Accurate and Efficient Multi-scale Flow Simulation using a Hybrid CFD-MD Approach

Soon-Heum Ko[b,a], Nayong Kim[a], Dimitris Nikitopoulos[d], Dorel Moldovan[d], Shantenu Jha[a,c,*]

[a] *Center for Computation & Technology, Louisiana State University, Baton Rouge, LA 70803, USA*
[b] *National Supercomputing Centre, Linköping University, Linköping, SE-581 83, Sweden*
[c] *Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA*
[d] *Department of Mechanical Engineering, Louisiana State University, Baton Rouge, LA 70803, USA*

Abstract

We investigate numerical and computational issues associated with hybrid computational fluid dynamics (CFD) - molecular dynamics (MD) simulation methodologies. Our current hybrid CFD-MD simulation framework is based on reliable CFD and MD codes linked together through hybrid interfaces that are often based on constrained Lagrangian dynamics for data exchange between continuum and MD regions. We argue that (1) the statistical error associated with the sampled average of the molecular solution, and (2) hybrid boundary conditions via an extrapolation methodology, may lead to non-negligible spatial and temporal inaccuracies of the hybrid solution. Hence, in this study we propose two strategies for minimizing the statistical noise inherent in the MD solution: (1) optimization of coupling parameters through the molecular dynamic simulation of a stationary flow field, and (2) averaging over multiple independent replicas of the simulation system. In addition, we introduce a temporal coupling scheme based on a 'prediction-correction' way together with interpolation in time, which provides a more time-accurate hybrid constraint/boundary condition. With

*Corresponding author, (Tel)+1-225-578-XXXX; (FAX)+1-225-578-XXXX
Email addresses: sko@nsc.liu.se (Soon-Heum Ko), nykim@cct.lsu.edu (Nayong Kim), meniki@me.lsu.edu (Dimitris Nikitopoulos), moldovan@me.lsu.edu (Dorel Moldovan), sjha@cct.lsu.edu (Shantenu Jha)

respect to computational performance, co-scheduling and load-balancing of logically distributed tasks are two important considerations in CFD-MD coupled simulations. We employ a BigJob computational framework, similar to a Pilot-job, and incorporate a load-balancing function for the computing efficiency of coupled simulation. The optimized hybrid framework is applied to solving two classical flow problems under Stokes (no interial effects) conditions. One is the impulsively started Couette flow, and the other is an oscillating boundary Stokes flow, which is induced by the periodic motion of one flat plate in the channel system. We emphasize that (1) the determination of coupling parameters through the molecular dynamic simulation of a stationary flow and the averaging of multiple independent replicas contribute much on suppressing the statistical noise in relatively low-speed flow fields of O(10) m/s velocity, (2) the accuracy of unsteady flow solution is increased by applying a 'prediction-correction' temporal coupling scheme, and (3) efficient coupled simulation is conducted by employing a BigJob computational framework along with the load-balancing capability.

Keywords: Hybrid computational fluid dynamics - particle dynamics approach, Constrained Lagrangian dynamics, Temporal coupling scheme, Simple API for Grid Applications (SAGA), BigJob abstraction

---

1. Introduction and Motivation

A hybrid continuum dynamics - particle dynamics approach is an approach capable of describing accurately a flow at both macroscopic and molecular scales. In this approach the system is divided into two domains. A continuum formulation is used appropriately for one domain and a particle formulation (e.g. molecular dynamics) is applied to the other. Naturally the particle domain is more appropriate for material interfaces (e.g. fluid/solid or fluid/fluid) where molecular effects are more likely to be important. On the other hand, the continuum approach provides a better computational efficiency than particle dynamics with acceptable accuracy in solving the bulk flow field. In the hybrid approach the two domains are coupled through an overlap region/interface in which both formulations are valid. The overlap domain enables the exchange of conservative flow properties between the continuum and particle ones so that the respective solutions are mutually consistent. The information from the continuum domain is passed to the particle domain by imposing additional numerical modeling to preserve

higher degree-of-freedom on the particle motion, while the particle domain provides boundary conditions to the continuum domain obtained through time and spatial averaging of the relevant variable.

Despite the recent developments reported in the literature, most of which tested against small idealized pure atomistic flow simulations for comparison, there are still methodological and implementation issues that need refinement, testing and validation. In this section, we will give a brief account of the numerical and computational issues associated with some of the previous hybrid CFD-MD implementations and outline our approach to address them effectively.

1.1. Previous Hybrid CFD-MD Implementations

Hybrid CFD-MD simulations have utilized constrained Lagrangian dynamics [1–7], the Schwarz method [8–12], or a direct flux exchange [13–18] method, depending on the methodology used to impose the solution matching in the overlap region/interface and the nature of the variables that exchange information in this region. O'Connell and Thompson [1] were among the first to implement a hybrid CFD-MD simulation approach by introducing the overlap region that allows matching of the solutions from the two domains to relax smoothly before they are coupled together. Namely, they employed a relaxation method according to which the average MD velocity in an overlap region is forced to follow the continuum solution in the same region. In their implementation they use constrained Lagrangian dynamics according to which the particles are accelerated/decelerated in the overlap region so that on average they follow the continuum velocity. The drawback of their implementation is the arbitrariness of the relaxation rate and the lack of particle exchange (particle flux) between the two domains. This approach is refined by Nie *et al.* [2] who imposed the spatial coupling between continuum equations and molecular dynamics through constrained dynamics implemented in the overlap region. The methodology has been applied to a variety of flow problems such as the impulsively started Couette flow [1], channel flow with rough walls [2], cavity flow [3], Poiseuille flow [6] and the oscillating boundary problem [5, 7]. The use of direct constrained dynamics equation makes this approach easy to implement and computationally efficient. However, the absence of energy exchange modeling limits the applications to isothermal systems [13].

To alleviate some of these shortcomings, Hadjiconstantinou *et al.* [8–10] have introduced a new hybrid simulation methodology based on the Schwarz

method. In this approach, the continuum solution is used to generate the solution in the particle domain in which the particle velocities are drawn from a Maxwellian distribution such that the mean velocity and the corresponding standard deviation are determined by the continuum solution and temperature [9]. This approach later was expanded to non-periodic boundary condition problems by Werder *et al.* [11], who also expanded and refined the previous boundary force models [1, 2, 13, 15] so as to minimize local disturbance. An alternating Schwarz approach has been applied to the moving contact line problem [9], Poiseuille flow [10], flow around the cylinder [11], and Couette flow of water [12]. The characteristics of decoupled physical time-scales between two domains makes this approach appealing when solving the flow field in problems in which hydrodynamic characteristic time scale is much larger than molecular dynamic time scale, i.e., micrometer system size [9].

In order to be able to simulate isothermal compressible flow, Flekkϕy *et al.* [13] introduced a new hybrid method based on continuity of mass and momentum fluxes across the MD-continuum interface, which later included energy transfer [14, 15]. Further developments were implemented by Delgado-Buscalioni and Coveney [16] by introducing a particle insertion algorithm which satisfies the continuity of the mass flux along the interface while preserving the mean potential energy of the system. This approach has been applied to the study of Couette and Poiseuille flow [13], transversal wave [15], and oscillating boundary problem [17]. According to Flekkϕy *et al.* [13], flux exchange directly implies adherence to the relevant conservation laws without the use of constitutive relations and equations of state (to maintain the conservation laws). However, it is pointed out that the sampling time to measure fluxes within acceptable statistical error is orders of magnitude larger than the time to measure densities [10].

1.2. Numerical and Computational Issues Associated with Hybrid CFD-MD Methodologies

Despite the recent developments and obvious advantages of a hybrid approach for flowing systems with scales straddling molecular to macroscopic (continuum) magnitudes over pure CFD or MD, a number of numerical and computational difficulties prevent this technique from being widely used. One of the difficulties is the lack of a clear methodology for defining an optimized set of parameters related to the coupling of the continuum and molecular domains. In general the flow solution obtained by a direct hybrid approach

suffers from the existence of significant noise in the spatially averaged particle velocity mainly due to the inherent statistical fluctuations in the molecular description. Therefore, coupling parameters such as the size of the overlap domain and its position, the width of various layers and bins in the overlap region, and the sampling conditions, should be appropriately determined a priori in order to achieve more accurate numerical solutions. Many of these coupling parameters are problem dependent and this complicates the problem even further. Given the stochastic nature of the noise characterized by the mean velocities in the molecular domain of the coupling region, the characteristics of fluid and solid elements, as well as the geometric characteristics of material (e.g. fluid/solid) interfaces, it is difficult to develop a mathematical model capable of predicting the optimum coupling parameters. There have been some attempts to mitigate the inaccuracies associated with inherent statistical fluctuations by introduction of certain numerical damping terms in the equations of motion of the atoms in the overlap region [1, 4] or by defining a specific dynamic parameter to controlling the coupling intensity [5], but all of these eventually lead to some violation of energy and momentum conservation.

Important issues remain to be addressed in the hybrid implementations for studying flow physics at moderate or low velocity conditions. So far, most of the reported studies have been confined to relatively high-speed flow conditions, of the order of 100 m/s, in nanoscale systems; flow conditions chosen mainly to obtain a high speed shear rate in the overlapping region and, more importantly, to reduce the relative stochastic thermal noise associated with the velocity flow field [6]. According to the mathematical models [10, 17] the noise level characterizing the average velocity in the coupling region in a hybrid CFD-MD implementation is proportional to $1/{u_\infty}^2$, where the u_∞ is the far field stream velocity, and to $1/N_0$, where N_0 denotes the number of particles contained in the sampling layer. Therefore, by reducing the free stream velocity by, say, 10 times, in order the maintain the same noise level (same statistical error) would require the increase of the number of particles in the averaging bins by 100 times, assuming that the sampling duration is maintained constant. In general the increase of the sampling duration beyond the limits imposed by the physical time-scales of a given unsteady flow problem is to be avoided because it leads to unphysical effects.

The accuracy of temporal coupling schemes is of great importance when solving unsteady problems. In general the temporal coupling scheme encompasses the timing of data exchange between the CFD and MD solvers,

so as to synchronize both solutions at the same physical time instant. In this one has to keep in mind that the interpretation of the term "instantaneous" regarding values of variables differs between the continuum and molecular formulations. The time-coupling models are inherently affected by time-lagging. This is more pronounced when backward-averaged molecular dynamic properties are in general communicated as instantaneous properties to the continuum domain. To address this shortcoming, Liu *et al.* [7] proposed a multi-timescale algorithm by considering quasi-steadiness and Wang and He [5] proposed passing the extrapolated continuum solution to the MD solver. Despite their success there are, however, serious limitations with these approaches as a multi-timescale algorithm is only valid if the characteristic time of the unsteady phenomenon is substantially longer than the integration (averaging) time of the MD solutions. This is also a prerequisite for the particle (MD) solution to be able to equilibrate subject to the constraint passed on from the continuum solution from one continuum time instant to another. The approach by Wang and He may therefore be a good compromise that may mitigate some of the time-lagging effects.

In addition to the above mentioned issues, there are computational challenges related to the integration of multiple application domains into a single problem set. Due to their very different computational kernels (e.g. one mesh-based, the other an unstructured particle-based), it is difficult to incorporate distinct CFD and MD codes under the umbrella of a single tightly-coupled application (i.e. unifying two application codes to share a single MPI communicator). One possible alternative can be to implement a coupling interface into one of the individual codes and design it such that this assumes control of the two separate codes as a virtually unified simulation package. This heterogeneous implementation raises additional computational challenges. For example, in the parallel execution of such a simulation package, on conventional production systems with batch queues, it cannot be guaranteed that two separate jobs will execute concurrently. Considering the computational characteristics of the current application, where the CFD and MD codes conduct frequent data exchange, the first job that completes a task will inevitably experience the idle waiting for its counterpart to finish its sequenced task without the explicit support for co-scheduling. Another important challenge is the need for efficient load-balancing which takes into account the individual application's performance. Even if the two simulations could run concurrently, without explicit load-management/balancing support, there is likely to be inefficient utilization of computing resources

due to load imbalance. As the performance of each simulation component changes with computing resource and problem size, re-adjustment of allocated resources to each task according to their performance is required during the hybrid simulation.

1.3. Objectives and Outline of the Paper

The focus of this paper is to investigate numerical issues associated with the implementation of a typical hybrid CFD-MD methodology applied to investigate the flow in a nano/micro-fluidic system as well as with designing and developing of an efficient runtime framework for coupled multi-scale simulations. We present our implementation of a 'generic' hybrid CFD-MD interface which can be easily put together from various 'off the shelf' incompressible CFD and MD packages, as well as our model of a 'portable' framework that can be ported easily on most present day computer architectures.

