Postdoc Fellowships for non-EU researchers

Final Report

Name	Wei Jiang
Selection	2012
Host institution	KULeuven
Supervisor	Giovanni Lapenta
Period covered by this report	from 22/10/2013 to 21/10/2014
Title	Comprehensive acceleration of implicit particle-in-cell code in hybrid architecture

1. Objectives of the Fellowship (1/2 page)

This research will focused on GPU/MIC acceleration for implicit PIC code and improving performance on massively parallel computers using hybrid (CPU-GPU/MIC) architecture, especially on the algorithms, data structures and program realization, which will include:

(1) Porting the implicit moment PIC algorithm to GPUs and to the new Intel MICs, this work will be based on the present iPIC3D code. The acceleration rate of GPU/MIC over CPU is expected larger than 5.

(2) Studying and improving performance on massively parallel computers using hybrid (CPU-MIC) architecture, this work will focus on the I/O part of the iPIC3D.

2. Methodology in a nutshell (1/2/ page)

2.1 GPU/MIC Acceleration

This work is based on iPIC3D, and PIC code based on new more advanced methods. Unlike the explicit scheme, implicit schemes are much more complex and must be treated carefully when porting to GPU/MIC. Some key issues need to be addressed:

(1) Memory arrangements are the most important issues, there are many distinct kinds of memories in GPU/MIC, register, L1/L2 cache, global memory and expansion memory. All have very different size, speed, etc. One must carefully coordinate these memories to increase the total efficiency.

(2) Particle sorting: due to large space and temporal steps in implicit schemes, more particles will cross the cell boundaries in one step. The particles resorting and exchanging will severely consume computation time.

We designed several different data structure and algorithms first, and benchmarks these different schemes to find out the optimum one. Atomic operation are also used to speed up the code.

2.2 Parallel computation on hybrid architecture.

We worked on studying and improving performance on massively parallel computers using hybrid (CPU-MIC) architecture. GPU/MIC, OpenMP and MPI will incorporated into a uniform framework.

The key issue is the load balancing between inter-node, intra-node and hardware accelerators. The highest level MPI is only used to allocate the work between inter-node. The medium level OpenMP is used allocate the work between cores of the CPUs between intra-node, and most parameters on grids are calculated and exchanged in CPUs. And the lowest level GPU/MIC is used more for updating the particles, not for allocating the work. The load balancing is first achieved with MPI, then OpenMP and finally the GPU/MICs.

All works is based on the present iPIC3D code, the code will be written in C++ language, many different data structure and algorithms are designed, benchmarked and compared to find the best solution.

3. Results (6-8 pages)

3.1 Introduction

As well known, PIC is a particle-based method and the PARTICLES and FIELDS are two fundamental classes of software structures in any PIC code. Particles are defined in the phase space continuously and consume most of the storage and the computation resources in most cases, while field are defined on the grids discretely and cost less resources. They are connected by the Particle-to-Cell and Cell-to-Particle interpolation.

PIC code runs in an iterative way, from the proper initial values, the information of particles and fields are updated alternately on steps (in full implicit cases they are updated simultaneously). PIC code can be restarted continuously if all the fundamental information of particles and fields are stored and loaded properly. What information is fundamental is related to the specific scheme. In most cases, all phase space of particles and at least electric or magnetic fields are fundamental.

Generally, a PIC code like iPIC3D used in our work contains at least four parts:

(1) Initialization and finalization, which gives the proper value, physically or numerically to perform the simulations, and also set the criterions to end;

(2) Execution, which includes the field solver, particle mover, Particle-to-Cell (P2C) or Cell-to-Particle (C2P) interpolation, as well as the cycles controls the program flow;

(3) Diagnostics, which gives the physical results of the simulations;

(4) File I/O, which covert the numerical data used to temporally backup and restart the simulations, or diagnostics as the output results.

