



The Puma Lightweight Kernel

Ron Brightwell

Sandia National Labs

Scalable Computing Systems Department

rbbrih@sandia.gov



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.





MPP Platforms at Sandia

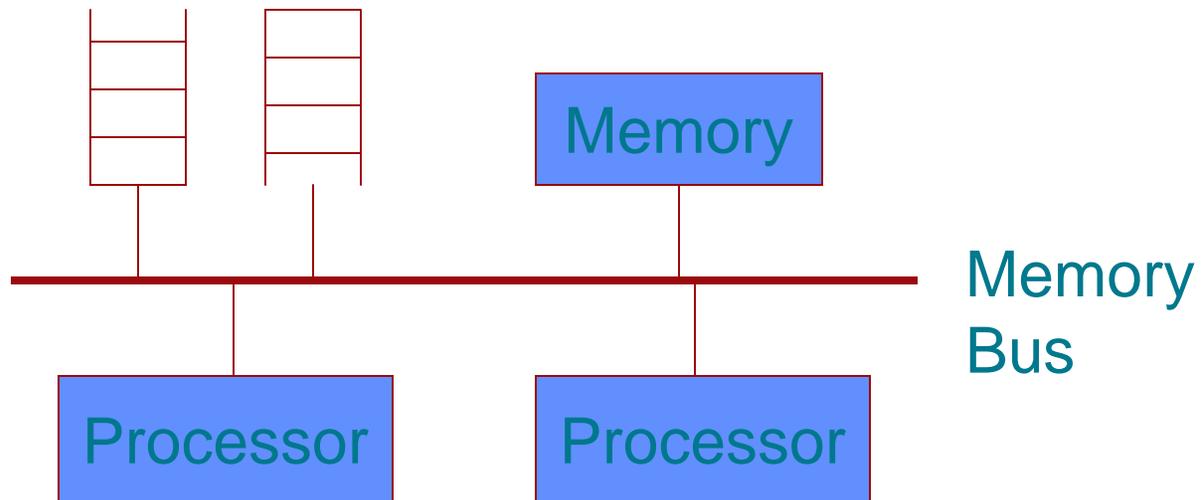
- Intel Paragon
 - 1,890 compute nodes
 - 3,680 i860 processors
 - 143/184 GFLOPS
 - 200 MB/sec network
-
- Intel TeraFLOPS
 - 4,576 compute nodes
 - 9,472 Pentium II processors
 - 2.12/3.15 TFLOPS
 - 400 MB/sec network





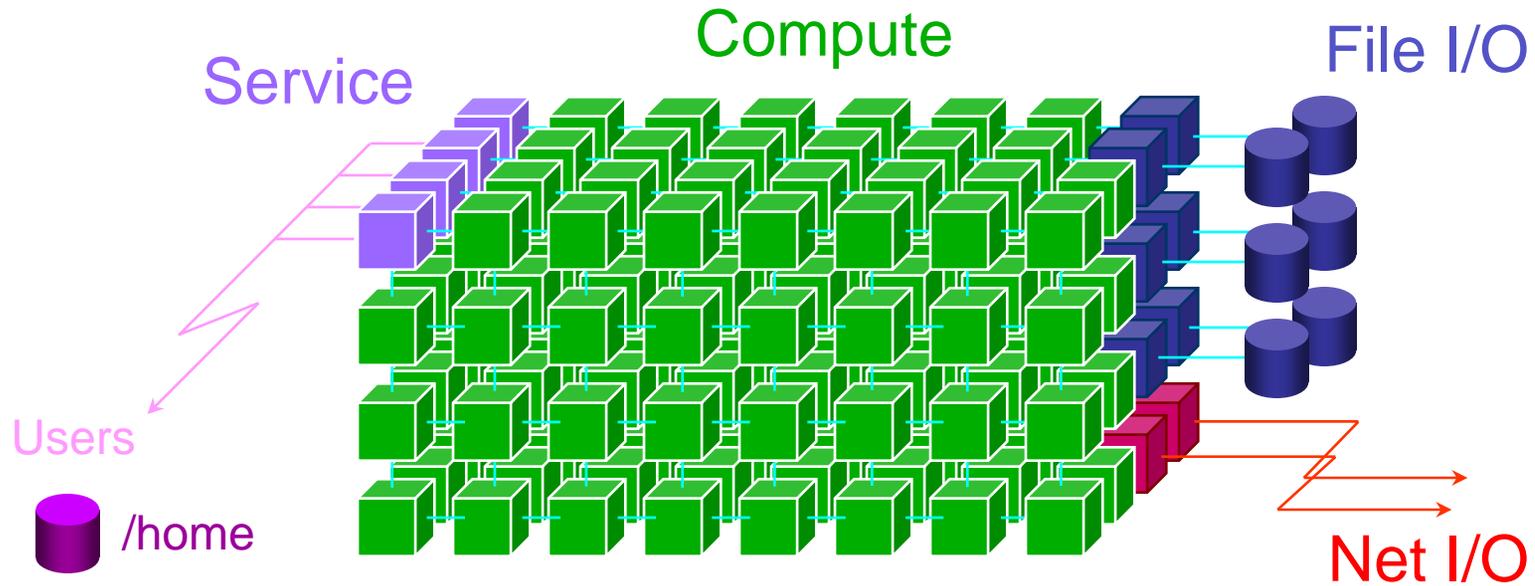
Compute Node Architecture

Network FIFOs





Conceptual Partition Model





MPP System Software R&D

- **SUNMOS lightweight kernel**
 - High performance compute node OS for distributed memory MPP's
 - Deliver as much performance as possible to apps
 - Small footprint
 - Started in January 1991 on the nCUBE-2 to explore new message passing schemes and high-performance I/O
 - Ported to Intel Paragon in Spring of 1993
- **Puma lightweight kernel**
 - Multiprocess support
 - Modularized (QK, PCT)
 - Developed on nCUBE-2 in 1993
 - Ported to Intel Paragon in 1995
 - Ported to Intel TFLOPS in 1996 (Cougar)
 - Portals 1.0
 - User/Kernel managed buffers
 - Portals 2.0
 - Avoid buffering and memory copies





Goals of Puma

- **Targets high performance scientific and engineering applications on tightly coupled distributed memory architectures**
- **Scalable to tens of thousands of processors**
- **Fast message passing and execution**
- **Small memory footprint**
- **Persistent (fault tolerant)**





