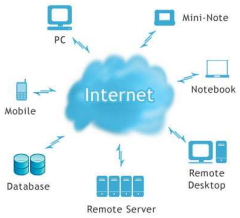


Up up to the Sky

Cloud Computing für Embedded Systems mit Amazon Webservices

Dr. Claus Kühnel, Daniel Zwirner

http://www.ckuehnel.ch/PDF/Up up to the Sky.pdf

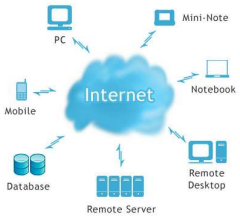
1. Vorbemerkung

Cloud Computing ist in aller Munde und soll durch konsequente Weiterentwicklung der Virtualisierung der IT-Infrastruktur die Informationstechnologie revolutionieren. Zahlreiche Anbieter, wie Microsoft, Google, Amazon u.a., investieren derzeit stark in den Ausbau von global verteilten Datenzentren mit großen Server-Kapazitäten. In einer leistungsstarken Infrastruktur werden verschiedene Dienste zur Verfügung gestellt. Die Geschäftsmodelle reichen vom reinen Storage Providing bis hin zur Bereitstellung von Applikationen.

Schon heute kann Software als Service aus dem Netz bezogen werden. Zu nennen sind hier u.a. Officeanwendungen von Google und Zoho, Online Bildbearbeitungsprogramme, Social Networking wie Facebook, NetWeaver von SAP bis hin zur CRM-Anwendung von Salesforce.

Cloud Computing erlaubt Unternehmen IT-Kapazitäten aller Art ins Netz zu verlagern. Wie schon beim klassischen IT-Outsourcing können ökonomische Gründe für das Cloud Computing geltend gemacht werden. Die Bedenken betreffen hauptsächlich die Sicherheit und Verfügbarkeit der Daten, da diese zwangsläufig die Unternehmensgrenzen verlassen.

Für das Cloud Computing spricht die Ineffizienz der gegenwärtigen IT, die heute hohe Wartungs- und Personalkosten aufweist. Außerdem erhofft man sich vom Cloud Computing, mehr Beweglichkeit in die Unternehmens-IT zu bringen, die heute der Unternehmensrealität oft hinterher hinkt. [1][2][3][4][5]



2. Anatomie des Cloud Computing

Mit anderen Worten: Daten und Anwendungen befinden sich nicht mehr auf dem lokalen Rechner, sondern – metaphorisch gesprochen – in einer Wolke (Cloud), womit eine Vielzahl von entfernten Datenzentren gemeint ist, die über das Internet miteinander vernetzt sind. Als Nutzer kann man auf diese Weise überall dort auf seine Daten und Anwendungen zugreifen, wo man eine Internetverbindung hat (Abbildung 1).

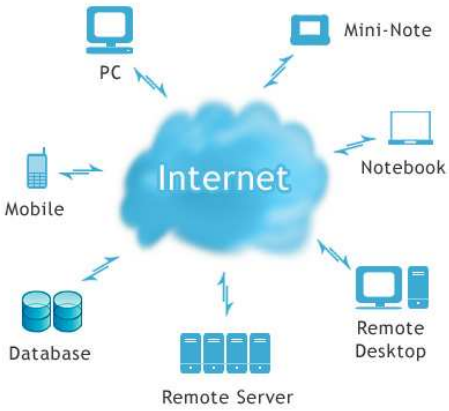


Abbildung 1 Cloud Computing Network

Bewegen wir uns in das Innere der Wolke, dann finden wir dort nicht nur einen Service, sondern wie Abbildung 2 zeigt eine Sammlung verschiedener Services [6][7].

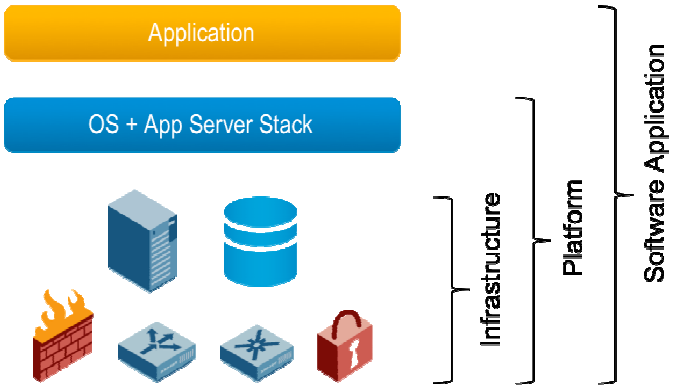
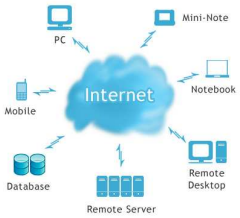


Abbildung 2 Cloud Computing Services

Der unterste Schicht des in Abbildung 2 dargestellten Schichtenmodells besteht in der Infrastruktur (*Infrastructure-as-a-Service, IaaS*). IaaS bezeichnet die Nutzung von Infrastuktur bestehend aus Computer- und Speicherressourcen, oder weitergefasst die Möglichkeit einer



über die Nutzung eines Computers oder Datencenters mit garantierter Performance und Bandbreite für Speicher- und Internetzugriff hinausgehenden Einsatzes dieser Infrastuktur eine beliebige Software auf einem Betriebssystem auszuführen.

