THIRTY METER TELESCOPE

Fast End-to-End MCAO Simulations with MAOS on GPU

Lianqi Wang

TMT AO Group

AO4ELT2, Victoria, BC

2011-09-30

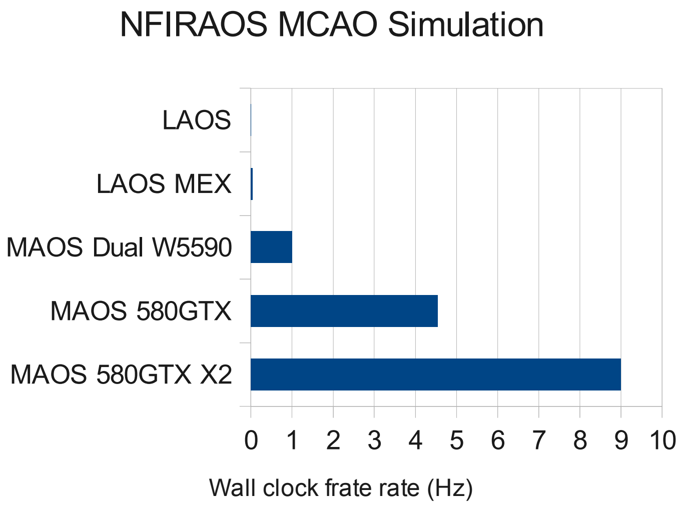
Outline

- Multi-threaded Adaptive Optics Simulator (MAOS)
- MAOS Features
- MAOS timing on CPU
- Porting MAOS to GPU using CUDA
- “Soft” RTC with GPU
- Conclusions

Multithreaded Adaptive Optics Simulator (MAOS)

- Initial development started in 2009 as a rewritten of the original MATLAB based linear adaptive optics simulator (LAOS) in C to make it efficient
- Speed up has been dramatic
  - 40 second a step with LAOS
  - 1 second with MAOS using 8 cores (dual W5590, Nehalem)
  - 0.1 second with MAOS using 2 GTX 580 GPU

MAOS is freely available at http://lianqiw.github.com/maos/



MAOS Features

- Written in C99, using cholmod, blas/lapack, and fftw
- Runs in Linux, Mac and Windows (with cygwin)
- Fully configurable through text .conf files and command line options. Easy for batch processing of large parameter space
- If built with GTK, we have
  - Built-in scheduler for job scheduler and monitoring
  - Built-in plotting routine

MAOS is freely available at
http://lianqiw.github.com/maos/

MAOS Features (continued)

- End to end simulation of all popular AO systems:
  - LGS or NGS based SCAO, MCAO, MOAO, GLAO, LTAO, etc
- FFT (or Frim) based atmosphere
- Telescope (M1, M2 or tilted M3) or instrument phase or amplitude effects, distortion, misregistration, etc
- Shark-Hartmann WFS
  - elongated spot based upon laser uplink and sodium profile
  - photon, detector noise, fratricide effect, etc.
  - tCoG or (constrained) matched filter pixel processing
- Deformable Mirror
  - bilinear or bi-cubic spline influence function
  - clipping, hysteresis. Simple integrator control.

