Validated thermal air management simulations of data centers using remote graphics processing units.

Johannes Sjölund*†, Mattias Vesterlund*, Nicolas Delbosc‡, Amirul Khan§, Jon Summers*§
*RISE SICS North, Luleå, Sweden
†Luleå University of Technology, Luleå, Sweden
‡Dassault Systèmes Madrid, Madrid, Spain
§University of Leeds, Leeds, United Kingdom
Email: {johannes.sjolund,mattias.vesterlund,jon.summers}@ri.se, {a.khan,j.l.summers}@leeds.ac.uk

***Abstract*—Simulation tools for thermal management of data centers help to improve layout of new builds or analyse thermal problems in existing data centers. The development of LBM on remote GPUs as an approach for such simulations is discussed making use of VirtualGL and prioritised multi-threaded implementations of an existing LBM code. The simulation is configured to model an existing and highly monitored test data center. Steady-state root mean square averages of measured and simulated temperatures are compared showing good agreement. The full capability of this simulation approach is demonstrated when comparing rack temperatures against a time varying workload, which employs time-dependent boundary conditions.**

***Index Terms*—data centers, thermal management, CFD, GPU, lattice Boltzmann methods**

I. Introduction

The electrical consumption of Information Communication Technology (ICT), data networks, personal devices and data centers, showed an annual growth of 7% during 2007 to 2012 and estimated to double every 10 years. The full ICT electrical energy footprint was estimated to be 900TWh in 2012, corresponding to 4.6% of the total global electricity consumption, which had increased from the 3.9% in 2007 [1].

In a study where 100 data centers were examined, the metric called power usage effectiveness (PUE) was examined and values were found to be from 1.33 to 3.85, with an average of 2.13 and where closer to 1 is most favourable. Nearly half had a PUE above 2, signifying an energy overhead on the ICT of more than 100%. Furthermore, air conditioning systems were often found ineffective using between 21 to 61% of the total facility energy, averaging at 38% [2].

In recent years there has been notable changes in the data center operating environmental conditions, leading to higher operating temperatures to save cooling costs and the adoption of economization. There is a wider acceptable operating temperature range that can reduce internal server fan speeds and result in raised exhaust temperatures [3]. However, the facility fan power can be at a higher level than experienced when operating according to best practices, since there is a requirement for greater fan work as part of the facility equipment when higher temperature differences are to be generated [4]. To ensure effectiveness of large data centers, tools that enable sensible decisions about the cooling arrangement are needed that accounts for cooling/heating sources and external thermal factors [5].

Thermal management of data centers includes the crucial matter of preventing hot air exiting the rear of the racks and recirculating to be ingested by front inlets or even the over provision of cold air that returns to the cooling units without passing through the racks. These issues often depend on supplying sufficient cold air to the front of the racks. Thermal management can also be improved by the addition of curtains, partitions, drop ceiling and ducting that targets cool air to or hot air from the racks [6]. When the airflow demand of the racks is targeted or contained and matches that supplied by the cooling units, then proper cooling is generally assured with minimised air short circuits and mixing of cold and hot air streams [7].

The preferred approach to cooling of data centers includes free cooling and aisle containment, since these technologies can yield greater elevated temperatures and improved efficiency gains over legacy data centers [2]. With pressurised aisle containment, it is possible for racks to experience up to 20% leakage, where the supplied air bypasses the servers when there is a large pressure difference between the hot and cold aisles. Improving the rack design and blocking leakage paths can lead to nearly 9% performance improvements. By optimizing the pressure difference between the hot and cold aisles, it is also possible to achieve a 16% reduction in total energy consumption [8].

Developing analytical optimization models to represent the dynamics and physical characteristics of the different subsystems in a data center can reduce electricity costs by 3% and the ventilation and air conditioning energy by 8% [9]. One often studied subsystem in a data center is the dynamics of the airflow, where air temperatures and velocities are of interest to determine the most appropriate layout of the data center for new sites or performing root cause analyses in terms of poor thermal management of existing data centers. This is achieved by creating numerical models of the data center where simulation software can be employed as a predictive tool, making use of Computational Fluid Dynamics (CFD) methods.

