(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2017-16541
(P2017-16541A)
(43)公開日　平成29年1月19日(2017.1.19)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/48	(2006.01)	G06F	9/46	452B	
G06F	9/50	(2006.01)	G06F	9/46	465C	

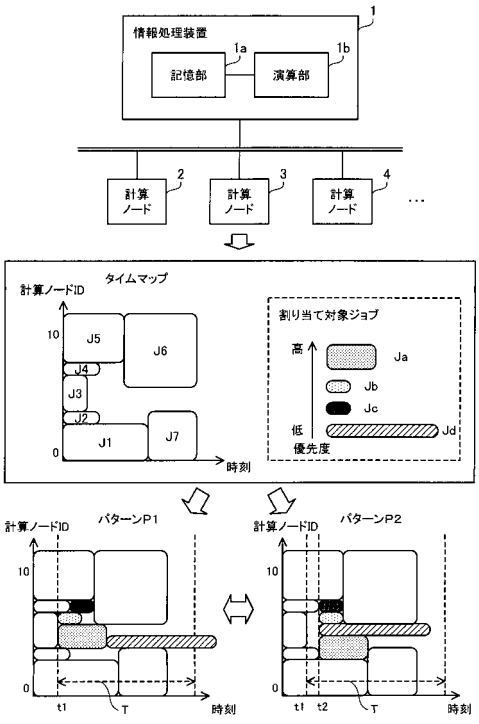
審査請求 未請求　請求項の数 8　ＯＬ　（全 44 頁）

(21)出願番号　特願2015-134895 (P2015-134895)
(22)出願日　平成27年7月6日 (2015.7.6)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100092152
弁理士　服部　毅巖
(72)発明者　佐藤　孝哉
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(54)【発明の名称】情報処理装置、並列計算機システム、ジョブスケジュール設定プログラムおよびジョブスケジュール設定方法

(57)【要約】
【課題】ジョブの実行を効率化すること。
【解決手段】記憶部１ａは、複数の計算ノードの利用可能な時間帯の情報を記憶する。演算部１ｂは、記憶部１ａに記憶された情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求める。演算部１ｂは、第１の時刻よりも遅い最優先のジョブの実行開始時刻の候補であり第１の時刻から最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、第２の時刻から最優先のジョブと他のジョブとを実行するよう複数の計算ノードに複数のジョブを割り当てる。
【選択図】図１



【特許請求の範囲】
【請求項１】
複数の計算ノードの利用可能な時間帯の情報を記憶する記憶部と、
前記情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる演算部と、
を有する情報処理装置。
【請求項２】
前記演算部は、前記第１の時刻および前記第２の時刻それぞれで前記最優先のジョブを実行開始する場合の前記複数のジョブの前記複数の計算ノードへの割り当てパターンの候補を示す情報を生成し、前記割り当てパターンの候補の比較に応じて、前記最優先のジョブを前記第１および前記第２の時刻の何れで実行するかを選択する、請求項１記載の情報処理装置。
【請求項３】
前記演算部は、前記割り当てパターンの候補に基づいて前記複数のジョブ全ての実行が完了する時刻を比較し、当該比較に応じて前記最優先のジョブを実行する時刻を選択する、請求項２記載の情報処理装置。
【請求項４】
前記演算部は、前記割り当てパターンの候補に基づいて前記複数の計算ノードの空き時間の量を比較し、当該比較に応じて前記最優先のジョブを実行する時刻を選択する、請求項２または３記載の情報処理装置。
【請求項５】
前記記憶部は、前記複数のジョブそれぞれの実行に要する計算ノードの数、および、当該計算ノードの利用時間を示す要求資源の情報を記憶し、
前記演算部は、前記複数の計算ノードの利用可能な時間帯と前記複数のジョブそれぞれの前記要求資源とに基づいて、前記第１および前記第２の時刻を特定する、
請求項１乃至４の何れか１つに記載の情報処理装置。
【請求項６】
複数のジョブを割り当て可能な複数の計算ノードと、
前記複数の計算ノードの利用可能な時間帯の情報に基づいて、前記複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる管理ノードと、
を有する並列計算機システム。
【請求項７】
コンピュータに、
複数の計算ノードの利用可能な時間帯の情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、
前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる、
処理を実行させるジョブスケジュール設定プログラム。
【請求項８】
コンピュータが、
複数の計算ノードの利用可能な時間帯の情報に基づいて、複数のジョブのうち最優先の

ジョブを実行可能な第１の時刻を求め、
前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる、
ジョブスケジュール設定方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は情報処理装置、並列計算機システム、ジョブスケジュール設定プログラムおよびジョブスケジュール設定方法に関する。
【背景技術】
【０００２】
複数の計算ノードを並列に用いて処理を実行するシステムが利用されている。当該システムでは、管理用のノードにより実行されるスケジューラと呼ばれるソフトウェアが、各計算ノードに対する処理の割り当てを行うことがある。例えば、バックフィルスケジューラにより、キュー内の優先順位の高いジョブからスケジューラマップ上の未使用空間にジョブを配置し、時間経過と共に予定されたジョブを実行する提案がある。バックフィルスケジューラにより、計算リソースを多用する大規模ジョブの実行開始を遅延させない範囲で小規模ジョブの実行を先行させて、システムの稼働率を向上させる提案もある。
【０００３】
また、複数の計算資源の何れかで実行中のジョブの一時停止を許容した場合および許容しない場合の２通りのスケジュールを作成し、両スケジュールの利益を示す値と損失を示す値との比較により適切なスケジュールを決定する提案もある。
【０００４】
なお、巡回セールスマン問題やジョブショップスケジューリングなどの組み合わせ問題を解く方法において、優先順位アルゴリズムにより発見される解に対し、再順序付けと呼ばれるプロセスによって、より良好な解を探索する提案がある。
【先行技術文献】
【特許文献】
【０００５】
【特許文献１】特開２０１２－２１５９３３号公報
【特許文献２】特開２０１２－１７３７５３号公報
【特許文献３】特開２０１３－４１５２９号公報
【特許文献４】国際公開第２０１２／０２０４７４号
【特許文献５】特開２００５－５６４２１号公報
【発明の概要】
【発明が解決しようとする課題】
【０００６】
上記のように、優先順位がつけられた各ジョブを実行できる最も早い時刻に、優先順位に従って順次実行させていくことが考えられる。しかし、この方法では、後続のジョブに対して計算資源の空き時間が優先順に従って割り当てられていくのみである。このため、計算資源の空き時間が細切れになって分断され易く、複数のジョブのスケジューリング結果をみた場合に、各ジョブに対して必ずしも効率的に計算資源を割り当てられているとは限らないという問題がある。例えば、全てのジョブを完了するまでの時間が長くなることがある。
【０００７】
１つの側面では、本発明は、ジョブの実行を効率化できる情報処理装置、並列計算機システム、ジョブスケジュール設定プログラムおよびジョブスケジュール設定方法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】
【０００８】
１つの態様では、情報処理装置が提供される。この情報処理装置は、記憶部と演算部とを有する。記憶部は、複数の計算ノードの利用可能な時間帯の情報を記憶する。演算部は、当該情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、第１の時刻よりも遅い最優先のジョブの実行開始時刻の候補であり第１の時刻から最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、第２の時刻から最優先のジョブと他のジョブとを実行するよう複数の計算ノードに複数のジョブを割り当てる。
【０００９】
また、１つの態様では、複数の計算ノードと管理ノードとを有する並列計算機システムが提供される。複数の計算ノードは、複数のジョブを割り当て可能である。管理ノードは、複数の計算ノードの利用可能な時間帯の情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、第１の時刻よりも遅い最優先のジョブの実行開始時刻の候補であり第１の時刻から最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、第２の時刻から最優先のジョブと他のジョブとを実行するよう複数の計算ノードに複数のジョブを割り当てる。
【００１０】
また、１つの態様では、ジョブスケジュール設定プログラムが提供される。このジョブスケジュール設定プログラムは、コンピュータに、複数の計算ノードの利用可能な時間帯の情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、第１の時刻よりも遅い最優先のジョブの実行開始時刻の候補であり第１の時刻から最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、第２の時刻から最優先のジョブと他のジョブとを実行するよう複数の計算ノードに複数のジョブを割り当てる、処理を実行させる。
【００１１】
また、１つの態様では、ジョブスケジュール設定方法が提供される。このジョブスケジュール設定方法では、コンピュータが、複数の計算ノードの利用可能な時間帯の情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、第１の時刻よりも遅い最優先のジョブの実行開始時刻の候補であり第１の時刻から最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、第２の時刻から最優先のジョブと他のジョブとを実行するよう複数の計算ノードに複数のジョブを割り当てる。
【発明の効果】
【００１２】
１つの側面では、ジョブの実行を効率化できる。
【図面の簡単な説明】
【００１３】
【図１】第１の実施の形態の情報処理装置を示す図である。
【図２】第２の実施の形態の並列計算機システムを示す図である。
【図３】管理ノードのハードウェア例を示す図である。
【図４】管理ノードの機能例を示す図である。
【図５】タイムマップの例を示す図である。
【図６】割り当て待ちジョブの優先度の例を示す図である。
【図７】割り当て待ちジョブ情報の例を示す図である。
【図８】割り当て対象ジョブを管理する構造体の例を示す図である。
【図９】優先度に従った割り当て対象ジョブの管理例を示す図である。
【図１０】利用可能資源を管理する構造体の例を示す図である。
【図１１】時刻毎の利用可能資源の管理例を示す図である。
【図１２】時刻ｔ１に対する利用可能資源の管理例を示す図である。

【図１３】時刻ｔ１に割り当てたジョブの管理例を示す図である。
【図１４】仮割り当てを管理する構造体の例を示す図である。
【図１５】時刻ｔ１を起点としたジョブの仮割り当ての例を示す図である。
【図１６】仮割り当てされた複数のジョブの管理例を示す図である。
【図１７】実行待ちジョブ情報の例を示す図である。
【図１８】パラメータ情報の例を示す図である。
【図１９】管理ノードのデータフローの例を示す図である。
【図２０】管理ノードの処理例を示す図である。
【図２１】割り当て候補時刻検出の例を示すフローチャートである。
【図２２】仮資源割り当ての例を示すフローチャートである。
【図２３】仮割り当て対象ジョブの選択例を示すフローチャートである。
【図２４】仮割り当て対象ジョブの配置例を示すフローチャートである。
【図２５】割り当て資源選択の例を示すフローチャートである。
【図２６】利用可能資源情報およびジョブ情報の具体例を示す図である。
【図２７】割り当て候補時刻の検出例を示す図である。
【図２８】割り当て候補時刻の上限の時刻の例を示す図である。
【図２９】仮割り当て対象ジョブの選択例を示す図である。
【図３０】仮割り当て対象ジョブの選択例（続き）を示す図である。
【図３１】仮割り当ての例を示す図である。
【図３２】仮割り当ての例（続き）を示す図である。
【図３３】更新後の利用可能資源情報の例を示す図である。
【図３４】仮割り当てパターンの例を示す図である。
【図３５】仮割り当て情報の例（パターンＰｔ１）を示す図である。
【図３６】仮割り当て情報の例（パターンＰｔ２）を示す図である。
【発明を実施するための形態】
【００１４】
以下、本実施の形態を図面を参照して説明する。
［第１の実施の形態］
図１は、第１の実施の形態の情報処理装置を示す図である。情報処理装置１は、ネットワークを介して複数の計算ノードに接続されている。複数の計算ノードは、計算ノード２，３，４を含む。情報処理装置１は、複数の計算ノードを用いて、複数のジョブを並列に実行させる。ジョブは、情報処理装置１が各計算ノードに割り当てる処理の単位である。ネットワークに接続された他の情報処理装置が情報処理装置１にジョブの実行要求を送信してもよい。
【００１５】
情報処理装置１は、各ジョブに優先度を付与して管理する。通常、優先度が相対的に高いジョブは、優先度が相対的に低いジョブよりも早く実行開始される。ジョブの優先度は、例えば、ジョブの実行要求の到着順、ユーザが属するグループやユーザの優先度、ジョブの実行要求を発行したアプリケーションの重要度などに応じて決定され得る。情報処理装置１は、優先度に従って、複数のジョブを複数の計算ノードに割り当てる。情報処理装置１は、１つのジョブの処理を複数の計算ノードに跨って割り当て、各計算ノードにより当該ジョブを並列実行させることもある。複数の計算ノードそれぞれは、自身に割り当てられたジョブを実行する。情報処理装置１は、「コンピュータ」と呼ばれてもよい。
【００１６】
ここで、情報処理装置１は、複数の計算ノードそれぞれを、計算ノードＩＤ（IDentifier）により識別する。計算ノードＩＤは、複数の計算ノードそれぞれの識別情報である。例えば、情報処理装置１は、１２個の計算ノードを、計算ノードＩＤ“０”～“１１”により識別する。
【００１７】
情報処理装置１は、記憶部１ａおよび演算部１ｂを有する。記憶部１ａは、ＲＡＭ（Ra

ndom Access Memory）などの揮発性の記憶装置でもよいし、ＨＤＤ（Hard Disk Drive）などの不揮発性の記憶装置でもよい。演算部１ｂは、例えば、プロセッサである。プロセッサは、ＣＰＵ（Central Processing Unit）やＤＳＰ（Digital Signal Processor）であってもよく、ＡＳＩＣ（Application Specific Integrated Circuit）やＦＰＧＡ（Field Programmable Gate Array）などの集積回路を含んでもよい。プロセッサは、例えば、ＲＡＭに記憶されたプログラムを実行する。また、「プロセッサ」は、２以上のプロセッサの集合（マルチプロセッサ）であってもよい。

【００１８】

記憶部１ａは、複数の計算ノードの利用可能な時間帯を示す第１の情報を記憶する。第１の情報は、複数の計算ノードそれぞれの利用可能時間のうち、何れのジョブも割り当てられていない時間帯を示す情報ということもできる。第１の実施の形態の例では、計算ノード２，３，４を含む複数のノードに対して、ジョブＪ１，Ｊ２，Ｊ３，Ｊ４，Ｊ５，Ｊ６，Ｊ７が既にスケジュール済である（図１のタイムマップに相当）。複数の計算ノードのうち、何れのジョブに対しても割り当てられていない時間帯が、該当の計算ノードの利用可能な時間帯である。

【００１９】

記憶部１ａは、複数のジョブそれぞれの計算ノードの所要量を示す第２の情報を記憶する。計算ノードの所要量は、例えば、ジョブを実行するために用いられる計算ノードの数や利用時間を含み得る。例えば、計算ノードを１つの計算機（情報処理装置、または、コンピュータ）と考えてもよい。また、計算機が複数のプロセッサを含む場合、複数のプロセッサのうち、ジョブの割り当て単位であるプロセッサ群（および、当該プロセッサ群によって利用されるメモリ）を１つの計算ノードと考えてもよい。あるいは、プロセッサが複数のコアを含む場合、複数のコアのうち、ジョブの割り当て単位であるコア群（および、当該コア群によって利用されるメモリ）を１つの計算ノードと考えてもよい。

【００２０】

例えば、第２の情報は、ジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄそれぞれの実行に用いられる計算ノードの所要量を含む。ジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄそれぞれは、複数の計算ノードに対して未割り当て（未スケジューリング）のジョブであり、今回の割り当て対象（スケジューリング対象）のジョブである。

【００２１】

ジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄそれぞれには優先度が付与されている。ジョブＪａの優先度は最も高い。ジョブＪｂの優先度は、ジョブＪａの優先度よりも低く、ジョブＪｃの優先度よりも高い。ジョブＪｃの優先度は、ジョブＪｂの優先度よりも低く、ジョブＪｄの優先度よりも高い。ジョブＪｄの優先度は最も低い。

【００２２】

演算部１ｂは、記憶部１ａに記憶された第１および第２の情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求める。第１の時刻は、最優先のジョブの実行を開始する候補時刻の１つであるともいえる。例えば、演算部１ｂは、ジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄのうち、最優先のジョブＪａを実行可能な第１の時刻として、時刻ｔ１を求める。演算部１ｂは、ジョブＪａを時刻ｔ１から開始した場合のジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄの優先度に従った仮のスケジューリングを試みる。この場合、優先度が高いジョブは、優先度が低いジョブよりも後に実行されないようにスケジューリングされる（優先度の高いジョブと優先度の低いジョブとが同じ時刻に実行開始されるようスケジューリングされることはある）。その結果、演算部１ｂは、スケジューリングの候補のパターンＰ１を得る。

【００２３】

演算部１ｂは、第１および第２の情報に基づいて、第１の時刻よりも遅い最優先のジョブの実行開始時刻の候補であり第１の時刻から最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻があるかを判定する。所定の時刻範囲の情報は、例えば、第１の時刻を起点とした期間Ｔであり、記憶部１ａに予め格納される。

【００２４】
例えば、演算部１ｂは、ジョブＪａの実行開始時刻を、時刻ｔ１（第１の時刻）よりも遅い時刻ｔ２（第２の時刻）とした場合の、ジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄの優先度に従った仮のスケジューリングを試みる。その結果、演算部１ｂは、スケジューリングの候補のパターンＰ２を得る。例えば、演算部１ｂは、パターンＰ１，Ｐ２を比較することで、期間Ｔ内におけるジョブの実行数が、パターンＰ１よりもパターンＰ２の方が多いと判断する。したがって、演算部１ｂは、ジョブＪａの実行開始時刻を時刻ｔ１よりも遅い時刻ｔ２とすることで、ジョブＪａを時刻ｔ１で実行するよりも所定の時刻範囲（例えば、期間Ｔ）内に多くのジョブを実行できると判定する。

【００２５】
具体的には、パターンＰ１では、期間Ｔ内にジョブＪｄの実行が完了しないので、期間Ｔ内に実行完了できるジョブの数は、ジョブＪａ，Ｊｂ，Ｊｃの３つである。一方、パターンＰ２では、期間Ｔ内にジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄの４つのジョブの実行を完了できる。よって、ジョブＪａを時刻ｔ１で実行開始するよりも、ジョブＪａを時刻ｔ２で実行開始する方が、期間Ｔ内に多くのジョブを実行できることになる。

【００２６】
この場合、演算部１ｂは、最優先のジョブの実行開始時刻を第１の時刻よりも遅らせて、第２の時刻から当該最優先のジョブと他のジョブとを実行するよう複数の計算ノードに複数のジョブを割り当てる。例えば、演算部１ｂは、ジョブＪａの実行開始時刻を時刻ｔ１から時刻ｔ２に遅らせる。すなわち、演算部１ｂは、パターンＰ２のスケジューリング結果を採用し、最優先のジョブＪａおよび他のジョブＪｂ，Ｊｃ，Ｊｄを時刻ｔ２に実行開始するよう、複数の計算ノードにジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄを割り当てる。

【００２７】
これにより、ジョブの実行を効率化できる。例えば、優先順位がつけられた各ジョブを実行できる最も早い時刻に、優先順位に従って順次実行させていく方法も考えられる。しかし、単に優先度に従って最も早い時刻にジョブを割り当てるのみでは、各ジョブに対して必ずしも効率的に計算資源を割り当てられているとは限らない。全てのジョブを完了するまでの時間が長くなることもある。例えば、パターンＰ１で例示したように、複数のジョブのうちの一部のジョブの実行が、期間Ｔ内に完了しないこともある。一方、パターンＰ２で例示したように、最優先のジョブＪａの実行開始時刻を時刻ｔ１から時刻ｔ２に遅らせることで、当該複数のジョブの実行が、期間Ｔ内に全て完了することもあり得る。

【００２８】
そこで、情報処理装置１は、ジョブＪａのスケジューリングの際に、後続のジョブＪｂ，Ｊｃ，Ｊｄも考慮する。具体的には、情報処理装置１は、ジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄのスケジューリングの候補となるパターンを複数取得し、当該複数のパターンの比較により、期間Ｔ内により多くのジョブを実行できるパターンを採用する。情報処理装置１は、採用したパターンに従って、各計算ノードへのジョブの割り当てを行う。例えば、パターンＰ２では、ジョブＪａ，Ｊｂ，Ｊｃ，Ｊｄの全ての実行が完了するまでの時間を、パターンＰ１よりも早めることができる。また、パターンＰ２ではパターンＰ１よりも計算ノードの空き時間を低減でき、ジョブの優先度を維持したままジョブスループットを向上できる。こうして、ジョブの実行を効率化することができる。

【００２９】
［第２の実施の形態］
図２は、第２の実施の形態の並列計算機システムを示す図である。第２の実施の形態の並列計算機システムは、管理ノード１００および複数の計算ノードを含む。複数の計算ノードは、計算ノード２００，３００，４００を含む。管理ノード１００および複数の計算ノードは、ネットワーク１０に接続されている。ネットワーク１０は、例えば、ＬＡＮ（Local Area Network）である。

【００３０】
管理ノード１００は、複数の計算ノードに対する複数のジョブの割り当てを行う。管理

ノード１００は、ネットワーク１０を介して、他の情報処理装置（図示を省略）からジョブの実行要求を受信してもよい。管理ノード１００は、第１の実施の形態の情報処理装置１の一例である。

【００３１】

計算ノード２００，３００，４００は、管理ノード１００により割り当てられたジョブを実行する。第２の実施の形態の例では、計算ノード２００，３００，４００それぞれは、１つの計算機である（他の計算ノードも同様）。計算ノード２００，３００，４００は、第１の実施の形態の計算ノード２，３，４の一例である。

【００３２】

第２の実施の形態の並列計算機システムでは、管理ノード１００により、複数のジョブを複数の計算ノードに割り当てることで、複数のジョブを並列に実行させることができる。第２の実施の形態の並列計算機システムの例では、ある時間帯に、あるジョブの実行が割り当てられた計算ノードは、該当の時間帯において該当のジョブを実行するために専有される。あるジョブに対して割り当てられる計算資源の所要量は、「計算ノードの数×専有時間」で表わすことができる（計算資源の所要量を計算ノードの所要量ということもできる）。