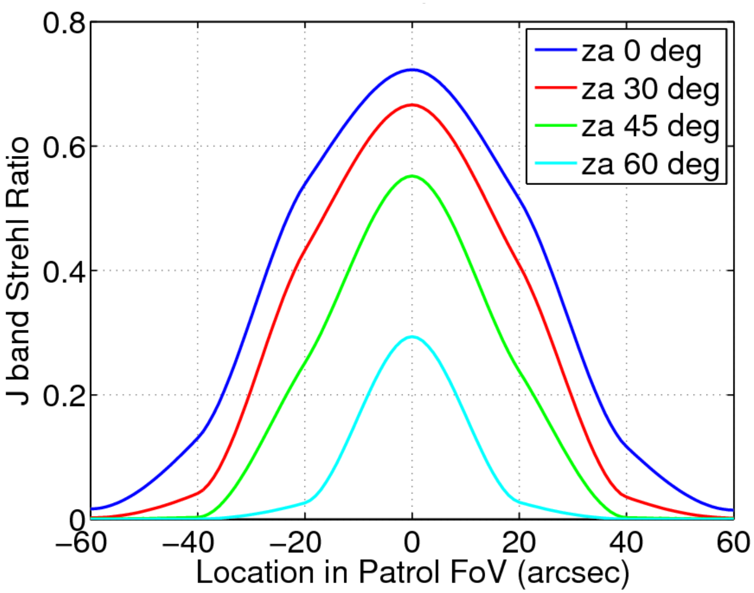
TMT
THIRTY METER TELESCOPE

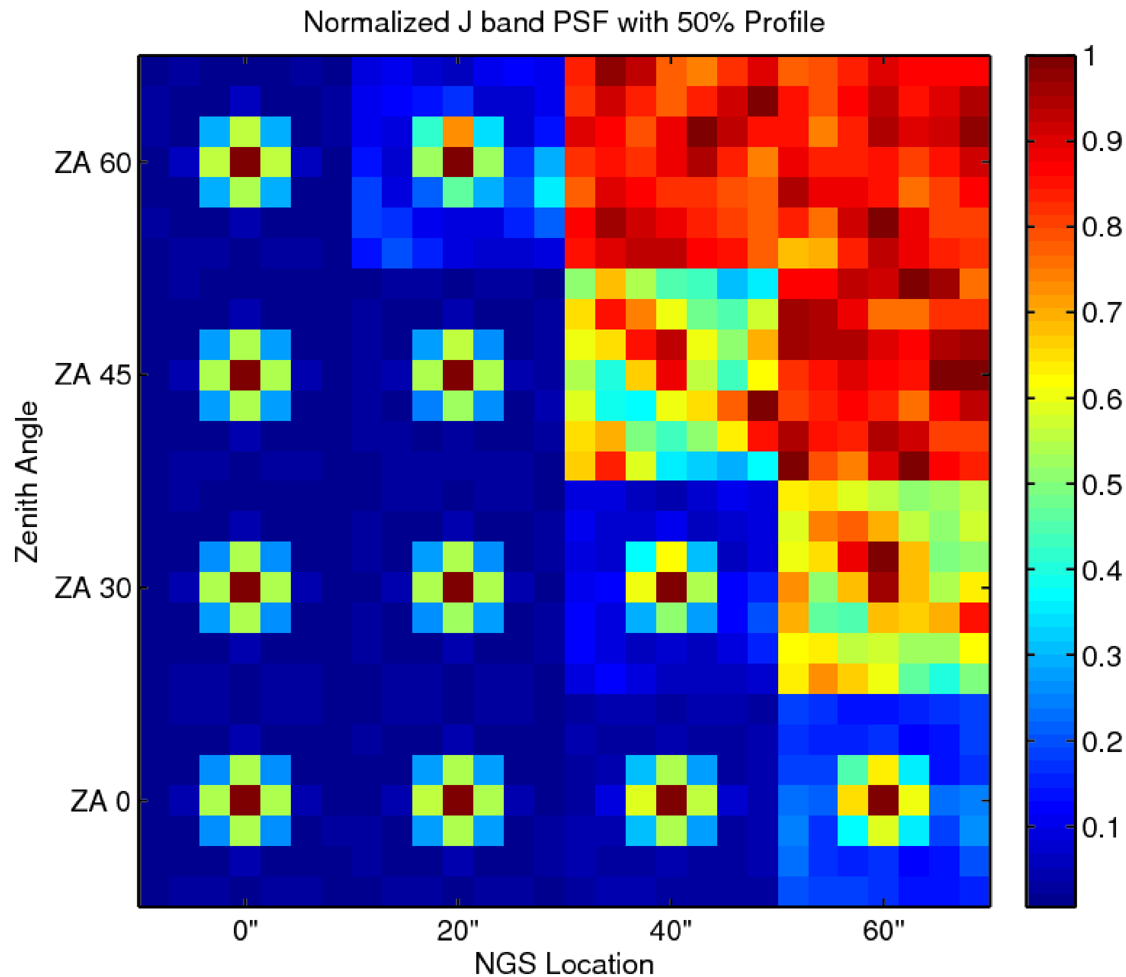
MAOS Features (cont.)

