システムソフトウェアと オペレーティング・システム 74－29
マルチメディア通信と分散処理 81－29
（1997. 2. 28）

課金を行なわないワークステーションクラスタにおけるジョブ管理

角田 剛　　碓崎 賢一

九州工業大学 情報工学部

部門コンピューティングの様に，課金が行なわれない小規模な計算機システムにおいては，経済的な自己規制が働かないため，特定の利用者による無秩序なジョブの投入が起こりやすい．このようなジョブ投入は他の利用者の利用権を侵害することになるが，UNIX などのクラスタシステムでは，利用者間の平等な利用権を管理する機構がないという問題がある．本稿では，負荷分散などの計算機中心の視点のみに基づくスケジューリングの問題点を述べ，それを解決する機構として，利用者のシステムの利用状態を調査，蓄積，参照しジョブの優先順位を決定することによって全ての利用者に平等に計算機資源を分配するシステムを提案している．

A Job Management Method for Workstation Cluster without Accounting

Tsuyoshi SUMIDA and Ken'ichi KAKIZAKI

Department of Computer Science and Electronics,
Faculty of Computer Science and Systems Engineering
Kyushu Institute of Technology, Iizuka, 820 Japan

This paper proposes a job management method to share system resources fairly between all users in workstation cluster. Workstation cluster which is operated without accounting has a problem: paticular users will monopolize all system resources since there is no mechanism to protect moderate users' jobs. First, we describe a problem of job scheduling system which focuses on only load sharing or load balancing. Secondly, we propose a system which allows all users to share system resources fairly by prioritizing jobs based on users' system resource usage.

1 はじめに

近年の，ワークステーション (WS) の低価格化とネットワーク技術の進歩により，大型汎用計算機やミニコンピュータを頂点に据えた集中処理方式から，WS を LAN に接続して利用する水平分散処理へと移行している．複数の WS を接続し，クラスタ構成とした分散処理システムの運用で最も重視されているのは，以下のような事柄である．

- 各ホストへの均等な負荷分散
- 各ホストの過負荷状態の防止
- 時間的な負荷の分散

これらの事柄は，クラスタ中の計算機をいかに効率的に利用して処理を行なうかという計算機中心の観点に立ったものである．このような観点に立った研究 [佐藤 94][山井 94] は数多くなされており，実装，製品化されているジョブ管理システム [Pop85][ダイ 95] もある．

クラスタ構成の分散処理システムは，研究，開発の一部門内で利用される部門コンピューティングに利用されることが多い．部門コンピューティングシステムは，無課金で運用されることが多いが，その場合には，経済的な観点に立った自己規制が行なわれないために，ある特定の利用者による無秩序なジョブ投入と，それによるジョブリクエスト・キューの占有が行なわれがちである．従来のジョブ管理システムは，前述の様に負荷分散などの計算機中心の運用管理のみであるため，節度を持って利用している利用者の利用権が制限を受けてしまうという問題がある．

本稿では，上記のような問題を解消するため，利用者中心の観点に立ち，利用者が各自の利用権を平等に行使できることに主眼を置いた，分散ジョブ管理システムの実現方式について提案する．提案方式では，各利用者の利用状況を蓄積し，その情報を基にジョブのスケジューリングを行なうことにより，各利用者の利用権を行使できるようにしている．

2 システムの概要

本システムは，以下のような計算機システムを想定している．

- 複数のワークステーションをネットワークで結合したクラスタ構成システム
- 1 台のワークステーションを管理用のフロントエンド，その他のワークステーションを処理用のバックエンドとする構成のシステム
- 共通ファイルシステム等を用いてクラスタを 1 台のワークステーションのように扱えるシステム

本システムが対象とするジョブは，ユーザとのリアルタイムでのインタラクションを持たないバッチプログラムである．また，それぞれのジョブは独立しており，互いの実行順序は問わないものとする．ジョブはジョブ投入用のプログラムによりジョブリクエスト・キューに投入され，実行されるのを待っている．ジョブリクエスト・キューから選択されたジョブは適当なバックエンド上でプロセスとして実行される．

