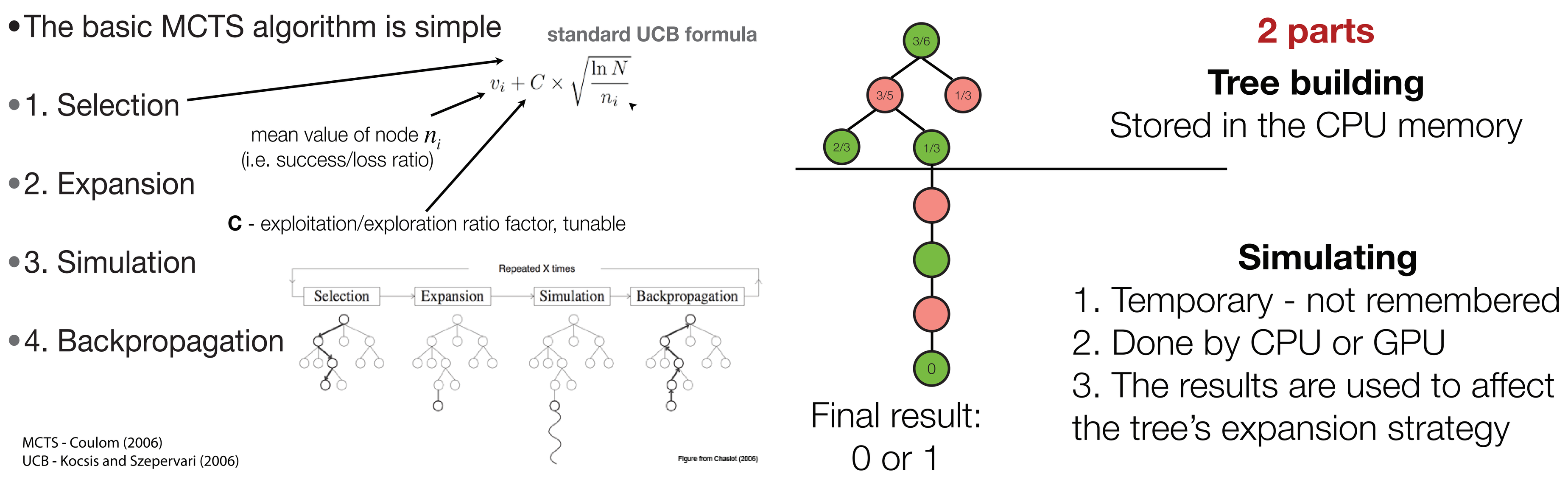
Accelerating Parallel Monte Carlo Tree Search using CUDA

Kamil Rocki and Reiji Suda
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo

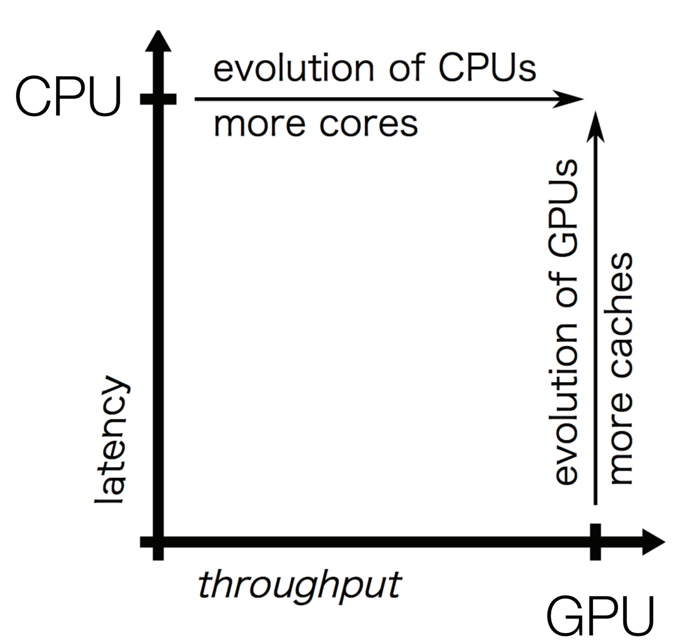
Introduction

Monte Carlo Tree Search (MCTS) is a method for making optimal decisions in artificial intelligence (AI) problems, typically move planning in combinatorial games. It combines the generality of random simulation with the precision of tree search. It can theoretically be applied to any domain that can be described in terms of state, action pairs and simulation used to forecast outcomes such as decision support, control, delayed reward problems or complex optimization.



