ハイパフォーマンスコンピューティング 60－12
（1996. 3. 4）

並列FFTクラスの試作

高橋 毅, 山口 隆弘
(株)アドバンテスト研究所 第1研究部
Email: {takeshi, jamax}@atl.advantest.co.jp

われわれは並列信号処理系にオブジェクト指向技術を適用して，計算処理の各ノードへの分割やノード間データ通信などをクラスのなかに隠蔽してしまい，ハードウェアの構造が変わっても再利用できる並列信号処理系を実現した。

An Object-Oriented Implementation of the Parallel Fast Fourier Transform

Takeshi Takahashi Yamaguchi Takahiro
1st Research Laboratory, ADVANTEST LABORATORIES LTD.
Email: {takeshi, jamax}@atl.advantest.co.jp

We have applied object-oriented technology to implement parallel digital signal processing, by encapsulating in classes such complex functions as partitioning of computation processes among nodes and data communication between nodes. This will enable us to realize reusable parallel digital signal processing system even if the architecture of the hardware changes.

1．はじめに

従来の並列処理ソフトウェアは，処理の分割やデータ通信などをハードウェア構造に1対1対応させざるえないため，再利用性に乏しかった。また，1つの計算を2つ以上の計算に並列化すると，複数の計算の間で余分の通信をおこなう必要が生じる。この通信オーバヘッドは，バスとメモリーアクセスのスピードにより下限をあたえられる。しかし，この通信スピードはバスよりコンピュータネットワークを適切にもちいたほうが，こんご高速になる可能性がある。

これらの技術展望のもと，われわれは並列信号処理系にオブジェクト指向技術を適用して，処理の分割やデータ通信などをクラスのなかに隠蔽してしまい，ハードウェアの構造が変わっても再利用できる並列信号処理系を実現しようとしている。とくに，データを送受信するバスとしては，オンチップのバス，内部バス，コンピュータ・ネットワークをターゲットとしている。これらのバスにたいしてスケーラブルに対応できる通信クラスを実現できると，ハードウェアの性能向上により，内部バス中心の計算パラダイムからネットワーク中心の計算パラダイムにシフトしても，ソフトウェアを変更せずに最適な性能を発揮できる並列信号処理系を最終的に実現できる。

2．FFTアルゴリズムとその並列化

N点の入力データに対するFFTは，Nが2のべきであるとき，データの並び換え処理とN/2組の演算対からなるlog2N段の演算ステージで構成される。図1にN=8のFFTアルゴリズムをしめす。

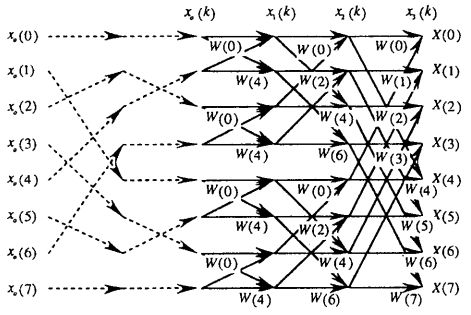
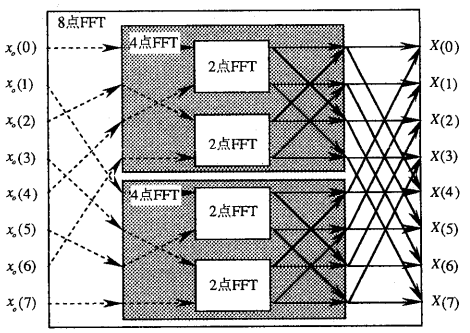


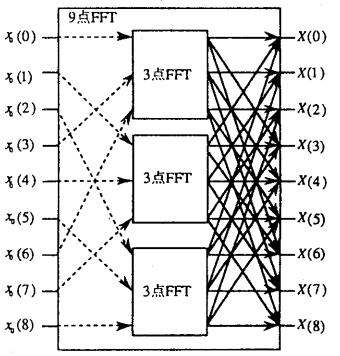
図 1: FFTアルゴリズム(N=8)

N点FFTにおける(log2N-1)段までの演算は2つのN/2点FFTに分解できる。この2つのN/2点FFTの間にはデータのやり取りがないため，独立に計算できる。この2つのN/2点FFTを異なる演算装置上で並列に処理すれば，全体の処理時間を短縮できる。同様に，N/2点FFTは2つのN/4点FFTに分解でき，独立に計算できる。この独立に計算できる構造を利用して，並列計算用のアルゴリズムを創った。

