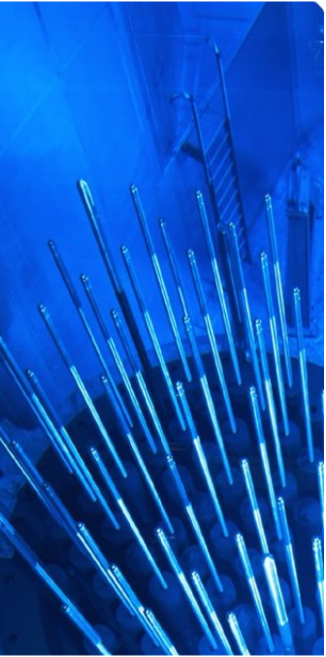
Parallel 3D Sweep Kernel with PaRSEC

Salli Moustafa
Mathieu Faverge
Laurent Plagne
Pierre Ramet

1st International Workshop on HPC-CFD in Energy/Transport Domains
August 22, 2014

Inria INVENTEURS DU MONDE NUMÉRIQUE
UNIVERSITÉ BORDEAUX 1 Sciences Technologies
EDF



Overview

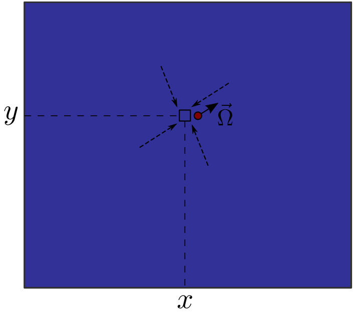
1. Cartesian Transport Sweep
2. Sweep on top of PaRSEC
3. Performances Studies
4. Conclusion

1. Cartesian Transport Sweep

The Boltzmann Transport Equation (BTE)

... describes the neutron flux inside a space region

We need to compute the neutron flux at the position (x, y), having energy E and traveling inside the direction $\vec{\Omega}$: $\psi(x, y, E, \vec{\Omega})$.

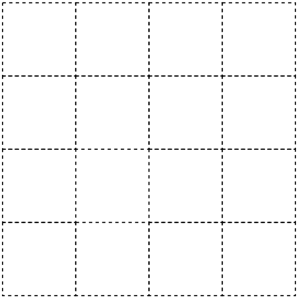


- BTE: balance between arrival and migration of neutrons at (x, y);
- Its resolution according to discrete ordinates (SN) method, involves a so-called Sweep operation consuming the vast majority of computation.
  - 10^{12} Degrees of Freedom (DoFs)

Solving the Boltzmann Transport Equation

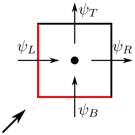
The Spatial Sweep Operation in 2D

- Discretization of the spatial domain into several cells



Solving the Boltzmann Transport Equation
The Spatial Sweep Operation in 2D

- Each spatial cell have 2 incoming dependencies (angular fluxes) per direction



$$\psi = \frac{\epsilon_x \psi_L + \epsilon_y \psi_B + \cdots}{\epsilon_x + \epsilon_y + \cdots}$$

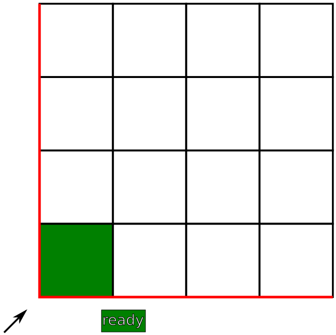
$$\psi_R = 2\psi - \psi_L$$

$$\psi_T = 2\psi - \psi_B$$

Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

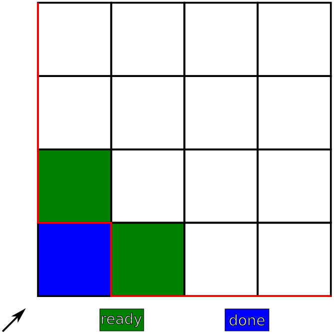
- At the beginning of the sweep, left and bottom fluxes are known;
- One cell ready to be processed.



Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

- The processing of the first cell sets incoming data for neighbouring cells;
- Two cells ready to be processed in parallel.

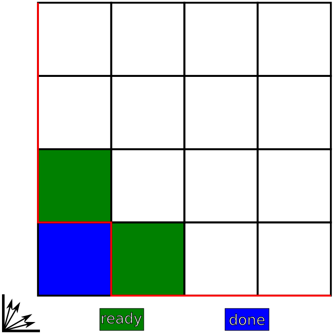


This process continues until all cells are processed for this direction...

Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

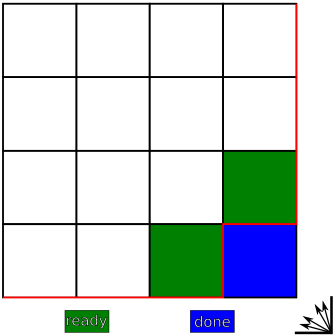
... for all directions belonging to a corner ...



Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

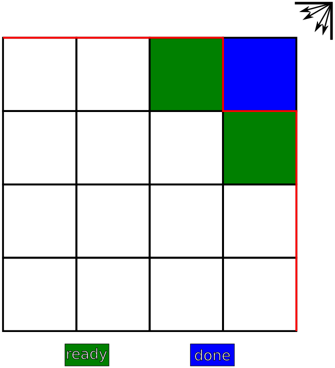
... and it is repeated for all the 4 corners.



Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

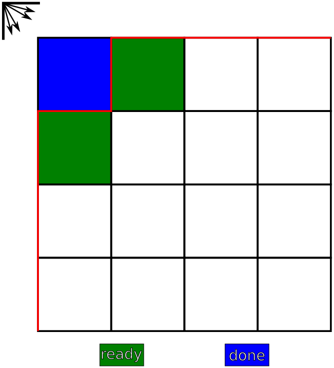
... and it is repeated for all the 4 corners.



Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

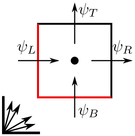
… and it is repeated for all the 4 corners.



Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

All directions belonging to a same quadrant can be processed in parallel.



$$\psi = \frac{\epsilon_x \psi_L + \epsilon_y \psi_B + \cdots}{\epsilon_x + \epsilon_y + \cdots}$$

$$\psi_R = 2\psi - \psi_L$$

$$\psi_T = 2\psi - \psi_B$$

Solving the Boltzmann Transport Equation

The Spatial Sweep Operation in 2D

2 levels of parallelism:

- spatial: several cells processed in parallel;
- angular: for each cell, several directions processed in parallel (SIMD).

2. Sweep on top of PaRSEC

The PaRSEC Framework

Parallel Runtime Scheduling and Execution Controller

The PaRSEC[1] runtime system is a generic data-flow engine supporting a task-based implementation and targeting distributed hybrid systems.

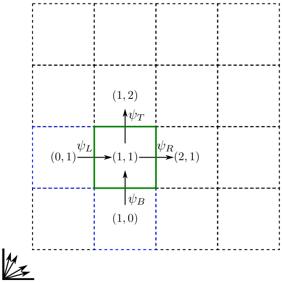
- An algorithm is represented in a graph whose nodes are tasks and edges data dependencies;
- Data distribution is specified through an API.

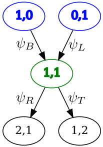
[1] George Bosilca et al. "DAGuE: A generic distributed DAG engine for High Performance Computing". In: *Parallel Computing* 38.1-2 (2012).

Implementation of the Sweep on top PaRSEC

Description of the DAG using the Job Data Flow (JDF) Representation

A task is defined as the processing of a cell for all directions of the same corner.

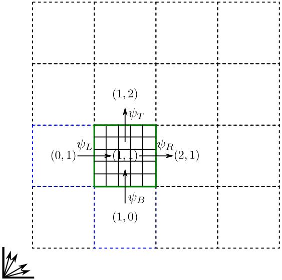


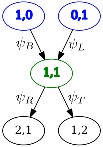


Implementation of the Sweep on top PaRSEC

Description of the DAG using the Job Data Flow (JDF) Representation

Grouping several cells into a `MacroCell` defines the granularity of the task.





Implementation of the Sweep on top PaRSEC

Description of the DAG using the Job Data Flow (JDF) Representation

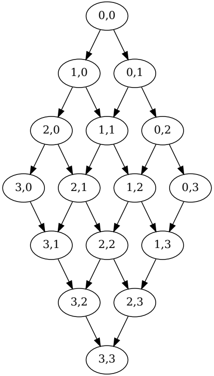
The whole DAG of the Sweep is described using the JDF symbolic representation:

```
T(a, b)
// Execution space
a = 0 .. 3
b = 0 .. 3

// Parallel partitioning
: mcg(a, b)

// Parameters
RW X    <- (a != 0) ? X  T(a-1, b)
        -> (a != 3) ? X  T(a+1, b)
RW Y    <- (b != 0) ? Y  T(b, b-1)
        -> (b != 3) ? Y  T(b, b+1)

RW MCG <- mcg(a, b)
        -> mcg(a, b)
BODY
{
  computePhi ( MCG, X, Y, ... );
}
END
```

