(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2024年1月18日(18.01.2024)

WIPO | PCT

(10) 国際公開番号
WO 2024/013831 A1

(51) 国際特許分類:
G06F 1/329 (2019.01) *G06F 9/50* (2006.01)
G06F 1/3287 (2019.01)

(21) 国際出願番号： PCT/JP2022/027327

(22) 国際出願日： 2022年7月11日(11.07.2022)

(25) 国際出願の言語： 日本語

(26) 国際公開の言語： 日本語

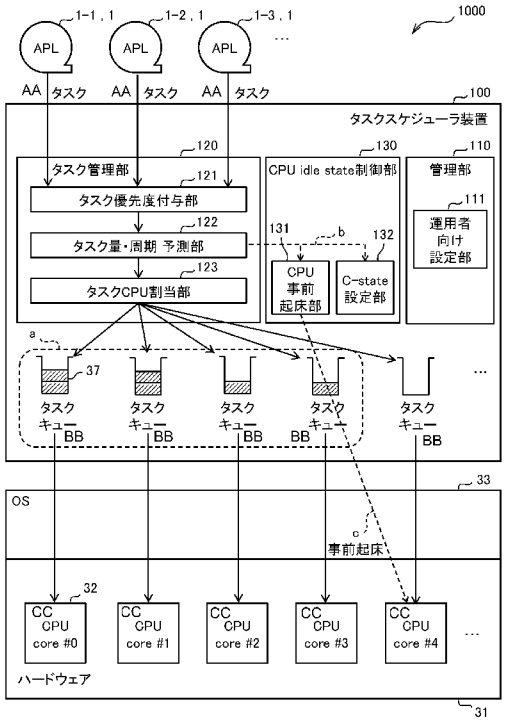
(71) 出願人: 日本電信電話株式会社 (**NIPPON TELEGRAPH AND TELEPHONE CORPORATION**) [JP/JP]; 〒1008116 東京都千代田区大手町一丁目５番１号 Tokyo (JP).

(72) 発明者: 藤本 圭 (**FUJIMOTO, Kei**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP). 斎藤 奨悟(**SAITO, Shogo**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP). 大谷 育生(**OTANI, Ikuo**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP). 名取 廣(**NATORI, Ko**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP).

(74) 代理人: 特許業務法人磯野国際特許商標事務所 (**ISONO INTERNATIONAL PATENT OFFICE, P.C.**); 〒1050001 東京都港区虎ノ門一丁目１番１８号 ヒューリック虎ノ門ビル Tokyo (JP).

(54) **Title:** TASK SCHEDULER DEVICE, CALCULATION SYSTEM, AND TASK SCHEDULING METHOD, AND PROGRAM

(54) 発明の名称： タスクスケジューラ装置、計算システム、タスクスケジューリング方法およびプログラム



31 Hardware
37, BB Task queue
100 Task scheduler device
110 Management unit
111 Setting unit for operator
120 Task management unit
121 Task priority adding unit
122 Task amount/cycle prediction unit
123 Task CPU allocation unit
130 CPU idle state control unit
131 CPU pre-wake up unit
132 C-state setting unit
c Pre-wake up
AA Task
CC core

(57) **Abstract:** In a calculation system (1000) for gradually reducing the operation state of a processor in accordance with a processing load and reducing a power consumption amount, a task scheduler device (100, 100A) for allocating tasks to a core group of the processor comprises: a task amount/cycle prediction unit (122) that acquires a use rate of the processor; and a task CPU allocation unit (123) that, on the basis of the use rate of the processor acquired by the task amount/cycle prediction unit (122), continuously allocates tasks for cores or the core group, of the processor, to be used

WO 2024/013831 A1

[続葉有]

(81) 指定国(表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

添付公開書類:

一　国際調査報告（条約第21条(3)）

at a prescribed frequency or more.

(57) 要約：処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システム（１０００）において、プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置（１００，１００Ａ）であって、プロセッサの使用率を取得するタスク量・周期 予測部（１２２）と、タスク量・周期 予測部（１２２）が取得したプロセッサの使用率をもとに、所定頻度以上で使用するプロセッサのコアまたはコア群については、継続的にタスクを割り当てるタスクＣＰＵ割当部（１２３）と、を備える。

明　細　書

発明の名称 ：

タスクスケジューラ装置、計算システム、タスクスケジューリング方法およびプログラム

技術分野

[0001] 本発明は、タスクスケジューラ装置、計算システム、タスクスケジューリング方法およびプログラムに関する。

背景技術

[0002] 無線アクセスシステムの概要について述べる。

図１９は、無線アクセスシステムの概要を説明する図である。

図１９に示すように、無線アクセスシステムは、図示しないユーザ端末（ＵＥ：User Equipment）、アンテナ（基地局アンテナ）２０、基地局（ＢＢＵ：Base Band Unit）３０、コア網４０を備える。

[0003] アンテナ２０は、ＵＥ１０と無線通信するアンテナおよび送受信部である（以下、「アンテナ」は、アンテナと送受信部、その電源部を総称して呼称する）。送受信データは、例えば専用ケーブルにより基地局３０に接続される。

[0004] 基地局３０は、ＵＥ１０と通信する陸上に開設する移動しない無線局である。無線信号処理を行う基地局（ＢＢＵ：Broad Band Unit）３０は、無線信号処理を行う専用ハードウェア（専用装置）である。または、基地局３０は、ＬＴＥ（Long Term Evolution）や５Ｇ（five generation）の信号処理集約システムにおける無線信号処理を、汎用サーバで処理を行うｖＲＡＮ（virtual Radio Access Network）である。ｖＲＡＮにおいては、基地局３０のハードウェアとして安価で大量に入手可能な汎用サーバを使用することができる。

基地局３０は、ハードウェア（ＨＷ）３１と、ハードウェア上のＣＰＵ（Central Processing Unit）３２と、ＯＳ等３３と、Ｌ１，Ｌ２，Ｌ３プロト

コル無線信号処理アプリケーション１－１，１－２，１－３（総称する場合は、ＡＰＬ１と呼ぶ）と、タスクスケジューラ装置３４と、を備える。

[0005] コア網４０は、ＥＰＣ（Evolved Packet Core）／（以下の説明において、「／」は「または」を表記する）５ＧＣ（5G Core Network）等である。

[0006] リアルタイム性が求められるシステムとして、ＲＡＮにおける基地局（ＢＢＵ）が挙げられる。

ＣＰＵ３２を使用して演算を行うＢＢＵでは、無線信号処理のタスクを、タスクスケジューラ装置３４がＣＰＵコアへ割り当てることで演算を行うことが多い（非特許文献１）。

[0007] 図２０は、基地局における無線アクセス処理のタスク管理例を説明する図である。図１９と同一構成部分には同一符号を付している。

基地局（ＢＢＵ）３０は、タスクスケジューラ装置３４を備える。タスクスケジューラ装置３４は、タスク管理、優先度付け、およびタスク割り振りを行う。

タスクスケジューラ装置３４は、ＡＰＬ１の無線信号処理のタスクを、タスクキュー３７に割り振り、タスクキュー３７からＣＰＵコア（CPUcore #0, CPUcore #1, …）３２へ割り当てる。

[0008] ［ＬＰＩ（Low Power Idle）ハードウェア制御］

ＣＰＵ３２には、ハードウェア制御によるＣＰＵ３２のidle状態を制御する機能があり、ＬＰＩと呼ばれる。ＬＰＩは、CPUidleやC-stateと呼称されることも多く、以下、ＬＰＩをC-stateとして説明する。

C-stateは、は、ＣＰＵ負荷が少なくなると、ＣＰＵ３２の回路の一部の電源をＯＦＦにすることで、省電力化を試行する（非特許文献２）。

[0009] 図２１は、C-stateの状態の一例を表にして示す図である。なお、ＣＰＵハードウェアに依って状態定義は異なるため、図２１はあくまでも参考例である。

図２１に示すように、CPUidle状態には、グレードＣ０～Ｃ６があり、ＣＰＵ３２の負荷がない時間が長くなるにつれ、深いsleep状態へ遷移する。深い

sleep状態の方がＣＰＵ消費電力は小さくなるが、一方で、それだけ復帰までに要する時間が長延化するため、低遅延の観点で課題となる場合がある。

[0010] C-stateは、ＣＰＵハードウェアに依って状態定義が異なる。例えば、Ｃ４やＣ５が無い機種、Ｃ１の次がＣ１Ｅというステートである機種等のバリエーションがある。

ステートが深くなるにつれ省電力効果は大きくなるが、それだけ、idle状態から復帰に要する時間も大きくなる。

[0011] また、どの深さまでCPUidle状態が遷移するかは、ＣＰＵのハードウェア制御になり、ＣＰＵ製品依存となる（kernel等のソフトウェアから制御できない場合が多い）。

[0012] 図２２は、ステートまでに遷移する時間（RESIDENCY）と、復帰に要する時間（WAKE-UP LATENCY）の最大値の一例を表にして示す図である。図２２は、Intel Xeon CPU E5-26X0 v4（登録商標）におけるC-state情報を表わしている。

[0013] [CPUidle state Governor]

Linux kernel4（登録商標）は、CPU idle state(C-state)を管理するため、２種類のGovernorを用意する。

[0014] 図２３は、CPU idle Governor種別を説明する図である。

図２３に示すように、ladder はtick有りシステム向け、menu はticklessシステム向けで使用されている。

例えば、ticklessシステムで使用される menu は、直近のidle期間の実績から、次のidle期間に適するidle stateの深さを推定する方式である。menuは、idle時間が規則的なワークロードについては効果を得られるが、不規則なワークロードでの効果は限定的となる。

[0015] 図２４は、menu のロジック概要を説明する図である。

図２４左図に示すように、menu は、直近のidle時間を記録する。そして、図２４中図に示すように、次のidle時間を推定する。例えば、図２４左図に示すidle時間をもとに、偏差が小さい場合は平均値 T_avrを次のidle時間と

して採用する。図２４右図に示すように、menu は、次のidle時間の推定をもとに、適するidle state深さを推定する。例えば、次のidle時間推定値 T_avr がidle state(Cx)からの復帰時間 exit latency と同等であれば、深いstateすぎると判断しCx-1のidle stateへ遷移させる。

[0016] 図２３に戻って、２種類のGovernorのladder、menu のうち、いずれのGovernorについても、新たなタスクが到着した際に、CPU idle stateから事前起床を命令し、復帰させて準備をさせておく機構は有していない。このため、深いStateに遷移している状態でのタスク割り当て時には、復帰時間分の遅延時間が発生する問題がある。

先行技術文献

非特許文献

[0017] 非特許文献1：New API(NAPI), [online]，[令和４年６月６日検索]，インターネット 〈URL :https://www.kernel.org/doc/html/latest/admin-guide/pm/cpuidle.html〉

非特許文献2：Daniel Molka & Michael Werner,Wake-up latencies for processor idle states on current x86 processors Robert Schone, [online]，[令和４年６月６日検索]，インターネット 〈 URL :https://link.springer.com/article/10.1007/s00450-014-0270-z#citeas〉

発明の概要

発明が解決しようとする課題

[0018] 図２１に示すように、CPUidle状態には、グレードがあり、ＣＰＵの負荷がない時間が長くなるにつれ、深いsleep状態へ遷移する。深いsleep状態の方がＣＰＵ消費電力は小さくなるが、一方で、それだけ復帰までに要する時間が長延化するため、低遅延の観点で課題となる場合がある。

[0019] 図２５および図２６は、演算に使用するＣＰＵコアのC-state遷移イメージ図である。図２５は、タスクなし時間が短い場合を示し、図２６は、タスクなし時間が長い場合を示す。

図２５に示すように、タスクなし時間が短い場合は、ＣＰＵコアが浅いsleep状態にあり、タスク発生後、復帰までの遅延時間が短い。しかし、図２６に示すように、タスクなし時間が長い場合は、ＣＰＵコアが深いsleep状態（CPUidle状態：グレードＣ６）にあり、タスク発生後、復帰までの遅延時間が長延化する。深いCPU idle stateまで落ちると、復帰するまでに大きな時間を要し、リアルタイム性が損なわれるという課題がある。

[0020] 上記課題は、図１９および図２０に示す基地局（ＢＢＵ）３０のようにリアルタイム性が最優先されるシステムでは、看過できない問題である。

このため、基地局（ＢＢＵ）にあっては、C-stateを無効にする、若しくは、idle stateの遷移をＣ１等の限られた深さに限定する設定を投入する対応が採られる。すなわち、省電力性を犠牲にし、リアルタイム性を指向するチューニングが行われる場合がある。

[0021] 従来の基地局（ＢＢＵ）にあっては、深いC-stateまで落ちたら最後、復帰するまで待つしかなかった。このため、省電力性を犠牲にし、深いところまでは落とさない対策が採られていた。

[0022] このような背景を鑑みて本発明がなされたのであり、本発明は、省電力を達成しつつ、低遅延に演算を行うことを課題とする。

課題を解決するための手段

[0023] 前記した課題を解決するため、処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システムにおいて、前記プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置であって、前記プロセッサの使用率を取得するプロセッサ使用率取得部と、前記プロセッサ使用率取得部が取得した前記プロセッサの使用率をもとに、所定頻度以上で使用する前記プロセッサのコアまたはコア群については、継続的にタスクを割り当てるタスク割当部と、を備えることを特徴とするタスクスケジューラ装置とした。

発明の効果

[0024] 本発明によれば、省電力を達成しつつ、低遅延に演算を行うことができる

。

図面の簡単な説明

[0025] [図1]本発明の実施形態に係る演算システムの概略構成図である。

[図2]本発明の実施形態に係る演算システムのタスクスケジューラ装置をuser spaceに配置した構成例である。

[図3]本発明の実施形態に係る演算システムのタスクスケジューラ装置をOSに配置した構成例である。

[図4]本発明の実施形態に係る演算システムのタスクスケジューラ装置のActiveなＣＰＵコア群の必要数推定ロジックを表にして示す図である。

[図5]本発明の実施形態に係る演算システムのタスクスケジューラ装置のタスク量閾値判定およびＣＰＵ使用率閾値判定動作を説明する図である。

[図6]本発明の実施形態に係る演算システムのタスクスケジューラ装置のタスク流入量に応じた閾値判定イメージを示す図である。

[図7]本発明の実施形態に係る演算システムのタスクスケジューラ装置のＣＰＵ使用率に応じた閾値判定イメージを説明する図である。

[図8]本発明の実施形態に係る演算システムのタスクスケジューラ装置のC-state上限設定を説明する図である。

[図9]本発明の実施形態に係る演算システムのC-state上限設定がない場合／C-state上限設定がある場合を説明する図である。

[図10]本発明の実施形態に係る演算システムのタスクスケジューラ装置の使用可能性の高いＣＰＵコアのプール化を説明する図である。

[図11]本発明の実施形態に係る演算システムのタスクスケジューラ装置の事前起床を説明する図である。

[図12]本発明の実施形態に係る演算システムのタスクスケジューラ装置の事前起床を説明する図である。

[図13]本発明の実施形態に係る演算システムのタスクスケジューラ装置のタスクスケジューリング処理を示すフローチャートである。

[図14]本発明の実施形態に係る演算システムのタスクスケジューラ装置の機

能を実現するコンピュータの一例を示すハードウェア構成図である。

[図15]本発明の実施形態に係る演算システムを、汎用Linux kernel（登録商標）およびＶＭ構成のサーバ仮想化環境における割込モデルに適用した例を示す図である。

[図16]本発明の実施形態に係る演算システムを、コンテナ構成のサーバ仮想化環境における割込モデルに適用した例を示す図である。

[図17]本発明の実施形態に係る演算システムを、汎用Linux kernel（登録商標）およびＶＭ構成のサーバ仮想化環境における割込モデルに適用した例を示す図である。

[図18]本発明の実施形態に係る演算システムを、コンテナ構成のサーバ仮想化環境における割込モデルに適用した例を示す図である。

[図19]無線アクセスシステムの概要を説明する図である。

[図20]基地局における無線アクセス処理のタスク管理例を説明する図である。

[図21]C-stateの状態の一例を表にして示す図である。

[図22]ステートまでに遷移する時間（RESIDENCY）と、復帰に要する時間（WAKE-UP LATENCY）の最大値の一例を表にして示す図である。

[図23]CPU idle Governor種別を説明する図である。

[図24]menu のロジック概要を説明する図である。

[図25]演算に使用するＣＰＵコアのC-state遷移イメージ図である。

[図26]演算に使用するＣＰＵコアのC-state遷移イメージ図である。

発明を実施するための形態

[0026] 以下、図面を参照して本発明を実施するための形態（以下、「本実施形態」という）における演算システム等について説明する。

［概要］

図１は、本発明の実施形態に係る演算システムの概略構成図である。図２０と同一構成部分には、同一符号を付している。

本実施形態は、演算システムとしてＣＰＵに適用した例である。ＣＰＵ以

外にも、ＧＰＵ（Graphic Processing Unit），ＦＰＧＡ（Field Programmable Gate Array），ＡＳＩＣ（Application Specific Integrated Circuit）等のプロセッサに、idle stateの機能がある場合には、同様に適用可能である。

[0027] 図１に示すように、演算システム１０００は、ハードウェア（ＨＷ）３１と、ハードウェア３１上のＣＰＵコア（CPUcore #0, CPUcore #1, …）３２（プロセッサ）と、ＯＳ３３と、Ｌ１，Ｌ２，Ｌ３プロトコル無線信号処理アプリケーション１－１，１－２，１－３（総称する場合は、ＡＰＬ１と呼ぶ）と、タスクスケジューラ装置１００と、を備える。

演算システム１０００は、処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システムである。演算システム１０００は、プロセッサが、複数のコア群を有しており、プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置１００を備える。

[0028] ［タスクスケジューラ装置１００］

タスクスケジューラ装置１００は、管理部１１０と、タスク管理部１２０と、ＣＰＵ idle state制御部１３０と、を備える。

[0029] 〈管理部１１０〉

管理部１１０は、運用者向け設定部１１１を有する。

運用者向け設定部１１１は、運用者がC-stateに関するパラメータを設定する。例えば、運用者向け設定部１１１は、管理者が、C-state上限設定投入から反映までのタイムラグを考慮して、事前にC-state上限設定を投入する。

[0030] 〈タスク管理部１２０〉

タスク管理部１２０は、タスク優先度付与部１２１と、タスク量・周期 予測部１２２（プロセッサ使用率取得部）と、タスクＣＰＵ割当部１２３（タスク割当部）と、を有する。

[0031] タスク優先度付与部１２１は、ＡＰＬ１から送られたタスクに対し、必要に応じてタスクの優先度を付与する。タスク管理部１２０は、この優先度に応じて、ＣＰＵコアへの割り当ての優先制御を実行する。

[0032] ここで、ＣＰＵ状態を、もっとも精確に表しているのは、「使用しているＣＰＵコアの単位時間当たりのＣＰＵ使用率」と考えられる。「使用しているＣＰＵコアの単位時間当たりのＣＰＵ使用率」は、ほぼタスク量で推定することができる。ＣＰＵがどれだけ忙しいかは、タスク量で推計することができるが、タスク量だけでは測れない、ＣＰＵが苦手とする処理がどれだけあるかにもよる。さらに、プロセッサが、ＣＰＵ以外のＧＰＵ，ＦＰＧＡ，ＡＳＩＣである場合には、プロセッサが得意／苦手とする処理についても変わる。本実施形態では、「使用しているＣＰＵコアの単位時間当たりのＣＰＵ使用率」に関連する物理量として、タスク量を用いている。

