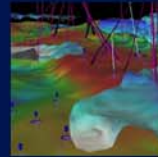




*Challenges Drive Innovation*



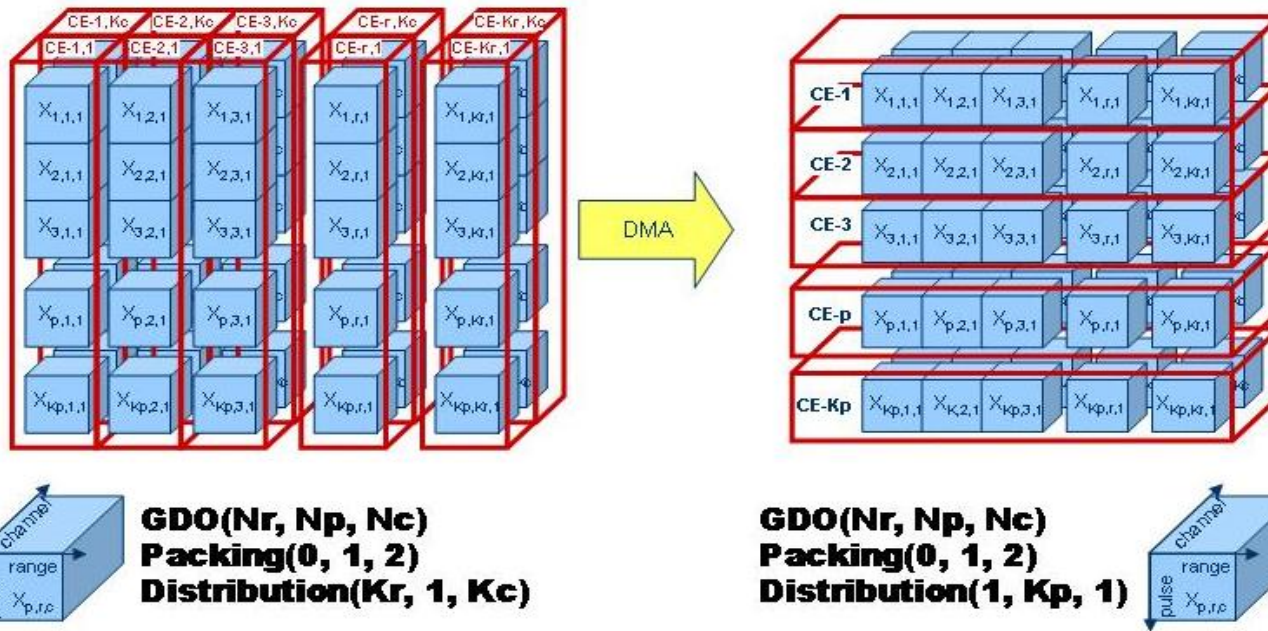
## Data Reorganization Interface (DRI): Past Perspective, Future Vision, New Results, Increased Availability

Kenneth Cain, [kcain@mc.com](mailto:kcain@mc.com)  
High Performance Embedded Computing Workshop  
September 21, 2006

- **Introduction**
- **DRI/PAS, MPI Direction and Vision**
- **Mercury Contribution to DRI Direction**
  - Evaluation software DRI/PAS over TCP/IP Linux clusters
- **DRI/PAS on Linux Clusters Architecture, Performance**
  - Highlights from separate DRI/PAS over InfiniBand package
- **Summary, Conclusions**

- **Data Reorganization Interface (DRI)**
  - <http://www.data-re.org>
- **Message Passing Interface (MPI)**
  - MPI Forum: <http://www.mpi-forum.org>
- **MVAPICH-gen2 (MPI over OpenIB),**
  - Network-Based Computing Laboratory, The Ohio State University: <http://nowlab.cse.ohio-state.edu/projects/mpi-iba>
- **Open Fabrics/OpenIB Software Stack**
  - Open Fabrics Alliance: <http://www.openfabrics.org>
- **User Direct Access Programming Library (uDAPL)**
  - Direct Access Transport (DAT) Collaborative: <http://www.datcollaborative.org>
- **Vector, Signal, and Image Processing Library (VS IPL, VS IPL++)**
  - VS IPL Forum: <http://www.vsipl.org>
  - High Performance Embedded Computing Software Initiative (HPEC-SI)  
<http://www.hpec-si.org>

