Collaborative Threads
Exposing and Leveraging Dynamic Thread State for Efficient Computation
Outline

• Views on Parallelism

• Thread Collaboration and Semantic State

• Representation of Semantic State with the CST

• Experimental Results
  • Result reuse
  • Orienting a Computation
Outline

• Views on Parallelism
• Thread Collaboration and Semantic State
• Representation of Semantic State with the CST
• Experimental Results
  • Result reuse
  • Orienting a Computation
Parallelism and Threads
Parallelism and Threads

- Parallelism today relies on threads
- Splitting-up of data with data-parallelism
- Splitting-up of work with task-parallelism
Parallelism and Threads

- Parallelism today relies on threads
  - Splitting-up of data with data-parallelism
  - Splitting-up of work with task-parallelism
- Higher-level models exist as well
  - TBB, Cilk to express task-parallelism
    - Implements the fork-join paradigm
    - Provides higher-level parallel abstractions (parallel_for, parallel_do,...)
  - CnC to express natural parallelism
What is Missing?
What is Missing?

- Thread interactions is restricted to
- Locks, barriers and TMs for synchronization
- Shared memory and message passing for shared data
What is Missing?
Current Use of Parallelism
Current Use of Parallelism

- Current models *break-up* a computation
  - Distribution of work is done just in time at best
  - Break-up oblivious to the state of the computation
  - Only the state of data-structures (what threads read/write to them) is used (for example, the Galois model)
  - Higher-level semantic information is lost
Alternative Uses Required
Alternative Uses Required

• Many HPC combinatorial optimization problems and search problems
  • Are resource bound
  • Have a performance dependency on more than how work is split-up
Alternative Uses Required

- Many HPC combinatorial optimization problems and search problems
  - Are resource bound
  - Have a performance dependency on more than how work is split-up
  - Performance depends on the direction of computation, scheduling of tasks, ordering of computations, pruning of the search space, etc...
  - For example: certain orderings will lead to a faster space pruning
Alternate Views on Parallelism
Alternate Views on Parallelism

- N-way parallelism leverages competition [HotPar 2009]
- Pick the best through competition amongst diverse ways
Alternate Views on Parallelism

• N-way parallelism leverages competition [HotPar 2009]
  • Pick the best through competition amongst diverse ways

• We propose to allow threads to collaborate
  • Share higher-level semantic information
  • Allows the dynamic adaptation of work and leveraging of the state of the entire computation
Outline

• Views on Parallelism

• Thread Collaboration and Semantic State

• Representation of Semantic State with the CST

• Experimental Results
  • Result reuse
  • Orienting a Computation
What is Thread Collaboration?
What is Thread Collaboration?

- Programmer identification of useful semantic state
What is Thread Collaboration?

- Programmer identification of useful semantic state
- Sharing of identified state and meta-information to dynamically determine the best way to
  - optimize for computational efficiency (do no more than required)
  - orient the computation (do what is most likely to yield results)
  - utilize resources (select adapted resources)
Examples of Semantic Information
Examples of Semantic Information

- Partially computed results (in a highly parallel computational problem)
Examples of Semantic Information

- Partially computed results (in a highly parallel computational problem)
- Successfulness (problems with searches over large spaces)
Examples of Semantic Information

- Partially computed results (in a highly parallel computational problem)
- Successfulness (problems with searches over large spaces)
- Execution time (similar computations on various types of cores)
Examples of Semantic Information

- Partially computed results (in a highly parallel computational problem)
- Successfulness (problems with searches over large spaces)
- Execution time (similar computations on various types of cores)
- Data footprints (problems with irregular read/write patterns)
Examples of Semantic Information

- Partially computed results (in a highly parallel computational problem)
- Successfulness (problems with searches over large spaces)
- Execution time (similar computations on various types of cores)
- Data footprints (problems with irregular read/write patterns)
- ...

...
Uses of Semantic State
Uses of Semantic State

• Semantic state can answer various questions, such as:
Uses of Semantic State

- Semantic state can answer various questions, such as:
  - Which other solved sub-problem can I leverage?
Uses of Semantic State

- Semantic state can answer various questions, such as:
  - Which other solved sub-problem can I leverage?
  - If I am looking for work amongst several possibilities, which should I choose?
Uses of Semantic State

- Semantic state can answer various questions, such as:
  - Which other solved sub-problem can I leverage?
  - If I am looking for work amongst several possibilities, which should I choose?
  - Which resource is the best for my sub-problem?
Uses of Semantic State

• Semantic state can answer various questions, such as:
  • Which other solved sub-problem can I leverage?
  • If I am looking for work amongst several possibilities, which should I choose?
  • Which resource is the best for my sub-problem?
  • What data are other threads likely to touch?
Challenges for the Model
Challenges for the Model

• **Expression** of higher-level semantic state

• Flexible and easy way to express state
Challenges for the Model

• **Expression** of higher-level semantic state
  • Flexible and easy way to express state

• **Organization** of semantic information in a useful way
  • Compact representation of shared state
  • Low-overhead storage and retrieval
Outline

• Views on Parallelism

• Thread Collaboration and Semantic State

• Representation of Semantic State with the CST

• Experimental Results
  • Result reuse
  • Orienting a Computation
Computational State Tree (CST)

- Clusters “Similar Sub-problems” together into a tree

○ : Represents thread state
Computational State Tree (CST)

- Clusters “Similar Sub-problems” together into a tree

- Hierarchical
- Incremental
- Approximate

 (): Represents thread state
Computational State Tree (CST)

- Hierarchical
- Logarithmic lookup time on the clusters
- Incremental
- Incrementally build without rebuilding from scratch
- Approximate
- Not guaranteed to form best clusters
- Results in quick lookups but not optimal
What Can We Do With This?

