Accelerating Many-Body Potentials with GPUs

William French

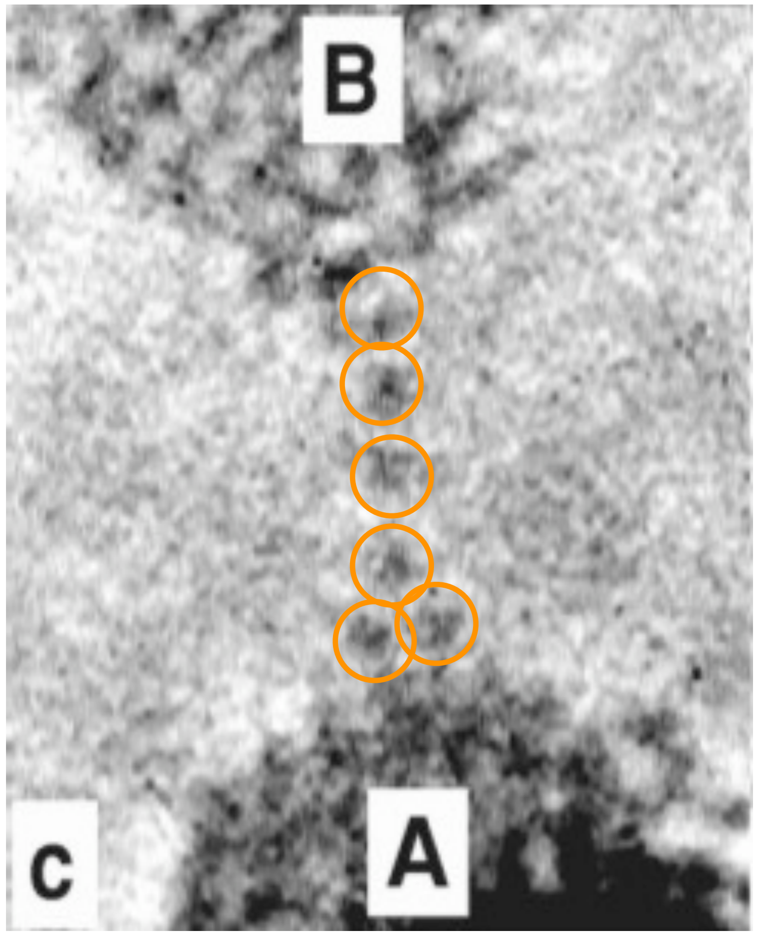
Vanderbilt University

