

Maximum Efficiency

For Any HPC/AI Workload

Axel Rosenberg | Sr. Systems Engineer | DACH

NUG XXXIV20-06-23



Why



- **Compute workload isolation**

- > **flexible** Processor utilization
- > **better** Application performance
- > **easy** Client QoS

- **Storage Hardware Economics**

- > **maximum** Value of your HW
- > **affordable** Data placement
- > **linear** (Meta) Data scaling

This is Important

Key Takeaways



Energy

Storage server resource efficiency



Performance

Application server performance optimization



Intelligence

(Meta) Data management and scalability



Maximum Efficiency

For Any HPC/AI Workload

Axel Rosenberg | Sr. Systems Engineer | DACH
HUG XXXX23-06-23

1

Why ? This is Important

- Compute workload isolation
- flexible Processor utilization
- better Application performance
- easy Client QoS

- Storage Hardware Economics
- maximum Value of your HW
- affordable Data placement
- linear (Meta) Data scaling

2

Key Takeaways

- Energy**
Storage server resource efficiency
- Performance**
Application server performance optimization
- Intelligence**
(Meta) Data management and scalability

3

IO Intensive Applications

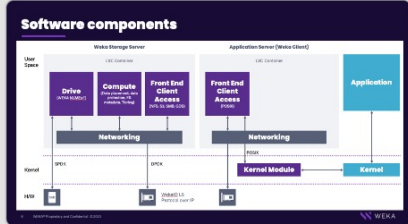
Client data access via POSIX, NFS, S3, Ceph and GPU Direct

4

Workload

WEKA User-Space Containers

5



6

Stateless client: Cores & NICs

Common mounting options

```
fronted process control
-> max_cores=1|2|4|... - controls the number of the FRONTEND drivers
```

Network device control

```
-> net=rook -> iSCSI
-> ceph_nic -> MUX CxG or newer, Intel E810, some Intel 10Gb NICs, vmxnet3
-> iqn -> any NIC supported by Linux, including virtualized ones
```

Core dedication control (relevant DPKM mounts)

```
-> dedicated_node=c1|11|1000 -> uses RX interrupts instead of "spinning on a core"
```

Mult NIC mounting with or without core pinning ("affiliation")

```
-> core=4,core=6,net=1-2,net=3,core=5,core=7,net=3-4,net=8
-> core=4,core=6,net=1-2,net=3,core=5,core=7,net=3-4,net=8
```

HA mounting

```
-> net_core=net-har-bp07000,net-har-bp22500,ipnet=10.0.4.102|10.0.10.102
```

7

Client Container resource alignment

DPDK mount

```
-> max_cores=2|4|8
-> core=1|8 -> core=63
-> core=1|8 -> core=63
-> core=1|8 -> core=63
-> core=1|8 -> core=63
```

Warning

It calls the VFS layer what system calls we can do, e.g. `read`, `write`, `fsync` and so on.

It fetches IO and suddenly discovers that it cannot complete it due to copying. In this case it returns `ENOSPC` and ends up on the IO and the kernel knows that it should retry such IO from the beginning. Second reason is `FS_CROST`. The low level reason of `FS_CROST` lies in the middle of the copying by the FS.

Warning

Client requests from the gateway driver and relay them to the FS.

The main driver manages file clusters used for metadata (higher priority) and cluster for reads and writes.

The main driver is flexible and adapts to the number of states. Free -> `fsync` -> `fsync` -> `fsync` -> `fsync`.

From End node manages similar queue on its end.

8

Client QoS

WEKA QoS is placed on the client container

Works for bandwidth and IOPS (read and write)

There is an option to set this globally for all clients, you can then override with a mount option with `-i` mount.

This can be set with a "guaranteed value" and a "burst value"

Preferred should be known cluster performance

Preferred should not be over-subscribed - This is the limit per container at or below the known cluster performance (lower cluster performance/client containers = IOPS and IOPS value)

Low priority allows for over-subscription - burst while other jobs are not using much IO

9

Resources

Leveraging HW resources at the most optimal mix including NUMA intelligence and power management.

10

Backend Core Efficiency

WEKA MCB (Multi Container Backend)
Lets you assign your processors for Hardware and Software resources dynamically.

11

Weka Configurator for MCB

```
Weka Configurator (Cores)
Host Configuration Reference
Cores per Host: 24
Drives per Host: 6
Number of Hosts: 5
Data Drives: 3
Parity Drives: 2
Hot Spares: 1
```

BEFORE: 100% CPU, 100% Memory, 100% IOPS

AFTER: 100% CPU, 100% Memory, 100% IOPS

12

WEKA Configurator for MCB

13

Data Management Efficiency

WEKA's zero-copy architecture enables intelligent data placement. Revolutionize research and results by managing data-oceans transparently to your (AI)-compute farm.

On-prem, hybrid and Multi-Cloud. WEKA runs anywhere.

14

Data Management Efficiency

```
weka fs tier s3 add myS3 --site local --
hostname=s3.localdomain.org --
ports=40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100 --
secret-key=SPN+XWkDUa5J94UqTdrivMppjPK6CSuZi3 --auth-method=AWSSignature4
```

15

Data Management Efficiency

16

Data Management Efficiency

```
tmpfs 7.6G 4.6K 7.6G 1% /opt/weka/data/agent/t
default 4.6T 482G 4.1T 11% /mnt/weka/
tmpfs 1.6G 0 1.6G 0% /run/user/1000
```

```
[root@node1 weka]# du -had 1 /mnt/weka/
128G /mnt/weka/csi-volumes
128G /mnt/weka/data
111G /mnt/weka/data3
128G /mnt/weka/data2
495G /mnt/weka/
```

17

Stateless client: Read & Writes

Common mounting options

Linux PageCache control

- `writesync` - writes and reads are cached. This is the default for POSIX mounts.
- `readcache` - only reads are cached (This is the default in NFS-V4 and S3 container mounts)
- `nocache` - neither reads or writes are cached.

