Vereinfachte Nutzung von Cluster-, Grid- und Cloud-Computing für komplexe Simulationaufgaben im Systementwurf

André Schneider, Manfred Dietrich
Fraunhofer-Institut für Integrierte Schaltungen IIS, Institutsteil Entwurfsautomatisierung EAS
Zeunerstraße 38, 01069 Dresden, Deutschland
andre.schneider@eas.iis.fraunhofer.de, manfred.dietrich@eas.iis.fraunhofer.de

Kurzfassung

Für den modellbasierten Systementwurf spielt die Simulation eine zentrale Rolle. Allerdings stieß man in der Vergangenheit bei der numerischen Simulation von komplexen Systemen oder Modellen mit hohem Detaillierungsgrad schnell an Grenzen: Entweder war der Speicherbedarf zu groß oder die Simulationszeit zu lang. Diese Situation ändert sich jedoch gegenwärtig dramatisch. Mit der Verfügbarkeit von Multi- und Many-Core-Architekturen, Compute-Clustern sowie Grid- und Cloud-Computing werden in naher Zukunft dem Systementwerfer enorme Rechenressourcen zur Verfügung stehen, die einerseits die Simulation sehr komplexer Modelle erlauben und andererseits gestatten, ein Modell in vielen Varianten zu simulieren. Der Beitrag stellt einen Ansatz und eine Software für die Variantensimulation vor, bei der ausgehend von einer einfachen Problembeschreibung viele Hundert bis Millionen Einzelsimulationen auf Cluster- bzw. Grid-Rechnern ausgeführt werden und die dabei entstehenden Ergebnisse für den Endanwender komprimiert und aufbereitet werden. Die vielfältigen Möglichkeiten, die sich mit der Variantensimulation künftig ergeben, werden diskutiert und an Beispielen illustriert.

1 Motivation

Modellbasierte und simulationsgestützte Verfahren sind seit vielen Jahren im Systementwurf etabliert. Die Mikroelektronik spielte hier ohne Zweifel eine Vorreiterrolle: Seit über 40 Jahren werden unter anderem elektrische Netzwerke mit Hilfe von SPICE bzw. SPICE-ähnlichen Simulatoren analysiert. Hinzugekommen ist später neben weiteren Bereichen die Mikrosystemtechnik, die den Aspekt domänenübergreifender Modellierung und Simulation stärker in den Vordergrund gerückt und den Bedarf an angepassten Modellierungssprachen verstärkt hat. Inzwischen gibt es kaum ein Gebiet aus den Technik-, Natur- und Sozialwissenschaften sowie der Medizin, in dem die Simulation nicht bereits Fuß gefasst hätte.

Während sich mit jedem neuen Einsatzgebiet die Vielfalt der Modellierungssprachen und Simulationswerkzeuge vergrößert hat, ist die prinzipielle Arbeitsweise nahezu unverändert geblieben. Ein typisches Szenario sieht wie folgt aus:

- Mit Hilfe einer Beschreibungssprache oder eines graphischen Eingabewerkzeugs wird ein Modell erstellt.
- Für Parameter, Anfangs- und Randbedingungen werden Werte festgelegt.
- Mit Hilfe eines Simulationsprogramms wird das Modell für einen definierten Zeit- oder Frequenzbereich (o.ä.) simuliert.
- Die Simulationsergebnisse werden geeignet visualisiert und können ausgewertet werden.

Hierbei ist unerheblich,

- ob das Modell komplett selbst erstellt wird oder aus Bibliothekskomponenten aufgebaut wird,
- ob graphisch oder textbasiert modelliert wird,
- ob frei wählbare Größen von Hand oder mehr oder weniger automatisch definiert werden,
- welche Simulationsalgorithmen verwendet werden oder
- ob die Simulationsergebnisse in Form von Signalverläufen (z.B. Spannung über der Zeit oder Dämpung über der Frequenz), Kennwerten (z.B. Verlustleistung

