立命館大学・21世紀COE
プログラム・シンポジウム

PCクラスタの構築と応用

2003年6月6日

立命館大学理工学部情報学科
山崎　勝弘

1

目　次

2

はじめに

研究背景

- 気象予測、環境問題、ゲノム解析、地球シミュレータ
- PCクラスタの普及
- 並列処理教育の必要性

研究目的

- 並列処理教育・研究用PCクラスタの構築
- OpenMP並列プログラミングによる問題解決
- 並列プログラムの蓄積と再利用

3

PCクラスタの構築

- PC16台＋ミリネット＋イーサネット
- クラスタシステムソフトウェアSCore
- 言語はPVMとOpenMP
- 並列処理教育・研究のプラットフォーム
- 2000年秋～2002年3月
- 2002年4月からOpenMP並列プログラミングが進行中

4

SCoreインストールの経緯

- Kondara 1.1
  - PCクラスタ8台
  - OpenMP

↓ SCore3.3.2

- Turbo 6.1
  - PCクラスタ8台
  - Easy Installation Tool
  - OpenMP

↓ SCore3.3.2

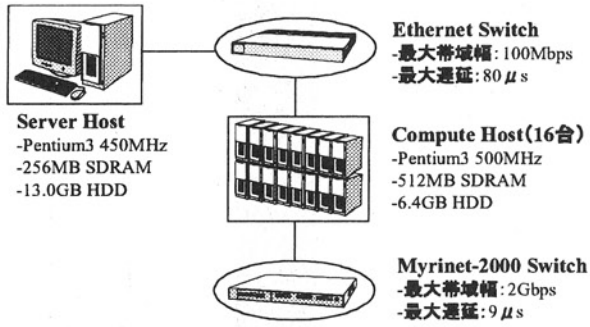
- Red Hat 7.1
  - PCクラスタ16台+サーバ1台
  - Easy Installation Tool
  - OpenMP, PVM

SCore4.2

- 現在　Red Hat 7.3+ SCore 5.2.0

5

PCクラスタの構成



6

構築の留意点

- Server Hostは制御用、Compute Host(16台)は計算用。
- スレッド間のデータ通信はMyrinetとEthernetどちらでもできる。制御はEthernet経由。
- MyrinetとEthernetが混在しているため、設定に注意する。

7

Hilditchの細線化アルゴリズム

- 条件1 図形画素である
- 条件2 境界点である
- 条件3 端点を削除しない
- 条件4 孤立点を保存する
- 条件5 連結性を保存する
- 条件6 線幅2の線分の片側だけを保存する
- 全ての条件を満たすとき、画素の階調値を変更する

P_4	P_3	P_2
P_5	P_0	P_1
P_6	P_7	P_8

8

並列化手法

- データ分割:単純なブロック分割では境界が正しく処理されない
  - 分割の境界で、線幅dだけ重複させる。
  - dは画像中の線幅の半分以上
  - 重複したブロック、サイクリック分割
- 並列アルゴリズム:プロセッサファーム
  - 各スレッドで独立に計算可能
- スケジューリング:静的、動的

9

実験

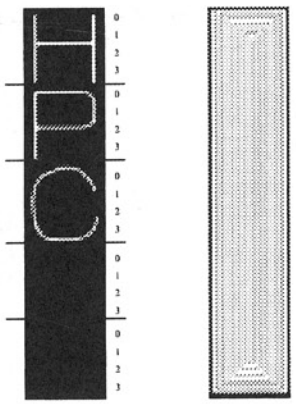
- 実験A: 縦方向だけの分割
  - ブロック分割、サイクリック分割
- 実験B: 縦・横 両方の分割
  - ランタイムスケジューリング
  - 実行前に環境変数で指定
  - 動的、静的ブロック分割、静的サイクリック分割
- 対象画像は均質と不均質2種類

