計算機アーキテクチャ　105－4
ハイパフォーマンス
コンピューティング　50－4
（1994．3．10）

ハイパフォーマンスコンピューティング プラットフォーム構想

関口智嗣　佐藤三久
電子技術総合研究所
Email: {*sekiguchi,msato*}*@etl.go.jp*

ハイパフォーマンスコンピューティング（High Performance Computing）とは従来のスーパーコンピュータが純粋に計算の高速処理を目指していたのとは異なり，スーパーコンピュータなどの計算機能力を用いて計算物理学，計算化学，遺伝子情報に代表されるような複雑事象の解明材料設計，薬物設計，様々な工学物設計を高精度，高品質に行なうことである．本稿では改めて超高速計算能力を提供するコンピュータシステムについて，それに関連するハードウエア，アーキテクチャ，オペレーティングシステム，ソフトウエア，アプリケーション，ネットワーク，性能評価などの各要素技術について再考を行なう．そして，このような超高速計算機システム利用の技術としてハイパフォーマンスコンピューティングプラットフォームという枠組を提示し，これにより洗い出された解決すべき問題点，開発すべき技術を位置付ける．

A Frame Work for High Performance Computing Platform Concept

Satoshi SEKIGUCHI　Mitsuhisa SATO
Electrotechnical Laboratory
Email: {*sekiguchi,msato*}*@etl.go.jp*

High Performance Computing System stands not only for high speed computing itself but also supporting high quality and accuracy artificial design for computational physics, chemical, and so on. In this article, we would like to reconsider a super-computer system and disscuss on technology of architecture, operating system, software, application, network, performance evaluation tool for high performance computing. The main objective here is proposing a new frame work named as "High Performance Computing Platform", with which we clarify the issues solved and the technologies developed.

1 はじめに

ハイパフォーマンスコンピューティング(High Performance Computing) とは従来のスーパーコンピュータが純粋に計算の高速処理を目指していたのとは異なり，スーパーコンピュータなどの計算機能力を用いて計算物理学，計算化学，遺伝子情報に代表されるような複雑事象の解明材料設計，薬物設計，様々な工学物設計を高精度，高品質に行なうことである．このため，ハイパフォーマンスコンピューティングを実現するシステムは高速コンピューティングシステムが中心であるばかりでなく，目的とする事象の解明，設計を支援するためのユーティリティも同様に重視したものとなる．例えば，高品質画像出力，格子点自動生成，データベースと有機的に連携したデータサービスなどの豊富な入出力機能，高速ネットワークによる利用などのユーティリティを充実する必要性は容易に理解できるであろう．

しかし，本稿では改めて超高速計算能力を提供するコンピュータシステムについて考察する，すなわち，それに関連するハードウエア，アーキテクチャ，オペレーティングシステム，ソフトウエア，アプリケーション，ネットワーク，性能評価などの各要素技術について再考を行なう．そして，このような超高速計算機システム利用の技術としてハイパフォーマンスコンピューティングプラットフォームという枠組を提示し，これにより洗い出された解決すべき問題点，開発すべき技術を位置付ける．

2 多様化する超高速計算システムアーキテクチャ

ハイパフォーマンスコンピューティングの核となる超高速計算システムをアーキテクチャとソフトウエアの両面からみてみる．超高速計算能力を供給するためのアーキテクチャとしては従来，素子技術により高速マシンサイクルを達成し，これをパイプライン処理することで高スループットを得たいわゆるベクトル型スーパーコンピュータ方式によるものであった．これは，単一CPUの内部で演算パイプラインの本数を複数とすることで，ピーク性能を向上させることができた(図1)．さらには，富士通 VPP500 に代表されるような単体性能が非常に高いプロセッサを 200 台以上並列に実行できる並列ベクトル型スーパーコンピュータシステムも登場してきた．

