

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第6120276号  
(P6120276)

(45) 発行日 平成29年4月26日(2017.4.26)

(24) 登録日 平成29年4月7日(2017.4.7)

(51) Int.Cl. F I  
G O 6 F 9/44 (2006.01) G O 6 F 9/06 6 2 O A

請求項の数 5 (全 28 頁)

<p>(21) 出願番号 特願2013-109741 (P2013-109741)                  (22) 出願日 平成25年5月24日 (2013.5.24)                  (65) 公開番号 特開2014-229173 (P2014-229173A)                  (43) 公開日 平成26年12月8日 (2014.12.8)                  審査請求日 平成28年5月18日 (2016.5.18)</p> <p>(出願人による申告)平成24年度、独立行政法人科学技術振興機構、戦略的創造研究推進事業(個人型研究(さきがけ))に関する委託研究、産業技術力強化法第19条の適用を受ける特許出願</p>	<p>(73) 特許権者 504171134                  国立大学法人 筑波大学                  茨城県つくば市天王台一丁目1番1                  (74) 代理人 100113608                  弁理士 平川 明                  (74) 代理人 100105407                  弁理士 高田 大輔                  (72) 発明者 山際 伸一                  茨城県つくば市天王台一丁目1番1 国立                  大学法人筑波大学内                  審査官 石川 亮</p> <p>(56) 参考文献 特開2012-022580(JP, A)                  )</p> <p style="text-align: right;">最終頁に続く</p>
--	--

(54) 【発明の名称】 アクセラレータ処理実行装置、及びアクセラレータ処理実行プログラム

(57) 【特許請求の範囲】

【請求項1】

C P Uと、  
 前記C P Uに接続され、データを記憶する第1の記憶手段と、  
 前記C P Uに接続され、前記第1の記憶手段とは異なる、データを記憶する第2の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備え、  
 前記C P Uは、前記アクセラレータに関する処理の実行を制御する実行制御手段を有し、  
 前記実行制御手段は、  
 前記アクセラレータプログラムと、前記アクセラレータプログラムが実行されて出力される出力データ毎に前記第2の記憶手段に確保される出力バッファと前記第1の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、  
 前記アクセラレータに関する処理の手順を決定するために前記実行形式とは別に予め定められている規則と前記実行情報とに基づいて決定した処理手順を実行する処理手順実行手段と、を有し、  
 前記処理手順実行手段は、  
 前記アクセラレータに前記アクセラレータプログラムを送信するプログラム送信処理と、  
 前記第2の記憶手段に前記出力バッファを確保する出力バッファ確保処理と、

10

20

前記プログラム送信処理及び前記出力バッファ確保処理よりも後に、前記アクセラレータに前記アクセラレータプログラムを実行させるプログラム実行処理と、

前記プログラム実行処理により前記出力バッファに出力された前記出力データを、当該出力バッファに対応する前記第1の記憶手段における出力先に送信する出力データ送信処理と、を実行する、

アクセラレータ処理実行装置。

【請求項2】

前記実行情報には、前記アクセラレータプログラムに対する入力データ毎に前記第1の記憶手段における記憶場所と前記第2の記憶手段に確保される入力バッファとを対応付ける入力データ対応情報が含まれ、

10

前記処理手順実行手段は、

前記プログラム実行処理よりも前に、前記第2の記憶手段に前記入力バッファを確保する入力バッファ確保処理と、

前記入力バッファ確保処理よりも後に、前記記憶場所に記憶されている前記入力データを、当該記憶場所に対応する前記入力バッファに送信する入力データ送信処理と、を更に実行し、

前記処理手順実行手段は、前記入力データ送信処理よりも後に前記プログラム実行処理を実行する、

請求項1記載のアクセラレータ処理実行装置。

【請求項3】

20

前記入力データ対応情報に含まれる前記第1の記憶手段における記憶場所、又は前記出力データ対応情報に含まれる前記第1の記憶手段における出力先として、メモリ上の共有領域が指定される、

請求項2記載のアクセラレータ処理実行装置。

【請求項4】

前記実行情報には、前記入力バッファ及び前記出力バッファの組み合わせと入替回数とを示すスワップ情報が含まれ、

前記処理手順実行手段は、前記プログラム実行処理において、前記アクセラレータプログラムの1回の実行が完了する毎に前記入力バッファと前記出力バッファとを前記組み合わせに基づいて入れ替えた後に前記アクセラレータプログラムを実行する処理を、前記入替回数だけ繰り返す、

30

請求項2又は3記載のアクセラレータ処理実行装置。

【請求項5】

CPUと、

前記CPUに接続され、データを記憶する第1の記憶手段と、

前記CPUに接続され、前記第1の記憶手段とは異なる、データを記憶する第2の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備えるコンピュータに、

前記アクセラレータに関する処理の実行を制御する実行制御手段を実現させるためのアクセラレータ処理実行プログラムであって、

40

前記実行制御手段は、

前記アクセラレータプログラムと、前記アクセラレータプログラムが実行されて出力される出力データ毎に前記第2の記憶手段に確保される出力バッファと前記第1の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、

前記アクセラレータに関する処理の手順を決定するために前記実行形式とは別に予め定められている規則と前記実行情報とに基づいて決定される処理手順を実行する処理手順実行手段と、を有し、

前記処理手順実行手段は、

前記アクセラレータに前記アクセラレータプログラムを送信するプログラム送信処理と

50

、  
 前記第2の記憶手段に前記出力バッファを確保する出力バッファ確保処理と、  
 前記プログラム送信処理及び前記出力バッファ確保処理よりも後に、前記アクセラレータに前記アクセラレータプログラムを実行させるプログラム実行処理と、  
 前記プログラム実行処理により前記出力バッファに出力された前記出力データを、当該出力バッファに対応する前記第1の記憶手段における出力先に送信する出力データ送信処理と、を実行する、  
 アクセラレータ処理実行プログラム。

【発明の詳細な説明】

【技術分野】

10

【0001】

本発明は、特にCPUに接続されたアクセラレータハードウェアに処理を実行させるアクセラレータ処理実行装置、及びアクセラレータ処理実行プログラムに関する。

【背景技術】

【0002】

GPU (Graphics Processing Unit) がもつ潜在的な超並列性を生かした計算技法が2006年くらいから市場に台頭するようになり、GPUを使ったハイパフォーマンスコンピューティング分野が急速に発達している。最近では、GPUがCPU (Central Processing Unit: 中央処理装置) の周辺デバイスとして動作するアーキテクチャモデルに基づいて、CPUで動作するホストプログラム (CPU側プログラム) が、GPUをアクセラレータハードウェアとして扱い、アクセラレータに実行させるアクセラレータ側プログラムをCPU側からアクセラレータ側に送信 (ダウンロード) し、アクセラレータに実行させるプログラミングスタイルが定着している。このアクセラレータ側プログラムは、GPUハードウェアのアーキテクチャの超並列性を引出すためにデータはストリームで提供されるといった理念に基づき、設計されている。

20

【0003】

アクセラレータにアクセラレータ側プログラムを実行させるためのプラットフォームとして、本発明者らが開発したプラットフォームが知られている (下記非特許文献1参照)。

【先行技術文献】

30

【非特許文献】

【0004】

【非特許文献1】 Shinichi Yamagiwa, Leonel Sousa, "Caravela: A Novel Stream-Based Distributed Computing Environment," Computer, vol.40, no.5, pp.70-77, May 2007.

【発明の概要】

【発明が解決しようとする課題】

【0005】

ところで、アクセラレータ側プログラムを作成する場合には、CPUとアクセラレータとはアーキテクチャが異なるため、これまでのフォンノイマン型のCPU側プログラムのプログラムコードと、ストリームコンピューティングに従ったアクセラレータ側プログラムのプログラムコードを別々に作成し、異なるコンパイラによってバイナリを2種類作成する必要がある。即ち、CPU側で実行させるバイナリとアクセラレータ側で実行させるバイナリとを作成する必要がある。ここで、CPU側プログラムには、CPUとアクセラレータとの間でのデータの受け渡し等に関する詳細な手続きを記述しなければならない。

40

【0006】

上記非特許文献1に示したプラットフォームでは、入力データストリームと定数パラメータとを入力して、プログラム (カーネルプログラム) を実行し、出力データストリームを得る、といった流れを規定した flow-model を定義することで、アクセラレータを用いるアプリケーションを開発するプログラムのプログラム作成にかかる負担を軽減している。しかしながら、このプラットフォームを用いたとしても、データ入出力 (CP

50

Uとアクセラレータとの間でのデータの受け渡し)等について詳細に記述したCPU側プログラムが、やはり必要となる。

【0007】

以上述べたように、アクセラレータを用いたアプリケーションを開発するプログラマは、アプリケーション開発の際に、アプリケーションの本来の計算処理(アクセラレータに実行させたい計算処理)とは直接的には関係のないCPU側プログラム(CPUとアクセラレータとの間でのデータの受け渡しの制御等を実行するプログラム)を作成する必要がある。このことは、アクセラレータを用いるアプリケーションを開発するプログラマのプログラム生産性を低下させる一因となっている。

【0008】

本発明は、上記の課題に鑑みてなされたものであり、アクセラレータを用いるアプリケーションを開発するプログラマのプログラム生産性を向上させることができるアクセラレータ処理実行装置、及びアクセラレータ処理実行プログラムを提供することを目的とする。

【課題を解決するための手段】

【0009】

本発明に係るアクセラレータ処理実行装置は、CPUと、CPUに接続され、データを記憶する第1の記憶手段と、CPUに接続され、第1の記憶手段とは異なる、データを記憶する第2の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備え、CPUは、アクセラレータに関する処理の実行を制御する実行制御手段を有し、実行制御手段は、アクセラレータプログラムと、アクセラレータプログラムが実行されて出力される出力データ毎に第2の記憶手段に確保される出力バッファと第1の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、アクセラレータに関する処理の手順を決定するために実行形式とは別に予め定められている規則と実行情報とに基づいて決定した処理手順を実行する処理手順実行手段と、を有し、処理手順実行手段は、アクセラレータにアクセラレータプログラムを送信するプログラム送信処理と、第2の記憶手段に出力バッファを確保する出力バッファ確保処理と、プログラム送信処理及び出力バッファ確保処理よりも後に、アクセラレータにアクセラレータプログラムを実行させるプログラム実行処理と、プログラム実行処理により出力バッファに出力された出力データを、当該出力バッファに対応する第1の記憶手段における出力先に送信する出力データ送信処理と、を実行する。

【0010】

上記アクセラレータ処理実行装置では、実行制御手段が、アクセラレータプログラムと、出力データのアクセラレータ側での保存先(出力バッファ)とCPU側での保存先(第1の記憶手段における出力先)とを対応付ける出力データ対応情報を含む実行情報を所定の形式(例えばXML形式等)で記述した実行形式を入力する。そして、CPUが備える実行制御手段は、実行形式及び予め定めた規則に基づいてアクセラレータに関する処理手順を決定し、実行する。これにより、CPU側とアクセラレータ側との間の出力データの受け渡しを含む各処理手順が、実行制御手段によって適切に決定及び実行されるため、アクセラレータを用いるアプリケーションを開発するプログラマ(以下、単に「プログラマ」と表記する)は、CPU側とアクセラレータ側との間のデータの受け渡し等の処理手順を詳細に規定したプログラム(CPU側プログラム)を作成する手間から解放される。即ち、上記アクセラレータ処理実行装置によれば、アクセラレータプログラムをアクセラレータに実行させるためには、上記実行形式を用意するだけでよいため、プログラマのプログラム生産性を向上させることができる。

【0011】

また、上記アクセラレータ処理実行装置では、実行情報には、アクセラレータプログラムに対する入力データ毎に第1の記憶手段における記憶場所と第2の記憶手段に確保される入力バッファとを対応付ける入力データ対応情報が含まれ、処理手順実行手段は、プロ