Oberhalb der Infrastruktur ist die Plattformschicht angeordnet (*Platform-as-a-Service, PaaS*), die ein Betriebssystem und weitere erforderliche Services für bestimmte Applikationen hinzufügt. Beispielsweise kann PaaS zusätzlich zu virtuellen Servern und Speichern ein Betriebssystem und Application Sets als Virtuelle Maschine (VM) bereitstellen.

Über der Plattformschicht ist die Applikationsschicht angeordnet (*Software-as-a-Service, SaaS*), die Software zur Nutzung anbietet, für die nur entsprechend ihrer Nutzung zu zahlen ist.

Die in Abbildung 2 gezeigte Sicht verdeckt einige weitere Aspekte bzw. Services, wie z.B. die (alleinige) Datenspeicherung (*data-Storage-as-a-Service, dSaaS*). dSaaS stellt Datenspeicher zur Verfügung, bei dem die Kosten wiederum nur durch Speicherkapazität und Bandbreite bestimmt werden.

Einen sicher kaum vollständigen Überblick über die bereits verfügbaren Services für das Cloud Computing zeigt Abbildung 3.

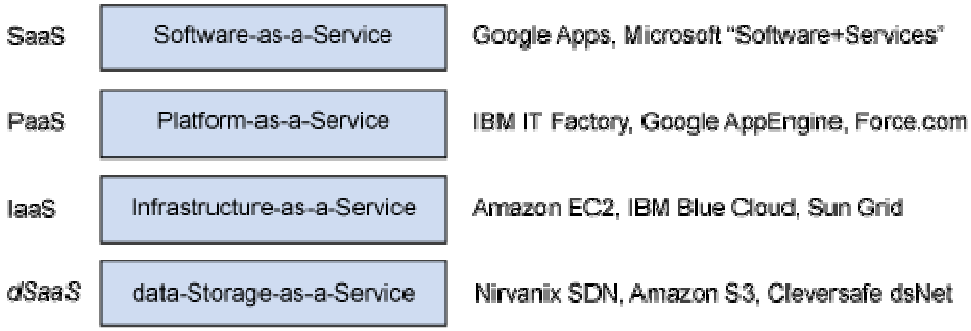
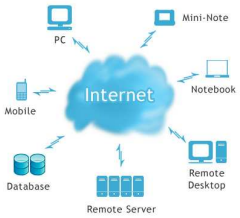


Abbildung 3 Verfügbare Services für das Cloud Computing (Auszug)



3. Top 10 Cloud Computing Service Providers

Ein Überblick über die von verschiedenen Firmen angebotenen Dienstleistungen zeigt Tabelle 1. Dabei ist das eigentliche Ranking eher unerheblich, aber gemäß [8] der erste Platz unumstritten.

Ranking	*Anbieter*	*Dienstleistung*
1	**Amazon**	**AWS - IaaS**
2	Google	Google Apps - SaaS; App Engine - PaaS
3	VMWare	vCloud
4	Rackspace	Rackspace Cloud- IaaS
5	Salesforce.com	Leader in SaaS
6	Microsoft	Azure - PaaS
7	Joyrent	SaaS, PaaS, IaaS - Facebook (u.a.
8	IBM	Blue Cloud
9	NetSuite	SuitCloud - SaaS
10	3Tera	CloudWare - open framework

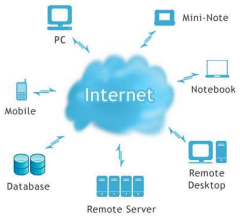
Tabelle 1 Cloud Computing Service Provider

4. Amazon Web Services

Seit Anfang 2006 unterstützt Amazon mit seinen Amazon Web Services (AWS) Anwender mit einer IT-Infrastuktur "in der Wolke". Mit AWS können Rechenleistung, Speicher und andere IT-Services durch Zugriff auf eine Suite "elastischer" IT-Infrastukturservices nach Bedarf bezogen werden. Mit AWS kann die Amazon.com's globale IT-Infrastruktur genutzt werden, die die Basis des Amazon.com's USD 15 Mrd. Versandgeschäfts und der damit verbundenen Logistik darstellt - einer IT Infrastruktur, die skalierbar, zuverlässig und verteilt seit mehr als 13 Jahren ihren Dienst tut.

AWS ist mehr als eine Sammlung verschiedenere Infrastrukturservices. Alle Leistungen von AWS stehen ohne Initialkosten und ohne das Eingehen längerfristiger Verbindlichkeiten zur Verfügung. Verrechnet werden nur die in Anspruch genommenen Leistungen. Das angebotene Leistungsspektrum und die einzugehenden Verbindlichkeiten waren ein erstes Entscheidungskriterium für den Einsatz der AWS.

Die folgende Übersicht (Tabelle 2) zeigt die derzeit verfügbaren Komponten der AWS [9]. Die Sicherheitsaspekte aller AWS sind in einem separaten White-Paper [10] beschrieben.