We present the development of our hybrid simulation framework, based on constrained Lagrangian dynamics, and apply it to two prototype classical flow problems: the impulsively started Couette flow and an oscillating boundary Stokes flow. From the pure MD solution on zero-velocity flow field, we elaborate on our strategy for determining the optimized set of coupling parameters that minimize the level of the statistical noise characterizing the flow field obtained by the hybrid solver. In addition we extend our investigation into the moderate-speed flow regime (flow velocities of the order of 10 m/s), which is challenging and novel. The inherent large fluctuations of the solution in the overlap region associated with relatively low flow speed are reduced by a multiple replica averaging methodology. For the oscillating boundary Stokes flow simulation, we demonstrate a novel temporal coupling scheme, called 'prediction-correction approach' which provides improved accuracy by eliminating the overshoot/undershoot phenomena and diminishes the time-lagging pattern.

In terms of computational science, our study introduces a novel approach to coupled multi-physics simulations by utilizing the virtually unified simulation package, called "Pilot-Job". We argue that using the Pilot-Job approach has distinct advantages, such as: (i) elimination of the need for a co-scheduler while preserving performance, and (ii) enables dynamic resource allocation, which in turn is important for load-balancing across coupled simulations.

The paper is organized as follows. In Section 2 we give a brief description of the fundamental equations describing the CFD and MD formulations as

well as details of the hybrid CFD-MD coupling scheme and its implementation. Technical details related to the implementation of an efficient CFD-MD simulation framework are presented in Section 3. In Section 4, we present the solutions to the two prototype problems (impulsively started Couette flow and oscillating boundary Stokes problem) obtained with our hybrid methodology. The performance and the optimization issues related to the solutions of the two test problems are described in Section 5. Recommendation for future work and conclusions are presented in Sections 6 and 7.

2. Fundamentals of the Hybrid Coupled CFD-MD Simulation Toolkit

The accuracy of a hybrid CFD-MD solution is governed by the underlying numerical schemes of the individual continuum and particle solvers as well as the coupling scheme. This section details and explains the features of baseline solvers and the structure of the hybrid coupling interface. Additional numerical treatments to improve the accuracy of hybrid CFD-MD simulations in various types of fluid systems are also addressed.

2.1. Overview of the Continuum-Molecular Coupled Approach

The hybrid CFD-MD approach stands on the proposition that (a) the molecular dynamic simulation produces a more physically accurate solution in its domain than the continuum model would, (b) a faster and more efficient CFD solver provides a physically accurate solution in the continuum domain, which would be prohibitive if a particle representation were to be used, and (c) the two solutions can be reconciled by the mutual exchange of information, the continuum solution providing the constraint to the particle solution while the latter providing a boundary condition to the former. So, the region within which molecular-level physics are important (e.g. a material interface) is treated and resolved by a molecular dynamic simulation, while the region within which the continuum-level physics adequately represent the transport processes on a macroscopic scale is treated and resolved by the continuum CFD. The continuum and particle (molecular) domains overlap, in this approach, so as to exchange information for the coupling. Obviously, the overlap domain must be one where both treatments/formulations are valid.

A detailed structure of the various domains associated with the hybrid CFD/MD methodology is described in Fig. 1. CFD solves the continuum equations in the "pure CFD region" while MD resolves the microscopic (particle) flow in the "pure MD region" The latter is a solid stationary wall

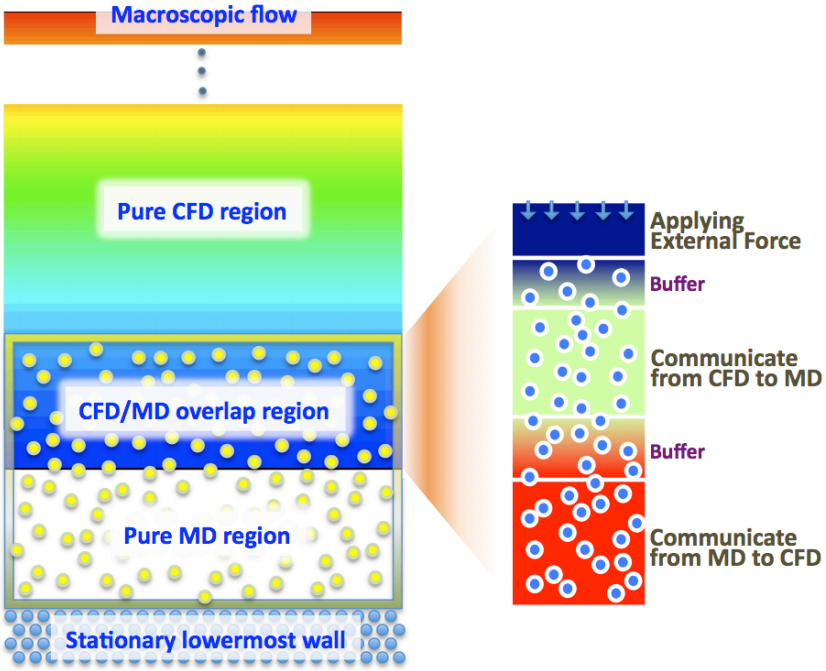


Figure 1: **Schematic Diagram of the Hybrid Domain with Detailed View of Overlapping Zone** ; Overall continuum/atomistic and hybrid computational domain including overlap region is presented in the left figure. Detailed layer by layer explanation of overlapping region is indicated by the right figure.

for the two test problems examined herein. The hybrid (overlap) region is placed sufficiently far from the solid wall to prevent direct constraint of the molecular-level physics in the proximity of the solid-fluid interface.

This overlap region is designed sufficiently large to contain five individual layers of sufficient width for the purpose of their existence. The widths of these layers are of course problem dependent. Their roles in the hybrid region are as follows. One layer (the bottom one in Fig. 1-left) provides information from the particle (MD) domain to the continuum one (denoted as "MDtoCFD". A second layer (the third one from the bottom in Fig. 1-left) provides information from the continuum domain to the particle one (denoted as "CFDtoMD". A third layer (the top one in Fig. 1-left) applies a fictitious external force which prevents particles from escaping from the hybrid domain. These three "active" layers are separated by buffer layers, which are placed so as to ensure smooth transitions and that there is no direct correlation between the "active" layers. Properties (e.g. velocity) of particles

spatially located in the MDtoCFD layer are ensemble-averaged over all the particles in the layer and also averaged over a finite time. The average value is communicated to the continuum solver periodically with a fixed period.

The height of each layer is the same as the CFD cell height and averaged conservative properties in two consecutive layers are passed to continuum domain to be directly imposed as the viscous boundary condition on the Navier-Stokes solver with collocated data structure. The CFDtoMD layer imposes the instantaneous values of properties, velocity in this case, resulting from the continuum solver. This is done via and appropriate constraint to the MD particle-based conservation of momentum on every single particle (constrained dynamics) in the CFDtoMD layer. Thus, particles in this layer are constrained to attain the macroscopic flow property (velocity in this case) on average, while preserving their degree-of-freedom of translational motion. In the uppermost layer, a fictitious external force is exerted on particles to prevent them from escaping the particle domain. This force function is designed to be short-range so as not to be strong enough to influence the motion of the particles past the buffer layer in the CFDtoMD domain. The force stiffens as the particles approach the location where the force becomes infinite in a way that minimizes reflections while preventing the particles from drifting out of the particle domain. The buffer layers existing in between each "active" layer are set up to be wider than the interaction length scale of the particles (cutoff radius), in order to prevent direct interaction between particles in neighboring "active" layers.

2.2. Governing Equations and Numerical Schemes

2.2.1. Continuum Incompressible Flow Formulation (CFD)

The current in-house continuum hydrodynamics code solves the unsteady incompressible Navier-Stokes equations:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(u_i u_j) = -\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j}$$

where ν is the kinematic viscosity.

In this work, the pseudo-compressibility method [19] is adopted to form a hyperbolic system of equations which can be marched in pseudo-time. A time derivative of pressure is added to the continuity equation resulting in

$$\frac{\partial(p/\rho)}{\partial\tau} = -\beta\frac{\partial u_i}{\partial x_i} \tag{2}$$

where β denotes a pseudo-compressibility parameter, currently set up to 2.5.

For time-accurate unsteady simulation, a dual time stepping method is adopted and it is combined with the LU-SGS (Lower-Upper Symmetric Gauss-Seidel) scheme [20] for the implicit time integration. The inviscid fluxes are upwind-differenced using Osher's flux-difference splitting scheme [21]. For higher-order spatial accuracy, the MUSCL (Monotone Upstream-centered Schemes for Conservation Laws) [22] approach is used on an inviscid flux calculation. Viscous fluxes are calculated using conventional second-order central differencing.

2.2.2. Molecular-Level Formulation (MD)

In MD, an initial velocity is assigned to each atom, and Newton's conservation of momentum is employed at the atomic level to propagate the system's motion through time evolution. In this work the most commonly used Lennard-Jones (12-6) intermolecular force potential model is employed to calculate pair-wise interactions of particles in the system, and is defined as:

$$u(|r_i - r_j|) = 4\epsilon_{ij}[(\frac{\sigma_{ij}}{r_{ij}})^{12} - (\frac{\sigma_{ij}}{r_{ij}})^{6}] \tag{3}$$

where ϵ_{ij} and σ_{ij} denote the pair-wise potential well depth and the atom size parameter respectively, and rij is the distance between the particle i and j. The repulsive term $1/r_{ij}^{12}$ dominating at short rij distance is based on the Pauli principle to avoid overlapping the electronic clouds when particles are very close to each other. The attractive term $1/r_{ij}^{6}$ dominates at long range representing Van der Waals dispersion forces. A cut-off distance σ_c is introduced here to reduce the computational cost and is set to be 2.2σ [28]. Namely when rij exceeds the cutoff the intermolecular force is set to zero without being calculated. The most common velocity Verlet algorithm is employed for time integration of the equations of motion of the interacting particles and to compute molecular trajectories in the simulation. In this work, the MD simulations were performed by using an appropriately modified version of the Large Atomic Molecular Massively Parallel Simulator (LAMMPS). It is a classical molecular dynamics open-source code written in C++ and developed by Sandia National Labs [23].

Table 1: **Implementation of hybrid interface on CFD and MD codes.** Both codes are equipped with the file-based information exchange routine, to update the hybrid boundary condition. CFD code experiences the global change of its data structure to store the information of the entire fluid system. MD code adopts hybrid equations to impose the macroscopic information on microscopic domain and to ensure numerical stability.

	CFD	MD
Global Change	Overset Data Structure (*optional)*	-
External Force	-	External Force Equation (Eqn. 4)
CFDtoMD	File Interface: Sender	File Interface: Receiver Constrained MD Equation (Eqn. 9)
MDtoCFD	File Interface: Receiver	File Interface: Sender

2.3. Hybrid Interfaces and Schemes

Hybrid simulation requires the implementation of hybrid interfaces and schemes based on an individual code. In the current study, the file interface is designed to schedule the information exchange between continuum and discrete particle descriptions. A constrained Lagrangian dynamics model is implemented for hybrid simulation. A unit conversion routine is also implemented in the application code. These changes are summarized in Table 1.

The CFD code employs the data structure of the overset mesh technique [24] to ease the handling of the coupling parameters. In other words, the entire fluid domain is generated because the CFD mesh system and pure MD region is turned off as the "hole" cell in the terminology of the overset technique. Likewise, MD to CFD and CFD to MD boundary cells are declared "fringe" and "donor" cells, respectively. The labor of mesh regeneration due to the change in coupling parameters (the position and depth of the hybrid layers) disappears with the overset data structure.

Both codes are equipped with the information exchange routine, which consists of one file sender and one file receiver. These file interfaces are scheduled to turn on every sampling interval. The instantaneous properties in the donor cells of the continuum domain are transferred to the MD site and referenced when the constrained Lagrangian dynamics equation is applied. The averaged molecular properties are sent to the CFD domain, and they are used as the boundary conditions of the fringe cells. All exchanged properties

are written in the MD unit: thus, the CFD code is equipped with a velocity unit conversion function and an equation of state, which changes the pressure solution from the CFD site to the equivalent density property in the MD domain.

In addition to the file interface, additional equations of motion are employed on the MD code to describe accurately the influence of macroscopic flow variation on the particle domain. First, external force should be imposed to prevent the particles from leaving the control domain; the force is applied in the normal direction of the uppermost MD layer. A cost-effective classical external force model by Nie *et al.* [2] is employed as

$$F_{ext,i} = -p_a \sigma \frac{y_i - Y_{n-1}}{1 - (y_i - Y_{n-1})/(Y_n - Y_{n-1})} \tag{4}$$

where p_a denotes the average pressure in the MD region, $Y_n - Y_{n-1}$ is the thickness of the uppermost layer to which the force is applied, and F_{ext} is the external force acting on ith particle located on position y_i.

