Entwicklung einer kompakten Visualisierungskomponente für die Cluster-Monitoring Werkzeuge Ganglia und Lemon

Matthias Bach

1. Dezember 2006

Zusammenfassung

Im Rahmen dieses Praktikums wurde eine Webanwendung entwickelt welche es ermöglicht die wichtigstens Zustandsdaten jedes Knotens eines Clusters auf einen Blick zu erfassen. Durch eine zeitnahe Visualisierung ist es möglich Entwicklungen im Cluster zu beobachten. Die Programmarchitektur stellt hierbei sicher, dass die Last auf den Cluster und das Monitoringsystem durch die zusätzliche Visualisierung begrenzt ist.

Inhaltsverzeichnis

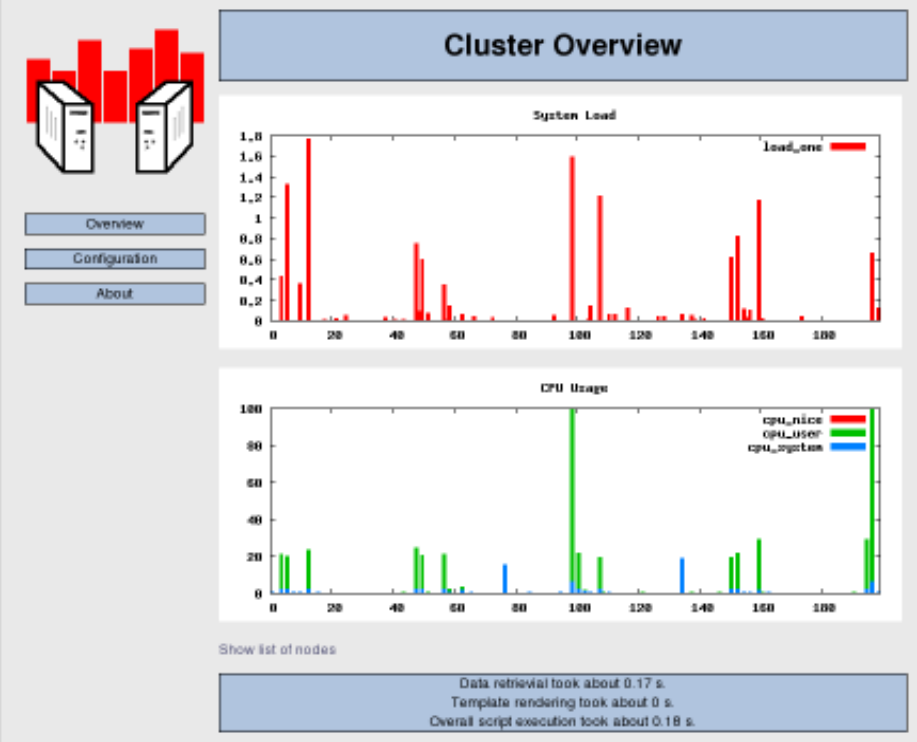

Abbildung 1: Übersichtsseite der Webanwendung

1 Problemstellung

Moderne Clustermonitoringsysteme wie Ganglia [2] und Lemon [1] sammeln während des Clusterbetriebs laufend Zustandsdaten. Diese Daten stellen sie per Webseite graphisch dar und über Datenschnittstellen nach außen zur Verfügung. Die Webseiten legen hierbei den Schwerpunkt auf die zeitliche Entwicklung der Daten und stellen die Summe oder den Mittelwert der Daten aller Knoten über einer Zeitachse dar. Dies erlaubt es zwar schnell zu sehen wann der Cluster wie ausgelastet war, man kann aber die Werte einzelner Knoten nicht schnell erfassen.

Ganglia bietet zwar die Möglichkeit für eine Metrik Graphen für jeden Knoten auf einer Seite darzustellen, dies hat aber mehrere Nachteile. Die Graphen aller Knoten passen nicht auf einen Bildschirm, man kann also immer nur einzelne Knoten beobachten. Auch lassen sich die Knotenwerte für den aktuellen Zeitpunk schlecht ablesen. Lemon fehlt ein solches Feature momentan komplett. Hier ist es lediglich möglich pro Knoten die Graphen mehrer Metriken zu sehen. In beiden Fällen ist es schwierig die Werte verschiedener Knoten zu beobachten.

