Automatic Generation of Efficient Codes from Mathematical Descriptions of Stencil Computation

Takayuki Muranushi[1] Seiya Nishizawa[1] Hirofumi Tomita[1]
Keigo Nitadori[1] Masaki Iwasawa[1] Yutaka Maruyama[1]
Hisashi Yashiro[1] Yoshifumi Nakamura[1] Hideyuki Hotta[2]
Junichiro Makino[3] Natsuki Hosono[4] Hikaru Inoue[5]

[1]RIKEN Advanced Institute for Computational Science
[2]Chiba University [3]Kobe University
[4]Kyoto University [5]Fujitsu Ltd.

Sep 22, 2016

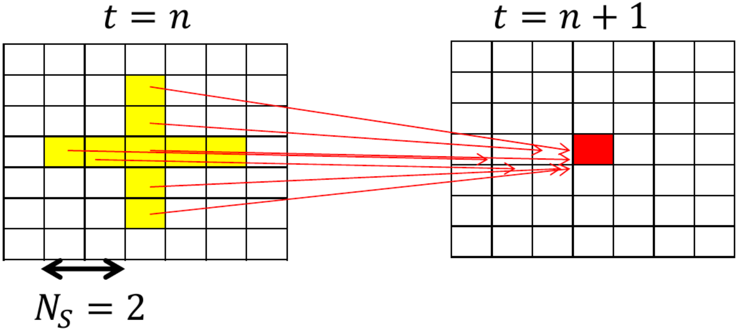
for FHPC 2016 workshop / ICFP'16 Nara, Japan

Programming Language

Formura

Programming language Formura

Domain specific language for stencil computaion



Good news of Formura 1/2

1.184 Petaflops (11.62% of the peak)
on 663,552 cores

Good news of Formura 1/2

ACM Gordon Bell Prize Finalist

Wednesday, November 16th

TIME	SESSION / PRESENTATION	PRESENTERS
11:30am - 12pm	ACM Gordon Bell Finalist: Simulations of Below-Ground Dynamics of Fungi: 1.184 Pflops Attained by Automated Generation and Autotuning of Temporal Blocking Codes	Muranushi, Hotta, Makino, Nishizawa, Tomita, Nitadori, Iwasawa, Hosono, Maruyama, Inoue…

Good news of Formura 2/2

$$\frac{\partial \rho}{\partial t} = - \sum_{i=1}^{3} \frac{\partial}{\partial x_i} (\rho v_i)$$

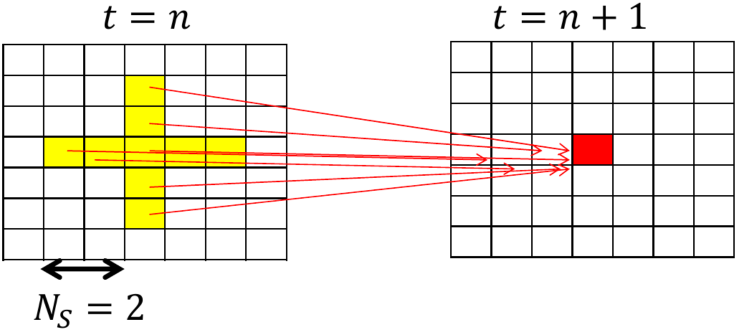
```
ddt_ρ = - Σ fun(i) ∂ i (ρ * v i)
```

Formura

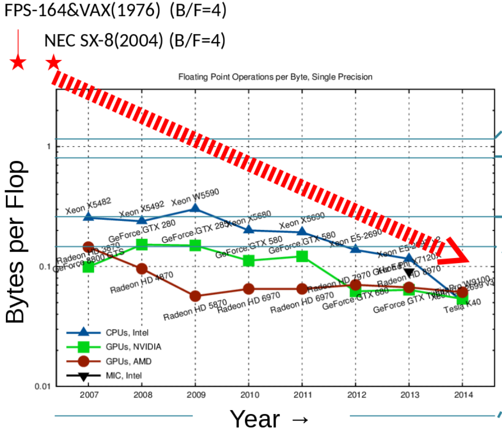
- is a functional programming language
- is implemented in a functional programming language (Haskell)

