修士学位論文

並列分散処理システムに関する研究

東北大学大学院工学研究科
電気・通信工学専攻

鶴田 進

目 次

第 1 章　序論 **1**

1.1　本研究の背景 1

1.2　本研究の目的 3

1.3　本論文の構成 4

第 2 章　負荷予測 **5**

2.1　はじめに 5

2.2　線形予測モデル 7

2.2.1　負荷の確率分布特性 7

2.2.2　負荷の変動期待値 8

2.2.3　負荷の変動期待値の不確実さ 8

2.3　カオス予測モデル 10

2.3.1　カオス概論 10

2.3.2　カオス予測 11

2.4　負荷予測実験 15

2.4.1　実験条件 15

2.4.2　予測結果 16

2.5　まとめ 21

第 3 章　動的負荷分散アルゴリズム **22**

3.1　はじめに 22

3.2　従来提案されている動的負荷分散アルゴリズム 23

3.3　分散処理システムモデル 25

3.3.1 分散システム 25
3.3.2 分散処理モニタ 26
3.4 予測を用いた処理計算機選択アルゴリズム 30
3.4.1 単純な予測選択アルゴリズム (pre) 30
3.4.2 履歴をもちいた予測選択アルゴリズム (his) 31
3.5 シミュレーション 32
3.5.1 シミュレーション条件 32
3.5.2 性能評価 33
3.6 まとめ 39

第 4 章 自律的仮想分散システム環境 40
4.1 はじめに 40
4.2 分散処理モニタの改良 42
4.3 仮想分散システム環境の形成 45
4.3.1 結合度の導入 45
4.4 シミュレーション 50
4.4.1 シミュレーション条件 50
4.4.2 性能評価 51
4.5 まとめ 58

第 5 章 結論・検討 59
5.1 結論 59
5.2 検討 60

謝辞 61

参考文献 62

研究業績 64

図 目 次

2.1 カオス的振る舞いをする時系列データの短期予測へのアプローチ 12
2.2 n 次元再構成空間への埋め込み $(\mathrm{n} = 3)$ 14
2.3 前件部のメンバーシップ関数 15
2.4 カオス予測モデルにおける誤差と次元×遅延の関係 18
2.5 予測データ 19
2.6 負荷のアトラクタの軌道 $(n = 3, \tau = 6)$ 20

3.1 重負荷回避 24
3.2 軽負荷選択 24
3.3 システム環境 25
3.4 分散処理モニタ 27
3.5 プロセス処理系のフローチャート 28
3.6 分散処理モニタ (負荷予測系) のフローチャート 29
3.7 能力が均一な分散システム (定常状態) での平均応答時間 34
3.8 能力が均一な分散システム (変動状態) での平均応答時間 35
3.9 分散の効果 (変動状態) 36
3.10 能力が異なる分散システム (変動状態) での平均応答時間 38

4.1 計算機台数に伴う通信数の増加 41
4.2 改良型分散処理モニタ 43
4.3 プロセス処理系のフローチャート (改良型) 44
4.4 計算機間の結合レベル 45
4.5 計算機間の結合度の変化 (プロセス依頼受理) 47
4.6 計算機間の結合度の変化 (プロセス依頼拒否) 48

4.7 提案モデルの分散システム規模によるパフォーマンス変化 52
4.8 プロセス到着間隔によるパフォーマンス変化 (各計算機能力は均一) 54
4.9 プロセス到着間隔によるパフォーマンス変化 (各計算機能力は不均一) . . . 55

表 目 次

2.1 各モデルの予測誤差 17

3.1 無負荷時におけるプロセスの処理時間 32
3.2 処理計算機の選択アルゴリズム 33

4.1 結合状態における通信情報の変化 46
4.2 結合の復活を容易にするための Signal 49
4.3 比較モデル 51
4.4 均一な計算機で構成される分散システム (図 4.8) での再通信数 57
4.5 不均一な計算機で構成される分散システム (図 4.9) での再通信数 57

第 1 章

序論

1.1　本研究の背景

世界初のコンピュータといわれる ENIAC(Electronic Numerical Integer And Calculator) が 1946 年に S.P.Eckert と J.W.Mauchly によって作られてから，50 年経った今，我々は高度情報化社会の扉を開こうとしている．この高度情報化社会を身近な存在に感じさせるものとして，クライアント・サーバー・コンピューティングの普及，さらにはインターネットの普及があげられるであろう．例えば，クライアント・サーバー・モデルに基づく WWW は「仮想共有空間」での情報 (知識) の共有を容易とした．この「仮想共有空間」の構築によって次のような波及効果をもたらすと考えられる．

- 地理的に分離している人々が協同作業を容易に行えるようになる．つまり，距離的な隔たりを意識せずに，情報を共有しつつ知的な産業活動が可能となる．
- 情報発信者は時間的な制約から開放される．それぞれのスケジュールに従い，最も効果的な時間に，知的活動に専心できるようになる．つまり，知的活動の生産性が向上する．

また，現在話題になっている全米情報基盤 (NII)，地球規模の情報基盤 (GII)，ATM ベースの情報ハイウェイ (I-WAY)，電子データ交換 (EDI)，生産・調達・運用支援統合情報シ

ステム (CALS) などが実現されれば，教育，医療，金融，製造，公共などの諸分野に大きな改革が起きると言われている．

これら情報化社会の基盤として，高度なネットワーキング技術，高速なコンピュータ，そしてそれらを協調して動作させ，その性能を引き出す分散システムは必要不可欠であり，これらの研究はますます重要になると考えられる．

分散システムにおいて重要な技術として考えられているのが，ネットワークの透過性 (network transparency) であり，今なお数多くの研究がなされている．ネットワークの透過性とは，分散システムの利用者にネットワーク上に存在する資源を自在に利用可能とする．つまり，利用者にネットワークの存在を意識させない性質をいう．この性質を包含する概念として次のようなものがある．[1]

1. 位置透過性：分散する情報・資源の物理的位置が意識されない．
2. 性能安定性：負荷の変動によって性能が左右されない．
3. 同時実行透過性：同時に実行される処理の干渉が意識されない．
4. 障害透過性：ハードウェア・ソフトウェアの障害が意識されない．
5. 複製透過性：資源の複製が意識されない．
6. 移動透過性：資源の移動が意識されない．
7. 拡張透過性：システムおよび応用規模の変更が意識されない．

ネットワーク透過な分散処理を行うには，プロセスをグローバルに管理し，各計算機に割り当てる機能 (負荷分散) が必要となる．負荷分散の概念は，分散システムにおいて，負荷が一部の計算機のみに集中して，他の計算機がアイドル状態になるといった計算機間の負荷の不均衡によるシステム全体のパフォーマンスの低下を，負荷を計算機間において移送することによって防ごうというものである．負荷分散をより適切に行うことにより，システム全体の応答時間の短縮や，資源の利用率の向上などの，分散システムの性能向上が期待できる．

負荷分散方式には，次の 2 つの方式がある．

- 静的負荷分散：プロセス割当てをあらかじめ決めておくもので，プロセスの負荷が正確に見積もれる場合に適しているが，システム稼働時に負荷の変動がある場合は負荷の不均衡が避けられない．

- 動的負荷分散：プロセスの処理結果や処理状況に応じて，負荷配分を変えていくもので，プロセス割当てを 1 つの計算機が集中的に行う方式と各計算機で行う分散型の方式とがある．前者は，制御や管理が簡単な反面，耐故障性や通信量の観点から実際の分散システムへの適用は望ましくない．後者は，分散システムの特徴をいかすもので多くの研究がある [3] [4]．

分散型の動的負荷分散方式は投入プロセスに対して，そのローカルの計算機が処理計算機の選択を行い，プロセスを送り込む．その選択方法が分散システム全体のパフォーマンスを大きく左右する．従来，選択方法として，ローカルの計算機を優先的に選択する方法，プロセス投入時の計算機の負荷をもとに選択する方法，負荷の大きさや処理能力などに関する統計量を基準にした選択方法 [7]，履歴による優先度を用いた方法 [8] などが提案されている．これらは，いずれもその時点までの計算機の状態に基づいて選択するのものであるが，投入されるプロセスはその時点以降の負荷を与えるものであり，投入後の負荷が特定の計算機に集中する可能性がある．それを避けるため，計算機の負荷の監視を高頻度で行うとその情報の通信のための通信量が増大する．これを避けるために，低頻度の監視で従来の負荷状況を予測し，対処することが考えられる．

1.2　本研究の目的

本研究では『並列分散処理システム』の上で，重要な機能である負荷分散について注目し，負荷を予測しながら負荷分散を動的に行う方式を提案する．
負荷予測については，実際の計算機の負荷の挙動を線形・非線形の両モデルにあてはめ予測を行い評価する．
また，同時に本方式を組み込んだ動的分散処理モニタを開発し，実際に LAN で接続された計算機環境においてシミュレーションを行い評価する．
さらに，通信量を削減するため，各計算機において自律的に仮想の分散システム環境を作り出す機構を今回開発したモニタに組み込み，その効果について考察する．

1.3　本論文の構成

本論文の構成は以下のとおりである．

第 1 章では，本研究の背景，目的及び本論文の構成について述べる．

第 2 章では，計算機負荷の予測法及びその予測結果について述べる．

第 3 章では，予測を用いた動的分散モニタを開発及び予測を用いた動的負荷分散アルゴリズムを提案し，その評価について述べる．

第 4 章では，動的分散処理モニタを改良し，通信量をへらす仮想的な分散環境を作製し，その環境での分散処理について述べる．

第 5 章では，本論文の結論及び今後の問題点について述べる．

第 2 章

負荷予測

2.1 はじめに

LAN などで接続された実際の分散環境では，スーパーコンピュータはもはや一種の計算サーバーであり，従来の汎用計算機は，各システムの重要なデータを蓄積し，またシステム管理するマネージャとしての役割が重要視され，利用者が直接関係する処理部分は，ほとんどが利用者に近いワークステーションで実行されるのが自然である．

このような環境において (特に，特定の研究室内などに限ってみると)，各計算機の利用状況は決まっている場合が多い．この場合，本来予測が難しいとされてきた計算機の負荷 (計算機に割り当てられているプロセスが休止状態か走行状態かを考慮した平均プロセス数) をある程度予測することが可能であると考えられる．そして，その予測を利用して負荷分散を行うことは，分散処理システム全体のパフォーマンス向上につながることが期待できると考えられる．

負荷の予測を行うためには，負荷の時系列データをある確率モデルにあてはめる必要がある．一般に確率モデルを選定するためには，次のような手順が必要となる．[5]

1. 現象の中で確率的に変動するものを数量化する．
2. 確率変数のとる値は離散値か連続値か，その値の変わる範囲はどれほどか，を明らかにする．

3. 離散分布，連続分布それぞれから，以下のような方法からモデルを選択する．

   (a) 直観または仮定

   (b) 度数分布の形から判定

   (c) 理論的に決定

   (d) 類似現象からの類推

4. 選択分布のパラメタの決定

実際に負荷の時系列データから確率モデルを求める場合，この手順において問題となるのは 3(a),(b),(c),(d) の内，(b),(c) での選定は難しいことがあげられる．そこで本論文では (d) として株価のデリバティブに注目し，そこで使われている手法 (モデル) を線形予測モデルとして利用した．[6]

また，近年カオスの研究が注目されており [12]，その応用の一つとしてカオス的振舞をする時系列データの短期予測問題があり，上下水道需要予測，高速道路交通量予測，電力需要量予測などに応用されている．本論文では．これに注目し，非線形予測としてカオス予測モデルを負荷予測に応用した．[11]

本章では，各計算機の負荷を一定間隔で取得しながら，その値を

- 線形予測モデル ... n 次元多項式モデル (負荷を対数正規分布に従うと仮定)
- 非線形予測モデル ... カオス予測モデル

にそれぞれあてはめ，各モデルにおいて近未来を予測しその評価を実際に計算機をもちいてシミュレーションを行った．

2.2　線形予測モデル

2.2.1　負荷の確率分布特性

まず，負荷はランダム・ウォークに従うという仮定をする．これは，短期間における負荷の変動率が正規分布に従うことを意味し，さらにこのことは，将来の任意の時点において，負荷が対数正規分布に従うことを示す．負荷の変動が対数正規分布に従うと仮定すると，キーとなるパラメータは，

- 負荷の変動期待値
- 負荷の変動期待値の不確実さ

の 2 つである．負荷の変動期待値 μ は微小期間 (Δt) の変化率であり，変動期待値の不確実さ σ は微小期間 Δt における負荷変化率の標準偏差を $\sigma\sqrt{\Delta t}$ と表した場合の σ と定義される．

対数正規分布に従う変数はその自然対数が正規分布に従う．したがって負荷の対数正規性を仮定すれば，将来時点 P の負荷を L_p とすると，$\ln L_p$ は正規分布に従い，$\ln L_p$ の平均 (式 (2.1)) と標準偏差 (式 (2.2)) はそれぞれ以下のようになる．

$$\ln L_n + \left(\mu - \frac{\sigma^2}{2}\right) P \tag{2.1}$$

$$\sigma\sqrt{P} \tag{2.2}$$

つまり，

$$\ln L_p \sim \phi\left(\ln L_n + \left(\mu - \frac{\sigma^2}{2}\right) P, \sigma\sqrt{P}\right) \tag{2.3}$$

となる．ここで，L_n は現在負荷値であり，$\phi(m, s)$ は平均 m ，標準偏差 s の正規分布を示す．このときの，L_p の期待値すなわち平均値 $E(L_p)$ は,

$$E(L_p) = L_n e^{\mu P} \tag{2.4}$$

となり，また L_p の分散 $var(L_p)$ は，

$$var(L_p) = L_n^2 e^{2\mu P}(e^{\sigma^2 P} - 1) \tag{2.5}$$

で与えられる．

2.2.2　負荷の変動期待値

2.2.1での仮定において，もっとも重要となるのは負荷の変動期待値μ の決定である．本論文では，μ が微小期間の負荷の変化率に対応することに着目し，以下のような手順で μ を決定した．

