A Desktop Grid Computing Service for Connect6

I-Chen Wu*, Chingping Chen*, Ping-Hung Lin*, Kuo-Chan Huang**, Lung-Ping Chen***, Der-Johng Sun* and Hsin-Yun Tsou*

*Department of Computer Science, National Chiao Tung University, Taiwan
**Department of Computer and Information Science, National Taichung University, Taiwan
*Department of Computer Science, Dong Hai University, Taiwan
April 2009
Outline

• Background
• Goals
• Architecture
• Experiment
• Conclusion
Background

• Connect6
• NCTU6
• Connect6Lib – A Connect6 Editor
Connect6

• [Wu and Huang, 2005] presented the first article about Connect6.

• Features of Connect6
  – Simplicity of Rules
  – Fairness
  – High Game Complexity

• Tournaments

• Connect6 Editor – Connect6Lib

• Connect6 Program – NCTU6
Simplicity of Rules

- Black (first) places one stone.
- Both alternately places two stones.
- The one who connects up to six wins.
Fairness

• Key: Balanced
• One player always has one more stone than the other after making each move.
  1. B: #1, W: #0
  2. B: #1, W: #2
  3. B: #3, W: #2
  4. B: #3, W: #4
  5. B: #5, W: #4
  6. B: #5, W: #6
  7. B: #7, W: #6
High Game Tree Complexity

• Game-Tree Complexity: [Herik, et al. 2002] [Wu and Huang, 2005]

1. Go: $10^{360}$
2. Shogi: $10^{226}$
3. Chinese Chess: $10^{150}$
4. Connect6: $10^{140} \sim 10^{188}$
   • Between Shogi & Chinese Chess.
5. Chess
6. ...
Connect6 Tournaments

• Tournaments:
  – Human vs. Human
    • NCTU-Cup Connect6 Open Tournaments (since 2006)
    • Promote-to-Dan Connect6 Contests (since 2007)
    • Russia Connect6 Open Tournament (since 2007)
    • World Connect6 Open Tournament (since 2007)
    • …
  – Computer vs. Computer:
    • Computer Olympiad (since 2006)
    • Chinese Computer Games Conferences-CCGC (since 2007)
  – Computer vs. Human
    • Man-Machine Connect6 Contest (since 2008)
NCTU6 – A Connect6 Program

• A program developed by the team at NCTU.
  – led by I-Chen Wu.

• Records:
  – 2006: The 11th Computer Olympiad
    • Won the gold
  – 2008: The 13th Computer Olympiad
    • Won the gold
  – 2007: Go Champion Jun-Shing Chou (棋王周俊勳)
    • NCTU6 won Chou.
  – 2008: Go Champion Jun-Shing Chou (棋王周俊勳)
    • NCTU6 won Chou again.
  – 2008: The First Annual Man-Machine Connect6 Contest
    • Won 11 among 12 games against top Connect6 players in Taiwan
NCTU6
NCTU6-Verifier

- Modified from NCTU6 [Wu and Lin, 2009]
  - Verify whether one player wins in a position.
  - If not winning,
    - Find the **defensive moves**.
      Listed as (11, 11)(5, 5), ...etc as the right Figure.
Connect6Lib – An Connect6 Editor

• Modified from RenLib by [C.P. Chen 2009].
  – RenLib is an editor for Renju (professional version of five-in-a-row).
Connect6Lib – An Connect6 Editor

• Features:
  – Move Tree
  – Comments
  – Hints
  – Programs
More Openings for Connect6

In order to promote Connect6,

• Need to develop more openings and puzzles.
  – Reason:
    • Lack of openings and puzzles, since Connect6 is a very young game.

But, how?

• Have some Connect6 experts:
  – Use NCTU6 to generate good moves.
  – Use NCTU6-Verifier to generate all defensive moves.

• Implement an automatic opening generating system.
Integration of Connect6Lib, NCTU6 and NCTU6-Verifier

• Add NCTU6 into Connect6Lib.
• Add NCTU6-Verifier into Connect6Lib.

• Problem:
  – Both takes huge amount of times.
  • NCTU6: takes about 60 sec per move.
  • NCTU6-Verifier: takes 1sec ~ 10hr per verification.
NCTU6 in Connect6Lib

Click “ab” button

Position:
;B[JJ];W[JH];W[HL]
;B[KK];B[II];W[HH]
;W[IK]

Return good moves:
;B[KM];B[LL]
;B[KM];B[KJ]
;B[KJ];B[HJ]

Generate these moves.
NCTU6-Verifier in Connect6Lib

1. Click "cw" button

2. Position format:
   - B[JJ]; W[JH]; W[JL]
   - B[KK]; B[II]; W[HH]
   - W[IK]; B[KM]; B[LL]

3. Results for all defensive moves:
   - W[MM]; B[KI]
   - B[MM]; B[NJ]
   - B[NN]; B[NJ]
   - B[NN]; B[KN]

4. Generate defensive moves base on results
Outline

• Background
• Goals
• Architecture
• Experiment
• Conclusion
Our Goals

• Parallelize jobs
  – NCTU6 (takes about 1 min.)
  – NCTU6-Verifier. (takes about 1 sec to 10 hr.)

