(19)日本国特許庁(JP)

# (12)特許公報(B2)

(11)特許番号
特許第7025104号
(P7025104)

(45)発行日　令和4年2月24日(2022.2.24)

(24)登録日　令和4年2月15日(2022.2.15)

(51)国際特許分類
*Ｇ０６Ｆ　16/28　(2019.01)*
*Ｇ０６Ｆ　9/448(2018.01)*

ＦＩ
Ｇ０６Ｆ　16/28
Ｇ０６Ｆ　9/448　１００

請求項の数　6　（全18頁）

(21)出願番号　特願2020-569481(P2020-569481)
(86)(22)出願日　令和2年1月10日(2020.1.10)
(86)国際出願番号　PCT/JP2020/000673
(87)国際公開番号　WO2020/158347
(87)国際公開日　令和2年8月6日(2020.8.6)
審査請求日　令和2年12月25日(2020.12.25)
(31)優先権主張番号　特願2019-14248(P2019-14248)
(32)優先日　平成31年1月30日(2019.1.30)
(33)優先権主張国・地域又は機関
日本国(JP)

(73)特許権者　000004226
日本電信電話株式会社
東京都千代田区大手町一丁目５番１号
(74)代理人　100108855
弁理士　蔵田 昌俊
(74)代理人　100075672
弁理士　峰 隆司
(74)代理人　100179062
弁理士　井上 正
(72)発明者　柏木 啓一郎
東京都千代田区大手町一丁目５番１号
日本電信電話株式会社内
(72)発明者　石井 久治
東京都千代田区大手町一丁目５番１号
日本電信電話株式会社内

最終頁に続く

(54)【発明の名称】　情報処理装置、方法およびプログラム

(57)【特許請求の範囲】
【請求項１】
シリアライズ処理に割り当てられる仮想アドレス空間のうちスタック領域に確保されるメモリ領域の長さがコンパイル時に決定されるメッセージ構造が示される情報を生成する生成処理部と、
前記生成処理部により生成された情報を用いて、機器からのデータに対するシリアライズ処理を行なってデータ蓄積装置に書き込むシリアライズ処理を行なうシリアライズ処理部と、
前記データ蓄積装置に書き込まれた、前記シリアライズ処理されたデータを読み出し、前記生成処理部により生成された情報を用いて、前記読み出されたデータに対するデシリアライズ処理を行なうデシリアライズ処理部と、
を備える情報処理装置。
【請求項２】
前記シリアライズ処理部は、
前記生成処理部により生成された情報のコンパイル時に、連続したメモリ領域を前記スタック領域にアロケートし、
書き込み対象となる前記シリアライズ処理されたデータを、前記アロケートされたメモリ領域の変数に代入し、前記代入されたデータを前記仮想アドレス空間のカーネル空間のメモリ領域に書き込む、
請求項１に記載の情報処理装置。

【請求項３】
前記生成処理部により生成された情報は、読み出し対象のデータへのポインタを含み、
前記デシリアライズ処理部は、
前記読み出し対象のデータが書き込まれた物理アドレス空間のメモリ領域と、前記仮想アドレス空間のカーネル空間のメモリ領域との対応付けを行ない、
前記カーネル空間のメモリ領域と、前記仮想アドレス空間のユーザ空間のメモリ領域との対応付けを行ない、
前記ユーザ空間のメモリ領域を参照し、
前記物理アドレス空間のメモリ領域に書き込まれているデータを前記対応付けられるカーネル空間のメモリ領域に読み出し、
前記生成処理部により生成された情報に含まれるポインタのダイナミックキャストにより前記カーネル空間のメモリ領域と前記ユーザ空間のメモリ領域との間のアドレスの読み替えを行なうことで、前記読み出し対象のデータのデシリアライズ処理を行なう、
請求項１に記載の情報処理装置。
【請求項４】
前記生成処理部により生成された情報は、実データ、および所定の順序の複数のメタデータに関する情報を含む、
請求項１に記載の情報処理装置。
【請求項５】
データ蓄積装置を具備する情報処理装置が行なう情報処理方法であって、
シリアライズ処理に割り当てられる仮想アドレス空間のうちスタック領域に確保されるメモリ領域の長さがコンパイル時に決定されるメッセージ構造を示す情報を生成することと、
前記生成された情報を用いて、機器からのデータに対するシリアライズ処理を行なって前記データ蓄積装置に書き込むことと、
前記データ蓄積装置に書き込まれた、前記シリアライズ処理されたデータを読み出し、前記生成された情報を用いて、前記読み出されたデータに対するデシリアライズ処理を行なうことと、
を備える情報処理方法。
【請求項６】
請求項１乃至４のいずれか１項に記載の情報処理装置の前記各部としてプロセッサを機能させる情報処理プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施形態は、情報処理装置、方法およびプログラムに関する。
【背景技術】
【０００２】
IoT(Internet of Things)システム（system）において、機器により生成されたデータ（data）がシリアライズ(serialize)されて送信され、ブローカー（broker）、ミドルウェア（middleware）を経由してデータベース（database）に書き込まれ、この書き込まれたデータが読み出される際に当該データに対するデシリアライズ(deserialize)が行なわれる場合がある。
【０００３】
一般的に知られているシリアライズ又はデシリアライズのフォーマット（format）（以下、シリアライズフォーマット）は以下の第１および第２のフォーマットに分類される。
第１のフォーマットはテキスト（text）によるシリアライズフォーマットであり、例としてはJSON(例えば非特許文献１を参照)、XML(Extensible Markup Language)(例えば非特許文献２を参照)などが挙げられる。
【０００４】
テキストによるシリアライズフォーマットは、可読性が高いという利点を有する。一方で

、欠点として、（１）データのサイズ（size）が大きいこと、および（２）一部の記号に係るエスケープ（escape）処理が必要であることに起因して、これらの記号が使用される場合又は種別が組み込みデータ型である場合にシリアライズ処理又はデシリアライズ処理が遅くなること、が挙げられる。

【０００５】

第２のフォーマットはバイナリ（binary）によるシリアライズフォーマットであり、例としてはMessagePack（例えば非特許文献３を参照）、Protocol Buffers（例えば非特許文献４を参照)などが挙げられる。バイナリによるシリアライズフォーマットの利点としては、データサイズが小さく、バイナリデータがそのまま扱われ得るため、シリアライズ処理又はデシリアライズ処理が高速に行なわれることが挙げられる（例えば非特許文献５を参照）。

【０００６】

バイナリのシリアライズフォーマットの欠点は可読性の低さであるが、以下の（１）～（３）が挙げられる。

（１）IoTシステムではセンサ（sensor）データ等の整数値のデータが扱われる場合が多く、文字列が少ない

（２）IoTシステム等の機器間の通信では人により読まれることが少ないため可読性が重要ではない

（３）IoTシステムにおいて機器のデータを集約する装置は、計算及びメモリ（memory）のリソース（resource）に制約がある場合が多い。

これらのことから、シリアライズ処理又はデシリアライズ処理には、負荷が小さいフォーマットである、バイナリのシリアライズフォーマットが適する。

【先行技術文献】

【非特許文献】

【０００７】

【文献】"JSONの紹介"、インターネット＜URL:https://www.json.org/json-ja.html＞

Bray, Tim, et al.、"Extensible Markup Language (XML)." 、World Wide Web Journal 2.4 (1997): 27-66.

