Commun. Comput. Phys.
doi: 10.4208/cicp.070115.030715a

Vol. **19**, No. 1, pp. 205-225
January 2016

Implementation of 2D Domain Decomposition in the UCAN Gyrokinetic Particle-in-Cell Code and Resulting Performance of UCAN2

Jean-Noel G. Leboeuf[1,*], Viktor K. Decyk[2], David E. Newman[1] and Raul Sanchez[3]

[1] *Department of Physics, University of Alaska, Fairbanks, AK 99775-5920, USA.*
[2] *Department of Physics and Astronomy, and Institute for Digital Research and Education (IDRE), University of California, Los Angeles, CA 90095-1547, USA.*
[3] *Departamento de Fsica, Universidad Carlos III, Leganes 28911, Madrid, Spain.*

Received 7 January 2015; Accepted (in revised version) 3 July 2015

Abstract. The massively parallel, nonlinear, three-dimensional (3D), toroidal, electrostatic, gyrokinetic, particle-in-cell (PIC), Cartesian geometry UCAN code, with particle ions and adiabatic electrons, has been successfully exercised to identify non-diffusive transport characteristics in present day tokamak discharges. The limitation in applying UCAN to larger scale discharges is the 1D domain decomposition in the toroidal (or z-) direction for massively parallel implementation using MPI which has restricted the calculations to a few hundred ion Larmor radii or gyroradii per plasma minor radius. To exceed these sizes, we have implemented 2D domain decomposition in UCAN with the addition of the y-direction to the processor mix. This has been facilitated by use of relevant components in the P2LIB library of field and particle management routines developed for UCLA's UPIC Framework of conventional PIC codes. The gyro-averaging specific to gyrokinetic codes is simplified by the use of replicated arrays for efficient charge accumulation and force deposition. The 2D domain-decomposed UCAN2 code reproduces the original 1D domain nonlinear results within round-off. Benchmarks of UCAN2 on the Cray XC30 Edison at NERSC demonstrate ideal scaling when problem size is increased along with processor number up to the largest power of 2 available, namely 131,072 processors. These particle weak scaling benchmarks also indicate that the 1 nanosecond per particle per time step and 1 TFlops barriers are easily broken by UCAN2 with 1 billion particles or more and 2000 or more processors.

PACS: 52.65.Tt

Key words: Gyrokinetic simulation, particle-in-cell method, massive parallelization.

*Corresponding author. *Email addresses:* `jnlscientific@hotmail.com` (J.-N. G. Leboeuf), `decyk@physics.ucla.edu` (V. K. Decyk), `denewman@alaska.edu` (D. E. Newman), `rsanchez@fis.uc3m.es` (R. Sanchez)

1 Introduction

Nonlinear gyrokinetic equations which describe complex plasma dynamics over time scales that are long compared to the Larmor or gyro motion perpendicular to the ambient magnetic field have come to play a fundamental role in our understanding of the confinement-time behavior of strongly magnetized plasmas [1]. Gyrokinetic particle-in-cell (PIC) codes which solve these gyro motion-averaged equations have proven to be powerful tools to study anomalous plasma transport in magnetic confinement devices such as tokamaks [2]. Among these, global codes [3–8] which cover the whole plasma cross-section or the entire plasma minor radius the short way around the torus, the other direction being the toroidal one along the major radius the long way around the torus, aim to describe at once plasma behavior throughout the whole device. Such global codes became feasible with the advent of the fully nonlinear δf method of Parker and Lee [9]. With this method, the perturbed part of the distribution function, δf, over its static equilibrium counterpart f_0, which defines the total distribution function as $f=f_0+\delta f$, is followed dynamically using discrete particles over a fixed background grid where the plasma response fields are evaluated. The δf description does however entail evolving in time and space weights which represent the fraction of phase space ($\delta f/f$) associated with every gyrokinetic particle in addition to velocities and positions as in conventional full f PIC codes. These global codes have become practical as physics tools with their massive parallelization for multi-processor mainframes such as the very recent Cray XC30 Edison at the National Energy Research Scientific Computing Center (NERSC) in the US or the IBM MareNostrum III at the Barcelona Supercomputing Center (BSC) in Spain.

The UCAN gyrokinetic code, developed jointly at the University of California at Los Angeles (UCLA) and University of Alberta, Canada, hence the acronym, is one of these early global codes to use PIC methods [10] to solve the gyrokinetic equations with only electrostatic effects included [4, 5]. The electrons are treated (analytically) as being adiabatic in UCAN and only the kinetic ions are followed as particles. UCAN has the added oddity that it uses Cartesian coordinates, as opposed to magnetic field-following coordinates, which encompass the toroidal geometry with the z-direction accounting for the toroidal one and the x-y plane representing the poloidal cross section. The plasma cross-section is however circular in that plane with the particles excluded from occupying the region outside the limit circle whose radius is specified as a certain, suitably large, number of ion gyroradii. Because of that exclusion, the boundary conditions for the field quantities can be taken as periodic without apparent deleterious effects. Spatial operations in UCAN are handled using finite differences on the fixed background Cartesian grid and Fast Fourier Transforms (FFTs) where appropriate, such as when solving the gyrokinetic Poisson equation to calculate the electrostatic potential from the gyrophase-averaged, or gyroaveraged for short, ion density accumulated from the particles. Along with the gyrokinetic Poisson equation, the equations of motion of the particle ions actually used in UCAN have been given in detail in Kniep [11]. We remark that a three-level predictor-corrector scheme is utilized for the time advance of the velocities, positions, and weights, with the particles pushed, the densities evaluated, and the Poisson equation solved, twice per cycle. UCAN was successfully calibrated linearly and nonlinearly against other gyrokinetic and gyrofluid codes available at that time as part of the Ion Temperature Gradient (ITG) driven turbulence benchmark performed by the Cyclone Team [2].

Through the course of its evolution since then, UCAN was modernized using Fortran90 constructs such as modules and interfaces to increase its robustness and ease of modification following prescriptions described in Norton et al. [12]. Massively parallel implementation, with message passing handled using MPI, was effected in UCAN using one-dimensional domain decomposition in the toroidal or z-direction. This was greatly leveraged by making ample use of many components of the PLIB library contained in the UPIC Framework of conventional PIC codes [13,14]. Of particular usefulness are the parallel FFTs, the global sum routines, and the particle manager PMOVE. The thus effected 1D domain decomposition means that the number of partitions is at most equal to the number of grid points retained in the z-direction. Moreover in that configuration, every processor handles the poloidal plane associated with that z-grid point or partition, which in turn places a restrictive limit, processor memory-wise, on the number of grid points retained in the x and y directions.

We single out two features which make UCAN sufficiently different from a conventional PIC code to warrant careful treatment. These peculiarities are clarified further in the next section where the equations used in UCAN are presented. The first, which is common to all gyrokinetic PIC codes, is the gyrophase average of the density as it is accumulated from the particle positions and weights and of the force as it is applied back to the particle ions. The classic realization of the gyrophase average, as first implemented by Lee [15], is performed over 4 points in the poloidal or x-y plane around a circle centered at the particle's position and of radial extent equal to the particle's Larmor radius. The second is specific to gyrokinetic codes where the adiabatic response of the electrons is used. Self-consistent flows generated by the turbulent fluctuations themselves act to regulate ITG driven turbulence. The electrons do not however respond to the global component of these so-called zonal flows and this is accounted for by subtracting the flux-surface averaged component from the electrostatic potential. The flux-surface average is along the poloidal and toroidal magnetic directions resulting in a purely radial quantity. In UCAN, this average is further complicated by the Cartesian geometry and a coordinate transformation is performed from x, y, z Cartesian coordinates to r, θ, ζ toroidal coordinates where the average over θ and ζ is performed. Next, the resulting radial quantity is transformed back to x, y, z space and subtracted from the electrostatic potential in the adiabatic electron response.