たとえば，N=8のFFTは2点FFTを4つ並列計算するアルゴリズムに変換できる(図2(a))。さらに，2以外の基数のFFTも同様に並列化できる(図2(b))。



(a)



(b)

図 2: FFTアルゴリズムの並列化
(a)N=8, (b)N=9

3．プロセス構成

FFTの並列処理を実現するために，分解したアルゴリズムをプロセスとみなし，プロセスの木構造を構成する。(1) 木構造の根は全体のFFTを管理する。(2) 根以外の分岐節点は中間的なデータ分割部分をおこなう。(3) 葉はそれ以上分割できない逐次FFTを実行する。ここで，木構造における節点間のリンクはデータ通信をおこなう通信経路に相当する。並列度(並列化する分割数)は分岐節点の層の深さにより制御できる。

図3にN=8，分割数4の並列FFTを実現するプロセス構成をしめす。各FFT計算プロセスは，サブFFT計算プロセスに再帰的に分解できる。

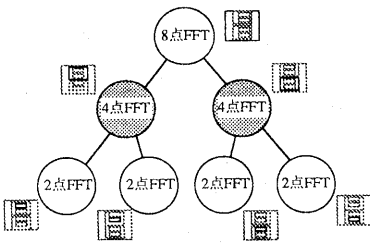


図 3: 並列FFTアルゴリズムの構成

4．クラス構成

つぎに，図4にしめした並列FFTのためのクラスを説明する。(1)根は，FFTを直接計算する代わりに，分割したデータをサーバへ送信し，結果を受信する内部クライアント・クラスである。このクラスは，内部サーバと通信するために‘通信クラス’をもつ。ユーザにたいしては‘並列FFTクラス’としてみえる。この並列FFTクラスは並列処理の詳細を情報隠蔽しており，逐次FFTクラスのインターフェースを継承している(ポリモフィズム)。(2)分岐節点は，内部クライアント・クラスから内部サーバとしてデータを受信する。つぎに，役割を内部クライアントにスイッチし，次段の分岐節点または葉のサーバへ分割したデータを送信する。(3)葉は，クライアントとデータ受送信をおこなうための‘通信クラス’とFFTを計算する‘逐次FFTクラス’からなる。

ベクトルソケットクラスは，ソケットクラスのインプリメントを継承し，ベクトルクラスの通信をおこなう。

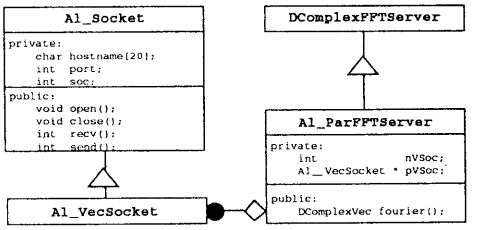


図 4: 並列FFTクラスの構成

5．イベント・トレース

並列FFTクラスのイベント・トレースを図5にしめす。同じ性能のマシンをもちい，分割数2としたときの並列FFTの計算時間は次式であたえられる。

$$
\begin{aligned}
&[\text{N点並列FFTの計算時間}] = \\
&\quad [\text{並び換え処理時間}] \\
&\quad + [\text{N/2点FFT計算時間}] \\
&\quad + [\text{最終段の計算時間}] \\
&\quad + [\text{N/2点データの通信時間(往復)}] \times 2
\end{aligned}
\qquad (1)
$$

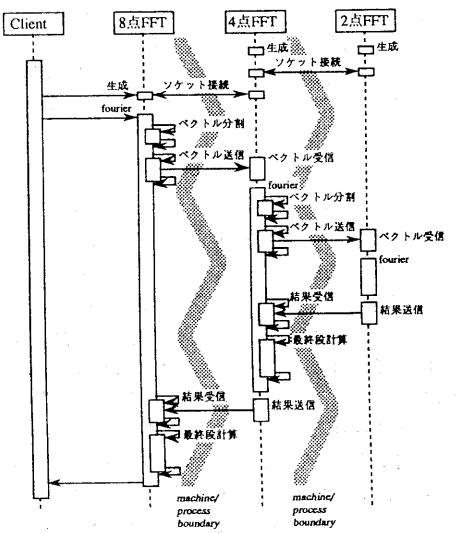


図 5: 並列FFTクラスのイベント・トレース

6．実験: 並列FFTの計算時間

試作したシステムのFFT計算時間をワークステーション・ネットワークおよびパーソナルコンピュータ・ネットワークにおいて測定した。ネットワークは共にイーサネットによりTCP/IP接続されており，通信性能はそれぞれ，920KBytes/sec, 450KBytes/secであった。

