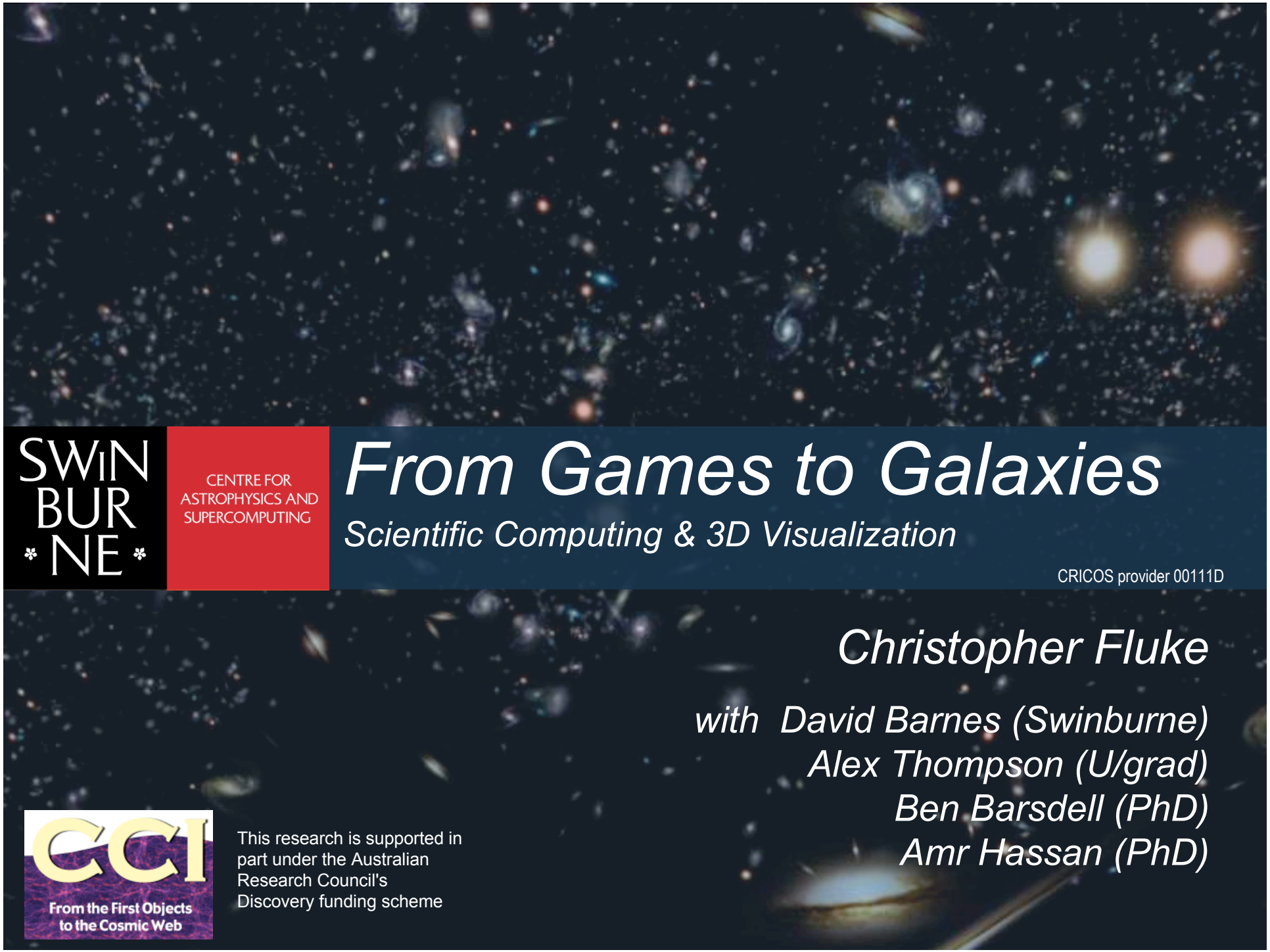


SWINBURNE

CENTRE FOR ASTROPHYSICS AND SUPERCOMPUTING

From Games to Galaxies

Scientific Computing & 3D Visualization

CRICOS provider 00111D

Christopher Fluke

with David Barnes (Swinburne)
Alex Thompson (U/grad)
Ben Barsdell (PhD)
Amr Hassan (PhD)

CCI
From the First Objects to the Cosmic Web

This research is supported in part under the Australian Research Council's Discovery funding scheme

A Talk in Two Halves

Scientific Computing

- Graphics Processing Units
- Example: direct ray-shooting for gravitational lensing

Visualisation

- S2PLOT programming library
- Advanced displays
- [Public outreach: AstroTour]
- Digital publication: 3d-PDF

Evolution

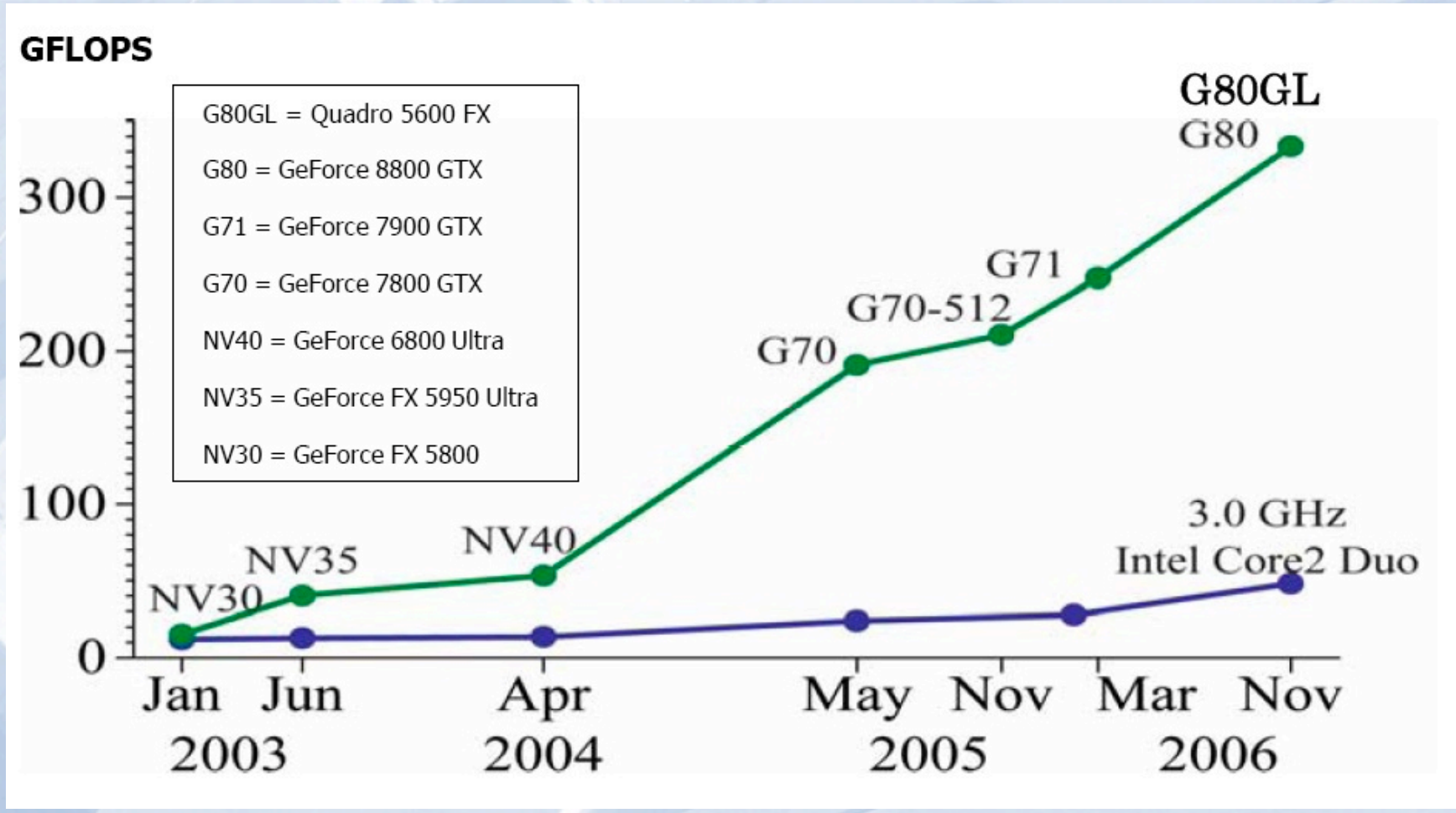
Computer Game Industry Revenue >$60 billion

