待ち行列研究部会チュートリアル講演

待ち行列ネットワークとその応用

紀　一誠

kino@info.kanagawa-u.ac.jp

神奈川大学理学部情報科学科

講演概要

- はじめに
- 待ち行列ネットワークの概要
- 網を構成する積形式ノード
- 各種の平衡方程式
- 積形式解をもつ待ち行列網
- 待ち行列網の計算法
- 待ち行列網の応用
- まとめ

はじめに

- 待ち行列ネットワークは実用に供されている (数少ない？) 待ち行列
- 理論と応用の境界あたりを重点的に
  - 本当に使いこなすにはある程度理論を知らねばならない
  - どう使われるかを知らなければ意味ある研究はできない

- 待ち行列網は「積形式解」が命
  - 積形式解の成立条件
  - 積形式解を用いた各種の表現と計算法
- 式の細部はあまり気にせず論理の進め方を理解して下さい
- 参考書
  - 森村・高橋 (2001)「混雑と待ち」，朝倉書店.
  - 宮沢政清 (2006)「待ち行列の数理とその応用」，牧野書店.
  - 紀一誠 (2002)「待ち行列ネットワーク」，朝倉書店.

待ち行列網の概要

待ち行列の構造

- ノード：客がサービスを受ける施設
- 客のノード間の移動情報（２種類の指定方法がある）
- サービスクラス：客のサービス要求時間の分布の種類
- ちょっと記号を付けておこう
  - N ノード網：　ノード番号 n, $n \in \{1, 2, \ldots N\}$
  - 網の外部を便宜的にノード 0 とする
  - K クラス網： クラス番号 k, $k \in \{1, 2, \ldots, K\}$
    番号付けは網全体の中で行われる点に注意
  - M 連鎖網：客の網内移動経路が M 種類, $m \in \{1, 2, \ldots, M\}$

2種類の移動経路の指定方法

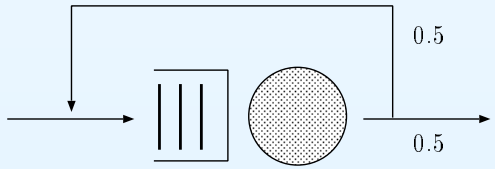
- **確率的経路選択：マルコフ連鎖による表現,(部分）連鎖という**

$$R = (r_{ij}), \quad r_{ij} = \text{ノード } i \text{ を退去した客がノード } j \text{ につく確率}$$

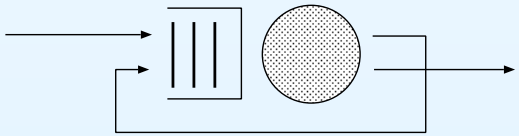
- **確定的経路選択 (ケリー網):クラス k の客が s 番めにつくノード番号 $r(k,s)$ のシークエンスで経路指定する**

$$r(k,1), r(k,2), \ldots, r(k,S(k))$$

- **平均訪問回数と絶対訪問回数の差がでるが，本質的ではない**



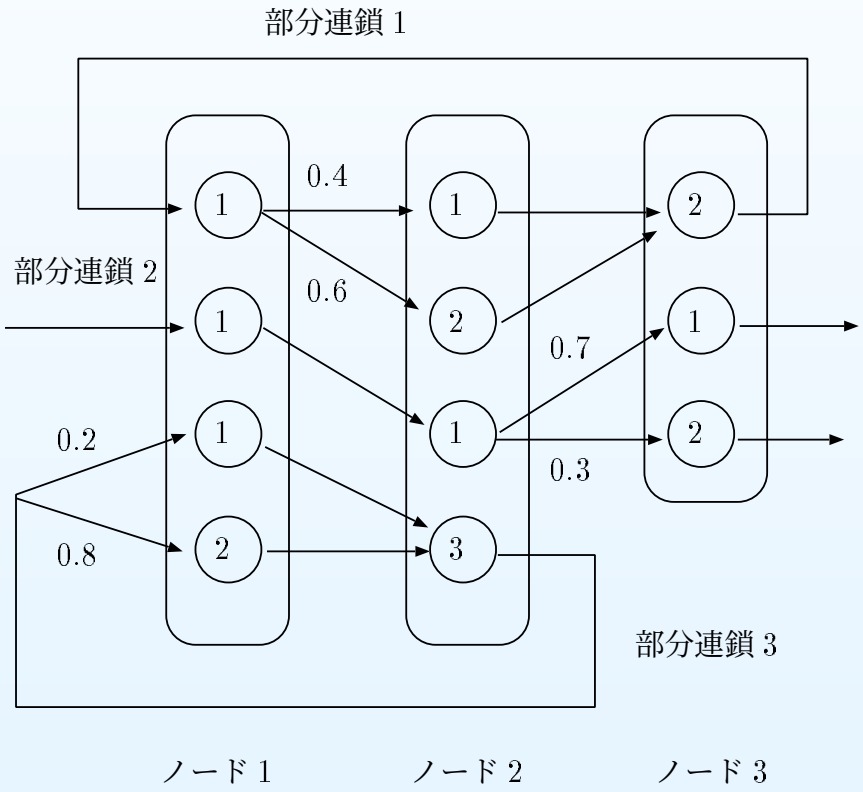
確率的経路選択



確定的経路選択

ノードとサービスクラスとの関係

$$N = 3, K = 3, M = 3$$



客のクラス：

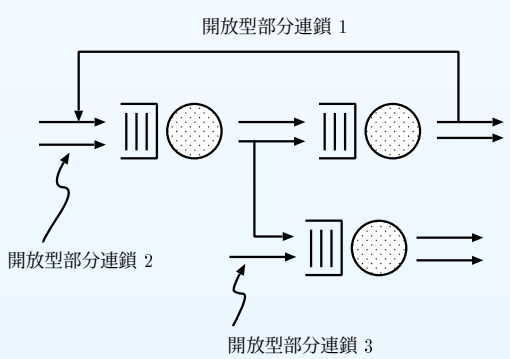
① : $F_1(x) = 1 - e^{-\mu_1 x}$ (数分布)

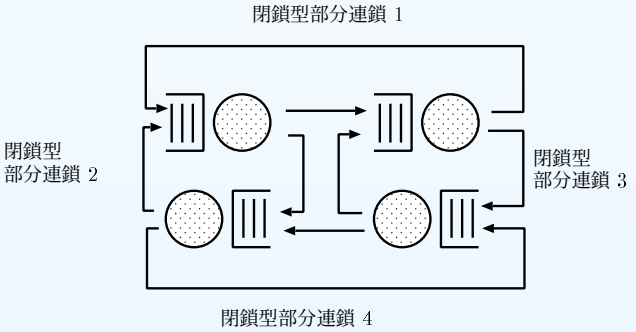
② : $F_2(x) = 1 - e^{-\mu_2 x}$ (数分布)

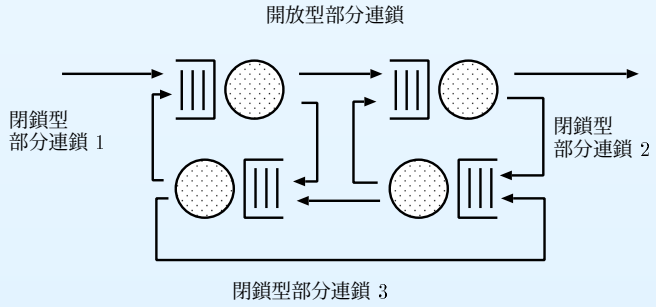
③ : $F_3(x) = 1 - e^{-2\mu_3 x}(1 + 2\mu_3 x)$ (アーラン分布)

客の移動経路による待ち行列網の分類

- **開放型：網外との出入りがある**
- **閉鎖型：網外との出入りがない**
- **混合型：開放型と閉鎖型が混在**







積形式解（原型）

$$
\begin{aligned}
P(\boldsymbol{c},\boldsymbol{y}) &= P(\boldsymbol{c})P(\boldsymbol{y}|\boldsymbol{c}) \\
P(\boldsymbol{c}) &= C\Lambda(\boldsymbol{m})\Phi(\boldsymbol{u})\prod_{j=1}^{N}\tau_j(\boldsymbol{c}_j) \\
\tau_j(\boldsymbol{c}_j) &= \frac{\theta_j(m_{j\ell},k_{j\ell})}{\mu_{k\ell}} \\
P(\boldsymbol{y}|\boldsymbol{c}) &= \prod_{j=1}^{N}\prod_{\ell}^{|\boldsymbol{c}|}\mu_{j\ell}(1-F_{k_{j\ell}}(y_{j\ell}))
\end{aligned}
$$

- $\boldsymbol{c}=(\boldsymbol{c}_1,\boldsymbol{c}_2,\ldots,\boldsymbol{c}_N)$: **滞在客のクラス（状態の離散部分）**
- $\boldsymbol{y}=(\boldsymbol{y}_1,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_1)$: **滞在客の残りサービス時間（連続部分）**

積形式解（周辺分布：1）

$$\int_0^\infty \cdots \int_0^\infty P(\boldsymbol{c}|\boldsymbol{y})d\boldsymbol{y} = 1$$

$$\begin{aligned} P(\boldsymbol{c}) &= C\Lambda(\boldsymbol{m})\Phi(\boldsymbol{u}) \prod_{j=1}^{N} \tau_j(\boldsymbol{c}_j) \\ \tau_j(\boldsymbol{c}_j) &= \frac{\theta_j(m_{j\ell}, k_{j\ell})}{\mu_{k\ell}} \end{aligned}$$

$\boldsymbol{m} = \left(m_m(\boldsymbol{c})\right)_{m=1}^{M}$: $m_m(\boldsymbol{c})$ **状態** $\boldsymbol{c}$ **のときの連鎖** m **の網内総客数**

$\boldsymbol{u} = \left(\left((u_{jmk}(\boldsymbol{c}))_{k=1}^{K}\right)_{m=1}^{M}\right)_{j=1}^{N}$: $u_{jmk}(\boldsymbol{c})$ **状態** $\boldsymbol{c}$ **のときのノード** j **に滞在する連鎖** m**, クラス** k **の客数**

積形式解（周辺分布：2）

- C：**正規化定数**
- $\mu_{j\ell}$: **ノード** j **の位置** ℓ **の客のクラスのサービス要求率**
- $k_{j\ell}$: **ノード** j **の位置** ℓ **の客のクラス**
- $\theta_j(m_{j\ell}, k_{j\ell})$: **ノード** j **への（相対）訪問回数**
- $\Lambda(\boldsymbol{m})$: **到着関数（各連鎖の客の到着率を定める）**

$$\lambda_m(\boldsymbol{m}) = \frac{\Lambda(\boldsymbol{m} + \boldsymbol{e}_m)}{\Lambda(\boldsymbol{m})}$$

- $\Phi(\boldsymbol{u})$: **容量係数（各ノードのサービス率を定める）**

$$\gamma_j(\ell, \boldsymbol{c}) = \frac{\Phi(\boldsymbol{u} - \boldsymbol{e}_j(m_{j\ell}, k_{j\ell}))}{\Phi(\boldsymbol{u})} \beta_j(\ell, \boldsymbol{c})$$

