

ASPIC – Application Service Providing für integrierte technisch-wissenschaftliche Simulationen und deren Visualisierung auf Hochleistungs-PC-Clustern

Sebastian Niedworok *
Universität Stuttgart

Thomas Ertl †
Universität Stuttgart

Zusammenfassung

ASPIC weitet das Konzept des Application Service Providing (ASP) auf technisch-wissenschaftliche Arbeitsbereiche aus. Ressourcenintensive Anwendungen auf Hochleistungsrechnern werden über die Technologien des World-Wide-Webs am lokalen Arbeitsplatz verfügbar gemacht. Die alternative Integration der ASP-Dienste in Workflow-Management-Produkte wurde ebenso durchgeführt, wie die Ausdehnung von ASP auf Remote-Visualisierung. Hierdurch werden die großen Rechenkapazitäten moderner Hochleistungsrechner auch kleineren Institutionen auf einfache Weise zugänglich.

1 Einführung

1.1 Application Service Providing

PC-Cluster finden mittlerweile weite Verbreitung als günstige Hochleistungsrechner in Wissenschaft und Industrie. Die Anschaffung eines leistungsfähigen Clusters ist aber dennoch eine umfangreiche Investition, die nicht immer ohne weiteres rechtfertigt ist. Einerseits bedarf nicht jede Institution ständig einer großen Rechenkapazität, andererseits sind kleinere Institute oder Ingenieurbüros auch bei vorhandenem Bedarf finanziell nicht in der Lage, große Rechenanlagen anzuschaffen.

Application-Service-Providing bietet einen Ausweg aus dieser Problematik:

- Rechenleistung kann bei Bedarf gekauft werden
- Betrieb und Wartung der Rechenanlagen entfallen für den Kunden
- Lizenzkosten für kommerzielle Softwarepakete können auf die tatsächliche Rechenzeit umgerechnet werden (pay per use)

* Institut für Informatik, Abt. Visualisierung und Interaktive Systeme,

† Institut für Informatik, Abt. Visualisierung und Interaktive Systeme,

Im technisch-wissenschaftlichen Bereich hat ASP jedoch bisher kaum Verbreitung gefunden. Das ASPIC-Projekt [ASP] erweitert die ASP-Idee auf diesen Bereich sowie um weitere Aspekte, etwa die Einbindung des ASP-Angebotes in wissenschaftliche Workflowtools und Remote-Visualisierung.

1.2 Herkömmlicher Einsatz von Hochleistungsrechnern

Viele Forschungs- und Entwicklungsprojekte zielen darauf ab, bestehende Rechnerressourcen einer effektiven Nutzung zuzuführen.

Queueing-Systeme wie Platform LSF [LSF] oder Sun GridEngine [Gri] verwalten die Last- und Ressourcenverteilung bei einzelnen Großrechnern. Batchjobs können über ein Kommandozeileninterface vom Benutzer konfiguriert und an das Queueing-System übergeben werden. Dieses ermittelt die Auslastung der von ihm verwalteten Rechner und startet die angeforderte Applikation auf den Rechnern, die ausreichende Ressourcen anbieten. Viele Compute-Server bieten lediglich einen Zugang über Secure-Shell an, die Daten müssen vom Benutzer per `scp` von seinem Arbeitsplatzrechner zum Server transferiert werden und nach Beendigung der Berechnungen wieder zurückkopiert werden.

grid toolkits [Tue01], wie Globus [Glo] oder Avaki [Ava], machen lokale Rechenressourcen auch über Institutionsgrenzen hinaus verfügbar. Sie beinhalten eine "Metaqueueing"-Funktionalität, mit der Rechenjobs bei Bedarf automatisch an entfernte Computer geschickt werden können, verwalten aber auch den Transfer der notwendigen Daten.

1.3 ASPIC

Der Endbenutzer ist normalerweise nicht an den technischen Details, wie die Anwendung, die er benutzen will, in die jeweiligen System-Dienste integriert ist, interessiert. Das ASPIC-System bietet dem Benutzer deshalb ein einfaches, intuitiv über einen Web-Browser bedienbares Interface, mit dem er Batch-Berechnungen starten, überwachen und die Daten transferieren kann.

2 Grundlegende Systemarchitektur des ASPIC-Systems

Die grundlegende Architektur des ASPIC-Systems ist in Abb. 1 dargestellt.

