GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

Dynamischer Lastausgleich optimistisch synchronisierter, verteilter Simulation

Rolf Schlagenhaft
(Schlagenhaft@ei.tum.de)
Lehrstuhl für Entwurfsautomatisierung
Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München
(http://eda.ei.tum.de/index.html)

1 Einleitung

Die diskrete, ereignisgesteuerte Simulation wurde in den letzten Jahren weltweit auf verschiedene Arten und für fast alle verteilten Rechnerarchitekturen parallelisiert. Die entwickelten Verfahren unterscheiden sich unter anderem in der Art der Synchronisation zwischen den Programmteilen auf den einzelnen Prozessoren. Unter den Alternativen hat sich das auch hier eingesetzte optimistische Time–Warp–Protokoll unter Verwendung von Message Passing als besonders geeignet für die Simulation auf vernetzten Arbeitsplatzrechnern herausgestellt.

Neben der Art der Synchronisation ist die sinnvolle Aufteilung der zu lösenden Aufgabe wie bei jedem verteilten Programm ein wesentlicher Aspekt für die erzielbare Beschleunigung. Fast alle bisherigen Arbeiten zur Synchronisation und Partitionierung gehen leider von einem sehr homogenen Verhalten der Hardware und des Simulationsmodells aus. Diese Idealisierung trifft in der Praxis meist nicht zu, was zu erheblichen Leistungseinbußen für Parallelsimulatoren führt.

Die vorliegende Arbeit stellt nun ein Verfahren zum dynamischen Lastausgleich vor, das es erlaubt, nicht nur in einem idealen Umfeld sondern auch in praxisgerechten Situationen gute Beschleunigungen unter Beibehaltung des Time–Warp–Protokolls zu erreichen.

2 Gestörte, verteilte Simulation

In der Literatur sind sehr gute Beschleunigungen der verteilten, ereignisgesteuerten Simulation unter Verwendung von optimistischen Synchronisationsprotokollen zu finden. Diese wurden meist in einem universitären Umfeld erzielt. Dort unterliegen die Experimente normalerweise weniger äußeren Einflüssen und Beschränkungen als in einer industriellen Umgebung. In praxisnahen, verteilten Rechnerumgebungen gehen die möglichen Beschleunigungen jedoch aufgrund des dort vorhandenen nichtidealen Umfeldes schnell verloren, da verteilte Simulatoren sehr empfindlich gegen verschiedene Arten von Störungen sind. Diese lassen sich in zwei Hauptgruppen einordnen:

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

- **Äußere Störungen, verursacht durch das eingesetzte Rechnersystem:**
  - Inhomogene Rechnerleistung: Nicht immer ist es möglich, gleich leistungsfähige Rechner zur verteilten Simulation einzusetzen. Insbesondere bei der Verwendung von Rechnernetzen handelt es sich meist um eine heterogene Umgebung. Nur selten werden die unterschiedlichen Rechenleistungen bei der Partitionierung des Simulationsmodells berücksichtigt.
  - Fremde Benutzerprozesse: In der Praxis steht die verwendete Hardware selten exklusiv für die verteilte Simulation zur Verfügung. Fremde Benutzer- oder Systemprozesse auf einzelnen Prozessoren zerstören ebenfalls die Homogenität des parallelen Systems und führen zu einer Störung des Simulationssystems.
  - Speicherplatzprobleme: Kommt es auf einem der eingesetzten Rechnerknoten durch den Simulationsprozeß oder andere Benutzerprozesse zu Speicherplatzengpässen, so beeinflußt dies ebenfalls sofort die gesamte verteilte Simulation.
- **Innere Störungen, verursacht durch das Simulationsmodell:**
  - Suboptimale Partitionierung: Jede Partitionierung geht von einem bestimmten Rechenzeitbedarf der Modellkomponenten aus. Dieser ist jedoch nicht immer im Voraus bekannt oder hängt zusätzlich von den anregenden Ereignissen, den Stimuli, ab. Deshalb nehmen die meisten Partitionieralgorithmen eine Gleichverteilung des Rechenzeitbedarfs aller Modellkomponenten an. Diese Vereinfachung führt häufig zu einer schlechten Startpartitionierung. Wird diese während der ganzen Simulation beibehalten (statische Partitionierung), wirkt sich das negativ auf die Programmlaufzeit aus.
  - Schwankungen der Modellaktivität: Selbst wenn eine ideale Startpartitionierung des Simulationsmodells möglich ist, kann es im Laufe der Simulation durch die anregenden Ereignisse zu größeren Verschiebungen der Modellaktivität zwischen den Partitionen kommen. Diese dynamischen Veränderungen können durch eine rein statische Partitionierung nicht berücksichtigt werden.

