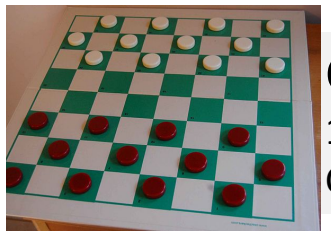
HPC Challenges in Artificial Intelligence Research
Massive Parallel Search
and
Parallel Deep Learning

Kazuki Yoshizoe (美添 一樹)

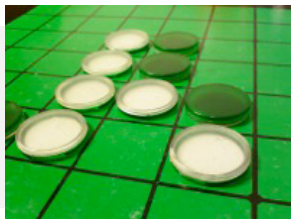
RIKEN Center for Advanced Intelligence Project
RIKEN Advanced Institute for Computational Science
IHPCSS2017

Man vs Machine history



Checkers
1994 CHINOOK vs Tinsley
CHINOOK (com) won

Othello (Reversi)
1997 Logistello vs Murakami
Logistello (com) won 6-0



Chess
1997 Deep Blue vs Kasparov
Deep Blue (com, IBM) won

Shogi (Japanese chess)
Computer is stronger than human champion

The game of Go
before 2005, 3kyu (weak amateur)
2006 breakthrough 1, MCTS
2011 breakthrough 2, DCNN
2016 Beat former champion
2017 beat top players 60-0

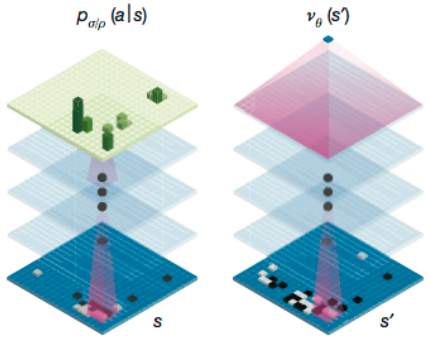
AlphaGo

A “Go” program by
Google DeepMind

Search
(Monte Carlo Tree Search)

Machine Learning
(Deep Learning)

[Coulom 2006]



[Silver, Huang et al. 2016] Fig. 1b

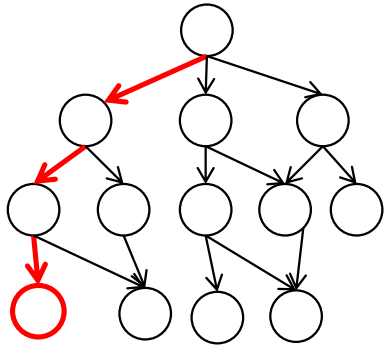
Scalable Parallel Search

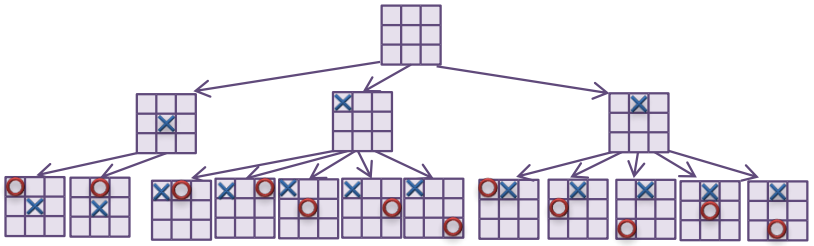
What is Search?

Search finds

(Set of) node(s) or path(s)

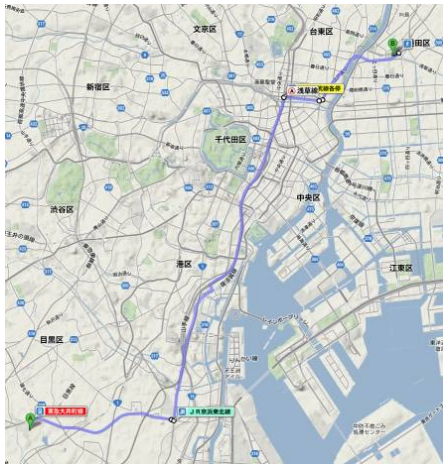
from a given Graph

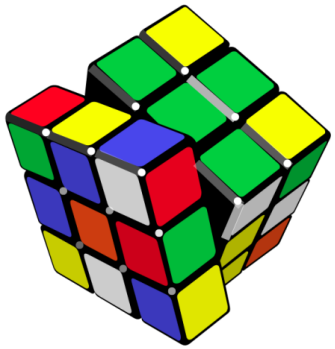


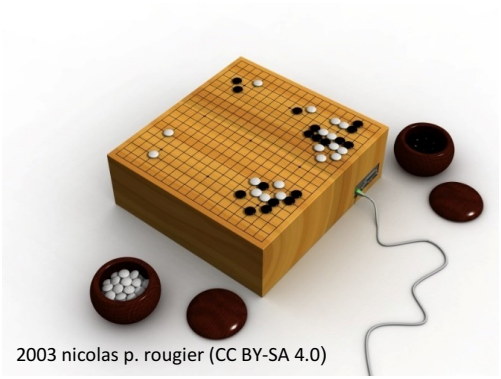


Node or path shows

- ✓ “shortest path”
- ✓ “optimal combination”
- ✓ “best play in games”







2003 nicolas p. rougier (CC BY-SA 4.0)

Explicitly given Graph

Trains

Road map

social network

Generated Graph

Combinatorial optimization

Games

SAT, CSP

Difficulty of Parallel Search

What are the difficulties?

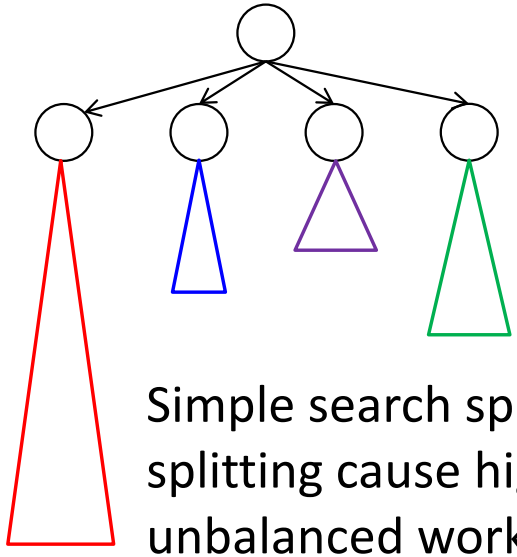
Computational time prediction

Unpredictable memory access pattern

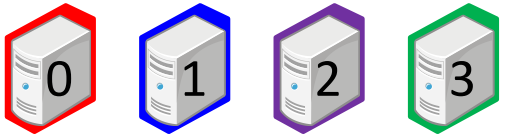
Difficult to divide to uniform workload tasks

Difficult to equally partition data structure

Naïve approach fails



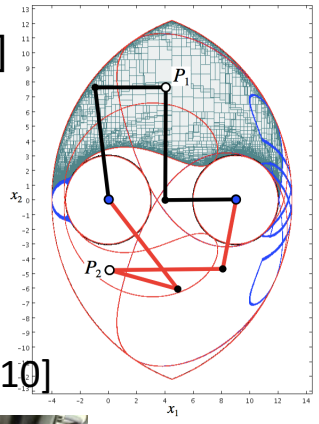
Simple search space splitting cause highly unbalanced workloads



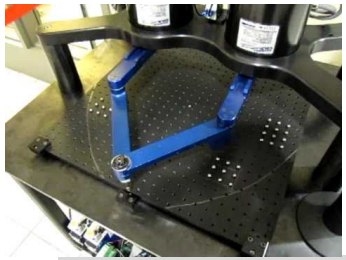
Parallel DFS (Depth First Search)

Two applications from our work

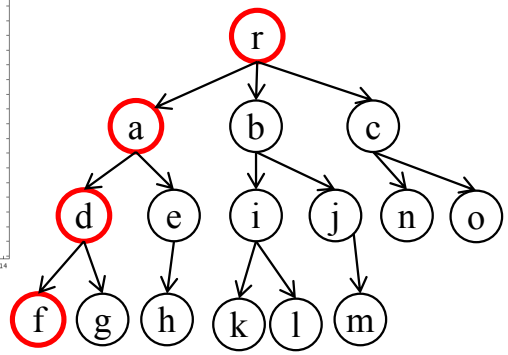
[Caro+ 12]



DexTAR
[L. Campos+ 2010]

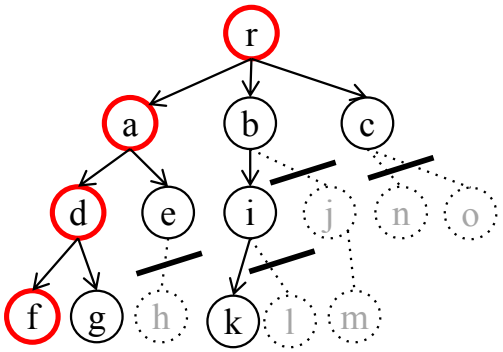


DFS w/o threshold



"enumeration"

DFS with **threshold**



"optimization"

```
…GTCTAAAACATGATT…  0
…GTCTGAATCATGATT…  1
…GTCTGAAACATGATT…  0
…GTCTGAATCATCATT…  1
```

Numerical CSP

A region is given by inequality constraints.
Find small "boxes" which cover the region.
750-fold speedup using **900cores**
An application of search and interval calculus.

