Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

PARALLEL DISCRETE-EVENT SIMULATION OF POPULATION DYNAMICS

Bhakti Satyabudhi Stephan Onggo

Department of Management Science
Lancaster University Management School
Lancaster University
Lancaster, LA1 4YX, United Kingdom

ABSTRACT

Research in parallel simulation has been around for more than two decades. However, the number of papers reporting on its application to real world problems is limited. At the 2002 PADS conference, researchers discussed the need to go beyond synchronization and performance issues, and, in particular, to demonstrate that parallel simulation could be used in real world applications outside military and network simulations. Since then, we have seen an increase in the number of papers on the parallel simulation applications in areas such as operations management and the physical sciences. This paper advocates the application of parallel simulation in population dynamics which is often used as the basis for policy planning and analysis. In this paper, we show how the simulation model is implemented using the μsik parallel simulation library. We conduct some experiments to measure the simulation performance. The result shows that good event parallelism can be achieved.

1 INTRODUCTION

The advent of parallel and distributed computing technologies has made parallel discrete-event simulation (PDES) possible. Parallelization can be achieved through partitioning a simulation model into a set of smaller components called *logical processes* (LPs) and running the LPs concurrently. PDES research in the last two decades has resulted in a number of synchronization protocols. Most of these protocols can be grouped into two categories: *conservative* and *optimistic*. This classification is based on how the *local causality constraint* (lcc) is maintained. Lcc imposes that if event `a` happens before event `b` and both events happen in the same LP, then event `a` must be executed before event `b`. Parallel simulation must adhere to lcc to produce correct simulation results. Conservative protocols do not allow any lcc violation throughout the duration of the simulation. Optimistic protocols allow lcc violation, but provide mechanisms to rectify it.

One of the frequently used conservative simulation protocols is the CMB protocol (Chandy and Misra 1979, Bryant 1984). This protocol uses a dummy message called a *null message*. When an LP sends a null message with a timestamp `t`, it tells the receiving LP that it will not send any message with a timestamp earlier than `t`. The timestamp `t` depends on what is called the *lookahead*. It represents the minimal amount of physical time that is required to complete a process in the real world. The LPs use this information to advance their local simulation time. Hence, the performance of the CMB protocol depends on the lookahead quality.

The Time Warp protocol (Jefferson 1985) is the most widely used optimistic protocol. In this protocol, once an LP detects an lcc violation, it rolls back to the last correct state and restarts the simulation from that point. Traditionally, the roll back process is made possible by using a state-saving mechanism. A more recent method includes the *reverse computation* method where the correct state is recovered by reversing the computation process (Carothers et al. 1999). Fujimoto (2000) provides a good overview on different parallel simulation protocols.

In the past two decades, PDES research has been dominated by the development of better synchronization algorithms and their performance evaluations. Researchers at the 2002 Workshop on Parallel and Distributed Simulation (PADS) in Washington, D.C. discussed "life beyond synchronization". It was felt that it was high time for researchers to promote the use of parallel simulation in wider application areas. Traditionally, parallel simulation has been applied in military and network simulations. Since then, we have seen an increase in the number papers reporting on parallel simulation applications outside traditional areas. Tang et al. (2005) demonstrated an initial study in applying parallel simulation to a plasma physics application. In the realm of biological science, Lobb et al. (2005) applied parallel simulation to a neuron model. Lan and Pidd (2005) applied parallel simulation to simulate a quasi-continuous manufacturing process. The situation is encouraging, although it is still far from ideal, where parallel simulation is widely used across many different applica-

tion areas including in the social sciences. This motivates us to study the feasibility of using parallel simulation in demography. There is general interest in using larger sample sizes in demographic simulation models.

Demography is significant in the social sciences because it is often used as the basis for government planning in areas such as education, healthcare, social welfare and taxation. For example, a fall in the number of births may lead to school closures. Similarly, a rise in the number of elderly people may lead to an increase in the demand for healthcare services related for the elderly.

