

Data Management and Volcano Plume Simulation with Parallel SPH Method and Dynamic Halo Domains

Zhixuan Cao¹ Abani Patra¹ Matthew Jones²

¹Department of MAE
University at Buffalo, Buffalo, New York, U.S.A.

²Center for Computational Research
University at Buffalo, Buffalo, New York, U.S.A.

7th International Workshop on Advances in High-Performance
Computational Earth Sciences: Applications & Frameworks, 2017

Outline

- 1 Motivation of Choosing SPH
- 2 Overview
- 3 Data Structure and Load Balance
- 4 Dynamic Halo Domain
- 5 Numerical Test

Outline

- 1 Motivation of Choosing SPH
- 2 Overview
- 3 Data Structure and Load Balance
- 4 Dynamic Halo Domain
- 5 Numerical Test

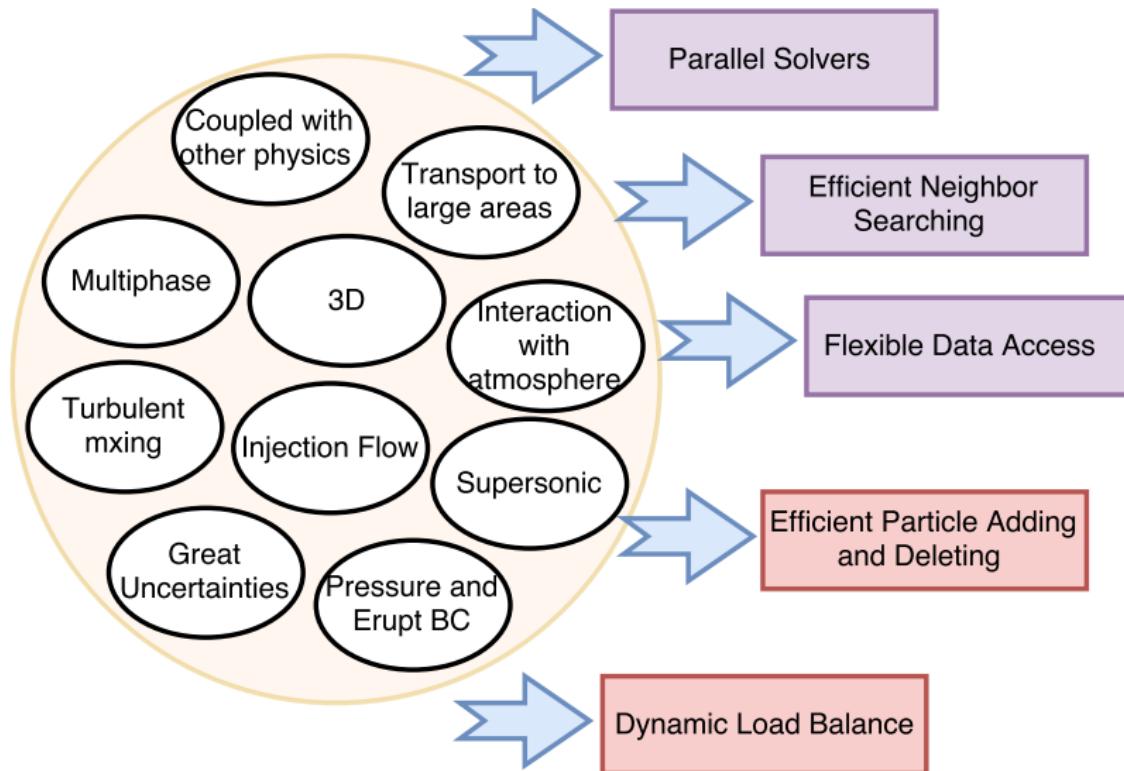
The development of volcano plume is essentially a multiple phase, turbulent mixing process coupled with heat transfer and other microphysics processes without pre-defined boundaries. SPH (Smoothed particle hydrodynamics), as a mesh-less method, is suitable for such problem for several reasons:

- SPH is able to automatically construct the interface.
- Multiphase is trivial for SPH
- Adding of new physics and new phases is easier in terms of programming in SPH
- With very limited global communication, SPH solvers are easier to scale up

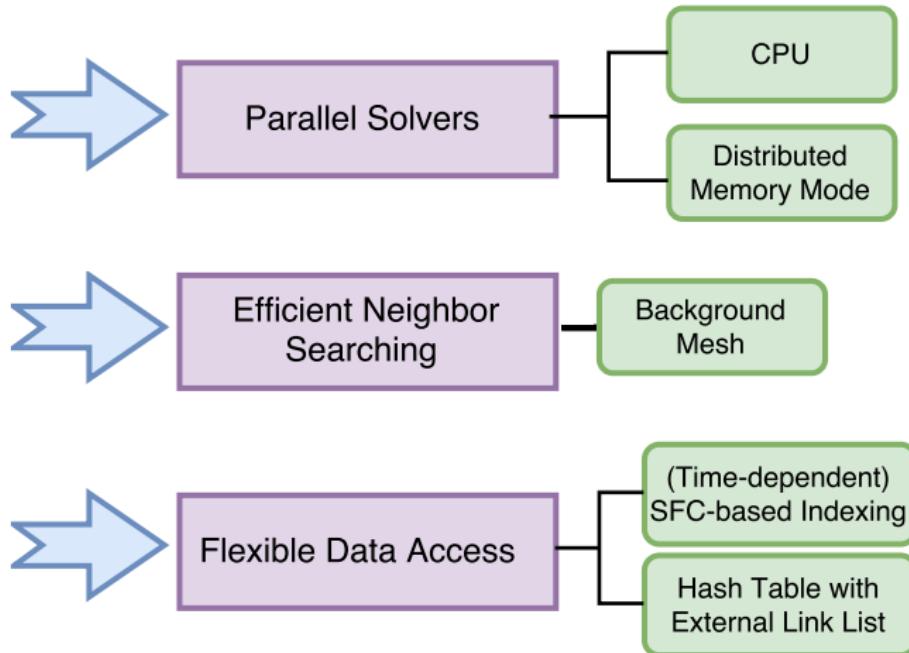
Outline

- 1 Motivation of Choosing SPH
- 2 Overview
- 3 Data Structure and Load Balance
- 4 Dynamic Halo Domain
- 5 Numerical Test

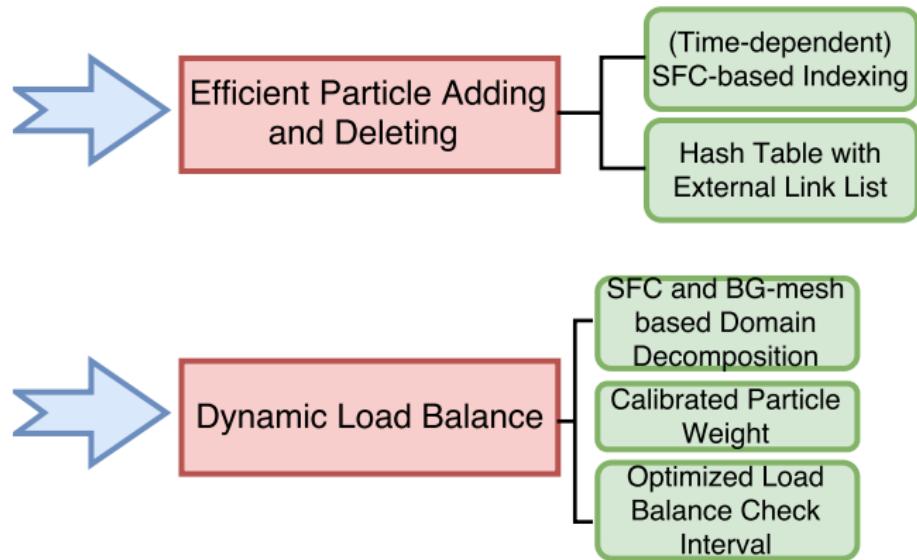
Requirements of the application



Our Strategies



Our Strategies



Outline

- 1 Motivation of Choosing SPH
- 2 Overview
- 3 Data Structure and Load Balance
- 4 Dynamic Halo Domain
- 5 Numerical Test

The most basic data structure of SPH are particles and buckets. Both are need as *C++* classes.

