

Checkmk #11
Conference



Performance tuning and more — Checkmk under the hood

May 20, 2025 - Live from Paulaner am Nockherberg, Munich

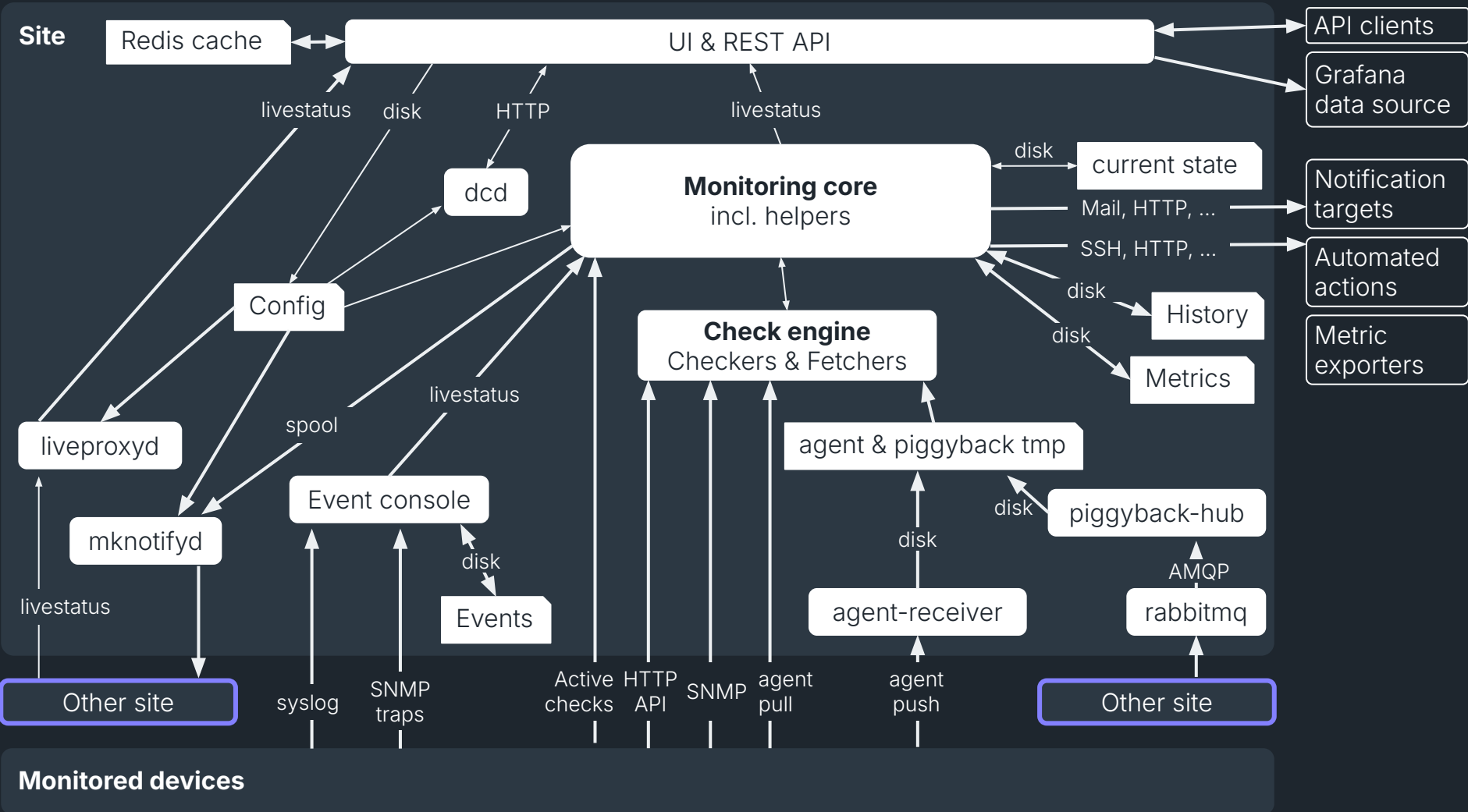


Lars Michelsen

Chief Technology Officer
Checkmk GmbH

**Checkmk
architecture**

**Tuning
Checkmk**



**Checkmk
architecture**

**Tuning
Checkmk**

Tuning for best performance



**UI &
REST API**

Site

Redis cache



UI & REST API

livestatus

disk

livestatus

Monitoring core
incl. helpers

disk

current state

Config

disk

History

disk

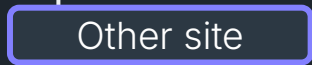
Metrics

liveproxyd

livestatus

Other site

Monitored devices



Checkmk Apache 101: prefork

apache parent

apache child process 1



apache child process 2



apache child process 3



apache child process 4



apache child process 5



apache child process 6



apache child process 7



●●● up to 64 (default)