A Revolution in Scientific Computing?

> 1 Tflop/s (peak)

13 Tflop/s (peak)

Moore's Law for Graphics Processing Units



Floating-Point Operations per Second for the CPU and GPU
NVIDIA CUDA Programming Guide V1.0 (2007)

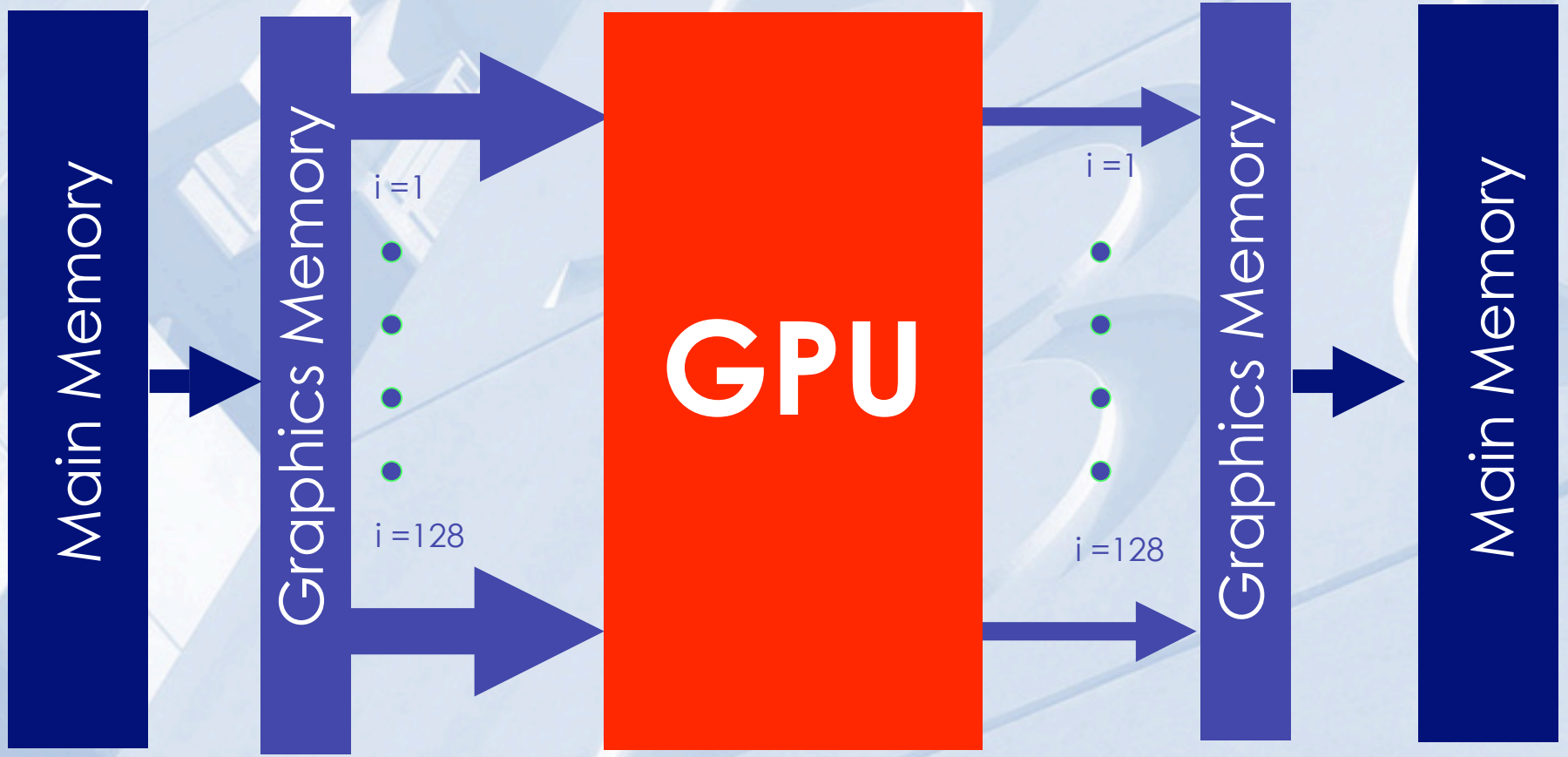
Graphics Processing Units (GPUs)

- Highly parallel architecture (“pushing pixels”)
- “Stream processing”
- One instruction acting on multiple sets of data

- General Purpose Computation on GPUs*
- NVIDIA and AMD development environments
- 2006: Compute Unified Device Architecture (CUDA)
  - www.nvidia.com/cuda

* E.g. Fournier & Fussell (1988); Tomov et al. (2003); Venkatasubramnian (2003); Owens et al. (2005)

