Individual Grant Review
Functional Distributed Interactive Systems
GR/M55633/01
1/9/99 - 31/1/2001

Phil Trinder Robert Pointon
Hans-Wolfgang Loidl
Dept. of Computing and Electrical Engineering,
Heriot-Watt University, Edinburgh, EH14 4AS.

Abstract

The project has achieved or exceeded all three objectives as follows. A distributed non-strict language, Glasgow distributed Haskell (GdH) has been designed, implemented and made publicly available. Several realistic distributed applications have been constructed in GdH and measured alongside object-oriented and other functional equivalents. The measurements are the basis for a positive preliminary evaluation of the suitability of lazy functional languages for small-scale distributed applications.

We have published widely, collaborated with industrial and both national and international academic partners. To support ongoing work we have obtained two international travel grants. The project research forms the core of a PhD thesis nearing completion and the starting point for a new PhD.

1 Background

The number of computers in our society is increasing exponentially, and they are no longer isolated: wireless and wired network technologies are vastly increasing machine connectivity. Constructing distributed software for these new systems has focussed research interest on distributed programming languages. The potential benefits of the functional paradigm for distribution have been recognised for some time: good abstraction mechanisms, sophisticated type systems, high-level computation and co-ordination notations, and most importantly the purely-functional components are amenable to reasoning and free from a rigid evaluation order. An elaborate implementation is required to support such sophisticated languages, and although a number of designs had been developed, at the start of the project there were very few robust, publicly-available distributed functional language (DFL) implementations.

The project has designed, implemented and evaluated Glasgow distributed Haskell (GdH), a modest conservative extension of the standard Haskell non-strict functional language[Peyton Jones *et al.*, 98]. The Glasgow Haskell Compiler (GHC) is arguably the best non-strict functional language implementation, and as from version 5.0, GdH is bundled with the standard release.

In academia several other DFLs are under development, e.g. Haskell with Ports [Huch *et al.*, 00], Distributed Haskell [Chakravarty, 98], Curry [Hanus 00], Brisk [Holyer *et al.*, 98], Oz [Haridi*et al.*, 97], and Pict [Wojciechowski, 00]. In contrast to GdH with its shared address space most of these languages rely on explicit communication between processes and therefore require more detailed control of the distribution. In the telecommunications industry the Erlang DFL and associated Open Telecom Platform are being used successfully by number of large companies including One-2-One, Ericsson and NorTel. For example Ericsson have constructed a 525K-line AXD301 ATM Switching System distributed over up to 32 processors [Blau & Rooth 98]. In both academia and industry the competing language technology is object oriented, and typified by Java. Consequently we have evaluated GdH with both the emerging DFLs and Java.

In the following text project publications are cited numerically, thus [2], while publications by others or published before the start of the project, are cited thus: [Kiteck, 92].

```
myPEId  :: IO PEId
allPEId :: IO [PEId]

class Immobile a where
  owningPE :: a -> IO PEId
  revalIO  :: IO b -> a -> IO b

instance Immobile PEId
```

Figure 1: GdH Distribution Primitives.

2 Key Advances

The project ran for 17 months from September 1999 to January 2001 and made technical contributions in the following three areas.

2.1 Distributed Language Design

The objective addressed by this subsection and the next is:
Design and implement a robust, architecture-independent distributed variant of the standard Haskell lazy functional language.
To meet the objective we have designed a modest conservative extension of Haskell'98 [Peyton Jones *et al.*, 98] named Glasgow Distributed Haskell (GdH). The Glasgow Haskell Compiler already supported functional interactive systems via the language extensions for concurrency, parallelism, and graphical interfaces, but lacked support for distribution.

A key feature of the language design is that it supports a distributed shared address space, rather than using the message passing approach found in most distributed languages. A consequence, few language level primitives are required to support distribution, but the implementation is rather sophisticated. The design [1] was developed after a survey of the literature and in consultation with other language designers. The primitives developed (Figure 1) [3] are minimal but when combined with Haskell's powerful abstractions and existing concurrent and parallel extensions the result is very expressive. The language is a strict superset of both the GpH and Concurrent Haskell language extensions.

