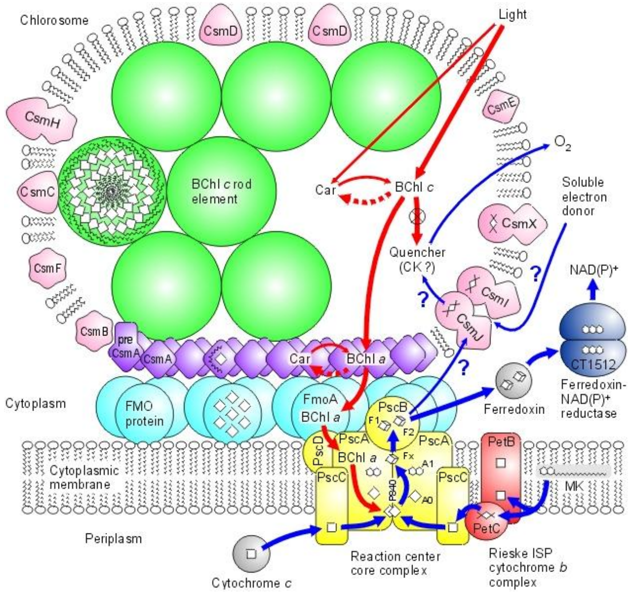
Hierarchical Equations of Motion:
~~- OpenCL on the Xeon Phi –~~
- What we can learn from OpenCL -

Matthias Noack

Zuse Institute Berlin (ZIB)
Germany

What's unique about my tuning work

- **HEOM (Hierarchical Equations Of Motion)**
  - Dr. Tobias Kramer, Dr. Christoph Kreisbeck
- **Simulation of energy transport in biological and artificial light harvesting complexes**
- **Domain:** where quantum physics meets biology
- **Execution mode:** OpenCL, native
- **Tools:**
  - OpenCL SDK
  - Intel Composer
  - Vtune
  - Manual assembler analysis



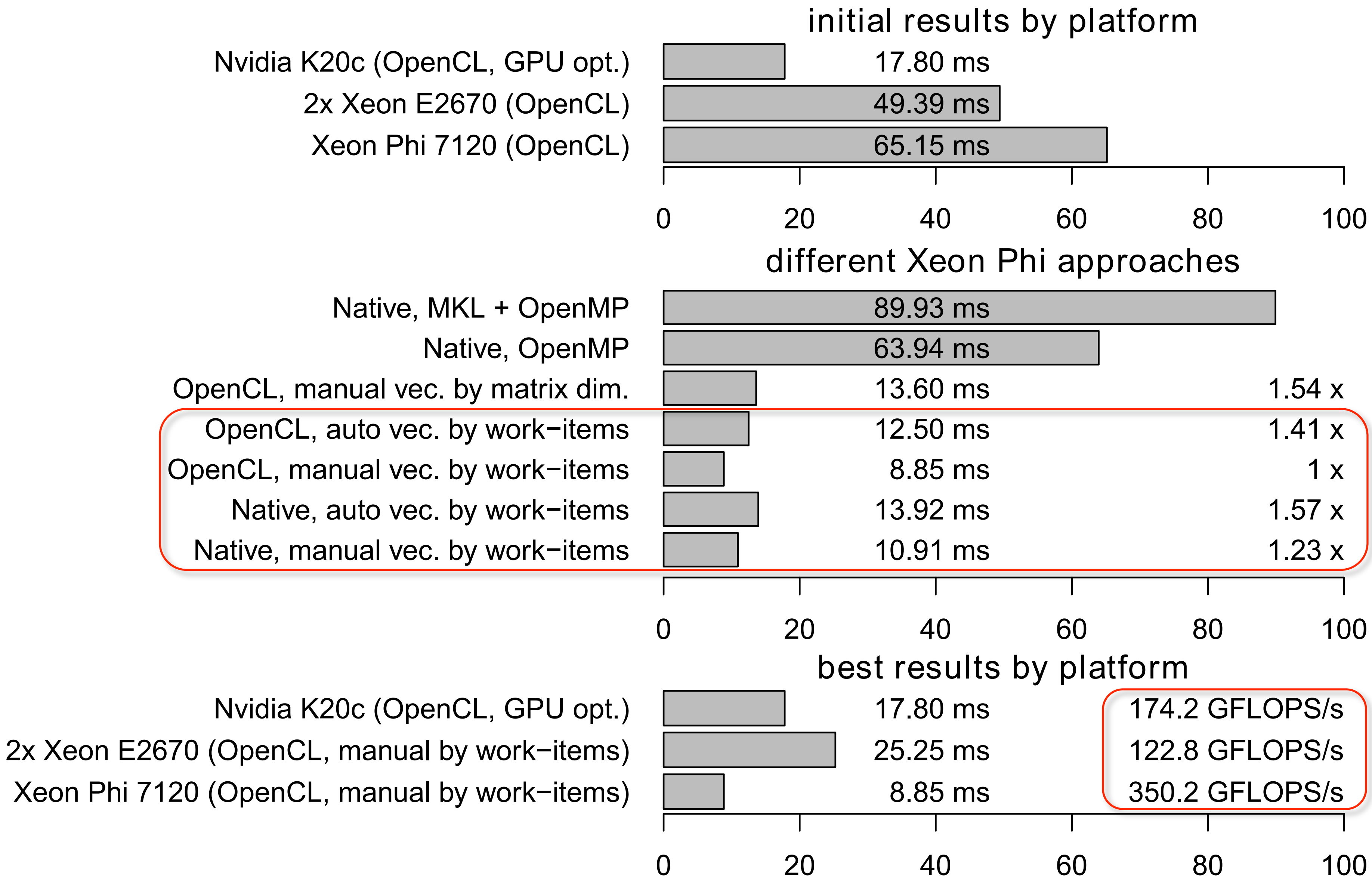
- **State:** 1 of 4 kernels tuned for the Xeon Phi

Performance

- **List of OpenCL optimisations**
  - Vectorisation-friendly memory layout (AoSoA) with automatic vectorisation (~2.1x)
  - Manual vectorisation (additional ~3.0x)
  - Index calculations using macros (additional ~1.1x)
  - Manual prefetching (additional ~1.1x)
  - Compile-time matrix dimension in loops and index calculations (difference of ~2.6x for the best optimised kernel)

- **Overall OpenCL tuning result:**
  - Xeon Phi performance improved by ~7.3x
  - Host performance improved by ~2.6x
  - Xeon Phi vs. GPU-optimised kernel on K20c: ~2.0x

Performance (Hexciton Kernel Runtime)



Insights

- **Try an OpenCL-like pattern for your kernels**
  - Decompose problem into work-items
  - Parallel loop over work-groups (SIMD-width items per group)
  - SIMD loop over work-items in a group
  - Recompile kernels with constants from input (JIT would be ideal)
- **Change memory layout (AoSoA) for contiguous vector loads**
  - Use macros for complex index computations
- **Try manual vectorisation over "work-items"**
  - No SIMD loop
  - Replace: `double` ⇒ `double_vec` (Vc, vectorclass, micvec.h, …)
- **Challenge:**
  - OpenCL compiler still generates faster code
  - C-Compiler needs help for this pattern: #pragma (no)unroll, (no)vector, ivdep; and manual loop-permutation