# Multicore Programming Contest Cell Challenge 2009 ツールキット解説書 ver.0.5対応版

Cell Challenge 2009 実行委員会

先進的計算基盤システムシンポジウムSACSIS2009併設企業 Multicore Programming Contest Cell Challenge 2009

#### これはなに?

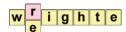
- Cell Challenge 2009規定課題に取り組む際のベースとなるツールキットの解説です
- 実際に公開されるツールキットは仕様が変更 される場合があります

## 規定課題 「**文字列の編集距離計算**」

- ・ 2つの文字列の近さを計る方法
  - かな漢字変換エンジンや、DNAの相同性検索などに利用される
- ・ 2つの文字列の 編集距離
  - 片方の文字列から、もう一方の文字列を得るための操作回数の最小値
  - 使用可能な操作は以下の3種類
    - 削除:1つの文字を取り除く
    - ・ 挿入:1つの文字を新たに付け加える
    - ・ 置換:1つの文字を別の文字で置き換える
  - 参考になるURI「wikipedia:レーベンシュタイン距離」 http://ja.wikipedia.org/wiki/ME3M83M4CNE3M83M8CM33M83M99ME3M83M83ME3M82M87ME3M83M8 5ME3M82M8FME3M82MA4ME3M83M83ME8M87M90ME9M98MA2

## 編集距離の操作例

- 「weight」と「write」の編集距離
- 以下の操作で「weight」から「write」になる
  - weight
  - 1. weight**e** (挿入:e)
  - 2. wrighte (置換:e → r)



- 3. wrihte (削除:g)
- 4. write (削除:h)
- 3回以下の操作では「weight」を「write」できない
- よって編集距離は「4」
- 削除, 挿入, 置換は文字列の中のどの位置で行ってもよい

# 編集距離計算のアルゴリズム(概要)

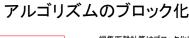
•「動的計画法(Dynamic Programming)」が有名

文字列1 weight 文字列2 write

• 操作回数を表を作って求める



#### 編集距離計算のアルゴリズム(手順) 最左列, 最上行のセルは1~N, このスライドには アニメーションが設定されています 1~Mの値と仮定 Ν 上, 左上, 左のセル値と, 当該の行 6 2 3 4 5 と列の文字列を使い, 以下の方法で w i h t g 各セルの値を計算する A) 文字列が同一ならば左上セルの値 0 2 3 4 5 異なるならば左上セルの値+1(置換) 2 3 4 5 C) 左セルの値+1(挿入) i 2 2 2 4 1 3 上セルの値+1(削除) E) (A)-(E)の最小値をセルの値に設定 3 3 2 2 3 3 テーブルの左上から値を求める 5 4 4 3 3 3 右下端のセル値が最終的な アニメーションでは列ごとに計算しているが、 「上、左上、左のセル値」が決定しているセル ならば計算手順は任意 編集距離となる





- Block1 Block3 Block2 Block4

- 編集距離計算はブロック化して部分ブロ ックごとに計算を行う
- 以下のデータがあれば、各ブロックの 計算が可能
- 文字列
- 最左列,最上列の値
- ブロック計算の入力は以下
  - 部分文字列(各問題文字列中の該当部分) - 最左列,最上列のセル値
- 出力は以下
  - 最右列、最下行のセル値
  - ツールキットでは、各ブロックの最右列、最下行の値をメモリのユーザ領域に 格納、適宜読み出して使用する

#### ツールキット ver0.1

- 2つのテキストファイルを入力すると、それらの中の 文字列を読み込んで編集距離を求める
- ・ 制約:各文字列の文字数は128の倍数
  - いろいろなサイズの例題ファイル付き (file1, file2, ..., file12)
- getrndstr.c を使用すると、任意長のランダム文字列 を生成できる

