Identifying and Harnessing Concurrency for Parallel and Distributed Network Simulation

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften/
Doktors der Naturwissenschaften

von der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

vorgelegte

Dissertation

von

Philipp Andelfinger

aus Weingarten

Tag der mündlichen Prüfung: 10. Februar 2016

Erster Gutachter: Prof. Dr. rer. nat. Hannes Hartenstein
Karlsruhe Institute of Technology (KIT)

Zweiter Gutachter: Dr. Kalyan Perumalla
Oak Ridge National Laboratory (ORNL)

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 08. Dezember 2015

Zusammenfassung

Da Netzwerke von Computern inhärent parallele Systeme darstellen, können Simulationen von Computernetzen häufig durch parallele und verteilte Ausführung auf mehreren Prozessoren substantiell beschleunigt werden. Simulationsmodelle von Computernetzen können sich jedoch grundlegend in ihrer Eignung für eine Parallelisierung unterscheiden. Obwohl sich bereits eine Vielzahl von vorausgehenden Arbeiten mit der effizienten parallelen Ausführung von Netzwerkmodellen befasst hat, besteht ein Mangel an Verfahren, die Entscheidungen bezüglich der Parallelisierung von Netzwerkmodellen und die Wahl geeigneter Hardware-Plattformen, Simulatorarchitekturen und Synchronisationsansätze unterstützen. Zusätzlich hat die breite Verfügbarkeit kostengünstiger Manycore-Hardware das Spektrum möglicher Realisierungen von Simulatoren über die Möglichkeiten traditioneller CPU-basierter Ansätze hinaus erweitert.

Diese Dissertation betrachtet die effiziente Ausführung von Netzwerksimulationen aus zwei Perspektiven: zunächst werden Evaluationsmethoden vorgeschlagen, die eine Abschätzung des Parallelisierungspotentials von Netzwerkmodellen erlauben. Im Anschluss werden Ansätze vorgestellt, die das identifizierte Parallelisierungspotential mittels moderner Manycore-Hardware effizient ausnutzen.

Identifizierung von Nebenläufigkeit: wir stellen einen analytischen Ansatz vor, der die durchschnittliche Anzahl an Recheneinheiten abschätzt, die von einem idealisierten parallelen oder verteilten Simulationslauf eines gegebenen Netzwerkmodells ausgelastet werden können. Die Abschätzung erfolgt auf Basis von Modellwissen und einfachen Netzwerkstatistiken, die in sequentiellen Simulationsläufen gewonnen werden. Da das vorgestellte Verfahren nicht auf automatisierten Methoden wie der “Critical Path Analysis” beruht, erlauben die Schätzungen ein Verständnis von Zusammenhängen zwischen den Kommunikationsmustern im simulierten Netzwerk und der resultierenden Nebenläufigkeit des Netzwerkmodells. Ein Verständnis dieser Zusammenhänge kann Modelloptimierungen und die Auswahl geeigneter Simulatorarchitekturen unterstützen. Das Verfahren basiert auf einer näherungsweisen Bestimmung des Simulationsfortschritts unter dem Synchronisationsalgorithmus YAWNS. Wir legen den Zusammenhang zwischen Critical Path Analysis und YAWNS dar und beweisen die Gültigkeit unseres Ansatzes sowie existierender Arbeiten, die unsere Annahmen teilen. Eine Evaluation der Akkuratheit konkreter Schätzungen wird anhand einer Anwendung des Schätzverfahrens auf Implementierungen dreier Netzwerkmodelle in bekannten Netzwerksimulatoren durchgeführt.

Um zusätzlich zu den Eigenschaften des untersuchten Netzwerkmodells auch die zur Durchführung verwendete Hardware und den Synchronisationsansatz zu berücksichtigen, stellen wir ein Werkzeug vor, dass die Laufzeit paralleler und verteilter Simulationen auf Basis sequentieller Simulationen und Hardware-Messungen prädiziert. Das Werkzeug führt eine Simulation einer geplanten parallelen oder verteilten Simulation durch ("Simulation zweiter Ordnung"). Wir zeigen, dass im Falle von Netzwerkmodellen mit nicht-trivialen Berechnungen pro simulierter Nachricht eine angemessen akkurate Prädiktion erreicht wird.

