Algorithmic skeleton (MURRAY COLE)

↳ ready to use parallel programming abstraction implementing a parallel design pattern in a parametric way } nestable

Cilk `_cilk_for ( ------ ) { Body }`

OpenMP `#pragma parallel for`
`for ( ---- ) { Body }`

Typical skeleton framework

DP (data parallel) { map (..), reduce (..), stencil, parallel for

SP (stream parallel) { pipeline (---, ---, ...), farm (...)

parameters

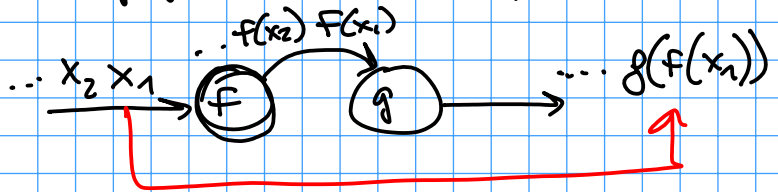
FUNCTIONAL "code" parameters (business logic code)

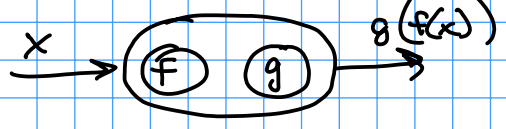
NON FUNCTIONAL define "how" the computation is performed

WHAT IS COMPUTED

```
float f(float x) { ... return ( x ); }
float g( -- -- ) { --- }
```

`pipeline ( f, g );` → par degree = 1





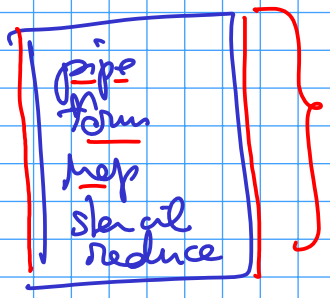
pipeline ({f, g}, 2)

farm(f, 8)

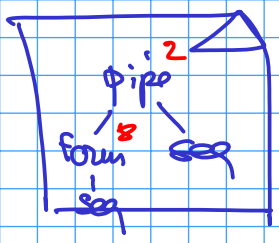
pipeline (farm(f, 8), g)

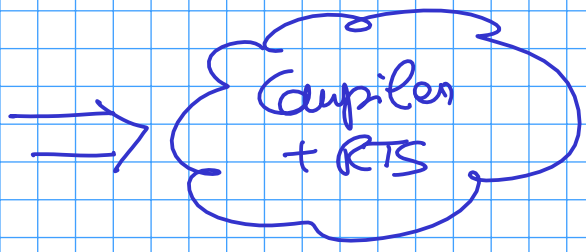
```
Pipeline<float, float>  p;
p.add_stage(f);
p.add_stage(g);
```

Problem

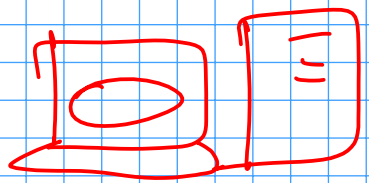


clear { functional / parallel } semantics





⇒ a.out



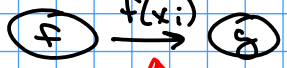
Multicore CPU + GPU

a) map (std::fun<Tin, Tout> f)

map_{CPU} map_{GPU}

b) { mapcpu(...)
mapgpu(...)

pipeline (f, g);



explicitly manage copy of data

COW/NOW
TCP/IP WRAPPER (NETWORK FRAMEWORK)
COPY DATA across C.A.

MULTICORE
threads / shared memory
synchronization

send "pointer" as capability

(Edinburgh) Cole '88 (4 annotations not mentionable)

Imperial College
J. Darlington
"functional"
skeletons = HOF
↓
FORTRAN - S

Pisa
P^3L
coordination language

```
                         float * x, int n
seq NameX in(           )  out(   )
  $c{ ≡≡≡ }c$              $F77{ ≡ }F77$
end seq

pipe Name  in( )  out( )
   NameX   in( )  out( )
                  out( )
 ...
end pipe
```

Gorlatch (Muenster)
"functional"

} 90s

FRANCE
- ORLEANS → **OSL**
- → CALCIUM ⇓ (Java) SKANDIUM
- CLERMONT FERRAND → SKIPPER †

GERMANY
⇓
MUESLI (MPI) + (OpenMP)
CPU + NOW   COW/NOW

Tokio
SkeTo
NOW

ITALY → ASSIST
→ **FASTFLOW**
CPU
GPU
COW

SWEDEN
↓
SKEPU
CPU
GPU

OPENMP

TBB (INTEL)