Folgende Softwarepakete wurden implementiert oder zur Implementierung verwendet:

- Apache HTTP-Server, OpenSSL und `mod_ssl` für sichere Datenübertragung (s. Abschnitt 3)
- HTML::Embperl zur Generierung dynamischer Inhalte, wie Jobsubmit-Panel, Directory Browser usw.
- ASPIC-Bibliothek (Perl) für den Zugriff auf System-Dienste (Filesystem, Queueing-System, Datenbank)
- Postgresql als Konfigurations- und Accounting-Datenbank

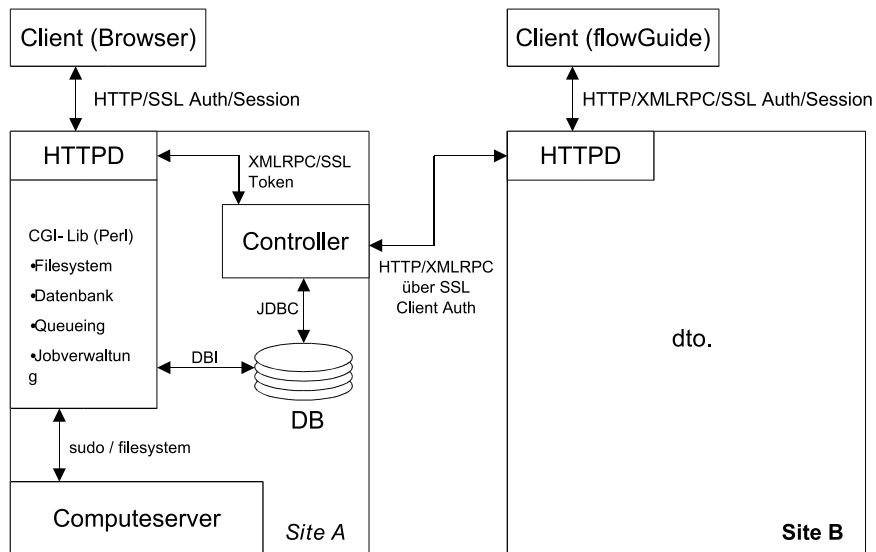


Abbildung 1: Grundlegende Architektur des ASPIC-Systems

- Platform LSF Queueing System
- ASPIC-Controller für Intercluster-Jobtransfer (s. Abschnitt 7)

Als Compute-Server kommt das Linux-MPP-Cluster KEPLER [Kep] des SFB382 der Universität Tübingen zur Anwendung.

3 Authentifizierung und Sicherheit

Da die übertragenen Berechnungsdaten durchaus sensibler Natur sein können, werden bei der Kommunikation mit dem ASPIC-Server konsequent kryptographische Verfahren zum Schutz der Datenübertragung verwendet. Zum Einsatz kommen hier SSL/TLS [All99] und X509-Zertifikate. Diese sind Standards, die sich im jahrelangen Einsatz im Internet bewährt haben.

3.1 Zertifikate

Authentifizierung beider Seiten (Benutzer und ASPIC-Server) ist zwingend notwendig. Einerseits muß der Benutzer bei sensiblen Daten zweifelsfrei feststellen können, wem er diese schickt, andererseits ist die Identifizierung der Benutzer für Abrechnungszwecke unabdingbar.

Standard zur gegenseitigen Authentifizierung im Internet sind X509-Zertifikate. X509-Zertifikate bauen auf der Technik der *public-key*-Kryptographie auf. Im Gegensatz zu symmetrischen Krypto-Verfahren, die mit nur einem geheimen Schlüssel arbeiten, verwendet man hier immer ein Schlüsselpaar, einen öffentlichen *public key* sowie einen geheimen *private key*. Verschlüsselt man Daten mit dem öffentlichen Schlüssel einer Person, so können diese nur mit dem privaten Schlüssel entschlüsselt werden. Umgekehrt kann man Daten "signieren", indem man sie mit dem privaten Schlüssel verschlüsselt. Diese können dann nur mit dem allgemein zugänglichen öffentlichen Schlüssel dekodiert werden.

