Aufbau einer Anwendungsinfrastruktur zur Unterstützung von LS-Dyna Abläufen auf Hochleistungsrechnern

Stefan Ciesla

GNS Systems GmbH, Braunschweig, Deutschland

Abstract:

Efficient usage of LS-Dyna within a high performance compute environment requires an optimized applications infrastructure. This includes the three building blocks job creation, job processing and job control, which, if seamlessly integrated, provide computational engineering results to the users quickly and reliably. Simple standard computations should be supported so that users are not kept from doing their primary tasks by complicated operational procedures. On the other hand, complex requirements from application areas like optimization, variant calculation and automatic results analysis should be enabled by extensive possibilities for configuration.

Zur effizienten Nutzung von LS-Dyna innerhalb einer hochperformanten Rechnerumgebung bedarf es einer optimalen Anwendungsinfrastruktur. Dazu gehören die drei Grundbausteine Joberstellung, -ablauf und -kontrolle. Optimal aufeinander abgestimmt, liefern sie Benutzern schnell und zuverlässig Berechnungsergebnisse. Sie sollten einfache Standardrechnungen so unterstützen, dass der Berechner nicht durch komplizierte Bedienung von seiner eigentlichen Aufgabe abgehalten wird. Außerdem wäre es auch wichtig, komplexe Anforderungen aus den Anwendungsbereichen der Optimierung, der Variantenberechnung und der automatisierten Ergebnisauswertung durch weitgehende Konfigurationsmöglichkeiten zu erfüllen.

Keywords:

Anwendungsinfrastruktur, Hochleistungsrechner, Joberstellung, Jobablauf, Jobkontrolle, Ressourcennutzung, Queueing System, Optimierung, Variantenrechnung, Jobverkettung, Automatisierung

1 Über GNS/GNS Systems

Als Tochterunternehmen der GNS mbH ist die GNS Systems GmbH ein IT-Dienstleister mit Sitz in Braunschweig und weiteren Niederlassungen in Ingolstadt, Rüsselsheim und Sindelfingen. Das Unternehmen wurde 1997 in Braunschweig gegründet und hat für das Jahr 2005 ein Umsatzziel von EUR 1,7 Mio. Seine Dienstleistungsschwerpunkte liegen im Bereich Unix/Windows Systemmanagement, High-Performance-Computing, technisches Datenmanagement und Softwareentwicklung.

Die Firma GNS konzentriert sich auf die Bereiche Simulation und Berechnung, Entwicklung und Vertrieb von Softwareprodukten, Entwicklung kundenspezifischer Software und CAE-Beratung.

2 Anforderungen an eine Anwendungsinfrastruktur aus Benutzersicht

Die einfachste Form einer Anwendungsinfrastruktur liegt in der direkten, interaktiven Nutzung von Großrechnern. Dabei ist es jedoch notwendig, dass die Nutzer die Ressourcen durch Absprachen unter sich aufteilen. Sonst werden Lizenzen und freie Maschinen insbesondere an Wochenenden, unzureichend ausgenutzt. Für den Umgang mit der Software muss jeder Anwender über detailliertes Wissen verfügen, auf welchen Maschinen welche Anwendungen in welchen Versionen installiert sind und wie mit welchen Parametern die Anwendungen zu starten sind. Bei einer ausgereiften Anwendungsinfrastruktur sind solche Absprachen und Kenntnisse nicht erforderlich.

Ferner müssen bei einer interaktiven Nutzung oft regelmäßig Dateien manuell zwischen Userworkstation und Supercomputer hin und her kopiert werden und eventuelle Konvertierungen nach Rechnungen manuell angestartet werden. Häufig gibt es Probleme mit dem Plattenplatz auf den Compute Servern, da die Anwender ihre Verzeichnisse nach abgeschlossener Berechnung nicht entfernen. Auch dies vermeidet eine ausgereifte Anwendungsinfrastruktur. Im Normalfall wünscht sich der Anwender das einfache und bequeme Einreichen von Rechenaufträgen ohne viele Handgriffe, um nach einer angemessenen Wartezeit die Ergebnisse automatisch schnell und zuverlässig zurückzubekommen. Alle gewünschten Nachbearbeitungsschritte wie z. B. Formatkonvertierungen sollten dabei schon automatisch berücksichtigt sein. Der Anwender wünscht über den Stand seiner Berechnungen auf dem Laufenden gehalten zu werden.