This paper reports on what we have done so far. First, we will show the development of a parallel discrete-event simulation of population dynamics using a parallel simulation library called μsik. Second, we will report the simulation performance. The rest of this paper is organized as follows. Section 2 presents an overview of the demographic simulation model that is used in this paper. Section 3 shows the implementation of the simulation model using μsik parallel simulation library. Section 4 summarizes the validation of the simulation model. We show the performance of the parallel simulation implementation in section 5. Section 6 presents our concluding remarks and highlights further work.

2 DEMOGRAPHIC MODELS

Demography is the study of human population in relation to changes brought about by the interplay of births, deaths and migration (Pressat 1985). Population projection is one of the main applications of demography. Traditionally, there are two methods commonly used in population projections: mathematical model and cohort component (Hinde 1998). In the first method, demographers use a set of mathematical equations to project the *size* of a future population. The cohort component method is used to project the *structure* of a population, i.e., the size of different groups within the population (such as different age groups). This is an iterative method where each iteration projects the structure of a population in the following year using a set of flow models. Hence, the latter method is able to produce a more detailed projection. For many purposes, the population structure is more useful than the total population size. Recently, simulation has also been used in demographic analysis. Simulation allows individual-specific explanatory variables to be included in the model. For example, we may include factors such as age, education level, salary group and ethnicity to model the number of children that an individual female will have. To take one example, Liddle (2002) developed a simulation model to assess the effect of demographic dynamics on sustainable development at three levels: economic, social and environmental.

Demographic models are often used in a number of simulation models for policy analysis and planning. To take two examples, Walker et al. (2000) used simulation to analyze the effects of demographic changes on government expenditure on pharmaceutical benefits in Australia. Similarly, Bonnet and Mahieu (2000) analyzed pension policy in France using simulation. Their simulation model comprised three main components: demographic, labour market and income.

In the following subsections, we briefly explain a number of models commonly used to model each demographic element. All models are taken from Hinde (1998). We also explain the models used in this paper. This paper focuses more on the implementation and the performance aspect of parallel simulation. Hence, at this stage, we prefer to use simpler models for the demographic elements. More complexity can be added to these models in later stages of this research.

2.1 Mortality

The most commonly used method in mortality analysis is the life table (and its variants). The life table summarizes the variation of mortality by age. In this paper, we use another common method called survival analysis. In this method, a distribution function of lifetime is used to sample an individual's lifetime at birth. Figure 1 shows the distribution function used in this paper.

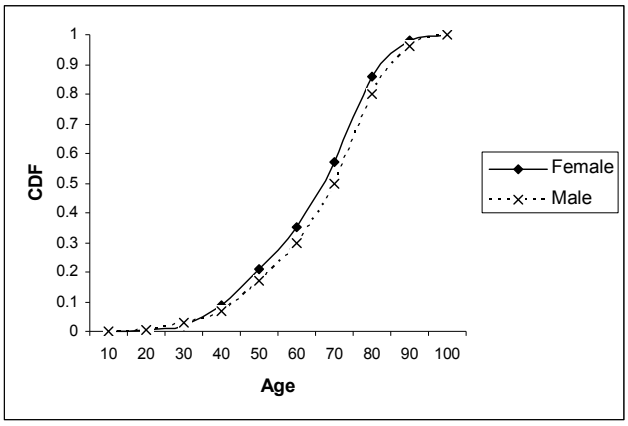

Figure 1: Distribution function of lifetime

2.2 Work Status

We plan to use our simulation infrastructure for public policy planning and analysis (such as: taxation, pensions and benefits). Therefore, it is essential that the model is able to track an individual's work status. In the model used in this paper, an individual spends the first 16 years of his/her life in a dependent status (or earlier, due to an early death). After reaching the end of the dependent age period, the individual can be in one of three states: working, in full-time education, or unemployed, based on a simple random sampling process. To simplify the model further, we sample

the length of education once. Hence, it assumes that individuals who follow this path will be in full-time education immediately after the dependent age period and s/he will never be in full-time education again for the rest of his/her life. After leaving the full-time education state, an individual is either employed or unemployed. Once an individual is in the employment or unemployment state, s/he can switch between these states many times during her/his life based on a simple random sampling process until s/he reaches pension age. The time spent at each stage is sampled based on a uniform distribution. Figure 2 shows the state-transition diagram for the work status in our model. Two states: unborn and dead are not shown but are used in the model.