"MessagePack: It's like JSON. but fast and small."、インターネット＜URL:https://msgpack.org/＞

"Protocol Buffers"、インターネット＜URL:https://developers.google.com/protocol-buffers/＞

Vanura, Jan, and Pavel Kriz.、"Perfomance Evaluation of Java（登録商標）, JavaScript（登録商標） and PHP Serialization Libraries for XML, JSON and Binary Formats."、International Conference on Services Computing. Springer, Cham, 2018.

【発明の概要】

【発明が解決しようとする課題】

【０００８】

近年、データを生成するデバイス（device）の近くに配置されるエッジ装置（エッジ（edge））にて処理を行なうIoTシステムが注目されている。

集約された豊富なリソースを利用可能なクラウド（cloud）サービス（service）と比較して、各地点に分散するエッジとして使用されるゲートウェイ（gateway）装置は安価である必要があるが、多数の機器からのデータがゲートウェイ装置に集約される。このため、従来のバイナリのシリアライズフォーマットが用いられた処理よりも、更に高速に処理が行なわれなければ、収容される機器の数を増やすことが難しい。

また、処理対象のデータ数が多いIoTデータ処理では、シリアライズ処理およびデシリアライズ処理を、より少ないメモリ容量で、より高速に実施可能であるシリアライズフォーマットが望まれるという課題があった。

【０００９】

本発明は、上記の事情に着目してなされたものであり、その目的は、データのシリアライズ処理およびデシリアライズ処理を少ないメモリ容量で高速に行なうことができる技術を提供することである。

【課題を解決するための手段】

【００１０】

上記目的を達成するために、この発明の一態様では、情報処理装置が、シリアライズ処理に割り当てられる仮想アドレス空間のうちスタック領域に確保されるメモリ領域の長さがコンパイル時に決定されるメッセージ構造が示される情報を生成する生成処理部と、前記生成処理部により生成された情報を用いて、機器からのデータに対するシリアライズ処理を行なってデータ蓄積装置に書き込むシリアライズ処理を行なうシリアライズ処理部と、前記データ蓄積装置に書き込まれた、前記シリアライズ処理されたデータを読み出し、前記生成処理部により生成された情報を用いて、前記読み出されたデータに対するデシリアライズ処理を行なうデシリアライズ処理部と、を備える。

【００１１】

本発明の一態様は、データ蓄積装置を具備する情報処理装置が行なう情報処理方法であって、シリアライズ処理に割り当てられる仮想アドレス空間のうちスタック領域に確保されるメモリ領域の長さがコンパイル時に決定されるメッセージ構造を示す情報を生成することと、前記生成された情報を用いて、機器からのデータに対するシリアライズ処理を行なって前記データ蓄積装置に書き込むことと、前記データ蓄積装置に書き込まれた、前記シリアライズ処理されたデータを読み出し、前記生成された情報を用いて、前記読み出されたデータに対するデシリアライズ処理を行なうことと、を備える。

【発明の効果】

【００１２】

本発明によれば、データのシリアライズ処理およびデシリアライズ処理を少ないメモリ容量で高速に行なうことが可能になる。

【図面の簡単な説明】

【００１３】

【図１】図１は、本発明の一実施形態に係る情報処理装置の適用例を示す図である。

【図２】図２は、本発明の一実施形態に係る情報処理装置に適用される構造体の一例を示す図である。

【図３】図３は、本発明の一実施形態に係る情報処理装置に適用されるメッセージ（message）フォーマットの一例を示す図である。

【図４】図４は、本発明の一実施形態に係る情報処理装置に適用されるメッセージフォーマットのヘッダ（header）定義の一例を示す図である。

【図５】図５は、本発明の一実施形態に係る情報処理装置に適用されるメッセージフォーマットのデータ型定義の一例を示す図である。

【図６】図６は、本発明の一実施形態に係る情報処理装置による構造体生成の手順の一例を示すフローチャート（flow chart）である。

【図７】図７は、本発明の一実施形態に係る情報処理装置によるデータ処理の手順の一例を示すフローチャートである。

【図８】図８は、一般的なシリアライズフォーマットが適用されたときのデータ書き込みおよびシリアライズ処理の手順を説明する図である。

【図９】図９は、本発明の一実施形態に係る情報処理装置によるデータ書き込みおよびシリアライズ処理の手順の一例を説明する図である。

【図１０】図１０は、本発明の一実施形態に係る情報処理装置によるデータ書き込みおよびシリアライズ処理の手順の一例を示すフローチャートである。

【図１１】図１１は、一般的なシリアライズフォーマットが適用されたときのデータ読み出しおよびデシリアライズ処理の手順の一例を説明する図である。

【図１２】図１２は、本発明の一実施形態に係る情報処理装置によるデータ読み出しおよびデシリアライズ処理の手順の一例を説明する図である。

【図１３】図１３は、本発明の一実施形態に係る情報処理装置によるデータ読み出しおよびデシリアライズ処理の手順の一例を示すフローチャートである。
【図１４】図１４は、一般的なシリアライズフォーマットが適用されたときのフィルタ処理の手順の一例を説明する図である。
【図１５】図１５は、本発明の一実施形態に係る情報処理装置によるフィルタ処理の手順の一例を説明する図である。
【図１６】図１６は、本発明の一実施形態に係る情報処理装置のハードウエア（hardware）構成の一例を示すブロック図である。
【発明を実施するための形態】
【００１４】
以下、図面を参照しながら、この発明に係わる一実施形態について説明する。
従来の、バイナリのシリアライズフォーマットでは、コンパイル（compile）時にデータサイズは決定されない。このデータサイズは、例えばデータが物理アドレス空間（address Space）のメモリ領域（単にメモリと称することもある）であるディスク（disk）に書き込まれる際に、仮想アドレス空間におけるユーザ（user）空間のうちスタック（stack）領域に確保されるメモリ領域のサイズ（長さ）を意味する。このため、データが書き込まれる度に、ユーザ空間にのうちヒープ(heap)に必要なメモリが確保される必要があり、この必要なメモリが確保される処理であるメモリアロケーション（allocation）に時間を要していた。
【００１５】
そこで、本発明の一実施形態では、以下の特徴を有する、MessagePack互換による高速なシリアライズまたはデシリアライズ方式であるutpackについて説明する。
utpackの特徴としては、少なくとも以下の（１）～（５）が挙げられる。
（１） データのシリアライズとデシリアライズのプロセスで共通の事前知識（構造体）を有する。
構造体とは、メタデータ(metadata)の個数、データ型、順番（相対アドレス）などで定義された、メッセージの構造である。この構造体は、コンパイル時に、データサイズとデータの先頭アドレスとから、各変数への相対アドレスの情報を生成する。
【００１６】
（２） 構造体がシリアライズに用いられることにより、データ書き込み時にスタック上のメモリ領域でシリアライズが直接行なわれることにより、高速にシリアライズが行なわれ得る。
構造体がシリアライズに用いられることとは、メッセージフォーマットの指定に沿うように構造体が予め定義され、この構造体として確保されたスタック領域の変数に値が代入されることで、シリアライズが行われることを意味する。この、メッセージフォーマットの指定に沿うように構造体があらかじめ定義されるために、後述されるデータ型が用いられて構造体が定義される。
【００１７】
（３） 構造体がデシリアライズに用いられることにより、データ読み出し時のメモリアロケーション、およびカーネル（Kernel）空間とユーザ空間との間のデータコピー（copy）を無くし、高速にデシリアライズが行なわれ得る。構造体がデシリアライズに用いられることとは、構造体が用いられて、当該構造体で定義される、変数を表す相対的なアドレスのポインタ（pointer）のダイナミックキャスト（dynamic cast）（動的キャスト）によりアドレスの読み替えが行なわれ、デシリアライズが行なわれることを意味する。
ここでは、構造体に基づいて、mapされた領域の先頭のアドレスが、変数を表す相対的なアドレスのポインタに設定され、このポインタのダイナミックキャストにより、カーネル空間とユーザ空間との間のアドレスの読み替えが行なわれることで、データのデシリアライズが行なわれる。
【００１８】
（４） MessagePackと互換性のある形式であるため、構造体なしでもデシリアライズは