Fortgeschrittene Benutzer wünschen sich darüber hinaus viel Freiheit und wollen oft Neues ausprobieren. Sie wollen Optimierungen oder Varianten automatisiert rechnen lassen oder beliebige Rechenabläufe miteinander verketten. Dies kann leicht zu widersprüchlichen Anforderungen unterschiedlicher Nutzergruppen führen. Die ideale Anwendungsinfrastruktur sollte jedoch beide unterstützen z. B. durch intelligente Defaults oder versteckte Komplexität.

3 Eine mögliche Implementierung einer Anwendungsinfrastruktur

Viele Anwender haben sich im Laufe der Zeit bestimmte Arbeits- und Vorgehensweisen angeeignet, die oft eng mit einer bestehenden Anwendungsinfrastruktur verknüpft sind und die sie gern auch weiterhin behalten möchten. Daher kommen zu den allgemeinen im letzten Abschnitt genannten Anforderungen weitere hinzu, die für die jeweilige Arbeitsweise spezifisch sind. Das sollte die Implementierung so weit wie möglich unterstützen. Die hier vorgestellte Implementierung ist daher nur eine von vielen möglichen und soll hier zur Verdeutlichung als Beispiel dienen.

Aufgrund des hohen Rechenaufwandes von realistischen Simulationen und den damit verbundenen Nachteilen eines interaktiven Betriebs können die Ressourcen im Allgemeinen durch den Batchbetrieb effizienter genutzt werden. Im Umgang mit einer batchorientierten Anwendungsinfrastruktur ergeben sich aus Sicht des Benutzers drei grundlegende Phasen (Bild 1): Phase 1 (Joberstellung) ist die Erzeugung und Einreichung eines Rechenauftrages mit allen später relevanten Informationen, Phase 2 (Jobablauf) ist die Durchführung des Rechenauftrages und Phase 3 (Jobkontrolle) ist die Kontrolle der Ergebnisse des Rechenauftrages während und nach der Durchführung.

