The Architecture of the MCAO RTC for the Daniel K. Inouye Solar Telescope

and

Matilda - a fast MVM library for AO RTC

Dirk Schmidt, Andrew Beard

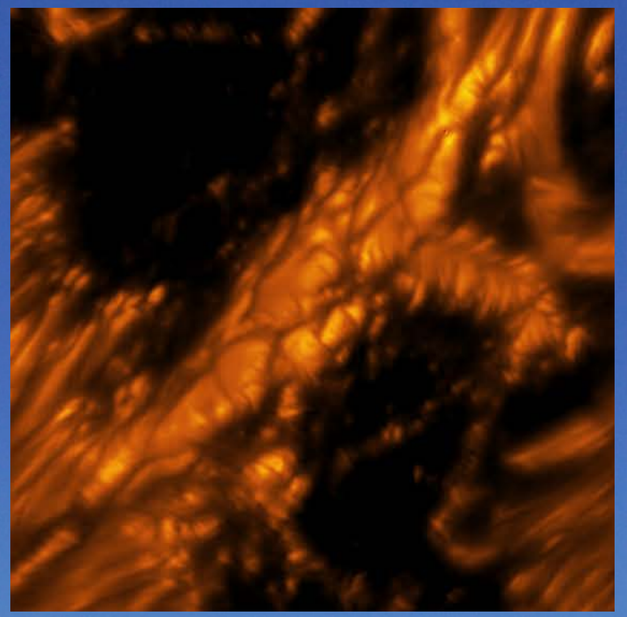
National Solar Observatory, Boulder (Colorado), USA

