

**Universitatea Politehnica Bucuresti**

**Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei**

**Sisteme de operare avansate**

***Partajarea resurselor in Azure Storage***

**Conducător științific,**

**Conf. Dr. Ing. Ștefan Stăncescu**

**Masterand,**

***Camelia CHELARU***

***Master IISC, An II***

**Bucuresti 2016**

# **Partajarea resurselor in Azure Storage**

## **Cuprins:**

<b>1. Imagine generală .....</b>	<b>3</b>
1.1 Arhitectura si servicii de tip Cloud .....	3
<b>2. Sistemul de stocare a datelor - Azure Storage .....</b>	<b>4</b>
2.1 Sisteme de stocare Azure: blob, tabele, cozi, fisiere .....	4
2.2 Criterii de performanta.....	6
2.3 Tipuri replicare .....	8
<b>3. Securitatea datelor .....</b>	<b>9</b>
3.1 Mapare permisiuni - tipuri cereri .....	10
3.2 Acces limitat la resurse - Shared access signature - SAS .....	11
3.2.1 Managementul resurselor (parametrii token, politica de acces) .....	12
3.2.1 Tipuri: Account SAS / Service SAS .....	12
<b>4. Managementul concurentei .....</b>	<b>114</b>
4.1 Tipuri.....	14
Concurenta optimista .....	14
Concurenta pesimista .....	14
“Ultimul castiga” .....	15
4.2 Accesul concurrential fisiere.....	17
SMB - Server Message Block .....	17
REST over HTTP .....	18
<b>5. Concluzii .....</b>	<b>19</b>
<b>Bibliografie .....</b>	<b>20</b>

## **1. Imagine generală**

Windows Azure storage (WAS) este un sistem masiv scalabil ce poate stoca și procesa sute de teraocte de date, potrivite pentru scenarii de BigData cerute de aplicații științifice, analize financiare sau media. Clientii WAS pot avea acces la date de oriunde și oricând. A fost lansat în folosita prima oară în Noiembrie 2008.

Sistemul folosește o partitionare automată ce rebalancează automat informațiile în funcție de trafic (controlul resurselor se face automat).

Sistemul este accesibil de oriunde în lume, pentru orice tip de aplicație (.net, Java, C++) și sistem de operare (Windows sau Linux), fie că rulează în cloud, pe un desktop local sau pe un server oarecare. Se poate folosi și pentru aplicații de mobil pentru a realiza sincronizarea cu celelalte instante din cloud. Resursele din Azure Storage sunt expuse prin REST API, folosind protocolul HTTP/HTTPS.

Un cont standard de Azure storage are acces la blob storage, table storage (date structurate), file storage și queue storage. Un cont standard poate conține până la 500 TB de date în formate combinate.

### **1.1 Arhitectura și servicii de tip Cloud**

WA este un mediu ce asigură rularea de aplicații, este o platformă de tip cloud-computing, cu abilități de scalare. Platforma Windows Azure include stratul de bază Windows Azure, precum și un set de servicii de dezvoltare ce se pot utiliza individual sau la un loc: Mașini virtuale – Azure Virtual machines, Servicii cloud – Azure cloud services, Site-uri Web – Azure Web Sites, Data Services, App Services, Network. [lista totală - <https://www.microsoft.com/ro-ro/azure/services/> ]

Oferă IaaS (Infrastructure as a Service) permitând dezvoltatorilor și speciaștilor IT crearea și utilizarea mașinilor virtuale în cloud. PaaS (Platform as a Service) este oferit prin Windows Azure Cloud Services, oferind suport pentru aplicații scalabile, sigure la costuri scăzute. Administrarea va fi făcută de managementul platformei.

Microsoft Azure este considerat ca unic urmaritor al Amazon WS după surse Gartner. ceilalți competitori ramanând mult în urmă.



Fig. 1 Windows Azure si competitorii [10]

## 2. Sistemul de stocare a datelor - Azure Storage

### 2.1 Sisteme de stocare Azure: blob, tabelle, cozi, fisiere

Contul de storage este cel mai mare nivel de spatiu de nume prin care se acceseaza celelalte servicii fundamentale din WAS.

