ELSEVIER

Future Generation Computer Systems 17 (2001) 911–918

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.nl/locate/future

Metacomputing across intercontinental networks

S.M. Pickles [a,*], J.M. Brooke [a], F.C. Costen [a], Edgar Gabriel [b], Matthias Müller [b], Michael Resch [b], S.M. Ord [c]

[a] *Manchester Computing, The University of Manchester, Oxford Road, Manchester M13 9PL, UK*
[b] *High Performance Computing Center, Stuttgart, Allmandring 30, D-70550 Stuttgart, Germany*
[c] *NRAL, Jodrell Bank, Macclesfield, Cheshire SK11 9DL, UK*

Abstract

An intercontinental network of supercomputers spanning more than 10 000 miles and running challenging scientific applications was realized at the Supercomputing '99 (SC99) conference in Portland, OR using PACX-MPI and ATM PVCs. In this paper, we describe how we constructed the heterogeneous cluster of supercomputers, the problems we confronted in terms of multi-architecture and the way several applications handled the specific requirements of a metacomputer. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Metacomputing; High-performance networks; PACX-MPI

1. Overview of the SC99 global network

During SC99 a network connection was set up that connected systems in Europe, the US and Japan. For the experiments described in this paper, four supercomputers were linked together.

- A Hitachi SR8000 at Tsukuba/Japan with 512 processors (64 nodes).
- A Cray T3E-900/512 at Pittsburgh/USA with 512 processors.
- A Cray T3E-1200/576 at Manchester/UK with 576 processors.
- A Cray T3E-900/512 at Stuttgart/Germany with 512 processors.

Together this virtual supercomputer had a peak performance of roughly 2.1 Teraflops. The network was based on:

- national high speed networks in each partner country,
- the European high speed research network TEN-155,
- a transatlantic ATM connection between the German research network (DFN) and the US research network Abilene,
- a transpacific ATM research network (TransPAC) that was connecting Japanese research networks to STAR-TAP at Chicago.

Details of the network configuration can be seen in Fig. 1. Using ATM, PVCs were set up with a bandwidth of 10 MBit/s in the European networks. On all other connections bandwidth was shared with other users. The sustained bandwidths and the latencies between sites were as described in Table 1.

2. Problems of multi-architecture

Heterogeneous metacomputing introduces some problems that are similar to those well known from

* Corresponding author.

PII: S0167-739X(01)00032-2

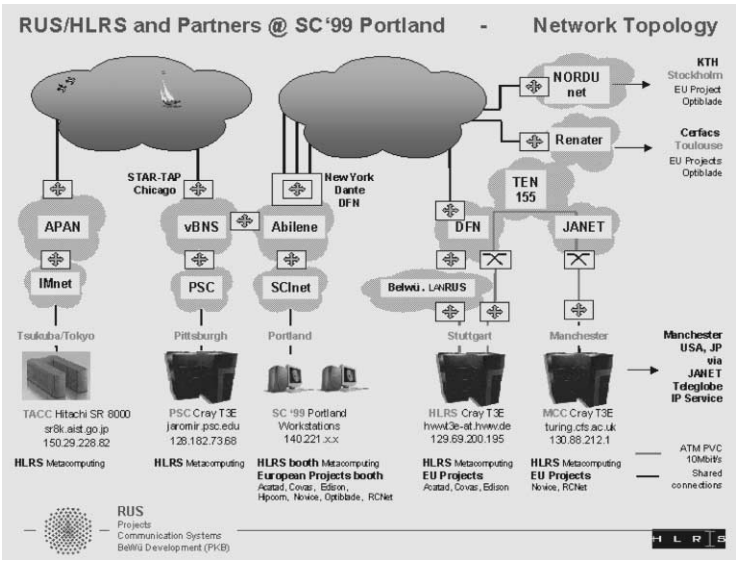

Fig. 1. Network configuration as set up during SC99.

cluster computing and some that are very specific [2]. The most important ones are

- different data representations on each system,
- different processor speeds on each system,
- different communication speeds for internal messages on each system,
- different communication speeds of messages internal to a system and messages between systems,
- lack of a common file system,
- lack of resource management.

