

can you think of any ways to load balance lenel, specifically rabbit mq and open access. Im thinking potentially setting up a middle man server service that can route traffic between separate service instantiations

Load balancing Lenel OnGuard components, particularly RabbitMQ and OpenAccess, is definitely doable. Here are some approaches:

Direct RabbitMQ Clustering

RabbitMQ has built-in clustering and load balancing capabilities. You could:

- Set up a RabbitMQ cluster with multiple nodes
- Use HAProxy or similar load balancer in front of the cluster
- Configure OpenAccess clients to connect through the load balancer endpoint
- RabbitMQ's native mirrored queues can provide redundancy

Middleware/Proxy Approach

Your middleman server idea has merit:

- Build a lightweight service that acts as a message broker proxy
- It receives requests and routes them to available OnGuard instances based on load
- Could use round-robin, least-connections, or health-check based routing
- This gives you more control over routing logic specific to Lenel workflows

Application-Level Load Balancing

- Deploy multiple OpenAccess server instances
- Use DNS round-robin or a Layer 4 load balancer (F5, Nginx, HAProxy)
- Each instance connects to its own or shared database
- May need to handle session affinity for certain operations

Considerations Specific to Lenel

- **Licensing:** Check if your Lenel licenses support multiple concurrent OpenAccess instances
- **Database contention:** Multiple OpenAccess instances hitting the same SQL database may create bottlenecks
- **Event synchronization:** Ensure real-time events (door opens, alarms) are properly distributed
- **State management:** Some operations may require sticky sessions

What's your current bottleneck - is it the message queuing, the OpenAccess API processing, or database queries? That would help determine the best approach.

can you give me some more specifics on what specifically I need to be in the middle of for something like open access. Like I like changing the fqdn to be the load balanced version but how would that be configured client side and what type of app could I use or would I need to build to forward that traffic

Great question - let me break down the OpenAccess architecture and what you'd need to intercept:

OpenAccess Communication Protocol

OpenAccess typically uses:

- HTTPS/RESTAPI on port 8443 (default)
- LNVR video streams on various ports if you're doing video integration
- Database connections (SQL Server) - though this is server-side, not client-facing

The client applications (badge readers, third-party integrations, custom apps) make HTTPS requests to the OpenAccess server.

Load Balancing Options

Option 1: Standard Reverse Proxy (Simplest)

You can use existing tools without building custom middleware:

HAProxy (my recommendation):

```
haproxy
frontend openaccess_frontend
    bind *:8443 ssl crt /path/to/cert.pem
    mode http
    default_backend openaccess_servers

backend openaccess_servers
    mode http
    balance roundrobin
    option httpchk GET /api/access/onguard/openaccess
    server oa1 oa-server1.domain.com:8443 check ssl verify none
    server oa2 oa-server2.domain.com:8443 check ssl verify none
    server oa3 oa-server3.domain.com:8443 check ssl verify none
```

Nginx alternative:

```
nginx

upstream openaccess_cluster {
    least_conn;
    server oa-server1.domain.com:8443 max_fails=3 fail_timeout=30s;
    server oa-server2.domain.com:8443 max_fails=3 fail_timeout=30s;
    server oa-server3.domain.com:8443 max_fails=3 fail_timeout=30s;
}

server {
    listen 8443 ssl;
    server_name openaccess.yourdomain.com;

    ssl_certificate /path/to/cert.crt;
    ssl_certificate_key /path/to/cert.key;

    location / {
        proxy_pass https://openaccess_cluster;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_ssl_verify off;
    }
}
```

Option 2: Custom Middleware (More Control)

If you need custom routing logic (like routing specific badge readers to specific servers), you'd build something like:

PowerShell-based proxy (since you're comfortable with PS):

```
powershell

# Simple HTTP proxy concept
$listener = New-Object System.Net.HttpListener
$listener.Prefixes.Add("https://+:8443/")
$listener.Start()

$backends = @(
    "https://oa-server1:8443",
    "https://oa-server2:8443",
    "https://oa-server3:8443"
)
$currentBackend = 0

while ($listener.IsListening) {
    $context = $listener.GetContext()
    $request = $context.Request
    $response = $context.Response

    # Round-robin selection
    $targetServer = $backends[$currentBackend]
    $currentBackend = ($currentBackend + 1) % $backends.Count

    # Forward request
    $targetUrl = $targetServer + $request.Url.PathAndQuery
    try {
        $result = Invoke-RestMethod -Uri $targetUrl -Method $request.HttpMethod `

        -Headers $request.Headers -Body $request.InputStream

        # Send response back
        $buffer = [System.Text.Encoding]::UTF8.GetBytes($result)
        $response.ContentLength64 = $buffer.Length
        $response.OutputStream.Write($buffer, 0, $buffer.Length)
    }
    catch {
        $response.StatusCode = 502
    }
    finally {
        $response.Close()
    }
}
```

Or a more robust C# service that runs as a Windows service with health checking.