周辺分布の特徴: ここまでで何がわかるか？

- 分布の情報は μ_k（平均サービス要求時間）しか残っていない
  $\Leftrightarrow$ 分布の形には依存しない (insensitive であるという)
- マルコフ連鎖の形で与えた経路情報は（相対）訪問回数 $\theta_j(m,k)$ の形でしか残っていない
  $\Leftrightarrow$ 同じ定常解を与える連鎖ならどれでも同じ積形式解をもつ
- 各連鎖の客の到着率は網全体に滞在する連鎖客数ベクトル m に依存する
  $\Leftrightarrow$ 各ノードへの滞在情報には依存しない
- 各ノードでのサービス率は網全体の状態 u に依存して定まる
  $\Leftrightarrow$ あるノードのサービス率はそのノードの状態だけに依存するわけではない
- 離散状態部分 (c) に関する周辺分布がきれいに計算できる
  $\Leftrightarrow$ 残りサービス時間 (y) は前方再帰時間の形でしか現れない

まとめ

- 到着関数 $\Lambda(m)$ や容量係数 $\Phi(u)$ に様々な形を想定し，積形式解の原型から周辺分布を次々に計算することにより，様々な形の積形式解を導くこと ができる
  $\Leftrightarrow$ ここから先は計算だけの問題となる
- 積形式解の原型が基本である
- 積形式解（原型）を構成するためには，待ち行列網を「積形式ノード」で構成することが必要．積形式ノードは積形式網を作る素材
- 積形式ノードとはなにか？
- 積形式解はどのように導かれるのか？

積形式ノード

お話の流れ

- 一般化 M/G 型待ち行列について紹介する
- その大域平衡方程式を導く
- その方程式の解を積形式に予想してみる
- 積形式解をもつための到着関数と容量係数を仮定する
- 対称型規律について紹介する
- 指数分布をもつクラスについて紹介する
- 到着関数や容量係数についての簡単な補足
- このやりかたを反省してみる

一般化 M/G 型待ち行列

- Poisson **到着過程にしたがい** K **クラスの客が到着する**
- **クラス** k **の客のサービス要求時間は確率変数** S_k **にしたがう**
- **分布関数** $F_k(x) = P(S_k \leq x)$, **密度関数** $f_k(x)$ **が与えられている**
- $E(S_k) = \dfrac{1}{\mu_k}$: **クラス** k **の客の平均サービス要求時間**
- **サービス率や到着率は系の状態に依存して変わる**
- **待ち行列中の客には「位置」といわれる番号がつけられている.**
- **サービス率の配分や到着客の位置決めはあるルールにしたがう**

ノードの状態表現 (n 人の客が滞在)

位置	1	2	3		n
客のクラス	c_1	c_2	c_3	$\cdots$	c_n
残りサービス要求時間	y_1	y_2	y_3	$\cdots$	y_n

ノードの状態：$(\boldsymbol{c}, \boldsymbol{y})$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$, $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$

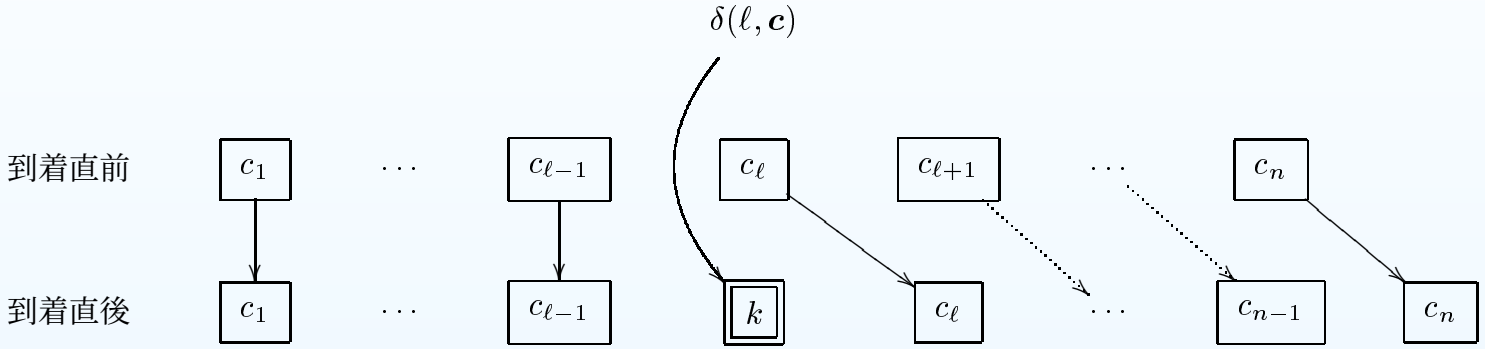
- c_ℓ: **位置 ℓ の客のクラス番号**, $c_\ell \in \{1, 2, \ldots, K\}$
- y_ℓ: **位置 ℓ の客の残りサービス時間**, $0 < y_\ell$
- $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$, $\boldsymbol{y} = (y_1, y_2 \ldots, y_n)$
- $(\boldsymbol{c}, \boldsymbol{y})$: **系の状態**
- **$\boldsymbol{c}$ のサイズ（要素数）を $|\boldsymbol{c}|$ と表すことにする**

到着とサービス

- **サービス率や到着率は系の状態 c に依存して決まる**
- $\lambda_k(\boldsymbol{c})$: **クラス** k **の客の到着率**, $k = 1, 2, \ldots, K$
- $\gamma(\ell, \boldsymbol{c}) = -\dfrac{d}{dt} y_\ell$: **状態** $\boldsymbol{c}$ **の時，位置** ℓ **の客へのサービス率**
  - **客の残りサービス時間は** dt **間に** $\gamma(\ell, \boldsymbol{c})dt$ **減少する**
  - $y_\ell + dy_\ell = y_\ell - \gamma(\ell, \boldsymbol{c})$

クラス k の到着客が状態 $\boldsymbol{c}$ に出合う場合

- **到着直前と直後の位置関係**



- $\delta(\ell, \boldsymbol{c})$: **位置 ℓ を選択する確率**
- **状態に依存するが，クラスには依存しない (クラス独立な選択規律)**
-

$$\sum_{\ell=1}^{n+1} \delta(\ell, \boldsymbol{c}) = 1$$

大域平衡方程式 (global barance equation)

$$\sum_{\ell=1}^{|\boldsymbol{c}|}\Big\{\lambda_{c_\ell}(\boldsymbol{c}^-_{[\ell]})\,\delta(\ell,\boldsymbol{c}^-_{[\ell]})\,f_{c_\ell}(y_\ell)\,P(\boldsymbol{c}^-_{[\ell]},\boldsymbol{y}^-_{[\ell]})+\frac{\partial}{\partial y_\ell}P(\boldsymbol{c},\boldsymbol{y})\gamma(\ell,\boldsymbol{c})\Big\}$$

$$=\sum_{k=1}^{K}\Big\{\lambda_k(\boldsymbol{c})P(\boldsymbol{c},\boldsymbol{y})-\sum_{\ell=1}^{|\boldsymbol{c}|+1}\gamma(\ell,\boldsymbol{c}^+_{[\ell(k)]})P\big(\,\boldsymbol{c}^+_{[\ell(k)]},\;\boldsymbol{y}^+_{[\ell(0)]}\big)\Big\}$$

- **こんな方程式，解けるわけがない！**

第１種の積形式解

それでも何とか解きたいので，以下の解を仮定してみる

$$
\begin{aligned}
P(\boldsymbol{c},\boldsymbol{y}) &= P(\boldsymbol{c})P(\boldsymbol{y}|\boldsymbol{c}) \\
P(\boldsymbol{c}) &= C\Lambda(\boldsymbol{w})\Phi(\boldsymbol{w})\tau(\boldsymbol{c}) \\
\tau(\boldsymbol{c}) &= \prod_{\ell=1}^{|\boldsymbol{c}|}\frac{1}{\mu_{c_\ell}} \\
P(\boldsymbol{y}|\boldsymbol{c}) &= \prod_{\ell=1}^{|\boldsymbol{c}|}\mu_{c_\ell}\bigl(1-F_{c_\ell}(y_\ell)\bigr)
\end{aligned}
$$

ここで，

$\boldsymbol{w}=(w_1,w_2,\ldots,w_K)$，　w_k：**状態 $\boldsymbol{c}$ のときのクラス k の客数**

状態に依存する到着率とサービス率

さらに以下の仮定をする

- **到着関数** $\Lambda(w)$ **(**w **の任意の正実関数)**

$$\lambda_k(\boldsymbol{w}) = \frac{\Lambda(\boldsymbol{w}+\boldsymbol{e}_k)}{\Lambda(\boldsymbol{w})}$$

- **容量係数** $\Phi(w)$ **(**w **の任意の正実関数)**

$$\gamma(\ell, \boldsymbol{c}) = \frac{\Phi(\boldsymbol{w}-\boldsymbol{e}(c_\ell))}{\Phi(\boldsymbol{w})}\beta(\ell, \boldsymbol{c})$$

  - $\beta(\ell, \boldsymbol{c})$: **状態** $\boldsymbol{c}$ **のときの位置** ℓ **へのサービス配分率**
  - $\displaystyle\sum_{\ell=1}^{|\boldsymbol{c}|}\beta(\ell, \boldsymbol{c}) = 1$

第１種の積形式解の成立

以下の場合にが成り立つことが確かめられる

- **すべての可能な状態 $\boldsymbol{c}$ について**

$$\delta(\ell, \boldsymbol{c}^{-}_{[\ell]}) = \beta(\ell, \boldsymbol{c}) \quad \ell = 1, 2, \ldots, |\boldsymbol{c}|$$

ここで， $\boldsymbol{c}^{-}_{[\ell]} = (c_1, c_2, \ldots, c_{\ell-1}, c_{\ell+1}, \ldots, c_{|\boldsymbol{c}|})$

- **分布がすべて指数分布である場合（パラメータは異なってよい）**

$$\sum_{\ell=1}^{|\boldsymbol{c}|} \frac{\Phi(\boldsymbol{w} - \boldsymbol{e}(c_\ell))}{\Phi(\boldsymbol{w})} \mu_{c_\ell} \left\{ \delta(\ell, \boldsymbol{c}^{-}_{[\ell]}) - \beta(\ell, \boldsymbol{c}) \right\} = 0$$

対称型待ち行列 (symmetric queue)

- $\delta(\ell, \boldsymbol{c}^-_{[\ell]}) = \beta(\ell, \boldsymbol{c})$ **を満たす規律を対称型規律といい，対称型規律をもつ待ち行列を対称型待ち行列という**
- **対称型待ち行列は積形式ノードとなる** $\Rightarrow$ **積形式網の素材になれる**
- **対称型規律は，到着客が位置** ℓ **を選択し，その結果状態が** $\boldsymbol{c}$ **となる割合と，状態** $\boldsymbol{c}$ **のとき位置** ℓ **の客にサービス率が配分される割合が均衡する規律**
- **例　プロセッサシェアリング：** $\beta(\ell, \boldsymbol{c}) = \delta(\ell, \boldsymbol{c}^-_{[\ell]}) = \dfrac{1}{|\boldsymbol{c}|}$
- **例** LCFS-PR : $\beta(1, \boldsymbol{c}) = \delta(1, \boldsymbol{c}^-_{[\ell]}) = 1$, **その他は** 0

指数分布型待ち行列

- $\delta(\ell, \boldsymbol{c}^-_{[\ell]}) \neq \beta(\ell, \boldsymbol{c})$ **である規律を持つ場合は，サービス要求時間のクラス** (K) **はすべて指数分布でなければならない**
- **サービス要求時間分布がすべて指数分布にしたがい，以下の規律を満たす待ち行列を「指数分布型待ち行列」ということにする**

$$\sum_{\ell=1}^{|\boldsymbol{c}|} \frac{\Phi(\boldsymbol{w} - \boldsymbol{e}(c_\ell))}{\Phi(\boldsymbol{w})} \mu_{c_\ell} \left\{ \delta(\ell, \boldsymbol{c}^-_{[\ell]}) - \beta(\ell, \boldsymbol{c}) \right\} = 0$$

