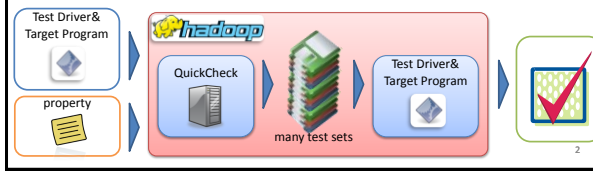
プロパティベースのテストデータの自動生成とその応用

九州大学大学院
システム情報科学府
生田裕樹

テストのコスト

- 大規模開発や信頼性を求められるソフトウェア開発ではテスト規模も大きくなるため、効率のいいテスト手法が求められる
- QuickCheckを用いてテスト仕様(property)からテストデータを自動生成し、生成した大量のテストデータを並列分散環境Hadoopを用いて高速に処理する手法を提案



なぜテストにクラウド環境を活用しないのか?

- テスト
  - 開発が進むにつれ必要なテストの量は徐々に増える
  - 多くのテストは並列化に向いている
    - 各テスト同士は依存関係を持たないほうが望ましい
- クラウド(AmazonEC2などelasticなクラウド)
  - 必要に応じた計算リソースの調整が容易
    - プロジェクトが終了したらテスト用の計算リソースは不要
    - リソースの追加による高速化が期待できる
  - MapReduceなどスケールアウトが容易な仕組み

クラウド環境はテストに非常に向いている

どのようなテストがクラウド向きか?

- 単体テストなど、バッチ処理できるもの

  →**Devver**: クラウドでRSpecやTest::Unitなど、Rubyの単体テストを高速に実施するサービス

devver.net

- 既存のテストだとテストケースを作成するコストは変わらない

propertyベースでテストデータの自動生成を行うQuickCheckに注目

QuickCheck

- QuickCheck?
  - Haskellプログラムの形式仕様をpropertyとして記述
  - property(テスト仕様)をもとに**テストデータを自動生成**する データ駆動型テストツール
- データ駆動型テスト
  - テスト仕様とテストデータを分離し、ひとつのテスト仕様に多数のテストデータを適用するテスト

5

Testing Telecoms Software with Quviq QuickCheck

- EricssonにおけるQuickCheckの適用事例
- Quviq QuickCheck: QuickCheckのErlang版
- Megaco(H.248)プロトコルの実装を検証
- 他のテスト手法で検出できない誤りや仕様の曖昧な点などを発見

- Thomas Arts, John Hughes, Joakim Johansson, Ulf Wiger, Proceedings of the Fifth ACM SIGPLAN Erlang Workshop - 2006

6

propertyの例 (sort関数の冪等性)

- 冪等性(Idempotence):

  f(f(x))=f(x)
- sortの冪等性に関する propertyとテストデータ

property

```
prop_Idempotentce  xs = (sort (sort xs)) == (sort xs)
  where types = xs::[Int]
  --xsはIntのリスト
```

テストデータ

テスト番号	値
0	0
1	2, 1
2	-1, 3
3	0, 4, 4
4	5, 2
..	..
96	79, 3, 42, 13, -7, 51,..
97	105, -24, -9, 41, 89,..
98	-65, 31, -13, -90, 1,..
99	120, -90, 32, 0, 13,..

テストデータ生成のポイント

property

```
prop_Idempotentce  xs = length xs > 2 ==> (sort (sort xs)) == (sort xs)
  where types = xs::[Int]
```

テストデータ

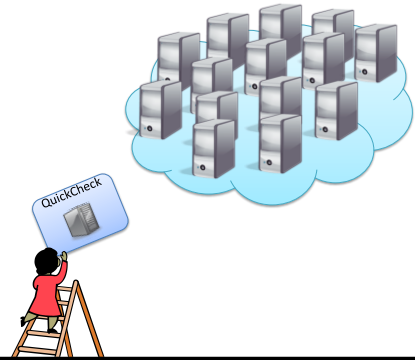
types = xs::[Int] で型を指定。
指定した型をもとにテストデータを自動生成

'==>'演算子を用いることで生成する値に制約条件を与えることが可能
(ここではリストの長さが2より大きい)

テストデータが取りうる値はランダム。
ただし、取りうる値の範囲は
テスト番号が増えるにつれて徐々に大きくなる

テスト番号	値
0	0, 3, 1
1	2, 1, -2
2	-1, 3, -4
3	0, 4, 4, 1
4	5, 2, 8, 9
..	..
96	79, 3, 42, 13, -7, 51,..
97	105, -24, -9, 41, 89,..
98	-65, 31, -13, -90, 1,..
99	120, -90, 32, 0, 13,..

