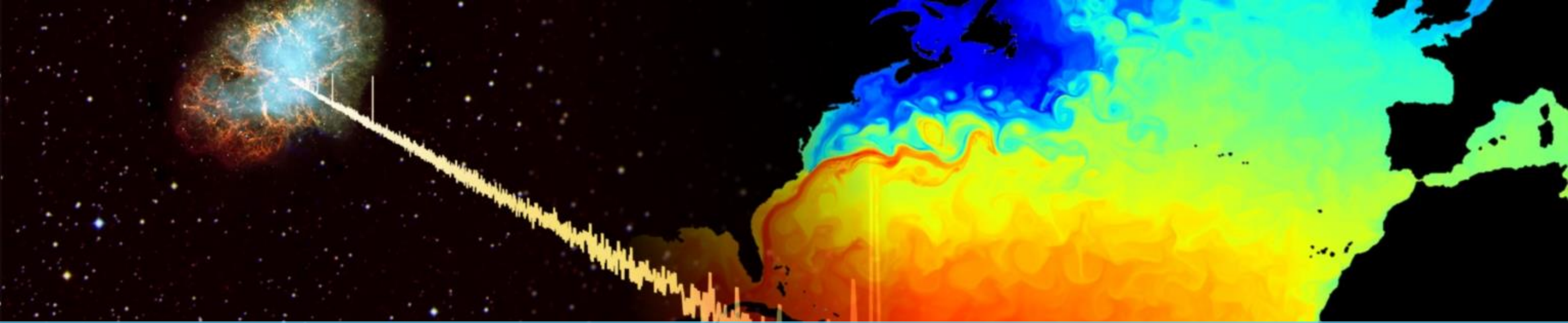
GPU Computing at the Netherlands eScience Center

Ben van Werkhoven

NIRICT – GPU Applications Workshop
Utrecht, June 8 2017

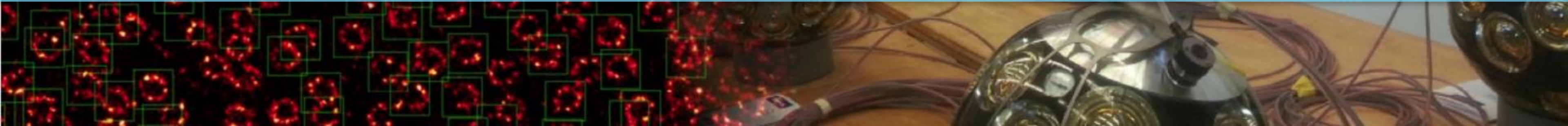
netherlands
eScience center

by SURF & NWO



GPU Applications

- Climate Modeling
- Radio Astronomy
- Super-resolution Microscopy
- Astro-particle Physics
- Life Sciences
- Computational Linguistics
- Digital Forensics



How we work

Yearly calls for proposals

Accepted projects receive:

- **250K to hire Postdoc or PhD student**
- **2.5FTE eScience Research Engineers**