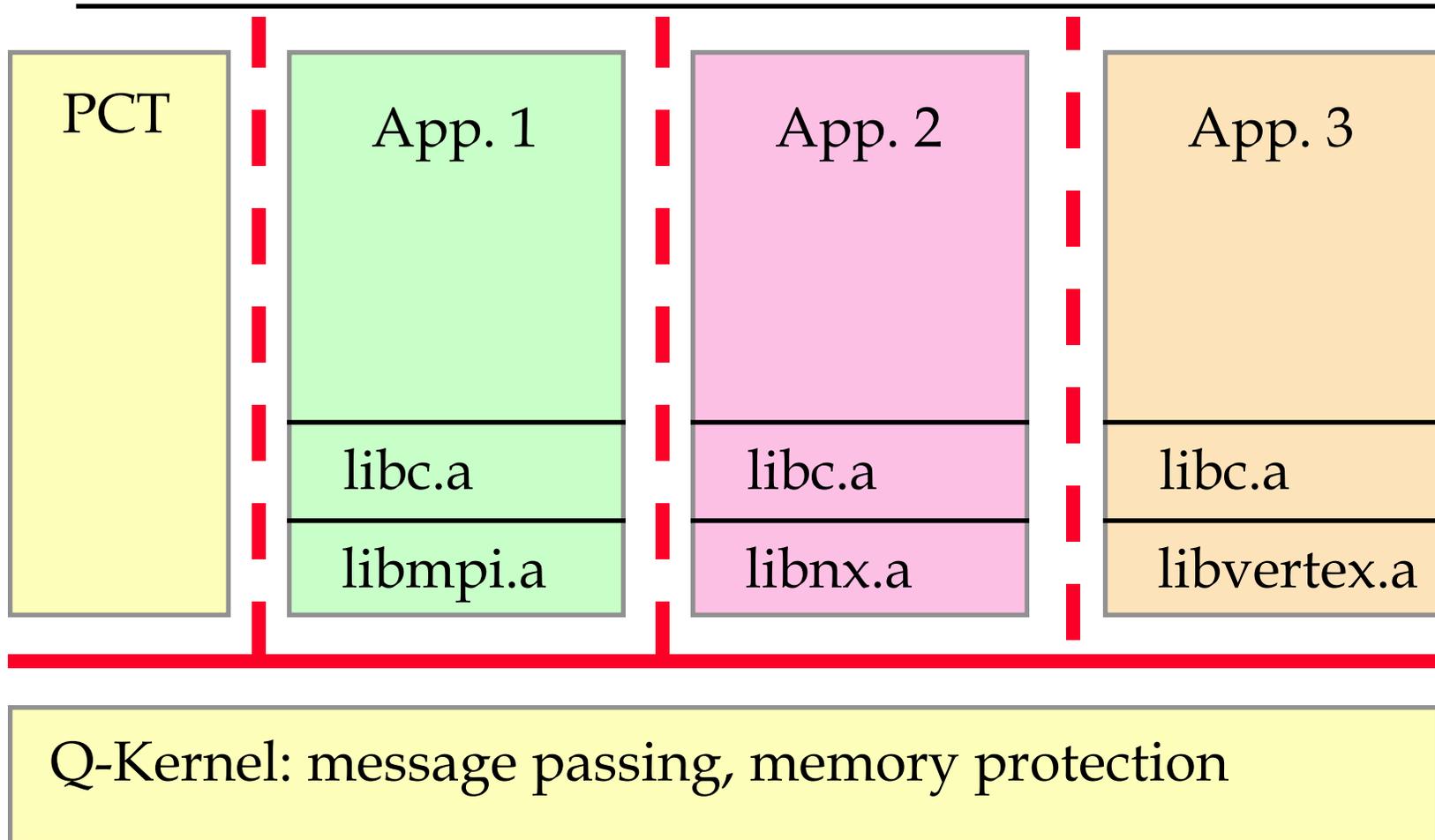
Approach

- **Separate policy decision from policy enforcement**
- **Move resource management as close to application as possible**
- **Protect applications from each other**
- **Let user processes manage resources**
- **Get out of the way**





General Structure





Quintessential Kernel (QK)

- **Policy enforcer**
- **Initializes hardware**
- **Handles interrupts and exceptions**
- **Maintains hardware virtual addressing**
- **No virtual memory support**
- **Static size**
- **Small size**
- **Non-blocking**
- **Few, well defined entry points**





Process Control Thread (PCT)

- **Runs in user space**
- **More privileged than user applications**
- **Policy maker**
 - **Process loading**
 - **Process scheduling**
 - **Virtual address space management**
 - **Name server**
 - **Fault handling**





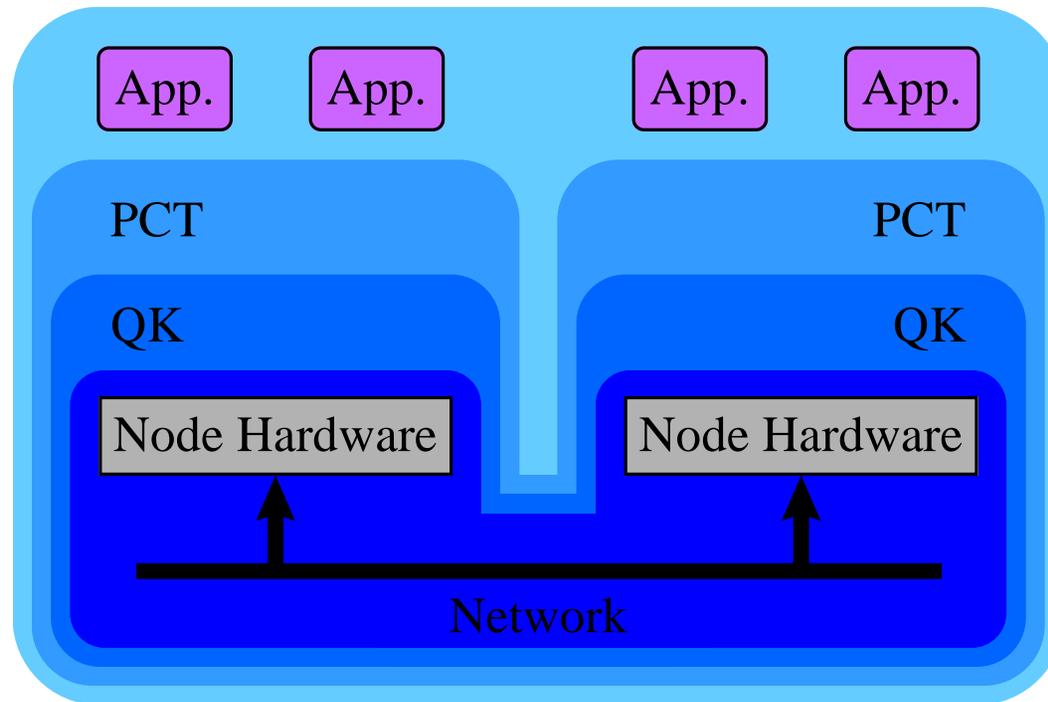
PCT (cont'd)

- **Customizable**
 - Singletasking or multitasking
 - Round robin or priority scheduling
 - High performance, debugging, or profiling version
- **Changes behavior of OS without changing the kernel**





Levels of Trust





CPU Modes

- Chosen at job launch time
- Heater mode (proc 0)
 - QK/PCT and application process on system CPU
- Message co-processor mode (proc 1)
 - QK/PCT on system CPU
 - Application process on second CPU
- Compute co-processor mode (proc 2)
 - QK/PCT and application process on system CPU
 - Application co-routines on on second CPU
- Virtual node mode (proc 3)
 - QK/PCT and application process on system CPU
 - Second application process on second CPU