**Blob storage:** - pentru stocarea datelor din fisiere, un blob poate stoca orice tip de text sau date binare (document, fisier media, sau installer de aplicatie). Mai poate fi referit in documentatie ca si "Object storage". [1] Se incapsuleaza intr-un container - set de bloburi; fiecare blob apartine unui container

- este potrivit pentru utilizatori cu volum mare de date nestrucutrate (documente, poze, muzica, back-upuri, imagini si text pentru aplicatii web, informatii de configurarea pentru aplicatiile cloud)
- blob storage ofera 3 tipuri de blob: block blob (streaming/stocarea documentelor/fisiere media/backup-uri), append blob (ca si block blobs dar sunt optimizate pentru operatii de adaugare, ex: loguri, unde informatiile se adauga la sfarsitul blobului), page blob (optimizate pentru reprezentarea

discurilor IaaS si suporta scrieri aleatoare, pot fi dimensionate pana la 1TB; un disc IaaS este un VHD - **Virtual Hard Disk** - stocat ca page blob).

**OBS:** *VHD - Virtual Hard Disk - este un format de fisier ce reprezinta un hard disk (HDD) virtual. Poate contine componentele unui HDD real: partitii de disc, si un sistem de fisiere. este hard diskul unei masini virtuale.* [2]

**Table storage:** - stocheaza seturi de date structurate. Table storage este un depozit de date de tip cheie-valoare NoSQL, permite acces scalabil rapid si dezvoltare rapida pentru cantitati mari de date. Fata de sistemele de baze de date traditionale, relationale, este lipsit de o structura predefinita pentru a se adapta la orice tip de date si pentru un acces rapid. Table storage urmareste un model cheie-atribut, insemand ca orice valoare din tabel va avea asociat un nume de proprietate. Numele de proprietate va putea fi folosit pentru filtrare si criterii de selectare. O colectie de proprietati si valorile asociate vor alcatui o entitate.

**OBS:** *Fara schema predefinita = 2 entitati din acelasi tabel pot contine colectii diferite de proprietati de tipuri diferite de date. Seturile de date pot fi flexibile. Numarul de entitati sau tabele nu este limitat.* [1]

Accesul la table storage se face prin protocoale standard REST. Tabele stocheaza informatiile ca entitati. O entitate este o colectie de perechi atribut-valoare, similar unui rand. Tabele sunt partitionate pentru a suporta rebalansarea. Fiecare tabel contine prima proprietate o cheie de partitionare ce specifica carei partitii ii corespunde o entitate - `getPartitionByEntity(entitate)`. A doua proprietate specifica o cheia de rand ce identifica entitatea cu o partitie data - `getEntityByPartition(partitie)`. Combinatia cheie-partitionare si cheia randului formeaza o cheie primara, unica. [7]

**Queue storage:** - suportul pentru procesarea fluxului de lucru si comunicare intre componente din serviciile cloud. Aceste componente sunt de cele mai multe ori decuplate si independente, din motive de scalabilitate. Queue Storage o solutie de mesagerie pentru comunicarea asincrona intre componentele unei aplicatii, fie ca ele ruleaza in cloud, pe mobil, desktop sau server dedicat. Oferta suport si pentru administrarea taskurilor/sarcinilor asincrone si pentru construirea de flux de lucru al proceselor.

Se expun 2 tipuri de resurse: cozi si mesaje. Fiecare cont are un numar nelimitat de cozi unic definite. Fiecare coada poate contine un numar nedeterminat de mesaje. Limita maxima a unui mesaj este de 64KB. Atunci cand un consumator citeste mesajul din coada, acesta il va procesa si il va sterge. Dupa ce mesajul este citit este facut invizibil altor consumatori pentru un interval specific. Daca mesajul nu a fost sters in intervalul specificat, visibilitatea se reface pentru ca alt consumator sa il proceseze. [7]

**File storage:** oferta mediu de stocare partajat pentru migrarea facilă a aplicatiilor vechi/legacy ce folosesc protocolul SMB. Masinile virtuale Azure si serviciile cloud partajeaza fisiere intre aplicatii prin partitii montate (mounted shares) fiind accesate prin Serviciul de fisiere REST API - File Service REST API. Montarea si accesarea simultana nu depinde de numarul de componente ale aplicatiei. File storage expune un serviciu de tip REST-API pentru a accesa datele dintr-o locatie partajata. Aplicatiile distribuite pot folosi File storage pentru a partaja de exemplu fisiere de configurare, loguri, metriki pentru a fi folosite de mai multe masini virtuale.