File I/O is one of essential part of PIC code, which is used both for diagnostics and check-point to restart the program. Here we give attention to efficiency and scaling of I/O and check pointing in CPU-MIC hybrid architecture. We therefore have measured the time required for iPic3D to write field data and particle data.

In the this project, we are changing from the old output format, in which each process appends output to its own procPP.h5 file, where PP is the process rank, to a new output format, where in each time step parallel HDF5 is used to save all particles to a single particles_NNNN.h5 file and a single fields_NNNN.h5 file, where NNNN is the number of the time step.

The motivation for the new output format is to streamline post-processing and visualization. With the old output format, we essentially have to run a post-processing tool which converts data from the old format to something like the new format. Specifically, the postprocessor must walk through all output files, extract the data for each output time for which we want to display output, and write the collected (and possibly reduced or transformed) data in something like the new output format. With the new output format, output can be directly viewed. No knowledge of the process topology used to generate it is needed.

3.2 Benchmarking results

We performed weak and strong scaling studies for writing output, based on varying one or another parameter in the reference study.

We first performed weak scaling on a single Xeon or MIC node, using the KNC systems at JSC, varying the number of processes. Since the Xeon has only 16 cores, using more than 16 processes overloads the Xeon. The results are displayed in Figure 1.

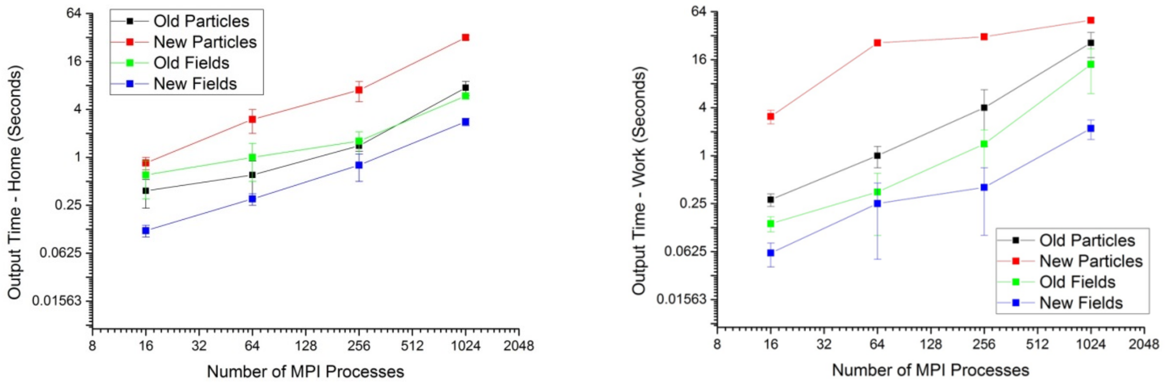

Figure 1: Time to write particles (Left) and fields (Right) in Xeon vs. Xeon Phi. Cells per process: 15x15x1. Particles per cell: 8x8x8x2

We then performed a weak scaling benchmark on the DEEP Cluster, increasing number of nodes, using a fixed value of 16=4x4 processes per node. In this and all subsequent studies, we reduced the number of particles per mesh cell from 1024 to 128, which is a more typical number for a PIC simulation. Figure 2 shows the results using the home and work filesystem. Since the home filesystem is NFS, which is not expected to scale well, we decided to try a similar scaling simulation on the work filesystem, which is a BeeGFS parallel filesystem,.

