Multicore, Multithreaded, Multi-GPU Kernel
VSIPL Standardization, Implementation, & Programming Impacts

Anthony Skjellum, Ph.D.

http://www.runtimecomputing.com
Fine-Grain Concurrency
Widespread in HPEC platforms

- SMP (not new)
- Multicore (e.g., Core i7, 8641D)
  - Hyperthreading and other weaker internal core concurrency
- SMP VxWorks (tasks vs. POSIX processes... use case new, issue not new)
- Customers are demanding standards-based approaches to programming VSIPL in multicore environments
- Thread-safe VSIPL increasingly in demand
Thread Safe User-Level

- Portability must be maintained
- Users may need to control/hint use of concurrency internally in a library vs. their own task concurrency
- Affinity issues and choices arise
- Single-threaded users don’t want to pay for overhead of multithreaded models
- VSIPL APIs are a good start
Model VSIPL Model MT1

- **Rules**
  - Every thread that uses VSIPL calls vsip_init() and vsip_finalize()
  - Every thread works with independent objects and memory
  - All existing VSIPL syntax is valid
  - No new VSIPL syntax is needed or provided
- Pthreads (POSIX) as threading library
- High quality implementations allow internal concurrent execution of user threads
- Low overhead
What’s in the poster

- Why Thread-safe VSIPL important
- A practical multithreaded programming model for VSIPL defined (MT1)
- An example of MT1-based FFT example
- Explanations of issues, concerns, and barriers going to multithreaded
- Discussion of GPU-related (offload-engine) related issues