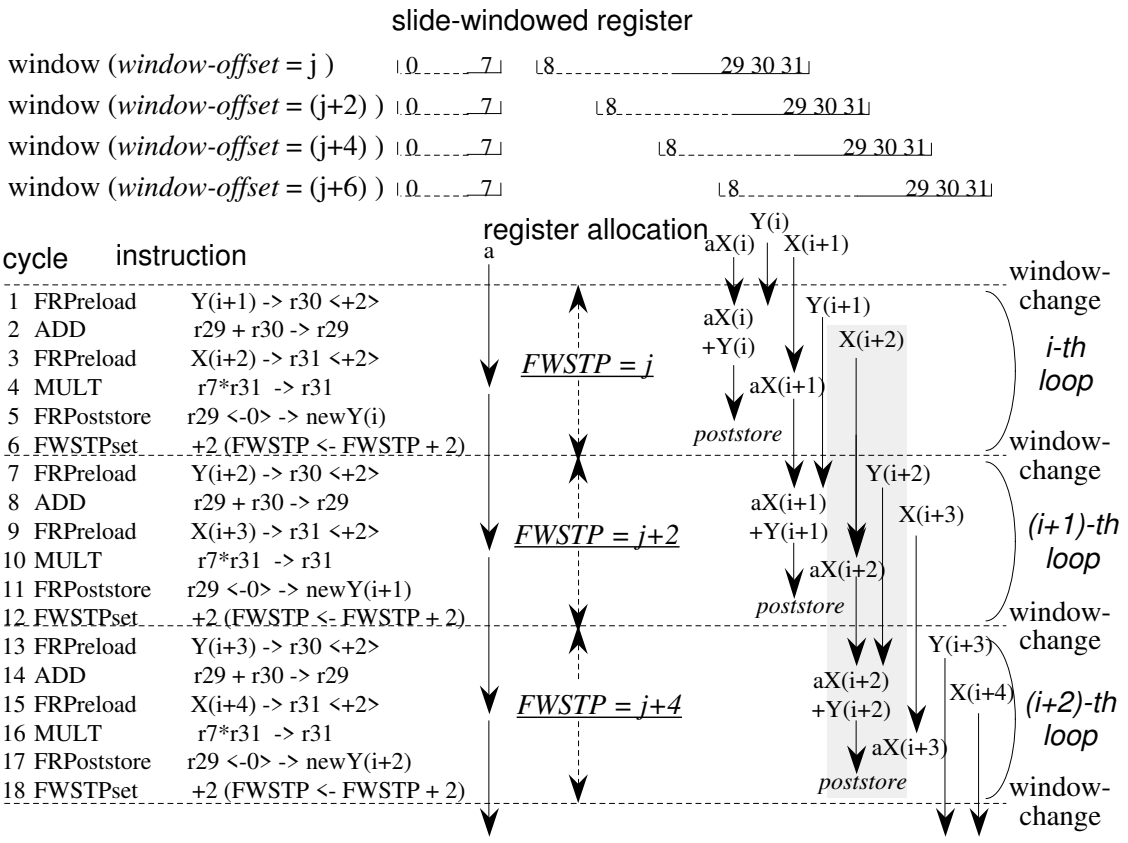

図 11: PVP-SW におけるレジスタ割り当て

的に $permitted\ latency$ は

$$7(\text{変更前の } permitted\ latency) + 6(1 \text{ ループの実行に要する時間}) = 13cycle$$

になる．しかし，このようにより遠くのウィンドウにデータをプリロードするためには，より多くの物理レジスタが必要になる．

9.4 CP-PACS で採用予定のプロセッサ

CP-PACS では Hewlett Packerd 社の PA-RISC 1.1 アーキテクチャにこの PVP-SW 機構を追加し，PU の中心となるプロセッサとする．PA-RISC 1.1 は 32 本の倍精度浮動小数点レジスタを持っており，これを拡張して物理レジスタ群とウィンドウ制御回路 (主に論理レジスタ番号と物理レジスタ番号の変換をする) を追加する．実装する物理レジスタの数は隠蔽すべき主記憶アクセスレーテンシと実装上の制約に依存するが，現在のところ 100 本あるいはそれ以上を予定している．また，グローバル ・レジスタの本数は 8 本程度を考えているが，アプリケーションによってはもう少し多くのグローバル ・レジスタを用意することによって処理効率を高くすることが可能であるため，ある程度の範囲でこれを可変に制御できるように考えている．

ハードウェアコストについて簡単に触れる．まず，数十の物理的なレジスタの追加だが，レジスタの入出力のポート数を増加する必要がないため，現在の実装技術から考えてこの程度のレジスタには特に問題がないと考えている．ウィンドウ制御回路については，面積よりもむしろプロセッサのマシンサイクルに与える影響を考慮すべきであるが，実装されるレジスタ数が 128 本以下であれば論理レジスタ番号と物理レジスタ番号の変換に必要なのは高々7bit の加算器であり，この点も問題がないと考えている．

10 ネットワーク

本項では CP-PACS のネットワーク ・アーキテクチャについて，その概略を述べる．CP-PACS は大きく分けて，データ転送を目的とするデータ転送ネットワークと，PU 間のバリア同期を目的とする同期用ネットワークの 2 種類のネットワークを持つ．以下，各々のネットワークの特徴と，高速データ転送のための特殊な機構について述べる．

10.1 データ転送ネットワーク

超並列計算機向けのネットワーク ・トポロジーとしては，Mesh/Torus，Binary n-Cube，多段結合網 (Multistage Interconnection Network)，等いろいろと提案されているが，CP-PACS ではデータ転送ネットワークとして Hyper-Crossbar ネットワークと呼ばれるトポロジを採用する (以下，HXB と略す)[9]．

CP-PACS 上で QCD 計算を行う際に必須要求となっているのが，隣接 PU (Processing Unit) 間でのデータ交換が無衝突で行われることである．すなわち，全 PU が一斉に同一方向の隣接 PU に対してメッセージを転送した場合，それらのメッセージはネットワーク上で互いに衝突することなく，ピーク性能に近いスループットで転送される必要がある．この要件を満たすためには QCD-PAX で用いられた Torus 型ネットワークを用いるのが最も簡単であるが，CP-PACS では単純な隣接交換だけでなく，より複雑なデータ転送パターンにも対応可能な，より柔軟なトポロジを持つネットワークとして HXB を採用する．これは，QCD 以外の広汎な科学技術計算にも柔軟に対応可能な計算機の設計を目指すことにつながる．

さらに，HXB ではHXB そのものの物理的なプロセッサの配置と同等なMesh/Torus だけでなく，総プロセッサ台数が等しく，構成の異なるようなMesh/Torus における隣接転送をも容易にシミュレートすることが可能である．例えば，$8 \times 8 \times 16 = 1024$ プロセッサの3 次元HXB 上では，同サイズの3 次元Mesh/Torus だけでなく，$4 \times 8 \times 32$ のように各次元のサイズが違うものや，32×32 のように次元数そのものが違うようなMesh/Torus に関しても，その隣接転送を無衝突でシミュレートすることができる．このことは，ユーザがプロセッサの結合形状に合わせて問題を特化させることなく，その性能を最大限に利用できるという利点がある．

HXB のトポロジはn 次元の超立方体の概念を拡張した形としてとらえることができる．ここで次元数n は任意に選択可能であるが，CP-PACS 実装時の実装技術を想定すると，3 次元の構成が妥当であると考えられる．次に，各次元のPU のサイズを決定する．これは全PU を3 次元の直方体の形に配置することに相当し，例えば1024 PU の場合，典型的な構成は$8 \times 8 \times 16$ という形になる (直方体であるため，各次元のサイズが一致している必要はない[1])．例として，$4 \times 4 \times 4 = 16$PU の場合を図12に示す．

一般的な3 次元HXB の特徴について，$X \times Y \times Z$ 構成の場合を想定して説明する．まず，全PU は$X \times Y \times Z$ の3 次元正方格子状に配置される．そして，1 つの次元方向 (例えばx 次元方向) に並んだPU 同士をクロスバ・スイッチ (以下，XB と省略する) で完全結合する．例えばx 次元方向の場合，X入力 X出力 のXB が必要となる．このようにして全ての次元方向について，その次元方向に並んだPU 間をXB で結合する．この場合，次元数は3 なので，各PU は3 つの次元方向のXB に対するリンクをそれぞれ持つことになる．

このネットワーク上でのメッセージ転送を考える．以下，座標(x, y, z) の位置にあるPU を$PU(x, y, z)$ と表すことにする．今，$PU(x_s, y_s, z_s)$ から $PU(x_d, y_d, z_d)$ までメッセージを転送するとする．最も簡単な方法としては，まず $PU(x_s, y_s, z_s)$ から $PU(x_d, y_s, z_s)$ までメッセージを送り，続いてそれを $PU(x_d, y_d, z_s)$ へ，そして最後に $PU(x_d, y_d, z_d)$ へ，というように3 ステップの転送を繰り返して送ればよい．

しかし，この方法では3 ステップの転送を繰り返すため転送遅延が大きいので，より高速なメッセージ転送を行うため，Wormhole 方式のメッセージ転送方式を行う．これは，中継点の各PU が一旦メッセージを蓄えるのではなく，その位置のPU を経由せず，あるXB から別のXB へのメッセージの直接転送を行うことにより，上記3 ステップの転送を一気に行う方法である．このために，各PU とXB を結合する部分にEX (Exchanger) と呼ばれるルータ・スイッチを置く．EX はPU → 任意のXB，あるXB → 別のXB，あるXB → PU というように，メッセージが全ての方向のPU とXB の間を自由に乗り替えられるようにするスイッチで，その制御はメッセージが持つヘッダ情報によって自動化されている．また，メッセージ転送経路の確保は，メッセージのヘッダ部がEX またはXB を通過する段階で行われるので，経路確保のための初期オーバヘッド等を小さくすることができる．

Wormhole 方式のHXB には以下のような特長がある．

- 通信距離が短い．これは短いメッセージを頻繁に転送する際に有利である．
- 同一サイズの正方格子結合 (Mesh/Torus) を直接無衝突でエミュレートできる．さらに，サイズが異なる場合でも，総PU 数が等しいか小さければ，それをエミュレートできる．
- Binary-n-Cube ネットワークのエミュレートが容易である．

[1] 各次元方向のサイズを 定数 m として等しくしたものは base m- n cube として知られている

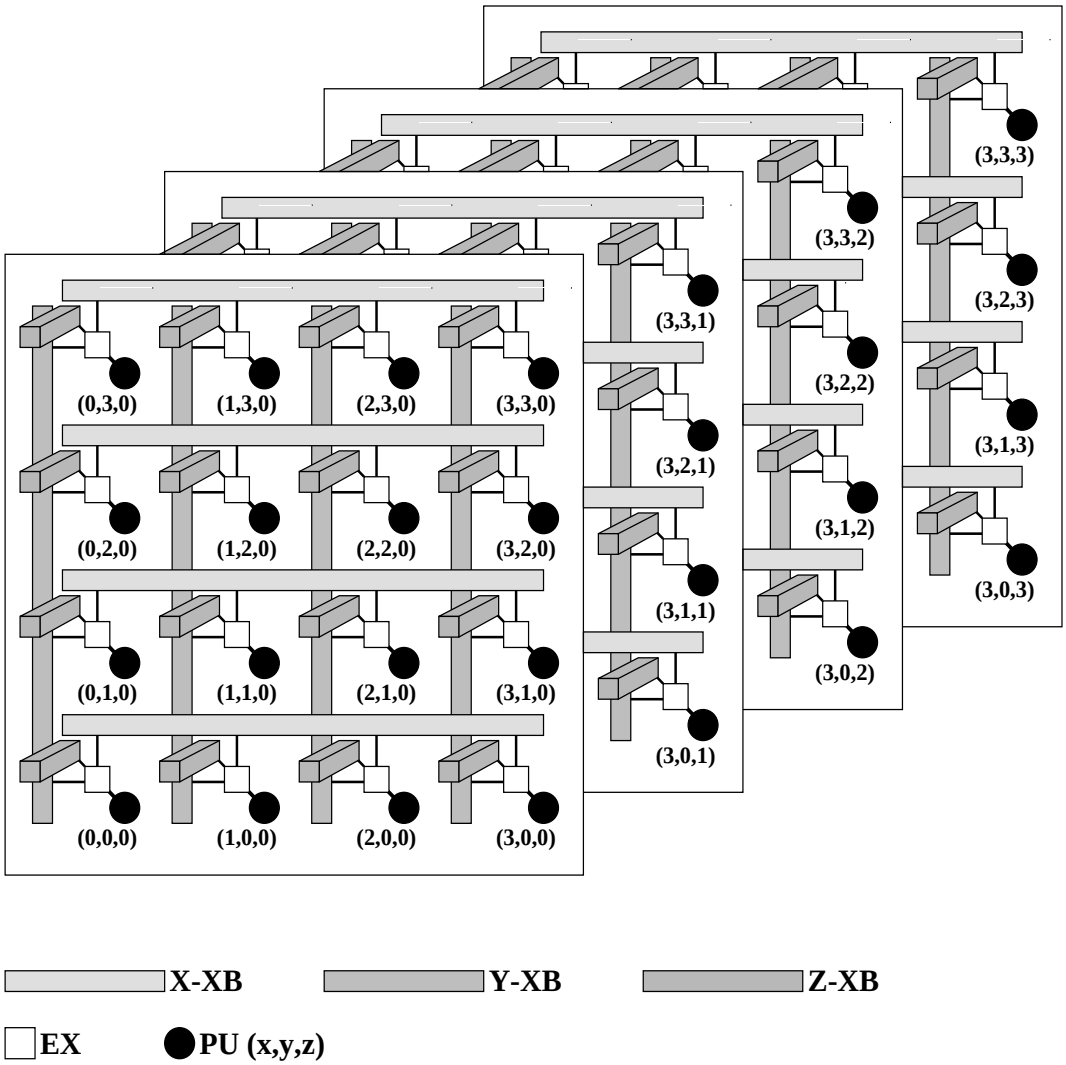

図 12: $4 \times 4 \times 4 = 64$ PU 構成の 3 次元ハイパクロスバ ・ネットワーク

- 各 XB の出力を全開放にすることにより，ブロードキャスト転送が容易に行える (通常の 1 対 1 転送と同じ時間で可能)．
- ランダム転送 (送受信の PU の対が動的にランダムに決定するような通信) におけるスループットが，他のネットワークに比べ非常に高い．

