論文の内容の要旨

論文題目　アイドル状態計算資源融通システムに関する研究

氏名　山口 実靖

ネットワーク上には多数の計算機が存在しているが, その全てが常時使われているわけではない. 不使用のまま放置されてい計算資源が大量にあると言える. 本研究では, この大量にあるアイドル状態計算資源を有効に利用するシステムとして “BG タスクスペース” を提案する.

提案手法では, ユーザが席を立つなどで計算機が不使用状態になるとシステムからタスクが投入され, ユーザが使っていない間はそのタスクを処理し続ける. ユーザが席に戻り作業を再開する際には, そのタスクはユーザの作業の妨げとなるので速やかにその計算機から移送される (図 1 参照). また, 提案システムの概要は図 2 のようになり, 各計算機のプロセスで協調してタスク実行環境を構築する.

本研究で下記の分散環境を想定している. (1) 使用する計算資源群はヘテロジーニアスである, (2) 計算機群そのものを管理しておらず計算機にはプロセスを立ち上げるのみである, (3) システムの環境が動的である, (4) 構成する資源群は信頼性が低い. 提案システムは様々なユーザから空いている計算資源を提供されるため計算機群がヘテロジーニアスになることは避けられない. 不使用資源を提供してもらうためにユーザに特定の OS などの使用を強要するのは現実的ではないと考え各ユーザにはプロセスを 1 個立ち上げることのみ要求することとした. ユーザは頻繁に計算機の使用/不使用を切り替えるためシステム環境は動的であり同時に信頼性が低いものとなる. また, 上記の環境において特に下記の 4 項目の実現を目指している. (1)“BG タスク”(後述) が “FG タスク”(後述) の作業を可能な限り妨げない, (2) 協調システムの一部に障害が発生してもシステム全体がダウンしない高い耐障害性がある, (3) 協

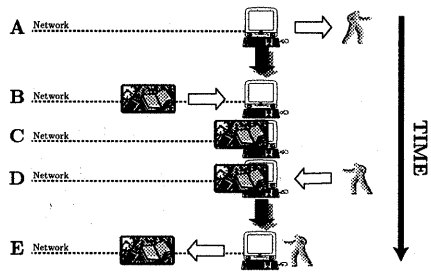

図 1: 提案システムの概要 (各計算機の動作)

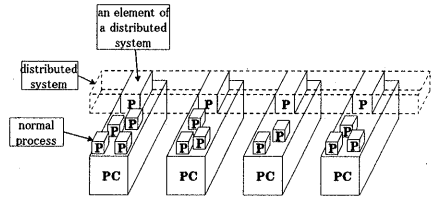

図 2: 提案システムの構造

調システムにユーザ認証機構があり認証ユーザレベルでの資源割り当てが可能である, (4) 動的な環境における適切な資源割当が可能である. ここで, "FG タスク" とは各ユーザが本来その計算機で行う対話的なタスク (ワードプロセッサやブラウザなど) のことであり, "BG タスク" とは計算機不使用時にシステムから投入されるバッチ型のタスク (シミュレーションや数値計算など) のことである. すなわち, 提案システムはユーザが席を離れるなどで "FG タスク" の処理が行われていない時に "BG タスク" を処理するシステムであると言える. ユーザがアイドル状態計算資源を提供するためにユーザ本来の "FG タスク" が妨げられてはならないと考え, "FG タスク" を妨げないことを最重要課題とした. 使用計算機群は信頼性が低いことが予想されるので耐障害性も重要課題とした. さらにタスクスケジューリングも重要と考え, 動的な環境に対応したスケジューリングや公平資源割当スケジューリングの実現も重要課題とした. ただし, これら 2 目標はともにスケジューリングであり必ずしも両立しない.

想定環境はユーザの動作に依存するため非常に不安定なものとしたが, Watch Dog Timer や Token の導入によりシステム内の一部の計算機が障害を起こしてもシステム全体ダウンしたり投入されたタスクが失われないなどを実現した.

スケジューリングとしては効率のよい (スループットの高い) スケジューリングや公平なスケジューリングなどが望まれるが, 効率の良いスケジューリング (公平性は無視) として負荷分散型スケジューリング, 公平なスケジューリングとしてシステム制御型と市場経済モデル型を提案し実装した. 具体的には, 効率の高いスケジューリングとして下記のような機能を実装し, 負荷分散を実現した. まず, 定期的に各スペースは自スペースの負荷状態を計測する. 各スペースは定期的に自スペースの負荷情報を他スペースに通知しする. これによりスペース内の負荷状況を把握しタスクをより負荷の低いスペースに移送する (スケーラビリティ, タスク移送に関しては後述). 1000 個のタスクを 1 台の計算機に投入した結果, 4 分程度で全ての計算機 (9 台) の負荷が均

等化された. また, 各計算機の能力が異なる状況でも計算機の能力を考慮することにより負荷を適切に分散することが可能であることが確認され低能力の計算機でもその能力を考慮することによりシステム全体のスループットの向上に貢献できることが確認された (PentiumIII 1GHz 程度の計算機群に 486 66MHz の計算機を追加し実験). また, 応用例としてパスワード解析プログラムを作成し提案システム上で処理させその実用性を確認した.