Client-Side Configuration

For Lenel Hardware (Badge Readers, Controllers)

In the **OnGuard System Administration**:
1. Go to the controller/panel configuration
2. Change the OpenAccess server address from 'oa-server1.domain.com' to 'openaccess'
3. The hardware will now connect to your load balancer FQDN

For Third-Party Integrations

Update their configuration files/connection strings:
- Old: 'https://oa-server1.domain.com:8443'
- New: 'https://openaccess-lb.domain.com:8443'

DNS Configuration

Create a DNS A record:
```
openaccess-lb.domain.com -> 10.0.1.100 (your load balancer IP)
```

Or use DNS round-robin (simpler but less sophisticated):
```

```
...
openaccess.domain.com -> 10.0.1.101
openaccess.domain.com -> 10.0.1.102
openaccess.domain.com -> 10.0.1.103
```

### Session Affinity Concerns

Some OpenAccess operations might require **sticky sessions**. If you see issues:

In HAProxy:

```
haproxy
backend openaccess_servers
 balance roundrobin
 cookie SERVERID insert indirect nocache
 server oa1 oa-server1:8443 check cookie oa1
 server oa2 oa-server2:8443 check cookie oa2
```

In Nginx:

```
nginx
upstream openaccess_cluster {
 ip_hash; # Same client IP always goes to same server
 server oa-server1:8443;
 server oa-server2:8443;
}
```

### My Recommendation

Start with **HAProxy** or **Nginx** rather than building custom middleware. They're battle-tested, handle SSL termination well, have health checking built-in, and you can always add custom logic later if needed.

The custom middleware approach makes sense if you need:

- Complex routing based on request content (like routing specific badge IDs to specific servers)
- Custom authentication/authorization layers
- Detailed logging of Lenel-specific operations
- Request transformation or filtering

Would you be load balancing for redundancy, performance, or both? That might affect whether you need active-active or active-passive setup.

More for performance, sometimes both of these services get overloaded and fail

Got it - performance bottlenecks causing service failures. Let me give you a more targeted approach for this scenario.

### Diagnosing the Bottleneck First

Before load balancing, it's worth identifying what's actually choking:

Quick PowerShell health check script:

```
powershell
Monitor OpenAccess and RabbitMQ resource usage
$servers = @("oa-server1", "rabbitmq-server1")

while ($true) {
 foreach ($server in $servers) {
 $cpu = (Get-Counter "\\\$server\Processor(_Total)\% Processor Time").CounterSamples.CookedValue
 $mem = (Get-Counter "\\\$server\Memory\Available MBytes").CounterSamples.CookedValue

 # Check OpenAccess process specifically
 $oaProcess = Get-Process -ComputerName $server -Name "LnlOpenAccess*" -ErrorAction SilentlyContinue
 if ($oaProcess) {
 Write-Host "$server - OA CPU: $($oaProcess.CPU)% | Memory: $($oaProcess.WorkingSet64)"
 }

 # Check RabbitMQ
 $rabbitProcess = Get-Process -ComputerName $server -Name "erl*" -ErrorAction SilentlyContinue
 if ($rabbitProcess) {
 Write-Host "$server - Rabbit CPU: $($rabbitProcess.CPU)% | Memory: $($rabbitProcess.WorkingSet64)"
 }
 }
 Start-Sleep -Seconds 5
}
```

### Load Balancing Strategy for Performance

For OpenAccess (HTTPS API Traffic)

HAProxy with health checks and circuit breaking:

```
haproxy
global
 maxconn 4096
 log /dev/log local0
```

```

defaults
 mode http
 timeout connect 5000ms
 timeout client 50000ms
 timeout server 50000ms
 option httplog
 option dontlognull

frontend openaccess_lb
 bind *:8443 ssl crt /etc/ssl/certs/openaccess.pem
 default_backend openaccess_pool