【００３３】

上記のように、複数のジョブを複数の計算ノードにより並列に処理することで、高速処理を実現するシステムを、ＨＰＣ（High Performance Computing）システムと呼ぶことがある。

【００３４】

図３は、管理ノードのハードウェア例を示す図である。管理ノード１００は、ＣＰＵ１０１，１０２、ＲＡＭ１０３、ＨＤＤ１０４、画像信号処理部１０５、入力信号処理部１０６、媒体リーダ１０７および通信インタフェース１０８を有する。これらのハードウェアユニットは、管理ノード１００内でバスに接続されている。ＣＰＵ１０１，１０２は、第１の実施の形態の演算部１ｂの一例であり、ＲＡＭ１０３またはＨＤＤ１０４は、第１の実施の形態の記憶部１ａの一例である。

【００３５】

ＣＰＵ１０１，１０２は、プログラムの命令を実行する１または２以上のコアを含むプロセッサである。例えば、ＣＰＵ１０１は、コア１０１ａ，１０１ｂを含む複数のコアを有する。同一または異なるＣＰＵに属する複数のコアは、互いに並列に命令を実行できる。ＣＰＵ１０１，１０２は、ＨＤＤ１０４に記憶されているプログラムやデータの少なくとも一部をＲＡＭ１０３にロードし、プログラムを実行する。なお、各コアを「プロセッサ」と呼んでもよいし、複数のプロセッサの集合を「プロセッサ」（マルチプロセッサ）と呼んでもよい。

【００３６】

ＲＡＭ１０３は、ＣＰＵ１０１，１０２が実行するプログラムやＣＰＵ１０１，１０２が演算に用いるデータを一時的に記憶する揮発性メモリである。なお、管理ノード１００は、ＲＡＭ以外の種類のメモリを備えてもよく、複数個のメモリを備えてもよい。

【００３７】

ＨＤＤ１０４は、ＯＳ（Operating System）やアプリケーションソフトウェアなどのソフトウェアのプログラム、および、データを記憶する不揮発性の記憶装置である。なお、管理ノード１００は、フラッシュメモリやＳＳＤ（Solid State Drive）などの他の種類の記憶装置を備えてもよく、複数の不揮発性の記憶装置を備えてもよい。

【００３８】

画像信号処理部１０５は、ＣＰＵ１０１，１０２からの命令に従って、管理ノード１００に接続されたディスプレイ１１に画像を出力する。ディスプレイ１１としては、ＣＲＴ（Cathode Ray Tube）ディスプレイ、液晶ディスプレイ（ＬＣＤ：Liquid Crystal Display）、プラズマディスプレイ（ＰＤＰ：Plasma Display Panel）、有機ＥＬ（ＯＥＬ：Organic Electro-Luminescence）ディスプレイなどを用いることができる。

【００３９】
入力信号処理部１０６は、管理ノード１００に接続された入力デバイス１２から入力信号を取得し、少なくとも１つのＣＰＵに出力する。入力デバイス１２としては、マウスやタッチパネルやタッチパッドやトラックボールなどのポインティングデバイス、キーボード、リモートコントローラ、ボタンスイッチなどを用いることができる。また、管理ノード１００に、複数の種類の入力デバイスが接続されていてもよい。
【００４０】
媒体リーダ１０７は、記録媒体１３に記録されたプログラムやデータを読み取る読み取り装置である。記録媒体１３として、例えば、フレキシブルディスク（ＦＤ：Flexible Disk）やＨＤＤなどの磁気ディスク、ＣＤ（Compact Disc）やＤＶＤ（Digital Versatile Disc）などの光ディスク、光磁気ディスク（ＭＯ：Magneto-Optical disk）、半導体メモリなどを使用できる。媒体リーダ１０７は、例えば、記録媒体１３から読み取ったプログラムやデータをＲＡＭ１０３またはＨＤＤ１０４に格納する。
【００４１】
通信インタフェース１０８は、ネットワーク１０に接続され、ネットワーク１０を介して複数の計算ノードを含む他の情報処理装置と通信を行うインタフェースである。通信インタフェース１０８は、スイッチなどの通信装置とケーブルで接続される有線通信インタフェースでもよいし、基地局などと無線リンクで接続される無線通信インタフェースでもよい。
【００４２】
なお、管理ノード１００が備えるＣＰＵの数は１個でもよい。また、管理ノード１００は、媒体リーダ１０７を備えていなくてもよい。また、管理ノード１００は、ユーザが操作する端末装置からネットワーク１０経由で制御される場合には、画像信号処理部１０５や入力信号処理部１０６を備えていなくてもよい。また、管理ノード１００は、通信インタフェース１０８を備えていなくてもよい。また、ディスプレイ１１や入力デバイス１２が、管理ノード１００の筐体と一体に形成されていてもよい。更に、計算ノード２００，３００，４００を含む各計算ノードも、管理ノード１００と同様のユニットを用いて実現できる。各計算ノードは、管理ノード１００と同様に、画像信号処理部１０５、入力信号処理部１０６および媒体リーダ１０７の少なくとも何れかを備えていなくてもよい。
【００４３】
図４は、管理ノードの機能例を示す図である。管理ノード１００は、割り当て待ちジョブ記憶部１１０、割り当て対象ジョブ記憶部１２０、資源情報記憶部１３０、実行待ちジョブ記憶部１４０、パラメータ記憶部１５０、ジョブ受付部１６０、ジョブ選択部１７０、資源選択部１８０およびジョブ実行指示部１９０を有する。
【００４４】
割り当て待ちジョブ記憶部１１０、割り当て対象ジョブ記憶部１２０、資源情報記憶部１３０、実行待ちジョブ記憶部１４０およびパラメータ記憶部１５０は、ＲＡＭ１０３やＨＤＤ１０４に確保された記憶領域として実現される。ジョブ受付部１６０、ジョブ選択部１７０、資源選択部１８０およびジョブ実行指示部１９０は、ＲＡＭ１０３に記憶されたプログラムをＣＰＵ１０１，１０２が実行することで実現される。ジョブ受付部１６０、ジョブ選択部１７０、資源選択部１８０およびジョブ実行指示部１９０の機能を有するソフトウェアを、「ジョブスケジューラ」と呼ぶこともある。
【００４５】
割り当て待ちジョブ記憶部１１０は、投入されたジョブの中で資源割り当てがされていないジョブの情報（割り当て待ちジョブ情報）を記憶する。以下の説明では、投入されたジョブの中で資源割り当てがされていないジョブを指して、「割り当て待ちジョブ」と称することがある。割り当て待ちジョブの情報は、当該ジョブを実行するための計算資源の所要量（「計算ノード数」×「専有時間」）を含む。また、割り当て待ちジョブ情報は、後述する優先度の判断に用いられる情報（ジョブ毎の到着時間、発行元アプリケーションおよび当該アプリケーションを利用するユーザの情報など）を含み得る。

【００４６】
割り当て対象ジョブ記憶部１２０は、今回、計算資源の割り当ての対象とするジョブの情報を記憶する。割り当て対象ジョブ記憶部１２０に記憶される情報は、割り当て対象ジョブ情報および後続対象ジョブ情報を含む。
【００４７】
割り当て対象ジョブ情報は、割り当て待ちジョブの中で最もスケジューリング優先度が高いジョブを示す情報を含む。ここで、割り当て待ちジョブの中で最もスケジューリング優先度が高いジョブを指して、「割り当て対象ジョブ」と称することがある。
【００４８】
後続対象ジョブ情報は、割り当て対象ジョブと共に仮割り当ての評価対象とするジョブを示す情報を含む。ここで、割り当て対象ジョブと共に仮割り当ての評価対象とするジョブを指して、「後続対象ジョブ」と称することがある。
【００４９】
また、「仮割り当て」とは、割り当て対象ジョブおよび後続対象ジョブを、ある割り当て候補時刻から仮割り当て終端時刻まで、計算資源の割り当てを仮に（本番のスケジューリングに影響しないよう）行うことを示す。「割り当て候補時刻」は、割り当て対象ジョブを割り当て可能な時刻の中で、仮割り当てを行う時刻を示す。「仮割り当て終端時刻」は、仮割り当てを行う際の終端の時刻を示す。
【００５０】
資源情報記憶部１３０は、ジョブの実行に利用可能な計算資源の情報（利用可能資源情報）を記憶する。計算資源の情報は、ある時刻において、ある計算ノードでどの程度の時間をジョブの実行に利用可能であるかを示す情報である。
【００５１】
実行待ちジョブ記憶部１４０は、計算ノードの計算資源を割り当て済であり（スケジューリング済であるともいえる）、計算ノードによる実行を待機しているジョブの情報（実行待ちジョブ情報）を記憶する。以下の説明では、計算ノードによる時刻を待機しているジョブを指して、「実行待ちジョブ」と称することがある。実行待ちジョブに割り当てる計算資源の選択は、資源選択部１８０によって実行される。
【００５２】
パラメータ記憶部１５０は、ジョブ選択部１７０および資源選択部１８０の処理に用いられるパラメータ情報を記憶する。例えば、パラメータ情報は、後続対象ジョブの選択の基準となる情報や、計算資源選択の基準となる情報を含む。
【００５３】
ジョブ受付部１６０は、ジョブの実行要求を受け付ける。ジョブ受付部１６０は、受け付けたジョブに応じた割り当て待ちジョブ情報を割り当て待ちジョブ記憶部１１０に格納する。ジョブの実行要求は、管理ノード１００上で動作するアプリケーションによって発行されてもよいし、管理ノード１００や計算ノード２００，３００，４００とは異なる情報処理装置によって発行されてもよい。
【００５４】
ジョブ選択部１７０は、割り当て待ちジョブ記憶部１１０に記憶された割り当て待ちジョブ情報に基づいて、割り当て待ちジョブを所定の優先度に従ってソートする。ジョブ選択部１７０は、例えば、ジョブの実行要求の到着順、ユーザが属するグループやユーザの優先度、ジョブの実行要求を発行したアプリケーションの重要度などに応じて、ジョブの優先度を決定する。ここで、ジョブ選択部１７０は、割り当て対象ジョブ選択部１７１および後続対象ジョブ選択部１７２を有する。
【００５５】
割り当て対象ジョブ選択部１７１は、割り当て待ちジョブ記憶部１１０に記憶された割り当て待ちジョブ情報に基づいて、割り当て対象ジョブ情報を生成し、割り当て対象ジョブ記憶部１２０に格納する。割り当て対象ジョブ情報は、今回の計算資源の割り当て対象とする複数のジョブのうち先頭のジョブ（割り当て対象ジョブ）および終端のジョブを示

す情報を含む。先頭のジョブ（割り当て対象ジョブ）は、当該複数のジョブのうち最高の優先度のジョブである。終端のジョブは、当該複数のジョブのうち最低の優先度のジョブである。

【００５６】

後続対象ジョブ選択部１７２は、割り当て待ちジョブ記憶部１１０に記憶された割り当て待ちジョブ情報に基づいて後続対象ジョブ情報を生成し、割り当て対象ジョブ記憶部１２０に格納する。後続対象ジョブ選択部１７２は、ジョブの優先度順に後続対象ジョブの選択を行う。例えば、割り当て待ちジョブの数が多くなると後続対象ジョブの評価に時間がかかるようになる可能性がある。このため、例えば、パラメータ情報において仮割り当ての評価対象とするジョブ数の上限値を予め定める。後続対象ジョブ選択部１７２は、当該上限値となるまで、後続対象ジョブを選択する。それ以外にも、パラメータ情報において、システムにおける計算資源の空きに応じた割り当て対象ジョブ数の制限値を予め指定しておいてもよい。あるいは、後続対象ジョブ数の上限値を、割り当て待ちジョブ記憶部１１０に格納された全割り当て待ちジョブ数としてもよい。

【００５７】

上記のように、管理ノード１００は、割り当て対象ジョブ情報により、先頭のジョブと終端のジョブとを指定することで、後続対象ジョブ情報を基に、先頭のジョブから終端のジョブまでの一連のジョブを割り当ての評価対象のジョブとして特定できる。

【００５８】

資源選択部１８０は、割り当て対象ジョブ記憶部１２０に記憶された割り当て対象ジョブ情報および後続対象ジョブ情報に基づいて、割り当て対象ジョブおよび後続対象ジョブに対する計算資源の割り当てを行う。資源選択部１８０は、割り当て候補時刻検出部１８１、仮資源割り当て部１８２および割り当て資源選択部１８３を有する。

【００５９】

割り当て候補時刻検出部１８１は、割り当て対象ジョブ記憶部１２０に記憶された割り当て対象ジョブ情報に基づいて、割り当て対象ジョブを特定する。割り当て候補時刻検出部１８１は、資源情報記憶部１３０に記憶された資源情報およびパラメータ記憶部１５０に記憶されたパラメータ情報に基づいて、割り当て対象ジョブに対して計算資源を割り当てる起点の時刻の候補（割り当て候補時刻）を複数検出する。

【００６０】

仮資源割り当て部１８２は、割り当て候補時刻検出部１８１により検出された複数の割り当て候補時刻それぞれを起点として、割り当て対象ジョブおよび後続対象ジョブに対する計算資源の仮割り当てを行う。例えば、仮資源割り当て部１８２は、第１の割り当て候補時刻に着目し、当該第１の割り当て候補時刻を起点に、割り当て対象ジョブおよび後続対象ジョブのタイムマップ上への配置を試みる。また、仮資源割り当て部１８２は、第２の割り当て候補時刻に着目し、当該第２の割り当て候補時刻を起点に割り当て対象ジョブおよび後続対象ジョブのタイムマップ上への配置を試みる。ジョブ配置にはジョブの優先度が考慮される。その結果、仮資源割り当て部１８２は、割り当て対象ジョブおよび後続対象ジョブに対する計算資源の割り当てパターンの複数の候補を得る。当該割り当てパターンの候補を、仮割り当てパターンと称することがある。仮資源割り当て部１８２は、複数の仮割り当てパターンの情報を、資源情報記憶部１３０に格納する。

【００６１】

割り当て資源選択部１８３は、資源情報記憶部１３０に記憶された複数の仮割り当てパターンの情報に基づいて、割り当て対象ジョブおよび後続対象ジョブに対して採用する計算資源の仮割り当てパターンを選択する。選択の基準は、例えばパラメータ記憶部１５０に記憶されたパラメータ情報に予め登録される。例えば、パラメータ情報は、仮割り当てパターンの選択のための基準として、（１）所定期間中に実行可能なジョブの数が大きい、（２）各ジョブの実行完了が早い、（３）所定期間中における空きの計算資源の量が少ない、などの条件を示す情報を含み得る。所定期間は、予め定められる時刻範囲であり、最優先のジョブを実行可能な最も早い時刻を起点とする当該時刻以降の一定の期間でもよ

いし、現時刻を起点とする当該時刻以降の一定の期間でもよい。各条件は、他の条件に対する相対的な重みをもつ。重みの大きな条件に適合するほど、パターン選択の優先度は高い。割り当て資源選択部１８３は、選択した仮割り当てパターンに基づいて、実行待ちジョブ情報を生成し、実行待ちジョブ記憶部１４０に格納する。

【００６２】

ジョブ実行指示部１９０は、実行待ちジョブ記憶部１４０に記憶された実行待ちジョブ情報に基づいて、複数の計算ノードそれぞれに対するジョブの実行を指示する。

図５は、タイムマップの例を示す図である。図５では、１２個の計算ノードに対して７つのジョブＪ１，Ｊ２，Ｊ３，Ｊ４，Ｊ５，Ｊ６，Ｊ７の実行が割り当てられた状態を示している。計算ノードは、計算ノードＩＤによって識別される。１２個の計算ノードには、例えば、計算ノードＩＤ“０”～“１１”が付与されている。図５のようなスケジュール内容を示した情報を、タイムマップと称することがある。図５では、何れかのジョブが開始または完了する時刻ｔ０～ｔ５も図示されている。

【００６３】

図５の例によれば、時刻ｔ０～ｔ４の期間において、計算ノードＩＤ“０”～“２”で示される３つの計算ノードにジョブＪ１が割り当てられている。時刻ｔ０～ｔ２の期間において、計算ノードＩＤ“３”で示される１つの計算ノードにジョブＪ２が割り当てられている。時刻ｔ０～ｔ１の期間において、計算ノードＩＤ“４”～“６”で示される３つの計算ノードにジョブＪ３が割り当てられている。時刻ｔ０～ｔ２の期間において、計算ノードＩＤ“７”で示される１つの計算ノードにジョブＪ４が割り当てられている。時刻ｔ０～ｔ３の期間において、計算ノードＩＤ“８”～“１１”で示される４つの計算ノードにジョブＪ５が割り当てられている。時刻ｔ３～ｔ５の期間において、計算ノードＩＤ“６”～“１１”で示される６つの計算ノードにジョブＪ６が割り当てられている。時刻ｔ４～ｔ５において、計算ノードＩＤ“０”～“３”で示される４つの計算ノードにジョブＪ７が割り当てられている。上記１２個の計算ノードについて、時刻ｔ０以降、ジョブＪ１，Ｊ２，Ｊ３，Ｊ４，Ｊ５，Ｊ６，Ｊ７に既に割り当て済の計算資源以外の計算資源が、利用可能資源Ｒ１である。利用可能資源Ｒ１は、新規のジョブに対して割り当てることが可能である。

【００６４】

図６は、割り当て待ちジョブの優先度の例を示す図である。ジョブは、ジョブＩＤによって識別される。以下の説明では、ジョブＩＤが“Ａ”であるジョブを指して、“ジョブＡ”のように表記することがある。

【００６５】

例えば、割り当て待ちジョブは、ジョブＡ，Ｂ，Ｃ，Ｄ，・・・を含む。割り当て対象ジョブ選択部１７１は、割り当て待ちジョブ記憶部１１０を参照して、例えば、ジョブＡを割り当て対象ジョブとして選択する。後続対象ジョブ選択部１７２は、ジョブＢ，Ｃ，Ｄを後続対象ジョブとして選択する。割り当て対象ジョブの選択の際には、各ジョブの優先度（スケジューリング優先度）が考慮される。すなわち、割り当て対象ジョブ選択部１７１は、割り当て待ちジョブの優先度の高い順に、割り当て対象ジョブを選択する。先頭のジョブはジョブＡである。終端のジョブはジョブＤである。この場合、割り当て対象ジョブ選択部１７１は、終端のジョブをジョブＤと認識する。幾つのジョブを割り当て対象ジョブとして選択するかは、前述のようにパラメータ情報としてパラメータ記憶部１５０に予め設定される（例えば、４つ選択する旨が予め設定される）。

【００６６】

図７は、割り当て待ちジョブ情報の例を示す図である。割り当て待ちジョブ情報１１１は、割り当て待ちジョブ記憶部１１０に格納される。割り当て待ちジョブ情報１１１は、ジョブＩＤ、要求資源量および資源使用時間の項目を含む。

【００６７】

ジョブＩＤの項目には、ジョブの識別情報（ジョブＩＤ）が登録される。要求資源量の項目には、該当のジョブを実行するために要求されている計算ノードの数が登録される。

資源使用時間の項目には、該当のジョブを実行するために要求されている各計算ノードの使用時間が登録される。

【００６８】

例えば、割り当て待ちジョブ情報１１１には、ジョブＩＤが“Ａ”、要求資源量が“２”、資源使用時間が“ｔＡ”という情報が登録されている。これは、割り当て待ちジョブＡが、計算ノードを２つ、時間ｔＡだけ専有して実行されるジョブであることを示す。この場合、ジョブＡの計算資源の所要量は、２×ｔＡである。

【００６９】

割り当て待ちジョブ情報１１１には、他のジョブについても、同様に要求資源量や資源使用時間の情報が登録される。ジョブＢは、計算ノードを１つ、時間ｔＢだけ専有して実行されるジョブである。ジョブＣは、計算ノードを１つ、時間ｔＣだけ専有して実行されるジョブである。ジョブＤは、計算ノードを１つ、時間ｔＤだけ専有して実行されるジョブである。

【００７０】

なお、割り当て待ちジョブ情報１１１は、優先度の判定に用いられる情報を含んでもよい。優先度の判定に用いられる情報は、例えば、ジョブの実行要求の到着時刻、ジョブの実行要求の発行元のソフトウェアや当該ソフトウェアを利用するユーザ（または、当該ユーザが属するグループ）などの情報を含み得る。あるいは、割り当て待ちジョブ情報１１１の各レコードが、優先度の順にソートされていてもよい。

【００７１】

図８は、割り当て対象ジョブを管理する構造体の例を示す図である。割り当て対象ジョブおよび後続対象ジョブの管理には、割り当て対象ジョブ情報および後続対象ジョブ情報が用いられる。割り当て対象ジョブ情報および後続対象ジョブ情報は、割り当て対象ジョブ記憶部１２０に格納される。図８では、割り当て対象ジョブ情報の構造体１２１（schedjobinfo）および後続対象ジョブ情報の構造体１２２（jobinfo）の例を示す。

【００７２】

構造体１２１は、割り当て対象ジョブ情報の構造体の例である。構造体１２１は、ジョブ情報の数（num__jobs）、先頭ジョブ情報へのポインタ（jobinfo *head__p）および終端ジョブ情報へのポインタ（jobinfo *tail__p）を含む。ジョブ情報の数は、割り当て対象ジョブ情報で示される後続対象ジョブ情報の数（先頭ジョブ、終端ジョブおよび先頭ジョブから終端ジョブの間のジョブの数）である。先頭ジョブ情報へのポインタは、先頭ジョブに相当する後続対象ジョブ情報のＲＡＭ１０３上の格納アドレスを示す。終端ジョブ情報へのポインタは、終端ジョブに対応する後続対象ジョブ情報のＲＡＭ１０３上の格納アドレスを示す。構造体１２１を用いてＲＡＭ１０３上に格納された具体的なデータを、構造体１２１のインスタンスと呼ぶことができる（他の構造体についても同様）。割り当て対象ジョブ情報は、構造体１２１の１つのインスタンスである。

