GeNiuS

Newsletter von GNS und GNS Systems

Ausgabe 1/2007

FETI-INDEED - Teil 2

Verteilte Umformsimulation auf Compute-Clustern

Im ersten Artikel über *FETI-INDEED* in der vorangegangenen Ausgabe von GeNiuS wurde die FETI-Methode zunächst allgemein vorgestellt. Darüber hinaus wurde die Gebietszerlegung mit dem Programm *METIS* erläutert. Im vorliegenden Artikel soll nun der mathematische Hintergrund der FETI-Methode beleuchtet werden.

Zerteilung des Gleichungssystems

Die Diskretisierung eines Problems der Strukturmechanik mit Hilfe der Methode der Finiten Elemente führt im Falle eines quasi-statischen Prozesses auf ein Gleichungssystem der Form

$$Ku = f \qquad (1)$$

Hierin bezeichnet u den gesuchten Vektor der (inkrementellen) Knotenverschiebungen und f den bekannten Vektor der (inkrementellen) äußeren Knotenkräfte. Der Zusammenhang zwischen diesen beiden Größen wird mathematisch durch die so genannte Steifigkeitsmatrix K des Gesamtsystems beschrieben. Die Randbedingungen, die sich durch den Kontakt ergeben, seien im obigen Gleichungssystem bereits enthalten.

Das zu behandelnde Gebiet Ω werde nun in P Teilgebiete Ω_1 bis Ω_P zerlegt (Partitionierung; vgl. Abb. 1). Dann kann für jedes Gebiet jeweils ein separates Gleichungssystem aufgestellt werden, das sich formal zunächst nicht vom Gleichungssystem (1) unterscheidet. Die Gleichungssysteme der P Teilgebiete sind jedoch nicht vollkommen unabhängig voneinander: Würde man die lokalen Systeme separat betrachten, so würde man die zwischen den Teilgebieten herrschenden Kräfte ignorieren und somit zu physikalisch falschen lokalen Lösungen kommen. Dieser Fehler kann mit Hilfe der Verschiebungen der Knoten auf den inneren Rändern Γ_{ij}, $i, j \in \{1, \ldots, P\}$, beschrieben werden. Da diese Knoten zu mindestens zwei Teilgebieten gehören, wird für jeden dieser Knoten für jedes Teilgebiet eine eigene Verschiebung berechnet. Auf Grund des erwähnten Fehlers werden diese Verschiebungen nicht übereinstimmen. Zur Lösung dieses Problems modelliert man die Abhängigkeiten auf den inneren Rändern. Dabei koppelt man die Verschiebungen bezüglich der Teilgebiete durch Gleichungen der Form

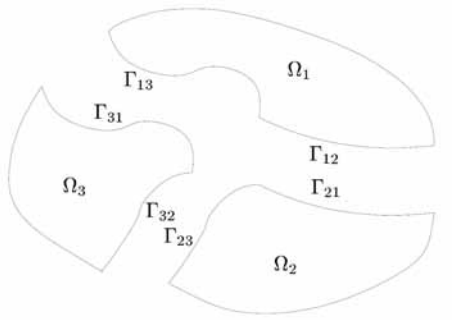


Abb.1: Zerlegung des Grundgebietes in Teilgebiete

Verschiebung im Teilgebiet i - Verschiebung im Teilgebiet j = 0.

Da es sich bei den Verschiebungen um Komponenten der Vektoren u_i bzw. u_j handelt, kann man alle diese Beziehungen in Matrix-Vektor-Notation zusammengefasst als

$$\sum_{i=1}^{P} B_i u_i = 0 \qquad (2)$$

schreiben, wobei die Matrizen B_i nur die Werte -1, 0 und 1 enthalten und die Zugehörigkeit von Randknoten zu den einzelnen

News

'Ford of Europe' steigt um auf Animator3 für das Postprocessing in der Crashberechnung

Keiper beauftragt GNS Systems mit der Entwicklung und dem Support einer CAE-Anwendungsinfrastruktur

GNS mbH und GNS Systems GmbH sind Industriepartner des Projektes PartnerGrid (www.partnergrid.de)

Beiträge

Teilgebieten wiedergeben. Die gleichen Matrizen B_i können eingesetzt werden, um für die einzelnen Teilgebiete Ω_i Korrekturterme für die Ränder zu beschreiben, wodurch sich das modifizierte Gleichungssystem

$$K_i u_i = f_i - B_i^T \lambda \quad (3)$$

ergibt. Die Komponenten des Zusatzterms auf der rechten Seite beschreiben dann gerade die in der bisherigen Betrachtung noch vernachlässigten Kräfte zwischen den Teilgebieten Ω_1 bis Ω_P.

Das ursprüngliche lineare Gleichungssystem (1) wurde somit in P Gleichungssysteme (3) mit einer Nebenbedingung (2) zerlegt. Diese Gleichungssysteme lassen sich bei bekanntem Vektor λ (den man auch als Lagrange-Multiplikator interpretieren kann) jeweils unabhängig voneinander lösen. Lediglich die Bestimmung des Lagrange-Multiplikators benötigt Informationen aus allen Teilgebieten.

