Parallel Implementation of Large Scale Agent-based Models in Economics

Prof C Greenough, LS Chin and Dr DJ Worth  
*Software Engineering, STFC Rutherford Appleton Laboratory*

Prof M Holcombe, Dr Simon Coakley and Dr Mariam Kiran  
*Computer Science, Sheffield University*
Contents

- Introduction – CSED/SoftEng
- Acknowledgement – EURACE
- HPC hardware
- FLAME – the software base
- Parallel FLAME implementation
- Verification and validation of FLAME
- Performance of FLAME infrastructure
- Performance of EURACE model
- Conclusions
The Computational Science and Engineering Department

- STFC (Science and Technology Facilities Council) has two main Laboratories with around 1500 staff in total:
  - Daresbury Laboratory (near Warrington) and at
  - the Rutherford Appleton Laboratory (near Oxford)
- Computation Science & Engineering has around 100 research and support staff
- Development and application simulation codes
  - Usually collaborating with Universities
  - Emphasis on high performance
- Interests in Science and Engineering

Rutherford Appleton Lab - ADACE Bielefeld 2010
Large Scale Facilities at the Rutherford Appleton Laboratory

Rutherford Appleton Lab

ISIS Neutron Source

Diamond Light Source

Rutherford Appleton Lab - ADACE Bielefeld 2010
Computational Science & Engineering

- Material Sciences
- Life Sciences
- Systems Biology

- Computational Engineering
- Ocean/Climate Modelling
- Advanced Algorithms
- Numerical Analysis
- Software Engineering
- Novel Hardware

Rutherford Appleton Lab - ADACE Bielefeld 2010
Software Engineering Group

- Software Engineering Support Programme (SESP)
  - Practical/Pragmatic software engineering methods
  - Development of software engineering tools
  - Education – workshop, seminars, tech reports...

- Intelligent Agent Technology
  - Agent-based frameworks and algorithm
  - Biological Systems
  - EURACE – EU Economic Modelling

- Applications
  - CFD
  - Heat transfer
  - Electromagnetics
  - Semi-conductors
  - Space detectors

- Parallel Algorithms
Acknowledgements

This research is funded by the European Union via the EURACE project (No 035086) which aimed to build a large scale agent-based model of the European economy to aid economic policy design.

The project required the development of an integrated multi-agent model of economic and financial markets and the development of software techniques and a software platform for large-scale agent-based economic simulations.
Acknowledgements

The EURACE Project Partners:

- **Economics/Finance**
  - Università degli Studi di Genova (UG) Italy (*Coordinator*)
  - Universitaet Bielefeld (UNIBI) Germany
  - Université de la Méditerranée (GREQAM) France
  - Università Politecnica della Marche (UPM) Italy

- **Software Engineering:**
  - University of Sheffield (USFD) UK
  - Università degli Studi di Cagliari (UNICA) Italy
  - STFC Computational Science & Engineering Dept (STFC) UK
  - National Research Institute of Electronics and Cryptology (UEKAE) Turkey

*Rutherford Appleton Lab - ADACE Bielefeld 2010*
Why High Performance Computing?

- Implementing software on an high performance system is difficult and time consuming so there must be a good reason to embark on the task:
  - Application can not be run on a conventional computing system – insufficient power and/or memory:
  - Agent population to large (m/p)
  - Agents are too complex (m/p)
  - Number of simulations to large (p)
    → Policy experiments
    → Validation process
    → Optimisation
Some Issues in High Performance Computing

- Parallel systems are in constant development
- Their hardware architectures are ever changing
  - simple distributed memory on multiple processors
  - share memory between multiple processors
  - hybrid systems –
    - clusters of share memory multiple processors
    - clusters of multi-core systems
  - the processors often have a multi-level cache system
Some More Issues in High Performance Computing

- Most have high speed multi-level communication switches
- Cloud/GRID architectures are now being used for very large simulations
  - many large high-performance systems
  - loosely coupled together over the internet
  - Specialised programming interfaces – no standards
- Performance can be improved by optimising to a specific architecture
- Can very easily become architecture dependent
- Cost – most serious HPC machines can be very expensive
Computing Systems

- **Workstations/Desktop Systems:**
  - Multi-core processors (4,8,...)
  - Add-on processors (GPGPU..)