【００７３】

構造体１２２は、後続対象ジョブ情報の構造体の例である。構造体１２２は、次のジョブ情報へのポインタ（jobinfo *next__p）、ジョブＩＤ（jid）、要求資源量（num__reqnids）および資源使用時間（timespec reqtime）の情報を含む。次のジョブ情報へのポインタは、次の後続対象ジョブ情報のＲＡＭ１０３上の格納アドレスを示す。ジョブＩＤ、要求資源量および資源使用時間の情報は、図７で例示した割り当て待ちジョブ情報１１１の同名の項目に相当する情報である。割り当て対象ジョブ情報は、構造体１２２の１つのインスタンスである。後続対象ジョブ情報は、構造体１２２を用いてジョブ毎に作成され得る。

【００７４】

図９は、優先度に従った割り当て対象ジョブの管理例を示す図である。割り当て対象ジョブおよび後続対象ジョブは、割り当て対象ジョブ情報および後続対象ジョブ情報により、各ジョブの優先度に従って管理される。

【００７５】

具体的には、先頭ジョブへのポインタ（head__p）で示される後続対象ジョブ情報が、先頭ジョブ（割り当て対象ジョブ）の情報である。一方、終端ジョブへのポインタ（tail__p）で示される後続対象ジョブ情報が、終端ジョブの情報である。終端ジョブは、割り当て対象ジョブおよび後続対象ジョブの中で最も優先度が低い。そして、次のジョブ情報へのポインタ（next__p）により、優先度が自ジョブの次に低いジョブのジョブ情報を示す。管理ノード１００は、このようなデータ構造により、先頭ジョブから終端ジョブまで、優先度の高い順に、ジョブの序列を管理する。

【００７６】

図１０は、利用可能資源を管理する構造体の例を示す図である。利用可能資源の管理には、利用可能資源管理情報、利用可能資源情報、資源情報、仮割り当て作業情報および仮割り当てジョブ情報が用いられる。利用可能資源管理情報、利用可能資源情報、資源情報、仮割り当て作業情報および仮割り当てジョブ情報は、資源情報記憶部１３０に格納される。図１０では、利用可能資源管理情報の構造体１３１（availrschead）、利用可能資源情報の構造体１３２（availrsc）、資源情報の構造体１３３（availrscinfo）、仮割り当て作業情報の構造体１３４（allocrscinfo）および仮割り当てジョブ情報の構造体１３５（allocrscjobinfo）の例を示す。

【００７７】

構造体１３１は、利用可能資源管理情報の構造体の例である。構造体１３１は、先頭の利用可能資源情報へのポインタ（availrsc *head__p）を含む。利用可能資源管理情報は、構造体１３１の１つのインスタンスである。

【００７８】

構造体１３２は、利用可能資源情報の構造体の例である。構造体１３２は、次の要素（利用可能資源情報）へのポインタ（availrsc *next__p）、利用可能資源の時刻（timespec time）、資源情報の数（num__rsc）、先頭の資源情報へのポインタ（availrscinfo *arihead__p）および終端の資源情報へのポインタ（availrscinfo *aritail__p）を含む。

【００７９】

利用可能資源の時刻は、計算資源の空き期間の開始時刻である。利用可能資源情報では、当該開始時刻から利用可能な１以上の計算資源の情報（資源情報）が管理される。資源情報の数は、利用可能資源情報で管理される資源情報の数である。先頭の資源情報へのポインタは、先頭の資源情報のＲＡＭ１０３上の格納アドレスを示す。終端の資源情報へのポインタは、終端の資源情報のＲＡＭ１０３上の格納アドレスを示す。利用可能資源情報は、構造体１３２の１つのインスタンスである。利用可能資源情報は、構造体１３２を用いて利用可能資源の時刻毎に作成され得る。

【００８０】

構造体１３３は、資源情報の構造体の例である。構造体１３３は、次の資源情報へのポインタ（availrscinfo *next__p）、前の資源情報へのポインタ（availrscinfo *prev__p）、計算資源の利用可能時間（timespec availtime）、利用可能ノード数（num__nids）、利用可能ノードＩＤを示すポインタ（*nids__p）および仮割り当て作業域（allocrscinfo allocinfo）を含む。

【００８１】

次の資源情報へのポインタは、次の資源情報のＲＡＭ１０３上の格納アドレスを示す。前の資源情報へのポインタは、前の資源情報のＲＡＭ１０３上の格納アドレスを示す。計算資源の利用可能時間は、利用可能資源情報に含まれる時刻を起点とした計算資源の利用可能時間である。当該時刻と利用可能時間とによって、タイムマップにおける１つの期間を指定できる。利用可能ノード数は、利用可能な計算ノードの数である。利用可能ノードＩＤを示すポインタは、利用可能な計算ノードの計算ノードＩＤのリストが格納されたＲＡＭ１０３上の格納アドレスを示す。仮割り当て作業域は、ジョブに対する計算資源の仮割り当てを行うための作業用の情報である。資源情報は、構造体１３３の１つのインスタンスである。資源情報は、構造体１３３を用いて複数作成され得る。

【００８２】

構造体１３４は、仮割り当て作業情報の構造体の例である。構造体１３４は、全仮割り当てノード数（total__alloc__nids）、仮割り当てジョブ情報の数（num__ajs）および先頭仮割り当てジョブ情報へのポインタ（allocrscjobinfo *aj__p）を含む。

【００８３】

全仮割り当てノード数は、ジョブを仮割り当て済である計算ノード数である。仮割り当てジョブ情報の数は、仮割り当てされたジョブの情報（すなわち、当該ジョブ）の数である。先頭仮割り当てジョブ情報へのポインタは、先頭の仮割り当てジョブ情報のＲＡＭ１０３上の格納アドレスを示す。仮割り当て作業情報は、構造体１３４の１つのインスタンスである。仮割り当て作業情報は、構造体１３４を用いて複数作成され得る。

【００８４】

構造体１３５は、仮割り当てジョブ情報の構造体の例である。構造体１３５は、次の仮割り当てジョブ情報へのポインタ（allocrscjobinfo *next__p）、ジョブＩＤ（jid）および獲得資源数（num__alloc__nids）を含む。次の仮割り当てジョブ情報へのポインタは、次の仮割り当て情報のＲＡＭ１０３上の格納アドレスを示す。ジョブＩＤは、仮割り当てされたジョブのジョブＩＤである。獲得資源数は、当該ジョブＩＤで示されるジョブによって獲得されている（当該ジョブに割り当てられている）計算ノードの数である。仮割り当てジョブ情報は、構造体１３５の１つのインスタンスである。仮割り当てジョブ情報は、構造体１３５を用いて複数作成され得る。

【００８５】

図１１は、時刻毎の利用可能資源の管理例を示す図である。時刻ｔ０，ｔ１，ｔ２，ｔ３，ｔ４，ｔ５それぞれに対する利用可能資源は、利用可能資源情報によって管理される。利用可能資源管理情報は、先頭の利用可能資源情報に対するポインタを含む。ここで、各利用可能資源情報は、時刻の早い順に、ポインタ（availrsc *next__p）によって連結される。時刻ｔ０～ｔ５では、時刻ｔ０が最も早い時刻である。よって、先頭の利用可能資源情報は、時刻ｔ０に関する利用可能資源情報となる。

【００８６】

以降、時刻ｔ０の利用可能資源情報のポインタは、時刻ｔ１の利用可能資源情報を示す。時刻ｔ１の利用可能資源情報のポインタは、時刻ｔ２の利用可能資源情報を示す（時刻ｔ２以降の利用可能資源情報の図示を省略している）。時刻ｔ３の利用可能資源情報のポインタは、時刻ｔ４の利用可能資源情報を示す。時刻ｔ４の利用可能資源情報のポインタは、時刻ｔ５の利用可能資源情報を示す。

【００８７】

図１２は、時刻ｔ１に対する利用可能資源の管理例を示す図である。時刻ｔ１において、利用可能資源は、次の２つに区分される。第１の利用可能資源は、計算ノードＩＤ“６”の１つの計算ノードの時刻ｔ１以降のｔ３－ｔ１（利用可能時間）の時間帯である。ここで、時刻に対するｔ３－ｔ１の演算は、時刻ｔ３と時刻ｔ１との時間差を求める演算である（他の時刻についても同様）。すなわち、計算ノードＩＤ“６”を時刻ｔ１から時刻ｔ３まで利用可能である。第２の利用可能資源は、計算ノードＩＤ“４，５”の２つの計算ノードの時刻ｔ１以降の時間帯（上限なし）である（図中、この場合の利用可能時間を無限大の記号“∞”により表わす）。

【００８８】

利用可能時間は、構造体１３３のavailtimeに相当する。利用可能ノード数は、構造体１３３のnum__nidsに相当する。利用可能ノードＩＤは、構造体１３３のポインタnids__pで示される情報に相当する。時刻ｔ１における上記の第１の利用可能資源および第２の利用可能資源は、２つの資源情報によって管理される。各資源情報は、利用可能時間の短い順に連結される。

【００８９】

具体的には、時刻ｔ１の利用可能資源情報では、num__rscが“２”となる。ポインタarihead__pは、第１の利用可能資源に対応する第１の資源情報を示す。ポインタaritail__pは、第２の利用可能資源に対応する第２の資源情報を示す。

【００９０】
第１の資源情報は、次の情報を含む。ポインタnext__pは、第２の資源情報を示す。第１の資源情報には前の資源情報が存在しないため、ポインタprev__pはＮＵＬＬである。availtimeは“ｔ３－ｔ１”である。num__nidsは“１”である。ポインタnids__pはノードＩＤ“６”を示す。allocinfoは、第１の利用可能資源に対する仮割り当て作業域である。
【００９１】
第２の資源情報は、次の情報を含む。第２の資源情報には次の資源情報が存在しないため、ポインタnext__pはＮＵＬＬである。ポインタprev__pは、第１の資源情報を示す。availtimeは“∞”（無限大）である。num__nidsは“２”である。nids__pはノードＩＤ“４，５”を示す。allocinfoは、第２の利用可能資源に対する仮割り当て作業域である。
【００９２】
図１３は、時刻ｔ１に割り当てたジョブの管理例を示す図である。図１３では、仮にジョブＪ８を時刻ｔ１で実行開始するように割り当てる場合の仮割り当て作業情報を例示している。ここで、ジョブＪ８の計算資源の所要量は、計算ノード３つ、および、利用時間ｔ３－ｔ１であるとする。この場合、前述の時刻ｔ１における利用可能資源を用いてジョブＪ８の実行が可能である。時刻ｔ１における利用可能資源をジョブＪ８に仮割り当てする場合、ジョブＪ８の仮割り当て状況は、図１２で例示した第１の資源情報および第２の資源情報によって管理される。
【００９３】
具体的には、第１の資源情報において、仮割り当て作業域（仮割り当て作業情報）allocinfoは、次の情報を含む。total__alloc__nidsは“１”である。num__ajsは“１”である。ポインタaj__pは、ジョブＪ８に関する第１の仮割り当てジョブ情報を示す。
【００９４】
第１の仮割り当てジョブ情報は、次の情報を含む。仮割り当てされているジョブは１つ（ジョブＪ８のみ）なので、ポインタnext__pはＮＵＬＬである。jidは、仮割り当てされたジョブＪ８のジョブＩＤである（図１３では、これを“仮割り当てジョブＪ８のＩＤ”と表記している）。num__alloc__nidsは“１”である（仮割り当てした計算ノードが計算ノードＩＤ“６”の１つの計算ノードであるため）。
【００９５】
また、第２の資源情報において、仮割り当て作業域（仮割り当て作業情報）allocinfoは次の情報を含む。total__alloc__nidsは“２”である。num__ajsは“１”である。ポインタaj__pは、ジョブＪ８に関する第２の仮割り当てジョブ情報を示す。
【００９６】
第２の仮割り当てジョブ情報は、次の情報を含む。仮割り当てされているジョブは１つ（ジョブＪ８のみ）なので、ポインタnext__pはＮＵＬＬである。jidは、仮割り当てされたジョブＪ８のジョブＩＤである。num__alloc__nidsは“２”である（仮割り当てした計算ノードが計算ノードＩＤ“４，５”の２つの計算ノードであるため）。
【００９７】
図１４は、仮割り当てを管理する構造体の例を示す図である。仮割り当ての管理には、仮割り当て管理情報、仮割り当て情報、ジョブ割り当て情報およびジョブ割り当て資源情報が用いられる。仮割り当て管理情報、仮割り当て情報、ジョブ割り当て情報およびジョブ割り当て資源情報は、資源情報記憶部１３０に格納される。図１４では、仮割り当て管理情報の構造体１３６（joballocsummaryhead）、仮割り当て情報の構造体１３７（joballocsummary）、ジョブ割り当て情報の構造体１３８（joballoc）およびジョブ割り当て資源情報の構造体１３９（joballocrscinfo）の例を示す。
【００９８】
構造体１３６は、仮割り当て管理情報の構造体の例である。構造体１３６は、仮割り当て情報の数（num__joballocsum）および先頭仮割り当て情報へのポインタ（joballocsummary *head__p）を含む。仮割り当て管理情報では、各ジョブに関する仮割り当てパターン

を表わす仮割り当て情報が管理される。１つの仮割り当て情報が１つの仮割り当てパターンに相当する。仮割り当て情報の数は、該当の仮割り当て管理情報で管理される仮割り当て情報の数である。先頭仮割り当て情報へのポインタは、先頭の仮割り当て情報のＲＡＭ１０３上の格納アドレスを示す。仮割り当て管理情報は、構造体１３６の１つのインスタンスである。

【００９９】

構造体１３７は、仮割り当て情報の構造体の例である。構造体１３７は、次の仮割り当て情報（要素）へのポインタ（joballocsummary *next__p）、割り当てたジョブ数（num__jids）、割り当て開始時刻（timespec alloc__start）、割り当て完了時刻（timespec alloc__end）、先頭のジョブ割り当て情報へのポインタ（joballoc *head__p）、および、利用可能資源情報へのポインタ（availrsc *ar__p）を含む。

【０１００】

次の仮割り当て情報へのポインタは、次の仮割り当て情報のＲＡＭ１０３上の格納アドレスを示す。仮割り当て情報では、複数のジョブに対する計算資源の仮割り当てを管理し得る。割り当てたジョブ数は、該当の仮割り当て情報で管理されるジョブの数である。割り当て開始時刻は、仮割り当てされた複数のジョブのうちの最初のジョブに割り当てた時刻である（当該最初のジョブの実行開始時刻に相当する）。割り当て完了時刻は、当該複数のジョブのうちの最後のジョブに割り当てた時刻である（当該最後のジョブの実行開始時刻に相当する）。先頭のジョブ割り当て情報へのポインタは、仮割り当てされたジョブに関する先頭のジョブ割り当て情報のＲＡＭ１０３上の格納アドレスを示す。利用可能資源情報は、仮割り当てに用いた計算資源に関する利用可能資源情報である。仮割り当て情報は、構造体１３７の１つのインスタンスである。仮割り当て情報は、構造体１３７を用いて仮割り当てパターン毎に作成され得る。

【０１０１】

構造体１３８は、ジョブ割り当て情報の構造体の例である。構造体１３８は、次のジョブ割り当て情報へのポインタ（joballoc *next__p）、ジョブＩＤ（jid）およびジョブ割り当て資源情報（joballocrscinfo ari）を含む。

【０１０２】

次のジョブ割り当て情報へのポインタは、次のジョブ割り当て情報のＲＡＭ１０３上の格納アドレスを示す。ジョブＩＤは、仮割り当てされたジョブのジョブＩＤである。ジョブ割り当て資源情報は、ジョブＩＤで示されるジョブに対する計算資源の割り当て状況に関する情報である。ジョブ割り当て情報は、構造体１３８の１つのインスタンスである。ジョブ割り当て情報は、構造体１３８を用いて複数作成され得る。

【０１０３】

構造体１３９は、ジョブ割り当て資源情報の構造体の例である。構造体１３９は、資源割り当て開始時刻（timespec start）、資源割り当て終了時刻（timespec end）、割り当て資源数（num__nids）、割り当てノードＩＤへのポインタ（*nids__p）を含む。

【０１０４】

資源割り当て開始時刻は、ジョブ割り当て情報に含まれるジョブＩＤのジョブに対する計算資源の割り当て開始時刻である（当該ジョブの実行を当該開始時刻に開始する）。資源割り当て終了時刻は、ジョブ割り当て情報に含まれるジョブＩＤのジョブに対する計算資源の割り当て終了時刻である（当該ジョブの実行が当該終了時刻に完了する）。割り当て資源数は、割り当てられた計算ノードの数である。割り当てノードＩＤは、割り当てられた計算ノードの計算ノードＩＤである。ジョブ割り当て資源情報は、構造体１３９の１つのインスタンスである。ジョブ割り当て資源情報は、構造体１３９を用いて複数作成され得る。

【０１０５】

図１５は、時刻ｔ１を起点としたジョブの仮割り当ての例を示す図である。図１５では、時刻ｔ１を起点として、ジョブＡ，Ｂ，Ｃ，Ｄを仮割り当てするパターンに相当する仮割り当て情報の例を示している。

【０１０６】
仮割り当て情報は、次の情報を含む。ポインタnext__pは次の仮割り当て情報を示す（図１５では次の仮割り当て情報の図示を省略している）。例えば、各仮割り当て情報は、割り当て開始時刻（alloc__start）の早い順に連結される。num__jidsは“４”である。４つのジョブＡ，Ｂ，Ｃ，Ｄに対する仮割り当てを示すからである。alloc__startは“ｔ１”である。時刻ｔ１が仮割り当ての起点の時刻（割り当て開始時刻）だからである。alloc__endは“ｔ１＋ｔＡ”である。最後に実行開始されるジョブＤの実行開始時刻が時刻ｔ１＋ｔＡだからである。ポインタhead__pは、先頭のジョブ割り当て情報を示す。ポインタar__pは、仮割り当てに用いた利用可能資源情報を示す。
【０１０７】
図１６は、仮割り当てされた複数のジョブの管理例を示す図である。例えば、時刻ｔ１を起点としてジョブＡ，Ｂ，Ｃ，Ｄを仮割り当てしたパターンでは、それぞれに対する仮割り当ての内容を次のように管理する。この場合、ジョブＡ，Ｂ，Ｃ，Ｄそれぞれについて、構造体１３８を用いてジョブ割り当て情報が作成されることになる（計４つ）。各ジョブ割り当て情報は、ジョブの優先度の高い順に連結される。ただし、図１６では、ジョブＣ，Ｄそれぞれのジョブ割り当て情報の図示を省略している。
【０１０８】
前述のように、仮割り当て情報のポインタhead__pは、先頭のジョブ割り当て情報を示す。ジョブＡ，Ｂ，Ｃ，Ｄのうち、ジョブＡの優先度が最高なので、先頭のジョブ割り当て情報は、ジョブＡのジョブ割り当て情報である。ジョブＡのジョブ割り当て情報は、次の情報を含む。ポインタnext__pは次のジョブ割り当て情報を示す。ジョブＡの次に優先度が高いジョブはジョブＢである。このため、ジョブＡに対する次のジョブ割り当て情報は、ジョブＢのジョブ割り当て情報である。jidはジョブＡのジョブＩＤ（“Ａ”）である。
【０１０９】
更に、ジョブＡのジョブ割り当て資源情報は、次の情報を含む。startが“ｔ１”である。endが“ｔ１＋ｔＡ”である（ジョブＡの資源使用時間がｔＡであるため）。ここで、ｔ１＋ｔＡは、時刻ｔ１から時間ｔＡが経過した時刻を求める演算である（他の時刻についての演算も同様）。num__nidsは“２”である。計算ノードＩＤ“４，５”で示される２つの計算ノードがジョブＡに割り当てられたからである。ポインタnids__pは、計算ノードＩＤ“４，５”の情報を示す。
【０１１０】
また、ジョブＢのジョブ割り当て情報は、次の情報を含む。ポインタnext__pは次のジョブ割り当て情報を示す。ジョブＢの次に優先度が高いジョブはジョブＣである。このため、ジョブＢに対する次のジョブ割り当て情報は、ジョブＣのジョブ割り当て情報である。jidはジョブＢのジョブＩＤ（“Ｂ”）である。
【０１１１】
更に、ジョブＢのジョブ割り当て資源情報は、次の情報を含む。startが“ｔ１”である。endが“ｔ１＋ｔＢ”である（ジョブＢの資源使用時間がｔＢであるため）。num__nidsは“１”である。計算ノードＩＤ“６”で示される１つの計算ノードがジョブＢに割り当てられたからである。ポインタnids__pは、計算ノードＩＤ“６”の情報を示す。
【０１１２】
ジョブＣのジョブ割り当て情報もジョブＡ，Ｂそれぞれのジョブ割り当て情報と同様に、ジョブＣに対応する情報となる。ジョブＤのジョブ割り当て情報もジョブＡ，Ｂそれぞれのジョブ割り当て情報と同様に、ジョブＤに対応する情報となる。
【０１１３】
図１７は、実行待ちジョブ情報の例を示す図である。実行待ちジョブ情報１４１は、実行待ちジョブ記憶部１４０に格納される。実行待ちジョブ情報１４１は、ジョブＩＤ、実行開始時刻および利用資源の情報を含む。
【０１１４】