Particle Class

Information that is contained in particle objects can be categorized into six categories. Unique ID (key), affiliation, primitive variabls, secondary variables, flags and neighbor information.

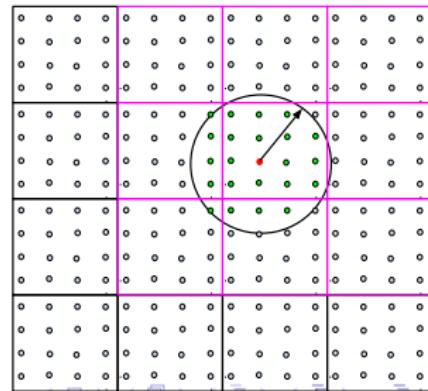
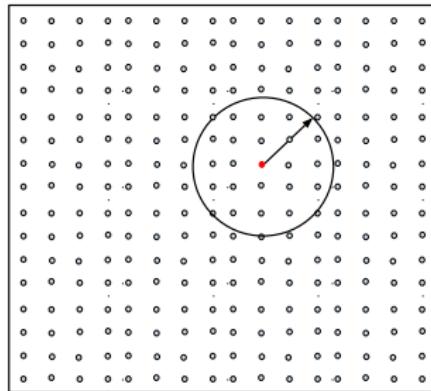
Bucket Class

Information that is contained in bucket objects can also be categorized into six categories. Unique ID (key), affiliation, dimension information, flags, neighbor information and contained particles.

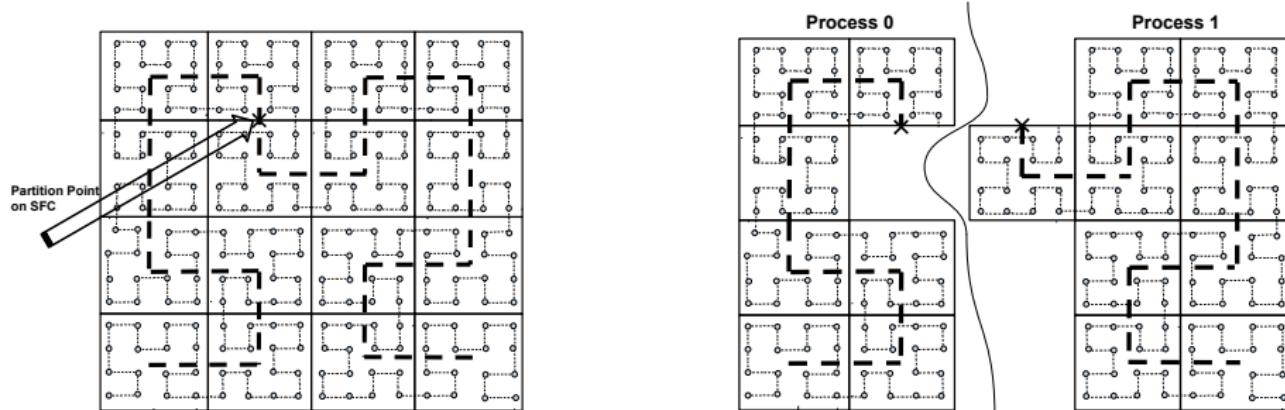
In SPH any function $A(\mathbf{x})$ and gradient of it can be approximated by.

$$\langle A(\mathbf{x}) \rangle \approx \sum_b m_b \frac{A_b}{\rho_b} w(\mathbf{x} - \mathbf{x}_b, h) \quad (1)$$

$$\langle \nabla A(\mathbf{x}) \rangle \approx \sum_b m_b \frac{A_b}{\rho_b} \nabla w(\mathbf{x} - \mathbf{x}_b, h) \quad (2)$$

