広域分散環境における大規模タスク群の挙動を求める高速シミュレータ

松本 真樹† 佐々木 敬泰† 大野 和彦† 近藤 利夫† 中島 浩‡
†三重大学大学院工学研究科情報工学専攻 ‡京都大学学術情報メディアセンター

1 はじめに

近年,広域ネットワーク上に分散している多くのクラスタ群を利用して,大規模並列処理を行うことへの需要が高まっている.大規模並列処理の実行効率はタスクスケジューリングの結果に大きく左右されるため多くの研究が行われてきたが,実環境を用意することが難しく,シミュレーションによる性能評価を行うことが多い.これらの性能評価ではより実環境の挙動に即したシミュレーションを行うことが求められるが,大規模並列処理を扱う場合の計算量は膨大になり,詳細なシミュレーションを行うことは難しい.

そこで,本研究では実環境の挙動を高速に求めることができるシミュレータを実現した.

2 背景

科学技術の分野における並列処理では,大量のデータを解析ソフトで処理し,その結果を統計ソフトを用いて評価するといったワークフローの形を取ることが多い.従って,本稿ではワークフロー型の大規模並列処理を扱う.ワークフローに対応したタスクスケジューリングでは,処理の単位をタスクとして扱い,タスク間にはデータ通信による依存関係が存在している.また,各タスクの計算量・タスク間のデータ通信量は予め与えられているものとする.

このようなタスクケジューリングで最適解を得ることはNP困難であることが示されている.そこで,最初のタスクの実行開始時から最後のタスクの実行終了時までの時間であるスケジューリング長をシミュレータで算出し,これを用いてスケジューリング手法の性能評価を行うことが多い.

3 設計

各タスクの計算処理のシミュレーションを,SimpleScalar[1]等の精度の高いシミュレータで行うと,タスクの計算量に従ってシミュレーション時間が増加する.したがって,大規模並列処理では現実的な時間でシミュレーションが終わらない.そこで本シミュレータでは,イベントドリブン型の抽象シミュレーションを行っている.本シミュレータで扱うイベントは,タスクの実行やデータ通信の開始・終了といったイベントの種類と,このイベントが処理される時刻（イベント発生時刻）の二つの値を持つ.これらのイベントはイベントキューによって管理されており,イベント発生時刻の早いイベントから順に処理される.このイベント処理によって,イベントキューに管理されている他のイベントのイベント発生時刻が更新されることもある.

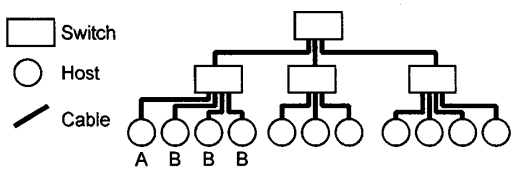


図1: シミュレーションモデルの例

本シミュレータでは,実行環境のモデルを木構造として定義して用いた.これは広域ネットワークが一般的に木構造のような形になっているためである.このモデルでは,葉を「計算機」,枝を「ケーブル」,葉以外の節点を「スイッチ」とした.図1は3クラスタから成る実行環境をモデル化した例である.

このシミュレーションモデルを元に,タスクの計算時間やデータ通信時間を算出し,各イベントの発生時刻を求める.タスクの計算時間はタスクの計算量をタスクを実行する計算機の性能で割った値とした.データ通信時間は,経路上のスイッチやケーブルのレイテンシの和と,データの送信時間を足した値とした.データ送信時間は,経路上のケーブルでバンド幅の中でもっとも低いものを選択し,データ通信量をその値で割った値とした.また,ネットワークの混雑による通信時間の増加を再現するために,ケーブルのバンド幅は,そのケーブルを利用してデータ通信を行っているタスク数に反比例するようにした.

これらの処理により,ネットワークの混雑といった実環境の挙動に即したシミュレーションを高速に行うことができる.