10

20

30

40

50

グラム実行処理よりも前に、第2の記憶手段に入力バッファを確保する入力バッファ確保処理と、入力バッファ確保処理よりも後に、上記記憶場所に記憶されている入力データを、当該記憶場所に対応する入力バッファに送信する入力データ送信処理と、を更に実行し、処理手順実行手段は、入力データ送信処理よりも後にプログラム実行処理を実行してもよい。

【0012】

上記アクセラレータ処理実行装置によれば、アクセラレータプログラムの実行に入力データが必要とされる場合であっても、入力データ対応情報を含む実行形式を入力した実行制御手段によって、CPU側とアクセラレータ側との間の入力データの受け渡しに関する処理が適切に決定及び実行される。従って、プログラマは、入力データ対応情報を含む実行形式を用意するだけでよく、CPU側プログラムを作成する手間から解放される。

10

【0013】

また、上記アクセラレータ処理実行装置では、入力データ対応情報に含まれる第1の記憶手段における記憶場所、又は出力データ対応情報に含まれる第1の記憶手段における出力先として、メモリ上に確保された共有領域が指定されてもよい。

【0014】

上記アクセラレータ処理実行装置によれば、アクセラレータプログラムの入力データ又は出力データの受け渡しを、ファイル（例えばCSVファイル等）を介することなく、仮想バッファ（メモリ上に確保された共有領域）を介して実行することができ、データの受け渡しにおいてファイルI/Oを排除することができる。例えば、複数のアクセラレータプログラムを連続して実行させ、先に実行させたアクセラレータプログラムにより出力された出力データを、後に実行させるアクセラレータプログラムの入力データとして用いたい場合等がある。この場合、先に実行させるアクセラレータプログラムを格納した実行形式に記述する出力データの出力先と、後に実行させるアクセラレータプログラムを格納した実行形式に記述する入力データの記憶場所とを、同一の仮想バッファに指定すれば、2つのアクセラレータプログラム間でのデータの受け渡しは、仮想バッファを介して実現されることとなる。これにより、データの受け渡しにおいてファイルI/Oを排除して、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。

20

【0015】

また、上記アクセラレータ処理実行装置では、実行情報には、入力バッファ及び出力バッファの組み合わせと入替回数とを示すスワップ情報が含まれ、処理手順実行手段は、プログラム実行処理において、アクセラレータプログラムの1回の実行が完了する毎に入力バッファと出力バッファとを上記組み合わせに基づいて入れ替えた後にアクセラレータプログラムを実行する処理を、入替回数だけ繰り返してもよい。

30

【0016】

例えば、高速フーリエ変換（FFT）のアルゴリズム（いわゆるバタフライ演算）等、アクセラレータプログラムの出力データを入力データとして同一のアクセラレータプログラムを再実行する処理を繰り返したいような場合がある。この場合、上記アクセラレータ処理実行装置によれば、アクセラレータプログラムの1回の実行が完了する毎に、例えばポインタの入替等により入力バッファと出力バッファとを入れ替えてアクセラレータプログラムを再実行することができるため、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。具体的には、出力バッファに出力されたアクセラレータプログラムの出力データを一旦CPU側の記憶領域（第1の記憶手段）に移動させてから再度アクセラレータ側に入力データとして入力するといった無駄なデータ受け渡し処理を省略することができる。

40

【0017】

ところで、本発明は、上記のようにアクセラレータ処理実行装置の発明として記述できる他に、以下のようにアクセラレータ処理実行プログラムの発明としても記述することができる。これはカテゴリが異なるだけで、実質的に同一の発明であり、同様の作用及び効

50

果を奏する。

【0018】

即ち、本発明に係るアクセラレータ処理実行プログラムは、CPUと、CPUに接続され、データを記憶する第1の記憶手段と、CPUに接続され、第1の記憶手段とは異なる、データを記憶する第2の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備えるコンピュータに、アクセラレータに関する処理の実行を制御する実行制御手段を実現させるためのアクセラレータ処理実行プログラムであって、実行制御手段は、アクセラレータプログラムと、アクセラレータプログラムが実行されて出力される出力データ毎に第2の記憶手段に確保される出力バッファと第1の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、アクセラレータに関する処理の手順を決定するために実行形式とは別に予め定められている規則と実行情報とに基づいて決定される処理手順を実行する処理手順実行手段と、を有し、処理手順実行手段は、アクセラレータにアクセラレータプログラムを送信するプログラム送信処理と、第2の記憶手段に出力バッファを確保する出力バッファ確保処理と、プログラム送信処理及び出力バッファ確保処理よりも後に、アクセラレータにアクセラレータプログラムを実行させるプログラム実行処理と、プログラム実行処理により出力バッファに出力された出力データを、当該出力バッファに対応する第1の記憶手段における出力先に送信する出力データ送信処理と、を実行する。

10

【発明の効果】

【0019】

本発明によれば、アクセラレータを用いるアプリケーションを開発するプログラムのプログラム生産性を向上させることができる。

20

【図面の簡単な説明】

【0020】

【図1】本発明の一実施形態に係るアクセラレータ処理実行装置のシステム構成を示すブロック図である。

【図2】flow-modelを概念的に示した図である。

【図3】モデルファイルの例(XMLファイル)を示す図である。

【図4】実行形式ファイルの例(XMLファイル)を示す図である。

【図5】実行制御部の位置付けを概念的に示した図である。

30

【図6】スワップ処理を含まない場合におけるアクセラレータ処理実行装置の動作の例を示すフロー図である。

【図7】スワップ処理を含む場合におけるアクセラレータ処理実行装置の動作の例を示すフロー図である。

【図8】(a)実行形式ファイルの実行形態の例を示す図及び(b)実行形式ファイルの実行パターンの例を示す図である。

【図9】バッチファイルの例を示す図である。

【図10】モデルファイル、実行形式ファイル、及びバッチファイルの包含関係を概念的に示す図である。

【図11】仮想バッファを用いたデータ受け渡しについて説明するための図である。

40

【図12】アクセラレータ処理実行装置による計算処理及びバッチファイルの応用例を示す図である。

【図13】図12に示した応用例における実行形式ファイル及びモデルファイルを示す図である。

【図14】本発明の一実施形態に係るアクセラレータ処理実行プログラムの機能構成を示すブロック図である。

【図15】従来のCPU側プログラムの例を示す図である。

【発明を実施するための形態】

【0021】

以下、添付図面を参照しながら本発明の実施形態を詳細に説明する。なお、図面の説明

50

において同一又は同等の要素には同一の符号を付し、重複する説明を省略する。

【0022】

図1は、本発明の一実施形態に係るアクセラレータ処理実行装置のシステム構成を示すブロック図である。本実施形態に係るアクセラレータ処理実行装置1は、CPU(Central Processing Unit: 中央処理装置)10に接続されたアクセラレータ30による所定の計算処理の実行を制御する機構を備えた装置として構成されている。具体的には、図1に示すように、アクセラレータ処理実行装置1は、CPU10、記憶部20、及びアクセラレータ30を備えている。CPU10は、実行制御部40を備えている。

【0023】

CPU10は、オペレーティングシステム及びアプリケーションプログラム等を実行するハードウェアである。図1に示すように、CPU10は、例えばCPUのローカルバス(CPUバス)でCPU側メモリ21と接続されている。また、CPUバスはバスブリッジ50を介して周辺バス(例えば「PCI Expressバス」等)に拡張され、周辺バスはアクセラレータ30に接続されている。すなわち、CPU10とアクセラレータ30とは、CPUバス、バスブリッジ50、及び周辺バスを介して接続されている。また、補助記憶部22とバスブリッジ50とは、SATA等のストレージ向けインターコネクで周辺バスからさらにブリッジ(不図示)を介して接続されている。このような接続構成により、CPU10は、記憶部20(CPU側メモリ21、補助記憶部22)及びアクセラレータ30の双方と互いにデータ通信可能とされている。一方、アクセラレータ30が記憶部20に直接アクセスする構成とすると、ブリッジによってオーバーヘッドが大きくなり、アクセラレータ30が記憶部20(補助記憶部22)を常に監視することによって性能低下を招いてしまう。そのため、通常、アクセラレータ30に記憶部20を直接認識させることはせず、記憶部20とアクセラレータ30との間におけるデータの受け渡しは、CPU10を介して行われる。

【0024】

記憶部20は、データを記憶する記憶手段(第1の記憶手段)であり、上述のようにCPU10に接続されている。記憶部20は、ROM(Read Only Memory)及びRAM(Random Access Memory)等で構成されるCPU側メモリ21とハードディスクメモリ等で構成される補助記憶部22とを有している。CPU側メモリ21には、共有領域(共有メモリ)である仮想バッファ21aと、規則情報記憶部21b(詳細は後述)とを有する。仮想バッファ21aは、例えば「POSIX Shared Memory」等の共有メモリライブラリにより実現される。また、本実施形態では、補助記憶部22には、入力データファイル22a、出力データファイル22b、モデルファイル22c、実行形式ファイル(実行形式)22d、及びバッチファイル22e(これらのファイルの詳細は後述)が記憶されるものとする。ここで、モデルファイル22c、実行形式ファイル22d、及びバッチファイル22eは、アクセラレータを用いるアプリケーションを開発するプログラマ(以下、単に「プログラマ」と表記する)によって作成又は用意されるものである。

【0025】

アクセラレータ30は、上述のようにCPU10に接続され、CPU10の計算処理の一部をCPU10に代わって実行するハードウェアである。具体的には、アクセラレータ30は、アクセラレータ30に所定の計算処理を実行させるためのプログラム(「カーネルプログラム」とも呼ばれる。以下「アクセラレータプログラム」と表記する。)を実行するプロセッサ31と、データを記憶するアクセラレータ側メモリ(第2の記憶手段)32とを有している。アクセラレータ側メモリ32には、アクセラレータプログラムの入力データ及び出力データ等が格納される。以降の説明において、「入力データ」及び「出力データ」と単に表記した場合には、これらはそれぞれ、「アクセラレータプログラムに対する入力データ」及び「アクセラレータプログラムが実行されて出力される出力データ」を指すものとする。

【0026】

アクセラレータ30は、例えばメニーコア方式(ストリームコンピューティングとも呼

10

20

30

40

50

ばれる)と呼ばれる計算方式によりアクセラレータプログラムの並列実行を実現する。この方式では、アクセラレータ30は、複数のプロセッサ31を有し、各プロセッサ31に異なるID(プロセッサID)を割り振り、プロセッサID毎に独立した計算処理をさせることで、計算処理の並列実行を実現する。例えば、ベクトルA、Bの加算結果をベクトルCとして保存する処理( $C = A + B$ )であれば、各プロセッサ31は、自身のプロセッサIDに対応するIDについて、 $C[ID] = A[ID] + B[ID]$ という計算を実行する。ここで $X[ID]$ ( $X = A, B, C$ )は、ベクトルXのID番目の要素を示している。このように、メニーコア方式のアクセラレータ30によれば、各ベクトル要素(各IDに対応する要素)のベクトル計算が複数のプロセッサ31により並列的に計算される。

【0027】

このようなメニーコア方式のアクセラレータ30としては、例えばGPU(Graphics Processing Unit)及びFPGA(Field-Programmable Gate Array)等を実現されたメニーコアフレームワーク等がある。

【0028】

ここで、上述したように、アクセラレータ30には記憶部20を認識させないため、アクセラレータ30にアクセラレータプログラムを実行させるためには、CPU10が、記憶部20からアクセラレータ30に必要なプログラム及びデータを受け渡したり、アクセラレータ30の処理結果を記憶部20に記憶させたりといったデータの受け渡し等の実行を制御する必要がある。具体的には、CPU10が以下のような処理手順を実行する必要がある。

