Feature Extraction Optimization for Multi-Core Architecture in Java

Maochen Guan (mg3364@nyu.edu)

Tianshuo Deng (td859@nyu.edu)

New York University 2012

Tweet Emotion

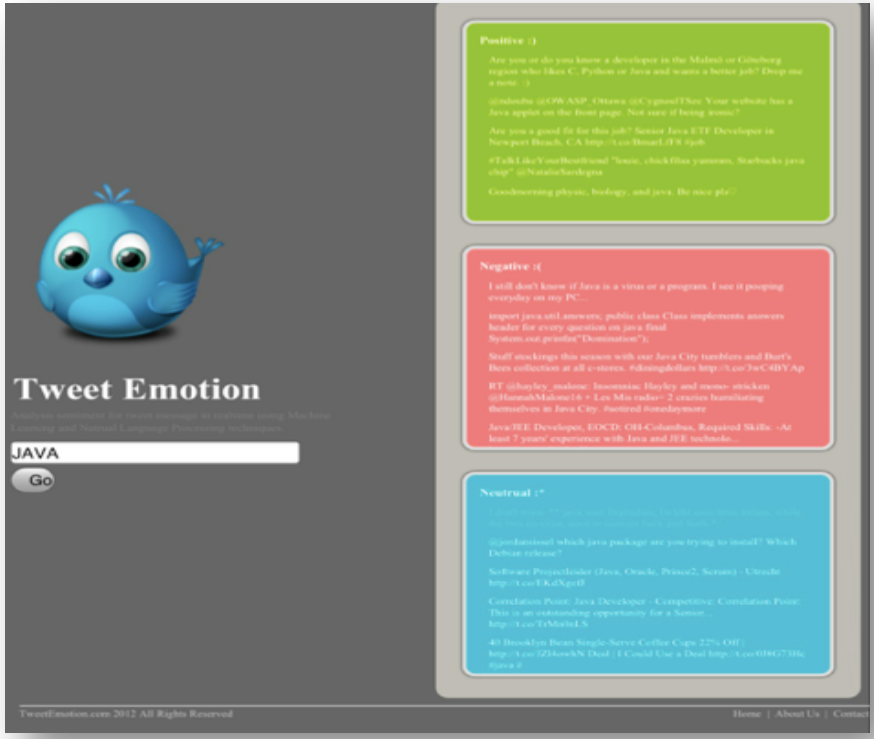
Website for analysis Tweet sentimental

Groups:

Negative,

Positive,

Neutral



Tweet Emotion

Conditional Probability

$$P(\text{Class} \mid \text{Tweet}) = P(\text{Class} \mid \text{Features})$$

P(Negative | Tweet)

P(Positive | Tweet)

P(Neutral | Tweet)

Java Concurrent Model

Data Structures

Barrier

Coordinate Threads

Atomic Integer

Aggregate Multiple Operations

Thread safe operation.

ConcurrentHashMap

Thread-safe, lock on key level.

Java Concurrent Model

Programming Models

Thread Pool Model

Fork and Join Model

Discussed in later slides.

Feature Extraction Procedure

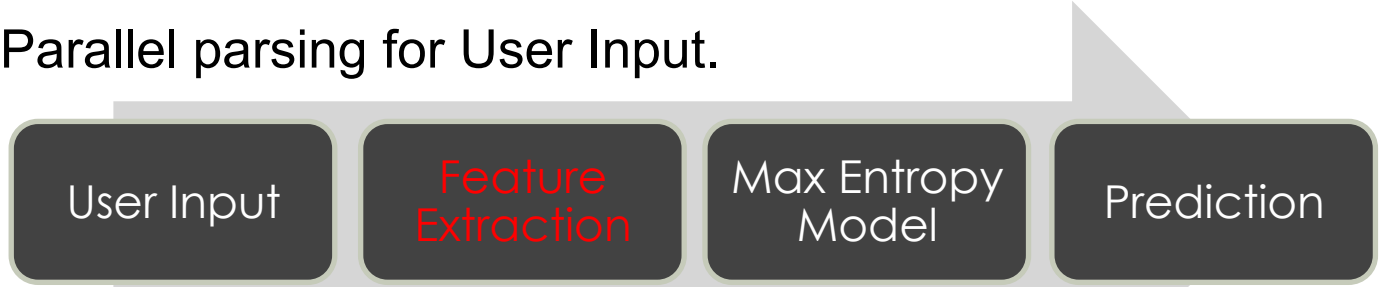
Two Kinds Feature Extraction

1. Model Training Process (Reduce training time)

Massive Training Data - Data Level Parallelism.

2. Execution Process (Reduce execution time)

Parallel parsing for User Input.