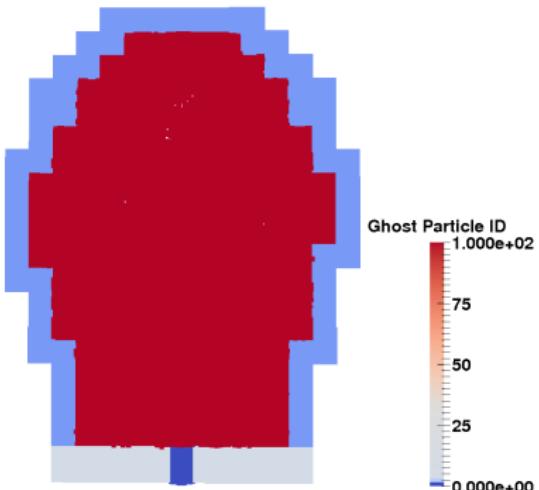


SFC based key is defined as: $k = h_n(\mathbf{x})$, Where $h_n(\mathbf{x}) : [0, 1]^n \rightarrow [0, 1]$

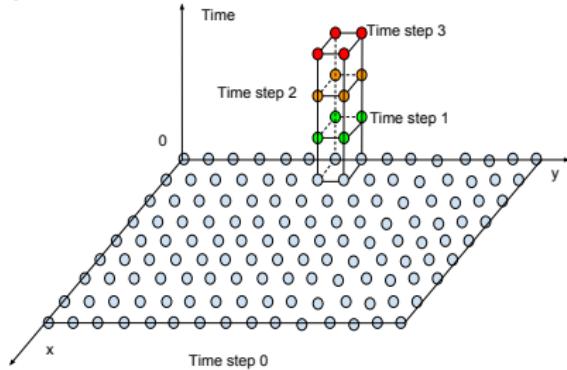


Features of SFC-based Index

- Guaranteed uniqueness of the indexing.
- Generating of index are fast and independent
- Conserve some locality in general
- It generate a global address space and can be used for domain decomposition



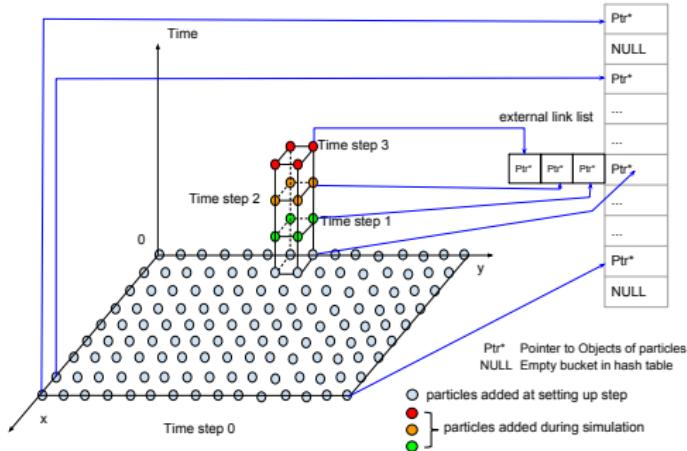
- particles added at setting up step
- particles added during simulation



Time-dependent SFC

$$h_n : [0, 1]^n \times \mathbf{T} \rightarrow [0, 1] \times \mathbf{N} \quad (3)$$

Where $\mathbf{T} \subset [0, \infty)$ is the time dimension, and $\mathbf{N} = \{0, 1, 2, 3, \dots\}$.
 Then the time-dependent key can be expressed as: (k, n)



Hash Function

$$\text{Index} = \frac{\text{Key} - \text{Min Key}}{\text{Max Key} - \text{Min Key}} \times \text{Hash Table Size} \quad (4)$$

To avoid a non-uniform sparse hash table, only plug the first number, k , of the key, into Eq. (4). All particles with the same birth location will hash to the same index and be handled by external linked lists.

