Challenges in Evolving Software for Cryo-Electron Microscopy:
From CPUs to GPUs and Back Again

Or: Software engineering in a more challenging situation

Erik Lindahl
Stockholm University & KTH Royal Institute of Technology
2017 NOBEL PRIZE IN CHEMISTRY
Jacques Dubochet
Joachim Frank
Richard Henderson

“For the greatest benefit to mankind”
Alfred Nobel
X-Ray Crystallography

Interference

Crystal

X-ray

X-Ray Crystallography

Cryo-Electron Microscopy

Diffraction detector

Sample pattern

Scattered X-rays

X-ray beam

Frozen protein sample

Lens

Electron detector

Electron beam

Nature Sep 9, 2015: The Revolution will not be crystallized
Talos Arctica (Falcon II/III)
Titan Krios
(Energy filter + Falcon II/III+K2)
Volta phase plate
Fully automated collection
Avoid the not-invented-here syndrome: When we find & use awesome software, contribute to it instead of competing.

Possible because RELION builds on XMIPP, which was licensed under GPLv2 - so it’s open.
A Bayesian approach to structure determination

\[ p(A | B)p(B) = p(B | A)p(A) \]

\[ p(A | B) = \frac{p(B | A)p(A)}{p(B)} \]

This way, we can include all microscope CTFs, noise estimates and possible orientational bias in the model, as well as the assumption of smoothly varying electron density.

You need a starting model.
Parallelization

We scan many independent
- Views (100,000s)
- Objects (10)
- Pixels (0.25 Mpix)
<table>
<thead>
<tr>
<th>Acceleration Approaches</th>
<th>GPU libraries</th>
<th>OpenAcc</th>
<th>Pure CUDA</th>
<th>Heterogeneous CPU/GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial effort / Expertise req.</td>
<td>Works if your code offloads to libraries</td>
<td>Always works, but success depends on you &amp; compiler</td>
<td>Lots of work, assumes impl. can run entirely on GPU</td>
<td>Even more work, less CUDA, can use both CPU &amp; GPU</td>
</tr>
<tr>
<td>Generality / Portability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code maintainability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
First lessons with a new code

• Amazing science

• High-impact program, but not good code (understatement…)

• Collaboration between different strong-minded PIs
  • No longer just scientific computing, but high-impact biology
  • Competition: cannot afford to wait 6M to improve code design

• Conflict: Science goals do not align to 100% with general open source projects developed for fun

• You need a very tasty carrot to convince people to change
You need long-term views

• Previous acceleration attempts (Xeon Phi) had only tried code & compiler tweaks, and achieved 20% speedup

• We reworked from the ground up instead, with the aim of being at least 10x faster

• Change algorithms, change data representation, change flows

• New manually developed CUDA memory allocators

• Aim for a high-impact publication, so students get credit

• Involving NVIDIA meant we got a lot of free PR for the work
Solution: With 10-50x acceleration, suddenly the community is requesting that new code is implemented more rapidly.
Challenging but fruitful collaborations

• Sjors Scheres - original RELION group in Cambridge
  Deep expertise on the reconstruction algorithms & RELION

• Erik Lindahl, Dari Kimanius, Björn Forsberg - Stockholm
  Scientific computing & GPU acceleration skills, GROMACS

• Mark Berger & Nikolay Markovskiy - NVIDIA
  GPU expertise, limited cryo-EM experience

• Users - sometimes close to completely ignorant about computing
  “Why doesn’t the code work on my 10-year-old RedHat installation?”
  “Is there a Windows version?”
Software Engineering version 0.1

• ~0% unit test coverage
• No continuous integration
• A few manual regression tests
• Code documentation limited
• Problematic usage of FFTW, C++, memory not released
• Extreme requirements on producing correct results
• We convinced people to switch to CMake
• Discussions about using single precision, which was eventually OK
• Not yet agreement on using C++11
Refinement quality

Plasmodium 80S subunit
New opportunities

• Based on the GPU acceleration, Intel got extremely interested in helping with CPU acceleration

• New collaboration with Intel. They contributed a skilled engineer (Charles Congdon), but who had no experience whatsoever from structural biology or the code base, and who wanted to backport our CUDA algorithms to work well on the CPU

• More decoupled development model than GROMACS:
  • Separate git repos for Stockholm, LMB Cambridge, NVIDIA, Intel
  • Use pull requests to merge codes (similar to Linux)
  • Much more fragile main development branch
    Some friction between developers when bugs are introduced by mistake - comes with the job
  • But: Very high scientific impact - multiple eLife publications from software!
Focus on algorithms, not tuning