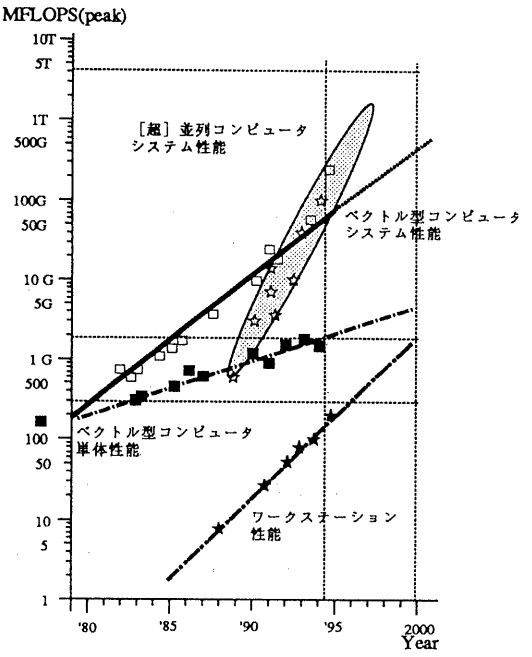


図 1: スーパーコンピュータ性能の動向

一方，マイクロプロセッサの処理能力並びに動作速度は上昇を続け，DEC Alpha AXP など200MHz を越える動作クロックを持つものも出現してきた．しかも，ベクトル型スーパーコンピュータとは異なり，量産効果が見込めるため，単位コストあたりの能力／性能ははるかに良好なものとなることがわかる．こうしたマイクロプロセッサを複数個，結合網やバスにより結合して，システム全体としての能力を向上させた並列コンピュータシステムも出現してきた．たとえば，CM-5, Intel/Paragon, Cray Superserver CS6400 といったシステムである (図2)．

このようなマイクロプロセッサを利用して高性能計算機を構成したシステムは片や Cray Research の T3D に代表されるような高

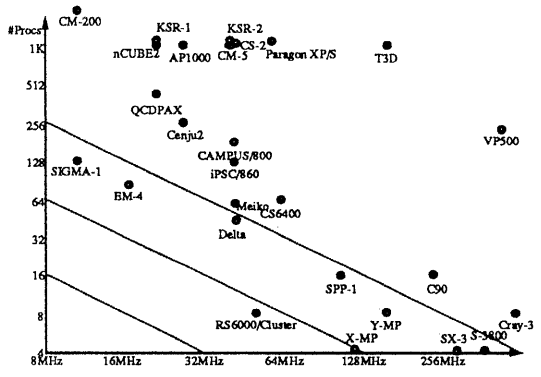


図 2: 動作クロックとプロセッサ台数 (推定値を含む)

速マイクロプロセッサを多数台接続し，スーパーコンピュータの実装と冷却技術を投入したシステムともう一方は特殊なネットワークを用ることなく，一般のイーサネットに接続された高速マイクロプロセッサをエンジンとするワークステーションを論理的にひとつのクラスタとして並列分散処理を実行するワークステーションクラスタがある [17]. 簡便なワークステーションクラスタでも速度的にはスーパーコンピュータに比肩するものもある [6] さらに，ワークステーションクラスタでは実装が容易なものの，ネットワークの性能がイーサネットで制限されるため，プロセッサの能力を活かすことができない．このため，IBM SP-1, Convex Exempler など特殊なネットワークスイッチを用意し高速化を目指したワークステーションラッククラスタがある．