Projects started in 2017

Data mining tools for abrupt climate change

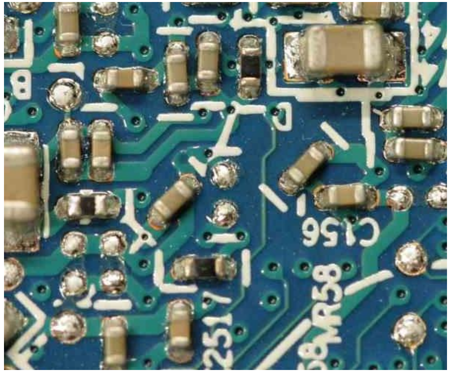
WAGENINGEN UNIVERSITY & RESEARCH

DIRAC - Distributed Radio Astronomical Computing

ASTRON

Accelerating Astronomical Applications 2

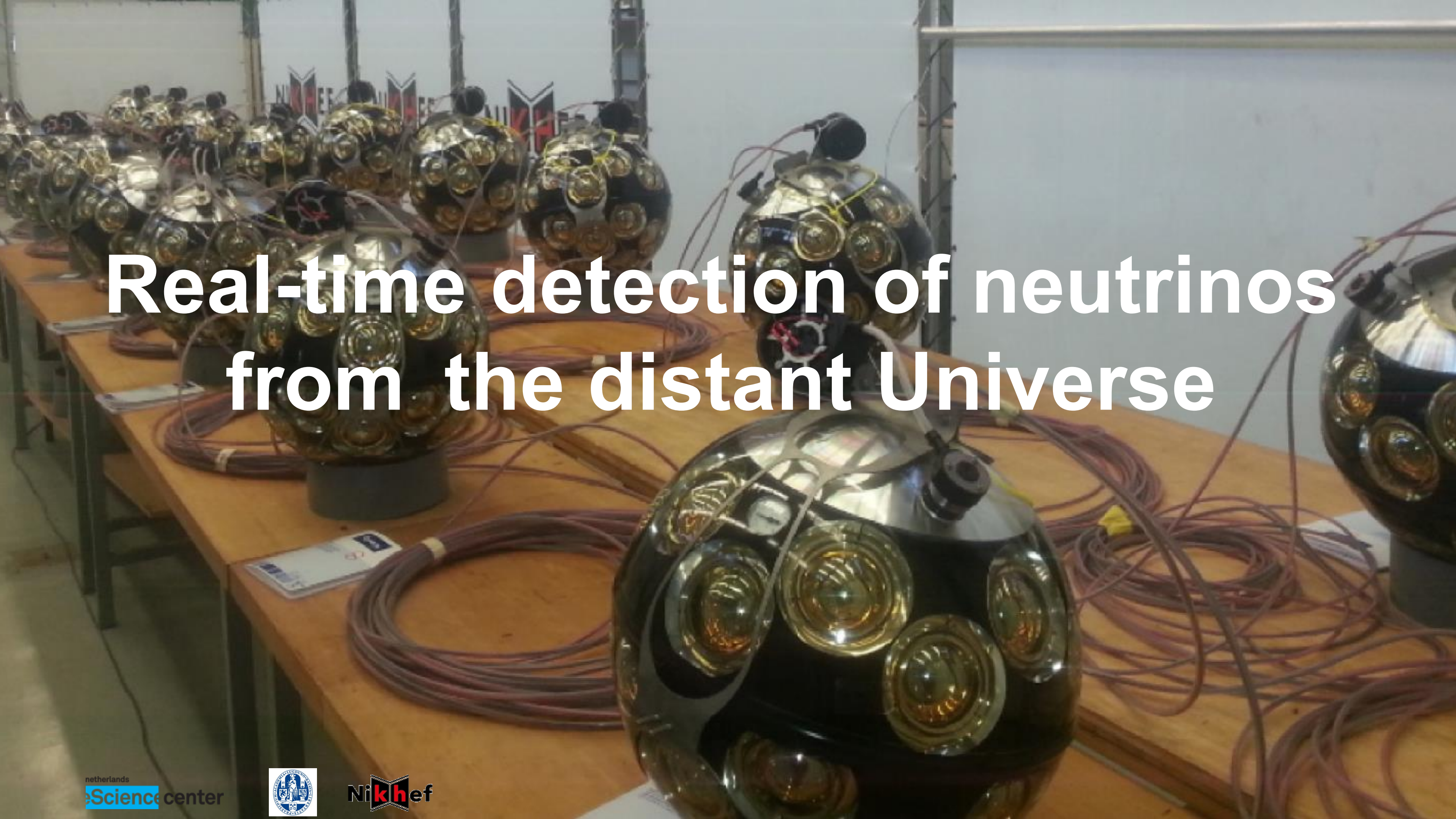
ASTRON



Methodology and ecosystem for many-core programming

VU VRIJE UNIVERSITEIT AMSTERDAM

Real-time detection of neutrinos from the distant Universe

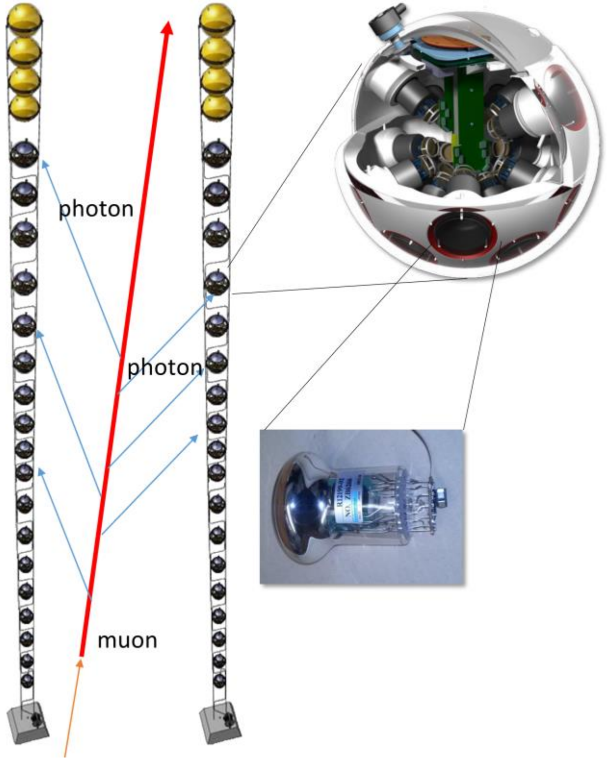


netherlands eScience center

Nikhef

KM3NeT – Neutrino Telescope

- **Huge instrument at the bottom of the Mediterranean Sea**
- **Pretty high data rate due to background noise from bioluminescence and Potassium-40 decay**