Each location is identified by a unique name known as a `PEId`, and methods are provided for detecting the current location, `myPEId`, and listing the available locations, `allPEId`. Resources such as threads are placed at particular locations. These resources are members of the `Immobile` class which indicates that they are stuck at certain location. Such resources can then be located by `owningPE` or have work sent to them by `revalIO`.

We are also considering extending GdH to better support fault-tolerance and have developed a preliminary design [2].

2.2 Distributed Language Implementation

A major focus for this project is its emphasis on developing a practical, reliable, and potentially long-lived system. This is motivated by the growing importance of distributed languages, as outlined in Section 1. The language primitives identified in the design were implemented by extending the GUM runtime system of GpH [Trinder *et al.*, 96]. The resulting runtime system is Sticky-GUM so named due to the techniques used to stick and then locate data and resources.

Sticky-GUM required new shared-memory mechanisms for co-ordinating access and identifying resources. For work sending, the communication subsystem had to be extended and the thread manager adapted to recognise the revalIO threads and for handling exceptions and termination correctly. Finally a set of Haskell libraries were written to provide a clean interface for programmers.

Enabling both the concurrent and parallel components of the runtime to co-exist in harmony also constituted an important component of the implementation effort, with many changes necessary throughout the entire runtime system. The relationship between the three runtime systems, and position of Sticky-GUM, is shown in Figure 2.

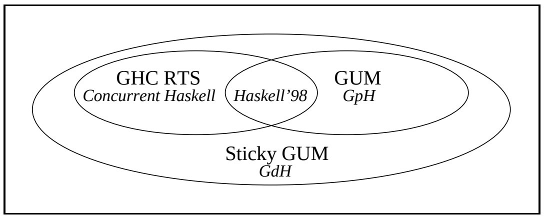

Figure 2: GdH Language & Runtime System.

Figure 3: Distributed Interfaces.

2.3 Distributed Language Evaluation

The objectives addressed by this subsection are:
Construct a realistic distributed application. In particular a multi-user interactive simulation.

Evaluate the suitability of lazy functional languages for implementing real small-scale distributed applications.

To meet these objectives the project team have constructed and measured several realistic small-scale applications, and the following section outlines each application and the evaluation performed. Comparisons are drawn with other programming languages, both object-oriented such as Java [Daconta *et al.* 98], and functional such as ERLANG [Wikstrom, 94], Haskell with Ports [Huch *et al.*, 00], and Eden [Breitinger *et al.* 97]. The comparison considered issues such as total program size, amount of code required to specify distribution, and performance [5, 4, 7]. The new techniques offered by some of the functional languages, such as lazy dynamic communication, were studied in detail [4].

The evaluation was based on a variety of programs ranging from simple test programs such as *Ping* (≈10 lines of code) used to measure network and remote thread creation time [5], to more substantial programs. The substantial programs include *Distributed ted* (≈300 lines of code) that allows multiple users to share and edit text files [3], *Distributed File Server* (≈150 lines of code) that simulates files access using multiple file servers [6], a factory simulator and a multi-user shared map [4].

The Kiteck's factory simulation [Kiteck, 92] was chosen as an exemplar because it was representative of the class of applications that deal with distributed co-operating interactive environments, other examples being distance learning, gaming, etc. Additionally the problem has been well studied and an existing Java implementation [Wilcox, 97] was available for comparison. The conclusions from implementing the system in GdH (Figure 3) can be summarised as follows.

- Equivalence of distribution constructs: both languages were equally expressive in specifying concurrency even though they use slightly different constructs, e.g. in place of Java's RMI, GdH uses `revalIO` for distribution.
- The Eden and GdH functional languages are far more concise: in this case the code was a third of the size with the GdH being ≈500 lines of code. This is due to the use of higher-order functions and the general verbose nature of the the Java syntax.
- Java has superior fault-tolerance and support for open client-server applications: this is unsurprising since we identified these as future work for GdH within our design phase.

