

THE COSMIC AO RTC PLATFORM

Damien Gratadour



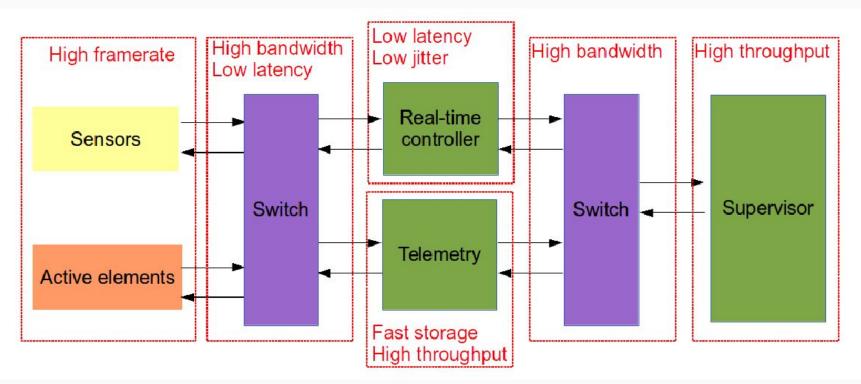
COSMIC PLATFORM



AO RTC GLOBAL SYSTEM ARCHITECTURE

Based on heterogeneous architecture to implement main functions

- Cope with various functional & non-functional requirements for the different sub-systems
- Mix high throughput Machine Learning (supervisor, a.k.a. SRTC) with low latency & low jitter computing (real-time controller, a.k.a. HRTC)



PROPOSED CONCEPT: SPARTA REVAMPED

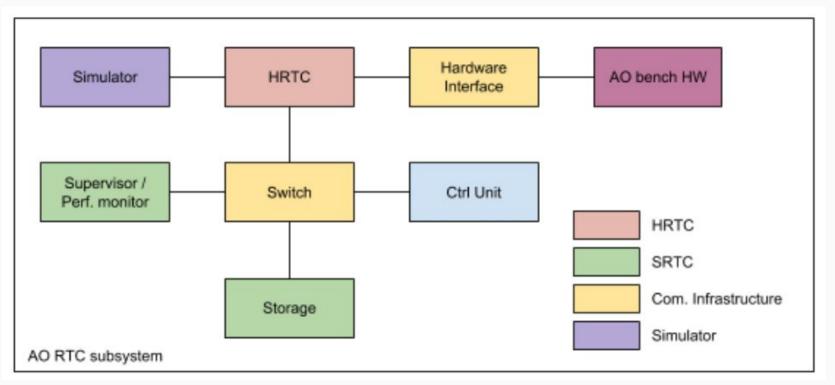




AO RTC DETAILED SYSTEM ARCHITECTURE

Inspired by SPARTA functional decomposition, inheriting from previous R&D

- Built to serve several AO instruments:
 - Keck Telescope: ongoing contract (first light yesterday !)
 - MICADO: ELT first light instrument (on-sky by 2025)
 - MAVIS: high complexity VLT instrument (on-sky by 2026)
 - 0





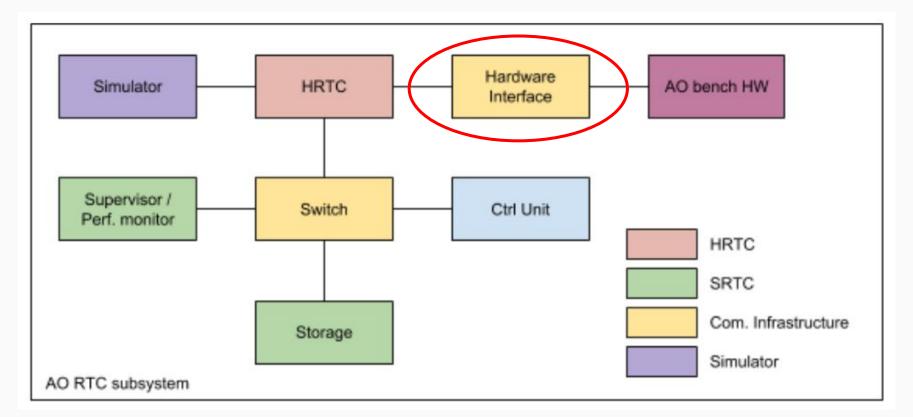
COSMIC platform



AO RTC DETAILED SYSTEM ARCHITECTURE

Inspired by SPARTA functional decomposition, inheriting from previous R&D

• Flexibility : dedicated HW interface



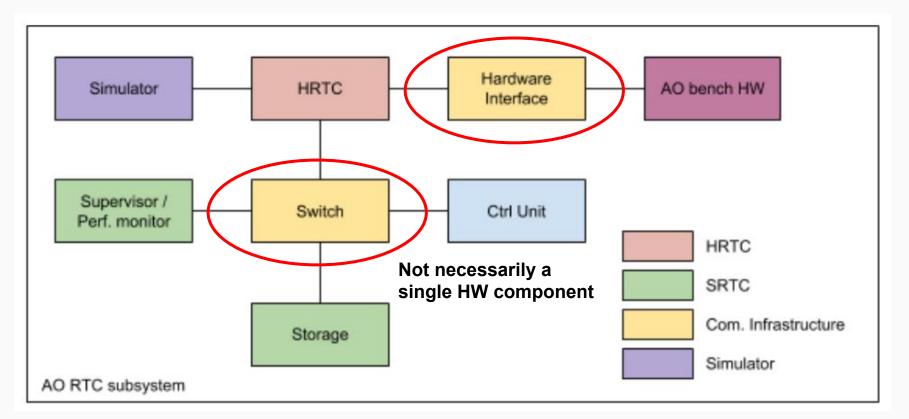




AO RTC DETAILED SYSTEM ARCHITECTURE

Inspired by SPARTA functional decomposition, inheriting from previous R&D

- Flexibility : dedicated HW interface
- SW based on abstraction layers





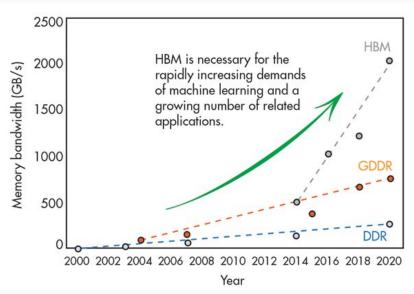
HARDWARE CHOICES

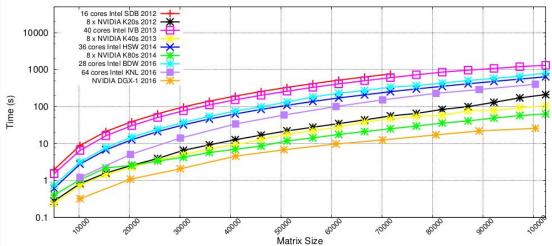
HRTC based on MVM: **memory bound**. GPU technology, equipped with HBM memory is the silver bullet ! (as compared to CPU with classical DDR)

SRTC based on embarrassingly parallel linear algebra: GPUs perform 10x better than CPU for this kind of workload, consistently over the past 6 years.

Several hardware design options:

- All-in-one HRTC+SRTC
- Separate servers in a cluster configuration







RTC HARDWARE DESIGN THE FUTURE IS NOW (AND ESO COMPLIANT !)

