Matthew Dillon
DragonFly BSD Project
08 January 2004

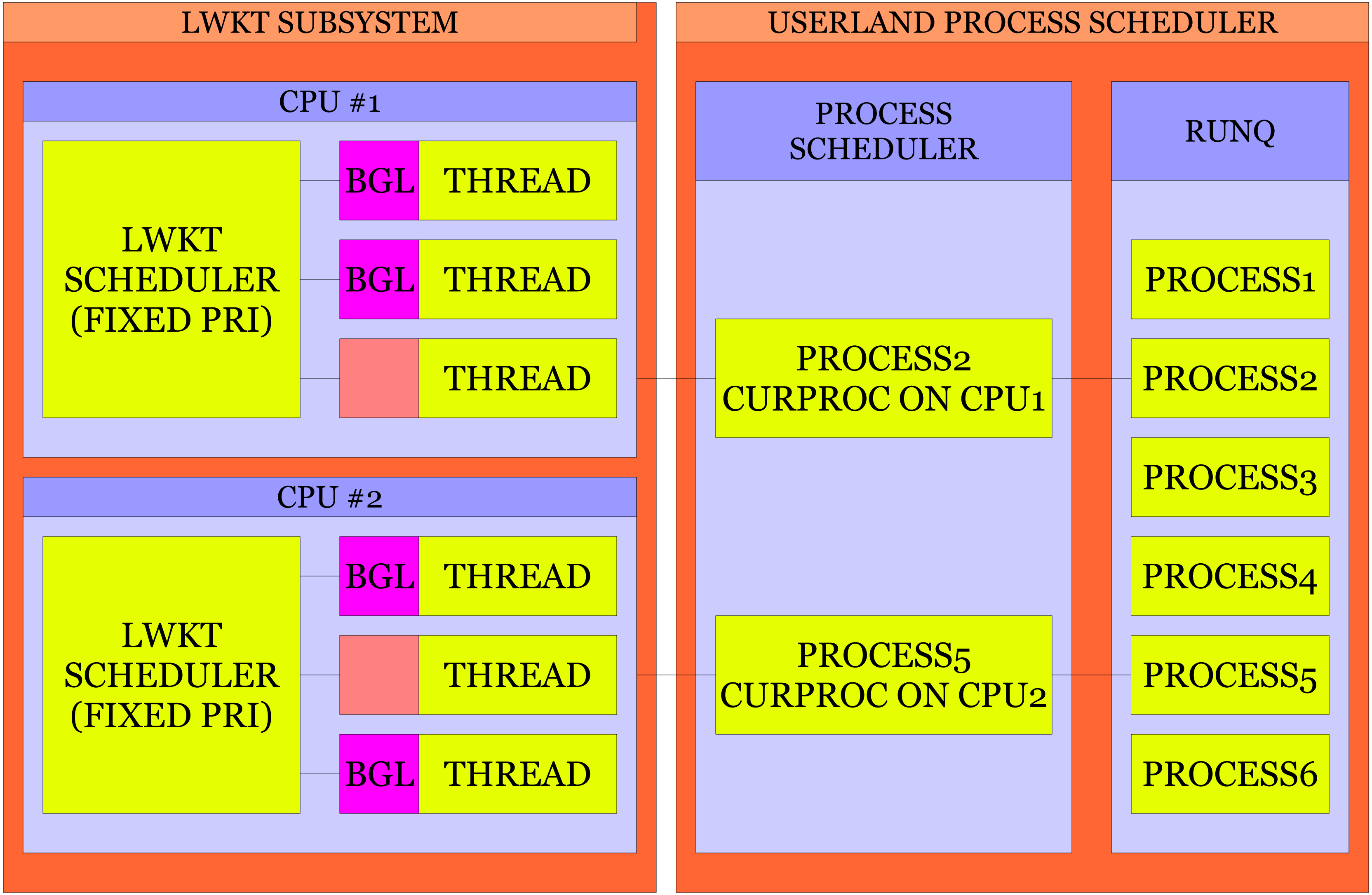
DragonFly Status

- Kernel Differentiators Completed
  - Light Weight Kernel Threading and Process Separation
  - IPI Messaging
  - Light Weight Kernel Messaging
  - Slab Allocator / KVM mapping simplifications
- Differences in Approach Between DFly and FreeBSD-5.x
  - Mutexes verses CPU Localization
  - Mutexes verses Thread-based Serialization
- Kernel Differentiators Near-Term Work
  - Robust IPC Messaging Mechanism
  - AMD64 Port W/Mixed Environment Support
- Future Work – Achieving SSI
  - Proxy Message Ports
  - Cache Coherency
  - Range Locking
  - Global File Handles
  - Piecemeal Process Migration
  - Piecemeal Device Driver Migration

Completed Kernel Differentiators

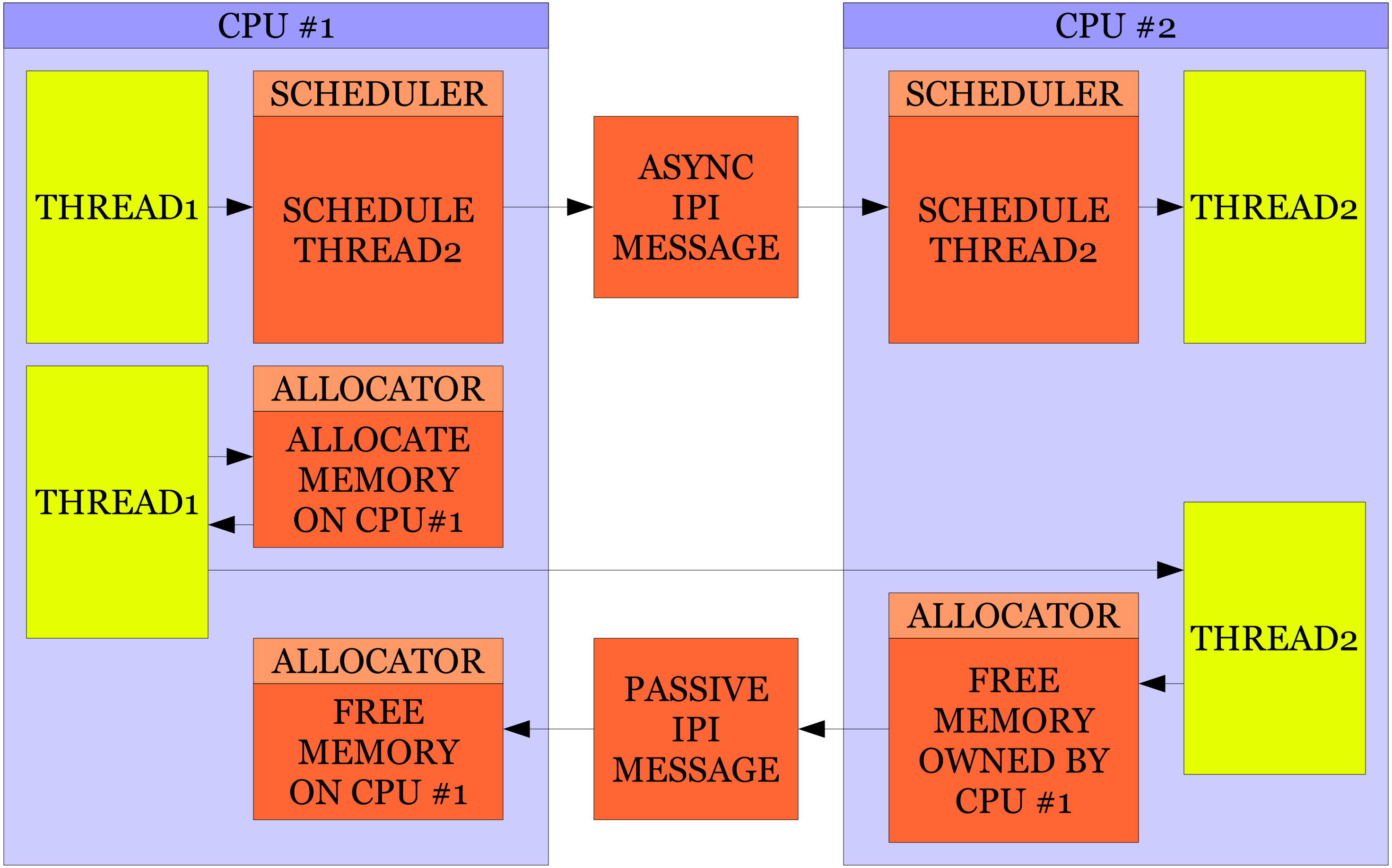
- Light Weight Kernel Threading and Process Separation
- IPI Messaging
- Light Weight Kernel Messaging
- Slab Allocator / KVM mapping simplifications