Ein X509-Zertifikat enthält u.a. die Daten des Zertifizierten, dessen öffentlichen Schlüssel, sowie die *digitale Signatur* der das Zertifikat ausstellenden *certification agency* (CA). Die digitale Signatur besteht im wesentlichen aus einer kryptographischen Prüfsumme (Hash, z.B. SHA-1 oder MD5) der Daten, die mit dem privaten Schlüssel der CA verschlüsselt wurde. Um die Datenintegrität des Zertifikats zu überprüfen, berechnet man mit dem selben Hashing-Algorithmus, mit dem die Signatur erstellt wurde, eine Prüfsumme und vergleicht diese mit der, die die CA berechnet hat. Diese erhält man durch Entschlüsseln der Signatur mit dem allgemein bekannten öffentlichen Schlüssel der CA. Stimmen beide Prüfsummen überein, so kann von der Richtigkeit der Zertifikatsdaten ausgegangen werden. Die öffentlichen Schlüssel der wichtigsten CAs sind in den gebräuchlichen Web-Browsern enthalten. Beim ASPIC-System authentifiziert sich lediglich der ASPIC-Server mit einem von einer bekannten CA ausgestellt Zertifikat. Für die Client-Authentifizierung verwendet ASPIC eigene Client-Zertifikate, die durch eine einfache CA-Funktionalität des ASPIC-Systems durch den Administrator selbst generiert werden. Der ASPIC-Server nimmt lediglich Anfragen entgegen, die durch ein solches Client-Zertifikat authentifiziert wurden. Zertifikate, die durch andere CAs ausgestellt wurden, werden zurückgewiesen.

3.2 SSL/TLS

Der Datenverkehr und das Routing im Internet werden überwiegend durch den TCP/IP-Protokoll-Stack geregelt. Applikationsprotokolle wie HTTP oder LDAP setzen auf dem TCP/IP-Stack auf. Will man diese Protokolle vor fremdem Zugriff schützen, so kann man sich des von Netscape zuerst entwickelten SSL/TLS-Protokolls bedienen. Dieses setzt auf dem TCP/IP-Stack auf und liegt unterhalb der Applikations-Protokolle, die sozusagen durch SSL/TLS getunnelt werden. SSL/TLS bietet folgende Funktionalität:

- Authentifizierung des Servers durch Zertifikate.
- Optionale Authentifizierung des Clients durch Zertifikate.
- Verschlüsselte Datenübertragung. Diese wird durchgeführt durch einen beim SSL-Handshake ausgehandelten symmetrischen Schlüssel, da Verschlüsselung mit symmetrischen Verfahren um Größenordnungen schneller ist, als mit Public-Key Verfahren.

3.3 Session-Verwaltung

Da HTTP ein zustandsloses Protokoll ist, muss sich ein Benutzer bei jedem Request gegenüber dem Server authentifizieren. Dies erfolgt, wie oben erwähnt, normalerweise über Zertifikate. Ein Nachteil dieser Methode ist jedoch der recht hohe Zeitaufwand, da ein voller SSL-Handshake mit beidseitigem Austausch der Zertifikate notwendig ist. Andererseits verfügen die handelsüblichen Browser nur über unzureichende Methoden, automatisch aus mehreren in der Browser-Datenbank abgelegten Zertifikaten auszuwählen. Der Benutzer kann hier entweder nur ein Client-Zertifikat benutzen, oder er muss bei jedem Request eines aus der Datenbank auswählen, was nicht akzeptabel ist. Um dieses Problem zu umgehen, muß der Client sich lediglich beim login in den ASPIC-Server mit seinem Zertifikat ausweisen und erhält dann eine 32-stellige hexadezimale Session-ID, die in die jeweiligen URLs eingebunden wird. Er wird dann in allen folgenden Requests über diese Session-ID authentifiziert. Die Datenübertragung ist nach wie vor verschlüsselt, so dass die Gefahr, in eine solche Session einzubrechen, als sehr gering angesehen werden kann.

Ist der Benutzer durch eine korrekte Session-ID authentifiziert, wird für jeden Request ein Benutzer-Objekt generiert, das bei allen Aufrufen in die ASPIC-Bibliothek durchgereicht wird.

3.4 User-Mapping und Datensicherheit auf dem Server

Da ein Application-Server möglichst nicht mit `root`-Rechten laufen sollte¹, die ASPIC-Benutzer aber mit den von UNIX zur Verfügung gestellten Mitteln voneinander abgeschirmt werden sollten, läuft der ASPIC-Server unter der User-ID `aspic_admin` und führt Programme im Namen des ASPIC-Users via `sudo` aus. Der einzige Befehl, der via `sudo` als `root` ausgeführt wird ist `chown`, mit dem die Zugriffsrechte auf auf den Server transferierte Dateien angepaßt werden. Hier werden die Pfade jedoch auf Plausibilität geprüft, so dass auch mit einem speziell erstellten HTTP-Request innerhalb einer Session das System nicht angegriffen werden kann.

