GPU TECHNOLOGY CONFERENCE

Playing Zero-Sum Games on the GPU

Avi Bleiweiss
NVIDIA Corporation

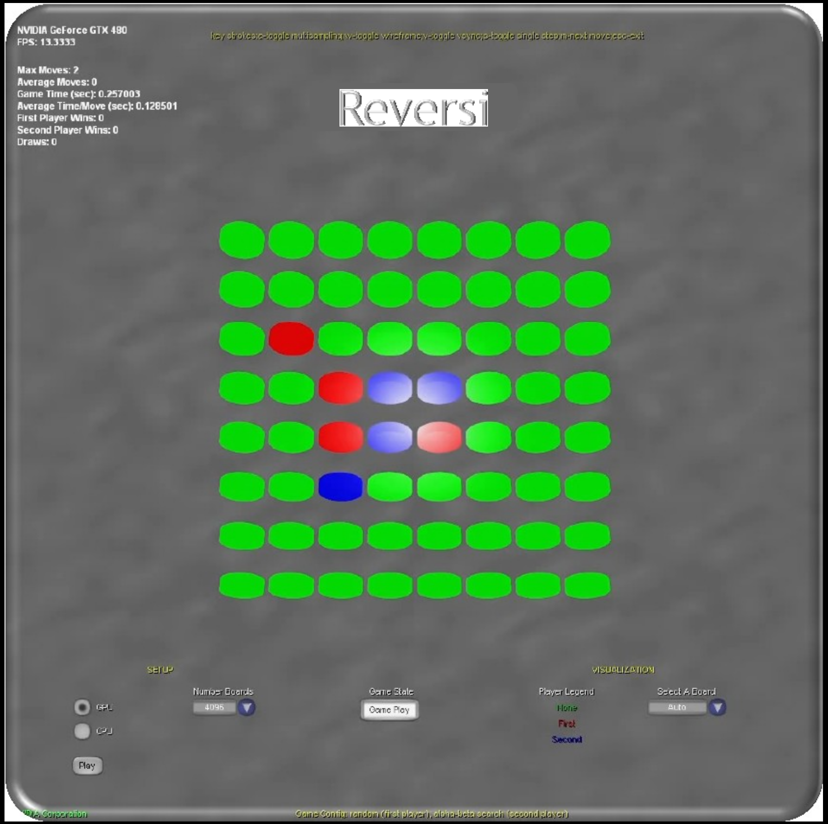
San Jose | 2010

PRESENTED BY NVIDIA.

Zero-Sum

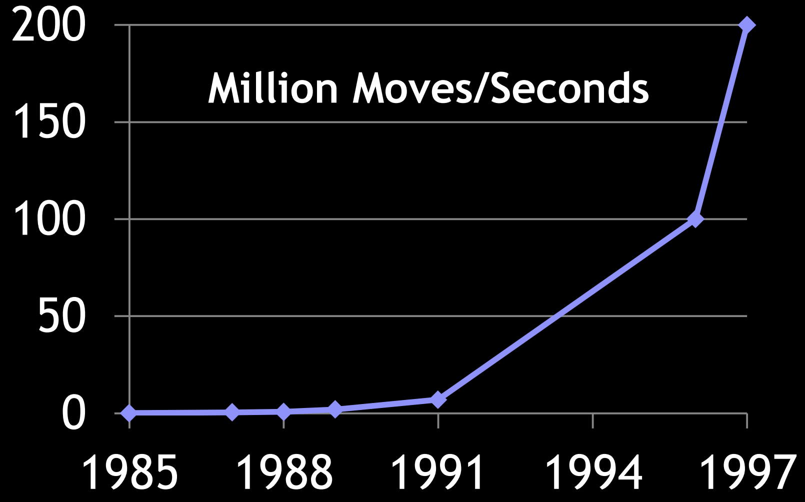
- One's gain, other's loss
- Perfect info
- Multi player game
  - Simple and involved

Matching Pennies	Head	Tail
Head	1,-1	-1,1
Tail	-1,1	1,-1



Deep Blue

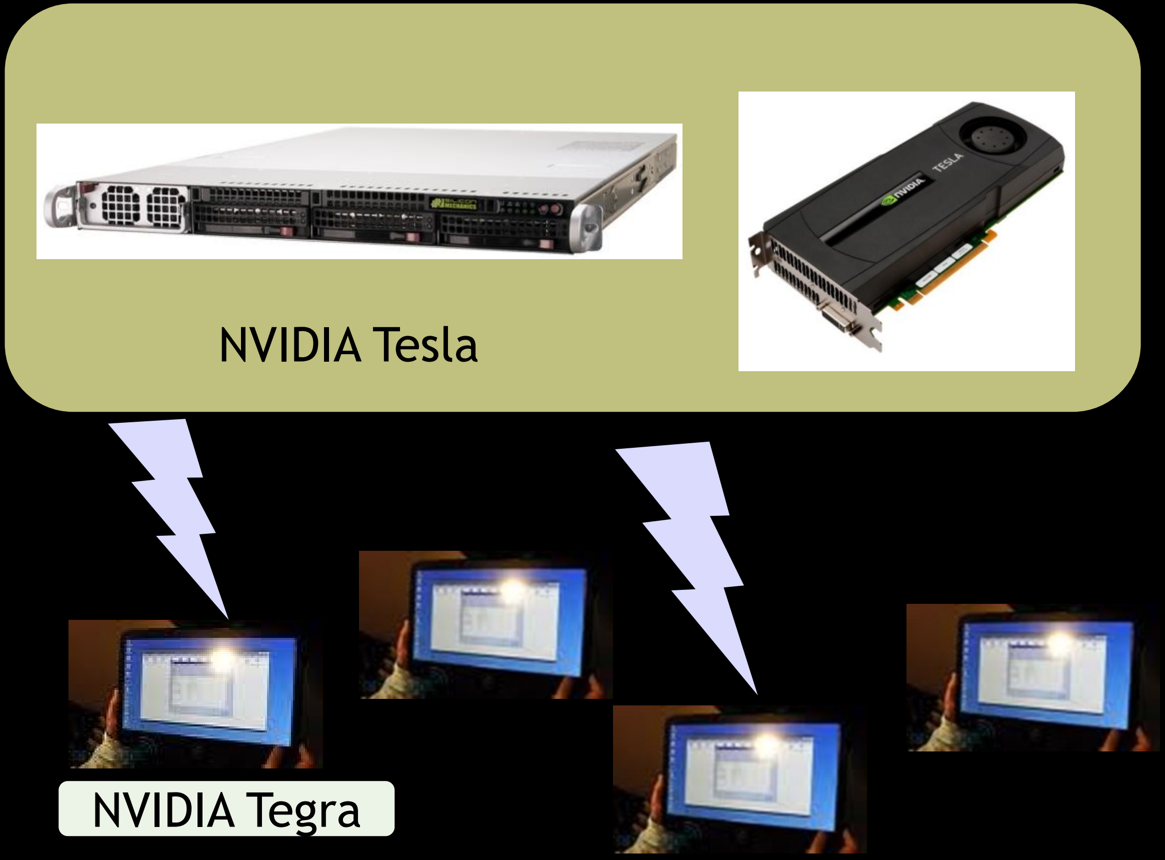
32 CPUs	
512 Chess ASICs	
1.15B transistors	



6’5” tower pair	
1.4 tons	
100M$	

Motivation

- Play as you go
  - Thousands players
- Game cloud
  - GPU computing
- Mobile computer
  - 3D graphics