可能である。
（５） 構造体が定義されるソースコード（source code）と、この定義に基づいてシリアライズまたはデシリアライズ処理が行なわれるためのソースコードとが、メタプログラミング(metaprograming)によって生成される。これにより、プログラマ（Programmer）は、シリアライズまたはデシリアライズ処理のソースコードを構造体の定義が変更される度に改修しなくても済む。
【００１９】
utpackは、コンパイル時にデータサイズが決定されるように、固定長の変数が用いられて構造体が規定され、構造体の情報がシリアライズとデシリアライズのプロセス（process）間で共有され得る。さらに、構造体に対応したメッセージフォーマットが定義されることで、連続したメモリ領域がスタック上に確保されることにより、メモリのアロケート時間が短縮される。
つまり、utpackは、構造体の定義とメモリ利用とに係る工夫により、シリアライズまたはデシリアライズ処理が少ないメモリ容量で高速に行なわれる方式である。
【００２０】
図１は、本発明の一実施形態に係る情報処理装置の適用例を示す図である。この情報処理装置は、utpackの方式を実現する装置である。
図１に示されるように、情報処理装置１０は、メタデータ情報設定部１１、構造体生成部１２、実データ受信部１３、メタデータ生成部１４、シリアライズ処理部１５、データ蓄積部１６およびデシリアライズ処理部１７を有する。図１に示される情報処理装置１０の機能は、プログラム（program）を実行するＣＰＵ（Central Processing Unit）等のプロセッサ（processor）、キーボード（keyboard）などの入力装置、ディスプレイ（display）などの出力装置、およびＲＡＭ（Random Access Memory）、ＲＯＭ（Read Only Memory）等の記憶媒体等が用いられることで実現される。データ蓄積部１６は、例えばシングルスレッド（single thread）の組み込みデータベースである。各部の動作については後述する。
【００２１】
図２は、本発明の一実施形態に係る情報処理装置に適用される構造体の一例を示す図である。
図２に示された例は、utpackが適用されたときの構造体の一例であり、オブジェクト（object）数、メタデータのデータ型、メタデータ、データサイズ、およびアクセス（access）対象のデータへのポインタを含む。この構造体では、変数の型、数および順番が定義される。この構造体により、コンパイル時にデータ書き込みに必要なメモリ領域のサイズが決定され、スタック上にメモリ領域を確保可能である。図２に示された例では、構造体の一例は可変長テンプレート（template）クラス（class）であるとする。
【００２２】
図３は、本発明の一実施形態に係る情報処理装置に適用されるメッセージフォーマットの一例を示す図である。
図３に示された例は、utpackが適用されたときのメッセージフォーマットの一例である。utpackが適用されたときのメッセージフォーマットはMessagePack互換で規定されている。このため、シリアライズとデシリアライズのプロセスが構造体の情報を共有しない場合でもデータの変換が可能である。
【００２３】
図３に示された例では、メッセージフォーマットは、１メッセージに含まれるオブジェクト数（可変）（図３のａ）、任意のデータ型のオブジェクトａ（メタデータａ）（図３のｂ）、任意のデータ型のオブジェクトｂ（メタデータｂ）（図３のｃ）およびLastオブジェクト（図３のｄ）を含む。
【００２４】
任意のデータ型のオブジェクトａは、任意のデータ型（図３のｅ）およびメタデータ（図３のｆ）を含む。任意のデータ型のオブジェクトｂは、任意のデータ型（図３のｇ）およ

びメタデータ（図３のｈ）を含む。Lastオブジェクトは、データ型（bin32）（図３のｉ）、実データの長さ（図３のｊ）および実データ（図３のｋ）を含む。

【００２５】

図３に示されたデータ型はfixarray(an array whose length is upto 15 elements)（メタデータのオブジェクト０～１４個＋Lastオブジェクト）である。

図３のａ、ｅ～ｊはヘッダ（実データ以外の部分）（図３のm）に対応し、図３のｋは、ボディ（body）（実データ）（図３のｎ）に対応する。

【００２６】

図４は、本発明の一実施形態に係る情報処理装置に適用されるメッセージフォーマットのヘッダ定義の一例を示す図である。図４ではutpackが適用されたときのヘッダ定義の一例を示す。

図５は、本発明の一実施形態に係る情報処理装置に適用されるメッセージフォーマットのデータ型定義の一例を示す図である。図５では、utpackが対応するデータ型定義の一例を示す。

【００２７】

図６は、本発明の一実施形態に係る情報処理装置による構造体生成の手順の一例を示すフローチャートである。

まず、メタデータ情報設定部１１はメタデータ設定情報を受け取る（Ｓ１１）。

メタデータ情報設定部１１は、構造体生成部１２にメタデータ設定情報を送る（Ｓ１２）。

メタデータ設定情報に基づき、構造体生成部１２は、生成処理部として、メタデータ生成ならびにシリアライズおよびデシリアライズにそれぞれ使用される構造体を生成する（Ｓ１３）。

【００２８】

図７は、本発明の一実施形態に係る情報処理装置によるデータ処理の手順の一例を示すフローチャートである。

上記のように構造体が生成された後、実データ受信部１３は、情報処理装置１０の外部の装置３１、センサ３２および機器３３等から送信された実データを受信する（Ｓ２１）。

メタデータ生成部１４は、構造体生成部１２により生成された構造体に基づきメタデータを生成する（Ｓ２２）。

【００２９】

構造体生成部１２により生成された構造体に基づき、シリアライズ処理部１５は、実データ受信部１３により受信された実データおよびメタデータ生成部１４により生成されたメタデータのシリアライズを行なう（Ｓ２３）。

データ蓄積部１６は、データ蓄積装置として、シリアライズされたデータを蓄積する（Ｓ２４）。

【００３０】

構造体生成部１２により生成された構造体に基づき、デシリアライズ処理部１７は、データ蓄積部１６から読み出されたデータのデシリアライズ処理を行ない、処理後のデータを情報処理装置１０の外部のデータ処理部２０に渡す（Ｓ２５）。

【００３１】

次に、データ書き込みおよびシリアライズ処理について説明する。

図８は、一般的なシリアライズフォーマットが適用されたときのデータ書き込みの手順の一例を示す図である。このデータ書き込みの手順は以下の（１）～（７）で表される。

（１）　まず、実行時に、シリアライズ処理部１５は、malloc関数により、ヒープ領域に必要なサイズのメモリをアロケートする。

（２）　シリアライズ処理部１５は、スタック領域上の変数に値を代入する。

（３）　シリアライズ処理部１５は、mmap関数などにより、ヒープ上にKeyとValueからなるmap構造を作る。mmap関数が利用される例は、C++のライブラリ（library）、例えば構造体の情報を含むヘッダオンリー（only）ライブラリが用いられたときの一例である。メッセージフォーマットがMessagePackである場合は、Arrayなどの規定された構造が

