自動分割システムRODSにおけるNDFD処理系について

4J−10

山下利夫　松山実　横井利彰

武蔵工業大学　工学部

1. はじめに

RODS(Repetitive Optimum Divising System)は、ネットワーク環境で動作するソフトウェアの開発を支援するシステムである[1]。本システムの基本構成を図1に示す。

RODS上では、ソフトウェアをNDFD(Network Data Flow Diagram)を用いて記述する。NDFDはDFDをRODS用に拡張したものである。記述したソフトウェアは、自動分割システムを用いて分割し、ネットワーク上の各マシンに配送される。分割前のNDFDを、NDFDソースと呼んで分割後のものと区別する。

配送したNDFDは、各マシン上のNDFD処理系を用いて実行可能コードに変換する。これらはネットワークを介してデータ交換を行いながら目的の処理を行なう。

本稿ではこのNDFD処理系について述べる。

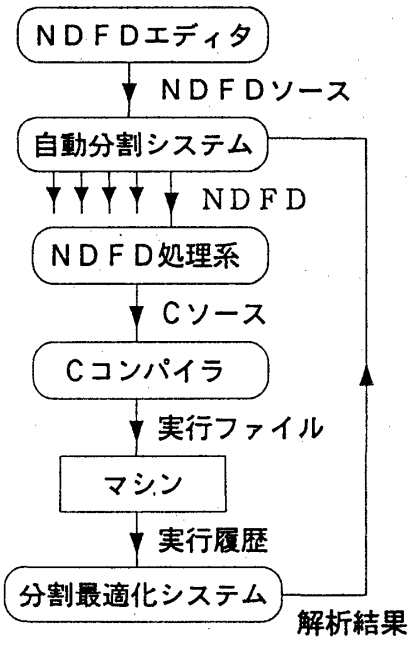

図1　RODSの基本構成

2. NDFD処理系

NDFD処理系は、NDFDからC言語ソースへのトランスレータとして実現した。ただし、単純な変換ではなく、変換後のそれぞれのC言語ソースが、ネットワークを介して協調して並列動作を行うような変換を行う。そのためにマネージャと呼ばれる、幾つかのC言語関数群を変換時に使用する。

2.1. ノード

NDFDを記述する処理の最小単位はノードである。ノードには基本的な機能を持つプリミティブ・ノードと、ユーザが定義するライブラリ・ノードの2種類があるが、本稿ではプリミティブ・ノードのみを扱い、単にノードと記す。

また、ノードの動作は以下のように5段階に大別できる。

生成　ノードが動作を開始する
待機　入力データが全て揃うまで待つ
実行　入力データを処理する
送出　出力データが全て受け取られるまで待つ
消滅　ノードが動作を終了する

2.2. NDFD形式データ

NDFDエディタで記述されたNDFDは、以下の3つの形式のファイルに保存される。それを以下に示す。

2.2.1. データフロー・ファイル

各ノードのつながりを格納するファイルである。ノードを識別するための通し番号が、分岐のあるリスト構造で格納されている。このファイルが、自動分割システムによって分割される。

2.2.2. ノード・ファイル

各ノードの情報を格納する、辞書型のファイルである。ノードが入出力するデータの型・数や、実際の処理を記述したC言語の関数、その関数の使用するワークエリア情報等を含む。

Compiler of NDFD (Network Data Flow Diagram)
Toshio YAMASHITA,Minoru MATSUYAMA,Toshiaki YOKOI
Musashi Institute of Technology

2.2.3. データ・ファイル

整数型や文字列型等、各ノードの扱うデータの型に関する情報を格納する、辞書型のファイルである。

2.3. トランスレータ

上記のＮＤＦＤ形式ファイルをもとに、Ｃ言語ソースを生成するのがトランスレータである。

ノード・ファイル及びデータ・ファイルを参照ながら、データフロー・ファイルに記述されたノードを、Ｃ言語の関数に変換する。それらの関数と、以下に示すマネージャを組み合わせ、Ｃ言語のソースファイルを出力する。

2.4. マネージャ

トランスレートする際に必要とする関数群をマネージャと呼び、目的によって以下の５種類に大別できる。

2.4.1. メイン・マネージャ

メイン・マネージャはＮＤＦＤ形式ファイルをトランスレートする際に、メイン部分となるＣプログラムである。ノードの接続に関するデータを保有し、全体を管理する。

2.4.2. ノード・マネージャ

ノード・マネージャは、ノードの生成と消滅を受け持つ。各ノードに対応するＣ言語の関数を保有し、必要なワークエリアの確保と初期化を行なう。また送出終了後のノードを終了させる。

ノードの生成が要求された場合、そのノードが２台以上のマシン内で生成可能であるならば、その時点でより負荷の軽いマシンを選択する。

2.4.3. データフロー・マネージャ

データフロー・マネージャは、ノードの待機と送出を受け持つ。待機中のノードに必要な入力データを用意し、動作終了後に出力データを対象ノードへ配送する。

2.4.4. ネットワーク・マネージャ

ネットワークに関する処理を受け持つマネージャである。ノード・マネージャが生成を行う際の優先度チェックや、データフロー・マネージャによるマシン間のデータ配送を受け持つ。

また実行開始時に、端末マシン以外のメイン・マネージャを起動させる。これを行うために、ＵＮＩＸマシンでは専用のデーモンプログラムを利用する。

2.4.5. ログ・マネージャ

ログ・マネージャは、ノードの実行の様子をファイル等に記録として残す。ＲＯＤＳの分割最適化システムがこの記録を解析し、より効率的な再分割を行なうことができる。

2.5. 生成されたCプログラムの動作

生成されたＣプログラムの実行の概略を、以下に示す。

1. 実行開始(メイン・マネージャ)
2. 他のマシン上のメイン・マネージャを起動し、接続する(ネットワーク・マネージャ)
3. 入力データの配送(データフロー・マネージャ)
4. 対象ノードが存在しない場合、ノード生成(ノード・マネージャ)
5. ノードに対応したＣ関数を呼びだし、処理。
6. 出力されたデータを収集(データフロー・マネージャ)
7. 処理を終えたノードを消滅させる(ノード・マネージャ)
8. 処理すべきデータがまだあれば、3.へ。
9. 終了処理(メイン・マネージャ)

3. おわりに

各種のマネージャと呼ばれるＣ言語プログラムを埋め込むことにより、ＮＤＦＤで記述されたソフトウェアをＣ言語にトランスレートすることができる。自動分割システムによって分割された各々のＮＤＦＤは、各マシン上のＮＤＦＤ処理系によってトランスレートされ、自動的に実行可能コードが生成できる。またそれらはネットワークを介して協調動作することができる。

ＮＤＦＤ処理系が動作し、ＮＤＦＤエディタや自動分割システム等、ＲＯＤＳの他の構成要素の出力を、実際に実行して検証することが可能になった。今後もそれぞれの要素を並行して研究を進めていく。

参考文献

[1] 山下利夫,松山実,横井利彰, “ソフトウェア自動分割システムＲＯＤＳの概要”, 情報処理学会第４５回全国大会講演論文集(分冊５), pp.319-320,1992