3 負荷分散主体のジョブ管理の問題点

課金が行なわれるシステムでは，実行したプロセスの数や実行時間などに基づいて利用料金が課せられる．このような環境では，ジョブの投入量に比例して支出が増大することになる．これにより，利用者は過度の経済的負担を避けるため，処理に長時間要したり，大量のメモリ領域を占有したりするような高負荷なジョブ投入を自発的に控えるようになる．

しかしながら，課金がないか一定額の利用料金しか課せられないような環境では，ジョブをどのように投入しようと利用者の負担額は変わらない．この場合，支出の増大という負担が課せられないため自己規制が働きにくく，ジョブの大量投入が起こりやすい．このような状況下では，ある特定の利用者によるジョブリクエスト・キューの占有が起こりやすいという問題がある．

この結果，節度を持って利用している利用者によって投入されたジョブリクエストの，キュー内部で占める割合が大幅に低下する．ジョブはその投入者に関係なく，リクエストキューの頭から取り出されて実行されるため，ジョブの投入数が少ない利用者のジョブの実行頻度が低下することとなる．このため，ジョブの大量投入者と同じ利用権を持っていても，利用できる計算機時間の割合が低下して権利を行使できない，あるいは実質的な計算機の処理速度が低下するという問題が生じる．

このような問題を回避するために，各利用者に対してシステム資源の割り当てを制限する方式が

ある．例えばLSF[ダイ95]では，各利用者に対して同時に実行できるジョブ数や使用できるホスト数を制限することができる．しかしながら，同時実行ジョブ数を制限した場合，システムの混雑時には効果を発揮するが，ある特定の利用者の全てのジョブを十分実行できる閑散期でも，制限されているジョブ数より多くのジョブを実行に移すことができず，計算機に遊びが生じシステムの稼働率を下げるという問題がある．

4 ジョブスケジューリングへの要求

課金のないシステムを円滑に運用するためには，以下のような項目の実現可能なジョブスケジューリング方式が必要である．

1. ジョブの投入数や頻度に影響されず，利用者が同等の権利を得られる
2. 無駄なジョブの投入を防ぐために，各利用者が自己規制を働かせる
3. 計算機の稼働率の低下を最小限に抑えるために，閑散期における利用者の積極的な利用を誘導する
4. 緊急度の高いジョブは優先して処理を行なう

1の項目は，計算機の利用状況に基づき，利用者間での計算機資源の利用の均衡化を図るジョブスケジューリング機構を導入することにより実現できる．2の項目は，1を実現することにより，ジョブの投入数による無秩序な利用権の獲得ができなくなるため，課金による経済的負担に準じた抑制が得られる．3,4の項目は，1を実現するために行なう利用者毎の計算機資源の利用量の積算に重み付けを導入することにより実現できる．

4.1 計算機資源の平等配分

4.1.1 利用者に対する配分

ある利用者が他の利用者に比べ，過去に利用した計算機時間が短い場合には，その利用者は他の利用者に比べ計算機資源を利用していないと見ることができる．また，システムに対する利用料金を多く支払っている利用者は，他の利用者に比べて計算機資源をより多く利用する権利があるといえる．

リクエストキューからジョブを取り出す際に，このような利用者の投入したジョブを優先的に選択することにより，利用者間の計算機資源配分の平等性を保持することができる．

4.1.2 グループに対する配分

計算機の利用権が，個人ではなく研究室や課などのグループ単位で設定されている場合がある．そこで，利用者個人だけではなく利用者が属するグループ間でも，資源の分配が平等になるように管理する必要もある．このような場合には，計算機資源の分配は図1に示すように，グループ単位の配分を行なった後に，個人単位の配分を行なう．

