Day Final – team06 PTSG-TW

Renovating the PTSG Code Suite with NCHC Open Hackathon

OpenACC
More Science, Less Programming

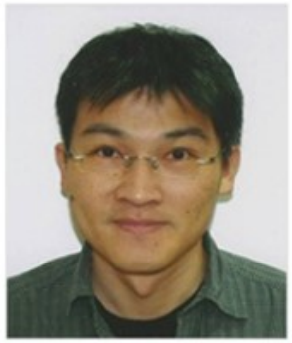
Total presentation time is 12 minutes

Team06 – PTSG-TW

Mentors:

- Leo Chen
- Yang-Hsien
- Chao-Shun (floating)
- Richard (floating)

- Host: Jay Chen
- Infra: Johnson Sun



Leader: Prof. Ming-Chieh Lin
National Taipei University of Technology

Member: Feng-Hua Chang
National Cheng Kung University

Member: Cheng-Cheng, Frank Yeh
National Cheng Kung University

Alexander von HUMBOLDT STIFTUNG

SIGMA XI
THE SCIENTIFIC RESEARCH HONOR SOCIETY

OpenACC More Science, Less Programming

OPEN HACKATHONS

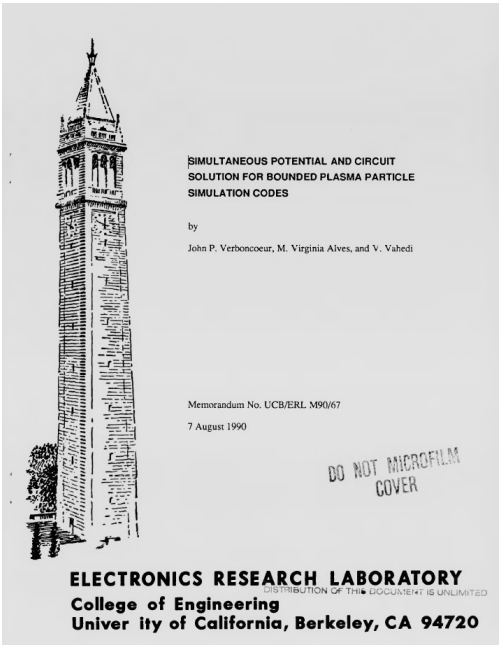
NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing

NVIDIA

PTSG (The Plasma Theory & Simulation Group)

- **Problem the team is trying to solve：**

應用領域	主要用途	技術方向
核融合能源	清潔能源	Tokamak、雷射融合
半導體製造	蝕刻與薄膜	PECVD、RIE
太空與天文	太陽風、推進器	MHD、霍爾推進
醫療環保	消毒、治療	冷電漿
材料工程	塗層與改質	熱電漿、表面處理


SIMULTANEOUS POTENTIAL AND CIRCUIT SOLUTION FOR BOUNDED PLASMA PARTICLE SIMULATION CODES

by

John P. Verboncoeur, M. Virginia Alves, and V. Vahedi

Memorandum No. UCB/ERL M90/67

7 August 1990

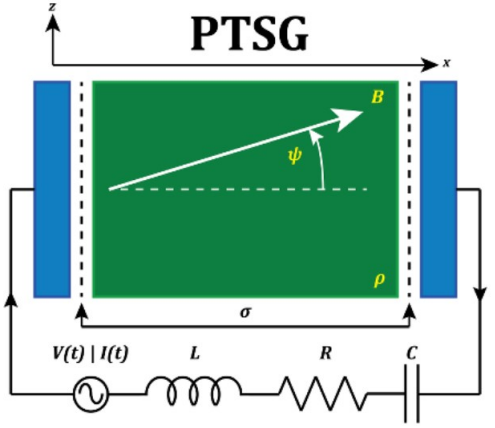
DO NOT MICROFILM COVER

ELECTRONICS RESEARCH LABORATORY

College of Engineering

Univer ity of California, Berkeley, CA 94720

The Plasma Theory and Simulation Group



The particle-in-cell Monte Carlo collision (PIC-MCC) code, XPDP1,

OpenACC More Science, Less Programming

OPEN HACKATHONS

NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing

NVIDIA

Particle-in-Cell (粒子式模擬數值分析方法)

- **Scientific driver for the chosen algorithm.**

We mainly work on Electromagnetics/Plasma Simulations using the particle-in-cell (PIC) Monte Carlo collision (MCC) code suite developed by the Plasma Theory and Simulation Group (PTSG), formerly at UC Berkeley, now at MSU. The PTSG codes are open sources written in C/C++ and we have been collaborated with the PTSG for 20 years. We would like to join this event to boost the PIC-MCC simulations with the state-of-the-art GPU capabilities, aiming at a 10x speed up.