# What is DRI?

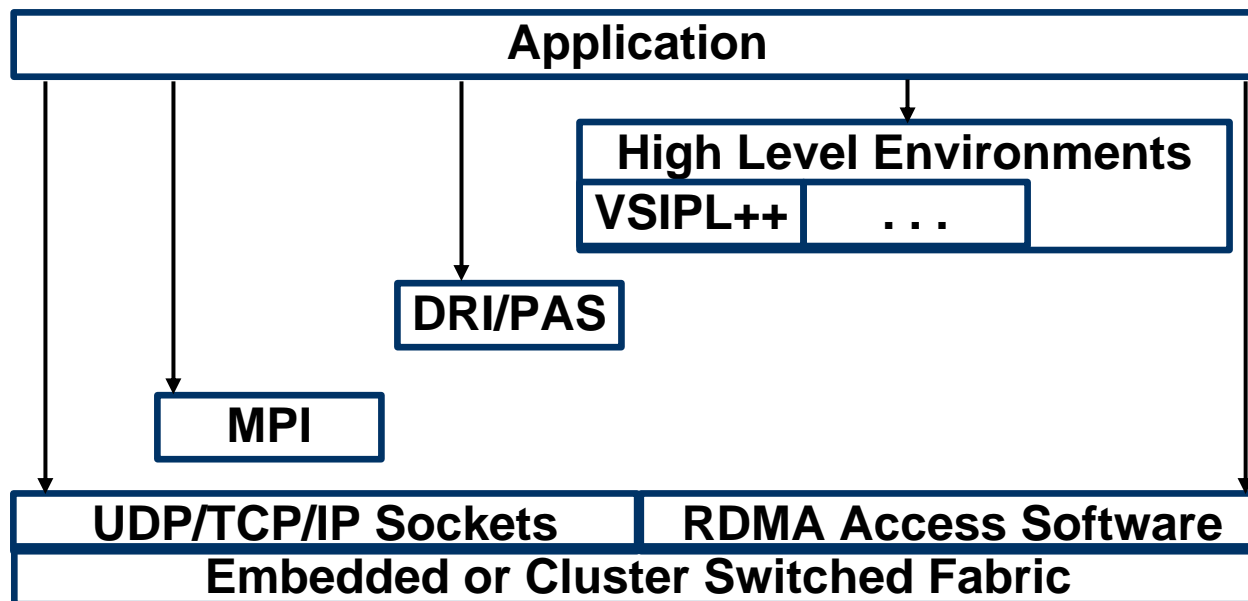


Additional Information  
<http://www.data-re.org>

- **Multi-dimensional data decomposition with overlap**
  - Calculates and internally maintains data element to processor map
- **Multi-point data movement between computational stages**
  - Performs all underlying point-to-point communications in one function call
  - Clique or pipeline
- **Multi-buffering for communication/computation overlap**
- **Reorganize memory packing order (e.g., matrix transpose)**

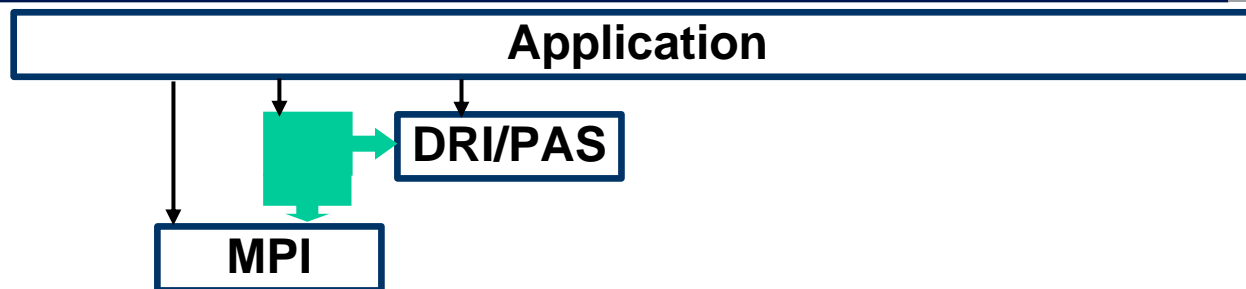
- **Mercury developed C language library/middleware**
- **Extended dataflow features all based on DRI foundation**
  - Planned and unplanned data transfers
    - (sources, destinations, memory buffers, transfer sizes, etc.)
  - Multi-buffered (channels) and single-buffered data transfers
  - Supports dynamically changing data sizes, data distributions
  - Fixed and flexible buffer sizes in multi-buffered channel
  - Aggregate channels (round-robin and next-available queuing policies)
  - Region of Interest (ROI)
  - Device integration (generic interface / protocol)
  - FPGA <-> PowerPC multi-buffered communication channels
  - Run-time host support

**Middleware Providing Standard DRI and Additional Capabilities Based on DRI**



- Application chooses interface entry point(s) considering all system constraints

**DRI has a unique value and role here**  
**DRI relates to most/all of these technologies**



## • Why?

- Leverage MPI investment: incrementally use DRI for dataflow-intensive parts
- Simpler data flows (1D, point-to-point): MPI good, DRI awkward
- Complex data flows (multi-D, multi-point, multi-buffer): DRI good, MPI tedious
- Parallel control flow: MPI good, DRI awkward

