Automatisierte Ressourcenbedarfsschätzung für Simulationsexperimente in der Cloud

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

Kurzfassung

Mit Hilfe von Grid und Cloud Computing eröffnen sich heute vollkommen neue Möglichkeiten, komplexe, ressourcenintensive Berechnungen auszuführen. Skalierbarkeit und Elastizität spielen hierbei eine Schlüsselrolle. Die mit den Grids und Clouds gewonnene Flexibilität hat jedoch auch einen Preis. Während sich ein Anwender bei der Nutzung der eigenen, lokal installierten Infrastruktur keine oder wenige Gedanken über die Kosten für eine CPU-Stunde machen musste, wird bei kommerziellen Cloud-Anbietern jede in Anspruch genommene Ressource wie CPU, Speicher und Netzwerkbandbreite für den Zeitraum der Nutzung konsequent abgerechnet. Im vorliegenden Beitrag wird ein Ansatz vorgestellt, der für Simulationsexperimente auf Cluster-, Grid- und Cloud-Infrastrukturen den Ressourcenbedarf vorab automatisiert abschätzt. Der Anwender bekommt auf diese Weise beispielsweise eine Vorstellung von den zu erwartenden Bearbeitungszeiten und den dafür anfallenden Kosten. Die Ressourcenabschätzung wurde für das Framework *GridWorker* implementiert und mit Anwendungsbeispielen aus dem Systementwurf evaluiert.

1 Motivation

Mit Hilfe von Grid und Cloud Computing ist es heute möglich, einem Anwender Rechenkapazitäten in Größenordnungen, die bisher Rechenzentren vorbehalten waren, binnen weniger Minuten zur Verfügung zu stellen. Insbesondere der Aspekt der Skalierbarkeit und Elastizität ermöglicht es in vielen rechenintensiven Bereichen, Experimente in völlig neuen Dimensionen durchzuführen und dabei Probleme zu lösen, die in der Vergangenheit primär an einem zu hohen Bedarf an Rechenleistung gescheitert sind. Die mit den Grids und Clouds gewonnene Flexibilität hat jedoch auch einen Preis. Während sich ein Anwender bei der Nutzung der eigenen, lokal installierten Infrastruktur keine oder wenige Gedanken über die Kosten für eine CPU-Stunde machen musste, wird bei kommerziellen Cloud-Anbietern jede in Anspruch genommene Ressource wie CPU, Speicher und Netzwerkbandbreite für den Zeitraum der Nutzung konsequent abgerechnet. Der Anwender ist also künftig stärker gefordert, wenn es um einen sinnvollen Kosten-Nutzen-Kompromiss beim Zugriff auf On-Demand-Ressourcen geht. Und selbst dann, wenn mit Flat Rates alternative Kostenmodelle angeboten werden, wird es auch beim Cloud Computing Grenzen bei der Skalierbarkeit und Elastizität geben und der Anwender wird sich Gedanken machen müssen, wie er seine Experimente so konfiguriert, dass er mit minimalem Aufwand zum gewünschten Ergebnis kommt.

Im vorliegenden Beitrag wird ein Ansatz vorgestellt, mit dem der Ressourcenbedarf für Simulationsexperimente in der Cloud vorab geschätzt und so dem Anwender geholfen werden kann, angemessene Experimentkonfigurationen zu finden und das Kosten-Nutzen-Verhältnis sinnvoll zu gestalten.

Im Rahmen der BMWi-Förderinitiative *Trusted Cloud* werden gegenwärtig im Projekt *Cloud4E – Trusted Cloud Computing for Engineering* prototypische Dienste für diverse Ingenieursdisziplinen entwickelt. Ziel ist es unter anderem, Systementwerfern die Durchführung von komplexen Simulationsexperimenten in der Cloud über generische Dienste zu ermöglichen.