ジョブＩＤの項目には、ジョブＩＤが登録される。実行開始時刻の項目には、実行開始時刻が登録される。利用資源の項目には、計算ノードの計算ノードＩＤが登録される。例えば、実行待ちジョブ情報１４１には、ジョブＩＤが“Ａ”、実行開始時刻が“ｔ２”、利用資源が“３，４”という情報が登録される。これは、ジョブＡの実行開始時刻が時刻ｔ２であり、ジョブＡの実行に利用される計算ノードが計算ノードＩＤ“３，４”で示される２つの計算ノードであることを示す。実行待ちジョブ情報１４１には、ジョブＢ，Ｃ，Ｄそれぞれについても同様に、実行開始時刻や利用資源の情報が登録される。

【０１１５】

図１８は、パラメータ情報の例を示す図である。パラメータ情報１５１は、パラメータ記憶部１５０に予め格納される。パラメータ情報１５１は、ジョブのスケジューリングを行う際に用いられる各種の情報を含む。具体的には、パラメータ情報１５１は、評価対象とする後続対象ジョブ数、割り当て候補時刻の上限および割り当て採用基準の情報を含む。

【０１１６】

評価対象とする後続対象ジョブ数は、仮割り当ての対象の後続対象ジョブの数であり、例えば４である。割り当て候補時刻の上限は、最も早い割り当て時刻に対する割り当て候補時刻の上限であり、例えば時間Δｔである。時刻ｔ１が最も早い割り当て時刻である場合、割り当て候補時刻の上限は時刻ｔ１からΔｔ後の時刻（ｔ１＋Δｔ）ということになる。

【０１１７】

割り当て採用基準は、各ジョブに対する計算資源の複数の仮割り当てパターンの中から、実際に採用するパターンを選択するための基準である。例えば、割り当て採用基準は、（１）仮割り当てジョブ数が多い、（２）ジョブの実行完了が早い、（３）対象期間の空き資源量が少ない、という基準を含む。上記の３つには優先度が設けられている。（１）の採用基準は最も優先度が高い。（２）の採用基準は２番目に優先度が高い。（３）の採用基準は最も優先度が低い。採用基準は、上記の複数のうちの何れか１つでもよい。また、例示した採用基準以外の採用基準でもよい。

【０１１８】

図１９は、管理ノードのデータフローの例を示す図である。図１９では、図４で例示した各部と、各部による参照あるいは更新される情報との関係を例示している。具体的には、ジョブ受付部１６０は、割り当て待ちジョブ情報を追加する。割り当て対象ジョブ選択部１７１は、割り当て待ちジョブ情報から、割り当て対象ジョブ情報を抽出する。後続対象ジョブ選択部１７２は、パラメータ情報（評価対象とする後続対象ジョブ数）に基づいて、割り当て待ちジョブ情報から、後続対象ジョブ情報を抽出する。割り当て対象ジョブ選択部１７１は、後続対象ジョブ選択部１７２によって抽出された最後の後続対象ジョブ情報を終端のジョブの情報として特定する（割り当て対象ジョブ情報に終端ジョブ情報へのポインタを登録する）。

【０１１９】

割り当て候補時刻検出部１８１は、割り当て対象ジョブ情報、パラメータ情報（割り当て候補時刻の上限）および利用可能資源情報に基づいて、割り当て候補時刻および仮割り当て終端時刻を求める。割り当て候補時刻は、ジョブの割り当ての起点となる時刻の候補である。仮割り当て終端時刻は、最優先のジョブを最も早く割り当て可能な時刻から割り当て候補時刻の上限に達する時刻である。例えば、最優先のジョブを最も早く割り当て可能な時刻が時刻ｔ１の場合、仮割り当て終端時刻は、時刻ｔ１＋Δｔである。割り当て候補時刻検出部１８１は、時刻ｔ１から時刻ｔ１＋Δｔの間で割り当て候補時刻を選択することになる。

【０１２０】

仮資源割り当て部１８２は、割り当て対象ジョブ情報、後続対象ジョブ情報、割り当て候補時刻、仮割り当て終端時刻および利用可能資源情報（仮割り当て情報に含まれる）に基づいて、各ジョブに対する計算資源の仮割り当てを行い、仮割り当て情報を変更する。

ここで、仮資源割り当て部１８２は、仮割り当てに伴って、利用可能資源情報を仮割り当て情報へコピーする（図１９では、当該コピー処理を仮資源割り当て部１８２の処理として符号１８２ａにより図示している）。仮割り当てに伴って、本番の利用可能資源情報を直接更新しないようにするためである。

【０１２１】

割り当て資源選択部１８３は、仮割り当て情報およびパラメータ情報（割り当て採用基準）を参照して、複数の仮割り当てパターンの中から、採用する仮割り当てパターンを選択する。割り当て資源選択部１８３は、採用した仮割り当てパターンに応じて、利用可能資源情報および実行待ちジョブ情報を更新する（仮割り当てパターンの内容を実際のスケジュールに反映させる）。

【０１２２】

ジョブ実行指示部１９０は、実行待ちジョブ情報に基づいて、各計算ノードに対するジョブの実行を指示する。

次に、管理ノード１００による処理手順を説明する。なお、割り当て対象ジョブ選択部１７１および後続対象ジョブ選択部１７２は、割り当て対象ジョブ情報および後続対象ジョブ情報を作成済であるとする。

【０１２３】

図２０は、管理ノードの処理例を示す図である。以下、図２０に示す処理をステップ番号に沿って説明する。

（Ｓ１）割り当て候補時刻検出部１８１は、複数の割り当て候補時刻を検出する。また、割り当て候補時刻検出部１８１は、仮割り当て終端時刻を求める。前述のように、仮割り当て終端時刻は、最も早い割り当て候補時刻に、割り当て候補時刻の上限の時間Δｔを加えた時刻である。処理の詳細は後述される。

【０１２４】

（Ｓ２）仮資源割り当て部１８２は、割り当て候補時刻の先頭（最も早い時刻）から終端（最も遅い時刻）まで順に１つずつ選択して、ステップＳ３，４を繰り返し処理する。

（Ｓ３）仮資源割り当て部１８２は、選択した割り当て候補時刻に関する仮割り当て情報を作成し、資源情報記憶部１３０に格納する。また、仮資源割り当て部１８２は、仮割り当て処理に利用する利用可能資源情報をコピーして、仮割り当て情報に追加する。コピーされた利用可能資源情報は、資源情報記憶部１３０の所定の作業領域に格納される。以後、選択した割り当て候補時刻に関する仮割り当てに伴い、コピーされた利用可能資源情報が参照される（当該コピーされた利用可能資源情報が更新されることもある）。

【０１２５】

（Ｓ４）仮資源割り当て部１８２は、選択した割り当て候補時刻から仮割り当て終端時刻までの期間について仮資源割り当てを行い、仮割り当て情報に結果を追加する。処理の詳細は後述される。

【０１２６】

（Ｓ５）仮資源割り当て部１８２は、複数の割り当て候補時刻の全てについてステップＳ３，Ｓ４の処理を完了すると、処理をステップＳ６に進める。ステップＳ４を実行することで、１つの割り当て候補時刻に対して１つの仮割り当てパターンが作成されることになる。割り当て候補時刻毎にステップＳ４が繰り返し実行されることで、複数の仮割り当てパターンが作成される。

【０１２７】

（Ｓ６）割り当て資源選択部１８３は、作成された複数の仮割り当てパターンを比較し、１つの仮割り当てパターンを選択する。割り当て資源選択部１８３は、パラメータ情報１５１に含まれる割り当て採用基準に基づいて、仮割り当てパターンの選択を行う。

【０１２８】

（Ｓ７）仮資源割り当て部１８２は、仮資源割り当て用に確保した作業領域を初期化する。

図２１は、割り当て候補時刻検出の例を示すフローチャートである。以下、図２１に示

す処理をステップ番号に沿って説明する。以下の手順は、図２０のステップＳ１に相当する。

【０１２９】

（Ｓ１１）割り当て候補時刻検出部１８１は、スケジューリングの優先度順に並んでいる割り当て対象ジョブ情報および後続対象ジョブ情報（struct schedjobinfo.head__pのリスト）から先頭のジョブ（最高優先度のジョブ）を取得する。割り当て候補時刻検出部１８１は、先頭のジョブに対する利用可能資源数を０とする。

【０１３０】

（Ｓ１２）割り当て候補時刻検出部１８１は、ステップＳ１８までの手順により先頭のジョブを割り当て可能な最も早い時刻を検索する。そのために、割り当て候補時刻検出部１８１は、複数の利用可能資源情報（struct availrschead.head__pのリスト）の先頭の時刻から終端の時刻まで、１つずつ時刻（struct availrsc.time）を選択して、ステップＳ１３～Ｓ１７を繰り返し実行する。ここで選択された時刻をステップＳ１３～Ｓ１９において「特定時刻」と称する。

【０１３１】

（Ｓ１３）割り当て候補時刻検出部１８１は、特定時刻における複数の資源情報（struct availrsc.arihead__pのリスト）から、利用可能時間（struct availrscinfo.availtime）の長い順に１つずつ選択して、ステップＳ１４～Ｓ１６を繰り返し実行する。なお、struct availrsc.arihead__pのリストでは、各資源情報が利用可能時間（struct availrscinfo.availtime）の長い順に並んでいる。

【０１３２】

（Ｓ１４）割り当て候補時刻検出部１８１は、選択した資源情報の利用可能時間が、先頭のジョブの資源使用時間より長いか否かを判定する。長い場合、処理をステップＳ１５に進める。長くない場合、処理をステップＳ１８に進める。当該判定は、資源情報の利用可能時間（struct availrscinfo.availtime）とジョブ情報の資源使用時間（struct jobinfo.reqtime）とを比較することで行える。ステップＳ１４でＮｏの場合は、現在着目している利用可能資源情報から先頭のジョブの割り当て可能な時刻を見出すことはできないため、ステップＳ１８に進めて次の利用可能資源情報から割り当て可能な時刻を探索することになる。

【０１３３】

（Ｓ１５）割り当て候補時刻検出部１８１は、着目する計算資源を先頭のジョブの実行に利用可能と判断し、利用可能資源数としてカウントする（struct availrscinfo.num__nidsの値を利用可能資源数に加算する）。

【０１３４】

（Ｓ１６）割り当て候補時刻検出部１８１は、先頭のジョブに対する利用可能資源数のカウント値が、先頭のジョブの要求資源量以上であるか否かを判定する。要求資源量以上である場合、処理をステップＳ１９に進める。要求資源量以上でない場合、処理をステップＳ１７に進める。当該判定は、先頭のジョブに対する利用可能資源数と、ジョブ情報の要求資源量（struct jobinfo.num__reqnids）とを比較することで行える。

【０１３５】

（Ｓ１７）割り当て候補時刻検出部１８１は、特定時刻における全ての資源情報について処理を終了すると、処理をステップＳ１８に進める。処理をステップＳ１８に進める場合、割り当て候補時刻検出部１８１は、先頭のジョブに対する利用可能資源数を０にリセットする。処理をステップＳ１８に進める場合、現在着目している利用可能資源情報では先頭のジョブに対して割り当て可能な時刻を得られなかったことを意味する。

【０１３６】

（Ｓ１８）割り当て候補時刻検出部１８１は、通常、各利用可能資源情報を順番に参照しながら、ステップＳ１３～Ｓ１７を繰り返し実行することで、何れかのタイミングにおいてステップＳ１６によりループを抜けることになる。ただし、探索可能な全ての利用可能資源情報を処理しても、先頭のジョブに対する割り当て候補時刻を検出できない場合、

割り当て候補時刻検出部１８１は、エラー情報を出力して割り当て候補時刻の検出処理を終了してもよい。

【０１３７】

（Ｓ１９）割り当て候補時刻検出部１８１は、現在着目している特定時刻（struct availrsc.time）に対して先頭のジョブを割り当て可能と判断し、当該特定時刻を、割り当て可能時刻Ｔ１としてＲＡＭ１０３に保持する。第２の実施の形態の例では、割り当て可能時刻Ｔ１＝ｔ１である。なお、割り当て候補時刻検出部１８１は、先頭のジョブに対する利用可能資源数を０にリセットする。

【０１３８】

（Ｓ２０）割り当て候補時刻検出部１８１は、ステップＳ２５までの手順により、先頭のジョブを割り当て可能な、時刻Ｔ１よりも後の時刻を検索する。そのために、割り当て候補時刻検出部１８１は、複数の利用可能資源情報（struct availrschead.head__pのリスト）の時刻Ｔ１から割り当て候補時刻上限（＝仮割り当て終端時刻Ｔ１＋Δｔ）まで、１つずつ時刻（struct availrsc.time）を選択して、ステップＳ２１～Ｓ２４を繰り返し実行する。ここで選択された時刻をステップＳ２１～Ｓ２６において「特定時刻」と称する。

【０１３９】

（Ｓ２１）割り当て候補時刻検出部１８１は、特定時刻における複数の資源情報（struct availrsc.arihead__pのリスト）から、利用可能時間（availtime）の長い順に１つずつ選択して、ステップＳ２２～Ｓ２４を繰り返し実行する。なお、struct availrsc.arihead__pのリストでは、各資源情報が利用可能時間（struct availrscinfo.availtime）の長い順に並んでいる。

【０１４０】

（Ｓ２２）割り当て候補時刻検出部１８１は、選択した資源情報の利用可能時間が、先頭のジョブの資源使用時間より長いか否かを判定する。長い場合、処理をステップＳ２３に進める。長くない場合、処理をステップＳ２７に進める。当該判定は、ステップＳ１４と同様に行える。

【０１４１】

（Ｓ２３）割り当て候補時刻検出部１８１は、着目する計算資源を先頭のジョブの実行に利用可能と判断し、利用可能資源数としてカウントする（struct availrscinfo.num__nidsの値を利用可能資源数に加算する）。

【０１４２】

（Ｓ２４）割り当て候補時刻検出部１８１は、先頭のジョブに対する利用可能資源数のカウント値が、先頭のジョブの要求資源量以上であるか否かを判定する。要求資源量以上である場合、処理をステップＳ２６に進める。要求資源量以上でない場合、処理をステップＳ２５に進める。当該判定は、ステップＳ１６と同様に行える。

【０１４３】

（Ｓ２５）割り当て候補時刻検出部１８１は、特定時刻における全ての資源情報について処理を終了すると、処理をステップＳ２７に進める。処理をステップＳ２７に進める場合、割り当て候補時刻検出部１８１は、先頭のジョブに対する利用可能資源数を０にリセットする。処理をステップＳ２７に進める場合、現在着目している利用可能資源情報では先頭のジョブに対して割り当て可能な時刻を得られなかったことを意味する。

【０１４４】

（Ｓ２６）割り当て候補時刻検出部１８１は、着目する特定時刻（struct availrsc.time）に対して先頭のジョブを割り当て可能と判断し、当該特定時刻を、割り当て可能時刻としてＲＡＭ１０３に保持する。そして、処理をステップＳ２７に進める。

【０１４５】

（Ｓ２７）割り当て候補時刻検出部１８１は、割り当て可能時刻Ｔ１よりも後の時刻の全ての利用可能資源情報について処理を終了すると、割り当て候補時刻検出処理を終了する。この段階でＲＡＭ１０３に保持されている割り当て可能時刻が、割り当て候補時刻で

ある。例えば、割り当て候補時刻検出部１８１は、ステップＳ２０～Ｓ２７の手順により、割り当て可能時刻Ｔ１＝ｔ１に対し、時刻ｔ１よりも後の割り当て候補時刻として、時刻ｔ２，ｔ３，ｔ４を検出する。

【０１４６】

図２２は、仮資源割り当ての例を示すフローチャートである。以下、図２２に示す処理をステップ番号に沿って説明する。以下の手順は、図２０のステップＳ４に相当する。

（Ｓ３１）仮資源割り当て部１８２は、着目している割り当て候補時刻に対応する利用可能資源情報（指定された利用可能資源情報）を取得する。仮資源割り当て部１８２は、指定された利用可能資源情報を起点とする複数の利用可能資源情報のうち、開始時刻から終了時刻までの各時刻（struct availrsc.time）に対応する利用可能資源情報を１つずつ選択し、ステップＳ３２，Ｓ３３を順に処理する。開始時刻は、着目している割り当て候補時刻に相当する。終了時刻は、複数の利用可能資源情報に含まれる時刻（struct availrsc.time）のうち仮割り当て終端時刻を超過しない最も遅い時刻に相当する。ここで、ステップＳ３１で選択された利用可能資源情報の時刻をステップＳ３２，Ｓ３３の処理において「特定時刻」と称する。

【０１４７】

（Ｓ３２）仮資源割り当て部１８２は、特定時刻に対して割り当て可能なジョブ（仮割り当て対象ジョブ）を全て選択する。具体的には、仮資源割り当て部１８２は、割り当て対象ジョブ情報および後続対象ジョブ情報から特定時刻の利用可能資源情報を参照して、割り当て可能なジョブを選択し、仮割り当て対象ジョブとしてＲＡＭ１０３に保持する。処理の詳細は後述される。

【０１４８】

（Ｓ３３）仮資源割り当て部１８２は、仮割り当て対象ジョブと仮割り当て情報に基づいて、仮割り当て対象ジョブに対する計算資源の仮割り当て（仮割り当て対象ジョブのタイムマップへの配置）を行う。仮資源割り当て部１８２は、仮割り当ての結果を、仮割り当て情報に反映させる。処理の詳細は後述される。

【０１４９】

（Ｓ３４）仮資源割り当て部１８２は、開始時刻から終了時刻までの各時刻について仮割り当て対象ジョブの選択および仮割り当て対象ジョブの配置を実行すると、着目している割り当て候補時刻を起点とした仮資源割り当ての処理を終了する。

【０１５０】

図２３は、仮割り当て対象ジョブの選択例を示すフローチャートである。以下、図２３に示す処理をステップ番号に沿って説明する。以下の手順は、図２２のステップＳ３２に相当する。

【０１５１】

（Ｓ１０１）仮資源割り当て部１８２は、割り当て対象ジョブ情報および後続対象ジョブ情報を、優先度順に処理する。具体的には、仮資源割り当て部１８２は、割り当て対象ジョブ情報で示される複数の後続対象ジョブ情報から優先度の高い順に１つずつ選択して、ステップＳ１０２～Ｓ１１０を繰り返し実行する。なお、各後続対象ジョブ情報は、優先度の高い順に連結されている（struct schedjobinfo.head_pのリスト）。ここで、選択された後続対象ジョブ情報に対応するジョブを、ステップＳ１０２～Ｓ１１０において、「対象ジョブ」と称する。仮資源割り当て部１８２は、対象ジョブに対する仮割り当て資源数を０とする。

【０１５２】

（Ｓ１０２）仮資源割り当て部１８２は、特定時刻の利用可能時間の短い順に資源情報を処理する。具体的には、仮資源割り当て部１８２は、特定時刻における複数の資源情報のうち、利用可能時間の短い順（struct availrsc.aritail_pのリスト）に資源情報を１つずつ選択して、ステップＳ１０３～Ｓ１０７を繰り返し実行する。

【０１５３】

（Ｓ１０３）仮資源割り当て部１８２は、着目する資源情報の利用可能時間（struct a

vailrscinfo.availtime）が、着目する後続対象ジョブ情報の資源使用時間（struct jobinfo.reqtime）よりも長いかを判定する。利用可能時間の方が長い場合、処理をステップＳ１０４に進める。利用可能時間の方が長くない場合、処理をステップＳ１０８に進める。

【０１５４】

（Ｓ１０４）仮資源割り当て部１８２は、着目する資源情報において、仮割り当てに未利用の部分があるか否かを判定する。未利用の部分がある場合、処理をステップＳ１０５に進める。未利用の部分がない場合、処理をステップＳ１０８に進める。ここで、ステップＳ１０４の判定は、資源情報に含まれる仮割り当て作業情報の全仮割り当てノード数（struct allocrscinfo.total_alloc_nids）と、資源情報の利用可能ノード数（struct availrscinfo.num_nids）との比較により行える。全仮割り当てノード数の方が、利用可能ノード数よりも小さければ、全ての計算資源が仮割り当てに利用されているわけではなく、未利用の計算資源が残っていることになる。一方、全仮割り当てノード数が利用可能ノード数に一致していれば、全ての計算資源が仮割り当てに利用済であり、未利用の計算資源が残っていないことになる。

【０１５５】

（Ｓ１０５）仮資源割り当て部１８２は、着目する計算資源を対象ジョブへの仮割り当て資源数としてカウントする。具体的には、仮割り当て資源数に加算されるカウント数は、ｍｉｎ（利用可能資源数－仮割り当て済資源数，対象ジョブの要求資源数－カウント済の仮割り当て資源数）である。ここで、演算ｍｉｎ（ｘ，ｙ）は、ｘ，ｙのうち小さい方を選択する演算である。また、仮割り当て済資源数は、着目する資源情報のstruct allocrscinfo.total_alloc_nidsの値である。

【０１５６】

（Ｓ１０６）仮資源割り当て部１８２は、利用可能資源情報の仮割り当て作業域を更新する。具体的には、仮資源割り当て部１８２は、ステップＳ１０５でカウントした仮割り当て資源数を、着目する資源情報の仮割り当て作業域（struct availrscinfo.allocinfo）の全仮割り当てノード数（struct allocinfo.total_alloc_nids）へ加算する。これにより、対象ジョブに対して仮割り当てした分の計算資源が利用済として管理される。

【０１５７】

（Ｓ１０７）仮資源割り当て部１８２は、対象ジョブへの仮割り当て資源数が対象ジョブの要求資源数に一致するか否かを判定する。一致する場合、処理をステップＳ１０９に進める（ループを抜ける）。一致しない場合、処理をステップＳ１０８に進める。この判定は、ステップＳ１０５でカウントした仮割り当て資源数が、対象ジョブの要求資源数（struct jobinfo.num_reqnids）に達しているか否かを確認することで行われる。