$$v_i + C \times \sqrt{\frac{\ln N}{n_i}}$$

The motivation behind this work is caused by the emerging GPU-based systems and their high computational potential combined with relatively low power usage compared to CPUs. As a problem to be solved we chose to develop an AI GPU-based agent in the game of Reversi (Othello) which provides a sufficiently complex problem for tree searching with non-uniform structure and an average branching factor of over 8.



- MCTS has many applications already
- New ones are appearing
- **The architecture is likely to follow the trend in the future**
- Programming GPUs **may become easier, rather not harder**

TSUBAME 2.0

- CPUs - Intel(R) Xeon(R) CPU X5670 @ 2.93GHz ~ 1400 Nodes of 12 cores
- GPU - NVIDIA Tesla C2050 - 14 (MP) x 32 (Cores/MP) = 448 (Cores) @ 1.15 GHz, ~ 1400 Nodes of 3 GPUs each (around 515GFlops max capability per GPU)
- If not specified otherwise, the MCTS search time = 500 ms, and GPU block size = 128

We present an efficient parallel GPU MCTS implementation based on the introduced 'block-parallelism' scheme which combines GPU SIMD thread groups and performs independent searches without any need of intra-GPU or inter-GPU communication. The obtained results show that using my GPU MCTS implementation on the TSUBAME 2.0 system one GPU can be compared to 100-200 CPU threads depending on factors such as the search time and other MCTS parameters in terms of obtained results. We propose and analyze simultaneous CPU/GPU execution which improves the overall result.

Parallel MCTS Schemes - Chaslot et al. (2008)

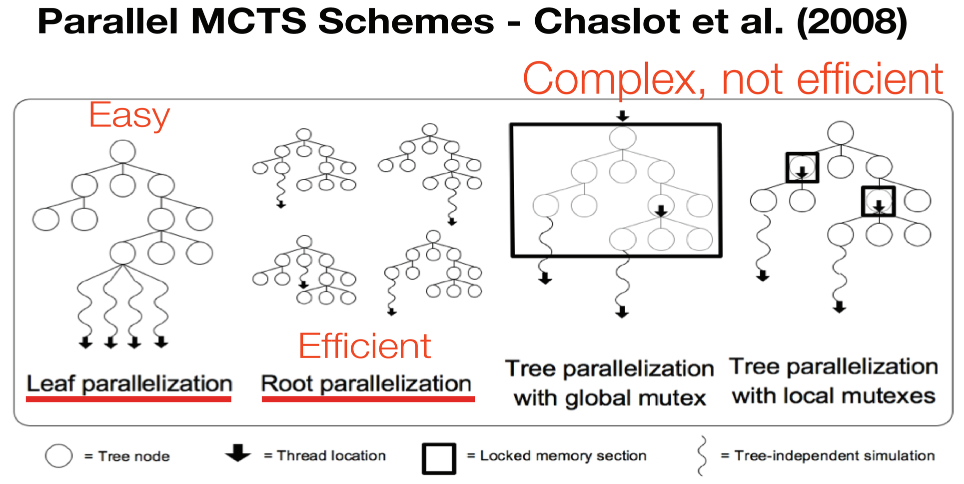
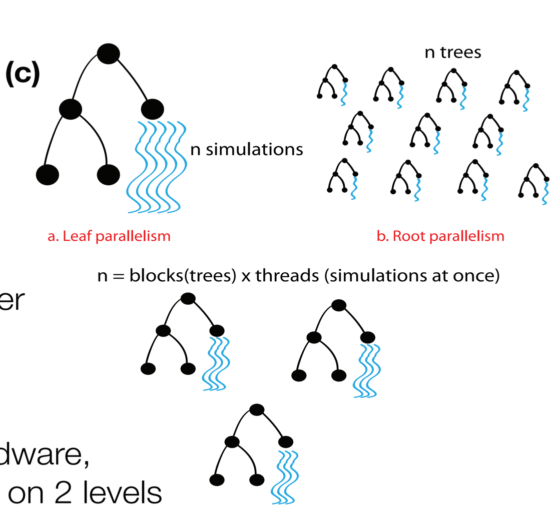