• on idle desktops.
  – Why not use the idle cycles of all desktops to help Connect6 experts speed up.
    • For example, while finding all defensive moves, try some good moves in other positions.
Parallelization of

• A scenario:
  – Prove
    • we win in “Root”.
• Parallelization
  – “NCTU6-Verifier”
  – “NCTU6”
Job Priority

• Jobs:
  – NCTU6-Verifier
  – NCTU6

• Job priority:
  – Experts or opening generator use priorities to determine which job to run next.
  – For example,
    • For squares generated by NCTU6, choose the one with high chances to win.
    • For circles generated by NCTU6-Verifier, choose the one with high chances not to win.
Abortion (Pruning)

- Abortion
  - or called Pruning in Game Tree Search.
  - A square wins → other squares are aborted.
  - A circle loses → other circles are aborted.
Abortion (Pruning)

NCTU6-Verifier:
• verify whether we win.
• Generate defensive moves by opponents.

NCTU6:
• Generate our good moves.
Abortion (Pruning)

Our good moves...
Abortion (Pruning)

NCTU6-Verifier:
• verify whether we win.
• Generate defensive moves by opponents.
Abortion (Pruning)

Case 1

Win
Abortion (Pruning)

Case 2

Root

... ... ...

Lose Lose ... Lose
Abortion (Pruning)

Case 2

Root

Lose

...Lose ...

Lose

...Lose
Features of the Parallelism

• Highly dynamic
  – Abortion and Priorities may greatly change the result including performance.
    • There could be more than one winning move.

• Anomalous Speedups. [?]
  – Similar to parallel alpha-beta search.
The Requirements of Desktop Grid

• Need fast response times.
  – Reasons:
    • Experts sometimes want to see the results as fast as possible.
    • A result may cause huge pruning.

• Use push technology
  – Use pull technology in most traditional desktops.
  – Prefer to connection-oriented lines, such as TCP.

• Penetrate Firewalls/NATs.
  – Some workers in our system may be inside Firewalls/NATs.

• Try not to overlap two jobs in a one-core desktop.
  – Simply slow down the response time.
Outline

• Background
• Goals
• Architecture
• Experiment
• Conclusion
Current Architecture

Our Desktop Grid System

Connect6Lib is connected to desktops
Finite State Machine for a Job

Initial

Put in waiting queue

Waiting

Dispatch to worker

Running

Worker error occur

Finish

User abort

User abort

Abort

End
Current Policy

As long as some desktops are free, dispatch jobs to them, even if these jobs have low priorities.
Potential Problem 1 – NAT Penetration

Our Desktop Grid System

NAT port mapping is required.
Potential Problem 2 – Job Overlapped

Our Desktop Grid System

Overlapped: Not the best scheduling.

Connect6Lib A

Connect6Lib B
Solution for These Potential Problems

Connect6Lib A

Dynamically dispatch jobs to desktops

Broker

Our Desktop Grid System

Allocated for A

Allocated for B

Connect6Lib B
Solution for These Potential Problems

B only needs one job.
Ongoing Projects

• Centralize job distribution.
  – Solve the potential problems.

• Decentralize job distribution.
  – Make it scalable.

• Automate the job distribution of Connect6 jobs
  – Use some algorithms like Proof Number Search.
  – Distribute jobs without Connect6 experts.
Centralize job distribution

User

Broker

User

User

Monitor

Worker

Worker

Worker

Monitor

Monitor

Monitor
Decentralize job distribution

User → Broker → Worker

User → Broker → Worker

User → Broker → Worker

Monitor → Broker → Worker

Monitor → Broker → Worker

Monitor → Broker → Worker
Outline

• Background
• Goals
• Architecture
• Experiment
• Conclusion
Experiments

• A winning position for Black
  – Used to test our desktop grid system.
Data for the Case

- #Cores: 28
- # Jobs: 10901
  - #NCTU6: 9434
  - #NCTU6-Verifier: 1220
  - #Aborted job: 247
- Averaged time per job: 155.148 sec
  - Average for NCTU6: 156.449 sec
  - Average for NCTU6-Verifier: 515.248 sec
  - The time for the fastest job: 2.422 sec
  - The time for the slowest job: 19 hr, 42 min, 5 sec.
- Aggregate time: 584 hr, 35min, 51sec.
- Total time: 23 hr, 49min, 11sec.
Outline

• Background
• Goals
• Architecture
• Experiment
• Conclusion
Conclusion

• In this paper, we are proposing a desktop grid system for Connect6 applications.
  – Use push technology instead of pull technology as traditional desktop grid.
  – Use broker to coordinate workers and clients.
    • Currently, the implementation of broker is ongoing.
  – Heavily use priorities as scheduling policy.
    • Especially, avoid from overlapping two jobs in one core.
• Currently, this system already demonstrates the feasibility to speed the verification up greatly.
• This type of desktop grid systems can be applied to many other computation-bound AI applications.
References