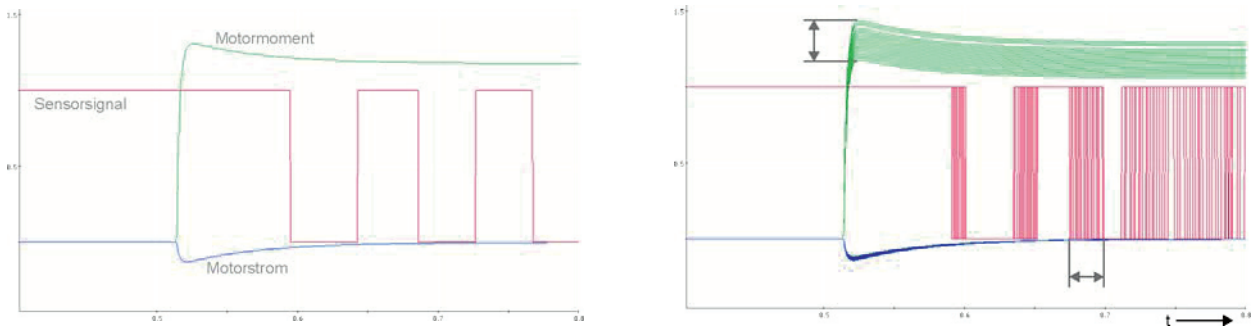

Bild 1. Schaltvorgang beim Fensterheber – links: konventionelle Einzelsimulation bei nominaler Bordspannung, rechts: Variantensimulation unter Berücksichtigung von Bordspannungsschwankungen.

oder Resonanzfrequenz), Feldern (z.B. Temperatur im 3D-Raum) oder beliebigen anderen Strukturen ausgegeben werden.

Charakteristisch für die beschriebene Vorgehensweise ist, dass jede Simulation einzeln betrachtet wird. Entsprechend fokussiert die Konfiguration auf einzelne Parameterwertkombinationen und als Ergebnis ergeben sich Aussagen zum Systemverhalten in einer konkreten Situation. Diese Herangehensweise reicht jedoch oft nicht aus. Stellvertretend seien vier Gründe genannt:

- Im Systementwurf können insbesondere in frühen Phasen die Parameter und Randbedingungen für ein Modell nicht exakt angegeben werden.
- In der Systemfertigung treten Prozessschwankungen auf und bei allen Teilkomponenten müssen Toleranzen berücksichtigt werden.
- Die Simulation ist im Vergleich zu einer Prototypfertigung kostengünstig. Sie erlaubt in der Entwurfsphase das Ausprobieren vieler Realisierungsmöglichkeiten und unterstützt damit den kreativen Prozess, einen optimalen und robusten Entwurf zu finden.
- Unter realen Einsatzbedinungen können Umgebungseinflüsse, Alterung oder andere Faktoren Abweichungen vom Nominalverhalten bewirken. Als Beispiel hierfür wurde das Modell eines Fensterhebers einer PKW-Fahrzeugtür für unterschiedliche Bordspannungen simuliert (vgl. Bild 1).

Grundsätzlich verbessern lässt sich die Situation im Systementwurf nur, wenn die Variabilität und damit die Ausnutzung der initial gegebenen Freiheitsgrade sowie die sich ändernden Umgebungsbedingungen von Anfang an in den Entwurfsprozess mit einbezogen und so integraler Bestandteil bei simulationsgestützten Verfahren werden. Der Preis hierfür ist allerdings hoch: Der Rechenaufwand und oft auch der Speicherbedarf steigt um Größenordnungen und lässt sich mit den innerhalb einer Firma oder Institution lokal verfügbaren Ressourcen in der Regel nicht abdecken.

Mit der Verfügbarkeit von Multi- und Many-Core-Architekturen, Compute-Clustern sowie Grid- und Cloud-Computing werden in naher Zukunft dem Systementwerfer Rechenressourcen zur Verfügung stehen, mit denen sich der geforderte Bedarf erstmalig kostengünstig abdecken lassen wird. Entscheidend ist hierbei, dass der Gewinn nicht darin besteht, die Compute-Power dauerhaft um den Faktor zwei, drei oder fünf zu verbessern, sondern um Größenordnungen und das aber nur auf Anforderung (on-demand) für die Zeit, für die die Rechenleistung zur Durchführung von Simulationen tatsächlich benötigt wird.

Cluster-, Grid- und Cloud-Computing erlauben in Kombination einerseits die Simulation sehr komplexer Modelle und andererseits gestatten die Technologien, ein Modell in vielen Varianten zu simulieren [6]. Der vorliegende Beitrag stellt einen Ansatz und eine Software für die *Variantensimulation* vor, bei der ausgehend von einer einfachen Problembeschreibung viele Hundert bis Millionen Einzelsimulationen auf Cluster- bzw. Grid-Rechnern ausgeführt werden und die dabei entstehenden Ergebnisse für den Endanwender komprimiert und aufbereitet werden.