Fig. 2. (a) Leaf parallelization (b) Root parallelization (c) Tree parallelization with global mutex (d) and with local mutexes

Our approach - Parallel MCTS on GPU

= Block parallelism **(c)**



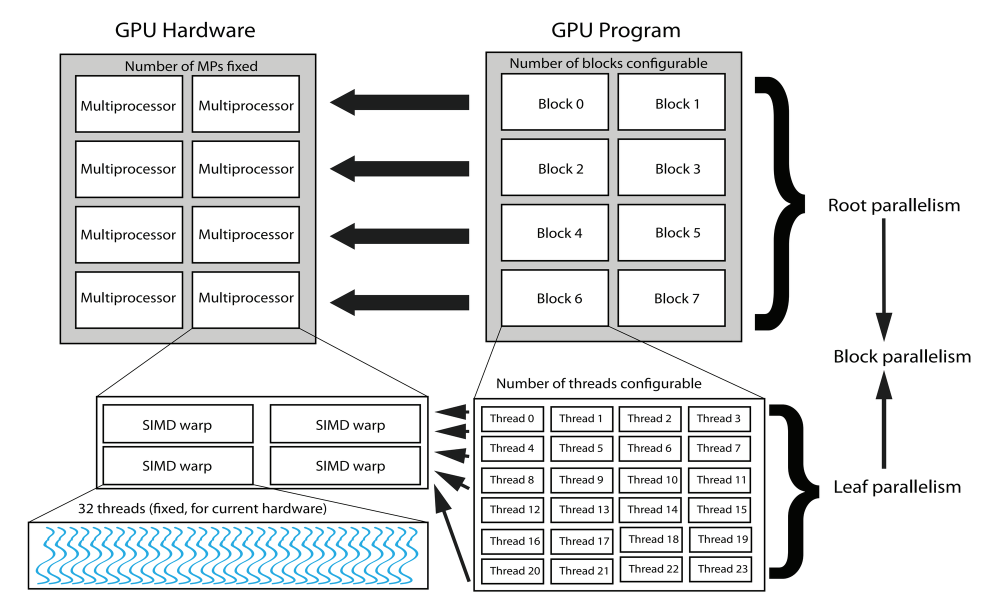
Weakness:
CPU sequential tree management part (proportional to the number

Advantage:
Works well with SIMD hardware, improves the overall result on 2 levels of parallelization

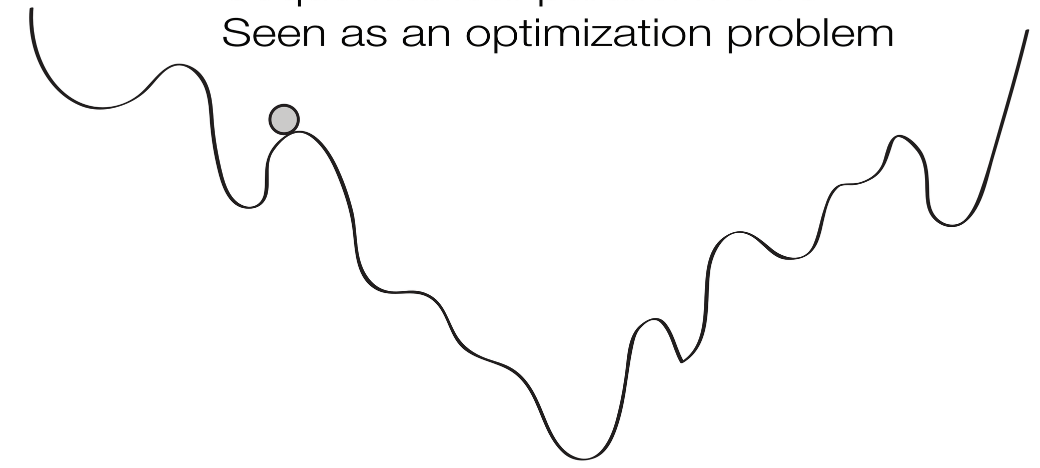
Problem statement

Parallel tree search is one of the basic problems in computer science. It is used to solve many kinds of problems. Effective parallelization is hard, especially for more than hundreds of threads. SIMD hardware (i.e. GPU) is fast, but hard to utilize. How to utilize GPUs/CUDA?

