Fully Adaptive Snapshots

Hagit Attiya and Idan Zach
Department of Computer Science
The Technion

The Collect Problem

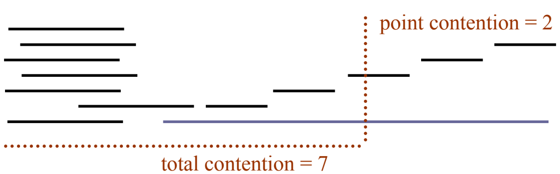
- Collect up-to-date values of active processes
- A simple solution uses an array with N entries
  - ⇨ $O(N)$ step complexity

But …

- N is large
- Often only few of the processes take steps

Adaptive Algorithms

- Step complexity depends only on the number of participating processes
  - Point contention during an operation: Max number of processes taking steps together



Adaptive Collect Algorithms

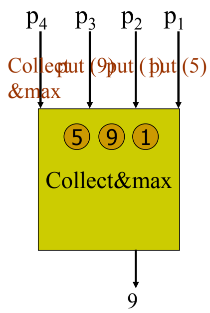
[Attiya, Fouren & Gafni, 1997] $O(K)$ (total contention)
[Afek, Stupp & Touitou, 1999] $O(k^4)$
[Attiya & Fouren, 2003] $O(k^2)$

But local step complexity depends on the total contention!

[Afek, Stupp & Touitou, 2000] $O(k^3)$ local + shared
This work $O(k)$ / $O(k^2)$ local + shared
For a generalized problem

Collect&f

- Compute a function f on the stored values
  - associative, commutative and idempotent
  - put&f(*val*), collect&f()
- A weaker variant: gather&f
  - Earlier gather may be more up-to-date than a later gather



Implementing Gather&f / Collect&f

- Naïve method: Use a gather / collect object
  - Local step complexity is not adaptive since f is applied to all stored values
- Better method: Incrementally calculate the result as values are stored, and save it for later use
  - Extend idea from [Afek, Stupp & Touitou, 2000]
  - Fit into [Attiya and Fouren, 2003]

Two Examples

- **Pile** maintains the maximum value stored

  [Afek et al. 2000] $O(k^4)$

  using our **gather&*max*** $O(k^2)$

- **Active set** returns the current set of active processes

  [Afek et al. 2000] $O(k^3)$

  using our **gather&*union*** $O(k^2)$

Repetetive Collect&f

- Many algorithms invoke many put / collect operations within one high-level operation

⇨ Amortize costs over all invocations

- put&f a single step
- collect&f $O(k)$ steps
- start and end $O(k^2)$ steps

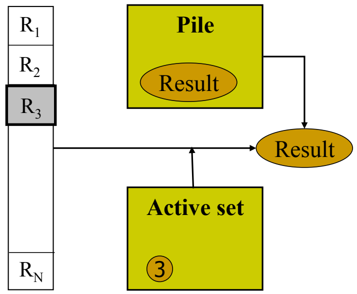
Implementing Repetetive Collect&f

```
start {
   sign-in (active set) }

put&f {
   Ri = value }

collect&f {
   get active set
   read regs of active proc.
   read from Pile
   merge the results }

end {
   put last result in pile
   sign-out (active set) }
```



Implications

- Atomic snapshots
  - [Afek, Stupp & Touitou, 2000] *O(k⁴)* local+shared
  - [Attiya & Fouren, 2003] *O(k³)* shared only
  - This work *O(k²)* local+shared
- Immediate snapshots
  - [Attiya, Fouren & Gafni, 1997] *O(K³)* shared (total)
  - [Afek, Stupp & Touitou, 2000] *O(k⁵)* local+shared
  - This work *O(k³)* local+shared
- Long-lived *(2k-1)*-renaming
  - [Attiya, Fouren & Gafni, 2003] *O(k³)* local+shared
  - Using our work