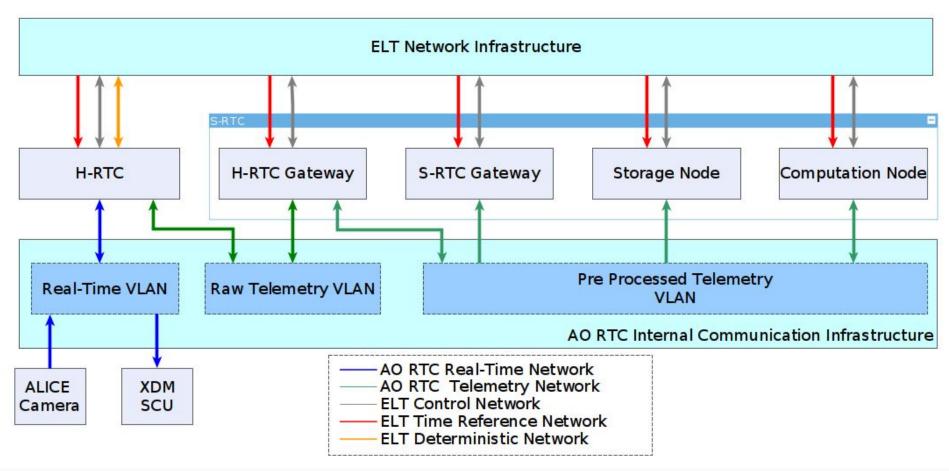




PROPOSED IMPLEMENTATION

Based on the ESO Standard RTC architecture for the ELT

- Proposed for MICADO and MAVIS
- Can be easily adapted to SPHERE+



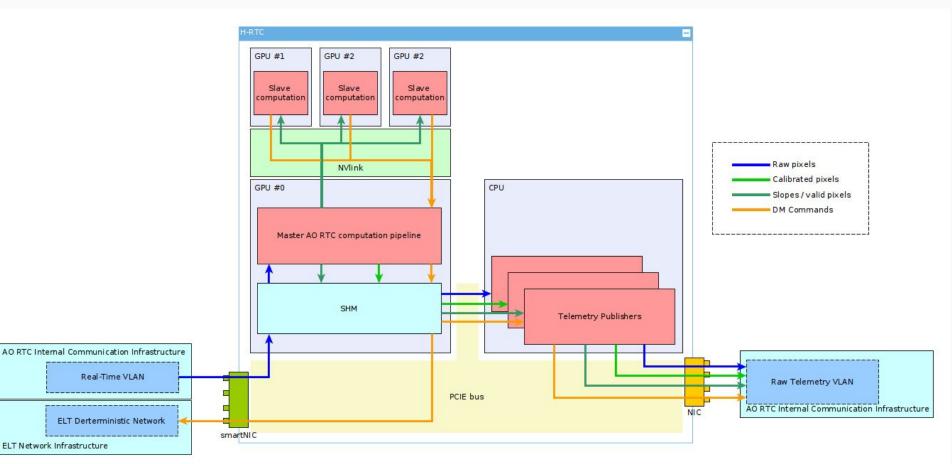




PROPOSED IMPLEMENTATION

Multi-GPU implementation of HRTC (dimensioning TBD for SPHERE+)

- MICADO & MAVIS design: leverage high density GPU cluster (PCIE + NVlink)
- Keck design: standard server with dual GPU (PCIe interface)







HRTC DATA STREAM INTERFACE

Based on µXlink board from Microgate

- Developed in the context of Green Flash, available as a COTS product
- Provide flexibility to any kind of hardware (WFS cameras, DMs, VLT / ELT frameworks) and efficient data transfer to/from compute units (GPUdirect)
- In production, used in ELT M4 design => long term availability to support ESO projects, already part of ESO ecosystem



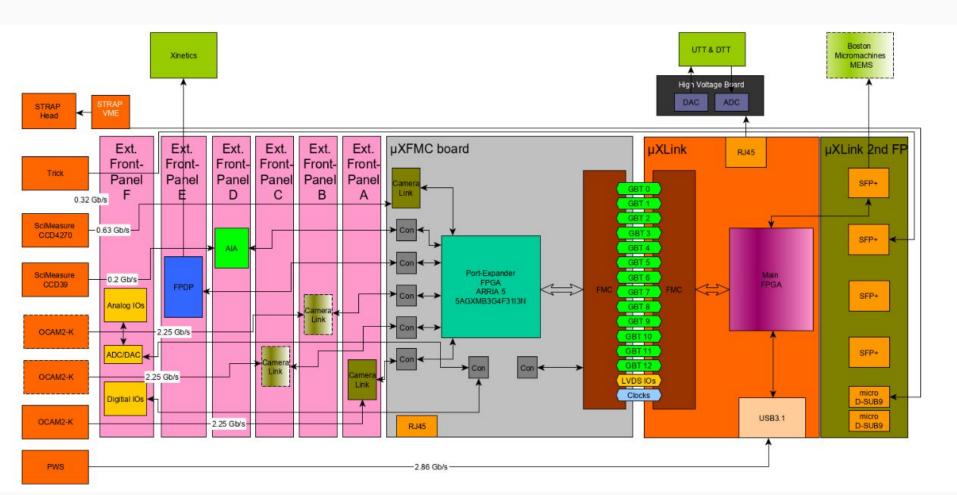




HRTC DATA STREAM INTERFACE

Currently under final integration phase for Keck

• Multiple interfaces: CameraLink, 10G Ethernet, USB, sFPDP





COSMIC PLATFORM



HRTC DATA STREAM INTERFACE

Currently under final integration phase for Keck

• Multiple interfaces: CameraLink, 10G Ethernet, USB, sFPDP







HRTC DATA STREAM INTERFACE PERFORMANCE

Testing the data acquisition interface to the GPU:

- direct memory access (relying on standard development tools and factory versions of drivers / API)
- Here compared to "classical" CPU memory access

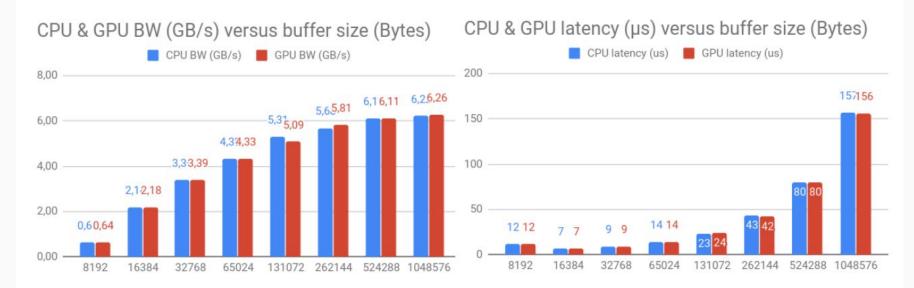


Figure 4-56 - Bandwidth and latency measurement of DMA operation between the μ Xcomp board and CPU (respectively GPU) with respect to buffer size

RTC Software design: Modular, comprehensive and powerful

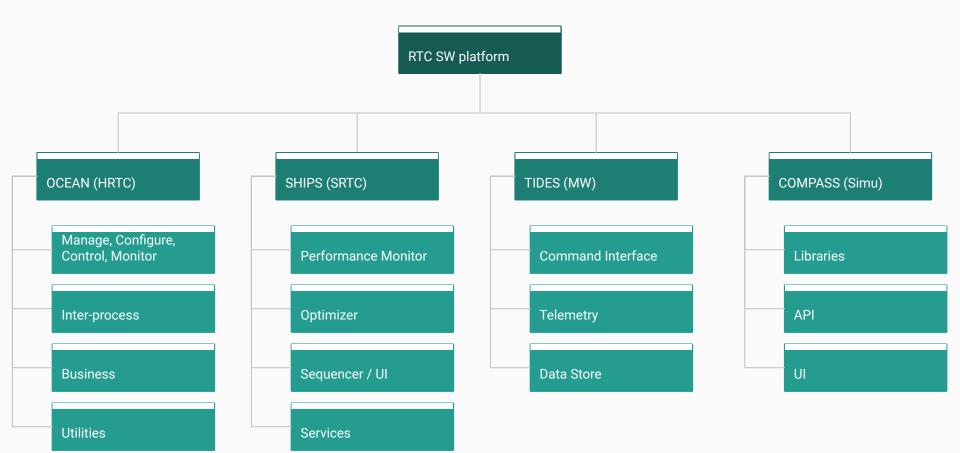




AO RTC SOFTWARE COMPONENTS

4 components:

• OCEAN, SHIPS, TIDES and COMPASS



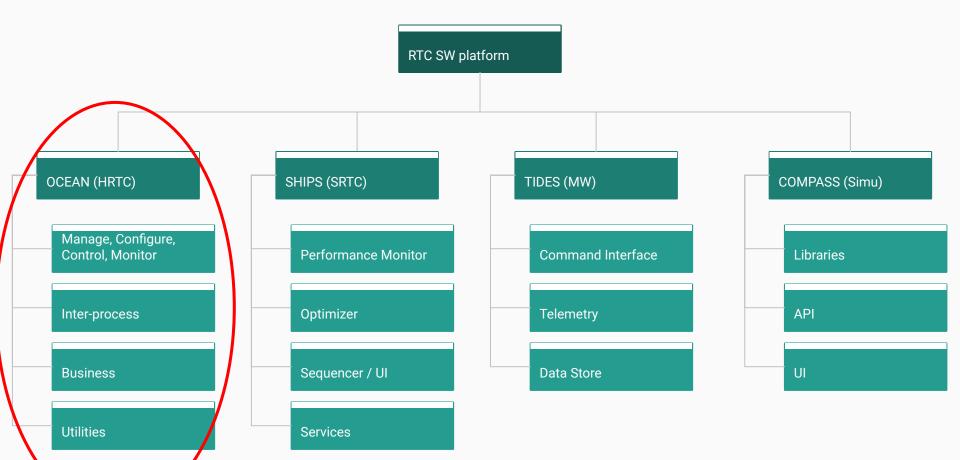




AO RTC SOFTWARE COMPONENTS

4 components:

• OCEAN, SHIPS, TIDES and COMPASS







HRTC SW DESIGN

Flexible implementation in heterogeneous environment through abstraction layers

OCEAN (Optimized Core Engine for Adaptive optics pipeliNes)

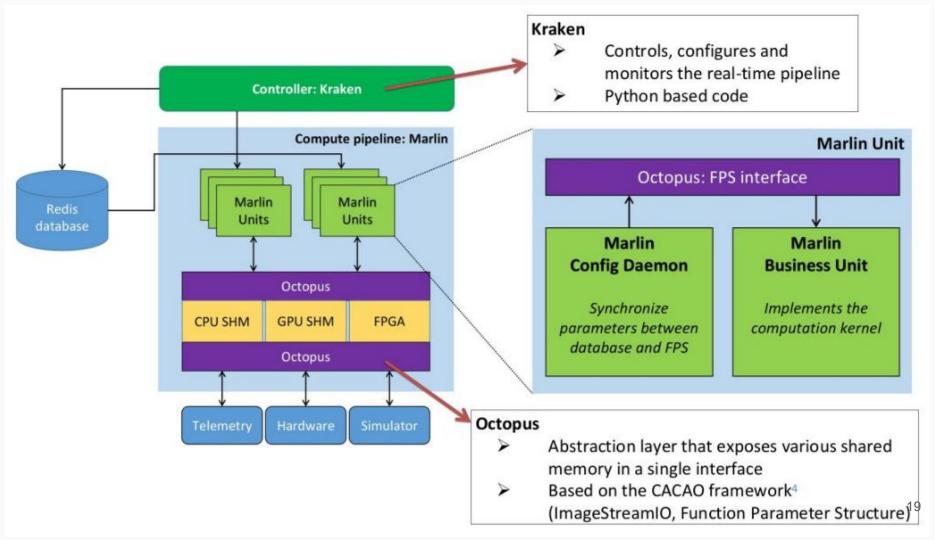
- Kraken: Python based control, management and monitoring:
- Marlin: <u>C++ library</u> for real-time kernels execution
- **Octopus**: <u>C++ library</u> for data interface:
- Other components include: **Seahorse, Wyrm, Fish** (utilities)





HRTC SW DESIGN

Core pipeline, made highly modular through the Marlin library



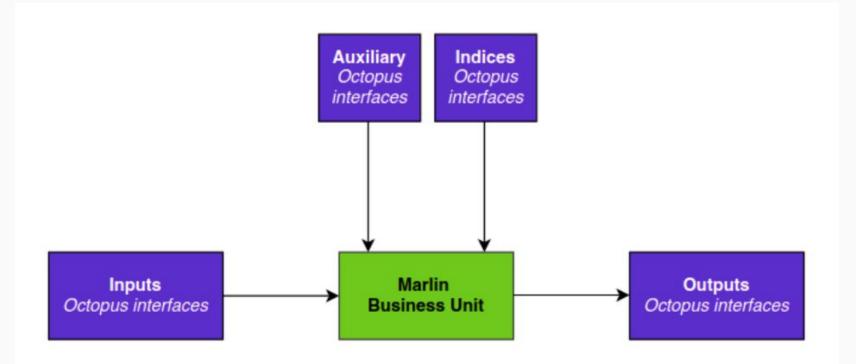




HRTC SW DESIGN

Main features of Marlin library:

- Moved away from persistent kernel solutions:
 - Better flexibility
 - Want to rely as much as possible on standard libraries
- Retained one concept: "active wait" for kernels sync
- Added support for semaphore based sync



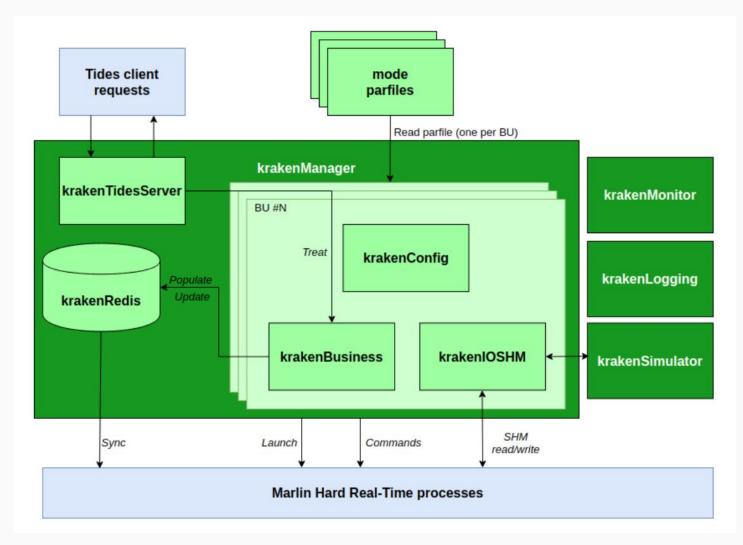


COSMIC PLATFORM



HRTC SW DESIGN

Kraken Software architecture







HRTC SW control / monitoring interface

Python + Qt graphical interface

- Based on backend (running on RTC) publishing on a dedicated socket + frontend (running anywhere)
- Start / Stop / Pause / Step / Kill Processes
- Access to logs

lan	ne	Status	Control	CPU	Memory	Loop count
•	CalPixTelemetry (2)	ALIVE	RUNNING	0.1%	188.3 MiB	27158
*	CoGCentroider (2)	ALIVE	RUNNING	0.1%	1168.1 MiB	27158
•	ControllerGeneric (2)	ALIVE	RUNNING	0.0%	1486.8 MiB	27158
•	LoopDataTelemetry (2)	ALIVE	RUNNING	0.0%	188.1 MiB	27158
•	RawPixTelemetry (2)	ALIVE	RUNNING	0.1%	187.7 MiB	27158
•	evtcam (2)	ALIVE	RUNNING	1.0%	179.3 MiB	27158
•	kacou (2)	ALIVE	RUNNING	0.1%	170.3 MiB	27158

STEP

PAUSE

STOP

RUN

KILL



COSMIC PLATFORM



HRTC SW control / monitoring interface

Python + Qt graphical interface

- Based on backend (running on RTC) publishing on a dedicated socket + frontend (running anywhere)
- Detailed view