Light Weight Kernel Threading and User Processes



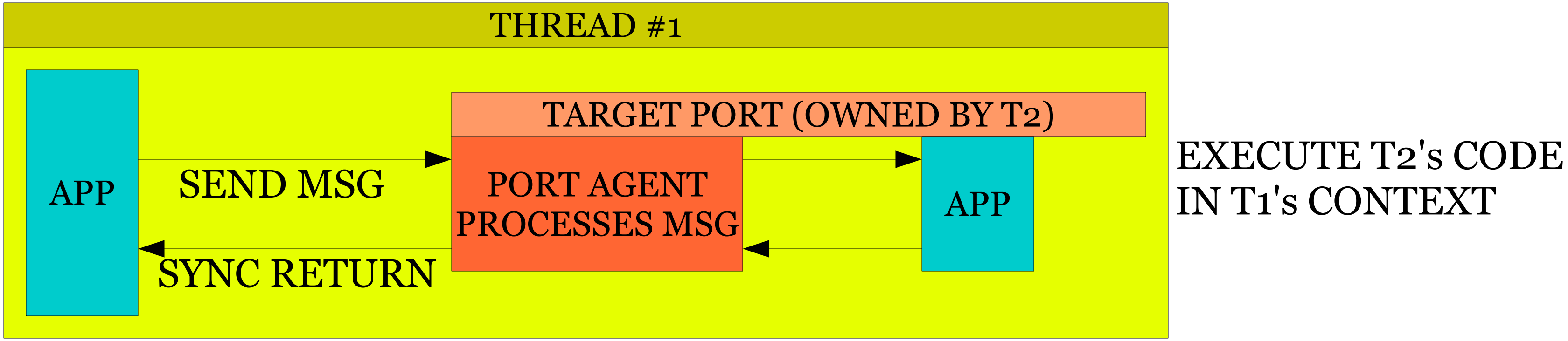
IPI Messaging

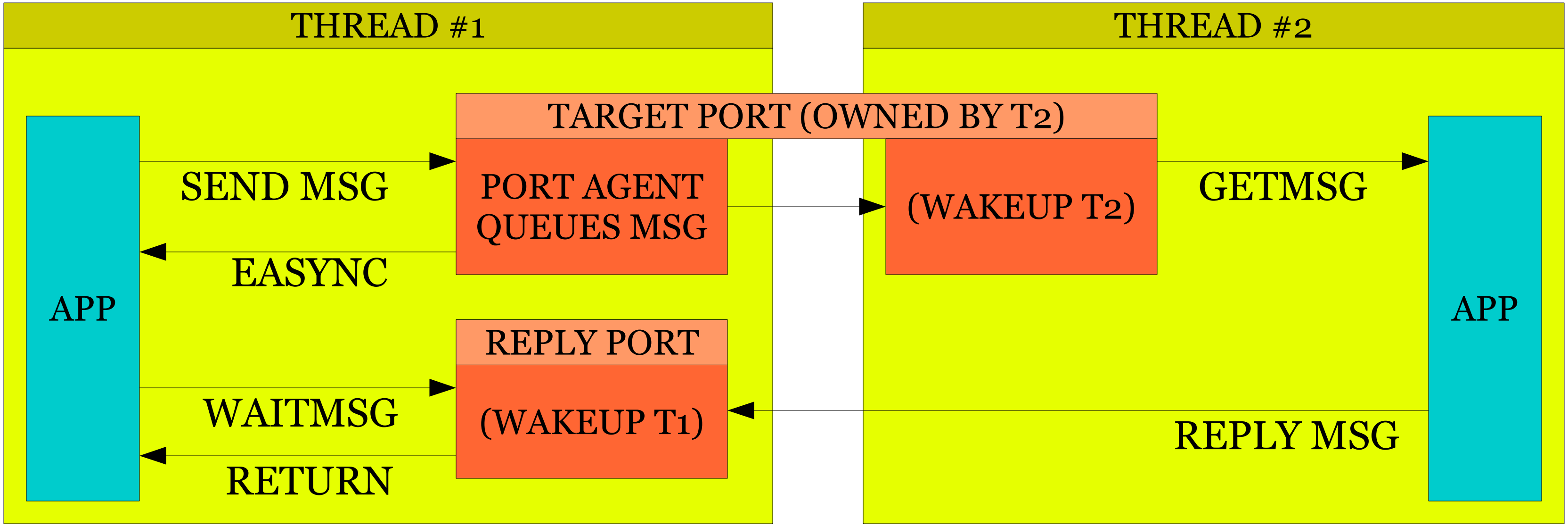
- •ABSTRACTION PROMOTES CPU ISOLATION
- •ASYNCHRONOUS IPI MESSAGING AVOIDS MUTEX OPS
- •SIMPLE CRITICAL SECTIONS FOR LOCAL ACCESS
- •MANY IPI OPS CAN BE PASSIVE / CONTRAST W/ RCU