- Conventional Least square reconstruction
- Minimum variance reconstruction with PSOL gradients
  - tomography, with *prior* C_{xx}^{-1} using
    - Bi-harmonic approximation (laplacian)
    - Frim
    - Fourier domain operator
  - DM Fitting:
    - Using cubic actuator influence function to simulate inter-actuator coupling
  - Minimum variance split tomography to get optimal NGS control
- Built-in SloDAR for automatic updating tomography
  - Wind profiling not yet done.

Physical Optics Sky coverage with MAOS

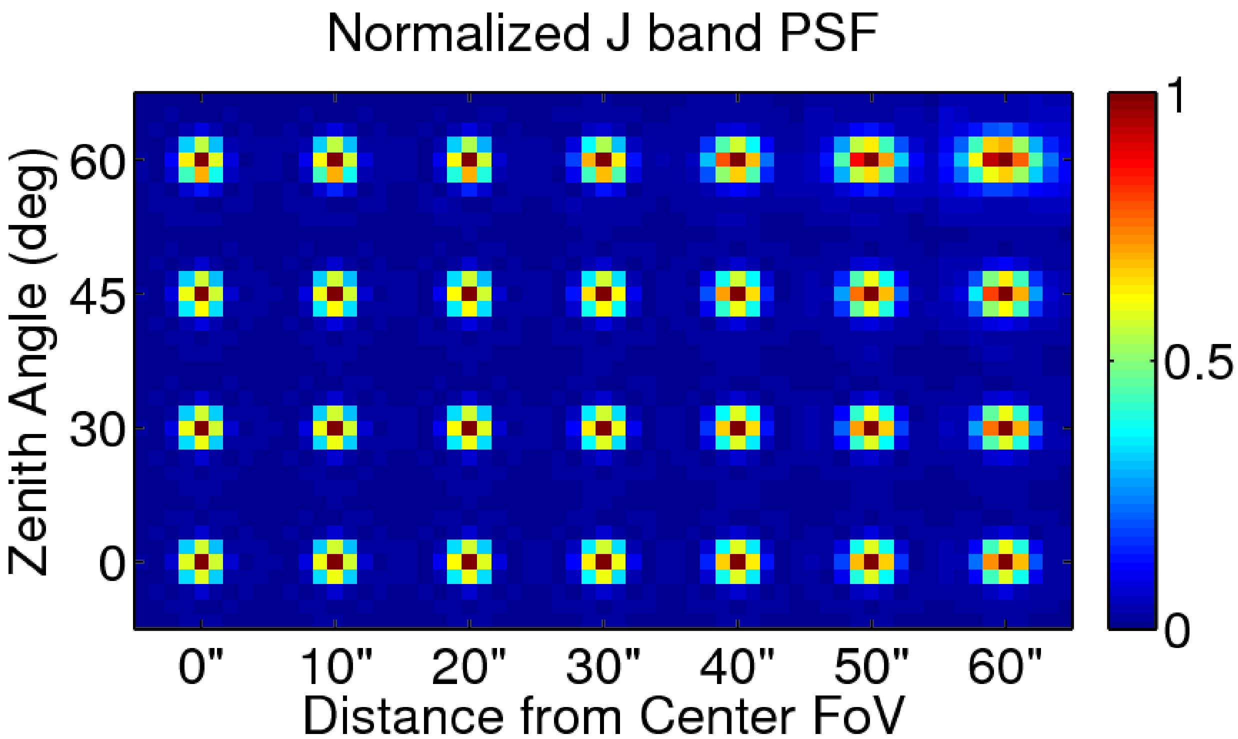
- High fidelity sky coverage using time domain physical optics simulations.
- We discovered PSF breaks down!





Physical Optics Sky Coverage: Long term solution: MOAO for OIWFS

- MOAO for OIWFS using MEMS DM
- Diffraction-limited PSF core maintained at all zenith angles with ideal MOAO correction



Timing on CPU
End-to-End Simulation of NFIRAOS

- Full end-to-end simulation for NFIRAOS (6 LGS, 2 DM)
  - 7 layer atmosphere sampled at 1/64 meter, (from 64 to 512 m)
  - Physical optics LGS with elongation and matched filter.
    - 32 points across subaperture. 16x6 pixels each subaps.
  - Tomography with CG30, DM fitting with CG4
  - Performance evaluation of 9 points for RMS OPD error