Backend: How we generate efficient codes

Stencil Computation



Byte / Flops of hardwares are decreasing



Applications

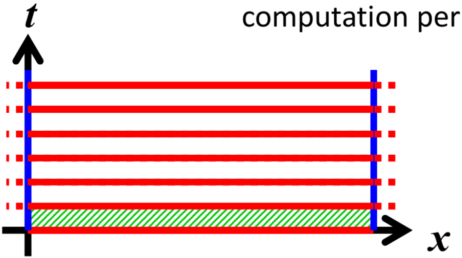
-
- 2d diffusion
  B/F=1.14
- 3d diffusion
  B/F=0.8
- Seismic wave
  B/F=0.275
- 3d CIP Hydro
  B/F=0.16
- Godunov Hydro
  B/F=5.92×10^{-3}

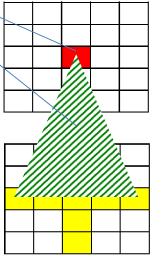
Data cited from:*Comparison of required floating point operations per byte when using single precision in order to transition between compute-limited and memory-bandwidth-limited regimes.*.(c) Karl Rupp

Naive implementation of stencil computation

$$\text{The optimal } \frac{B}{F} = \frac{2H_e}{C_e}$$

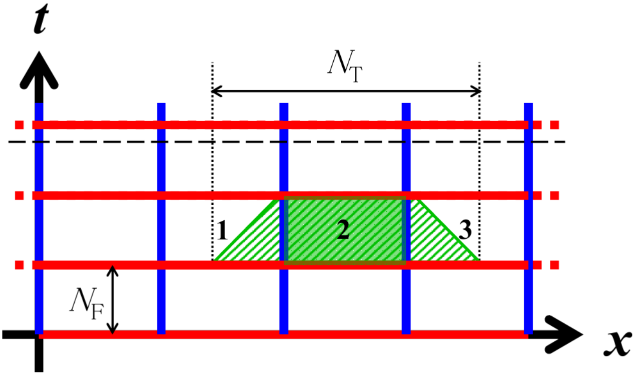
Information per cell: H_e
computation per cell: C_e



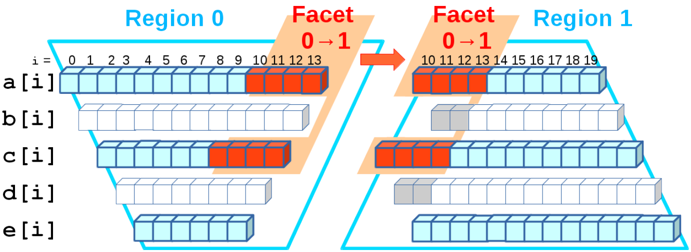


Temporal Blocking

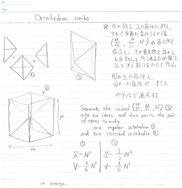
$$\text{The optimal } \frac{B}{F} = \frac{2H_e}{C_e}\left(\frac{1}{N_F} + \frac{2dN_s}{N_T}\right)$$

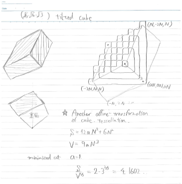


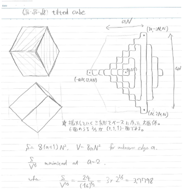
Decompose & fuse array computations in space-time

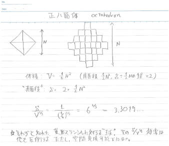



```
manifest :: a[i]
            b[i] = a[i-1] + a[i] + a[i+1]
manifest :: c[i] = b[i-1] * b[i] * b[i+1]
            d[i] = c[i-1] + c[i] + c[i+1]
manifest :: e[i] = d[i-1] * d[i] * d[i+1]
```







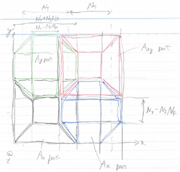


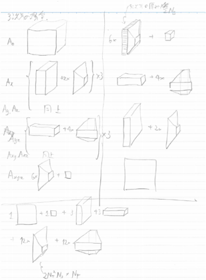
No.