- **High Performance Computing (HPC) Systems:**
  - Large multi-processor system (thousands of processors)
  - Coupled Multi-core systems
  - Complex communications hardware
  - Specialised attached processors (vector units, cells..)
Parallelism does not come for free!!

- Cannot magically transform a program to run efficiently on a parallel system
- Algorithm must be suitable for parallelisation
- There are such things as non-parallelisable algorithms
- Elements of work must localised – minimal dependencies on other task!
- Communication and synchronisation between processors significant overheads – so communication between task must be minimised
An agent-based modelling framework
- Initially developed by Simon Coakley (University of Sheffield). Extended in collaboration with STFC.
- Originally targeted at biological systems
- Developed further under the EURACE project:
  - Now support larger class of models (e.g. economic models)
  - Extension of the X-Machine Markup Language (XMML)
  - Optimised performance (serial and parallel)
  - Ported to various HPC machines (supercomputers) and Operating Systems
How is FLAME different?

- It is a generic ABM program generator
- It has been design with HPC in mind – written in C using MPI to manage communications
- Components are: model parser, a template library and a run-time library
- Uses a model definition written a dialect of XML together with user provided C code for agent functions
- It uses the concept of Message Boards for inter-agent communication
- Uses an agent dependency graph to schedule and optimise agent function (state change) activations
- Generates: the application and builder files for serial and parallel execution

Rutherford Appleton Lab - ADACE Bielefeld 2010
Message Boards

- All inter-agent communications are through messages boards
- There is a message board each message type within the model
- Messages only have two states – read or write (no read/write).
- The *message board library (libmboard)* manages these

*Filters*
Households can selectively read messages using filters.
- salary > 1000 and skill level = 4
The FLAME Process

- **Input from the modeller:**
  - Model – XMML file
  - C-code for functions

- **Input from FLAME**
  - Template file
  - Header files

- **Output from FLAME**
  - Applications code
  - State diagram
Creating a model

- What do we need to define:
  - Agents
    - Memory
    - Behaviour – functions/states/communications
  - Messages (information flow between agents)
  - Optional extras
    - Environment constants
    - Custom data types
    - Custom time units
Specifying a Model using XMML

- XMML is a dialect of XML (X-Machine Agent Mark-up Language)
- Standard set of XML tags
- Simple editable text file
- File has three main sections: models, environment, agents and messages
- EURACE develop a tool set to help model developers and users
XMML – Overall Structure

<xmodel>
  <name>circles</name>
  <models>
    <model>.... </model>
  </models>
  <environment>
    <constants>.... </constants>
    <functionfiles>.... </functionfiles>
    <timeUnits>.... </timeUnits>
  </environment>
  <agents>
    <xagent>.... </xagent>
  </agents>
  <messages>
    <message>.... </message>
  </messages>
</xmodel>
Programmers FLAME API

- Interface to agents and Framework
- Sending & receiving messages
  - `get_next_<message name>_message`
  - `add_<message name>_message`
- Accessing agent memory
  - `get_<variable>`
  - `set_<variable>`
- Creating & removing agents
Application Programmers Interface

The programmers interface to agent memory and to message board information is through the FLAME Programmers API.

Example 1: Circle agent with memory of x, y, id and radius communicating through the \textit{location} message.

```c
int outputdata()
{
  double x, y, radius;
  int id;

  x = get_x();  y = get_y();
  id = get_id();  radius = get_radius();

  add_location_message(id, (radius * 3), x, y, 0.0);

  return 0;
}
```

\textit{Rutherford Appleton Lab - ADACE Bielefeld 2010}
Application Programmers Interface

Example 2: Circle agent get-ing and set-ing memory values x, y, fx and fy using API functions.

```c
int move()
{
    double fx, fy;

    fx=get_fx();
    fy=get_fy();

    set_x(fx);
    set_y(fy);

    return 0;
}
```
API - MACROS

The FLAME parser generates predefined macros which allow the programmer to generate loops over message boards.