4 Web-basierte Anwendung

Es wurden prototypisch zwei Anwendungen in das ASPIC-System eingebunden:

- **SPH 2000**² ist ein an der Universität Tübingen entwickelter Code zur numerischen Lösung der hydrodynamischen Grundgleichungen. Dieser wird benutzt, um den Primärstrahlzerfall bei der Einspritzung von Diesel in eine Brennkammer zu simulieren.
- **Feko** ist ein kommerzieller Code zur Berechnung von elektromagnetischen Feldverteilungen bei Körpern beliebiger Oberfläche. Feko wird vorwiegend bei der Automobilindustrie eingesetzt.

¹Dies würde bei Ausnutzung einer etwaigen Sicherheitslücke im HTTP-Server das gesamte System kompromittieren

²SPH steht für Smoothed Particle Hydrodynamics

Der Jobsubmit und die Ausführung der Simulation auf dem Server gliedern sich in folgende Schritte:

- Der Benutzer konfiguriert über das Webinterface Simulationsparameter, wie Inputfiles, Anzahl der CPUs, Arbeitsverzeichnis auf dem Server usw. (siehe Abb. 2). Vor dem Submit werden die eingegebenen Daten vom Browser mit einigen JavaScript-Funktionen auf Vollständigkeit überprüft.
- Der Job wird in der Accounting-Datenbank registriert und dem Queuing-System übergeben
- Sind die angeforderten Ressourcen frei, wird der Job vom Queuing-System gestartet. Während des Rechenlaufs kann der Benutzer über den Directory-Browser die Daten im Arbeitsverzeichnis betrachten und herunterladen (Abb. 3)

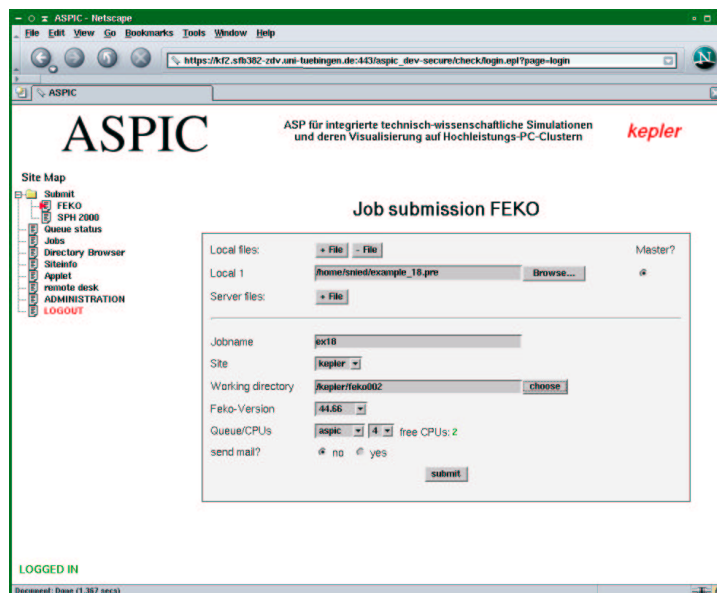


Abbildung 2: Feko job submit

Die Accounting-Datenbank wird sowohl bei Anlauf der Berechnungen als auch bei deren Beendigung durch pre- und post-exec-Methoden der jeweiligen Queue aktualisiert. Das Queuestatus-Panel bietet Informationen zu den aktuell im Queuing-System befindlichen Jobs und ermöglicht es diese zu überwachen und modifizieren (Abb. 4).