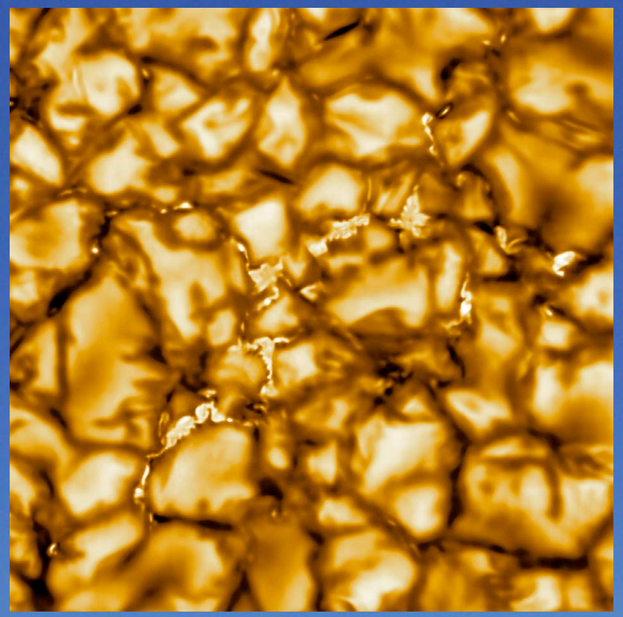
ESO RTC4AO Workshop

November 2023

Sponsored by the
National Science Foundation of the United States of America

The Daniel K. Inouye Solar Telescope

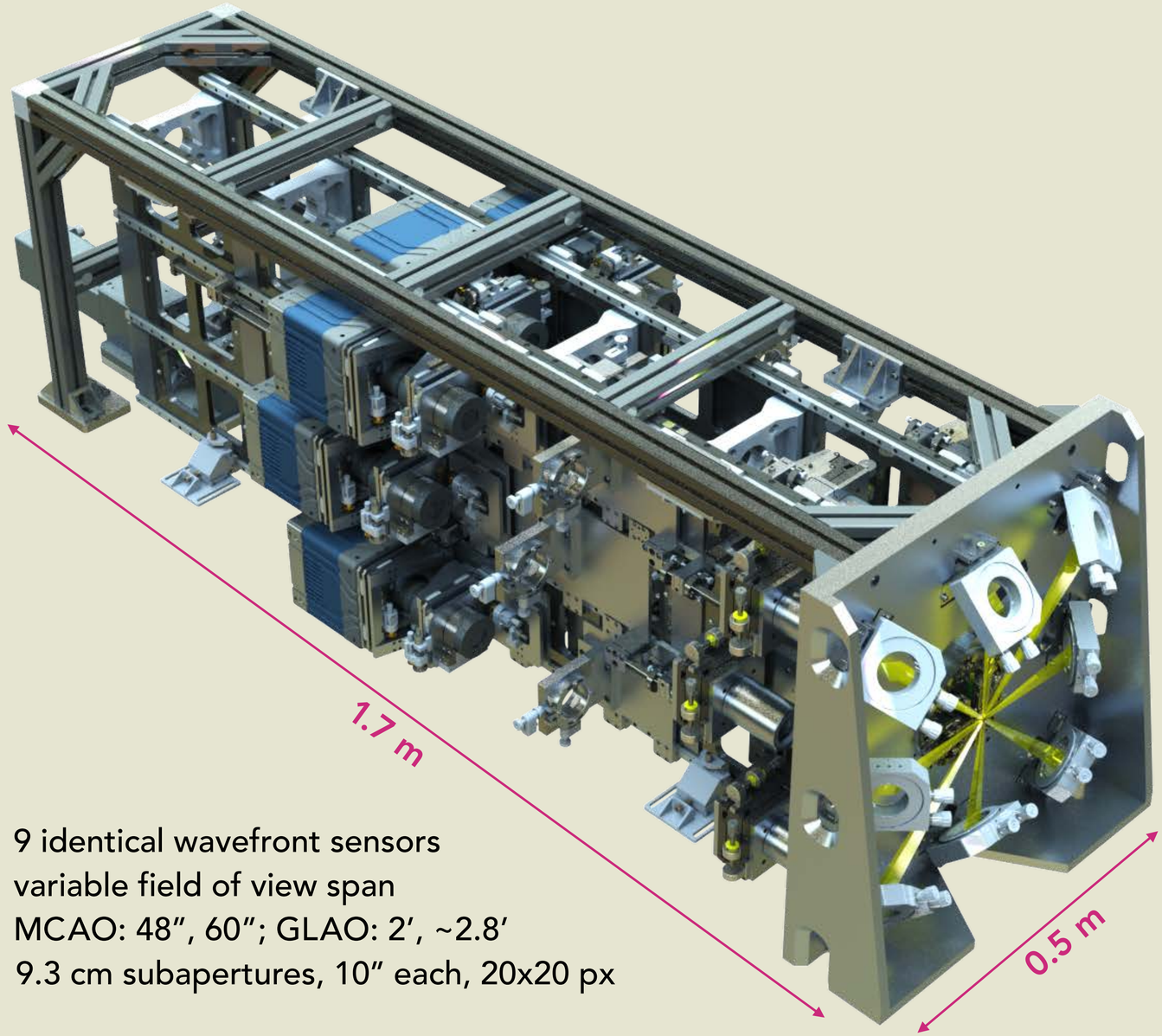




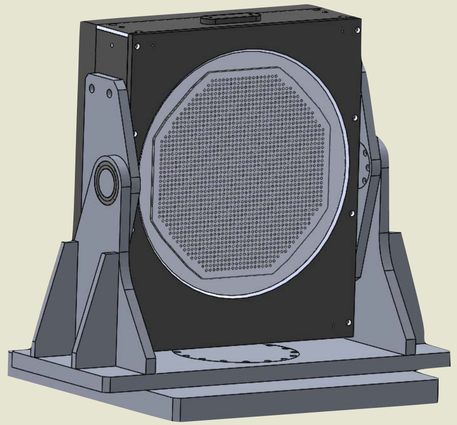
- 4-meter off-axis Gregory design (no spider)
- Science commissioning phase since 2022
- Classical Adaptive Optics with 1600 actuators
  - FPGA+CPU hybrid, 1970 fps

- MCAO / GLAO upgrade in fabrication
- GLAO deployment (new RTC + new wavefront sensor) by end of 2024, additional DMs later

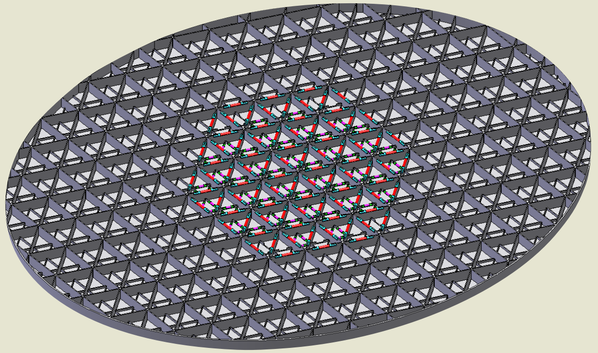
MCAO Upgrade: new WFS system, 2 additional DMs, new RTC



- 9 identical wavefront sensors
- variable field of view span
  MCAO: 48″, 60″; GLAO: 2′, ~2.8′
- 9.3 cm subapertures, 10″ each, 20x20 px

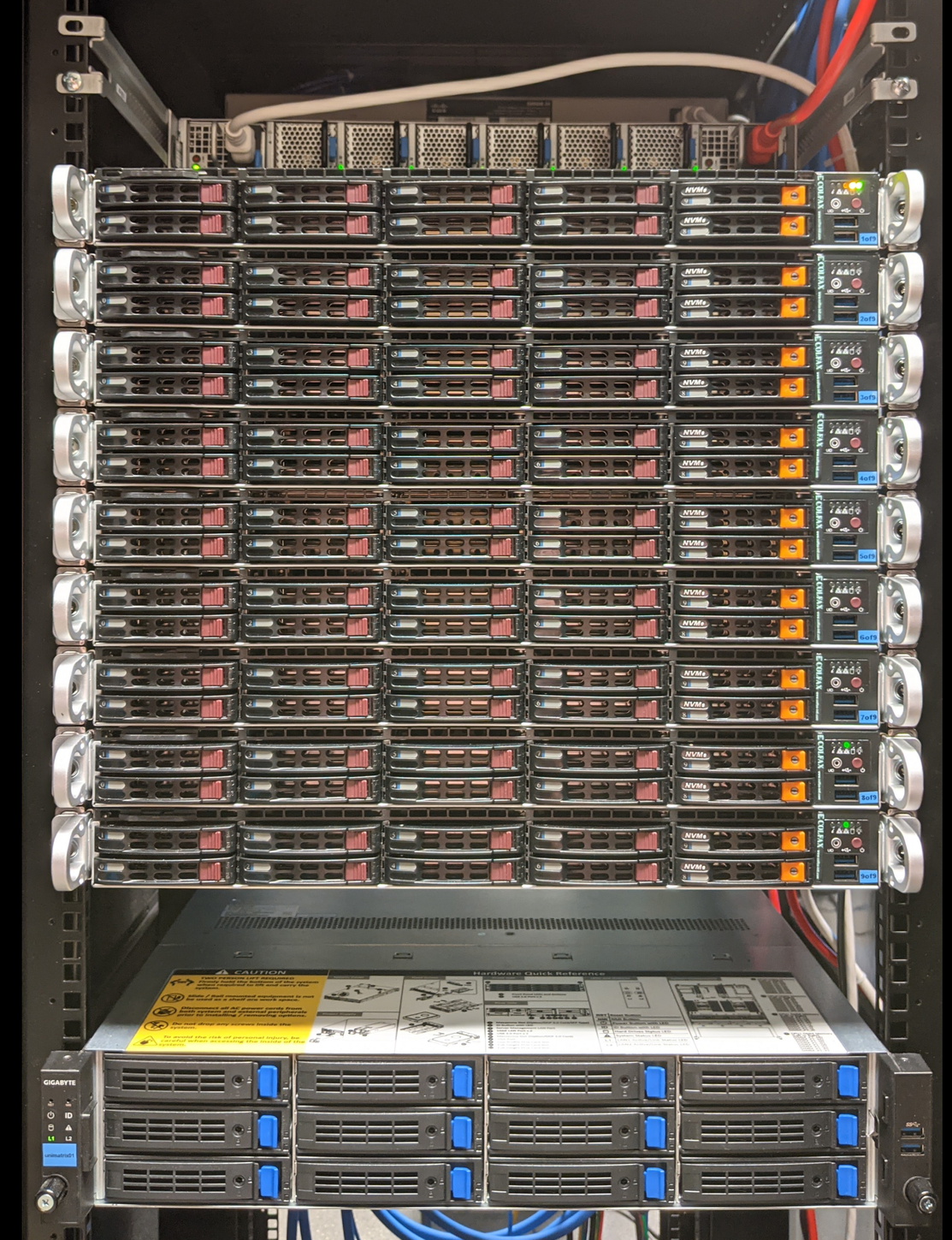


$DM_{4\ km}$
- 1600 actuators
- Final polishing



$DM_{11\ km}$
- 252 actuators
- FDR pending

11817 subapertures and 3454 actuators total



- Infiniband HDR 200 Gb/s network

- 9× Supermicro A+ 1114S-WTRT (“1of9”,…,“9of9”, “The Nine”)
  - 1× AMD Epyc 7742 CPU (64 cores, 2.25 GHz)
  - 1× Euresys Coaxlink Octo frame grabber (currently)
    1× Kaya Komodo II CLHS frame grabber (in final system)
  - 1× CoaXPress camera, 864×860, 8 bit, 2 kHz (currently)
    1× CameraLink HS camera, 1200×1200, 12 bit, 2 kHz (in final system)
  - 1× Mellanox ConnectX-6, dual-port Infiniband adapter

