A Lock-Free concurrent algorithm for Linked lists

Speaker: Alexandre Walzberg

Date: April, 2nd 2009

Paper: **MikhailFomitchev and Eric Ruppert, Lock-Free Linked Lists and Skip Lists.*PODC'04*, 2004.**

Plan

1. Introduction

The goals

Simply allow to make concurrent operations on a same shared ordered linked list…but without using OS synchronisation primitives (lock-free).

Operations: Search (for a key), **Insert** and **Delete** an element

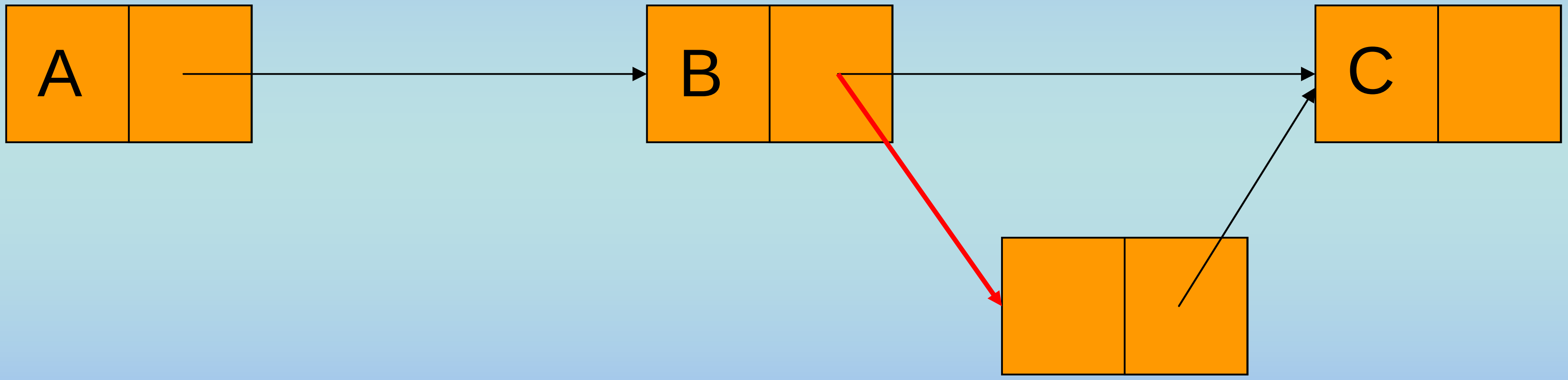
The motivation

- Interested in the CAS instruction
- Already Worked on a Spin-locks algorithm, wanted to see further : **free-locks algorithm**
- Linked lists are everywhere and are the base of all the data structures