Aufgrund der oben genannten Einschränkungen erlauben es die Darstellungen von Lemon und Ganglia nicht auf einen Blick zu sehen ob die Resourcen aller Rechner gleichmäßig genutzt werden und Ausreißer zu erkennen.

Um diese Probleme zu lösen sollte im Rahmen des Praktikums eine Anwendung entwickelt werden welche es erlaubt für die wichtigsten Metriken die Daten aller Knoten nebeneinander darzustellen. Diese Darstellung ist in Abbildung 1 zu sehen. Um den Overhead gering zu halten sollten die Daten aus dem sowieso laufenden Clustermonitoringsystem geholt werden. Da allerdings noch keine Entscheidung über das später verwendete Monitoringsystem gefallen ist sollte die Anbindung möglichst flexibel gestaltet werden.

2 Milestones

Um der Aufgabenstellung gerecht zu werden wurden zunächst folgende Milestones definiert:

- Evaluation von Monitoring- und Visualisierungstools.

  In diesem Abschnitt wurde analysiert welche Schnittstellen Ganglia und Lemon zur Verfügung stellen. Außerdem wurde nachgeforscht welche Tools und Libraries die gewünschten Graphen erstellen können.

- Programmarchitektur

  Mithilfe der Ergebnisse des vorherigen Abschnittes wurde eine Abstraktionsschicht über Ganglia und Lemon definiert und Entscheidungen bezüglich Platform, Programmiersprache und des Visualisierungstools getroffen.

- Anbindung von Ganglia

  Implementierung und Test eines Moduls zum Auslesen der Daten aus Ganglia.

- Anbindung von Lemon

  Implementierung und Test eines Moduls zum Auslesen der Daten aus Lemon.

- Visualisierung

  Implementierung der Visualisierung der ausgelesenen Daten.

3 Clustermonitoring-Systeme

Beide Monitoringsysteme sammeln ihre Daten durch Agenten welche auf den einzelnen Knoten des Clusters laufen und die Daten von diesen Auslesen. Diese werden dann auf einem oder mehreren Servern zur Speicherung und anschließenden Analyse zusammengeführt. Es gibt allerdings unterschiede darin wie diese Daten propagiert werden.

Ganglia nutzt ein Broadcast-Protokoll bei dem jeder Knoten stets den Zustand aller Knoten des Clusters kennt. Der Zentrale Server fragt in regelmäßigen Intervallen einzelne Knoten nach den aktuellen Daten. Hierzu öffnet er eine TCP/IP-Verbindung zu diesem Rechner und bekommt die Daten in Form eines XML-Dokuments zurück. Diese speichert er anschließend im RRD-Tool, von wo sie dem Benutzerinterface zur Verfügung stehen. Genau wie die Clusterknoten selbst stellt auch er die gesammelten Daten wieder als XML, welches via TCP/IP-Verbindung abgefragt werden kann, zur Verfügung.

Bei Lemon hingegen sammelt jeder Knoten nur die einzelnen Daten und kontaktiert den Server welcher die Daten sammelt über SOAP. Dieser sammelt die Daten entweder in einer Flatfilestruktur oder in einer Oracle-Datenbank. Das RRD-Tool und das Benutzerinterface lesen die Daten aus dieser Ablage aus. Für externe Anwendungen steht eine SOAP-basierte Bibliothek zur Verfügung welche die Abfrage der Daten aus der Datenbank erlaubt. Diese Schnittstelle hat allerdings einige Performanceprobleme und wird laut Aussagen der Lemonentwickler Ende September durch eine neue XML-basierte Schnittstelle ersetzt.

4 Programmarchitektur

Da die bestehenden nativen Visualisierungen der beiden Clustermonitoringsysteme mithilfe von PHP auf einem Apache-Server realisiert sind wurde diese Platform auch für diese Visualisierung gewählt. Dies bietet den Vorteil, dass keine zustätzliche Software installiert und gepflegt werden muss. Außerdem stehen so bereits nutzbare Bibliotheken zur Verfügung und es muss keine Zeit auf das Schreiben von Sprachanbindungen verwendet werden. Da Objekte in PHP-Versionen vor PHP5 einige Probleme bezüglich Performance haben, und das Objektverhalten sich zwischen PHP4 und PHP5 ändert, wurde in den speziell für dieses Programm geschriebenen Teilen auf Objekte verzichtet. Das Programm ist aber dennoch in einem objektorientierten Stil aufgebaut, bei dem jede PHP-Datei ein Singleton-Objekt darstellt.

