

並列計算入門

降旗 大介

大阪大学サイバーメディアセンター

2014.11.11

はじめに

この講習の対象者

- 普段は Windows, Mac を使っていて, Unix についてはあまり…
- 研究対象についてはよく知っている.
- 普通にプログラミングは出来る.
- 計算対象がやや大規模になりそうだ.
- 少しでも計算が速いと有難い.
- 並列計算に興味はあるが, まず全体像がよくわからない. どのような技術があるのか? 難しいのか簡単なのか? 高価につくのか?

0437-J

Part I

Unix 入門

- GUI と CUI: Unix, Windows, Mac OS の同じところと違うところ
- ごく簡単な入門

Part II

並列計算入門

- 原理的に並列計算でどれくらい高速化が可能か
- そもそも高速化の手法には何があるのか
- 各種並列計算の紹介

詳しく知りたくなったら？

ぜひ、それぞれのソフトウェアの講習会 (下記: 2014 年度後期) へ！

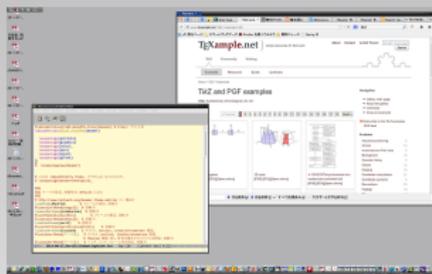
- 11 月 11 日 並列計算入門 (本日)
- 12 月～2 月 スーパーコンピュータ概要とスーパーコンピュータ利用入門
- 12 月～2 月 スーパーコンピュータと並列コンピュータの高速化技法の基礎
- 12 月～2 月 MPI プログラミング入門
- 12 月～2 月 HPF プログラミング入門

0437-J

Part I: UNIX 入門

コマンドで操作する CUI とは

GUI と CUI, OS の関係



GUI

```
-b Forces a "break" from option processing, causing any further
shell arguments to be treated as non-option arguments. The remain-
ing arguments will not be interpreted as shell options. This may
be used to pass options to a shell script without confusion or pos-
sible subterfuge. The shell will not run a set-user ID script
without this option.

-c Commands are read from the following argument (which must be
present, and must be a single argument), stored in the command
shell variable for reference, and executed. Any remaining argu-
ments are placed in the argv shell variable.

-d The shell loads the directory stack from /.cshdirs as described
under Startup and shutdown, whether or not it is a login shell. (+)

-Dname[=value]
  Sets the environment variable name to value. (Domain/OS only) (+)

-e The shell exits if any invoked command terminates abnormally or
yields a non-zero exit status.

omdsrv@paon> <102>
```

CUI

- 今の OS (Windows, MacOS X, Unix) の GUI (Graphical User Interface) の見かけはあまり変わらない。
- CUI (Character User Interface)/ CLI (Command Line Interface) の充実度は OS によって異なる。
- Unix は CUI/CLI が非常に充実、かつ、Unix の強みはそこに。
- Unix がよくわからない = CUI/CLI がよくわからない。

GUI

- 俯瞰的. 同時並行表示. 情報量多し.
- 直感的に使える.
- 環境・状況依存性高し.
- 自動化しにくい.
- ソフトウェア間の連携しにくい.

CUI

- 局所的. 表示情報は原則 1 度に 1 つ. 情報はミニマム.
- 理論的な操作.
- 環境・状況依存性低し.
- 自動化しやすい.
- ソフトウェア間の連携しやすい.

CUI を理解するコツ I

GUIは「地図」で、CUIは「写真」である！



GUI = 地図



CUI = 写真

CUIでは「今何処にいるか」が重要。
見たいもの(ファイル等)があったらそこまで移動しないとイケない。

CUI を理解するコツ II

CUI の操作 = 執事 (shell) への命令 である。



ユーザー



命令



シェル

CUI 操作 = シェルとよばれるソフトウェアへの命令。
付き合いにくいシェルもあるので、好みで変えよう!

CUI は遠隔操作でよく使われる



手元の端末



ネットワーク



Unix サーバ

遠隔操作はネットワークに負荷がかかるので、普通は CUI で。
通常は ssh というプロトコルが使われる。

0438-J

CUI で使われるエディタは事実上 Emacs か vi に限られている

```
Emacs 22.1.1 (GNU Emacs)
Welcome to GNU Emacs, a half of the GNU operating system.

Get help      C-h (Hold down CTRL, and press h)
Emacs manual  C-h r (Browse manual) C-h i
Emacs tutorial C-h t (Undo changes) C-h u
Buy manuals   C-h c (Exit Emacs) C-h C-c
Activate menu  F10 or ESC, or M-
[?C means use the CTRL key, M- means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the character.)

Useful tags:
Visit New File      Open Remote Directories
Customize Startup   Open *customize* buffer

GNU Emacs 22.1.1 (GNU Emacs) - Free Software Foundation, Inc.
Copyright (C) 2008 Free Software Foundation, Inc.

GNU Emacs comes with ABSOLUTELY NO WARRANTY; you can redistribute copies
of Emacs and modify it. See C-h C-c to see the conditions.
Type C-h C-d for information on getting the latest version.

If an Emacs session crashed recently, type Meta-x recover-session RET
to recover the files you were editing.

GNU Emacs 22.1.1 (GNU Emacs) - Free Software Foundation, Inc.
For information about GNU Emacs and the GNU system, type C-h C-c.
```

