関数型言語 Haskell による並列データベースの実装

當眞大千 (並列信頼研究室)

高信頼性かつ高速なデータベース

・脆弱性を悪用されると多大な被害が出る
・CPUはマルチコア化が進んでおり、マルチコア対応は必須
・Haskellを用いて信頼性の高い並列データベースを実装

・読み込みに関して、12コアで10.37倍の性能向上
・Web掲示板サービスを開発し、Javaと比較して
読み込みで3.25倍、書き込みで3.78倍の性能差

関数型言語Haskell

・Buffer overflowや、XSS、SQL injection
といった脆弱性は型検査で防ぐことができる
・Haskellでは全ての関数や式に型があり、コンパイル時に
プログラム中の型の不整合がないことが保証される

・Haskellでは、Evalモナドといったモナドで並列実行の
機能を提供
・モナドはHaskellでメタ計算を提供するAPIのようなもの
と考えてよい (例: I/O モナド、Error モナド、STMモナド)

・Evalモナドの利用例
・sumという関数をrparの引数にすることで並列に
実行可能であることを示す
・do ~ return で複数のrparを1つにまとめrunEvalに
渡すことで並列に実行される

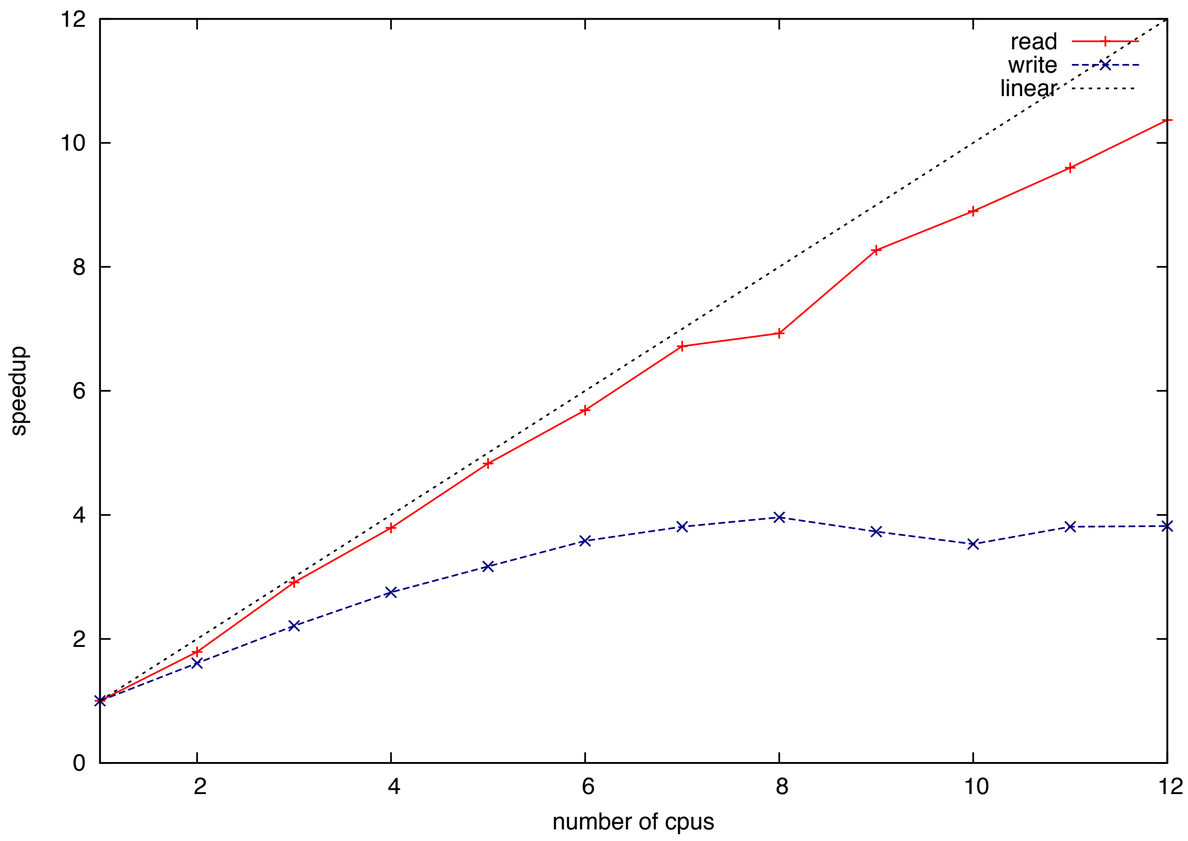
```
main = print (runEval test)

test = do
       a <- rpar (sum 0     10000)
       b <- rpar (sum 10000 20000)
       return (a,b)
```

Haskellの並列機能を活かした設計

・非破壊的木構造という手法を用いて並列にデータへ
アクセスできるように設計
・どのノードが最新なのかという情報は状態を持つため、
スレッドセーフに扱う必要がある
・HaskellのSoftware Transactional Memory(STM)を利用
・STMは、Non-Blockingで共有データを扱える
・共有データの変更は以下のように行われる
  ・他から変更がなければそのまま適用
  ・変更している間に、他から変更されれば変更処理を
  やり直す

Javaより高速なデータベース

・開発した並列データベースがマルチコア環境で性能が
出るのか計測
・12コアで実行時に読み込みで10.37倍、
書き込みで3.82倍の性能向上



・読み込みが高速なデータベース
・書き込みより読み込みが多用されるシステムに向いている

・Web掲示板サービスを開発し、Javaと性能比較を行う
・100万リクエストを処理するのにかかる時間を計測

測定方法	Haskell	Java
読み込み	16.31 s	53.13 s
書き込み	20.17 s	76.40 s

・読み込みで3.25倍、書き込みで3.78倍の性能差
・Haskellは実用的なWebサービスを提供できる

今後の課題

・分散機能(通信、トポロジーの形成)
・複数の変更をマージするアルゴリズム

・永続性(ログへの書き出し)
・現在はオンメモリ上で動作するデータベース
・ログへの書き出しを担当するスレッドを用意するなど、
並列性を損なわない形で永続性を実装