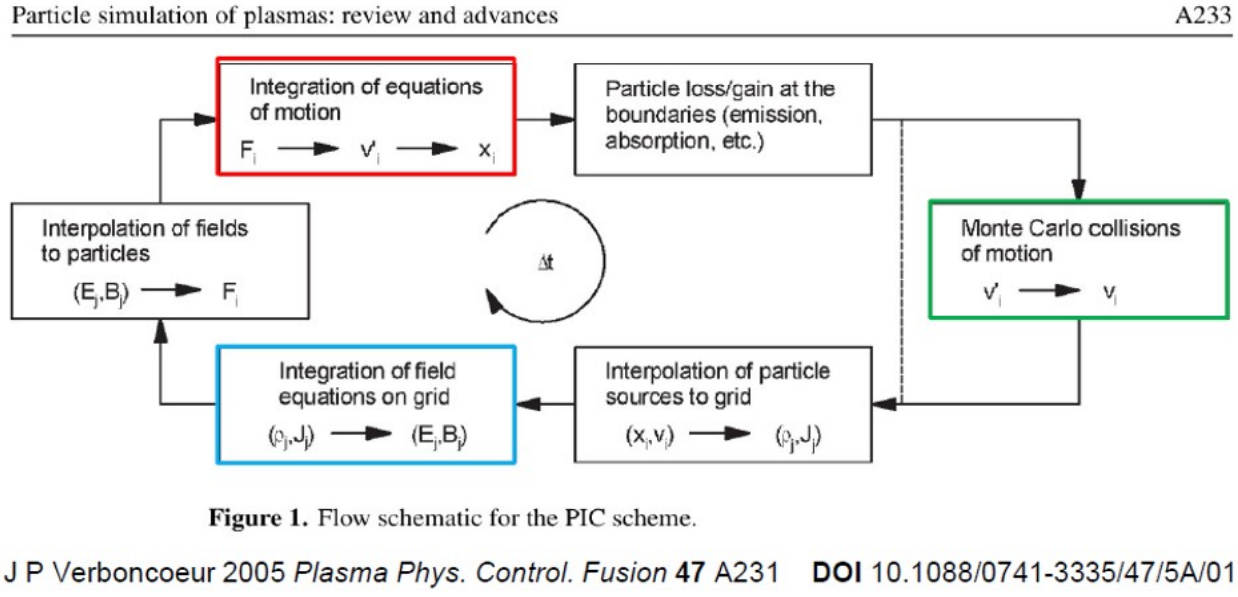


Figure 1. Flow schematic for the PIC scheme.

J P Verboncoeur 2005 *Plasma Phys. Control. Fusion* **47** A231 **DOI** 10.1088/0741-3335/47/5A/017

Xgrafix (C code real-time UI)

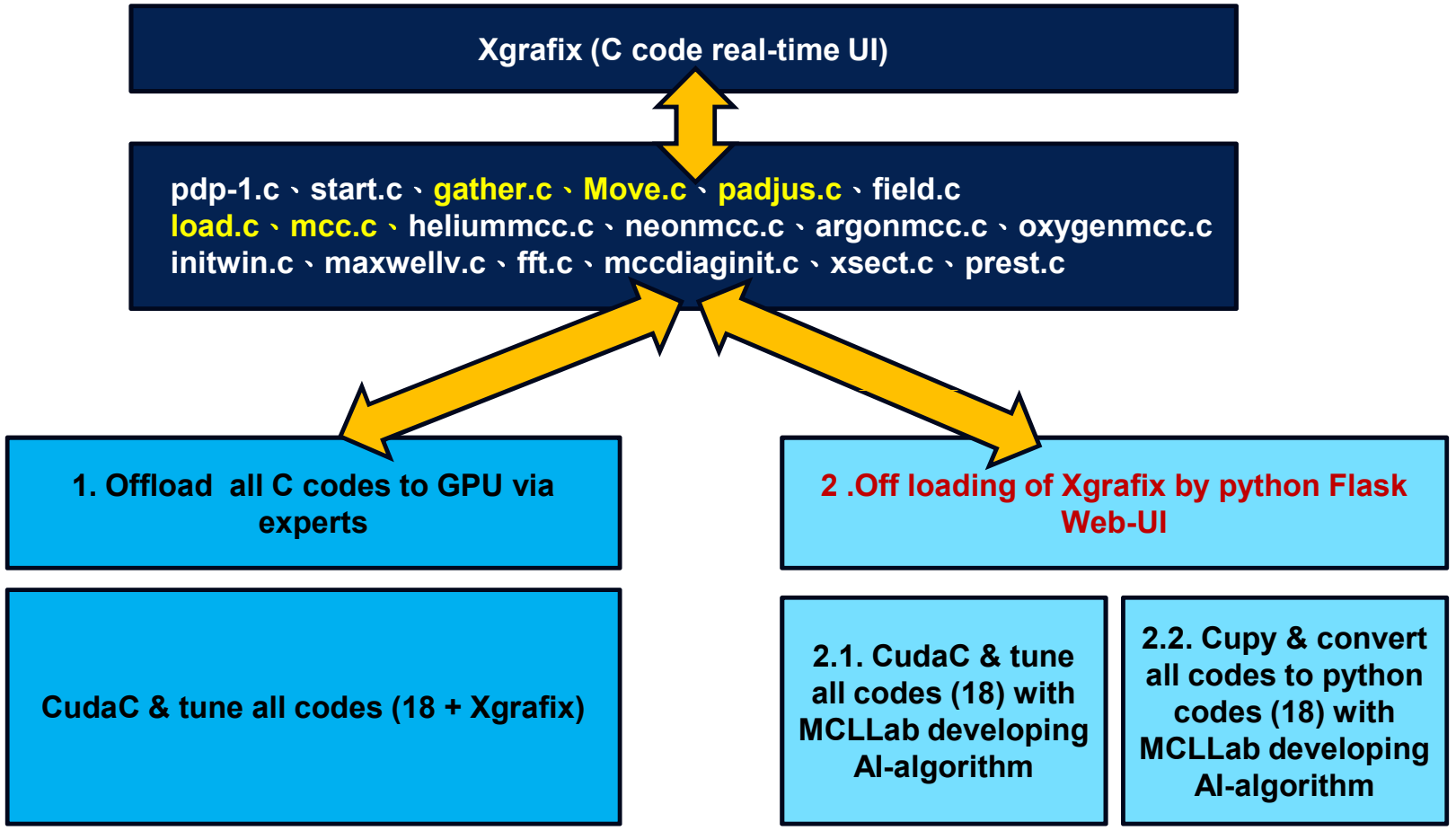
pdp-1.c、start.c、gather.c、Move.c、padjus.c、field.c