Der Schwerpunkt der Arbeit besteht darin, mit *GridWorker* die gegenwärtig existierende Lücke zwischen den etablierten Simulations- und Entwurfswerkzeugen auf der einen und den sich stark verbreitenden Grid- und Cloud-Infrastrukturen auf der anderen Seite zu schließen. Die im EDA-Umfeld etablierten System- und Verhaltenssimulatoren nutzen bisher nahezu ausschließlich die lokale Hardware für ihre Berechnungen. Zwar ist hier in den letzten Jahren ein Trend zur Unterstützung von Multi-Core- und GPU-Architekturen sichtbar, Grid- oder Cloud-Ressourcen können jedoch in der Regel nicht einbezogen werden. Betrachtet man umgekehrt die zahlreich verfügbaren Middleware-Lösungen für das Grid Computing, stellt man fest, dass diese zwar wesentliche Basisfunktionen wie Authentifizierung/Autorisierung, Job-Verteilung und -Scheduling, Monitoring und Abrechnung unterstützen, aber nur ungenügende Schnittstellen und Funktionen für einen unmittelbaren Einsatz im Systementwurf bieten. Im akademischen Umfeld wie beispielsweise in der Hochenergie- und Astrophysik wird diese Lücke zwischen Endanwendung und Grid-Middleware häufig über Skripte, die die Endanwender selbst implementieren, geschlossen und die oft als Open Source vorliegenden Simulationscodes werden direkt an spezielle Grid-Infrastrukturen angebunden. Dies ist jedoch kein akzeptables Vorgehen für einen Systementwerfer aus der Industrie, der im Rahmen vorgegebener Design Flows kommerzielle Simulationswerkzeuge einsetzt und nach einer flexibel und robust funktionierenden Knopfdrucklösung verlangt.

2 *GridWorker*

Auf dem Gebiet der parallelen Verarbeitung sehr großer Datenmengen, wie sie bei Google, Facebook und Yahoo vorliegen, hat sich in den letzten Jahren der in [2] beschriebene *MapReduce*-Ansatz etabliert. Im Vergleich zu vielen Middleware-Lösungen und Frameworks aus dem Grid- und Cluster-Computing-Umfeld (Globus Toolkit, gLite, UNICORE, ARC, Grid Engine, Condor, ...) bietet das *MapReduce*-Paradigma einen entscheidenden Vorteil für den Endanwender: Er muss sich auf die Implementierung von genau zwei Funktionen konzentrieren. Die problem- oder anwendungsspezifische Verarbeitungslogik wird in den beiden Funktionen *map()* und *reduce()* beschrieben. Alles andere wird vom *MapReduce*-Framework (zum Beispiel Apache Hadoop [1]) organisiert.

Mit dem *GridWorker*-Ansatz wird für den Anwendungsbereich der massiv parallelen numerischen Simulationen ein zu *MapReduce* analoges Ziel verfolgt. Einerseits soll *GridWorker* dem Systementwerfer gestatten, sich auf die Ausführung der Simulation und die Auswertung der Ergebnisse zu konzentrieren, andererseits soll der Ansatz hinreichend allgemein sein, so dass sich ganze Problemklassen aus dem Systementwurf darauf abbilden lassen.

Der *GridWorker*-Ansatz ist in Bild 2 dargestellt. Entscheidend ist hier die einfache Schnittstelle zwischen dem Anwender und der Infrastruktur. Mit Hilfer einfacher ASCII-Dateien [5] wird das letztlich parallel abzuarbeitende Problem beschrieben. Die *GridWorker*-Komponenten *Splitter*, *Mapper* und *Reducer* übernehmen in generischer