- **指数分布型待ち行列は積形式ノードである** $\Rightarrow$ **積形式網の素材になれる**
- **例** FCFS: $\mu_1 = \mu_2 = \cdots = \mu_K$ **，**

$$\delta(\ell, \boldsymbol{c}) = \begin{cases} 1 & \ell = |\boldsymbol{c}| + 1 \\ 0 & \text{その他} \end{cases}, \quad \beta(\ell, \boldsymbol{c}) = \begin{cases} 1, & \ell = 1 \\ 0, & \text{その他} \end{cases}$$

到着関数

- **到着関数 $\Lambda(w)$ の形により様々な到着を表せる**
- **クラス k の客の到着率**: $\lambda_k(\boldsymbol{w}) = \dfrac{\Lambda(\boldsymbol{w}+\boldsymbol{e}_k)}{\Lambda(\boldsymbol{w})}$
  - **例 一定到着率：$\Lambda(\boldsymbol{w}) = \lambda_1^{w_1}\lambda_2^{w_2}\ldots\lambda_K^{w_K}$ とすれば $\lambda_k(\boldsymbol{w}) = \lambda_k$**
  - **例　呼損型到着 S までしか入れない**

$$\Lambda(\boldsymbol{w}) = \begin{cases} \lambda^{|\boldsymbol{c}|}, & |\boldsymbol{c}| \le S-1 \\ 0, & S \le |\boldsymbol{c}| \end{cases} \quad \textbf{とすれば} \quad \lambda_k(\boldsymbol{w}) = \begin{cases} \lambda, & |\boldsymbol{c}| \le S \\ 0, & S < |\boldsymbol{c}| \end{cases}$$

容量係数

- **容量係数 $\Phi(w)$ の形により様々なサービス能力を表せる**
- **状態 $\boldsymbol{c}$ のときの位置 ℓ の客へのサービス率**

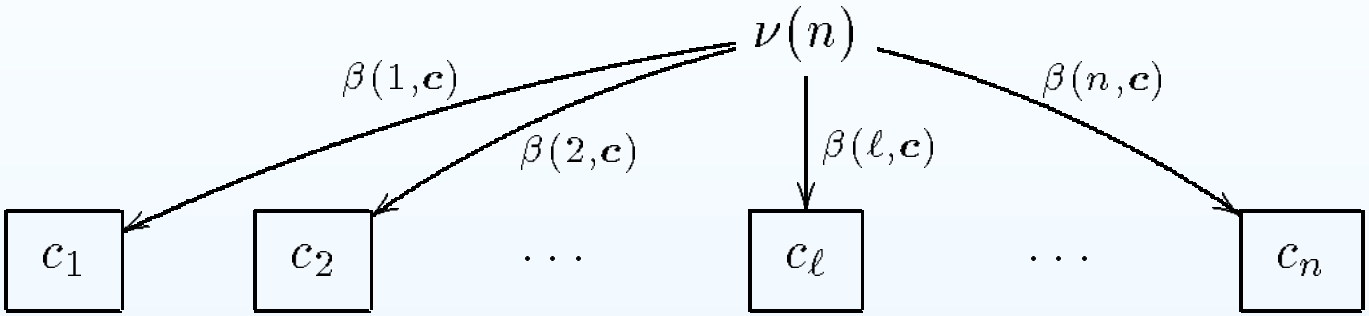
$$\gamma(\ell, \boldsymbol{c}) = \frac{\Phi(\boldsymbol{w} - \boldsymbol{e}(c_\ell))}{\Phi(\boldsymbol{w})} \beta(\ell, \boldsymbol{c})$$

- **例　客数依存サービス率：$\Phi(\boldsymbol{w}) = \Phi(|\boldsymbol{w}|)(= \Phi(|\boldsymbol{c}|))$ の場合**

$$\nu(|\boldsymbol{c}|) = \frac{\Phi(|\boldsymbol{c}| - 1)}{\Phi(|\boldsymbol{c}|)}, \quad |\boldsymbol{c}| = 1, 2, \ldots$$

$$\Phi(n) = \Big\{ \prod_{s=1}^{n} \nu(s) \Big\}^{-1}, \quad n = 1, 2, \ldots, \quad \Phi(0) = 1$$

状態依存サービス率 $\nu(n)$ とその配分率 $\beta(\ell, \boldsymbol{c})$



- $\nu(n)$ **あるいは** $\Phi(n)$ **を適当に選ぶことにより，複数サーバ，無限サーバ等を表すことができる**
- **さらには，** $\Phi(w)$ **を適当に選ぶことにより，さらに複雑なサービス能力を表現することができる**

このやりかたを反省してみよう

- 大域平衡方程式は線形であるから，予想解がそれを満たすことが示せたなら，それは真の解である（みつけたもの勝ち？）
  $\Rightarrow$ この言い分は確かに 正しい
- それではしかし，積形式解はどのようにして予想されたのであろうか？
- また，到着関数や容量係数がこのような形になる必然性はあるのだろうか？
- 方程式の構造の中にこのような積形式解を導き出す論理構造が隠されているのではないだろうか？
- ということで，方程式の構造について詳しく調べてみよう

各種の平衡方程式

大域平衡方程式をもう一度

$$\sum_{\ell=1}^{|\boldsymbol{c}|}\Big\{\lambda_{c_\ell}(\boldsymbol{c}^-_{[\ell]})\,\delta(\ell,\boldsymbol{c}^-_{[\ell]})\,f_{c_\ell}(y_\ell)\,P(\boldsymbol{c}^-_{[\ell]},\boldsymbol{y}^-_{[\ell]})+\frac{\partial}{\partial y_\ell}P(\boldsymbol{c},\boldsymbol{y})\gamma(\ell,\boldsymbol{c})\Big\}$$

$$=\sum_{k=1}^{K}\Big\{\lambda_k(\boldsymbol{c})P(\boldsymbol{c},\boldsymbol{y})-\sum_{\ell=1}^{|\boldsymbol{c}|+1}\gamma(\ell,\boldsymbol{c}^+_{[\ell(k)]})P\big(\,\boldsymbol{c}^+_{[\ell(k)]},\;\boldsymbol{y}^+_{[\ell(0)]}\big)\Big\}$$

- **大域平衡方程式を解くことをひとまずあきらめ，方程式が成り立つための十分条件を探すことにする．**
- **十分条件として左辺=0, 右辺の和の各項=0 を考える**

局所平衡方程式 (部分平衡方程式)

$$(1)\quad \sum_{\ell=1}^{|\boldsymbol{c}|}\{\lambda_{c\ell}(\boldsymbol{c}^-_{[\ell]})\,\delta(\ell,\boldsymbol{c}^-_{[\ell]})\,f_{c\ell}(y_\ell)\,P(\boldsymbol{c}^-_{[\ell]},\boldsymbol{y}^-_{[\ell]})+\frac{\partial}{\partial y_\ell}P(\boldsymbol{c},\boldsymbol{y})\gamma(\ell,\boldsymbol{c})\}=0$$

および $k=1,2,\ldots,K$ **について**

$$(2)\quad \lambda_k(\boldsymbol{c})P(\boldsymbol{c},\boldsymbol{y})-\sum_{\ell=1}^{|\boldsymbol{c}|+1}\gamma(\ell,\boldsymbol{c}^+_{[\ell(k)]})P\big(\,\boldsymbol{c}^+_{[\ell(k)]},\ \boldsymbol{y}^+_{[\ell(0)]}\big)=0$$

- **これらの方程式を局所平衡方程式** (local balance equation) **という**
- **局所平衡方程式は成り立つかどうかは一般にはわからない**
- **局所平衡方程式を満たす解は大域平衡方程式の解となっている**
- **想定解が局所平衡方程式を満たすための条件を探していく**

局所平衡方程式の解の確認

- (2) 式に積形式解を代入し，満たすことを確認する
  $\Rightarrow$ この時に到着関数の設定法が見えてくる
- 次いで，(1) 式に積形式解を代入してみる．計算の結果，積形式解が局所平衡方程式を満たすためには以下の関係が成り立っていなければならないことがわかる

$$(3) \quad \sum_{\ell=1}^{|\boldsymbol{c}|} \frac{\Phi(\boldsymbol{w}-\boldsymbol{e}(c_\ell))}{\Phi(\boldsymbol{w})} \frac{f_{c_\ell}(y_\ell)}{1-F_{c_\ell}(y_\ell)} \left\{ \delta(\ell, \boldsymbol{c}^-_{[\ell]}) - \beta(\ell, \boldsymbol{c}) \right\} = 0$$

- (3) を満たす解は局所平衡方程式を満たし，したがって，大域平衡方程式を満たす
- (3) を満たすための十分条件を再び探してみる

ステーションバランス方程式

- (3) を満たす十分条件のひとつは，すべての可能な状態 $\boldsymbol{c}$ について次が成り立つことである

$$(4) \quad \delta(\ell, \boldsymbol{c}^{-}_{[\ell]}) - \beta(\ell, \boldsymbol{c}) = 0, \quad \ell = 1, 2, \ldots, |\boldsymbol{c}|$$

- (4) をステーションバランス方程式という
- 対称型規律はステーションバランス方程式を満たしている
- したがって，対称型待ち行列は積形式解をもつ

指数分布型待ち行列

- $\delta(\ell, \boldsymbol{c}^-_{[\ell]}) \neq \beta(\ell, \boldsymbol{c})$ **の場合，(3) がすべての** $0 < y_\ell$ **について成り立つためには**

$$\frac{f_{c_\ell}(y_\ell)}{1 - F_{c_\ell}(y_\ell)} = \text{定数}$$

  でなければならない．
- **このような性質をもつ分布は指数分布に限られる**
- **すべての分布が指数分布に従う場合には (3) は次となる**

$$\sum_{\ell=1}^{|\boldsymbol{c}|} \frac{\Phi(\boldsymbol{w} - \boldsymbol{e}(c_\ell))}{\Phi(\boldsymbol{w})} \mu_{c_\ell} \left\{ \delta(\ell, \boldsymbol{c}^-_{[\ell]}) - \beta(\ell, \boldsymbol{c}) \right\} = 0$$

- **この条件を満たす指数分布型待ち行列が積形式ノードになる**

大域平衡方程式と局所平衡方程式

- S: **状態空間**, $q(\boldsymbol{x}, \boldsymbol{x'})$: **状態推移率** $(\boldsymbol{x} \to \boldsymbol{x'})$, $\pi(\boldsymbol{x})$: **定常状態確率**
- **大域平衡方程式：状態** $\boldsymbol{x}$ **への ”出入り ”が平衡**

$$\pi(\boldsymbol{x}) \sum_{\boldsymbol{x'} \in S} q(\boldsymbol{x}, \boldsymbol{x'}) = \sum_{\boldsymbol{x'} \in S} \pi(\boldsymbol{x'}) q(\boldsymbol{x'}, \boldsymbol{x}), \quad \boldsymbol{x} \in S$$

- **局所平衡方程式：**$S = S_1 \cup S_2 \cup \cdots \cup S_n$ **各** S_i **について**

$$\pi(\boldsymbol{x}) \sum_{\boldsymbol{x'} \in S_i} q(\boldsymbol{x}, \boldsymbol{x'}) = \sum_{\boldsymbol{x'} \in S_i} \pi(\boldsymbol{x'}) q(\boldsymbol{x'}, \boldsymbol{x}), \quad \boldsymbol{x} \in S_i$$