Floating Subdomains

Bei der Partitionierung des Gebietes können Teilgebiete entstehen, die nicht statisch bestimmt gelagert sind und die daher Starrkörperbewegungen ausführen können. Die zugehörigen Gleichungssysteme (3) besitzen somit mehr als eine Lösung. Solche Gebiete werden als "Floating Subdomain" bezeichnet. Die Steifigkeitsmatrizen K_i dieser Teilgebiete sind singulär, d. h. nicht invertierbar. Daher ergibt sich die gesuchte Lösung u_i in der Form

$$u_i = K_i^+ (f_i - B_i^T \lambda) + R_i \alpha_i \quad (4)$$

wobei K_i^+ eine verallgemeinerte Inverse von K_i darstellt. Die Matrix R_i basiert auf den redundanten Zeilen von K_i. Sie erfüllt die Gleichung $K_i R_i = 0$. In der Praxis können die Spalten von R_i als Richtungsvektoren der Starrkörperbewegungen des Gebietes Ω_i interpretiert werden. Daher wird R_i zu einer nulldimensionalen (also de facto nicht vorhandenen) Matrix und K_i^+ zur klassischen Inversen von K_i, wenn kein Floating Subdomain vorliegt. Die a_i geben die Linearkombination der Bewegungen an. Um diese so zu bestimmen, dass die Lösungen auf den einzelnen Teilgebieten tatsächlich klaffungsfrei ineinander übergehen, wird ausgenutzt, dass der Starrkörperbewegung eines Teilgebietes keine innere Energie zugeordnet ist. Somit ist das Produkt aus R_i und der Differenz von Kraftvektor f_i und den Kräften $B_i^T \lambda$ auf den inneren Rändern Null

$$R_i^T \left(f_i - B_i^T \lambda \right) = 0 \quad (5)$$

bzw. umgeformt

$$R_i^T f_i = R_i^T B_i^T \lambda \quad (6)$$

Duales Randproblem

Um Gleichung (3) mit Hilfe der Darstellung (4) lösen zu können, müssen zunächst λ und a_i bestimmt werden. Die Gleichung (6) liefert einen Teil der hierfür benötigten Bedingungen. Die noch fehlenden Gleichungen ergeben sich, indem wir Gleichung (4) in Gleichung (2) einsetzen:

$$\sum_{i=1}^{P} B_i u_i = \sum_{i=1}^{P} B_i \left(K_i^+ \left(f_i - B_i^T \lambda \right) + R_i \alpha_i \right) = 0 \quad (7)$$

In der geringfügig umgeformten Fassung

$$\sum_{i=1}^{P} B_i K_i^+ B_i^T \lambda = \sum_{i=1}^{P} B_i K_i^+ f_i + B_i R_i \alpha_i \quad (8)$$

ist dies gerade die noch fehlende Forderung. Damit ergibt sich insgesamt durch Zusammenfassung von Gleichung (8) mit den Gleichungen (6) für alle Teilgebiete ein Gleichungssystem für die inneren Randknoten:

$$\begin{pmatrix} F_I & -G_I \\ -G_I^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \alpha \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{P} B_i K_i^+ f_i \\ -R_{F_1}^T f_{F_1} \\ \vdots \\ -R_{F_{N_f}}^T f_{F_{N_f}} \end{pmatrix} \quad (9)$$

mit $F_I = \sum_{i=1}^{P} B_i K_i^+ B_i^T$,

$$G_I = \begin{pmatrix} B_{F_1} R_{F_1} & \dots & B_{F_N} R_{F_N} \end{pmatrix}$$

und $$\alpha = \begin{pmatrix} \alpha_{F_1} \\ \vdots \\ \alpha_{F_N} \end{pmatrix}$$

Der Index N steht im Gleichungssystem (9) für die Anzahl der Floating Subdomains, da die anderen Teilgebiete kein R_i und a_i besitzen und somit keinen Beitrag liefern.

In der Matrix

$$L = \begin{pmatrix} F_I & -G_I \\ -G_I^T & 0 \end{pmatrix} \quad (10)$$

sind alle Bedingungen zusammengefasst, die für die gemeinsamen Ränder der Teilgebiete Ω_1 bis Ω_P gelten. Die Einschränkung der Gleichungen auf die Knoten der inneren Ränder wird hierbei von den Matrizen B_i vorgenommen. Das Gleichungssystem (9) wird daher als duales Randproblem bezeichnet.

Nach der Lösung dieses dualen Problems auf den inneren Rändern werden mit Gleichung (4) die Verschiebungsvektoren u_i auf den Gebieten Ω_1 bis Ω_P berechnet. Hierbei ist keine gesonderte Behandlung der Floating Subdomains notwendig.

Zusammenfassung: Ablauf eines Iterationsschrittes

Um mittels Gleichung (4) die Verschiebungsvektoren u_i berechnen zu können, ist zunächst das duale Randproblem (9) zu lösen, das die Lagrange-Multiplikatoren bestimmt.

Aufgrund der Struktur der Matrix L aus Gleichung (10), die die Bedingungen für die Ränder repräsentiert, bietet sich das CG-Verfahren (Conjugate Gradients) zur Lösung des Gleichungssystems an. Es benötigt einen geringeren Rechenaufwand als ein direktes Verfahren wie etwa das Gaußsche Eliminationsverfahren.

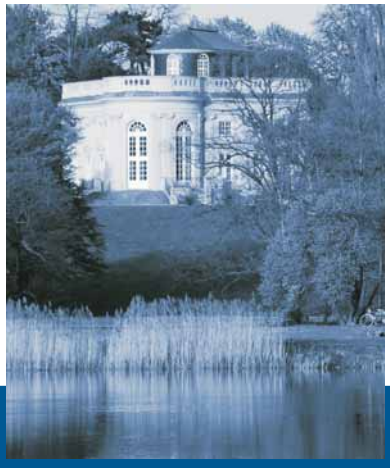
Die Dimension des dualen Problems ist sehr viel geringer als diejenige des Gesamtproblems, da hier nur die Freiheitsgrade der inneren Randknoten und die Anzahl der Starrkörperbewegungen berücksichtigt werden müssen. Außerdem

sind zumindest Teile dieses Problems verteilt lösbar, weil die einzelnen Matrix-Vektor-Produkte der Matrizen aus F_I unabhängig voneinander berechnet werden können.

In jedem Iterationsschritt ist somit zunächst das kleine duale Randproblem mittels CG-Verfahren zu lösen. Diesen Teil der Aufgabe bezeichnet man auch als FETI-Iteration. Mit den so bestimmten Lagrange-Multiplikatoren lassen sich anschließend über die Beziehungen (3) und (4) in unabhängigen Rechnungen auf den einzelnen Cluster-Knoten die Verschiebungsvektoren bestimmen.