1. 計算機の負荷を T(:サンプリング間隔) 秒ごとに取得し，その値を負荷の離散的な時系列データ $(y(t-nT), y(t-(n-1)T), .., y(t-T), y(t))$ として格納する．
2. その時系列データをを最小 2 乗法により n 次多項式 (式 (2.6)) にあてはめる．

$$f(t) = \sum_{i=0}^{n} a_i t^i \tag{2.6}$$

3. 現時点 $(t=0)$ での傾き$\alpha(= f'(0) = a_1)$ を求め，この値を変動期待値 μ とする．

これにより，将来時刻 t_pにおける予測負荷値 L_pの期待値 $E(L_p)$ は式 (2.4) を用いて，次のように表わせる．

$$E(L_p) = L_n e^{\alpha t_p} \tag{2.7}$$

つまり，1step 後の予測負荷値 $y(T)$ の期待値 $(E(y(T))$ は，

$$E(y(T)) = y(0) e^{a_1 T} \tag{2.8}$$

となる．

2.2.3　負荷の変動期待値の不確実さ

負荷の変動期待値の不確実さ σ は，負荷の変動が大きいほど大きな値となる．つまり実際にプロセスを分散する際に，2 つ計算機の負荷の予測期待値が同じ場合，この値が小さい方がより処理計算機として適当であることを示すことになる．

σ の推定は，次のように行われる．

まず ,2.2.2の手順 1 と同様に負荷の時系列データを格納し ,u_i (式 (2.9),$i = 0, 1, 2, \ldots, n$) を求める．

$$u_i = \ln\left(\frac{y(i)}{y(i-1)}\right) \tag{2.9}$$

u_i の標準偏差の不変推定値 s は，

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(u_i - \overline{u})^2} \tag{2.10}$$

すなわち，

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}u^2 - \frac{1}{n(n-1)}\left(\sum_{i=1}^{n}u_i\right)^2} \tag{2.11}$$

で，与えられる．ここで，$\overline{u}$は u_iの平均である．

また， 式 (2.3) より，次のように表すことができるので，

$$\ln\left(\frac{y(i)}{y(i-1)}\right) \sim \phi\left(\left(\mu - \frac{\sigma^2}{2}\right)T, \sigma\sqrt{T}\right) \tag{2.12}$$

u_i の標準偏差は $\sigma\sqrt{T}$ となる．つまり，変数 s は，$\sigma\sqrt{T}$ の推定値となる．よって σ は次のように推定される．

$$\sigma = \frac{s}{\sqrt{T}} \tag{2.13}$$

この推定の標準誤差は $\sigma/\sqrt{2n}$ で近似される．

2.3　カオス予測モデル

2.3.1　カオス概論

1970 年代にカオスという概念が生まれてから，いままで完全なカオスの定義は確立していない．カオスは非常に複雑な現象であるため，従来はその概念を理解することが非常に困難であった．しかし，コンピュータの発達した現在では，カオスの近似的な姿を直接コンピュータで映し出すことができるようになり，カオスは様々なシステムに広く存在する現象として数多くの研究がなされるようになった．

現在，提唱されている代表的なカオスの定義 [14] は次のようなものである．

- 解は非周期的である
- 自己相関関数は遅れ時間の増大とともに 0 に収束する
- 初期値に対する鋭敏な依存性を有する

一見複雑で不規則に見える現象のなかに，実はストレンジアトラクタ[1]で現れるような構造が内在していて，その構造が意外と単純なモデル表せることがある．この点が，カオスを工学的に利用可能にしている 1 つである．特に，カオスを工学的な立場から観察すると，次のような応用が考えられる．[15]

- 非線形システム制御
- 並列分散処理 (カオスニューラルネットワーク [13])
- パターン認識 (フラクタル画像認識・特徴抽出)
- カオスメモリ
- 時系列の非線形予測 [12]
- バイオカオス

[1] カオス的現象をある相空間上にマッピングしたときに，収束して落ち着く先のこと．またのその軌道をトラジェクトリという

本論文で注目している「時系列の非線形予測」は，周期性のない複雑で不規則な現象から，なんらかの決定論的な力学系の規則性を見い出し，その近未来の状態を予測しようとするものであり，

- 気候・地球環境の変動予測
- 経済予測
- 電力・上下水需要量予測 [16] [17]

などが研究され，利用されている．

本論文は，計算機負荷の時系列データがカオス的振る舞いをするとみなし，これを応用して予測を行う．

2.3.2　カオス予測

カオス的振る舞いをする時系列データとしては，明確な数式モデルに従っているものから，自然現象・社会現象のなかにもカオス的振る舞いがあることが知られており，これらの分野ではその予測は重要課題として様々な研究がなされている [16] [17]．

短期予測へのアプローチは図 2.1に示すように，観測された時系列データの振る舞いがカオス的であるならば，その振る舞いは決定論的な法則に従っていると考えることができ，その非線形的な決定論的規則性を推定することができれば，近未来のデータを予測することが可能となる．このような決定論的力学系理論の立場からの予測は，

- 1 本の観測時系列データから，もとの力学系の状態空間とアトラクタを再構成する

という，タケンスの理論 [10] に基づいている．この理論は，対象システムが決定論的力学系であって，観測時系列データ $(y(t), y(t-\tau), ..., y(t-(n-1)\tau)$:τ は時間遅延$)$ がこの力学系の状態空間から 1 次元ユークリッド空間 Rへの C^1連続写像に対応した観測系を介して得られたものと仮定することにより，この再構成軌道が，n を十分大きくとれば元の決定論的力学系の埋め込みになっているというものである．つまり，観測時系列データが元の力学系のアトラクタに由来しているものであるならば，再構成状態空間にはこのアトラクタの位相構造を保存したアトラクタが再現されていることになるというものであり，アトラクタを推定することにより，予測できることになる．

本論文では，この理論に基づいて次の手順で負荷の予測を行う．

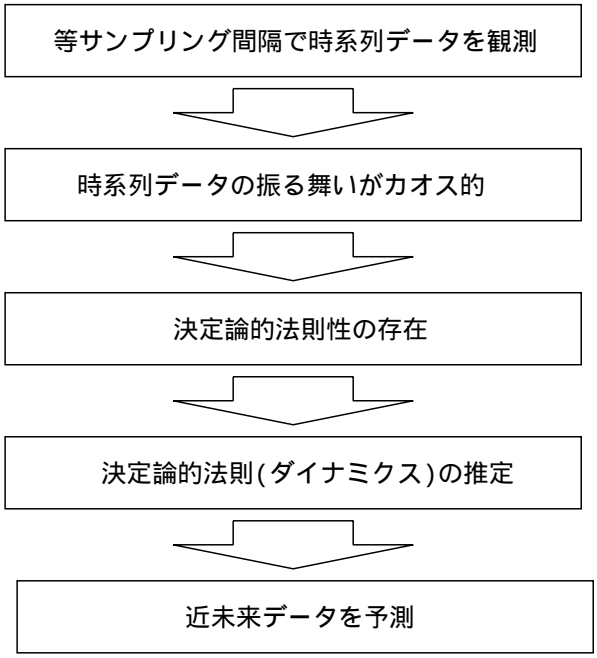

図 2.1: カオス的振る舞いをする時系列データの短期予測へのアプローチ

1. 等サンプリング間隔観測された負荷の時系列データ $(y(t))$ を埋め込み次元 n，時間遅延 τ で，n 次元再構成状態空間に変換する次のようなベクトル x (式 (2.14)) を構成する．

$$\mathbf{x}(i) = (y(i), y(i-\tau), ..., y(i-(n-1)\tau) \tag{2.14}$$

2. 構成したベクトルをタケンスの埋め込み理論に基づいて，n 次元再構成状態空間に埋め込む (図 2.2)．

3. 埋め込みにより再構成された状態空間とアトラクタの軌道に関して，現在の時系列データを含むデータベクトル $\mathbf{x}(i) = (y(i), y(i-\tau), ..., y(i-(n-1)\tau)$ と，その近傍のデータベクトル (ユークリッド距離で近いものから，$\mathbf{x}(i_0), \mathbf{x}(i_1), ...)$ の軌道を用いて，近未来の軌道を推定する．これを局所再構成という．

今回，この局所再構成法として，局所ファジィ再構成法 $[11]$ を用いた．
この方法の具体的な例として，埋め込み次元 $n = 3$，近傍のデータベクトル数が 3 の場合を説明する．

各々データベクトルは，式 (2.15) のようになり，

$$\begin{aligned} \mathbf{x}(i) &= (y(i), y(i-\tau), y(i-2\tau)) \\ \mathbf{x}(i_0) &= (y(i_0), y(i_0-\tau), y(i_0-2\tau)) \\ \mathbf{x}(i_1) &= (y(i_1), y(i_1-\tau), y(i_1-2\tau)) \\ \mathbf{x}(i_2) &= (y(i_2), y(i_2-\tau), y(i_2-2\tau)) \end{aligned} \tag{2.15}$$

推定するデータベクトル $\mathbf{x}(i+1) = (y(i+1), y(i+1-\tau), y(i+1-2\tau))$ のうち，欲しい予測データ $y(i+1)$ を得るため，以下のようなファジィルール (式 (2.16), $j = 0, 1, 2$) を考える．

$$\text{IF } y(i) \text{ is } \check{y}(i_j) \text{ THEN } y(i+1) \text{ is } \check{y}(i_j+1) \tag{2.16}$$

$\check{y}(i_j), \check{y}(i_j+1)$ はファジィ関数を表しており，その前件部のメンバーシップ関数は図 2.3となる．後件部のメンバーシップ関数は簡単のためシングルトン表現とした．

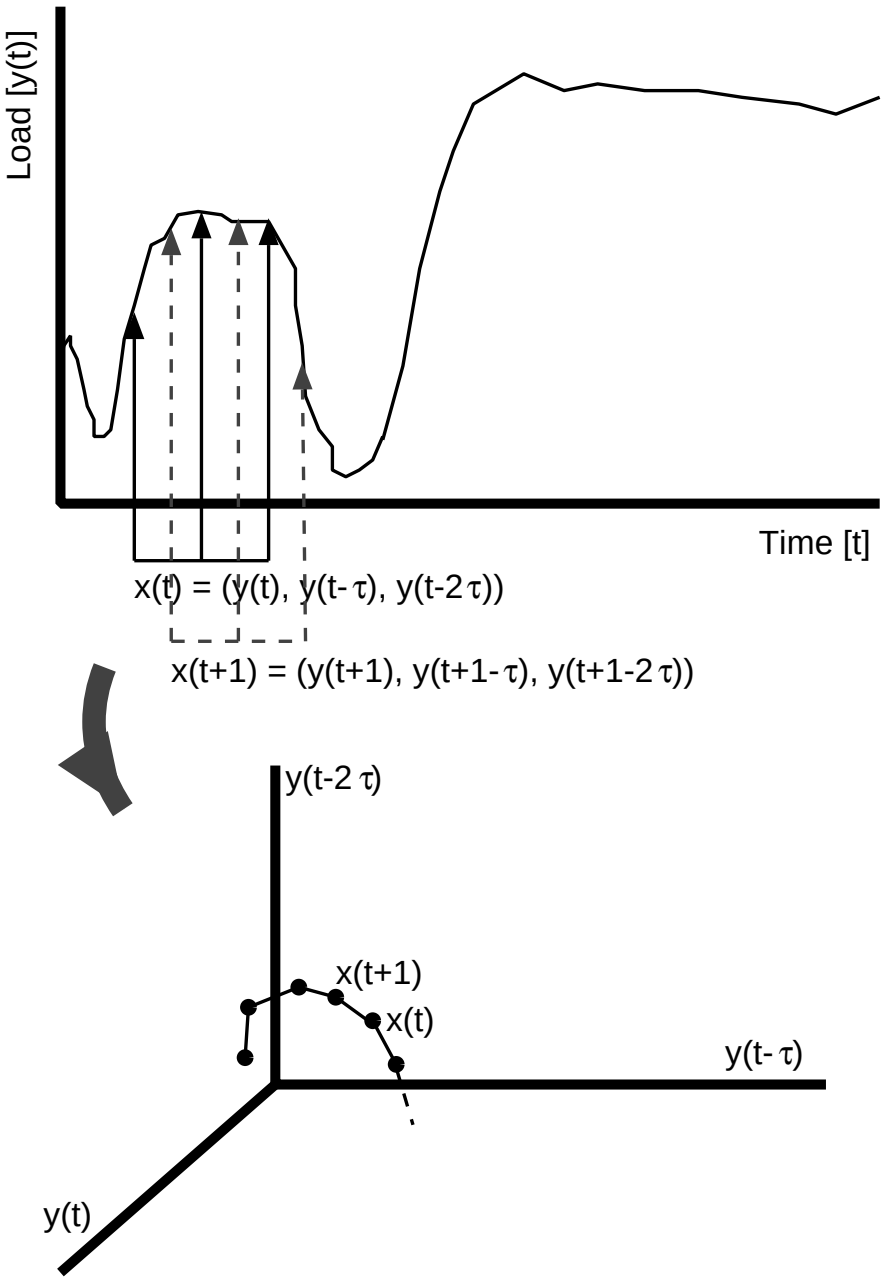

図 2.2: n 次元再構成空間への埋め込み $(\mathrm{n} = 3)$

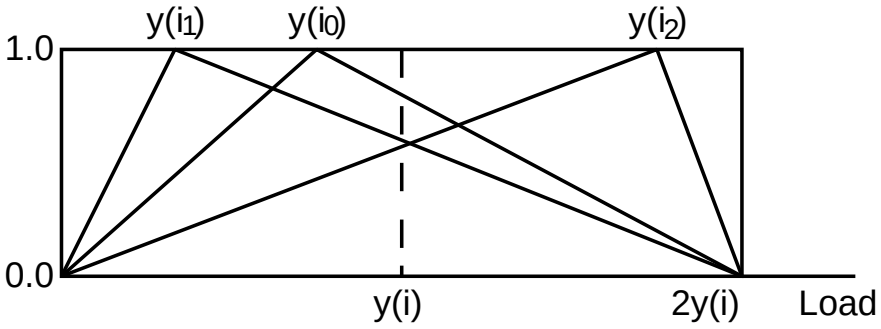

図 2.3: 前件部のメンバーシップ関数

2.4　負荷予測実験

本節では，2.2, 2.3で示した両モデルの負荷予測実験の評価について述べる．

2.4.1　実験条件

以下のような条件のもとで予測実験を行った．

- 負荷の時系列データ $y(t)$ は，サンプリング間隔 T を 6 秒間隔 $(1\mathrm{step})$ として，実際に稼働している計算機 $(\mathrm{SPARK\ station\ 20})$ から取得した．
- 線形予測モデル
  - 過去 $20\mathrm{step}$ の時系列データを 3 次多項式にあてはめ，負荷の変動期待値 $\mu(=a_1)$ を算出し，式 (2.8) より予測負荷値を算出．
- カオス予測モデル
  - 過去 24 時間以上の時系列データを n 次元再構成状態空間に埋め込んでおく．
  - 局所再構成法として 3 つの近傍ベクトルから再構成を行う．