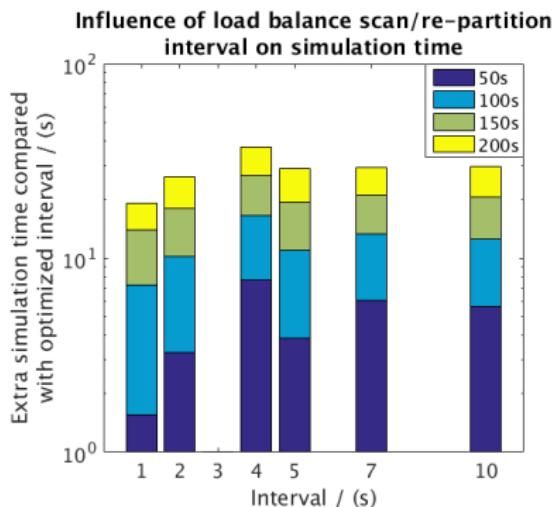
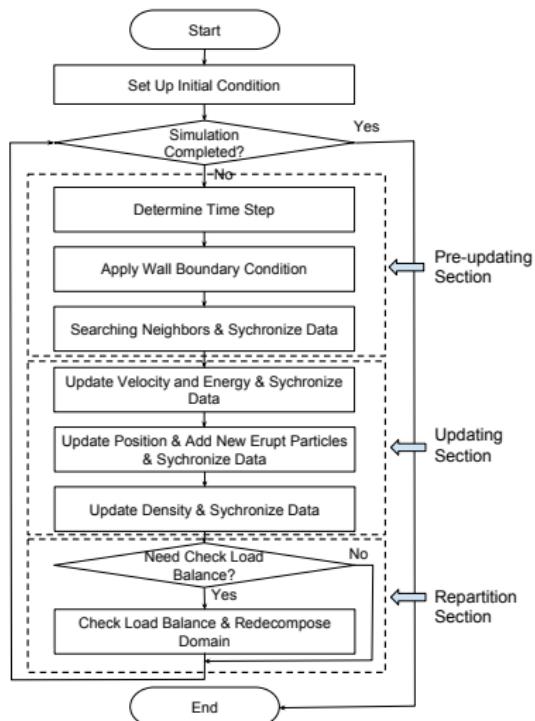
The domain is decomposed by cutting SFC of buckets into pieces of equal work load. The work load of each bucket is calculated by summing up weights of all particles contained by that bucket. The work load associated to each particle is calibrated based on profiling data.

Step	Cost (ms)	Real	wall	eruption	pressure
neighbor search	0.41	Yes	No	No	No
update momentum and energy	0.70	Yes	No	No	No
update density	0.42	Yes	No	No	No
update position	0.01	Yes	No	Yes	No
velocity filtering	0.43	Yes	No	No	No
apply wall boundary condition	0.75	No	Yes	No	No
summation (ms)	-	1.97	0.75	0.01	0.00

Compared with uniform particle weight, calibrated particle weights is computationally more efficient.

Physical time	10 s	20s	30 s	40 s
Same weight	1141.7	4119.4	10371.0	12453.7
Calibrated weights	1108.2	4057.0	10281.5	12166.3

The movement of particles and expansion of computational domain can lead to large load imbalance.



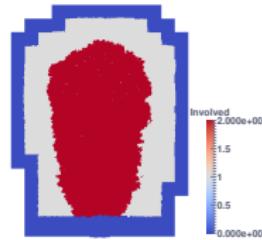
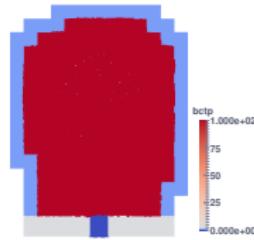
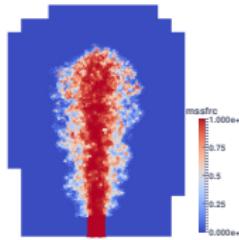
Outline

- 1 Motivation of Choosing SPH
- 2 Overview
- 3 Data Structure and Load Balance
- 4 Dynamic Halo Domain
- 5 Numerical Test

A lot of CPU time will be spent on computing associated with these stationary particles. The computational cost will be greatly reduced if the computational domain changes adaptively during simulation.