Eine Darstellung der Skalierbarkeit sowie eine Beschreibung des Funktionsumfangs von *FETI-INDEED* erfolgt in der nächsten *GeNiuS*-Ausgabe.

Martin Washausen, GNS Systems GmbH
indeed@gns-mbh.de



CORA steht nun in Evaluator zur Verfügung

Immer mehr und komplexere Vorgänge im Bereich der Fahrzeugsicherheit werden durch Berechnungsmodelle abgebildet. Die Ergebnisauswertung erfordert damit einen ständig steigenden Zeitaufwand. Möglichst viele der anfallenden Routinearbeiten sollten daher automatisiert werden, so dass mehr Zeit für die eigentliche Analyse zur Verfügung steht.

Im Auftrag der *PDB* (Partnership for Dummy Technology and Biomechanics) hat *GNS* das Programm *CORA* erstellt. Es bewertet die Übereinstimmung von xy-Kurven. Hierbei kann es sich um den Vergleich von Ergebnissen aus Versuch und Berechnung, aber auch um den Vergleich von Ergebnissen handeln, die mit verschiedenen FEM-Programmen ermittelt wurden. Als Ergebnis liefert *CORA* einen Korrelationswert C im Bereich 0=schlecht bis 1=gut.

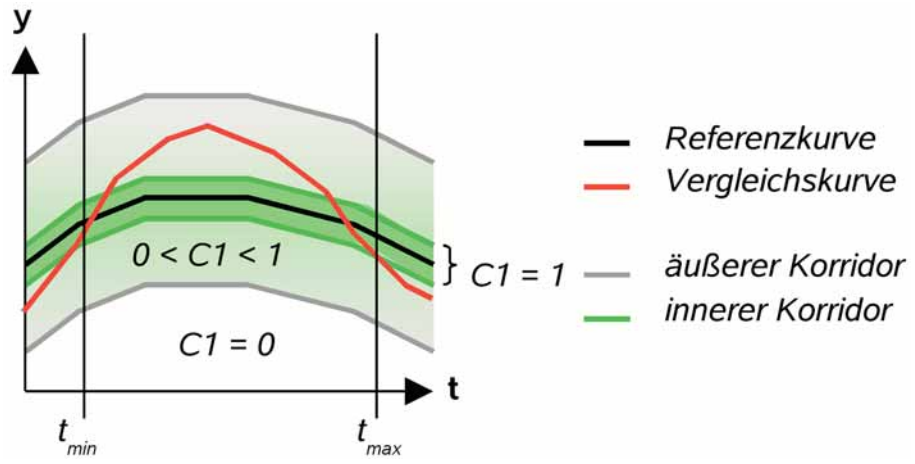


Abb.1: Das Korridorverfahren

Die Reduzierung von Kurvenverläufen auf einen einzelnen Wert stellt immer eine Gefahr da. Betrachtet man z.B. den HIC (Head Injury Criterion), einen Wert, der sich aus der Beschleunigung ermittelt und im Fußgänger- und Inssassenschutz verwendet wird, so stellt man fest, dass zwei stark unterschiedliche Kurven durchaus den gleichen HIC ergeben können. Damit wird deutlich, dass sich ein Kurvenvergleichskriterium aus den Werten beider Kurven ergeben muss und nicht ein Vergleich von zwei Werten sein kann, die sich jeweils aus einer Kurve ergeben.

CORA enthält zwei Verfahren zur Bewertung der Kurvenübereinstimmung, die getrennt oder kombiniert benutzt werden können.

1. Das Korridorverfahren

Um eine der beiden Kurven, im weiteren Referenzkurve genannt, werden zwei Korridore gebildet. Im einfachsten Fall haben diese Korridorkurven einen konstanten Abstand zur Referenzkurve (Abb. 1).

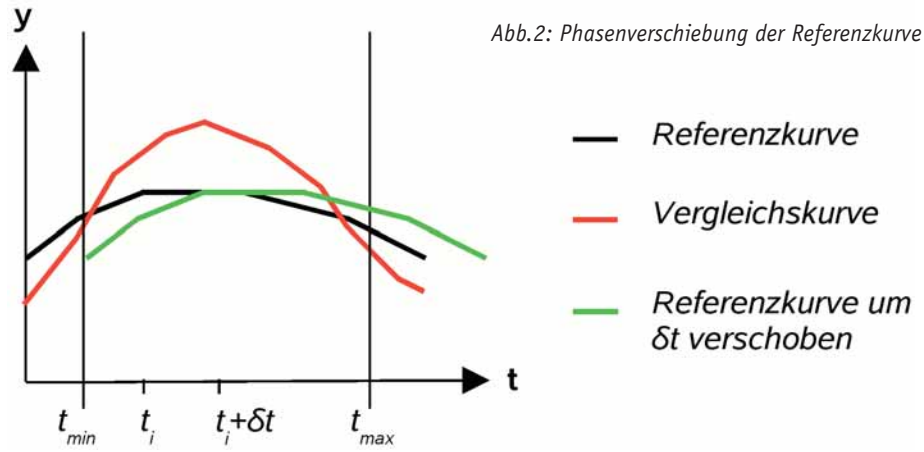


Abb.2: Phasenverschiebung der Referenzkurve

Liegt nun ein Funktionswert der Vergleichskurve innerhalb des kleineren Korridors, so wird dies mit 1 bewertet. Liegt er hingegen außerhalb des äußeren Korridors, so wird dies mit 0 bewertet. Innerhalb des äußeren Korridors wird das Ergebnis zwischen 0 und 1 interpoliert. Als Ergebnis C1 ergibt sich der Mittelwert aller Stützstellen im gewählten Intervall.