【0029】

即ち、CPU10は、例えば補助記憶部22等に予め記憶したアクセラレータプログラムを、アクセラレータ30上で実行可能な形式のバイナリにコンパイルした上でCPU側メモリ21を介してアクセラレータ側メモリ32にダウンロード(送信)し、当該バイナリをプロセッサ31で実行可能な状態にする。また、アクセラレータプログラムが入力データを必要とする場合には、CPU10は、入力データを記憶する領域である入力バッファ32aを入力データ毎にアクセラレータ側メモリ32上に確保する。ここで、CPU10は、入力バッファ32aに対して、当該入力バッファ32aがどの入力データに対応するバッファであるかを特定する情報を付与する。そして、CPU10は、例えば補助記憶部22に予め記憶したCSVファイル等の入力データファイル22aに含まれる入力データを、当該入力データに対応付けて用意した入力バッファ32aにCPU側メモリ21を介してコピーする。また、CPU10は、アクセラレータプログラムによる処理結果(出力データ)を記憶する領域である出力バッファ32bを出力データ毎にアクセラレータ側メモリ32上に確保する。ここで、CPU10は、出力バッファ32bに対して、当該出力バッファ32bがどの出力データに対応するバッファであるかを特定する情報を付与する。

【0030】

続いて、CPU10は、アクセラレータ30に対してアクセラレータプログラムの実行を指示する。そして、アクセラレータ30によるアクセラレータプログラムの実行が完了し、出力バッファ32bに出力データが出力された後に、CPU10は、当該出力データをCPU側メモリ21に読み出す。その後、CPU10は、CPU側メモリ21に読み出された出力データを、例えば補助記憶部22の出力データファイル22bに書き出す。

【0031】

従来、CPUに上述した処理手順を実行させるために、プログラマは、上述した処理手順を詳細に規定したプログラム(以下「CPU側プログラム」と表記する)と、アクセラレータプログラムとの両方を作成している。

【0032】

図15に、従来プログラマが作成しているプログラムの例を示す。図15に示すプログラムは、GPUを用いるプログラムを作成するためのプログラミングツールであるOpenCLを用いて記述されたプログラムである。同図上段の破線で囲まれた記述D1がCP

10

20

30

40

50

U側プログラムに対応する記述であり、同図下段の破線で囲まれた記述D2がGPUのプログラム(アクセラレータプログラム)に対応する記述である。同図に示すように、CPU側プログラムとして、GPUのプログラムをコンパイルする処理、GPUのメモリ上に入出力用バッファを用意する処理、入力データを用意する処理、GPUのプログラムをGPUに実行させる処理、及び、GPUのプログラムの処理結果をGPUのメモリから読み出す処理等がプログラマによって詳細に記述される。

【0033】

実行制御部40は、従来プログラマが作成していたCPU側プログラムを不要として、アクセラレータ30に関する処理の実行を制御する実行制御手段である。具体的には、実行制御部40は、アクセラレータプログラムと、アクセラレータプログラムの入出力データの対応付けの定義を含むデータ構造に基づいて、従来プログラマが作成していたCPU側プログラムにより規定されていた処理に対応する処理を決定及び実行する。これにより、プログラマは、アクセラレータプログラムと、当該アクセラレータプログラムの入出力データの対応付けの定義を作成するだけでよくなり、従来のCPU側プログラムを作成する必要がなくなる。

10

【0034】

例えば後述するアクセラレータ処理実行プログラムP1がCPU10に読み込まれることで、CPU10上において実行制御部40の機能が実現される。本実施形態では、実行制御部40は、アクセラレータ処理実行装置1上で、例えばシェルプログラムのように動作するものとして説明する。即ち、実行制御部40は、アクセラレータ処理実行装置1が備えるキーボード等の入力装置(不図示)を介してコマンドラインでのユーザ入力等によって入力された所定のコマンドを受け付け、当該コマンドに対応して予め記憶された処理を実行する。コマンドの詳細については後述する。実行制御部40は、実行形式入力部41、及び処理手順実行部42を有している。

20

【0035】

実行形式入力部41は、アクセラレータ30に関する処理の実行制御に必要な情報(実行情報)をXML形式(所定の形式)で記述した実行形式ファイル(実行形式)22dを入力する実行形式入力手段である。実行形式入力部41は、例えばコマンドラインでのユーザ入力等により、補助記憶部22に記憶された実行形式ファイル22dを入力する。実行形式ファイル22dには、モデルファイル22cを特定するモデルファイル特定情報と、入力データに関する情報と、出力データに関する情報とが含まれている。さらに、モデルファイル22cには、後述するflow-modelの情報が含まれている。具体的には、モデルファイル22cには、flow-modelの情報として、アクセラレータプログラムに関する情報と、入力バッファ32aに関する情報と、出力バッファ32bに関する情報とが含まれている。ここで、実行情報とは、実行形式ファイル22d及びモデルファイル22cに含まれる情報、並びにこれらの情報を組み合わせて得られる情報を含む全ての情報を指す。

30

【0036】

図2~図4を用いて、モデルファイル22c、実行形式ファイル22d、及びこれらのファイルから得られる実行情報について詳細に説明する。

40

【0037】

まず、図2及び図3を用いて、アクセラレータプログラムを実行するための標準化モデルとして本発明者らが開発したflow-modelと、このflow-modelの情報を格納したモデルファイル22cについて説明する。図2は、flow-modelを概念的に示した図である。図2に示すように、flow-modelは、アクセラレータプログラム自体と、当該アクセラレータプログラムに対する入力データ(入力データストリーム及び定数)及び出力データ(出力データストリーム)とからなるデータ構造として定義される。

【0038】

図3は、モデルファイル22cの例を示す図である。図3に示すように、モデルファイ

50

ル 2 2 c は、XML 形式のファイルとして実現される。

【 0 0 3 9 】

モデルファイル 2 2 c には、アクセラレータプログラムに関する情報として、プログラムコード (Kernel) と、アクセラレータプログラムの言語情報 (LangType) と、アクセラレータプログラムを解釈して実行するプラットフォーム (開発環境及び実行ランタイム等を含むシステム) である OpenCL 及び CUDA 等のランタイムタイプを示す情報 (RuntimeType) と、アクセラレータプログラムの実行における並列度を示すスレッド数及びブロック数等の情報 (TotalNumThreads、TotalNumBlocks) とが含まれている。

【 0 0 4 0 】

また、入力バッファ 3 2 a に関する情報 (Input) として、入力データの型 (DataType) と、入力データの個数 (Length) と、入力バッファ 3 2 a とアクセラレータプログラムにおいて定義される入力引数 (入力変数) との対応を示す情報とが含まれている。ここで、入力データの型 (DataType) は、整数型 (INT) や実数型 (FLOAT) 等を示す情報である。入力データの個数 (Length) は、入力データの大きさを示す情報であり、具体的には、入力データの型 (DataType) で示される個々のデータの個数である。入力バッファ 3 2 a と入力引数との対応を示す情報は、入力バッファ 3 2 a を特定する情報 (Index) と入力引数名 (Name) との組を示す情報である。図 3 の例では、Index が「0」の入力バッファ 3 2 a が、入力引数「a」に対応する入力バッファ 3 2 a として対応付けられており、Index が「1」の入力バッファ 3 2 a が、入力引数「b」に対応する入力バッファ 3 2 a として対応付けられている。また、入力引数「a」には、1024 個の整数型 (INT) の入力データ (データストリーム) が対応付けられている。

【 0 0 4 1 】

また、出力バッファ 3 2 b に関する情報 (Output) として、出力データの個数 (Length) と、出力バッファ 3 2 b とアクセラレータプログラムにおいて定義される出力引数 (出力変数) との対応を示す情報と、出力データの型 (DataType) とが含まれている。ここで、出力バッファ 3 2 b と出力引数との対応を示す情報は、出力バッファ 3 2 b を特定する情報 (Index) と出力引数名 (Name) との組を示す情報である。図 3 の例では、Index が「0」の出力バッファ 3 2 b が、出力引数「c」に対応する出力バッファ 3 2 b として対応付けられている。

【 0 0 4 2 】

なお、アクセラレータプログラムが入力データを必要としない場合、即ちアクセラレータプログラムにおいて入力引数が定義されない場合には、入力バッファ 3 2 a に関する情報 (Input) は省略される。また、入力バッファ 3 2 a に関する情報は、アクセラレータプログラムにおいて定義された入力引数の数、即ちアクセラレータプログラムへの入力データの数だけ定義される。同様に、出力バッファ 3 2 b に関する情報は、アクセラレータプログラムにおいて定義された出力引数の数、即ちアクセラレータプログラムからの出力データの数だけ定義される。

【 0 0 4 3 】

このように、モデルファイル 2 2 c には、アクセラレータプログラムの入力引数毎に対応する入力バッファ 3 2 a と、出力引数毎に対応する出力バッファ 3 2 b とに関する情報とが含まれる。しかし、モデルファイル 2 2 c (即ち flow-model として定義されるデータ構造) には、入力バッファ 3 2 a に用意される具体的な入力データの記憶場所 (例えば入力データファイル 2 2 a 及び仮想バッファ 2 1 a 等)、及び出力バッファ 3 2 b に出力された出力データの CPU 10 側における具体的な出力先 (例えば出力データファイル 2 2 b 及び仮想バッファ 2 1 a 等) に関する情報が含まれていない。即ち、モデルファイル 2 2 c には、アクセラレータ 3 0 に計算処理を実行させるために必要となる具体的な入力データ及び出力データに関する情報が含まれていない。そこで、これらの入力データ及び出力データに関する情報を flow-model に関連付けたデータ構造を記述

10

20

30

40

50

したものが、上述の実行形式ファイル 2 2 d である。

【 0 0 4 4 】

図 4 は、実行形式ファイル 2 2 d の例を示す図である。図 4 に示すように、実行形式ファイル 2 2 d は、モデルファイル 2 2 c と同様に、XML 形式のファイルとして実現される。

【 0 0 4 5 】

実行形式ファイル 2 2 d には、モデルファイル特定情報として、アクセラレータ 3 0 に実行させるアクセラレータプログラムを含むモデルファイル 2 2 c の場所（例えばファイル名等）を示す情報（Model File）が含まれている。

【 0 0 4 6 】

また、入力データに関する情報（Input）として、モデルファイル 2 2 c に含まれるアクセラレータプログラムの入力引数名（Name）と、入力データの記憶場所（例えば入力データファイル 2 2 a 及び仮想バッファ 2 1 a 等）を示す情報（Data File）とが含まれている。図 4 の例では、ファイル名が「sample1\_\_a.csv」の入力データファイル 2 2 a が、入力引数「a」に対応する入力データとして対応付けられ、ファイル名が「sample1\_\_b.csv」の入力データファイル 2 2 a が、入力引数「b」に対応する入力データの記憶場所として対応付けられている。ここで、モデルファイル 2 2 c に含まれるアクセラレータプログラムが入力データを必要としない場合、即ちアクセラレータプログラムにおいて入力引数が定義されない場合には、入力データに関する情報（Input）は省略されてもよい。

【 0 0 4 7 】

また、出力データに関する情報（Output）として、モデルファイル 2 2 c に含まれるアクセラレータプログラムの出力引数名（Name）と、出力データの出力先（例えば出力データファイル 2 2 b 及び仮想バッファ 2 1 a 等）を示す情報（Data File）とが含まれている。図 4 の例では、ファイル名「sample1\_\_c.csv」で示されるファイルが、出力引数「c」に対応する出力データの出力先として対応付けられている。

【 0 0 4 8 】

また、実行形式ファイル 2 2 d には、後述するスワップ機能に関するスワップ情報（Swap Pair）として、スワップペアの入力引数名（Input）と、スワップペアの出力引数名（Output）と、スワップ回数（入替回数、Num Swap）とが含まれている。ただし、スワップ機能の実行が不要な場合には、スワップ情報（Swap Pair）は省略されてもよい。

【 0 0 4 9 】

実行形式入力部 4 1 は、実行形式ファイル 2 2 d を入力することにより、当該実行形式ファイル 2 2 d に直接記述されている情報（入力データに関する情報、及び出力データに関する情報）とともに、当該実行形式ファイル 2 2 d に含まれるモデルファイル特定情報（Model File）に示されるモデルファイル 2 2 c に記述されている情報（アクセラレータプログラムに関する情報、入力バッファ 3 2 a に関する情報、及び出力バッファ 3 2 b に関する情報）を実行情報として取得する。