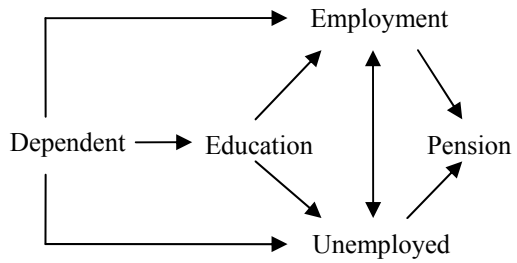

Figure 2: State diagram of work status

2.3 Marital Status

Another essential requirement in the policy planning and analysis is for the model to track an individual's marital status. Figure 3 shows that during his/her life, an individual can be in one of these marital statuses: single (never married), married, cohabitating, divorced, separated or widowed. The time spent in each status is sampled from a uniform distribution, except for married and single (some marriages will last until one partner dies and some individuals may choose to remain single). Again, the unborn and dead states are not shown in the figure but are used in the model.

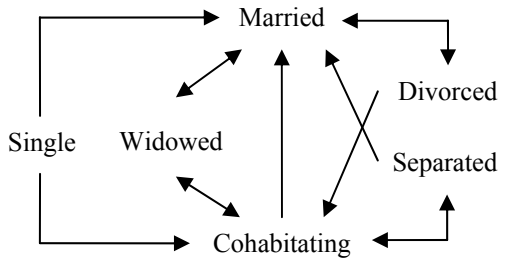

Figure 3: State diagram of marital status

Demographers consider the pattern analysis of marriage and cohabitation to be important. One of the reasons is that after age, the most important factor that determines the fertility rate of female individuals is marital status. In demographic simulation, two marriage models are commonly used: open and closed. An open marriage model is simpler because it only changes the status of the individual. The closed marriage model, which is used in our simulation, is more complicated. In this model, the simulation can only schedule the start of a 'partner search' rather than a marriage per se. If a suitable individual is found from the list of prospective partners, then the marriage event will be scheduled for both individuals. Otherwise, the individual will be added to the list of prospective partners. The same algorithm applies for cohabiting individuals.

2.4 Fertility

Common models for fertility analysis include age-specific fertility, parity-specific fertility and birth spacing. Age-specific fertility uses age to determine the probability of having a child. Parity-specific fertility takes into account the number of children a female individual has already had. The birth spacing model focuses more on the time between each birth. It is rather complicated because it includes factors such as birth control, abstinence period, employment, etc. In this paper, we use a combination of simple birth spacing model and simple parity-specific fertility. Once a female individual has entered a marriage or cohabitation stage, the number of births is sampled from a uniform distribution based on the duration of marriage (or cohabitation) and the end of her reproductive stage, whichever is earlier.

2.5 Migration

The effect of migration has become more significant in recent years. Unlike births and deaths, an individual may migrate (either domestically or internationally) more than once. Furthermore, migration depends on a complex combination of individual-specific factors, economic factors, political factors, and many more. Currently, our simulation model considers domestic migration only. It uses a simple random sampling to decide whether a working individual or an individual going into full-time education is going to migrate to another local authority. The model assumes that only an individual who are not married or cohabiting will migrate. In principle, international migration can be modelled in a similar way.