It is critical that you understand and re-implement the original physical model algorithm, rather than just move around or tune code. Being able to stream through data is critical!
movaps xmm0, xmm4
movaps xmm1, xmm4

shufps xmm2, xmm6, 0b10001000
shufps xmm0, xmm5, 0b10001000
shufps xmm1, xmm5, 0b11011101

/* move ix0-iz0 to xmm4-xmm6 */
movaps xmm4, [esp + _ix0]
movaps xmm5, [esp + _iy0]
movaps xmm6, [esp + _iz0]

/* calc dr */
subps xmm4, xmm0
subps xmm5, xmm1
subps xmm6, xmm2

/* store dr */
movaps [esp + _dx0], xmm4
Lines of raw SIMD code

GROMACS

~600,000

Used to be raw assembly, but now intrinsics

RELION

0

• It is extremely likely that we could get even better speed-ups in RELION with manual tweaking, but if the compiler performs this well with only hints, why bother?

• There is a huge advantage in only having a single code path. All modifications introduced by anybody will now be x86-accelerated, even if they do not understand SIMD
Good Kernels Express Parallelism, not Architecture

template <typename T>
__global__ void cuda_translate3D(T * g_image_in,
T * g_image_out,
int image_size,
int xdim,
int ydim,
int zdim,
int dx,
int dy,
int dz)
{
    int tid = threadIdx.x;
    int bid = blockIdx.x;

    int x,y,z,xp,yp,zp,xy;
    int voxel=tid + bid*BLOCK_SIZE;
    int new_voxel;

    int xdim = xdim*ydim;

    if(voxel<image_size)
    {
        z = voxel / xdim;
        zp = z + dz;
        xy = voxel % xdim;
        y = xy / xdim;
        yp = y + dy;
        x = xy % xdim;
        xp = x + dx;

        if( zp>=0 && yp>=0 && xp>=0 && zp<zdim && yp<ydim && xp<xdim)
        {
            new_voxel = zp * xdim + yp * xdim + xp;
            if(new_voxel >= 0 && new_voxel < image_size)
                g_image_out[new_voxel] = g_image_in[voxel];
        }
    }
}
Single precision

- Most modern CPUs provide *twice* the throughput in single compared to double.
- This also saves cache, memory bandwidth – and RELION is quite memory-hungry.
- **Do you need more than 7 valid digits in the output?**
- Converting scientific code to single-precision is not trivial: Most codes fail if you just do search & replace, but it can be made to work:
  - Identify sensitive parts (maximization step), and leave those in double.
  - Only perform a few operations (exponentials) in double.
  - Sum small numbers first, or use tree summation instead of linear order.
  - Use *strength-reduction algorithms* (check open source math libraries).
  - Beware of double-single-double conversions inside critical code paths.
Some development lessons we want to bring back to GROMACS

- There is a large scientific value in rapidly getting stuff out
- Large value in allowing non-programmers to contribute ideas
- Both of these goals are partially in conflict with good software engineering - and we might have to live with that as a compromise
- Accept that not everyone is focused on scientific computing
- Don’t aim for perfection, but aim for a project where other people are happy contributing to your project - even if it means occasional bugs
- For multiple separate teams with different goals/competence, the Linux development model with PRs has advantages over a central repository
The Social Dimension to Engineering

• Scientists get credit for publishing - in particular as first/last author
• You can get a lot of impact from developing widely used software
  • Leads to strong incentive to fork & create your own program
• We need to find better ways to reward people from contributing to someone else’s code instead of constantly reinventing the wheel
  • This must start with existing scientific computing groups - start contributing to other codes!
• Make sure people (in particular students) who contribute to your code get credit - Sjors Scheres did a great job at pulling us into RELION
GROMACS: Berk Hess, Szilard Pall, Mark Abraham, Paul Bauer, Aleksei Iiupinov, John Eblen, Roland Shultz, Christian Wennberg, Viveca Lindahl
RELION: Dari Kimanius, Björn Forsberg, Sjors Scheres, Alexey Amunts, Marta Carroni, Shintaro Aibara
NVIDIA: Mark Berger, Duncan Poole, Julia Levites, Jiri Kraus, Nikolay Markovskiy
INTEL: Charles Congdon, Sheng Fu, Kristina Kermanshahche, Yuping Zhao
CSCS: Thomas Schulthess, Victor Holanda, Prashant Kanduri
PDC: Erwin Laure