シミュレー ションの高速 化と精度向上

和田 篤

る高速化 離散イベントシミュ

レーション 悲観的な方法 楽観的な方法

データ同化による高精度化 三次元データ同化 四次元データ同化

シミュレーションの高速化と精度 向上

シミュレーションの並列処理とデータ同化

和田篤

October 27, 2016

動機と狙い

シミュレー ションの高速 化と精度向上

和田 篤

並列処理による高速化 る高速化 離散イベントシミュ レーション 悲観的な方法 楽観的な方法

データ同化による高精度化 三次元データ同化 四次元データ同化

テーマ選択理由

- シミュレーションの精度を上げたい,計算時間を短くしたいという声をよく聞く
- 得意分野の機械学習, アルゴリズムでシミュレーション技 術の役に立ちたい

狙い

- 並列化とデータ同化技術のアイデアと基本的な概念を 紹介
- 技術開発方針の一助に

目次

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化 ^{業敗イベントシミ・}

離散イベントシミュ レーション 悲観的な方法 楽観的な方法

データ同化による高精度化 ニ次元データ同化 四次元データ同化

- 1 並列処理による高速化
 - 離散イベントシミュレーション
 - 悲観的な方法
 - 楽観的な方法
- 2 データ同化による高精度化
 - 三次元データ同化
 - 四次元データ同化

シミュレーションの分類

シミュレー ションの高速 化と精度向上

和田 篤

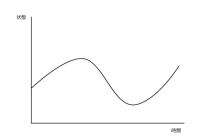
业列処理による高速化 都放イベントシミュ

を観的な方法 を観的な方法

データ同化による高精度化 ニ次元データ同化

状態変化の数による分類

- 連続ミュレーション: 状態変化が無限回
- 離散イベントシミュレーション: 状態変化が有限回



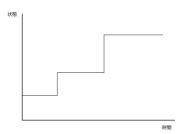


Figure: 連続シミュレーション

Figure: 離散シミュレーション。

離散イベントシミュレーション (DES)

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化

離散イベントシミュ レーション 悲観的な方法

悲観的な方法 楽観的な方法

データ同化に よる高精度化 三次元データ同化 四次元データ同化

DES の主要構成要素

時刻 シミュレーション上の時刻

状態 シミュレーションで計算したい値

イベント 状態を変化させる & 新たなイベントを生成

①処理開始@5:00 ③処理開始@6:10 ②処理終了@6:00 ④処理終了@7:00 装置A 装置B

Figure: DES の例

Table: DES の例 (詳細)

イベント 時刻 状態 4:00 DES 開始 5:00 { 装置 A: 処理中 } 6:00 { 装置 A 工数:1} 6:10 装置 A 工数:1, 装置 B: 処理中 } 7.00 装置 A 工数:1. 装置 B 工数: 1} 8:00 DES 終了 装置 A 工数:1, 装置 B 工数: 1}

DES に対するアルゴリズム: time-bucket 法

シミュレー ションの高速 化と精度向上

和田 篤

る高速化 離散イベントシミュ

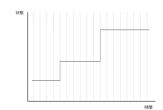
悲観的な方法 楽観的な方法

データ同化による高精度化 三次元データ同化 四次元データ同化

time-bucket 法 (単位時間ごとに計算)

単位時間を定義し,単位時間ごとに処理すべきイベントがあるか確認する.

- 1: while 終了条件を満たさない do
- 2: 時刻を単位時間すすめる
- 3: if 現在時刻に処理すべきイベントがある then
- 4: イベントを処理する.



DES に対するアルゴリズム: event-driven 法

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化

離散イベントシミュ レーション 悲観的な方法 楽観的な方法

データ同化による高精度化 三次元データ同化

event-driven 法 (状態変化時のみ計算)

イベントキューを利用して,イベントが発生する時刻のみ計 算する.

