Cloud-Computing Seminar (CLCP) Universität Heidelberg

Fikret Kaplan

Universität Heidelberg
kaplan.fikret@gmail.com

25.5.2009

Inhalt

- Überblick
- Windows Azure
- Azure Fabric
- Azure Storage
  - Blob
  - Table
  - Queue
- .NET Services
- Zusammenfassung
- Zukunft
- Quellen

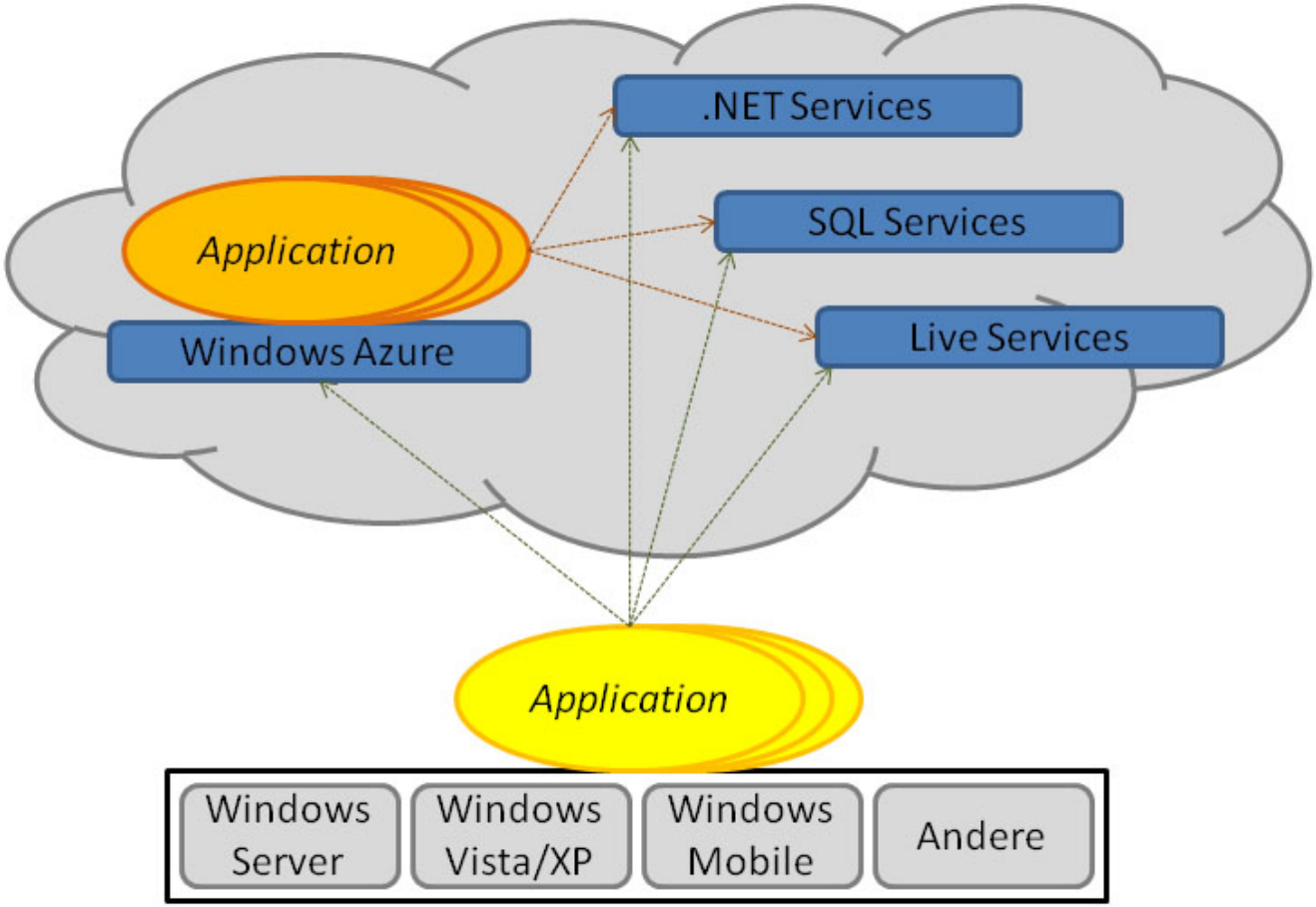
Uberblick

- Mit Azure Services Platform präsentiert Microsoft seine Cloud.
- Community Technology Preview (CTP) seit Herbst 2008
  - nur .NET-basierte Anwendungen (managed Code)
  - kostenfrei (mit Quota)
- Soll noch dieses Jahr den Betrieb aufnehmen.
- Ausfall am 13. März 2009 für mehrere Stunden.

Überblick

- Anschaffung und Verwaltung von Systemen entfällt
- Rechen- und Speicherkapazitäten nach dem „on-demand“-Prinzip