Example 3: A loop to scan over LOCATION message board

```c
#define START_LOCATION_MESSAGE_LOOP
for (location_message = get_first_location_message();
     location_message != NULL;
     location_message = get_next_location_message(location_message))
{
}

#define FINISH_LOCATION_MESSAGE_LOOP
}
```
The FLAME Process

- **Input from the modeller:**
  - *Model* - XMML file
  - *C*-code for functions

- **Input from FLAME**
  - *Template* file
  - Header files

- **Output from FLAME**
  - Applications code
  - State diagram
Two simple models

C@S Model
- mix of agents: Malls, Firms, People
- a mixture of state complexities
- all have position data
- agents have range of influence
- 9 message types
- 9 functions

Circles Model
- very simple agent
- all have position data
- $x$, $y$, $fx$, $fy$, radius in memory
- moves by repulsion from neighbours
- 1 message type
- 3 functions

Rutherford Appleton Lab - ADACE Bielefeld 2010
Dependency Graphs

Simple three-agent model

Firm

Person

Mall

Circle

Communications

State changes

Rutherford Appleton Lab - ADACE Bielefeld 2010
Issues with HPC and FLAME

- FLAME is an applications generator.
- Parallelism is hidden in the XML model and the modeller provided C-code – this is in terms of agent locality or population groupings.
- Inter-agent communications are captured in XML:
  - In agent function descriptions
  - In message descriptions
  - The frequency of messages is not known
- The *agent functions* are the computational load:
  - Their weight not known until runtime
  - They could be fine or coarse grained
  - Their activation is irregular – not lock stepped
Parallel Implementation

- Based on:
  - the distribution of agents – *computational load*
  - distribution of message boards (MB) – *data load*
- Agents only communicate via MBs
- Cross-node message information is made available to agents by message board synchronisation
- FLAME uses MPI to manage inter-node communications
- Communication between nodes are minimised
  - Multi-threading on computation and communication
  - Message filtering
  - Domain/group halos
Parallelism in FLAME
The goal of using a high performance parallel computer is to minimise the time taken to perform a simulation. We must balancing the use of resources available to achieve this.

Some issues:
- communicating between processors takes time
- communication must overlap computation
- the model must contain parallelism
- the model must be sufficiently large

Using all the available processors is not the solution.
A Very Simple Model

The *circle* agent is our basic test agent

- Very simple agents – zero size points in 2D space
- all have a 2D (x,y) positional data
- all have a *radius* of influence
- values of x, y and *radius* are in memory
- they move by repulsion from neighbours
- there is only 1 message type
- there are 3 functions
Model Partitioning

- **Round-robin: simple agent by agent allocation**
  - partitions are given a geometry
  - agents are allocated to partition’s centroid
  - agents distributed for load balance

- **Geometric: based on prime factors**
  - using position as separator
  - partitions are defined uniformly over $x$ and $y$ space
  - for prime numbers $x$ is preferred direction
  - could be used of multi-variable separators
Round-Robin Partitioning

- Range covers whole domain
- Centroids
Geometric Partitioning

![Geometric Partitioning Diagram]

- Processors $P_i$
- Halos
- Radius

Rutherford Appleton Lab - ADACE Bielefeld 2010
Partitioning by Region

- For economics geographical regions seem to be natural.
- We still need to understand the agent interaction the work they perform – the communication and computation load.
- Very difficult in unsteady multi-agent systems.
- Multiple agent weights.
- Start with a static analysis!
Parallelism in FLAME

Parallel agents grouped on parallel nodes.
Messages synchronised across nodes as necessary
Message board library allows both serial and parallel versions to work

Rutherford Appleton Lab - ADACE Bielefeld 2010
Dependency Graphs

Simple three-agent model

Firm

Person

Mall

Circle

Communications

State changes

S1

S2

S3

S4a

S4b

S5

S6a

S6b

Rutherford Appleton Lab - ADACE Bielefeld 2010
Message Filtering

- The XMML filter provides a way of selecting only the required data is transferred

```xml
<function><name>inputdata</name>
<currentState>1</currentState>
<nextState>2</nextState>
<inputs>
<messageName>location</messageName>
<filter>
  <lhs><value>a.id</value></lhs>
  <op>NEQ</op>
  <rhs><value>m.id</value></rhs>
</filter>
</input>
</inputs>
</function>
```

- Used to control scanning loops
- Used in message board synchronisation
Message Board Synchronisation

- At these critical points we need to synchronise the message information.
- To continue every agent must have in place the information it needs before the simulation can continue.
- Local message boards must be updated with necessary current information.
- In its simplest form synchronisation by full replication of all messages within each node – cannot be done in large populations – insufficient memory.
- We only transfer the information required as defined in the model XMML.
Multi-threading