Um die Anwendung sowohl wartbar als auch leicht erweiterbar zu halten wurden ist die in der Abbildung 2 dargestellte Architektur möglichst modular gehalten. Sie folgt dem Unixprinzip der kleinen Anwendungen für kleine Aufgaben welche dann über Skripte zusammengebunden werden.

Um die auf dem System und dem Cluster verursachte Last durch das Monitoring nach oben zu begrenzen beschränkt die Anwendung die Häufigkeit der Refreshs vom Cluster und des Erstellens der Graphiken. Hierzu wird in der Konfiguration ein Zeitintervall festgelegt. Wurden die Daten innerhalb des Intervalls

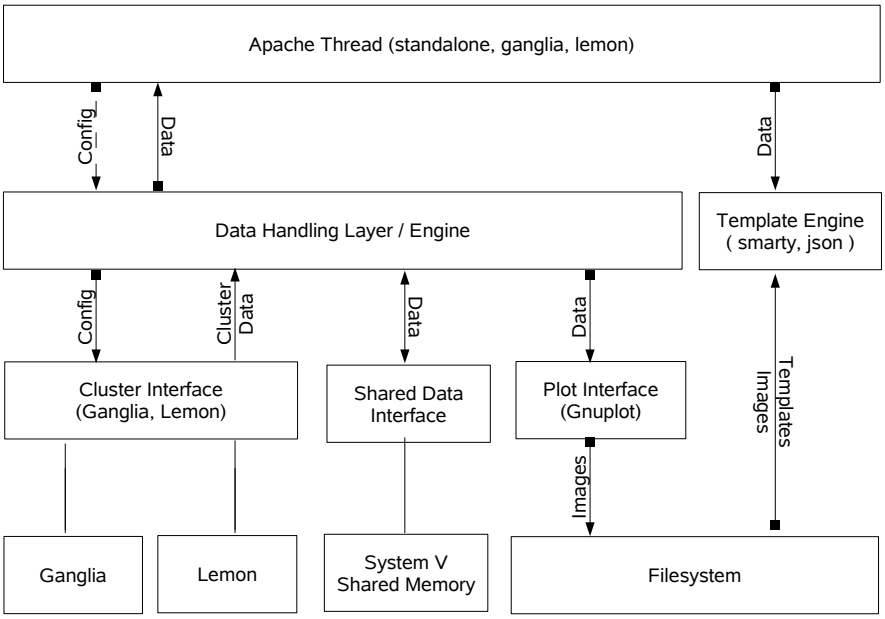

Abbildung 2: Architekturübersicht

bereits aktualisiert werden die alten Daten und Graphiken zurückgegeben. Nur wenn dies nicht der Fall war werden die Daten über das entsprechende Plugin vom Cluster geladen und über das Plotplugin neue Graphen gerendert. Da ein Apache mehrere Anfragen parallel beantworten kann wird dieser Vorgang über einen gemeinsamen Datenspeicher synchronisiert.

In der zweiten Ebene der Abbildung sieht man gut die grundsätzliche Seperation von Daten- und Darstellungsschicht. Dies ermöglicht es einerseits die Daten auf verschiedene Art und Weise zur Verfügung zu stellen, z.B. als HTML oder im JSON-format [5] zur automatisierten Abfrage, ermöglicht aber vor allem an Daten- oder Darstellungsschicht zu arbeiten ohne hierbei Schäden am jeweils anderen Programmteil hervorzurufen. Erst der Apache-Thread welcher das HTTP-Request abarbeitet bringt diese beiden Teile zusammen, theoretisch könnte er aber auch durch eine lokal laufende Anwendung ersetzt werden, was während der Entwicklung zu Testzwecken auch genutzt wurde.

Zur Erzeugung der HTML-Seiten wird mit Smarty [7] die am weitesten verbreiteste Template-Engine für PHP eingesetzt. Werden die Daten im JSON-Format angefordert wird Services_JSON [6] genutzt, da dies nicht die zusätzliche Installation eines nativen JSON-Bibliothek benötigt. Aufgrund der lockeren Bindung zwischen Daten und Darstellung lässt sich dies aber auch leicht an eine native Bibliothek mit besserer Performance anpassen. Prinzipiell ist beim einfachen Benutzerinterface der Overhead durch das Laden der ganzen Seite zusätzlich zu den Bildern vernachlässigbar. Die JSON-Repräsentation bietet aber die Möglichkeit die Graphiken via JavaScript z.B. in die Templates von Ganglia einzubinden ohne am Code von Ganglia Änderungen vorzunehmen.

