

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7186541号
(P7186541)

(45)発行日 令和4年12月9日(2022.12.9)

(24)登録日 令和4年12月1日(2022.12.1)

(51)国際特許分類

F I

H 0 4 N 1/00 (2006.01)

H 0 4 N 1/00 0 0 2 A

G 0 6 F 3/12 (2006.01)

G 0 6 F 3/12 3 3 0

G 0 6 F 3/12 3 2 0

請求項の数 14 (全23頁)

(21)出願番号 特願2018-150641(P2018-150641)
 (22)出願日 平成30年8月9日(2018.8.9)
 (65)公開番号 特開2019-204477(P2019-204477 A)
 (43)公開日 令和1年11月28日(2019.11.28)
 審査請求日 令和3年7月30日(2021.7.30)
 (31)優先権主張番号 特願2018-95632(P2018-95632)
 (32)優先日 平成30年5月17日(2018.5.17)
 (33)優先権主張国・地域又は機関
 日本国(JP)

(73)特許権者 000001007
 キヤノン株式会社
 東京都大田区下丸子3丁目30番2号
 (74)代理人 110003281
 弁理士法人大塚国際特許事務所
 (72)発明者 佐々木 祐人
 東京都大田区下丸子3丁目30番2号
 キヤノン株式会社内
 審査官 征矢 崇

最終頁に続く

(54)【発明の名称】 画像処理装置、その制御方法、及びプログラム

(57)【特許請求の範囲】

【請求項1】

1 以上の画像処理モジュールがブレインストールされた画像処理装置であって、アプリケーションによる要求に応じて第一の画像処理モジュールで画像処理を実行中に、アプリケーションによる新たな画像処理が要求されると、前記新たな画像処理を実行可能な第二の画像処理モジュールを探索する探索手段と、

前記探索手段によって探索された第二の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量、及び前記第一の画像処理モジュールによって既に実行中の画像処理のメモリ使用量を取得する取得手段と、

前記取得手段によって取得された各画像処理のメモリ使用量及び前記新たな画像処理を実行した際のメモリ使用量から得られる総メモリ使用量と、画像処理モジュールが動作するプロセスでの使用が許容されるメモリ使用量を示す許容メモリ量とを比較する比較手段と、

前記比較手段によって比較した結果、前記総メモリ使用量が前記許容メモリ量を超えない場合は前記新たな画像処理を実行し、前記総メモリ使用量が前記許容メモリ量を超える場合は前記新たな画像処理を実行せず、要求元の動作中のアプリケーションに対してメモリ不足により前記新たな画像処理が実行できない旨のエラーを通知する実行制御手段とを備えることを特徴とする画像処理装置。

【請求項2】

前記第一の画像処理モジュールによる画像処理を実行中に、アプリケーションによる新

10

20

たな画像処理が要求されると、前記第一の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量と、前記第一の画像処理モジュールによって既に実行中の画像処理のそれぞれのメモリ使用量とから並列処理が実行可能か否かを判断する判断手段と、

前記判断手段によって判断した結果、前記並列処理が実行可能でない場合は前記新たな画像処理と、前記既に実行中の画像処理を順次実行し、前記並列処理が実行可能な場合は前記新たな画像処理と前記既に実行中の画像処理とを並行し実行する制御手段とをさらに備えることを特徴とする請求項 1 に記載の画像処理装置。

【請求項 3】

前記判断手段は、画像処理のページごとに並列処理が可能か否かを動的に判断することを特徴とする請求項 2 に記載の画像処理装置。

【請求項 4】

前記制御手段は、画像処理を実行するための実行メモリを確保できない場合には、画像処理が実行できない旨のエラーを通知することを特徴とする請求項 2 又は 3 に記載の画像処理装置。

【請求項 5】

前記実行制御手段は、前記メモリ不足により前記新たな画像処理が実行できない場合は、所定時間が経過した後に、再度、前記新たな画像処理が実行可能か判断することを特徴とする請求項 1 乃至 4 の何れか 1 項に記載の画像処理装置。

【請求項 6】

前記要求元のアプリケーションは、前記新たな画像処理が実行される画像処理モジュールのプロセスとは異なるプロセスで動作していることを特徴とする請求項 1 に記載の画像処理装置。

【請求項 7】

画像処理装置であって、
第一の実行環境で動作中のアプリケーションからの画像処理の要求に応じて、前記第一の実行環境とは論理メモリ空間が異なる第二の実行環境で動作する第一の画像処理モジュールでの画像処理の実行中に、前記アプリケーションによる新たな画像処理の実行が要求されると、該新たな画像処理を実行可能な第二の画像処理モジュールを探索する探索手段と、

前記探索手段によって探索された前記第二の画像処理モジュールとともに保持されている、該第二の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量と、前記第二の実行環境で既に実行中の画像処理のそれぞれのメモリ使用量とを取得する取得手段と、

前記取得手段によって取得された各画像処理のメモリ使用量及び前記要求された新たな画像処理を実行した際のメモリ使用量から得られる総メモリ使用量と、前記第二の実行環境での使用が許容されるメモリ使用量を示す許容メモリ量とを比較する比較手段と、

前記比較手段による比較の結果、前記総メモリ使用量が前記許容メモリ量を超えない場合は前記新たな画像処理を実行し、前記総メモリ使用量が前記許容メモリ量を超える場合は前記新たな画像処理を実行せず、要求元の前記第一の実行環境で動作中のアプリケーションに対してメモリ不足により前記新たな画像処理が実行できない旨のエラーを通知する実行制御手段と、

を備えることを特徴とする画像処理装置。

【請求項 8】

前記第二の実行環境での前記第一の画像処理モジュールによる画像処理の実行中に、前記第一の実行環境で動作しているアプリケーションによる新たな画像処理が要求されると、該第一の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量と、前記第二の実行環境で既に実行中の画像処理のそれぞれのメモリ使用量とから並列処理が実行可能か否かを判断する判断手段と、

前記判断手段による判断の結果、前記並列処理が実行可能でない場合は前記新たな画像処理と、前記既に実行中の画像処理を順次実行し、前記並列処理が実行可能な場合は前記新たな画像処理と前記既に実行中の画像処理とを並行し実行する制御手段と

10

20

30

40

50

をさらに備えることを特徴とする請求項 7 に記載の画像処理装置。

【請求項 9】

前記判断手段は、画像処理のページごとに並列処理が可能か否かを動的に判断することを特徴とする請求項 8 に記載の画像処理装置。

【請求項 10】

前記制御手段は、画像処理を実行するための実行メモリを確保できない場合には、画像処理が実行できない旨のエラーを通知することを特徴とする請求項 8 又は 9 に記載の画像処理装置。

【請求項 11】

前記実行制御手段は、前記メモリ不足により前記新たな画像処理が実行できない場合は、所定時間が経過した後に、再度、前記新たな画像処理が実行可能か判断することを特徴とする請求項 7 乃至 10 の何れか 1 項に記載の画像処理装置。

【請求項 12】

画像処理を実行する際の前記メモリ使用量は、該画像処理に含まれる関数ごとに保持されていることを特徴とする請求項 1 乃至 11 の何れか 1 項に記載の画像処理装置。

【請求項 13】

アプリケーションによる要求に応じて第一の画像処理モジュールで画像処理を実行中に、アプリケーションによる新たな画像処理が要求されると、前記新たな画像処理を実行可能な第二の画像処理モジュールを探索する探索ステップと、

前記探索ステップによって探索された第二の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量、及び前記第一の画像処理モジュールによって既に実行中の画像処理のメモリ使用量を取得する取得ステップと、

前記取得ステップによって取得された各画像処理のメモリ使用量及び前記新たな画像処理を実行した際のメモリ使用量から得られる総メモリ使用量と、画像処理モジュールが動作するプロセスでの使用が許容されるメモリ使用量を示す許容メモリ量とを比較する比較ステップと、

前記比較ステップによって比較した結果、前記総メモリ使用量が前記許容メモリ量を超えない場合は前記新たな画像処理を実行し、前記総メモリ使用量が前記許容メモリ量を超える場合は前記新たな画像処理を実行せず、要求元の動作中のアプリケーションに対してメモリ不足により前記新たな画像処理が実行できない旨のエラーを通知する実行制御ステップと、

を備えることを特徴とする画像処理装置の制御方法。

【請求項 14】

コンピュータに、

アプリケーションによる要求に応じて第一の画像処理モジュールで画像処理を実行中に、アプリケーションによる新たな画像処理が要求されると、前記新たな画像処理を実行可能な第二の画像処理モジュールを探索する探索手段と、

前記探索手段によって探索された第二の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量、及び前記第一の画像処理モジュールによって既に実行中の画像処理のメモリ使用量を取得する取得手段と、

前記取得手段によって取得された各画像処理のメモリ使用量及び前記新たな画像処理を実行した際のメモリ使用量から得られる総メモリ使用量と、画像処理モジュールが動作するプロセスでの使用が許容されるメモリ使用量を示す許容メモリ量とを比較する比較手段と、

前記比較手段によって比較した結果、前記総メモリ使用量が前記許容メモリ量を超えない場合は前記新たな画像処理を実行し、前記総メモリ使用量が前記許容メモリ量を超える場合は前記新たな画像処理を実行せず、要求元の動作中のアプリケーションに対してメモリ不足により前記新たな画像処理が実行できない旨のエラーを通知する実行制御手段とを備えることを特徴とする画像処理装置として動作させるためのプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、画像処理装置、その制御方法、及びプログラムに関する。

【背景技術】

【0002】

従来から、画像処理装置で提供されていない画像処理を含む各種ジョブを実行したい場合、新規画像処理を含むジョブ実行アプリケーションをプラグインとして作成し、当該画像処理装置にインストールして実現することが知られている。ここで、各種ジョブとは、例えば、コピー、プリント、スキャン保存、及びスキャン送信といった、画像処理装置上で実行可能な画像処理を含むデバイス制御処理を表す。また、新規画像処理とは、例えば、入力データに対する斜行補正、フォーマット変換、OCR (optical character recognition) 後の翻訳変換といった、画像処理装置の既存のジョブでは実現できない処理を表す。