Department of Chemical and Biomolecular Engineering

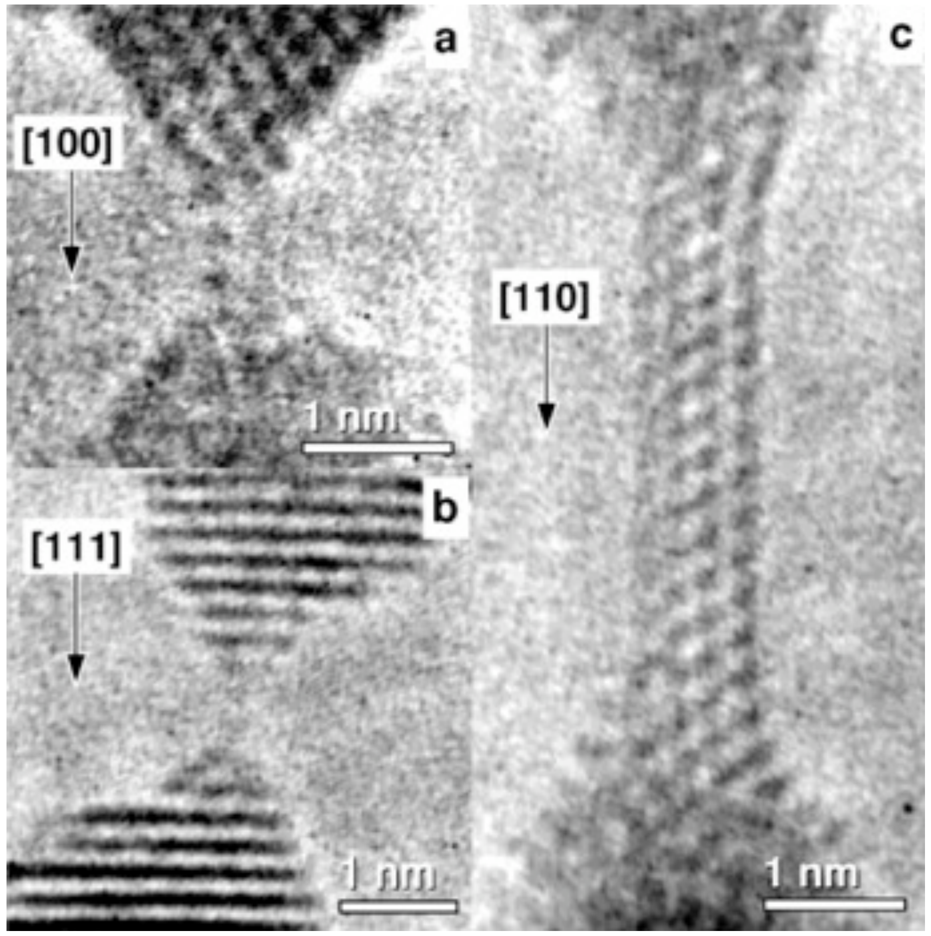
Research Adviser: Peter Cummings

Background