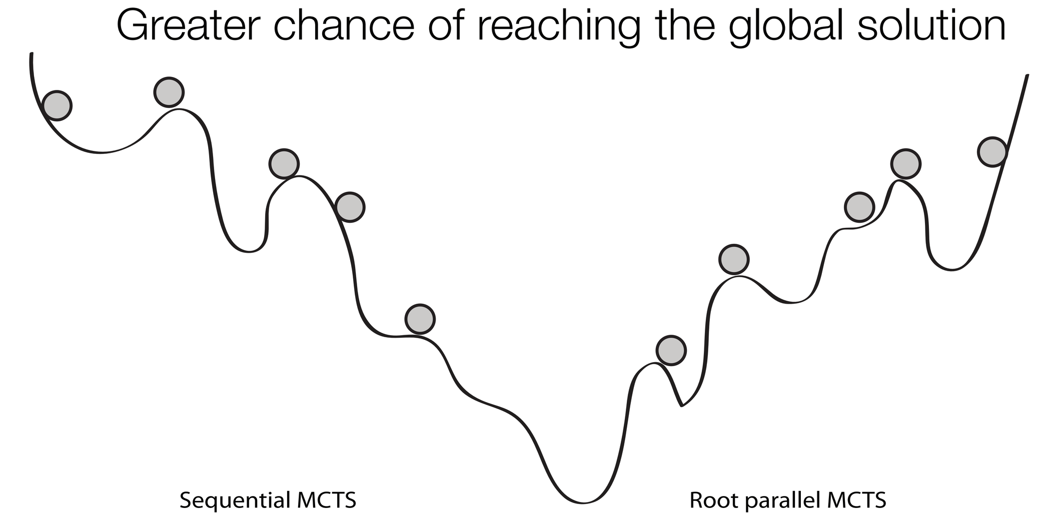
Mapping MCTS trees to blocks

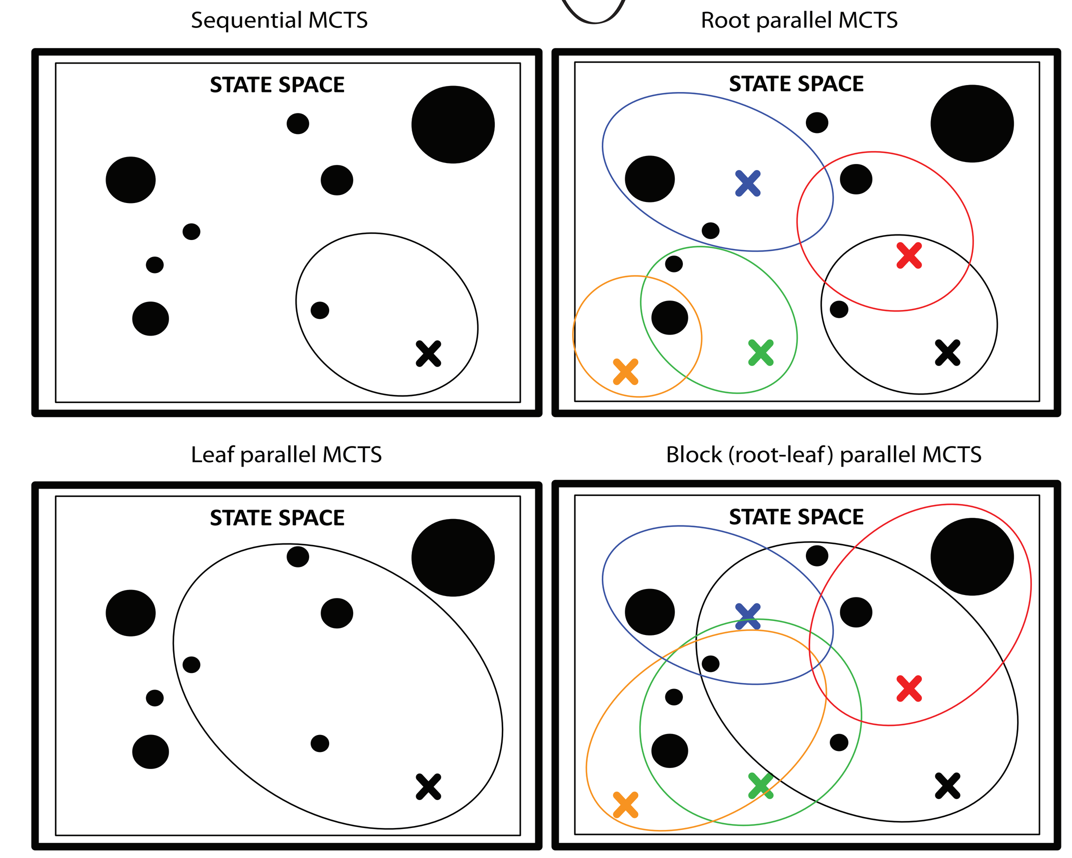


Sequential/leaf parallel MCTS