10

【0003】

複数の新規画像処理を追加した画像処理装置では、追加した新規画像処理を並列動作させることが可能である。しかし、メモリを多く利用する複数の画像処理が同時に実行されると、画像処理装置の保有するメモリの上限を超えてしまう場合がある。

【0004】

メモリの上限を超えてしまうと、OSは、使用頻度の低いメモリ領域をハードディスクに一旦退避して領域を空け、必要に応じてまたメモリに書き戻すswap操作を行う。この際、HDDに書き込み処理を行うため、同プロセス内の様々な処理のパフォーマンスを低下させる場合がある。例えばUIモジュールが同プロセスに存在する場合、ユーザの操作に対してUI画面がすぐに反応しない現象が発生しうる。また、同プロセス上で実行している複数の画像処理の完了までの時間が、swapしていない場合と比較して遅くなる現象も発生しうる。さらに、Swap領域までも使い果たしてしまった場合には、OSがメモリリソースを多く消費しているプロセスを強制終了させてしまう。この場合、画像処理を行うプロセスが終了してしまうため、画像処理装置を再起動しないと画像処理を再開することはできない。

20

【0005】

そこで、特許文献1には、個々のプラグインで使用されるメモリ使用量をリアルタイムで計測し、予め決められたメモリ最大量を上回る場合はプラグインを起動させない技術が提案されている。

30

【先行技術文献】

【特許文献】

【0006】

【文献】特開2013-0120103号公報

【発明の概要】

【発明が解決しようとする課題】

【0007】

しかしながら、上記従来技術には以下に記載する課題がある。上記従来技術では、例えば、プラグイン起動時に各プラグインに定義された使用メモリ宣言量と、既に画像処理装置で利用しているメモリ使用量の合計を計算する。さらに、予め決められたメモリ最大量を合計のメモリ使用量が上回る場合は、プラグイン自体を起動させないことによって、上述のようなメモリ不足を回避している。上記制御では、メモリ使用量が上回る計算となった場合、既に起動しているプラグインを停止させない限り、新規のプラグインを起動することができない。したがって、実際には並列で実行しなければ実行できるプラグインであっても起動して処理を行うことができず、ユーザの利便性が低下するといった課題が存在する。

40

【0008】

本発明は、上述の問題の少なくとも一つに鑑みて成されたものであり、処理実行中に（

50

ランタイムで) 総メモリ使用量を動的に計算し、プラグイン自体の起動を制限することなく、好適にメモリ不足を回避しつつ、複数の画像処理を並行して実行する仕組み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】

本発明は、例えば、1以上の画像処理モジュールがプレインストールされた画像処理装置であって、アプリケーションによる要求に応じて第一の画像処理モジュールで画像処理を実行中に、アプリケーションによる新たな画像処理が要求されると、前記新たな画像処理を実行可能な第二の画像処理モジュールを探索する探索手段と、前記探索手段によって探索された第二の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量、及び前記第一の画像処理モジュールによって既に実行中の画像処理のメモリ使用量を取得する取得手段と、前記取得手段によって取得された各画像処理のメモリ使用量及び前記新たな画像処理を実行した際のメモリ使用量から得られる総メモリ使用量と、画像処理モジュールが動作するプロセスでの使用が許容されるメモリ使用量を示す許容メモリ量とを比較する比較手段と、前記比較手段によって比較した結果、前記総メモリ使用量が前記許容メモリ量を超えない場合は前記新たな画像処理を実行し、前記総メモリ使用量が前記許容メモリ量を超える場合は前記新たな画像処理を実行せず、要求元の動作中のアプリケーションに対してメモリ不足により前記新たな画像処理が実行できない旨のエラーを通知する実行制御手段とを備えることを特徴とする。

【0010】

また、本発明は、例えば、画像処理装置であって、第一の実行環境で動作中のアプリケーションからの画像処理の要求に応じて、前記第一の実行環境とは論理メモリ空間が異なる第二の実行環境で動作する第一の画像処理モジュールでの画像処理の実行中に、前記アプリケーションによる新たな画像処理の実行が要求されると、該新たな画像処理を実行可能な第二の画像処理モジュールを探索する探索手段と、前記探索手段によって探索された前記第二の画像処理モジュールとともに保持されている、該第二の画像処理モジュールによる新たな画像処理を実行した際のメモリ使用量と、前記第二の実行環境で既に実行中の画像処理のそれぞれのメモリ使用量とを取得する取得手段と、前記取得手段によって取得された各画像処理のメモリ使用量及び前記要求された新たな画像処理を実行した際のメモリ使用量から得られる総メモリ使用量と、前記第二の実行環境での使用が許容されるメモリ使用量を示す許容メモリ量とを比較する比較手段と、前記比較手段による比較の結果、前記総メモリ使用量が前記許容メモリ量を超えない場合は前記新たな画像処理を実行し、前記総メモリ使用量が前記許容メモリ量を超える場合は前記新たな画像処理を実行せず、要求元の前記第一の実行環境で動作中のアプリケーションに対してメモリ不足により前記新たな画像処理が実行できない旨のエラーを通知する実行制御手段と、を備えることを特徴とする。

【発明の効果】

【0011】

本発明によれば、処理実行中に総メモリ使用量を動的に計算し、プラグイン自体の起動を制限することなく、好適にメモリ不足を回避しつつ、複数の画像処理を並行して実行することができる。

【図面の簡単な説明】

【0012】

【図1】一実施形態に係る画像処理形成システムの構成例を表す図。

【図2】一実施形態に係る画像処理装置のハードウェア構成の一例を表す図。

【図3】一実施形態に係る画像処理装置のソフトウェア構成の一例を示す図。

【図4】一実施形態に係る画像処理装置の処理手順を示すフローチャート。

【図5】一実施形態に係るNative画像処理サーバの処理手順を示すフローチャート。

【図6】一実施形態に係る画像処理装置のソフトウェア構成の一例を示す図。

【図7】一実施形態に係る画像処理装置の処理手順を示すフローチャート。

【図 8】一実施形態に係る Native 画像処理モジュールの処理手順を示すフローチャート。

【図 9】一実施形態に係る Native 画像処理サーバの処理手順を示すフローチャート。

【図 10】一実施形態に係る Native 画像処理サーバの処理手順を示すフローチャート。

【図 11】一実施形態に係る Native 画像処理モジュールの処理手順を示すフローチャート。

【発明を実施するための形態】

【0013】

以下に本発明の一実施形態を示す。以下で説明される個別の実施形態は、本発明の上位概念、中位概念及び下位概念など種々の概念を理解するために役立つであろう。また、本発明の技術的範囲は、特許請求の範囲によって確立されるのであって、以下の個別の実施形態によって限定されるわけではない。なお、実施形態に係る印刷装置として複合機（デジタル複合機／MFP／Multi Function Peripheral）を例に説明する。しかしながら適用範囲は複合機に限定はせず、印刷装置であればよい。

【0014】

<第 1 の実施形態>

<システム構成>

以下では、添付図面を参照して、本発明の第 1 の実施形態について説明する。まず、図 1 を参照して、画像形成システムの構成の一例を説明する。本実施形態に係る画像形成システムは、画像処理装置 101、102 と、情報処理端末 103、104 と、サーバ 105 とを含んで構成される。画像処理装置 101、102 と、情報処理端末 103、104 と、サーバ 105 とは、ネットワーク 106 により相互に通信可能に接続される。

【0015】

図 1 では、画像処理装置 101、102 が 2 台である場合を一例として示すが、画像処理装置の数は任意（1 つ又は 2 以上）である。また、それぞれの画像処理装置 101、102 は、同様の構成で実現することができる。従って、以下では、画像処理装置 101、102 を代表して画像処理装置 101 の構成について説明し、画像処理装置 102 についての詳細な説明を省略する。なお、ネットワーク 106 は、LAN（Local Area Network）、Internet 等、画像形成システム内の装置が相互に通信できるものであればよい。また、情報処理端末やサーバについてもその台数や機能を限定する意図はない。サーバ 105 については、その機能を分散して複数台の装置で実現されてもよい。なお、本発明を適用する画像処理装置は、画像形成機能を必ずしも含む必要はなく、複数の画像処理を同時に（並行して）実行可能な装置であれば適用することができる。

【0016】

画像処理装置 101 は、情報処理端末 103、104 から画像データの印刷依頼（印刷データ）を受信して印刷することや、画像処理装置 101 に備わるスキャナで画像データを読み取ることや、スキャナで読み取られた画像データを印刷することが可能である。また、画像処理装置 101 は、情報処理端末 103、104 から受信した印刷データを保存したり、画像処理装置 101 のスキャナで読み取られた画像を情報処理端末 103、104 に送信したりすることが可能である。更に画像処理装置 101 は、サーバ 105 を用いて画像処理を行ったり、サーバ 105 に格納されている文書を印刷したりすることが可能である。画像処理装置 101 は、この他に、MFP（Multi function Peripheral）等の公知の画像処理装置が有する機能を実現することが可能である。

【0017】

<画像処理装置のハードウェア構成>

次に、図 2 を参照して、画像処理装置 101 のハードウェア構成の一例を説明する。画像処理装置 101 は、コントローラ 201、プリンタ 202、スキャナ 203、及び操作部 204 を備える。コントローラ 201 は、CPU 211、RAM 212、HDD 213、ネットワーク I/F 214、プリンタ I/F 215、スキャナ I/F 216、操作部 I

10

20

30

40

50

／F217、及び拡張I／F218を備える。CPU211は、RAM212、HDD213、ネットワークI／F214、プリンタI／F215、スキャナI／F216、操作部I／F217、及び拡張I／F218とデータを授受することが可能である。