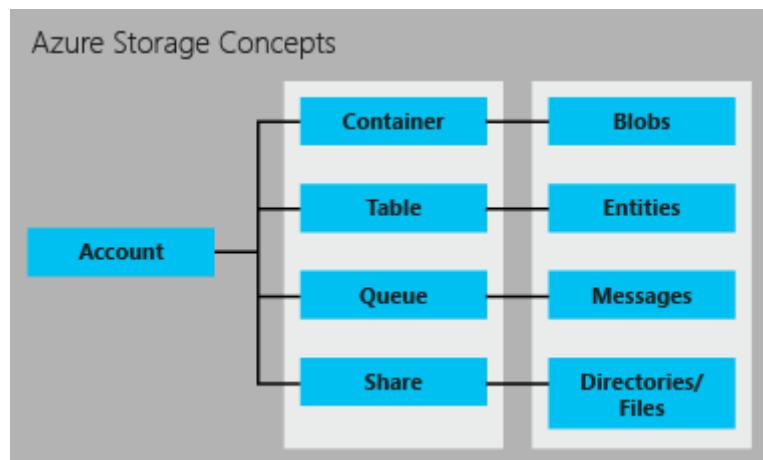


Fig.1 Incapsularea tipurilor de date in Azure Storage[1]

## 2.2 Criterii de performanta

Atunci cand aplicatia atinge limitele volumului de munca al partitiei, Azure Storage va returna erori cu codul 503 (Server Busy) sau 500 (Operation Timeout). Atunci cand apare efectul de bottleneck aplicatia ar trebui sa foloseasca o politica de revenire exponentiala pentru reincercari. Revenirea exponentiala permite incarcarea pe partitie sa scada, si de a usura traficul pe acea partitie. [3]

Daca aplicatia depaseste capacitatea de scalabilitate a contului, se pot folosi conturi multiple prin partionarea obiectelor de date de-a lungul mai multor conturi de storage.

Fiecare obiect ce retine informatii apartine unei partitii ce e identificata printr-o cheie. Partitia determina mecanismul prin care Azure Storage balanseaza incarcarea in mai multe servere din motive de trafic. Cheia partitiei este unica intr-un cont si e folosita pentru a localiza datele de tip blob, mesaj sau entitate. [3]

**Tabel 1 - Cheia de partitie**

<b>Tip de date</b>	<b>Tipul partitiei</b>
<i>Blob</i>	Numele containerului + numele blob-ului; fiecare blob are partitia lui, pot fi distribuite
<i>Fisiere</i>	Numele contului + numele partitiei de fisiere; toate fisiere dintr-o partitie impart aceeasi partitie
<i>Mesaje</i>	Numele cozii, astfel incat toate mesajele dintr-o coada sunt grupate intr-o singura partitie si servite de un singur server.
<i>Entitati</i>	Numele tabelului + cheia partitiei, unde cheia partitiei este valoarea definita de utilizator pentru entitate. Avantajul gruparii entitatilor intr-o singura partitie este efectuarea de operatii atomice de tip batch pe mai multe entitati, dat fiind faptul ca o partitie exista pe un singur server. Se poate si situatia in care am mai multe entitati din acelasi tabel pe mai multe partitii pentru o scalabilitate marita.

**Tabel 2 - Obiective de scalabilitate**

	<b>Resursa</b>	<b>Limita implicita</b>
<i>Blobs queues tabele fisiere</i>	<i>TB/per cont de storage</i>	<i>500 TB</i>
	<i>dimensiune max a unui blob block sau append blob</i>	<i>50000 x 4 MB (aprox 195 GB)</i>
	<i>numar max de proprietati dintr-un tabel entitate</i>	<i>252</i>
	<i>dimensiunea max al unui mesaj dintr- o coada</i>	<i>64 kb</i>
	<i>dimensiunea max al unei partitii de fisiere(file share)</i>	<i>5 TB</i>

	<i>dimensiunea max al unui fisier dintr-un file share</i>	<i>1TB</i>
	<i>rata cererilor totale</i>	<i>pana la 20 000 IOPS, intrari pe secunda sau mesaje pe secunda</i>
	<i>Capacitatea pentru blob simplu</i>	<i>pana la 60 MB pe secunda, sau pana la 500 cereri pe secunda</i>
	<i>Max ingress* per cont (Europa si regiuni asiatiche)</i>	<i>5 Gbps pentru GRS/ZRS 10 Gbps pentru LRS</i>
	<i>Max egress** per cont (Europa si regiunile asiatiche)</i>	<i>10 Gbps pentru RA-GRS/GRS/ZRS 15 Gbps pentru LRS</i>
	<i>Max ingress* per cont (US)</i>	<i>10 Gbps pentru GRS/ZRS 20 Gbps pentru LRS</i>
	<i>Max egress** per cont (US)</i>	<i>20 Gbps pentru RA-GRS/GRS/ZRS 30 Gbps pentru LRS</i>
<i>Azure resource manager</i>	<i>operatii de administrare a contului de storage - READ</i>	<i>800 in 5 minute</i>
	<i>operatii de administrare a contului de storage - WRITE</i>	<i>200 pe ora</i>
	<i>operatii de administrare a contului de storage - LIST</i>	<i>100 in 5 minute</i>

\* Ingress = toate datele/cererile ce sunt trimise catre un cont de storage.