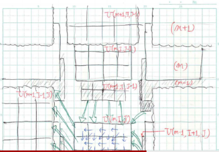
Date

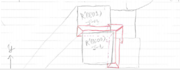
tessellation

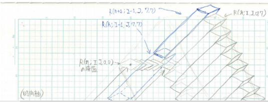
	充?	$S/V^{2/3}$	
octahedron	×	3.3019	$6^{2/3}$
$\sqrt{8}$-tilted cube	○	3.7798	$3 \times 2^{1/3}$
($\sqrt{3}$)-tilted cube	○	4.1602	$2 \times 3^{2/3}$
octabitetrahedron	○	3.7798	$3 \times 2^{1/3}$
pseudooctahedron	○	4.1602	$2 \times 3^{2/3}$
octahedron combo	○	4.4026	$4/3 \times 6^{2/3}$
mothership 1	×	3.5851	$(12/5)^{2/3}$
mothership 2 ~ARK~	○	3.9685	$5/2 \times 2^{2/3}$
studoctahedron	○	3.7798	$3 \times 2^{1/3}$
radioactive	×	3.7090	$7/2 \times (12/11)^{2/3}$
rhombic dodecahedra	○	3.7798	$3 \times 2^{1/3}$
cube	○	6	



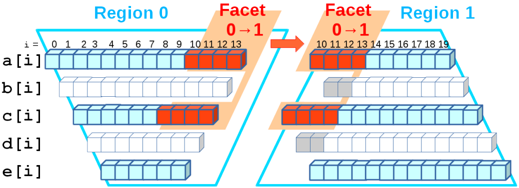


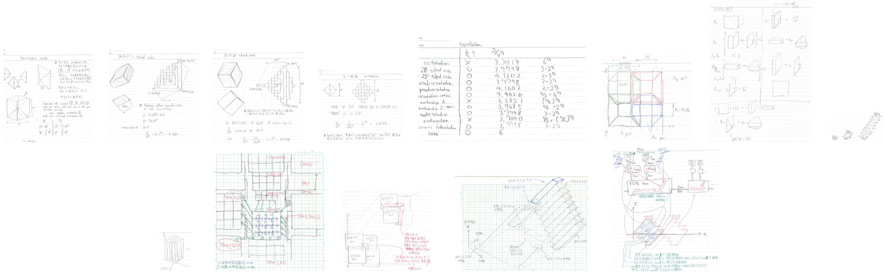






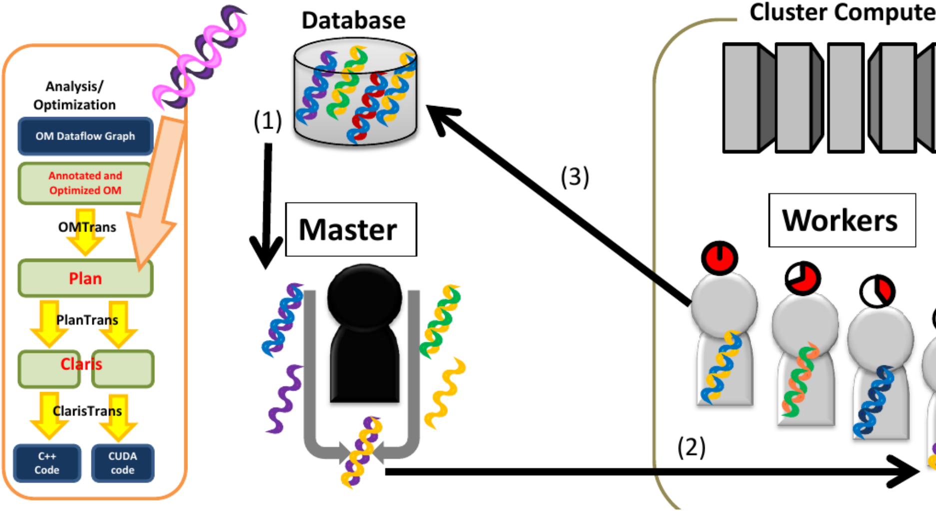
In which language shall we code?