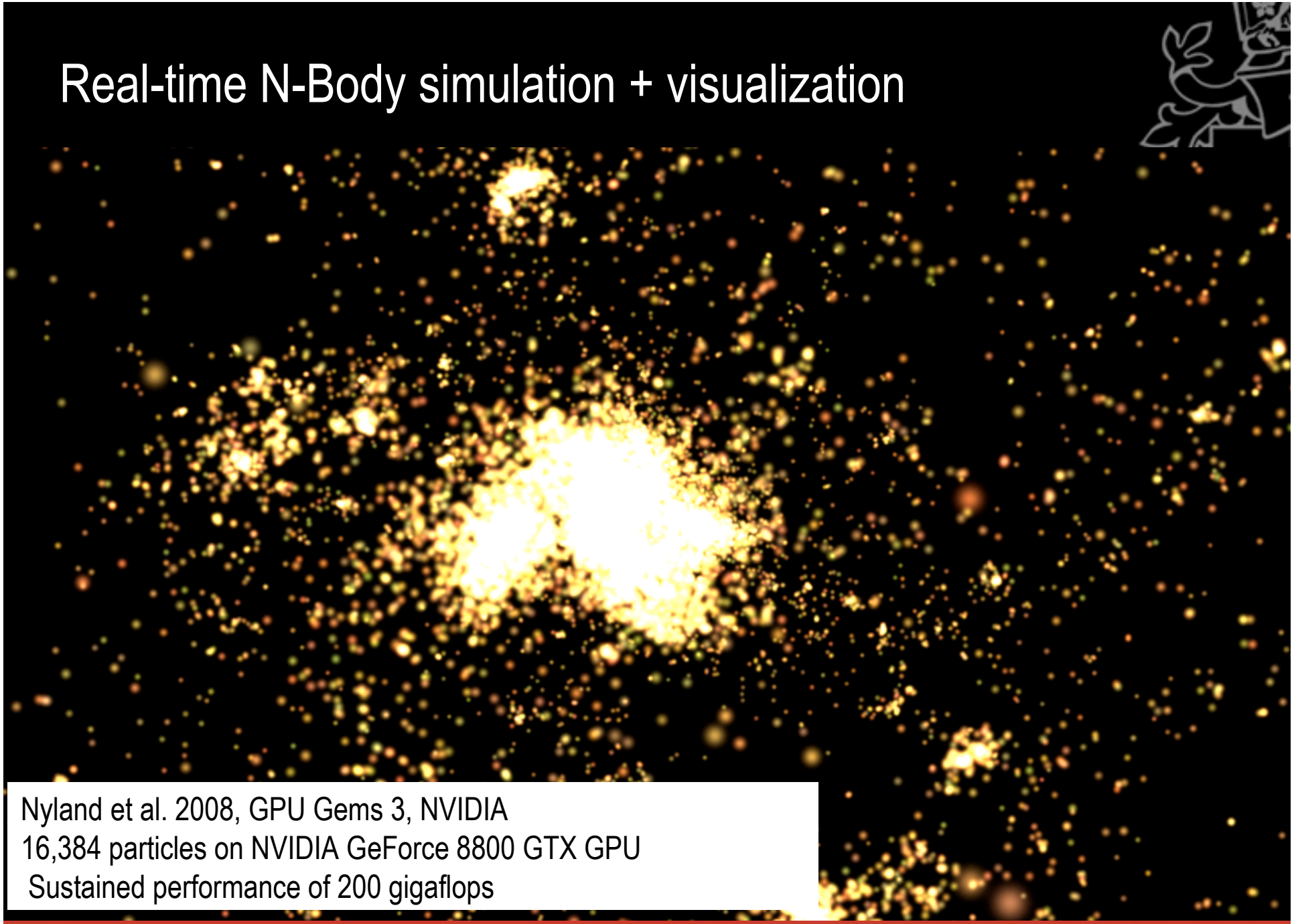
GPU Programming Pipeline



GPUs in astronomy

- N-body: $O(N^2)$ force calculation
  - $O(10)$ times speed-up with shader language
    - Nyland et al. (2004), Portegies Zwart et al. (2007)

Real-time N-Body simulation + visualization

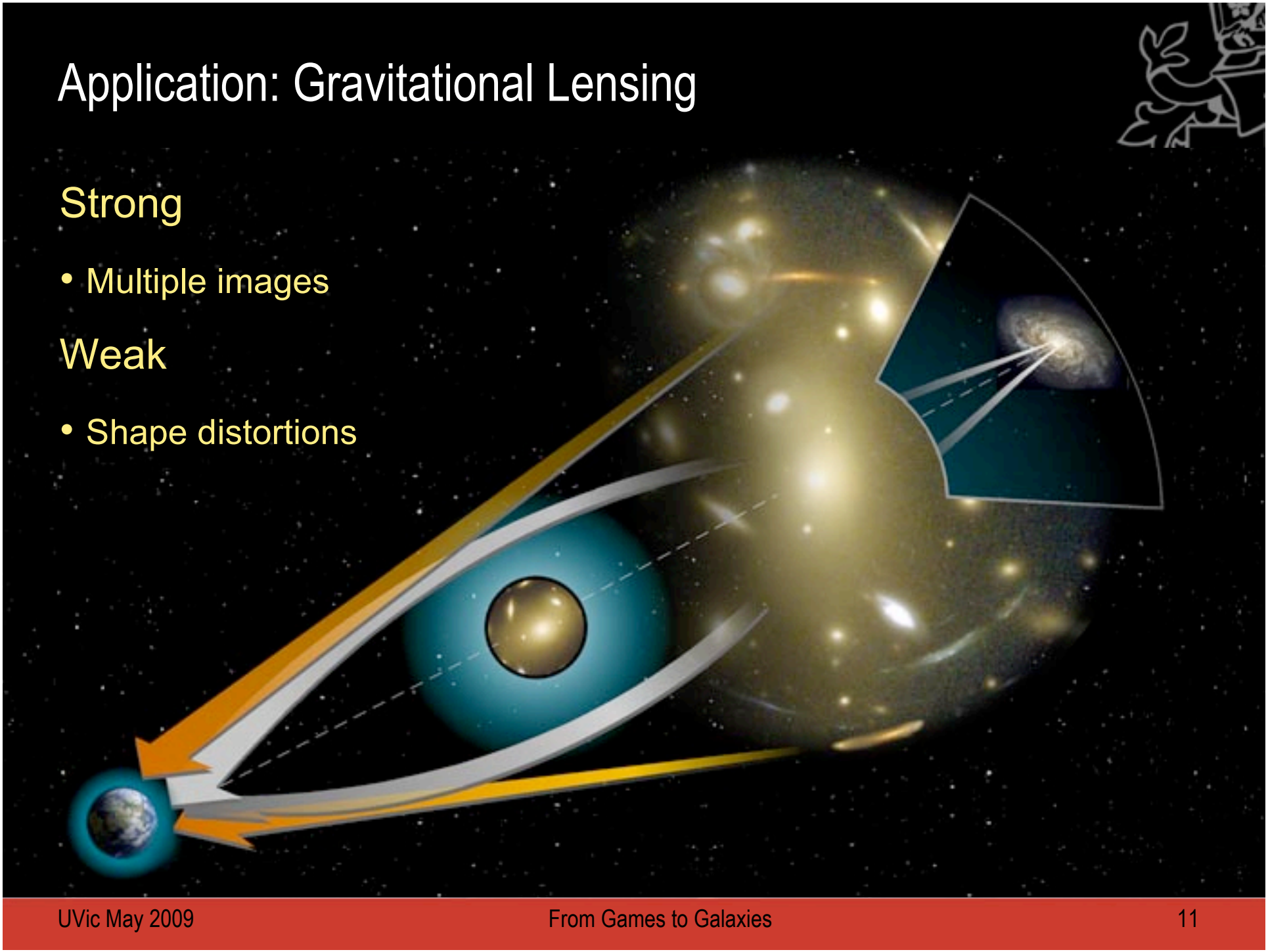


Nyland et al. 2008, GPU Gems 3, NVIDIA
16,384 particles on NVIDIA GeForce 8800 GTX GPU
Sustained performance of 200 gigaflops

GPUs in astronomy

- N-body: $O(N^2)$ force calculation
  - $O(10)$ times speed-up with shader language
    - Nyland et al. (2004), Portegies Zwart et al. (2007)
  - O(100) with speed-up CUDA
    - Hamada & Iitaka (2007), Belleman et al. (2007); Schive et al. (2007); Nyland et al. (2008); Moore et al (2008)
- Radio-telescope signal correlation (Schaaf & Overem 2004; Wayth et al. 2007; Harris et al. 2008; Ord et al. 2009)
- Solution of Kepler's equations (Ford 2008)
- Real-time visualization of large datasets (Szalay et al. 2008)