Das Kreuzkorrelationsverfahren

Für das Kreuzkorrelationsverfahren wird der Korrelationswert K berechnet. Er ergibt sich aus allen Wertepaaren innerhalb des betrachteten Intervalls (t_min bis t_max): Um eine Phasenverschiebung der beiden Kurven gegeneinander zu erfassen, wird die Referenzkurve um das Vielfache ihres Stützstellenabstandes dt verschoben (Abb. 2) und so ein maximaler Korrelationswert ermittelt. Aus diesem Korrelationswert, der dazugehörigen Phasenverschiebung und dem Flächenverhältnis der beiden Kurven wird dann der Kreuzkorrelationswert

$$K_{xy}(m) = \frac{\sum_i x(t_{i+m})\, y(t_i)}{\sqrt{\left(\sum_{i=i_1}^{i_2} x^2(t_{i+m}) \sum_{i=i_1}^{i_2} y^2(t_i)\right)}}$$

C2 bestimmt.

Die Ergebnisse der beiden Verfahren können mit Wichtungsfaktoren versehen werden, um eine kombinierte Bewertung zu erhalten. Darüberhinaus ist es möglich, eine Gesamtbewertung für alle zu vergleichenden Kurvenpaare zu bestimmen. Die Aufgabe des Berechnungs- oder Versuchsingenieurs besteht darin, sinnvolle Parameter, wie das zu berücksichtigende Zeitintervall und die Wichtungsfaktoren, festzulegen, um danach eine automatische Kurvenbewertung zu erhalten.

Die vollständige Funktionalität des Programms *CORA* ist auch im aktuellen Release von *Evaluator* implementiert.

Carsten Thunert, GNS mbH
eva@gns-mbh.com

Windows Compute Cluster Server als Basis für High Performance Computing

Die große Mehrheit der heutzutage verwendeten Compute Server laufen unter dem Betriebssystem *Linux*. Von den 500 weltweit leistungsfähigsten Rechnersystemen werden 389 unter *Linux* betrieben. 338 der 500 Systeme beruhen auf *AMD x86_64* oder *Intel EM64T* Prozessortechnologie. In diesem stark wachsenden Marktsegment bietet nun auch *Microsoft* mit dem *Windows Compute Cluster Server 2003* eine Software-Infrastrukturlösung.

Einfache Komplettlösung

Der *Compute Cluster Server (CCS)* ist für die oben bereits genannten Prozessortypen

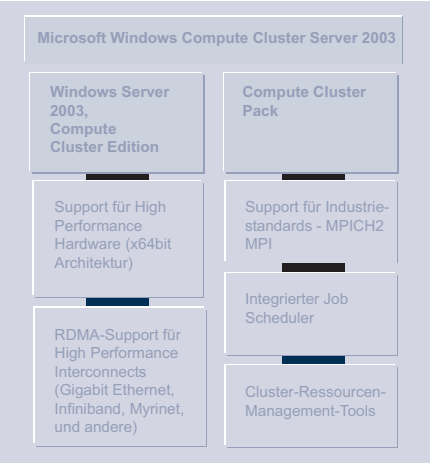


Abb. 1: Bestandteile des Microsoft Windows Compute Cluster 2003

verfügbar und beinhaltet einen kompletten Satz an Funktionsmodulen für den Einsatz in Berechnungsumgebungen bis zu mittlerer Größe.

Wie in Abbildung 1 dargestellt, besteht *Windows CCS* aus zwei Hauptkomponenten, dem kostenpflichtigen *Windows Server 2003* in der Compute Cluster Edition und dem kostenfreien Compute Cluster Pack. *Windows Server 2003* unterstützt die bereits erwähnte 64 bit Hardware. Unterstützt wird auch der Remote Direct Memory Access, notwendige Basis für den Einsatz von High Performance Interconnects wie z.B. *Myrinet* oder *Infiniband*.

Das Cluster Pack besteht aus folgenden Teilkomponenten:

- Microsoft MPI
- Job Scheduler
- Cluster Management Tools

Das *Microsoft MPI* ist eine Implementierung, die auf MPICH2 basiert. Der Job Scheduler besitzt einen rudimentären Funktionsumfang. So gibt es z.B. nur eine Queue mit fünf Prioritätsklassen, in die Jobs eingereicht werden können. Die Cluster Management Tools ermöglichen die komplette Einrichtung, Konfiguration und Überwachung eines Clusters.

Einrichtung

Bei der Einrichtung und Konfiguration eines Clusters muss zwischen drei relevanten Systemtypen unterschieden werden: dem Head Node, der den Cluster verwaltet, den Compute Nodes, auf denen die eigentlichen Berechnungen durchgeführt werden und den Clients, von denen aus ein Anwender-Jobs einreicht und überwacht.

Auf dem Head Node und den Compute Nodes muss für die Verwaltung von relevanten Job- und Systeminformationen *Microsoft SQL Server 2000 Desktop Engine* mitinstalliert werden. Außerdem wird das *Microsoft .NET Framework 2.0* auf allen drei oben erwähnten Systemtypen benötigt, um grafische Oberflächenelemente anzuzeigen. Im Gegensatz zu Head und Compute Nodes können CCS Clients unter *Windows 2003*, *XP64* und *XP32* betrieben werden.

Die Einrichtung des Clusters erfolgt unter Verwendung integrierter Wizards in folgenden Schritten:

- Einrichtung des Head Nodes, inkl. Active Directory und DHCP
- Konfiguration der Netzwerktopologie
- Erstellung eines Image für Compute Nodes
- Einrichtung von Compute Nodes über Image-Verteilung
- Einrichtung von Usern und User-Gruppen

Abbildung 2 zeigt einen der im Rahmen der Cluster-Einrichtung verwendeten Wizards.

Nutzung

Microsoft CCS bietet dem Anwender Oberflächenelemente an, mit denen Batch Jobs erstellt und eingereicht werden können. Die Jobs selbst bestehen wiederum aus einer oder mehreren Tasks, die ggf. in Abhängigkeit zueinander stehen. Sämtliche Tasks können unterschiedliche Systemanforderungen wie z.B. die benötigte Anzahl CPUs stellen. Abbildung 3 zeigt das Oberflächenelement für die Definition einer Task.

