ツールキット ver1.0について

先進的計算基盤システムシンポジウム
SACSIS2008併設企画
マルチコアプログラミングコンテスト
「Cellスピードチャレンジ2008」

この資料について

- ツールキットver1.0の解説です
  - ver0.1の資料を基本に作られています
  - SPEを複数使って，連立1次方程式の解を求めます
  - コンテスト参加の際に，実装の一例として参考にしてください
- 連立1次方程式の求解アルゴリズムの概要を説明します
  - コードの具体的な処理についてはREADME.txtをお読みください

規定課題の概要

- Cellスピードチャレンジ2008 の規定課題は
  「連立一次方程式の求解」です
  (概要はWebページの「規定課題詳細」を参照)
- 係数行列A, 右辺ベクトルbが与えられたときに,
  解ベクトルxを求めるプログラムの性能を競います.

$$Ax = b$$

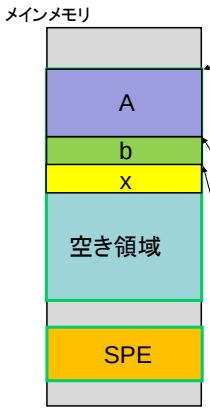
- AはN×N行列(各要素はfloat型:4バイト)
- x, bはM×N行列

ツールキットver1.0でやっていること

- SPE複数で連立1次方程式を解く
  - SPEの数は変えられますが，デフォルトで7個(最大)使う
- 問題の制約
  - 行列サイズNは32の倍数
- main_spe.cの関数spe_soleqs()内のコードを書き換えて実装する
  - コンテスト参加時はツールキットの内容を保持する必要はありません.
    自由にコードを記述してください

データの初期配置 1/2

- 行列A,b,xは以下のように配置される



アドレス

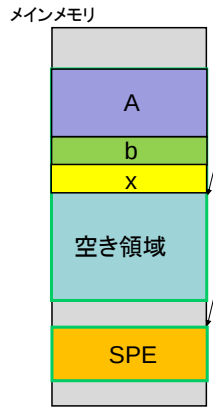
buf
- 参加者が自由に利用可能な作業用メモリ領域の先頭アドレス
- 128バイトアライメント&アクセスヒント設定済
- 先頭から係数行列A (サイズ：N×N×4 バイト)が格納されている

b = buf+N×N×sizeof(float)
- 右辺ベクトルb(サイズ：M×N×4バイト)が格納される先頭アドレス
- 列方向にデータが並んでいることに注意

(先頭アドレスから，第1列N要素，第2列N要素，…，第M列N要素の順)

x = buf+(N×N+M×N)×sizeof(float)
- 解ベクトルx(サイズ：M×N×4バイト)が格納される先頭アドレス
- データの並びはbと同様

データの初期配置 2/2



空き領域
- アドレス：buf+N×(N+2M)×sizeof(float)
- 参加者が自由に利用可能
- サイズは 行列サイズと同じ
  N×(N+2M)×sizeof(float)

SPE間通信用アドレス ls_addr[7]
- SPE間通信用にマップされたアドレス
- 物理メモリが確保されているわけではない
- ls_addr[0]～ls_addr[6]まで，各256KB
- このアドレス空間にDMA通信することで，各SPEのLocalStoreへ書き込むことができる

