Computer Systems Research at Edinburgh

The Computer Systems Group

Abstract

We introduce a new series of technical reports produced by the Computer Systems Group within the Computer Science Department at the University of Edinburgh. We consider the ethos of the group and its relationship with the wider subject. Some highlights of past "systems" research in the department are reviewed and the current interests of group members are surveyed.

1 Computer Systems Research

The inauguration of a new report series begs the question "Why?". What should the reader expect to find here which might not appear in an existing series? To attempt an answer, it seems appropriate to consider what might be the nature of "Computer Systems Research".

The range of disciplines which shelter beneath the umbrella of Computer Science, both here in Edinburgh and elsewhere, is very wide. While it might be possible to pick two superficially disparate topics and search in vain for a substantial direct link between them, it is certainly a strength of the subject that it is difficult (and in any case highly undesirable) to draw isolating boundaries around wider groupings, labelled "theory", "practice", "science", "engineering" or whatever. Interaction across such boundaries is the lifeblood of the subject, whether it occurs between smaller research teams, or within the work of individuals. In founding the report series of the Department's Laboratory for the Foundations of Computer Science (LFCS), Robin Milner argues forcefully that [4]

> "We reject any artificial barrier between theory and engineering in Computer Science, and anticipate a healthy flow of ideas between the Laboratory and the Department as a whole."

With such a continuum in mind, we can define "systems" research in terms of a different focus, on aspects of the subject which concern the design, capabilities and performance of actual computer systems, rather than in terms of dividing lines. In any piece of work we may expect to find influences from various points on the spectrum.

As the name suggests, we hope that this series will act as a natural repository for reports in which the dominant flavour is of "system building" (of whatever kind) but in recognition of the fact that such a categorisation is very loose, we encourage the reader to browse the complementary LFCS and Departmental series. Just as the value of a theory can only be tested (and indeed properly developed) by exposure to application, so good practice must be informed by sound principle. It is in this spirit of interaction, exchange and interdependency that the new report series is launched.

2 Past Highlights

The department has (we believe!) a long and distinguished history of systems research which, over the years, has produced a number of significant systems in a range of areas. We now briefly highlight a selection of these which we hope are representative both historically and in breadth of coverage. It must be emphasized that this is a selection, rather than a comprehensive catalogue.

2.1 The EMAS Operating System

The Edinburgh Multi Access System [2] developed in the late 1960's and early 1970's was a time sharing operating system, which originated in the department and went on to support the University's main computing service for twenty years on a variety of hardware including ICL 4/75, ICL 2900 and IBM XA compatible machines.

2.2 The Fred Machine

The 'Fred Machine' project [1] was active in the late 1970's and early 1980's. Its focus was on the development of a flexible computer system, in which the emphasis was on the ability to configure and reconfigure soft and hard components to meet varying computational demands. The central feature was the 'Fred Bus' to which other building blocks were attached as required. One popular configuration produced the 'Advanced Personal Machine', a general purpose workstation much used in the Department for both teaching and research for several years.

2.3 The Standard ML Programming Language

Standard ML [5] is a strict, functional programming language with all the powerful, elegant features one would expect of such an artefact. It was developed within the department (and latterly, more specifically in the LFCS) over a number of years, particularly the early to mid 1980's culminating in 1987 with receipt of the British Computer Society Technical Award. Extensions and developments are still active areas of research (reported on in the LFCS report series).

2.4 VLSI Design Tools

In the early to mid 1980's a number of projects were active in the creation and application of languages, tools and methodologies for the design of VLSI circuits. As an example, the Configurable Array Logic (CAL) project emerged from a departmental PhD thesis [3] and has since developed into a commercial "virtual hardware" product. A simple programmable cellular array can be customised dynamically to produce fast implementations of specialised compute-intensive functions.

2.5 The Posie Project

In the late 1980's and early 1990's, the Posie project [6] pulled together a number of strands of research relating to the performance monitoring and scheduling of parallel computations, particularly those expressed in a message-passing style. A "testbed" parallel machine was constructed, with an emphasis on transparent monitoring of processor and communication activity, and a number of scheduling and load balancing schemes were investigated both by simulation and in practice on the testbed and on the Meiko Computing Surface, a 400 transputer parallel machine (one of the most powerful available at the time) housed by the Edinburgh Parallel Computing Centre.