[Ishii, Yoshizoe, Suzumura 2014] CP2014

Statistical Pattern Mining

Finding statistically significant patterns.
Applied to GWAS
(Genome Wide Association Studies)
1,175-fold speedup using **1,200cores**

[Yoshizoe, Terada, Tsuda 2015] arXiv

Frequent Itemset Mining

database	items 1	2	3	4	5	6
A	x	x	x	x	x	x
B		x	x		x	
C		x			x	
D	x	x		x	x	x
E		x		x		
F	x			x		x
G			x	x		x

transactions

Counting / Enumerating
Frequent **itemsets**
from a given database

ex. itemset with freq 3 or higher

{1}, {2}, {3}, {4}, {5}, {6},{1,4},
{1,6}, {2,4}, {2,5}, {4,6}, **{1,4,6}**

ex1. Market Basket Analysis
items: products
trans.: customers
x: purchased items

ex2. Genomics data
items: SNP
trans.: human
x: SNP

…GTCT**A**AAACATGATT…
…GTCTGAA**T**CATGATT…
…GTCTGAAACATGATT…
…GTCTGAA**T**CAT**C**ATT…

SNP: Single Nucleotide Polymorphism

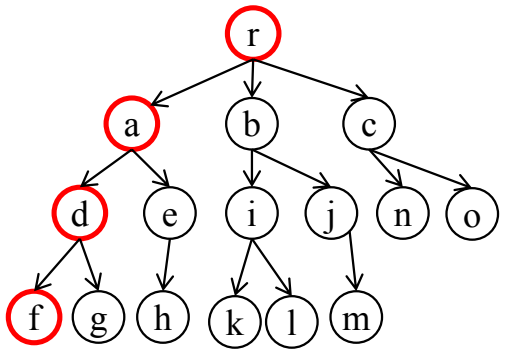
Depth First Search (w/o threshold)

Back tracking DFS

```
DFS() {
  Recur(r)
}
Recur(node n) {
   foreach (child c of n) {
      // do something for c
      Recur(c)
   }
}
```

back tracking can be naturally implemented with *recursive* call

Simply traverses all nodes in the tree



Memory usage $O(d)$
Only current path is needed

NCSP and Frequent Itemset Mining are DFS w/o threshold

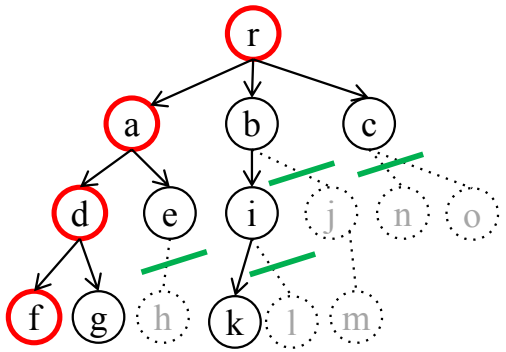
Depth First Search with threshold update

DFS with threshold

```
DFS() {
  Recur(r)
}
Recur(node n) {
   foreach (child c of n) {
      // do something for c
       if (c is within threshold) Recur(c)
      UpdateThreshold()
   }
}
```

Prune search space by
dynamically updating threshold

Update threshold during search.
More branches are pruned in the right.
(Search progresses from left to right.)



Ex. finding top-k nodes

Statistical Patten Mining is DFS with threshold
significance threshold is dynamically updated

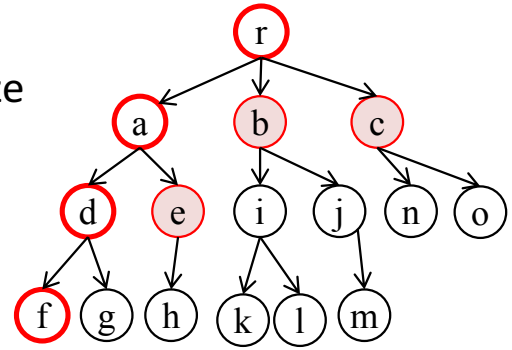
Parallel DFS, preparation

```
DFS() {
  Recur(r)
}
Recur(node n) {
   foreach (child c of n) {
      // do something for c
       if (c is within threshold) Recur(c)
      UpdateThreshold()
   }
}
```

convert recursion to stack + loop

```
StackDFS() {
   push(r)
   Loop()
}
Loop() {
   while(stack not empty) {
      pop n from stack
      foreach (child c of n) {
         // do something for c
          if (c is within threshold) push(c)
         UpdateThreshold()
      }
   }
}
```

pros: O(d) memory
cons：difficult to parallelize



cons: O(db) memory
pros: easy to parallelize

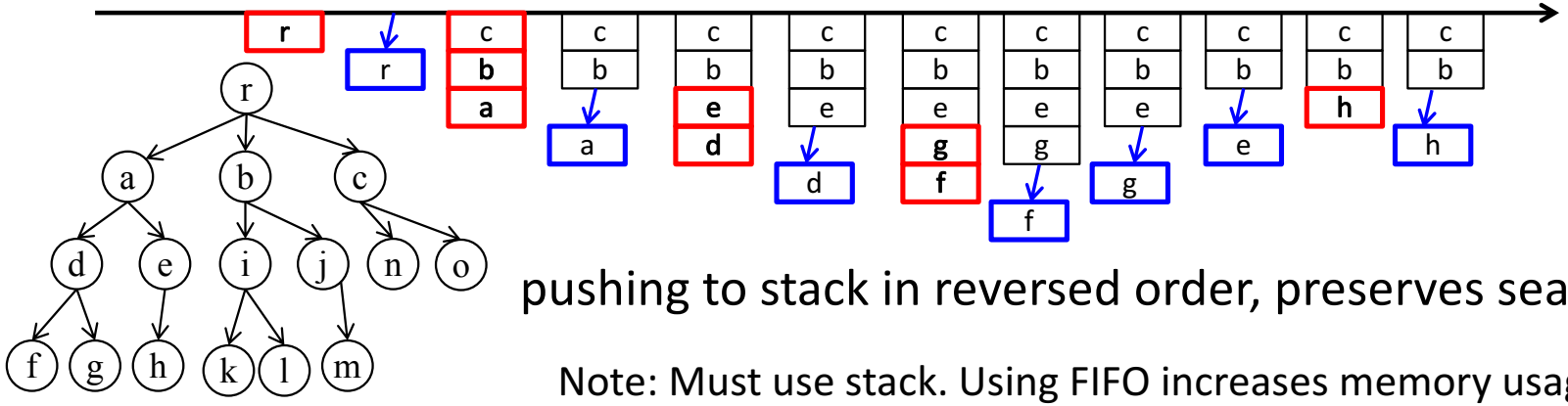
For depth d, branch nu. b search space

```
DFS() {
  Recur(r)
}
Recur(node n) {
   foreach (child c of n) {
      // do something for c
      if (c is within threshold) Recur(c)
      UpdateThreshold()
   }
}
```

```
StackDFS() {
   push(r)
   Loop()
}
Loop() {
   while(stack not empty) {
      pop n from stack
      foreach (child c of n) {
         // do something for c
         if (c is within threshold) push(c)
         UpdateThreshold()
      }
   }
}
```

foreach in reverse order

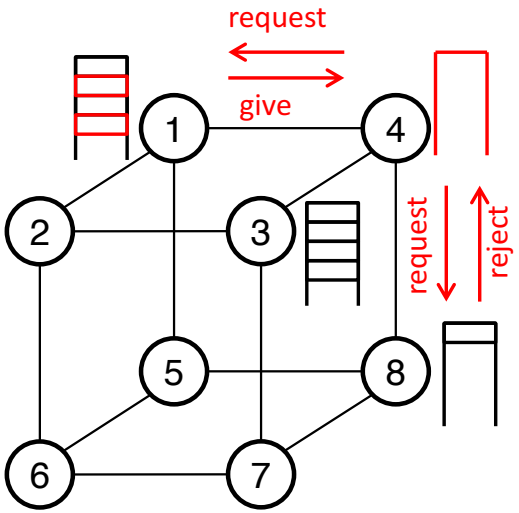
Convert recursive call to stack + loop



pushing to stack in reversed order, preserves search order

Note: Must use stack. Using FIFO increases memory usage.

Work Stealing based parallelization



Steal work from “victim”

Receiver initiated Work stealing

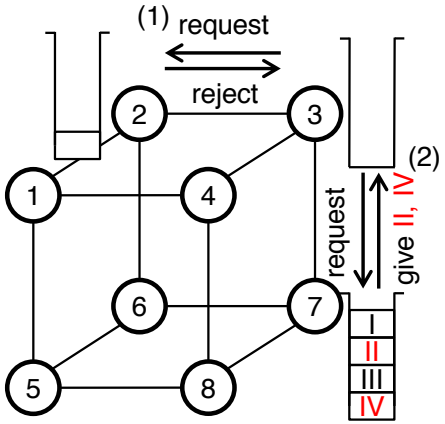
Workers with empty stack (empty job)
1, Select a victim worker
2, Send job request to the victim
3, The victim gives jobs if available. Rejects otherwise
(details are omitted)

Simple method for victim selection
“Select randomly”

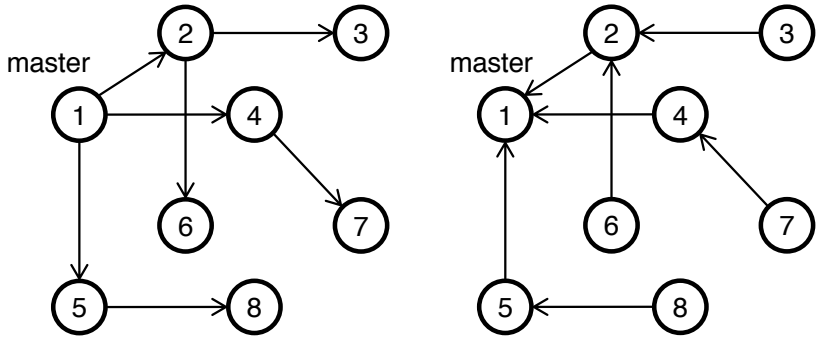
A better method
Select victims from neighbors on hypercube
(virtual hypercube is prepared ignoring actual topology)

Lifeline graph [Saraswat et al. 2011]

Threshold Broadcast / Reduce



Work stealing
on hypercube



Distributed Termination Detection

Threshold broadcast/reduce and DTD
on spanning tree

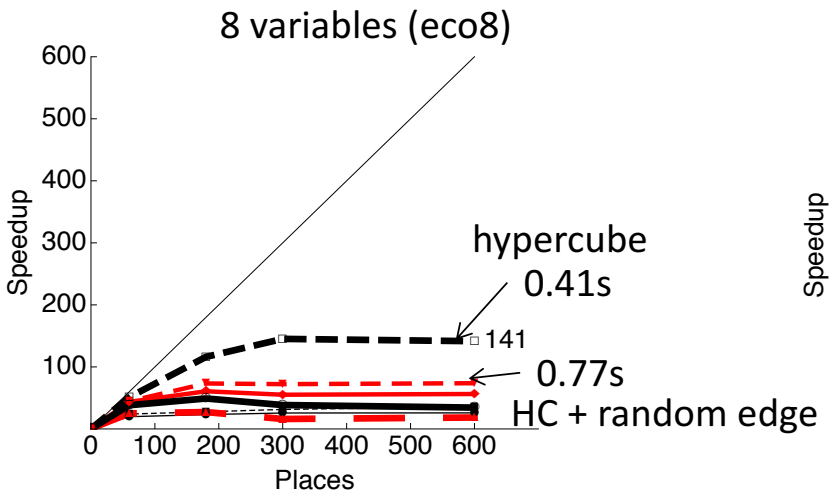
Applied DTD on spanning tree [Mattern 1990]

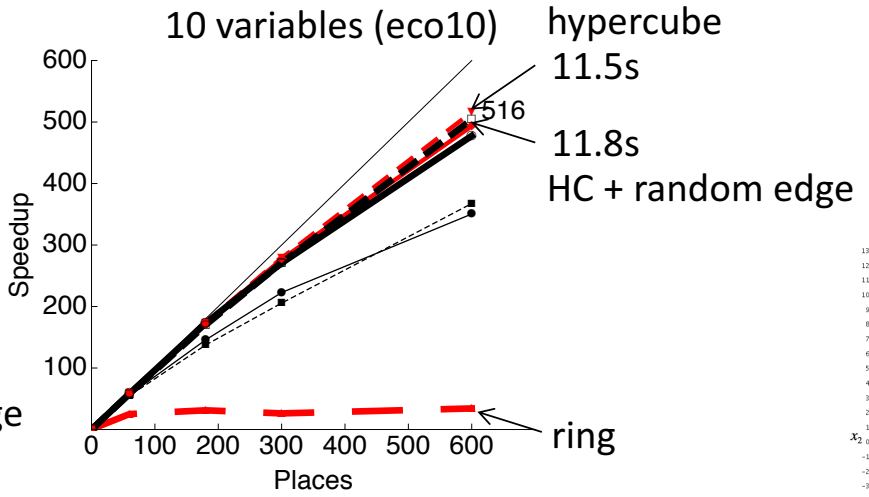
Proof is needed to confirm all stacks are empty
in distributed environment (details omitted).