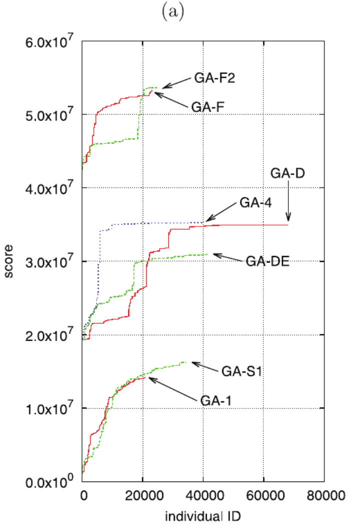


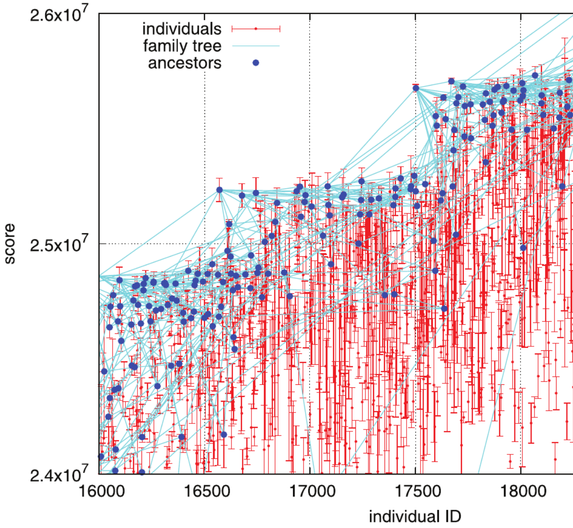


Paraiso : a DSL embedded in Haskell (Muranushi, 2012)

among Nikola (Mainland & Morrisett, 2010), Obsidian (Svensson, 2011), Accelerate (Chakravarty et al., 2011), SPOC (Bourgoin et al., 2012), NOVA (Collins et al., 2014), and LMS series (Rompf, 2012).


Analysis/
Optimization
OM Dataflow Graph
Annotated and
Optimized OM
OMTrans
Plan
PlanTrans
Claris
ClarisTrans
C++
Code
CUDA
code
Database
(1)
Master
(3)
Cluster Compute
Workers
(2)


(a)
6.0x10^7
5.0x10^7
4.0x10^7
3.0x10^7
2.0x10^7
1.0x10^7
0.0x10^0
GA-F2
GA-F
GA-D
GA-4
GA-DE
GA-S1
GA-1
score
0
20000
40000
60000
80000
individual ID


2.6x10^7
2.5x10^7
2.4x10^7
individuals
family tree
ancestors
score
16000
16500
17000
17500
18000
individual ID

Paraiso: a bad sell

```haskell
proceedSingle :: Int -> BR -> Dim BR -> Hydro BR -> Hydro BR -> B (Hydro BR)
proceedSingle order dt dR cellF cellS = do
  let calcWall i = do
        (lp,rp) <- interpolate order i cellF
        hllc i lp rp
  wall <- sequence $ compose calcWall
  foldl1 (.) (compose (\i -> (>>= addFlux dt dR wall i))) $ return cellS
```

Our team

Formura : a standalone DSL

Design principle of Formura

- Simple enough
- Rich enough

Syntax of Formura

```
# dimension declaration
dimension :: 3
# array declaration
double [] :: vx, vy, vz
# array computation
A2[i,j,k] = A[i-1] + A[i+1]
# Tuple
v = (vx, vy, vz)
# Lambda expression
tripe = fun (x) 3 * x
```