- 1× Gigabyte R282-Z91 (“unimatrix01”)
  - 2× AMD Epyc 7742 CPUs (128 cores, 2.25 GHz)
  - 2× Mellanox ConnectX-6, dual-port Infiniband adapter
  - 1× 10 Gb/s adapter for DM commands
  - 1× 8K Nvidia graphics adapter for GUI

Real-time Control Loop Cycle

KAOS Evo 2 rev 1756

GUI on unimatrix01 (6000 x 3400 pixels)

DKIST MCAO Development System

Main Menu

Run AO | Calibrate AO | System Service

Check WFS Alignment... | Check WFS Alignment... | Move Mirrors...

Take Flat Images... | Reference WFS... | Move Motors...

Ignore Subapertures... | Take Dark Images... | Camera Settings...

Run AO... | Take Flat Images... | Save Camera Snapshot...

Calibrate System... | Save Camera Burst

Quit

Run AO Menu

Leave Menu

AO Control Mode

- No correction
- Image Stabilization (on-axis)
- Image Stabilization (full field)
- Classical AO
- Ground-layer AO
- MCAO (0 + 4km)
- MCAO (0 + 11km)
- MCAO (0 + 4 + 11km)

Utilities

- Update Reference Image
- Periodic Reference Update
- Hold avg. tip-tilt position — Reset
- Wavefront Sensor Settings...
- Servo Parameters...
- Wavefront Error Analysis...
- Mode Settings...
- Update Static Aberrations
- .. — 0/10000

Automatic Gain Optimization

Enabled | Save | Reset

Status

Mode: No correction
Seeing (r₀): 0.0 cm
Subaperture Contrast: 0.0% (#0)
Brightness: 0.0000 (#0)
Wavefront error after correction: NA
mean ccr shift: 3.91 px
Active modes: 0 + 0 + 0 + 0 = 0
Timing: 500 µs (2000 Hz)
Jitter / Latency: 16.9 µs / 40 µs
Skipped frames: 0 events
Current Buffer: 19
Reference buffer: 40%
Reference period: 0%

Display Settings

Display reduced subaperture data | Display wavefront slope vectors | Display control loop data

Table view | Plot view

Image shifts in correlation fields

cam	dir	subap	sx [px]	sy [px]
0	0	0	-4.756	1.232
0	0	1	-1.133	-3.519
0	0	2	-1.801	-0.253
0	0	3	-1.939	4.578
0	0	4	4.487	0.427
0	0	5	-2.000	-2.000
0	0	6	-4.310	3.168
0	0	7	0.062	5.005
0	0	8	4.851	2.073
0	0	9	-4.869	-0.056
0	0	10	2.748	-0.618
0	0	11	5.247	-2.021
0	0	12	3.114	-2.984
0	0	13	-2.878	-5.287
0	0	14	4.993	3.529
0	0	15	4.018	-0.669
0	0	16	1.957	4.123
0	0	17	2.945	1.471
0	0	18	0.155	0.074
0	0	19	1.021	3.057
0	0	20	-0.873	-0.754
0	0	21	-0.848	0.594
0	0	22	0.449	-0.760
0	0	23	1.903	3.785
0	0	24	-3.144	-3.126
0	0	25	-1.564	0.756
0	0	26	-1.980	4.004
0	0	27	2.416	2.619
0	0	28	4.897	-2.095
0	0	29	-3.101	-4.087
0	0	30	0.118	1.755
0	0	31	2.248	4.187
0	0	32	-6.070	4.974
0	0	33	-0.562	-3.092
0	0	34	3.947	-0.102
0	0	35	-5.648	-6.343
0	0	36	-0.285	-2.995
0	0	37	2.017	1.705
0	0	38	2.430	-1.299
0	0	39	-4.801	2.043
0	0	40	0.121	-2.103
0	0	41	0.834	-0.020
0	0	42	-3.103	5.051
0	0	43	-5.806	-2.098
0	0	44	-0.287	1.133
0	0	45	0.235	1.408
0	0	46	2.000	3.000
0	0	47	2.446	-6.043
0	0	48	-0.175	2.102
0	0	49	-1.425	2.661
0	0	50	-4.869	0.105
0	0	51	-1.248	-2.185
0	0	52	0.934	-3.000
0	0	53	-6.114	-3.679
0	0	54	-3.026	-0.076
0	0	55	0.591	3.887
0	0	56	0.259	3.099
0	0	57	2.914	-0.941
0	0	58	1.684	-0.325
0	0	59	-1.000	-2.000
0	0	60	-1.000	0.000
0	0	61	-5.903	1.101
0	0	62	-0.764	-3.366
0	0	63	-0.864	-4.056
0	0	64	4.133	-3.919
0	0	65	1.763	4.991
0	0	66	-1.814	2.000
0	0	67	-2.221	2.461
0	0	68	-1.291	0.801
0	0	69	2.129	-0.179
0	0	70	3.781	-1.910
0	0	71	2.854	3.044
0	0	72	-3.935	-3.836
0	0	73	4.000	-3.000
0	0	74	3.529	-3.712
0	0	75	2.699	0.003
0	0	76	3.186	2.990
0	0	77	-0.471	4.924
0	0	78	-1.559	0.200
0	0	79	-6.000	-6.000
0	0	80	0.082	-3.469
0	0	81	0.931	-0.969
0	0	82	-0.523	-6.018
0	0	83	-4.740	-1.841
0	0	84	-4.620	-3.328
0	0	85	-4.328	-2.455
0	0	86	-1.000	-5.000
0	0	87	-6.598	3.969
0	0	88	4.107	-0.214
0	0	89	-1.864	1.949
0	0	90	-0.972	2.983
0	0	91	1.073	1.962
0	0	92	-3.030	-4.055
0	0	93	-1.896	2.568
0	0	94	-1.922	-3.290
0	0	95	-1.328	3.823
0	0	96	-5.927	-0.823
0	0	97	4.054	3.818
0	0	98	0.932	-2.130
0	0	99	-1.255	-3.094
0	0	100	-4.764	-5.478
0	0	101	1.728	1.763
0	0	102	-5.000	1.000
0	0	103	-4.137	1.207
0	0	104	-3.256	3.518
0	0	105	-0.917	1.202