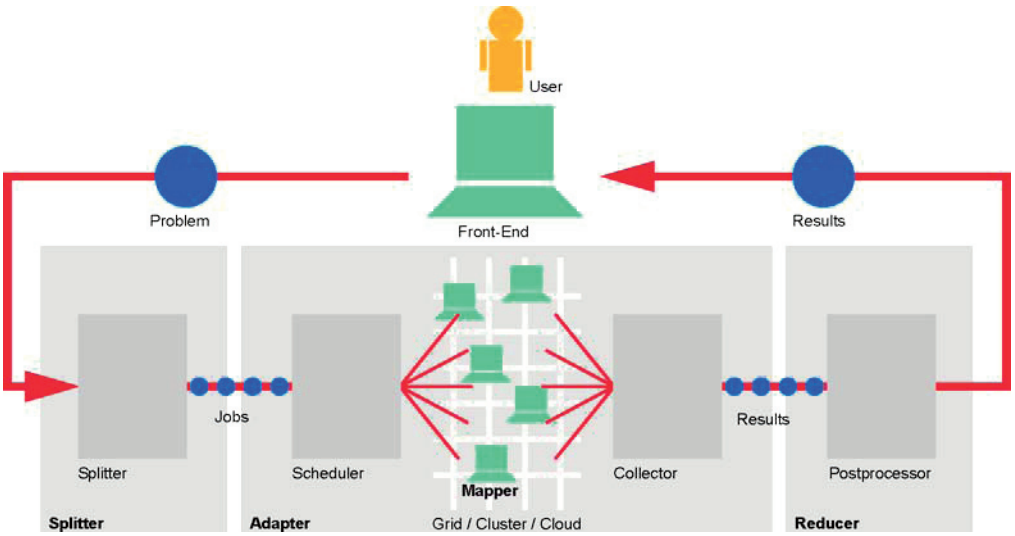

Bild 2. Architektur der *GridWorker*-Software.

Form die Problemzerlegung, die Problembearbeitung und die Ergebnisaufbereitung und -zusammenfassung.

Der *Splitter* besitzt die Aufgabe, ein komplexes Simulationsproblem in Teilaufgaben (*Tasks*) zu zerlegen. Eine *Task* entspricht hierbei genau einer Simulation. Da *GridWorker* auf die Variantensimulation fokusiert, besteht die Problemzerlegung in einer Aufteilung von Parameterwertemengen und ist damit generisch und automatisiert lösbar. Um den Verwaltungsoverhead zu minimieren, ist es oft sinnvoll, mehrere *Tasks* zu einem *Job* zusammenzufassen. Die Anzahl der *Tasks* innerhalb eines *Jobs* bestimmt dessen „Gewicht“ (Bedarf an Speicher und Simulationszeit).

Der *Adapter* erhält vom *Splitter* die *Jobs*, die er an eine beim Anwender verfügbare Grid- oder Cluster-Middleware übergibt. Für *GridWorker* stehen derzeit *Adapter* für folgende Middlewares zur Verfügung: GridWay, Globus Toolkit, Sun Grid Engine (SGE), Portable Batch System (PBS), Load Sharing Facility (LSF) und DRMAA [3][4]. Die Jobs werden in der Regel als Array-Job weitergereicht. Bei der Job-Ausführung werden auf dem Zielknoten alle Tasks beziehungsweise Einzelsimulationen über einen ThreadPool abgearbeitet.

Der *Mapper* übernimmt gemäß dem oben genannten *MapReduce*-Paradigma die Abbildung eines Simulationsmodells (Modellinstanz mit konkreter Parameterwertkonfiguration) auf ein Simulationsergebnis. Dabei ruft der Mapper den dazu benötigten Simulator auf. Das kann ein Skript, ein eigener Simulationscode, ein Open-Source-Simulator wie SPICE oder aber auch ein kommerzielles Simulationsprogramm wie Cadence Spectre oder MATLAB Simulink sein. Nach erfolgter Simulation bereitet der *Mapper* die Ergebnisse so auf, dass sie für eine weitere Verarbeitung und Zusammenfassung geeignet sind.

Der *Reducer* hat zum Schluss die Aufgabe, alle Simulationsergebnisse zu einem Gesamtergebnis zusammenzufassen. Dabei ist es möglich, ein umfangreiches Postprocessing durchzuführen und die für den Systementwerfer gewünschten Kenndaten zu extrahieren, oder aber alle Einzelergebnisse lediglich unverändert in einem Container zusammenzufassen.

