Declarative Tuning for Locality in Parallel Programs

Sanjay Chatterjee, Nick Vrvilo, Zoran Budimlic, Kathleen Knobe, Vivek Sarkar

Rice University
Habanero Extreme Scale Software Research Project

http://habanero.rice.edu/
Tuning Goal: A separation of concerns

Application problem → Domain Program → Optimized mapping to platform

The domain expert:
- Finance
- Gaming
- Chemistry

The tuning expert:
- Parallelism
- Locality
- Load balancing

Domain expert doesn’t need to know a lot about parallelism

Tuning expert doesn’t need to know a lot about the app

- Domain expert, but later
- Different person, different skills
- Automated: static analysis / autotuning

http://habanero.rice.edu/
The Concurrent Collections (CnC) Programming Model

- CnC is a *coordination* language, paired with a programming language
- Declarative *dependence*-based programming model
- CnC programming constructs:
  - Step collection
    - specifies computation tasks in the application
  - Item collection
    - specifies data sets for the application
  - Tag collection
    - specifies pure control flow in the application

https://wiki.rice.edu/confluence/display/HABANERO/CnC
http://habanero.rice.edu/
CnC Dependence Relations

- Producer must execute before consumer
- Controller must execute before controllee

CnC Domain Spec

($init:) \rightarrow (cholesky:0...N)

(cholesky:iter) \rightarrow (trisolve:k+1...N,iter)
(cholesky:iter) \leftarrow [array:iter,iter,iter]
(cholesky:iter) \rightarrow [array:iter,iter,iter+1]

(trisolve:row,iter) \rightarrow (update:iter+1...row+1,row,iter)
(trisolve:row,iter) \leftarrow [array:row,iter,iter]
(trisolve:row,iter) \leftarrow [array:iter,iter,iter+1]
(trisolve:row,iter) \rightarrow [array:row,iter,iter+1]

(update:col,row,iter) \leftarrow [array:col,row,iter]
(update:col,row,iter) \leftarrow [array:col,iter,iter+1]
(update:col,row,iter) \leftarrow [array:row,iter,iter+1]
(update:col,row,iter) \rightarrow [array:col,row,iter+1]

(Domain Spec)
CnC Tuning Spec

Domain expert knows:
physics, finance, data analytics, ...

Tuning expert knows:
parallelism, locality, energy, ...

Domain spec

X86 Tuning:
Goal: Performance

Xeon Phi tuning:
Goal: Energy

Xeon Phi Tuning
Goal: Performance

Shared memory
Cluster
Xeon Phi
Future Arch.

http://habanero.rice.edu/
CnC Tuning Spec for Locality

- Hierarchical Distribution Functions
  - **Declarative step/item placement**
  - Enables spatial locality
    
    `[array:col,row,iter]: { iter % $RANKS };`

- Hierarchical Affinity Groups
  - **Declarative step/item grouping**
  - Enables temporal locality
    
    `(CTU:iter) →  
    (cholesky:iter), (TU:0...iter, iter);`
    
    `(TU:row,iter) →  
    (trisolve:row,iter), (update:0...iter, row, iter);`

http://habanero.rice.edu/
CnC Tuning Abstract Machine Model

INTERCONNECT

ABSTRACT MACHINE MODEL

TUNING TREE

http://habanero.rice.edu/
CnC Tuning Execution Model: Affinity Grouping

Affinity Group A

A, 1  A,2  A,3

Tuning Tree (Staging area)

Hierarchical Place Tree (Execution Area)

http://habanero.rice.edu/
CnC Tuning Execution Model: Affinity Group Distribution

Tuning Tree (Staging area)

Hierarchical Place Tree (Execution Area)
CnC Tuning Execution Model: Runtime Step Execution

Tuning Tree (Staging area)

Hierarchical Place Tree (Execution Area)

http://habanero.rice.edu/
CnC Tuning Example: Rician Denoising

- Iterative stencil computation

Tile computations:
- uDiffCompute
- gCompute
- rCompute
- ugCompute
- uCompute

Data from computation in previous iteration

http://cdsc.ucla.edu/
http://habanero.rice.edu/
Affinity Grouping for Temporal Locality

(rdTunedStep: iter, row, col, itero, rowo, colo = 
rdStep: rowo + row, colo + col, itero + iter);

(pyramid: rowo, nRow, colo, nCol, nIter, itero) 
→ (rdTunedStep:
iter @ 0...nIter, iter...(nRow - iter),
iter ... (nCol - iter),
itero, rowo, colo)
Tuning Rician Denoising for Multi-Core

- Dual-socket 12-core Intel Westmere
- Up to 3.5x improvement on specific tile size
- Overall improvement of 19% over best optimal tiled version

http://habanero.rice.edu/
Tuning Rician Denoising for Distributed Systems

RicianDenoising 2D (8192 x 8192) (30 iterations)

Time (s)

- UnTuned Shared Mem (tile 128): 61.62
- Pyramid Tuned Shared Mem (tile 256): 39.10
- Tuned Row Distributed (tile 256) (2 Nodes): 54.12
- Tuned Row Distributed (tile 256) (4 Nodes): 24.22
- Tuned Row Distributed (tile 256) (8 Nodes): 10.65
- Tuned Row Distributed (tile 256) (16 Nodes): 6.59

Tunings (16 ppn)

http://habanero.rice.edu/
Conclusions

• The CnC tuning framework helps allow locality tuning of applications separately from its domain specification.
• Declarative tuning specification simplifies exploration of optimal mapping of application for different targets.
• Habanero task-based dynamic runtime system enables balanced execution for optimized locality and parallelism.

http://habanero.rice.edu/
Source Code:

- CnC Tuning:

- Habanero Project:
  - [https://github.com/habanero-rice](https://github.com/habanero-rice)
Backup
CnC Tuning Workflow

- Tuning Spec
  - CnC Graph Translator Tool
  - Generated Tuning Support Code
  - Habanero C Compiler
  - Custom-tuned Executable

- Domain Spec
  - CnC Graph Translator Tool
  - Generated Step Code Skeleton
  - Fully Implemented Step Code

http://habanero.rice.edu/
Runtime Locality Control: Hierarchical Place Trees (HPT)

Example: Intel Xeon Dual Quad Core
-2 sockets with shared L3
-2 shared L2 per socket

- Each place has one queue per worker
- Ensures non-synchronized push and pop
- Workers attach to (own) leaf places
- Any worker can push a task at any place
- Pop / steal access permitted to subtree workers
- Workers traverse path from leaf to root
- Tries to pop, then steal, at every place
- After successful pop / steal worker returns to leaf
- Worker threads are bound to cores

Tuning Tiled Cholesky Factorization for Distributed Systems

Tiled Cholesky Factorization (6000 x 6000)

- Tuned Iteration Distributed
- Tuned Row Distributed

Nodes (16 ppn)

- 2 nodes: 2.09 s (Tuned Iteration), 3.67 s (Tuned Row)
- 4 nodes: 1.40 s (Tuned Iteration), 3.30 s (Tuned Row)
- 8 nodes: 1.34 s (Tuned Iteration), 2.84 s (Tuned Row)
- 16 nodes: 1.33 s (Tuned Iteration), 0.97 s (Tuned Row)

http://habanero.rice.edu/
Tuning Smith-Waterman for Distributed Systems

SmithWaterman (185600, 192000)
- Tuned Wave Distributed
- Tuned Row Distributed
- Tuned Col Distributed

Time (s)

Nodes (16 ppn)