\*\* Egress = toate datele/raspunsurile ce sunt primite de la un cont de storage. [3]

### **2.3 Tipuri replicare**

Datele din Windows Azure sunt mereu replicate pentru a asigura durabilitate si disponibilitate ridicata pentru a face fata defectiunilor hardware. Sunt mai multe optiuni pentru selectia replicarii: [1]

- LRS - Locally redundant storage - se mentin 3 copii ale datelor intr-o singura unitate dintr-o singura regiune.

- ZRS - Zone redundant storage - se mentin 3 copii ale datelor dar in 3 unitati si pana la 2 regiuni. Durabilitatea este mai crescuta fara de LRS. (este disponibila momentan doar pentru block blob).
- GRS - Geo-redundant storage - selectia default/implicita. Se mentin 6 copii ale datelor, 3 in regiunea primara si 3 intr-o regiune secundara situata la mare distanta geografica de prima.
- RA-GRS - Read access geo-redundant storage - datele se replica intr-o zona geografica secundara, dar in acelasi timp se ofera drepturi de citire acestei locatii auxiliare. Resursele pot fi accesate din ambele locatii.

Replication strategy	LRS	ZRS	GRS	RA-GRS
Data is replicated across multiple facilities.	No	Yes	Yes	Yes
Data can be read from the secondary location as well as from the primary location.	No	No	No	Yes
Number of copies of data maintained on separate nodes.	3	3	6	6

Fig. 2 Diferentierea intre tipurile de replicare

### 3. Securitatea datelor

#### Probleme de acces la resurse:

Pentru securitatea resurselor orice cerere de resurse dintr-un cont trebuie sa fie autentificata. Autentificarea pe bazeaza pe un model de cheie partajata. Bloburile pot fi configurate sa suporte autentificare anonima.

Un cont de storage are asociate la crearea 2 chei private de acces pentru autentificare. Aceste doua chei asigura disponibilitatea aplicatiei pentru momentul cand acestea se regeneaza din motive de securitate.

Pentru permiterea utilizatorilor acces controlat la resursele de stocare se pot crea semnaturi de acces partajat (shared access signature - SAS). SAS este un token/jeton ce se poate atasa unui URL si permite accesul la o resursa. Fiecare

utilizator ce detine tokenul poate accesa resursa in functie de permisiunile mapate si pentru un interval de timp valid. [1]

### **3.1 Mapare permisiuni - tipuri cereri**

Implicit in WAS doar posesorul contului poate accesa resursele din contul aferent. Doar pentru datele de tip blob se pot seta permisiuni la nivel de container pentru a se permite citiri anonime ca nivel de acces. Pentru restul tipurilor de date se va folosi cheia contului pentru accesul partajat. [4]

Pentru un control constrans al accesului se poate crea un semnatura de acces partajat (SAS - Shared access signature) ce permite acces restrictionat prin permisiuni si limitarea intervalului de timp.

#### **Accesul utilizatorilor anonimi la containere(blob-uri) - acces public**

Pentru a permite acces utilizatorilor anonimi unui container si blob-urilor continute se poate seta permisiune de acces public (nu mai e nevoie de autentificarea cererilor). Permisiunile de container se pot schimba din portalul Azure (<https://portal.azure.com/>) sau programatic folosind REST API-ul din libraria clientului sau din PowerShell. [4]

Optiuni pentru accesul la nivel de container:

- *acces public total pentru citire* - full public access read access - clientii pot enumera in cererea anonima blob-urile din container, dar nu se pot enumera containerele dintr-un cont.
- *acces public pentru citirea blob-urilor* - public read access for blobs only - clientii pot accesa prin cereri anonime doar continutul blob-urilor, nu si al containerelor.
- *fara acces public pentru citire* - no public read access - datele de tip blob si containerele pot fi accesate doar de proprietar.

### 3.2 Acces limitat la resurse - Shared access signature - SAS

Semnatura de acces partajat este o modalitate de garantă acces limitat obiectelor dintr-un cont de WAS, fără a fi nevoie de expunerea cheii contului.