2.4.2　予測結果

予測結果を表 2.1，図 2.5に示す．

表 2.1は，1step ごとに予測をした 1 時間分の予測誤差である．カオス予測モデルについては，再構成状態空間の次元数 n，データベクトルの遅延 τ をふって予測した．

図 2.5は，その 1 時間の予測のうちある区間の実データを表わしたもので，それぞれ，測定した実際のデータを real，線形予測を liner，カオス予測を chaos として，表している．

これら結果からわかるように，どの次元にあてはめてもカオス予測の方が，線形予測より良い結果を示した．また，カオス予測もその埋め込み次元や遅延によって予測誤差が大きく変わることが観測され，$n \times \tau$が大きい程，予測誤差が小さくなる傾向を示していた (図 2.4)．

また，カオス予測モデルにおける負荷データのアトラクタの軌道 $(n = 3, \tau = 6)$ を図 2.6に示す．

線形予測	カオス予測		
誤差 (%)	誤差 (%)	次元 n	遅延 τ
3.07	1.81	2	1
	1.91	3	1
	1.73	3	2
	1.95	4	1
	1.76	4	2
	1.79	4	3
	1.86	5	1
	1.88	5	2
	1.69	5	3
	1.65	5	4
	1.85	6	1
	1.75	6	2
	1.55	6	3
	1.37	6	4
	1.18	6	5
	1.78	7	1
	1.79	7	2
	1.47	7	3
	1.24	7	4
	1.15	7	5
	0.98	8	5
	1.01	9	8

表 2.1: 各モデルの予測誤差

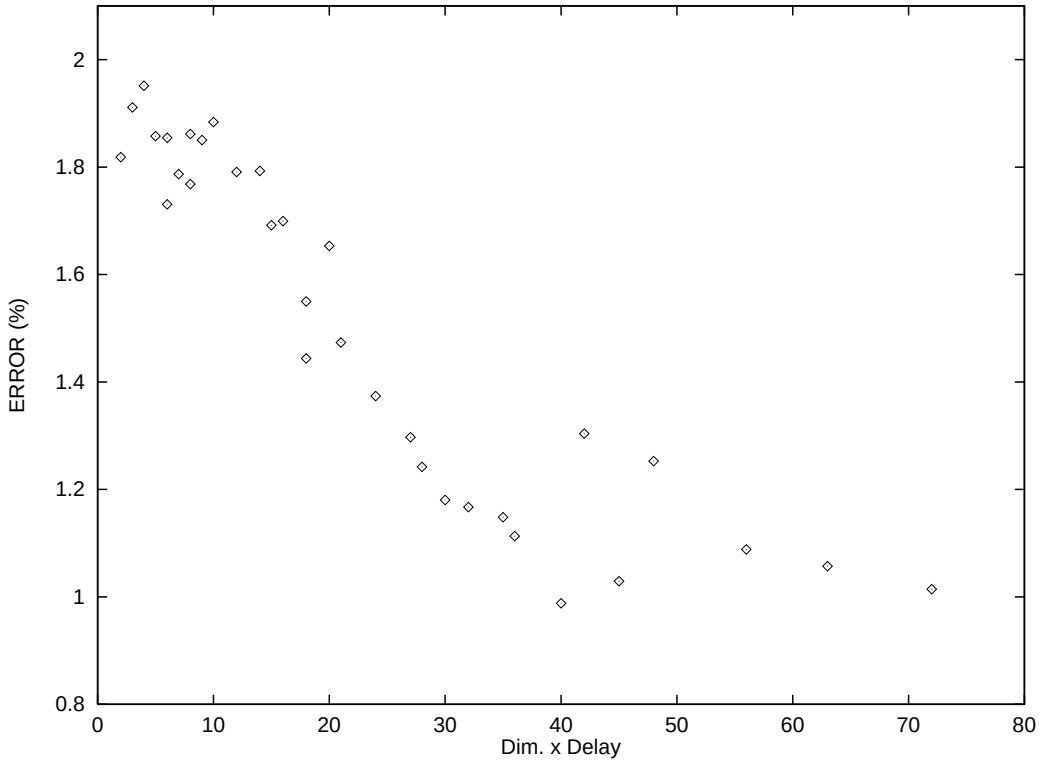

図 2.4: カオス予測モデルにおける誤差と次元×遅延の関係

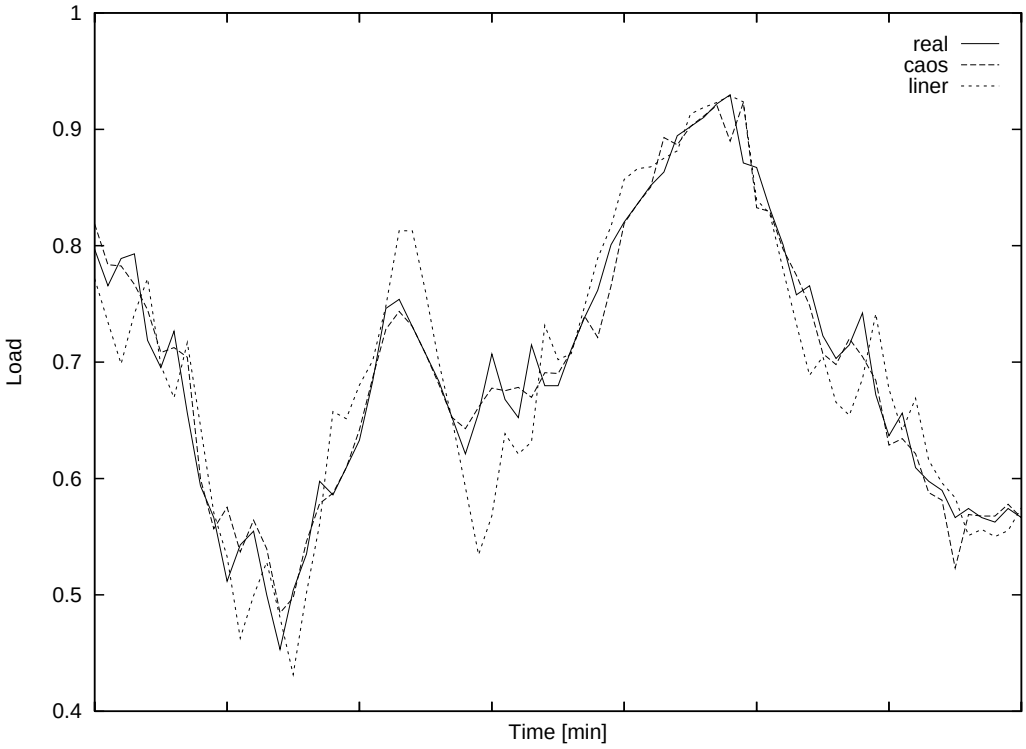

図 2.5: 予測データ

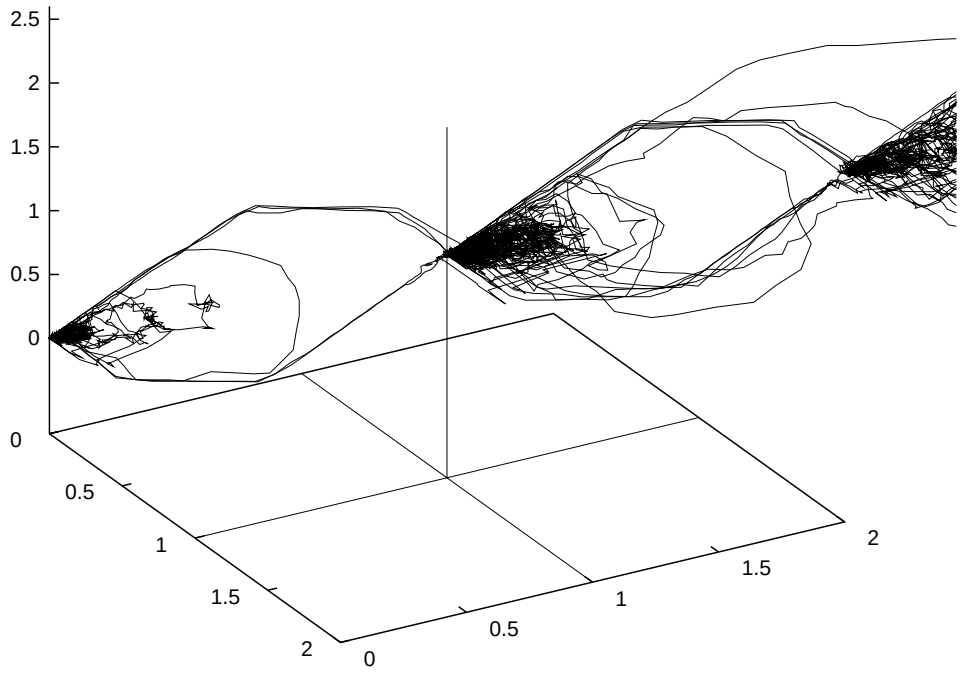

図 2.6: 負荷のアトラクタの軌道 $(n = 3, \tau = 6)$

2.5　まとめ

本章では，負荷予測手法として，

- 線形予測モデル ... n 次元多項式モデル (負荷を対数正規分布に従うと仮定)
- 非線形予測モデル ... カオス予測モデル

を提案し，評価を行った．

実験の結果から，線形予測モデルよりも，カオスを用いた非線形予測モデルの方がよい予測をおこなうことが観測された．これは，負荷の変動自体が不確定な要素に大きく起因していているので，より複雑な系を表現できる非線形モデルほうがよい予測値を示したのは妥当な結果と言える．

カオス予測モデルについては，[埋め込み次元 (n) × 遅延 (τ)] が，大きいほど予測誤差が小さくなる傾向をしめした．これは，この値が大きいほど，より多くの (より広い間隔の) データのから状態空間を再構成するためであると考えられる．

この結果をふまえて，分散処理モニタ (3.3.2，図 3.4) の予測処理系には，カオス予測モデルを組み込んだ．また，表 2.1，図 2.4より，より低次元で割合予測誤差の小さかった[2]次元数 6 (遅延数 5) をカオス予測のパラメータとした．

今回の実験では 24 時間分のデータをもとに実験を行ったが，人間のサイクルにより近い 1 週間のデータをもとにカオス予測を行えば，より効果的な予測が可能であると考えられ，また従来予測が困難とされてきたプロセスの挙動の予測ができる期待もあり，これからの研究課題といえよう．

[2]埋め込み次元が大きい程，予測コストがかかる [11]

第 3 章

動的負荷分散アルゴリズム

3.1 はじめに

待ち行列にの理論によると，分散処理システムの各計算機を単一サーバーの待ち行列としてモデル化すると，プロセスの到着率が処理率より大きい過負荷の計算機の待ち行列の長さと平均待ち時間は極端に増大する．これを，負荷を各計算機間で分散させ，均等化させようという考えが負荷分散である．しかし，負荷分散をより効果的に行うためには，プロセス間通信や入出力のコストを最小化しながらシステムのスループットの効率を上げるといった，相反する目的を達成しなければならない．

本章では，以下のような内容について述べる．

- 従来提案されている動的負荷分散アルゴリズムの紹介
- 負荷予測機構を組み込んだ分散処理システム (モニタ) を開発
- 負荷予測を用いた動的負荷分散アルゴリズムの提案
- 提案アルゴリズムの評価，検討

3.2　従来提案されている動的負荷分散アルゴリズム

分散処理システムにおいて，他の計算機の情報をより詳細に得ることはシステム全体のパフォーマンスに大きく作用する．その情報として第一に考えられるのが計算機の負荷情報である．しかし，他の計算機のより正確な負荷情報を得るためには，多量の通信が必要となり，逆に分散システム全体のパフォーマンスを低下させる原因となる．

そこで，通信量を抑えながら負荷情報を得る方法として，負荷を Heavy と Light という 2 つのランクに分け，その状態が変化したときにのみ同報通信で負荷情報を伝えるという方式が提案されている [7] [8]，しかし，この方式では通信量を抑えることができるものの，2 段階の負荷情報であるため，ある場合には不均衡が生じやすい．ランクを用いた処理計算機選択アルゴリズムとして，

- 重負荷回避 (図 3.1)
- 軽負荷選択 (図 3.2)

がある．

重負荷回避は，プロセスが投入された計算機のランクが Heavy でないなら，その計算機にプロセスを割り当て，Heavy なら Light な計算機を探し，そこへプロセスを割り当てるアルゴリズムである．この方法はもっとも単純であるが，Heavy でなければ必ずその計算機で処理されることになり，プロセスの投入された計算機の能力によってシステム全体のパフォーマンスが大きく異なってしまう．

また，軽負荷選択は，プロセスが投入された計算機のランクに関係なく，ランクが Light の計算機全体の中からより適切な計算機を選択する方法である．この「適切さ」の評価法として，

- 統計的負荷情報として計算機が Light であった時間 (軽負荷累積時間 T) を考え，その時間より算出した指数関数的な重み ($10^{\alpha T}$: αは重み係数) を利用した手法 [7]
- 指数関数的な重みを用いずに軽負荷累積時間そのものや，その時間に過去のスループットをたし合わせた量を優先度として用いた手法 [8]

が提案されている．

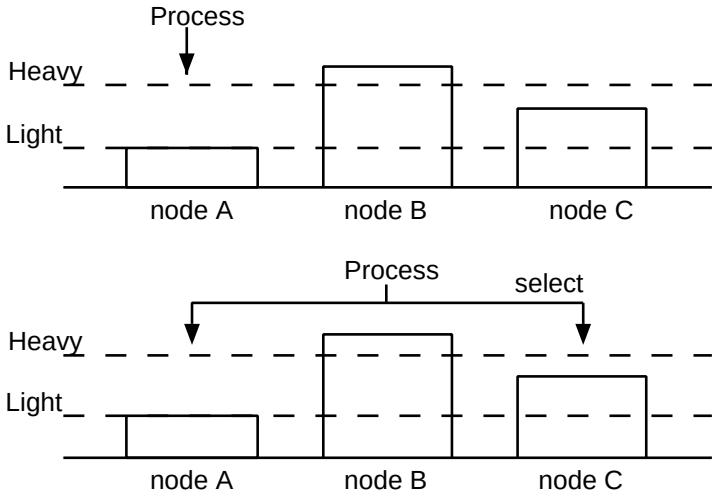

図 3.1: 重負荷回避

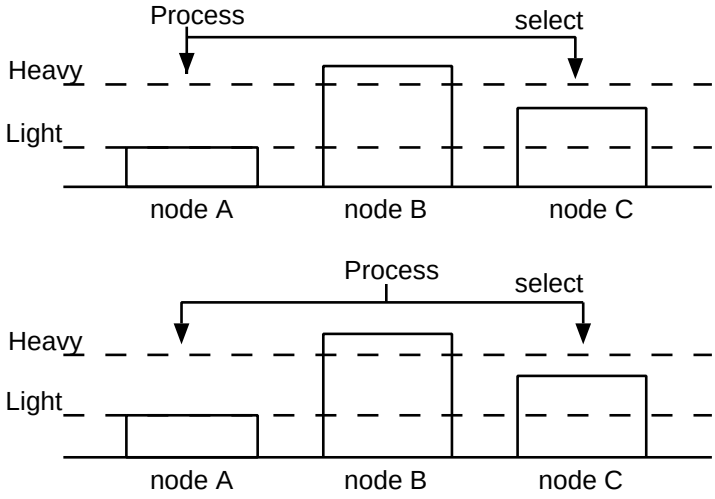

図 3.2: 軽負荷選択

3.3　分散処理システムモデル

本稿で開発したシステムは以下のようなモデルのもとで設計した．

3.3.1　分散システム

- 今回想定している分散システム環境 (図 3.3) は，複数の計算機 (node) がバス型の LAN で接続された環境である．
- 計算機の性能は必ずしも同一ではない．
- 計算機における処理単位はプロセスとする．
- プロセスは各計算機で独立に生成される．
- プロセスの輸送は任意の計算機間で可能である．
- 各計算機におけるプロセッサは単一で，プロセスの実行は逐次処理とする．