作られる。
【００３２】
（４）　シリアライズ処理部１５は、append関数などにより、スタック上の変数をヒープ上のmapに割り当てる。この（４）で１回目のメモリコピーがなされる。
（５）　シリアライズ処理部１５は、上記割り当てられた、ヒープ上の構造化されたデータをSerialize等の関数によりコピーし、このデータをヒープ上の連続したメモリ領域にvectorとして並べて直列化する。この（５）で２回目のメモリコピーがなされる。
【００３３】
（６）　シリアライズ処理部１５は、ヒープ上にvectorとしてシリアライズされたデータを、write関数などにより、仮想アドレス空間におけるカーネル空間のメモリのバッファ（buffer）にコピーする。この（６）で３回目のメモリコピーがなされる。
（７）　シリアライズ処理部１５は、カーネル空間のメモリのバッファ上にコピーされた、シリアライズされたデータを、flusher threadによる定期的な書き込みなどによりディスクにコピーする。
【００３４】
これらの（１）～（７）の利点としては、ヒープ上にmap構造、vectorが生成されるため、実行時にKeyとValueの構造が動的に変更され得ることが挙げられる。ただし、シリアライズが行なわれる過程で複数回のメモリコピーが発生する。
【００３５】
図９は、本発明の一実施形態に係る情報処理装置によるデータ書き込みおよびシリアライズ処理の手順の一例を説明する図である。図１０は、本発明の一実施形態に係る情報処理装置によるデータ書き込みおよびシリアライズ処理の手順の一例を示すフローチャートである。
この手順はutpackが適用されたときのデータ書き込みおよびシリアライズ処理の手順であり、以下の（１）～（５）で表される。
【００３６】
（１）　構造体生成部１２により生成された構造体に従って、シリアライズ処理部１５は、コンパイル時に、データが書き込まれる際にスタック上に確保されるメモリ領域の長さ(固定長)を決定する（Ｓ３１）。
【００３７】
（２）　シリアライズ処理部１５は、関数呼び出し時に、連続したメモリ領域をスタック上に確保する（Ｓ３２）。
（３）　構造体生成部１２により生成された構造体に従って、シリアライズ処理部１５は、データの値をスタック上の領域の変数に代入し、スタック上の変数をコピーし、このコピーされた結果をスタック領域上で直列化する(メモリコピー１回目)。
（４）　シリアライズ処理部１５は、この代入された値を、writev関数によりヒープ上の実データと共にカーネルのバッファ上に書き込む(メモリコピー２回目)。この書き込みはヒープを介さずに行われる（Ｓ３３）。
【００３８】
（５）　シリアライズ処理部１５は、カーネルのバッファ上のシリアライズされたデータをflusher threadによる定期的な書き込みなどによりディスクにコピーする（Ｓ３４）。これにより、データのシリアライズおよび書き込みが完了する（Ｓ３５）。
【００３９】
これらの（１）～（５）の利点を説明する。まず、第１の利点を説明する。これらの（１）～（５）では、書き込み対象のデータが、スタックからカーネル空間上にwritev関数により直接書き込まれる。これにより、第１の利点としては、一般的なシリアライズフォーマットが適用されたときと比較して、メモリコピー回数が減り、シリアライズと書き込みが少ないメモリ容量で高速に実施可能であることが挙げられる。
第２の利点としては、構造体により、スタックに確保されるメモリ領域のサイズがコンパイル時に固定長で決定されるため、メモリアロケートを高速化できることが挙げられる。

【００４０】
また、これらの（１）～（５）の制約事項について説明する。第１の制約事項としては、上記の高速な書き込みが行なわれるには、書き込みと読み込みで共通の事前知識（構造体）が必要である(事前知識がない場合は一般的なフローで書き込みがなされる)ことが挙げられる。第２の制約事項としては、コンパイル時に固定長となる変数のみ対応し、実行時の動的なデータ構造の変更に対応しないこととが挙げられる。
【００４１】
次に、データ読み出しおよびデシリアライズ処理について説明する。図１１は、一般的なシリアライズフォーマットが適用されたときのデータ読み出しの手順の一例を示す図である。このデータ読み出しの手順は以下の（１）～（４）で表される。
（１） デシリアライズ処理部１７は、open関数により、ディスク上のデータとカーネル空間上のメモリのページの対応付けを行なう。つまり、ディスク上のデータがページングのsyncによりカーネルのバッファ上にコピーされる。
【００４２】
（２） デシリアライズ処理部１７は、カーネル空間のバッファ上でディスク上のファイルをopen関数により開き、この開いた、シリアライズされたデータをread関数によりヒープ（ユーザ空間）上にコピーする。
（３） デシリアライズ処理部１７は、ヒープ上にコピーされた、シリアライズされたデータのメタデータを読み、メタデータの解釈およびパース（perth）を行なう。
【００４３】
（４） デシリアライズ処理部１７は、ヒープ上にKeyとValueとからなるmap構造を生成し、デシリアライズを完了する。Mapが利用されるのはC++の一例でありMessagePackの場合は、array、mapなどの規定された構造が生成される。
ここまでの過程では、メモリアロケーション、およびカーネル空間とユーザ空間との間のメモリコピーがそれぞれ発生する。
【００４４】
図１２は、本発明の一実施形態に係る情報処理装置によるデータ読み出しおよびデシリアライズ処理の手順の一例を説明する図である。図１３は、本発明の一実施形態に係る情報処理装置によるデータ読み出しおよびデシリアライズ処理の手順の一例を示すフローチャートである。
この手順はutpackが適用されたときのデータの読み出しおよびデシリアライズ処理の手順であり、以下の（１）～（５）で表される。
【００４５】
（１） デシリアライズ処理部１７は、open関数を用いて、ディスク上の読み込み対象であるデータのメモリ領域と、カーネル空間上のメモリ領域とを対応付ける（Ｓ４１）。
（２） デシリアライズ処理部１７は、mmap関数を用いて、カーネル空間上のメモリ領域にあるページ（page）と、ユーザ空間上のメモリ領域にあるページとを対応付ける（Ｓ４２）。
【００４６】
（３） デシリアライズ処理部１７は、ユーザ空間上のメモリを参照する。ページフォルト（fault）が発生し、Ｓ４２で対応付けられた、カーネル空間上のメモリ領域にあるページを参照する（Ｓ４３）。ここではカーネル空間とユーザ空間との間のメモリコピーは行われない。
【００４７】
（４） デシリアライズ処理部１７は、デマンドページングまたはプリページング（prepaging）によりディスク上のデータをカーネルに読み込む（Ｓ４４）。
（５） 構造体生成部１２により生成された構造体に従って、デシリアライズ処理部１７は、ポインタのダイナミックキャストによりアドレスを読み替えることにより、どのアドレスにどのデータが入っているかが分かる。このため、mmap関数が用いられた時点で、実質的なデシリアライズが、カーネル空間とユーザ空間との間におけるゼロ（zero）コピ