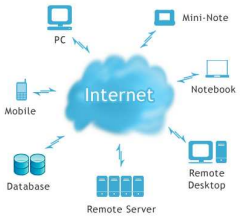
Amazon Elastic Compute Cloud (Amazon EC2™)	Dieser Webservice stellt mit Hilfe einer konfigurierten Amazon Machine Instance (AMI) skalierbare Rechenleistung zur Verfügung und kann damit an die Leistungsanforderungen angepasst werden.
Amazon Simple Storage Service (Amazon S3™)	Dieser Webservice kann zum Speichern grosser Datenmengen verwendet werden, auf die zu jeder Zeit von jedem Ort aus zugregriffen werden kann. Als Entwickler hat man so Zugang zur gleichen hoch skalierbaren, zuverlässigen, schnellen und preiswerten Datenspeicher-Infrastuktur, die Amazon für eigene Zwecke nutzt.
Amazon Cloud-Front™	Dieser Webservice unterstützt die Verbreitung von Dokumenten und Multimediadaten. Zusammen mit anderen Amazon Web Services erhalten Entwickler einen einfache Möglichkeit digitale Inhalte an Enduser mit niedrigen Latenzzeiten und hoher Gewchwindigkeit zu verteilen.
Amazon SimpleDB™	Dieser Webservice ermöglicht Abfragen auf strukturierte Daten in Echtzeit durch eine enge Verbindung zu Amazon Simple Storage Service (Amazon S3) und Amazon Elastic Compute Cloud (Amazon EC2). Amazon SimpleDB stellt einfach zu nutzende Kernfunktionen einer Datenbank zur Verfügung.
Amazon Simple Queue Service (Amazon SQS™)	Dieser Service ist eine zuverlässige, hoch skalierbare Warteschlange zum Speichern von Messages, die zwischen Computern ausgetauscht werden. Durch die Verwendung von Amazon SQS können Entwickler auf einfache Weise Daten zwischen verteilten Komponenten ihrer Anwendungen bewegen, die ohne Mitteilungen zu verlieren unterschiedliche Tasks ausführen oder Komponenten benötigen, die immer verfügbar sein müssen.

Tabelle 2 Komponenten der Amazon Web Services (AWS)

Amazon stellt keine fertigen Anwendungen zum Zugriff auf die AWS zur Verfügung - wohl aber die erforderlichen Schnittstellen, um entsprechende Anwendungen zu erstellen.

Dabei sind muss man die Anwendungen nicht zwangsläufig auf ressourcenhungrige Programme mit hohem Datendurchsatz beschränken, sondern kann die angebotene Flexibilität auch für Embedded Systems nutzbringend einsetzen.

Die Schnittstellen sind sehr gut dokumentiert und mit Tools unterstützt, was weitere Entscheidungskriterien für den Einsatz der AWS waren.



5. Cloud Computing mit Embedded Devices

5.1. Geräteplattform QIAsymphony

Die Geräteplattform *QIAsymphony*™ von QIAGEN dient der Probenvorbereitung für die molekulare Diagnostik, die auf dem Nachweis von Nukleinsäuren (DNA oder RNA) aus Blut-, Gewebe- oder anderen Proben beruht. Nach Entnahme der Proben vom Patienten müssen die in der Probe enthaltenen Nukleinsäuren stabilisiert und so extrahiert und aufgereinigt werden, dass eine nachfolgende Nukleinsäurendiagnostik zuverlässig und reproduzierbar durchgeführt werden kann.

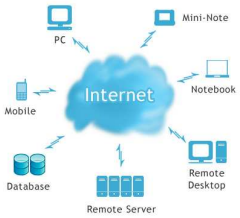
Die in Abbildung 4 gezeigte Geräteplattform QIAsymphony ist ein komplexes mechatronisches System mit zahlreichen Aktoren zur Ausführung von Steuerungsfunktionen (xyz-Portal mit Pipettierfunktion etc), Sensoren zur Prozessüberwachung und integriertem Steuerungsrechner mit grafischem Userinterface (Touchscreen), welche für Labornutzer in den Bereichen Forschung, angewandte Testverfahren und molekularer Diagnostik eine völlig neue Dimension der Probenhandhabung und eine erhebliche Zeit- und Kostenersparnis sowie eine massive Leistungssteigerung erlauben.

Abbildung 4 QIAsymphony Platform

Hinter dem System verbirgt sich die in Abbildung 5 vereinfacht dargestellte Systemarchitektur. Die CPU mit dem angeschlossenen Touchscreen als grafisches Userinterface bilden den zentralen Rechnerkern, auf welchem Linux als Betriebssystem läuft.

Von diesem über CAN/CANopen abgesetzt ist eine Reihe von lokalen Controllern angeschlossen. Diese lokalen Controller steuern die in einem solchen System meist zahlreich vorhandenen Antriebe (DC-Motoren, Schrittmotoren, Elektromagnete u.a.) und fragen Sensoren bis hin zu 2D-Barcode-Kameras ab, um dem zentralen Rechnerkern aufbereitete Informationen über Standard-Interfaces zur Verfügung stellen bzw. dessen Kommandos in den lokalen Controllern umsetzen.