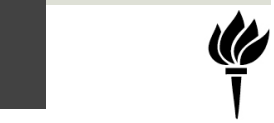


Data Level Parallelism

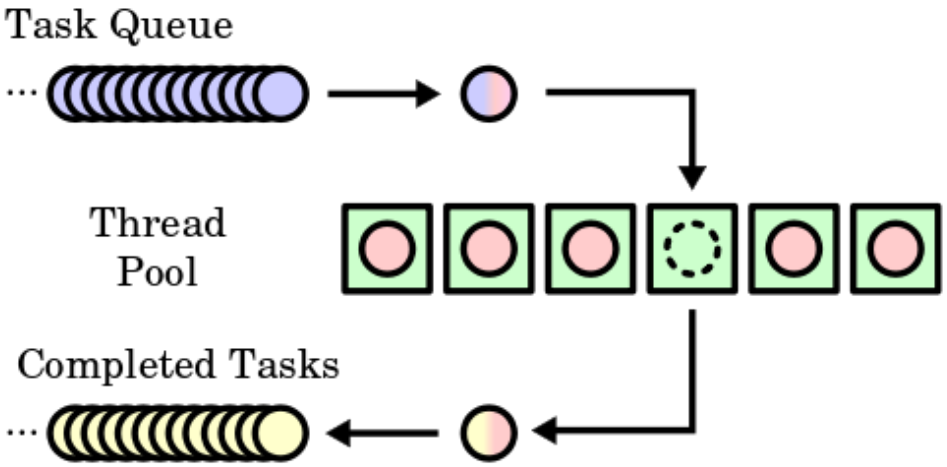
Data level parallelism (Reduce Training Time)

1. Thread Pool Model
2. Fork and Join Model

Data Level Parallelism (Cont')



Thread Pool Model



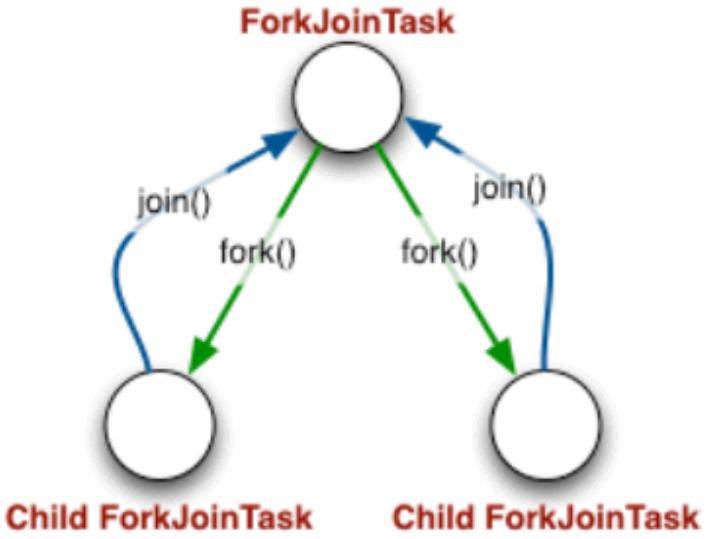
Using ThreadPoolExecutor Service.

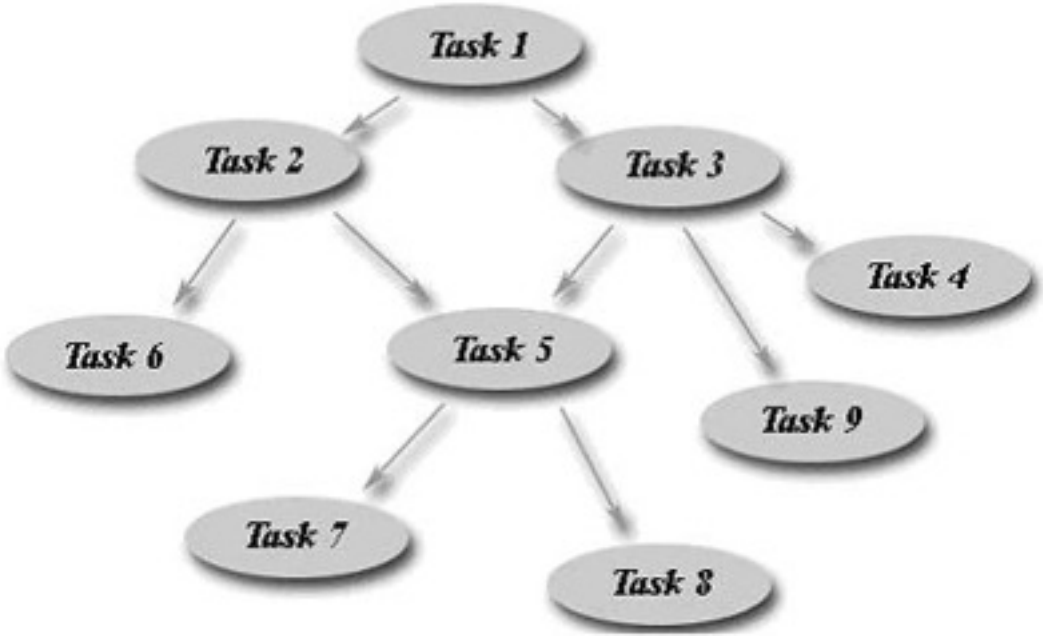
Advantages

- Thread Scheduling by JVM.
- Reduce the cost of spawning new thread (reuse thread).