Emacs

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

vi

普通は (まだ) とっつきやすい emacs がお勧め。
管理人は vi が使えないと困るよ。

Unix コマンド入門
これだけ知っていれば戦える

- `pwd` 今何処のディレクトリに居るかを表示.
- `cd ..` 上階層ディレクトリへ移動.
- `cd hoge` hoge ディレクトリへ移動.
- `mkdir hoge` ここに hoge ディレクトリを作る.
- `rmdir hoge` この hoge ディレクトリを削除.
- `mv hoge poku` この hoge ディレクトリを poku ディレクトリに移動 or 名前変更.
 - poku ディレクトリが既に有るならその中へ移動,
 - 無いならその名前に変更.

- `ls` 今のディレクトリに有るファイルのリスト表示.
- `touch hoge` hoge というファイルを作る.
- `rm hoge` hoge というファイルを削除.
- `mv hoge poku` hoge というファイルを poku ディレクトリに移動
or 名前変更.
 - poku ディレクトリが既に有るならその中へ移動,
 - 無いならその名前に変更.

Unix コマンド: ファイル中身 操作

- `less hoge` `hoge` というファイルの中身を表示.
ほぼ同様の動作をするコマンド: `more`, `cat`
- `grep kore *` このディレクトリで `kore` という文字列を含むファイルを表示.

0438-J

- `emacs hoge` hoge というファイルを読み込んで起動.

(以下, emacs 中で.

C- は Ctrl キー同時押し, M- は Esc キーを押してから.)

- C-x C-s 保存.
- C-x C-c 終了.
- C-g emacs がしていることを止める. 困ったらこれ.
- C-s hoge hoge という文字列を探す.
- C-スペースキー 選択開始.
- M-w コピー.
- C-w カット (削除).
- C-y ペースト (貼付け).

Unix エディタ (ファイル編集) : vi

- vi hoge hoge というファイルを読み込んで起動.

(以下, vi 中.

文字挿入モードと, コマンドモードを切り替えて使う.)

- i 文字挿入モードへ切替え.
- Esc キー コマンドモードへ切替え.

(以下, コマンドモードで.)

- h, j, k, l 左, 下, 上, 右へ移動.
- :wq 保存して終了.
- :q! 保存せず終了.
- x, dd 1文字, 1行カット (削除).
- yy 1行コピー.
- p ペースト (貼付け).

- Unix がわからない = CUI に慣れてないというだけのことが多い。
- CUI を理解するにはいくつかコツが有る。
 - CUI はその性質が「写真」によく似ている。
 - CUI の操作 = 執事 (shell) への命令。
 - CUI は遠隔操作でよく使われる。
 - CUI で使われるエディタは事実上 Emacs か vi に限られている。
- コマンドは沢山あるが、今回紹介したものがわかれば充分戦える。
- 薄いものでいいので Unix の本を買って持っておこう。

Part II: 並列計算入門

この Part の構成

話すこと

- スーパーコンピュータの歴史
- 高速化の仕組み, 手法
- 原理的に, 並列計算でどれくらい速くなるのか?
- どんなハードウェアとソフトウェアの組み合わせが?

話さないこと

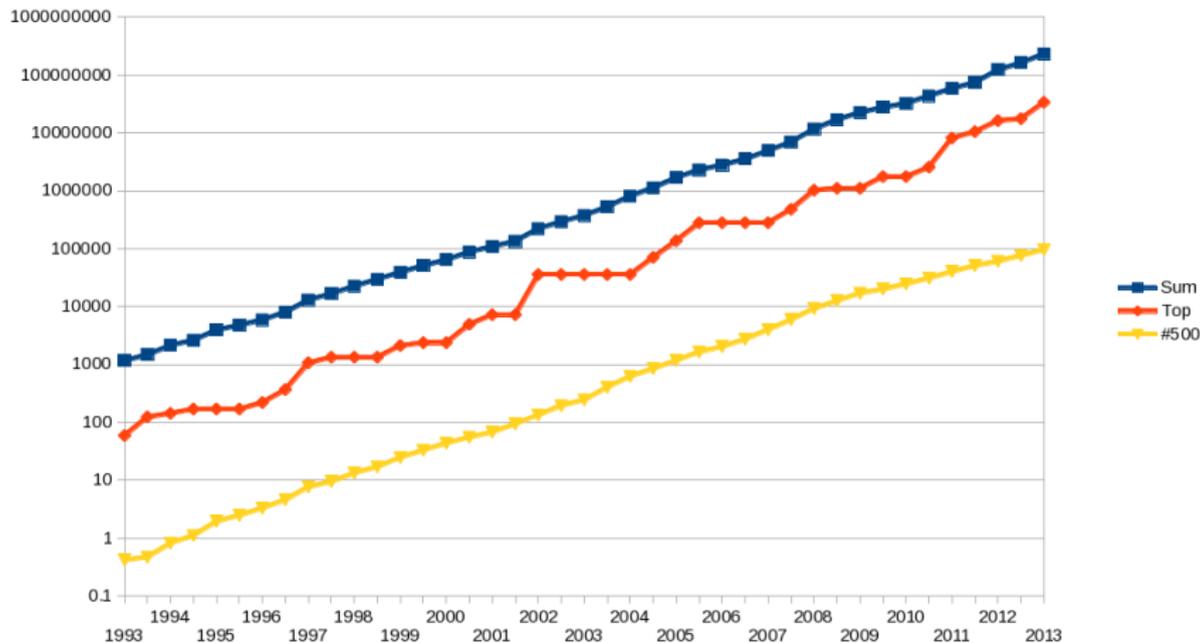
- 各種ソフトウェアのプログラミングの詳細
→ **それぞれの講習会へ GO!**
- 並列計算のチューニング等