Mesh/Torus ネットワークの隣接転送を無衝突で行える点は，多くの科学技術計算において重要である．さらに，対象とする問題がある程度任意の大きさの場合でもエミュレーションが可能であるため，プログラミングの自由度という点から見ても，ユーザにとっては非常に使い易く，この点は QCD 計算において重要であると思われる．また，ブロードキャストや全対全通信についても，Mesh/Torus ネットワークに比べ非常に高速に処理することができる．

一般に，n 次元 HXB では，最短経路の転送を行う場合，$n!$ 通りの経路が存在する．これらの経路の選択方法には，静的にそれを決定しておくもの (固定ルーティング) と，ネットワークの混雑状況等を評価しながら動的にそれを決定するもの (適応ルーティング) の 2 種類がある．Wormhole 方式の HXB において，単純な適応ルーティングを行った場合，メッセージ同士がネットワーク上でデッドロック (メッセージ同士が互いに相手をブロックし合い，ネットワークが停止してしまう状況) を起こす可能性がある．この問題は仮想的に多重化した転送経路を用意することで解決できるが，現在計画されている CP-PACS 用 HXB ではハードウェア・コストを抑えるため，固定ルーティング方式を用いている．

10.2 データの高速転送のための機構

最近の超並列計算機におけるネットワーク性能においては，メッセージの転送スループットだけでなく 1 つのメッセージの転送の立ち上げ時間に対する配慮がなされるのが普通である．この時間をセットアップ時間と呼ぶ．1 つのメッセージ長が非常に長い場合，セットアップ時間の影響は無視することができるが，メッセージ長が短くなるにつれ，メッセージの転送開始から終了までの時間のうちセットアップ時間が占める割合が大きくなってしまう．QCD 計算においては比較的長いメッセージが扱われるが，問題の範囲をより広汎な科学技術計算に広げると，この問題が重要になってくる．

CP-PACS では高速なメッセージ転送立ち上げを実現するために，ユーザ・アプリケーションから直接ネットワークへのメッセージ転送起動が行えるように，特別な機構を用意する．この機構を用いる転送を高速転送モードと呼ぶ．この様子と，通常転送との違いを図 13及び図 14に示す．

通常の転送では，メッセージ転送要求は各 PU 上のオペレーティング・システムによって処理され，ネットワーク・インタフェース (NIA: Network Interface Adapter) に対する命令の発行やメッセージ到着処理は全てオペレーティング・システムによって行われる．ユーザ・アプリケーションは転送終了まで待たされ，その後で処理が継続される．メッセージ転送はオペレーティング・システムが管理するメモリ上で行われる．まずアプリケーションのメモリからオペレーティング・システムのメモリへデータがコピーされ，それがネットワークを通じて相手 PU のオペレーティング・システムのメモリに転送される．その後，相手側のアプリケーションが再度オペレーティング・システムからデータをコピーし，受信が完了する．この方式では，アプリケーションからオペレーティング・システムに処理が移行する際の処理が非常に重く，結果としてメッセージのセットアップ時間が無視できない大きさになってしまう．

高速転送モードでは，オペレーティング・システムを経由せず，アプリケーションが直接 NIA にアクセスし，データをネットワークに送出する．一般に NIA はアプリケーションが誤ったメッ

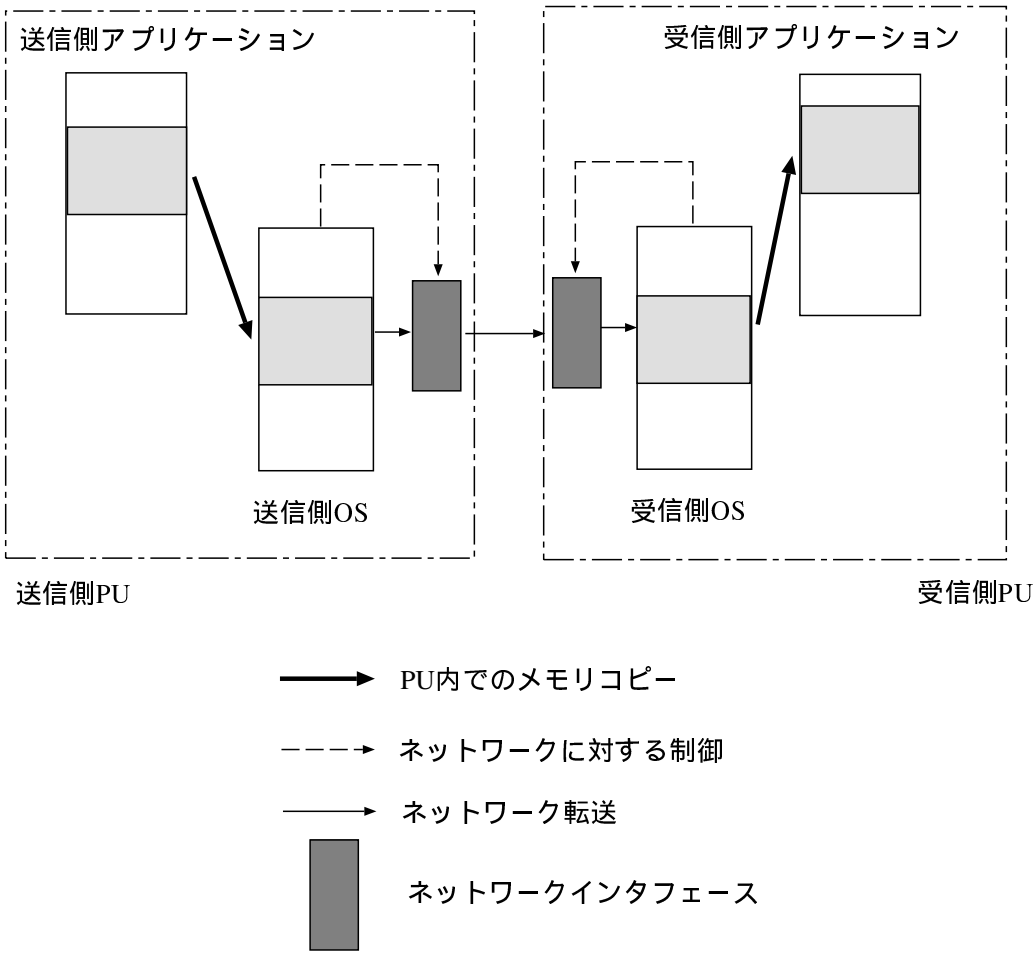

図 13: 通常転送モードにおけるデータと制御の流れ

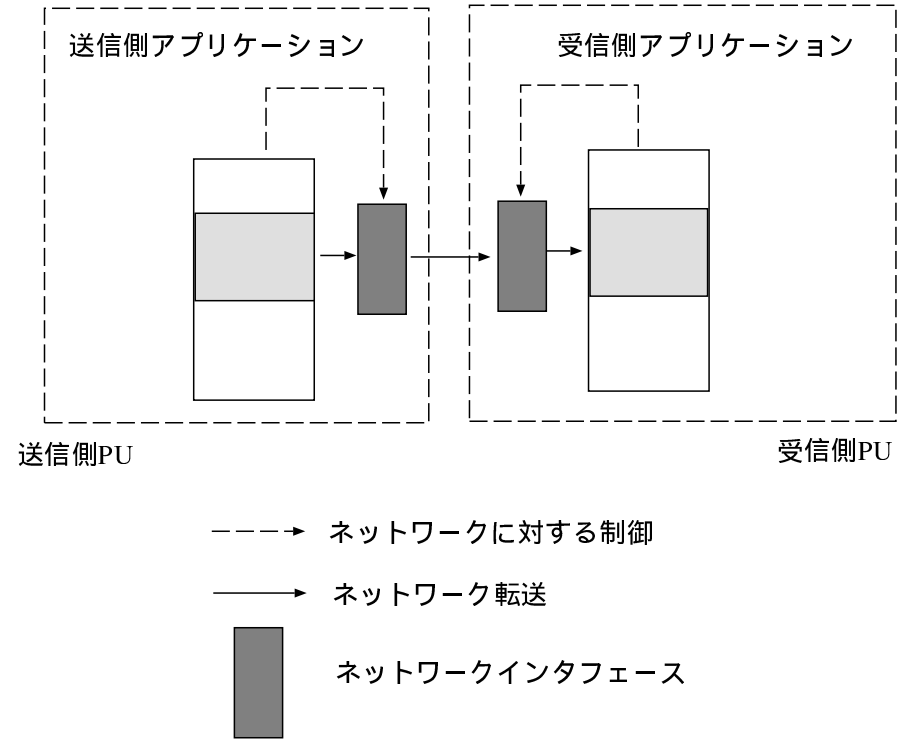

図 14: 高速転送モードにおけるデータと制御の流れ

セージを送出しないように (例えば処理と無関係な PU への送出や，システム上で転送を禁止されている PU への送出等)，ハードウェアによる保護がなされている．高速転送モードではこの保護を一部解除し，システムに破綻をきたさない範囲でアプリケーションが直接通信を行うことを許可する．オペレーティング ・システムを介さないことにより，以下の 2 つの利点が得られる．

- オペレーティング ・システムへの処理の移行に伴うオーバヘッドが解消される．
- 送信側アプリケーションのメモリ空間から受信側アプリケーションのメモリ空間に対して直接データが転送されるため，データ ・コピーの時間が短縮される．

メッセージ長が短い場合は第 1 の利点が，メッセージ長が長い場合は第 2 の利点がそれぞれ活かされ，結果として高速なメッセージ交換が可能となる．

10.3 メッセージのブロードキャスト転送

並列処理においてしばしば現れる特殊なメッセージ交換の例としてブロードキャスト通信がある．これはある PU から他の全ての PU に対して同一メッセージを一斉に転送するもので，ある数値を集計しその結果を分配する場合等，多くのアルゴリズムで多用される．そのため，CP-PACS では HXB の特性を活かし，ブロードキャスト ・メッセージを高速に処理できるような設計がなされている．

HXB においては，ある XB の入力に届いたメッセージを，その XB の全出力に同時に送出することにより，単一次元内でのブロードキャストが容易に実現できる．これを全次元に対して行

えば，全 PU に対するブロードキャストが実現できる．CP-PACS では $x \rightarrow y \rightarrow z$ という固定的な経路決定を用いるので，ある PU から送られたメッセージがまず x 次元方向の XB においてブロードキャストされ，続いてそれらのメッセージが y 次元方向の XB においてブロードキャストされる．そして，最後にそれらのメッセージが z 次元方向にブロードキャストされることにより，全 PU に対するブロードキャストが行われる．

ブロードキャストにおいて注意すべき点は，HXB ではその構造上，複数の PU からの別々のブロードキャスト要求を同時に処理することが不可能であるということである．上述のブロードキャスト ・アルゴリズムを HXB 上で複数のメッセージに対して実行すると，ネットワーク上でデッドロックを生じる．これに対する対策として，例えばある PU だけがブロードキャスト ・メッセージを流すことができるという制限を設け，他の PU からの要求は全て一旦その PU に送ってから改めて処理するという方法が考えられる．この方法ではブロードキャスト要求が逐次処理されるため，ブロードキャスト ・メッセージが同時に複数存在することがなくなる．現在は基本的にこの手法を元に設計を行っている．

10.4 ストライド転送

一般的なメッセージ転送では，各 PU のメモリ上で 1 次元の連続領域上に配置されているデータが一つのメッセージとして転送される．メモリ上の連続データは，PU が NIA にメモリ上でのデータの開始番地とデータの長さをセットすることにより，自動的にネットワーク上に送出され相手 PU に届く．

