Core Technological Competences

Dr. Jason Maassen
Technology Lead
9/4/2019
Our team’s technological expertise areas

- **Optimized data handling**
  - Data Assimilation and Integration
  - Databases
  - Handling sensor data
  - Linked Data and Semantics
  - Real-time Data Analysis

- **Big Data Analytics**
  - Computer Vision
  - Data Mining
  - Machine Learning
  - Natural Language Processing
  - Statistics
  - Visualization

- **Efficient Computing**
  - Accelerators
  - Distributed Computing
  - High Performance Computing
  - Numerical Modelling and Algorithms
  - Workflows and Orchestration
Our team’s technological expertise areas

- **Optimized data handling**
  - Data Assimilation and Integration
  - Databases
  - Handling sensor data
  - Linked Data and Semantics
  - Real-time Data Analysis

- **Big Data Analytics**
  - Computer Vision
  - Data Mining
  - Machine Learning
  - Natural Language Processing
  - Statistics
  - Visualization

- **Efficient Computing**
  - Accelerators
  - Distributed Computing
  - High Performance Computing
  - Numerical Modelling and Algorithms
  - Workflows and Orchestration

---

Research Software Development
Expertise areas

• Expertise areas typically evolve faster than the broad competence areas

• They are driven by:
  • demand from the current projects
  • latest developments in technology
Specialization

- Focus on expertise relevant to our projects:
  - Impact on research
  - Mature enough to be applied in research
  - Preference for reusable technologies

- Specialization is vital for us to be a valuable addition to research teams
Software quality

Software Sustainability Protocol

October 8, 2018

Netherlands eScience Center Software Sustainability Protocol

Speaker: Jansen, Mark J.; Meeussen, James

Today, software plays a crucial role in advancing and accelerating state-of-the-art academic research. It is therefore important to adhere to proven best practices when developing research software, as it will help avoid errors, improve maintenance and sustainability, while accelerating the overall development process. Also, it will help other researchers to better understand the intricacies of the software, and the analyses performed within. This is an important prerequisite for reproducibility of scientific results, and will allow other researchers to adapt the software into their own workflows, possibly even contributing to the software or expanding it.

To promote the practice of Open Science, and the principles of FAIR data and software, the Netherlands eScience Center actively participates in the National Platform Open Science. In line with this initiative, the eScience Center strives to make the software developed in its projects available to become freely and sustainably available, as much as possible, for reuse by other researchers. To this end, the Principal Investigator of every project (co)funded by the eScience Center is required to submit a Software Sustainability Plan (SSP), i.e. a document that makes explicit the researcher’s thirteen actions regarding software sustainability.

Software Development Guide

This is a guide to software development and projects at the Netherlands eScience Center. It both serves as a source of information for exactly how we work at the eScience Center, and as a basis for discussions and reaching consensus on this topic.

This Guide is a work in progress.

The source of this book can be found on GitHub:
https://www.github.com/nlesc/guide
Research Software Directory
*a unique resource for sharing research software*

FAIR software:
- Finding software
- Making software accessible
- Quickly judge relevance and quality

[www.research-software.nl](http://www.research-software.nl)
Googling the cancer genome: Identification of cancer-causing structural variations in whole-genome sequencing data

UMC Utrecht team
Jeroen de Ridder (PI)
Luca Santuari
Tilman Schaefers
Wigard Kloosterman (co-PI)

eScience team
Sonja Georgievksa
Arnold Kuzniar
Stefan Verhoeven
Jason Maassen
Lars Ridder (coordinator)
The genomics era

Human genome
- 23 chromosome pairs
- 3.2 billion base pairs
- ~25,000 genes
- ~350GB data / patient

Whole genome sequencing
short DNA reads
(Illumina)
The grand question

Can we identify cancer-causing mutations from a patient's DNA sequence?
Genome sequencing after shredding DNA into millions of short sequences or 'reads'

Structural variants
- affect long stretches of DNA
- implicated in cancer
Challenge:
Reliable detection of structural variants (SVs) in massive genomic data

Why is this a challenge?
• Existing tools give widely varying results – which ones are true?
• ... depend on manual parameter settings – how to reproduce results?
• ... computationally prohibitive for thousands of genomes – how to scale up / reduce costs?