## • What is MPI and DRI/PAS integration: use MPI and DRI libraries

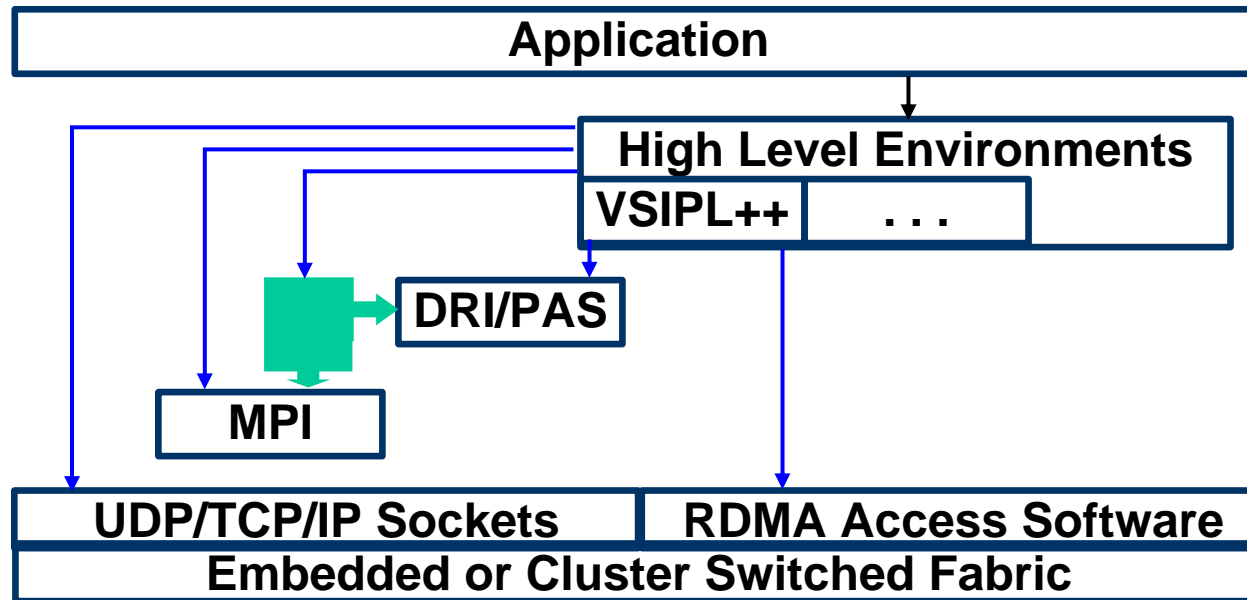
- DRI calls accept MPI\_Comm or PAS\_id (enables MPI+DRI standards-only)
- Translation routines MPI\_Comm→PAS\_id (enables MPI+DRI/PAS extensions)
- MPI can access application data stored in “special” DRI/PAS RDMA memory

## • Extension of the idea: single MPI + DRI implementation

- Optimize connection resources
- Common architecture (MPI process sets, RDMA memory registration)

## • MPI/DRI/PAS experiments conducted in these environments to date:

- x86: OpenMPI (over TCP/IP sockets)
- ppc64: MVAPICH-gen2 (over IB verbs), LAM and MPICH (over sockets),
- Mercury Powerstream 7000: Verari MPI/Pro (over RapidIO)



- High-level methods frequently offer DRI-like parameters to end user
- Less of an implementation/abstraction “leap” to use DRI

**DRI/PAS can be used “inside” higher level run-time library**  
**Complementary, integrated MPI and DRI/PAS middleware is a compelling choice**



- **DRI/PAS over TCP/IP sockets available free of charge**
  - First installation: GTRI Cluster (HPEC-SI / VSIPL++ Resource)
    - Dan Campbell, GTRI
  - Generally available (contact the author, [www](http://www) download soon)
  - Provides “two libraries” MPI/DRI/PAS integration interfaces
  - DRI/PAS on Linux Clusters Packaging
    - Red Hat RPM
    - Shared libraries provide some binary portability
  - Targets
    - x86, x86\_64
    - Fedora Core 4 Linux distribution
    - Other Linux distributions possible via binary portability
- **Higher performance software in development**
  - DRI/PAS on ppc64, InfiniBand transport

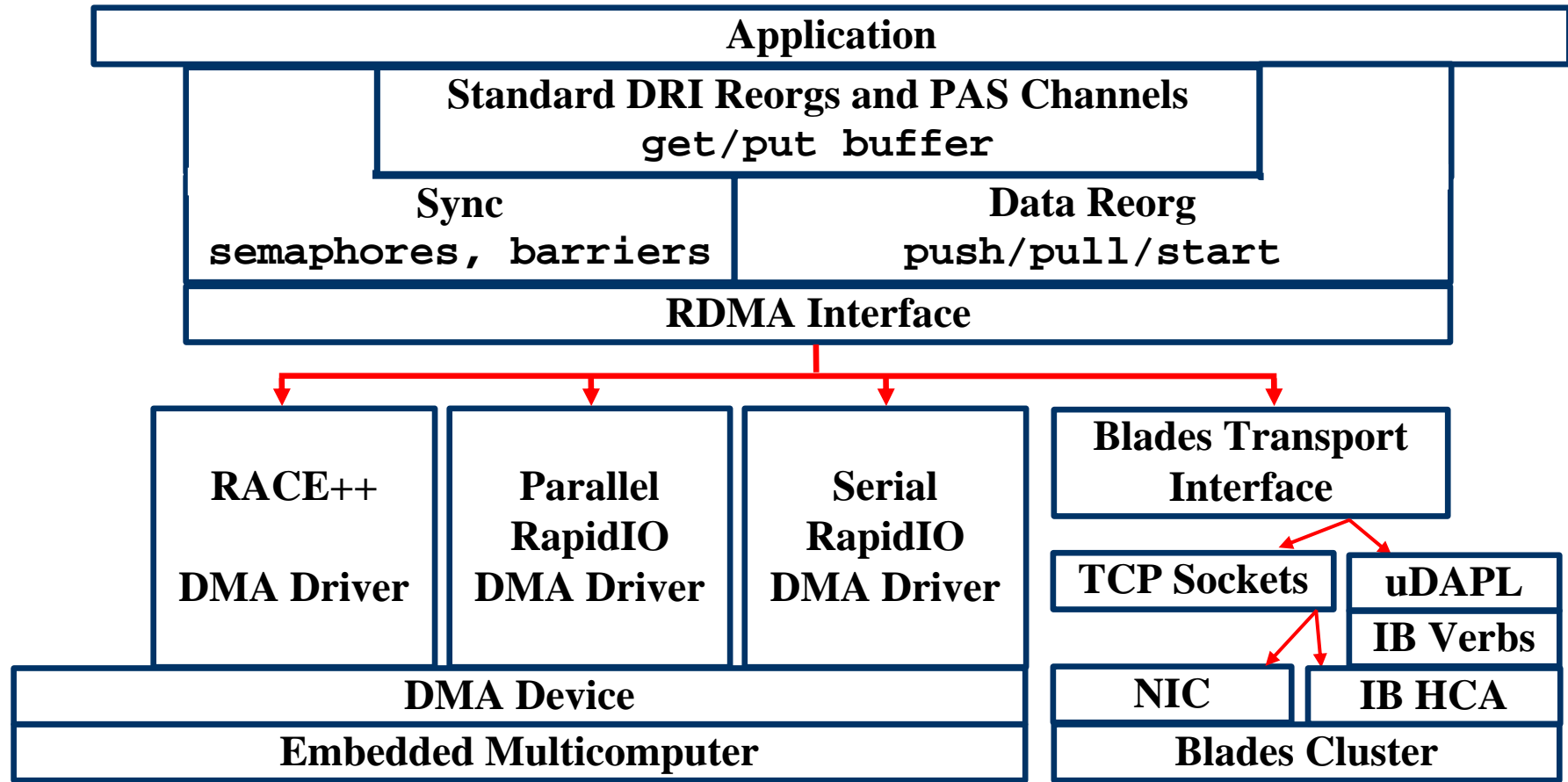
- **CodeSourcery (Sourcery VSIPL++ implementation):**