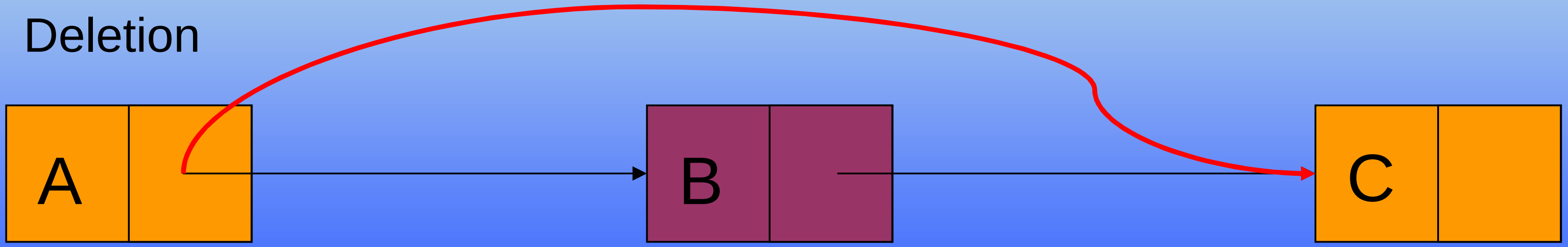
2. The Problem

Reminder

Insertion



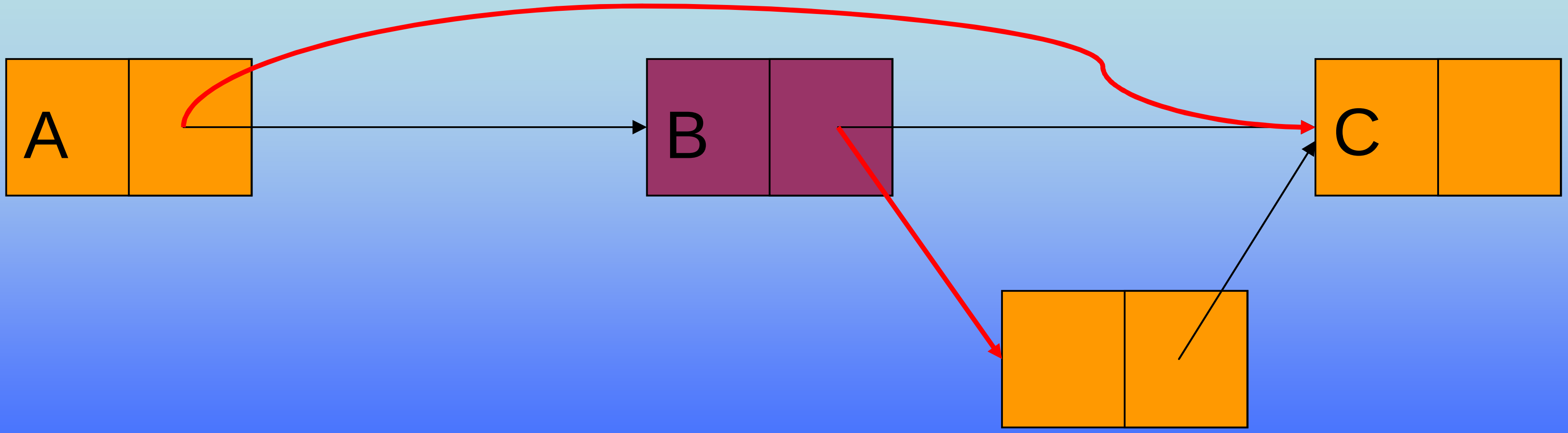
Deletion



2. The Problem

The problem

Problem: Several concurrent process => can lead to loss of data



2. The Problem

The solution: CAS

⇒ We do NOT want to change a pointer if another process change it between the time we read it and the time we write it

⇒ To avoid that: we need a way to verify that the pointer did not change and to change it value atomically.

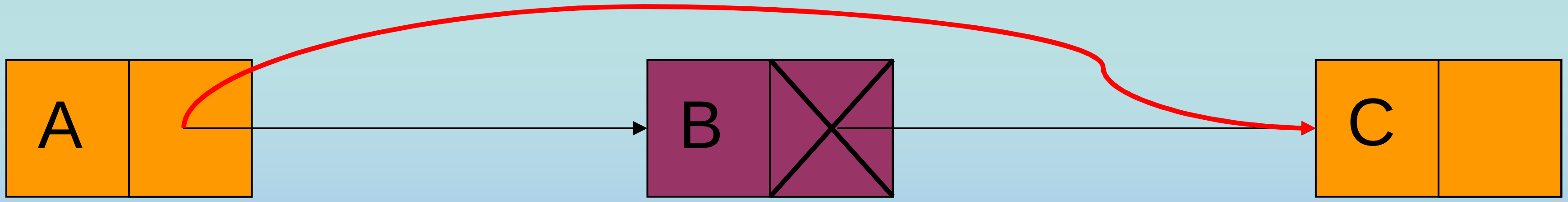
⇒ The Compare And Swap atomic instruction is designed for this purpose :

```
word CAS ( word* Address, word exceptedValue, word
newValue){

            value = *Address;
            if ( value == expectedValue ) *Address = newValue;
            return value;

}
```

3. The Algorithm

The idea (1)

First solution (Harris):



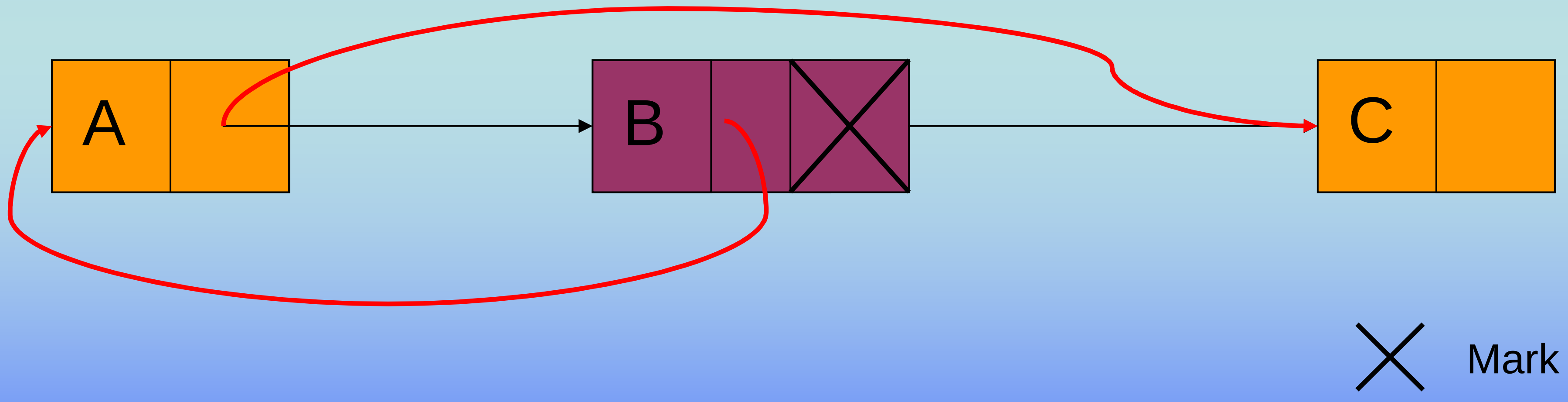
A marked node can be deleted safely, as any of the process can then insert a node between B and C or delete C.