- **Current event detection / reconstruction happens on pre-filtered data (so called L1 hits)**
- **Our goal: Work towards event detection based on unfiltered data (so called L0 hits)**

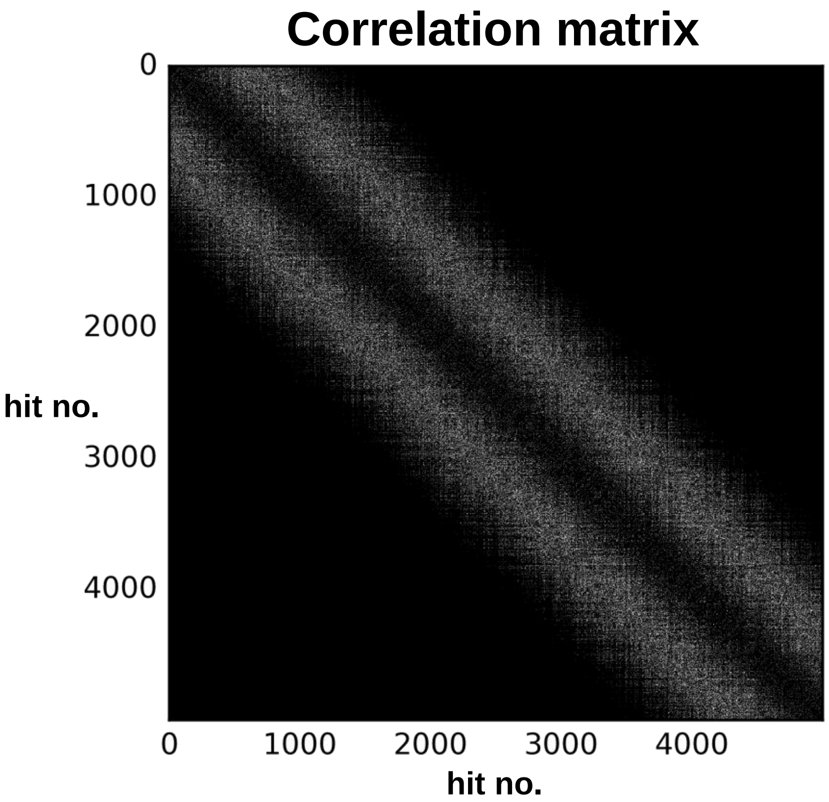


Correlating hits

- Hits are correlated based on their time and location
- Correlations can only occur in a small window of time
- Density of the narrow band depends on correlation criterion in use

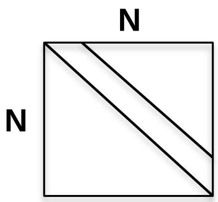
Try-out two designs:

- Dense pipeline that stores the narrow band as a table
- Sparse pipeline that stores the matrix in compressed sparse row (CSR) form