The shared multiuser map is the core of a shared-virtual world game. The conclusions for implementing the program in GdH can be summarised as follows.

- Our evaluation strategy mechanisms for parallel co-ordination [Trinder *et al.*, 98] also proved useful for co-ordinating distributed computations.
- The shared, lazy and dynamic communication of GdH was shown to have acceptable performance when compared to the eager approach of copying all of the data at startup or the more traditional explicit Java-like approach of a client requesting parts of the data from the server. Evaluation strategies allowed these three

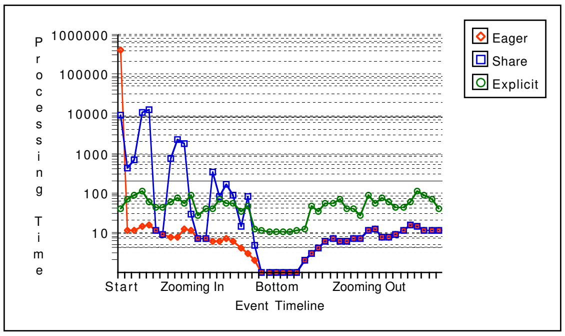

Figure 4: Communication Comparision.

> approaches to be easily implemented and compared evaluated, as reported in [4], and depicted in Figure 4. In short, the eager approach has a huge startup cost, but subsequently minimal map traversal cost; the explicit client/server approach has the relatively constant communication cost; and the shared lazy and dynamic communication approach results in quickly improving performance as more data is transfered, in the end matching the eager approach.

To support this strand of work beyond the current project we have gained a 4-year British Council/DAAD funded travel grant (Project No. 1097), together with partners at Phillips-Universität Marburg, and St Andrews University. The grant *Evaluating Functional Distribution* will evaluate two non-strict distributed functional languages, namely Eden and GdH.

2.4 Other Achievements

The following contributions have been made in areas related to the project but not immediately within its remit. Publications in related areas include [8, 9, 10, 11, 12, 13, 14], and other publications include [15, 16, 17].

In addition to the grant detailed above, we have gained a 4-year British Council/Acciones Integradas travel grant (No. MDR 8472). Our partners are St Andrews University and Universidad Complutense de Madrid in Spain. Entitled *Performance Evaluation of Parallel Functional Programs*, the grant aims to evaluate the performance of the Eden and GpH parallel functional programming languages, and to construct frameworks for reasoning about them.

3 Project Management and Plan Review

3.1 Project Team and Management

The project was directed jointly by Dr. Phil Trinder and Mr. Robert Pointon, the latter being appointed as an experienced Research Associate. Dr Hans-Wolfgang Loidl, a Postdoctoral Research Fellow working primarily on GpH made regular useful contributions. Ben Dale, a 4th year undergraduate, undertook a project constructing a distributed file server in GdH. A steering group comprising Dr Kevin Hammond (St Andrews University) and Prof. Phil Broughton (ICL) was constituted and periodically reviewed the project as detailed below.

3.2 Plan Review

The plan identified the following phases and review activities.

The *GdH Language Design* phase was completed on schedule and reviewed by industrial and academic collaborators.

The start and hence completion of the *GdH Implementation* phase was delayed by approximately 1 month while the GHC runtime system was upgraded from version 3 to 4 in a related project.

The *Simulation Construction* phase took less time than anticipated, and hence we were able to construct and evaluate the additional programs detailed in Section 2.3.

The *Project Review* by the steering group was undertaken in St Andrews on 19th July 2000.

The *Language Evaluation* phase was completed on schedule despite being extended to cover not only Java, but also new DFLs, namely Haskell with Ports and Eden.

3.3 Collaboration

The main project collaborations have been with the following groups.

Industrial: Prof. Simon Peyton Jones' group at Cambridge Microsoft Research Labs; the Software group at Motorala UK Research Labs, Basingstoke; Prof. Phil Broughton, High Performance Systems group at ICL West Gorton.