【０１５８】

（Ｓ１０８）仮資源割り当て部１８２は、特定時刻における全ての資源情報について処理を終了すると、処理をステップＳ１１０に進める。

（Ｓ１０９）仮資源割り当て部１８２は、対象ジョブに関する情報を仮割り当て対象ジョブとして保持する。具体的には、仮資源割り当て部１８２は、対象ジョブに対応する仮割り当て対象ジョブ情報を、ＲＡＭ１０３上に保持する。仮割り当て対象ジョブ情報は、後続対象ジョブ情報と同様の情報を含む。ステップＳ１０９が繰り返し実行されることで、複数の仮割り当て対象ジョブを示す複数の仮割り当て対象ジョブ情報が保持され得る。このとき、仮資源割り当て部１８２は、ジョブ資源使用時間（struct jobinfo.reqtime）の長い順に仮割り当て対象ジョブ情報を並べておく。そして、処理をステップＳ１１０に進める。

【０１５９】

（Ｓ１１０）仮資源割り当て部１８２は、今回の対象ジョブが仮割り当て対象ジョブとして保存されたか否かを判定する。今回の対象ジョブが仮割り当て対象ジョブとして保存された場合、処理をステップＳ１１１に進める。今回の対象ジョブが仮割り当て対象ジョブとして保存されなかった場合、処理をステップＳ１１２に進める（ループを抜ける）。

対象ジョブが仮割り当て対象ジョブとして保存されていない場合に、次のジョブの仮割り当てを行わない（ステップＳ１１２に進める）理由は、スケジュール優先度の低いジョブが、スケジュール優先度の高いジョブよりも前に実行されないようにするためである。

【０１６０】

（Ｓ１１１）仮資源割り当て部１８２は、後続対象ジョブについて全て処理を完了すると、処理をステップＳ１１２に進める。

（Ｓ１１２）仮資源割り当て部１８２は、利用可能資源情報の仮割り当て作業域をクリアする。具体的には、仮資源割り当て部１８２は、特定時刻に対する全ての仮割り当て対象ジョブの保存が完了すると、着目する資源情報の仮割り当て作業域（struct availrscinfo.allocinfo）をクリアする。

【０１６１】

図２４は、仮割り当て対象ジョブの配置例を示すフローチャートである。以下、図２４に示す処理をステップ番号に沿って説明する。以下の手順は、図２２のステップＳ３３に相当する。

【０１６２】

（Ｓ１２１）仮資源割り当て部１８２は、特定時間に対応する利用可能資源情報を利用時間が短い資源から長い資源へ順に処理する。具体的には、仮資源割り当て部１８２は、利用可能資源情報で示される複数の資源情報から、利用可能時間が短い資源情報から長い資源情報へ順（struct availrsc.aritail_pのリスト）に、１つずつ選択し、ステップＳ１２２～Ｓ１３０を繰り返し実行する。利用可能時間が短い方から計算資源を利用することで、計算資源の空き時間を削減し得る。

【０１６３】

（Ｓ１２２）仮資源割り当て部１８２は、保持した仮割り当て対象ジョブを資源使用時間が長い順に処理する。具体的には、仮資源割り当て部１８２は、ＲＡＭ１０３上に保持した仮割り当て対象ジョブ情報を１つずつ選択し、ステップＳ１２３～Ｓ１２９を繰り返し実行する。ここで、選択された仮割り当て対象ジョブ情報に対応する仮割り当て対象ジョブを、ステップＳ１２３～Ｓ１２９において、「対象ジョブ」と称する。仮資源割り当て部１８２は、対象ジョブの仮割り当て資源数を０とする。

【０１６４】

（Ｓ１２３）仮資源割り当て部１８２は、着目する資源情報に含まれる利用可能時間（struct availrscinfo.availtime）が、対象ジョブの資源使用時間（struct jobinfo.reqtime）よりも長いか否かを判定する。利用可能時間の方が長い場合、処理をステップＳ１２４に進める。利用可能時間の方が長くない場合、処理をステップＳ１３０に進める。

【０１６５】

（Ｓ１２４）仮資源割り当て部１８２は、着目する計算資源を対象ジョブへの仮割り当て資源数としてカウントする。具体的には、仮割り当て資源数に加算されるカウント数は、ｍｉｎ（利用可能資源数－仮割り当て済資源数，対象ジョブの要求資源数－カウント済の仮割り当て資源数）である。仮割り当て済資源数は、着目する資源情報のstruct allocrscinfo.total_alloc_nidsの値である。

【０１６６】

（Ｓ１２５）仮資源割り当て部１８２は、利用可能資源情報の仮割り当て作業域を更新する。具体的には、仮資源割り当て部１８２は、ステップＳ１２４でカウントした仮割り当て資源数を、着目する資源情報の仮割り当て作業域（struct availrscinfo.allocinfo）の全仮割り当てノード数（struct allocrscinfo.total_alloc_nids）へ加算する。これにより、対象ジョブに対して仮割り当てした分の計算資源が利用済として管理される。更に、仮資源割り当て部１８２は、ステップＳ１３２以降の処理に備えて、当該計算資源を利用するジョブＩＤおよび使用資源量（獲得資源数）も合わせて仮割り当て作業域に反映する（構造体１３５の情報に相当）。

【０１６７】

（Ｓ１２６）仮資源割り当て部１８２は、対象ジョブへの仮割り当て資源数が対象ジョ

ブの要求資源数に一致するか否かを判定する。一致する場合、処理をステップＳ１２７に進める。一致しない場合、処理をステップＳ１３０に進める。この判定は、カウントした仮割り当て資源数が、対象ジョブの要求資源数（struct jobinfo.num__reqnids）に達しているか否かを確認することで行える。

【０１６８】

（Ｓ１２７）仮資源割り当て部１８２は、仮割り当て対象ジョブから、現在着目している対象ジョブ（該当ジョブ）を削除する。具体的には、仮資源割り当て部１８２は、ＲＡＭ１０３上に保持された複数の仮割り当て対象ジョブ情報の中から、現在着目している対象ジョブに対応する仮割り当て対象ジョブ情報を削除する。

【０１６９】

（Ｓ１２８）仮資源割り当て部１８２は、仮割り当て対象ジョブがあるか否かを判定する。仮割り当て対象ジョブがある場合、処理をステップＳ１２９に進める。仮割り当て対象ジョブがない場合、処理をステップＳ１３２に進める。具体的には、ＲＡＭ１０３上に保持した仮割り当て対象ジョブ情報が残っている場合、仮割り当て対象ジョブがあることになる。一方、ステップＳ１２７を繰り返し実行することでＲＡＭ１０３上に保持されていた複数の仮割り当て対象ジョブ情報が全て削除された場合、仮割り当て対象ジョブがないことになる。

【０１７０】

（Ｓ１２９）仮資源割り当て部１８２は、利用可能資源情報の仮割り当て作業域（struct availrscinfo.allocinfo）を確認し、着目する利用可能資源において、仮割り当てに未利用の部分があるか否かを判定する。未利用の部分がある場合、処理をステップＳ１３０に進める。未利用の部分がない、すなわち、仮割り当てで全て利用済である場合、処理をステップＳ１３１に進める。具体的には、着目する資源情報の仮割り当て作業域における全仮割り当てノード数（struct allocrscinfo.total__alloc__nids）が、資源情報の利用可能ノード数（struct availrscinfo.num__nids）よりも小さい場合は、仮割り当てに未利用の部分がある。一方、全仮割り当てノード数が利用可能ノード数に一致する場合には、該当の資源情報を全て仮割り当てに利用済である（次の資源情報の処理に移る）。

【０１７１】

（Ｓ１３０）仮資源割り当て部１８２は、次の仮割り当て対象ジョブの処理に移る（ステップＳ１２２に進む）。また、仮資源割り当て部１８２は、全ての仮割り当て対象ジョブを処理済の場合は、処理をステップＳ１３１に進める。

【０１７２】

（Ｓ１３１）仮資源割り当て部１８２は、次の資源情報の処理に移る（ステップＳ１２１に進む）。なお、仮資源割り当て部１８２は、仮割り当て対象ジョブが残っているにも関わらず、割り当て先を判断する資源情報がない場合には、エラーを出力して処理を終了してもよい。

【０１７３】

（Ｓ１３２）仮資源割り当て部１８２は、仮割り当て作業域の情報（仮割り当て作業情報）に基づいて、仮割り当て情報（struct joballocsummary）を更新する。

（Ｓ１３３）仮資源割り当て部１８２は、仮割り当て情報に基づいて、利用可能資源情報を更新する。具体的には、仮資源割り当て部１８２は、次の時刻以降の処理を行うために、ステップＳ１３２で更新した仮割り当て情報から利用可能資源情報を更新する。図２０のステップＳ３で述べたように更新は、コピーされた利用可能資源情報に対して行われる。仮資源割り当て部１８２は、仮割り当て対象ジョブの仮割り当て状況に応じて、新規時刻に対する利用可能資源情報を作成することもある。例えば、仮資源割り当て部１８２は、既存の資源情報について利用可能時間の前半部を仮割り当てに利用し、後半部が仮割り当てに未利用である場合、当該後半部の開始時刻に対して新規の利用可能資源情報を作成する。

【０１７４】

図２５は、割り当て資源選択の例を示すフローチャートである。以下、図２５に示す処

理をステップ番号に沿って説明する。以下の手順は、図２０のステップＳ６に相当する。
（Ｓ４１）割り当て資源選択部１８３は、先頭の仮割り当て情報を取得し、選択割り当て情報として保持する。具体的には、割り当て資源選択部１８３は、全ての仮割り当て情報を比較するため、まず先頭の仮割り当て情報（struct joballocsummaryhead.head_p）を取出し、選択割り当て情報としてＲＡＭ１０３上に保持する。
【０１７５】
（Ｓ４２）割り当て資源選択部１８３は、仮割り当て情報を順番に処理する。具体的には、割り当て資源選択部１８３は、先頭の仮割り当て情報を除く仮割り当て情報（struct joballocsummaryhead.head_pのリスト）を１つずつ選択し、ステップＳ４３～Ｓ４５を繰り返し実行する。ここで選択された仮割り当て情報を、以降のステップにおいて、「着目する仮割り当て情報」と称する。
【０１７６】
（Ｓ４３）割り当て資源選択部１８３は、選択割り当て情報と、着目する仮割り当て情報とを、パラメータ情報１５１に含まれる割り当て採用基準に基づいて比較する。
（Ｓ４４）割り当て資源選択部１８３は、採用基準に照らして、着目する仮割り当て情報の方が、選択割り当て情報よりも優先されるか否かを判定する。着目する仮割り当て情報の方が優先される場合、処理をステップＳ４５に進める。着目する仮割り当て情報の方が優先されない場合、処理をステップＳ４６に進める。例えば、パラメータ情報１５１の割り当て採用基準は、（１）仮割り当てできたジョブが多いもの、（２）ジョブの実行完了が早いもの、（３）対象期間の空き資源量が少ないもの、を優先させる旨を示す。すなわち、割り当て資源選択部１８３は、まず、仮割り当てできたジョブ数が多い方を優先させる。仮割り当てできたジョブ数が同数の場合は、全ジョブの実行がより早く完了する方を優先させる。全ジョブの実行完了が同じ時刻の場合は、対象期間（実行完了時刻までの期間）における空き資源量が少ない方を優先させる。なお、空き資源量は、当該期間における空き資源のノード時間積として表わせる。
【０１７７】
（Ｓ４５）割り当て資源選択部１８３は、着目する仮割り当て情報を選択割り当て情報として保持する。すなわち、割り当て資源選択部１８３は、現在の選択割り当て情報に代えて、着目する仮割り当て情報を、選択割り当て情報として保持する。
【０１７８】
（Ｓ４６）割り当て資源選択部１８３は、仮割り当て情報を全て処理すると、処理をステップＳ４７に進める。
（Ｓ４７）割り当て資源選択部１８３は、選択割り当て情報から利用可能資源情報を更新する。具体的には、割り当て資源選択部１８３は、最も優先すべき仮割り当てパターン（選択割り当て情報）を基に、利用可能資源情報（struct availrschead.head_p）を更新する。ここで、更新先の利用可能資源情報は、ステップＳ３でコピーされた利用可能資源情報ではなく、コピー元の利用可能資源情報である。すなわち、複数の仮割り当てパターンの中から確定された割り当てパターンの内容が、本番の利用可能資源情報に反映される。
【０１７９】
（Ｓ４８）割り当て資源選択部１８３は、実行待ちジョブ情報に割り当て結果を保存する。具体的には、確定された割り当てパターンに従って、実行待ちジョブ情報に、ジョブの実行スケジュールを追加する。
【０１８０】
このようにして、管理ノード１００は、仮割り当てパターンを複数作成し、複数の仮割り当てパターンの中から、採用基準に最もよく合致するものを選択する。なお、採用基準の所定期間は、割り当て候補時刻上限までの期間でもよいし、割り当て候補時刻上限までの期間とは異なる期間として予め指定することもできる。
【０１８１】
次に、管理ノード１００によるジョブＡ，Ｂ，Ｃ，Ｄに対する計算資源の割り当ての具

体例を説明する。
図２６は、利用可能資源情報およびジョブ情報の具体例を示す図である。まず、図５で例示したように、ジョブＪ１，Ｊ２，Ｊ３，Ｊ４，Ｊ５，Ｊ６，Ｊ７が既にスケジューリング済であるとする。また、今回割り当てを行うジョブは、ジョブＡ，Ｂ，Ｃ，Ｄである。このとき、利用可能資源情報１３２ａは、次の情報を含む。
【０１８２】
時刻ｔ０では、利用可能な計算資源はない。
時刻ｔ１では、２種類の計算資源を利用可能である。１つ目は、計算ノードＩＤ“６”の１つの計算ノードの、時刻ｔ１から時間ｔ３－ｔ１の時間帯である。２つ目は、計算ノードＩＤ“４”，“５”の２つの計算ノードの、時刻ｔ１からの上限なしの時間帯である。
【０１８３】
時刻ｔ２では、３種類の計算資源を利用可能である。１つ目は、計算ノードＩＤ“６”，“７”の２つの計算ノードの、時刻ｔ２から時間ｔ３－ｔ２の時間帯である。２つ目は、計算ノードＩＤ“３”の１つの計算ノードの、時刻ｔ２から時間ｔ４－ｔ２の時間帯である。３つ目は、計算ノードＩＤ“４”，“５”の２つの計算ノードの、時刻ｔ２からの上限なしの時間帯である。
【０１８４】
利用可能資源情報１３２ａは、時刻ｔ３，ｔ４，ｔ５についても同様に利用可能な計算資源の情報を含む。また、割り当て対象ジョブ情報１２１ａは、先頭のジョブ情報としてジョブＡのジョブ情報（後続対象ジョブ情報）を示すポインタを含む。ジョブＡは、要求資源量が“２”であり、資源使用時間が“ｔＡ”である。割り当て対象ジョブ情報１２１ａは、終端のジョブ情報として、ジョブＤの情報（後続対象ジョブ情報）を示すポインタを含む（図２６では図示を省略している）。
【０１８５】
後続対象ジョブ情報１２２ａは、各ジョブの情報を含む。各ジョブの情報は、優先度の高い順にポインタにより連結されている。後続対象ジョブ情報１２２ａは、ジョブＢ，Ｃ，Ｄの情報を含む。ジョブＢは、要求資源量が“１”であり、資源使用時間が“ｔＢ”である。ジョブＢは、ジョブＡの次に優先度の高いジョブである。ジョブＣは、要求資源量が“１”であり、資源使用時間が“ｔＣ”である。ジョブＣは、ジョブＢの次に優先度の高いジョブである。ジョブＤは、要求資源量が“１”であり、資源使用時間が“ｔＤ”である。ジョブＤは、ジョブＣの次に優先度の高いジョブである。
【０１８６】
図２７は、割り当て候補時刻の検出例を示す図である。割り当て候補時刻検出部１８１は、ジョブＡ，Ｂ，Ｃ，Ｄのうち、最も優先度の高いジョブＡにより、割り当て候補時刻を検出する。例えば、時刻ｔ０は、利用可能な計算資源が存在しないので、ジョブＡの割り当て候補時刻とはなり得ない。
【０１８７】
時刻ｔ１には、利用可能ノード数“２”、かつ、利用可能時間の制限のない（“∞”）計算資源が存在する。ｔＡ＜∞である。この計算資源は、ジョブＡの要求資源量“２”、かつ、資源使用時間“ｔＡ”の要件を満たす。よって、時刻ｔ１は、ジョブＡを割り当て可能な時刻であり、割り当て候補時刻である。
【０１８８】
時刻ｔ２には、利用可能ノード数“２”、かつ、利用可能時間“ｔ３－ｔ２”の計算資源が存在する。しかし、ｔＡ＞ｔ３－ｔ２である。すなわち、当該計算資源は、資源使用時間“ｔＡ”の要件を満たさないので、当該計算資源にジョブＡを割り当てることはできない。また、時刻ｔ２には、利用可能ノード数“２”、かつ、利用可能時間の制限のない計算資源が存在する。この計算資源は、ジョブＡの要求資源量“２”、かつ、資源使用時間“ｔＡ”の要件を満たす。よって、時刻ｔ２は、ジョブＡを割り当て可能な時刻であり、割り当て候補時刻である。

【０１８９】
こうして、割り当て候補時刻検出部１８１は、割り当て候補時刻を順次検出する。ここで、割り当て候補時刻検出部１８１は、未来の時刻を無制限に選択して割り当て候補時刻を検出するわけではない。ジョブＡの実行をある程度以上遅延させないように、また、スケジューリングを効率的に行えるように、割り当て候補時刻に上限を設ける。
【０１９０】
図２８は、割り当て候補時刻の上限の時刻の例を示す図である。パラメータ情報１５１は、割り当て候補時刻の上限Δｔを含む。割り当て候補時刻検出部１８１は、最も優先度の高いジョブＡに対して決定した、最も早い割り当て候補時刻ｔ１とΔｔとを用いて、割り当て候補時刻の上限となる時刻（割り当て終端時刻）を算出する。具体的には、割り当て候補時刻の上限の時刻は、ｔ１＋Δｔである。本例の場合、時刻ｔ１＋Δｔは、時刻ｔ４よりも遅く、時刻ｔ５よりも早い時刻である。このため、割り当て候補時刻検出部１８１は、時刻ｔ１，ｔ２，ｔ３，ｔ４を割り当て候補時刻として検出する。時刻ｔ５は、上限の時刻ｔ１＋Δｔよりも遅い時刻であるため、割り当て候補時刻からは除外される。
【０１９１】
図２９は、仮割り当て対象ジョブの選択例を示す図である。仮資源割り当て部１８２は、１つの割り当て候補時刻に着目して、今回の割り当て対象であるジョブＡ，Ｂ，Ｃ，Ｄの中から、割り当て候補時刻を起点とした上限の時刻（ｔ１＋Δｔ）までの時刻範囲に含まれる各時刻に対して仮割り当て対象ジョブを選択する。例えば、割り当て候補時刻ｔ１に着目する場合、仮資源割り当て部１８２は、まず、時刻ｔ１に対する仮割り当て対象ジョブを選択する。仮割り当て対象ジョブは、各ジョブの優先度の順番に従って選択される。
【０１９２】
より具体的には、仮資源割り当て部１８２は、利用可能資源情報１３２ａにおける時刻ｔ１の資源情報を参照して、ジョブＡ，Ｂ，Ｃ，Ｄの順に仮割り当て対象ジョブの選択を行う。まず、ジョブＡには、時刻ｔ１において、計算ノードＩＤ“４”，“５”の２つの計算ノードを割り当てることができる。このため、ジョブＡは、時刻ｔ１における仮割り当て対象ジョブである。
【０１９３】
次に、ジョブＢには、時刻ｔ１において、計算ノードＩＤ“６”の１つの計算ノードを割り当てることができる。ジョブＢの資源使用時間ｔＢが、ｔＢ＜ｔ３－ｔ１だからである。このため、ジョブＢは、時刻ｔ１における仮割り当て対象ジョブである。
【０１９４】
この場合、時刻ｔ１では、利用可能な全ての計算ノードがジョブＡ，Ｂに対して選択されてしまっている。このため、ジョブＣ以降のジョブに対して割り当て可能な計算ノードが存在しない。したがって、ジョブＣ，Ｄは、時刻ｔ１における仮割り当て対象ジョブとはならない。
【０１９５】
図３０は、仮割り当て対象ジョブの選択例（続き）を示す図である。図２９の処理を、図３０で例示するように説明することもできる。まず、仮資源割り当て部１８２は、利用可能資源情報１３２ａに基づいて、ジョブＡに対して、時刻ｔ１において計算ノードＩＤ“４，５”の計算ノードを割り当て可能と判断する。仮資源割り当て部１８２は、ジョブＡを、時刻ｔ１に対する仮割り当て対象ジョブと決定する。
【０１９６】
次に、仮資源割り当て部１８２は、利用可能資源情報１３２ａに基づいて、ジョブＢに対して、時刻ｔ１において計算ノードＩＤ“６”の計算ノードを割り当て可能と判断する。仮資源割り当て部１８２は、ジョブＢを、時刻ｔ１に対する仮割り当て対象ジョブと決定する。
【０１９７】
この時点で、時刻ｔ１において他のジョブに割り当て可能な計算ノードは存在しない。

したがって、仮資源割り当て部１８２は、ジョブＣ，Ｄを、時刻ｔ１における仮割り当て対象ジョブとして選択しない。よって、この場合、仮割り当て対象ジョブとして選択されるジョブは、ジョブＡ，Ｂである。すると、仮資源割り当て部１８２は、仮割り当て対象ジョブＡ，Ｂに対する時刻ｔ１の計算資源への仮割り当てを実行する。

