CONGEST's SYNC ASSUMPTION IST UNREALISTISCH

ABER

unsere sync algos sind auf async "simulierbar"

Wdh

Async: keine Runden, nur Init und Received ev.s

Nachrichtenübermittlung in endlicher Zeit

↳ zur Laufzeitanalyse cappen wir diese Zeit (delay) auf 1 abstrakte Einheit

↳ es kann aber z.B. sein, dass manche Nachr. andere "überholen"

d.h. OCR) ← Runden bedeutet R. diese Einheit

↳ Korrektheit muss für beliebige Delays gelten

↳ Hieraus entspringt das Problem

- darum gehen sync algos idR nicht (ex Folie 4)

Nun: sim. Sync in Async

mit Pulsgebern / synchronizern

- jeder Puls = eine Runde
- Puls muss nur "lokal" bewirkt sein

⇒ ein Knoten kann schnell werden

d.h. Puls $i+1$ kann starten, wenn alle Nachbarn Nachrichten i empfangen haben

Problem: man weiß nicht, von wem man Nachrichten erhalten sollte

Safety ist ein Knoten, wenn alle seine Nachbarn seine Nachrichten erhalten haben

Safety rausfinden = Bestätigungs-Antwort

⇒ doppelte Laufzeit, moderater Overhead

dadurch: Puls ist korrekt, wenn Puls i+1 erst generiert wird, wenn der Knoten + alle Nachbarn safe für Puls i sind

⇒ wie stellt man fest, ob alle Nachbarn safe sind ?

hierdurch unterscheiden sich die Synchronizer algos überhaupt erst

Synchronizer α

- sobald ein Knoten safe ist, sendet er "safe" an alle Nachbarn

⇒ Algos mit R Runden, M Nachrichten in:

gut: $O(R)$ Runden 2R Runden

schlecht: $O(M + Rn)$ Nachrichten Rn = jede Runde an alle Nachbarn

jeder Knoten sendet R Runden

$\sum \deg(v) = m = |E|$

Synchronizer β Puls durch Leader in BST

Pulsgeben: Downcast eines Signals wenn alle safe sind

Safety: on Leader upcasted

$O(RD)$ Runden (wegen 2 up/downcasts)

$O(M + Rn)$ Nachr.

(pro Runde pro Knoten 1 Nachricht an Parent "safe")

+ Breitensuchbaum in

① $O(D)$ Runden, $O(m \cdot D)$ Nachrichten
↳ durch α-Synchronizer + BST algo für synced CONGEST

② "Dijkstra" mehr Runden, weniger Nachrichten

α/β Hybrid:

kombiniere Vorteile der beiden Algos

via Graphzerlegung / Cover (wie Cluste mit überlappung)

beeinflussen Laufzeit

(ρ, φ, ℓ)-Cover

ℓ ↖ Anz. Cluste

⇒ Cluste $C_{1 \dots \ell}$

- Jeder Knoten ist in irgendeinem Cluste i
- Jedes Cluste hat ein Zentrum, was alle Knoten in Distanz ρ erreichen kann
- φ: wieviele Knoten es zw. Clustern geben darf (cp. Folie 10)

α in Clustern

β außerhalb Clustern

Bsp. - $\rho = 1$
- $\ell = 4$
- $\varphi = ???$

Boswana-Sen Algo berechnet diese Cover!

⇒ mit Sync α können wir BoswanaSen auf async laufen in

$O(k^2)$ Zeit
$O(mk^2)$ Nachrichten

} k irgeine Konstante

Synchronizer γ

- Intra-Cluster: Sync β
- Cluster ist safe, wenn alle im C. safe sind
- Safety im Cluster checkt nicht – alle Nachbarn sollen safe sein und es gibt ng extra-cluster Kanten

Knackpunkt

also: extra-cluster sync über α

- Ist das eigene Cluster safe, informieren Knoten mit Kanten zu anderen Clustern jene über die eigene safety
- Knoten upcasten diese empfangene Information

⇒ Sind alle Nachbarcluster safe, kann neue Puls gen. werden

ABER:

ABER: CONGESTION CONSTRAINT

↳ können nicht wild safety nachrichten spammen

↳ pro Knoten pro Nachbarcluster mit $\geq$ Index nur eine Kante zum Versenden benehmen

wie? Folie 14 Voraussetzungen (inkl. gezählt)

die gewählten Kanten sind bidirektional! das ist nur um bessere Komplexität zu kriegen

WEITERS - CLUSTER ÜBERLAPPEN SICH

→ Safety de Nachbarcluster muss an alle Cluster, in denen der Knoten ist, upcasted werden

→ Knoten darf erst ticken, wenn alle seine Cluste safe sind

Laufzeit wenn wir Cover haben:

$$O(R \cdot l \cdot P) \qquad O(M + R(\varphi + l_n))$$

Jede Runde up/downcasts

Jeder Knoten in l Clustern, jedes braucht up/downcasts

für die l up/downcasts

Laufzeit mit Baswana Sen Coverberechnung $O(Rk^2)$

da $k = O(k^2)$

$p = O(k^2)$

$O(M + R(n^{1+1/k} + kn) + mk^2)$

Ist $k = \log n$, so ist ein allgemeines Ergebnis:

SYNC ALGOS IN ASYNC SIMULIEREN

IN

wenig overhead

$O(R \log^2 n)$ Runden

nicht größer als M maximal...

$O(M + R n \log n + m \log^2 n)$ Nachrichten

pro Runde pro Knoten $\log n$ Nachrichten

⇒ synchrones CONGEST ist fine – nur logarithmische Overheads um auf realitätsnahe Systeme zu übertragen