In spite of its simplicity, UCAN has been successfully exercised since, for example, to describe the effect of sheared flows on ITG turbulence in tokamaks [16], to ascertain whether ITG is active in specially tailored DIII-D discharges [17], to compare experimental and simulated radial correlation lengths of core turbulence in the DIII-D tokamak [18], to assess the effect of the so-called parallel nonlinearity on ITG turbulence [11,19].

More recently, UCAN has been used to tackle the challenging task of identifying non-diffusive transport characteristics in present day tokamak discharges [20–22]. This is accomplished through the use of tracer/tagged particles which are a more or less voluminous subset of the dynamical particles and which need to be tracked as they meander through processor space. Efficient particle tracking in a massively parallel environment is a design and implementation challenge. An important design decision was to tag the subset of tracked dynamical particles with an initial uniformly assigned z-position. The parallel particle manager PMOVE was used to move the tagged particles to the processor where they were born. The particles within each processor and in successive processors were then ordered on the supercomputer rather than on the post-processor and the thus ordered, tagged particle positions, weights, and velocities were then written out in processor order number to the supercomputer. The final step was a post-processing module which reads in ordered, tagged particles in processor order number at each time dump.

The limitation in applying UCAN to larger scale discharges is the 1D domain decomposition in the toroidal (or z-) direction for massively parallel implementation using MPI which has restricted the calculations to a few hundred ion Larmor radii or gyroradii per minor radius. To exceed these sizes, we have implemented 2D domain decomposition in UCAN with the addition of the y-direction to the processor mix. This paper describes in succession the equations used in UCAN, the strategy and details in the implementation of 2D domain decomposition in UCAN, with gyrophase averaging, flux surface averaging, and particle tracking in 2D processor space succinctly singled out, and benchmarks of the thus-minted UCAN2 on mostly the Cray XC30 Edison at NERSC. A summarizing discussion completes this presentation.

2 Equations in the UCAN gyrokinetic code

For the sake of completeness and to clarify further some issues raised in the preceding section, the particle and field equations used in UCAN are briefly presented. The notation is adapted from Hahm [23]. The particle ions follow the characteristics of the nonlinear Vlasov equation for the perturbed part of the distribution function δf with equilibrium distribution function f_0 so that the full distribution function is $f=f_0+\delta f$:

$$\frac{\partial \delta f}{\partial t}+\frac{d\vec{R}}{dt}\cdot\vec{\nabla}\delta f+\frac{dv_{||}}{dt}\frac{\partial \delta f}{\partial v_{||}}=-\frac{d\vec{R}^{(1)}}{dt}\cdot\vec{\nabla}f_0-\frac{dv_{||}^{(1)}}{dt}\frac{\partial f_0}{\partial v_{||}}, \tag{2.1}$$

with R the denoting the displacement, $v_{||}$ the parallel velocity, ∇ the spatial gradient and t the time. Following Parker and Lee [9], Eq. (2.1) can be transformed into an equation for the weight $w=\delta f/f$:

$$\frac{\partial w}{\partial t}=-(1-w)\left[\frac{d\vec{R}^{(1)}}{dt}\cdot\vec{\nabla}f_0+\frac{dv_{||}^{(1)}}{dt}\frac{1}{f_0}\frac{\partial f_0}{\partial v_{||}}\right]. \tag{2.2}$$

This is the essence of the so-called fully nonlinear δf method. The linear characteristics which appear on the right hand side of the weight equation in association with the equilibrium quantities through f_0 are written as:

$$\frac{d\vec{R}^{(1)}}{dt} = \vec{v}_E = c\frac{\vec{E}\times\hat{b}}{B}, \tag{2.3}$$

$$\frac{dv_{||}^{(1)}}{dt} = \frac{q}{M}E_{||} = \frac{q}{M}(\hat{b}\cdot\vec{E}), \tag{2.4}$$

with E denoting the gyroaveraged electric field, B the (herein static) total magnetic field, $\hat{b}=\vec{B}/B$ the unit vector associated with the magnetic field, $\vec{v}_E$ the $E\times B$ velocity perpendicular to the magnetic field, c the speed of light, q the ion charge, and M the ion mass (For simplicity of presentation we take the ions to be protons so that $q=|e|$, where e denotes the electron charge).

The nonlinear characteristics of Eq. (2.1) which appear in association with the perturbed part δf of the distribution function and which are followed by the particle ions are written as:

$$\frac{d\vec{R}}{dt} = v_{||}\hat{b} + \vec{v}_E + \frac{c}{qB}\mu\hat{b}\times\vec{\nabla}B + \frac{v_{||}^2}{\Omega}\hat{b}\times(\hat{b}\cdot\nabla)\hat{b} \tag{2.5}$$

for the displacement and

$$\frac{dv_{||}}{dt} = \frac{q}{M}E_{||} - \frac{1}{M}\mu(\hat{b}\cdot\vec{\nabla}B) \tag{2.6}$$

for the parallel velocity. The third term on the right hand side of Eq. (2.5) is the grad-B drift with magnetic moment $\mu = Mv_\perp^2/2B$ and the fourth term the curvature drift with ion cyclotron frequency $\Omega = qB/Mc$. The first term on the right hand side of Eq. (2.6) represents the acceleration of the ions by the parallel component of the electric field and the second the mirror force due to the nonuniform spatial character of the magnetic field. With these notations the ion Larmor radius or gyroradius is defined as $\rho_i = v_\perp/\Omega$.

The gyrokinetic Poisson equation which yields the electrostatic potential ϕ and subsequently the self consistent electric fields is written as:

$$\frac{1}{\lambda_{Di}^2}\phi\left[1-\Gamma_0(k_\perp^2\rho_i^2)\right] = 4\pi|e|(n_i - n_e), \tag{2.7}$$

where $\lambda_{Di}^2 = T_i/4\pi n_0 e^2$ is the square of the ion Debye length. The term with $\Gamma_0(k_\perp^2\rho_i^2) = I_0(k_\perp^2\rho_i^2)e^{-k_\perp^2\rho_i^2}$, with I_0 the modified Bessel function of order zero and $k_\perp$ the perpendicular component of the wavevector, accounts for the polarization drift of the ions.

The ion density n_i in Eq. (2.7) is actually the density averaged over the gyrophase φ and is expressed as:

$$n_i = \oint\frac{d\varphi}{2\pi}N_i(\vec{R}+\rho_i), \tag{2.8}$$

where N_i is the particle density in real space. The electric fields in Eqs. (2.3)-(2.6) are also the averages over the gyrophase.

The gyrophase averaging in UCAN is performed numerically as the density is gathered from the particle weights and positions around a charged ring in the poloidal or x-y plane of radius equal to the ion gyroradius of each individual particle centered at the particle's position. The same procedure is applied to the electric fields as they are scattered to the particles when they are advanced in time according to Eqs. (2.2), (2.6) and (2.5). Lee [15] who pioneered the technique has shown that this 4-point averaging around the Larmor circle is accurate to $k_\perp \rho_i = 2$. More points extend the domain of validity of this representation.