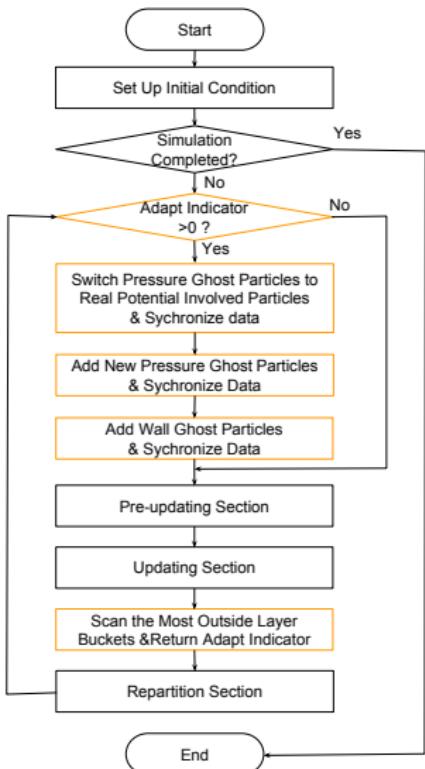
Our strategy

- An involvement flag to distinguish different states of involvement
- A scan function to monitor the outermost layer of the domain
- Shift pressure ghost particles into real particles when necessary

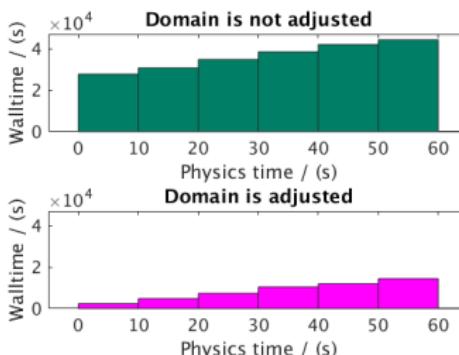


Dynamic Halo Domain

SWCH: switch pressure ghost particles to real. ADPP: add new pressure ghost particles. ADWP: add wall ghost particles. SCN: scann the outmost layer of the domain



Functions	Total time (s)	Called times
UPME	2954.8	201
UPP	38.55	201
ADPP	21.51	3
ADWP	8.88	3
SWCH	0.08	2
SCN	7.72	201



Outline

- 1 Motivation of Choosing SPH
- 2 Overview
- 3 Data Structure and Load Balance
- 4 Dynamic Halo Domain
- 5 Numerical Test