- Jules Bergmann
- Product information: <http://www.codesourcery.com/vsiplplusplus>
- DRI/PAS model is well suited for implementing Parallel VSIPL++
  - Same reasons as listed earlier for “DRI inside”
- Why DRI/PAS on TCP/IP Linux fits
  - Deployment on RACE++ and RapidIO multicomputers
  - Flexibility to easily switch between commodity cluster and embedded development
  - Relative ease of debugging in Linux cluster environment
- Porting success with DRI/PAS software using common set of DRI/PAS calls
  - GTRI Intel/AMD Linux cluster and Mercury MCOE RACE++ multicomputer

- **MITRE (DARPA HPCS program)**

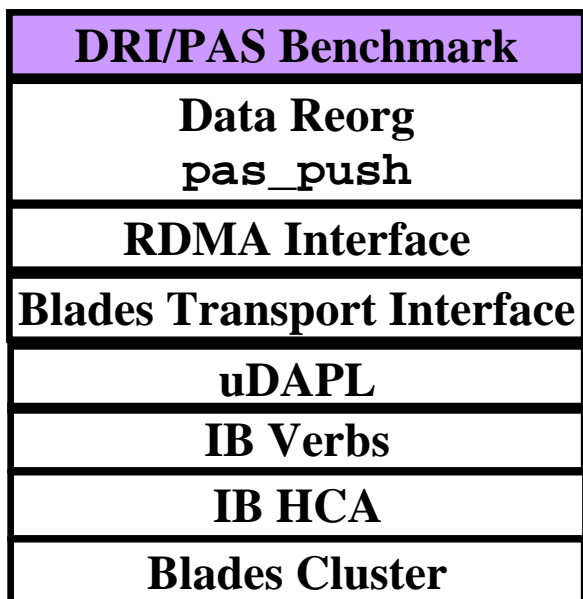
- Brian Sroka
- MITRE exploring productivity enhancement (using PAS) in SAR demonstration
- Offers development flexibility for planned port to Mercury PowerStream7000

# DRI/PAS Inner Loop Data Transfer Design



**Productivity:** Multi-dimensional, Multi-point, Multi-buffering  
**Performance:** OS bypass, RDMA, Low overhead design  
**Portability:** Enabled by modular software layering

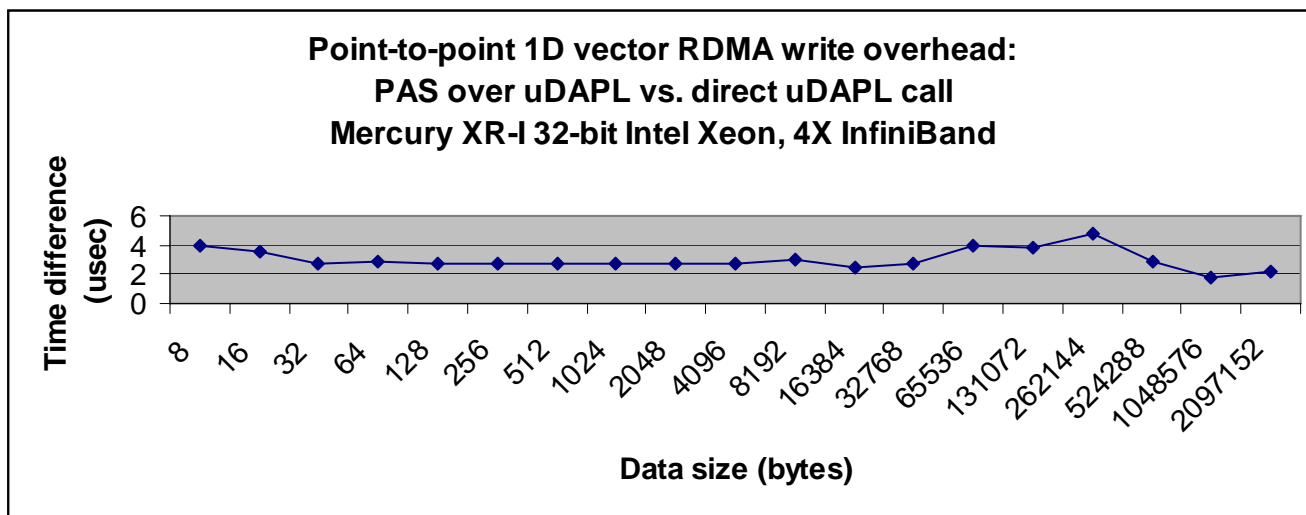
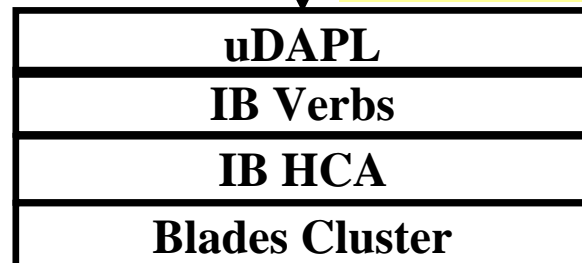
# DRI/PAS Over InfiniBand: Low Overhead