Diese Liste ist keinesfalls vollständig, sondern soll nur eine Auswahl möglicher Ursachen für Leistungseinbußen bei der verteilten Simulation zeigen. Alle angesprochenen und darüberhinaus denkbaren Störungen haben die gemeinsame Eigenschaft, daß sie die Leistungsfähigkeit beziehungsweise die Geschwindigkeit einzelner oder mehrerer Simulatoren beeinträchtigen und das Simulationssystem dadurch aus dem Gleichgewicht bringen. Da die auf Modellpartitionierung basierende Parallelsimulation aber erst dann vollständig beendet ist, wenn alle Simulatoren das Ende der Modellzeit erreicht haben, wirkt jede Störung verlängernd auf die Verweilzeit des gesamten Simulationsauftrags im Rechnersystem und erhöht daher die Wartezeit des Benutzers auf die Ergebnisse. Diesem Effekt kann durch eine rein statische Partitionierung nicht begegnet werden. Nur eine dynamische Umverteilung der Simulationsarbeit während der Programmlaufzeit kann dieses Problem lösen.

3 Dynamischer Lastausgleich

Das System zum dynamischen Lastausgleich verteilter Simulation beobachtet ähnlich wie ein Regelungssystem laufend die Situation und reagiert bei Abweichungen von einem Sollzustand

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

adaptiv darauf. Die Hauptblöcke und der Informationsfluß (Abb. 1) der im Folgenden vorgestellten Lösung ähnelt daher in seiner Struktur auch stark einem herkömmlichen Regelkreis.

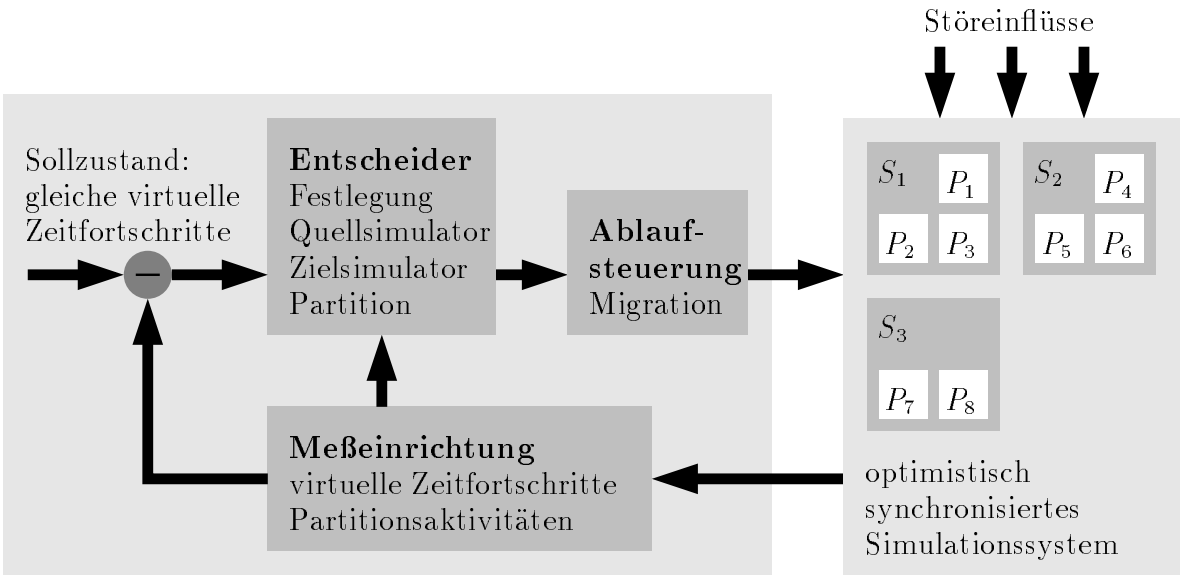

Abbildung 1: Blockbild und Informationsfluß

Der hellgraue Block rechts stellt das zu optimierende verteilte Simulationssystem bestehend aus den Simulatoren S_i dar. Der linke hellgraue Block zeigt das System zum dynamischen Lastausgleich. Es besteht aus drei Hauptblöcken:

- **Meßeinrichtung:** Sie ist die erste wichtige Komponente im Informationsfluß und dient der Beurteilung der Lastsituation anhand von wenigen, aber aussagekräftigen Meßgrößen. Diese zeigen Veränderungen im System schnell an, stören dabei aber das Simulationssystem so wenig wie möglich.
- **Entscheider:** Basierend auf der Abweichung des beobachteten Systemzustands von dem angestrebten Optimum werden hier die Entscheidungen zur Durchführung von Lastausgleichsschritten, das heißt zur Umverteilung von Simulationsaufgaben, gefällt.
- **Ablaufsteuerung:** Die dritte Hauptkomponente ist verantwortlich für die technische Umsetzung der festgelegten Aktionen zum dynamischen Lastausgleich. Sie sorgt für die tatsächliche Verlagerung von Simulationsaufgaben. Dabei wird ebenfalls die normale Arbeit des Simulationssystems so wenig wie möglich beeinträchtigt.

Die Lösung dieser drei Teilproblemstellungen wird in den folgenden drei Unterabschnitten jeweils einzeln vorgestellt.