MICROSOFT
TPL → DATA FLOW

SKELETON FRAMEWORKS

{ skeletons } — nestable

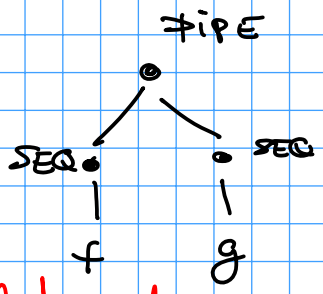
$\Delta = seq(f) \mid pipeline(\Delta, \Delta) \mid farm(\Delta) \mid map(\Delta) \mid$

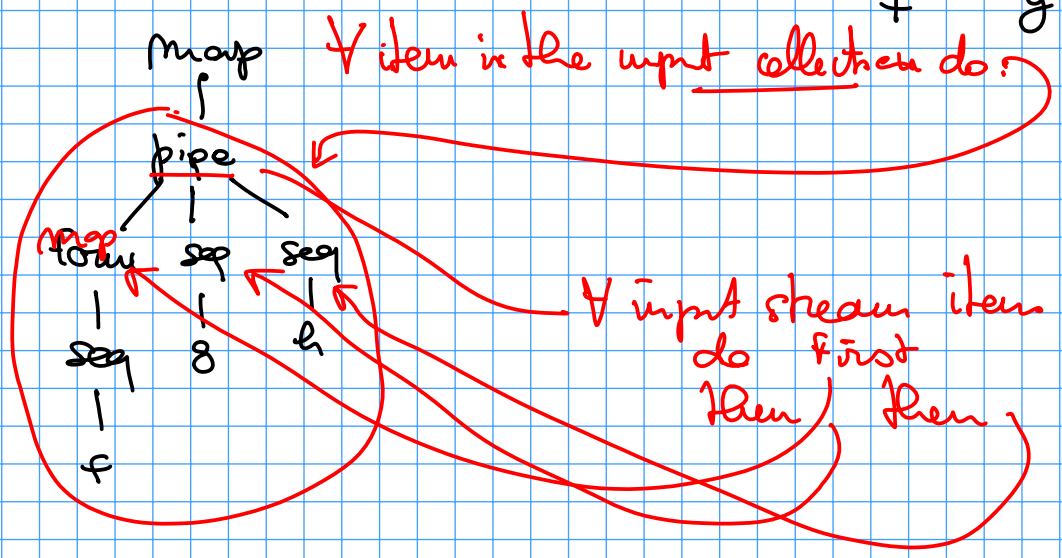
$pipeline(\Delta list)$

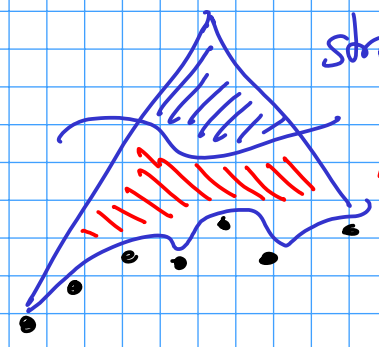
$reduce(\Delta)$;

f : fusion

$pipeline(seq(f), seq(g))$







stream parallelism

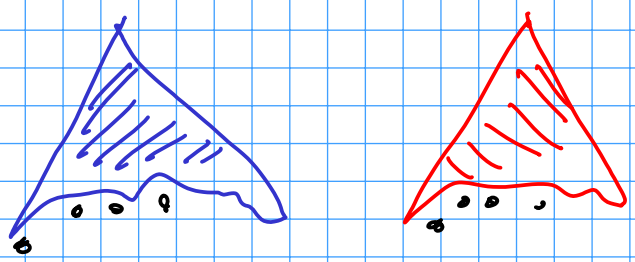
map parallelism

seq wrappers

map

"TWO TIER MODEL"

(KUCHEN)



IMPLEMENTATION

TEMPLATE BASED

DATA FLOW

{skeletons} PIPE

COW

template(pipe, cow, f g process network)

template(pipe, cow, ○→○→○)

sk tree

{templates}

t_i models & implements skeleton sk_j on architecture HW_k

algorithm → annotated skeleton tree: sk <t>

pgm = pipe(f, g);

⇒