II. Data Center Computational Fluid Dynamics

The application of CFD enables the creation of a complete computer model of the data center, with raised floor, cooling units, perforated tiles and server racks. Such simulations provide detailed distributions of air velocity, pressure and temperature in the data hall. It is then possible to determine airflow and thermal management issues, such as hotspots [7], which can be prevented without overcooling the entire data center and in addition, trends in the rack inlet temperature distributions can be captured. Moreover, transient simulations can be used to analyse different failure scenarios, such as determining the time until overheating of the IT equipment when either the chilled water pump or Computer Room Air Handling (CRAH) fans cease to work [10].

CFD can be applied to exergy destruction analyses to identify zones with inefficiencies and energy losses that results from inadequate deliverly of server air cooling [11]. In contrast, CFD can also be used for detailed analyses of different floor tile designs and their influence on thermal performances of aisles from features, such as vane angle, number of grills and their percentage openness [12]. The CFD simulations must be validated against reality to provide trustworthyness of the model, which includes an appropriate choice of turbulence model, constraints and boundary conditions that can yield simulation results closer to reality [13].

Transient and real-time data center simulations are challenging, since the dynamic environment thermal profile must be generated fast enough to capture server thermal generation interference [14]. By using real-time visualization of an entire data center, the thermal transients of individual servers can be assessed, which helps to determine sources of potential hotspots before they occur i.e. servers exceeding their specified temperature threshold [15].

The lattice Boltzmann method (LBM) is a recently developed method for modeling fluid mechanics problems [16]. The method has been applied to many physical situations including airflows in data centers [17], where a successful comparison of the simulated results against CFD simulations based on the finite volume and finite element methods is demonstrated. Comparing different computational approaches for data center airflow simulations, the LBM demonstrates clear advantages in computational performance and applicability for transient and real-time flow simulations if executed on GPUs [18]. The purpose of this paper is two-fold:

- to demonstrate the real-time simulation and visualisation capability of the LBM on GPUs that are remotely located inside a data center without any performance degradation, and
- to validate the LBM simulation air velocity and temperature field results against real measured results from a highly monitored test data center.

The contribution of this work is therefore a strategy for executing an existing optimised GPU based LBM code developed by Delbosc [19] on remote GPUs and to validate for the first time the numerical results of data center air flows against detailed experiments.

III. The Lattice Boltzmann Method

The LBM is originally based on the concept of a *cellular automaton*, which is a discrete computational model based on a regular grid of *lattice* sites. The grid can have any finite number of dimensions and each site has a finite number of states and at time $t = 0$ the state of each site is initialized to some predetermined value. As time progresses in discrete steps such that $t = t + \Delta t$ the state of each site is changed according to fixed rules that involve the state of the lattice site and that of its neighbours on the grid.

Applying the LBM as a CFD modeling tool, the states of the lattice sites are given by continuous distribution functions, and the rule of the automaton is based on the discrete lattice Boltzmann equation, which is a special discretisation of the Boltzmann equation [20].

A. The Boltzmann Equation

The Boltzmann equation is derived from *kinetic theory*, where gases are composed of particles following the laws of classical mechanics and particle interactions are described statistically. Groups of particles are affected by adjacent groups and particles streaming between each other in a billiard ball-like fashion. A single particle distribution function $f^{(1)}$ is sufficient to describe all properties of non-dilute gases [21], with

$$f^{(1)}(\vec{x}, \vec{p}, t) \tag{1}$$

which gives the probability of finding a particle at location given by the displacement $\vec{x}$ with momentum $\vec{p}$ at the time t.

When an external force $\vec{F}$ acts on the particles, their future positions and momentum are described by

$$f^{(1)}(\vec{x} + d\vec{x}, \vec{p} + d\vec{p}, t + dt) \tag{2}$$

as long as no collisions occur between particles. This is called the particle streaming motion and captures fluid advection, which is the transport of fluid properties by the bulk motion. Accounting for particle collisions, the evolution of this distribution function by particle interactions over time can be described by the *Boltzmann Equation*

$$\left(\vec{u} \cdot \frac{\partial}{\partial \vec{x}} + \vec{F} \cdot \frac{\partial}{\partial \vec{p}} + \frac{\partial}{\partial t}\right) f^{(1)}(\vec{x}, \vec{p}, t) = \Gamma^{(+)} - \Gamma^{(-)}. \tag{3}$$

The left hand side describes the streaming motion introduced by the external force $\vec{F}$ during the time interval dt. The right hand side contains the so called *collision operators*, which act on the fluid velocity $\vec{u}$. $\Gamma^{(-)}$ represents the number of particles starting at $(\vec{x},\ \vec{p})$ and not arriving at $(\vec{x} + d\vec{x},\ \vec{p} + d\vec{p})$ due to particle collisions. Conversely, $\Gamma^{(+)}$ is the number of particles not starting at at $(\vec{x}, \vec{p})$ but ending up at $(\vec{x} + d\vec{x},\ \vec{p} + d\vec{p})$ [21]. The right hand side of the Boltzmann equation captures fluid diffusion.