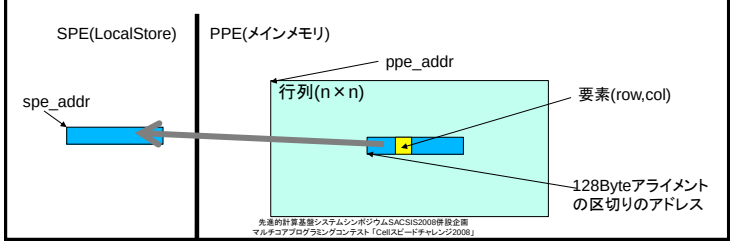
ツールキットで採用したアルゴリズム

- 基本はver0.1と同様
- LU分解，前進代入，後退代入を順番に計算(オーバラップ無し)
- pivot選択は行列の形状，サイズに無関係に必ず行う

DMA転送用サブルーチン 1/2

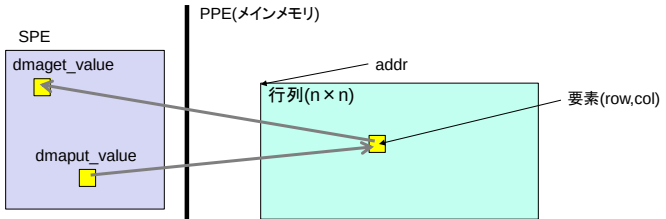
- 転送用の命令の準備
  - dmaget, dmaput : 資料等で使用されるDMA読み書き関数
  - void dmaget_burst(unsigned int ppe_addr, unsigned int spe_addr, unsigned int row, unsigned int col, unsigned int n)

メインメモリのアドレスppe_addrから始まる行列(各要素はfloat型)の，(row, col)の要素を含む128Byteを読み出し，LocalStoreのアドレスspe_addrから始まる要素に格納する．
(目的の要素は，*(float*)(spe_addr+row%32*sizeof(float))で取り出せる)



DMA転送用サブルーチン 2/2

- float dmaget_value(unsigned int addr, unsigned int row, unsigned int col, unsigned int n)
  メインメモリのアドレスaddrから開始される行列(要素はfloat型，サイズはn×n)の，(row, col)の要素を読み出して返す
- void dmaput_value(unsigned int addr, unsigned int row, unsigned int col, unsigned int n, float value)
  メインメモリのアドレスaddrから開始される行列(要素はfloat型，サイズはn×n)の，(row, col)の要素に，値valueを書き込む(SPE間で同期しないので注意)

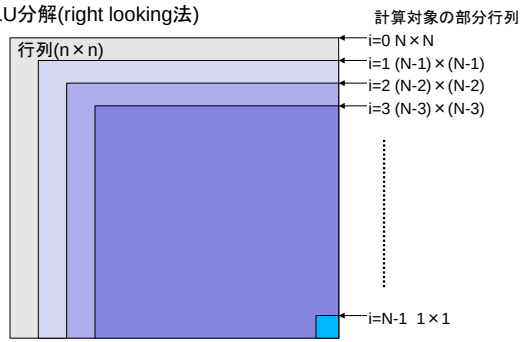


同期の方法

- 2種類の同期用サブルーチン
- SPE0を司令塔にして，SPE間DMA転送で同期を取る
- void sync_dist(unsigned int id, // SPEのID
  unsigned int* ppe_ls, // 各SPEのLocalStoreがマップされたアドレス配列
  volatile struct spe_sync* sd, // 同期用の構造体の配列
  unsigned int key) // 同期用キー
  SPE0から他のSPEのsdで指定された領域(変数start_flag)へ，特定の値keyを書き込む．他のSPEは，start_flagの値がkeyになるまで待つ(SPE0がそれ以外へ計算の開始を指示する)
- void sync_collect(unsigned int id,
  unsigned int* ppe_ls,
  volatile struct spe_sync* sd,
  unsigned int key)
  SPE1～6は，SPE0の自分のフラグ(変数sd[id].end_flag)へ値keyを書き込む．SPE0はその他のSPEに関するend_flagの値がkeyになるまで待つ(各SPEがSPE0へ計算の終了を指示する)

LU分解(LU decomposition)