2.6 Justification for Parallel Simulation

Social scientists, such as: Liddle (2002), Walker et al. (2000), and Bonnet and Mahieu (2000), use the term microsimulation for their simulation models. This type of simulation inspects each individual at each simulation time point, in which one or more random sampling processes

are performed to determine the state of each individual at the next simulation time point. At one extreme, the sampling process requires simple random sampling. At another extreme, it may require a complex regression model. Many simulation textbooks refer to this simulation as the timestep simulation or simulation with a fixed increment time advancement algorithm (e.g., Pidd 2004). This type of simulation may be relatively slow if the model includes a large number of individuals and the simulation time unit is very small. If the state changes do not occur very frequently, discrete-event simulation should be faster because it inspects an individual only when his/her state changes. It should be noted that the simulation execution time depends not only on the processor speed but also the memory capacity (from cache to virtual memory). Hence, if the model includes a large number of individuals, it is possible that sequential simulation might be too slow. Parallel simulation may offer an alternative.

3 SIMULATION IMPLEMENTATION

We implemented the simulation model using μsik parallel simulation library (Perumalla 2005). This library supports multiple synchronization algorithms such as: lookahead-based conservative protocol and rollback-based optimistic protocol (state-saving and reverse-computation). It has been reported to be scalable on a large number of processors (Perumalla 2007). This library adopts the process interaction world-view; hence the simulation model is formed by a set of interacting (logical) processes. *Logical processes* (LPs) communicate through events. Multiple LPs are mapped onto a *physical process* (PP) that is run on top of a *processing element* (PE). A machine can have more than one PE (i.e., multi-core architecture). It should be noted that in μsik documentation, PP is often referred to as federate. To avoid confusion with the federate in High Level Architecture, in this paper, the term PP is used instead of federate.

To implement a simulation model in μsik parallel simulation library, we must specify three main components: a physical process (must inherit from class `Simulator`), a set of logical processes (each must inherit from one of these classes: `NormalSimProcess`, `PeriodicSimProcess`, or `ThreadedSimProcess`), and a set of events (each must inherit from class `SimEvent`). The detail explanation on the structure of a simulation model written in μsik can be found from <http://www.cc.gatech.edu/~kalyan/musik.htm>. The following sections explain how the demographic simulation model described in section 2 is implemented using μsik.

3.1 Physical Process - Population Simulator

The physical process, implemented as class `PopulationSimulator`, is defined as a subclass of class `Simulator`. In this class, we use a variable called `endtime` to specify the simulation stopping condition. The number of local authorities mapped onto a physical process is stored in `nLocalAuth`. The variables for statistical counters, such as average age and average time spent in marriage, are not shown. Any class that inherits class `Simulator` must implement methods `init` and `run`. They are used to initialize the simulation (such as setting up simulation parameters and distributing LPs across PEs) and to run the simulation, respectively. The method `Write` is used to produce a detailed report (in CSV format). This report is very useful for model validation or post-analysis using other software (such as Excel and SAS). Finally, the method `GetNewActiveIndividual` is used to add a new individual to a physical process (due to events such as: migration and birth).

```
class PopulationSimulator : public Simulator
{
public:
  SimTime endtime;
  int nLocalAuth;

  // other variables are not shown

  PopulationSimulator(void);
  virtual ~PopulationSimulator(void);

  virtual void init(int ac, char *av[]);
  virtual void run(void);
  void Write(const PersonalData& p_data,
       const PersonalStat& p_stat);
  long GetNewActiveIndividual(void);
};
```

Each PP in μsik hosts a number of LPs. In our demographic simulation, two types of LPs are used, namely: local authority and individual. A local authority represents an administrative area where a number of individuals live. Each individual process represents a person in the population. Local authority and individual processes are distributed evenly across PEs. The specifications for class Individual and class Local Authority are shown in the following two subsections.