Yod

- **Parallel job launcher**
- **Runs in the service partition**
- **Communicates via Puma message passing module in TeraFLOPS OS (TOS)**
- **Command line arguments for**
 - **Size of job**
 - **Processor mode**
 - **Stack size**
 - **Heap size**
 - **Sharing**
- **Services I/O and system call requests from compute node processes**





Key Ideas

- **Protection**
- **Kernel is small**
 - Very reliable
- **Kernel is static**
 - No structures depend on how many processes are running
 - All message passing structures are in user-space
- **Resource management pushed out to application processes and runtime system**
- **Services pushed out of kernel to PCT and runtime system**





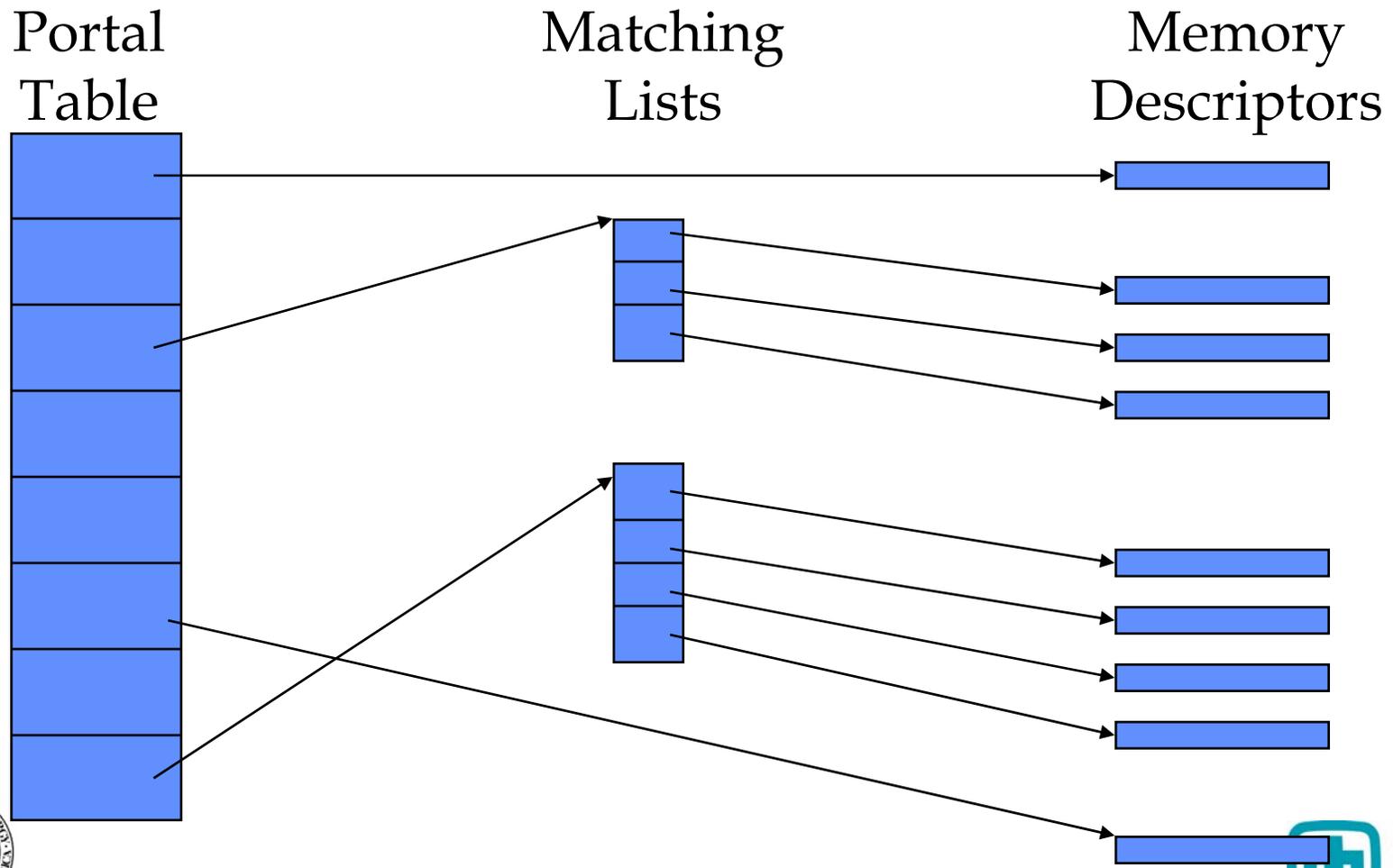
Portals

- **Openings in a process' address space**
- **Basic building blocks for any high-level message passing system**
- **All data structures are in user space**
- **A Portal consists of one or more of the following:**
 - A memory descriptor
 - A matching list
- **Avoids costly memory copies**
- **Avoids costly context switches to user mode (up call)**





Portal Objects





Match Entries

- **Optional layer**
- **Linked list in bounded region**
- **Matching criteria**
 - **Source group, source rank**
 - **64 matching bits**
 - **64 ignore bits**
- **Three-way branching**
 - **No match**
 - **No fit**
 - **No buffer**