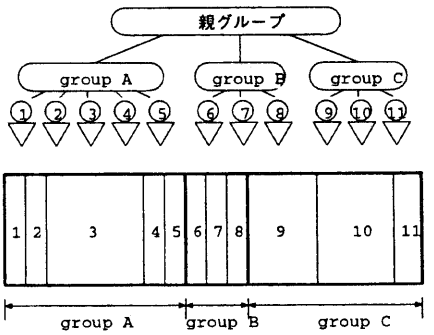

図 1: グループに対する資源の配分

4.2 閑散期のジョブ投入

計算機に過度のジョブリクエストが投入されることは，効果的な運用を行なう上で望ましいことではない．また，計算機はジョブの実行の有無に関わらず，一定の運用コストを必要とする．このため，計算機の稼働率が低下することは，経済的に望ましいことではない．計算機の経済性を向上させるためには，夜間や休日などの計算機の閑散期にジョブが積極的に実行されるような利用者への動機づけが必要である．

そこで，システムの負荷が低い期間にジョブの実行を依頼した場合には，計算機時間の利用量を積算する際に，一定の割合で減じて積算する方式をとる．これにより，利用権の減少を低く抑えることができるため，利用者が積極的に利用することになる．この結果，閑散期の計算機時間も，緊急度は低いが処理が必要なジョブにより利用されることになるとともに，混雑時の度合を低減させることができる．

4.3 緊急時のジョブ投入

納期や〆切が間近になり緊急にジョブを実行させたいような場合でも，自分より優先度の高い利用者のジョブが投入されていると，ジョブが実行され結果が得られるまで時間がかかることになる．緊急時には，通常の利用権に基づく優先度を越えた高い優先度でジョブが実行され，実行結果がより早く得られることが望まれる．

そこで，必要な時に優先順位を高めに設定してジョブを投入できる機能も必要となる．しかしながら，この機能を利用すると，一時的に不平等な計算機資源の配分が行なわれることになる．したがって，平等性を維持するため，この機能を利用すると後のジョブの選択順位の決定時に精算を行なうような処置も必要である．

5 実現方式

5.1 システム利用状況の蓄積

我々の提案するシステムでは，次に実行するジョブをリクエストキューから取り出す際に，過去のシステムの利用状況を参照し，利用権の充足率が利用者間で均等になるようにジョブ管理を行なう．過去の利用状況を参照するためには，各利用者とグループの利用状況の情報を蓄積したデータベースが必要となる．

データベースには，利用者のシステム利用状況のデータとして次のような情報を蓄積する．

- ユーザ名
- グループ名
- ジョブの名前
- ジョブの実行時間（CPU 時間）
- 使用した記憶領域

これらの情報は，ジョブの実行が終了した時に Exit Status とともに返される情報により得ることができる．データベースに格納するのはこれらの確定した情報であり，各バックエンドのロードアベレージなどのリアルタイムに変化する情報は，各バックエンドにシステムが逐次アクセスすることよって得ることができる．

5.2 優先度の算出

5.2.1 利用者の優先度

実行するジョブの選択の指標となる利用者毎の優先度の算出には，以下の値を用いる．

- 過去の一定期間内に実行したジョブの累積 CPU 時間
- 優先権（利用料金を余計に払っているなど）

前者は，過去の利用状況を蓄積したデータベースを検索することで得ることができる．後者は利用者の情報を設定ファイル等であらかじめ設定しておく必要がある．

各利用者の優先度の算出は次の手順で行なう．

1. 現時点でリクエストキューにジョブが投入されている各利用者の累積 CPU 時間を優先権で割る．
2. 1 の値を積算する．
3. 各利用者毎に 1 の値を 2 の値で割る．
4. 各利用者毎に 3 の値の逆数をとる．

このようにして算出された優先度は 1 以上の値を持ち，優先度が高いほど大きな値を持つことになる．

5.2.2 グループを単位とした優先度

グループを単位としたジョブスケジューリングを行なっている場合には，各利用者の優先度の算出は，次に示す 3 段階の処理によって行なわれる．

最初の段階では，グループ単位の優先度の計算を次の手順で行なう．

1. 現時点でリクエストキューにジョブを投入している利用者を持つ各グループ毎に，全利用者の累積 CPU 時間を算出し，各グループの優先権で割る．
2. 1 の値を積算する．
3. 各グループ毎に 1 の値を 2 の値で割る．
4. 各グループ毎に 3 の値の逆数をとる．