Edit global setting

Setup > General > Global settings > Edit global setting

No pending changes

Setting Display Help ^

✓ Save

↔ Remove explicit setting

⬆ Global settings

Apache process tuning

Current setting ✖ Number of apache processes

64

Factory setting Number of apache processes: 64

Current state This variable is at factory settings.

**Default too high in
most cases!**

Process monitoring for Apache (built-in since 2.4.0)



OK

Process stable
apache



Processes: 5, Virtual memory: 969 MiB, Resident memory: 748 MiB, CPU: 1.52%, Youngest running for: 43 minutes 16 seconds, Oldest running for: 7 hours 23 minutes

5

Number of processes

2025-04-30 @ 5m



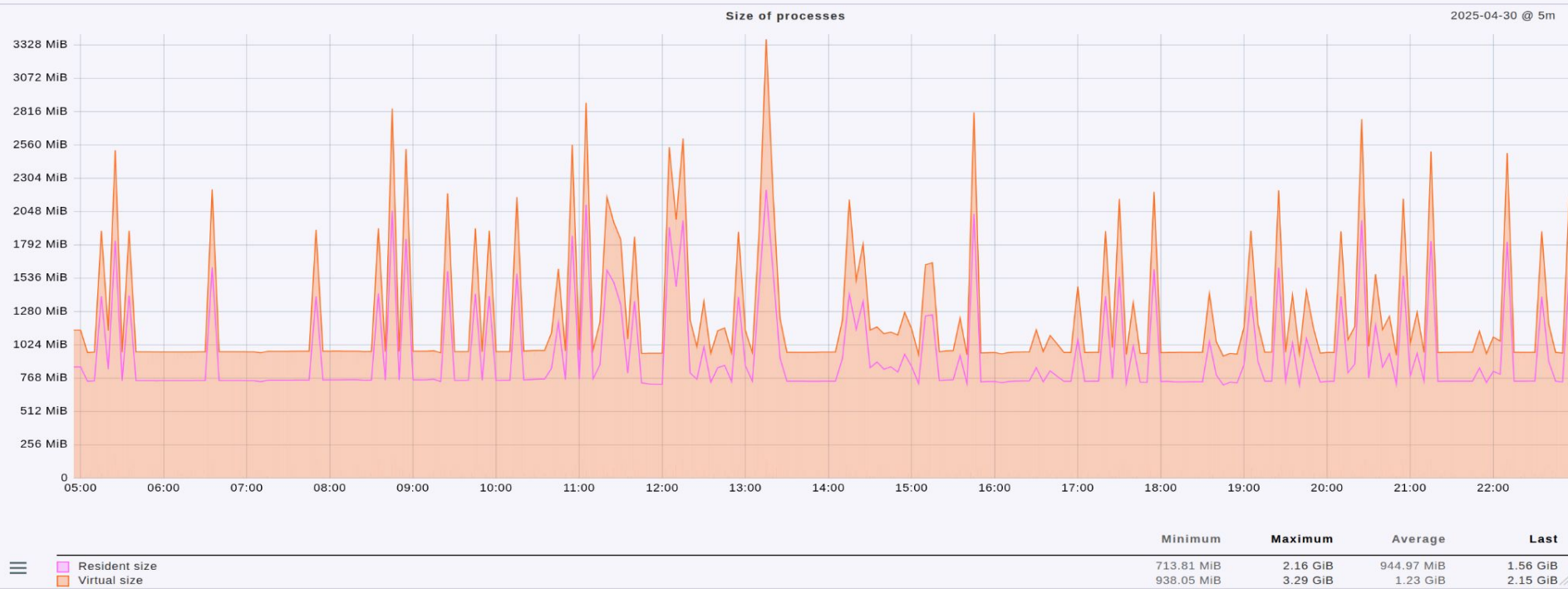
Process monitoring added for all key components