Academic: Prof. Rita Loogen's Eden group at Phillips-Universität Marburg; Prof Ricardo Pena's group at Universidad Complutense de

Madrid. Dr. Kevin Hammond's group, at St Andrews University; Dr Greg Michaelson's group at Heriot-Watt University.

4 Research Impact and Benefits to Society

Society is adapting to the explosion of networked computers, and demands that these systems use the *latest technology*, are *low cost*, *easy to use*, and always both *reliable* and *available*. Our hypothesis is that the functional paradigm is an appropriate technology for constructing systems to meet these objectives. The high level co-ordination and computation notations mean that programs are shorter, as we show in our evaluation paper [4], thereby reducing the time to market: making the latest technology available at low cost. Reliability is improved because the languages are strongly-typed and automatically manage much of the distributed co-ordination. By implementing GdH and performing an initial evaluation with alternative language technologies, this project has provided preliminary proof of these concepts. Moreover the robust language implementation is a foundation for more thorough investigation of these hypothesis.

The project has contributed to the programming language research community in the following ways. We have described the design and implementation of our language and engaged in discussion with other language designers over alternatives [1, 3]. We have presented a design for a language extension [2]. We have made the language publicly available via the GHC distribution bundle. We have published evaluations comparing GdH with alternative object-oriented and functional distributed languages [5, 4].

5 Explanation of Expenditure

The overall project expenditure is very close (0.03%) to budget (£51816). The Staff, Equipment, Travel and Overhead expenditures are all close to budget. Some £610 of the small Consumables budget (£1039) was not required, giving a large variance (58.7%). Under Staff, in addition to the budgeted research associate, a computer technician was employed part-time to install and support the Beowulf workstation cluster used to evaluate the distributed languages.

6 Further Research and Dissemination

The research has been disseminated by the publications detailed above, most of which have been supported by conference or workshop presentations. The project team have given seminars at Microsoft UK Research Labs, Motorola UK Research Labs, Phillips Universität Marburg (Germany), Universidad Complutense de Madrid (Spain), Johannes Kepler Universität Linz (Austria), St Andrews University, and Heriot-Watt University.

The GdH language implementation has been made available as part of the standard GHC haskell bundle (version 5.0 onwards), announced on appropriate mailing lists, and is supported by a web page: `http://www.cee.hw.ac.uk/ dsg/gdh/`.

The German British Council travel grant (Section 2.3) runs until June 2003, and the Spanish grant (Section 2.4) until December 2001. Based on this project, the team has two EPSRC proposals in preparation.

The *High Level Techniques for Advanced Telecommunications Software* proposal is collaborative work with Motorola Research Labs and aims to investigate the use of distributed functional technology to reduce time to market and increase reliability of telecommunications software.

The *Runtime System Level Fault Tolerance* proposal aims to interpose a new, runtime system level of fault tolerance for distributed language implementations. The key observation is that pure computations can be automatically repeated, and the project builds on our design [2].

Robert Pointon is continuing to research distributed functional languages using GdH and some of the project results will form the core of his PhD Thesis [7]. The plan is to complete the thesis by Christmas 2001. Andre Rauber Du Bois has been recruited from Brazil to start PhD research in October 2001, initially adapting GdH to support decentralised distribution, funded by ORS and James Watt Scholarships. The language will also be used in MSc and undergraduate projects. The publications [7, 18] are currently in preparation.

Project Publications

[1] Trinder, P.W. 1999 (Jul.). Motivation for Glasgow distributed Haskell, a non-strict Functional Language *PDSIA'99 — Parallel and Distributed Computing for Symbolic and Irregular Applications*. Sendai, Japan.

[2] Trinder, P.W., Pointon, R.F., & Loidl, H-W. 2000 (Jul.). Towards Runtime System Level Fault Tolerance for a Distributed Functional Language. Pages 103–113 of: *SFP'00 — Scottish Functional Programming Workshop*. Trends in Functional Programming, vol. 2. University of St Andrews, Scotland: Intellect.