Core dedication control (relevant DPKM mounts)

```
dedicated_node=c1|11|1000 -> uses RX interrupts instead of "spinning on a core"
```

18

Weka Zero Copy Architecture

What is SENSE

SENSE is a new data management paradigm. It is a new way of thinking about data management. It is a new way of thinking about data management. It is a new way of thinking about data management.

Infrastructure features

- SENSE is a new data management paradigm. It is a new way of thinking about data management. It is a new way of thinking about data management.

Flexible, easy to use & reliable

19

What is SENSE

20

WEKA vs Local NVMe

WEKA vs Local NVMe

WEKA vs OTHERFS

HPC/AI & Multicloud

Thank You!

Twitter, LinkedIn, Facebook icons

I/O intensive Applications



AI/ML



Fraud Detection



Financial Analysis



Microscopy

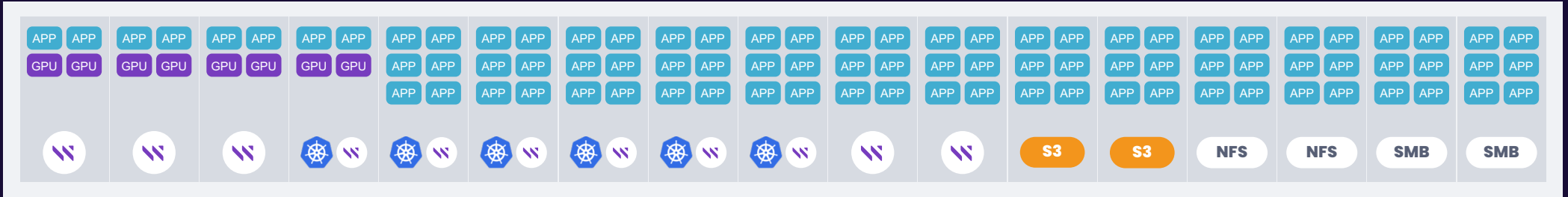


Geospatial Research

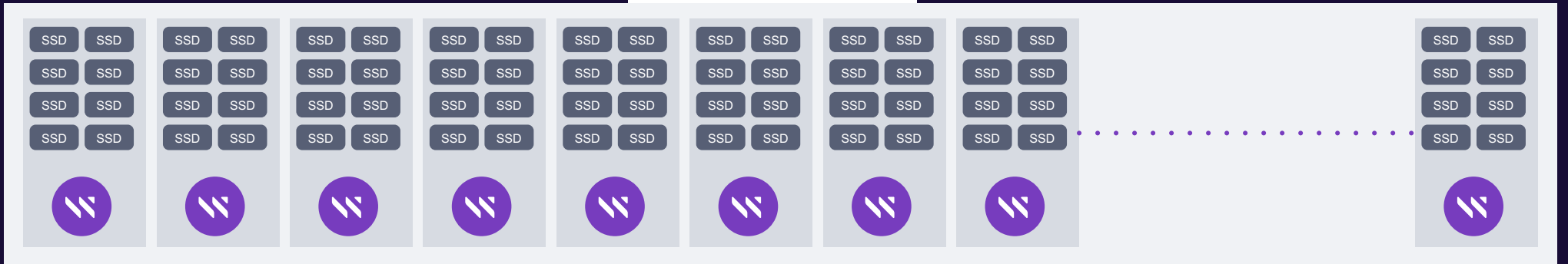


Genomics / Life Sciences

Clients data access via POSIX, NFS, SMB, S3, CSI and GPU Direct



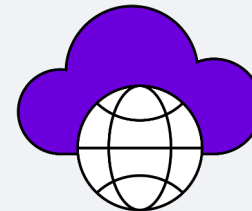
GLOBAL NAMESPACE



S3 DATA LAKE



PRIVATE



PUBLIC



Client



Backend

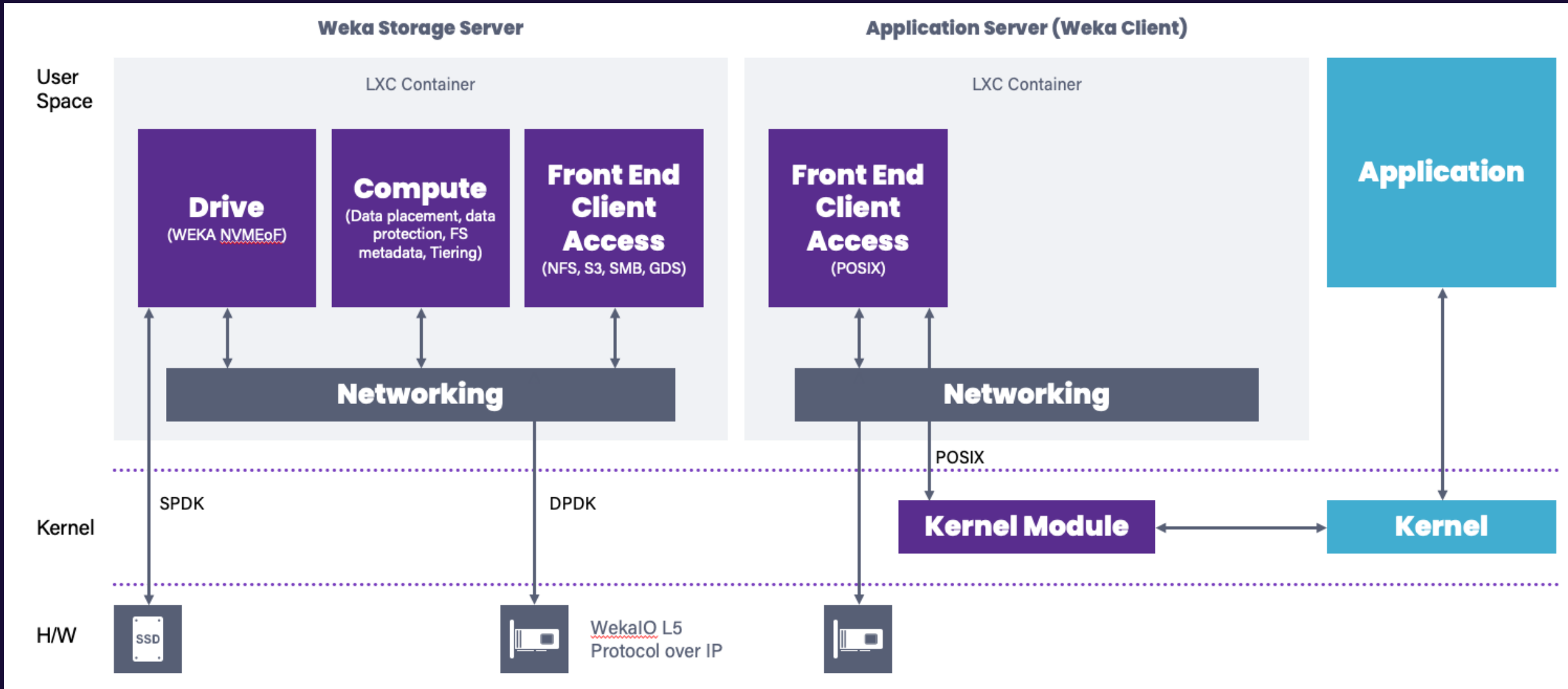
Automatic Tiering and Data Protection via S3 to On-Prem or Public Cloud Object Store