- Synchronisation is a potential bottleneck as the simulation must wait for inter-node communication.
- To reduce this problem libmboard runs multiple threads:
  - one for communication - data transfer
  - one for computation - doing agent based work
- MB_SyncStar and MB_SyncComplete control this process
Parallel Platforms

- The FLAME framework has been successfully ported to various HPC systems:
  - SCARF – 360x2.2 GHz AMD Opteron cores, 1.3TB total memory
  - HAPU – 128x2.4 GHz Opteron cores, 2GB memory / core
  - NW-Grid – 384x2.4 GHz Opteron cores, 2 or 4 GB memory/core
  - HPCx – 2560x1.5GHz Power5 cores, 2GB memory / core
  - Legion (Blue Gene/P) – 1026xPowerPC 850 MHz; 4,096 cores
  - HECToR (Cray XT4) – 1416xQuad Core Opterons, 2GB / core, 22,656 cores
  - Leviathan (UNIBI) – 3xIntel Xeon E5355 (Quad Core), 24 cores
Verification and Validation

- It is important to ensure that applications generated by the FLAME framework execute correctly in both their serial and parallel modes.
- A set of simple test models and problems have been developed based on the Circles agent:
  - Test 1: single Circles agent type; Initial population of no agents.
  - Test 2: single Circles agent type; Initial population of one agent at (0,0).
  - Test 3: Two Circles agent type; Initial population of agents at (-1,0) and (+,0).
  - Test 4: Four Circles agent type; Initial population of one agent at (+/-1,+/-1).
  - Test 5: Four Circles agent type; Initial population of one agent at (0,+/-1) and (+/-1,0).
  - Test 6: Four Circles agent type; Initial population of one agent at random positions.
- In each of these models the expected results can be specified and they can provide a very simple check of the correctness serial and parallel implementations.

Rutherford Appleton Lab - ADACE Bielefeld 2010
Circles Model (1 Million Agents)

Benchmark of FLAME on Circles model (1 x 10^6 agents)

Rutherford Appleton Lab - ADACE Bielefeld 2010
C@S Model (124,000 agents)

Benchmark of C@S model using 124,000 agents

Time per iteration, seconds

Number of Processors

MPI Sends per iteration per processor

Rutherford Appleton Lab - ADACE Bielefeld 2010
What we don’t know!

- Size of agent population – agent could be created and/or destroyed
- Granularity of agents
  - Is there a large computational load
  - How often do they communicate
- Inherent parallelism (locality) in model
  - Are the agents in groups
  - Do they have short range communication
The EURACE Model

- The main goal of EURACE was to develop:
  - an agent-based software platform for European economic policy design with heterogeneous interacting agents
  - discover new insights from a bottom up approach to economic modeling and simulation.
  - multi-agent
  - multi-market
  - regional and global effects
EURACE markets and their interactions
The EURACE Model

Model Stats
• 9 Agent
• 55 Messages
• 159 Functions

Markets
• Labour
• Goods
• Credit
• Financial
EURACE Agent Populations

- Default unit of population
- 5 fixed national agents
- 4 regional agent groups
- Larger populations are cloned using the this basic population unit and replicating the regional agents

<table>
<thead>
<tr>
<th>Agent type</th>
<th>Number of agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>National</td>
<td></td>
</tr>
<tr>
<td>Government</td>
<td>1</td>
</tr>
<tr>
<td>Central_Bank</td>
<td>1</td>
</tr>
<tr>
<td>Clearinghouse</td>
<td>1</td>
</tr>
<tr>
<td>Eurostat</td>
<td>1</td>
</tr>
<tr>
<td>IGFirm</td>
<td>1</td>
</tr>
<tr>
<td>Regional</td>
<td></td>
</tr>
<tr>
<td>Mall</td>
<td>1</td>
</tr>
<tr>
<td>Bank</td>
<td>2</td>
</tr>
<tr>
<td>Firm</td>
<td>80</td>
</tr>
<tr>
<td>Household</td>
<td>1600</td>
</tr>
</tbody>
</table>
Performance analysis tools

- Two types of analysis tools have been developed for FLAME generated applications: static and dynamic. Static analysis tools process the model XMML and the C-code. The dynamic analysis tools provide information on the code during execution.