Über Ethernet-TCP/IP gemäß Standard kann das Gerät mit einem Host verbunden und damit in ein (Firmen- resp. Klinik-) Netzwerk integriert oder mit der „Cloud“ verbunden werden.



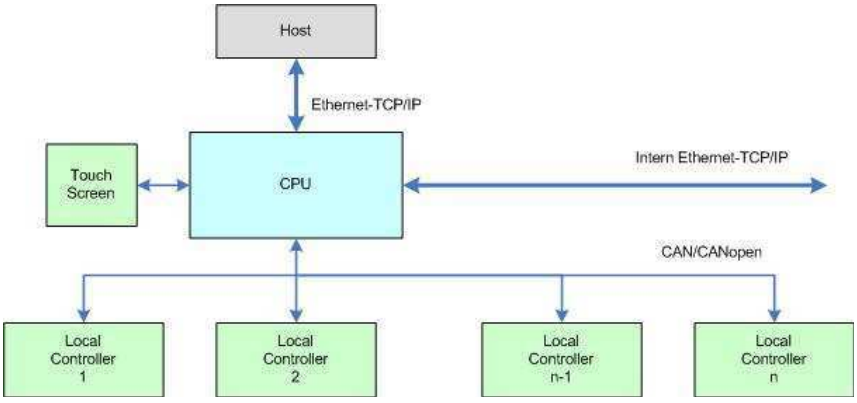


Abbildung 5 Vereinfachte Systemarchitektur QIAsymphony

Zur Protokollierung des Systemverhaltens werden alle relevanten Daten (Positionierungen, Messwerte, Stati, System- und Fehlermessages etc.) in ein Tracefile geschrieben. Die Mächtigkeit dieser Tracefiles wird über den Tracelevel (error, warning, info, debug) gesteuert. In frühen Entwicklungsphasen wird man den Tracelevel auf debug stellen und entsprechend umfangreiche Tracefiles für spätere Analysen erhalten.

Darüber hinaus werden bestimmte Daten in einer Datenbank abgelegt, die zu jeder Zeit Rückschlüsse auf den Systemzustand der installierten Gerätebasis ermöglichen sollen.

In den folgenden Kapiteln soll gezeigt werden, wie Amazon Simple Storage zur Speicherung von Tracefiles und Amazon SimpleDB zur Ablage von Zustandssignalen (Stati, Alerts, Errors) aus einem Embedded System eingesetzt werden kann. Über den Amazon Simple Queue Service werden wir an ausgesuchte Geräte der installierten Gerätebasis Mitteilungen versenden.

Für die Amazon Services S3, SimpleDB und SQS stehen C Bibliotheken zur Verfügung. Zusätzlich wird noch die libxml2, libcurl und OpenSSL benötigt. Die Bibliotheken libsdb und aws4c können problemlos in kommerzielle Applikationen gelinkt werden, da deren Lizenz das erlaubt. Die libs3 hingegen steht unter der GPL v3 und ist daher mit Vorsicht in kommerziellen Applikationen zu verwenden.

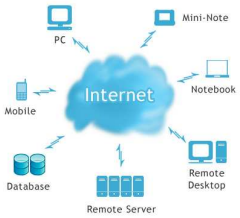
In der betrachteten QIAsymphony Plattform war Lua als Skriptinterpreter implementiert worden, um bereits vor dem Vorliegen der Applikationssoftware die darunter liegende hardwarenahe Software einschließlich der Hardware selbst in Betrieb nehmen zu können. Mit Hilfe von Lua konnten dann auch die folgenden Test auf recht einfache Weise durchgeführt werden konnten [11][12].

5.2. Amazon Simple Storage Service

Erste Tests zum Versenden von Tracefiles haben wir mit Hilfe des Kommandozeilen-Tools S3 durchgeführt. Dieses Tool baut auf der C-Library libs3 auf, welche den Zugriff auf den Amazon S3 Service ermöglicht [13].

Mit dem Kommando

```
s3 put 3030/ trace_2010_01_21_12_39_51.txt filename=/mnt/data/logs/logger.txt
```



kann bspw. das Tracefile logger.txt als File trace_2010_01_21_12_39_51.txt in das Bucket 3030 (QIAsymphony Seriennummer) des Amazon S3 Service abgelegt werden. Für unsere Tests haben wir das Kommandozeilen-Tool S3 aus Lua heraus mit dem Kommando os.execute() direkt aufgerufen.

Die Library Libs3 unterliegt der GPL, weshalb das Linken von libs3.so nicht zulässig ist, wenn der Quelltext nicht weitergegeben werden kann/darf. In einer definitiven Umsetzung wird man dann beispielsweise auf die unter der LGPL verfügbare Library aws4c ausweichen [14].

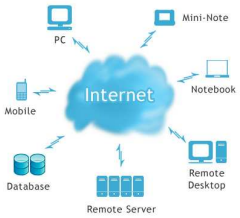
Um ein solches in der „Amazon Cloud“ abgelegtes Tracefile zu betrachten, kann man sich beispielsweise den Tools von Gladinet (www.gladinet.com) bedienen. Bereits mit dem für nicht kommerzielle Anwendung freien Gladinet Cloud Desktop stehen die folgenden Funktionen zur Verfügung:

- Maps Cloud Storage als Netzlaufwerk
- Upgrades Explorer zu einem Storage Portal
- Backups Files (Pictures, Music, Videos etc.) Online
- Schneller Filetransfer

Abbildung 6 zeigt die Oberfläche des Gladinet Cloud Desktops und verdeutlicht damit bereits dessen Funktionalität.

