HLA Simulation of Agent-Based Bacterial Models

Michael Lees
Brian Logan
School of Computer Science and IT
University of Nottingham
Nottingham, UK
{mhl|bsl}@cs.nott.ac.uk

John King
School of Mathematical Sciences
University of Nottingham
Nottingham, UK
john.king@nottingham.ac.uk

ABSTRACT *Modelling and simulation form an integral part of a predictive and explanatory approach to computational systems biology. However the development of such complex simulation applications is extremely challenging, typically requiring collaborative effort from researchers with different domain knowledge and expertise, often at different locations. We believe these challenges can potentially be met by a combination of two emerging standards and their supporting middleware: the Grid and the High Level Architecture. To evaluate the suitability of the HLA as a standard for systems biology simulations on the Grid, we are developing a prototype HLA-compliant, Grid-based simulator for systems biology which we call* BACGRID. *In this paper, we focus on the distribution of* BACGRID *using HLA. We outline the systems biology problem we have adopted as a test case, briefly describe the simulation model and the architecture of the simulator, and present preliminary results of experiments to investigate the scalability of the* BACGRID *HLA distribution.*

1 Introduction

Modelling and simulation form an integral part of a predictive and explanatory approach to computational systems biology. Simulation can be used to produce *in silico* behaviours that can be compared with those observed through experimental studies. The resulting quantitative understanding of the interactions between elements of a biological system (and their relative importance) can be used to guide future experimental studies, and ultimately to discover the true organisation and *in vivo* behaviour of the system of interest. However, systems biology simulations are extremely challenging in requiring the modelling of many complex phenomena at multiple spatial and temporal scales. The development of such complex simulation applications typically requires collaborative effort from researchers with different domain knowledge and expertise, often at different locations. Such large scale collaborative model development presents significant challenges for the distributed simulation community. Work to date has not focused on inter-operability, with the result that the simulation models and model components developed by different research groups are difficult to integrate.

We believe these challenges can potentially be met by a combination of two emerging standards and their supporting middleware: the Grid and the High Level Architecture. The Grid supports e-Science through resource discovery, secure access to remote computational resources, data archiving and sharing etc., allowing virtual research teams to collaborate to solve research problems. The High Level Architecture (HLA) is an IEEE standard [4] for simulator interoperability, which supports the creation of distributed, composable simulations. The combination of these two complementary technologies offers great promise for the "on-demand" development of systems biology simulations by facilitating the reuse of existing simulation components. In addition, many of the services necessary to support dynamic composition of simulations, e.g., model discovery and matching, secure execution, migration and load balancing, sharing and archival of simulation results etc., which are not addressed by the HLA standard, can potentially be provided by the Grid infrastructure.

In recent work, some initial steps have been taken towards the composition and subsequent execution of simulation models on the Grid, e.g., HLA_GRID [15], XMSF [12]. However this work is still relatively immature and the appropriateness of the HLA for biological simulation has yet to be established. To evaluate the suitability of the HLA as a standard for systems biology simulations on the Grid, we are developing a prototype HLA-compliant, Grid-based simulator for systems biology which we call BACGRID. In this paper, we focus on the distribution of BACGRID using HLA and the potential of the HLA for systems biology simulation. (See [5] for details of the model and [16] for an overview of the HLA-Grid interface.)

The remainder of this paper is organised as follows. In section 2 we outline the systems biology problem we have adopted as a test case before briefly describing the simulation model and the architecture of the BACGRID simulator in sections 3 and 5. In section 6 we present preliminary results of some experiments to investigate the scalability of the HLA distribution and in section 7 we conclude and outline directions for future work.

2 Multiscale Bacterial Models

In recent years major advances have been made in understanding the gene and signalling networks that control the behaviour of individual cells, and the need to understand the implications of these breakthroughs at the population level is increasingly widely recognised. Biofilms comprise communities of diverse individuals which may interact in both mutually beneficial and competitive fashions. They thus provide comparatively simple (and experimentally relatively well characterised) systems in which variety, 'altruism' and antagonism all have scope to flourish as a heterogeneous population develops. They are sufficiently complex to exemplify many of the *generic* features of multi-cellular behaviour, without such complexity becoming overwhelming. They are in addition of enormous environmental, industrial and medical importance.

Although relatively simple in biological terms, the interactions within a biofilm span a vast range of spatial scales (from sub-cellular to population), with scope for generating a huge variety of emergent behaviour. Macroscale processes are susceptible to continuum modelling approaches, e.g., systems of (possibly stochastic) differential equations and differential-delay equations. However interactions between individuals (e.g., signalling systems such as quorum sensing, which result in coordinated changes in phenotype, see [14]) lead to phenomena that cannot readily be captured by traditional multiscale mathematical procedures such as homogenisation. Individual-based models are becoming increasingly widely used in this type of context (see, for example, [8, 3, 7, 2, 11, 10]). Such models use agent-based simulation techniques to investigate the interactions between (often relatively small) groups of cells and their environment. However, existing work in this area has tended to focus on the emergence of complex organisation in biofilms, with the individual cells being treated as 'black boxes' as far as possible, e.g., the use of cellular-automata approaches to describe biofilm growth.

However, to investigate many phenomena of interest, it is essential that individual-based models incorporate in an appropriate way information about the macroscale behaviour and their results must in turn be coupled back into the rules adopted in the cell-scale modelling. Accounting adequately for the relevant subcellular behaviour in a population of millions of distinct, diverse individuals in order to bridge the scales presents significant modelling and simulation challenges (for example, to fully investigate the effect of quorum sensing on colony development it is necessary to simulate bacterial colonies consisting of millions or tens of millions of bacteria) but offers the potential for significant benefits for biology and medicine.

