Alex Auvolat, Deuxfleurs Association

https://garagehq.deuxfleurs.fr/
Matrix channel: #garage:deuxfleurs.fr
Who I am

Alex Auvolat
PhD; co-founder of Deuxfleurs

Deuxfleurs
A non-profit self-hosting collective, member of the CHATONS network
Our objective at Deuxfleurs

Promote self-hosting and small-scale hosting as an alternative to large cloud providers
Our objective at Deuxfleurs

Promote self-hosting and small-scale hosting as an alternative to large cloud providers

Why is it hard?
Our objective at Deuxfleurs

Promote self-hosting and small-scale hosting as an alternative to large cloud providers

Why is it hard?

Resilience
we want good uptime/availability with low supervision
Building a resilient system with cheap stuff

- Commodity hardware (e.g. old desktop PCs)
Building a resilient system with cheap stuff
Building a resilient system with cheap stuff
Building a resilient system with cheap stuff

- Commodity hardware (e.g. old desktop PCs)
  (can die at any time)

- Regular Internet (e.g. FTTB, FTTH) and power grid connections
  (can be unavailable randomly)

- Geographical redundancy (multi-site replication)
Building a resilient system with cheap stuff

- Commodity hardware (e.g. old desktop PCs)
  (can die at any time)

- Regular Internet (e.g. FTTB, FTTH) and power grid connections
Building a resilient system with cheap stuff

- Commodity hardware (e.g. old desktop PCs)
  (can die at any time)

- Regular Internet (e.g. FTTB, FTTH) and power grid connections
  (can be unavailable randomly)
Building a resilient system with cheap stuff

- Commodity hardware (e.g. old desktop PCs)
  (can die at any time)

- Regular Internet (e.g. FTTB, FTTH) and power grid connections
  (can be unavailable randomly)

- Geographical redundancy (multi-site replication)
Building a resilient system with cheap stuff
Object storage: a crucial component

S3: a de-facto standard, many compatible applications
Object storage: a crucial component

Amazon S3

MinIO

S3: a de-facto standard, many compatible applications

MinIO is self-hostable but not suited for geo-distributed deployments
Object storage: a crucial component

Amazon S3

MinIO

Garage

S3: a de-facto standard, many compatible applications

MinIO is self-hostable but not suited for geo-distributed deployments

**Garage is a self-hosted drop-in replacement for the Amazon S3 object store**
CRDTs / weak consistency instead of consensus

Internally, Garage uses only CRDTs (conflict-free replicated data types)

Why not Raft, Paxos, ...? Issues of consensus algorithms:
CRDTs / weak consistency instead of consensus

Internally, Garage uses only CRDTs (conflict-free replicated data types)

Why not Raft, Paxos, ...? Issues of consensus algorithms:

► Software complexity
CRDTs / weak consistency instead of consensus

Internally, Garage uses only CRDTs (conflict-free replicated data types)

Why not Raft, Paxos, ...? Issues of consensus algorithms:

- **Software complexity**
- **Performance issues:**
CRDTs / weak consistency instead of consensus

Internally, Garage uses only CRDTs (conflict-free replicated data types)

Why not Raft, Paxos, ...? Issues of consensus algorithms:

- **Software complexity**

- **Performance issues:**
  - The leader is a **bottleneck** for all requests
CRDTs / weak consistency instead of consensus

Internally, Garage uses only CRDTs (conflict-free replicated data types)

Why not Raft, Paxos, ...? Issues of consensus algorithms:

- **Software complexity**

- **Performance issues:**
  - The leader is a bottleneck for all requests
  - Sensitive to higher latency between nodes
CRDTs / weak consistency instead of consensus

Internally, Garage uses only CRDTs (conflict-free replicated data types)

Why not Raft, Paxos, ...? Issues of consensus algorithms:

- **Software complexity**

- **Performance issues:**
  - The leader is a **bottleneck** for all requests
  - **Sensitive to higher latency** between nodes
  - **Takes time to reconverge** when disrupted (e.g. node going down)
The data model of object storage

Object storage is basically a **key-value store**:

<table>
<thead>
<tr>
<th>Key: file path + name</th>
<th>Value: file data + metadata</th>
</tr>
</thead>
</table>
| index.html            | Content-Type: text/html; charset=utf-8  
  Content-Length: 24929  
  <binary blob>         |
| img/logo.svg          | Content-Type: text/svg+xml  
  Content-Length: 13429  
  <binary blob>         |
| download/index.html   | Content-Type: text/html; charset=utf-8  
  Content-Length: 26563  
  <binary blob>         |
The data model of object storage

