A scalable load-balancing algorithm for massively-parallel plasma particle simulations

Y. Miyake*¹, and H. Nakashima²

1. Graduate School of System Informatics, Kobe Univ., JAPAN
2. Academic Center for Computing and Media Studies, Kyoto Univ., JAPAN

(*Electronic address: y-miyake@eagle.kobe-u.ac.jp)

US-Japan JIFT Workshop 2011, Dec. 2
Plasma particle (PIC) simulations on supercomputer

- Huge number of plasma macro-particles
- EM-field on large number of computational grid points

Modern supercomputer
- Max. number of CPU cores > $10^5$
- Distributed memory systems
- Massively parallel computing
- Load-balancing issue
What’s the problem?

Inherently has much parallelism but believed hardly scalable because ...

- **Particle decomposition** copying fields cannot sustain large space domain and global operations on it.
- **Domain decomposition** cannot work when particles are distributed non-uniformly.
- **Dynamic domain decomposition** also has pitfalls for particles populating a small subdomain too densely.

→ Need another approach!
If particles distributed uniformly…

Primary subdomain assignment

- **uniform** block decomposition
  - simulate primary particles
  - neighboring comm. *Only*

- well-balanced condition:
  \[
  \forall \#\text{particle-in-subdomain} \leq P/N(1 + \alpha),
  \]
  where
  \[
  P: \text{total particle number},
  \]
  \[
  N: \text{number of computation nodes}
  \]
  \[
  \alpha: \text{tolerance factor}
  \]

Node IDs \equiv Subdomain IDs

⇒ **Primary mode**
**OhHelp – an advanced load-balancer: Overview**

If particles distributed **non-uniformly**…

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
</tr>
</tbody>
</table>

**Secondary subdomain assignment**

- Each node helps another node having particle-dense subdomain
  - Ex) Node 33 helps node 22, by sharing particles inside SD-ID: 22.
  - (Both nodes 22 and 33 compute the field of SD-ID: 22 redundantly).

- Balanced #particles
- Balanced subdomain size
- Well-balanced
  - Stable subdomain assignment

**Secondary mode**
Secondary subdomain assignment

move p from heaviest to lightest so that lightest has av. #p

give p even if becoming less than average ➔ get from somebody afterward

• Computational cost in terms of #p completely balanced
• Comp. cost in terms of domain size (~2D/N) almost balanced
Transition between primary and secondary modes

For every simulation time step...

1. create particle histogram for each SD
2a. if well-balanced for all SDs then move to P-mode
2b. else if well-balanced condition can be satisfied keeping the current secondary SD assignment then stay at S-mode
2c. else rearrange secondary SD assignment
3. transfer particles

SD: subdomain, P-mode: primary mode, S-mode: secondary mode
Checking and keeping local balancing

**Helper-Tree** is traversed each time-step (i.e., each particle movement)

- **bottom-up:** does helper assignment sustain the load variance?
- **top-down:** how (re)distribute particles among family members?
OhHelp Library
(developed by H. Nakashima)
... helps people struggling with parallel PIC simulations.

3 library layers
Level 1: particle transfer scheduling based on the OhHelp scheme
Level 2: (level 1 +) particle transfer execution
Level 3: (level 1 + level 2 +) inter-node communications

Useful API functions
help users to implement above tasks onto their own codes
ex) function oh_transbound: for load-balancing and particle transfer
function oh_exchange_borders: for boundary comm. of EB (& current)-fields
function oh_allreduce_field: for summation of current (& charge) density among Pri. and Sec. subdomains
function oh_bcast_field: for broadcast from Pri. to Sec. subdomains
Applying OhHelp to PIC simulators: Data structure

**Duplication of field data structure**

- `real*8 :: Ex(1:Nx,1:Ny,1:Nz), etc…`
- `real*8 :: EB(6, 1:Nx, 1:Ny, 1:Nz, 2)`

**Particle data structure**

```fortran
  type particle
    real*8 :: x, y, z, vx, vy, vz
    integer :: nid, spec
  end type

  type(particle) :: pbuf(npmax)
```