3 The BACGRID Biofilm Model

To address these challenges, we have developed BACGRID, a simulator for biofilm models which integrates continuum models of population scale processes with individual-based models of cellular level processes. It thus attempts to incorporate, albeit in a simplified way, the types of complex signalling pathways and gene networks which are currently the subject of a vast amount of experimental study within a population model. In what follows, we focus on those aspects of the model relevant to its distribution, i.e., the key components of the model's state and the data interchange necessary for its evolution. A complete description of the model can be found in [5].

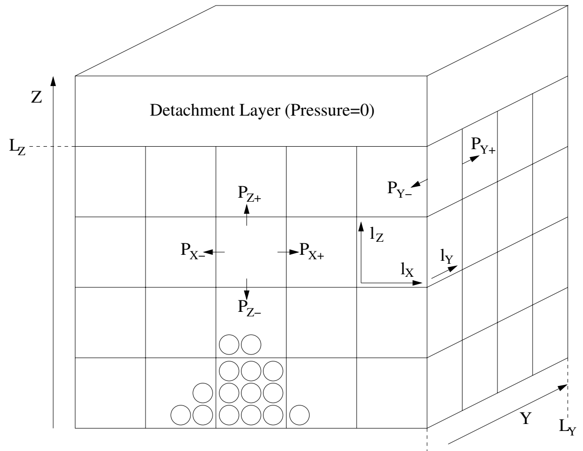

Figure 1: Computational domain

BACGRID models a 3D biofilm reactor consisting of two compartments: bulk liquid and biofilm [11] (see Figure 1). The bulk liquid compartment contains a (well mixed) solution of S different soluble substrates at constant concentrations. The biofilm compartment contains

B different types of biomass which grow on a planar support inside a rectangular box with periodic x and y boundaries. In addition to the biomass itself, the biofilm compartment contains a single type of extracellular polysaccharide (EPS) and Q different types of quorum sensing molecule. The biofilm and bulk liquid compartments are in contact and exchange solutes only by diffusion. Substrate and biomass which move beyond the x and y boundaries reappear at the opposite boundary. Bacteria, substrates and other material are assumed to be washed away once they reach the z boundary (detachment layer).

3.1 Bacterial Particles

For efficiency of computation, individual cells are aggregated into bacterial *particles* as in [11]. Each particle represents one or more cells of a single bacterial strain. The smallest possible particle is equivalent to a single cell; the largest possible particle is that which will fit in a voxel (see section 3.2). As a particle consumes substrate its radius increases until it reaches a user-specified maximum particle radius, R, at which point it divides resulting in the creation of a new particle. The maximum size of a particle, i.e., its size at division, thus determines the resolution of the model. Particles of all biomass types (including EPS) are assumed to have the same maximum radius, but their density and hence the number of bacterial cells they represent depends on the biomass type. For strains of bacteria with larger cells the corresponding particles will contain fewer cells; conversely strains with smaller cells will have particles that contain a larger number of cells.

Particles allow the use of aggregate models of continuous processes (growth, division and displacement) for small collections of cells, and also facilitate visualisation of the relative proportions of different biomass types within a voxel. However some processes must be modelled at the level of individual cells. For example, inter-cell signalling based on quorum sensing molecule (QSM) sensing, production and up-regulation encompasses both stochastic and discrete processes, and must be modelled at the level of individual cells. Particles therefore also contain information about the state of the individual cells they represent. Cells within a particle exist in one of two different states: up-regulated and down-regulated. Particles keep track of the number of up-regulated and down-regulated cells they currently contain and cells can change from one state to another at each timestep, in response to the concentration of QSMs (see section 4.4).

Up-regulated cells produce extracellular polysaccharide (EPS). The EPS produced by cells in biomass particles is aggregated into EPS particles.

3.2 Voxels

The biofilm compartment is discretised into sub-compartments or *voxels* containing particles, substrate and QSM. The size of the computational domain (L_X, L_Y, L_Z) is assumed to be an integer multiple of the size of a voxel l_X. Substrate and QSM concentrations are assumed to be uniform across an individual voxel, and the upper bound on the size of a voxel is chosen such that the substrate and QSM concentration values approximate the continuous values. The size of voxels, l_X, is chosen appropriately for the system to be modelled, with smaller values giving greater resolution at increased computational and communication cost. However the voxels are typically fairly large in relation to the size of a cell, e.g. each voxel may contain of order 10^2 particles or 10^4 cells. The bulk liquid compartment is represented by a single point for the purposes of discretisation, and this point is adjacent to all the voxels immediately below the detachment layer.

Each voxel contains zero or more particles of each biomass type (including EPS). The particles in a voxel exert a 'pressure' on the particles in the neighbouring voxels which is a function of the relative number of particles in the voxels, and these pressures are used to displace particles during the division of biomass. Each voxel has six adjacent voxels, connected at each face, which are considered in determining relative pressures, and into which particles may be displaced. Voxels have a pre-determined maximum particle capacity, N, and the pressure in the voxel is considered to be infinite when this maximum is reached. N is calculated using l_X and the maximum radius of a particle, R, assuming simple cubic packing [1]. EPS particles behave in the same way as biomass particles for the purposes of the pressure calculation.