3.1 Lasterfassung

Für die vorliegende Problemstellung wird ein völlig eigenständiger Lastbegriff entwickelt. Er orientiert sich nicht an Systemeigenschaften wie zum Beispiel der CPU-Auslastung, der Pro-

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

zeßwarteschlangenlänge oder dem Kommunikationsaufkommen, sondern ist ausschließlich auf den verteilten Simulationsalgorithmus zugeschnitten. Dem im Folgenden vorgestellten Lastbegriff liegt die Betrachtungsweise zugrunde, daß ein einzelner Simulator dann als belastet (entlastet) eingestuft wird, wenn er langsam (schnell) in der Modellzeit fortschreitet. Durch diese anwendungsorientierte Sicht entsteht ein Lastmaß, welches völlig unabhängig von der konkreten Ursache einer einzelnen Störung die von ihr verursachte Verlangsamung anzeigt. Auf diese Art und Weise wird eine Entkopplung der Lastbeurteilung von den Systemeigenschaften erreicht und das Verfahren eignet sich gleichermaßen für alle denkbaren Störquellen.

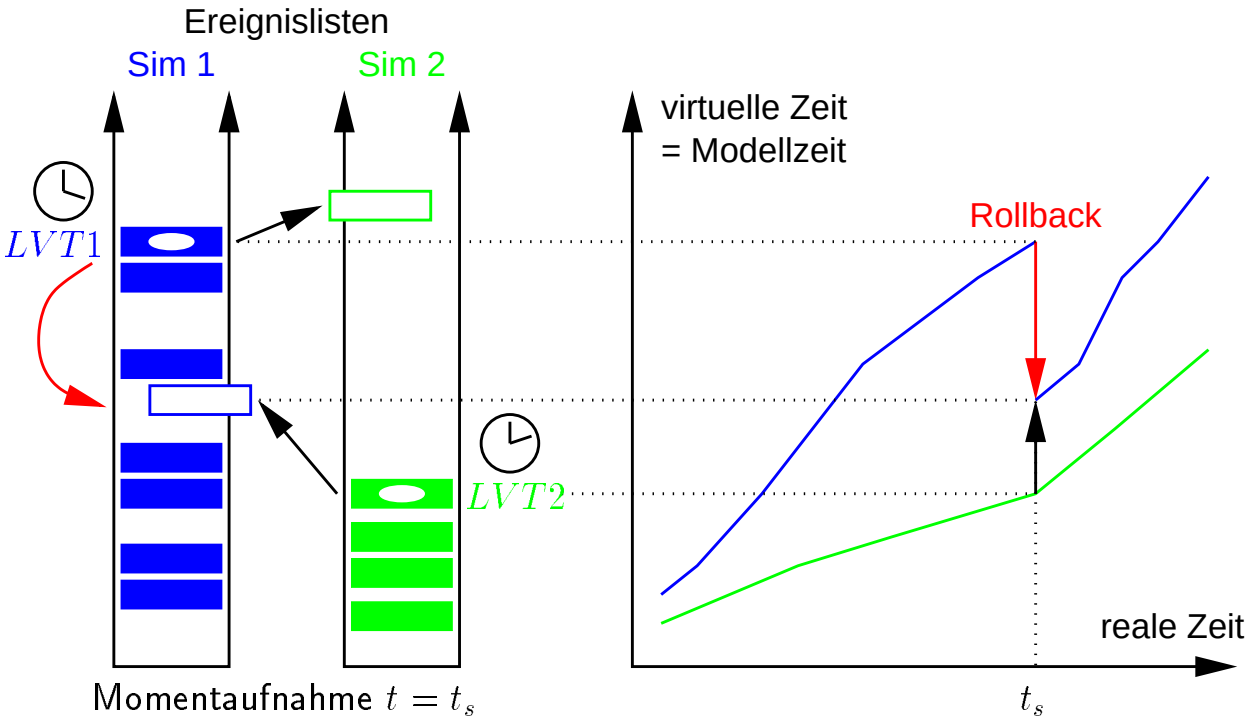

Abbildung 2: Time–Warp–Protokoll

Für das Verständnis des Lastmaßes ist es hilfreich, anhand von Abbildung 2 nochmal kurz einige Aspekte der optimistischen Synchronisation des verteilten Simulators durch das Time–Warp–Protokoll zu rekapitulieren. Bei der optimistischen Synchronisation ist es prinzipiell jedem Simulator erlaubt, Ereignisse so schnell er kann auszuführen und dadurch in der Modellzeit fortzuschreiten. Dies führt dazu, daß sich die Simulatoren zu einem realen Zeitpunkt t_s bei unterschiedlichen Modellzeitpunkten befinden können (lokale virtuelle Zeiten $LVT1, LVT2$). Zwischen den Simulatoren werden Nachrichten ausgetauscht, die zeitstempelbehaftete Ereignisse beinhalten. Durch die fehlende explizite Synchronisation der Modelluhren kann es nun vorkommen, daß ein Simulator (Sim 1) ein Ereignis empfängt, welches zu einem Modellzeitpunkt ausgeführt hätte werden müssen, der von ihm bereits abgearbeitet ist. In diesem Fall muß er Teile der durchgeführten Simulation verwerfen und durch Zustandsrestaurierung zu der Modellzeit des empfangenen Ereignisses zurückkehren. Dieser Mechanismus wird als Rollback bezeichnet. Eine hohe Anzahl von Rollbacks wirkt sich negativ auf die Leistung des Simulationssystems aus. Die Wahrscheinlichkeit für das Auftreten von Rollbacks wächst mit der Differenz zwischen den Modellzeiten der einzel-

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

nen Simulatoren. Hier läßt sich nun erkennen, warum Time–Warp–Systeme so empfindlich auf Störungen reagieren. Störungen bewirken, daß die Simulatoren nicht mehr gleich schnell in der Modellzeit fortschreiten können. Dadurch wird die Differenz zwischen den Modellzeiten größer und die Anzahl von Rollbacks steigt.