【0018】

CPU211は、HDD213から読み出した命令をRAM212に展開し、RAM212に展開した命令を実行し、後述する各種処理を実行する。HDD213には、CPU211で実行可能な命令、画像処理装置101で使用する設定値、及びユーザから依頼された処理に関するデータ等を記憶しておくことが可能である。RAM212は、CPU211がHDD213から読み出した命令を一時的に格納するための領域である。また、RAM212は、命令の実行に必要な各種のデータを記憶しておくことも可能である。例えば画像処理では、入稿されたデータをRAM212に展開することで処理を行うことが可能である。

10

【0019】

ネットワークI／F214は、画像形成システム内の装置とネットワーク通信を行うためのインタフェースである。ネットワークI／F214は、データ受信を行ったことをCPU211に伝達することや、RAM212上のデータをネットワーク106に送信することが可能である。プリンタI／F215は、CPU211から送信された印刷データをプリンタ202に送信することや、プリンタ202から受信したプリンタの状態をCPU211に伝達することが可能である。スキャナI／F216は、CPU211から送信された画像読み取り指示をスキャナ203に送信し、スキャナ203から受信した画像データをCPU211に伝達することや、スキャナ203から受信した状態をCPU211に伝達することが可能である。

20

【0020】

操作部I／F217は、操作部204から入力されたユーザからの指示をCPU211に伝達することや、ユーザが操作するための画面情報を操作部204に伝達することが可能である。拡張I／F218は、画像処理装置101に外部機器を接続することを可能とするインタフェースである。拡張I／F218は、例えば、USB(Universal Serial Bus)形式のインタフェースを具備する。画像処理装置101は、USBメモリ等の外部記憶装置が拡張I／F218に接続されることにより、当該外部記憶装置に記憶されているデータの読み取り及び当該外部記憶装置に対するデータの書き込みを行うことが可能である。

30

【0021】

プリンタ202は、プリンタI／F215から受信した画像データを用紙やシート等の記録媒体に印刷することや、プリンタ202の状態をプリンタI／F215に伝達することが可能である。スキャナ203は、スキャナI／F216から受信した画像読み取り指示に従って、自身に置かれた用紙に表示されている情報を読み取ってデジタル化してスキャナI／F216に伝達することが可能である。また、スキャナ203は、自身の状態をスキャナI／F216に伝達することが可能である。操作部204は、画像処理装置101に対して各種の指示を行うための操作をユーザに行わせるためのインタフェースである。例えば、操作部204は、タッチパネルを有する液晶画面を具備し、ユーザに操作画面を提供すると共に、ユーザからの操作を受け付ける。

40

【0022】

<画像処理装置のソフトウェア構成>

次に、図3を参照して、本実施形態に係るCPU211が処理するソフトウェアの構造の一例を説明する。図3に示す構造を有するソフトウェアは、画像処理装置101のHDD213に記憶されたプログラムを用いることにより構成される。図3において、一部の例外を除いて、下位の階層が提供するサービスを上位の階層が利用する関係になっている。

【0023】

図3において、最下層は、オペレーティングシステム317を含み、プログラムの実行の管理やメモリ管理等を行う階層である。オペレーティングシステム317の中には、プ

50

リント制御ドライバ318、スキャナ制御ドライバ319、及びネットワークI/F制御ドライバ320が組み込まれる。プリンタ制御ドライバ318、スキャナ制御ドライバ319、及びネットワークI/F制御ドライバ320は、相互に連携することにより機能することが可能である。プリンタ制御ドライバ318はプリンタI/F215を介してプリンタ202を制御するためのソフトウェアである。スキャナ制御ドライバ319はスキャナI/F216を介してスキャナ203を制御するためのソフトウェアである。ネットワークI/F制御ドライバ320はネットワークI/F214を制御するためのソフトウェアである。

【0024】

最下層から2段目の層には、第2の実行環境の一例となる言語デバイス制御ライブラリ309と画像処理制御部340とが含まれる。本実施形態においては、例えば、デバイス制御ライブラリ309と画像処理制御部340とはC言語やC++言語などのコンパイル言語（第2のプログラミング言語）で記述される。従って、CPU211で直接実行できるオブジェクトファイルの形式でHDD213に格納されているものとする。デバイス制御ライブラリ309は、後述の単独機能プラグインアプリケーション302又はデバイス制御アプリケーション304と静的又は動的にリンクされる。さらに、デバイス制御ライブラリ309は、各アプリケーションプログラムによる指示に基づいて、オペレーティングシステム317を利用する。また、各デバイス制御ライブラリ309は、Native画像処理接続ライブラリ314に対して画像処理を依頼することも可能である。

【0025】

次に、デバイス制御ライブラリ309の構成の一例について説明する。プリントライブラリ310は、プリンタ制御ドライバ318の機能を利用してプリントジョブの制御を行うためのAPI（Application Programming Interface）を提供するライブラリである。プリントジョブは、画像処理装置101のHDD213に保存されている印刷データを印刷することや、ネットワークI/F214を通して外部装置から受信した印刷データを印刷するといった一連の処理を示す。外部装置は、例えば、情報処理端末103、104である。コピーライブラリ311は、スキャナ制御ドライバ319及びプリンタ制御ドライバ318の機能を利用して、コピージョブの制御を行うためのAPIを提供するライブラリである。コピージョブは、スキャナ203でスキャンされた画像データをプリンタ202で印刷する一連の処理を示す。

【0026】

スキャン保存ライブラリ312は、スキャナ制御ドライバ319の機能を利用してスキャン保存ジョブの制御を行うためのAPIを提供するライブラリである。スキャン保存ジョブは、スキャナ203でスキャンされた画像データを印刷データ又は汎用フォーマットに変換後、HDD213又は拡張I/F218に接続されているUSB等の外部記憶装置に保存する一連の処理を示す。なお、汎用フォーマットは、例えば、PDF（Portable Document Format）やJPEG（Joint Photographic Experts Group）といったデータフォーマットである。

【0027】

スキャン送信ライブラリ313は、スキャナ制御ドライバ319及びネットワークI/F制御ドライバ320の機能を利用して、スキャン送信ジョブの制御を行うためのAPIを提供するライブラリである。スキャン送信ジョブは、スキャナ203でスキャンされた画像データを汎用フォーマットに変換後、ネットワークI/F214を通してファイルサーバに送信したり、電子メールに添付して外部装置に送信したりする一連の処理を示す。ここで、ファイルサーバは、例えばサーバ105であり、外部装置は、例えば情報処理端末103、104である。

【0028】

画像処理制御部340は、Native画像処理接続ライブラリ314、Native画像処理サーバ315、及びNative画像処理モジュール（ネイティブアプリケーション）316を含む。ここで、ネイティブアプリケーションとは、画像処理装置101に

10

20

30

40

50

プリインストールされ、画像処理等の各種機能を提供するアプリケーションである。一方、後述するプラグインアプリケーションは、画像処理装置101に対して、製品出荷後に追加可能なアプリケーションである。Native画像処理接続ライブラリ314は、デバイス制御ライブラリ309から画像処理の実行依頼を受けると、画像処理接続ライブラリ332に依頼内容を転送する。Native画像処理サーバ315は、後述するJava（登録商標）言語で記述されたソフトウェアから依頼を受けてNative画像処理モジュールを実行する機能を提供する。Native画像処理モジュール316は、様々な種類の画像処理を実行することの可能なソフトウェアである。なお、Native画像処理サーバ315及びNative画像処理モジュール316は、図3に記載の他のソフトウェアとは分離された論理メモリ空間を有するプログラム実行単位である、Native画像処理制御プロセス350上で実行される。メモリ空間を分離する具体的な方法としては、一般的なOS（Operating System）が備えるプロセス機構を用いることが考えられるが、他の方法でも構わない。このように、画像処理を行う場合に論理メモリ空間を分離する構成とすることにより、複雑な演算やこれに伴うメモリ操作を行う場合でも、処理の不具合が処理の依頼側に及ぼす影響を抑制することができる。

【0029】

最上位層であるJava言語実行環境330は、第1の実行環境の一例であり、プラグインアプリケーション301及びデバイス制御アプリケーション304を含む、アプリケーション層である。本実施形態では、プラグインアプリケーション301及びデバイス制御アプリケーション304はJava言語（第1のプログラミング言語）で記述され、Java仮想マシンが解釈するJavaバイトコード形式でHDD213に格納される。そのため、CPU211はJava仮想マシンのプログラムを実行し、Java仮想マシンがJavaバイトコードを読み出して実行することによって処理を行う。このようなプログラミング言語を利用する理由の一つにプログラム記述の簡易さが挙げられる。Javaにおいては、メモリ領域の管理を開発者で行う必要がなく、自動的に行われる。従って、プログラムを記述する際の手間が低減され、開発効率が向上することが見込まれる。

【0030】

各アプリケーションは、デバイス制御ライブラリ309や画像処理接続ライブラリ332の各APIを利用しながら動作することで様々な機能を提供する。また、デバイス制御アプリケーション304は、ファームウェアの更新により、その機能を拡張することが可能である。デバイス制御アプリケーション304に含まれるプリントアプリケーション305、コピーアプリケーション306、スキャン保存アプリケーション307、及びスキャン送信アプリケーション308は、それぞれ画面情報321、322、323、324を有する。CPU211は、操作部I/F217を通して操作部204にそれぞれの画面情報321、322、323、324を表示する。CPU211は、ユーザが操作部204を操作することによりデバイス制御アプリケーション304の設定の変更を検知すると、その変更の内容をHDD213に書き込む。

【0031】

CPU211（デバイス制御アプリケーション304）は、操作部204を介してジョブの実行依頼を検知すると、デバイス制御ライブラリ309のAPIをコールすることにより当該ジョブを開始する。また、デバイス制御アプリケーション304は、画像処理接続ライブラリ332に対して、画像処理を依頼することも可能である。