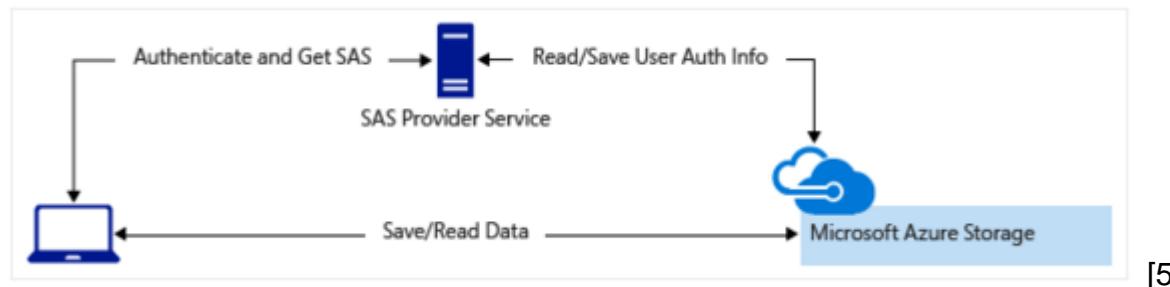
Practic SAS este un URI (**Uniform Resource Identifier**) ce localizează una sau mai multe resurse din storage și înglobează într-un token un set parametri de interogare Web(query parameters). Acest token conține informații pentru autentificarea accesului la resurse, cum vor fi acestea accesate de către client. Cheia contului este sensibilă, cu rol de administrare și acces total la resursa și partajarea acesteia nu este în siguranta.

Pentru autentificarea cererilor, se pot folosi 2 modele de design:

- prin Proxy - clientii fac upload și download de date printr-un serviciu de proxy ce autentifică cererile. Avantaje: validări complexe; dezavantaje: pentru volum mare de transacții pot apărea probleme de scalabilitate și incarcare.



- prin Serviciu dedicat - autentifică clientul și generează SAS. Dupa ce clientul primește semnatura, poate accesa direct resursele în funcție de permisiuni și intervalul de timp setat de serviciul SAS în prealabil. Avantaje: Elimina filtrarea cererilor prin proxy.



O cerere autentificată necesită 2 header - Date header și Authorization Header.

Date header conține marca de timp de tip UTC - Coordinated Universal Time pentru a specifica timpul creării cererii. Serviciile WAS se asigură ca fiecare cerere

sa nu aiba mai mult de 15 minute trecute de la lansare si pana ajung in sistem. Daca acesta verificare esueaza, atunci sistemul va returna eroare cu codul 403 (Forbidden)

Header-ul de autentificare - fara acesta, cererea este considerata anonima.

### **3.2.1 Managementul resurselor (parametrii token, politica de acces)**

Parametri folositi in token-urile din semnaturile SAS: (Account sau Service SAS) [5]

- **versiunea API** - parametru optional ce specifica versiunea serviciului de storage folosita pentru executarea cererii
- **\*versiunea serviciului** - parametru obligatoriu ce specifica versiunea serviciului de storage folosita pentru autentificarea cererii.
- **timpul de start** - parametru optional, format ISO 8601 timpul cand SAS devine valida; daca se omite atunci semnatara este valida din momentul cererii
- **\*timpul de expirare** - timpul dupa care SAS nu mai este valida; format ISO 8601
- **permisiuni** - operatiile ce pot fi executate de client
- **ip** - parametru optional - specifica ip-ul sau un set de ip-uri externe Azure de la care se vor accepta cereri
- **protocol** - parametru optional ce specifica protocolul permis pentru o cerere. (valori posibil: HTTP/HTTPS)
- **\*semnatura** - semnatura construita din ceilalti parametri si criptata folosind algoritmul SHA256 pentru criptare si Base64 pentru codare. Se foloseste practic pentru autentificare.

### **3.2.2 Tipuri: Account SAS / Service SAS**

In WAS semnaturile pot fi de 2 feluri: **Service SAS** si **Account SAS**.

## **Service SAS**

**Service SAS** deleaga acces la resurse doar in unul din servicii: blob/queue/table/file.

Parametri extra pentru Service SAS:

- **Resursa de stocare** - pentru ce se deleaga acces - containere sau blob-uri, partitii sau fisiere, cozi., tabele si intervale de entitati de tabele.