10

実験Aのテスト画像

ピクセル数:1000×5000
線幅: 70
重複: 40ライン

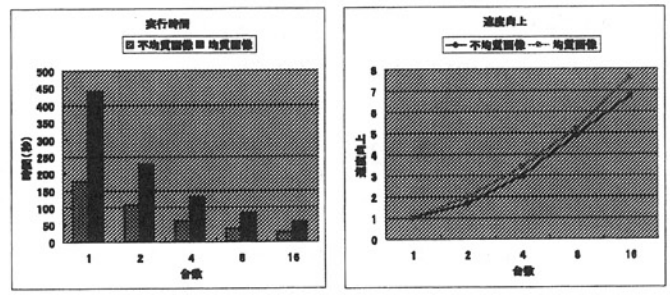
サイクリック分割の
サイクル数: 5



不均質画像　均質画像

11

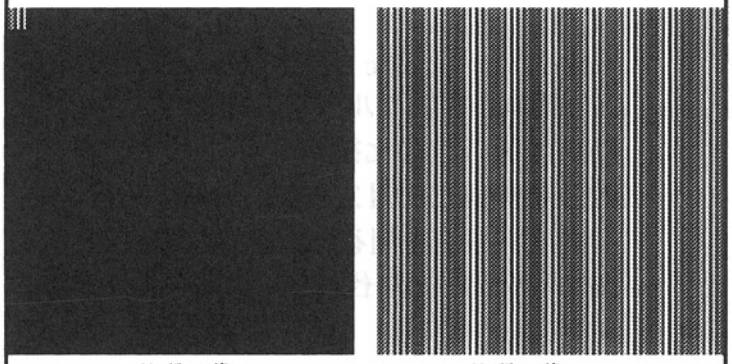
実験A(サイクリック分割)の実験結果



12

実験Bのテスト画像

ピクセル数:3000×3000　線幅:30　重複:16ライン
画像全体を200×200の225個のエレメントに分割

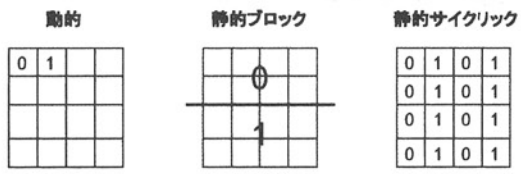


不均質画像　　均質画像

13

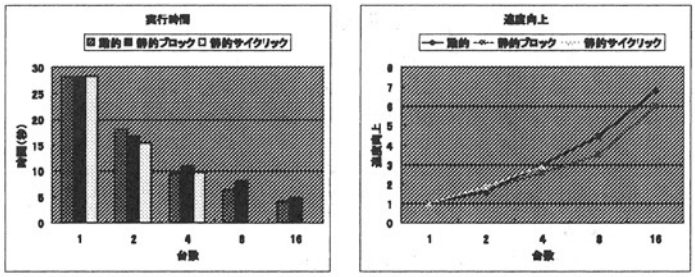
ランタイムスケジューリング

16エレメントを2プロセッサで実行の場合



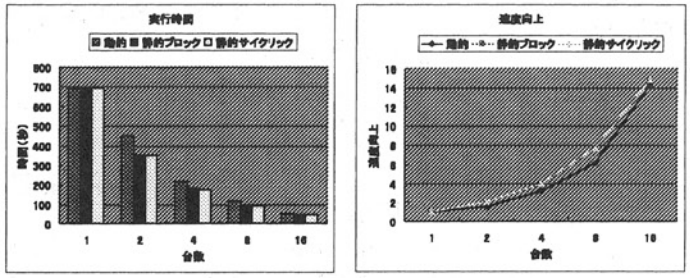
14

実験B(不均質画像)の実験結果



15

実験B(均質画像)の実験結果



16

考察

実験A

- 重複ラインの割合が大きく、性能が出ない
  - サイクリック分割、サイクル数5、16台で56%
  - 均質画像　16台で7．6倍

実験B

- 不均質画像では動的分割が良い。
  - しかし、16台で6．8倍程度
- 均質画像では静的サイクリック分割が良い。
  - 16台で14.8倍程度

17

ミリネット使用上の課題

- 計算ノードのPCIバスが32ビット/33MHzのため、ミリネットの性能が十分に出ていない。
  - 64ビット/66MHzのPCIバスを使用すべき
- メッセージの取りこぼしが起こる場合がある。
- 実験B　静的サイクリック分割8台以上で実行できないのはこの理由によると考えられる。

