SUNMOS/PUMA

Stephen Wheat (Sandia), Barney Maccabe (UNM), Rolf Riesen (UNM/Sandia), David van Dresser (UNM), and Mack Stallcup (UNM/Intel)

SUNMOS—Sandia/UNM OS.

- Portable (written in C)
- Vertex clone
- Single tasking
- nCUBE implementation completed in December 1991
- Ported to the Paragon April to August 1993

PUMA—Performance-oriented, User-managed, Messaging Architecture

- Design started in January of 1992
- Enhanced communication facilities
- Improved protection strategies
- Multi-tasking
- nCUBE implementation started in January of 1993

SUNMOS/PUMA Features

Application memory

- communication space (buffer for messages)
- application data space (includes heap and stack)

Message passing

- synchronous and asynchronous
- messages sent to communication space or data space

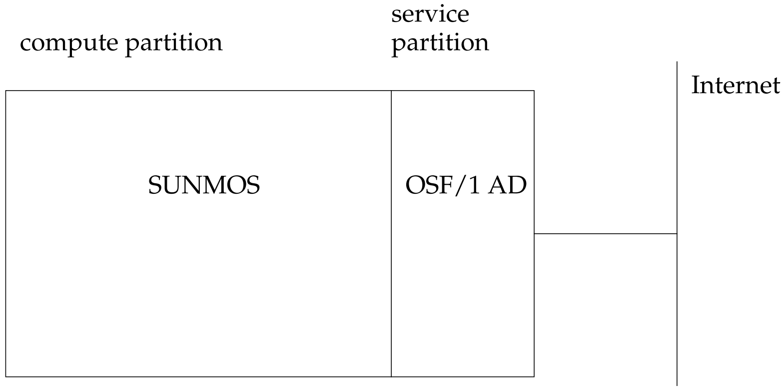
Compatibility libraries

- nCUBE—nwrite, nread, whoami
- NX—isend, irecv, csend, crecv, mypart
- FORTRAN and C on nCUBE
- C and C++ on Paragon

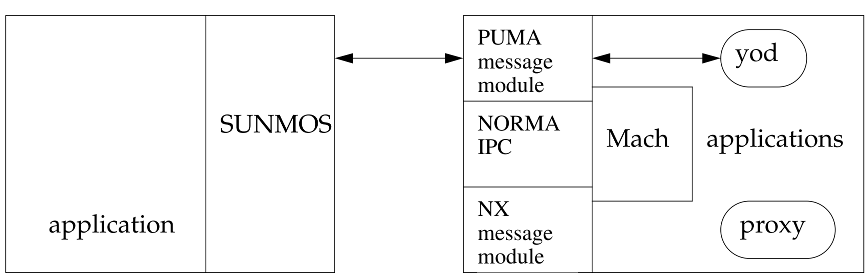
Standard I/O library

SUNMOS Implementation Strategy

Service partition/Compute partition



OSF/1 AD and SUNMOS Interactions



OSF/1 AD and SUNMOS Interaction

OSF/1 AD on the service partition

- PUMA message module
- yod runs as an application on service nodes

SUNMOS on the compute partition

- OSF (PUMA) messages
- inter-process messages

YOD

SUNMOS version of xnc (nCUBE's generic host program)

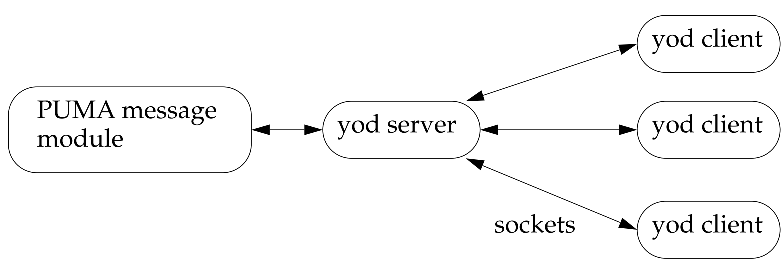
Mesh allocation

Application loading

Host services (e.g., sequential file access)

Current limitation

- one yod per service node
- fix requires Internet sockets (bug in local sockets)