The electrons in UCAN are taken to be adiabatic and are treated analytically. The electron density in Eq. (2.7) is therefore expressed as:

$$n_e = n_0 \frac{|e|}{T_e}[\phi - <\phi>]. \tag{2.9}$$

The second term in brackets on the right hand side of Eq. (2.9), $<\phi>$, represents the spatial average of the electrostatic potential along the poloidal and toroidal directions, in other words its flux surface average, or equivalently the $k_{||}$ component of the self-consistent electrostatic potential. That component of the electron response is not adiabatic and does not therefore contribute to the electron density. Its subtraction in Eq. (2.9) is essential to the correct treatment of the self-consistent flows or zonal flows generated by the turbulent fluctuations themselves and which in turn greatly contribute to regulate the turbulence.

3 Implementation of 2D domain decomposition in UCAN

3.1 Strategy

Two-dimensional domain decomposition was accomplished in UCAN by adding the y-direction to the processor mix which originally only included the z-direction when the decomposition was one dimensional. In the 2D domain-decomposed UCAN2, both the y- and z-directions are then distributed.

The overall strategy for implementation of 2D domain decomposition in UCAN was largely dictated by the fact that we started with a production code which used 1D domain decomposition. The outcome was to keep the 1D domain-decomposed code as is and along side it insert the 2D domain decomposition machinery. The 1D domain part would then be discarded when the 2D domain-decomposed code was verified.

The obvious advantage of this strategy is that it makes it easier to directly check the 1D to 2D domain decomposition modifications by collapsing the 2D domain-decomposed parts of the code to 1D. The check can be and has been done in either the original z-direction or the new parallel y-direction as the upgrade from 1D to 2D domain decomposition proceeds.

The leverage for the successful outcome of this strategy arises from the 2D domain decomposition machinery anchored by the P2LIB suite of 2D domain procedures and routines such as FFTs, global sums, and the all important particle manager PMOVE, all readily available in UCLAs UPIC Framework [13, 14]. As implementation proceeds, the P2LIB library gradually replaces but still coexists with its 1D domain-decomposed antecedent PLIB before the latter is discarded when 2D domain decomposition is deemed complete.

3.2 Implementation details

To implement this strategy, all of the space (real and Fourier) arrays have been duplicated with a 2 appended or inserted in place of a 1. Their indexing has also been modified from (nx,ny,nz/nproc) to (nx,ny/nprocy,nz/nprocz), where nx, ny and nz are the total number of grid points in the x-, y-, and z-directions, nprocy and nprocz the number of processors along y and z and nproc=nprocy$\times$nprocz is the total number of processors, to now reflect 2D domain decomposition in the z (original) and y (new) directions.

The same goes for the duplicated particle arrays such as particle position, parallel velocity, magnetic moment, and weight so that they now reflect 2D domain decomposition in the z and y directions. Interfaces, modules, and subroutines have also been duplicated with a 2 appended and modified to reflect 2D domain decomposition.

In effect, we really had both 1D and 2D domain-decomposed codes running simultaneously. This allowed us to perform direct head to head comparisons for verification whereupon the 1D domain pieces were discarded.

3.3 Critical design elements

The first critical design element is the accumulation of the gyroaveraged charge density from the positions and weights of the ions. For 1D domain decomposition, the gyrophase averaging discussed after Eq. (2.8) is rather trivial since the x and y directions are not distributed. But for 2D domain decomposition, one of the directions (y) is now also distributed and the relevant data will have to be retrieved from the appropriate partition. This also applies to the gyroaveraged fields in the particle push. The solution for this critical design element has been motivated by seminal work of Naitou et al. [24] where replicas are first mentioned, followed by that of Kim and Parker [25] and that of Hatzky [26] where domain cloning is invoked. It also guided by recent work of Madduri et al. [27] with the global gyrokinetic PIC code GTC [6] where the emanent shared grid is brought into play. In fact, their Fig 4c shows that this convenient replication maintains acceptable performance on GTC.

In UCAN2, for the 2D domain accumulation (gather) of the gyroaveraged ion charge density, shared grid translates into only replicating the gyroaveraged ion charge density in y for each z. This is only done during the deposit. After the deposit, these extra copies are then added and distributed and the field solver is fully decomposed with 2D domains. The same procedure of array replication is used for the gyroaveraged electric fields in the (scatter) particle push.

However, our use of replication of grid data in y does not imply we are using 1D domain decomposition. The replication is primarily to handle the worst case scenario that some energetic particles might have Larmor radii comparable to the system size. So long as there is enough memory, the cost of replication is negligible. Using 2D domain decomposition for particles improves cache performance compared to particle cloning if most of the data the particles need to read are within the domain. For computers with small memories per nodes, it may not be possible to replicate the entire domain in y. If the maximum Larmor radius was known to be smaller, however, it would be straightforward to replicate only the neighboring fields in y (the replication is done by a loop over y, not with an MPI collective). If the maximum Larmor radius was less than the size of a domain, then only one neighbor would be needed. If it was only a few cells then only guard cells would need to be replicated. This optimization was left for future work, however.

The second critical design element or peculiarity is the flux surface-averaged electrostatic potential $<\phi>$ needed in the gyrokinetic Poisson equation to properly account for zonal flows, as discussed after Eq. (2.9). To calculate its flux surface average, the potential is first transformed from Cartesian (x,y,z) to toroidal (r,θ,ζ) coordinates and then integrated or numerically summed in the θ and ζ coordinates. Again, this sum is relatively easy for 1D domains where only the z or ζ direction is distributed. Fortunately, since it is a sum, it is equally easy to use 2D domains where y or θ is now also distributed. Everyone just calculates the sum for the piece it owns and then we add it all up. The result is then mapped back to the 2D domain-decomposed flux surface-averaged potential array in x-y-z space.

3.4 Particle tracking in 2D processor space

The ultimate goal is to characterize transport using particles with UCAN2 in the presence of ITG turbulence in ITER-size tokamak discharges [6]. As was successfully done with the 1D domain-decomposed UCAN [20–22], a selected subset of dynamical particles is tracked, typically 1% of the total [21]. The caveat is that particles change locations in memory as the run proceeds, so these test particles had to be identified and reordered. The post processor does not (usually) have the resources to reorder a large number of particles, so it needed to be done on the supercomputer.

In the 1D (along z) domain-decomposed UCAN, the subset of tracked particles was identified with an initial, fictitious, uniformly assigned z-position uniquely determined by an integer tag attached to each tracked particle. In the 2D (along y and along z) domain-decomposed UCAN2, the tracked particles are now identified with fictitious uniform y and z positions. In 2D, this requires recording not only the integer tag assigned to each tracked particle which defines the fictitious z position (as it did in 1D) but also the processor identifier in y which is used to determine the fictitious y position. Here again, P2LIB's particle manager PMOVE is used to relocate the tagged particles to the processor where they were born. The new ORDER_TRACK2 subroutine then orders the particles locally within the processor. The P2LIB subroutine WRDATA writes out the ordered, real positions, weights, and velocities of the tagged particles in processor order number.

