Vol.2012-MPS-89 No.7
2012/7/16

複数のメモリ資源要求をもつ計算機システムの待ち行列網による近似性能評価法

アフィザ ラザリ[†1]　木下 俊之[†1]　田辺 睦[†1]

計算機システムの性能を評価する手法のひとつに，待ち行列網を用いる方法がある．本研究では，メモリ資源要求が複数ある計算機システムを待ち行列網でモデル化することを考える．ジョブは網に到着にするとメモリの一部分を確保し，メモリを保持したまま CPU や入出力処理を実行する．すべての CPU や入出力処理を完了すると，メモリを解放して網から離れる．このメモリ資源要求が複数ある計算機システムの待ち行列網は積形解を持たないため，厳密解を求めることは困難である．

そこで，メモリ資源要求が複数ある計算機システムの性能指標を計算するための近似手法を提案する．マルコフ連鎖の状態数が増大するのを防ぐために，網を 2 つの部分に分けて解析する．一方は“処理部”と呼ばれる CPU や入出力処理が実行される部分であり,もう一方は“メモリ部”と呼ばれるジョブがメモリをどのように使うかをモデル化した部分である．数値実験により提案手法を評価し，近似による性能指標の特性を明らかにした．

Queuing Network Approximation Technique for Evaluating Performance of Computer Systems with Multiple Memory Resource Requirements

Afiza Razali[†1]　Toshiyuki Kinoshita[†1]　Akira Tanabe[†1]

Queuing network techniques are effective for evaluating the performance of computer systems. We discuss a queuing network technique for computer systems with multiple memory resource requirements. When a job arrives from outside the network, it occupies one of the memory resources and executes CPU and I/O processing in the network occupying the memory. When the job completes the CPU and I/O processing, it releases the memory and leaves the network. However, because the queuing network model of computer systems with memory resources is an open one, we cannot calculate its exact solutions.

We propose here an approximation technique for calculating the performance measures of computer systems with multiple memory resource requirements using the queuing network technique. This technique involves dividing a network into two parts; one is a “processing part” in which a job executes CPU and I/O processing, and the other is a “memory part” that indicates how the memory resources are used by jobs. By dividing the network into two parts, we can prevent the number of network states from increasing and can approximately calculate the performance measures of the network. We evaluated the proposed approximation technique using numerical experiments.

１．はじめに

計算機システムの性能を評価する手法のひとつに，待ち行列網を用いる方法がある．一般に計算機システムの中では複数のジョブが並列に実行され，CPU や I/O 装置といったハードウェア資源やファイルなどのソフトウェア資源へのアクセスがぶつかることによって，処理の遅れが生じている．待ち行列網を用いて，この処理の遅れが計算機システムの性能にもたらす影響を評価することができる．いくつかの待ち行列網は，積形解と呼ばれる明示的な厳密解を持つ．この積形解によりハードウェアのビジー率やジョブの応答時間といった，計算機システムの性能指標を容易に計算することができる．しかし排他制御が働いていたりメモリ資源があったりすると，待ち行列網は積形解を持たない．積形解を持たない待ち行列網の厳密解を求めるために網の確率的な特徴を記述するマルコフ連鎖を構成し，その平衡方程式を数値的に解く必要がある．この平衡方程式は待ち行列網の状態数と同数の未知数を持つ連立 1 次方程式になる．待ち行列網の状態数は，網内のジョブ数やハードウェア数が増えると劇的に増大するので，平衡方程式の未知数の数も同様に増大し，このため待ち行列網の厳密解を数値的に求めることが非常に困難になる．さらに待ち行列網が外部との出入りのある開モデルの場合は，網の状態数は無数になり得るため(網内のジョブ数が無数になり得る)，その意味でも厳密解を求めることは困難である．

本研究では，メモリ資源を持つ計算機システムを待ち行列網でモデル化することを考える．ジョブが網に到着にするとメモリの一部分を確保し，メモリを保持したまま CPU や入出力処理を実行する．すべての CPU や入出力処理を完了すると，メモリを解放して網から離れる．ゆえにメモリ資源は CPU や I/O 装置に対して 2 次資源と考えることがで