Modal RMS errors [rad]

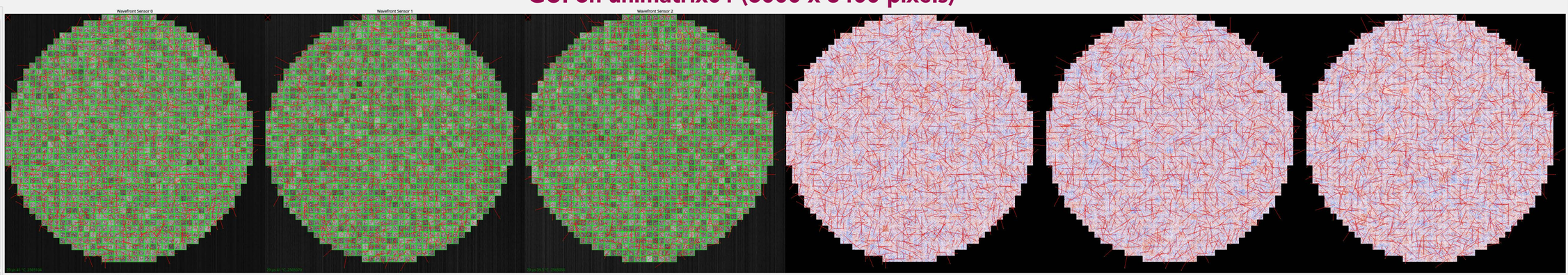
mode	rms	res avg	tot	tot avg
0	62.456	20.992	62.456	
1	63.362	20.904	63.362	
2	62.352	19.683	62.352	
3	62.352	19.683	62.352	
4	62.352	19.683	62.352	
5	62.352	19.683	62.352	
6	62.352	19.683	62.352	
7	62.352	19.683	62.352	
8	62.352	19.683	62.352	
9	62.352	19.683	62.352	
10	62.352	19.683	62.352	
11	62.352	19.683	62.352	
12	62.352	19.683	62.352	
13	62.352	19.683	62.352	
14	62.352	19.683	62.352	
15	62.352	19.683	62.352	
16	62.352	19.683	62.352	
17	62.352	19.683	62.352	
18	62.352	19.683	62.352	
19	62.352	19.683	62.352	
20	62.352	19.683	62.352	
21	62.352	19.683	62.352	
22	62.352	19.683	62.352	
23	62.352	19.683	62.352	
24	62.352	19.683	62.352	
25	62.352	19.683	62.352	
26	62.352	19.683	62.352	
27	62.352	19.683	62.352	
28	62.352	19.683	62.352	
29	62.352	19.683	62.352	
30	62.352	19.683	62.352	
31	62.352	19.683	62.352	
32	62.352	19.683	62.352	
33	62.352	19.683	62.352	
34	62.352	19.683	62.352	
35	62.352	19.683	62.352	
36	62.352	19.683	62.352	
37	62.352	19.683	62.352	
38	62.352	19.683	62.352	
39	62.352	19.683	62.352	
40	62.352	19.683	62.352	
41	62.352	19.683	62.352	
42	62.352	19.683	62.352	
43	62.352	19.683	62.352	
44	62.352	19.683	62.352	
45	62.352	19.683	62.352	
46	62.352	19.683	62.352	
47	62.352	19.683	62.352	
48	62.352	19.683	62.352	
49	62.352	19.683	62.352	
50	62.352	19.683	62.352	
51	62.352	19.683	62.352	
52	62.352	19.683	62.352	
53	62.352	19.683	62.352	
54	62.352	19.683	62.352	
55	62.352	19.683	62.352	
56	62.352	19.683	62.352	
57	62.352	19.683	62.352	
58	62.352	19.683	62.352	
59	62.352	19.683	62.352	
60	62.352	19.683	62.352	
61	62.352	19.683	62.352	
62	62.352	19.683	62.352	
63	62.352	19.683	62.352	
64	62.352	19.683	62.352	
65	62.352	19.683	62.352	
66	62.352	19.683	62.352	
67	62.352	19.683	62.352	
68	62.352	19.683	62.352	
69	62.352	19.683	62.352	
70	62.352	19.683	62.352	
71	62.352	19.683	62.352	
72	62.352	19.683	62.352	
73	62.352	19.683	62.352	
74	62.352	19.683	62.352	
75	62.352	19.683	62.352	
76	62.352	19.683	62.352	
77	62.352	19.683	62.352	
78	62.352	19.683	62.352	
79	62.352	19.683	62.352	
80	62.352	19.683	62.352	
81	62.352	19.683	62.352	
82	62.352	19.683	62.352	
83	62.352	19.683	62.352	
84	62.352	19.683	62.352	
85	62.352	19.683	62.352	
86	62.352	19.683	62.352	
87	62.352	19.683	62.352	
88	62.352	19.683	62.352	
89	62.352	19.683	62.352	
90	62.352	19.683	62.352	
91	62.352	19.683	62.352	
92	62.352	19.683	62.352	
93	62.352	19.683	62.352	
94	62.352	19.683	62.352	
95	62.352	19.683	62.352	
96	62.352	19.683	62.352	
97	62.352	19.683	62.352	
98	62.352	19.683	62.352	
99	62.352	19.683	62.352	
100	62.352	19.683	62.352	
101	62.352	19.683	62.352	
102	62.352	19.683	62.352	
103	62.352	19.683	62.352	
104	62.352	19.683	62.352	
105	62.352	19.683	62.352	