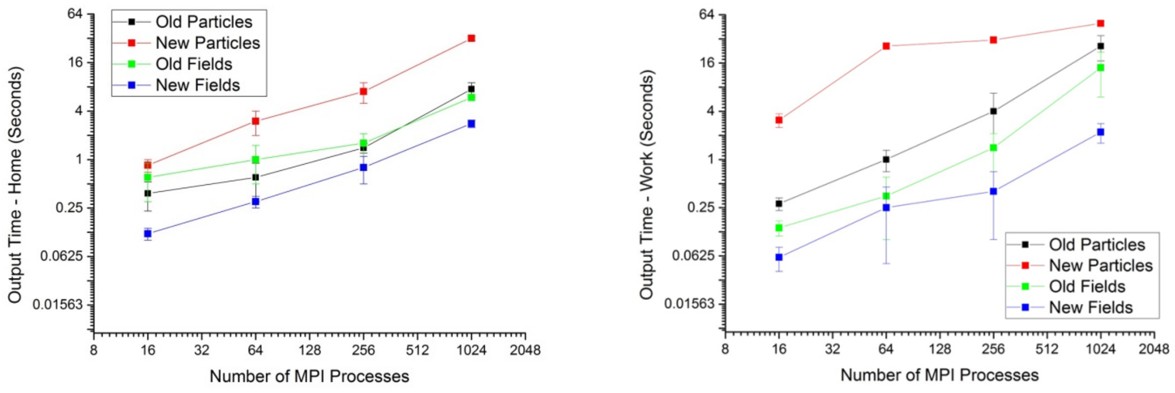

Figure 2: Time to write particles and fields in the home (Left) and work (Right) filesystem of the DEEP Cluster. Cells per process: 15x15x1. Particles per cell: 4x4x4x2. Process per node: 16.

These studies were done on the DEEP Cluster. We use the system designation deep:home to mean that the study was done in the user's home directory, and the designation deep:work to mean that the study was done in the deep:/work directory. The old output format is trivially parallel, since each process writes to its own file, and therefore we felt compelled to attribute the lack of weak scaling of output time to bandwidth limitations in the deep filesystem itself, which, unlike the planned DEEP-ER system, was not designed with I/O scaling specifically in mind. We therefore sought to repeat the study on other systems where we expected better scaling of the underlying I/O capabilities of the filesystem and processors.

In order to perform larger experiments with more capable I/O hardware, we turned to Stampede, briefly described in Annex A. Stampede has three file systems: home, work, and a high speed file system called scratch. The larger size of this system and steadier user demand allowed us to obtain more consistent and thorough results. Ignoring high-end and also low-end outliers, times seen are shown in Figure 3.

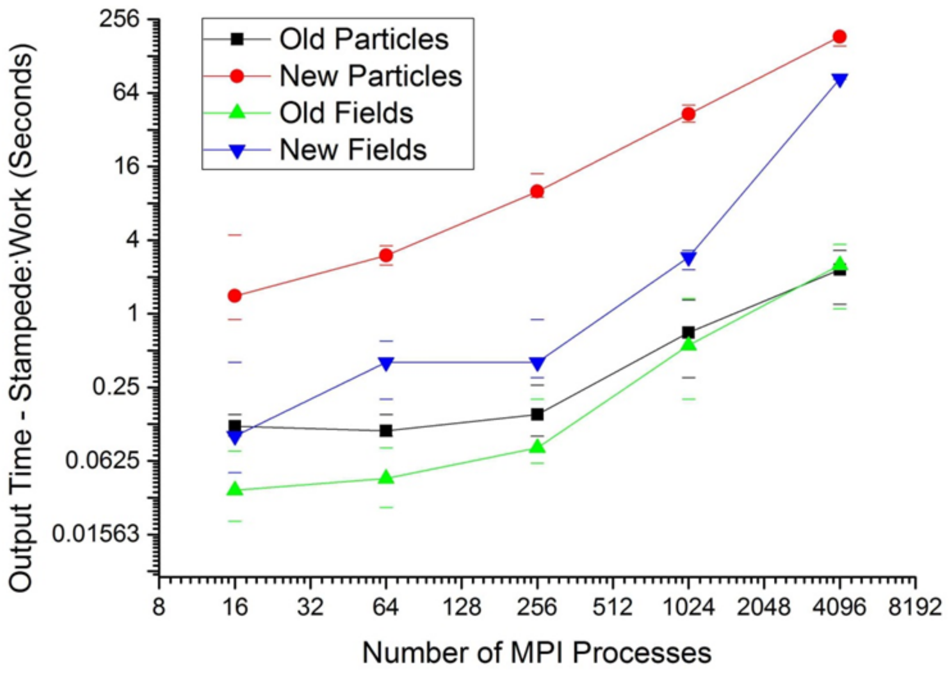

Figure 3: Time to write particles and fields in the work filesystem of Stampede. Cells per process: 15x15x12. Particles per cell: 4x4x4x2. Process per node: 16.