【０１９８】

図３１は、仮割り当ての例を示す図である。仮資源割り当て部１８２は、仮割り当て処理の際、利用可能な計算資源の利用可能時間が短い順に、割り当て可能なジョブを探索する。ここでは、仮資源割り当て部１８２は、時刻ｔ１に対して特定した仮割り当て対象ジョブＡ，Ｂの中からジョブを探索することになる。このとき、仮資源割り当て部１８２は、仮割り当て対象ジョブのうち、資源使用時間が長いジョブを優先する。

【０１９９】

より具体的には、利用可能資源情報１３２ａでは、時刻ｔ１において利用可能な計算資源として、利用可能時間ｔ３－ｔ１および利用可能時間∞の２種類の資源情報を含む。仮資源割り当て部１８２は、このうち、利用可能時間が短い方（利用可能時間ｔ３－ｔ１）の計算資源を割り当てるジョブを検索する。ジョブの検索順序は、前述のように資源使用時間が長い順（ｔＡ＞ｔＢなのでＡ，Ｂの順）である。この場合、ジョブＡの資源使用時間“ｔＡ”は、ｔＡ＞ｔ３－ｔ１である。よって、ジョブＡを割り当てることはできない。次に、ジョブＢの資源使用時間“ｔＢ”は、ｔＢ＜ｔ３－ｔ１である。よって、ジョブＢを割り当てることができる。こうして、仮資源割り当て部１８２は、仮割り当て対象ジョブであるジョブＢに対し、時刻ｔ１における計算ノードＩＤ“６”の計算ノードを割り当てる。次に、同様にして、仮資源割り当て部１８２は、仮割り当て対象ジョブであるジョブＡに対して、時刻ｔ１における計算ノードＩＤ“４”，“５”の計算ノードを割り当てる。

【０２００】

図３２は、仮割り当ての例（続き）を示す図である。図３１の処理を、図３２で例示するように説明することもできる。まず、仮資源割り当て部１８２は、時刻ｔ１において計算ノードＩＤ“６”の計算ノードに割り当てるジョブを、仮割り当て対象ジョブＡ，Ｂの中から検索する。仮割り当て対象ジョブＡ，Ｂでは、ジョブＡの方が、ジョブＢよりも使用資源時間が長い（ｔＡ＞ｔＢ）。よって、仮資源割り当て部１８２は、時刻ｔ１における計算ノードＩＤ“６”の計算ノードに対するジョブＡの割り当てを試みるが、ジョブＡの使用資源時間の要求を満たさないため、ジョブＡを割り当てることはできない。次に、仮資源割り当て部１８２は、時刻ｔ１における計算ノードＩＤ“６”の計算ノードに対するジョブＢの割り当てを試みて、ジョブＢの使用資源時間の要求を満たすと判断し、ジョブＢに当該計算ノードを割り当てる。続いて、仮資源割り当て部１８２は、時刻ｔ１において計算ノードＩＤ“４”，“５”の計算ノードに、仮割り当て対象ジョブＡを割り当てる（残った仮割り当て対象ジョブはジョブＡのみである）。

【０２０１】

こうして、仮資源割り当て部１８２は、時刻ｔ１での仮割り当て対象ジョブに対する計算資源の仮割り当てを完了する。そして、仮資源割り当て部１８２は、仮割り当ての結果に応じて、利用可能資源情報１３２ａを更新する。

【０２０２】

図３３は、更新後の利用可能資源情報の例を示す図である。仮資源割り当て部１８２は、時刻ｔ１に対して、ジョブＡ，Ｂを割り当てた結果を、利用可能資源情報１３２ａに反映させる。具体的には、仮資源割り当て部１８２は、時刻ｔ１の資源情報のうち、計算ノードＩＤ“６”の計算ノードを、ジョブＢへ割り当てた旨を追加する。また、仮資源割り当て部１８２は、時刻ｔ１の資源情報のうち、計算ノードＩＤ“４”，“５”の２つの計算ノードを、ジョブＡへ割り当てた旨を追加する。

【０２０３】

更に、仮資源割り当て部１８２は、時刻ｔ１＋ｔＢに対する情報を利用可能資源情報１３２ａに追加する（構造体１３２および構造体１３２に含まれる構造体１３３のインスタ

ンスが追加されることになる）。具体的には、時刻ｔ１＋ｔＢにおいて、利用可能ノード数が“２”、利用可能時間が“ｔ３－ｔ１－ｔＢ”、ノードＩＤが“６”，“７”という資源情報を追加する。また、時刻ｔ１＋ｔＢにおいて、利用可能ノード数が“１”、利用可能時間が“ｔ４－ｔ１－ｔＢ”、ノードＩＤが“３”という資源情報を追加する。利用可能資源情報１３２ｂは、当該資源情報が利用可能資源情報１３２ａに追加された後を例示している。

【０２０４】

同様に、仮資源割り当て部１８２は、時刻ｔ１＋ｔＡに対する情報も利用可能資源情報１３２ａに追加する（ただし、図３３では図示を省略している）。

こうして、仮資源割り当て部１８２は、更新後の利用可能資源情報１３２ｂに基づいて、次の時刻に対する仮割り当て対象ジョブを、残りのジョブＣ，Ｄの中から選択し、選択したジョブへの計算資源の仮割り当てを実行する。仮資源割り当て部１８２は、上記の処理を繰り返し実行することで、割り当て候補時刻毎のジョブの仮割り当てパターンを得る。

【０２０５】

図３４は、仮割り当てパターンの例を示す図である。図３４（Ａ）は、割り当て候補時刻ｔ１に対する仮割り当てパターンＰｔ１を例示している。具体的には、仮資源割り当て部１８２は、図３２，３３で例示した方法によりジョブＡ，Ｂに対する時刻ｔ１の計算資源の割り当てを終えた後、ジョブＣに対し、時刻ｔ２の計算資源（計算ノードＩＤ“７”）を割り当てる。また、仮資源割り当て部１８２は、ジョブＤに対し、時刻ｔ１＋ｔＡの計算資源（計算ノードＩＤ“４”）を割り当てる。こうして、仮資源割り当て部１８２は、仮割り当てパターンＰｔ１を得る。なお、ジョブＤに対しては、時刻ｔ１＋ｔＡの計算資源のうち、計算ノードＩＤ“４”および計算ノードＩＤ“５”の何れか一方の計算ノードを割り当てる。計算ノードＩＤ“４”，“５”の両方で利用可能な計算資源が同じである場合、例えば、計算ノードＩＤの小さい方を優先的に割り当てる（以下の場合も同様）。

【０２０６】

図３４（Ｂ）は、割り当て候補時刻ｔ２に対する仮割り当てパターンＰｔ２を例示している。割り当て候補時刻ｔ２の場合、仮資源割り当て部１８２は、時刻ｔ２における仮割り当て対象ジョブとして、ジョブＡ，Ｂ，Ｃ，Ｄを選択する。そして、仮資源割り当て部１８２は、計算ノードＩＤ“６”の計算ノードをジョブＢに割り当てる。仮資源割り当て部１８２は、計算ノードＩＤ“７”の計算ノードをジョブＣに割り当てる。仮資源割り当て部１８２は、計算ノードＩＤ“３”，“４”の２つの計算ノードをジョブＡに割り当てる。仮資源割り当て部１８２は、計算ノードＩＤ“５”の計算ノードをジョブＤに割り当てる。

【０２０７】

図３４（Ｃ）は、割り当て候補時刻ｔ３に対する仮割り当てパターンＰｔ３を例示している。割り当て候補時刻ｔ３の場合、仮資源割り当て部１８２は、時刻ｔ３における仮割り当て対象ジョブとしてジョブＡを選択する。そして、仮資源割り当て部１８２は、計算ノード“４”，“５”の２つの計算ノードをジョブＡに割り当てる。時刻ｔ３では、計算ノード“３”も利用可能であるが、時間ｔ４－ｔ３がジョブＢの資源利用時間ｔＢよりも短い（ｔＢ＜ｔ４－ｔ３）ため、時刻ｔ３にはジョブＡ以外のジョブを割り当てることはできない。また、時刻ｔ３＋ｔＡは、時刻ｔ１＋Δｔよりも遅い時刻である。したがって、仮資源割り当て部１８２は、ジョブＡ以外のジョブの仮割り当てを行わない（仮割り当てパターンＰｔ３にはジョブＡ以外は配置されていない）。

【０２０８】

割り当て候補時刻ｔ４に対しても、同様に仮割り当てパターンが作成される。割り当て候補時刻ｔ４に対する仮割り当てパターンは、割り当て候補時刻ｔ４を起点にジョブＡが配置されるパターンとなる（図示を省略している）。また、時刻ｔ４＋ｔＡは、時刻ｔ１＋Δｔよりも遅い時刻となるので、割り当て候補時刻ｔ３の場合と同様に、ジョブＡ以外

のジョブの仮割り当ては行われない（割り当て候補時刻ｔ４に対する仮割り当てパターンに、ジョブＡ以外は配置されない）。

【０２０９】

図３５は、仮割り当て情報の例（パターンＰｔ１）を示す図である。仮割り当て情報１３７ａおよびジョブ割り当て情報群１３８ａは、仮割り当てパターンＰｔ１を示す情報の具体例である。

【０２１０】

仮割り当て情報１３７ａは、次の情報を含む。割り当て開始時刻は“ｔ１”である。割り当て完了時刻は“ｔ１＋ｔＡ”である（最後のジョブＤの実行を開始する時刻に相当）。割り当てたジョブ数は“４”である。ジョブＡ，Ｂ，Ｃ，Ｄの計４つのジョブの仮割り当てを行った結果だからである。また、仮割り当て情報１３７ａは、ジョブ割り当て情報群１３８ａの先頭のジョブ割り当て情報（ジョブＡの情報）を示すポインタを含む。ジョブ割り当て情報群１３８ａは、ジョブＡ，Ｂ，Ｃ，Ｄそれぞれに対するジョブ割り当て情報を含む。各ジョブ割り当て情報は、優先度の順にポインタにより連結されている。

【０２１１】

ジョブＡに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ１”である。割り当て終了時刻は、“ｔ１＋ｔＡ”である。割り当て資源数は、“２”である。割り当て資源（計算ノードＩＤ）は、“４，５”である。

【０２１２】

ジョブＢに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ１”である。割り当て終了時刻は、“ｔ１＋ｔＢ”である。割り当て資源数は、“１”である。割り当て資源（計算ノードＩＤ）は、“６”である。

【０２１３】

ジョブＣに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ２”である。割り当て終了時刻は、“ｔ２＋ｔＣ”である。割り当て資源数は、“１”である。割り当て資源（計算ノードＩＤ）は、“７”である。

【０２１４】

ジョブＤに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ１＋ｔＡ”である。割り当て終了時刻は、“ｔ１＋ｔＡ＋ｔＤ”である。割り当て資源数は、“１”である。割り当て資源（計算ノードＩＤ）は、“５”である。

【０２１５】

図３６は、仮割り当て情報の例（パターンＰｔ２）を示す図である。仮割り当て情報１３７ｂおよびジョブ割り当て情報群１３８ｂは、仮割り当てパターンＰｔ２を示す情報の具体例である。

【０２１６】

仮割り当て情報１３７ｂは、次の情報を含む。割り当て開始時刻は“ｔ２”である。割り当て完了時刻は“ｔ２”である。仮割り当てパターンＰｔ２の場合、ジョブＡ，Ｂ，Ｃ，Ｄの全てのジョブが時刻ｔ２に割り当てられるため、割り当て開始時刻および割り当て完了時刻は、共に“ｔ２”となる。割り当てたジョブ数は“４”である。ジョブＡ，Ｂ，Ｃ，Ｄの計４つのジョブの仮割り当てを行った結果だからである。また、仮割り当て情報１３７ｂは、ジョブ割り当て情報群１３８ｂの先頭のジョブ割り当て情報（ジョブＡの情報）を示すポインタを含む。ジョブ割り当て情報１３７ｂは、ジョブＡ，Ｂ，Ｃ，Ｄそれぞれに対するジョブ割り当て情報を含む。各ジョブ割り当て情報は、優先度の順にポインタにより連結されている。

【０２１７】

ジョブＡに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ２”である。割り当て終了時刻は、“ｔ２＋ｔＡ”である。割り当て資源数は、“２”である。割り当て資源（計算ノードＩＤ）は、“３，４”である。

【０２１８】

ジョブＢに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ

２”である。割り当て終了時刻は、“ｔ２＋ｔＢ”である。割り当て資源数は、“１”である。割り当て資源（計算ノードＩＤ）は、“６”である。

【０２１９】

ジョブＣに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ２”である。割り当て終了時刻は、“ｔ２＋ｔＣ”である。割り当て資源数は、“１”である。割り当て資源（計算ノードＩＤ）は、“７”である。

【０２２０】

ジョブＤに関するジョブ割り当て情報は、次の情報を含む。割り当て開始時刻は、“ｔ２”である。割り当て終了時刻は、“ｔ２＋ｔＤ”である。割り当て資源数は、“１”である。割り当て資源（計算ノードＩＤ）は、“５”である。

【０２２１】

割り当て候補時刻ｔ３，ｔ４それぞれに対する仮割り当てパターンについても、仮割り当てパターンＰｔ１，Ｐｔ２と同様に管理できる。割り当て資源選択部１８３は、上記の各仮割り当てパターンの情報を比較することで、各仮割り当てパターンにおける所定時刻範囲内のジョブの実行数、実行完了時刻および計算ノードの空き時間の割合などを比較可能である。

【０２２２】

割り当て資源選択部１８３は、パラメータ情報１５１に含まれる割り当て採用基準の情報を基に、複数の仮割り当てパターンの中から、最優先の仮割り当てパターンを選択する。例えば、仮割り当てジョブ数が多いという基準に照らせば、候補時刻ｔ１，ｔ２では仮割り当てジョブ数が“４”であるのに対し、候補時刻ｔ３，ｔ４では仮割り当てジョブ数が“１”である。このため、候補時刻ｔ３，ｔ４に対する２つの仮割り当てパターンは、候補時刻ｔ１，ｔ２に対する仮割り当てパターンＰｔ１，Ｐｔ２に比べて優先度が低い。よって、割り当て資源選択部１８３は、候補時刻ｔ３，ｔ４に対する２つの仮割り当てパターンを選択の候補から除外する。

【０２２３】

次に、割り当て資源選択部１８３は、割り当て採用基準によって、仮割り当てパターンＰｔ１，Ｐｔ２を比較する。仮割り当てパターンＰｔ１，Ｐｔ２は、両方とも仮割り当てジョブ数が“４”で一致している。このため、割り当て資源選択部１８３は、ジョブの実行完了が早いという基準により仮割り当てパターンＰｔ１，Ｐｔ２の何れか一方を選択する。ここで、仮割り当てパターンＰｔ２は、ジョブＡ，Ｂ，Ｃ，Ｄの全てのジョブの実行が完了する時刻が、仮割り当てパターンＰｔ１よりも早い。

【０２２４】

具体的には、仮割り当てパターンＰｔ１，Ｐｔ２の何れも、最後に実行完了するジョブはジョブＤである。ジョブ割り当て情報群１３８ａによれば、仮割り当てパターンＰｔ１において、ジョブＤの割り当て終了時刻は、“ｔ１＋ｔＡ＋ｔＤ”である。また、ジョブ割り当て情報群１３８ｂによれば、仮割り当てパターンＰｔ２において、ジョブＤの割り当て終了時刻は、“ｔ２＋ｔＤ”である。時刻ｔ２＋ｔＤは、時刻ｔ１＋ｔＡ＋ｔＤよりも早い時刻である。したがって、割り当て資源選択部１８３は、仮割り当てパターンＰｔ１よりも仮割り当てパターンＰｔ２の方が選択の優先度が高いと判断する。

【０２２５】

最終的に、割り当て資源選択部１８３は、ジョブＡ，Ｂ，Ｃ，Ｄに対する計算資源の割り当てとして、仮割り当てパターンＰｔ２を採用する。割り当て資源選択部１８３は、選択した仮割り当てパターンＰｔ２の内容を、実行待ちジョブ情報１４１に反映させる。こうして、選択した仮割り当てパターンＰｔ２に対応するスケジュールに従って、ジョブＡ，Ｂ，Ｃ，Ｄが複数の計算ノードにより実行されることになる。

【０２２６】

管理ノード１００によれば、ジョブの実行を効率化できる。例えば、優先順位がつけられた各ジョブを実行できる最も早い時刻に、優先順位に従って順次実行させていく方法も考えられる。しかし、単に優先度に従って最も早い時刻にジョブを割り当てるのみでは、

各ジョブに対して必ずしも効率的に計算資源を割り当てられているとは限らない。全てのジョブを完了するまでの時間が長くなることもある。

【０２２７】

そこで、管理ノード１００は、最も優先度の高いジョブＡのスケジューリングの際に、後続のジョブＢ，Ｃ，Ｄも考慮する。具体的には、管理ノード１００は、ジョブＡ，Ｂ，Ｃ，Ｄのスケジューリングの候補となる仮割り当てパターンを複数取得し、当該複数のパターンの比較により、パラメータ情報１５１に従い、所定期間内により多くのジョブを実行できるパターンを採用する。管理ノード１００は、採用した仮割り当てパターンに従って、各計算ノードへのジョブの割り当てを行う。例えば、仮割り当てパターンＰｔ２では、ジョブＡ，Ｂ，Ｃ，Ｄの全ての実行が完了するまでの時間を、仮割り当てパターンＰｔ１よりも早めることができる。また、仮割り当てパターンＰｔ２では仮割り当てパターンＰｔ１よりも計算ノードの空き時間を低減できる。このため、ジョブの優先度を維持したままジョブ実行のスループットを向上できる。こうして、ジョブの実行を効率化することができる。

【０２２８】

なお、第１の実施の形態の情報処理は、演算部１ｂにプログラムを実行させることで実現できる。また、第２の実施の形態の情報処理は、ＣＰＵ１０１，１０２にプログラムを実行させることで実現できる。プログラムは、コンピュータ読み取り可能な記録媒体１３に記録できる。

【０２２９】

例えば、プログラムを記録した記録媒体１３を配布することで、プログラムを流通させることができる。また、プログラムを他のコンピュータに格納しておき、ネットワーク経由でプログラムを配布してもよい。コンピュータは、例えば、記録媒体１３に記録されたプログラムまたは他のコンピュータから受信したプログラムを、ＲＡＭ１０３やＨＤＤ１０４などの記憶装置に格納し（インストールし）、当該記憶装置からプログラムを読み込んで実行してもよい。

【０２３０】

以上の第１～第２の実施の形態を含む実施形態に関し、更に以下の付記を開示する。

（付記１）　複数の計算ノードの利用可能な時間帯の情報を記憶する記憶部と、

前記情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる演算部と、

を有する情報処理装置。

【０２３１】

（付記２）　前記演算部は、前記第１の時刻および前記第２の時刻それぞれで前記最優先のジョブを実行開始する場合の前記複数のジョブの前記複数の計算ノードへの割り当てパターンの候補を示す情報を生成し、前記割り当てパターンの候補の比較に応じて、前記最優先のジョブを前記第１および前記第２の時刻の何れで実行するかを選択する、付記１記載の情報処理装置。

【０２３２】

（付記３）　前記演算部は、前記割り当てパターンの候補に基づいて前記複数のジョブ全ての実行が完了する時刻を比較し、当該比較に応じて前記最優先のジョブを実行する時刻を選択する、付記２記載の情報処理装置。

【０２３３】

（付記４）　前記演算部は、前記割り当てパターンの候補に基づいて前記複数の計算ノードの空き時間の量を比較し、当該比較に応じて前記最優先のジョブを実行する時刻を選択する、付記２または３記載の情報処理装置。

【０２３４】

（付記５）　前記演算部は、前記第２の時刻の候補である複数の候補時刻それぞれで前記最優先のジョブを実行開始する場合の前記複数のジョブの前記複数の計算ノードへの割り当てパターンの候補を示す情報を生成し、前記割り当てパターンの候補の比較に応じて、前記最優先のジョブを実行する時刻を選択する、付記２記載の情報処理装置。

【０２３５】

（付記６）　前記第１の時刻は、前記最優先のジョブを実行可能な最も早い時刻であり、

前記演算部は、前記第１の時刻に基づいて、前記複数の候補時刻の上限の時刻を決定する、付記５記載の情報処理装置。

【０２３６】

（付記７）　前記記憶部は、前記複数のジョブそれぞれの実行に要する計算ノードの数、および、当該計算ノードの利用時間を示す要求資源の情報を記憶し、

前記演算部は、前記複数の計算ノードの利用可能な時間帯と前記複数のジョブそれぞれの前記要求資源とに基づいて、前記第１および前記第２の時刻を特定する、

付記１乃至６の何れか１つに記載の情報処理装置。

【０２３７】

（付記８）　複数のジョブを割り当て可能な複数の計算ノードと、

前記複数の計算ノードの利用可能な時間帯の情報に基づいて、前記複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる管理ノードと、

を有する並列計算機システム。

【０２３８】

（付記９）　コンピュータに、

複数の計算ノードの利用可能な時間帯の情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、

前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる、

処理を実行させるジョブスケジュール設定プログラム。

【０２３９】

（付記１０）　コンピュータが、

複数の計算ノードの利用可能な時間帯の情報に基づいて、複数のジョブのうち最優先のジョブを実行可能な第１の時刻を求め、

前記第１の時刻よりも遅い前記最優先のジョブの実行開始時刻の候補であり前記第１の時刻から前記最優先のジョブを実行するよりも所定の時刻範囲内に多くのジョブを実行できる第２の時刻がある場合、前記第２の時刻から前記最優先のジョブと他のジョブとを実行するよう前記複数の計算ノードに前記複数のジョブを割り当てる、