3.2 Logical Process – Individual

The personal information of each individual (such as ID, gender, work status and marital status) is stored in the variable called `data`. The statistics for each individual (such as time spent in each work status or marital status) is stored

in the variable called `stat`. The next couple of lines show the list of event handlers (such as change in the marital status). Note the naming convention: `Birth` is the handler for event `BirthEvent`. These event handlers are followed by their anti-event handlers that will be used to reverse the computation when a rollback is required. The first four event handlers are self-explanatory. The next three event handlers are used for the migration of individuals. An individual who is going to migrate to another local authority (in another PE) will send an event (`RequestPIDEvent`) to the destination local authority's process asking for a new process ID (PID). The destination local authority will send the new PID through event `TransferPIDEvent`. Finally, the individual's process will send its personal data and statistics (through `MigrDataEvent` and event `MigrStatEvent`, respectively) to the new process. The data and statistics are not sent in one event because their combined size exceeds the maximum limit on event size in μsik.

All LPs in μsik must implement methods `init`, `execute`, `undo_event` and `wrapup`. Method `init` is used to initialize the individual (such as setting-up gender and time of death). Methods `execute` and `undo_event` are used to execute the event handlers and anti-event handlers, respectively. The last method, `wrapup`, is used when the process is deleted.

```
class Individual : public NormalSimProcess
{
private:
  PersonalData data;
  PersonalStat stat;

  // other variables are not shown

  void ChangeWorkStatus(
       ChangeWorkStatusEvent* event);
  void ChangeMaritalStatus(
       ChangeMaritalStatusEvent* event);
  void Death(DeathEvent* event);
  void Birth(BirthEvent* event);
  void MigrData(MigrDataEvent* event);
  void MigrStat(MigrStatEvent* event);
  void TransferPID(TransferPIDEvent* event);

  void UndoChangeWorkStatus(
       ChangeWorkStatusEvent* event);
  void UndoChangeMaritalStatus(
       ChangeMaritalStatusEvent* event);
  void UndoDeath(DeathEvent* event);
  void UndoBirth(BirthEvent* event);
  void UndoMigrData(MigrDataEvent* event);
  void UndoMigrStat(MigrStatEvent* event);
  void UndoTransferPID(
       TransferPIDEvent* event);
  // supporting methods are not shown

protected:
  virtual void undo_event(SimEventBase *e);
  PopulationSimulator *psim();

public:
  Individual(const long p_id,
     const int p_auth,
     const bool optimistic);
  virtual ~Individual(void);

  virtual void init(void) {}
  virtual void wrapup(void);
  virtual void execute(SimEvent *event);
};
```

3.3 Logical Process - Local Authority

The structure of class `LocalAuthority` and class `Individual` is the same. There are three event handlers in the local authority's process. An individual who is going to migrate to this local authority will send a `RequestPIDEvent`. This event will instruct the physical process to allocate a new process ID for the individual. An individual who is going to marry (or be cohabiting) will send a `SeekPartnerEvent`. When a partner can be found from the list of prospective partners, this will initiate event `ChangeMaritalStatusEvent` and the partner will be removed from the list. Otherwise, the individual will be added to the list of prospective partners. When an individual who is on the list dies or migrates to another local authority, an event `CancelSeekPartnerEvent` is sent to remove the individual from the list. Finally, when a female individual is scheduled to give birth, the individual's process will send an `InitBirthEvent`. This event will instruct the physical process to allocate a new process ID for the baby and schedule a `BirthEvent`.

```
class LocalAuthority:public NormalSimProcess
{
private:
  vector <PartnerData> male;  // partner list
  vector <PartnerData> female;// partner list

  // other variables are not shown

  void SeekPartner(SeekPartnerEvent* event);
  void CancelSeekPartner(
       CancelSeekPartnerEvent* event);
  void InitBirth(InitBirthEvent* event);
  void RequestPID(RequestPIDEvent* event);

  void UndoSeekPartner(
       SeekPartnerEvent* event);
  void UndoCancelSeekPartner(
```

```
        CancelSeekPartnerEvent* event);
  void UndoInitBirth(InitBirthEvent* event);
  void UndoRequestPID(
        RequestPIDEvent* event);

  // supporting methods are not shown

protected:
  virtual void undo_event(SimEventBase *e);
  PopulationSimulator *psim();

public:
  LocalAuthority (const int p_id,
     const bool optimistic);
  virtual ~LocalAuthority (void);

  virtual void init(void);
  virtual void wrapup(void);
  virtual void execute(SimEvent *event);
};
```