しかし，問題空間の形状によって，あるいは問題を解くアルゴリズムによっては，一つの PU に一度に転送したいデータが，必ずしも送信 PU 側のメモリの連続領域に格納されているとは限らない場合がある．逆に受信 PU 側で，送信 PU から送られてきたデータをメモリの連続領域ではなく，飛び飛びの領域に格納したい場合がある．一例として，2 次元空間上のデータ列をメモリ上に配列として配置した場合，どちらか一方の次元方向のデータはメモリ上で連続領域に格納されるが，もう一方のデータは必ず一定間隔を空けた番地に配置される．この時，第二の次元方向で連続なデータ群をある PU に転送しようとすると，メモリ上で (周期性はあるが) 不連続な番地上のデータ群を転送することになる．

一般の超並列計算機では，メッセージのストライド転送はユーザのメモリ空間からオペレーティング ・システムのメモリ空間にデータをコピーする際にソフトウェア的に行うことができる．すなわち，ユーザのメモリからストライドを意識しながらデータをコピーしパッキングしたものをメッセージとして送出する．しかし，CP-PACS においては，高速転送モードによるユーザ空間からの直接転送がサポートされており，この転送モードではユーザ空間から直接 NIA にデータが送られるため，ソフトウェア的なパッキング手法が使えない．そのため，NIA がハードウェア的にこのような処理を行う必要がある．その結果，高速転送モードにおいても柔軟なストライド転送処理が実現でき，各種データ配列を高速に処理することが可能になっている．もし NIA が単純な連続転送機能しか持っていないとすると，このような転送はデータ一つずつの転送の繰り返しとして処理しなければならず，転送時間のほとんどがそのセットアップによって消費されてしまい，処理効率低減の重大な要因となる．

そこで CP-PACS では，このようなメモリ上で周期性を持つ不連続番地に配置されたデータ群を，一度のセットアップによってネットワーク転送するための機構が NIA に用意されている．このようなデータ転送方式をストライド転送と呼ぶ．ストライドとは不連続なデータの間隔のことである．一般の連続転送では，NIA に対してメモリ上のデータの開始番地とデータ長のみをセッ

トするが，ストライド転送においては，データの開始番地，各データのサイズ，データとデータの間隔 (ストライド)，全転送データ長といったパラメータをセットする．ストライド転送が始まると，NIA は指定された間隔でデータをメモリから取り出し，ネットワークに送出する．この時，ネットワーク上では (元のメモリ上の番地が不連続であっても) データは密に送られるため，データ転送に要する時間は実際に送信するデータ量に対してかかる分だけで済む．

10.5 同期ネットワーク

QCD-PAX においては，隣接 PU 間での共有メモリ方式によって PU 間のデータ交換が行われてきた．このため，全 PU によるバリア同期はアルゴリズム上不可欠なものとなっていた．CP-PACS でもその流れを汲み，同期用ネットワークを用意している．一般的に，メッセージ・パッシング系の PU 間交信では，メッセージの送受信処理そのものが point-to-point の同期操作を同時に実現しているため，その他の同期処理が不要な場合が多い．しかし，現実的には並列プログラムのデバッグ時や，非均質なデータ転送を行った後で一斉転送を行う場合等，ある程度の粒度で PU の同期をとりたい場合が存在する．そこで，CP-PACS ではこのような場合に簡単に全 PU でバリア同期をとることができるような機構を用意している．

CP-PACS における同期ネットワークは，基本的にデータ転送ネットワークを部分的に用いることにより実装される．具体的には，全 PU で一斉同期をとる必要が生じた場合，各 PU は同期用の特殊なメッセージをデータ転送ネットワークに流す．HXB 上の各 XB では，全ての入力からの同期用メッセージの待合わせが行われ，全入力からのメッセージが揃った時点で次の段の XB に送られる．これが繰り返されることにより，最終段 (z 次元方向 XB) にメッセージが揃った段階で全 PU の同期が確認され，それが全 PU に通達される．

このように，同期ネットワークは仮想的にデータ転送ネットワーク上にマップされているため，その動作が本来のデータ転送の妨げにならないよう配慮しなければならない．このため，データ転送ネットワーク上には，同期用メッセージを保持するための専用バッファをデータ保持用のバッファとは別に確保しており，互いの通信が円滑に進むような構造になっている．これにより，少なくともデータ転送メッセージと同期用メッセージがデッドロックを生じることはない．

11 システム構成概念図

この節では、システム構成の概略を図示する。PU(Processing Unit) と IOU(input/output unit) の構成概念図を図 15に示した。Instruction Processor は節 9に詳述した今回新たに開発するスライドウィンドウ機能による擬似ベクトルプロッセッサである。Local Storage として DRAM を使用予定で、１個の PU あたり 64 Mbytes の容量を持つ。Storage Controller は メモリをパイプライン的に制御する。IOU には、これ以外に、分散磁気ディスク記憶装置などを接続するためのアダプターがある。

図 16に示したように、PU が３次元ハイパークロスバーの各点に位置し、IOU が３次元直方体の１つの側面をなし、分散磁気ディスク記憶装置が IOU に接続している。

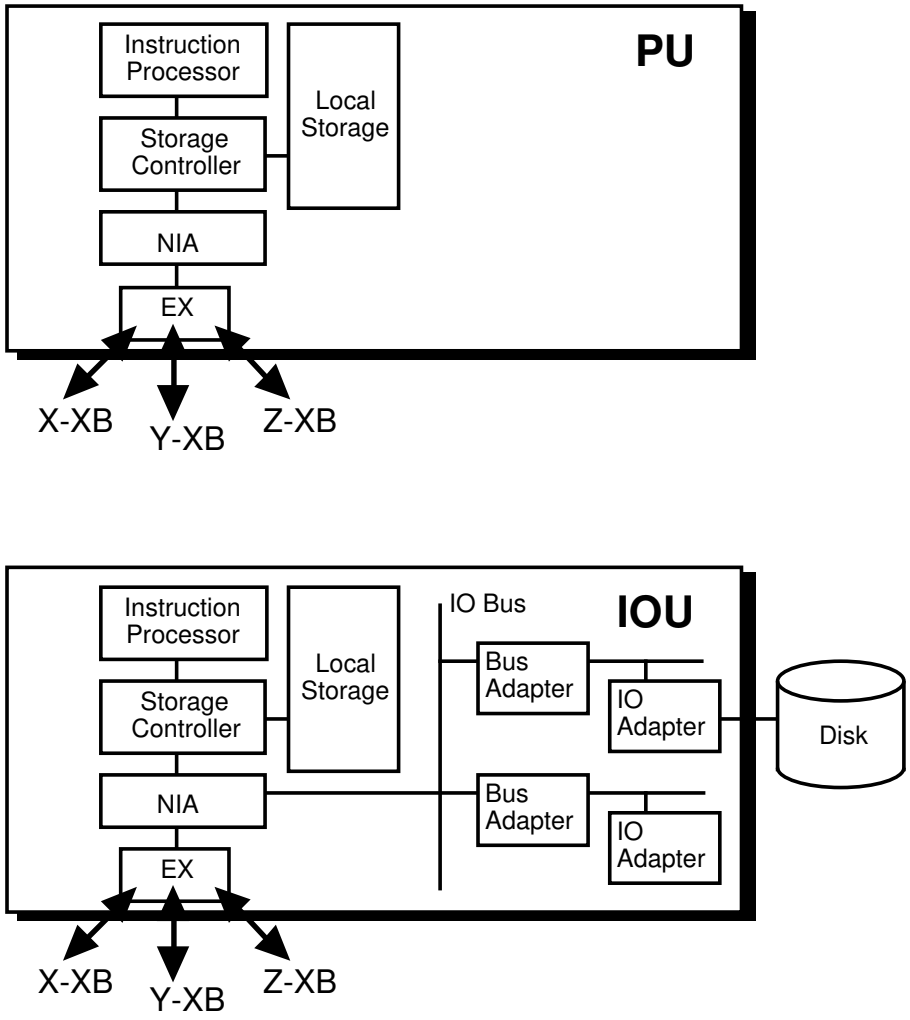

図 15: PU と IOU の構成概念図

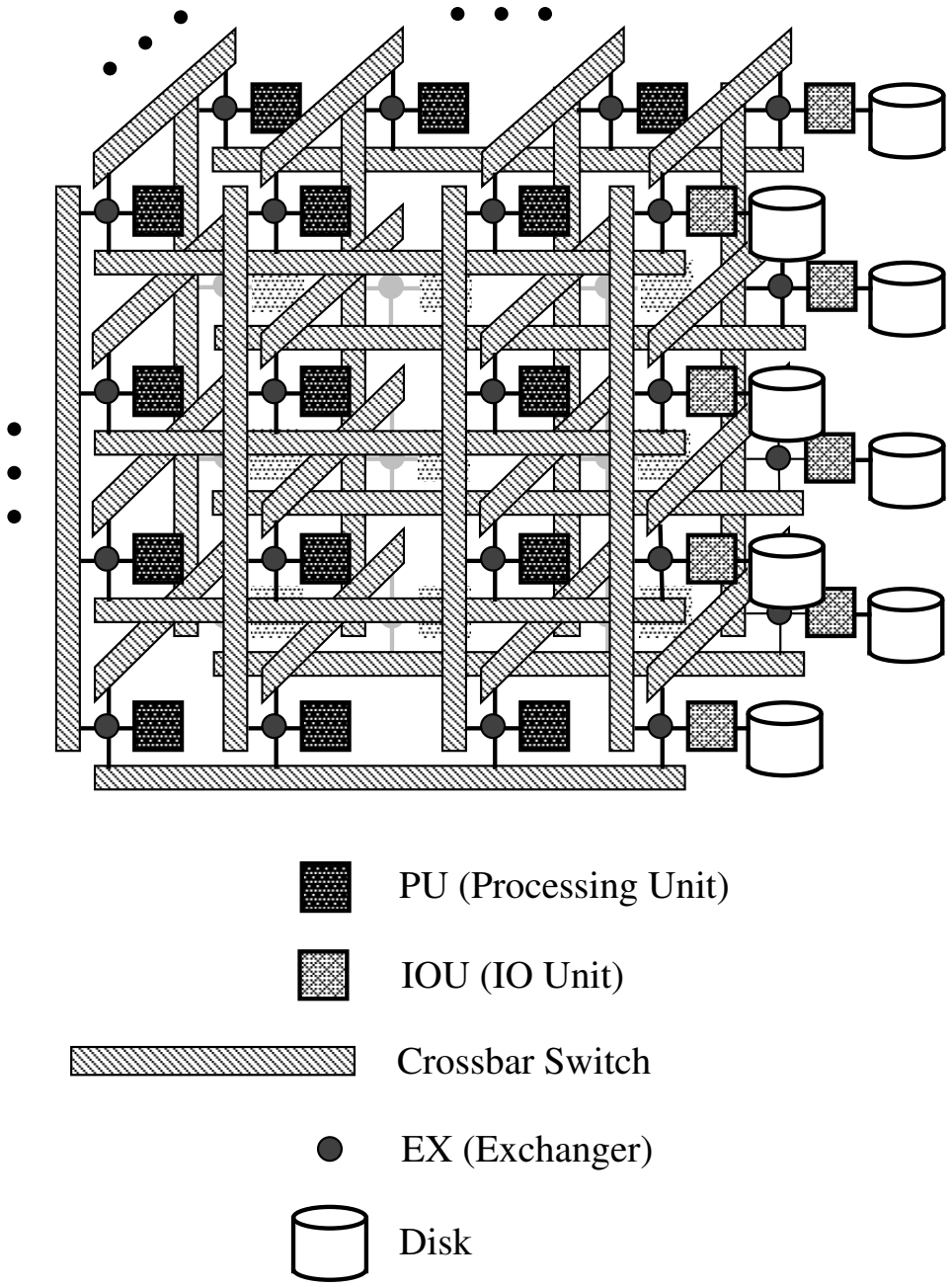

図 16: CP-PACS システム構成概念図

12 ソフトウエア

CP-PACS ソフトウェアの体系とその中の各ソフトウェアの機能の基本設計をした．その体系は以下のようなものである．

1. OS
   (a) 各 PU の基本 OS(UNIX 系の Mach をベース)
   (b) 並列実行 OS 機能 (PU 群、プロセス、ファイルに関する機能)
   (c) プロセス間通信機能
2. プログラミング環境
   (a) プログラミング言語 (C, Fortran, HPF)
   (b) プログラム開発・デバッグ機能

以下，このソフトウェア体系の中で，重要項目として検討を進めてきたものについて記す．

12.1 並列実行 OS 機能

並列実行の機能としては，プログラムを各 PU 上で並列に実行する機能が最も重要であり，さらに，複数台の PU に分散しているディスクに並列にアクセスする機能が重要である．

12.1.1 PU 群/プロセスに関する機能

一つのユーザプログラムは，一般には，複数の PU 上で並列に実行される複数のプロセスとして実行される．また，同時に複数のユーザのプログラムを実行させることも必要である．これらのことを可能にするために，以下の機能を設けることにし，その詳細仕様を定めた．