load.c、mcc.c、heliummcc.c、neonmcc.c、argonmcc.c、oxygenmcc.c

initwin.c、maxwellv.c、fft.c、mccdiaginit.c、xsect.c、prest.c

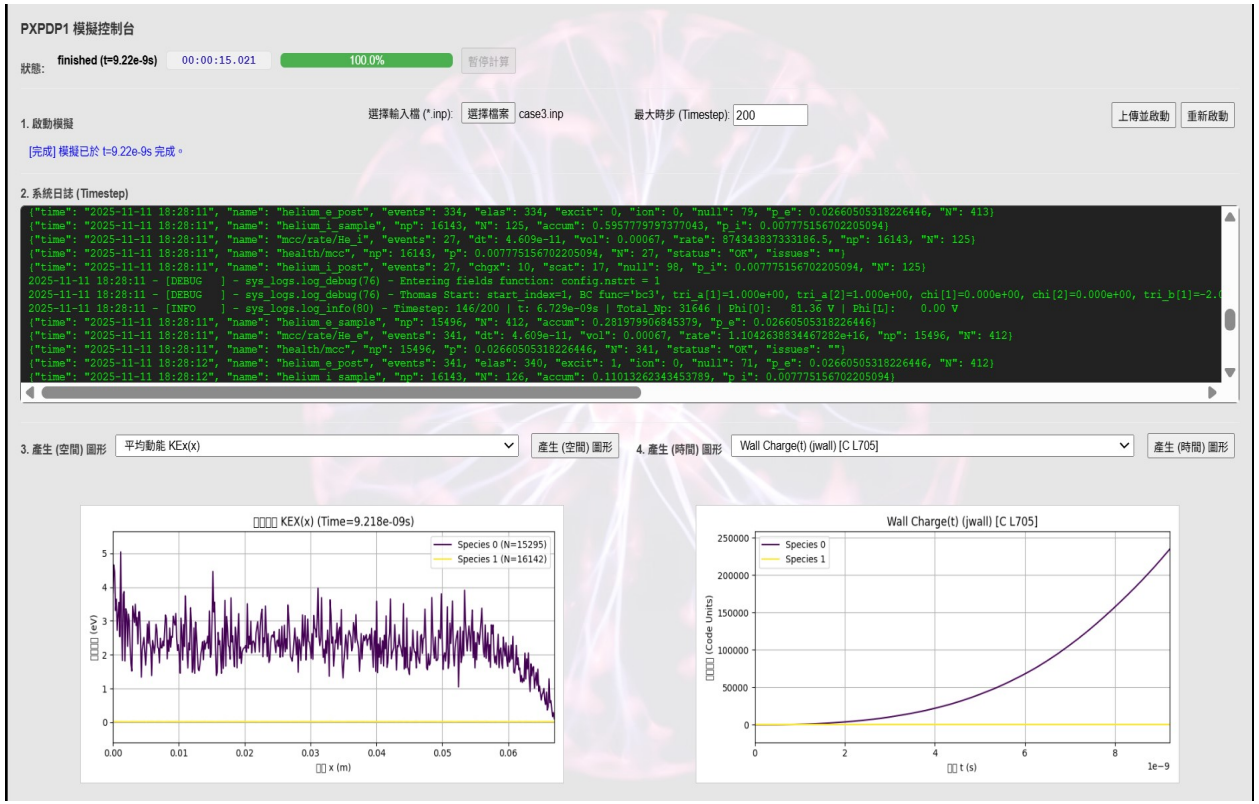
OpenACC More Science, Less Programming · OPEN HACKATHONS · NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing · NVIDIA

Evolution and Strategy



OpenACC More Science, Less Programming · OPEN HACKATHONS · NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing · NVIDIA

Results and Final Profile



2 .Off loading of Xgrafix by python Flask Web-UI

2.1. CudaC & tune all codes (18) with MCLLab developing AI-algorithm

2.2.a Successfully convert all codes to python codes with developing AI CodeGen support

2.2.b Successfully convert all python codes to Cupy 18) with developing AI-CodeGen support.