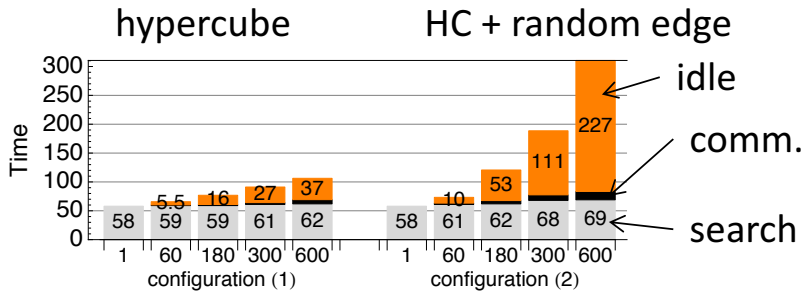
[Ishii, Yoshizoe, Suzumura 2014]

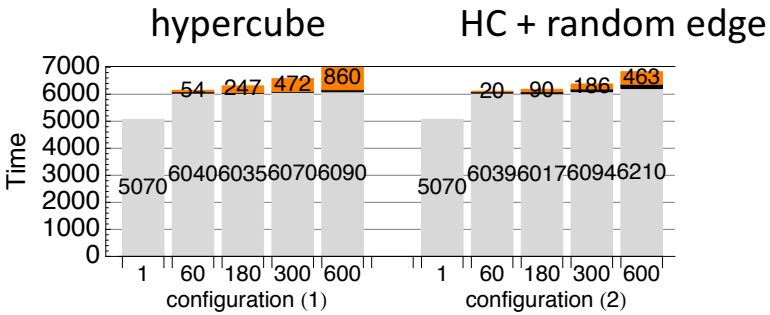
Numerical CSP Speedup and Analysis

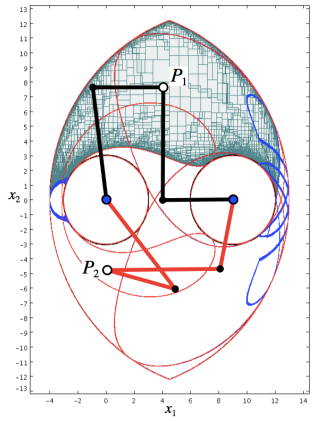
for different
victim selection methods

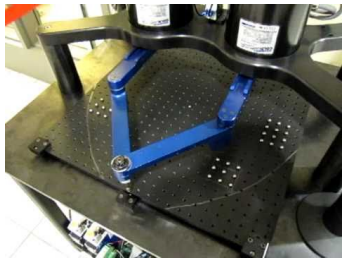




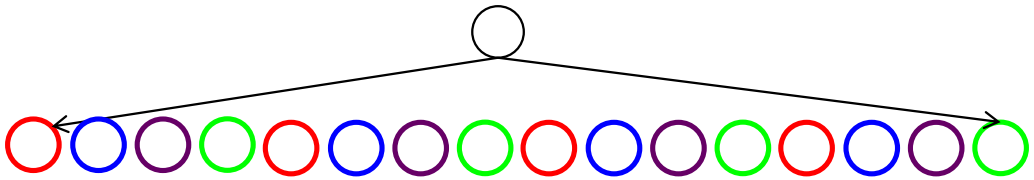




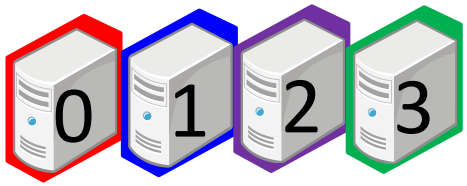


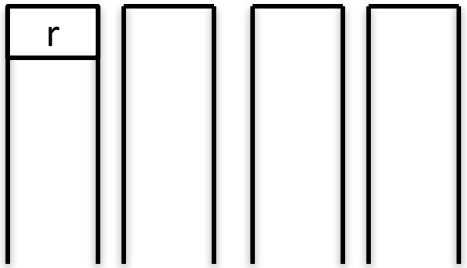


Combine with Static load balancing

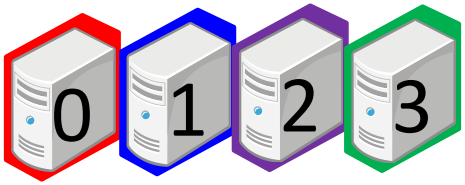


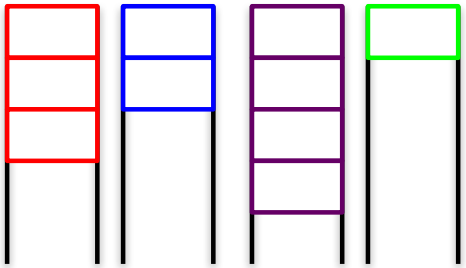
distribute depth 1 nodes at the initialization





Original algorithm starts with only the root node pushed





If initialized like this, load balancing can be easier

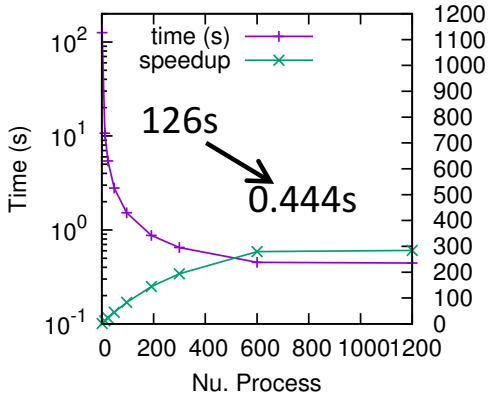
Applied only to Statistical Pattern Mining

Preprocess phase: distribute depth-1 nodes to workers

Statistical Pattern Mining: Speedup

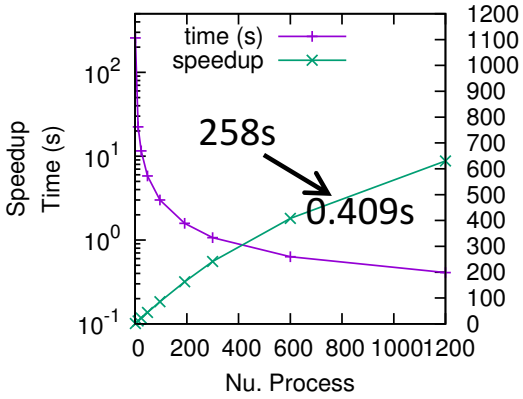
JPN dom. 10%

item: 11253, trans:697, dens:1.0%



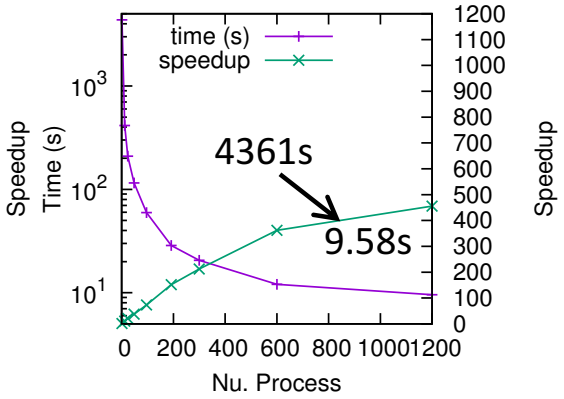
Alz. dom. 5%

item:44052, trans:364, dens:5.4%



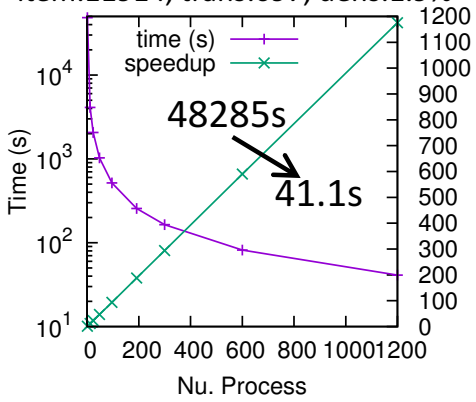
Alz. rec. 30%

item: 250120, trans:364, dens:2.9%



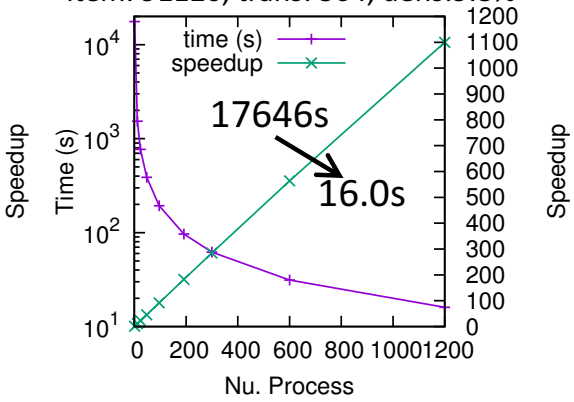
JPN dom. 20%

item:11914, trans:697, dens:1.9%



Alz. dom. 10%

item: 91126, trans: 364, dens:9.8%



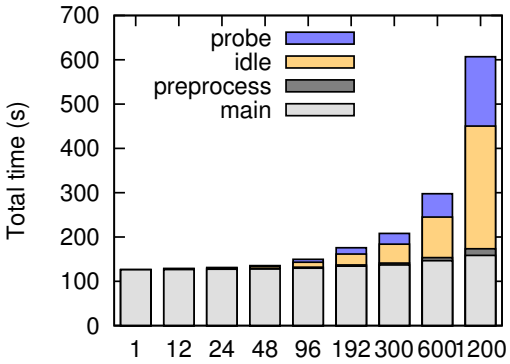
Speedup for Finding combination of SNPs related to Alz. or Japanese