Light Weight Kernel Messaging

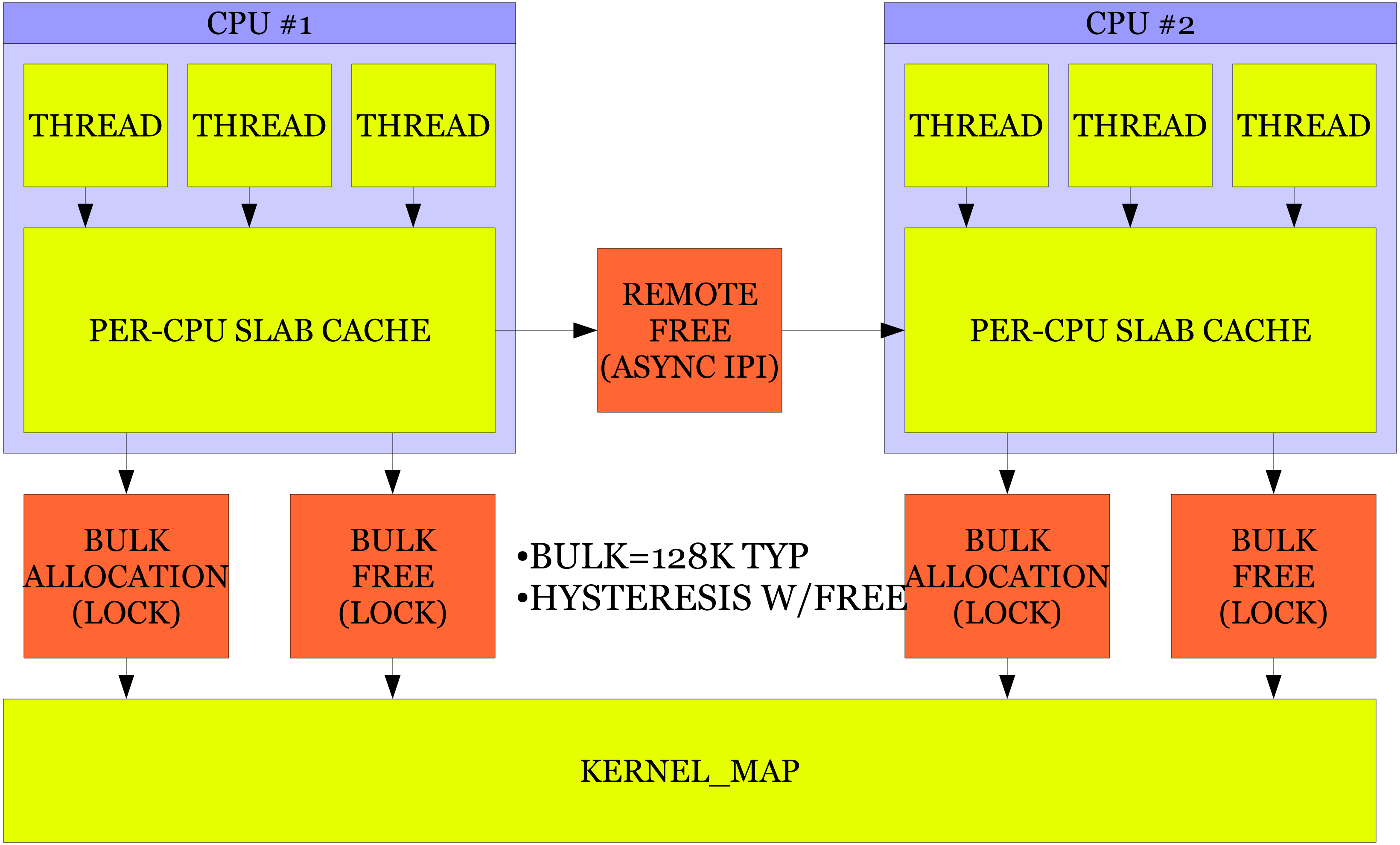
- AMIGA STYLE MESSAGES AND PORTS
- SEMI SYNCHRONOUS / PORT AGENT
- FAST SYNCHRONOUS PATH





Slab Allocator

- PER-CPU LOCALIZATION
- BACKED BY KERNEL_MAP
- NO MORE KMEM_MAP

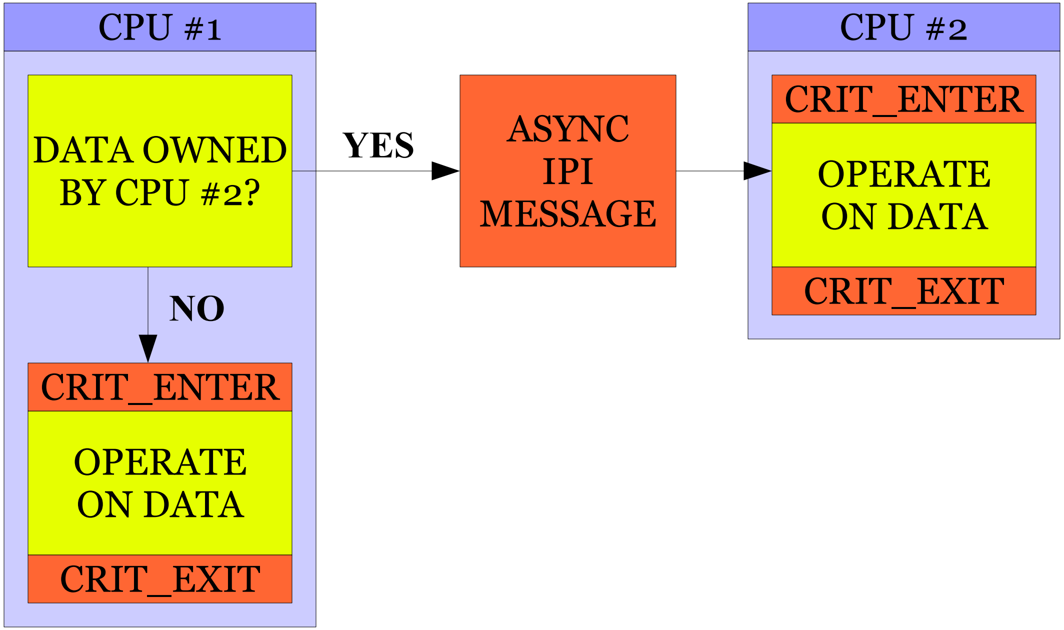


- BULK=128K TYP
- HYSTERESIS W/FREE

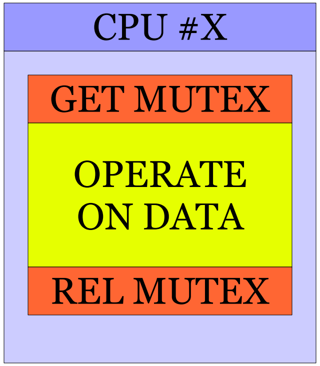
Differences in Approach between DragonFly and FreeBSD-5