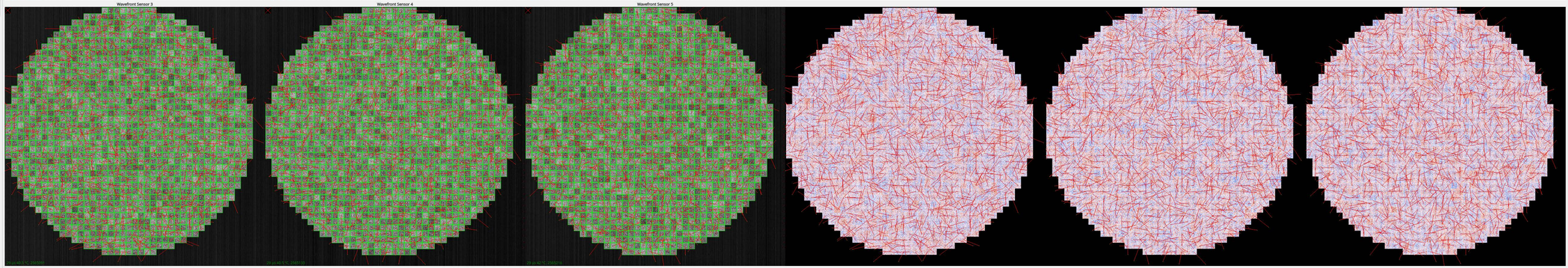
Actuator values

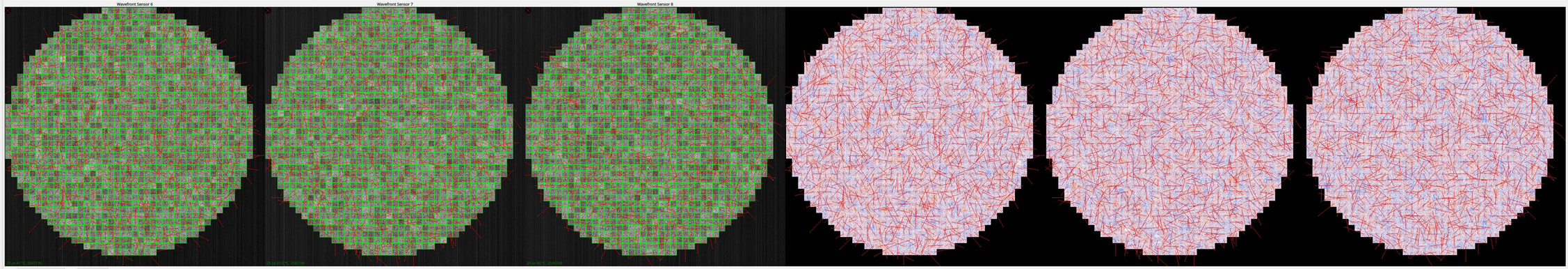
act	value
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100
101	101
102	102
103	103
104	104
105	105

Directional error

dir	mean shift [px]







Show live raw image with boundaries of correlation fields

Image levels

Raw: min: 0 max: 255

Calibrated: min: 0.000 max: 1.000

Display: min: 0 max: 255 gamma: 1.00

camera images

correlation maps

RTC Software is enhanced version KAOS Evo 2

- KAOS Evo 2 originally developed for Gregor at Kiepenheuer-Institute for Solarphysics (Thomas Berkefeld & I)
- Branched off in 2013 at National Solar Observatory for Big Bear Solar Observatory
  "Clear" (MCAO), AO308 II (CAO), GLAO, prominence AO

- Linux application, runs completely on x86-64 CPUs
- Complete: Control Loop and all calibrations integrated, nothing is done via external software
- GUI: designed for the operator
- Super flexible: almost everything is in config files, few performance sensitive parameters are compile time constants.
- Supports multiple SH-WFSs (cameras), each WFS can have multiple guide-regions

- Plugin-based hardware support
  - Euresys, EDT, Kaya, Active Silicon frame grabbers
  - various camera models (Mikrotron, Adimec, Photonfocus, First Light Imaging, Optronis, Dalsa)
  - Riptide Realtime DM Xinetics interface (+ obsolete in-house solutions)

- Low single-digit RMS jitter (on well tuned systems, < 2 µs on any of The Nine)

RTC is enhanced version KAOS Evo 2

- divided into real-time back-end and GUI (Qt),
- uses both POSIX Shared Memory and threads
- “Real-time” application
  - Tuned hardware and operating system
  - Control loop threads use SCHED_FIFO with priority 94
  - Threads pinned on dedicated CPU cores
  - Threads take 100% of each assigned core and never yields the CPU (no mutexes, usleep(), etc.)
  - RMS jitter < 2 µs (per host)

Low-latency tuning and customization for DKIST MCAO RTC

Enhancing hardware determinism and isolating CPU cores

- Consult tuning guides for HPC and low-latency applications by hardware vendors, e.g.
  - AMD "*Workload Tuning Guide for AMD EPYC 7002 Series Processor Based Servers*"
  - AMD "*Performance Tuning Guidelines for Low Latency Response on AMD EPYC-Based Servers*"
  - AMD "*Performance Tuning Guidelines for Low Latency Response on AMD EPYC™ 7002 Series Processor Based Servers*"
  - AMD "*High Performance Computing: Tuning Guide for AMD EPYC 7002 Series Processors*"
  - AMD "*HPC Tuning Guide for AMD EPYC Processors*"
- Custom BIOS from Supermicro on The Nine
  - Exposures features not found in default BIOS
- BIOS settings synchronized across servers via Redfish
  - Disabled: ASPMSupport, CorePerformanceBoost, DRAMScrubTime, HighPrecisionEventTimer, IOMMU, SMEE, SMTControl, SR-IOVSupport, SVMMode, TSME, WatchDogFunction
  - APBDIS=1, DeterminismSlider=Performance, EfficiencyModeEn=Auto, FixedSOCPstate=P0, LocalAPICMode=xAPIC

Low-latency tuning and customization for DKIST MCAO RTC

Linux tweaking, isolating CPU cores and reducing operating system jitter

- Debian Linux 11 with stock “linux-image-amd64” kernel
- unimatrix01 runs KDE desktop on 8k display (primary user interface)
- Follow “Reducing OS jitter due to per-cpu kthreads” guide
  - https://www.kernel.org/doc/Documentation/kernel-per-CPU-kthreads.txt
- LD_PRELOAD a custom library to prevent unwanted programs from setting their CPU affinities to isolated cores
  - overwrites sched_setaffinity() and pthread_setaffinity_np()
  - keeps “intrusive” programs which ignore preset affinities from freezing when AO RTC software is running
- All root file systems are clones from 1of9
  - Root file systems are read-only to prevent “accidental” and unrecorded changes
  - /home and /var on separate partitions, no swap partition (/var is read+write and contains symlinks for adjtime, resolv.conf, blkid.tab, etc.)