ジョブスケジュール設定方法。

【符号の説明】

【０２４０】

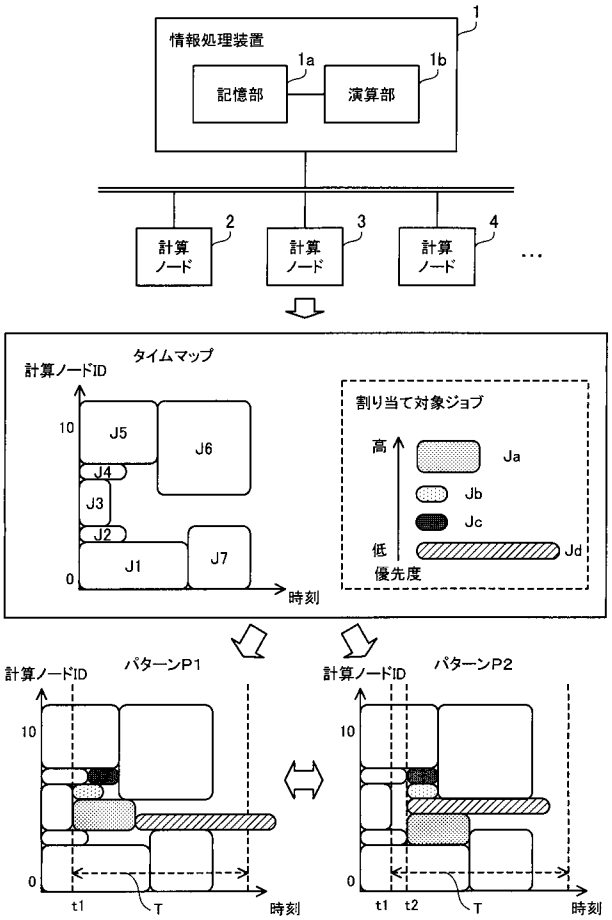
１　情報処理装置

１ａ　記憶部

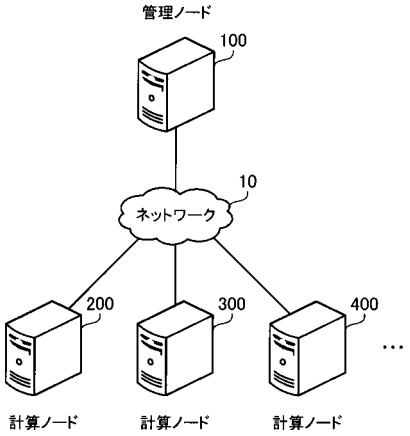
１ｂ　演算部

２，３，４　計算ノード

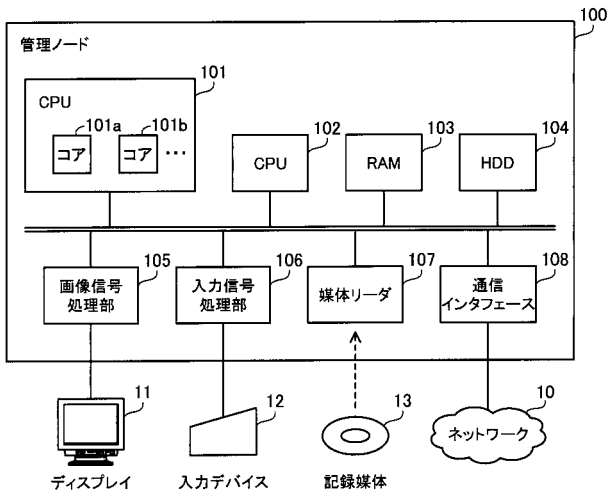
【図１】


情報処理装置
1
記憶部
1a
演算部
1b
計算ノード
2
計算ノード
3
計算ノード
4
タイムマップ
計算ノードID
10
0
J5
J6
J4
J3
J2
J1
J7
時刻
割り当て対象ジョブ
高
低
優先度
Ja
Jb
Jc
Jd
計算ノードID
パターンP1
10
0
t1
T
時刻
計算ノードID
パターンP2
10
0
t1
t2
T
時刻

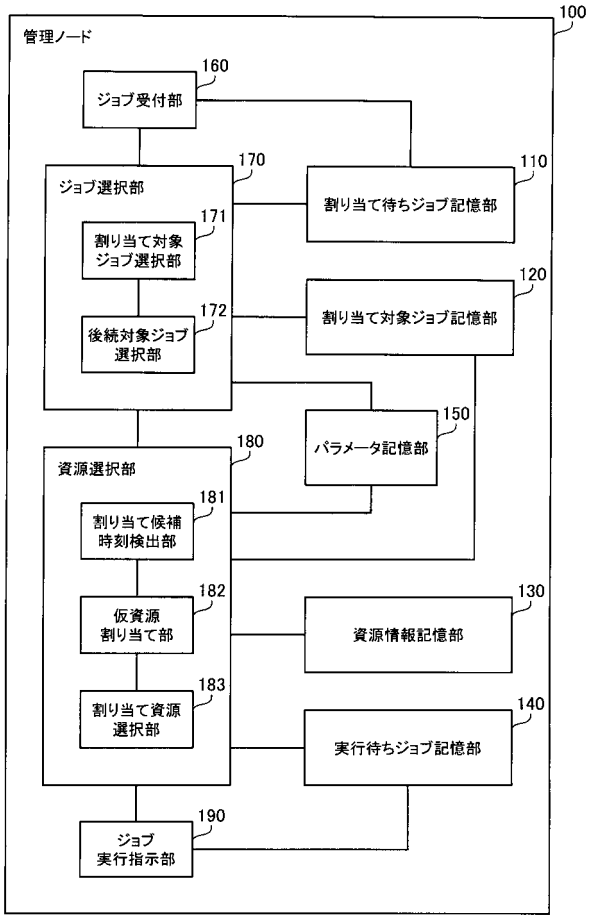
【図２】


管理ノード
100
10
ネットワーク
200
300
400
計算ノード
計算ノード
計算ノード

【図３】

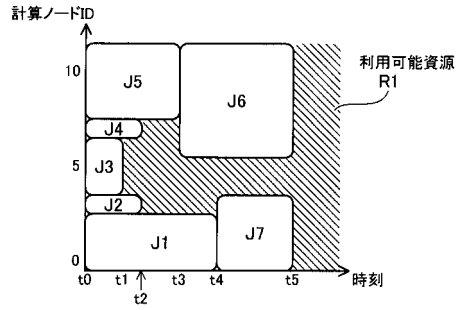

管理ノード
100
CPU
101
コア
101a
コア
101b
CPU
102
RAM
103
HDD
104
画像信号処理部
105
入力信号処理部
106
媒体リーダ
107
通信インタフェース
108
11
ディスプレイ
12
入力デバイス
13
記録媒体
10
ネットワーク

【図４】


管理ノード
100
ジョブ受付部
160
ジョブ選択部
170
割り当て対象ジョブ選択部
171
後続対象ジョブ選択部
172
割り当て待ちジョブ記憶部
110
割り当て対象ジョブ記憶部
120
パラメータ記憶部
150
資源選択部
180
割り当て候補時刻検出部
181
仮資源割り当て部
182
割り当て資源選択部
183
資源情報記憶部
130
実行待ちジョブ記憶部
140
ジョブ実行指示部
190

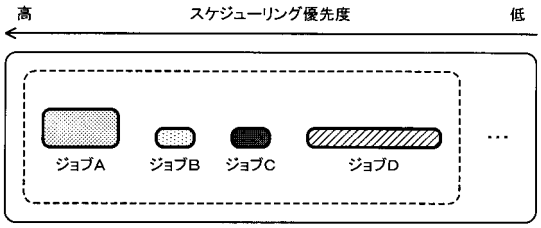
【図５】

タイムマップの例



【図６】

割り当て待ちジョブの優先度の例



【図７】

111

割り当て待ちジョブ情報		
ジョブID	要求資源量	資源使用時間
A	2	tA
B	1	tB
C	1	tC
D	1	tD
…	…	…

【図８】

割り当て対象ジョブ情報の構造体の例

121

```
1  struct schedjobinfo {
2      unit32_t num_jobs;          //ジョブ情報の数
3      struct jobinfo *head_p;     //先頭ジョブ情報へのポインタ
4      struct jobinfo *tail_p;     //終端ジョブ情報へのポインタ
5  };
```

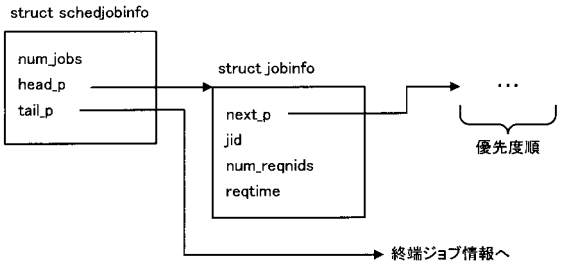
後続対象ジョブ情報の構造体の例

122

```
1  struct jobinfo {
2      struct jobinfo *next_p;     //次のジョブ情報へのポインタ
3      jid_t jid;                  //ジョブID
4      unit32_t num_reqnids;       //要求資源量
5      struct timespec reqtime;    //資源使用時間
6  };
```

【図９】

優先度に従った割り当て対象ジョブの管理例



【図１０】

利用可能資源管理情報の構造体の例 131

```
struct availrschead {
    struct availrsc *head_p;     //先頭利用可能資源情報へのポインタ
};
```

利用可能資源情報の構造体の例 132

```
struct availrsc {
    struct availrsc *next_p;              //次の要素へのポインタ
    struct timespec time;                 //利用可能資源の時刻
    unit32_t num_rsc;                     //資源情報の数
    struct availrscinfo *arihead_p;       //先頭の資源情報へのポインタ
    struct availrscinfo *aritail_p;       //終端の資源情報へのポインタ
};
```

資源情報の構造体の例 133

```
struct availrscinfo {
    struct availrscinfo *next_p;          //次の資源情報へのポインタ
    struct availrscinfo *prev_p;          //前の資源情報へのポインタ
    struct timespec availtime;            //計算資源の利用可能時間
    unit32_t num_nids;                    //利用可能ノード数
    nid_t *nids_p;                        //利用可能ノードID
    struct allocrscinfo allocinfo;        //仮割り当て作業域
};
```

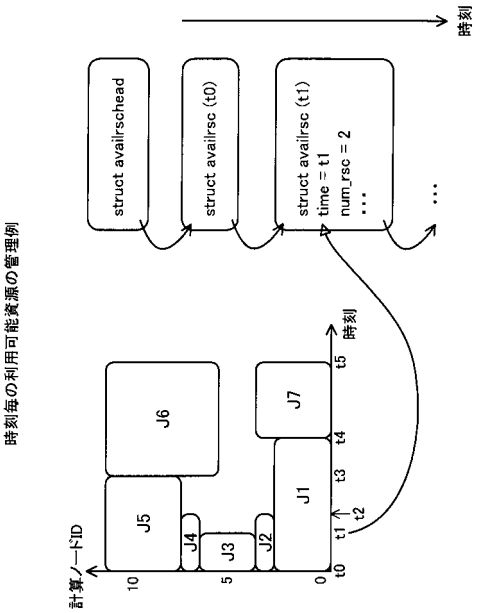
仮割り当て作業情報の構造体の例 134

```
struct allocrscinfo {
    unit32_t total_alloc_nids;            //全仮割り当てノード数
    unit32_t num_ajs;                     //仮割り当てジョブ情報の数
    struct allocrscjobinfo *aj_p;  //先頭仮割り当てジョブ情報へのポインタ
};
```

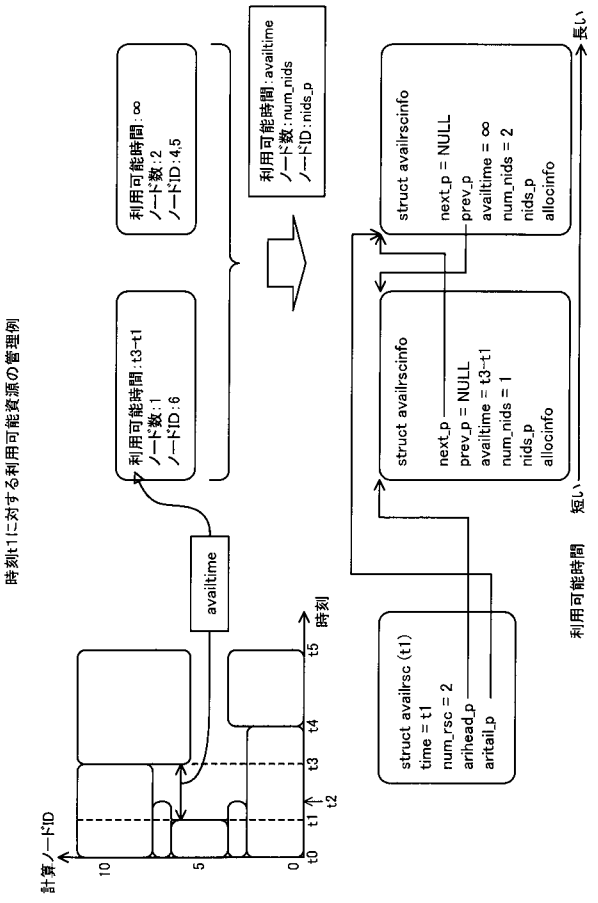
仮割り当てジョブ情報の構造体の例 135

```
struct allocrscjobinfo {
    struct allocrscjobinfo *next_p;   //次の要素へのポインタ
    jid_t jid;                        //ジョブID
    unit32_t num_alloc_nids;          //獲得資源数
};
```

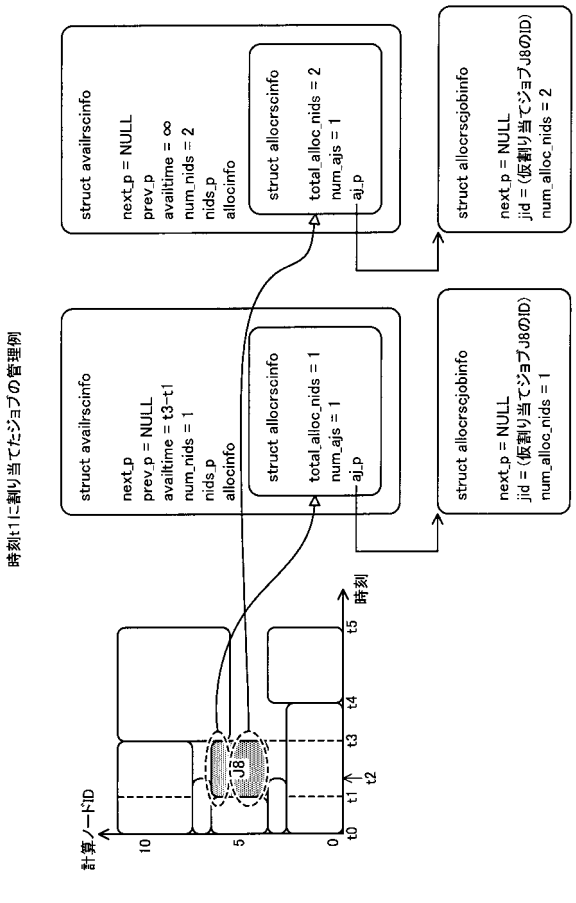
【図１１】



【図１２】



【図１３】



【図１４】

仮割り当て管理情報の構造体の例 136

```
1 struct joballocsummaryhead {
2     unit32_t num_joballocsum;          //仮割り当て情報の数
3     struct joballocsummary *head_p;  //先頭仮割り当て情報へのポインタ
4 };
```

仮割り当て情報の構造体の例 137

```
1 struct joballocsummary {
2     struct joballocsummary *next_p;    //次の要素へのポインタ
3     unit32_t num_jids;                 //割り当てたジョブ数
4     struct timespec alloc_start;       //割り当て開始時刻
5     struct timespec alloc_end;         //割り当て完了時刻
6     struct joballoc *head_p;      //先頭のジョブ割り当て情報へのポインタ
7     struct availrsc *ar_p;        //利用可能資源情報
8 };
```

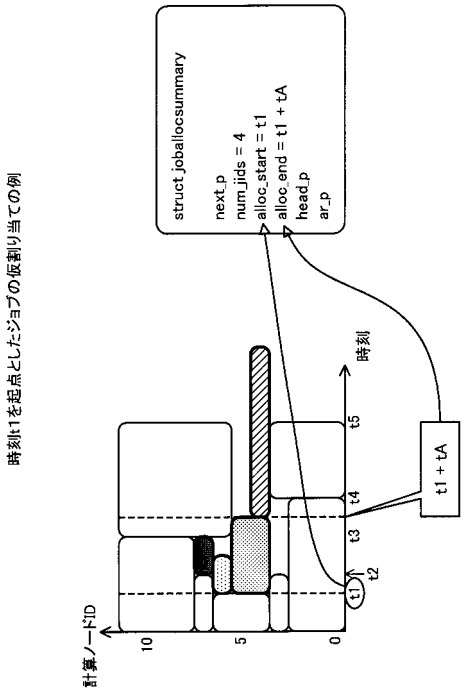
ジョブ割り当て情報の構造体の例 138

```
1 struct joballoc {
2     struct joballoc *next_p;      //次のジョブ割り当て情報へのポインタ
3     jid_t jid;                    //ジョブID
4     struct joballocrscinfo ari;   //ジョブ割り当て資源情報
5 };
```

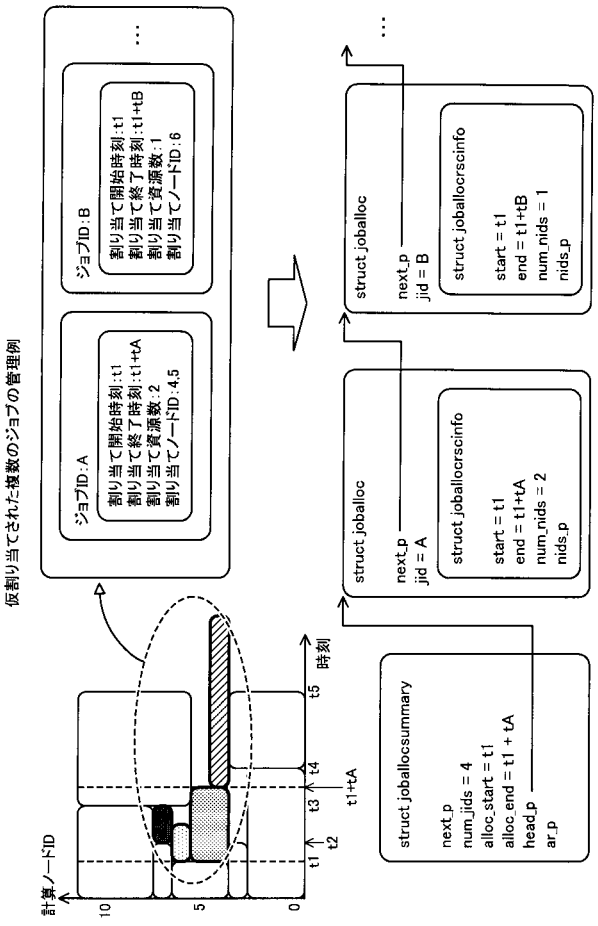
ジョブ割り当て資源情報の構造体の例 139

```
1 struct joballocrscinfo {
2     struct timespec start;        //資源割り当て開始時刻
3     struct timespec end;          //資源割り当て終了時刻
4     unit32_t num_nids;            //割り当て資源数
5     nid_t *nids_p;                //割り当てノードID
6 };
```

【図１５】



【図１６】



【図１７】

141

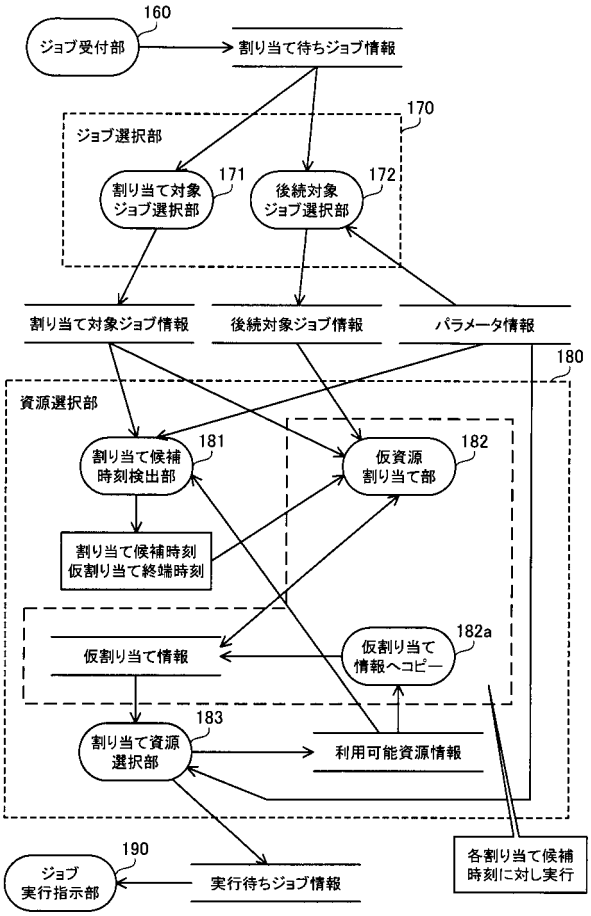
実行待ちジョブ情報		
ジョブID	実行開始時刻	利用資源
A	t2	3, 4
B	t2	6
C	t2	7
D	t2	5
…	…	…

【図１８】

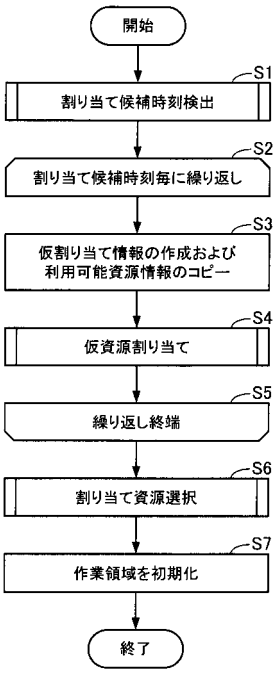
151

パラメータ情報	
評価対象とする後続対象ジョブ数	4
割り当て候補時刻の上限	Δt
割り当て採用基準	(1)仮割り当てジョブ数が多い (2)ジョブの実行完了が早い (3)対象期間の空き資源量が少ない