このように，ハイパフォーマンスコンピューティングを支える高性能計算システムは，ベクトル型スーパーコンピュータ，並列ベクトル型スーパーコンピュータ，マイクロプロセッサ並列システム，ワークステーション (ラック) クラスタというように，多品種，多様式となってきた．従来はベクトル型スーパーコンピュータに対してプログラムの開発，最適化技術の開発を行なってくれば，時代の最高速計算機システムを十分に活用できたのと比較して，現在の状況は複雑である．すなわち，ベクトル型スーパーコンピュータに対して有効であった最適化手法と並列システムにおける最適化手法では共通するものと独自のものがある．例えば，共通する技法としてはループ長を長くする，すなわち並列計算可能部分を増やすことや，ループ中に例外処理を含まず均質な処理を行なうといったことがある．一方，データの分散の最適化についてはメモリバンクへのアクセス競合を意識していればよかったベクトル型スーパーコンピュータとアドレスによるアクセスレーテンシまで考慮しなくてはならない分散共有メモリシステムでは当然手法が異なってくる．ユーザはこれまで以上にシステムアーキテクチャを意識しなくてはプログラムが書けなくなってくるのであろうか．

3 多様化するソフトウエア / 記述モデル

ハイパフォーマンスコンピューティングのアプリケーションでは計算能力を活かして計算物理学，計算化学，遺伝子情報に代表されるような複雑事象の解明材料設計，薬物設計，様々な工学物設計を高精度，高品質に行なうことが可能となってきた．すなわち，計算機の演算速度，記憶容量の向上につれて簡略化されたモデルから，より詳細なモデルへ，実験解析から実験代替手段へ，さらには現実の実験が不可能な現象へのシミュレーションへと変化しつつある．こうした大規模格子系シミュレーション，連続系シミュレーションにおいては大規模連立一次方程式の解法，大規模固有値問題の解法，モンテカルロ計算等の莫大な計算量を必要とする計算がその計算時間を支配しており，超高速計算機システムの利用が不可欠となっている．

多様化したアーキテクチャに対応して，ソフトウエア，特に並列プログラムの記述言語ならびに並列実行モデルが多様化してきている．

本節ではこれまでに発表された並列マシンとそれ上での並列記述言語について比較を行なう．すなわち，あるマシンアーキテクチャを考えた場合，どのような並列言語をそのために，並列化に必要なプリミティブを整理

し，これまでのマシンでどのように実現されてきたかを考察する．

3.1 並列計算機用言語の分類

並列処理の研究からこれまでに様々な並列記述用言語が提案また実装されてきた。具体的な言語について簡単な例を交えて分類または解説を行なったものとして，[8] がある。またデータ駆動計算機用の言語に関しては [13] の例がある。

並列記述言語は大まかにユーザの記述と言語の実行モデルから分類を行なう．まず，ユーザの記述という観点からはユーザが並列性を陽に記述するのかしないのかという分類である．ユーザが並列性を記述するのかまたはコンパイラやアーキテクチャが並列性を自動的に抽出するのかという違いである．並列性の記述には fork/join などといった並列アクティビティの生成・消滅制御のためのプリミティブ， send/receive といった並列アクティビティ間通信のためのプリミティブ，**do-across** や **do-all** といったデータ並列のためのプリミティブ，単一代入規則の導入といった変数レベルの並列性のためのプリミティブが存在する．

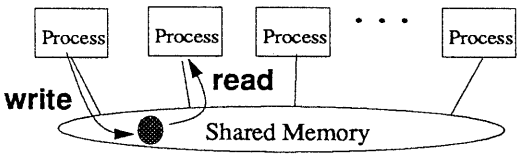
また，言語のセマンティクスとしては逐次型か並列型に大別する．すなわち，あるプログラムの記述を行なった場合，その実行モデルとして逐次計算が仮定されているか，または並列計算が仮定されているかの違いにより分類を行なう．並列記述または逐次記述が行なわれていない場合に default ととして逐次実行と並列実行のいずれが仮定されるかと換言してもよい．

この 2 つの視点からそれぞれの組合せで 4 つに分類を行なう．従来のこうした分類は主に，関数型，論理型と手続き型言語，専用言語と汎用言語という視点でなされてきたが，以下の分類ではそれぞれの項目において関数型，論理型，手続き型といった種別や専用，汎用といった種別がそれぞれ可能である．もちろん，イメージの困難な組合せや，実現が不可能な場合もあるが，ここでは従来の分類とは異なった視点からの試みのひとつである．

