Gedae: Automated Multithreading and System Requirements for Interoperability

William I. Lundgren
Gedae, Inc.

bill.lundgren@gedae.com
Multicore Programming Challenges

- Multicore processors demand multithreaded applications
- Multithreading increases complexity of software
  - Balancing load
  - Controlling concurrency
  - Debugging is difficult
  - Software is architecture dependent
- Executing at high speeds and in parallel is the domain of experts
Gedae – The Bridge Between Art and Engineering

- We observed the art
  - 20 years of working with clients relying on parallelism for realtime applications
- Created a tool to allow engineering
  - Automate the pitfalls and tedium of parallel programming
- Gedae used for large production software
  - Substantial productivity and cost savings
Multicore Demands

Multicores need a software development tool and process that:

- produces efficient and reliable software,
- maintains or improves developer productivity,
- targets a broad range of processors,
- applies to all compute intense problems, and
- ensures portability to all future hardware
Gedae’s Approach

- Functional Model
- Implementation Specification

Verifier and Compiler

Threaded Application

- Thread Manager
- Vector & IPC Libraries
- Multicore Hardware

Hardware Model
Guiding Principle

- Functional model
- Abstract out architecture-specific details
- Generate software closer to hardware
- Libraries

Move implementation from language and libraries into compiler
Give the compiler opportunity and flexibility
What is Gedae?

- A stream programming language extended to support general purpose programming
- A multithreading compiler
  - Thread definition
  - Thread decomposition for distribution with data transfers added
  - Concurrency control
  - Deadlock avoidance
  - Memory sharing among threads
  - Memory optimization within threads
- A thread scheduler
- And …
What is Gedae? (continued)

- A suite of tools to use with the language and compiler
  - Graphical editor
  - Data display and analysis tools
  - Implementation specification tools
  - Distributed debugging tools
    - Implementation analysis tools
    - Execution analysis tools
  - Automated build system
What is Gedae? (continued)

An automated build system:
- Create functions to speed up compute-intensive functions in legacy code
  - Simple interface for typical use
  - Fully customizable when needed
- Alternately creates fully-contained standalone executables
  - Create a deployable application entirely in Gedae
  - Offers a variety of boot schemes and options, including booting from EPROM and executable compression
Cell/B.E. Benchmark Results - Summary

- Monte Carlo Black-Scholes Simulation
  - Exceed performance of hand optimized code
- Matrix multiply
  - Block data layout
  - 194 gflops – 95% of theoretical max
- SAR (synthetic aperture RADAR)
  - End to end timing including 0 flop cornerturn
  - Sustained 88 gflops/sec
  - 87x algorithm on a 500 Mhz quad altivec board (normalized clock speed – 13.6 x)
The Physical Problem

- Actual implementation uses a clever choice of the pulse and only a vector multiply and fast Fourier transform are required.
- The data processed is the return from Npulse pulses each with Nrange range gates.
RADAR Pulse Compression

Matrix of data in System Memory – each row is the vector return from a transmitted pulse

The orange arrows represent actions taken on each time slice. The idea is to move data in parallel with the processing of the data. That is: move the next data in, move the last data out and process the current data.
System Requirements for Interoperability

- Controlling access to SPEs
  - Function on PPE to request N SPEs that alternately:
    - Returns number available
    - Blocks until SPEs are available
  - Function on the SPEs to indicate SPEs have completed processing so the PPE can retask

- Fast context switch
  - Fast code load and start
    - ELF provides most of the capability we need
    - Specify multiple segments with the size, source and target locations and amount to be loaded
    - We need to pass parameters to a code overlay function
    - Load multiple code segments
    - All features should be optional to keep the loader small
DEMONSTRATION OF RADAR PULSE COMPRESSION ON IBM’S PX CAB BOARD