3 Beispiel: Filter-Simulation

An einem einfachen Beispiel soll die Funktionsweise von *GridWorker* illustriert werden. Für eine Filterschaltung soll der Einfluss der Widerstands- und Kapazitätsparameter untersucht werden. Hierfür bietet sich eine Variantensimulation an. Um diese mit GridWorker durchführen zu können, muss der Systementwerfer folgende Dateien bereitstellen: ein SPICE-Modell mit der RC-Netzwerkbeschreibung *(filter.cir)*, eine Parameterspezifikation mit allen zu simulierenden R-C-Parameterwertkombinationen *(parameters)* sowie eine *GridWorker*-Konfiguration *(configuration)*, in der unter anderem der *Mapper* und der *Adapter* festgelegt wird.

Die SPICE-Netzliste *filter.cir* unterscheidet sich von einer regulärenNetzliste dadurch, dass anstelle der Parameterwerte Platzhalter wie *@R1@* eingetragen werden, die später durch GridWorker unmittelbar vor der Simulation durch konkrete Parameterwerte ersetzt werden.

```
* filter.cir
v0 6 0  dc 0 ac 2.0 0
r0 6 1  1
ra 1 0  1
r1 1 2  @R1@
c1 2 0  @C1@
r2 2 3  @R2@
c2 3 0  @C2@
r3 3 4  @R3@
c3 4 0  @C3@
r4 4 5  @R4@
c4 5 0  @C4@

.ac dec 10 1 100k
.control
set options nobreak
run
print vdb(1)
.endc
.end
```

Die Parameterspezifikation wird in kompakter Weise in der Datei *parameters* zusammengefasst. Die Syntax ist in [5] beschrieben.

```
# parameters
R1 = {0.01:0.01:1.0}
C1 = {0.1}
R2 = {0.01:0.01:0.1}
C2 = {0.015}
R3 = {0.4:0.1:0.6}
C3 = {0.05}
R4 = {0.4:0.05:0.6}
C4 = {0.004}
```

```
# configuration
adapter = Drmaa
mapper = Spice
reducer = Default
model.name = Filter
index.base = 0
split.level = 0
node.constraints = -l h_vmem=12G
```

Der Systementwerfer kann die Variantensimulation beispielsweise mit dem Kommando

```
gridworker run -n 100
```

starten. Nach einem erfolgreichen Abschluss der Variantensimulation liegen im Arbeitsverzeichnis das Simulationsergebnis *(results.jar)* und gegebenenfalls weitere Ausgabe- und Log-Daten *(outputs.jar, logs.jar)* sowie eine Datei mit Statistikinformationen zum Ressourcenverbrauch *(summary)*.

```
# summary Wed Aug 31 19:08:17 CEST 2011
jobs.memory.min=2327.31640625
jobs.memory.max=9090.90625
jobs.jvm.memory.total.mean=195.88125
tasks.time.wallclock.mean=0.07163859999999997
jobs.jvm.memory.free.mean=155.94858894348144
tasks.time.wallclock.min=0.019
jobs.jvm.memory.max.min=1820.5
tasks.time.wallclock.max=2.959
tasks.number.total=10000
jobs.jvm.memory.max.max=1820.5
jobs.number.total=20
jobs.jvm.memory.total.min=126.5625
jobs.memory.mean=5153.925
gridworker.retries=0
jobs.jvm.memory.total.max=598.4375
jobs.jvm.memory.free.min=88.30038452148438
jobs.jvm.memory.free.max=562.5065841674805
gridworker.time.total=220.516
jobs.time.wallclock.mean=16.8218
jobs.jvm.processors.available.min=4
jobs.jvm.processors.available.mean=7
jobs.jvm.processors.available.max=16
reducer.time.total=149.794
jobs.jvm.memory.max.mean=1820.5
jobs.time.wallclock.min=6.883
jobs.time.wallclock.max=43.0
```

Das Beispiel verdeutlicht, dass der Mehraufwand bei einer Grid-basierten Variantensimulation im Vergleich zur konventionellen Einzelsimulation gering ist und der Systementwerfer sich bei *GridWorker* auf die wesentlichen Spezifikationen konzentrieren kann. Insbesondere wird er nicht mit Middleware-spezifischen Job-Beschreibungen

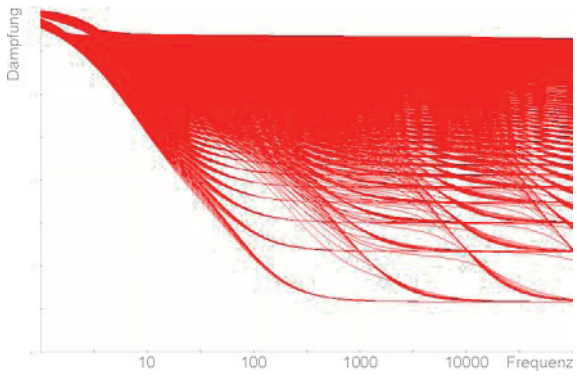

Bild 3. Ergebnisse für die Variantensimulation einer Filterschaltung

und Kommandos konfrontiert, was die Akzeptanz von *GridWorker* im EDA-Umfeld deutlich erhöht.