スーパーコンピュータについて

スーパーコンピュータとは

- 科学技術計算を主目的とする大規模コンピュータのこと.
- **大規模, 超並列処理**
 - ベクトル演算 (SIMD): 1 クロックで数百の演算を同時実行
 - SX-9 (NEC) は 1 チップ 1 コアプロセッサで 100GFlops 以上 (2007 年当時世界最高速).
 - 数百万規模のプロセッサコア
 - 京 (Top500 1 位: 2011.06-12): コア数 705,024.
 - 天河 2 号 (中国, Top500 1 位: 2013.06-): コア数 3,120,000
- **巨大メモリ**
 - 京: 1.26 P バイト, PrimeHPC FX10(京の商用版) は spec 上 6Pbyte 積める
- **大電力** (… 困るが)
 - 京は 12,659KW, 天河 2 号は 17,808 KW. 淀川水系にある 30 の水力発電所のうちでも 10,000 KW を越えるものはたった 3 つ.

スーパーコンピュータの歴史 I



スーパーコンピュータの性能の変遷

(creative commons -attribution, share alike 3.0 by A.I.Graphic)

スーパーコンピュータの歴史 II

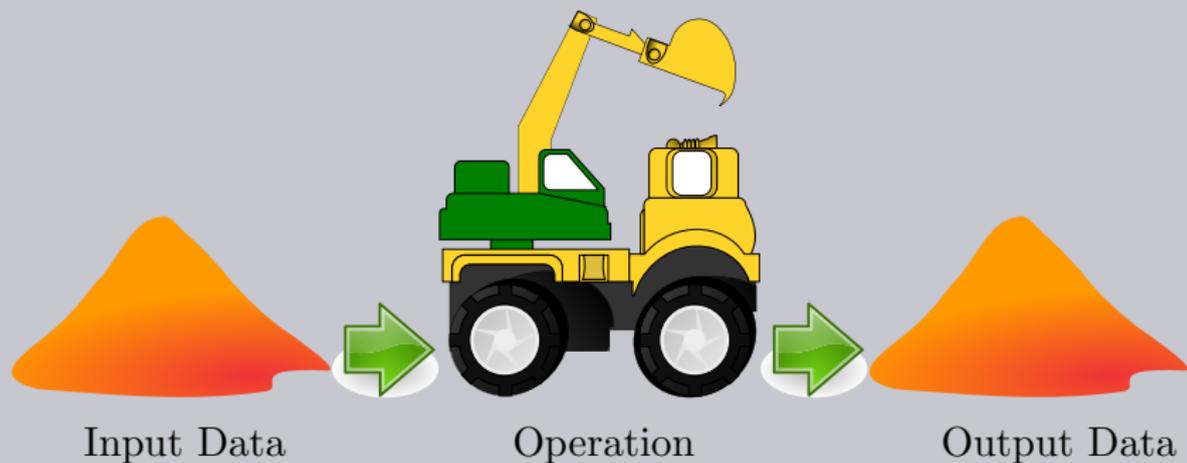
- ILLIAC I, II, III (真空管 1952, トランジスタ 1962, SIMD 1966)
スパコン黎明期. 並列計算の概念, 技術の基礎が登場.
- Cray-1 (商用スパコン, 1976, 80-160MFLOPS).
計 80 台以上が飛ぶように売れた.
- 地球シミュレータ (SX-5, 41 TFLOPS, 2002: SX-9, 131 TFLOPS, 2009)
TOP500 1 位 2002.06-2004.06. 当時としては「化け物」. アメリカでは Japanese 'Computenik' 扱い.
- Blue Gene (2004)
シンプルにコア数を増やす戦略. 登場時で既に 32,768 コア, 2007 年には 212,992 コア.
- 京 (2011)
10.62 PFLOPS の超並列機.
- 天河 2 号 (2013)
intel CPU をとにかかく数多く繋ぐ. 33.8 PFLOPS.

トップ性能のスーパーコンピュータに絡んでいる国は次の3つ.