  - **局所平衡方程式の成立は保証されているわけではない**
  - **局所平衡方程式を満たす解は大域平衡方程式の解である**
  - **局所平衡方程式の本数は大域平衡方程式より多い**
  - **局所平衡方程式の方が解きやすい（解の見当をつけやすい）**

詳細平衡方程式と可逆過程

- **定常マルコフ過程 $X(t)$ を時間反転させた $X(-t)$ を逆過程という**
- **可逆過程：$X(t)$ と $X(-t)$ が同じ確率過程 (同一の結合分布をもつ)**
- **$X(t)$ が可逆であるための必要十分条件**

$$\pi(\boldsymbol{x})q(\boldsymbol{x},\boldsymbol{x'}) = \pi(\boldsymbol{x'})q(\boldsymbol{x'},\boldsymbol{x}), \quad \boldsymbol{x},\boldsymbol{x'} \in S$$

- **この方程式を詳細平衡方程式** (detailed balance equation) **という**
- **詳細平衡方程式を満たす解は大域平衡方程式を満たす**
- **詳細平衡方程式は局所平衡方程式の特別な場合になる**
- **定常** M/M/1 **待ち行列は可逆過程の一例**

準可逆な待ち行列と M⇒M プロパティ

- $x(t)$: K クラスの客が到着する一般の待ち行列の時刻 t の状態
- 次の性質を持つときこの待ち行列は準可逆 (quasi-reversible)
  - $x(t_0)$ が t_0 以降に到着するクラス k 客の到着時刻と独立
  - $x(t_0)$ が t_0 以前のクラス k 客の退去時刻と独立
- 準可逆性は待ち行列が積形式ノードとなる十分条件をあたえる
- 準可逆な待ち行列は M⇒M プロパティをもつともいわれる
- M⇒M プロパティとは，客の到着が Poisson 過程に従うなら，退去過程も Poisson 過程に従う，という性質をいう
- M⇒M プロパティをもつ待ち行列は積形式ノードとなる
- この性質を用いて局所平衡方程式を導くやりかたもある

方程式の意味するもの

- 積形式解は方程式を解いて得られるのではなく，予想し確認するものである
- その助けとして，局所平衡方程式をはじめ大域平衡方程式に関する各種の十分条件が使われる
- とまあ，ここまでは公式見解だが，実際のところ，具体的なモデルに関して積形式解を見出すには必ずしもそのような手順を踏んでいるわけではなく，解は別な方法で見出し，それを方程式に代入して確かめることが多い
- その際，局所平衡方程式に代入して確かめるのも大域平衡方程式に代入して確かめるのも，計算の手間は実はほぼ同じ程度である
- したがって，局所平衡方程式をはじめとする各種平衡方程式の意義は，対象としているモデルにおいて積形式解が得られた後に，平衡を保つのはどの量なのかを明示的に知り，モデルの動きに関する理解と洞察を深めることができる，ということにある（紀 ローカル)

積形式解をもつ待ち行列網

お話の流れ

- **積形式網は積形式ノードを素材として構成 $\Rightarrow$ 網の状態表現**
- **積形式解をもう一度**
- **トラヒック方程式**
- **分離型到着関数と分離型容量係数**
- **各種の周辺分布**

網の状態表現

ノード j の状態：$(\boldsymbol{c}_j, \boldsymbol{y}_j)$

位置	1	2		n
$\boldsymbol{c}_{j\ell}$	(m_{j1}, k_{j1})	(m_{j2}, k_{j2})	$\ldots$	(m_{jn}, k_{jn})
残りサービス要求時間	y_{j1}	y_{j2}	$\ldots$	y_{jn}

$\boldsymbol{c}_j = (\boldsymbol{c}_{j1}, \boldsymbol{c}_{j2}, \ldots, \boldsymbol{c}_{jn})$　　$\boldsymbol{c}_{j\ell} = (m_{j\ell}, k_{j\ell})$

$(m_{j\ell}, k_{j\ell})$ =(位置 ℓ の客の連鎖番号, 位置 ℓ の客のクラス番号)

$$\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_N) \quad \boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N)$$

- **ノード，位置，クラス，連鎖 に関する情報が必要（添字の山！）**
- **網の状態表現**　$(\boldsymbol{c}, \boldsymbol{y})$

積形式解（原型）をもう一度

$$
\begin{aligned}
P(\boldsymbol{c},\boldsymbol{y}) &= P(\boldsymbol{c})P(\boldsymbol{y}|\boldsymbol{c}) \\
P(\boldsymbol{c}) &= C\Lambda(\boldsymbol{m})\Phi(\boldsymbol{u})\prod_{j=1}^{N}\tau_j(\boldsymbol{c}_j) \\
\tau_j(\boldsymbol{c}_j) &= \prod_{\ell=1}^{|\boldsymbol{c}_j|}\sigma_j(m_{j\ell},k_{j\ell}) \\
\sigma_j(m_{j\ell},k_{j\ell}) &= \frac{\theta_j(m_{j\ell},k_{j\ell})}{\mu_{k_{j\ell}}} \\
P(\boldsymbol{y}|\boldsymbol{c}) &= \prod_{j=1}^{N}\prod_{\ell=1}^{|\boldsymbol{c}_j|}\mu_{k_{j\ell}}\bigl(1-F_{k_{j\ell}}(y_{j\ell})\bigr).
\end{aligned}
$$

トラヒック方程式(1)

- $m_{j\ell}$: ノード j, 位置 ℓ に滞在する客の連鎖番号
- $k_{j\ell}$: ノード j, 位置 ℓ に滞在する客のクラス番号
- $\theta_j(m_{j\ell}, k_{j\ell})$: $(m_{j\ell}, k_{j\ell})$ の客の相対訪問回数
- 相対訪問回数は経路情報を係数行列とする連立一次方程式（トラヒック方程式）の解として定まる
- 部分連鎖 m に関する相対訪問回数ベクトル

$$\boldsymbol{\theta}_m = (\boldsymbol{\theta}_{m1}, \boldsymbol{\theta}_{m2}, \ldots, \boldsymbol{\theta}_{mN})$$

および

$$\boldsymbol{\theta}_{mi} = \big(\theta_i(m,1), \theta_i(m,2), \ldots, \theta_i(m,K)\big), \quad i = 1, 2, \ldots, N$$

- $\theta_i(m,k)$: ノード i における連鎖 m, クラス k の客の相対訪問回数

トラヒック方程式(2)

- $r_m\big((i,k),(j,h)\big)$:**ノード i でクラス k の客としてサービスを受けた連鎖 m の客が，ノード j にサービスクラス h の客としてつく確率**
- **閉鎖型部分連鎖 m の経路行列**

$$\boldsymbol{R}(m) = \begin{pmatrix} R_m(1,1) & R_m(1,2) & \dots & R_m(1,N) \\ R_m(2,1) & R_m(2,2) & \dots & R_m(2,N) \\ f & \dots\dots\dots\dots & & \\ R_m(N,1) & R_m(N,2) & \dots & R_m(N,N) \end{pmatrix}$$

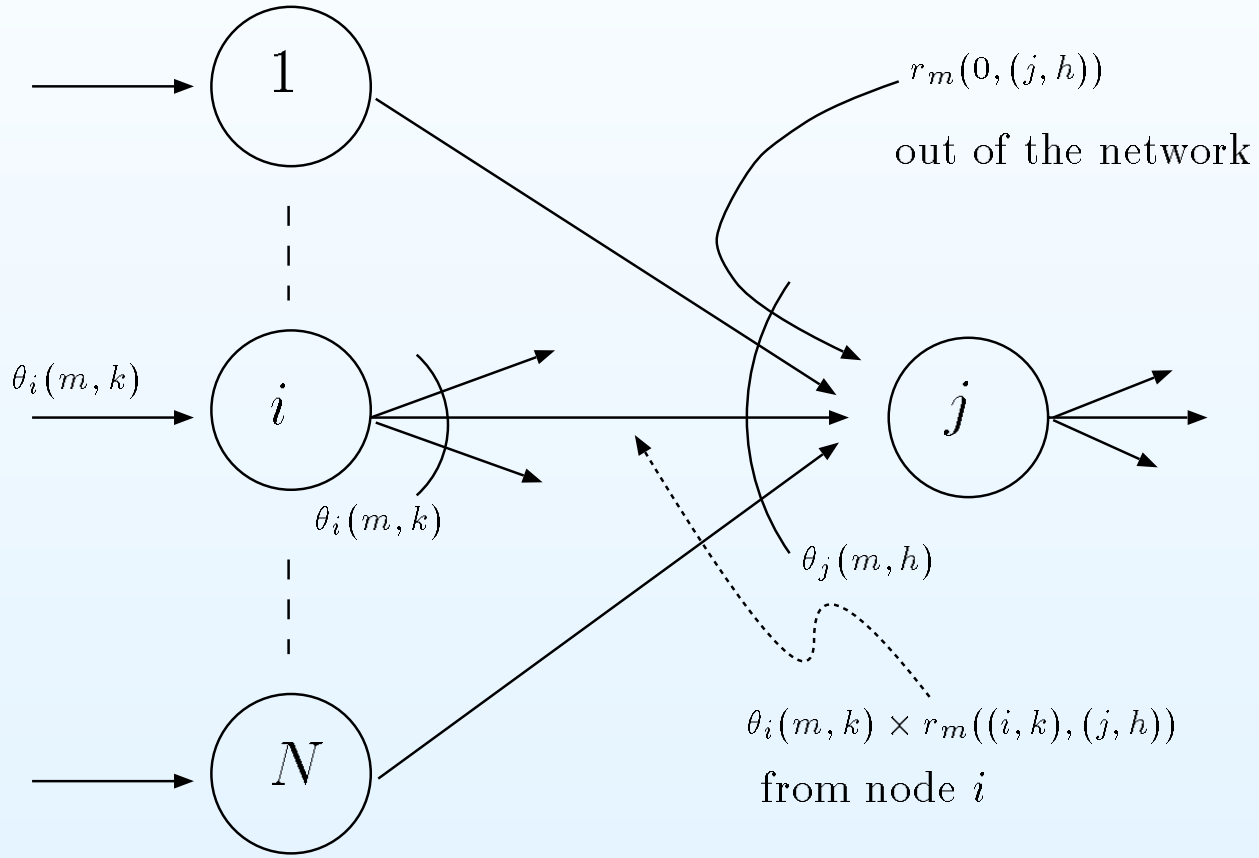
- **閉鎖型部分連鎖 m に関するトラヒック方程式（同次方程式）**

$$\boldsymbol{\theta}_m = \boldsymbol{\theta}_m \boldsymbol{R}(m)$$

- **開放型の場合も外部（ノード 0）を考慮し同様にトラヒック方程式を作れる**

$$(1, \boldsymbol{\theta}_m) = (1, \boldsymbol{\theta}_m)\boldsymbol{R}(m)$$

トラヒック方程式(3)


1
i
N
j
$\theta_i(m,k)$
$\theta_i(m,k)$
$r_m(0,(j,h))$
out of the network
$\theta_j(m,h)$
$\theta_i(m,k) \times r_m((i,k),(j,h))$
from node i

分離型到着関数と分離型容量係数

- **分離型到着関数**
  - **状態 m のときの連鎖 m の客の到着率**

$$\lambda_m(\boldsymbol{m}) = \frac{\Lambda(\boldsymbol{m} + \boldsymbol{e}(\boldsymbol{m}))}{\Lambda(\boldsymbol{m})}$$

