CSE 704 Spring 2011

Computer Science and Engineering
University at Buffalo

John Longanecker

Parallel Computing with Browsers
Reversi Edition
May 2nd, 2011

Browsers

- HTML Hypertext Markup Language
- HTML is not a programming language
- One native scripting language JavaScript
- Standards exists but not always followed well

Web Architecture

- Server side
- Client side
- Client enters web address into the browser
- Browser sends a request to the web server
- Web server sends the information to the browser
- Browser receives the information

How Can We Use This?

How Can We Use This?

- The client visits a web page
- Server sends the client instructions
- The client processes these instructions with JavaScript in a browser
- Client sends back results to the web server

Additional Information

- Browsers are not made to do heavy computations
- In fact they are made to stop heavy computations
- Browsers become unresponsive
- Traditional JavaScript is single threaded
- Web workers allows JavaScript to spawn threads

Disadvantages of this Model

- Web workers are new
- Not supported by all browsers
- Internet speed
- Potential loss of computed results
- JavaScript is Slow
- Not good for real-time problem solving
- Not dependable

Advantages of this Model

- Multi-Platform (Windows, Mac, Linux)
- Easy to use
- No installation required just a browser
- No extra software required besides a browser
- JavaScript is getting faster
- Applications moving to browser
- Does work on local network
- Browsers to support hardware acceleration

How Am I Going to Use This?

Technologies Used

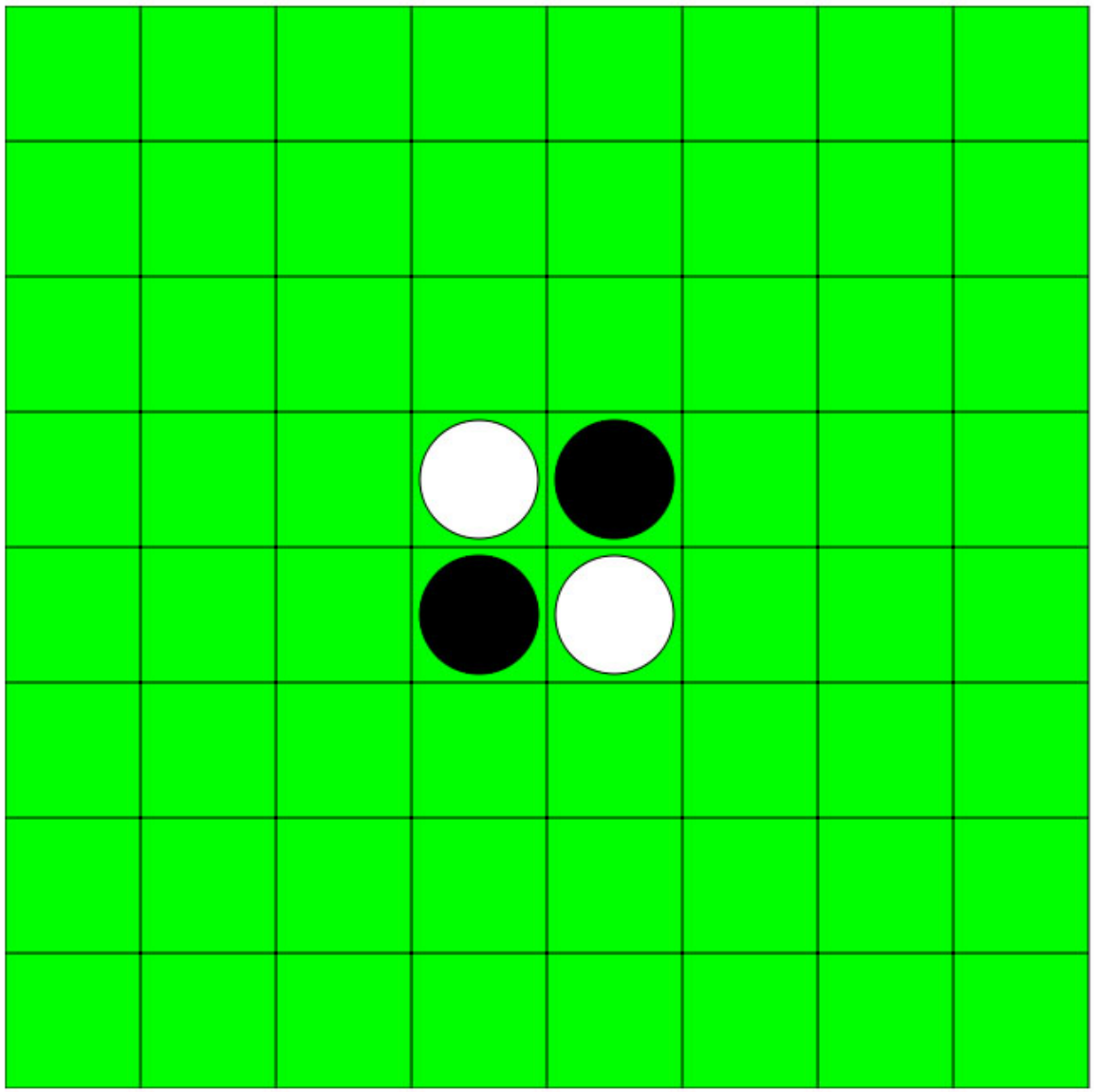
- Server Side: PHP, MySQL, (VPS)
- Client Side: JavaScript (Chrome)

- Server Side: Why PHP and MySQL?
- Client Side: Why not Java or Flash?

Scheduling

- Client receives an unique identifier
- The client checks out a board position
- Board positions that have not been checked back in 4 minutes get put back into the pool
- Each board state has its own row in the database
- Results are written back to the database
- The board state is now flagged as being used and can never be checked out again

Reversi



How it works

1. Start game by going to domainname.com/play
2. Computer player goes first
3. Calculates first 2 levels of potential moves
4. Send this board state to the server
5. Server then writes each state to its own row in the database
6. Computing clients go to domainname.com/compute
7. Computing clients then receives a board state
8. Traverses 4 levels then runs a Minimax algorithm
9. Sends the results back to the server
10. Once all board states are computed the playing client downloads the results from the server
11. Playing client runs the same Minimax algorithm on the last two levels
12. Computer plays the best possible move

Potential Additions

- Opening Book
- Interactive Site Design
  - e-mail invites for computing clients
- Multiple Games at Once
  - Unique computing client URL
- Better board evaluation
  - Alpha Beta Pruning
  - Add in better board edge strategies
    - (http://home.datacomm.ch/t_wolf/tw/misc/reversi/html/tuning.html)

Problems

- JavaScript code is slow
- Not using Alpha-Beta pruning to reduce the number of branches
- Board evaluation numbers are fixed

Thank You For Your Time
Contact Info:

johnlonganecker@gmail.com
http://www.johnlonganecker.com