Workload

WEKA User-Space Containers



Software components



Stateless client: Cores & NICs

Common mounting options

Frontend processes control

- `-o num_cores=<1|2|4|...> --` controls the number of the FRONTEND drivers

Network device control

- `-o net=<dpdk_nic|udp>`
 - `<dpdk_nic>` – MLNX CX5 or newer; Intel E810, some Intel 10Gb NICs, vmxnet3
 - `udp` – any NIC supported by Linux, including virtualized ones.

Core dedication control (relevant DPDK mounts)

- `-o dedicated_mode=<full|none> --` uses RX interrupts instead of “spinning on a core”

Multi-NIC mounting with or without core pinning (“affinization”)

- `-o num_cores=4,net=<nic1>,net=<nic2>`
- `-o core=48,core=49,net:s1-2=ibp12s0,core=50,core=51,net:s3-4=ibp18s0`

HA mounting

- `-o num_cores=6,net:ha=ibp97s0f0,net:ha=ibp225s0f0,mgmt_ip=10.0.4.102+10.0.10.102`

Client Container resource alignment

DPDK mount

```
-o net=enp225s0f1 / ib0
-o num_cores=2/4/8
-o core=16 -o core=63
-o net:s1-2=enp186s0 -o net:s3-
  4=enp12s0
-o memory_mb=32000
```

wekafsgw

- It tells the VFS layer what system calls we can do, e.g. read, write, lookup and so on.
- FE fetches IO and suddenly discovers that it cannot complete it due to choking. In this case FE returns special error code on the IO and GW knows that it should replay such IO from the beginning. Second reason is FE crash, the GW will replay all IOs that were in the middle of processing by the FE

wekafsio

- Gets requests from the gateway driver and relays them to the FE
- The main driver manages two queues, one for metadata (higher priority) and another for reads and writes
- The queue size is flexible and each request can be in a number of states: Free -> Allocated -> FE Pending -> Sent -> Replied
- Front End node manages similar queue on its end

Client QoS

WEKA QOS is placed on the client container

Works for bandwidth and IOPS (read and write).

There is an option to set this globally for all clients, you can then override with a mount option with `-o remount`.

This can be set with a "guaranteed value" and a "burst value"

Preferred should be known cluster performance

Preferred should not be oversubscribed – This is the limit per container at or below the known cluster performance (known cluster performance/client containers = GB/s and IOPS value)

Low priority allows for over subscription – burst while other jobs are not using much IO

Resources

Leveraging HW resources at the most optimal mix including NUMA intelligence and power management.



Backend Core Efficiency

WEKA MCB (Multi Container Backend)
Lets you assign your processors for
Hardware and Software resources
dynamically.



Weka Configurator for MCB

Weka Configurator (Cores)

Host Configuration Reference

Cores per host: 24
Drives per host: 6
Number of hosts: 5

Data Drives: 3
Parity Drives: 2

Hot Spares: 1

Bias:

- Enable Protocols
- Protocols are Primary
- DRIVES over COMPUTE

Cores for OS: 2

Cores for Protocols: 0

Usable Weka Cores: 22

Used Weka Cores: 22

FE Cores: 1


DRIVES Cores: 6

COMPUTE Cores: 15

Cluster Name:

- WEKA v4.1 documentation
- WEKA SYSTEM OVERVIEW**
- About the WEKA system
- SSD capacity management
- Filesystems, object stores, and filesystem groups
- WEKA networking
- Data lifecycle management
- WEKA client and mount modes
- WEKA containers architecture overview
- Glossary
- GETTING STARTED WITH WEKA**
- Quick installation guide
- Manage the system using the WEKA CLI
- Manage the system using the WEKA GUI
- Run first IOs with WEKA filesystem
- Getting started with WEKA REST API

<code>device-paths</code>	Space-separated list of strings	List of block devices that identify local SSDs, e.g., <code>/dev/nvme0n1</code> <code>/dev/nvme1n1</code>	Must be a valid Unix network device name
---------------------------	---------------------------------	---	--

 **Note:** If, due to some technical limitation, the use of an NVMe device through the kernel is required, contact the [Customer Success Team](#).

8. Configure the CPU resources

Command: `weka cluster container cores`

This stage in the installation process is used to configure the number of CPU resources, which are physical rather than logical cores. To perform this operation, use the following command line:

```
weka cluster container cores <container-id> <cores> [--frontend-dedicated-cores <frontend-dedicated-cores>] [--drives-dedicated-cores <drives-dedicated-cores>] [--cores-ids <cores-ids>] [--compute-dedicated-cores <compute-dedicated-cores>] [--only-drives-cores] [--only-compute-cores] [--only-frontend-cores]
```

Parameters

Name	Type	Value	Limitations
<code>container-id</code>	String	Identifier of the container in which a core count is	Must be a valid container identifier

 Copy link

ON THIS PAGE

Workflow

1. Install the WEKA soft...
- Stage 2: Formation of a cl.
2. Create a cluster from ...
3. Set a name for the clu...
4. Enable event notificat...
- Stage 5: Set containers d...
5. Set containers dedica...
6. Configure the networ...
7. Configure the SSDs
- 8. Configure the CPU re...**
9. Configure the memor...
10. Configure failure do...
11. Configure WEKA sys...
12. Configure hot spare ...
- Stage 13: Apply container...
13. Apply containers co...
14. Verify the containers...
15. Set a license
16. Run the Start IO co...

