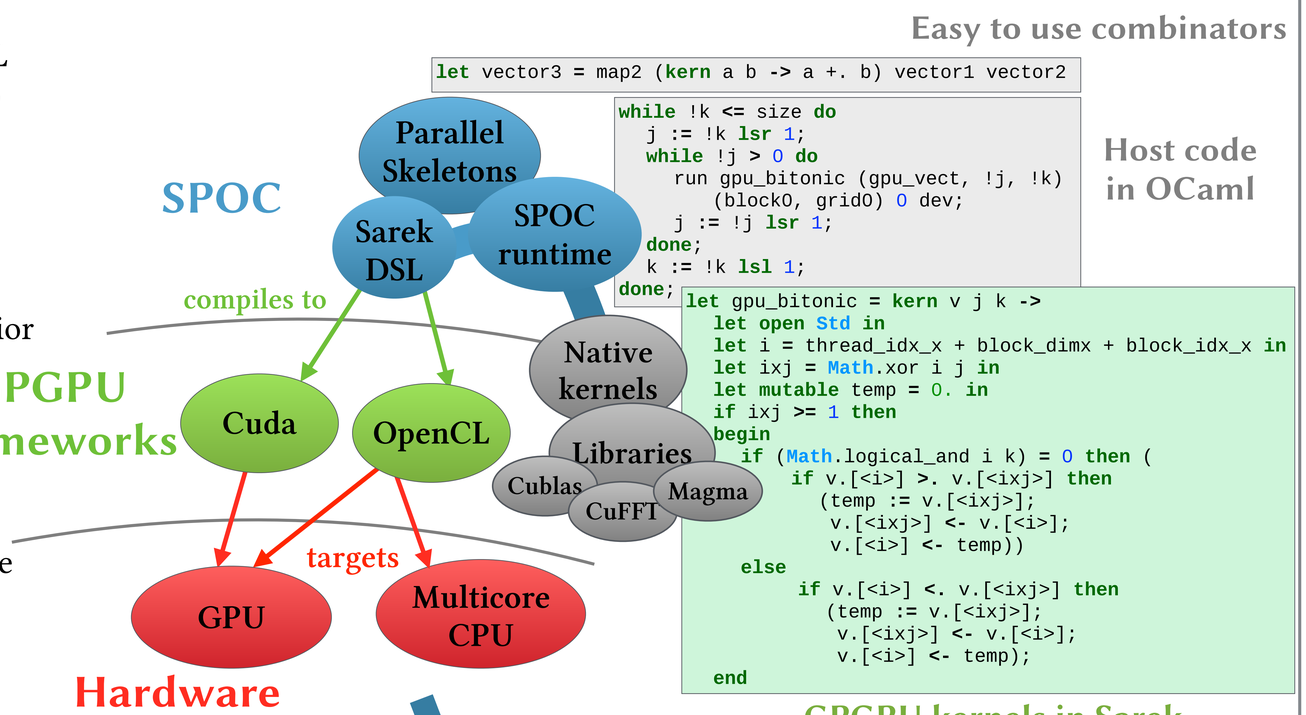
Mathias Bourgoin & Emmanuel Chailloux
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, APR, F-75005, Paris, France
firstname.lastname@lip6.fr

High Performance Web-Client Programming with SPOC

SPOC : Efficient Abstractions for GPGPU Programming

- **Portable**
  - Unifies Cuda/OpenCL/**WebCL**
  - JIT compilation from Sarek to Cuda/OpenCL
- **Heterogenous**
  - Compatible with multiple devices
  - Dynamically adapts its behavior for any target
- **Accessible and Safe**
  - Automatic memory transfers
  - Static type checking
  - Host/GPU code in a single file with a coherent language
- **High Performance**
  - Compatible with hand-tuned native kernels and libraries
  - Parallel skeletons offer automatic optimisations



Easy to use combinators

```
let vector3 = map2 (kern a b -> a +. b) vector1 vector2
```

Host code in OCaml

```
while !k <= size do
  j := !k lsr 1;
  while !j > 0 do
    run gpu_bitonic (gpu_vect, !j, !k)
        (block0, grid0) 0 dev;
    j := !j lsr 1;
done;
k := !k lsl 1;
done;
```

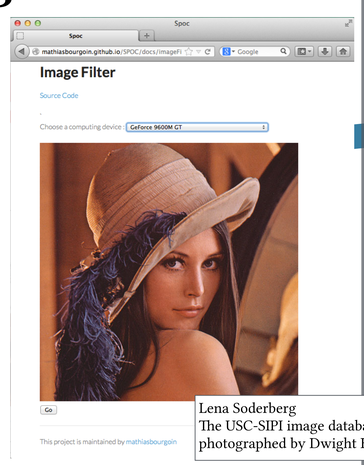
```
let gpu_bitonic = kern v j k ->
  let open Std in
  let i = thread_idx_x + block_dimx + block_idx_x in
  let ixj = Math.xor i j in
  let mutable temp = 0. in
  if ixj >= 1 then
  begin
    if (Math.logical_and i k) = 0 then (
      if v.[<i>] >. v.[<ixj>] then
        (temp := v.[<ixj>];
         v.[<ixj>] <- v.[<i>];
         v.[<i>] <- temp))
    else
      if v.[<i>] <. v.[<ixj>] then
        (temp := v.[<ixj>];
         v.[<ixj>] <- v.[<i>];
         v.[<i>] <- temp);
  end
```