[3] Pointon, R.F., Trinder, P.W., & Loidl, H-W. 2000 (Sept.). The Design and Implementation of Glasgow distributed Haskell. Pages 101–116 of: *IFL'00 — International Workshop on the Implementation of Functional Languages*. Aachen, Germany. LNCS, Springer-Verlag.

[4] Pointon, R.F., S., Priebe., Loidl, H-W., Loogen, R., & Trinder, P.W. 2001 (Feb.). Functional vs Object-Oriented Distributed Languages. *Eurocast'01 — International Conference on Computer Aided Systems Theory*. Palmas de Gran Canaria, Spain. Springer-Verlag.

[5] Trinder, P.W.,Loidl, H-W., & Pointon, R.F. 2001. Parallel and Distributed Haskells. To appear in *Journal of Functional Programming*.

[6] Dale, B. 2001 A GdH Distributed File Server Honours Dissertation, Heriot-Watt University

[7] Pointon, R.F. 2001. The Design, Implementation, and Use of a Distributed Functional Language. *PhD Thesis*. Department of Computing and Electrical Engineering, Heriot-Watt University. In Preparation.

Related Publications

[8] Michaelson, G., Trinder, P.W., & Loidl, H-W. (Eds) 2000. *Trends in Functional Programming*. Bristol, UK: Intellect, ISBN 1-84150-024-0.

[9] Trinder, P.W., Loidl, H-W., & Hammond, K. 2000. Large-scale Functional Applications. *Chapter of: Parallel Functional Programming*. Hammond, K., & Michaelson, G. (Eds): Springer Verlag, ISBN 1-85233-092-9.

[10] Loidl, H-W., Klusik, U., Hammond, K., Loogen, R., & Trinder, P.W. 2000 (Jul.). GpH and Eden: Comparing Two Parallel Functional Languages on a Beowulf Cluster. Pages 39-52 of *SFP'00 — Scottish Functional Programming Workshop*. Trends in Functional Programming, vol. 2. University of St Andrews, Scotland: Intellect.

[11] Baker-Finch, C., King, D.J., & Trinder, P.W. 2000 (Sept.). An Operational Semantics for Parallel Lazy Evaluation. Pages 162–173 of: *International Conference on Functional Programming*. Montreal, Canada.

[12] Trinder, P.W., Loidl, H-W., Barry, E., & Rebon Portillo, A.J. 2000 (Sept.). The Multi-Architecture Performance of the Parallel Functional Language GPH, Pages 739–743 of: *Euro-Par 2000 — Parallel Processing*. Munich, Germany. Bode, A., Ludwig, T., & Wismuller, R. (Eds): Springer Verlag, LNCS 1900.

[13] Loidl, H-W., Trinder, P.W., Hammond, K., Junaidu, S.B., Morgan, R.G.,& Peyton Jones, S.L. 1999 (Oct.). Engineering Parallel Symbolic Programs in GpH. 11(12), Pages 701–752 of: *Concurrency: Practice and Experience*.

[14] Loidl, H-W., Trinder, P.W., & Butz, C. 2001 (Mar.). Tuning Task Granularity and Data Locality of Data Parallel GpH Programs. To appear in *Parallel Processing Letters*

Other Publications

[15] Yang, J., Michaelson, G., & Trinder, P.W. 1999 (Aug.) Explaining Polymorphic Types Through Visualisation. *7th Annual Conference on the Teaching of Computing*. University of Ulster, Northern Ireland.

[16] Yang, J., Michaelson, G., & Trinder, P.W. 2000 (Aug.). Helping Students Understand Polymorphic Type Errors. *LTSN-ICS Annual Conference*. Edinburgh, Scotland.