Ziel des dynamischen Lastausgleichs ist daher die Angleichung der Geschwindigkeit, mit der die Simulatoren in der Modellzeit fortschreiten. Als Lastmaß dient daher genau diese Geschwindigkeit der einzelnen Simulatoren. Durch den diskreten Charakter der Modellzeit und das Auftreten von Rollbacks kann dazu nicht einfach die Modellzeit (LVT) nach der realen Zeit differenziert werden. Stattdessen wird ein Maß verwendet, welches auf den diskreten Zeitschritten ΔT_n der Modellzeit der Simulatoren beruht (siehe Abb. 3). Bei der normalen Simulation ist ΔT_n jeweils größer Null, bei Rollbacks kleiner Null. Die erste Stufe auf dem Weg zu dem gewünschten Lastmaß besteht in der Vernachlässigung von Rollbacks bei der Bildung der Summe virtueller Zeitschritte (IVT). Anschließend wird basierend auf einem Differenzenquotienten und einer Filterung (nicht dargestellt) der virtuelle Zeitfortschritt bestimmt (VTP).

lokale, virtuelle Zeit (LVT):
$LVT_{n+1} = LVT_n + \Delta T_n$

Summe virtueller Zeitschritte (IVT):
$IVT_{n+1} = IVT_n + \Delta T_n, \quad \text{falls } \Delta T_n \geq 0$
$IVT_{n+1} = IVT_n, \quad \text{falls } \Delta T_n < 0$

virtueller Zeitfortschritt (VTP):
$VTP_n = \frac{IVT_n - IVT_{n-1}}{t_n - t_{n-1}}$

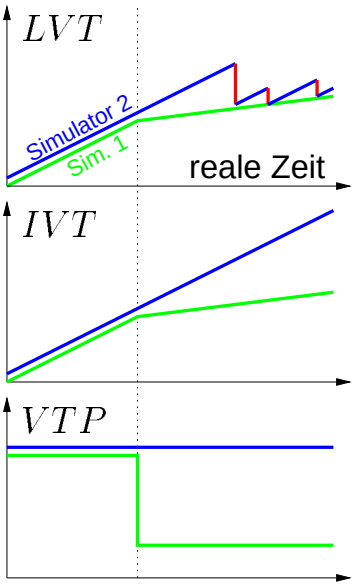

Abbildung 3: Virtueller Zeitfortschritt

Die virtuellen Zeitfortschritte der einzelnen Simulatoren bilden das in diesem Verfahren verwendete inverse Lastmaß. Leistungsfähige Simulatoren zeichnen sich durch einen hohen Wert des virtuellen Zeitfortschritts aus, stark belastete durch einen niedrigen. Die Besonderheit dieses Lastmaßes besteht darin, daß es sich ausschließlich an den Bedürfnissen der verteilten, ereignisgesteuerten Simulation orientiert und abgesehen von der Systemuhr keine Informationen vom Betriebssystem über Zustände der eingesetzten Rechner benötigt.

3.2 Lastverlagerung

Als zweite der drei Hauptkomponenten wird die **Migrationseinheit** vorgestellt. Ihre Aufgabe ist es, Teile des Simulationsmodells von überlasteten zu gering belasteten Simulatoren zu verlagern.

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

Herkömmliche verteilte, ereignisgesteuerte Simulatoren arbeiten mit einer Modellpartition und Ereignisliste pro Simulator. Um Teile des Simulationsmodells zu verlagern, wäre eine komplette Repartitionierung des Modells und eine Umsortierung aller beteiligten Ereignislisten nötig. Dies würde einen erheblichen Rechenaufwand bedeuten und dadurch das Simulationssystem eher verlangsamen als beschleunigen. Aus diesem Grund wird hier der Ansatz der Multipartitionierung verfolgt, bei dem jeder Simulator mehrere Partitionen und ihre zugehörigen Ereignislisten bearbeitet (Abb. 4).