次に, システム制御型の公平なスケジューリングであるが, ユーザをアカウントで管理し, 各アカウントごとの消費資源量を監視し制御することにより各ユーザアカウントごとの消費資源量を等しくすることを実現した. 実環境 (各タスクの大きさが異なり計算量比 1, 5, 10 のタスクが等確率で発生. 計算機 7 台, アカウント数 100 の環境で実際にネットワーク越しにタスクを投入し処理させ測定) において, 消費資源量 (CPU) の標準偏差 10% 程度の公平性を実現した. 大規模システムへの対応としては, 計算機 200 台での環境を理想的な環境 (各タスクの計算量と計算機能力が完全に評価できる環境) を想定しシミュレーションした結果, 消費資源の偏りは, 物理的近隣への接続方式 (計算機が 2 次元平面上に配置されているとし距離が一定以内の計算機と論理的に接続をし, 接続がされている計算機のみを管理し公平化を図る) において標準偏差 10% 以下を実現し, ランダム接続方式 (ランダムに計算機同士の論理リンクを確立し, 管理単位ごとの偏りを軽減する) においては標準偏差 4% 以下を実現させた. これにより, システム全体の把握が不可能な大規模システムにおいても接続されている計算機のみを管理することにより各ユーザの消費資源量を公平とすることが実現されたと言える.

最後に, 市場経済モデル資源融通システムでは資源消費ユーザが資源提供ユーザに代価を払い資源を購入することにより資源を消費しタスクを処理する. この方式では各資源提供者エージェントおよび各資源消費者エージェントが資源販売および資源購入を各エージェントの戦略により行う. これにより各ユーザは自分の消費権以内で自分の嗜好に応じた資源消費を自由に行えるようになり, 様々な嗜好を持つユーザが同一システム内で資源を消費や提供を行える. 資源の割り当ては各ユーザの戦略に依存するが, 提案システムの実装では提供者エージェントが強い (資源提供による利益が高い) ほど価格は安定し, 結果としてシステムが安定となり消費者の利益となることが確認された. すなわち, ユーザが利益を求めるほどシステムは安定すると言える. また, 提案システムの実装の経済システムとしての健全性を評価したところ, 均衡価格は経済学における需要と供給が一致する価格 (需要曲線と供給曲線の交点) とほぼ一致し (5% 程度のずれ), 実装システムの経済システムとしての健全性も確認された.

提案システムは Java 言語で実装されているが, タスクの移送は以下の方法で実現されている. Java 言語では Java 言語ソースコードをコンパイルし Java Byte Code と呼ばれる中間言語で記述されたファイルを作成し, その Byte

Code を各プラットフォームのインタプリタで実行する. 提案システムでは Java コンパイラで作成された通常 (移送不可能) の Byte Code ファイルを移送可能な Byte Code ファイルに変換することによりタスク移送を実現している. 具体的には, Byte Code のメソッドコールをフックしプログラム実行コンテンツ (Stack や PC) を獲得できる命令を挿入する. この方法の利点としては, 拡張 VM を使わないためタスクのポータビリティーが失われない, 言語拡張を行う必要がなく従来の Java 言語と高い親和性を持つことがあげられる.

既存の研究としては, まず CONDOR があげられる. これは各計算機 Native のタスクを用いるため高機能な負荷分散やタスク移送 (ファイルなどのリソースの管理など) が実現されているが, 提案システムのような高いタスクの移送性は実現されていない. 市場経済モデルに基づくアイドル状態計算資源融通システムとして POPCORN があげられるがこれは細かいサブタスク ("computelet" と呼ぶ) に分割できる特殊なタスクのみを対象としているが, 提案システムはタスクの移送が可能であり汎用的なタスク実行プラットフォームを実現したと言える.

本研究では, アイドル状態計算資源融通システムとして "BG タスクスペース" を提案し実装した. 提案システムを用いることにより, 自分の本来の作業に支障をきたすことなしに他人のアイドル状態計算資源を用いてタスクを処理させることが可能となった. また, タスク移送により "FG タスク" を妨げないことが, トークンや Watch Dog Timer により耐障害性が, 前述のスケジューリングにより動的な環境な対応したスケジューリングや公平なスケジューリングが実現されたと言える.