During update from 2.3 to 2.4

```
-| 14/36 Process discovery for self monitoring...
-| Adding: Shipped rule to monitor sites rrdcached
-| Adding: Shipped rule to monitor sites rrd helper
-| Adding: Shipped rule to monitor sites redis-server
-| Adding: Shipped rule to monitor sites real-time helper
-| Adding: Shipped rule to monitor sites rabbitmq
-| Adding: Shipped rule to monitor sites piggyback hub
-| Adding: Shipped rule to monitor sites notify helper
-| Adding: Shipped rule to monitor sites notification spooler
-| Adding: Shipped rule to monitor sites livestatus proxy
-| Adding: Shipped rule to monitor sites jaeger
-| Adding: Shipped rule to monitor sites fetcher helpers
-| Adding: Shipped rule to monitor sites event console
-| Adding: Shipped rule to monitor sites dcd
-| Adding: Shipped rule to monitor sites cmc
-| Adding: Shipped rule to monitor sites checker helpers
-| Adding: Shipped rule to monitor sites automation helpers
-| Adding: Shipped rule to monitor sites apache
-| Adding: Shipped rule to monitor sites alert helper
-| Adding: Shipped rule to monitor sites agent receiver
-| Adding: Shipped rule to monitor sites active check helpers
```

Apache is memory hungry

Avoid OOM. Also check your normal Memory service!



Verify that: Total apache memory demand < estimated RAM for new processes

Can't scale further? Look at OMD Apache monitoring



OK

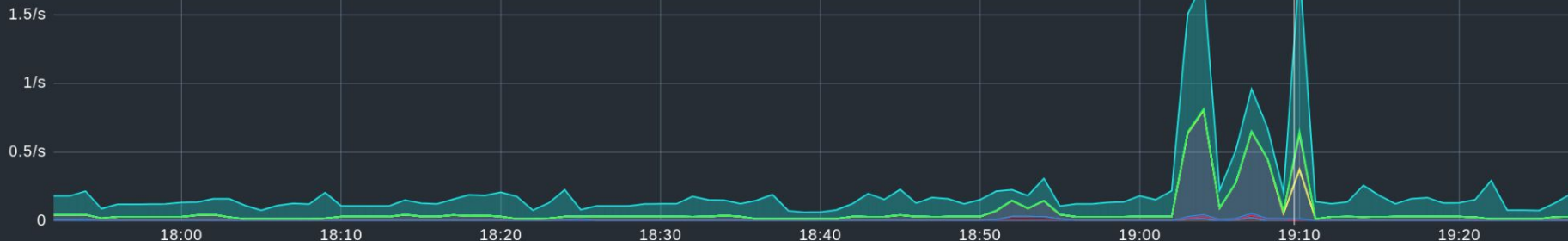
OMD stable apache



Requests: 0.02/s, Seconds serving: 0.01/s, Sent: 8.56 kB/s

Handled Requests

2025-05-04 @ 1m



Minimum Maximum Average Last 2025-05-04 19:09

Other: Requests	0.05/s	1.19/s	0.15/s	0.17/s	1.19/s
Scripts: Requests	0/s	0.02/s	0.0004/s	0/s	0.02/s
Styles: Requests	0/s	0.02/s	0.0002/s	0/s	0.02/s
Image: Requests	0/s	0.25/s	0.003/s	0/s	0.25/s
NagVis: Other: Requests	0/s	0/s	0/s	0/s	0/s
NagVis: AJAX: Requests	0/s	0/s	0/s	0/s	0/s
NagVis: Sidebar element: Requests	0/s	0/s	0/s	0/s	0/s
Checkmk: Other: Requests	0.02/s	0.76/s	0.06/s	0.03/s	0.36/s
Checkmk: Dashboards: Requests	0/s	0.02/s	0.0005/s	0/s	0.02/s
Checkmk: Sidebar elements: Requests	0/s	0.03/s	0.002/s	0/s	0/s

