Heterogenous Parallel Computing with Ada Tasking

Jan Verschelde

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
`http://www.math.uic.edu/~jan`
`jan@math.uic.edu`

Ada devroom, FOSDEM 2016, 30 January, Brussels, Belgium

Outline

1. Problem Statement
   - heterogeneous parallel computing
   - high level parallel programming

2. a Mathematical Application
   - solving polynomial systems
   - computational results

Heterogeneous Ada Tasking

1. Problem Statement
   - heterogeneous parallel computing
   - high level parallel programming

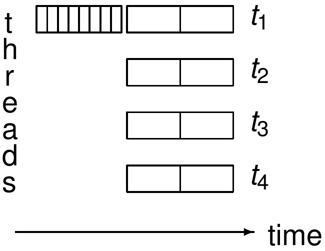
2. a Mathematical Application
   - solving polynomial systems
   - computational results

what is the problem?

Consider a computation in two stages:

1. one producer makes one item after the other;
2. many consumers work in parallel.

Example: 8 items produced for 4 consumers, run by 4 threads.



Problem:

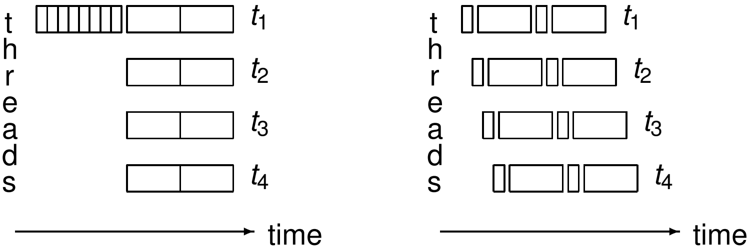
there is only one producer,
who works sequentially
$\Rightarrow$ there can be a long wait
for parallelism to happen.

pipelining

An algorithmic solution is to apply pipelining:

- items produced are placed in a job queue;
- consumers can start as soon as there are jobs in the queue.

Example: 8 items produced for 4 consumers, run by 4 threads.



Heterogeneous Ada Tasking

1. Problem Statement
   - heterogeneous parallel computing
   - high level parallel programming
2. a Mathematical Application
   - solving polynomial systems
   - computational results

high level parallel programming

Shared memory parallel execution:

- no manager/worker paradigm as in message passing,
- but a number of threads processing jobs.

Job scheduling and data distribution:

- A generic procedure defines the launching of tasks.
  This procedure is instantiated with a job procedure.
- The data for the jobs is managed in a queue.
  The job counter is protected by a semaphore.

Why high level? Starting worker tasks...

The generic procedure `Workers` is instantiated with a `Job` procedure, executing its code based on the `id` number.

```
procedure Workers ( n : in natural ) is
   task type Worker ( id,n : natural );
   task body Worker is
   begin
     Job(id,n);
   end Worker;
   procedure Launch_Workers ( i,n : in natural ) is
     w : Worker(i,n);
   begin
     if i < n
      then Launch_Workers(i+1,n);
     end if;
   end Launch_Workers;
begin
   Launch_Workers(1,n);
end Workers;
```

refactoring code

The code for the producer can be a blackbox:

- however: frequent memory allocations should not happen,
- a callback function is called after each produced item.

Data is passed from the producer to the consumers:

- The callback function appends the items to a job queue.
- The consumers process the jobs.

Why heterogeneous parallelism?

- Producer executes entirely different code than consumers.
- Code of the producer is harder to run in parallel.

Heterogeneous Ada Tasking

1. Problem Statement
   - heterogeneous parallel computing
   - high level parallel programming

2. a Mathematical Application
   - solving polynomial systems
   - computational results

mathematical software to solve polynomial systems

Some examples of algorithms and computed objects:

- evaluation and differentiation of polynomials (algebra),
- trajectories of solutions defined by parameters (geometry).

Example of an input:

```
2
 x**2 + 4*y**2 - 4;
        2*y**2 - x;
```

The polynomials define an ellipse and a parabola.
The solutions of the system are points of intersection.
How? Deform systems, starting at the solutions of

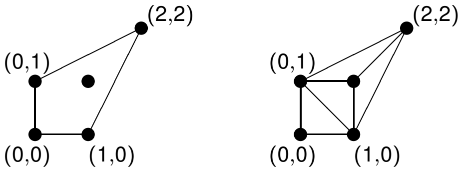
```
2
 x**2 - 1;
 y**2 - 1;
```

Newton polytopes and their volumes

$$f(\mathbf{x}) = \begin{cases} \alpha_1 x_1^2 x_2^2 + \alpha_2 x_1 x_2 + \alpha_3 x_1 + \alpha_4 x_2 + \alpha_5 = 0 \\ \beta_1 x_1^2 x_2^2 + \beta_2 x_1 x_2 + \beta_3 x_1 + \beta_4 x_2 + \beta_5 = 0. \end{cases}$$

Assume that the coefficients, α's and β's, are random numbers.

$A = \{(2,2),(1,1),(1,0),(0,1),(0,0)\}$ is the support of the polynomials and a triangulation is below:



Kushnirenko's theorem: at most 4 solutions.

solving polynomial systems with PHCpack

PHCpack is a package for Polynomial Homotopy Continuation.
ACM Transactions on Mathematical Software achived version 1.0 (Ada 83) as Algorithm 795, vol. 25, no. 2, pages 251–276, 1999.

blackbox solver:

`phc -b` computes all isolated solutions of a polynomial system.

Version 2.0 was rewritten using concepts of Ada 95 and extended with arbitrary multiprecision arithmetic.

PHCpack contains **MixedVol** (ACM TOMS Algorithm 845), developed by Tangan Gao, Tien-Yien Li, Xing Li, and Mengnien Wu. MixedVol computes mixed volumes fast.

Distributed under the GNU General Public License.

Public repository under version control at `https://github.com/janverschelde/PHCpack`.

Heterogeneous Ada Tasking

1. Problem Statement
   - heterogeneous parallel computing
   - high level parallel programming

2. a Mathematical Application
   - solving polynomial systems
   - computational results

benchmark system: cyclic 13-roots

The cyclic n-roots problem defines a family of n polynomial equations in n variables. For example, $n = 5$:

$$\left\{ \begin{array}{rcl} x_1 + x_2 + x_3 + x_4 + x_5 & = & 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1 & = & 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_1 + x_5x_1x_2 & = & 0 \\ x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_1 + x_4x_5x_1x_2 + x_5x_1x_2x_3 & = & 0 \\ x_1x_2x_3x_4x_5 - 1 & = & 0. \end{array} \right.$$

If n is prime, then all solutions are isolated.

For $n = 13$: there are 2,704,156 solutions.

before the upgrade

On two 8-core Intel Xeon E5-2670 processors with hyperthreading, the fastest times are obtained with 32 tasks.

Historical timings, with version 2.3.05 on Thu 10 Dec 2015:

```
$ time phc -m -t32 < incyc13mtmvc > outcyc13mtmvc

real    176m35.846s
user    4350m50.681s
sys     1m2.537s
$
```

Good speedup: `4350/176` = 24.7, or 3 days versus 3 hours.

Problem: for the first half hour only one core works!

after the upgrade

On two 8-core Intel Xeon E5-2670 processors with hyperthreading, the fastest times are obtained with 32 tasks.

Since version 2.3.06 of PHCpack, timings on Fri 11 Dec 2015:

```
$ time phc -m -t32 < incyc13mvct32n > outcyc13mvct32n

real    144m4.982s
user    4409m52.973s
sys     1m19.984s
$
```

Improvement in the wall clock time: $176 - 144$ = 32 minutes.