  - $\Lambda(\boldsymbol{m}) = \Lambda_1(m_1)\Lambda_2(m_2)\ldots\Lambda_M(m_M)$
- **分離型容量係数**
  - u_{jmk}: **ノード j に滞在する連鎖 m, クラス k の客数**
  - x_{jm}: **ノード j に滞在する連鎖 m の客数**
  - $\boldsymbol{u} = \left(\left((u_{jmk})_{k=1}^{K} \right)_{m=1}^{M} \right)_{j=1}^{N}, \quad \boldsymbol{x} = \left((x_{jm})_{m=1}^{M} \right)_{j=1}^{N}$
  - $\Phi(\boldsymbol{u}) = \Phi_1(\boldsymbol{u}_1)\Phi_2(\boldsymbol{u}_2)\cdots\Phi_N(\boldsymbol{u}_N)$, **$\boldsymbol{u}$ に関する分離形**
  - $\Phi(\boldsymbol{x}) = \Phi_1(\boldsymbol{x}_1)\Phi_2(\boldsymbol{x}_2)\cdots\Phi_N(\boldsymbol{x}_N)$, **$\boldsymbol{x}$ に関する分離形**

各種の周辺分布

- 状態 (c, y) に関する積形式（原型）は状態が細かすぎる
- さらに到着関数や容量係数の一般形も実用的には一般的過ぎる
- ということで，容量係数や到着関数を分離形として，積形式解（原型）の周辺分布を計算することで，使いやすい形を導く
  $\Rightarrow$ ここから先は計算のみ
- ノードと連鎖は実用上大切，クラスは無視してもよい
  $\Rightarrow$ 網の状態として $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jM})$
  x_{jm}: ノード j に滞在する連鎖 m の客数
- 簡単のため，閉鎖型 M 連鎖網を例にとり紹介する
- 周辺分布の計算に際しては，多項定理（ベクトルバージョン）を使いまくる

ベクトルの表記法とベクトル版多項定理

- $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, $\boldsymbol{\rho} = (\rho_1, \rho_2, \ldots, \rho_n)$ **について以下を定義**
  - $$|\boldsymbol{x}| = n, \quad \|\boldsymbol{x}\| = x_1 + x_2 + \cdots + x_n$$
  - $$\boldsymbol{x}! = x_1! x_2! \cdots x_n! \quad \boldsymbol{\rho}^{\boldsymbol{x}} = \rho_1^{x_1} \rho_2^{x_2} \cdots \rho_n^{x_n}$$
- **多項定理（一般表現）**

$$(a_1 + a_2 + \cdots + a_n)^m = \sum_{x_1 + x_2 + \cdots + x_n = m} \frac{m!}{x_1! x_2! \cdots x_n!} a_1^{x_1} a_2^{x_2} \cdots a_n^{x_n}$$

- **多項定理（ベクトル版）**

$$\|\boldsymbol{a}\|^m = \sum_{\|\boldsymbol{x}\| = m} \frac{\|\boldsymbol{x}\|!}{\boldsymbol{x}!} \boldsymbol{a}^{\boldsymbol{x}}$$

標準的な閉鎖型待ち行列網の周辺分布 (1)

- **状態** $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ **に関する周辺分布**
  - $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jM})$, $j = 1, 2, \ldots, N$
  - x_{jm}: **ノード** j **に滞在する連鎖** m **の客数**
- **客数ベクトル：** $\boldsymbol{K} = (K_1, K_2, \ldots, K_M)$, K_m: **連鎖** m **の網内客数**
- **分離形をもつ容量係数**
  - **ノードごとの分離形 (滞在客数のみに依存する)**

$$\Phi(\boldsymbol{x}) = \Phi_1(\|\boldsymbol{x}_1\|)\Phi_2(\|\boldsymbol{x}_2\|)\cdots\Phi_N(\|\boldsymbol{x}_N\|)$$

  - **ノード** j **において滞在客数が** n **のときのサービス率**

$$\nu_j(n) = \frac{\Phi_j(n-1)}{\Phi_j(n)}$$

- $\Phi_j(n) = \dfrac{1}{\nu_j(1)\nu_j(2)\cdots\nu_j(n)}, \qquad \Phi_j(0) = 1$

標準的な閉鎖型待ち行列網の周辺分布 (2)

- **分離形をもつ容量係数（続き）**
  - **単一サーバ：** $\nu_j(n) = \nu_j(1), n = 1, 2, \ldots$
  - **無限サーバ：** $\nu_j(n) = n\nu_j(1), n = 1, 2, \ldots$
  - **複数サーバ** (S_j):

$$\nu_j(n) = \begin{cases} n\nu_j(1), & n \leq S_j \\ S_j\nu_j(1), & S_j < n \end{cases}$$

- **トラヒック係数**：$\boldsymbol{\rho} = (\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \ldots, \boldsymbol{\rho}_N)$
  - $\boldsymbol{\rho}_j = (\rho_{j1}, \rho_{j2}, \ldots, \rho_{jM}), j = 1, 2, \ldots, N$
  - $\rho_{jm} = \sum_{k=1}^{K} \sigma_j(m,k), \quad \sigma_j(m,k) = \dfrac{\theta_j(m,k)}{\mu_k}$

標準的な閉鎖型待ち行列網の周辺分布 (3)

- **積形式解 (周辺分布)**

$$P(\boldsymbol{x}) = \frac{1}{G(\boldsymbol{K})} \prod_{j=1}^{N} \Phi_j(\boldsymbol{x}_j)\varphi_j(\boldsymbol{x}_j), \quad \varphi_j(\boldsymbol{x}_j) = \frac{\|\boldsymbol{x}_j\|!}{\boldsymbol{x}_j!}\boldsymbol{\rho}_j^{\boldsymbol{x}_j}$$

- $G(\boldsymbol{K})$: **正規化定数**

$$G(\boldsymbol{K}) = \sum_{\boldsymbol{x}_1+\boldsymbol{x}_2+\cdots+\boldsymbol{x}_N=\boldsymbol{K}} \prod_{j=1}^{N} \Phi_j(\boldsymbol{x}_j)\varphi_j(\boldsymbol{x}_j)$$

  - $G(\boldsymbol{K})$ **は確率条件** $\sum_{\boldsymbol{x}} P(\boldsymbol{x}) = 1$ **から定まる**
  - $N = 10, M = 3, \boldsymbol{K} = (5,5,5)$ **では約 80 億項からの和**
  - **正規化定数の効率的な計算法は応用上の大きな課題**

混合型待ち行列網の周辺分布

- **積形式解 (周辺分布)**

$$P(\boldsymbol{x}) = \frac{1}{G(\boldsymbol{K})} \prod_{j=1}^{N} q_j(\boldsymbol{x}_j)$$

-

$$\begin{aligned} q_j(\boldsymbol{x}_j) &= \Phi_j(\|\boldsymbol{x}_j\|) \frac{\|\boldsymbol{x}_j\|!}{\boldsymbol{x}_j!} \boldsymbol{\rho}_j^{\boldsymbol{x}_j} \boldsymbol{\lambda}^{\boldsymbol{x}_j^o} \\ &= \Phi_j(\|\boldsymbol{x}_j^c\| + \|\boldsymbol{x}_j^o\|) \frac{(\|\boldsymbol{x}_j^c\| + \|\boldsymbol{x}_j^o\|)!}{\boldsymbol{x}_j^c! \, \boldsymbol{x}_j^o!} (\boldsymbol{\rho}_j^c)^{\boldsymbol{x}_j^c} (\boldsymbol{\rho}_j^o)^{\boldsymbol{x}_j^o} \boldsymbol{\lambda}^{\boldsymbol{x}_j^o} \end{aligned}$$

- $G(\boldsymbol{K})$: **正規化定数**

待ち行列網の計算法

概要

- 待ち行列網を実際に利用するためには計算法の開発が必要
- ２種類の 計算法：
  - たたみこみ (convolution) 法：たたみこみ演算により，正規化定数を直接計算. 指数部のあふれをおこしやすい
  - MVA(Mean Value Analysis): ノード数と客数に関する漸化式を手がかりに計算．仮数部の桁落ちをおこしやすい
- 計算法に関する論文は多いが，いずれもどちらかの方法の改善・改良版か，両者の融合版に分類できる
- 紀は「たたみこみ法」の方がシンプルなので好きだ！

たたみこみ法

概要

- **たたみこみ演算を用いて正規化定数 $G(K)$ を直接計算する**
- **たたみこみ演算を素朴に適用するとひどいめにあう**
- **単一サーバ，複数サーバ，無限サーバについては高速化技法を使う**
- **混合型待ち行列網の場合には，たたみこみ領域が有界にならないので，一工夫が必要になる**
- **指数部問題** (over flow/under flow) **が発生しやすい**
- **指数部問題にはスケーリングで対応するが万全ではない**

たたみこみ演算

- $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$: **非負の整数値を要素とするベクトル**
- $a(\boldsymbol{x})$ **および** $b(\boldsymbol{x})$ **:** $\boldsymbol{x}$ **の関数**
- $c(\boldsymbol{x}) = (a * b)(\boldsymbol{x})$: a, b **のたたみこみ**

$$
\begin{aligned}
c(\boldsymbol{x}) &= (a * b)(\boldsymbol{x}) \\
&= \sum_{\mathbf{0} \le \boldsymbol{i} \le \boldsymbol{x}} a(\boldsymbol{x} - \boldsymbol{i}) \cdot b(\boldsymbol{i}) \\
&= \sum_{i_1=0}^{x_1} \sum_{i_2=0}^{x_2} \cdots \sum_{i_n=0}^{x_n} a(x_1 - i_1, x_2 - i_2, \cdots, x_n - i_n) \cdot b(i_1, i_2, \cdots, i_n)
\end{aligned}
$$

  - **結合則，交換則が成り立つ**

$$(a * b)(\boldsymbol{x}) = (b * a)(\boldsymbol{x}), \quad ((a * b) * c)(\boldsymbol{x}) = (a * (b * c))(\boldsymbol{x})$$

- $(a_1 * a_2 * \cdots * a_n)(\boldsymbol{x})$: $a_1, a_2, \ldots, a_n$ **に関するたたみこみ**

たたみこみによる正規化定数の表現

- M **連鎖，客数** $\boldsymbol{K}$**，閉鎖型網の積形式解**

$$P(\boldsymbol{x}) = \frac{1}{G(\boldsymbol{K})} \prod_{j=1}^{N} q_j(\boldsymbol{x}_j), \quad q_j(\boldsymbol{x}_j) = \Phi_j(\|\boldsymbol{x}_j\|) \frac{\|\boldsymbol{x}_j\|!}{\boldsymbol{x}_j!} \boldsymbol{\rho}_j^{\boldsymbol{x}_j}$$

- **正規化定数：**

$$G(\boldsymbol{K}) = \sum_{\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_N = \boldsymbol{K}} \prod_{j=1}^{N} q_j(\boldsymbol{x}_j)$$