Seen as an optimization problem



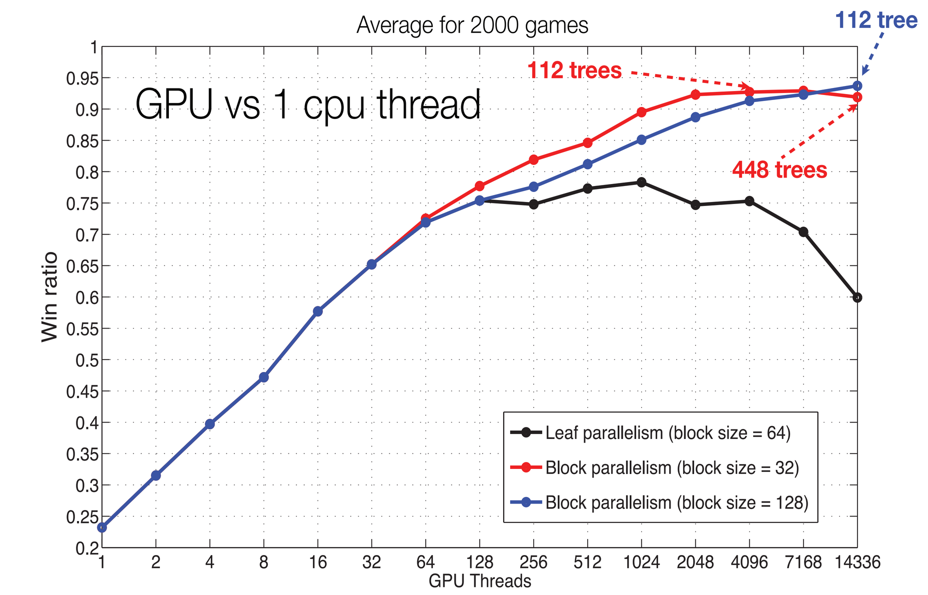
Root parallel MCTS - many starting points
Greater chance of reaching the global solution