OpenACC More Science, Less Programming

OPEN HACKATHONS

國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing

NVIDIA

- MLC-Lab Developing GenAI Code algorithm
- With Novel Context-Engineering for CodeGen strategy of deployment of Python conversion、Pycu-Numba & Physical-understanding model
- Multi-LLM designs HMI (Human-Machine Inter-cooperation) working pattern.

OpenACC More Science, Less Programming
OPEN HACKATHONS
NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing
NVIDIA

經自製AI演算法評估：採用CuPy+Numba (numba.cuda) 混合加速策略。CPU擔任「指揮」，Nvidia GPU 擔任「平行計算」

move.py (粒子移動)加速策略： Numba @cuda.jit 核心執行細節, 最高優先級offload目標。原始的 for i in range(np_val):移除迴圈 (CPU上序列執行 87,151次)。建立一個 move_p_kernel Numba CUDA核心。CPU的 pdp1.py 主迴圈每一步啟動核心時, GPU啟動 87,151執行緒 (threats)，同時平行處理所有粒子速度和位置更新。建構「完美並行」計算任務，無需原子操作 (atomics)，加速效果極好。

mcc.py(蒙地卡羅碰撞)加速策略 採用Numba@cuda.jit 核心 + GPU RNG執行細節, 原始 for i in range(np_val): 迴圈移除。建立一個 mcc_p_kernel Numba CUDA 核心。關鍵：原 np.random.rand() (CPU函數) 替換為 numba.cuda. random，允許 GPU 高速平行生成隨機數。heliummcc.py 載入碰撞資料 (NumPy 陣列) 會被 Numba 自動快取到 GPU 常數記憶體中，供核心高速讀取。

gather.py (密度/電場收集)加速策略： Numba @cuda.jit 核心 + Atomic的操作執行, gather_n (粒子到網格) 存在競爭條件 (Race Condition),多個粒子執行緒(threat)可能會同時嘗試寫入同一個網格點(如 sp_n[128]). 加速的關鍵技術, 使用 numba. cuda.atomic.add() 取代 sp_n[ix] += ...確保 GPU平行寫入資料的準確性。gather_e (網格到粒子) 則無需 Atomic，因每個執行緒只寫入自己的粒子記憶體。

field.py (Poisson 求解器)加速策略： 使用NumPy, 刻意保留在 CPU 上計算。solve_poisson _thomas 函數實作的Thomas演算法是序列 (Serial) 架構，上下步計算依賴高。強行在GPU上執行序列任務，效能會遠低於CPU。另外的通訊成本分析：每一步 執行一次 GPU->CPU (下載 sp_n) 和 CPU->GPU (上傳 e)。通訊成本(約 16 KB)是微秒 (μs)等級，低於在CPU上執行 87,151次粒子迴圈毫秒 (ms) 等級成本。

padjus.py (粒子壓縮)加速策略： 混合CPU + GPU, 基於CuPy布林索引 (Boolean Indexing)執行 history_ plot.每12,800 步執行一次，在 CPU上讀取config.phi, 並particle_ adjust必須在GPU執行。避免每一步會在CPU/ GPU間來回複製所有87,151個粒子資料, 以致拖累效能。關鍵策略使用 mask=(BIT[isp] >= 0.0) 和 config.x[isp] = config.x [isp] [mask]。CuPy直接在 GPU 內部平行執行壓縮，通訊成本為零

HPC效能測試成果

XPDP1 Benchmark-(Serial run on AMD EPYC 7742 64-Core Processor v.s. Offload mover to A100)

	CPU	A100
Case1	2m5.670s	8m32.614s
Case2	49m11.255s	144m47.385s
Case3	102m4.364s	305m45.760s
Case4	687m39.295s	1980m19.260s

XPDP1 Benchmark-(Serial run on Intel(R) Xeon(R) Gold 6154 18-Core Processor v.s. Offload mover to V100)

	CPU	V100
Case1	1m55.528s	10m58.565s
Case2	42m39.145s	164m8.632s
Case3	90m40.165s	339m35.767s
Case4	659m53.983s	2198m2.231s

裝置名稱	RTX-4070
處理器	Intel(R) Core(TM) i7-14700F (2.10 GHz)
已安裝記憶體(RAM)	32.0 GB (31.8 GB 可用)