Gold Nanowire Elongation

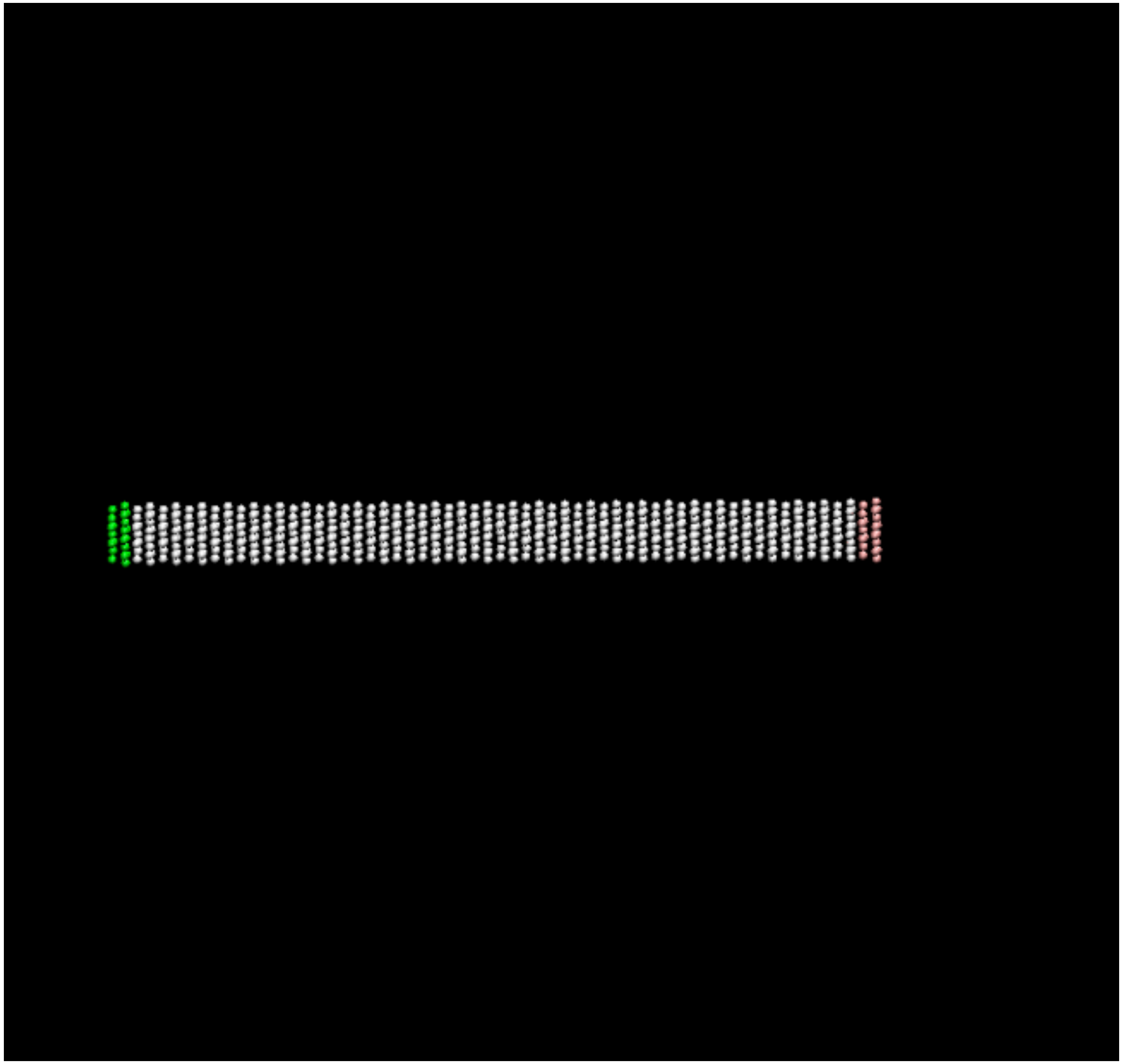


Rodrigues and Ugarte, Phys. Rev. B 63, 2001



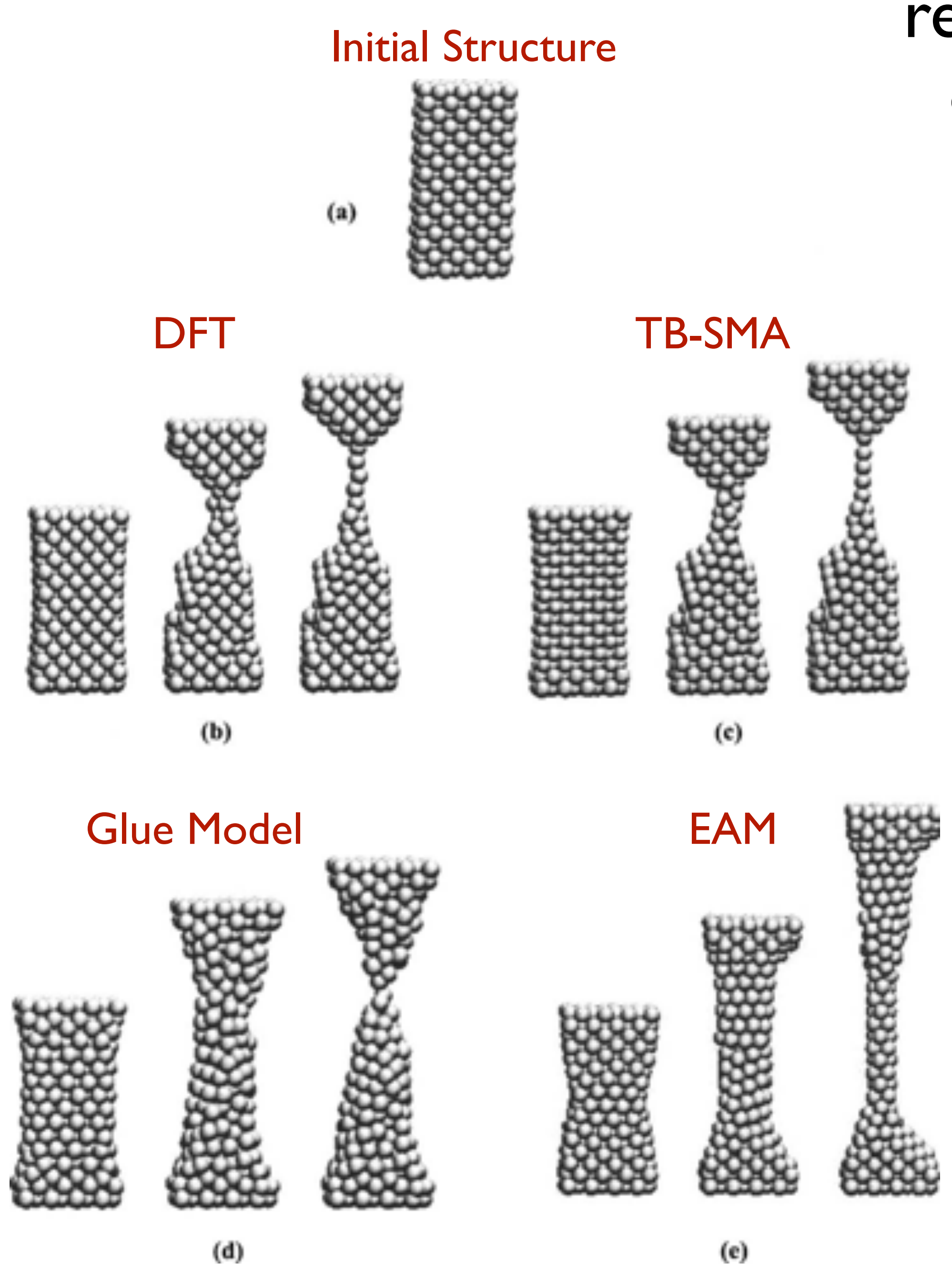