†1 東京工科大学バイオ・情報メディア研究科コンピュータサイエンス専攻
Graduate School of Computer Science, Tokyo University of Technology

きる．待ち行列網が 2 次資源を含むときは，網は積形解を持たない．またメモリ資源のある計算機システムの待ち行列網は開モデルのため，その意味でも厳密解を求めることは困難である．

そこで，メモリ資源要求が複数ある計算機システムの性能指標を計算するための近似手法を提案する．
メモリ資源要求が 1 種類の場合については，すでに[6]で報告した．本論文では[6]の結果を拡張して，複数種類のジョブとメモリ要求が存在する場合を考える（ジョブの各種類をジョブクラスと呼ぶ）．マルコフ連鎖の状態数が増大するのを防ぐために，ジョブが 1 種類の場合と同様に網を 2 つの部分に分割して考える．一つは，“処理部”と呼ばれる CPU や入出力処理の実行をモデル化した部分である．もう一方は“メモリ部”と呼ばれ，ジョブがメモリをどのように使うかをモデル化した部分である．CPU や入出力処理の振る舞いと同様に，メモリ資源の使い方も各ジョブによって異なる．単一のジョブクラスの場合は，処理部もメモリ部も積形解を持っていた．しかし複数ジョブクラスの場合は，処理部は積形解を持つがメモリ部は一般に積形解を持たない．このため，メモリ部の解析には近似手法が必要である．

待ち行列網を 1 次資源と 2 次資源に分けて考えることは，2 階層待ち行列モデルの考え方である．提案手法も，メモリ資源を持つ計算機システムについての 2 階層モデルの一つと考えられる．

2．モデルの記述

処理部は，複数ジョブクラスの通常のセントラルサーバモデルの等価である．網内には K 個のジョブが存在し，各ジョブは添字 k によって k=1, 2, …, K と番号付けられる．処理部は，単一の CPU ノードと M 個の I/O ノードから構成される．I/O ノードは添字 m によって m=1, 2, … , M と番号付けられ，CPU ノードは同じ添字 m によって m=0 と番号付けられる．ジョブクラス k の CPU ノードのサービス率を $\mu_0{}^k$，I/O ノード m のサービス率を $\mu_m{}^k$ とする(m =1, 2, … , M)．各ノードでのサービス時間は，共通の指数分布に従う互いに独立な確率変数である．ジョブはすべてのノードで到着順(FCFS) でスケジュールされる．CPU 処理が終了すると，ジョブは確率的に I/O ノードの一つを選んでそれに移動するか，またはすべての CPU，入出力処理を完了して網から離れる．I/O ノード m の選択確率を $p_m{}^k$ (k =1, 2, … , K; m =1, 2, … , M)，網から離れる確率を $p_0{}^k$ (k =1, 2, … , K)とする．明らかに $\sum_{m=0}^{M} p_m^k = 1$ (k =1, 2, … , K)である．

