The FUGU Scalable Workstation Prototype

Kenneth Mackenzie, Matthew Frank, Silvina Hanono
John Kubiatowicz, Victor Lee and Jon Michelson*
Alewife Group, MIT Lab for Computer Science
{kenmac,mfrank,silvina,kubitron,wklee,jonm}@cag.lcs.mit.edu

1 Introduction

A scalable workstation combines the communication mechanisms of scalable multiprocessors with protection features required to isolate applications in a general purpose, multiuser workstation. Our ideal workstation is *scalable* in the sense that the communication substrate and mechanisms are physically scalable, provides *multimodel* communication facilities, i.e. support for both shared memory operations and user-level message passing operations, supports *multiuser* operation with sufficient protection mechanisms for general-purpose applications and still offers *high performance* to the user by exposing hardware resources (such as the network) directly to the user.

Providing both user-level messaging with hardware-supported shared memory is desirable because it allows applications to combine the efficiency of message passing, for bulk data transfer and combining data with synchronization, and the flexibility and convenience of shared memory for fine-grained and dynamic computations. The challenge is to provide hardware and software mechanisms for general-purpose, multiuser operation, i.e. protection and virtual memory, while retaining sufficient user access to the hardware to provide parallel application performance equivalent to running on dedicated, single-user hardware. Our approach is to retain user access to hardware features (including the network interface and interrupt handling) and devise means for the kernel to detect user misbehavior and recover control [MKAK94].

As a demonstration system and expermental platform, we are constructing a prototype scalable workstation, FUGU, based on the Alewife machine, an existing, single-user multiprocessor with efficient, multimodel communication mechanisms [ABC+95]. FUGU is being built by modifying the Alewife hardware and adding a Exokernel-style operating system [EKO94]. This abstract outlines the design and implementation status of our prototype system.

*We are advised by Anant Agarwal and supported in part by ARPA contract N00014-94-1-0985, by NSF Experimental Systems grant MIP-9012773, and by an NSF Presidential Young Investigator Award to Prof. Agarwal. Address: Room 632, 545 Technology Square, Cambridge MA 02139.

Communication Mechanism	Translation/Protection Mechanism
Shared memory	Memory mapping and page-level protection via TLB.
Short messages	GID-stamped messages. Receive handler timeout. Rudimentary second network.
Messages w/DMA	Uses the processor's TLB. Frames implicitly locked.

Table 1: Translation and protection for each communication mechanism.

2 Protection and Translation

The communication mechanisms of the Alewife machine can be classed into three categories: shared memory, short messages and bulk transfer messages. Bulk transfer is implemented in Alewife by requesting DMA at the sending or receiving end of a message. FUGU applies protection and translation to each communication mechanism using the techniques listed in Table 1 [MKAK94]. The prototype implements these protection and translation mechanisms as described in the next section.

3 Prototype System

The protection and translation features of the FUGU prototype are implemented in four components of hardware/software: (1) a User Communication Unit (UCU) implemented in an FPGA added to the Alewife chipset, (2) small changes implemented in the respin of the Alewife Communication and Memory Management Unit (CMMU), (3) code restrictions enforced by a trusted compiler/loader and (4) protection checks made in the kernel of the exokernel operating system, "Glaze".

The UCU includes a Translation Lookaside Buffer (TLB) and a simple second network. The TLB implements translation and protection for virtual memory, supporting shared memory and bulk transfer communication mechanisms. The second network is reserved for the supervisor to permit the system to guarantee forward progress of messages without discarding packets when the user has blocked the primary network. The TLB in the UCU translates all addresses from the pro-