The combined streaming and collision motion on the particle level as represented by the Boltzmann equation yields what is called *advection-diffusion* transport processes that are central to modeling fluid flows.

B. Discrete Lattice Boltzmann

The goal of LBM programs is to provide a numerical solution to the Boltzmann Equation, using a discrete approximation called the *Discrete Lattice Boltzmann Equation* which can be written as [19]

$$f_i(\vec{x}+\vec{e_i}\Delta t, t+\Delta t) = f_i(\vec{x},t) + \Gamma(f_i(\vec{x},t)). \qquad (4)$$

Therefore the basis of the LBM algorithm is the discretization of space and time into a lattice of suitable number of dimensions. The unique solution to the equation varies depending on these properties and the initial and boundary conditions of the fluid domain.

For a specific problem domain, the conversion of length in m and velocity in ms^{-1} defines the lattice time step, Δt. By defining a dimensional physical length, L_{phys} and dimensional velocity, V_{phys}

$$\Delta t = \frac{C_L}{C_U} \text{ where } C_L = \frac{L_{phys}}{L_{lbm}} \text{ and } C_U = \frac{V_{phys}}{V_{lbm}}. \qquad (5)$$

Therefore the number of lattice sites, L_{lbm}, in any direction and the choice of V_{lbm}, less than 0.2 for stability, dictates the time step. A grid spacing Δx on a 3D lattice can be expressed from L_{phys} and the number of lattice sites in the whole domain $N = N_x \cdot N_y \cdot N_z$ with

$$\Delta x = \frac{L_{phys}}{\sqrt[3]{N}} \qquad (6)$$

Time in the simulation domain is not continuous as in nature, but is measured in constant time steps Δt. A time step is completed after all sites in the lattice have been updated from the previous time. This means that for each update, a constant time period of Δt seconds in simulated time has passed. If Δt is equal to, or greater than the time it took to compute the lattice update, the simulation is considered real-time or faster.

Each direction vector $\vec{e_i}$ on the 3D D3Q19 lattice shown in Figure 1 is called a *lattice velocity* vector and is scaled such that during a time step, Δt, a particle can move exactly from one site to an adjacent one. When velocity is measured in lattice units per time step ($\Delta x\ \Delta t^{-1}$) the magnitude of the i^{th} lattice velocity is given as $\|\vec{e_i}\|$.

C. The LBM Algorithm

The collision operator Γ in the Boltzmann Equation (3) can be implemented in multiple ways, the simplest employs the Bhatnagar-Gross-Krook (BGK) approximation [21], which is modeled by

$$\begin{aligned} f_i(\vec{x}+\vec{e_i}\Delta t, t+\Delta t) &= f_i(\vec{x},t) - \frac{\Delta t(f_i(\vec{x},t) - f_i^{eq}(\vec{x},t)}{\tau} \\ &+ \frac{\Delta t w_i \vec{e_i}\cdot\vec{F}}{c^2}. \end{aligned} \qquad (7)$$

As with the Boltzmann Equation (3), Equation (7) contains a streaming part, represented by $f_i(\vec{x}+\vec{e_i}\Delta t, t+\Delta t) = f_i(\vec{x},t)$, a collision term $\frac{\Delta t(f_i(\vec{x},t)-f_i^{eq}(\vec{x},t))}{\tau}$, where $\vec{x}$ is the particle displacement and forcing term $\frac{\Delta t w_i \vec{e_i}\cdot\vec{F}}{c^2}$. The function f_i^{eq} is called the *equilibrium distribution function* given as [21]

$$f_i^{eq}(\vec{x},t) = w_i\rho(\vec{x},t)\left(1 + \frac{\vec{e_i}\cdot\vec{u}}{c_s^2} + \frac{(\vec{e_i}\cdot\vec{u})^2}{2c_s^4} - \frac{\vec{u}^2}{2c_s^2}\right) \qquad (8)$$

where c_s is the speed of sound and the vector product is defined as the inner product. The macroscopic fluid density and velocity are given by

$$\rho(\vec{x},t) = \sum_i f_i(\vec{x},t) \text{ and } \vec{u}(\vec{x},t) = \frac{1}{\rho}\sum_i \vec{e_i} f_i(\vec{x},t) \qquad (9)$$

where the lattice weights are given in Table I. In equation (7)