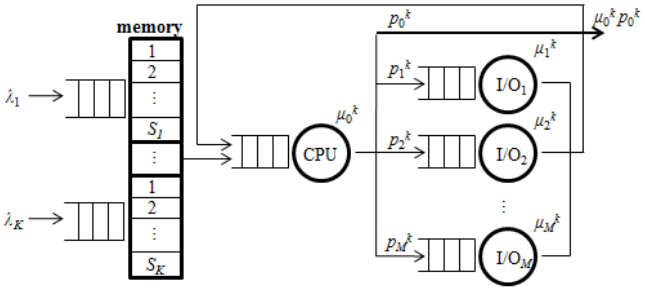

図 1: 複数メモリ資源要求のある
セントラルサーバモデル

このセントラルサーバモデルにメモリ資源を付加する（図 1）．S_k をジョブクラス k のメモリ資源数とする．ジョブクラス k のジョブは，網の外部から到着率 λ_k でランダムに到着する． ジョブは処理部に入るのに先立ちメモリ資源を要求する．ジョブクラス k のジョブが到着したときにクラス k のメモリがすべて占有されていれば，到着したジョブはメモリ待ち行列に入ってメモリが空くのを待つ．ジョブがすべての CPU, I/O 処理を完了したときは，メモリを解放して網から離れる．ジョブは処理部に入る際にメモリを一つ確保するので，ジョブが占有しているメモリ数は常に処理部にいるジョブ数と等しい．ゆえに最大で S_k 個のジョブが処理部に入ることができる．処理部にいるジョブクラス k のジョブ数を n_k (= 0, 1, 2, … , S_k) とする．待ち行列網は外部との出入りのある快モデルだが，“CPU→外部の遷移”を“CPU → CPU の遷移”に置き換えることで処理部を閉セントラルサーバモデルに変更でき，これにより処理部のジョブ数は一定となる（図 2）．この閉モデルでは，“CPU → CPU の遷移”が起きると，そのジョブは終了して新しいジョブが生成されたと考える．ゆえにジョブの平均応答時間は，連続する 2 つの“CPU → CPU の遷移”の間隔時間である．この平均応答時間は，ジョブの生存時間と考えることができる．

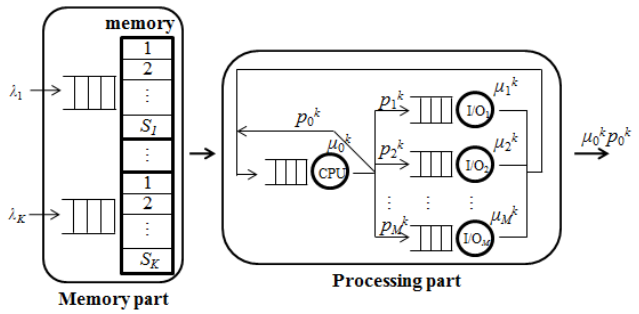

図 2: 近似の考え方

3．近似モデル

セントラルサーバモデルの厳密解を求めるためには，各ジョブクラス毎にマルコフ連鎖を構成してモデル全体を記述する必要がある．しかしこの方法は，待ち行列網のノード数やジョブ数が増えるとマルコフ連鎖の状態数が劇的に増大して数値計算が困難になるという問題がある。セントラルサーバモデルを処理部とメモリ部に分割することで，この状態数の増大を防ぐことができる（図 2）．以下，次の記号を用いる．

τ_{km}：ジョブクラス k のジョブが生存中にノード m で受ける総サービス時間 $(k=1, 2, \dots, K; m=0, 1, \dots, M)$

$\boldsymbol{n}^* = (n_1, n_2, \dots, n_K)$
：ジョブ数ベクトル $(n_k=0, 1, 2, \dots, S_k)$

n_{km}：ノード m におけるジョブクラス k のジョブ数 $(m=0, 1, \dots, M)$

$\boldsymbol{n} = (n_{10}, n_{11}, \dots, n_{1M}, n_{20}, n_{21}, \dots, n_{2M}, \dots, n_{K0}, n_{K1}, \dots, n_{KM})$
：処理部の状態ベクトル

$$\boldsymbol{F}(\boldsymbol{n}) = \{\boldsymbol{n} \mid \sum_{m=0}^{M} n_{km} = n_k, n_{km} \geq 0 \quad (m=0, 1, \dots, M)\}$$
$$(n_k=0, 1, 2, \dots, S_k, k=1, 2, \dots, K)$$

：ジョブクラス k のジョブ数が n_k のときの処理部の可能な状態の集合

$P_s(\boldsymbol{n})$：状態が $\boldsymbol{n}$ のときの定常確率

処理部は複数クラスの通常のセントラルサーバモデルと等価なので，処理部を記述するマルコフ連鎖は積形解を持つ．処理部の定常確率 $P_s(\boldsymbol{n})$ は，次のように書かれる[1][2].