18

細線化のまとめ

- データ分割の境界で、線幅の半分だけ重複させ、プロセッサファームアルゴリズムで並列化を行った。
- 不均質画像では動的分割、均質画像では静的サイクリック分割が良い。

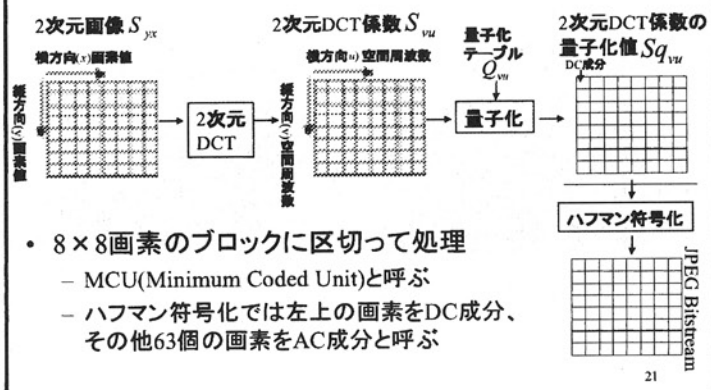
19

JPEGとは

- "Joint Photographic Experts Group"の頭文字　標準化検討グループの名称
- 静止画像の符号化技術
- 画像品質劣化が目立たない場所で圧縮
- 高周波数成分を削るため自然画像向き
- 1986年開始　90年代前半　実際の標準化

20

JPEGエンコーダのアルゴリズム



- 8×8画素のブロックに区切って処理
  - MCU(Minimum Coded Unit)と呼ぶ
  - ハフマン符号化では左上の画素をDC成分、その他63個の画素をAC成分と呼ぶ

21

DCTと量子化

- DCT(離散コサイン変換)
  - 高周波数成分を削る処理

$$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} S_{yx} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

$$C_u, C_v = \begin{cases} 1/\sqrt{2} & for \quad u, v = 0 \\ 1 & otherwise \end{cases}$$

- 量子化
  - 四捨五入して指定ビット数に制限

$$Sq_{vu} = round\left(\frac{S_{vu}}{Q_{vu}}\right)$$

22

ハフマン符号化

- ジグザグスキャン：2次元配列を1次元へ並べ替え

0→1	5→8	14→15	27→28				
2	4	7	13	16	26	29	42
3	6	12	17	25	30	41	43
8	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35→36	48	49	57	58	62	63	

- DC成分の符号化：カテゴリ値とハフマン符号を決め、付加ビットをつける

入力値	カテゴリ値	ハフマン符号	付加ビット
-7…-4,4…7	3	100	000…011,100,111
-3,-2,2,3	2	11	00,01,10,11
-1,1	1	10	0,1
0	0	0	なし

23

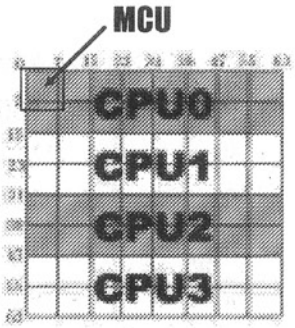
- AC成分の符号化
  - AC成分のほとんどは0！
  - 非0成分の前までの0の個数をRRRR
  - 非0成分の値をSSSS
  - RRRRとSSSSの組合せにより符号化(下表)
    - 0が16個以上の場合はZRL(Zero Run Length)を出力
    - 最後のAC成分が0の場合はEOB(End Of Block)を出力