 # Rate limiting to prevent overload
 stick-table type ip size 100k expire 30s store http_req_rate(10s)
 http-request track-sc0 src
 http-request deny deny_status 429 if { sc_http_req_rate(0) gt 100 }

backend openaccess_pool
 balance leastconn # Send to least busy server
 option httpchk GET /api/access/onguard/openaccess/version
 http-check expect status 200

 # Aggressive health checking
 server oa1 10.0.1.101:8443 check inter 2000 rise 2 fall 3 maxconn 500 ssl verify
 server oa2 10.0.1.102:8443 check inter 2000 rise 2 fall 3 maxconn 500 ssl verify
 server oa3 10.0.1.103:8443 check inter 2000 rise 2 fall 3 maxconn 500 ssl verify

 # If server response time > 5s, mark as down temporarily
 timeout server 5000ms

```

#### Key settings for performance:

- `balance leastconn` - routes to server with fewest active connections
- `maxconn 500` - prevents any single server from being overwhelmed
- `inter 2000` - checks health every 2 seconds to quickly detect failures
- Rate limiting prevents cascading failures

#### For RabbitMQ (Message Queue Traffic)

RabbitMQ clustering with quorum queues is your best bet here:

##### Set up RabbitMQ cluster:

On each RabbitMQ server:

```

bash
Server 1 (primary)
rabbitmqctl stop_app
rabbitmqctl reset
rabbitmqctl start_app

Server 2
rabbitmqctl stop_app
rabbitmqctl reset
rabbitmqctl join_cluster rabbit@rabbitmq-server1
rabbitmqctl start_app

Server 3
rabbitmqctl stop_app
rabbitmqctl reset
rabbitmqctl join_cluster rabbit@rabbitmq-server1
rabbitmqctl start_app

Enable high availability policy
rabbitmqctl set_policy ha-all ".*" '{"ha-mode":"all","ha-sync-mode":"automatic"}'

```

#### HAProxy for RabbitMQ:

```

haproxy
frontend rabbitmq_frontend
 bind *:5672
 mode tcp
 default_backend rabbitmq_cluster

backend rabbitmq_cluster
 mode tcp
 balance roundrobin
 option tcp-check

 server rabbit1 10.0.1.201:5672 check inter 3000 rise 2 fall 3
 server rabbit2 10.0.1.202:5672 check inter 3000 rise 2 fall 3
 server rabbit3 10.0.1.203:5672 check inter 3000 rise 2 fall 3

```

#### Custom Middleware Approach (If You Want More Control)

Given your PowerShell experience, here's a more production-ready proxy service:

```

powershell
OpenAccessLoadBalancer.ps1
Run as a Windows Service using NSSM

param(
 [int]$ListenPort = 8443,
 [string[]]$BackendServers = @(
 "https://oa-server1.domain.com:8443",
 "https://oa-server2.domain.com:8443",
 "https://oa-server3.domain.com:8443"
)
)

```

```

 "https://oa-server3.domain.com:8443"
),
[string]$LogPath = "C:\Logs\OpenAccessLB.log"
)

Server health tracking
$serverHealth = @{}
foreach ($server in $BackendServers) {
 $serverHealth[$server] = @{
 IsHealthy = $true
 ActiveConnections = 0
 LastCheck = Get-Date
 FailureCount = 0
 }
}

function Write-LBLog {
 param($Message)
 $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
 "$timestamp - $Message" | Out-File -FilePath $LogPath -Append
}

function Test-ServerHealth {
 param($ServerUrl)

 try {
 $response = Invoke-WebRequest -Uri "$ServerUrl/api/access/onguard/openaccess"
 -TimeoutSec 3 -UseBasicParsing -ErrorAction Stop

 if ($response.StatusCode -eq 200) {
 $serverHealth[$ServerUrl].IsHealthy = $true
 $serverHealth[$ServerUrl].FailureCount = 0
 return $true
 }
 }
 catch {
 $serverHealth[$ServerUrl].FailureCount++
 if ($serverHealth[$ServerUrl].FailureCount -ge 3) {
 $serverHealth[$ServerUrl].IsHealthy = $false
 Write-LBLog "Server $ServerUrl marked unhealthy after 3 failures"
 }
 return $false
 }
}

function Get-BestServer {
 # Get healthy servers sorted by active connections
 $healthyServers = $serverHealth.GetEnumerator() |
 Where-Object { $_.Value.IsHealthy } |
 Sort-Object { $_.Value.ActiveConnections }

 if ($healthyServers.Count -eq 0) {
 Write-LBLog "ERROR: No healthy servers available!"
 return $null
 }

 return $healthyServers[0].Name
}

Health check background job
$healthCheckJob = Start-Job -ScriptBlock {
 param($servers, $healthHash)
 while ($true) {
 foreach ($server in $servers) {
 Test-ServerHealth -ServerUrl $server
 }
 Start-Sleep -Seconds 5
 }
} -ArgumentList $BackendServers, $serverHealth