- 中国： 速い商用 CPU を数多く使う戦略.
- アメリカ： CPU 数を多くというシンプルな戦略は同じ.
Cray 社と IBM 社が強い.
CPU を自国で開発できる強みあり.
- 日本： 今や世界で唯一のベクトル型スーパーコンピュータを作る国.
NEC と富士通, 日立が主なメーカー.
CPU は自国開発ものと輸入物の混在.

計算の高速化手法, 概要とハードウェア

そもそも計算とは



アルゴリズムそのものの改善

(理想的) 「より速い」 or 「並列化により適している」アルゴリズムへ



例： $1 + 2 + 3 + \dots + 100 = 5050$ に対して、

```
for i := 1 to 100 do result += i;
```

を改善して、次のように。

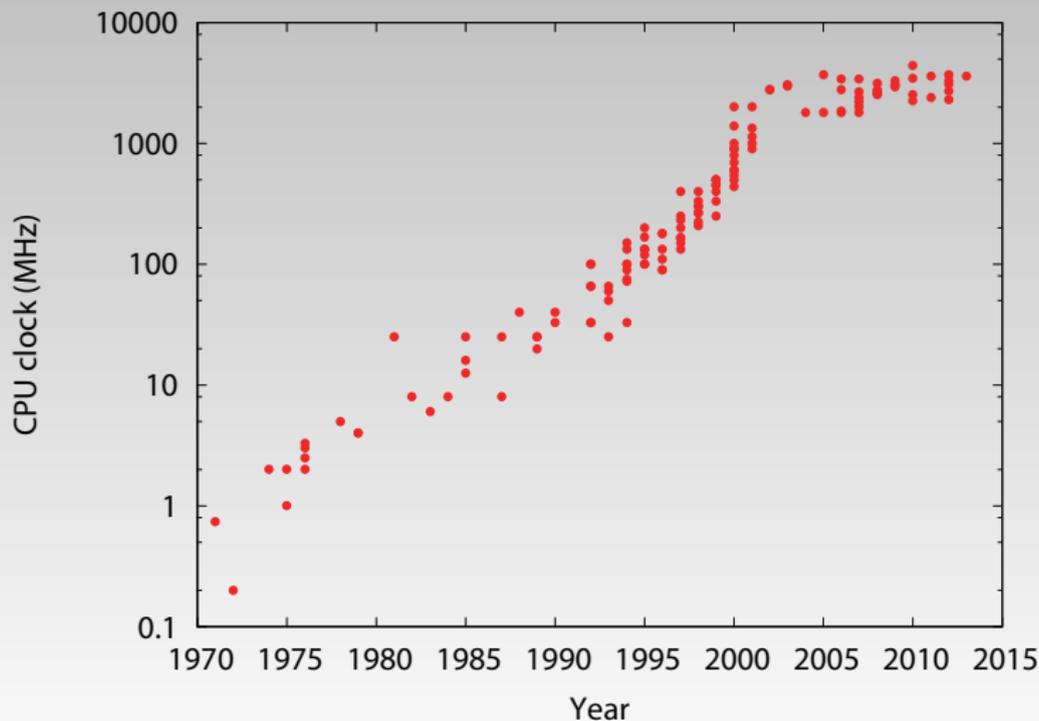
$$(100 + 1) * 100 / 2$$

速く回せ！

CPU, メモリ, HDD などのスピードを単純に上げる.