Tuples are functions

```
      (a,b) 1   = b
(f,(h,p,c)) 1 2 = c
```

Inferred promotion to tuples and functions

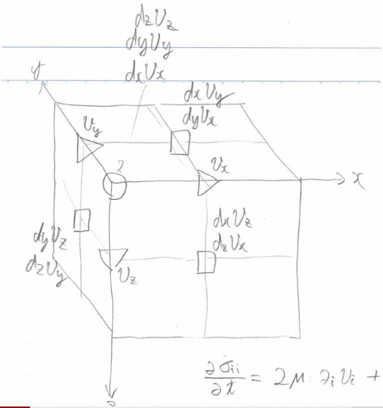
```
      x + (a,b) = (x+a,x+b)
  (x,y) + (a,b) = (x+a,y+b)
(x,y) + (a,b,c) = ⊥

      (f + g) x = f x + g x
  (f + g + 1) x = f x + g x + 1
```

```
rk4 = fun(ddt) \
  fun(sys_0) let \
    sys_q4 = sys_0 + dt/4 * ddt(sys_0)
    sys_q3 = sys_0 + dt/3 * ddt(sys_q4)
    sys_q2 = sys_0 + dt/2 * ddt(sys_q3)
    sys_next = sys_0 + dt * ddt(sys_q2)
  in sys_next
```

Differentiation Operators

```
ddx = fun(a) (a[i+1/2,j,k] - a[i-1/2,j,k])/dx
ddy = fun(a) (a[i,j+1/2,k] - a[i,j-1/2,k])/dy
ddz = fun(a) (a[i,j,k+1/2] - a[i,j,k-1/2])/dz
```



Nabla and Summation

```
∂ = (ddx,ddy,ddz)
Σ = fun (e) e 0 + e 1 + e 2
```

Evaluation of formura expression

```
Σ fun(i) ∂ i (ρ * v i)
```

Evaluation of formura expression

```
Σ fun(i) ∂ i (ρ * v i)
```

```
Σ = fun (e) e 0 + e 1 + e 2
```

Evaluation of formura expression

$$\Sigma\ \texttt{fun(i)}\ \partial\ \texttt{i}\ (\rho\ *\ \texttt{v i})$$

```
Σ = fun (e) e 0 + e 1 + e 2
```

$$\longrightarrow\ (\texttt{fun(i)}\ \partial\ \texttt{i}\ (\rho\ *\ \texttt{v i}))\ 0$$
$$+\ (\texttt{fun(i)}\ \partial\ \texttt{i}\ (\rho\ *\ \texttt{v i}))\ 1$$
$$+\ (\texttt{fun(i)}\ \partial\ \texttt{i}\ (\rho\ *\ \texttt{v i}))\ 2$$

Evaluation of formura expression

```
(fun(i) ∂ i (ρ * v i)) 0
```

Evaluation of formura expression

$$\texttt{(fun(i) } \partial \texttt{ i (} \rho \texttt{ * v i)) 0}$$
$$\longrightarrow \partial \texttt{ 0 (} \rho \texttt{ * v 0))}$$

Evaluation of formura expression

$$\texttt{(fun(i)}\ \partial\ \texttt{i}\ (\rho\ *\ \texttt{v i))}\ \texttt{0}$$
$$\longrightarrow\ \partial\ \texttt{0}\ (\rho\ *\ \texttt{v 0))}$$

```
        ∂ = (ddx,ddy,ddz)
        v = (vx,vy,vz)
(a,b,c) 0 = a
```

Evaluation of formura expression

$$\texttt{(fun(i)}\ \partial\ \texttt{i}\ (\rho\ *\ \texttt{v i))}\ \texttt{0}$$

$$\longrightarrow\ \partial\ \texttt{0}\ (\rho\ *\ \texttt{v 0))}$$

```
        ∂ = (ddx,ddy,ddz)
        v = (vx,vy,vz)
(a,b,c) 0 = a
```

$$\longrightarrow\ \texttt{ddx}\ (\rho\ *\ \texttt{vx})$$

Evaluation of formura expression

$$\texttt{ddx} \ (\rho \ * \ \texttt{vx})$$

Evaluation of formura expression

$$\texttt{ddx}\ (\rho * \texttt{vx})$$

```
ddx = fun(a) (a[i+1/2,j,k] - a[i-1/2,j,k])/dx
```

Evaluation of formura expression

$$\texttt{ddx } (\rho * \texttt{vx})$$

```
ddx = fun(a) (a[i+1/2,j,k] - a[i-1/2,j,k])/dx
```

$$\longrightarrow ((\rho * \texttt{vx})[\texttt{i+1/2,j,k}] - (\rho * \texttt{vx})[\texttt{i-1/2,j,k}])/\texttt{dx}$$

