DSL Toolchains for Performance Portable Geophysical Fluid Dynamic Models

Carlos Osuna
Meteoswiss

carlos.osuna@meteoswiss.ch

DSL Toolchains for Performance Portable

Geophysical Fluid Dynamic Models

COSMO 2km ensembles (21 members)

COSMO 1km
DSL Toolchains for Performance Portable
Geophysical Fluid Dynamic Models
Disruptive emergence of accelerators in HPC. Large memory bandwidth have speeded up our models. On NVIDIA GPUs:

- dynamical cores – 2-3x
- physical parametrizations 3-7x
- spectral transforms > 10x
How to achieve portability for our GFD models?
DSL Toolchains for Performance Portable Geophysical Fluid Dynamic Models
COSMO global 1km

Discussion paper under review
https://www.geosci-model-dev-discuss.net/gmd-2017-230/
DSL Toolchains for

Performance Portable

Geophysical Fluid Dynamic Models
Carlos Osuna, MPI Seminar, 21 November 2017

```c
#pragma omp parallel!
#pragma omp directive_schedule!
#pragma omp target!
	data i_startdx, i_enddx, i, j, k, gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
	data gb, jn, jn1, jn2, jn3, jn4, jn5, jn6, jn7, jn8
```
A domain specific language (DSL) for GFDs:
• abstracts away all unnecessary details of a programming language (Fortran) implementation
• concise syntax: only those keywords needed to express the numerical problem

The GridTools Ecosystem
• Set of C++ APIs / Libraries
• DSL for PDEs
• Large class of problems
• Performance Portability
• Separation of concerns
• Interface to Fortran
• Open source license
DSL for Weather Codes on Unstructured Grids

DSLs are a successful approach to deal with the software “productivity gap.” ESCAPE develops a DSL based on GridTools: **library tools for solving PDEs on multiple grids.**

The DSL syntax requires only the grid-point operation:

\[
\text{lap}() = -4 \times u + u(i+1) + u(i-1) + u(j+1) + u(j-1)
\]

Tiled loop templates and parallelization are abstracted. Use of fast on-chip memory is also abstracted. Grid is abstracted (in combination with Atlas)
Writing Operators using DSL

```cpp
struct Laplace {
    typedef in_accessor<0, range<-1,1>-1,1> > u;
    typedef out_accessor<1> lap;

    template<typename Evaluation>
    static void Do(Evaluation const& eval, full_domain) {
        eval(lap()) = eval(-4*u() + u(i+1) + u(i-1) + u(j+1) + u(j-1));
    }
};
```
The COSMO Weather Model

- Init
- Input
- Physics
- Dynamics
- Relaxation
- Nudging
- Output
- Cleanup

Programming Languages
- Fortran: OpenACC
- C++: CUDA

The C++ Dycore Rewrite using STELLA

STELLA is a domain specific language directly embedded in C++ geared towards stencil computations. STELLA can generate CUDA, OpenMP and X86 code.
Global Grids

- New DSL constructs for stencils on global grids
- Expose a unstructured grid API
- Leverage grid structure for performance
Grid Abstraction

- The DSL syntax elements are grid independent.
- The same used code using DSL can be compiled for multiple Grids.
- GridTools will support efficient native layouts for octahedral/icosahedral.
- GridTools will use Atlas in order to support any grid.

```cpp
const auto& offsets = connectivity<edges, vertices, Color>::offsets();
for (auto off : offsets) {
    eval(flux) += eval(pD(off));
}

eval(flux) = eval(sum_on_vertices(0.0, pD));
```
Example of GridTools DSL: MPDATA

template <uint_t Color> struct upwind_flux {
  using flux = accessor<0, enumtype::Inout, icosahedral_topology_t::edges>;
  using pD =
    in_accessor<1, icosahedral_topology_t::vertices, extent<0, 1, 0, 1>>;
  using vn = in_accessor<2, icosahedral_topology_t::edges, extent<0, 1, 0, 1>>;

typedef boost::mpl::vector<flux, pD, vn> arg_list;

  template <typename Evaluation> static void Do(Evaluation &eval, k_full) {

    constexpr auto neighbors_offsets =
      connectivity<edges, vertices, Color>::offsets();
    constexpr auto ip0 = neighbors_offsets[0];
    constexpr auto ip1 = neighbors_offsets[1];