[0033] タスク量・周期 予測部１２２は、プロセッサの使用率を取得する。具体的には、タスク量・周期 予測部１２２は、図４に示す判断ロジックに従い、単位時間当たりのタスク量や、ＣＰＵ使用率を計測し、演算に使用するＣＰＵコア数を増やす／減らす必要があるかを判断する。

タスク量・周期 予測部１２２は、新たに使用するＣＰＵコアに対して、事前起床を行う。ＣＰＵコアにより何かしらの処理が実行されれば、C-stateはＣ０に戻るＣＰＵ製品が多い。そのため、小さな演算や標準出力に文字列を出力する等の処理を該当ＣＰＵコアで実行する方法がある。

[0034] タスクＣＰＵ割当部１２３は、タスク量・周期 予測部１２２（プロセッサ使用率取得部）が取得したプロセッサの使用率をもとに、所定頻度以上で使用するプロセッサのコアまたはコア群については、継続的にタスクを割り当てる。タスクＣＰＵ割当部１２３は、例えば、使用するＣＰＵコアを減らす。ＣＰＵコアを減らす方法として、以降使用しないＣＰＵコアを決定し、以降当該ＣＰＵコアへタスクを割り当てないようにする。

タスクＣＰＵ割当部１２３は、到着したタスクを、ＣＰＵコアに割り当て(ＣＰＵコアへの割り当ては、使用しているＣＰＵコアをラウンドロビンしてもよいし、ＣＰＵコア毎に用意したタスクキューの残数が少ないＣＰＵコアに割り当ててもよい)、ＣＰＵコア毎に用意したタスクキューにタスクをスケジューリングする。

この時、付与された優先度に応じて、優先度の高いタスクを空いているＣＰＵコアへ優先的に割り当ててもよい。また、ＣＰＵコア毎のタスクキューを、優先度に応じて並び替えてもよい。

[0035] 〈ＣＰＵ idle state制御部１３０〉

ＣＰＵ idle state制御部１３０は、ＣＰＵ事前起床部１３１（事前起床部）と、C-state設定部１３２（動作状態設定部）と、を有する。

ＣＰＵ事前起床部１３１は、新たに使用するコア、または、所定期間使用せずに再び使用するコアに対しては、タスクを割り当てる際に、プロセッサの動作状態を元の状態に復帰させる事前起床を行う（図１の符号ｃ参照）。

[0036] C-state設定部１３２は、所定頻度以上で使用するプロセッサのコアまたはコア群については、プロセッサの動作状態をより深い状態に遷移できないように上限を設定する。具体的には、C-state設定部１３２は、新たに使用するＣＰＵコアに対して、C-state上限設定を投入する。上限stateは、運用者が設定することにより決定する（図１の符号ｂ参照）。

なお、定常的にタスクが流入し、継続的にタスクを割り当て続けられる場合は、ActiveなＣＰＵコア群は深いC-stateに遷移しないため、上限設定は投入しなくてもよい場合もある。

[0037] ここで、事前起床ではなく、新たに使用するＣＰＵコアに対して、タスクを割り当てる前にC-state上限設定を入れて、浅いstateに戻しておいてもよい。この場合、C-state上限設定投入から反映までのタイムラグを考慮して、事前にC-state上限設定を投入する。

C-state設定部１３２は、使用しないと決定したＣＰＵコアに対して、C-state上限設定が入っている場合は、上限設定を解除する。

C-state設定部１３２は、タスクＣＰＵ割当部１２３（タスク割当部）が、タスクを割り当てなかったコアまたはコア群については、プロセッサの動作状態をより深い状態に遷移できるように設定する。

[0038] ［タスクスケジューラ装置の配置］

図２および図３は、図１のタスクスケジューラ装置１００の配置を説明す

る図である。

・user spaceへのタスクスケジューラ装置の配置

図２は、図１のタスクスケジューラ装置１００をuser spaceに配置した構成例である。

図２に示す演算システム１０００は、User space６０にタスクスケジューラ装置１００が配置される。演算システム１０００は、User space６０に配置されたパケット処理ＡＰＬ１を実行する。演算システム１０００は、OSを備えるサーバ上で、パケット処理ＡＰＬ１を実行する。

[0039] 本発明を、Intel DPDK（Intel Data Plane Development Kit）（ＤＰＤＫ）（登録商標）のように、user space６０にthreadがある場合に適用することができる。

[0040] ・OSへのタスクスケジューラ装置の配置

図３は、図１のタスクスケジューラ装置１００をOS５０に配置した構成例である。

図３に示す演算システム１０００は、OS５０にタスクスケジューラ装置１００が配置される。演算システム１０００は、OS５０を備えるサーバ上で、パケット処理ＡＰＬ１を実行する。

[0041] 本発明を、New API(NAPI)（登録商標）（非特許文献１）やＫＢＰ（kernel-based virtual machine）のように、kernel内部にthreadがある場合に適用することができる。

[0042] ［ActiveなＣＰＵコア群の必要数推定ロジック］

ActiveなＣＰＵコア群（図１の破線囲みａ参照）の必要数推定ロジックについて説明する。

本発明者らは、ActiveなＣＰＵコア群には、継続してタスクを割り当て、それ以外のＣＰＵコアは、深いC-stateに遷移させて消費電力量を削減させることを考察した。

そのためには、ActiveなＣＰＵコア群の管理に際して、流入するタスク量等をもとに、必要なＣＰＵコア数を算出する必要がある。

[0043] 図４は、ActiveなＣＰＵコア群の必要数推定ロジックを表にして示す図である。

図４に示すように、推定ロジック判断は、「１．タスクの到着パターンによる判定」と「２．ＣＰＵ使用率による判定」とがあり、それぞれ、閾値判定、周期性、機械学習による判定を行う。

図４に示すように、判断ロジック毎に、適用性のあるタスクが異なるため、図４の表の右列に記載した適用性を考慮して、ロジックを選択して使い分ける。また、複数の判断ロジックを併用してもよい。

[0044] 以下、上述のように構成された演算システム１０００の動作を説明する。

［タスクスケジューラ装置１００動作の基本的な考え方］

本発明は、リアルタイム性の求められる演算システムにおいて、ＣＰＵコアへタスクを割り当てるタスクスケジューラが、ＣＰＵコアのC-stateの状態を考慮して、タスクの割り当てを行う。

[0045] 特徴<１>：継続タスク割当

使用頻度の高いＣＰＵコア群を確保し、継続してタスクを割り当てることで、深いC-stateへ遷移することを防ぐ（図５～図７）。

図５の破線囲みａに示すように、タスク量に応じて、頻繁に利用するＣＰＵコア群（ActiveなＣＰＵコア群）を、図４に示す判断ロジックにより確保し、該当ＣＰＵコア群へ継続的にタスクを割り当てる。これにより、深いC-stateへ遷移することを防ぎ、C-stateからの復帰遅延を軽減する。ActiveなＣＰＵコア以外は、深いC-stateに遷移でき、省電力を図ることができる。

[0046] 特徴<２>：C-state上限設定

タスクを継続して割り当てられない場合の対応策として、使用頻度の高いＣＰＵコア群に対して、遷移可能なC-state上限を設定する（図８～図１０）。

ActiveなＣＰＵコアへ継続してタスクを割り当てられない場合の対応として、ActiveなＣＰＵコア群へ遷移可能なC-state上限を設定することで、深いC-stateへ遷移できないようにし、復帰遅延を軽減する。

[0047] 特徴<３>：事前起床

深いstateに遷移したＣＰＵコアへタスクを割り当てる前に、事前起床をすることで、深いstateからの復帰時間を軽減する（図１１～図１２）。

新たに使用するＣＰＵコア（久しぶりに使用するＣＰＵコア）に対して、タスクを割り当てる前に、事前起床を行うことで、深いC-stateからの復帰時間を軽減する。

以下、特徴<１>、特徴<２>、特徴<３>について順に説明する。

[0048] ［タスク量閾値判定およびＣＰＵ使用率閾値判定］（特徴<１>：継続タスク割当）

図５は、タスクスケジューラ装置１００のタスク量閾値判定およびＣＰＵ使用率閾値判定動作を説明する図である。

[0049] <タスク量閾値判定動作>

まず、タスク量閾値判定動作を説明する。

タスク管理部１２０は、タスク量閾値判定動作を行う。

図５の破線矢印ｄに示すように、タスク優先度付与部１２１は、必要に応じてタスクの優先度を付与する。

タスク量・周期 予測部１２２は、図４に示す判断ロジックに従い、単位時間当たりのタスク量や、ＣＰＵ使用率を計測し、演算に使用するＣＰＵコア数を増やす／減らす必要があるかを判断する。

[0050] 図６は、タスク流入量に応じた閾値判定イメージを示す図である。

タスク管理部１２０は、図６に示すタスク流入量に従ってＣＰＵコアを管理する。

図６において、単位時間当たりのタスク流入量：W_inputと使用するＣＰＵコア数とを対応付ける。例えば、W_inputが「ｄｄ～ｅｅ」の場合、使用するＣＰＵコア数は「４」であり、ActiveなＣＰＵコア群（図５の破線囲みａ参照）は、「４」である。

[0051] 図５に戻って、タスクＣＰＵ割当部１２３は、到着したタスクを、ＣＰＵコアに割り当て、ＣＰＵコア毎に用意したタスクキューにタスクをスケジュ

ーリングする。この時、付与された優先度に応じて、優先度の高いタスクを空いているＣＰＵコアへ優先的に割り当てる。また、ＣＰＵコア毎のタスクキューを、優先度に応じて並び替える。

[0052] タスク管理部１２０は、演算時間が固定時間で完了するタスクについては、単位時間のタスク流入量を、図６の閾値判定表に従って、タスク流入量の演算に必要なＣＰＵコア数を設計する。また、タスク流入量が、図６の閾値判定表の閾値（W_inputの閾値）を超えた場合に、使用するＣＰＵコア数を増やす必要があるかを判定することが可能となる。図６においては、ｄｄ＜W_input＜ｅｅの状態であり、ＣＰＵコアは４つ使用している。タスク流入量が、W_input≧ｅｅとなった場合に、ＣＰＵコアを４つ→５つに増やす。

[0053] <必要なＣＰＵコア数算出例>

次に、必要なＣＰＵコア数算出例について説明する。

タスク管理部１２０（図５）は、必要なＣＰＵコア数算出を行う。

タスクの演算時間が固定的な（若しくは、変動が小さい）場合には、サービスで許容される最大待ち時間Ｔｗを定義すれば、待ち行列理論を用いて、最大待ち時間を満足するために必要なＣＰＵコア数Ｎを算出することが可能になる。下記計算式の式（３）には、C-stateからの復帰時間は考慮に含まないが、本実施形態によりC-stateからの復帰時間を限りなくゼロに近づけることができれば、考慮不要となる。

[0054] ・必要なＣＰＵコア数算出例

λ[個/s]：単位時間当たりのタスク流入量（W_input）

μ[個/s]：単位時間当たりにシステムが処理できるタスクの数

Ｔｓ[s]：システムが１つのタスクの処理に要する時間

t_s[s]：１つのＣＰＵコアで１つのタスク演算に必要な時間

Ｔｗ[s]：サービスで許容される最大待ち時間

Ｎ[個]：使用しているＣＰＵコア数

ρ＝λ／μ：混み具合

[0055] μ＝Ｎ／t_s

$$Ts = 1/\mu = t\_s/N$$

[0056] $$Tw = (\rho/(1-\rho)) * Ts \quad \cdots (1)$$

[0057] 上記式（１）に、ρとＴｓを代入すると、式（２）となる。

$$Tw = ((\lambda/\mu)/(1-(\lambda/\mu))) * (t\_s/N) \quad \cdots (2)$$

[0058] 上記式（２）に、μを代入し解くと、式（３）得る。

$$N^2 - \lambda \cdot t\_s \cdot N - \lambda \cdot t\_s^2/Tw = 0 \quad \cdots (3)$$

この方程式を満たす最小の自然数を求めればよい。

[0059] 〈ＣＰＵ使用率閾値判定動作〉

次に、ＣＰＵ使用率閾値判定動作を説明する。

図７は、ＣＰＵ使用率に応じた閾値判定イメージを説明する図である。縦軸は、使用しているＣＰＵコアのＣＰＵ使用率平均、横軸は時間である。

ＣＰＵ使用率平均の１００％に対して、２つの閾値、Threshold_upper（上限）（図７の破線ｅ参照）および、Threshold_base（下限）（図７の破線ｆ参照）を設定する。

タスク管理部１２０（図５）は、ＣＰＵ使用率平均が、上限を超えた場合、使用するＣＰＵコアを増やす。また、ＣＰＵ使用率平均が、下限を下回ったら、使用するＣＰＵコアを減らす。

[0060] 使用しているＣＰＵコアのＣＰＵ使用率平均が、Threshold_upper（上限）を上回る場合は、使用しているＣＰＵコアが足りなくなっている予兆であり、新しくＣＰＵコア数を増やすようにする判断ができる。一方、使用しているＣＰＵコアのＣＰＵ使用率平均が、Threshold_base（下限）を下回る場合は、ＣＰＵコアが過剰であり、使用しているＣＰＵコアを縮退させる判断ができる。

[0061] ここで、上記、新しくＣＰＵコア数を増やす場合、当該ＣＰＵコアは、いままで使用していなかったＣＰＵコアであるため、深いC-stateまで落ちていることが想定される。特に、このような深いC-state状態に落ちているＣＰＵコアを復帰させる場合の復帰時間の算出方法については、上記〈必要なＣＰＵコア数算出例〉で説明した。

以上、タスク量閾値判定およびＣＰＵ使用率閾値判定について説明した。

[0062] ［C-state上限設定］（特徴<２>：C-state上限設定）

<C-state上限設定動作>

図８は、タスクスケジューラ装置１００のC-state上限設定を説明する図である。

ＣＰＵ idle state制御部１３０のC-state設定部１３２は、使用頻度の高いＣＰＵコア群に対して、遷移可能なC-state上限を設定する（図８の符号ｂ参照）。

[0063] 使用する頻度の高いＣＰＵコアについては、タスクを割り当てる可能性が高いため、深いidle stateに遷移させても、該当stateの滞在期間が短く省電力効果はあまり享受できない。省電力効果が享受できないにも関わらず、idle stateからの復帰時間オーバーヘッドが発生してしまうデメリットが生じる場合がある。

[0064] 本実施形態では、この問題に対して、使用頻度の高いＣＰＵコアについては、遷移可能なＣＰＵ idle stateの上限（例えば、図２２のＣ１やＣ１Ｅ等参照）を定めておき、該当stateよりも深いstateに遷移できないように設定する。これにより、タスク割り当て時に浅いstateからの復帰にすることができ、復帰遅延時間を低減できる。

[0065] ＣＰＵ idle stateの上限を設定／解除するＣＰＵコアの対象は、図４に示す判断ロジックを用いることで、使用するＣＰＵコア数を判断可能である。C-state設定部１３２は、図４に示す判断ロジックを使用して、ＣＰＵ idle stateの上限を動的に設定する。

[0066] また、新たに（久しぶりに）使用するＣＰＵコアを増やす場合に、新たに増やすＣＰＵコアに対して、タスクを割り当てる前に。C-state上限設定を行い、浅いC-stateに戻しておいてもよい。

[0067] 図９は、C-state上限設定イメージを説明する図である。図９上図は、C-state上限設定がない場合を示し、図９下図は、C-state上限設定がある場合を示す。

図９上図に示すように、C-state上限設定がない場合は、CPUidle状態（グレードＣ２）まで落ち、Ｃ０復帰までの遅延時間が長延化する。

これに対して、図９下図に示すように、使用頻度の高いＣＰＵコア群に対して、C-state上限設定することで（図９下図の符号ｇ参照）、深いstateに遷移できないようにし、浅いstateからの復帰の際の復帰遅延時間を低減する。

[0068] 〈使用可能性の高いＣＰＵコアのプール化〉

次に、C-state上限設定の拡張機能として、使用可能性の高いＣＰＵコアのプール化について説明する。

図１０は、タスクスケジューラ装置１００の使用可能性の高いＣＰＵコアのプール化を説明する図である。

図１０に示すように、今後使用が見込まれる場合は、浅いidle stateにしか落ちないように設定したＣＰＵコア群（図１０の破線囲みｈ）を準備しておいてもよい。例えば、図１０に示すＣＰＵコア（CPUcore #4, CPUcore #5）３２は、Ｃ１までしか遷移できないＣＰＵコア群としてプール化しておく態様をとる。

[0069] 夜間／日中のトラヒック変動に対する処理のように、タスクの傾向から、今後使用する可能性の高いＣＰＵコアを予測できる場合には、使用する可能性の高いＣＰＵコア群（図１０の破線囲みｈ）に対して、浅いidle stateにしか遷移できないように設定（例：Ｃ１まで等）しておく。そして、必要になった際に、直ちに使用できるように、アクティブスタンバイなＣＰＵコアをプール化して準備しておく。

このように、今後使用が見込まれる場合は、浅いidle stateにしか落ちないように設定したＣＰＵコア群を準備しておく(例えば、Ｃ１までしか遷移できないＣＰＵコア群をプール化しておく）態様をとる。

以上、C-state上限設定について説明した。

[0070] ［事前起床］（特徴〈３〉：事前起床）

図１１は、タスクスケジューラ装置１００の事前起床を説明する図である

。

ＣＰＵ事前起床部１３１は、深いＣＰＵ idle state(C-state)に落ちているＣＰＵコアを使用する際に、ＣＰＵコアを事前起床させる（図１１の符号ｃ参照）。

[0071] 図１２は、タスクスケジューラ装置１００の事前起床を説明する図である。図１２上図は、既存技術の場合を示し、図１２下図は、タスクスケジューラ装置１００の事前起床を行う場合を示す。

図１２上図に示すように、既存技術の場合（事前起床がない場合）は、タスク割り当て（図１２上図の符号ｉ）から、演算を開始するまでの遅延が大きい（図１２上図の符号ｊ）。

これに対して、図１２下図に示すように、事前起床がある場合は、事前起床により（図１２下図の符号ｃ）、タスク割り当てから演算開始までが低遅延となる（図１２下図の符号ｋ）。

[0072] このように、使用頻度の低いＣＰＵコアは、深いC-stateに遷移させ、消費電力を削減することが可能となり、かつ、深いC-stateからの復帰に伴う遅延時間の影響を軽減することの両立が可能となる。

[0073] ［タスクスケジューラ装置１００の動作フロー］

図１３は、タスクスケジューラ装置１００のタスクスケジューリング処理を示すフローチャートである。

ステップＳ１１でＡＰＬ１（図１）は、タスク管理部１２０（図１）へタスクを登録する。

[0074] ステップＳ１２でタスク優先度付与部１２１（図１）は、必要に応じてタスクの優先度を付与する。タスクＣＰＵ割当部１２３（図１）は、この優先度に応じて、ＣＰＵコアへの割り当ての優先制御を実行する。

[0075] ステップＳ１３でタスク量・周期 予測部１２２（図１）は、図４に示す判断ロジックに従い、単位時間当たりのタスク量や、ＣＰＵ使用率を計測し、演算に使用するＣＰＵコア数を増やす／減らす必要があるか判断する（判断後の分岐は、下記ステップＳ１４，ステップＳ１５）。

[0076] ステップＳ１４でＣＰＵコア数を減らす必要がある場合には（Ｓ１４：Ｙｅｓ）、ステップＳ１７に進み、ＣＰＵコア数を減らす必要がない場合には（Ｓ１４：Ｎｏ）、ステップＳ１５に進む。