Next, at a specific time, the macroscopic flow properties are introduced on the CFD to MD layer to lead the motion of multiple particles in that layer. A very complicated numerical intervention is required to maintain the momentum conservation. The averaged velocity of particles in J_{th} cell equals to the velocity u_J in the continuum cell.

$$u_J(t) = \frac{1}{N_J} \sum_i v_i \tag{5}$$

where v_i is the velocity of i^{th} particle and N_J is the number of particles in the cell. The Lagrangian derivative of Eq. 5 obtains,

$$\frac{Du_J(t)}{Dt} = \sum_i \frac{\ddot{x}_i}{N_J} \tag{6}$$

The Classical MD equation of motion can be generalized to obtain the constraint by adopting the fluctuation in the acceleration of each particles, ζ_i

$$\frac{F_i}{m_i} = \ddot{x}_i(t) = \frac{Du_J(t)}{Dt} + \zeta_i = \frac{\sum_i F_i(t)}{\sum_i m_i} + \zeta_i \tag{7}$$

where F_i is the force on i^{th} particle based on the interactions between particles, m_i is mass of each atom and Eq. 8 satisfies,

$$\sum_i \zeta_i m_i = 0 \tag{8}$$

Finally, the constrained particle dynamics with the conventional equation of motion can be written as:

$$\ddot{x_i}(t) = \frac{F_i}{m_i} - \frac{\sum_i F_i(t)}{\sum_i m_i} - \frac{1}{\Delta t_{MD}}\{\frac{\sum_i m_i \dot{x_i}}{\sum_i m_i} - u_J(t + \Delta t_{MD})\} \tag{9}$$

The continuum velocity and the mean microscopic velocity from MD over the control domain provide the synchronization of the mass and momentum consistent with Eqn. 9.

2.4. Statistical Error and Coupling Parameters

One of the standard issues in obtaining coupled continuum to particle-based (MD) solutions is the fact that the particle-based (MD) solutions have noise, inherent because of limited size of particle ensemble averages and time-sampling limitations. This noise, if not kept under check, is passed on to the continuum solution yielding unphysical results. According to the mathematical expressions in statistical error [10, 17], the ratio of sampling noise compared to the macroscopic velocity is inversely proportional to the square root of spatial layer size and temporal sampling duration. For example, reducing the macroscopic velocity by half requires either 4 times larger system domain or 4 times longer sampling to maintain the same order of accuracy.

Four important coupling parameters determine the scale and pattern of the sampling noise: The size of sampling layer and its location in space, sampling duration with interval in time. The layer size denotes the volume of a virtual box in which spatial sampling takes place. The layer location is the perpendicular distance of the sampling layer from the solid obstacle. The sampling duration Δt_s is denoted as the time length over which spatially-sampled instantaneous solutions are accumulated. The sampling interval Δt is defined as the time increment between each hybrid solution exchange. The layer size and sampling duration collevtively contribute to the suppression of sampling noise by increasing the number of sampled particles in space or time. The layer size is free to be stretched along the periodic direction, while

it results in excessive computational cost. Incresing the layer size along the non-periodic direction is relatively constrained to secure the sufficient space for pure CFD and MD domains. The scale of sampling duration is bound by the sampling interval, which is desired to be short for accurately describing the temporal variation of flow evolution in the unsteady flow field. The location of the sampling layer also influences the size of fluctuations. It has been noted that the sampling layer shall be placed sufficiently far from the solid obstacle to avoid the effect of strong interaction between fluid and solid particles [2, 6].

We reference to the determination of coupling parameters in previous studies [2, 6–9, 11, 13, 15, 17] and intuitively design these parameters considering our application targets. We start from determining the location of the sampling layer according to other previous suggestions, e.g., at least 10σ above the bottom wall in the nano-scale liquid argon system. The height of the sampling layer on the non-periodic direction is designed in a way that the sum of pure MD and hybrid regions does not exceed the half of the fluid system. For our deterministic transient/periodic applications, the sampling interval is roughly set to be within 1% of the transition time or 10% of the period. The sampling duration is designed the same as the sampling interval.

We also apply the replica sampling approach [25] to supplement the individual hybrid solution. Instead of changing coupling parameters for solving the same fluid system different flow conditions, we simulate the same coupling configuration with the variance in the initial Maxwell-Boltzmann distribution and average these replicas to find the non-fluctuating solution. In principle, the sampled solution from N replicas has the same order of accuracy as the individual solution from N times larger domain. The superiority of replica simulations over a single large-scale simulation can be found from the computational point of view: multiple moderate-sized jobs are more prone to get faster allocation than a single excessive resource requirement on most supercomputing batch queues. For the more, this approach is free from the geometric complexity so that it can be applied to the non-periodic three-dimensional flow simulations in which the virtual increase of sampling size is impossible.

2.5. Temporal Coupling Schemes

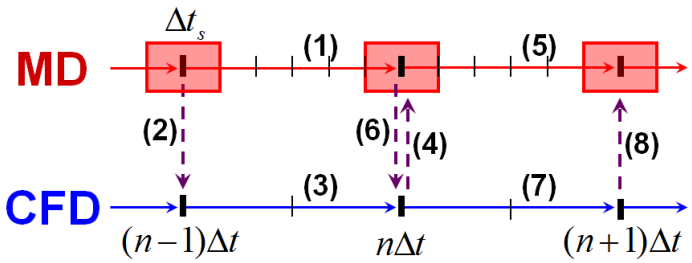
Synchronized and sequential coupling strategies [17] are conventional temporal coupling schemes, which are depicted in Fig. 2. In synchronized coupling, CFD and MD codes exchange their hybrid boundary conditions at the

same physical time step and individually evolve by a sampling interval. In constrast, temporal evolution of each solver takes place one-by-one in sequential coupling, in a fashion that MD solver advances to the CFD solver by a sampling interval.

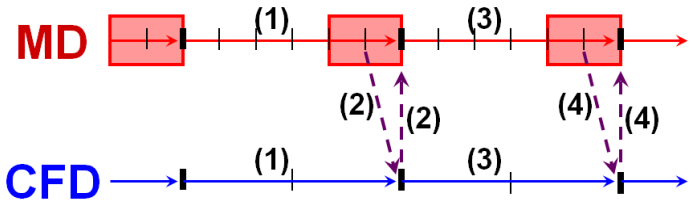
The synchronized coupling approach is recommended in view of parallel efficiency. As can be noticed in the procedure of the sequential approach in Fig. 2-(a), prossessors statically assigned to individual task cannot avoid idling while another code is evolving to the next communication point. It necessitates the integration of distinct codes to a single MPI executable or the switching of allocated resource to the active task during runtime. On the other hand, the synchronized approach introduces the unfavorable time-lagging phenomenon on CFD boundary condition. As observed in Fig. 2-(b), molecular dynamic properties averaged over the backward sampling duration (from $n\Delta t - \Delta t_s$ to $n\Delta t$) represent the CFD boundary condition at that instance ($n\Delta t$).

Wang and He [5] proposed extrapolating hybrid boundary conditions to resolve the temporal delays in synchronized coupling approach, as seen in Fig. 2-(c). In their approach, molecular dynamic time axis is shifted forward by half of the sampling interval Δt to get rid of the time-lagging event in CFD boundary condition. Also, MD hybrid boundary condition is imposed by the extrapolated CFD solution in time. The benefit of this approach is that, in principle, time-lagging effect completely diminishes regardless of the sampling duration. Thus, the sampling duration can be set as much as the sampling interval. However, the accuracy of the extrapolated solution is still debatable. In this scheduling, two *previous* CFD solutions at $n\Delta t$ and $(n-1)\Delta t$ are extrapolated to predict the *expected* MD hybrid boundary conditions from $(n+\frac{1}{2})\Delta t$ to $(n+\frac{3}{2})\Delta t$. The extrapolated property is prone to be incorrect from the start except when the velocity variation is linear in time. In such cases, shortening the sampling interval Δt is a way of reducing this extrapolation error. It, in turn, results in the short sampling duration, which is unfavorable for reducing the statistical error.

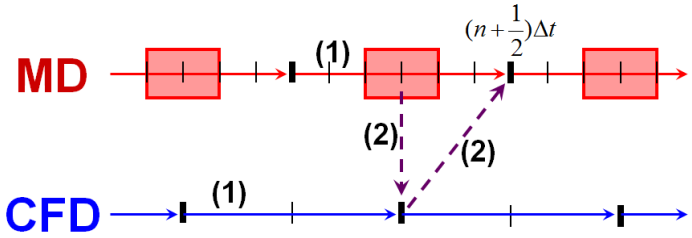
A "prediction-correction approach" is designed to reduce this extrapolation error by introducing the *predicted* continuum solution at later time instance, as presented in Fig. 3. The prediction-correction approach numerically provides non-delayed hybrid boundary conditions, like Wang and He's formulation. Thus, the sampling duration is designed as long as the sampling interval. The main difference from Wang and He's approach is that the CFD code iterates additional time intervals after it reaches the next communica-



(a) Sequential Coupling



(b) Synchronized Coupling



(c) Synchronized Coupling with Extrapolation [5]

Figure 2: **Conventional time evolution mechanisms of a hybrid CFD-MD approach.** CFD and MD codes are scheduled to conduct data exchange in the overlapping region at every Δt sampling interval. (a) MD code first advances to the next communication point ($n\Delta t$) and passes the centrally-sampled data around $(n-1)\Delta t$ to CFD solver so that CFD solver evolves to the $n\Delta t$. One code stays idle while another code evolves. (b) Both codes individually evolve to the communication point ($n\Delta t$) and exchange individual solution. CFD passes the instantaneous solution at $n\Delta t$ while the backward time-averaged molecular dynamic sample is imposed as the CFD boundary condition at that time. (c) MD code approaches $(n+\frac{1}{2})\Delta t$ while CFD solver evolves from $(n-1)\Delta t$ to $n\Delta t$. Centrally-sampled MD solution around $n\Delta t$ is supplied as the CFD boundary condition from $n\Delta t$ to $(n+1)\Delta t$. CFD solution at $n\Delta t$ is extrapolated for the MD boundary condition from $(n+\frac{1}{2})\Delta t$ to $(n+\frac{3}{2})\Delta t$.

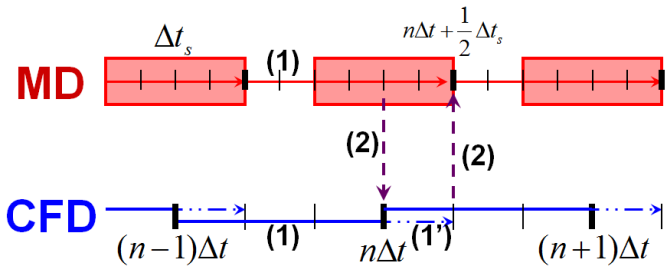
tion point. For example, in Fig. 3-(a), the CFD code additionally evolves by half of sampling duration after it approaches $n\Delta t$. The CFD code sends these predicted properties at $n\Delta t + \frac{1}{2}\Delta t_s$ to the MD site and receives the centrally-sampled molecular dynamic solution around $n\Delta t$. The CFD code rolls back to its previous flow profile at $n\Delta t$ and applies the received molecular dynamic solution to advance to the next communication point. A clear benefit of the current approach is that both solvers extrapolate their boundary conditions from *current* solutions instead of relying on *previous* history. This advantage eliminates the sensitivity of the extrapolated solution to the size of the sampling duration, which enables the increase of the sampling duration to the sufficient level.

The increase of the prediction time scale further increases the accuracy of the hybrid boundary condition. In Fig. 3-(b), the prediction time scale is increased by one more sampling intervals. While the MD solver evolves for one sampling interval from $(n-1)\Delta t+\frac{1}{2}\Delta t_s$ to $n\Delta t+\frac{1}{2}\Delta t_s$, the CFD code runs the actual evolution from $(n-1)\Delta t$ to $n\Delta t$ and adds the predictive evolution for $\Delta t + \frac{1}{2}\Delta t_s$ time interval. Predicted CFD solution at $(n+1)\Delta t + \frac{1}{2}\Delta t_s$ is provided for the interpolated hybrid boundary condition on MD solver. from $n\Delta t + \frac{1}{2}\Delta t_s$ to $(n+1)\Delta t + \frac{1}{2}\Delta t_s$. Figure 3-(c) demonstrates the imposition of implicit boundary conditions on both domains. In this formulation, CFD time is shifted backward by one sampling interval and the prediction step is scheduled as $2\Delta t + \frac{1}{2}\Delta t_s$.

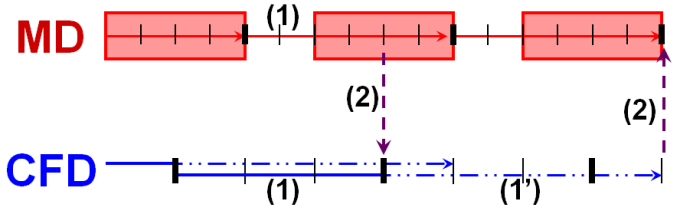
The current numerical approach provides more accurate time-variant solution by decreasing or eliminating the unfavorable overshoot/undershoot phenomena in extrapolations. Nevertheless, an additional computational cost is inevitable for the CFD simulation. Thus, the current approach is applicable to the problems in which computational cost on CFD is quite smaller than that of MD simulation.