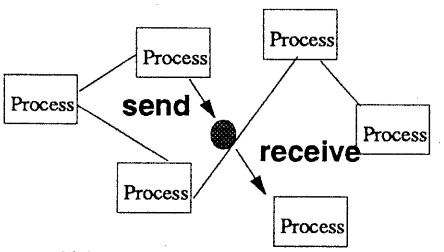
- **逐次セマンティクス & 並列記述**
  この分類に属する言語は従来の逐次型言語，すなわち，Fortran，C などに fork/join のような並列記述プリミティブ，send/receive のような並列アクティビティ間通信のプリミティブなどが導入されるたものである．例えば，HEP Fortran[9]．また，do-all や do-across や種々のコンパイラ指示文により並列実行可能な部分を明示するためのプリミティブもある．言語として通信のプリミティブを導入せずとも，メッセージパッシングライブラリを利用した並列記述はこの範疇に入る [2][3][4][7].
- **逐次セマンティクス & 自動抽出**
  この分類に属する言語は Fortran や C といった従来の逐次型言語からコンパイラが並列実行可能な部分を自動的に抽出することを前提とした言語である．ベクトル型スーパーコンピュータにおける自動ベクトル化コンパイラを前提とした Fortran などが典型的な例である．
- **並列セマンティクス & 並列記述**
  この分類に属する言語は例えば data parallel のようなプログラミングパラダイムを考える．ここでは，個々のデータに関して並列に実行されることを仮定しているため，ユーザが個々の処理を記述することが並列性を記述することであり，並列実行がなされる言語である．例えば，MASPAR の C と Fortran[11], CM-2 の C*[10]．また，関数型言語に単一代入規則を導入することにより並列性を記述したような言語もこの属に含まれる．例えば SISAL[1].
- **並列セマンティクス & 自動抽出**
  この分類に属する言語はユーザが何も記述しない場合にはコンパイラにより並列性が抽出され，並列実行が行なわれるものである．ユーザは並列性を記述するのではなく，どこを逐次に実行する必要があるかを指示する，すなわ

ち同期による並列性の制御を記述する必要がある．例えばDFC II[18] などである．

並列 / 分散処理を行なうシステムでは，それぞれ同時並行して実行される並列アクティビティ相互の同期 / 通信に関するソフトウエアのモデルとして『共有メモリモデル』と『メッセージパッシングモデル』がある (図3).



a) Shared Memoy Model



b) Message Passing Model

図 3: プロセス間通信モデル

ソフトウエアからみた場合，『共有メモリモデル』の場合にはすべてのアクティビティから参照・変更可能な変数空間がある (global naming space) ため，相互の同期，通信はその空間に read / write をすることにより実現することができる (図 3a)．一方，『メッセージパッシングモデル』はそれぞれの並列アクティビティは独立した変数空間を持っているため，特定のメモリを通じての通信ができず，陽に send / receive という命令を用いてデータ授受をする記述する (図 3b)．

このように並列処理を記述するだけでも多様なソフトウエアのバリエーションが存在することがわかる．しかも，実行させるシステムではユーザに選択の余地はなく，提供された並列記述モデルを利用するため，システム毎にプログラムを変更する必要がある．

4 3つのAとHPCプラットフォーム

現在のマイクロプロセッサ並列システムの先駆けともなったPAXを開発された星野力氏は，かつて著書[5]の中でハイパフォーマンスコンピューティングにおける最適設計について『三つのA』の重要性について述べている．すなわち，コンピュータが何に使われるのか (応用，Application) という意識と利用しうる素子や回路技術の前提のもとで，コンピュータのアーキテクチャ (Architecture) とアルゴリズム (Algorithm) を最適に設計することが重要であり，応用も理論の定式化やモデル化をコンピュータに適したものに改めるという自由度を考えれば，最適な設計とはこれらの三つのAの局面全体にわたって行なわれなければならない．