Nutzung von Nebenläufigkeit: traditionelle Ansätze der parallelen und verteilten Simulation greifen zur Ausführung des Netzwerkmodells auf einen Verbund von CPUs zurück. In einigen existierenden Arbeiten ergaben sich für Modelle von Peer-to-Peer-Netzwerken durch Parallelisierung der Simulation nur geringe Laufzeitverbesserungen. Wir analysieren und vergleichen zwei Partitionierungsstrategien für Modelle von Netzwerken, die auf dem Protokoll Kademlia basieren. Ein Beispiel für ein solches Netzwerk ist die BitTorrent DHT, eines der größten öffentlichen Peer-to-Peer-Netzwerke. Mittels einer Partitionierung der Simulation basierend auf der logischen Topologie des Netzwerkes erreichen wir eine Beschleunigung der Simulation um einen Faktor von 6,0 im Vergleich mit einer sequentiellen Ausführung sowie eine nahezu lineare Reduktion des Speicherbedarfs pro Rechenknoten.

Da die mittels einer CPU-basierten parallelen und verteilten Ausführung erreichte Beschleunigung einer Simulation die hierzu erforderlichen Hardware-Ressourcen nicht in jedem Falle rechtfertigen kann, untersuchen wir die Ausführung von Netzwerksimulationen auf Grafikprozessoren (Graphics Processing Units, GPUs). Heutige GPUs sind in der Lage, allgemeine Berechnungen auf hunderten oder tausenden paralleler Recheneinheiten durchzuführen. Eine im Arbeitsplatzrechner eines Forschers vorhandene kostengünstige GPU kann dazu dienen, die Wartezeit zwischen Änderungen an einem Netzwerkmodell und dem Erhalten von Simulationsergebnissen zu verringern.

Zuächst vergleichen und evaluieren wir Architekturen zur GPU-Beschleunigung rechenaufwändiger Schritte einer CPU-basierten detaillierten Simulation drahtloser Netzwerke. Obwohl eine einzelne simulierte Nachrichtenübertragung bereits Möglichkeiten zur parallelen Verarbeitung bietet, zeigen unsere Messungen, dass eine signifikante Beschleunigung der Simulation es erforderlich macht, mehrere Nachrichtenübertragungen aggregiert zu betrachten. Um die Korrektheit der Simulation zu gewährleisten, muss hierbei muss die Möglichkeit von Interaktionen zwischen mehreren Sendevorgängen berücksichtigt werden. Unsere Ergebnisse demonstrieren daher, dass bereits im betrachteten Fall einer GPU-Beschleunigung einzelner Schritte einer durch eine CPU verwalteten Simulation Synchronisationsmechanismen aus dem Feld der parallelen und verteilten Simulationen erforderlich sind.

Schließlich stellen wir einen rein GPU-basierten Simulationsansatz vor, in welchem neben dem Netzwerkmodell auch die gesamte Simulationslogik auf einer GPU ausgeführt wird. Da auf eine Interaktion zwischen einer CPU des Host-Systems und der GPU weitestgehend verzichtet wird, eignet sich der rein GPU-basierte Ansatz auch im Falle von Netzwerkmodellen, deren Simulationsereignisse jeweils nur ge-

ringfügige Berechnungen erfordern. Im Gegensatz zu existierenden Arbeiten werden in unserem Ansatz die simulierten Knoten zu Gruppen zusammengefasst, welche jeweils gemeinsam betrachtet werden. Diese Aggregation erlaubt es, die Auslastung der Rechneneinheiten der GPU gegenüber dem Verwaltungsaufwand der Simulation abzuwägen, indem der Grad an Aggregation basierend auf Messungen der Simulationsleistung dynamisch zur Laufzeit angepasst wird. Eine Leistungsbewertung unserer Implementierung des Ansatzes anhand eines Modells Kademlia-basierter Netzwerke und des Benchmark-Modells PHOLD zeigt eine Beschleunigung der Simulationen um einen Faktor von bis zu $19,5$ bzw. $27,5$ im Vergleich mit einer sequentiellen CPU-basierten Ausführung, sowie eine Ereignisrate von bis zu $6,8 * 10^6$ bzw. $39,3 * 10^6$ Ereignissen pro Sekunde auf einer einzelnen GPU.