3. Coupled Concurrent Multi-Scale (Continuum-Molecular) Simulation Framework

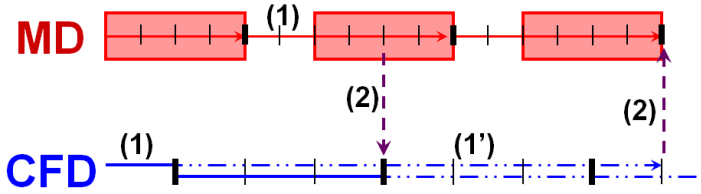
Two important issues regarding the performance of hybrid simulations are co-scheduling and load balancing. Both can be resolved by adopting the Pilot-job concept. We explain the design of a multi-physics simulation framework that operates in the form of a single Pilot-job and contains a load-balancing function between distinct tasks.



(a) Extrapolated Hybrid Boundary Conditions



(b) Interpolated MD Boundary Condition



(c) Interpolated Hybrid Boundary Conditions

Figure 3: **A prediction-correction approach with extrapolated / interpolated hybrid boundary conditions.** (a) While MD code evolves for one sampling interval from $(n-1)\Delta t + \frac{1}{2}\Delta t_s$, CFD solver runs the actual evolution from $(n-1)\Delta t$ to $n\Delta t$ and adds the the prediction step for $\frac{1}{2}\Delta t_s$. MD code gets the *predicted* continuum hybrid solution at $n\Delta t + \frac{1}{2}\Delta t_s$ while CFD solver receives the centrally-sampled MD data around $n\Delta t$. Both codes extrapolate received solutions for imposing hybrid boundary conditions for the next evolution loop. (b) The prediction step in CFD solver is increased to $\Delta t + \frac{1}{2}\Delta t_s$. MD code interpolates two *predicted* CFD solutions at $n\Delta t + \frac{1}{2}\Delta t_s$ and $(n+1)\Delta t + \frac{1}{2}\Delta t_s$ for the hybrid boundary conditions when the code evolves over the same interval. (c) Time axis of CFD domain shifts backward by 1 sampling interval and the prediction step is set $2\Delta t + 0.5\Delta t_s$. Hybrid boundary conditions at both solvers are acquired through the interpolation.

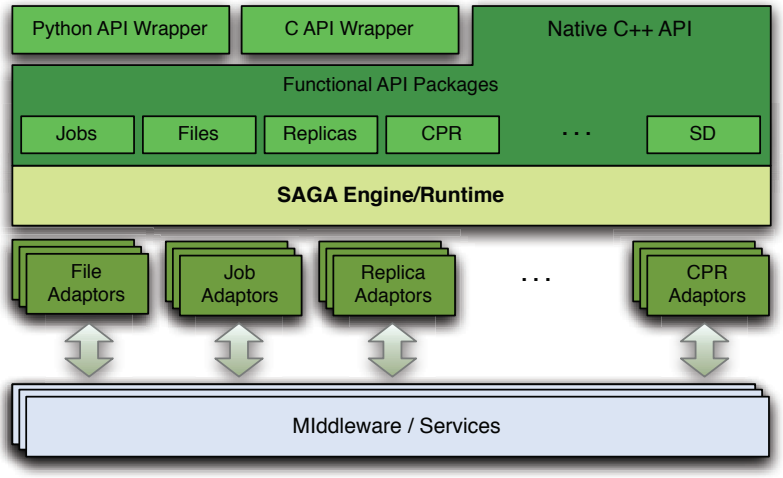


Figure 4: **Layered schematic of the different components of the SAGA landscape.** At the topmost level is the simple integrated API which provides the basic functionality for distributed computing. Our BigJob abstraction is built upon this SAGA layer using Python API wrapper

3.1. SAGA and SAGA-based Frameworks - An Efficient Runtime Environment for Coupled Multi-component Computations

The simple API for grid applications (SAGA) [26] is an API standardization effort within the open grid forum (OGF) [27], an international standards development body concerned primarily with standards for distributed computing. SAGA provides a simple, POSIX-style API to the most common grid functions at a sufficiently high-level of abstraction in order to be independent of diverse and dynamic grid environments. The SAGA specification defines interfaces for the most common grid-programming functions grouped as a set of functional packages (Fig. 4). Some key packages are the following:

- File package - provides methods for accessing local and remote file systems, browsing directories, moving, copying, and deleting files, setting access permissions, as well as zero-copy reading and writing
- Job package - provides methods for describing, submitting, monitoring, and controlling local and remote jobs. Many parts of this package were derived from the largely adopted DRMAA

- Stream package - provides methods for authenticated local and remote socket connections with hooks to support authorization and encryption schemes.

- Other Packages, such as the RPC (remote procedure call) and Replica package

BigJob [28] is a SAGA-based Pilot-Job application, by which a number of sub-tasks can run in a pre-defined schedule with the specified number of processors whether or not they are coupled. We devised this solution to overcome the concurrent scheduling requirement of coupled CFD and MD jobs and to allocate resources for the load balancing of these codes. The advantage of the BigJob application over other Pilot-Job implementations is that it is infrastructure-neutral, thanks to various adaptors in SAGA.

Fig. 5 shows the structure of BigJob and its operation flow. When a BigJob item is submitted to the remote resource, the application manager monitors the status of this Pilot-Job through the advert service. When resources are allocated to BigJob, the application manager allots the obtained resources to its sub-jobs, and a coupled simulation starts under the control of a multi-physics agent in the remote resource. The advert service is constantly notified about the status of a Pilot-Job from the queuing system and the status of sub-jobs from the multi-physics agent. It also delivers this information to the application manager by a push-pull mechanism. The application manager watches the status of the sub-jobs and, when the coupled simulation is finished, decides on the next event. If an individual simulation is of interest, the manager closes the BigJob allocation when the simulation is finished. In cases of multiple replica simulations or load-balanced coupled simulation, the manager re-launches the sub-jobs on the same BigJob allocation until all replicas or simulation loops are completed.

3.2. Load-Balancing of Coupled Multi-Physics Simulation

The load-balancing of a coupled simulation implies sufficient flexibility to re-distribute resources to an individual job according to each performance. We will discuss the implementation and algorithm of a simple load balancer (LB) [29]; it is important to mention that the LB functions in the context of the SAGA-BigJob framework.

The idea is to assign more resources to heavier sub-jobs under the fixed resource allocation, until all sub-jobs elapse at the same execution time. As

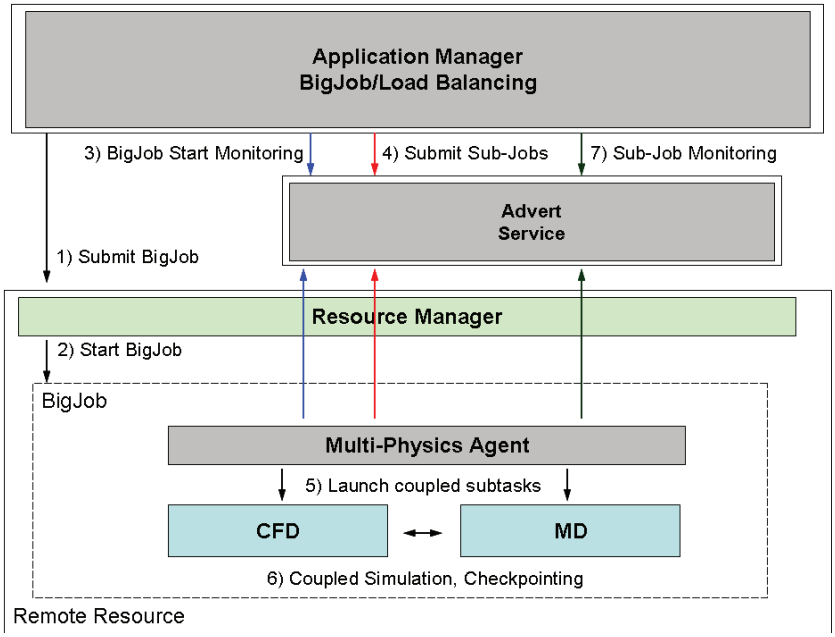


Figure 5: **Architecture of the controller/manager and control flow.** The application manager is responsible for job management including BigJob and sub-job submission, their status monitoring functions. We implement a load-balancing module, and migration service based on job information. The application agent system resides on each HPC resource and conducts job information gathering and also communicates with the application manager via the advert service

it is impossible to predict the performance of each code in advance, we let the LB monitor the wall-clock time between the information exchange points of the coupled sub-jobs and iteratively change the resource distribution until load-balancing is achieved. As the individual solver is considered as black-box, each application code is assumed to have the ideal parallel efficiency. In case the application codes are highly scalable, the LB can find the best condition after a few dynamic re-distributions. In addition, all processors in one node are assigned to one single task to prevent the interference (and performance degradation) observed when multiple MPI tasks share the node.

Let the computation time (between exchanges) of the two sub-jobs are t_{CFD} and t_{MD}, and the number of cores assigned to each domain being PE_{CFD} and PE_{MD}, respectively. Subscripts C and N denote current and next states. Assuming ideal parallel efficiency, the total load of each application remains the same after resource re-allocation,

$$\begin{aligned} W_{CFD} &= PE_{CFD,C} \times t_{CFD,C} = PE_{CFD,N} \times t_{CFD,N} \\ W_{MD} &= PE_{MD,C} \times t_{MD,C} = PE_{MD,N} \times t_{MD,N} \end{aligned} \tag{10}$$

In spite of the re-allocation, the total number of cores utilized remains the same:

$$PE_{TOT} = PE_{CFD,C} + PE_{MD,C} = PE_{CFD,N} + PE_{MD,N} \tag{11}$$

Our objective is to reduce the computation time of a sub-job to the point where the two application components show the same computation between the exchange points, i.e., $t_{CFD,N} = t_{MD,N}$. Derived from Eqn. 10 and Eqn. 11 the optimal number of cores distributed for the CFD sub-job would be:

$$PE_{CFD,F} = \frac{W_{CFD}}{(W_{CFD} + W_{MD})} \times PE_{TOT} \tag{12}$$

The MD simulation (sub-job) will follow a similar expression.

The above non-integer values proceed in discrete values expressed as the multiples of the number of CPU cores in a node. We chose the nearest discrete number to our load-balancing as the optimal number of processors in each application.

3.3. Implementation of an Execution Framework and Application-level Corrections

A hybrid CFD-MD framework is evaluated by implementing the application manager shown in Fig. 5, which is written in PYTHON script language.

By default, an application manager calls a number of SAGA functions in sequence, to allocate a vacant job, to run individual MPI simulations, to monitor its status, and to finalize the BigJob allocation.

In the case that the load-balancing capability is turned on, the situation becomes complicated. A single MPI job is not able to change its number of processors during the simulation, which implies that coupled codes should stop-and-restart to be assigned with the changed number of processors. Thus, sub-jobs are scheduled to have multiple restarts from the previous check-pointing solution, which we denote "simulation loop". An LB is provided as a separate function in an application manager and is scheduled to run in between each restart of the sub-jobs.

The efficient functioning of the LB is predicated on application codes being able to restart effectively from their check-pointing data. Application codes should also be equipped with a generalized domain partitioning routine to run a simulation with any number of processors without harming their parallel efficiency. Another change implemented in the application codes is the time checking routine. The runtime of each application is meaningless in running a LB since this runtime contains idle waiting on the inter-domain information exchange as well as the individual simulation time. The actual runtime can be counted by putting a wall-time function before and after the information exchange routine.

The generation of an application manager and the changes in application codes raise the possible simulation scenarios shown in Fig. 6. The first (extreme left) scenario shows the time evolution of a coupled simulation under a conventional job submission (which we defined as scenario S0), and others using a BigJob application (denoted as S1). For S0, individual tasks with resource requirements of PE_{CFD} and PE_{MD}, respectively, are independently submitted to the conventional queuing system, and the job scheduler recognizes these coupled tasks as two distinct jobs. Thus, on average, they start at different times. In this case, both tasks wait in the queue when no job is allocated (waiting stage). The first allocated job idles to perform data exchange with its counterpart (idling stage), and the actual simulation starts when both jobs are allocated (running stage). On the other hand, for scenario S1, a BigJob of size $PE_{CFD} + PE_{MD}$ is submitted to the queue, and coupled simulation directly starts when the resource is assigned to this BigJob. Because of the co-scheduling of sub-jobs, a BigJob is free from a long inactive mode, which is frequent in conventional job submission, while the total runtime is the same if the resource distribution to the sub-jobs is

identical. However, eliminating the inactive mode in itself does not guarantee a reduction in the total runtime because a larger single allocation may result in a greater queue waiting time than that when two simulations request a smaller number of processors each (but with the same total). The same situation can arise for the load-balanced case with one BigJob ($S1_{LB}$). From the comparison between S1 and S0, we can estimate the performance gain by the concurrent start of distinct coupled codes. Compared with other scenarios, the $S1_{LB}$ solution demonstrates the benefit of a load-balancing function on coupled simulations.

4. Multi-physics Flow Simulations in Various Flow Conditions

A hybrid CFD-MD framework is employed to solve the nano-scale multi-physics flow field. We solve the transient Couette flow and the oscillating boundary problem, to evaluate the capability of solving moderate-speed flow field and to verify a "prediction-correction" temporal coupling scheme.