Problem

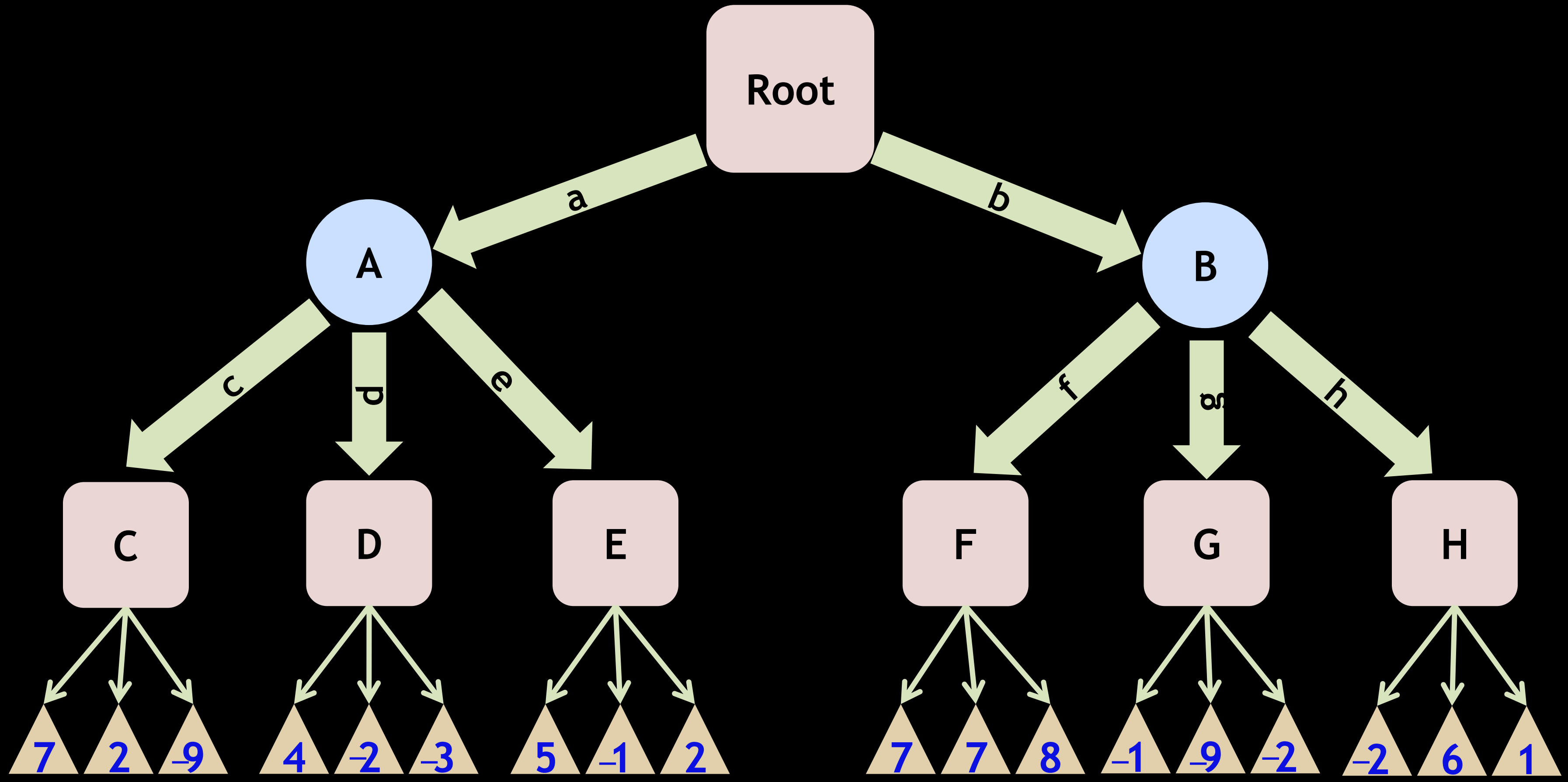
Game

- Two player
- Maximize look ahead
- Complexity
  - 10^{25} for 4x4x4 Tic-Tac-Toe
  - 10^{120} for Chess

Search

- Efficient parallel
  - Single game
  - Simultaneous matches
- Graceful stack
- Linear scale

Game Tree



Mini-Max

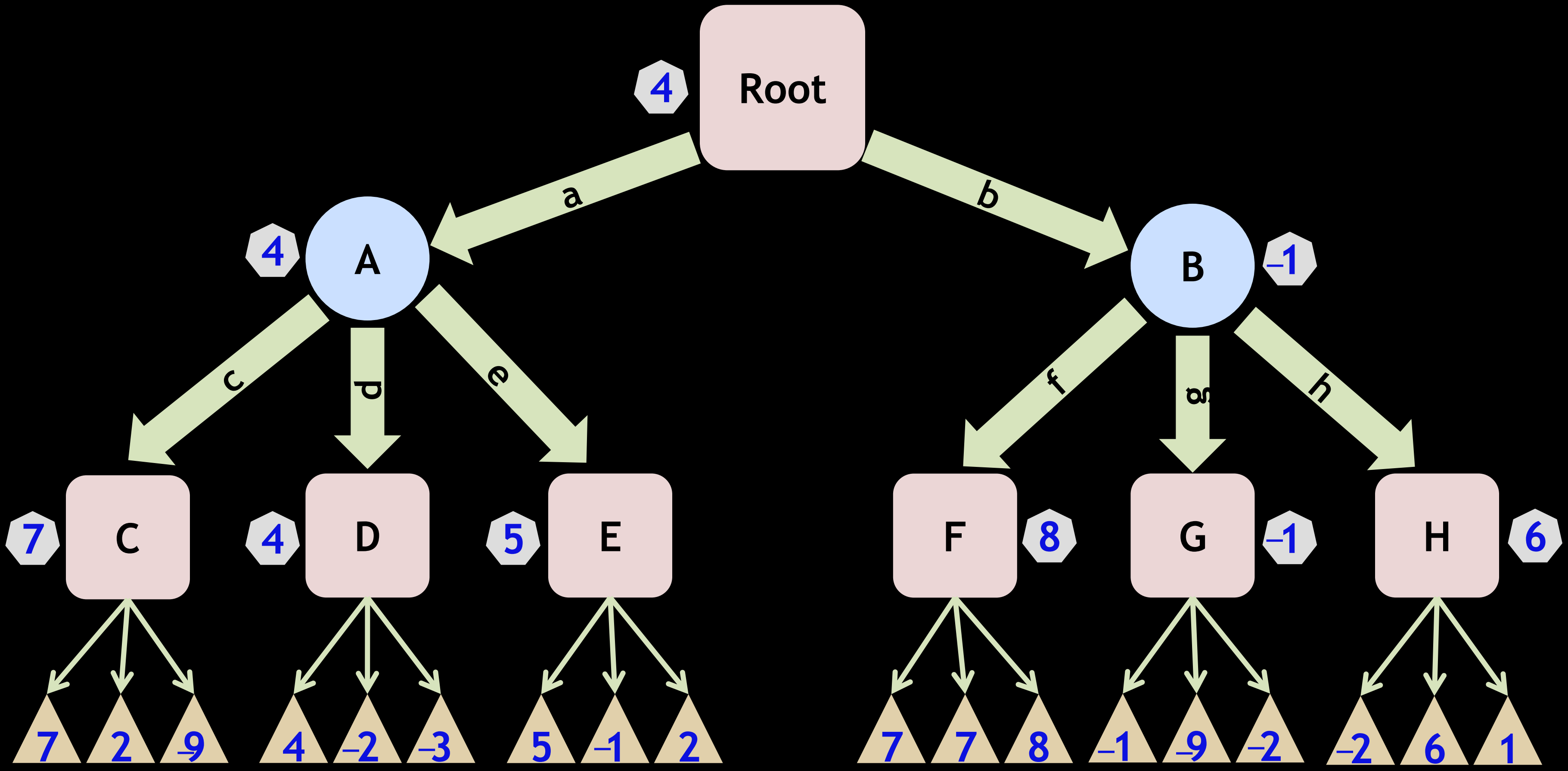
- Build and search
- Unbound DFS
  - All nodes visited
- Non tail recursive
- Depth limited

```
MiniMax(node,max)
  if(isLeaf(node))
    return evaluate(node)

  if(max) then rank ← −∞
  else rank ← ∞

  foreach(sᵢ in successors(node))do
    tmp ← MiniMax(sᵢ,!max)
    if(max) then
      rank ← MAX(rank,tmp)
    else
      rank ← MIN(rank,tmp)
  return rank
```