Object storage is basically a **key-value store**:

<table>
<thead>
<tr>
<th>Key: file path + name</th>
<th>Value: file data + metadata</th>
</tr>
</thead>
</table>
| index.html            | Content-Type: text/html; charset=utf-8  
Content-Length: 24929  
<binary blob>           |
| img/logo.svg          | Content-Type: text/svg+xml  
Content-Length: 13429  
<binary blob>           |
| download/index.html   | Content-Type: text/html; charset=utf-8  
Content-Length: 26563  
<binary blob>           |

▶ Maps well to CRDT data types
Performance gains in practice

S3 endpoint latency in a simulated geo-distributed cluster

100 measurements, 5 nodes, 50ms RTT + 10ms jitter between nodes
no contention: latency is due to intra-cluster communications
colored bar = mean latency, error bar = min and max latency

S3 Endpoint

- removeobject
- putobject
- listobjects
- listbuckets
- getobject

Daemon
- garage 0.7.3
- garage 0.8.0 beta
- minio RELEASE.2022-09-17

Request duration (ms)

Get the code to reproduce this graph at https://git.deuxfleurs.fr/Deuxfleurs/mknet
Recent developments
Focus on observability and ecosystem integration

- **Monitoring**: metrics and traces, using OpenTelemetry
- Replication modes with 1 or 2 copies / weaker consistency
- Kubernetes integration for node discovery
- Admin API (v0.7.2)
Metrics (Prometheus + Grafana)
Traces (Jaeger)

Alex Auvolat, Deuxfleurs
Garage, the low-tech storage platform for geo-distributed clusters
FOSDEM’24, 2024-02-03
v0.6.0  
Feb 2, 2022

v0.7.0  
Apr 8, 2022

v0.8.0  
Nov 21, 2022

v0.9.0  
Oct 10, 2023

Towards v1.0 ?  
Apr/May, 2024

v0.10.0 beta

FOSDEM’22  
Feb 6, 2022

Capitole du Libre  
Nov 19, 2022

PSES  
Jun 6, 2023

FOSDEM’24  
Feb 3, 2024
Focus on performance

- **Alternative metadata DB engines** (LMDB, Sqlite)
- **Performance improvements**: block streaming, various optimizations...
- Bucket quotas (max size, max #objects)
- Quality of life improvements, observability, etc.
About metadata DB engines

Issues with Sled:

- Huge files on disk
- Unpredictable performance, especially on HDD
- API limitations
- Not actively maintained

LMDB: very stable, good performance, file size is reasonable
Sqlite also available as a second choice

Sled will be removed in Garage v1.0
DB engine performance comparison

Comparison of Garage’s metadata engines with "minio/warp"
Daemon: Garage v0.8 no-fsync to avoid being impacted by block manager
Benchmark: warp, mixed mode, 5min bench, 256B objects, initialized with 200 objects.
Environment: mknet (Ryzen 5 1400, 16GB RAM, SSD). DC topo (3 nodes, 1Gb/s, 1ms latency).

- SQLite
- sled-nosync
- sled
- Imdb

Get the code to reproduce this graph at https://git.deuxfleurs.fr/Deuxfleurs/mknet

NB: Sqlite was slow due to synchronous mode, now configurable
Block streaming

Without Block Streaming

Garage Node (handle req)

RPC Request

1MB Block

Garage Node (store block)

RPC Response

Block

S3 Request

S3 Response

Alex Auvolat, Deuxfleurs

Garage, the low-tech storage platform for geo-distributed clusters

FOSDEM'24, 2024-02-03
Block streaming

With Block Streaming

Garage Node (handle req)

Garage Node (store block)

S3 Request

RPC Request

S3 Response

RPC Response

Block

1MB Block
TTFB benchmark

TTFB (Time To First Byte) on GetObject over a slow network (5 Mbps, 500 μs)
A 1MB file is uploaded and then fetched 60 times.
Except for Minio, the queried node does not store any data (gateway) to force net. communications.