しかしながら，ここまでに見てきたように，ハイパフォーマンスコンピューティングをとりまくシステムアーキテクチャならびにソフトウエアは百家争鳴といっていいほど，様々なアーキテクチャのモデルとソフトウエアのモデルが存在する．ユーザの立場に立ってみれば，何を信じてプログラムを書いたらいいのか，また，何を対象として最適化をすればいいのか，そもそも，今書いているプログラムはその場限りのものなのか未来があるのかといった不安に対して何の回答も得られない．逆に，システムを設計する側からにしてもどのようなプログラム，ソフトウエアモデルに対して設計したらいいのか，最適化すればいいのか，など同様な不安が生じている．こうした背景のもと，ハイパフォーマンスコンピューティングに適したシステムを設計するには，多様な方向性をもったアーキテクチャ，ソフトウエア，アルゴリズムを相互に関連させるような共通の基盤を考えることは重要である (図4)．

それぞれに発展する3つのAの技術が関連する共通基盤になるとどのような利点が考えられるであろうか．アーキテクチャ，ソフトウエアモデルを同一ビューでとらえることにより，プログラムを異なるシステムに対するPortablity，プログラムの問題サイズに対する

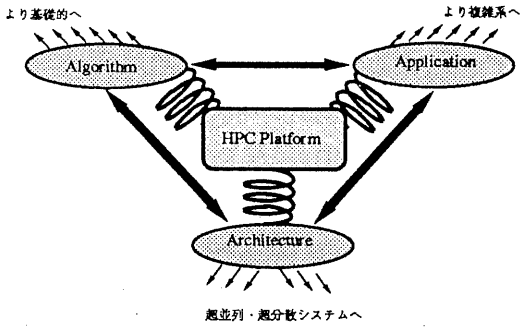


図 4: 3 つの A と HPC プラットフォーム

Scalability また，利用する計算システムに対する Scalability, プログラム開発における Efficiency, と同時に Reliability の向上が期待される (図 5).

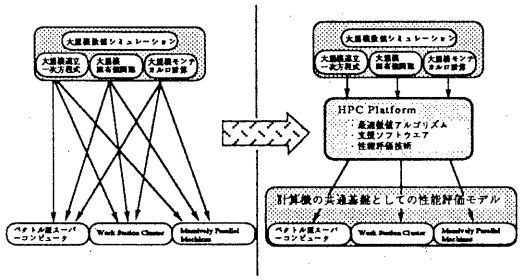


図 5: HPC プラットフォームによる利点

5 HPC プラットフォームと性能評価モデル

HPC プラットフォームの中心には性能評価モデルをおく．性能評価モデルとは，異なる計算機システムの能力の比較を可能とするために，それぞれの計算機システムで計算に必要な共通要素を抽出しモデル化 (抽象化，測定可能と) したものである [19][20] [16][15]. 重要な点はそれらのモデルを構成する要素は何らかの意味で測定が可能な点である．一般にはプログラムを実行することで測定を行なう．このプログラムのことはベンチマークプログラムと呼ばれる．また，任意のプログラムの実行プロファイルなどもプログラムの分析により，ベンチマークプログラムの測定結果から予測できることが必要である．したがって，単に適当なプログラムを実行してその実行速度を測定したり，Flops 値，MIPS 値を計算することが性能評価とはとらえない．すなわち，HPC プラットフォームの目標とするところは，データフロー／フォンノイマンといった違いや，命令レベル並列処理／データパラレル並列処理／ multi-threading 並列処理といったアーキテクチャやソフトウエアのモデルを越えたところで基盤となる性能評価モデルを構築することで，そのモデルに基づいた性能測定を可能とすることである．このことは，性能評価モデルに基づいて性能測定を行なったその結果を利用すれば，任意のプログラムをある並列化手法を用いて並列化した場合 (プログラミングと呼ぶ) の実行時間や速度といったプロファイルの予測が可能となる．さらに，