次の段階では，各グループ毎に，利用者単位の優先度の計算を次の手順で行なう．

1. 現時点でリクエストキューにジョブを投入している利用者の累積 CPU 時間を算出する．
2. 1 の値を積算する．
3. 各利用者毎に 1 の値を 2 の値で割る．
4. 各利用者毎に 3 の値の逆数をとる．

最後に，グループ単位の優先度と利用者単位の優先度の積を求め，それを各利用者の優先度とする．

このように，利用者の優先度の算出に所属するグループの優先度を関係させると，利用者個人の優先度は高いにもかかわらず，グループの優先度

が低いためにジョブが選択されにくくなるといった状態になり得る．グループに属する各利用者はこのような状態を避けるため，協力して無駄なジョブ投入を抑えるようになる．

5.3 優先度に基づくジョブの選択

ジョブのリクエストキューから実行するジョブの選択は，各ジョブの投入者の優先度に基づき行なわれる．現時点でリクエストキューに投入されているジョブの利用者の各優先度をその合計値で割ったものをジョブの選択率と呼ぶ．各利用者のジョブは，その選択率の割合で実行される．

これにより，利用者1人が使用している時でも実行ジョブ数を制限されることなく全ての資源を最大限に利用して処理が行なえる．また，複数の利用者がジョブを投入していても，利用者同士の優先度のバランスによって確率的にジョブ選択が行なわれることになる．この選択は，現時点におけるジョブの投入数の多少に関わらず行なわれるため，平等にジョブの発行ができ特定の利用者が実行権を占有することもない．

すでに計算機資源の利用量が多く，優先度が低下している利用者でも，永久に待たされることがないようジョブの発行を保証する必要がある．この方式では，リクエストキュー中からのジョブの選択は選択率という確率に基づいて行なうため，優先度の低い利用者でも適当な頻度で実行されることになる．

実際のジョブの選択，実行は以下のような手順で行なう．

1. フロントエンドが各バックエンドに問い合わせて負荷情報を得る
2. 1の情報から最も負荷の低いバックエンドを選択する
3. 2のバックエンドの負荷が閾値以下ならばジョブを実行できる．できない場合は一定時間待って再び1から始める
4. ジョブを実行できる状態なら，ジョブを投入している各利用者の優先度を計算し，現時点の各利用者の選択率を求める
5. ジョブを投入している利用者のうち1人を4で求めた選択率の確率により選択
6. 選択された利用者のリクエストキューの先頭からジョブを取りだす．
7. 選択されたバックエンドで選択されたジョブを実行させる．

このようにしてジョブを選択，実行させることにより，負荷を分散しつつ各利用者間の平等性を保つことができる．

5.4 閑散期のジョブ管理

我々のシステムでは，各バックエンドにおいて1グループもしくは1利用者のジョブの実行数を1つに限定している．このため，システムが閑散期にあるか否かは，全バックエンドのロードアベレージの平均が1以下か否かで確認することができる．

システムの閑散期に実行されたジョブに関しては，あらかじめ設定されている1未満の値をジョブの実行時間に掛け合わせ，その値をデータベースに蓄積する．実際の実行時間より短い時間しか利用していないものとして累積されるため，通常時に同じジョブを投入した時に比べ，優先度の低下の幅が小さくなる．これにより，利用の促進をはかることができる．

5.5 緊急時のジョブ管理

緊急にジョブを実行させたい場合は，投入用のプログラムに,「-p(優先倍率)」というオプションをつけてジョブを投入する．例えば，solveという名前のジョブを2倍の優先度で実行させたい場合，

```
run -p2 solve
```

というようにして投入する．このようにして投入した場合，利用者の優先度は，本来の優先度に優先倍率を掛けたものとする．

優先度を上げて投入した後は，同じジョブを優先倍率回数分実行させたものとしてデータベースに記録する．こうすることにより，後にジョブを実行させようとした時，通常の使用に比べ優先度の低下が大きいため，長期的にはジョブの選択率が低くなる．したがって，高い優先倍率で大量に投入するなど，この機能を乱用しようとすると，選択率が限りなく0に近付き，最終的には最高倍率でジョブ投入しても最後になるまで選択されなくなる．このように，この機能を自由に利用することは可能だが，乱用することは構造的に不可能である．