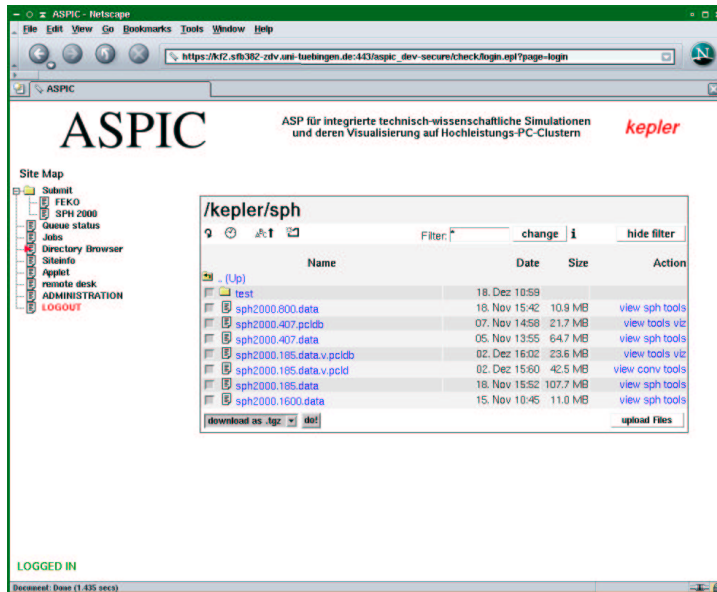


Abbildung 3: Directory Browser

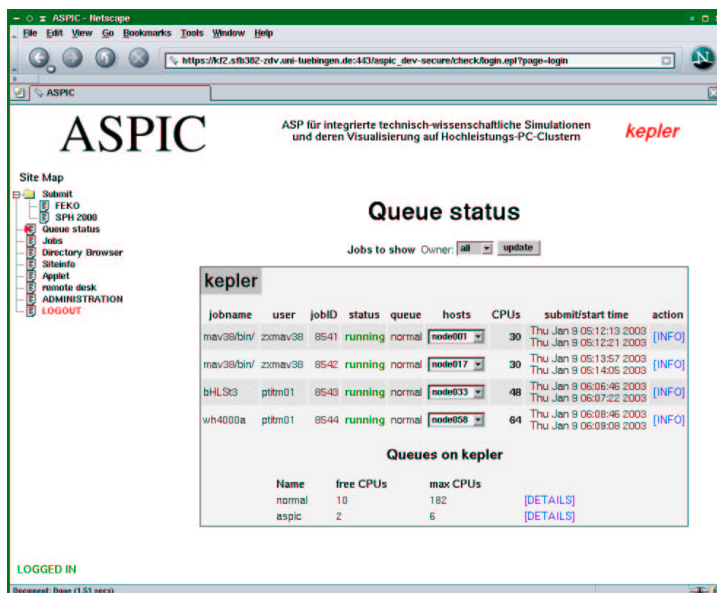


Abbildung 4: Job status panel

5 Integration der ASPIC-Services in andere Softwarepakete

Ein wichtiger Punkt im ASPIC-Projekt ist es, die ASPIC-Services auch Software zur Verfügung zu stellen, die nicht über das Web-Interface auf diese zurückgreifen kann. Hierzu wurde eine auf XMLRPC [XML] basierende Schnittstelle zur ASPIC-Bibliothek geschaffen, die es ermöglicht, Berechnungen zu starten und zu überwachen. Die Kommunikation erfolgt auch hier geschützt durch SSL, jedoch mit dem Unterschied, dass der XMLRPC-Client sich bei jedem Request mit einem Client-Zertifikat authentifizieren muss. Eine Session wird hierbei nicht verwaltet. Für den reinen Datentransfer werden einfache HTTP POST- und GET-Requests verwendet, da der Transfer von Binärdaten via XMLRPC nicht vorgesehen ist.

Prototypisch wurde für die von der science+computing ag [sA] entwickelte Workflowmanagement-Software flowGuide eine Anbindung implementiert. flowGuide hat eine Client-Server-Architektur, der Benutzer kann mit dem flowGuide-Client seine Workflows über die sogenannte Quick-GUI konfigurieren und dann vom flowGuide-Server abarbeiten lassen kann. flowGuide-Client und Server kommunizieren über CORBA. flowGuide-Workflows werden in der auf XML basierenden Sprache flowScript beschrieben, die von der flowGuide-Engine interpretiert und ausgeführt wird. Die ASPIC-Services stehen innerhalb von flowScript zur Verfügung und können bei Bedarf in Workflows eingebunden werden. Der in flowGuide eingebundene *ASPIC-Controller* greift auf eine Datenbank mit den Konfigurationsoptionen des Benutzers zurück und ermittelt alle ASPIC-Server, für die der Benutzer ein Zertifikat hat und die die angeforderte Applikation in der richtigen Version zur Verfügung stellen. Von diesen Servern fordert er Statusinformationen via XMLRPC an und submittet den Job auf einen passenden Server via HTTP und XMLRPC. Ist der Job beendet werden die Ergebnisdaten zur Client-Seite retransferiert und stehen zur Weiterbearbeitung innerhalb des Workflows zur Verfügung. Der momentane Status des Jobs wird ständig in der Datenbank aktualisiert, so dass bei einem eventuell notwendigen Reboot der Client-Seite die Kontrolle über nicht beendete Jobs wiederaufgenommen werden kann.

6 Remote Visualisierung

Der Wunsch, Simulationsergebnisse auf dem ASP-Server selbst hardwarebeschleunigt zu visualisieren, beruht im wesentlichen auf drei Punkten:

- Der Client-Rechner muß nicht über leistungsfähige Grafikhardware verfügen
- Die während eines Simulationslaufs produzierten Datensätze sind zumeist sehr umfangreich so dass ein Retransfer aller Daten zum Client zum Zwecke der Aufarbeitung und Visualisierung nicht immer praktikabel ist. Abbildung 7 zeigt einen SPH-Datensatz von etwa 30 MB Größe, etwa 500 dieser Datensätze werden in einem Simulationslauf erstellt.

