卒業論文

プロセス情報を活用する
実行時間予測に関する研究

東北大学 工学部 通信工学科
阿曽研究室　4 年
丹野 祐樹

目 次

第 1 章　序論 **1**
1.1　研究の背景 1
1.2　研究の目的 2
1.3　本論文の構成 2

第 2 章　CPU 実行時間予測 **3**
2.1　定義 3
2.2　CPU 実行時間予測手法の概要 4
2.3　類似プロセスの取得 5
2.4　仮予測値の算出 5
2.4.1　Normal, Args 時の仮予測値算出 5
2.4.2　Mem, Mem + Args 時の仮予測値算出 6
2.5　信頼度による予測手法の選択 8
2.5.1　t 分布 8
2.5.2　信頼度の設定 8

第 3 章　実実行時間補正 **10**
3.1　計算タスク数の確認 10
3.2　計算タスク数に応じた実実行時間補正 11

第 4 章　実験 **13**
4.1　実験環境 13
4.2　CPU 実行時間予測実験 14
4.3　信頼度選択の有効性 15
4.4　実実行時間予測実験 16
4.5　まとめ 17

第 5 章　結論 **18**
5.1　本論文の成果 18
5.2　今後の課題 18

参考文献 **20**

謝辞 **21**

図 目 次

2.1 プロセス情報の取得例 4
2.2 CPU 実行時間予測の流れ 4
2.3 クラス間分散による分割例 6
2.4 実メモリ使用量の累積誤差算出範囲 7
2.5 累積誤差算出に使用される過去状態の選択例 8
2.6 t 分布の概形 9

3.1 搭載 CPU 数 = 2 の場合の延長区間 11

4.1 CPU 実行時間予測結果 15
4.2 信頼度 0.9 以上の CPU 実行時間予測結果 16
4.3 実実行時間予測結果 16

表 目 次

4.1 実験条件 13

第1章

序論

1.1 研究の背景

近年, 複数の計算機による並列分散処理の利用が拡大している. 計算機を高速ネットワークで接続し, 仮想的に一つの計算機環境を構築する計算機クラスタや, インターネットを用いて数多くの計算機の余剰計算資源を集め, それぞれに処理を分担させるグリッドコンピューティング等が並列分散処理の例である.

並列分散処理環境の利用が拡大している要因の1つとして, コストパフォーマンスの良さが挙げられる. よく比較される計算機環境として, スーパーコンピュータと呼ばれるものが存在する. これは非常に高い処理性能を実現できるのだが, その特殊なハードウェア構成のため, 製造・運用コストが非常に高いという問題がある. 対して計算機クラスタやグリッドコンピューティングは, 例え処理性能がそれほど高くない計算機であっても, 並列分散処理をさせることで全体としての処理性能を向上させることができる. そのため, スーパーコンピュータよりもはるかに低いコストで環境を整えることができる.

並列分散処理環境の効率的な利用には, 実行中プロセスの実行時間予測が非常に有用となる. 計算機クラスタを例に挙げると, ユーザーが新たにプロセスを投入する際, クラスタを構成する各マシンで実行中にあるプロセスがどの程度の時間で実行完了となるかが分かれば, どのマシンにプロセスを投入すれば早く実行完了するか分かる. よって, 複数のプロセスが投入された際, 実行時間予測を用いてスケジューリングを行なえば, 全体としての処理完了時間を改善できる.

実行時間予測は, 実行中プロセスと類似したプロセスが過去に実行されていれば, ある程度の予測が可能だと考えられる. 何故なら, プロセスの実行時間は, 全く共通点のないプロセスの実行時間よりは, プロセス名等で何か共通点のあるプロセスの実行時間に近い値をとると推測されるためである.

Smith ら [2] は, 過去に実行された類似プロセスの実行時間を用いた実行時間予測の有効性を示した. そして, ユーザ名, プロセス名といった, 類似プロセスを検索する際に使用する情報を適切に

選択することが重要だと述べている.

1.2　研究の目的

本研究では, 実行中プロセスの実行時間を高精度に予測することを目的とする.
本研究で提案する実行時間予測手法は, 以下に示す 2 段階にて行なわれる.

1. CPU 実行時間予測
2. 実実行時間補正

まず, 過去に実行された類似プロセスを元に, 現在実行中にあるプロセスの CPU 実行時間予測値を算出する. その後, 現在の状態に合わせて予測値を補正し, 実実行時間予測結果とする.

1.3　本論文の構成

本論文の構成は以下の通りである.

第 1 章　序論

研究の背景や目的, 論文の構成を述べる.

第 2 章　CPU 実行時間予測

プロセス情報を用いた CPU 実行時間予測ついて述べる.

第 3 章　実実行時間補正

CPU 実行時間予測を基にした実実行時間補正処理について述べる.

第 4 章　実験

予測実験の結果とそれに対する考察を述べる.

第 5 章　結論

本研究の成果, 今後の課題について述べる.

第2章

CPU 実行時間予測

本章では, プロセス情報を用いたCPU 実行時間予測手法について述べる.

ここで提案する予測手法は, 実行中プロセスのCPU 実行時間は過去に実行された類似するプロセスのCPU 実行時間と同じ値をとる, との仮定に基づいたものである. これは, 同じ処理や動作をするプロセスは, その実行時間も似ていると考えられるためである.

2.1 定義

まず, 用語や予測対象とするプロセスについて定義する.

本論文で使用する主な用語を以下に示す.

CPU 実行時間

あるプロセスと派生した子プロセスについて, CPU が処理を行なった時間

実実行時間

プロセス開始後の実際の経過時間

CPU 利用率

$$\text{CPU 利用率}\,[\%] = \frac{\text{CPU 実行時間}}{\text{実実行時間}} \cdot 100$$

プロセス処理はI/O 処理とCPU 処理の2つに大別することができる. I/O 処理とは, ユーザーからの入力やディスクの読み書きといったような, データ入出力処理のことである. 対してCPU 処理とはCPU による計算処理を意味する.