Die Datenschicht übernimmt das Handling von Konfiguration, Monitoring-Plugins, gemeinsamer Datenspeicherung, und Plot-Plugins. Sie versucht stets

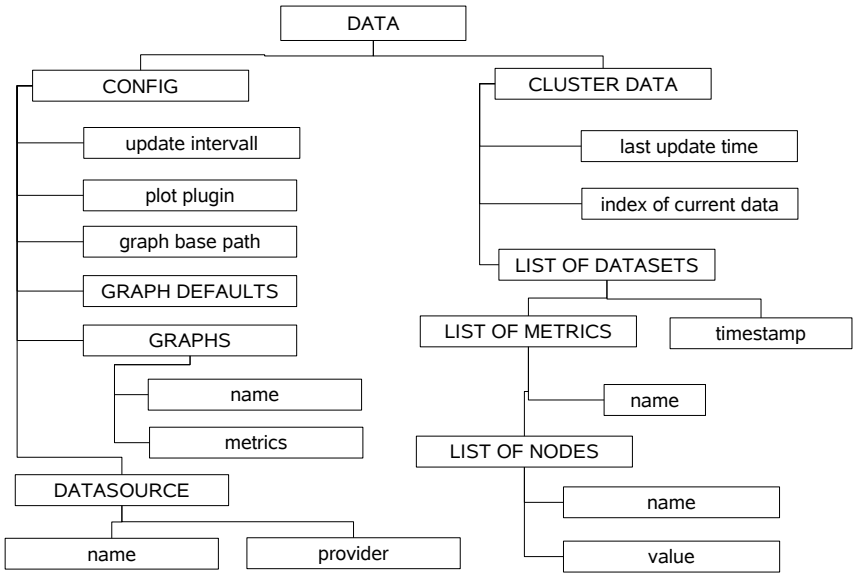

Abbildung 3: Vereinfachte Darstellung der Datenstruktur für Konfigurations- und Clusterdaten. Einträge in Großschrift stellen Arrays dar.

zunächst die Daten aus dem gemeinsamen Datenspeicher zu lesen. Findet sie dort keine Konfiguration vor wird diese von der Festplatte geladen. Falls die Daten noch aus dem vorherigen Intervall stammen wird das konfigurierte Datenplugin geladen und darüber die Daten vom Cluster gezogen. Anschließend wird der gemeinsame Datenspeicher aktualisiert, das Plotplugin geladen und damit die neuen Daten geplottet.

Als gemeinsamer Datenspeicher wird System-V-Shared-Memory verwendet. Dieses hat den Vorteil dass man PHP-Objekte darin direkt ablegen kann. Lediglich eine Serialisierung der Daten ist notwendig, welche von PHP aber sprachintern abgearbeitet wird. Dies passt sehr gut mit der verwendeten Datenstruktur zusammen, in dieser werden alle Daten des Programms in einem multidimensionalen assoziativen Array abgelegt. Auch verzichtet man so auf Festplattenzugriffe und es müssen keine zusätzlichen Programme auf dem Rechner laufen. In der Standareinstellung ist das Shared-Memory-Segment nur 250 Kilobyte groß, so dass der verwendete Speicher keine Problem darstellt. Ein Alternative wäre die Ablage der Daten in einer SQLite-Datebank gewesen. Da diese aber dateibasiert ist hätte man definitiv einen größeren Overhead beim Zugriff auf die Daten gehabt. Außerdem ist SQLite erst seit PHP5 ohne zusätzliche Installation in PHP verfügbar. Aufgrund dieser beiden Nachteile wurde das Shared-Memory bevorzugt dessen einziger Nachteil ist auf Windows nicht verfügbar zu sein. Da diese Plattform aber sowieso nicht im Scope der Anwendung liegt kann dies hier getrost ignoriert werden.