Each particle has a notional 3D position within its containing voxel which is used for visualisation purposes (see Figure 2). These notional positions are chosen such that the particles do not overlap. The pressure model and maximum particle size are chosen to ensure that there is enough free space in the voxel for this to be possible. Note that the resolution of the model with respect to substrate and QSM concentrations is determined by the size of voxels, l_X. The resolution of the model with respect to the distribution of biomass is also determined by the size of a voxel, in that the mass of each biomass type in each voxel is known. The 3D positions of particles simply make it easier to visualise the distribution of different types of biomass.

At any given point the state of the model is specified by: the amount and distribution of each type of biomass, the number of up- and down-regulated cells of each biomass type, the amount and distribution of extracellular polysaccharide, the concentration and distribution of each type of substrate and quorum sensing molecule, and the pressure distribution.

4 Model Evolution

There are three main processes which determine the evolution of the model: the diffusion of substrate and QSM from voxel to voxel, the displacement of particles between voxels in response to proliferative pressures, and changes in the state of the particles themselves in response to the substrate and QSM concentrations in their containing voxel. The transport of substrate and QSMs between voxels corresponds precisely to a simple central-difference discretisation of the relevant continuum reaction-diffusion equations. The pressure model underlying particle displacement builds on multiphase formulations for growing populations as described in [1]. Particles are modelled as agents and implement a simple model of growth and division similar to that in [8], and up-regulation (e.g., the production of extracellular polysaccharide) in the presence of QSMs [13]. These processes interact: particles consume substrate and produce QSMs, leading to transport associated with the diffusion gradients. Consumption of substrate results in particle growth, which in turn results in increased pressures and particle displacement. Finally, the number of cells in a voxel determines QSM production and hence QSM concentration and up-regulation.

4.1 Diffusion

Diffusion is performed globally over all voxels. The diffusion algorithm iterates over each voxel using the concentration of substrate in the neighbouring voxels to determine the change in concentration at the current voxel. The bulk liquid compartment is assumed to have constant concentration and the substratum is assumed to have zero concentration. The x and y boundaries are treated as periodic, i.e., the concentration at $(-1, y, z)$ is the same as the concentration at (L_X, y, z). The volume fraction of cells is ignored when doing diffusion calculations—the uptake of substrate by cells (see section 4.2) is much more important.

At the voxel level, QSMs are treated as a substrate, i.e., the QSM concentration is constant across a voxel and diffuses between voxels. The QSM concentration(s) in a voxel at a given timestep is therefore given by the number of QSMs produced by all particles in the voxel (see section 4.4) and by diffusion of QSM between neighbouring voxels.

4.2 Growth and Division

The growth of each particle is a function of the substrate concentration(s) in the voxel containing the particle. The model of particle growth comprises three separate processes: uptake, anabolism (creation of new biomass) and maintenance [8]. At each time-step each particle consumes an amount of substrate proportional to the concentration of substrate in the voxel and the mass of the particle. The substrate consumed is split between anabolism (increase in particle mass) and maintenance. At lower concentrations, particles grows more slowly, and at sufficiently low concentrations they start to shrink (when they are unable to consume sufficient substrate to satisfy their maintenance requirement). At present we do not consider the death of particles: instead particles continually shrink until the uptake and maintenance balance and the cell becomes dormant.

As noted in section 3.1, a single particle represents a collection of cells some of which may be up-regulated and some down-regulated. When the number of cells represented by a particle increases as a result of growth, i.e., when the mass of the particle increases by the average mass of a cell of the appropriate biomass type, a new down-regulated cell is "created", by incrementing the number of down-regulated cells in the particle.

Particle division occurs when the mass of a particle exceeds the user-specified maximum particle mass. At this point, the original particle is split and a new daughter particle is created in the same voxel as the original particle. (The actual 3D position of the particle is not relevant, since it will be adjusted for the purposes of visualisation.) The mass of the daughter particle is randomly chosen between 0.4 and 0.6 of the mass of the particle at division, with the original particle retaining the remainder of the mass. The daughter particle is also allocated (up and down regulated) cells from the original particle in proportion to its mass. The cells transferred from parent to daughter are chosen randomly using a uniform distribution.

4.3 Displacement

As we are not currently considering real positions of particles within voxels (except for visualisation purposes), spreading does not occur within a single voxel. However, spreading and displacement may occur between voxels.

Particles are transferred between voxels if the difference in pressure between the voxels is large enough. The pressure in a voxel is a function of the number of particles it contains. The pressure is considered to be infinite when the voxel is full, i.e., when it contains the maximum number of particles possible at close packing. The pressure in the substratum is assumed to be infinite, and so particles cannot disperse down through the bottom layer. Particle pressure in the bulk liquid is assumed to be zero, so particles can transfer freely into the bulk liquid.

The number of particles of each biomass of type to be displaced from a voxel to each of its neighbouring voxels is determined by the pressure and a transfer coefficient which specifies how easy it is for biomass of that type to be displaced. The individual particle(s) of each biomass type to displace are chosen randomly. The direction in which to displace each of these particles is chosen probabilistically, with the probability of transferring a particle to a neighbouring voxel is a function of the pressure difference between the voxels. This approach also avoids any bias in the direction in which particles are transferred when the number of particles to be displaced is small. Particles which are displaced beyond the boundary layer, i.e., L_Z above the substratum, are simply discarded.

4.4 Inter-Cell Signalling

Quorum sensing molecules (QSM) are generated by particles and provide a form of cell to cell communication known as *quorum sensing*. The molecules, which are typically different for each strain of bacteria, control a number of aspects of bacterial growth and development, e.g., bioluminescence, population expansion by swarming, virulence, and the production of extracellular polysaccharides.