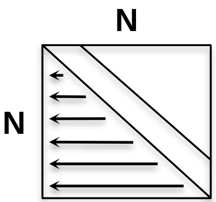


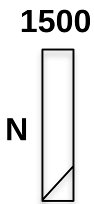
Data representation

- Dense

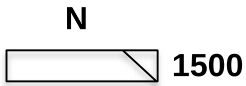


correlation matrix



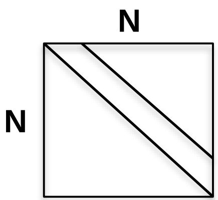


correlations table



on the GPU

- Sparse



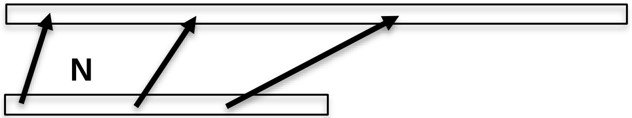
correlation matrix

CSR format

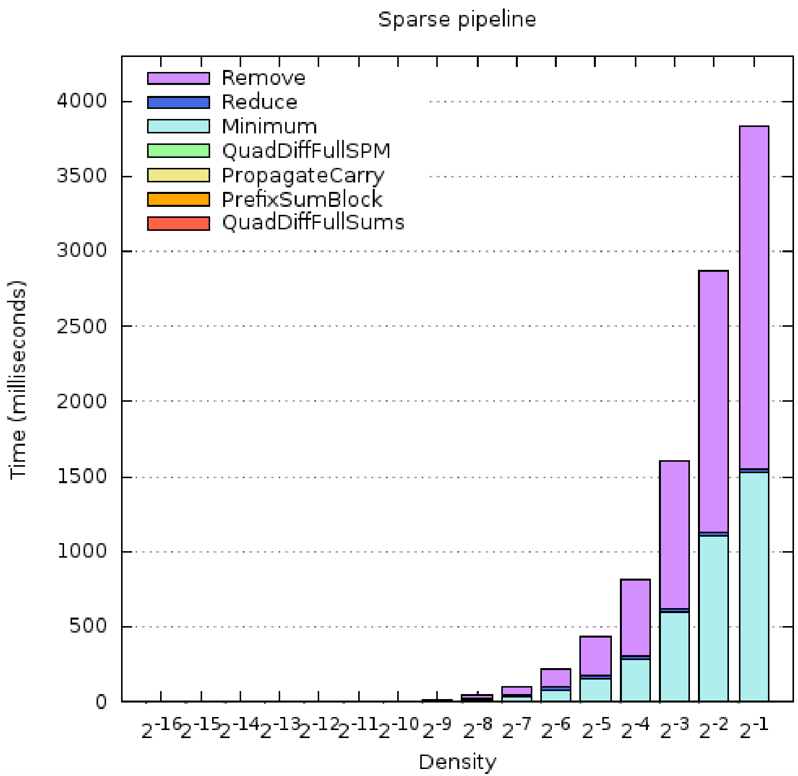
correlations

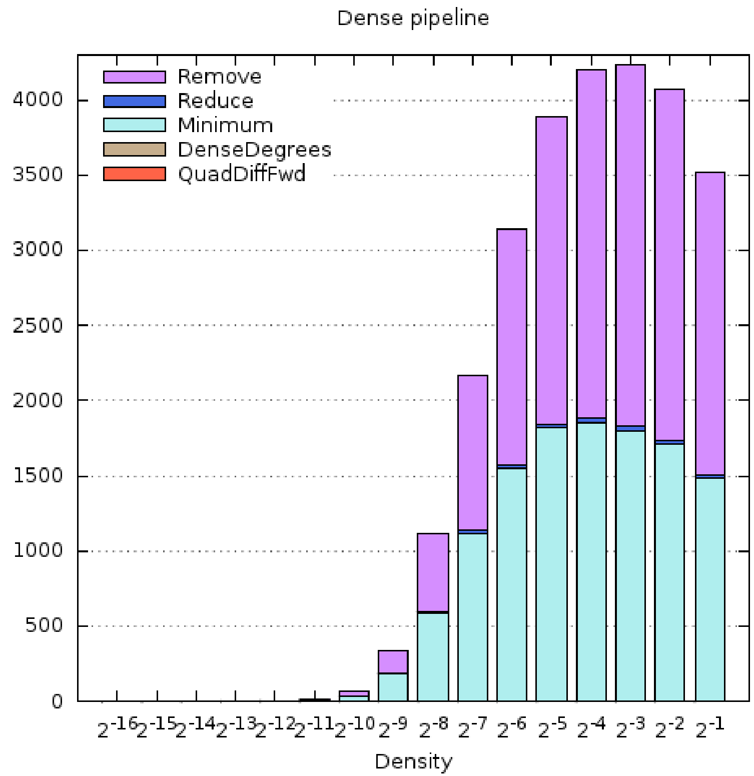
column indices

start of row

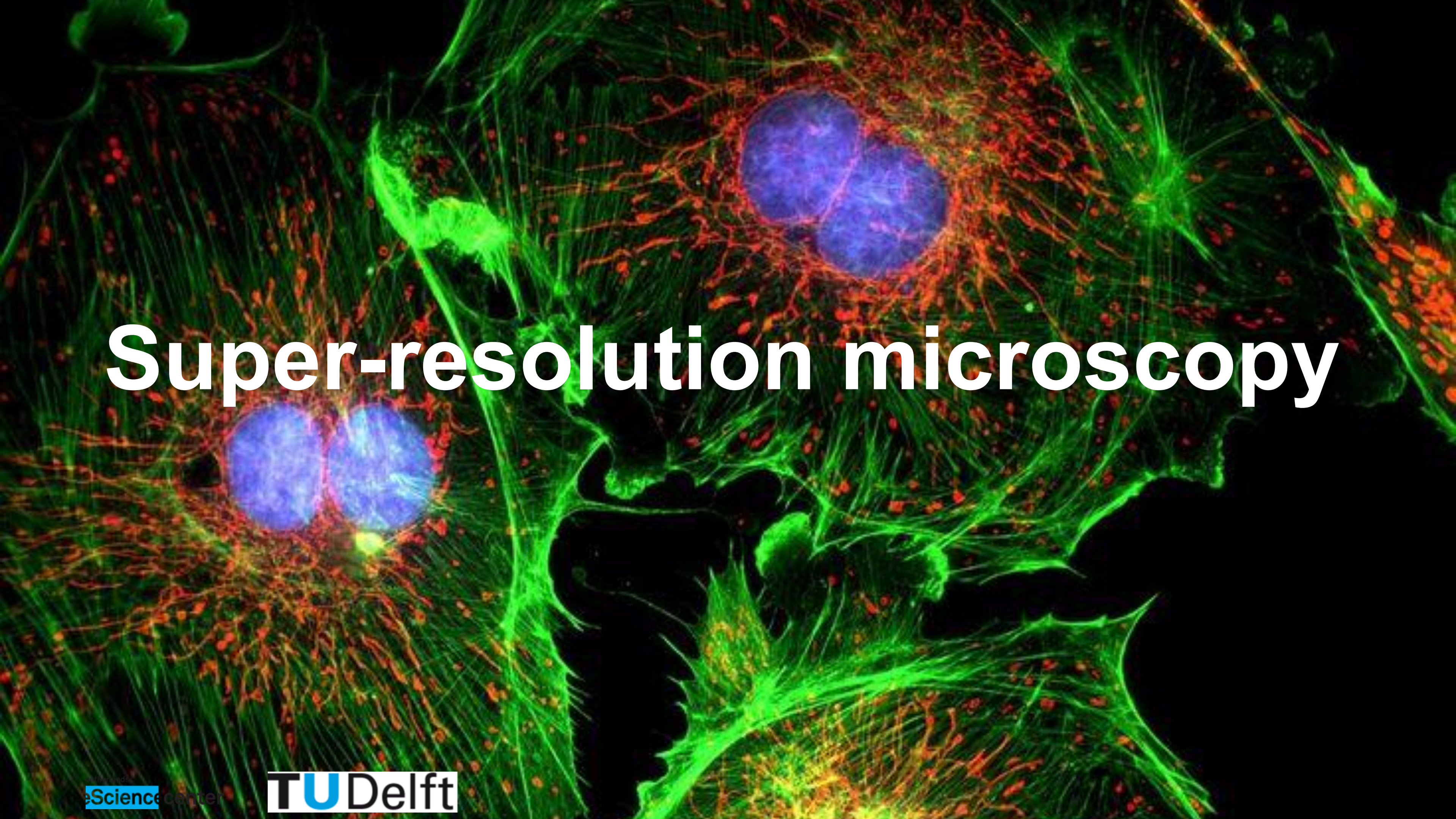


Comparing performance



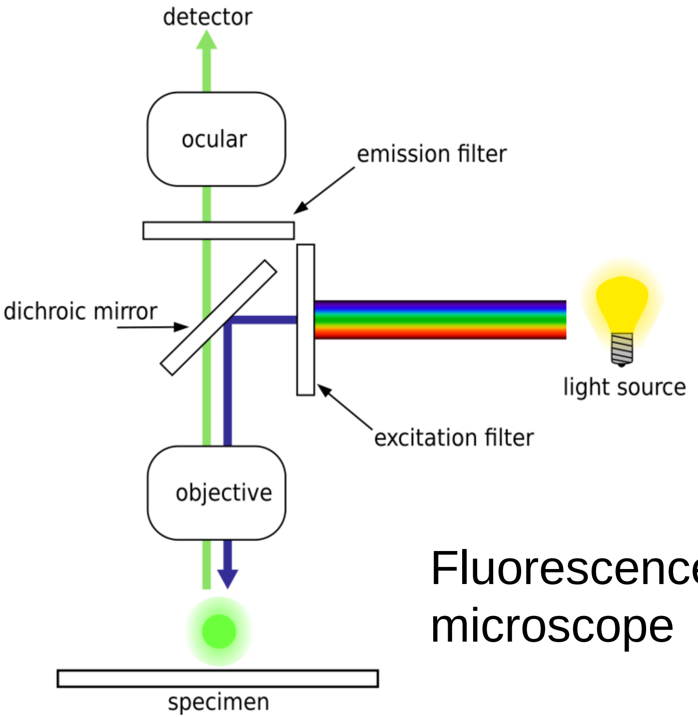


Super-resolution microscopy



Super-resolution microscopy

- Collect a large number of images from fluorescence microscope
- Localize fluorophores using fitting code
- Create single super-resolution image from all localized fluorophores
- Segment all individual molecules in the image
- Create single reconstruction by combining identical copies in the data



Fluorescence microscope

Existing GPU code

- GPU code for maximum likelihood estimation developed in 2009-2010
  - ”Fast, single-molecule localization that achieves theoretically minimum uncertainty” Smith et al. *Nature Methods (2010)*
- Estimates the locations and several other parameters of points in noisy image data for various fitting schemes and pixel area sizes

- State of the code:
  - Each thread worked on exactly one fitting