Machine	Per time step
Xeon W5590, single thread	7 s
Dual W5590 quad core at @ 3.33 GHz	1 s
Desktop core i7 quad core @ 2.8 GHz	2.1 s
Laptop dual Core i5 @ 2.4 Ghz	5 s

Porting MAOS to GPU

- Motivation:
  - For faster simulation
  - Assess the capability of GPU for RTC
- Hardware
  - NVIDIA GTX 580 in a Desktop with Core i7 860 @ 2.80 GHz
    - 3 GB graphics memory
    - 512 stream processors
- Software
  - CUDA 4.0 C runtime library with special compiler, nvcc
  - cublas, cuFFT, cuSparse, cuRand, etc
  - Use single precision floating number for best GFlops

Wavefront sensing and performance evaluation in GPU

- Atmosphere generated in CPU and transported to GPU
- Wavefront sensing
  - Ray-tracing from atm. and DM to pupil grid at 1/64 m sampling
  - Multiple FFT per subaps. to get images sampled on pixels
  - Adding noise
  - Calculating gradients using constrained matched filter
- Performance evaluation
  - RMS WFE computation
  - PSF computed using FFT and accumulated in device memory.
- Minimum communication between CPU/GPU, GPU/GPU
  - Easily parallelization across for multiple GPUs

Minimum Variance Reconstructor

- Minimizing σ^2 over target FoV

  $$\sigma^2 = \| H_x x - H_a a \|^2$$

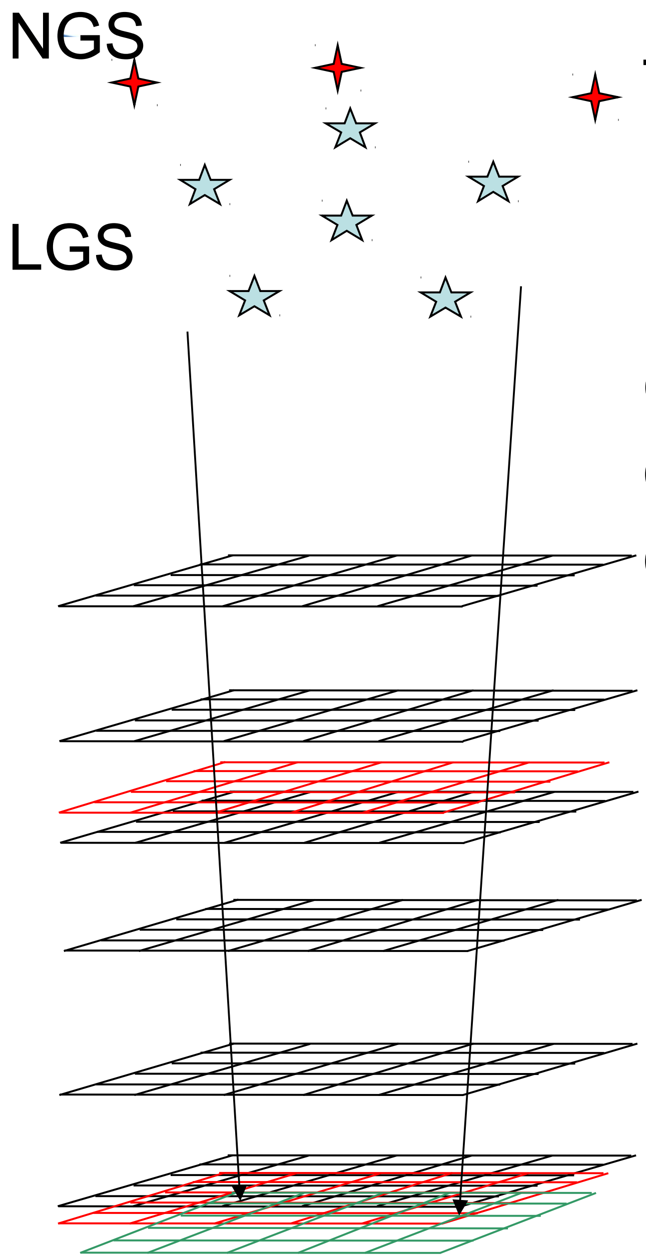
  $$\text{with } g = \Gamma x + n$$

- Gives tomography

  $$x = \left(H_x^T G_p^T C_{nn}^{-1} G_p H_x + C_{xx}^{-1} \right)^{-1} H_x^T G_p^T C_{nn}^{-1} g$$