Coura, et al. Nano Letters, 4 (7), 2004

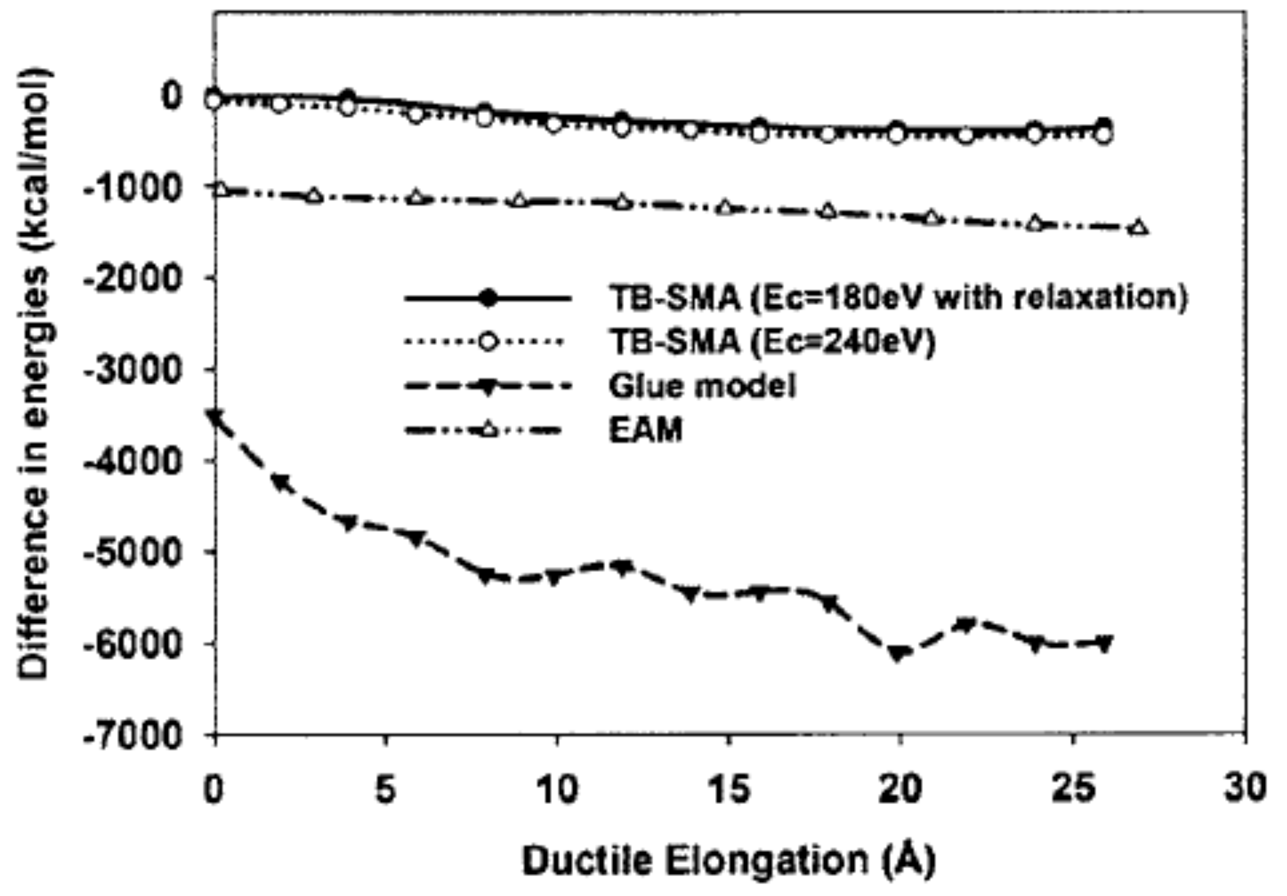
*Single-atom chains and helical ribbons observed