Data Level Parallelism (Cont')

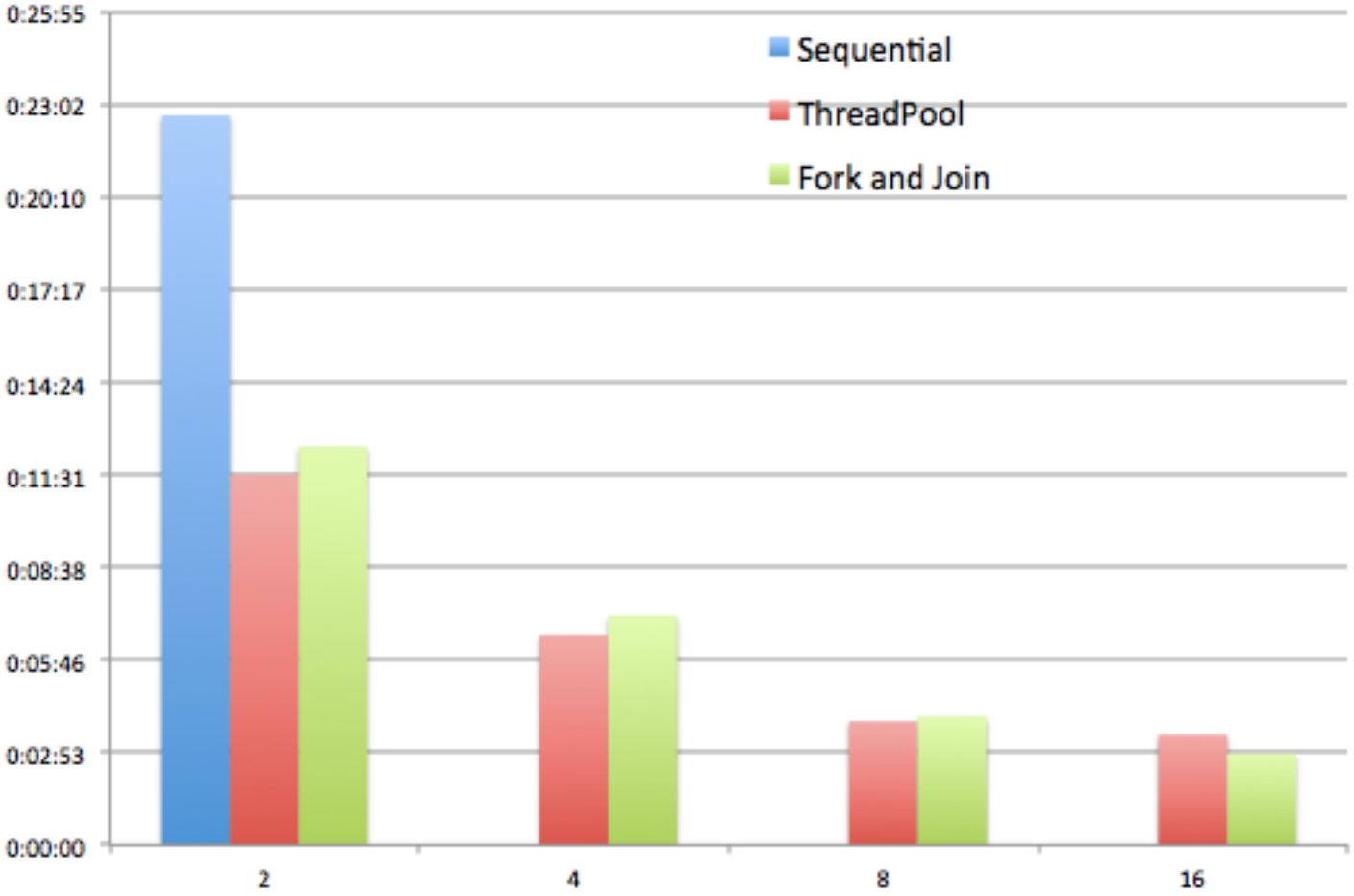
Fork And Join Model





Performance Speed Up

Reduce the training time



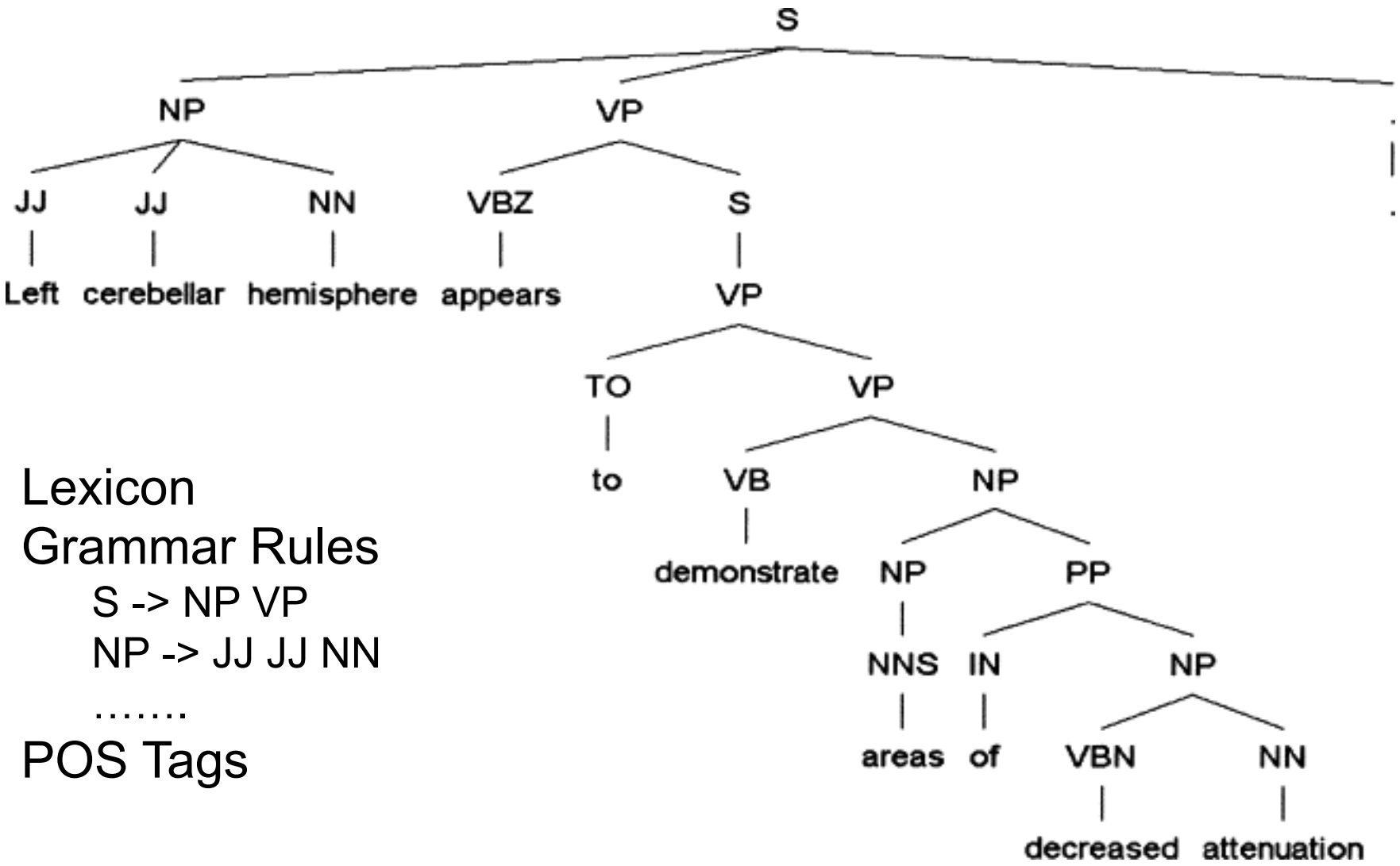
Parsing

User Input -> feature extraction -> output

Parsing is most time consuming in feature extraction

Parallel Parsing-> reduce response time

Parsing



Lexicon

Grammar Rules

S -> NP VP

NP -> JJ JJ NN

…….

