Panel: Programming Models at Exascale: Are We Ready for the Challenges?

Brad Chamberlain, Cray Inc.

Future Approaches to Data-Centric Programming for Exascale

May 20th, 2011

Q1: What is good and bad with current programming models?

A: (Thinking primarily of MPI + OpenMP/pthreads/CUDA)

the good

the bad

- maps well to hardware
  - lots of control
  - decent generality
  - supports a division of labor
  - relatively easy to implement

Q1: What is good and bad with current programming models?

A: (Thinking primarily of MPI + OpenMP/pthreads/CUDA)

the good

- maps well to hardware
  - lots of control
  - decent generality
  - supports a division of labor
  - relatively easy to implement

the bad

- low-level of expression
  - many details to manage
  - barriers to portability
  - requires a hybrid programming model
  - implementation is fairly constrained

Q1: What is good and bad with current programming models?

A: (Thinking primarily of MPI + OpenMP/pthreads/CUDA)

the good

- maps well to hardware
  - lots of control
  - decent generality
  - supports a division of labor
  - relatively easy to implement
- based on familiar languages

the bad

- low-level of expression
  - many details to manage
  - barriers to portability
  - requires a hybrid programming model
  - implementation is fairly constrained

Q1: What is good and bad with current programming models?

A: (Thinking primarily of MPI + OpenMP/pthreads/CUDA)

the good

- maps well to hardware
  - lots of control
  - decent generality
  - supports a division of labor
  - relatively easy to implement
- based on familiar languages

the bad

- low-level of expression
  - many details to manage
  - barriers to portability
  - requires a hybrid programming model
  - implementation is fairly constrained
- based on increasingly antiquated languages

Q1: What is good and bad with current programming models?

A: (Thinking primarily of MPI + OpenMP/pthreads/CUDA)

the good

- maps well to hardware
  - lots of control
  - decent generality
  - supports a division of labor
  - relatively easy to implement
- based on familiar languages
- works today

the bad

- low-level of expression
  - many details to manage
  - barriers to portability
  - requires a hybrid programming model
  - implementation is fairly constrained
- based on increasingly antiquated languages

Q1: What is good and bad with current programming models?

A: (Thinking primarily of MPI + OpenMP/pthreads/CUDA)

the good

- maps well to hardware
  - lots of control
  - decent generality
  - supports a division of labor
  - relatively easy to implement
- based on familiar languages
- works today

the bad

- low-level of expression
  - many details to manage
  - barriers to portability
  - requires a hybrid programming model
  - implementation is fairly constrained
- based on increasingly antiquated languages
- may not work tomorrow

Q2: What is needed to cope with future system complexity?

A:

- hierarchy in our execution and machine models
  - flat set of cooperating binaries no longer sufficient
  - need to expose different processor and memory types
- greater investment in software to help manage system complexity
  - increasingly autonomous, resource-aware runtimes
  - optimizing compilers
  - abstractions through frameworks and languages
- willingness on users' part to yield some control
  - rely more on frameworks, abstractions, automation
  - resilience as a first-class concern, not an afterthought

Q3: What will be the impact on mere mortals?

A: Mortals will always find a way to use a new technology effectively

- though certain changes in approach could help:
  - unified notation for parallelism & locality
  - multiresolution design to support diverse skillsets
- programming models will challenge us, but my bigger concern is whether system imbalance will render the machines unusable
  - not enough memory to make good use of the flops
  - “program smarter, not harder” is not an answer
  - a renaissance for out-of-core algorithms?

CRAY
THE SUPERCOMPUTER COMPANY

http://chapel.cray.com   chapel_info@cray.com   http://sourceforge.net/projects/chapel/

TODO

- Move sub-bullets to speaker's points or make non-sentences
- Other presentation:
  - assembly language programming as joke to break up talk
  - family feud: Why did HPF fail?
- Notes: Commodore 64 vs. today's machines
  - we are not used to hardship and working under severe constraints
  - WWII vs. today's wars