3 Current Interests

Members of the department currently associated with the group have interests in many areas. We now provide a snapshot of current activities in order to illustrate the breadth of coverage. In contrast to the preceding historical section, we do not provide a reference list - the report series itself is intended to become a bibliography of active research within the group. The activities have been arranged under the headings "Computer Architecture", "Computer Graphics", "Human Factors in Computer Systems Design", "Object-Oriented Database Systems", "Parallel Software and Algorithms", and "Quantitative System Evaluation". Of course, the categorisation is meant to be helpful rather than exclusive or definitive.

3.1 Computer Architecture

Architectural Simulation

A computer architecture and its implementation can be represented in a number of ways and at different levels of abstraction. At the highest level are multiprocessor systems, while at the lowest an architecture consists of an ensemble of transistors on a VLSI chip (or set of chips). A variety of simulators has been designed for each different level of abstraction, and some design tools can generate implementations automatically at lower levels. The Hierarchical Architectural design and

Simulation Environment (HASE) allows designers to create architectural designs at different levels of abstraction, to explore designs through a graphical interface and to invoke the appropriate simulator at each level. The output of a simulation run can be used to animate the graphical display of the design, allowing the user to visualise what is happening inside the architecture as programs are run on it. Working at the processor architecture level, for example, a designer could investigate bottlenecks in the flow of instructions and try alternative design strategies to eliminate them.

The component parts of a computer can be treated very naturally as objects, and so HASE uses the object oriented simulation language Sim++ and an object oriented database, ObjectStore, to store both the object libraries and the designs created from them.

High-Performance Computer Architecture

Computer Architecture is a continually evolving discipline. It embraces a wide spectrum of activities in computer science: from the design of hardware components, and the technology of integrated circuits, through the organisation and structure of computing systems, to compiler and language issues, and ultimately to the applications which use the resulting computer systems.

Our research in the area of high-performance computing is currently centred around the notion of decoupled architectures. This is an implementation technique for high-performance systems in which the activities of control-flow management, address generation and operation evaluation are partitioned across closely coupled and highly specialised sub-processors. These represent a form of MISD architecture, and their principal goal is to overcome the ever-increasing discrepancy between processor cycle times and memory latency. Work in this area involves looking at not only the structures out of which one can construct such architectures, but also at the decoupling behaviour of applications and the ways in which new compilation strategies can be used to optimise the run-time characteristics of decoupled systems.

Tools for investigating decoupled systems are under construction, and include both a prototype compilation system and architecture simulators. These are being developed in order to discover the extent to which real-life applications can be decoupled, and the extent to which compilation decisions for decoupled architectures can be deferred until run-time. Effectively, the memory system in a decoupled architecture behaves as if it were part of the processor's pipeline, and with large memory latencies there may be data dependencies within this extended pipeline. Such dependencies interfere with the pipeline flow, and therefore need to be avoided. However, in many cases, these dependencies cannot be resolved at compile time but could be resolved straightforwardly at run-time. This is a problem similar to, but subtlely different from, the traditional vectorization decision problem, and new approaches to finding efficient solutions are being sought.

Evaluation of Multiprocessor Interconnection Networks

Message passing networks are now an established part of multiprocessor systems design, with machines incorporating such networks being produced by a variety of vendors. The networks used vary dramatically both in topology and protocol, and in the design of the interconnect switching components from which they are constructed.

In order to assess the relative value of these networks it is necessary to ascertain the variation in system performance with changes in the network. It is inappropriate to run a simulation of a complete parallel machine, so in this project HASE (see above) is used to set up a testbed containing simple processor models which can generate network activity corresponding to that found in standard benchmarks used in the evaluation of real parallel systems.

Full models of the various networks under test can then be instantiated in the testbed and simulations run. This arrangement will allow both the impact of different networks on system performance and the utilisation of these networks by different algorithms to be assessed.

The knowledge gained will be used in combination with previous experience gained in developing the Xbar network IC to determine the practicality of providing improvements in the network to enhance system performance.

Computational Nanotechnology

Molecular nanotechnology is predicted to remake technology from the bottom up via the explicit control of atoms and molecules as building blocks. Nanoscale molecular machines will be able to guide the placement of molecules and atoms, enabling the construction of atomically precise materials and devices. While there is debate about the time frame for developing molecular manufacturing capabilities, it is clear that computational tools can substantially reduce the development time. Molecular computer aided design and modeling (CAD/CAM) software, and related specification tools, will allow “molecular engineering” to be planned and analysed via computer before construction is undertaken, just as in other engineering fields. Research in this area involves the development of computational chemistry tools for specifying, designing and modelling molecular machines at the nanoscale level.

Vector Processing

The Sparse Project is concerned with the design and implementation of hardware and software for a sparse vector processing system. The hardware is incorporated into an existing workstation where it acts as an accelerator providing specialist support for the processing of sparse vectors. The system is also being simulated using HASE.