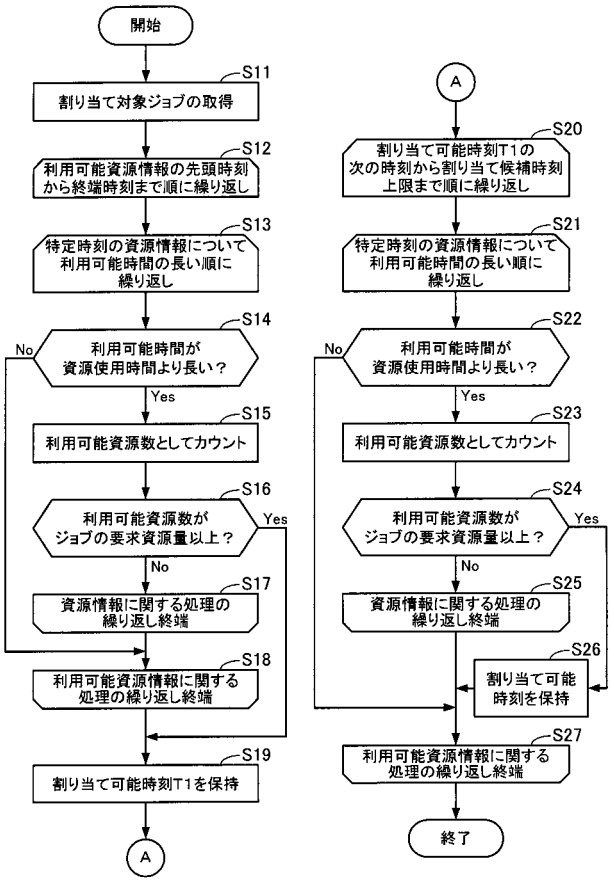
【図１９】



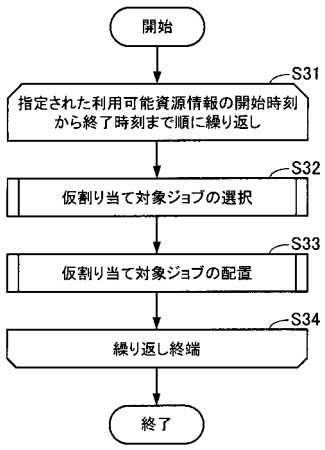
【図２０】



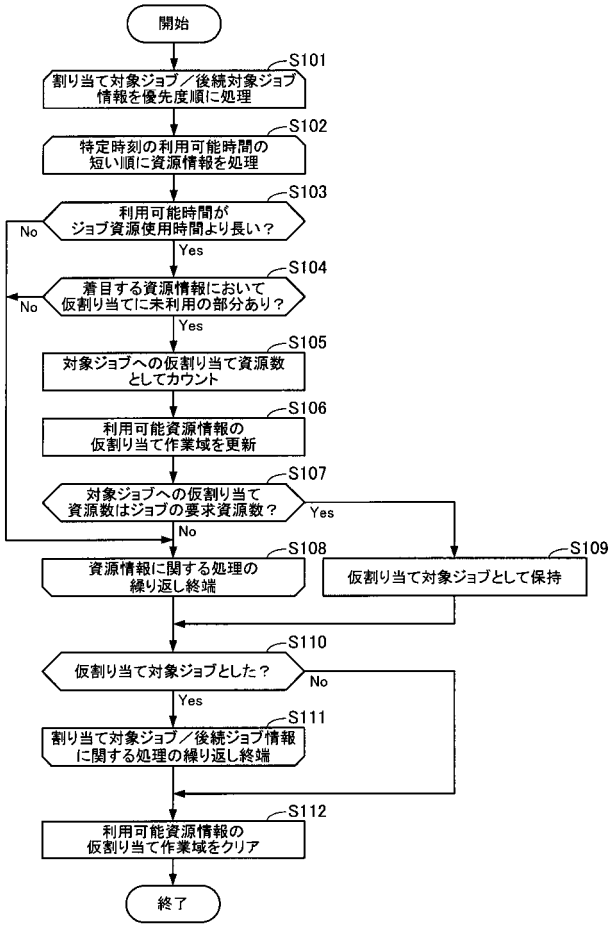
【図２１】



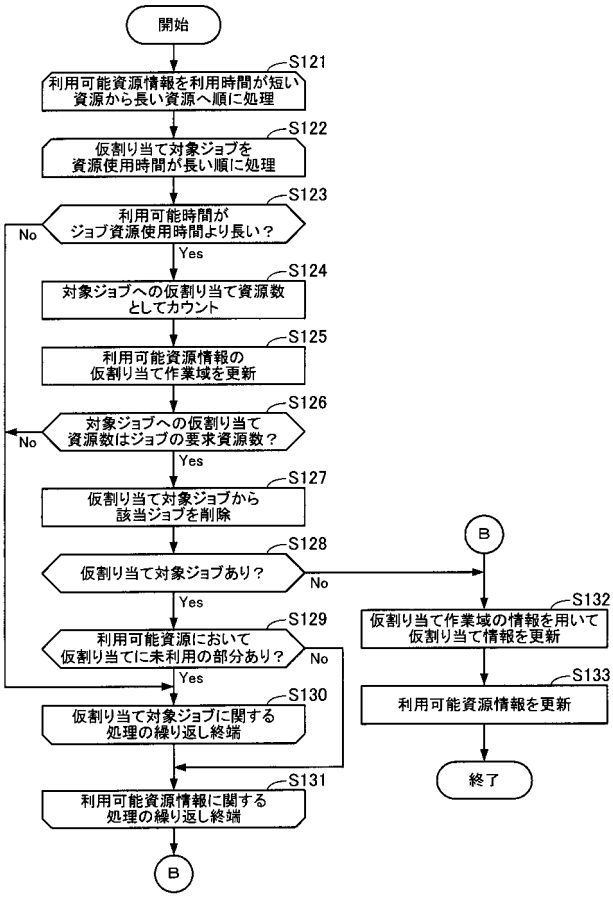
【図２２】



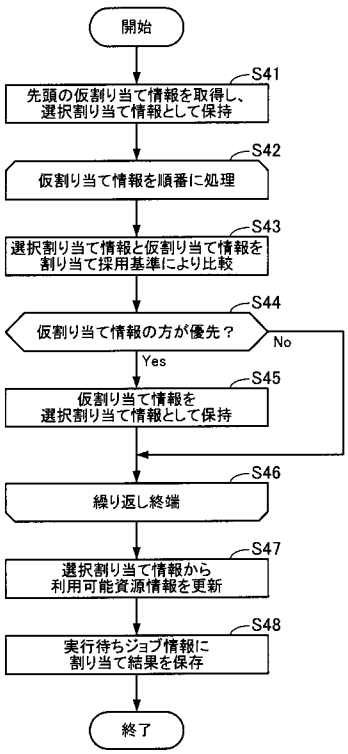
【図２３】



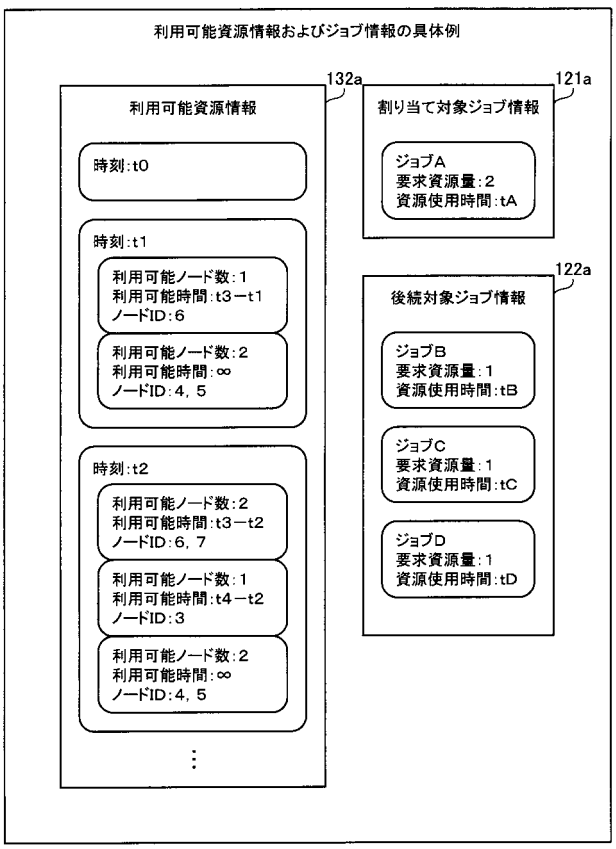
【図２４】



【図２５】

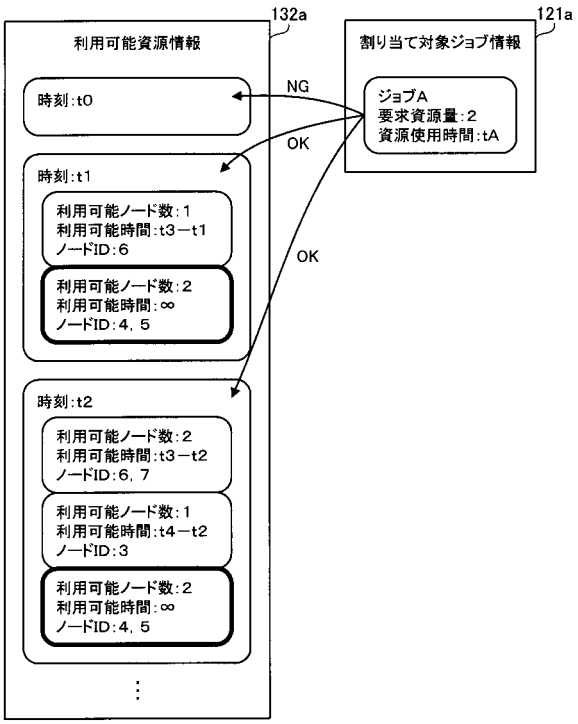


【図２６】



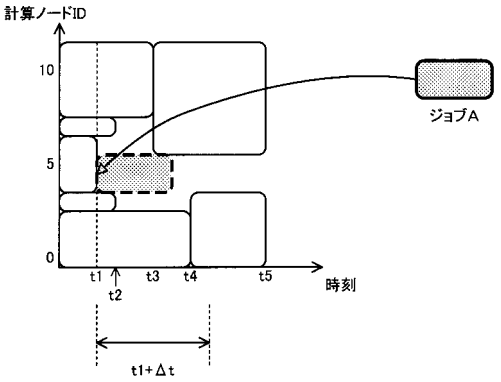
【図２７】

割り当て候補時刻の検出例



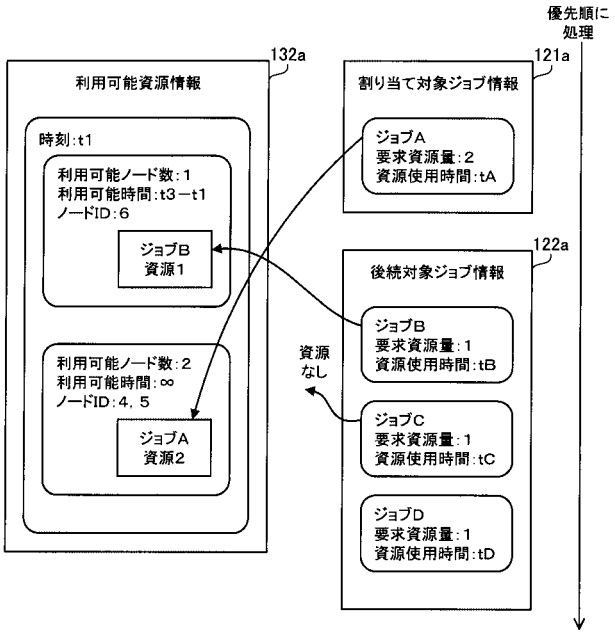
【図２８】

割り当て候補時刻の検出例(続き)



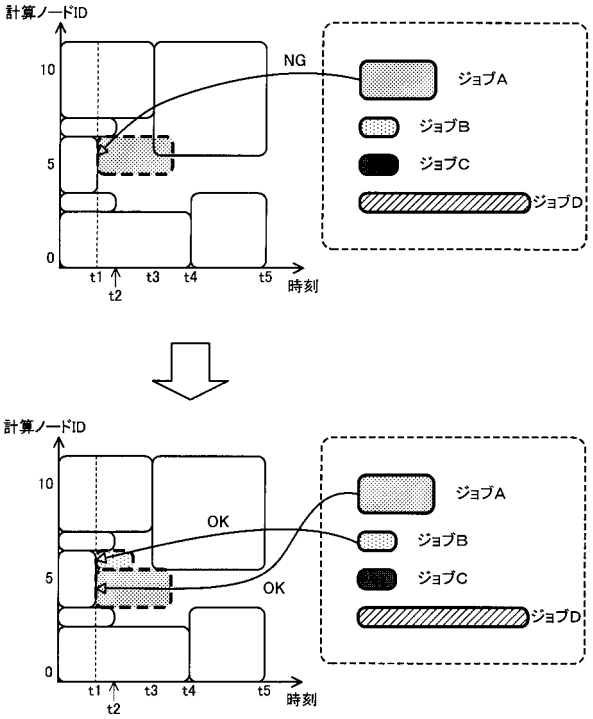
【図２９】

仮割り当て対象ジョブの選択例

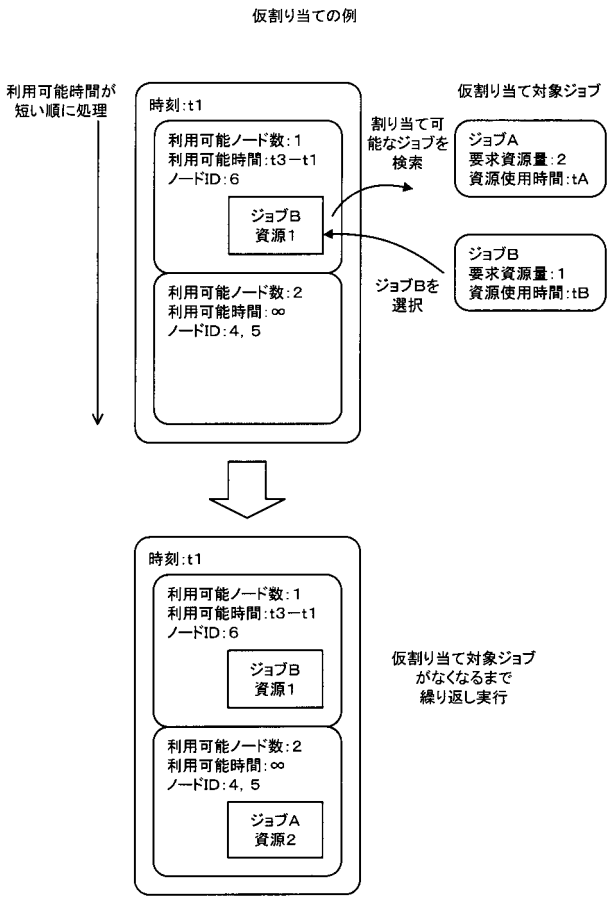


【図３０】

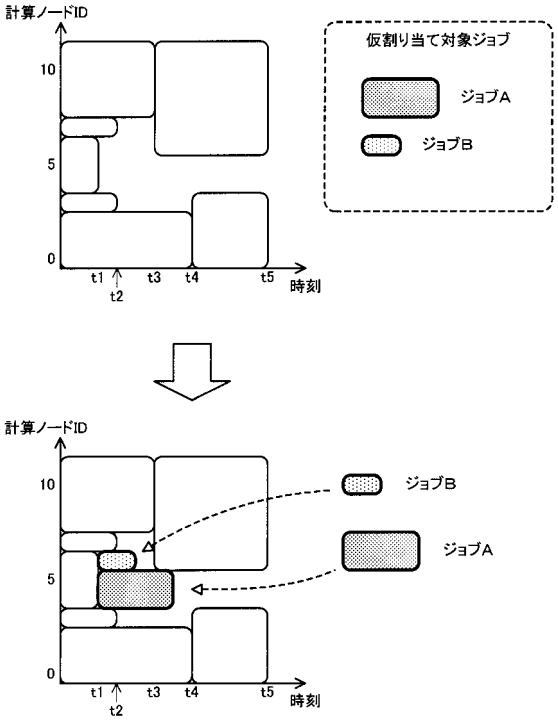
仮割り当て対象ジョブの選択例(続き)



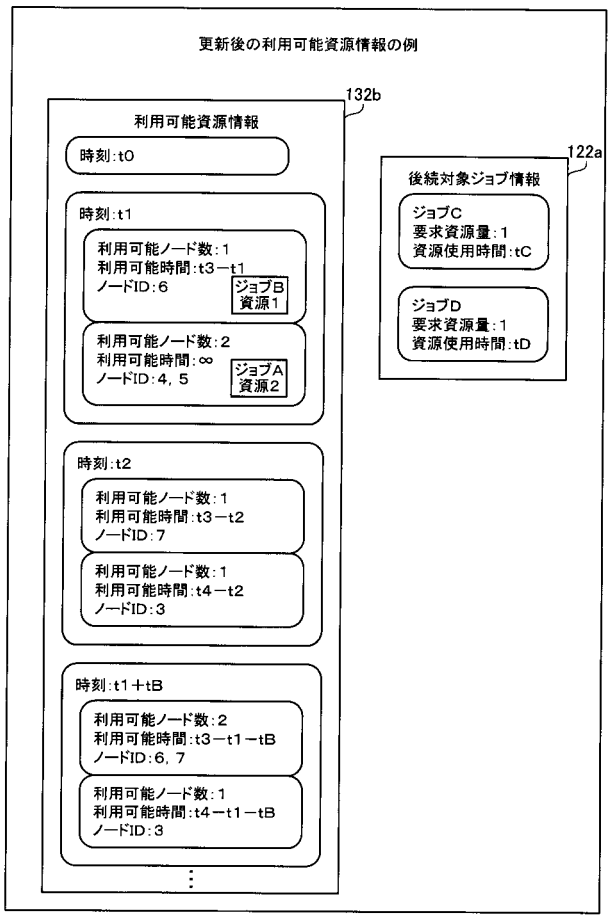
【図３１】



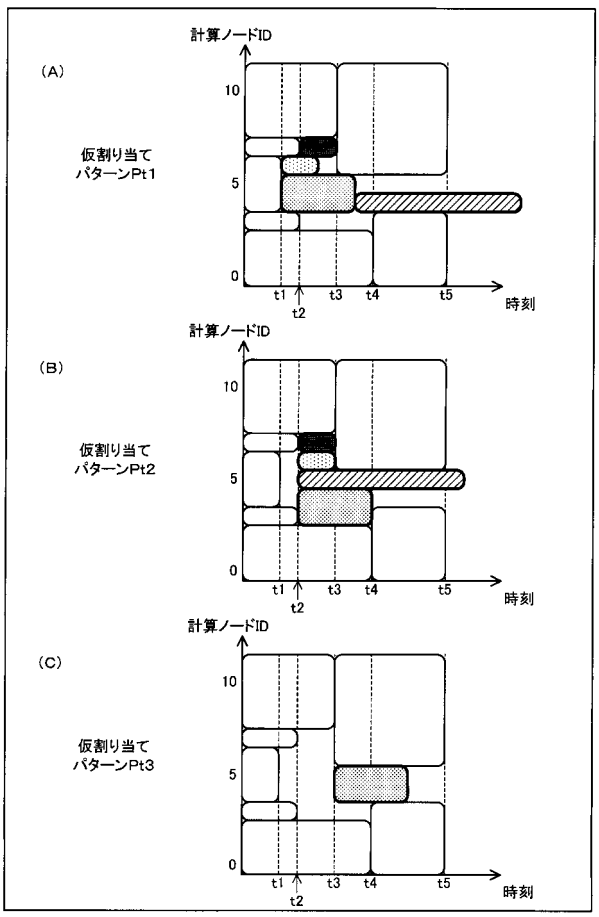
【図３２】



【図３３】

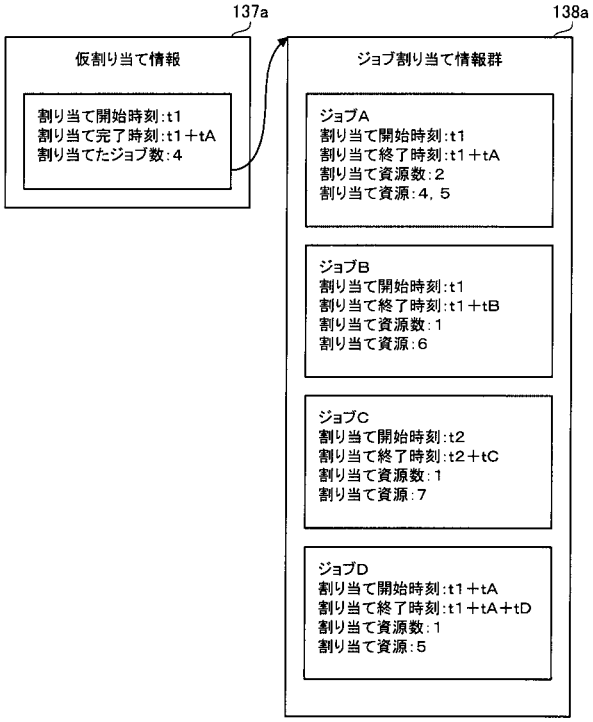


【図３４】



【図３５】

仮割り当て情報（パターンPt1）の例



【図３６】

仮割り当て情報（パターンPt2）の例