Da es das Handling mit dem Shared-Memory vereinfacht werden alle Daten in einer gemeinsamen, in Abbildung 3 vereinfacht dargestellten, Baumstruktur verwaltet. So können alle Daten auf einmal ins Memory geschoben und daraus

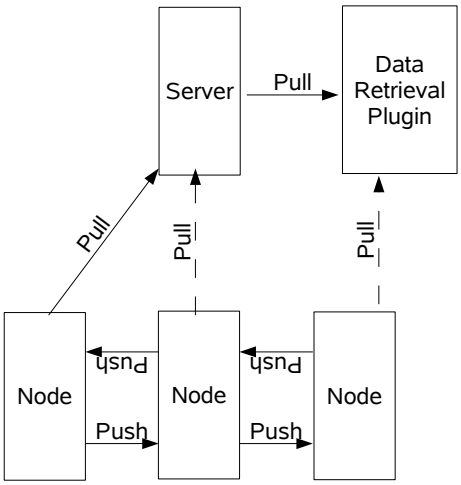

Abbildung 4: Ganglia Data Retrieval

gelesen werden. An einzelne Programmteile kann dann jeweils der benötigte Unterbaum weitergegeben werden.

Die Monitoring-Plugins werden Abhängig von der Konfiguration geladen, die darin enhaltenen, nach Konvention benannten, Funktionen werden anschließend per Reflection aufgerufen. Zuvor wird aus den Graphendefinitionen eine Liste aller benötigten Metriken extrahiert. Das Plugin hängt anschließend an jede Metrik eine Liste aller Knoten mit den dazugehörigen aktuellen Werten.

Auch das Plotplugin wird abhängig von der Konfiguration geladen und die Funktionen per Reflection aufgerufen. Das Plugin liest aus den Graphendefinition die Metriken aus, erhält darüber die Wertelisten und gibt diese Daten in das Graphenverzeichnis aus.

5 Anbindung von Ganglia

Für die Anbindung an Ganglia wird dessen XML-Schnittstelle genutzt. Wie in Abbildung 4 zu sehen ist senden sich die einzelnen Knoten eines mit Ganglia überwachten Clusters ständig ihren aktuellen Zustand per Broadcast zu. So kennt jeder Knoten stets den Zustand des gesamten Clusters und stellt diesen über TCP als XML zur Verfügung. Der zentrale Ganglia-Server kontaktiert regelmäßig einen der Knoten um seine Datenbank zu füllen und diese Daten wiederum per TCP als XML zur Verfügung zu stellen.

Das Ganglia-Plugin kontaktiert nun, je nach Konfiguration, entweder den zentralen Server oder einen der Knoten und liest das XML aus. Dazu wird der in PHP enthaltene SAX-Parser verwendet. Hierbei schreibt es für jeden Knoten den aktuellen Wert und den Namen des Knotens in die Liste der konfigurierten Metriken. Wird eine Metrik von keinem Graph benötigt wird sie ignoriert um Speicherplatz zu sparen.

6 Anbindung von Lemon

Die Anbindung an Lemon ist leider etwas komplizierter, da die alten SOAP-basierten APIs welche Lemon zur Verfügung stellt vor der Ablösung stehen. Leider stehen die neuen APIs noch nicht zur Verfügung, weshalb diese nicht zur Implementierung genutzt werden konnten. Aufgrund des Pluginsystems sollte es jedoch, sobald die neue API zur Verfügung steht, ohne große Probleme möglich sein diese einzubinden.

Anstatt die alte, sogenannte Simplified Repository API, zu benutzen, welche recht kompliziert zu installieren ist, entschied ich mich eine inoffizielle, vom Lemon Webinterface genutzte, API zu benutzten. Diese API greift intern direkt auf die von Lemon genutzte Flatfile- bzw. Oracle-Datenbank zu. Da diese API auch vom UI genutzt wird kann davon ausgegangen werden, dass diese auch an künftige Datenbankänderungen angepasst wird, was bei einer eigenen Implementierung nicht sichergestellt wäre.

Das Lemon-LRF genannte Plugin setzt voraus, dass das Lemon RRD Framework, welches das Lemon GUI enthält, auf dem gleichen Rechner installiert ist. Allerdings ist es nicht notwendig den lemonrrd tatsächlich laufen zu lassen. Das Plugin nutzt die Methode get_MR_instance() aus der Datei mr_load.php um eine das Repository representieren Klasse zu bekommen. Besonders schön an dieser Lösung ist, dass die Konfiguration aus der config.php von Lemon gelesen werden kann. Dadurch muss, bei korrekt konfiguriertem Lemon, in der Anwendung nur noch der Pfad zu Lemon gepflegt werden. Bei Verwendung der offiziellen API müsste die komplette Konfiguration doppelt gepflegt und umständlich über Umgebungsvariablen weitergegeben werden. Der einzige Trick bei der aktuellen Lösung besteht hingegen in einer temporären Änderung des Arbeitsverzeichnisses damit das Laden weiterer PHP-Dateien innerhalb von Lemon funktioniert.