- **Static Analysis Tools:**
  - *Analyses_model.py*: a static analysis of the FLAME model which gives detailed information on the components of a model: agent, function and messages types, number and sizes, a static communications table, a weighted communications table.
  - *Check_message_consistency.py*: a static consistency checker which compares the XMML definition with C code and ensures that the number and usage of messages is consistent.
Performance analysis tools

- **Dynamic Analysis Tools:**
  - *The MM Package*: The MM package is a dynamic to monitor message tracing in the simulation. It is a set of additional directives included in the FLAME Templates which are embedded in the application code that monitor all message tracing and output to an SQL database. The database can be post-processed by the developer to assess the message tracing in the model.
  - *The Time Package*: The Timer package is a collection of timing utilities which allow detailed timing analysis of any FLAME generated application. The Timer package has been used to measure elapsed CPU time for functions and message board synchronizations.
Tools for Performance Analysis

- We need to assess the performance overhead of the FLAME Framework: message management; iterator creation and use; data input and output.
- The Timer Package is used to monitor the main message board activities.
- A FLAME uses multiple threads for computation and communication. FLAME attempts to overlap computational and communication. We test non-overlapped and overlapping overheads.
## Static Analysis

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firm</td>
<td>0.000</td>
<td>0.000</td>
<td>2.857</td>
<td>1.429</td>
<td>1.464</td>
<td>10.000</td>
<td>2.143</td>
<td>1.464</td>
<td>4.286</td>
</tr>
<tr>
<td>Central_Bank</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.036</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.036</td>
</tr>
<tr>
<td>Clearinghouse</td>
<td>2.143</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.429</td>
<td>0.714</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>IGFirm</td>
<td>1.429</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.036</td>
<td>0.714</td>
<td>0.000</td>
<td>0.000</td>
<td>0.714</td>
</tr>
<tr>
<td>Government</td>
<td>0.714</td>
<td>2.143</td>
<td>1.429</td>
<td>0.714</td>
<td>0.000</td>
<td>1.464</td>
<td>0.000</td>
<td>0.000</td>
<td>0.714</td>
</tr>
<tr>
<td>Household</td>
<td>6.464</td>
<td>0.000</td>
<td>0.714</td>
<td>0.000</td>
<td>2.179</td>
<td>0.000</td>
<td>1.429</td>
<td>0.036</td>
<td>0.714</td>
</tr>
<tr>
<td>Mall</td>
<td>0.714</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>2.857</td>
<td>0.000</td>
<td>0.036</td>
<td>0.000</td>
</tr>
<tr>
<td>Eurostat</td>
<td>0.036</td>
<td>0.036</td>
<td>0.000</td>
<td>0.000</td>
<td>0.714</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Bank</td>
<td>1.429</td>
<td>1.429</td>
<td>0.000</td>
<td>0.000</td>
<td>0.036</td>
<td>0.714</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

### Weighted Communications Matrix

*Rutherford Appleton Lab - ADACE Bielefeld 2010*
Effects of multi-threading

Effect of overlapping Communication and Computation

With no overlap - Node 0

With overlap - Node 0

With no overlap - Node 1

With overlap - Node 1
## Information from MM Package

<table>
<thead>
<tr>
<th>Writing</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>57,557</td>
</tr>
<tr>
<td>bank_account_update</td>
<td>1,551</td>
</tr>
<tr>
<td>job_application</td>
<td>666</td>
</tr>
<tr>
<td>job_application2</td>
<td>552</td>
</tr>
<tr>
<td>order_status</td>
<td>337</td>
</tr>
<tr>
<td>accepted_consumption_1</td>
<td>274</td>
</tr>
<tr>
<td>consumption_request_1</td>
<td>274</td>
</tr>
<tr>
<td>tax_payment</td>
<td>62</td>
</tr>
<tr>
<td>hh_subsidy_notification</td>
<td>60</td>
</tr>
<tr>
<td>hh_transfer_notification</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reading</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>2,935,407</td>
</tr>
<tr>
<td>quality_price_info_1</td>
<td>13,700</td>
</tr>
<tr>
<td>info_firm</td>
<td>7,850</td>
</tr>
<tr>
<td>accountInterest</td>
<td>6,000</td>
</tr>
<tr>
<td>dividend_per_share</td>
<td>3,000</td>
</tr>
<tr>
<td>bank_account_update</td>
<td>1,551</td>
</tr>
<tr>
<td>job_application</td>
<td>666</td>
</tr>
<tr>
<td>vacancies</td>
<td>666</td>
</tr>
<tr>
<td>job_application2</td>
<td>552</td>
</tr>
<tr>
<td>vacancies2</td>
<td>552</td>
</tr>
</tbody>
</table>