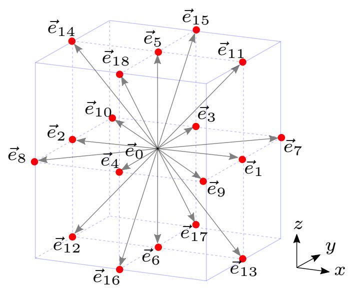

Fig. 1. D3Q19 lattice site with its lattice velocities $\vec{e}_i$.

TABLE I
D3Q19 LATTICE VELOCITIES $\vec{e}_i$ AND WEIGHTS w_i.

$\vec{e}_0$	= (0, 0, 0)	w_0	= 1/3	$\vec{e}_1$	= (1, 0, 0)	w_1	= 1/18
$\vec{e}_2$	= (-1, 0, 0)	w_2	= 1/18	$\vec{e}_3$	= (0, 1, 0)	w_3	= 1/18
$\vec{e}_4$	= (0,-1, 0)	w_4	= 1/18	$\vec{e}_5$	= (0, 0, 1)	w_5	= 1/18
$\vec{e}_6$	= (0, 0,-1)	w_6	= 1/18	$\vec{e}_7$	= (1, 1, 0)	w_7	= 1/36
$\vec{e}_8$	= (-1,-1, 0)	w_8	= 1/36	$\vec{e}_9$	= (1,-1, 0)	w_9	= 1/36
$\vec{e}_{10}$	= (-1, 1, 0)	w_{10}	= 1/36	$\vec{e}_{11}$	= (1, 0, 1)	w_{11}	= 1/36
$\vec{e}_{12}$	= (-1, 0,-1)	w_{12}	= 1/36	$\vec{e}_{13}$	= (1, 0,-1)	w_{13}	= 1/36
$\vec{e}_{14}$	= (-1, 0, 1)	w_{14}	= 1/36	$\vec{e}_{15}$	= (0, 1, 1)	w_{15}	= 1/36
$\vec{e}_{16}$	= (0,-1,-1)	w_{16}	= 1/36	$\vec{e}_{17}$	= (0, 1,-1)	w_{17}	= 1/36
$\vec{e}_{18}$	= (0,-1, 1)	w_{18}	= 1/36				

the distribution function f_i is relaxed towards the equilibrium function f_i^{eq} at a collision frequency of $1/\tau$. The constant τ is fixed to correspond to the correct kinematic viscosity, ν, of air given by [21]

$$\nu = \frac{1}{3}\left(\frac{\tau}{\Delta t} - \frac{1}{2}\right)\frac{\Delta x^2}{\Delta t}. \qquad (10)$$

The inclusion of the energy equation via an additional particle distribution function, buoyancy effects as a body force to the momentum equation, a turbulence model and appropriate boundary conditions are described in detail in [17]–[19].

IV. THE LBM GPU CODE IMPLEMENTATION

The original codebase was written using a combination of C++, CUDA[1] and LUA[2]. The code depends on shared dynamic visualisation libraries, including the OpenGL toolkit, X

[1]https://developer.nvidia.com/cuda-toolkit
[2]https://www.lua.org

Window System (X11), OpenGL Extension Wrangler Library (GLEW), and the Joint Photographic Experts Group (JPEG) image codecs.

A. Remote OpenGL Visualization through VirtualGL

While the CUDA framework used by the LBM code did not need any graphics rendering capability to perform the CFD simulation, the user interactive part of the program, such as keyboard and mouse event handling and graphical image rendering required an X11 server with hardware accelerated OpenGL functionality. In this configuration, the application is allowed to directly access the GPU hardware through the Direct rendering Interface (DRI), allowing for hardware accelerated graphics *direct rendering* to take place.