POS Tags

Parsing – CKY Algorithm

Dynamic Programming

Dependency between cells – Potential Thread Blocking

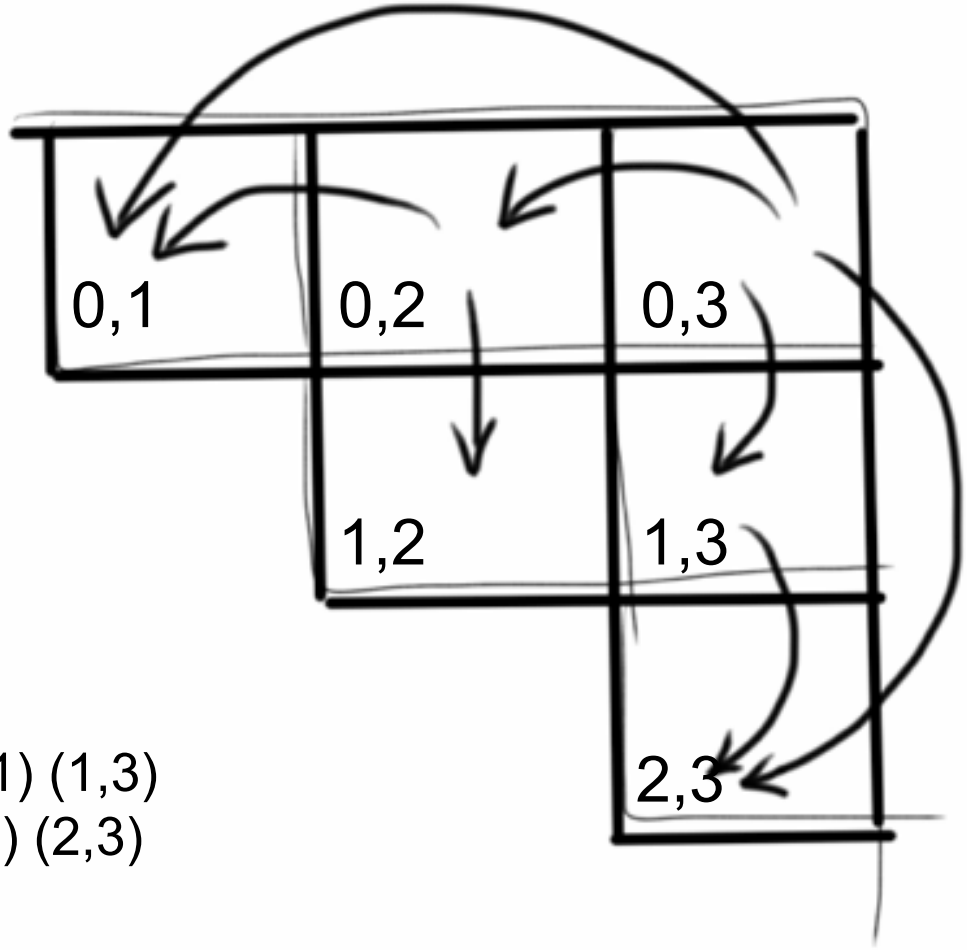
	The	clam	's	group	had	knowledge
	1	2	3	4	5	6
0	D [0,1]	NP [0,2]	POSSP [0,3]	NP [0,4]	S [0,5]	S [0,6]
1		N, NP [1,2]	POSSP [1,3]	NP [1,4]	S [1,5]	S [1,6]
2			POSS [2,3]			
3				N,NP [3,4]	S [3,5]	S [3,6]
4					V, VP [4,5]	VP [4,6]
5						N,NP [5,6]