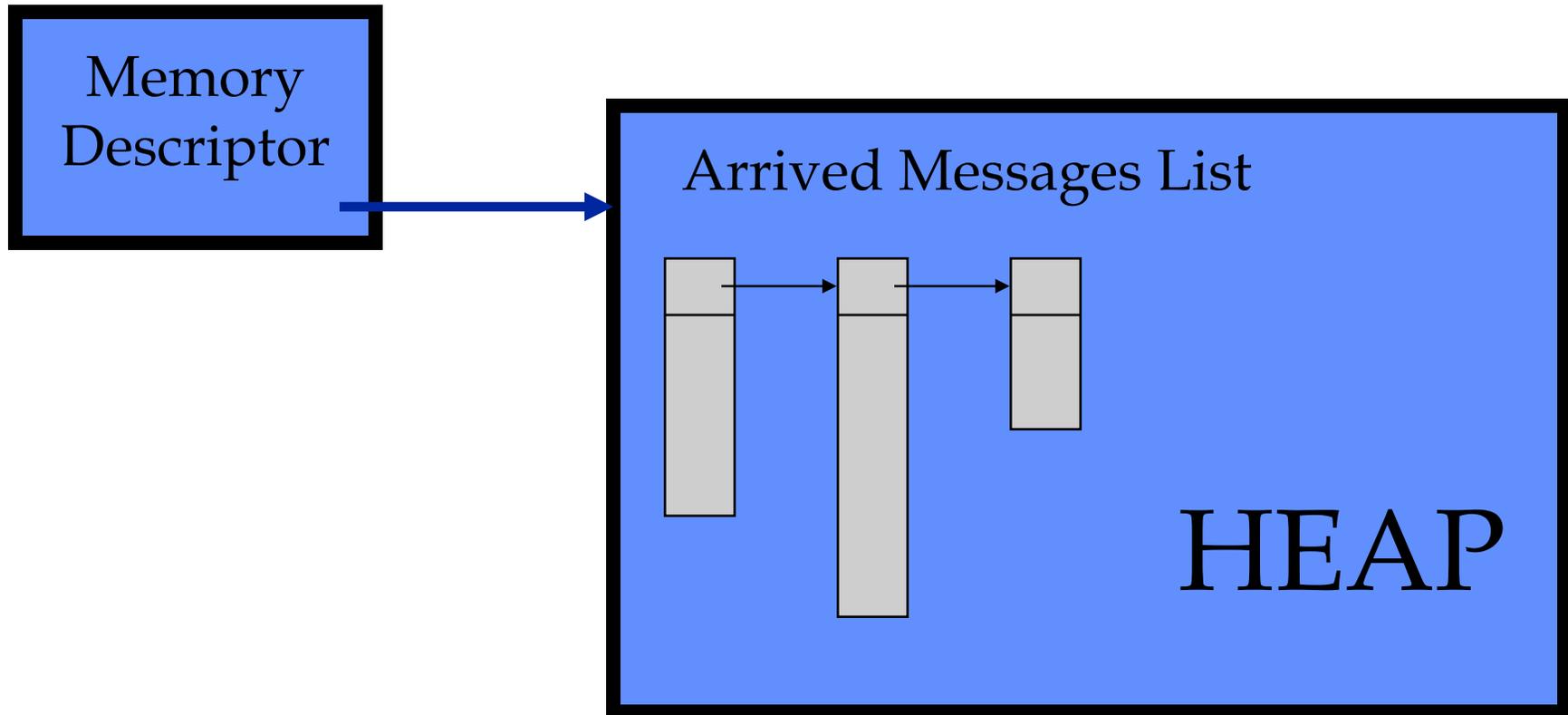
Memory Descriptors

- **Describes layout of application's buffers**
- **Attached directly to Portal table or to match entry**
- **Types**
 - **Dynamic**
 - **Single Block**
 - **Independent Block**
 - **Combined Block**