High Speed Simulator for Large-Scale Workflow in Widly Distributed Enviromnet.
†Masaki MATSUMOTO †Takahiro SASAKI †Kazuhiko OHNO
†Toshio KONDO ‡Hiroshi NAKASHIMA
†Graduate School of Engineering, Mie University
‡Academic Center for Computing and Media Studies, Kyoto University

4 実装

本シミュレータでは高速化を実現するために, イベントドリブン型でシミュレーションを行う. しかし, 大規模並列処理を扱う場合イベントキューが管理するイベント数が膨大になり, イベント発生時刻の順にイベントをソートする処理や, イベント発生時刻の更新処理に時間がかかる. そこで二つの手法でさらなる高速化を行った.

イベントキューからのイベント取得・登録を高速化するために, 完全二分木を利用したヒープを使った. イベントの発生時間をキーとして考えれば, イベントキューからは常にキーの一番小さいイベントを取得するため, イベントの追加・削除ともに $O(log(n))$ で実行できるからである. 実装においては, データ構造に完全二分木のヒープを利用した STL の *priority_queue* を利用した.

また, 連続してデータ通信の開始イベントを取得した場合, 最後のデータ通信の開始イベントを取得した後にイベント発生時刻の更新処理を行った. これにより, イベント発生時刻の更新処理が行われる回数を減らすことができる.

5 評価

広域分散環境で大規模分散環境を想定したシミュレーションの実行時間について評価を行った. 図 1 の三種類のワークフローで評価を行った. シミュレーション条件はタスク数を 100 万, 計算機を 10 万台とした. Xeon X3330(2.66GHz, 4 core, L2 6MB), メモリ 2GB の環境でシミュレートし、シミュレーションに要した時間を表 1 に示す.

(a) は (b) のシミュレーション時間と比較すると 2 倍以上速い結果になった. これは,(a) はタスク間のデータ通信イベントが発生しないためであり, また, タスクの開始・終了イベントがデータ通信の開始・終了イベントより処理が軽いためである. (c) は (b) に比べイベント数が 50%ほど多いが, シミュレーション時間は 4%程度しか増加しなかった. データ通信の開始・終了イベントは, イベント発生時刻の更新処理の処理がボトルネックになっているが, この更新処理をまとめて行うことができたためである.

また実環境におけるバンド幅の性能低下を測定するために, 図 1 のホスト A とホスト B のような 4 台の PC を一台のスイッチに接続した環境を用意し, ホスト A からホスト B への 1:N 通信を行い, 評価を行った. 表 2 は全てのホスト B へのデータ転送にかかった時間で, 用いたデータのサイズは 1GB とした.

本シミュレータでは 1 対多通信を行った場合, ケーブルの通信性能は通信しているホスト数に反比例するようなモデルにすることで, シミュレーションの高速化を行っている. このような単純なモデルでも実機評価との誤差は 10%程度であり, ケーブルのバンド幅の性能低下の傾向を再現できていると言える. 今後はさらにモデルを改良し, この誤差を小さくしていく必要がある.

表 1: シミュレーション時間

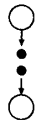
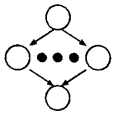
ワークフロー	(a)	(b)	(c)
			
シミュレーション時間 (秒)	97.46	227.04	235.45
イベント数	2,000,000	3,999,998	5,999,992

表 2: 1 対多のデータ転送時間の評価

	1:1 通信	1:2 通信	1:3 通信
データ転送時間 (秒)	10.24	18.51	27.77

6 おわりに

本稿では, 広域分散環境における大規模タスク群の挙動を求める高速シミュレータについての実装方法について述べた. 大規模タスク群において, ネットワークの混雑によるデータ通信時間の増加の傾向を再現しつつ, 高速なシミュレーションが行えた. 今後, 様々な実機評価を行い, より誤差の小さいシミュレーションモデルにしていく必要がある.

謝辞

本研究の一部は文部科学省科学研究費補助金 (特定領域研究, 研究課題番号 21013025,「タスクと実行環境の高精度モデルに基づくスケーラブルなタスクスケジューリング技術」) による.

参考文献

[1] Todd Austin, Eric Larson, and Dan Ernst. Simplescalar: An infrastructure for computer system modeling. *Computer*, Vol. 35, pp. 59–67, 2002.