Abbildung 6 Gladinet Cloud Desktop

Abbildung 7 zeigt das Netzlaufwerk Z: im Explorer, welches den Zugriff auf den Amazon S3 Service (aber auch Google Docs und Picasa) ermöglicht. Abbildung 8 hingegen zeigt die in Bucket 3030 abgelegten Files, worunter sich auch erwartungsgemäß unser Tracefile trace_2010_01_21_12_39_51.txt befindet.



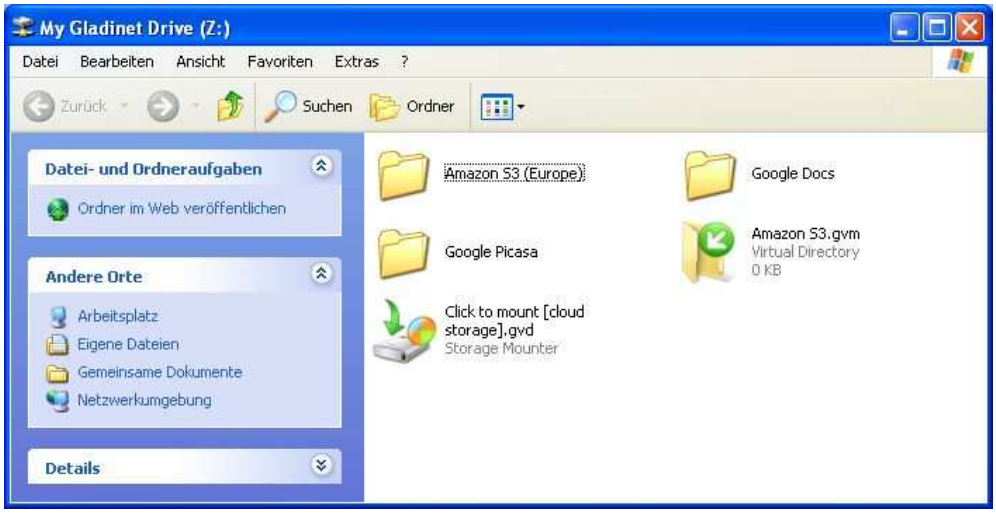


Abbildung 7 Gladinet Netzlaufwerk (Z:)

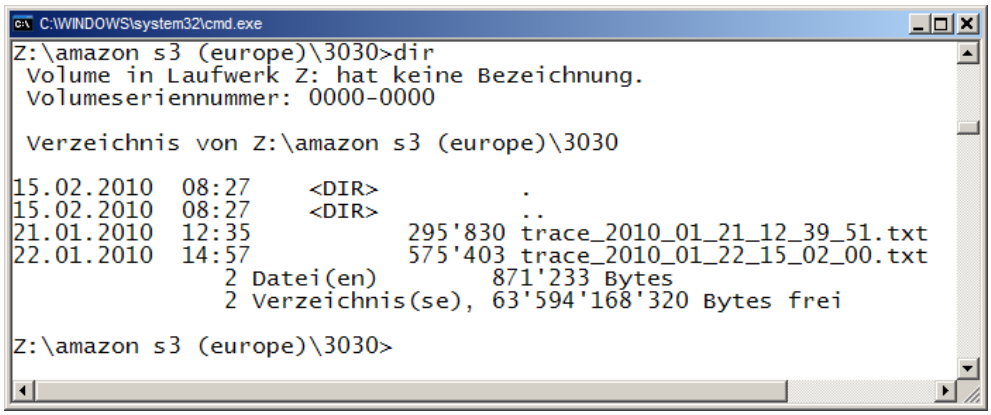


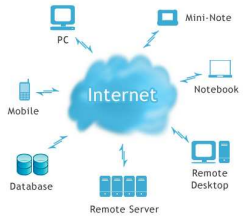
Abbildung 8 Amazon S3 - Files in Bucket 3030

5.3. Amazon SimpleDB

Um Werte in der Amazon SimpleDB abzulegen, habe wir uns der Library libsdb bedient. Die C-Library erlaubt dem Entwickler, von einem C/C++-Anwendungsprogramm auf die SimpleDB zuzugreifen. Die Library unterstützt ein einfaches Interface zur SimpleDB (API), ohne dass der Anwender Kenntnisse über den Zugriff über das HTTP-Protokoll haben muss.

Der Aufruf in C erfolgt in der Form

```
sdb_put_many(sdb, domain, item, num, keys, values);
```



Für unsere Tests wurde die Library libsdb in Lua eingebunden, wodurch der Zugriff über eine Tabelle erfolgen konnte [15].