Simulationen spielen beim Entwurf und der Analyse von Systemen im Engineering-Bereich und auch in naturwissenschaftlichen Disziplinen wie der Physik und der Systembiologie eine zentrale Rolle. Dabei sind es weniger die Einzelsimulationen, die die aktuellen Herausforderungen bestimmen, sondern vielmehr neue Methoden, die meist auf einer Vielzahl von Einzelsimulationen aufbauen. So sind beispielsweise Monte-Carlo-Simulationen für statistische Analysen bei Empfindlichkeits- und Robustheitsuntersuchungen Voraussetzung für einen guten Systementwurf. Ebenso gehören Parameterstudien, Worst-Case-Analysen und Optimierung zum Alltag eines Entwurfsingenieurs. In der Regel werden für eine komplexe Entwurfs- oder Analyseaufgabe Experimente definiert, die die Durchführung vieler Hundert, Tausend oder gar Millionen Einzelsimulationen erfordern.

Damit derartige ressourcenintensiven Aufgaben für den Anwender beherrschbar bleiben, wurden in den letzten Jahren Middleware-Lösungen, Frameworks und Dienste entwickelt, die die Ausführung sehr vieler Berechnungen auf Cluster-, Grid- und Cloud-Infrastrukturen organisieren und automatisieren. Mit *GridWorker* [5][9] steht beispielsweise ein Framework zur Verfügung, das die parallele, fehlertolerante Ausführung umfangreicher Variantensimulationen auf Basis des Map/Reduce-Paradigmas gestattet. Der Anwender kann mit Hilfe einer sehr einfachen Syntax kompakt und flexibel Simulationsexperimente beschreiben [11], die über *GridWorker* vollständig automatisiert