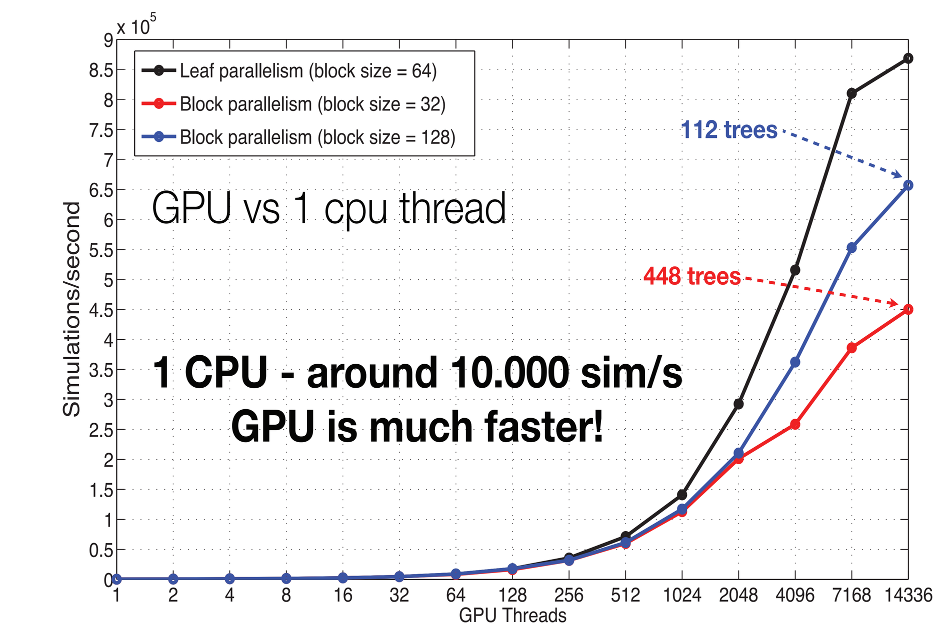




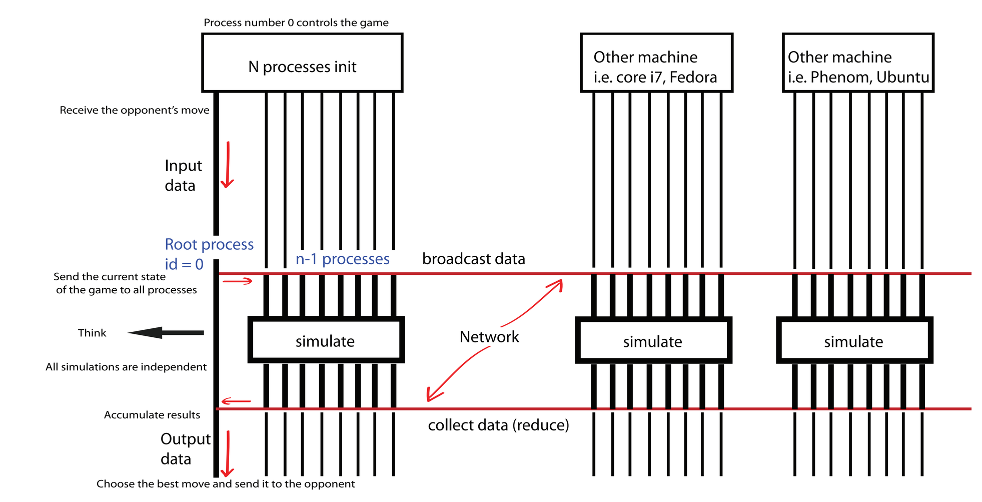
✖ Starting point - with root parallelism, more chance of finding the global solution
● Local solution (extremum)
○ Search scope - with leaf parallelism, the search is broader/more accurate (more samples)