GPGPU kernels in Sarek

```
let gpu_to_gray = kern v ->
  let open Std in
  let tid = thread_idx_x + block_dimx + block_idx_x in
  if tid <= (512 * 512) then (
    let i = (tid * 4) in
    let res = int_of_float ((0.21 *. (float (v.[<i>]))) +.
                            (0.71 *. (float (v.[<i + 1>]))) +.
                            (0.07 *. (float (v.[<i + 2>]))) ) in
    v.[<i>] <- res;
    v.[<i+1>] <- res;
    v.[<i+2>] <- res )
```

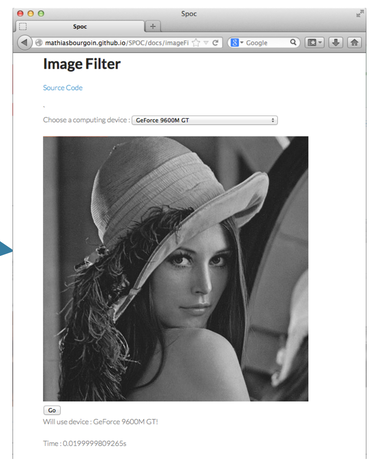
js_of_ocaml : web-client programming with OCaml

```
let canvas = createCanvas document in
 canvas##width <- 512;
 canvas##height <- 512;

 let image : imageElement Js.t = createImg document in
 image##src <- Js.string "lena.png";

 let c = canvas##getContext (Dom_html._2d_) in
 image##onload <-
   handler (fun _ ->
       c##drawImage (image, 0., 0.);
       Dom.appendChild body (newLine ());
     Dom.appendChild body canvas;
     let devs =
     Devices.init ~only:Devices.OpenCL () in

     let imageData = c##getImageData (0., 0., 512., 512.) in
     let data = imageData##data in
     Dom.appendChild body (newLine ());

     Array.iter
       (fun (n) ->
          let option = createOption document in
          append_text option n.Devices.general_info.Devices.name;
          Dom.appendChild select_devices  option)
       devs;

     Dom.appendChild body (button (f select_devices
                                      devs data imageData c));
   Js._true);
```

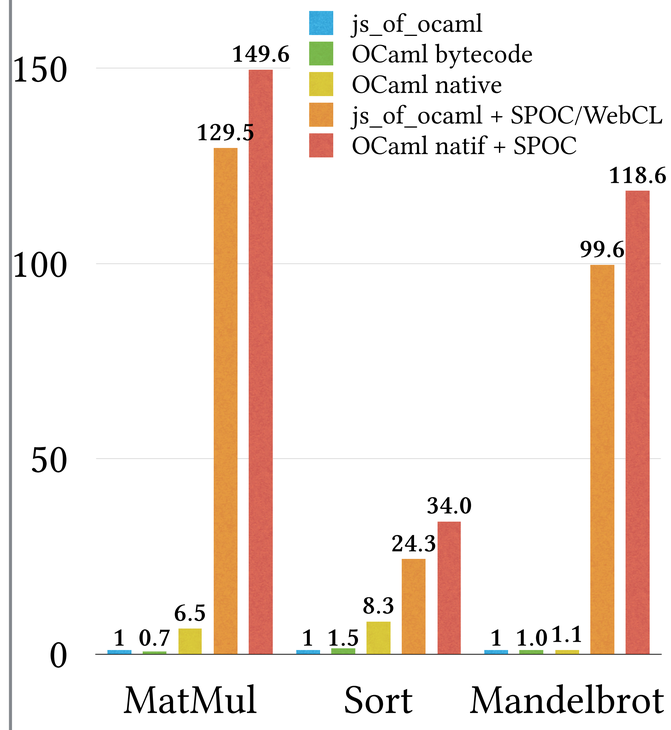


Lena Soderberg
The USC-SIPI image database
photographed by Dwight Hooker.

WebCL



Benchmarks



Speedups (higher, better) using
- Intel Core-i7 3770 (CPU)
- Geforce GTX 680 (GPU)

SPOC website

http://www.algo-prog.info/spoc

References

J. Vouillon and V. Balat
From bytecode to javascript: the js_of_ocaml compiler.
Software: Practice and Experience, 2013

M. Bourgoin, E. Chailloux and J.-L. Lamotte
Efficient Abstractions for GPGPU Programming
International Journal of Parallel Programming, 2013

M. Bourgoin, E. Chailloux and J.-L. Lamotte
Spoc: GPGPU Programming Through Stream Processing with OCAML
Parallel Processing Letters, 2012