When an OpenGL application running on a remote headless server is accessed through a remote access system such as a Virtual Network Computer (VNC) or X11-forwarding through Secure Shell (SSH), LibGL creates GLX protocol messages and sends them to the local client X11 server via a network socket. The local client then passes the messages to the local 3D rendering system for displaying on a monitor[3]. There are two main problems with this approach. First, in the case where the LBM code is executed through X11-forwarding and rendered on a local client, some openGL extensions require that the application has direct access to the GPU causing restrictions over a network. Second, 3D graphics data such as textures and large geometries can occupy several megabytes of space. Since an interactive 3D application requires tens of frame updates per second to be free of lag, indirect rendering requires an extremely high bandwidth and latency. A cross-

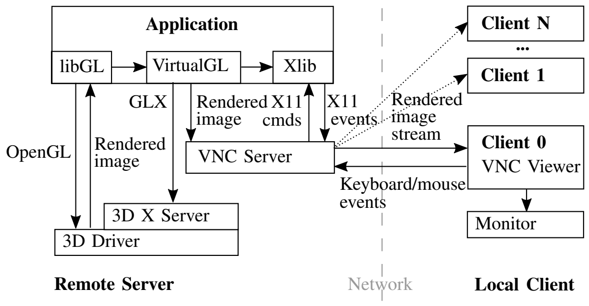

Fig. 2. In-Process GLX Forking with an X11 proxy over a network, in the form of a VNC server and client.

platform VNC client solution to the problem of OpenGL rendering on a remote server can be found in the VirtualGL[4] software toolkit. This is called *in-process GLX forking* and involves interposing application GLX calls and redirecting them to the server GPUs. The rendered images can be sent to an X11-proxy such as a VNC server. The local client can then connect to the VNC server for visualisation of the simulations running on a remote server.

[3]https://dri.freedesktop.org/wiki/libGL
[4]https://www.virtualgl.org/

B. Multithreading

The original LBM code by Delbosc [19] was written as a single-threaded application and all code was executed in a single loop. This approach limited the performance of the LBM application in the graphical OpenGL visualization part, which is not an issue when the GPU is local. Even though VirtualGL allows excellent performance for remote OpenGL rendering through VNC, it introduces an overhead when interposing GLX calls and transporting the rendered image to the VNC X11-proxy. When visualising on a local GPU, the rendering time of OpenGL visualization is negligible, but several CUDA kernel simulations steps can be performed during the VirutalGL overhead.

Adding multi-threaded support to the LBM code allows the CUDA kernel to execute as often as possible, while also allowing the user to modify the execution parameters, such as simulation boundary conditions. A single CPU thread runs in an infinite loop, always trying to execute the kernel again as soon as the previous execution has completed. Through the common kernel interface, other threads are able to signal suspend and resume of kernel executions as well as reading simulation data and setting simulation boundary conditions. Thread access to the kernel is configured and protected by Mutual Exclusion (mutex) locking to ensure no race conditions occur. The order in which mutex locking grants access to shared memory is based on different thread privileges. Adding a common communication interface enables mutex locking for thread synchronization.

The overhead from VirtualGL rendering is eliminated with respect to the amount of CUDA kernel executions performed during a certain time period. The low priority simulation kernel execution thread runs on a dedicated GPU stream as often as possible. The GPU thread responsible for rendering the OpenGL visualization copies simulation output from this thread and streams when a new visualization frame is required based on a set frame rate. Copying is asynchronous using device-to-device copy (between GPU memory banks) to a memory buffer, resulting in rendering being performed independently of simulation kernel execution. There remains a small overhead from performing the visualization and disabling the visualization by minimizing or hiding the drawing window improves the simulation performance. The technical detail of the multi-threaded implementation can be found in the masters thesis of Sjölund [22].

V. Data Center CFD Model

The racks in the test data center are configured with a central hot aisle as depicted in Figure 3, which includes a schematic of the thermal airflows. The LBM in its simplest form makes use of regular grids with Euclidian coordinates and therefore modeling sloped surfaces requires a high resolution lattice, therefore curved features have a simplified geometry, whilst maintaining the same areas of inlet and exhaust airflows.