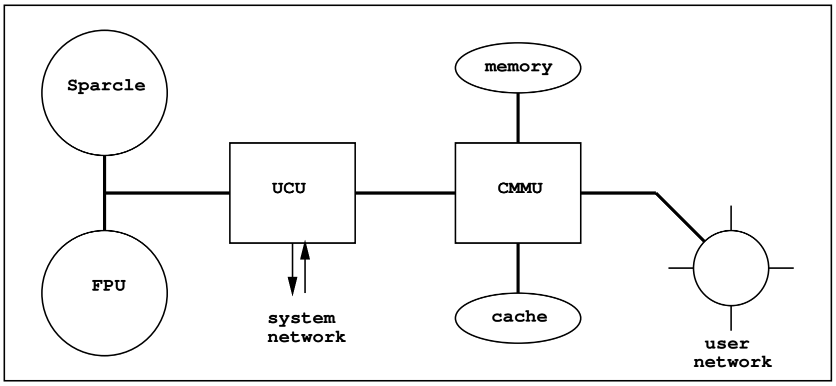

Figure 1: The FUGU node chipset includes the Alewife components (Sparcle IPU, SPARC FPU, Alewife CMMU, Caltech MRC network, 64KB unified cache and 32MB DRAM) and the FUGU UCU. The UCU adds the TLB and a rudimentary second network for the supervisor.

cessor before applying them to the CMMU (Figure 1), placing the UCU in the critical path of the system and thus introducing a severe limit on the performance of the prototype. An ideal FUGU system would integrate the TLB with the processor, as is commonly done. The first cut UCU will employ a Xylinx 4025 FPGA part. We expect to implement a TLB with 128 entries and 4-way associativity and a secondary network with approximately 1MB/sec bandwidth in a ring topology.

The Alewife CMMU respin includes, beyond bug fixes, several minor changes that will be useful in the FUGU system. Chief among these changes is the inclusion of a mechanism for providing atomicity to user code and user interrupt handlers with respect to network interrupts [BCL+95]. The respin includes a specialized timer to allow the system to recover control from a misbehaving user application.

For expediency in the prototype, the FUGU compiler/loader system will be trusted to prohibit certain instructions user mode and to provide protected code sequences. For example, messages are tagged with a group identifier (GID) that must be securely stamped on all messages at the sender and checked by all messages at the receiver.

Finally, protection checks not made in hardware or trusted to the compiler are made by the operating system kernel, "Glaze". For instance, the kernel software implicitly locks page frames in use for DMA by delaying certain virtual memory operations until DMA completes. The FUGU goal of providing communication resources directly to user code meshes naturally with the exokernel notion that the sole function of the kernel is the secure multiplexing of physical resources. We expect the implementation of FUGU and Glaze to yield insights into exokernels and vice versa.

4 Conclusion and Status

The FUGU prototype is an experimental implementation of a scalable workstation, combining multimodel communication mechanisms with support for protection and virtual memory. At this time the hardware work is well along with the CMMU respin and FUGU PCB board expected to be completed this summer. The UCU design is mostly operational in verilog. The operating system kernel and initial libraries currently provide crude uniprocessor functionality while the multiprocessor support for the operating system is in design. Simulation is currently performed on a version of the Alewife simulator modified to implement the UCU. Work on a new, fast simulator is in progress.

[ABC+95] Anant Agarwal, Ricardo Bianchini, David Chaiken, Kirk Johnson, David Kranz, John Kubiatowicz, Beng-Hong Lim, Ken Mackenzie, and Donald Yeung. The MIT Alewife Machine: Architecture and Performance. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture (ISCA'95)*, June 1995.

[BCL+95] Eric Brewer, Fred Chong, Lok Liu, Shamik Sharma, and John Kubiatowicz. Remote Queues: Exposing Message Queues for Optimization and Atomicity. In *To appear in Proceedings of the Symposium on Parallel Algorithms and Architectures (SPAA'95)*. ACM, 1995.

[EKO94] D.E. Engler, M.F. Kaashoek, and J. O'Toole. The operating system as a secure programmable machine. In *Proc. of the 6th ACM SIGOPS European Workshop*, Wadern, Germany, Sept. 1994.

[MKAK94] Ken Mackenzie, John Kubiatowicz, Anant Agarwal, and M. Frans Kaashoek. FUGU: Implementing Protection and Virtual Memory in a Multiuser, Multimodel Multiprocessor. Technical Memo MIT/LCS/TM-503, October 1994.