What does  
productivity  
cost\*?

## Non-DRI/PAS Benchmark

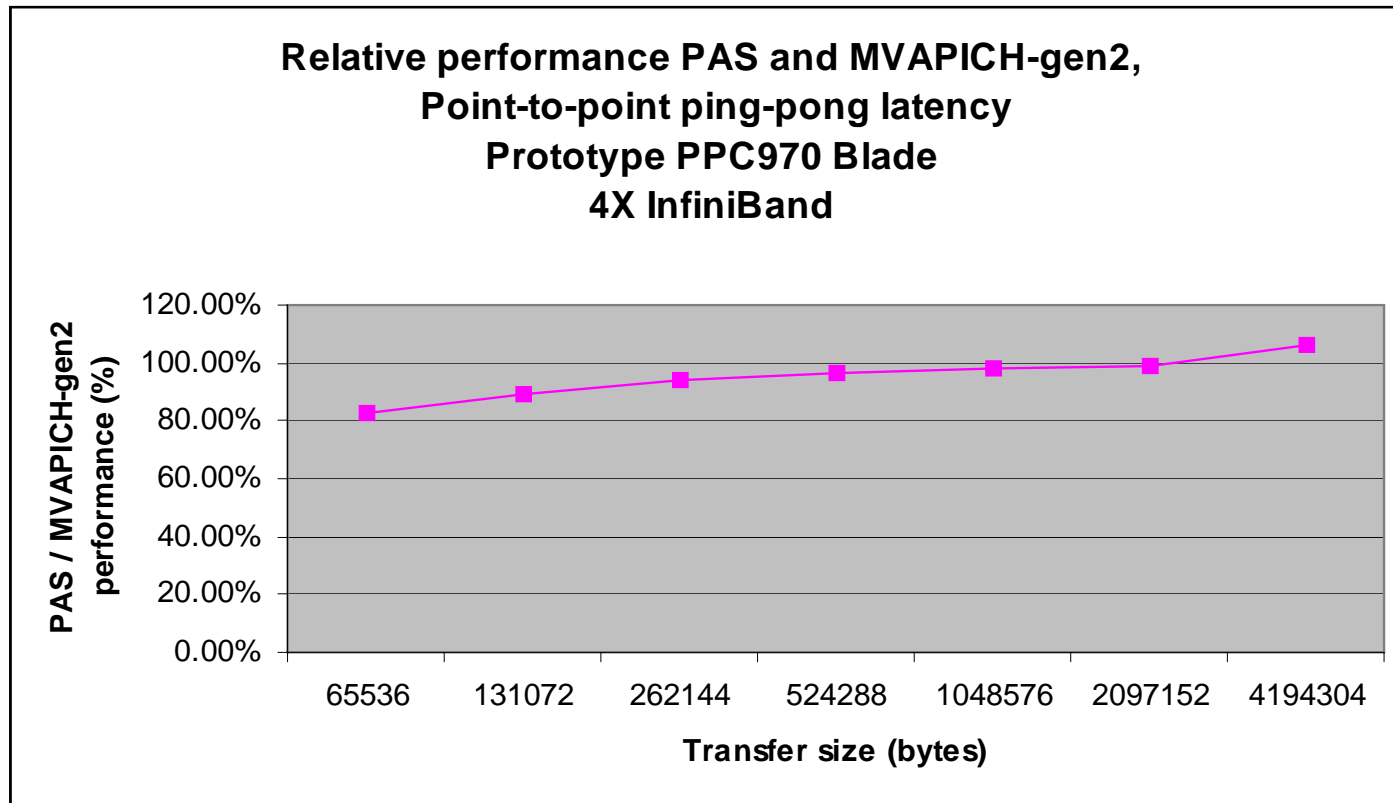
Call uDAPL  
RDMA  
Directly  
(NO DRI/PAS)



\* “dynamic” transfer overhead shown here can be optimized by using planned transfers