【 0 0 5 0 】

さらに、実行形式入力部 4 1 は、実行形式ファイル 2 2 d における入力データの記憶場所を示す情報（Data File）と、モデルファイル 2 2 c における入力バッファ 3 2 a を特定する情報（Index）とを、同一の入力引数（Name）で対応付けることにより、入力データの記憶場所と入力バッファ 3 2 a との対応付けを把握する。即ち、実行形式入力部 4 1 は、アクセラレータプログラムへの入力データ毎に、記憶部 2 0 における記憶場所とアクセラレータ側メモリ 3 2 に確保される入力バッファ 3 2 a とを対応付ける情報（入力データ対応情報）を実行情報として取得する。

【 0 0 5 1 】

図 3 及び図 4 の例では、実行形式入力部 4 1 は、入力引数「a」での対応付けを行うこ

10

20

30

40

50

とにより、ファイル名が「sample1\_\_a.csv」の入力データファイル22aとIndexが「0」の入力バッファ32aとの対応付けを把握する。同様に、実行形式入力部41は、入力引数「b」での対応付けを行うことにより、ファイル名が「sample1\_\_b.csv」の入力データファイル22aとIndexが「1」の入力バッファ32aとの対応付けを把握する。

#### 【0052】

また、実行形式入力部41は、実行形式ファイル22dにおける出力データの出力先を示す情報(DataFile)と、モデルファイル22cにおける出力バッファ32bを特定する情報(Index)とを、同一の出力引数(Name)で対応付けることにより、出力データの出力先と出力バッファ32bとの対応付けを把握する。即ち、実行形式入力部41は、アクセラレータプログラムからの出力データ毎に、アクセラレータ側メモリ32に確保される出力バッファ32bと記憶部20における出力先とを対応付ける情報(出力データ対応情報)を実行情報として取得する。

10

#### 【0053】

図3及び図4の例では、実行形式入力部41は、出力引数「c」での対応付けを行うことにより、ファイル名が「sample1\_\_c.csv」の出力先ファイルとIndexが「0」の出力バッファ32bとの対応付けを把握する。

#### 【0054】

実行形式入力部41は、上述のように取得した実行情報を処理手順実行部42に出力する。

20

#### 【0055】

処理手順実行部42は、アクセラレータ30に関する処理の手順を決定するために実行形式ファイル22dとは別に予め定められている規則と、実行形式入力部41から取得した実行情報とに基づいて決定した処理手順を実行する処理手順実行手段である。具体的には、処理手順実行部42は、例えば上述の規則を示す規則情報が予め記憶されたCPU側メモリ21上の規則情報記憶部21bを参照することで、処理手順を決定する。規則情報記憶部21bは、例えば、アクセラレータ処理実行プログラムP1のプログラムコード自体や、当該プログラムから参照可能なファイル等により実現される。この場合、規則情報は、アクセラレータ処理実行プログラムP1のプログラムコードにハードコーディングされた情報や、当該プログラムから参照可能なファイル等に記述(記憶)された情報等である。

30

#### 【0056】

規則情報は、どの実行情報を用いてどのような処理手順を決定するかを予め定めた規則を示す情報であり、後述するアクセラレータ処理実行プログラムP1を作成する管理者等によって作成されるものである。規則情報には、例えば、実行形式入力部41により入力された実行形式ファイル22dあるいはモデルファイル22cに記述されたタグ要素の読取方法、及び、読み取った各タグ要素の情報に基づいてどのような処理手順をどのような順序(又はタイミング)で実行するかを示す情報等が含まれる。

#### 【0057】

処理手順実行部42は、具体的には、上述した実行情報及び規則情報に基づいて、プログラム送信処理、入力バッファ確保処理、入力データ送信処理、出力バッファ確保処理、プログラム実行処理、及び出力データ送信処理を実行する。以下、各処理について詳細に説明する。

40

#### 【0058】

プログラム送信処理は、アクセラレータ30にアクセラレータプログラムを送信する処理である。具体的には、処理手順実行部42は、実行形式入力部41によって読み込まれたモデルファイル22cに含まれる言語情報(Language)及びランタイムタイプを示す情報(RuntimeType)に基づいて該当するコンパイラを特定する。このようなコンパイラの特定は、例えば言語情報及びランタイムタイプの組に対して使用されるコンパイラを関連付けた情報を規則情報として規則情報記憶部21bに格納しておくこ

50

とで実現される。続いて、処理手順実行部42は、特定したコンパイラで当該モデルファイル22cに含まれるプログラムコード(Kernel)をコンパイルしてバイナリ(バイナリカーネル)を取得する。続いて、処理手順実行部42は、取得したバイナリをアクセラレータ30にダウンロード(送信)し、プロセッサ31で当該バイナリを実行可能な状態にする。

**【0059】**

入力バッファ確保処理は、プログラム実行処理よりも前に、アクセラレータ側メモリ32に入力バッファ32aを確保する処理である。具体的には、処理手順実行部42は、モデルファイル22cの入力バッファに関する情報(Input)において指定された入力データの個数(Length)及びデータ型(Data Type)に基づいて必要な領域の大きさを計算し、当該大きさの入力バッファ32aを、識別情報(Index)を付与した上でアクセラレータ側メモリ32上に確保する。入力バッファ32aとして必要な領域の大きさを計算は、例えば、入力データの型(Data Type)毎のデータ1つあたりに必要な領域の大きさ、及び当該大きさと入力データの個数(Length)とを掛け合わせるという計算規則を規則情報として規則情報記憶部21bに格納しておくことで実現できる。

**【0060】**

また、処理手順実行部42は、入力バッファ32aに対応する入力データの記憶場所がファイル(入力データファイル22a)であるか否かを判定する。例えば、処理手順実行部42は、入力バッファ32aに付与されたIndexに対応付けられる入力データの記憶場所を入力データ対応情報に基づいて特定し、特定された入力データの記憶場所がファイル名を示すものか否かを判定する。このような判定は、例えば、ファイルであることを示す特定の文字列を、規則情報として規則情報記憶部21bに格納しておき、実行形式ファイル22dに含まれる入力データの記憶場所(InputのData File)にファイル名を示す特定の文字列(例えば「.csv」等)が含まれているか否かを判定することにより行うことができる。このようにして、入力バッファ32aに対応する入力データの記憶場所がファイル(入力データファイル22a)であると判定された場合には、処理手順実行部42は、入力バッファ32aとして確保したサイズと同じサイズのバッファ領域をCPU側メモリ21上にも確保する。

**【0061】**

入力データ送信処理は、入力バッファ確保処理よりも後に、記憶部20における記憶場所に記憶されている入力データを、当該記憶場所に対応する入力バッファ32aに送信する処理である。具体的には、処理手順実行部42は、上述のように、入力データ対応情報に基づいて、入力バッファ32aに対応する入力データの記憶場所が入力データファイル22aであるか仮想バッファ21aであるかを判定する。入力データの記憶場所が入力データファイル22aであると判定された場合には、処理手順実行部42は、当該入力データファイル22aを開き、当該入力データファイル22aに含まれる入力データを、入力バッファ確保処理においてCPU側メモリ21上に確保したバッファ領域に読み出す。さらに、処理手順実行部42は、このバッファ領域に読み出した入力データを入力バッファ32aにコピーする。一方、入力データの記憶場所が仮想バッファ21aであると判定された場合には、処理手順実行部42は、仮想バッファ21aから入力データを直接読み込み、読み込んだ入力データを入力バッファ32aにコピーする。ここで、どの入力データ(入力データファイル22a又は仮想バッファ21aに含まれる入力データ)をどの入力バッファ32aにコピーするかについては、入力データ対応情報及び入力バッファ32aに付与されたIndexに基づいて特定される。

**【0062】**

処理手順実行部42は、入力バッファ確保処理及び入力データ送信処理を、モデルファイル22cにおいて定義された全ての入力バッファ32aに関して実行する。図3に示した例では、処理手順実行部42は、入力引数「a」に対応する入力バッファ32a及び入力引数「b」に対応する入力バッファ32aのそれぞれについて、上述の入力バッファ確

10

20

30

40

50

保処理及び入力データ送信処理を実行する。

【0063】

出力バッファ確保処理は、アクセラレータ側メモリ32に出力バッファ32bを確保する処理である。具体的には、処理手順実行部42は、モデルファイル22cの出力バッファに関する情報(Output)において指定された出力データの個数(Length)及びデータ型(Data Type)に基づいて必要な領域の大きさを計算し、当該大きさの出力バッファ32bを、識別情報(Index)を付与した上でアクセラレータ側メモリ32上に確保する。出力バッファ32bとして必要な領域の大きさの計算は、例えば上述した入力バッファ確保処理と同様の方法により実現できる。

【0064】

また、処理手順実行部42は、出力バッファ32bに対応する出力データの出力先がファイル(出力データファイル22b)であるか否かを判定する。処理手順実行部42は、出力バッファ32bに付与されたIndexに対応付けられる出力データの出力先を出力データ対応情報に基づいて特定し、特定された出力データの出力先がファイル名を示すものか否かを判定する。例えば、処理手順実行部42は、出力データの出力先と同名の仮想バッファ21aがCPU側メモリ21上に存在するか否かを判定し、仮想バッファ21aがCPU側メモリ21上に存在しなければ、出力データの出力先がファイル名を示すものと判定することができる。また、このような判定は、例えば、ファイルであることを示す特定の文字列を、規則情報として規則情報記憶部21bに格納しておき、実行形式ファイル22dに含まれる出力データの出力先(OutputのData File)にファイル名を示す特定の文字列(例えば「.csv」等)が含まれているか否かを判定することにより行うこともできる。このようにして、出力バッファ32bに対応する出力データの出力先がファイル(出力データファイル22b)であると判定された場合には、処理手順実行部42は、出力バッファ32bとして確保したサイズと同じサイズのバッファ領域をCPU側メモリ21上にも確保する。

【0065】

図3に示した例では、出力バッファの定義は、引数「c」に対応して定義された出力バッファのみであるが、モデルファイル22cにおいて複数の出力バッファが定義されている場合には、処理手順実行部42は、出力バッファ確保処理を、全ての出力バッファ32bに関して実行する。

【0066】

プログラム実行処理は、プログラム送信処理、入力データ送信処理、及び出力バッファ確保処理よりも後に、アクセラレータ30にアクセラレータプログラムを実行させる処理である。処理手順実行部42は、プログラム送信処理において送信したアクセラレータプログラムの実行をアクセラレータ30に指示する。その後、アクセラレータ30によってアクセラレータプログラムが実行され、その計算結果が、出力バッファ確保処理で出力引数毎に確保された出力バッファ32bに出力される。

【0067】

ここで、実行形式ファイル22dにスワップ情報が含まれており、実行形式入力部41によりスワップ情報が読み込まれている場合には、処理手順実行部42は、プログラム実行処理において、アクセラレータプログラムの1回の実行が完了する毎に、スワップ情報(Swap Pair)の入力引数名(Input)に対応する入力バッファ32aとスワップ情報(Swap Pair)の出力引数名(Output)に対応する出力バッファ32bとを入れ替え、アクセラレータプログラムを再実行する。処理手順実行部42は、入力バッファ32aと出力バッファ32bとの入替を、例えばポインタの入替により実際のデータ移動を伴うことなく実行する。処理手順実行部42は、このような入出力バッファの入替及びアクセラレータプログラムの再実行を入替回数(Num Swap)だけ繰り返す。