1. PU 分割の機能: たとえば，大規模並列計算用の大 PU 集合と，デバッグ用の小 PU 集合に分ける．分割された PU 集合はパーティションと呼ぶ．この分割の指定はシステム管理者が行う．
2. プロセス生成の機能: ユーザプログラムは一つのパーティションの中で実行される．ユーザプログラムでは，そのパーティションの中の PU 群を指定し，その各 PU の中にプロセスを生成することが出来る．この PU 群の境界がハードウェアの機能を使った分割の境界と一致する場合はハードウェアのブロードキャスト機能を使うことが出来る．その他の場合の PU 間通信で，PU 群の境界を越えた転送が起こらないようにする保護はソフトウェア的に行う．

12.1.2 ファイル関連機能

ファイルに関しては，通常の Unix 4.3 BSD と互換性のあるファイルシステムであること，並列機全体で一つのツリーをなすファイルシステムであること，を基本とする．さらに，PU が 3 次元の配列になっており，その中の一つの 2 次元平面をなす PU にディスクがついている (それを IOU と呼ぶ) システム構成を最大限に活用して，ファイル入出力の並列実行を実現するために，以下の機能を設けることにし，その詳細仕様を定めた．

1. 一つのファイルを複数の IOU に分散配置する機能

2. 各 PU から，それに最も近い (ハイパークロスバ ・ネットワーク上で)IOU に分散配置されている，あるいは分散配置しようとする，ファイルの位置を求める機能

12.2 プロセス間通信機能

プロセス間のメッセージ通信の性能は，並列計算機の性能を左右する重要な要素である．そこで，通常の， OS を介した通信機能の他に，OS をほとんど介しない特別に高速な通信機能を設計した．この通信は，送信者主導であり，送信者の PU のメモリから受信者の PU のメモリに直接コピーする方式である．そのためのハードウェアの設計に基づき，ソフトウェアとしての仕様，その機能の各プログラム言語での呼びだし方，その目的コード，などを検討して高速性を追求した．

12.3 プログラミング環境

プログラミング環境とは，プログラムの作成，コンパイル，デバッグ，実行，実行結果の表示，など，プログラムの開発を支援するためのソフトウェアシステムである．それは，どの計算機の上で動くかなどにより，以下のように分類される．

1. セルフ ・システムまたは実機システム: 目的とする並列計算機上のもの
2. クロス ・システム: ホスト計算機またはワークステーション上のソフトウェアであり，実機上で動くプログラムを生成する．
3. シミュレーション ・システム: ホスト計算機またはワークステーション上で実機での動きをシミュレートするシステム

超並列計算機用のプログラムを開発するためには，これらはいずれも必要なものである．以下の各項目の中で，それらをより具体的に述べる．

12.4 プログラム言語

プログラム言語としては，アセンブラ言語，C，Fortran，HPF(High Performance Fortran) を備えることとした．超並列計算機用のプログラムを書くのは容易ではない．メッセージ通信などを意識せずにプログラムを書く一般ユーザは，通常の Fortran プログラムにデータの分散配置を指定するだけで並列プログラムが出来る HPF のような言語が必要である．エキスパートには，C や Fortran で並列を意識したプログラムを書き，それに並列に実行されるプロセス間でのメッセージ通信の命令 (手続き呼びだし) を入れられるようにする必要がある．いずれの場合も，並列計算機の性能を最大限に引きだすには，コンパイラによる徹底した最適化が必要であり，各種の最適化の方式を検討した．しかし，コンパイラによる最適化にも限度があるので，アセンブラ言語でのプログラミングも可能とし，そのプログラムのデバッグシステムも考慮する必要がある．以上が，これらの言語を備えることにした理由である．なお，これらの言語については，ワークステーション上のクロスコンパイラと実機上のセルフコンパイラの両方を開発する予定で設計を進めている．

12.4.1 最適化コンパイラ

並列計算機の性能を最大限に発揮させるためにはコンパイラによる最適化がその鍵になる．並列プログラムの最適化として PU 間通信の最適化が重要であることは言うまでもないが、高い数値演算性能を実現するためには個々の PU 上で動くプログラム (オブジェクト ・コード) 自体が高度に最適化されていることも重要である。本研究で開発予定のプロセッサはスライド ・ウインドウ機能によって疑似ベクトル処理を行うスーパースカラ型プロセッサであり、きわめて高い演算能力を有する。しかし、その能力を十二分に引き出すには高度に最適化されたオブジェクト ・コードを作成せねばならない。ところが、レジスタ ・ウインドウのスライドによって論理レジスタ番号が順次変化していく点が人手によるコード作成を困難にする。したがって、最適化のアルゴリズムも当然複雑になるし、作成されたコードの検証も容易ではない。

このコード最適化の研究の端緒として、最適なコード作成を支援するツールを試作した。これを便宜上 簡易コンパイラ」と呼んでいる。詳細を節 12.6に記述する。

12.5 プログラム開発 ・デバッグ機能

プログラム開発用のツールとしては，エディタ，コンパイラ，デバッガ，性能モニタなどがある．ここでは，並列計算機で重要なものとなるデバッガと性能モニタ，および実機を使わずにそれらを可能とするシミュレータについて検討してきたものを述べる．

12.5.1 リモートデバッガ

ワークステーションなどをホストとして，並列計算機 (ターゲットマシン) のプログラムをデバッグするためのものであり，ホストマシン上から，並列計算機上のプログラムの実行制御をしたり，メモリやレジスタの内容の参照や更新が出来る．

12.5.2 トレーサ ・モニタ

並列計算機上のプログラムの実行の履歴 (トレース) をとるものであり，その結果をワークステーションなどで解析して性能の解析などをするためのものである．実行の履歴としては，各機械語命令のトレース，システム ・コールなどのイベントのトレース，PU 間通信のトレース，などがある．

12.5.3 シミュレータ

並列計算機用のプログラムを，ソースプログラムの変更なしに，ワークステーション上で走らせるシステムである．そのためにはスライドウィンドウの機能と本並列計算機特有の通信機能をシミュレートする機能が必要であり，その設計をした．

12.6 簡易コンパイラ

C 言語で書かれたループ ・プログラム片 (ループ本体は代入文の並びとする) をスライド ・ウインドウ機能を利用したオブジェクト ・コードへ変換するものである。たとえば以下のプログラム片

```
for(int i = START_INDEX; i < END_INDEX; i++){
```

```
        tmp = array3[i] * (tmp + array4[i - 1]);
        array0[i] = tmp;
        array1[i] = array4[i];
    }
```

は次のような疑似オブジェクト・コードへ変換される。

```
0 : Memory(array1[i]) := R(30)   | R(29) := R(28) + R(30)
1 : Memory(array0[i]) := R(28)
2 : R(125) := Memory(array3[i])  | R(30) := R(31) * R(29)
3 : R(124) := Memory(array4[i])
4 : Slide+ = -2
5 : CondBranch
```

(煩雑であるから上のコードの詳細は述べない)。以下はこの簡易コンパイラの概要である。

簡易コンパイラにおける最適化技法 (コード作成の技法) はいわゆるソフトウェア・パイプライニング (software pipielining)、あるいはそれをアルゴリズム化したモジュロ・スケジューリング (modulo scheduling) を基本としている。ソフトウェア・パイプライニングとは、ループ実行の命令列を複数のステージ (あるいはフェーズとも呼ぶ) に分割し、それらステージをパイプライン的に同時実行する方法であり、これによって命令の latency を隠し、プロセッサ・リソースを効率的に利用できる。複数のステージの同時実行をソフトウェア的に実現するためには各命令の実行のタイミングおよび使用するレジスタの番号に関してある種の制約問題を解く必要があるが、この解法は次の二つに大別される。ひとつは、まず各命令の実行タイミングを決定し (これを命令スケジューリングと呼ぶ)、その後にレジスタ番号を決定する (これをレジスタ割付と呼ぶ) 方法 (pre-pass scheduling) である。もうひとつは逆に、まずレジスタ割付を行い、次に命令をスケジューリングする方法 (post-pass scheduling) である。スライド・ウインドウを利用するときには論理レジスタ番号が動的に変化するため、後者の方法を適用することは難しいと考えられる。試作した簡易コンパイラは前者の方法に基づいている。前者をさらに詳細に述べると、これは以下のような手続きからなる。

1. データ依存グラフ (data dependence graph) の作成
2. 命令の優先順位の決定
3. ループ立ち上げ間隔 (loop initiation interval) の決定
4. 空なリソース予約テーブル (resource reservation tabel) の作成
5. 優先順位に基づくリソース予約テーブルへの命令の埋め込み
6. レジスタ干渉グラフ (register interference graph) の作成
7. レジスタ干渉グラフの彩色

疑似ベクトル処理は、従来のソフトウェア・パイプライニングと以下の点で異なる。

まず、繰り返し処理の度に論理レジスタ番号が一定の刻み幅で変化していくから、複数のパイプライン・ステージで使用するレジスタ番号の間の干渉が容易に回避できるという利点がある。いわゆる register renaming が実現されているのである。通常のソフトウェア・パイプライニン

グでは、このレジスタ間の干渉のため、ループ立ち上げ間隔はそのループ処理に含まれる命令の中の最大の latency 以下にはできない。ほとんどの場合そのような命令は load 命令であり、その latency は相当に大きな値であるため、問題は深刻である。その対策として一般には loop unrolling を行い、かつ命令スケジューリングをより巧妙にする。疑似ベクトル処理ではそのような技巧を凝らす必要がないため、ループ立ち上げ間隔の算出や命令スケジューリングのアルゴリズムがごく自然な発想に基づいて素直に実現できる。このことは最適化の研究を見通しのよいものにすると思われる。逆に論理レジスタ番号が変化することの欠点を挙げれば、ひとつの物理レジスタに複数の論理レジスタ番号を与えることとなるから、見かけ上使用するレジスタ数が増えてしまう。しかしながら時間効率が空間効率と引き替えられたものと考えれば、必然の理とも言える。むしろ問題は、使用するレジスタ数がコード生成の最終段階、つまりレジスタ割り付けが終了するまで把握できない点にある。これはコード最適化の戦略を不確実なものとする。またレジスタ干渉グラフは従来のものとは異なり、レジスタ番号が変化していくという情報を付加したものとなる。そのためレジスタ彩色が従来のものよりも複雑になる。簡易コンパイラでは、レジスタ彩色の近似アルゴリズムを一部拡張したアルゴリズムで彩色を行っている。拡張にもいくつかの案があり、現在さらに検討中である。

疑似ベクトル処理では大きな memory access latency を隠すための preload も重要である。データがあるレジスタ (ほとんどの場合、ウインドウ外のレジスタ) に preload され、そのレジスタがスライドされウインドウ内へ入って演算命令で使用されるまでの時間間隔 (これを preload 命令の permitted latency と呼ぶ) は大きければ大きいほど、バンクコンフリクト等による効率低下は小さい。簡易コンパイラでは、まず preload 命令の latency を適当な値 (現在は 30machine cycle) と仮定して命令スケジューリングを行い、一旦レジスタ割付までを終える。そしてその後に、レジスタの使用状況を基に permitted latency を可能な限り大きくするように、preload 命令のターゲット ・レジスタを計算し直す。

試作した簡易コンパイラは、属性文法に基づくコンパイラ生成系 Rie のソース ・プログラム約 400 行および C++のソース ・プログラム約 9000 行で記述した。最適化アルゴリズムとプロセッサ ・アーキテクチャの記述を独立させるため、プロセッサ ・アーキテクチャは C++のクラスとして実現している。これにより種々のアーキテクチャによって実行時間がどのように異なるか、使用されるレジスタ数はどうか等の実験が容易にできる。現在は、以下のようなアーキテクチャを記述し、その違いを検討中である。

1. 命令並列度が 1 の最も単純なアーキテクチャ
2. 本プロジェクトで設計中のアーキテクチャ
3. 上のものにさらに load/store 系ユニットを加えたアーキテクチャ

この簡易コンパイラ自身のバグを早期になくすことを目的として、テスト ・プログラムを自動生成し、テスト ・プログラムから生成されたオブジェクト ・コードを自動検証するツールも開発した。これによって約 10 万個のテスト ・プログラムを自動生成し、 1 ヶ月間で 40 数個のバグを発見することができた。単純なバグはほとんど駆逐できたものと考えている。なお、このテスト ・プログラム自動生成ツールは、それ自体が研究対象として興味深いだけでなく、ベンチマーク ・プログラム生成ツールとしても今後の実験に利用していく予定である。

研究の現状はおおむね以上の通りである。今後は次の二つを大きな目標としている。ひとつは、先出の簡易コンパイラを基にループ本体が代入文の並びであるような場合の最適化アルゴリズムを完成させることである。そのためにはレジスタが足りなくなった場合の対処法などまだ解

決すべき問題は数多い。もうひとつは、ループ本体が条件分岐を含む場合の最適化アルゴリズムの研究である。多くの基本関数の計算には条件分岐が現れる。ループ・プログラム中で基本関数を使おうとするとき、条件分岐のあるプログラムのソフトウェア・パイプライン化は避けられない。現在の研究動向ではいくつかの基本的なアイディアが提案されている段階である。それらをほぼそのまま疑似ベクトル処理へ応用できることは既に分かっている。スライド・レジスタの特徴を利用した改良案を検討中である。