Low-latency tuning and customization for DKIST MCAO RTC

Linux tweaking, isolating CPU cores and reducing operating system jitter

```
# kernel command line
acpi_irq_nobalance apparmor=0 audit=0 clocksource=tsc cpufreq.default_governor=performance cpuidle.off=1 idle=poll
irqaffinity=0-7 isolcpus=8-63 mce=off mitigations=off nmi_watchdog=0 nohz=on nohz_full=8-63 noirqbalance nosoftlockup=0
nowatchdog processor.max_cstate=0 rcu_nocbs=8-63 rcu_nocb_poll=8-63 selinux=0 skew_tick=1 tsc=reliable


# /etc/systemd/system/cpus-offline-online.service
[Unit]
Description=Disable and re-enable CPUs 8-63
After=network.target sshd.service
[Service]
Type=oneshot
ExecStart=/bin/sh -c 'sleep 60 && chcpu --disable 8-63 && chcpu --enable 8-63'
RemainAfterExit=no
StandardOutput=journal
[Install]
WantedBy=default.target

# /etc/sysctl.d/kaos.conf
kernel.nmi_watchdog           = 0
kernel.numa_balancing         = 0
kernel.hung_task_timeout_secs = 0
kernel.sched_rt_runtime_us    = -1
kernel.randomize_va_space     = 0
vm.swappiness                 = 0
kernel.sched_tunable_scaling  = 0
kernel.timer_migration        = 0
vm.stat_interval              = 10

# /etc/security/limits.d/kaos.conf
@aousers            hard    memlock         40000000
@aousers            soft    memlock         40000000
@aousers            -       rtprio          99
@aousers            -       nice            -19
```

KAOS Evo 2 design principles

- "C-style" C++
- Data layout designed (refactored) for SIMD architectures
  - "structures of arrays" where vectorization and contiguous data access is beneficial, e.g. slope, mode and actuator data
  - "arrays of structures" where not beneficial and to prevent "false sharing" of cache lines between several threads, e.g. slope, other subaperture data
  - all arrays forcefully aligned on cache line boundaries
  - not NUMA optimized (yet?), all data is local to first real-time core
- Automatic and mandated vectorization using #pragma omp simd
  - Analysis of vectorization reports, benchmarking and profiling (Intel ICC, Advisor, V-Tune, Compiler Explorer) to verify and improve
  - Intel ICC usually creates faster code than GCC, Clang (AOCC, Intel ICX) **even on AMD**
- Lock-free
  - no simultaneous reads and writes to the same variable by different threads
  - simultaneous writes to the same cache line are rare, too
- Thread synchronization based on busy-looping (100% CPU use)
  - only three thread synchronization barriers per loop cycle (compute shifts, modes, and actuators)
  - one atomic integer per thread, value specifies which task to execute or finished state
  - lowest overhead on AMD Epyc appears with **4 integers per cache line** despite false sharing (2 µs for 64 threads on AMD 7742)
- Aiming for "branchless" code where it's easy and reasonable

Clusterization of KAOS Evo 2

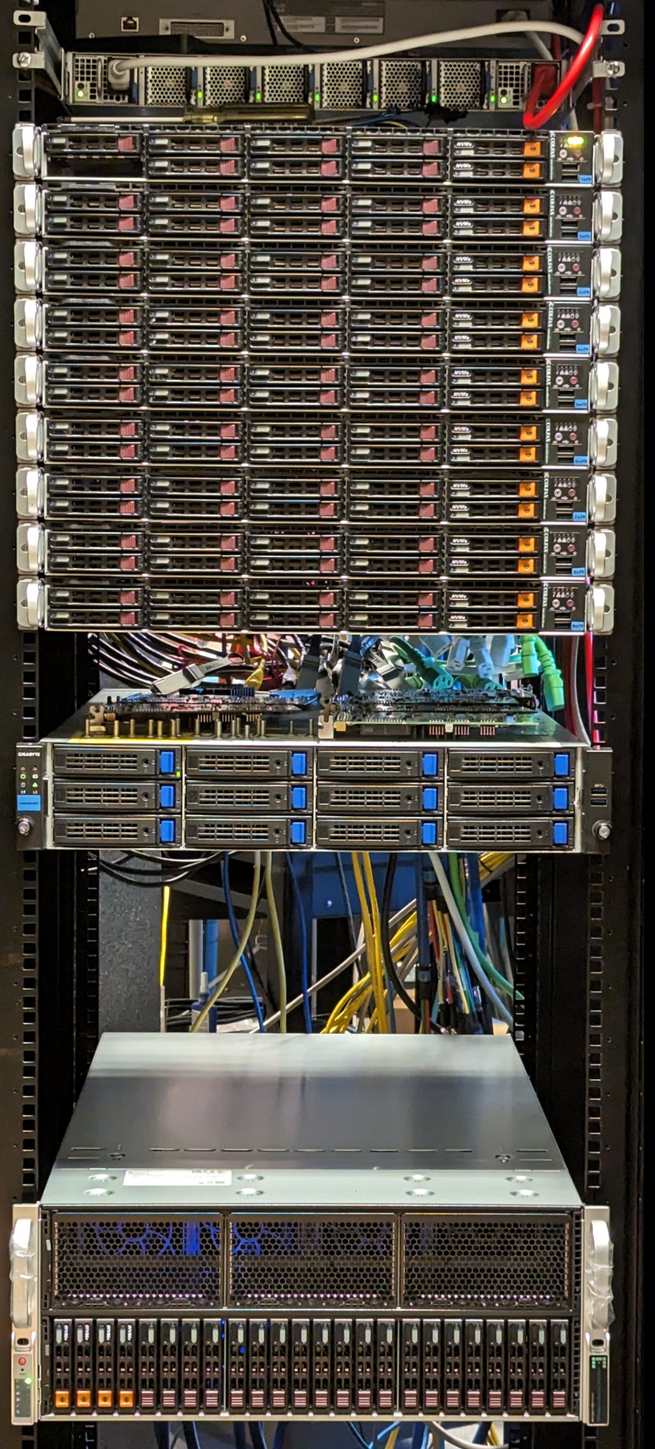
- Originally shared memory application

- Extending to cluster application
  - RDMA over Infiniband (libibverbs, librdmacm)
    - tested various write/read mechanisms to identify lowest latency approach
      - sender: RDMA_WRITE operation
      - receiver: polling raw memory (busy-looping)
    - latency: ~10 µs to send 4000 32-bit numbers from all Nine to unimatrix01

