# A Hybrid CFD-MD Simulation Framework with Coscheduling and Load Balancing

Soon-Heum Ko[1], Nayong Kim[1], Abhinav Thota[1,2] (?),
Dimitris Nikitopoulos[3], Dorel Moldovan[3] (?), Shantenu Jha[*1,2]

[1]*Center for Computation & Technology, Louisiana State University, USA*

[2]*Department of Computer Science, Louisiana State University, USA*

[3]*Department of Mechanical Engineering, Louisiana State University, USA*

[*]*Contact Author*

## Abstract

****Jeff: Title needs change*
*Something*

## I. Introduction and Motivation

The need for accurately solving micro/nanoscale flow field is increasing these days with more invent (or design/development) of biofluidic systems.
***Jeff: Necessity of solving micro/nanoflow: 1 Paragraph

A hybrid CFD-MD approach is a modern numerical technique to resolve the micro/nanofluid flow satisfying both accuracy and efficiency.
***Jeff: Briefly introducing the hybrid CFD-MD: advantages from the problems in conventional CFD and MD: 1 Paragraph

Severals of numerical researches have been performed in developing/improving numerical techniques for the hybrid CFD-MD simulation or figuring out the flow physics in nano-scale systems. [**?**], [**?**], [**?**], [**?**], [**?**]
***Jeff: About the previous works: 2 Paragraphs

Though much improvement have been made on developing numerical techniques for the hybrid CFD-MD simulation, still a lot of issues are raised on refining the hybrid CFD-MD scheme.
***Jeff: Raise the issue and our focus of parametric study: 1 paragraph

Meanwhile, integrating two (or more) application domains for a multi-physics research raises another issue in view of computational science: how to couple two independent software packages from different disciplines and control/manage the operation flow of separate codes within the computer systems.
***Jeff: Issues of coupling on supercomputers: coscheduling and load balancing. 1 paragraph

Conceptually, the adoption of a container job can be an answer to resolve the coscheduling and load balancing issues of running a coupled multi-task simulation.
***Jeff: Introduce container job and related researches from other researchers: show their limitation as well - i.e., LB issue, etc.. Include REMD, co-scheduling of multiple tasks via SGE in Ranger. 2 paragraphs

Collectively, we are focusing on investigating numerical issues of applying the hybrid CFD-MD scheme to a nanofluidic system, as well as designing and developing an efficient runtime framework for a coupled multi-component simulation.
***Jeff: Tell the objectives of this research: Develop a Hybrid CFD/MD Simulation Framework including Domain Science Application Codes and Computational Framework. Mention that 'we basically used an in-house CFD code and famous LAMMPS MD package and modified both codes for coupled multi-physics simulation. We focus more on building an efficient coupled multi-task environment for hybrid CFD-MD simulation.' Also include the composition of chapters in the follow. 2 paragraphs

## II. A Hybrid CFD-MD Simulation Approach

### A. Macroscopic Flow Characteristics by Continuum Hypothesis

CFD simulations are mostly the procedure of numerically solving Navier-Stokes equations, which is the formulation of Newton's second law to a sufficiently large flow system where the continuum hypothesis is commonly accepted.
***Jeff: Brief Introduction of CFD: 1 paragraph. What is continuum hypothesis and how small is the acceptable range.

In this research, we have based on the "in-house" incompressible CFD code [**?**] as the CFD component for the coupled simulation.
***Jeff: Features of a CFD Code: Governing equations and discretization / numerical methods. 2-3 paragraphs. Also mention that the code is "parallelized" and additional implementation of coupling scheme on CFD part has been done, which is to be discussed in the below.

### B. Detailed Flow Simulation with Intermolecular Effect

What is MD? by Nayong
***Jeff: Brief Introduction of MD

Governing equations, numerical schemes inside: by Nayong
***Jeff: Features of an MD Code: Governing equations with force field/ numerical methods

### C. Implementation of Hybrid Schemes

The hybrid CFD-MD approach basically starts from the concept that the intertial force dominates the system while intermolecular effect cannot be negligible in the low-speed flow region. As given in Fig. 1-(a), CFD approach analyzes the macroscopic flow characteristics with the moderate flow velocity, while MD independently solves low-speed flow region. These two domains overlap in the middle, where CFD and MD exchange their solutions.
***Jeff: How to Couple: Schematic of CFD and MD zones. 1 paragraph.