5.6 システムの構成

本システムの構成を図2に示す．本システムでは，ネットワークで接続された複数の計算機のう

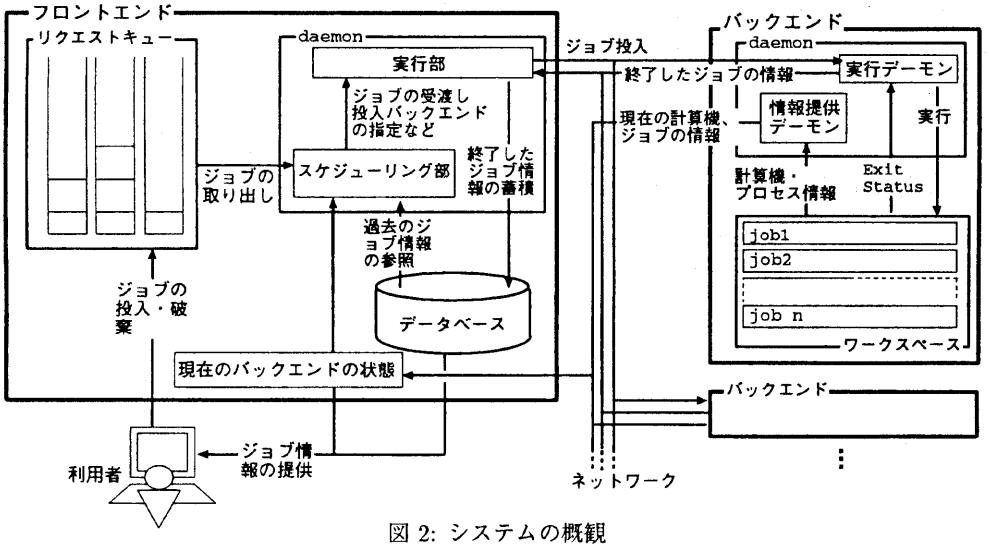

図 2: システムの概観

ちの 1 台をフロントエンドと呼ばれるジョブ管理を集中的に行なうホストとし，残りの計算機をバックエンドと呼ばれるジョブを実行，処理するホストとする構成をとる．

利用者はジョブ投入用のプログラムを用いてフロントエンドが所有，管理する実行待ち行列にジョブを投入する．システムに投入されたジョブは全てフロントエンドの管理下に置かれ，適当なバックエンドで実行される．

フロントエンドはデータベースや各バックエンドに逐次アクセスしてジョブ情報を得る．得た情報を元にしてスケジューリング部が実行するジョブや実行させるバックエンドを決定する．バックエンドはフロントエンドから渡されたジョブを実行し，実行結果や ExitStatus とともに得られるジョブの情報をフロントエンドに返す．返ってきた情報はデータベースに蓄積され，過去のジョブ情報としてジョブ選択の際参照される．

6 おわりに

本稿では，課金を行なわないワークステーションクラスタにおいて，不平等な計算機資源の割り当てにより発生する問題と，それを解決するため，利用者の過去の利用状況を蓄積，参照することにより，利用者間の不平等を排するジョブ管理方式を示した．

参考文献

[Pop85] Popec, G. J. and Walter, B. J.: “The LOCUS Distributed System Architecture”, The MIT Press (1985).

[ダイ 95] ダイキン工業株式会社電子システム事業部：“Load Sharing Facility 管理者ガイド” (1995).

[佐藤 94] 佐藤令子, 佐藤裕幸, 中島克人, 田中千代治：“疎結合型マルチプロセッサ上の拡散型動的負荷分散方式”, 情報処理学会 論文誌, Vol. 35, No. 4, pp. 571–580 (1994).

[山井 94] 山井成良, 若林進, 下條真司, 宮原秀夫：“UNIX におけるコマンド単位の負荷分散機能の設計と実装”, 電子情報通信学会 論文誌, Vol. J77-D-1, No. 7, pp. 483–492 (1994).