    float_type pos = math::max(eval(vn()), (float_type)0.);
    float_type neg = math::min(eval(vn()), (float_type)0.);

    eval(flux()) = eval(pos * pD(ip0) + neg * pD(ip1));
  }
};
Example of GridTools DSL: MPDATA

```c++

template <typename Color> struct upwind_flux {
    using flux = accessor<0, enumtype::inout, icosahedral_topology_t::edges>;
    using pD = in_accessor<1, icosahedral_topology_t::vertices, extent<0, 1, 0, 1>>;
    using vn = in_accessor<2, icosahedral_topology_t::edges, extent<0, 1, 0, 1>>;

typedef boost::mpl::vector<flux, pD, vn> arg_list;

template <typename Evaluation> static void Do(Evaluation &eval, k_full) {
    constexpr auto nlevels = icosahedral_topology_t::edges;
    constexpr auto nedges = icosahedral_topology_t::vertices;

    float_type pos = math::max(eval(vn)), (float_type)0.;
    float_type neg = math::min(eval(vn)), (float_type)0.;

    eval(flux()) = eval(pos * pD(ip0) + neg * pD(ip1));
}
```

```fortran

subroutine compute_upwind_flux(this, pflux, pD, pvH)
    type(MPDATA_type) , intent(out) :: this
    real(wp), intent(out) :: pflux(:, :, :)
    real(wp), intent(in)  :: pvH(:, :, :)
    integer :: nlevels
    integer :: nedges
    integer :: nlevels
    integer :: nedges
    call atlas_logdebug('compute_upwind_flux')
    nedges = thisxdimensions*nedges
    nlevels = thisxdimensions*nlevels
    ISGMR PARALLEL DO SCHEDULE(STATIC) PRIVATE(jedge, jlev, ip1, ip2, zpos, zneg)
        do jedge = 1, nedges
            ip1 = ledge2node(1, jedge)
            ip2 = ledge2node(2, jedge)
            do jlev = 1, nlevels
                zpos = max(0., wp, pvH(jlev, jedge))
                zneg = min(0., wp, pvH(jlev, jedge))
                pflux(jlev, jedge) = pD(jlev, ip1)*zpos + pD(jlev, ip2)*zneg
            enddo
        enddo
    ISGMR END PARALLEL DO
end subroutine compute_upwind_flux
```

Carlos Osuna, MPI Seminar, 21 November 2017
Composition of multiple MPDATA operators

```
m_upwind_fluxes = make_computation(gpu)(
    domain_uwf, grid_,
    make_multistage(execute<forward>(),
        make_stage<upwind_flux, octahedral_topology_t, edges>(
            p_flux(), p_pD(), p_vn()),
        make_stage<upwind_fluz, octahedral_topology_t, vertices>(
            p_fluz(), p_pD(), p_wn()),
        make_stage<fluxzdiv, octahedral_topology_t, vertices>(
            p_divVD(), p_flux(), p_fluz(),
            p_dual_volumes(), p_edges_sign()),
        make_stage<advance_solution, octahedral_topology_t, vertices>(
            p_pD(), p_divVD(), p_rho())));

Full MPDATA implemented using the GridTools DSL:
✓ upwind fluxes
✓ minmax limiter
✓ compute fluxes
✓ Limit fluxes
✓ Flux solution
✓ Rho correction
```
Grid Abstraction: Performance Portability

Thanks to abstraction of the mesh we can implement any partitioning:

- O32 mesh
- Equal partitioner implemented with Atlas
- Structured partitioner implemented with Atlas
Atlas: a library for NWP and climate modelling

Willem Deconinck

Deconinck et al., 2017
Grid Abstraction: Performance Portability

Abstraction of the mesh can implement any indexing layout

Indexing has a large impact in performance:

NVIDIA P100, 128x128x80, Bandwidth GB/s

| B = \((\sum_{neigh\ cells} A)\) | 
|---|---|
| SN DA | 211 |
| SN IA | 270 |
| UN IA | 130 |
| HN IA | 256 |
DSLs and abstractions: state of the art

GridTools and Atlas …
✓ solely relies on a C++ compiler → No external tools
✓ provides a high-level of abstraction → Performance portability

But writing in this new programming model…
✗ is hard (C++ expertise) → Decreased productivity
✗ is unsafe → No protection from violating the parallel model
✗ often requires expert knowledge to produce efficient code
What is the Future of GFD Models in a complex HPC environment?