Um nun vom Embedded Device Werte zur Simple DB zu senden, werden diese in der Tabelle values abgelegt. Im folgenden Beispiel ist das die aktuelle Uhrzeit (hour, min), die über einen Aufruf von Funktionen des Betriebssystems ermittelt wird

```
values = {}
table.insert(values, {key = “hour”, value = os.date("%H")})
table.insert(values, {key = “min”, value = os.date("%M")})
```

Das Versenden erfolgt dann über den Aufruf

```
sdb:put(DOMAIN, sdb_uid(), values)
```

Für die Abfrage der SimpleDB gibt es im Netz wieder zahlreiche Tools. Wir haben mit dem SimpleDB Explorer gearbeitet, der SQL-ähnliche Zugriff auf die SimpleDB ermöglicht. Abbildung 9 zeigt die Abfrage unserer Datenbank nach Einträgen, die Fehler in den Monaten Januar und Februar dieses Jahres selektieren.

Das Gerät mit der Seriennummern 4585 hat am 3.02.2010 um 11:19 den Fehler 2000 gemeldet, während das Gerät mit der Seriennummer 3030 am 22.01.2010 um 15:01 den Fehler 41 und am 3.02.2010 um 9:17 den Fehler 12 gemeldet hat.

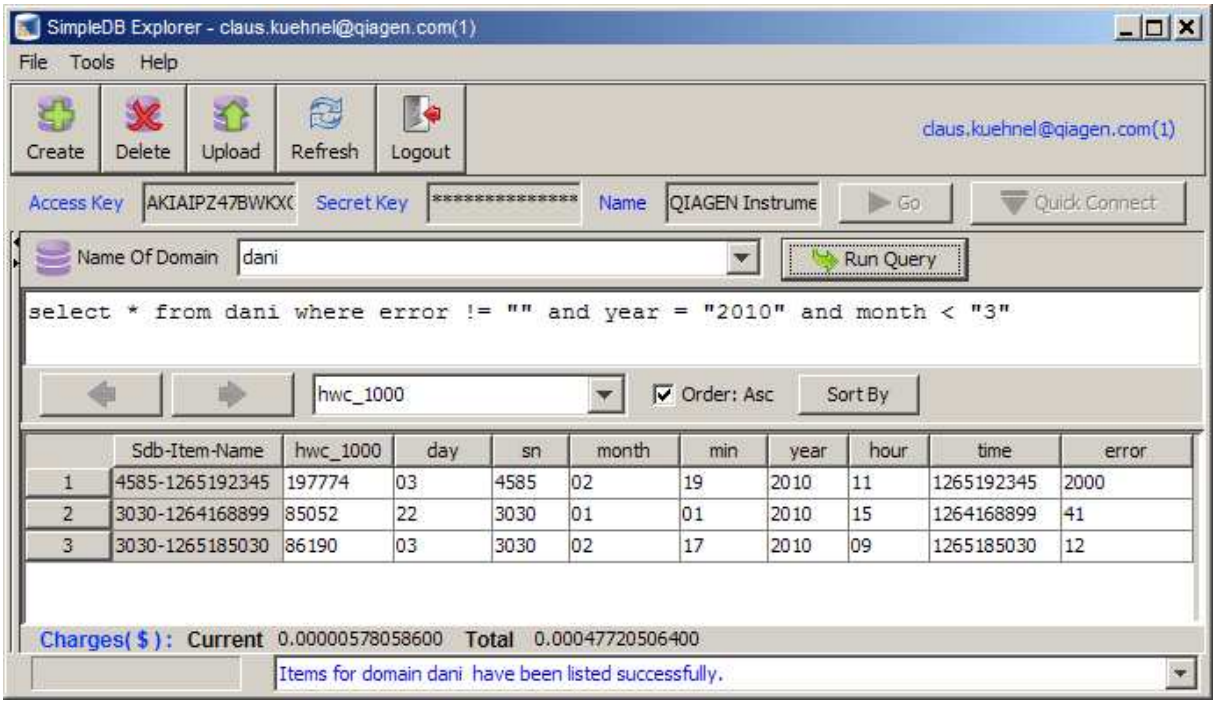
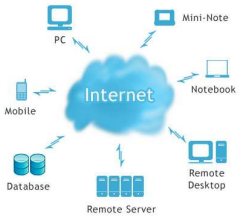


Abbildung 9 SimpleDB Abfrage im SimpleDB Explorer



5.4. Amazon Simple Queue Service

Ein Architekturmerkmal skalierbarer Systeme sind lose gekoppelte Softwarekomponenten, die über sogenannte Message Queues kommunizieren. Messages Queues implementieren eine asynchrone Kommunikation zwischen Sender und Empfänger durch ein Zwischenspeichen der Messages bis diese vom Empfänger abgeholt werden [16].

Amazon SQS dient damit als "transient buffer" zwischen Producer und Consumer (Abbildung 10). Der Producer kann über den Tag verteilt Mitteilungen erzeugen, die der Consumer (unser Embedded Device) nach dem Einschalten des Gerätes bzw. während seiner Laufzeit abholen kann.

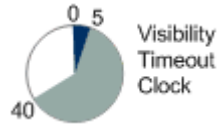
Abbildung 10 Amazon Simple Queue Service

Mit dem Visibility Timeout wird die lose Kopplung der Softwarekomponenten im Amazon SQS wirkungsvoll unterstützt. Die folgende Zusammenstellung erläutert dieses Merkmal.