BMT timestep set as = 20,000

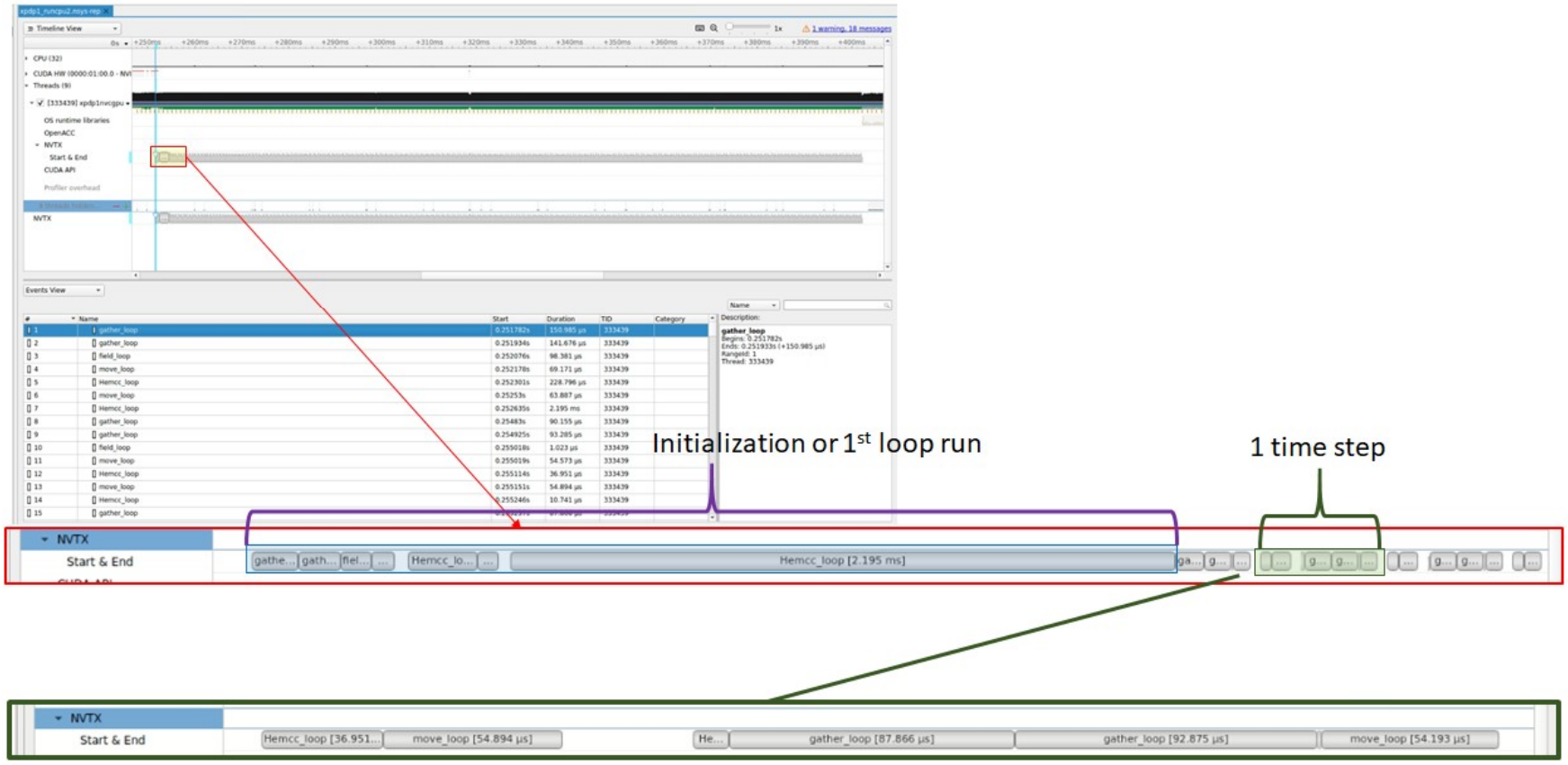
	CPU	RTX4070
Case 1	18m45s	X
Case2		
Case3	**00:33:04.504**	**00:11:36.364**
Case4		