This shows good scaling of the old output format, but very bad scaling for the new format. We wanted to know whether the problem was with H5hut or with parallel HDF5 itself. So we saved the fields directly with parallel HDF5 and timed the output. On stampede:scratch we observed that the stripe count per file was only 2, regardless of how many processes were used. We therefore experimented with increasing the stripe count. We found that increasing the stripe count from 2 to 8 improved the initial scaling, as seen in Figure 4.

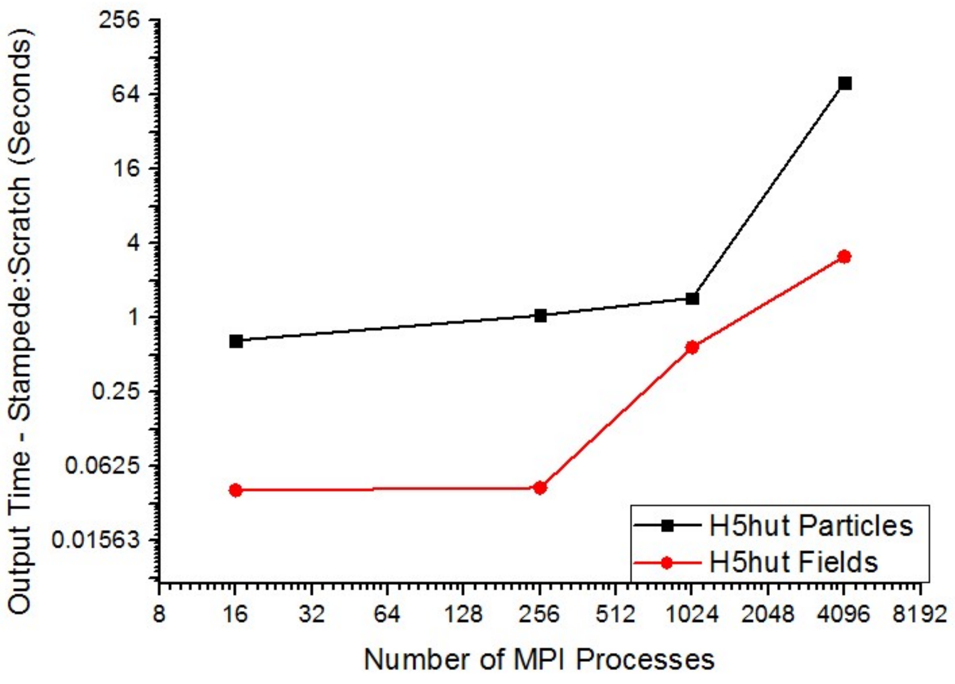

Figure 4: Time to write particles and fields in the scratch filesystem of Stampede, with stripe count equals to 8. Cells per process: 16x16x12. Particles per cell: 4x4x4x2. Process per node: 16.

We observed that the output was not scaling to 4096 processes, so we experimented with increasing the stripe count and changing the stripe length. The results are shown in Figure 5.