Get the code to reproduce this graph at https://git.deuxfleurs.fr/Deuxfleurs/mknet
Throughput benchmark

"minio/warp" benchmark, "cluster total" result
Ran on a local machine (Ryzen 5 1400, 16GB RAM, SSD) with mknet
DC topology (3 nodes, 1GB/s, 1ms lat)
warp in mixed mode, 5min bench, 5MB objects, initialized with 200 objects

Get the code to reproduce this graph at https://git.deuxfleurs.fr/Deuxfleurs/mknet
Focus on streamlining & usability

- Support multiple HDDs per node
- S3 compatibility:
  - support basic lifecycle configurations
  - allow for multipart upload part retries
- LMDB by default, deprecation of Sled
- New layout computation algorithm
Garage stores replicas on different zones when possible
Layout computation

Garage stores replicas on different zones when possible

Each chunk of data is replicated in 3 zones
What a "layout" is

A layout is a precomputed index table:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>df-ymk (bespin)</td>
<td>Abricot (scorpio)</td>
<td>Courgette (neptune)</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas (scorpio)</td>
<td>Courgette (neptune)</td>
<td>df-ykl (bespin)</td>
</tr>
<tr>
<td>Partition 2</td>
<td>df-ymf (bespin)</td>
<td>Celeri (neptune)</td>
<td>Abricot (scorpio)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Partition 255</td>
<td>Concombre (neptune)</td>
<td>df-ykl (bespin)</td>
<td>Abricot (scorpio)</td>
</tr>
</tbody>
</table>

The index table is built centrally using an optimal algorithm, then propagated to all nodes.

What a "layout" is

A layout is a precomputed index table:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>df-ymk (bespin)</td>
<td>Abricot (scorpio)</td>
<td>Courgette (neptune)</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas (scorpio)</td>
<td>Courgette (neptune)</td>
<td>df-ykl (bespin)</td>
</tr>
<tr>
<td>Partition 2</td>
<td>df-ymf (bespin)</td>
<td>Celeri (neptune)</td>
<td>Abricot (scorpio)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition 255</td>
<td>Concombre (neptune)</td>
<td>df-ykl (bespin)</td>
<td>Abricot (scorpio)</td>
</tr>
</tbody>
</table>

The index table is built centrally using an optimal algorithm, then propagated to all nodes.
What a "layout" is

A layout is a precomputed index table:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>df-ymk (bespin)</td>
<td>Abricot (scorpio)</td>
<td>Courgette (neptune)</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas (scorpio)</td>
<td>Courgette (neptune)</td>
<td>df-ykl (bespin)</td>
</tr>
<tr>
<td>Partition 2</td>
<td>df-ymf (bespin)</td>
<td>Celeri (neptune)</td>
<td>Abricot (scorpio)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition 255</td>
<td>Concombre (neptune)</td>
<td>df-ykl (bespin)</td>
<td>Abricot (scorpio)</td>
</tr>
</tbody>
</table>

The index table is built centrally using an optimal algorithm, then propagated to all nodes.

Focus on consistency

- Fix consistency issues when reshuffling data
Working with weak consistency

Not using consensus limits us to the following:

- Conflict-free replicated data types (CRDT)
  - Non-transactional key-value stores such as S3 are equivalent to a simple CRDT: a map of last-writer-wins registers (each key is its own CRDT)
- Read-after-write consistency
  - Can be implemented using quorums on read and write operations
Working with weak consistency

Not using consensus limits us to the following:

- **Conflict-free replicated data types** (CRDT)

  Non-transactional key-value stores such as S3 are equivalent to a simple CRDT: a map of **last-writer-wins registers** (each key is its own CRDT)
Working with weak consistency

Not using consensus limits us to the following:

- **Conflict-free replicated data types (CRDT)**
  
  Non-transactional key-value stores such as S3 are equivalent to a simple CRDT: a map of **last-writer-wins registers** (each key is its own CRDT)

- **Read-after-write consistency**
  
  Can be implemented using quorums on read and write operations
CRDT read-after-write consistency using quorums

**Property:** If client 1 did an operation \( write(x) \) and received an OK response, and client 2 starts an operation \( read() \) after client 1 received OK, then client 2 will read a value \( x' \sqsubseteq x \).
**Property:** If client 1 did an operation \( \text{write}(x) \) and received an OK response, and client 2 starts an operation \( \text{read()} \) after client 1 received OK, then client 2 will read a value \( x' \sqsubseteq x \).
CRDT read-after-write consistency using quorums