RRRR/SSSS	0	1	2	…	10
0	1010(EOB)	0	1	…	1111111110000010
1		1100	11011	…	1111111110001000
2		11100	11111001	…	1111111110001110
…		…	…	…	…
15	11111111001(ZRL)	1111111111110100	1111111111110110	…	1111111111111110

24

並列化手法

- 画像全体をブロック分割
- DCT、量子化：MCU単位で独立に計算可能
  - ブロック毎の並列処理
- ハフマン符号化
  - DC成分は直前MCUのDC成分に依存：逐次処理
  - AC成分はブロック毎の並列処理



画像分割方法

25

入力ファイル

- 480×360画素と1024×768画素と1600×1200画素の3種類
- Bitmap形式

26

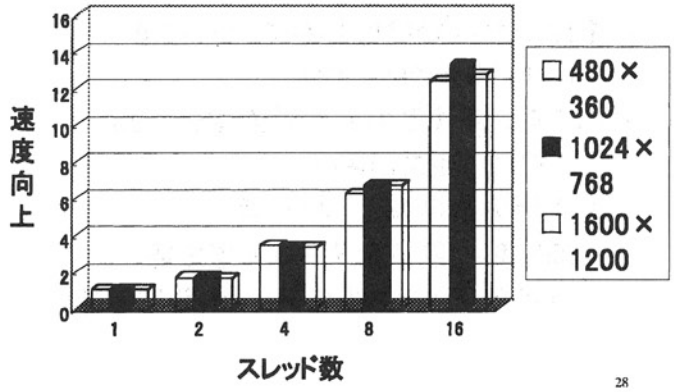
実行時間

単位：秒

スレッド数	1	2	4	8	16
DCT変換					
480×360	7.81	4.60	2.30	1.24	0.63
1024×768	35.60	21.14	10.58	5.29	2.68
1600×1200	86.24	51.31	25.65	12.83	6.76
量子化					
480×360	85ms	51ms	26ms	23ms	18ms
1024×768	0.38	0.23	0.11	0.06	0.03
1600×1200	0.95	0.56	0.27	0.14	0.14
ハフマン符号化					
480×360	0.21	0.16	0.08	0.05	0.03
1024×768	0.95	0.71	0.38	0.23	0.16
1600×1200	2.34	1.71	0.89	0.51	0.32
処理全体					
480×360	8.18	4.89	2.48	1.38	0.75
1024×768	37.25	22.40	11.37	5.87	3.14
1600×1200	90.43	54.42	27.57	14.18	7.90