QuickCheckをどのように クラウド上で動作させるか?



9

並列分散環境Hadoop

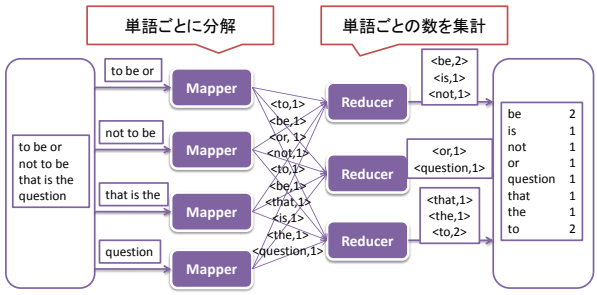
- Hadoop?
  - MapReduceのオープンソース実装版(Java)

hadoop

- MapReduce?
  - Googleが開発した並列分散環境(C++)
  - スケールアウトが容易
  - 耐障害性に優れる
  - mapとreduce関数の定義だけで分散計算が可能

10

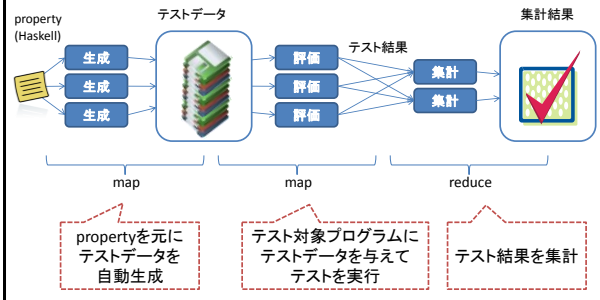
MapReduceの例 (Word Count)



入力の分割、中間結果のソートと分配、出力のマージは MapReduce側が面倒を見てくれる

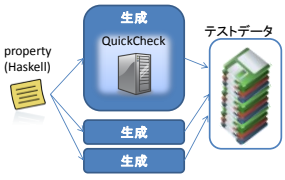
11

Hadoopを用いたQuickCheckの分散化



12

Hadoopを用いたQuickCheckの分散化 : 生成フェーズ

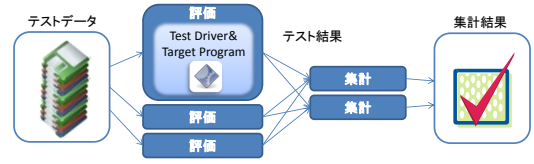



```
prop_Idempotentce  xs =
  length xs > 2 ==> (sort (sort xs)) == (sort xs)
  where types = xs::[Int]
```

- 生成フェーズ(map)ではpropertyからテストデータを生成
- map結果はreduceせずにそのままHDFSに

13

Hadoopを用いたQuickCheckの分散化 : 評価フェーズと集計フェーズ



- 評価フェーズ(map)ではテストドライバがテストデータを受け取り、テスト対象プログラムを呼び出し評価
- 集計フェーズ(reduce)ではすべての評価結果を集計

評価と集計は次のコマンドと同じ
$ cat GeneratorOutput | ./Eval | ./Aggregate

14

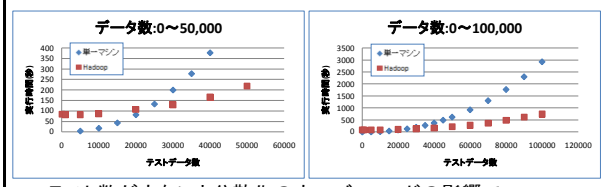
簡単な性能評価

Hadoop環境	単一マシン環境
• 4台の計算ノード • 計算ノードのスペック: – CPU: Xeon X3320 4core@2.50GHz L2 cache 6MB – メモリ: 3.2GB	• 以下のマシン1台で Hadoopを用いずに実施 • 比較用マシンのスペック: – CPU: Xeon E5420 4core@2.50GHz L2 cache 12MB – メモリ: 2.0GB

sort関数のIdempotenceをチェックするpropertyについて
さまざまなテストデータ数で実行時間を比較

15

簡単な性能調査結果



- テスト数が少ないと分散化のオーバーヘッドの影響でHadoopのほうが遅い
- テスト数が多くなると分散化の効果によってHadoopのほうが高速に処理できる

テスト数が非常に大きな場合に有効であることを確認

16