** Case1&3 are both Helium base inputs.

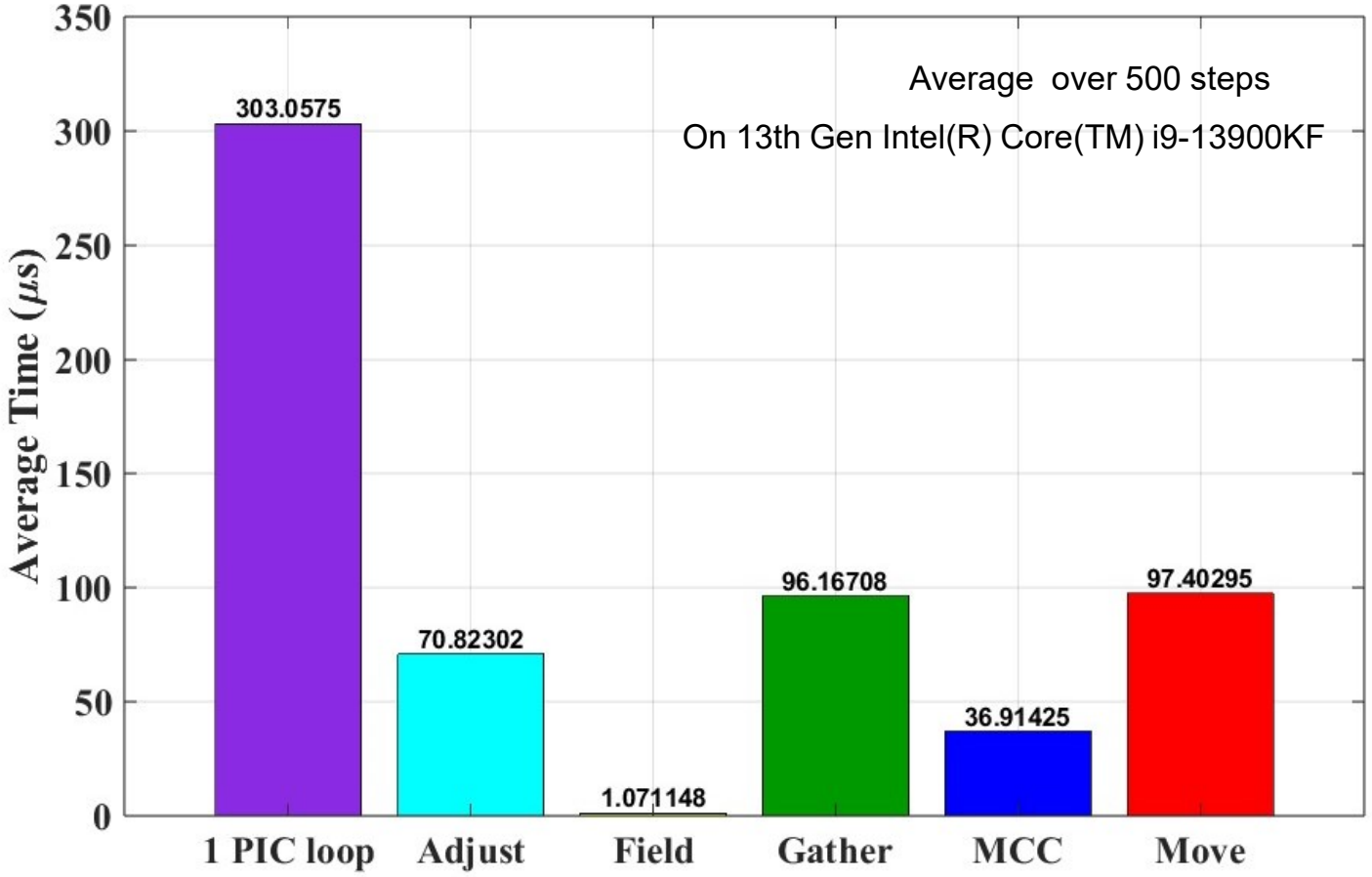
OpenACC More Science, Less Programming | OPEN HACKATHONS | NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing | NVIDIA

NVIDIA Nsight Systems - Analysis

CPU time analysis for loops in: move, field, gather and mcc



Average run time: (both species 1 &2 combined)