**Latency Issue for  
Small Transfers**

**Good Performance for  
Large Transfers**



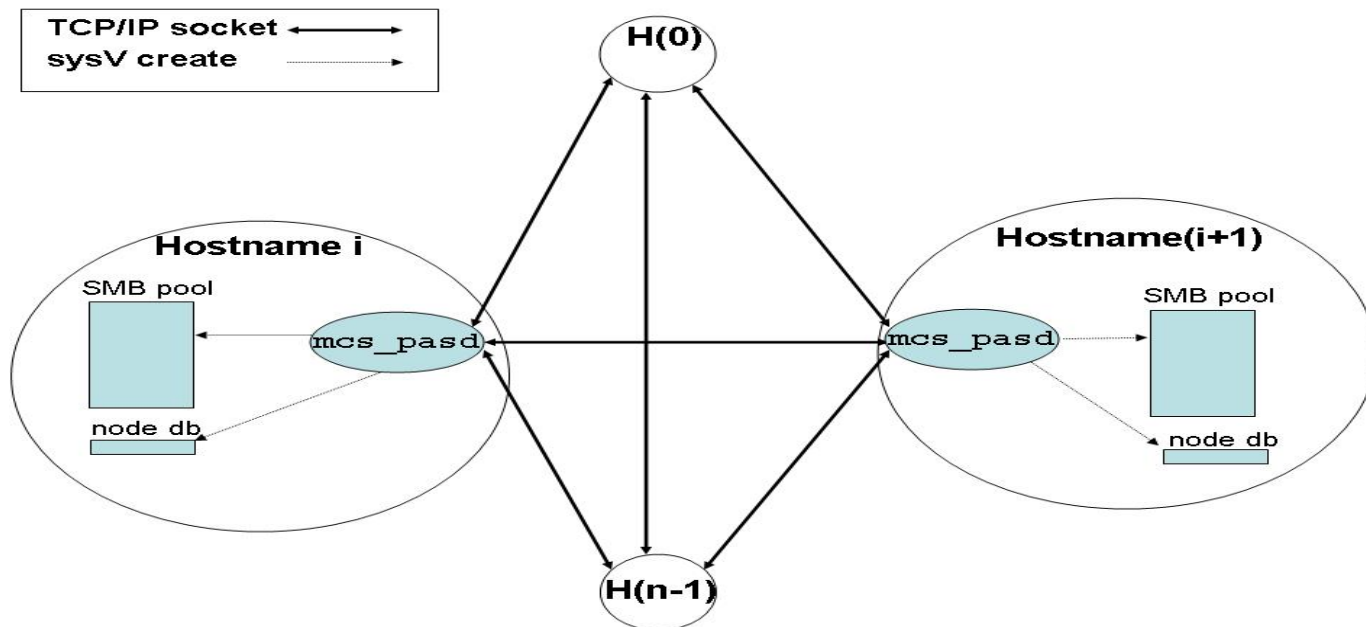
- **Transport uses uDAPL (MVAPICH-gen2 uses IB Verbs)**
- **Using uDAPL DAT\_EVD\_HANDLE for synchronization**
  - NetPIPE over uDAPL (NPuDAPL) benchmark provides insight
    - <http://www.scl.ameslab.gov/netpipe/>
  - PAS over uDAPL/EVD roughly matches NPUuDAPL/EVD performance
  - DAT\_CNO\_HANDLE is faster than DAT\_EVD\_HANDLE
  - 43% faster latency measured NPUuDAPL/EVD → NPUuDAPL/CNO
  - Optimization (possible): change PAS transport to use uDAPL CNO
- **DRI/PAS Multi-threaded transport**
  - Issue 1: application queues RDMA requests to a “transport” thread
  - Optimization (possible): use a single-threaded transport
  - Issue 2: transport thread blocks until work is queued
  - Optimization (10% faster latency measured ): use non-blocking (CPU yield)

- **A DRI direction was presented**
  - A freely available reference implementation is needed
  - DRI inter-operation or integration with MPI
  - DRI model also being applied within multicore processor (Cell BE)
- **Actions were taken to contribute to this direction**
  - Free of charge DRI/PAS on TCP/IP Linux clusters evaluation
  - Offering new interfaces to allow applications to mix and match MPI and DRI/PAS usage
- **Further community innovation based on the DRI development model**