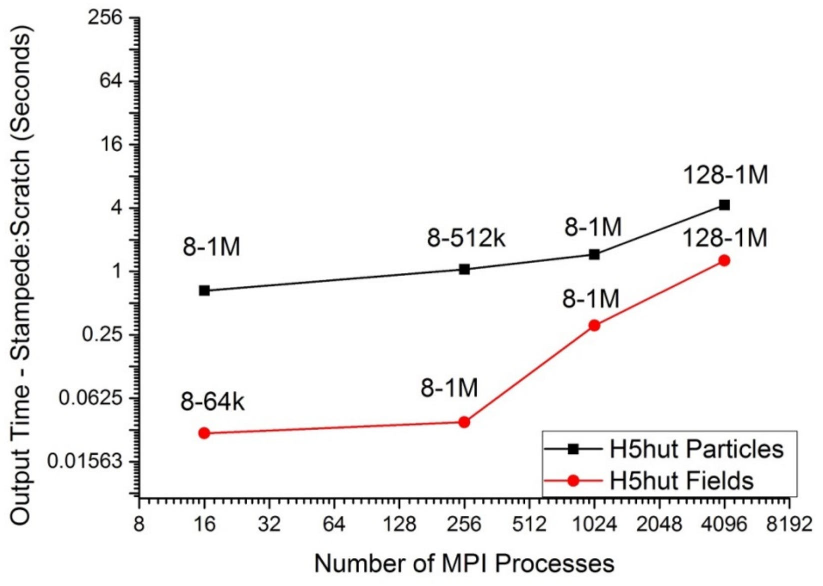

Figure 5: Time to write particles and fields in the scratch filesystem of Stampede, with various stripe counts and lengths. Cells per process: 16x16x12. Particles per cell: 4x4x4x2. Process per node: 16. Striping: 8-64k means 8 stripes of length 64KB.

In these studies the striping is chosen to be optimal for particles. The optimal striping for fields was not determined. We observe that with a large stripe count output times scale reasonably to 4096 processors.

3.3 Analysis of I/O performance

The foregoing studies have identified performance issues with parallel output. The goal of the following analysis is to identify possible reasons for the deficient performance that we have seen and to anticipate performance issues that we are likely to encounter with post-processing of data.

3.3.1 Tuning parallel I/O

I/O performance analysis begins with an understanding of the filesystems involved. For concreteness, we describe stampede:scratch, a Lustre parallel filesystem. This system has 348 "Output Storage Targets" ("OSTs"), i.e. storage disks, handled by a smaller number of "Output Storage Servers" ("OSSs"). Each file is segmented into consecutive segments called "stripes", and each stripe resides in a single OST. Each file has a "stripe length" (the number of bytes per stripe, by default 1 megabyte and constrained to be a multiple of 64 kilobytes) and a "stripe count" (the number of OSTs to which consecutive stripes are written in parallel, a number at most 160 and by default 2).

The other major component of I/O performance analysis involves how process memory is mapped to the filesystem in terms of static layout and how the data transfer is parallelised.

3.3.2 Data transfer and buffering

In parallel HDF5, output can be written independently or collectively. With independent output, each process independently writes to the appropriate segments of the output file. With collective output, output destined for specific file segments is gathered from multiple processes by an intermediate buffering agent. Collective I/O can greatly improve efficiency, but the current implementation of HDF5 reverts to independent I/O when the user attempts to use collective I/O unless the output of each process satisfies restrictive similarity constraints.

An essential concern when doing I/O is to minimize non-contiguous access. Filesystems naturally transfer data most efficiently in contiguous segments, and non-contiguous access by a single process to a file can result in much unnecessary transfer of data. The technique of data sieving attempts to ameliorate this problem by using an intermediate sieving buffer. Tuning the size of the sieving buffer can improve independent I/O.

In the case of a parallel filesystem, there is a second issue: simultaneous access by multiple processes to the same contiguous region in a filesystem can result in contention. Collective buffering attempts to ameliorate this problem by combining output from multiple processes destined for the same data segment before writing. Tuning the size of the collective buffer can improve collective I/O.

3.3.3 Data layout

Saving particle and grid data requires that we serialize it. A major question is how to serialize our data in a way that achieves good output performance on the parallel filesystem that we are using. Since grid data lives in a 3-dimensional space and particles live in a 6-dimensional phase space, it is not obvious how best to serialize our data.