4 Ergebnispräsentation

Da bei einer Variantensimulation sehr viele Einzelsimulationen ausgeführt werden, entstehen entsprechend viele Simulationsergebnisse (vgl. Bild 3). Die Datenmengen betragen damit ein Vielfaches im Vergleich zur konventionellen Simulation und können schnell die Terabyte-Grenze überschreiten.

Vermieden werden kann dieser Engpass nur, wenn bereits im *Mapper* und im *Reducer* die Daten geeignet zusammengefasst und nach Möglichkeit so vorverarbeitet werden, dass der Anwender nur die signifikanten Informationen erhält.

Im Zusammenhang mit *GridWorker* wird ein Waveform-Viewer entwickelt, der die Anzeige von Kurvenscharen gestattet und die Navigation über Parameterwerten unterstützt. In Bild 4 sind die Ergebnisse einer Variantensimulation für einen Fensterheber dargestellt. Der Anwender kann sich die Signalverläufe beispielsweise für den Motorstrom und das Drehmoment über der Zeit ansehen und dabei verschiedene Wertebereiche für den Parameter Bordspannung auswählen – vgl. unterer Auswahlbereich zur Navigation über Parameterwerte.

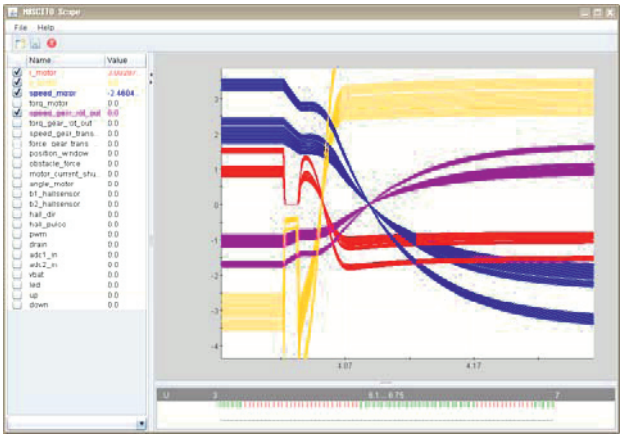

Bild 4. Ergebnisse einer Variantensimulation eines Fensterhebers für unterschiedliche Bordspannungen

5 Leistungsuntersuchung

Entscheidend für den Endanwender sind in der Regel der Zeitaufwand und die Kosten. Untersucht wurde in diesem Zusammenhang der Gesamtzeitaufwand für verschiedene Variantensimulationen, wobei das Problem jeweils in unterschiedlich viele Jobs zerlegt wurde. *GridWorker* gestattet hier eine flexible, problemangepasste Zerlegung, über die insbesondere „das Gewicht“ eines einzelnen Jobs so verändert werden kann, dass einerseits der Overhead, der bei der Verteilung entsteht, minimiert werden kann und andererseits die verfügbare Parallelität der Ressourcen effektiv ausgenutzt wird. Erste Ergebnisse sind in Bild 5 dargestellt.

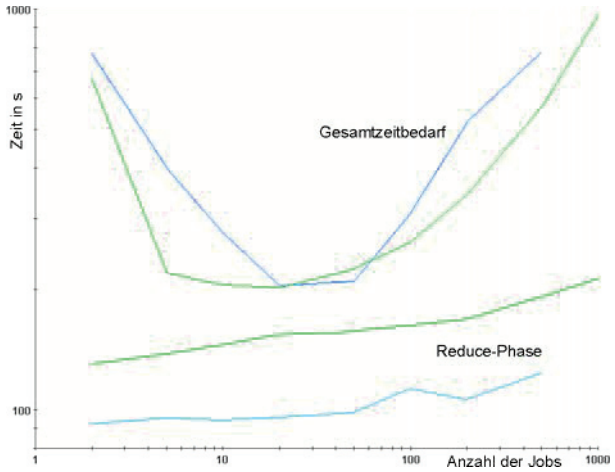

Bild 5. Gesamtzeitbedarf und Zeitbedarf für Reduce-Phase für eine Variantensimulation in Abhängigkeit von der Job-Anzahl