Application: Gravitational Lensing



Strong

- Multiple images

Weak

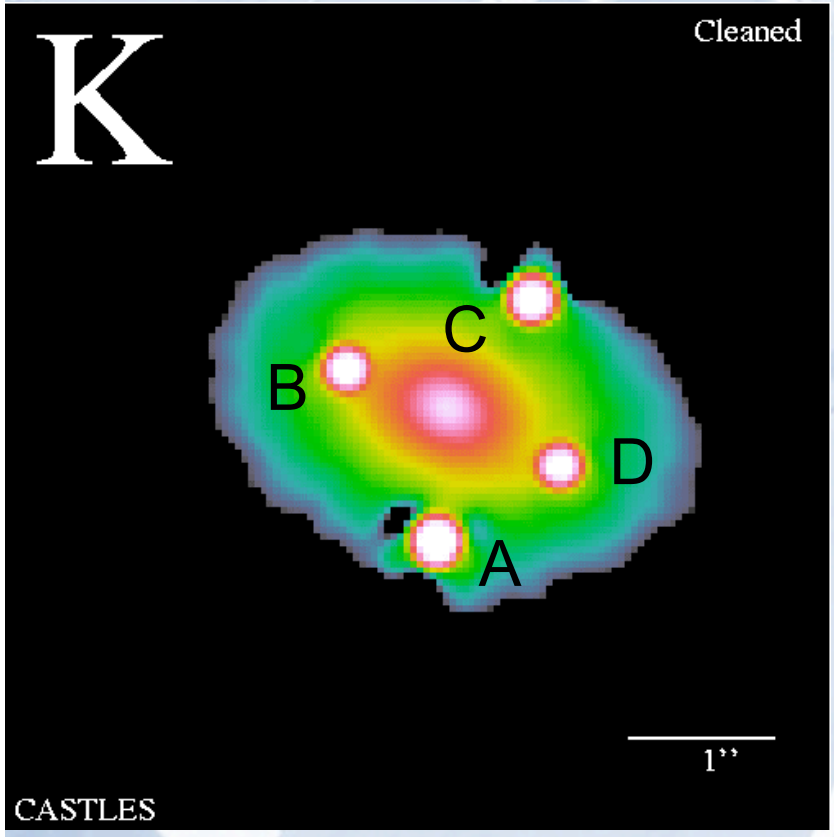
- Shape distortions

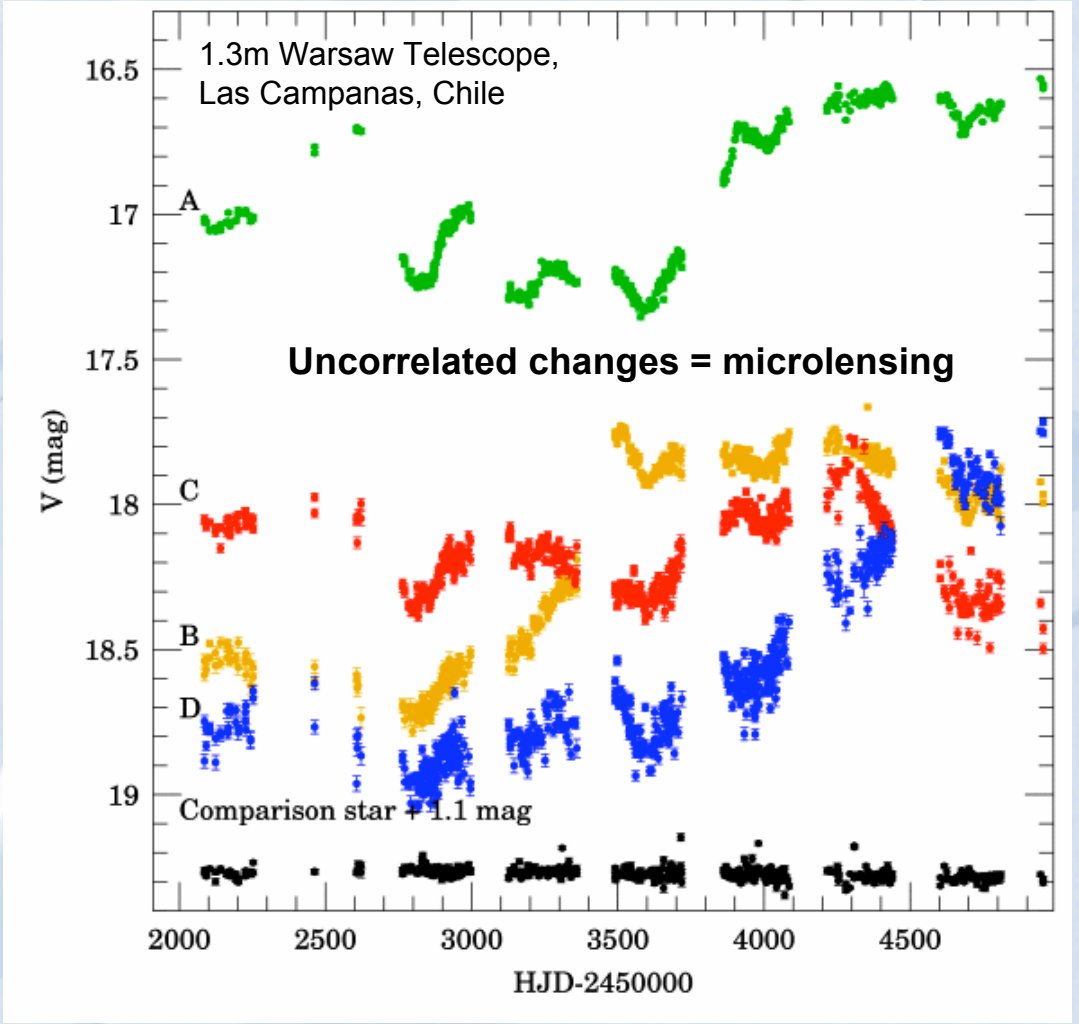
Gravitational Microlensing

- High magnification
- Multiple images
- Individual micro-images not resolved
- Time-varying change in brightness

- Compact objects in Galactic Bulge and Halo (MACHO; OGLE; PLANET)
- Compact objects in macrolenses at cosmological distances (QSO microlensing)