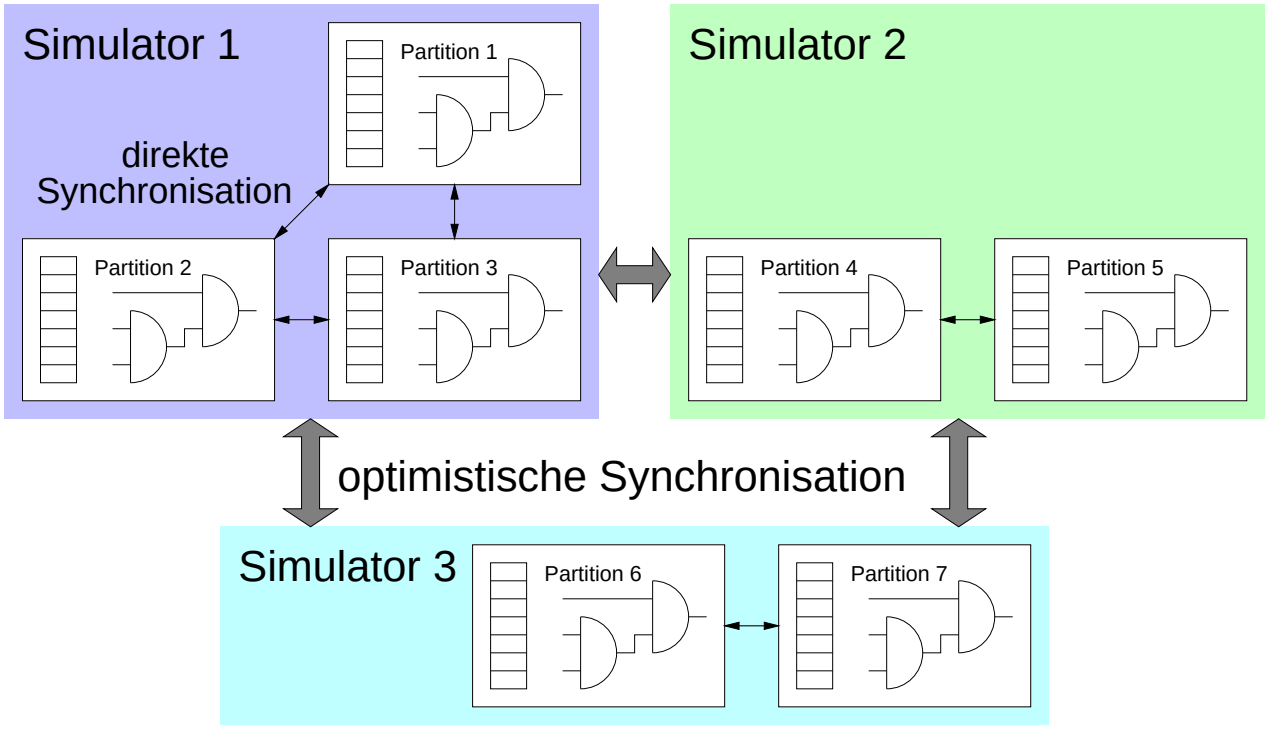

Abbildung 4: Multipartitionierung

Die teure Repartitionierung und Neusortierung der Ereignislisten kann entfallen, indem komplette Partitionen zwischen den Simulatoren verschoben werden, um den gewünschten Lastausgleich zu erzielen. Darüberhinaus kann nun bei der Partitionierung durch die von der Simulatoranzahl entkoppelte, frei wählbare Partitionsanzahl eine für den Lastausgleich günstige Partitionsgröße eingestellt werden. Die Migration einer Partition (Abb. 5) gliedert sich grob in mehrere Stufen:

- Im ersten Schritt wird beim Empfänger, dem Zielsimulator (Simulator 2), das Simulationsmodell der migrierten Partition aufgebaut. Es besteht aus der statischen Modellstruktur und den Zuständen der Modellkomponenten zu einem gewissen Modellzeitpunkt.

- Als nächstes wird die Ereignisliste der migrierten Partition übertragen. Danach kann die Partition beim Sender, dem Quellsimulator (Simulator 1), bereits gelöscht werden und der Zielsimulator die Simulation der migrierten Partition fortsetzen.

- Anschließend werden alle unbeteiligten Simulatoren über den neuen Aufenthaltsort der

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

migrierten Partition informiert, damit diese ihre Simulationsnachrichten richtig adressieren können.

- Durch nicht vermeidbare Verzögerungen auf den Kommunikationskanälen zwischen den Simulatoren kann nicht ausgeschlossen werden, daß nach der Migration noch verspätet Nachrichten für die verlagerte Partition beim Quellsimulator ankommen. Diese werden einfach an den Zielsimulator weitergeleitet und dort durch das Time–Warp–Protokoll richtig in die dortige Simulation integriert.

Darüberhinaus sorgen weitere kleine Erweiterungen des Time–Warp–Protokolls für korrekte Synchronisation und Simulation auch bei der dynamischen Verlagerung von Partitionen. Ein Vorteil dieses Migrationsverfahrens ist, daß nur bei den beiden unmittelbar an einer Migration beteiligten Simulatoren zusätzlicher Rechenaufwand entsteht. Die unbeteiligten Simulatoren können während einer Migration unverändert ihren normalen Simulationsaufgaben nachkommen.