The project also involves work on compilers and on language extensions to give programmers access to the sparse vector support mechanisms provided in hardware. The applicability of these mechanisms to other types of computing is also being investigated.

3.2 Computer Graphics

Current research interests include tree-based hidden surface removal, bidirectional ray tracing, visualization and virtual reality. Close collaboration is maintained with researchers in the Edinburgh Parallel Computing Centre. Work on BSP trees concentrates on complexity issues of constructing good trees and in restructuring trees for dynamic scenes. The visualization work proceeds in collaboration with industry to investigate techniques for visualizing large datasets for different functional requirements. Current virtual reality interests include the modeling and simulation of artificial environments, distributed simulations and integrated frameworks for optimising visual detail. Most of this work is in conjunction with, or strongly influenced by, researchers in the Virtual Environment Laboratory in the Department of Psychology. Resources include immersive technology (e.g. Head Mounted Displays) and non-immersive technology (e.g. a simple driving simulator) in Psychology and a Silicon Graphics Reality Station in Computer Science.

3.3 Human Factors in Computer Systems Design

Human factors addresses the problems inherent in matching the functionality of computer systems with the needs of their users. Understanding users and their working environment is critical to the success of any computer system. Human factors research has been instrumental in the improvements in system usability achieved over the past decade. There are still significant gaps, however, in our understanding of user interface design requirements for many specialised application environments. Moreover, advances in the *individual* user interface, and the proliferation of desktop computing, have served to highlight the importance of the *group* user interface. On a larger scale, the problems of ensuring that IT fully addresses business objectives are growing in complexity. A recent survey revealed that only a small proportion of commercial IT investment currently leads to successful applications.

Human factors research is by nature inter-disciplinary. It covers themes ranging from technologies and techniques for interaction, through user performance and behaviour, to design methodologies, the study of innovation, and industrial sociology. There are number of related active areas of research within the department.

In the design of user interfaces and systems for computer-supported collaborative work, recent research has been studying the impact of computer mediation on group processes such as the sharing of information and coordination of activities.

In the area of user interfaces for medical imaging applications, work is in progress on the human factors of computer-aided digital mammography. This project is a collaboration with radiologists in the U.K. breast screening programme. It involves the design and evaluation of specific user interfaces, and the detailed study of work practices in screening mammography.

In collaboration with the University's Research Centre for Social Sciences, work proceeds on organisational factors in the design and implementation of IT systems. A major recent project focused on the problems companies face in meeting the growing demand for specialist IT expertise, and in integrating it effectively with the more traditional forms of business expertise such as that held by managerial and administrative staff.

Finally, improved user interfaces for parallel program development and performance analysis tools are being investigated in collaboration with the Edinburgh Parallel Computing Centre.

3.4 Object-Oriented Database Systems

In the past, database systems were passive repositories of data, limited in use to areas such data processing. Object-oriented database systems have made it possible to apply database technology to new areas of computing. For example, we are using object-oriented database systems in two research projects in the Department, namely medical imaging and architecture simulation. These new application areas have created a number of research issues. In particular, we are investigating the possibilities for exploitation of the advanced transaction processing capabilities of object-oriented databases to support distributed applications in a client/server computing environment.

3.5 Parallel Software and Algorithms

Models of Parallelism

A *model* of parallelism is an abstract view of a parallel computing system, or more appropriately a part of a system, obtained by removing details in order to allow one to discover and work with the basic principles. Within the context of a hierarchy of model types at different levels of abstraction - architectural, computational and programming models - research interests include the definitions of particular models and the mappings between different model types. The aim is to find a means of bridging the gap between theory and practice of parallel computing. Particularly desired system principles are cost-effectiveness and scalability.

The Hierarchical PRAM (H-PRAM) model is the main focus of work into computational models. Based on this model, work in progress includes parallel algorithm design and analysis, investigation into its relationships to overlying programming models based on Algorithmic Skeletons, and mapping of the model to parallel architectures as simulated by the department's HASE architecture simulation environment.

Skeletal Parallelism

Many efficient algorithms for message-passing parallel computers share common patterns of data distribution, process distribution and communication. The complexity of programming such machines can be reduced (for some problems) by defining a library of such patterns of computation, expressed imperatively as equivalent sequential program skeletons, or declaratively as higher order functions. Work in this area involves the collection of a set of such structures and an investigation of the implementation decisions involved when a program is constructed as a complex composition of several skeleton instances. When presented in a functional context (for example as in the Bird-Meertens theory of lists) interesting opportunities for program development by transformation arise.