- CPU Localization verses Mutexes
- Thread based Serialization verses Mutexes

CPU Localization vs Mutexes

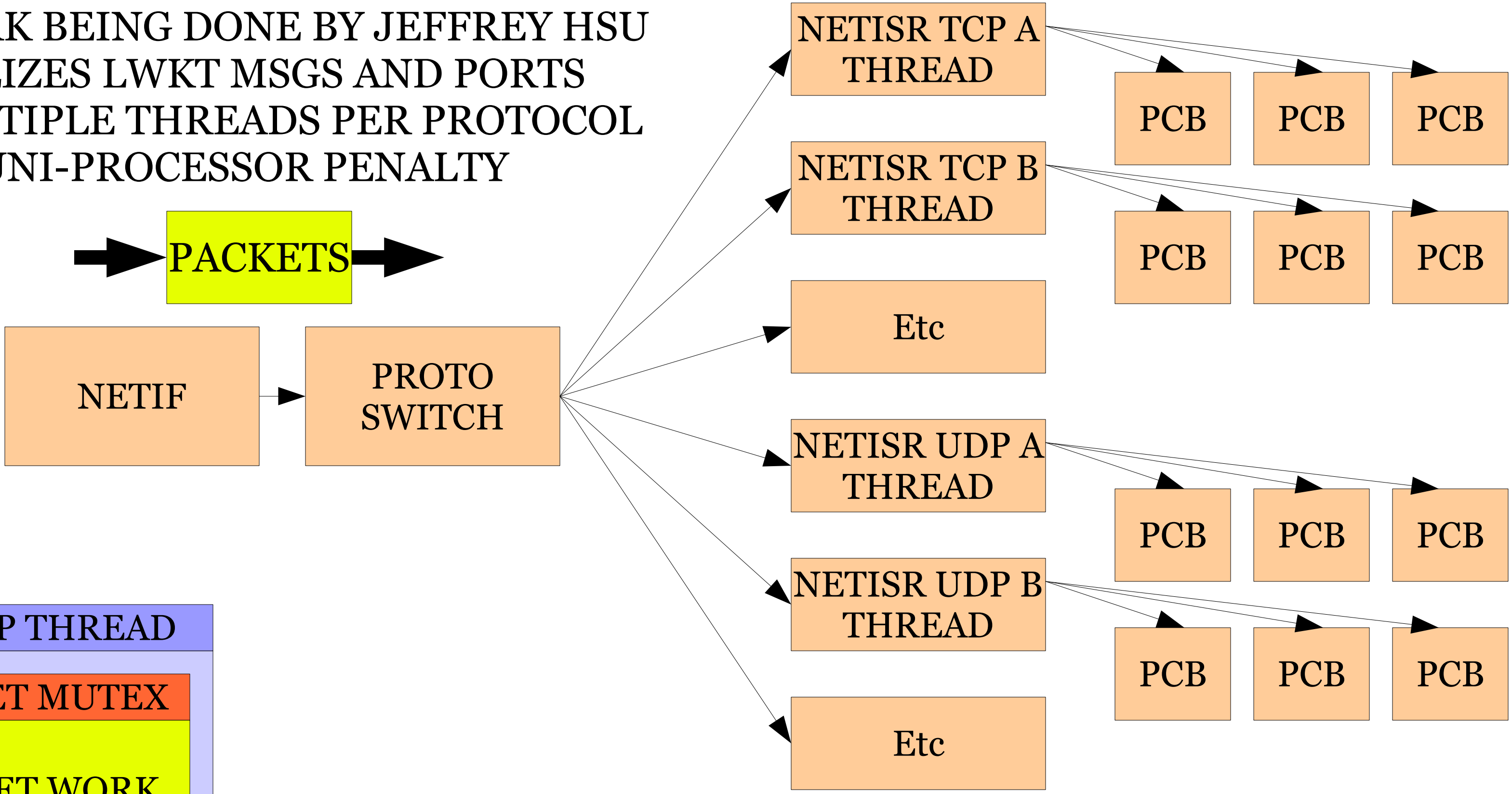


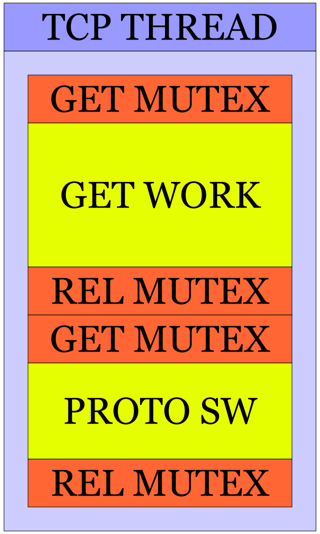
- •API ABSTRACTION
- •BLOCKING ISSUES IN MAINLINE CODE
- •BLOCKING ISSUES WITH INTERRUPTS
- •CACHE MASTERSHIP CHANGES
- •COMPLEX RECURSIONS
- •MUTEX OVERHEAD VERSES IPI OVERHEAD



Thread Based Serialization vs Mutexes

- WORK BEING DONE BY JEFFREY HSU
- UTILIZES LWKT MSGS AND PORTS
- MULTIPLE THREADS PER PROTOCOL
- NO UNI-PROCESSOR PENALTY