ーで完了する（Ｓ４５）。
【００４８】
これらの（１）～（５）の利点を説明する。まず、第１の利点を説明する。これらの（１）～（５）では、構造体がデシリアライズに用いられることで、カーネル空間上のメモリ領域に記録されるデータのどの領域が、どの変数に対応するかが分かる。このため、第１の利点としては、カーネル空間で読み出しが完結することが挙げられる。
第２の利点としては、カーネル空間からユーザ空間のヒープへのメモリコピーおよびヒープ上でのデータのパースが不要であるため、デシリアライズと読み出しが少ないメモリ容量で高速で実行可能であることが挙げられる。
【００４９】
一方、制約事項としては、シリアライズとデシリアライズのプロセスで共通の事前知識(構造体とメッセージフォーマット)を有する必要があることと、事前知識がない場合は一般的なフローにより読み出しがなされることが挙げられる。
【００５０】
次に、複数のデータにおける同じメタデータが参照されるなどのフィルタ（filter）処理について説明する。
上記の一般的なシリアライズフォーマットが適用されたときのデータ読み出しでは、メッセージに含まれるヘッダ内の複数のメタデータの順序が決まっていない場合、各メタデータのデータ形式が読み取られて、目的のメタデータに到達するまでポインタが辿られなければ目的のメタデータおよび次のデータのアドレスが判明しない。このため、複数のデータのフィルタ処理および異なるメタデータの参照処理に時間を要していた。
【００５１】
これに対し、本発明の一実施形態のようにutpackが適用されたときのデータの読み出しでは、構造体により各メタデータの順序が決定され、コンパイル時に先頭アドレスから各メタデータまでの相対アドレスが生成済みである。このため、メタデータが先頭アドレスと組み合わせられることで、上記フィルタ処理が高速に行なわれ得る。
【００５２】
図１４は、一般的なシリアライズフォーマットが適用されたときのフィルタ処理の手順の一例を説明する図である。
ここでは、一般的なシリアライズフォーマットが適用されたときに、複数のセンサデータについて、あるメタデータの値によりフィルタ処理が行なわれる場合について説明する。
【００５３】
この処理では、Key, Valueからなるデータ組の並びが可変であり、パース後、目的のメタデータが参照されるまで各データ組のポインタが辿られる。このため、if文・ポインタの移動回数が比較的多い（図１４の「４」、「１０」および「１６」参照）。
【００５４】
図１５は、本発明の一実施形態に係る情報処理装置によるフィルタ処理の手順の一例を説明する図である。この手順はutpackが適用されたときのフィルタ処理の手順である。
ここでは、utpackが適用されたときに、複数のセンサデータについて、あるメタデータの値によりフィルタ処理が行なわれる場合について説明する。
この処理では、目的のメタデータの位置が判明している（図２，５，８参照）。このため、ポインタの移動回数を、より少なくして目的のデータが参照可能である。
【００５５】
図１６は、本発明の一実施形態に係る情報処理装置のハードウエア構成の一例を示すブロック図である。
図１６に示された例では、上記の実施形態に係る情報処理装置１０は、例えばサーバコンピュータ（server computer）またはパーソナルコンピュータ（personal computer）により構成され、ＣＰＵ等のハードウエアプロセッサ（hardware processor）１１１Ａを有する。そして、このハードウエアプロセッサ１１１Ａに対し、プログラムメモリ（program memory）１１１Ｂ、データメモリ（data memory）１１２、入出力インタフェ

ース１１３及び通信インタフェース１１４が、バス（bus）１２０を介して接続される。データ処理部２０についても同様である。

【００５６】

通信インタフェース１１４は、例えば１つ以上の無線の通信インタフェースユニットを含んでおり、通信ネットワークＮＷとの間で情報の送受信を可能にする。無線インタフェースとしては、例えば無線ＬＡＮ（Local Area Network）などの小電力無線データ通信規格が採用されたインタフェースが使用される。

【００５７】

入出力インタフェース１１３には、情報処理装置１０に付設される、オペレータ（operator）用の入力デバイス５０および出力デバイス６０が接続される。

入出力インタフェース１１３は、キーボード、タッチパネル（touch panel）、タッチパッド（touchpad）、マウス（mouse）等の入力デバイス５０を通じてオペレータにより入力された操作データを取り込むとともに、出力データを液晶または有機ＥＬ（Electro Luminescence）等が用いられた表示デバイスを含む出力デバイス６０へ出力して表示させる処理を行なう。なお、入力デバイス５０および出力デバイス６０には、情報処理装置１０に内蔵されたデバイスが使用されてもよく、また、ネットワーク（network）ＮＷを介して情報処理装置１０と通信可能である他の情報端末の入力デバイスおよび出力デバイスが使用されてもよい。

【００５８】

プログラムメモリ１１１Ｂは、非一時的な有形の記憶媒体として、例えば、ＨＤＤ（Hard Disk Drive）またはＳＳＤ（Solid State Drive）等の随時書込みおよび読出しが可能な不揮発性メモリ（non-volatile memory）と、ＲＯＭ（Read Only Memory）等の不揮発性メモリとが組み合わせて使用されたもので、一実施形態に係る各種制御処理を実行する為に必要なプログラムが格納されている。

【００５９】

データメモリ１１２は、有形の記憶媒体として、例えば、上記の不揮発性メモリと、ＲＡＭ（Random Access Memory）等の揮発性メモリ（volatile memory）とが組み合わせて使用されたもので、各種処理が行なわれる過程で取得および作成された各種データが記憶される為に用いられる。

【００６０】

本発明の一実施形態に係る情報処理装置１０は、ソフトウエア（software）による処理機能部として、図１に示されるメタデータ情報設定部１１、構造体生成部１２、実データ受信部１３、メタデータ生成部１４、シリアライズ処理部１５、データ蓄積部１６およびデシリアライズ処理部１７を有するデータ処理装置として構成され得る。

【００６１】

データ処理装置におけるデータ記憶領域およびデータ処理領域は、図１６に示されたデータメモリ１１２が用いられることで構成され得る。ただし、これらの領域は情報処理装置１０内に必須の構成ではなく、例えば、ＵＳＢ（Universal Serial Bus）メモリなどの外付け記憶媒体、又はクラウド（cloud）に配置されたデータベースサーバ（database server）等の記憶装置に設けられた領域であってもよい。

【００６２】

上記のメタデータ情報設定部１１、構造体生成部１２、実データ受信部１３、メタデータ生成部１４、シリアライズ処理部１５、データ蓄積部１６およびデシリアライズ処理部１７の各部における処理機能部は、いずれも、プログラムメモリ１１１Ｂに格納されたプログラムを上記ハードウエアプロセッサ１１１Ａにより読み出させて実行させることにより実現され得る。なお、これらの処理機能部の一部または全部は、特定用途向け集積回路（ＡＳＩＣ（Application Specific Integrated Circuit））またはＦＰＧＡ（Field-Programmable Gate Array）などの集積回路を含む、他の多様な形式によって実現されてもよい。

【００６３】

以上説明したように、本発明の一実施形態では、データのシリアライズ処理、デシリアライズ処理が少ないメモリ容量で高速に実施可能であり、デバイスのリソースに制約のあるエッジ装置においても大量のIoTデータを高速に処理することができる。

【００６４】

この発明の情報処理装置の第１の態様は、情報処理装置が、シリアライズ処理に割り当てられる仮想アドレス空間のうちスタック領域に確保されるメモリ領域の長さがコンパイル時に決定されるメッセージ構造が示される情報を生成する生成処理部と、前記生成処理部により生成された情報を用いて、機器からのデータに対するシリアライズ処理を行なってデータ蓄積装置に書き込むシリアライズ処理を行なうシリアライズ処理部と、前記データ蓄積装置に書き込まれた、前記シリアライズ処理されたデータを読み出し、前記生成処理により生成された情報を用いて、前記読み出されたデータに対するデシリアライズ処理部とを備える。