Gold Potential: TB-SMA

*A comparative study conducted by our group has revealed the accuracy of the TB-SMA potential versus other popular semi-empirical metallic potentials in describing Au-Au interactions in elongated Au nanowires





Pu *et al.*, *Journal of Chemical Physics*, 126, 144707 (2007).

Second-Moment Approximation of the Tight-Binding Potential (TB-SMA)

$$E_B^i = -\left\{\sum_j \xi_{\alpha\beta}^2 e^{-2q_{\alpha\beta}(r_{ij}/r_0^{\alpha\beta}-1)}\right\}^{1/2}$$

*Many-Body Term

$$E_R^i = \sum_j A_{\alpha\beta} e^{-p_{\alpha\beta}(r_{ij}/r_0^{\alpha\beta}-1)}$$

*Pairwise Repulsive Term

$$E_c = \sum_i (E_R^i + E_B^i)$$

*Total Potential

F. CLERI and V. ROSATO, "TIGHT-BINDING POTENTIALS FOR TRANSITION-METALS AND ALLOYS," *PHYSICAL REVIEW B*, vol. 48, pp. 22–33, JUL 1 1993.

TB-SMA Potential for GPUs

Goals for GPU Implementation

- Implement the TB-SMA potential for classical molecular dynamics simulations
  - ❖ Preferably within the framework of an open-source package (e.g. NAMD, LAMMPS, HOOMD)
- Achieve large speedups

Impact of Work

- Research contribution:
  - ❖ Make large problems more tractable
  - ❖ More closely simulate relevant length/time scales of real systems
- Contribution to body of GPU software users:
  - ❖ Continue the extension of MD codes to GPU-based architectures

Key Algorithms in CPU Approach

Neighbor List Routine

```
for (i=0;i<natoms;i++) {
    for (j=i+1;j<natoms;j++) {
        ...
        ...
        if (rij < r_neigh_cut) {
            add neighbor to list
        }
    }
}
```

Force Computation

```
for (i=0;i<natoms;i++) {
    for (j=0;j<n_neigh;j++) {
        ...
        ...
        if (rij < rcut) {
            U = U(rij)
            F = F(rij)
        }
    }
}
```

Available Parallelism

- N-body problem; well-suited for parallelism
- Populate neighbor list and compute force on each atom
    - ❖ Sub-divide the loops amongst a large number of threads to do work on independent parts of the loop simultaneously

Challenges

- Lack of CUDA programming experience
- Minimizing host-device and device-host data transfer
- Data structuring to optimize performance

Questions?

(Or Suggestions?)