Evaluation of formura expression

$$\mathtt{ddx}\ (\rho * \mathtt{vx})$$

```
ddx = fun(a) (a[i+1/2,j,k] - a[i-1/2,j,k])/dx
```

$$\longrightarrow ((\rho * \mathtt{vx})[\mathtt{i+1/2,j,k}] - (\rho * \mathtt{vx})[\mathtt{i-1/2,j,k}])/\mathtt{dx}$$

$$\longrightarrow (\rho[\mathtt{i+1/2,j,k}] * \mathtt{vx}[\mathtt{i+1/2,j,k}] - \rho[\mathtt{i-1/2,j,k}] * \mathtt{vx}[\mathtt{i-1/2,j,k}])/\mathtt{dx}$$

Evaluation of formura expression

$$\Sigma\ \mathtt{fun(i)}\ \partial\ \mathtt{i}\ (\rho\ *\ \mathtt{v}\ \mathtt{i})$$

Evaluation of formura expression

$$\Sigma\ \mathtt{fun(i)}\ \partial\ \mathtt{i}\ (\rho\ *\ \mathtt{v\ i})$$

$$\longrightarrow (\rho\mathtt{[i+1/2,j,k]} * \mathtt{vx[i+1/2,j,k]} - \rho\mathtt{[i-1/2,j,k]} * \mathtt{vx[i-1/2,j,k])/dx} + (\rho\mathtt{[i,j+1/2,k]} * \mathtt{vy[i,j+1/2,k]} - \rho\mathtt{[i,j-1/2,k]} * \mathtt{vy[i,j-1/2,k])/dy} + (\rho\mathtt{[i,j,k+1/2]} * \mathtt{vz[i,j,k+1/2]} - \rho\mathtt{[i,j,k-1/2]} * \mathtt{vz[i,j,k-1/2])/dz}$$

Evaluation of formura expression

$$\sum_{i=1}^{3} \frac{\partial}{\partial x_i} (\rho v_i)$$

```
Σ fun(i) ∂ i (ρ * v i)
```

⟶ (ρ[i+1/2,j,k] * vx[i+1/2,j,k] -
ρ[i-1/2,j,k] * vx[i-1/2,j,k])/dx +
(ρ[i,j+1/2,k] * vy[i,j+1/2,k] -
ρ[i,j-1/2,k] * vy[i,j-1/2,k])/dy +
(ρ[i,j,k+1/2] * vz[i,j,k+1/2] -
ρ[i,j,k-1/2] * vz[i,j,k-1/2])/dz

More to talk about

- Modular Reifiable Matching (MRM)(Oliveira et al., 2015) + Pattern synoynm solves “expression problem”
- Details of code transformation paths
- Varieties of temporal blocking methods
- How we have gave proof to certain types of temporal blocking methods

Conclusion

Functional programming

- is a good choice for user interface
  $\longrightarrow$ weather scientists and astronomers can use it
- is crucial in implementing all the program transformations
  $\longrightarrow$ achieves high performance

Conclusion

1.184 Pflops

Formura

Bibliography I

Bourgoin, M., Chailloux, E., & Lamotte, J.-L. 2012, Parallel Processing Letters, 22, 1240007

Chakravarty, M. M., Keller, G., Lee, S., McDonell, T. L., & Grover, V. 2011, in Proceedings of the sixth workshop on Declarative aspects of multicore programming, ACM, 3–14

Collins, A., Grewe, D., Grover, V., Lee, S., & Susnea, A. 2014, in Proceedings of ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming, ACM, 8

Mainland, G., & Morrisett, G. 2010in , ACM, 67–78

Oliveira, B. C. d. S., Mu, S.-C., & You, S.-H. 2015, in Proceedings of the 8th ACM SIGPLAN Symposium on Haskell, ACM, 82–93

Rompf, T. 2012, PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Svensson, J. 2011, PhD thesis, Chalmers University of Technology