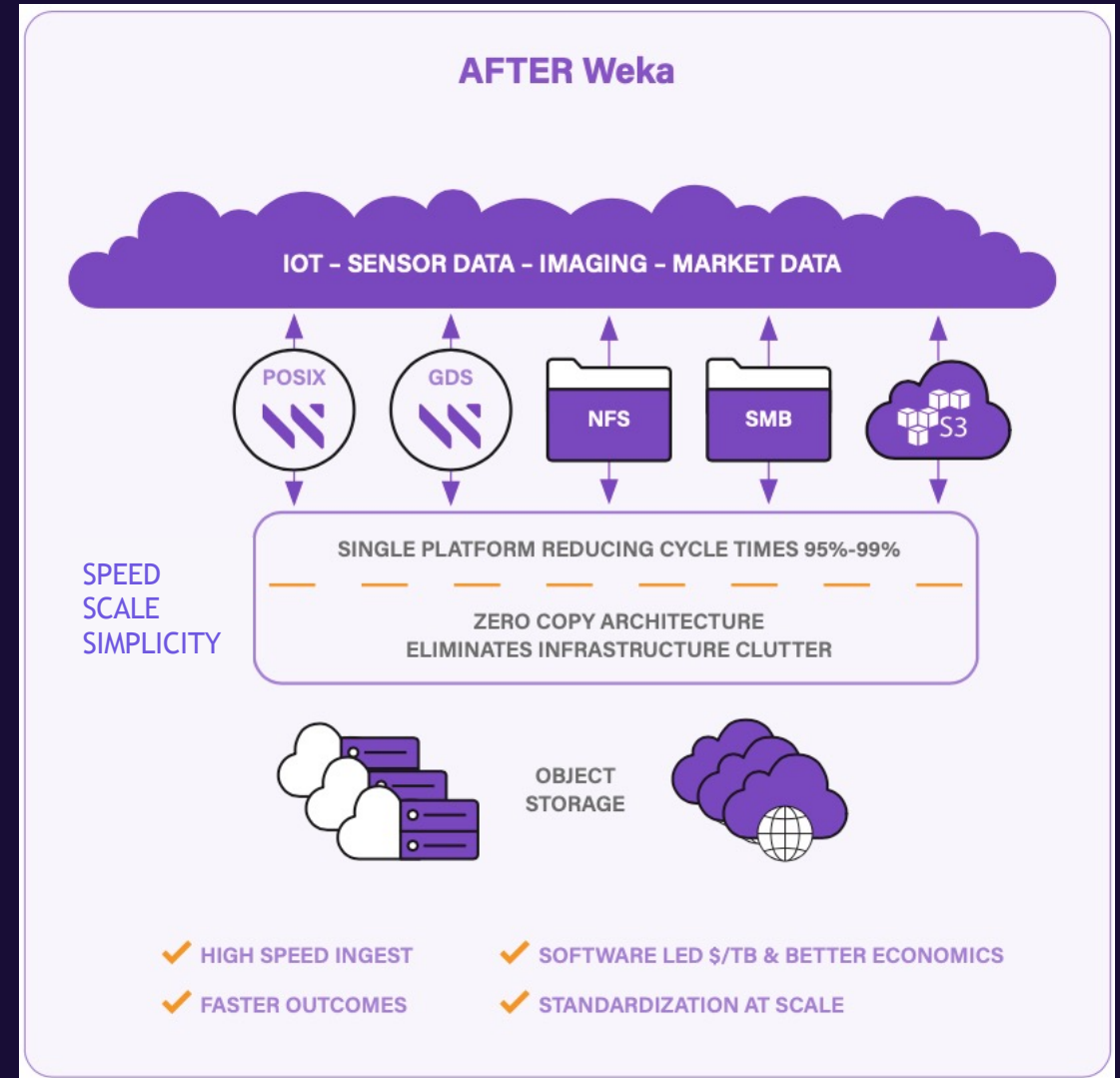
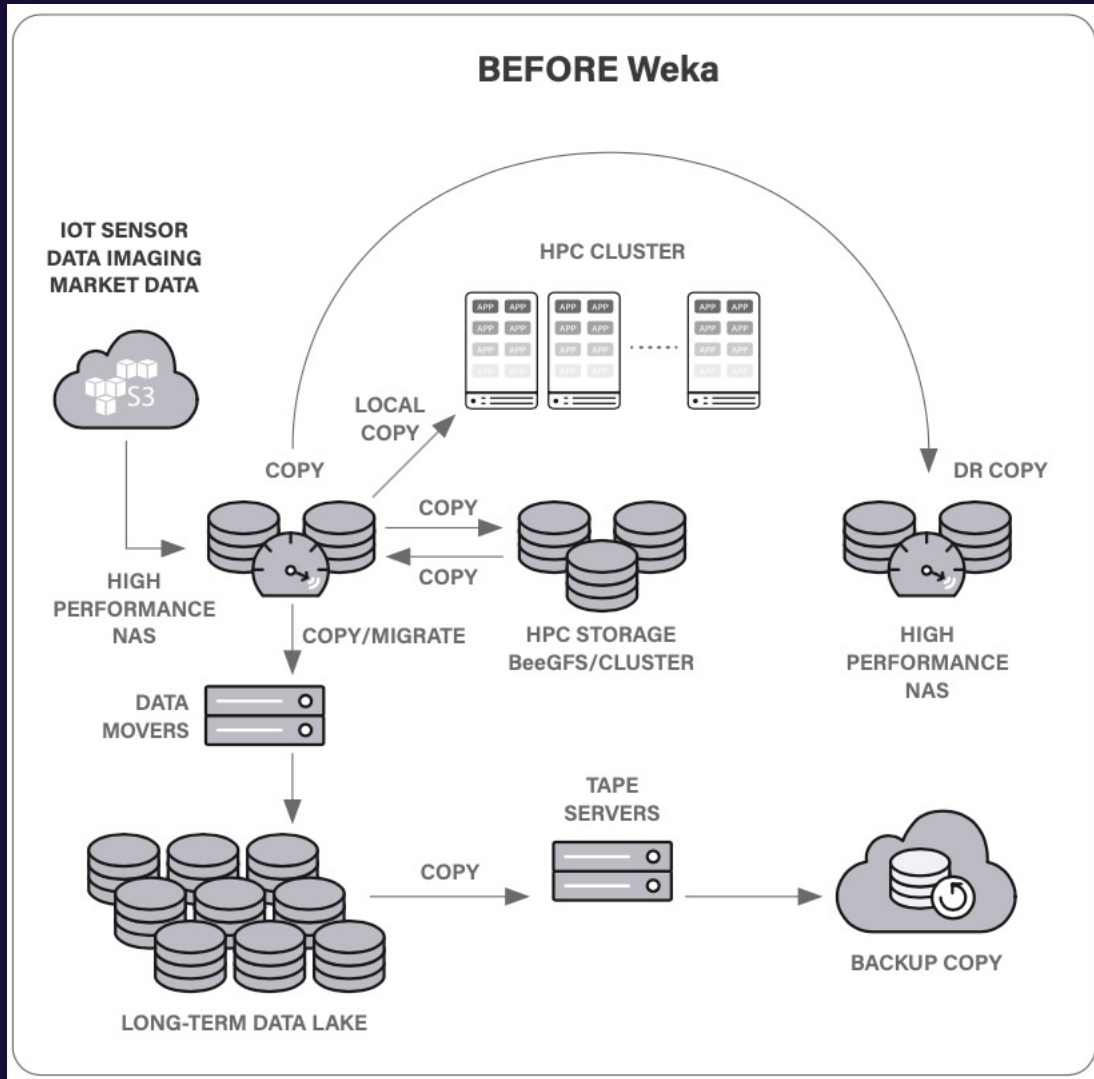
Data Management Efficiency