- Kosten nach dem „pay as you go“-Prinzip
- SLA für Anwendung
  - läuft immer
  - Daten immer verfügbar
  - Skalierbarkeit

Überblick

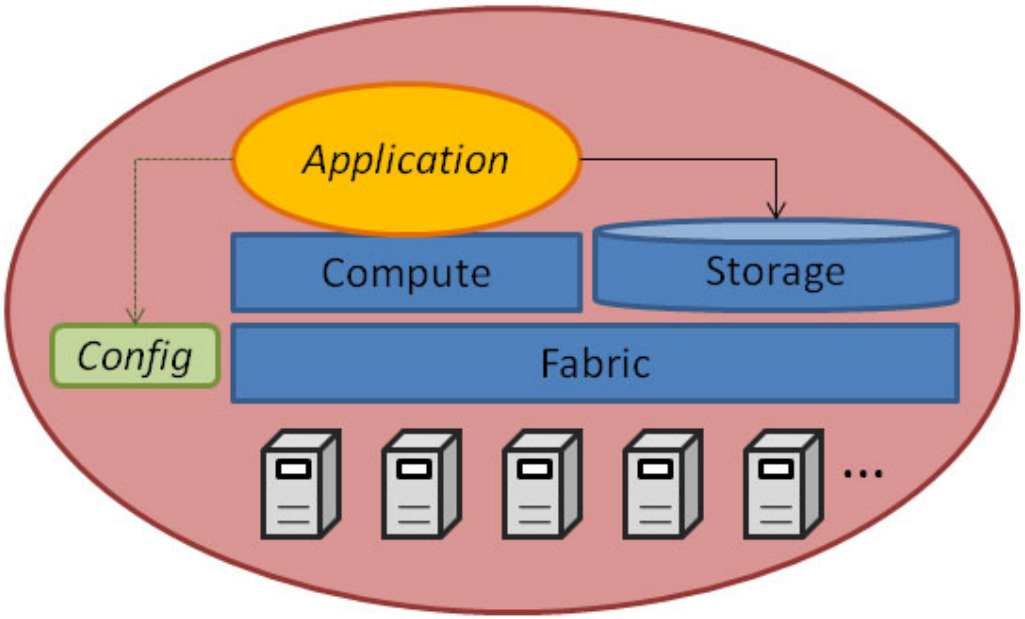

.NET Services
SQL Services
Live Services
Application
Windows Azure
Application
Windows Server
Windows Vista/XP
Windows Mobile
Andere

Windows Azure

- Betriebssystem der Cloud
- Windows-basierte Umgebung für das...
  - Ausführen von Windows-Anwendungen,
  - Speichern von Daten.
- Läuft auf einer Menge von physikalischen Rechnern.
- Azure Fabric fügt diese zu einer logischen Einheit zusammen und verwaltet.
- Dienste für das Ausführen und Speichern bauen auf der Fabric auf.
- Erlaubt Webanwendungen und Hintergrundprozesse.