  - RDMA layer complete, basic control loop running at 2000 Hz
    - no correlation reference updating, frequency filtering, etc. yet

  - Command layer (rpclib) almost complete

  - GUI working



Simulation-based Testing of RTC with Blur

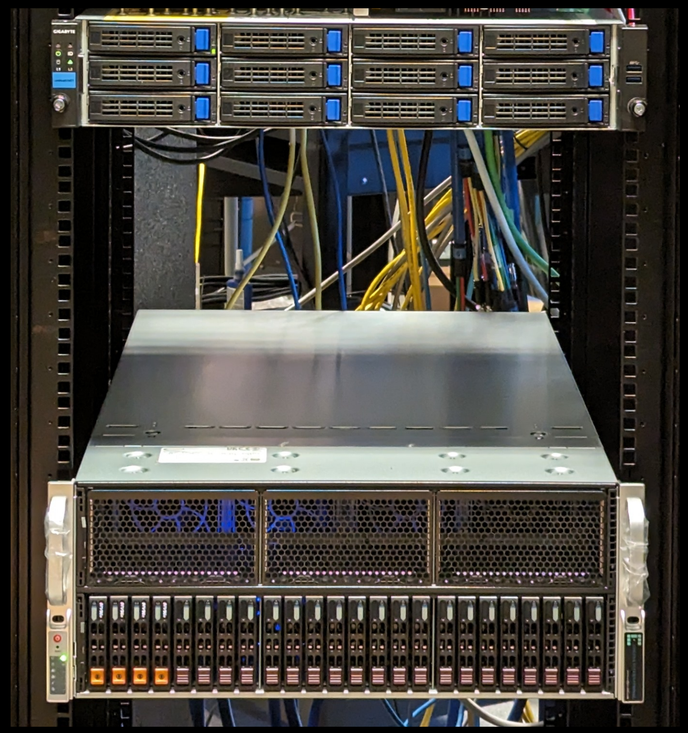
Blur: NSO's solar AO simulator

- receives:
  actuator commands
  (via ethernet)
- models:
  deformable mirrors,
  turbulent and windy atmosphere,
  imaging of extended source through atmosphere,
  imaging noise
- outputs:
  simulated wavefront sensor camera images
  (via **CoaXPress / CameraLink simulators**, shared memory, or ethernet)

- Plugs directly into RTC instead of cameras

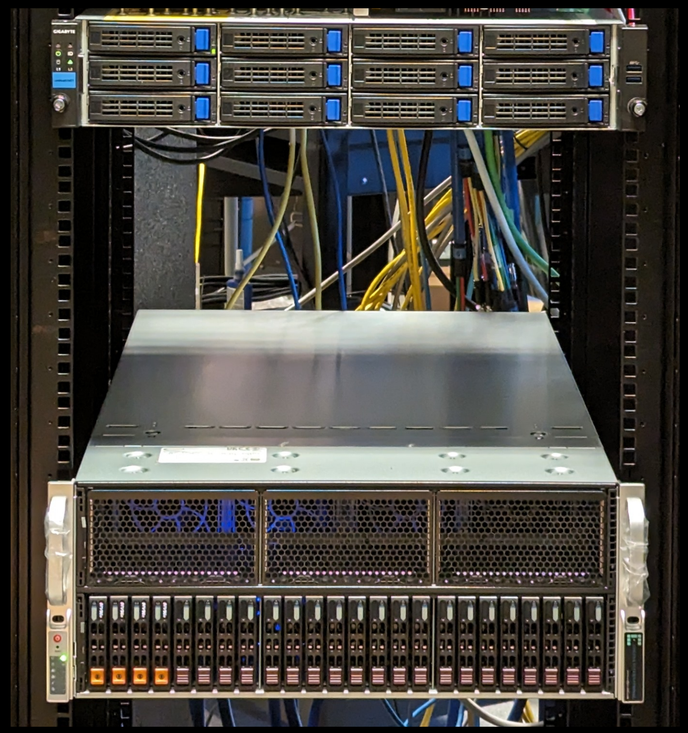
- Full DKIST MCAO at 10 fps on dual AMD 7742

- 1× Supermicro AS4125GS-TNRT ("chaotica")
  - 2× AMD Epyc 9654 CPUs (192 cores, 2.4 GHz)
  - 9× Kaya Chameleon II CoaXPress simulator



9 PCIe slots, 192 Zen 4 cores....

How many wavefront sensors can KAOS Evo 2 process on chaotica?



9 PCIe slots, 192 Zen 4 cores....

How many wavefront sensors can KAOS Evo 2 process on chaotica?