TSUBAME@TITECH (CPU only)
Used Max. **1,200 cores** (100nodes)
1,175-fold speedup in the best case (self comparison)

Breakdown of computational time (search, comm., idle)

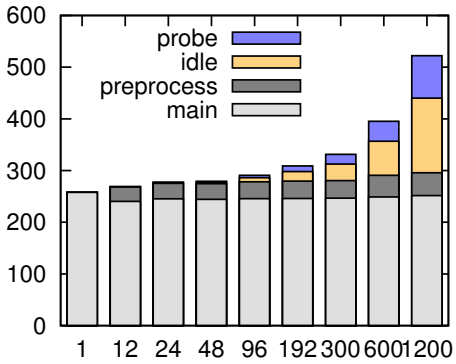
JPN dom. 10%

item: 11253, trans:697, dens:1.0%



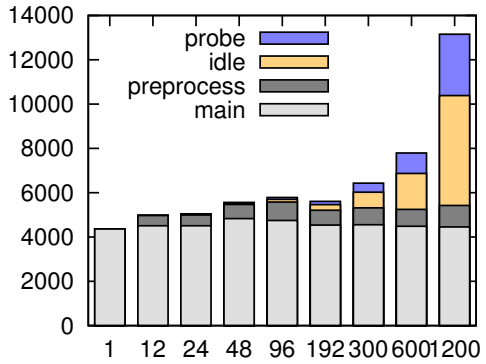
Alz. dom. 5%

item:44052, trans:364, dens:5.4%



Alz. rec. 30%

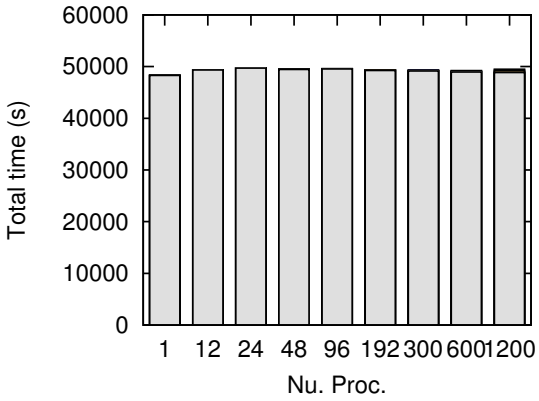
item: 250120, trans:364, dens:2.9%



Showing total time for all workers

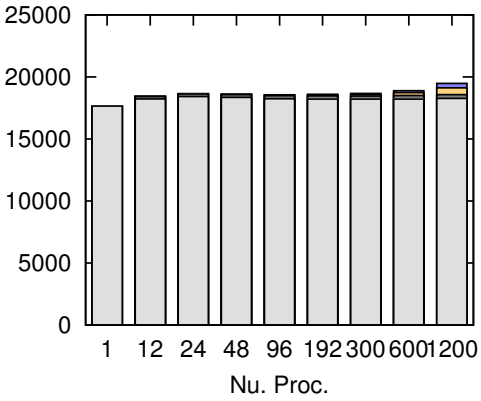
JPN dom. 20%

item:11914, trans:697, dens:1.9%



Alz. dom. 10%

item: 91126, trans: 364, dens:9.8%



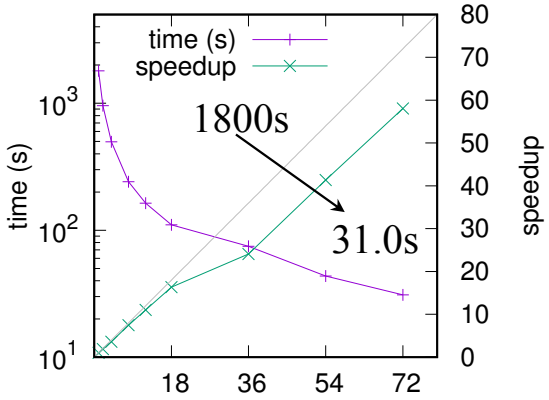
Implemented using
c++ and MPI
(runs also on ethernet)

amazon web services

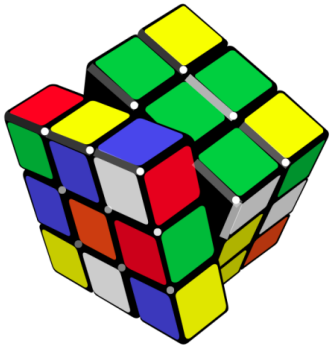
EC2 c4.8xlarge
Xeon E5-2666v3
(2.9GHz, 18cores)
10G ethernet,
using ***cfncluster***

JPN dom. cSNP 15%

item:11914, trans:697, dens:1.9%



Examples of Non-DFS Algorithms



"God's number is 20"

A* search

(pronounced "A star")

Dijkstra's algorithm
+ **heuristic**
almost 50 years old

MCTS

Monte Carlo Tree Search

Random sampling
based search
invented on 2006

material science

scheduling

[Cazenave, Balbo, Pinson 2009]
"Monte-Carlo bus regulation"

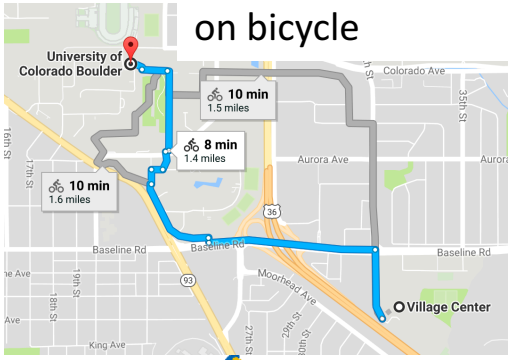
[Tanabe, Yoshizoe, and Imai 2009]
"A study on security evaluation methodology for image-based biometrics authentication systems"

NLP

biometric security

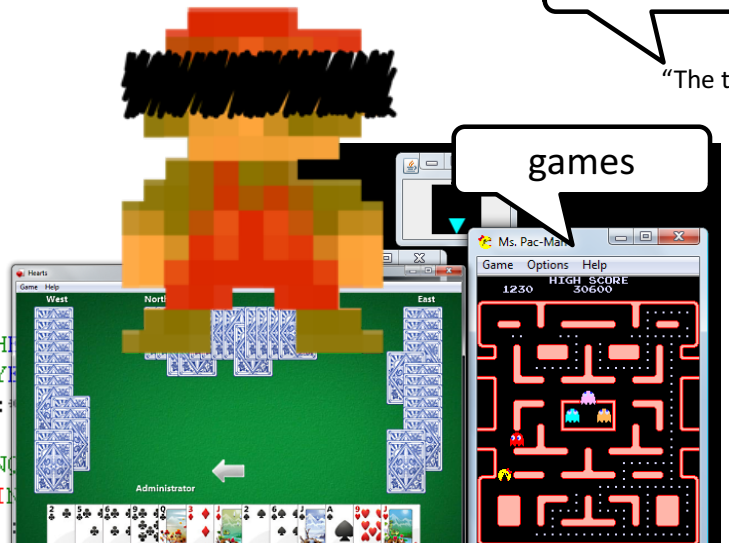
[Chevelu, Putois, Lepage 2010]
"The true score of statistical paraphrase generation"

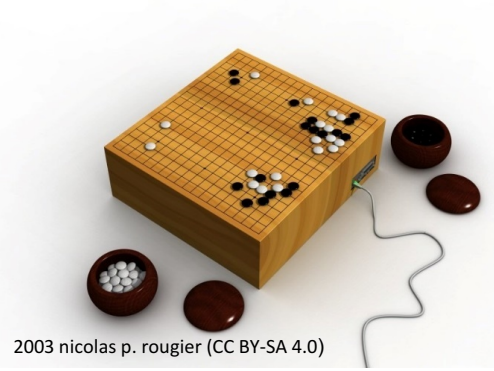
Shortest path from here to UCB on bicycle



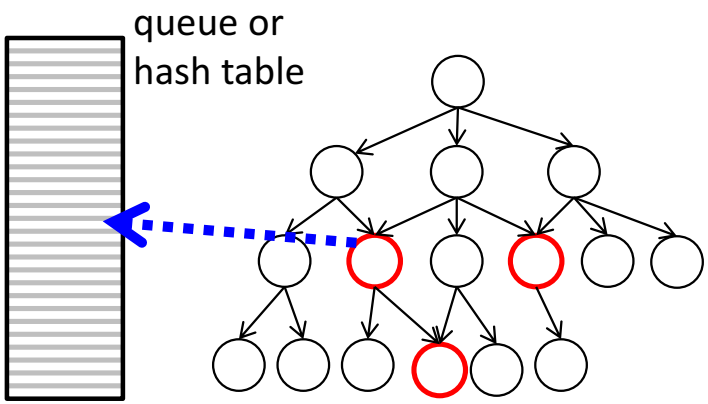
DNA sequence alignment

```
AAB24882        TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTH
AAB24881        ----------------------YECNQCGKAFAQHSSLKCHYRTHIGEKPY
                                       ****: .***:  * *:** * :****.:
AAB24882        PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYE-CN
AAB24881        HSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEKPYMNVI
                 **** *:***********:***:**.: .****************
```

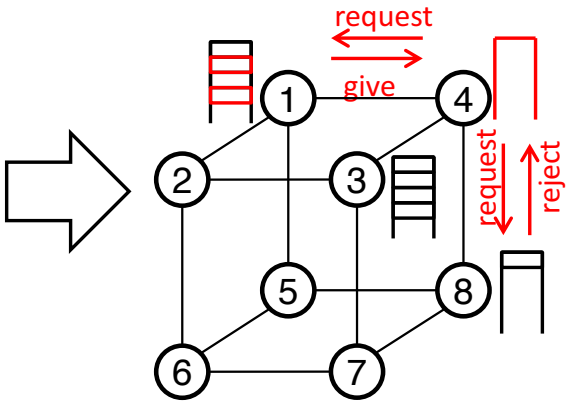




What's Needed for Non-Depth First Search?




```
DFS_Recur(node n) {
   foreach (child c of n) {
      // do something for c
       DFS_Recur(c)
   }
}
```