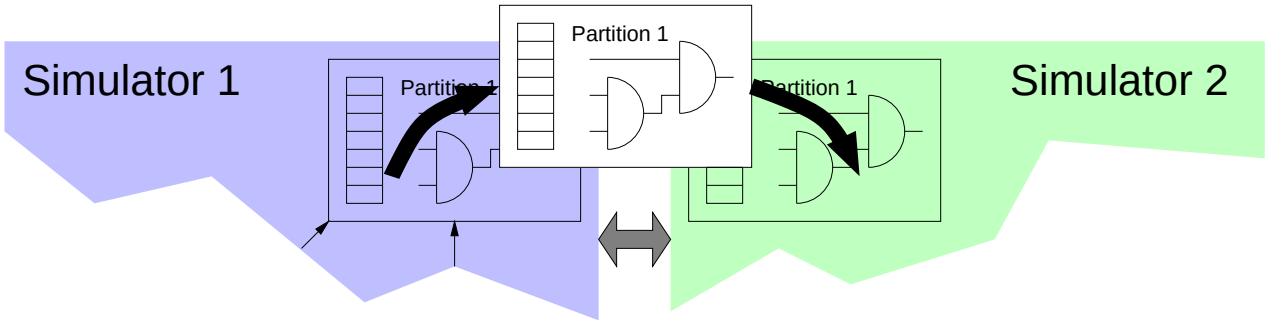

Abbildung 5: Migration einer Partition

3.3 Ausgleichsstrategie

Die dritte und letzte Komponente im Informationskreislauf von Abbildung 1 ist der **Entscheider**, in dem die Strategie festgelegt ist, wann welche Partition zwischen welchen Simulatoren verschoben werden soll. Zur Lösung dieser Aufgabe werden kontinuierlich die virtuellen Zeitfortschritte (VTP) der Simulatoren beobachtet. Bei jeder Veränderung wird jeweils der schnellste und langsamste Simulator festgestellt. Basierend auf ihrem Geschwindigkeitsunterschied wird beim langsamsten Simulator nach einer Partition gesucht, die am besten geeignet wäre, durch ihre Verlagerung diese Differenz auszugleichen. Die beiden bestimmten Simulatoren und die ausgewählte Partition dienen als Kandidaten für die nun folgende Rentabilitätsabschätzung. Dabei wird das Verhalten des Systems mit und ohne Durchführung der Migration vorhergesagt und ein Break-Even-Punkt (t_{BE}) bestimmt, ab dem sich die Verlagerung voraussichtlich ausgezahlt haben wird (Abb. 6). Grundlage hierfür sind Prognosen für den virtuellen Zeitfortschritt des Systems mit und ohne Migration (VTP_{neu} bzw. VTP_{min}) und die zu erwartende Verzögerung bei Quell- und Zielsimulator (t_M) durch die Migration. Die Migration wird nur dann tatsächlich veranlaßt, wenn der prognostizierte Break-Even-Punkt unter einer festgelegten Grenze t_{BEmax} liegt. Läge er zu weit in der Zukunft, bestünde die Gefahr, daß sich die Lastsituation bis dorthin erneut verändert und der Lastausgleichsschritt wirkungslos wird.

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

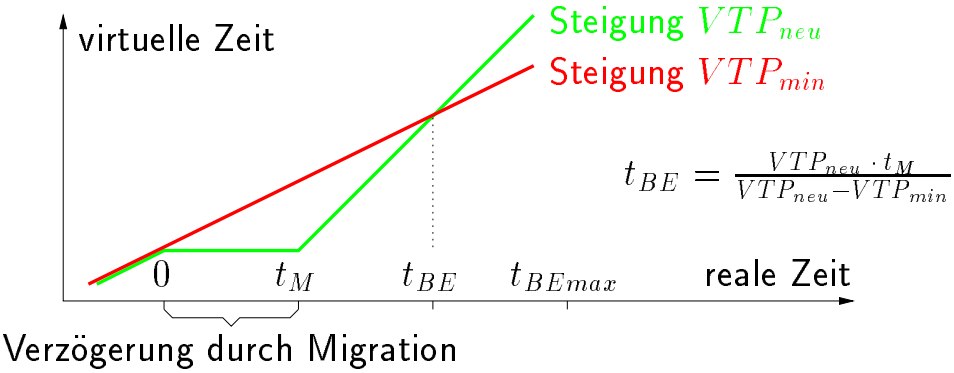

Abbildung 6: Rentabilitätsabschätzung

Durch diese letzte Komponente ist der Kreislauf der adaptiven Lastregelung geschlossen. Daten vom Betriebssystem über den Lastzustand des Rechnernetzes werden für dieses Verfahren nicht benötigt. Eine Ursachenbestimmung, quantitative Erfassung und Berücksichtigung einzelner konkreter Störquellen ist ebenfalls nicht notwendig.

4 Ergebnisse

In Abbildung 7 sind mit dem System experimentell bestimmte Programmlaufzeiten (nicht CPU-Zeiten!) dargestellt. Als Simulationsmodell für die Versuche dienten die Schaltungen aus dem MCNC-Benchmark-Set [BBK89]. Als repräsentatives Beispiel werden hier die Laufzeiten für Schaltung s35932 abgebildet und beschrieben.

Nach rechts sind verschiedene parametrisierte Simulationsläufe beziehungsweise Partitionierungen aufgetragen. Die Beschriftung 6/24 bedeutet zum Beispiel, daß sechs Prozessoren zur Simulation der in 24 Partitionen aufgeteilten Schaltung verwendet wurden und kein dynamischer Lastausgleich durchgeführt wurde. Die verschiedenen Reihen des Diagramms nach hinten stellen verschiedene Lastszenarien dar. Die vorderste Reihe repräsentiert ein homogenes, exklusiv nutzbares Netzwerk aus Alpha-Arbeitsplatzrechnern. Für die Szenarien 'Last 1' bis 'Last 4' wurde dieses Netzwerk mit bis zu neun synthetischen Benutzerprozessen pro Simulator zusätzlich belastet. Durch die Verwendung künstlicher statt realer Benutzerprozesse wurde eine exakte Reproduzierbarkeit der Lastszenarien erreicht. Die Laufzeit und Verteilung dieser störenden Prozesse auf die einzelnen Rechnerknoten orientierte sich dabei an praktischen Beobachtungen.