Thus, a large number of realizations of actual ion guiding-center orbits are at our disposal, and they can be used to compute the probability $P_i(r,t\,|\,r_0,t_0)$ of finding one ion guiding center at some radius r and time t if it was at another radius r_0 at a previous time $t_0 \le t$. This probability $P_i(r,t|r_0,t_0)$ is what is known as a propagator or, in our case, the ion guiding-center radial propagator. This propagator can then be used to probe the nature of radial transport [20–22].

4 Results

Throughout this section, the results to be presented have been obtained mostly on the Cray XC30 Edison at NERSC, but also on the Cray XE6 and the IBM iDataPlex at NERSC, as well as the IBM iDataPlex with Intel Sandy Bridge processors referred to as Mare Nostrum III at the Barcelona Supercomputer Center in Spain where both UCAN and UCAN2 have been easily ported, benchmarked, and exercised. All of the results were obtained using double precision no matter the mainframe or compiler. Having access to Mare Nostrum III at the Barcelona Supercomputer Center was instrumental to UCAN2 being readily ported, as it was, to the first configuration of the Cray XC 30 at NERSC which also had Intel Sandy Bridge processors. The current configuration of the Cray XC30 uses the successor higher performance Intel Ivy Bridge processors.

4.1 Comparison of results and of performance with UCAN and UCAN2

We start the results part of our paper with a comparison of the 2D domain-decomposed UCAN2 with its 1D domain UCAN counterpart for a calculation with parameters and profiles subject to the ITG instability which was previously featured in Rettig et al. [17]. This calculation which covers 9600 time steps goes well into the deep nonlinear saturated state with fully developed ITG turbulence. The system size of $256\times256\times128$ grid points and $512\times152\times128$ particles in the x-, y- and z-directions respectively is minimal for these calculations to be physically meaningful. The extent of a grid cell in x or y is roughly equal to the ion Larmor radius. The UCAN calculation uses 64 processors in the z-direction which is the direction in which domain decomposition is effected. The 2D domain decomposition in UCAN2 is two-fold: the first with 1 processor in the y-direction and 64 processors in the z-direction is effectively 1D; the second uses 8 processors in the y-direction and 8 processors in the z-direction and is 2D symmetric. All of these calculations were performed on the Cray XE6 and the Cray compiler was used. We customarily choose the time evolution of the normalized electrostatic energy as a comparative measure. It is displayed in Fig. 1 for the UCAN case and the two UCAN2 cases.

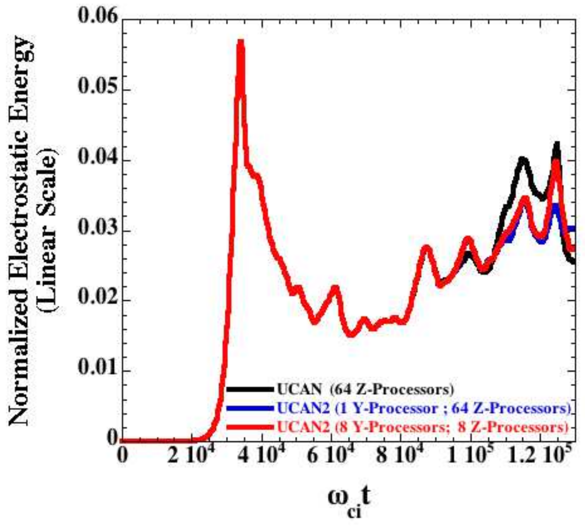

Figure 1: Time evolution of the normalized electrostatic energy with UCAN and UCAN2 with 2 different 2D domains.

The normalized electrostatic energy in the calculations with UCAN and UCAN2 displayed on a *linear* scale in Fig. 1 clearly shows the linear growth phase of the ITG instability, the typical overshoot when sheared (poloidal) flow self-generated by the unstable fluctuations in turn quenches these same fluctuations, and the turbulent nonlinear saturated phase after the overshoot. The electrostatic energy obtained with UCAN and UCAN2 is identical to within round off for more than 6000 steps, including through the overshoot, and only diverges in the more chaotic deep nonlinear saturated phase in the last 3000 steps. This is rather remarkable given the differing number and order of operations in both formulations and in turn lends confidence to the correctness of the UCAN2 implementation.

Now that we have ascertained that UCAN2 yields equivalent results to those obtained with UCAN, we assess whether 2D domain decomposition significantly taxes performance. For these comparisons, the calculations only cover 100 steps and the number of processors used in UCAN ranges from 1 to 128 (the maximum number of domains that can be accommodated with 128 grid points in the z-direction). For UCAN2, the hardwired domain decomposition is effectively 1D with only one processor used in the y direction and from 1 to 128 in the z-direction. Here the measure of performance is the time per particle per time step (in seconds) which has been adopted as a de facto standard for PIC codes. Its inverse which is the number of particles moved 1 time step in 1 second, termed compute power, is also interchangeably used. The results of this exercise are displayed in Fig. 2.

From Fig. 2, it is clear that UCAN2 is slightly more time consuming. In fact the data of Fig. 2 indicate that the. overhead, here defined as the difference between UCAN time and

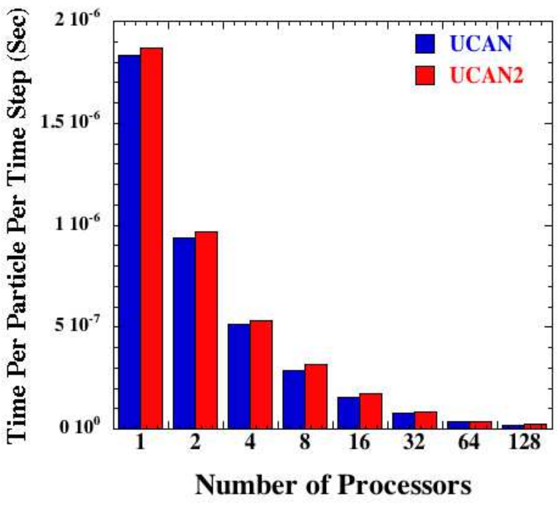

Figure 2: Time per particle per time step with UCAN for 1 to 128 processors in z and UCAN2 for 1 processor in y and from 1 to 128 processors in z.

UCAN2 time divided by UCAN time (in per cent), associated with the (rather extensive) 2D domain decomposition machinery ranges from 2% to at worst 10% (for 8 processors). This appears to be a small price to pay for the vast processor space opened up and large system sizes now facilitated by 2D domain decomposition.

One additional feature of UCAN2 is the P2LIB subroutine FCOMP which automatically sets the number of processors in y and z at the onset given the total number of processors to be used in the calculations and the problem size to be tackled. It offers as square a decomposition as it can, and if it cannot do a square one, it gives more processors to z than y. We have tested whether there is additional overhead associated with this automatic choice of 2D domain decomposition instead of the hardwired numbers of processors we could use as an alternative. This also serves as an indication of the overhead associated with a particular 2D decomposition. The tests have been performed over 100 time steps with our standard problem size of $256\times256\times128$ grids and $512\times512\times128$ particles with the same 128 processors used so far with the hardwired 2D domain (y,z) decomposition starting at (1,128) through (2,64), (4,32), (8,16), (16,8), (32,4), (64,2) and finishing with (128,1). FCOMP in this case returns (8,16). These tests were performed on the Cray XE6 where the Cray compiler was used and the Cray XC30 at NERSC with the Intel compiler and their results are displayed in Fig. 3.