The problem of load imbalance due to different processor speed has to be tackled on the application level. So far there is no standard way for an application to determine the speed of a processor. Moreover, it is difficult for a given application to get information about its sustained performance. An approach that is currently being investigated is to start the application with a standard distribution. The application would then collect information about the amount of load imbalance. Based on these data and on the characteristics of the network connection, the application would then decide how much effort should be spent to redistribute the load. Such an approach, however, is based on the assumption that the basic parameters (processor load, network speed, application load) remain constant. Allowing these parameters to vary — as is necessary if working in a non-dedicated environment or using adaptive methods — will further complicate the problem.

Table 1
Network bandwidths and latencies between sites during SC99

	Latency (ms)	Bandwidth (MBit/s)
HLRS-CSAR	20	6.4
HLRS-PSC	78	1.6
CSAR-PSC	58	4.0

The problem of scheduling resources is a continuing research topic. For metacomputing it is necessary to make sure that resources are available concurrently. That means that the scheduling of several systems and of networks must be coordinated. No automatic mechanism was available to us at SC99. The problem of distributed file systems was not tackled in our approach. We assumed that data would be in place whenever necessary.

All other problems relate mainly to communication and have to be tackled by the message passing library that is used. To specially focus on the problems of metacomputing HLRS has developed an MPI library called PACX-MPI [1,3] that couples big systems into a single resource from the communication point of view. Based on the experience of several other projects the

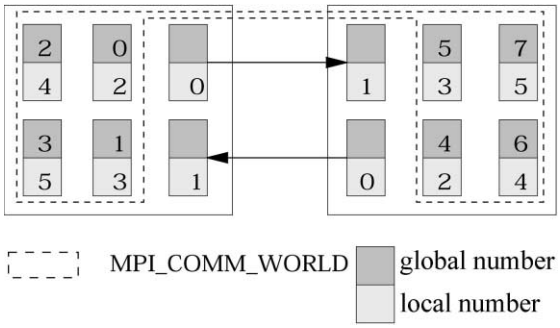

Fig. 2. Basic concept of PACX-MPI. Two systems integrated into one single MPI_COMM_WORLD.

library relies on three main concepts:

- *Single MPI_COMM_WORLD for an application*. When starting an application with PACX-MPI a local MPI_COMM_WORLD is created on each system in the metacomputer. These are combined by PACX_MPI to form a single MPI_COMM_WORLD.
- *Usage of communication daemons*. On each system in the metacomputer two daemons take care of communication between systems. This allows bundling of communication and to avoid having thousands of open connections between processes. In addition it allows handling of security issues centrally.
- *Splitting of internal and external communication*. For all internal communication PACX-MPI makes use of the native library of the system. It only implements communication for external messages, that is messages that have to be sent between systems. This has the advantage that optimized communication inside a system can be combined with portable communication protocols between systems.

In Fig. 2 global numbering denotes the global MPI_COMM_WORLD, while local numbering corresponds to the local communicators. Processes 0 and 1 on each system are acting as daemons for external communication.

3. Use of experimental data

In this section, we describe a novel use of metacomputing, namely the processing of data from an experimental facility (in this case the Jodrell Bank radio telescope). This presents particular challenges to our metacomputer since the distribution of the data makes severe demands on the intercontinental bandwidth. This experiment involved coupling of the three T3E machines only.

3.1. The Jodrell bank de-dispersion pulsar search code [6]

The application searches for pulsars in the radio signal from a globular cluster. The observed radio signal from a pulsar manifests at the telescope as a periodic increase in broadband radio noise. In order to observe the pulse with a high signal to noise ratio, we need to observe across a wide band of radio frequencies. As the space between the earth and the pulsar (the interstellar medium) is slightly charged it is dispersive and therefore different frequencies propagate at different velocities. To reconstruct the pulsar signal it is necessary to first determine the relationship between a given radio frequency and the time lag in the arrival of the pulse (called the dispersion measure, DM) and then use this to reconstruct the signal. Since a globular cluster is a compact object, the DM will be approximately constant for all pulsars in the cluster and reconstructing at several trial DMs in a narrow range yields an effective method of searching.