WEKA's zero-copy architecture enables intelligent data placement. Revolutionize research and results by managing data-oceans transparently to your (AI)-compute farm.

On-prem, hybrid and Multi-Cloud. WEKA runs anywhere.



Weka Zero Copy Architecture

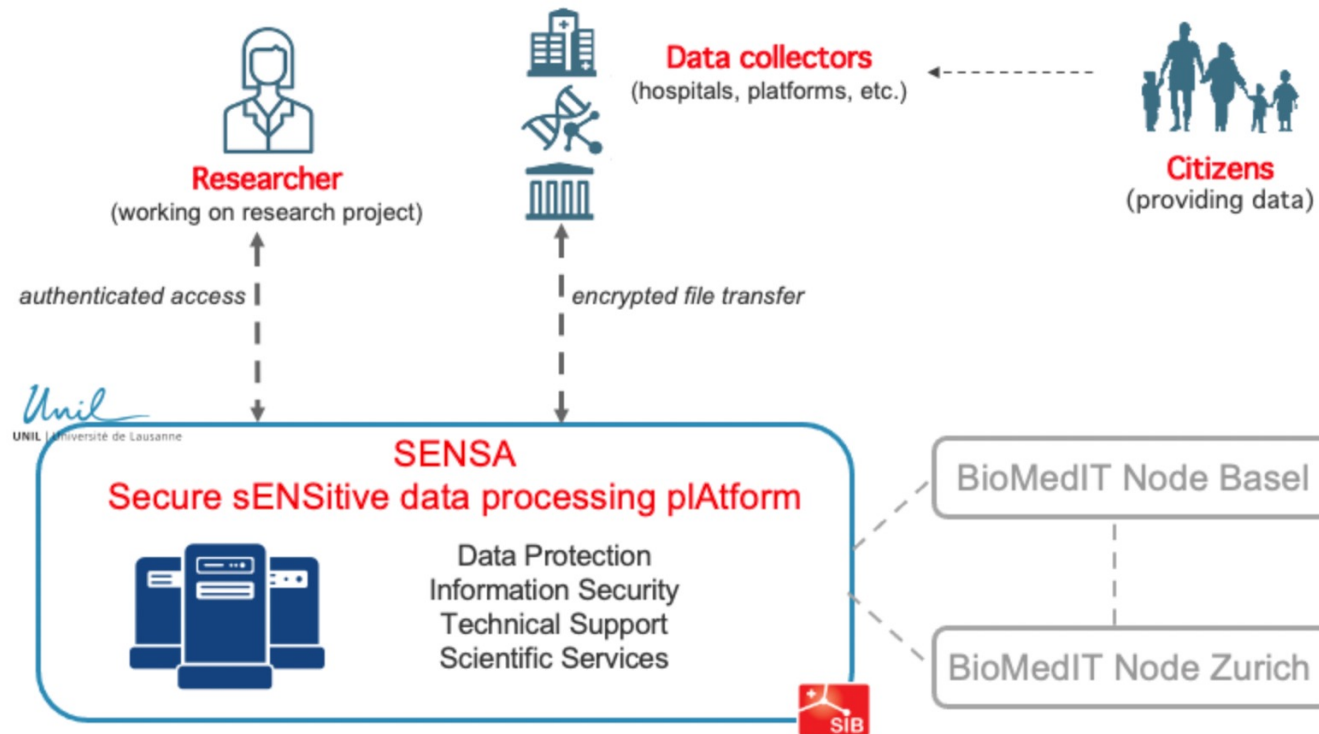


Weka Zero Copy Architecture in Practice

What is SENSEA

<https://sensa.biomedit.ch/index.html#infrastructure>

The **Secure sENSitive data processing plATform (SENSEA)** offers biomedical researchers a full service for the processing of sensitive data, from a tailored compute and storage environment to expertise in data protection and bioinformatics support. SENSEA being connected to the national network of secure IT infrastructures (**BioMedIT**), it also acts as a gateway to the Swiss Personalized Health Network (**SPHN.ch**) to enable nationwide biomedical projects. SENSEA is provided by the **University of Lausanne** and the **SIB Swiss Institute of Bioinformatics**, and builds on over 15 years of experience in operating high performance computing infrastructure for biomedical researchers.