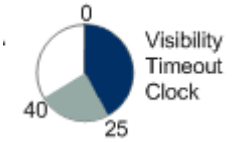
Vom Producer wird eine Message in die Queue gestellt und ist über den SQS Server verfügbar. Verbunden mit der Message wird ein Visibility Timeout gesetzt, der die Sichtbarkeit der Message definiert.

SQS löscht automatisch alle Messages, die länger als 4 Tage in der Queue stehen.

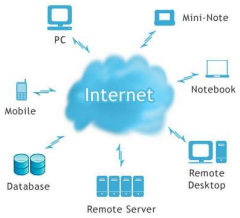
Ist der Consumer bereit, eine Message zu prozessieren, dann kann dieser die Message aus der Queue beziehen. Während der Zeit der Bearbeitung bleibt die Message in der Queue ist aber für die Zeit des während der Visibility Timeouts für weitere Abfragen nicht sichtbar



Hat der Consumer die Message erfolgreich prozessiert, dann löscht er diese aus der Queue. Durch diese Massnahme wird verhindert, dass vor Ablauf des Visibility Timeouts die Message erneut prozessiert wird.



In unseren Test haben wir Amazon SQS für ein System eingesetzt, welches beliebige Mitteilungen an ausgewählte oder auch alle Geräte unserer installierten Gerätebasis versenden kann. Abbildung 11 zeigt das betrachtete System. Das Notebook steht in unserem Beispiel für den Producer der Messages, während die angeschlossenen Laborgeräte (Embedded Devices) für die Consumer stehen.



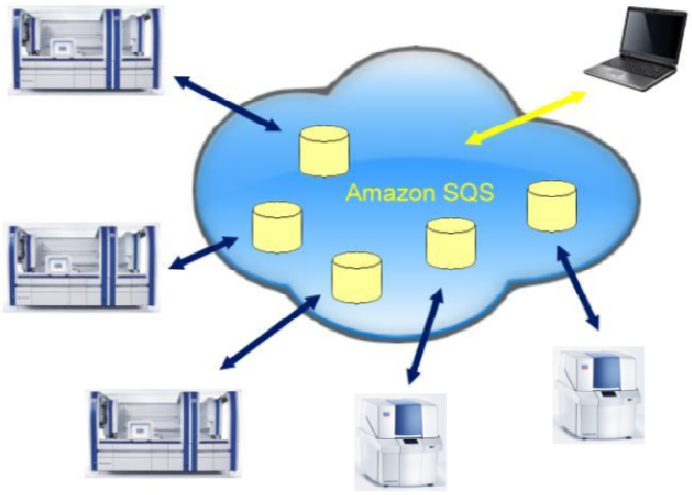


Abbildung 11 Versenden von Mitteilungen zur installierten Gerätebasis

Für den Zugriff zu Amazon SQS verwendeten wir wieder die Library aws4c, die neben Amazon S3 auch SQS unterstützt.

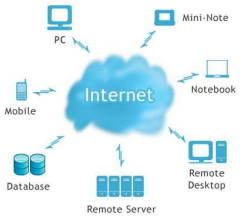
Im Abstand von wenigen Sekunden sendet das Embedded Device aus Lua heraus die folgende Anfrage an Amazon SQS:

```
ok, errmsg, err, msg, receipt = sqs.get(queueURL)
if ok then
-- Show message
sqs.del(queueURL, receipt)
end
```

Die Bearbeitung der abgerufenen Message ist hier nur durch den Kommentar (-- Show message) verdeutlicht.

Da für die PC-Seite keine brauchbaren Applikationen zum Versenden einer Message zum Amazon SQS gefunden wurden, verwendeten wir wieder die Library aws4c mit der gleichen Lua Anbindung wie auf dem Embedded Device, diesmal aber für x86 kompiliert.

Mit dem Funktionsaufruf list = sqs.list(prefix) bekommt man eine URL für jede Queue die mit dem Prefix „prefix“ beginnt. Danach muss über die Tabelle „list“ iteriert werden und mit sqs.send(url, msg) wird die Nachricht in die jeweilige Queue gesendet.




```
function sqs_send(prefix, msg)
  local ok, errmsg, err, list = sqs.list(prefix or "")
  for i=1,#list do
     sqs.send(list[i], msg)
  end
end
```

Um nun vom PC (Producer) eine Meldung an das Gerät mit dem Namen "qssp3030" (Consumer) zu senden, bedarf es nur noch des folgenden Kommandos:

```
sqs_send(„qssp3030", „Eine Meldung")
```

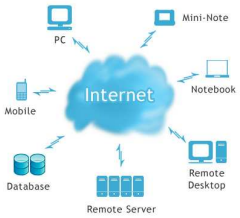
In unseren Tests wurden die versendeten Mitteilungen auf dem Bildschirm des Embedded Devices angezeigt können dort aber grundsätzlich beliegen Aktionen auslösen.

6. Schlussfolgerungen

Ziel unserer Untersuchungen war es, das Cloud Computing im Zusammenhang mit Embedded Devices zu untersuchen. Wir haben uns dabei für die Amazon Web Services entschieden, weil die Schnittstellen offengelegt und sehr gut dokumentiert sind. Die Abrechnung erfolgt ohne administrativen Overhead über eine Kreditkarte.