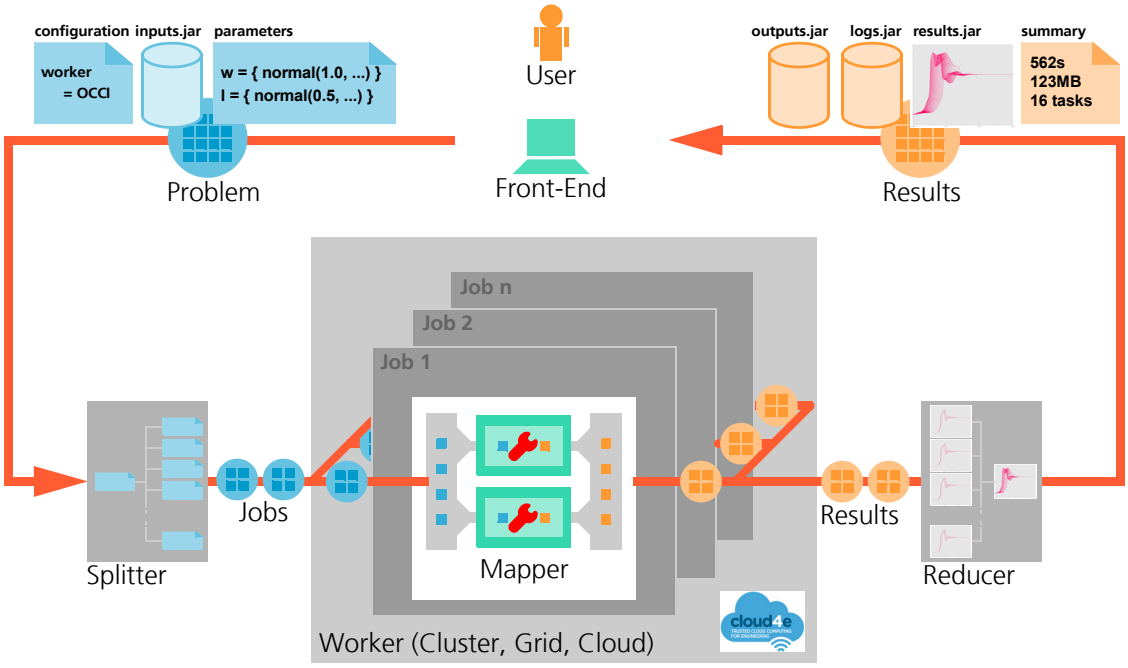

Bild 1. Variantensimulation mit GridWorker

auf den jeweils verfügbaren Compute-Ressourcen ausgeführt werden. *GridWorker* steuert hierfür über Adapter – sogenannte *Worker* – lokal im LAN erreichbare Rechner, Cluster (DRMAA, SGE, PBS, LSF, ...), Grids (Globus Toolkit, GridWay, ...) oder Clouds (OpenStack, OCCI, ...) an und organisiert die parallele Bearbeitung auf Job- und Task-Ebene vollkommen transparent für den Anwender. Die Ergebnisse werden in Containern zusammengefasst und an den Anwender direkt oder als Referenz zurückgegeben.

In Bild 2 ist die Beschreibung für eine Monte-Carlo-Simulation im Rahmen einer Fehlerdiagnose [6] illustriert.

```
# total number of tasks: 10,000
fault = { 0:1:100 }
R1 = [ normal(1.0, 0.01, 100) ]
R2 = [ normal(7.5, 0.05, 100) ]

# total number of tasks: 1,000,000
fault = { 0:1:1000 }
R1 = [ normal(1.0, 0.01, 1000) ]
R2 = [ normal(7.5, 0.05, 1000) ]
```

Bild 2. Beispiele für Parameterspezifikationen für eine kombinierte Fehler-/Monte-Carlo-Simulation

Im Beispiel müssen insgesamt 10.000 Einzelsimulationen – für 100 Fehler je 100 Monte-Carlo-Varianten – durchgeführt werden, wofür beispielsweise auf einem 128-Core-Compute-Cluster etwa 4 Minuten benötigt werden (vgl. Bild 3). Erweitert nun der Anwender sein Experiment und nimmt weitere Fehler und Monte-Carlo-Varianten hinzu, wächst die Zahl der durchzuführenden Einzelsimulationen bereits auf eine Million und der Anwender müsste vermutlich nicht nur 4 Minuten sondern etwa 6 Stunden auf das Ergebnis warten.

Kritisch ist, dass ein Anwender die Experimentkonfiguration im Hinblick auf den erforderlichen Ressourcenbedarf oft nicht hinreichend überschaut. Aufgrund der kompakten Syntax können kleine Änderungen mitunter große Auswirkungen haben, die sich dann bei der Nutzung kommerzieller Cloud-Angebote dramatisch auf die Kosten niederschlagen. Es ist daher zwingend notwendig, Anwendern Hilfswerkzeuge zur Verfügung zu stellen, mit denen sie sich beim Experimentieren jederzeit einen Überblick über die erwartete Ressourcennutzung beschaffen können. Nur wer Aufwand und Kosten im Blick behält, wird letztlich von der Flexibilität, Skalierbarkeit und Elastizität beim Experimentieren unter Nutzung von Cloud-Diensten profitieren.

2 Ansätze für Ressourcenschätzung

In der Literatur werden seit vielen Jahren unterschiedliche Ansätze für die Ressourcenschätzung beim Cloud Computing diskutiert [7]. Oft wird hierbei aus der Perspektive des Ressourcenanbieters argumentiert. Ziele sind entsprechend eine gleichmäßige oder energieeffiziente Auslastung der verfügbaren Ressourcen und die Vermeidung von Engpässen oder Leerlaufzeiten.

Aus der Perspektive des Anwenders ergeben sich häufig andere Anforderungen. Beispielsweise können die folgenden beiden Ziele relevant sein:

- Die benötigte Gesamtzeit für die Ausführung eines Simulationsexperiments sollte minimal sein.
- Die Kosten dürfen – bei Nutzung kommerzieller Ressourcen – gewisse Grenzen nicht überschreiten.