CPU 処理の占める割合が高いプロセスを計算タスクと呼び, 本研究ではこれらについて予測を行なう.

本研究で使用するプロセス情報は, Linux の ps コマンドを 一定時間毎に実行することでデータベース化している. ここで取得している情報は, ユーザー名, プロセス名, 経過実実行時間, 経過 CPU 実行時間, 使用メモリ量, 引数等がある. 取得例を図 2.1 に示す.

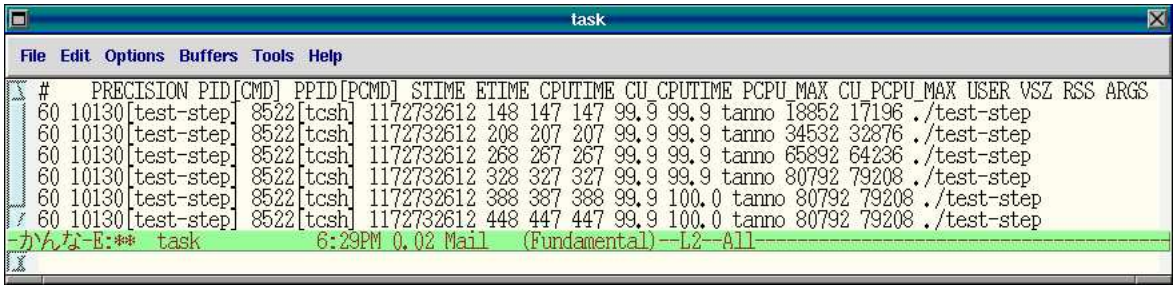

図 2.1: プロセス情報の取得例

取得間隔が短ければ短いほど詳細に状態の変化を記録できるが, 情報の取得回数が増加し, マシンに掛かる負荷が大きくなってしまう. よって今回は取得間隔を 1 分と設定した.

2.2 CPU 実行時間予測手法の概要

本提案手法で行なわれる CPU 実行時間予測の流れを図 2.2 に示す.

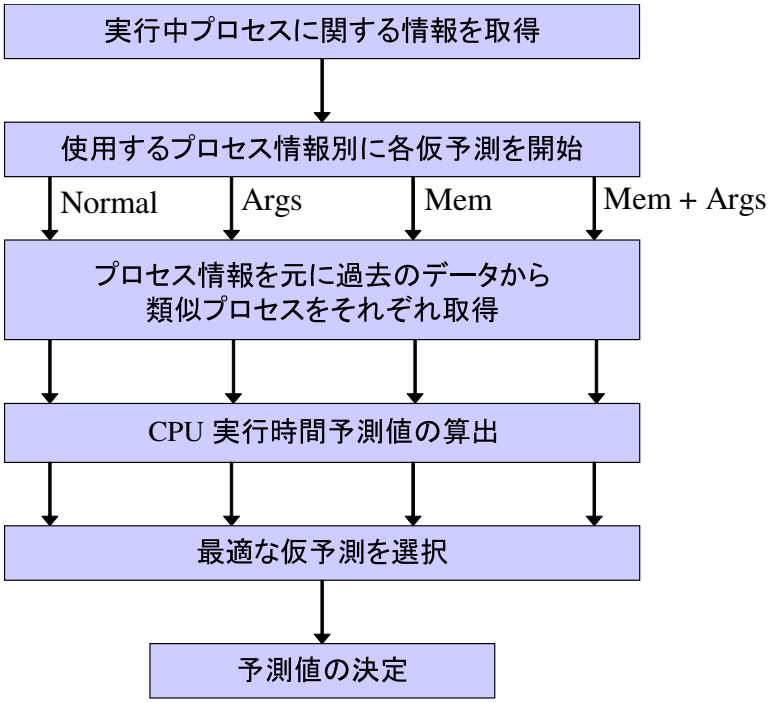

図 2.2: CPU 実行時間予測の流れ

まず, 実行中プロセスのプロセス情報を基にして仮予測を行なう. 実行中プロセスと類似するプロセスを過去の実行データから検索し, 取得した類似プロセス集合から仮予測値を算出する. この

仮予測は, 類似プロセス検索時に使用するプロセス情報の種類により, 複数実行される.

各仮予測値を算出した後, その中から最も適していると思われる値を選択し, CPU 実行時間予測結果とする. ある仮予測では類似プロセスを適切に取得できなかったが別の仮予測では適切に取得できた場合, 後者の仮予測のほうがより良い予測値を算出すると推測される. よって, 複数の仮予測を行ないその中から選択することで, CPU 実行時間予測の精度を向上できると考えられる.

2.3 類似プロセスの取得

予測時点から過去一定期間内に実行されたプロセスを検索し, その中からプロセス情報が一致するものを類似プロセスとして取得する.

本提案手法では, 類似プロセスの検索時に使用するプロセス情報の組合せにより, 複数の仮予測を行なう. 今回は, Normal, Args, Mem, Mem + Args の 4 つの仮予測を行なう. それぞれ使用するプロセス情報を以下にまとめる.

- Normal
  ユーザー名, プロセス名
- Args
  ユーザ名, プロセス名, 引数
- Mem
  ユーザ名, プロセス名, メモリ情報
- Mem + Args
  ユーザ名, プロセス名, 引数, メモリ情報

本提案手法で用いるメモリ情報は, 実メモリ使用量とする.

なお, プロセス情報の条件は満たすものの, そのCPU 実行時間が実行中プロセスの経過CPU 実行時間以下となっているものについては, 当然の事ながら取得しない.