2237+0305: The Einstein Cross





z_{lens} = 0.04

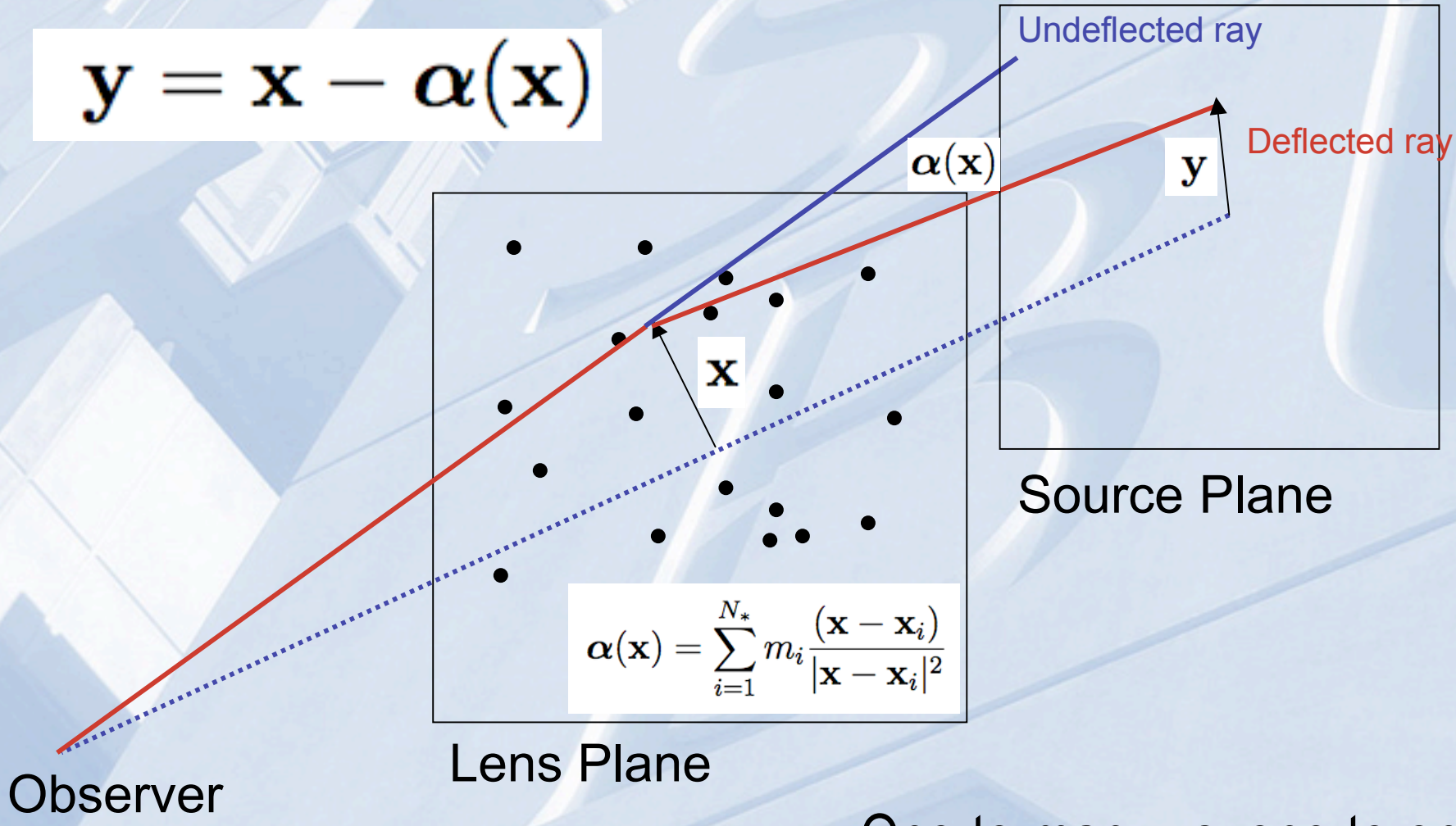
z_{qso} = 1.69

ΔT = 2-30 hours

http://ogle.astrouw.edu.pl/

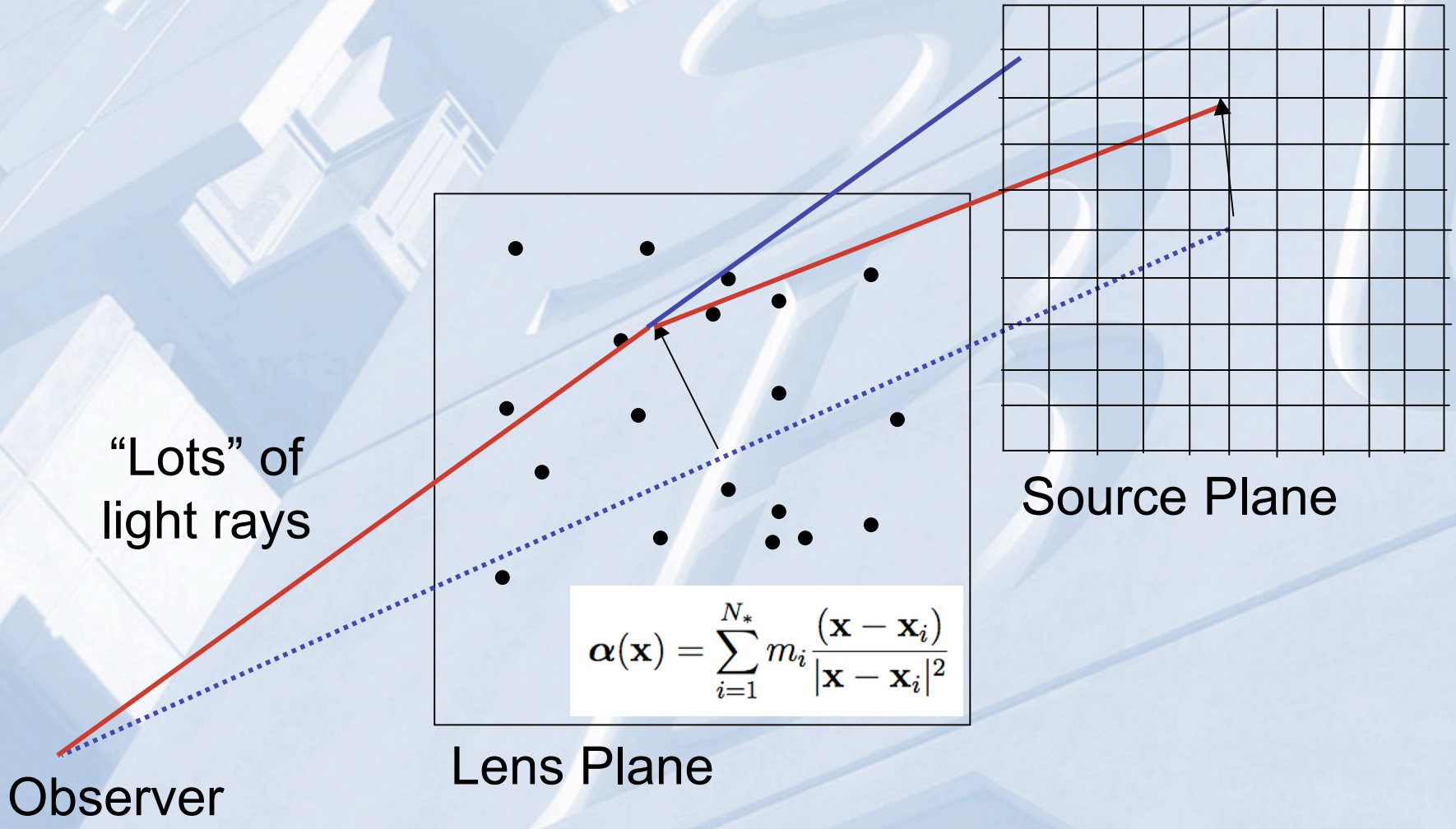
Gravitational Lens Equation

$$\mathbf{y} = \mathbf{x} - \boldsymbol{\alpha}(\mathbf{x})$$



One-to-many vs. one-to-one

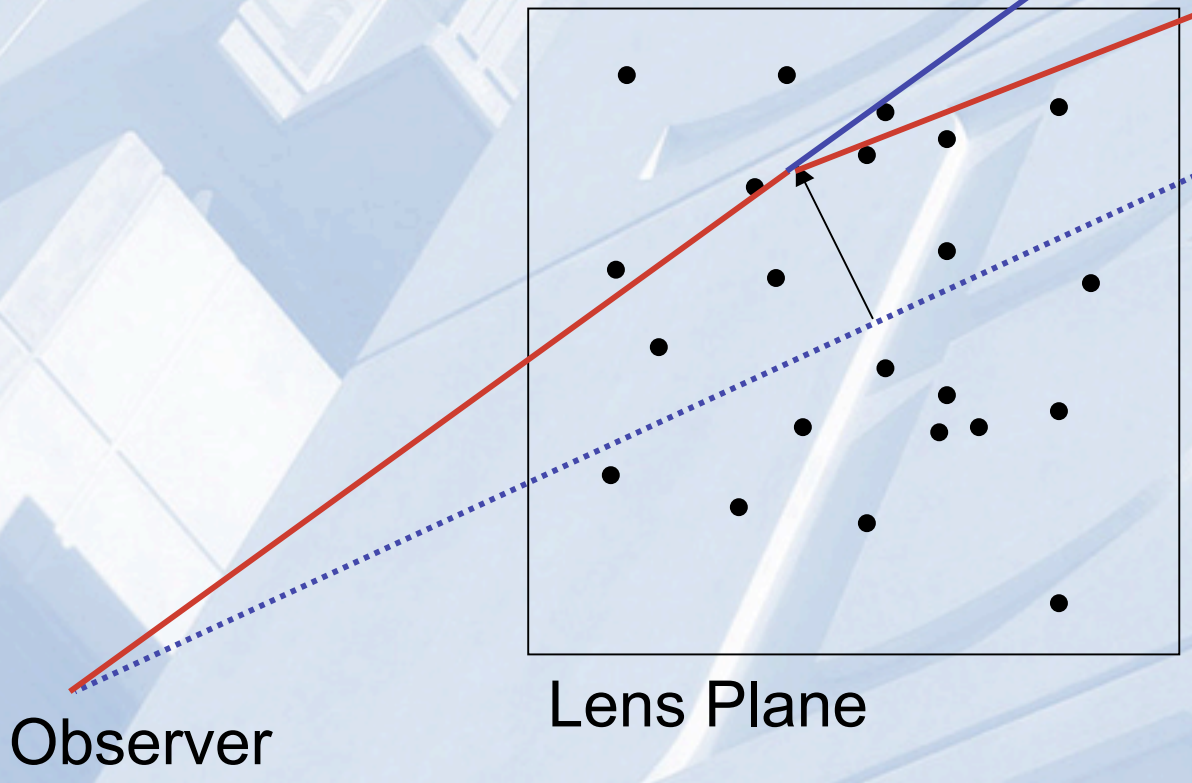
Inverse Ray-shooting

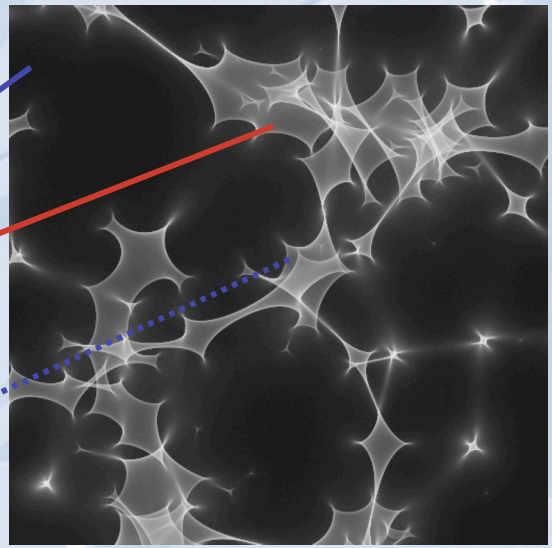


Inverse Ray-shooting: Magnification Map

$$\mathbf{A} = \partial \mathbf{y} / \partial \mathbf{x}$$ Jacobian

$$\mu = 1 / \det \mathbf{A}$$ Magnification



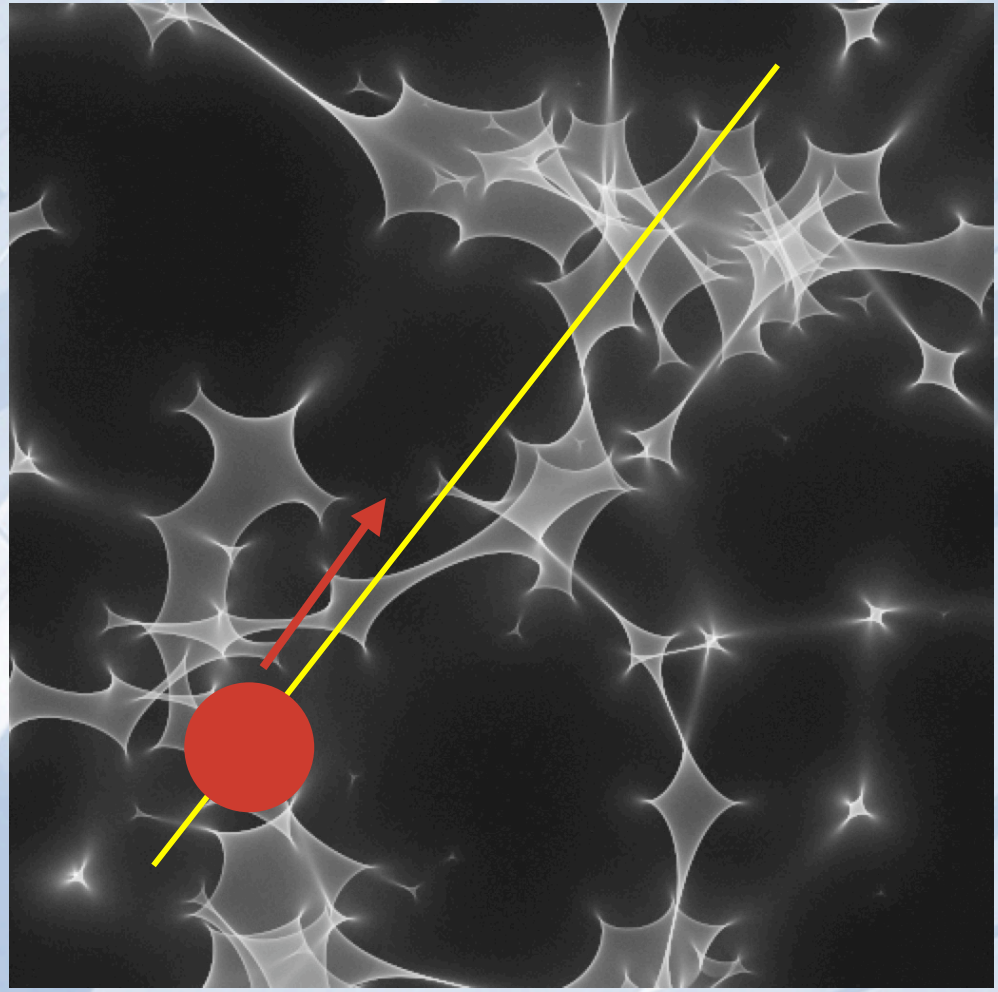


Source Plane

Magnification Map

$$\mu_{ij} = N_{ij} / N_{\mathrm{av}}$$

Light curves



Science outcomes:

- Stellar mass function
- Mean stellar mass
- Structure of quasars
- Constrain emission region geometry at different wavelengths

Need:

- Lots of light rays (>10^6)
- Lots of lenses (>10^6 ?)
- High-resolution grid
- Statistically valid (N_{av} > 100)

Computationally Challenge

n = Number of repeat maps

N_{flop} = Floating point operations per deflection

N_{pix} = Source plane pixels

N_* = Number of lenses

N_{av} = Average rays per pixel

$$\Phi = n \times N_{\text{flop}} \times N_{\text{pix}} \times N_* \times N_{\text{av}}$$

$$\lesssim O\left(10^{16}\right)$$ **Months of processing time...**

Hierarchical alternative: approximate mass at large range (Wambsganss 1990; 1999)