A strategy to avoid non-contiguous access is to attempt as much as possible to store the data corresponding to each process in its own contiguous memory. In particular, it is desirable that data residing in each process's subdomain be saved to a contiguous region of the file and that storage accessed simultaneously from different processes resides on different physical devices.

3.3.4 Efficiency of reading

Regardless of whether we are dealing with particles or fields, if data is written contiguously then reading the data will be contiguous only if the process topology is restricted. In particular, given that data maps contiguously onto the process topology used in writing (and no other information), data can be guaranteed to map contiguously onto the process topology used in reading only if its dimensions divide the dimensions of the process topology used in writing.

Fields and particles present different types of challenges in optimizing output performance, so we now consider them separately. Note that we save particles and fields to separate files; this will allow us to optimize the choices of striping and other output parameters separately for particle and field output.

Particles present two challenges in comparison to fields: (1) the amount of data is much greater and (2) the amount of data per process is not a known, fixed constant. Writing particle data to a single file therefore requires an initial global communication (e.g. via MPI_Allgather) to determine the segment to which each process must write. Also, some kind of non-fixed padding would be necessary to ensure that the segment of particle data written by each process is aligned with stripe boundaries.

We conclude this section with a summary of the main parameters involved in tuning I/O:

1. Parallel file system parameters:

a) Striping unit: bytes per stripe.

b) Striping factor (i.e. stripe count): number of devices to stripe across (-1 means stripe across all devices).

2. MPI-IO parameters for non-contiguous data access (subset of ROMIO parameters):

a) Data-sieving parameters (relevant even to non-contiguous access from a single process).

i. ind_wr_buffer_size, ind_rd_buffer_size: size of intermediate sieving buffer for writing and reading.

b) Collective buffering (two-phase I/O) parameters: (relevant to non-contiguous access from multiple processes).

i. cb_buffer_size: size of intermediate buffer (e.g. 4MB).

ii. cb_nodes: number of aggregators to use when writing output (automatically set to the stripe count of the file).

3. Parallel HDF5 parameters:

a) chunk dimensions

i. H5hut: H5Block3dSetChunk(file,xdim,ydim,zdim).

ii. pHDF5: H5Pset_chunk(dataset, 3, chunk_dims).

b) alignment

i. pHDF5: H5Pset_alignment(plist, threshold, alignment).

3.4 Conclusion

The conclusion of this analysis is that it appears that to improve writing performance we need to do a few simple things:

1. Increase the striping count,

2. Possibly also decrease the stripe length, and

3. Use array chunking with HDF5 to ensure that field data written by each process is contiguous.

We do not expect non-alignment to present a fundamental scaling barrier to writing output. Indeed, when writing output, there is no parallelization benefit to be gained by having a 3D scalar array occupy a number of stripes significantly greater than the number of disks; if the number of stripes is thus restricted, then, as the number of processes is increased while holding constant the subdomain size of each process, stripe size becomes large relative to the chunk size, and only a few chunks will occupy multiple stripes.

3.5 Other works done in the framework of this project

The ROSETTA mission offers a unique opportunity to monitor the plasma activity around the comet 67P/Churyumov–Gerasimenko from the collisionless outer coma to the collision-dominated inner coma. The aim of this work is to model and characterize the transition between the collisionless and collisional behaviours of cometary electrons. To achieve this goal, we plan to develop and implement the treatment of collisions and photo-ionization in a pre-existing full kinetic plasma code (iPIC3D), in order to study of the relaxation of cometary electron distribution functions in the cometary environment. Finally, we will combined these numerical plasmas simulations with in situ plasma measurements from different RPC instruments on-board ROSETTA.

The main objective of this research is to develop a full set of PIC/MC model that is capable to describe the physical processes in the SWC interactions. These results are anticipated:(1) Plasma (Mostly electrons, H2O+, H+, O-) density and temperature should be calculated, especially, the energy distribution function of the electrons.(2) Cometopause, where the plasma density and temperature varies abruptly and the collioness to collisional transition occurs, can be predicted.