**Property:** If client 1 did an operation \( \text{write}(x) \) and received an OK response, and client 2 starts an operation \( \text{read}() \) after client 1 received OK, then client 2 will read a value \( x' \sqsupseteq x \).

\[
\text{write}\{\{a\}\}:
\]

- \( \not\sqsubseteq \{a\} \)
- \( \not\sqsubseteq \{a\} \)
- \( \not\sqsubseteq \{a\} \)

\[
\begin{align*}
\{a\} & \rightarrow \{a, b\} & \rightarrow & \{a, b, c\} \\
\{a\} & \rightarrow \{a, c\} & \rightarrow & \{a, b, c\} \\
\{b\} & \rightarrow & \rightarrow & \{b, c\} \\
\{c\} & \rightarrow & \rightarrow & \{c\} \\
\{\} & \rightarrow & \rightarrow & \{\} \\
\end{align*}
\]
**Property:** If client 1 did an operation $\text{write}(x)$ and received an OK response, and client 2 starts an operation $\text{read}()$ after client 1 received OK, then client 2 will read a value $x' \sqsubseteq x$. 

\[
\text{write}\{\{a\}\}:
\begin{align*}
\bullet & \sqsubseteq \{a\} \rightarrow \text{OK} \\
\bullet & \not\sqsubseteq \{a\} \\
\bullet & \not\sqsubseteq \{a\}
\end{align*}
\]
CRDT read-after-write consistency using quorums

**Property:** If client 1 did an operation \( \text{write}(x) \) and received an OK response, and client 2 starts an operation \( \text{read()} \) after client 1 received OK, then client 2 will read a value \( x' \sqsupseteq x \).
**CRDT read-after-write consistency using quorums**

**Property:** If client 1 did an operation \( \text{write}(x) \) and received an OK response, and client 2 starts an operation \( \text{read}() \) after client 1 received OK, then client 2 will read a value \( x' \supseteq x \).

\[
\begin{align*}
\text{write}(\{a\}): \\
\& \quad \boxed{\{a\}} \rightarrow \text{OK} \\
\& \quad \boxed{\{a\}} \rightarrow \text{OK} \\
\& \quad \boxed{\{a\}} \\
\text{return} \quad \text{OK}
\end{align*}
\]

\[
\begin{align*}
\text{read}(): \\
\{a\} \\
\{a\} \\
\{b\} \\
\{c\} \\
\end{align*}
\]
**Property:** If client 1 did an operation \( write(x) \) and received an OK response, and client 2 starts an operation \( read() \) after client 1 received OK, then client 2 will read a value \( x' \sqsupseteq x \).
CRDT read-after-write consistency using quorums

**Property:** If client 1 did an operation $write(x)$ and received an OK response, and client 2 starts an operation $read()$ after client 1 received OK, then client 2 will read a value $x' \sqsupseteq x$.

\[\text{write}\(\{a\}\):}\
\begin{align*}
\bullet \sqsupseteq \{a\} & \rightarrow \text{OK} \\
\bullet \sqsupset \{a\} & \rightarrow \text{OK} \\
\bullet \nsubseteq \{a\} & \rightarrow \text{OK} \\
\text{return } \text{OK}
\end{align*}

\[\text{read():}\
\begin{align*}
\bullet & \rightarrow \{\} \\
\bullet & \rightarrow \{a\} \\
\text{return } \{\} \sqcup \{a\} & = \{a\}
\end{align*}
**Property:** If client 1 did an operation $\text{write}(x)$ and received an OK response, and client 2 starts an operation $\text{read}()$ after client 1 received OK, then client 2 will read a value $x' \sqsubseteq x$. 

\[
\text{write}({a}): \\
\quad \bullet \sqsubseteq \{a\} \rightarrow \text{OK} \\
\quad \bullet \sqsubseteq \{a\} \rightarrow \text{OK} \\
\quad \bullet \sqsubseteq \{a\} \\
\text{return OK}
\]

\[
\text{read}(): \\
\quad \bullet \rightarrow \{\} \\
\quad \bullet \rightarrow \{a\} \\
\text{return } \{\} \sqcup \{a\} = \{a\}
\]
CRDT read-after-write consistency using quorums

Property: If client 1 did an operation \( write(x) \) and received an OK response, and client 2 starts an operation \( read() \) after client 1 received OK, then client 2 will read a value \( x' \sqsupseteq x \).