- Zur Kontrolle laufender Simulationen ist eine Visualisierung vorteilhaft, da Zwischenergebnisse auf Plausibilität überprüft werden können und die Simulation gegebenenfalls vorzeitig abgebrochen werden kann.

Für Remote-Visualisierung im Rahmen des ASPIC-Systems wurde die in [Ert02] vorgestellte Methode gewählt. Hierbei bedient man sich einerseits eines X-Servers, der die für hardwarebeschleunigtes Rendern unter OpenGL notwendigen GLX-Extensions beherrscht (*Renderserver*). Die Interaktion mit dem Benutzer übernimmt ein weiterer X-Server, der nicht unbedingt hardwarebeschleunigt sein muß (*Interaktionsserver*). Im ASPIC-System dient ein XFree-Server für NVidia-Grafikkarten als Renderserver und TightVNC [Tig] als Interaktionsserver. VNC stellt einen X-Server bereit, der remote durch einen VNC-Viewer bedient werden kann. Das VNC-Protokoll ist darauf optimiert, möglichst wenig Daten über das Netzwerk zu senden, so werden lediglich komprimierte Bitmaps der Änderungen auf dem Desktop transferiert.

Die Visualisierungsapplikation wird lokal auf dem Renderserver gestartet, jedoch wird ihr Environment so konfiguriert, dass sie auf dem Interaktionsserver dargestellt wird. Da VNC keine GLX-Extensions anbietet, müssen die GLX-Aufrufe an den Renderserver weitergeleitet werden. Dies geschieht, durch eine spezielle Bibliothek (*glxforker*), die via *Preloading* zur OpenGL-Applikation hinzugelinkt wird. Preloading ist eine Funktionalität vieler UNIX-ähnlicher Betriebssysteme, mit der man den Linker dazu veranlassen kann, beliebige *shared objects* vor dem Linken der von der Applikation angeforderten Bibliotheken einzubinden. Befinden sich in den durch Preloading eingebundenen *shared objects* Symbole, auf die die Applikation zugreifen will so werden diese verwendet und nicht die in den angeforderten Bibliotheken. Ruft also die Applikation eine GLX-Funktion auf, so wird diese durch eine spezielle Version in der Bibliothek abgefangen und an den Renderserver weitergeleitet. Wenn ein *buffer swap* angefordert wird, wird der Framebuffer ausgelesen, in ein *XImage* geschrieben und via *XPutImage* zum Interaktionsserver gesendet.

Da immer nur ein X-Server hardwarebeschleunigt auf eine Grafikkarte zugreifen kann, gibt es auf Kepler spezielle Visualisierungsrechner³, auf denen nur ein Benutzer zur selben Zeit arbeiten darf.

Die Visualisierungsrechner sind im privaten Subnetz des Clusters eingebunden und somit von aussen nicht direkt erreichbar. Der Benutzer muss daher einen SSH-Tunnel für die VNC-Ports zum Frontendrechner aufbauen. Auf dem Frontend laufen einfache Portforwarding-Proxies, die auf den eigentlichen Visualisierungsrechner weiterleiten (Abb. 6). Auf Client-Seite bedarf es lediglich einer Secure-Shell sowie eines VNC-Viewers. Eine Remote-Visualisierung wird über den Directory Browser des Web-Frontends gestartet. Ist ein Visualisierungsknoten verfügbar, so wird dieser in der ASPIC-Datenbank als belegt markiert und die entsprechenden X-Server sowie die Visualisierungsapplikation via Remote-Shell gestartet. Der Benutzer beendet eine Visualisierungs-Session wiederum über das Web-Interface, wobei der Visualisierungsrechner in der Datenbank freigegeben wird und die auf ihm laufenden Applikationen beendet werden. Session-Start und Session-Ende werden in der Datenbank für eine später erfolgende Abrechnung festgehalten.