	PROCESSES		27			DETAILS						SHARE	E .		
Name	PID	Status	CPU	Memory	Loop	Contr	Tmux session	Loop S	tatu	Creation time (UTC)	CPU affinity	Context sw	Thre	Message	
MarlinBU_CalPixTelemetry	614105	ALIVE	00	181,784 KiB	30419	0	krakenBusiness_CalPixTelem	1	0	2020-10-12T07:57:	16,17	144118	12	4 frame(s) lost, reset loop	
MarlinBU_CoGCentroider	613372	ALIVE	00	1,185,184	30419	0	krakenBusiness_CoGCentroid	1	0	2020-10-12T07:57:	9,10	314223	4	4 frame(s) lost, reset loop	
MarlinBU_ControllerGeneric	613548	ALIVE	00	1,511,384	30419	0	krakenBusiness_ControllerGe	1	0	2020-10-12T07:57:	10,11	240623	6	4 frame(s) lost, reset loop	
MarlinBU_LoopDataTelemetry	613913	ALIVE	00	181,660 KiB	30419	0	krakenBusiness_LoopDataTel	1	0	2020-10-12T07:57:	13,14	192888	12	Slopes and commands ar	
MarlinBU_RawPixTelemetry	614300	ALIVE	00	181,324 KiB	30419	0	krakenBusiness_RawPixTele	1	0	2020-10-12T07:57:	16,15	94766	12	4 frame(s) lost, reset loop	
MarlinBU_evtcam	614498	ALIVE	00	172,516 KiB	30419	0	krakenBusiness_evtcam	1	0	2020-10-12T07:57:	8,5,6,7	35320	6	4 frame(s) lost, reset loop	
MarlinBU_kacou	613730	ALIVE	00	163,324 KiB	30419	0	krakenBusiness_kacou	1	0	2020-10-12T07:57:	12,13	244328	4	4 frame(s) lost, reset loop	
MarlinConfigDaemon_CalPixT	614093	ALIVE	00	11,060 KiB	1	1	krakenBusiness_CalPixTelem	0	0	2020-10-12T07:57:	0,1,2,3,4,18,19,20,21,22,2	6148803	3	Running	
MarlinConfigDaemon_CoGCe	613360	ALIVE	00	10,996 KiB	1	1	krakenBusiness_CoGCentroid	0	0	2020-10-12T07:57:	0,1,2,3,4,18,19,20,21,22,2	6252265	3	Running	
MarlinConfigDaemon_Contro	613536	ALIVE	00	11,068 KiB	1	1	krakenBusiness_ControllerGe	0	0	2020-10-12T07:57:	0,1,2,3,4,18,19,20,21,22,2	6226754	3	Running	
MarlinConfigDaemon_LoopD	613901	ALIVE	00	11,048 KiB	1	1	krakenBusiness_LoopDataTel	0	0	2020-10-12T07:57:	0,1,2,3,4,18,19,20,21,22,2	6175046	3	Running	
MarlinConfigDaemon_RawPix	614288	ALIVE	00	10,928 KiB	1	1	krakenBusiness_RawPixTele	0	0	2020-10-12T07:57:	0,1,2,3,4,18,19,20,21,22,2	6122675	3	Running	
MarlinConfigDaemon_evtcam	614486	ALIVE	00	11,072 KiB	1	1	krakenBusiness_evtcam	0	0	2020-10-12T07:57:	0,1,2,3,4,18,19,20,21,22,2	6096047	3	Running	
MarlinConfigDaemon_kacou	613718	ALIVE	00	11,048 KiB	1	1	krakenBusiness kacou	0	0	2020-10-12T07:57:	0,1,2,3,4,18,19,20,21,22,2	6195374	3	Running	

STOP

HRTC PERFORMANCE COSMIC IS ALREADY THERE.





HRTC SW PERFORMANCE

- WFS frames are sent by a hardware emulator at a regular rate (1 kHz)
- GPU is mostly "active waiting"

	35525 ms 35525.1 ms 35525.2 ms 35525	5.3 ms 35525.4 ms 35525.5 ms 3 <mark>1.08441 ms</mark> 35525.7 ms 35525.8 ms 35525.9 ms 355	26 ms 35526.1 ms 35526.2 ms 35526.3 ms 35526.4 ms 35526
Process "marlinMaintypes=f			
E Thread 345784320			
Runtime API			
L Driver API			
Profiling Overhead			
E Process "marlinMaintypes=fff			
Thread 4170813440			
Runtime API		cudaEventSynchronize	
L Driver API			
Profiling Overhead			
Process "marlinMaintypes=fff			
Thread 1039544320			
Runtime API		cudaGr cudaEventSynchronize	cudaGr
L Driver API			
Profiling Overhead			
😑 [0] Tesla V100-SXM2-16GB			
Context MPS (CUDA)			
🗏 🍸 MemCpy (HtoD)		—— Incoming frame ———	
ြ 🍸 MemCpy (PtoP)		5	
1 Compute	void gemvNSP	void ipc::active_wait_multi_kernel <float>(float volatile *, int) void ipc::active_wait_multi_kernel<float>(float volatile *, int) Single RTC</float></float>	void gemvNSP
Streams		Single RTC	
[1] Tesla V100-SXM2-16GB			
Context MPS (CUDA)		pipeline iteration	
- 🍸 MemCpy (HtoD)			
Compute	void gemvNSP		void gemvNSP
Streams			
[2] Tesla V100-SXM2-16GB			
Context MPS (CUDA)		<hr/>	
🗏 🍸 MemCpy (HtoD)		•	
🛨 Compute	void gemvNSP	A ative \A/ait	void gemvNSP
 Streams 	Compute	Active Wait	
[3] Tesla V100-SXM2-16GB			
Context MPS (CUDA)	-		
🗆 🍸 MemCpy (HtoD)			and the second
🛨 Compute	void gemvNSP		void gemvNSP





HRTC SW performance

- WFS frames are sent by a hardware emulator at a regular rate (1 kHz)
- GPU is mostly "active waiting"

	5 ms 35525,05 ms 35525,075 ms <u>35525,175 ms 35525,125 ms 35525,15 ms 35525,15 ms 35525,2 ms 35525,225 ms 35525,25 ms 35525,275 ms</u> 35525,3 ms 35525,325 ms 3
E Thread 345784320	
L Runtime API	cudaMemcpyAsync cudaEventSync
🕒 Driver API	
Profiling Overhead	
Process "marlinMaintypes=fff	
E Thread 4170813440	
L Runtime API	cudaGrap
L Driver API	
Profiling Overhead	
Process "marlinMaintypes=fff	
🖃 Thread 1039544320	
- Runtime API	cudaGraphLaunci
L Driver API	
Profiling Overhead	
[0] Tesla V100-SXM2-16GB	
Context MPS (CUDA)	
ြ 🍸 MemCpy (HtoD)	Memcpy HtoD
ြ 🍸 MemCpy (PtoP)	
🗄 Compute	vo i gemvNSP_kernel <float, cublasge<="" float,="" int="1024," td=""></float,>
🗄 Streams	
[1] Tesla V100-SXM2-16GB	
Context MPS (CUDA)	Frame readined:
🍯 🐨 MemCpy (HtoD)	Frame received:
🛨 Compute	void gemvNSP_kernel <float, cublasge<="" float,="" int="1024," td=""></float,>
🛨 Streams	 End of active wait
🖃 [2] Tesla V100-SXM2-16GB	
Context MPS (CUDA)	 Execute centroiding sequence
ြ 🍸 MemCpy (HtoD)	
E Compute	void gemvNSP_kernel <float, cublasge<="" float,="" int="1024," td=""></float,>
Streams	
[3] Tesla V100-SXM2-16GB	
Context MPS (CUDA)	
🗆 🍸 MemCpy (HtoD)	
🗄 Compute	void gemvNSP_kernel <float, cublasge<="" float,="" int="1024," td=""></float,>
🛨 Streams	





HRTC SW performance

- WFS frames are sent by a hardware emulator at a regular rate (1 kHz)
- GPU is mostly "active waiting"