【0032】

画像処理接続ライブラリ332は許容メモリ定義部325を保持する。許容メモリ定義部325には画像処理接続ライブラリ332が同時実行する際の最大の許容メモリ量が定義されている。許容メモリ定義部325に定義された許容メモリは、プログラムに直接記載してHDD213から取得する方法や、予め許容量を定義したファイルを設置した上で、実行時にロードする方法などが挙げられる。なお、許容メモリを取得する方法は上述以外の方法でも構わない。画像処理接続ライブラリ332は、さらに実行中メモリ管理部326を保持する。実行中メモリ管理部326は、画像処理接続ライブラリ332経由で実

10

20

30

40

50

行中の画像処理の総メモリ量を管理している。

【0033】

次に、デバイス制御アプリケーション304の一例について説明する。プリントアプリケーション305は、プリントライブラリ310のAPIをコールすることによりプリントジョブを実行する。コピーアプリケーション306は、コピーライブラリ311のAPIをコールすることによりコピージョブを実行する。スキャン保存アプリケーション307は、スキャン保存ライブラリ312のAPIをコールすることによりスキャン保存ジョブを実行する。スキャン送信アプリケーション308は、スキャン送信ライブラリ313のAPIをコールすることでスキャン送信ジョブを実行する。

【0034】

次に、プラグインアプリケーション301の一例について説明する。プラグインアプリケーション301は、常駐アプリケーションであるデバイス制御アプリケーション304とは別に、プラグインとしてインストール及びアンインストールが可能なアプリケーションである。プラグインアプリケーション301は、単独機能プラグインアプリケーション302と、画像処理プラグインアプリケーション303とを有する。プラグインアプリケーション301には、それぞれ動作に必要なプログラムがパッケージングされている。プラグインアプリケーション301は、リモートUI (User Interface) 等を用いることにより、画像処理装置101にインストールされる。なお、リモートUIは、外部装置におけるWebブラウザからネットワークI/F214を経由して画像処理装置101にアクセスし、画像処理装置101の状況の確認、印刷ジョブの操作、及び各種設定などを行うことができる仕組みである。外部装置は、例えば、情報処理端末103、104である。また、プラグインアプリケーション301 (単独機能プラグインアプリケーション302、及び画像処理プラグインアプリケーション303) は、それぞれ個別に起動及び停止を行うことが可能である。

【0035】

以下に、プラグインアプリケーション301のインストールから起動、停止、アンインストールまでの一連の流れの一例を示す。CPU211は、プラグインアプリケーション301のインストールを検知すると、当該プラグインアプリケーション301の情報をHDD213に保存する。続いて、CPU211は、プラグインアプリケーション301に対する開始指示を検知すると、当該プラグインアプリケーション301に対する起動指示を行う。プラグインアプリケーション301は起動している間、各プログラムの内容を実行することが可能である。続いて、CPU211は、プラグインアプリケーション301に対する停止指示を検知すると、当該プラグインアプリケーション301に対して停止指示を行う。さらに、CPU211は、プラグインアプリケーション301に対するアンインストール指示を検知すると、当該プラグインアプリケーション301の情報をHDD213から削除する。なお、これらの指示は、例えば、リモートUIや操作部204から行うことが可能であるが、これら以外の方法でこれらの指示が行われても構わない。

【0036】

次に、単独機能プラグインアプリケーション302の一例について説明する。単独機能プラグインアプリケーション302は、画面情報321を有する。単独機能プラグインアプリケーション302は、デバイス制御ライブラリ309のAPIをコールすることで常駐のデバイス制御アプリケーション304とは別の機能や画面をユーザに提供することができる。例えば、単独機能プラグインアプリケーション302は、ある画像データをコピーすると共にスキャンし、自身が保持する送信先データベース内の特定の宛先に文書を送信する機能等、複数の機能を組み合わせて提供することが可能である。なお、単独機能プラグインアプリケーション302は、画像処理を行う機能を有していなくてもよい。この場合、画像処理の設定は行われない。更に、デバイス制御ライブラリ309は、印刷データ又は汎用フォーマットに変換後の画像データを受信すると、適切なオペレーティングシステム317に対して処理制御を指示し、ジョブを実行させる。

【0037】

次に、画像処理プラグインアプリケーション303の一例について説明する。画像処理プラグインアプリケーション303は、特定の画像処理を提供するアプリケーションである。画像処理プラグインアプリケーション303は、単独機能プラグインアプリケーション302やデバイス制御アプリケーション304などから画像処理接続ライブラリ332経由で処理対象の画像データと処理パラメータを受け付け、依頼された画像処理を実行する。また、画像処理プラグインアプリケーション303はデバイス制御ライブラリ309からも、Native画像処理接続ライブラリ314と画像処理接続ライブラリ332を経由して画像処理依頼を受信することが可能である。なお、画像処理プラグインアプリケーション303は、HDD213内に複数存在してもよい。例えば、入力画像に対する画像形式変換、斜行補正や、帳票認識やOCR処理等を行うことができる画像処理プラグインアプリケーション303がそれぞれ存在してもよい。

10

【0038】

画像処理プラグインアプリケーション303は、それ自体が画像処理機能を有していなくてもよい。例えば、画像処理プラグインアプリケーション303は、Native画像処理クライアント331を利用し、Native画像処理モジュール316の画像処理機能を利用することもできる。画像処理プラグインアプリケーション303がNative画像処理モジュール316を利用する理由の一例として、画像処理を行う際の処理速度の高速化が挙げられる。画像処理を行う際には大量かつ複雑な数値演算を行ったり、処理の過程で大量のメモリを必要とする場合がある。そのような場合にはJavaのような仮想マシンを仲介して処理を行うプログラミング言語処理系を利用するよりも、直接CPUが実行可能なオブジェクトファイルを生成できるコンパイル言語を利用の方が処理速度を向上させることができる。

20

【0039】

各画像処理プラグインアプリケーション303は、実行メモリ定義部327を保持する。実行メモリ定義部327には、各プラグインが実行する画像処理によって利用される最大メモリ量が定義されている。なお、上述のメモリ量は、プラグインごとに保持してもよいし、画像処理の関数ごとに保持してもよい。

【0040】

<処理手順>

次に、図4を参照して、本実施形態に係る画像処理装置101の処理手順の一例を説明する。ここでは、画像処理接続ライブラリ332が画像処理を受け付け、メモリ制御を行った上で画像処理プラグインアプリケーション303に処理依頼を行うケースを想定する。本フローチャートでは、画像処理プラグインアプリケーション303を用いた画像処理の一例を示す。なお、以下で説明する処理は、例えばCPU211がHDD213に保存されたプログラムをRAM212に読み出して実行することにより実現される。

30

【0041】

まず、S401で、画像処理接続ライブラリ332は、デバイス制御アプリケーション304、Native画像処理接続ライブラリ314、又は単独機能プラグインアプリケーション302から画像処理の実行依頼を受信する。このとき、受信した実行依頼には処理を依頼する画像処理プラグインアプリケーション303を特定するために必要なパラメータが入力されているものとする。

40

【0042】

次に、S402で、画像処理接続ライブラリ332は、取得した実行依頼に対応する画像処理を実行可能な画像処理プラグインアプリケーション303を探索する。続いて、S403で、画像処理接続ライブラリ332は、S402の探索結果に基づいて、対象となる画像処理プラグインアプリケーションが存在したか否かを判定する。

【0043】

S403で受信した画像処理依頼を実行できるアプリケーションが存在しないと判定するとS404に進み、画像処理接続ライブラリ332は、受信した画像処理を実行できるアプリケーションが存在しない旨のエラーを依頼元（要求元）のモジュールに返却する。

50

その後、処理を終了する。例えば、Native 画像処理接続ライブラリ 314 が依頼元であった場合、デバイス制御ライブラリ 309 を通じて、デバイス制御アプリケーション 304 又は単独機能プラグインアプリケーション 302 に上述のエラー内容が伝播される。デバイス制御アプリケーション 304 又は単独機能プラグインアプリケーション 302 は受信した上述のエラー内容を各アプリケーションの操作画面に表示する。本エラーにより、処理に必要なモジュールが存在していないことをユーザが知ることが可能となる。したがって、サービスマン又は IT 管理者が上述のインストール手法によって必要な画像処理プラグインアプリケーション 303 をインストールすることにより、処理を継続することが可能である。

【0044】

一方、S403 で対象の画像処理プラグインアプリケーション 303 が存在したと判定すると、S405 に進み、画像処理接続ライブラリ 332 は、対象の画像処理プラグインアプリケーション 303 の実行メモリ定義部 327 から実行メモリ量を取得する。この実行メモリ量は、上述したように、各プラグイン単位で定義されてもよいし、各プラグインにおける画像処理の各関数単位で保持されてもよい。続いて、S406 で、画像処理接続ライブラリ 332 は、自身の許容メモリ定義部 325 から許容メモリ量を取得する。続いて、S407 で、画像処理接続ライブラリ 332 は、自身の実行中メモリ管理部 326 から実行中総メモリ量を取得する。

【0045】

次に、S408 で、画像処理接続ライブラリ 332 は、取得した実行中メモリ量と、自身が管理している実行中総メモリ量を加算した結果が、許容メモリ量を超えるか否か（メモリ不足となるか否か）を判断する。超えないと判断した場合、S409 で、画像処理接続ライブラリ 332 は、実行中メモリ管理部 326 で管理している実行中メモリ量に実行メモリ定義部 327 で定義された実行メモリ量を加算する。続いて、S410 で、画像処理接続ライブラリ 332 は、対象の画像処理プラグインアプリケーション 303 に処理を依頼する。画像処理プラグインアプリケーション 303 は、画像処理接続ライブラリ 332 から処理パラメータと共に、画像処理依頼を受信すると、処理パラメータに応じて画像処理を実行する。画像処理は画像処理プラグインアプリケーション 303 内で行う手法と、Native 画像処理モジュール 316 で実行する手法があるが、詳細については後述する。なお、画像処理実行時に、画像処理プラグインアプリケーション 303 又は Native 画像処理モジュール 316 では、CPU 211 が入力データを RAM 212 に展開する。この際に画像処理内容、及び入力データサイズに応じた量のメモリを確保する必要がある。その後、S411 で、画像処理接続ライブラリ 332 は、画像処理プラグインアプリケーション 303 から処理結果を受信すると、S412 で実行中総メモリ量から実行メモリ量を減算する。さらに、S413 で、画像処理接続ライブラリ 332 は、画像処理結果を依頼元の各モジュールに返却し、処理を終了する。