Flexible, easy to use & reliable

Weka Zero Copy Architecture

What is SENSEA

The **Secure sENSitive data processing pLATFORM (SENSEA)** offers biomedical researchers a full service for the processing of sensitive data, from a tailored compute and storage environment to expertise in data protection and bioinformatics support. SENSEA being connected to the national network of secure

IT
SI
hi

Infrastructure features

SENSEA is created following the principle of "data protection by design", and implemented through a dedicated technical architecture that enforces the latest standards in security, data protection and service virtualization. The platform offers the following features:

- Encrypted transfer and storage
 - **Encrypted data transfer** into and out of the platform via a Secure File Transfer Service
 - Superior data protection via an **encrypted storage system** (WekaIO)
 - Unless agreed otherwise, the infrastructure providers have no access to the data
- Access control
 - **Access** to the platform restricted to **trusted network locations** such as white-listed IP addresses or ranges (incl. VPN)
 - Federated identity management via SWITCH edu-ID, including 2-factor authentication
 - Isolation of distinct project spaces via **virtualization** based on OpenStack technology, compliant with the legal requirements for sensitive personal data
 - Users interact with the platform through remote desktop in web browser (via Apache Guacamole web application) or through Secure Shell (SSH)-based terminal
- Hardware resources
 - 10 CPU nodes with 40 cores each (400 CPU cores in total) - 64 to 512 GB RAM
 - 2 GPU nodes with 20 cores each (graphics cards for optimised computation) - 64 GB RAM
 - 200 TB of encrypted storage

The hardware listed above is dynamically allocated to projects via the virtualization system OpenStack. If a scientific project needs more physical resources than currently available, the compute and storage capacity can be extended on demand.

Testimonials and documents

<https://sensa.biomedit.ch/index.html#infrastructure>





Flexible, easy to use & reliable

Data Management Efficiency

```
weka fs tier s3 add myS3 --site local --  
hostname=s3.localdomain.org --port=80 --  
bucket=weka-HPC-AI --access-key-  
id=AKIA3EOCJV63VELPTTWY --secret-  
key=8PN+XWkDUa53JI94UtqTdwrivM/opjtPK  
6CSuZi3 --auth-method=AWSSignature4
```

```
weka fs tier s3 attach <fs-name>  
<obs-name> [--mode mode]
```


Data Management Efficiency

Status	Tiering	Remote Backup	Encrypted	SSD Capacity	Total Capacity
				176.96 GB out of 400.00 GB (44.2%) 	494.77 GB out of 5.00 TB (9.9%) 

```
456G /mnt/weka/
[root@node1 weka]# weka fs
FILESYSTEM ID  FILESYSTEM NAME  USED SSD  AVAILABLE SSD  USED TOTAL  AVAILABLE TOTAL
0              default          169.55 GB  400 GB         503.94 GB  5 TB
[root@node1 weka]#
```

Data Management Efficiency

```
tmpfs      7.6G  4.0K  7.6G    1% /opt/weka/data/agent/tr
default    4.6T  482G  4.1T   11% /mnt/weka
tmpfs      1.6G   0    1.6G    0% /run/user/1000
```


WEKA vs Local NVMe

```
issued rwts: total=0,1249493,0,0 short=0,0,0,0 dropped=0,  
latency : target=0, window=0, percentile=100.00%, depth
```

Run status group 0 (all jobs):

WRITE: bw=26.7GiB/s (28.7GB/s), 26.7GiB/s-26.7GiB/s (28.7GB/s)

Disk stats (read/write):

```
md127: ios=693/10692498, merge=0/0, ticks=0/0, in_queue=0,  
nvme0n1: ios=14/1333226, merge=0/3686, ticks=6/5930341, in_queue=0,  
nvme6n1: ios=14/1333162, merge=0/3708, ticks=5/3955353, in_queue=0,  
nvme9n1: ios=709/1334830, merge=0/26157, ticks=734/11635935, in_queue=0,  
nvme5n1: ios=14/1333124, merge=0/3472, ticks=7/3584518, in_queue=0,  
nvme8n1: ios=14/1333106, merge=0/3506, ticks=7/19338552, in_queue=0,  
nvme1n1: ios=14/1333110, merge=0/3627, ticks=5/12946548, in_queue=0,  
nvme4n1: ios=14/1333130, merge=0/3466, ticks=6/3689937, in_queue=0,  
nvme7n1: ios=14/1333148, merge=0/3550, ticks=8/4742643, in_queue=0
```

root@a100:~# fio fio.job.axel.local --section=W1024ds

W1024ds: (g=0): rw=rw, bs=(R) 1024KiB-1024KiB, (W) 1024KiB-1024KiB

...

fio-3.35

Starting 230 processes

Jobs: 230 (f=230): [W(230)][49.0%][w=25.2GiB/s][w=25.8k IOPS]

```
root@a100:~#
```

```
top - 13:49:21 up 19 days, 55 min, 0 users, load average: 197.68, 120.84, 92.37  
Tasks: 2750 total, 2 running, 2745 sleeping, 0 stopped, 3 zombie  
%Cpu(s): 0.1 us, 0.7 sy, 0.0 ni, 30.9 id, 68.3 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 1031871.0 total, 858001.4 free, 38533.4 used, 135277.0 buff/cache  
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 988025.2 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
11900	root	20	0	9491440	225468	24048	S	6.2	0.0	1135:44	nvsm_core
4141930	root	20	0	318312	104496	50436	S	7.9	0.0	0:12.54	fio
3685945	root	20	0	7730272	99808	23940	S	0.3	0.0	51:55.84	weka
16717	root	20	0	4499288	83276	20048	S	0.0	0.0	70:53.46	nvsm_api_gat
3706264	root	20	0	6858400	78904	20852	S	0.0	0.0	3:19.63	snappd
11892	root	20	0	9879.1m	71012	50632	S	0.0	0.0	38:54.31	dockerd
4141968	root	20	0	253104	51352	2236	D	1.0	0.0	0:01.26	fio
4142107	root	20	0	253656	51332	2216	D	1.0	0.0	0:01.22	fio
4141975	root	20	0	253132	51328	2212	D	1.0	0.0	0:01.28	fio
4141979	root	20	0	253148	51328	2212	D	1.0	0.0	0:01.29	fio
4142105	root	20	0	253648	51300	2184	D	0.7	0.0	0:01.19	fio
4142102	root	20	0	253636	51284	2152	D	0.7	0.0	0:01.24	fio

WEKA vs Local NVMe