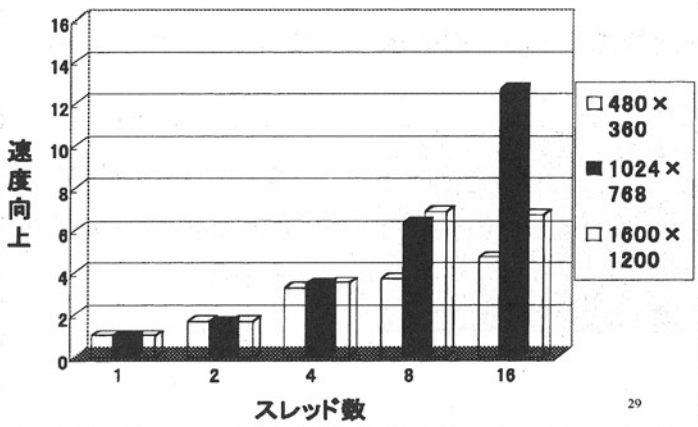
27

DCTの速度向上



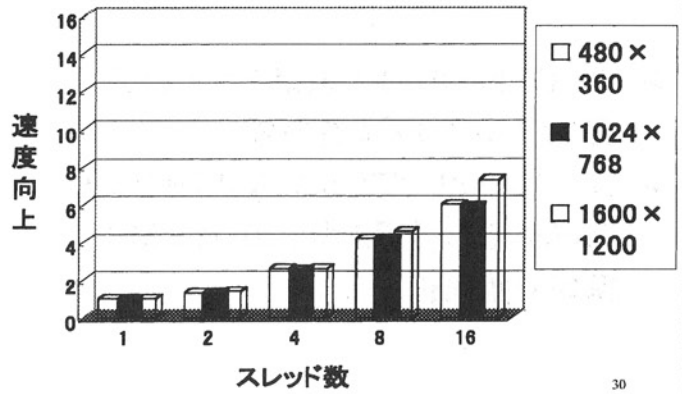
28

量子化の速度向上



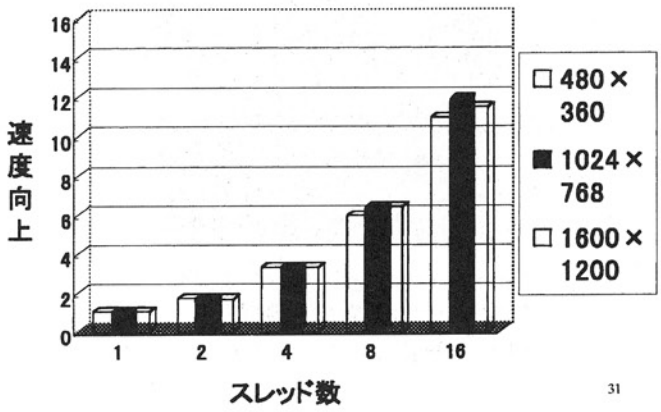
29

ハフマン符号化の速度向上



30

処理全体の速度向上



31

所要メモリ量

- 入力ファイル　5.76MB
- RGBファイル　1600×1200×4×3　23MB
- S_{yx}　23MB
- S_{vu}　23MB
- Sq_{vu}　S_{vu} と共通
- 量子化テーブル　8×8×4×2　512B ×スレッド数

32

考察

- 全体の並列効果は16台で11～12倍
  - DCTが全体の85～95%
- ハフマン符号化の並列効果は6～7倍
  - メモリアクセスが多い
- 量子化の速度向上　検討要
  - 画素毎に量子化テーブルを参照
  - 量子化テーブルを各スレッド毎に保持

33

JPEG2000とは

✓ 静止画像データ圧縮の新国際規格

✓ ISO/IEC 15444 (ITU-T Rec. T.800)

✧ JPEG2000の特長

➢ 高圧縮率（JPEGに対してファイルサイズで30~50%）

➢ 高画質　（ブロック・モスキートノイズが発生しない）

➢ 統一的なアルゴリズムでの可逆/不可逆圧縮

➢ アニメーション、著作権保護などの機能

× ただし、圧縮/伸張の処理負荷が大きい
→ 並列化による高速処理！

34

実装方法

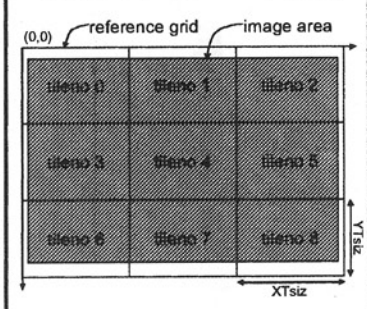
➢ JPEG2000 Part1で規定されている圧縮アルゴリズムを対象

➢ JPEG2000のリファレンスソフトウェアJasPerをPCクラスタ上のOpenMP環境で並列化

35

JPEG2000のアルゴリズム（Ｉ）

➢ 画像のタイルへの分割



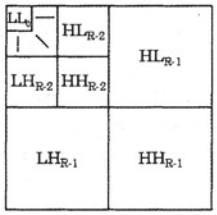
➢ 各タイルのエンコード

1. DCレベルシフト
2. コンポーネント間変換
3. 離散ウェーブレット変換
4. 量子化
5. エブコット

36

JPEG2000のアルゴリズム(Ⅱ)