Main proxy listener
$listener = New-Object System.Net.HttpListener
$listener.Prefixes.Add("https://+:$ListenPort/")
$listener.Start()

Write-LBLog "Load balancer started on port $ListenPort"

try {
 while ($listener.IsListening) {
 $context = $listener.GetContext()
 $request = $context.Request
 $response = $context.Response

 # Get best available server
 $targetServer = Get-BestServer

 if ($null -eq $targetServer) {
 $response.StatusCode = 503
 $response.StatusDescription = "Service Unavailable"
 $response.Close()
 continue
 }

 # Increment connection count
 $serverHealth[$targetServer].ActiveConnections++

 $targetUrl = $targetServer + $request.Url.PathAndQuery

 try {

```

```

Forward the request
$proxyRequest = @{
 Uri = $targetUrl
 Method = $request.HttpMethod
 TimeoutSec = 30
 UseBasicParsing = $true
}

Copy headers
$headers = @{}
foreach ($key in $request.Headers.Keys) {
 $headers[$key] = $request.Headers[$key]
}
$proxyRequest.Headers = $headers

Copy body if present
if ($request.HasEntityBody) {
 $reader = New-Object System.IO.StreamReader($request.InputStream)
 $body = $reader.ReadToEnd()
 $proxyRequest.Body = $body
}

$proxyResponse = Invoke-WebRequest @proxyRequest

Return response
$response.StatusCode = $proxyResponse.StatusCode
$buffer = $proxyResponse.RawContentStream
$buffer.CopyTo($response.OutputStream)

Write-LBLog "Routed $($request.HttpMethod) $($request.Url.PathAndQuery)"
}

catch {
 Write-LBLog "ERROR routing to $targetServer : $_"
 $response.StatusCode = 502
 $response.StatusDescription = "Bad Gateway"
}
finally {
 $serverHealth[$targetServer].ActiveConnections--
 $response.Close()
}
}

finally {
 $listener.Stop()
 Stop-Job -Job $healthCheckJob
 Remove-Job -Job $healthCheckJob
}

```

#### Install as a service:

```

powershell

Download NSSM (Non-Sucking Service Manager)
Then install the script as a service:

nssm install OpenAccessLB "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
 "-ExecutionPolicy Bypass -File C:\Scripts\OpenAccessLoadBalancer.ps1"

nssm set OpenAccessLB AppDirectory "C:\Scripts"
nssm set OpenAccessLB Start SERVICE_AUTO_START
nssm start OpenAccessLB

```

#### Quick Wins Before Full Load Balancing

1. Increase OpenAccess thread pool:
  - Edit OpenAccess configuration
  - Increase `MaxConcurrentRequests` setting
2. RabbitMQ tuning:

```

bash

In rabbitmq.conf
vm_memory_high_watermark.relative = 0.6
disk_free_limit.absolute = 50GB

```

3. Add connection pooling if your clients support it
4. Database optimization - often the real bottleneck:

```

sql

-- Check for missing indexes on Lenel tables
SELECT
 OBJECT_NAME(s.object_id) AS TableName,
 i.name AS IndexName,
 s.user_seeks, s.user_scans, s.user_lookups
FROM sys.dm_db_index_usage_stats s
INNER JOIN sys.indexes i ON s.object_id = i.object_id AND s.index_id = i.index_id
WHERE database_id = DB_ID('OnGuard')
ORDER BY s.user_seeks DESC