- And DM fitting over target FoV

  $$a = \left(H_a^T W H_a \right)^{-1} H_a^T W \tilde{H}_x x$$



Tomography

$$x=\left(H_x^T G_p^T C_{nn}^{-1} G_p H_x + C_{xx}^{-1}\right)^{-1} H_x^T G_p^T C_{nn}^{-1} g$$

H_x: ray tracing from x to p

G_p: compute gradient from p

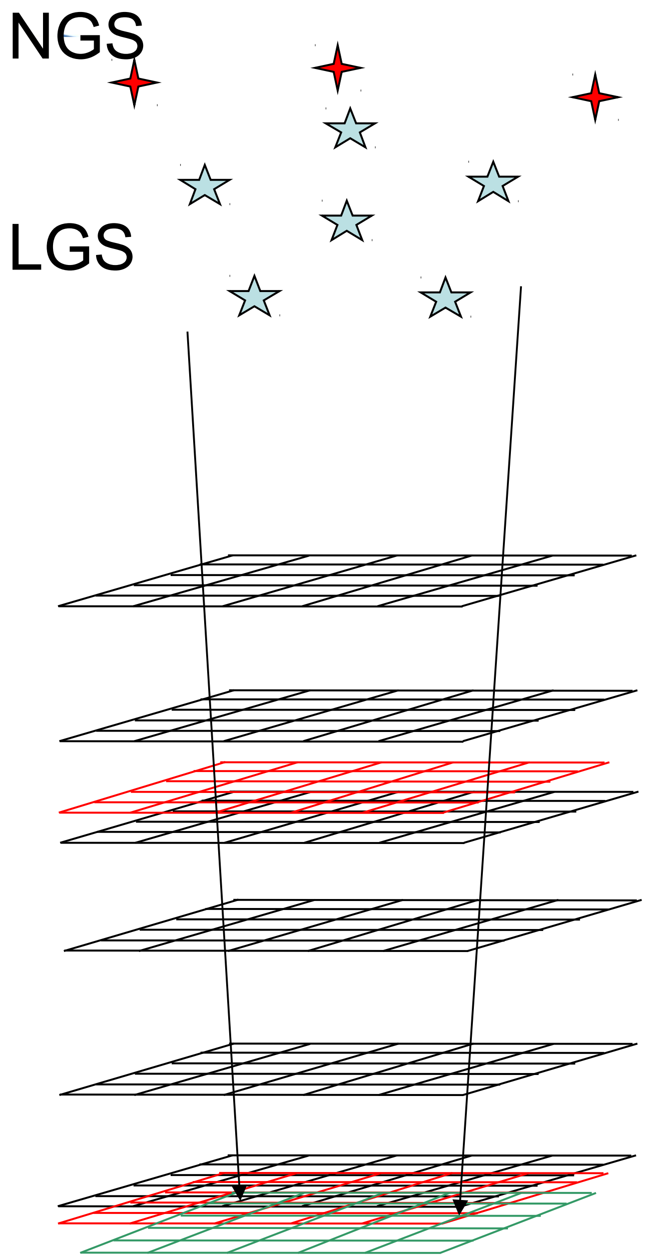
C_{nn}: Noise covariance matrix

C_{xx}^{-1}: Using bi-harmonic approximation

x: Trubulence grid at ¼ or ½ m

a: Actuator grid at ½ m

p: pupil grid at ½ m.



DM Fitting

$$a = \left(H_a^T W H_a\right)^{-1} H_a^T W \tilde{H}_x x$$

Use sparse matrix based operation for the moment

x: Trubulence grid at ¼ or ½ m

a: Actuator grid at ½ m

p: pupil grid at ½ m.