$$P_s(\boldsymbol{n}) = \frac{\prod_{k=1}^{K}\prod_{m=0}^{M}\tau_{km}^{\,n_{km}}}{\varphi(n_1, n_2, \cdots, n_K, M)}$$

ここで $\varphi(n_1, n_2, \cdots, n_K, M) = \sum_{n\in F(n)}\prod_{k=1}^{K}\prod_{m=0}^{M}\tau_{km}^{\,n_{km}}$ は，処理部のジョブクラス k のジョブ数が n_k $(k=1, 2, \dots, K)$ の時の定常確率の正規化定数である．これらの定常確率から処理部のジョブ数が n_k $(k=1, 2, \dots, K)$ のときのジョブクラス k のジョブの処理部の平均応答時間 $T_{\boldsymbol{n}}^{\,k}$ を，次のように求めることができる．

$$T_n^{\,k} = \frac{n_k \cdot \varphi(n_1, \cdots, n_k, \cdots, n_K, M)}{\varphi(n_1, \cdots, n_k - 1, \cdots, n_K, M)}$$

一定であるが，メモリ部を記述する M/M/S_k モデルは占有されているメモリ数に依存してサービス率が変化する．処理部のジョブ数が $\boldsymbol{n}^* = (n_1, n_2, \dots, n_K)$ のときのジョブクラス k のジョブの処理部の平均応答時間 $T_{\boldsymbol{n}}^{\,k}$ $(k=1, 2, \dots, K)$ は，メモリの平均占有時間に等しい．処理部のサービス率 $\nu_{\boldsymbol{n}}^{\,k}$ は $T_{\boldsymbol{n}}^{\,k}$ の逆数だから，$\nu_{\boldsymbol{n}}^{\,k}$ もまた占有されているメモリ数 n_k に依存する．サービス率がサービス中の窓口数に依存するような M/M/S_k 待ち行列モデル $(k=1, 2)$ の状態遷移図を，図 3 に示す．これは 2 次元の生成死滅過程である．メモリ資源数が $\boldsymbol{S}=(S_1, S_2)$ で占有されているメモリ数が $\boldsymbol{n}^*=(n_1, n_2)$ の時の定常確率 $Q_{\boldsymbol{S}}(\boldsymbol{n}^*)=Q_{\boldsymbol{S}}(n_1, n_2)$ の平衡方程式は，次のようになる（さらに高次元の場合も同様である）．

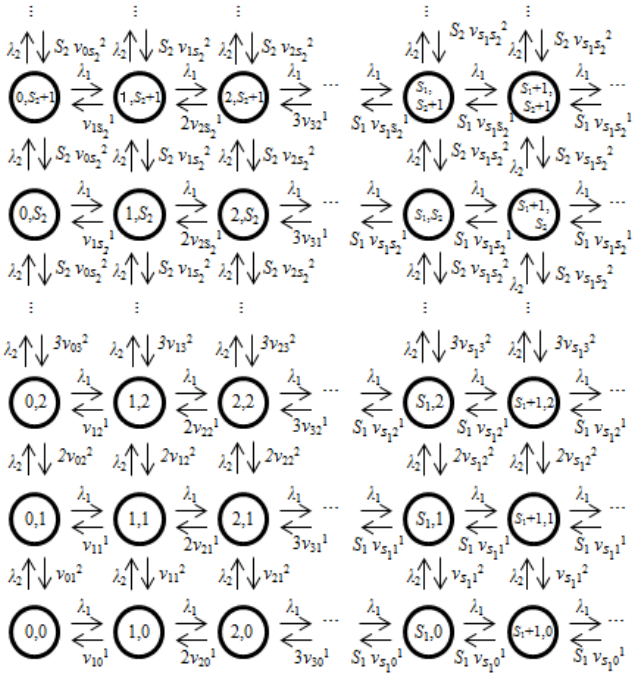

図 3: メモリ部の状態遷移図（2 ジョブクラス）

$$(\lambda_1+\lambda_2)\cdot Q_{\boldsymbol{S}}(0, 0) = \nu_{10}^{\,1}\cdot Q_{\boldsymbol{S}}(1, 0) + \nu_{01}^{\,2}\cdot Q_{\boldsymbol{S}}(0, 1)$$

