並列分散環境時代のプログラミング言語に向けて

安部達也

戦略的高性能計算システム開発に関するワークショップ
2010 年 8 月 2 日

Real World Haskell のコードを GHC 6.12.1 用に改変

注意：以下は単なる記述の例の紹介で，並列化による効果は無い！

```
module Sorting where

import Control.Parallel (par,pseq)
import Control.Parallel.Strategies (rnf,NFData)

sort :: (Ord a)=>[a]->[a]
sort (x:xs) = lesser ++ x:greater
    where lesser  = sort [y | y<-xs, y<x]
          greater = sort [y | y<-xs, y>=x]
sort _ = []

sort' :: (NFData a)=>(Ord a)=>[a]->[a]
sort' (x:xs) = rnf greater `par` (rnf lesser `pseq` (lesser ++ x:greater))
    where lesser  = sort' [y | y<-xs, y<x]
          greater = sort' [y | y<-xs, y>=x]
sort' _ = []
```

一般に「逐次 → 並列」で期待すること

かってにしてくれるようになること：

- par の引数ごとにスレッドを生成してくれる，
- 利用環境にあわせて**適当に**割り当ててくれる．

しかも，かってにしてくれていたことを失わない．

かってにしてくれていたこと：

- 型検査（型推論）してくれる，
- 参照透明性を保証してくれる，
- 非停止性を警告してくれる，等．

「並列」ゆえに問題となること

デッドロックもかってにどうにかしてくれるのが望ましい．

どうにかする方法：

- 検知して警告してくれる，
- ソフトウェアトランザクションメモリを勧めてくる，

がすぐに思いつくが，それぞれ

- 人的コスト（定理証明 $\leftarrow \cdots \rightarrow$ モデル検査）が高い，
- 実行効率が悪い，

といったことがある．

「分散」ゆえに問題となること

かってに局所性を高く保持してくれるのが望ましい．

どうにかする方法：

- 計算を一つの空間とした上で距離が導入されている，
- 統計情報を使ってデータを再配置してくれる，

というものを聞いたことがある．

気にしていること

「5 年後にセンター運用可能な高性能並列計算機システムは，どういうシステム仕様が考えられ，そのために，今後どういう研究開発をしていくべきなのか.」

アプリケーション開発者，数値計算ライブラリ，ミドルウェア，システムソフトウェア，ハードウェア開発者がよいものをつくってくれても，高水準プログラミング言語の理論はついていけないのではないか.

システムプログラミング言語 C が計算機に命令を伝えるということだけでなく，今後，プログラミングをする全てのユーザにとって最も優れた言語となってしまうのでは？