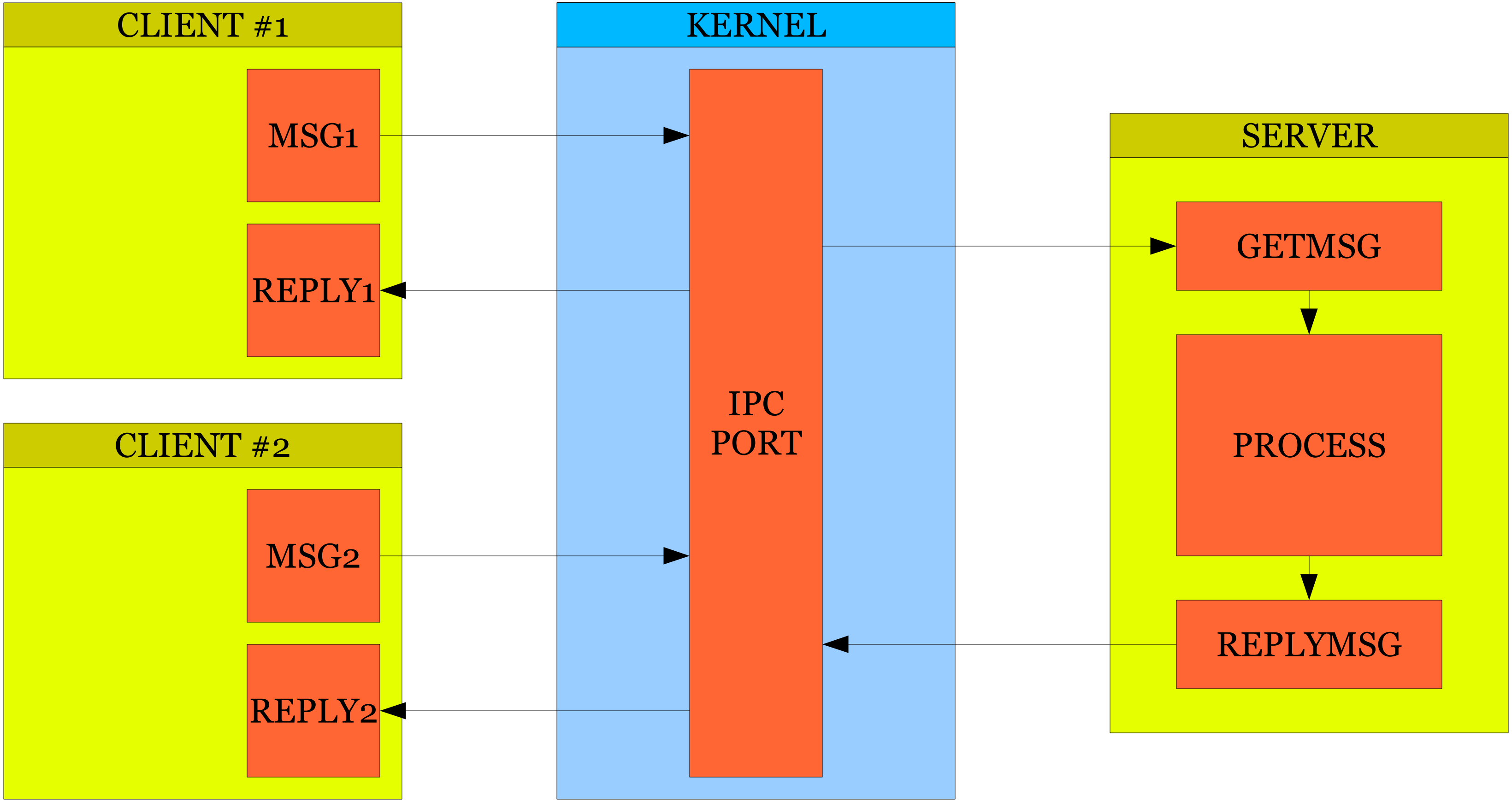


Kernel Differentiators Near-Term Work

- Robust IPC Messaging Mechanism
- Upcoming AMD64 Work

Robust IPC Messaging Mechanism

- CLIENT/SERVER UID+NAMED IPC RENDEZVOUS MODEL
- MESSAGES ARE AGGREGATED IN THE KERNEL TO SIMPLIFY SERVER
- AUTO REPLY UNSERVICED MESSAGES ON SERVER EXIT
- AUTO REPLY TIMED OUT MESSAGES
- LWKT IN-PLACE MESSAGING ABSTRACTION

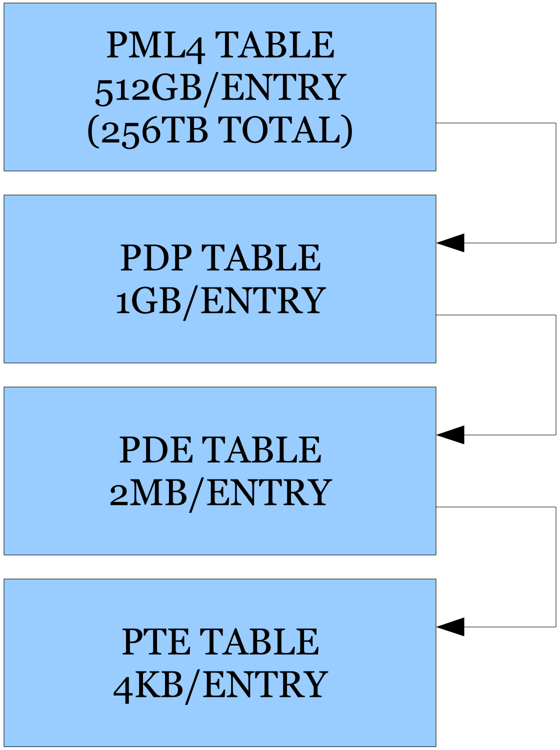


Upcoming AMD64 Work

- 64 BIT NATIVE MODE KERNEL
- 4-LEVEL MMU, SPECIAL CASE THE TOP LAYER, LINEARIZE PDE TABLE
- SWITCH CHANGEOUT PML4 ENTRY REPRESENTING USERLAND
- DIRECT MAP PHYSICAL MEMORY IN KVM
- SUPPORT 32 BIT EMULATION AND 32 BIT BINARY COMPATIBILITY

- EACH TABLE HAS 512 ENTRIES
- PML4/PDP/PDE: 9 ADDRESS BITS
- PTE: 12 ADDRESS BITS (4K PAGES)
- PAE SUPPORTED ON PDE ENTRIES

- 8 EXISTING REGISTERS CAN HOLD 64 BITS
- 8 ADDITIONAL GENERAL (64 BIT) REGISTERS
- DESCRIPTOR TABLES ARE LARGER



PML4 TABLE
512GB/ENTRY
(256TB TOTAL)

PDP TABLE
1GB/ENTRY

PDE TABLE
2MB/ENTRY

PTE TABLE
4KB/ENTRY

Achieving a Single System Image (SSI)