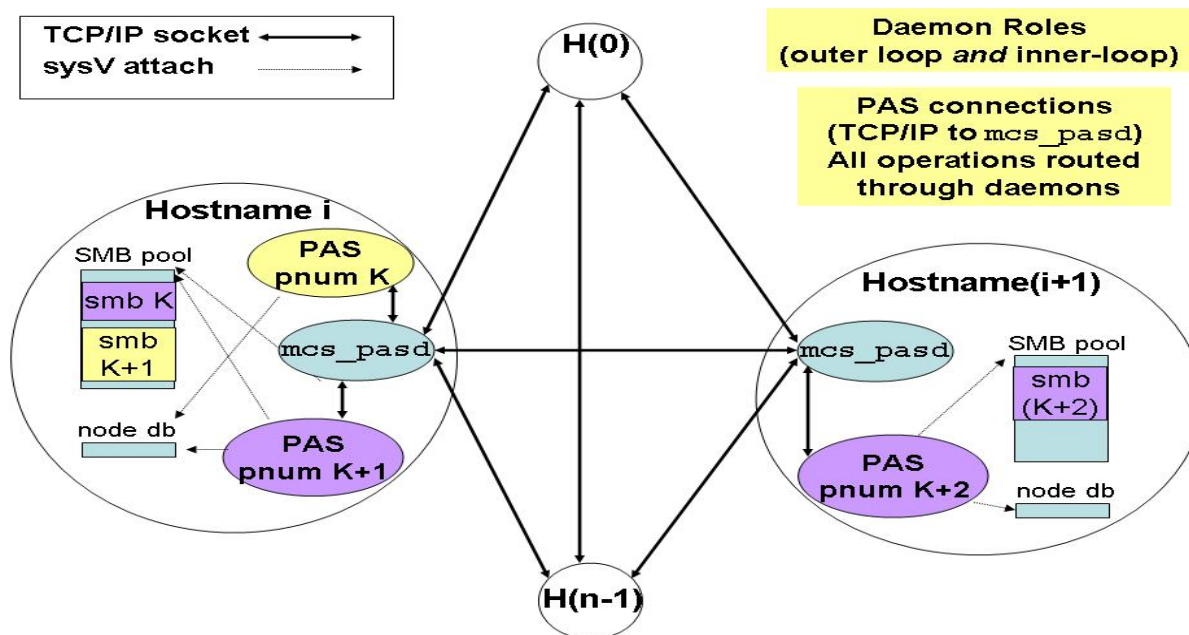
# Backups



# Architecture: DRI/PAS Node Daemons



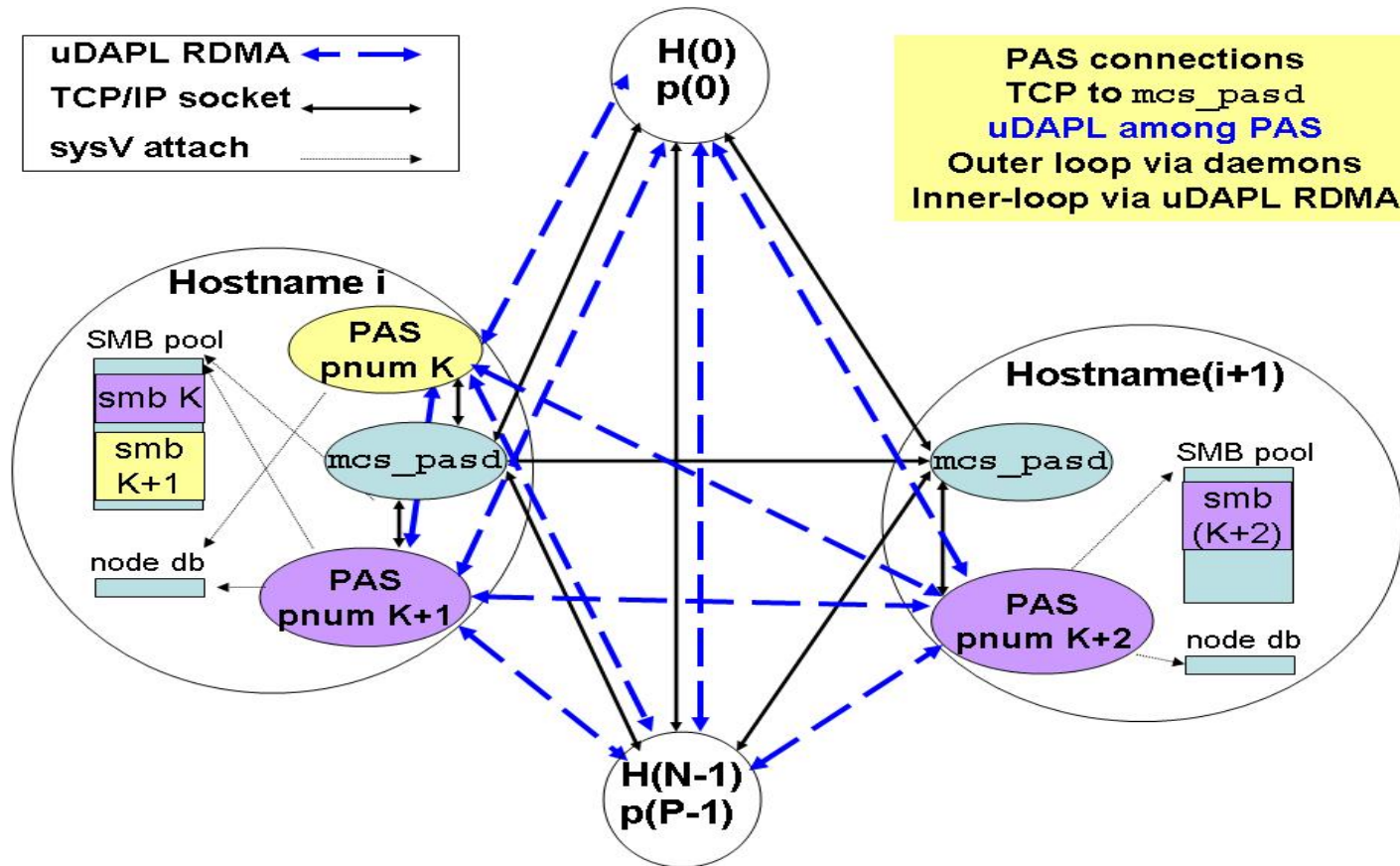
- Launched one per hostname, before any DRI/PAS processes
- TCP/IP socket connections among all daemons
- Multicomputer services offered to DRI/PAS “client” processes
  - Named shared memory buffers (SMBs)
  - Multicomputer-scoped semaphores (named and unnamed)
  - Cluster information (node database shared-memory segment)
  - RDMA (emulation over TCP/IP socket connections)