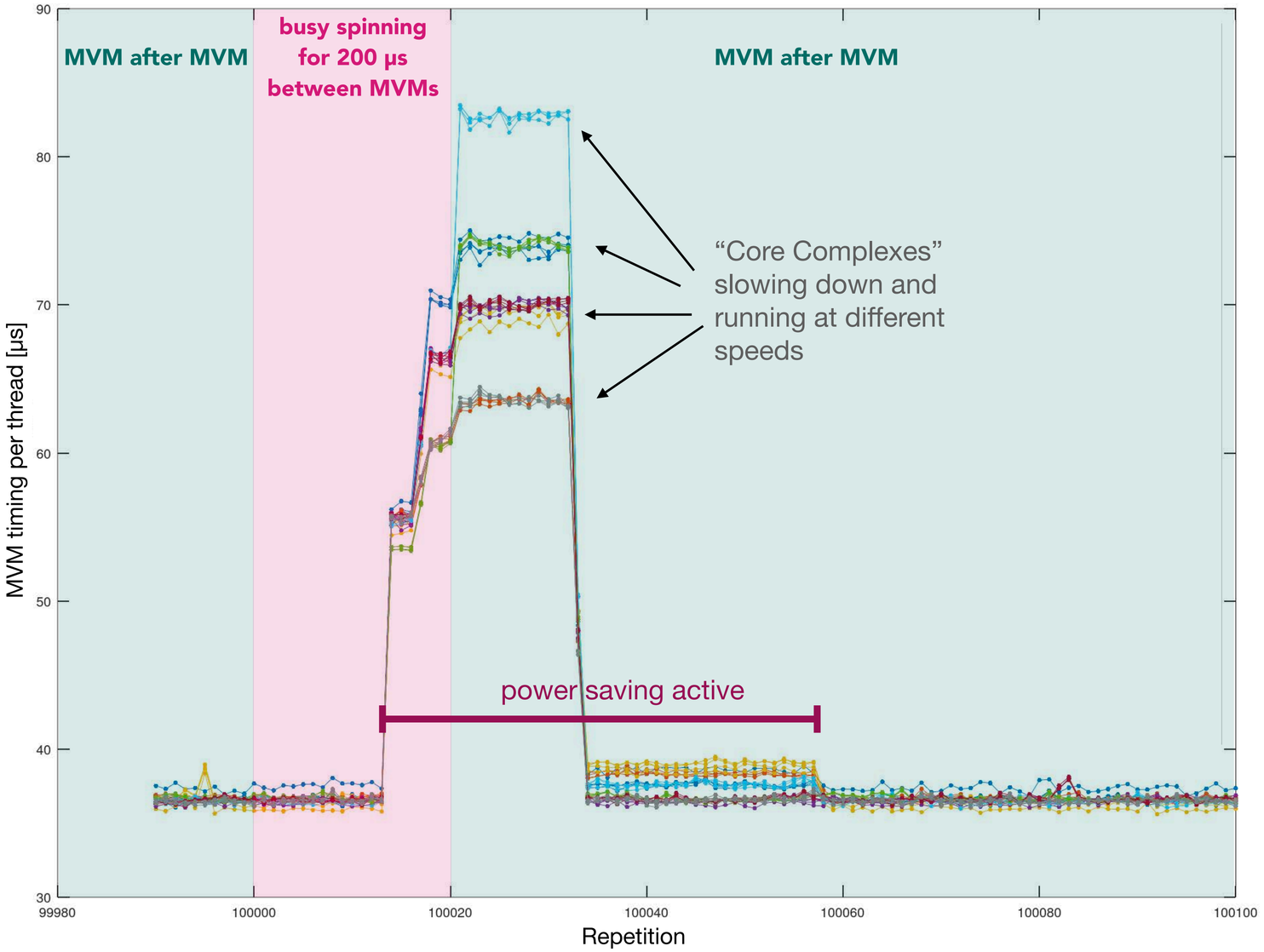
Nine!

(at 1860+ fps using 160 cores)

Now considering and evaluating switching gears
and run RTC on 1 or 2 computers only.
Final MVM on GPU or unimatrix01?

Throtteling in AMD's 2nd Gen Epyc CPUs

MVM slows down when busy waiting for next MVM



AMD 7742 CPUs slow down

- All power savings and frequency scalings deactivated in BIOS and Linux (system tuned for low latency)
- Only seen on AMD Zen 2 CPUs (not on Intel Broadwell, Skylake, Cascade Lake, or AMD Zen 4)
- Seems to happen inside the CPU
- CPU slows down whenever deviated from full-size MVM for a short while

- "Solutions":
  - Keep MVM going **at full size,** disregard output
    - Costs additional latency for waiting until dummy MVM has been completed
  - Avoid Zen 2 (can anybody test on Zen 1 and Zen 3?)

Matilda - a fast, low-jitter MVM library for AO RTC

Designed and optimized for repeated MVMs with constant matrix on many-core CPUs

rows x columns	MVM timing	CPUs	threads	remarks
1024 x 1024	**~4 µs**	AMD Epyc 7742	64	
	~6 µs	AMD Epyc 7742	16	
1280 x 3500	**~10 µs**	AMD Epyc 9654	32	
	~13 µs	AMD Epyc 7742	32	
2048 x 2048	**~9 µs**	AMD Epyc 7742	64	
	~10 µs	Intel Xeon Gold 6254	32	
	~25 µs	Intel Xeon E5-2698 v4	32	
3584 x 3456 (DKIST MCAO, modes to actuators)	**~15 µs**	AMD Epyc 9654	64	
	~24 µs	AMD Epyc 7742	64	
	~27 µs	Intel Xeon Gold 6254	32	F16C
	~67 µs	Intel Xeon E5-2698 v4	32	
	~107 µs	Intel Xeon Gold 6254	32	
4096 x 4096	**~17 µs**	AMD Epyc 9654	128	
	~32 µs	AMD Epyc 7742	64	
	~40 µs	Intel Xeon Gold 6254	32	F16C
	~95 µs	Intel Xeon E5-2698 v4	32	
	~200 µs	Intel Xeon Gold 6254	32	
4608 x 4096	~25 µs	AMD Epyc 7742	96	
	~38 µs	AMD Epyc 7742	64	Intel MKL
	~105 µs	AMD Epyc 7742	96	Intel MKL
8192 x 8192	~55 µs	AMD Epyc 9654	128	
12800 x 3500	**~24 µs**	AMD Epyc 9654	160	F16C
	~31 µs	AMD Epyc 9654	160	
19200 x 4092	**~31 µs**	AMD Epyc 9654	150	F16C
	~52 µs	AMD Epyc 9654	150	
33280 x 8192	**~105 µs**	AMD Epyc 9654	160	F16C
	~820 µs	AMD Epyc 9654	160	

- Matilda is not a general purpose BLAS gemv() interface
  MVM is usually memory bandwidth bound, but BLAS (=contiguous block of matrix data) is not bandwidth optimal on modern many-core CPUs with NUMA caches and RAM
- Matilda: First optimize data layout, then optimize the computation
  Create an “mvm_plan” (memory, threads) for one particular matrix in the beginning, then execute the plan for to compute the MVM.
- MVM plan is executed on dedicated, reserved CPU cores
  Matrix data cannot be evicted from caches
- Parallelized using its own thread pool designed for low latency/jitter
  - Threads are pinned to fixed CPUs and scheduling priorities can be elevated
  - Threads remain active when MVM has finished
- Explicit, hand-tuned vectorization via AVX2 and AVX-512 intrinsics
- Supports F16C 16-bit floating point format for matrix data storage to minimize cache use (MVM computed in 32-bit precision)

https://github.com/nsomatilda/Matilda

Matilda - a fast MVM library for AO RTC

Matrix data partitioning and layout

- Matrix data partitioned for threads
  - each thread gets a horizontal slice with multiples-of-8 rows
  - number depends on selected kernel and thread number
- Slice data transposed to column-major order
- Copied into core-local memory pages
  - aligned on cache line boundary
- Matrix data is not contiguous anymore, fundamental difference to BLAS.
  - High setup overhead upfront
  - More efficient memory access later