Eine Ausnahme stellt Spalte 1/1 dar. In ihr ist in der vordersten Reihe als Referenz die Programmlaufzeit der sequentiellen Simulation aufgetragen. Da sich die verteilt definierten Lastszenarien nicht auf einen einzelnen Rechner übertragen lassen, wurde aus den Lastszenarien eine durchschnittliche CPU-Last errechnet und die sequentielle Simulationsdauer damit multipliziert. Die weißen Balken in Spalte 1/1 stellen deshalb als einzige keine tatsächlich durchgeführten Experimente sondern rechnerisch bestimmte Werte dar.

Die zweite Spalte (6/6) zeigt die Rechendauer bei herkömmlicher Partitionierung ohne dynamischen Lastausgleich. Dies entspricht somit herkömmlichen, optimistisch synchronisierten, ver-

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

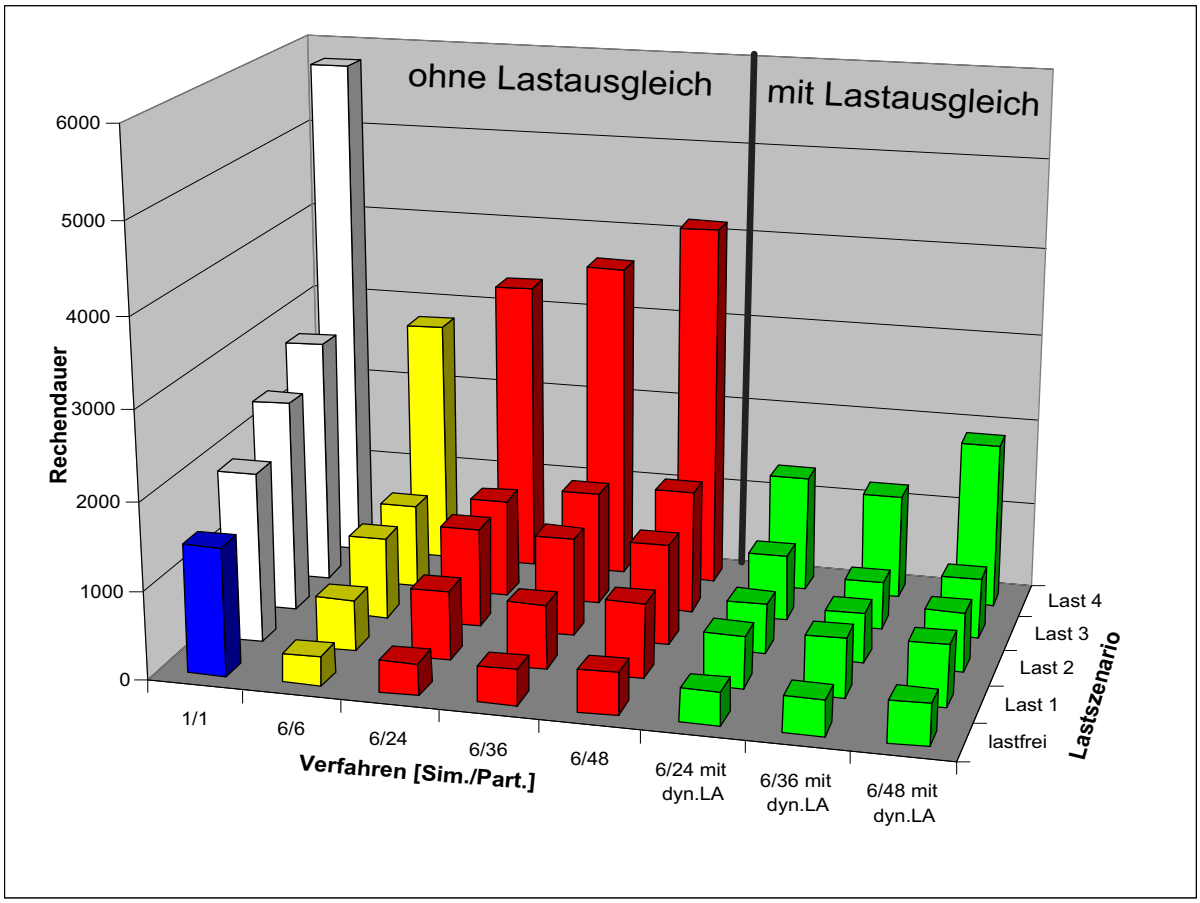

Abbildung 7: Programmlaufzeiten für Logikschaltung s35932

teilten Simulatoren. Es ist zu erkennen, daß durch die Verwendung eines belasteten Netzwerks der Zeitvorteil gegenüber der sequentiellen Simulation deutlich sinkt. Bei den Spalten 6/24 bis 6/48 kommt nun die Multipartitionierung ohne dynamischen Lastausgleich zum Einsatz. Der Mehraufwand bei der Partitionsverwaltung und beim Nachrichtenaustausch zwischen den Simulatoren schlägt in einer Erhöhung der Rechendauer umso stärker zu Buche, je mehr Partitionen eingesetzt werden. Für die drei rechten Spalten wurde schließlich der dynamische Lastausgleich aktiviert. Die Simulationszeiten sind in fast allen Fällen kürzer als bei der herkömmlichen Parallelsimulation ohne dynamischen Lastausgleich in Spalte 6/6.

