Transposition Driven Work Scheduling in Distributed Search

John W. Romein
Aske Plaat
Henri E. Bal

Jonathan Schaeffer

Department of Computer Science
vrije Universiteit  amsterdam

Department of Computing Science
University of Alberta
Transposition tables

• Transposition table contains values for positions that have been searched before
• Accessed 1000s times per second
• Problem: how to share transposition tables efficiently on a distributed-memory system
• TDS solves this problem for 1-person games (puzzles)
• 2-person TDS would be more complicated
Outline

- Traditional parallel search + distributed transposition tables
- Transposition Driven Scheduling
- Performance comparison
- Summarize TDS advantages
- Conclusions
Traditional search: the search algorithm

- Parallel IDA*
  - Uses work-stealing
- Many games have *transpositions*
  - Same position reached through different sequence of moves
- A transposition table caches positions that have been analyzed before
Traditional search: the distributed transposition table

Partitioned (based on a hash function)
  More processors ⇒ increased table size
  high lookup latency (blocking reads)

Replicated
  updates expensive
    (broadcast writes)
Transposition Driven Scheduling

- Integrates IDA* and transposition table
- Send work to table (non-blocking)
- Advantages:
  - All communication is asynchronous
  - Asynchronous messages can be combined
Performance

- **Approaches:**
  - **TDS**: Transposition Driven Scheduling
  - **WS/Part**: Work Stealing + Partitioned tables
  - **WS/Repl**: Work Stealing + Replicated tables
Performance

- Applications:
  - 15-puzzle
  - double-blank puzzle
  - Rubik’s cube
- 128 Pentium Pros, 1.2 Gbit/s Myrinet
- Highly optimized
  - WS/Part uses customized network firmware
  [ICPP ’98]
Performance (Cnt’d)

15-puzzle

double-blank puzzle

Rubik’s cube
Performance breakdown
(double blank puzzle)
TDS advantages

- No duplicate searches
- Table access is local
- Communication is non-blocking
- Scales well
- No separate load balancing
Conclusions

• TDS
  – scheduling algorithm for parallel search
  – integrates parallel IDA* with transposition table
• Performance comparison
  – TDS scales well and outperforms work-stealing
• Illustrates power of asynchronous communication
• Same approach used to solve Awari