3.2. Implementation on the metacomputer

The total data from a continuous sequence of measurements from the telescope is first split into lengths at least twice as long as the dispersive delay time across the observing bandwidth. Each length is then subjected to a Fourier transform and multiplied by the Fourier transform of the inverse filter. The resultant data is then transformed back into the time domain and processed to produce a de-dispersed time series. The next length is then processed, and the resultant series are then concatenated together. In practice, the input data has to be forward transformed only once, but transformed back into the time domain as many times as we have trial DMs.

This process defines the challenge for the metacomputer. If the data is read onto one T3E (in this case at Manchester) then subsets of the data have to be sent to the remote T3Es via a connection, that is two orders of magnitude slower than the inter-processor connection

within a T3E. This might seem to invalidate the whole metacomputing concept, but the requirement that many trial DMs be applied to each length means that we can perform sufficient processing on the remote machines to cover the cost of transmission. Moreover, since our ultimate goal is real-time processing of experimental data (i.e. processing directly after the observations via network links to a large computing facility), we would find that the remote host has far more processing power than the machines onto which the data is initially read. In this context it is important that we operate with subsets of the observations, so that the FFT processing can be applied to such chunks independently and the remote machines can begin processing as soon as they receive a chunk of data that is sufficiently large to satisfy the constraint imposed by the maximum dispersive delay time.

In the SC99 experiment the observational data was pre-loaded onto the Manchester T3E. The transmission of data to the remote T3Es for processing is an essential part of this application.

3.3. Work distribution model

The original intention was to adopt a master–slave model in which one (or more) master process reads the disk and distributes work to slave processes. This approach is highly attractive because it naturally furnishes an effective method of load-balancing. The main drawback is that, in the absence of complicated and time-consuming preventive steps, no one processor would get two contiguous chunks of the global time series, thus the occurrence of gaps between work units would be maximized, a fact which would detract from the integrity of the results. Instead, we adopted a client–server model. A handful of server nodes poll for requests for data from clients, and the bulk of intelligence resides with the clients who do the actual work. By allocating a single, contiguous portion of the global data set to each worker, gaps occur only at processor boundary, just as in the original code. By letting each processor keep track of where it is up to in the global data set, the impact on the logic and structure of the code is minimized. The disadvantage of the client–server model adopted here is that we are compelled to tackle the load-balancing problem statically, taking into account *estimates* of the MPI bandwidth, processor speed, disk performance and so on.

3.4. Static load-balancing algorithm

Let N be the number of hosts, and let n_i be the number of client processors to use on host i, discounting the extra two processors required by PACX-MPI and those processors on host 1 which have been assigned to server duties. Denote the bandwidth from host 1 (where the input data resides) to host i by β_i. The rate at which data can be read from disk on host 1 is denoted by r; it is assumed that this rate is approximately independent of the number of processors accessing the disk at any one time.

The size of one record is denoted by w. The computational time required to process one record of data on host 1 is determined experimentally and denoted by t_τ. The time to process the same quantity of data on other hosts is estimated by multiplying t_τ by the ratio $p_1{:}p_i$, where p_i is the peak speed in Mflops of the processors on host i.

Any job will process a total of v records. The load-balancing problem is to determine the proportion $v_1{:}\ldots{:}v_n$, where $\sum_{i=1}^{N} v_i = v$, in which to distribute data to the hosts.

The elapsed wall-clock time t_i to process v_i records on host i is estimated by $t_i = v_i t_{\mathrm{proc}}(i)/n_i + n_i t_{\mathrm{wait}}(i)$, where $t_{\mathrm{wait}}(i) = (w/\beta_i + w/r)$ is the time that a client processor has to wait for a single work unit, and $t_{\mathrm{proc}}(i) = t_\tau p_1/p_i$ is the time that it takes the client to process it. The first term in the expression for t_i is the time required for computation. The second term is essentially twice the *start-up time*, i.e. the time elapsed until all processors on the host have received their first work unit. A similar cost is paid at the end (*run-down time*) as some processors will finish their allocated work before others.