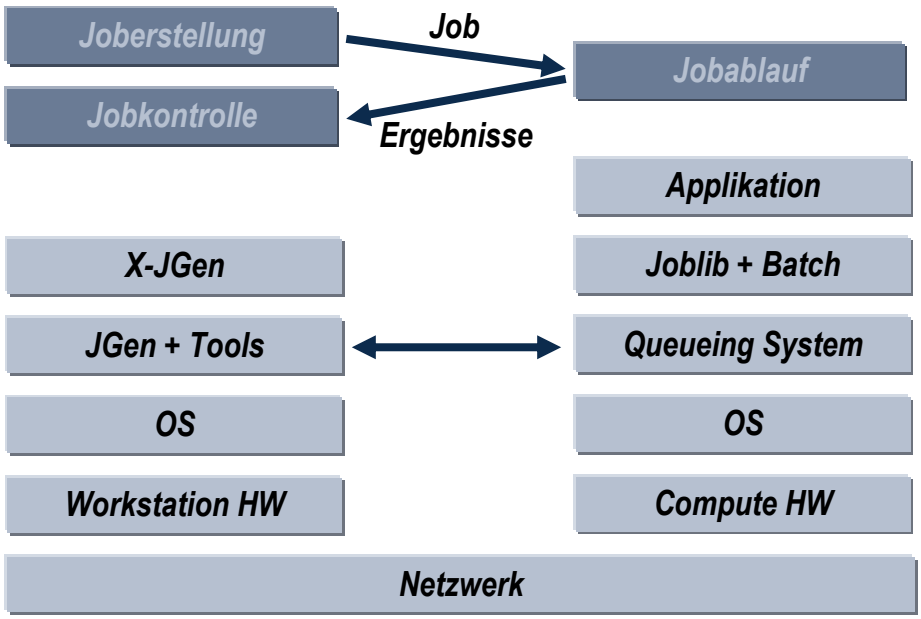

Bild 1: Überblick über eine Anwendungsinfrastruktur

Die Anwendungsinfrastruktur setzt sich aus Hardware- und Softwareinfrastruktur zusammen. Die Hardwareinfrastruktur besteht wiederum aus zwei Bereichen, den User Workstations, mit denen die Benutzer interaktiv arbeiten, und den Batchressourcen (Multiprozessorserver und Cluster). Workstations und Großrechner sind über ein Netzwerk miteinander verbunden. Die Softwareinfrastruktur, die dementsprechend auch in diese Bereiche aufgeteilt wird, unterstützt mit Ihren Komponenten die drei bereits erwähnten Phasen.

3.1 Joberstellung

Die Joberstellung erfolgt im Normalfall auf der Workstation des Benutzers. Um diesen Vorgang möglichst einfach zu gestalten, empfiehlt sich die Nutzung von entsprechenden Tools. So muss der Anwender nicht für jeden Job ein komplettes Shellscript mit allen gewünschten Vorgängen selbst programmieren und bei Änderungen jedes Mal entsprechend anpassen. Ein Joberstellungstool 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 automatisiert erkennen. Durch eine automatische Includefile-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-Einstellungen viele explizite Benutzerangaben entfallen können. Ferner ist es sehr hilfreich, wenn alle Einstellungen durch Option- und Crosschecks auf Ausführbarkeit geprüft werden, bevor der Job überhaupt erst eingereicht werden kann (z. B. wenn ein Includefile fehlt oder eine bestimmte Kombination von Optionen keinen Sinn machen kann).

Oft müssen die Anwender auch mehrere parallel vorhandene Queueing Systeme adressieren können und sich genau deren unterschiedlicher Handhabungsweise 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. Scriptfähige 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 Scripten eingebettet werden, insbesondere wenn es auf das Ende des Rechenjobs warten kann. Es ist nützlich für die Anbindung von Optimie-

rungssoftware und anderen Metatools an die Anwendungsinfrastruktur. Ein Beispiel für ein kommandozeilenorientiertes Joberstellungstool ist in Bild 2 wiedergeben:

```
ciesla@lxws1:~$ jgen -h
JGen JobMaker -- JGen v0.5 by GNS Systems GmbH
----------------------------------------------------------------
Usage: jgen [-tdws] -a application
            [-b bsys] [-v version] [-j jobname]
            [-m mparam1=value1,mparam2=value2,...]
            [-o oparam1=value1,oparam2=value2,...]
…

ciesla@lxws1:~$ jgen -d -a Lsdyna -m cpus=16 -o mptype=MPP
```

Bild 2: Kommandozeilenorientiertes Joberstellungstool: Jgen (Hilfeausgabe)

Ein grafisches Joberstellungstool mit einer Parameterablage ermöglicht es, Einstellungen dauerhaft abzulegen und jederzeit wieder zu verwenden. Alle Parameter und ihre Einstellungsmöglichkeiten sind sofort ersichtlich. Ein Beispiel eines grafisch orientierten Joberstellungstools ist in Bild 3 wiedergeben:

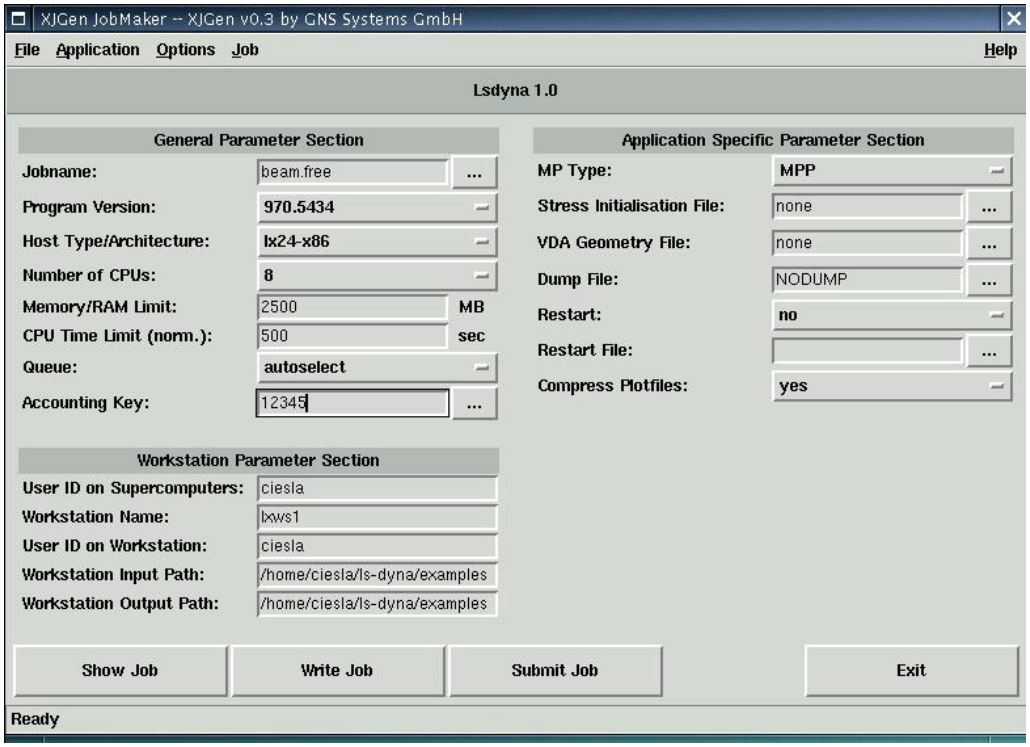

Bild 3: Grafisches Joberstellungstool: XJgen mit LS-Dyna-Modul

Eine Zuhilfenahme dieser Tools bedeutet eine erhebliche Zeitersparnis sowohl bei der Einarbeitung neuer Mitarbeiter als auch im täglichen Betrieb. Die Unterstützung der Anwender durch den Support wird durch standardisierte Jobs erheblich vereinfacht, denn diese erleichtern die Nachvollziehbarkeit und die Fehlersuche bei Jobabbrüchen.

3.2 Jobablauf

Der Jobablauf findet komplett auf Großrechnern statt, die mit einem Batch Queueing System verwaltet werden. Er hat jedoch eine ganze Reihe von Schnittstellen, die über das Netzwerk die Workstation des Benutzers einbeziehen. Wie die Abbildung 4 zeigt, gibt es drei verschiedene Schnittstellen. Der Beginn des Ablaufs wird beim Einreichen des Jobs durch den Benutzer getriggert. In der Regel ist vom Queueing System ein so genannter Submitserver definiert worden, der im 1. Schritt den Job entgegennimmt. Dieser wird im 2. Schritt einem freien Großrechner, dem so genannten Execution Host zugeteilt, der sich während der Vorbereitung die Inputdaten von der User Workstation holt (3. Schritt). Nun kann die eigentliche Rechnung beginnen (4. Schritt). Am Ende der Rechnung werden die erzeugten Ergebnisse wiederum an die Workstation übergeben (5. Schritt).