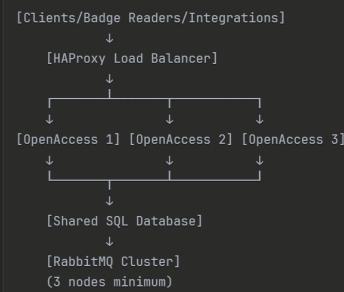
```

Would you be able to spin up additional VM instances for OpenAccess/RabbitMQ, or are you limited to the existing servers? That would determine whether you go with true load balancing or just implement better queuing/throttling on existing infrastructure.

I could spin up multiple vms

Perfect - then you can do proper horizontal scaling. Here's a complete implementation plan:

## Architecture Overview



## Step-by-Step Implementation

### Phase 1: RabbitMQ Cluster (Do This First)

RabbitMQ clustering is critical since OpenAccess depends on it.

#### Server Setup:

- VM1: rabbitmq-01.domain.com (10.0.1.201)
- VM2: rabbitmq-02.domain.com (10.0.1.202)
- VM3: rabbitmq-03.domain.com (10.0.1.203)

#### Installation on each node:

```
powershell

Install Erlang first (RabbitMQ dependency)
choco install erlang -y

Install RabbitMQ
choco install rabbitmq -y

Enable management plugin
rabbitmq-plugins enable rabbitmq_management

Set erlang cookie (must be IDENTICAL on all nodes)
$cookieFile = "$env:USERPROFILE\.erlang.cookie"
"YOURSECRETCOOKIEHERE" | Out-File -FilePath $cookieFile -Encoding ASCII -NoNewline

Restart service
Restart-Service RabbitMQ
```

#### Create the cluster:

```
powershell

On rabbitmq-02 and rabbitmq-03, run:
rabbitmqctl stop_app
rabbitmqctl join_cluster rabbit@rabbitmq-01
rabbitmqctl start_app

Verify cluster status (run on any node)
rabbitmqctl cluster_status
```

#### Configure high availability policy:

```
powershell

This ensures queues are replicated across all nodes
rabbitmqctl set_policy ha-all "^" '{"ha-mode":"all","ha-sync-mode":"automatic","ha-s

For quorum queues (recommended for critical queues)
rabbitmqctl set_policy quorum-policy "^(quorum)." '{"queue-type":"quorum","delivery-l
```

#### RabbitMQ HAProxy Config:

```
haproxy
/etc/haproxy/haproxy.cfg

listen rabbitmq_cluster
 bind *:5672
 mode tcp
 balance roundrobin
 option tcplog

 server rabbitmq-01 10.0.1.201:5672 check inter 5s rise 2 fall 3
 server rabbitmq-02 10.0.1.202:5672 check inter 5s rise 2 fall 3
 server rabbitmq-03 10.0.1.203:5672 check inter 5s rise 2 fall 3

listen rabbitmq_mgmt
 bind *:15672
 mode http
 balance roundrobin

 location /status/rabbitmq-01 10.0.1.201:15672 check
```

```
server rabbitmq-01 10.0.1.201:15672 check
server rabbitmq-02 10.0.1.202:15672 check
server rabbitmq-03 10.0.1.203:15672 check
```

## Phase 2: OpenAccess Instances

### Server Setup:

- VM1: openaccess-01.domain.com (10.0.1.101)
- VM2: openaccess-02.domain.com (10.0.1.102)
- VM3: openaccess-03.domain.com (10.0.1.103)

Install Lenel OpenAccess on each VM (standard installation)

Key configuration changes for each instance:

```
xml
<!-- In OpenAccess config file (typically C:\Program Files\Lenel\OnGuard\OpenAccess)\-->
<configuration>
 <connectionStrings>
 <!-- All instances point to SAME database -->
 <add name="OnGuardConnection"
 connectionString="Server=sql-server.domain.com;Database=OnGuard;Integrated
 /></connectionStrings>

 <appSettings>
 <!-- Point to RabbitMQ load balancer -->
 <add key="RabbitMQHost" value="rabbitmq-lb.domain.com" />
 <add key="RabbitMQPort" value="5672" />

 <!-- Increase thread pool for performance -->
 <add key="MaxConcurrentRequests" value="100" />
 <add key="RequestQueueLimit" value="500" />

 <!-- Unique identifier for each instance (important for logging) -->
 <add key="InstanceId" value="OA-01" /> <!-- Change per server -->
 </appSettings>
</configuration>
```

Performance tuning on each OpenAccess server:

```
powershell
Increase IIS application pool settings
Import-Module WebAdministration

$appPoolName = "OpenAccessAppPool"

Increase queue length
Set-ItemProperty "IIS:\AppPools\$appPoolName" -Name queueLength -Value 2000