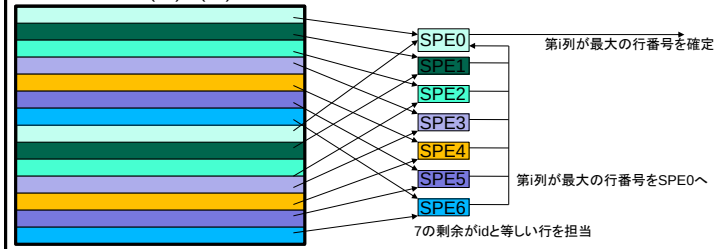
- 3つの部分をN回繰り返す(i=0～N-1)
  1. pivot 選択(最大の要素を持つ行の探索)
  2. 行の交換
  3. LU分解(right looking法)



1. 部分pivot選択

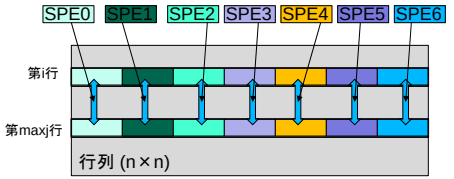
- pivot選択(pivoting関数)：第i列が最大値を持つ行を探索する
  - 各SPEは，(N-i)だけある行を7分割し，担当の行から，それぞれ第i列が最大値を持つ行を探索する
    - SPEは各行の第i列の値を読み出す(dmaget_valueを使う)
    - 最大値を持つ行番号maxjをSPE0に通知(sync_collectを使う)
  - SPE0が計算結果をとりまとめ，第i列の最大値を持つ行を選択する

計算対象：サイズ(n-i)×(n-i)の部分行列



2. 行(列)の交換

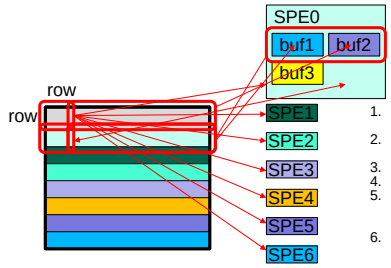
- 行の交換(swap_row関数)
  - 行列Aの第i行と第maxj行を交換する
  - 各SPEで32要素ずつ交換する(dmaget, dmaput関数)



- 列の交換(swap_col関数)
  - 行列bの第i列と第maxj列を交換する
  - 各SPEでMだけある列を分割して担当する

3. Right Looking法によるLU分解

- spe_lu_decomposition関数
  - 部分行列を行単位でSPEに処理を分割(部分pivot選択と同様の分割)
  - 各行の計算は以下の図の通り(簡易なアニメーション付き)



1. (row, row)の要素を変数diagに読み出す(dmaget_value)
2. row行の要素をrow+1列目からbuf2に読み込む(SPE0～6すべてで読み込む)
3. diagを(row, i)の要素で除算した結果t1を書き戻す
4. 第i行の要素をrow+1列目からbuf1に読み込む
5. buf1, buf2の各要素について，buf1の要素 - buf2の要素 ×t1を計算し，第i行目の該当部分に書き戻す
6. 行の末尾にくるまで2., 4. , 5. を繰り返す(繰り返す場合はbuf2の部分がbuf3になっているが，同じ処理を行っている)

- ここまでの1, 2, 3の処理をN回繰り返すことで，行列AがLU分解される

前進代入＆後退代入

- forward_substitution&backward_substitution関数
- 動作はソースコード参照
  - ver0.1とほぼ同じ簡易なアルゴリズム

- 解ベクトルごとにSPEを割り当てる
  - 解ベクトルの数が小さいと，何もしないSPEが出てくる場合がある
- 32要素ごとにまとめてDMA転送する

- 前進代入時の中間データ保存用に，メインメモリの「空き領域」を使用(N×M×sizeof(float)Byte)
- 後退代入後の結果をx (解ベクトル格納領域)へ書き込む

参考資料

- 奥村晴彦著「C言語による最新アルゴリズム辞典」技術評論社
- 小国力編著「行列計算ソフトウエアーWS、スーパーコン、並列計算機」丸善株式会
- 斉藤 宏樹，廣安 知之，三木 光範「 LU分解の並列化について」
  http://mikilab.doshisha.ac.jp/dia/research/report/2002/0612/018/report20020612018.html