類似プロセスの検索は, 全ての仮予測において取得類似プロセス数が最大値に達するか, 一定期間検索し終えるまで続けられる. この時, 予測時点に近い順に検索を行なう. つまり, 実行時刻が新しいものから類似プロセスを取得していくことになる. これは, 実行時刻が古いプロセスよりも新しいプロセスのほうが, より実行中プロセスに類似していると考えられるためである.

2.4 仮予測値の算出

各グループにおける類似プロセス取得条件の詳細と, 仮予測値の算出について述べる.

2.4.1 Normal, Args 時の仮予測値算出

過去のデータを検索し, 全てのプロセス情報が一致した場合に類似プロセスとして取得する.

検索を終えた時, 取得した類似プロセス群は実行時刻順に並べられている. ここでそれらの CPU 実行時間に着目し, クラス間分散が最大となるところで 2 分割をする (図 2.3). クラス間分散は, 各集団の平均値の分散である.

分割された 2 つの集合のうち実行時刻が新しい側を選択し, その CPU 実行時間平均値を仮予測値とする.(式 (2.1))

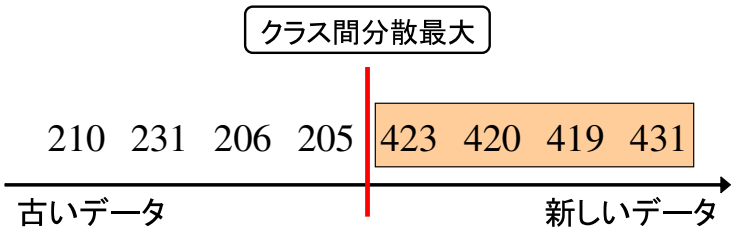

図 2.3: クラス間分散による分割例

$$P.ctime = \frac{1}{|C|}\sum_{i \in C} ctime(i) \tag{2.1}$$

C は選択した類似プロセス集合, $|C|$ は選択した類似プロセス数, $ctime(i)$ は類似プロセス i の CPU 実行時間を表す. クラス間分散による分割作業を行なうことで, 初期の実行時にはデバッグし, 後にプログラムを本動作させるといったような, CPU 実行時間が大きく変化する状況に対応することが可能となる.

次に, 選択した集合の CPU 実行時間の分散を確認する. この時, 分散が仮予測値よりも大きいならば仮予測を破棄する. 分散が大きいということは値がまとまっていないということであり, つまりは, 今回の仮予測で使用したプロセス情報では, 類似するプロセスの集合を適切に取得できなかったと見なせる. よって, 分散が平均値以上の場合は仮予測を破棄し, 使用プロセス情報が異なる他の仮予測に処理を任せることとする.

2.4.2 Mem, Mem + Args 時の仮予測値算出

過去のデータを検索し, メモリ情報以外のプロセス情報が全て一致するプロセスを候補として検出する.

候補プロセスが見つかったならば, 窓を設定し, その内に含まれる実行中プロセスと候補プロセスの実メモリ使用量変動を比較して累積誤差を計算する. 窓の範囲は, 実行中プロセスの記録された経過状態中, 予測点 T から過去数ステップの状態とする. (図 2.4)

窓の範囲 D は, 記録された過去状態数 S により式 (2.2) のように設定される.

$$D = \begin{cases} 5 & (S \geq 5) \\ S & (\text{上記以外}) \end{cases} \tag{2.2}$$

窓の範囲は予測点付近の最大 5 ステップである. 過去の実メモリ使用量は殆ど誤差がないが, 予測点付近では違うというようなプロセスは, 類似したプロセスではないと考えられる. そのような

プロセスを除外するため, 実メモリ使用量累積誤差の算出範囲を予測点付近としている.

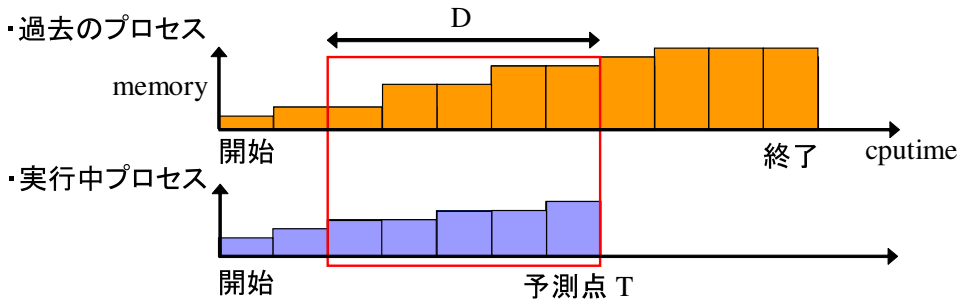

図 2.4: 実メモリ使用量の累積誤差算出範囲

窓の設定後, その範囲内の過去状態を用いて, 式 (2.3) のように実メモリ使用量の累積誤差を計算する.

$$CumError = \sum_{k=0}^{D-1} |M(T-k) - M'_{close}| \tag{2.3}$$

$M(T-k)$ は実行中プロセスの予測点 T から過去 k 状態目の実メモリ使用量である. M'_{close} は候補プロセスの実メモリ使用量だが, 経過 CPU 実行時間が実行中プロセスの $T-k$ 時点に最も近い時の値を用いる.

この累積誤差が実行中プロセスの窓内実メモリ使用量合計値の 10% 以内ならば, 類似プロセスとして取得する.

なお, 誤差を算出する際, 候補プロセスと実行中プロセスのサンプリング時刻が一致しないことがある. ps コマンドによる情報取得は実時間 1 分毎に行なわれており, 記録されている経過 CPU 実行時間の間隔は一定ではないため, このようなことが起こり得る. 誤差を算出する際の詳細は以下のようになる.