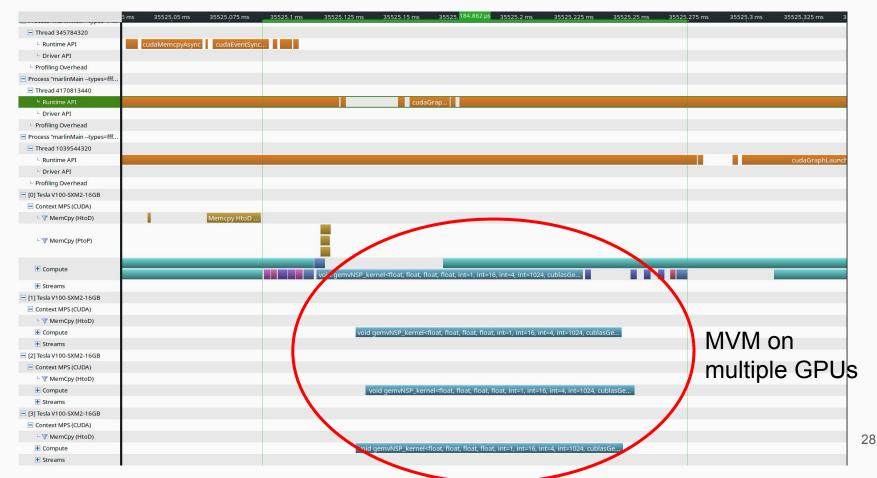
	5 ms 35525,05 ms	35525.075 ms	35525.1 ms	35525.125 ms	35525.15 ms	35525. <mark>184.862 µs</mark> 355	525.2 ms 🛛	35525.225 ms	35525.25 ms	35525.275 ms	35525.3 ms	35525.325 ms 3
Thread 345784320												
- Runtime API	cudaMemcpyAsync	cudaEventSync										
L Driver API												
Profiling Overhead												
Process "marlinMaintypes=fff												
Thread 4170813440												
► Runtime API					cuda	aGrap						
L Driver API												
Profiling Overhead												
Process "marlinMaintypes=fff												
🖃 Thread 1039544320												
- Runtime API												cudaGraphLauncl
L Driver API												
Profiling Overhead												
[0] Tesla V100-SXM2-16GB				_								
Context MPS (CUDA)				H	sroado	cast dat	a to					
🗆 🍸 MemCpy (HtoD)		Memcpy HtoD		-								
L 🍸 MemCpy (PtoP)			(n	nultipl	le GPUs	5					
🗄 Compute				void gemvNSP_k	ernel <float, fl<="" float,="" td=""><td>float, float, int=1, int=16, int=</td><td>=4, int=1024, cul</td><td>blasGe</td><td></td><td></td><td></td><td></td></float,>	float, float, int=1, int=16, int=	=4, int=1024, cul	blasGe				
🗄 Streams												
🖃 [1] Tesla V100-SXM2-16GB												
Context MPS (CUDA)												
🗆 🍸 MemCpy (HtoD)												
💽 Compute				void	gemvNSP_kernel<	<float, float,="" int="</td"><td>1, int=16, int=4,</td><td>int=1024, cublas</td><td>5e</td><td></td><td></td><td></td></float,>	1, int=16, int=4,	int=1024, cublas	5e			
🛨 Streams												
🖃 [2] Tesla V100-SXM2-16GB												
Context MPS (CUDA)												
🗆 🍸 MemCpy (HtoD)												
E Compute					void gemvNSP_ker	rnel <float, float,="" float,<="" td=""><td>int=1, int=16, in</td><td>t=4, int=1024, cub</td><td>lasGe</td><td></td><td></td><td></td></float,>	int=1, int=16, in	t=4, int=1024, cub	lasGe			
🛨 Streams												
[3] Tesla V100-SXM2-16GB												
Context MPS (CUDA)												
🗏 🍸 MemCpy (HtoD)												
🗄 Compute				void	l gemvNSP_kernel	<float, float,="" int="</td"><td>1, int=16, int=4,</td><td>int=1024, cublast</td><td>5e</td><td></td><td></td><td></td></float,>	1, int=16, int=4,	int=1024, cublast	5e			
+ Streams												





HRTC SW PERFORMANCE

- WFS frames are sent by a hardware emulator at a regular rate (1 kHz)
- GPU is mostly "active waiting"



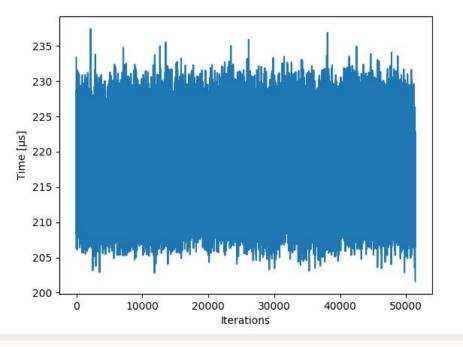


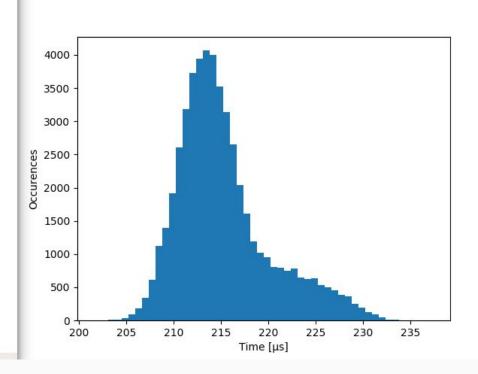


HRTC SW performance

Initial performance analysis : MAVIS pipeline (6+3 WFS, 5k x 20k command matrix) on 4 GPUs (NVIDIA V100)

- Mean time-to-solution (pure RTC latency) : ~215µs
- P2V jitter: ~40µs
- RMS jitter: ~5µs





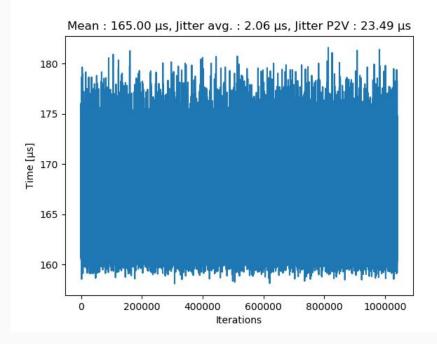


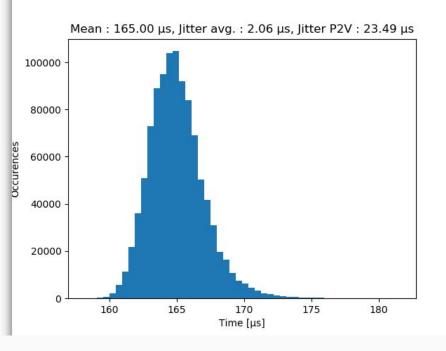


HRTC SW performance

Current performance figures : MICADO pipeline (1 Pyr WFS, 5k x 25k command matrix) on 4 GPUs (NVIDIA V100)

- Mean time-to-solution (pure RTC latency) : ~165µs
- P2V jitter: ~23µs
- RMS jitter: ~2µs







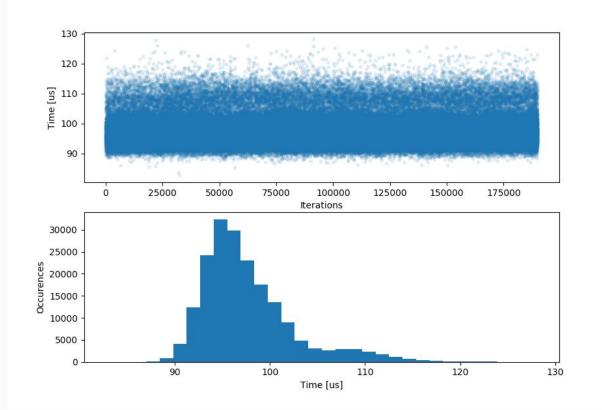
COSMIC PLATFORM



HRTC SW performance

End-to-end performance (i.e. with hardware interface) : Keck pipeline (1 SH WFS, ~350 x 600 command matrix, **11 BUs sequence**) on 1 GPU (NVIDIA V100)

• Mean time-to-solution (pure RTC latency) : ~105µs



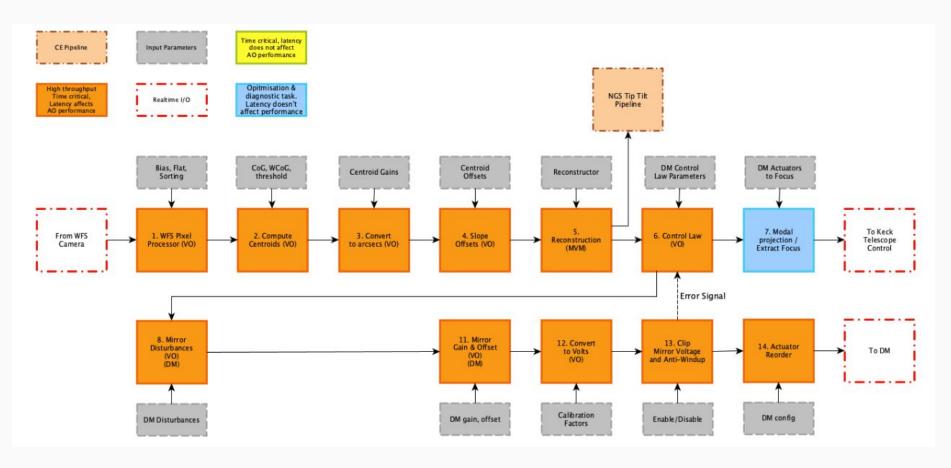




HRTC SW PERFORMANCE

End-to-end performance (i.e. with hardware interface) : Keck pipeline (1 SH WFS, ~350 x 600 command matrix, **11 BUs sequence**) on 1 GPU (NVIDIA V100)