Nodes can be visited multiple times
Result are recorded and reused later

using either
Priority Queue (A* search) or
Hash Table (MCTS, IDA*)

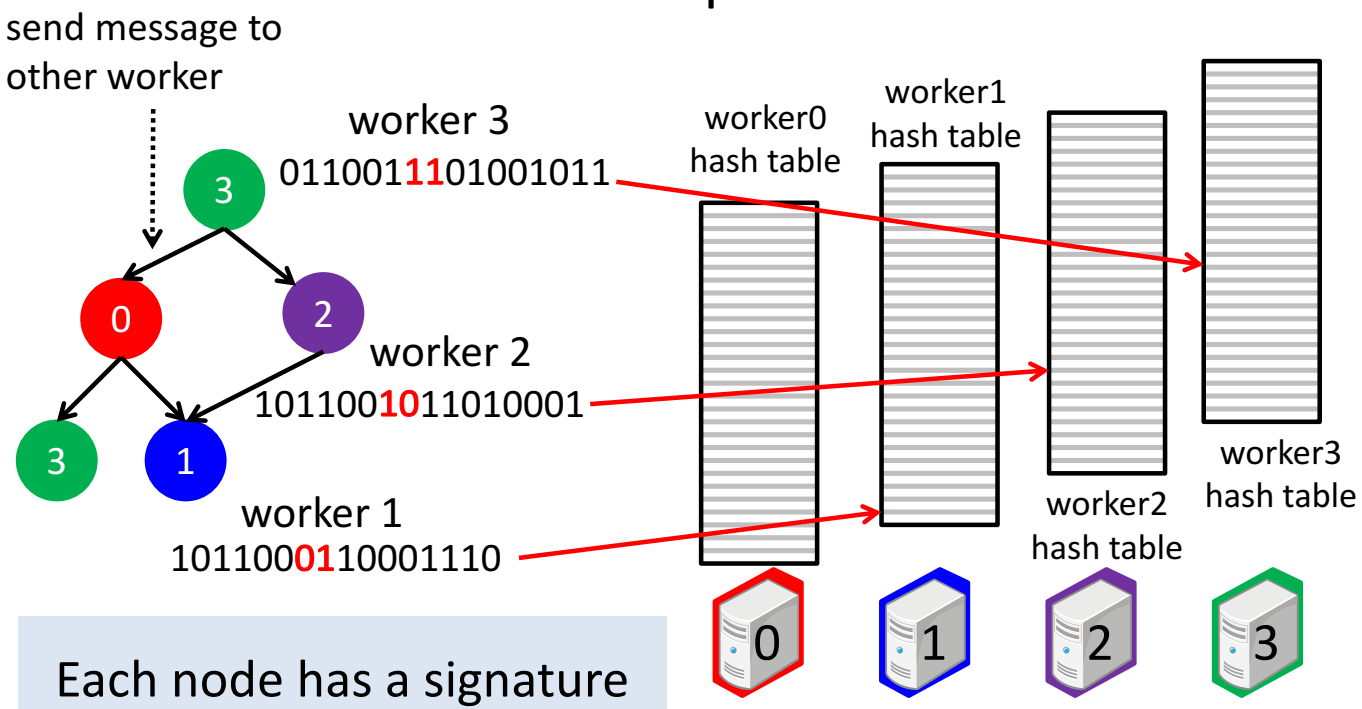
```
NonDFS(node n) {
   while(not_finished) {
      ReadFromTable(n)
      foreach (child c of n) {
         // do something for c
      }
      WriteToTable(n)
   }
}
```

?

Distributed Hash Table
Distributed Queue

Distributed Hash Table Driven Parallelization

Transposition table Driven Scheduling [Romein et al. 1999]



Uniform load balancing

tradeoff

Frequent 1-to-1 comm.

Each node has a signature

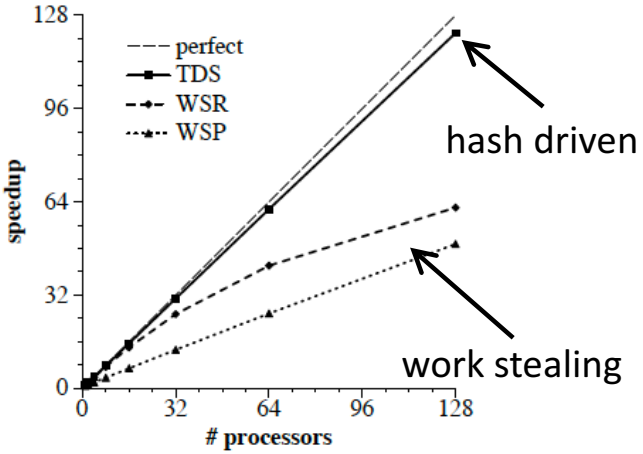
Part of the signature shows the “home worker”

workers sends messages to home worker of children

signature is calculated by a hash function

Hash driven Parallel Search Performance

TDS algorithm
(Parallel IDA*)



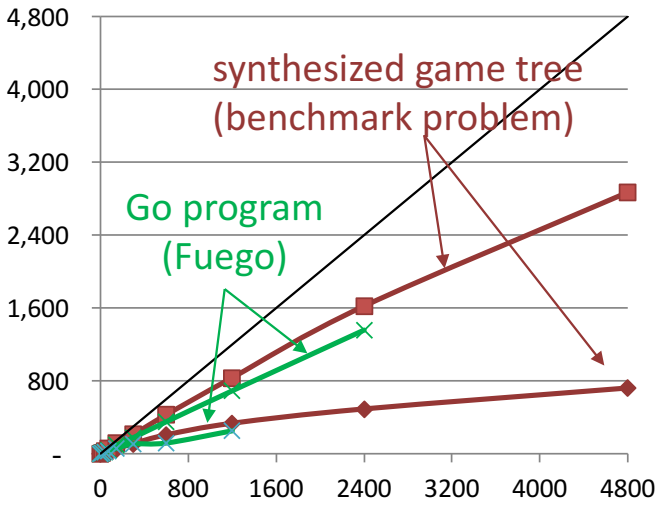
[Romein+ 1999] Fig. 4 (c)

HDA*
(Parallel A*)

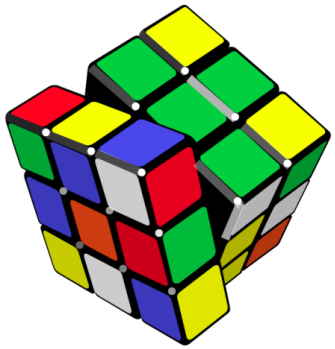
Applied to puzzles, planning, and sequence alignment

[Kishimoto+ 2012]

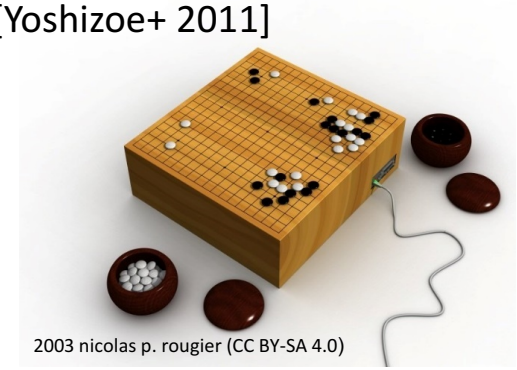
TDS-df-UCT algorithm
(Parallel MCTS)



[Yoshizoe+ 2011]



Note: These performances are achieved if communication congestion is removed by reformulation of algorithm



Parallel Search Summary

Depth First Search

is simple but has applications.
Can often be efficiently parallelized with stack + loop and work stealing.

Old Algorithms are useful

Parallel algorithm research in 1980s are sometimes useful.
An example is Distributed Termination Detection [Mattern 1990]

A* and MCTS

are more complex but more applications exists.
Hash driven distributed search works. Needs improvements for 1,000 or more workers.
(probably a better hash function)

Implementation

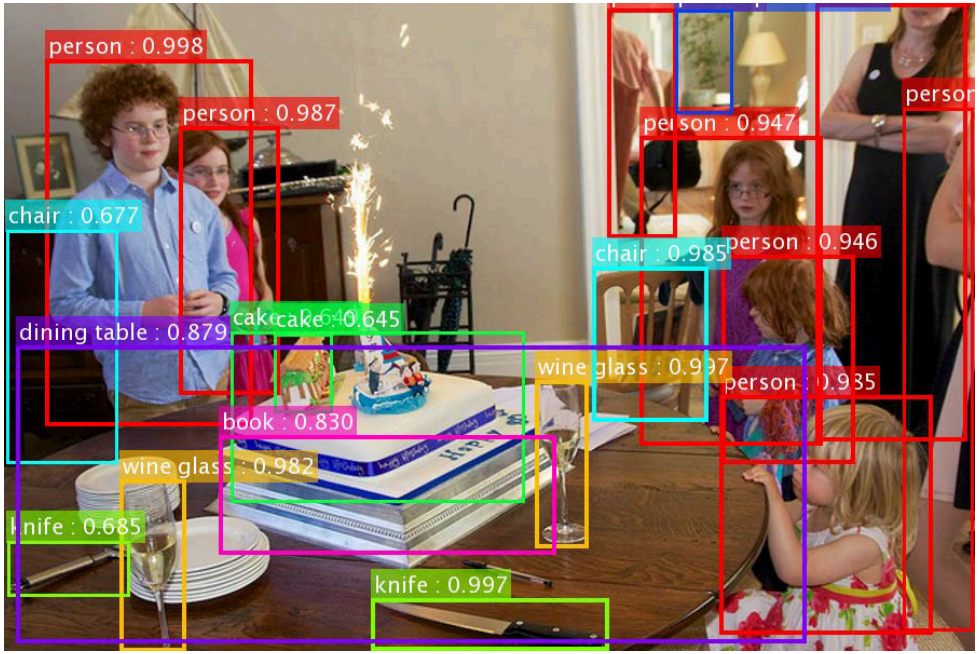
Our implementation is mainly based on C++ and MPI. We use MPI_Bsend and MPI_Iprobe / MPI_Recv.
It is because message target, size, and timing are all unknown.
(If you know a better way, please let us know)

Parallel Training of Deep Neural Network

Or, can we solve

“Large Mini-batch Problem?”

What Deep Learning (DCNN) can do?