1. 実行中プロセスの記録状態中, 窓内のある状態に着目
2. その時点の経過 CPU 実行時間を基準時間に設定
3. 候補プロセスの記録状態中, 経過 CPU 実行時間が基準時間に最も近い状態に着目
4. 両状態の実メモリ使用量から誤差を算出

この流れに沿って決定された, 累積誤差算出に使用される過去状態の選択例を図 2.5 に示す.

検索を終えたならば, 取得した類似プロセス群について累積誤差が小さい順に並びかえを行なう. その後, CPU 実行時間のクラス間分散が最大となるところで, 集合を 2 分割する. 分割後, 累積誤差が小さい側の集合を選択し, その CPU 実行時間平均値を仮予測値とする. (式 (2.1))

Normal, Args 時と同様に, CPU 実行時間の分散による仮予測破棄判定も行ない, 分散が仮予測値以下ならば, 選択した類似プロセス集合と仮予測値を保持する.

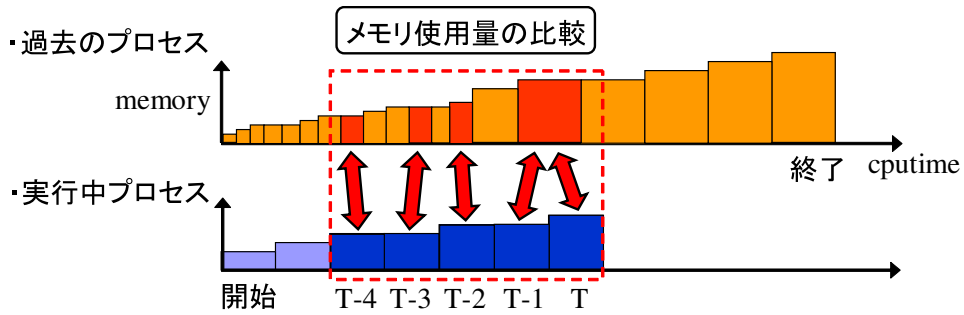

図 2.5: 累積誤差算出に使用される過去状態の選択例

2.5　信頼度による予測手法の選択

4 つの仮予測を行ないそれぞれ仮予測値を算出した後, その中から最も適していると思われるものを 1 つ選択し, 実行中プロセスの CPU 実行時間予測結果とする. ここで各仮予測に対してそれぞれ信頼度を求め, 値が最大のものを採用する. 信頼度の算出には, t 分布を用いる.

2.5.1　t 分布

t 分布は確率分布の一種であり, 標本集団の値から母集団の平均値がどの程度の値になるか推定するといったような, 統計学における検定の問題に用いられる.

t 分布の確率密度関数は式 (2.4) で表される.

$$f(t) = \frac{1}{\sqrt{k}B(k/2, 1/2)} \left(1 + \frac{t^2}{k}\right)^{-\frac{k+1}{2}} \tag{2.4}$$

k は自由度と呼ばれ, この値が高いほど確率密度関数は正規分布時の関数に近づく. $B\left(\lambda_1\,\lambda_2\right)$ はベータ関数と呼ばれる値であり, $B\left(\lambda_1, \lambda_2\right) = \frac{\Gamma(\lambda_1)\cdot\Gamma(\lambda_2)}{\Gamma(\lambda_1+\lambda_2)}$ のようにガンマ関数で表される.

確率密度関数 $f(t)$ のグラフの概形は図 2.6 のようになる.

t 分布を用いる時, 集団は正規分布に従うと仮定される. よって, 取得した類似プロセスの集合が正規分布に従うか考える必要がある.

CPU 実行時間予測法は, 実行中プロセスの CPU 実行時間は過去に実行された類似プロセスの値にほぼ等しいという考えに基づく手法である. 従って, 類似プロセス集合を適切に取得できたならば, その CPU 実行時間はある程度のまとまりを持っていると言える. つまり, その分布は集合の平均値を中心とした正規分布のような形状になっていると推測される.

以上により, 仮予測で得た類似プロセス集合と仮予測値に対して t 分布を用いることができると考えられる.

2.5.2　信頼度の設定

取得した類似プロセス集合と仮予測値に対して t 分布を用い, 各仮予測の信頼度を設定する.

実測値が予測値の前後 10%以内に収まると仮定し, その信頼度を次式により計算する.

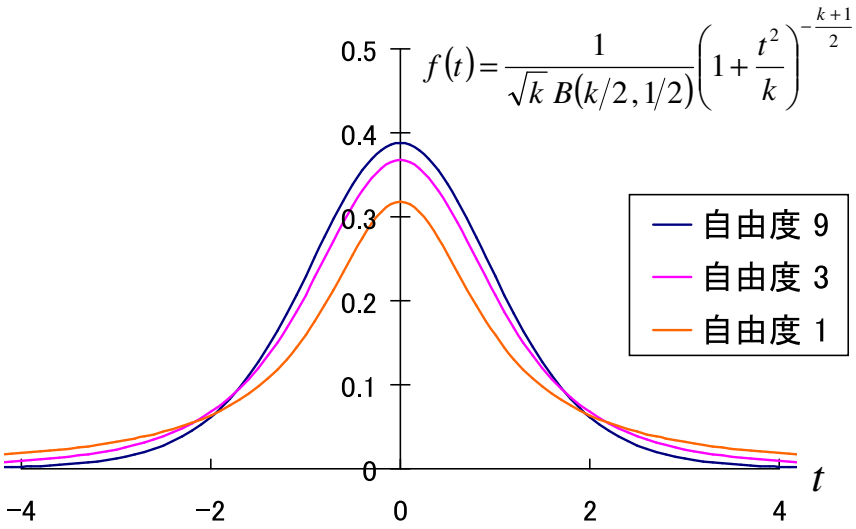

図 2.6: t 分布の概形

$$Confidence = 2\int_0^A f(t)\cdot dt \tag{2.5}$$

$$A = \frac{P\cdot 0.1}{\sqrt{S/n}} \tag{2.6}$$

P は前段階で算出した仮予測値, S は取得した類似プロセス群における CPU 実行時間の不偏分散, n は取得した類似プロセス数である. また, このときの自由度は $k = n - 1$ とする. 式 (2.5) から分かるように, 信頼度は図 2.6 の面積部分に当たる. 取得した類似プロセス数が多いほど, またその CPU 実行時間がまとまっているほど信頼度は高くなる.