Porting SUNMOS to the Paragon

April ‘93

- started working on SUNMOS port

May ‘93

- SUNMOS runs as a diagnostic

June ‘93

- lock free communication
- multi-kernel boot

July ‘93

- yod interacting with PUMA message module

August ‘93

- PCTH runs on 416 nodes (zuni)
- enhanced application support (showmesh, improved yod, getpcb, getcomm)

SUNMOS Performance

Memory

- Currently, 232 Kbytes
- 98% of 16 Mbyte node available to application programs
- Text, data, stack, heap and communication space

Communication bandwidth

- 98% of available bandwidth

Communication latency

- ~50 microseconds in single processor mode
- ~30 microseconds in message coprocessor mode

SUNMOS Messaging Bandwidth

Message Coprocessor Mode

Message size in bytes	Time in seconds	Throughput (million bytes/sec)
1024	0.00005271	19.4271
2048	0.00004830	42.4017
4096	0.00005638	72.6499
8192	0.00008161	100.3799
16384	0.00012802	127.9800
32768	0.00022447	145.9794
65536	0.00042364	154.6974
131072	0.00082148	159.5559
262144	0.00158163	165.7429
524288	0.00320914	163.3734
1048576	0.00640430	163.7300
2097152	0.01277103	164.2117
4194304	0.02509769	167.1191

High Performance Messaging

Minimize overhead

- remain in kernel for duration of transfer
- use maximum transfer size

16Kbyte DMA (LTU) transfers for point to point communication

1K DMA sends, 16K DMA receives for user level broadcast

Experience with PCTH

all runs produced visualization data

mid July '93

- ws4—10 materials, 2 high explosives, fills 52 16-MByte compute nodes
- running on 52 node compute partition

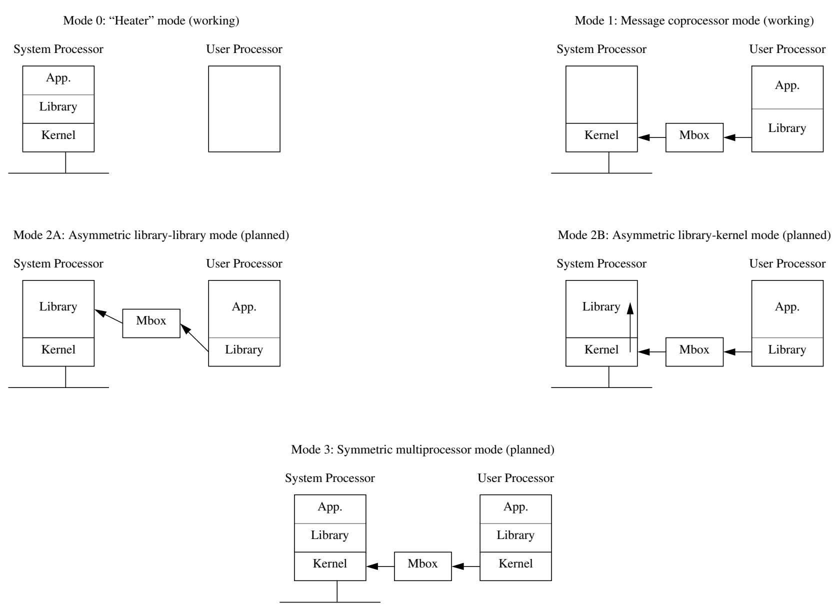
late July '93

- ws4 runs on 416 nodes—5 hours, physics went bad
- ws4 runs on 52 nodes—12 hours, time out

mid August '93

- ws4a—4,000,000 cells, fills 512 16-MByte compute nodes
- ran for 36 hours on 512 nodes
- communication lockup at cycle 1086, 65 microseconds of simulated time

Multiprocessor Modes



Issues

Parallel file system
- low level drivers or pfs
- using nodes as disk cache

Program restart

Enhanced application support
- additional languages (FORTRAN)
- additional message passing support libraries (MPI)

High performance networks
- ATM drivers and protocols
- HiPPI

PUMA

Portals—an opening in the address space of a process
- kernel managed portals
- sender managed portals
- receiver managed portals
- read memory portals

The Q-Kernel
- message transmission
- portal protection

The PCT (Process Control Thread)
- access to servers (e.g., file servers)
- multi-tasking/single-tasking
- flexibility