Windows Azure

- *Compute*: in CTP nur managed Code
- *Storage*: Blobs, Queues und Tables
- *Config*: steuert Verhalten der Anwendung

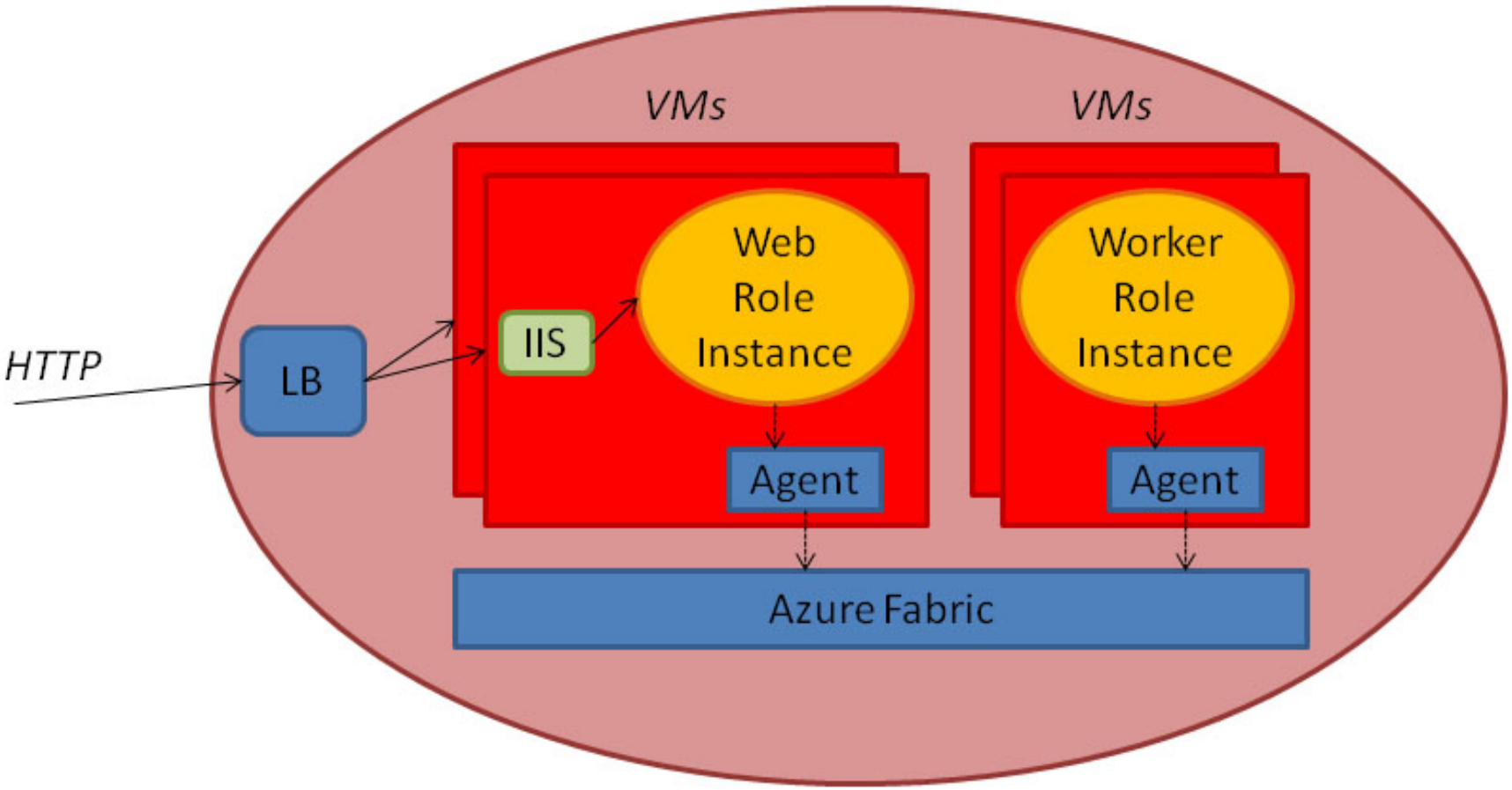


- *Fabric*: Resourcenmanagment

Windows Azure

- Jede Instanz einer Anwendung läuft in einer eigenen VM.
- Zwei Arten von Anwendungen entsprechend der Funktion:
  - *Web Role*:
    - verarbeitet eingehende HTTP(S)-Anfragen
    - beendet wenn Anfrage abgearbeitet
  - *Worker Role*:
    - Input kommt von Web Role
    - nur abgehende Netzwerkverbindungen
    - Laufzeit beliebig
- Kommunikation zwischen Web Role und Worker Role über Queue
- Azure Agent läuft in jeder VM und stellt API zur Verfügung, z.B. für Logging oder Versenden von Störungsmeldungen

Windows Azure



Azure Fabric

- Verwaltet die Resourcen (Recheneinheit, Switch, Router, VIPs[1],...) im Datacenter[2].
- Ordnet Anwendungen und Daten physikalischen Maschinen zu.
- Gruppiert die Maschinen in *fault domains* und *update domains*
  - *fault domain*:
    - Topologie des Datacenters
    - Switch- oder Stromausfall
  - *update domain*:
    - Anteil einer Anwendung, die beim Upgrade ausfallen darf
- Agiert entsprechend Anwendungs- und Systemkonfiguration
- Monitor und Eventhandler für Anwendungen.

[1] Virtual IPs

[2] Serverfarm

Azure Fabric

- Virtualisierung
  - Images sind Virtual Hard Disks
  - Betriebssystem = Datei
- Wartung offline
- Keine Installationen da nur Dateien kopiert werden

- Hypervisor stellt Umgebung für VMs
- Host-Betriebssystem: Windows Server 2008 64bit (Core)

- Agent läuft in jeder VM und kommuniziert mit Fabric

Azure Fabric