統計学の検定の問題に置き換えて考えると, 取得した類似プロセス集合を標本集団, 仮予測値を標本平均, 求めたい実測値を母集団の平均と見なし, 標本平均と母平均の差が標本平均の 10%以内に収まる確率を算出することになる.

この信頼度を各仮予測に対して計算する. そして, 信頼度が最大となる仮予測を採用し, その仮予測値を最終的な CPU 実行時間予測結果とする.

第3章

実実行時間補正

前章の手法によりCPU実行時間の予測値が求まるが, その値をそのまま実実行時間予測値とすることは適さない場合が多い. 本章では, 予測点以後, 終了するまでに経過する実実行時間を予測する手法について述べる.

CPUを最大限使用する計算タスクが複数実行されているならば, その分CPUの処理が割り振られ, 終了までにかかる実実行時間が長くなる. また, 非計算タスクについては, CPUを最大限に使用しないため, CPU実行時間よりも実実行時間のほうが値が大きい.

よって, 予測対象のプロセスが計算タスクかどうかを判断し, それに応じた予測値の補正を行なう必要がある.

3.1 計算タスク数の確認

計算タスクはCPUを最大限使用するため, CPU利用率は高い値となっている. よって, 実行中プロセスが計算タスクかどうかの判断にCPU利用率を用いることができる.

CPU利用率は, 過去1分間の値とプロセス開始から予測時点までの値を比較し, 値が高いほうを選択する. これは, psコマンドによる記録条件に満たないような, 実行時間の短いプロセスの影響をできる限り排除するためである.

全ての実行中プロセスについてCPU利用率を設定したあと, 計算タスクかどうかを判定する. CPU利用率が閾値を越えた場合に計算タスクと判断する. 判定式を以下に示す.

$$Pro_i\_task = \begin{cases} 1 & (\text{CPU 利用率} \geq PCPU_{TH}) \\ 0 & (\text{上記以外}) \end{cases} \tag{3.1}$$

$$PCPU_{TH} = \frac{\text{搭載CPU 数} \cdot 80}{\text{実行中プロセス数}} \tag{3.2}$$

つまり, 実行中プロセスの CPU 利用率が, その時点で取ることのできる CPU 利用率最大値の 80% 以上ならば, そのプロセスは計算タスクであると判断される.

3.2　計算タスク数に応じた実実行時間補正

計算タスク数が搭載 CPU 数以上ならば, CPU の処理が割り振られ, 実実行時間が大きく延びると考えられる. よって, 実行中の計算タスク数を確認し, その数に応じた実実行時間補正を行なう.

- **計算タスク数 > 搭載 CPU 数の場合**

計算タスクに対する補正処理

計算タスクと判断された実行中プロセスに対して延長補正を行なう. 予測された CPU 実行時間中, 計算タスク数が a である期間 $I(a)$ を式 (3.3) のように補正する.

$$I'(a) = \begin{cases} I(a) \cdot \dfrac{a}{\text{搭載 CPU 数}} & (a > \text{搭載 CPU 数}) \\ I(a) & (\text{上記以外}) \end{cases} \tag{3.3}$$

$$a = \sum_i Pro_i\_task \tag{3.4}$$

補正区間例を図 3.1 に示す.

- CPU 数 = 2 の場合

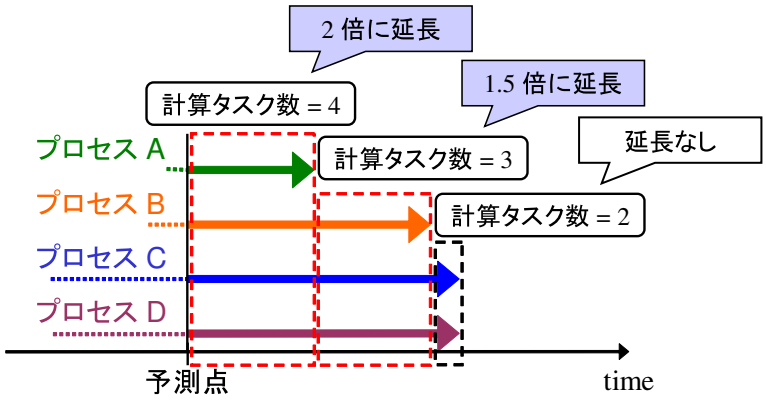

図 3.1: 搭載 CPU 数 = 2 の場合の延長区間

2 台の CPU を搭載している場合, 計算タスク数が 4 となる区間を 2 倍に延長し, 3 となる区間を 1.5 倍に延長する.

補正後の値をまとめ, 実実行時間予測結果とする. (式 (3.5))

$$P.rtime = \sum_a I'(a) \tag{3.5}$$

計算タスクではない実行中プロセスに対する補正処理

計算タスクと判断されなかった実行中プロセスは, 他のプロセスの影響を受けることなく実行されていると考えられる. これについては, CPU 実行時間予測値に前段階で設定した CPU 利用率を除算することで延長補正を行なう.

$$P.rtime = \frac{P.ctime}{\text{CPU 利用率}} \tag{3.6}$$

つまり, 単純に CPU 利用率と CPU 実行時間予測値から実実行時間予測値を割り出している.

