PARDS:
データフロー同期を用いた
並列プログラム用ライブラリ

荒木 拓也 （NEC）

これは何？

- 並列プログラムを書くためのライブラリです。
- オープンソースです。是非使ってください
  - http://pards.sourceforge.jp/ からダウンロード
  - yum install pards pards-devel でも可
    - Fedora 7以降
  - 修正BSDライセンス

どんな時に使えるの？

- マルチプロセッサマシンでの並列処理
  - C++ / UNIX系OSで
  - α版ながらWindowsでも
- 特に、既存のプログラムの並列化に適しています
  - これまでよりずっと簡単！

どう使うの？

- SPAWN(f)でfをプロセスとしてfork
- Sync<T> a; でプロセス間通信と同期
  - a.write(...)で書き込み
  - a.read()で読み込み
  - read()はwrite()が行われるまでブロック
    - データフロー同期
    - ロック/アンロックで悩む必要無し！

例えば…

```
int main(){
   pards_init();                          // 初期化
   Sync<int> a, b;
   SPAWN(add(1,a,b));                     // addをfork
   a.write(1);                            // addが実行可能に
   printf("b = %d\n", b.read());          // addの終了を待つ
   pards_finalize();                      // 終了処理
}

void add(int x, Sync<int> y, Sync<int> z){
   z.write(x + y.read());                 // yのwriteを待つ
}
```

どうやって実装してるの？

- SPAWNはマクロ
  - #define SPAWN(f) if(fork()==0){f; exit(0)}
- Sync<T>はSystem V IPC
  - 共有メモリ、セマフォ
  - コンストラクタで確保、その情報を変数に保持

プロセスって遅くない？

- 最近のOSではそうでも無いです
  - メモリコピーはCopy-On-Writeで遅延
  - カーネルスレッドと大差無し
- アドレス空間を分離できる利点の方が大きい
  - スレッドセーフかどうか考える必要が無い！

機能はこれで十分なの？

- リストも用意しました
  - パイプライン並列を記述するため
- SyncList<T> , SyncQueue<T>
  - SyncQueueはサイズ制限付き

例えば…

```
SyncList<int> *a = new SyncList<int>; // リスト作成
SPAWN(generator(a));                  // 生産者
SPAWN(consumer(a));                   // 消費者
```

```
for(i = 0; i < 10; i++){
   a->write(i);                       // 値の設定
   a = a->create();                   // 次のセル作成
} ...                                 // (終端設定)
```

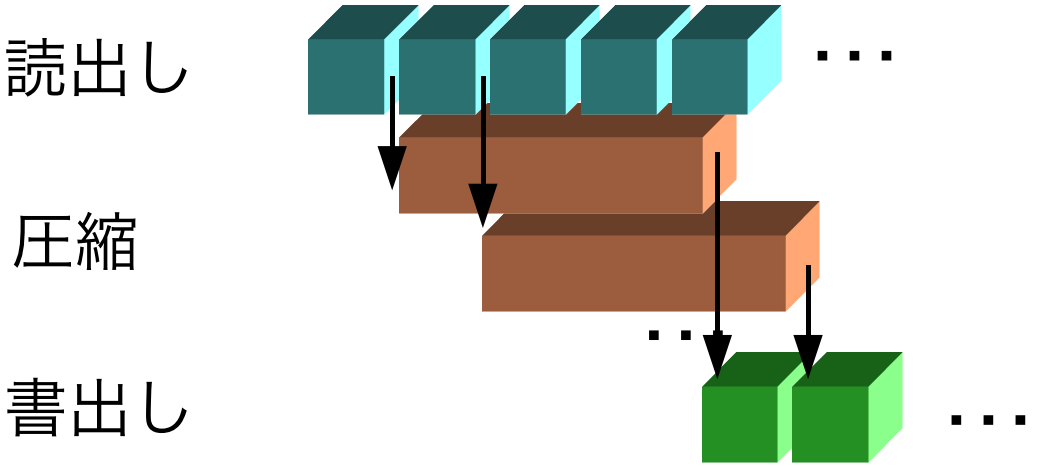
```
while(1){
   printf("val = %d\n",a->read());    // 値取り出し(待つ)
   a = a->release();                  // 解放&次のセルへ(待つ)
   if(a == 0) break;                  // 終端チェック
}
```

他にも…

- プールから仕事を取り出すスタイルをサポートするWorkPool<T1, T2>
  - プロセス生成数を押さえるため
- SPMDスタイルのサポート
- 並行処理のサポート
  - タイムアウト、インタラプト
  - リストの非決定的マージ

実際に使えるの？

- bzip2で試してみました
  - 読出し、圧縮、書出しをパイプライン的に並列実行
  - ブロック毎に並列圧縮



- わずかな変更で並列化！
  - ７１行削除、１４１行追加

性能は？

マシン	逐次版（秒）	並列版（秒）	速度向上
HP-UX (2CPU IA-64)	14.76	8.96	1.64
Fedora 6 (2 * 2CPU Xeon)	3.76	1.43	2.65
Fedora 5 (2CPU Xeon)	8.96	5.55	1.61

10MB程度のファイル、ブロックサイズは900KB

おおむね十分な速度向上

他と比べてどう？

- データフロー同期は良く知られた手法
  - future (MultiLisp), I-structure (Id), 並列論理型言語
  - ライブラリ層での実現 + fork によるアドレス空間の分離が特徴
- 世の中の他のライブラリ
  - Intel Threading Building Blocks
    - スレッド間の同期は無し
    - 大量タスクの効率の良い並列実行を目指す
  - Apple NSOperation (Mac OS X)
    - タスク間の依存を指定

おわりに

- 並列プログラムを簡単に書けるライブラリです
  - 既存のプログラムを並列化するのに適しています
- ぜひ使ってください
- 今後の課題
  - 他のプログラムの並列化
  - スクリプト言語等他言語への移植