- **正規化定数のたたみこみ表現：**

$$G(\boldsymbol{K}) = (q_1 * q_2 * \cdots * q_N)(\boldsymbol{K})$$

補完網

- i-補完網：ノード i を「取り除いた」網
  - 「取り除く」：ノード i でのサービス要求時間を 0 とする
  - その他の条件は何も変えない
  - あたかもノード i が存在しなかったかのような網
  - $\boldsymbol{x}_{[i]} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1} \ldots, \boldsymbol{x}_N)$：$i$-補完網の状態ベクトル
- i-補完網に関する積形式解と正規化定数：

$$P(\boldsymbol{x}_{[i]}) = \frac{1}{G_{[i]}(\boldsymbol{K})} \prod_{\substack{j=1 \\ j \neq i}}^{N} q_j(\boldsymbol{x}_j), \quad q_j(\boldsymbol{x}_j) = \Phi_j(\|\boldsymbol{x}_j\|) \frac{\|\boldsymbol{x}_j\|!}{\boldsymbol{x}_j!} \boldsymbol{\rho}_j^{\boldsymbol{x}_j}$$

$$G(\boldsymbol{K}) = \sum_{\boldsymbol{x}_1 + \cdots \boldsymbol{x}_{i-1} + \boldsymbol{x}_{i+1} + \cdots + \boldsymbol{x}_N = \boldsymbol{K}} \prod_{\substack{j=1 \\ j \neq i}}^{N} q_j(\boldsymbol{x}_j)$$

- $(i_1, i_2, \ldots, i_m)$-補完網も同様に定義される

正規化定数のたたみこみ表現：証明

- x_1 **に着目**:

$$G(\boldsymbol{K}) = \sum_{\mathbf{0} \le \boldsymbol{x}_1 \le \boldsymbol{K}} q_1(\boldsymbol{x}_1) \sum_{\boldsymbol{x}_2 + \cdots + \boldsymbol{x}_N = \boldsymbol{K} - \boldsymbol{x}_1} \prod_{j=2}^{N} q_j(\boldsymbol{x}_j)$$

- **上式に次式を代入：** $G_{[1]}(\boldsymbol{K} - \boldsymbol{x}_1) = \sum_{\boldsymbol{x}_2 + \cdots + \boldsymbol{x}_N = \boldsymbol{K} - \boldsymbol{x}_1} \prod_{j=2}^{N} q_j(\boldsymbol{x}_j)$

- **以下を得る：** 1-**補完網と** q_1 **のたたみこみ**

$$G(\boldsymbol{K}) = \sum_{\mathbf{0} \le \boldsymbol{x}_1 \le \boldsymbol{K}} q_1(\boldsymbol{x}_1) G_{[1]}(\boldsymbol{K} - \boldsymbol{x}_1) = (q_1 * G_{[1]})(\boldsymbol{K})$$

- **この手続きを繰り返していき以下を得る**

$$G(\boldsymbol{K}) = (q_1 * q_2 * G_{[1,2]})(\boldsymbol{K}) = \cdots = (q_1 * q_2 * \cdots * q_N)(\boldsymbol{K})$$

基本アルゴリズム A

A1: **初期化**: $\mathbf{0} \le \boldsymbol{x} \le \boldsymbol{K}$ **なるすべての** $\boldsymbol{x} = (x_1, x_2, \ldots, x_M)$ **について**

$$G(\boldsymbol{x}) \leftarrow \begin{cases} 1 & \boldsymbol{x} = \mathbf{0} \\ 0 & \boldsymbol{x} \neq \mathbf{0} \end{cases}$$

A2: $j = 1, 2, \cdots, N$ **について A3, A4, A5 を実行**

A3: $q_j(\boldsymbol{x}_j)$ **を作成**: $\mathbf{0} \le \boldsymbol{x} \le \boldsymbol{K}$ **なるすべての** $\boldsymbol{x}$ **について**

$$q(\boldsymbol{x}) \leftarrow \Phi_j(\|\boldsymbol{x}\|) \frac{\|\boldsymbol{x}\|\,!}{\boldsymbol{x}\,!} \boldsymbol{\rho}_j^{\boldsymbol{x}}$$

A4: **たたみこみ**: $k = \|\boldsymbol{K}\|, \cdots, 1, 0$ **についてこの順序で A5 を実行**

A5: $\|\boldsymbol{x}\| = k, \mathbf{0} \le \boldsymbol{x} \le \boldsymbol{K}$ **なるすべての** $\boldsymbol{x}$ **について**

$$G(\boldsymbol{x}) \leftarrow \sum_{\mathbf{0} \le \boldsymbol{y} \le \boldsymbol{x}} G(\boldsymbol{y}) q(\boldsymbol{x} - \boldsymbol{y})\ .$$

高速化技法（例：単一サーバ）

- **アルゴリズム A を素朴に実装してはいけない**
- **単一サーバノードの場合以下の関係が成り立つ**

$$G(\boldsymbol{x}) = G_{[j]}(\boldsymbol{x}) + \sum_{m=1}^{M} \rho_{jm} G(\boldsymbol{x} - \boldsymbol{e}(m))$$

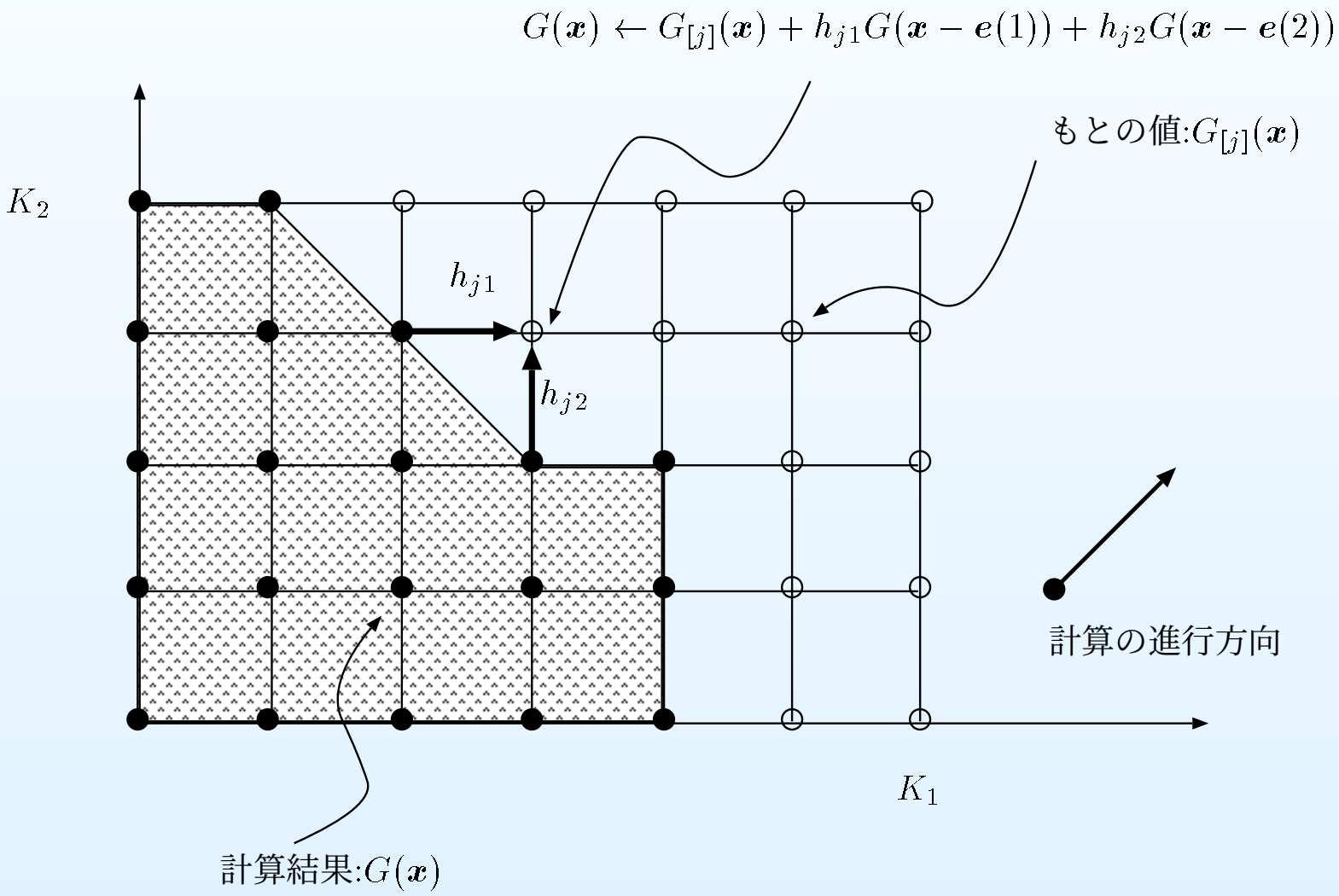
- **この関係を利用し以下のアルゴリズム S を A3,A4,A5 に変える**
- **アルゴリズム S**

S1: $k = 1, 2, \cdots, \|\boldsymbol{K}\|$ **についてこの順序で S2 を実行**

S2: $\|\boldsymbol{x}\| = k, \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{K}$ **なるすべての** $\boldsymbol{x}$ **について**

$$G(\boldsymbol{x}) \leftarrow G(\boldsymbol{x}) + \sum_{m=1}^{M} \rho_{jm} G(\boldsymbol{x} - \boldsymbol{e}(m)) \ .$$

単一サーバノードの計算



待ち行列網の計算法： まとめ

- 高速化技法は実用化には欠かせない
- 使用率（呼量）等はたたみこみ終了段階で直ちに得られる
- ノード j の平均滞在客数（平均待ち時間）を求めるためには，$G_{[j]}(x), 0 \leq x \leq K$ が必要
- j が単一サーバ型または無限サーバ型の場合には $G_{[j]}(x)$ は $G(x)$ から簡単に得られるが，複数サーバ型ノードの場合には工夫が必要
- 複数サーバ型ノードの場合には，$G(x)$ からたたみこみ逆変換により $G_{[j]}(x)$ を求めるアルゴリズムをつくることはできる
- しかし，このアルゴリズムは数値計算上不安定である
- 複数サーバノードの場合，結局そのノード以外のノードをすべてたたみこむ，という操作をすべての複数サーバノードについて行うのが一番簡単ではやい
- 混合型網の場合には，たたみこみ領域を有界とするために閉鎖型網に関する周辺分布を計算しアルゴリズムをつくることができるが，形は複雑となる

待ち行列網の応用

お話の概要

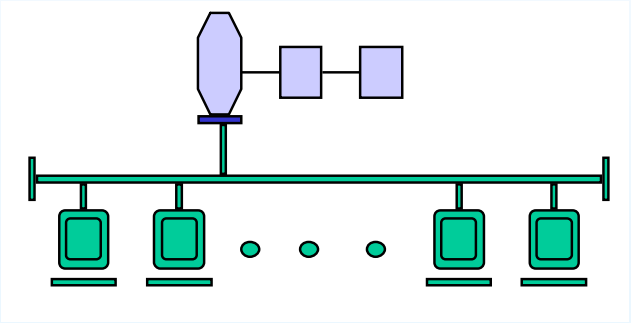
- コンピュータシステム性能評価の特徴
- 性能評価をめぐる環境
- シングルプロファイル法
- まとめ