- **計算タスク数 $\leq$ 搭載 CPU 数の場合**

計算タスク数が搭載 CPU 数以下の場合, 1 つの CPU が一つの計算タスクの処理に専念していると考えられる. (実際には処理を担当する CPU が切り替えられることもあるが, 全体的に見れば, 1 つの CPU につき 1 つ以下の計算タスクを処理していることになる.)

この場合, 前段階で設定した CPU 利用率の逆数を CPU 実行時間予測値に乗算することで, 延長補正とする. 計算式は式 (3.6) と同じである.

第4章

実験

本章では, 本研究で提案した手法による実行時間予測を行なう. また, t 分布信頼度による予測値選択の有効性を確認する.

4.1 実験環境

データ収集期間からランダムに予測時点を選択し, その時実行されていたプロセスについて予測を行なった.

本実験で使用した計算機等の実験条件を, 表 4.1 に示す.

表 4.1: 実験条件

データ収集対象計算機	CPU : Xeon 2.4 GHz × 2, メモリ : 1 GB
データ収集期間	2007 年 2 月
予測対象から除外するプロセス	対話的シェル CPU 実行時間 < 60 秒 CPU 利用率 $< 10\%$
評価対象から除外するプロセス	類似プロセス数 $= 0$ Normal 予測のみ有効かつ類似プロセス数 $= 1$
類似プロセス検索期間	予測時点からデータ収集開始時点まで
取得類似プロセス最大数	10

本手法では, CPU 処理の割合が低いプロセスは対象としていないため, それらについては予測を行なわない.

類似プロセスを全く見つけることができなかった場合, 本提案手法では予測ができないため, そのようなプロセスは評価対象から除外する. また, Normal 予測のみ有効で取得類似プロセス数が 1 つであるプロセスも評価対象から除外する. Normal 予測では類似プロセスは取得できたが他の予測では取得できないということは, 予測対象プロセスと類似プロセスでは, 引数やメモリ情報などのプロセス情報が異なっているということである. つまり, 取得したプロセスは予測対象プロセスと状態があまり類似しておらず, そのプロセス 1 つのみを用いて予測をすることは, 本提案手法には適さない. よって, そのようなプロセスも評価対象から除外する.

今回, 予測実験を行なうためのデータベースの作成が遅れ, 予測対象とするプロセスの情報をあまり取得できなかった. そのため, 実験用に作成したプロセスを計算機に投入し, 予測実験の大部分はそれらについての予測となっている. 以下に作成したプロセスの詳細を示す.

task-360
: CPU 実行時間が約 6 分かかり, 引数として整数を渡すと CPU 実行時間が入力整数倍になる

task-600
: CPU 実行時間が約 10 分かかり, 実メモリ使用量が多い場合は CPU 実行時間が 2 倍になる

task-m-incre3
: 実メモリ使用量が増加し, 処理の半分を終えたところで実メモリ使用量がクリアされ, また徐々に増加する

task-m-step
: 実メモリ使用量が task-m-incre3 と同様の変化をするが, その変動幅が大きい

投入するプロセスとその引数はランダムに決定した.

予測精度を評価するため, 式 (4.1) に示される評価関数を用いる.

$$Error = \frac{|R_t - P_t|}{R_t} \cdot 100 \tag{4.1}$$

R_t は実測値, P_t は予測値を意味する. 式 (4.1) は予測値に対する実測値との誤差率を表しており, 値が 20%以内に収まっていれば, 高精度に予測ができたと判断できる.

以上の条件に従い, ランダムに予測点を定め 50 回予測実験を行なった. 予測点で動作していた予測対象プロセス数は合計 93 個であり, そのうち予測値が求まったプロセス 61 個について評価を行なった. 残りの 32 個のプロセスは, 全て過去に類似プロセスが 1 つも見つからず予測が行なえなかった.

各仮予測単体での予測結果と, それらを信頼度選択により統合した予測結果について比較を行なう.

4.2　CPU 実行時間予測実験

提案した手法を用いて CPU 実行時間を予測した結果を図 4.1 に示す.

■ 予測有効プロセス数

信頼度選択 : 61　　Mem + Args : 52　　Mem : 55　　Args : 57　　Normal : 60

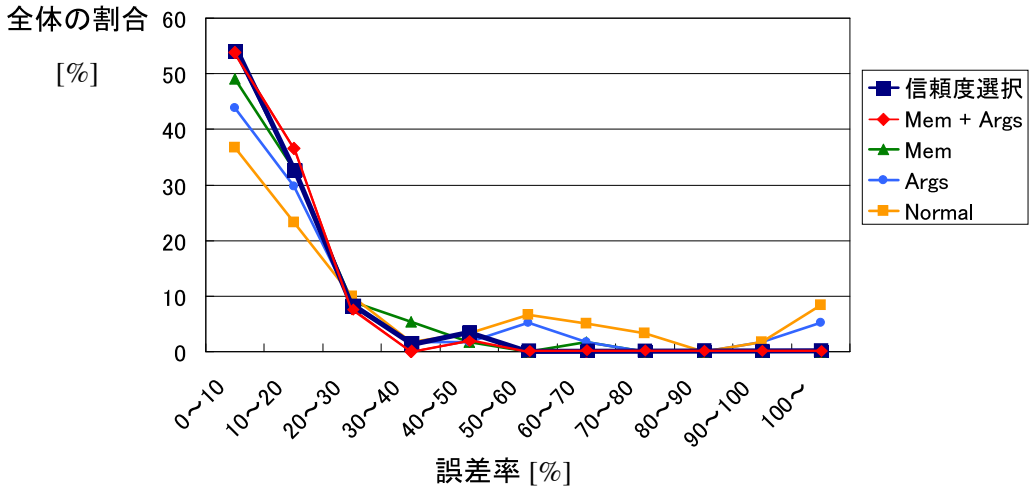

図 4.1: CPU 実行時間予測結果

予測が有効であったプロセス数は, 信頼度による選択を行なった場合は 61, Mem+Args 予測のみの場合は 52, 続いて Mem が 55, Args が 57, Normal が 60 となった. 予測精度に関しては, 誤差率が 20%以内となった割合は Mem+Args 時が最も高く, 次いで信頼度選択, Mem, Args, Normal という順となった.