• Mean time-to-solution (pure RTC latency) : ~105µs



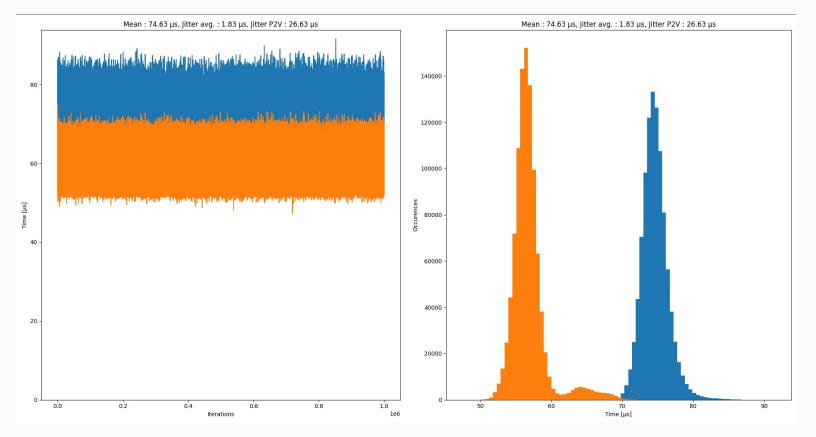




HRTC SW performance

End-to-end performance (i.e. with hardware interface) : Keck pipeline (1 SH WFS, ~350 x 600 command matrix, **11 BUs sequence**) on 1 GPU (NVIDIA V100)

• Already supporting two stages at different framerates (orange: TT stage, blue HO stage)



SRTC SOFTWARE A FEATURE-RICH STACK FITTING ALL AO FLAVORS NEEDS

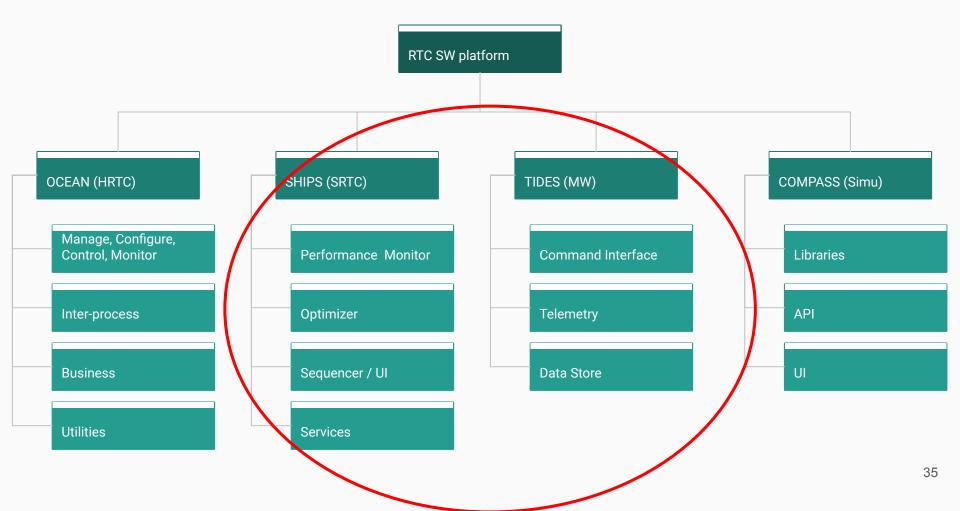




AO RTC SOFTWARE COMPONENTS

4 components:

• OCEAN, SHIPS, TIDES and COMPASS





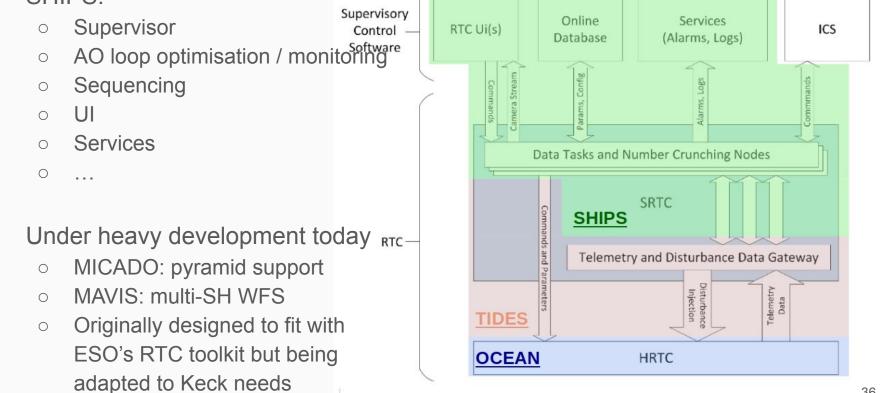
COSMIC PLATFORM



SRTC COMPATIBILITY WITH RTC TOOLKIT

Two components in which RTC Tk will fit:

- TIDES:
 - **control / command**: support for ZeroMQ, Pyro, others (TBD) Ο
 - data distribution: support DDS, MUDPI, custom protocol, others (TBD) Ο
- SHIPS:

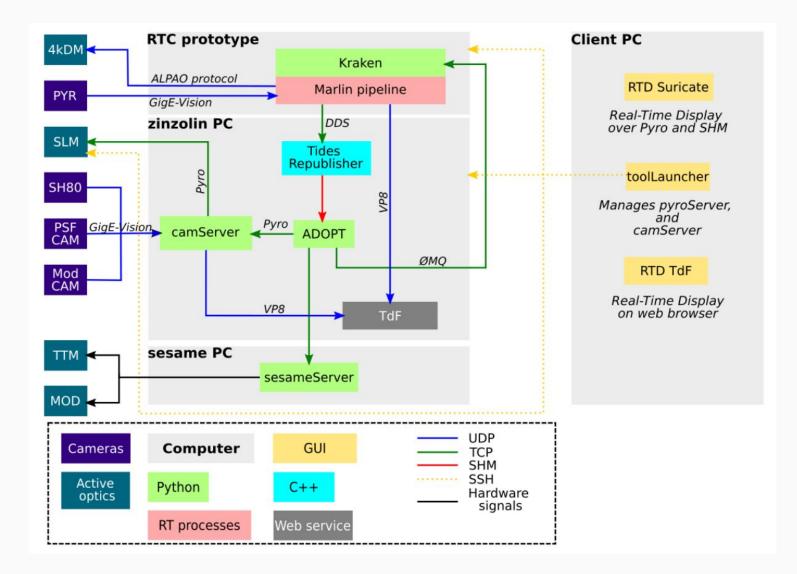


COSMIC IN ACTION





AO BENCH EXPERIMENT @ LESIA

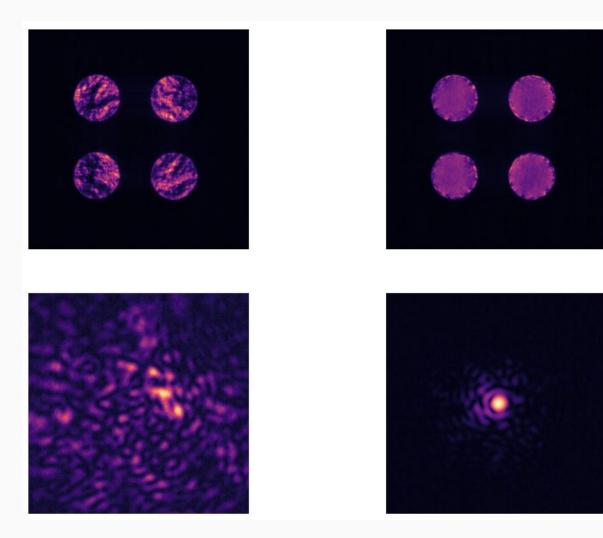






AO BENCH EXPERIMENT @ LESIA

Open / Closed loop operation with pyramid WFS (resp. Left and Right)