Die Job Queue und die darin enthaltenen Jobs bzw. Tasks können ebenfalls über eine dedizierte grafische Oberfläche verwaltet werden. Die Automatisierung von Prozessabläufen innerhalb der einzelnen Tasks sollte über die *Windows PowerShell* erfolgen.

Die von den Oberflächenelementen bereitgestellte Funktionalität kann der User auch über folgende Kommandozeilenbefehle bedienen:

`job`	Jobs verwalten
`task`	Tasks verwalten
`node`	Knoten aufnehmen und verwalten
`cluscfg`	Queue überwachen und verwalten
`clusrun`	Befehl auf Cluster-Knoten ausführen

Job-Informationen werden im XML-Format gespeichert und so an den Job Scheduler übergeben. Über diesen Mechanismus können auch andere Software-Werkzeuge für die Erstellung von Jobs verwendet werden. Diese müssen lediglich die relevanten Informationen im vorgegebenen XML-Format erzeugen.

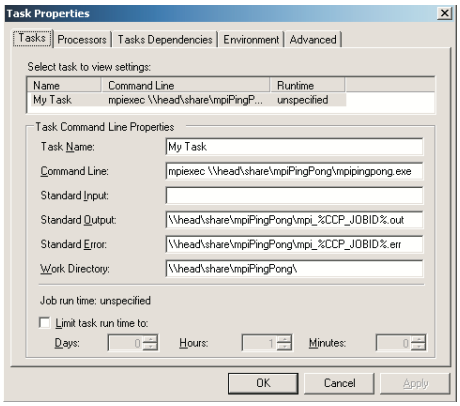


Abbildung 3: Grafische Oberfläche für die Definition von Job Tasks.

Performance

Performance-Messungen haben gezeigt, dass Berechnungen unter *Linux* momentan durchschnittlich 10-20% schneller sind als unter *Windows*. Das trifft insbesondere für größere Crash- oder Strömungssimulationen zu.

Vor- und Nachteile

Der größte Vorteil des *Windows Compute Cluster Servers* ist sicherlich die Möglichkeit der engen Integration mit anderen *Windows*-Software-Komponenten, die bereits in vielen Unternehmen vorhanden sind (z.B. Systems Management Server und Active Directory). Außerdem gibt es mittlerweile einige CAE-Softwarepakete, wie z.B. *Ansys Workbench*, die auf dem Arbeitsplatzrechner unter *Windows* laufen und eine enge Integration mit dem *Compute Cluster Server* bereitstellen (z.B. zum direkten Einreichen von *Ansys* Jobs).

Nachteile liegen vor allem im momentanen Umfang der Gesamtumgebung. So ist die unterstützte Hardware auf *AMD x86_64* und *Intel EM64T* Systeme beschränkt. Die Verfügbarkeit von CAE-Anwendungen und deren Performance sind insgesamt geringer als unter *Linux*. Außerdem würde die Umstellung einer bestehenden *Linux*-basierten Infrastruktur mit einem großen Arbeits- und Kostenaufwand verbunden sein.

Ausblick

Microsoft Compute Cluster Server 2003 stellt eine vollständige Software-Infrastrukturlösung für den sehr rechenintensiven Anwendungsbereich dar. Vorerst wird CCS für kleine und mittelständische Betriebe eine interessante Alternative zu Linux-basierten Infrastrukturen sein.

Es wird demnächst eine verbesserte und erweiterte CCS-Version auf Basis von *Windows Server 2008* geben. Außerdem werden in naher Zukunft weitere CAE-Anwendungen und angrenzende Third-Party-Produkte verfügbar sein. Somit wird der *Windows Compute Cluster Server* dann auch als Basis für große High-Performance-Computing-Umgebungen dienen können.

Jan Martini, GNS Systems GmbH
info@gns-systems.de

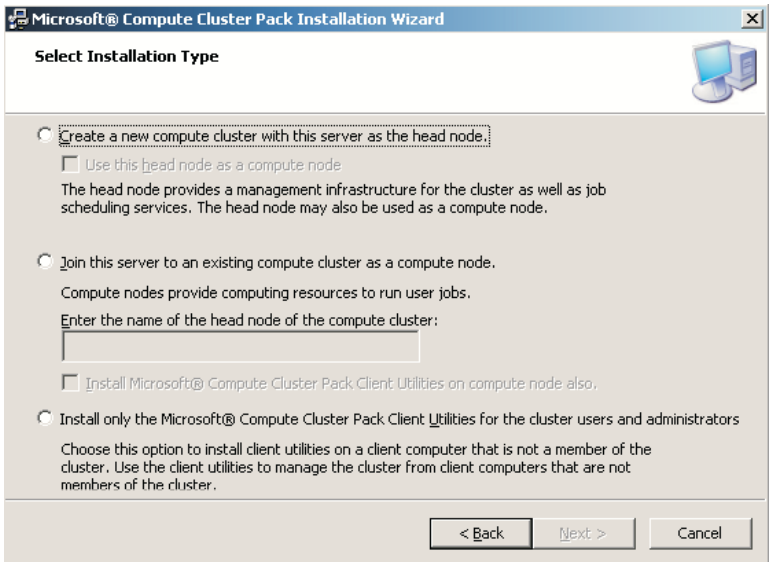
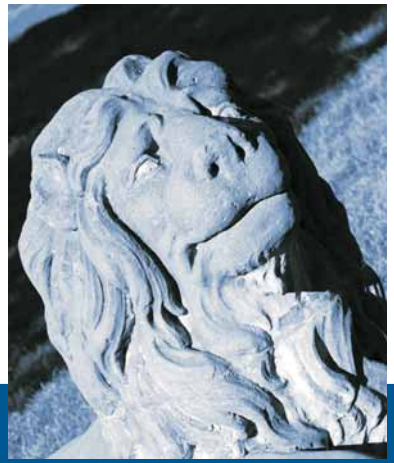