[K. He et al. 2015, Microsoft Research Asia]

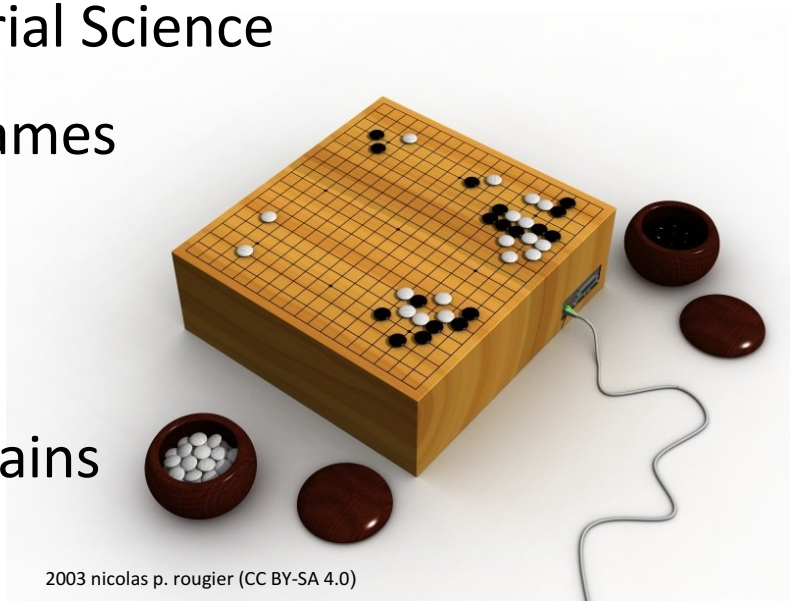
Image recognition

Natural Language Processing

Sound / Voice recognition

Material Science

Games



2003 nicolas p. rougier (CC BY-SA 4.0)

Applied to almost any domains
in computer science

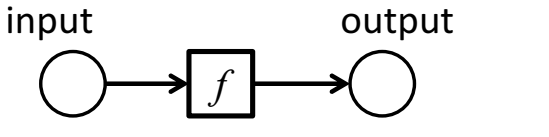
Shallow

What is **Neural Network?**

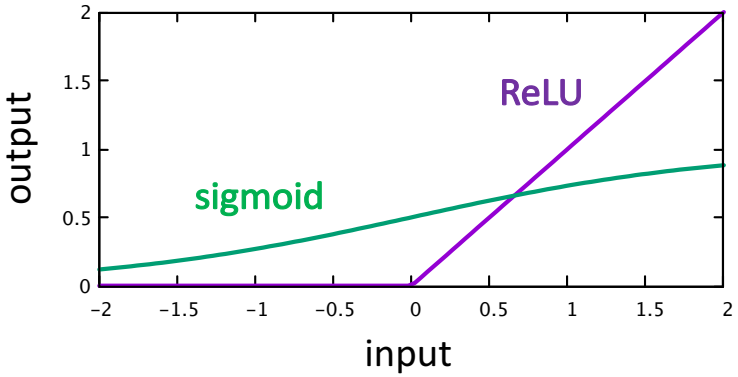
A algorithm inspired by mechanism of neurons

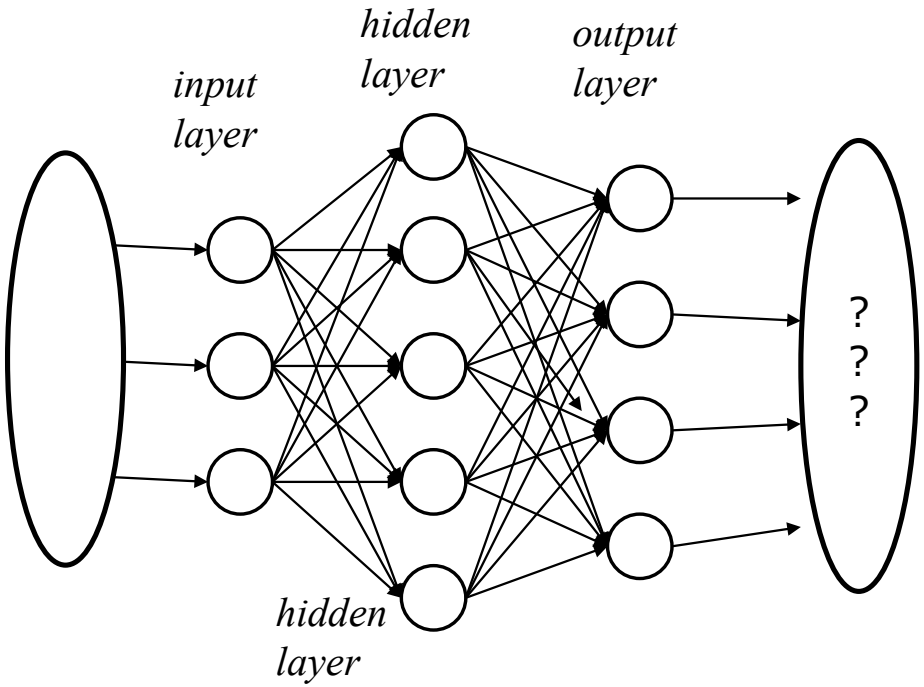
A neuron outputs

- small value for small input
- large value for large input



activation function





Convolutional Filters

Multiply and add surrounding pixel values

Original image

Examples of filters

blur

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

edge detect

0	1	0
1	-4	1
0	1	0

emboss

-2	-1	0
-1	1	1
0	1	2

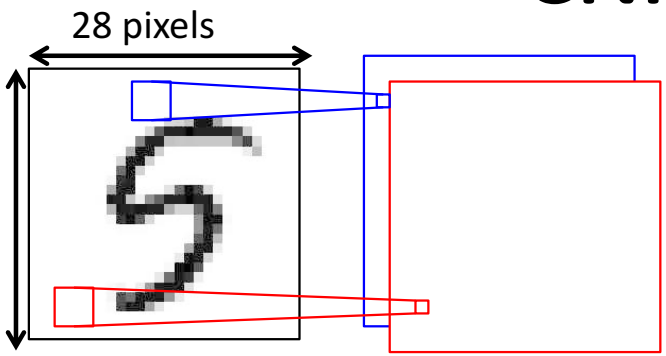
Many types of operations are possible by adjusting filters' weights and size

Neural Networks can calculate Convolutional Filters

Examples are from the manual of GIMP

8.2. Convolution Matrix http://docs.gimp.org/en/plug-in-convmatrix.html

CNN: Convolutional Neural Network



Famous benchmark

MNIST handwritten digit database
http://yann.lecun.com/exdb/mnist/

vertical line filter (3x3)

-1	2	-1
-1	2	-1
-1	2	-1

horiz. line filter (5x5)

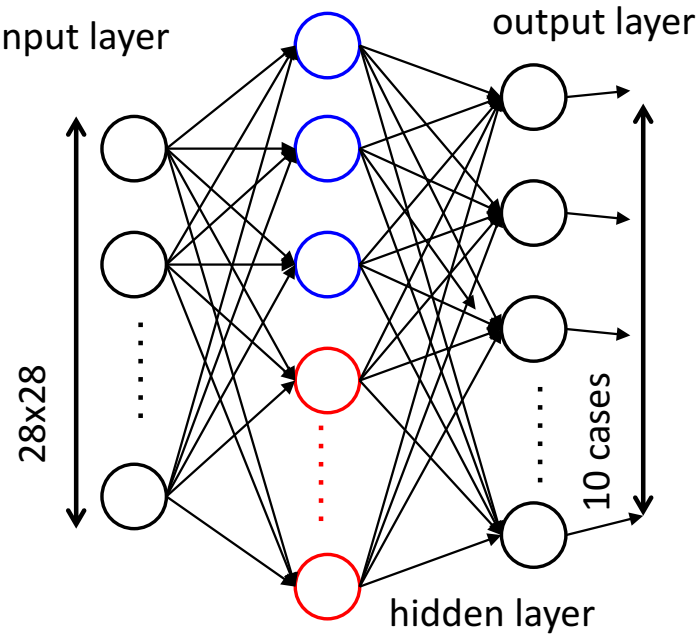
0	0	0	0	0
-1	-1	-1	-1	-1
2	2	2	2	2
-1	-1	-1	-1	-1
0	0	0	0	0

corner filter (3x3)

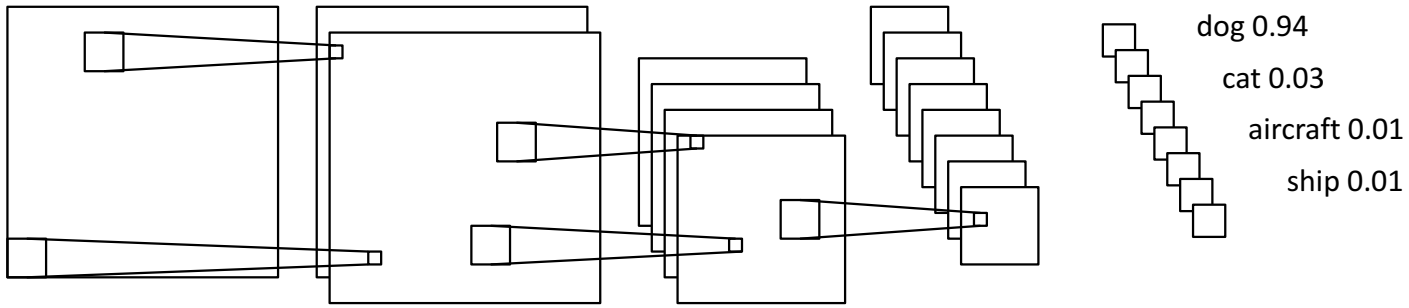
1	1	1
-1	-1	1
-1	-1	1

I made these filters up in my head

Three layer CNN can recognize numbers if filters are adjusted.