計算の高速化手法 III-2



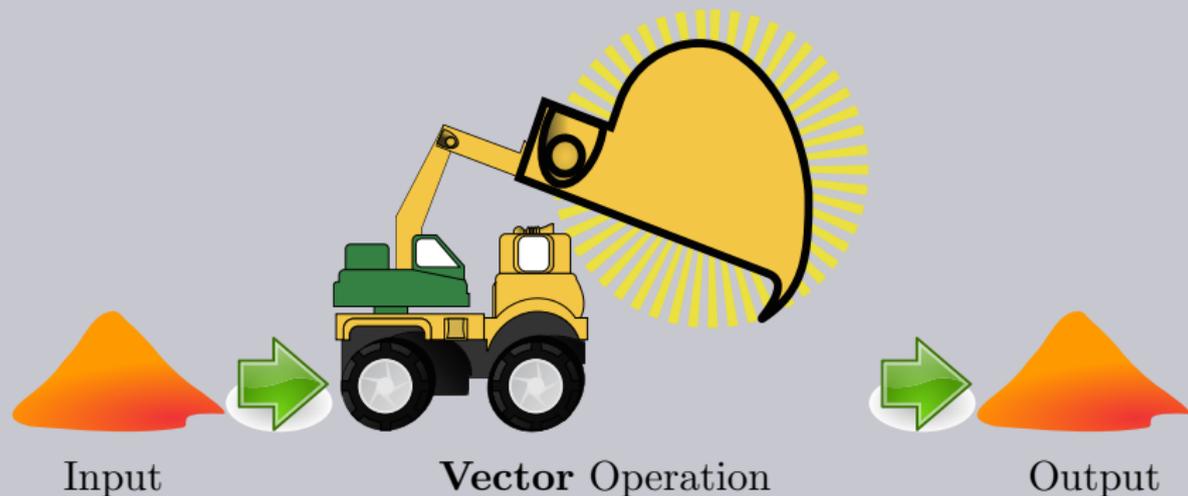
単純なスピードアップはそろそろ限界？

計算の高速化手法 IV-1

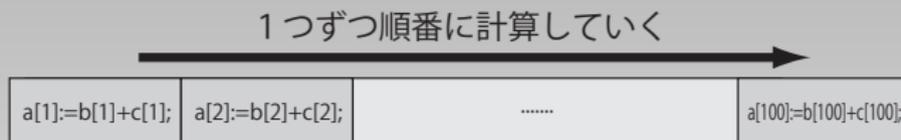
一度に沢山行え！

似たような処理をまとめて一度に行う。

- ベクトル計算
- SIMD (Single Instruction Multiple Data)