【0068】

出力データ送信処理は、プログラム実行処理により出力バッファ32bに出力された出

10

20

30

40

50

力データを、当該出力バッファ32bに対応する記憶部20における出力先に送信(コピー)する処理である。具体的には、処理手順実行部42は、上述のように、出力データ対応情報に基づいて、出力バッファ32bに対応する出力データの出力先が出力データファイル22bであるか仮想バッファ21aであるかを判定する。出力データの出力先が出力データファイル22bであると判定された場合には、処理手順実行部42は、出力バッファ32bに格納された出力データを、出力バッファ確保処理においてCPU側メモリ21上に確保したバッファ領域にコピーする。さらに、処理手順実行部42は、このバッファ領域に読み出したデータを出力データファイル22bに書き込む。一方、出力データの出力先が仮想バッファ21aであると判定された場合には、処理手順実行部42は、出力バッファ32bに格納された出力データを仮想バッファ21aに書き込む。

10

**【0069】**

図5を用いて、以上説明した実行制御部40による処理を実現する構成について補足する。図5は、実行制御部40の位置付けを概念的に示した図である。図5に示すように、実行制御部40は、実行形式ファイル22dを入力し、実行形式ファイル22dのモデルファイル特定情報により関連付けられたモデルファイル22cに定義されたアクセラレータプログラムに入力データを与え、アクセラレータ30に計算処理を実行させ、出力データを得る処理を実現している。実行制御部40は、このような処理を実現するために、GPU及びFPGA等を実現されたメニーコアフレームワーク等のアクセラレータ30を制御するソフトウェア(アクセラレータ用ドライバ)、並びにアクセラレータに応じたコンパイラ及びライブラリ等を備えた各種の実行ランタイムシステムを利用している。ここで

20

**【0070】**

ここで、Caravelaプラットフォームは、下位層に位置付けられるOpenCL及びCUDA等のプラットフォームを利用して、flow-modelをアクセラレータ30にマップする(flow-modelで定義されたアクセラレータプログラムをアクセラレータ30上で実行可能にする)処理等を実行するライブラリ(Caravelaライブラリ)を備えたプラットフォームである。

**【0071】**

次に、図6及び図7を用いて、処理手順実行部42により実行される処理の動作について説明する。なお、各処理の詳細については、既に説明しているため、適宜省略する。

30

**【0072】**

まず、図6を用いて、スワップ処理を含まない場合におけるアクセラレータ処理実行装置の動作について説明する。

**【0073】**

まず、実行形式ファイル22dが、実行形式入力部41に入力される(ステップS101)。続いて、実行形式入力部41は、実行形式ファイル22dのモデルファイル特定情報により関連付けられるモデルファイル22cを特定し、当該モデルファイル22cを入力する(ステップS102)。続いて、実行形式入力部41は、実行形式ファイル22d及びモデルファイル22cに基づいて、上述した実行情報を取得する(ステップS103)

40

**【0074】**

続いて、処理手順実行部42は、実行形式入力部41によって取得された実行情報と規則情報記憶部21bに記憶された規則情報とを入力する(ステップS104)。そして、処理手順実行部42は、上述したように、実行情報及び規則情報に基づいて処理手順(各処理の内容、処理順序、及び処理実行のタイミング等)を決定し、決定した処理手順(ステップS105~S111)を実行する。

**【0075】**

処理手順実行部42は、プログラム送信処理において、モデルファイル22cに含まれるアクセラレータプログラムをコンパイルした上でアクセラレータ30にダウンロードし

50

、実行可能な状態にする（ステップS 1 0 5）。

【 0 0 7 6 】

続いて、処理手順実行部 4 2 は、入力バッファ確保処理において、アクセラレータ側メモリ 3 2 上に入力バッファ 3 2 a を確保する（ステップ S 1 0 6）。続いて、処理手順実行部 4 2 は、入力データ送信処理において、記憶部 2 0 における記憶場所に記憶されている入力データを、当該記憶場所に対応する入力バッファ 3 2 a に送信する（ステップ S 1 0 7）。ここで、ステップ S 1 0 6、S 1 0 7 の処理は、処理手順実行部 4 2 により、モデルファイル 2 2 c に定義されている全ての入力バッファ 3 2 a に対して実行される。

【 0 0 7 7 】

続いて、処理手順実行部 4 2 は、出力バッファ確保処理において、アクセラレータ側メモリ 3 2 上に出力バッファ 3 2 b を確保する（ステップ S 1 0 8）。続いて、処理手順実行部 4 2 は、プログラム実行処理において、アクセラレータ 3 0 にアクセラレータプログラムを実行させる（ステップ S 1 0 9）。続いて、処理手順実行部 4 2 は、出力データ送信処理において、アクセラレータプログラムの実行により出力バッファ 3 2 b に出力された出力データを、当該出力バッファ 3 2 b に対応する記憶部 2 0 における出力先に送信する（ステップ S 1 1 0）。

【 0 0 7 8 】

その後、処理手順実行部 4 2 は、後処理として、全ての入力バッファ 3 2 a 及び全ての出力バッファ 3 2 b を解放し、アクセラレータプログラムをアクセラレータ 3 0 上から無効化する処理を実行する（ステップ S 1 1 1）。

【 0 0 7 9 】

次に、図 7 を用いて、スワップ処理を含む場合におけるアクセラレータ処理実行装置の動作について説明する。ここでは、図 6 を用いて説明したスワップ処理を含まない場合の動作と異なる部分を主に説明する。

【 0 0 8 0 】

まず、スワップ情報（S w a p P a i r）が含まれた実行形式ファイル 2 2 d が、実行形式入力部 4 1 に入力される（ステップ S 2 0 1）。ここで、実行形式入力部 4 1 により、スワップ情報に含まれる入力引数名（I n p u t）に対応する入力バッファ 3 2 a と出力引数名（O u t p u t）に対応する出力バッファ 3 2 b との全てのスワップペア（組み合わせ）の情報が読み込まれる。続いて、実行形式入力部 4 1 は、実行形式ファイル 2 2 d のモデルファイル特定情報により関連付けられるモデルファイル 2 2 c を特定し、当該モデルファイル 2 2 c を入力する（ステップ S 2 0 2）。続いて、実行形式入力部 4 1 は、実行形式ファイル 2 2 d 及びモデルファイル 2 2 c に基づいて、上述した実行情報を取得する（ステップ S 2 0 3）。

【 0 0 8 1 】

続いて、処理手順実行部 4 2 は、実行形式入力部 4 1 によって取得された実行情報と規則情報記憶部 2 1 b に記憶された規則情報とを入力する（ステップ S 2 0 4）。そして、処理手順実行部 4 2 は、上述したように、実行情報及び規則情報に基づいて処理手順（各処理の内容、処理順序、及び処理実行のタイミング等）を決定し、決定した処理手順（ステップ S 2 0 5 ~ S 2 1 1）を実行する。

【 0 0 8 2 】

処理手順実行部 4 2 は、ステップ S 1 0 5 ~ S 1 0 8 と同様に、プログラム送信処理、入力バッファ確保処理、入力データ送信処理、及び出力バッファ確保処理を実行する（ステップ S 2 0 5 ~ S 2 0 8）。

【 0 0 8 3 】

続いて、処理手順実行部 4 2 は、プログラム実行処理（ステップ S 2 0 9）として、ステップ S 2 0 9 a ~ S 2 0 9 e の処理を実行する。まず、処理手順実行部 4 2 は、スワップ回数（N u m S t e p）をカウントするカウンタ S をスワップペア毎に初期値 0 で用意する（ステップ S 2 0 9 a）。続いて、処理手順実行部 4 2 は、アクセラレータ 3 0 にアクセラレータプログラムを実行させる（ステップ S 2 0 9 b）。アクセラレータプログラ

10

20

30

40

50

ムの実行が完了したら、処理手順実行部42は、各スワップペアのカウンタSをインクリメントし(ステップS209c)、カウンタSの値が実行形式ファイル22dに指定されたスワップ回数(NumStep)に達しているか否かをスワップペア毎に判定する(ステップS209d)。

【0084】

ステップS209dにおいてカウンタSの値がスワップ回数に達していないと判定されたスワップペアに関しては、処理手順実行部42は、入力バッファ32aと出力バッファ32bとを交換する(ステップS209e)。一方、ステップS209dにおいてカウンタSの値がスワップ回数に達していると判定されたスワップペアに関してはそれ以降のスワップは行わない。ステップS209dにおいてカウンタSの値がスワップ回数に達していないと判定されるスワップペアが一つでもある場合には、処理手順実行部42は、上述のスワップ処理(ステップS209e)を実行した上で、ステップS209bに戻ってアクセラレータ30にアクセラレータプログラムを再実行させる。一方、ステップS209dにおいて全てのスワップペアのカウンタSの値がスワップ回数に達していると判定された場合(ステップS209d: YES)には、処理手順実行部42は、出力データ送信処理において、アクセラレータプログラムの実行により出力バッファ32bに出力された出力データを、当該出力バッファ32bに対応する記憶部20における出力先に送信する(ステップS210)。

10

【0085】

その後、処理手順実行部42は、後処理として、全ての入力バッファ32a及び全ての出力バッファ32bを解放し、アクセラレータプログラムをアクセラレータ30上から無効化する処理を実行する(ステップS211)。

20

【0086】

次に、図8を用いて実行制御部40による実行形式ファイル22dの実行形態について説明する。図8(a)は、実行制御部40による実行形式ファイル22dの実行形態の例を示す図である。実行制御部40の実行形態の仕様は何でもよいが、例えば、実行制御部40は、上述したように、シェルプログラムのように動作し、ユーザによるコマンドライン入力を受け付けるように実装される。この場合、ユーザは、例えばコマンドラインで実行形式ファイル22dのファイル名(「flow-model」)を直接指定することで、当該実行形式ファイル22dを実行制御部40に実行させることができる(図8(a)の(1)参照)。また、ユーザは、例えば実行制御部40の機能を実行することを示すコマンド(cars)に実行形式ファイル22dのファイル名を引数として与えることで、当該実行形式ファイル22dを実行制御部40に実行させることができるようにしてもよい(図8(a)の(2)参照)。また、実行制御部40は、ユーザによる直接のコマンド入力だけでなく、コマンド列を羅列したバッチファイル22e(「batchfile」)の入力を受け付け、バッチファイル22eに記載されたコマンド列を順に実行する仕様としてもよい(図8(a)の(3)参照)。

30

【0087】

このような実行形態の仕様、即ち実行制御部40が対応可能な入力方法及びコマンド文字列に関連付けられる処理内容等は、例えば規則情報記憶部21bに予め記憶させておけばよい。また、本実施形態では、実行形式入力部41が、実行形式ファイル22d以外のコマンド入力及びバッチ形式での入力を受け付ける機能を備えており、コマンド文字列に対応する処理を実行する機能は、例えば処理手順実行部42が備えているものとする。

40

【0088】

処理手順実行部42は、例えば、実行形式ファイル22d又はバッチファイル22eの実行に関しては、プロセス及びスレッド等の独立した実行単位で実行する。このプロセスは、例えばUNIX(登録商標)系OS等のシステムコールの一つである「fork」等により実現される。このような実行単位は、後述する実行管理コマンドで制御可能とされている。

【0089】

50

処理手順実行部 4 2 が実行可能なコマンドとしては、例えば、実行終了コマンド ( e x i t )、ヘルプコマンド ( h e l p )、実行管理コマンド ( p s , k i l l )、仮想バッファ管理コマンド ( v i r t b u f )、実行同期コマンド ( s y n c )、タイマーコマンド ( t i m e r )、繰り返し実行コマンド ( r e p e a t ) 等が実装されている。

#### 【 0 0 9 0 】

実行終了コマンドは、コマンドラインでのコマンド実行を終了する (例えばコンソール画面を閉じる) 機能を実現するコマンドである。ヘルプコマンドは、操作方法を画面上に表示させる機能を実現するコマンドである。実行管理コマンドは、現在実行されている実行単位の状態を画面上に表示したり、現在実行されている実行単位を停止・排除したりする機能を実現するコマンドである。仮想バッファ管理コマンドは、仮想バッファ 2 1 a の作成、削除、情報表示、データファイル ( C S V ファイル ) から仮想バッファ 2 1 a へのデータ読み込み、仮想バッファ 2 1 a からデータファイル ( C S V ファイル ) へのデータ書込み、仮想バッファ 2 1 a の名前変更、2 つの仮想バッファ 2 1 a 間の名前入替等の操作を実現するコマンドである。実行同期コマンドは、既に実行されている実行単位の実行が全て完了するのを待つ機能を実現するコマンドである。タイマーコマンドは、開始時刻を保存したり、保存された開始時刻からの経過時間を計算し出力したりする機能を実現するコマンドである。繰り返し実行コマンドは、引数として、繰り返し回数及び実行対象ファイル ( 実行形式ファイル 2 2 d 又はバッチファイル 2 2 e ) を与えることで、引数として指定された実行対象ファイルを繰り返し回数だけ繰り返す機能を実現するコマンドである。