Disable idle timeout
Set-ItemProperty "IIS:\AppPools\$appPoolName" -Name processModel.idleTimeout -Value

Increase max worker processes
Set-ItemProperty "IIS:\AppPools\$appPoolName" -Name processModel.maxProcesses -Value

Memory limits
Set-ItemProperty "IIS:\AppPools\$appPoolName" -Name recycling.periodicRestart.memory

Restart-WebAppPool -Name $appPoolName
```

## Phase 3: HAProxy Load Balancer

Single dedicated VM for HAProxy:

- openaccess-lb.domain.com (10.0.1.100)

Install HAProxy on Windows:

```
powershell
Using chocolatey
choco install haproxy -y

Or download from: https://www.haproxy.org/download/windows/
```

Complete HAProxy configuration:

```
haproxy
C:\ProgramData\HAProxy\haproxy.cfg

global
 maxconn 10000
 log 127.0.0.1 local0
 stats socket /var/run/haproxy.sock mode 660 level admin
 stats timeout 30s

defaults
 log global
 mode http
 option httplog
 option dontlognull
 timeout connect 10s
 timeout client 60s
 timeout server 60s
 retries 3
 option redispatch
```

```

Statistics page
listen stats
 bind *:8404
 stats enable
 stats uri /stats
 stats refresh 10s
 stats admin if TRUE

OpenAccess HTTPS Frontend
frontend openaccess_https
 bind *:8443 ssl crt /path/to/combined-cert.pem
 mode http

 # Logging
 option httplog
 log-format "%ci:%cp [%tr] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Ta %ST %B %CC %CS %tsc %ac/"

 # Security headers
 http-response set-header X-Frame-Options SAMEORIGIN
 http-response set-header X-Content-Type-Options nosniff

 # Rate limiting per IP
 stick-table type ip size 100k expire 30s store http_req_rate(10s),http_err_rate(
 http-request track-sc0 src
 http-request deny deny_status 429 if { sc_http_req_rate(0) gt 200 }

 default_backend openaccess_pool

OpenAccess Backend Pool
backend openaccess_pool
 mode http
 balance leastconn # Route to server with fewest connections

 # Health check
 option httpchk GET /api/access/onguard/openaccess/version
 http-check expect status 200

 # Cookie-based session persistence (if needed)
 cookie SERVERID insert indirect nocache

 # Server definitions
 server oa-01 10.0.1.101:8443 ssl verify none check inter 3s rise 2 fall 5 maxcon
 server oa-02 10.0.1.102:8443 ssl verify none check inter 3s rise 2 fall 5 maxcon
 server oa-03 10.0.1.103:8443 ssl verify none check inter 3s rise 2 fall 5 maxcon

 # Backup server (if all primaries fail)
 # server oa-backup 10.0.1.104:8443 ssl verify none check backup

RabbitMQ Frontend
frontend rabbitmq_amqp
 bind *:5672
 mode tcp
 default_backend rabbitmq_cluster

backend rabbitmq_cluster
 mode tcp
 balance roundrobin
 option tcp-check

 server rabbit-01 10.0.1.201:5672 check inter 5s
 server rabbit-02 10.0.1.202:5672 check inter 5s
 server rabbit-03 10.0.1.203:5672 check inter 5s

RabbitMQ Management Interface
frontend rabbitmq_mgmt
 bind *:15672
 mode http
 default_backend rabbitmq_mgmt_cluster

backend rabbitmq_mgmt_cluster
 mode http
 balance roundrobin
 option httpchk GET /api/alive ness-test/%2F
 http-check expect status 200

 server rabbit-01 10.0.1.201:15672 check
 server rabbit-02 10.0.1.202:15672 check
 server rabbit-03 10.0.1.203:15672 check

```

#### Install as Windows service:

```

powershell

Create service
New-Service -Name "HAProxy" `
 -BinaryPathName "C:\Program Files\HAProxy\haproxy.exe" -f C:\ProgramData\HAProxy\`
 -DisplayName "HAProxy Load Balancer" `
 -StartupType Automatic