[0077] ステップＳ１５でＣＰＵコア数を増やす必要がある場合には（Ｓ１５：Ｙｅｓ）、ステップＳ１９に進み、ＣＰＵコア数を増やす必要がない場合には（Ｓ１５：Ｎｏ）、ステップＳ１６に進む。

[0078] ステップＳ１６でタスクＣＰＵ割当部１２３（図１）は、到着したタスクを、ＣＰＵコアに割り当て、ＣＰＵコア毎に用意したタスクキューにタスクをスケジューリングして本フローの処理を終了する。ＣＰＵコアへの割り当ては、使用しているＣＰＵコアをラウンドロビンしてもよいし、ＣＰＵコア毎に用意したタスクキューの残数が少ないＣＰＵコアに割り当ててもよい。

[0079] また、タスクＣＰＵ割当部１２３（図１）は、付与された優先度に応じて、優先度の高いタスクを空いているＣＰＵコアへ優先的に割り当ててもよいし、ＣＰＵコア毎のタスクキューを、優先度に応じて並び替えてもよい。

[0080] 上記ステップＳ１４でＣＰＵコア数を減らす必要がある場合には、ステップＳ１７でタスクＣＰＵ割当部１２３（図１）は、使用するＣＰＵコアを減らす。例えば、ＣＰＵコアを減らす方法として、以降使用しないＣＰＵコアを決定し、以降該当ＣＰＵコアへタスクを割り当てないようにする。

[0081] ステップＳ１８でC-state設定部１３２（図１）は、使用しないと決定したＣＰＵコアに対して、C-state上限設定が入っている場合は、C-state上限設定を解除して本フローの処理を終了する。

[0082] 上記ステップＳ１５でＣＰＵコア数を増やす必要がある場合には、ステップＳ１９でタスク量・周期 予測部１２２（図１）は、新たに使用するＣＰＵコアに対して、事前起床を行う。

新たに使用するＣＰＵコアに対して、事前起床を行う理由は、下記の通りである。すなわち、ＣＰＵコアに何かしらの処理が実行されれば、C-stateはＣ０に戻るＣＰＵ製品が多い。そのため、小さな演算や標準出力に文字列を出力する等の軽微な処理を該当ＣＰＵコアで実行しておくことで、該当ＣＰ

Ｕコアが深いC-stateに遷移することを抑制する。

[0083] ここで、事前起床ではなく、〈C-state上限設定〉（図８）で述べた、新たに使用するＣＰＵコアに対して、タスクを割り当てる前にC-state上限設定を行い、浅いstateに戻しておいてもよい。この場合、C-state上限設定投入から反映までのタイムラグを考慮して、事前にC-state上限設定を投入する。

[0084] ステップＳ２０でC-state設定部１３２は、新たに使用するＣＰＵコアに対して、C-state上限設定を投入して本フローの処理を終了する。上限stateは、あらかじめ運用者が設定することにより決定する。

[0085] なお、定常的にタスクが流入し、継続的にタスクを割り当て続けられる場合は、ActiveなＣＰＵコア群は深いC-stateに遷移しないため、C-state上限設定は投入しなくてもよい場合もある。

[0086] ［ハードウェア構成］

上記実施形態に係るタスクスケジューラ装置１００，１００Ａ（図１、図１０）は、例えば図１４に示すような構成のコンピュータ９００によって実現される。

図１４は、タスクスケジューラ装置１００，１００Ａ（図１、図１０）の機能を実現するコンピュータ９００の一例を示すハードウェア構成図である。

コンピュータ９００は、ＣＰＵ９０１、ＲＯＭ９０２、ＲＡＭ９０３、ＨＤＤ９０４、通信インターフェイス（Ｉ／Ｆ：Interface）９０６、入出力インターフェイス（Ｉ／Ｆ）９０５、およびメディアインターフェイス（Ｉ／Ｆ）９０７を有する。

[0087] ＣＰＵ９０１は、ＲＯＭ９０２またはＨＤＤ９０４に格納されたプログラムに基づいて動作し、タスクスケジューラ装置１００，１００Ａ（図１、図１０）の各部の制御を行う。ＲＯＭ９０２は、コンピュータ９００の起動時にＣＰＵ９０１によって実行されるブートプログラムや、コンピュータ９００のハードウェアに依存するプログラム等を格納する。

[0088] ＣＰＵ９０１は、入出力Ｉ／Ｆ９０５を介して、マウスやキーボード等の

入力装置９１０、および、ディスプレイ等の出力装置９１１を制御する。ＣＰＵ９０１は、入出力Ｉ／Ｆ９０５を介して、入力装置９１０からデータを取得するとともに、生成したデータを出力装置９１１へ出力する。なお、プロセッサとしてＣＰＵ９０１とともに、ＧＰＵ（Graphics Processing Unit）等を用いてもよい。

[0089] ＨＤＤ９０４は、ＣＰＵ９０１により実行されるプログラムおよび当該プログラムによって使用されるデータ等を記憶する。通信Ｉ／Ｆ９０６は、通信網（例えば、ＮＷ（Network）９２０）を介して他の装置からデータを受信してＣＰＵ９０１へ出力し、また、ＣＰＵ９０１が生成したデータを、通信網を介して他の装置へ送信する。

[0090] メディアＩ／Ｆ９０７は、記録媒体９１２に格納されたプログラムまたはデータを読み取り、ＲＡＭ９０３を介してＣＰＵ９０１へ出力する。ＣＰＵ９０１は、目的の処理に係るプログラムを、メディアＩ／Ｆ９０７を介して記録媒体９１２からＲＡＭ９０３上にロードし、ロードしたプログラムを実行する。記録媒体９１２は、ＤＶＤ（Digital Versatile Disc）、ＰＤ（Phase change rewritable Disk）等の光学記録媒体、ＭＯ（Magneto Optical disk）等の光磁気記録媒体、磁気記録媒体、導体メモリテープ媒体又は半導体メモリ等である。

[0091] 例えば、コンピュータ９００が本実施形態に係る一装置として構成されるタスクスケジューラ装置１００，１００Ａ（図１、図１０）として機能する場合、コンピュータ９００のＣＰＵ９０１は、ＲＡＭ９０３上にロードされたプログラムを実行することによりタスクスケジューラ装置１００，１００Ａの機能を実現する。また、ＨＤＤ９０４には、ＲＡＭ９０３内のデータが記憶される。ＣＰＵ９０１は、目的の処理に係るプログラムを記録媒体９１２から読み取って実行する。この他、ＣＰＵ９０１は、他の装置から通信網（ＮＷ９２０）を介して目的の処理に係るプログラムを読み込んでもよい。

[0092] ［適用例］

（User space６０にタスクスケジューラ装置を配置する形態）

図２に示すように、User space６０にタスクスケジューラ装置１００を配置した構成例に適用できる。この場合、ＯＳは限定されない。また、サーバ仮想化環境下であることも限定されない。したがって、演算システムは、図１５および図１６に示す各構成に適用が可能である。

[0093] 〈ＶＭ構成への適用例〉

ＮＦＶ（Network Functions Virtualization：ネットワーク機能仮想化）による仮想化技術の進展などを背景に、サービス毎にシステムを構築して運用することが行われている。また、上記サービス毎にシステムを構築する形態から、サービス機能を再利用可能なモジュール単位に分割し、独立した仮想マシン（ＶＭ：Virtual Machineやコンテナなど）環境の上で動作させることで、部品のようにして必要に応じて利用し運用性を高めるといったＳＦＣ（Service Function Chaining）と呼ばれる形態が主流となりつつある。

[0094] 仮想マシンを構成する技術としてLinux（登録商標）とＫＶＭ（kernel-based virtual machine）で構成されたハイパーバイザー環境が知られている。この環境では、ＫＶＭモジュールが組み込まれたHost OS（物理サーバ上にインストールされたＯＳをHost OSと呼ぶ）がハイパーバイザーとしてカーネル空間と呼ばれるユーザ空間とは異なるメモリ領域で動作する。この環境においてユーザ空間にて仮想マシンが動作し、その仮想マシン内にGuest OS（仮想マシン上にインストールされたＯＳをGuest OSと呼ぶ）が動作する。

[0095] Guest OSが動作する仮想マシンは、Host OSが動作する物理サーバとは異なり、（イーサネット（登録商標）カードデバイスなどに代表される）ネットワークデバイスを含むすべてのＨＷ（hardware）が、ＨＷからGuest OSへの割込処理やGuest OSからハードウェアへの書き込みに必要なレジスタ制御となる。このようなレジスタ制御では、本来物理ハードウェアが実行すべき通知や処理がソフトウェアで擬似的に模倣されるため、性能がHost OS環境に比べ、低いことが一般的である。

[0096] 図１５は、汎用Linux kernel（登録商標）およびＶＭ構成のサーバ仮想化環境における割込モデルに、演算システム１０００Ａを適用した例を示す図

である。図１と同一構成部分には、同一符号を付している。

図１５に示すように、演算システム１０００Ａは、仮想マシンおよび仮想マシン外に形成された外部プロセスが動作可能なHost OS８０を備え、Host OS８０は、Kernel８１およびDriver８２を有する。また、演算システム１０００Ａは、Host OS８０に接続されたＨＷ７０のＣＰＵ７１、ハイパーバイザー（ＨＶ）９０に組み込まれたＫＶＭモジュール９１を有する。さらに、演算システム１０００Ａは、仮想マシン内で動作するGuest OS９５を備え、Guest OS９５は、Kernel９６およびDriver９７を有する。

そして、演算システム１０００Ａは、Guest OS９５に接続されUser space６０に配置されたタスクスケジューラ装置１００を備える。

[0097] このようにすることにより、ＶＭの仮想サーバ構成のシステムにおいて、HostOS８０とGuest OS９５とのいずれのOSにおいても、省電力を達成しつつ、低遅延に演算を行うことができる。

[0098] <コンテナ構成への適用例>

図１６は、コンテナ構成のサーバ仮想化環境における割込モデルに、演算システム１０００Ｂを適用した例を示す図である。図１および図１５と同一構成部分には、同一符号を付している。

図１６に示すように、演算システム１０００Ｂは、図１５のGuest OS９５をContainer９８に代えた、コンテナ構成を備える。Container９８は、vNIC（仮想NIC）を有する。

[0099] コンテナなどの仮想サーバ構成のシステムにおいて、省電力を達成しつつ、低遅延に演算を行うことができる。

以上、user space６０にタスクスケジューラ装置１００を配置する形態について説明した。次にkernelにタスクスケジューラ装置１００を配置する形態について説明する。

[0100] (OS内のkernelにタスクスケジューラ装置を配置する形態)

図３に示すように、OS５０内にタスクスケジューラ装置１００を配置した構成例に適用できる。この場合、ＯＳは限定されない。また、サーバ仮想化

環境下であることも限定されない。したがって、演算システムは、図１７および図１８に示す各構成に適用が可能である。

[0101] 〈ＶＭ構成への適用例〉

図１７は、汎用Linux kernel（登録商標）およびＶＭ構成のサーバ仮想化環境における割込モデルに、演算システム１０００Ｃを適用した例を示す図である。図１および図１５と同一構成部分には、同一符号を付している。

図１７に示すように、演算システム１０００Ｃは、Host OS８０のKernel８１内にタスクスケジューラ装置１００が配置され、Guest OS９５のKernel９６内にタスクスケジューラ装置１００が配置される。

[0102] このようにすることにより、ＶＭの仮想サーバ構成のシステムにおいて、HostOS８０とGuest OS９５いずれのOSにおいても、省電力を達成しつつ、低遅延に演算を行うことができる。

[0103] 〈コンテナ構成への適用例〉

図１８は、コンテナ構成のサーバ仮想化環境における割込モデルに、演算システム１０００Ｄを適用した例を示す図である。図１および図１６と同一構成部分には、同一符号を付している。

図１９に示すように、演算システム１０００Ｄは、Host OS８０のKernel８１内にタスクスケジューラ装置１００が配置される。

[0104] コンテナなどの仮想サーバ構成のシステムにおいて、省電力を達成しつつ、低遅延に演算を行うことができる。

[0105] 〈ベアメタル構成（非仮想化構成）への適用例〉

本発明は、ベアメタル構成のように非仮想化構成のシステムに適用できる。非仮想化構成のシステムにおいて、省電力を達成しつつ、低遅延に演算を行うことができる。

[0106] 〈ＣＰＵ以外のプロセッサへの適用〉

本発明は、ＣＰＵ以外にも、ＧＰＵ／ＦＰＧＡ／ＡＳＩＣ等のプロセッサに、idle stateの機能がある場合には、同様に適用可能である。

[0107] 〈ＣＰＵ動作周波数制御への適用〉

本発明は、実施形態に係る事前起床を、プロセッサ動作周波数が低くなっている際に、タスク割り当ての前に事前に周波数を元に戻しておく仕組みへの応用も可能である。

[0108] 〈ＣＰＵ以外の機能に対する適用〉

本発明は、ＣＰＵ以外にも、メモリやストレージ（例えば、ＨＤＤ，ＳＳＤ）などＣＰＵ以外の機能に対する適用も可能である。さらに、外部接続周辺機器等への部品が省電力モードに入っていた際に、事前に起床させて使用するまでに準備させることに応用することも可能である。

[0109] ［効果］

以上説明したように、処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システム１０００（図１、図１０）において、プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置１００，１００Ａ（図１、図１０）であって、プロセッサの使用率を取得するプロセッサ使用率取得部（タスク量・周期 予測部１２２）と、プロセッサ使用率取得部が取得したプロセッサの使用率をもとに、所定頻度以上で使用するプロセッサのコアまたはコア群については、継続的にタスクを割り当てるタスク割当部（タスクＣＰＵ割当部１２３）と、を備える。

[0110] このようにすることにより、プロセッサの使用率（例えば、タスク量）に応じて、頻繁に利用するＣＰＵコア群（ActiveなＣＰＵコア群）を図４に示す判断ロジックにより確保し、当該ＣＰＵコア群へ継続的にタスクを割り当てることで、深いC-stateへ遷移することを防ぐ。これにより、C-stateからの復帰遅延を軽減することができる。一方、ActiveなＣＰＵコア以外は、深いC-state（例えばＣ６）に遷移できるので、省電力を図ることができる。その結果、計算システム１０００全体からみて、省電力を達成しつつ、低遅延に演算を行うことができる。

[0111] ＣＰＵコアに対して、深いC-stateまで落とすことができるので、電力削減効果を最大限享受することができる一方、深いC-stateまで落ち復帰に時間がかかる課題を回避することができる。

[0112] また、タスク量やＣＰＵ使用率に応じて、動的に使用するＣＰＵコア数を増減することが可能となるため、タスク量に応じた動的なスケールイン／アウトが可能となる。

[0113] また、タスク量やＣＰＵ使用率に応じて、使用するＣＰＵコア数を変更する判断ロジック（図４）を複数選択することが可能であるため、多種多様なタスクに対応することが可能となる。

[0114] タスクスケジューラ装置１００，１００Ａ（図１、図１０）において、所定頻度以上で使用するプロセッサのコアまたはコア群については、プロセッサの動作状態をより深い状態に遷移できないように上限を設定する動作状態設定部（C-state設定部１３２）を備える。

[0115] このように、頻繁に使用するＣＰＵコアに対して、遷移可能なCPU idle stateの上限を設定することにより、深いC-stateへ遷移できないようにし、復帰遅延を軽減する。これにより、タスク割り当てから演算開始までの遅延時間を抑制し、リアルタイム性高く演算することを可能とする。

[0116] 特に、使用頻度の高いＣＰＵコア群に対して、タスクを継続して割り当てられない場合の対応策として有効である。

[0117] また、タスクに対して優先度を付与することにより、優先したいタスクを、優先的に演算することが可能である。優先的に演算するタスクには、例えばQoS（Quality of Service）の保証がある。

[0118] タスクスケジューラ装置１００，１００Ａ（図１、図１０）において、動作状態設定部（C-state設定部１３２）は、タスク割当部（タスクＣＰＵ割当部１２３）が、タスクを割り当てなかったコアまたはコア群については、プロセッサの動作状態をより深い状態に遷移できるように設定する。

[0119] このようにすることにより、頻繁に利用するＣＰＵコア群（ActiveなＣＰＵコア群）へ継続的にタスクを割り当て、深いC-stateへ遷移することを防ぎつつ（C-stateからの復帰遅延軽減）、ActiveなＣＰＵコア以外は、深いC-stateに遷移できるので、省電力を図ることができる。

[0120] タスクスケジューラ装置１００，１００Ａ（図１、図１０）において、新

たに使用するコア、または、所定期間使用せずに再び使用するコアに対しては、タスクを割り当てる際に、プロセッサの動作状態を元の状態に復帰させる事前起床を行う事前起床部（ＣＰＵ事前起床部１３１）を備える。

[0121] 基本的な考え方としては、タスクを割当ててから、処理が終わり復帰するまでに時間がかかるのであれば、復帰するまでの時間分について事前に起床させ、ＣＰＵが使える状態までに事前に準備させておくものである。

このようにすることにより、深いstateに遷移したＣＰＵコアへタスクを割り当てる前に、事前起床をすることで、深いstateからの復帰時間を軽減することができる。新たに使用するコア、または、所定期間使用せずに再び使用するコアは、深いC-stateにあることが想定される。タスクを割り当てる前に、事前起床を行うことで、深いC-stateからの復帰時間を軽減することができる。例えば、使用するＣＰＵコア数を増やす際に、事前に該当ＣＰＵコアを起床させることにより、深いCPU idle stateに落ちていても、タスク割り当てから演算開始までの遅延時間を抑制し、リアルタイム性高く演算することを可能とする。

[0122] また、処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システム１０００（図１、図１０）であって、プロセッサ（ＣＰＵ）は、複数のコア群（ＣＰＵコア（CPUcore #0, CPUcore #1, …）３２）を有しており、プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置１００，１００Ａ（図１、図１０）を備え、タスクスケジューラ装置１００，１００Ａは、プロセッサの使用率を取得するプロセッサ使用率取得部（タスク量・周期 予測部１２２）と、プロセッサ使用率取得部が取得したプロセッサの使用率をもとに、所定頻度以上で使用するプロセッサのコアまたはコア群については、継続的にタスクを割り当てるタスク割当部（タスクＣＰＵ割当部１２３）と、を備える。

[0123] 計算システム１０００～１０００Ｄ（図１、図１０、図１５～図１８）は、パケット到着・処理のタスクに限らず、一般のタスク、例えばπの計算などにも拡張した技術である。

プロセッサの使用率に応じて、頻繁に利用するＣＰＵコア群（ActiveなＣＰＵコア群）へ継続的にタスクを割り当てることで、深いC-stateへ遷移することを防ぎ、C-stateからの復帰遅延を軽減することができる。一方、ActiveなＣＰＵコア以外は、深いC-stateに遷移できるので、省電力を図ることができる。すなわち、C-stateからの復帰遅延の軽減を図りつつ、事前起床は、なるべく深いＬＰＩ（C-state）をできるだけ長く保つことができるので、低遅延性と省電力化を両立させることができる。

[0124] 上記プロセッサは、ＣＰＵ以外にも、ＧＰＵ／ＦＰＧＡ／ＡＳＩＣ等のプロセッサに、idle stateの機能がある場合には、同様に適用可能である。

[0125] 計算システム１０００（図１、図１０）において、低遅延性を達成しつつ、省電力も同時に達成することができる。特に、頻繁に利用するＣＰＵコア群（ActiveなＣＰＵコア群）へ継続的にタスクを割り当てることで、ＣＰＵコアが深いidle stateに落ちることにより、復帰時に遅延時間が長大してしまう課題を、回避することができる。これにより、低遅延性を保証することができる。