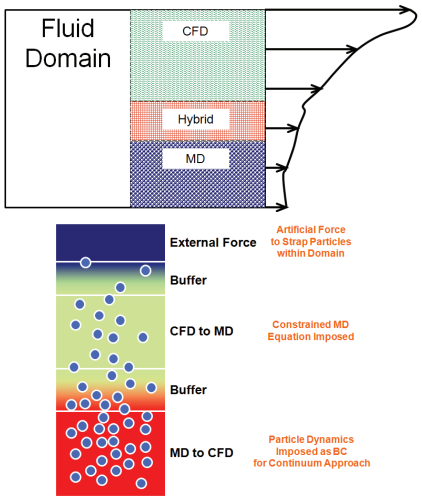


**Fig. 1.** Title: CFD-MD Schematic

The overlap region can be divided to five layers, as given in Fig. 1-(b).
***Jeff: Meaning of each layer within overlap region. 1 paragraph.

In detail, different numerical modelings are applied on these layers.
***Jeff: Numerical modeling and implementation on each solver. 3 paragraphs. Start with external force layer: specify the equation, mention that 'this artificial force modeling is implemented on MD package, to trap escaping particles within the computational domain'. Next paragraph about CFDtoMD: Discuss the meaning of constrained MD equation and specify how the code is changed, mention that 'this constrained MD equation specifies that particles in this layer eventually follow the averaged flow speed from the continuum approach.' Last paragraph about MDtoCFD: express that the average of particle velocity in this layer is applied to CFD's boundary condition, changing the CFD code.

Figure 2 expresses the coupled simulation flow.
***Jeff: Explain the flow of each simulation code in detail. Mention about when and where the data exchange takes place: if needed, have additional figure to explain data exchange interval with the evolution timeline. Mention also which data to exchange. Mention that 'we

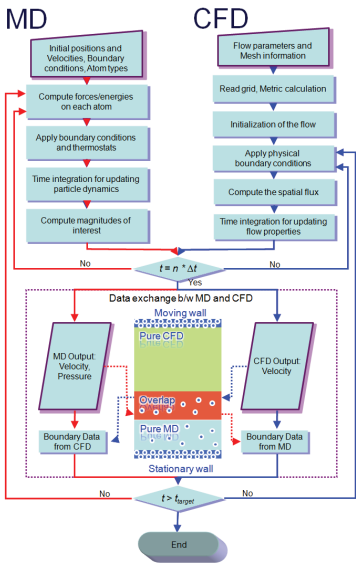


**Fig. 2.** Title: Flowchart of Codes

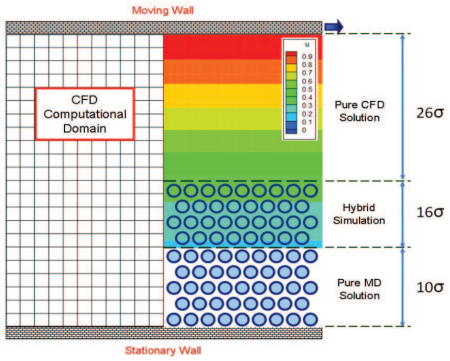


**Fig. 3.** Title: Channel System

currently adopt the file-based data exchange mechanism.' 2 paragraphs.

### D. Numerical Result of a Nano-scale Couette Flow Simulation

The geometry of a nano-scale channel along with the size of each domain is given in Fig. 3.
***Jeff: Description of a problem: 1 paragraph. Include size of each domain, layers in hybrid region, velocity of an upper wall.

The result of a Couette flow in this channel is shown in Fig. 4.
***Jeff: Solution of Couette Flow: Compare the result with pure CFD, pure MD and Nies work. Also discuss

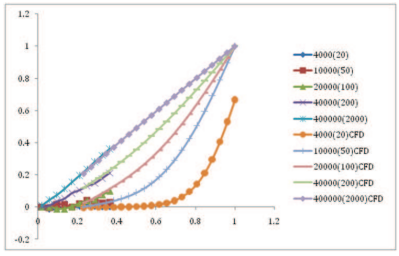


**Fig. 4.** Title: Couette Flow Solution

the limit of current simulation in the next paragraph: it's an unrealistic simulation, algorithms and models are still improving (like governing equations in CFD - secondary flow, flux scheme for low speed flow, artificial external force in hybrid region) Totally 2 paragraphs.

