Strategies for Effective Offloading in GROMACS

Dr. John D. Eblen
Center for Molecular Biophysics
Oak Ridge National Laboratory/University of Tennessee, Knoxville

Offloading Overview

- **GROMACS - Molecular Dynamics**
- **Developed with Eclipse PTP Synchronized Projects (full disclosure: I am a developer for PTP)**
- **Overall Strategy**
  - Offload nonbonded-force calculations - roughly 30-40% of compute time
  - Asynchronously run bonded-force calculations on host
- **Challenge - Minimize Offload Overhead**
  - Minimize amount of data transfer and number of transfers
  - Minimize memory allocation and management
  - Avoid heavy use of hidden offload mappings

Minimizing Data Transfer

- **Serialize data - avoid multiple transfers even in same offload**
- **Minimize amount of data transferred**
- **3 types of data**
  - Volatile: Update every offload
  - Periodic: Update occasionally
  - Constant: Never update

Data Type	Size	Offload Overhead Time
Volatile	382 KB	4.28 ms
Periodic (every 5th time step)	4.80 MB	8.02 ms

Minimizing Memory Overhead

- **Allocations on coprocessor can degrade performance**
  - Avoid allocating during offload
  - Allocate buffers separately and only once if possible
  - Avoid using hidden offload table for memory mappings
- **Mysterious slowdowns occurred when using offload tables**
  - Failure Case 1: Automatically mirror all (most) allocations
  - Failure Case 2: Allocate a new buffer prior to offload
  - Failure Case 3: Use internal mapping instead of pre-allocate option
- **Failure case 2 is root cause?**

Data Type	Reuse Buffer Overhead Time	New Buffer Overhead Time
Volatile	4.28 ms	9.71 ms
Periodic (every 5th time step)	8.02 ms	16.32 ms

Final Offload Pragma

```
#pragma offload target(mic:0) \
    nocopy(nbl_lists) \
    nocopy(nbl_buffer) \
    nocopy(ci_buffer) \
    nocopy(sci_buffer) \
    nocopy(cj_buffer) \
    nocopy(cj4_buffer) \
    nocopy(type_buffer) \
    nocopy(lj_comb_buffer) \
    nocopy(q_buffer) \
    nocopy(phi_buffer_sizes) \
    in (cpu_out_packet[0:packet_in_size] :  into(phi_in_packet[0:packet_in_size]) REUSE targetptr) \
    out(phi_out_packet[0:packet_out_size] : into(cpu_in_packet[0:packet_out_size]) REUSE targetptr) \
    signal(&off_signal)
```

Insights

- **Benchmarking is essential**
- **Code design is key**
  - Large data structures were the biggest problem
  - Other problem was built-in assumptions about architecture
  - Minimize disruption by separating out offload code
  - Suggestion: Design with coprocessor interface and multiple architectures in mind
- **Recommendations**
  - Benchmarking is essential. GROMACS has built-in timing library so it is easy to measure sections of code
  - Simplify - avoid heavy use of offload API
  - More transparency needed for offload API
- **Remaining problems and current work**
  - Offload overhead still a bit high
  - Work balancing can be improved
  - Extending to multiple coprocessors