4.1. Problem Description and Coupling Conditions

Both the Couette flow and oscillating boundary problems solve the same geometric configuration of a nano-scale channel which is filled with the liquid argon. The characteristic length of the liquid argon is $\sigma = 3.405\times10^{-10}$ and the time scale is $\tau = 2.2\times10^{-12}$. The density is $0.81m\sigma^{-3}$, which implies that 0.81 atoms are placed in the characteristic volume. Wall materials are artificially designed to have the same characteristics as the liquid argon particle. The slip ratio between fluid and solid particles is set at 0.6 to satisfy the linear velocity gradient along a vertical direction. [2] The channel is 52 σ in height, which is $O(10^{-8})$ meters. Applications covered in this work are periodic systems in a perpendicular direction, and the flow is derived by the horizontal motion of the upper plate.

The computational domain for the hybrid simulation is depicted in Fig. 7. The pure MD region is specified as 10 σ, which was reported to be sufficient to prevent strong fluctuation between fluid particles and wall materials from direct transportation to CFD domains. [6] The hybrid region is designed to be 16 σ to guarantee that the molecular simulation domain does not exceed the half of the fluid system. The hybrid region is divided to 8 layers. Two layer boundary zones from the particle to the continuum domain are placed ahead of the pure MD region, from 10 to 14 σ. Two layers of the continuum domain to the particle boundary are positioned from 18 to 22

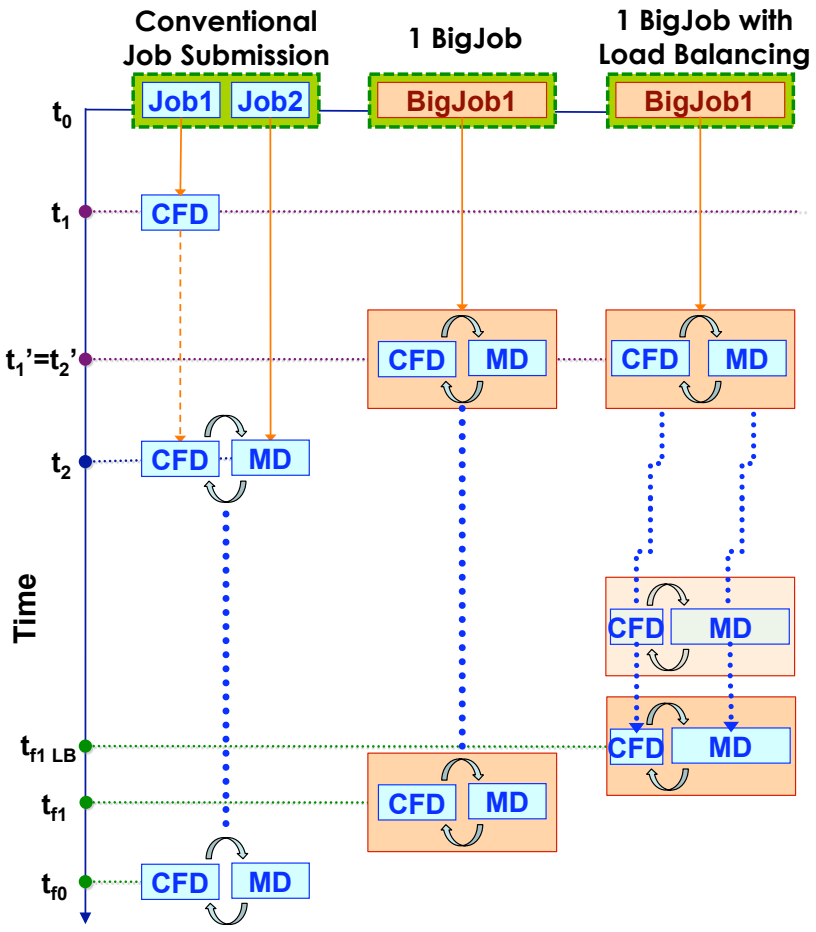


Figure 6: **Comparison of dependencies encountered for coupled simulations submitted as conventional jobs, to the scenario when using Pilot-Jobs.** Here we use only 1 BigJob (S1). The conventional mode of submission experiences three phases based upon their queue state: (i) All jobs are waiting: $(t_1 - t_0)$; (ii) Inactive mode (where one job is waiting for another: $t_2 - t_1$), and (iii) Active mode (running a coupled simulation: $t_f - t_2$). The Inactive stage, disappears when a coupled simulation runs within a BigJob, as an allocated BigJob distributes its resource to both sub-jobs.

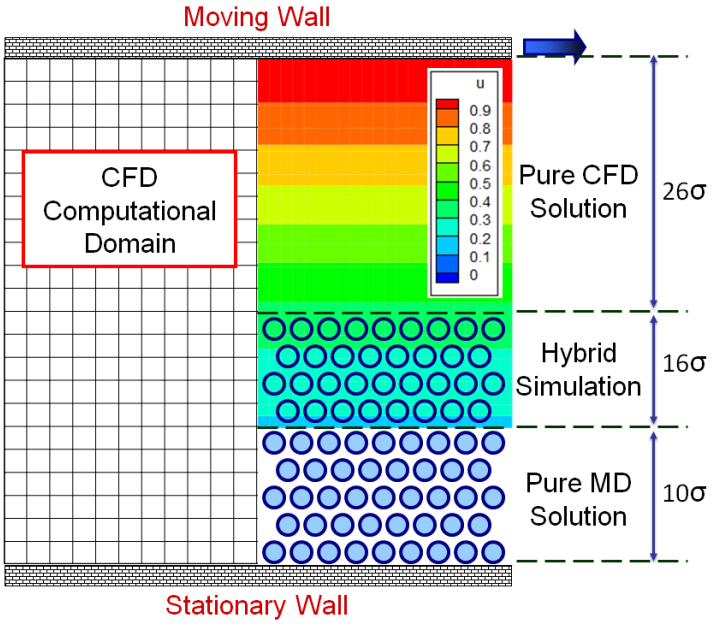


Figure 7: **Computational domain of the Couette flow simulation.** The height of the fluid domain is 52σ ($\approx 177\mathring{A}$). CFD mesh size is 71×27 and CFD cells at the pure MD region are treated as holes. MD domain size is about 140σ in the X-direction and around 26σ in the Y-direction, including the bottom wall. Periodic boundary condition is applied on the principal flow direction.

σ and the external force region is placed at the top of hybrid region, from 24 to 26 σ. Both the sampling interval and the sampling duration are set to be 10 τ, considering the characteristics of our deterministic application targets. Finally, the width of the MD domain along the periodic direction is determined at 140 σ, after a number of numerical experiments.

4.2. A Transient Couette Flow

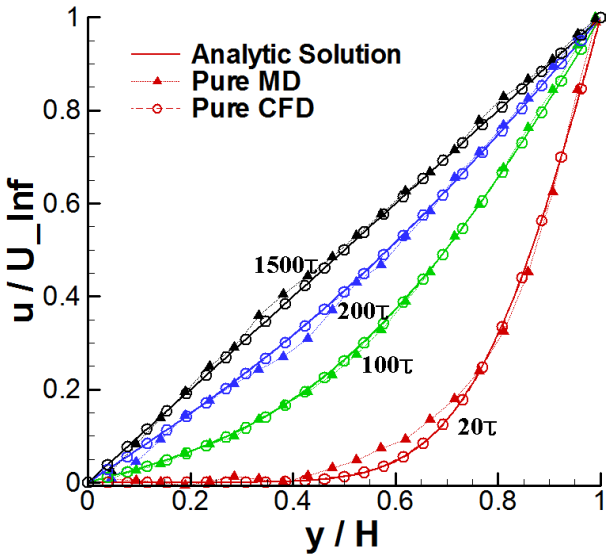
A transient Couette flow is simulated to verify the accuracy of the current framework with a multiple replica sampling approach for the moderate-speed flow simulation. This application has been widely used for the validation of a hybrid CFD-MD solver [2, 6]. The flow is initially set at stationary and the upper wall starts the horizontal motion with a constant velocity ($1\ \sigma/\tau$). Figure 8 presents transient Couette flow profiles by CFD, MD, and hybrid simulations. Pure CFD produces the identical result as the analytic solution: Pure MD also describes the same flow profile with the slight sampling noise.

These results verify the accuracy of the individual solver. The hybrid simulation also provides the same flow profile, again with the slight deviation due to the fluctuation in the sampled MD property. It proves that the current hybrid framework accurately analyzes the steady flow field in nano-scaled systems.

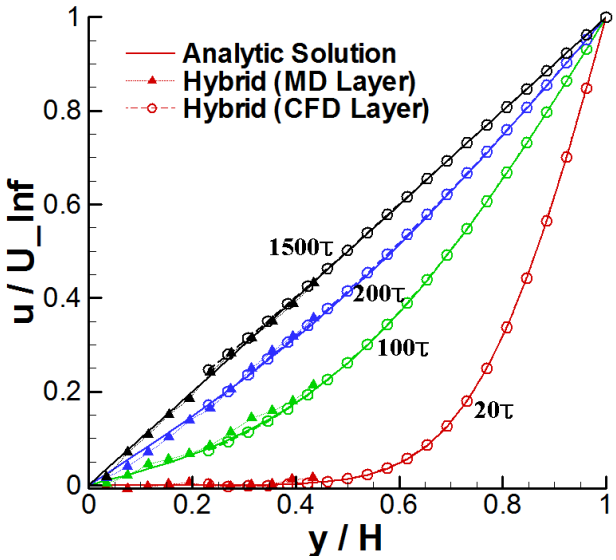
However, the flow condition in the above experiment is rather unrealistic, considering the nanometer-level distance between two parallel walls. Thus, we challenge ourselves to solve the moderate-velocity flow field by the hybrid simulation approach. As discussed in Sec. 2.4, the hybrid simulation of the low-speed flow field is mathematically possible but technically bound by the computing capacity, since the computational domain becomes u^2 times bigger in solving for a $1/u$ velocity field. Instead, we ran u^2 independent simulations with the initial system size and different random number seeds in the LAMMPS package.

Fig. 9 presents the Couette flow profiles at different numbers of samples. All configurations and parameters are identical to the above validation problem except the upper plate velocity of 0.25 σ/τ. Changing the velocity to 1/4 implies that an average of 16 samples is required to achieve an acceptable numerical solution. Our result supports the above supposition. The reduction of statistical noise by the multiple replica sampling is clearly verified in the graph presented in Fig. 10. The noise is seen to reduce by roughly half with 4 times more samples. The solution of sampling 16 runs shows about 5% of the noise compared with the analytic velocity solution.

We compared the solution by multiple replica sampling with the solution in the increased system domain. Figure 11 shows the Couette flow profile on a 16 times larger system size in the horizontal direction and the temporal velocity variation by the multiple replica simulation and large domain solution in the middle of the overlapping region. From the result, both ways (multiple replica sampling and increasing system size) produce accurate numerical solutions compared with the analytic solution. The scale of sampling noises are roughly the same between two simulations according to Fig. 11, which verifies that the multiple replica sampling apporach is as effective as the increment of the simulation domain. In view of total time-to-complete, multiple replica sampling can be more efficient, because a massively large-scaled job (in view of wall-time limit or number of resources requested) is very hard get allocated. The labor of multiple job submissions and the postprocessing of replicas can be relieved by the use of a BigJob abstraction.

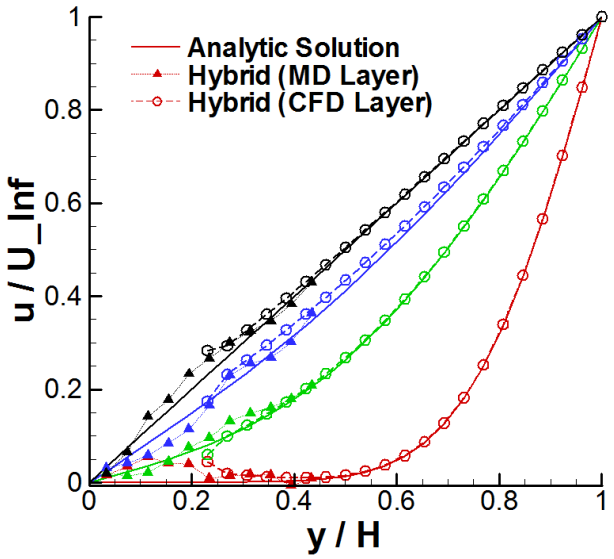


(a) Pure CFD and MD Results

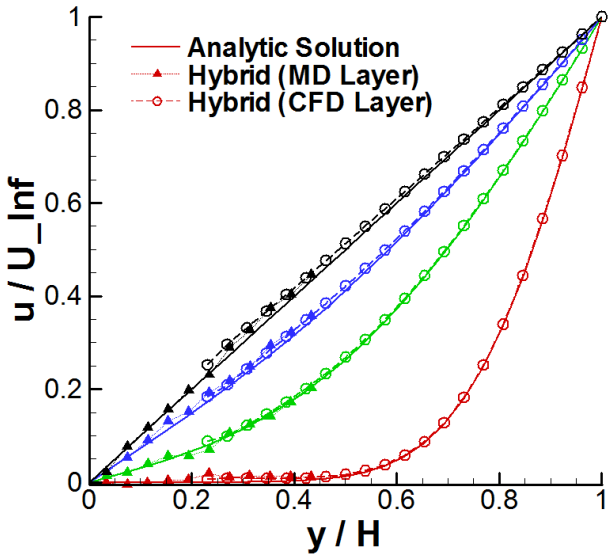


(b) Hybrid CFD-MD Solution

Figure 8: **A time-accurate Couette flow profile.** The evolution of velocity field along the vertical direction is presented. CFD solution is the instantaneous profile at specified time and MD solution is spatially averaged over 2 σ in height and temporally averaged for 1 sampling durations (=10τ). (a) Pure CFD solution is exactly the same as analytic solution. MD solution shows the same flow pattern as analytic solution, though some fluctuation is observed. This verifies that CFD and MD represents the same flow physics. (b) The steady result by a hybrid approach produces the same numerical result as analytic solution, though the slight time-lagging in the hybrid boundary is observed during the evolution.