The condition used to balance the work load is that all hosts finish at the same time, $t_1 = t_2 = \cdots = t_N$. This leads to a linear system with $N + 1$ equations and $N + 1$ unknowns $(v_1, \ldots, v_n, t)$, expressed in terms of the quantities $a_i = (t_\tau p_1)/(n_i p_i)$ and $b_i = n_i w(1/\beta_i + 1/r)$.

$$\begin{pmatrix} a_1 & \cdots & 0 & -1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_N & -1 \\ 1 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_N \\ t \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_N \\ v \end{pmatrix}$$

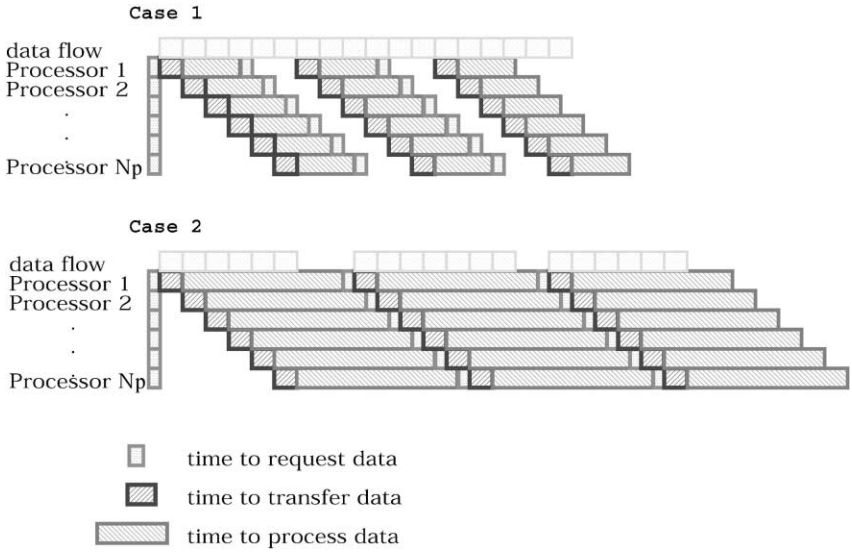

Fig. 3. Data flow. In case 1, bandwidth is saturated and each processor experiences dead time waiting for its next work unit. In case 2, the ratio of communications to computations is more favorable and there is no dead time between work units.

The validity of the above depends on the implicit assumption that no client processor experiences *dead time* waiting for other clients to receive their data. Fig. 3 illustrates this graphically. In case 1, client processors have to wait for their next record because the bandwidth is too small. In case 2, the number of trial DMs has been increased so that processors spend sufficiently long on one record so that the next record can be supplied without dead time.

4. Computational fluid dynamics

Another application used during SC99 was a CFD-code called URANUS (Upwind Algorithm for Nonequilibrium Flows of the University of Stuttgart) [4]. This program has been developed to simulate the reentry phase of a space vehicle in a wide altitude velocity range. The reason why URANUS was tested in such a metacomputing environment is that the non-equilibrium part has been finished and will be parallelized soon. For this version of URANUS, the memory requirements for a large configuration exceeds the total memory which can be provided by a single machine today.

URANUS consists mainly of three parts. The first part (pre-processing) reads the input data and distributes all information to the application nodes. Since the code handles different blocks in the mesh, we could distribute in an ideal case each block on a machine. Thus, the pre-processing step remained local on each host. [1] The second part is the processing part. For the single block code, different strategies were developed to hide latency [4], but these methods were not yet adapted to the multiblock code. Finally, the third part is the post-processing, including some calculations and the writing of the results to disk. For this part, the same techniques could be used as in the pre-processing part.

During SC99 we were able to simulate the crew-rescue vehicle (X-38) of the new international space station using a four block mesh with a total of 3.6 million cells. For this, we used 1536 nodes on the three Cray T3Es.

5. Molecular dynamics

A further application that was adapted for metacomputing is a molecular dynamic program for short range interaction [5] simulating granular matter. The

[1] This did not work for large simulations, since we had to distribute four blocks on three Cray T3Es.

parallel paradigm applied here is domain decomposition and message passing with MPI (message passing interface). Therefore every CPU is responsible for a part of the domain and has to exchange informations about the border of its domain with its adjacent neighbors. Static load balancing can be implemented by assigning domains of different size to each processor. To reduce the amount of load imbalance between the T3Es and the SR8000, the nodes of the SR8000 were used as eight processors.