It is clear from Fig. 3 that the decomposition automatically chosen by FCOMP indicated by open squares is close to optimal in this particular example. We thereafter tacitly assume that the 2D domain decomposition achieved by FCOMP is acceptable for an even greater number of processors (Non-systematic spotchecks performed at system sizes which accommodate a greater total number of processors do indeed confirm this is the case).

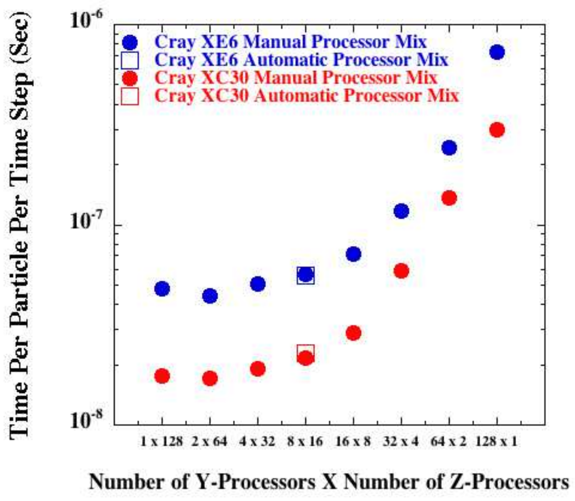

Figure 3: Time per particle per time step in seconds for manual or hardwired 2D domain decomposition (full circles) and the decomposition automatically chosen by FCOMP (open squares) in UCAN2.

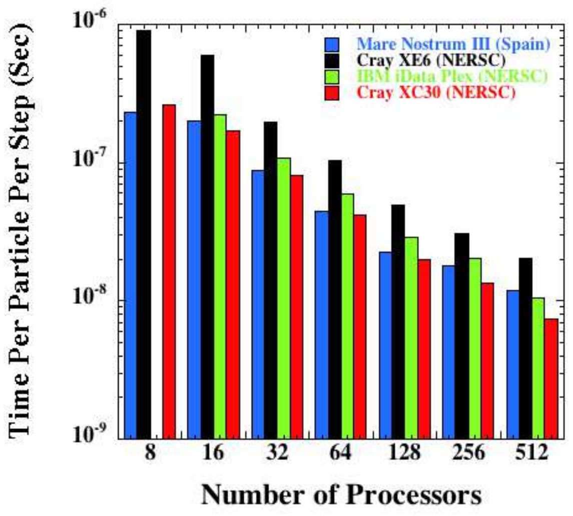

Figure 4: Comparative performance of UCAN2 on the IBM iDataPlex, the Cray XE6 and the Cray XC30 at NERSC, as well as on the IBM iDataPlex Mare Nostrum III at the Barcelona Supercomputer Center.

As a final preliminary, we emphasize the versatility of UCAN2 by showing, in Fig. 4, its comparative achievements in terms of time per particle per time step for a case with 4 particles per cell and a system size of $256\times256\times128$ grids on all of the (still operational) mainframes we have been granted access to, namely the IBM iDataPlex at NERSC, the Cray XE6 at NERSC, the Cray XC30 at NERSC, and the Mare Nostrum III IBM iDataPlex with Intel Sandy Bridge processors at the Barcelona Supercomputer Center.

These comparative studies encompass from 4 to 512 processors which is the maximum number at our disposal on the IBM iDataPlex at NERSC. The Intel compiler was used on all mainframes. According to the data thus obtained and displayed in Fig. 4, the performance of UCAN2 is overall better on the Cray XC30 than the other mainframes.

4.2 Performance of UCAN2

Now that we have set the rules of engagement, we present results probing the performance of UCAN2 mainly on the Cray XC30 Edison at NERSC with the higher resolutions in spatial extent and particle numbers made possible by the vast processor space liberated by the 2D domain decomposition in UCAN2.

We begin with the standard benchmark system size of $256\times256\times128$ grid points in the x-, y- and z-directions respectively to evaluate the performance of UCAN2 on as many processors as possible. With the standard benchmark size, the maximum number of processors which it has been possible to access turns out to be 16,384 ($=128\times128$), which corresponds to half the maximum number of available partitions or y-z grid points at 256 in $y\times128$ in z.

The object of this exercise is to see how well UCAN2 would perform when carrying out convergence studies with the number of particles in a production environment as was carried out in a limited way with UCAN in [20–22] with the calculations covering 100 ion Larmor radii or so. This would help to ascertain further that the non-diffusive characteristics of transport observed there were not an artifact of noise pollution at low numbers of particles [21].

We start with the time per particle per time step in seconds as a measure of performance for UCAN2. This is accomplished at this benchmark system size but with increasing number of particles per cell or grid point and with increasing numbers of processors starting with 8 and going up to 16,384. We let FCOMP automatically choose the 2D partitions which for instance gives $8=4y\times4z$, $512=16y\times32z$, $2048=32y\times64z$ and $16384=128y\times128z$. The results of this scan are displayed in Fig. 5. The number of particles is ramped up from 4 particles per cell to 2048 particles per cell by increasing the number of particles in the z-direction resulting in a maximum of 17,179,869,184 particles in the calculations, after which we run out of memory at either the small or large end of the processor space. So doing, we also compare the time measured with the ideal timing expected whereby the time goes down by a factor of 2 if the number of processors is increased by a factor of 2. The data of Fig. 5 were obtained on the Cray XC30 using the Cray compiler.

It is apparent from Fig. 5 that ideal scaling is maintained for a larger number of processors as more and more particles are used in the calculations. The data displayed in Fig. 5 also show that the 1 nanosecond (1 ns) barrier for the time per particle per time step is broken with more than 2000 processors and more than 64 particles per cell or about one-half billion particles and thereafter.

We follow with the floating point operations per second for UCAN2 measured in

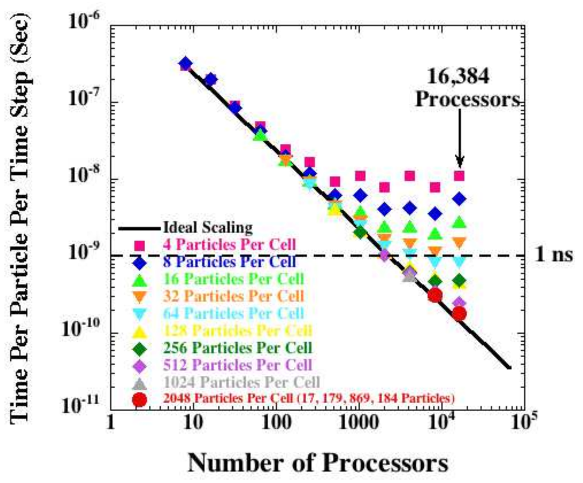

Figure 5: Time per particle per time step versus number of processors for a system size of $256\times256\times128$ and from 4 to 2048 particles per cell for a maximum of 17,179,869,184 particles.