[0,2] Grammar Rule: NP -> D N

Parsing – CKY Algorithm

Dynamic Programming

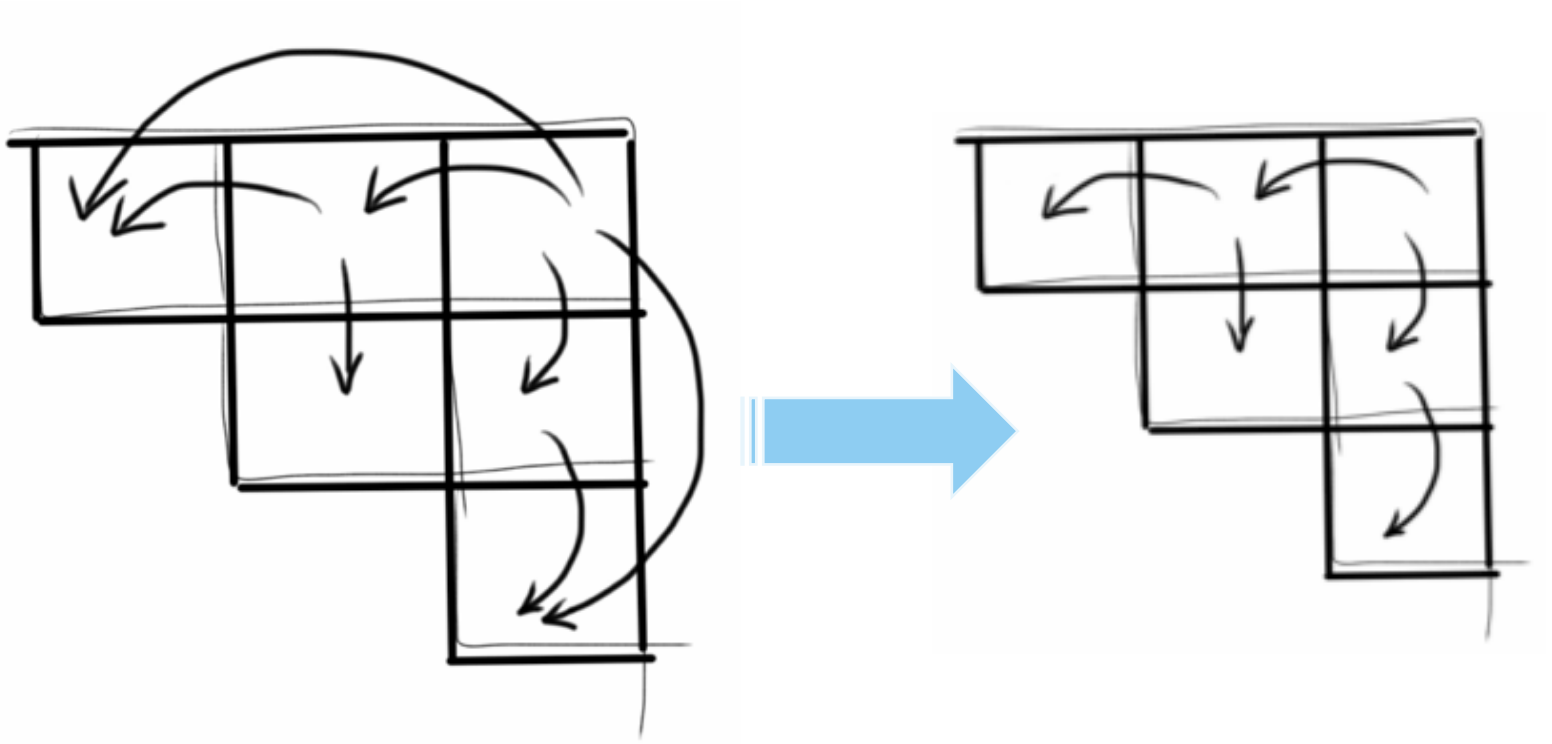
Dependency between cells – Potential Thread Blocking



(0,3)<- (0,1) (1,3)
(0,3)<-(0,2) (2,3)

Parsing – CKY Algorithm

Pruning Dependency



Transitive dependencies are redundant and can be reduced.

Top Down vs. Bottom Up Parsing

Top Down Parsing

Recursive Call

Stack space consumption directly related to the sentence length.

Requires Threads Join

Bring thread block in the Join Step.