Eine von Lemon zur Verfügung gestellte Klasse bietet eine Funktion welche es erlaubt für eine Menge von Metriken die Werte aller Knoten zu erhalten. Genau wie bei der Anbindung an Ganglia werden dann an jede Metrik eine Liste mit Namen und Werten der Knoten gehängt.

7 Visualisierung der Daten mit Gnuplot

Zur Visualisierung wurde Gnuplot [4] gewählt, da dieses mit guter Performance ansprechende Graphiken erzeugt. Um Gnuplot von PHP aus nutzen zu können wurde eine modifizierte Version von PHP-GNUPlot [3] verwendet. Die Verfügbarkeit dieser Bibliothek war ein große Vorteil gegenüber ROOT, für welches erst eine PHP-Anbindung hätte entwickelt werden müssen.

PHP-GNUPlot startet aus dem PHP-Skript heraus Gnuplot in einem neuen Prozess, wobei es einen Filehandle auf STDIN des neuen Prozesses hält. Dies ermöglicht es Gnuplot programmatisch genau so wie auf der Konsole zu verwenden. Hierduch erspart man sich insbesondere das umständliche erzeugen von Dateien welche den Plotablauf enthalten.

Die originale Version von PHP-GNUPlot schrieb die bereits im Speicher gehaltenen Daten zunächst in temporäre Dateien welche anschließend von Gnuplot ausgelesen wurden. Um eine Bilddatei zu speichern wurden die Daten erst auf das Default-Terminal ausgegeben und anschließend mit dem neuen Terminal ein replot ausgeführt. Um Festplattenzugriffe zu sparen und die Anzahl der

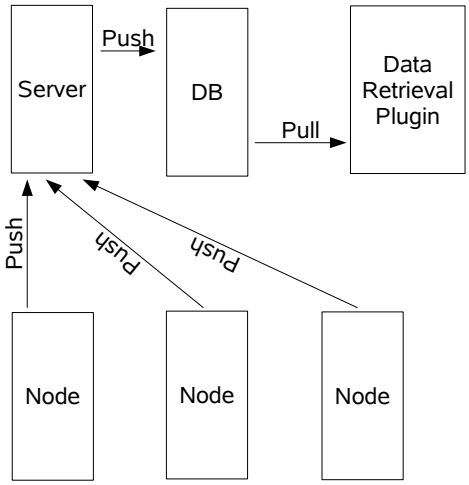

Abbildung 5: Lemon Data Retrieval

vom Webserver benötigten Schreibberechtigungen zu minimieren wurde PHP-GNUPlot so weiterentwickelt, dass es nun auch die Daten über STDIN direkt an Gnuplot weitergibt. Außerdem würde es insofern verändert, dass es nun möglich ist das Terminal zu setzen bevor man die Daten plottet. Dies beschleunigte den Plotvorgang, wie in Tabelle 1 zu sehen, um einen Faktor 2 (Intel P-III) bis 4 (Intel Centrino).

Aktuell plottet das Plugin lediglich den aktuellen Datensatz. Theoretisch könnten auch mehrere Datensätze aus aufeinanderfolgenden Intervallen dargestellt werden, mit Gnuplot ist hierbei jedoch leider keine ansprechende Darstellung möglich. ROOT bietet hierfür die schöne Möglichkeit zweidimensionale Histogramme zu plotten. Allerdings müsste man ausprobieren ob man bei diesen wirklich noch schnell erfassen kann was dargestellt wird.

Da einige Daten in den Clustermonitoringsystemen über mehrere Metriken verteilt dargestellt werden ist es wichtig mehrere Metriken in einem Graph kombinieren zu können. Ein Beispiel hierfür ist die CPU-Auslastung, bei welcher in den Clustermonitoringsystemen User, System und Nice unterschieden wird. Sollen in einem Graph alle drei dargestellt werden, so werden diese übereinander dargestellt. Da Gnuplot dies leider nicht durch einen speziellen Plotstyle untersützt wird der normale Block-Plottstyle verwendet. Auf die zuoberst darzustellende Metrik werden die anderen Metriken aufaddiert und diese anschließend vor die addierte Metrik geplottet, so dass sie den hinzuaddierten unteren Anteil des Balkens der oberen Metrik verdecken. Durch verschiedene Farben der einzelnen Metriken ist es so möglich sowohl die Verteilung der Metriken zueinander als auch ihren additiven Gesamtwert auf einen Blick zu sehen. Ein Beispiel für einen solchen Graphen ist in Abbildung 6 zu sehen.