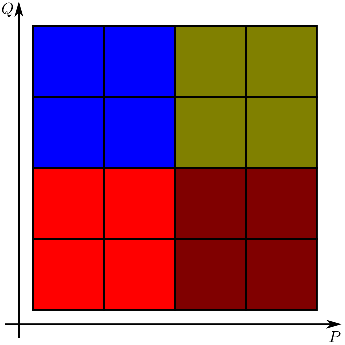


`computePhi()` is a call to a vectorized (over directions) kernel targeting CPUs.

Implementation of the Sweep on top PaRSEC

Data Distribution

We are using a blocked data distribution of cells;




```
int rank_of(int a, int b, ...){
  // Rank of the node containing cell (a,b)

  int lp = a / (ncx / P);
  int lq = b / (ncy / Q);

  return lq * P + lp;
}

void * data_of(int a, int b, ...){
  // Address of the cell object (a,b)

  int aa = a % (ncx / P);
  int bb = b % (ncy / Q)

  return &(mcg[bb][aa]);
}
```

$(P = 2, Q = 2)$ defines the process grid partitioning; Cells of the same color belong to the same node.

Implementation of the Sweep on top PaRSEC

Hybrid Implementation

At runtime:

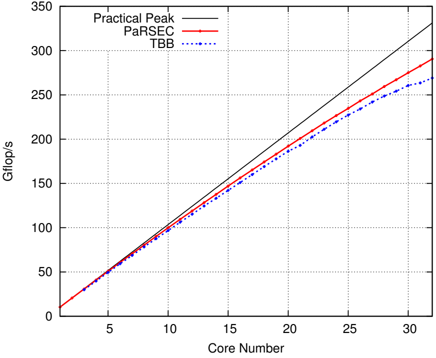
- Individual tasks are executed by threads;
- Data transfers between remote tasks: asynchronous MPI send/receive calls.

3. Performances Studies

Shared Memory Results

IVANOE/BIGMEM – 32 cores – Intel X7560

3D spatial mesh: $480 \times 480 \times 480$ cells; N_{dir}: 288 directions

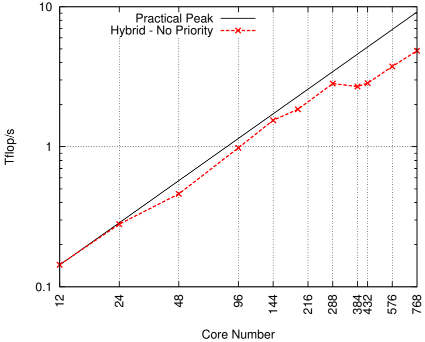


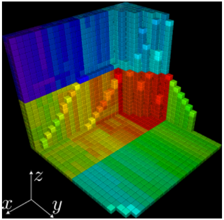
- PaRSEC implementation achieves 291 Gflop/s (51% of Theoretical Peak Perf.) at 32 cores; 8% faster than Intel TBB implementation.
- This is a sign of a reduced scheduling overhead for PaRSEC.

Distributed Memory Results – Hybrid

IVANOE – 768 cores (64 nodes of 12 cores) – Intel X7560

3D spatial mesh: $480 \times 480 \times 480$ cells; N_{dir}: 288 directions



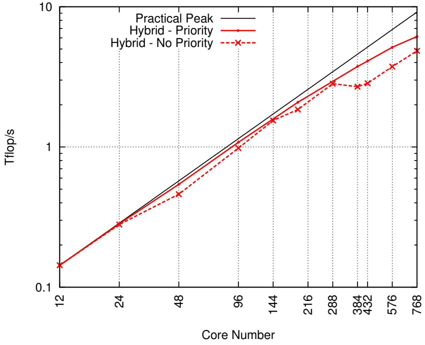


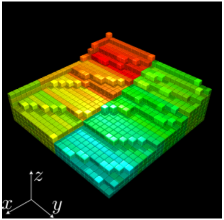
- Parallel efficiency: 52.7%
- 4.8 Tflop/s (26.8% of Theoretical Peak Perf.) at 768 cores

Distributed Memory Results – Hybrid

IVANOE – 768 cores (64 nodes of 12 cores) – Intel X7560

3D spatial mesh: $480 \times 480 \times 480$ cells; N_{dir}: 288 directions





- Parallel efficiency: 66.8%
- 6.2 Tflop/s (34.4% of Theoretical Peak Perf.) at 768 cores

Defined priorities allow to first execute tasks belonging to the same z plane: 26% of performance improvement.