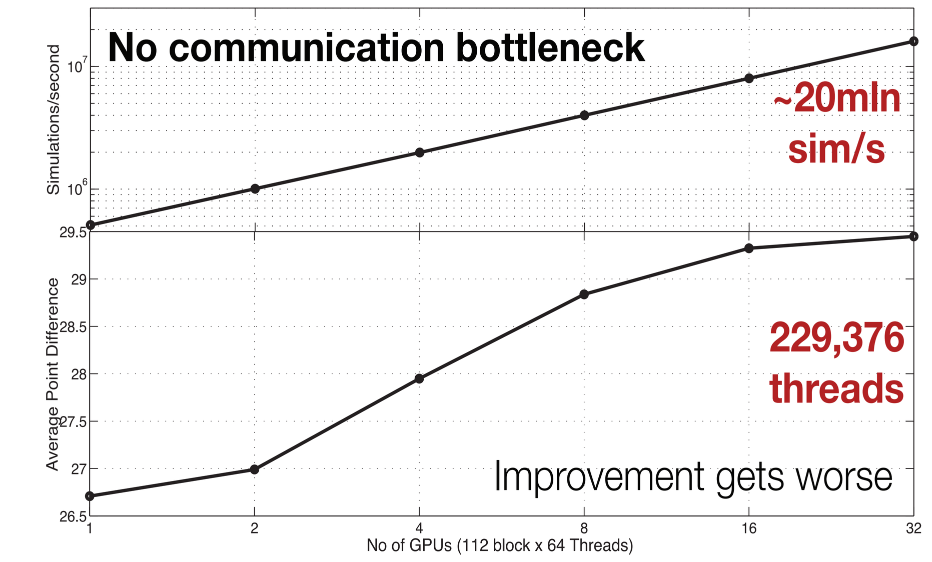
Results and findings





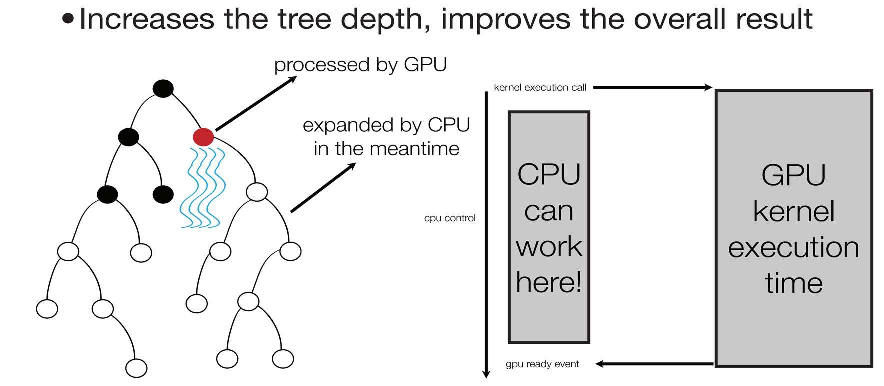
Scalability - MPI Parallel Scheme

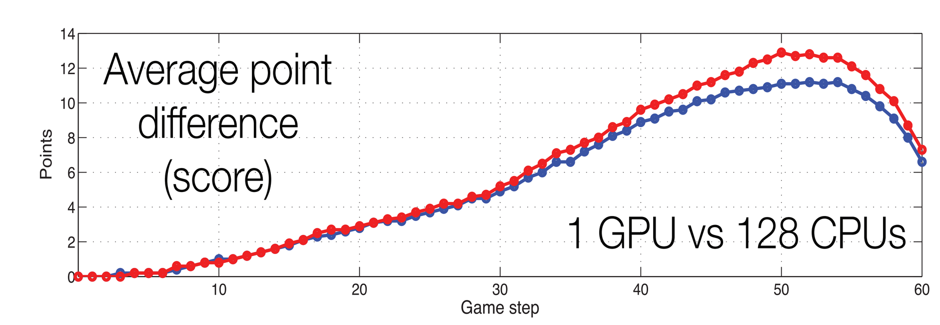


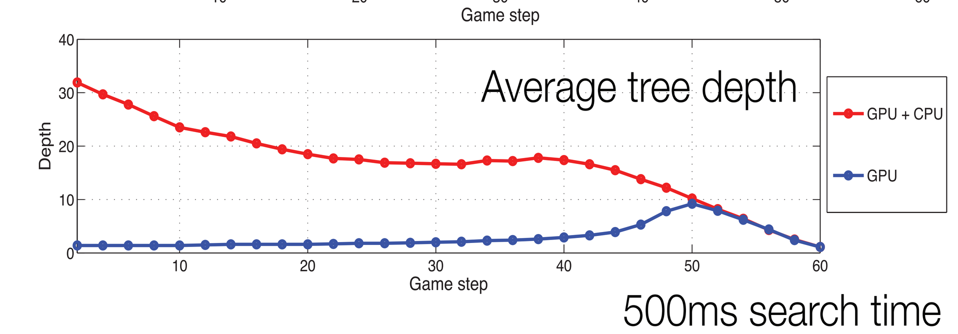


Hybrid CPU/GPU search

- While GPU runs a kernel CPU can work too
- Increases the tree depth, improves the overall result