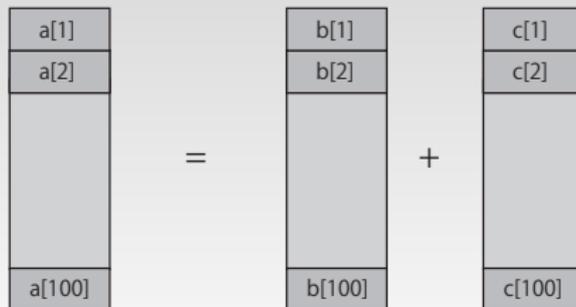
計算の高速化手法 IV-2



スカラー計算



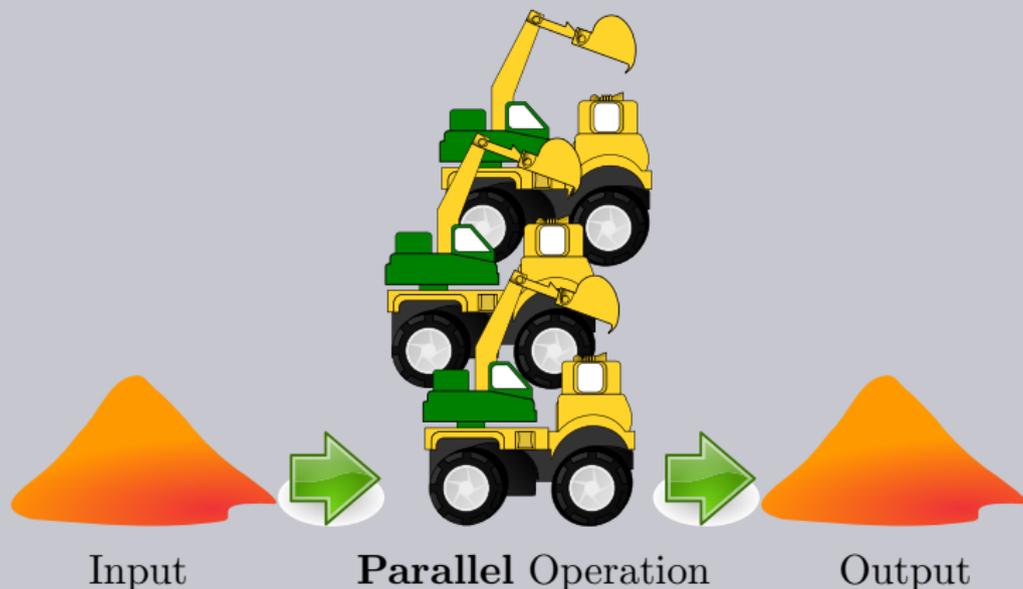
一度に全部計算できる



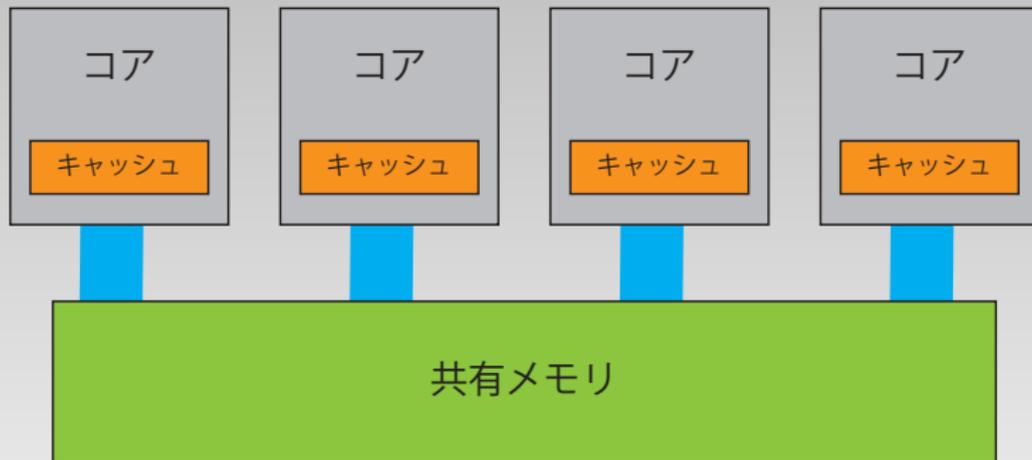
ベクトル計算

皆でやれ！ メモリ共有型

メモリ (データ) を共有した状態で，処理を並列化．



実装しやすいが，メモリアクセス競合が起きやすく，高速化しにくい．



メモリ共有型計算機概念図

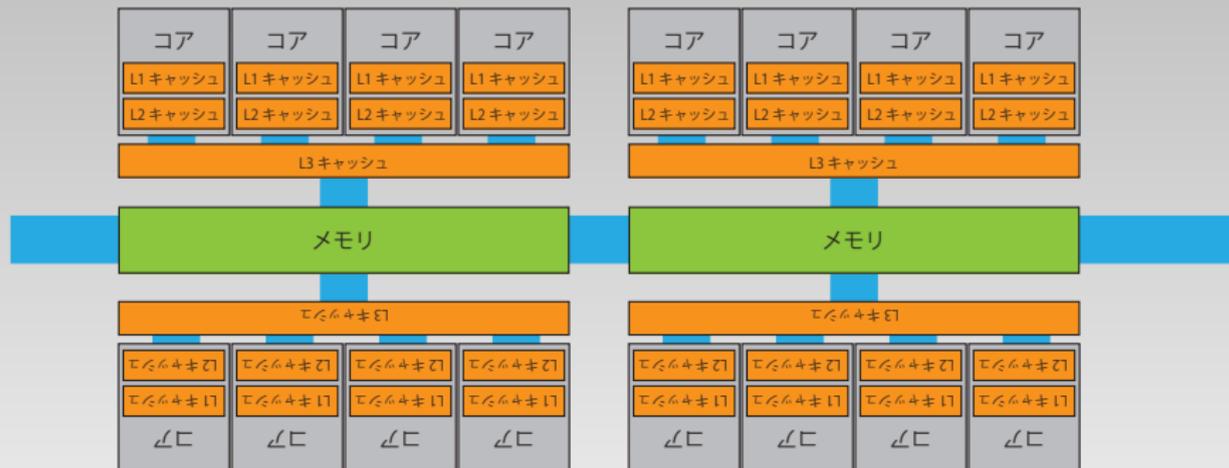
皆でやれ！ メモリ分散型

メモリ (データ) を分散して, 処理を並列化.



メモリアクセス競合が起きにくく, うまくいけば高速化しやすい.

計算の高速化手法 VI-2



メモリ分散型計算機 の概念図

(ついでに) 平行化

プログラムを、「どういう順序で実行しても良いように分割する」ことを平行化などと言う.

- これができれば，並列化にとっても役に立つ.
- 難しい.

0434-J

現状は

- 「単純に速く」というのは段々困難に.
- ベクトル型スーパーコンピュータは今や NEC の SX シリーズぐらいか.
PC のチップや, GPU 計算は SIMD をサポート.
- 多くのスーパーコンピュータは超並列 (つなぎ方がすごい!).
メモリは, 多段分散型, つまり, 分散しているが近くとは共有のキャッシュがあるような多段構造が多い.

0434-J

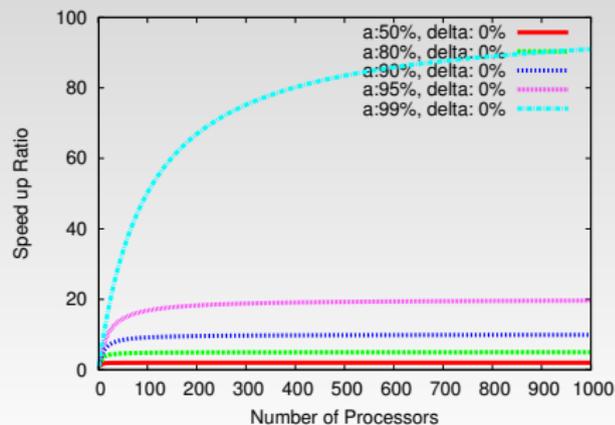
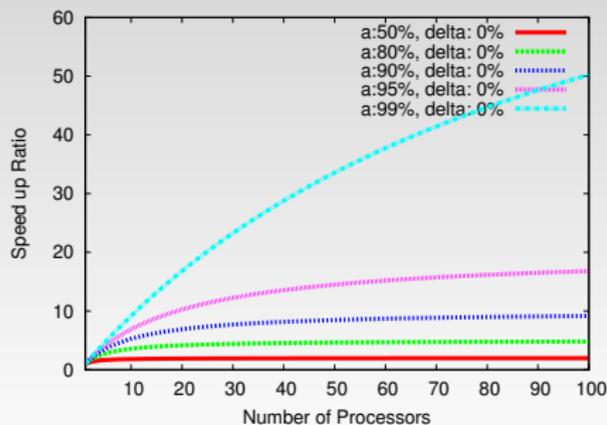
並列計算の効率**限界**