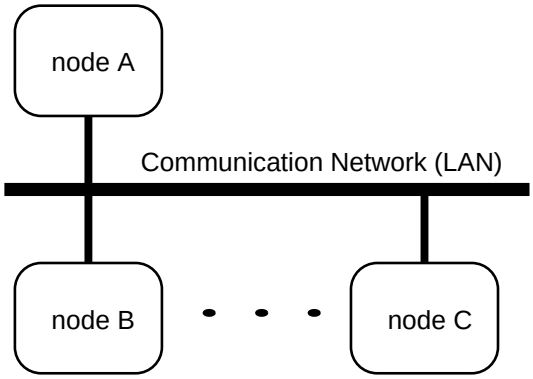

図 3.3: システム環境

3.3.2　分散処理モニタ

分散処理ための制御は，各計算機上の分散処理モニタで実現する．
分散処理モニタ (図 3.4) は次の 2 つの系から構成される．

- プロセス処理系
  - Queue
  - Distributor
  - Processor
- 負荷予測系
  - Loader
  - LoadTable

プロセス処理系では次の処理を行う (図 3.5)．
まず，Queue に到着プロセスを格納し，そのプロセスの発生順にソートする．そして，Distributor で LoadTable 参照しながら処理計算機が選択され，プロセスを輸送，または処理する．プロセスは各計算機にライブラリとして実装されているものとし，プロセスを他の計算機に輸送する場合は，キーワードとパラメータを送ることで実現する．

負荷予測系では次の処理を行う (図 3.6)．
Loader で OS が監視している負荷の値 (計算機に割当てられているプロセスが休止状態か走行可能状態かを考慮した平均プロセス数) を一定間隔で取り込みながら，負荷予測を行う．その結果 (計算機の現在負荷値と予測負荷値) を，同報通信で他の計算機に伝え，それぞれの LoadTable に格納する．すなわち，各計算機の LoadTable には，分散システムにぶらさがっている他の計算機の負荷情報が格納されている．負荷情報の 1 つとして計算機にログインしている user 数も同報通信し，各計算機の LoadTable に格納する．

また，実際にシステムとして稼働しているときは，各計算機の Queue の情報やプロセスの処理時間 (Throughput) なども，LoadTable に格納する．

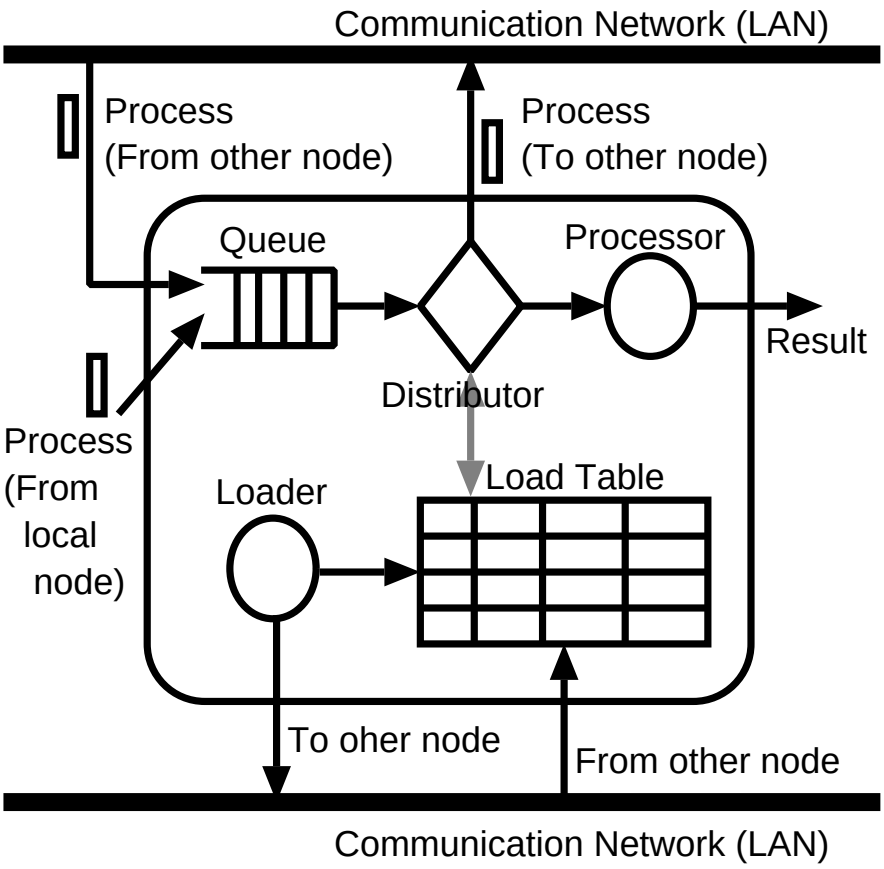
Communication Network (LAN)
Process
(From other node)
Process
(To other node)
Queue
Processor
Result
Distributor
Process
(From
local
node)
Loader
Load Table
To oher node
From other node
Communication Network (LAN)

図 3.4: 分散処理モニタ

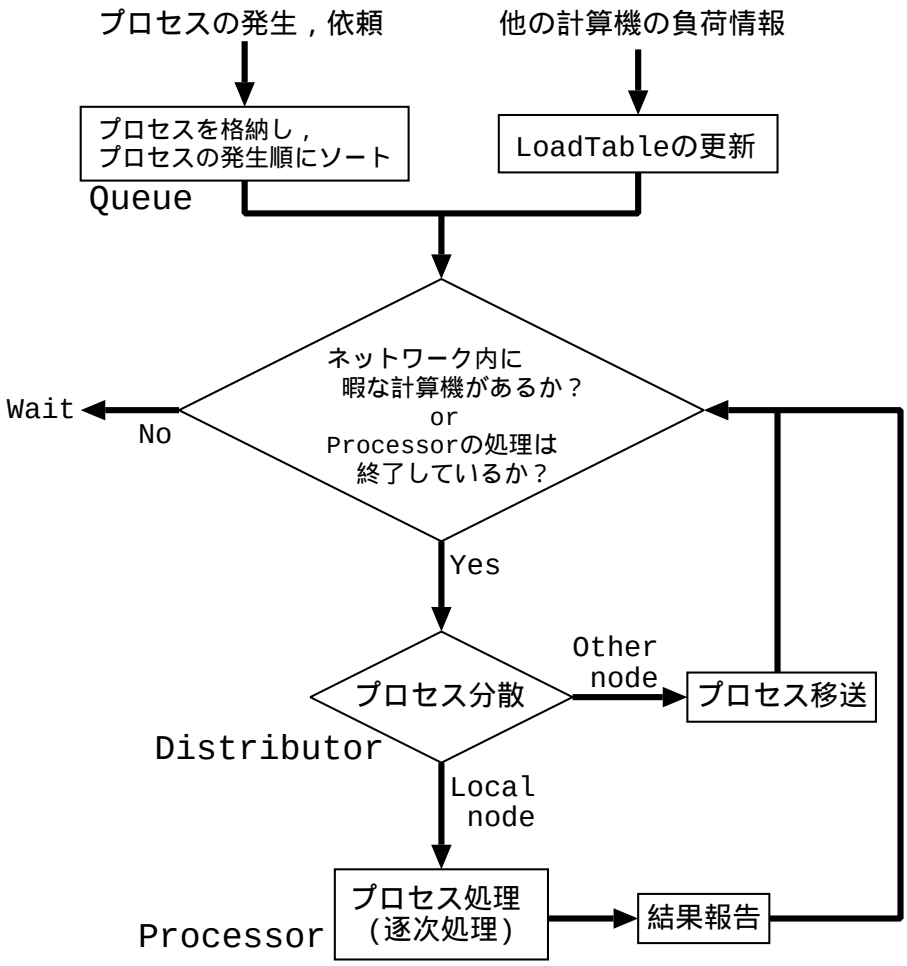

図 3.5: プロセス処理系のフローチャート

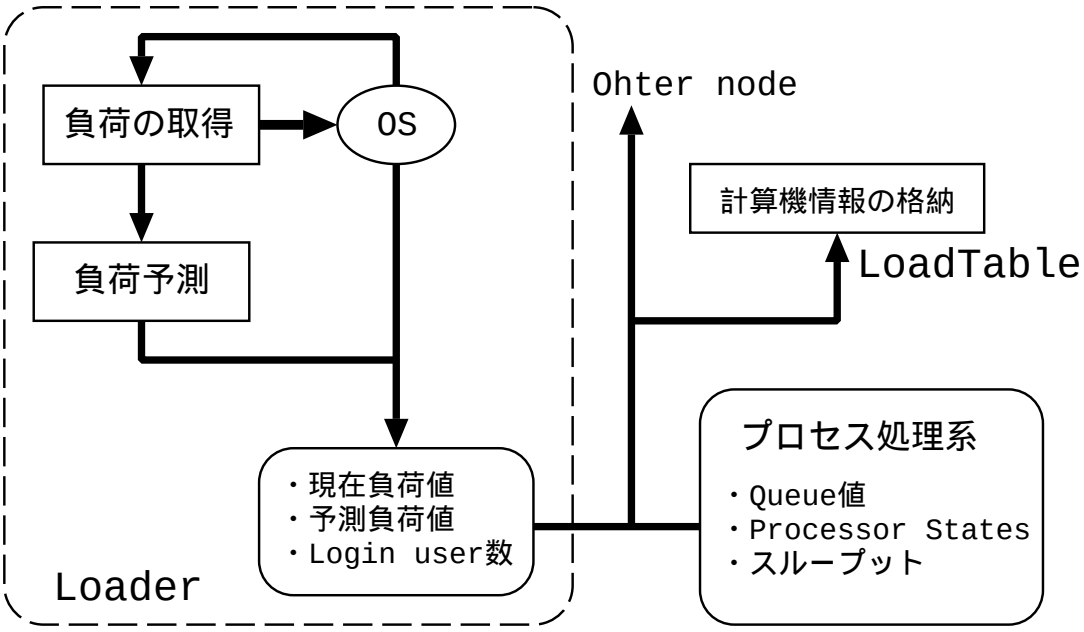

図 3.6: 分散処理モニタ (負荷予測系) のフローチャート

3.4　予測を用いた処理計算機選択アルゴリズム

Distributor は個々の計算機は各計算機 i の負荷情報に基づいた選択基準値 S_i を計算し，S_i の値をもとにシステム全体の中から適切な処理計算機を選択する．そして，選択した計算機に処理を依頼する．

3.2で述べた従来方式では選択基準をその時点での負荷に基づいていたが，現在，負荷が少ない状態でも，負荷がこれから上昇するであろうと予測される計算機を避け，負荷がある程度高くてもこれから負荷が減少すると予測される計算機にプロセスを割り当てる方がシステム全体としてのパフォーマンスは向上すると考えられる．

この考えのもとに，予測負荷に基づいたアルゴリズムを考察した．

3.4.1　単純な予測選択アルゴリズム (pre)

この選択アルゴリズムは負荷予測値とその変化 (上昇 or 下降 or 一定) をもとに選択基準値を求め，処理計算機を決定する．手順を以下に示す．

1. Distributor に分散依頼のプロセスが到着する．
2. 各計算機について選択基準値 S_i (式 (3.1)) を計算する．

$$S_i = W_1(L_{p_i}, L_{n_i}) \times \frac{P_i}{L_{p_i}} - W_2 \times N_i \tag{3.1}$$

3. S_i が大きい順に計算機のキューを参照し，プロセスを処理していない計算機，または，キュー内のプロセス数が少ない計算機を処理計算機として選択する．
   1. へもどる．

ここで，P_i は計算機の能力，L_{p_i} は予測負荷値，L_{n_i} は現在負荷値であり，N_i はその計算機にログインしている人数であり，各値はプロセスを分散する計算機の LoadTable 内の値である．また，W_1 は負荷の変化 (上昇 or 下降) に応じた重みであり，W_2 は，計算機ごとに定められた定数である．

3.4.2　履歴をもちいた予測選択アルゴリズム (his)

このアルゴリズムは，過去のスループットと負荷予測値をもとに処理計算機を選択する．このアルゴリズムでは，各計算機の LoadTable に分散システム中の各計算機がどれぐらいの負荷のときに，どれくらいのスループットを示すかを記録するようにしており，処理を実行したときの負荷のランク (12 段階) ごとに，そのスループットの (最大，最小，平均) が格納される．

ある計算機でプロセスが発生したときの具体的な処理計算機の選択手順を以下に示す．

1. Distributor に分散依頼のプロセスが到着する．
2. その計算機の現在負荷値を参照し，各計算機の負荷のランクを求める．
   ランクは負荷値が 0.25 以下からはじまって，0.5 ずつ 5.75 以上までの 12 段階である．(~ 0.25, 0.25 ~ 0.75, ..., 5.25 ~ 5.75, 5.75 ~)
3. 各計算機の予測負荷値を現在負荷値と比較して，負荷が上昇傾向 (負荷変化量が 0.1 以上) にあればそのランクのスループットの最大値を，下降傾向にあれば最小値を，そうでなければ平均値を求める．
4. 各計算機のキューの状態を参照し，3. の値が最も小さいものを処理計算機として選択し，計算を依頼する．
   スループットに 2 倍以上の差がある時には，処理をしていない計算機よりも，処理中でもより速い計算機を選択し依頼する．
5. プロセスが処理されたら，その処理計算機とそのスループットを LoadTable に格納する．
   1. へもどる．