Matthew Dillon
DragonFly BSD Project

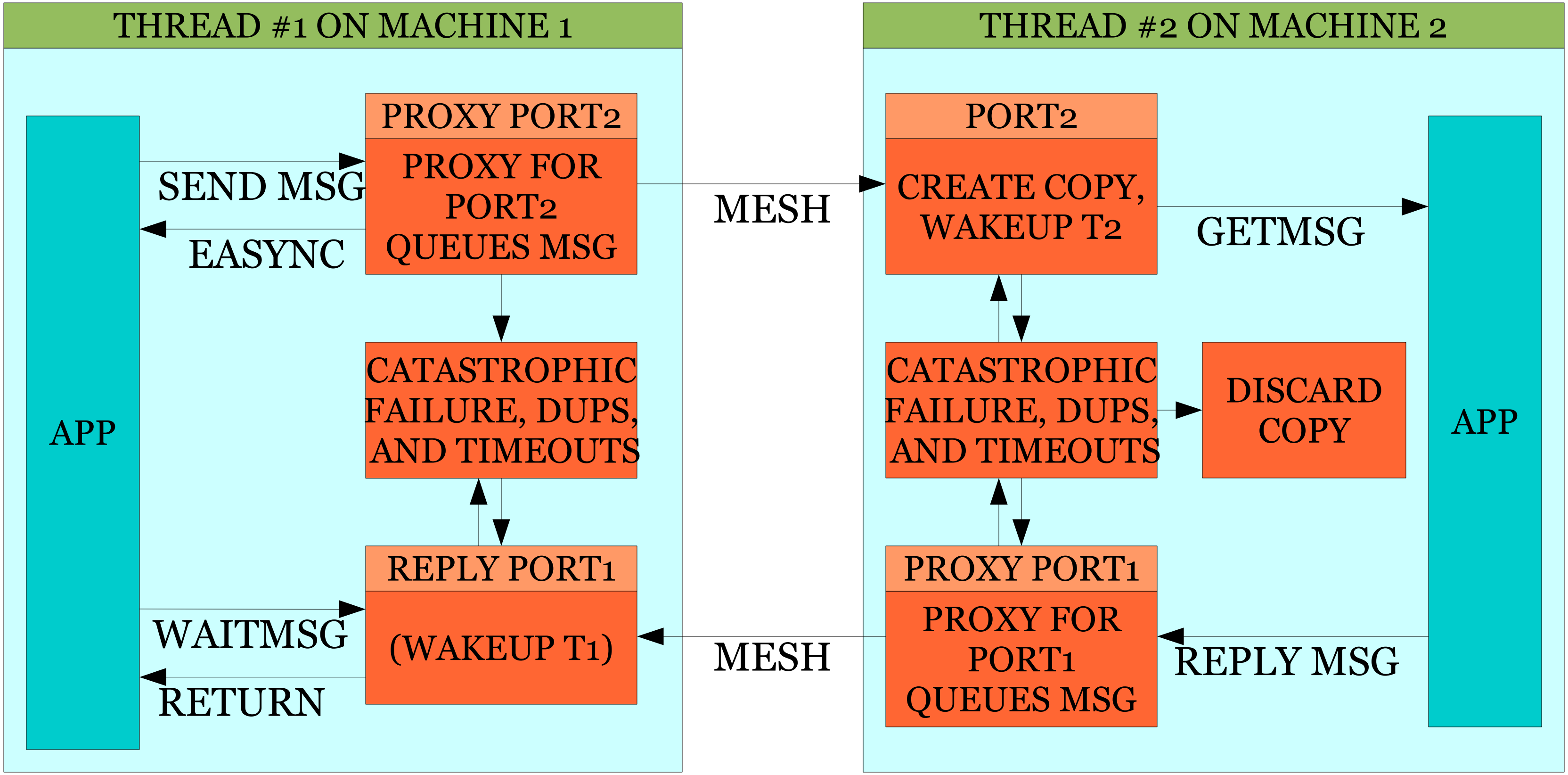
23 October 2003

Upcomming SSI Implementation Details

- CLUSTER MULTIPLE BOXES USING STANDARD NETWORK PROTOCOLS
- THE THREAD MESSAGING ABSTRACTION BECOMES VITAL
- THE CPU MESSAGING ABSTRACTION BECOMES VITAL (IPIs vs MUTEXs)
- IMPLEMENT A NETWORKED MESI CACHE COHERENCY MODEL
- PAGE-LEVEL CACHE COHERENCY, RANGE BASED LOCKING
- COPY DATA INDIRECTLY VIA THE CACHE COHERENCY MODEL
- GLOBAL FILE HANDLES, MESSAGING INTERFACE
- WHAT IS THE ULTIMATE TEST? PROCESS MIGRATION IN PIECES
- ADDING ROBUSTNESS (TIMEOUTS, TRANSACTIONS, VOTING)
- CONTRIBUTING RESOURCES TO A CLUSTER

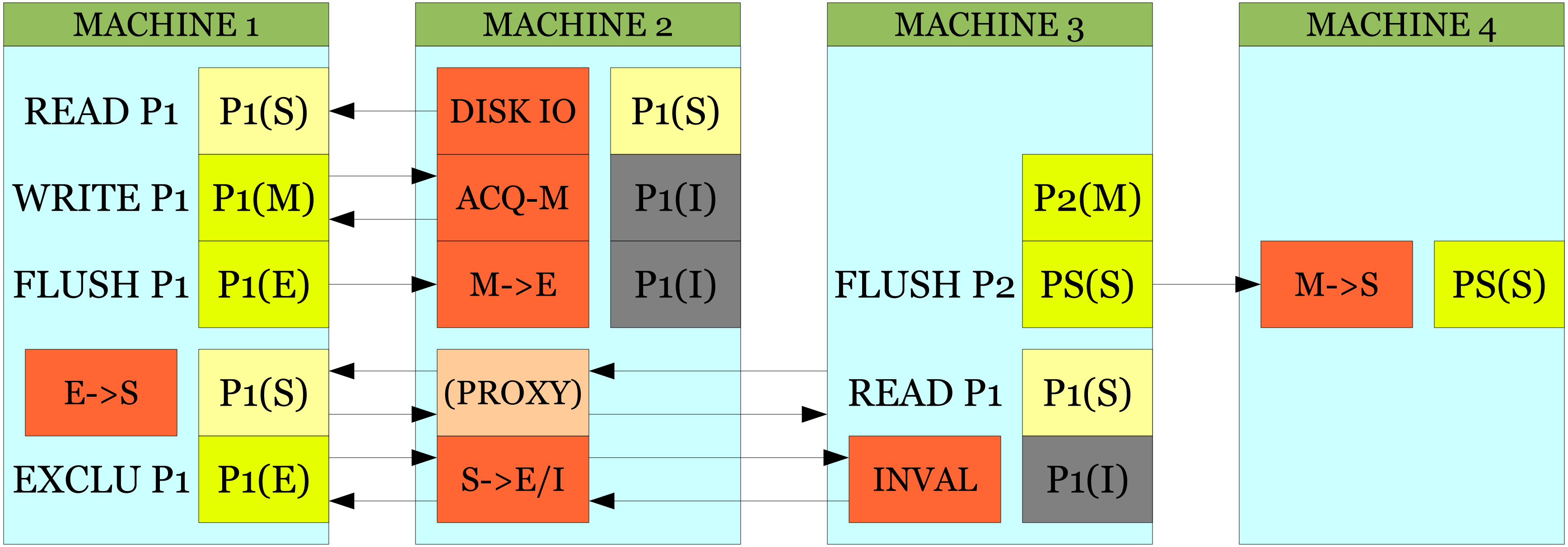
Using Proxy Message Ports

- PROXY PORT REPRESENTS REAL PORT
- LOCAL COPY OF MESSAGE HELD UNTIL REMOTE REPLY OR FAILURE
- PROXY PORT HANDLES MESH FAILURES, TIMEOUTS, AND PROTOCOL ISSUES
- ASSOCIATED DATA HANDLED BY CACHE COHERENCY PROTOCOLS
- PATH FOR ASSOCIATED DATA DICTATED BY CACHE COHERENCY PROTOCOLS
- PATH FOR ASSOCIATED DATA CAN BE OPTIMIZED