3.4 Events

In this simulation model, we use the following events:

- `ChangeWorkStatusEvent`: an event that changes an individual's work status
- `ChangeMaritalStatusEvent`: an event that changes an individual's marital status
- `SeekPartnerEvent`: an event that indicates that an individual is actively seeking a spouse/partner
- `CancelSeekPartnerEvent`: an event that is used to remove an individual from the list of prospective partners when s/he dies or migrates to another local authority
- `BirthEvent`: an event that creates a new individual
- `InitBirthEvent`, an event that initializes necessary system settings before `BirthEvent` is executed
- `DeathEvent`: the death of an individual
- `RequestPIDEvent`, `TransferPIDEvent`, `MigrDataEvent`, and `MigrStatEvent`: are used to implement migration

4 MODEL VALIDATION

The main purpose of this paper is to demonstrate how parallel simulation can be applied to population dynamics. Hence, at this stage, validation against real demographic data has not been performed. Instead, we focus more on the correctness of the basic mechanism of each component in the demographic model, such as:

- The lifetime for each individual should follow the distribution function given in Figure 1 closely
- The proportion of male and female individuals should reflect the parameters given in the model
- For all individuals, the total time spent in all periods of work status should be the same as their lifetime and consistent with the state diagram given in Figure 2
- For all individuals, the total time spent in all periods of marital status should be the same as their lifetime and consistent with the state diagram given in Figure 3
- The number of migrations should reflect the proportions given in the model
- The reported birth rate per female should be reflected in the population growth (i.e., below the replacement rate, close to the replacement rate, and above the replacement rate)

5 PERFORMANCE ANALYSIS

We conducted two types of experiments to study simulation performance. We concentrate on a performance metric called *event parallelism*, i.e., the number of (useful) events that are executed per second. The simulation mode was set to the optimistic execution using the default μsik setting (state saving, infinite run ahead and zero resilience). This setting gives the traditional rollback-based optimistic execution with copy state saving mechanism. The program was compiled using gcc version 3.3.5 with the optimization flag O3 turned on. We ran all experiments on a cluster of Sun Fire X4100 servers connected via a gigabit Ethernet switch. Each node has two dual-core 2.4GHz Opteron CPUs and 8GB of memory.

The objective of the first experiment was to study the effect of varying the number of processors and the inter-processor communications on the event parallelism for a fixed population size (*strong scaling*). In our simulation, the inter-processor communications are caused by the migrations. Thus, an increase in the proportion of individuals who are going to migrate increases the number of inter-processor communications. In the experiment, we fixed the initial population size at 4,096 individuals and distributed them equally across eight local authorities. The simulation run length was fixed at 200 years. The fertility rate was set to be close to the replacement rate. The result is shown in Figure 4.

The result shows that, for the same total number of individuals, an increase in the number of processors increases computing power, but at the same time more synchronization overheads are required. This explains the diminishing performance gain for four and eight processors. Hence, it is likely that a further increase in the number of processors will eventually decrease the parallelism exploited. For the same number of processors, an increase in the proportion of individuals who are going to migrate, from 10% to 50%, reduces the exploited event parallelism.

This is due to the increase in the number of inter-processor communications. This also explains why the performance gain for four and eight processors diminishes faster for the higher proportion of migrations.