Abbildung 2: Wizard für Cluster-Einrichtung und -Konfiguration



Joberstellung mit dem JGen/X-JGen

Für den Umgang mit den komplexen CAE-Softwareumgebungen muss ein Anwender oft über detailliertes Wissen verfügen. Er muss z.B. darüber informiert sein, auf welchen Maschinen oder Clustern welche CAE-Anwendungen in welchen Versionen installiert sind und mit welchen Parametern die entsprechende Anwendung zu starten ist. Im Normalfall aber wünscht sich der Anwender ein einfaches und bequemes Einreichen von Rechenaufträgen ohne viele Handgriffe, um nach einer angemessenen Wartezeit die Ergebnisse zuverlässig automatisch zurückzubekommen.

Fortgeschrittene Benutzer wünschen sich darüber hinaus Freiheit in der Handhabung ihrer Werkzeuge. Sie wollen Optimierungen oder Varianten automatisieren oder beliebige Rechenabläufe miteinander verketten. Dies kann leicht zu widersprüchlichen Anforderungen unterschiedlicher Nutzergruppen führen.

Das ideale Joberzeugungstool sollte jedoch sowohl Anfänger als auch fortgeschrittene Nutzer unterstützen. Die Anwendungsinfrastruktur wird soweit abstrahiert, dass detailliertes Wissen darüber für die Anwender nicht mehr notwendig ist.

Ansprüche an die Joberstellung

Die Joberstellung erfolgt normalerweise auf der Workstation des Benutzers. Damit der Anwender nicht für jeden Job ein komplettes Shellscript mit allen gewünschten Vorgängen selbst programmieren und bei Änderungen jedes Mal entsprechend anpassen muss, empfiehlt sich die Verwendung eines Joberstellungstools. Dieses verringert den Aufwand erheblich. Dabei sollte der Benutzer möglichst wenige Angaben machen müssen. Beispielsweise könnte das Inputdeck automatisch erkannt werden, wenn sich im aktuellen Verzeichnis nur eines befindet.

Viele Einstellungen lassen sich durch eine Auswertung des Inputdecks bereits automatisch erkennen. Durch eine automatisierte Include-File-Suche kann z.B. festgestellt werden, welche weiteren Dateien für die Rechnung benötigt werden. Viele Einstellungen werden nur selten geändert, so dass mit günstig gewählten Default-Werten viele explizite Benutzerangaben entfallen können. Ferner ist es sehr hilfreich, wenn alle Einstellungen durch Options- und Crosschecks auf Ausführbarkeit geprüft werden, bevor der Job eingereicht werden kann. Oft müssen die Anwender auch mehrere parallel vorhandene Queueing-Systeme adressieren können und sich genau deren unterschiedlicher Handhabung bewusst sein. Auch in diesen Fällen ist ein Joberstellungstool sehr hilfreich, das die entsprechenden Unterschiede kennt und den Benutzer durch den Vorgang führt.

Grundsätzlich unterscheidet man zwei Arten von Joberstellungstools: script-fähige kommandozeilenorientierte Tools, welche sich insbesondere für eine Automatisierung der Joberstellung anbieten, und Tools mit grafischer Benutzeroberfläche, welche schnell einen Überblick über alle Optionseinstellungen ermöglichen. Ein kommandozeilenorientiertes Tool kann sehr einfach in Scripte eingebettet werden, insbesondere wenn es auf das Ende des Rechenjobs warten kann. Es ist nützlich für die Anbindung von Optimierungssoftware und anderen Metatools an die Anwendungsinfrastruktur.

Das Konzept von JGen und X-JGen

Der *JGen/X-JGen* Jobmaker will durch ein zweistufiges Konzept genau diesen Ansprüchen genügen. Der *JGen* ist das zentrale Element der Joberstellung und kann in verschiedenen Modi von der Kommandozeile aus aufgerufen werden. Der X-JGen bietet die entsprechende grafische Oberfläche dazu und steuert den JGen über eben jene Kommandozeile selbst an. Für jede CAE-Anwendung gibt es ein Plugin-Modul, welches die Funktionalität des *JGen* und *X-JGen* erweitert, um Jobs für diese Anwendung zu erstellen. Dieser streng modulare Aufbau erlaubt eine flexible Anpassung an beliebige Anwendungsinfrastrukturen und gewährleistet darüber hinaus die Möglichkeit einer schrittweisen Einführung und einer einfachen dynamischen Erweiterbarkeit für beliebige neue CAE-Softwarepakete (Abb. 1).

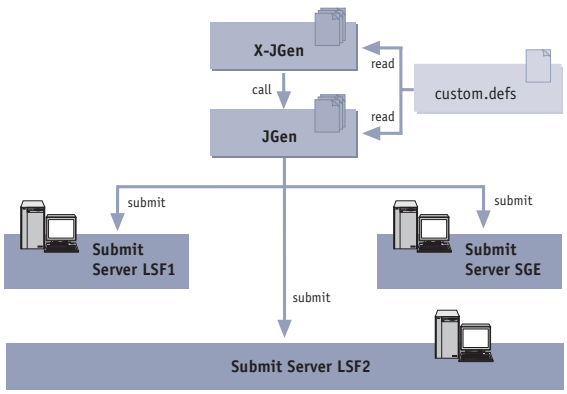


Abb. 1: Modulares Konzept des JGen und X-JGen

Ebenso flexibel ist auch die Anbindung an das Batch-Queueing-System ausgelegt. Unterstützungen für *LSF, Sun Grid Engine* und *PBS* sind bereits integriert, weitere sind in Planung. Eine Besonderheit ist, dass sogar mehrere Queueing-System-Installationen parallel angesteuert werden können.