Top Down vs. Bottom Up Parsing (Cont')

Bottom Up Parsing

Non blocking

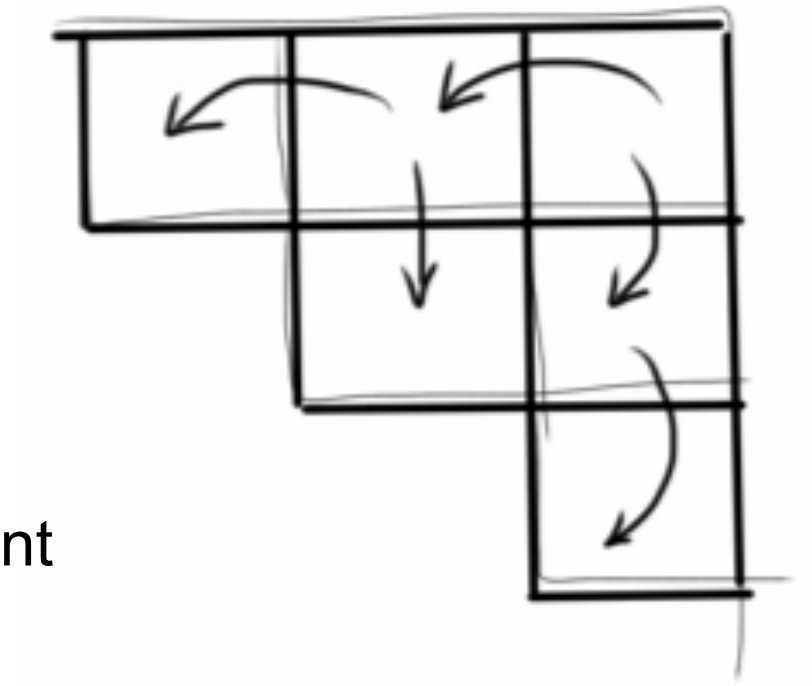
Eliminate the thread join operation.

Spawn threads bottom up

Curtail the blocking threads, especially in the initial iterations.

Use This Parsing!

Parallel On Cell



Bottom Up Parsing

Dependency Reversion

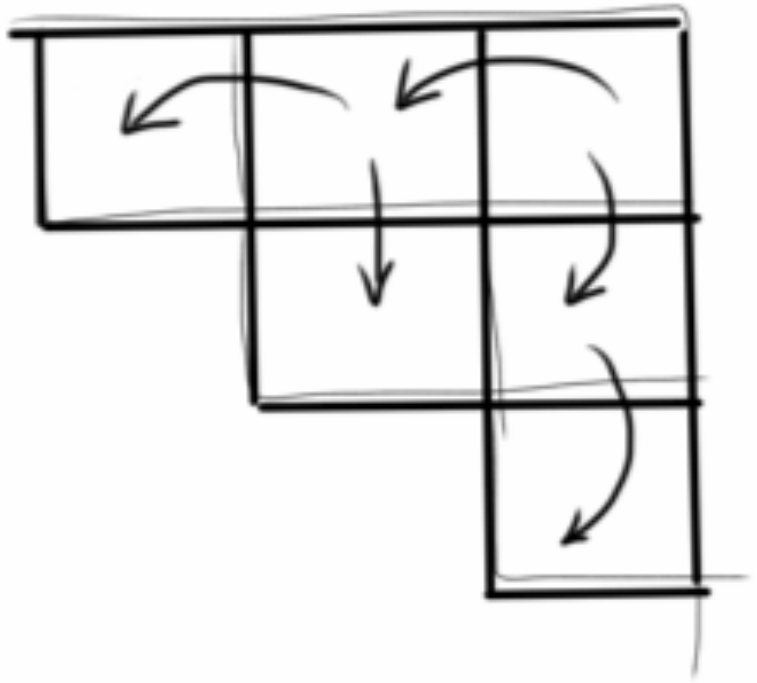
Track Cell Dependency Count

Spawn New Thread when dependencies are satisfied

Disadvantage of Parallel On Cell

Imbalance Load

- Thread Number differ between different layers
- Bottom Layer: N Threads (N is Sentence Length)
- Last Layer: 1 thread
- i^{th} Layer: N-i+1 threads



Refined On Cell Parallelism (Cont')

Pair Level Parallelism

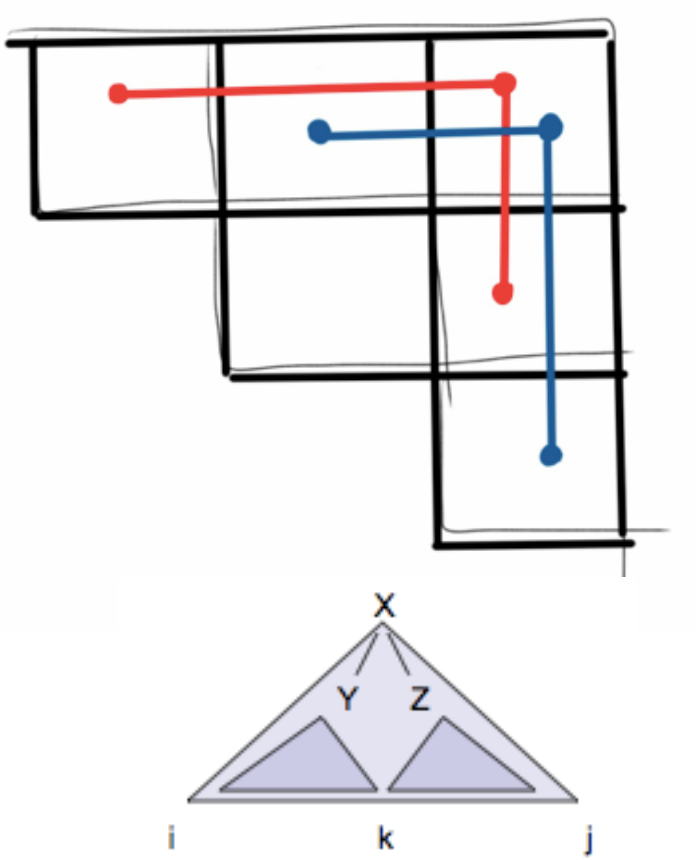
Non-Blocking

Cell Pair Coordination

Finish computing current cell when all dependent pairs are finished.