The floor, walls and ceiling of the room are modeled using the half-way bounce-back scheme, [19], which defines zero air velocity along these boundaries. Likewise, the temperature

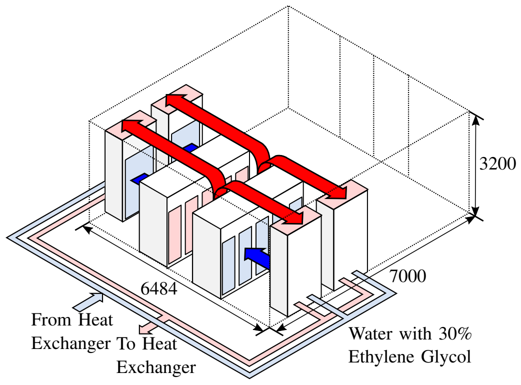

Fig. 3. Schematic heat flow in the test data center at RISE.

distribution functions were also implemented with bounce-back to maintain no heat transfer at these surfaces. The boundary conditions for the CRAC and rack inlets and outlets requires specialized definitions.

A. CRACs and Server Racks

Conditioned air inlets from the CRACs blow cold air at a constant temperature, T_{supply}, and flow rate, Q_{supply}. The return CRAC inlets take a constant static pressure p_{return} by setting the velocity and temperature gradients to zero, [19].

Each rack in the test data center contains between 16-30 servers and power consumption is monitored on a per-rack basis. The air inlet at the front of the racks is modeled using a zero-gradient boundary condition with constant flow rate, Q_{in}. Each server contains case mounted fans which provide a temperature dependent server flow rate, Q_{server}. The temperature on the back of the racks T_{out} is thus dependent on the inlet temperature, T_{in}, plus a temperature increase, ΔT, which is affected by the server workload, which in turn relates directly to its power consumption P and fan flow rate Q_{out}. This is given by [17]

$$T_{out} = T_{in} + \Delta T, \Delta T = \frac{P \cdot \nu}{Q_{out} \cdot k \cdot Pr}, \tag{11}$$

where the constants $\nu = 1.568 \cdot 10^{-5}$ m^2/s is the kinematic viscosity, $k = 2.624 \cdot 10^{-5}$ kW/m K is the thermal conductivity and $Pr = 0.707$ is the Prandtl number of air at 30°C.

B. Simulation Input Data

The data center LBM model has unknown input parameters, namely volumeteric flow rates and air exhaust temperatures of the CRAC units, $Q_{supply,i}$ and $T_{supply,i}$ with $i = 1 \ldots 4$, rack flow rates, $Q_{out,j}$, and temperature increases, ΔT_j, $j = 1 \ldots 10$. Temperature increases depend on rack power consumption P_j. Sensor data was recorded every minute on 23/01/2018 from 09:00 for 36 hours from the test data center to provide these time varying input parameters.

The CRAC air flow rate, Q_{supply}, was measured using a mass flow sensor and the air supply temperature T_{supply} was simply measured by a thermometer. The server rack power consumption, P_j, was calculated as the sum of the three monitored phases to provide the ΔT_j for each rack. However, there is no direct measurement of the air flow rate through the racks, apart from the monitored rotations per minute (RPM) of the fans. Flow rates are approximated using specifications of the rack fans and the fan *affinity laws*, where fan power is proportional to the cube of the shaft speed. This means the ratio between maximum input power P_{max} and an operational point P_{op} is equal to the cube of the ratio of maximum fan speed f_{max} to operational speed f_{op}. Around this operating point, the power ratio can be assumed to be proportional to the volumetric flow rates Q_{max} and Q_{op}. Since each server has $n_{fans} = 6$ integrated fans and experimental data logs of average fan speeds, the boundary conditions are available for server air flow rates by using an average fan speed f_{rack} in RPM of all rack servers. The total flow rate for a rack is

$$Q_{out} = f_{rack} \cdot n_{fans} \cdot n_{servers} \cdot \frac{Q_{op}}{f_{op}} \tag{12}$$

$$= f_{rack} \cdot n_{fans} \cdot n_{servers} \cdot Q_{max} \frac{(f_{op})^2}{(f_{max})^3}. \tag{13}$$

C. Simulation Output Data

Both front and back of the racks in the test data center are fitted with temperature sensor strips that provides three temperature measurements at different heights above the floor. The CRAC units contain integrated sensors that record temperatures of the intake and exhaust air.

The lattice sites in the simulation correspond to the positions of the three rack based temperature sensors and are sampled during simulation runtime. The CRAC unit intake and exhaust temperatures are averaged from the lattice sites adjacent to sites containing the boundaries.

Temperature readings from both experimental and simulation data were averaged over one minute, after which they were recorded in CVS format files for the four CRACs and ten racks.