```
upid=0, jobs=230): err= 0: pid=4155038: Tue Jun 1
=35.7k, BW=34.8GiB/s (37.4GB/s)(1253GiB/35960msec
c): min=973, max=92411, avg=6439.68, stdev=4444.3
s): min=20535, max=50103, per=100.00%, avg=35747.
: min=20475, max=50038, avg=35687.13, stdev=28.
: usr=0.64%, sys=0.16%, ctx=1286213, majf=0, mi
: 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32
: 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 6
: 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 6
wts: total=0,1282803,0,0 short=0,0,0,0 dropped=0,
: target=0, window=0, percentile=100.00%, depth
```

```
oup 0 (all jobs):
4.8GiB/s (37.4GB/s), 34.8GiB/s-34.8GiB/s (37.4GB/
fio fio.job.axel --section=W1024ds
): rw=rw, bs=(R) 1024KiB-1024KiB, (W) 1024KiB-102
```

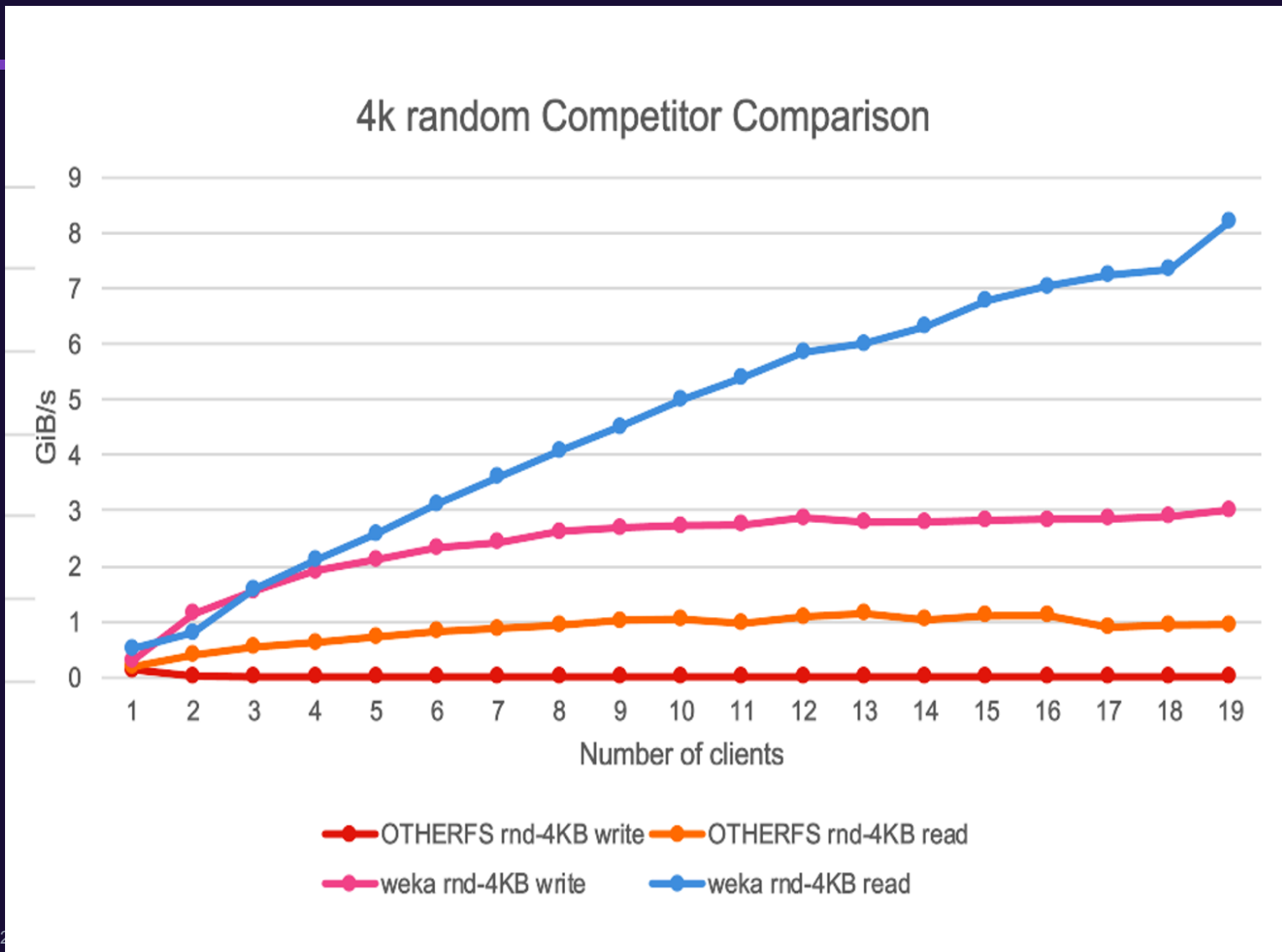
```
processes
19): [/(1),W(1),/(12),W(1),/(5),W(1),/(56),W(2),/
33): [/(1),W(2),/(5),W(1),/(5),W(2),/(4),W(1),/(2
230): [W(230)][1.6%][w=36.3GiB/s][w=37.2k IOPS][e
230): [W(230)][7.6%][w=36.7GiB/s][w=37.6k IOPS][e
```