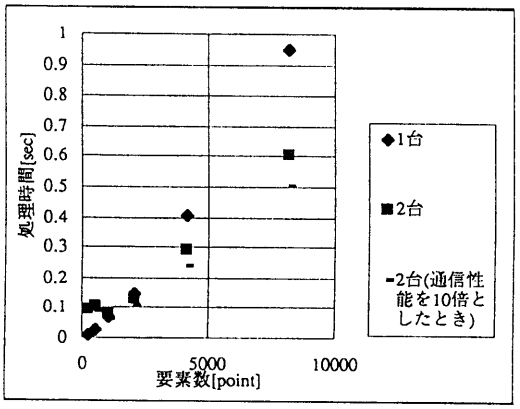


図 6: ワークステーションでのFFT計算時間

ワークステーション・ネットワークは2台のUNIXマシン(Fujitsu S-4/20, OS:Solaris2.3)で構成した。2台にすると，1台のときの64%の計算時間である(N=8192点)。(1)式の最後の項の通信時間が0.1倍(0.01倍)になったときは59%(57%)となる(図6)。これから通信用ハードウェアに投入すべきコストと計算時間短縮効果の関係がわかり，ハードウェア構成を決定する指標にできる。

一方，パーソナルコンピュータ・ネットワークは演算性能の異なる4台のWindowsマシン(IBM ThinkPad シリーズ 755c, 755cd, 750c, 701c, OS:Microsoft-Windows95)で構成した。2台, 4台による計算時間は1台のときの64%, 67%である(N=8192点)。4台のマシンは演算性能が異なるため，全体の計算時間は最も処理性能のおとるマシンに依存する。1台での測定値をもとに，同じ性能のマシン4台をもちいたときの計算時間を見積ると，1台のときの53%になる。また，4台のマシンが同じ性能,かつ,通信時間が0.1倍(0.01倍)になったときは33%(31%)になると見込まれる(図7)。

並列計算環境は，オンチップのバス，システムの内部バス，コンピュータ・ネットワークとスケールが変化する。しかし，並列FFTクラスは通信を内部にカプセル化している。したがって，通信方式の進化は並列FFTクラスと，その利用者には直接影響しない。

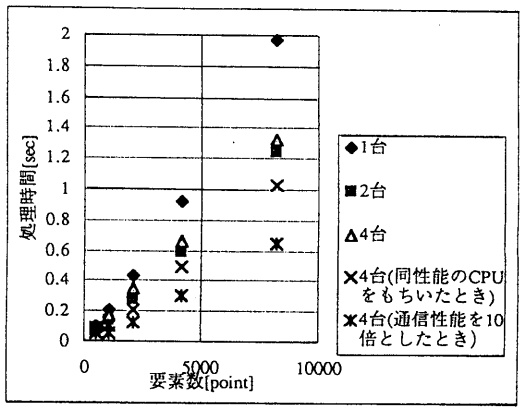


図 7: パーソナルコンピュータでのFFT計算時間

7．おわりに

並列FFTにオブジェクト指向技術を適用することにより，ハードウェアの‘構造変化’と‘性能向上’に対応できることをしめした。しかし，最適なプロセッサ構成の決定(負荷分散)は手動でおこなっている。こんご，負荷分散の自動化をおこなうクラスを創る予定である。

参考文献

[1] E. Oran Brigham, 高速フーリエ変換，科学技術出版，1979.

[2] P. Valkenburg, USENET group comp.sources.unix, fft, Vol.18, Issue 20.

[3] B. Stroustrup, プログラミング言語C++第2版，トッパン，1993.

[4] SunOS 5.3 Network Interfaces Programmer's Guide, SunSoft, 1993.

[5] E. Gamma, et al. Design Patterns, Addison Wesley, 1994 .