Start-Service HAProxy

```

#### Phase 4: DNS Configuration

```

powershell

In your DNS server, create:
A record for load balancer

```

```
Add-DnsServerResourceRecordA -Name "openaccess-lb" -ZoneName "domain.com" -IPv4Address
A record for RabbitMQ load balancer (can be same server)
Add-DnsServerResourceRecordA -Name "rabbitmq-lb" -ZoneName "domain.com" -IPv4Address
Optional: CNAME for easier migration
Add-DnsServerResourceRecordCName -Name "openaccess" -HostNameAlias "openaccess-lb.domain.com" -ZoneName "domain.com"
```

#### Phase 5: Client Configuration Changes

##### Update Lenel hardware/controllers:

1. Open OnGuard System Administration
2. Navigate to Hardware → Communication Servers
3. Change OpenAccess server address:
  - Old: [openaccess-01.domain.com:8443](http://openaccess-01.domain.com:8443)
  - New: [openaccess-lb.domain.com:8443](http://openaccess-lb.domain.com:8443)

##### Update third-party integrations:

```
powershell

Example: Update connection strings in config files
$configFiles = Get-ChildItem -Path "C:\Integrations" -Filter "*.config" -Recurse

foreach ($file in $configFiles) {
 (Get-Content $file.FullName) `
 -replace 'openaccess-01\domain.com', 'openaccess-lb.domain.com' `
 -replace 'rabbitmq-01\domain.com', 'rabbitmq-lb.domain.com' |
 Set-Content $file.FullName
}
```

#### Phase 6: Monitoring & Alerting

##### Monitoring script for the entire stack:

```
powershell

ClusterHealthMonitor.ps1

$config = @{
 HAProxyStats = "http://openaccess-lb.domain.com:8404/stats"
 OpenAccessServers = @(
 "https://openaccess-01.domain.com:8443/api/access/onguard/openaccess/version"
 "https://openaccess-02.domain.com:8443/api/access/onguard/openaccess/version"
 "https://openaccess-03.domain.com:8443/api/access/onguard/openaccess/version"
)
 RabbitMQNodes = @(
 "http://rabbitmq-01.domain.com:15672/api/nodes",
 "http://rabbitmq-02.domain.com:15672/api/nodes",
 "http://rabbitmq-03.domain.com:15672/api/nodes"
)
 AlertEmail = "admin@domain.com"
 SMTPServer = "smtp.domain.com"
}

function Send-Alert {
 param($Subject, $Body)

 Send-MailMessage -To $config.AlertEmail `
 -From "lenel-monitor@domain.com" `
 -Subject $Subject `
 -Body $Body `
 -SmtpServer $config.SMTPServer
}

function Test-OpenAccessHealth {
 $results = @()

 foreach ($server in $config.OpenAccessServers) {
 try {
 $response = Invoke-RestMethod -Uri $server -TimeoutSec 5
 $results += [PSCustomObject]@{
 Server = $server
 Status = "Healthy"
 ResponseTime = (Measure-Command { Invoke-RestMethod -Uri $server }).TotalSeconds
 }
 }
 catch {
 $results += [PSCustomObject]@{
 Server = $server
 Status = "Down"
 Error = $_.Exception.Message
 }
 Send-Alert "OpenAccess Server Down" "Server $server is not responding: $Error"
 }
 }

 return $results
}

function Test-RabbitMQHealth {
 $results = @()
 $cred = Get-Credential # Store credentials securely

 foreach ($node in $config.RabbitMQNodes) {
 try {
 $response = Invoke-RestMethod -Uri $node -Credential $cred -TimeoutSec 5
 }
 catch {
 $results += [PSCustomObject]@{
 Node = $node
 Status = "Down"
 Error = $_.Exception.Message
 }
 Send-Alert "RabbitMQ Node Down" "Node $node is not responding: $Error"
 }
 }

 return $results
}
```

```

 $results += [PSCustomObject]@{
 Node = $node
 Status = "Running"
 MemoryUsed = $response[0].mem_used
 MemoryLimit = $response[0].mem_limit
 DiskFree = $response[0].disk_free
 }
 }
 catch {
 $results += [PSCustomObject]@{
 Node = $node
 Status = "Down"
 Error = $_.Exception.Message
 }
 Send-Alert "RabbitMQ Node Down" "Node $node is not responding"
 }
}

return $results
}

function Get-HAProxyStats {
 try {
 $stats = Invoke-WebRequest -Uri $config.HAProxyStats -UseBasicParsing
 # Parse HAProxy stats page
 # (Implementation depends on your stats format)
 return $stats
 }
 catch {
 Send-Alert "HAProxy Unreachable" "Cannot retrieve HAProxy statistics"
 }
}