1. DCレベルシフト ..... DLS
   - 各画素値を 0～255 から -128～127の範囲へシフトする
2. コンポーネント間変換 ..... MCT
   - RGB 信号を YCbCr 信号へ変換する
3. 離散ウェーブレット変換 ..... DWT
   - 5/3フィルタ(可逆)、9/7フィルタ(不可逆)を適用し、YCbCr信号を周波数成分に変換する
   - 変換は、ひとつ前の水平ローパス垂直ローパス(LL)サブバンドに対して繰り返し行われる



JPEG2000のアルゴリズム(Ⅲ)

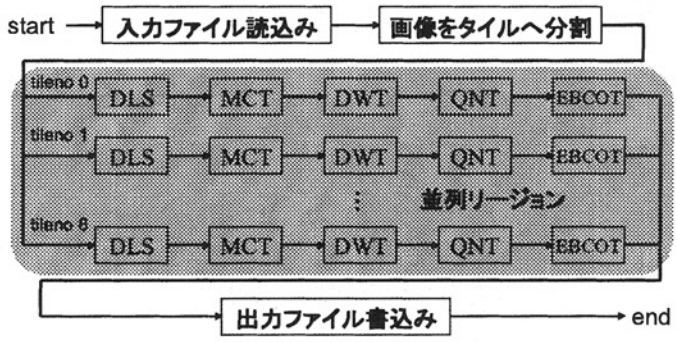
4. 量子化 ..... QNT
   - 各サブバンドを量子化ステップサイズで割り、ダイナミックレンジの削減を行う
   - 可逆圧縮の場合、量子化処理はバイパスされる
5. エブコット ..... EBCOT
   - 各サブバンドをさらにコードブロックに分割する
   - コードブロックはビットプレーンとして扱われる
   - MSBからLSBへビットプレーンを処理する
   - 適応二進算術符号化によりエントロピー符号化を行う

OpenMPによる並列化方法

各タイルのエンコードをスレッドへ分担させる

```
画像をタイルへ分割する
#pragma omp parallel for        ← 並列化指示文
for (tileno = 0; tileno < numtiles; tileno++) {
  タイル番号 tileno のタイルをエンコードする
}
```

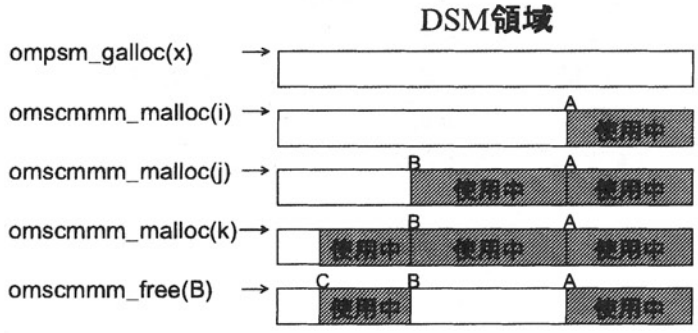

並列化されたJPEG2000エンコードの流れ



ソフトウェア分散共有メモリ(DSM)の動的確保と解放(Ⅰ)

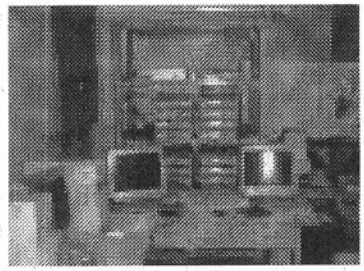
- Omni-SCASH では DSM を動的に確保する関数 ompsm_galloc() が用意されている
- ただし、ompsm_galloc() に対する free() が用意されていない
- ompsm_galloc() で確保したDSM領域に対する omscmmm_malloc(), omscmmm_free() を作成

ソフトウェア分散共有メモリ(DSM)の動的確保と解放(Ⅱ)



実験

- 入力画像
  1960×1420 pixels
  (RGB, 24bit/pixel)
- 圧縮モード
  可逆(lossless)
- タイルサイズ
  50×50, 100×100, … ,
  250×250
- 実行時間測定範囲にファイル入出力は含まない