  - Pixel area analyzed by single thread is 7x7, 19x19, and expected to grow in future
  - Requires many registers and a lot of shared memory per thread block
  - Results in low utilization on modern GPUs
  - Multiple fitting schemes implemented with lots of code duplication

New parallelization

- One fitting is now computed by a whole thread block cooperatively
- Used CUB library for thread block-wide reductions

- Code quality
  - Used function templates to de-duplicate code between different fitting methods
  - Wrote scripts for testing and tuning of device functions and kernels

- Results
  - Currently, speedup of 5.8x to 6.6x over old GPU code on Nvidia GTX Titan X
  - Code can handle arbitrary pixel area per fitting
  - Makes it possible to do termination detection
  - Easier to maintain and extend the code with new fitting schemes

Lessons Learned

Software Engineering Practice

“Throw all good practices out of the window for the sake of high performance”

- Examples:
  - Thousands of code lines in a single function
  - Only acronyms as variable names
  - No comments or external documentation about the code
  - Unnecessary optimization

- Recommendations:
  - Start GPU code from simple code
  - Write and use tests
  - Write C++ and not C, whenever possible
  - Trust the compiler to handle simple stuff

Evaluating results

Results from the CPU and GPU codes are not bit-for-bit the same

- GPUs today implement the IEEE standard just like CPUs