• Results re-use
• Orient a computation
• Sub-problem prioritization
• Core selection
Outline

- Views on Parallelism
- Thread Collaboration and Semantic State
- Representation of Semantic State with the CST
- Experimental Results
  - Result reuse
  - Orienting a Computation
Results Re-use

• Leverage results of “Similar sub-problems”
• Locate sub-problems which are similar
• Share partial results

• Examples
  • Sum of subsets
  • K-Means
Given a set of integers and ‘s’, does any non-empty subset sum to ‘s’?

Naive parallelization makes each subset computation a different task

\{3, 4, 5, 6, 7, 8, 9\}

\{2, 3, 4, 5, 6, 7, 8, 9\}

\{3, 4, 5, 6, 9\}

\{3, 4, 5, 6, 7, 8, 9, 12\}
Sum of Subsets

• Given a set of integers and ‘s’, does any non-empty subset sum to ‘s’?

• Naive parallelization makes each subset computation a different task
  \{3, 4, 5, 6, 7, 8, 9\}

  \{2, 3, 4, 5, 6, 7, 8, 9\}

  \{3, 4, 5, 6, 9\}

  \{3, 4, 5, 6, 7, 8, 9, 12\}

• Large amount of redundancy can be exploited
• Programmer specifies a similarity metric:

• Here, cardinality of the symmetric difference
  • \(|(A-B) \cup (B-A)|\)

• Threads can share their current computation:
  • share ({1, 10, 9, 8, 3, 4}, 35);
  • share ({8, 3, 4}, 15);
Sum of Subsets

- Before computing a sum, threads can lookup the currently best available result:
  - `lookup_closest({10, 9, 8, 3, 4});`
- The return value varies based on scheduling
- Might need to add or subtract a few values to generate needed sum
- Automatically makes best use of previously computed values
K-Means

- Partitions ‘n’ points into ‘k’ clusters
- Choose ‘k’ random centroids
- Associate point to closest centroid
- Re-compute centroids
- Iterate
K-Means

• Each point performs ‘k’ computations to determine closest centroid
K-Means

• Points which are close to each other can potentially share their closest centroid
• Points which are close to each other can potentially share their closest centroid

share( (x1, y1) , (A) );
• Points which are close to each other can potentially share their closest centroid

```c
lookup_closest((x2, y2));
```
K–Means

- Points which are close to each other can potentially share their closest centroid
• Ran more than twice as fast with collaboration turned on
Where Does the Speedup Come From?

- Computation Reduction
  - Original: Makes ‘k’ comparisons for each point
  - Collaborative: Single point computes, and those around it share the value (the more dense the points, the more potential for collaboration)

- More efficient computation with collaboration
  - Re-wrote key sub-step in a collaborative manner
Outline

• Views on Parallelism

• Thread Collaboration and Semantic State

• Representation of Semantic State with the CST

• Experimental Results

  • Result reuse

• Orienting a Computation
Orienting a Computation

- Computational space is large
- Some parts of this space may be more fruitful to process
- Guide threads to these parts of the space
- Example: SAT Solver, Finding maxima/minima for non-convex spaces
SAT Computational Difficulty

- $k$-SAT with $m$ clauses and $n$ variables
- $\alpha = \frac{m}{n} \approx 4.26$

- Satisfiable solutions are clustered in small pockets
GSat

• Similar to WalkSAT
• Local Search Algorithm
• Start with Random Assignment
• Flip a variable, minimizing number of unsatisfied clauses
• Iterate till you find a solution
Orienting GSsat

- Use GSsat as an All-Solutions finder
- When one Satisfiable solutions is found
- Publish the location of the successful solution
- share ( current_truth_assignment )
- Guide other close-by threads into these pockets
Guiding other threads

Solution space
Guiding other threads

Solution space
Results

Solutions found in 120 seconds

\[ n = 38, \alpha = \frac{m}{n} \approx 4.26 \]
Conclusion

• For some problems, everything is parallel, but not everything is useful

• We proposed an alternative view of parallelism

  • State exposure coupled with collaboration, facilitates a new paradigm for writing parallel algorithms

  • It provides improved computational efficiency, orientation and dynamic scheduling.

• We showed improvements for K-Means and GSat

• We are exploring more applications of the collaborative paradigms and different representations of the CST
Q & A

• ?
Q & A

- ?
- ?
- ?
- ?
Q & A

• ?

• ?

• ?

• ?

• ?