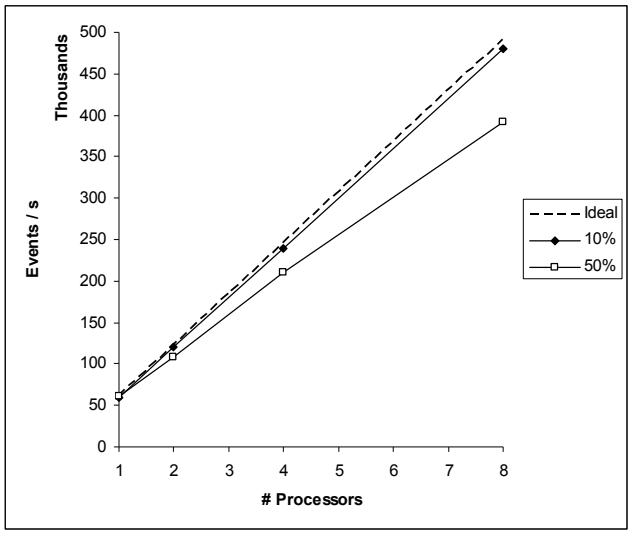

Figure 4: Event parallelism – strong scaling

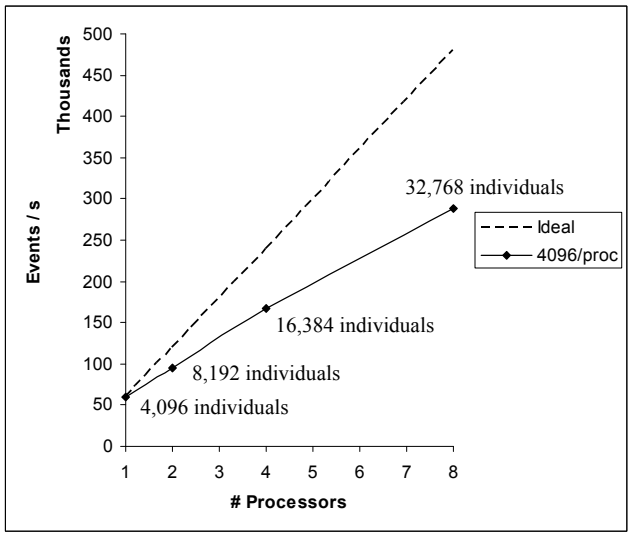

Figure 5: Event parallelism – weak scaling

In the second experiment, we wanted to study the speed-up as the number of processors and the population size are increased in proportion (*weak scaling*). In this experiment, we ran the simulation with an initial population size of 4,096 individuals on one processor and increased the number of individuals and processors proportionally up to 32,768 individuals running on eight processors. Individuals were distributed equally across eight local authorities. The proportion of individuals who are going to migrate was fixed at 10%. Similar to the first experiment, the simulation run length was fixed at 200 years and the fertility rate was set to be close to the replacement rate. An increase in the number of processors increases the computing power, while an increase in the number of individuals increases the number of migrations (hence, inter-processor communications). Thus, in this experiment, we wanted to see whether the increase in computing power can cope with the increase in the communication workload. The result is shown in Figure 5.

It shows that the increase in computing power can cope reasonably well with the increase in inter-processor communications. However, the performance gain diminishes as we increase the number of processors and individuals. Hence, if we increase them further, the exploited parallelism will eventually decrease.

6 CONCLUSION AND FUTURE WORK

This paper advocates the application of parallel simulation in Demography. Although this research is still at an early stage, we are able to show that it is possible to implement a parallel simulation for population dynamics. The availability of a parallel simulation library, such as: μsik, helps cut down the software development time. Furthermore, it frees modellers from the detailed synchronization mechanisms (although some knowledge on the synchronization mechanisms is very useful in analyzing the performance as well as during the debugging process). It is hoped that this paper could promote the application of parallel simulation in the social sciences. The experimental results also show that the parallel simulation implementation exhibits good event parallelism and reasonable scalability.