[0126] なお、上記実施形態において説明した各処理のうち、自動的に行われるものとして説明した処理の全部又は一部を手動的に行うこともでき、あるいは、手動的に行われるものとして説明した処理の全部又は一部を公知の方法で自動的に行うこともできる。この他、上述文書中や図面中に示した処理手順、制御手順、具体的名称、各種のデータやパラメータを含む情報については、特記する場合を除いて任意に変更することができる。

また、図示した各装置の各構成要素は機能概念的なものであり、必ずしも物理的に図示の如く構成されていることを要しない。すなわち、各装置の分散・統合の具体的形態は図示のものに限られず、その全部又は一部を、各種の負荷や使用状況などに応じて、任意の単位で機能的又は物理的に分散・統合して構成することができる。

[0127] また、上記の各構成、機能、処理部、処理手段等は、それらの一部又は全部を、例えば集積回路で設計する等によりハードウェアで実現してもよい。

また、上記の各構成、機能等は、プロセッサがそれぞれの機能を実現するプログラムを解釈し、実行するためのソフトウェアで実現してもよい。各機能を実現するプログラム、テーブル、ファイル等の情報は、メモリや、ハードディスク、ＳＳＤ（Solid State Drive）等の記録装置、または、ＩＣ（Integrated Circuit）カード、ＳＤ（Secure Digital）カード、光ディスク等の記録媒体に保持することができる。

符号の説明

[0128] ３１　ハードウェア（ＨＷ）

３２　ＣＰＵコア（プロセッサ）

１，１－１，１－２，１－３　アプリケーション（ＡＰＬ）

１００，１００Ａ　タスクスケジューラ装置

１１０　管理部

１１１　運用者向け設定部

１２０　タスク管理部

１２１　タスク優先度付与部

１２２　タスク量・周期 予測部（プロセッサ使用率取得部）

１２３　タスクＣＰＵ割当部（タスク割当部）　１２０　タスク管理部

１３０　ＣＰＵ idle state制御部

１３１　ＣＰＵ事前起床部（事前起床部）

１３２　C-state設定部（動作状態設定部）

１０００，１０００Ａ，１０００Ｂ，１０００Ｃ，１０００Ｄ　演算システム（計算システム）

CPUcore #0, CPUcore #1, …　ＣＰＵコア

請求の範囲

[請求項1] 処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システムにおいて、前記プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置であって、

前記プロセッサの使用率を取得するプロセッサ使用率取得部と、

前記プロセッサ使用率取得部が取得した前記プロセッサの使用率をもとに、所定頻度以上で使用する前記プロセッサのコアまたはコア群については、継続的にタスクを割り当てるタスク割当部と、を備える

ことを特徴とするタスクスケジューラ装置。

[請求項2] 所定頻度以上で使用する前記プロセッサのコアまたはコア群については、前記プロセッサの動作状態をより深い状態に遷移できないように上限を設定する動作状態設定部を備える

ことを特徴とする請求項１に記載のタスクスケジューラ装置。

[請求項3] 前記動作状態設定部は、

タスク割当部が、タスクを割り当てなかったコアまたはコア群については、前記プロセッサの動作状態をより深い状態に遷移できるように設定する

ことを特徴とする請求項２に記載のタスクスケジューラ装置。

[請求項4] 新たに使用するコア、または、所定期間使用せずに再び使用するコアに対しては、タスクを割り当てる際に、前記プロセッサの動作状態を元の状態に復帰させる事前起床を行う事前起床部を備える

ことを特徴とする請求項１に記載のタスクスケジューラ装置。

[請求項5] 処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システムであって、

前記プロセッサは、複数のコア群を有しており、

前記プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置を備え、

前記タスクスケジューラ装置は、

前記プロセッサの使用率を取得するプロセッサ使用率取得部と、

前記プロセッサ使用率取得部が取得した前記プロセッサの使用率をもとに、所定頻度以上で使用する前記プロセッサのコアまたはコア群については、継続的にタスクを割り当てるタスク割当部と、を備える

ことを特徴とする計算システム。

[請求項6] 処理負荷に応じてプロセッサの動作状態を段階的に減らして消費電力量を削減する計算システムにおいて、前記プロセッサのコア群へタスクを割り当てるタスクスケジューラ装置のタスクスケジューリング方法であって、

前記プロセッサの使用率を取得するステップと、

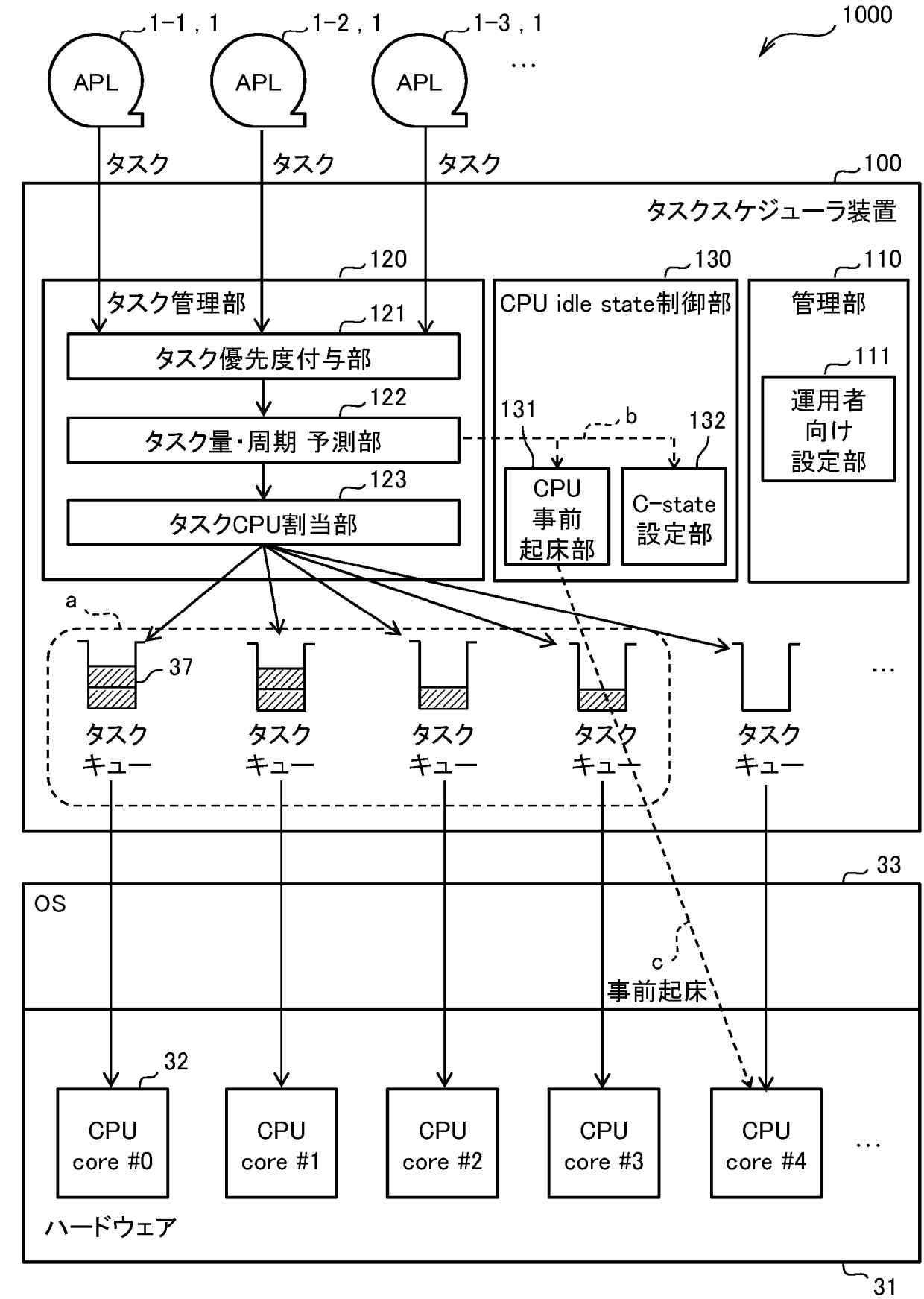
取得した前記プロセッサの使用率をもとに、所定頻度以上で使用する前記プロセッサのコアまたはコア群については、継続的にタスクを割り当てるステップと、を実行する

ことを特徴とするタスクスケジューリング方法。

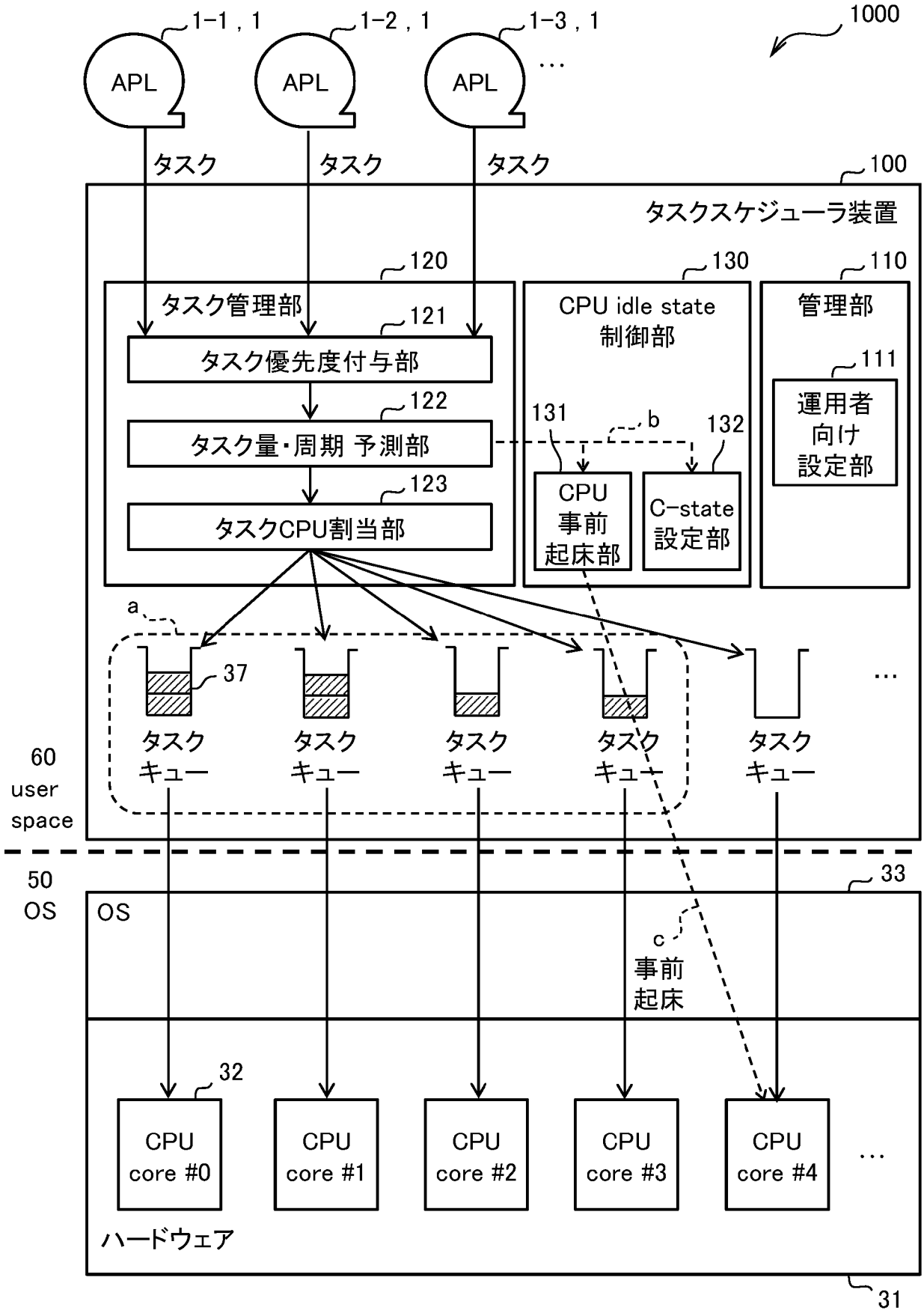
[請求項7] コンピュータを、請求項１乃至４のいずれか一項に記載のタスクスケジューラ装置として機能させるためのプログラム。

WO 2024/013831 PCT/JP2022/027327

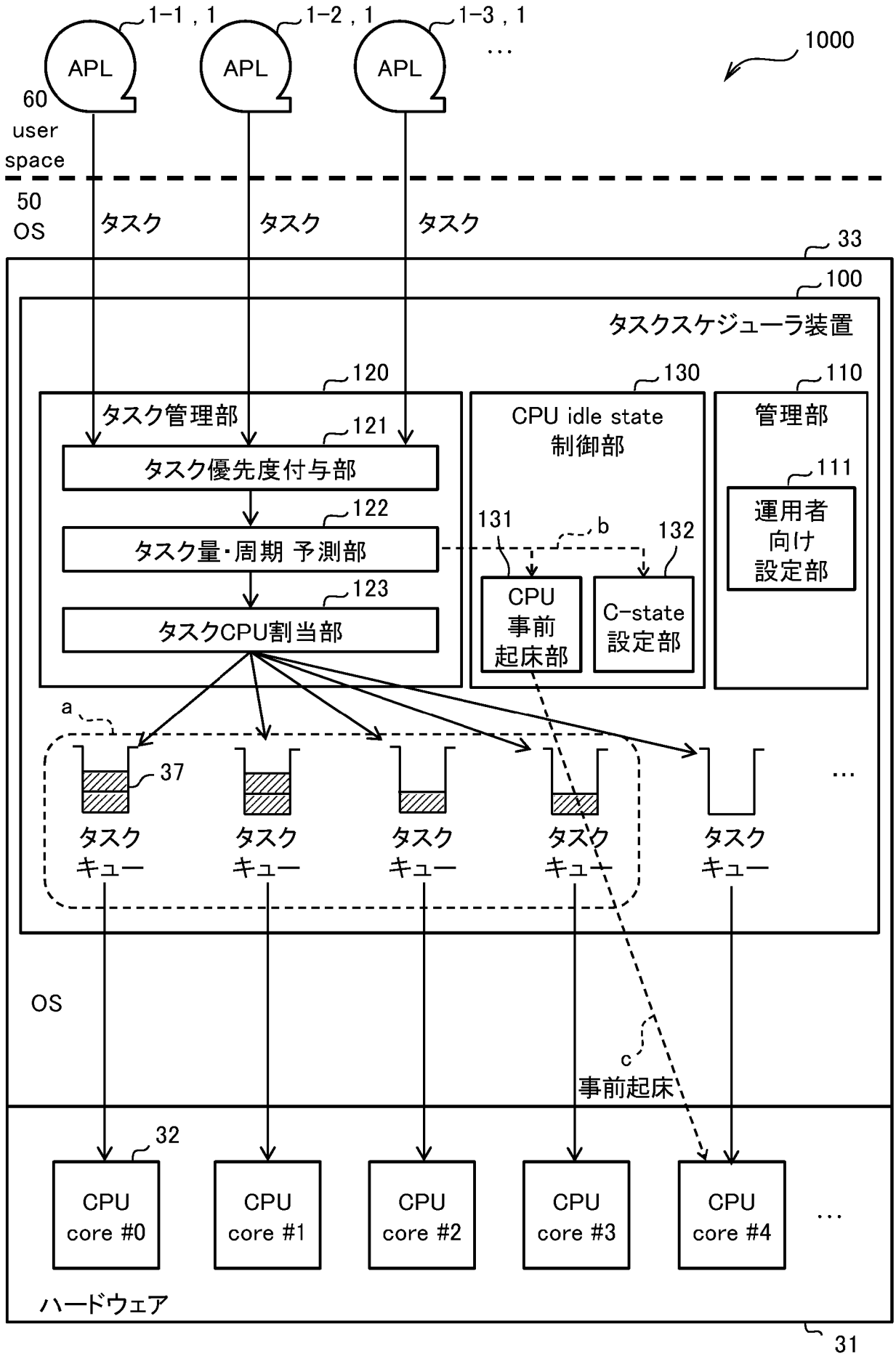
[図1]



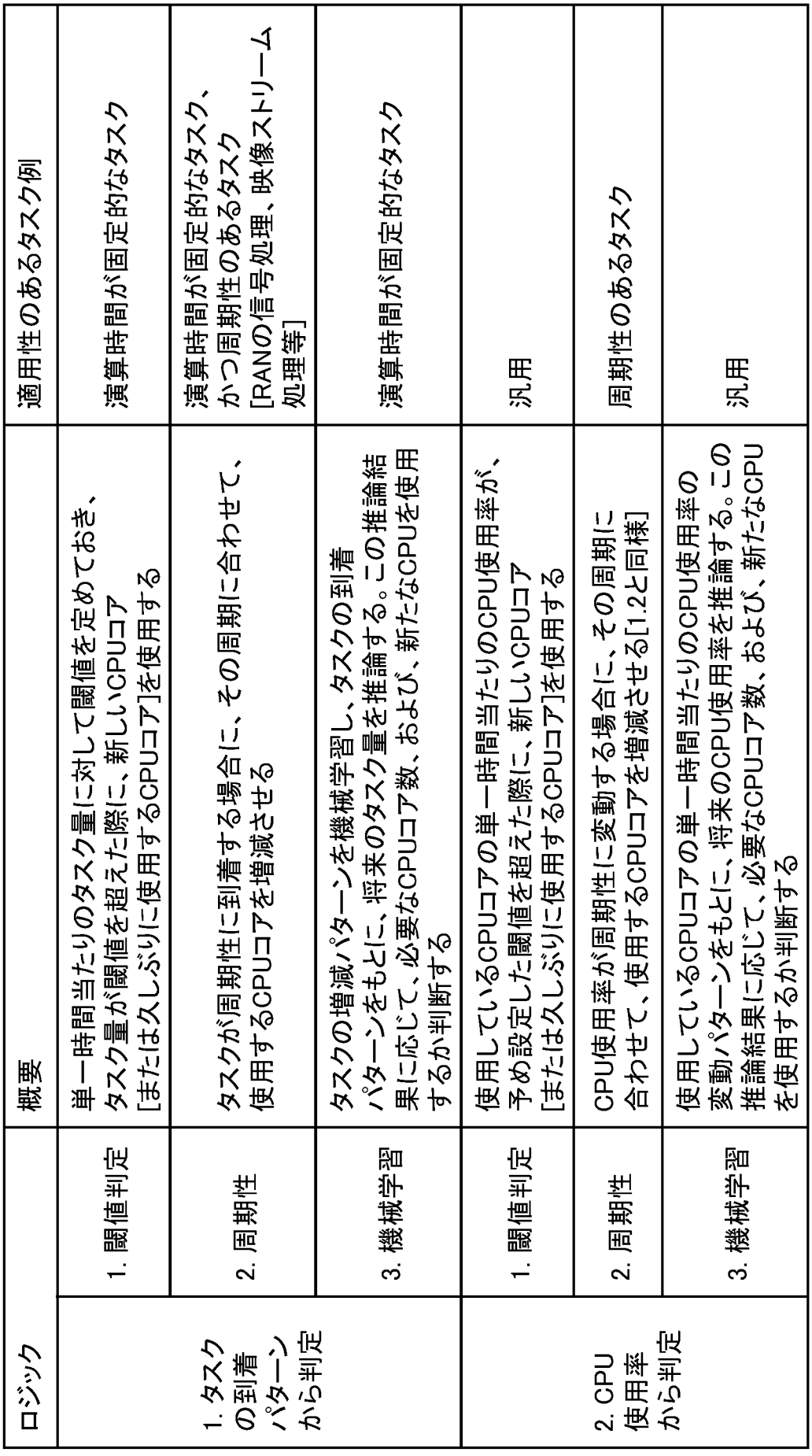
[図2]