Principal Variation

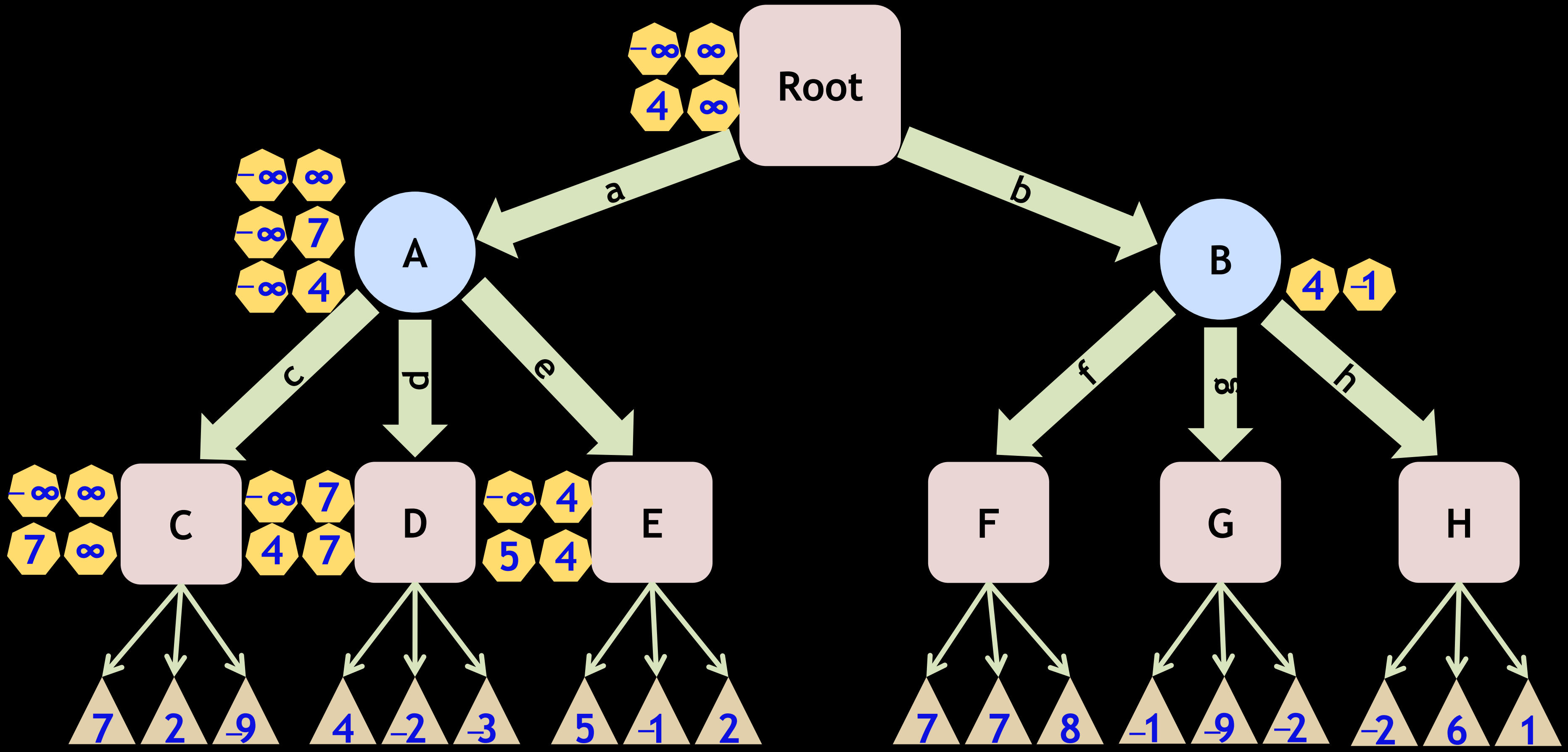


Alpha-Beta

- Enhances Mini-Max
- Elegant, efficient
  - Prunes nodes
- Perfect game
  - Possible in cases

```
AlphaBeta(node,max,α,β)
  // termination, initialization
  if (α > β) return rank
  foreach(s_i in successors(node))do
    tmp ← AlphaBeta(s_i,!max,α,β)
    if (max) then
      rank ← MAX(rank,tmp)
      α ← MAX(α,rank)
    else
      rank ← MIN(rank,tmp)
      β ← MIN(β,rank)
  return rank
```

Pruning



Root
−∞ ∞
4 ∞
a
b
A
−∞ ∞
−∞ 7
−∞ 4
B
4 −1
c
d
e
f
g
h
C
−∞ ∞
7 ∞
D
−∞ 7
4 7
E
−∞ 4
5 4
F
G
H
7 2 −9
4 −2 −3
5 −1 2
7 7 8
−1 −9 −2
−2 6 1

Optimization

Aspiration Search

- Tree value known
- Narrow window
- Re-search if fails

Principal Variation Search

- Non PV nodes
  - beta = alpha + 1
- Full window PV nodes
- Perfect ordered tree

Iterative Deepening

- Fixed depth searches
- Transposition table
- Hash branch positions

[Marsland, T. A. 1986]

Parallelism

Principal Variation Split

- Strongly ordered tree
- Synchronization bound
- Load imbalance

Young Brothers Wait Concept

- Parallel at any node
- Processor owns node
- Scales to # processors

Dynamic Tree Splitting

- Processors share node
- Global job list
- Reasonable speedup

[Feldmann, R. 1993]