Revised Priority of chosen algorithms after Nsight Systems Analysis

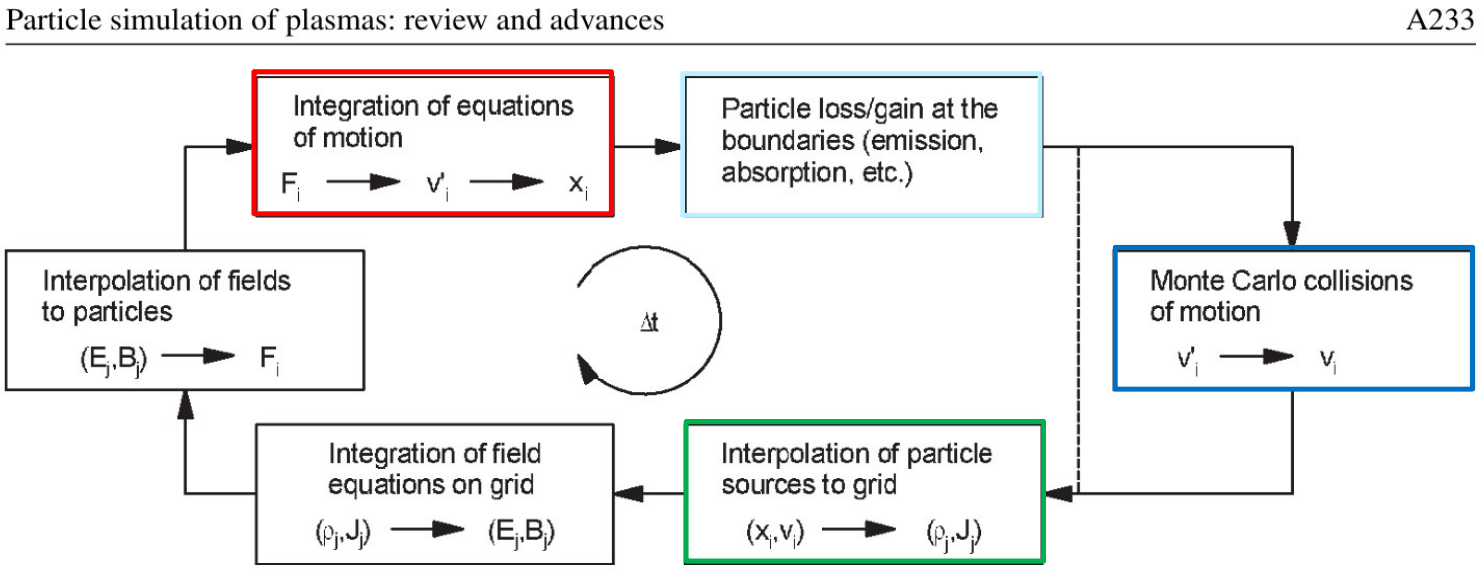
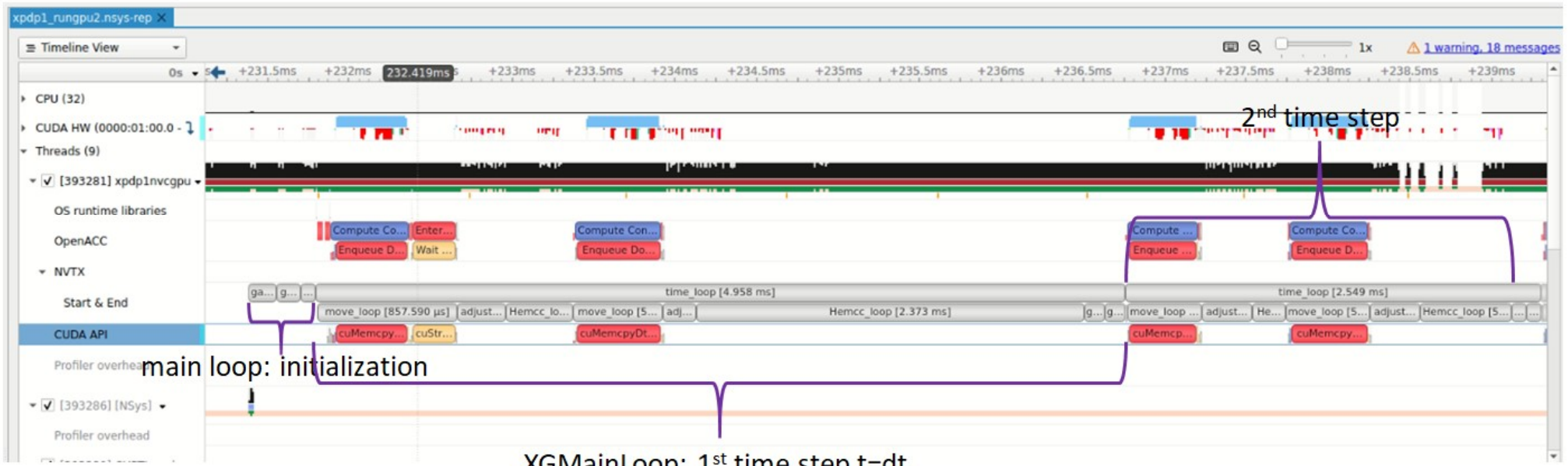


Figure 1. Flow schematic for the PIC scheme.

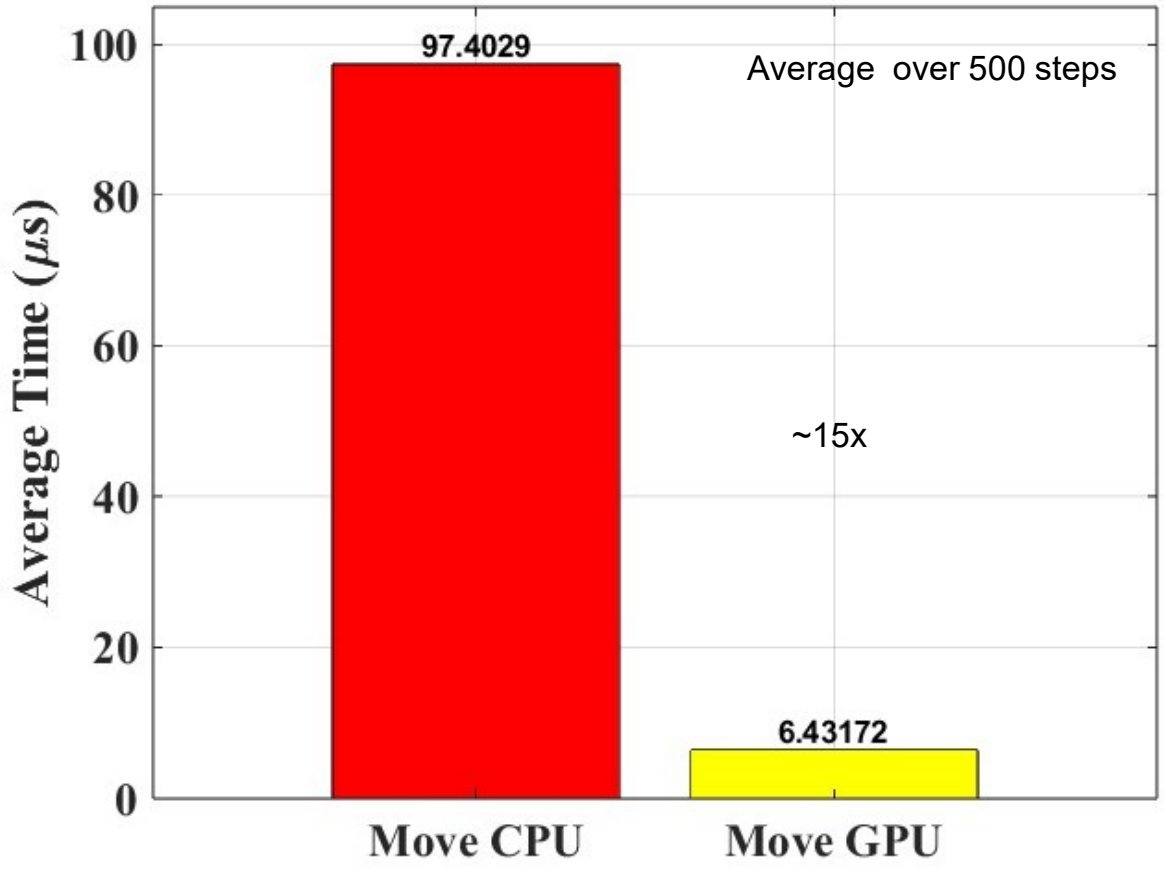
J P Verboncoeur 2005 *Plasma Phys. Control. Fusion* **47** A231 **DOI** 10.1088/0741-3335/47/5A/017