[17] Trinder, P.W., & Junaidu, S. 2000 (Oct.). Measuring Large Software Systems: An Experience with Naira. *The 3rd Workshop on Information and Computer Science.* King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

[18] Loidl, H-W., Rubio Diez, F., Scaife, N.R., Priebe, S.M., Hammond, K., Klusik, U., Loogen, R., Michaelson, G.J., Pena Mari, R.,Rebon Portillo, A.J., & Trinder, P.W. Implicit Approaches to High Level Parallel Functional Programming (working title). *Higher-order and Symbolic Computation.* In preparation.

References

[Blau & Rooth 98] , Blau, S. & Rooth, J. 1998. AXD301 - A New Generation ATM Switching System, *Ericsson Review*, 75(1).

[Breitinger *et al.* 97] Breitinger, S., Loogen, R., Ortega Mallén, Y., & Peña Mari, R. 1997. Eden — The Paradise of Functional Concurrent Programming. *Glasgow Workshop on Functional Programming.*

[Chakravarty, 98] Chakravarty, M., Guo, Y. & Khler, M. 1998 Distributed Haskell: Goffin on the Internet, *Fuji International Symposium on Functional and Logic Programming.*

[Daconta *et al.* 98] Daconta, M.C., Saganich, A., Monk, E., & Snyder, M. 1998. Java 1.2 and JavaScript for C and C++ Programmers. *John Wiley & Sons.*

[Hanus 00] Hanus, M. 1999. Distributed Programming in a Multi-Paradigm Declarative Language, *PPDP'99 — Principles and Practice of Declarative Programming*, Paris, France.

[Haridi*et al.*, 97] Haridi, S.,van Roy, P. & Smolka, G. 1997 An Overview of the Design of Distributed Oz, PASCO 97 — International Symposium on Parallel Symbolic Computation, New York, USA.

[Holyer *et al.*, 98] Holyer, I., Davies, N. & Spiliopoulou, E. 1998 Distribution in a Demand Driven Style, *International Workshop on Component-based Software Development in Computational Logic.*

[Huch *et al.*, 00] Huch, F. & Norbisrath, U. 2000 (Sept.). Distributed Programming in Haskell with Ports. Pages 101–116 of: *IFL'00 — International Workshop on the Implementation of Functional Languages.* Aachen, Germany.

[Kiteck, 92] Kiteck, P. 1992 (Sept.). Analysis of Component Interaction in a Distribution Facility using Simulation. Pages 633–638 of: *EUROSIM'92 — Simulation Congress.*

[Peyton Jones *et al.*, 98] Peyton Jones, S.L., Hughes, J., Augustsson, L., Barton, D., Boutel, B., Burton, W., Fasel, J., Hammond, K., Hinze, R., Hudak, P., Johnsson, T., Jones, M., Launchbury, J., Meijer, E., Peterson, J., Reid, A., Runciman, C., & Wadler, P. 1999 (Feb.). Haskell 98: A Non-strict, Purely Functional Language. *Electronic document available on-line at* `http://www.haskell.org/`.

[Trinder *et al.*, 96] Trinder, P.W., Hammond, K., Mattson Jr., J.S., Partridge, A.S., & Peyton Jones, S.L. 1996 (May). GUM: a portable implementation of Haskell. *PLDI'96 — Programming Language Design and Implementation.* Philadephia, USA.

[Trinder *et al.*, 98] Trinder, P.W., Hammond K., Loidl H-W., &Peyton Jones S.L. 1998 (January) Algorithm + Strategy = Parallelism. *Journal of Functional Programming*, 8(1), pp 23-60.

[Wikstrom, 94] Wikstrom, C. 1994 (Sept.). Distributed programming in Erlang. *PASCO'94 — International Symposium on Parallel Symbolic Computation.* Linz.

[Wilcox, 97] Wilcox, P.A. 1997. Object-Oriented Factory Simulation in Java. *Master's thesis.* Department of Computer Science, Edinburgh University.

[Wojciechowski, 00] Wojciechowski,P. 2000. Nomadic Pict: Language and Infrastructure Design for Mobile Computation, PhD Thesis, University of Cambridge.