The quorum sensing mechanism involves the QSMs triggering increased expression of certain genes in the bacterium. One of the genes codes for the QSM itself, creating a positive feedback loop. The QSM therefore functions as an 'autoinducer', and bacteria will create more of the same QSM when they are surrounded by it. A cell that is in a QSM triggered state is referred to as 'up-regulated', and one that is not is referred to as 'down-regulated'. A QSM can combine with a down-regulated cell to produce an up-regulated cell and an up-regulated cell can spontaneously revert to being down-regulated (by the loss of the bound QSM). A down regulated cell produces QSMs at a low (basal) rate. Once the cell becomes up-regulated it produces QSMs at a much higher rate. The amount of QSM produced by a particle is determined by the relative number of up-regulated and down-regulated cells within the particle.

Other signalling molecules function as inhibitors, which prevent an autoinducer combining with a cell, or prevent up-regulation when a cell combines with the autoinducer. Inhibitors therefore restrict the production of QSM by the bacteria. Inhibitors are particularly interesting from a biological point of view as they allow control of up-regulation and hence development of the bacterial colony.

The probability of a cell moving from down to up-regulated or vice versa is determined by the appropriate QSM and inhibitor concentrations and the relative ease with which a QSM and the corresponding inhibitor can combine with a cell.

4.5 Model Timestep

In this section we give a high level description of a single iteration of one timestep of the model.

The state of the model at a given timestep t is determined by the state of all the voxels and all the particles at that timestep. The state of each voxel is given by the number of particles of each biomass type it contains, and the concentrations of each substrate and signalling molecule. The state of each particle is given by its biomass type, its mass, the number of up- and down-regulated cells it represents, its containing voxel and its (notional) 3D position within that voxel. Determining the state of the model at $t + 1$ involves determining, for each voxel, the change in substrate and signalling molecule concentrations due to diffusion, consumption (in the case of substrate) and production (in the case of signalling molecule), and for each particle, its change in mass over the timestep (and hence the number of cells the particle represents), the number of up- and down-regulated cells it contains and the particle's containing voxel.

The processing of the voxels at timestep t occurs in two phases. The first involves the execution of the voxels to calculate the consumption of substrate by particles, particle growth and particle division. Processing of phase one within each voxel itself occurs in three steps. Firstly, the growth step increases the mass of each particle given the concentration of substrate at this timestep. (For $t = 0$, the concentrations and number of particles are taken as parameters of the simulation.) This also gives the total consumption of all substrates by all particles in the voxel at this timestep. The second step computes the production of signalling molecule by each particle in the voxel. The third step is particle division: each particle which reached the maximum allowable mass during the growth step is split into two particles, increasing the number of particles in the voxel.

The second phase of the timestep involves computing the changes in substrate and signalling molecule concentrations due to diffusion, as a result of substrate consumption and signalling molecule production at this timestep. In parallel with the diffusion calculation, each voxel also executes a displacement step, which uses the difference in pressure between the voxel and each of its neighbouring voxels to determine movement of particles between voxels at this timestep

The timestep is then incremented and the cycle repeats with the voxels using the newly calculated concentrations and particle counts.

5 System Architecture

To facilitate distribution, the implementation of the model is decomposed into a 'diffusion module' and one or more 'model regions'. The diffusion module is responsible for diffusing substrate and signalling molecules throughout the entire computational domain. Each model region processes one or more voxels, and handles the growth and division of particles within voxels, and the displacement of particles between voxels. In addition there is a visualisation module for run-time monitoring of the simulation progress and post-simulation analysis of results (see Figure 2).

The model region and visualisation modules are implemented using the Mason agent toolkit [9]. The diffusion module is written in Java. Interaction between mod-

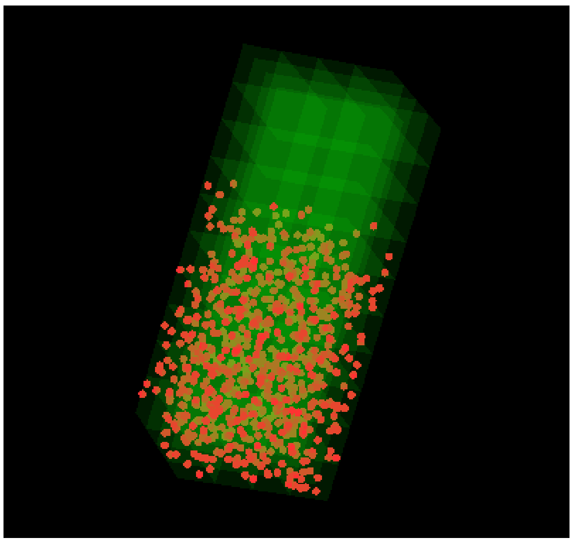

Figure 2: Model visualisation

ules can be by means of procedure calls, HLA [4] (RTI) calls, or Grid invocations. The HLA distribution uses the DMSO RTI 1.3NGv6 Java bindings, and the Grid distribution is based on HLA_GRID [15], which allows HLA-compliant simulators to be instantiated and linked using Grid services. HLA_GRID employs a Federate-Proxy-RTI architecture, in which different participants (clients) in the same simulation run their federate codes at their local sites, and the RTIexec and fedExec are executed at a remote site. Proxies act on behalf of the client federate's code and communicate with the proxies of other clients through the RTI. Federate codes and their respective proxies communicate with each other through Grid services and a Grid-enabled HLA library, which provides the standard HLA API to the federate codes, and translates RTI calls into Grid service invocations. HLA_GRID includes additional Grid services to support the creation of the RTI, discovery of federations, etc. Previous work [16] suggests that an RTI call in HLA_GRID incurs an overhead of about a factor of 5 compared to HLA. Where possible we have therefore designed the simulator to minimise the number of RTI and hence Grid service invocations.