```
Calling the mount command
Mount completed successfully
root@a100:~#
```

```
top - 14:00:13 up 19 days, 1:00, 9 users, load average: 199.69, 177.45, 140.90
Tasks: 2839 total, 16 running, 2821 sleeping, 0 stopped, 2 zombie
%Cpu(s): 5.2 us, 5.5 sy, 0.0 ni, 89.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1031871.+total, 829610.9 free, 61585.4 used, 140675.6 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 961677.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4147767	root	20	0	9778876	1.8g	523944	R	100.7	0.2	7:47.77	wekanode
4147771	root	20	0	9778876	1.8g	523504	R	100.7	0.2	7:47.60	wekanode
4147773	root	20	0	9778876	1.8g	527404	R	100.7	0.2	7:47.83	wekanode
4147787	root	20	0	9778876	1.8g	523648	R	100.7	0.2	7:47.60	wekanode
4147750	root	20	0	9778876	1.8g	526312	R	100.3	0.2	7:47.86	wekanode
4147751	root	20	0	9778876	1.8g	524080	R	100.3	0.2	7:47.75	wekanode
4147752	root	20	0	9778876	1.8g	523504	R	100.3	0.2	7:47.90	wekanode
4147777	root	20	0	9778876	1.8g	523896	R	100.3	0.2	7:47.76	wekanode
4147779	root	20	0	9778884	1.8g	523960	R	100.3	0.2	7:47.86	wekanode
4147780	root	20	0	9778876	1.8g	523960	R	100.3	0.2	7:47.69	wekanode
4147790	root	20	0	9778876	1.8g	523568	R	100.3	0.2	7:47.74	wekanode
4156193	root	20	0	253344	51424	2344	S	14.4	0.0	0:02.45	fio
4156083	root	20	0	252908	51360	2280	S	13.8	0.0	0:02.44	fio

WEKA vs OTHERFS



HPC/AI & Multicloud aka Cloud-Bursting



Products > Weka® Data Platform



Weka® Data Platform [Save to my list](#)

WEKA

[Overview](#) [Plans + Pricing](#) [Ratings + reviews](#)

Get It Now

Pricing information

Price varies

Categories

[AI + Machine Learning](#)
[Storage](#)

Support

[Support](#)
[Help](#)

Legal

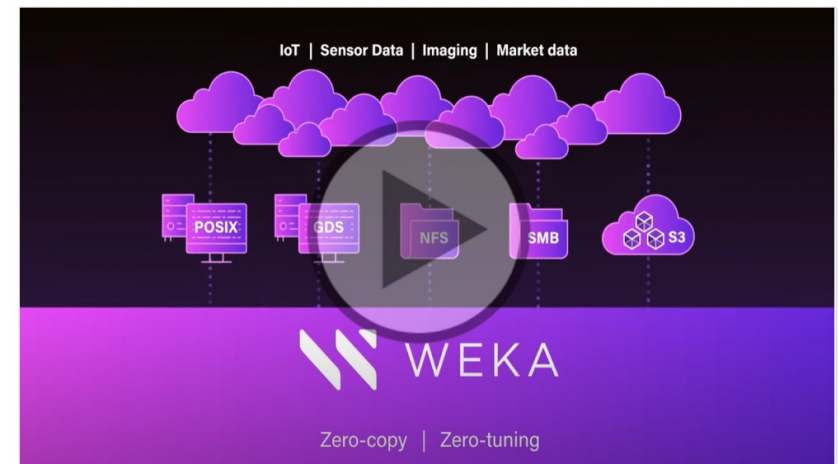
[License Agreement](#)
[Privacy Policy](#)

WEKA Data Platform is the fastest storage platform for compute accelerated applications.

Experience up to 10x data pipeline velocity improvements over traditional and cloud-based storage solutions, saving time and money.

Application environments that need low-latency access to millions of small files and high bandwidth for large files have limited storage options. The WEKA Data Platform combines dense NVMe storage of Azure Virtual Machine Lsv3-series instances with Azure Blob Storage in a single, efficient namespace, for your high-performance workloads, scaling to billions of files and hundreds of petabytes. It has a rich feature set that includes transparent object tiering, instantaneous snapshots, snap-to-object (remote clouds), backup, disaster recovery ("DR"), encryption, quotas, Active Directory integration, Kubernetes CSI driver, and much more. The WEKA Data Platform supports multiple file service protocols including full POSIX, NFS, SMB and S3, with full data share-ability across protocols. The WEKA Data Platform delivers the speed, simplicity and scalability for demanding workloads including AI, machine learning, visual effects rendering, genomics, high frequency trading, data analytics, and software builds.

WEKA Data Platform Cloud Edition



HPC/AI & Multicloud aka Cloud-Bursting

Start with Your Requirements

CAPACITY ?

Total Capacity ▼

Tiering SSD Only SSD+S3

PERFORMANCE ?

IOPS BW

R/W %

IOPS K

Region ^ ?

View all Your Options

Instance Type	Cluster Size	Usable Capacity (TB)	IOPS (K) ?	BW (Gbytes/sec) ?	
i3en.2xlarge	150	596.0	Up to 2,001	Up to 69.6	Deploy to AWS
i3en.6xlarge	66	780.0	Up to 2,020	Up to 91.4	Deploy to AWS
i3en.3xlarge	151	900.0	Up to 2,009	Up to 109.9	Deploy to AWS
i3en.12xlarge	45	1,056	Up to 2,030	Up to 129.2	Deploy to AWS
i3en.24xlarge	38	1,776	Up to 2,052	Up to 219.1	Deploy to AWS

Show Less

Deploy your WekaIO cluster

Version: 4.1.0.77
[change version](#)

Region: EU (Frankfurt) [change region](#)

Change backend instances (total cost will vary)

Backend Type: **Number Of Backends:**

Add clients to cluster

Clients Type: **Number Of Clients:**

AMI:

Deploy to AWS

this will take you to AWS CloudFormation

Template

Template URL

<https://getwekaio--release-files-prod.s3.amazonaws.com/ja0n5xqj6z7jkirej99p4mfz.json>

Stack description

[WekaIO] To learn more about this template visit <https://docs.weka.io/install/aws/cloudformation>

Stack name

Stack name

Stack name can include letters (0-9, A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Network Configuration

VPC

VPC ID in which the cluster would be installed

Thank You!

 @wekaio

 /wekaio

 @wekaio