$$(\lambda_1+\lambda_2+n_1\nu_{n_1 0}^{1})\cdot Q_{\boldsymbol{S}}(n_1, 0) = \lambda_1\cdot Q_S(n_1-1, 0) + (n_1+1)\,\nu_{n_1+1\,0}^{1}\cdot Q_{\boldsymbol{S}}(n_1+1, 0) + \nu_{n_1 1}^{2}\cdot Q_{\boldsymbol{S}}(n_1, 1)$$
$$(n_1=1, 2, \dots, S_1-1)$$

$$(\lambda_1+\lambda_2+S_1\nu_{S_1 0}^{1})\cdot Q_{\boldsymbol{S}}(n_1, 0) = \lambda_1\cdot Q_S(n_1-1, 0) + S_1\nu_{S_1 0}^{1}\cdot Q_{\boldsymbol{S}}(n_1+1, 0) + \nu_{n_1 1}^{2}\cdot Q_{\boldsymbol{S}}(n_1, 1)$$
$$(n_1=S_1, S_1+1, \dots)$$

$$(\lambda_1+\lambda_2+n_2\nu_{0 n_2}^{2})\cdot Q_{\boldsymbol{S}}(0, n_2) = \lambda_2\cdot Q_S(0, n_2-1) + \nu_{1 n_2}^{1}\cdot Q_{\boldsymbol{S}}(1, n_2) + (n_2+1)\,\nu_{0 n_2+1}^{2}\cdot Q_{\boldsymbol{S}}(0, n_2+1)$$
$$(n_2=1, 2, \dots, S_2-1)$$

$$(\lambda_1+\lambda_2+S_2\nu_{0 n_2}^{2})\cdot Q_{\boldsymbol{S}}(0, n_2) = \lambda_2\cdot Q_S(0, n_2-1) + \nu_{1 n_1}^{1}\cdot Q_{\boldsymbol{S}}(1, n_2) + S_2\nu_{0 S_2}^{2}\cdot Q_{\boldsymbol{S}}(0, n_2+1)$$
$$(n_2=S_2, S_2+1, \dots)$$

$$(\lambda_1+\lambda_2+n_1\nu_{n_1 n_2}^{1}+n_2\nu_{n_1 n_2}^{2})\cdot Q_{\boldsymbol{S}}(n_1, n_2) = \lambda_1\cdot Q_S(n_1-1, 0) + \lambda_2\cdot Q_S(0, n_2-1) +$$

$$(n_1+1)\ \nu^1_{n_1+1n_2}\cdot Q_S(n_1+1, n_2)+$$
$$(n_2+1)\ \nu^2_{n_1n_2+1}\cdot Q_S(n_1, n_2+1)$$
$$(n_1=1, 2, \ldots, S_1-1;\quad n_2=1, 2, \ldots, S_2-1)$$

$$(\lambda_1+\lambda_2+S_1\nu^1_{S_1n_2}+n_2\ \nu^2_{n_1S_2})\cdot Q_S(n_1, n_2)=$$
$$\lambda_1\cdot Q_S(n_1-1, 0)+\lambda_2\cdot Q_S(0, n_2-1)+$$
$$S_1\nu^1_{S_1n_2}\cdot Q_S(n_1+1, n_2)+(n_2+1)\ \nu^2_{n_1n_2+1}\cdot Q_S(n_1, n_2+1)$$
$$(n_1=S_1, S_1+1, \ldots;\quad n=1, 2, \ldots, S_2-1)$$