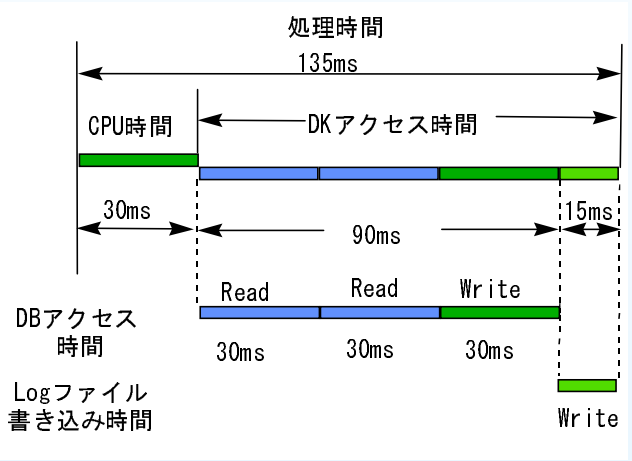
コンピュータシステム性能評価の特徴

- 通信システム性能評価に比べて歴史が浅い $\Rightarrow$ 技術の普及度が低い
- 性能評価指標や重点項目が異なる．最大の技術課題は
  - マルチプログラミングの多重度問題に応えられること
  - 性質の異なる複数のサブシステムに対応できること
    $\Rightarrow$ 複数連鎖（混合型）があつかえること
- 多様な個別システムへの短期間，少工数，迅速な対応が求められる
- コンピュータシステムモデル化の重要な要素
  - システム性能は単体性能の延長上では捉えられない
  - コンピュータシステムには I/O 系が存在する
  - 重要性の順位は（1） 構造 （2） 制御 （3） 分布
- 性能評価精度は過剰であってはならない
  - コストをかければ評価精度はあがる
  - しかし，設計者はある判断をするために性能評価を行う
  - したがって「結論を間違えない」程度の精度があればよい
- システム性能評価は保険：評価コスト $\ll$ トラブル処理コスト

マルチプログラミングの多重度問題

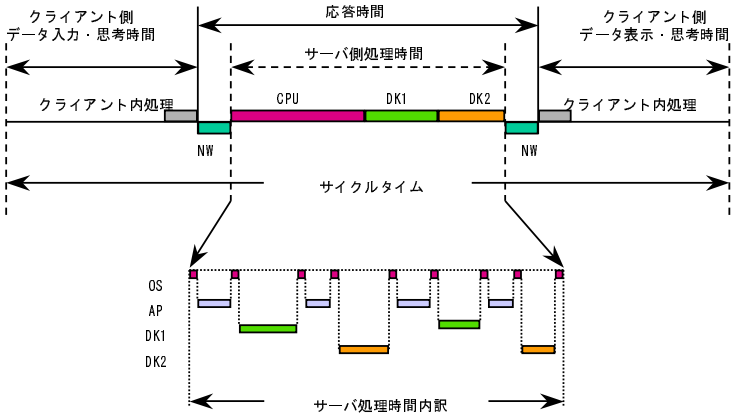
- 多重度問題：システム構成とシングルプロファイルが与えられたとき，それを M 多重で稼動したなら，処理時間（競合による待ち時間を含む）はどの程度になるか？

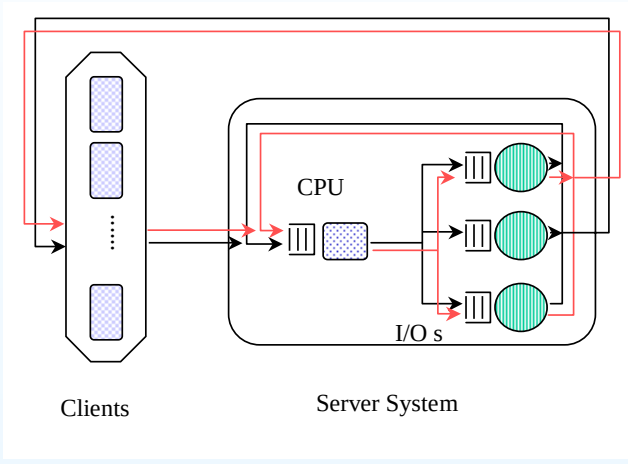


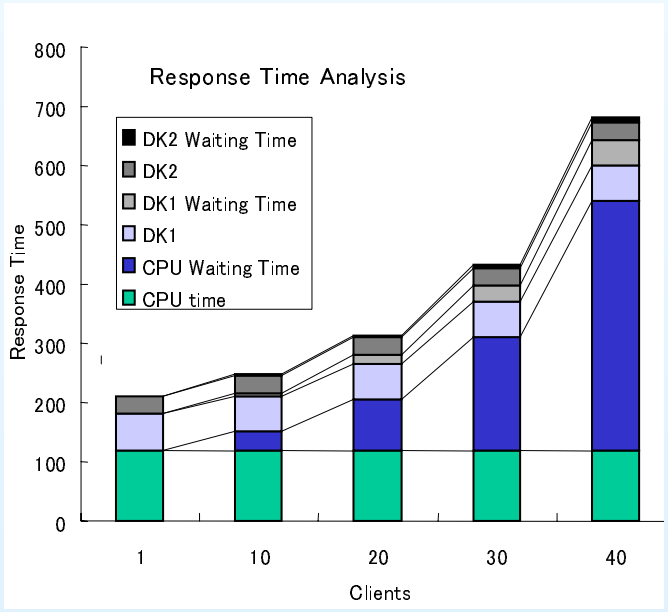


- シングルプロファイル：処理の単位当りの資源使用時間
  - シングルプロファイル例：(CPU,DB,Log)=(30,90,15)=ρ
  - トラヒック方程式を解く必要はない
- マルチプログラミング多重度：同時並列的に実行される処理数
- 処理単位：ジョブ，トランザクション，他

セントラルサーバモデル







待ち行列網モデルの特徴

- I/O 系を含むシステム構造のモデル化に適している
- ロバストネス（モデルの頑強性）
  - 条件の変化に対する結果の追随性のよさ
- 経済性のよさ
  - プログラムレスのモデル化
  - 計算実行時間の短縮
- 柔軟性と簡易性
  - ソフトウエアパッケージ化による簡便性
  - 待ち行列理論を知らなくても使える

性能評価事故例（公表されたもの）

- **動かないコンピュータ：国会図書館**
  - **ピーク時の処理能力を読み違え実用化目前にダウン**
  - **日経コンピュータ，**1987/2/2.
- **続発するオンライン事故：大量トランザクションゆえの落とし穴**
  - **郵貯，東証，都銀などにみるシステム技術の現状と限界**
  - **日経コンピュータ** 1988/7/18.
- **動かないコンピュータ：博報堂**
  - **レスポンスの悪さに耐えられず** C/S **基幹システムを再度延期**
  - **日経コンピュータ** 1994/8/22.
- **創刊** 400 **号記念 動かないコンピュータ：**
  - **続発する** C/S **システムのトラブル**
  - **日経コンピュータ** 19996/9/16.
- **当然のことながら，公表されていない例はこの＊＊＊倍はある**

性能評価問題はなぜ多発するのか

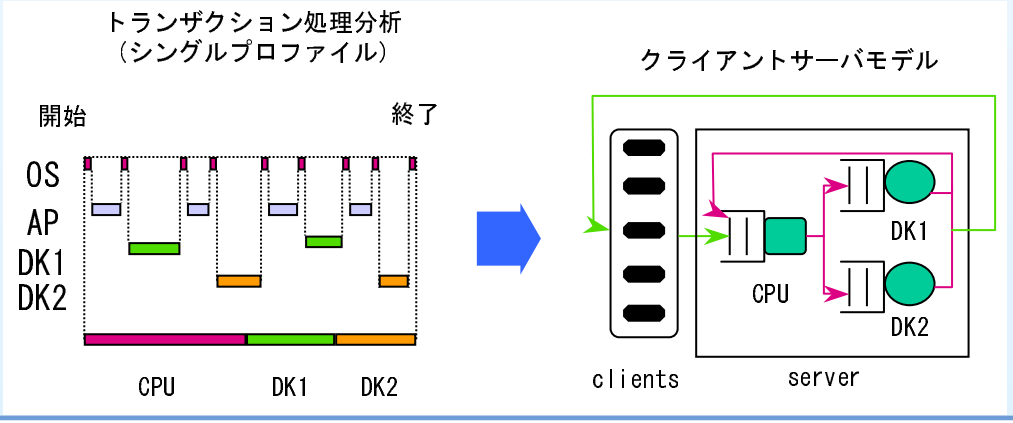
- 契約時点での経験と勘に頼った見積もり
- プロジェクトの中での位置づけの不明確さ
  - 問題が発生してから対応を考え出す
  - 性能問題の軽視，担当者がアサインされていない
- 性能評価技術をみにつけた SE が少なくなってきている
  - 忙しくて勉強できない
  - 教科書，教育コースも不十分，勉強のしかたがわからない
  - 他の IT 技術に比べて数学の勉強が少々必要 $\Rightarrow$ 参入障壁
- 技術的問題点
  - システム性能評価の実践的方法論がまだ未成熟
  - 測定ツール，測定法も未成熟：山のようなクズデータ
  - 作業用ツールの開発も断片的で統一性がない

性能評価をめぐる環境 (紀が会社にいたころ)

- 汎用機の時代からオープンシステムの時代へ
- マルチベンダ化，ダウンサイジング
  - 自社開発ではない製品の性能分析が必要
  - 性能データが入手困難
  - 価格競争の激化，受注価格の低下
  - 性能評価工数の削減 $\Rightarrow$ より少工数，短期間
  - SE はより少ない工数でより質の高い設計，評価を求められる
- 性能トラブルの多発（これは昔もそうだったが）
  - 汎用機の技術がそのままでは使えない
  - 従来の「経験と勘」が通用しない
  - あともどり工数の発生による収益性の低下
- 何が求められていたか
  - 性能評価の方法論の明確化
  - それを支える技術の確立
  - 方法論の効果的実施のためのツール群の開発

シングルプロファイル法

- 紀のグループが提案・展開した社内向け性能評価技法
  - 解析技術と測定技術の組み合わせ
  - 方法論の明確化とマニュアル化 ⇒ 新人でも 70 点の評価を
  - 作業支援ツールの開発（測定器＋計算パッケージ）
- スローガン：「１を聞いてｎを知る」
  - １を聞く：１クライアント接続時の測定
    - 無競合時での実測による基礎データの測定
    - 待ち時間は決して測ってはならない
  - ｎを知る：待ち行列網モデルによる性能予測
    - ｎクライアント接続時のシステム性能予測