Algorithm \( write(x) \):
1. Broadcast \( write(x) \) to all nodes
2. Wait for \( k > n/2 \) nodes to reply OK
3. Return OK

Algorithm \( read() \):
1. Broadcast \( read() \) to all nodes
2. Wait for \( k > n/2 \) nodes to reply with values \( x_1, \ldots, x_k \)
3. Return \( x_1 \sqcup \ldots \sqcup x_k \)
A hard problem: layout changes

- We rely on quorums $k > n/2$ within each partition:

$$n = 3, \quad k \geq 2$$

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>df-ymk</td>
<td>Abricot</td>
<td>Courgette</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas</td>
<td>Courgette</td>
<td>df-ykl</td>
</tr>
<tr>
<td>Partition 2</td>
<td>df-ymf</td>
<td>Celeri</td>
<td>Abricot</td>
</tr>
</tbody>
</table>

- During the rebalancing, new nodes don’t yet have the data, and old nodes want to get rid of the data to free up space, risk of inconsistency, how to coordinate?
A hard problem: layout changes

▶ We rely on quorums $k > n/2$ within each partition:

$$n = 3, \quad k \geq 2$$

▶ When rebalancing, the set of nodes responsible for a partition can change:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>df-ymk</td>
<td>Abricot</td>
<td>Courgette</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas</td>
<td>Courgette</td>
<td>df-ykl</td>
</tr>
<tr>
<td>Partition 2</td>
<td>df-ymf</td>
<td>Celeri</td>
<td>Abricot</td>
</tr>
</tbody>
</table>

→

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>Dahlia</td>
<td>Abricot</td>
<td>Eucalyptus</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas</td>
<td>Euphorbe</td>
<td>Doradille</td>
</tr>
<tr>
<td>Partition 2</td>
<td>Dahlia</td>
<td>Echinops</td>
<td>Abricot</td>
</tr>
</tbody>
</table>

During the rebalancing, new nodes don’t yet have the data, and old nodes want to get rid of the data to free up space → risk of inconsistency, how to coordinate?
# A hard problem: layout changes

▶ We rely on quorums $k > n/2$ within each partition:

$$n = 3, \quad k \geq 2$$

▶ When rebalancing, the set of nodes responsible for a partition can change:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>df-ymk</td>
<td>Abricot</td>
<td>Courgette</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas</td>
<td>Courgette</td>
<td>df-ykl</td>
</tr>
<tr>
<td>Partition 2</td>
<td>df-ymf</td>
<td>Celeri</td>
<td>Abricot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partition</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>Dahlia</td>
<td>Abricot</td>
<td>Eucalyptus</td>
</tr>
<tr>
<td>Partition 1</td>
<td>Ananas</td>
<td>Euphorbe</td>
<td>Doradille</td>
</tr>
<tr>
<td>Partition 2</td>
<td>Dahlia</td>
<td>Echinops</td>
<td>Abricot</td>
</tr>
</tbody>
</table>

▶ During the rebalancing, new nodes don’t yet have the data, and old nodes want to get rid of the data to free up space

→ risk of inconsistency, **how to coordinate?**
Handling layout changes without losing consistency

Solution:

- keep track of data transfer to new nodes
- use multiple write quorums (new nodes + old nodes while data transfer is in progress)
- switching reads to new nodes only once copy is finished

Implemented in v0.10

Validated with Jepsen testing

Garage v0.9.0

Garage v0.10 beta
Towards v1.0...