【００６５】

この発明の情報処理装置の第２の態様は、第１の態様において、前記シリアライズ処理部は、前記生成処理部により生成された情報のコンパイル時に、連続したメモリ領域を前記スタック領域にアロケートし、書き込み対象となる前記シリアライズ処理されたデータを、前記アロケートされたメモリ領域の変数に代入し、前記代入されたデータを前記仮想アドレス空間のカーネル空間のメモリ領域に書き込む。

【００６６】

この発明の情報処理装置の第３の態様は、第１の態様において、前記生成処理部により生成された情報は、読み出し対象のデータへのポインタを含み、前記デシリアライズ処理部は、前記読み出し対象のデータが書き込まれた物理アドレス空間のメモリ領域と、前記仮想アドレス空間のカーネル空間のメモリ領域との対応付けを行ない、前記カーネル空間のメモリ領域と、前記仮想アドレス空間のユーザ空間のメモリ領域との対応付けを行ない、前記ユーザ空間のメモリ領域を参照し、前記物理アドレス空間のメモリ領域に書き込まれているデータを前記対応付けられるカーネル空間のメモリ領域に読み出し、前記生成処理部により生成された情報に含まれるポインタのダイナミックキャストにより前記カーネル空間のメモリ領域と前記ユーザ空間のメモリ領域との間のアドレスの読み替えを行なうことで、前記読み出し対象のデータのデシリアライズ処理を行なう。

【００６７】

この発明の情報処理装置の第４の態様は、第１の態様において、前記生成処理部により生成された情報は、実データ、および所定の順序の複数のメタデータに関する情報を含む。

【００６８】

本発明の一実施形態に係る情報処理プログラムの一つの態様は、第１乃至第４の態様のいずれか１つにおける情報処理装置の前記各部としてプロセッサを機能させるものである。

【００６９】

この発明の一実施形態に係る情報処理装置の第１の態様によれば、シリアライズ処理に割り当てられる仮想アドレス空間のうちスタック領域に確保されるメモリ領域の長さがコンパイル時に決定されるメッセージ構造を示す情報を用いて機器からのデータに対するシリアライズ処理およびデシリアライズ処理を行なうので、データ書き込みおよびデータ読み出しを少ないメモリ容量で高速に行なうことができる。

【００７０】

この発明の一実施形態に係る情報処理装置の第２の態様によれば、スタック領域に連続したメモリ領域をアロケートし、アロケートされたメモリ領域の変数に代入された書き込み対象となるシリアライズ処理されたデータがカーネル空間のメモリ領域に書き込まれるので、データ書き込みの際のデータコピー回数を減らすことができる。

【００７１】

この発明の一実施形態に係る情報処理装置の第３の態様によれば、読み込み対象のデータが書き込まれた物理アドレス空間のメモリ領域と、カーネル空間のメモリ領域との対応付け、およびカーネル空間のメモリ領域と、ユーザ空間のメモリ領域との対応付けを行ない

、物理アドレス空間のメモリ領域に書き込まれているデータをカーネル空間のメモリ領域に読み込み、生成された情報に含まれるポインタを用いてユーザ空間とカーネル空間との間でアドレスの読み替えを行なうので、ユーザ空間とカーネル空間との間のメモリコピーを行なわずにデシリアライズ処理を行なうことができる。

【００７２】

この発明の一実施形態に係る情報処理装置の第４の態様によれば、生成処理により生成された情報は、実データ、および所定の順序の複数のメタデータに関する情報を含むので、複数のデータにおける同じメタデータを参照する処理を高速に行なうことができる。

【００７３】

また、各実施形態に記載された手法は、計算機（コンピュータ）にインストール可能な形式のファイルまたは実行可能な形式のファイルとしてのプログラム（ソフトウエア手段）として、例えば磁気ディスク（フロッピー（登録商標）ディスク（Floppy disk）、ハードディスク等）、光ディスク（ＣＤ－ＲＯＭ、ＣＤ－Ｒ、ＤＶＤ－ＲＯＭ、ＤＶＤ－Ｒ、光磁気ディスク（ＭＯ）等）、半導体メモリ（ＲＯＭ、ＲＡＭ、フラッシュメモリ（Flash memory）等）等の、コンピュータで読み取り可能な記録媒体（または記憶媒体）に格納され、また通信媒体により伝送されることで頒布され得る。なお、媒体に格納されるプログラムは、計算機に実行させるソフトウエア手段（実行プログラムのみならずテーブル、データ構造も含む）を計算機内に構成させる設定プログラムをも含む。本装置を実現する計算機は、記録媒体に記録されたプログラムを読み込み、また場合により設定プログラムによりソフトウエア手段を構築し、このソフトウエア手段によって動作が制御されることにより上述した処理を実行する。なお、本明細書でいう記録媒体または記憶媒体は、頒布用に限らず、計算機内部あるいはネットワークを介して接続される機器に設けられた磁気ディスク、半導体メモリ等の記憶媒体または記憶媒体を含むものである。また、上記処理を実現するプログラムを、インターネットなどのネットワークに接続されたコンピュータ（サーバ）上に格納し、ネットワーク経由でコンピュータ（クライアント）にダウンロードさせてもよい。

【００７４】

なお、本発明は、上記実施形態に限定されるものではなく、実施段階ではその要旨を逸脱しない範囲で種々に変形することが可能である。また、各実施形態は適宜組み合わせて実施してもよく、その場合組み合わせた効果が得られる。更に、上記実施形態には種々の発明が含まれており、開示される複数の構成要件から選択された組み合わせにより種々の発明が抽出され得る。例えば、実施形態に示される全構成要件からいくつかの構成要件が削除されても、課題が解決でき、効果が得られる場合には、この構成要件が削除された構成が発明として抽出され得る。

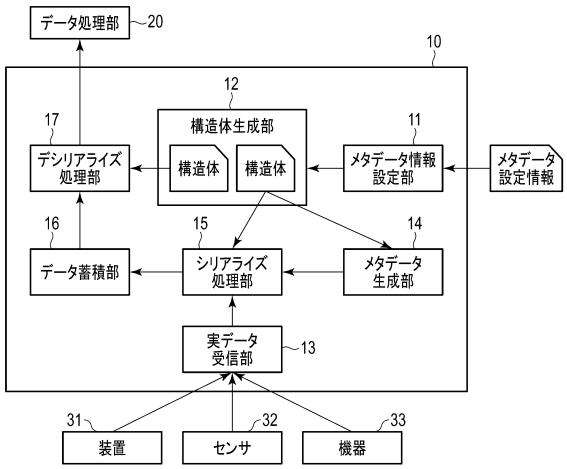
【符号の説明】

【００７５】

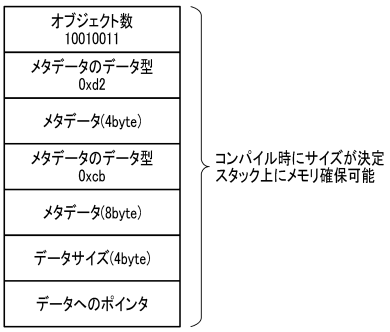
１０…情報処理装置
１１…メタデータ情報設定部
１２…構造体生成部
１３…実データ受信部
１４…メタデータ生成部
１５…シリアライズ処理部
１６…データ蓄積部
１７…デシリアライズ処理部
２０…データ処理部
３１…装置
３２…センサ
３３…機器