- 1: while 終了条件を満たさない do
- 2: イベントキューの先頭にあるイベントを処理
- 3: 時刻を取り出したイベントの発生時刻にする
- 4: イベントを処理する

Parallel DESの基本アイデア

シミュレー ションの高速 化と精度向上

和田 篤

並列処理による高速化 る高速化 ないとときュ レーション

レーション 悲観的な方法 楽観的な方法

データ同化はよる高精度化 よる高精度化 三次元データ同代

アイデア

別々の装置は並列に処理できるのではないか?

基本概念

LogicalProcess イベントを処理する仮想エンティティ.

イベント 1 つの LogicalProcess で発生し, その LogicalProcess の状態のみ変化させる.

メッセージ イベントが発生した際に,他の Logical Process に 送信するイベントのこと.

並列処理可能である十分条件

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化 ^{離散イベントシミコ} レーション

悲観的な方法 楽観的な方法

データ同化による高精度化 こ次元データ同化

並列処理可能である十分条件

Q: どんな時に並列に処理しても結果が変わらないか? A: イベントを処理する順序が変わらない場合.ある LogicalProcess は今後,送られてきうるメッセージの開始可能 時間より小さい時刻では処理してもよい. シミュレー ションの高速 化と精度向上

和田 篤

並列処理による高速化

ルーション

悲観的な方法 楽観的な方法

データ同化による高精度化 三次元データ同化 ネットワークベースの方法

null-message

シミュレー ションの高速 化と精度向上

和田 篤

並列処理による高速化 高速化 ^{離散イベントシミ} レーション 悲観的な方法

※銀門な方法 データ同化に よる高精度化

課題

Queue が空の場合処理できない安全なメッセージのみ処理すればいいのか

解決方法

null-message: 状態を変化させないメッセージ arc の時間をす すめるために,時刻のみのメッセージを追加する

time-warp 法の動機

シミュレー ションの高速 化と精度向上

和田 篤

並列処理による高速化 離散イベントシミ: レーション 悲観的な方法 楽観的な方法

データ同化による高精度化 こ次元データ同化

time-link 法の課題

time-link 法では, safe である可能性が高いが確実でない場合に, 並列処理できない. また, null-message を多用した場合オーバーヘッドが大きい.

time-warp 法のアイデア

time-warp 法では上記の状況でまず並列に処理し, safe でなかった場合 rollback することで並列性を高める.

rollback

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化 ^{離散イベントシミニ} レーション 悲報的な方法 **楽報的な方法**

データ同化は よる高精度化 EX元データ同化 ある LP で起きたイベントが safe でなかった場合 , 何をロール バックすればよいだろうか ?

イベントの作用

LPの状態を変化させる 他のLPにメッセージを送信する

<u>ロー</u>ルバックの種類

そこで,以下2種類のロールバックが必要となる.

restolation: LP の状態 antimessage: 他の LP に

restolation

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化

離散イベントシ: レーション

悲観的な方法 楽観的な方法

楽観的な方法

ナーダ同化による高精度化 三次元データ同化 а

antimessage

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化

離散イベントション

悲観的な方法 楽観的な方法

楽観的な方法

テータ同化に よる高精度化 E次元データ同化 а

動機

シミュレー ションの高速 化と精度向上

和田 篤

а

シミュレー ションの高速 化と精度向上

和田 篤

並列処理によ る高速化

ルーション 悲観的な方法 楽観的な方法

データ同化による高精度化 ニ次元データ同化 四次元データ同化

- Fujimoto, Richard M. "Parallel discrete event simulation." Communications of the ACM 33.10 (1990): 30-53.
 - Fujimoto, Richard M. Parallel and distributed simulation systems. Vol. 300. New York: Wiley, 2000.
 - Jefferson, David, and Henry A. Sowizral. "Fast concurrent simulation using the time warp mechanism." (1982).
 - Chandy, K. Mani, and Jayadev Misra. "Distributed simulation: A case study in design and verification of distributed programs." IEEE Transactions on software engineering 5 (1979): 440-452.