Das erste Ziel ist sehr strikt und gilt vermutlich bei den meisten Anwendern. Das zweite Ziel dagegen berücksichtigt, dass es in kreativen Bereichen wie dem Systementwurf oder generell in Forschung und Entwicklung hinreichend große Freiräume zum Experimentieren geben muss

und zu strikte Kostensenkungsregularien sinnvolles Arbeiten unmöglich machen. Hier wird eher ein Aufwand-Nutzen-Kompromiss angestrebt. Letzlich ergibt sich aus den beiden Zielen, für welche Größen quantitative Schätzungen benötigt werden. Im Kontext Cloud Computing zählen dazu primär der Gesamtzeitbedarf und die Gesamtkosten pro Simulationsexperiment.

Diese beiden Größen werden bei allen Backend-Technologien (Multi-/Many-Core-Computer, Cluster, Grid, Cloud) durch folgende Ressourcenkonfigurationsparameter bestimmt:

- Anzahl der Jobs, wobei pro CPUs, Cluster-Knoten oder VM (virtuelle Maschine) in der Regel *ein* Job läuft;
- Anzahl der parallelen Threads pro Job, die unter anderem von der Anzahl der physisch verfügbaren Cores abhängt.

Die beiden Parameter müssen unter Berücksichtigung konkret verfügbarer Ressourcen und deren Ausstattung gewählt werden. Einfluss haben unter anderem die folgenden Größen:

- Hauptspeicherbedarf pro Job, der sich aus dem Bedarf für eine Einzelsimulation, der Anzahl der parallelen Threads und dem Job-Overhead ergibt;
- Plattenspeicherbedarf pro Job für die temporäre Datenspeicherung während der Jobausführung
- Netzwerkbandbreite für die Übertragung von Eingangs- und Ausgangs- bzw. Ergebnisdaten.

Will man die genannten Größen in quantifizierbare Relationen setzen, sind grundsätzlich die folgende Vorgehensweisen möglich:

- Die komplette Ressourceninfrastruktur wird in einem Modell erfasst, mit dem über Simulation quantitative Aussagen zur Auslastung und zu den Kosten (Gesamtzeitbedarf, Speicherbedarf, etc.) abgeleitet werden können. Ein Beispiel hierfür wird in [2] mit dem Simulator *CloudSim* beschrieben. Der Aufwand ist groß, zumal auch das Verhalten einzelner Simulationsexperimente mit modelliert werden muss. Diese Option ist für Provider geeignet, die ihre Ressourcen energie- und kosteneffizient betreiben wollen.
- Aus Anwenderperspektive ist es eher interessant, grundsätzlich bei der Durchführung von Simulationsexperimenten den jeweils aktuellen Ressourcenverbrauch zu erfassen und zu protokollieren. Die so gesammelten Daten können später ausgewertet und als Grundlage zur Ressourcenbedarfsschätzung für künftige, ähnliche Experimente dienen.
- Eine weitere Möglichkeit bietet das testweise Ausführen kleiner, abgerüsteter Experimente in Verbindung mit der Ressourcenverbrauchserfassung. Hier besteht die Möglichkeit, unmittelbar vor einer Ausführung eines geplanten Experiments auf den Zielressourcen unter den jeweils aktuell vorliegenden Lastbedingungen sogenannte *probes* auszuführen und die dabei für die konkrete Experimentkonfiguration Messdaten zu gewinnen, die wiederum Grundlage für weitere Schätzungen sein können.

Für *GridWorker* wurde der letztgenannte Ansatz implementiert und evaluiert. Die Details werden im Folgenden beschrieben.

3 *GridWorker*-Kommando *probe*

Um *GridWorker*-Anwendern eine einfache Möglichkeit zu bieten, für Simulationsexperimente in der Cloud oder auf anderen Compute-Backends den Ressourcenbedarf vorab zu schätzen, wurde das Kommando *probe* implementiert:

```
gridworker probe
             [ -n <jobs>.<threads> ]
             [ -t <timeout> ]
```