\$ gcc -O3 -o getrndstr getrndstr.c \$ ./getrndstr 128 13 > file9999

乱数種13で生成される128文字の文字列を格納したファイルfile9999を生成する

#### 実行方法(1/2)

Makefileの「USERNAME」を各ユーザ名に編集

USERNAME = USERNAME IDIR = /export/home/\$(USERNAME)/toolkit0.1 EXE\_FILE = /export/home/\$(USERNAME)/toolkit0.1/main = main spe1 OBJS

コンパイル

mmon ppt/cell/toolchain/bin/ppu-gcc -00 -Wall -m32 -lspe2 main\_ppe.c -o main lib.o ppt/cell/toolchain/bin/spu-gcc -03 -Wall main\_spe.c spel.c -o spel

#### 実行方法(2/2) [ Bdev toolkit0.1]\$ make run4 pellexec -t 30 /export/home/yoshimi/toolkit0.1/main /export/home/yoshimi/toolkit0.1/file8 10.1/file8 /export/home/yoshimi/toolkit0.1/file8 precute command: /export/home/yoshimi/toolkit0.1/file8 it0.1/file8 /export/home/yoshimi/toolkit0.1/file10 timeout 30 ますのは、2000年128 ますのは、2000年128 まずいるようなよりには、2000年128 まずいるような 第1ファイル、第2ファイルの文字列長 各文字列のブロックの数 128 : blk 1 256 : blk 2 : 224 : 224 SPE.PPEそれぞれの計算結果 ock : 0.00609088 cnt : 0.00140768 SPE,PPEそれぞれの計算時間 (後期パージョンで修正されます)

巨大な文字列長を対象にした場合、計算時間が

## Regulation:メモリ領域の割当

- PPEからSPEに渡される変数について(先頭アドレス)
- 各文字列が配置されるメインメモリ上の領域
  - buf1, buf2 それぞれ20MB
- ユーザが自由に使用できるユーザ領域
  - 20MB - buf3
- 距離計算結果格納領域(buf4[0]に回答を格納すること)
  - buf4 128B
- 各文字列の長さnum1, num2

参加者は、buf1~buf4までのメモリを 自由に使用できます

## ツールキットver0.1の計算手順

- SPE1個のみ用いて計算する(PPEのプログラムは未変更)
- 各文字列に対して、128文字を1ブロックとして計算を行う(各文字 はchar型)
  - ブロックの計算前に、以下をSPFのローカルストアに読み込む
    - ブロックの再左上セルの値(tlm:top-left-most)
    - ・ ブロックの最左列(vbuf), 最上行(hbuf)を読み込む
  - 部分文字列
  - 各SPEが処理する表は、1ブロックあたり128x128x4B=64KB
  - 計算後, 表の最右列(vbuf), 最下行(hbuf)を書き出す
  - 回答(最終ブロック最右下セルの値)をメインメモリ上のbuf4[0]に書き込む

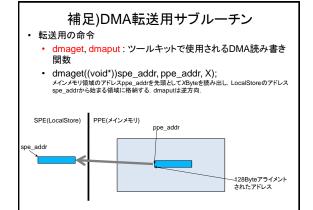
プログラムコードにもコメントがあります

## ツールキットver0.5の計算手順

- ツールキットver0.1に加え、SPEの処理を開始した後に、 参加者がPPEで処理できるようコードを修正 (ppe.cのppe\_user()関数で処理が可能)
- ・ 参加者が変更可能(不可能)なファイルの明確化
  - 修正不可なファイルはmain\_ppe.c, define.hのみ

## 規定課題への取り組み方

- 規定課題では、SPEのプログラム(spe1.c)を実装する
  - 以下のプログラムを実装してください
    - 編集距離を計算するPPE, SPEプログラム(ppe.c, spe1.cなど)
    - Makefile
    - (SPEごとに別のバイナリを実行したい場合は修正が必要です)
  - 以下のファイルは変更が許可されません
    - PPEプログラム(main\_ppe.c, define.h)



おわり 皆様のご参加をお待ちしております