Focus on security & stability

▶ **Security audit** in progress by Radically Open Security

▶ Misc. S3 features (SSE-C, ...) and compatibility fixes

▶ Improve UX

▶ Fix bugs
Operating big Garage clusters
Operating Garage

```bash
$ garage status

<table>
<thead>
<tr>
<th>ID</th>
<th>Hostname</th>
<th>Address</th>
<th>Tags</th>
<th>Zone</th>
<th>Capacity</th>
<th>DataAvailable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec5753c546756825</td>
<td>df-pw5</td>
<td>[2a02:a03f:6510:5102:223:24ff:feb0:e8a7]:3991</td>
<td>[df-pw5]</td>
<td>bespin</td>
<td>500.0 GB</td>
<td>429.1 GB (89.0%)</td>
</tr>
<tr>
<td>76797283f6c7e162</td>
<td>carcajou</td>
<td>[2001:470:ca43::22]:3991</td>
<td>[carcajou]</td>
<td>neptune</td>
<td>200.0 GB</td>
<td>166.3 GB (73.5%)</td>
</tr>
<tr>
<td>8073f25ffbd6944</td>
<td>piranha</td>
<td>[2a01:cb05:91e:ec00:223:24ff:feb0:ea82]:3991</td>
<td>[piranha]</td>
<td>corrin</td>
<td>500.0 GB</td>
<td>457.3 GB (94.0%)</td>
</tr>
<tr>
<td>3aef398e82972b</td>
<td>origan</td>
<td>[2a01:e0a:5e4:1d0:223:24ff:feaf:dec]:3991</td>
<td>[origan]</td>
<td>jupiter</td>
<td>500.0 GB</td>
<td>457.1 GB (93.1%)</td>
</tr>
<tr>
<td>967786691f28bb79</td>
<td>caribou</td>
<td>[2001:470:ca43::23]:3991</td>
<td>[caribou]</td>
<td>neptune</td>
<td>500.0 GB</td>
<td>453.1 GB (92.3%)</td>
</tr>
</tbody>
</table>
```
### garage status

<table>
<thead>
<tr>
<th>ID</th>
<th>Hostname</th>
<th>Address</th>
<th>Tags</th>
<th>Zone</th>
<th>Capacity</th>
<th>DataAvail</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec5753c546756b25</td>
<td>df-pw5</td>
<td>[2a02:a03f:6510:5102:223:24ff:feb0:e8a7]:3991</td>
<td>[df-pw5]</td>
<td>bespin</td>
<td>500.0 GB</td>
<td>429.1 GB (89.0%)</td>
</tr>
<tr>
<td>76797283f6c7e162</td>
<td>carcajou</td>
<td>[2001:478:ca43::22]:3991</td>
<td>[carcajou]</td>
<td>neptune</td>
<td>200.0 GB</td>
<td>166.3 GB (73.5%)</td>
</tr>
<tr>
<td>8073f25fbb7d6944</td>
<td>piranha</td>
<td>[2a01:cb05:911e:ec00:223:24ff:feb0:ea82]:3991</td>
<td>[piranha]</td>
<td>corrin</td>
<td>500.0 GB</td>
<td>457.3 GB (94.0%)</td>
</tr>
<tr>
<td>3aedd398ee82972b</td>
<td>origan</td>
<td>[2a01:e0a:5e4:1d0:223:24ff:feaf:6dec]:3991</td>
<td>[origan]</td>
<td>jupiter</td>
<td>500.0 GB</td>
<td>457.1 GB (93.1%)</td>
</tr>
<tr>
<td>967786691f20bb79</td>
<td>caribou</td>
<td>[2001:478:ca43::23]:3991</td>
<td>[caribou]</td>
<td>neptune</td>
<td>500.0 GB</td>
<td>453.1 GB (92.3%)</td>
</tr>
</tbody>
</table>

### HEALTHY NODES

<table>
<thead>
<tr>
<th>ID</th>
<th>Hostname</th>
<th>Address</th>
<th>Tags</th>
<th>Zone</th>
<th>Capacity</th>
<th>DataAvail</th>
</tr>
</thead>
<tbody>
<tr>
<td>76797283f6c7e162</td>
<td>carcajou</td>
<td>[2001:478:ca43::22]:3991</td>
<td>[carcajou]</td>
<td>neptune</td>
<td>200.0 GB</td>
<td>166.3 GB (73.5%)</td>
</tr>
<tr>
<td>8073f25fbb7d6944</td>
<td>piranha</td>
<td>[2a01:cb05:911e:ec00:223:24ff:feb0:ea82]:3991</td>
<td>[piranha]</td>
<td>corrin</td>
<td>500.0 GB</td>
<td>457.3 GB (94.0%)</td>
</tr>
<tr>
<td>3aedd398ee82972b</td>
<td>origan</td>
<td>[2a01:e0a:5e4:1d0:223:24ff:feaf:6dec]:3991</td>
<td>[origan]</td>
<td>jupiter</td>
<td>500.0 GB</td>
<td>457.1 GB (93.1%)</td>
</tr>
<tr>
<td>967786691f20bb79</td>
<td>caribou</td>
<td>[2001:478:ca43::23]:3991</td>
<td>[caribou]</td>
<td>neptune</td>
<td>500.0 GB</td>
<td>453.1 GB (92.3%)</td>
</tr>
</tbody>
</table>