- Eckdaten der VM
  - Betriebssystem: Windows Server 2008 64bit (Enterprise)
  - CPU: 1.5GHz - 1.7GHz x64
  - RAM: 1.7GB
  - Netzwerk: 100Mbps
  - Lokal temporärer Speicher: 250GB
  - Azure Storage: 50GB
- Post-CTP: größere Auswahl

Azure Fabric

- Sicherheit für Anwendungen
  - Shared Environment
  - Deshalb Sicherheit durch Isolation
    - Zugriff nur auf Resourcen, die im Modell angegeben sind
    - Firewall
    - VM
    - IP-Filter
  - Sicherheitsupdates
- Ausfallsicherheit
  - 5-7 Fabric-Replicas
  - wenn alle ausfallen, laufen die Anwendungen weiter (?!)

Azure Storage

- Azure Storage speichert . . .
  - Blobs
  - Queues
  - Tables
- Daten werden mind. dreifach gehalten
- Im Produktivbetrieb „Geo-Distribution“

Azure Storage : Blob

- Speichern von Dateien mit Metadaten.
- Sichtbarkeit wird auf Container-Level festgelegt: public oder private.
- PUT / GET / DELETE via REST[3].
- Blob URL:
  `http://<Account>.blob.core.windows.net/<Container>/<BlobName>`
- Max. Blob-Größe 50GB.
- Blobs können in Blöcke bis zu 4MB geteilt werden.

[3] Restructured Text

Azure Storage : Blob

Upload eines großen Blobs:

```
blobName=''piratecopy.avi'';
PutBlock(blobName, blockId1, block1Bits);
PutBlock(blobName, blockId2, block2Bits);
...
PutBlock(blobName, blockIdN, blockNBits);
PutBlockList(blobName, blockId1, blockId2,...,blockIdN);
```

- Resume: ein Block anstatt der ganze Blob
- Upload parallel und ungeordnet

Azure Storage : Blob

- *PutBlockList* bestimmt den Blob, egal in welcher Reihenfolge die Blöcke kamen.
- Mehreren identische Blöcke: Letzter gewinnt.
- Ungenutzte Blöcke werden gelöscht.