(a) Sampling of 4 Replicas



(b) Sampling of 16 Replicas

Figure 9: **A Couette flow profile with the upper plate velocity of 0.25** σ/τ**.** The noisy solution when 4 individual simulations are averaged is resolved by sampling 16 independent runs. Red lines denote the solution at 20 τ; Green, blue and black lines are solutions at 100, 200 and 1500 τ, respectively.

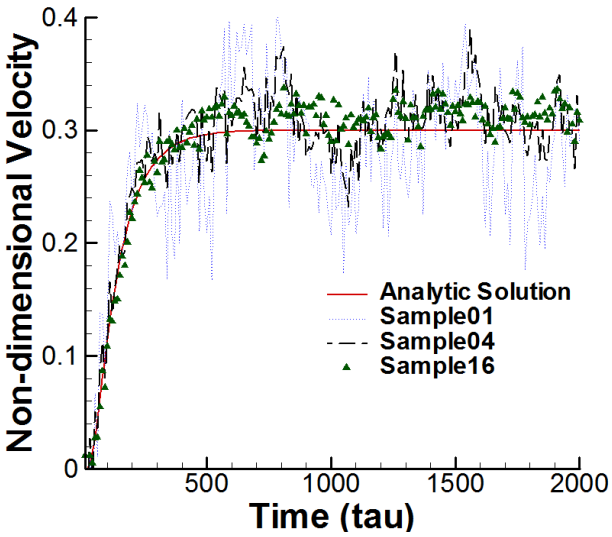


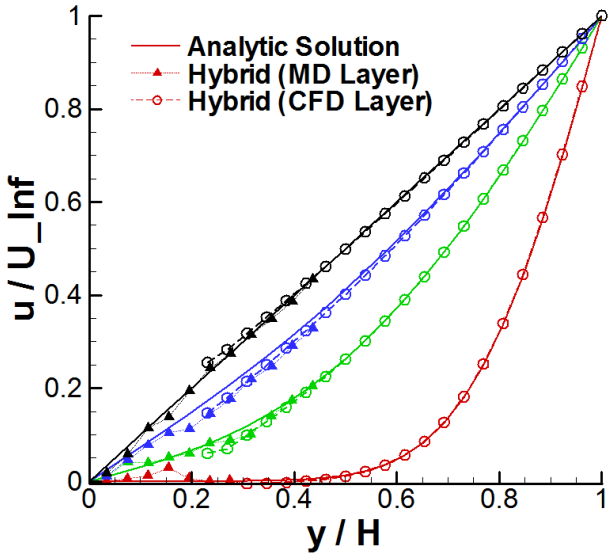
Figure 10: **The variation of continuum velocity in the middle of the overlapping region.** Velocity changes at various replica samples are compared with the analytic solution. In case 16 simulations are averaged, the noise is about 5 % compared to the analytic solution.

4.3. A Physically Unsteady Flow Field: Oscillating Boundary Problem

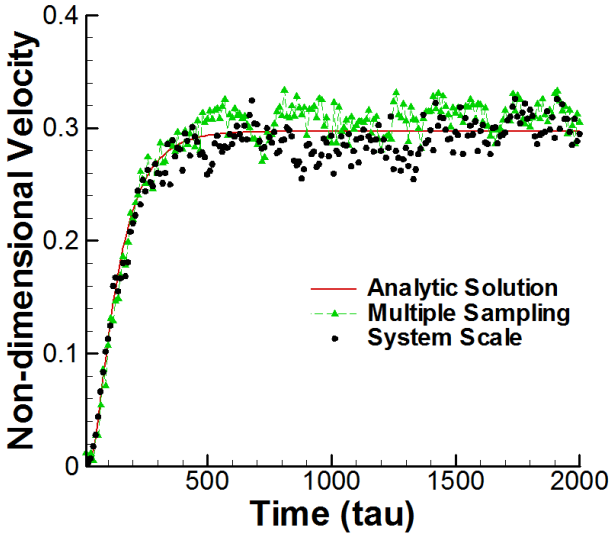
The accuracy of the designed temporal coupling scheme is verified by solving the unsteady oscillating boundary problem.

The flow field converges to a specific profile in a transient flow physics, so the minor disturbance in the middle of flow evolution can possibly be damped out at time of convergence. On the other hand, the local variation at each instance might be incremented/accumulated during the flow evolution, to end up with the completely un-physical solution in case of unsteady flow simulation. Therefore, the accuracy of the temporal coupling scheme becomes more important in the unsteady flow simulation. Suppression of the sampling noise is very important as well, considering the low physical velocity in the hybrid layer. The computational domain and coupling parameters of the current problem are the same as in the above Couette flow simulation. In this case, the upper wall boundary condition changes from the fixed velocity to the oscillatory wall, which is $u_{wall}(t) = (\sigma/\tau)\times sin(2\pi t/T)$. Period T is set to be 200τ.

Figure 12 shows the oscillatory velocity profile by pure CFD and hybrid simulations. Figure 12-(a) shows that the velocity profiles at each time in-



(a) A Single Run at 16 Times Larger Domain



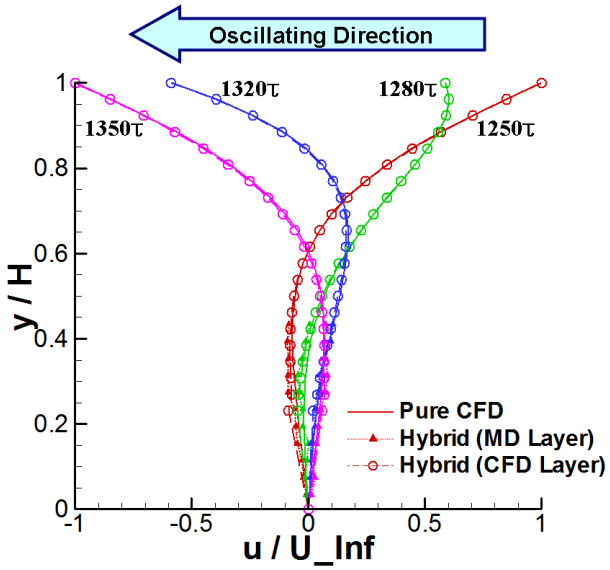
(b) Temporal Variation of Velocity between Replica Sampling and System Increase

Figure 11: (a) **A Couette flow profile in increased system size.** Accurate flow profile can be obtained by increasing the system size 16 times larger when the velocity is reduced to 1/4. Red lines denote the solution at 20 τ; Green, blue and black lines are solutions at 100, 200 and 1500 τ, respectively. (a) **The variation of the velocity in the middle of the overlapping region.** Solutions by multiple replica sampling and increasing the system contain the similar strength of the noise in the solution, which is around 5 % of analytic velocity profile. This implies that multiple replica sampling approach can replace the increase of system size for solving moderate velocity flow.

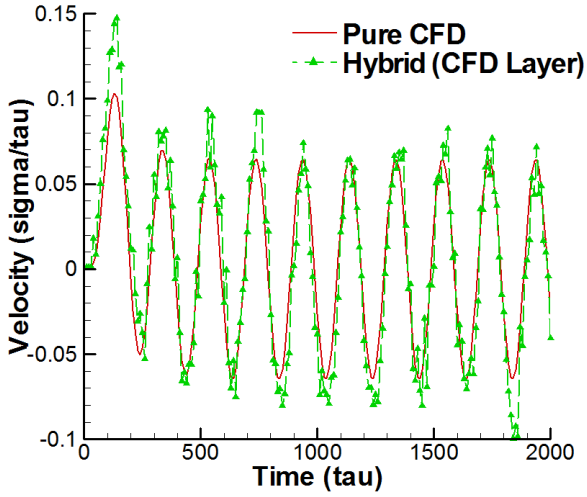
stance are roughly the same between the pure CFD and hybrid simulations in magnitude. This indicates that the hybrid simulation approach is also applicable to time-varying flow simulations. Meanwhile, the temporal variation of the horizontal velocity in the middle of the hybrid region expresses that the noise in the hybrid solution is not negligible considering the difference between continuum and hybrid velocities. We claim that this noisy solution is caused by the combination of the sampling error and an inaccurate temporal coupling scheme. In order to determine the effect of the temporal coupling scheme, we applied multiple replica sampling over sufficient number of independent experiments and compared solutions from different temporal coupling schemes.

We compared velocity history profiles from different temporal coupling schemes. Figure 13 indicate a clear difference in the resolution of the overshoot/undershoot phenomena by the prediction-correction approach. A prediction time scale of 2.5 sampling intervals was chosen in the prediction-correction approach to impose the interpolated boundary conditions on both domains. In the conventional model, the maximal error is seen at peak points, which is a natural characteristic of the linear extrapolation in that it fails to predict correct vlaues on a convex/concave graph. This inaccuracy is resolved by a prediction-correction approach with interpolated hybrid boundary conditions. Yet, the time-lagging pattern in the conventional model is still observed in the prediction-correction model, which is expected to be resolved by the adoption of a higher-order interpolation scheme.

The plots shown in Fig. 14 quantify the scale of inaccuracy according to the imposition of the hybrid boundary condition. In the conventional model, the velocity difference ranges from -0.017 to 0.015 σ/τ. The magnitude of this error reduces as we apply the prediction-correction approach and increase the prediction time scale from 0.5 (corrected extrapolation) to 1.5 (MD interpolation) and 2.5 (both interpolation) sampling intervals. The clear difference between the conventional extrapolation model and corrected extrapolation in the prediction-correction approach demonstrates that the extrapolation from "previous" properties can contain far more inaccuracy than extrapolating from the current solution. The similar noise level between corrected extrapolation and MD interpolation express that the accuracy of the hybrid solution is equally affected by the accuracy of each solver. At last, the solution error in the conventional model is roughly reduced by half with the imposition of interpolated boundary conditions on both domains.

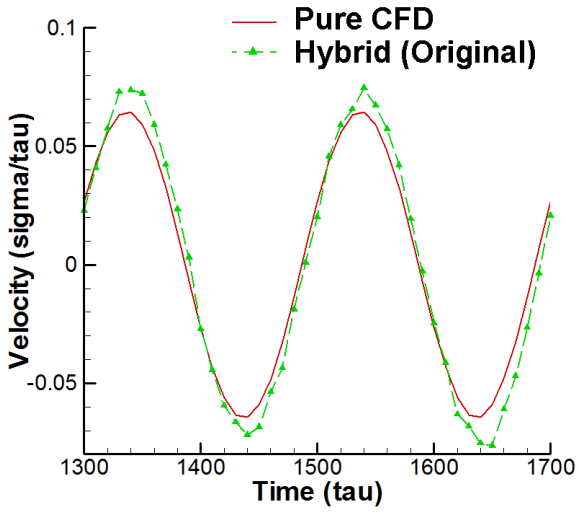


(a) Velocity Profile of an Oscillating Boundary Problem

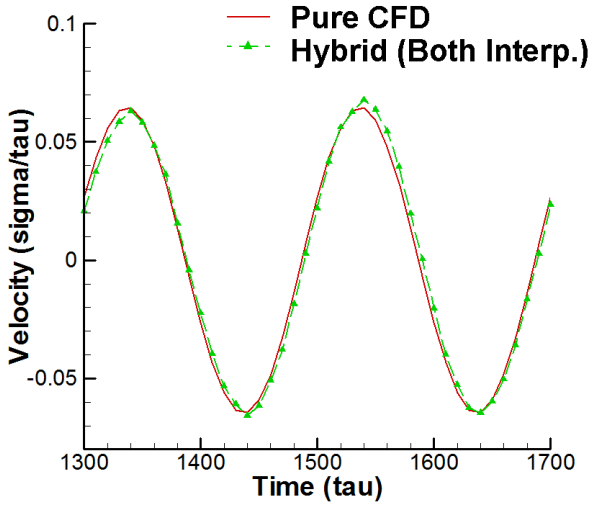


(b) Temporal Variation of Velocity in the Middle of Hybrid Region

Figure 12: **A time-accurate flow profile of an oscillating boundary problem.** (a) Difference between hybrid and pure CFD solutions in hybrid region shrinks in the continuum domain. Noise from the MD simulation does not affect the global velocity profile a lot, because the driving force in this simulation is the oscillating velocity from CFD domain. (b) History of the velocity field reveals that the hybrid solution contains much noise in this experiment.