Challenges

Deep recursion, limited stack

Divergent, irregular threads

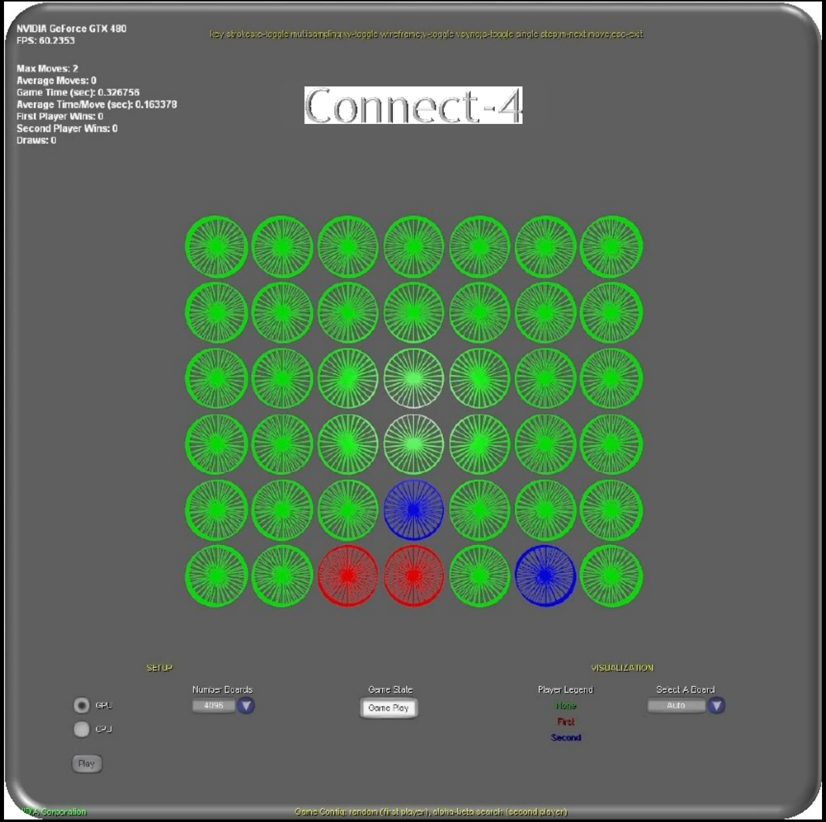
Dynamic parallelism

Low arithmetic intensity

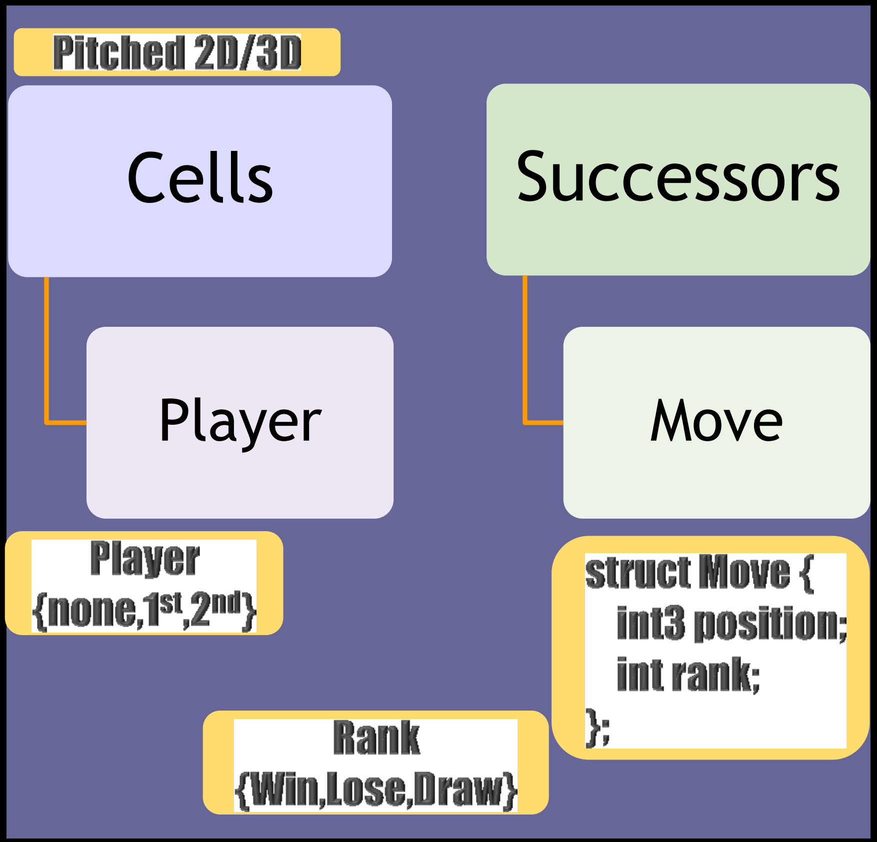
Implementation

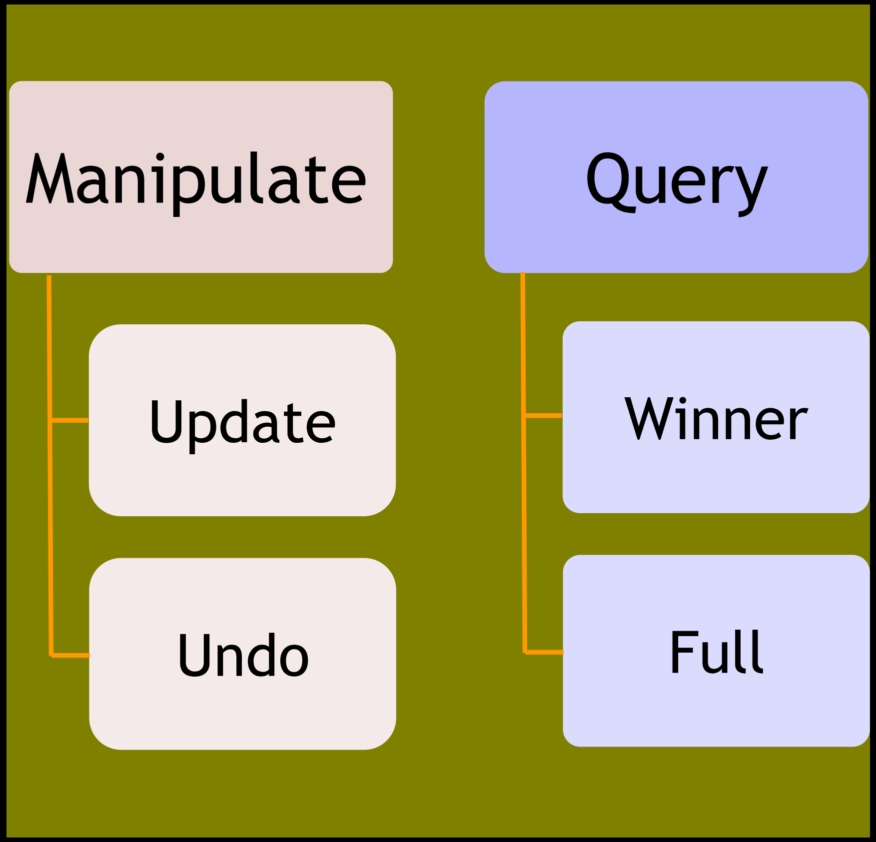
- Kernel for each
  - Mini-Max, Alpha-Beta
- Board C++ class
  - Rules specific
- Games

3D Tic-Tac-Toe	Connect-4	Reversi



Board


Pitched 2D/3D
Cells
Successors
Player
Move
Player
{none,1st,2nd}
struct Move {
int3 position;
int rank;
};
Rank
{Win,Lose,Draw}


Manipulate
Query
Update
Winner
Undo
Full

Stack

- Recursion depth >1000
- Greedy allocation

$$\mathit{StackBytes/Thread} * \mathit{SM\,\#} * \mathit{MaxWarps/SM} * \mathit{Threads/Warp}$$