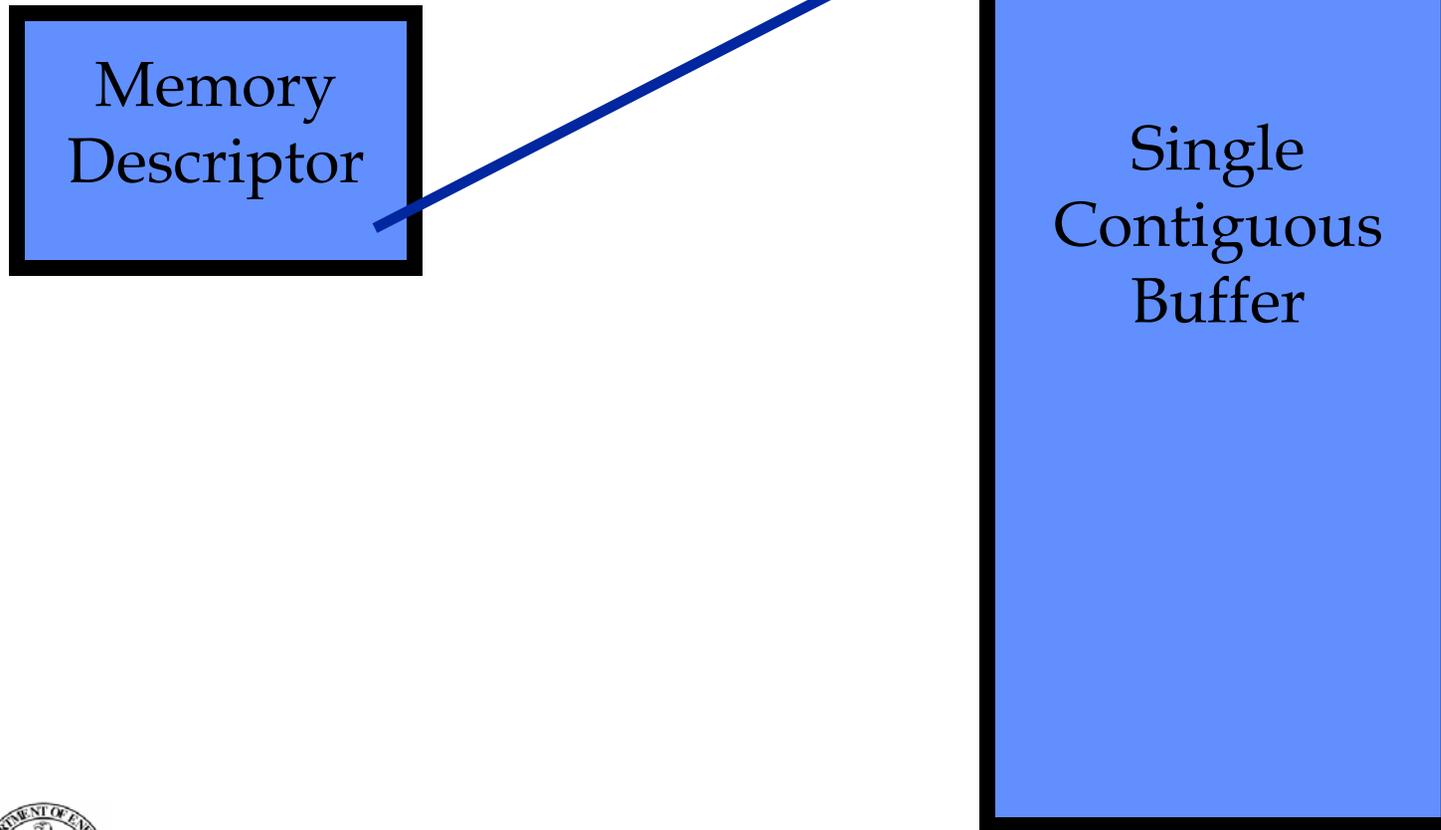


Dynamic



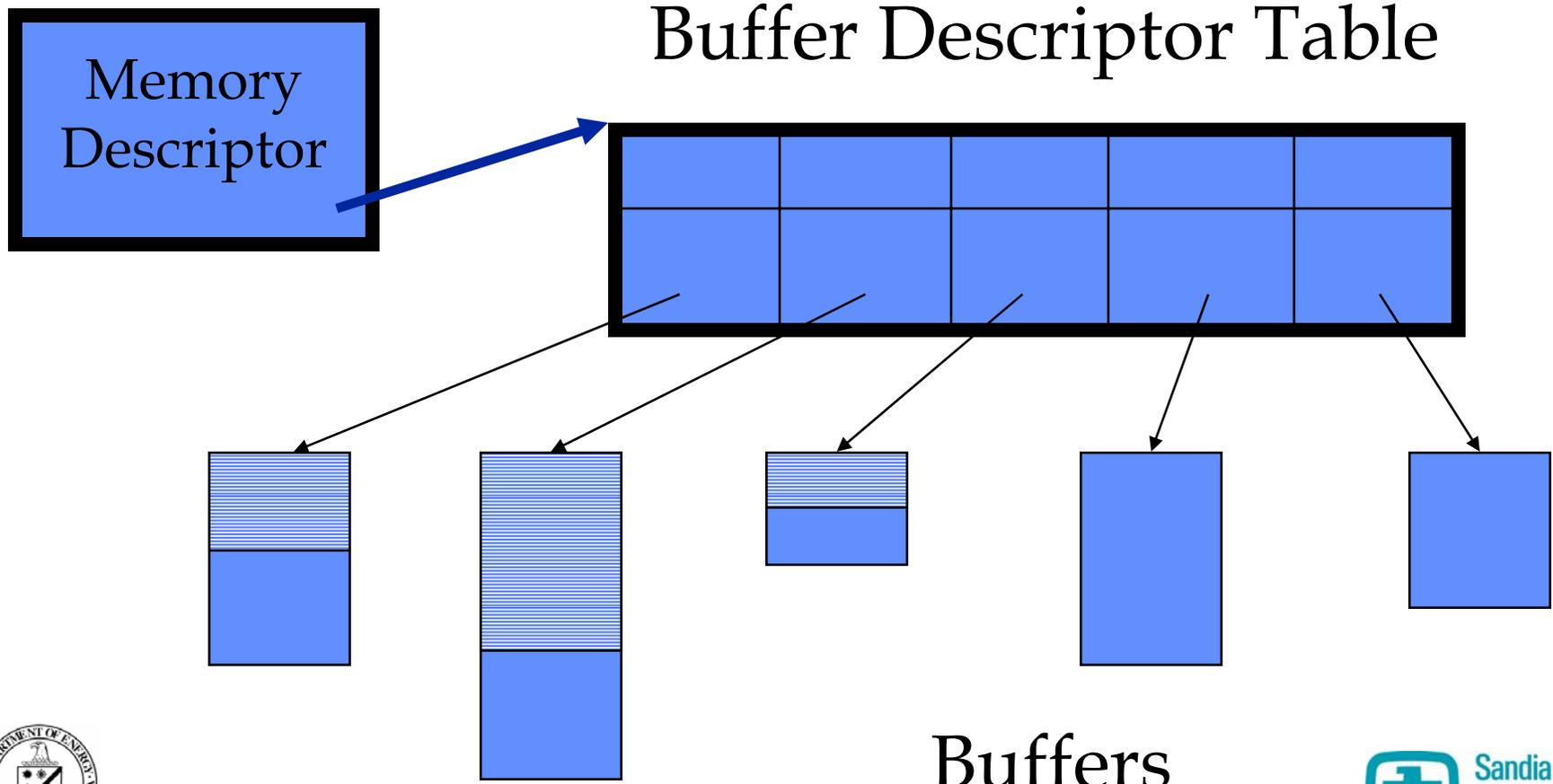


Single Block





Independent Block

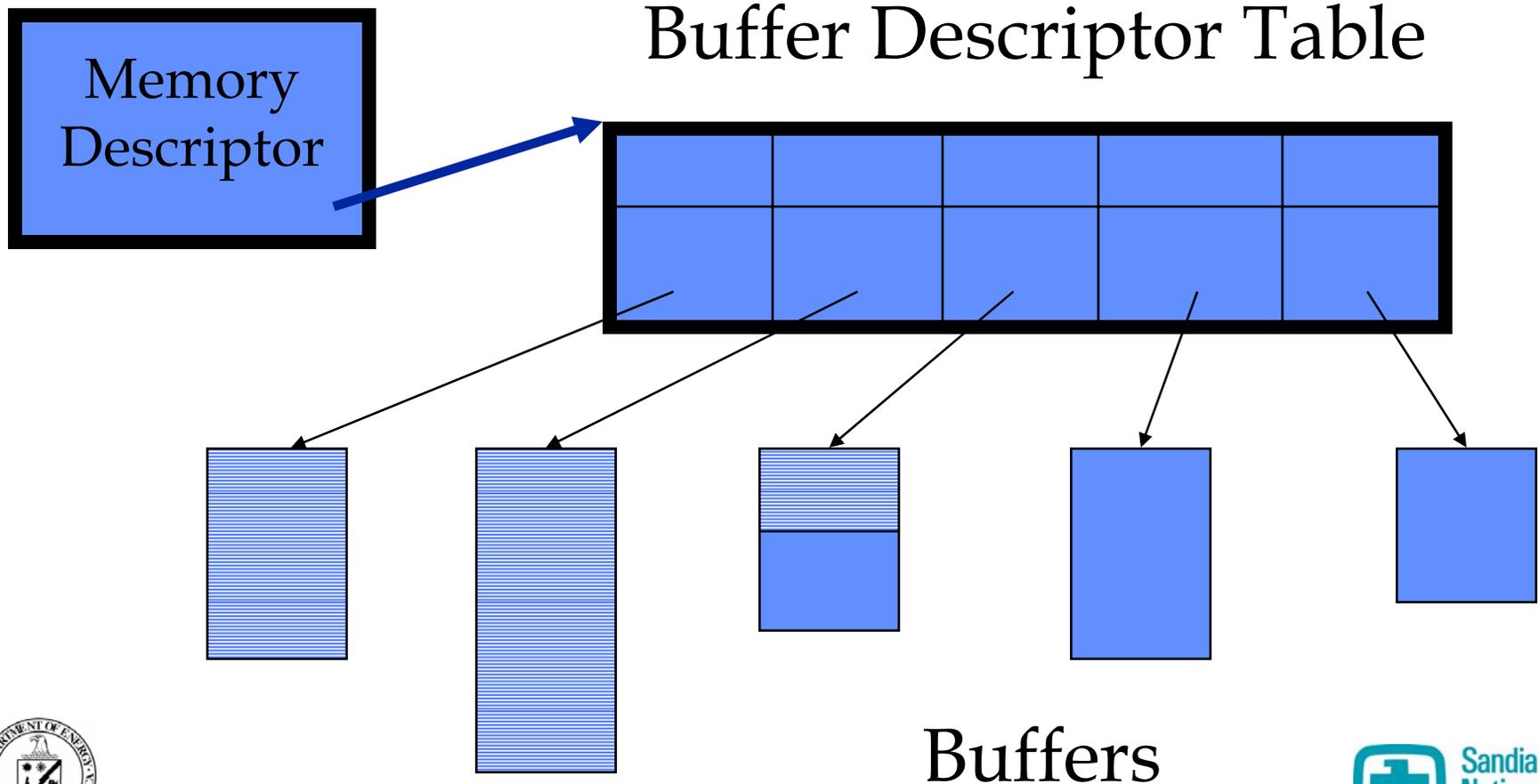


Buffers





Combined Block



Buffers





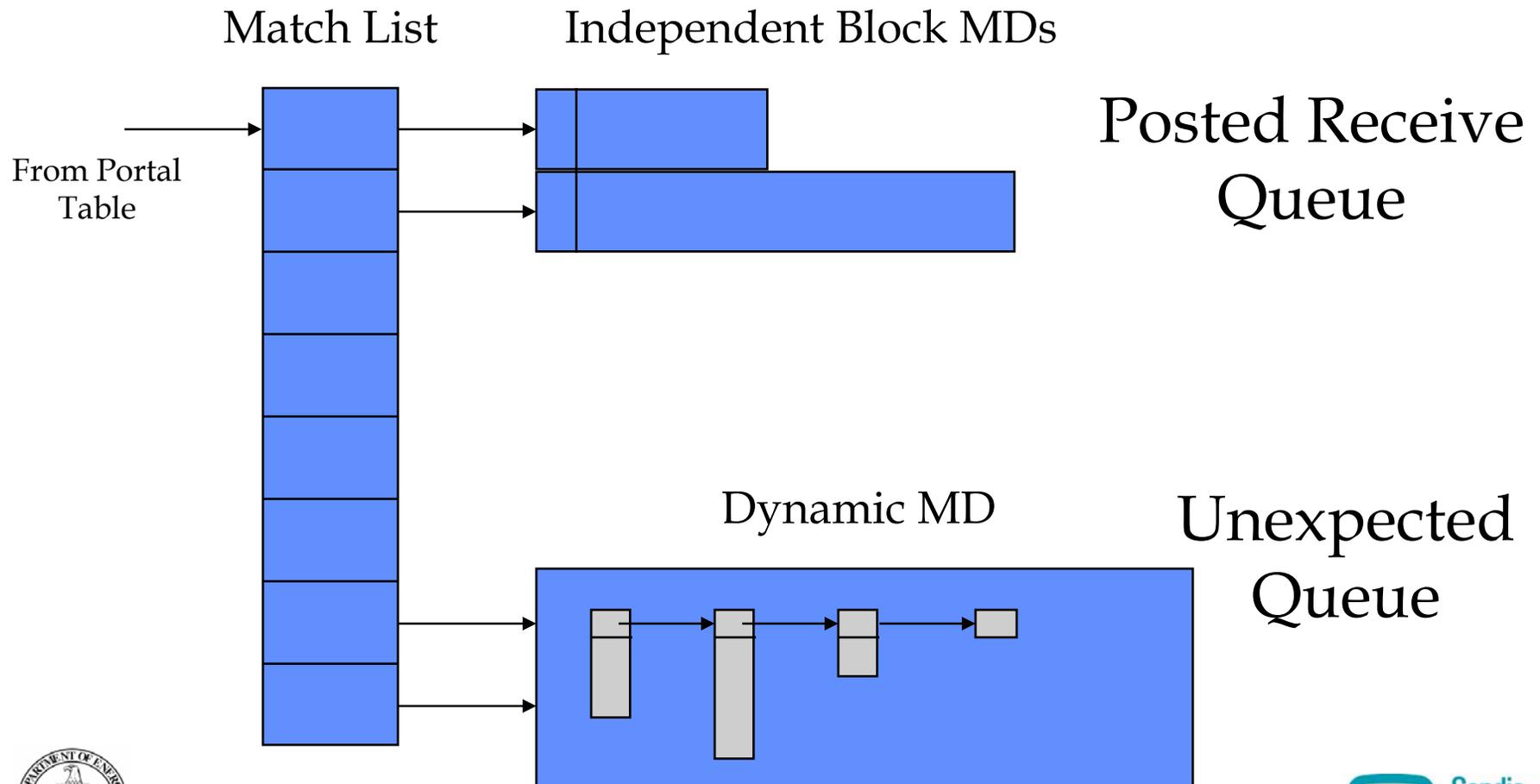
Memory Descriptor Options

- **Read or write**
- **On write**
 - **Save header and body**
 - **Save header only**
 - **Save body only**
- **Offset management**
 - **Sender managed**
 - **Incoming message determines offset**