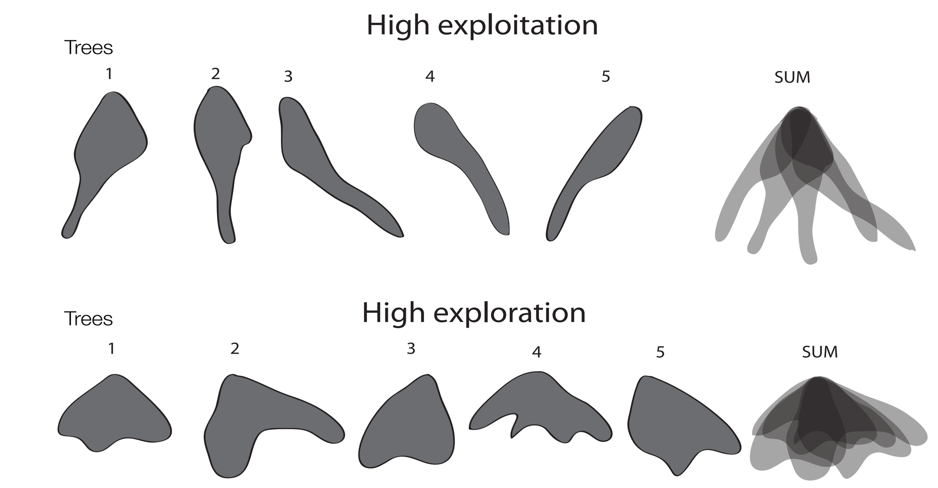


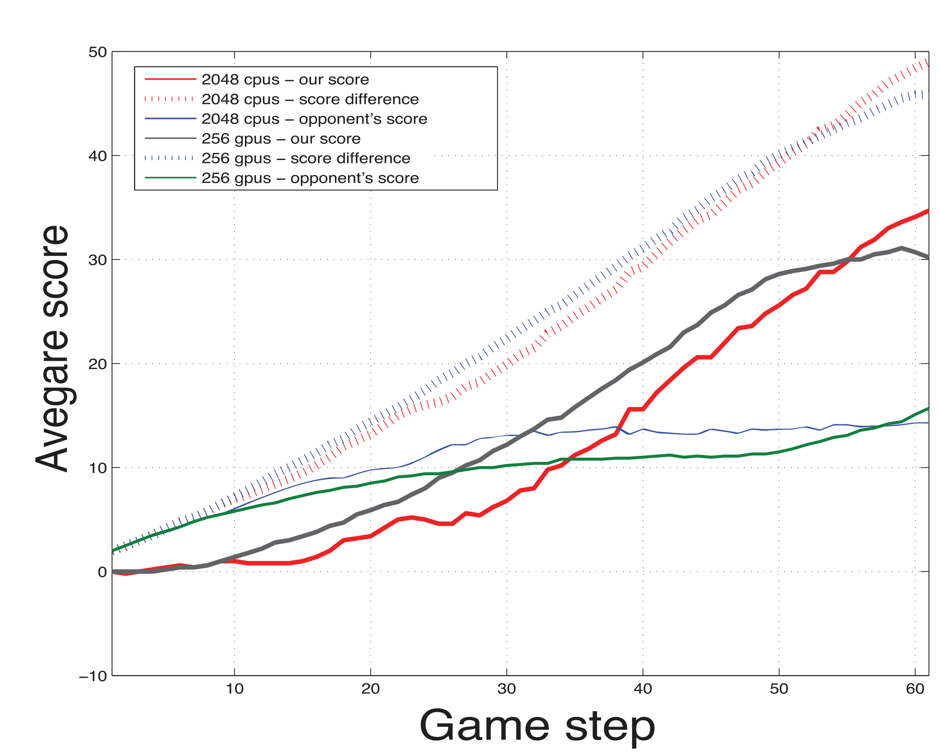
500ms search time

•**Findings:**

- **Weak scaling of the algorithm** - problem's complexity affects the scalability
- **Exploitation/exploration ratio** - higher exploitation needed for more trees
- **No communication bottleneck**
- **Much more efficient than the CPU version**

Exploration/exploitation in parallel MCTS





256 GPUs (3,670,016 threads) and 2048 CPU threads vs sequential MCTS

- **More trees = higher score**
- **More simulations = higher score**
- **More trees = fewer simulations**
- **Block size needs to be adjusted**
- **1 GPU ~ 64-128 CPUs (AI power)**

This work was partially supported by Core Research of Evolutional Science and Technology (CREST) project "ULP-HPC: Ultra Low-Power, High-Performance Computing via Modeling and Optimization of Next Generation HPC Technologies" of Japan Science and Technology Agency (JST) and Grant-in-Aid for Scienti c Research of MEXT Japan.