このアルゴリズムで用いられている各値は，プロセスを分散する計算機の LoadTable 内の値である．

このアルゴリズムの場合，3.4.1で用いた計算機の能力 P_i を用いる必要がないため，未知の計算機が分散システム中に存在する場合においても，自ら他の計算機の能力を認識し，より適当に選択できる特徴を持つ．

3.5　シミュレーション

3.4で提案した予測を用いた選択アルゴリズムが，予測を用いない選択アルゴリズムと比較して，どの程度分散システムのパフォーマンスを改善するかを調べるために，比較シミュレーションを行った．

3.5.1　シミュレーション条件

- LAN によってつながれた 3 台の計算機を使用．
- プロセスは各計算機でポアソン分布に従って独立に生成される．
- プロセスは，四則演算からなる単純な計算とする．
  (本シミュレーションで使用した計算機の無負荷時の処理時間を表 3.1 に示す.)
- プロセスの処理時間と比較して Distributor の処理時間は十分無視できる．
- 処理計算機の選択アルゴリズムで用いた計算機の能力 P_i は，ベンチマークプログラムを実行し，はじめに既知として与える．
- 負荷の予測の効果を調べるために，各計算の負荷 (プロセス) 状態を以下の 2 状態とする．
  1. 定常状態.... 計算機内にシミュレーションの発生プロセス以外は存在しない．
  2. 変動状態.... 他のプロセスは自由に出入り可能．

計算機	処理時間 (sec)
SS20	2.8
SS10	4.1
S-4/2A	8.2
S-4/2B	3.1

表 3.1: 無負荷時におけるプロセスの処理時間

3.5.2　性能評価

プロセスの平均発生率を変化させた時の，システム全体の平均応答時間を評価値として，各選択アルゴリズム (表 3.2) を比較した．
また，シミュレーション時間は，各計算機において最低 200 個のプロセスが発生するまでとした．

non	負荷分散をしない． (常に local で処理する)
ran	ランダムに選択．
now	現在負荷値のみで選択． (選択基準値 $S_i = P_i/L_{n_i}$ とする． L_n は現在負荷値)
pre	単純な予測選択アルゴリズム (3.4.1)．
his	履歴をもちいた予測選択アルゴリズム (3.4.2)．

表 3.2: 処理計算機の選択アルゴリズム

計算機能力が均一な分散システムの場合

計算機として 3 台の S-4/2B (表 3.1) を選択し，プロセスは平均 3.5 ~ 8.0 (sec) のポアソン分布に従って発生させ，シミュレーションを行った．

そのときの平均応答時間を図 3.7，図 3.8，図 3.9 に示す．

図 3.7 は，定常状態においては各選択アルゴリズムにほとんど差がないことを示している．これは，同一の計算機で，同一状態 (定常状態) の場合は計算機間に大きな差がでないためであると考えられる．図 3.8 が示すように．変動状態においては，予測を用いた pre の選択アルゴリズムが予測を用いない sim より悪い時もあったが，his はすべてのプロセス発生率において良い結果を示した．図 3.9 からは，負荷分散をする場合としない場合との差が明らかにわかり，負荷分散の効果が確認できる．

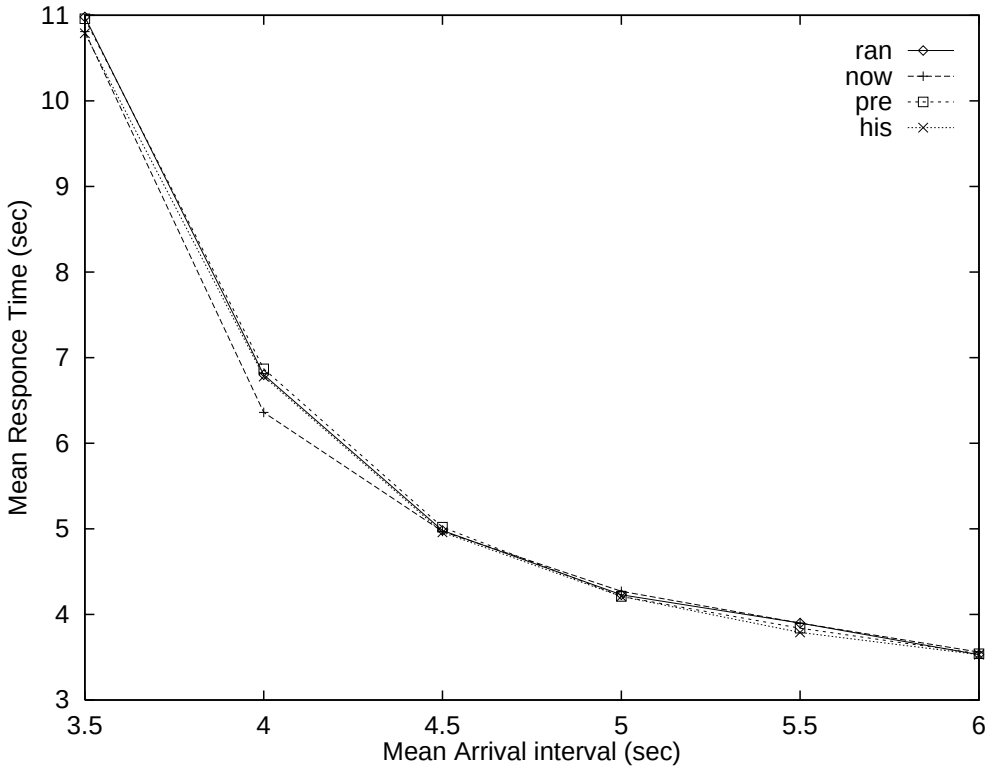

図 3.7: 能力が均一な分散システム (定常状態) での平均応答時間

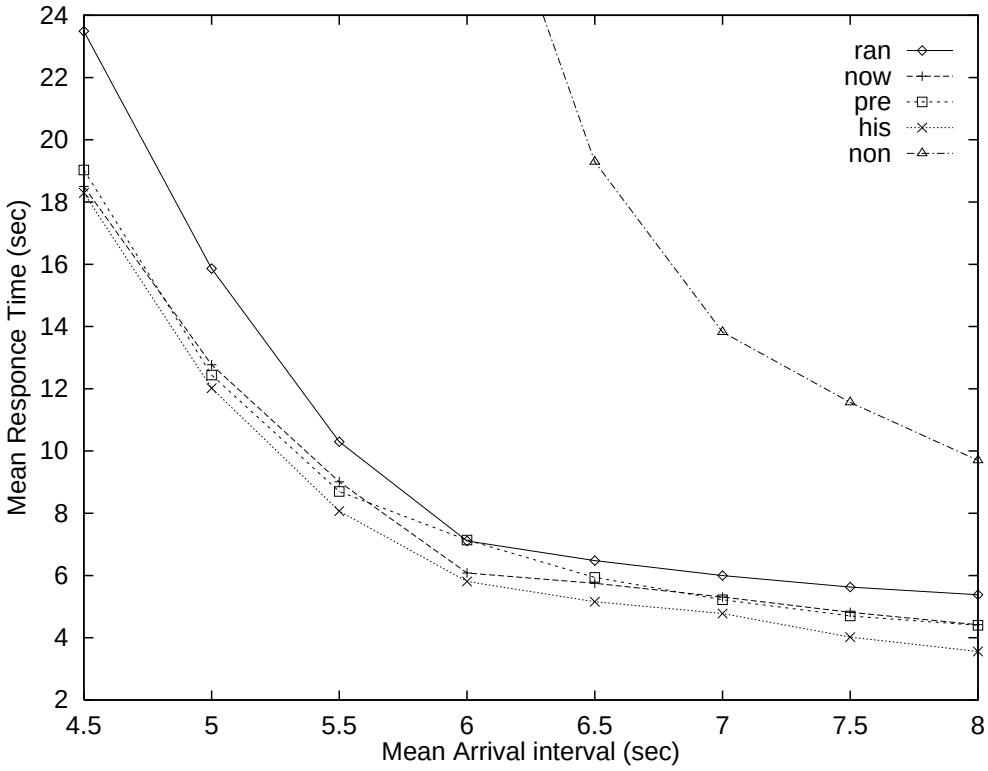

図 3.8: 能力が均一な分散システム (変動状態) での平均応答時間

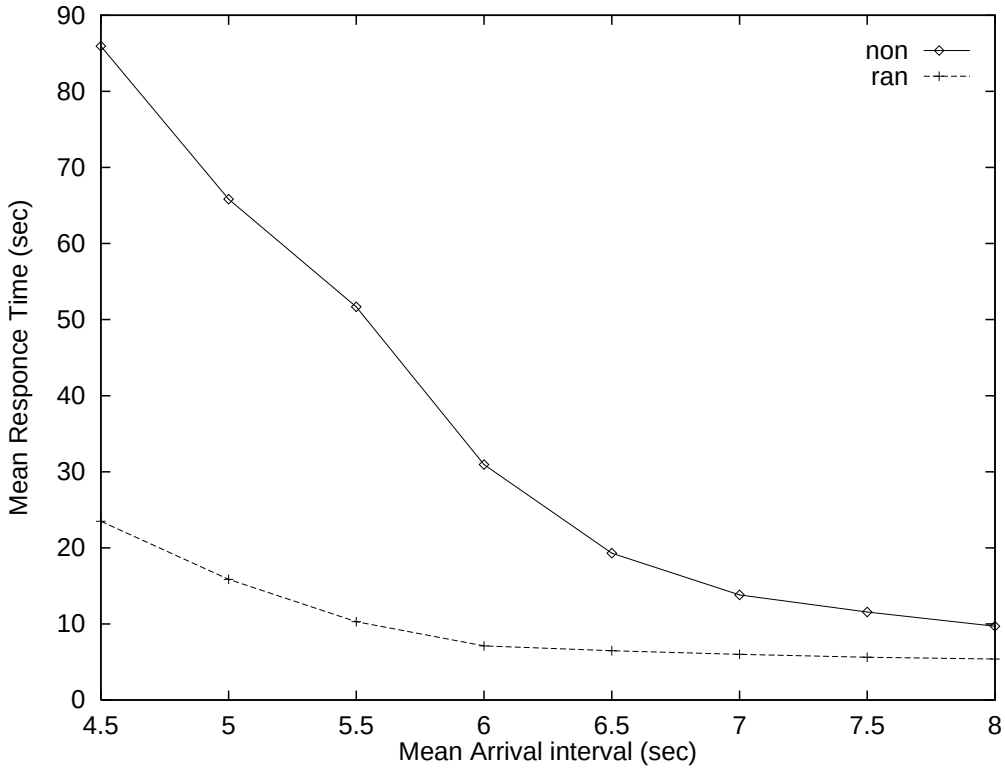

図 3.9: 分散の効果 (変動状態)

計算機能力が異なる分散システムの場合

計算機として SS20 , SS10 , S-4/2A(表 3.1) の 3 台を選択し , プロセスは平均 7.0 ~ 10.0 (sec) のポアソン分布に従って発生させ , シミュレーションを行った .

そのときの平均応答時間を図 3.10 に示す .

各選択アルゴリズムの平均応答時間に大きな差がみられた . これは , 計算機間に能力差があるため選択の適切さが結果に現れたものと考えられる . また , その結果も負荷予測の効果がきちんと現れていた . 特に , his の選択アルゴリズムの平均応答時間が他の選択アルゴリズムと比較して , プロセスの平均発生間隔が 7.0 (sec) , つまり , 計算機がわりと忙しいときに , ran に対して約 40 % , now に対して約 34 % , pre に対して約 28 % 短縮されていることが確認でき , 提案方式がより適切な選択方法であることを示している .

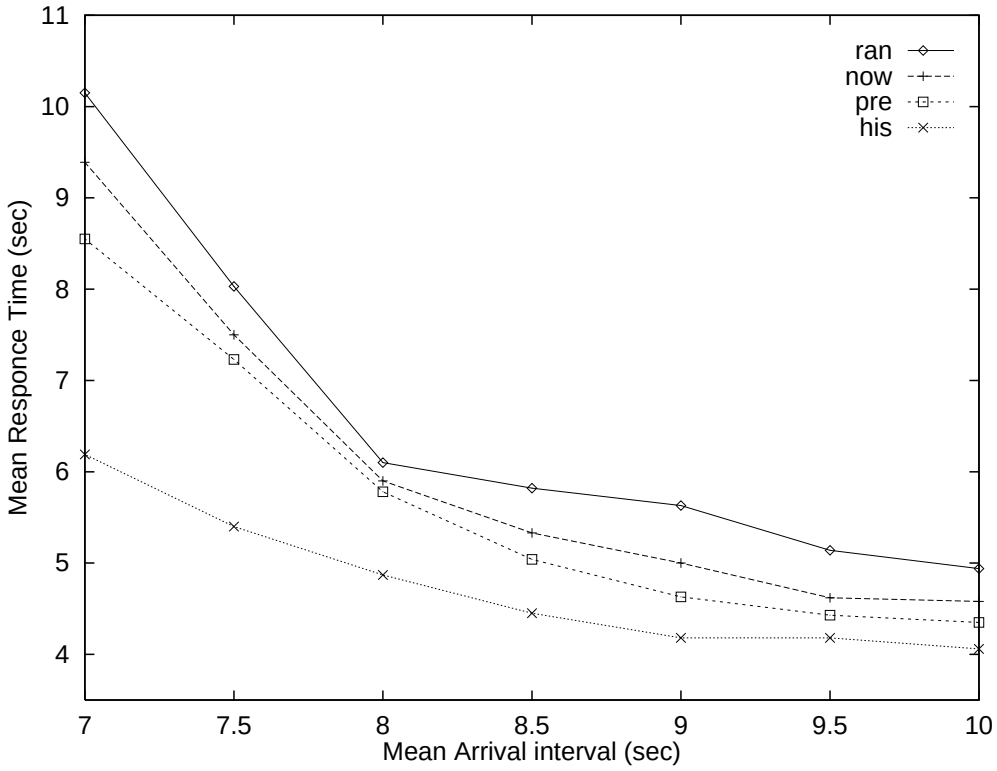

図 3.10: 能力が異なる分散システム (変動状態) での平均応答時間

3.6　まとめ

負荷分散をより効果的に行うためには，負荷の変動を動的に考慮しながらプロセスを割り当てる必要がある．しかし，負荷変動の監視のコストは分散システムのパフォーマンスを大きく左右する．