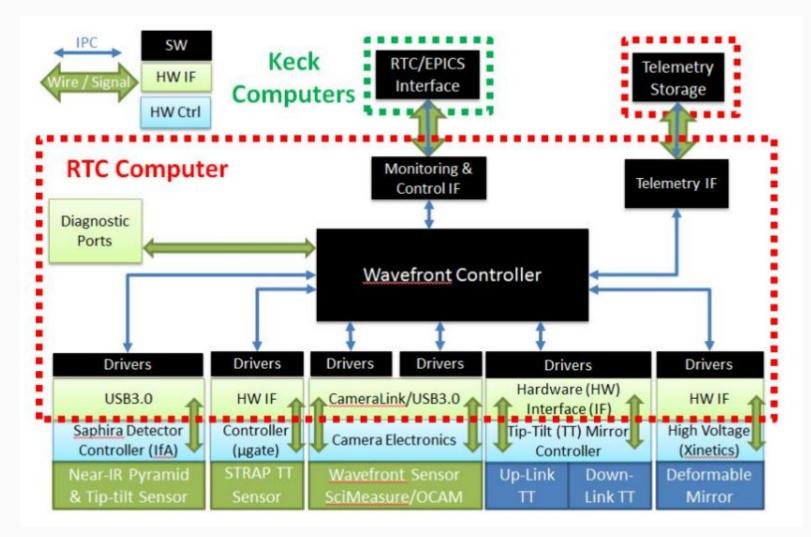






COSMIC @ KECK

Ongoing commissioning phase @ Keck







COSMIC @ KECK

Ongoing commissioning phase @ Keck

- 2 instances: Keck I and Keck II
- Multiple modes supported: SH / pyr / LGS / LTAO
- First-light last week. Should be offered to the community by end of June

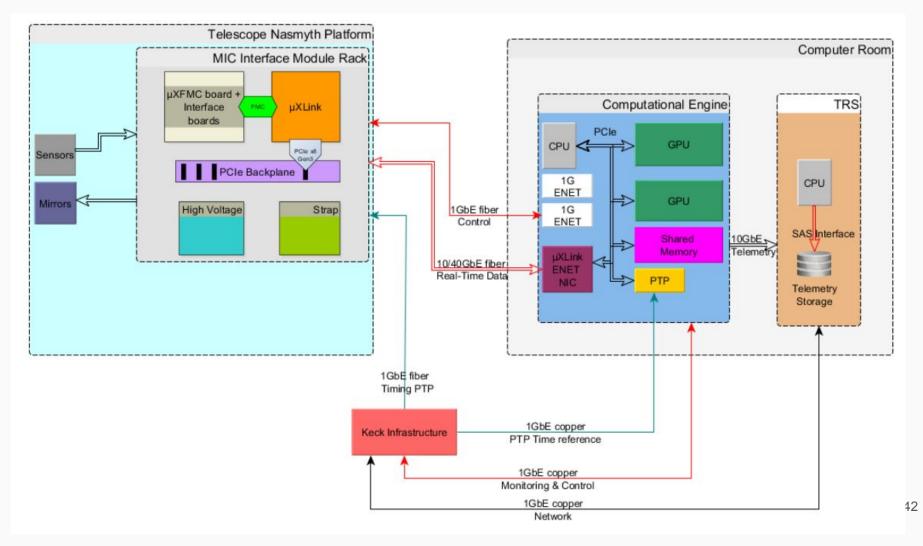
Mode	Name	SH WFS		PWS	Tip-tilt (+ focus)			DM		Tip-tilt	
#	Name	SciMeasure	OCAM2K	PWS	STRAP	TRICK	PWS	Xinetics	MEMS	ΤТМ	UTT
1	NGS w/ Sci Measure	1						1		1	
2	NGS w/ OCAM2K		1					1		1	
3	NGS w/ PWS			1				1		1	
4	NGS w/ PWS & MEMS			1					1	1	1
5	LGS w/ SciMeasure & STRAP+TRICK	1			1	1		1		1	1
6	LGS w/ OCAM2K & STRAP+TRICK		1		1	1		1		1	1
7	LGS w/ OCAM2K & PWS		1	1				1		1	1
8	LGS w/ OCAM2K & PWS TT		1				1	1		1	1
9	LTAO w/ STRAP+TRICK		1		1	1		1		1	3
10	LTAO w/ PWS TT		1				1	1		1	3
11	LTAO w/ STRAP+TRICK & new DM		3		1	1		new		1	3





COSMIC @ KECK

Ongoing commissioning phase @ Keck







COSMIC @ KECK

HO and TT loops closed @ > 1 kHz

🖲 💿 🤯 🥰 You are viewing Sam Ra	agland's screen View Options	× .							
	We vm-psfr:1 (vm-psfr:1 (sragland)) - VNC Viewe	r.							
rile Home send / Neverve Founds avia	-osiris:1 (osiris-osrseng-control0) - VNC Viewer Iobsao's k1ao-k1obsao-control0 desktop (kaalua)	1. W. Mar Window							
	k1ao Desktop	More about W	(AL 04/61D) (01		-				
New New Schedule a Nighting	kiao Besktop		IDOS						
Email Ite View klaoserver-new:2 (klao-klobsao-controll) - VNC View	iewer							and the second se	
Keck I WFS Configuration	×		Inde		k1obTop.ui			7	MAORI (v9.0.0) : Keck 1
PROGRAM (0-7) 6 8						. PM/		Menu Al TARGET	Control Tel WFS WFC Track DAR Exit 2607-0232-1 WAVEFRONT SENSOR
REPETITIONS (0-65535) 60 (60	Testal -		Wavefro Sensor Pa	th Path	Tilt/Acq Path	tics Cont	rol		6 200 Frame rate 650 B00
	WFS Stat		SETNO WLS	ROT	TSS SND			2 RECC 2 RECC	h8//ex
0: M2 basi	X MAKEBKGND: Background make failed!	ue	FSM	IDC1	AFS SFD	PMAC		Z RECO LOOP Z RECO Z RECO TT	OPEN CLOSE GAN
Ref = FRAME RATE (0-2000) 650 50 ad	X TAKE_WFS_BKG: Noving back to target X TAKE_WFS_BKG: Waiting for telescope to tr X TAKE_WFS_BKG: Waiting for telescope to tr	ack ack	FCS	ISM	AFM SFP	PMAC		Z Coxe Z Coxe Z Coxe DM	OPEN CLOSE 0.200 0.2
	X THKE_WFS_BKG: Waiting for telescope to tr X THKE_WFS_BKG: Nove complete	ack	FSS	IDCS	LBS DFM			2 Coxpl 2 Coxpled nodul	e: LOADAR.
000	X THVE_WF5_BNG; LIL> fsmofFload X Competing mathims SEMMERIOND		wcs	WFS In	TTD SPM	PMAC	-4 06 Apr	112 X LOADMR: Loadin X LOADMR: Recons	g reconstructor tructor file load successful
CALCULATE SET DISHISS	2 Compiled module: FSMOFFLOAD. 2 Compiled module: WAIT_FOR_COMPLETE. 2 WAIT_FOR_COMPLETE: Waiting for device for 2 WAIT_FOR_COMPLETE: Waiting for device for	to complete move.	80: NOS	FSH: INPOS		and SKY 30°E	er Te	00 10L> [] mis	
2 NEKEDKUNUS AO Mode NGS-AO	X WAIT_FOR_COMPLETE: ., hang on I X WAIT_FOR_COMPLETE: fsm move complete		Rnag; 4,			cord blogd only		nit Halt Stby	More
Z KRKEEKOND:	IIL> % TAKE_WFS_BMG: Recording UFS background 30 % TAKE_WFS_BMG: Waiting for telescope to ta	arcsecond East from the target	Cx: -0.01	2 Cy: -0.094				nit Halt	Tip Tilt Graphs
Z NAKETKCAD:	% TAKE_WFS_BAG: Waiting for telescope to to % TAKE_WFS_BAG: Waiting for telescope to to % TAKE_WFS_BAG: Waiting for telescope to to	ack	-		Rep	on, mask is OFF		nit Halt	Residual Error Mirror 1 (arcsec) (arc
Z MAKEIMGRO: Z MAKEIMGRO: X MAKEIMGRO:	X THOE_WFS_BKG: Nove complete X MHKEBKGND: Record, write, load background		2000 2000		Sec. of Sec.		X	nit Halt	(acces) 1.0 (ac
Z TAKE_UFS_E Ref: 3,30,-13,91 record	X MAKEBKGND: Background nake NDT complete! X MAKEBKGND: Background nake NDT complete! X MAKEBKGND: Background nake NDT complete!		0000	ec sample				nit Halt	
2 TAKE_NFS_I FSM: 3,30,-13.91 to ref	0	1002004.00						nit Halt	· • • • • •
Desk: 0 8 Z TAKE_UPS_E Z SI	ETUPLE DTCL x (mm)	0.00			Dismiss				. / .
0. WFS 8 2 C	DTCL y (mm)	0.00	the second se	y centroid (mm				nit Halt (
2. k1ob	TTO x (mm)	0.00		seqnum	78			nit nait a	-0.3 x-filt 0.3 -1.0 x-f
9. WFS Backgr	TTO y (mm)	0.00		cycle	20			nit Halt Suy	
5. osrse UFS Futo	TTO loop state	closed open	closed	enable delay (s		1.0	S	nit Halt Stby	More
7. ktao Acquisition Loops	TTO loop enabled	false		gain	0.6	0.6	S	nit Halt Stby	
8 osrseSetup benchOpen	TTO loop paused	false		period (s)	3.0	3.0	<u>s</u>	nit Halt Star	screen
Roquire star Reclose	TTO error (mm)	13.91		ethresh (mm)	0.0	0.0	S		9.py*, line 30, in <nodule></nodule>
Offset to target	TTO oneshot	Send	ПС	tthresh (s)	2.0	2.0	S		0tsize*sizel-bg[0tsize*size]).reshape(s ted operand type(s) for *t 'NoneType' s
	Comment:	Comment:							caneraqui.csh
02:58:09 02:58:09 >> Bench setup IONE <<		Save		Dismiss				1.0	S rtcTestCCU39.py caneragu1.csh S re/kicbsao/scetre/testRTC}135;
02:58:09 Loading cog: 2405105-N 02:58:09 Loading cog: 2405105-N 02:58:09 FES CO setup For: 05181	1x1.cog	O loop state	closed	open close	WFO period (s)	2	14.0	14.0	S e/k1cbsao/scetre/testRTC)135; e/k1cbsao/scetre/testRTC)135;
02:58:09 02:58:08 Setting STRAP centroid s	gain to 1,000	D loop enabled	false		WFO ethresh (u		5.0	5.0	S e/klobsao/scetre/testRTC)135; e/klobsao/scetre/testRTC)135;
02:58:08 Setting centroid gain to 02:58:08 Updating centroid gain 02:58:03 Setting up the lenslet o	0 0,540 NE	O error (um)	0.01		WFO tthresh (s)	20.0	20.0	<pre>set/kicbsao/scetre/testRTC}135; set/kicbsao/scetre/testRTC}135; set/kicbsao/scetre/testRTC}135;</pre>
02:55755 Setting up the lensiet of 02:57555 Setting UFS for effectin 02:57556 Waiting for fam		O oneshot	5	end	WFO SFP Z fac	tor	-72.00	-72.00	S e/kicbsao/scetre/testRTC)135; e/kicbsao/scetre/testRTC)135; caneragui
	M D	nment:							agui.pro e/kiobsao/scetre/testRTC}135; cameragui
DISKISS			Save	Dismi			smiss		<pre>ie/kiobsao/scetre/testRTC}136: cameragui ie/kiobsao/scetre/testRTC}137: cameragui</pre>
		ocalestick1obsao@k1	FST caQt	DN FCaQIDN FCaQI	DM caQtDM caQtDM	CaQIDN			