*Rutherford Appleton Lab · ADACE Bielefeld 2010*
### Serial Performance

<table>
<thead>
<tr>
<th>Function</th>
<th>Time (s)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearingHouse_receive_orders_and_run</td>
<td>82.81</td>
<td>72.00</td>
</tr>
<tr>
<td>Household_stock_beliefs Formation</td>
<td>25.32</td>
<td>22.00</td>
</tr>
<tr>
<td>Household_send_orders</td>
<td>2.06</td>
<td>1.70</td>
</tr>
<tr>
<td>Household_bond_beliefs_formations</td>
<td>0.44</td>
<td>0.38</td>
</tr>
<tr>
<td>Household_rank_and_buy_goods_1</td>
<td>0.42</td>
<td>0.36</td>
</tr>
<tr>
<td>Firm_read_job_applications_send_job_offer_or_rejection</td>
<td>0.37</td>
<td>0.32</td>
</tr>
<tr>
<td>Household_update_its_portfolio</td>
<td>0.16</td>
<td>0.14</td>
</tr>
<tr>
<td>Household_receive_dividends</td>
<td>0.11</td>
<td>&lt;0.10</td>
</tr>
<tr>
<td>Household_receive_info_interest_from_bank</td>
<td>0.09</td>
<td>&lt;0.10</td>
</tr>
<tr>
<td>Household_send_account_update</td>
<td>0.09</td>
<td>&lt;0.10</td>
</tr>
</tbody>
</table>

**Initial performance analysis**

<table>
<thead>
<tr>
<th>Function</th>
<th>Time (s)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Household_stock_beliefs_formations</td>
<td>245.78</td>
<td>61.00</td>
</tr>
<tr>
<td>Household_send_orders</td>
<td>46.44</td>
<td>11.00</td>
</tr>
<tr>
<td>ClearingHouse_receive_orders_and_run</td>
<td>41.76</td>
<td>10.00</td>
</tr>
<tr>
<td>Household_bond_beliefs_formations</td>
<td>4.98</td>
<td>1.20</td>
</tr>
<tr>
<td>Household_update_its_portfolio</td>
<td>1.48</td>
<td>0.30</td>
</tr>
<tr>
<td>Household_rank_and_buy_goods_1</td>
<td>1.04</td>
<td>0.28</td>
</tr>
<tr>
<td>Household_rank_and_buy_goods_2</td>
<td>0.90</td>
<td>0.22</td>
</tr>
<tr>
<td>Household_receive_dividends</td>
<td>0.80</td>
<td>0.20</td>
</tr>
<tr>
<td>Household_receive_info_interest_from_bank</td>
<td>0.79</td>
<td>0.20</td>
</tr>
<tr>
<td>Firm_read_job_applications_send_job_offer_or_rejection</td>
<td>0.62</td>
<td>0.15</td>
</tr>
</tbody>
</table>

**Performance analysis after initial optimisation**

*Rutherford Appleton Lab - ADACE Bielefeld 2010*
Amdahl’s Law

\[
\frac{1}{1 - P} \leq \frac{P}{N}
\]

Where \( N \) is the number of processors and \( P \) the fraction of the code that can be parallelised.

As \( N \to \infty \) \( P/N \to 0 \) and the \((1-P)\) term dominates. The proportion of serial code dominates the parallel performance.