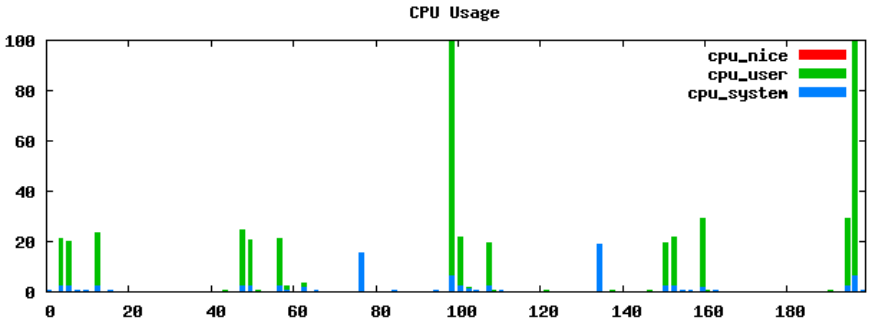

Abbildung 6: CPU-Nutzung durch User, Nice und System in einem Graph

	Original	Nur ohne Replot	Diese Projekt
Pentium III 800 MHz	25.4	9.87	9.79
Pentium M 1400 MHz	12.92	3.14	2.95

Tabelle 1: Laufzeiten des originalen PHP-GNUPlot und der für diese Projekt veränderten Version gemessen mit synthetischen Daten.

8 Zusammenfassung und Ausblick

Die entwickelte Lösung erfüllt das Ziel für einzelne Metriken die Werte aller Knoten auf einen Blick ersichtlich zu machen. Die Performance des Systems reicht hierbei selbst auf einem alten Intel P-III mit 800 MHz aus um bei 200 Knoten und vier Graphen ein Updateintervall von einer Sekunde zu nutzen ohne das System voll auszulasten. Bei Tests auf einem System auf dem auch der Lemonrrd lief war die zusätzlich Last durch die eigene Visualisierung nicht messbar. Der Lemonrrd verursachte bereits über 50% Systemlast. Die von Lemon gelieferten Lastwerte änderten sich nicht wenn die eigene Visualisierung gestartet wurde. Es ist allerdings möglich, dass der Lemonrrd dann weniger Systemleistung abbekam. Durch das eingebaute Throtteling ist die Last durch zusätzliche Nutzer nicht höher als die Abfrage einer PHP-Seite die statischen Content ausgibt.

Wenn die aktuelle Version der Lemon-APIs erscheint wäre es sinnvoll ein neues Lemon-Plugin zu schreiben welches dann die offizielle API nutzt um sich langfristig auf die genutzte Funktionalität verlassen zu können. Allerdings ist zu befürchten, dass die neue API, genau wie die alte, nicht den vollen Funktionsumfang liefern wird, so dass es auch langfristig sinnvoll sein kann die eigentlich lemoninterne Funktionalität zu nutzen.

Da es bereits ein PHP-Skript gibt welches die Namen der Graphendateien in JSON zurückgibt wäre auch eine Integration in die UIs von Lemon und Ganglia mit ein wenig Javascript-Coding ohne große Probleme möglich.

Darstellungstechnisch wäre es denkbar die Datensätze mehrer Intervalle in einem Graphen auszugeben. Solche Graphen können allerdings momentan nur mit ROOT ansprechend dargestellt werden. Sollte es in Zukunft eine PHP-Anbindung für ROOT geben wäre ein Plugin das mithilfe von ROOT solche Graphen zur Verfügung stellt sicher wünschenswert. Allerdings sollte man, wie bereits erwähnt, überprüfen ob sich diese Graphen dann überhaupt noch schnell

erfassen lassen.

9 Installationsanleitung

Bevor mit der eigentlichen Installation begonnen wird sollte sichergestellt sein, dass die verwendeten Applikationen auf dem System vorhanden sind. Ganglia oder Lemon sollten im Cluster installiert sein. Auf dem Server wird ein Webserver (z.B. Apache) der PHP4 interpretieren kann benötigt. PHP muss hierbei mit den Modulen sysvshm und sysvsem, die leider nur auf UNIX/Linux-System verfügbar sind, installiert sein. Desweiteren wird Gnuplot benötigt. Ein Gnuplot der Version 3 reicht, falls in den Diagrammen gefüllte Boxen verwendet werden sollen ist allerdings mindistens Version 4 notwendig.

