Dynamic Prioritization for Parallel Traversal of Irregularly Structured Spatio-Temporal Graphs

Vol. 48, Issue 899.8s, 0.08ms
Vol. 100, Issue 216.7s
Vol. 8K, Issue 3K
Vol. 0.1e3
Vol. 0.4s
Vol. 2.3s
Jaguar@ORNL
Vol. 100, Issue 24

Status Quo: Coscaling across Algorithm-Architecture Concurrency

Fast Adaptive Algorithms: linear complexity, multiscale in space and time.

New Architectures: multigranularities in memory space and processing time.

Objective: Fast Parallel Traversal of a ST-DAG.

- Sufficient resources: prioritize by the absolute node ranking (dependency depth).
- Insufficient resources: prioritize by the conditional ranking:
  - Increase/Maximize the concurrency width.
  - Associate each frontier node $c$ with an order pair $(d^{(1)}_m(c), n_r(c))$.
  - $n_r(c) = \{d : (c, d) \in \mathcal{E}(G), \deg^- (d) = 1\}$
  - Node $v$ has higher priority than node $u$ if in lexicographical order.

- Illustrations:
  - (DAG-i): with conditional ranking among frontier nodes.
  - (DAG-i'): with random tiebreaker.

Parallel FMM promises a greater impact on algorithm-architecture coscaling and shows existing algorithm-architecture gap.

Dynamic Prioritization: Dependency Depth & Concurrency Width

Experiment: Parallel FMM on Multicore Machines

Computation Problem:

$$\Phi(x_i) = \sum_{i \neq j} e^{-|x_i - x_j|}, x_j \in \mathbb{R}^3, \lambda \in \mathbb{R}^+.$$ Implementations:

- Multiple priority queues for upward, bipartite, and downward traversals, each containing spatial node IDs with time stamp while nonempty.
- Concurrent operations on multiple priority queues by all active threads without centralized control.
- All threads are created only once, initially with upward traversal assignment.
- An available thread is assigned with the task of the highest conditional rank.
- Update of priority queues upon each task completion.
- Mutual exclusion mechanism is used.
- Significant reduction in the number of mutex variables through changing read, write, and aggregation patterns.

Setup:

- 2X 12-core AMD 6168 processors & 64 GB memory.
- For weak scalability tests: the problem size per core is kept fixed at 3,750,000 for $10^{-8}$ relative accuracy, and at 625,000 for $10^{-4}$ relative accuracy.
- For strong scalability tests: the total problem size remains fixed at 15,000,000.

Experimental results:

Main Challenges in Distributed Implementation

- Distributed priority queues & queueing operations.
- Migration of tasks & data across a network of processors and memories.

Key References & Support


The work is supported in part by NSF-0905164, NSF-0905473 and the MC-IRG.