$$(\lambda_1+\lambda_2+n_1\nu^1_{n_1n_2}+S_2\nu^2_{n_1S_2})\cdot Q_S(n_1, n_2)=$$
$$\lambda_1\cdot Q_S(n_1-1, n_2)+\lambda_2\cdot Q_S(n_1, n_2-1)+$$
$$(n_1+1)\ \nu^1_{n_1+1n_2}\cdot Q_S(n_1+1, n_2)+S_2\ \nu^2_{n_1S_2}\cdot Q_S(n_1, n_2+1)$$
$$(n_1=1, 2, \ldots, S_1-1;\quad n_2=S_2, S_2+1, \ldots)$$

$$(\lambda_1+\lambda_2+S_1\nu^1_{S_1n_2}+S_2\nu^2_{n_1S_2})\cdot Q_S(n_1, n_2)=$$
$$\lambda_1\cdot Q_S(n_1-1, n_2)+\lambda_2\cdot Q_S(n_1, n_2-1)+$$
$$S_1\nu^1_{S_1n_2}\cdot Q_S(n_1+1, n_2)+S_2\nu^2_{n_1S_2}\cdot Q_S(n_1, n_2+1)$$
$$(n_1=S_1, S_1+1, \ldots;\quad n_2=S_2, S_2+1, \ldots)$$

この平衡方程式は積形解を持たないので，これを解くためには近似手法が必要である．

メモリ資源が 1 種類の場合は，メモリ部は 1 次元の生成死滅過程で記述される．その状態遷移図を図 4 に示す．またその平衡方程式は，次の通りである．

$$\lambda_1\cdot Q_S(0)=\nu_1^1\cdot Q_S(1)$$
$$(\lambda_1+n_1\ \nu^1_{n_1})\cdot Q_S(n_1)=\lambda_1\cdot Q_S(n_1-1)+$$
$$(n_1+1)\ \nu^1_{n_1+1}\cdot Q_S(n_1+1)\quad (n_1=1, 2, \ldots, S_1-1)$$
$$(\lambda_1+S_1\nu^1_{S_1})\cdot Q_S(n_1)=\lambda_1\cdot Q_S(n_1-1)+)+S_1\nu^1_{S_1}\cdot Q_S(n_1+1)$$
$$(n_1=S_1, S_1+1, \ldots)$$

この平衡方程式は次のような積形解を持つ．

$$Q_S(n_1)=Q_S(0)\cdot\frac{1}{n_1!}\prod_{i=1}^{n_1}\left(\frac{\lambda_1}{\nu_i^1}\right)\qquad (n_1=1, 2, \ldots, S_1-1)$$

$$=Q_S(0)\cdot\frac{1}{S_1!S_1^{n_1-S_1}}\prod_{i=1}^{S_1}\left(\frac{\lambda_1}{\nu^1{}_i}\right)\cdot\left(\frac{\lambda_1}{\nu^1{}_{S_1}}\right)^{n_1-S_1}$$
$$(n_1=S_1, S_1+1, \ldots)$$

この式で $i=1, 2, \ldots, S_1-1$ の状態遷移で $\frac{\lambda_1}{i\cdot\nu_i^1}$ が乗算され，$i=S_1, S_1+1, \ldots$ の状態遷移で $\frac{\lambda_1}{S_1\cdot\nu^1_{S_1}}$ が乗算されている．そこで 2 次元の場合には，図 3 における原点(0, 0) から格子点 (n_1, n_2) までの（最短）経路を考え，この経路上の右向きの遷移に対しては $\frac{\lambda_1}{i\cdot\nu_i^1}$ $(i=1, 2, \ldots, S_1-1)$ または $\frac{\lambda_1}{S_1\cdot\nu^1_{S_1}}$ $(i=S_1, S_1+1, \ldots)$ を乗じ，上方向の遷移に対しては $\frac{\lambda_2}{j\cdot\nu^2_j}$ $(j=1, 2, \ldots, S_2-1)$ または $\frac{\lambda_2}{S_2\cdot\nu^2_{S_2}}$ $(j=S_2,\ S_2+1, \ldots)$ を乗ずる．