Der Anwender bereitet wie gewohnt sein Simulationsexperiment vor und erstellt eine Konfiguration. Im Anschluss kann mit *probe* mittels verschiedener Messungen in Kombination mit einer Extrapolation eine Übersicht zum erwarteten Ressourcenbedarf ausgegeben werden. Ein Beispiel für ein *probe*-Ergebnis ist in Bild 3 dargestellt. Für eine Variantensimulation müssen ca. 100000 Parameterkombinationen simuliert werden. Mit *probe* kann binnen weniger Minuten ermittelt werden, dass bei einer Verteilung des Problems auf 26 Knoten eines Clusters mit Quadcore-CPUs (26 Jobs zu je 4 Threads) eine reichliche halbe Stunde benötigt wird. Eine genauere Betrachtung der Messer-

```
lina5> gridworker probe -n 26.4

MEASUREMENTS
================================================================================
tasks                1     2     5    10    20    50   100   200   500  1000
jobs.threads
--------------------------------------------------------------------------------
        26.4       32s   27s   34s   32s   31s   28s   31s   32s   46s   51s
================================================================================

ESTIMATIONS
================================================================================
       tasks         1       10      100     1000    10000   100000 1000000
jobs.threads
--------------------------------------------------------------------------------
        26.4       30s      31s      33s      52s       4m      37m     6h
================================================================================
```

Bild 3. Mess- und Schätzergebnisse von *GridWorker probe*

gebnisse zeigt zudem, dass eine Mindestlaufzeit von einer halben Minute gibt, die *GridWorker* für die Organisation der Jobs generell benötigt.

Der Algorithmus, der *probe* zugrunde liegt, ist Folgender: *GridWorker* führt nacheinander Simulationsexperimente aus und erfasst dabei den jeweils benötigten Ressourcenbedarf. Es wird mit einer einzelnen Task, was einer einzelnen Simulation entspricht, begonnen und anschließend wird die Task-Anzahl schrittweise vergrößert. Die Task-Zahlen sind dabei konfigurierbar. Entscheidend ist, dass *GridWorker* nach jedem *probe*-Experiment grob abschätzen kann, wie lange das nächste Experiment dauern wird. Der Anwender kann beim *probe*-Kommando optional eine Maximalzeit (*timeout*) angeben, nach der die Messungen beendet werden sollen. Damit besteht die Möglichkeit, für Experimente, für die keinerlei Erfahrungen über die erforderliche Simulationszeit vorliegen, innerhalb einer definierten Zeitspanne quantitative Aussagen zum erwarteten Ressourcenbedarf zu erhalten. Es ist einzusehen, dass bei langen Simulationszeiten auch für die *probe*-Phase mehr Zeit eingeräumt werden muss oder aber keine quantitativen Aussagen möglich sind, was in diesem Fall ein Indiz für den Anwender ist, dass der Ressourcenbedarf offenbar (extrem) groß ist und eventuell das Simulationsmodell vereinfacht oder der Parameterraum bei Variantensimulationsexperimenten eingeschränkt werden muss.

Auf Basis der Messwerte werden zusätzlich mittels Extrapolation Schätzwerte für weitere Experimentkonfigurationen bestimmt. Hierfür werden intern verschiedene Regressionsverfahren eingesetzt. Es ist geplant, für ein weiteres *GridWorker*-Kommando *estimate* die Schätzverfahren weiter zu verbessern.

```
#jobs = <jobs>
#threads = <threads>

for #tasks in 1 2 5 10 20 50 ...
do
   begin measurement

   solve problem using #jobs with
   #threads per job

   end measurement

   if <timeout> exceeded then break
done

print measurement results
calculate estimations
print estimation results
```

Bild 4. *probe*-Algorithmus zur Messung und Schätzung des Ressourcenbedarfs

4 Evaluation

Um die Funktionsweise des Kommandos *probe* nachzuweisen und die implementierten Mess- und Schätzalgorithmen hinsichtlich Praxistauglichkeit zu bewerten, wurden zahlreiche Anwendungsbeispiele getestet.

