5年後のスーパーコンピュータ像？

平石　拓
京都大学学術情報メディアセンター

お題（石川先生より）

- ５年後設置されるマシンを設計開発しろと言われたどうしますか？ (10分 x 6 = 60 分)
  - 自己紹介
  - どういうマシンを作りますか？
    - CPUアーキテクチャ、ネットワーク、性能、メモリ、ストレージ、ＯＳ，ミドルウェア、ライブラリ、言語、ツール、アプリケーション
  - どうやって開発を進めますか？
    - マイルストーンは？
    - 開発体制は？
  - 提案するシステムあるいは開発の進め方にキャッチフレーズを考えてください
  - １０年後のマシンも開発しないといけないとしたら、どうしますか？

自己紹介：これまでの研究

- B4
  - SC言語（S式ベースC言語）を利用したプログラム変形に基づく言語拡張機構
    - Lisp風構文のC言語

SC

```
(def (sum ar n) (fn long (ptr long) int)
  (def s long 0)
  (def i int 0)
  (do-while 1
    (if (>= i n) (break))
    (+= s (aref ar (inc i))) )
  (return s) )
```

C

```
long sum(long *ar, int n){
  long s=0;
  int i=0;
  do{
    if (i >= n) break;
    s += ar[i++];
  } while(1);
  return s;
}
```

自己紹介：これまでの研究

- 修士～博士～PD
  - L-closure：計算状態操作機構
  - 自らの計算状態を書き換えるための言語機構とその軽量実装
    → 中間言語として利用し様々な高水準機能をポータブルかつ高性能に実装
    - マルチスレッディング
    - ごみ集め（garbage collection）
    - バックトラックに基づく負荷分散（並列言語）
  - 対象言語はCの拡張言語
  - 実装言語はやっぱりLisp

現在

- ...という研究をやっていたところ，e-scienceプロジェクトの話をいただき，並列スクリプト言語Xcryptの開発に携わることに

私のバックグラウンド

- 学生の時にHPCの研究をやっていたわけではありません
- OpenMP/MPI，数値系アプリの開発経験はほとんどありません
- Fortran使いではなくLisp使い
- HPC系のほとんどのアプリ開発が今でもC/Fortran，OpenMP/MPI で行われているというのはわりと衝撃だった（数年前）
  - 話としては知っていたが，これほど支配的とは…

私の立ち位置

- アプリ開発者は C/Fortran-OpenMP/MPI で満足しているのか？
  - 基本的に好きでプログラムを書いてるわけではない
  - メモリ管理や通信を自分で書くなんて！
  - アーキテクチャの変更にプログラマが合わせるなんて！
    - ベクトルマシン→汎用クラスタ，GPGPUの台頭
- 低水準な言語であることは必須か？
  - 高水準言語の実装用言語としては必要
  - 性能を追求したい人が使うのはわかる
  - 高生産＋（スパコンで）そこそこの性能が出せる言語がほしい
  - 今のままの高水準言語でいいとは思わない
    - ある程度のデータ配置に関する指示がannotationとして書けるとか
  - もっと気楽に大規模計算環境を使える言語環境があれば，潜在的なユーザを掘り起こせるのでは
    - MapReduceは成功例の1つ

５年後設置されるマシンを設計開発しろと言われたら

- CPUアーキテクチャ、ネットワーク、性能、メモリ、ストレージ、ＯＳ，ミドルウェア、ライブラリ、言語、ツール、アプリケーション
  - 何らかの高水準HPC言語環境も導入したい
  - 5年後ということなので，現時点での有望株で
    - X10，Fortress，XcalableMP，Xcrypt，etc.
  - それ以外はわかる人にまかせます
- どうやって開発を進めますか？
  - マイルストーンは？ 開発体制は？
    - 上記の言語での実アプリ実装，開発コスト評価，ベンチマーク （約3年？）
    - 残りで普及活動
    - 上記ができる人間を集める＆教育（自分も勉強）
- 提案するシステムのキャッチフレーズ
  - 高生産超並列環境

１０年後のマシンも開発しないといけないとしたら、どうしますか？

- HPC用途に耐え得る高水準言語の（言語機能）開発をやっておきたい
- どんな機能が必要？
  - （分散並列環境対応）GC
  - 型チェック（次元チェック）
  - インタプリタ実行が可能
  - ネイティブコードへのコンパイルが可能
  - 並列プログラミングが可能
  - データ配置の指示が（やろうと思えば）可能
  - アセンブリレベルのチューニングも（やろうと思えば）可能