## • DRI/PAS processes:

- Connect to local daemon process via TCP socket
- Attach to local SMB pool, node db created earlier by daemon
- Allocate an SMB to use for DRI/PAS “parallel buffer” allocations
- Use daemon-provided RDMA, sync to rendez-vous (indirect)
- Use daemon-provided RDMA, sync for DRI/PAS transfers (indirect)




# Architecture: TCP/IP and uDAPL (InfiniBand)



## • DRI/PAS processes:

- Establish all connectivity previously mentioned except:
- Do **NOT** use daemon-provided RDMA, sync for DRI/PAS transfers
- Use uDAPL/RDMA connections among DRI/PAS processes (direct)

# Ideal Use of DRI/PAS on Linux/TCP

Embedded Deployed Limited Development	Embedded Switched Fabrics (e.g., RapidIO)	
Blade chassis Rack-mounted Development Deployment	Switched Fabrics (InfiniBand, Gigabit Ethernet, etc.)	
Development	Ordinary Network Infrastructure (e.g., Ethernet)	

When you will  
deploy here, high  
performance  
essential

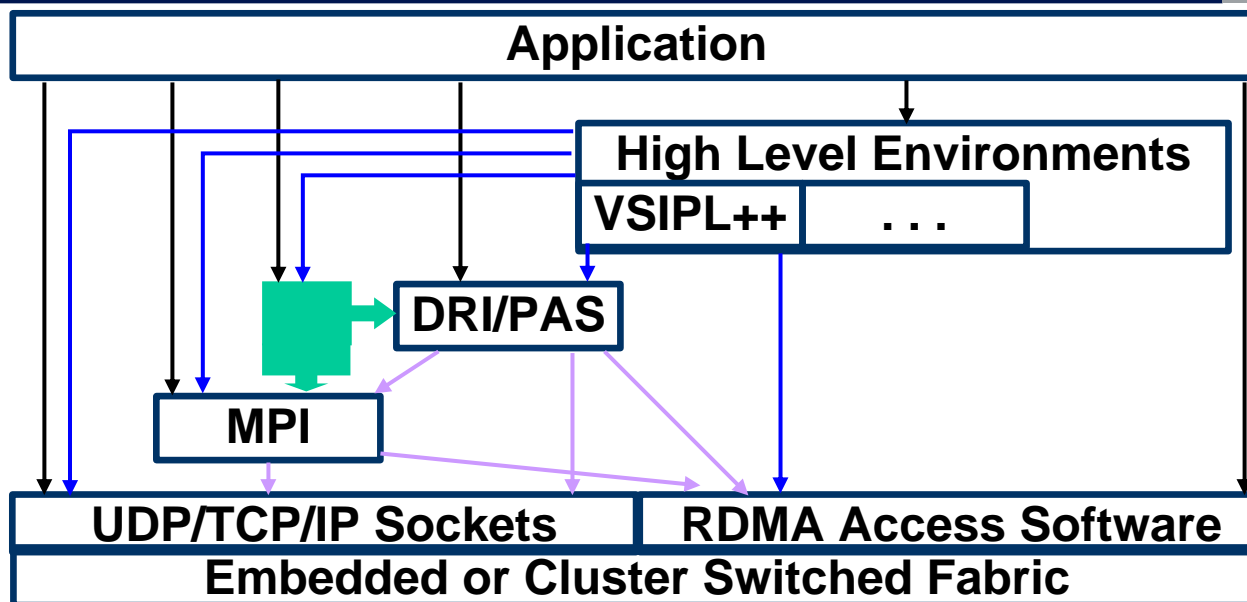
And when you  
want to develop  
(in part) here,  
performance not  
essential

- It is a reference (not commercial) software package
- Best fit is when developing for functional correctness

- **HPEC-SI: Contribute to ongoing Parallel VSIPL++ spec:**
  - DRI/PAS software is a working demonstration of several important features under consideration for Parallel VSIPL++
  - Planned and unplanned data flow APIs
  - Pipeline and clique data flows
  - Multi-buffered communication interface
- **DARPA HPCS**
  - MITRE exploring productivity enhancement (using PAS) in SAR demonstration
  - Offers development flexibility for planned port to Mercury PowerStream7000
- **Broader HPEC, HPC?**
  - Seeking feedback from the community

- **Idea: write application to DRI, PAS, and MPI interfaces**
  - DRI and related software has something to add to just MPI
  - Each interface has particular strengths
  - Interesting use cases (choose most suitable for requirements):
    - MPI+DRI (e.g., standard API requirement)
    - MPI+DRI+PAS (standard APIs, use extensions where necessary)
    - MPI+PAS
- **Current integration technique in DRI/PAS software**
  - Get DRI/PAS software, get MPI separately
  - Compile with preprocessor symbol to enable MPI integration
  - Link with both DRI/PAS and MPI libraries
  - DRI functions will accept MPI Communicators (not PAS psets)
    - DRI\_Reorg\_create() and DRI\_Reorg\_get\_group()
  - DRI\_Init() opens PAS inside, so PAS calls are not required
  - Provides MPI Communicator -> PAS pset conversion functions

# Direction and Vision (All Together)



## Legend

- Application implementation choice
- High level middleware implementation choice
- Intermediate level middleware implementation choice

- Application chooses interface entry point(s) considering all system constraints
- Complementary, integrated MPI and DRI/PAS middleware is a compelling choice
- DRI/PAS can be used “inside” higher level run-time library
- DRI model has been successfully applied in heterogeneous multicore architecture (Cell Broadband Engine)