VI. Validation and Results

Temperature measurements of the back and front of the racks was performed using temperature sensor MCP9808, which according to specification has a ±0.5°C accuracy. The CRACs have integrated sensors for temperature and mass flow rate with an unknown sensor accuracy.

Different lattice resolutions were tested and 36 lu per metre, or 2.7 cm per lu, resulted in $\approx 7 \times 10^6$ lattice sites and reasonably accurate flow rates with tractable computational times. The simulated average root mean square (RMS) temperature difference of the CRAC inlets were found to be approximately 1° C off from experiments. The steady-state temperature differences between measured and simulated for the rack inlets (front) and exhausts (rear) are shown in Table II, the average RMS difference between simulation and measured at the lowest point in the racks was accurate to within ±1°C, the middle within ±2°C and at the top ±4°C.

Since the LBM simulations are time-dependent it is possible to look at varying the power consumption of the racks. This is achieved in the test data by changing the server

TABLE II

RMS of the difference between simulated and measured temperatures in °C at different rack positions.

Rack	Front Bot	Front Mid	Front Top	Back Bot	Back Mid	Back Top
1	0.275	2.67	3.28	2.28	2.59	7.07
2	0.838	1.8	4.05	3.32	2.64	3.29
3	1.04	1.34	3.82	5.82	3.12	2.44
4	0.617	2	5.28	2.21	1.4	1.07
5	0.78	2.22	3.03	0.859	2.46	1.89
6	1.11	1.25	1.63	1.43	1.06	5.11
7	0.226	1.31	1.17	2.59	1.99	5.72
8	1.11	1.19	4.48	1.4	2.53	0.653
9	0.656	1.63	4.71	0.859	1.57	0.707
10	0.665	3.04	4.46	2.36	3.93	1.93

workloads, which is a feature of the test data center. With a time varying power consumption profile as an time varying boundary condition it is then possible to obtain a time series of the rack temperatures. Figure 4 shows the middle front and rear temperature as a function of time against the varying power consumption profile of rack 6 in the data center.

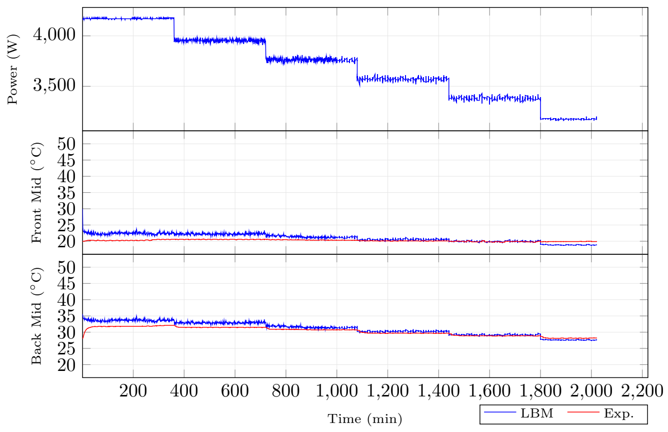

Fig. 4. Comparison of simulated and experimental temperatures for rack 6.

VII. Conclusions

The computational and visualisation advantages of implementing an LBM code with OpenGL functionality using VirtualGL offers the possibility to execute thermal management interactive simulations of data centers on remote GPUs. An existing single-threaded LBM on GPU code has been successfully augmented to a multi-threaded version for increased throughput when VirtualGL for remote GPU execution is used.

The multi-threaded version is then configured to acquire time-varying boundary conditions to validate the simulations against a highly monitored test data center. The validation results demonstrate a good comparison, however there is divergence in the measured and simulated temperatures at increased distances from the data center floor. It is speculated that the buoyancy and turbulence modeling do not fully represent reality in addition to the fact that servers are not individually modeled, but lumped together at the rack level.

References

[1] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, “Trends in worldwide ICT electricity consumption from 2007 to 2012,” Comput. Commun., vol. 50, pp. 6476, Sep. 2014.

[2] J. Ni and X. Bai, “A review of air conditioning energy performance in data centers,” Renew. Sustain. Energy Rev., vol. 67, pp. 625640, Jan. 2017.

[3] T. C. ASHRAE, “Data Center Networking Equipment - Issues and Best Practices,” Whitepaper Prep. by ASHRAE Tech. Comm., 2015.

[4] T. C. ASHRAE, “9.9 (2011) Thermal guidelines for data processing environmentsexpanded data center classes and usage guidance,” Whitepaper Prep. by ASHRAE Tech. Comm., vol. 9, 2011.