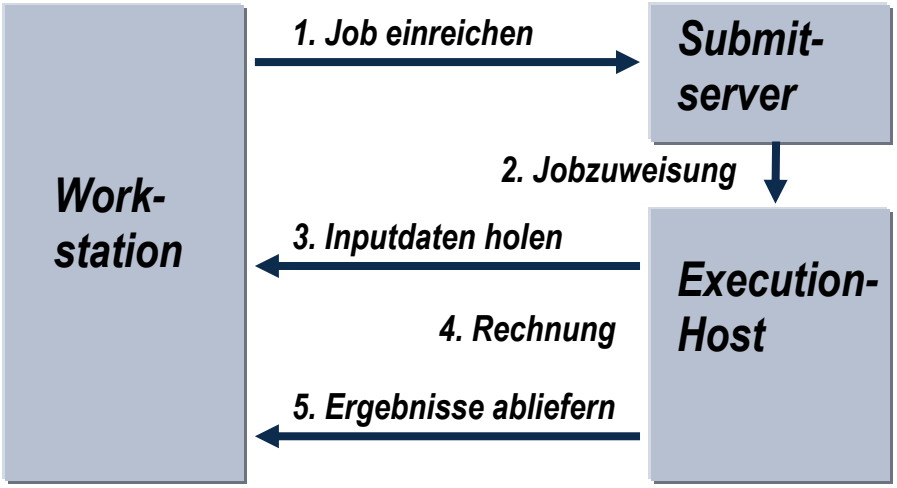

Bild 4: Jobablauf aus Benutzersicht

Auf dem Execution Host sind neben der eigentlichen Applikation (Solver) drei weitere Softwarekomponenten involviert:

- Queueing-System
- Joblib
- Batch- und Postprocessingscripte

Das Queueing-System übernimmt die gesamte Jobverwaltung inklusive Queueing, Scheduling, Dispatching und Running. Ferner gehört dazu die Ressourcenverwaltung, die durch den Abgleich von Ressourcenanforderungen der wartenden Jobs und den jeweils freien Ressourcen des Systems ein Loadbalancing zu erreichen versucht.

Die Joblib führt unabhängig von der Art des Jobs die Initialisierung des Berechnungsvorganges durch. Dazu gehört:

- das Anlegen einer Directorystruktur, in der die Berechnung später stattfinden soll,
- das Erstellen eines Info Files,
- das Kopieren der Inputdateien
- und die Joblog-Vorbereitung für stdout und stderr inklusive der Anzeige der Directory-Inhalte vor und nach Rechnungsanfang bzw. -ende.

Ferner übernimmt die Joblib das Error- und Signalhandling während der eigentlichen Rechnung und sorgt für das Zurückkopieren der Outputdateien sowie für das Cleanup nach Rechnungsende. Das Batchscript übernimmt das Starten des eigentlichen Berechnungsvorgangs durch Programmaufrufe der Applikation (Solver) und der Steuerung des Anwendungsablaufs. Dies kann z. B. auch den Aufruf

einer entsprechenden MPI-Umgebung beinhalten, wenn der ausgewählte Execution Host ein Cluster ist. Das Postprocessingscript (Postscript) führt jegliche Nachbearbeitungsschritte z. B. Merges oder Formatkonvertierungen durch Aufruf von Applikationstools durch. Auch die Ablage und erneute Nutzung von Restart Files kann hierdurch vorbereitet werden. Einen Überblick wie diese drei Komponenten zusammenhängen und in welcher Reihenfolge sie ablaufen, vermittelt die Abbildung 5.