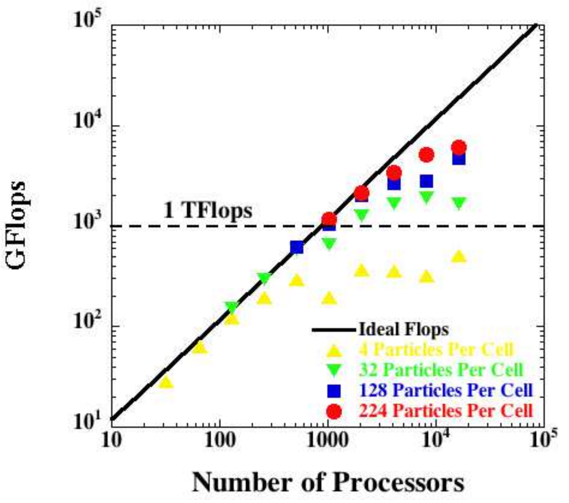

Figure 6: Floating point operations per second in GFlops versus number of processors for a system size of $256\times256\times128$ and from 4 to 224 particles per cell for a maximum of 2,046,820,352 particles.

GFlops at the standard benchmark system size again with increasing numbers of processors starting with 8 and going up to 16,384 and with the number of particles ramped up from 4 particles per cell to 244 particles per cell by increasing the number of particles in the z-direction. The floating point operations are obtained with the Cray PAT toolset on the Cray XC30 using the Cray compiler. We also compare the measured floating point operations with the ideal count expected whereby the count goes up by a factor of 2 if the number of processors is increased by a factor of 2. All of this is summarized in Fig. 6.

It is clear from Fig. 6 that ideal scaling for the floating point operations per second is maintained for a larger number of processors as the number of particles in the calculations is increased. It is also apparent from Fig. 6 that the figure of merit of 1 TFlops is exceeded with 1000 processors or more and 32 particles per cell, about a quarter of a billion particles, or more.

Profiling tools such as Perftools, PAT, and Apprentice on the Cray XC30 have been used to understand what causes the departure from ideal scaling with UCAN2. These tools tend to indicate that the slowdown results from load imbalance as the domains become smaller and smaller and the number of particles per processor decreases. For the scans with number of particles and number of processors in Fig. 5, departure from ideal scaling occurs at 131,072 particles per processor and below. The resulting load imbalance is characterized by increasing MPI_Wait times. This is also the conclusion reached with GTC in [28]. The same performance tools also reveal that the in the scalable regime, i.e. with more than 131,072 particles per region, the subroutines invoked in UCAN2 for replication do not even register in the piecharts showing time spent so that the overhead thus incurred is not significant in these instances.

While the standard benchmark system size of $256\times256\times128$ grid points has been adequate to accommodate 16,384 processors with UCAN2, larger system sizes which allow for more partitions are needed to access more processors up to the maximum number of processors available on the Cray XC30 with the largest power of 2, namely $2^{17}=131,072$, out of 133,824 processors in total. This restriction to a power of 2 is a legacy feature of UCAN when powers of 2 were necessary for the FFTs and as a machine feature for processor choice, for instance of the Cray T3D, which we carried over to UCAN2. This restriction has been removed in the UPIC Framework but has yet to be eliminated in UCAN2.

The object of this final exercise is to find out whether UCAN2 can access a larger number of processors efficiently and so doing reach system sizes commensurate to future generation devices such as ITER where the minor radius covers 1000 ion Larmor radii or more.

In any case, we have achieved accessing 65,536 processors by only doubling the number of grid points in each direction from the standard benchmark system size resulting in $512\times512\times256$ grid points being used in these more compute intensive calculations. This is the minimal system size which can be accommodated with 65,536 processors and the number of available y-z partitions is $512\times256=131,072$ which is twice the number of processors accessed. We again ramp up the number of particles per cell from 4 to 1024 by increasing the number of particles in the z-direction resulting in a substantial maximum of 68,719,476,376 particles in the calculations. We also compare the time measured with the ideal timing in this case. This is accomplished in Fig. 7.

The data displayed in Fig. 7 indicate that ideal scaling for the time per particle per time step in seconds is sustained up to a larger number of processors as more particles per cell are used in the calculations. In fact, ideal scaling is still obeyed up to the maximum number of 65,536 processors used in the scan when 1024 particles per cell are involved in the calculations for a grand total of 68,719,476,736 particles. It is again to be noted that the

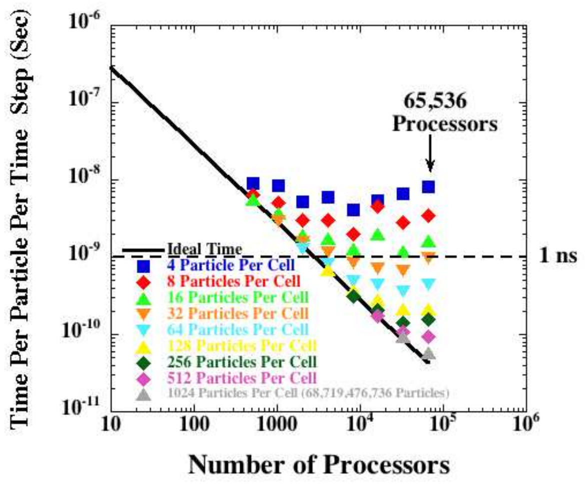

Figure 7: Time per particle per time step in seconds versus number of processors up to 65,536 processors. The system size for these calculations is 512×512×256 with 4 up to 1024 particles per cell for a maximum of 68,719,476,736 particles.

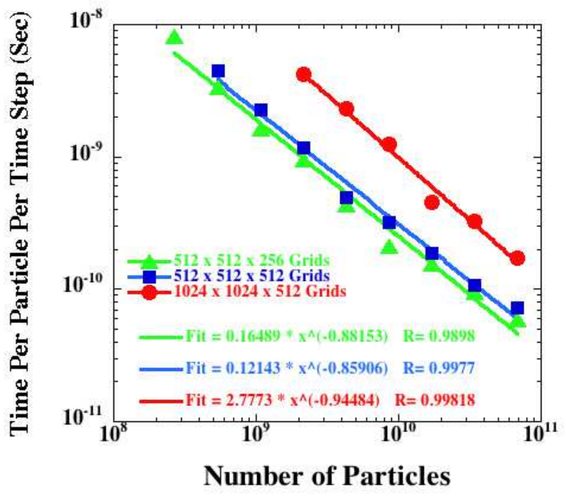

Figure 8: Time per particle per time step in seconds on 65,536 processors at 3 different system sizes of 512×512×256, 512×512×512 and 1024×1024×512 grids with increasing numbers of particles at each system size.

1 nanosecond (1 ns) barrier is broken with 2000 processors or more and from 32 particles per cell and up.

In addition, we have investigated how well UCAN2 performs at half the maximum number of accessible processors on the Cray XC30, namely 65,536. To achieve this we have simultaneously changed the system size and the number of particles at each system size investigated. The result of this exercise is given in Fig. 8.

The data displayed in Fig. 8 show that the time per particle per time step measured in seconds decreases with the number of particles in the calculations performed on a large number of processors fixed at 65,536. Moreover, it is to be noted that the power law fits at each system size in Fig. 8 all have an exponent within 15% of the expected ideal scaling of -1 whereby the time goes down by a factor of 2 if the number of particles is increased by a commensurate factor.