13 計算物理学の典型的問題に対する仮想ベンチマークの結果

この節では、計算物理学の典型的問題において、CP-PACS でどの程度の実効性能が達成されるかを調べる為に行なった、仮想ベンチマーク テストの結果を述べる。

13.1 ブロック・ストライド転送

並列計算機全体の高い実効性能を保つ為には、データ転送が PU 性能に比例して十分高速に行なわれる必要がある。一回の転送にかかる転送時間は、転送オーバーヘッドと、転送量を最高転送速度で割った時間との和で与えられる。ここで転送オーバーヘッドはネットワーク経由で相手の PU との接続を確保する為にソフトとハードから要求される時間で、高速転送モードでも数 μ sec 程度必要である。200 MByte/sec の転送速度の場合、これは数百 Byte 分の転送時間に相当する。小量の転送では、転送時間のほとんどが転送オーバーヘッドで決定されてしまうが、転送オーバーヘッドを大幅に削減することは技術的に困難である。従って、転送を高速化する為には、転送回数を減少させ、1回当たりの転送量を増大させる必要がある。

最も単純な転送機能は、メモリ上で連続、または一定間隔で並んだデータを、相手 PU のメモリにコピーするというものだが、転送データがメモリ上にどのように分布しているかは解こうとしている問題の性質に依存しており、多くの場合このパターンには当てはまらない。転送データの全ての分布パターンに対応することは、転送プロトコールの肥大化につながりかえってオーバーヘッドを大きくしてしまうが、幸い、QCD 計算をはじめとして多くの応用においてしばしば現れる転送パターンは、一定の長さの連続データが一定の間隔をおいて他数つながるという、規則的な形をしている。

そこで、CP-PACS では、この転送パターンに対応した転送機能を用意し、これを “ブロック・ストライド転送” と命名した。また、同じ形の転送命令が繰り返される時、それをより少ないペナルティーで連続するコマンド・チェイン機能も用意されている。

以下では、QCD の典型的な転送におけるネットワーク性能を評価し、ブロック・ストライド転送の効果を調べる。転送データ量と転送命令数を特徴づけるパラメータを B : 1命令の転送量, N_{chain} : チェインされる転送命令の個数とおくと、1回の転送における全転送量は $B_{total} = BN_{chain}$ で与えられ、転送時間 T と転送速度 R は、それぞれ、

$$T = a_s + a_h + B/\nu + (N_{chain} - 1)\{a_c + a_h + B/\nu\} + a_t,$$

$$R = B_{total}/T$$

とあらわされる。ここで、ν : 最大転送速度, a_s : software overhead, a_c : command chain overhead, a_h : hardware overhead, a_t : traveling time である。以下の評価では、$\nu = 200$ MByte/sec, $a_s = 1.5\mu$sec, $a_c = 0.5\mu$sec, $a_h = 0.5\mu$sec を仮定し、a_t は実際は通信相手までのネットワーク上の距離に依存するが $a_t = 0.5\mu$sec と仮定した。

図 17: QCD 計算の典型的な転送パターン

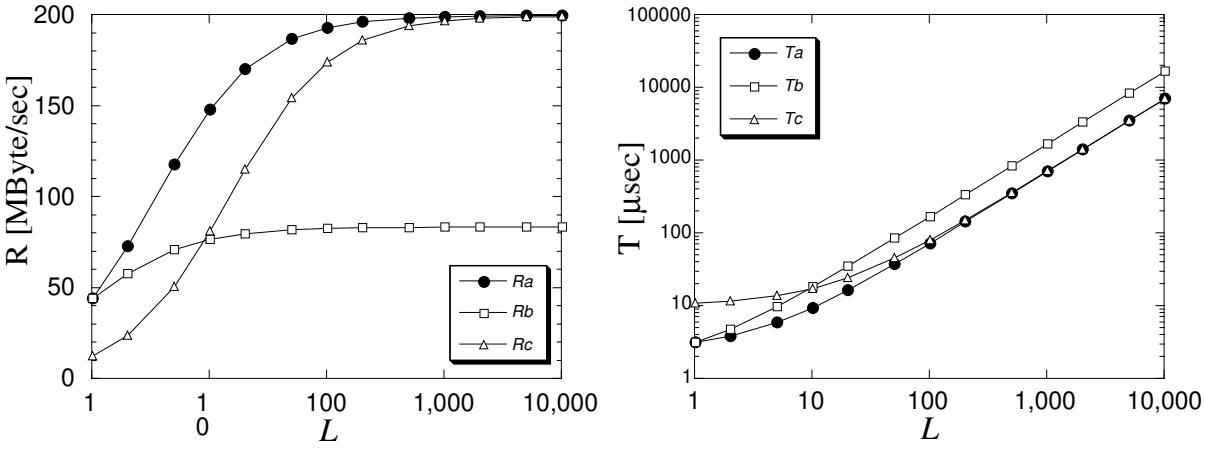

図 18: QCD の典型的な転送における転送速度と転送時間

7.1.1に概説したように、格子 QCD の基本変数は格子の各最小線分上に定義された 3×3 複素行列 U 及び各格子点に定義された 12 次元複素ベクトル q である。従って、QCD 計算では、図17に示したように、倍精度複素数が 9 個 (または 1 2 個) 連続したデータを一組とし、それがある一定の間隔 (stride) で数個から数百個並んだデータを転送することが多い。 (配列の宣言を工夫することにより、重い計算の転送パターンをこの形だけにすることが出来る。) ここでは、図 17 のパターンが L 個続いたデータを

(a) そのまま 1 回のブロック・ストライド転送で送る

(b) 8 Byte $\times$ 18 の連続データの転送を、コマンド・チェインで L 回連続

(c) 16 Byte のデータが一定間隔 (図 17の stride + 8 Byte $\times$ 16) で L 個並んだデータの転送を、コマンド・チェインで 9 回連続

の 3 つの場合について、転送性能を評価する。(b) と (c) はブロック・ストライド転送がサポートされていない場合の、それぞれ L が小さい時と大きい時に最も効率的な転送方法で、(c) では、16 Byte (4 倍精度) までのデータなら一定間隔で並んだものを 1 回で転送する機能が存在すると仮定している。転送データのパラメータはそれぞれ、

(a) $N_{chain} = 1,\ B = B_{total} = 8 \times 18 \times L$

(b) $N_{chain} = L,\ B = 8 \times 18$

(c) $N_{chain} = 9,\ B = 8 \times 2 \times L$

となる。QCD の場合、問題のサイズを固定しても L は広い範囲の値を変化し、例えば、格子サイズが 64^4 で PU 構成が $8 \times 8 \times 16$ の場合には full QCD の計算で $L = 4$ — 512 である。

評価の結果を図 18に示す。$L > 10$ では (b) より (c) の方が有利だが、ブロック ・ストライド転送 (a) と比較すると $L \leq 512$ で 5 — 80 % 遅い。$L < 10$ では (c) より (b) の方が有利だが、(a) よりも 20 — 100 % 遅い。結論として、QCD 計算に最も多く現れる数十から数百程度の L の場合に通信性能の低下を防ぐ為に、ブロック ・ストライド転送機能が有効であることが確認された。

13.2 仮想ベンチマーク QCD MULT の評価結果

QCD 計算において最も演算量が多く計算時間がかかる部分は、(行列) × (ベクトル) という構造をしている。即ち、

$$G(n:a) = \sum_{m=1}^{V} \sum_{b=1}^{12} D(n,m:a,b) G(m:b) \tag{3}$$

である。ここで L_i を i 方向の格子サイズとして $V = L_x \cdot L_y \cdot L_z \cdot L_t$ は格子点数であり、a, b はカラーとスピノルの自由度である。QCD の場合、行列 D は 、$n = m$ 及び n と m は隣同士、の二つの場合以外に値を持たない疎行列である。我々は上の演算を QCD MULT (または、単に MULT) と呼び、これを仮想ベンチマークとしてノードプロセッサやネットワークの性能評価を行なった。以下、CP-PACS の PU 配列として、$8 \times 16 \times 16$ を仮定し、格子サイズは、64^4を代表例に取って、その結果を報告する。

13.2.1 スライドウィンドウを用いた場合の QCD MULT のノードプロセッサあたり実効速度効率

CP-PACS のノードプロセッサの特徴のひとつはスライドウィンドウ機能である。この機能と preload, poststore 命令を組み合わせることでスカラープロセッサであるにもかかわらずあたかもベクトルプロセッサであるかのように動作させること (疑似ベクトルプロセッサ機能) が可能になる。この機能をフルに活用して、 単体のノードプロセッサ上での QCD MULT のアセンブラ ・コードを、日立製作所の協力を得て作成した。ノードプロセッサ単体あたりの実効速度効率に対する結果は以下の通りであり、スライドウィンドウ機能を使うことで 70 % 以上の高い実効速度効率を達成するアセンブラ ・コードの作成が可能となった。

(1) 浮動小数点演算数

$$N_{flop} = 1344 \cdot l_x l_y l_z L_t \tag{4}$$

但し、物理的格子 $L_x \cdot L_y \cdot L_z \cdot L_t$の空間三次元を CP-PACS を構成する PU の三次元配列により分割する。i 方向の PU 台数を n_i とすると、$L_i = n_i \cdot l_i$ $(i = x, y, z)$ という関係がある。

(2) 演算に必要なマシンサイクル数

$$MC_{flop} = (900 L_t + 219) l_x l_y l_z + 17 l_y l_z + 17 l_z + 35 \tag{5}$$

ただし、preload 命令を発行してからメモリ上のデータがノードプロセッサに届くまでに必要なマシンサイクル数は７４以下であると仮定した。

(3) 実効速度効率

ノードプロセッサは 1 マシンサイクルあたり 2 演算可能なスーパースカラーなので、ピーク演算速度と比較したときの実効速度効率は

$$\text{実効速度効率} = \frac{N_{flop}}{2 \cdot MC_{flop}} \tag{6}$$

で与えられる。上記 (1) 及び (2) によれば、有限格子サイズの影響を無視した場合の効率は $1344/(2 \times 900) = 74.7\%$、又 64^4 格子に即して効率を計算すると 74.4 % ($L_t = 64, l_x = 8, l_y = 4, l_z = 4$ の場合)、74.3 % ($L_t = 64, l_x = l_y = l_z = 1$ の場合) となる。

13.2.2 QCD MULT の効率に対するメモリのバンクコンフリクトの影響

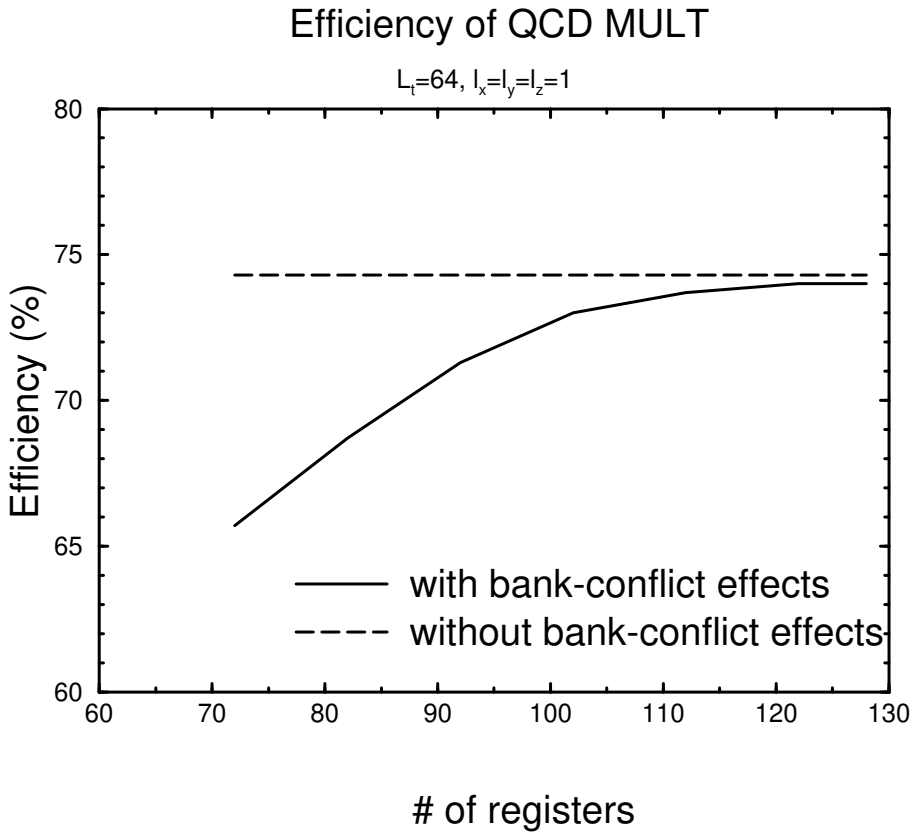

図 19: QCD MULT の効率に対するメモリのバンクコンフリクトの影響とレジスタ数との関係

上の QCD MULT の効率評価では preload 命令を発行してからデータがノードプロセッサに届くのに必要なマシンサイクル数を 7 4 以下と仮定した。この条件はメモリの連続アドレスからの preload では容易に満たされるが、QCD MULT のように非連続アドレスからの preload の場合、バンクコンフリクトの影響 (同じバンクにアクセスしようとすると 2 回目は遅くなること) で、仮定が満たされない場合がある。バンクコンフリクトの影響を採り入れた効率を評価するために、日立製作所作成のメモリの実際の動作を反映するシミュレーターを筑波大学のワークステーション上で動かし、QCD MULT の効率に対するバンクコンフリクトの影響を評価した。シミュ