 - **Receiver managed**
 - **Memory descriptor determines offset**
- **Write acknowledgements**
- **Buffer list traversal**
 - **Circular**
 - **Linear**



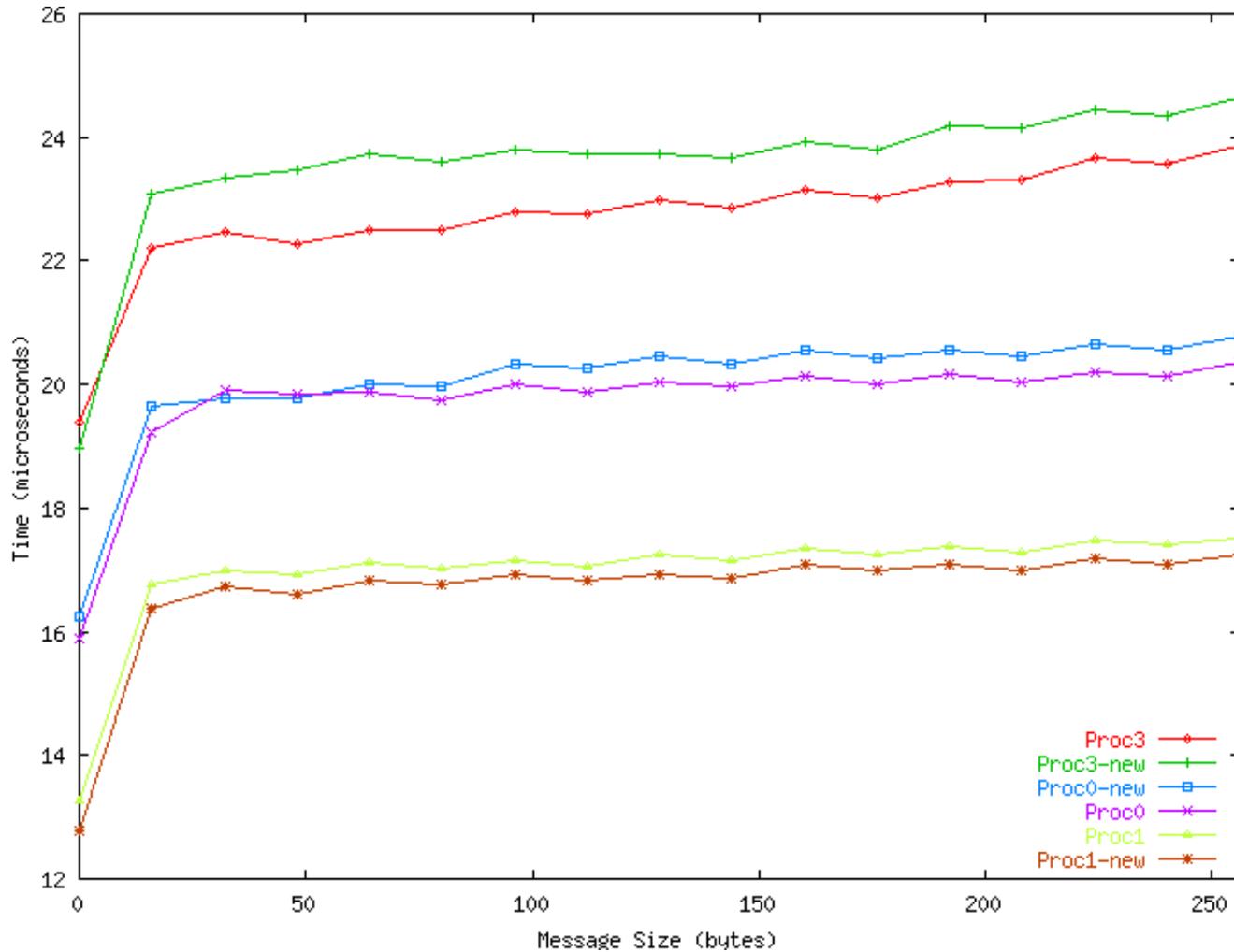


Portal Example





ASCI/Red Ping-Pong MPI Latency Performance





NOP Trap Performance

- **Proc 0 mode**
 - 2.5052 μs
- **Proc 1 mode**
 - 1.6675 μs
- **Proc 3 mode**
 - 2.2968 μs
 - 2.7146 μs
- **Proc 0 mode with –share**
 - 4.4973 μs
 - 3.3056 μs





Lightweight Kernel?