- Hybrid design

Runtime/Compiler

- Local variables
- Function parameters

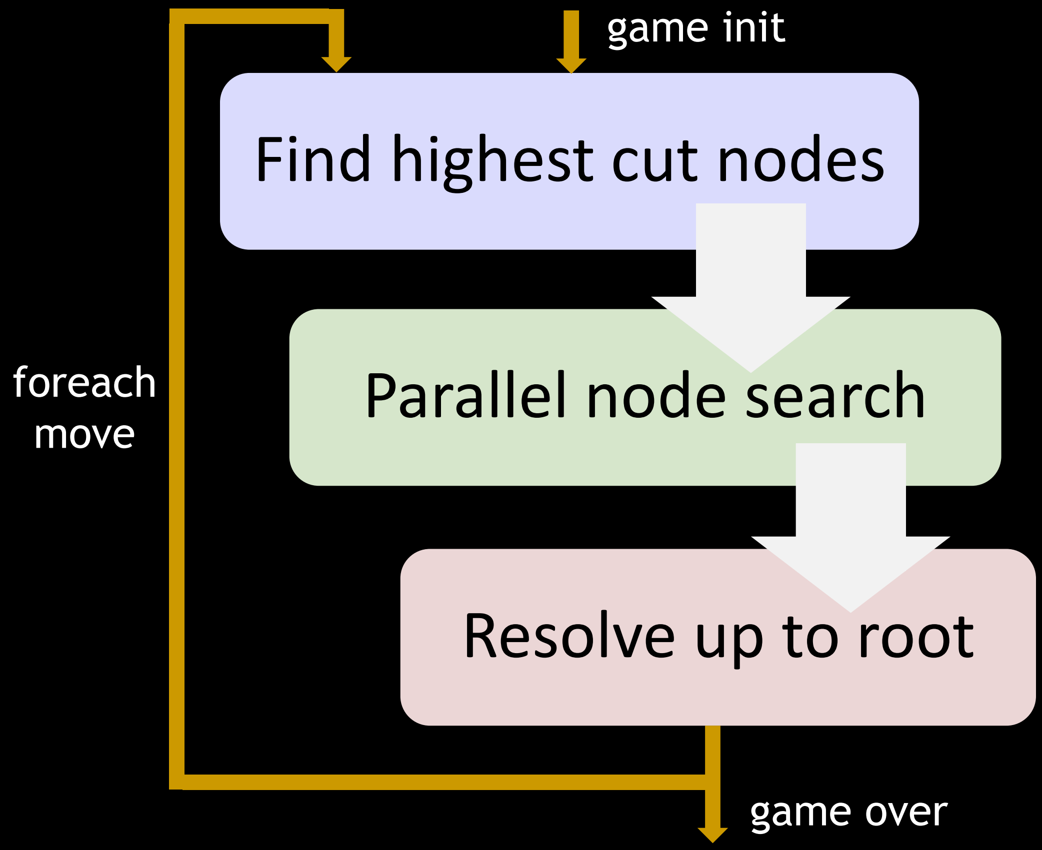
Local Memory

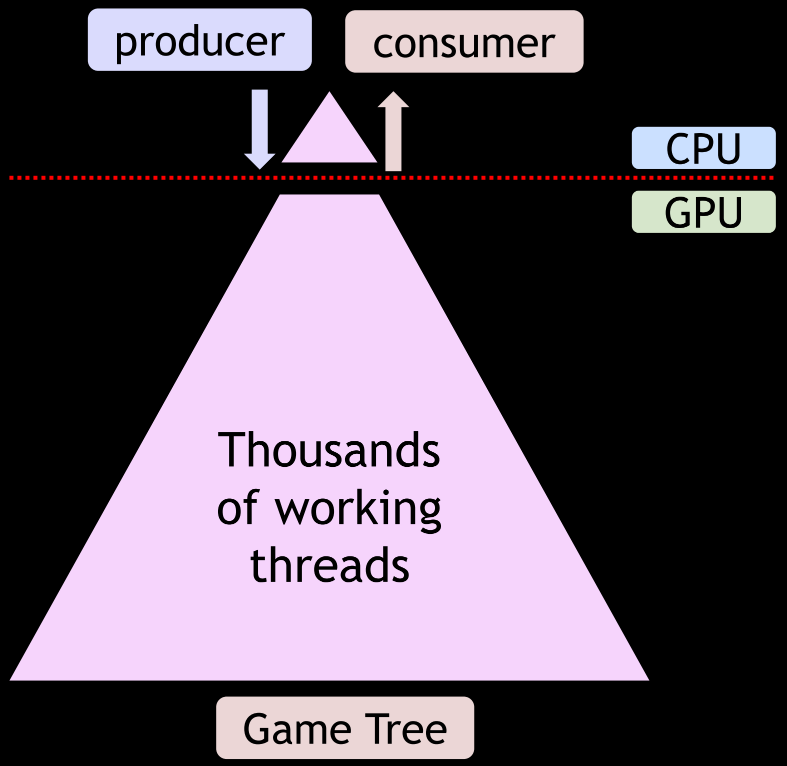
User

- Successors

Global Memory

Split


game init
Find highest cut nodes
Parallel node search
Resolve up to root
foreach
move
game over


producer
consumer
CPU
GPU
Thousands
of working
threads
Game Tree

Shared αβ

Kernel global scope

```
__device__ int galpha, gbeta;

__device__ void
resolve(int alpha, int beta)
{
  if(alpha ≤ galpha) alpha = galpha;
  else atomicMax(&galpha, alpha);

  if(beta ≥ gbeta) beta = gbeta;
  else atomicMin(&gbeta, beta);
}
```

Check private αβ

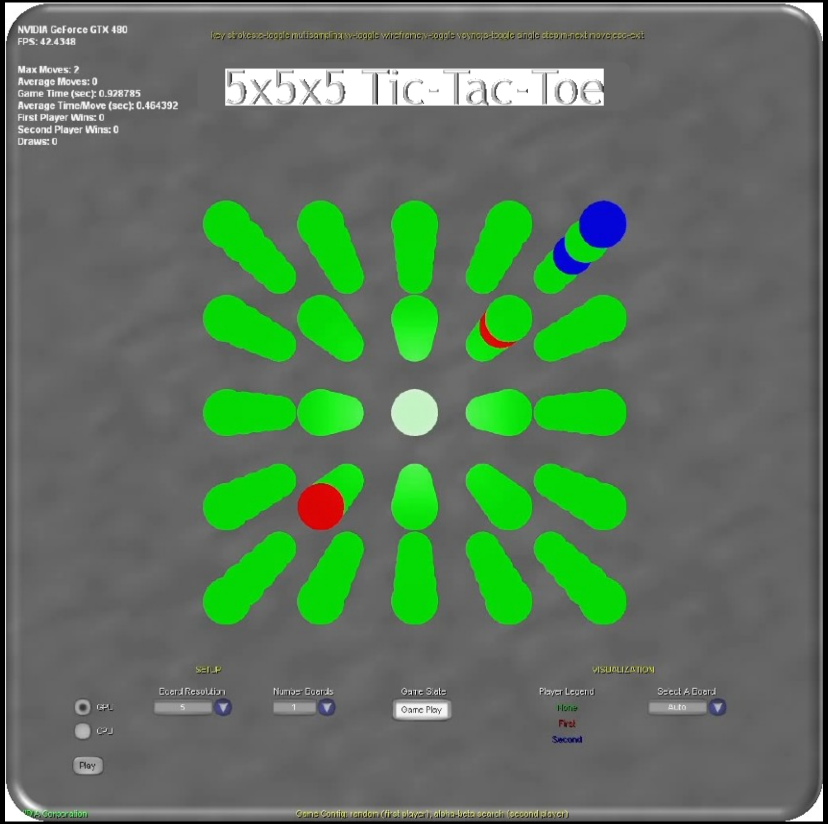
Global atomic update

GPU TECHNOLOGY CONFERENCE

Limitations

- Stack allocation
  - Bounds parallelism
- Split constraints

Depth	Tic-Tac-Toe Threads		
	3x3x3	4x4x4	5x5x5
1	650	3906	15252
2	15600	238266	1860744