このように $Q_S(0, 0)$ に関する $Q_S(n_1, n_2)$ の係数は，(0, 0) から (n_1, n_2) に至る経路に沿った積の形で表現されるものとする．(0, 0)から (n_1, n_2) への経路は複数存在するので，$Q_S(0, 0)$ に関する $Q_S(n_1, n_2)$ の係数は，近似的にすべての経路に沿った積の平均値として表現される．例えば(0, 0) から(2, 1) へは 3 つの経路があるから，$Q_S(2, 1)$ は次のように表される．

$$Q_S(2, 1)\ =\ Q_S(0, 0)\times\frac{1}{3}\cdot\left\{\left(\frac{\lambda_1}{\nu^1_{10}}\right)\left(\frac{\lambda_1}{2\cdot\nu^1_{20}}\right)\left(\frac{\lambda_2}{\nu^2_{21}}\right)\ \text{..... (i)}\right.$$
$$+\left(\frac{\lambda_1}{\nu^1_{10}}\right)\left(\frac{\lambda_2}{\nu^2_{11}}\right)\left(\frac{\lambda_1}{2\cdot\nu^1_{21}}\right)\ \text{.... (ii)}$$
$$\left.+\left(\frac{\lambda_2}{\nu^2_{01}}\right)\left(\frac{\lambda_1}{\nu^1_{11}}\right)\left(\frac{\lambda_1}{2\cdot\nu^1_{21}}\right)\right\}\text{.... (iii)}$$

ここで(i)(ii)(iii)は，図 5 の破線で示された経路に沿った因数の積である．ジョブクラス数が 2 より大きい場合の状態確率も，同様にして近似的に求めることができる．

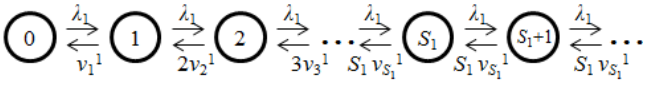

図 4: メモリ部の状態遷移図（単一ジョブクラス）

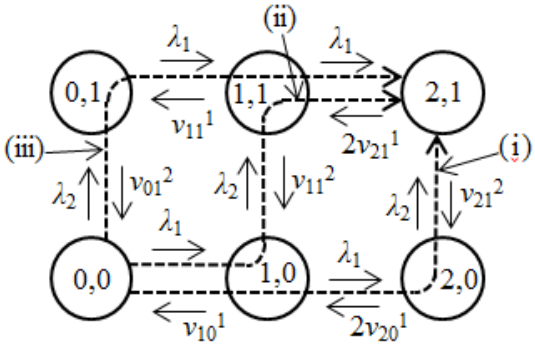

図 5: (2,1)への経路に沿った状態確率の計算

４．数値実験

数値実験により，提案手法を評価する．以下のパラメータ値を用いる．

1. メモリ資源数: (S_1, S_2)=(2, 2), (3,3), (4, 4)
2. 到着率: λ_1=0.0, 1.0, 2.0, …, 21.0,　λ_2 =1.0,　4.0
3. I/O ノード数: M =2
4. 各ノードでの総サービス時間

   τ_{10}=1.0, τ_{11}=τ_{12}=0.5

   τ_{20}=1.0, τ_{21}=τ_{22}=1.0

   ここで τ_{km} はジョブクラス k のジョブがノード m で受ける総サービス時間である．

λ_2 を 1.0 と 4.0 に固定し，λ_1 を 0.0 から 21.0 に変化させた場合のジョブクラス 1 と 2 の平均応答時間 T_1, T_2 を，図 6 と 7 に示す．このグラフの $T_k(S_1, S_2)$ は，ジョブクラス 1, 2 のメモリ資源数がそれぞれ S_1, S_2 のときのジョブクラス k のジョブの平均応答時間である．単一ジョブクラスの場合と同様に，両ジョブクラスのジョブの平均応答時間は下に凸で単調増加な曲線である．λ_2 を固定し λ_1 を増加させると（ジョブクラス 1 の負荷が増加した場合），ジョブクラス 1 だけでなくジョブクラス 2 の平均応答時間も増加する．またジョブクラス 2 の平均応答時間はジョブクラス 1 のそれよりも早く増大するが，これはジョブクラス 2 の方がジョブクラス 1 よりも I/O 時間が長いためと考えられる．