- `npmax > (1 + α)(P / N)`
- Particle buffer as 1D array of structures

**Node 03**

Duplication for primary & secondary subdomains
Applying OhHelp: Computation procedure

initialization

particle push

Current computation

Field solve

Secondary p

Primary p

\[ \frac{m}{dt} = q(E + v \times B) \]

\[ \frac{dx}{dt} = v \]

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{J} = 0 \]

\[ \nabla \times \mathbf{E} = \frac{\partial \mathbf{B}}{\partial t} \]

Load balancing & Particle transfer

oh_init

oh_allreduce_field

oh_broadcast_field

oh_transbound

oh_exchange_borders
Performance evaluation: Simulation setup

- **weak scaling:** 1 ~ 1024 processes
  - Setting 1: uniform particle distribution
    - subdomain size: 32x32x32
    - Number of processes: 1
  - Setting 2: extremely non-uniform particle distribution
    - subdomain size: 32x32x32
    - Number of processes: 1024

- **T2K open-supercomputer@Kyoto university**
  - CPU: Quad Core AMD Opteron
  - Number of CPU (cores) on node: 4 (16)
  - Main memory on node: DDR2-667 32GB
  - Total number of nodes: 416
Performance evaluation: Results

**Full particle simulation**

- uniform: $721^*$
- non-uniform: $626^*$

**Particle-fluid hybrid simulation**

- uniform: $1026^*$
- non-uniform: $936^*$

*Speedup of the OhHelp’ed simulations compared with serial executions of not-OhHelp’ed simulators*

- domain size: $D = 32^3 \times \#\text{proc}$
- particles / cell (P/D) = 256 (full particle), 240 (hybrid)

Good scalability for both uniform & non-uniform cases
Applications of “OhHelp’ed” PIC simulator

- Spacecraft-plasma interactions (full particle simulation)
- Magnetic field structure associated with Mirror instability (hybrid simulation) [Shoji and Omura, JGR]
- Cold ion flow
- Proton density
- 100 ~ 103-fold parallelization, 200 GB ~ 1.5 TB problem size

Diagram showing spacecraft wake and proton density.
Conclusion

OhHelp: a load balancer for particle simulations

- uniform block decomposition realizing simple MPI-comm.
- each node helps at most one other node → \( #p \) balanced
- not-frequent rearrangement of secondary SD assignment
- good scalability confirmed up to \( 10^3 \)-fold parallelization

OhHelp Library (currently, version 0.9.5) helps people applying OhHelp to their own code

- useful, and optimized API functions provided
- available at [http://www.para.media.kyoto-u.ac.jp/ohhelp/]
This research work is proceeded by a **tight collaboration** between “**Computational scientists**” and “**Computer scientists**”.

Joint paper related to OhHelp:

**Thank you for your attention!**
Backup
### Discussion: Scalability comparison

N: number of nodes, P: total particle number, D: whole domain size

<table>
<thead>
<tr>
<th>Approach</th>
<th>Particle number (Ideal : Worst)</th>
<th>Domain size (Ideal : Worst)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle decomposition</td>
<td>P/N : P/N</td>
<td>D : D</td>
</tr>
<tr>
<td>Domain decomp. (static)</td>
<td>P/N : P</td>
<td>D/N : D/N</td>
</tr>
<tr>
<td>Domain decomp. (dynamic)</td>
<td>P/N : P/N</td>
<td>D/N : ~D</td>
</tr>
<tr>
<td>OhHelp</td>
<td>P/N : (1+α) (P/N)</td>
<td>D/N : 2D/N</td>
</tr>
</tbody>
</table>
Breakdown

- **exec. time/step**
  - balanced
  - unbalanced

- **comm. time/step**
  - balanced
  - unbalanced

<table>
<thead>
<tr>
<th>Component</th>
<th>balanced</th>
<th>unbalanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>comm (part)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>part. packing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>comm (hgram)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>comm (field)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>comm (curr)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>particle</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Graph showing time breakdown with categories and values for different components.]
Checking satisfiability of local balancing by the tree

- **examination passes** if one can reach the root node without failing all aforementioned tests

- **responsible for all P-particles** that cannot be delegated to its helpers – if impossible, **test failed**

- **assume to take S-particles as many as possible up to its limit**

- **responsible for all P-particles** – if impossible, **test failed**

- **assume to take S-particles as many as possible up to its limit, where “limit” = (P/N)(1 + α)**