本章では，負荷の監視コストを下げるために負荷の監視を低頻度 (一定間隔) で行い，その間を負荷予測によって補う手法を提案し，この機構を組み込んだ分散処理システム (モニタ) を開発した．また，負荷予測を用いた動的負荷分散アルゴリズムとして，

- 単純な予測選択アルゴリズム (3.4.1)
- 履歴を用いた予測選択アルゴリズム (3.4.2)

を提案し，分散処理モニタにおいて提案アルゴリズムの有効性を確認した．

第 4 章

自律的仮想分散システム環境

4.1 はじめに

分散システムの規模が大きくなる．つまり，計算機の台数が増加すればするほど，計算機間で行われる負荷情報や状態情報の通信は指数関数的に増大する．特に，本研究において第 3 章で提案した予測を用いた負荷分散アルゴリズムでは，負荷の予測を一定間隔ごとに行うため，その通信コストは計算機の台数が増大すると，無視することができなくなる．

図 4.1は，3.3で設計した分散処理モニタ (図 3.4) において，分散システムの計算機台数の増加に伴い，通信量がどのように変化するかをシミュレーションした結果である．(シミュレーション時間は各計算機において最低 300 個のプロセスが発生するまでとした．) この結果から，計算機台数が 3 台のときと 10 台のときとでは通信数が約 10 倍近くになっていることがわかる．

本章では，本提案モデル (アルゴリズム) において実際に分散処理を行う際に無駄になっていると考えられる通信をできるだけ省き，分散処理システムとしての性能を維持したまま，通信数を削減することを目的とする．

本章では以下の内容について述べる．

- 分散処理モニタの改良
- 仮想分散システム環境の形成

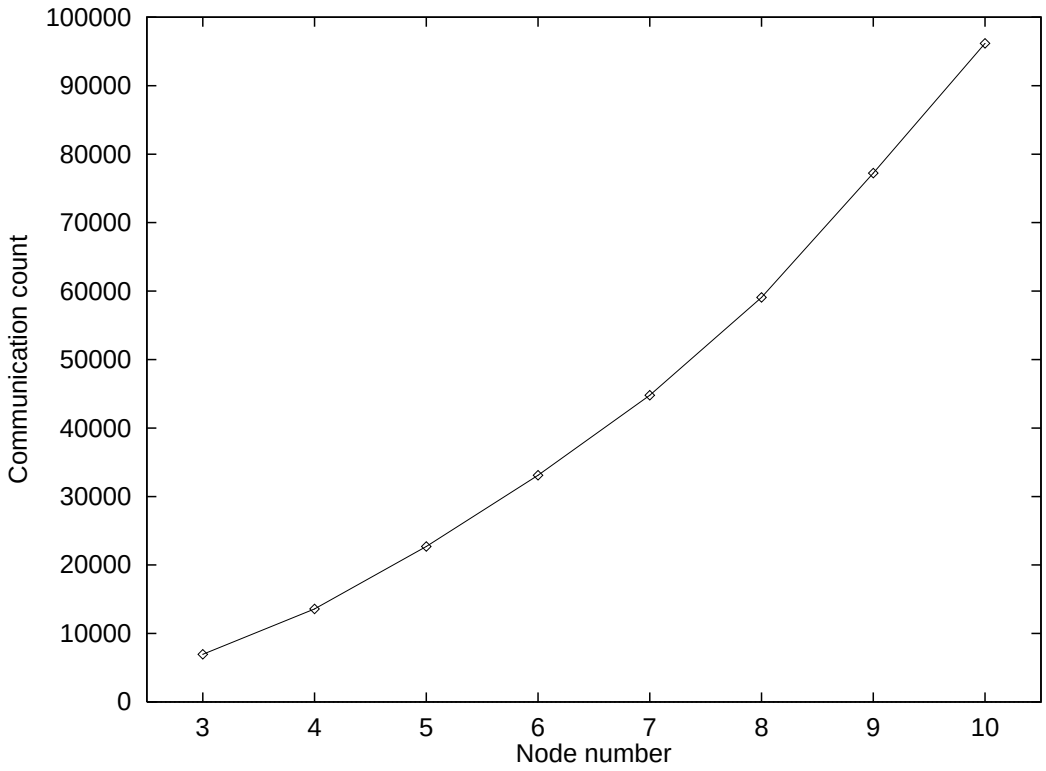

図 4.1: 計算機台数に伴う通信数の増加

4.2　分散処理モニタの改良

3.3で設計した分散処理モニタ (図 3.4) で，実際の分散処理の流れを観察していると，特に負荷の変動が大きいときに 1 つのプロセスが計算機間でたらい回しになることが確認できた．これは，分散システムの平均応答時間が悪くなるばかりでなく，通信数の増加にもつながり，分散システムの性能を下げていることになる．

本節では，これを防ぐために 3.3で設計した分散処理モニタ (図 3.4) を改良する．
改良した分散処理モニタを図 4.2に示す．
改良はプロセス処理系について行った．改良点は，以下のとおりである．

- Queue をその計算機で発生するプロセスを格納する My Queue と，他の計算機から依頼されたプロセスを格納しておく Other Queue に分割する．
- Other Queue にはプロセスを格納できる最大数を設けておき，その数を越えてプロセス処理依頼が到着するとプロセスを処理依頼計算機に送り返す．
- Switcher を設けておき My Queue , Other Queue に格納されているプロセスのうち，プロセスの発生時間の早いものから処理を開始する．

つまり，この改良により受理されたプロセスは必ずその計算機で処理されることになる．
また，プロセス処理系のフローチャートを図 4.3に示す．なお，負荷予測系は前のモニタと同様，カオスを用いた非線形予測でそのフローチャートも図 3.6のとおりである．

分散処理モニタの改良による分散システムのパフォーマンスについては，次節 (4.3) で提案する仮想分散システム環境もまじえて，4.4節のシミュレーションで考察する．

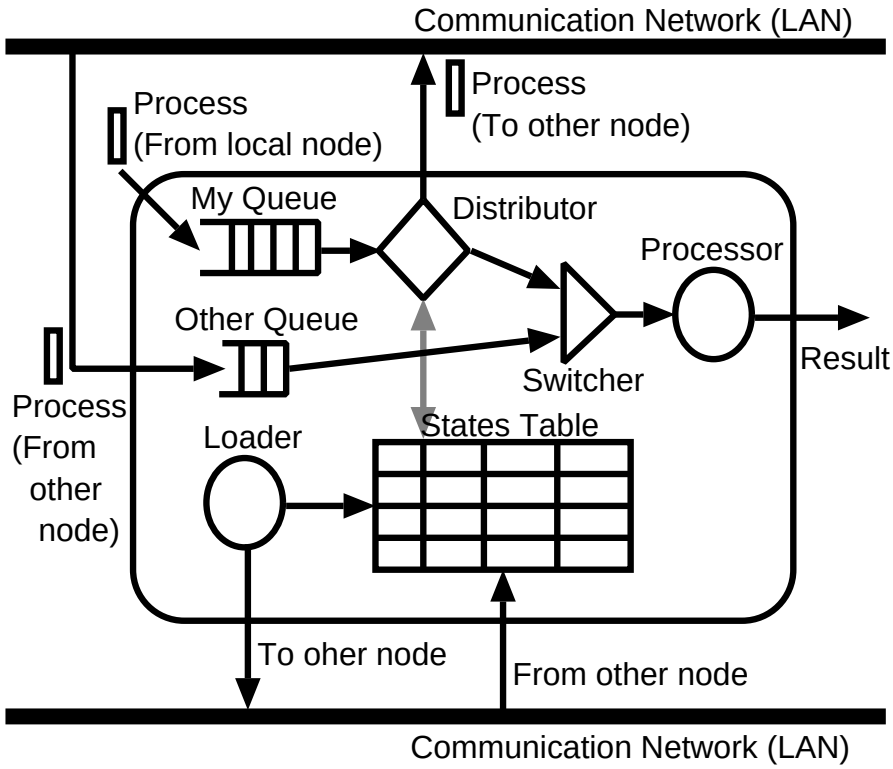

図 4.2: 改良型分散処理モニタ

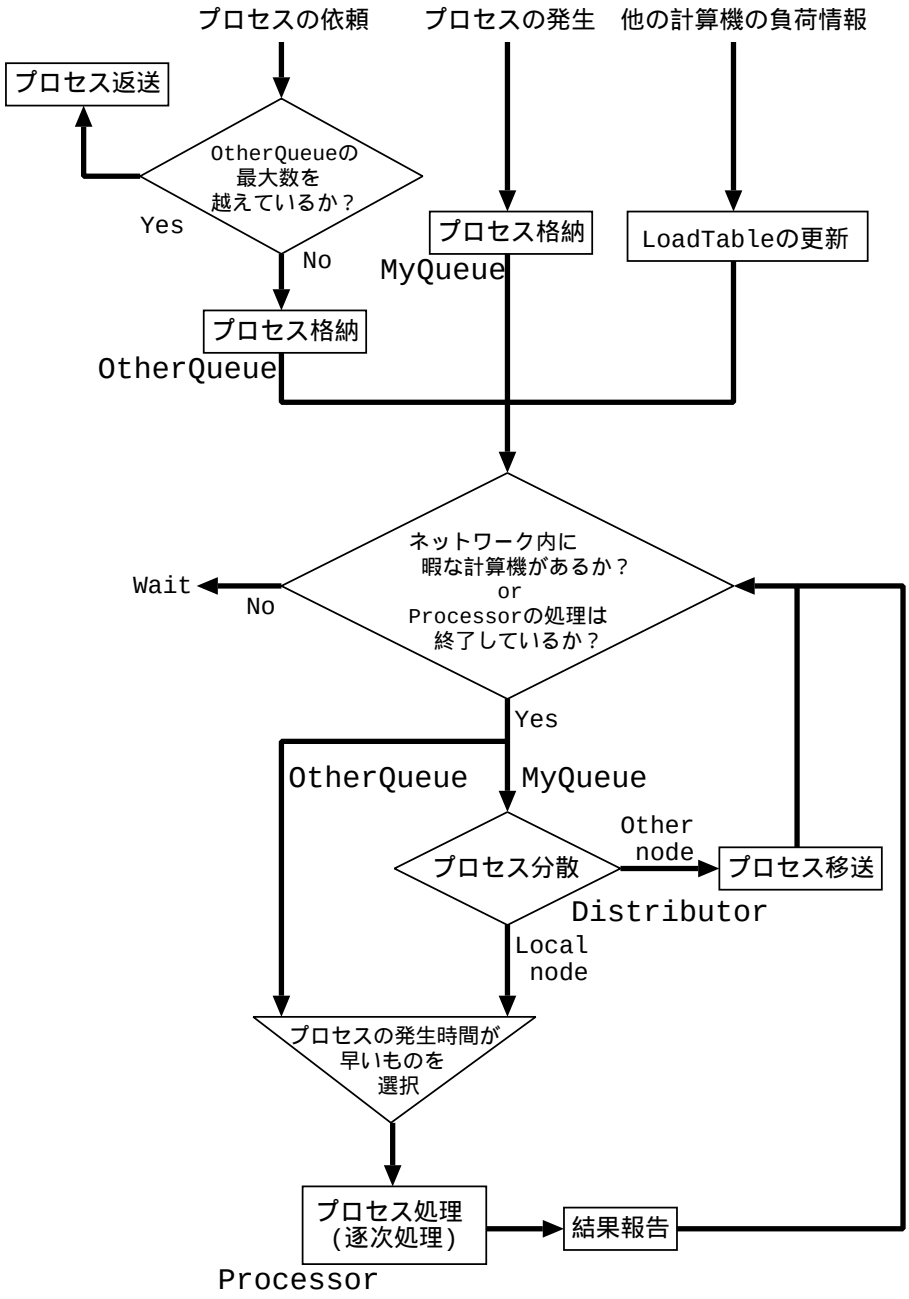

図 4.3: プロセス処理系のフローチャート (改良型)

4.3　仮想分散システム環境の形成

分散システムにおける負荷情報，状態情報の通信は，プロセスを効果的に分散するのに必要不可欠である．しかし，プロセスが次から次へと生起し，分散，処理しなければならない計算機が 2 台あるとすると，その計算機間での情報の交換はあまり意味をなさない．つまり，どちらかの計算機がもう一方の計算機にプロセスを依頼することは，まず起こり得ない (起こったとしてもそれは意味をなさない)．同様に，全くプロセスが生起せず暇な計算機間での情報の交換もあまり意味がない．

本節では，これら無駄だと考えられる通信をできるだけ削減するため，各計算機に結合度という概念を導入し，各計算機で自律的に自分に必要な分散環境を作り出す手法を提案する．

4.3.1　結合度の導入

各計算機間での結び付きを表すものとして結合度 : C を提案する．
結合度は-3 から+3 までの連続値で，大きくなればなるほどその計算機との結び付きが強いことを示します．そして，その値より 3 つの状態を決定 (図 4.4) し，その状態ごとに計算機間の情報の交換を変化 (表 4.1) させます．
表 4.1において，Cold の結合状態にある計算機間では負荷情報も状態情報も通信されません．これは，この状態にある計算機間では相手の存在が意識されないことになり，この結合度を導入した分散システムでは，複数の仮想の分散システムが存在することになります．

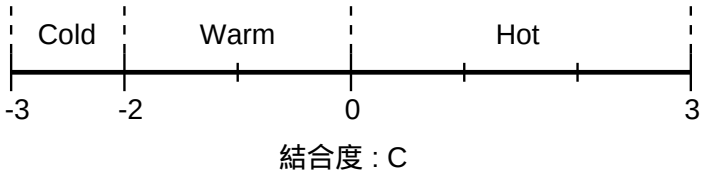

図 4.4: 計算機間の結合レベル

結合状態	負荷情報	状態情報
Hot	○ $(T_{inteval})$	○
Warm	○ $(2T_{inteval})$	×
Cold	×	×

表 4.1: 結合状態における通信情報の変化

結合度はプロセスの移送に対して変化します．具体的にはプロセス依頼を受理されればその計算機間での結合度は増加し，拒否されれば結合度は減少します．

- プロセス依頼を受理
  - 受理計算機 (i) との結合度増加 $(C_a > 0)$
  - それ以外の計算機 (j) との結合度減少 $(C'_a < 0)$
- プロセス依頼を拒否
  - 拒否計算機 (i) との結合度減少 $(C_r < 0)$
  - それ以外の計算機 (j) の結合度増加 $(C'_r > 0)$

その結合度の変化の様子 (4 台の計算機 A,B,C,D で構成される分散システム) を図 4.5 (プロセス依頼を受理), 図 4.6 (プロセス依頼を拒否) に示す．図中の点線がその計算機間の結合度を表し，太いほど両者の結び付きが強いことを表す．