Pair / Pair Coordination

Shared state structure, multiple threads have to update same cell.

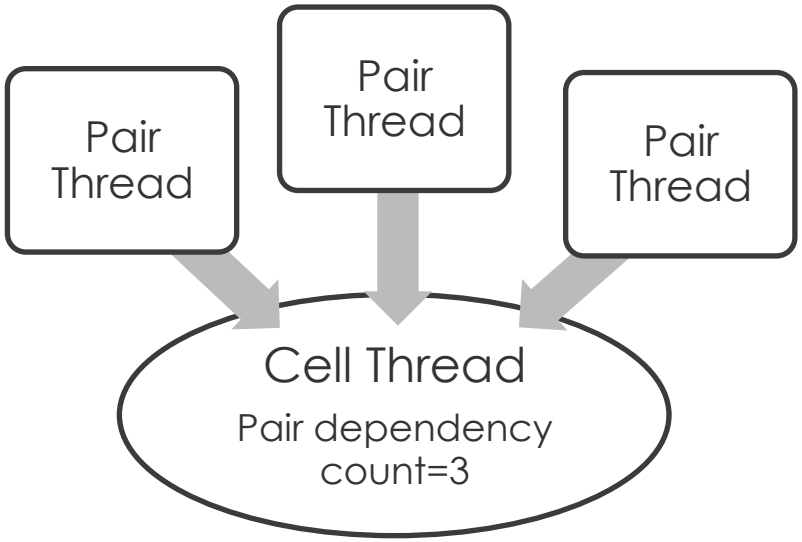


Refined On Cell Parallelism

Pair Level Parallelism

Cell Pair Coordination

Pair Dependency Count



Refined On Cell Parallelism (Cont')

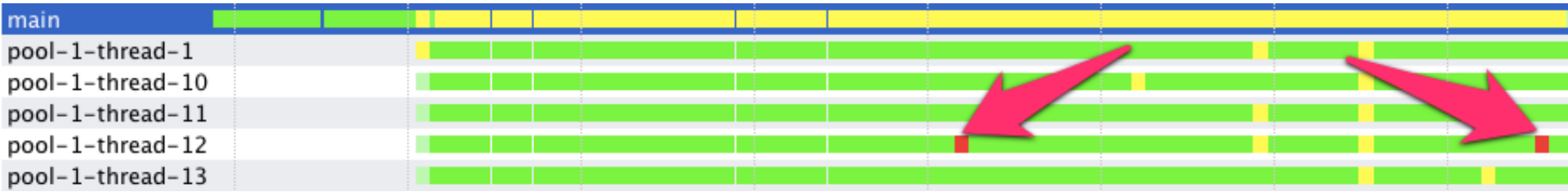
Pair Level Parallelism

Share State Structure

Lock on stateMap Object

Cause thread blocking

```
synchronized (currentState) {
    if (currentState.getScore() < updateScore) {
        currentState.state = stateX;
        currentState.score = updateScore;
        currentState.children = Arrays.asList(
                leftChildState, rightChildState);
    }
}
```



Refined On Cell Parallelism (Cont')

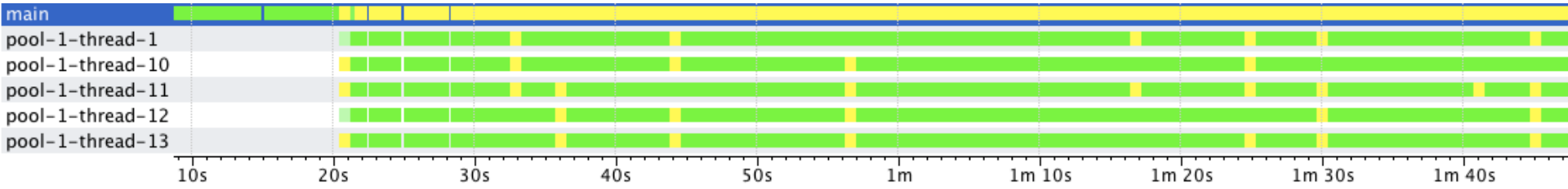
Pair Level Parallelism

Non-Share State Structure

Every pair threads has its own stateMap object.

Avoid Locking Shared Objects

stateMap needs to be Merged by Cell Level Thread.



Refined On Cell Parallelism (Cont')