【0046】

一方、S408 で超えたと判断した場合、S414 に進み、画像処理接続ライブラリ 332 は、依頼元のモジュールに対し、現在画像処理に必要なメモリリソースが足りていないため（メモリ不足のため）実行できない旨のエラーを返却し、処理を終了する。Native 画像処理接続ライブラリ 314 が依頼元であった場合、デバイス制御ライブラリ 309 を通じて、上述のエラー内容がデバイス制御アプリケーション 304 又は単独機能プラグインアプリケーション 302 に伝播される。デバイス制御アプリケーション 304 又は単独機能プラグインアプリケーション 302 は受信した上述のエラー内容に基づき、例外処理を実行する。

【0047】

上記例外処理の一例としてリトライ処理が挙げられる。各アプリケーションは上述のエラーの受信により、画像処理接続ライブラリ 332 上で複数の画像処理が同時に実行されていると判断できる。従って、各アプリケーションは、一旦現在実行中の画像処理が終わるのをリトライにより待ち、実行中の画像処理が終わったタイミングで再び画像処理要求

10

20

30

40

50

を送信する。これにより、処理を中断することなく継続することが可能となる。

【0048】

また、その他の例外処理として、各アプリケーションの操作画面において、複数の画像処理が同時実行中であるため、時間を置いて（所定時間が経過した後に）再度ジョブを投入するように促す旨のエラーを表示する制御が挙げられる。ユーザは上述のエラーによって、時間を置けばジョブが実行可能であることを知ることが可能となる。

【0049】

上述のように、本実施形態によれば、画像処理実行を行うモジュール（プラグインアプリケーション）に処理依頼を行う前に、実行する画像処理によって許容メモリ量を超えるか否かを予め判断し、超える場合にはそもそも画像処理を依頼しない。これにより、本実施形態では、許容メモリ量を超えたメモリの確保を回避することができる。したがって、システムが許容するメモリ量を超えたメモリを確保することでシステムの動作が不安定になる前に、事前にメモリ確保の制御を行うことが可能となる。

【0050】

＜画像処理プラグインアプリケーションによる画像処理＞

以下では、画像処理プラグインアプリケーションによる画像処理について説明する。画像処理を依頼された画像処理プラグインアプリケーション303は、依頼された処理内容に基づき、画像処理を実行する。画像処理プラグインアプリケーション303は、少なくとも2種類の方法で画像処理を実行することができる。1つ目は、画像処理プラグインアプリケーション303内で実行する方法である。この場合、画像処理プラグインアプリケーションは、画像処理接続ライブラリ332から受信した処理依頼及び処理パラメータを元に画像処理を実行する。この場合、画像処理プラグインアプリケーションはJava言語で記載する必要がある。2つ目は、Native画像処理モジュール316に依頼する方法である。画像処理プラグインアプリケーション303は、Native画像処理クライアント331に処理を依頼することにより、画像処理プラグインアプリケーション303とは異なる画像処理専用プロセスにおいて画像処理を実行することが可能となる。

【0051】

ここで、図5を参照して、各画像処理プラグインアプリケーション303が画像処理を実行する処理手順を説明する。本フローチャートでは、Native画像処理サーバ315に依頼して画像処理を実行する処理手順を示す。なお、以下で説明する処理は、例えばCPU211がHDD213に保存されたプログラムをRAM212に読み出して実行することにより実現される。

【0052】

画像処理プラグインアプリケーション303はNative画像処理クライアント331に処理パラメータと共に画像処理依頼を送信する。Native画像処理クライアント331はNative画像処理サーバ315に対して、受信した処理パラメータと共に処理依頼を行う。

【0053】

S501で、Native画像処理サーバ315は、画像処理の実行依頼を受信すると、S502で、受信した処理パラメータを元に、対象となるNative画像処理モジュール316を探索する。続いて、S503で、Native画像処理サーバ315は、対象のNative画像処理モジュール316が存在するか否かを判定する。存在しない場合は、S504に進み、Native画像処理サーバ315は、対象のNative画像処理モジュール316が存在しない旨のエラーをNative画像処理クライアント331に返却し、処理を終了する。

【0054】

一方、対象のNative画像処理モジュール316が存在する場合、S505に進み、Native画像処理サーバ315は、対象のNative画像処理モジュール316において、受信した処理パラメータを元に画像処理を実行させる。続いて、S506で、Native画像処理サーバ315は、S505の画像処理結果をNative画像処理

10

20

30

40

50

クライアント 331 に返却し、処理を終了する。Native 画像処理クライアント 331 は、画像処理結果又はエラー内容を受信すると、その内容を画像処理プラグインアプリケーション 303 に返却する。

【0055】

以上説明したように、本実施形態に係る画像処理装置は、1 以上のプラグインアプリケーションによる複数の画像処理を実行可能な画像処理装置である。本画像処理装置は、プラグインアプリケーションによる画像処理の実行中に、プラグインアプリケーションによる新たな画像処理の実行が要求されると、当該新たな画像処理を実行可能なプラグインアプリケーションを探索する。また、探索されたプラグインアプリケーションとともに保持されている、プラグインアプリケーションによる新たな画像処理を実行した際のメモリ使用量と、プラグインアプリケーションによって既に実行中の画像処理のそれぞれのメモリ使用量とを取得する。また、取得された各画像処理のメモリ使用量及び要求された新たな画像処理を実行した際のメモリ使用量から得られる総メモリ使用量と、当該プラグインアプリケーションを実行するプロセスでの使用が許容されるメモリ使用量を示す許容メモリ量とを比較する。さらに、本画像処理装置は、比較の結果、総メモリ使用量が許容メモリ量を超えない場合は新たな画像処理を実行する。一方、総メモリ使用量が許容メモリ量を超える場合は新たな画像処理を実行せず、要求元に対してメモリ不足により新たな画像処理が実行できない旨のエラーを通知する。これにより、画像処理プラグインアプリケーション 303 の起動（インストール）を制限することなく、画像処理実行時に実行中の画像処理の総メモリを計算することにより、複数の画像処理に対するメモリ制御をランタイムで行うことができる。つまり、本実施形態によれば、好適にメモリ不足を回避しつつ、複数の画像処理を実行することができる。その結果ユーザの利便性を向上することができる。

【0056】

本発明は上記実施形態に限らず様々な変形が可能である。例えば、各画像処理プラグインアプリケーション 303 の実行メモリ定義部 327 に定義されている実行メモリは、Native 画像処理モジュール 316 で画像処理を実行する際の実行メモリを記載することも可能である。

【0057】

<第 2 の実施形態>

以下では、本発明の第 2 の実施形態について説明する。上記第 1 の実施形態では、画像処理接続ライブラリ 332 が、複数の画像処理に対するメモリ制御をランタイムに行う例を説明した。本実施形態では、プラグインアプリケーションから画像処理を依頼された際に、Native 画像処理制御プロセス 350 に閉じて Native 画像処理サーバ 315 が複数の画像処理に対するメモリ制御を行う例について説明する。

【0058】

<画像処理装置のソフトウェア構成>

まず、図 6 を参照して、本実施形態に係る CPU 211 が処理するソフトウェアの構造の一例を説明する。なお、基本的な構成は上記第 1 の実施形態において図 3 を用いて説明した内容と同様であるため、主に差分について説明する。従って、同様の構成については同一の参照符号を付し、説明を省略する。

【0059】

Native 画像処理サーバ 315 は、許容メモリ定義部 601 を保持する。許容メモリ定義部 601 には Native 画像処理サーバ 315 上で同時実行される際の可能な最大の許容メモリ量が定義されている。許容メモリ定義部 601 に定義された許容メモリ量は、プログラムに直接記載して HDD 213 から取得する方法や、予め許容メモリ量を定義したファイルを設置した上で、実行時にロードする方法などが挙げられる。なお、許容メモリを取得する方法は上述以外の方法でも構わない。Native 画像処理サーバ 315 は、さらに実行中メモリ管理部 602 を保持する。実行中メモリ管理部 602 は、Native 画像処理サーバ 315 経由で実行中の画像処理の総メモリ量を管理している。

【0060】

<処理手順>

次に、図7を参照して、本実施形態に係る画像処理装置101の処理手順の一例を説明する。ここでは、Native画像処理サーバ315が実行中の画像処理のメモリを管理して、画像処理を実行するケースを想定する。ここでの画像処理の依頼元は、画像処理プラグインアプリケーション303となる。なお、以下で説明する処理は、例えばCPU211がHDD213に保存されたプログラムをRAM212に読み出して実行することにより実現される。

【0061】

S701で、Native画像処理サーバ315は、Native画像処理クライアント331から処理パラメータと共に画像処理の実行依頼を受信すると、S702でNative画像処理モジュール316を探索する。続いて、S703で、Native画像処理サーバ315は、S702の探索結果に基づき、対象のNative画像処理モジュール316が存在するか否かを判断する。

【0062】

存在しない場合はS704に進み、Native画像処理サーバ315は、対象のNative画像処理モジュール316が存在しない旨のエラーを返却し、処理を終了する。エラーを受信したNative画像処理クライアント331は、依頼元の画像処理プラグインアプリケーション303にエラー内容を伝播する。この場合、画像処理装置101のHDD213に必要なNative画像処理モジュール316がインストールされていないということなので、通常の操作においては発生しない。したがって、開発者がソースコードを修正する必要がある。