Here we focus on the HLA distribution in which the modules are implemented as federates and communicate via RTI calls.

5.1 BACGRID Federates

A BACGRID federation consists of two types of federate: a diffusion federate and one or more model region federates. The diffusion federate is a wrapper for the diffusion module, and as such is responsible for diffusion calculations throughout the entire computational domain.[1] Each model region federate wraps a model region (i.e., a Mason process responsible for simulating a contiguous collection of voxels), and handles communication with the diffusion federate and with adjoining model region federates (for pressure calculations and particle displacement).

The diffusion federate maintains a local record for each voxel in the system containing its position, substrate and signalling molecule concentrations and the model region to which the voxel belongs. During the simulation each model region federate sends changes in concentrations of substrates and signalling molecules to the diffusion federate. When the diffusion federate has received this information for all voxels in the system, it executes its diffusion algorithm until steady state is achieved. At this point the diffusion federate reports the new concentrations of substrate and signalling molecules back to each of the model region federates which in turn update the voxels.

The model region federates not only interact with the diffusion federate but also with other model regions federates. The voxels within each model region execute in two phases. The first phase grows and divides particles within the voxels, resulting in substrate consumption, signalling molecule production and particle division. The new substrate and signalling molecule concentrations are sent to the diffusion federate as described above. Particle division results in an increase in the number of particles and hence the pressure in each voxel. Once the new pressures have been calculated, each voxel uses the difference between its own pressure and that of each of its neighbours to calculate the number of particles which must be transferred to equalise the pressure. For most voxels within a model region, its neighbouring voxels are managed by the same the same model region federate. However, for those voxels which lie on the boundaries of a model region, some of the particle counts needed to calculate the pressure differences are associated with voxels managed by neighbouring model regions. Each model region federate therefore maintains *proxy voxels* for those voxels in neighbouring model regions which are adjacent to the its boundary voxels.

Figure 3 shows two adjacent model regions and their overlapping proxy voxels. Voxel b in model region 2 has a corresponding proxy voxel (proxy b) in model region 1. The arrow indicates the direction of communication between the voxel and its proxy. Note that communication is one way—model region 1 does not update proxy b. The state of proxy b (particle count) is updated by model region 2 once voxel b has finished its growth and division step.

As the x and y boundaries of the computational domain are periodic, the model regions at each domain boundary maintain proxies for the corresponding voxels at the op-

[1]To date, we have not investigated the distribution of the diffusion module across multiple federates.

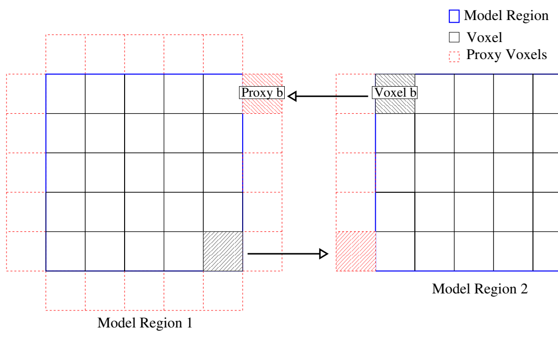

Figure 3: Two model regions and the proxy voxel overlap

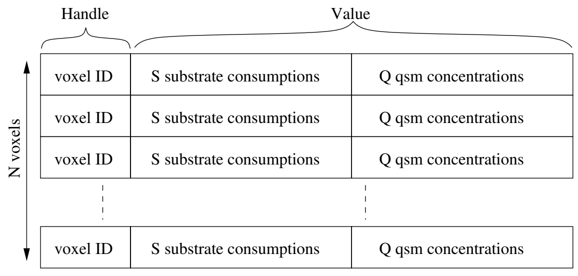

Figure 4: Structure of a `modelToDiffuser` interaction

posite extreme of the computational domain. The z axis boundaries are treated as a special case, with a particle count of infinity being returned for voxels 'below' $z = 0$, and a particle count of zero for voxels 'above' $z = L_Z$.

Once the relative pressures have been determined, each voxel computes the number of particles to transfer in each direction. A list of particles to transfer is then passed to the model region which determines if the receiving voxel is local. If the transfer is between two voxels on the same model region, the particles and their state are transferred through method calls. However, if the receiving voxel is remote the model region packages up the particle ready for transmission. Once all the particles to be transferred in a given direction have been assembled, the model region sends the particles to the adjacent model region, where they are unpacked and new particles created in the appropriate voxels.

5.2 Object and Declaration Management

To reduce the number of RTI calls (and ultimately the number of Grid service invocations in HLA_GRID), we chose to implement the communication between federates using interactions rather than updates of object attribute values.[2]

Four interaction classes were defined, all subclasses of an (abstract) `BacGridInteraction` class: `modelToDiffuser`, `diffuserToModel`, `particleCount` and `particleTransfer`. `modelToDiffuser` and `diffuserToModel` interactions are used to transfer substrate and signalling molecule concentrations from the model region federates to the diffusion federate and vice versa. `particleCount` and `particleTransfer` interactions are used to transfer particle counts between the boundary voxels of model region federate and their proxies managed by the neighbouring model region, and to transfer particle state information between model region federates during particle displacement. In each case, a single interaction is used to transfer the relevant information for all voxels in a model region (or all the boundary voxels on one face of a model region). For example, figure 4 illustrates the structure of a single `modelToDiffuser` interaction. The content of the interaction is a `handleValuePairSet`, with voxel IDs as handles and a list of substrate and signalling molecule concentrations as the value. If each concentration is represented as a double (8 bytes) and a voxel's 3D position is used as its ID (3 ints, 12 bytes), the total size of one interaction sent from a model region to the diffusion federate is then:

$$N(12 + 8(S + Q))$$

bytes, where S is the number of substrates, Q is the number of signalling molecule types and N is the number of voxels managed by the model region federate.