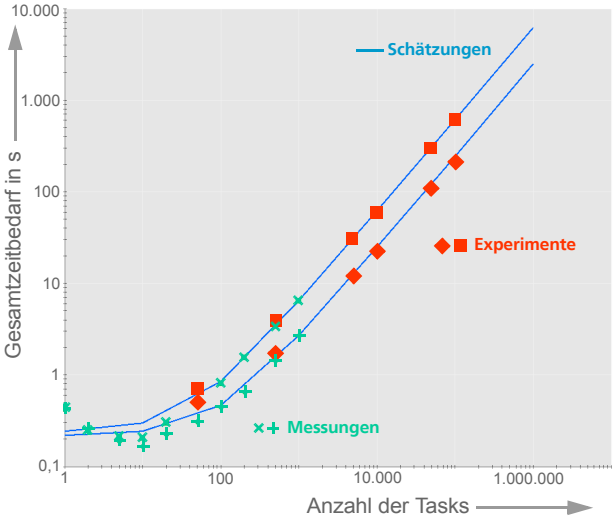

Bild 5. Ergebnisse (Gesamtsimulationszeit) des Kommandos *probe* für zwei Experimentkonfigurationen für die Parameterstudie zur Filterschaltung (grüne Kreuze: *probe*-Messungen, blaue Linie: *probe*-Schätzung, rote Quadrate: reale Experimente)

In Bild 5 sind die Ergebnisse für zwei Konfigurationen für ein Simulationsexperiment (Parameterstudie für eine Filterschaltung) dargestellt. Hierfür wurden Simulationsexperimente auf Multi-Core-Rechnern sowie auf einem 128-Core-Compute-Cluster durchgeführt.

Die Schätzung stimmt sehr gut mit den bei realen Experimenten gemessenen Simulationszeiten überein.

Bei einem zweiten Beispiel wurden der Gesamtzeitbedarf für die in [6] beschriebene Fehlerdiagnose mittels *GridWorker probe* ermittelt. Die Experimente wurden dabei auf unterschiedlichen Compute-Ressourcen durchgeführt: ein Laptop mit Dual-Core-CPU, zwei Compute-Server mit 8 und 16 CPU-Cores, ein 128-Core-Compute-Cluster und eine OpenStack-Test-Cloud mit drei VM-Instanzen. Ziel war es, die Eignung von *probe* in den unterschiedlichen Leistungsklassen der vier Back-ends zu prüfen. Da *GridWorker* über Adapter, die sogenannten *Worker*, die verschiedenen Compute-Ressourcen direkt und für den Nutzer transparent ansprechen kann, wurde jeweils das gleiche Beispielszenario verwendet und lediglich der Konfigurationsparameter *gridworker.worker* angepasst.

Beim Fehlerdiagnose-Beispiel [6] werden für ein SPICE-Modell mit ADC-DAC-Loop-Back-Struktur (ADC – analog digital converter, DAC – digital analog converter) für *1420* Fehlersituationen (*1197* ADC-Fehler + *223* DAC-Fehler) jeweils *1000* Monte-Carlo-Simulationen, bei denen Schaltungsparameter zufällig variiert werden, simuliert. Berücksichtigt man die beiden fehlerfreien Situationen und das jeweilige Nominalverhalten, müssen im konkreten Fall *1422 x 1001 = 1.423.422* Einzelsimulationen durchgeführt werden. Damit die *probe*-Evaluierung in einer akzeptablen Zeit und mit einer Vielzahl von Experimenten durchgeführt werden konnte, wurde die Zahl der Monte-Carlo-Simulationen auf *10* begrenzt. Pro Experi-