並列計算でどれくらい速くなるのか: アムダール則 I

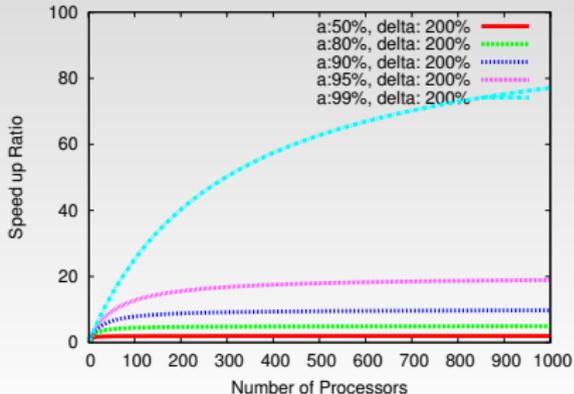
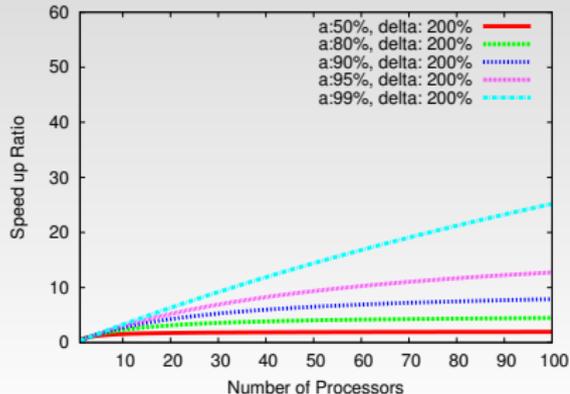
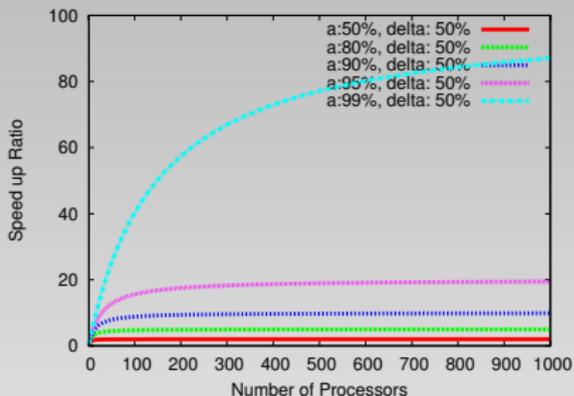
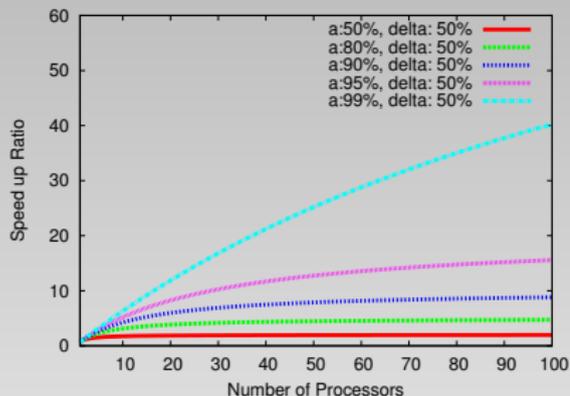
プログラム中, 割合 a の部分を n 個のプロセッサで並列化し, 残り $1 - a$ の部分を並列化しない場合, 全体は

$$\frac{1}{(1 + \delta)\frac{a}{n} + (1 - a)} \text{ 倍}$$

に高速化される. δ は並列化に伴って発生する通信等による遅延率.



並列計算でどれくらい速くなるのか II



結局…

- 並列化できない箇所が「信じられないくらい足を引っ張る」.
- 並列化に伴う通信等で遅延があると、全体をじわじわと遅くする.
- ただ並列化するだけでは効率は悪いかも…

0434-J

どのようなソフトウェアを使うべきか

ハードウェアとソフトウェアの組み合わせ

ハードウェア的な結合度の緩い方から並べると…

結合方法	メモリ	ソフトウェア等
マシン間並列	分散	MPI (Message Passing Interface)
CPU 間並列	共有	OpenMP (Multi Processing)
CPU 内並列	共有	OpenMP, 他沢山
SIMD	共有	各種ライブラリ, CUDA 等

一般的に、下のほうがプログラミングは容易。
ハードウェアは上のほうがスケールしやすい。

並列計算ソフトウェア I

小規模もしくは、使いやすい方から紹介する。

ベクトル化, SIMD

- ハードウェア, ソフトウェア, ライブラリの「準備」をしさえすれば…
- プログラミング的な意味での特殊なテクニックはほぼ不要。
「ここはベクトル化する, しない」という強制的なディレクティブ (必須ではない) を入れたりするぐらい。
- アムダールの法則があるので,
ベクトル化率を高めるための変更は必要かつ重要。

例えば,

```
for i:=1 to 10000 do a[i] := 2*i;
```

というコードはコンパイルするだけで1クロックで実行される。

CPU 内 / 間 並列化: メモリ共有

- プログラミングは比較的容易.
「ここからここまで並列」というディレクティブをプログラムに書き入れるぐらいで済む.
- 移植性は高い.
- 解説書多し.
- やや高速化しにくい.
- ソフトウェアとしては
 - OpenMP
 - OS や言語の用意するライブラリ
 - Grand Central Dispatch (MacOS X 10.6, FreeBSD),
 - intel TBB など.