The publication and subscription of each federate is straightforward. The diffusion federate publishes `diffuserToModel` interactions, and subscribes to `modelToDiffuser` interactions. A model region federate publishes `modelToDiffuser`, `particleCount` and `particleTransfer` interactions, and subscribes to `diffuserToModel`, `particleCount` and `particleTransfer` interactions.

5.3 Time Management

The model timestep outlined in section 4.5 consists of two main phases each comprising a number of steps. In this section we sketch a single iteration of the model timestep in BACGRID, indicating the points at which information is exchanged between the diffusion and model region federate(s) and how federates synchronise using HLA time management services.

Each model timestep begins with the model region federates executing their voxels. Each voxel in turn executes its particles' growth and division step. All model

[2]If voxels are modelled as objects, the number of attribute updates required for communication between the diffusion and model region federates is linear in the number of voxels, and quadratic in the number of voxels on one side of a model region for communication between model region federates. In contrast, the number of interactions required for communication between the diffusion and model region federates is linear in the number of model regions and constant for communication between model regions.

region federates execute in parallel, however voxels and particles are executed sequentially within each model region. Once the growth and division step is complete, each model region has new values for substrate and signalling molecule concentrations and particle counts. Each model region federate then sends the substrate and signalling molecule concentrations to the diffusion federate. It also sends the particle counts for its boundary voxels to its neighbouring model region federates.

At this point phase two of timestep begins. Once the diffusion federate receives `modelToDiffuser` interactions from all model region federates, it executes the diffusion algorithm until steady state is achieved. The diffusion federate then sends an interaction containing the new substrate and signalling molecule concentrations to each model region federate. In parallel, each model region federate calculates the number of particles to transfer between local and remote voxels using the particle counts computed during phase one. It then determines which particles are to be transferred to voxels managed by neighbouring model region federates and sends an interaction to each neighbouring model region federate containing the state(s) of the transferred particle(s). The diffusion federate (diffusion algorithm) and the model region federates (pressure calculations and particle transfer) execute concurrently in phase two.

Each exchange of information (via an interaction) has an associated timestamp. All interactions generated in phase one have timestamp $t + 1$. Once a model region federate has sent particle counts to its neighbouring model region federates it makes an HLA time advance request to time $t + 1$. The diffusion federate also makes a time advance request to $t + 1$ at the beginning of the model timestep.[3] All interactions generated in phase two have a timestamp of $t + 2$. Once the diffusion federate has sent an interaction containing the updated substrate and signalling molecule concentrations for the next model timestep to each model region federate, it makes an HLA time advance request to time $t + 2$. Similarly each model region federate makes a time advance request to $t + 2$ once it has sent an interaction to each neighbouring model region federate containing the particles to be transferred at this model timestep. When time advances to $t{+}2$ each model region federate updates their boundary voxels with any particles transferred from neighbouring model regions. This completes processing at this model timestep, and all federates then advance to the next model timestep. A model timestep therefore corresponds to two HLA timesteps.

6 Results

In this section we present preliminary results from experiments designed to determine the overhead of the HLA/RTI in the BACGRID federation and at what point (if at all) distribution of the model results in lower elapsed times.

The biofilm model used in the experiments consisted of a single biomass type, a single substrate, and two types of signalling molecule. The parameter values for the bacterial growth and division models were taken from [8] and the parameters for the inter-cellular signalling and up-regulation model from [6]. The experiments used a $17 \times 17 \times 17$ micron voxel as in [11], and a maximum cell radius of 0.756 microns, which allows up to 1371 cells per voxel. The maximum particle radius was five times larger than the maximum cell radius, with the result that a particle of maximum size contains 125 cells. To obtain a reasonable approximation to steady state CPU requirements, the experiments were initialised with all voxels containing 50% of their maximum particle capacity and each particle at half its maximum volume. The total size of the computational domain was $4 \times 8 \times 100$ voxels, or $68 \times 136 \times 1700$ microns. The model timestep was 0.1 seconds (determined by diffusion requirements), and the simulations ran for 2 hours of simulated time or 72,000 timesteps. Over the whole run, the maximum number of particles in the entire model was approximately 35000, representing approximately 4.4 million bacterial cells. This model size was selected as being the largest that can be effectively simulated on a processor with 1GB of memory without excessive swapping (i.e., it is memory bound). However it is still fairly small in biological terms.

We developed a simple BACGRID federation, consisting of a single diffusion federate, and variable numbers of model region federates. The model was divided into 1, 2 and 4 model regions, by repeatedly splitting the computational domain along the y axis. So in configuration *1MR*, there is a single model region federate which manages a model region $4 \times 8 \times 100$ voxels in size; in configuration *2MR* there are two model region federates each of which manages a model region $4 \times 4 \times 100$ voxels in size, and in the the *4MR* configuration there are four model region federates each of which manages a model region $4 \times 2 \times 100$ voxels in size. All configurations also included a single diffusion federate. Each federate and the RTI executive were executed on a separate compute node.