DCNN: Deep Convolutional Neural Network



Complex shape can be recognized with combination of filters
(e.g. edge recognition followed by line detection)

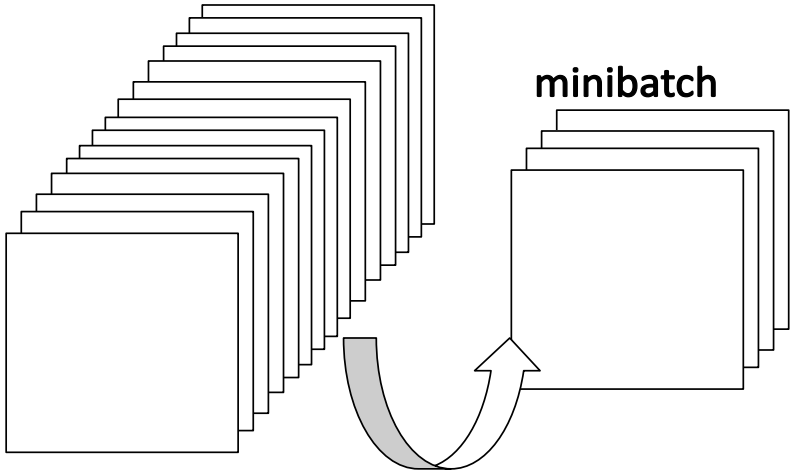
An example is the “cat neuron” found in DCNN
for image recognition (by google)
https://googleblog.blogspot.jp/2012/06/using-large-scale-brain-simulations-for.html

Microsoft ResNet used 152 layers [K. He et al. 2015]

“Cat neuron”

Training of Neural Network

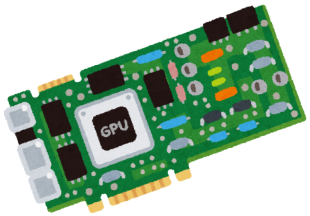
training data

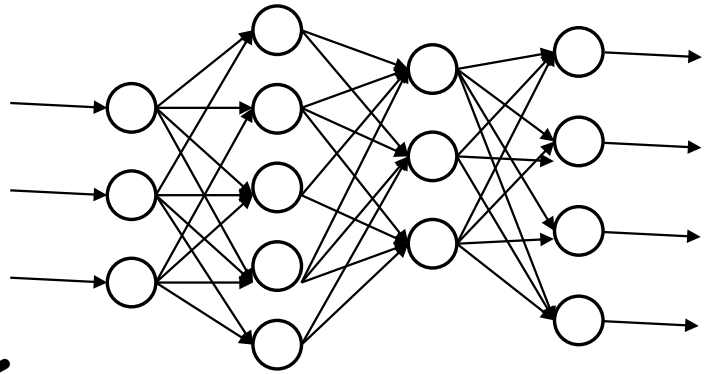


minibatch

shuffle

forward (inference, prediction)



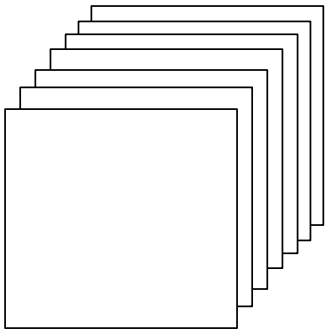


compare with training data

calculate **error** E

backward (backpropagation)

validation data



Repeat until validation error stops improving

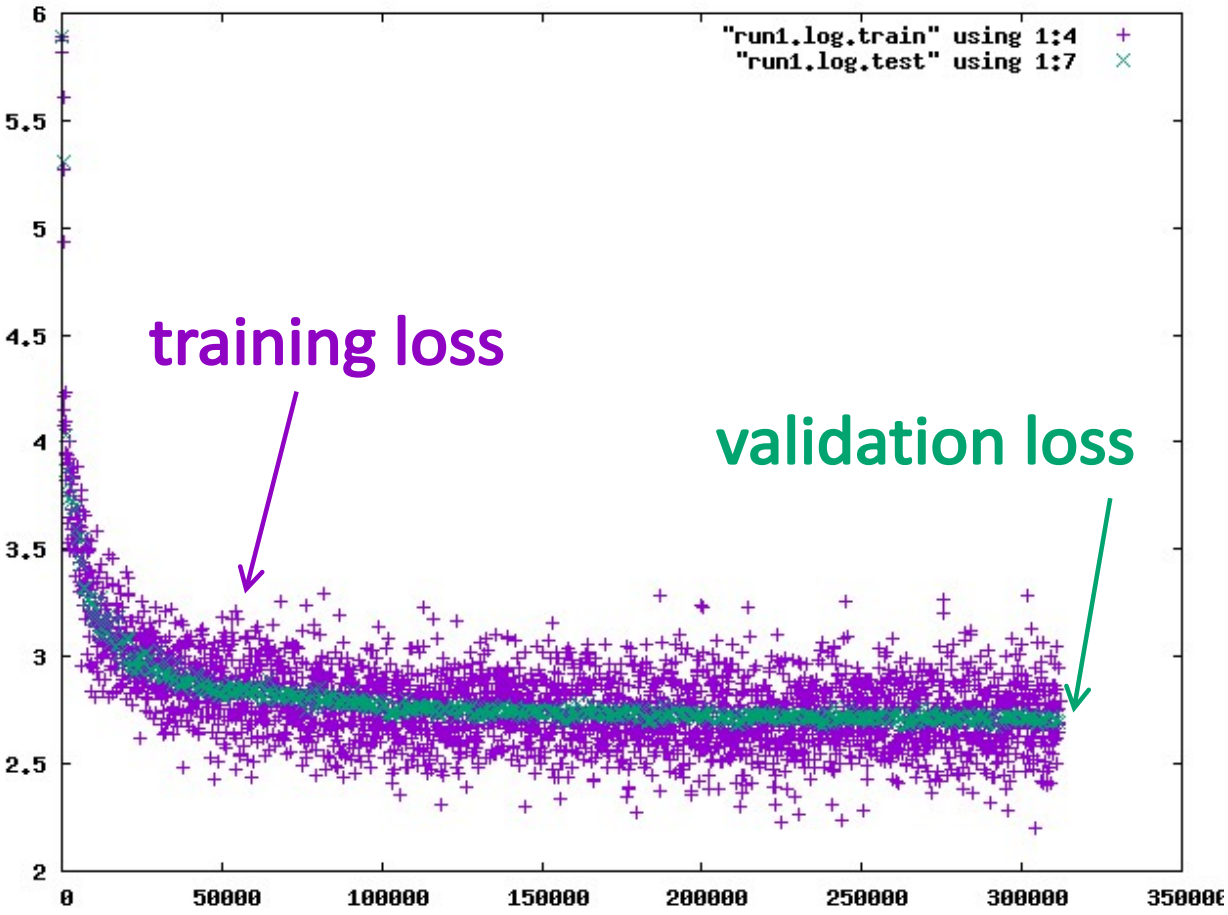
Calculate gradient $\frac{\delta E}{\delta W}$ for each connection

update weights

Learning Curve Example

minibatch size is small
32-256 typically

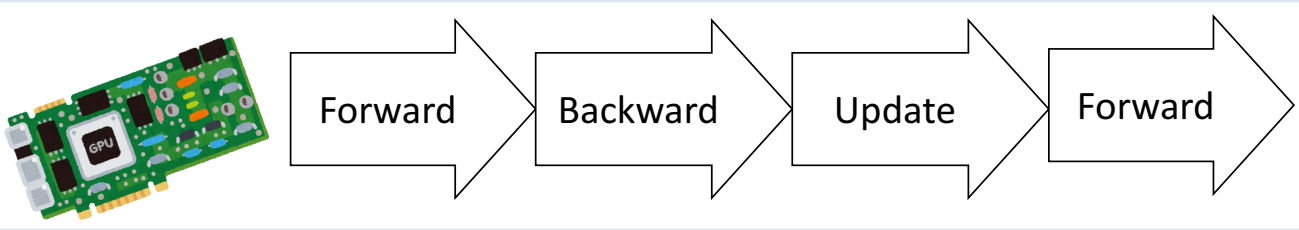
Noise of each update
is very noisy



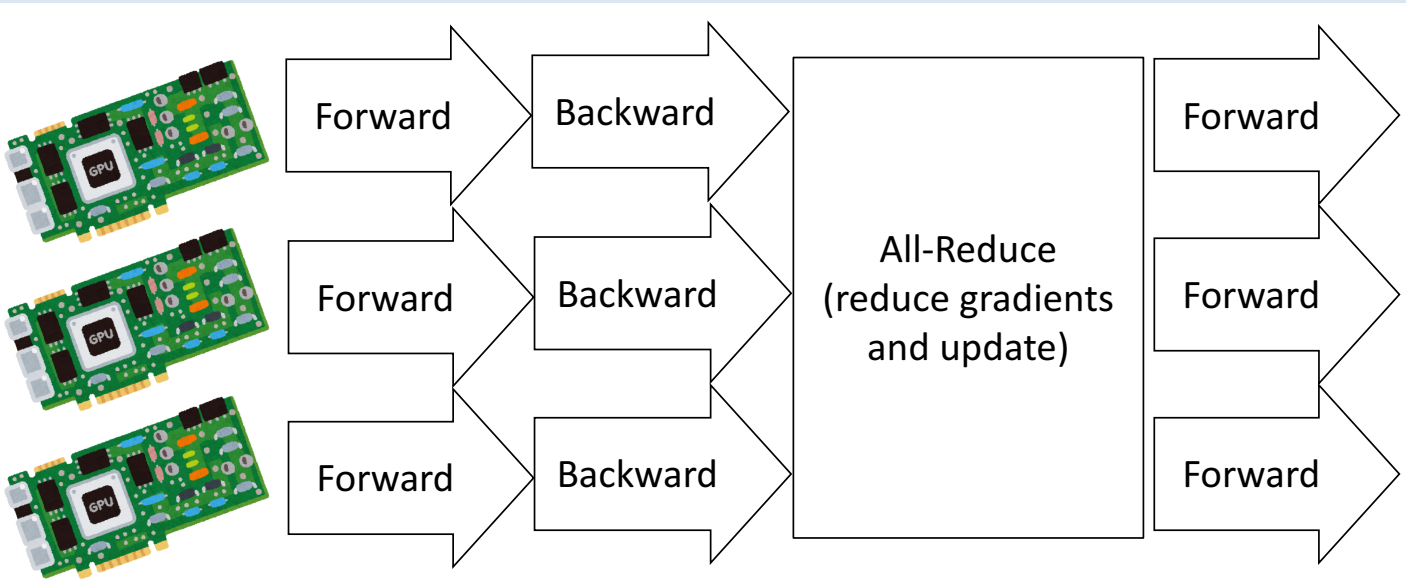
An example from
training on human
move prediction in Go
(by us, unpublished)

Training: Single GPU, Multi-GPU

Single GPU



Multi GPUs



Note: It is a “synchronous” approach. “asynchronous” approach is omitted because it‘s simply worse.