Die Parameter-Sets eines Jobs können je nach gewünschter Arbeitsweise in verschiedenen Formaten abgelegt werden. *JGen* und *X-JGen* sind auch darüber eng miteinander verbunden (Abb. 2). Das Job File ist das Shellscript, das bei Submission direkt an das Queueing-System eingereicht und ausgeführt wird. Dort lassen sich noch einmal alle Optionen prüfen. Das jdefs-File bietet die Möglichkeit, alle Parameter eines *JGen*-Aufrufs in einer Datei abzulegen. Der *JGen* kann daraus dann beliebig viele Jobs mit den gleichen Optionen erstellen, wobei das Inputdeck jedes Mal auf Fehler geprüft wird. Daher kann das jdefs-File auch unbedenklich editiert werden, um die Joboptionen zu modifizieren. Das xdefs-File kann dazu genutzt werden, die aktuellen Einstellungen der Maske des *X-JGen* abzuspeichern und später wieder zu laden. Auch dieses File kann unbedenklich editiert werden, da nur solche Einträge gelesen werden, die für die derzeitige Maske einen Sinn haben. Das custom.defs-File erlaubt es dem Nutzer, die systemweiten Default-Werte aller *JGen*-Parameter individuell einzustellen. Sowohl der *JGen* als auch der *X-JGen* beziehen ihre Customizing-Informationen aus diesem File.

Der JGen-JobMaker

Anhand von Dateiendungen erkennt der *JGen* mögliche Inputdecks im aktuellen Verzeichnis und leitet daraus bereits den

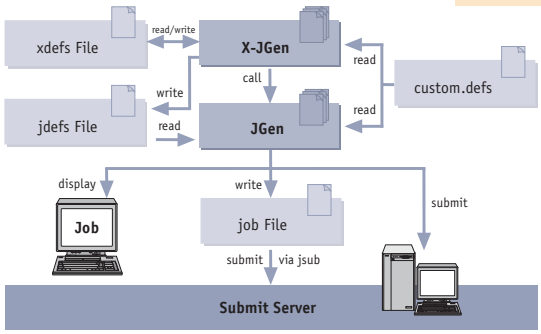


Abb. 2: Datenfluss des JGen und X-JGen

Jobnamen ab. Daher bedarf es keiner Angaben für Inputdeck und Jobname, wenn nur ein Inputdeck im Verzeichnis liegt. Abgeleitet vom Inputdeck wird eine Liste benötigter Include Files erstellt, deren Vorhandensein automatisch geprüft und in die Liste der zu kopierenden Dateien aufgenommen wird. Anschließend werden alle Optionen auf Fehler und Ausführbarkeit geprüft und daraus der Job erstellt. Der *JGen* kann über folgenden Kommandozeilenbefehl aufgerufen werden:

```
jgen -h [-a application]
jgen [-tdwsc] -a application
  [-b bsys] [-v version]
  [-j jobname]
  [-m key1=value1,key2=value2,...]
  [-o key1=value1,key2=value2,...]
```

Zum Anzeigen der Hilfe und zum Testen, Darstellen, Schreiben und Submitten von Jobs mit allen Optionen direkt von der Kommandozeile.

Die notwendigen Kommandozeilenparameter können auch aus einer Definitionsdatei eingelesen werden.

Befehlsbeispiele für einige Applikationen

Im einfachsten Fall sieht ein Aufruf wie folgt aus:

```
jgen -a Abaqus
```

Prüft einen ABAQUS Job (Analysis) im aktuellen Verzeichnis auf Gültigkeit. Es ist nur die Angabe der Applikation nötig, alle anderen Einstellungen sind mit systemweiten oder benutzerspezifischen Defaults belegt.

```
jgen -d -a Abaqus -m \
arch=hp11 \
-o mode=datacheck
```

Zeigt einen ABAQUS-Job (Datacheck) aus dem aktuellen Verzeichnis an, mit dem Wunsch auf HP-UX zu rechnen.

```
jgen -s -a Abaqus -m cpus=2
```

Schickt einen 2-CPU-ABAQUS-Job (Analysis) aus dem aktuellen Verzeichnis ab.

```
jgen -s -a Starcd -v 3400 \
-m arch=irix65,cpus=8 \
-o prec=double,decmeth=sets
```

Schickt einen 8-CPU-Double-Precision-StarCD-Job (Version v3400) für SGI IRIX aus dem aktuellen Verzeichnis ab mit der Dekompositionsmethode `sets`.

Der X-JGen-JobMaker GUI

Der *X-JGen* gibt dem Anwender einen schnellen Überblick über alle wesentlichen Optionen. Insbesondere neue Anwender oder solche, die nur gelegentlich Berechnungen starten, finden sich hier schnell zurecht. Der Aufruf des *JGen* ist in der Oberfläche ersichtlich, so dass der Anwender nebenbei mit den *JGen*-Kommandos vertraut wird. Die Ablage der Parameter ermöglicht es, Einstellungen dauerhaft zu speichern und jederzeit wieder zu verwenden. Ein Beispiel des *X-JGen* mit dem *LS-DYNA*-Modul ist in Abbildung 3 wiedergegeben.

Zusammenfassung und Ausblick

Der Einsatz des *JGen* und *X-JGen* bei der Joberstellung bedeutet eine erhebliche Zeitersparnis sowohl bei der Einarbeitung neuer Mitarbeiter als auch im täglichen Betrieb für den Anwender. Weiterhin wird der Support der Anwender durch die Verwendung standardisierter Jobs erheblich vereinfacht. Diese erleichtern die Nachvollziehbarkeit und die Fehlersuche bei Jobabbrüchen. Der Anwender kann sich somit besser auf den Inhalt seiner eigentlichen Aufgabe, die Berechnung, konzentrieren.