Slow view log

May help to identify slow (custom) views

- When enabled, one log entry per view rendering that exceeds the configured threshold is logged to `var/log/web.log` and contains the following information
- See werk [#11517](#) for further information
- Possible ways to improve view load times:
 - Tune regexes for efficiency
 - Reduce data to be fetched
(limit number of rows, increase update interval)

```
OMD[lm]:~$ tail -f  
var/log/web.log
```

```
2025-05-06 14:33:25,918 [10]  
[cmk.web.slow-views 793137]  
View name: host,  
User: cmkadmin,  
Row limit: 1000,  
Limit type: soft,  
URL variables: ['host=beta',  
'site=beta', 'view_name=host',  
'_display_options=htbfcoderu',  
'_ajaxid=0'],  
View context: {'siteopt': {},  
'svcstate': {}, 'serviceregex':  
{}, 'host': {'host': 'beta',  
'neg_host': ''}, 'site': {'site':  
'beta'}},  
Unfiltered rows: 1337,  
Filtered rows: 1337,  
Rows after limit: 0,  
Duration fetching rows: 15.03s,  
Duration filtering rows: 15.00s,  
Duration rendering view: 15.23s,  
Rendering page exceeds 0s: 15.29s
```

Site

Redis cache



UI & REST API

livestatus

disk

livestatus

Monitoring core
incl. helpers

disk

current state

disk

History

disk

Metrics

Config

liveproxyd

livestatus

Other site

Monitored devices



Livestatus performance monitoring



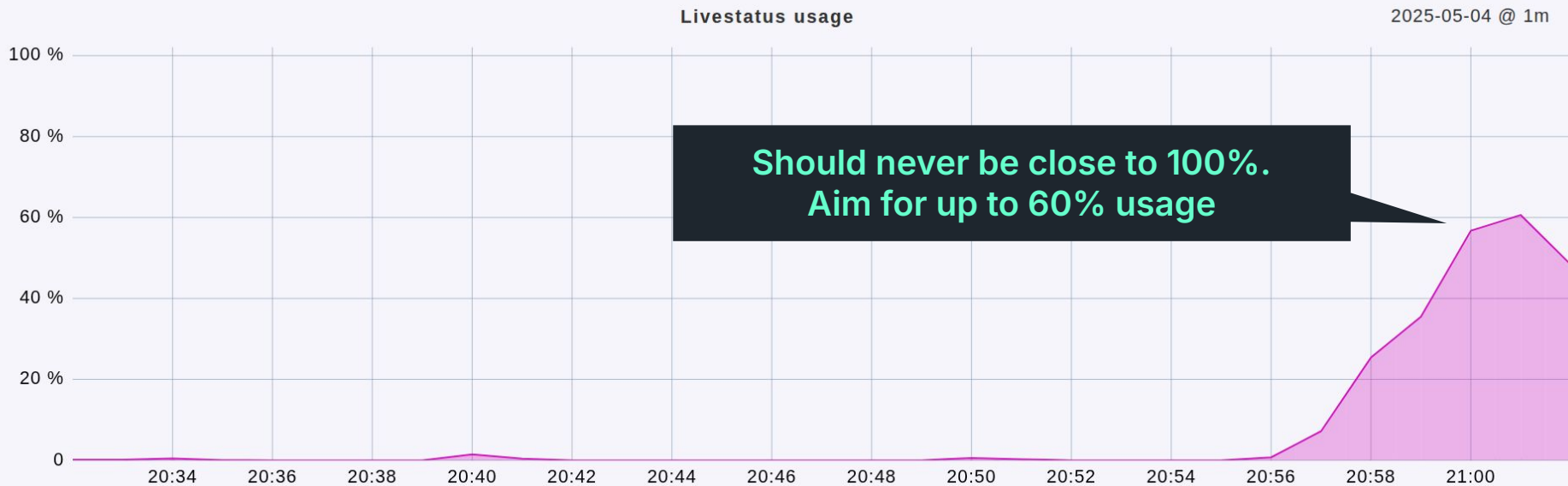
OK

OMD prod performance



Livestatus version: 2.4.0, Host checks: 60.5/s,
Service checks: 165.3/s

60.49/s / 165.31/s



Configure via global setting: Maximum concurrent Livestatus connections

Balance with livestatus proxy channels

Global settings matching 'connection'

Setup > General > Global settings matching 'connection'

Related Display Help connection x 🔍

Monitoring core

Maximum concurrent
Livestatus connections

20

Livestatus proxy

Livestatus proxy
default connection
parameters

Number of channels to
keep open:

5

Regular heartbeat: 5 sec,
2.0 sec

Timeout waiting for a free
channel: 3.0 sec

Total query timeout: 120.0
sec

Cooling period after failed
connect/heartbeat: 4.0 sec

Enable Caching: on

Livestatus proxy establishes N fixed connections, known as channels

Configure in global settings or site connection

Must be lower than the number of maximum concurrent livestatus (here $5 < 20$)

⬡ This buffer allows for ad-hoc connections

Use `cat var/log/liveproxymd.state` to get liveproxymd usage details

Tuning for best performance



**UI &
REST API**

**Stale
services**



Checkmk staleness 101

Overview

Hosts	Unhandled p.	Stale
100	4	0
Services	Unhandled p.	Stale
8605	214	1683
Events	Unhandled p.	Stale
0	0	0

Bookmarks

Add bookmark Edit

Master control

Core statistics

Service checks:	130.60/s
Host checks and smart pings:	4.86/s
Livestatus connections:	0.43/s
Average active check latency:	0.00s
Average checker latency:	0.00s
Average fetcher latency:	0.00s
Active check helper usage:	0.0%
Fetcher helper usage:	96.2%
Checker helper usage:	11.7%
Livestatus usage:	0.6%
Queued notifications:	0
Queued alerts:	0

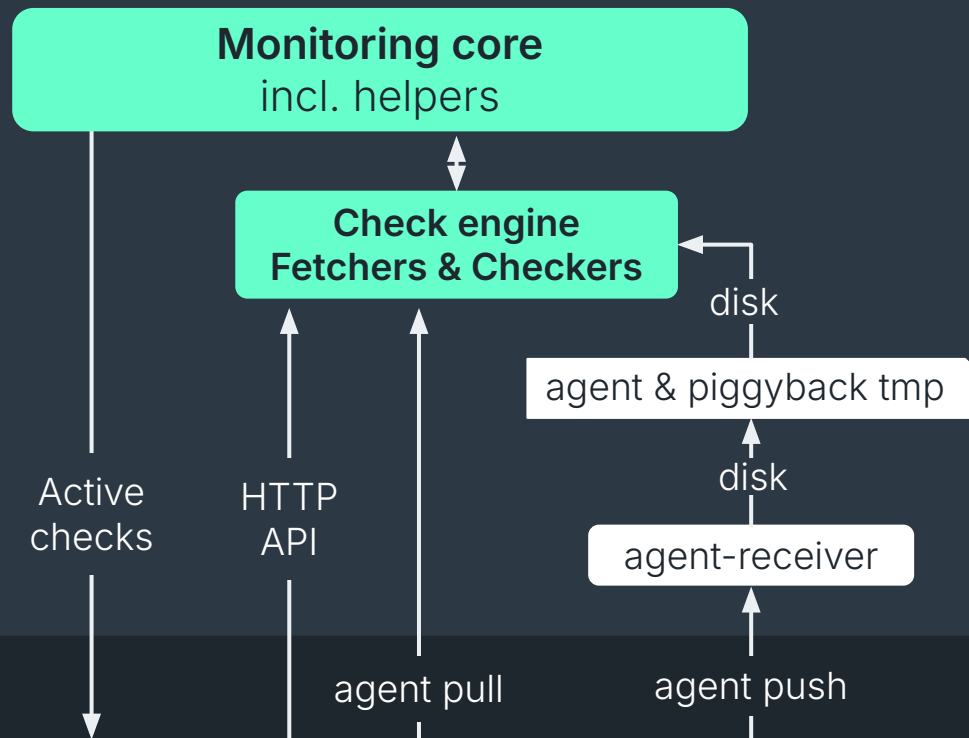
State	Service	Icons	Summary	Age	Checked
OK	Memory	☰ 📁 ⚙️	Total virtual memory: 7.90% - 3.12 GiB of 39.6 GiB, 9 additional details available	10 m	102 s

Stale: not refreshed for the last 1.5 check intervals

Example: 90s, if check interval = 60s.