レーター作成時に想定したメモリの動作と、現時点で使用する予定のメモリの動作は若干異なるので、シミュレーターの結果は、バンクコンフリクトの影響を評価するだいたいの目安として考える。

図 19 に、$L_t = 64,\ l_x = l_y = l_z = 1$ の場合のシミュレーションの結果を示す。表 12 に、数値を掲げる。点線はバンクコンフリクトの影響のない場合の効率 74.4 % を示し、実線はバンクコンフリクトの影響を採り入れた効率である。横軸はスライドウインドに使用できるレジスタ数である。レジスタ数が多ければ、より" 遠く" に preload できるのでバンクコンフリクトの影響を低減することが可能になる。図 19 の結果からレジスタ数が 110 以上であればバンクコンフリクトの影響による性能低下を 1%以下に押えられることがわかった。

レジスタ数	72	82	92	102	112	122	∞
効率 (%)	65.7	68.7	71.3	73.0	73.7	74.0	74.3

表 12: QCD MULT の効率に対するメモリのバンクコンフリクトの影響とレジスタ数との関係

13.2.3 QCD MULT の効率に対する通信の影響

前述の効率は単体のノードプロセッサの性能であり、並列計算機の性能を評価するためにはノードプロセッサ間のデータ転送のための通信時間を考慮する必要がある。QCD MULT の場合にデータ転送に必要な通信時間を計算し並列計算機全体での QCD MULT の実行性能効率を評価した。結果は以下の通りであり、CP-PACS での量子色力学シミュレーションの重要目標の一つである 64^4格子に対して、 QCD MULT は並列計算機全体で 60 % に近い性能を達成可能と考えられる。

(1) 転送に必要なマシンサイクル数

$$MC_{comm} = 2(192(L_t + 2) + \Delta T)(l_y l_z + l_x l_z + l_x l_y) \tag{7}$$

ここで、ΔT は通信を起動するためのオーバヘッドであり、暫定的に$\Delta T = 1000$ マシンサイクルを仮定する。また、通信ネートワークの転送のスループットは 1 Byte/マシンサイクル とする。

(2) 転送時間と計算時間の比

この比は MC_{comm}/MC_{flop}で与えられる。$L_t = 64,\ l_x = 8, l_y = 4, l_z = 4$ の場合の値は、30.0%である。

(3) 全体の効率

転送時間を含めた全体の実効速度効率は

$$\text{実効速度効率} = \frac{N_{flop}}{2 \cdot (MC_{flop} + MC_{comm})} \tag{8}$$

で与えられる。いろいろな l_i の場合の通信時間を考慮した効率を表 13 に載せる。特に、64^4 格子に対する $L_t = 64,\ l_x = 8, l_y = 4, l_z = 4$ の場合、全体効率は 57.4% である。格子サイ

ズが小さい場合は通信時間の比率が大きくなり効率はかなり低下する。従ってこの場合には PU を分割利用することにより通信時間を低減することが有効である。

L_t	l_x	l_y	l_z	実効速度効率
80	80/8	80/16	80/16	60.4 %
64	64/8	64/16	64/16	57.4 %
48	48/8	48/16	48/16	52.8 %
16	64/8	64/16	64/16	53.2 %
12	48/8	48/16	48/16	47.0 %
8	32/8	32/16	32/16	36.5 %

表 13: 通信時間を含めた QCD MULT の効率の格子サイズ依存性。上半分はハドロン質量計算に、下半分は有限温度シミュレーションに用いる格子サイズの典型例である。

13.3 格子量子色力学シミュレーションに要する計算時間の評価

13.3.1 quenched QCD ハドロン質量計算

格子量子色力学のシュミレーションは 1) ゲージ配位の生成、2) クォーク伝搬関数の計算、3) ハドロン物理量の計算の 3 ステップからなる。クォーク伝搬関数の計算には red/black 最小残差法を用いる。その計算ルーチン QCDMR は、最も計算時間を必要とするサブルーチン MULT に加え、内積等の演算 ・データ転送 ・DISK I/O を含む。各々の実行時間を 13.2節の評価結果を基礎に評価し、積み上げ式に QCDMR の全実行時間を評価した。その他のステップについては 3 ステップ相互の計算時間比の経験値を用いて、quenched QCD によるハドロン質量計算に必要な全計算時間の評価を行なった。格子サイズは 64^4を仮定し、各ステップで必要な格子パラメタの値は 7.1節　素粒子物理学に於ける格子量子色力学」での値を踏襲した。

性能評価の仮定は、以下の通り。

PU 構成:　$8 \times 16 \times 16 = 2048$ PU $+ 8 \times 16 = 128$ IOU
PU 性能:　クロック 150 MHz　2 命令スーパースカラー + PVP-SW
ネットワーク 性能: スループット =150 MB/sec　レイテンシ = 7 micro sec
ディスク性能:　RAID-5　SCSI2　ブロックサイズ = 32KB
クォーク質量:　21 MeV 以上の 6 点 (m_π/m_ρ =0.9,0.8,0.7,0.6,0.5,0.4)
問題規模:　64^4 格子　3000 スイープ 毎に 500 ゲージ 配位
アルゴリズム:　Red/Black 最小残差法

表 14(a) に QCDMR 繰り返し一回に要する実行時間を示す。表 14(b) に、一配位当り 6 種類のクォーク質量に対するクォーク伝播関数を QCDMR により求めるのに要する実行時間を示す。表 14(c) に、総計算時間とその内訳を示す。

(a) 格子 QCDMR 繰り返し一回の実行時間。

	# flop/PU	演算時間 (sec)	通信時間 (sec)	実行時間 (sec)
MULT	11010048	0.0495	0.0	0.0495
QCDMR	3145728	0.0123	0.0133	0.0256
合計	14155776	0.0618	0.0133	0.0752
実行時間比		82.3 %	17.7 %	100 %

(b)quenched QCD 一配位当たりのクォーク伝播関数計算時間。

	演算	通信	DISK I/O	合計
時間 (hour)	0.882	0.190	0.373	1.445
比	61.0 %	13.2 %	25.8 %	100 %

(c) quenched QCD ハドロン質量計算必要日数。

	ゲージ配位生成	クォーク伝搬関数	ハドロン物理量	合計
時間 (days)	22.2	30.1	5.7	58.0
比	38.3 %	51.9 %	9.8 %	100 %

表 14: 64^4格子 quenched QCD ハドロン質量計算に要する総計算時間及び内訳。 (flop=浮動小数点演算)

64^4格子に対するシュミレーションの総計算時間は 58 CPU 日 であり、これを 3 つの格子間隔に対して繰り返すとすれば、クエンチ近似のハドロン質量計算の総計算時間は 174 CPU 日 である。

13.3.2 full QCD ハドロン質量計算

full QCD のシュミレーションも、quenched QCD 同様 1) ゲージ配位の生成、2) クォーク伝搬関数の計算、3) ハドロン物理量の計算の 3 ステップからなる。full QCD の場合、計算の大部分は ゲージ配位の生成に必要な正規乱数ソースに対するクォーク伝播関数の計算に費やされる。従って、ここでは配位生成時間に主要部分を占める QCDMR の実行時間の評価により、配位生成時間の評価を行なった。

表 15 に $32^3 \times 64$ 格子及び $48^3 \times 64$ 格子の場合の場合の QCDMR 繰り返し一回あたりの実行時間の内訳を示す。

表 16 に、ゲージ配位生成時間を、代表的クォーク質量の各値に対して示す。

$(a) 32^3 \times 64$ 格子

	# flop/PU	演算時間 (sec)	通信時間 (sec)	実行時間 (sec)
MULT	1376256	0.00619	0.0	0.00619
QCDMR	393216	0.00154	0.00333	0.00487
合計	1769472	0.00773	0.00333	0.01106
実行時間比		69.9 %	30.1 %	100 %

$(b) 48^3 \times 64$ 格子

	# flop/PU	演算時間 (sec)	通信時間 (sec)	実行時間 (sec)
MULT	4644864	0.02089	0.0	0.02089
QCDMR	1327104	0.00520	0.00750	0.01270
合計	5971968	0.02609	0.00750	0.03359
実行時間比		77.7 %	22.3 %	100 %

表 15: full QCD の場合の QCDMR 繰り返し一回あたりの実行時間

13.3.3 有限温度 QCD のシュミレーション

有限温度 QCD のシュミレーションは、空間方向の格子サイズ L_s が 時間方向の格子サイズ L_t よりも大きい格子での full QCD 計算を必要とする。従って、計算時間のほとんどは、ハイブリッドモンンテカルロ法でのゲージ配位生成における MULT の実行によるので、ここでは、MULT の全実行時間を評価する。PU 単体は前節と同じくクロック 150 MHz とし、通信スループット 150 MByte/sec、通信オーバーヘッド 1000 マシンサイクルを仮定し、PU 構成は $8 \times 16 \times 16$ とする。

格子サイズ		m_π/m_ρ			
L_t	L_s	0.7	0.6	0.5	0.4
64	48	210	367	618	1067
64	32	46	81	136	234

表 16: full QCD シュミレーション の ゲージ配位生成時間 (日)。但し、空間的に等方的な格子をとり、$L_x = L_y = L_z = L_s$とする。分子動力学ステップサイズは$\delta\tau = 0.02(8/L_s)$ を仮定し、$\tau = 5$ 毎に 500 ゲージ 配位 を生成する場合の評価。

格子サイズを $L_x \times L_y \times L_z \times L_t$ とし、各 PU 内の格子サイズを $l_x \times l_y \times l_z \times L_t$ ($l_x = L_x/8$, $l_y = L_y/16$, $l_z = L_z/16$) とすると、１回の MULT の演算と通信に要求される計算ステップ数は、それぞれ (5), (7) で与えられる。ハイブリッドモンテカルロ法では、配位を１分子動力学時間ステップ$\delta\tau$ だけ進める毎に逆行列計算が２回要求される。１回の逆行列計算に呼び出される MULT の回数 N_{iter} は表 4 を仮定し、ステップ幅は $\delta\tau = 0.02 \times (8/L)$ を仮定した。

表 17 に $\tau = 2000$ だけシミュレーションするのに必要な計算日数をまとめる。配位の蓄積に要求される Disk I/O 時間などを無視すれば、これが、ひとつのデータを計算するのに必要な時間である。実際のシミュレーションでは、各格子サイズについて、結合定数のパラメーター空間の十点程度でこの計算を繰り返す必要がある。

格子サイズ		m_π/m_ρ			
L_t	L_s	0.7	0.5	0.3	0.18
16	64	81.0	238	781	2279
12	48	21.7	64.0	210	612
8	32	3.7	10.8	35.4	103

表 17: 有限温度 QCD シュミレーションで、$\tau = 2,000$ の計算に要する計算日数。空間的に等方的な格子を考え、$L_x = L_y = L_z \equiv L_s$ とする。

13.4 KS fermion の場合の実効性能評価

13.4.1 QCD MULT の効率

格子 QCD の数値シミュレーションでは、我々が考察してきた Wilson fermion と呼ばれているクォークの作用関数以外にも、KS fermion と呼ばれる別の形のクォークの作用関数も使われている。そこで、我々は KS fermion の場合の QCD MULT の効率も同様に考察した。ここに、その結果を簡単に紹介する。

(1) 浮動小数点演算数

$$N_{flop} = 624 l_x l_y l_z L_t \tag{9}$$

(2) 演算に必要なマシンサイクル数

$$MC_{flop} = 384 L_t l_x l_y l_z \tag{10}$$

ただし、計算の立ち上がりに必要な部分のマシンサイクル数は除いた。

(3) 効率

上記 (1) と (2) から 理論性能は 81.25 % である。

(4) データ転送に必要なマシンサイクル数

$$MC_{comm} = 2(48(L_t + 2) + \Delta T)(l_y l_z + l_x l_z + l_x l_y) \tag{11}$$

ここで、ΔT は前述の、通信の起動のためのオーバヘッドである。

(5) 全体の効率

$L_t = 64$, $l_x = 8$, $l_y = 4$, $l_z = 4$ の場合、

$$\text{全体の効率} = \frac{N_{flop}}{2(MC_{flop} + MC_{comm})} = 67.0\% \tag{12}$$

いろいろな l_i の場合の全体効率を表 18 に載せておく。

L_t	l_x	l_y	l_z	実効速度効率
80	80/8	80/16	80/16	70.0%
64	64/8	64/16	64/16	67.0%
48	48/8	48/16	48/16	62.1%
16	64/8	64/16	64/16	58.9 %
12	48/8	48/16	48/16	50.6 %
8	32/8	32/16	32/16	36.9 %