43

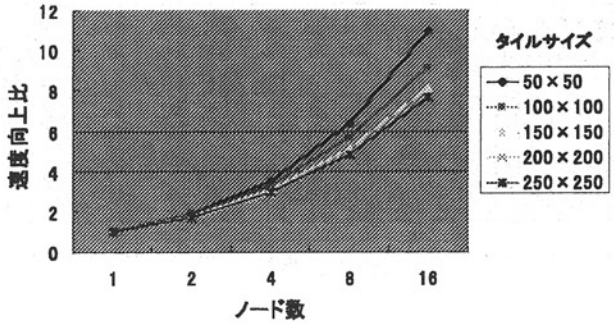
タイルサイズによる実行時間と速度向上

タイルサイズ		ノード数				
タイル数		1	2	4	8	16
50×50	実行時間[sec]	47.28	25.02	13.43	7.42	4.33
1200	速度向上	1.00	1.89	3.53	6.39	10.95
100×100	実行時間[sec]	29.08	15.76	8.83	5.13	6.39
300	速度向上	1.00	1.85	3.30	5.66	9.13
150×150	実行時間[sec]	24.53	13.75	7.96	4.78	3.02
140	速度向上	1.00	1.78	3.08	5.13	8.14
200×200	実行時間[sec]	23.07	13.98	8.14	4.71	2.99
80	速度向上	1.00	1.65	2.83	4.90	7.71
250×250	実行時間[sec]	22.50	13.06	7.66	4.65	2.92
48	速度向上	1.00	1.72	2.94	4.84	7.70

44

実行結果

タイルサイズ　: 50×50
実行時間　: 1ノード47秒 → 16ノード4秒
ファイルサイズ　: 8.2MB → 1.7MB (圧縮率20%)



45

考察

- タイルサイズの大小
  - 実行速度と圧縮率　vs　速度向上とエラー回復性
- タイルサイズ 50×50 の速度向上が最善
  - 16ノードで約11倍の速度向上
- ソフトウェア分散共有メモリのオーバーヘッド
  - メモリ配置の工夫：共有変数と局所変数
- 各タイルの処理負荷の大きさに違いがある
  - 動的負荷分散による高速化

46

立命館大情報学科での並列処理教育

- 2000年度～、3回生前期、130名程度
- 講義の位置づけ

2後期 計算機構成論I → 3前期 計算機構成論II → 4後期 並列コンピューティング
2後期 計算機構成論I → 特殊講義 → 4後期 並列コンピューティング

47

特殊講義の内容

- PVM並列プログラミング演習
- Bit別冊：はじめての並列プログラミング
- 講義9回：イントロ、並列処理全体像、並列マシン、並列プログラミング言語、並列アルゴリズム、スレッド、PVMとMPI
- 演習5回：PVM使用法、例題実行、自作並列プログラミング
- PC65台、100Mイーサ、Linux, PVM　2室

48

教育結果と課題

- 優秀プログラム:画像処理(ディザ画像、エッジ検出、濃度ヒストグラム)、ソートなど
- 並列効果の測定と考察

課題

- 並列処理の有用性の理解→ビデオ
- 学生は課題設定で悩む→例題追加
- TA人数: 130名にTA2名→TA4名

49

今後の研究方向

- PCクラスタによる大規模問題の解決
  - JPEG/JPEG2000デコーダ
  - 流体問題
  - N-body 問題
- 応用向きプロセッサの研究
  - 応用、アルゴリズムからアーキテクチャへのフィードバック
- 大規模システムにおける負荷分散法の研究
  - グリッドコンピューティング

50

まとめ

- PCクラスタ構築の経緯
- 細線化、JPEG/JPEG2000エンコーダの並列化
- PVM並列プログラミング教育

今後の課題

- 並列処理研究の活性化
- 並列処理研究者と応用研究者の協力

51