Note: Configurable via global setting: Staleness value to mark hosts / services stale

Site



Monitored devices



Reasons for staleness

Check_MK and Check_MK discovery services

- Highly utilized fetcher helpers
- Highly utilized checker helpers

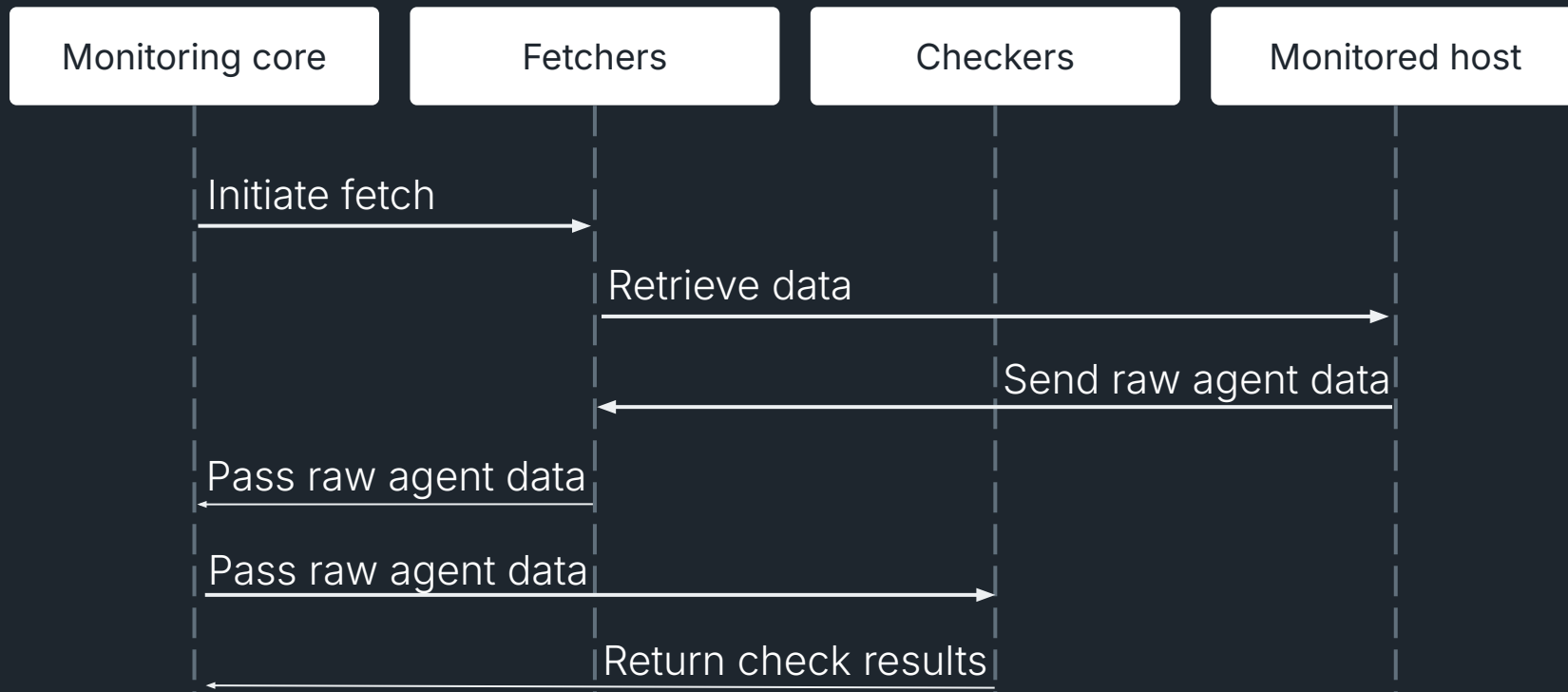
Active checks (Check_MK inventory , HTTP, TCP, ...)

- Highly utilized active check helpers

Other services, like CPU load or Memory

- Typically related to a failing Check_MK service

An execution of a Check_MK service





Reasons for staleness

Check_MK and Check_MK discovery services

- Hexagon icon Highly utilized fetcher helpers
- Hexagon icon Highly utilized checker helpers

Active checks (Check_MK inventory , HTTP, TCP, ...)