【図面】

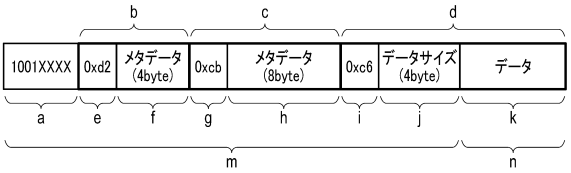
【図１】


データ処理部
20
10
12
構造体生成部
構造体
構造体
17
デシリアライズ処理部
11
メタデータ情報設定部
メタデータ設定情報
16
データ蓄積部
15
シリアライズ処理部
14
メタデータ生成部
実データ受信部
13
31
装置
32
センサ
33
機器

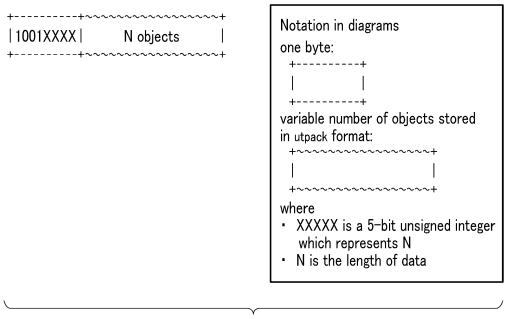
【図２】


オブジェクト数
10010011
メタデータのデータ型
0xd2
メタデータ(4byte)
メタデータのデータ型
0xcb
メタデータ(8byte)
データサイズ(4byte)
データへのポインタ
コンパイル時にサイズが決定
スタック上にメモリ確保可能

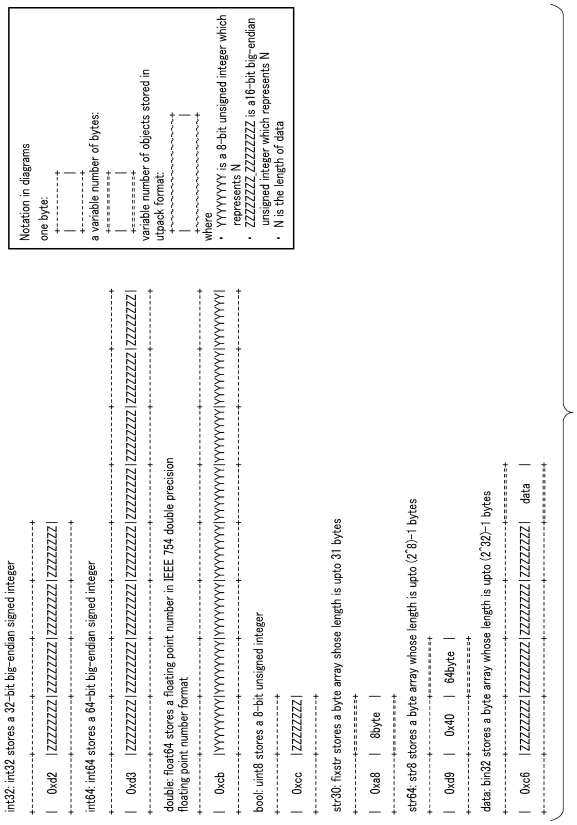
【図３】


b
c
d
1001XXXX
0xd2
メタデータ (4byte)
0xcb
メタデータ (8byte)
0xc6
データサイズ (4byte)
データ
a
e
f
g
h
i
j
k
m
n

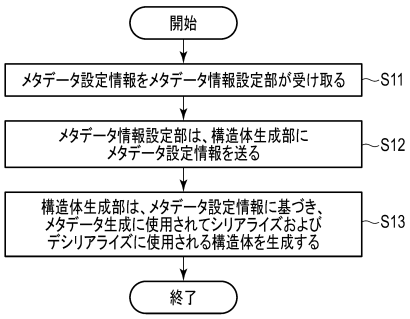
【図４】


1001XXXX
N objects
Notation in diagrams
one byte:
variable number of objects stored in utpack format:
where
・ XXXXX is a 5-bit unsigned integer which represents N
・ N is the length of data

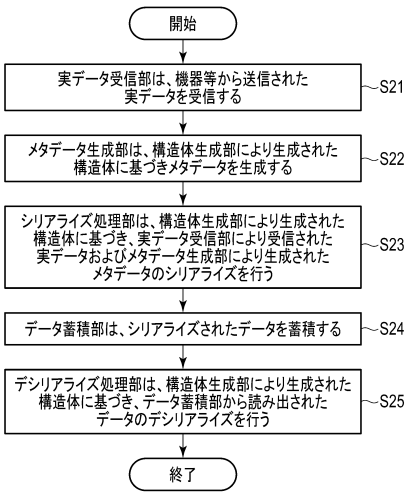
【図５】


Notation in diagrams
one byte:
a variable number of bytes:
variable number of objects stored in
utpack format:
where
• YYYYYYYY is a 8-bit unsigned integer which represents N
• ZZZZZZZZ ZZZZZZZZ is a16-bit big-endian unsigned integer which represents N
• N is the length of data
int32: int32 stores a 32-bit big-endian signed integer
0xd2
int64: int64 stores a 64-bit big-endian signed integer
0xd3
double: float64 stores a floating point number in IEEE 754 double precision floating point number format
0xcb
bool: uint8 stores a 8-bit unsigned integer
0xcc
str30: fixstr stores a byte array shose length is upto 31 bytes
0xa8
8byte
str64: str8 stores a byte array whose length is upto (2^8)−1 bytes
0xd9
0x40
64byte
data: bin32 stores a byte array whose length is upto (2^32)−1 bytes
0xc6
data

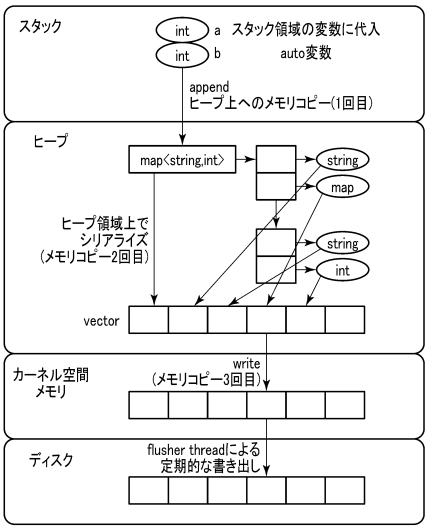
【図６】


開始
メタデータ設定情報をメタデータ情報設定部が受け取る
S11
メタデータ情報設定部は、構造体生成部に
メタデータ設定情報を送る
S12
構造体生成部は、メタデータ設定情報に基づき、
メタデータ生成に使用されてシリアライズおよび
デシリアライズに使用される構造体を生成する
S13
終了