- **Puma**

– section	size
– .text	83357
– .data	21856
– .bss	105440
– Total	210653

- **PCT**

– section	size
– .text	274993
– .data	50928
– .bss	642344
– Total	968265





Memory Use Breakdown

- **Executable**
 - $164833(.text) + 35340(.data) + 135816(.bss) = 335989$
- **Proc 0 / Proc 1 mode**
 - Text = 167936
 - Data = 172032
 - Stack = 2097152
 - Heap = 260046848
 - NX Heap = 262144
- **Proc 3 mode**
 - Text = 167936
 - Data = 172032
 - Stack = 2097152
 - Heap = 127926272
 - NX Heap = 262144



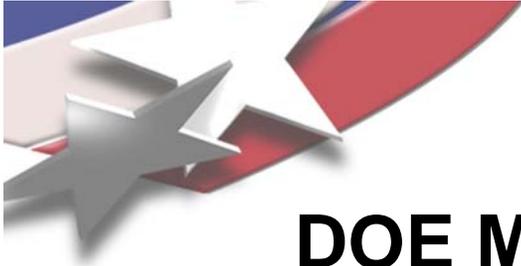


Source Lines Of Code (SLOC)*

- **QK**
 - 9078 CPU independent
 - 10061 x86-specific
 - 6088 i860-specific
- **PCT - 12823**
- **yod - 13150**
- **Libraries**
 - I/O and C – 16343
 - MPI
 - Device independent - 23849
 - Portals – 5234
- **Linux (kernel, init, mm, include/linux) - 87453**



*Generated using 'SLOCCount' by David A. Wheeler



DOE MICS Lightweight Kernel Project

- **Three-year project to design and implement next-generation lightweight kernel for compute nodes of a distributed memory massively parallel system**
- **Assess the performance and reliability of a lightweight kernel versus a traditional monolithic kernel**
- **Investigate efficient methods of supporting dynamic operating system services**
- **Leverage Linux as much as possible**

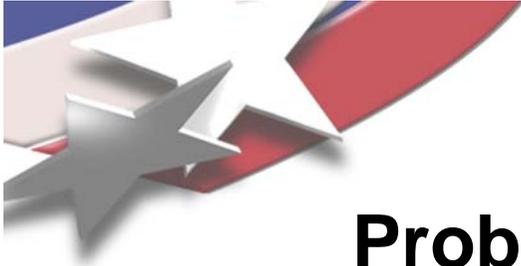




FY02 Plans

- **Restructure Cougar to work with Linux-based boot kernel**
- **Complete port of Cougar to Alpha EV67 processors**
- **Integrate Portals 3.0 into the Cougar (using Myrinet)**
- **Interface Cougar to the Cplant™ service partition**
- **Demonstrate CTH and Pronto on a development system**
- **Assess the performance and scalability of applications relative to a Cplant™ running Linux**





Problems with Portals 2.0 in Puma

- **No API**
 - Data structures entirely in user-space
 - Protection boundaries have to be crossed to access data structures
 - Data structures must be copied, manipulated, and copied back
 - Requires interrupts
- **Address validation/translation on the fly**
 - Incoming messages trigger address validation
 - Doesn't fit Linux model of validating addresses on a system call for the currently running process





Portals 3.0

- **Operational API with unified memory descriptor**
- **Provides elementary building blocks for supporting higher-level protocols, not just MPI**
- **Allows structures to be placed in user-space, kernel-space, or NIC-space depending on underlying transport layer**
- **Allows for high-performance OS-bypass implementations**
- **Receiver-managed offset allows for efficient and scalable buffering of MPI unexpected messages**
- **Supports multiple protocols within a process**
- **Application-bypass is a good thing**
 - **OS-bypass is necessary but not sufficient for high-performance**





MPI Double-Buffer Benchmark

Rank 0

```
isend A;  
isend B;  
for ( ) {  
    fill A; wait CTS A;  
    isend A;  
  
    fill B; wait CTS B;  
    isend B;  
}
```

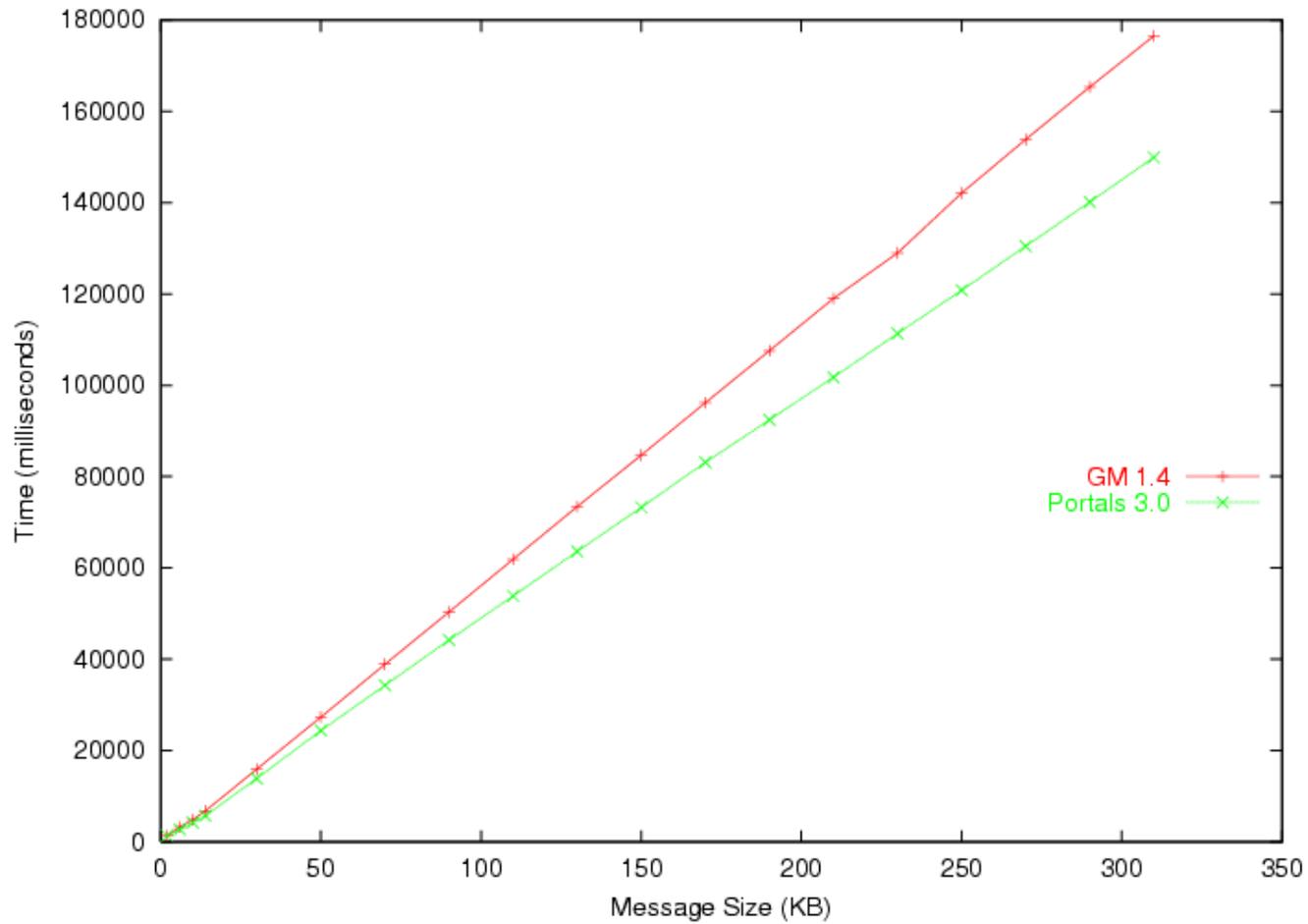
Rank 1

```
start = get_time();  
for ( ) {  
    wait A; sum A;  
    isend CTS A;  
  
    wait B; sum B;  
    isend CTS B;  
}  
end = get_time();
```