これより, 引数を考慮することで適切に類似プロセスの取得ができ, またそれ以上にメモリ使用量が有用となることが分かった.

信頼度選択時については, Mem+Args 時よりも誤差率 20%以内となる割合が低下するが, 図 4.1 から分かるようにほとんど差は表れなかった. また, 予測が有効なプロセス数は最も多くなった. これより, t 分布を用いた信頼度選択を行なうことで, 複数の予測結果から適切なものをある程度選択できていると考えられる.

次節で信頼度選択の有効性について詳しく述べる.

4.3　信頼度選択の有効性

t 分布信頼度による予測選択の有効性について確認するため, CPU 実行時間予測結果のうち信頼度が 0.9 以上のものを図 4.2 に示す.

信頼度 0.9 以上のものはほとんどが誤差率 20%以内に収まるという結果になった.

Normal と Args に関しては誤差率が 100%を越えるものが見られた. 今回設定した信頼度は, 取得した類似プロセスの値がまとまっているほど高くなるという値である. つまり, 類似プロセスと予測対象プロセスで CPU 実行時間の傾向が違うならば, 信頼度が高い場合でも予測値は実測値と全く違う値になり得る, ということを示している. しかし, 信頼度選択時の結果には誤差率が 100%以上となるプロセスは存在していない. これは, メモリ情報を考慮した場合に, 考慮しない場合より

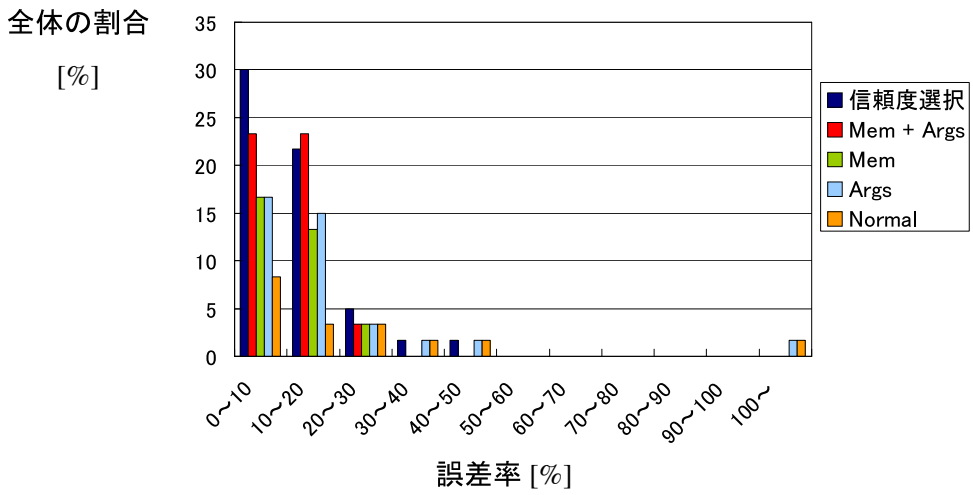

図 4.2: 信頼度 0.9 以上の CPU 実行時間予測結果

も正確な予測が行なわれ, 信頼度選択によりそれらが選択されたということを意味する. よって, 複数の予測を算出し, t 分布を用いた信頼度による選択を行なうことで, そのような状況にもある程度の対応が可能であると言える.

4.4　実実行時間予測実験

CPU 実行時間予測結果をもとに, 予測点以後の実実行時間を予測した結果を図 4.3 に示す.

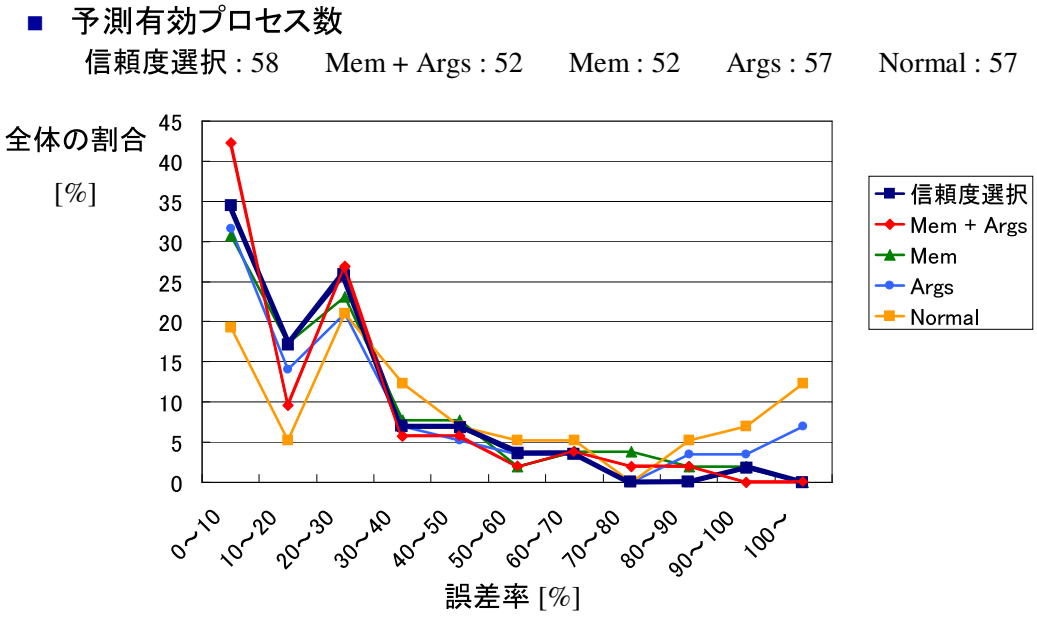

図 4.3: 実実行時間予測結果

予測が有効となったプロセス数は, 信頼度選択時が 58, Mem+Args, Mem が 52, Args, Normal

が 55 となった. また予測精度に関しては, Mem+Args 時に誤差率が 20%以内の割合が最も高くなり, 僅差で信頼度選択, 次いで Mem, Args, Normal という結果となった. よって, この場合でもメモリ情報と信頼度選択が予測に有用であるということが確認できた.