#### 【 0 0 9 1 】

また、処理手順実行部 4 2 は、実行対象ファイルの実行の種類として、フォアグラウンド実行及びバックグラウンド実行の 2 種類が実装されている。フォアグラウンド実行とは、実行対象ファイルの実行が完了するまで次のコマンドを実行しない実行方式をいう。バックグラウンド実行とは、実行対象ファイルの実行の完了を待たずに次のコマンドを実行する実行方式をいう。例えば図 8 ( a ) に示すように、「 f l o w - m o d e l 2 & 」と、実行形式ファイル 2 2 d のファイル名の後に「 & 」がある場合に、実行制御部 4 0 は、当該実行形式ファイル 2 2 d をバックグラウンド実行する。

#### 【 0 0 9 2 】

実行制御部 4 0 は、このような 2 種類の実行方式 ( フォアグラウンド実行、バックグラウンド実行 ) が実装されていることにより、複数の実行形式ファイル 2 2 d の様々な実行パターン ( 実行順序 ) を容易に実現することができる。具体的には、例えば図 8 ( a ) の下欄に示すような 3 行のコマンド入力により、図 8 ( b ) に示すような複数の実行形式ファイル 2 2 d ( 「 f l o w - m o d e l 1 」、 「 f l o w - m o d e l 2 」、 「 f l o w - m o d e l 3 」 ) の実行パターンが実現される。この場合、「 f l o w - m o d e l 1 」の出力データの出力先と「 f l o w - m o d e l 2 」及び「 f l o w - m o d e l 3 」の入力データの記憶場所とが互いに対応するように、それぞれの実行形式ファイル 2 2 d を定義する必要がある。ただし、このように実行形式ファイル 2 2 d を定義し、上述のようにコマンド列をバッチ形式に並べて実行させるだけで、図 8 ( b ) のような実行パターンを実現することができるという点で、プログラマによるアクセラレータプログラムの実行順序の実装は、アクセラレータプログラムの実行順序だけを考慮したプログラミングにより実現可能であり、従来 C P U 側プログラムを作成した場合と比較して容易となっている。

#### 【 0 0 9 3 】

図 9 に、実行制御部 4 0 が読み取り実行可能なバッチファイルの例を示す。図 9 に示すように、バッチファイルは、例えば X M L 形式で記述され、上述したコマンドと、実行形式ファイル 2 2 d 又はバッチファイルとが羅列されている。図 9 に示すコマンド処理の一部について、その処理内容を説明する。

#### 【 0 0 9 4 】

「 v i r t b u f c r e a t e i n t 1 0 2 4 b u f \_ a 」は、 i n t 型デー

タを1024個格納可能な「buf\_a」という名前の仮想バッファ21aを作成(create)する処理を実行するコマンドである。

【0095】

「virtbuf fill sample1\_\_a.csv buf\_a」は、「buf\_a」という名前で作成された仮想バッファ21aに「sample1\_\_a.csv」というファイル名のデータファイルに含まれる入力データを読み込む(fill)処理を実行するコマンドである。

【0096】

「sample\_\_virt &」は、「sample\_\_virt」という名前の実行対象ファイル(実行形式ファイル22d又はバッチファイル)をバックグラウンド  
10 実行するコマンドである。なお、XMLでは、「&」の文字がそのままの意味(&)を示すものとして使用できないため、XMLにおいて「&」を意味する表記として定められている「&」が記述されている。

【0097】

「sync」は、既に実行されている実行単位の実行が全て完了するのを待つ機能を実現するコマンドである。図9の例では、このコマンドにより、実行中の「sample\_\_virt」の実行が完了するのを待つことができる。即ち、「sample\_\_virt」の実行が完了してから次のコマンド処理を実行させることができる。

【0098】

「repeat 10 sample\_\_virt.xml」は、実行対象ファイル(「  
20 sample\_\_virt.xml」)を10回繰り返して実行するコマンドである。

【0099】

「virtbuf dump c\_\_tmp.csv buf\_c」は、「buf\_c」という名前の仮想バッファ21aに格納された出力データを、「c\_\_tmp.csv」という名前のデータファイルに書き込む(ダンプする)コマンドである。

【0100】

「virtbuf swap buf\_a buf\_c」は、「buf\_a」という名前の仮想バッファ21aと「buf\_c」という名前の仮想バッファ21aとを交換する(例えばポインタを入れ替える)コマンドである。

【0101】

「virtbuf delete buf\_a」は、「buf\_a」という名前の仮想  
30 バッファ21aを削除する処理である。

【0102】

図10は、モデルファイル22c、実行形式ファイル22d、及びバッチファイル22eの包含関係を示す概念図である。図10に示すように、モデルファイル22cは実行形式ファイル22dに包含される。また、実行形式ファイル22dはバッチファイル22eに包含され得る。さらに、バッチファイル22eの中に別のバッチファイル22eを包含する入れ子構造とすることも可能である。

【0103】

次に、図11を用いて、仮想バッファを用いたデータ受け渡しについて説明する。図1  
40 1は、ファイル名が「flow-model1」の実行形式ファイル22dの実行により得られた出力データを、ファイル名が「flow-model2」の実行形式ファイル22dに対して入力データとして与えることで、2つの実行形式ファイル22dを連続して実行する場合を示している。

【0104】

同図左側は、2つの実行形式ファイル22d間でのデータの受け渡しにCSVファイルを介する場合を示している。このようなデータの受け渡しは、ファイル名が「flow-model1」の実行形式ファイル22dの出力データの出力先(OutputのDataFile)と、ファイル名が「flow-model2」の実行形式ファイル22dの入力データの記憶場所(InputのDataFile)とに、同一のCSVファイルを  
50

指定することで実現できる。

【0105】

一方、同図右側は、2つの実行形式ファイル22d間でのデータの受け渡しに仮想バッファ21aを介する場合を示している。このようなデータの受け渡しは、ファイル名が「flow-model1」の実行形式ファイル22dの出力データの出力先(OutputのDataFile)と、ファイル名が「flow-model2」の実行形式ファイル22dの入力データの記憶場所(InputのDataFile)とに、同一の仮想バッファ21aを指定することで実現できる。

【0106】

同図右側のように仮想バッファ21aを介してデータの受け渡しを行うことで、ファイルからのデータ読み込みや、ファイルへのデータ書き込み等のファイルI/Oを排除することができる。

10

【0107】

次に、図12及び図13を用いて、アクセラレータ処理実行装置1による計算処理の応用例について説明する。図12は、アクセラレータ処理実行装置1による計算処理及びバッチファイル22eの応用例を示す図である。図13は、図12に示した応用例における実行形式ファイル22d及びモデルファイル22cを示す図である。

【0108】

図12左側に示される計算処理は、画像処理における高速フーリエ変換(FFT)のアルゴリズム(いわゆるバタフライ演算)である。この計算処理では、実部(Real part)と虚部(Imaginary part)との役割を交換しながら、図中において「Reorder」、「1D FFT」、及び「Transpose」で示される一連の計算処理を2回繰り返した後に「Filter」で示される処理を実行するものである。ここで、各計算処理の詳細(画像処理アルゴリズムの詳細)についての説明は、本発明の内容から外れるため省略する。

20

【0109】

「Reorder」、「1D FFT」、「Transpose」、及び「Filter」で示される処理は、それぞれ異なるアクセラレータプログラムとして実装され、アクセラレータ30により実行される処理である。図12右側に示されるバッチファイル22e中における「reorder.xml」、「fft.xml」、「trans.xml」、及び「lowpass.xml」は、上述の各アクセラレータプログラムを含むモデルファイル22cに関連付けられた実行形式ファイル22dを示している。

30

【0110】

図13の左から1列目は、「fft.xml」で示される実行形式ファイル22dの中身を示している。図13の左から2列目~4列目は、当該実行形式ファイル22dのモデルファイル特定情報(ModelFile)により特定された「fft\_flowmodel.xml」で示されるモデルファイル22cの中身を示している。

【0111】

図13の例では、アクセラレータプログラム(Kernel)における入力引数「in\_real」、「in\_imag」に対してそれぞれ、Indexが「0」、「1」の入力バッファ32aが対応付けられ、「buf\_in\_real」、「buf\_in\_imag」という名前の仮想バッファ21aが対応付けられるように定義されている。同様に、出力引数「out\_real」、「out\_imag」に対してそれぞれ、Indexが「0」、「1」の出力バッファ32bが対応付けられ、「buf\_out\_real」、「buf\_out\_imag」という名前の仮想バッファ21aが対応付けられるように定義されている。

40

【0112】

図12右側に示されるバッチファイル22eが実行形式入力部41によって入力され、処理手順実行部42により実行される処理の概要について説明する。まず、入力データの仮想バッファ21a(buf\_in\_real、buf\_in\_imag)と出力デー

50

タ用の仮想バッファ21a (buf\_out\_real、buf\_out\_imag)をCPU側メモリ21上に作成する処理が実行される(図12の(A)参照)。

【0113】

続いて、入力データ用の仮想バッファ21a (buf\_in\_real、buf\_in\_imag)にそれぞれ、「lenna128\_real.csv」及び「lenna128\_imag.csv」という名前の入力データファイル22aに含まれる入力データを読み込む処理が実行される(図12の(B)参照)。

【0114】

続いて、上述の各実行形式ファイル22dを順に実行する処理が実行される(図12の(C)参照)。ここで、各実行形式ファイル22dの実行毎に、入力データ用の仮想バッファ21aと出力データ用の仮想バッファ21aとを交換する処理(virtbuf\_swap)が実行される。これにより、例えば「Reorder」の処理(「reorder.xml」で示される実行形式ファイル22dを実行する処理)によって得られた出力データを、一旦ファイルに書き出すことなく、次の「1D FFT」の処理(「fft.xml」で示される実行形式ファイル22dを実行する処理)に対する入力データとして受け渡す処理が実現される。

【0115】

続いて、「Filter」の処理(「lowpass.xml」で示される実行形式ファイル22dを実行する処理)まで完了した後に、出力データ用の仮想バッファ21に出力された出力データを、それぞれ対応する出力データファイル22bに書き出す処理が実行される(図12の(D)参照)。

【0116】

その後、後処理として、全ての仮想バッファ21aを削除する処理が実行される(図12の(E)参照)。

【0117】

次に、図14を用いて、CPUと、記憶部(CPU側メモリ、補助記憶部)と、アクセラレータとを備えるコンピュータをアクセラレータ処理実行装置1として機能させるためのアクセラレータ処理実行プログラムP1について説明する。

【0118】

図14は、アクセラレータ処理実行プログラムP1のモジュールを示すブロック図である。図14に示すように、アクセラレータ処理実行プログラムP1は、実行形式入力モジュールP41と、処理手順実行モジュールP42と、を備える。上記の実行形式入力モジュールP41及び処理手順実行モジュールP42が実行されることにより実現される機能は、上述したアクセラレータ処理実行装置1において対応する実行形式入力部41及び処理手順実行部42の機能と同様である。

