

Xeon Phi

David Jenkins, Ali Basden, Matt Townson









A many-core CPU (64-72 cores, depending on model)

A standard CPU - self hosted and self booting

Runs standard Linux, standard compiler tools

We currently use Centos

Not an accelerator (in accelerators work package for historical reasons)

Though an accelerator version will be made available

A key design decision (important for AO RTC): 16GB High bandwidth memory

320GB/s theoretical, xxx GB/s measured at Durham

Wide vectorisation unit: 16 floats operated on simultaneously in each core









RTC pipeline operations

First step - a simple test using simple code performing the basic RTC operations

Image calibration, Slope calculation, Reconstruction

Using standard Intel BLAS libraries where appropriate

To get best performance

For ELT SCAO case (74x74 sub-apertures):

~800us computation time -> 1.2kHz

Therefore, this becomes the maximum goal for a real RTCS





Durham AO Real-time Controller

The RTC for CANARY

C-based

Compiles without modification on the Phi

But not performance optimised









Improvements to DARC code

Vectorisation and memory alignment

Previously mostly aligned to 16 byte boundaries

Now more aligned to 64 byte boundaries

Thread synchronisation

Previously well optimised for smaller numbers of cores

Now, mutexes are replaced by spin locks and spin waiting

Thread affinity, count and priority

Has always been user-tunable

We have now investigated optimal affinity, count and priority for the Phi









Current state

Runs at about 800Hz for ELT SCAO, jitter O(10) us rms

Bottlenecks: MVM (as expected), DM command addition, calibration

A full on-sky tested code

		current project> - Intel VTune Amplifier		=
	r CPU Usage viewpoint (change) @			INTEL VTUNE AMPLIFIER XE 201
	nalysis Type ដ Summary 🔏 Bottom-up 🚱 Caller/Callee 🔹 To	n Journ Warn 199 Distingue		
	nalysis ype 🚡 Summary 🚳 Bottom-up 🍫 Caller/Callee 🚺 Io	p-down iree E Platform		
ouping: Function / Call Stack			~ <u>*</u> Q %	CPU Time
Function / Call Stack		CPU Time 🔻		Viewing ∢ 1 of 1 ▷ selected stack
	Effective Time by Utilization	Spin Time	Overhead Time	100.0% (80450.519s of 80450.519s
MKL BLAS]@sgemv	80450.519s	05		libre common. sol (<u>Mkk</u> , <u>BLAS</u>) Elegae. Libre common. sol (<u>klas</u> , <u>ageme</u>) 4043 - (<u>un</u>) libre common. sol (<u>klas</u> , <u>ageme</u>) 4043 - (<u>un</u>) Libre common. <u>sol (<u>klas</u>, <u>ageme</u>) 4043 - d. <u>darmaning coestrange</u> + 0045 - d. Libpthread so. 01<u>start, thread</u> + 0x64 - <u>(</u><u>unknow</u>) - <u>thread</u> + </u>
thread spin lock	25147.934s	0s		
ubapPxlCalibration	8693.283s	Os		
vaitNextSubaps	5281.396s	0s		
opySubap	5040.388s	0s		
alcCentroid	3747.156s 🦲	0s		
thread_barrier_wait	2748.854s 🥚	0.280s		
alibrateNewSubap	1306.519s	Os		
MKL BLAS]@saxpy	1078.695s	Os		
pplyPowerFactor	811.397s	0s		
opeCalcSlope	649.051s	0s		
intel_mic_avx512f_memset	587.631s	0s		
rocessFrame	378.393s	Os		
gemv	158.765s	Os		
/KL BLAS]@errchk_sgemv	146.466s	0s		
conEndFrame	146.361s	0s		
blas_sgemv	103.310s	0s		
omputePixelsRequired	93.520s	05		
ахру	86.226s	0s		
	81.414s	0s		
MKL BLAS]@xsaxpy	81.414s 76.600s	0s 0s		
MKL BLAS]@xsaxpy lopeStartFrame MKL SERVICE]@set xerbla interface ℃Q+Q+Q=000 3000 3000			s 2500s 2700s	2900s Thread
MKL BLAS]@xsaxpy :lopeStartFrame MKL SERVICE]@set xerbla interface	76.600s 51.963s	05 05	s <u>2580s</u> 2700s	29005 ♥ Thread ♥ ■ Running ♥ ■ Running ♥ ■ CPU Sam ♥ CPU Usam ♥ CPU Usam ♥ CPU Usam



Tree-addition for DM commands (rather than linear) Allowing removal of global barriers Investigation of half-precision float storage Investigation of calibration points Attaching a 10G camera (now attached) Real-time kernel









AO Supervisor

To provide calibration and control

Non-real-time

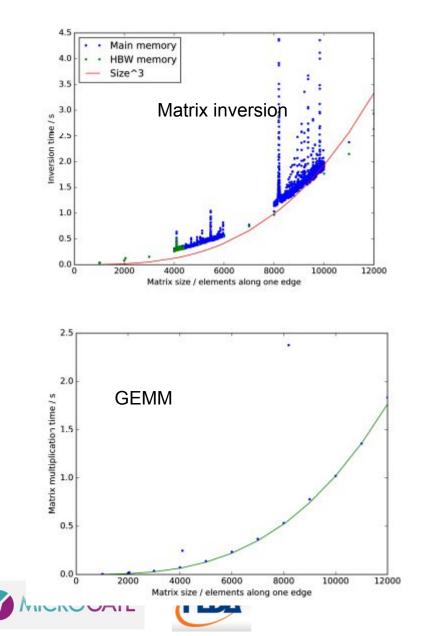
But requires regular update of some parameters

Specifically, the large reconstructor

- Matrix inversion and matrix-matrix multiplication being key operations
- Such operations are well supported on the Phi

bservatoire - LESIA

- Many such libraries exist, including extension to multiple Phi's
- (standard HPC stuff!)



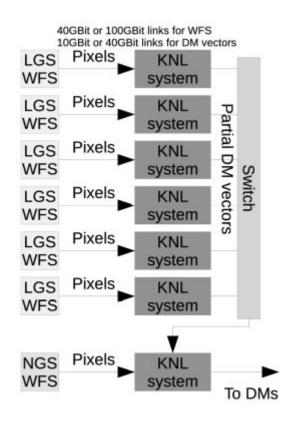


Potential hardware design

4U rack system, cost approx £40k

Other options could be:

- 1 LGS into multiple KNL -
 - Allowing more advanced processing algorithms
- Each LGS multicast into another server dedicated to saving of pixel data
- KNL (or GPU) accelerator within each node for **MVM**





University







Xeon Phi technology is highly suited to ELT-scale AO RTC

- Low cost
- Low barrier to adoption
- Standard computer software techniques
 - We can use standard HPC libraries
- Code fully portable to other platforms:
 - Power8
 - ARM
 - AMD/Intel CPU
 - I486, Pentium, possibly i386 (with old Linux kernel)
 - A fully generic solution