- Hexagon icon Highly utilized active check helpers

Other services, like CPU load or Memory

- Hexagon icon Typically related to a failing Check_MK service

Events	Unhandled p.	Stale
0	0	0

Bookmarks

Add bookmark

Edit

Master control

Core statistics

Service checks: 130.60/s

Host checks and smart pings: 4.86/s

Livestatus connections: 0.43/s

Average active check latency: 0.00s

Average checker latency: 0.00s

Average fetcher latency: 0.00s

Active check helper usage: 0.0%

Fetcher helper usage: 96.2%

Checker helper usage: 11.7%

Livestatus usage: 0.6%

Queued notifications: 0

Queued alerts: 0

Helper performance monitoring



OK

OMD prod performance

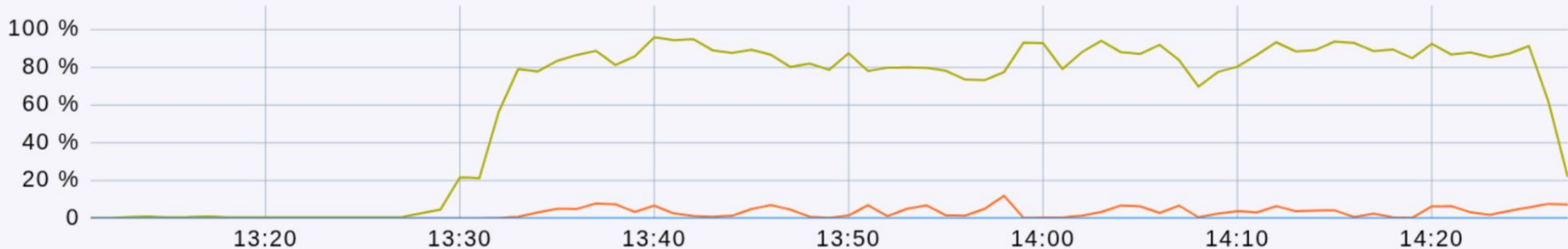


Livestatus version: 2.4.0, Host checks: 60.5/s,
Service checks: 165.3/s

60.49/s / 165.31/s

Helper usage

2025-05-05 @ 1m



Minimum

Maximum

Average

Last

- Fetcher helper usage
- Checker helper usage
- Active check helper usage

0.15 %	95.99 %	62.3 %	21.87 %
0 %	11.89 %	2.72 %	7.3 %
0 %	0 %	0 %	0 %

Configure via global setting: Maximum concurrent Checkmk fetchers / checkers / active checks

Guidance for configuring fetchers/checkers

Fetchers (default: 13)

- ⬢ Network bound
- ⬢ ~70MB per process
- ⬢ Limiting factor: Memory
- ⬢ Too many slow down startup
- ⬢ Aim for less than 60% utilization

Checkers (default: 4)

- ⬢ CPU bound
- ⬢ >200MB per process
- ⬢ Limiting factors: CPU, Memory
- ⬢ # checkers smaller than # CPU cores
- ⬢ Too many checkers ⇒ Context switches
- ⬢ Aim for less than 80% utilization

Also covered by new process monitoring!

Tuning: Identify long-running checks

Checkers only apply to Check_MK & Check_MK Discovery service!




checkmk


Monitor


Customize


Setup

Service check durations

Monitor > History > Service check durations

Ready to monitor your own services

Commands Services Export Display Help     

Local site play				
State	Host	Service	Duration	Summary
OK	deployment_eks_otel-demo_shop-flagd	Check_MK	30.5 s	[special_otel] Success, [piggyback] Success, [kubernetes.company.com], execution time 11.0 sec
OK	aws-web	Check_MK	11.1 s	[agent] Success, [special_otel] Success, [kubernetes.company.com], execution time 11.0 sec
OK	play.checkmk.com	Check_MK	4.39 s	[special_otel] Success, [piggyback] Success, [kubernetes.company.com], execution time 11.0 sec
OK	deployment_eks_otel-demo_shop-frontend	Check_MK	4.16 s	[special_otel] Success, [piggyback] Success, [kubernetes.company.com], execution time 11.0 sec
OK	deployment_eks_otel-demo_shop-frauddetectionservice	Check_MK	3.93 s	[special_otel] Success, [piggyback] Success, [kubernetes.company.com], execution time 11.0 sec
OK	deployment_eks_otel-demo_shop-checkoutservice	Check_MK	3.82 s	[special_otel] Success, [piggyback] Success, [kubernetes.company.com], execution time 11.0 sec

Tuning fetchers: 99% of time tuning SNMP



Monitoring with SNMP: Troubleshooting in God Mode



By Alexander Wilms on Jul 14, 2020

🕒 *Reading time 5 minutes*

<https://checkmk.com/blog/snmp-troubleshooting>

