1. 研究分野
　　可逆計算，並列シミュレータ
2. 目的
　　PDESのロールバック方法に可逆計算を応用する.
3. 背景
　　楽観的な並列シミュレーションは，正しくない計算（計算順序の間違い）が起こった場合に計算を戻す必要がある. 計算を戻すにはメモリの状態を復元すればよい. 今までは状態保存の方法が使われていたが，可逆計算は代替手法として検討されていなかった.
4. アプローチ
　　コードを可逆化し，シミュレータに可逆計算を導入して従来の方法と2つのモデルで比較する.
5. 結果
　　オーバヘッドの削減により，並列実行では最大で300~500%もの性能向上が見られた.
6. 有用性
　　粒度の小さい並列シミュレーションを実用レベルにまで引き上げることができた.
7. 限界・短所
　　すべての並列シミュレーションにおいて最適という訳ではなく，粒度の大きい（イベントごとの計算量が多い）ものは従来の方法と変わらない可能性がある.
8. 次に何を読めばいいか？

コードの可逆化
・スナップショット法よりも保存する状態が少なくなる
・イベント計算が細粒度（計算量が少ない）ものほど有利
・自動化可能
・状態の保存が支配的な計算となっているため、それが少なくなれば実行性能も上がる
・状態を保存するコストを下げることは実行性能を上げることに繋がる
乱数生成の可逆化
・コードを単純に可逆化しただけではスナップショット法と変わらない
・mod計算では逆数を求める方法が存在し、その計算はコードの中にもある
・つまり、個々の計算が非可逆的であっても、全体の計算は可逆的である場合がある

パフォーマンス比較
GTWは楽観的並列シミュレータ
手作業での可逆化GTW、自動変換での可逆化GTW、スナップショット法GTW（イベント一回ごと、定期的、インクリメンタルステートセービング）、最適化されたシーケンシャルシミュレータ（並列実行はできない）、並列実行できないGTWでの比較を行った

GTWに逆計算を追加するために、APIを拡張した。
具体的には、LPのイベントハンドラの逆実行が可能になるTWLP[i].RevProcを導入した。
TWLP[i].RevProcの引数にはLPの現在の状態と順実行中に送信されたイベントが含まれる。
次に、ロールバックの際にイベントをタイムスタンプに沿った逆の順序で逆実行できるように変更した。

最後に、状態の保存をオフにした。その代わりに、全てのイベントに制御情報を保存するための1ビット列(bit vector)を追加した。

GTW-RCはテーブルの規則を手動で適用したもの。GTW-RCCはRCCを用いたもの。
GTW-CSSはイベントの実行前に状態を保存する従来のもの。GTW-PSSはイベントのpステップごとに状態を保存するもの。GTW-ISSは変化した部分のみが変化前に保存されるもの。

ATM マルチプレクサカスケード
ATMプロトコルを利用したネットワークのモデルのシミュレーション。粒度が小さい。先読みができる。

PCSネットワーク
PCSのネットワーク（無線通話）のモデルのシミュレーション。粒度が大きい。先読みができない複雑な通信パターン。現実世界のシミュレーションモデルで逆計算のロールバックダイナミクスをどう実行するかの代表的な例となっている。

順実行の比較
状態を保存しないGTW-NONEとGTW-RCは同等の速度だが、GTW-CSSよりも早い。これは状態保存のオーバヘッドが排除されたから。ただし、GTW-SEQよりは遅い。その理由はメモリバッファがGTW-SEQよりも長い間使えないから。
ファンイン(入力数)が増えるほど各シミュレータは同じ実行速度に収束する。これは、ファンインが増えるとイベントリスト管理のオーバヘッドが支配的になるため

並列実行の比較
ATMマルチプレクサモデルでは最大300%、PCSネットワークモデルでは最大500%ものパフォーマンスが向上した。GTW-RCは状態保存よりも一貫して高速。
状態はメモリの中に保存されるため，シミュレーションのメモリ占有率が上がる。そのためCSSではTLBミスが増加し、全体の性能が低下する。更に、状態のコピーを作成するために多くのメモリページにアクセスする。これによりTLBミス、キャッシュミスが更に増加し、全体の性能が低下する。
PSSでは、CSSよりも性能が向上することが期待される。だが、イベントサイズが状態サイズ以上である場合、CSSとあまり性能が変わらないことがある。それは、CSSよりも多くのイベントを保持しておく必要があるため。
ISSでは、状態が変更された部分のみ保存するため、CSSよりも性能が向上することが期待される。だが、状態のサイズが小さい場合、CSSとあまり性能が変わらないことがある。それは、変更のログを維持するオーバヘッドが大きく、CSSと同じほどのオーバヘッドがかかるからである。
RCとRCCでは、RCの方が性能が少し高い。それは、RCの方がより最適化されているからである。

性能のまとめ
イベント粒度が小さいシミュレーションにおいて、可逆計算を用いたものは許容可能なレベルの性能を達成できた。これは、状態の保存のコストが支配的であり、それを解消できたからだ。これは、メモリ操作とCPU計算のギャップに起因しており、このギャップが開くほど、可逆計算法は状態保存よりも性能が向上すると期待される。

まとめ
逆計算は破壊的でない代入があるモデルに適している。ただし、破壊的な代入が十分な数存在する場合、従来の状態保存と変わらないことがある。
逆計算を適用するのは難しい。なぜなら、ロールバック時にイベントを処理の逆順で逆実行しなければならないため、高度なイベント間分析を行わなければ通常の状態保存よりもオーバヘッドが増加してしまうからだ。
逆計算は楽観的な小さな粒度のシミュレーションを実用的なレベルにまで引き上げるのに魅力的な方法である。
逆計算法を使用することで、今までコストが高すぎると考えられていた楽観的シミュレーションが、現実的なコストで行えるようになる可能性がある。