HiHAT Proof of Concept Update
Hierarchical Heterogeneous Asynchronous Tasking

Millad Ghane, John Stone, CJ Newburn

Version 17071100

OUTLINE

- Goals
- Current infrastructure capabilities
- Microbenchmark overheads
- Molecular Orbitals app
- Portability

GOALS OF POC

- Drive and demonstrate tangible progress
- Offer an initial strawman design to iterate and get feedback on
- Demonstrate feasibility of low overheads, good absolute performance
- Identify key opportunities, e.g. ease of use wrt API

CURRENT INFRASTRUCTURE CAPABILITIES

The basics are already working

- Current test platform: 2 CPU sockets + 2 GPUs in one node
- Data movement
  - User Layer: <dst, src, size> addresses without memory resource association
  - Common Layer: add memory resources associated with addresses
  - Set up comms, establish visibility as needed
- Data management
  - User Layer: Allocate or wrap, and create address-memory resource association
    - Also support tagging to clean up a set of allocations/wraps at once
  - Common Layer: Actual alloc/free
- Invocation
  - Register target-specific implementations, invocation with closure (args + member data)

MICROBENCHMARK OVERHEADS

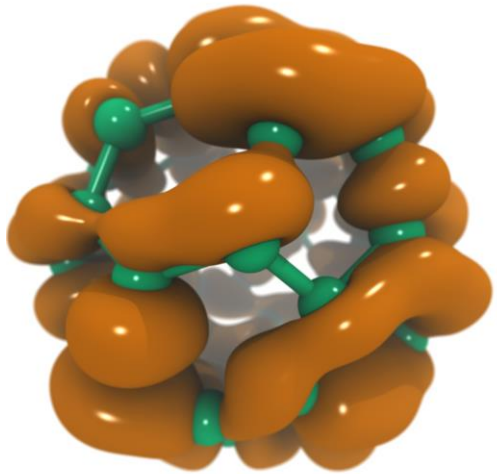
Always less than 2% or 500ns, before perf tuning

Semantics	Baseline			HiHAT Prototype		
	256B	4KB	4MB	256B	4KB	4MB
Alloc data, GPU	5.7us	3.6us	1634.6us	6.0us (1.05x)	3.8us (1.05x)	1.00x
Alloc data, CPU	10.9us	6.6us	2168.8us	11.4us (1.05x)	6.9us (1.04x)	1.00x
Free data, GPU	4.4us	3.3us	141.7us	4.6us (1.05x)	3.5us (1.06x)	1.00x
Free data, CPU	13.5us	22.1us	782.0us	13.8 (1.02x)	14.1 (0.81x)	1.00x
Wrap data, CPU	-	-	-	0.11us	0.11us	0.08us
Wrap data, GPU	-	-	-	0.04us	0.04us	0.05us
Unwrap data, CPU	-	-	-	0.04us	0.04us	0.05us
Unwrap data, GPU	-	-	-	0.11us	0.11us	0.08us
Move data, CPU ⇒ GPU0Mem	8.0us	9.0us	350.2us	8.4us (1.06x)	9.1us (1.01x)	353.0us (1.01x)
Move data, CPU ⇐ GPU0Mem	7.5us	7.3us	330.0us	7.9us (1.05x)	7.6us (1.04x)	335.4us (1.02x)
Move data, CPU ⇒ GPU0 const	0.13us	0.13us	-	0.35us (2.76x)	0.35us (2.76x)	-
Move data, GPU0Mem ⇔ GPU0Mem	3.8us	3.8us	4.6us	4.0us (1.05x)	4.0us (1.06x)	4.9us (1.06x)
Move data, GPU0Mem ⇔ GPU1Mem	12.1us	11.5us	391.3us	12.5us (1.03x)	12.0us (1.04x)	397.3us (1.02x)
Overhead to look up addr (User-Common)	-	-	-	0.06us	0.06us	0.06us

Still improving confidence in overhead quantification (power mgt, timing overhead)

MOLECULAR ORBITALS (MO) APPLICATION

- Compute wavefunction amplitudes on a grid for visualization
- Evaluate linear combination of Gaussian contractions (polynomials) at each grid point, function of distance from atoms
- Algorithm made arithmetic bound via fast on-chip memory systems
- Three different algorithms for different data sizes and hardware:
  - GPU constant memory
  - shared memory tiling
  - L1 global memory cache
- Representative of a variety of other grid-oriented algorithms, stencil codes, etc.
- Initial adaptation from CUDA to HiHAT proof-of-concept took roughly 90 minutes
- Use of special GPU hardware features, APIs helped drive completeness of HiHAT proof-of-concept implementation already at an early stage



MO PERFORMANCE

- Performance of MO algorithm on HiHAT User Layer PoC implementation closely tracks CUDA performance.
- CPU baseline results for the tests below: Xeon E5-2260v3 301 ms, 2xPOWER8 520ms

Molecular Orbital Algorithm and Kind of APIs	Time (speedup)	HiHAT API Overhead
ConstMem CUDA	49.32 ms (1.469x)	-
SharedMem CUDA	72.46 ms (1.000x)	-
L1CachedGlobMem CUDA	66.76 ms (1.085x)	-
ConstMem HiHAT	50.78 ms (1.461x)	1.029x
SharedMem HiHAT	74.21 ms (1.000x)	1.024x
L1CachedGlobMem HiHAT	68.64 ms (1.081x)	1.028x

PORTABILITY ON MO

Mapping between CUDA and HiHAT

- Time to port MO: 90 minutes
- Mostly 1:1 mapping
  - Some CUDA APIs not covered yet
- HiHAT has fewer unique APIS (9 vs. 11)
  - AllocBuffer handles flags
  - WrapBuffer handles symbols
- HiHAT has fewer static API calls (32 vs. 39) when tagging scheme is used for cleanup

Category	Original CUDA		Ported to HiHAT	
Data mvt	cudaMemcpy() cudaMemcpyToSymbol()	7 7	MoveData() MoveData()	7 2
Invoke	<<<>>>	3	Invoke()	3
Configuration	cudaSetDeviceFlags() cudaFuncSetCacheConfig()	1 2	(same) (same)	1 2
Data mgt, minimal	cudaMalloc() cudaMallocHost() cudaHostAlloc() cudaFree() cudaFreeHost() cudaHostGetDevicePointer() -	7 1 1 7 2 1 -	AllocBuffer() (implicit) AllocBuffer() CleanBuffers() - (same) WrapBuffer()	7 0 1 (1) 0 1 7
Data mgt, eliminatable	- -	- -	DeallocBuffer() UnwrapBuffer()	7 7
Totals static static min'l unique	 14+0+3+3+19+0 14+0+3+3+19+0 2+1+2+6	 39 39 11	 7+7+3+3+16+14 7+2+3+3+17 1+1+2+5 (+2)	 50 32 9