**Exemple de URI de tip SAS - Service: [5]** - acces citire si scriere pentru un blob

<https://myaccount.blob.core.windows.net/sascontainer/sasblob.txt?sv=2015-04-05&st=2015-04-29T22%3A18%3A26Z&se=2015-04-30T02%3A23%3A26Z&sr=b&sp=rw&sip=168.1.5.60-168.1.5.70&spr=https&sig=Z%2FRHIX5Xcg0Mq2rqI30LWTjEg2tYkboXr1P9ZUXDtkk%3D>

- **Blob URI - adresa resursei**  
<https://myaccount.blob.core.windows.net/sascontainer/sasblob.txt>
- **versiune serviciului ~sv=2015-04-05**
- **timpul de start ~st=2015-04-29T22%3A18%3A26Z**
- **timpul de expirare ~se=2015-04-30T02%3A23%3A26Z**
- **permisiuni ~sp=rw (permisiuni de read si write)**
- **ip ~sip=168.1.5.60-168.1.5.70**
- **protocol ~spr=https**
- **semnatura~sig=Z%2FRHIX5Xcg0Mq2rqI30LWTjEg2tYkboXr1P9ZUXDtkk%3D**
- **Resursa de stocare ~sr=b (resursa specificata in exemplu este blob)**

## **Account SAS**

**Account SAS** deleaga acces la resurse in unul mai multe servicii din storage. Permisiiunile la nivel de serviciu pot fi de citire, scriere, si stergere.

Parametri extra pentru Account SAS:

- **Serviciu/Servicii:** specifica ce tipuri de servicii sunt vizate (Servicii Blob sau de fisiere, Queue sau Table)
- **Tipuri de resurse** - specifica ce clase de servicii pot fi incluse, nu neaparat o resursa specifica:
  - **Service-level API** - la nivel de resurse de administrare cont. Ex: Get/Set Service Properties, Get Service Stats, List Containers/Queues/Tables/Share
  - **Container-level API** - se mapeaza pe clasa obiectelor din fiecare serviciu: container de blob-uri, cozi, tabele sau partitii de fisiere. Ex:

- Create/Delete Container, Create/Delete Queue, Create/Delete Table, Create/Delete Share, List Blobs/Files and Directories
- **Object level API** - blob-uri/ mesaje din cozi/ entitati de tabele si fisiere. ex: Put Blob, Query Entity, Get Messages, Create File

**Exemplu de URI de tip SAS - Account:** [5] - acces citire si scriere pentru un blob  
<https://myaccount.blob.core.windows.net/?restype=service&comp=properties&sv=2015-04-05&ss=bf&srt=s&st=2015-04-29T22%3A18%3A26Z&se=2015-04-30T02%3A23%3A26Z&sr=b&sp=rw&ip=168.1.5.60-168.1.5.70&spr=https&sig=F%6GRVAZ5Cdj2Pw4tgU7ILSTkWgn7bUkkAg8P6HESXwmf%4B>

- Resource URI - adresa resursei ~ <https://myaccount.blob.core.windows.net/?restype=service&comp=properties>
- serviciile ~ss=bf (**Blob si servicii de fisiere**)
- tipuri de resurse ~srt=s (**operatii la nivel de serviciu - Service-level API**)
- permisiuni ~sp=rw (**permisiuni de read si write**)

Alte detalii despre SAS si formarea cererilor autorizate pot fi accesate la <https://msdn.microsoft.com/en-us/library/azure/ee395415.aspx>

## 4. Managementul concurentei

### 4.1 Tipuri:

- **concurrenta optimista**

O aplicatie ce efectueaza o operatie de update va verifica starea resurselor, daca ele s-au modificat fata de ultima citire. De exemplu daca 2 utilizatori vizualizeaza si editeaza aceiasi pagina text, atunci aplicatia trebuie sa verifice daca cele doua modificarile nu se suprapun si nu apar suprascrieri, utilizatorii fiind informati despre statusul modificarilor. - folosit in general pentru aplicatii web

- **concurrenta pesimista**

O aplicatie ce efectueaza o operatie de update va bloca obiectul pentru a se asigura de siguranta executarii operatiei. Astfel pana cand acest lock expira se previn alti utilizatori de a modifica aceleasi resurse (interval 15-60 secunde). De exemplu, intr-un scenariu de replicare cu arhitectura master-slave, doar masterul are

drept de modificare a resurselor si va detine un lock exclusiv pentru o perioada extinsa de timp pentru a asigura integritatea datelor.

- “**Ultimul castiga**”

O abordare ce permite orice operatie de update sa fie efectuata fara a se verifica daca informatiile s-au schimbat intre timp, de la ultima citire. Se foloseste atunci cand infrastructura permite - atunci cand resursele sunt partionate astfel incat utilizatori multipli nu au sanse de a accesa aceleasi date. [8]

#### ➔ **Concurrenta la nivel Blob**

Pentru acest nivel se poate opta pentru concurrenta optimista sau pesimista. Daca nu se specifica nicio optiune atunci va fi aleasa implicit strategia de “ultimul castiga”.