Offload move.c to GPU RTX 4080

Nsight systems analysis:



Offload move.c to GPU RTX 4080



Summary

• We gain a lot during participating the NCHC Open Hackathon (NOH) and thank for the support of NCHC, NVIDIA, and OpenACC staff scientists and engineers.

• It is discovered that the GUI of the PTSG Code Suite, Xgrafix, despite a good dynamic interface, becomes the main bottleneck of the GPU acceleration.

•Decomposition of Xgrafix and XPDP1 is successful so that a new UI has been developed during this NOH.

• HPC-SDK is a good tool, easy to install and use to offload existing C codes to GPU using ACC.

• Nsight Systems Analysis is very useful. With the scientific experimentation, we have timed the algorithms in a PIC loop and determined the priority of immigrating the algorithms from CPU to GPU.

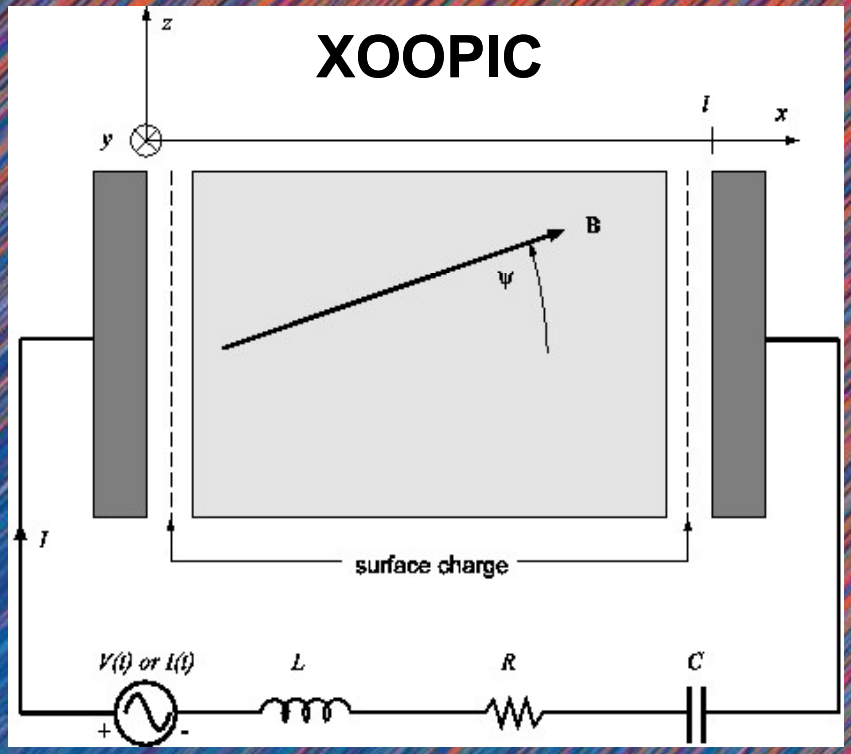
•As a preliminary offload the move.c to GPU RTX4080, we got a 15x speedup! This effort will be continued…

OpenACC More Science, Less Programming OPEN HACKATHONS NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing NVIDIA

Thank You Very Much!

The Plasma Theory and Simulation Group

Charles K. Birdsall
EECS, UCB



John P. Verboncoeur
NE, UCB

OpenACC
More Science. Less Programming