Distributed Memory Results – Hybrid vs Flat

What is the best approach?

Hybrid approach:

- two-level view of the cluster;
- threads within nodes and MPI between nodes.

Classical Flat approach:

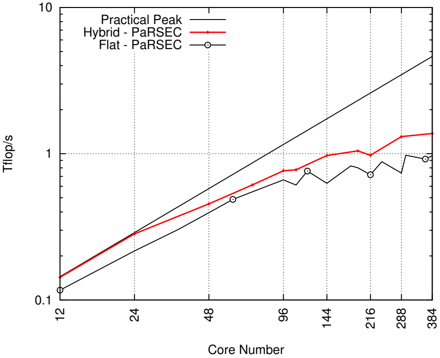
- one-level view of the cluster as a collection of computing cores;
- using only MPI.

Distributed Memory Results – Hybrid vs Flat

IVANOE – 384 cores – Intel X7560

3D spatial mesh: $120 \times 120 \times 120$ cells; N_{dir}: 288 directions

- Performance measurements for the Flat are also obtained using PaRSEC



- The Flat version involves much more MPI communications than the Hybrid one; consequently it is less performant (by 30% at 384 cores);
- A "hand-crafted" MPI version will it be more efficient?

4. Conclusion

A Building Block for a Massively Parallel SN Solver

We achieved:

- Parallel implementation of the Cartesian transport sweep on top of PaRSEC framework; a task-based programming model for distributed architectures;
- No performance penalty compared to Intel TBB in shared memory;
- 6.2 Tflop/s (34.4% of Theoretical Peak Perf.) at 768 cores of the IVANOE supercomputer.

Perspectives:

- Finishing theoretical models;
- Integrate this new implementation inside the DOMINO solver;
- Acceleration step: coupling with a distributed SPN solver;
- ...

Thank you for your attention! Questions

The Sweep Algorithm

The Sweep Operation

forall $o \in \mathtt{Octants}$ **do**
  forall $c \in \mathtt{Cells}$ **do**
    $\triangleright$ $c = (i, j, k)$
    forall $d \in \mathtt{Directions[o]}$ **do**
      $\triangleright$ $d = (\nu, \mu, \xi)$
      $\epsilon_x = \frac{2\nu}{\Delta x}; \quad \epsilon_y = \frac{2\eta}{\Delta y}; \quad \epsilon_z = \frac{2\xi}{\Delta z};$
      $\psi[o][c][d] = \frac{\epsilon_x \psi_L + \epsilon_y \psi_B + \epsilon_z \psi_F + S}{\epsilon_x + \epsilon_y + \epsilon_z + \Sigma_t};$
      $\psi_R[o][c][d] = 2\psi[o][c][d] - \psi_L[o][c][d];$
      $\psi_T[o][c][d] = 2\psi[o][c][d] - \psi_B[o][c][d];$
      $\psi_{BF}[o][c][d] = 2\psi[o][c][d] - \psi_F[o][c][d];$
      $\phi[k][j][i] = \phi[k][j][i] + \psi[o][c][d] * \omega[d];$
    end
  end
end

9 add or sub; 11 mul; 1 div (5 flops) $\longrightarrow$ 25 flops.

Gflops Evaluation

Gflops value is estimated by dividing the floating point operation number by the completion time:

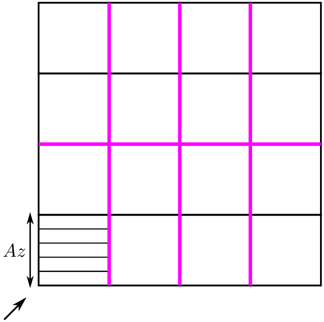
$$\text{GFlops} = \frac{25 \times N_{cells} \times N_{dir}}{\text{Time in nanoseconds}}.$$

The Critical Arithmetic Intensity Issue

$$i = \frac{\text{Number of floating points operations}}{\text{Number of RAM access (Read+Write)}}$$

Flat Approach

Data Distribution



- 4 nodes, each having 2 cores $\rightarrow$ 8 processors;
- $(P = 4, Q = 2)$ grid of processors;
- Az: number of planes to compute before comm.