1000
1-1，1
1-2，1
1-3，1
APL
APL
APL
タスク
タスク
タスク
100
タスクスケジューラ装置
120
タスク管理部
121
タスク優先度付与部
122
タスク量・周期 予測部
123
タスクCPU割当部
130
CPU idle state
制御部
131
b
132
CPU
事前
起床部
C-state
設定部
110
管理部
111
運用者
向け
設定部
a
37
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
60
user
space
50
OS
33
OS
c
事前
起床
32
CPU
core #0
CPU
core #1
CPU
core #2
CPU
core #3
CPU
core #4
ハードウェア
31

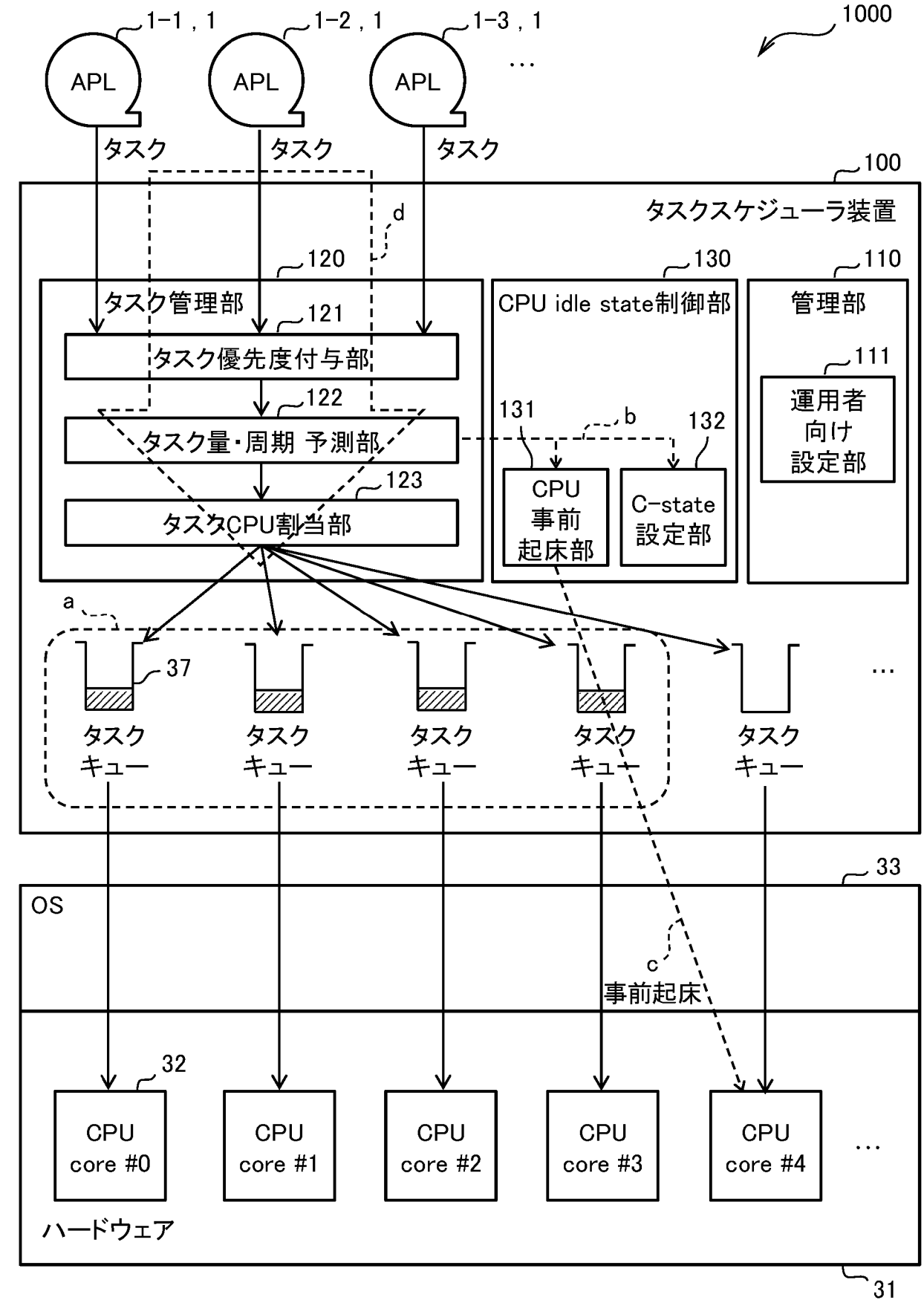
[図3]


1-1，1
1-2，1
1-3，1
APL
APL
APL
1000
60
user space
50
OS
タスク
タスク
タスク
33
100
タスクスケジューラ装置
120
タスク管理部
121
タスク優先度付与部
122
タスク量・周期 予測部
123
タスクCPU割当部
130
CPU idle state 制御部
131
b
132
CPU 事前 起床部
C-state 設定部
110
管理部
111
運用者 向け 設定部
a
37
タスク キュー
タスク キュー
タスク キュー
タスク キュー
タスク キュー
OS
c
事前起床
32
CPU core #0
CPU core #1
CPU core #2
CPU core #3
CPU core #4
ハードウェア
31

[図4]


ロジック
概要
適用性のあるタスク例
1. タスクの到着パターンから判定
1. 閾値判定
単一時間当たりのタスク量に対して閾値を定めておき、タスク量が閾値を超えた際に、新しいCPUコア[またはは久しぶりに使用するCPUコア]を使用する
演算時間が固定的なタスク
2. 周期性
タスクが周期性に到着する場合に、その周期に合わせて、使用するCPUコアを増減させる
演算時間が固定的なタスク、かつ周期性のあるタスク[RANの信号処理、映像ストリーム処理等]
3. 機械学習
タスクの増減パターンを機械学習し、タスクの到着パターンをもとに、将来のタスク量を推論する。この推論結果に応じて、必要なCPUコア数、および、新たなCPUを使用するか判断する
演算時間が固定的なタスク
2. CPU使用率から判定
1. 閾値判定
使用しているCPUコアの単一時間当たりのCPU使用率が、予め設定した閾値を超えた際に、新しいCPUコア[またはは久しぶりに使用するCPUコア]を使用する
汎用
2. 周期性
CPU使用率が周期性に変動する場合に、その周期に合わせて、使用するCPUコアを増減させる[1.2と同様]
周期性のあるタスク
3. 機械学習
使用しているCPUコアの単一時間当たりのCPU使用率の変動パターンをもとに、将来のCPU使用率を推論する。この推論結果に応じて、必要なCPUコア数、および、新たなCPUを使用するか判断する
汎用

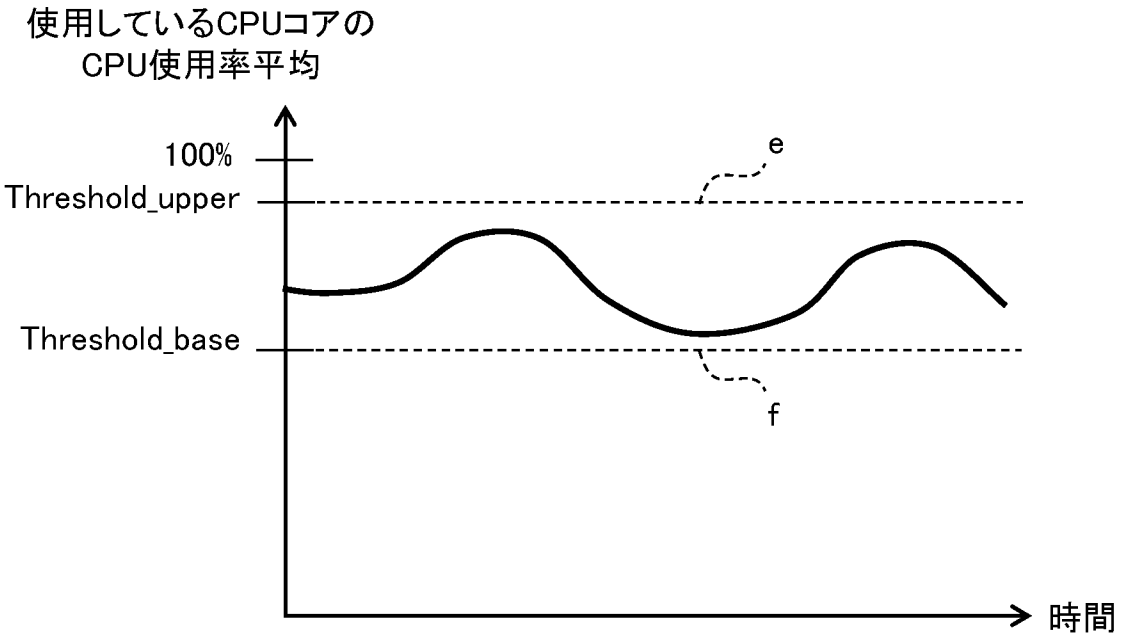
[図5]


1000
1-1，1
1-2，1
1-3，1
APL
APL
APL
タスク
タスク
タスク
100
タスクスケジューラ装置
d
120
タスク管理部
121
タスク優先度付与部
122
タスク量・周期 予測部
123
タスクCPU割当部
130
CPU idle state制御部
131
b
132
CPU
事前
起床部
C-state
設定部
110
管理部
111
運用者
向け
設定部
a
37
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
33
OS
c
事前起床
32
CPU
core #0
CPU
core #1
CPU
core #2
CPU
core #3
CPU
core #4
ハードウェア
31

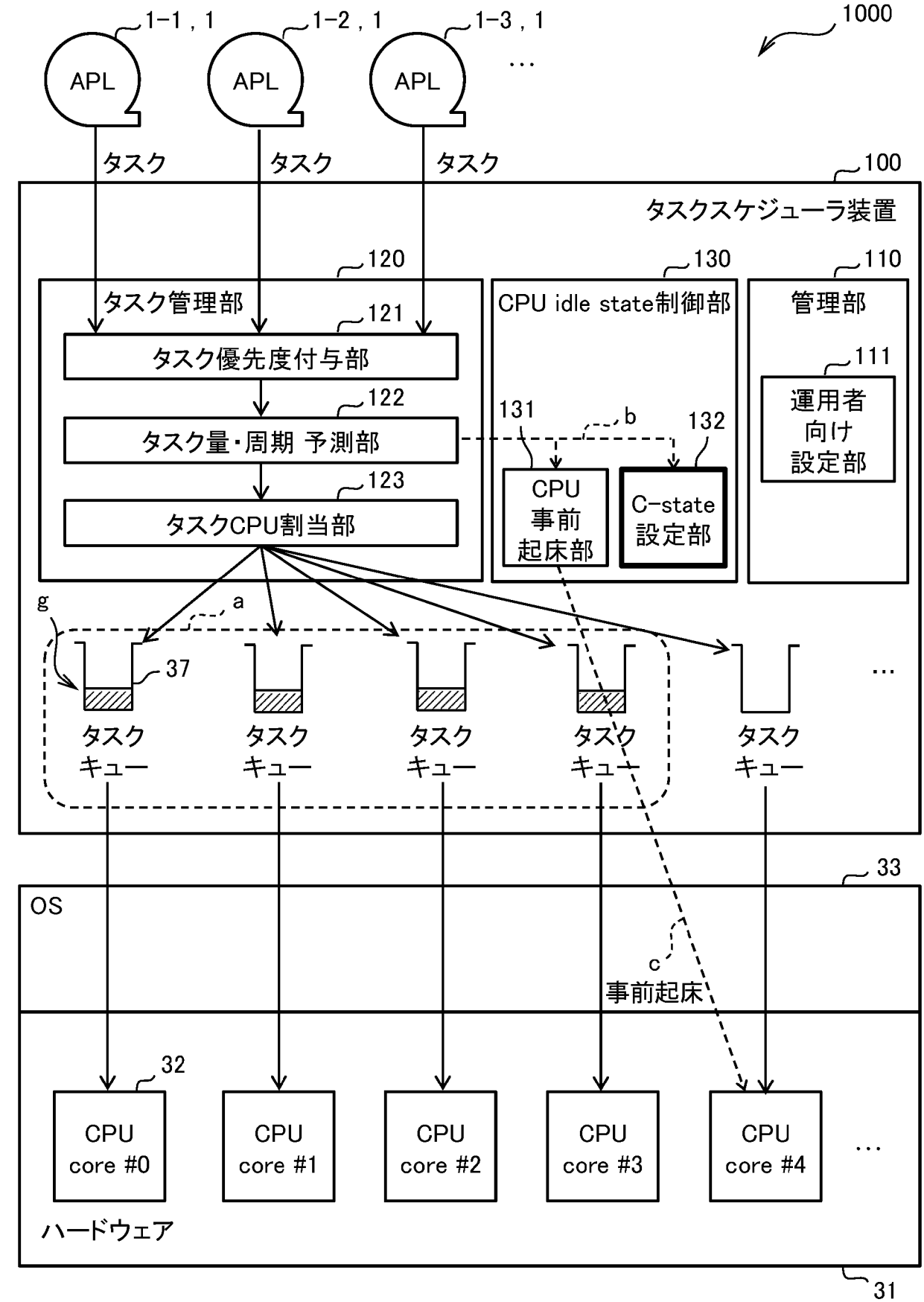
[図6]

W_input	使用するCPUコア数
aa～bb	1
bb～cc	2
cc～dd	3
dd～ee	4
ee～ff	5
…	…

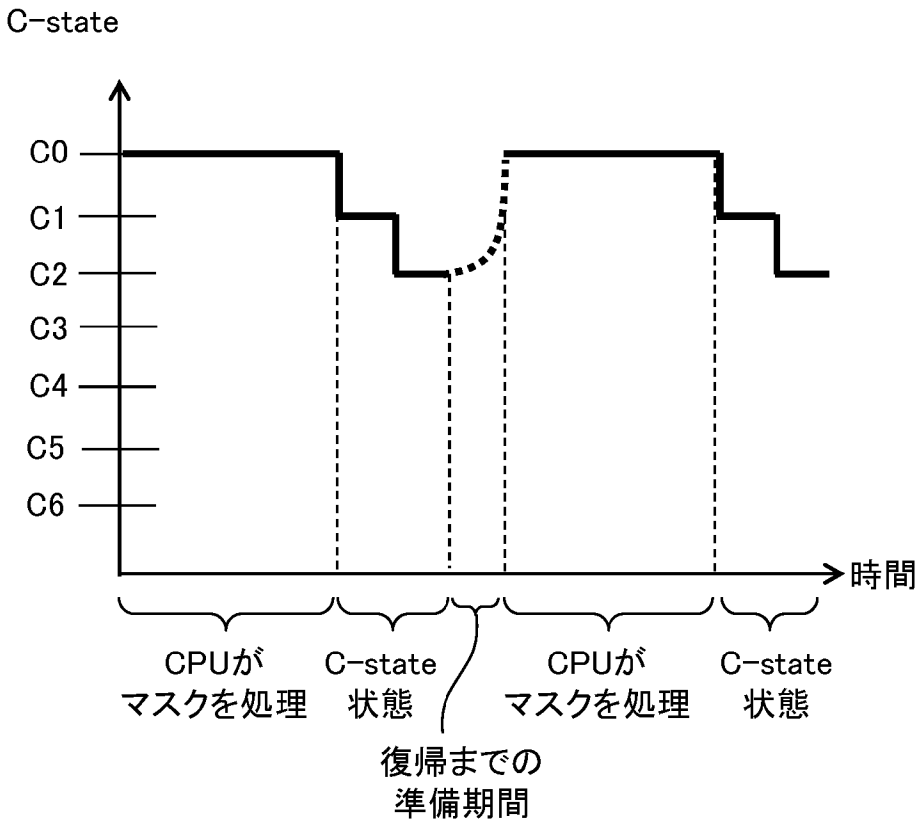
[図7]

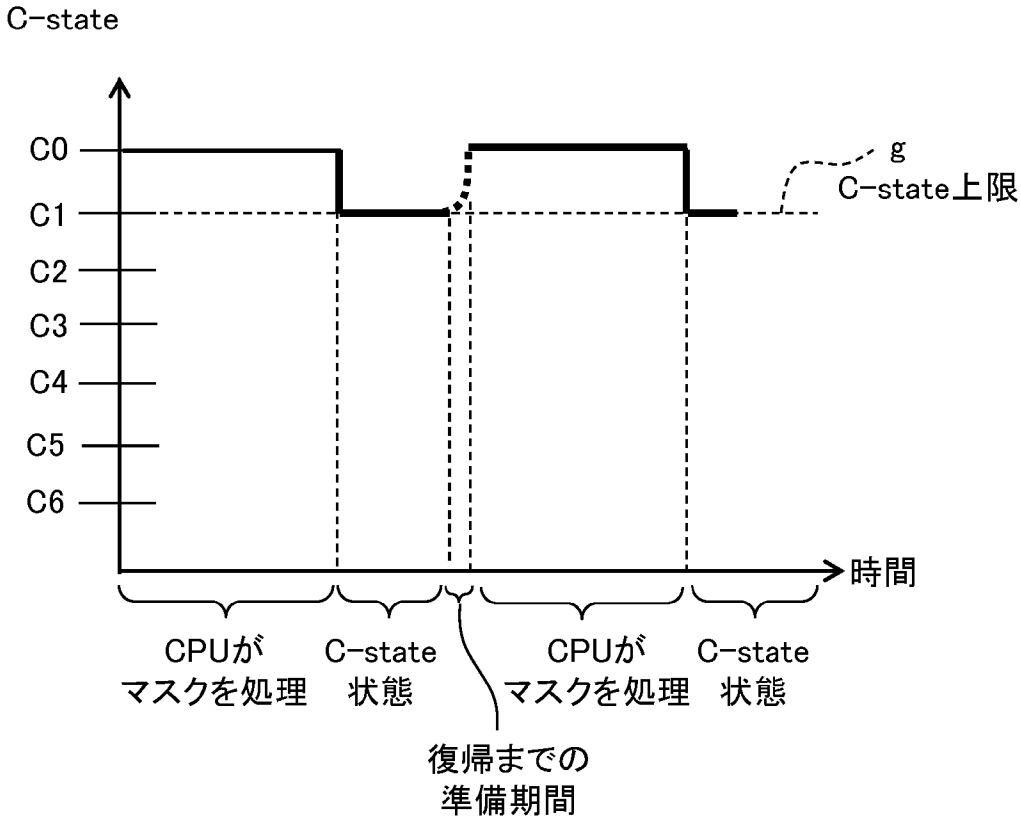


[図8]