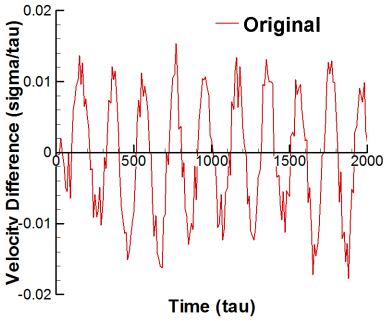


(a) Velocity History at Conventional Synchronized Approach

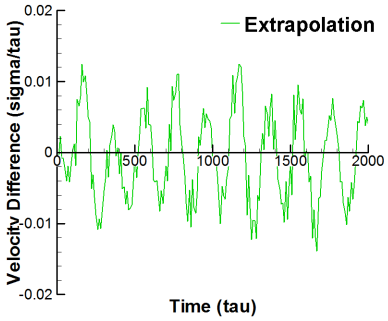


(b) Velocity History at Prediction-Correction Approach

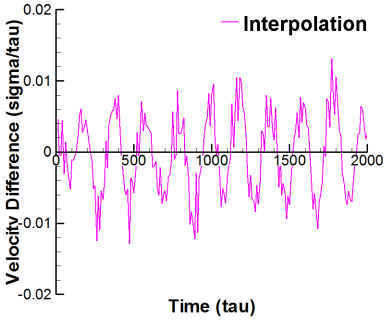
Figure 13: **The temporal variation of the velocity in the middle of an overlapping region.** (a) With conventional synchronized coupling model, large overshoot/undershoot phenomena are observed on the peak points: the statistical noise around 0.01 σ/τ is nearly 20 % of the continuum velocity in that location. The time-lagging effect is also captured. (b) Overshoot/undershoot phenomena in the conventional approach is almost resolved by applying the prediction-correction approach. The interpolated hybrid boundary conditions are imposed on CFD and MD domains.



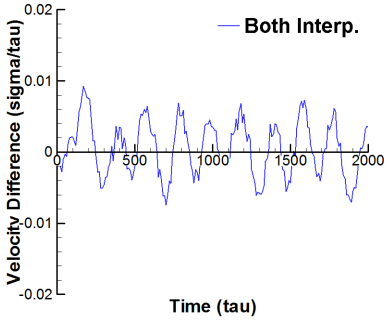
(a) Conventional Synchronized Coupling with Extrapolation



(b) Prediction-Correction with Extrapolation



(c) Prediction-Correction with Interpolation on MD Boundary



(d) Prediction-Correction with Interpolation on Both Domains

Figure 14: **The reduction of unsteady error by a prediction-correction approach.** Figures show the velocity difference between pure CFD and hybrid solutions in the middle of hybrid layer. Plotted data include the conventional extrapolation model and 3 different types of prediction-correction approaches, which are corrected extrapolation model, interpolated boundary model on MD domain and interpolated boundary condition on both domains. The velocity difference in the conventional model ranges from -0.017 to 0.015. This error reduces to -0.007 and 0.009 at the prediction-correction model.

5. Performance Analysis of a Multi-physics Simulation Framework

The BigJob framework is expected to have a shorter total runtime in a coupled multi-task simulation than conventional (and direct) job submissions. The logical bases are as follows:

1. Many supercomputing centers adopt a queueing policy of assigning higher priority to bigger jobs, which increases the probability that bigger-sized task are allocated faster than the smaller one(s).
2. Independlently submitted multiple tasks are usually allocated at different physical time. Therefore, the requested wall-time limit should be sufficiently large to cover both maximal waiting time and computation time in case of coupled simulations. On the other hand, co-scheduling of coupled tasks is inherently guaranteed in a packaged job. Therefore the wall-time limit can be determined according to the actual computation time.
3. Load-balancing among multiple tasks can be dynamically achieved dynamically by changing the resource allocation to individual task during the simulation. This change is be possible only when coupled yet distributed tasks are scheduled in the umbrella of a packaged job.

In this section, we present our numerical experiments to verify the above reasons.

5.1. Waiting Time according to the Size and Wall-time Limit of an Individual Job

Although many factors affect the waiting times, arguably the most important are existing job requests by the resource at the instant of submission, requested wall-time limits, and the number of processors requested. Two other factors are the backfilling capability (which allows the running of a small job between higher priority jobs with larger and longer resource requests) and the changes in the priority of the test job when a particular higher priority job joins the queue. Among these factors, the number of requested resources and wall-time limits can be handled by users and systematically accounted for.

We designed experiments to determine if running larger and/or longer simulations affects the actual waiting time in the queue. We submitted jobs of different sizes and different wall-time limits at the same time. Each time we submitted a job, we gathered the actual waiting time in the queue. We

performed our experiments on a sufficiently large and crowded system of around 65000 cores. We argue that the use of this large and crowded system increases the credibility of our experiments: it reduces the possibility of self-interference between our job submissions (i.e., our requests are allocated in a succeeding pattern). Considering the internal queuing policy of supercomputing centers, such as credential, fair share, resource and service priority, each user account has a restricted number of concurrent job submission, and the allocation history affects the priority of the next submitted job. Thus, we submitted different wall-time limit jobs from different user accounts in the same group allocation, and we allowed enough idling time between individual experiments to recover the priority of each account to the initial level. Results from 10 independent experiments were averaged for measurement.

Table 2 demonstrates the queue waiting time according to the size and wall-time limit of the job. Regarding the influence of job sizes on waiting time, jobs with larger core counts have typically lower wait times, with the exception at 128 cores which is presumably affected by the backfilling capability. Waiting time with 1000+ cores shows a stiff decrease particularly because it is granted higher priority in this queuing policy. The same applies to the increase in waiting time to the wall-time limit of 48, which is administrated by a different queue. The effect of the wall-time limit on waiting time is not clear in our experiment, in the case where they are in the same queue (6- and 24-hour jobs). Collectively, submitting a BigJob for the coupled simulation instead of individually submitting multiple smaller tasks *at least* provides the comparable waiting times in the queue even when small jobs are "ideally" allocated (i.e., all jobs are allocated at the same time).

5.2. Waiting Time for a Coupled Simulation

Regarding the scenario maps in Fig. 5, the result in Sec. **??** provides the waiting time between a BigJob and a first-allocated conventional task. We expect more performance gain by eliminating the inactive idling time (i.e., the difference between the waiting times of the two jobs) with the BigJob application.

Table3 presents the waiting time of a BigJob sized 2X and two conventional job submissions sized X each. We experimented with two different wall-time limits (6 and 24 hours) and two different processor requests (256 and 512 cores in total). A BigJob submission shows faster allocation, except a small number of processors are requested for a long time, although the waiting time for the first-to-start job was smaller in the conventional job

Table 2: **The effect of job conditions on waiting times.** The tables show the queue waiting times depending on the number of cores and requested wall-time limits. Measurements are made on Ranger system, which is a TeraGrid resource located at Texas Advanced Computing Center. Analyzing the actual waiting time as a function of the number of cores at different wall-time limits, it can be said that better more often than not, the waiting time decrease as the requested number of cores increases. The relationship between the waiting time and wall-time limit is harder to quantify. However, obtained numbers provide a good case study for showing the variance of actual queue waiting times. 10 independent experiments are sampled and expressed in seconds as "mean±SD".

Number of Processors	Requested Wall-time on Ranger		
	6 Hrs	24 Hrs	48 Hrs
128	24287.14±15724.91	25941.71±13878.78	44012.14±44537.75
256	27606.57±19207.95	26878.86±15108.29	67876.29±70713.95
512	27930.43±18496.98	24361.43±14849.23	32097.14±27617.86
1024	13514.57±11373.68	16372.43±10857.48	22134.71±11089.26

submission mode (S0) than in the BigJob (S1) application. Interestingly, the inactive mode in conventional job submissions was observed to increase substantially as the wall-time limit increased: users get to waste more allocation time as they increase the wall-time limit. From the result, bigger and longer jobs tend to start faster (comparing individual experiments), but it is difficult to affirm this tendency as a general phenomenon because individual experiments were performed at different times.

5.3. Performance Gain through Load Balancing

A BigJob framework in itself does not provide any performance gain during the simulation: a load balancer (LB) in a BigJob framework provides better parallel performance by redistributing processors to individual tasks within the context of the packaged job. LB measures the performance of individual tasks during a temporary stop and evolves application codes with a changed number of processors. Thus, each code is scheduled to stop and restart several times for the complete simulation, necessitating the capability of a application-level check-pointing and generic domain partitioning routine in application codes (both of which are incorporated capacities in most legacy codes).

Table 3: **Waiting and inactive time for conventional job submissions, and a single BigJob submission.** Average of 10 independent runs are presented in seconds. In the first two cases, conventional job is submitted to use 2×128 cores and a BigJob requests 256 cores: latter two cases use 2×256 and 512 cores, respectively. Conventional job submission mode showed faster time-to-start (i.e., waiting time of the first job + inactive mode) with 24 hr - 256 core request, and a BigJob is allocated faster in all other cases.

Number of Cores	Wall-time	Scenario	Time-to-start
256	6 Hrs	S0	20237.9 + 1045.3
		S1	19720.3
	24 Hrs	S0	5035.8 + 9933.3
		S1	16165.0
512	6 Hrs	S0	15515.7 + 446.1
		S1	15825.4
	24 Hrs	S0	4182.6 + 11096.3
		S1	13627.9

The benefit of a LB in a BigJob framework is that it can be applied to any type of coupled applications. On the other hand, it is not highly recommended for the current application for two reasons: 1) in a hybrid CFD-MD simulation, MD usually requires far more computing power than most resources that are dedicated to molecular dynamic simulation; 2) computational cost for the current application in Sec. 4 is so small that it is not necessary to apply the load-balancing capability. These reasons led us to increase the computational cost of each simulation for evaluating the LB in a BigJob framework. In this experiment, the CFD domain size is increased by 500 times and the MD domain is increased by 20 times. Both CFD and MD codes are scheduled to have 10 simulation loops: start with the same number of processors and experience nine stop-and-restarts with changed resource allocation according to the result of the LB.

The runtimes of the coupled simulation with a single BigJob are shown in Table 4. In all scenarios, the same numbers of processors were initially assigned to CFD and MD simulations. Processors distribution in $S1_{LB}$ changed during the simulation according to the result of the LB function. For both simulations, the time difference between S0 and S1 is within 1%, which explains why the possible overhead of a BigJob due to communication with the

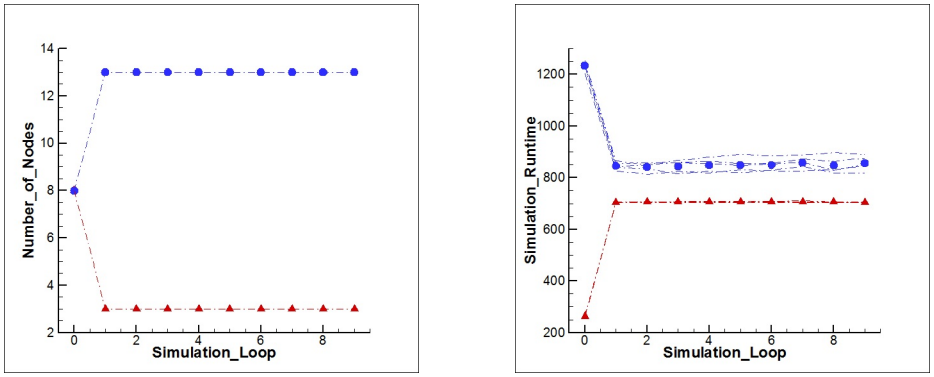
Table 4: **Simulation runtimes for S0 (conventional job submissions), S1 (default BigJob) and $S1_{LB}$ (a BigJob with load-balancing).** 256 and 512 cores are used for the coupled simulation. For both cases, S0 and S1 show nearly identical computational cost because the same resource distribution is applied for the coupled simulation. With the load-balancing capability enabled, $S1_{LB}$ shows about 23.5% and 6.1% runtime save compared to S0 when 256 and 512 cores are used. All measured times are in seconds and 5 distinct experiments are averaged.

Number of Cores	Scenario		
	S0	S1	$S1_{LB}$
256	12306.0	12248.7	9413.7
512	7834.2	7910.6	7357.6

advert server (for monitoring the status of sub-jobs) is negligible. In cases of load-balanced BigJob simulations, there is a significant reduction in the runtime (23.1%) compared with the default BigJob application when 256 processors are used. For a larger problem set, the performance gain through the application of a LB is relatively small (7.0%), the reason for which is discussed below.