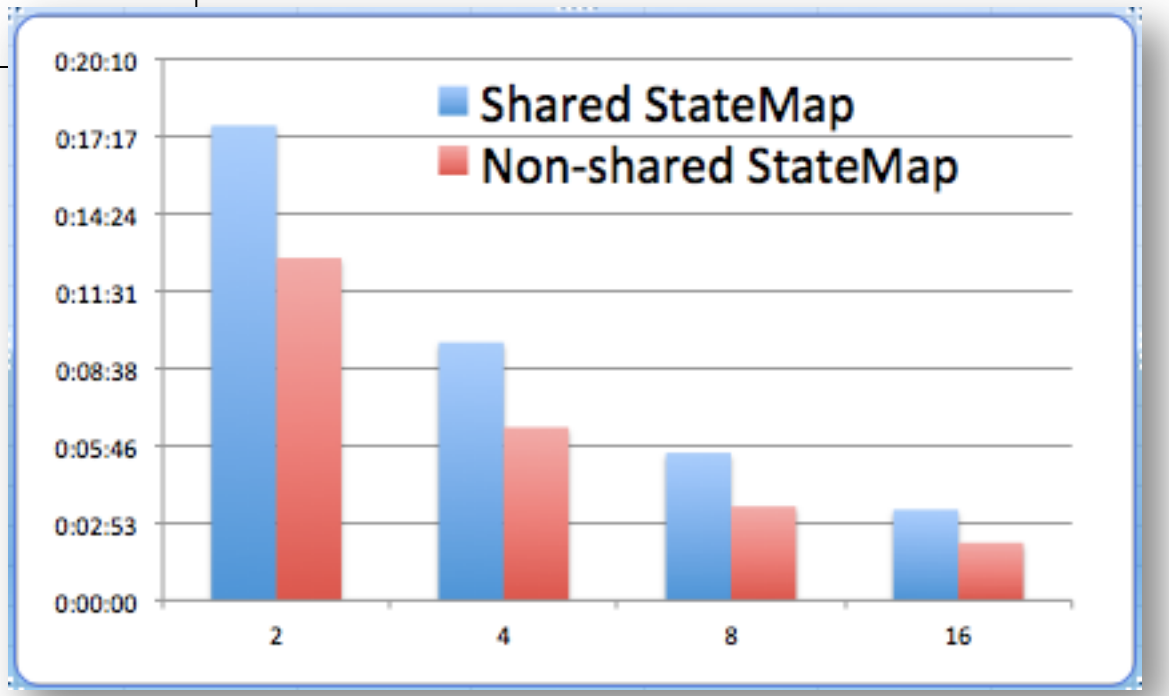
Performance Comparsion (Share State/Non Share State)

Configuration

CPU 16-Core 2.1GHz AMD Opteron 6272

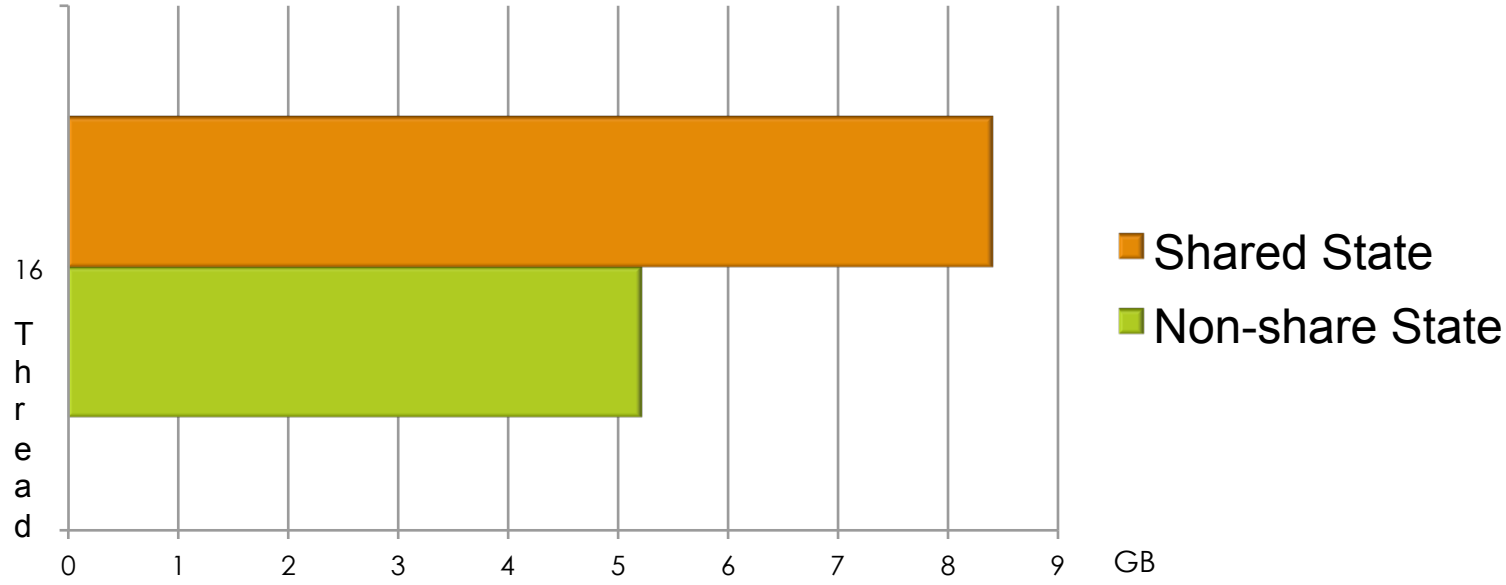
Memory 70GB

Data size: 86 sentences



Refined On Cell Parallelism (Cont')

Memory Footprint



Refined On Cell Parallelism (Cont')

Pair Level Parallelism

Share state structure vs. Non-share state structure

	Pros	Cons
Non-Share State	No lock -> non blocking	Requires Merge Requires more Memory
Share State	No merge required	Less efficiency compared to Non-share model when enough memory ensured

Conclusion

Feature extraction achieves performance boost by parallelization.

Parallel data reduces training time.

Parallel parsing reduces response time.

- Bottom up eliminates lock.
- Trade off between shared data access and non-share data access.