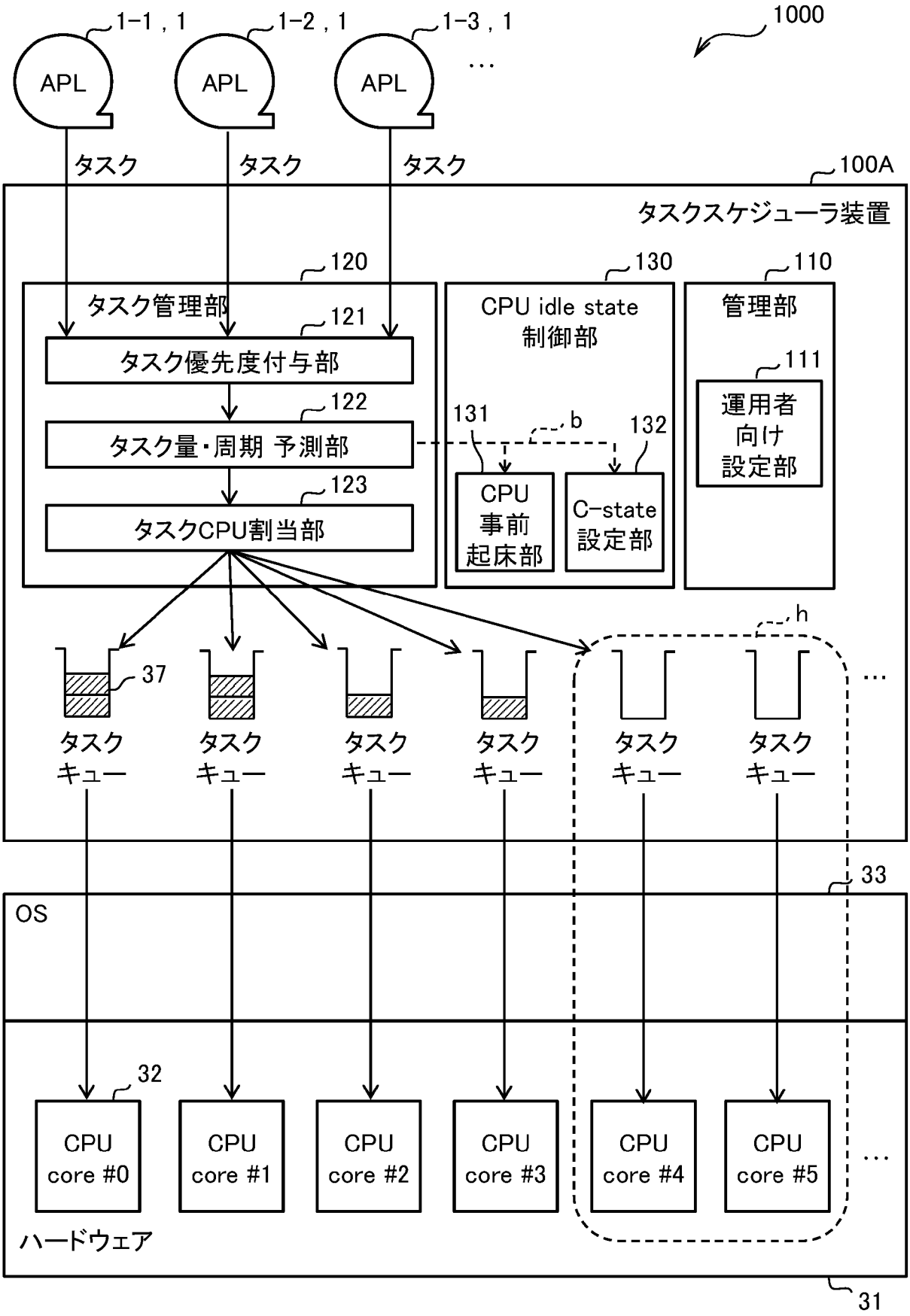

1-1，1
1-2，1
1-3，1
1000
APL
APL
APL
タスク
タスク
タスク
100
タスクスケジューラ装置
120
タスク管理部
121
タスク優先度付与部
122
タスク量・周期 予測部
123
タスクCPU割当部
130
CPU idle state制御部
131
b
132
CPU
事前
起床部
C-state
設定部
110
管理部
111
運用者
向け
設定部
g
a
37
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
33
OS
c
事前起床
32
CPU
core #0
CPU
core #1
CPU
core #2
CPU
core #3
CPU
core #4
ハードウェア
31

[図9]

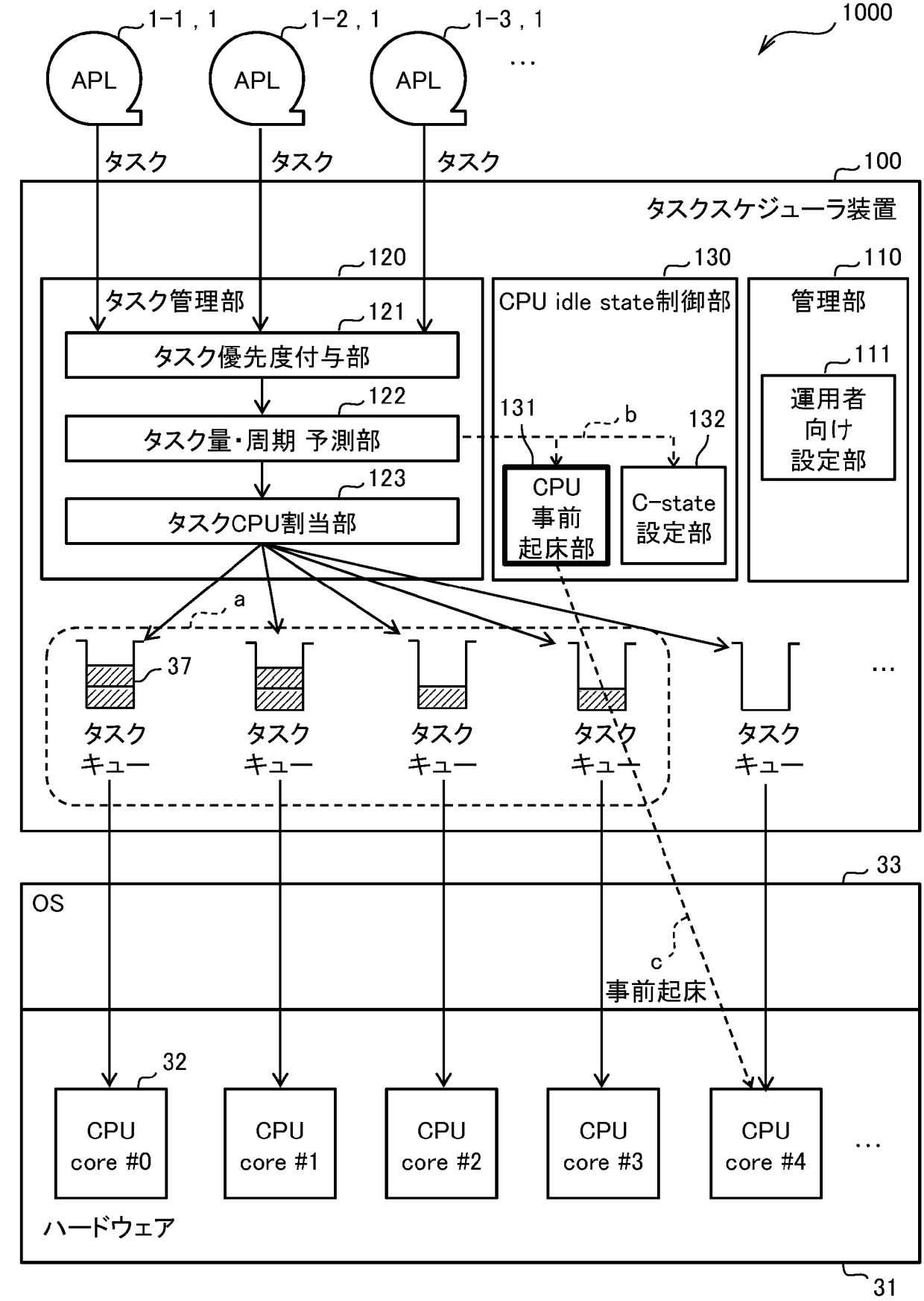

C-state
C0
C1
C2
C3
C4
C5
C6
時間
CPUが
マスクを処理
C-state
状態
CPUが
マスクを処理
C-state
状態
復帰までの
準備期間


C-state
C0
C1
C2
C3
C4
C5
C6
g
C-state上限
時間
CPUが
マスクを処理
C-state
状態
CPUが
マスクを処理
C-state
状態
復帰までの
準備期間

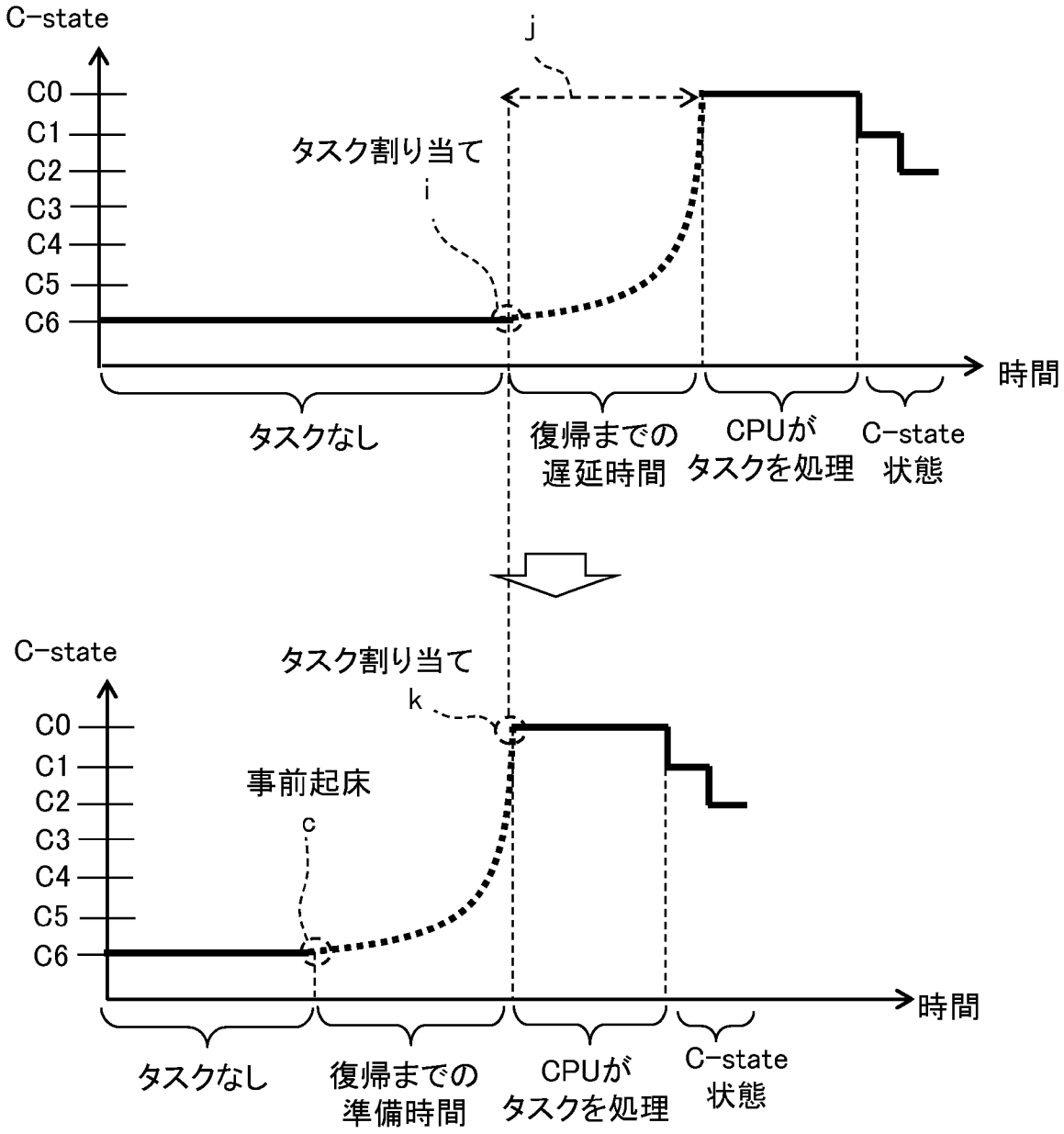
[図10]


1000
1-1，1
1-2，1
1-3，1
APL
APL
APL
タスク
タスク
タスク
100A
タスクスケジューラ装置
120
タスク管理部
121
タスク優先度付与部
122
タスク量・周期 予測部
123
タスクCPU割当部
130
CPU idle state
制御部
131
b
132
CPU
事前
起床部
C-state
設定部
110
管理部
111
運用者
向け
設定部
h
37
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
33
OS
32
CPU
core #0
CPU
core #1
CPU
core #2
CPU
core #3
CPU
core #4
CPU
core #5
ハードウェア
31

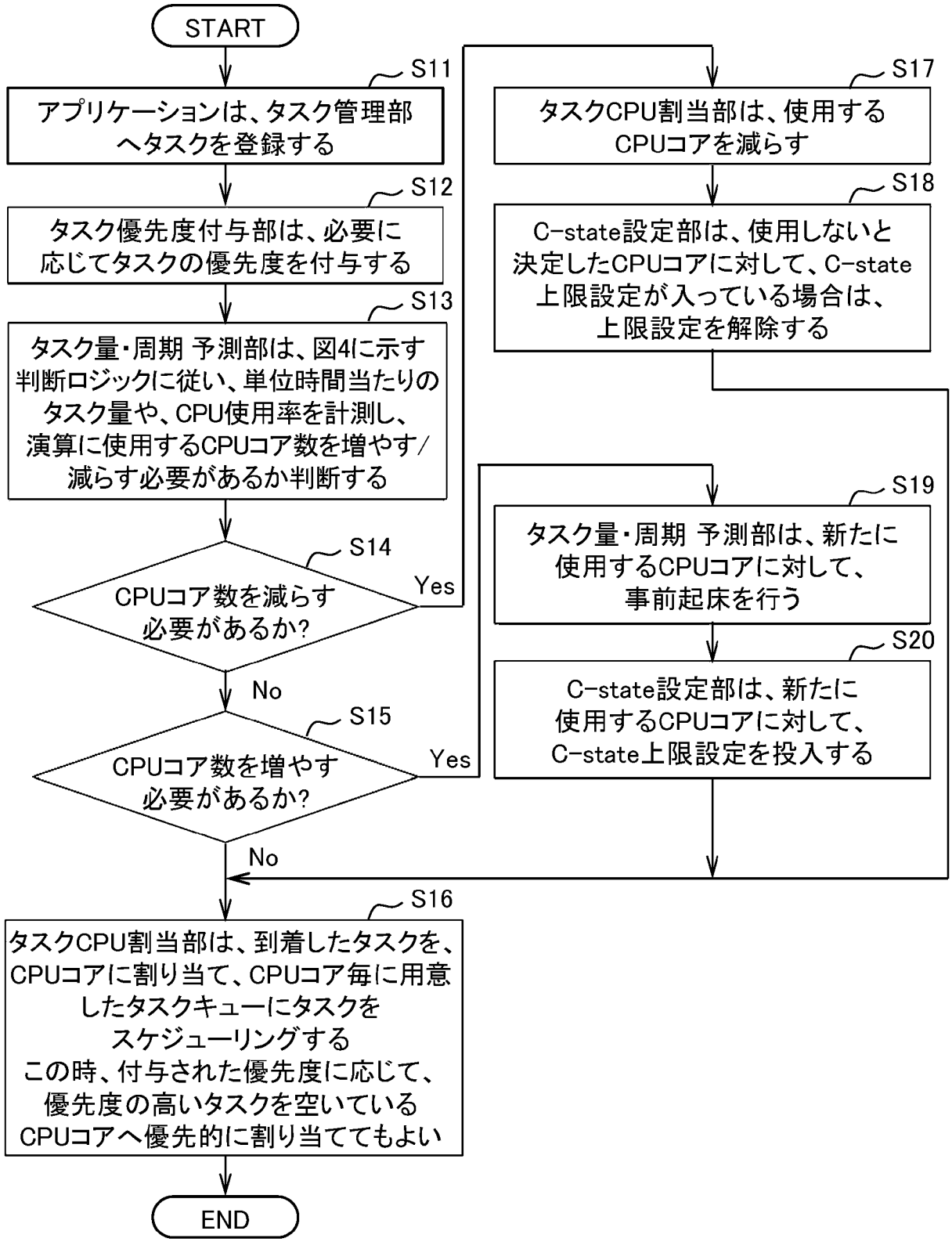
[図11]


1-1，1
1-2，1
1-3，1
APL
APL
APL
…
1000
タスク
タスク
タスク
100
タスクスケジューラ装置
120
タスク管理部
121
タスク優先度付与部
122
タスク量・周期 予測部
123
タスクCPU割当部
130
CPU idle state制御部
131
b
132
CPU
事前
起床部
C-state
設定部
110
管理部
111
運用者
向け
設定部
a
37
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
…
33
OS
c
事前起床
32
CPU
core #0
CPU
core #1
CPU
core #2
CPU
core #3
CPU
core #4
…
ハードウェア
31

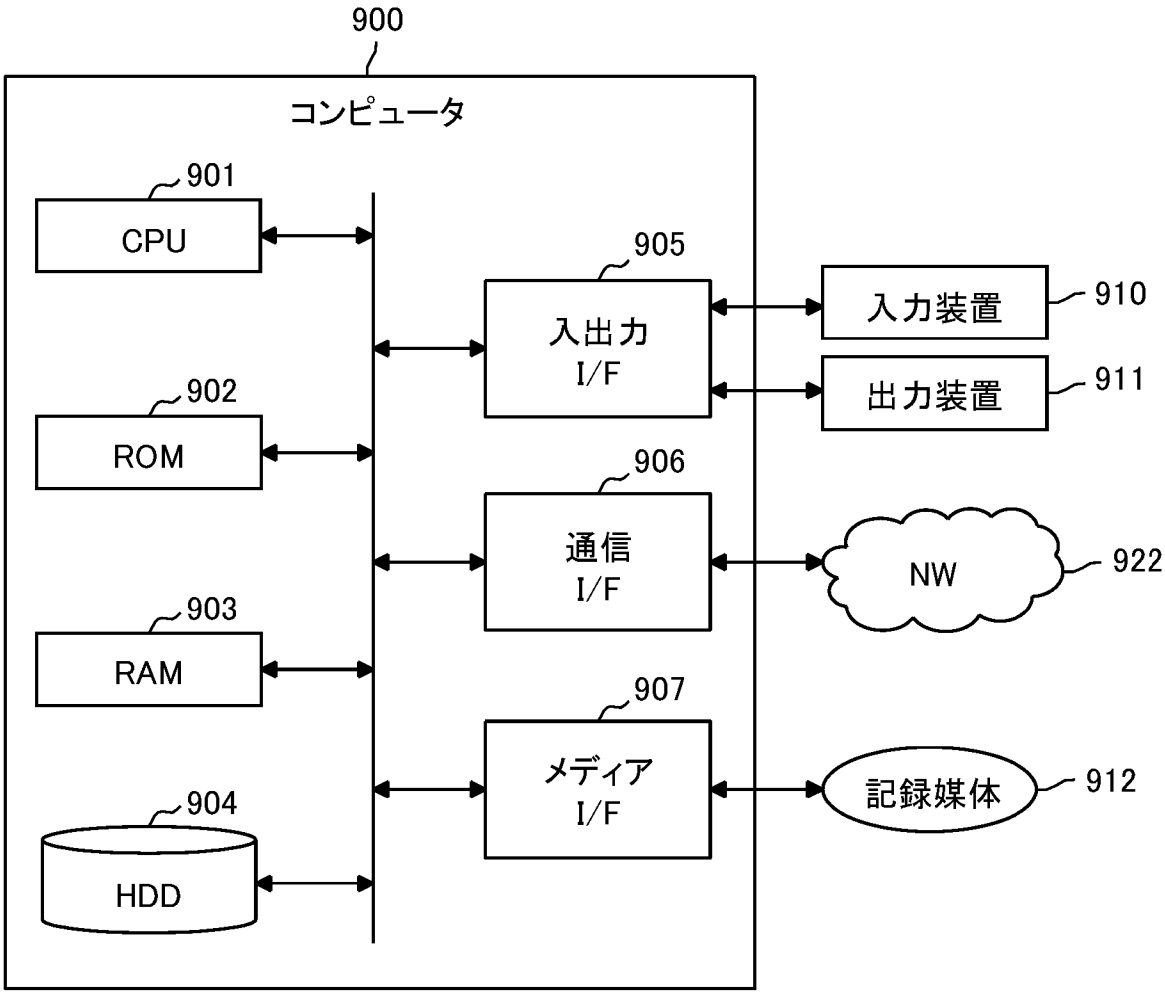
[図12]