Die Gesamtheit aller durchgeführten Versuche und deren Ergebnisse zeigen, daß das vorgestellte Verfahren zum dynamischen Lastausgleich verteilter, ereignisgesteuerter Simulation die Simulationsdauer bei nichtidealen Verhältnissen um bis zu 60% verkürzt. Schlüssel hierzu sind das angepaßte Maß zur Lasterfassung *virtueller Zeitfortschritt*, die schnelle Migrationstechnik durch *Multipartitionierung* und die Rentabilitätsabschätzung durch die Vorhersage des *Break-Even-Punktes*.

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999.
Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur
Veröffentlichung angenommen.

Literatur

[AT96] H. Avril und C. Tropper: *The Dynamic Load Balancing of Clustered Time Warp for Logic Simulation*, in: *ACM/SCS/IEEE Workshop on Parallel and Distributed Simulation (PADS)*, Seiten 20–27. IEEE Computer Society Press, Mai 1996.

[Bau94] H. Bauer: *Verteilte diskrete Simulation komplexer Systeme*, Dissertation, Lehrstuhl für Rechnergestütztes Entwerfen, Technische Universität München, 1994.

[BBK89] F. Brglez, D. Bryan und K. Kozminski: *Combinational Profiles of Sequential Benchmark Circuits*, in: *IEEE International Symposium on Circuits and Systems (ISCAS)*, Band 3, Seiten 1929–1934, Mai 1989.

[BD97] A. Boukerche und S. K. Das: *Dynamic Load Balancing Strategies for Conservative Parallel Simulations*, in: *ACM/SCS/IEEE Workshop on Parallel and Distributed Simulation (PADS)*, Seiten 20–28, Lockenhaus, Austria, Juni 1997, IEEE Computer Society Press.

[BGG+98] A. Bode, A. Ganz, C. Gold, S. Petri, N. Reimer, B. Schiemann und T. Schnekenburger: *Anwendungsbezogene Lastverteilung ALV'98*, Technischer Bericht 342/01/98 A, Sonderforschungsbereich 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen, München, Februar 1998.

[CCT97] M. Cermele, M. Colajanni und S. Tucci: *Check-Load Interval Analysis for Balancing Distributed SPMD Applications*, in: *Int. Conf. on Parallel and Distributed Processing Techniques and Applications PDPTA*, Band 1, Seiten 432–441, Las Vegas, Nevada, USA, Juni 1997.

[CF96] C. D. Carothers und R. M. Fujimoto: *Background Execution of Time Warp Programs*, in: *ACM/SCS/IEEE Workshop on Parallel and Distributed Simulation (PADS)*, Seiten 12–19, Philadelphia, Pennsylvania, Mai 1996, IEEE Computer Society Press.

[FSP97] J. Fleischmann, R. Schlagenhaft und M. Peller: *Objektorientierte Logiksimulation nach dem VITAL Standard*, in: *ITG/GI/GMM Workshop Hardwarebeschreibungssprachen und Modellierungsparadigmen*, Seiten 155–162, Holzhau, Februar 1997.

[RJ90] P. L. Reiher und D. Jefferson: *Virtual Time Based Dynamic Load Management in the Time Warp Operating System*, in: *ACM/SCS/IEEE Workshop on Parallel and Distributed Simulation (PADS)*, Seiten 103–111, 1990.

[Sch97] R. Schlagenhaft: *MPSIM - Parallel Event Driven Simulation of Logic Circuits by Time Warp*, in: T. Schnekenburger und G. Stellner, Herausgeber, *Dynamic Load Distribution for Parallel Applications*, Band 24 von *Teubner-Texte zur Informatik*, Kapitel 4.4, Seiten 160–170, Teubner Verlag, Stuttgart, September 1997.

[Spo94] C. Sporrer: *Verfahren zur Schaltungspartitionierung für die parallele Logiksimulation*, Dissertation, Technische Universität München, 1994.

GI-Workshop Verteilte Simulation und parallele Prozesse, Magdeburg, 1999. Arbeitsgemeinschaft Simulation in der Gesellschaft für Informatik. Zur Veröffentlichung angenommen.

[SRSB95] R. Schlagenhaft, M. K. Ruhwandl, C. Sporrer und H. Bauer: *Dynamic Load Balancing of a Multi-Cluster Simulator on a Network of Workstations*, in: *ACM/SCS/IEEE Workshop on Parallel and Distributed Simulation (PADS)*, Seiten 175–180, Lake Placid, New York, Juni 1995.

[WT98] J. Watts und S. Taylor: *A Practical Approach to Dynamic Load Balancing*, IEEE Transactions on Parallel and Distributed Systems, 9 (3), 235–248, März 1998.