【0063】

一方、対象のNative画像処理モジュール316が存在する場合はS705に進み、Native画像処理サーバ315は、対象のNative画像処理モジュール316の実行メモリ定義部603から実行メモリ量を取得する。続いて、S706で、Native画像処理サーバ315は、自身の許容メモリ定義部601から許容メモリ量を取得する。さらに、S707で、Native画像処理サーバ315は、自身の実行中メモリ管理部602から実行中総メモリ量を取得する。

【0064】

次に、S708で、Native画像処理サーバ315は、取得した実行中メモリ量と実行中総メモリ量を加算した結果が、許容メモリ量を超えるかどうかを判断する。許容メモリ量を超えないと判断した場合は、S709に進み、Native画像処理サーバ315は、実行中総メモリ量に実行メモリ量を加算し、S710で、Native画像処理モジュール316に対して画像処理実行を依頼する。Native画像処理モジュール316は、処理パラメータと共に画像処理依頼を受信すると、処理パラメータに応じて画像処理を実行する。なお、画像処理実行する際、Native画像処理モジュール316では、CPU211が入力データをRAM212に展開し、処理を行う。この際に画像処理内容、及び入力データサイズに応じて、メモリを確保する必要がある。

【0065】

次に、S711で、Native画像処理サーバ315は、画像処理結果をNative画像処理モジュール316から受け取ると、S712で実行中総メモリ量から実行メモリ量を減算する。続いて、S713で、Native画像処理サーバ315は、画像処理結果をNative画像処理クライアント331に返却し、処理を終了する。

【0066】

一方、S708で許容メモリ量を超えると判断した場合はS714に進み、Native画像処理サーバ315は、現在必要なメモリリソースが足りていないため、画像処理を実行できない旨のエラーをNative画像処理クライアント331に返却する。その後、処理を終了する。

【0067】

Native画像処理クライアント331は上述の画像処理結果及びエラー内容を画像

10

20

30

40

50

処理プラグインアプリケーション303に伝播する。画像処理プラグインアプリケーションは画像処理結果を受け取ると、画像処理接続ライブラリ332を通じて依頼元のモジュールに伝播する。一方、メモリリソースが足りていないエラーを受信した場合、例外処理を実行する。ここでの例外処理の一例としてリトライ処理が挙げられる。画像処理プラグインアプリケーション303は、上述のエラーを受信した場合、既に実行中の画像処理が存在するため実行できないと判断できる。したがって、一定の時間を置いて再度画像処理クライアント331に処理を依頼することで、実行することが可能になる。このように画像処理プラグインアプリケーションに閉じてリトライ処理を行うことで、依頼元のモジュールでリトライ処理を行わなくてもよいといったメリットがある。また、他の例外処理として依頼元の単独機能プラグインアプリケーション302及びデバイス制御アプリケーション304まで上述のエラーを伝播することが挙げられる。アプリケーションにより、リトライしたい場合と、中断したい場合の、両者が想定できる場合には、アプリケーション側に上述のエラーを伝播することによって、アプリケーション内でリトライ及び中断を切り替えることが可能となる。

10

【0068】

上述のようにNative画像処理モジュール316に処理依頼を行う前に、予め実行する画像処理によって許容メモリ量を超えているか否かを判断して制御することで、超えた場合にはそもそも画像処理を依頼しないためメモリを無駄に確保しない。従って、システム（装置）が許容するメモリ量を超えてメモリを確保することでシステムの動作が不安定になる前に、事前にメモリ確保の制御を行うことが可能となる。

20

【0069】

本実施形態において、Native画像処理制御プロセス350に閉じた上記のメモリ制御と並行して、プラグインによって実現される他の画像処理制御プロセスについても同様のメモリ制御を行うこととしてもよい。即ち、プロセス毎にメモリ許容量を定義してもよい。例えば、本実施形態の構成を上記第1の実施形態の構成と組み合わせて適用してもよい。

【0070】

本実施形態によれば、画像処理プラグインアプリケーション303の起動の制限はしないが、画像処理実行時に実行中の画像処理の総メモリを計算することにより、複数の画像処理に対するメモリ制御をランタイムで行うことが可能となる。その結果ユーザの利便性を向上することができる。さらに、上記第1の実施形態とは異なり、画像処理を専門に行うプロセス（Native画像処理制御プロセス350）内に制限してメモリ管理を行うことが可能となる。よって、各画像処理プラグインアプリケーション303がNative画像処理制御プロセス350上での画像処理のメモリ制御を意識する必要がなくなり、画像処理プラグインアプリケーション303の開発効率が向上するといったメリットもある。

30

【0071】

<第3の実施形態>

以下では、本発明の第3の実施形態について説明する。本実施形態では、Native画像処理モジュール316が主体となってメモリ制御を行う例を説明する。また、本実施形態においては、メモリ制御を行ったうえで、上記第1及び第2の実施形態のように並列処理が行えない場合はエラーを直ぐに返却することなくそのまま処理を実行し、複数ページの画像処理に対する並列処理が行える場合は並列処理を用いることで高速化を行う例を示す。

40

【0072】

<処理手順>

まず、図8を参照して、本実施形態に係る画像処理装置101の処理手順の一例を説明する。ここでは、Native画像処理モジュール316がNative画像処理サーバ315に実行メモリを登録し、メモリ制御を行う一例を説明する。なお、以下で説明する処理は、例えばCPU211がHDD213に保存されたプログラムをRAM212に読

50

み出して実行することにより実現される。

【0073】

まず、S801で、Native画像処理モジュール316は、Native画像処理サーバ315から画像処理の実行依頼を受信する。続いて、S802で、Native画像処理モジュール316は、受信した画像処理に必要なメモリサイズ（メモリ使用量）を実行メモリ定義部603から取得する。さらに、S803で、Native画像処理モジュール316は、受信した画像処理に対して並列処理が可能かどうかを判断する。この判断は、Native画像処理モジュール316に対して並列処理が可能か否かの情報を持たせ、当該情報を用いて判断してもよい。なお、並列処理が可能か否かの情報は、上記第1及び第2の実施形態で説明したように、ネイティブアプリケーションによって既に実行中の画像処理のそれぞれのメモリ使用量と、上記必要なメモリ使用量とに基づき、並列処理が可能であるか否かを判断してもよい。例えば、上記2つの総メモリ使用量で並行処理を行った場合に許容メモリ量に収まるような、総メモリ使用量に対応する閾値メモリ量を定義し、それらの比較結果に基づいてもよい。また、実行メモリ定義部603に予めメタ情報として並列処理可能か否かの情報（例えば、閾値メモリ量）を付加しておき、それを読み出すことで判断してもよい。

10

【0074】

次に、S804で、Native画像処理モジュール316は、S803の判断結果に基づいて、受信した画像処理に対して並列処理が可能でなければS805に処理を進め、可能である場合にはS811に処理を進める。

20

【0075】

S805は、並列処理が可能でない（各画像処理を順次実行する）場合におけるS806乃至S810の処理を同期的に受信した画像処理のページ数分繰り返す処理を示す。S806で、Native画像処理モジュール316は、Native画像処理サーバ315に自身の実行メモリ定義部603から取得したメモリ情報を登録する。Native画像処理サーバ315によるメモリ情報の登録の詳細については図9を用いて後述する。S807で、Native画像処理モジュール316は、S806で実行メモリを登録できたか否かを判断する。登録できなかった場合はS808に進み、Native画像処理モジュール316は、画像処理に必要なメモリリソースが足りていないため実行できない旨のエラーをNative画像処理サーバ315に返却し、処理を終了する。ここで、Native画像処理サーバ315はNative画像処理クライアント331を通して呼び出し側のアプリに上記エラーを伝播する。

30

【0076】

Native画像処理サーバ315に実行メモリを登録できた場合はS809に進み、Native画像処理モジュール316は、対象の画像処理を実行する。画像処理が完了すると、S810で、Native画像処理モジュール316は、Native画像処理サーバ315に対して解除するメモリ情報と共に実行メモリ解除依頼を行う。繰り返しのループ処理が終了すると、処理を終了する。

【0077】

一方で、S804で並列処理が可能であると判断した場合は、S811に進み、並列処理が可能である場合におけるS812乃至S814の処理を同期的に受信した画像処理のページ数分繰り返す。S812で、Native画像処理モジュール316は、自身の実行メモリ定義部603から取得した実行メモリをNative画像処理サーバ315に対して登録する。なお、Native画像処理サーバ315によるメモリ登録のフローについては図9を用いて後述する。S813で、Native画像処理モジュール316は、実行メモリを登録できたか否かを判断する。登録できた場合はS814に進み、Native画像処理モジュール316は、並列実行数をインクリメントする。ここで、並列実行数とは、並列して実行するスレッド数を示し、RAM212やHDD213等で管理している変数を示す。一方で実行メモリが登録できなかった場合は本ループ処理から抜けて、S815に進む。また、繰り返しのループ処理が終了すると、S815に進む。このルー

40

50

ブ処理を抜けた段階で、並列実行数には、0から設定されたページ数分までの領域の正の整数が入力されている状況になっている。

【0078】

次に、S815で、Native画像処理モジュール316は、計算した並列実行数が0より大きいかどうかを判断する。並列実行数が0より大きければS816に進み、そうでない場合はS808に進む。S808で、Native画像処理モジュール316は、画像処理に必要なメモリリソースが足りていないため実行できない旨のエラーをNative画像処理サーバ315に返却し、処理を終了する。Native画像処理サーバ315はNative画像処理クライアント331を通して呼び出し側のアプリに上記エラーを伝播する。