Bei den beiden durchgeführten Experimenten wurden jeweils 10000 Einzelsimulationen (Tasks) gestartet. Variiert wurde die Zahl der Jobs, in die das Gesamtproblem zerlegt wurde. Im unteren Diagrammbereich ist der Zeitbedarf für die Reduce-Phase dargestellt. Da diese nicht auf die parallelen Ressourcen verteilt werden kann, ist hier eine Zunahme bei steigender Job-Anzahl zu beobachten. Für den darüber abgebildeten Gesamtzeitbedarf lässt sich dagegen sehr deutlich ein optimaler Bereich bei etwa 20 bis 50 Jobs erkennen, das entspricht 500 bis 200 Tasks (Einzelsimulationen) pro Job. Eine Ursache ist hier in der Cluster-Konfiguration und der aktuellen Auslastung zu sehen: Es konnten etwa 20 Knoten parallel genutzt werden. Bei 10 Jobs ist dabei die verfügbare Parallelität nur ungenügend genutzt worden und bei 100 und mehr Jobs mussten zu viele Jobs in den Warteschlangen warten und der Verwaltungsoverhead war zu groß.

Die Ergebnisse zeigen, dass die aktuelle Implementierung von GridWorker noch genügend Optimierungspotenzial bietet. Die Leistungsuntersuchungen werden für weitere Beispiele fortgesetzt und vertieft.

6 Zusammenfassung und Ausblick

Mit der Verfügbarkeit leistungsfähiger, hochgradig paralleler Rechenressourcen wird es für den Systementwerfer zunehmend attraktiv, Systemmodelle in vielen Varianten zu simulieren. Beschrieben wurde ein Ansatz, der auf der vom Anwender konfigurierbaren Variation von Parameterwerten basiert. Dieser als Variantensimulation bezeichnete Ansatz ermöglicht eine Vielzahl simulationsgestützter Analyseverfahren nach einem einheitlichen und einfachen Grundprinzip. Mit *GridWorker* wurde eine prototypische Implementierung vorgestellt, mit der Variantensimulationen auf Cluster-, Grid- und künftig Cloud-Infrastrukturen ausgeführt werden können. Die Verbesserung der Fehlertoleranz und das Vermeiden von Ressourcenengpässen stellen die wichtigsten Herausforderungen bei der Weiterentwicklung von *GridWorker* dar.

Die vorliegende Arbeit ist im Rahmen des Verbundprojektes „OptiNum-Grid – Optimierung technischer Systeme und naturwissenschaftlicher Modelle mit Hilfe numerischer Simulationen im Grid“, das vom Bundesministerium für Bildung und Forschung unter dem Kennzeichen 01IG0901D gefördert wird, entstanden.

Literatur

[1] Apache Hadoop: http://hadoop.apache.org

[2] Dean, J.; Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Sixth Symposium on Operating Systems Design and Implementation (OSDI '04), San Francisco, California, USA, December 6-8, 2004

[3] Schneider, A.: Variantensimulation mit *GridWorker*. ASIM-Workshop „Simulation technischer Systeme und Grundlagen und Methoden in Modellbildung und Simulation“, Krefeld, 24.-25. Februar 2011

[4] Schneider, A.; Dietrich, M.: Variantensimulation im Grid. Workshop „Numerische Simulationen im D-Grid“, Göttingen, 26. November 2009.

[5] Schneider, A.; Schneider, P.; Dietrich, M.: Grid-spezifische Problembeschreibung und -aufbereitung im Systementwurf. Workshop „Grid-Technologie für den Entwurf technischer Systeme“, Dresden, 12. Oktober 2007.

[6] Schneider, A.; Schneider, P.; Schwarz, P.; Dietrich, M.: Grid-Computing – Anwendungsszenarien für den Systementwurf. 2. Workshop „Grid-Technologie für den Entwurf technischer Systeme“, Dresden, 6.-7. April 2006