Problem:
If another process wants to delete C or insert an element between B and C, it has to re-search the element from the head !!

3. The Algorithm

The idea (2)

Improvement 1:

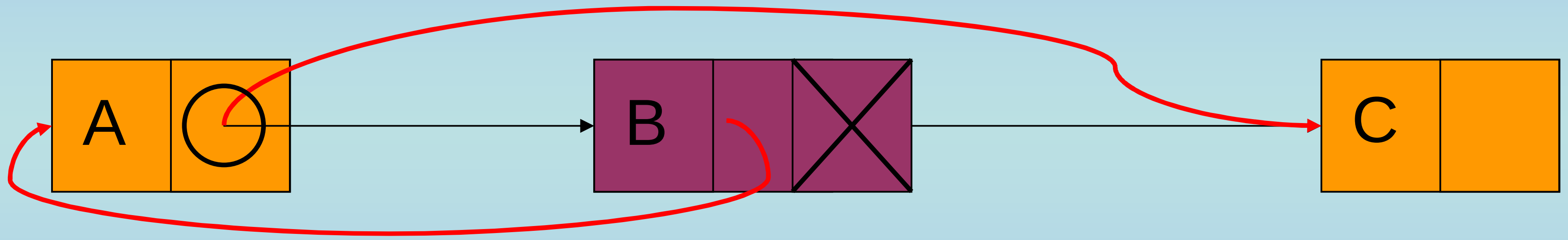


Problem:
Long back-links chain can appear !!

3. The Algorithm

The idea (3)

Improvement 2:

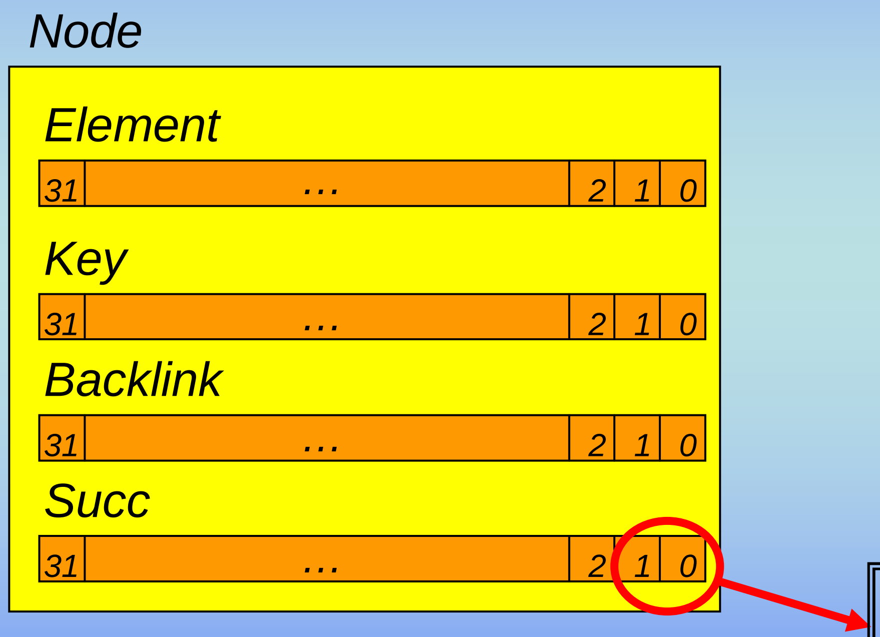


A flag prevent any operation on the node (except the deletion of the next node)

✕	Mark
○	Flag

3. The Algorithm

The Data Structure



2 LSB of a pointer : always 0 !
- one will represent the flag
- the other one the mark

Allow to update the 3 information atomically

Head key = - ∞

Tail's key = +∞

3. The Algorithm

The functions

Insert

Delete

Search : just use SearchFrom

SearchFrom : used by Delete, Insert, TryFlag and Search

TryFlag : set the flag of a node

HelpFlagged : logically delete a node

TryMark : set the mark of a node

HelpMarked : physically delete a node

Try-er: loop that run until a CAS operation succeed (additional check before CAS, update after the CAS)

Helpers: allow other process to help another one deleting his node (pre-emptive system)

Conclusion

Algorithm complexity : n for each operation (+ contention)

Allow several process (running on different processor or the same one) to make safe operations on shared linked lists in the same time

No System Call, a process is never blocked !

No locks ! => No dead-locks !!

Useful for SMP OS design but also for any other concurrent programs

Constitute a good start point for concurrent algorithm for other data-structures

Any questions ??

Thank you for your attention!