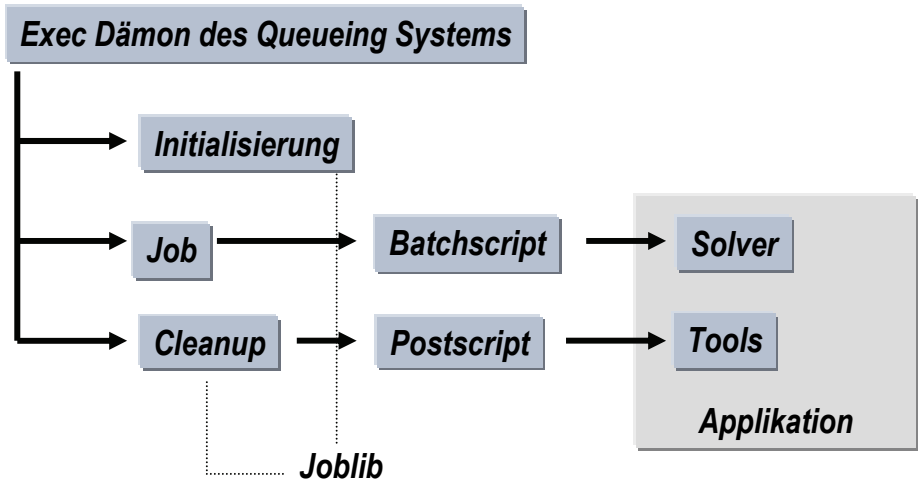

Bild 5: Ablauf auf Execution Host

3.3 Jobkontrolle

Neben den bereits vom Queueing System zur Verfügung gestellten Werkzeugen zur Kontrolle und Anzeige des Jobstatus und zur Einflussname auf den Ablauf können weitere Tools eingesetzt werden, die den Stand der Berechnungen aufzeigen, beispielsweise mit einer Restlaufzeitangabe. Diese sind meist auf eine spezielle Applikation abgestimmt, wie der grafische Jobviewer für LS-Dyna in Abbildung 6.

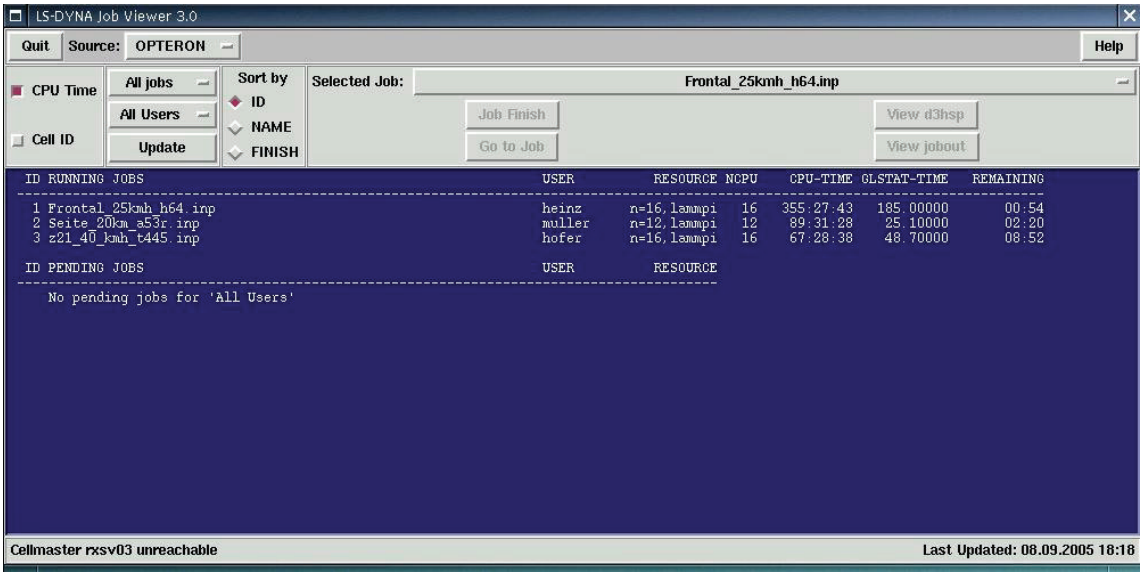

Bild 6: Jobviewer

Zwei nicht-grafische Tools sind das Info File und das Copy Return File. Das Info File ist ein Shellscript, das beim Anlaufen des Jobs zum Anwender kopiert wird. Dadurch weiß dieser, dass die Berechnung begonnen hat. Wird das Info File Script ohne Optionen ausgeführt, liefert es Informationen darüber,

wo der Job angelaufen ist und mit welchen weiteren Optionen es aufgerufen werden kann. Der Aufruf mit einer entsprechenden Option erlaubt beispielsweise das Auflisten der Dateien im Batchverzeichnis, das Anzeigen des stdout/stderr und das Holen von Dateien aus dem Batchverzeichnis zum aktuellen Zeitpunkt, solange der Job läuft.