The hardware platform used for our experiments was a small Linux cluster comprising 2.4GHz Intel Pentium 4 processors, each with 512 KB cache and 1GB of RAM, interconnected by a standard 1Gbps Ethernet switch. The nodes were otherwise unloaded during the experiments. For each model region configuration, we report the total (user) CPU time for diffusion and model region federates (in seconds), the elapsed time required for the simulation (in seconds), and the ratio of elapsed to virtual time (i.e., the elapsed time necessary to advance simulated time by one second).

Figure 5 shows the total elapsed time, the average CPU

[3]The diffusion federate remains blocked until the model region federates request time advance to $t + 1$.

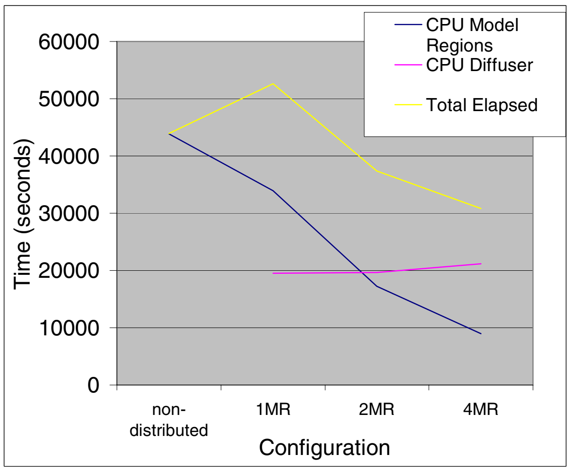

Figure 5: CPU and elapsed times each configuration

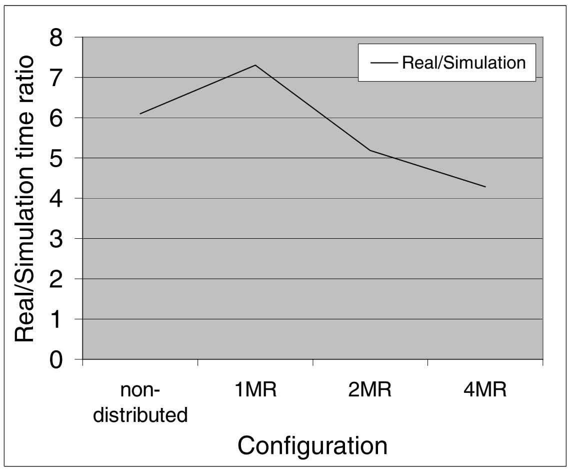

Figure 6: Elapsed vs. simulation time for each configuration

time for all model region federates and the CPU time for the diffusion federate for each configuration. The total elapsed and CPU times for the non-distributed version of BACGRID are also shown for comparison.

In the non distributed case the CPU time is 43865 seconds, and accounts for 99.9% of the elapsed time. In the *1MR* configuration, the computation is distributed between the model region and the diffusion federates. However while the CPU for each federate is less than that in the non-distributed case, total elapsed time required for the simulation increases. This is because the communication (RTI) overhead introduced by distribution more than offsets the reduction in CPU time which results from splitting the model region and diffusion computations. Moreover, the total elapsed time for the simulation of 52590 seconds is only slightly less than the combined CPU time of 53503 seconds for the diffusion federate (19543 seconds) and the model region federate (33960 seconds). This is perhaps not surprising, as most of the computation done at the model region (i.e., growth) cannot occur in parallel with diffusion.

In the two model region federate (*2MR*) case we begin to see speedup, with the total elapsed time dropping to 37351 seconds. Again the total elapsed time corresponds closely to combined CPU time for a single model region federate and the diffusion federate. The reduction in total elapsed time seen in this case is due to the growth, division etc. of particles in the two model regions being executed in parallel. Finally, in the *4MR* case the total elapsed time drops even further to 30861 seconds, as the particle computation is now distributed across four model region federates.

Figure 6 shows the ratio of elapsed to simulation time (i.e., the elapsed time necessary to advance one unit of simulation time) for each configuration. In the non-distributed case, the elapsed/simulation time ratio is approximately 6.1, i.e., it takes approximately 6.1 hours of elapsed time to advance one hour of simulated time. The elapsed/simulation time ratio increases to 7.3 in the *1MR* case before declining to 5.2 in the *2MR* case and 4.25 in the *4MR* case, i.e., a speedup of 1.44.

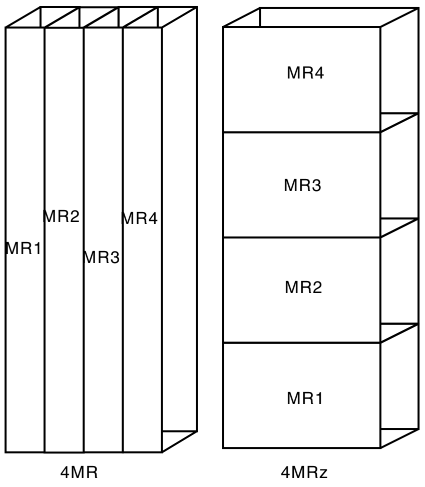

Figure 7: Alternative arrangements of four model regions (Not to scale)

The results above are for a particular decomposition of the computational domain. To investigate the effect of alternative assignments of voxels to model regions, we repeated the *4MR* experiment, this time splitting the computational domain along the z axis rather than the y axis. In this case (*4MRz*) each model region is $4 \times 8 \times 25$ voxels as opposed to $4 \times 2 \times 100$ voxels in the *4MR* case. Figure 7 illustrates the alternative decompositions.