4.5　まとめ

本研究で提案した手法を用いて実行時間予測実験を行ない, 本手法の予測精度について評価を行なった.

実行時間予測実験では, プロセス情報としてメモリ情報を用いることで予測精度が向上することを確認した. また, t 分布信頼度による予測選択の有効性も確認した.

また, 信頼度が 0.9 以上の予測結果のみ抽出することで, t 分布信頼度の有効性について再確認を行なった.

第5 章

結論

5.1 本論文の成果

本論文では, プロセス情報を使用した予測を複数実行し, その中から適切な予測を信頼度により選択する手法を提案した. そして, 実験により本手法の有効性を示した.

第2 章では, プロセス情報を考慮したCPU 実行時間予測法を提案した. 予測を行なうために, 予測対象のプロセスに類似したプロセスを過去の実行データから探し出すが, この時使用するプロセス情報として, ユーザ名, プロセス名, 引数, メモリ情報を用いた. ユーザ名, プロセス名, 引数を使用する場合は単純に一致するもの, メモリ情報を使用する場合はその変動を考慮し類似プロセスを取得する. 取得した類似プロセス集合から, より予測対象プロセスに状態が近いと思われるものを抽出し, それらのCPU 実行時間平均値を仮予測値とする. 使用するプロセス情報の組合せにより, Normal, Args, Mem, Mem+Args の4 つでそれぞれ仮予測を行なう. 複数の仮予測結果に対してt 分布による信頼度を設定し, 信頼度が最大の仮予測値を最終的な予測結果として採用する.

第3 章では, 予測対象プロセスのCPU 利用率とCPU 実行時間予測値を基にした実実行時間補正処理について述べた. CPU 利用率から予測対象プロセスが計算タスクかどうか判断し, 計算タスク数に応じてCPU 実行時間予測値延長補正を行なう.

第4 章では, 提案した予測手法を用いて, CPU 実行時間及び実実行時間予測実験を行なった. 予測実験結果より, プロセス情報としてメモリ情報を用いると予測精度が向上することを確認した. また, t 分布による信頼度を用いた予測選択が有効であることを示した.

5.2 今後の課題

本提案手法のCPU 実行時間予測において, 仮予測時に用いるメモリ情報は実メモリ使用量である. 他のメモリ情報として仮想メモリ使用量が存在するが, 本提案手法では使用していない. これ

は, 予備実験において, 仮想メモリではなく実メモリの使用量を用いた場合のほうが予測精度が良かったためである. しかし, 実メモリ使用量という値は, その時の計算機の状態により, 全く同じプロセスを動作させても値が変化する可能性がある. よって, 使用するメモリ情報を仮想メモリ使用量にするか, もしくは両方使用し信頼度による選択を行なう, といったような変更を行なう予定である.

実実行時間補正処理の際, 予測対象のプロセス中, CPU 実行時間の予測を行なえなかったプロセスが 1 つでも含まれているならば, 本論分ではそれらのプロセスを実実行時間予測の評価対象から外している. しかし, そのような場合であっても予測結果を出力しなければならない状況が考えられる. 従って, CPU 実行時間予測ができなかった場合の対処法を検討する必要がある.

本論文では, 提案手法の予測速度について特に考慮をしていないため, それについても検討が必要である. 現時点では類似プロセスを検索するデータ期間が 1ヵ月だが, 検索期間が長くなれば, それに応じて提案手法の実装法による予測速度への影響が大きくなる.

本提案手法では, 類似プロセス取得最大数や計算タスクと判断するための CPU 利用率閾値などといったパラメータが数多く存在する. これらは予備実験により設定されているが, 本当にその値が適切なのか, 根拠を詰める必要がある. もしくは, 動的にパラメータを設定できるならば, そのように改良すべきである.

今回, 実験に使用したデータ期間が非常に短いものとなった. そのため, 予測対象とするプロセスの情報をあまり取得できず, 予測実験の大部分が実験用に作成したプロセスに対するものとなった. よって, 実際に計算機に投入されるプロセスについて予測を行ない, 提案した予測手法が有効かどうかを判断する必要がある.

参考文献

[1] 立見博史 : “並列分散処理のための計算機負荷予測に関する研究”, 修士学位論文, Mar. 2006.

[2] Warren Smith, Ian Foster, Valerie Taylor : “Predicting application run times with historical information”, J. Parallel Distrib. Comput, Vol.64, pp. 1007-1016, 2004.

[3] 鈴木義也 他 : “概説 数理統計”, 共立出版株式会社, Dec. 1994.

[4] Wikipedia : *http://ja.wikipedia.org/*

謝辞

本研究を進めるにあたり，指導教官として多大なご指導を賜りました東北大学大学院工学研究科 阿曽 弘具 教授に心より御礼申し上げます．

本研究を進めるにあたり，ご指導，ご意見を賜りました東北大学大学院工学研究科 大町 真一郎 助教授に心より感謝致します．

本研究を進めるにあたり，終始貴重なご指導，ご意見を賜りました東北大学大学院工学研究科 菅谷 至寛 助手に心より感謝致します．

また，本研究を支えてくださいました阿曽研究室の皆様, 両親, 家族に心より感謝致します．