Dynamic Adaptive Multithreading: Programming Models and System Software

ParalleX: A Study of a New Computation Model

Guang R. Gao

Electrical & Computer Engineering
University of Delaware

ggao@udel.edu

Thomas Sterling

Department of Computer Science
Louisiana State University
	ron@cct.lsu.edu
Contributors

• Collaborators
  – Maciej Brodowicz (LSU/CCT)
  – Hartmut Kaiser (LSU/CCT)
  – Rick Stevens (U. Chicago/ANL, Co-PI)
  – Mark Herald (U. Chicago/ANL)
  – Weirong Zhu (U. Delaware)
  – Jeff Becker (NASA ARC)

• Acknowledgements
  – NSF (Darema, etc.)
  – DOE OS, NSA
  – Argonne National Laboratory
  – Sandia National Laboratory
  – IBM
  – ET International Inc.
  – Members of UDel/CAPSL
  – LSU Center for Computation and Technology
Previous/On-Going Work

- MIT Dataflow Model (1973 – 1992)
- EARTH (McGill/UDel: 1994-2004)
- CARE (UDel: 1999-2004)
- OpenMP-XN (UDel: 2003- )
- Percolation (Gao, Sterling, et al)
- Parcel model (Sterling, et al.)
- Gilgamesh (Sterling, Brodowicz, et al)
- LITL-X (Gao, Sterling, et al.)
- ParalleX (Sterling, Gao, et al.)
Goals for Future Parallel Execution Models

• Technology trends
  – Multicore components
  – Heterogeneous structures and accelerators

• Performance degradation
  – Latency (idle time due to round trip delays)
  – Overhead (critical path support mechanisms)
  – Contention (inadequate bandwidth)
  – Starvation (sufficient parallelism and load balancing)

• Power consumption
  – Just too much!
  – Dominating practical growth in mission critical domains

• Reliability
  – Single point failure modes cannot be tolerated
  – Reduced feature size and increased component count

• Changing application workload characteristics
  – Data (meta-data) intensive for sparse numerics and symbolics

• Programmability & ease of use
  – System complexity, scale and dynamics defy optimization by hand
Key Ideas in HTVM/Parallel-X
Execution & Programming Models

- Hierarchical multi-grain multithreading
- Global name space
  - Does not assume memory coherence
  - Location consistency
- Fine grain synchronization
  - Futures
  - Lightweight event driven
    - Message driven
    - Data driven
  - In-Memory atomic synchronization
    - Memory contains parallel control state
  - Split-phase transactions
- API: Runtime Optimization Aware
  - SPMD and beyond (general fork/join)
  - Future + fine-grain threads
  - Locality specification
  - Percolation
  - Domain-specific knowledge input
In the above execution scenario: three large grain threads are in progress, within each a number of small grain threads are forked each in turn invokes the execution of a collection of tiny grain threads.

Note: the lower two levels of the two threads are fine-grain.
Classes of Target Architectures

- **Application space**
  - Data intensive
  - Dynamic sparse data structures
  - e.g.: AMR, directed graphs, heuristic driven search problems

- **Conventional systems**
  - MPP and commodity clusters
  - Multicore
  - Heterogeneous
  - FPGA enhanced for system software acceleration

- **Cyclops**
  - IBM
  - NSA sponsored
  - Fine grain architecture

- **Gilgamesh-2 point design**
  - Heterogeneous with respect to temporal locality
  - Combines PIM structures with dataflow streaming accelerators
ParalleX Semantics

- Locality domains
  - Intra-locality: Controlled synchronous
  - Inter-locality: Asynchronous between localities
- Distributed shared memory
  - Not cache coherent
  - Copy semantics
  - Affinity relationships
- Split-phase transactions
  - Work queue model
    - Only do work on local state
    - No blocking or idle time for remote access
- Message-driven computation
  - Parcels carry data, target address, action, continuation
  - Percolation
- Multi-threaded
  - First class objects
  - Dataflow on transient values
- Local (lightweight) control objects
  - Futures
  - Dataflow
  - Data-directed parallel control
- Meta-data embedded address translation
- Dynamic optimization
Strategic Accomplishments

• LITLX running on Cyclops simulator
• ParalleX reference implementation
  – Written in Lisp (for rapid prototyping)
  – Running kernel applications
• Distributed ParalleX in-work
  – Based on C++ libraries
  – PRECISE binary instruction set
• Exploits prior NSF Percolation studies (UDel & Caltech)
  – Prestaging of data at remote sites
  – Work distribution and load balancing
  – Offload overhead and latency from precious resources
Gilgamesh-2 System Elements

- **Executable Memory**
  - Supports low-temporal (e.g. touch once) locality global data operations
  - Threads in memory with wide ALUs
- **Dataflow Accelerator**
  - Supports high-temporal locality operations
  - Very high throughput low latency processing
  - Low power per operation
- **Data Vortex optical network**
  - Innovative topology
  - Low latency, low logic
  - Graceful degradation of injection rate with traffic density
  - High degree switches
- **Penultimate store**
  - Fast backing store for core computing
  - Exploits highest density semiconductor memory
  - Reconfigurable for fault tolerance
Gilgamesh-2 System

Data Vortex

D_0

D_1

D_2

D_{128k-1}

PS_0

PS_1

PS_2

PS_{128k-1}

File I/O
Enhancing Conventional Multicore
FPGA ParalleX Accelerator

- Based on prior work performed on MIND architecture as part of Caltech/JPL Gilgamesh project
- Goal: enhance scalability and efficiency
  - Hide system wide latency
  - Reduce parallelism control overhead
- Design FPGA-based hardware drivers and co-processors to support ParalleX model
  - Parcel message-driven computation handler
  - Medium grained multithread execution scheduler
  - Global address translation support
  - Percolation pre-staging task manager
  - (possibly) local control object synchronization acceleration
Future Work at LSU

• Full function distributed ParalleX implementation
  – Exploiting existing C++ libraries
  – Commodity cluster
• Optimized performance implementation
  – Custom threads model
• Application driven evaluation
  – Performance advantage
  – Time cost model of critical mechanisms
  – Scalability sensitivity studies
• FPGA accelerators
  – Key mechanisms
  – CCT reconfigurable testbed
• Specification for Agincourt
  – API for ParalleX
  – Compiles to PXIF