1. Need to increase model development productivity
2. Need to decrease maintainence cost of models
3. Need to extend tools to a large set of models, methods, domains.
4. Need to maximize efficiency
DSL Toolchains for Performance Portable

Geophysical Fluid Dynamic Models
Need to increase model development productivity

High level DSLs, concise, safe and intuitive syntax that remove boiler-plate from embedded GridTools
Need to decrease maintainence of models

Check for parallel errors,

Auto-generation of boundary conditions and halo exchanges

Check for out of bounds

Safe generation of loop data dependencies

....
Need to extend tools to a large set of models, methods, domains.

Options:
- Unify all tools in a single DSL software stack -> maintainence issue.
- One DSL (programming model) per model, currently leading to explosion of DSL solutions.
- ESCAPE2
Need to extend tools to a large set of models, methods, domains.

Options:
- Unify all tools in a single DSL software stack -> maintainence issue.
- One DSL (programming model) per model, currently leading to explosion of DSL solutions.
- ESCAPE2
In the ESCAPE2 DNA...

- Develop programming languages for GFD models that can increase productivity, and lower maintainance.
- Standardization, avoid explosion of tools.
- Support multiple domains (FV weather, FV ocean, structured, unstructured, FE) in the standardization.
- Multiple domain languages or customizations, single tool.
- Language to be defined together with scientists.
ESCAPE2 base technology

Modular compiler toolchain (based on modern compilers –llvm- )
ESCAPE2 base technology

Multiple variants of a language

<table>
<thead>
<tr>
<th>FE plugin</th>
<th>IFS collocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICON Ocean plugin</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a[i+1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>on_cells(+,a)</td>
</tr>
</tbody>
</table>

Unstructured

Structured

GridTools4Py (Python)

clang DSL (C++)

CLAW DSL (Fortran)

Stencil IR

Stencil Checkers

Read before write

Data dependency race condition

Missing Update Boundary

DAWN
Example – Horizontal Diffusion

\[ \text{hd} = \nabla^4 u \]
\[ \text{lap} = \nabla^2 u \]
\[ \text{hd} = \nabla^2 \text{lap} \]
Example – Horizontal Diffusion

```c
stencil_function laplacian {
  storage phi;
  Do {
    return 4.0 * phi - phi[i+1] - phi[i-1] - phi[j+1] - phi[j-1];
  }
};

stencil hd_type2 {
  storage hd, in;
  temporary_storage lap;
  Do {
    vertical_region(k_start, k_end) {
      lap = laplacian(in);
      hd = laplacian(lap);
    }
  }
};
```
Safety First

```c
stencil hd_type2 { 
    storage u, utens;
    temporary_storage uavg;
    Do { 
        vertical_region(k_start, k_end) { 
            uavg = (u[i+1]+u[i-1])*0.5; 
            utens = (uavg[k+1]);
        }
    }
};
```
`stencil hd_type2 {`  
  `storage u, utens;`  
  `temporary_storage uavg;`  
  `Do {`  
  `vertical_region(k_start, k_end) {`  
  `uavg = (u[i+1]+u[i-1])*0.5;`  
  `utens = (uavg[j+1] + uavg[j-1]);`  
  `}`  
  `}`  
  `};`
Safety First

```c
stencil hd_type2 {
    storage u, utens;
    temporary_storage uavg;
    Do {
        vertical_region(k_start, k_end) {
            uavg = (u[i+1]+u[i-1])*0.5;
            utens = (uavg[j+1] + uavg[j-1]);
        }
    }
};
```

Access uninitialized data
Interoperability: Escaping the language

```c
stencil_function laplacian {
    storage phi;
    Do {
        return 4.0 * phi - phi[i+1] - phi[i-1] - phi[j+1] - phi[j-1];
    }
};

stencil hd_type2 {
    storage hd, in;
    temporary_storage lap;
    Do {
        vertical_region(k_start, k_end) {
            lap = laplacian(in);
            hd = laplacian(lap);
        }
    }
};

#pragma omp parallel for nowait
#pragma acc ...
for(int i=0; i < isize; ++i) {
    for(int j=0; j < jsize; ++j) {
        for(int k=0; k < ksize; ++k) {
            udiff(i,j,k) = hd(i+1,j,k) + hd(i-1,j,k);
        }
    }
}
```