We plan to use more accurate demographic models in the simulation and investigate the performance of parallel simulation further. The use of more accurate demographic models will improve the applicability of our research work to policy analysis and planning. It should be noted that a more accurate model will affect simulation performance. For example, if the model includes the migration of married individuals, when those individuals die, events must be sent to the spouses (who may live in different local authorities). When a more complete and accurate model is ready, we will need to carry out thorough performance analyses to understand the parallelism, the scalability, and the memory requirements of the parallel simulation.

ACKNOWLEDGMENTS

This research is funded by the Lancaster University small grant SGS/14/17 (2008).

REFERENCES

Bonnet, C., and R. Mahieu. 2000. Public pensions in a dynamic microanalytic framework: the case of France. In *Microsimulation modelling for policy analysis: chal-*

lenges and innovations, ed. L. Mitton, H. Shuterland, and M. Weeks, 175–199. Cambridge, UK: Cambridge University Press.

Bryant, R.E. 1984. A Switch-Level Model and Simulator for MOS Digital Systems. *IEEE Transaction on Computers* 33 (2): 160-177.

Carothers, C., K.S. Perumalla, and R.M. Fujimoto. 1999. Efficient optimistic parallel simulations using reverse computation. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, 126-135.

Chandy, K.M., and J. Misra. 1979. Distributed Simulation: a Case Study in Design and Verification of Distributed Programs. *IEEE Transaction on Software Engineering* 5 (5): 440-452.

Fujimoto, R.M. 2000. *Parallel and distributed simulation systems*. John Wiley & Sons.

Hinde, A. 1998. *Demographic methods*. London, UK: Arnold.

Jefferson, D.A. 1985. Virtual Time. *ACM Transaction on Programming Language System* 7 (3): 404-425.

Lan, C. and M. Pidd. 2005. High performance simulation in quasi-continuous manufacturing plants. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 1367–1372. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Liddle, B. 2002. Demographic dynamics and sustainability: insights from an integrated multi-country simulation model. Working Paper WP 2002-039, Max Planck Institute for Demographic Research, Rostock, Germany.

Lobb, C.J., Z. Chao, R.M. Fujimoto, and S.M. Potter. 2005 Parallel event-driven neural network simulations using the Hodgkin-Huxley neuron model. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, 16-25.

Perumalla, K.S. 2005. μsik – a micro-kernel for parallel/distributed simulation systems. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, 59-68.

Perumalla, K.S. 2007. Scaling time warp-based discrete-event execution to 104 processors on a blue gene supercomputer. In *Proceedings of the 4th International Conference on Computing Frontiers*, 69-76.

Pidd, M. 2004. *Computer simulation in management science*. 5th ed. Chichester, England: John Wiley & Sons.

Pressat, R. 1985. *Demographic analysis*. Chicago: Aldine Atherton.

Tang, Y., K.S. Perumalla, R.M. Fujimoto, H. Karimabadi, J. Driscoll, and Y. Omelchenko. 2005. Optimistic parallel discrete event simulations of physical systems using reverse computation, In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, 26-35.

Walker, A., R. Percival, and A. Harding. 2000. The impact of demographic and other changes on expenditure on pharmaceutical benefits in 2020 in Australia. In *Microsimulation modelling for policy analysis: challenges and innovations*, ed. L. Mitton, H. Shuterland, and M. Weeks, 149–171. Cambridge, UK: Cambridge University Press.

AUTHOR BIOGRAPHY

BHAKTI S. S. ONGGO is a research associate at the Department of Management Science, Lancaster University Management School, Lancaster University, UK. He received his MSc in Management Science from the Lancaster University and completed his PhD in Computer Science from the National University of Singapore. His research interests are in the areas of simulation methodology, simulation technology (including parallel and distributed simulation), and simulation applications. He is a member of the ACM and the Operational Research Society. Further information can be found from <http://www.lancs.ac.uk/staff/onggo>