Scalability

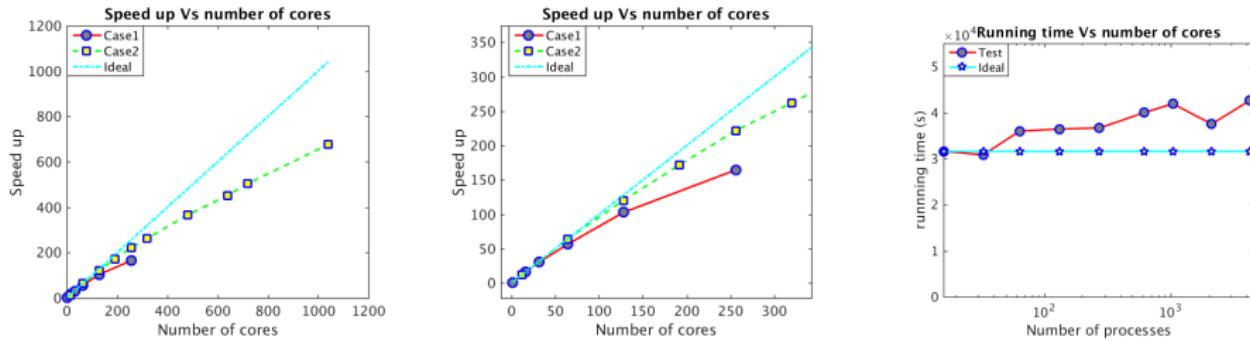
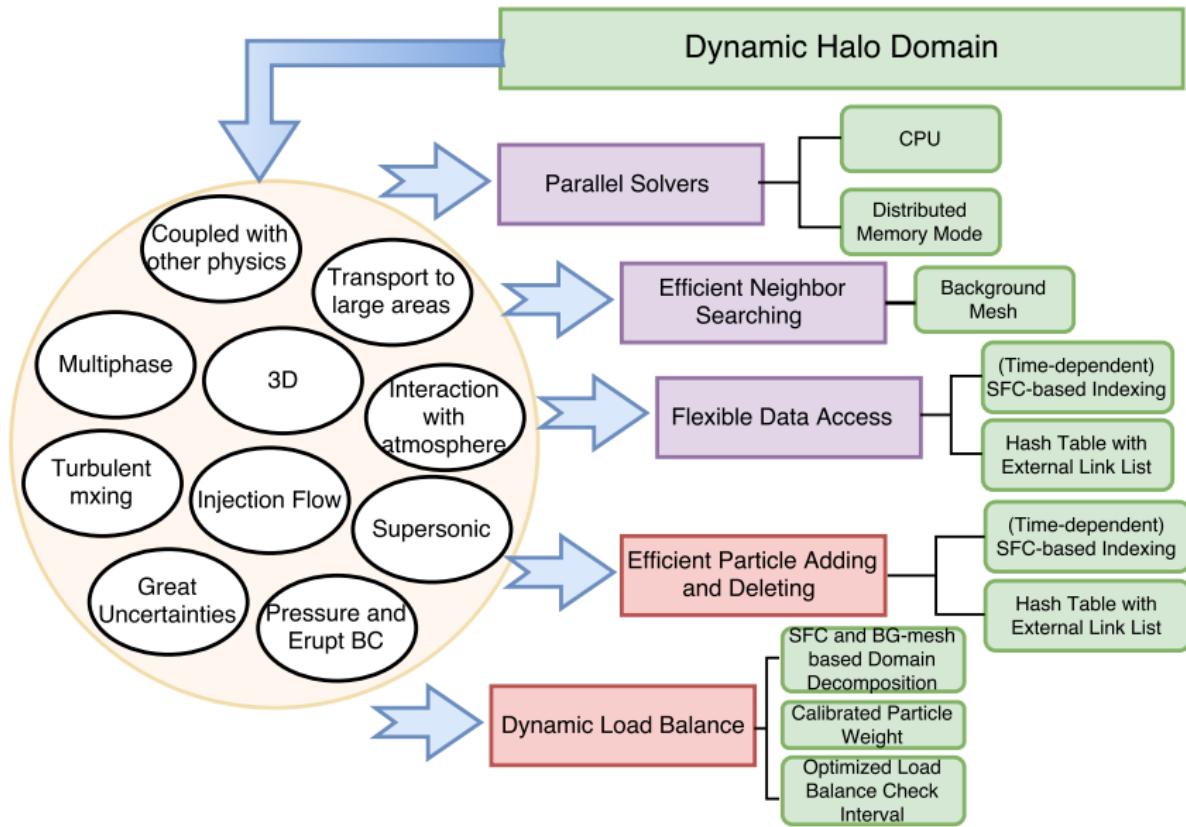


Figure: The left figure shows strong scalability tests result. middle figure is the zoomed view of first one. It is obviously shown that strong scalability is better when the problem size is larger. The right figure is weak scalability test results

A Typical Simulation Results

Requirements of the application



Summary

- We developed data management strategies for a MPI-parallel implementation of the SPH method to simulate volcanic plumes. Acceptable scalability was achieved.
- The flexibility of our data access methodology enables implementing mesh-free methods for solving more complicated problems and using more advanced techniques, such as dynamic particle splitting techniques.
- The data structure, particle and bucket indexing strategies, domain decomposition, dynamic load balancing method and domain adjusting strategies in this paper can be adopted to other mesh-free methods (not just SPH).
- Outlook
 - Better dynamic load balancing strategy.
 - Adaptive background grid and better domain decomposition algorithm.

Thank you!
Questions are welcome.