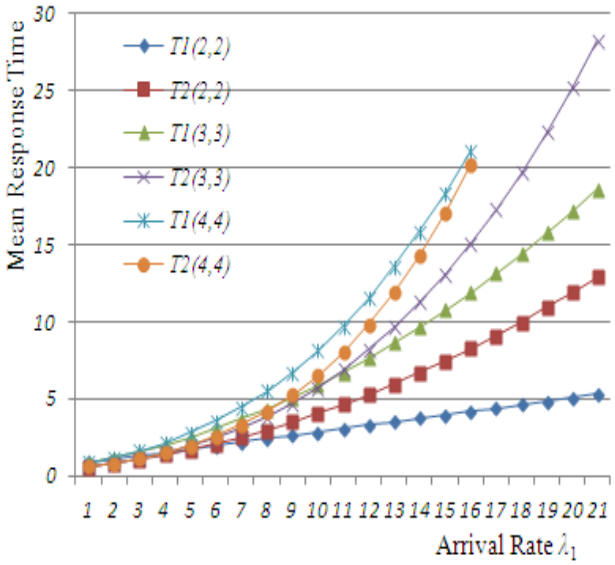

図 6: 平均ジョブ応答時間（λ_2=1.0）

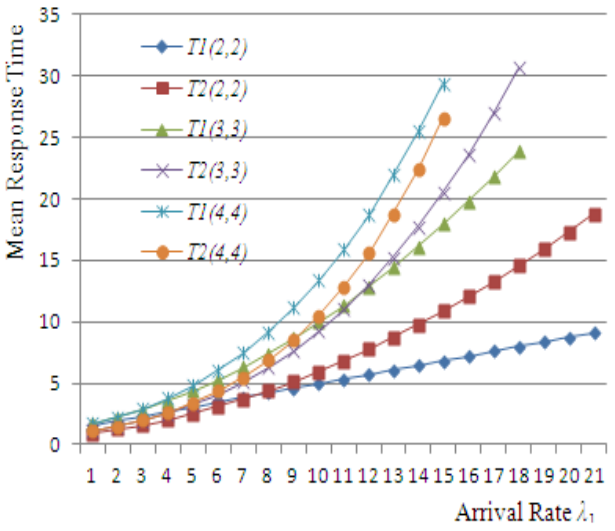

図 7: 平均ジョブ応答時間（λ_2=4.0）

５．おわりに

待ち行列網を用いて複数のメモリ要求を伴う計算機システムを性能評価する近似手法を提案し，数値実験によりその性能指標を解析した．近似の考え方は，待ち行列網の処理部とメモリ部を別々に解析したことである．数値実験によりジョブの平均応答時間の特性を明らかにした．

今後は提案した近似手法の精度を，厳密解またはシミュレーションとの比較で確認する必要がある．

References

[1] F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacious, “Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," J. ACM, Vol.22, No.2, pp.248-260, April 1975.

[2] H. Kobayashi, “Modeling and Analysis," Addison-Wesley Publishing Company, Inc. 1978.

[3] T. Kurasugi and I. Kino, “Approximation Method for Two-layer Queueing Models," Performance Evaluation 36--37, pp.55-70, 1999.

[4] J. A. Rolia and K. C. Sevcik, “The Method of Layers," IEEE Trans. on Software Engineering, Vol.21, No.8, pp.689-700, Aug. 1995.

[5] T. Kinoshita and Y. Takahashi, “A Queuing Network Modeling and Performance Evaluation Method for Computer Systems with Resource Requirement,"
IEICE D-I, Vol. J 82-D-I, No.6, pp.701-710, Jun. 1999.

[6] T. Kinoshita and X. Gao, “Queuing Network Approximation Technique for Evaluating Performance of Computer Systems with Memory Resources,” PDPTA2010, pp.640-645, July 2010