Resilient software using modular programming techniques that normal humans can use (and understand)

Tim Mattson (Intel Labs)
Parallel programming principles: Designed around the Human programmer (not the computer)

Tim Mattson (Intel Labs)
Psychology of Programming in one slide

• Human reasoning is model based … Programming is a process of successive refinement of a problem over a hierarchy of models.[1]

• The models are informal, but detailed enough to support simulation.

• Programmers use an informal, internal notation based on the problem, mathematics, programmer experience, etc.
  – Within a class of programming languages, the solution generated is only weakly dependent on the language.[2] [3]

• Opportunistic Refinement:[4]
  – Progress is made at multiple levels of abstraction with effort focused on the most productive level.

Psychology of Programming in one slide

• Human reasoning is model based … Programming is a process of successive refinement of a problem over a hierarchy of models.[1]

• Opportunistic Refinement:

  – Progress is made at multiple levels of abstraction with effort focused on the most productive level.

This means our programming principles must help us:

  – Define a productivity layer for domain-specialist-programmers that supports model-building in the application domain.

  – Expose a machine model programmers can understand (programmers run mental simulations to understand execution)

  – Allow programmers to move back and forth within the hierarchy of models (Opportunistic refinement).

  – Abstract the hardware but don’t hide it!


Models that hide Communication and network details are “dead on arrival”

- Jim Demmel’s group at UC Berkeley has shown dramatic performance improvements by:
  1. Communication avoiding algorithms
  2. Algorithms that exploit the details of a computer’s network

Source: Jim Demmel, UC Berkeley

2.5D Matmul on BG/P, 16K nodes / 64K cores

Distinguished Paper Award, EuroPar’11
SC’11 paper by Solomonik, Bhavele, D.
We spend too much time on the less important problems (programming models)

- Modern software engineering stresses modular development by large distributed teams.
  - The Parallel Composition problem ... Expose resource management and schedulers to safely mix parallel software modules.
  - Example: Heidi Pan’s dissertation from MIT & UC Berkeley.

- There will never be “one programming model” to rule them all ... So lets stop trying to find that ideal model.
  - Make the choice of programming model irrelevant by establishing a common Intermediate representation.
    - Composition across programming models.
    - Example: SPIR project in OpenCL based on LLVM.
Turning Patterns expressed as Python code into high performance parallel code

ASP ... a platform to write domain specific frameworks.

Helps turn design patterns into code.

Intel/UCB test projects:
(1) molecular modeling.
(2) Data analytics
Conclusion

• Put the Human (not the computer first) and create a foundation of parallel computing principles grounded in how programmers think.

• Models at every level must be available to the programmer and provide insights for REAL hardware (consider the success of communication avoiding algorithms).

• The Parallel Composition problem is where the most work is needed:
  – Composition across software modules (e.g. Lithe)
  – Composition across programming models (e.g. SPIR)