### FAILED NODES

<table>
<thead>
<tr>
<th>ID</th>
<th>Hostname</th>
<th>Address</th>
<th>Tags</th>
<th>Zone</th>
<th>Capacity</th>
<th>Last seen</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec5753c546756b25</td>
<td>df-pw5</td>
<td>[2a02:a03f:6510:5102:223:24ff:feb0:e8a7]:3991</td>
<td>[df-pw5]</td>
<td>bespin</td>
<td>500.0 GB</td>
<td>5 minutes ago</td>
</tr>
</tbody>
</table>
Garage’s architecture

Garage as a set of components

- S3 API
- Custom API
- KV Store
- Block Manager
- Anti Entropy
- CRDT
- Scheduler
- Layout
- Network

Alex Auvolat, Deuxfleurs Garage, the low-tech storage platform for geo-distributed clusters FOSDEM’24, 2024-02-03
Garage’s architecture

- Merkle tree sync
- Object metadata table
- Version metadata table
- Block_ref metadata table
- Locally stored data blocks
- Local block reference counters
- Block resync queue/scheduler
Digging deeper

```bash
$ garage stats

Garage version: 20240116133343 [features: k2v, sled, lmdb, sqlite, consul-discovery, kubernetes-discovery, metrics, telemetry-otlp, bundled-libs]
Rust compiler version: 1.68.0

Database engine: LMDB (using Heed crate)

<table>
<thead>
<tr>
<th>Table</th>
<th>Items</th>
<th>Nk1Items</th>
<th>Nk1Todo</th>
<th>GcTodo</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket_y2</td>
<td>19</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>key</td>
<td>12</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>object</td>
<td>67391</td>
<td>89964</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>version</td>
<td>33909</td>
<td>42045</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>block_ref</td>
<td>334735</td>
<td>378927</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Block manager stats:
- number of RC entries (~ number of blocks): 42376
- resync queue length: 0
- blocks with resync errors: 0

If values are missing above (marked as NC), consider adding the --detailed flag (this will be slow).

Storage nodes:

<table>
<thead>
<tr>
<th>ID</th>
<th>Hostname</th>
<th>Zone</th>
<th>Capacity</th>
<th>Part.</th>
<th>DataAvail</th>
<th>MetaAvail</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec5753c546756825</td>
<td>df-pw5</td>
<td>bespin</td>
<td>500.0 GB</td>
<td>175</td>
<td>429.1 GB/482.1 GB (89.0%)</td>
<td>429.1 GB/482.1 GB (89.0%)</td>
</tr>
<tr>
<td>96797283f6c7e162</td>
<td>caracal</td>
<td>neptune</td>
<td>200.0 GB</td>
<td>70</td>
<td>166.3 GB/226.2 GB (73.5%)</td>
<td>166.3 GB/226.2 GB (73.5%)</td>
</tr>
<tr>
<td>8073f25f7bd65944</td>
<td>piranha</td>
<td>corin</td>
<td>500.0 GB</td>
<td>173</td>
<td>457.3 GB/486.4 GB (94.8%)</td>
<td>457.3 GB/486.4 GB (94.8%)</td>
</tr>
<tr>
<td>369b0ec4f297b2b</td>
<td>origan</td>
<td>jupiter</td>
<td>500.0 GB</td>
<td>175</td>
<td>457.1 GB/490.7 GB (93.1%)</td>
<td>457.1 GB/490.7 GB (93.1%)</td>
</tr>
<tr>
<td>967786691f20bb79</td>
<td>caribou</td>
<td>neptune</td>
<td>500.0 GB</td>
<td>175</td>
<td>453.1 GB/490.8 GB (92.3%)</td>
<td>453.1 GB/490.8 GB (92.3%)</td>
</tr>
</tbody>
</table>

Estimated available storage space cluster-wide (might be lower in practice):
data: 668.3 GB
metadata: 668.3 GB
```
Digging deeper