- CPU compilers sometimes more aggressive than GPU compilers
- Fused multiply-add rounds differently
- Floating-point arithmetic is not associative

Things to keep in mind

- It depends on the application whether bit-for-bit difference is a problem
- Testing with random input can give a false sense of correctness

Talking about performance

- Many computer scientists I know think
  - The only way to properly way to discuss GPU performance is to fully optimize and tune for both CPU and GPU
  - Then (and only then) you are allowed to say anything about GPU performance
  - Answering the question: “Which architecture performs the best for this application?”

- Many scientists from others fields that I work with just want to know:
  - “How much faster is that Matlab/Python code I gave you on the GPU?”

Summary

- Choose your starting point carefully
- High-performance and high quality software can co-exist
- Application dependent if small differences in results is a problem
- When talking about performance, be very clear on what is compared to what

www.esciencecenter.nl

Ben van Werkhoven
b.vanwerkhoven@esciencecenter.nl

Project Partners

Nikhef

Nederlands Forensisch Instituut
Ministerie van Veiligheid en Justitie

ACADEMIA LUGDUNO BATAVA · LIBERTATIS PRAESIDIUM · 15 75

VU VRIJE UNIVERSITEIT AMSTERDAM

UNIVERSITY OF AMSTERDAM

ASTRON

TU Delft

WAGENINGEN
UNIVERSITY & RESEARCH