##### **Blob - Abordarea optimista**

In WAS fiecare obiect stocat are asociat un identificator, ce se actualizeaza la fiecare operatie. Identificatorul face parte din raspunsul cererii de tip HTTP GET prin header-ul ETag, tag de entitate definit de protocolul HTTP. Utilizatorii ce efectuaza operatii de actualizare a obiectelor vor trimite ETag-ul original intr-un header conditional If-Match si se vor compara valorile corespund (ETag-ul din header cu ETag-ul din storage/depozit). Pasii verificarii:

1. La primirea unui blob de la WAS, raspunsul va include un ETag header ce identifica valoarea curenta a obiectului cautat
2. La actualizarea blob-ului se va include ETag-ul de la pasul 1 in header-ul If-Match al cererii
3. Serviciul compara cele 2 valori
4. Daca cele 2 valori sunt diferite, serviciul va returna o eroare cu codul 412 - inseamna ca blob-ul a fost actualizat intre timp de alt proces.
5. Daca cele 2 valori corespund, serviciul va actualiza blob-ul cu noua versiune.

##### **Blob - Abordarea pesimista**

Resursele sunt blocate pana cand acestea se elibereaza la expirarea timpului. Aceste rezervari de resursa pot activa mai multe strategii de sincronizare: scriere exclusive/citiri partajate, scriere exclusive/citiri exclusive sau scrieri

partajate/citiri exclusive. Scrituri = operatii de put, set, delete. Citirile exclusive se fac in functie de ID-ul rezervarii folosit de aplicatiile client, prin care WAS se asigura ca la un moment dat, doar o singur ID de rezervare va fi activ. Citirile partajate nu contin ID de rezervare. Daca apar conflicte, WAS va returna un raspuns HTTP eroare cu codul 409 (Conflict)

#### → **Concurrenta la nivel Table**

Se poate trata doar concurrenta entitatilor. Se poate opta doar pentru concurrenta optimista sau ultimul castiga. Procesul este similar celui de la blob - comparand ETag-ul. Daca e nevoie de lock-uri pesimiste, atunci se poate asociate un blob fiecarui tabel.

#### → **Concurrenta la nivel Queue**

Concurrenta poate apare atunci cand mai multi clienti consuma mesaje dintr-o coada. Atunci cand mesajul este luat din coada de un client, aceste nu se sterge, ci doar devine invizibil altor clienti pentru o perioada de timp(aceasta perioada este dat de parametrul de visibilitytimeout). dupa ce mesajul este procesat clientul va trebui sa stearga mesajul, inainte de timpul de a redeveni vizibil. Nu se ofera suport pentru concurrenta optimista sau pesimista.[8]

#### → **Concurrenta la nivel fisiere**

Serviciul de fisiere poate fi accesat prin 2 cai: protocolul SMB si REST. Serviciul REST nu ofera suport pentru concurrenta optimista sau pesimista. Clientii SMB ce monteaza partitiile de fisiere pot profita de mecanismele de lock. Atunci cand un client SMB deschide un fisier, specifica nivelul de acces (Write sau Read/Write) si modul de partajare. Daca modul de partajare nu este specificat atunci fisierul va fi blocat pana ce acesta va fi inchis. Daca un serviciu REST incercă sa acceseze un fisier in timp ce este blocat prin SMB, serviciul REST va returna un raspuns eroare cu codul 409 - conflict - SharingViolation. [8]

## **4.2 Accesul concurrential fisiere**

Serviciul de fisiere Azure se poate accesa in 2 modalitati: prin protocolul SMB

- Server Message Block sau prin REST - Representational State Transfer, prin HTTP.

### **○ SMB - Server Message Block**

Clientii SMB ce monteaza partitii de fisiere pot profita de mecanismele de lock are sistemelor de fisiere. Atunci cand un client SMB deschide un fisier, acesta specifica atat nivelul de acces al fisierului si modul de partajare. [9]

#### **➔ Niveluri de acces:**

- read - deschide fisierul doar pentru citiri
- write - deschide fisierul doar pentru scrieri
- read/write - deschide fisierul doar pentru citiri/scrieri
- delete - deschide fisierul doar pentru stergeri

#### **➔ Moduri de partajare:**

- none - nu se permite accesul la fisier pana cand acesta este inchis
- shared read - se permit deschideri repeatate pentru drepturi de citire
- shared write - se permit deschideri repeatate pentru drepturi de scriere
- shared read/write - se permit deschideri repeatate pentru drepturi de scriere/citire
- shared delete - se permit stergeri secentiale.