Perhaps the closest working relative of UCAN2 is the global gyrokinetic PIC code GTC [6]. As a matter of fact, the first true 2D decomposition work for both grids and particles was carried out for GTC by Adams, Ethier, and Wichmann [28]. It is therefore warranted to compare the performance of UCAN2 with that of GTC for the commonly used aggregate quantities of time per particle per time step (in seconds) and floating point operations count per processor. Short of case to case correspondence, these comparisons require some manipulations. Recent GTC performance is documented in [29]. From their Fig. 6 and Case C therein, there are 128 nodes with 2 MPI processes per node and 8 OpenMP threads per process for effectively 2048 cores. With 12,871,300 particles per process and 256 processes for a total of 3.3 billion particles, the total time per time step is around 2 sec. This comes to about 1.2 per particle per time step per core or a time of 0.6 ns per particle per time step. It is shown in Fig. 5 of the present paper that UCAN2 gets about 1 ns per particle per time step on 2048 cores of the Cray XC 30. It also appears from Table 4 of [29] that the original GTC is nearly twice as slow as the recent GTC on the IBM Blue Gene mainframe. This entails that UCAN2 performance is equivalent to that of the original GTC. Similarly the oft-quoted number of 1-3 GFlops per processor for the original GTC [30] corresponds at the low-end to the UCAN2 performance of 1 TFlops on 1000 processors or 1 GFlops per processor at the intercept in Fig. 6 of the present paper. This translates into 5% of (per processor) peak for UCAN2 on the Cray XC30. Another measure of performance which is of increased importance for efficient implementation of PIC codes on emerging architectures is memory utilization [31]. From the data of Fig. 5, and more specifically the 1ns per particle per time step actually achieved on 2048 processors of the Cray XC30, each particle carrying 18 double-precision words of information, and this information being accessed 5 times per time step, we estimate a bandwidth to main memory of 0.35 GBytes/s per core or processor (assuming the other memory reads and writes occur in cache). The thus realized memory bandwidth, when compared to the advertised peak of 89 GBytes per node, each with 24 cores or processors, or 3.7 GBytes/s per core or processor, translates into UCAN2 attaining 9.45% of peak memory bandwidth to main memory on the Cray XC 30.

In closing, we show that UCAN2, with the vastly expanded partition and processor space made available by 2D domain decomposition, can readily access the maximum number of processors which is a power of 2, namely $131{,}072=2^{17}$, on both the Cray XE6 and Cray XC30 at NERSC. This is demonstrated in Fig. 9 where the time per particle per time step is displayed as a function of increasing particle number. The system size for these tests was fixed at $512\times512\times512$ grids, with twice as many y-z partitions at 262,144 as number of processors at 131,072, but the number of particles was steadily increased from 4 up to 128 particles per cell or from 536,870,912 particles up to a grand total of

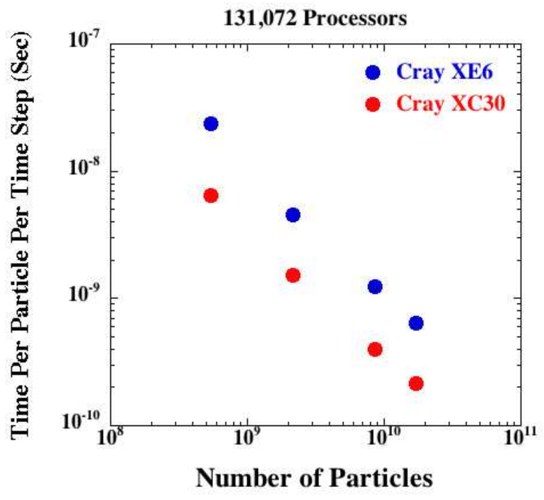

Figure 9: Time per particle per time step in seconds with 131,072 processors at a fixed system size of $512\times 512\times 512$ grid points and increasing number of particles per cell from 4 to 128 plotted versus the total number of particles from 536,870,912 to 17,179,869,184 in the test calculations on the Cray XE6 and Cray XC30 at NERSC.

17,179,869,184 particles. The size and scope of these test calculations which covered only 10 time steps were limited primarily by administrative reasons (scheduling, charging, allocation size). Nevertheless, it is clear from Fig. 9 that scalability, whereby the time per particle per time step decreases when the total number of particles increases, is maintained even at the maximum number of processors accessible with UCAN2 on the Cray mainframes.

The performance studies reported in this portion of the paper have shown that UCAN2 can access without difficulty the vast processor space made available by 2D domain decomposition with a suitable choice of problem size and number of processors, even up to the maximum number of processors which is still a multiple of 2 available on the Cray XC30 at NERSC, namely 131,072 processors. In a production environment, this in turn will greatly facilitate convergence studies with system size and number of particles and also extension of the characterization of transport to problem dimensions compatible with larger scale devices. Indeed, the system sizes of 100 million grid points or more routinely achieved so far with UCAN2 appear adequate for the required resolution of 1000 ion Larmor radii per minor radius at ITER size. The 1 billion particles or more regularly accommodated so far with UCAN2 should also provide sufficient accuracy for useful predictive calculations of ITG turbulence at these scales [6].

5 Discussion

In this paper, we have presented a process to implement 2D domain decomposition in a legacy production code which already contains 1D domain decomposition. While the code in question happens to be a rudimentary yet still useful gyrokinetic particle-in-cell code, we are confident that the implementation strategy and the implementation details are nevertheless of broader interest. We single out two such features which appear to be somewhat novel. One is the algorithm to collect test particle data in the order they were created. Another is with the limited use we make of replicated arrays in the sense that the field solver and particles are still parallelized over 2D domains although the particle deposit and push have replicated data. The 2D domain decomposition for the particles preserves the future possibility of only replicating the grid data needed depending on the extent of the Larmor radius. The broader interest attribute also applies to the demonstration with the thus rejuvenated gyrokinetic PIC code UCAN2 that vastly expanding the processor space with 2D domain decomposition leads to much improved code performance. This is shown to be so in terms of time per particle per time step and floating point operations per second up to very large system sizes, particle numbers, and number of processors in the weak scaling studies reported here. In turn, 2D domain decomposition makes possible production runs at system sizes commensurate with a tokamak of the size of ITER which could not have been accommodated with 1D domain decomposition.

Some of the legacy features remaining in UCAN2 like multiples of 2 in system size, FFTs, and processor number have their drawbacks and could stand to be relaxed as they have already been in the UPIC framework. On the other hand, the particle gather and scatter routines, which are the most compute intensive in a particle-in-cell code such as UCAN2, that were not originally optimized for RISC processors but for vector machines have already been beneficial on the Cray XC30 at NERSC. Moreover this vestigial vectorization positions us well for the future where vectorization appears to be a must to take full advantage of emerging architectures which include vector pipelines [31], such as the Cray-based system known as NERSC-8 for instance. We finally remark that implementation of hybrid programming models, such as MPI plus threads for one, deserves to be explored as a worthwhile path to alleviate the memory issue for large size plasma simulation.

Acknowledgments

This work was supported in part in the USA by Grant No. DE-FG02-04ER54741 to the University of Alaska, Fairbanks, AK, from the Office of Fusion Energy Sciences, Office of Science, United States Department of Energy. It was also supported in part at Universidad Carlos III, Madrid, Spain, by Spanish National Project No. ENE2009-12213-C03-03. This research used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. It also took advantage of resources at the Barcelona Supercomputing Center (BSC), Centro Nacional de Supercomputación, Barcelona, Spain. One of us (Leboeuf) would particularly like to thank David Vicente from BSC and Zhengji Zhao from NERSC for their help in the porting, debugging, and optimization of UCAN2 on the mainframes at their respective centers.

References

[1] A. J. Brizard, and T. S. Hahm, Foundations of nonlinear gyrokinetic theory, Reviews of Modern Physics, 79 (2007), 421-468.