MESI Cache Coherency

- MESI = MODIFIED EXCLUSIVE SHARED INVALID
- DATA SHARING IS VITAL FOR EFFICIENT OPERATION OVER WAN INTERFACES
- CACHE COHERENCY MAKES THE CLUSTER INVISIBLE
- MEI IS EASIER TO IMPLEMENT BUT FAR LESS EFFICIENT
- E->M, M->E TRANSITIONS REQUIRE NO MESSAGE TRAFFIC
- FLUSHING MODIFIED DATA CAN MOVE FROM 'M' TO EITHER 'E' OR 'S'
- MACHINE HOLDING E OR M DECIDES DISPOSITION, ELSE BS DECIDES DISPOSITION
- IF E/M HOLDER IS UNKNOWN, BACKING STORE CAN PROXY REQUEST



MACHINE 1

READ P1 P1(S)

WRITE P1 P1(M)

FLUSH P1 P1(E)

E->S P1(S)

EXCLU P1 P1(E)

MACHINE 2

DISK IO P1(S)

ACQ-M P1(I)

M->E P1(I)

(PROXY)

S->E/I

MACHINE 3

P2(M)

FLUSH P2 PS(S)

READ P1 P1(S)

INVAL P1(I)

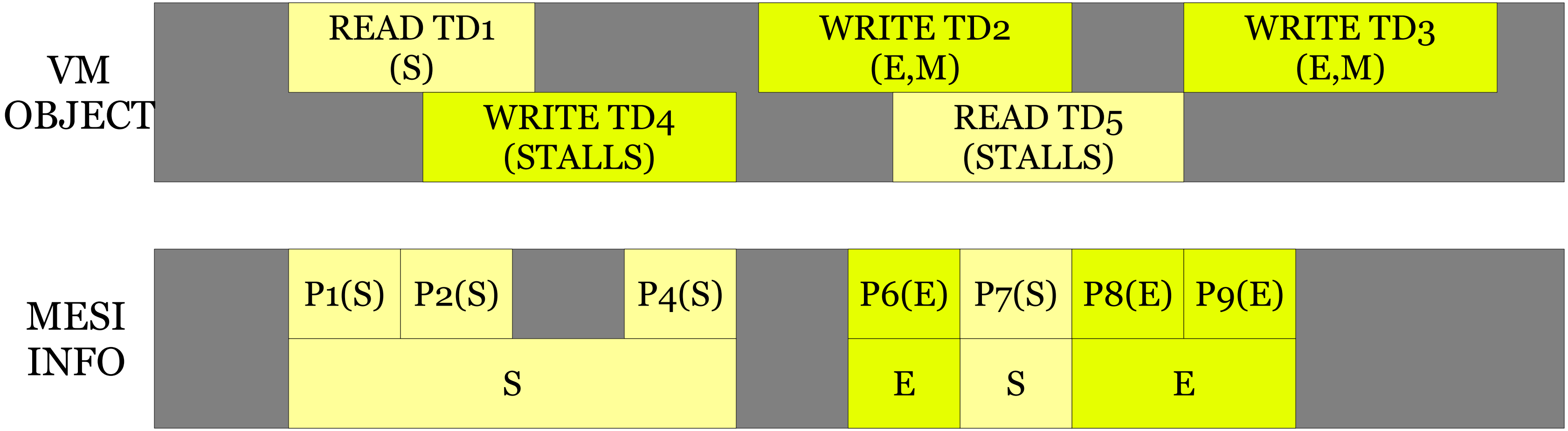
MACHINE 4

M->S PS(S)

BACKING STORE
FOR P1

Range Locking

- •RESERVE OFFSET RANGE IN OBJECT FOR UPCOMING I/O OPERATION (HEURISTIC)
- •PRESERVE UNIX READ/WRITE ATOMICY WITHOUT LIMITATION
- •ALLOWS PARALLEL READS, WRITES, AND COMBINATIONS ON THE SAME FILE
- •AGGREGATE INTO LARGER GRANULARITIES TO REDUCE MANAGEMENT OVERHEAD
- •POTENTIALLY KEEP TRACK OF MESI STATUS IN A FIXED AMOUNT OF RAM
- •RESERVE MULTIPLE RANGES TO SUPPORT TRANSACTIONS



VM OBJECT

READ TD1 (S)

WRITE TD2 (E,M)

WRITE TD3 (E,M)

WRITE TD4 (STALLS)

READ TD5 (STALLS)

MESI INFO

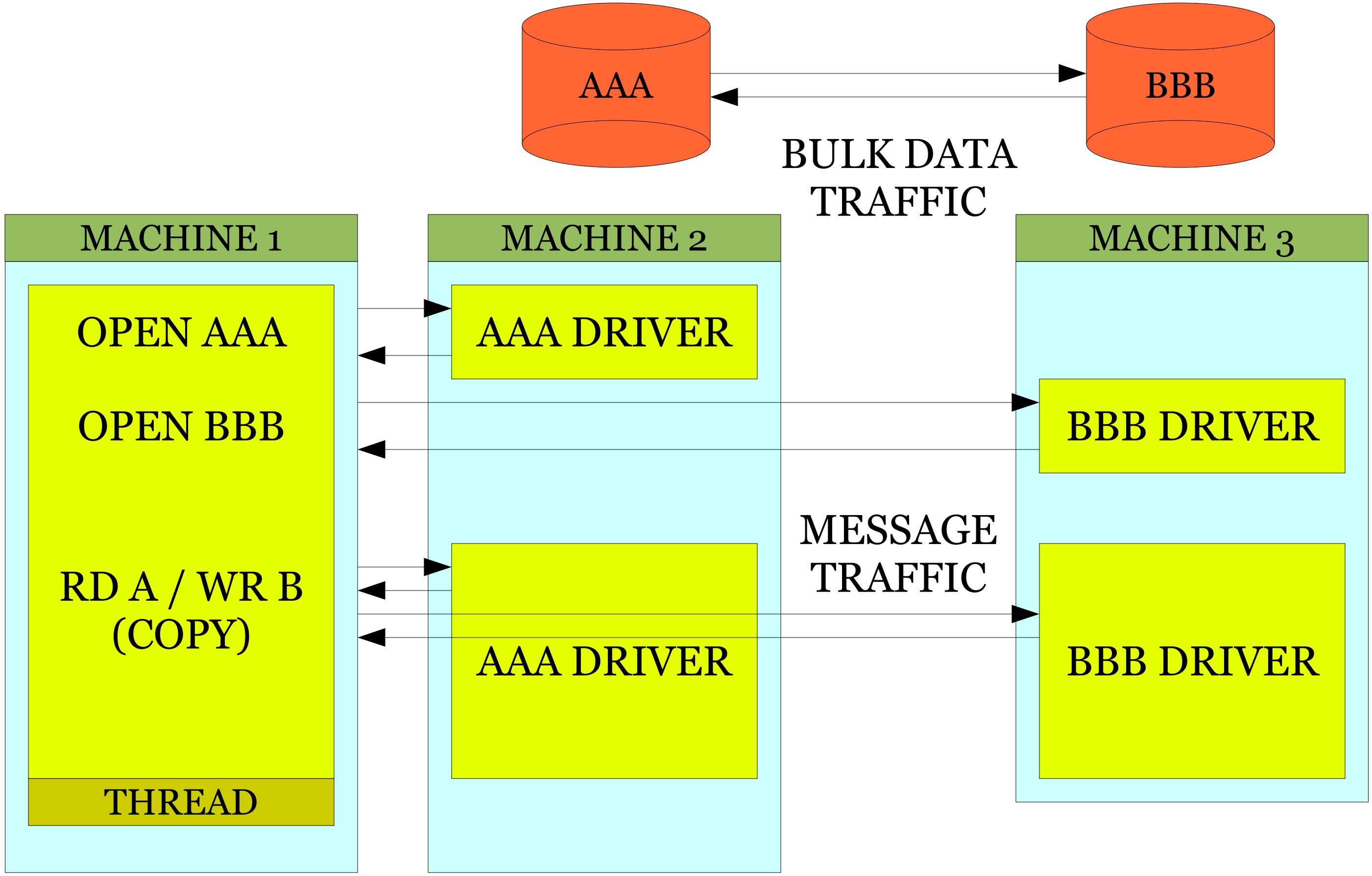
P1(S) P2(S) P4(S) P6(E) P7(S) P8(E) P9(E)