<table>
<thead>
<tr>
<th>ID</th>
<th>State</th>
<th>Name</th>
<th>TID</th>
<th>Done</th>
<th>Queue</th>
<th>Errors</th>
<th>Consec</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Idle</td>
<td>Block resync worker #1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Idle</td>
<td>Block resync worker #2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Idle</td>
<td>Block resync worker #3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Idle</td>
<td>Block resync worker #4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Idle</td>
<td>Block resync worker #5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Idle</td>
<td>Block resync worker #6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Idle</td>
<td>Block resync worker #7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Idle</td>
<td>Block resync worker #8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Idle</td>
<td>Block sync worker</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Idle</td>
<td>bucket_v2 Merkle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Idle</td>
<td>bucket_v2 sync</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Idle</td>
<td>bucket_v2 GC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Idle</td>
<td>bucket_v2 queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Idle</td>
<td>bucket_alias Merkle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Idle</td>
<td>bucket_alias sync</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Idle</td>
<td>bucket_alias GC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Idle</td>
<td>bucket_alias queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Idle</td>
<td>key Merkle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Idle</td>
<td>key sync</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Idle</td>
<td>key GC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Idle</td>
<td>key queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Idle</td>
<td>object Merkle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Idle</td>
<td>object sync</td>
<td></td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td>17 hours ago</td>
</tr>
<tr>
<td>24</td>
<td>Idle</td>
<td>object GC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Idle</td>
<td>object queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Idle</td>
<td>bucket_object_counter Merkle</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Idle</td>
<td>bucket_object_counter sync</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Idle</td>
<td>bucket_object_counter GC</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Idle</td>
<td>bucket_object_counter queue</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Idle</td>
<td>multipart_upload Merkle</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Idle</td>
<td>multipart_upload sync</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Idle</td>
<td>multipart_upload GC</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Idle</td>
<td>multipart_upload queue</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Idle</td>
<td>bucket_mpu_counter Merkle</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Idle</td>
<td>bucket_mpu_counter sync</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Idle</td>
<td>bucket_mpu_counter GC</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Idle</td>
<td>bucket_mpu_counter queue</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Idle</td>
<td>version Merkle</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Idle</td>
<td>version sync</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td></td>
<td></td>
<td>17 hours ago</td>
</tr>
<tr>
<td>40</td>
<td>Idle</td>
<td>version GC</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Idle</td>
<td>version queue</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Idle</td>
<td>block_ref Merkle</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Idle</td>
<td>block_ref sync</td>
<td>0</td>
<td>45</td>
<td>0</td>
<td></td>
<td></td>
<td>17 hours ago</td>
</tr>
<tr>
<td>44</td>
<td>Idle</td>
<td>block_ref GC</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Idle</td>
<td>block_ref queue</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>Idle</td>
<td>object lifecycle worker</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Digging deeper

```bash
$ garage worker get
8073f25fffb7d6944 lifecycle-last-completed 2024-01-23
8073f25fffb7d6944 resync-tranquility 1
8073f25fffb7d6944 resync-worker-count 4
8073f25fffb7d6944 scrub-corruptions_detected 0
8073f25fffb7d6944 scrub-last-completed 2023-12-27T13:49:33.2342
8073f25fffb7d6944 scrub-next-run 2024-01-31T03:23:02.2342
8073f25fffb7d6944 scrub-tranquility 4

$ garage worker get -a resync-tranquility
3aed398eec82972b resync-tranquility 1
76797283f6c7e162 resync-tranquility 1
8073f25fffb7d6944 resync-tranquility 1
967786691f20bb79 resync-tranquility 1
ec5753c546756825 resync-tranquility 1
```
Potential limitations and bottlenecks

- **Global:**
  - Max. $\sim 100$ nodes per cluster (excluding gateways)

- **Metadata:**
  - One big bucket = bottleneck, object list on 3 nodes only

- **Block manager:**
  - Lots of small files on disk
  - Processing the resync queue can be slow
Deployment advice for very large clusters

- **Metadata storage:**
  - ZFS mirror (x2) on fast NVMe
  - Use LMDB storage engine

- **Data block storage:**
  - Use Garage’s native multi-HDD support
  - XFS on individual drives
  - Increase block size (1MB → 10MB, requires more RAM and good networking)
  - Tune resync-tranquility and resync-worker-count dynamically

- **Other:**
  - Split data over several buckets
  - Use less than 100 storage nodes
  - Use gateway nodes

Our deployments: < 10 TB. Some people have done more!
Where to find us

https://garagehq.deuxfleurs.fr/
mailto:garagehq@deuxfleurs.fr
#garage:deuxfleurs.fr on Matrix