[2] A. M. Dimits, G. Bateman, M. A. Beer, B. I. Cohen, W. Dorland, G. W. Hammett, C. Kim, J. E. Kinsey, M. Kotschenreuther, A. H. Kritz, L. L. Lao, J. Mandrekas, W. M. Nevins, S. E. Parker, A. J. Redd, D. E. Shumaker, R. Sydora, and J. Weiland, Comparisons and physics basis of tokamak transport models and turbulence simulations, Physics of Plasmas, 7 (2000), 969-983.

[3] S. E. Parker, W. W. Lee, and R. A. Santoro, Gyrokinetic simulation of ion temperature gradient driven turbulence in 3D toroidal geometry, Physical Review Letters, 71 (1993), 2042-2045.

[4] R. D. Sydora, Toroidal gyrokinetic particle simulations of core fluctuations and transport, Physica Scripta, 52 (1995), 474-480.

[5] R. D. Sydora, V. K. Decyk, and J. M. Dawson, Fluctuation-induced heat transport results from a large global 3D toroidal particle simulation model, Plasma Physics and Controlled Fusion, 38 (1996), A281-A294.

[6] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White, Turbulent transport reduction by zonal flows: Massively parallel simulations, Science, 281 (1998), 1835-1837.

[7] Y. Idomura, S. Tokuda, and Y. Kishimoto, Global gyrokinetic simulation of ion temperature gradient driven turbulence in plasmas using a canonical Maxwellian distribution, Nuclear Fusion, 43 (2003), 234-243.

[8] S. Jolliet, A. Bottino, P. Angelino, R. Hatzky, T. M. Tran, B. F. Mcmillan, O. Sauter, K. Appert, Y. Idomura, and L. Villard, A global collisionless PIC code in magnetic coordinates, Computer Physics Communications, 177 (2007), 409-425.

[9] S. E. Parker, and W. W. Lee, A fully nonlinear characteristic method for gyrokinetic simulation, Physics of Fluids B, 5 (1993), 77-86.

[10] J. M. Dawson, Particle simulation of plasmas, Reviews of Modern Physics, 55 (1983), 403-448.

[11] J. C. Kniep, Gyrokinetic Particle-in-Cell Simulations of Strong Poloidal Flow Radial Gradient and Parallel Nonlinearity Effects on Toroidal Ion Temperature Gradient Driven Turbulence, Ph. D. Thesis, University of California, Los Angeles, CA, 2006.

[12] C. D. Norton, V. K. Decyk, B. K. Szymanski, and H. Gardner, The transition and adoption of modern programming concepts for scientific computing in Fortran, Scientific Programming, 15 (2007), 27-44.

[13] V. K. Decyk, and C. D. Norton, UCLA parallel PIC framework, Computer Physics Communications, 164 (2004), 80-85.

[14] V. K. Decyk, UPIC: A framework for massively parallel particle-in-cell codes, Computer Physics Communications, 177 (2007), 95-97.

[15] W. W. Lee, Gyrokinetic particle simulation model, Journal of Computational Physics, 72 (1987), 243-269.

[16] J. N. Leboeuf, J. M. Dawson, V. K. Decyk, M. W. Kissick, T. L. Rhodes, and R. D. Sydora, Effect of externally imposed and self-generated flows on turbulence and magnetohydrodynamic activity in tokamak plasmas, Physics of Plasmas, 7 (2000), 1795-1801.

[17] C. L. Rettig, G. M. Staebler, T. L. Rhodes, J. N. Leboeuf, W. A. Peebles, E. J. Doyle, K. H. Burrell, and R. A. Moyer, Search for the ion temperature gradient mode in a tokamak plasma and comparison with theoretical predictions, Physics of Plasmas, 8 (2001), 2232-2237.

[18] T. L. Rhodes, J.-N. Leboeuf, R. D. Sydora, R. J. Groebner, E. J. Doyle, G. R. McKee, W. A. Pee-

bles, C. L. Rettig, L. Zeng, and G. Wang, Comparison of turbulence measurements from DIII-D low-mode and high-performance plasmas to turbulence simulations and models, Physics of Plasmas, 9 (2002), 2141-2148.

[19] J. C. Kniep, J.-N. Leboeuf, and V. K. Decyk, Gyrokinetic particle-in-cell calculations of ion temperature gradient driven turbulence with parallel nonlinearity and strong flow corrections, Computer Physics Communications, 164 (2004), 98-102.

[20] R. Sanchez, D. E. Newman, J.-N. Leboeuf, V. K. Decyk, and B. A. Carreras, Nature of transport across sheared zonal flows in electrostatic ion-temperature-gradient gyrokinetic plasma turbulence, Physical Review Letters, 101 (2008), 205002.

[21] R. Sanchez, D. E. Newman, J.-N. Leboeuf, B. A. Carreras, and V. K. Decyk., On the nature of radial transport across sheared zonal flows in ion-temperature-gradient gyrokinetic tokamak plasma turbulence, Physics of Plasmas, 16 (2009), 055905.

[22] R. Sanchez, D. E. Newman, J.-N. Leboeuf, and V. K. Decyk, On the nature of turbulent transport across sheared zonal flows: insights from gyro-kinetic simulations, Plasma Physics and Controlled Fusion, 53 (2011), 074018.

[23] T. S. Hahm, Nonlinear gyrokinetic equations for turbulence in core transport barriers, Physics of Plasmas, 3 (1996), 4658-4664.

[24] H. Naitou, T. Sonoda, S. Tokuda, and V. K. Decyk, Parallelization of gyrokinetic particle code and its application to internal kink mode simulation, Journal of Plasma and Fusion Research, 72 (1996), 259-269.

[25] C. C. Kim and S. E. Parker, Massively parallel three-dimensional toroidal gyrokinetic flux-tube turbulence simulation, Journal of Computational Physics, 161 (2000), 589-604.

[26] R. Hatzky, Domain cloning for a particle-in-cell (PIC) code on a cluster of symmetric-multiprocessor (SMP) computers, Parallel Computing, 32 (2006), 325-330.

[27] K. Madduri, S. Williams, S. Ethier, L. Oliker, J. Shalf, E. Strohmaier, K. Yelick, Memory-efficient optimization of gyrokinetic particle-to-grid interpolation for multicore processors, Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09), IEEE Computer Society, Los Alamitos, CA, 2009, pp. 1-12. http://www.computer.org/csdl/proceedings/sc/2009/8744/00/87440048-abs.html

[28] M. F. Adams, S. Ethier, and N. Wichmann, Performance of particle in cell methods on highly concurrent computational architectures, Journal of Physics: Conference Series, 78 (2007), 012001.

[29] B. Wang, S. Ethier, W. Tang, T. Williams, K. Ibrahim, K. Madduri, S. Williams, and L. Oliker, Kinetic turbulence simulations at extreme scale on leadership-class systems, Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13), Association for Computing Machinery, New York, NY, USA, 2013, Article No. 82. http://dl.acm.org/citation.cfm?doid=2503210.2503258

[30] L. Oliker, A. Canning, J. Carter, J. Shalf, and S. Ethier, Scientific application performance on leading scalar and vector supercomputering platforms, International Journal of High Performance Computing Applications, 22 (2008), 5-20.

[31] V. K. Decyk, and T. V. Singh, Particle-in-cell algorithms for emerging computer architectures, Computer Physics Communications, 185 (2014), 708-719.