*Rutherford Appleton Lab - ADACE Bielefeld 2010*
### Parallel Performance

#### Node 0: Performance analysis of agent functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Time (s)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Household_send_orders</td>
<td>2083.30</td>
<td>14.2</td>
</tr>
<tr>
<td>Household_stock_beliefs_formation</td>
<td>211.14</td>
<td>1.4</td>
</tr>
<tr>
<td>order</td>
<td>137.02</td>
<td>0.9</td>
</tr>
<tr>
<td>Household_receive_dividends</td>
<td>104.89</td>
<td>0.7</td>
</tr>
<tr>
<td>Household_receive_data</td>
<td>43.66</td>
<td>0.3</td>
</tr>
<tr>
<td>Household_receive_info_interest_from_bank</td>
<td>37.16</td>
<td>0.3</td>
</tr>
<tr>
<td>Household_update_its_portfolio</td>
<td>34.56</td>
<td>0.2</td>
</tr>
<tr>
<td>Firm_read_stock_transactions</td>
<td>17.74</td>
<td>0.1</td>
</tr>
<tr>
<td>Household_rank_and_buy_goods_1</td>
<td>16.10</td>
<td>0.1</td>
</tr>
</tbody>
</table>

#### Node 1: Performance analysis of agent functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Time (s)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearingHouse_receive_orders_and_run</td>
<td>5125.29</td>
<td>35.0</td>
</tr>
<tr>
<td>Household_send_orders</td>
<td>2067.41</td>
<td>14.1</td>
</tr>
<tr>
<td>Household_stock_beliefs_formation</td>
<td>222.10</td>
<td>1.5</td>
</tr>
<tr>
<td>Household_receive_dividends</td>
<td>104.26</td>
<td>0.7</td>
</tr>
<tr>
<td>order</td>
<td>75.31</td>
<td>0.5</td>
</tr>
<tr>
<td>Household_receive_data</td>
<td>43.50</td>
<td>0.3</td>
</tr>
<tr>
<td>Household_receive_info_interest_from_bank</td>
<td>37.06</td>
<td>0.3</td>
</tr>
<tr>
<td>Household_update_its_portfolio</td>
<td>34.41</td>
<td>0.2</td>
</tr>
<tr>
<td>Household_rank_and_buy_goods_1</td>
<td>16.49</td>
<td>0.1</td>
</tr>
</tbody>
</table>

---

*Rutherford Appleton Lab - ADACE Bielefeld 2010*
Gustafson’s Law

- Gustafson's law addresses scaling to match availability of computing power as the machine size increases.

\[ S(N) = N - \alpha \bullet (N - 1) \]

where \( \alpha \) is the serial fraction and \( N \) the number of processors.

- It removes the fixed problem size or fixed computation load on the parallel processors: instead, he proposed a fixed time concept which leads to scaled speed up for larger problem sizes (i.e. weak or soft scaling).
## Parallel Performance

<table>
<thead>
<tr>
<th>Model</th>
<th>Regions per Processor</th>
<th>No. of Agents</th>
<th>Time (0.02)</th>
<th>Time (0.01)</th>
<th>Time (0.005)</th>
<th>Time (0.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10R_10P</td>
<td>1</td>
<td>16,844</td>
<td>9.790</td>
<td>3.716</td>
<td>1.840</td>
<td>0.232</td>
</tr>
<tr>
<td>20R_10P</td>
<td>2</td>
<td>33,684</td>
<td>61.712</td>
<td>33.914</td>
<td>9.671</td>
<td>0.682</td>
</tr>
<tr>
<td>30R_10P</td>
<td>3</td>
<td>50,524</td>
<td>453.321</td>
<td>144.203</td>
<td>59.872</td>
<td>1.344</td>
</tr>
<tr>
<td>40R_10P</td>
<td>4</td>
<td>67,364</td>
<td>854.781</td>
<td>254.785</td>
<td>107.772</td>
<td>1.707</td>
</tr>
<tr>
<td>50R_10P</td>
<td>5</td>
<td>84,204</td>
<td>2083.108</td>
<td>578.451</td>
<td>305.411</td>
<td>2.605</td>
</tr>
<tr>
<td>60R_10P</td>
<td>6</td>
<td>101,044</td>
<td></td>
<td>262.061</td>
<td>107.489</td>
<td>3.535</td>
</tr>
<tr>
<td>70R_10P</td>
<td>7</td>
<td>117,884</td>
<td></td>
<td>101.094</td>
<td>55.440</td>
<td>4.504</td>
</tr>
<tr>
<td>80R_10P</td>
<td>8</td>
<td>134,724</td>
<td></td>
<td>76.171</td>
<td>30.587</td>
<td>5.739</td>
</tr>
<tr>
<td>90R_10P</td>
<td>9</td>
<td>151,564</td>
<td></td>
<td>73.339</td>
<td>41.525</td>
<td>6.660</td>
</tr>
<tr>
<td>100R_10P</td>
<td>10</td>
<td>168,404</td>
<td></td>
<td>97.151</td>
<td>96.784</td>
<td>8.174</td>
</tr>
<tr>
<td>200R_10P</td>
<td>20</td>
<td>336,804</td>
<td></td>
<td>361.985</td>
<td>25.716</td>
<td></td>
</tr>
<tr>
<td>300R_10P</td>
<td>30</td>
<td>504,712</td>
<td></td>
<td></td>
<td>64.041</td>
<td></td>
</tr>
</tbody>
</table>