C-state
C0
C1
C2
C3
C4
C5
C6
j
タスク割り当て
i
時間
タスクなし
復帰までの
遅延時間
CPUが
タスクを処理
C-state
状態
C-state
タスク割り当て
k
C0
C1
C2
C3
C4
C5
C6
事前起床
c
時間
タスクなし
復帰までの
準備時間
CPUが
タスクを処理
C-state
状態

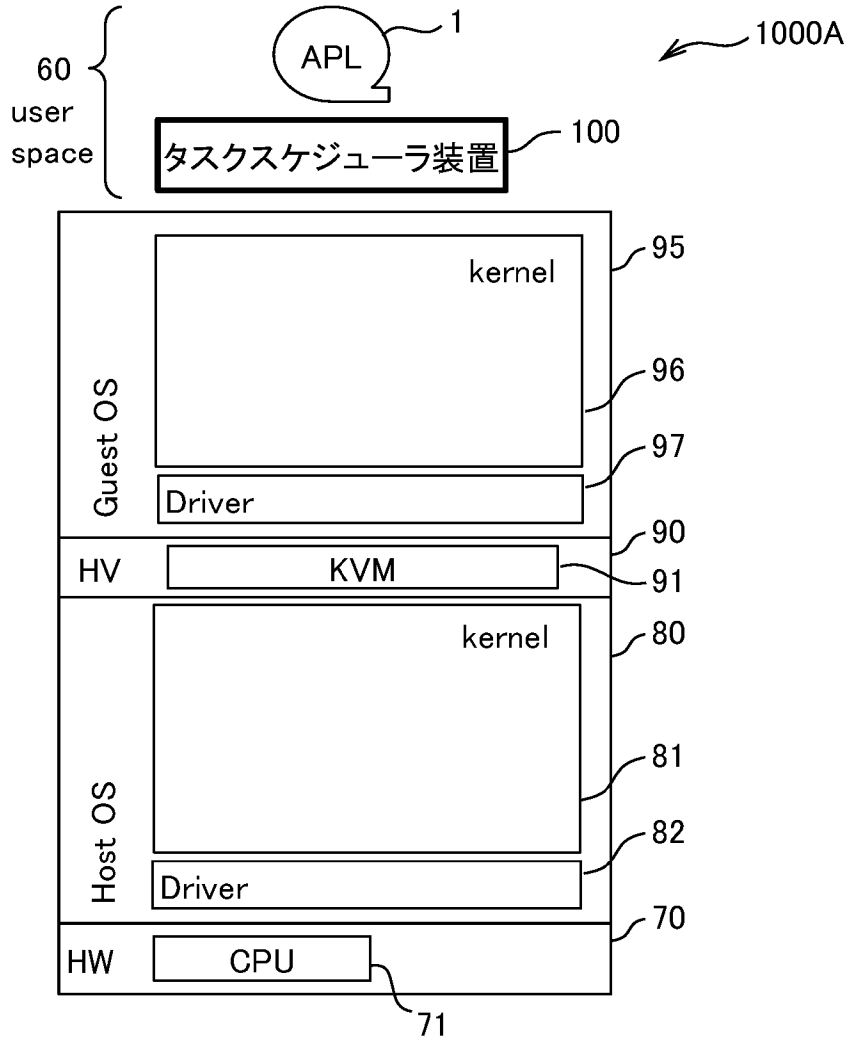
[図13]


START
S11
アプリケーションは、タスク管理部へタスクを登録する
S12
タスク優先度付与部は、必要に応じてタスクの優先度を付与する
S13
タスク量・周期 予測部は、図4に示す判断ロジックに従い、単位時間当たりのタスク量や、CPU使用率を計測し、演算に使用するCPUコア数を増やす/減らす必要があるか判断する
S14
CPUコア数を減らす必要があるか?
Yes
No
S15
CPUコア数を増やす必要があるか?
Yes
No
S16
タスクCPU割当部は、到着したタスクを、CPUコアに割り当て、CPUコア毎に用意したタスクキューにタスクをスケジューリングする
この時、付与された優先度に応じて、優先度の高いタスクを空いているCPUコアへ優先的に割り当ててもよい
END
S17
タスクCPU割当部は、使用するCPUコアを減らす
S18
C-state設定部は、使用しないと決定したCPUコアに対して、C-state上限設定が入っている場合は、上限設定を解除する
S19
タスク量・周期 予測部は、新たに使用するCPUコアに対して、事前起床を行う
S20
C-state設定部は、新たに使用するCPUコアに対して、C-state上限設定を投入する

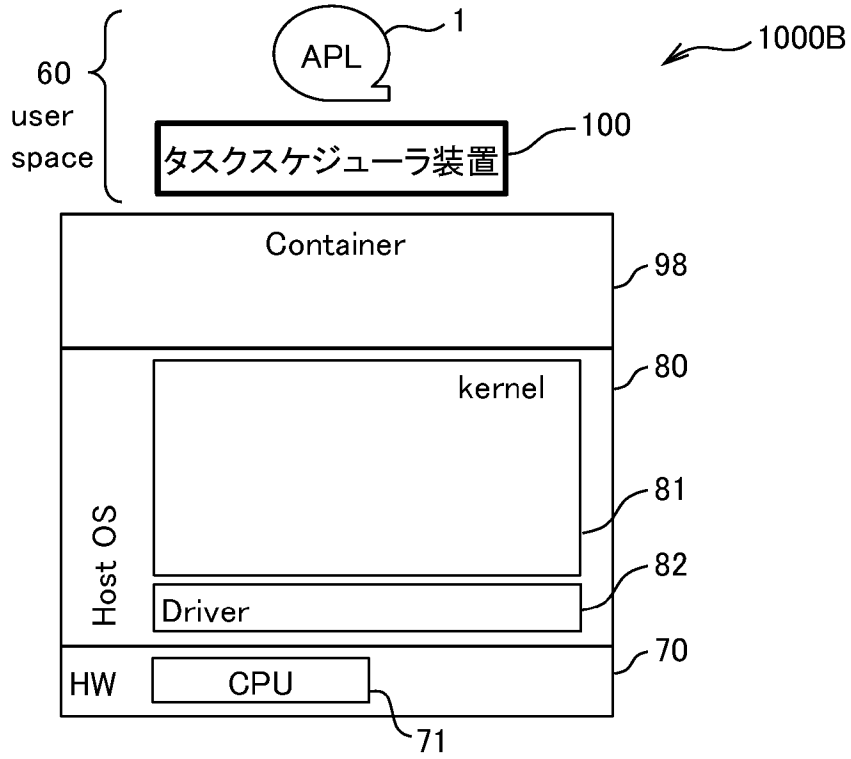
[図14]


900
コンピュータ
901
CPU
902
ROM
903
RAM
904
HDD
905
入出力
I/F
906
通信
I/F
907
メディア
I/F
910
入力装置
911
出力装置
922
NW
912
記録媒体

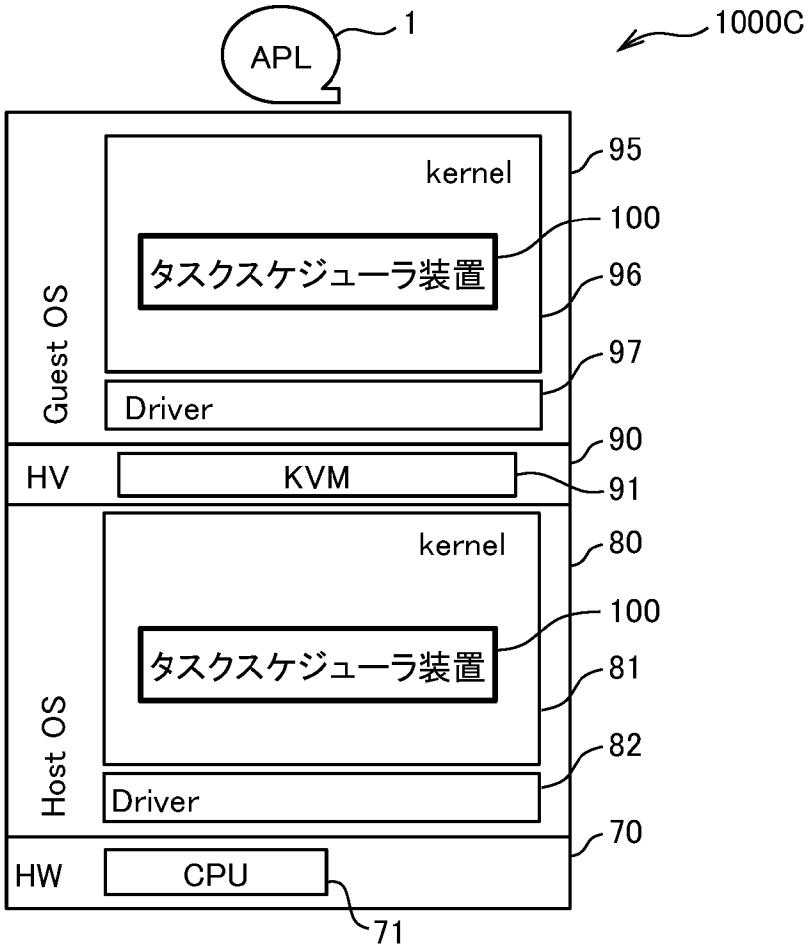
[図15]


60
user
space
APL
1
1000A
タスクスケジューラ装置
100
Guest OS
kernel
95
96
Driver
97
HV
KVM
90
91
Host OS
kernel
80
81
Driver
82
HW
CPU
70
71

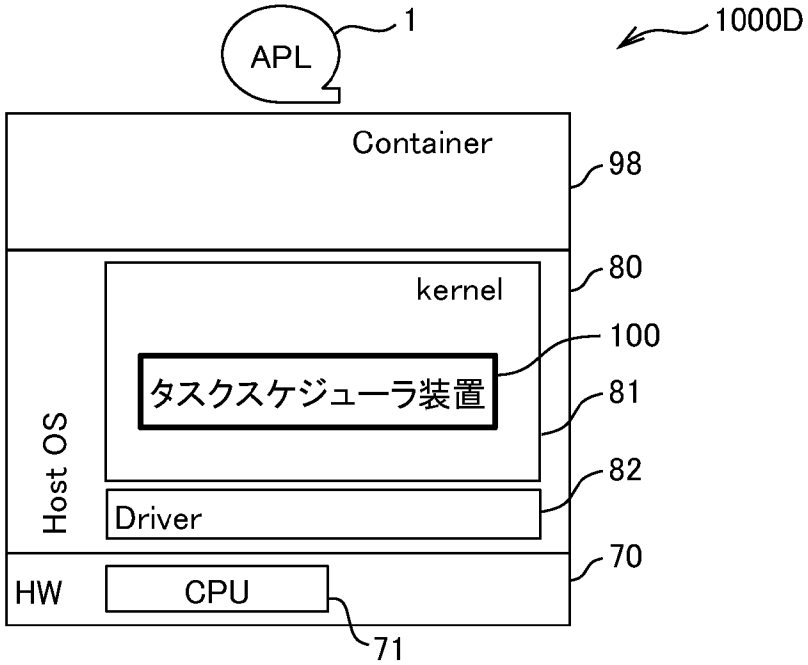
[図16]


60
user
space
APL
1
1000B
タスクスケジューラ装置
100
Container
98
Host OS
kernel
80
81
Driver
82
HW
CPU
70
71

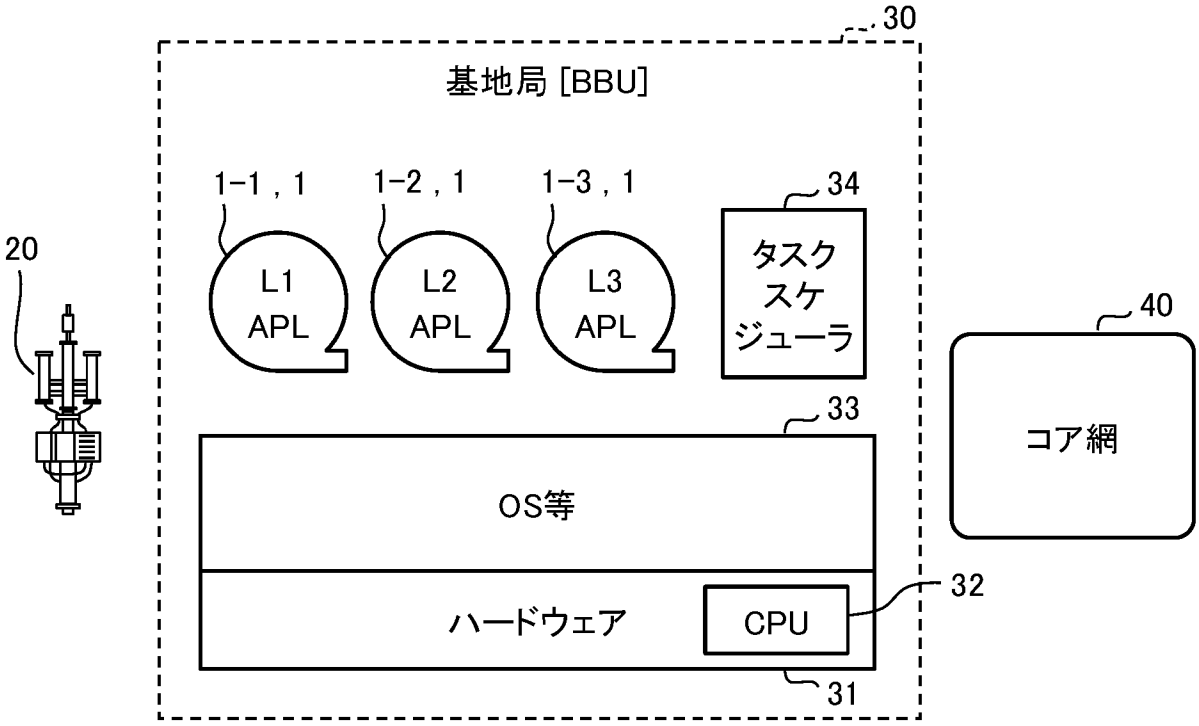
[図17]


APL
1
1000C
95
kernel
100
タスクスケジューラ装置
96
Guest OS
97
Driver
90
HV
KVM
91
80
kernel
100
タスクスケジューラ装置
81
Host OS
82
Driver
70
HW
CPU
71

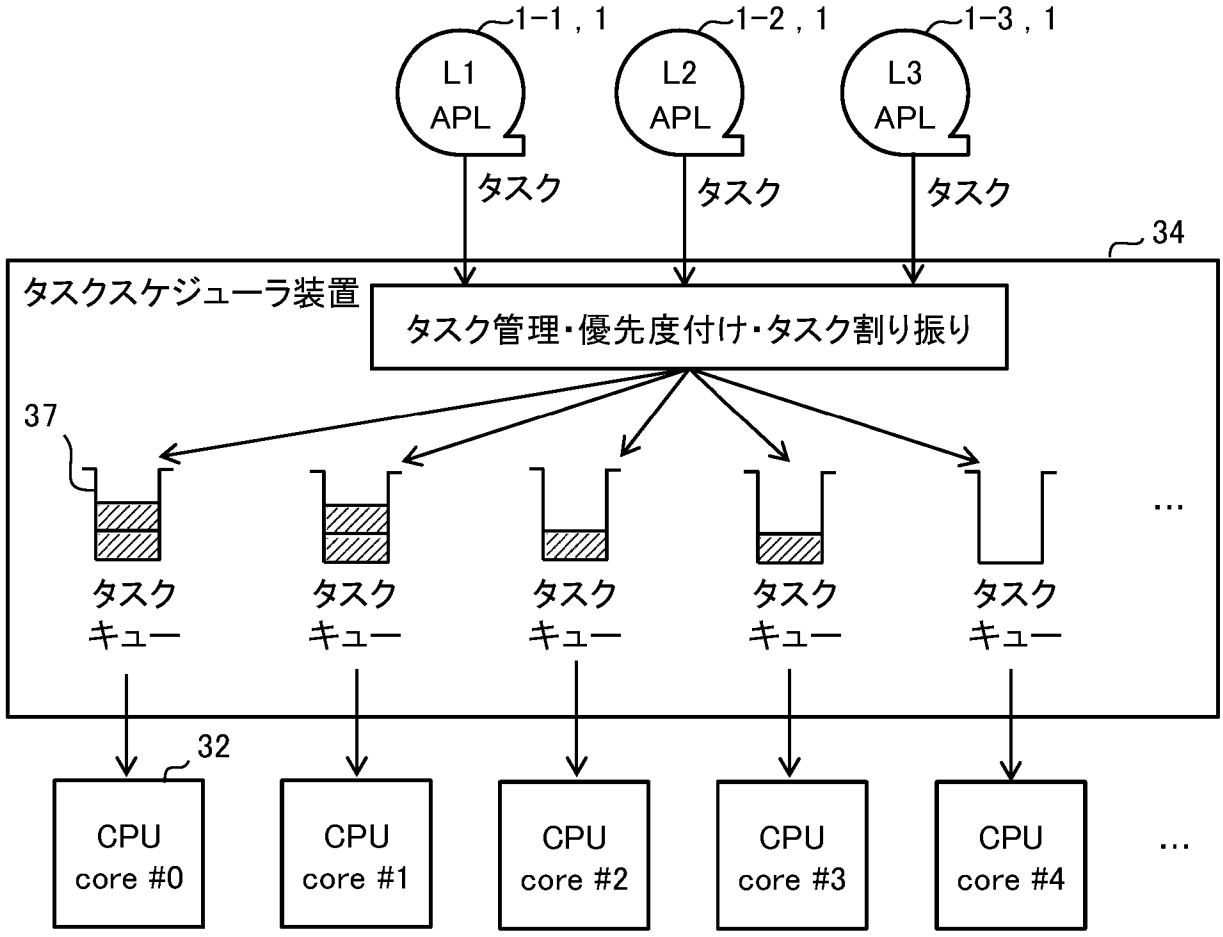
[図18]


APL
1
1000D
Container
98
80
kernel
100
タスクスケジューラ装置
81
Host OS
82
Driver
70
HW
CPU
71