Embarrassingly parallel

Light rays are independent: $O(N_{av} \times N_{pix})$

$$\mathbf{y} = \mathbf{x} - \boldsymbol{\alpha}(\mathbf{x})$$

Deflection due to each lens independent:

$$\boldsymbol{\alpha}(\mathbf{x}) = \sum_{i=1}^{N_*} m_i \frac{(\mathbf{x} - \mathbf{x}_i)}{|\mathbf{x} - \mathbf{x}_i|^2}$$

Break into separate sub-summations

Gravitational Lensing with GPUs

The Questions:

- What type of speed-up can we get for direct inverse ray-shooting on GPU versus CPU?
- As a consequence, how does that affect parameter space searches?

$$\Phi = n \times N_{\mathrm{flop}} \times N_{\mathrm{pix}} \times N_{*} \times N_{\mathrm{av}}$$

Procedure

- Implemented direct inverse ray-shooting algorithm
  - Careful use of GPU memory increases efficiency
    - Device Memory
    - Shared Memory
    - Register Memory
- Timing tests: compared performance on CPU and GPU
- Check accuracy by comparing magnification maps
- Median of three runs

- Full details: Thompson et al., 2009, arXiv:0905.2453

Hardware configurations

Name	Description	Type	Memory
1CPU	Single thread on Intel Q6600	CPU	4 GB
4CPU	OpenMP with four threads on Intel Q6600 CPU	CPU	4 GB
1GPU	Single NVIDIA GeForce 8800 GT	GPU	512 MB
2GPU	Two NVIDIA GeForce 8800 GTs	GPU	2×512 MB
1TES	NVIDIA S1070 Tesla with 4 GPU cores	GPU	4×4 GB

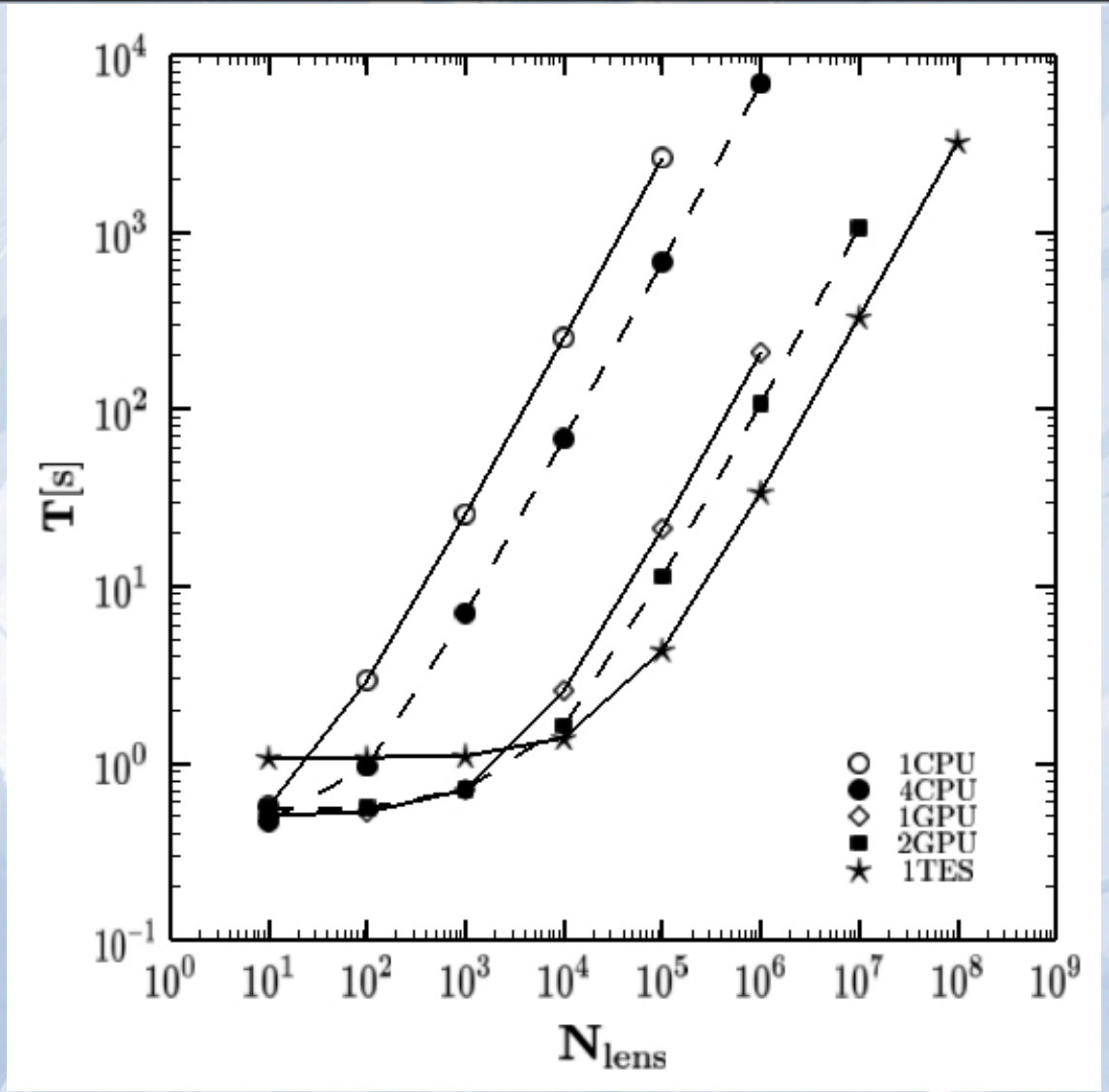

GeForce 8800


Tesla

GPU Algorithm

1. Generate N_* lens positions on CPU, move to GPU
2. Generate N_{batch} light ray coordinates and move to GPU
3. Split GPU computation into groups of 128 threads
4. Each thread loads 128 lenses and deflects 128 rays
   Repeat until lenses and rays exhausted
5. Copy GPU results back to CPU memory
6. CPU maps ray locations onto source pixel grid
7. Repeat steps 2-6 until N_{av} rays per pixel achieved

Dependence on number of lenses



$N_{av} = 100$

$N_{pix} = 192^2$

Linear with N_{lens} (after plateau)