表 18: KS fermion の場合の QCD MULT の全体効率の格子サイズ依存性。上半分はハドロン質量計算に、下半分は有限温度シミュレーションに用いる格子サイズの典型例である。

13.4.2 KS fermion の場合の QCD シミュレーション計算時間評価

KS fermion を用いた場合の QCD シュミレーションに必要な計算時間の評価をまとめる。quenched QCD の場合に、クォーク伝播関数を計算するのに要する時間をクォーク質量 (格子単位) の関数として表 19 に示す。但し、ゲージ配位は既に生成済みとし、その為に必要な時間は含

格子サイズ		m_q			
L_t	L_s	0.02	0.01	0.005	0.0025
64	64	1.8	3.5	7.0	14.1
48	48	0.6	1.2	2.4	4.8

表 19: KS fermion の場合の quenched QCD ハドロン質量計算於けるクォーク伝播関数計算時間 (CPU 日)。空間的に等方的な格子を考え、$L_x = L_y = L_z \equiv L_s$ とする。ゲージ配位数は 500 、CG 法の収束回数は経験値 $1000(0.01/m_q)$ を仮定。

まない。クォーク伝播関数計算には共役傾斜法 (CG と略称 ;繰り返し一回当たりの MULT 呼びだし回数二回) を用い、通信オーバヘッドは$\delta T = 1000$ MC を仮定した。

表 20 に full QCD の場合について、配位生成に要する時間を示す。配位生成法はハイブリッドモンテカルロ法の変形であるハイブリッド R 法を用い、分子動力学ステップサイズは、経験値 $\delta\tau = 0.01(m_q/0.01)$ を仮定する。クォーク逆行列の計算は共役傾斜法により、収束の為の繰り返し回数は経験値 $500(0.01/m_q)$ を用いた。

格子サイズ		m_q			
L_t	L_s	0.02	0.01	0.005	0.0025
64	64	11	40	154	601
64	48	4.9	18	69	268
64	32	1.6	5.9	23	88
16	64	2.1	8.3	33	133
12	48	0.8	3.0	12.2	48.7
8	32	0.2	0.8	3.3	13

表 20: KS fermion の場合の full QCD 配位生成に要する計算時間 (CPU 日)。空間的に等方的な格子を考え、$L_x = L_y = L_z \equiv L_s$ とする。表の上半分はハドロン質量計算に典型的格子サイズであり、配位間隔 $\tau = 5$、配位数 500 を仮定。下半分は有限温度の場合であり、$\tau = 2000$ に対する評価値を示す。ステップサイズは $\delta\tau = 0.01(m_q/0.01)$、CG 法の収束回数は $500(0.01/m_q)$ を仮定する。

13.5 評価

以上の仮想ベンチマーク結果は、PU 台数 2048 に対して、7.1節で評価した必要計算能力がほぼ満足されることを示す。特に、quenched QCD については、より計算時間を要する Wilson fermion 法に対しても CPU 時間１８０日程度を用いることにより、系統誤差と統計誤差双方を充

分に押えた計算が、64^4格子規模に対して可能である。

full QCD については、KS fermion 作用を用いて場合には、PU 台数 2048 台により、64^4 規模の計算が CPU 時間１カ年程度で可能と考えられる。一方、Wilson fermion 作用の場合は、32^4 格子規模の計算が既に CPU 時間１ケ年程度以上を要する。但し、Wilson fermion 作用を用いた場合、ここでの評価に仮定した red/black 最小残差法の収束回数は 1/2 程度に緩めても結果に影響を及ぼさない可能性がある。また、ILU 法による前処理は収束回数を 1/4-1/5 に低減する効果があることが知られおり、これを効率的に並列プログラムできれば計算時間の一層の削減に継る。以上の諸点の検討により、Wilson fermion 作用を用いた full QCD に対して実行可能な格子規模の拡大を計ることは今後の重要課題である。

14　これまでの研究成果及び反省すべき点

平成４年度及び５年度の研究成果は節 7から節 13に述べたとおりである。特に強調すべき点を簡潔にまとめる。

(1) 超並列計算機 CP-PACS 主要目標性能の決定

本計画に於いて中心となる計算物理学の諸問題の検討、中でも素粒子物理学の格子量子色力学の数値的研究を飛躍的に推進するために必要となる計算能力の詳細な評価を基礎に、これを実現する並列計算機の設計・製作上の技術的検討を行ない、以上に基づいて超並列計算機 CP-PACS の基本アーキテクチャ及び主要目標性能を定めた。特に、理論ピークスピードを６４ビットデータに対して、600 GFLOPS 以上とした。

(2) スライド・ウィンドウ方式によるノードプロセッサの機能強化。

大型科学計算に於ては、汎用 RISC チップのキャッシュメモリが有効に機能せず、最高性能の 1/10 程度の能力しか発揮出来ない。この点の機能強化のために、スライド・ウィンドウ方式に基づく疑似ベクトルプロセッサアーキテクチャを考案し、その有効性を詳細な検討により確認すると共に、論理仕様の纏め及び具体的な論理設計の大部分を行った。

(3) ネットワークとして３次元ハイパークロスバーの採用。

データ通信ネットワークは並列計算機の能力の鍵の一つであるので、転送能力及び柔軟性等の基本性能、価格、実装面から、種々のネットワーク構成を比較検討することにより、３次元ハイパークロスバーが最適との結論に達し、その仕様の詳細を纏め具体的な設計に着手した。

(4) スライド・ウィンドウ方式プロセッサの為のコンパイラー開発の基本方針の決定。

スライド・ウィンドウ機能を有効利用するプログラムの生成のためのコンパイラー・アルゴリズムについて、モジュロ・スケジューリング方式に基づく方式を考案し、その有効性を確認した。さらにコンパイラーの雛形を得ることができた。

(5) 物理学の応用例に対する実効速度の評価。

本計画の中心課題の一つである格子量子色力学の計算において最も重要な部分に対して、CP-PACS の基本設計に忠実に則った仮想ベンチマークを行ない、上記 (2) の機能強化が極めて有効であることを確認した。さらに、個々のノードプロセッサでの計算に必要な時間に

加えて、プロセッサ間通信に要する時間を合わせた詳細な検討を行ない、その結果、格子量子色力学計算に於いては、実効速度として最高速度の 50 – 60%程度を達成可能との結論を得た。

(6) 最高速度 600 GFLOPS は技術的に可能。

上記の技術的諸点に加え、デバッグ、メインテナンス、実装など実際の製作・運用において重要となる諸々の点をも詳細に考察し、その結果最高速度 600 GFLOPS の並列計算機は現状の汎用大型機の上位機種と同程度の大きさと消費電力で技術的に製作可能であるとの結論に達し、その実現を目指して詳細設計を行なってきた。

以上の成果は、計算物理学研究センターを研究推進母体として、本計画参加者間の密接な研究連絡体制を作り、そこでの議論の結果を、研究員会議及びワーキンググループを通じて集約することにより得られた。この研究推進方法は、元来異なった研究組織に属する物理学者と並列計算機工学者が有機的に一体となって研究を進める上で極めて有効であり、特に問題とすべき点はなかったと考えるが、今後もこの研究推進方法をさらに充実させることにより、超並列計算機 CP-PACS の完成に向け全力を集中することが重要であると考えている。

15 当初計画からの変更点

当初計画からの変更点は、理論最高計算速度性能を 300GFLOPS 以上から 600GFLOPS 以上に引き上げた点にある。

計画段階当初において、米国では理論最高計算速度 1 TFLOPS の計算機を製作するプロジェクトが提案されていたので、本計画としてもそれになるべく近い性能を持つ計算機を製作したい希望は持っていたが、技術的に可能であるかどうかの点が当時明らかでなかったこと、また、予算面の条件も考慮し、目標値 300 GFLOPS 以上で開発・製作をスタートした。しかし、平成４年度及び５年度の研究の結果、600 GFLOPS の計算機は技術的に可能であることが明らかになった。

本プロジェクトで取り組む予定の物理学の重要諸問題、中でも、格子量子色力学に基づく素粒子の強い相互作用の解明、宇宙物理学に於ける銀河・星・ブラックホール生成過程の解明などのためには 300 GFLOPS では計算能力が不十分であること、さらに米国・欧州に於ても１９９６年頃には 500 GFLOPS 程度の能力を持つ並列計算機が製作され、物理学研究に用いられる可能性があること、以上を考慮に加え、600 GFLOPS の並列計算機を開発・製作することに計画を変更した。

Part III
今後の研究計画

16 平成６年度以降の研究計画

16.1 平成６年度

平成４年度及び５年度の研究成果である超並列計算機 CP-PACS の基本設計及び詳細設計に基づき、実装設計を加味して CP-PACS の詳細アーキテクチャを確定する。これを基に、（1）

ノードプロセッサ及び主記憶系、(2)並列ネットワーク部、(3)システム制御・入出力制御部、各々の論理設計及び実装設計を行なう。さらに、CP-PACS を構成する、(1)ノードプロセッサ用 VLSI、(2)結合ネットワーク用 VLSI、(3)ネットワークボード、(4)入出力制御系ボード、(5)メモリ系サブユニット部、各々の部品の試作を行なう。

ソフトウエアに関しては、前年度までの成果を基に、OS、ネットワーク通信用ライブラリ、コンパイラの設計・開発を行なう。また、実機の性能評価用のシミュレータの設計・開発を行なう。

これと並行して、素粒子物理学、物性物理学、宇宙物理学の研究課題の計算プログラム開発を行い、これらをプロトタイプ上などで実行して、プロトタイプのテストと同時に、プログラム自身の検討、計算アルゴリズムの改良に使用する。

16.2 平成 7 年度

演算ユニットボード、ネットワーク・ボードの製作を開始し、少数のプロセッサから構成されるサブシステムを製作する。このサブシステムに対して、ハードウエア、ソフトウエア両面からの入念な機能・動作テストを行ない、不都合な点が発見された場合は改良、修正を行う。これらの機能・動作テストの後、ボードの量産を開始し、CP-PACS の組立を進める。平成 7 年度末までに、超並列計算機 CP-PACS の組立を完了し、計算物理学研究センターに設置する。

ソフトウエアに関しては、前年度に開発した OS、ネットワーク通信用ライブラリ、コンパイラのデバッグを行ない、必要な場合は改良作業を行う。

素粒子物理学、物性物理学、宇宙物理学の研究課題の計算プログラム開発をさらに行い、シミュレータを用いて実行性能を評価し、プログラム自身の検討、計算アルゴリズムの改良を行う。

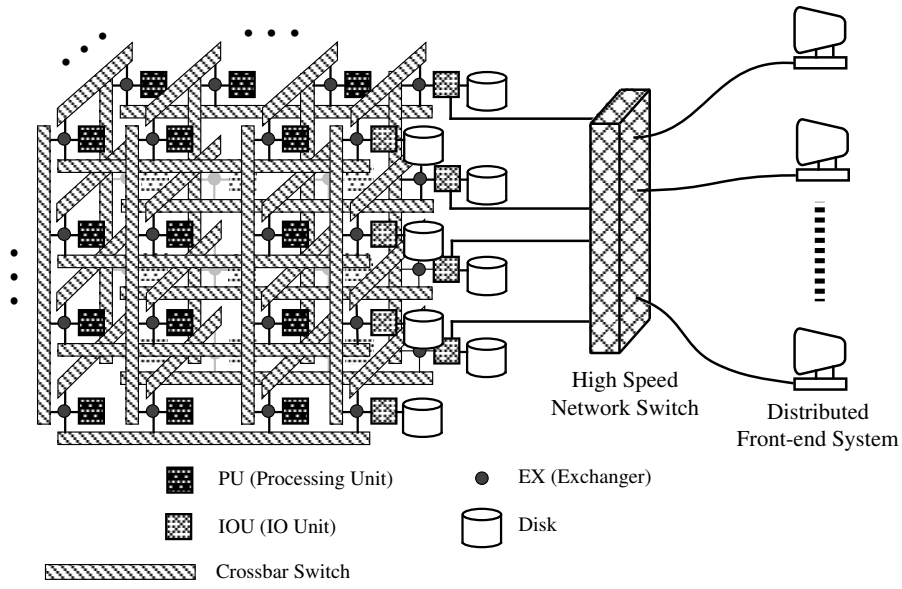

図 20: CP-PACS と密に結合した分散型フロントシステム

将来 CP-PACS へのデータの入出力をパラレルに行うための研究を開始し、物理学の研究のための基盤をさらに整備するための準備を行なう。このために、高いデータ処理能力を持つワークステーションクラスタを分散型フロントエンドシステムとして用い、CP-PACS と密に結合さ