[5] C. Conficoni, A. Bartolini, A. Tilli, C. Cavazzoni, and L. Benini, “HPC Cooling: A Flexible Modeling Tool for Effective Design and Management,” IEEE Trans. Sustain. Comput., pp. 11, 2018.

[6] J. Cho, T. Lim, and B.S. Kim, “Cooling systems for IT environment heat removal in (internet) data centers,” Journal of Asian Architecture and Building Engineering, vol. 7, no. 2, pp.387-394, 2008.

[7] S. V. Patankar, “Airflow and Cooling in a Data Center,” J. Heat Transfer, vol. 132, no. 7, pp. 117, 2010.

[8] M. Tatchell-Evans, N. Kapur, J. Summers, H. Thompson, and D. Oldham, “An experimental and theoretical investigation of the extent of bypass air within data centres employing aisle containment, and its impact on power consumption,” Appl. Energy, vol. 186, pp. 457469, Jan. 2017.

[9] L. Cupelli, T. Schutz, P. Jahangiri, M. Fuchs, A. Monti, and D. Muller, “Data Center Control Strategy for Participation in Demand Response Programs,” IEEE Trans. Ind. Informatics, 2018.

[10] J. Athavale, Y. Joshi, and M. Yoda, “Experimentally Validated Computational Fluid Dynamics Model for Data Center With Active Tiles,” J. Electron. Packag., vol. 140, no. 1, pp. 110, Mar. 2018.

[11] L. Silva-Llanca, A. Ortega, K. Fouladi, M. del Valle, and V. Sundaralingam, “Determining wasted energy in the airside of a perimeter-cooled data center via direct computation of the Exergy Destruction,” Appl. Energy, vol. 213, pp. 235246, Mar. 2018.

[12] S. Khalili, M. I. Tradat, K. Nemati, M. Seymour, and B. Sammakia, “Impact of Tile Design on the Thermal Performance of Open and Enclosed Aisles,” J. Electron. Packag., vol. 140, no. 1, pp. 112, Mar. 2018.

[13] E. Wibron, A.-L. Ljung, and S. Lundström, “Computational Fluid Dynamics Modeling and Validating Experiments of Airflow in a Data Center,” Energies, vol. 11, no. 3, pp. 644, 2018.

[14] M. A. Oxley, E. Jonardi, S. Pasricha, A. A. Maciejewski, H. J. Siegel, P. J. Burns, and G. A. Koenig, “Rate-based thermal, power, and co-location aware resource management for heterogeneous data centers,” J. Parallel Distrib. Comput., vol. 112, pp. 126139, Feb. 2018.

[15] R. Ullah, N. Ahmad, S. U. R. Malik, S. Akbar, and A. Anjum, “Simulator for modeling, analysis, and visualizations of thermal status in data centers,” Sustain. Comput. Informatics Syst., Jan. 2018.

[16] S. Chen, and G.D. Doolen, “Lattice Boltzmann method for fluid flows,” Annual review of fluid mechanics, vol. 30, no. 1, pp. 329-364, 1998.

[17] G. N. de Boer, A. Johns, N. Delbosc, D. Burdett, M. Tatchell-Evans, J. Summers, and R. Baudot, “Three computational methods for analysing thermal airflow distributions in the cooling of data centres,” Int. J. Numer. Methods Heat Fluid Flow, vol. 28, no. 2, pp. 271288, Feb. 2018.

[18] N. Delbosc, J.L. Summers, A.I. Khan, N. Kapur, and C.J. Noakes, “Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation,” Computers and Mathematics with Applications, vol. 67 no. 2, pp. 462-475, 2014.

[19] N. Delbosc, “Real-Time Simulation of Indoor Air Flow using the Lattice Boltzmann Method on Graphics Processing Unit” (Doctoral dissertation, University of Leeds), 2015.

[20] X. He, and L.S. Luo, “Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation,” Physical Review E, vol. 56, no. 6, p.6811, 1997.

[21] M.C. Sukop, and D.T. Thorne, “Lattice Boltzmann Modeling - An Introduction for Geoscientists and Engineers,”, Springer Berlin Heidelberg, 2006.

[22] J. Sjölund, “Real-time Thermal Flow Predictions for Data Centers,” Master Thesis, Luleå University of Technology University, 2018.