Dependence on number of pixels

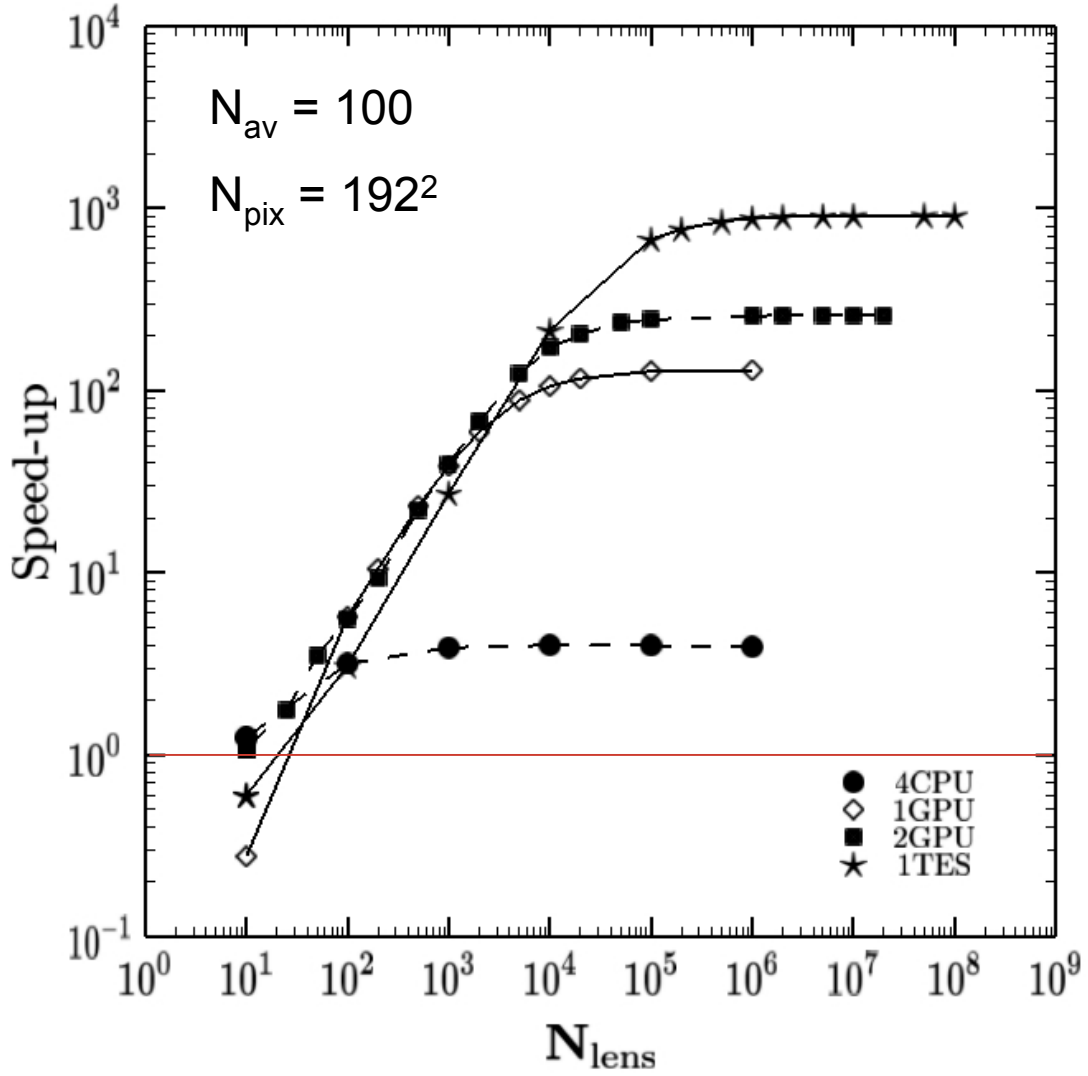
Run-time in Hours

N_{pix}	1CPU	4CPU	1GPU	2GPU	1TES
192^2	7.5	1.91	0.06	0.03	0.01
512^2	[55]	[14]	0.40	0.20	0.06
1024^2	[200]	[55]	1.59	0.80	0.23
2048^2	[850]	[200]	6.35	3.20	0.93
4096^2	[3400]	[850]	25.4	12.8	3.74
8192^2	[13700]	[3400]	[100]	50.4	14.75

[ ] values were not run

$N_* = 10^6$, $N_{av} = 100$

Speed-up relative to 1CPU



1CPU = 1.4 Gflop/s

4CPU = 5.6 Gflop/s

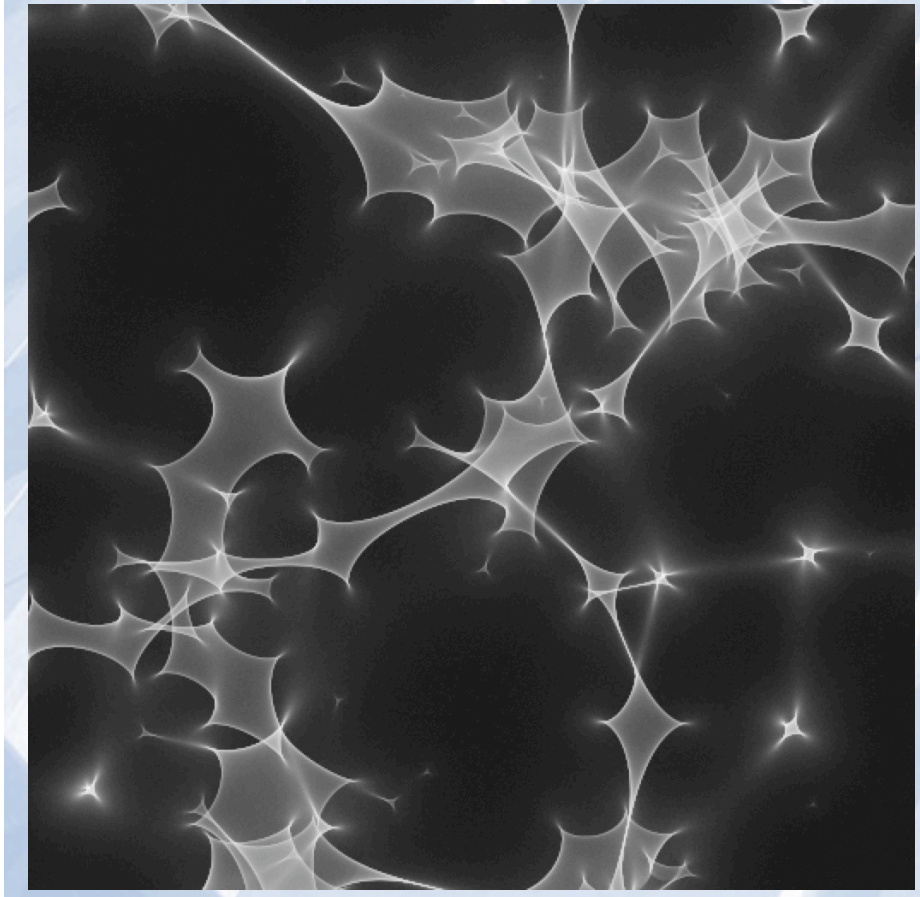
1GPU = 182 Gflop/s

2GPU = 364 Gflop/s

1TES = 1287 Gflop/s

~50% of quoted "peak" due to nature of calculation

Applications



- Time-varying lens positions
  (Schramm et al. 1993)
- Rapid microlensing variability?
  (Schecter et al. 2003; Kochanek et al. 2007)
- Statistical independence of light curves
- “Real-time” lensing

$$T_{1\mathrm{TES}} = 1.4 \times 10^4 \left(\frac{N_{\mathrm{pix}}}{4096^2}\right)\left(\frac{N_*}{10^6}\right)\left(\frac{N_{\mathrm{av}}}{100}\right)\ \mathrm{s}$$

$N_{pix} = 1024^2$, $N_* = 10^5$, $N_{av} = 50$ in 60 seconds

Cost Effective

4CPU:

>$250,000USD per Tflop/s (200 machines + networking)

2GPU:

>$7,000USD per Tflop/s (4 machines + networking cost)

1TES:

~$8,000USD per Tflop/s (1 machine + no networking!)

Does not factor in air-conditioning + power

A Talk in Two Halves

Scientific Computing

- Graphics Processing Units
- Example: direct ray-shooting for gravitational lensing

Visualisation

- S2PLOT programming library
- Advanced displays
- [Public outreach: AstroTour]
- Digital publication: 3d-PDF