³Dual Athlon XP 2000, 2GB Ram, GeForce 4 Ti 4600 Grafikkarte

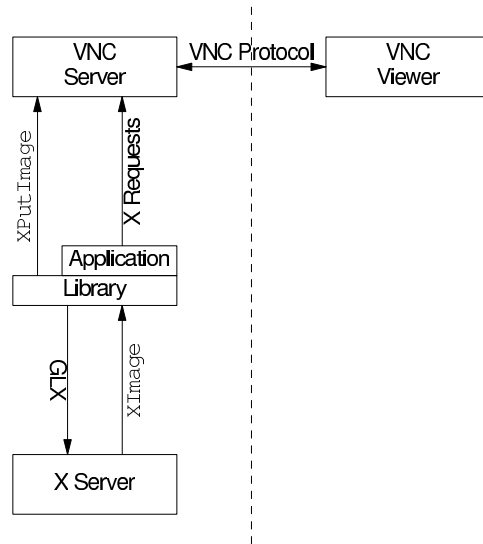


Abbildung 5: Prinzip der glxforker-Bibliothek (aus [Ert02])

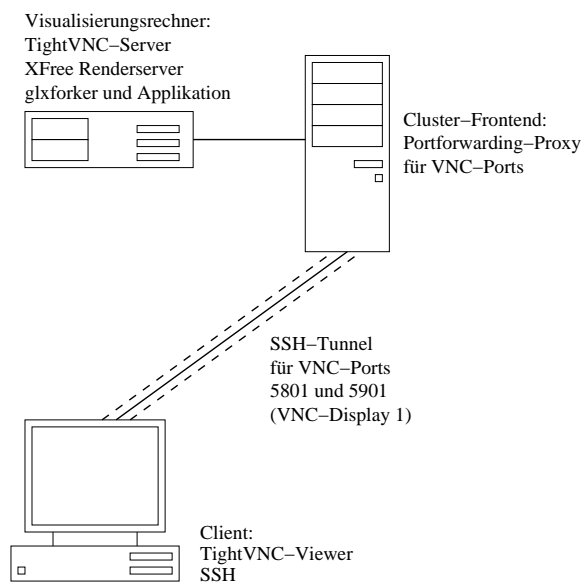


Abbildung 6: Setup der Remote-Visualisierung auf Kepler

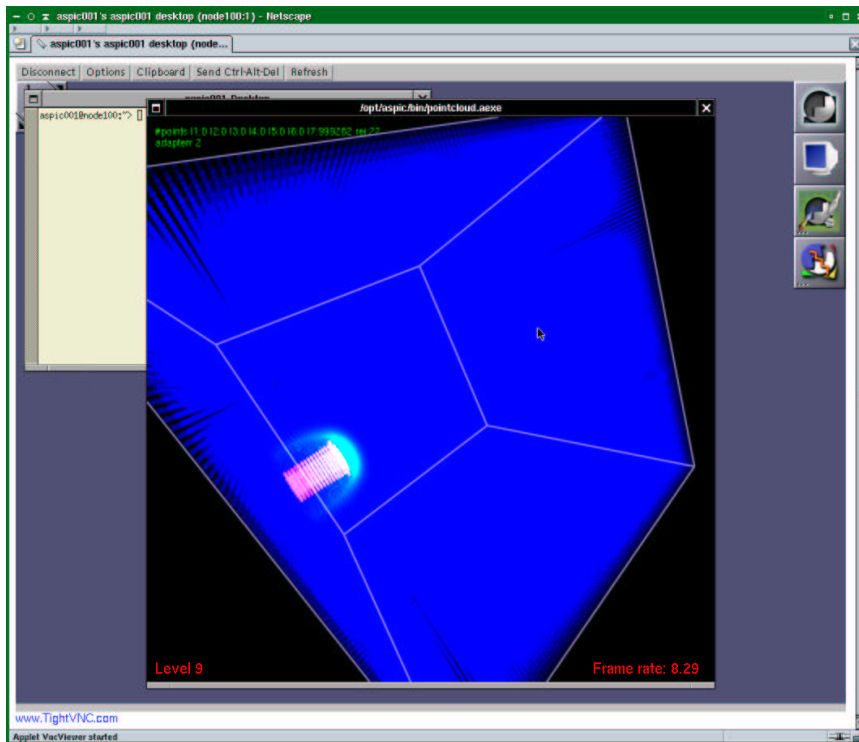


Abbildung 7: Remote-Visualisierung eines SPH-Datensatzes (Dieseleinspritzung) im VNC-Viewer-Java-Applet

Abbildung 7 zeigt die Remote-Visualisierung eines SPH-Datensatzes im VNC-Viewer. Auf dem Client-Rechner, der nicht hardwarebeschleunigt rendern kann, erreicht man maximal etwa 0.2 fps. Direktes hardwarebeschleunigtes Rendern erreicht ca 14 fps, bei der Remote-Visualisierung sind es im Durchschnitt 9 fps. Bei einer Internetanbindung über T-DSL erreicht man immerhin noch etwa 3 fps.