Formal Methods of Parallelisation

Programming should be a problem solving activity. Issues that have nothing to do with the problem should, if at all possible, not be the burden of the programmer but should be part of the compilation process. In many applications, parallelism and communication are implementation, not problem solving issues. In this sense, they are 'low-level' concepts like gotos or pointers, except that they are more difficult to use and verify.

There are classes of programs into which maximum parallelism can be infused mechanically. The resulting parallel program emulates a *systolic array*. A systolic array is a distributed network of sequential processors that are linked together by channels in a particularly regular structure. Typical applications are highly repetitive algorithms on large data structures such as those which occur in image or signal processing, meteorology, etc.

Automatic methods distribute the operations of source programs that do not specify concurrency or communication into time and space (*systolic design*). The target descriptions of these methods are distribution functions that form an abstract specification of the systolic array. The array can then be refined into software (i.e. into a distributed program) by a process of *systolizing compilation* or into hardware (i.e. into a chip layout).

Current research includes (1) the systolization of application problems, (2) classifications and extensions of systolic design methods and (3) the development of systolizing compilation techniques.

The Dynamic Behaviour of Parallel Programs

It is difficult for programmers to utilize parallel machines effectively unless they are prepared to learn a considerable amount about the idiosyncrasies of each new machine and spend time tuning up their programs for that particular environment. The resulting programs are then not necessarily portable to a new environment. Apart from the waste of time, the overheads involved in using parallel machines have worked against their widespread acceptance by the general community of applications programmers.

The aim behind the work described here is to underpin the design of operating systems that would remove some of the burden from programmers by supporting run time load-balancing. The hope would be that generally good performance could be achieved, even if the performance was not optimal for any particular program. However, to provide a firm basis for design, it is first necessary to understand more about the run-time behaviour of parallel programs; to see whether there are similarities, from a statistical point of view in the behaviour of various types of programs, and to study the evolution of programs in time. Unlike most performance studies of parallel programs, where the main interest has been to minimize the execution times of particular programs by an investigation of the detailed instruction sequences, this work is directed towards characterizing *classes* of parallel program in terms of a small number of parameters which refer to global properties of the program, like average process granularity, or average message size. This approach works well for one of the simplest (and commonest) types of parallel program with a fixed number of communicating processes. Good, quantitative predictions of performance in a given environment can be obtained from empirically-determined formulae. It is also possible to relate the improvement that can be brought about by run-time process migration to the parameter values of the program model. These studies show that it is feasible to interpret performance in terms of a very simple performance model, which can be extended to describe time-varying behaviour.

Work is currently proceeding in a number of directions: extending the empirical models to include parameters describing the machine environment, studying the relation between probabilistic changes on individual processors and what is observed on a macroscopic scale, and looking at the requirements for the design of operating systems that are both provably correct and support good user program performance.

3.6 Quantitative System Evaluation

Quantitative system evaluation is concerned with the capture and analysis of the dynamic behaviour of systems such as computers, communication networks and flexible manufacturing systems. This involves the investigation of the flow of data, control or goods within and between components of the system. The aim is to understand the behaviour of the system and identify the aspects of the system which are sensitive from a quantitative point of view.

The size and complexity of many modern systems result in large, complex models. This has led to a renewed interest in simulation techniques and has stimulated new work in the areas of compositional approaches to model construction; model simplification and state space reduction techniques; and efficient solution algorithms.

Functional Design and Performance Prediction

Historically, quantitative analysis and investigation of the temporal properties of systems has been distinct from qualitative analysis which aims to establish the functional behaviour of systems. However, recent work with stochastic process algebras and the use of pure process algebra to study simulation models allow these important approaches to integrated.

One such project concerns the development and application of Performance Evaluation Process Algebra (PEPA), a stochastic process algebra for the design of distributed systems. Used as a design tool, a stochastic process algebra encourages the early consideration of the timing characteristics of computer systems. PEPA has been used to model local area networks and multi-server/multi-queue systems, representing performance information and functional information in a single model.

The advantages of such an approach include the natural compositionality of process algebra models and the availability of several novel approaches to model simplification and state space reduction which are based on equivalence relations developed for the PEPA language. The novel combination of performance information and functional information facilitates the study of the performance implications of the functional properties of a system: examples of these include computing the mean time until deadlock or computing the optimal time-out of a service.