1. システムを固定した場合それを実行するのに最適なプログラミングを定める
2. プログラミングを固定した場合それを実行するのに最適なシステムを定める

ことを可能とすることが性能評価モデル構築の目標である．現在の最適化技術において「最適化」とは何かを HPC プラットフォームに基づいて改めて定義する必要がある．また，これを定義できる理論基盤を与えるのが HPC プラットフォームと言って良い．

こうした指標の一例をあげる．計算モデルにより実現される方法は異なるが，細粒度並列処理において並列アクティビティの制御という観点から同期の性能と分岐の性能は非常に重要となる．すなわち，並列アクティビティが増加すれば並列実行の可能性が増加するものの，同期や分岐処理においてアクティビティを制御するコストもそれに伴って増加する．よって，並列処理システムの評価としてどの程度の並列性が適正であるかは同期・分岐処理のコストに関わってくる．この処理能力を示す指標として同期性能の評価指標 SYOPS，生成性能の評価指標 PAGPS，分岐性能の評価指標 SWIPS について以前提案を行なったことがある [14].

ここまで，見てきたような HPC プラットフォームを構築するのに必要な技術を具体的に考えてみる．中心になるのは性能評価モデ

ルであるが，これは，アーキテクチャとソフトウエア・アプリケーションを連続的に結び付けるものでなくてはならない．我々は，以前，階層化ベンチマークと呼ばれるモデルを提案し [20][16] [15]，それぞれの階層間で相互に情報を交換する手法を述べた．このモデルでは，階層に所属するモジュールを分解，合成，交換が可能となることが必要用件である．そのために，こうしたライブラリの入出力，機能の標準化，ソケット式にする技術を開発する．もちろん，性能モデルに基づいたシステムを対象とし，個々のアーキテクチャに最適化を施したようなライブラリの開発を行なうわけではない．これにより，異なったアーキテクチャ間で，実際に公平な速度の比較が可能となる．また，スケーラビリティ，ポータビリティ等を性能評価モデルに基づいて改めて定義する必要がある [12].

6 おわりに

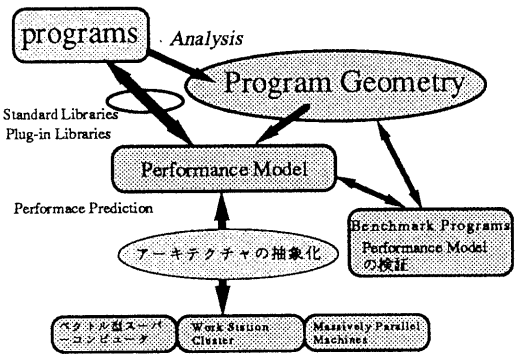


図 6: HPC プラットフォーム周辺技術

ハイパフォーマンスコンピューティングにおける共通基盤としてハイパフォーマンスコンピューティング・プラットフォーム技術の必要性について述べた．まとめにかえて，これに関連する周辺技術を列挙すると (図 6)，

- 計算モデル等の差異を越えた性能評価モデルの構築
- 標準 (数値計算用) ライブラリの構成，差し替えを容易にするプラグイン化
- 評価用グラフを作成するための指標と軸
- プログラムの特性・特徴を記述するための技術，特に並列性などのプログラム幾何的特徴を記述をする手法
- 性能モデルを用いて，性能予測を行なう手法とこれを修正し，新たなモデルを構築する技術
- 性能モデルの個々の数値を与えるベンチマークプログラムとそれによる検証法
- 性能モデルに基づく最適化 (ハード，ソフト) 技術の開発

こういった，技術の研究・開発を個別に行なうのではなく，提案したプラットフォームに基づいた方法で行なうことが今後の重要な課題であると考える．

謝辞 日頃より御討論いただく，電子技術総合研究所 計算機方式研究室各位ならびに The Hokke-Club に感謝致します．

参考文献

[1] J. M., et al., *"SISAL Streams and Iteration in a Single Assignment Language Language Reference Manual Version 1.2"*, Technical Report M-146, Lawrence Livermore National Laboratory, March 1985.