Parametric study or unsteady simulation result to be included here
***Jeff: 1-2 paragraphs.
Parametric Study of (choose one of the following issues for more numerical results)
a. Domain Size (CFD/MD/Hybrid) for Solutions Accuracy and Efficiency
b. Layer Size (CFD/MD) and Its Position, Imposing the Velocity via Function
c. Sampling (Averaging) Time Scale and Its Interval

## III. Construction of a Coupled Multi-physics Simulation Framework with Co-scheduling and Load Balancing

### A. SAGA and SAGA-based Frameworks for Coupled Multi-component Computations

***Jeff: Refer to Section 3 in CCGrid2010.

***Jeff: Exclude the mention about the multiple BigJobs. Also, focus on explaining BigJob's advert service module in detail, including referring to Andre's paper

### B. Load Balancing for Coupled Multi-Physics Simulation

***Jeff: Based on Section 4 in CCGrid 2010: Refine LB, include motivation etc. (e.g., Not touching application

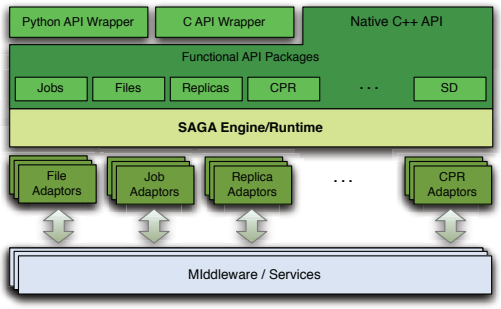


**Fig. 5.** Layered schematic of the different components of the SAGA landscape. At the topmost level is the simple integrated API which provides the basic functionality for distributed computing. Our BigJob abstraction is built upon this SAGA layer using Python API wrapper

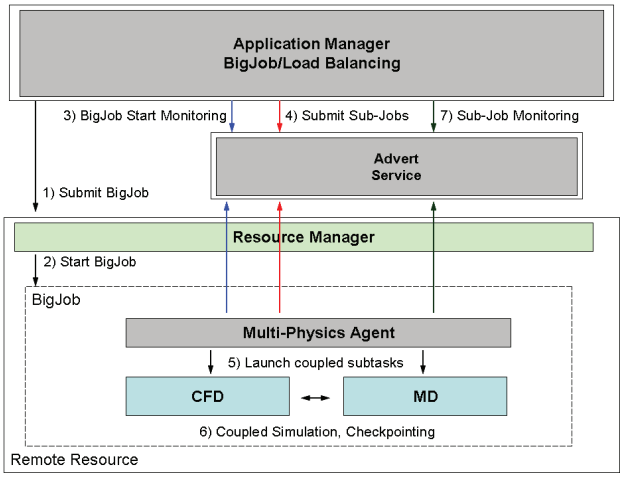


**Fig. 6.** Architecture of the Controller/Manager and Control Flow: Application manager is responsible for job management including BigJob and sub-job submission, their status monitoring functions. We implement a load balancing module, and migration service based on job information. Application agent system resides on each HPC resource and conducts job information gathering and also communicates with the application manager via the advert service

codes)

## C. Implementation of an Execution Framework to Support Multi-physics Applications

***Jeff: Refer to Section 5.a in CCGrid 2010

The application manager along with its load balancing functionality is implemented as follows.
***Jeff: Features of an application manager: Mention

