組込みシステム向け FRP 言語の静的スケジューリングを用いた並列化

櫻井義孝・渡部卓雄

東京工業大学

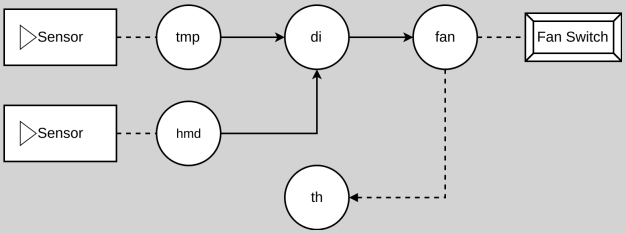
概要

- 本研究の目的は小規模組込みシステム向け FRP 言語の応答性向上である．
- 既存の小規模組込みシステム向け FRP 言語である Emfrp はシングルスレッドで動作する．
  - マルチスレッドが使用可能な環境では計算資源を十分に活用できない．
- 本研究では応答性向上のために静的スケジューリングを用いて Emfrp を並列化することを提案する．

Emfrp [Sawada2016]

小規模組込みシステム向け FRP 言語

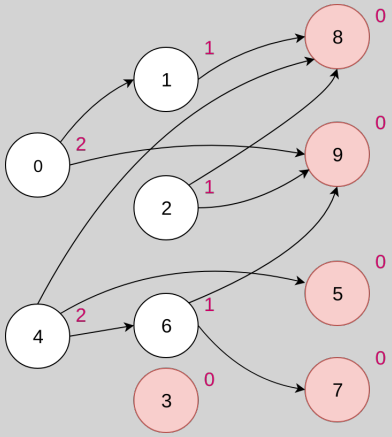
```
module FanController     # 不快指数からファンの
in  tmp: Double,         # オンオフを決定するアプリケーション
    hmd: Double
out di:  Double,
    fan: Bool
use Std

# discomfort index
node di = 0.81*tmp+0.01*hmd*(0.99*tmp-14.3)+46.3

# fan switch
node init[False] fan = di >= th

# threshold
node th = 75.0 + (if fan@last then -0.5 else 0.5)
```



- Emfrp のプログラムは，時変値をノードとし依存関係を辺とした有向非巡回グラフ (DAG) を構成する．
- 現在の Emfrp の処理系ではノードの更新は逐次的に行われる．左に挙げた例の場合，DAG をトポロジカルソートした tmp→hmd→di→th→fan の順に更新が行われる．この 1 回の更新 (サイクル) を繰り返すことでリアクティブな動作を実現している．
- 1 サイクルにかかる時間がシステムの応答性を決める．本研究では，マルチコアシステムでの応答性向上を目的とした並列化方式を提案する．プログラムの実行環境として，OS のない環境，あるいは FreeRTOS 程度の小規模 RTOS を考える．
- これらの環境では OS のスケジューラによる適切なスケジューリングは期待できない．本研究では各サイクルの計算時間を短縮し，可能であれば応答時間を予測できるような静的スケジューリング方式を提案する．

並列化アルゴリズム

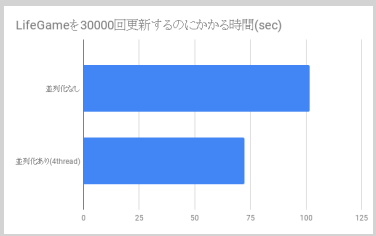
1. 各ノードから sink ノードへの最長距離を計算する．
   - sink ノードは出次数が 0 のノード (右図では赤いノード)
2. 同じ距離にあるノードを並列に更新する．
   - OS のスケジューラを利用できない環境ではそれぞれのノードをどのプロセッサが更新するかを指定する必要がある．
   - これは各スレッドがなるべく同じ数になるように割り当てる．
     - 下の表の sink ノードへの距離 0 のノード $3, 5, 7, 8, 9$ を 2 スレッドで処理するならば $3, 5$ をスレッド 1 が $7, 8, 9$ をスレッド 2 が更新．

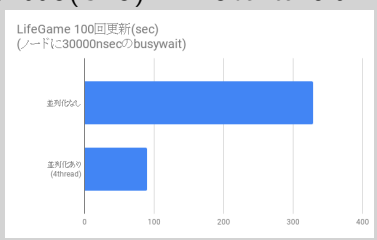


出力ノードまでの距離	2	1	0
同時に処理するノード	0,4	1,2,6	3,5, 7,8,9

実験

実験は上記のアルゴリズムで並列化した際にどの程度プログラムの応答性が向上するかを調べることを目的とする．実験対象として 100x100 のサイズの LifeGame を実装し，並列化しなかった場合と Pthread を用いて並列化した場合での実行時間を計測した．実験環境は i5-7200U(CPU) 上の Ubuntu18.04 で行った．





左は LifeGame を 30000 ループするのにかかった時間であり，右はノードに 30000 ナノ秒のビジーウェイトを挟んで 100 ループしたときにかかった時間である．左では 4 スレッド使用したにも関わらず 30%程度しか時間が削減できていない．一方で，ノードの実行時間が大きくると右のように 4 スレッドで実行時間が 27%になった．