We have developed a Monte Carlo (MC) module in the frame work of iPIC3D program to study this process self-consistently, which solves the Vlasov equation in the (weakly) collisional regime, and include both the kinetic and the collisional treatment of cometary electrons, ions with molecules. In this model, we consider the charged particles of electrons, H+, H2O+, O-, and the neutrals of H2O are treated as background with analytical formulation. One dimensional electrostatic and implicit PIC method is used to follow the motions of the charged particles. Photoionization and dissociation are also included. 12 electronic and 2 ionic collision processes are included, with the cross sections from LXCat (www.lxcat.net/). Recombination processes between the electrons and ions are also considered with temperature dependent recombination rate.

4. Perspectives for future collaboration between units (1 page)

(1) In the future work, we will improve the writing of parallel HDF5 output of iPIC3D, by choosing optimal values for parallel I/O parameters such as stripe count and width. We will also try to improve the flexibility of array dimensions, by eliminate divisibility constraints relative to the processor array.

(2) With Monte Carlo module coupled with iPIC3D, By coupled PIC/MC method, the spatial-temporal electron density and temperature can be calculated in 3D, which can be directly compared with MIP and LAP observation. Moreover, the kinetic treatment allow us to give the electron energy distributions, which is always non-Maxwellian when the electron-molecule collisions are dominate, and can not be treated by fluid or hybrid model. The microscope treatment of the collision process can also grantee the correctness and accuracy of the results, which is only depend on the cross sections data and has been assured by the many related researches from lab plasma physics. So we can compare the results with observations of Rosseta mission directly.

5. Valorisation/Diffusion (including Publications, Conferences, Seminars, Missions abroad...

5.1 Publications:

(1) Computational study of plasma sustainability in radio frequency microdischarges, J. Appl. Phys. 115, 193301 (2014).

(2) Kinetic simulation of direct-current driven microdischarges in argon at atmospheric pressure. J. of Phys. D: Appl. Phys. 47 (43), 435201 (2014).

(3) Numerical characterization of local electrical breakdown in sub-micrometer metallized film capacitors. New J. of Phys. 16 (11), 113036 (2014).

(4) 2D implicit PIC/MCC simulation of Ar gas breakdown process in a field reversed configuration with bias magnetic field. To be submitted.

5.2 Conferences

(1) Deeper Meeting, Kaiserslautern, Germany, Feb.24-26, 2014.

(2) "Numerical Simulations of Solar-Wind Comet interactions based on implicit Particle-In cell/Monte Carlo Method", General Scientific Meeting 2014 of the Belgian Physical Society. Leuven, Belgium May.28, 2014.

5.3 Seminars

(1) "Monte Carlo Method in Gas Discharge Plasmas and Its Potential Applications to Solar Physics", CmPA. KULeuven, Apl. 22, 2014.

6. Skills/Added value transferred to home institution abroad (1/2 page)

(1) The File I/O part of iPic3D on CPU/MIC hybrid architecture is studied, tested, modified, benchmarked and improved by several means. The added values of this works are: The particle and field are now treated in one single file in one node instead of thousands of separated files in each node, which will allow much efficient to load and save. The post processing is also treated more conveniently, with significant CPU resources and time savings. The file I/O is also benchmarked and improved on MIC based clusters. The results can scale to very large CPU numbers. Above improvements will allow efficient treatments of File I/O of iPIC3D, and thus extend the capability of iPIC3D.

(2) With Monte Carlo module coupled with iPIC3D, By coupled PIC/MC method, the spatial-temporal electron density and temperature can be calculated in 3D, which can be directly compared with MIP and LAP observation. Moreover, the kinetic treatment allow us to give the electron energy distributions, which is always non-Maxwellian when the electron-molecule collisions are dominate, and can not be treated by fluid or hybrid model. The microscope treatment of the collision process can also grantee the correctness and accuracy of the results, which is only depend on the cross sections data and has been assured by the many related researches from lab plasma physics. So we can compare the results with observations of Rosseta mission directly.