Eine Embedded System mit Linux als Betriebssystem bietet alle Voraussetzungen für eine effektive Netzwerkeinbindung. Bei Systemen, die mit weniger Ressourcen ausgestattet sind, kann es möglich sein, dass das hier verwendete https durch http ersetzt werden muss. Die Sensibilität der Daten wird entscheiden, ob dieser Weg möglich ist.

Unabhängig davon, ob man die hier gezeigten Dienste in einem Produkt zugänglich machen resp. nutzen will, bietet sich in der Entwicklungsphase eines Embedded Devices eine hervorragende Möglichkeit umfangreiche Daten des Systems zu gewinnen. Diese Daten können den jeweiligen Gerätestatus transparent machen und helfen das spätere Produkt stabiler zu gestalten. Für Feldtests neuer Geräte, die häufig bei Schlüsselkunden über die Welt verteilt durchgeführt werden, stellt das eine neue Qualität dar.



7. Referenzen

[1] Rodenhäuser, B.:
Cloud Computing – Damit Sie nicht aus allen Wolken fallen.
http://www.manager-magazin.de/it/cio-spezial/0,2828,582750,00.html

[2] Chiu, W.:
Cloud Computing: Hype Versus Reality
http://www.cio.com/article/438371/Cloud_Computing_Hype_Versus_Reality

[3] Armbrust, M. et. al.:
Above the Clouds: A Berkeley View of Cloud Computing
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

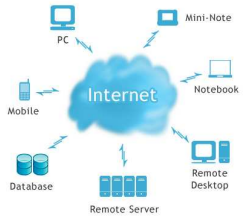
[4] Stienhans, F.:
Cloud Computing at SAP
http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/14095

[5] Cloud Computing - Evolution in der Technik, Revolution im Business
BITKOM-Leitfaden
http://www.bitkom.org/de/publikationen/38337_61111.aspx

[6] Zhen, J.:
Three Cloud Resource Consumption Models
http://www.zhen.org/zen20/tag/iaas/

[7] Jones, M.T.:
Cloud computing with Linux - Cloud computing platforms and applications
http://www.ibm.com/developerworks/linux/library/l-cloud-computing/

[8] Top 10 Cloud Computing Service Providers of 2009
http://ow.ly/Led1



[9] Varia, J.:
Cloud Architectures.
Amazon Web Services, January 2010
jineshvaria.s3.amazonaws.com/public/cloudarchitectures-varia.pdf

[10] Amazon Web Services: Overview of Security Processes
http://awsmedia.s3.amazonaws.com/pdf/AWS_Security_Whitepaper.pdf

[11] Kühnel, C.; Zwirner, D.:
Die Skriptsprache Lua.
http://www.elektroniknet.de/home/embeddedsystems/fachwissen/uebersicht/software/entwicklungssoftware/die-skriptsprache-lua/

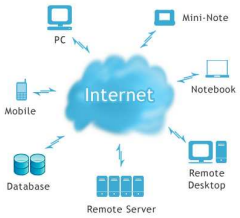
[12] Kühnel, C.; Zwirner, D.:
Symphonie aus C++ und Lua.
Skriptsprache mit kleinem Interpreter
http://www.mechatronik.info/mech/o_archiv.asp?ps=ME101758&task=03&o_id=25122133231-80

[13] The libs3 C Library API for Amazon S3
http://libs3.ischo.com.s3.amazonaws.com/index.html

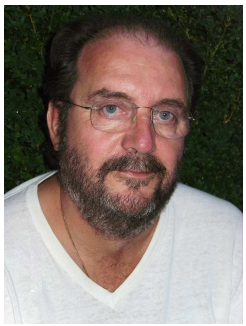
[14] aws4c - C library to access Amazon S3 and SQS services
http://code.google.com/p/aws4c/

[15] libsdb: C API Library for Amazon SimpleDB
http://code.google.com/p/libsdb/

[16] SQS: Super Queue Service
http://aws.typepad.com/aws/2007/05/sqs_super_queue.html



Autoren



Dr.-Ing. Claus Kühnel studierte und promovierte an der Technischen Universität Dresden auf den Gebiet der Informationselektronik und bildete sich später in Biomedizintechnik weiter. Seit 2004 ist er bei der QIAGEN Instruments AG in Hombrechtikon (CH) als Associate Director Electronic Engineering für die Entwicklung von Elektronik-Hardware und hardwarenaher Software verantwortlich.

claus.kuehnel@qiagen.com

Dipl.-Ing HTL Daniel Zwirner studierte an der Hochschule für Technik Rapperswil Elektrotechnik und bildete sich mit einem Nachdiplom-Studium in Software Engineering weiter. Seit 2005 ist er bei der QIAGEN Instruments AG in Hombrechtikon (CH) als Software-Engineer beschäftigt und dort an der Entwicklung einer neuen Geräteplattform für die molekulare Diagnostik beteiligt. Er arbeitet im Bereich der Betriebssystem und Gerätesteuerungsentwicklung.

daniel.zwirner@qiagen.com