Non disruptive integration

Current experience with Keck:

- AO system must remain operational throughout the integration of new RTC
- RTC system being integrated over 3 continents: Italy, Australia, Hawaii
- Incremental integration
 - Validate hardware interfaces (with hardware simulators / emulators): Italy
 - Validate compute engine (with a single simulator): Australia
 - Integrate both: 2 instances in both Italy and Australia
 - Ship a third instance to Hawaii
 - Fully integrate 3rd instance in Hawaii during daytime
 - Validate interfaces with hardware during daytime
 - Close loop with hardware during daytime
 - First light
- Total amount of nighttime lost so far: 0.25 night (issues with pipeline)
- Total amount of engineering nights scheduled: 1.5 nights (3 half nights)

Similar strategy with other instruments upgrade

• Need to secure hardware simulators / emulators for the first stage

COSMIC: READY FOR NEXT LEVEL





WHAT'S NEXT?

GPUs have enabled many progresses in deep learning in recent years **To date the best hardware platform to run DL workloads**

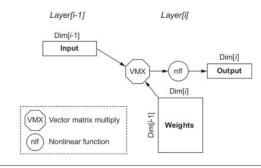
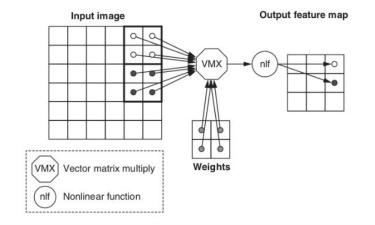


Figure 7.7 MLP showing the input Layer[*i*-1] on the left and the output Layer[*i*] on the right. ReLU is a popular nonlinear function for MLPs. The dimensions of the input



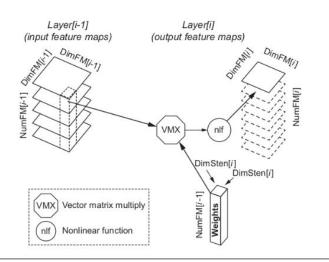


Figure 7.9 CNN general step showing input feature maps of Layer[i-1] on the left, the output feature maps of Layer[i] on the right, and a three-dimensional stencil over input feature maps to produce a single output feature map. Each output feature map



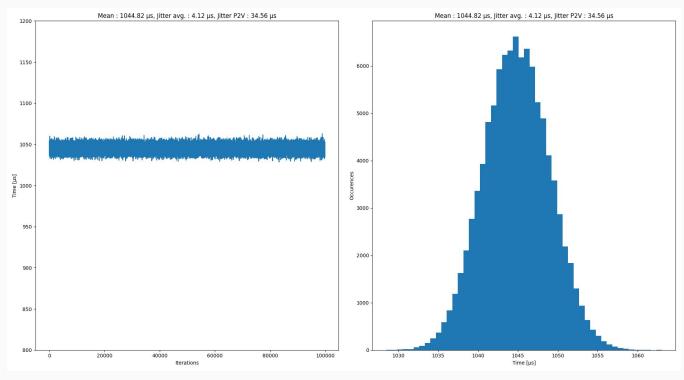




WHAT'S NEXT?

COSMIC RTC platform:

- Delivers performance for model-driven ML approaches for AO: example of MAORY HRTC below (time-to-solution: 1ms on 4 V100 GPUs)
- Is ready (by design) to run DL workloads



• Collab. with Stephen Jones, NVIDIA on new programming models





SUMMARY

COSMIC is a proven RTC solution for facility instruments

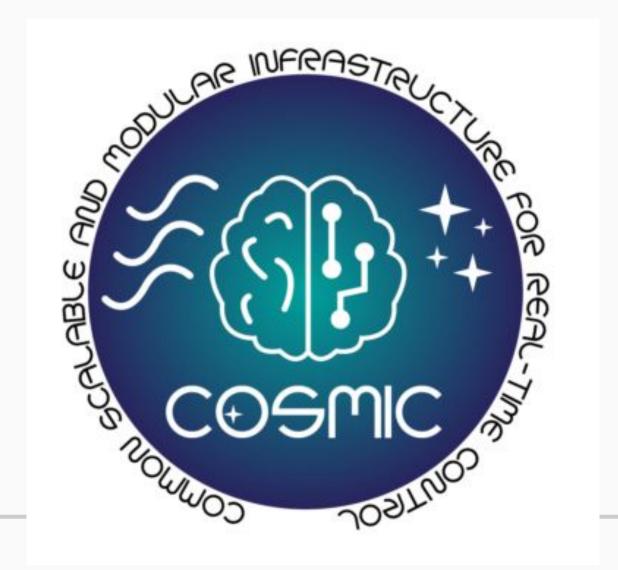
- Already well integrated into ESO ecosystem
- Already at scale for the most challenging dimensioning
- Multiple loops / multiple framerates by design
- Already supporting various AO flavors: SH / Pyr / Laser / tomography

COSMIC is a way to mitigate risk:

- SPARTA obsolescence and permanent failure risks
- Consortium development: addressing schedule risk
- Adaptable to various workloads / approaches (multiple control strategies, multiple instrument configurations)

COSMIC is ready for the next level:

- Scalable
- Future proof: AI friendly



That's it for today !