5.1. Latency hiding

As described in Section 1, the latency of a metacomputer is in the range of milliseconds and thus several orders of magnitudes larger than in todays parallel computers. For a tightly coupled algorithm like an MD simulation latency hiding has to be implemented to get reasonable performance. To achieve this, it was decided to group the communication in two sections. First, the communication for the first dimension is initiated by calls to MPI_Isend. In a second stage this communication is completed and the calls for all remaining dimensions are performed, allowing a partial overlap of communication with computation. In this application the force calculation for particles that interact only with particles of the core domain is performed between the first and the second stage. If the calculation for one particle takes about 100 μs one needs around 750 particles to hide the latency. As soon as latency hiding is implemented this is no major restriction.

5.2. Bandwidth

As a first approach latency hiding also helps to hide the increased communication duration due to a small bandwidth. Increasing the number of particles does, however, not only increase computing time but also the amount of data to be sent. For a latency of 74 ms and a bandwidth of 5 MBit/s in both directions even with particles consisting of only 24 bytes data (e.g. three doubles representing a three-dimensional position), there is a break even between latency and communication duration above 15 000 particles, a value easily reached in real applications. In the problem treated here, a particle consisted of 10 floating point numbers: three vectors position, velocity and the force acting on the particle plus one scalar variable for the particles radius. The force, however, is local information, because it is calculated on every processor. It is therefore not necessary to transmit this information. In addition, the force is a function of the position and size of the particles. Thus, the velocity only needs to be transmitted whenever a particle crosses the domain between two processors, and not if the information about the particle is only needed to calculate the interaction. With this considerations, the amount of required data per particle can be reduced from ten to about four floating point numbers. In addition to the compression techniques offered by PACX-MPI it reduces the bandwidth requirement significantly and is a key point for efficient metacomputing.

6. Conclusion and future plan

We have described the realization of a global metacomputer which comprised supercomputers in Japan, the USA, Germany and the UK. The network spanning 10 000 miles and several research networks was optimized by using either dedicated PVCs or improved routes. During SC99, the usability of this metacomputer has been demonstrated by several demanding applications. This takes the concept of a global metacomputer forward in two major ways. Firstly, we have shown that it is possible to link together different types of machine, in this case Cray-T3E and Hitachi SR8000. Secondly, we have shown that even data-intensive applications, such as processing output from experimental facilities, can be successfully undertaken on such a metacomputer.

In the future we hope to extend the cooperation between the participating sites to incorporate the experiences gained into both the future development of the PACX-MPI library and the application codes themselves. Furthermore, we plan to extend the Jodrell bank de-dispersion pulsar search code so that we can process experimental data as the experiment is in progress rather than transferring to storage media for post-processing. This could be vital: if the experiment needs recalibration or intervention, this can be detected immediately rather than wasting precious time on a valuable resource; if results can be interpreted as they are being produced the experimental regime can

be adapted accordingly, again maximizing the usage made of the facility.

Acknowledgements

The authors would like to thank Tsukuba Advanced Computing Center, Pittsburgh Supercomputing Center, Manchester Computing Center, High-performance Computing Center Stuttgart, BelWü, DFN, JANET, TEN-155, Abilene, vBNS, STAR-TAP, TransPAC, APAN and IMnet for their support in these experiments.

References

[1] E. Gabriel, M. Resch, T. Beisel, R. Keller, Distributed computing in a heterogeneous computing environment, in: V. Alexandrov, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springer, Berlin, Lecture Notes in Computer Science, 1998, pp. 180–188.
[2] M. Resch, D. Rantzau, R. Stoy, Metacomputing experience in a transatlantic wide area application testbed, Future Gener. Comp. Syst. 15 (5–6) (1999) 807–816.
[3] M.A. Brune, G.E. Fagg, M. Resch, Message-passing environments for metacomputing, Future Gener. Comp. Syst. 15 (5–6) (1999) 699–712.
[4] T. Bönisch, R. Rühle, Adapting a CFD code for metacomputing, in: C.A. Lin, et al. (Eds.), Parallel Computational Fluid Dynamics, Development and Applications of Parallel Technology, Elsevier, Amsterdam, 1999, pp. 119–125.
[5] M. Müller, Molecular dynamics with C++: an object oriented approach, in: Proceedings of the Workshop on Parallel Object-oriented Scientific Computing, Portugal, 1999.
[6] S. Ord, I. Stairs, F. Camilo, Coherent de-dispersion observations at Jodrell bank, in: M. Kramer, N. Wex, R. Wielebinski (Eds.), ASP Conference Series 202: IAU Colloquium 177 on Pulsar Astronomy — 2000 and beyond, 2000.