Timing on GPU vs CPU
End-to-End Simulation of NFIRAOS

- Hardware Setup
  - CPU: 2.8 G Core i7 Quad Core
  - GPU: Single Nvidia 580 GTX

Timing in seconds	CPU	GPU
Geometric WFS	0.36	0.03
Physical Optics WFS	1.36	0.11
Perf. evl. with RMS OPD	0.47	0.04
Reconstruction	0.31	0.0314
Total (thread pool)	2.02	0.17

- We have $\geqslant 10$ x speed up in every case

Detailed timing on GPU for tomography ("soft" RTC)

- Timing on GPU 580. Fastest in the planet, however
  - Kernel launch overhead:
    - ~2.3 micro-second for asynchronous launch,
      - because the GPU driver has to instrument memory copy and device configuration
    - ~6.5 for synchronization
      - mutex locking and device query, etc
  - Computing peak throughput: 1581 GFlops
  - Device memory throughput, 192 GB/s
  - Device memory latency: 600 cycles, ~0.3 micro-second
  - PCI-E interface: 8GB/s, 11 micro-second latency.
- Knowing the hardware is the key to get best performance

Tomography ray tracing Timing on GPU

- Hx: ray tracing from 6 layers to 6 WFS, 4 layer at ¼ m and 2 layer at ½ meter sampling. WFS OPD at ¼ meter.

	μs	Gflops	GB/s	Operations	Memory R/W
Hx	161	28.4	83.2	4.6 M	13.7 MB
Hx'	246	18.6	54.5	4.6 M	13.7 MB

- Theoretical throughput: 1581 GFlops, 192 GB/s

Gradient operation Timing on GPU

- Gradient operator Gp and its transpose Gp'
- Store coefficients for every subaperture.
  - Too many partially illuminated sub-apertures (>50%) due to primary mirror segment gaps and edges.
- Gp' is slower than Gp due to atomic operations.

	μs	Gflops	GB/s	Operations	Memory R/W
Gp	48	11.6	39.0	556,848	1.9 MB
Cnn^{-1} Gp'	91	7.5	22.6	680,592	2.1 MB

- Theoretical throughput: 1581 GFlops, 192 GB/s

Laplacian (C_{xx}^{-1}) Timing on GPU

- OPD grid is sampled at ¼ m with ½ m subaperture
- Matrix free implementation with periodic boundary conditions
- Good efficiency with regular memory access

	μs	Gflops	GB/s	Operations	Memory R/W
Cxx^{-1}	85	19.7	63.2	1.7 M	5.5 MB

- Theoretical throughput: 1581 GFlops, 192 GB/s

Tomography Timing on GPU

- Tomography takes 23.5 ms

$$x = (H_x^T G_p^T C_{nn}^{-1} G_p H_x + C_{xx}^{-1})^{-1} H_x^T G_p^T C_{nn}^{-1} g$$

	μs	Gflops	GB/s	Operations	Memory R/W
RHS	482	10.9	32.1	5.2 M	16 MB
LHS	603	20.0	59.9	12 M	37 MB
CG30	23500	16.2	49.2	382 M	1.2 GB
RTC	500	763.2	2367	382 M	1.2 GB

- Theoretical throughput: 1581 GFlops, 192 GB/s

Further Improvement

- We are limited by
  - Memory throughput and latency (~600 cycles, 0.3 micro-second)
  - Kernel launch overhead:
    - ~2.3 micro-second for asynchronous launch,
      - because the GPU driver has to instrument memory copy and device configuration
    - ~6.5 for stream synchronization
      - mutex locking and device query, etc
- Reduce number of kernel launches by merging kernels
- Reduce CG iterations to reduce penalty due to latency: FDPCG or something else?

Multiple GPU

- You may think multiple GPU will help, but
  - Memory bandwidth of 16x PCI-E is only 8GB/s, 25x slower than device memory.
  - 10 micro-second GPU to GPU communication latency
  - Distributing the task within each iteration will have a severe overhead
- Need break through in GPU technology.

TMT
THIRTY METER TELESCOPE

Conclusions

- MAOS is an efficient software for ELT AO simulations
  - It might be used to benchmark pixel processing, reconstruction algorithms from different groups to help understand the pros and cons of each algorithm.
- Run MAOS in GPU provides 10x speedup compared to state of art dual quad core CPU.
- NFIRAOS tomography in GPU takes 24 ms. 10x faster than CPU, but still a long way to go before achieving desired latency of 0.5 ms.

MAOS is freely available at
http://lianqiw.github.com/maos/

THIRTY METER TELESCOPE

Thank you