*Rutherford Appleton Lab - ADACE Bielefeld 2010*
Serious Parallel Performance

- We have used two test models each with two different populations in our testing:
  - Population 1:
    - 20,212 agents, 12 regions
    - Run on 1, 2, 3, 4, 6 and 12 processors
  - Population 2:
    - 101,044 agents, 60 regions
    - Run on 5, 10, 15, 20, 30 and 60 processors
- The value of trading_activity also been varied.
EURACE Results – Pop1

Pop1: 20,212 agents, 12 regions

Rutherford Appleton Lab - ADACE Bielefeld 2010
EURACE Results – Pop2

Pop2: 101,044 agents, 60 regions

Rutherford Appleton Lab • ADACE Bielefeld 2010
Some observations

- We do get limited improvement of performance
- The performance is very model dependent
  - the serial component of the model
  - the weight of each agent task
- Performance is very architecture dependent
  - the speed of the processors
  - the speed of the interconnect
  - the size of the available memory
- It is difficult for modellers to express the parallelism in their applications
Research Issues

A short list of research subjects:
- definition and use of message filtering
- optimisation of scheduling from task graph
- generating communications overlap
- maintaining load balance over the system
- detecting serialism in the model and transforming
- coupling models with computational steering
- Use of multi-core processors and GPUs using OpenMP or OpenCL/CUDA
- Verification and Validation methods and tools
- Use of in-complete data – try to carry on regardless!
Dynamic Load Balancing

- Goal to move agents between compute nodes:
  - reduce overall elapsed time
  - increase parallel efficiency
- There is an interaction between computational efficiency and overall elapsed time
- The requirements of communications and load may conflict!
Balance - Load vs. Communication

- **Distribution A**
  - P1: 13 agents
  - P2: 3 agents
  - P2 <-> P1: 1 channel

- **Distribution B**
  - P1: 9 agents
  - P2: 7 agents
  - P1 <-> P2: 6 channels
Moving Wrong Agents

Moving wrong agents could increase elapsed time

Problem of Load Imbalance

![Bar chart showing work by agents and elapsed time for different numbers of partitions]
Conclusions

- FLAME has proven to be a very versatile program generator for agent-based applications
- The FLAME overhead in both the serial and parallel implementations of FLAME applications is acceptably small - ~5% of total elapsed time
- The parallel performance of the FLAME application is very dependent of the inherent locality expressed in the model and the architecture of the target hardware
- By using parallelisation techniques we have successfully run populations of 500,000 agents
- To gain the best possible performance the modeller must understand and exploit the nature of parallel computing
Contacts

Prof Chris Greenough
Software Engineering Group
Computational Science & Engineering Dept
STFC Rutherford Appleton Laboratory
Harwell Science & Innovation Campus
DIDCOT
Oxfordshire OX11 0QX

Tel: +44 1235 445307
Email: christopher.greenough@stfc.ac.uk
Web: http://www.cse.scitech.ac.uk/seg
Based on Publications/Reports

- C. Greenough, DJ Worth, LS Chin: *Parallel Optimisation of the EURACE Agent-Based Economic Model*, Rutherford Appleton Laboratory Technical Report, Jul 2010
- C. Greenough, DJ Worth, LS Chin: *Porting of agent models to parallel computers*, Deliverable D1.4, EURACE Project, 2006
- C. Greenough, DJ Worth, LS Chin: *Porting of agent models to parallel computers*, Deliverable D8.4, EURACE Project, 2009