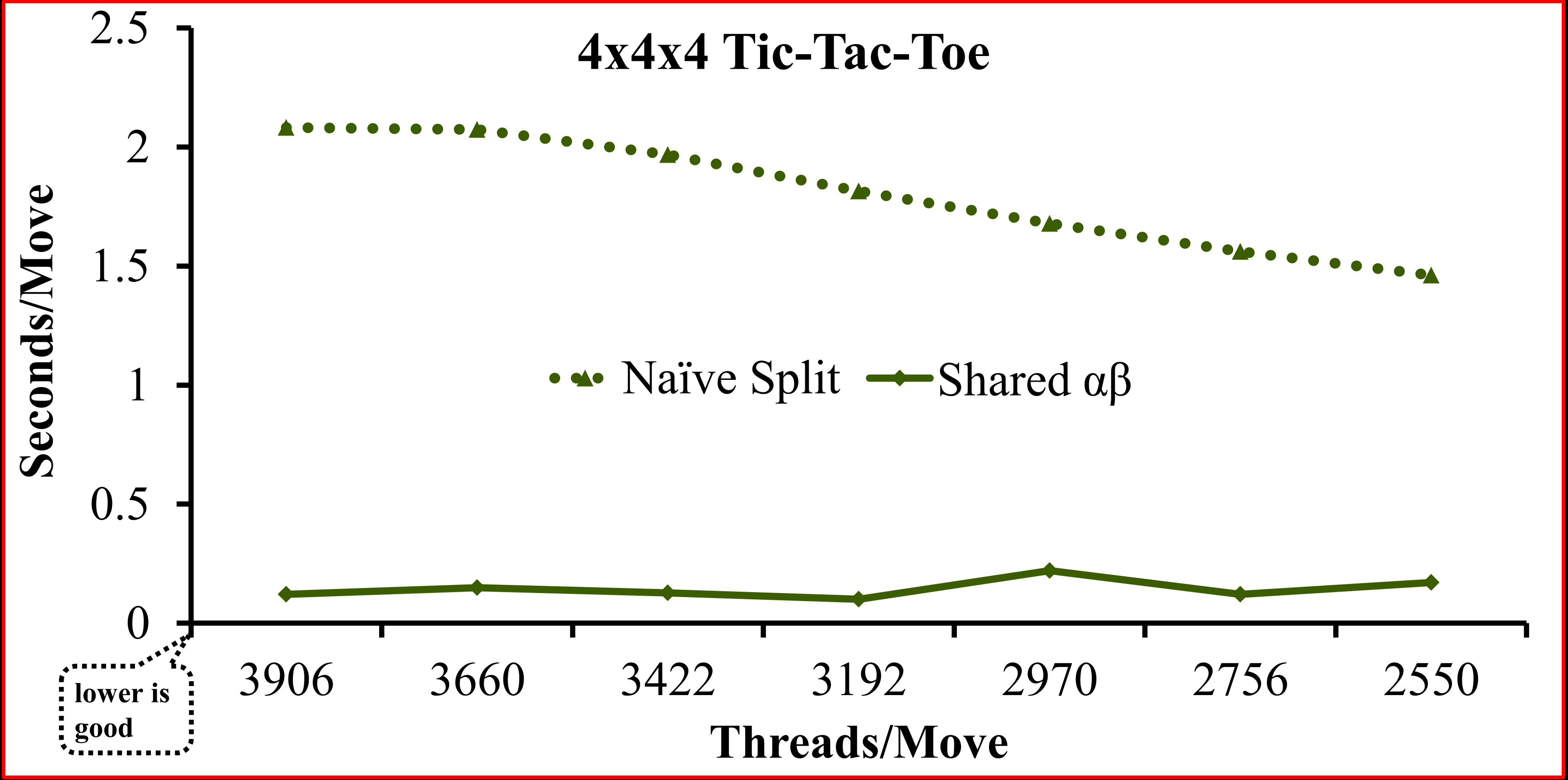
Methodology

- CUDA Toolkit 3.1, Windows
- Processors

GPU	SMs	Warps/SM	Clocks(MHz)	L1/Shared (KB)	L2(KB)
GTX480	15	2	723/1446/1796	48/16	640

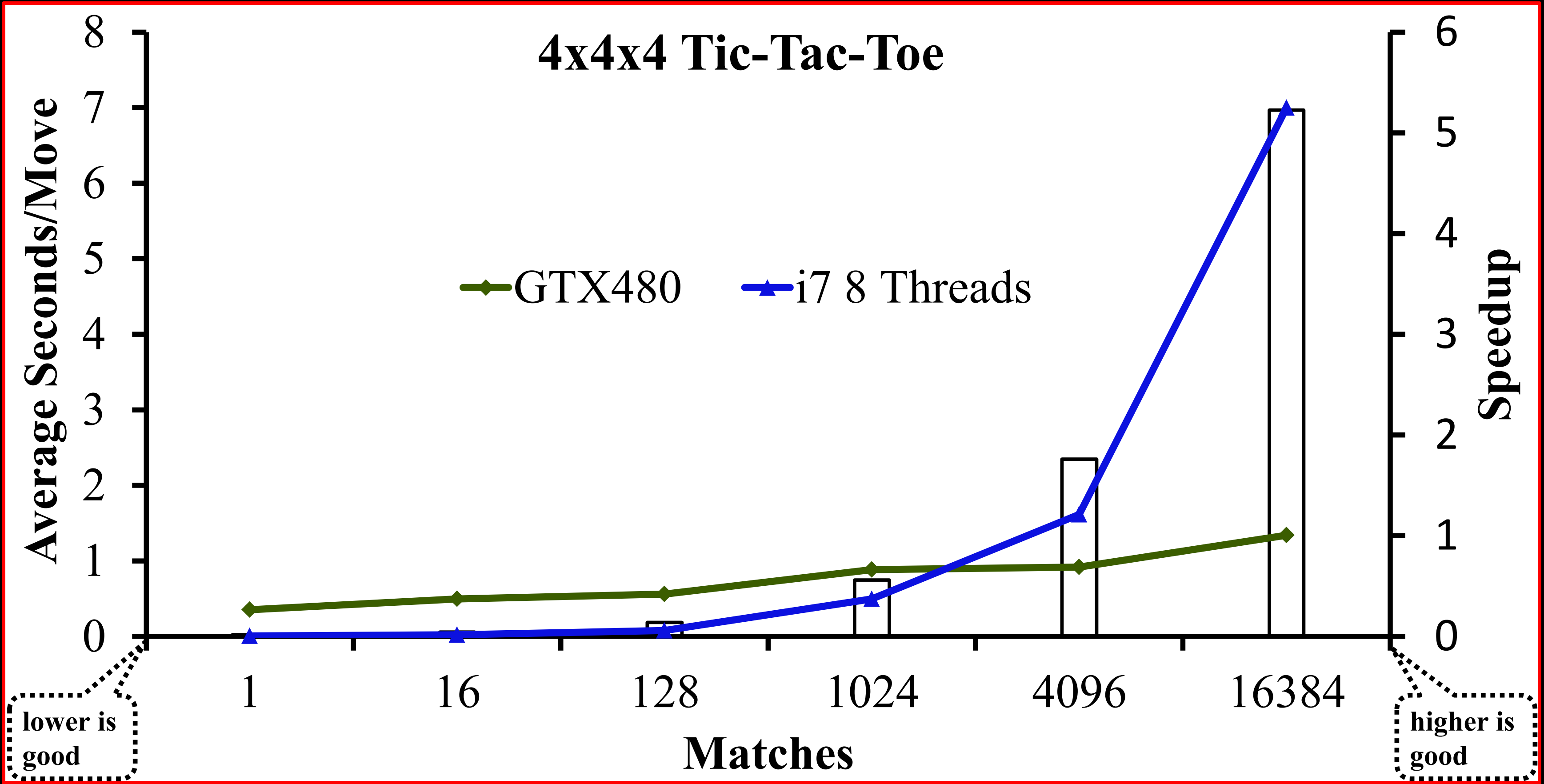
CPU	Cores	Clocks(MHz)	L1/L2 (KB)
I7-940	8	2942/(3*1066)	32/8192