10

【0079】

一方で並列実行数が1以上の整数であればS816に進み、Native画像処理モジュール316は、その並列実行数分のスレッドを起動する。本実施形態ではスレッドを起動したが、プロセスを起動する構成であってもよい。続いて、S817に進み、S818の処理を設定されたページ数分繰り返す。S818で、Native画像処理モジュール316は、空いているスレッドに1ページ単位の画像処理を割り当てて実行する。このループ処理を抜けると、全てのスレッドの画像処理実行が完了し、受信した全てのページに対する画像処理が完了する。最後に、S819で、Native画像処理モジュール316は、Native画像処理サーバ315に対して実行メモリの解除依頼を起動スレッド数分言い、処理を終了する。

20

【0080】

<実行メモリ登録>

次に、図9を参照して、S806及びS812において、Native画像処理サーバ315がNative画像処理モジュール316から実行メモリ登録依頼を受けた際の処理手順を説明する。なお、以下で説明する処理は、例えばCPU211がHDD213に保存されたプログラムをRAM212に読み出して実行することにより実現される。

【0081】

S901で、Native画像処理サーバ315は、Native画像処理モジュール316から実行メモリ情報と共に実行メモリ登録依頼を受信する。続いて、S902で、Native画像処理サーバ315は、受信したメモリサイズと実行中の画像処理メモリサイズとの合計が許容メモリサイズを超えているかどうかを判断する。超えていればS903に進み、Native画像処理サーバ315は、登録できなかった旨をNative画像処理モジュール316に返却し、処理を終了する。一方、超えていなければS904に進み、Native画像処理サーバ315は、実行中メモリ管理部602に受信したメモリサイズを加算し、S905で登録できた旨をNative画像処理モジュール316に返却し、処理を終了する。

30

【0082】

<実行メモリ解除>

次に、図10を参照して、S810において、Native画像処理サーバ315がNative画像処理モジュール316から実行メモリの解除依頼を受けた際の処理手順を説明する。なお、以下で説明する処理は、例えばCPU211がHDD213に保存されたプログラムをRAM212に読み出して実行することにより実現される。

40

【0083】

S1001で、Native画像処理サーバ315は、Native画像処理モジュール316から実行メモリ解除依頼を受信する。続いて、S1002で、Native画像処理サーバ315は、実行中メモリから受信したメモリサイズを減算し、処理を終了する。

【0084】

以上説明したように、本実施形態によれば、Native画像処理モジュール316が主体となってメモリ制御を実施することが可能となる。また、各Native画像処理モジュール316の特性上、並列処理可能なもの、可能ではないものに応じて適切にスレッ

50

ド制御を行うことが可能となるため、メモリ制御を行いつつ、複数ページにおける高速化を実現することが可能となる。

【0085】

＜第4の実施形態＞

以下では、本発明の第4の実施形態について説明する。上記第3の実施形態では、並列実行が可能な画像処理モジュールにおいては、複数ページの画像処理の実施前に、並列実行数を決定し、並列処理を行う例を示した。本実施形態では、上記第3の実施形態の変形例として、画像処理の実行中に動的に並列実行数を増やす例を説明する。

【0086】

＜処理手順＞

図11を参照して、本実施形態に係る画像処理中に動的に並列実行数を増やす処理手順について説明する。本フローチャートは図8のフローチャートのS816からS819までの処理の変形例となる。S816より前に行われる処理は図8のフローチャートと同じである。なお、以下で説明する処理は、例えばCPU211がHDD213に保存されたプログラムをRAM212に読み出して実行することにより実現される。

【0087】

S1101で、Native画像処理モジュール316は、並列実行数分のスレッドを起動する。続いて、S1102に進み、S1103乃至S1107のループ処理を設定されたページ数分繰り返し実行する。具体的には、S1103で、Native画像処理モジュール316は、空いているスレッドに1ページ単位の画像処理を割り当てて実行する。さらに、S1104に進み、S1105乃至S1107のループ処理を画像処理待ちのページ数分繰り返し実行する。

【0088】

S1105で、Native画像処理モジュール316は、Native画像処理サーバ315に実行メモリを登録する。続いて、S1106で、Native画像処理モジュール316は、実行メモリを登録できたか否かを判断する。実行メモリが登録できなかった場合はS1104のループを抜け、S1102のループへ戻る。一方、S1106で実行メモリを登録できたと判断すると、Native画像処理モジュール316は、実行スレッドを生成し、S1104に戻る。これらのループを設定された画像処理に対するページ数分繰り返し実行することにより（S1102）、全てのページの画像処理が完了し、S1108に進む。

【0089】

S1108で、Native画像処理モジュール316は、Native画像処理サーバ315に実行メモリの解除依頼を起動スレッド数分行って、処理を終了する。なお、実行メモリ登録（S1105）及び実行メモリ解除（S1108）に対するNative画像処理サーバ315の処理フローは、図9及び図10で説明したとおりである。

【0090】

以上説明したように、本実施形態によれば、複数ページの画像処理において、毎ページの画像処理の間に、Native画像処理サーバ315に問い合わせることにより実行スレッドを動的に増やすことが可能となる。したがって、上記第3の実施形態より、さらに効率よく複数ページの画像処理を実行することが可能となる。

【0091】

＜その他の実施形態＞

本発明は、上述の実施形態の1以上の機能を実現するプログラムを、ネットワーク又は記憶媒体を介してシステム又は装置に供給し、そのシステム又は装置のコンピュータにおける1つ以上のプロセッサがプログラムを読み出し実行する処理でも実現可能である。また、1以上の機能を実現する回路（例えば、ASIC）によっても実現可能である。