[2] Boyle, J., Butler, R., Disz, T., Glickfeld, B., Lusk, E., Overbeek, R., Patterson, J., and Stevens, R., *Portable Programs for Parallel Processors*, Holt, Rinehart, and Winston, 1987.

[3] Butler, R., and Lusk, E., *"User's guide to the p4 programming system"*, Technical Report ANL–92/17, Argonne National Laboratory, Argonne, IL 60439, October 1992.

[4] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., *"PVM 3.0 user's guide and reference manual"*, Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, February 1993.

[5] 星野力, *PAX* コンピュータ —高並列処理と科学計算—, オーム社, March 1985.

[6] 日向寺祥子, 長嶋雲兵, 細矢治夫, 関口智嗣, 佐藤三久, "大規模実対称行列の状態密度の計算とその並列化", 研究報告 93-HPC-48-8, 情報処理学会, August 1993.

[7] Ikudome, K., Kolawa, A., and Flower, J., "自動並列化システム ASPAR と並列処理実行環境 Express", 電子情報通信学会技術研

究報告コンピュータシステム研究会, Vol. 91, No. 130, 1991, pp. 213–220.

[8] Karp, A. H., "Programming for Parallelism", *IEEE Computer*, Vol. 20, No. 5, 1987, pp. 43–57.

[9] Kowalik, J. S. (ed.), *Parallel MIMD Computation: HEP Supercomputer and Its Applications*, The MIT Press, 1985.

[10] 日本シンキングマシンズ(株), "コネクションマシン CM-5", 1991.

[11] (株) 理経, "MASPAR MP1 アーキテクチャー紹介セミナー資料", 1991.

[12] 佐藤三久, 関口智嗣, "並列計算機システムのスケーラビリティについて", 研究報告 93-HPC-49-2, 情報処理学会, October 1993.

[13] 関口智嗣, 山口喜教, "データ駆動計算機用の高級言語と処理系", 情報処理, Vol. 31, No. 6, 1990, pp. 753–762.

[14] 関口智嗣, 佐藤三久, "細粒度並列処理における性能評価モデル — 節度ある並列性を求めて —", "並列処理シンポジウム JSPP'92", 情報処理学会, June 1992.

[15] 関口智嗣, 佐藤三久, *"HPC における性能評価 — 測定から科学へ —"*, 研究報告 93-HPC-46-5, 情報処理学会, April 1993.

[16] 関口智嗣, 小柳義夫, "スーパーコンピュータの性能評価の現状", 応用数理, Vol. 3, No. 1, 1993, pp. 27–38.

[17] 関口智嗣, 長嶋雲兵, "ワークステーションクラスタとメッセージパッシングライブラリ", 研究報告 93-HPC-47-3, 情報処理学会, June 1993.

[18] Sekiguchi, S., Shimada, T., and Hiraki, K., "Sequential description and parallel execution language DFC II for dataflow supercomputers", *Proc. of 1991 International Conference on Supercomputing, Cologne, Germany*, pp. 57–66, June 17-21 1991.

[19] 日本応用数理学会 次世代スーパーコンピュータ性能評価技術調査委員会, 平成 *2* 年度 次世代スーパーコンピュー タ性能評価技術調査報告書, (社) 日本機械工業連合会, March 1991.

[20] 日本応用数理学会 次世代スーパーコンピュータ性能評価技術調査委員会, 平成 *3* 年度 次世代スーパーコンピュー タ性能評価技術調査報告書, (社) 日本機械工業連合会, March 1992.