Abstract

Since computer networks are inherently parallel systems, simulations of computer networks can in many cases be accelerated substantially through parallel and distributed execution on a set of interconnected processors. Still, simulation models of computer networks vary significantly in their parallelization potentials. Although an enormous number of works consider the efficient parallel execution of specific network models, there is still a lack of guidelines that help in decisions on parallelization and in the selection of hardware platforms, simulator architectures and synchronization approaches that enable an efficient execution. Further, the advent of commodity many-core devices has broadened the range of possible simulator realizations beyond the possibilities of traditional CPU-based approaches.

This dissertation addresses the efficient execution of simulation models of computer networks from two perspectives: we propose evaluation methods to determine a model's parallelization potential and provide simulator realizations that efficiently exploit the identified potentials using modern many-core hardware.

Identifying Concurrency: we propose an analytical approach to estimate the concurrency of network models, i.e., the number of processors that can be occupied in an idealized parallel and distributed simulation run based only on model knowledge and simple network statistics from sequential simulation runs. By not relying on an automated "black-box" method such as critical path analysis, our estimation approach exposes the relationships between the communication patterns in a simulated network and the resulting concurrency of the simulation. Insights into these relationships may guide model optimizations and the selection of suitable simulator architectures. Our estimations approximate the progress of simulations performed using the well-known synchronization algorithm YAWNS. After clarifying the relationship between critical path analysis and YAWNS, we provide a proof that shows the validity of the general approach and that substantiates the results of previous works that share our assumptions. Empirical results on the example of three network models implemented in popular simulators demonstrate that the estimations are sufficiently accurate to evaluate the models' parallelization potential, while avoiding an automated "black-box" analysis of sequential simulation runs.

In order to take into account both the properties of the considered network model as well as the execution hardware and synchronization approach, we present a tool that predicts the runtime of parallel and distributed simulations on the basis of sequential simulation runs and hardware measurements. The tool performs a simulation of an envisioned parallel and distributed simulation ("second-order simulation"). We show

that reasonably accurate runtime predictions are achieved for distributed simulation runs in the case of network models where simulated messages require non-trivial amounts of computation.

Harnessing Concurrency: traditionally, parallel and distributed simulations rely on interconnected CPUs for execution of the simulation model. In some previous works, models of peer-to-peer networks have been reported to benefit only to a small degree from parallelization in CPU-based execution environments. We analyze and compare two partitioning strategies for models of Kademlia-based networks such as the BitTorrent DHT, one of the largest public peer-to-peer networks. When applying a partitioning strategy based on the logical topology of the network, we achieve a simulation speedup of 6.0 compared to a sequential execution and a near-linear reduction of the memory requirements per execution node.

Since high-performance CPU-based parallel and distributed simulations can consume enormous amounts of hardware resources that may not be justified by the achieved runtime reductions, we consider the acceleration of network simulations using graphics processing units (GPUs). GPUs have evolved to support general-purpose computations on hundreds or thousands of parallel processing elements. An inexpensive GPU in a researcher's workstation can be used to shorten the feedback loop between changes to the network model and the retrieval of the corresponding simulation results.

We compare and evaluate architectures for a GPU-based coprocessing of detailed wireless network simulations. Although each simulated transmission already provides opportunities for parallel processing, we show that significant runtime reductions require an aggregated consideration of multiple transmissions in parallel. Since, in order to maintain correctness, the potential for interactions between multiple transmissions must be considered, the results show that even a simple GPU-based coprocessing requires synchronization mechanisms from the field of parallel and distributed simulation.

Subsequently, we propose a fully GPU-based simulation approach that executes all simulation logic as well as the network model on a GPU. Due to avoiding most interaction between a host CPU and the GPU, the fully GPU-based approach is applicable to network models were individual events require only small amounts of computation. Contrary to existing works, our approach aggregates sets of simulated nodes that are considered jointly. The aggregation enables the exploitation of a tradeoff between the utilization of the GPU's processing elements and simulation overheads by dynamically adapting the degree of aggregation according to performance measurements at simulation runtime. We conduct a performance evaluation of our implementation on the example of a model of Kademlia-based networks and the well-known PHOLD benchmark model. Using a single commodity GPU, we achieve a speedup of up to 19.5 and event rates up to 6.8×10^6 for the model of Kademlia-based networks. A speedup of up to 27.5 and event rates of up to 39.3×10^6 events per second of wall-clock time are achieved in the case of the PHOLD model.

Contents

List of Figures

List of Tables