また，このときの変化分を以下の式で定義します．

- プロセスを依頼した計算機 (Request Node)

$$C_x(i) = w_x \times \frac{N_{send} - N_{acc}(i)}{N_{send}} \tag{4.1}$$

$$C'_x(j) = -C_x(i)/(n-2) \tag{4.2}$$

- プロセスを依頼された計算機 (Receive Node)

$$C_x(i') = w_x \times \frac{N_{recv} - N_{acc}(i')}{N_{recv}} \tag{4.3}$$

$$C'_x(j') = -C_x(i')/(n-2) \tag{4.4}$$

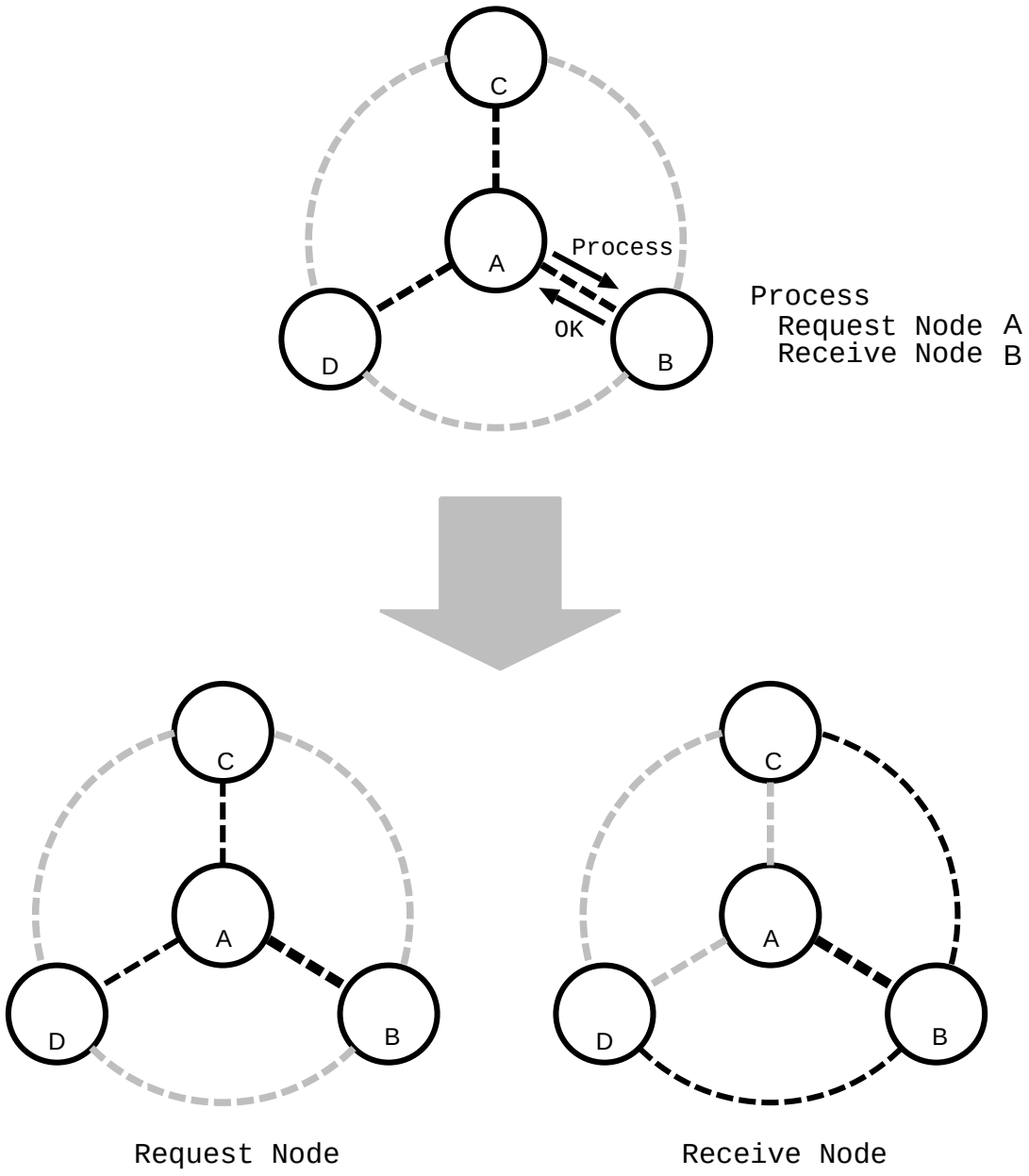

図 4.5: 計算機間の結合度の変化 (プロセス依頼受理)

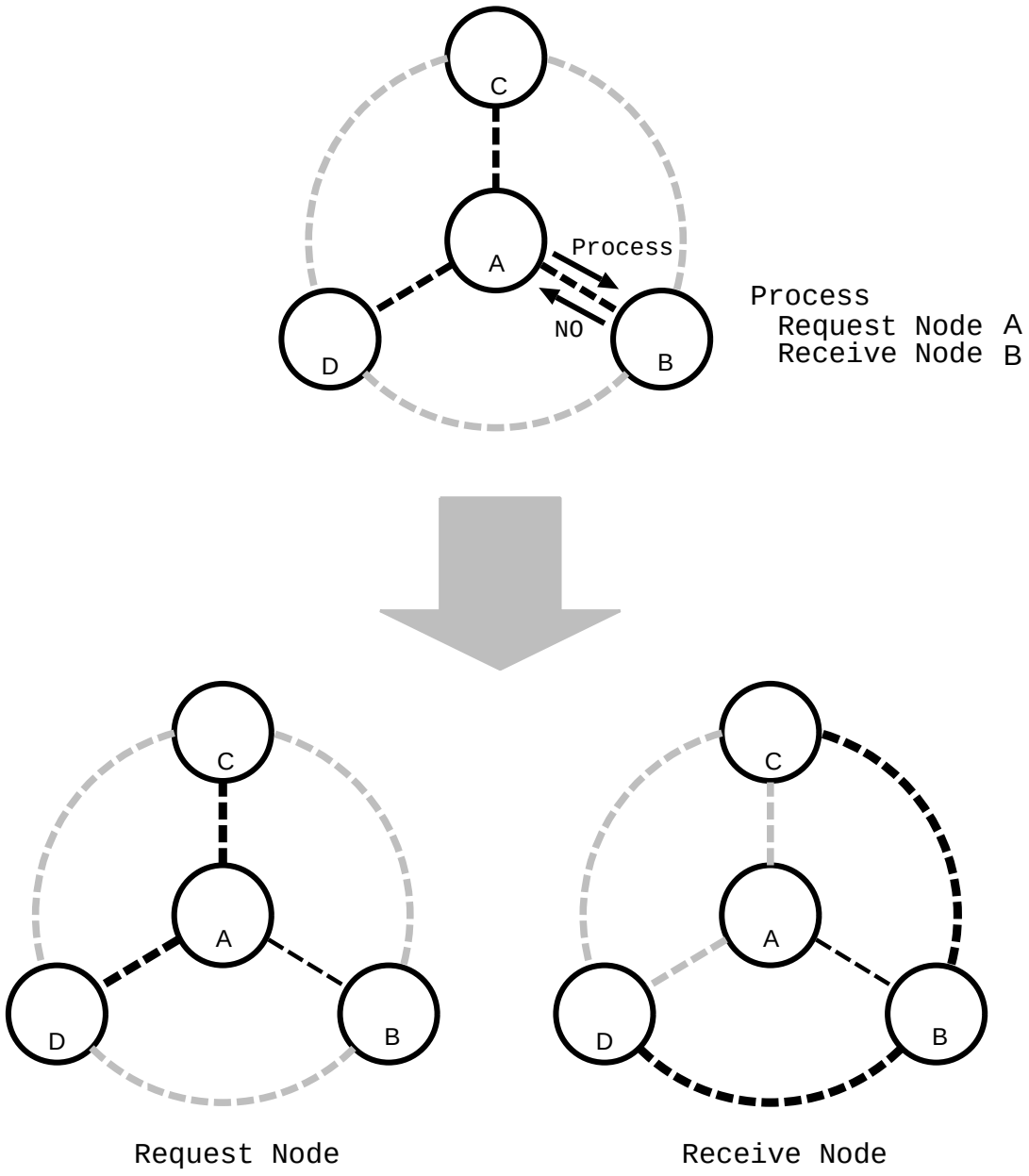

図 4.6: 計算機間の結合度の変化 (プロセス依頼拒否)

ここで，計算機 $i = 1 \sim n, j \neq i$，N_{send}はプロセスを依頼した total 数，N_{recv}はプロセスを依頼された total 数，$N_{acc}(i)$ は計算機 i の受理数，w_xは重みである．$(x = a \text{ or } r)$
式 (4.1) で示される変化分は，その計算機との結び付きが強い (結合度が大きい) ほどその値が飽和するような式になっていて，逆にいえば，結び付きの弱い (結合度が小さい) 計算機間で，プロセスの依頼，受理が成立すれば結合度は一気に大きく増加する式になっています．

また，今のままでは一度結合状態が Cold になると，その計算機と結び付きを復活させるのは容易ではありません．これでは負荷の変化に完全に対応できるはいえず，動的に負荷分散していることにはつながりません．そこで本研究では，Cold 状態からの復帰を容易にするために，表 4.2で示される 2 つの Signal を導入し，一度切れた結合がもう一度簡単に復活するようになっています．

Help signal	計算機のキュー数がある閾値を越えると発生
Idle signal	一定時間プロセス処理をしないと発生

表 4.2: 結合の復活を容易にするための Signal

4.4　シミュレーション

4.2，4.3で提案したモデルがどの程度，通信数を削減し，かつ，分散システムとしてのパフォーマンスはどのように変化したかを調べるため，シミュレーションを行った．

4.4.1　シミュレーション条件

- LAN によってつながれた 10 台の計算機 (S-4/20) を使用．
- 負荷分散アルゴリズムは，3.5で一番結果のよかった履歴を用いた予測選択アルゴリズムとする．
- Other Queue の最大値は 2 とする．
- プロセスは各計算機でポアソン分布に従って独立に生成される．
- プロセスは，四則演算からなる単純な計算とする．
- 計算機の負荷の状態は変動状態 (3.5.1) とする．
- プロセスの処理時間と比較して Distributor の処理時間は十分無視できる．

4.4.2　性能評価

分散システム規模に対する評価

分散システムの規模 (計算機台数) を変化させたときの，通信数とシステム全体の平均応答時間を，以下の 3 つのモデル (表 4.3) について比較シミュレーションを行った．

old	前モデル (図 3.4)
new1	Queue を分割しプロセスのたらい回しを防いだモデル (図 4.2)
new2	結合度を導入 (仮想分散環境の形成)

表 4.3: 比較モデル

使用計算機は同一処理能力であり，プロセスの条件は以下のようにした．

- 無負荷時におけるプロセスの処理時間 ... 2.4 [sec]
- プロセスの平均発生間隔 ... 6.0 [sec]

また，シミュレーション時間は，各計算機において最低 300 個のプロセスが発生するまでとした．

シミュレーション結果を図 4.7に示す．今回提案したモデルが，計算機台数の増加に対し前モデル (old) より通信数の増加を抑えることができることが確認できた．これは，各計算機が自律的に情報を必要とする計算機を選び出し通信していることの現れである．また，そのときの分散システムのパフォーマンスを表わす平均応答時間も，new1 には若干の増加も見られたが，new2 においては old とほぼ変わらない結果を示した．今回のシミュレーションでは，無負荷時におけるプロセスの処理時間と，プロセスの平均発生率を比較するとわかるとおり，各計算機にわりと余裕があるためにこのような結果になったと考えられる[1]．

[1] しかし，前節で述べたように各計算機の負荷は変動状態であるので，一概に各計算機に余裕があるかどうかは難しい議論となる

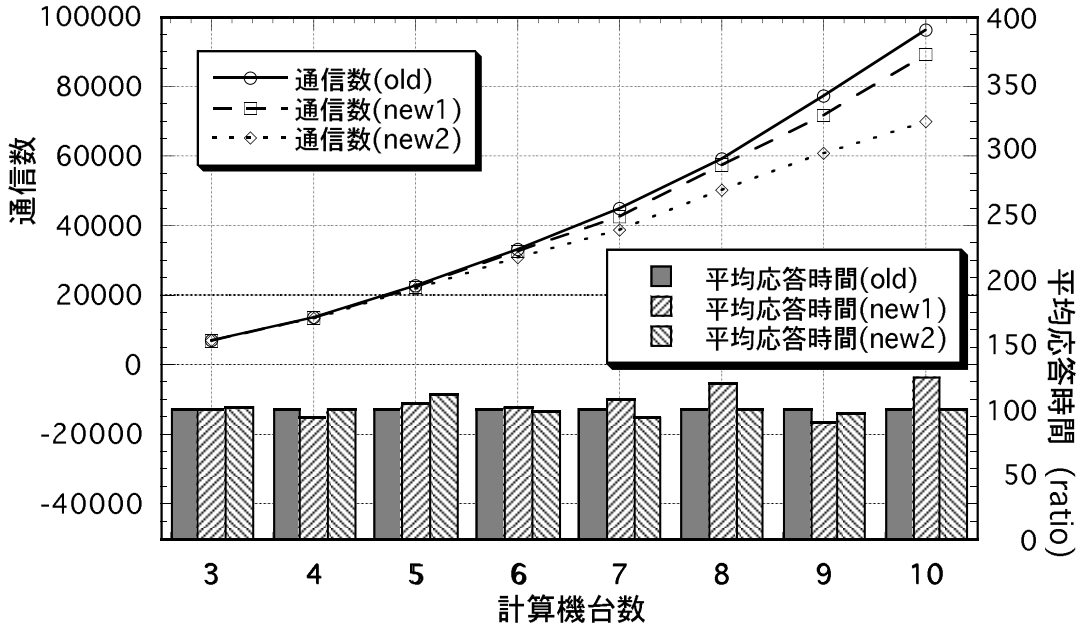

図 4.7: 提案モデルの分散システム規模によるパフォーマンス変化

プロセスの到着間隔に対する評価

仮想的な分散環境を作り出すためのコストと，それによって削減される通信コストの関係を調べるため，各計算機でのプロセスの平均到着間隔を変化させたときの，通信数とシステム全体の平均応答時間を表 4.3の old, new2 の 2 つモデルで比較シミュレーションを行った．また，分散環境によって結合度による影響がどのように変化するかを観測するために，分散環境が同一処理能力の計算機で構成される場合と，異なった処理能力の計算機で構成される場合の 2 通りについても比較シミュレーションを行った．