DSL code

Standard C++ code (+OpenMP, +OpenACC, +CUDA)
Standardization: SIR

Multiple DSL (thin) frontends can generate standard SIR scheme

C++
GTCLANG
Python
GridToolsT4Py
Fortran
CLAW
ESCAPE2 / Ocean
Fortran
PsyClone

b = a(i+1) + a(i-1)
DAWN: Compiler optimization passes
COSMO dycore evaluation of DSL toolchain

Horizontal Diffusion (P100)

Advection (metric and time - P100 -)

<table>
<thead>
<tr>
<th>Version</th>
<th>Vert. + Hori. Advection</th>
<th>Combined Advection</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>gtclang</td>
<td>33</td>
<td>1.169 ms</td>
<td>27</td>
</tr>
<tr>
<td>gtclang + shared-memory opt.</td>
<td>33</td>
<td>1.169 ms</td>
<td>21</td>
</tr>
</tbody>
</table>
Conclusions

• The future of GFD models, high resolution, multidecade simulation will bring serious computational challenges: efficiency and production cost, maintenance cost
• We need to find right programming models so that adaptation to new architectures do not hinder scientific development -> high productivity
• Single efforts are not a valid model anymore, we need collaborations
• ESCAPE2 is a great opportunity to define the new language with scientists, we have the technology in place.
BACKUPS
<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>System</th>
<th>Cores</th>
<th>TFlop/s</th>
<th>TFlop/s (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Supercomputing Center in Wuxi, China</td>
<td>Sunway TaihuLight - Sunway MPP; Sunway SW26010 260C 1.65GHz, Sunway NRCPC</td>
<td>10,689,600</td>
<td>93,014.6</td>
<td>125,435.9, 15,271</td>
</tr>
<tr>
<td>2</td>
<td>National Super Computer Center in Guangzhou, China</td>
<td>Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.20GHz, TH Express-a, Intel Xeon Phi 3151P NUDT</td>
<td>3,120,000</td>
<td>33,862.7</td>
<td>54,902.4, 17,808</td>
</tr>
<tr>
<td>3</td>
<td>Swiss National Supercomputing Centre (CSCS), Switzerland</td>
<td>Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100 Cray Inc.</td>
<td>381,760</td>
<td>19,590.0</td>
<td>25,326.3, 2,272</td>
</tr>
<tr>
<td>4</td>
<td>Japan Agency for Marine-Earth Science and Technology, Japan</td>
<td>Gyukou - ZettaScalar-2.2 HPC system, Xeon D-1571 16C 1.39GHz, Infiniband EDR, PEZY-SC2 700MHz ExaScaler</td>
<td>19,060,000</td>
<td>19,135.0</td>
<td>28,192.0, 1,350</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/Oak Ridge National Laboratory, United States</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.20GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.</td>
<td>550,640</td>
<td>17,590.0</td>
<td>27,112.5, 8,209</td>
</tr>
<tr>
<td>6</td>
<td>DOE/NNSA/LLNL, United States</td>
<td>Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM</td>
<td>1,872,864</td>
<td>17,172.3</td>
<td>20,132.7, 7,890</td>
</tr>
<tr>
<td>7</td>
<td>DOE/NNSA/LANL/SNL, United States</td>
<td>Trinity - Cray XC40, Intel Xeon Phi 7250 68C 1.6GHz, Aries interconnect Cray Inc.</td>
<td>979,960</td>
<td>14,137.3</td>
<td>43,902.6, 3,844</td>
</tr>
<tr>
<td>8</td>
<td>DOE/SC/LBNL/NERSC, United States</td>
<td>Cori - Cray XC40, Intel Xeon Phi 7250 68C 1.6GHz, Aries interconnect Cray Inc.</td>
<td>622,336</td>
<td>10,014.7</td>
<td>27,880.7, 3,939</td>
</tr>
<tr>
<td>9</td>
<td>Joint Center for Advanced High Performance Computing, Japan</td>
<td>Oakforest-PACS - PRIMERGY CX1660 M1, Intel Xeon Phi 7250 68C 1.6GHz, Intel Omni-Path</td>
<td>556,104</td>
<td>13,554.6</td>
<td>24,913.5, 2,719</td>
</tr>
<tr>
<td>10</td>
<td>RIKEN Advanced Institute for Computational Science (AICS), Japan</td>
<td>K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu</td>
<td>705,024</td>
<td>10,510.0</td>
<td>11,280.4, 12,660</td>
</tr>
</tbody>
</table>