The validity of a LB can be discussed in terms of the change of processor distribution between sub-jobs throughout the simulation. For the result of the 256 cores (=16 nodes) simulation in Fig. 15, both CFD and MD sub-tasks were assigned with 8 nodes initially. From the next simulation loops, the processor distribution converged to 3 to 13 nodes between CFD and MD, respectively. This ratio was maintained throughout the simulation. CFD and MD computation times changed from 260 to 1235 seconds at the initial loop to 705 to 850 seconds after load balancing. Simulation runtimes were measured to be about 35 seconds longer than the slower simulation (MD simulation) per each simulation loop, which is the overhead by the initialization , I/O (input/output) and communication between coupled applications.

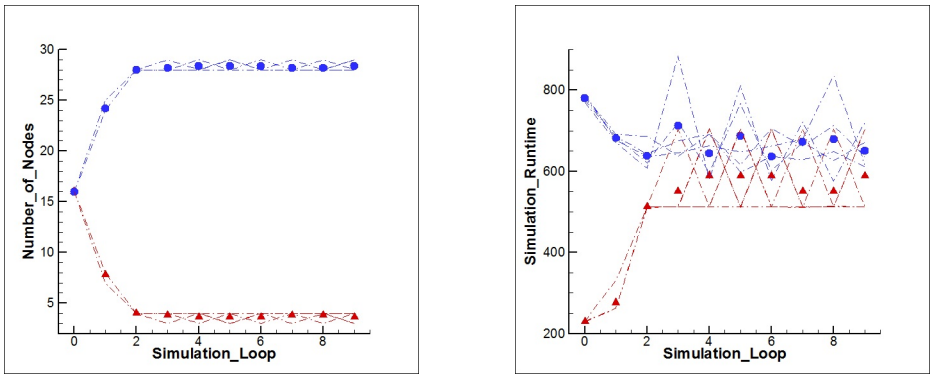
The result for the case of computation time evolution in 512 cores (=32 nodes) is presented in Fig. 16. Compared with the above experiment, which uses smaller cores, the load-balancing solution showed a noisier pattern. In detail, an LB failed to find the optimal solution at the first simulation loop, and the node allocation after load balancing fluctuated between 3 – 29 and 4 – 28 nodes in each application, which was caused by two reasons. First, the individual code has poor scalability. The initial simulation time for CFD



(a) Resource Allocation during the Load Balancing Loops

(b) Simulation Time during the Load Balancing Loops

Figure 15: **Change of resource distribution and computation time between CFD and MD sub-jobs using 256 cores (=16 nodes).** A LB detects the distribution of 3 – 13 nodes to CFD and MD sub-jobs as the optimal solution. In view of simulation runtime, 1270 seconds for the initial simulation loop reduces to 885 seconds after load-balancing is achieved. Note that each node (which contains 16 cores in the current system) is dedicated to a single application. Triangle and circle symbols denote averaged values from CFD and MD sub-jobs. Dashed lines are solutions from the individual experiment.



(a) Resource Allocation during the Load Balancing Loops

(b) Simulation Time during the Load Balancing Loops

Figure 16: **Change of resource distribution between CFD and MD sub-jobs and resultant computation time using 512 cores (=32 nodes).** A LB solution fluctuates between 3 – 29 and 4 – 28 nodes assigned to CFD and MD sub-jobs, because of temporary internal overhead of a system. Initial simulation runtime around 815 seconds is reduced to be 725 seconds after the load-balancing is achieved, because of the poor scalability of application codes for this problem set. The same caption is used as in Fig. 15.

and MD applications measured around 230 and 780 seconds, and the LB proposed a node distribution from 16 – 16 to 8 – 24. However, the computation time at the next simulation loop was measured to be 275 and 680 seconds, respectively. Therefore, the LB has to search for the optimal solution one more time. Load balancing functions that are incorporated in schedulers cannot avoid this iterative searching, since they have to manage black-box applications without getting any information to estimate the problem size. Another reason for the fluctuation is the momentary overhead in the computing system. In our experiment, when 28 nodes are used, MD simulation times vary from 610 seconds to 880 seconds, depending on the magnitude of the internal overhead. This variation indicates the necessity of an LB as a self-correcting tool in response to the instability of the system.

Two things remained for open discussion: 1) how to effectively measure the actual simulation time of individual task, and 2) how to determine the number of simulation loops. We put time checking routines between the inter-domain communication routine and accumulated the time for running the iteration loop of each code. Based on our knowledge, there is no way to systemically gather the individual simulation times except by this manual

deployment. Regarding the number of simulation loops, an excessive number of loops increases the I/O-related overhead (storing the check-pointing solution and restarting from that). An insufficient number of loops increases the simulation time at the initial imbalanced configuration and degrades the capability of the LB for adapting to the unpredicted internal overhead.

6. Next Step: Further Investigations

Numerical simulations in Sec. 4 demonstrates that the determination of coupling parameters along with the supplementary use of multiple replica sampling is very important for the accurate hybrid simulations. Unfortunately, coupling conditions have been rather intuitively determined according to the flow physics of the target problem, and existing mathematical expressions in statistical error [10, 17] unfortunately does not consider the variation of sampling noise depending on the geometric configuration. Sampling noise expression should be refined to investigate the interaction with solid obstacles.

Unsteady flow simulation in Sec. 4.3 verifies that the prediction-correction approach provides more accurate solution than conventional temporal coupling scheme. The new approach is especially powerful in resolving the unfavorable overshoot/undershoot phenomena. On the other hand, the slight time-lagging effectis still existent in the prediction-correction approach. We expect that this phenomenon can be resolved by applying a higher-order extrapolation/interpolation for hybrid boundary conditions.

7. Conclusions

Accurate and efficient multi-scale flow simulations by a hybrid CFD-MD simulation framework have been presented in this paper. Hybrid schemes and file-based hybrid interfaces are incorporated in a highly-reliable LAMMPS molecular dynamics package and a verified in-house incompressible CFD code. They are virtually integrated as a single BigJob framework.

Numerical issues which harm the accuracy of a hybrid solution have been explored. First, sampling multiple independent replicas has been introduced to refine the sampling noise of an individual solution and to explore to the low-speed flow regimes. This approach is superior to simulating a single large-scale problem set which is technically bound by computing capacity. The application to a Couette flow simulation in O(10) m/s velocity field is the first

successful report of a moderate-speed flow simulation using a hybrid CFD-MD approach. Also, a prediction-correction approach has been designed for the accurate unsteady simulation. This approach acquires better solution by enabling the imposition of interpolated hybrid boundary conditions. The application to the oscillating boundary problem expresses that the current approach diminishes the overshoot/undershoot phenomena in the conventional methods. We emphasize that these numerical investigations ease the challenge to the hybrid simulation and broaden the application area.

Along with numerical issues, computational issues for the efficient coupled simulations have been also discussed. We introduced a BigJob framework which provides the co-scheduling capability among logically separated sub-tasks. A simple load-balancing function is also implemented on a BigJob framework, to achieve the load-balancing among those distributed-yet-coupled codes. From numerical experiments, we evaluate that a BigJob is very powerful in reducing the waiting time of the coupled simulation. Also, a simple load-balancing function employed in a BigJob is effective in reducing the simulation runtime. We value that our computational experiments contribute on how to efficiently conduct coupled simulations. Our design concept and the BigJob abstraction can be applied to the similar distributed multi-scale framework development.

Acknowledgements

This work is part of the Cybertools (http://cybertools .loni.org) project and primarily funded by NSF/LEQSF (2007-10)-CyberRII-01. Important funding for SAGA has been provided by the UK EPSRC grant number GR/D0766171/1 (via OMII-UK) and HPCOPS NSF-OCI 0710874. This work has also been made possible thanks to computer resources provided by TeraGrid TRAC TG-MCB090174 and LONI resources.

References

[1] S. T. O'Connell, P. A. Thompson, Molecular dynamics continuum hybrid computations: a tool for studying complex fuid flows, Phys. Rev. E 52 (1995) R5792–R5795.

[2] X. B. Nie, S. Y. Chen, W. N. E, M. O. Robbins, A continuum and molecular dynamics hybrid method for micro- and nano-fluid flow, J. Fluid Mech. 500 (2004) 55–64.

[3] X. Nie, S. Chen, M. O. Robbins, Hybrid continuum-atomistic simulation of singular corner flow, Phys. Fluids 16 (10) (2004) 3579–3591.

[4] J. Cui, G. W. He, D. Qi, A constrained particle dynamics for continuum-particle hybrid method in micro- and nano-fluidics, Acta. Mech. Sin. 22 (2006) 503–508.

[5] Y. C. Wang, G. He, A dynamic coupling model for hybrid atomistic-continuum computations, J. Comput. Phys. 62 (2007) 3574–3579.

[6] T. H. Yen, C. Y. Soong, P. Y. Tzeng, Hybrid molecular dynamics-continuum simulation for nano/mesoscale channel flow, Microfluid Nanofluid 3 (2007) 665–675.

[7] J. Liu, S. Chen, X. Nie, M. O. Robbins, A continuum-atomistic multi-timescale algorithm for micro/nano flows, Commun. Comp. Phys. 4 (5) (2008) 1279–1291.

[8] N. G. Hadjiconstantinou, A. T. Patera, Heterogeneous atomistic continuum representations for dense fluid systems, Comput. Phys. Commun. 4 (1997) 967–976.

[9] N. G. Hadjiconstantinou, Hybrid atomistic-continuum formulations and the moving contact-line problem, J. Comput. Phys. 154 (1999) 245–265.

[10] N. G. Hadjiconstantinou, A. L. Garcia, M. Z. Bazant, G. He, Statistical error in particle simulations of hydrodynamic phenomena, J. Comput. Phys. 187 (2003) 274–297.

[11] P. K. T. Werder, J. H. Walther, Hybrid atomistic-continuum method for the simulation of dense fluid flows, J. Comput. Phys. 205 (2005) 373–390.

[12] E. M. Kotsalis, J. H. Walter, E. Kaxiras, P. Koumoutsakos, Control algorithm for multiscale flow simulations of water, Phys. Rev. E. 79 (2009) 045701.

[13] E. G. Flekkϕy, G. Wagner, J. Feder, Hybrid model for combined particle and continuum dynamics, Europhys Lett. 52 (2000) 271–276.

[14] G. Wagner, E. G. Flekkϕy, J. Feder, T. Jossang, Coupling molecular dynamics and continuum dynamics, Comput. Phys. Commun. 147 (2002) 670–673.

[15] R. Delgado-Buscalioni, P. V. Coveney, Continuum-particle hybrid coupling for mass, momentum and energy transfers in unsteady flow, Phys. Rev. E 67 (046704) (2003) 1–13.

[16] R. Delgado-Buscalioni, P. V. Coveney, Usher: An algorithm for particle insertion in dense fluids, J. Chem. Phys. 119 (2) (2003) 978–987.

[17] R. Delgado-Buscalioni, P. V. Coveney, Hybrid molecular-continuum fluid dynamics, Phil. Trans. R. Soc. Lond. A 362 (2004) 1639–1654.

[18] G. Giupponi, G. D. Fabritiis, P. V. Covney, A hybrid method coupling fluctuating hydrodynamics and molecular dynamics for the simulation of macromolecules, J. Chem. Phys. 126 (2007) 154903–1–154903–8.

[19] S. E. Rosers, D. Kwak, An upwind differecing scheme for the time-accurate incompressible navier-stokes equations, AIAA J. 28 (1990) 253–262.

[20] S. Yoon, A. Jameson, Lower-upper symmetric-gauss-seidel method for the euler and navier-stokes equations, AIAA J. 26 (1988) 1025–1026.

[21] M. M. Rai, S. R. Chakaravarthy, An implicit form of the osher upwind scheme, AIAA J. 24 (1986) 735–743.

[22] B. V. Leer, Towards the ultimate conservative difference scheme. v. a second order sequel to godunov's methods, J. Comput. Phys. 32 (1979) 101–136.

[23] LAMMPS, `http://lammps.sandia.gov`.

[24] J. L. Steger, F. C. Dougherty, J. A. Benek, A chimera grid scheme, Advances in Grid Generation FED Vol. 5 (1983) 59–69.

[25] Y. Sugita, Y. Okamoto, Replica-exchange molecular dynamics method for protein folding, Chemical Physics Letters 314 (1999) 141–151.

[26] SAGA, `http://saga.cct.lsu.edu` (2011).

[27] O. G. Forum, http://www.ogf.org.

[28] A. Luckow, S. Jha, J. Kim, A. Merzky, B. Schnor, Adaptive Replica-Exchange Simulations, Royal Society Philosophical Transactions A (2009) 2595–2606.

[29] S.-H. Ko, C. Kim, O.-H. Rho, K. W. Cho, Load balancing for cfd applications in grid computing environment, KSAS International Journal 5 (2004) 77–87.