せることにより、強力なデータ前後処理機能を備えた並列計算機システム (図 20) を構築し、この問題を研究する。

16.3 平成 8 年度

CP-PACS システムの動作確認、性能評価、簡単なテストプログラムによる試用を行う。設計仕様どうり稼働しない場合にはその原因を解明し、その解決を図る。システム調整、機能検証の後、システムの増強を図り、理論最高速度 600 GFLOPS 以上のシステムを完成させる。

これらの検討と並行し、素粒子物理学、物性物理学、宇宙物理学の応用計算プログラムを試験的に実行し、並列計算機システム全体に対するプログラムの実行動作確認と実行性能評価を行う。

その結果を踏まえて、本格的な計算の実行を開始し、 「場の物理」の難問題の解決を目指す。特に、素粒子の強い相互作用の基礎理論と考えられている量子色力学に対して高精度の計算を実行し、クオークからなる素粒子の質量スペクトル等の分光学的諸性質、宇宙初期に於ける低温ハドロン状態から高温プラズマ状態への相転移の特性等の解明を図る。さらに、ヒッグス粒子の性質等素粒子物理学の他の問題、物性物理学、宇宙物理学の問題に対しても順次計算を実行する。

これらの計算が効率的に行われるよう、CP-PACS の部分改良、修正を行って安定して稼働する状態を実現し、またシステムの動作状況チェックのためのルーチン及び並列処理向きソフトウエアの一層の充実を行う。さらに、素粒子物理学、物性物理学、宇宙物理学の研究課題のためのアルゴリズムの高速化と実行プログラムの開発・改良を前年度に引続き行い、計算効率の向上を図る。

これらの研究成果を公表するとともに、米国、欧州の専用並列計算機を開発製作している研究グループ、高性能計算機を用いて場の物理学を研究している研究者との情報交換・討論のために、国際シンポジュームを開催する予定である。

16.4 平成 9 年度以降

新プログラム研究は平成 8 年度で終了するが、計算物理学研究センターを中心として、CP-PACS を有効に活用し、既述の素粒子物理学、物性物理学、宇宙物理学の重要問題の研究の推進を図る。また、計算物理学発展のために欠かせない、物理学者と計算機工学者の共同研究体制を充実させ、使用者にとってさらに快適で高性能な計算環境を実現するための研究を推進する。

17 世界における専用並列計算機の開発状況

米国では、コロンビア大学、フェルミ国立加速器研究所 (ACPMAPS), IBM のワトソン研究所 (GF11)、欧州ではイタリアのローマ大学を中心として (APE)、専用並列計算機を開発するプロジェクトが進展し、日本の筑波大学における QCDPAX と同程度あるいは一桁遅い計算機が製作され、素粒子物理の研究に用いられている (表 21の第 1 グループ)。さらに、将来計画として、米国では、 MIT を中心とした Teraflops プロジェクト, コロンビア大学を中心とした 0.5 Teraflops プロジェクトが計画され、CP-PACS と同程度の能力の並列計算機の開発を目指しているが、まだ予算は認められていない (第 2 グループ)。また、イタリアのローマ大学を中心として次期計画 APE1000 が考えられているが、詳細はまだ不明である (第 3 グループ)。

表 21: 専用並列計算機のリスト

プロジェクト	理論最高速度 Gflops	稼働 (予定) 年
Columbia 16	0.25	1985
64	1.0	87
256	16	89
APE 4	0.25	86
16	1.0	88
QCDPAX	14	90
GF11	11	91
ACPMAPS	5	91
APE100	$6(\rightarrow 100)$	$92(\rightarrow 94)$
ACPMAPS upgraded	50	93
Teraflops	1,600	96
CP-PACS	≥ 300	96
0.5Teraflops	800	95
APE1000	~ 1000	

18 今後の目標

前節からも明らかなように、米国、欧州に於ても１９９６年頃には 500 GFLOPS 程度の計算機が製作され、物理学研究に用いられる可能性がある。従って、国際的な競争を考慮したとき、優れた物理学上の成果を挙げるためには、600 GFLOPS 以上の能力の並列計算機をなるべく早期に製作し、それを用いて物理学の研究を行なうことが必須である。

この能力の並列計算機は、平成４年度及び５年度の研究により、アーキテクチャ・半導体技術・実装等を含む製作面、さらには運用上問題となる占有床面積・電力消費量・発熱量等の総合的検討の結果、現実的に製作・運用が可能であることが明白になった。従って、本計画の今後として、その最優先目標は、以下の三点である。即ち、第一に、CP-PACS の組立てを本計画４年度末に完成させること、第二に組立て後の機能検証・調整を可能な限り早期に終了して本計画５年度目に物理計算のための本格稼働に入ること、そのため機能検証等の為の準備を現時点で開始すること、第三に、本格稼働開始までに応用プログラムの開発・改良を完成して本格稼働開始次第格子量子色力学等本計画の目標とする物理学の課題に関する計算を実行して世界に先進的な成果を挙げることである。

さらに本計画終了後を視野に入れるならば、より大きな目標として、計算物理学研究センターを中心として、一つには CP-PACS の有効利用を通じて計算物理学全般の推進を図り、また他方に於いて計算物理学発展の基盤となる物理学者と計算機工学者の共同研究体制の充実を図ることが挙げられよう。

参考文献

[1] 筑波大学計算物理学研究センター　平成４年度」研究報告.

[2] 岩崎洋一, “超並列計算機で素粒子、宇宙、物質を探る”, 日本物理学会誌 49 (1994)191-199.

[3] 日本物理学会編: 計算物理学 (培風館、1991).

[4] 専用並列計算機に関する最新の review として Y. Iwasaki, Computers for Lattice Field Theories, Nucl. Phys. B(Proc. Suppl.)34(1994)78 を挙げておく。また専用並列計算機に関する文献はその references を参照のこと。

[5] Y. Iwasaki, T. Hoshino, T. Shirakawa, Y. Oyanagi and T. Kawai, QCDPAX: A parallel computer for lattice QCD simulation, Comp. Phys. Comm. 49 (1988) 449-455; Y. Iwasaki, K. Kanaya, T. Yoshie, T. Hoshino, T. Shirakawa, Y. Oyanagi, S. Ichii and T. Kawai, Status of QCDPAX, Nucl. Phys. B(proc. Suppl.) 17 (1990) 259-262; T. Shirakawa, T. Hoshino, Y. Oyanagi, Y. Iwasaki, T. Yoshie, K. kanaya, S. Ichii and T. Kawai, QCDPAX - An MIMD array of vector processors for the numerical simulation of quantum chromodynamics, Proceedinds of Supercomputing '89, Reno, USA, Nov. 13-17, 1989, 495-504.

[6] Y. Iwasaki, K. Kanaya, T. Yoshie, T. Hoshino, T. Shirakawa, Y. Oyanagi, S. Ichii and T. Kawai, QCDPAX: Present Status and First Physical Results, Nucl. Phys. B (Proc. Suppl.) 20 (1991) 141-144; K. Kanaya, Y. Iwasaki, T. Yoshie, T. Hoshino, T. Shirakawa, Y. Oyanagi, S. Ichii and T. Kawai, Pure QCD at Finite Tempreture: Results from QCDPAX, Nucl. Phys. B (Proc. Suppl.) 20 (1991) 300-304; Y. Iwasaki, K. Kanaya, T. Yoshie, T. Hoshino, T. Shirakawa, Y. Oyanagi, S. Ichii and T. Kawai, Deconfining Transition of SU(3) Gauge Theory on $N_t = 4$ and 6 Lattices, Phys. Rev. Lett. 67 (1991) 3343-3346; T. Yoshie, Y. Iwasaki, K. Kanaya, S. Sakai, T. Hoshino, T. Shirakawa and Y. Oyanagi, Quenched Hadron Spectrum on a $24^3 \times 54$ Lattice, Nucl. Phys. B (Proc. Suppl.) 26 (1992) 281-283; K. Kanaya, Y. Iwasaki, T. Yoshie, T. Hoshino, T. Shirakawa, Y. Oyanagi, S. Ichii and T. Kawai, Deconfining Transition of SU(3) Gauge Theory on $N_t = 6$ Lattices, Nucl. Phys. B (Proc. Suppl.) 26 (1992) 302-304; Y. Iwasaki, K. Kanaya, S. Sakai and T. Yoshie, Quark Confinement and Number of Flavors, Nucl. Phys. B (Proc. Suppl.) 26 (1992) 311-313; Y. Iwasaki, K. Kanaya, S. Sakai and T. Yoshie, Quark Confinement and Number of Flavors in Strong Coupling Lattice QCD, Phys. Rev. Lett. 690 (1992) 21-24; Y.Iwasaki, K.Kanaya, T.Yoshié, T.Hoshino T.Shirakawa, Y.Oyanagi, S.Ichii, and T.Kawai, Finite Temperature Phase Transition of SU(3) Gauge Theory on $N_t = 4$ and 6 Lattices, Phys. Rev. D46 (1992) 4657-4667; Y. Iwasaki, K. Kanaya, S. Sakai, T. Yoshié, T. Hoshino, T. Shirakawa and Y. Oyanagi, Contamination of Excited States in Quenched QCD Hadron Propagators, Nucl. Phys. B (Proc. Suppl.)30 (1993) 397-400; Y. Iwasaki, K. Kanaya, S. Sakai and T. Yoshié, Quark Confinement in Multi-Flavor Quantum Chromodynamics, Nucl. Phys. B (Proc. Suppl.)30 (1993) 327-330; Y. Iwasaki, K. Kanaya, Leo Karkkainen, K. Rummukainen and T. Yoshie Interface tension in Quenched QCD, Phys. Rev. D49 (1994) 3540-3545; Y.Iwasaki, K.Kanaya, S.Sakai, T.Yoshié, Quantum Chromodynamics with Various Number of Flavors, Nucl. Phys. B (Proc. Suppl.)34 (1994) 314-316; Y.Iwasaki, K.Kanaya, S.Sakai, T.Yoshié

T.Hoshino and T.Shirakawa, High Statistics Calculations of Quenched QCD Spectrum using Various Quark Sources, Nucl. Phys. B (Proc. Suppl.)34 (1994) 354-356;

[7] 星野力編著: 「PAX コンピュータ」(オーム社、1985).

[8] 中村宏, 位守弘充, 伊藤元久, 中澤喜三郎, "レジスタウィンドウとスーパースカラ方式による擬似ベクトルプロセッサの提案", 並列処理シンポジウム JSPP '92, pp367-374, 1992 年 6 月 ; K.Nakazawa, H.Nakamura, H.Imori, and S.Kawabe, "Pseudo Vector Processor based on Register-Windowed Superscalar Pipeline", Proc. of Supercomputing '92, pp 642-651, Nov., 1992 ; 中村宏, 位守弘充, 中澤喜三郎, "レジスタウィンドウ方式を用いた擬似ベクトルプロセッサの評価", 情報処理学会論文誌, Vol34, No4, pp669-680, 1993 (平成 5 年度情報処理学会論文賞を授賞); H.Nakamura, H.Imori, K.Nakazawa, T.Boku, I.Nakata, Y.Yamashita, H.Wada, and Y.Inagami, "A Scalar Architecture for Pseudo Vector Processing based on Slide-Windowed Registers", Proc. of ACM International Conference on Supercomputing '93, pp.298-307, Jul., 1993; 中村宏, 中澤喜三郎, 李航, 位守弘充, 朴泰祐, "スライドウィンドウ方式に基づく擬似ベクトルプロセッサ", 情報処理学会研究報告, ARC-101-11, 1993 年 8 月 ; 位守弘充, 中村宏, 朴泰祐, 中澤喜三郎, "スライドウィンドウ方式による擬似ベクトルプロセッサ", 情報処理学会論文誌, Vol34, No12, pp2612-2623, 1993 ; H.Nakamura, K.Nakazawa, H.Li, H.Imori, T.Boku, I.Nakata, and Y.Yamashita, "Evaluation of Pseudo Vector Processor based on Slide-Windowed Registers", Proc. of Hawaii International Conference on System Sciences 27 (IEEE,ACM), pp368-377, Jan., 1994 ; H.Nakamura, T.Wakabayashi, K.Nakazawa, T.Boku, H.Wada, and Y.Inagami, "Pseudo Vector Processor for High-speed List Vector Computation with Hiding Memory Access Latency", Proc. of IEEE TENCON '94, Aug., 1994 (To appear).

[9] 朴 泰祐, 曽根 猛, 三島 健, 板倉 憲一, 中澤 喜三郎, 中村 宏, "ハイパクロスバ ・ネットワークにおける転送性能向上のための手法とその評価", 並列処理シンポジウム JSPP'94 予稿集, pp.129-136, 1994 年 5 月; 斉藤哲也, 朴泰祐, 中村宏, 中澤喜三郎, "超並列計算機のネットワークの実現可能性と性能評価", 情報処理学会研究報告 92-ARC-95, 1992 年 8 月; 朴 泰祐, 斉藤哲也, 板倉 憲一, 中澤 喜三郎, 中村 宏, "ハイパクロスバ ・ネットワークの性能評価", 電子情報通信学会技術報告 CPSY-93-40, 1993 年 11 月.