【0119】

このように構成されたアクセラレータ処理実行プログラムP1は、例えばCD-ROM及びDVD等の記録媒体に記憶され、アクセラレータ処理実行装置1として用いられるコンピュータのCPUにより実行される。具体的には、当該コンピュータは、例えばCD-ROMドライブ及びDVDドライブ等の記録媒体読取部を備え、当該記録媒体読取部に記録媒体が挿入されると、記録媒体読取部から記録媒体に格納されたアクセラレータ処理実行プログラムP1にアクセス可能となる。そして、当該アクセラレータ処理実行プログラムP1を当該コンピュータに実行させることによって、当該コンピュータを、本実施形態に係るアクセラレータ処理実行装置1として動作させることが可能となる。

【0120】

なお、アクセラレータ処理実行プログラムP1は、搬送波に重畳されたコンピュータデータ信号としてネットワークを介して提供されるものであってもよい。この場合、アクセラレータ処理実行装置1として用いられるコンピュータは、モデム等のデータ通信を行う通信制御部によって受信したアクセラレータ処理実行プログラムP1をCPU側メモリ21に格納することにより、当該アクセラレータ処理実行プログラムP1を実行することが

10

20

30

40

50

できる。

【0121】

以上説明したように、本実施形態に係るアクセラレータ処理実行装置1では、実行形式入力部41が実行形式ファイル22dを入力して実行情報を取得し、処理手順実行部42が当該実行情報及び予め定めた規則（規則情報記憶部21bに記憶された規則情報）に基づいてアクセラレータに関する処理手順を決定し、実行する。このように、CPU10側とアクセラレータ30側との間の入力データ及び出力データの受け渡しを含む各処理手順が、実行制御部40によって適切に決定及び実行されるため、プログラマは、CPU10側とアクセラレータ30側との間のデータの受け渡し等の処理手順を詳細に規定したCPU側プログラムを作成する手間から解放される。即ち、上記アクセラレータ処理実行装置1によれば、アクセラレータプログラムをアクセラレータ30に実行させるために、プログラマは、実行形式ファイル22d及びアクセラレータプログラムを含むモデルファイル22cを用意するだけでよいため、プログラマのプログラム生産性を向上させることができる。

10

【0122】

特に、複数のアクセラレータプログラムの実行パターン（図8（b）に示したような直列実行又は並列（同時）実行）を変更したり、この変更に伴い入出力データを変更したりする場合には、従来のCPU側プログラムでは、多くの修正が必要となる。これに対し、本実施形態に係るアクセラレータ処理実行装置1によれば、バッチファイル22eにおいて実行形式ファイル22dの実行順序、及び実行形式ファイル22dの入出力データに関する定義（Input及びOutputのData File）を変更するだけでよく、プログラマの負担が軽減される。

20

【0123】

以上から、プログラマは、モデルファイル22cにアクセラレータプログラムを実装し、実行形式ファイル22d及びその他の実行可能なコマンドを羅列したバッチファイル22eを作成することで、複数のアクセラレータプログラムの直列実行及び並列実行を含めて、アクセラレータプログラムの実行順序の設計及び実装を容易に行うことができる。

【0124】

また、プログラマがCPU側プログラムを作成する必要がなくなるということは、裏を返せば、プログラマに対して、CPU10の機能の実行及び制御の自由度が高く、CPU側メモリ21及びアクセラレータ側メモリ32等を直接操作すること等が可能なCPU側プログラムの実行を許可しないことを意味する。これにより、データ破壊及び機密情報の書き換え等を防止し、ハードウェアの安全性を高めることができる。

30

【0125】

また、CPU側メモリ21等の記憶領域を自由にアクセスしたり、CPU10のハードウェアをアクセスしたり、CPU10のプログラムを書き換えたりすることは、アクセラレータプログラムからはできない。従って、いわゆるトロイの木馬型のコンピュータウイルス等を本発明で扱う実行方式（モデルファイル22c、実行形式ファイル22d、バッチファイル22e等を定義する方式）では作成することができない。

【0126】

また、上記アクセラレータ処理実行装置1によれば、アクセラレータプログラムの入力データ又は出力データの受け渡しを、ファイル（例えばCSVファイル等）を介することなく、仮想バッファ21aを介して実行することができる。従って、データの受け渡しにおいてファイルI/Oを排除して、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。

40

【0127】

また、上記アクセラレータ処理実行装置1によれば、アクセラレータプログラムの1回の実行が完了する毎に、例えばポインタの入替等により入力バッファ32aと出力バッファ32bとを入れ替えてアクセラレータプログラムを再実行することができるため、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。

50

具体的には、出力バッファ 3 2 b に出力されたアクセラレータプログラムの出力データを一旦 CPU 1 0 側の記憶部 2 0 に移動させてから再度アクセラレータ 3 0 側に入力データとして入力するといった無駄なデータ受け渡し処理を省略することができる。

【 0 1 2 8 】

以上、本発明に係る実施形態について詳細に説明した。しかし、本発明は、上記実施形態に限定されるものではない。本発明は、その要旨を逸脱しない範囲において様々な変形が可能である。

【 0 1 2 9 】

本実施形態では、処理手順実行部 4 2 が処理手順を決定するための実行情報が、実行形式ファイル 2 2 d 及びモデルファイル 2 2 c に含まれるものとして説明したが、実行情報は、このような形態に限定されない。例えば、実行形式ファイル 2 2 d とモデルファイル 2 2 c とは同一のファイルとして構成されていてもよいし、3 つ以上のファイルに情報が分散されていてもよい。また、これらのファイルのファイル形式は XML 形式以外の形式であってもよい。また、本実施形態では、モデルファイル 2 2 c として本発明者らが開発した flow - model を用いる例を説明したが、必ずしも flow - model を用いなくてもよい。即ち、実行形式入力部 4 1 に入力させるものは、本実施形態で説明した実行情報に相当する情報を抽出可能なものであれば何でもよい。

【 0 1 3 0 】

また、入力データファイル 2 2 a、出力データファイル 2 2 b、モデルファイル 2 2 c、及び実行形式ファイル 2 2 d は、必ずしも補助記憶部 2 2 に記憶されなくともよく、例えば、ネットワークを介してアクセラレータ処理実行装置と互いに通信可能なリモートコンピュータ等が備える記憶装置に記憶されてもよい。

【 0 1 3 1 】

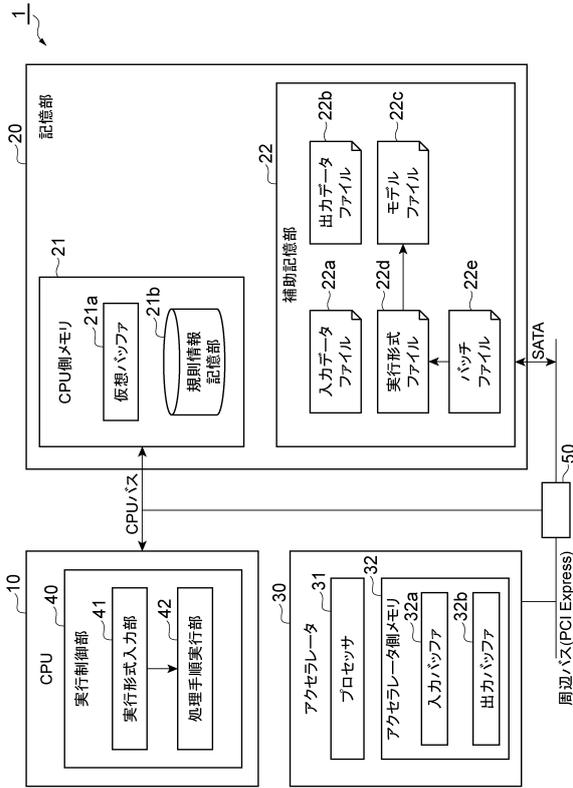
また、モデルファイル 2 2 c、実行形式ファイル 2 2 d、及びバッチファイル 2 2 e は、例えば RSA 等により暗号化してもよい。この場合における実行形式ファイル 2 2 d の実行の一例について以下に示す。例えば、暗号化のパブリックキー（復号のためのキー）は、実行制御部 4 0（実行形式入力部 4 1）がアクセスできる場所（例えばディレクトリ等）に保存され、暗号化のプライベートキー（暗号化に用いられたキー）は、プログラムによって与えられる構成とする。この場合、実行制御部 4 0 は、コマンドライン等により実行形式ファイル 2 2 d を入力したら、所定の方法でプログラムに対してプライベートキーを要求する。そして、プログラムから所定の方法でプライベートキーを受け取り、当該プライベートキーで暗号化された上記ファイルを展開し、その内容から実行情報を取得し、当該実行情報に基づいて処理手順を決定及び実行する。このような実行方式によれば、実行制御部 4 0 は、プログラムによって入力されたファイルの安全性（プログラムによって作成された真正のファイルであること）を確認した上で、上記ファイルを実行することができるため、アクセラレータプログラムの実行におけるセキュリティが向上する。また、モデルファイル 2 2 c、実行形式ファイル 2 2 d、及びバッチファイル 2 2 e 等に記述された機密情報の漏えいを防止することができる。

【 符号の説明 】

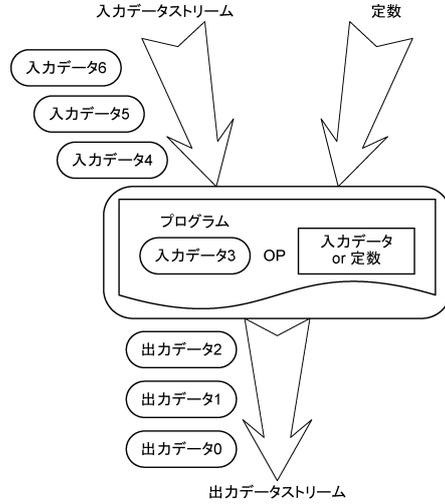
【 0 1 3 2 】

1 ... アクセラレータ処理実行装置、 1 0 ... CPU、 2 0 ... 記憶部（第 1 の記憶手段）、 2 1 ... CPU 側メモリ、 2 1 a ... 仮想バッファ、 2 1 b ... 規則情報記憶部、 2 2 ... 補助記憶部、 2 2 a ... 入力データファイル、 2 2 b ... 出力データファイル、 2 2 c ... モデルファイル、 2 2 d ... 実行形式、 2 2 e ... バッチファイル、 3 0 ... アクセラレータ、 3 1 ... プロセッサ、 3 2 ... アクセラレータ側メモリ、 3 2 a ... 入力バッファ、 3 2 b ... 出力バッファ、 4 0 ... 実行制御部、 4 1 ... 実行形式入力部、 4 2 ... 処理手順実行部、 5 0 ... バスブリッジ、 P 1 ... アクセラレータ処理実行プログラム、 P 4 1 ... 実行形式入力モジュール、 P 4 2 ... 処理手順実行モジュール。

【 図 1 】



【 図 2 】



【 図 3 】

```

<?xml version="1.0" encoding="ASCII"?>
<FlowModel xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/
2001/XMLSchema">
  <Input>
    <Name>a</Name>
    <DataType>INT</DataType>
    <Length>1024</Length>
    <Index>0</Index>
  </Input>
  <Input>
    <Name>b</Name>
    <DataType>INT</DataType>
    <Length>1024</Length>
    <Index>1</Index>
  </Input>
  <Output>
    <Name>c</Name>
    <DataType>INT</DataType>
    <Length>1024</Length>
    <Index>0</Index>
  </Output>
  <Kernel>
    __kernel void test(__global int *a, __global int *b, __global int *c
    {
      int id;
      id = get_global_id(0);
      //c[id] = a[id] - 100 + b[id] ;
      c[id] = a[id] + b[id] + 100;
      return;
    }
  </Kernel>
  <KernelName>test</KernelName>
  <LangType>SHADERLANG_OPENCL1.0</LangType>
  <RuntimeType>RUNTIME_OPENCLGPU</RuntimeType>
  <Binary />
  <TotalNumThreads>1024</TotalNumThreads>
  <TotalNumBlocks>128</TotalNumBlocks>
  <SwapCounter>0</SwapCounter>
</FlowModel>