7 MultiCluster

Bietet ein Cluster nicht genügend Ressourcen für einen Job, so wäre ein Load-Balancing über mehrere gekoppelte Cluster wünschenswert. Für eine solche Kopplung gibt es mehrere Ansätze:

Grid-Toolkits bieten eine "Meta-Queueing"-Funktionalität und verwalten neben der Last auch das Verteilen der benötigten Daten für einen Rechenlauf und berücksichtigen dabei auch Sicherheitsaspekte, wie Authentifizierung und Verschlüsselung. Spezielle Kommunikationsbibliotheken ermöglichen es, parallele Applikationen auf mehrere Cluster gleich-

zeitig zu verteilen.

LSF-MultiCluster koppelt mehrere Cluster zu einem MultiCluster. Das hat einerseits den Nachteil, dass auf allen Clustern LSF vorhanden sein muss, bringt andererseits jedoch den Vorteil, dass ein an LSF gewöhnter Benutzer sich nicht umstellen muss. Es steht im einfachsten Falle eine weitere Queue, die für transferierbare Jobs zuständig ist, zur Verfügung. Auch das ASPIC-System ist einfach auf MultiCluster aufsetzbar. Um die Datenübertragung abzusichern, sollte, solange die Cluster nicht in einer Institution stehen und vor Angreifern durch eine Firewall geschützt werden, eine VPN-Kopplung, etwa über IPSEC, erwogen werden. Diese Konfiguration wird im moment zwischen Kepler und einem kleinen Cluster der science+computing AG erprobt.

Eine weitere Möglichkeit ist, den in Abschnitt 5 erwähnten ASPIC-Controller ebenfalls auf dem ASPIC-Server laufen zu lassen. Hierzu verfügt der ASPIC-Controller über einen einfachen internen XMLRPC-Server, über den beim Jobsubmit transferierbare Jobs angemeldet werden können. Die weitere Jobkontrolle übernimmt der ASPIC-Controller analog zu flowGuide.

8 Zusammenfassung

In dieser Veröffentlichung beschreiben wir, wie das ASPIC-System Ressourcenintensive Anwendungen auf sichere Art und Weise dem Benutzer über einen einfachen Web-Browser zur Verfügung stellt. Die ASPIC-Dienste können alternativ mit Hilfe von XMLRPC in eigene Softwarepakete integriert werden, wie wir am Beispiel von flowGuide, einem Workflow-Management-Tool der science+computing AG, aufzeigen. Wir beschreiben die Integration von Remote-Visualisierung in das ASP-Angebot, die sich in Tests als funktional und brauchbar erwiesen hat.

9 Ausblick

Die Einbindung von Grid-Toolkits wie Globus oder Avaki oder weiterer Queueing-Systeme ist relativ einfach möglich ist, da die ASPIC-Bibliothek eine Abstraktion bietet, so dass lediglich ein Perl-Modul ausgetauscht werden muss. Ein weiterer Punkt wäre der Austausch von XMLRPC als Kommunikationsprotokoll gegen SOAP [Win00]. Da SOAP auch binäre Attachments erlaubt, könnte dann auch der eigentliche Datentransfer mit SOAP durchgeführt werden.

Literatur

[All99] T. Dierks C. Allen. *The TLS Protocol Version 1.0*. Interner Bericht RFC 2246, The Internet Society, 1999.

[ASP] ASPIC. <http://www.tat.physik.uni-tuebingen.de/ASPIC/>.

[Ava] Avaki. <http://www.avaki.com>.

- [Ert02] S. Stegmaier M. Magallón T. Ertl. *A Generic Solution for Hardware-Accelerated Remote Visualization*. In: I.Navazo D. Ebert, P.Brunet, Hrsg., Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '02, 2002.
- [Glo] Globus. <http://www.globus.org>.
- [Gri] SUN GridEngine. <http://www.sun.com/software/gridware/>.
- [Kep] Kepler. <http://kepler.sfb382-zdv.uni-tuebingen.de/kepler/index.shtml>.
- [LSF] Platform LSF. <http://www.platform.com>.
- [sA] science+computing AG. <http://www.science-computing.de>.
- [Tig] TightVNC. <http://www.tightvnc.com>.
- [Tue01] I. Foster C. Kesselman S. Tuecke. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. International J. Supercomputer Applications, 15(3), 2001.
- [Win00] D. Box D. Ehnebuske G. Kakivaya A. Layman N. Mendelsohn H. Nielsen S. Thatte D. Winer. *Simple Object Access Protocol (SOAP) 1.1*. Interner Bericht, World Wide Web Consortium, 2000.
- [XML] XMLRPC. <http://www.xmlrpc.com>.