S E S E

- •CREATE A SINGLE (S) RECORD FOR P1-P4 BY OBTAINING A SHARED LOCK ON P1-P4
- •CREATE A SINGLE (E) RECORD FOR P8-P9
- •AGGREGATE OR THROW AWAY RECORDS TO REDUCE MEMORY USE
- •ADD SERIAL NUMBER TO VM PAGES TO ALLOW REVALIDATION OF CACHE STATUS
- •CAN MANAGE CACHE ON A BYTE RANGE BASIS RATHER THEN ON A PAGE BASIS

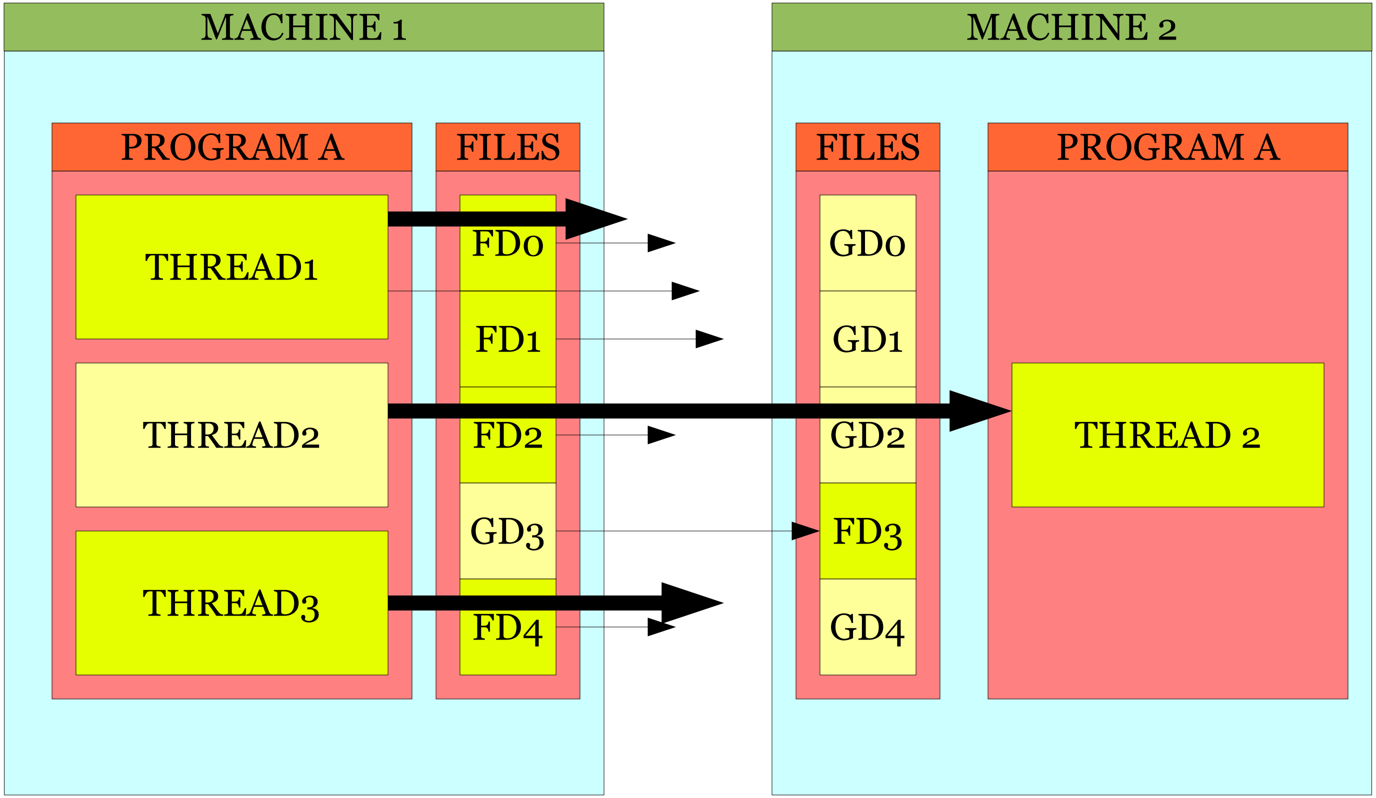
Global File Handles

- ACCESSIBLE FROM ANY HOST WITHIN THE CLUSTER
- POTENTIALLY ACCESSIBLE FROM OUTSIDE THE CLUSTER
- ALLOWS DEVICE DRIVERS TO DECIDE WHETHER TO MIGRATE OR NOT
- DATA ASSOCIATED WITH I/O SEPARTELY MANAGED VIA CACHE COHERENCY MODEI



Piecemeal Process Migration

- •CACHE COHERENCY MODEL ALLOWS ADDRESS-SPACE SHARING ACROSS MACHINES
- •DRIVERS FOR FILE DESCRIPTORS CAN MIGRATE ASYNCHRONOUSLY
- •SOME DRIVERS MIGHT STAY ON THE MACHINE HODING THE PHYSICAL STORAGE
- •TTYS AND PIPES CAN MIGRATE COMPLETELY OVER
- •SOCKETS ARE MORE COMPLEX, BUT MIGRATION IS STILL POSSIBLE


MACHINE 1
PROGRAM A
FILES
THREAD1
THREAD2
THREAD3
FD0
FD1
FD2
GD3
FD4
MACHINE 2
FILES
PROGRAM A
GD0
GD1
GD2
FD3
GD4
THREAD 2