Our solution:
• Combine state-of-the-art SV detection tools in an efficient workflow that runs anywhere
• Develop a novel SV detection tool based on machine learning
sv-callers | comprehensive detection of structural variants in cancer genomes

- highly portable parallel workflow
- **reproducible** state-of-the-art SV analyses
- easy to use, extend and deploy
- extensively tested and production ready
DeepSV | deep learning-based detection of structural variants

- Applying machine learning non-trivial

<table>
<thead>
<tr>
<th>DeepSV</th>
<th>DeepVariant (Google)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection</td>
<td>Structural Variants</td>
</tr>
<tr>
<td>Input data</td>
<td>Multi-channels</td>
</tr>
<tr>
<td>Parameters</td>
<td>10K</td>
</tr>
<tr>
<td>Comp. resource</td>
<td>Laptop</td>
</tr>
</tbody>
</table>

Sequence alignment

- coverage
- discordant read pairs
- clipped reads

Multi-channel series

- breakpoint

Convolutional neural network

Variant type

- deletion
- insertion
- inversion
- duplication
- translocation
Scientific workflow

*sv-callers* using *Snakemake* (CWL)

Compute & storage middleware

*XENON*

Deep learning

*DeepSV* using *mcfly* (time-series)

Package & environment management

*CONDA*, *S*, *UCL*
Example 2
Real-time detection of neutrinos from the distant Universe

Leiden University / Nikhef
Dorothea Samtleben (PI)
Ronald Bruijn

eScience team
Hanno Spreeuw
Ben van Werkhoven
Cosmic rays interact with atmosphere
  => Showers, muons, neutrinos

Neutrinos arrive from astrophysical sources

Neutrino interaction in Earth
  => Muon passes detector

Cherenkov light from muon

3D PMT array
Real-time detection of neutrinos from the distant Universe

KM3NeT neutrino detector: 
12,000 spheres with 31 photo detectors on 600 strings in the Mediterranean. Process 55 GB/s in real time to remove interference from potassium decay and bio-luminescence. Produce output when a hit is likely (22 MB/s).
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
Quadratic difference kernel

- Input: N hits denoted as 4 floats (x,y,z,ct)
- Given hits $i$ and $j$ it computes $(ct_i - ct_j)^2 < (x_i - x_j)^2 \times (y_i - y_j)^2 \times (z_i - z_j)^2$
- Output: correlations table of size $N \times \text{window}_\text{width} (=1500)$

![diagram showing correlation matrix and table]
Using sparse matrix notation

Motivation

• The expected density of correlated hits is rather low (0.2%)
• Storing the correlations table currently takes up $N \times 1500$ bytes
  – for 4.5 million hits that is 6.75 gigabytes
  – at 0.2% density this table contains 13.5 MB of 1s and more than 6 GB of 0s
• Storing only the locations of the non zero values takes up $(N + NNZ) \times 4$ bytes
  – for 4.5 million hits and 0.2% density the CSR format requires about 72 MB

Advantage

  – Reduces the data kernels need to load/store, reducing the memory bandwidth consumption

Disadvantage

  – Uses more random access, which may hurt memory throughput (bad on GPUs!)
Synchronization problem

- On a GPU it is simply not possible to keep a single global counter and increment it every time a thread needs to write a value.

- Two solutions are possible:
  - 1) Run the correlation kernel twice, first only to determine the number of outputs per hit, secondly to actually store the results.
  - 2) Run the correlation kernel once, temporarily store the dense data and the number of hits, read in the dense data to actually store the results.

- Started with option 2, because correlation kernel seemed expensive. Option 1 is actually faster because reading 6 GBs takes a lot of time.
Pipeline performance

Sparse pipeline

Dense pipeline
Optimizing GPU code is hard!

Requires that you get all the details exactly right:

- Mapping of the problem to threads and thread blocks
- Thread block dimensions
- Data layouts in the different memories
- Tiling factors
- Loop unrolling factors
- How to overlap computation and communication
- ...

Problem:
Creates a very large and discontinuous search space
Tuning GPU performance

2D Convolution on GTX Titan X (Maxwell)
Kernel Tuner

A generic OpenCL, CUDA, and C kernel auto-tuner in Python

Easy to use:
- Directly on existing kernels and code generators
- Inserts no dependencies in the kernels
- Kernels can still be compiled with regular compilers

Supporting:
- Large number of effective search optimizing algorithms
- Output verification for auto-tuned kernels and pipelines
- Tuning parameters in both host and device
- Python-based unit testing of GPU code

https://research-software.nl/software/kernel_tuner/
Scalable machine learning & AI  
(see genomics example)

Processing of streaming data  
(see neutrino detector example)

Large scale (distributed) data organization, management & semantics  
(FAIR data, FAIR software, semantic workflows, etc.)