Das Copy Return File ist das Log File, das den stdout/stderr des Jobs enthält und zum Ende des Jobs als letzte Datei zum Anwender kopiert wird. Dadurch weiß dieser, dass der Job beendet ist. Es liefert detaillierte Informationen über den eingereichten Job und den gesamten Jobablauf (Initialisierung, Batch Script, Postscript und Clean Up) inklusive Erfolg/Misserfolg des Jobs und dessen Gründe. Diese Datei ist der Schlüssel für eine erfolgreiche Fehlerdiagnose durch den Support, falls der Job nicht erfolgreich war.

4 Beispiele

Im Folgenden sollen einige Beispiele erläutert werden, die verdeutlichen, welche Möglichkeiten sich durch den Einsatz einer ausgereiften Anwendungsinfrastruktur ergeben. So wurde z. B. relativ einfach eine Integration der Optimierungssoftware LMS Optimus von Noesis erreicht, indem in den jeweiligen Analysisknoten eines Optimus-Netzwerkes das scriptfähige Joberstellungstool JGen mit Wait-Option aufgerufen wird. Dadurch ist es möglich, automatisch zahlreiche Jobs parallel einzureichen, deren Ergebnisse dann von Optimus eingelesen werden. Daraufhin können neue Jobs erzeugt werden (Bild 7).

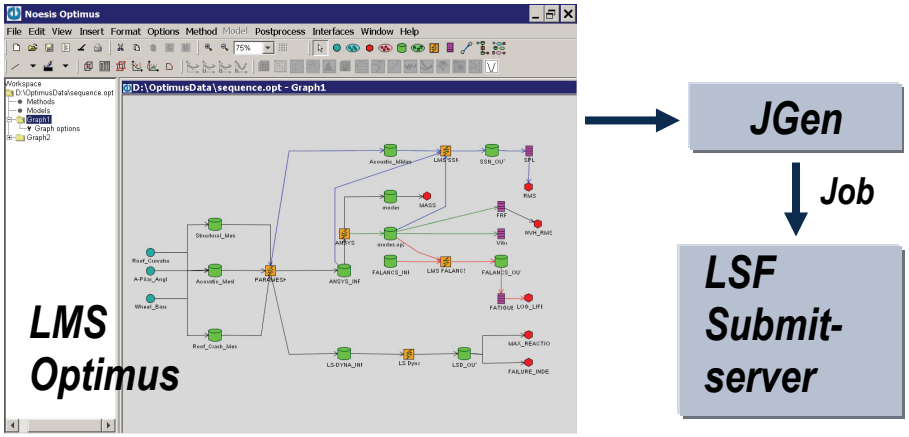

Bild 7: Optimierungsablauf mit Optimus

Ein weiteres Beispiel ist die einfache Möglichkeit, viele Varianten gleichzeitig rechnen zu lassen. (Bild 8). Sind alle Varianten mit ihren Inputdecks in je einem Unterverzeichnis angeordnet, so können mit einer einfachen Schleife über das JGen-Tool und eventuell einer kleinen jdefs-Parameterdatei als Input alle Jobvarianten auf einmal gestartet werden. Dabei genügen auch symbolische Links für identische Include Files.