Performance of ChainerMN

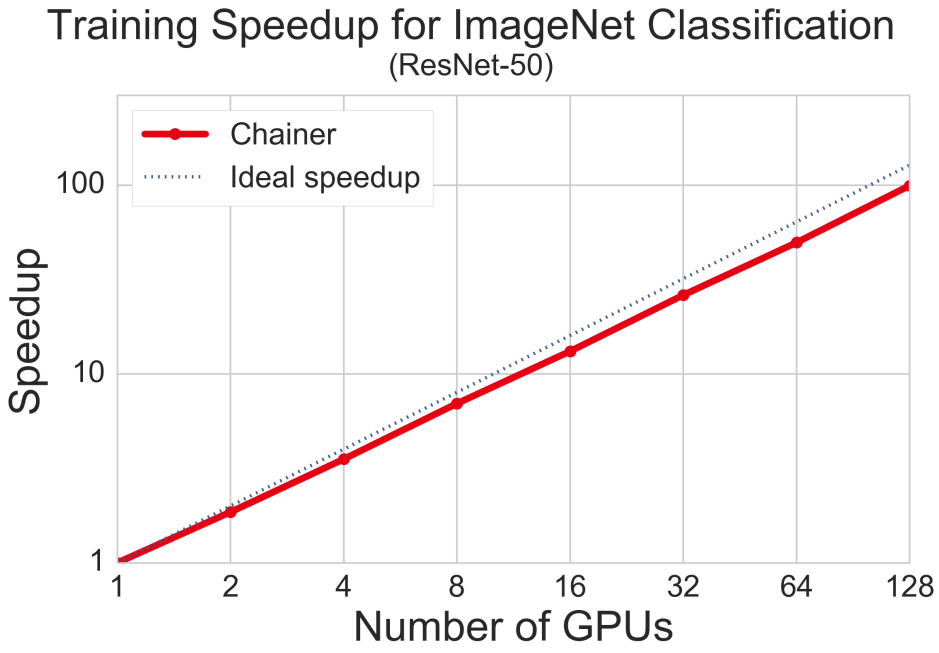
[Akiba 2017]

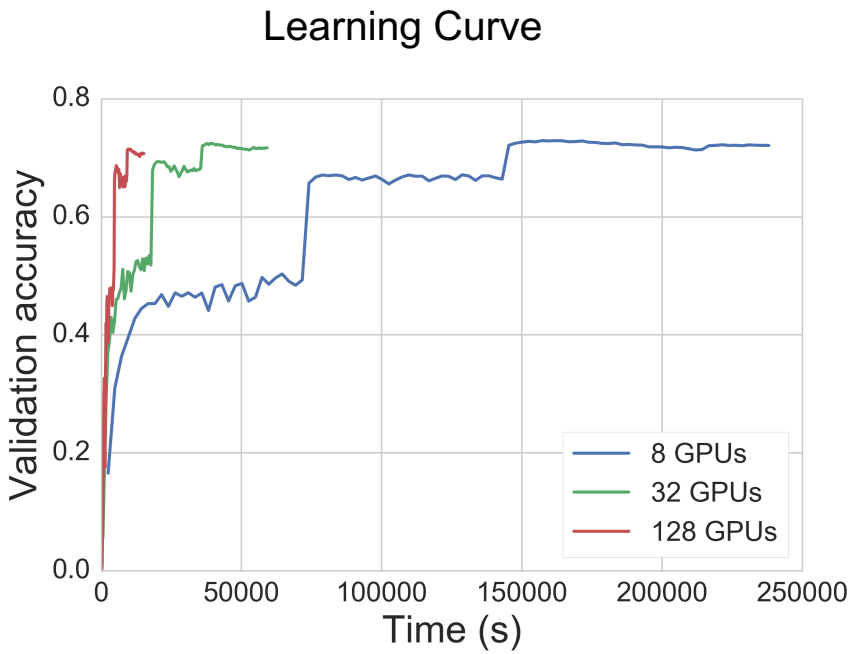
Chainer (a DL framework)

NCCL (Nvidia Collective Comm. Library)

CUDA Aware MPI (uses GPUDirect)

mpi4py (MPI for python)





4 GPUs x 32 nodes = 128 GPUs

NVIDIA GeForce Titan X (Maxwell)

ChainerMN: https://github.com/chainer/chainermn

Performance of Distributed Deep Learning using ChainerMN

https://chainer.org/general/2017/02/08/Performance-of-Distributed-Deep-Learning-Using-ChainerMN.html

Large Batch Problem

[Hoffer+ 2017] arXiv:1705.08741
Train longer, generalize better: closing the generalization gap in large batch training of neural networks

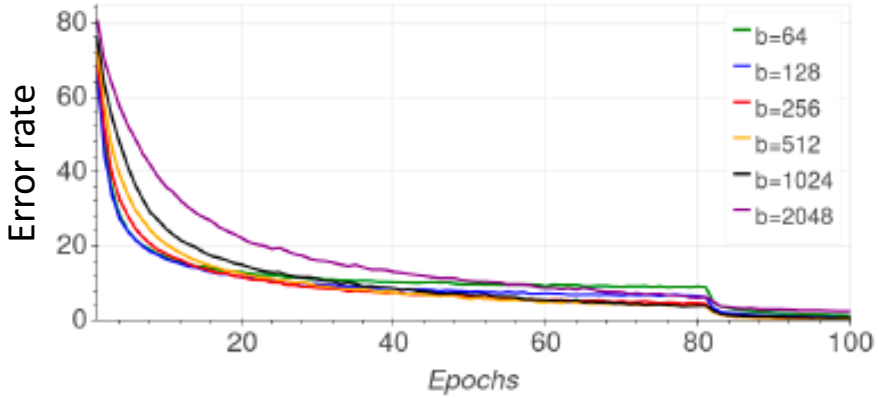
Around 100 GPU is the limit of current approach. Why?

Larger batch results in greater validation error! (long known phenomenon [Lecun+ 1998])

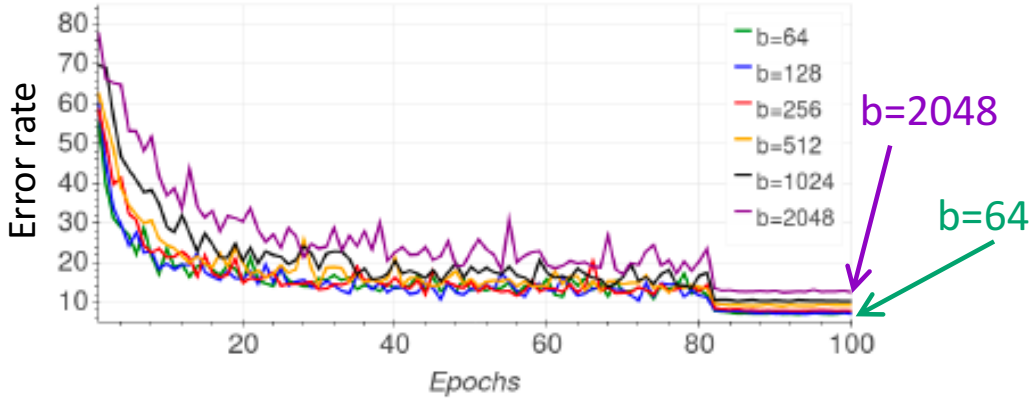
Synchronous parallel training makes minibatch size greater
(N-fold for N GPUs)

The paper partly solved it, but not enough for larger scale parallelization.

training error

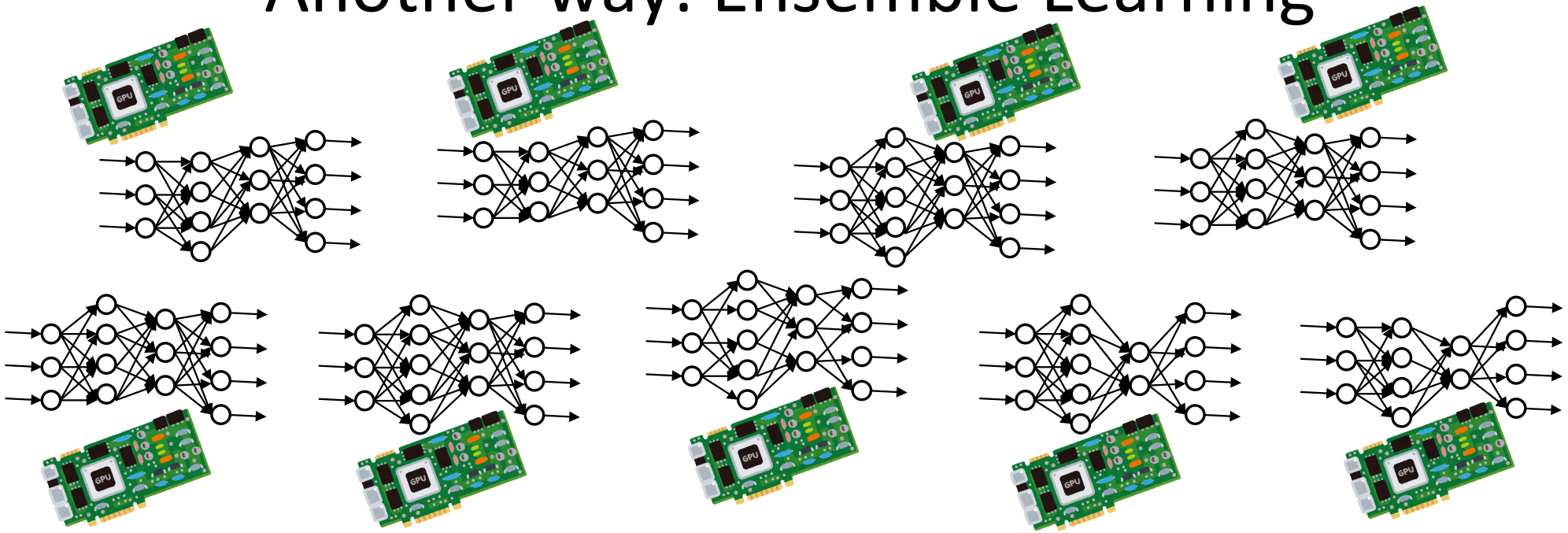


validation error



[Hoffer+ 2017] Figure 1

Another way: Ensemble Learning



train multiple models (30—100?)

do averaging / voting

improves accuracy!

So, who am I?

Parallel computing lab at graduate school	Search Algorithms
Digital wireless communication (at **FUJITSU**)	Game AI algorithms
Biometric security (finger vein recognition)	Parallel Search

Computer Go book (in Japanese)

I am now working for RIKEN AIP
(Center for Advanced Intelligence Project)

Wanted!
People with HPC background
and interested in AI

Our supercomputer RAIDEN ranked 4th in Green500.
(I was in charge of the procurement)

雷電 RAIDEN