In dieser Installationsanleitung wird davon ausgegangen, dass die Anwendung sich in der Datei `qco.tar.bz2` im Home-Verzeichnis befindet. Als Wurzel des Webservers wird `/srv/www` angenommen. Dies kann natürlich je nach Distribution und Konfiguration variieren, allerdings verändert dies lediglich die zur Installation notwendigen Pfade, erfordert aber keine Änderungen an der Konfiguration der Anwendung.

Zunächst muss das die Anwendung enthaltene Archiv entpackt werden. Dies kann zum Beispiel wie folgt erfolgen.

```
tar -xjf qco.tar.bz2
```

Das hierbei entpackte Archiv enthält neben der eigentlichen Webapplikation auch noch Dokumentation und Tests, welche für auf dem Server nicht benötigt werden. Um die Applikation über den Webserver verfügbar zu machen muss nur der Inhalt des Verzeichnisses `www` in ein Verzeichnis des Webservers kopiert werden.

```
cp -R qco/www /srv/www/qco
```

Da die Anwendung zur Laufzeit Dateien generiert müssen die Verzeichnisse `configs`, `templates_c` und `graphs` für den Webserver schreibbar sein. Geht man davon aus, dass der Webserver unter dem User `www` läuft kann dies wie folgt geschehen. Der gegebene Befehl sorgt auch dafür, dass alle anderen Dateien weiterhin lesbar sind. Um sicherzustellen, dass die Gruppe `root` weiterhin Dateien schreiben kann sollte man allerdings `root` auch noch Schreibberechtigungen geben. Für eine möglichst sichere Konfiguration sollte man allerdings die Schreibberechtigung des Server-Users auf die drei angegebenen Verzeichnisse beschränken sowie allen anderen Benutzer alle Zugriffsrechte nehmen.

```
chown -R www:root /srv/www/qco
chmod -R g+w /srv/www/qco
```

Hiermit ist die Installation abgeschlossen und die Anwendung muss nur noch konfiguriert werden. Dies geschieht über die Datei `baseconfig.php` deren kompletter Pfad im Beipiel `/srv/www/qco/configs/baseconfig.php` ist. Die Datei `baseconfig.php` ist schon für die Verwendung einer auf dem gleichen Rechner laufenden Ganglia-Installation vorkonfiguriert. Läuft der Ganglia-Server auf einem anderen Rechner oder nicht auf dem Standard-Port ist die

Variable `$BASECONFIG['DATASOURCE']['PROVIDER'][0]['name']` und eventuell auch die Variable `$BASECONFIG['DATASOURCE']['PROVIDER'][0]['port']` anzupassen. In dieser Datei können auch weitere Graphen definiert werden. Die Definition erfolgt analog zu den bereits definierten Beispielgraphen. Zu jedem Graph müssen ein Titel, eine Metrik und ein Dateiname für das Bild angegeben werden. Sollen mehrere Metriken in einen Graph geplottet werden müssen ihre Namen mit `|` konkatiniert werden.

Soll anstelle von Ganglia Lemon zur Datenbeschaffung verwendet werden so sollte man die Datei `baseconfig.php` durch die Datei `baseconfig_lemon.php` ersetzen, diese ist bereits für die Verwendung von Lemon vorkonfiguriert. Wichtig ist hierbei, dass die Variable `$BASECONFIG['DATASOURCE']['PROVIDER'][0]['name']` auf das Wurzelverzeichnis des Lemon-Web-UIs zeigen muss, also z.B. `/srv/www/lrf`. Das Lemon-Web-UI muss konfiguriert sein, da die Konfiguration automatisch aus diesem ausgelesen wird. Die Namen der Metriken sind in der Datei `metric_map.php` des Lemon-UIs definiert.

Generell sind weitere Informationen zur Konfiguration im Verzeichnis `doc` des Installationsarchives zu finden.

Literatur

[1] Lemon: `http://lemon.web.cern.ch/lemon/index.shtml`

[2] Ganglia: `http://ganglia.sourceforge.net`

[3] PHP-GNUPlot: `http://php-gnuplot.sourceforge.net`

[4] Gnuplot: `http://www.gnuplot.info`

[5] JSON: `http://json.org`

[6] Services_JSON: `http://pear.php.net/pepr/pepr-proposal-show.php?id=198`

[7] Smarty: `http://smarty.php.net`