【符号の説明】

【0092】

101、102：画像処理装置、103、104：情報処理端末、105：サーバ

10

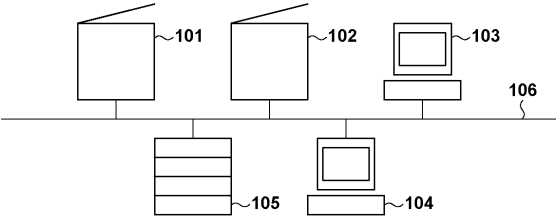
20

30

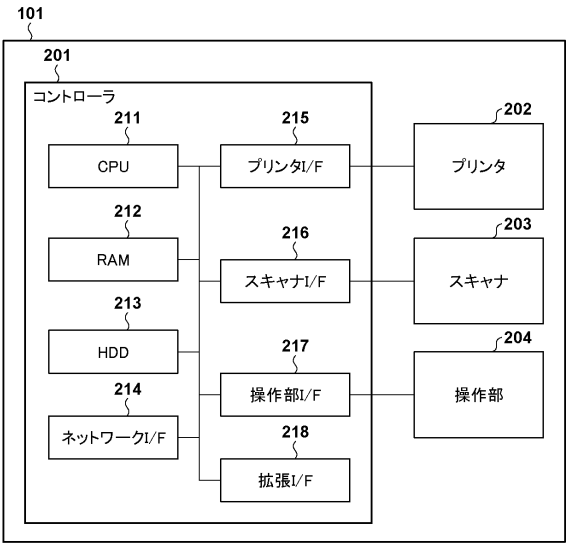
40

50

【図面】
【図 1】

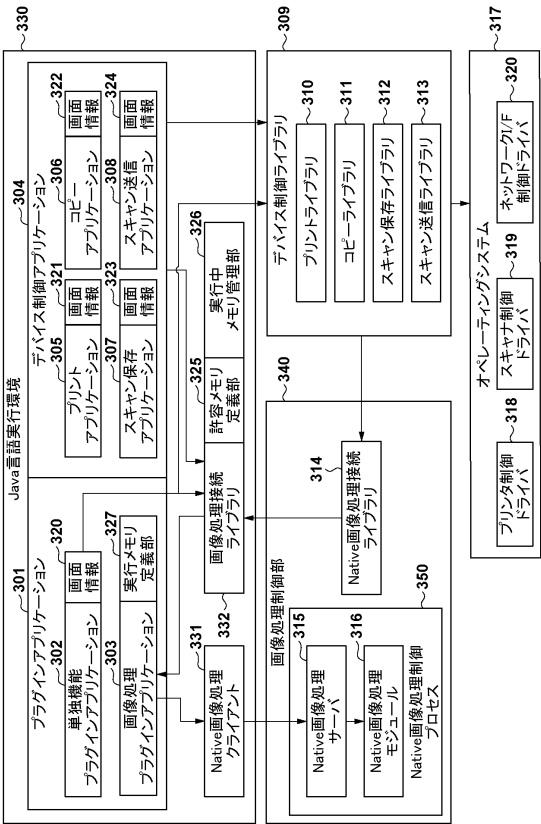


【図 2】

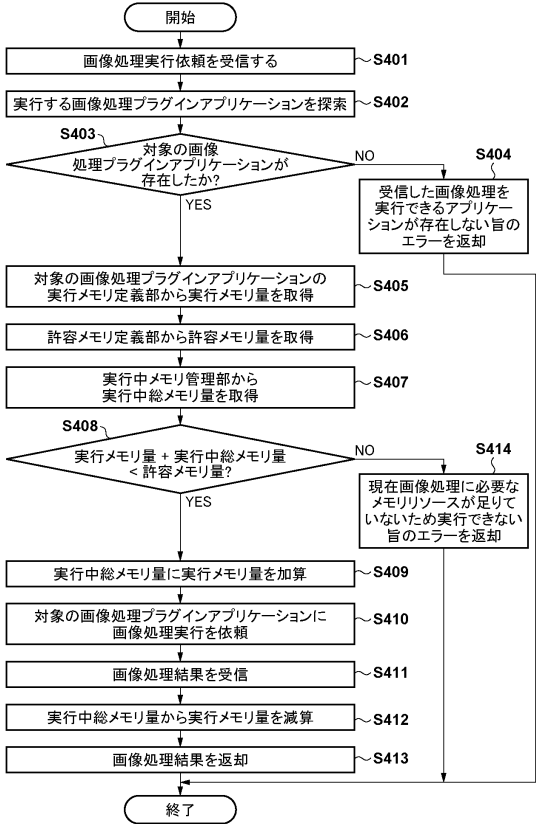


10

【図 3】



【図 4】



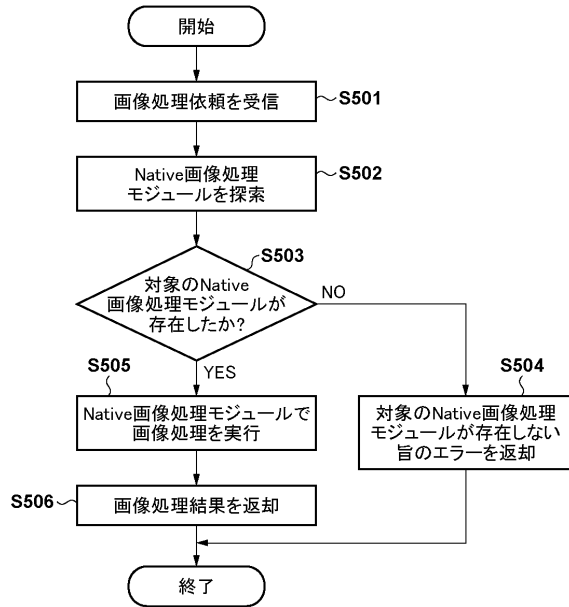
20

30

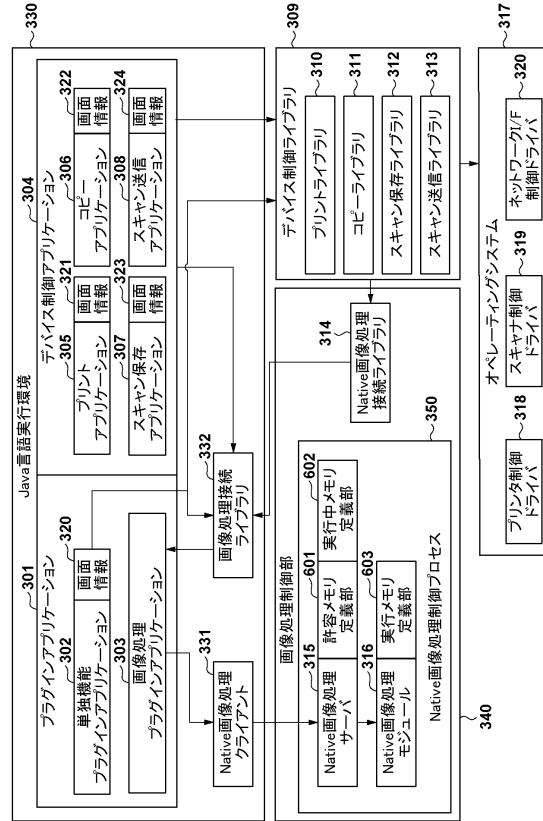
40

50

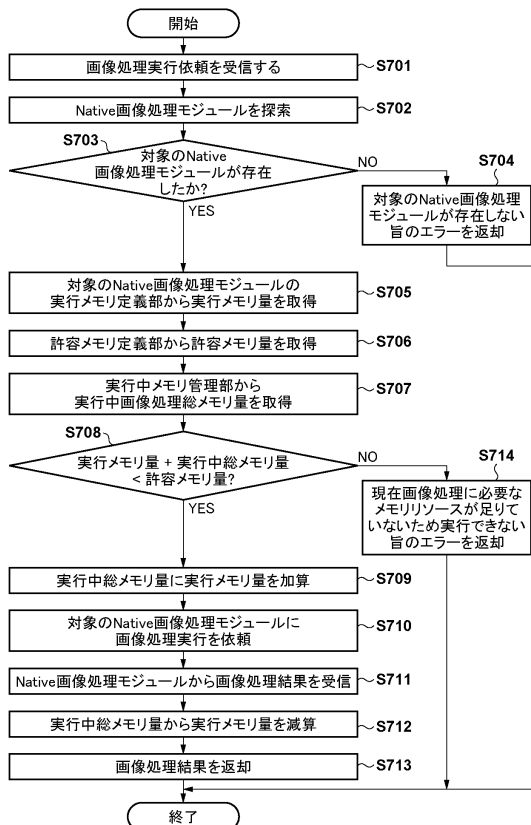
【図 5】



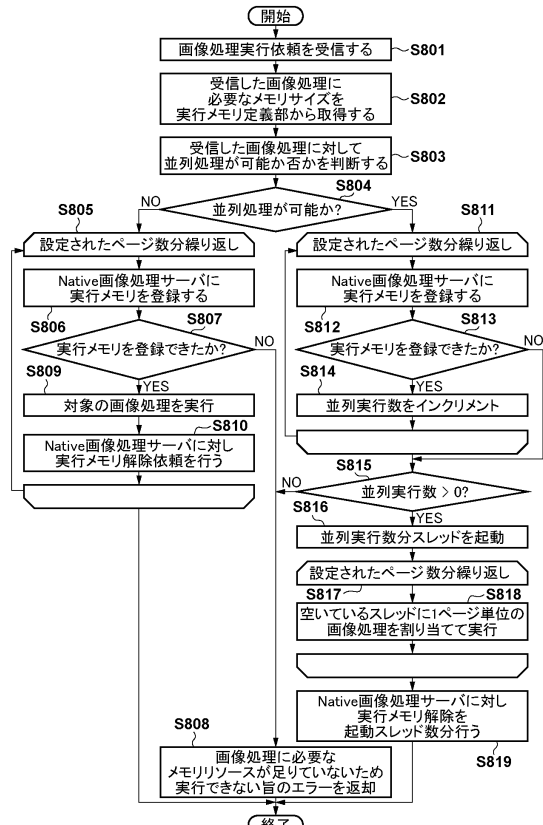
【図 6】



【図 7】



【図 8】



10

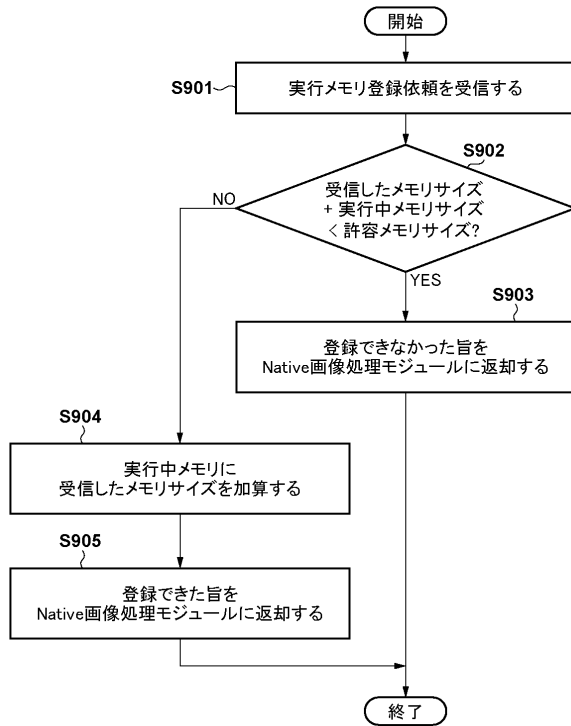
20

30

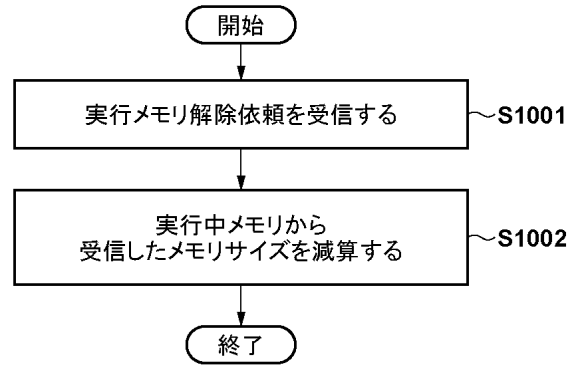
40

50

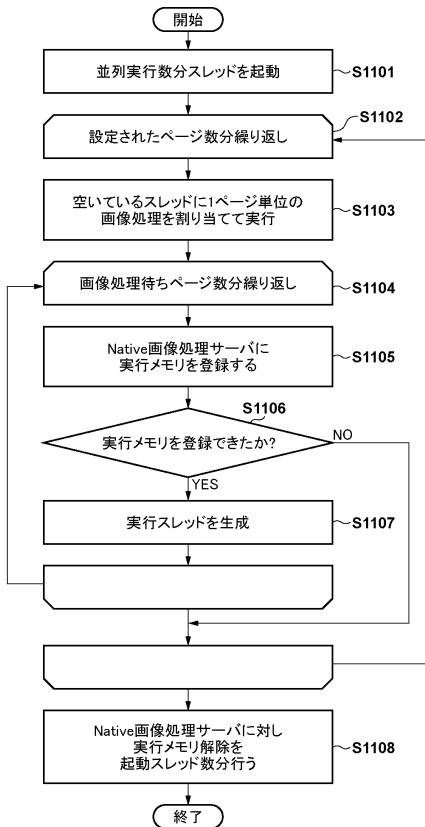
【図 9】



【図 10】



【図 11】



10

20

30

40

50

フロントページの続き

- (56)参考文献 特開２０１３－０４２５４５（ＪＰ，Ａ）
 特開２００４－３０６０１（ＪＰ，Ａ）
 特開２０１１－２４２８９０（ＪＰ，Ａ）
 特開２００８－２９３３１３（ＪＰ，Ａ）
 特開２０１４－１４９６０６（ＪＰ，Ａ）
- (58)調査した分野 (Int.Cl., DB名)
 Ｈ０４Ｎ１／００
 Ｇ０６Ｆ３／１２
 Ｇ０６Ｆ９／４５５－９／５４