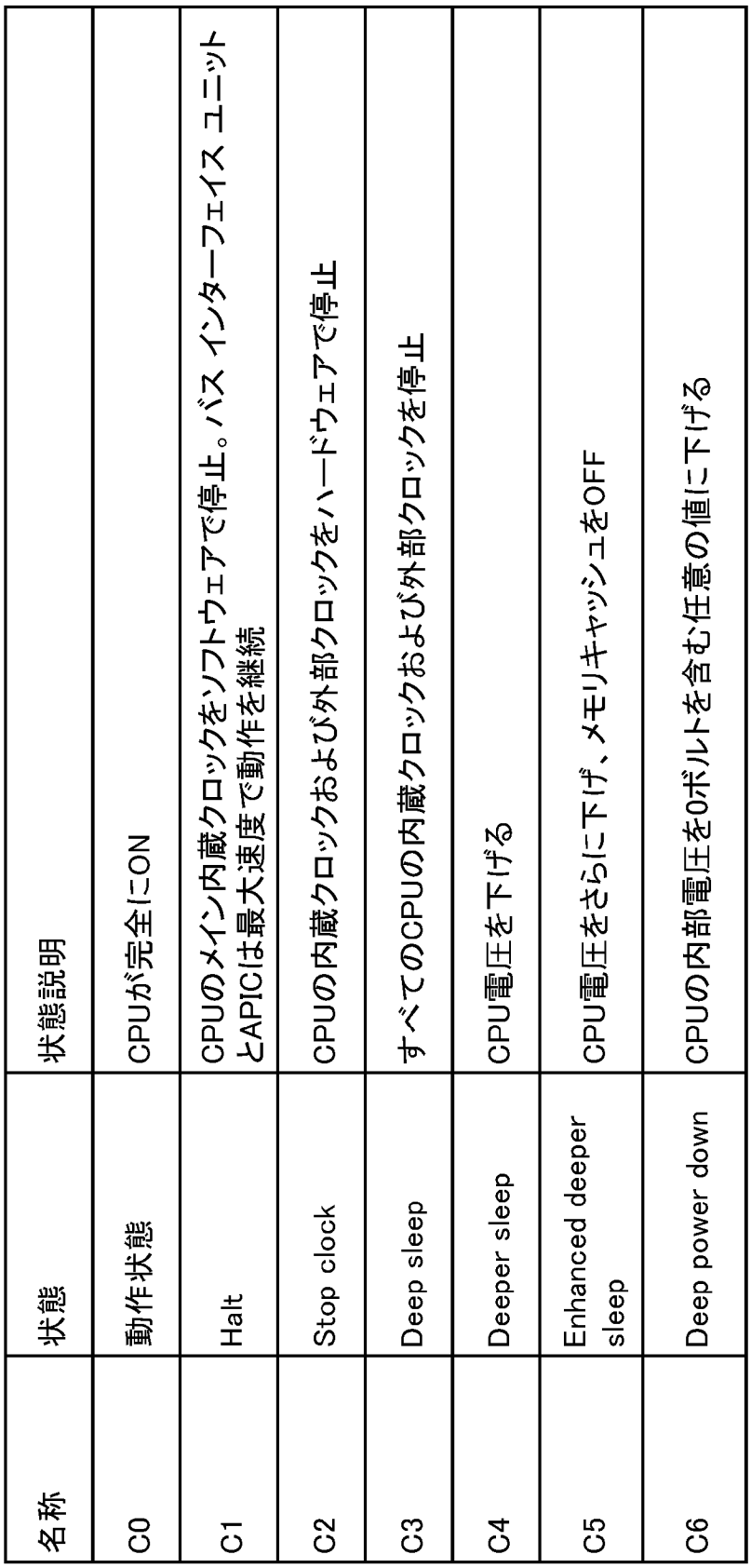
[図19]


30
基地局 [BBU]
1-1，1
1-2，1
1-3，1
34
20
L1
APL
L2
APL
L3
APL
タスク
スケ
ジューラ
40
コア網
33
OS等
ハードウェア
CPU
32
31

[図20]


1-1，1
1-2，1
1-3，1
L1
APL
L2
APL
L3
APL
タスク
タスク
タスク
34
タスクスケジューラ装置
タスク管理・優先度付け・タスク割り振り
37
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
タスク
キュー
…
32
CPU
core #0
CPU
core #1
CPU
core #2
CPU
core #3
CPU
core #4
…

[図21]

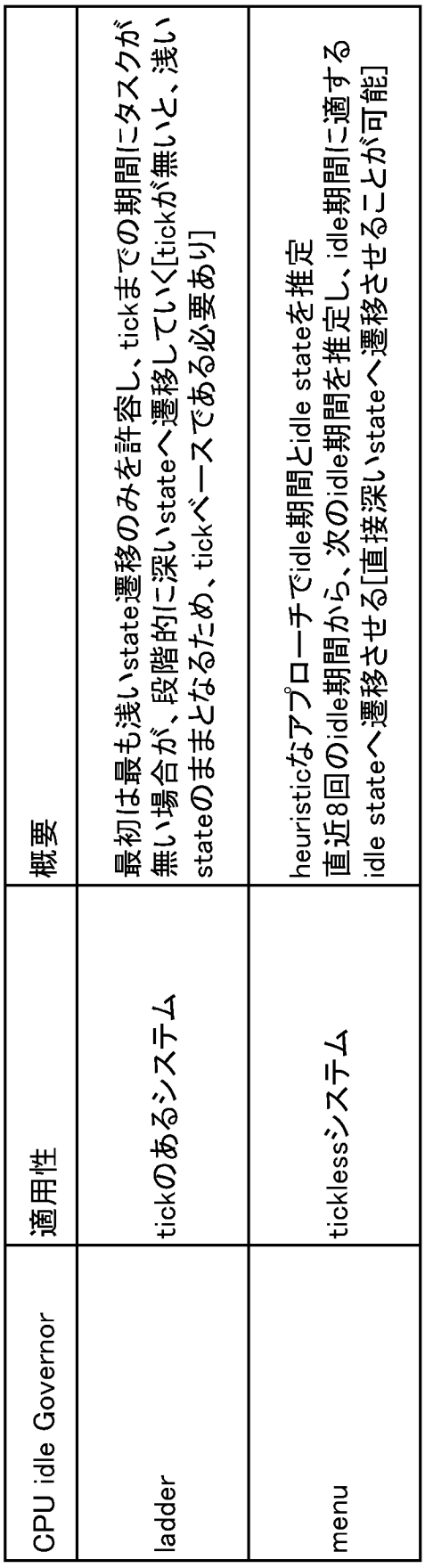


名称	状態	状態説明
C0	動作状態	CPUが完全にON
C1	Halt	CPUのメイン内蔵クロックをソフトウェアで停止。バス インターフェイス ユニットとAPICは最大速度で動作を継続
C2	Stop clock	CPUの内蔵クロックおよび外部クロックをハードウェアで停止
C3	Deep sleep	すべてのCPUの内蔵クロックおよび外部クロックを停止
C4	Deeper sleep	CPU電圧を下げる
C5	Enhanced deeper sleep	CPU電圧をさらに下げ、メモリキャッシュをOFF
C6	Deep power down	CPUの内部電圧を0ボルトを含む任意の値に下げる

[図22]

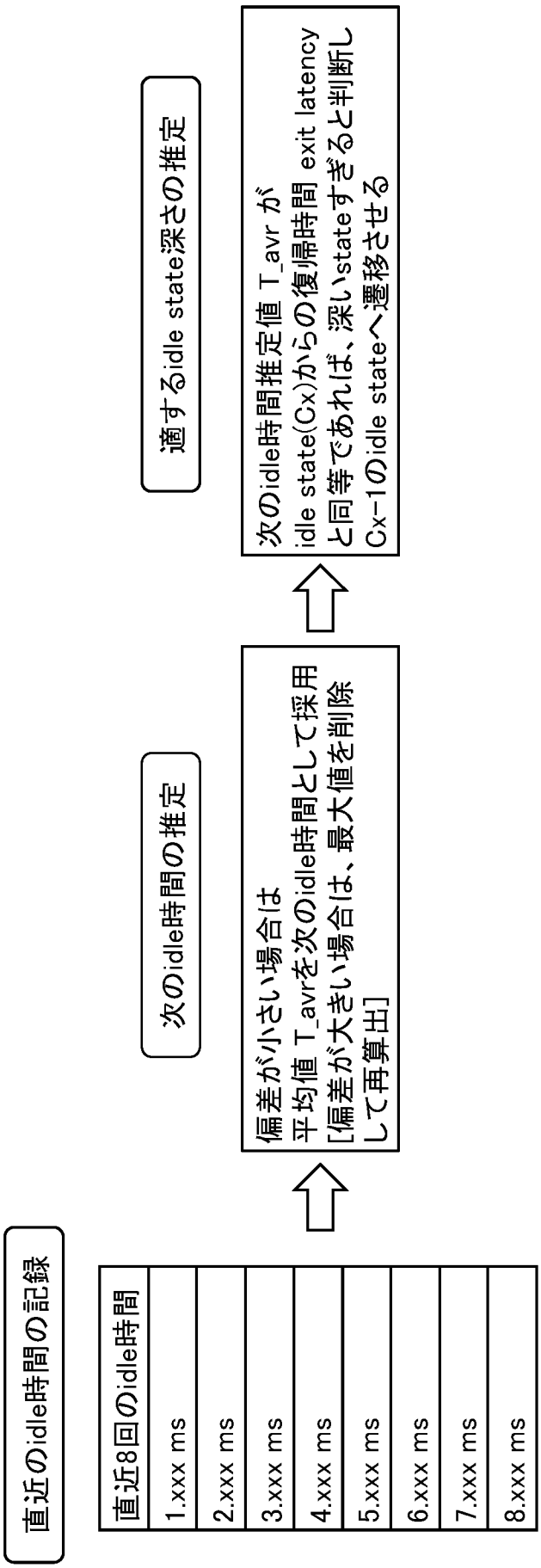
C-state	Residency	Wake-up latency
C0	Active	Active
C1	2us	2us
C1E	20us	10us
C3	100us	33us
C6	400us	133us

[図23]

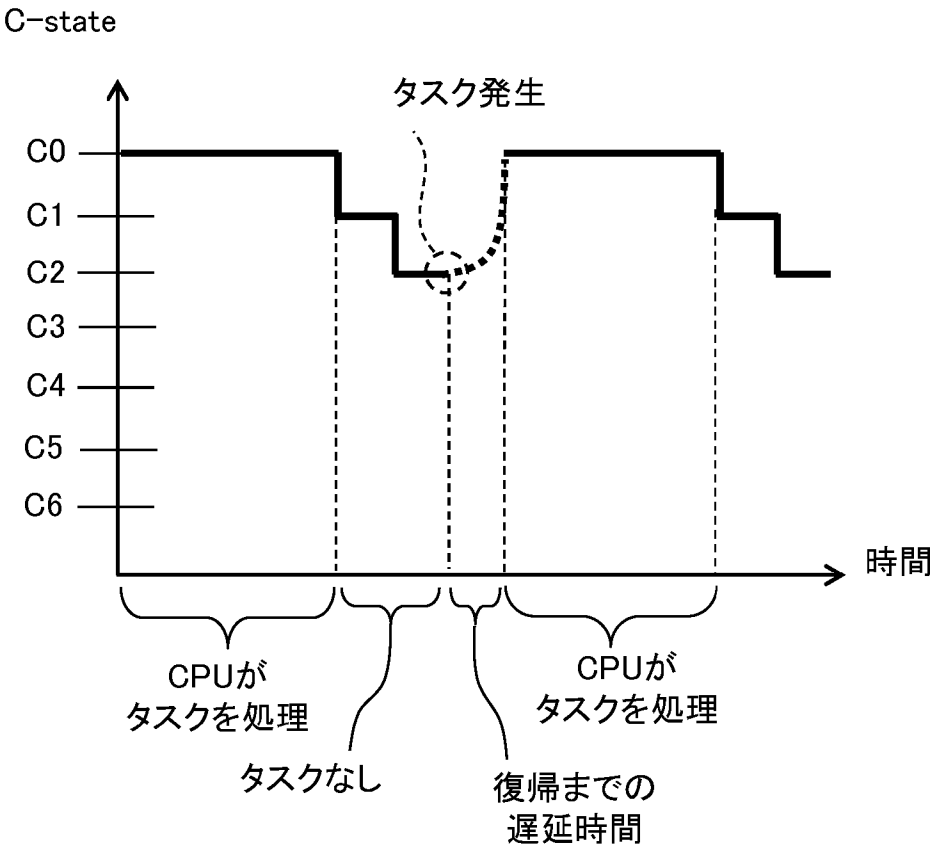


CPU idle Governor	適用性	概要
ladder	tickのあるシステム	最初は最も浅いstate遷移のみを許容し、tickまでの期間にタスクが無い場合が、段階的に深いstateへ遷移していく[tickが無いと、浅いstateのままとなるため、tickベースである必要あり]
menu	ticklessシステム	heuristicなアプローチでidle期間とidle stateを推定 直近8回のidle期間から、次のidle期間を推定し、idle期間に適するidle stateへ遷移させる[直接深いstateへ遷移させることが可能]

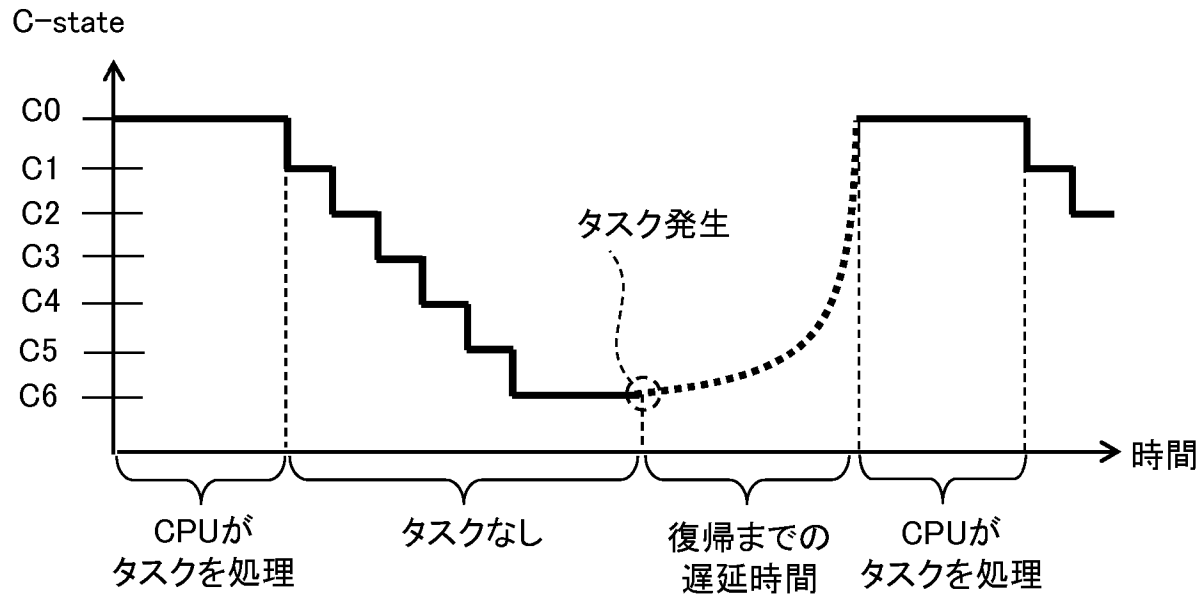
[図24]


直近のidle時間の記録
直近8回のidle時間
1.xxx ms
2.xxx ms
3.xxx ms
4.xxx ms
5.xxx ms
6.xxx ms
7.xxx ms
8.xxx ms
次のidle時間の推定
偏差が小さい場合は
平均値 T_avrを次のidle時間として採用
[偏差が大きい場合は、最大値を削除
して再算出]
適するidle state深さの推定
次のidle時間推定値 T_avr が
idle state(Cx)からの復帰時間 exit latency
と同等であれば、深いstateすぎると判断し
Cx-1のidle stateへ遷移させる

[図25]


C-state
タスク発生
C0
C1
C2
C3
C4
C5
C6
時間
CPUが
タスクを処理
タスクなし
復帰までの
遅延時間
CPUが
タスクを処理

[図26]


C-state
C0
C1
C2
C3
C4
C5
C6
タスク発生
時間
CPUが
タスクを処理
タスクなし
復帰までの
遅延時間
CPUが
タスクを処理

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2022/027327

A. CLASSIFICATION OF SUBJECT MATTER

G06F 1/329(2019.01)i; ***G06F 1/3287***(2019.01)i; ***G06F 9/50***(2006.01)i
FI: G06F1/329; G06F1/3287; G06F9/50 120A

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F1/329; G06F1/3287; G06F9/50

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2022
Registered utility model specifications of Japan 1996-2022
Published registered utility model applications of Japan 1994-2022

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2015/0033055 A1 (KRISHNAPURA, Sheshaprasad G.) 29 January 2015 (2015-01-29) paragraphs [0014]-[0038], fig. 1-3	1, 4-7
A		2-3
Y	US 2008/0244227 A1 (GEE, Timothy W.) 02 October 2008 (2008-10-02) paragraphs [0023], [0052], [0054], fig. 1, 3-4	1, 4-7
Y	JP 2015-513336 A (TEXAS INSTRUMENTS JAPAN LIMITED) 07 May 2015 (2015-05-07) paragraph [0022]	4
A	JP 2018-169882 A (NEC CORP) 01 November 2018 (2018-11-01) entire text, all drawings	1-7

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
08 September 2022	**27 September 2022**
Name and mailing address of the ISA/JP **Japan Patent Office (ISA/JP)** **3-4-3 Kasumigaseki, Chiyoda-ku, Tokyo 100-8915** **Japan**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/JP2022/027327

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2015/0033055 A1	29 January 2015	(Family: none)	
US 2008/0244227 A1	02 October 2008	CN 101105711 A	
JP 2015-513336 A	07 May 2015	US 2013/0198540 A1 paragraph [0025] CN 104094191 A	
JP 2018-169882 A	01 November 2018	US 2018/0284875 A1 entire text, all drawings	

国際調査報告

国際出願番号
PCT/JP2022/027327

A. 発明の属する分野の分類（国際特許分類（ＩＰＣ））

G06F 1/329(2019.01)i; G06F 1/3287(2019.01)i; G06F 9/50(2006.01)i
FI: G06F1/329; G06F1/3287; G06F9/50 120A

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

G06F1/329; G06F1/3287; G06F9/50

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２０２２年
日本国実用新案登録公報	１９９６－２０２２年
日本国登録実用新案公報	１９９４－２０２２年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献のカテゴリー*	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
Y	US 2015/0033055 A1 (KRISHNAPURA SHESHAPRASAD G.) 29.01.2015 (2015-01-29) 段落[0014]-[0038],FIG.1-3	1,4-7
A		2-3
Y	US 2008/0244227 A1 (GEE TIMOTHY W) 02.10.2008 (2008-10-02) 段落[0023],[0052],[0054],FIG.1,FIG.3-FIG.4	1,4-7
Y	JP 2015-513336 A (日本テキサス・インスツルメンツ株式会社) 07.05.2015 (2015-05-07) 段落[0022]	4
A	JP 2018-169882 A (日本電気株式会社) 01.11.2018 (2018-11-01) 全文、全図	1-7

☐ Ｃ欄の続きにも文献が列挙されている。

☑ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー
“A” 特に関連のある文献ではなく、一般的技術水準を示すもの
“E” 国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
“L” 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
“O” 口頭による開示、使用、展示等に言及する文献
“P” 国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
“T” 国際出願日又は優先日後に公表された文献であって出願と抵触するものではなく、発明の原理又は理論の理解のために引用するもの
“X” 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
“Y” 特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
“&” 同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
08.09.2022	27.09.2022

名称及びあて先
日本国特許庁(ISA/JP)
〒100-8915
日本国
東京都千代田区霞が関三丁目４番３号

権限のある職員（特許庁審査官）
豊田　真弓 5E 8386
電話番号 03-3581-1101 内線 3521

国際調査報告
パテントファミリーに関する情報

国際出願番号
PCT/JP2022/027327

引用文献	公表日	パテントファミリー文献	公表日
US 2015/0033055 A1	29.01.2015	（ファミリーなし）	
US 2008/0244227 A1	02.10.2008	CN 101105711 A	
JP 2015-513336 A	07.05.2015	US 2013/0198540 A1 段落[0025] CN 104094191 A	
JP 2018-169882 A	01.11.2018	US 2018/0284875 A1 全文、全図	