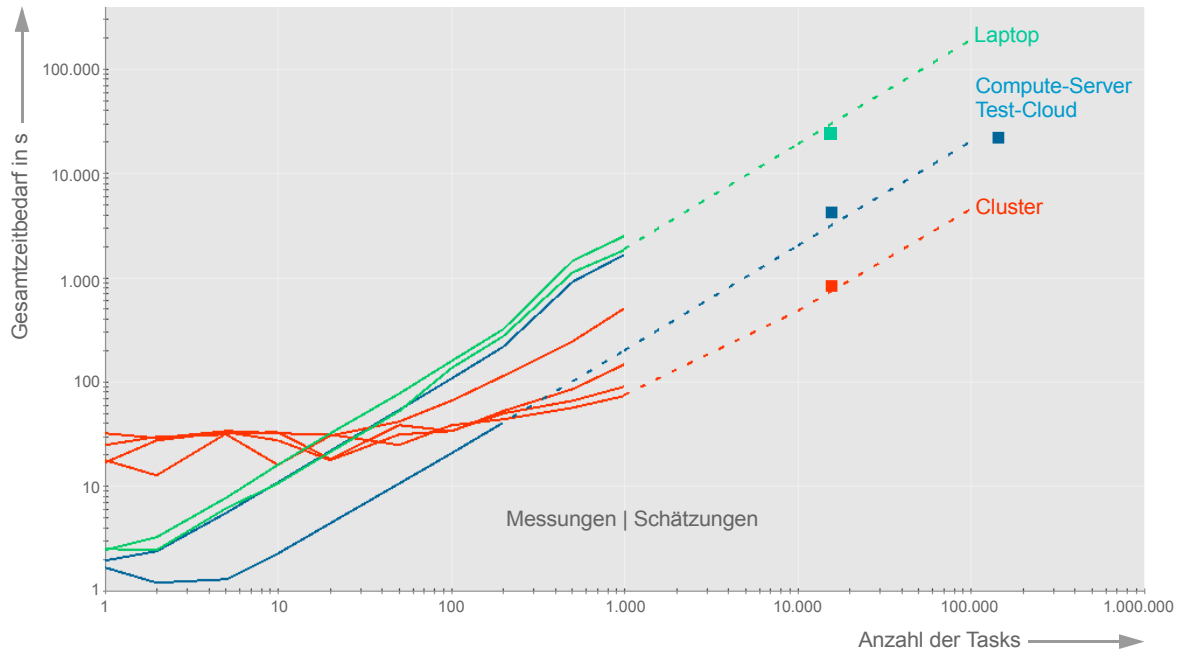

Bild 6. Evaluierungsergebnisse für das Beispiel Fehlerdiagnose. Die Linien im linken Diagrammbereich zeigen *probe*-Messungen für unterschiedliche Parallelisierungsgrade, die Strichlinien im rechten Diagrammbereich zeigen die *probe*-Schätzungen für je einen typischen Parallelisierungsgrad pro Backend. Die Quadrate repräsentieren konkrete Simulationsexperimente.

ment mussten damit nur noch *1422 x 11 = 15642* Simulationen ausgeführt werden.

In Bild 6 sind die Evaluierungsergebnisse zusammengefasst. Für Back-end-Plattformen in drei unterschiedlichen Leistungsklassen wurde mittels *GridWorker probe* zunächst der Gesamtzeitbedarf automatisiert geschätz. Anschließend wurden einzelne konkrete Simulationsexperimente durchgeführt. Teilweise konnten – analog zu den Ergebnissen in Bild 5 – recht gute Übereinstimmungen für die geschätzten und tatsächlich benötigten Zeiten erzielt werden. Teilweise traten noch Abweichungen auf, die im weiteren Projektverlauf näher untersucht werden. Deutlich sichtbar wird in Bild 6 auch der enorme Zeitaufwand für größere Simulationsexperimente mit sehr vielen Einzelsimulationen. Werden die Vorteile von Cluster und Grid Computing mit denen von Cloud Computing verbunden, können die Simulationszeiten teilweise um Größenordnungen reduziert werden, ohne dass lokal eine Hochleistungsinfrastruktur vorgehalten werden muss.

Insgesamt arbeitet das *probe*-Verfahren bereits in der aktuellen Implementierung sehr robust und hinreichend zuverlässig und stößt, nicht zuletzt aufgrund der einfachen Bedienbarkeit, auf sehr gute Resonanz bei den Anwendern.