OpenMP

Fortran の例:

```
program hello
  :
  !$omp parallel
  並列化したい計算部分
  !$omp end parallel
  :
end
```

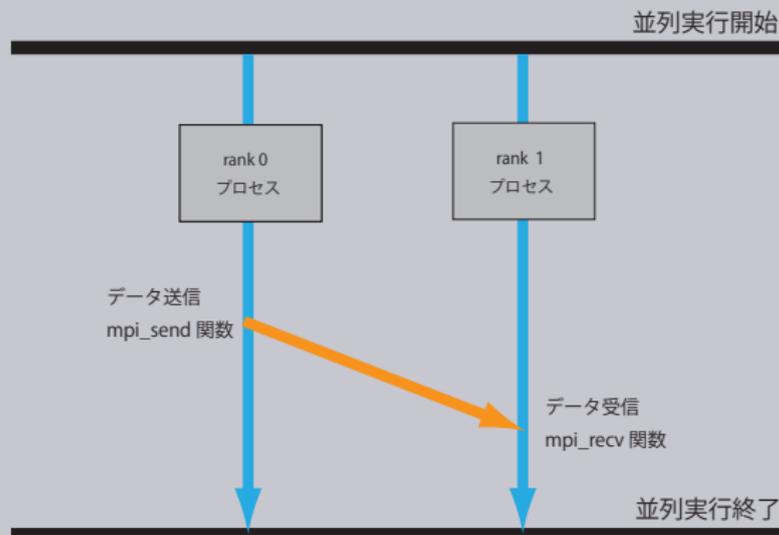
上のように、並列化したいところをディレクティブで挟むのが基本。
並列度等はコンパイラや環境変数で指定。

マシン間並列化: 分散メモリ

- 並列して動く各部分同士の情報のやりとりを 通信 (Message) という形で行う.
- プログラムの動作を並列化用に設計する必要あり.
- プログラムは面倒.
- ハードウェアへの依存性はやや高い. そのため, 移植性はやや低い.
- 解説書多し.
- スーパーコンピュータはほぼこれ (NEC 除く).
- ソフトウェアは MPI (Message Passing Interface) という共通規格に沿った言語, ライブラリを利用.

MPI

- データは分散しており，他プロセスのデータに触れない．
- データはお互いに「明示的に」通信する必要がある．



並列計算のまとめ

- ハードウェアによって並列化の方法が異なるので、ソフトウェアもそれに合わせて選択する。
- 他のソフトウェアに比較すると、MPI はプログラムを書く人が並列化を考えねばならず、やや敷居が高い。
- 阪大のスーパーコンピュータ (SX-9) はベクトル型なので、プログラムテクニク的には難しいことはない。ベクトル化率を高める為の工夫はまた別に必要だが。
- 実は普通の PC でも 4 コア持っていたりするので、4 倍ぐらいまでの並列化は容易にできたりする。
- ベクトル化, MPI についてはサイバーで講習会あり。興味のある方はぜひそちらへ。

UNIX

- (F) “応用数理学 7”, 降旗 大介.
- (K)§1 “UNIX の復習”, 中村 匡秀.

並列化について

- (K)§3 “並列計算とは”, 山本 有作.

ベクトル化

- “ベクトル型計算機のためのベクトル計算”, 菊池 誠 (阪大 サイバーメディアセンター), スパコン 2011 配布資料.
- “パソコン & スーパーコンピュータで計算するためのベクトル & 並列入門”, 福田 優子 (阪大 レーザー研).

OpenMP

- (K)§4 “OpenMP を用いた並列計算”, 谷口 隆晴.
- (T) “科学技術計算のためのマルチコアプログラミング入門”, 中島 研吾.

MPI

- (K)§5 “MPI を用いた並列計算”, 山本 有作.
- (T) “MPI 基礎: 並列プログラミング初級入門”, 片桐 孝洋.

その他

- (O) “大規模計算機システムガイドブック”.
- (O) “パソコン&スーパーコンピュータで計算するための基礎知識”, 福田 優子 (阪大 レーザー研).
- (O) “スーパーコンピュータ利用入門 (講習会入門編資料)”.

F: (降旗) 応用数理学 7 授業用 web.

<http://www.cas.cmc.osaka-u.ac.jp/paoon/Lectures/2012-7Semester-AppliedMath7/>

K: 神戸大学大学院システム情報学研究科・計算科学専攻の計算科学演習用 web.

http://exp.cs.kobe-u.ac.jp/wiki/comp_practice/index.php?%B7%D7%BB%BB%B2%CA%B3%D8%B1%E9%BD%AC

O: 大阪大学サイバーメディアセンター大規模計算機システム

<http://www.hpc.cmc.osaka-u.ac.jp/j/index.html>

T: 東京大学情報基盤センタースーパーコンピューティング部門講習会資料

http://www.cc.u-tokyo.ac.jp/support/kosyu/schedule_kosyu.html

0434-J

Thank You!

Thank You!

0434-J