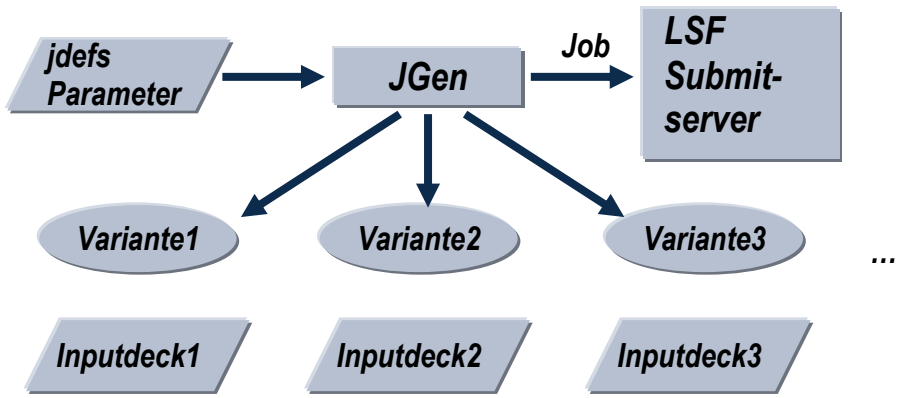

Bild 8: Variantenrechnung mit Jobdefinitions Parametern

Ein häufig genannter Wunsch ist die so genannte Verkettung von Jobs. Das bedeutet, dass eine Reihe von Jobs Abhängigkeiten aufweisen, da ein nachfolgender Job auf den Ergebnissen eines Vorgängerjobs aufsetzt. Diese Form der Automatisierung hat vielfältige Anwendungsmöglichkeiten, wie z. B. automatisiertes Pre- und Postprocessing (oder andere Auswertungen sowie Datenbankablagen), der Wechsel des Solvers nach einer bestimmten Zeitschrittanzahl (die eine Teilrechnung implizit, die andere explizit) für Rückfederungsvorgänge, oder die Verkettung von Tiefziehsimulation und Crash usw. Für die Umsetzung solcher verketteter Abläufe gibt es mehrere Ansätze. Am flexibelsten bleibt natürlich das Scripting vom JGen. Für immer wiederkehrende Verkettungsfälle kann der JGen aber so erweitert werden, dass mit einem Aufruf eine bestimmte Jobkette erzeugt wird, die auch im Queueing System als dependent gekennzeichnet ist und entsprechend abgearbeitet wird (Bild 9)

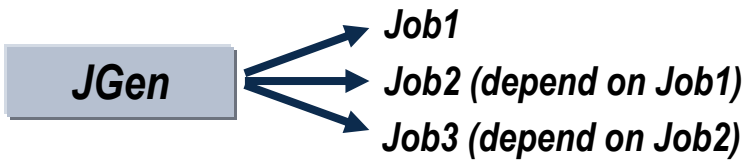

Bild 9: Explizite Jobverkettung

Eine andere Form der Erweiterung des JGen beinhaltet die Möglichkeit, einen so genannten Masterjob einzureichen. Dieser realisiert die Verkettung mit sekundären JGen-Aufrufen auf den Großrechnern (Bild 10).

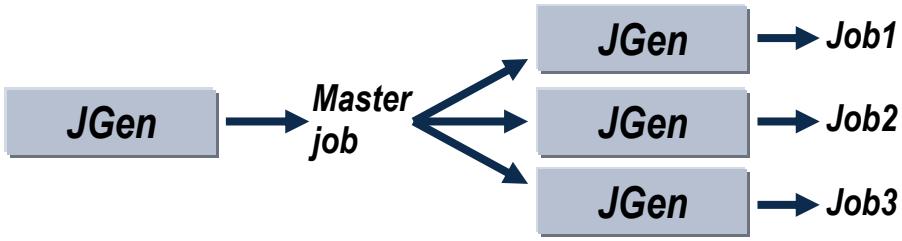

Bild 10: Implizite Jobverkettung

Die hier aufgeführten Beispiele sind nur eine kleine Auswahl der Einsatzmöglichkeiten für eine ausgereifte Anwendungsinfrastruktur.

5 Zusammenfassung

Mit einer optimal abgestimmten Anwendungsinfrastruktur lassen sich auch scheinbar widersprüchliche Anforderungen von Benutzern erfüllen. Dazu bedarf es unter anderem einer mit intelligenten Defaults und Checks versehenen Joberstellung, eines weitgehend automatisierten Jobablaufs und einer modularen, flexibel anpassbaren Softwarestruktur. Eine effiziente Ressourcennutzung geht einher mit einer durchdachten Einbindung des Queueing Systems und einem zuverlässigen, fehlertoleranten Jobablauf.