Stefan Ciesla, GNS Systems GmbH
info@gns-systems.de

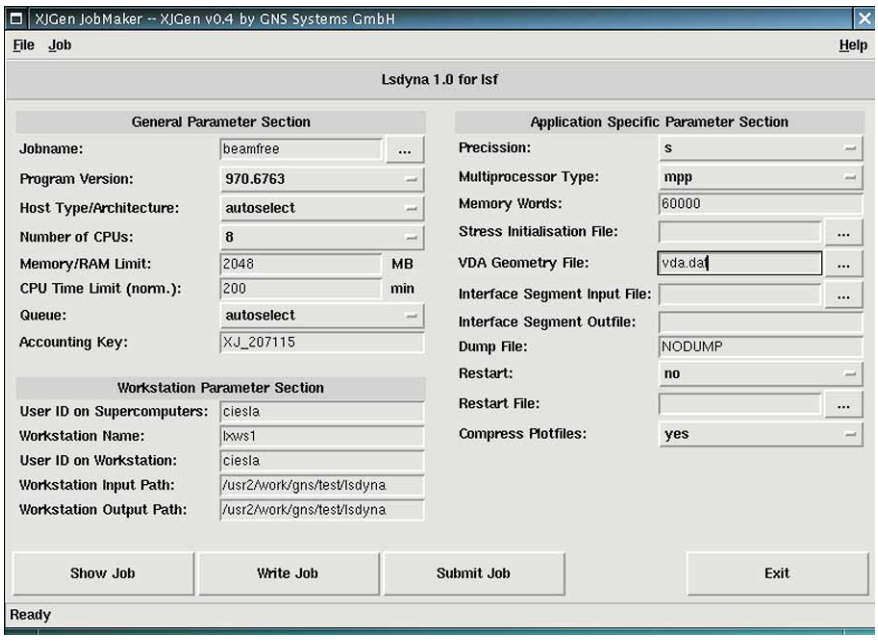


Abb. 3: X-JGen mit dem LS-DYNA Modul

Termine

6. LS-DYNA FORUM 2007 (GNS Systems GmbH + GNS mbH)

Datum: 11.-12. Oktober 2007
Veranstaltungsort: CongressForum Frankenthal
Veranstalter: DYNAmore GmbH

Fahrzeugsicherheit 2007 / Car Safety 2007 (GNS mbH)
Innovativer Kfz-Insassen- und Partnerschutz

Datum: 18.-19. Oktober 2007
Veranstaltungsort: Maritim Berlin
Veranstalter: VDI

Weitere Termine und kurzfristige Änderungen finden Sie auf unseren Webseiten.

Wenn Sie den „GeNiuS" lieber in digitaler Form erhalten möchten, können Sie ihn auf unseren Webseiten herunterladen oder sich auf unsere E-Mail-Verteilerliste setzen lassen.

GNS mbH
Am Gaußberg 2
38114 Braunschweig
Telefon: 05 31-8 01 12 0
Fax: 05 31-8 01 12 79
www.gns-mbh.com

GNS Systems GmbH
Am Gaußberg 2
38114 Braunschweig
Telefon: 05 31-1 23 87 0
Fax: 05 31-1 23 87 11
www.gns-systems.de

Impressum

Ausgabe 01/2007
Erscheinungstermin: August 2007
Herausgeber: GNS Systems GmbH

Verantwortlich: Jan Martini
Redaktion: Anette Tröger
Layout: Anette Tröger

Tipps und Tricks zu Animator3

Übersicht über alle Befehle zum Messen von Längen, Abständen und Winkeln in der aktuellen *Animator3*-Version:

Winkel

```
ide an2:  angle 2 points (*)
```

Mit diesem Befehl können die Winkel einer Geraden gegenüber den drei Raumebenen ausgegeben werden. Es werden immer die kleinsten eingeschlossenen Winkel angegeben.

Abb.1: Messen in der Schnittebene

```
ide ang:  angle 4 points (*)
```

Bei diesem Befehl wird der eingeschlossene Winkel zwischen den Geraden durch die Punkte 1-2 und 3-4 angegeben.

Längen *(Abb. 1)*

```
ide crd: distance in cross section
```

Ermittelt den Abstand zwischen zwei Punkten auf den Schnittlinien eines Modelles. Diese müssen nicht unbedingt Elementknotenpunkte sein.

```
ide dbp: distance pid pid
```

Ermittelt den kürzesten Abstand zwischen zwei PIDs.

```
ide dse: distance node element
```

Ermittelt den kürzesten Abstand zwischen einem Knoten und einem Element.

```
ide dsp: distance node pid
```

Ermittelt den kürzesten Abstand zwischen einem Knoten und einer PID.

```
ide dst: distance, 2 node
```

Bestimmt den Abstand zweier Knoten.

```
ide elo: elongation (*)
```

Längenänderung der Abstände; diese Werte sind vorzeichenbehaftet.

Gemeinsames

Alle Befehle, bis auf die mit (*) markierten, geben als Rückmeldung die Kennzeichnung in der Graphikfläche. Die Farbe der Markierung kann als Trajektorienfarbe mit `col tra` geändert werden. Die Markierung der Items kann mit `opt did on/off` geändert werden.

Christoph Kaulich, GNS mbH
animator@gns-mbh.com

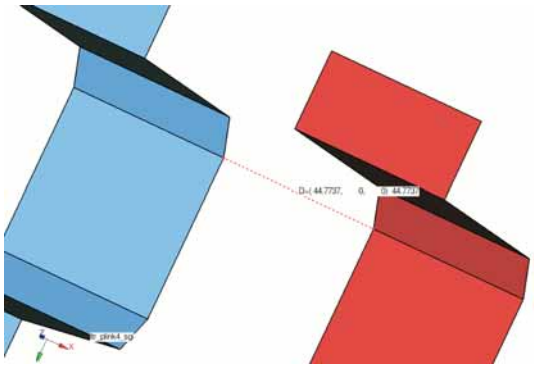


Abb.2: Messen zwischen zwei Slots