The differences between the two decompositions lies in the amount and type of communication that occurs between model regions. In the *4MRz* case, there are 32 proxy voxels for each connected face (a connected face is a side of the model region which connects to another model region) whereas in the *4MR* case there are 100 proxy voxels for each connected face. In addition, in the *4MRz* case there are fewer connected faces than in the *4MR* case (6 vs. 8) as the x and y axes are both periodic, whereas the z axis is not. However in the *4MR* case, we would expect fewer particles to be transferred between model regions, as most of the particle movement occurs vertically, along the pressure gradient. Transferring one or more particles implies a change in particle count, however particle counts may change without any particles being transferred.[4] The total elapsed time for the *4MRz* case is 32743 seconds, giving an elapsed to simulation time ratio of 4.55. This is slightly higher than in the *4MR* case, suggesting that, at least for this computational domain, the increase in particle transfers in the *4MRz* case more than offsets the smaller number of proxy voxels whose particle counts may potentially have to be updated.

In future work, we plan to further investigate alternative decompositions of the computational domain to better understand the relationship between model evolution, allocation of voxels to model regions and elapsed time.

7 Summary

We have presented BACGRID, a prototype HLA-compliant, Grid-based simulator for agent-based bacterial models. We sketched the distribution of BACGRID using HLA and and presented preliminary results of experiments to investigate the scalability of the BACGRID HLA distribution. Our results indicate the feasibility of using HLA for systems biology simulations, and suggest that, at least for memory bound simulations, the overhead of the HLA/RTI is small enough that speedup is possible.

The BACGRID prototype adopts a spatial decomposition in which federates implement model regions. While relatively simple, it creates a starting point for the integration of other simulators and models, e.g., at the cellular level through the inclusion of different kinds of bacteria, and at the sub-cellular level through the inclusion of models of gene expression and signalling. For example, different types of bacteria, or sub-cellular (gene and signalling networks) and cell-level (growth, division and displacement) phenomena could be implemented by different federates.

Acknowledgements

This work was supported by BBSRC project number BB/D006619/1 and by EPSRC projects EP/C549406/1 and EP/C549414/1.

[4] To minimise communication overhead, `particleCount` interactions are only sent when the number of particles in a voxel changes.

References

[1] H. M. Byrne, J. R. King, D. L. S. McElwain, and L. Preziosi. A two-phase model of solid tumour growth. *Applied Mathematics Letters*, 16(4):567–573, 2003.

[2] I. Chang, E. S. Gilbert, N. Eliashberg, and J. D. Keasling. A three-dimensional, stochastic simulation of biofilm growth and transport-related factors that affect structure. *Microbiology*, 149:2859–2871, 2003.

[3] M. Ginovart, D. Lopez, and J. Valls. INDISIM, an individual-based discrete simulation model to study bacterial cultures. *Journal of Theoretical Biology*, 214:305–319, 2002.

[4] IEEE Standard for modeling and simulation (M&S) High Level Architecture (HLA) — Framework and rules. IEEE, 2000. (IEEE Standard No.: 1516-2000).

[5] J. King, M. Lees, and B. Logan. Agent-based and continuum modelling of populations of cells. Technical report, University of Nottingham, December 2006.

[6] A. J. Koerber, J. R. King, J. P. Ward, P. Williams, J. M. Croft, and R. E. Sockett. A mathematical model of partial-thickness burn-wound infection by *Pseudomonas aeruginosa*: Quorum sensing and the build-up to invasion. *Bulletin of Mathematical Biology*, 64:239–259, 2002.

[7] K. Krawczyk, W. Dzwinel, and D. A.Yuen. Nonlinear development of bacterial colony modeled with cellular automata and agent objects. *International Journal of Modern Physics C*, 14(10):1385–1404, 2003.

[8] J.-U. Kreft, G. Booth, and J. W. T. Wimpenny. BacSim, a simulator for individual-based modelling of bacterial colony growth. *Microbiology*, 144:3275–3287, 1998.

[9] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. MASON: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[10] R. Paton, R. Gregory, C. Vlachos, J. Saunders, and H. Wu. Evolvable social agents for bacterial systems modeling. *IEEE Transactions on Nanobioscience*, 3(3):208–216, September 2004.

[11] C. Picioreanu, J.-U. Kreft, and M. C. M. van Loosdrecht. Particle-based multidimensional multispecies biofilm model. *Applied and Environmental Microbiology*, 70(5):3024–3040, May 2004.

[12] J. M. Pullen, R. Brunton, D. Brutzman, D. Drake, M. Hieb, K. Morse, and A. Tolk. Using Web services to integrate heterogeneous simulations in a Grid environment. *Future Generation Computer Systems*, 21:97–106, 2005.

[13] J. P. Ward, J. R. King, A. J. Koerber, J. M. Croft, R. E. Sockett, and P. Williams. Early development and quorum sensing in bacterial biofilms. *Journal of Mathematical Biology*, (47):23–55, 2003.

[14] K. Winzer, K. R. Hardie, and P. Williams. Bacterial cell-to-cell communication: sorry, can't talk now—gone to lunch! *Current Opinion in Microbiology*, 5(2):216–222, April 2002.

[15] Y. Xie, Y. M. Teo, W. Cai, and S. Turner. Service provisioning for HLA-based distributed simulation on the Grid. In *Proceedings of the Nineteenth ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2005)*, pages 282–291, Monterey, CA, USA, June 2005.

[16] Y. Zhang, G. Theodoropoulos, R. Minson, S. Turner, W. Cai, Y. Xie, and B. Logan. Grid-aware large scale distributed simulation of agent-based systems. In *Proceedings of the 2005 European Simulation Interoperability Workshop*, Toulouse, June 2005. Simulation Interoperability Standards Organisation, IEEE/ITCMS. 05E-SIW-047.