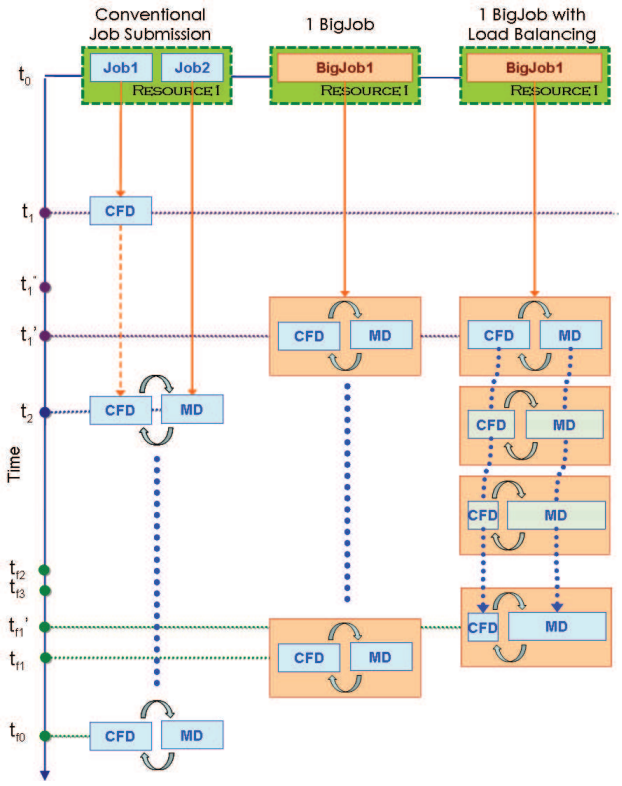


**Fig. 7.** Comparison of dependencies encountered for coupled simulations submitted as conventional jobs, to the scenario when using Pilot-Jobs. Here we use only 1 BigJob (S1). The conventional mode of submission experiences three phases based upon their queue state: (i) All jobs are waiting: $(t_1 - t_0)$; (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.

that it is written in PYTHON, load balancing function is implemented inside, etc. 1 paragraph

To use the BigJob and its load balancing function, application codes also need minor modifications.
***Jeff: Changes of Application Codes. 1 paragraph

The generation of an application manager and the changes in application codes raise the possible simulation scenarios as given in Fig. 7.
***Jeff: Explain S1 and S1-LB scenarios. Just copy from Section 5.a in CCGrid 2010

### D. Performance Analysis of a Multi-physics Simulation Framework

***Jeff: Refer to Section 5.b and 5.c in CCGrid 2010

The benefit of a BigJob as a way of reducing the waiting time on the local queueing system can be discussed from our preliminary test of the relationship between the requested number of processors and the waiting time.
***Jeff: Queue Wait Time Analysis: Preliminary Test. Waiting Time with Different PX Requests, Wall Time Limit (Mention that BQP is not perfect, especially measuring inactive waiting time of TWO CONVENTIONAL JOBS). Include Table 1 here.
More test needed: Submit 128, 256 and 512 PX jobs with 6, 24, 48 wall limit times synchronously - to satisfy the same condition among testsuites. Also include BQP test at the same condition.

A BigJob submission shows the saving of waiting time on the queue, not only the bigger job size has the higher priority in the queueing system but also one BigJob submission eliminates the inactive idling time of first allocated job in conventional job submission.
***Jeff: Queue Wait Time Analysis: Waiting time of a BigJob and two conventional jobs. Emphasize the inactive time as well - it makes the conventional job submission to have longer wall time limit and it will even reduce the possibility of getting the backfilling capability. Include Table 2 here.
More test needed: One BigJob and two conventional jobs at 2-3 locations, with 3 PX settings (128,256,512) and 3 wall time limit (6,24,48)
1. BigJobs Faster Start by Queuing Policy
2. Co-scheduling and the Safety: Different wall limit time between BigJob and Conventional Jobs due to Highly Unpredictable Inactive Time

The simulation runtime with/without a BigJob is given in table 3.
***Jeff: Runtime Analysis: Comparison of LB with conventional simulation time: different system size (with different ratio between CFD and MD), different number of simulation loop and different interval of simulation loop. Table 3 with 2 graphs - change of CPU allocation to each subjob with iteration time in 2 simulation cases.

## IV. Next Step: Further Refinement

## V. Conclusions

## Acknowledgment

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). This work has also been made possible thanks to computer resources provided by LONI. We thank Andre Luckow for initial work on BigJob, Lukasz Lacinski for help with SAGA deployment (via HPCOPS NSF-OCI 0710874) and Joao Abecasis for his work on the SAGA Condor adaptors.