Stephen Pickles has been involved in both commercial and scientific computing since 1982 when he joined ICL (Australia) as trainee programmer. He graduated in physics from Macquarie University in 1994, where he won the university medal. He obtained his PhD in lattice QCD from the University of Edinburgh in 1998. He is currently the leader of the CSAR Technology Refresh Team at the University of Manchester. He has publications in quantum optics, lattice QCD, and grid computing. His current research interests are in parallel and distributed computing techniques and their application to scientific problems.

John Brooke graduated in mathematics from the University of Manchester in 1973 and gained a Post-Graduate certificate in education in 1974. From 1974 to 1990 he taught mathematics to children aged between 7 and 16, specializing in work with pupils from disadvantaged social and economic backgrounds. In 1990, he joined the University of Manchester to develop a training programme in high-performance computing and completed a PhD in mathematics in 1997. He currently leads a team of researchers and consultants working on projects to develop grid computing on local, national and global scales. He is also active in research in astrophysics and nonlinear mathematics, with special reference to the study of intermittency and symmetry-breaking in rotating systems. He is a Fellow of the Royal Astronomical Society and an honorary lecturer in mathematics at the University of Manchester. More details of his research can be found at http://www.csar.cfs.ac.uk/staff/brooke.

F.C. Costen Fumie Costen received the B.E. and M.E. degrees in electronics from Kyoto University, Japan in 1991 and 1993, respectively. In 1993 and 1996, she joined respectively Optical and Radio Communications Research Laboratories and Adaptive Communications Research Laboratories in Advanced Telecommunication Research International to continue her work on the direction-of-arrival estimation. In 1998, she joined Manchester Computing in the University of Manchester and carried out her research on Metacomputing. Since 2000, she has been a lecturer in the Department of Computer Science at the University of Machester. She is widening her research area from the DOA estimation to the signal processing for the wireless mobile telecommunication with CDMA/OFDM.

Edgar Gabriel received his diploma degree at the University of Stuttgart in 1998 in mechanical engineering. Since June 1998 he is a member of the Parallel Computing Department at the High-performance Computing Center Stuttgart (HLRS). There, he is involved in the metacomputing activities, and coordinating the development of PACX-MPI. Furthermore, he is responsible for the technical management of several national and international projects. Since January 2001, he is leading the working group Parallel and Distributed Systems.

Matthias Mueller is now a research scientist at HLRS. He received his diploma degree in physics at the University of Stuttgart in 1996. His master thesis about quasicrystals was carried out at the department of theoretical and applied physics. After his master thesis he wrote his PhD thesis at the Institute of Computer Applications/Stuttgart about fast algorithms for particle simulations. During this period he started to work in a metacomputing project where he participated with his applications. His scientific interests are metacomputing with a focus on applications, physics on high-performance computers and object oriented programming for parallel computing.

Michael M. Resch received his diploma degree in technical mathematics from the Technical University of Graz/Austria in 1990. He presented his PhD thesis about "Metacomputing for engineering applications" at the University of Stuttgart/Germany in 2001. From 1990 to 1993 he was with JOANNEUM RESEARCH — a leading Austrian research company. Since 1993 he is with the High-performance Computing Center Stuttgart (HLRS). Since 1998 he is the head of the Parallel Computing Group of HLRS and since 2000, head of HLRS. His current research interests include parallel programming models, metacomputing and numerical simulation of blood flow. Michael M. Resch was and is involved in a number of European projects like CAESAR, HPS-ICE, METODIS, DECAST and DAMIEN and is responsible for the international collaborations of the HLRS.

Stephen Ord has obtained an honours degree in Physics from Leicester University; a Masters degree in Astronomy from the University of Sussex and has recently completed a PhD in Radio Astronomy at the University of Manchester. He is currently working as a research fellow in the Astrophysics and Supercomputing Department of Swinburne University of Technology in Melbourne, Australia.