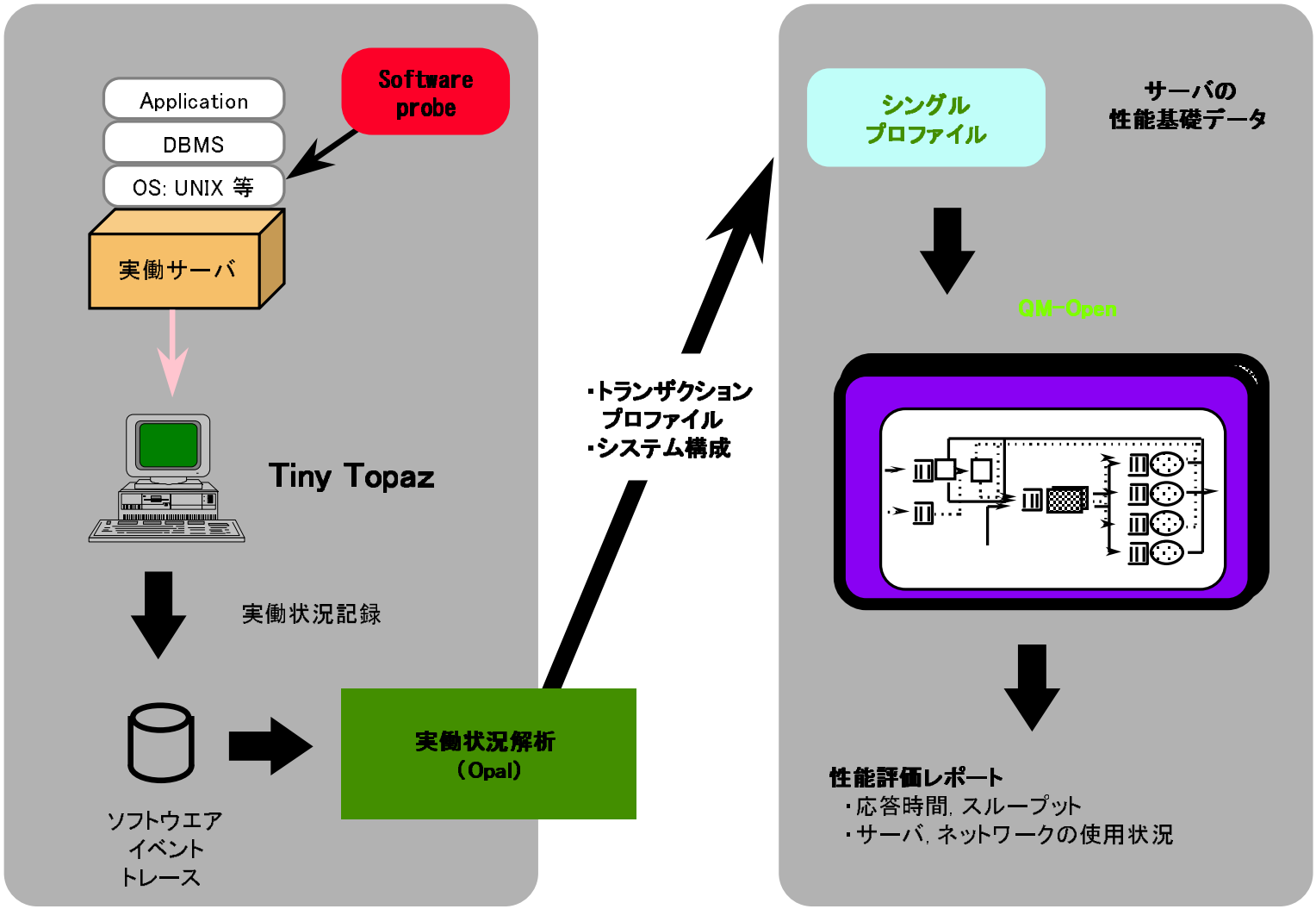
シングルプロファイル法の開発方針

- 測定を基本にすえる
  - 測らなければ何も分からない
  - しかし，やみくもに測定してはならない
  - 最小限の測定で予測につながるデータを
- 待ち行列網モデルの活用
  - 最小工数で的確な性能評価
  - 技術的蓄積が豊富である（QM-X)
- 検証実験の充実
  - 数理的モデルにつきまとう不安の解消
    $\Rightarrow$ 事業部に受け入れてもらうため

シングルプロファイル法の開発の背景

- 開発方針には必然性がある
- システムのブラックボックス化
  - 内部動作，動作原理の隠蔽
  - 性能データの透明度低下
  - 測らなければなにもわからない
- 測定環境の進展（昔ほど大変ではなくなった）
  - PC, WS の普及
  - プロトタイピング技術の進歩
- システム価格の下落
  - 性能評価に多くの時間と工数はさけない
  - 大規模な測定実験はもうできない（やるべきではない）

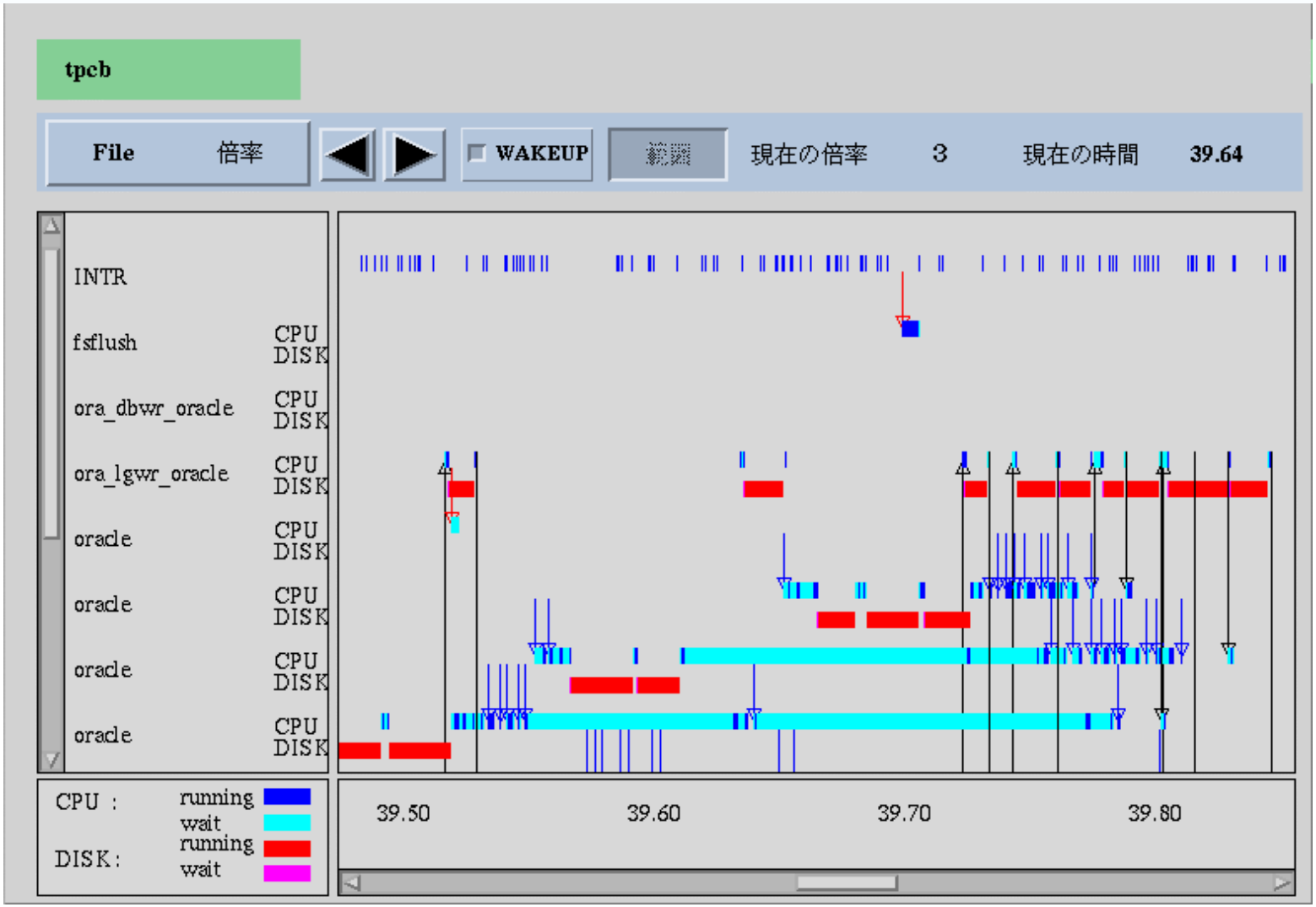
Tiny Topaz と QM-Open



Tiny Topaz

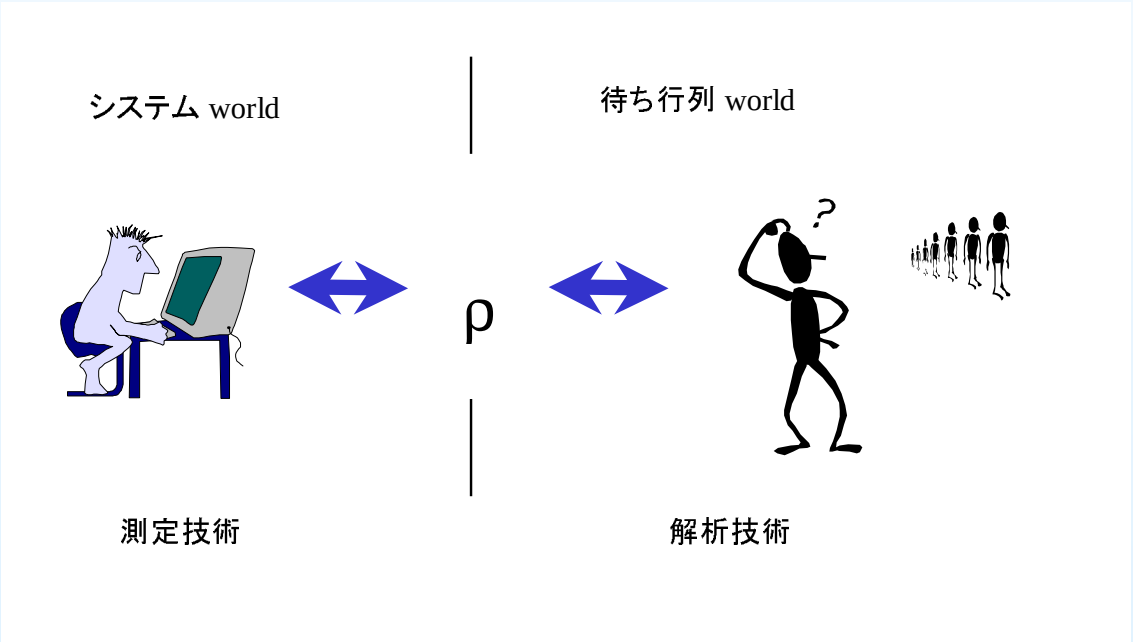
- Dynamic Hook **方式ソフトウエアプローブ**
- **マルチプロセッサ対応**
- **データ解析ソフト** Opal

Tiny Topaz 測定例



技術の選択

- 状況にあった技術の最適な組み合わせ
- 測定技術者は解析技術に疎い $\Leftrightarrow$ 解析技術者は測定技術に疎い
- 両者に分かる共通言語は ρ である！
- 測定技術者は測定結果として，解析技術者は基礎データとして ρ を理解する
- ρ に名前をつけよう！$\Rightarrow$ シングルプロファイル



待ち行列網の応用のまとめ

- 待ち行列網モデルは確かに実用になった
  - 使用実績，結果の比較等々のデータは省略（NEC)
- 測定技術と協力することではじめて「机上の空論」から脱却できた
- 展開のしかたには工夫が必要
  - 待ち行列網の計算パッケージを売る方針は失敗する
  - ユーザに直接待ち行列網を触らせるな！
  - ソリューションを売る（待ち行列網モデルは自分たちで使う）
- どのような技術を組み合わせるのか，どのようにユーザにアピールしていくのかについては性能評価の現場経験が役にたつ

まとめ

- 待ち行列網は理論的にも応用上も結構奥の深いものがある
- 初学者はまずは応用から入るとよいのではないだろうか
  - 計算ソフトの開発は簡単（卒研レベル）
  - 使ってみると応用の可能性が見え，理論への興味がわく
- 待ち行列網は複雑ではあるが難しくはない
  ⇒ 数式はたくさん出てくるが数学はあまり出てこない
- 待ち行列（網）理論は一人では生きていけない
  - 良きパートナ（応用）に恵まれてこそ発展できる
  - パートナ探しは研究そのものよりも大変である
  - 可能性を感じさせる研究テーマ
- 理論的な確かさと応用への誠実な結びつきが研究分野の未来を開く
- 若い研究者が新しい扉を開く ⇒ 若い人，頑張ってください．
- I thank you for your attention.