Main monitoring loop
while ($true) {
 Write-Host "`n== Health Check $(Get-Date) ==="

 $oaHealth = Test-OpenAccessHealth
 $oaHealth | Format-Table -AutoSize

 $rabbitHealth = Test-RabbitMQHealth
 $rabbitHealth | Format-Table -AutoSize

 # Check for overall cluster health
 $healthyOA = ($oaHealth | Where-Object { $_.Status -eq "Healthy" }).Count
 $healthyRabbit = ($rabbitHealth | Where-Object { $_.Status -eq "Running" }).Count

 if ($healthyOA -eq 0) {
 Send-Alert "CRITICAL: All OpenAccess Servers Down" "Immediate attention required"
 }
 elseif ($healthyOA -lt 2) {
 Send-Alert "WARNING: Only $healthyOA OpenAccess server(s) healthy" "Consider action"
 }

 if ($healthyRabbit -lt 2) {
 Send-Alert "WARNING: RabbitMQ cluster degraded" "Only $healthyRabbit node(s) running"
 }

 Start-Sleep -Seconds 60
}

```

#### Install as scheduled task:

```

powershell

$action = New-ScheduledTaskAction -Execute "PowerShell.exe" `
 -Argument "-ExecutionPolicy Bypass -File C:\Scripts\ClusterHealthMonitor.ps1"

$trigger = New-ScheduledTaskTrigger -AtStartup

$principal = New-ScheduledTaskPrincipal -UserId "SYSTEM" -LogonType ServiceAccount

Register-ScheduledTask -TaskName "LenelClusterMonitor" `
 -Action $action `
 -Trigger $trigger `
 -Principal $principal

```

#### Testing & Validation

##### Load testing script:

```

powershell

LoadTest.ps1 - Simulate multiple clients

$endpoint = "https://openaccess-lb.domain.com:8443/api/access/onguard/openaccess/car"
$concurrent = 50
$iterations = 100

$jobs = 1..$concurrent | ForEach-Object {
 Start-Job -ScriptBlock {
 param($url, $count)

 $results = @()
 for ($i = 0; $i -lt $count; $i++) {
 $sw = [System.Diagnostics.Stopwatch]::StartNew()
 try {
 $response = Invoke-RestMethod -Uri $url -Method GET -TimeoutSec 30
 $results += $response
 }
 catch {
 $results += $_.Exception.Message
 }
 }
 $sw.Stop()
 $results += $sw.Elapsed
 }
}

```

```

$results += [PSCustomObject]@{
 Success = $true
 ResponseTime = $sw.ElapsedMilliseconds
}
}
catch {
 $results += [PSCustomObject]@{
 Success = $false
 Error = $_.Exception.Message
 }
}
$sw.Stop()
}
return $results
} -ArgumentList $endpoint, $iterations
}

Wait for all jobs and collect results
$allResults = $jobs | Wait-Job | Receive-Job
$jobs | Remove-Job

Analyze
$successful = ($allResults | Where-Object { $_.Success }).Count
$failed = ($allResults | Where-Object { -not $_.Success }).Count
$avgResponseTime = ($allResults | Where-Object { $_.Success }) | Measure-Object -Prop
Write-Host "Load Test Results:"
Write-Host "Successful requests: $successful"
Write-Host "Failed requests: $failed"
Write-Host "Average response time: $($avgResponseTime)ms"

```

### Rollback Plan

Keep your original single-server configs backed up:

```

powershell

Backup current configs
$backupPath = "C:\Backups\PreLoadBalancing_$(Get-Date -Format 'yyyyMMdd')"
New-Item -ItemType Directory -Path $backupPath

Backup OpenAccess config
Copy-Item "C:\Program Files\Lenel\OnGuard\OpenAccess\web.config" `
-Destination "$backupPath\web.config.bak"

Export DNS records
Get-DnsServerResourceRecord -ZoneName "domain.com" |
Export-Csv "$backupPath\dns_records.csv"

```

This setup should handle your performance issues by distributing load across multiple servers. The key is that RabbitMQ clustering provides message queue redundancy, and OpenAccess instances can scale horizontally since they're stateless (state is in the database).

What's your timeline for this? I'd recommend doing it in phases - RabbitMQ first, then add OpenAccess instances one at a time to minimize risk.



Reply...



Sonnet 4.5



Claude is AI and can make mistakes. Please double-check responses.