Single Game

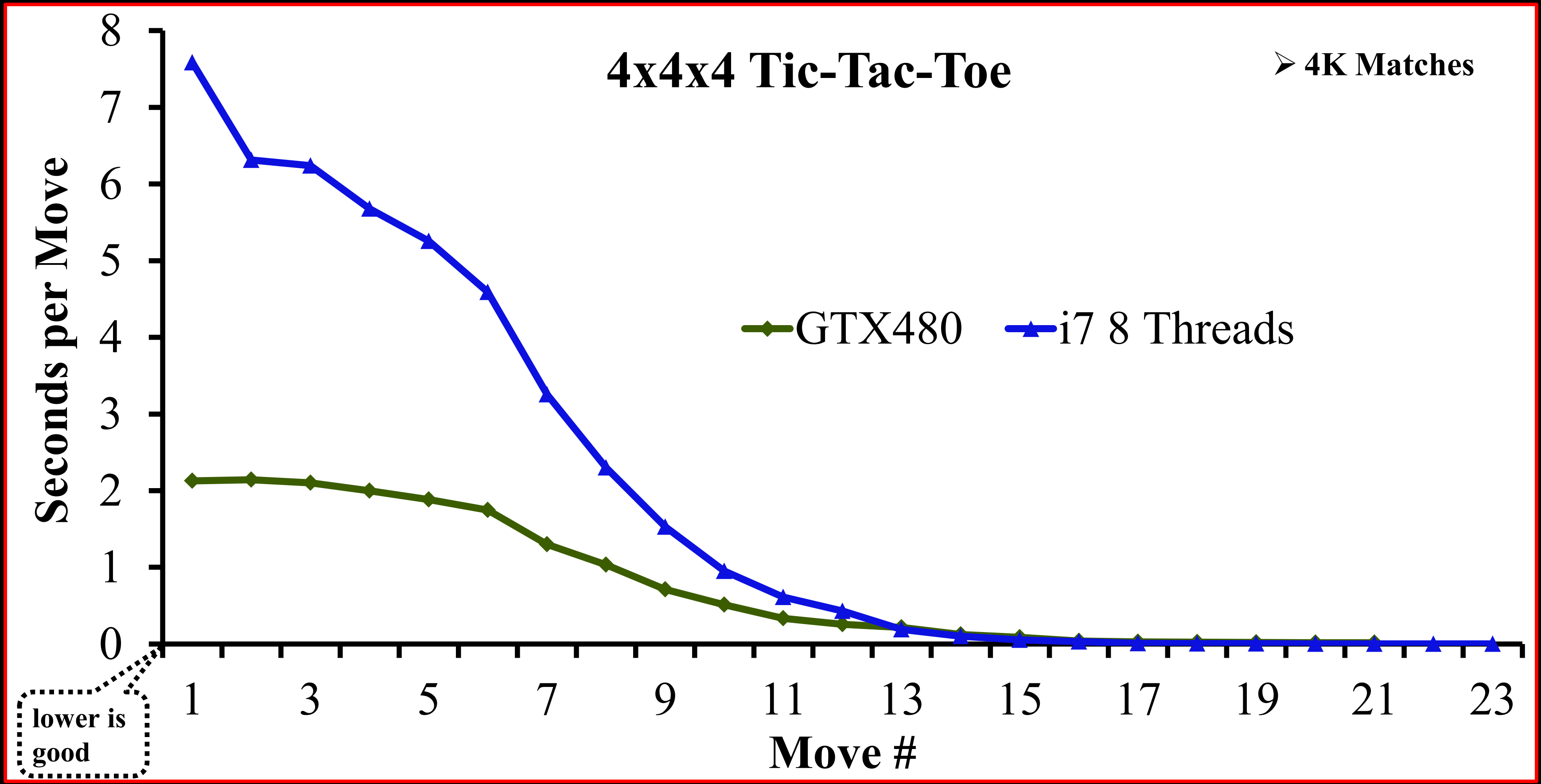


4x4x4 Tic-Tac-Toe
2.5
2
1.5
1
0.5
0
Seconds/Move
Naïve Split
Shared αβ
3906
3660
3422
3192
2970
2756
2550
Threads/Move
lower is good

Simultaneous Matches

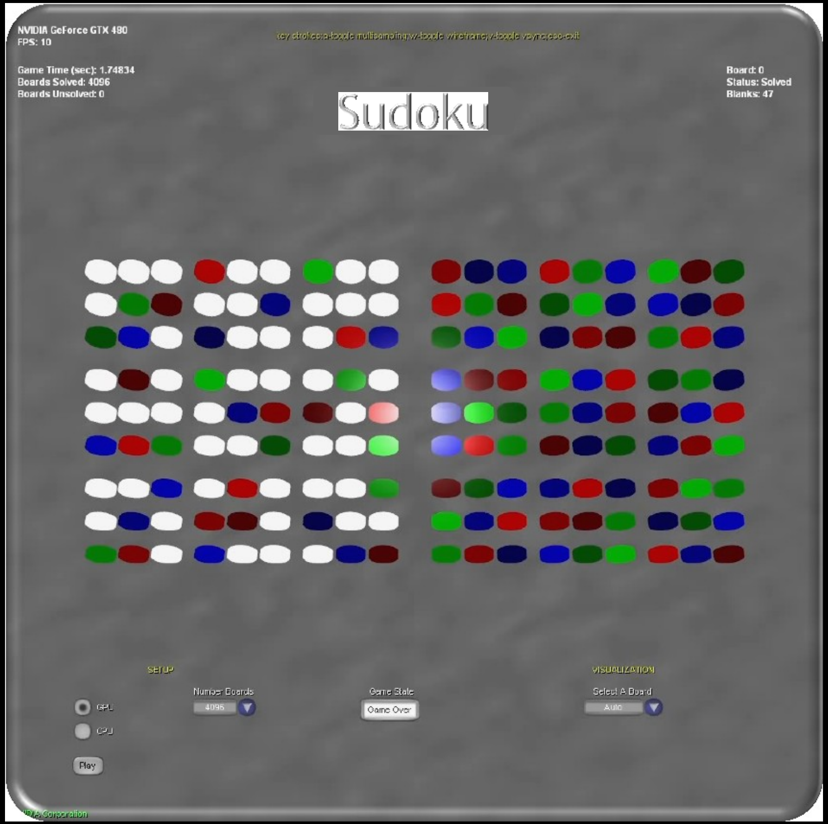


Game Analysis



Future Work

- Data packing
- Backtracking search
- Sudoku, toy game
  - Generator, solver
- Multi recursion



GPU Performance

Metric	Game	Dimension	Speedup
Shared αβ vs. Naïve Split	Tic-Tac-Toe	4x4x4	13.37X mean

Simultaneous Matches vs. CPU	Tic-Tac-Toe	4x4x4	5.22X @ 16K
	Connect-4	7x6	6.26X @ 32K
	Reversi	8x8	5.96X @ 16K

Summary

- Efficient hybrid stack
- Dynamic parallelism
- Tegra, Tesla solution
- 3D Chess on GPU!

	Deep Blue	Tesla
$/(Moves/Sec)	0.5	0.02

Courtesy freegames4all

Thank You!