```

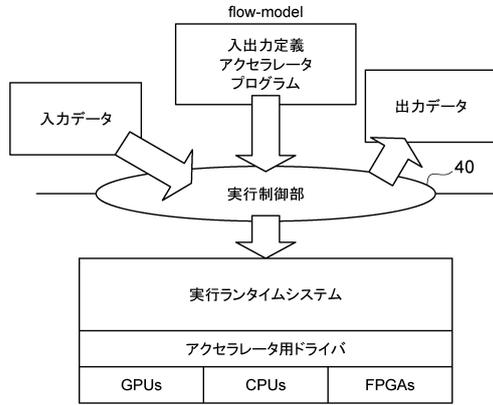
【 図 4 】

```

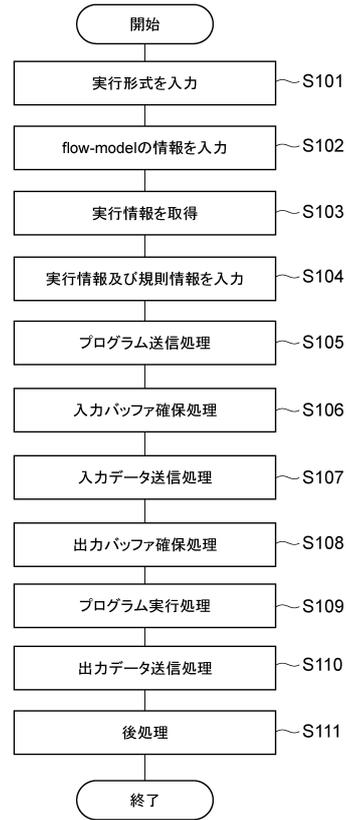
<?xml version="1.0" encoding="ASCII"?>
<CarshEx>
  <ModelFile>sample1.xml</ModelFile>
  <Input>
    <Name>a</Name>
    <DataFile>sample1_a.csv</DataFile>
  </Input>
  <Input>
    <Name>b</Name>
    <DataFile>sample1_b.csv</DataFile>
  </Input>
  <Output>
    <Name>c</Name>
    <DataFile>sample1_c.csv</DataFile>
  </Output>
  <SwapPair>
    <Input>a</Input>
    <Output>c</Output>
    <NumSwap>1000</NumSwap>
  </SwapPair>
</CarshEx>

```

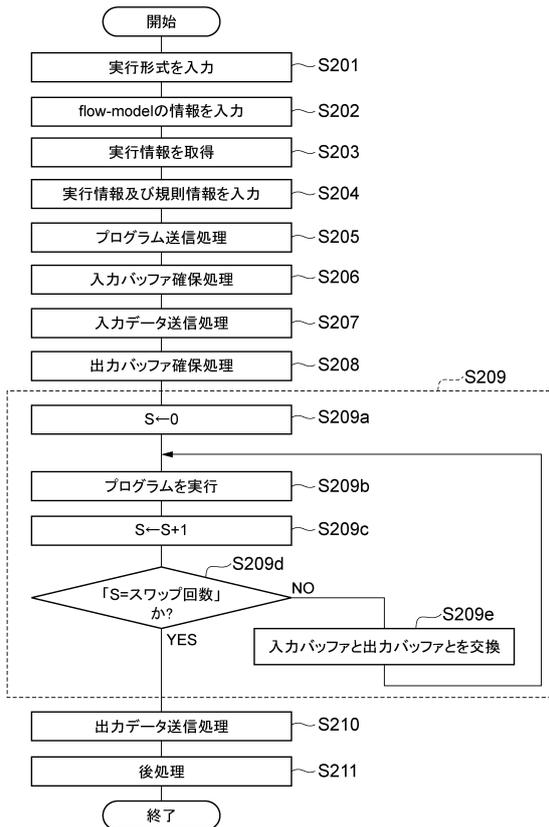
【図5】



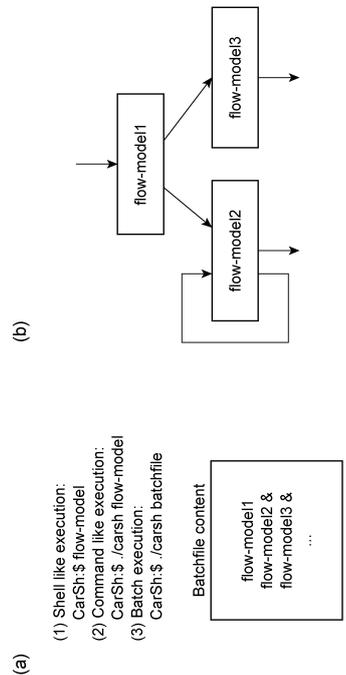
【図6】



【図7】



【図8】



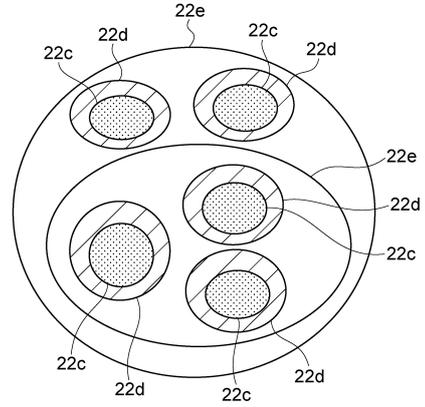
【 図 9 】

```

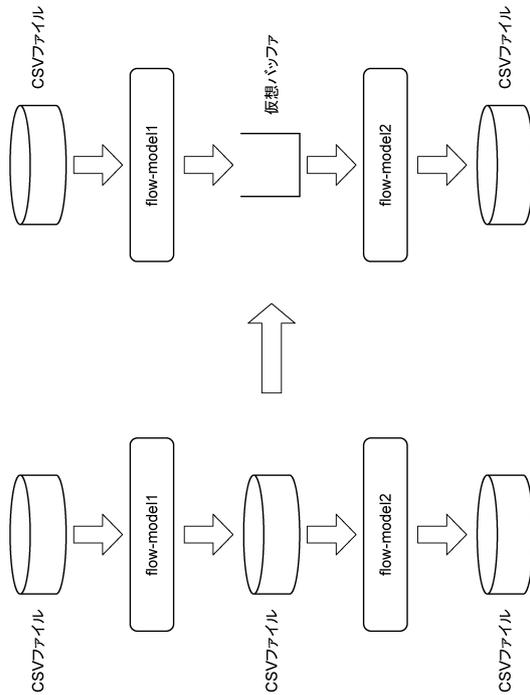
<?xml version="1.0" encoding="ASCII"?>
<CarshBat>
virtbuf create int 1024 buf_a
virtbuf create int 1024 buf_b
virtbuf fill sample1_a.csv buf_a
virtbuf fill sample1_b.csv buf_b
virtbuf create int 1024 buf_c
debug off
timer start
sample_virt &amp;
sync
timer stop
timer start
repeat 10 sample_virt.xml
sync
timer stop
virtbuf list
virtbuf dump c_tmp.csv buf_c
virtbuf list
virtbuf swap buf_a buf_c
virtbuf list
virtbuf delete buf_a
virtbuf list
virtbuf delete buf_b
virtbuf list
virtbuf delete buf_c
virtbuf list
exit
</CarshBat>

```

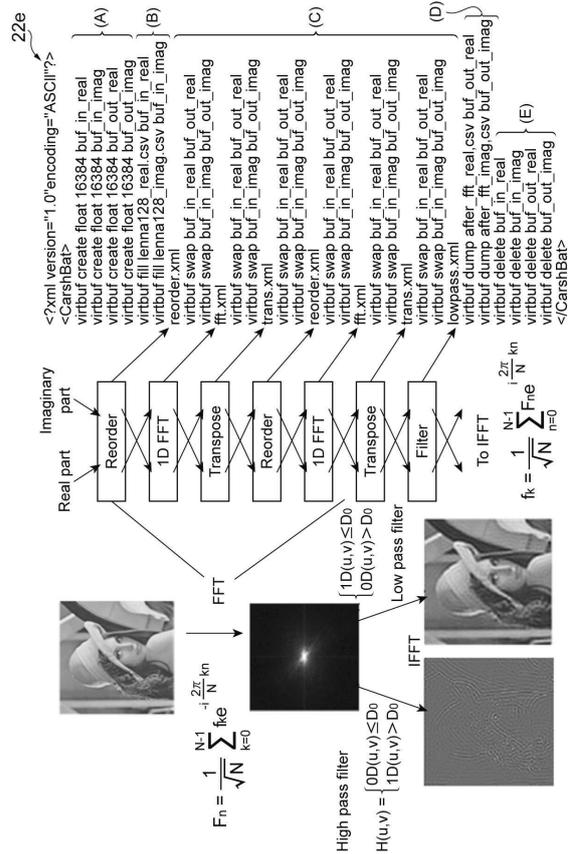
【 図 10 】



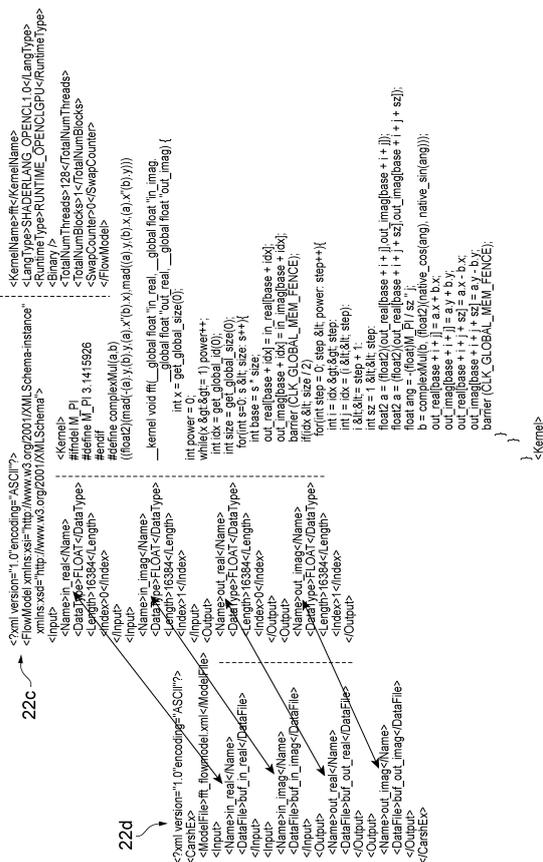
【 図 11 】



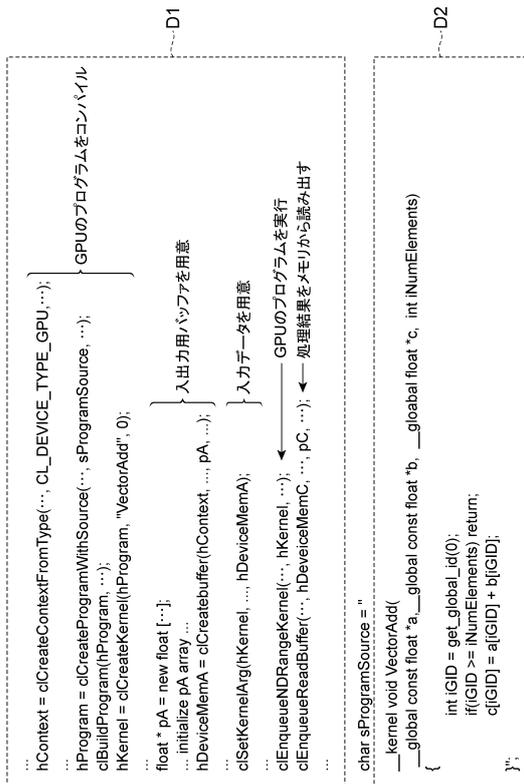
【 図 12 】



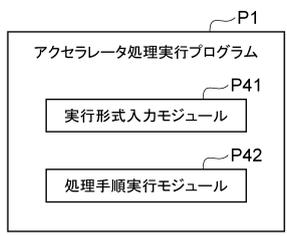
【 図 1 3 】



【 図 1 5 】



【 図 1 4 】



フロントページの続き

(58)調査した分野(Int.Cl. , DB名)

G 0 6 F      9 / 4 4