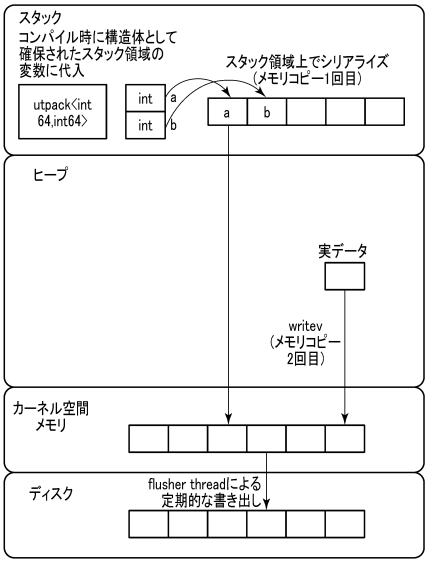
【図７】


開始
実データ受信部は、機器等から送信された
実データを受信する
S21
メタデータ生成部は、構造体生成部により生成された
構造体に基づきメタデータを生成する
S22
シリアライズ処理部は、構造体生成部により生成された
構造体に基づき、実データ受信部により受信された
実データおよびメタデータ生成部により生成された
メタデータのシリアライズを行う
S23
データ蓄積部は、シリアライズされたデータを蓄積する
S24
デシリアライズ処理部は、構造体生成部により生成された
構造体に基づき、データ蓄積部から読み出された
データのデシリアライズを行う
S25
終了

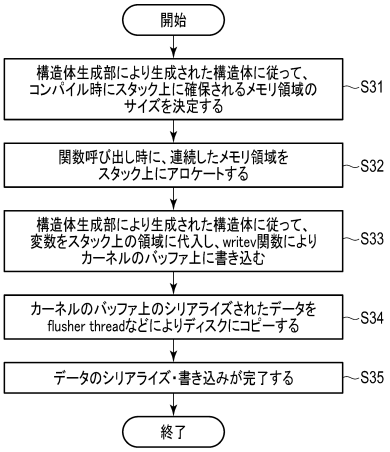
【図８】


スタック
int a スタック領域の変数に代入
int b auto変数
append
ヒープ上へのメモリコピー(1回目)
ヒープ
map<string,int>
string
map
ヒープ領域上で
シリアライズ
(メモリコピー2回目)
string
int
vector
write
(メモリコピー3回目)
カーネル空間
メモリ
ディスク
flusher threadによる
定期的な書き出し

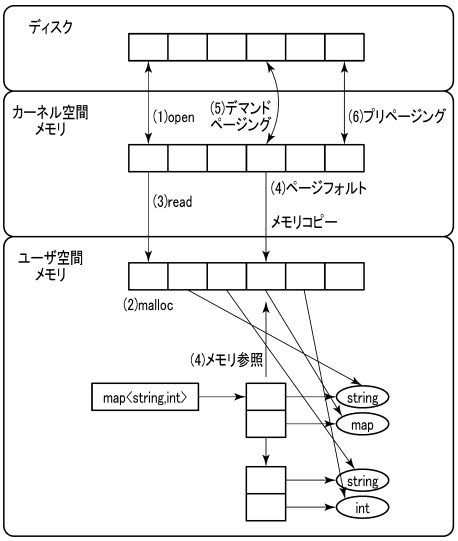
【図９】



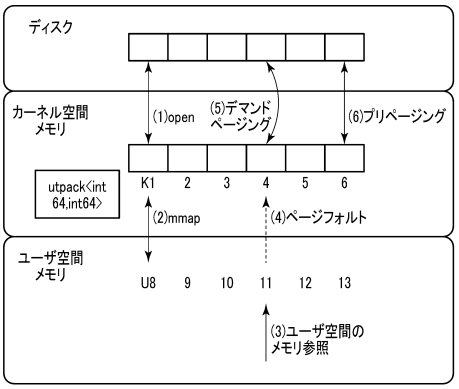

【図１０】



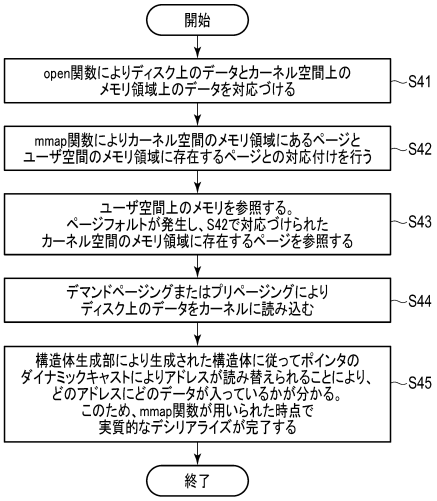

【図１１】



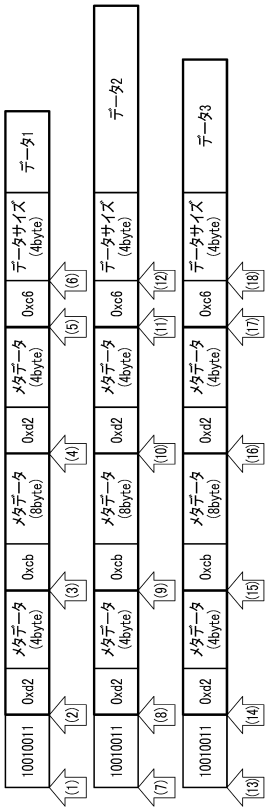

【図１２】



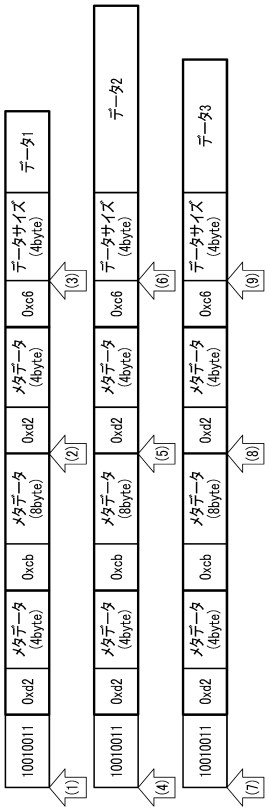
【図１３】



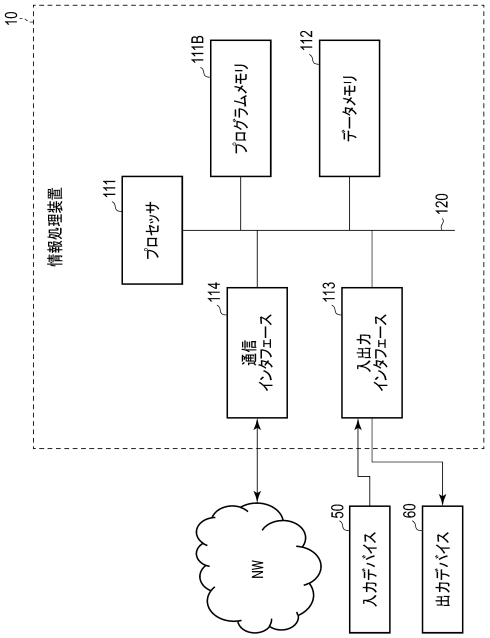

【図１４】




【図１５】




【図１６】



フロントページの続き

(72)発明者　齋藤 由唯
東京都千代田区大手町一丁目５番１号 日本電信電話株式会社内
審査官　原 秀人

(56)参考文献　国際公開第２０１８／０８８４９０（WO，Ａ１）
特表２００５－５１９３７６（ＪＰ，Ａ）
鶴岡 邦敏 外，オブジェクト指向ＤＢＭＳのアーキテクチャに関する考察，情報処理学会研究報告，日本，社団法人情報処理学会，1993年07月23日，Ｖｏｌ．９３ Ｎｏ．６５，pp. 109--116
まつもと ゆきひろ，まつもとゆきひろ 技術を斬る シリアライズ，日経Ｌｉｎｕｘ，日本，日経ＢＰ社，2009年12月08日，第１２巻 第１号，pp. 147--155

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１６／００－１６／９５８
Ｇ０６Ｆ　９／４４８