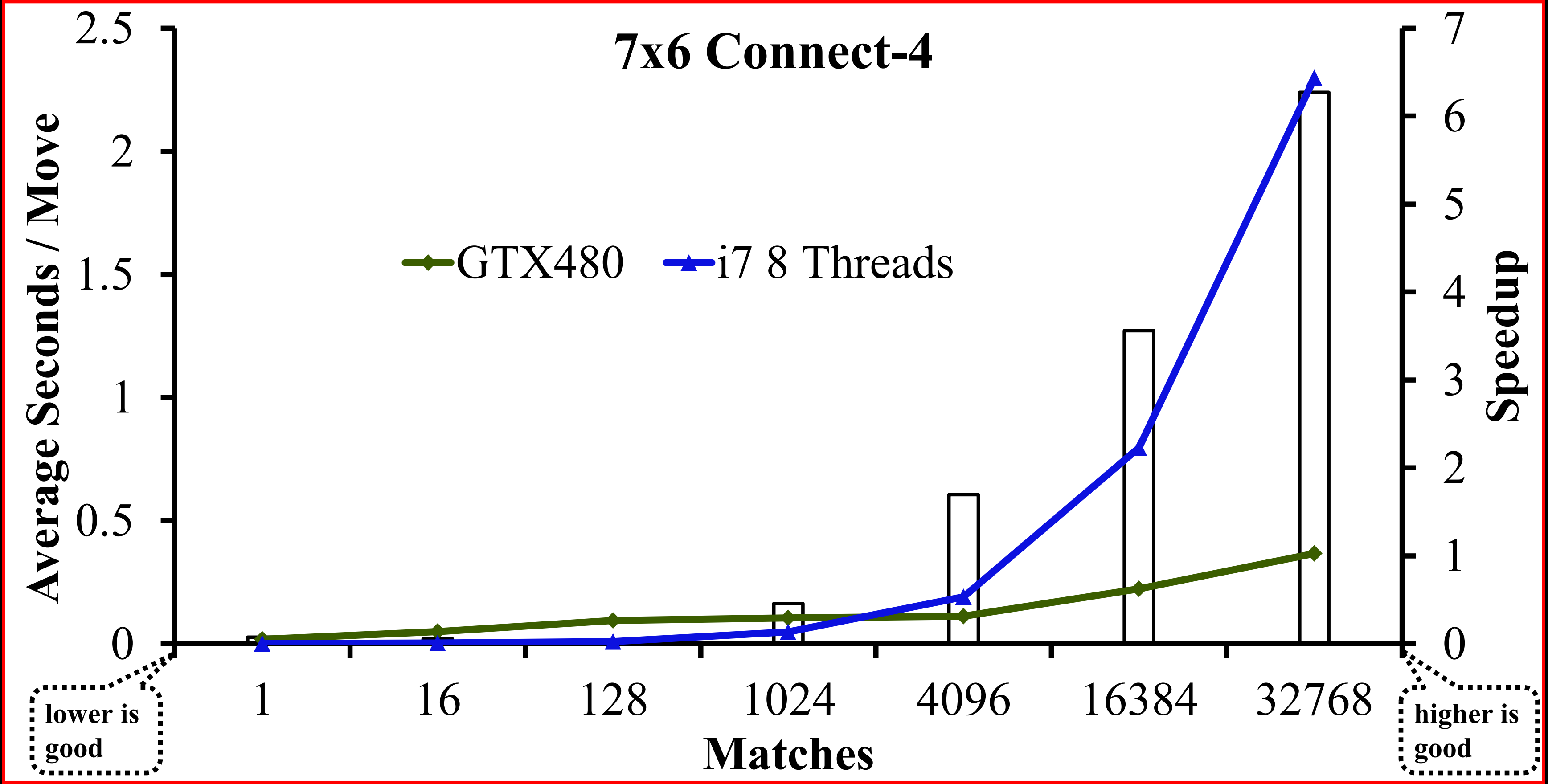
Questions?

Info

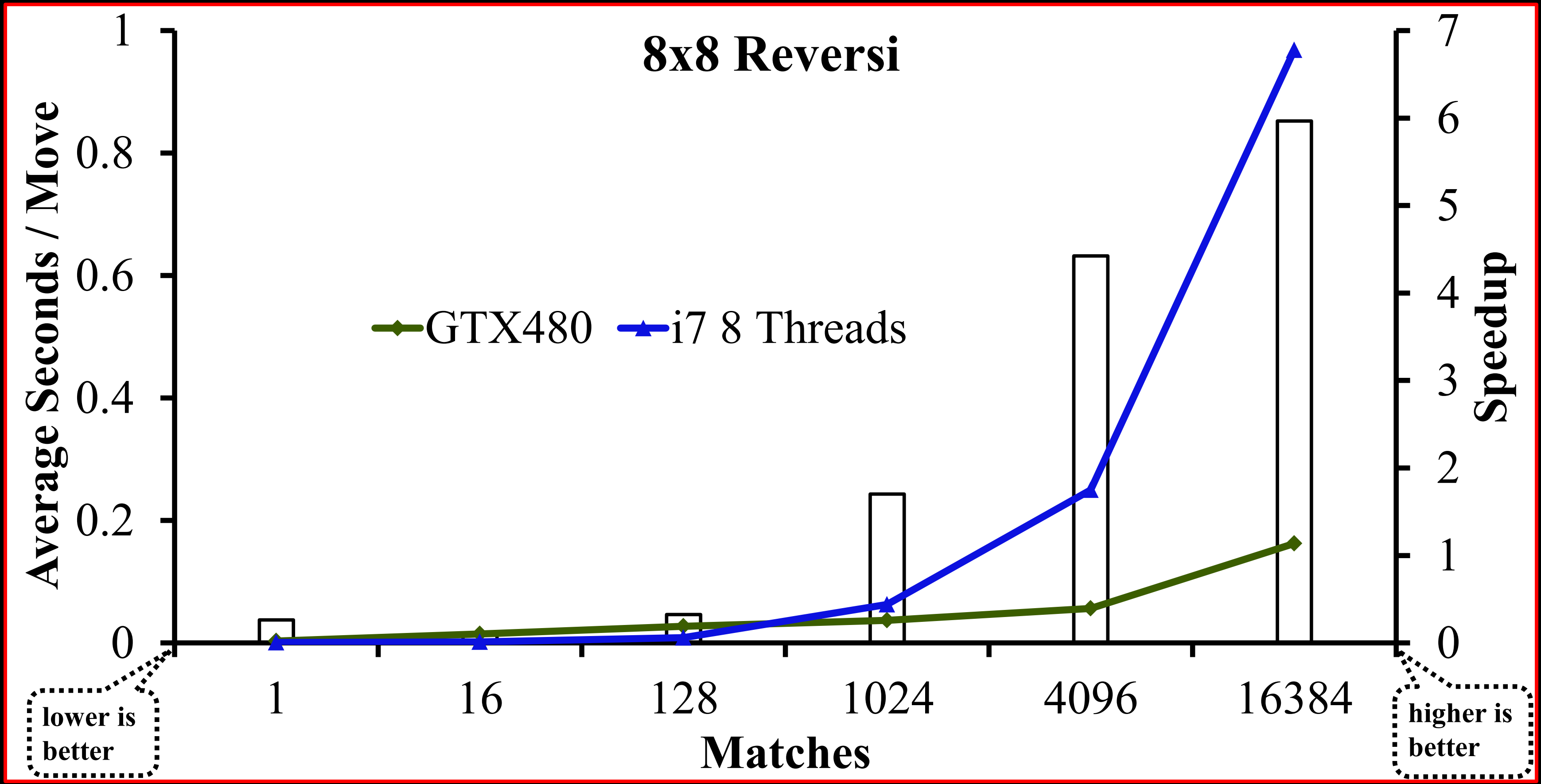
- Base
  - http://developer.nvidia.com
- GPU AI
  - Technology Preview
- Toolkit
  - CUDA Zone
- Debugger
  - Parallel Nsight

Backup

Simultaneous Matches (1)



Simultaneous Matches (2)



Simultaneous Puzzles