#### **Exemple:**

**Tabel 3 – Acces concurrential fara conflict:**

<i>Client A</i>	FileAccess.Read FileShare.Write	refuza read/delete	Nu exista conflict intre nivelurile de acces si modurile de partajare
<i>Client B</i>	FileAccess.Write FileShare.Read	refuza write/delete	

**Tabel 4 - Acces concurrential cu conflict cauzat de nivelul de acces:**

<i>Client A</i>	FileAccess.Write FileShare.Read	refuza write/delete	Clientul B are un conflict, cerand un nivel de acces interzis de clientul A
<i>Client B</i>	<b>FileAccess.Write</b> FileShare.Write	refuza read/delete	

**Tabel 5 - Acces concurrential cu conflict cauzat de modul de partajare:**

<i>Client A</i>	FileAccess.Write FileShare.Write	refuza read/delete	Clientul B are un conflict, cerand un mod de partajare ce interzice accesul la scrieri al unui fisier ce inca e deschis pentru acces de write. Clientul B cere doar read dar A deja scrie in el.
<i>Client B</i>	FileAccess.Write <b>FileShare.Read</b>	refuza write/delete	

- o **REST over HTTP**

Atunci cand se executa o cerere de la serviciu, aceste cereri trebuie sa respecte modurile de partajare specificate de clientii SMB. Apelurile de get corespund unor operatii de read, iar cele de set corespund unora de write.[9]

**Tabel 6 – Accesul concurrential prin REST read:**

<i>Client SMB</i>	FileAccess.Read FileShare.Write	refuza read/delete	Cererea clientului REST este eronata (cod 409 - conflict), clientul SMB inca are deschis fisierul si nu permite citiri in acest timp.
<i>Client REST</i>	Get File (FileAccess.Read)		

**Tabel 7 – Accesul concurrential prin REST write:**

<i>Client SMB</i>	FileAccess.Write FileShare.Read	refuza write/delete	Cererea clientului REST este eronata (cod 409 - conflict), clientul SMB inca are deschis fisierul si nu permite scrierii in acest timp.
<i>Client REST</i>	Get File (FileAccess.Write)		

Este posibil ca pentru unii clienti SMB pot seta atribute de fisiere: Archive, Read-only, Hidden, System. Daca un fisier este marcat cu Read-Only atunci orice operatie REST de write va fi eronata (cod 412 - Precondition Failed - Read Only Attribute). Aceste atribute se seta sau citi prin REST.

## **5. Concluzii**

Mecanismele folosite in WAS furnizeaza un mecanism de stocare pentru cantitati mari si variate de date. Sunt masiv scalabile, datele pot fi distribuite pe mai multe noduri iar accesul la date este controlat de mecanisme de load-balancing. Se furnizeaza un mecanism de persistenta fiabil: datele sunt replicate pe noduri de stocare diferite, in centre diferite – pana la 3 replicari.

Contul de storage este punctul de intrare pentru toate serviciile de stocare. WAS – Windows Azure Storage poate fi accesat de o aplicatie Windows Azure, de o aplicatie complexa sau de o aplicatie ce ruleaza in alt cloud. Defineste o interfata uniforma, toate operatiile pe resurse folosesc conventiile REST pentru identificare si expunerea datelor prin verbe HTTP

## Bibliografie

[1] *Introduction to Microsoft Azure Storage* <https://azure.microsoft.com/en-us/documentation/articles/storage-introduction/>

[2] [https://en.wikipedia.org/wiki/VHD\\_\(file\\_format\)](https://en.wikipedia.org/wiki/VHD_(file_format))

[3] *Azure Storage Scalability and Performance Targets*  
<https://azure.microsoft.com/en-us/documentation/articles/storage-scalability-targets/>

[4] *Manage anonymous read access to containers and blobs*  
<https://azure.microsoft.com/en-us/documentation/articles/storage-manage-access-to-resources/>

[5] *Shared Access Signatures, Part 1: Understanding the SAS model*  
<https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-shared-access-signature-part-1>

[6] *Authentication for the Azure Storage Services*  
<https://msdn.microsoft.com/library/azure/dd179428.aspx>

[7] *Azure Storage Services REST API Reference*  
<https://msdn.microsoft.com/en-us/library/azure/dd179355.aspx>

[8] *Managing Concurrency in Microsoft Azure Storage*  
<https://azure.microsoft.com/en-us/blog/managing-concurrency-in-microsoft-azure-storage-2/>

[9] *Managing File Locks* <https://msdn.microsoft.com/ro-ro/library/azure/dn194265.aspx>

[10] <https://www.gartner.com/doc/reprints?id=1-2G45TQU&ct=150519&st=sb>