- *GetBlockList* liefert die Blöcke (ID und Größe) eines Blobs.

Azure Storage : Blob

REST *PutBlock*

```
PUT
http://mystorage.blob.core.windows.net/movies/piratecopy.avi
?comp=block &blockid=BlockId1 &timeout=60
HTTP/1.1 Content-Length: 4194304
Content-MD5: **********
Authorization: SharedKey mystorage: **************
X-ms-date: Tue, 9 Jul 2009 18:00:00
```

— Block Data Contents —

- Content-MD5 optional
- *SharedKey* ist HMACSHA256

Azure Storage : Blob

REST *PutBlockList*

```
PUT
http://mystorage.blob.core.windows.net/movies/piratecopy.avi
?comp=blocklist &timeout=60
HTTP/1.1 Content-Length: 161213
Authorization: SharedKey mystorage: **************
X-ms-date: Tue, 9 Jul 2009 18:00:00
<?xml version=''1.0'' encoding=''utf-8''?>
<BlockList>
<Block>BlockId1</Block>
<Block>BlockId2</Block>
...
</BlockList>
```

Azure Storage : Blob

REST *GetBlock*

```
GET
http://mystorage.blob.core.windows.net/movies/piratecopy.avi
HTTP/1.1
Authorization: SharedKey mystorage: ************
X-ms-date: Tue, 9 Jul 2009 18:00:00

GET
http://mystorage.blob.core.windows.net/movies/piratecopy.avi
?timeout=120 HTTP/1.1
Authorization: SharedKey mystorage: ************
Range: bytes=1024000-2048000
```

Azure Storage : Table

- Strukturierter Speicher: Menge von Entitäten mir einer Menge von Eigenschaften.
- Keine relationale Datenbank-Tabelle.
- Tables können partitioniert werden.
- Zwei Schlüssel bilden den Primärschlüssel: *PartitionKey* + *RowKey*
- Entitäten einer Partition können am selben Ort gespeichert werden: effizientes Caching und schnellere Queries
- Skalierbarkeit (Traffic auf Partitionen wird beobachtet)
  mehr Partitionen: besser skalierbar

Azure Storage : Table

- Jede Entität kann bis zu 255 Eigenschaften besitzen.
- *PartitionKey* und *RowKey* müssen existieren.
- Restliche Eigenschaften können beliebig gesetzt werden.
- Innerhalt der Table können Entitäten unterschiedliche Anzahl und Typ von Eigenschaften haben.

Typen:

- PartitionKey und RowKey: String
- Eigenschaften: String, Binary, Boolean, DateTime, GUID, Integer, Integer64, Double

Azure Storage : Table

- Entitäten werden als Objekte via ADO.NET angesprochen
  - .NET-Klassen und LINQ[4]
  - REST
- Funktionalität: Insert / Update / Select / Delete
- Konkret in REST: POST / PUT / GET / DELETE

[4] Language Integrated Query

Azure Storage : Table

```
[DataServiceKey("PartitionKey", "RowKey")]
public class Message
{
        public string PartitionKey { get; set; }
        public string RowKey { get; set; }
        public string Text { get; set; }
        public int Rating { get; set; }
}
```

```
Message msg = new Message {
        PartitionKey = „FavoriteSongs",
        RowKey = DateTime.UtcNow.ToString(),
        Text = "The Knife - Heartbeats",
        Rating = 5
};
```

```
serviceUri = new Uri("http://mystorage.table.core.windows.net");
var context = new DataServiceContext(serviceUri);

context.AddObject("Messages", msg);
DataServiceContext response = context.SaveChanges();
```