5 Zusammenfassung

Mit dem Ausbau leistungsfähiger, skalierbarer Compute-Ressourcen auf Grid- und Cloud-Basis ergeben sich vollkommen neue Möglichkeiten für die Bearbeitung komplexer, rechenintensiver Simulationsexperimente. Im Rahmen des Projektes *Cloud4E* [3] werden hierfür prototypische Dienste entwickelt, die eine Erschließung dieser neuen Möglichkeiten für Aufgaben aus dem Ingenieursbereich vereinfachen sollen.

```
ESTIMATIONS
================================================================================
       tasks          1       10      100     1000    10000   100000 1000000
jobs.threads
--------------------------------------------------------------------------------
        20.3        31s       31s       35s       76s        8m       75m   12h
        10.3        26s       26s       33s       96s       12m      117m   19h
         5.3        24s       25s       36s        2m       20m        3h   34h
         2.3        27s       30s       54s        4m       45m        7h    3d
         1.3        20s       25s       68s        8m       79m       13h    5d
================================================================================
```

Bild 7. *probe*-Zeitschätzungen für das Fehlerdiagnose-Experiment [6] mit bis zu einer Million Einzelsimulationen bei einer Aufteilung auf 1 bis 20 parallele Jobs mit je 3 Threads pro Job.

Eine Herausforderung besteht für Anwender darin, bei der Konfiguration von Simulationsexperimenten vorab den Ressourcenbedarf abzuschätzen. Neben modellbasierten Ansätzen erweisen sich vor allem Ansätze auf der Basis von Probemessungen vielversprechend. Der Beitrag stellte hierfür eine für das *GridWorker*-Framework implementierte Methode *probe* vor, bei der zunächst einfache, danach schrittweise komplexere Simulationsexperimente durchgeführt werden und die dabei gewonnenen Messergebnisse für den Ressourcenverbrauch als Grundlage für eine Ressourcenbedarfsschätzung dienen. Das Verfahren wurde erfolgreich an praxisrelevanten Beispielen evaluiert.

Es ist geplant, den derzeit für die Gesamtsimulationszeit realisierten Ansatz auf weitere Ressourcenindikatoren wie Speicherbedarf und Netzwerkbandbreite auszudehnen und schließlich über hinterlegte Preismodelle auch die erwarteten finanziellen Kosten mit in die Schätzung einzubeziehen.

Die vorliegende Arbeit ist im Rahmen des Verbundprojektes „Cloud4E – Trusted Cloud Compting for Engineering“, das vom Bundesministerium für Wirtschaft und Technologie unter dem Kennzeichen 01MD11053 gefördert wird, entstanden.

Literatur

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

[2] Buyya, R.; Ranjan, R.; Calheiros, R.N.: Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challanges and Opportunities. IEEE Int. Conference on High Performance Computing & Simulation, HPCS '09, Leipzig, Germany, June 21-24, 2009, pp. 1-11

[3] Cloud4E – Trusted Cloud Computing for Engineering. http://www.cloud4e.de

[4] 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

[5] *GridWorker*: http://www.gridworker.de

[6] Gulbins, M.; Schneider, A.; Rülke, S.: Ein Cloud-basierter Workflow für die effektive Fehlerdiagnose von Loop-Back-Strukturen. Grid4Sys-Workshop „Grid-, Cloud- und Big-Data-Technologien für Systementwurf und -analyse“, Dresden, 27.-28. November 2013

[7] Iosup, A.; Ostermann, S.; Yigitbasi, M.N.; Prodan, R.; Fahringer, T.; Epema, D.H.J.: Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. IEEE Trans. on Parallel and Distributed Systems, Vol. 22, No. 6, June 2011

[8] Limmer, S.; Schneider, A.; Boehme, C.; Fey, D.; Schmitz, S.; Graupner, A.; Sülzle, M.: Services for Numerical Simulations and Optimizations in Grids. In: Dr. Nitin; Prof. Satya Prakash Ghrera (Ed.): Proceedings of 2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, December 6-8, 2012, pp. 214 -219

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

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

[11] 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

[12] 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