Reasoning about PEPA models proceeds by considering the derivation graph obtained from the model using the underlying operational semantics of the PEPA language. The derivation graph is systematically reduced to a form where it can be treated as the state transition diagram of the underlying stochastic (in fact, Markovian) process. From this can be obtained the infinitesimal generator matrix of the Markov process. A steady state probability distribution for the system can then be obtained, if it exists. Alternatively, transient probability distributions may be calculated.

Models can be analysed using the PEPA workbench: a set of simple tools to allow a skilled user of the PEPA language to delegate to machine assistance some of the routine tasks in checking descriptions and performing calculations of transition graphs and rewards.

Future work will consider extensions of traditional model-checking approaches for labelled transition systems to the multi-transition systems used to define stochastic process algebras.

Simulation Techniques

Work in improving the effectiveness of simulation modelling concentrates on easier description of simulation models, especially use of graphical and hierarchical formalisms, on efficient exploitation of simulation (and potentially other) models, through the use of object oriented database technology to control experimentation, and increasing integration with qualitative methods, such as protocol specification languages and process algebras. These approaches are exploited to build novel tools for modelling.

Applications of Performance Modelling

The application of performance analysis techniques is a central aspect of computer systems research. Recent examples include the study of parallel program execution using discrete event simulation, the design of efficient process migration strategies using specially built monitoring hardware and the modelling of heterogeneous computer networks linked by FDDI rings.

Work is now beginning on evaluating and managing high bandwidth networks using the asynchronous transfer mode (ATM) technology which is likely to be the standard for high speed communication networks. It is desired to operate these networks in a manner that ensures very low cell loss probabilities, of the order of 10^{-6} or less. This work deals with estimating the probability of such events, using techniques from large deviations theory. The goal is to develop analytical tools that enable the study of large systems starting from an understanding of the behaviour of individual units.

Other related work includes the problem of pricing resources in multimedia networks, so as to achieve efficient and equitable sharing of resources between diverse applications and empirical studies on real multimedia traffic, designed to test the validity of theoretical work in this field.

3.7 Very Large Scale Integration

The main themes of activity are Computer Aided Design (CAD) of VLSI circuits and novel VLSI Architectures. CAD tools being developed include those involving the use of formal techniques for synthesis and verification of designs. Behavioural (high-level) synthesis is also under investigation.

Formal Techniques

Current CAD tools for VLSI manage complexity by manipulating a structural hierarchy. To manage the increasing complexity resulting from technological advances of VLSI fabrication, tools are required which can manage the hierarchy of behavioural abstractions used in system design. A formal model which naturally and accurately represents circuit behaviour is central to such tools.

Behavioural models with the generality necessary to describe behaviour at many levels have been introduced by researchers studying formal verification of hardware designs.In this work, techniques from formal logic have been used to establish the correctness of a circuit design by mathematical proof prior to fabrication.

The major thrust of current research at Edinburgh involves the development, in cooperation with industry, of practical formal models of digital behaviour, and of the temporal and data abstractions employed in system design. This work is applied to the development of sound design methodologies and to the development of design tools for interactive and automated behavioural synthesis and verification of digital systems, both hardware and software.

Novel Architectures

A cellular array structure has been developed in which the array elements are simple but configurable logic blocks. Not only is the function of each element configurable, but also its communication with adjacent elements. The increasing density of fabrication technologies means that increasingly large arrays of elements can be fabricated, and so realistically-sized computations can be mapped onto the arrays with large speed gains resulting from concurrency. Various applications of arrays are being investigated, including the design of program-specific arithmetic pipelines, the emulation of hypercube algorithms and the implementation of data compression and encryption algorithms. This work develops the foundations laid by the CAL project as reported in the "Past Highlights" section of this report.

References

[1] Brebner, G., and King, F. *The Evolution of the Fred Machine.* Technical Report CSR–246–87, Dept. of Computer Science, University of Edinburgh, 1987.

[2] H. Whitfield and A. Wight. EMAS - The Edinburgh Multi-Access System. *Computer Journal 16* (1973), 331–346.

[3] Kean, T. *Configurable Logic: A Dynamically Programmable Cellular Architecture and its VLSI Implmentation.* PhD thesis, University of Edinburgh, Computer Science Department, 1989.

[4] Milner, R. *Is Computing an Experimental Science?* Technical Report ECS–LFCS–86–1, Laboratory for Foundations of Copmuter Science, Department of Computer Science,University of Edinburgh, 1986.

[5] Milner, R., Tofte, M., and Harper, R. *The Definition of Standard ML.* MIT Press, 1990.

[6] Pooley, R. *Introduction to the Second Posie Report.* Technical Report CSR–6–90, Dept. of Computer Science, University of Edinburgh, 1990.