シミュレーション結果ををを図 4.8, 図 4.9に示す．

図 4.8の分散環境の条件は，

- 同一プロセス処理能力の 10 台の計算機を使用
- 各計算機の無負荷時におけるプロセスの処理時間 ... 5.0[sec]

図 4.9の分散環境の条件は，

- 標準プロセス処理 5 台，高速処理 5 台の計 10 台の計算機を使用
- 標準計算機の無負荷時におけるプロセスの処理時間 ... 5.0[sec]
- 高速計算機の無負荷時におけるプロセスの処理時間 ... 2.5[sec]

である．

また，シミュレーション時間は，各計算機において最低 300 個のプロセスが発生するまでとした．

図 4.8, 図 4.9から，プロセスの平均到着間隔がどう変化しようとも，通信量を抑えることができることが確認できた．また，分散環境に変化に対しても特に通信量の削減量に大きな差は確認できなかった．単純に分散環境を構成する計算機間の能力に大きな差があればあるほど通信量は削減されると考えていたが，今回のシミュレーションの規模，条件ではそこのところは現れてこなかった．もっと大きな分散環境やスーパーコンピュータを含むような環境でなら，通信量の削減量に差が出てくると考えられ，これからの課題である．

分散システムの平均応答時間は，プロセスの平均到着間隔が広いとき，つまり各計算機がわりと暇なときは old と new2 の間ではほぼ同じような値を示したが，間隔が狭くなる

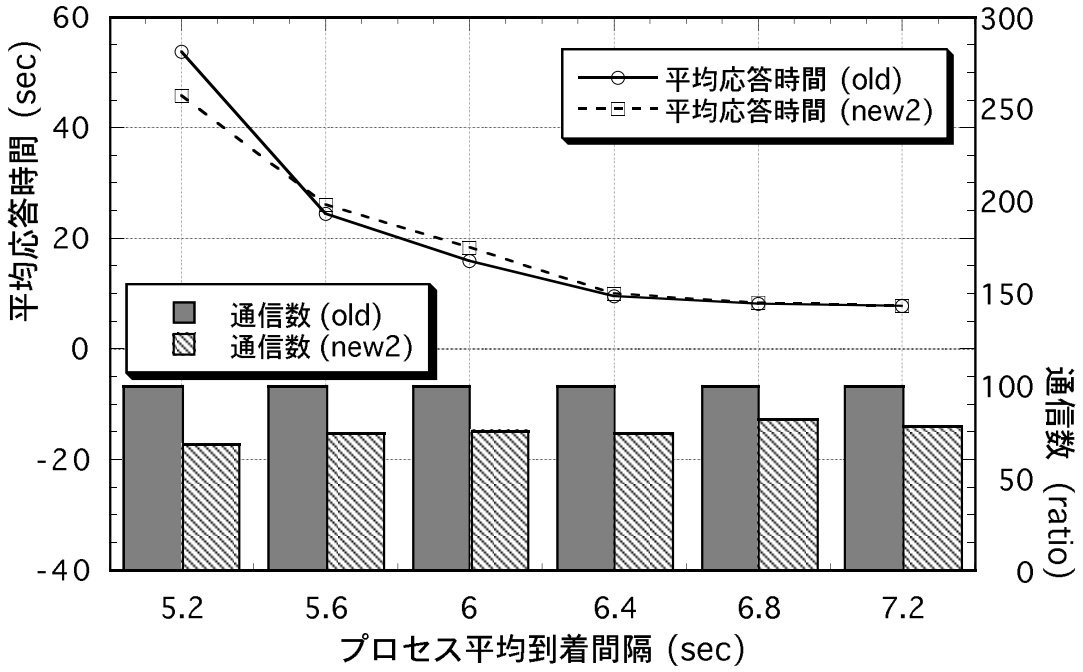

図 4.8: プロセス到着間隔によるパフォーマンス変化 (各計算機能力は均一)

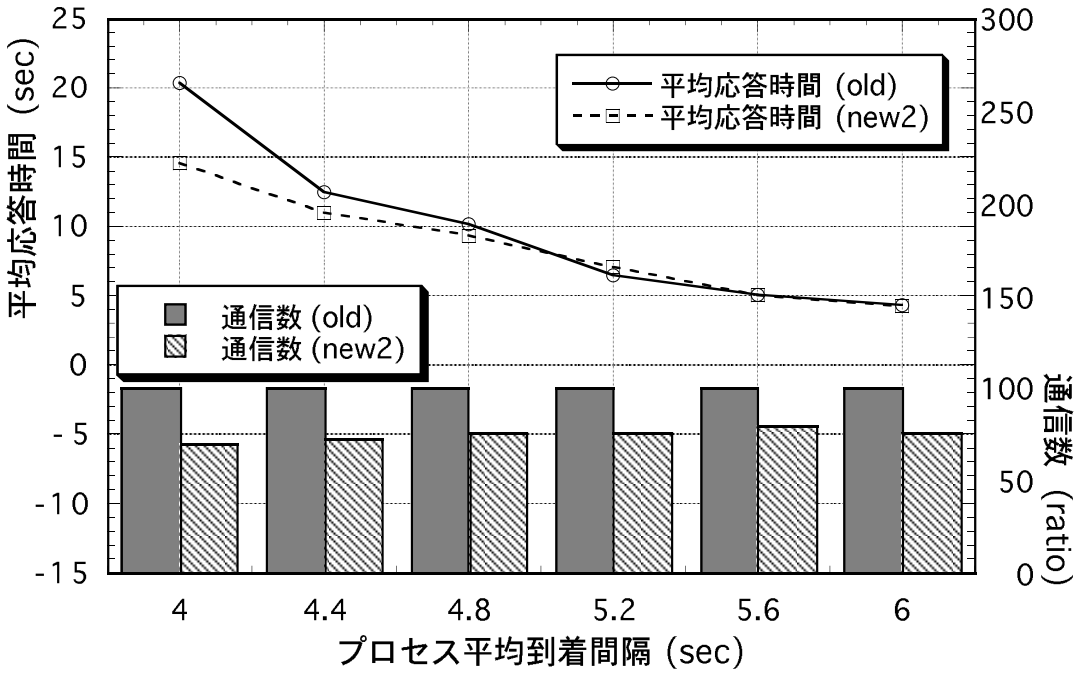

図 4.9: プロセス到着間隔によるパフォーマンス変化 (各計算機能力は不均一)

と，若干，new2 のほうが応答時間が悪くなる傾向が見られた (特に図 4.8)．これは，仮想的な分散環境を作り出すためのコスト，つまり結合度の導入による計算機選択の幅が狭くなったことが，システムのパフォーマンスを低下させていると考えられる．しかし，さらにプロセスの平均到着間隔が狭くなると，逆に new2 のほうが応答時間がよくなっている．これは，本モデルの通信プロトコルはソケットを用いた UDP(User Datagram Protocal) を用いてリクエスト-リプライ-ACK(RRA) プロトコルでプログラミングをしているため，通信がうまくいかないときは，再通信をかけるようになっている．つまり，これが増えること (表 4.4, 表 4.5) によって分散システムのパフォーマンスが低下したと考えることができる．また分散環境の変化に対しては，分散システムを構成する計算機間に差がある方が，今回提案した手法がより有効に働いているような結果を示した．これは，計算機間に差があるため，仮想的な分散環境がより明確に形成され，負荷や状態の通信のコストや処理計算機選択のコストが低くなることによりこのような結果が現れたと考えられる．

しかし，再通信が数多くかかる状態は本プログラムではシステムが不安定な状態に近いので，仮想的な分散環境を作り出すためのコストと，通信コストの関係を正確に議論するためには，もう少し安定したシステムのもとでシミュレーションをする必要があると思われる．

平均到着間隔	再通信数	
(sec)	old	new2
5.2	45	11
5.6	16	8
6.0	5	7
6.4	0	1
6.8	1	2
7.2	0	0

表 4.4: 均一な計算機で構成される分散システム (図 4.8) での再通信数

平均到着間隔	再通信数	
(sec)	old	new2
4.0	59	10
4.4	12	8
4.8	7	7
5.2	3	2
5.6	0	0
6.0	0	0

表 4.5: 不均一な計算機で構成される分散システム (図 4.9) での再通信数

4.5　まとめ

分散システムの規模の増加に伴う通信量の増加を抑えるため，3.3で設計した分散処理モニタ (図 3.4) を改良し，また，結合度という概念を導入して，仮想的な分散システム環境を自律的に形成する手法を提案した．

シミュレーションの結果から，仮想分散システム環境が仮想的な概念を導入しない計算機全体で形成する分散システム環境と比較して，システムの性能をほぼ維持したまま，通信量の削減に成功していることが確認できた．特に，分散システムを構成する計算機間に差が大きい場合，また各計算機の通信が飽和状態に近づくような環境では，本手法が有効に働くことが確認できた．

第 5 章

結論・検討

5.1 結論

本研究では『並列分散処理システム』の上で，重要な機能である負荷分散について注目しながら，種々の評価を行った．

第 2 章では，負荷の分散基準に欠かすことのできない計算機の負荷の監視を低頻度で行いながら対処する手法として，負荷の予測を提案し，次の 2 つのモデルを考察した．

- 線形予測モデル ... n 次元多項式モデル (負荷を対数正規分布に従うと仮定)
- 非線形予測モデル ... カオス予測モデル

実際に稼働している計算機において両モデルを比較したところ，線形予測モデルよりも，カオスを用いた非線形予測モデルの方がよい予測をおこなうことが確認された．

第 3 章では，第 2 章で提案した負荷の予測を組み込んだ分散処理システム (モニタ) を開発し，負荷予測を用いた動的負荷分散アルゴリズムとして，

- 単純な予測選択アルゴリズム
- 履歴を用いた予測選択アルゴリズム

を提案した．

分散処理モニタにおいて，提案アルゴリズムを予測を用いないアルゴリズムと比較し，その有効性を確認した．

第 4 章では，分散システムの規模の増加に伴う通信量の増加を抑えるため，第 3 章で設計した分散処理モニタを改良し，また，結合度という概念を導入して，仮想的な分散システム環境を自律的に形成する手法を提案した．その結果，システムの性能を維持したまま，通信量の削減に成功した．また，提案手法が分散システム構成する計算機間に差が大きい場合，また各計算機の通信が飽和状態に近づくような環境では，有効に働くことが確認できた．

5.2　検討

本研究において，分散プロセスの粒度，各閾値など，各種パラメータの設定を一意に決めることは難しい．これは実際のシステムのチューニング同様，自分の望む環境を作り出すことに似ている．本研究では，プロセスの粒度をまず決めた段階で発見的に閾値を決定したが，実際にシステムとして運用する際には，そのガイドライン的なものが必要になると考えられ，これからの課題といえよう．

また，本研究では負荷を予測しながら負荷分散を行う手法を提案したが，負荷の予測にかかるコストと分散システムとしてのパフォーマンスの向上率にどのような関係があるかを調べるのも課題の 1 つである．

謝辞

本研究を進めるに当って，全般的に御指導いただいた，東北大学工学部阿曽弘具教授に心より感謝いたします．

本研究をまとめるに当って，適切な御助言をいただいた，東北大学工学部白鳥則郎教授，川又政征教授に大変感謝いたします．

いつも熱心に御討論していただいた，山口大学経済学部助教授成富敬氏，東北大学大学院工学研究科博士課程藤岡豊太氏をはじめ，東北大学工学部通信工学科阿曽研究室並列グループの皆様に深く感謝いたします．

また，日頃より研究活動を共にし，計算機環境を常に快適な状態に整備していただいた，東北大学工学部通信工学科阿曽研究室の皆様，ならびに同研究室の同級生諸氏に深く感謝いたします．

最後に本研究を成し遂げることができたのも，日頃影ながら支えてくれた両親，友人たち，そして五十嵐さおり嬢あっての成果であり，ここに心から深く感謝いたします．

参考文献

[1] 清水 謙多郎: “分散オペレーティングシステム”
信学誌, Vol.73, No.9, pp.977–983, Sep. 1990 .

[2] J.Bacon : “CONCURRENT SYSTEM”
Addison-Wesley Publishing Company,Inc. ,1996.

[3] Niranjan G.Shivaratri, Phillip Krueger and Mukesh Singhal : “Load Distributing for Locally Distributed Systems”
IEEE, Computer, Dec. 1992.

[4] Katherine M.Baumgartner and Benjamin W.Wah : “GAMMON: A Load Balancing Strategy for Local Computer Systems with Multiaccess Networks”
IEEE, Trans.on Computers, vol.38, No.8, Aug. 1989.

[5] 近藤 次郎 :「数学モデル」
丸善株式会社 .

[6] ジョン ハル :「デリバティブ入門」
金融財政事情研究会 .

[7] 辻 敦宏, 上野 仁, 山本 幹, 池田 博昌 : “統計的負荷情報を用いた自律負荷分散制御”
信学技報, CPSY95-61, 1995 .

[8] 杉浦 弘幸, 市川 周一, 島田 俊夫 : “履歴を考慮した動的負荷分散法”
信学技報, VLD95-106, 1995 .

[9] 五百旗頭 正, 木村 孝, 合原一幸 : “決定論的非線形短期予測手法の上水道需要量データへの応用”
電学論 D, 144(4), 1994 .

[10] F.Takens : “In Dynamical Systems and Turbulesnce”
(eds. Rand and Young), pp.366-381, Springer, Berlin, 1981.

[11] 合原, 五百旗頭 :「カオス応用システム」
朝倉書店, 1995 .

[12] 合原 一幸 :「カオス -応用をめざして」
数理科学, 1992 .

[13] 合原 一幸 :「カオス -カオス理論の基礎と応用」
サイエンス社, 1990 .

[14] E.Ott : “Strange attractors and chaotic motions of dynamical systems”
Rev.Modern Phy, 53(4/1), pp.655-671, 1981.

[15]「数学セミナー」
日本評論社, Dec. 1992 .

[16] 岩坪 , 萬谷 , 池口 , 合原 : “再構成軌道を用いた小規模電力需要の非線形短期予測”
電気学会産業システム情報化研究会資料, 11S-93-7-11, pp.23-28, 1993 .

[17] 五百旗頭 正, 木村 孝, 合原一幸 : “決定論的非線形短期予測手法の上水道需要量データへの応用”
電学論 D, 144(4), 1994 .

研究業績

- “負荷予測を用いた動的分散処理システムとその評価”
  鶴田 進，阿曽弘具
  電子情報通信学会コンピューテーション研究会，COMP96-19(1996)