Azure Storage : Table

Voriges Beispiel: Einfügen in Table in C#, hier via REST

```
POST http://mystorage.table.core.windows.net/Messages

… <!-- Atom envelope -->
<m:properties>
        <d:PartitionKey>FavoriteSongs</d:PartitionKey>
        <d:RowKey>30-06-2009</d:RowKey>
        <d:Text>The Knife - Heartbeats</d:Text>
        <d:Rating>5</d:Rating>
</m:properties>
```

Azure Storage : Table

Master-Table *Tables*

```
[DataServiceKey("TableName")]
public class TableStorageTable
{
        public string TableName { get; set; }
}
```

```
serviceUri = new Uri("http://mystore.table.core.windows.net");
DataServiceContext context = new DataServiceContext(serviceUri);
```

```
TableStorageTable table = new TableStorageTable("Messages");
Context.AddObject("Tables", table);
DataServiceContext response = context.SaveChanges();
```

Azure Storage : Table

Query via C# und LINQ

```
serviceUri = new Uri("http://mystore.table.core.windows.net");
DataServiceContext context = new DataServiceContext(serviceUri);

var messages =
        from message in context.CreateQuery<Message>("Messages")
        where message.Rating == 5
        select message;

foreach (Message msg in messages){ … }
```

- REST

```
GET http://<serviceUri>/Messages?$filter= Rating eq 5
```

Azure Storage : Table

ADO.NET	**REST**
API in .NET Framework 3.5 SP1	Standard HTTP Interface
Daten in Objekten	Daten in Atom (XML)
LINQ für Queries	URLs für Queries

Azure Storage : Table

- Queue enthält Nachrichten.
- Realisieren Nachrichtenaustausch zwischen Web Role und Worker Role.
- Asynchron.
- Queue URL: `http://<`*Account*`>.blob.core.windows.net/<`*QueueName*`>`
- Max. Größe einer Nachricht 8KB.
  Bei größeren Nachrichten: in Table oder Blob speichern und entsprechende Information in die Nachricht.

Azure Storage : Table

Queue API

- Queues
  - Erstellen / Leeren / Löschen
  - Länge abfragen
  - Priorität/ Gewicht vergeben
- Nachrichten
  - Enqueue(queue, message)
  - Dequeue(queue, invisibilityTime)
  - Löschen(queue, messageId)

Azure Storage : Table

REST PUT

```
PUT
http://mystorage.queue.core.windows.net/myqueue
?messagettl=3600
HTTP/1.1 Content-Length: 3900
Content-MD5: **********
Authorization: SharedKey mystorage: **************
X-ms-date: Tue, 9 Jul 2009 18:00:00
```

— Block Data Contents —

- PUT: Klartext oder Binär
- GET: Base64-kodiert

.NET Services

- Access Control Service (Security Token Service)
  - Prüft und erstellt digital signierte SAML-Tokens[5]
  - Token enthält User-Information und definiert Authorisation für eine Anwendung
- Service Bus
  - Anwendungen registrieren sich mit ihrer URL und können gefunden werden
  - keine Adressumsetzung
  - Verbindung zur Anwendung wird hinter der Firewall aufgebaut
  - Clients sehen nur eine IP-Adresse („anonyme Anwendung“)
- Workflow Service
  - besonders für lange laufende Prozesse
  - definiert Aktivitäten, die Aktionen ausführen (z.B. Senden/ Backup)

[5] Security Assertion Markup Language

Zusammenfassung

- Konkurrent zu Google und Amazon
- verschiedene Storages für Anwendungen
- (eher) für Systementwickler
- nur managed Code (gut für diejenigen, die bereits .NET kennen)
- Fokus auf internetbasierte Services

Zukunft

- weitere Programmiersprachen (z.B. PHP)
- Integration in Eclipse
- Storage: Filestream, Cache, Lock
- weitere 20 Datacenter
- Ferne Zukunft: allmählich den Vertrieb von Software an den Endkunden runterschrauben

Quellen

- Windows Azure Hompage (und weiterführende Links)
- Wikipedia (und weiterführende Links)