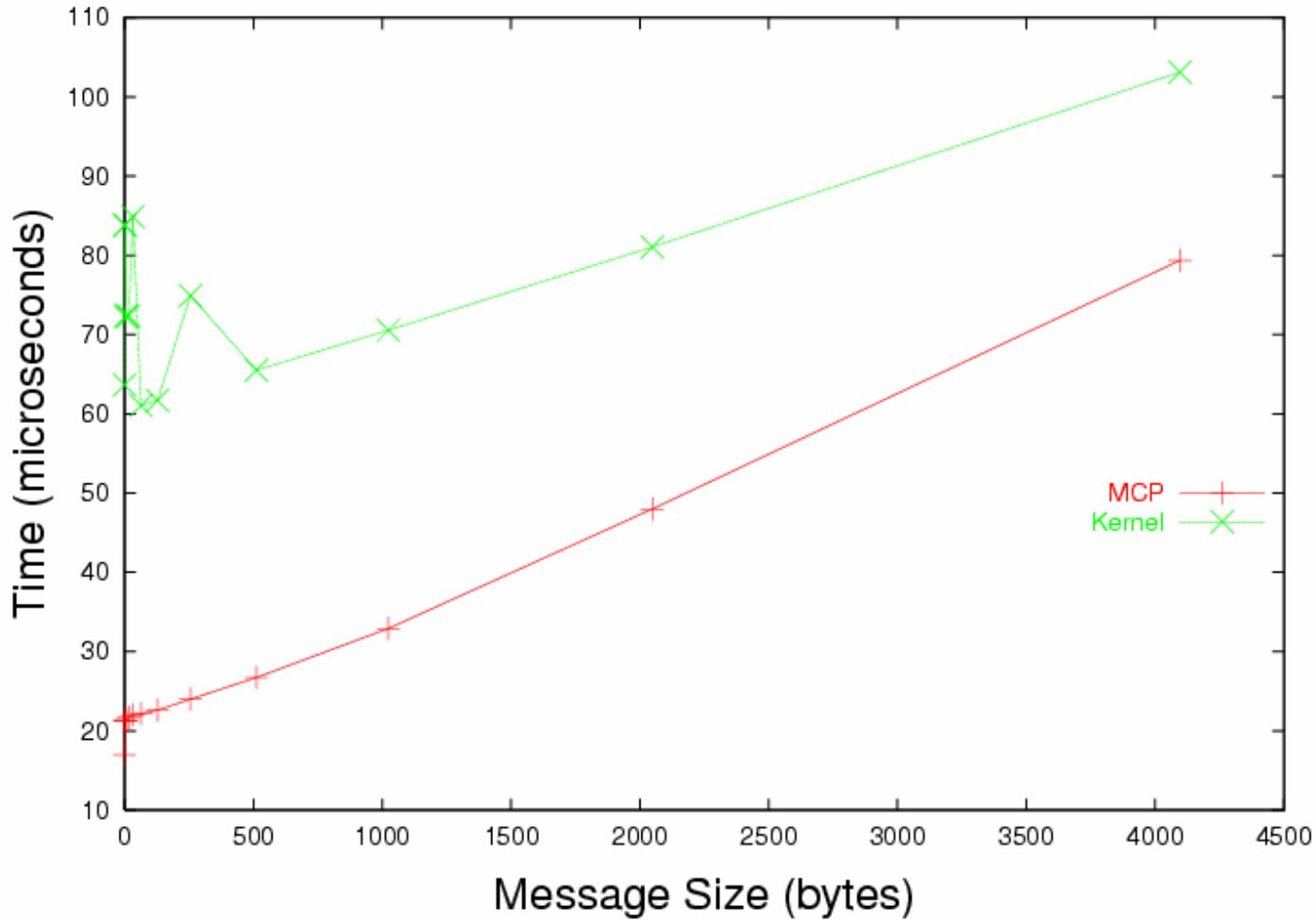


MPI Double-Buffer Performance





Portals 3.0 Myrinet MCP Implementation





Portals 3.0 NAL Implementations

- **RTS (Sandia)**
 - Linux kernel module that does reliability and flow control
 - Can use any Linux network device (skbufs)
- **IP (Sandia)**
 - Reference implementation
 - Really UNIX Pipes
- **Quadrics ELAN3 (Sandia)**
 - Uses ELAN thread and DMA queues
- **Myrinet MCP (Sandia)**
 - Designed to work with Cougar
- **Alteon GigE (UNM)**
- **In-kernel TCP/IP (Peter Braam, Cluster File Systems, Inc.)**
- **Quadrics ELAN Kernel Comms (Marcus Miller, LLNL)**
- **Quadrics Tports (Unlimited Scale, Inc.)**





New OS Initiative - Filling the Gap

- **Current and future initiatives for tera-scale and peta-scale systems are focusing on hardware architecture and programming models, not on operating systems and runtime systems**
- **SciDAC Scalable System Software project is capturing current state of the art and not focused on addressing fundamental problems of future large-scale systems**
- **Two most scalable systems did not use full UNIX-based operating systems (ASCI/Red,Cray T3)**
- **Still many basic research questions regarding operating systems and runtime systems for 100 teraOPS and petaOPs platforms (extensibility, fault tolerance, etc.)**
- **Need to start gathering support for new initiative now**

